8-92

ZiLOG

MICROCONTROLLERS

1991

ZiLOG

MICROCONTROLLERS

1991

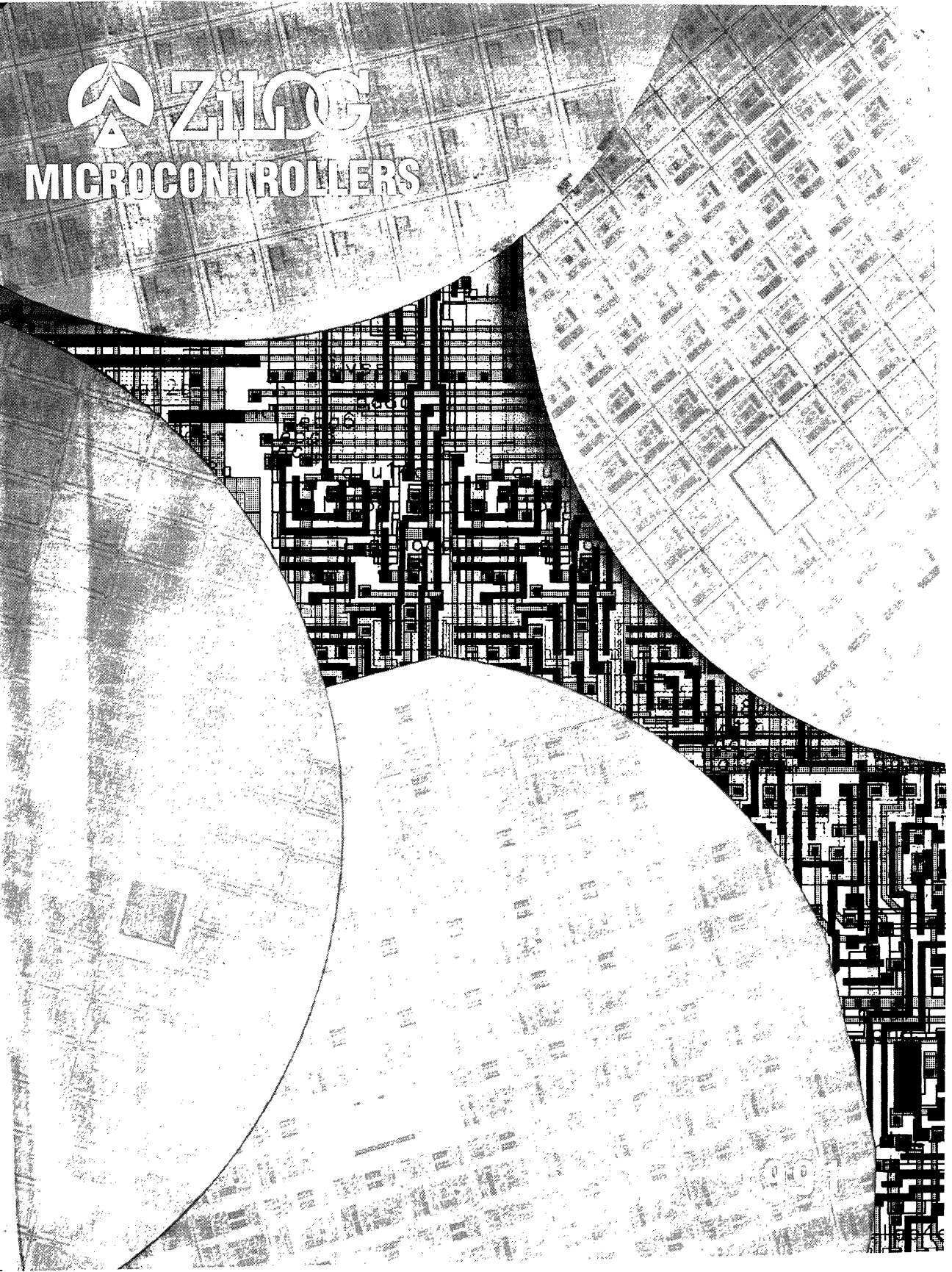

ZiLOG
MICROCONTROLLERS

Table of Contents

Z8® Product Family

Page

Table of Contents

Z8 Application Notes and Technical Articles

Super8 Application Notes and Technical Articles

Additional Information

MICROCONTROLLERS

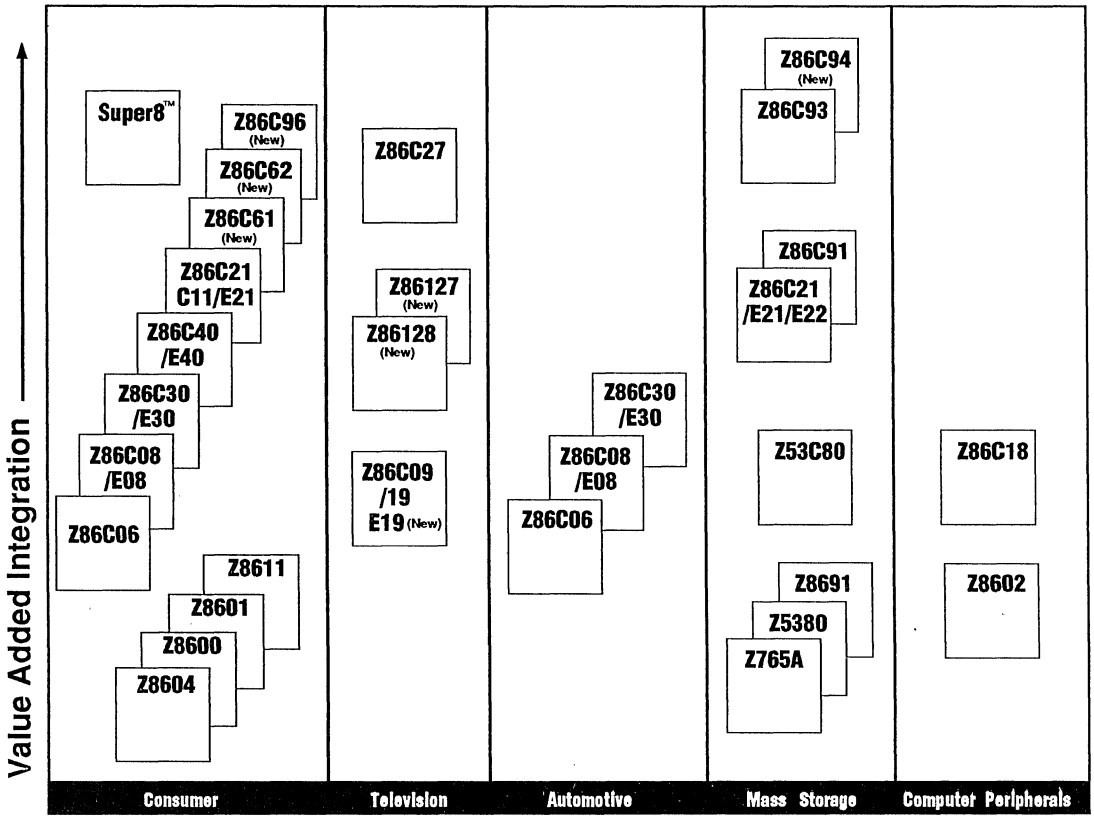
ZILOG Z8® FAMILY

AN INDUSTRY STANDARD 8-BIT SINGLE-CHIP ARCHITECTURE WITH VALUE ADDED INTEGRATION

The Zilog Z8 Microcontroller Family has long been a recognized leader in single-chip architecture. Today, Zilog continues to expand that leadership with an ever growing array of new high integration Z8-based products including the Close Caption chip, the Z86128 and the Z86C94, a high integration controller with DSP. The expanded family offers Value Added Integration for a wide variety of applications such as consumer products, television control, automotive, mass storage and computer peripherals. This edition of the Z8 Microcontroller Data Book describes this entire family of devices including ROM, ROMless and OTP (One-Time-Programmable) versions, as well as the development support tools available from Zilog.

All of Zilog's high integration family are created using Zilog's Superintegration™ technology. Zilog pioneered the Superintegration process as the science of creating highly efficient, powerful ICs with fully characterized cores and cells. Zilog's core and cell library is one of the largest proprietary libraries in the semiconductor industry. Zilog uses Superintegration technology to produce a wide variety of ASSPs (Application Specific Standard Products) that offer great cost/performance benefits and the ability to customize in software rather than hardware with minimal cost and the highest possible quality.

Product Application Portfolio



Market Segments

Z8 REFERENCE CHART MICROCONTROLLERS

Product	Pin Count	ROM (Kbyte)	I/O	Interrupts	UARTS	Comparators	CTCs	WDT	Package Type	Speed	Temp	Low Noise
CMOS												
Z86C00 MCU	28	2	22	3	-	-	2	-	DIP	8,12	S,E	-
Z86C06 MCU	18	1	14	5	-	2	2	X	DIP,SOIC†1	4*,8,12	S,E	X
Z86C08 MCU	18	2	14	5	-	2	2	X	DIP,SOIC†1	4*,8,12	S,E	X
Z86E08 MCU	18	2 (OTP)	14	5	-	2	2	X	DIP	4*,8,12	S	X
Z86C09/19	18	2/4	14	5	-	2	2	X	DIP,SOIC†1	4*,8,12	S,E	-
Z86C10 MCU	28	4	22	3	-	-	2	-	DIP	8,12	S,E	-
Z86C11 MCU	40,44	4	32	6	1	-	2	-	QFP,DIP,PLCC	12,16	S,E	-
Z86C12 ICE	84	32	16	6	1	-	2	-	PGA	16	S	-
Z86E19 MCU†1	18	4 (OTP)	14	5	-	2	2	X	DIP	12	S	X
Z86C20 MCU	28	8	22	3	-	-	2	-	DIP	12	S	X
Z86C21 MCU	40,44	8	32	6	1	-	2	-	QFP,DIP,PLCC	4*,12,16	S,E	X
Z86E21/E22* MCU	40,44	8 (OTP)	32	6	1	-	2	-	QFP,DIP,PLCC	4*,12,16	S	X
Z86C91 MCU	40,44	-	16	6	1	-	2	-	QFP,DIP,PLCC	12,16,20	S,E	-
Z86C30 MCU	28	4	24	6	-	2	2	X	DIP	4*,8,12	S,E	X
Z86E30 MCU	28	4 (OTP)	24	6	-	2	2	X	DIP	12	S	X
Z86C40 MCU	40,44	4	32	6	-	2	2	X	QFP,DIP,PLCC	4*,8,12	S,E	X
Z86E40 MCU	40,44	4 (OTP)	32	6	-	2	2	X	QFP,DIP,PLCC	12	S	X
Z86C61 MCU†1	40,44	16	32	6	1	-	2	-	QFP,DIP,PLCC	16	S,E,	-
Z86C62 MCU†1	64,68	16	52	6	1	-	2	-	DIP,PLCC	16	S,E	-
Z86C89**/90 MCU	40,44	-	16	6	-	2	2	X	QFP,DIP,PLCC	4*,8,12	S,E	X
Z86C27 TV Controller	64	8	43	6	-	-	2	X	DIP	4	S	X
Z86C96 MCU†1	64,68	-	44	6	1	-	2	-	DIP,PLCC	20	S,E	-
Z86C97 TV Controller	64	-	16	6	-	-	2	X	DIP	4	S	X
Z86C93 MCU	44	-	16	6	1	-	3	-	QFP,PLCC	20	S,E	-
Z86C94 DSP MCU†1	80	-	24	6	1	-	3	-	QFP	20	S,E	-
Z88C00 Super8†1	48,68	-	24	27	1	-	2	-	DIP,PLCC	25	S	-

* Low EMI version
** RC Oscillator Option
† Estimate Release Date
1 Q3/91

Temperature Range: S = Standard 0°C to +70°C
E = Extended -40°C to +105°C
M = Military -55°C to +125°C

Product	Pin Count	ROM (Kbyte)	I/O	Interrupts	UARTS	Comparators	CTCs	WDT	Package Type	Speed	Temp	Low Noise
NMOS												
Z8600 MCU	28	2	22	3	-	-	2	-	DIP	8	S,E	-
Z8601/02 MCU	40,44	2	32	6	1	-	2	-	DIP,PLCC	4*,8	S,E	X*
Z8603 PROTOPACK	40	2	32	6	1	-	2	-	DIP	8,12	S	-
Z8604 MCU	18	1	14	5	-	-	2	X	DIP	8	S	-
Z8610 MCU	28	4	22	3	-	-	2	-	DIP	8,12	S,E	-
Z8611 MCU†	40,44	4	32	6	1	-	2	-	DIP,PLCC	8,12.5	S,E,M	-
Z8681 MCU†	40,44	-	16	6	1	-	2	-	DIP,PLCC	8,12	S,E,M	-
Z8691 MCU	40,44	-	16	6	1	-	2	-	DIP,PLCC	8,12	S,E	-
Z8612 ICE	64,68	4	32	6	1	-	2	-	PLCC,Ceramic	12	S	-
Z8613 PROTOPACK	40	4	32	6	1	-	2	-	DIP	8,12	S	-
Z8800 SUPER8	48,68	-	24	27	1	-	2	-	DIP,PLCC	20	S	-
Z8801 SUPER8	44	-	17	27	1	-	2	-	PLCC	20	S	-
Z8820 SUPER8	48,68	8	40	27	1	-	2	-	DIP,PLCC	20	S	-
Z8821 SUPER8	44	8	33	27	1	-	2	-	PLCC	20	S	-
Z8884 SUPER8 ICE	84	16	40	27	1	-	2	-	PGA	20	S	-
Z5380 SCSI	40,44	-	-	-	-	-	-	-	-	-	S	
Z765A FDC	40,44	-	-	-	-	-	-	-	-	-	S	-

† Available in Military version

* Z8602 Low Noise, 4 MHz, DIP only.

Temperature Range:
S = Standard 0°C to +70°C
E = Extended -40°C to +105°C
M = Military -55°C to +125°C

CMOS PRODUCTS

Z86C00/C10/C20 CMOS Z8® MCU

FEATURES

- Complete microcomputer, 2K (86C00), 4K (86C10), or 8K (86C20) bytes of ROM, 124 bytes of RAM (256 bytes - Z86C20), and 22 I/O lines.
- 144-byte register file, including 124 (238 - Z86C20) general-purpose registers, four I/O port registers, and 14 status and control registers.
- Average instruction execution time of 1.5 us, maximum of 2.8 us.
- Vectored, priority interrupts for I/O and counter/timers.
- Two programmable 8–bit counter/timers, each with a 6–bit programmable prescaler.
- Register Pointer so that short, fast instructions can access any of nine working–register groups in 1.0 us.
- On–chip oscillator which accepts crystal, external clock drive, LC, ceramic resonator.
- Standby modes –– Halt and Stop.
- Single +5V power supply –– all pins TTL–compatible.
- 8 and 12 MHz
- CMOS process.

GENERAL DESCRIPTION

Z86C10/C20 microcomputer (Figures 1 and 2) introduces a new level of sophistication to single-chip architecture. Compared to earlier single-chip microcomputers, the **Z86C10/C20** offers faster execution; more efficient use of memory; more sophisticated interrupt, input/output and bit-manipulation capabilities; and easier system expansion.

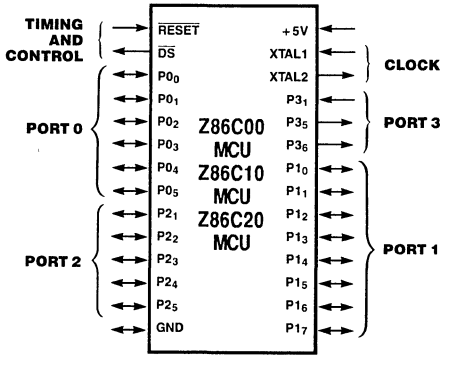


Figure 1. Pin Functions

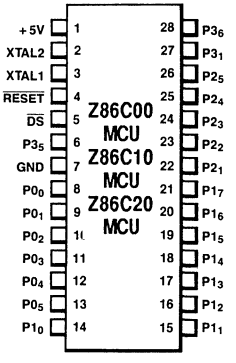


Figure 2. Pin Assignments

PIN DESCRIPTIONS

$\overline{DS}$. *Data Strobe* (output, active Low). Data Strobe is activated once for each memory transfer.

$P0_0$-$P0_5$, $P1_0$-$P1_7$, $P2_1$-$P2_5$, $P3_1$, $P3_5$, $P3_6$. *I/O Port lines* (bidirectional, TTL-compatible). These 22 I/O lines are grouped in four ports that can be configured under program control for I/O.

$\overline{RESET}$. *Reset* (input, active Low). $\overline{RESET}$ initializes the MCU. When $\overline{RESET}$ is deactivated, program execution begins from internal program location $000C_H$.

XTAL1, XTAL2. *Crystal 1, Crystal 2* (time-base input and output). These pins connect a parallel-resonant crystal to the on-chip clock oscillator and buffer.

ARCHITECTURE

The MCU's architecture is characterized by a flexible I/O scheme, an efficient register and address space structure, and a number of ancillary features that are helpful in many applications. (Figure 3).

Microcomputer applications demand powerful I/O capabilities. The MCU fulfills this with 22 pins dedicated to input and output. These lines are grouped in four ports and are configurable under software control to provide timing, status signals, and parallel I/O.

Two basic internal address spaces are available to support this wide range of configurations: program memory and the register file. The 144-byte random-access register file is composed of 124 general-purpose registers, four I/O port registers, and 14 control and status registers.

To unburden the program from coping with real-time problems such as counting/timing, two counter/timers with a large number of user-selectable modes are offered on-chip.

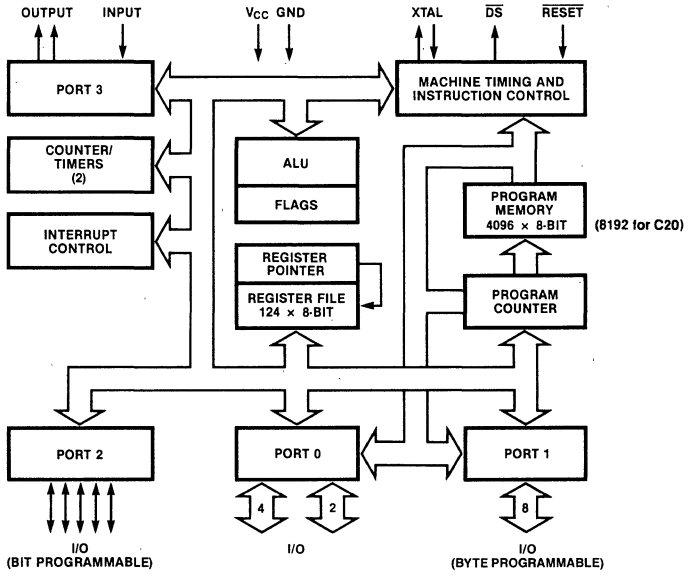


Figure 3. Functional Block Diagram

STANDBY MODE

The Z86C00/C10/C20's standby modes are:

- Stop
- Halt

The Stop instruction stops the internal clock and clock oscillation; the Halt instruction stops the internal clock but not clock oscillation.

A reset input releases the standby mode.

To complete an instruction prior to entering standby mode, use the instructions:

```
LD TMR, #00
NOP
STOP or HALT
```

ADDRESS SPACES

Program Memory. The 16-bit program counter addresses 4K **or 8K** bytes of program memory space as shown in Figure 4.

The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain three 16-bit vectors that correspond to the three available interrupts.

Register File. The 144-byte register file includes four I/O port registers (R_0-R_3), 124 general-purpose registers (R_4-R_{127}) and 15 control and status registers (R_{241}-R_{255}). These registers are assigned the address locations shown in Figure 5.

Instructions can access registers directly or indirectly with an 8-bit address field. The MCU also allows short 4-bit register addressing using the Register Pointer (one of the control registers). In the 4-bit mode, the register file is divided into nine working-register groups, each occupying 16 contiguous locations (Figure 6). The Register Pointer addresses the starting location of the active working-register group.

Stacks. An 8-bit Stack Pointer (R_{255}) is used for the internal stack that resides within the 124 general-purpose registers (R_4-R_{127}).

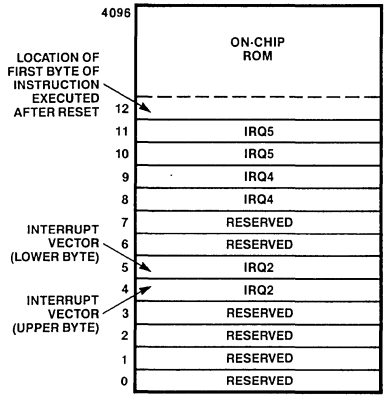


Figure 4. Program Memory Map

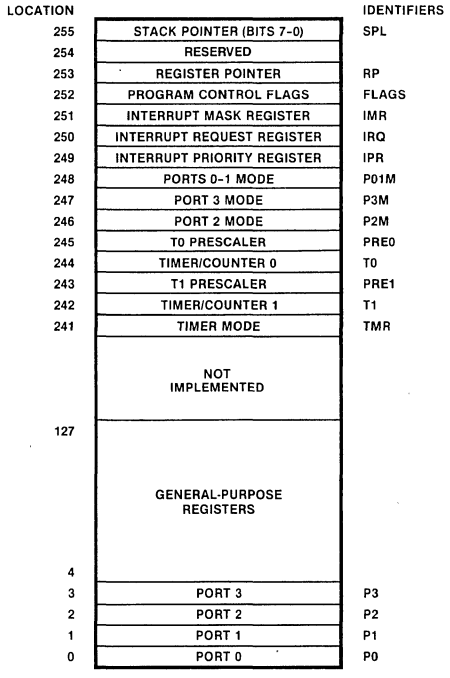


Figure 5. Register File

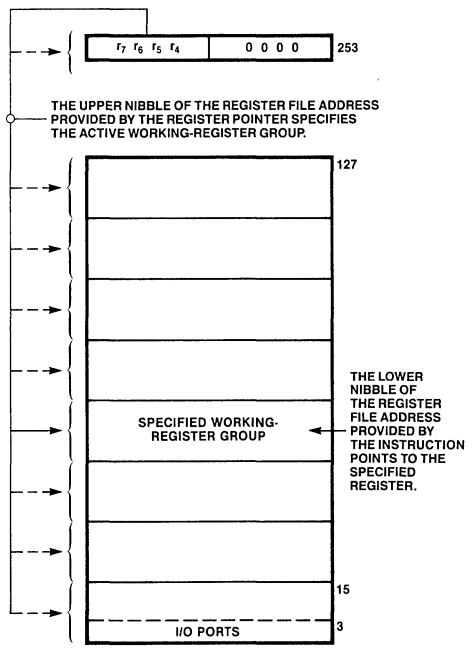


Figure 6. Register Pointer

COUNTER/TIMERS

The MCU contains two 8-bit programmable counter/timers (T_0 and T_1), each driven by its own 6-bit programmable prescaler. The T_1 prescaler can be driven by internal or external clock sources; however, the T_0 prescaler is driven by the internal clock only.

The 6-bit prescalers can divide the input frequency of the clock source by any number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request—IRQ_4 (T_0) or IRQ_5 (T_1)—is generated.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. The counters can also be programmed to stop upon reaching zero (single-pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode). The counters, but not the prescalers, can be read any time without disturbing their value or count mode.

The clock source for T_1 is user-definable and can be the internal microprocessor clock divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input as an external clock , a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. The counter/timers can be programmably cascaded by connecting the T_0 output to the input of T_1. Port 3 line $P3_6$ also serves as a timer output (T_{OUT}) through which T_0, T_1 or the internal clock can be output.

I/O PORTS

The MCU has 22 lines dedicated to input and output grouped in four ports. Under software control, the ports can be programmed to provide address outputs, timing, status signals, and parallel I/O. All ports have active pull-ups and pull-downs compatible with TTL loads.

Port 0 can be programmed as an I/O port.

Port 1 can be programmed as a byte I/O port.

Port 2 can be programmed independently as input or output and is always available for I/O operations. In addition, Port 2 can be configured to provide open-drain outputs.

Port 3 can be configured as I/O or control lines. $P3_1$ is a general purpose input or can be used for an external interrupt request signal (IRQ_2). $P3_5$ and $P3_6$ are general purpose outputs. $P3_6$ is also used for timer input (T_{IN}) and output (T_{OUT}) signals.

INTERRUPTS

The MCU allows three different interrupts from three sources, the Port 3 line $P3_1$ and the two counter/timers. These interrupts are both maskable and prioritized. The Interrupt Mask register globally or individually enables or disables the three interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All interrupts are vectored. When an interrupt request is granted, an interrupt machine cycle is entered. This disables all subsequent interrupts, saves the Program Counter and status flags, and branches to the program memory vector locations reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

Polled interrupt systems are also supported. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

CLOCK

The on-chip oscillator has a high-gain parallel-resonant amplifier for connection to a crystal or to any suitable external clock source (XTAL1 = Input, XTAL2 = Output).

Crystal source is connected across XTAL1 and XTAL2 using the recommended capacitors (C1 $\leqslant$ 15 pf) from each pin to ground. The specifications are as follows:

- AT cut, parallel resonant
- **Fundamental type, 16 MHz maximum.**
- Series resistance, Rs $\leqslant$ 100 n

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

dst	Destination location or contents
src	Source location or contents
cc	Condition code (see list)
@	Indirect address prefix
SP	Stack pointer (control registers 254-255)
PC	Program counter
FLAGS	Flag register (control register 252)
RP	Register pointer (control register 253)
IMR	Interrupt mask register (control register 251)

Assignment of a value is indicated by the symbol "←". For example,

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

dst (7)

refers to bit 7 of the destination operand.

Flags. Control Register R252 contains the following six flags:

C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

Affected flags are indicated by:

0	Cleared to zero
1	Set to one
*****	Set or cleared according to operation
—	Unaffected
X	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000		Always true	—
0111	C	Carry	C = 1
1111	NC	No carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	[Z OR (S XOR V)] = 0
0010	LE	Less than or equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned greater than or equal	C = 0
0111	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1
0000		Never true	—

INSTRUCTION FORMATS

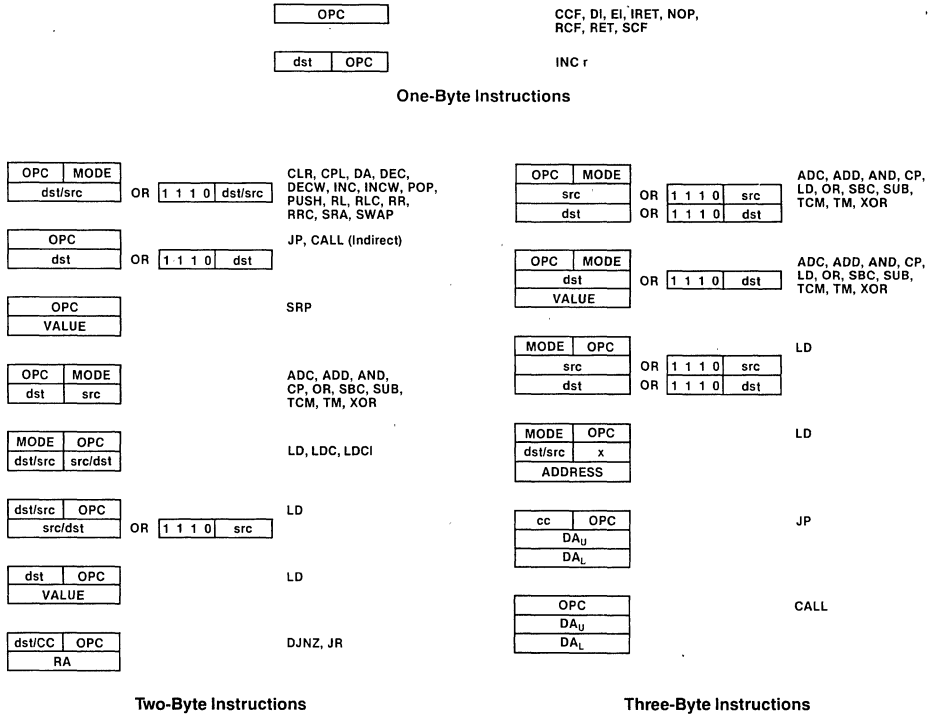


Figure 7. Instruction Formats

INSTRUCTION SUMMARY

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst,src dst ← dst + src + C	(Note 1)		1□	*	*	*	*	0	*
ADD dst,src dst ← dst + src	(Note 1)		0□	*	*	*	*	0	*
AND dst,src dst ← dst AND src	(Note 1)		5□	—	*	*	0	—	—
CALL dst SP ← SP – 2 @SP ← PC; PC ← dst	DA IRR		D6 D4	—	—	—	—	—	—
CCF C ← NOT C			EF	*	—	—	—	—	—
CLR dst dst ← 0	R IR		B0 B1	—	—	—	—	—	—
COM dst dst ← NOT dst	R IR		60 61	—	*	*	0	—	—
CP dst,src dst – src	(Note 1)		A□	*	*	*	*	—	—
DA dst dst ← DA dst	R IR		40 41	*	*	*	X	—	—
DEC dst dst ← dst – 1	R IR		00 01	—	*	*	*	—	—
DECW dst dst ← dst – 1	RR IR		80 81	—	*	*	*	—	—
DI IMR (7) ← 0			8F	—	—	—	—	—	—
DJNZ r,dst r ← r – 1 if r ≠ 0 PC ← PC + dst Range: + 127, – 128	RA		rA r = 0 – F	—	—	—	—	—	—

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags C	Z	S	V	D	H
EI IMR (7) ← 1			9F	—	—	—	—	—	—
HALT			7F						
INC dst dst ← dst + 1	r		rE r = 0 – F	—	*	*	*	—	—
	R		20						
	IR		21						
INCW dst dst ← dst + 1	RR		A0	—	*	*	*	—	—
	IR		A1						
IRET FLAGS ← @SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; IMR (7) ← 1			BF	*	*	*	*	*	*
JP cc,dst if cc is true PC ← dst	DA		cD c = 0 – F	—	—	—	—	—	—
	IRR		30						
JR cc,dst if cc is true, PC ← PC + dst Range: +127, – 128	RA		cB c = 0 – F	—	—	—	—	—	—
LD dst,src dst ← src	r	Im	rC	—	—	—	—	—	—
	r	R	r8						
	R	r	r9 r = 0 – F						
	r	X	C7						
	X	r	D7						
	r	Ir	E3						
	Ir	r	F3						
	R	R	E4						
	R	IR	E5						
	R	IM	E6						
	IR	IM	E7						
	IR	R	F5						
LDC dst,src dst ← src	r	Irr	C2	—	—	—	—	—	—
	Irr	r	D2						
LDCI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir	Irr	C3	—	—	—	—	—	—
	Irr	Ir	D3						
LDE dst,src dst ← src	r	Irr	82	—	—	—	—	—	—
	Irr	r	92						
LDEI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir	Irr	83	—	—	—	—	—	—
	Irr	Ir	93						
NOP			FF	—	—	—	—	—	—
OR dst,src dst ← dst OR src	(Note 1)		4□	—	*	*	0	—	—
POP dst dst ← @SP; SP ← SP + 1	R		50	—	—	—	—	—	—
	IR		51						
PUSH src SP ← SP – 1; @SP ← src		R	70	—	—	—	—	—	—
		IR	71						

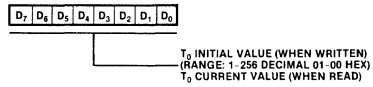
Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags C	Z	S	V	D	H
RCF C ← 0			CF	0	—	—	—	—	—
RET PC ← @SP; SP ← SP + 2			AF	—	—	—	—	—	—
RL dst		R	90	*	*	*	*	—	—
		IR	91						
RLC dst		R	10	*	*	*	*	—	—
		IR	11						
RR dst		R	E0	*	*	*	*	—	—
		IR	E1						
RRC dst		R	C0	*	*	*	*	—	—
		IR	C1						
SBC dst,src dst ← dst ← src ← C	(Note 1)		3□	*	*	*	*	1	*
SCF C ← 1			DF	1	—	—	—	—	—
SRA dst		R	D0	*	*	*	0	—	—
		IR	D1						
SRP src RP ← src		Im	31	—	—	—	—	—	—
STOP			6F						
SUB dst,src dst ← dst ← src	(Note 1)		2□	*	*	*	*	1	*
SWAP dst		R	F0	X	*	*	X	—	—
		IR	F1						
TCM dst,src (NOT dst) AND src	(Note 1)		6□	—	*	*	0	—	—
TM dst,src dst AND src	(Note 1)		7□	—	*	*	0	—	—
XOR dst,src dst ← dst XOR src	(Note 1)		B□	—	*	*	0	—	—

NOTE: These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a □ in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

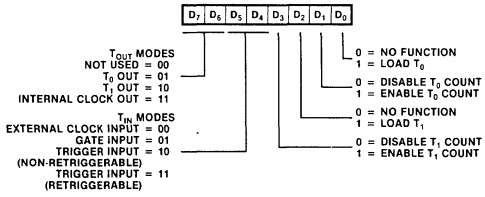
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Addr Mode dst	Addr Mode src	Lower Opcode Nibble
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7

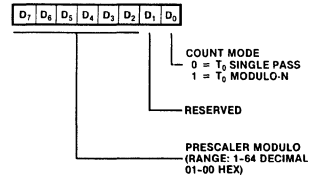
R244 T0
COUNTER/TIMER 0 REGISTER
($F4_H$; Read/Write)



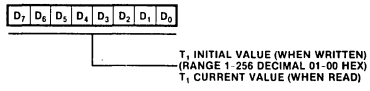
R241 TMR
TIMER MODE REGISTER
($F1_H$; Read/Write)



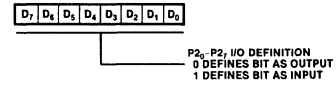
R245 PRE0
PRESCALER 0 REGISTER
($F5_H$; Write Only)



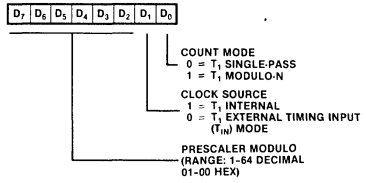
R242 T1
COUNTER TIMER 1 REGISTER
($F2_H$; Read/Write)



R246 P2M
PORT 2 MODE REGISTER
($F6_H$; Write Only)



R243 PRE1
PRESCALER 1 REGISTER
($F3_H$; Write Only)



R247 P3M
PORT 3 MODE REGISTER
($F7_H$; Write Only)

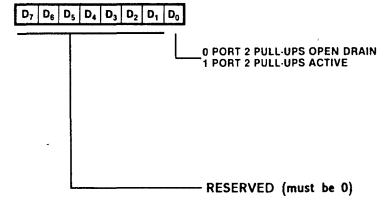
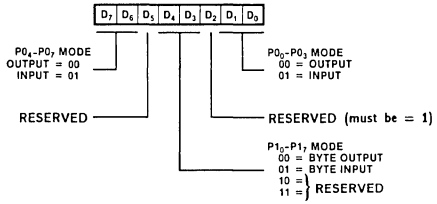
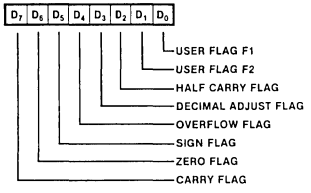


Figure 11. Control Registers

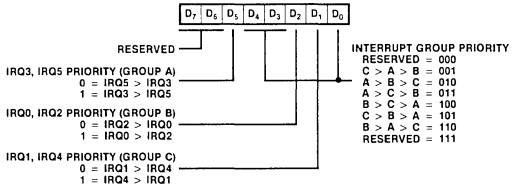
R248 P01M
PORT 0 AND 1 MODE REGISTER
($F8_H$; Write Only)



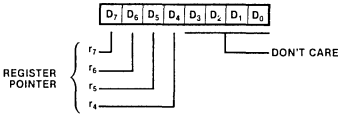
R252 FLAGS
FLAG REGISTER
(FC_H; Read/Write)



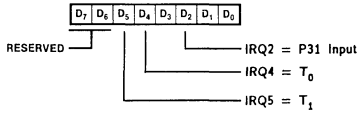
R249 IPR
INTERRUPT PRIORITY REGISTER
($F9_H$; Write Only)



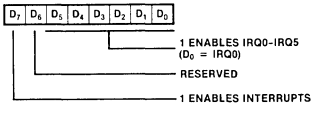
R253 RP
REGISTER POINTER
(FD_H; Read/Write)



R250 IRQ
INTERRUPT REQUEST REGISTER
(FA_H; Read/Write)



R251 IMR
INTERRUPT MASK REGISTER
(FB_H; Read/Write)



R255 SPL
STACK POINTER
(FF_H; Read/Write)

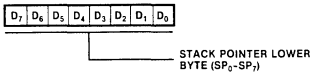
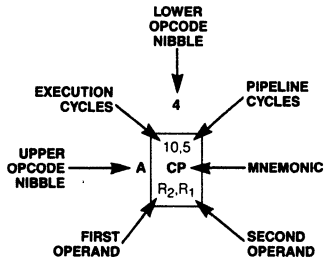


Figure 11. Control Registers (Continued)

OPCODE MAP

Upper Nibble (Hex) \ Lower Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R_1	6.5 **DEC** IR_1	6.5 **ADD** $r_1.r_2$	6.5 **ADD** $r_1.Ir_2$	10.5 **ADD** $R_2.R_1$	10.5 **ADD** $IR_2.R_1$	10.5 **ADD** $R_1.IM$	10.5 **ADD** $IR_1.IM$	6.5 **LD** $r_1 R_2$	6.5 **LD** $r_2 R_1$	12/10.5 **DJNZ** r_1 RA	12/10.0 **JR** cc.RA	6.5 **LD** $r_1.IM$	12/10.0 **JP** cc DA	6.5 **INC** r1	
1	6.5 **RLC** R_1	6.5 **RLC** IR_1	6.5 **ADC** $r_1 r_2$	6.5 **ADC** $r_1.Ir_2$	10.5 **ADC** $R_2.R_1$	10.5 **ADC** $IR_2.R_1$	10.5 **ADC** $R_1.IM$	10.5 **ADC** $IR_1.IM$								
2	6.5 **INC** R_1	6.5 **INC** IR_1	6.5 **SUB** $r_1.r_2$	6.5 **SUB** $r_1.Ir_2$	10.5 **SUB** $R_2.R_1$	10.5 **SUB** $IR_2.R_1$	10.5 **SUB** $R_1.IM$	10.5 **SUB** $IR_1.IM$								
3	8.0 **JP** IRR_1	6.1 **SRP** IM	6.5 **SBC** $r_1.r_2$	6.5 **SBC** $r_1.Ir_2$	10.5 **SBC** $R_2.R_1$	10.5 **SBC** $IR_2.R_1$	10.5 **SBC** $R_1.IM$	10.5 **SBC** $IR_1.IM$								
4	8.5 **DA** R_1	8.5 **DA** IR_1	6.5 **OR** $r_1.r_2$	6.5 **OR** $r_1.Ir_2$	10.5 **OR** $R_2.R_1$	10.5 **OR** $IR_2.R_1$	10.5 **OR** $R_1.IM$	10.5 **OR** $IR_1.IM$								
5	10.5 **POP** R_1	10.5 **POP** IR_1	6.5 **AND** $r_1.r_2$	6.5 **AND** $r_1.Ir_2$	10.5 **AND** $R_2.R_1$	10.5 **AND** $IR_2.R_1$	10.5 **AND** $R_1.IM$	10.5 **AND** $IR_1.IM$								
6	6.5 **COM** R_1	6.5 **COM** IR_1	6.5 **TCM** $r_1.r_2$	6.5 **TCM** $r_1.Ir_2$	10.5 **TCM** $R_2.R_1$	10.5 **TCM** $IR_2.R_1$	10.5 **TCM** $R_1.IM$	10.5 **TCM** $IR_1.IM$								6.0 **STOP**
7	10/12.1 **PUSH** R_2	12/14.1 **PUSH** IR_2	6.5 **TM** $r_1.r_2$	6.5 **TM** $r_1.Ir_2$	10.5 **TM** $R_2.R_1$	10.5 **TM** $IR_2.R_1$	10.5 **TM** $R_1.IM$	10.5 **TM** $IR_1.IM$								7.0 **HALT**
8	10.5 **DECW** RR_1	10.5 **DECW** IR_1														6.1 **DI**
9	6.5 **RL** R_1	6.5 **RL** IR_1														6.1 **EI**
A	10.5 **INCW** RR_1	10.5 **INCW** IR_1	6.5 **CP** $r_1.r_2$	6.5 **CP** $r_1.Ir_2$	10.5 **CP** $R_2.R_1$	10.5 **CP** $IR_2.R_1$	10.5 **CP** $R_1.IM$	10.5 **CP** $IR_1.IM$								14.0 **RET**
B	6.5 **CLR** R_1	6.5 **CLR** IR_1	6.5 **XOR** $r_1.r_2$	6.5 **XOR** $r_1.Ir_2$	10.5 **XOR** $R_2.R_1$	10.5 **XOR** $IR_2.R_1$	10.5 **XOR** $R_1.IM$	10.5 **XOR** $IR_1.IM$								16.0 **IRET**
C	6.5 **RRC** R_1	6.5 **RRC** IR_1	12.0 **LDC** $r_1.Irr_2$	18.0 **LDCI** $Ir_1.Irr_2$				10.5 **LD** $r_1.x.R_2$								6.5 **RCF**
D	6.5 **SRA** R_1	6.5 **SRA** IR_1	12.0 **LDC** $r_2.Irr_1$	18.0 **LDCI** $Ir_2.Irr_1$	20.0 **CALL*** IRR_1		20.0 **CALL** DA	10.5 **LD** $r_2.x\ R_1$								6.5 **SCF**
E	6.5 **RR** R_1	6.5 **RR** IR_1		6.5 **LD** $r_1.IR_2$	10.5 **LD** $R_2.R_1$	10.5 **LD** $IR_2.R_1$	10.5 **LD** $R_1.IM$	10.5 **LD** $IR_1.IM$								6.5 **CCF**
F	8.5 **SWAP** R_1	8.5 **SWAP** IR_1		6.5 **LD** $Ir_1.r_2$		10.5 **LD** $R_2.IR_1$										6.0 **NOP**
Bytes per Instruction	2				3				2					3	1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_1 = Dst address
R_2 or r_2 = Src address

Sequence:
Opcode, First Operand, Second Operand

NOTE: The blank areas are not defined.

*2-byte instruction, fetch cycle appears as a 3-byte instruction

ABSOLUTE MAXIMUM RATINGS

Voltages on all pins with respect to GND . . . . . −0.3V to +7.0V
Operating Ambient Temperature . . . . . See Ordering Information
Storage Temperature . . . . . −65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

STANDARD TEST CONDITIONS

The DC characteristics listed below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin.

Standard conditions are as follows:

- +4.5 ≤ Vcc ≤ +5.5
- GND = 0V
- 0°C ≤ T_A ≤ +70°C

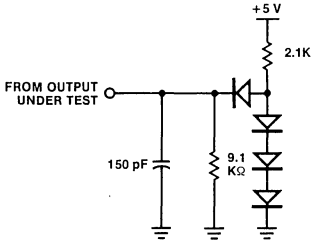


Figure 12. Test Load 1

DC CHARACTERISTICS

Symbol	Parameter	Min	Typ	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	3.8		V_{CC}	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	−0.3		0.8	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0		V_{CC}	V	
V_{IL}	Input Low Voltage	−0.3		0.8	V	
V_{RH}	Reset Input High Voltage	3.8		V_{CC}	V	
V_{RL}	Reset Input Low Voltage	−0.3		0.8	V	
V_{OH}	Output High Voltage	2.4			V	I_{OH} = −250 μA
V_{OH}	**Output High Voltage**	**V_{CC} -100 mV**			**V**	**I_{OH} = -100μA**
V_{OL}	Output Low Voltage			0.4	V	I_{OL} = +2.0 mA
I_{IL}	Input Leakage	−10		10	μA	0V ≤ V_{IN} ≤ + 5.25V
I_{OL}	Output Leakage	−10		10	μA	0V ≤ V_{IN} ≤ + 5.25V
I_{IR}	Reset Input Current			−50	μA	V_{CC} = +5.25V, V_{RL} = 0V
I_{CC}	Supply Current			50	mA	All outputs and I/O pins floating
I_{CC1}	Standby Current		5		mA	Halt Mode
I_{CC2}	Standby Current			10	μA	Stop Mode

NOTE:

Icc2 low power requires loading TMR (%F1) with any value prior to stop execution.
Use sequence:

```
LD TMR, #%00.
NOP
STOP
```

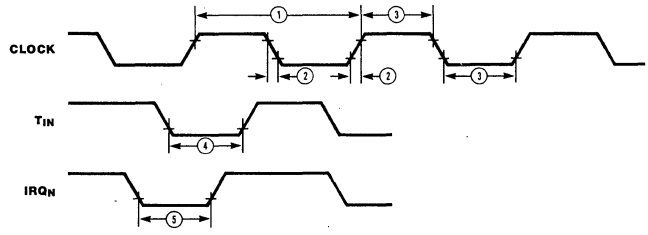


Figure 14. Additional Timing

AC CHARACTERISTICS

Additional Timing Table

Number	Symbol	Parameter	Z86C10 Min	Z86C10 Max	Notes*
1	TpC	Input Clock Period	**83**	**100,000**	1
2	TrC,TfC	Clock Input Rise and Fall Times		**15**	1
3	TwC	Input Clock Width	**70**		1
4	TwTinL	Timer Input Low Width	**70**		2
5	TwIL	Interrupt Request Input Low Time	**70**		2,3

NOTES:
1. Clock timing references use 3.8V for a logic "1" and 0.8V for a logic "0".
2. Timing references use 2.0V for a logic "1" and 0.8V for a logic "0".
3. Interrupt request via Port 3.
* Units in nanoseconds (ns).

Z86C06

CMOS Z8® CCP™ CONSUMER CONTROLLER PROCESSOR

FEATURES

- 8-bit CMOS microcontroller
- 18-pin DIP package
- Low Cost
- 3.0 to 5.5 volt operating range
- Fast instruction pointer - 1.0 microseconds @ 12 MHz
- Two standby modes - STOP and HALT
- 14 input/output lines (two with Comparator inputs)
- 1 Kbyte of ROM
- 124 bytes of RAM
- Four Expanded Register File Control Registers and two SPI Registers
- Two programmable 8-bit Counter/Timers
- 6-bit programmable prescaler
- Six vectored, priority interrupts from five different sources
- Clock speeds 4, 8, and 12 MHz
- Brown-Out protection
- Watchdog/Power-On Reset Timer
- Two Comparators with programmable interrupt polarity
- On-chip oscillator that accepts a crystal, ceramic resonator, LC, RC, or external clock drive.
- Serial Peripheral Interface (SPI)
- Low EMI Noise Mode
- Up to -40°C to 105°C operation

GENERAL DESCRIPTION

The Z86C06 CCP (Consumer Controller Processor) is a member of the Z8 single-chip microcontroller family with 1 Kbyte of ROM, and 124 bytes of General Purpose RAM. The device is housed in an 18-pin DIP, and is manufactured in CMOS technology. Zilog's CMOS microcontroller offers fast execution, efficient use of memory, sophisticated interrupts, input/output bit manipulation capabilities, and easy hardware/software system expansion along with low cost and low power consumption.

The Z86C06 architecture is based on Zilog's 8-bit microcontroller core with the addition of an Expanded Register File which allows access to register mapped peripheral and I/O circuits. The CCP offers a flexible I/O scheme, and a number of ancillary features that are useful in many consumer, industrial, automotive, and advanced scientific applications.

The device applications demand powerful I/O capabilities. The CCP fulfills this with 14 pins dedicated to input and output. These lines are grouped into two ports, and are configurable under software control to provide timing, status signals, or parallel I/O.

Three basic address spaces are available to support this wide range of configurations; Program Memory, Register File, and Expanded Register File. The Register File is composed of 124 bytes of General-Purpose Registers, two I/O Port registers and fifteen Control and Status registers. The Expanded Register File consists of four control registers, SPI Receive Buffer, and the SPI compare register.

GENERAL DESCRIPTION (Continued)

With powerful peripheral features such as on-board comparators, counter/timers, watch dog timer, and serial peripheral interface, the Z86C06 meets the needs for most sophisticated controller applications (Figure 1).

Note: All Signals with a preceding front slash, "/", are active Low, e.g.: B//W (WORD is active Low); /B/W (BYTE is active Low, only); /N//S (NORMAL and SYSTEM are both active Low).

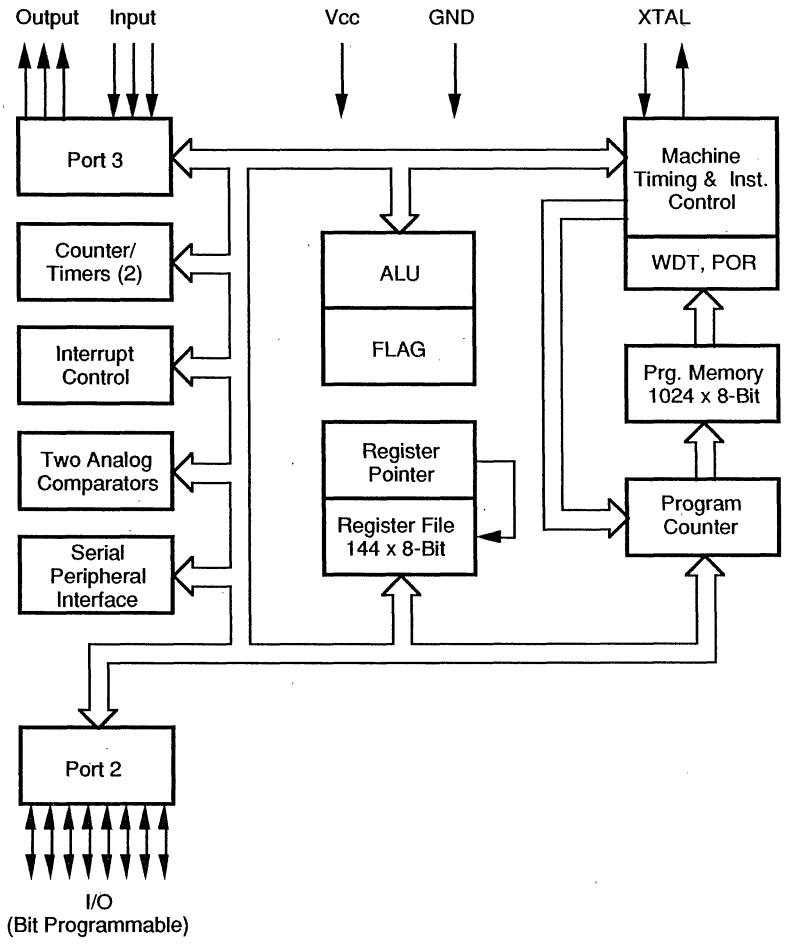


Figure 1. Functional Block Diagram

PIN DESCRIPTION

Table 1. Pin Identification

No	Symbol	Function	Direction
1-4	P24-7	Port 2 pin 4, 5, 6, 7	In/Output
5	VCC	Power Supply	Input
6	XTAL2	Crystal Oscillator Clock	Output
7	XTAL1	Crystal Oscillator Clock	Input
8-10	P31-3	Port 3 pin 1, 2, 3	Fixed Input
11-13	P34-6	Port 3 pin 4, 5, 6	Fixed Output
14	GND	Ground	Input
15-18	P20-3	Port 2 pin 0, 1, 2, 3	In/Output

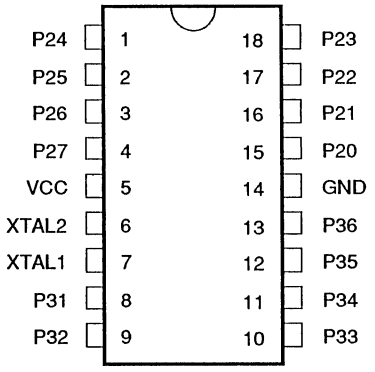


Figure 2. Pin Configuration

PIN FUNCTIONS

XTAL1. *Crystal 1* (time-based input). This pin connects a parallel-resonant crystal, ceramic resonator, LC or RC network or an external single-phase clock to the on-chip oscillator input.

XTAL2. *Crystal 2* (time-based output). This pin connects a parallel-resonant crystal, ceramic resonator, LC or RC network to the on-chip oscillator output.

Port 2 P20-P27. Port 2 is an 8-bit, bi-directional, CMOS compatible I/O port. These 8 I/O lines can be configured under software control to be an input or output, independently. Input buffers are Schmitt-triggered and contain Auto-Latches. Bits programmed as outputs may be globally programmed as either push-pull or open drain (Figure 3a and 3b). In addition when the SPI is enabled, P20 functions as data-in (DI), and P27 functions as data-out (DO) for the SPI.

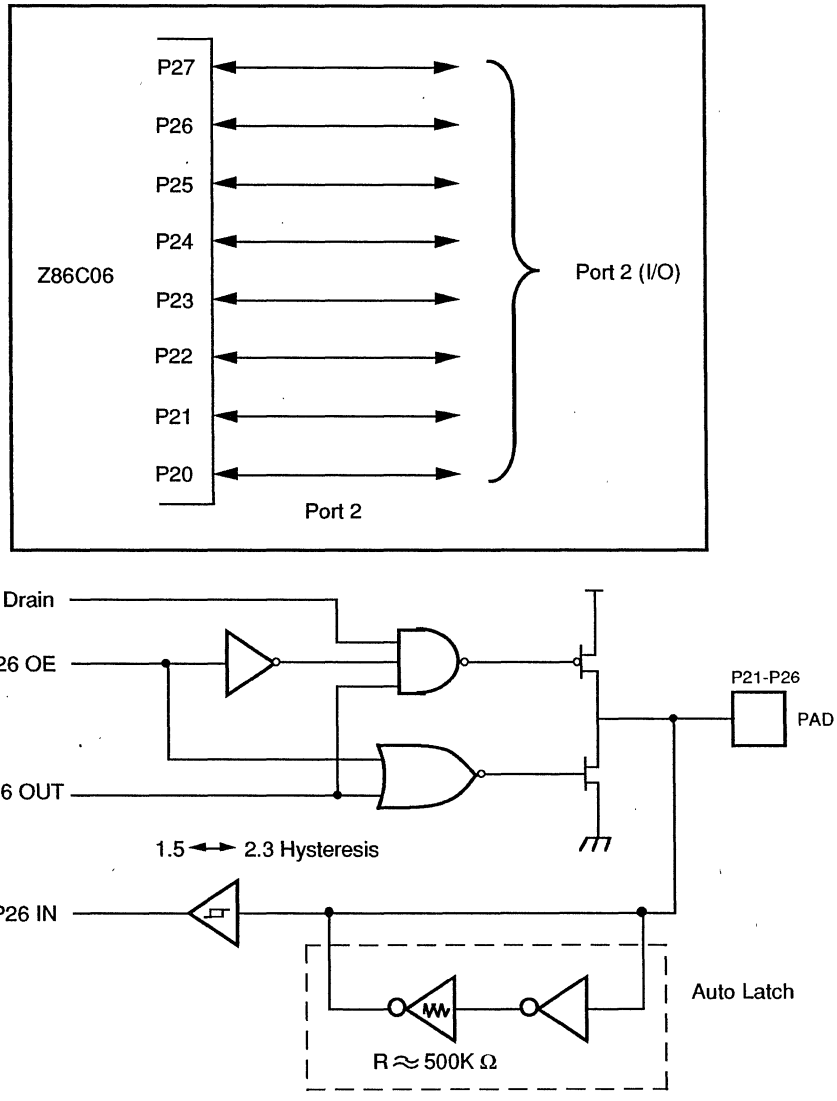


Figure 3a. Port 2 Configuration

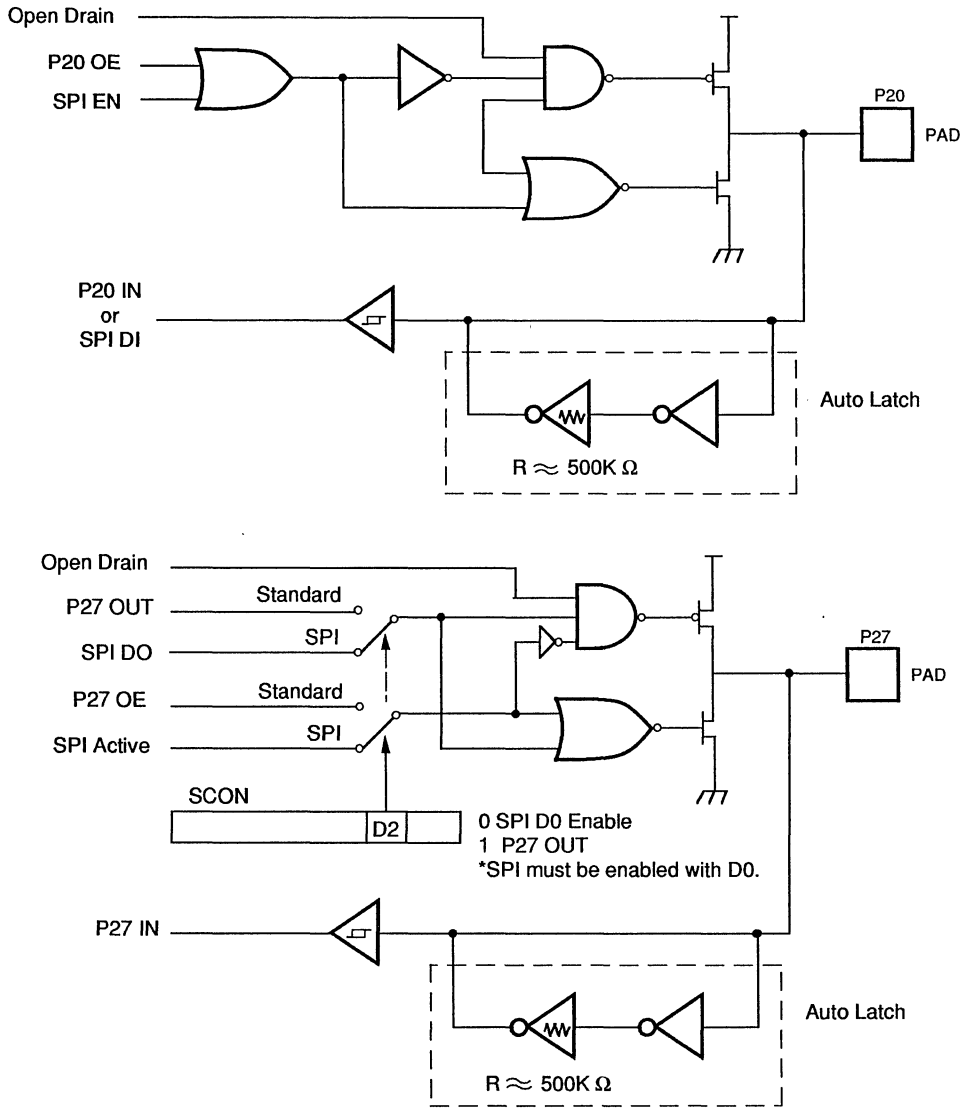
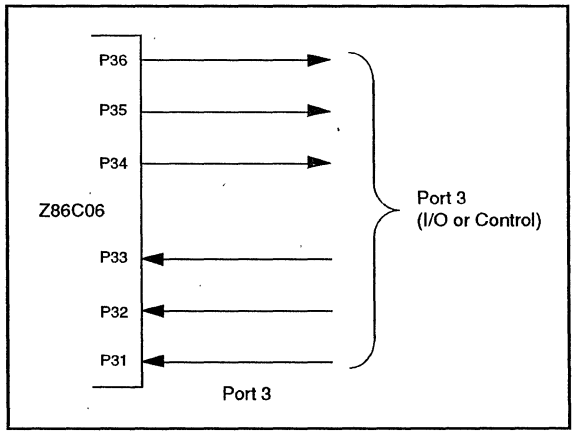


Figure 3b. Port 2 Configuration

PIN FUNCTIONS (Continued)

Auto-Latch. The Auto-Latch puts valid CMOS levels on all CMOS inputs that are not externally driven. Whether this level is zero or one cannot be determined. A valid CMOS level rather than a floating node reduces excessive supply current flow in the input buffer.

Port 3 P31-P36. Port 3 is a 6-bit, CMOS compatible, port. These six lines consist of three fixed inputs (P31-P33) and three fixed outputs (P34-P36). Pins P31, P32 and P33 are standard CMOS inputs (no auto-latches) and pins P34, P35, and P36 are push-pull outputs. Two on-board comparators can process analog signals on P31 and P32 with reference to the voltage on P33. The analog function is enabled by programming Port 3 Mode Register (bit 1). Pins P31 and P32 are programmable as falling, rising, or both edge triggered interrupts (IRQ register bits 6 and 7). P33 is the comparator reference voltage input. Access to Counter/Timer 1 is made through P31 (Tin) and P36 (Tout). Pin P34 can also be configured as SPI clock (SK), input and output, and pin P35 can be configured as Slave select (SS) in slave mode only, when the SPI is enabled (Figures 4a. and 4b.).



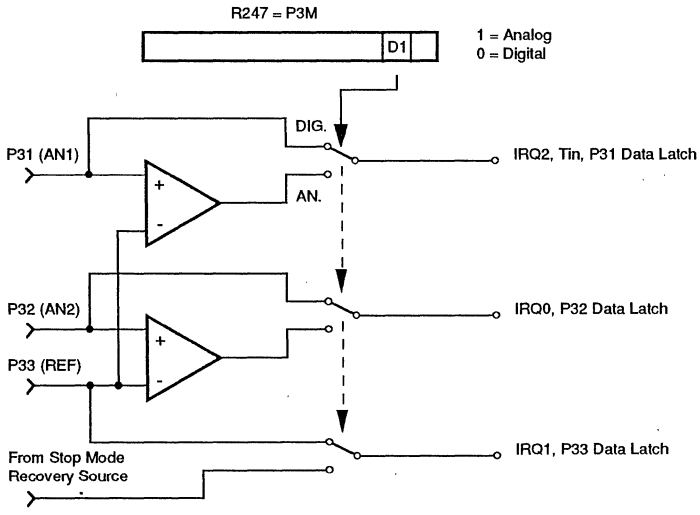


Figure 4a. Port 3 Configuration

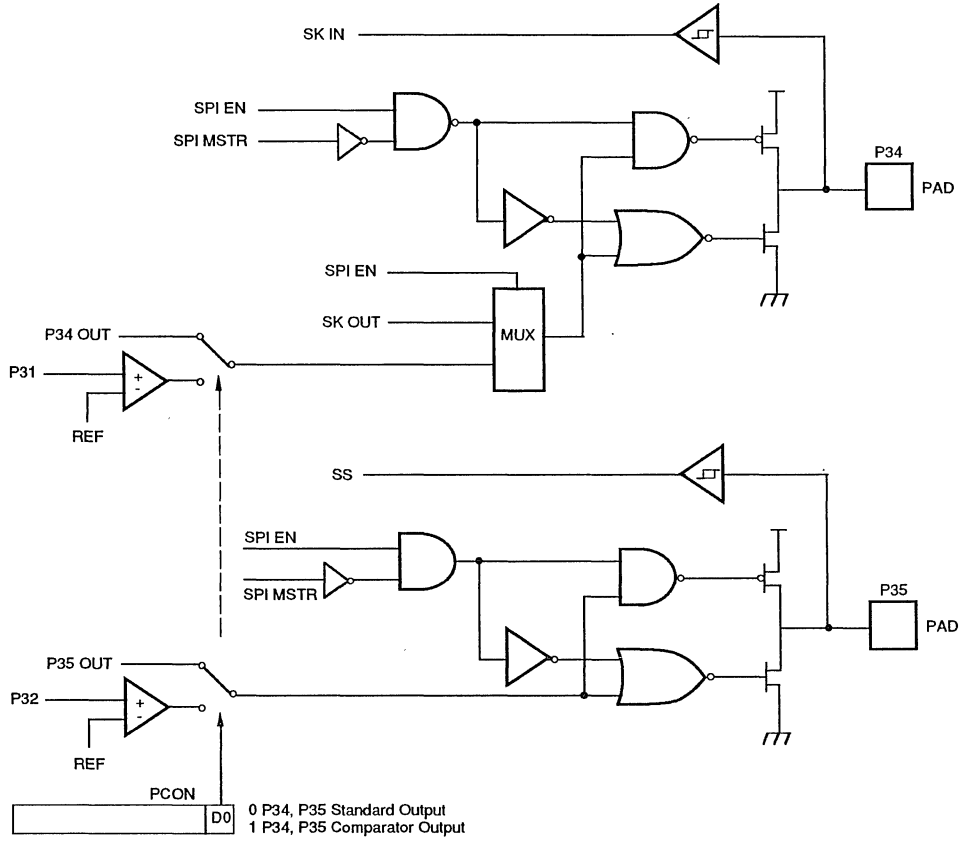


Figure 4b. Port 3 Configuration

PORT Configuration Register (PCON). The PORT Configuration Register (PCON) configures the port's individually for; comparator output on Port 3, low EMI noise on Port's 2 and 3, and low EMI noise oscillator. The PCON Register is located in the Expanded Register File at bank F, location 00 (Figure 5). Bit 0 controls the comparator use in Port 3. A 1 in this location brings the comparator outputs to P34 and P35 (Figure 4b), and a 0 releases the Port to its standard I/O configuration. Bits 5 and 6 of this register configure Port's 2 and 3, respectively, for low EMI operation. A 1 in these locations configures the port for standard operation, and a 0 configures the port for low EMI operation. Finally, bit 7 of the PCON Register controls the low EMI noise oscillator. A 1 in this location configures the oscillator with standard drive, while a 0 configures the oscillator with low noise drive.

PIN FUNCTIONS (Continued)

Low EMI Option. The Z86C06 can be programmed to operate in a low EMI emission mode by the PCON register. The oscillator and all I/O ports can be programmed as low EMI emission mode independently. Use of this feature results in:

- Less than 1 mA current consumption during the HALT mode.
- The pre-drivers slew rate reduced to 10 ns typical.
- Low EMI output drivers have resistance of 200 ohms (typical).
- Oscillator divide-by-two circuitry is eliminated.
- Internal SLCK/TCLK operation limited to a maximum of 4 MHz (250 ns cycle time)

Comparator Inputs. Port 3, Pin P31 and Pin P32 each have a comparator front end. The comparator reference voltage, Pin P33, is common to both comparators. In analog mode, the P33 input functions as a reference voltage to the comparators. The internal P33 register and its corresponding IRQ1 is connected to the STOP Mode Recovery source selected by the SMR. In this mode, any of the STOP Mode Recovery sources are used to toggle the P33 bit or generate IRQ1. In digital mode, Pin P33 can be used as a P33 register input or IRQ1 source (Figure 17).

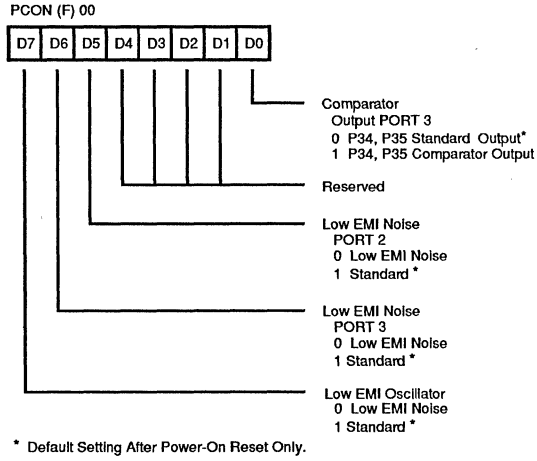


Figure 5. PORT Configuration Register (PCON)

FUNCTIONAL DESCRIPTION

The Z8 CCP incorporates special functions to enhance the Z8's application in consumer, automotive, industrial, scientific research, and advanced technologies applications.

RESET. The device is reset in one of the following conditions:

- Power-On Reset
- Watch-Dog Timer
- STOP Mode Recovery Source

The device does not re-initialize the WDTMR, SMR, P2M, or P3M registers to their reset values on a STOP Mode Recovery operation.

Program Memory. Z86C06 can address up to 1 Kbytes of internal program memory (Figure 6). The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts. Byte 13 to byte 1023 consists of on-chip, mask-programmed ROM.

ROM Protect. The 1 Kbytes of Program Memory is mask programmable. A ROM protect feature will prevent "dumping" of the ROM contents by inhibiting execution of the LDC and LDCI instructions to program memory in all modes.

Expanded Register File. The register file has been expanded to allow for additional system control registers and for mapping of additional peripheral devices and input/output ports into the register address area. The Z8 register address space R0 through R15 is implemented as 16 groups of 16 registers per group (Figure 7). These register groups are known as the ERF (Expanded Register File). Bits 3:0 of the Register Pointer (RP) select the active ERF group. Bits 7:4 of register RP select the working register group (Figure 8). Three system configuration registers reside in the Expanded Register File address space in Bank F, while three SPI registers reside in Bank C. The rest of the Expanded Register addressing space is not physically implemented, and is open for future expansion. To write to the ERF, the upper nibble of the RP must be zero. To write to the rest of the register file, the lower nibble must be zero.

Note:
When using Zilog's cross assembler Version 2.1 or earlier, use the LD RP, #0X instruction rather than the SRP #0X instruction to access the ERF.

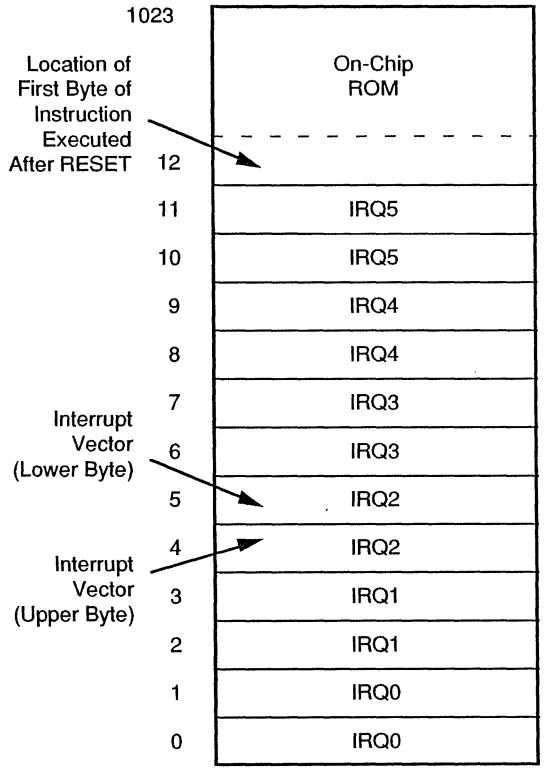


Figure 6. Program Memory Map

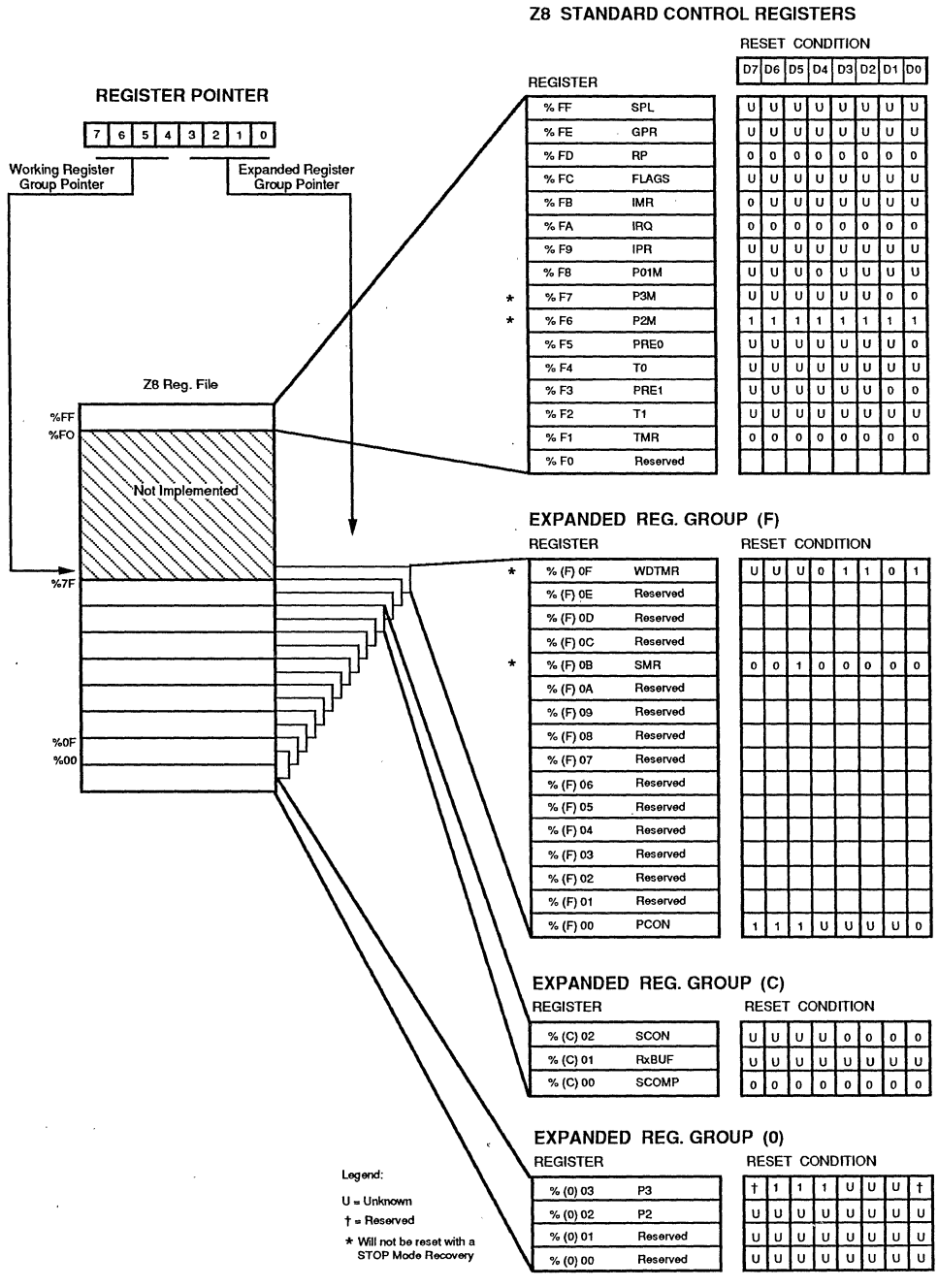
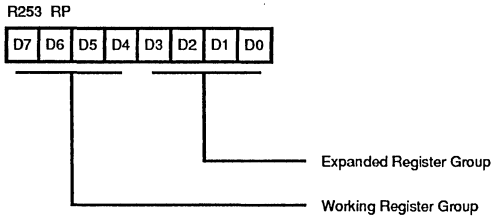


Figure 7. Expanded Register File Architecture



Note: Default Setting After Reset = 00000000

Figure 8. Register Pointer Register

Register File. The Register File consists of two I/O port registers, 124 general purpose registers, 15 control and status registers, and four system configuration registers in the Expanded Register Group (Figure 7). The instructions can access registers directly or indirectly via an 8-bit address field. This allows a short 4-bit register address using the Register Pointer (Figure 9). In the 4-bit mode, the Register File is divided into 16 working register groups, each occupying 16 continuous locations. The Register Pointer addresses the starting location of the active working-register group.

Note: Register Bank E0-EF is only accessed through working registers and indirect addressing modes.

Caution: D4 of Control Register P01M (R251) must be 0. If the Z86C06 is emulated by Z86C90, D4 of P01M has to change to 0 before submission to ROM code.

GPR. The Z86C06 has one extra General Purpose Register located at %FE(R254).

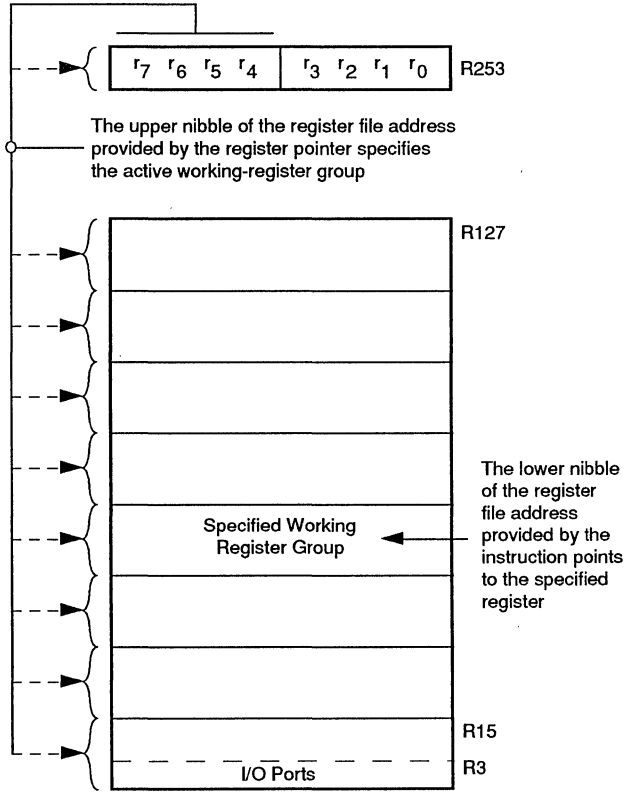


Figure 9. Register Pointer

Stack. The Z86C06 has an 8-bit Stack Pointer (R255) used for the internal stack that resides within the 124 general-purpose registers.

Counter/Timers. There are two 8-bit programmable counter/timers (T0-T1), each driven by its own 6-bit programmable prescaler. The T1 prescaler can be driven by internal or external clock sources, however, the T0 prescaler is driven by the internal clock only (Figure 10).

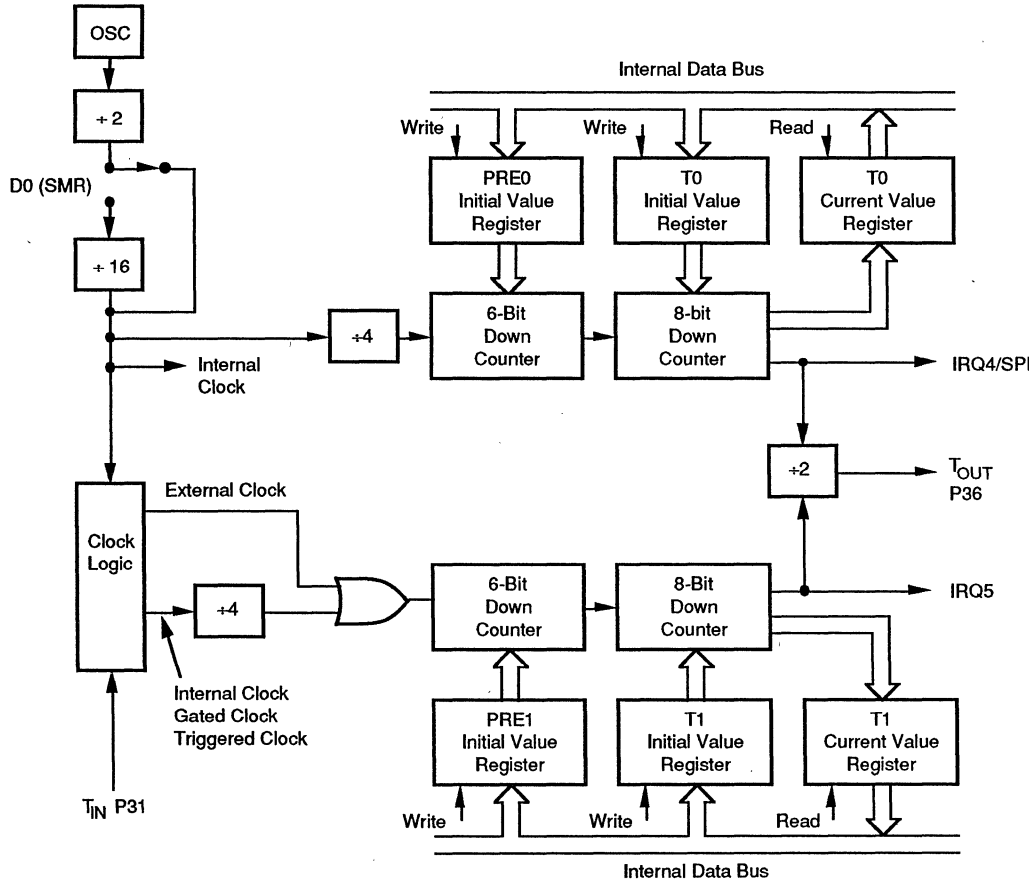


Figure 10. Counter/Timer Block Diagram

The 6-bit prescalers divide the input frequency of the clock source by any integer number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request-IRQ4 (T0) or IRQ5 (T1), is generated.

The counters are programmed to start, stop, restart to continue, or restart from the initial value. The counters can also be programmed to stop upon reaching zero (single-pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode).

The counters, but not the prescalers, are read at any time without disturbing their value or count mode. The clock source for T1 is user-definable and can be either the internal microprocessor clock divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input (P31) as an external clock, a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. Port 3, line P36 serves as a timer output (Tout) through which T0, T1 or the internal clock can be output. The counter/timers can be cascaded by connecting the T0 output to the input of T1.

Interrupts. The Z86C06 has six different interrupts from six different sources. The interrupts are mask-able and prioritized (Figure 11). The six sources are divided as follows; three sources are claimed by Port 3 lines P31-P33, two sources in the counter/timers, and one source for the SPI. The Interrupt Mask Register globally or singularly enables or disables the six interrupt requests (Table 2).

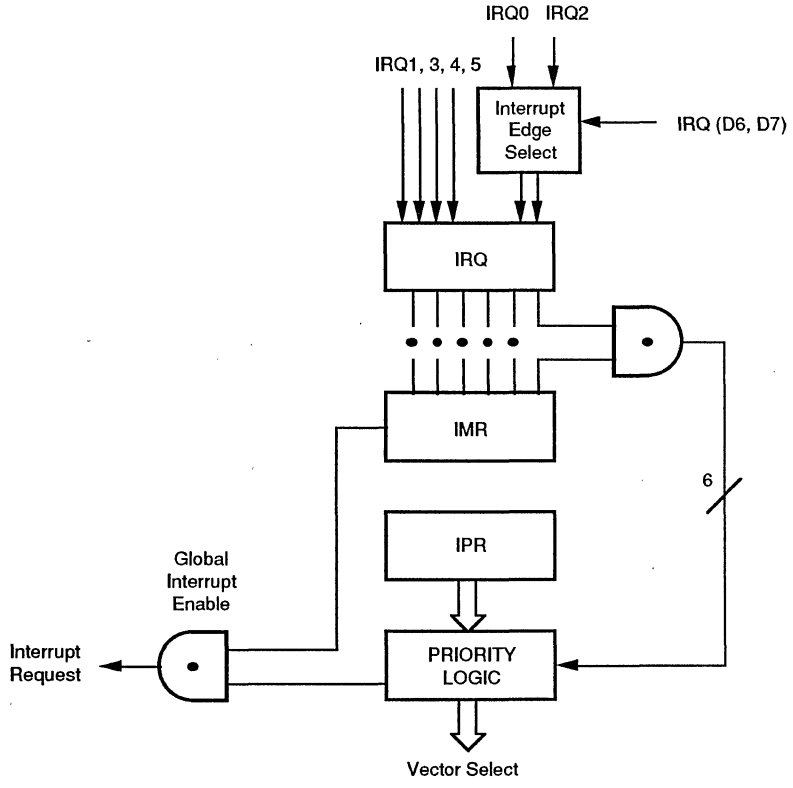


Figure 11. Interrupt Block Diagram

FUNCTIONAL DESCRIPTION (Continued)

Table 2. Interrupt Types, Sources, and Vectors

Name	Source	Vector Location	Comments
IRQ 0	IRQ 0	0, 1	External (P32), Rising Falling Edge Triggered
IRQ 1	IRQ 1	2, 3	External (P33), Falling Edge Triggered
IRQ 2	IRQ 2, T_{IN}	4,5	External (P31), Rising Falling Edge Triggered
IRQ 3		6, 7	Software Generated
IRQ 4	T0	8, 9	Internal
IRQ 5	TI	10, 11	Internal

Note:
When SPI is enabled IRQ3 is an internal interrupt.

When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register. An interrupt machine cycle is activated when an interrupt request is granted. This disables all subsequent interrupts, saves the Program Counter and Status Flags, and then branches to the program memory vector location reserved for that interrupt. All Z86C06 interrupts are vectored through locations in the program memory. This memory location and the next byte contain the 16-bit starting address of the interrupt service routine for that particular interrupt request.

To accommodate polled interrupt systems, interrupt inputs are masked and the interrupt request register is polled to determine which of the interrupt requests needs services. When the SPI is disabled, IRQ3 has no hardware source but can be invoked by software (write to IRQ3 Register). When the SPI is enabled, an interrupt will be mapped to IRQ3 after a byte of data has been received by the SPI Shift Register.

An interrupt resulting from AN1 is mapped into IRQ2, and an interrupt from AN2 is mapped into IRQ0. Interrupts IRQ2 and IRQ0 may be rising, falling, or both edge triggered, and are programmable by the user. The software can poll to identify the state of the pin.

The programming bits for the INTERRUPT EDGE SELECT are located in the IRQ register (R250), bits D7 and D6. The configuration is shown in Table 3.

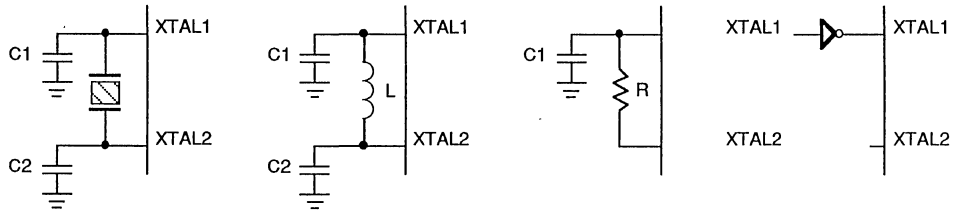
Table 3. IRQ Register

IRQ		Interrupt Edge	
D7	D6	P31	P32
0	0	F	F
0	1	F	R
1	0	R	F
1	1	R/F	R/F

Notes:
F = Falling Edge
R = Rising Edge

Clock. The Z86C06 on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal, RC, ceramic resonator, or any suitable external clock source (XTAL1 = Input, XTAL2 = Output). The crystal should be AT cut, 10 KHz to 12 MHz max, with a series resistance (RS) less than or equal to 100 Ohms.

The crystal is connected across XTAL1 and XTAL2 using the recommended capacitors (C1=C2 is more than or equal to 22 pF) from each pin to ground. The RC oscillator option is mask-programmable, to be selected by the customer at the time the ROM code is submitted. The RC oscillator configuration must be an external resistor connected from XTAL1 to XTAL2, with a frequency-setting capacitor from XTAL1 to ground (Figure 12).



Ceramic Resonator or Crystal
C1, C2 = 47 pF TYP *
F = 8 MHz

LC
C1, C2 = 22 pF

L = 130 μH *
F = 3 MHz *

RC
@ 5V Vcc (TYP)

External Clock

* Preliminary Value Including Pin Parasitics

Figure 12. Oscillator Configuration

The RC value vs Frequency curves are shown in Figure 54 and 55. (**Limitation:** The RC option is not available in the 12 MHz part.) In addition, a special feature has been incorporated into the Z86C06; in low EMI noise mode (bit 7 of PCON register=0) with the RC option selected, the oscillator is targeted to consume considerately less ICC current at frequencies of 10 KHz or less.

Power-On Reset. A timer circuit clocked by a dedicated on-board RC oscillator or by the XTAL oscillator is used for the Power-On Reset (POR) timer function. The POR time allows V_{CC} and the oscillator circuit to stabilize before instruction execution begins. The POR timer circuit is a one-shot timer triggered by one of the three conditions:

- Power fail to Power OK status
- STOP mode recovery (If D5 of SMR=1)
- WDT timeout

The POR time is a nominal 5 ms. Bit 5 of the STOP Mode Register determines whether the POR timer is bypassed after STOP mode recovery (typical for external clock, and RC/LC oscillators with fast start up time).

HALT. Will turn off the internal CPU clock but not the XTAL oscillation. The counter/timers and external interrupts IRQ0, IRQ1, and IRQ2 remain active. The device is recovered by interrupts, either externally or internally generated.

STOP. This instruction turns off the internal clock and external crystal oscillation and reduces the standby current to 10 microamperes or less. The STOP mode is terminated by a RESET of either WDT timeout, POR, SPI compare, or SMR recovery. This causes the processor to restart the application program at address 000C (HEX). Note, the crystal remains active in STOP mode if bits 3 and 4 of the WDTMR are enabled. In this mode, only the watch dog timer runs in STOP mode.

In order to enter STOP (or HALT) mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. To do this, the user executes a NOP (opcode=FFH) immediately before the appropriate sleep instruction, i.e.:

```
FF    NOP; clear the pipeline
6F    STOP; enter STOP mode
or
FF    NOP; clear the pipeline
7F    HALT; enter HALT mode
```

Serial Peripheral Interface (SPI). The Z86C06 incorporates a serial peripheral interface for communication with other microcontrollers and peripherals. The SPI includes features such as STOP Mode Recovery, Master/Slave selection, and Compare mode. Table 4 contains the pin configuration for the SPI feature when it is enabled. The SPI consists of four registers; SPI Control Register (SCON), SPI Compare Register (SCOMP), SPI Receive/Buffer Register (RxBUF), and SPI Shift Register. SCON is located in bank (C) of the Expanded Register Group at address 02 (Figure 13). This register is a read/write register that controls; Master/Slave selection, interrupts, clock source and phase selection, and error flag. Bit 0 enables/disables the SPI with the default being SPI disabled. A one in this location will enable the SPI, and a 0 will disable the SPI. Bits 1 and 2 of the SCON register in Master mode select the clock rate. The user may choose whether internal clock is divide by 2, 4, 8 or 16. In slave mode, Bit 1 of this register flags the user if an overrun of the RxBUF Register has occurred. The RxCharOverrun flag is only reset by writing a 0 to this bit. In slave mode, bit 2 of the Control Register disables the data-out I/O function. If a 1 is written to this bit, the data-out pin is released to its original port configuration. If a 0 is written to this bit, the SPI shifts out one bit for each bit received. Bit 3 of the SCON Register enables the compare feature of the SPI, with the default being disabled. When the compare feature is enabled, a comparison of the value in the SCOMP Register is made with the value in the RxBUF Register. Bit 4 signals that a receive character is available in the RxBUF Register. If the associated IRQ3 is enabled, an interrupt is generated. Bit 5 controls the clock phase of the SPI. A 1 in Bit 5 allows for receiving data on the clock's falling edge and transmitting data on the clock's rising edge. A 0 allows receiving data on the clock's rising edge and transmitting on the clock's falling edge. The SPI clock source is defined in bit 6. A 1 uses Timer0 output for the SPI clock, and a 0 uses TCLK for clocking the SPI. Finally bit 7 determines whether the SPI is used as a Master or a Slave. A 1 puts the SPI into Master mode and a 0 puts the SPI into Slave mode.

Table 4. SPI Pin Configuration

Name	Function	Pin Location
DI	Data-In	P20
DO	Data-Out	P27
SS	Slave Select	P35
SK	SPI Clock	P34

SPI Operation. The SPI is used in one of two modes; either as system slave, or a system master. Several of the possible system configurations are shown in Figure 14. In the slave mode, data transfer starts when the slave select (SS) pin goes active. Data is transferred into the slave's SPI Shift Register, through the DI pin, which has the same address as the RxBUF Register. After a byte of data has been received by the SPI Shift Register, a Receive Character Available (RCA/IRQ3) flag and interrupt is generated. The next byte of data will be received at this time. The RxBUF Register must be cleared, or a Receive Character Overrun (RxCharOverrun) flag will be set in the SCON Register, and the data in the RxBUF Register will be overwritten. When the communication between the master and slave is complete, the SS goes inactive.

Unless disconnected, for every bit that is transferred into the slave through the DI pin, a bit is transferred out through the DO pin on the opposite clock edge. During slave operation, the SPI clock pin (SK) is an input. In master mode, the CPU must first activate a SS through one of it's I/O ports. Next, data is transferred through the master's DO pin one bit per master clock cycle. Loading data into the shift register initiates the transfer. In master mode, the master's clock will drive the slave's clock. At the conclusion of a transfer, a Receive Character Available (RCA/IRQ3) flag and interrupt is generated. Before data is transferred via the DO pin, the SPI Enable bit in the SCON Register must be enabled.

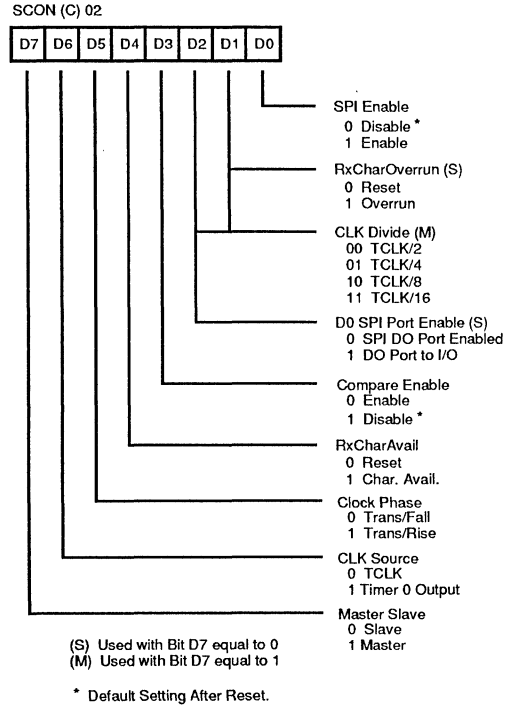


Figure 13. SPI Control Register (SCON)

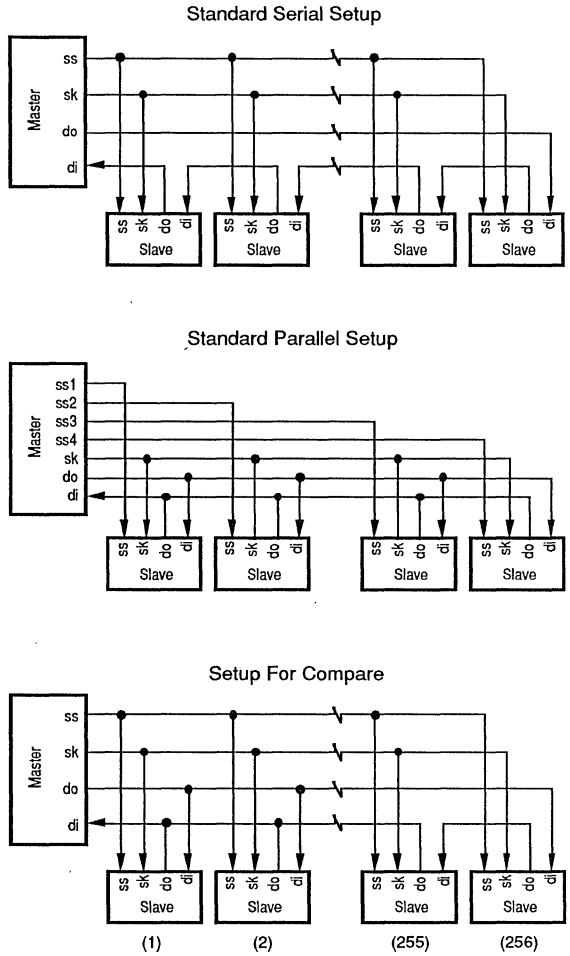


Figure 14. SPI System Configuration

SPI Compare. When the SPI Compare Enable bit, D3 of the SCON Register is set to 1, the SPI Compare feature is enabled. The compare feature is only valid for slave mode. A compare transaction begins when the (SS) line goes active. Data is received as if it were a normal transaction, but there is no data transmitted to avoid bus contention with other slave devices. When the compare byte is received, IRQ3 is not generated. Instead, the data is compared with the contents of the SCOMP Register. If the data does not match, DO remains inactive and the slave ignores all data until the (SS) signal is reset. If the data received matches the data in the SCOMP register, then a SMR signal is generated. DO is activated if it is not tri-stated by D2 in the SCON Register, and data is received the same as any other SPI slave transaction.

When the SPI is activated as a slave, it operates in all system modes; STOP, HALT, and RUN. Slaves' not comparing remain in their current mode, whereas slaves' comparing wake from a STOP or HALT mode by means of an SMR.

SPI Clock. The SPI clock is driven from three sources; with Timer0, a division of the internal system clock, or an external master when in slave mode. Bit D6 of the SCON Register controls what source drives the SPI clock. A 0 in bit D6 of the SCON Register determines the division of the internal system clock if this is used as the SPI clock source. Divide by 2, 4, 8, or 16 is chosen as the scaler.

Receive Character Available and Overrun. When a complete data stream is received, an interrupt is generated and the RxCharAvail bit in the SCON Register is set. Bit 4 in the SCON Register is for enabling or disabling the RxCharAvail interrupt. The RxCharAvail bit is available for interrupt polling purposes and is reset when the RxBUF Register is read. RxCharAvail is generated in both master and slave modes. While in slave mode, if the RxBUF is not read before the next data stream is received and loaded into the RxBUF Register, Receive Character Overrun (RxCharOverrun) occurs. Since there is no need for clock control in slave mode, bit D1 in the SPI Control Register is used to log any RxCharOverrun (Figure 15 and Figure 16).

No	Parameter	Min	Units
1	DI to SK Set-up	10	ns
2	SK to D0 Valid	15	ns
3	SS to SK Set-up	.5 Tsk	ns
4	SS to D0 Valid	15	ns
5	SK to DI hold time	10	ns

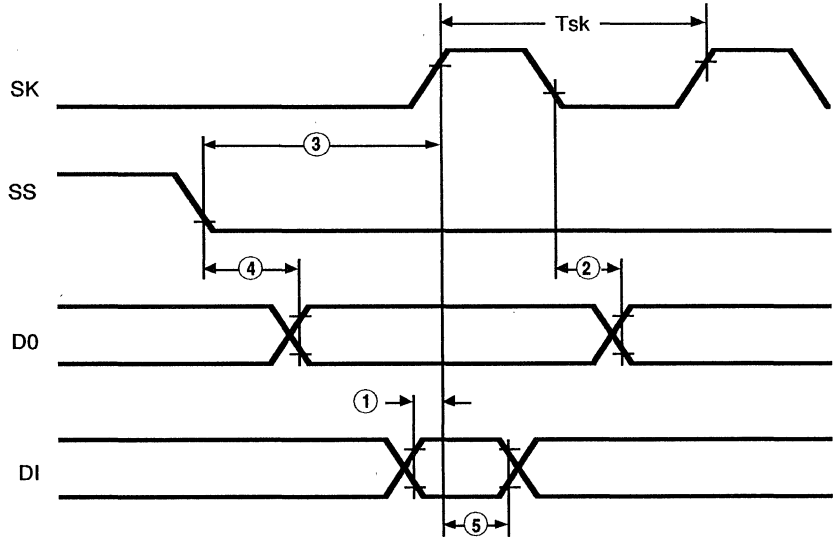


Figure 15. SPI Timing

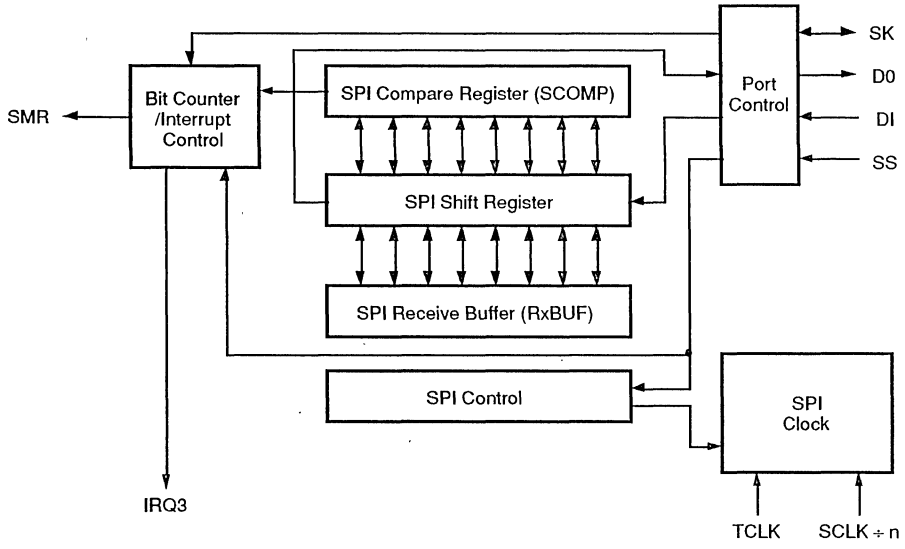


Figure 16. SPI Timing

STOP Mode Recovery Register (SMR). This register selects the clock divide value and determines the mode of STOP mode recovery (Figure 17). All bits are write only except bit 7, which is read only. Bit 7 is a flag bit that is hardware set on the condition of a STOP recovery and reset on a power-on cycle. Bit 6 controls whether a low level or high level is required from the recovery source. The recovery level must be active LOW to work with SPI. Bit 5 controls the reset delay after recovery. Bits 2, 3, and 4 of the SMR specify the source of the STOP mode recovery signal. Bit 1 determines whether the XTAL is divided by 1 or 2. A 0 in this location uses XTAL divide-by-two, and a 1 uses XTAL. The default for this bit is XTAL divide by two. Bit 0 controls the divide-by-16 prescaler of SCLK/TCLK.

SCLK/TCLK divide-by-16 select (D0). D0 of the SMR controls a divide-by-16 prescaler of SCLK/TCLK. The purpose of this control is to selectively reduce device power consumption during normal processor execution (SCLK control) and/or HALT mode (where TCLK sources the counter/timers and interrupt logic).

XTAL Clock divide-by-2 (D1). This bit determines whether the XTAL clock is divided by two or one. When this bit is set to 1, the SCLK/TCLK is equal to the XTAL clock. This option can work together with the low EMI options in PCON register to reduce the EMI noise. Maximum frequency is 4 MHz when divide-by-1 selection is active.

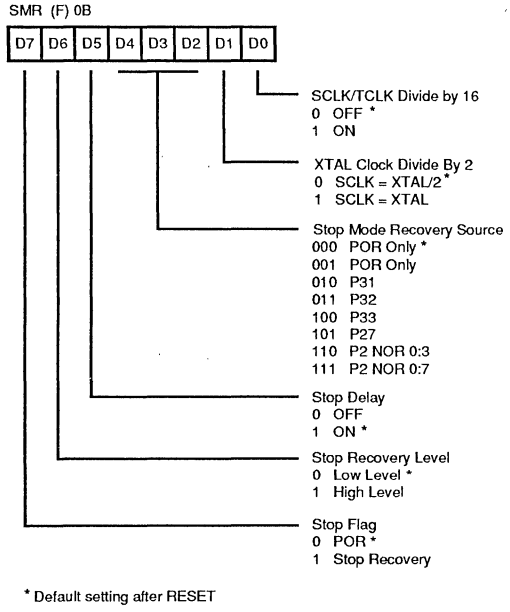


Figure 17. STOP Mode Recovery Register

STOP Mode Recovery Source (D2,D3,D4). These three bits of the SMR specify the wake-up source of the STOP Mode recovery (Figure 18 and Table 5).

Table 5. STOP Mode Recovery Source

SMR D4	SMR D3	SMR D2	Operation Description of Action
0	0	0	POR recovery only
0	0	1	POR recovery only
0	1	0	P31 transition
0	1	1	P32 transition
1	0	0	P33 transition
1	0	1	P27 transition
1	1	0	Logical NOR of Port 2 bits 0:3
1	1	1	Logical NOR of Port 2 bits 0:7

P31-P33 cannot wake up from STOP mode if the input lines are configured as analog inputs. When the SPI is enabled and the Compare feature is active, a SMR is generated upon a comparison in the SPI Shift Register and SCOMP Register, regardless of the above SMR Register settings. If SPI Compare is used to wake up the part from STOP mode, it is still possible to have one of the other STOP mode recovery sources active. Note: These other STOP mode recovery sources have to be active level low (bit D6 in SMR set to 0 if P31, P32, P33, and P27 selected, or bit D6 in SMR set to 1 if logical NOR of Port 2 is selected).

STOP Mode Recovery Delay Select (D5). This bit disables the 5 ms RESET delay after STOP Mode Recovery. The default condition of this bit is 1.

STOP Mode Recovery Level Select (D6). A 1 in this bit position indicates that a high level on any one of the recovery sources wakes the device from STOP mode. A 0 indicates low level recovery. The default is 0 on POR (Figure 18).

Cold or Warm Start (D7). This bit is set by the device upon entering STOP mode. It is active high, and is 0 (cold) on POR/WDT RESET. This bit is READ only. It is used to distinguish between cold or warm start.

Watch Dog Timer Mode Register (WDTMR). The WDT is a retriggerable one-shot timer that resets the Z8 if it reaches its terminal count. The WDT is initially enabled by executing the WDT instruction and retriggered on subsequent executions of the WDT instruction. The timer circuit is driven by an on- board RC oscillator or external clock source. The POR clock source is selected with bit 4 of the WDTMR.

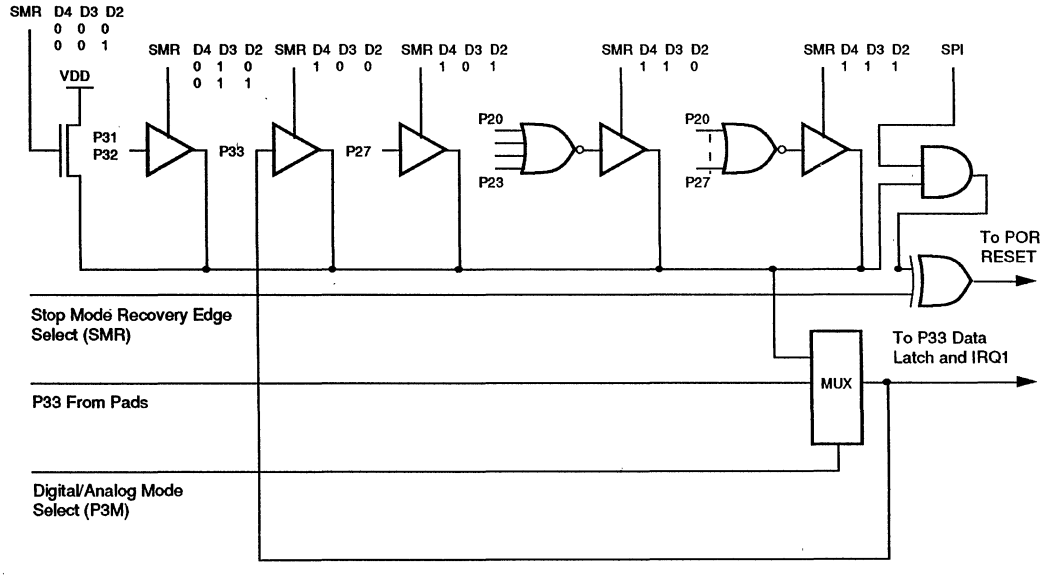


Figure 18. STOP Mode Recovery Source

Bits 0 and 1 control a tap circuit that determines the timeout period. Bit 2 determines whether the WDT is active during HALT and bit 3 determines WDT activity during STOP. If bits 3 and 4 of this register are both set to 1, the WDT is only driven by the external clock during STOP mode. This feature makes it possible to wake up from STOP mode from an internal source. Bits 5 through 7 of the WDTMR are reserved (Figure 19). This register is accessible only during the first 64 processor cycles (128 XTAL clocks) from the execution of the first instruction after Power-On-Reset, Watch Dog Reset or a STOP Mode Recovery (Figure 20). After this point, the register cannot be modified by any means, intentional or otherwise. The WDTMR cannot be read and is located in bank F of the Expanded Register Group at address location 0FH. It is organized as follows:

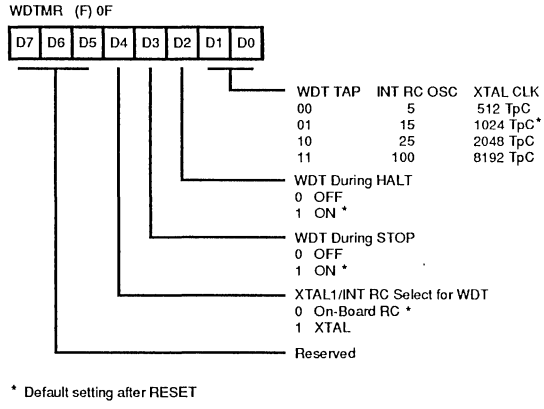


Figure 19. Watchdog Timer Mode Register

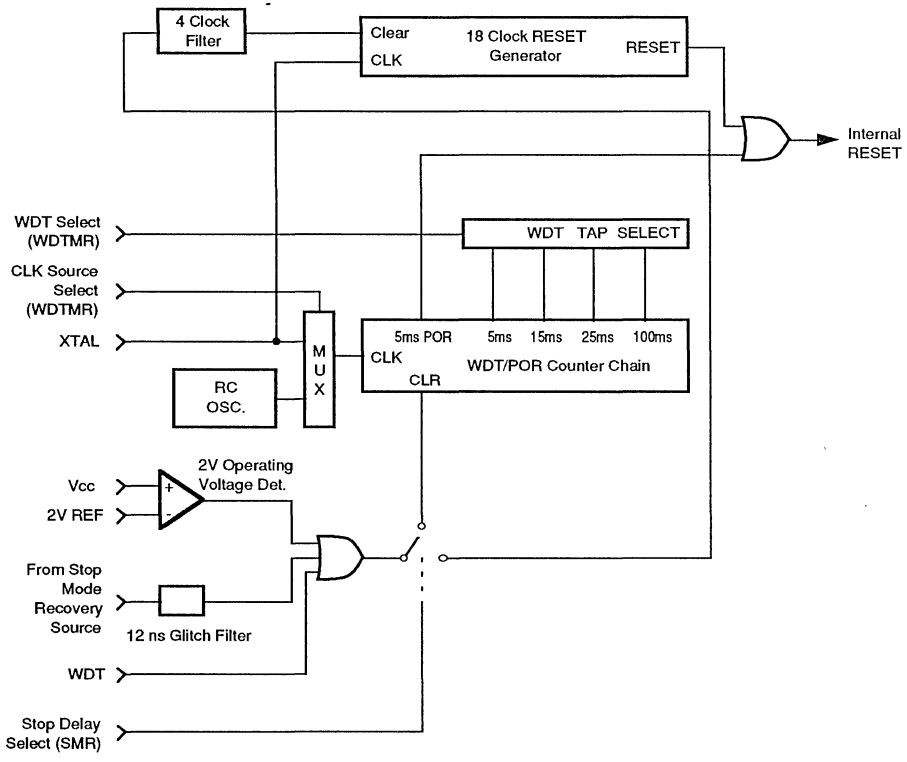


Figure 20. Resets and WDT

FUNCTIONAL DESCRIPTION (Continued)

WDT Time Select (D1,D0). Selects the WDT time-out period. It is configured as shown in Table 6.

Table 6. WDT Time Select

D1	D0	Timeout of internal RC OSC	Timeout of XTAL clock
0	0	5 ms min	512TpC
0	1	15 ms min	1024TpC
1	0	25 ms min	2048TpC
1	1	100 ms min	8192TpC

Notes:
The default on a WDT initiated RESET is 15 ms.
See Figures 50 through 53 for details.

WDT During HALT (D2). This bit determines whether or not the WDT is active during HALT mode. A 1 indicates active during HALT. The default is 1.

WDT During STOP (D3). This bit determines whether or not the WDT is active during STOP mode. Since XTAL clock is stopped during STOP mode, unless as specified below, the on-board RC has to be selected as the clock source to the POR counter. A 1 indicates active during STOP. The default is 1. If bits D3 and D4 are both set to 1, the WDT only, is driven by the external clock during STOP mode.

On-Board, Power-On-Reset RC or External XTAL1 Oscillator Select (D4). This bit determines which oscillator source is used to clock the internal POR and WDT counter chain. If the bit is a 1, the internal RC oscillator is bypassed and the POR and WDT clock source is driven from the external pin, XTAL1. The default configuration of this bit is 0, which selects the RC oscillator.

V_{CC} Voltage Comparator. An on-board Voltage Comparator checks that V_{CC} is at the required level to ensure correct operation of the device. Reset is globally driven if V_{CC} is below the specified voltage (typically 2.1V).

Brown-Out Protection (V_{BO}). The brown-out trip voltage (V_{BO}) will be less than 3 volts and above 1.4 volts under the following conditions.

Maximum (V_{BO}) Conditions:

Case 1 T_A = -40°, +105°C, Internal Clock Frequency equal or less than 1 MHz

Case 2 T_A = -40°, +85°C, Internal Clock Frequency equal or less than 2 MHz

Note:
The internal clock frequency is one half the external clock frequency, unless the device is in low EMI mode.

The device functions normally at or above 3.0V under all conditions. Below 3.0V, the device functions normally until the Brown-Out Protection trip point (V_{BO}) is reached, for the temperatures and operating frequencies in cases 1 and 2 above. The device is guaranteed to function normally at supply voltages above the brown-out trip point. The actual brown-out trip point is a function of temperature and process parameters (Figure 21).

ROM Protect. ROM protect is mask-programmable. It is selected by the customer at the time the ROM code is submitted. The selection of ROM protect disables the LDC and LDCI instructions.

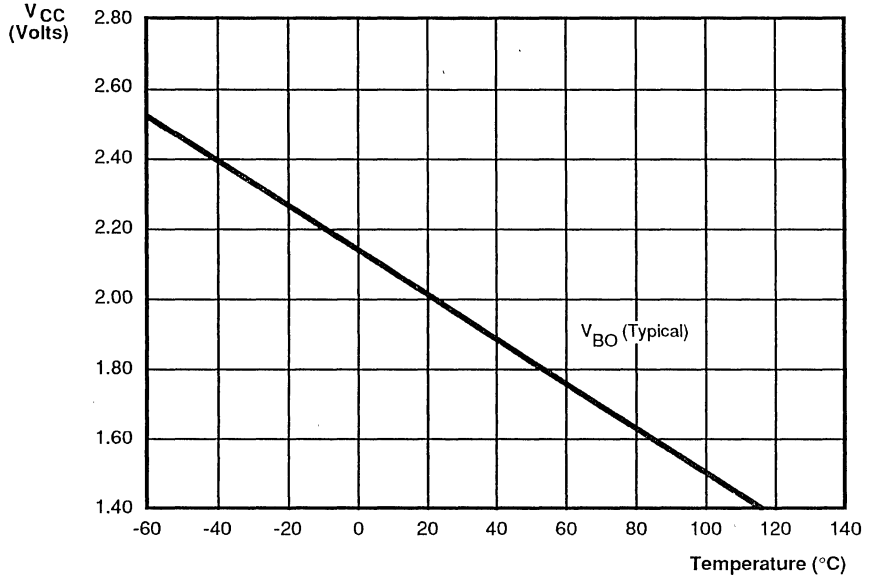


Figure 21. Typical Z86C06 V_{BO} Voltage vs Temperature

ABSOLUTE MAXIMUM RATINGS

Symbol	Description	Min	Max	Units
V_{CC}	Supply Voltage*	-0.3	+7.0	V
T_{STG}	Storage Temp	-65	+150	C
T_A	Oper Ambient Temp	†		C

Notes:
* Voltage on all pins with respect to GND.
† See Ordering Information

Stress greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended period may affect device reliability.

STANDARD TEST CONDITIONS

The characteristics listed below apply for standard test conditions as noted. All voltages are referenced to ground. Positive current flows into the referenced pin (Figure 22).

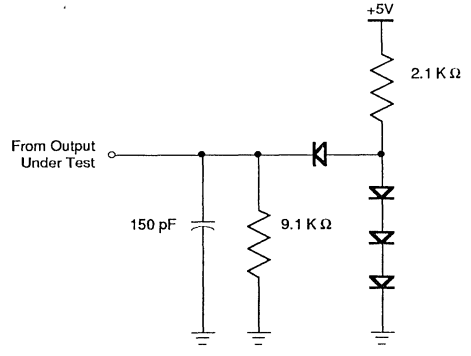


Figure 22. Test Load Configuration

DC ELECTRICAL CHARACTERISTICS

Symbol	Parameter	V_{CC} Note [3]	T_A = 0°C to 70°C Min	Max	T_A = -40°C to 105°C Min	Max	Typical @ 25°C	Units	Conditions	Notes
	Max Input Voltage	3.3V		12		12		V	$I_{IN} \leq 250\ \mu A$	
		5.0V		12		12		V	$I_{IN} \leq 250\ \mu A$	
V_{CH}	Clock Input High Voltage	3.3V	0.9 V_{CC}	V_{CC}+0.3	0.9 V_{CC}	V_{CC}+0.3	2.4	V	Driven by External Clock Generator	
		5.0V	0.9 V_{CC}	V_{CC}+0.3	0.9 V_{CC}	V_{CC}+0.3	3.9	V	Driven by External Clock Generator	
V_{CL}	Clock Input Low Voltage	3.3V	V_{SS}-0.3	0.2 V_{CC}	V_{SS}-0.3	0.2 V_{CC}	1.6	V	Driven by External Clock Generator	
		5.0V	V_{SS}-0.3	0.2 V_{CC}	V_{SS}-0.3	0.2 V_{CC}	2.7	V	Driven by External Clock Generator	
V_{IH}	Input High Voltage	3.3V	0.7 V_{CC}	V_{CC}+0.3	0.7 V_{CC}	V_{CC}+0.3	1.8	V		
		5.0V	0.7 V_{CC}	V_{CC}+0.3	0.7 V_{CC}	V_{CC}+0.3	2.8	V		
V_{IL}	Input Low Voltage	3.3V	V_{SS}-0.3	0.2 V_{CC}	V_{SS}-0.3	0.2 V_{CC}	1.0	V		
		5.0V	V_{SS}-0.3	0.2 V_{CC}	V_{SS}-0.3	0.2 V_{CC}	1.5	V		
V_{OH}	Output High Voltage	3.3V	V_{CC}-0.4		V_{CC}-0.4		3.1	V	I_{OH} = -2.0 mA	
		5.0V	V_{CC}-0.4		V_{CC}-0.4		4.8	V	I_{OH} = -2.0 mA	
V_{OL1}	Output Low Voltage	3.3V		0.8		0.8	0.2	V	I_{OL} = +4.0 mA	
		5.0V		0.4		0.4	0.1	V	I_{OL} = +4.0 mA	
V_{OL2}	Output Low Voltage	3.3V		1.0		1.0	0.4	V	I_{OL} = 6 mA, 3 Pin Max	
		5.0V		1.0		1.0	0.5	V	I_{OL} = +12 mA, 3 Pin Max	
V_{OFFSET}	Comparator Input Offset Voltage	3.3V		25		25	10	mV		
		5.0V		25		25	10	mV		
I_{IL}	Input Leakage (Input bias current of comparator)	3.3V	-1.0	1.0	-1.0	1.0		µA	V_{IN} = 0V, V_{CC}	
		5.0V	-1.0	1.0	-1.0	1.0		µA	V_{IN} = 0V, V_{CC}	
I_{OL}	Output Leakage	3.3V	-1.0	1.0	-1.0	1.0		µA	V_{IN} = 0V, V_{CC}	
		5.0V	-1.0	1.0	-1.0	1.0		µA	V_{IN} = 0V, V_{CC}	
I_{CC}	Supply Current	3.3V		6		6	3.0	mA	@ 8 MHz	[4,5]
		5.0V		11.0		11.0	6.0	mA	@ 8 MHz	[4,5]
		3.3V		8.0		8.0	4.5	mA	@ 12 MHz	[4,5]
		5.0V		15		15	9.0	mA	@ 12 MHz	[4,5]

Symbol	Parameter	V_{CC} Note [3]	T_A = 0°C to 70°C		T_A = -40°C to 105°C		Typical @ 25°C	Units	Conditions	Notes
			Min	Max	Min	Max				
I_{CC1}	Standby Current	3.3V		3.0		3.0	1.3	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 8 MHz	[4, 5]
		5.0V		5		5	3.0	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 8 MHz	[4, 5]
		3.3V		4.5		4.5	2.0	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 12 MHz	[4, 5]
		5.0V		7.0		7.0	4.0	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 12 MHz	[4, 5]
		3.3V		1.4		1.4	0.7	mA	Clock Divide by 16 @ 8 MHz	[4, 5]
		5.0V		3.5		3.5	2.0	mA	Clock Divide by 16 @ 8 MHz	[4, 5]
		3.3V		2.0		2.0	1.0	mA	Clock Divide by 16 @ 12 MHz	[4, 5]
		5.0V		4.5		4.5	2.5	mA	Clock Divide by 16 @ 12 MHz	[4, 5]
I_{CC2}	Standby Current	3.3V		10		20	1.0	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is not Running	[6]
		5.0V		10		20	3.0	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is not Running	[6]
		3.3V					TBD	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is Running	[6]
		5.0V		TBD		TBD	200	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is Running	[6]
I_{ALL}	Auto Latch Low Current	3.3V		7.0		14.0	4.0	µA	$0V < V_{IN} < V_{CC}$	
		5.0V		20.0		30.0	10	µA	$0V < V_{IN} < V_{CC}$	
I_{ALH}	Auto Latch High Current	3.3V		-4.0		-8.0	-2.0	µA	$0V < V_{IN} < V_{CC}$	
		5.0V		-9.0		-16.0	-5.0	µA	$0V < V_{IN} < V_{CC}$	
T_{POR}	Power On Reset	3.3V	7	24	6	25	13	ms		
		5.0V	3	13	2	14	7	ms		
V_{BO}	V_{CC} Brown Out Voltage		1.50	2.65	1.2	2.95	2.1	V	2 MHz max Ext. CLK Freq.	[3]

Notes:

[1] I_{CC1}

	Typ	Max	Unit	Freq
Clock Driven on Crystal	3.0	5.0	mA	8 MHz
or Ceramic Resonator	0.3	5.0	mA	8 MHz

[2] V_{SS} = 0V = GND

[3] 5.0V ± 0.5V, 3.0V ± 0.3V. The V_{BO} increases as the temperature decreases.

[4] All outputs unloaded, I/O pins floating, inputs at rail.

[5] $C_{L1} = C_{L2}$ = 100 pF

[6] Same as note [4] except inputs at V_{CC}.

AC ELECTRICAL CHARACTERISTICS

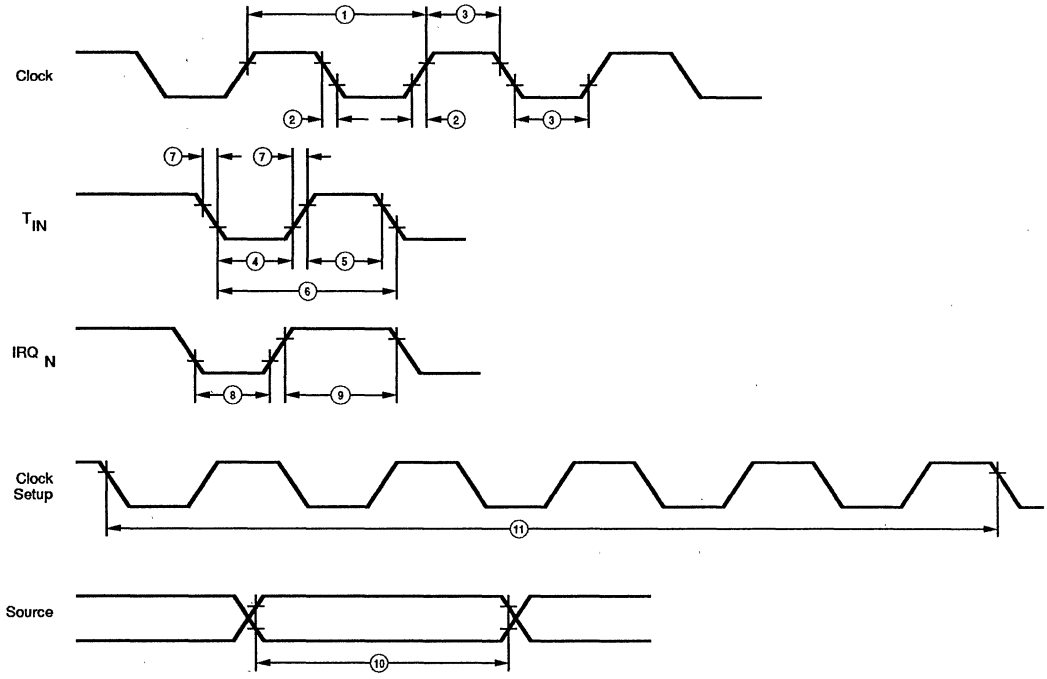


Figure 23. Additional Timing

AC ELECTRICAL CHARACTERISTICS

No	Symbol	Parameter	V_{CC} Note[3]	T_A = 0°C TO 70°C 8 MHz Min	8 MHz Max	12 MHz Min	12 MHz Max	T_A = -40°C TO 105°C 8 MHz Min	8 MHz Max	12 MHz Min	12 MHz Max	Units	Notes
1	TpC	Input Clock Period	3.3V	125	100,000	83	100,000	125	100,000	83	100,000	ns	[1]
			5.0V	125	100,000	83	100,000	125	100,000	83	100,000	ns	[1]
2	TrC,TfC	Clock Input Rise and Fall Times	3.3V		25		15		25		15	ns	[1]
			5.0V		25		15		25		15	ns	[1]
3	TwC	Input Clock Width	3.3V	37		26		37		26		ns	[1]
			5.0V	37		26		37		26		ns	[1]
4	TwTinL	Timer Input Low Width	3.3V	100		100		100		100		ns	[1]
			5.0V	70		70		70		70		ns	[1]
5	TwTinH	Timer Input High Width	3.3V	3TpC		3TpC		3TpC		3TpC			[1]
			5.0V	3TpC		3TpC		3TpC		3TpC			[1]

No	Symbol	Parameter	V_{CC} Note[3]	T_A = 0°C TO 70°C 8 MHz Min	Max	12 MHz Min	Max	T_A = -40°C TO 105°C 8 MHz Min	Max	12 MHz Min	Max	Units	Notes
6	TpTin	Timer Input Period	3.3V	8TpC		8TpC		8TpC		8TpC			[1]
			5.0V	8TpC		8TpC		8TpC		8TpC			[1]
7	TrTin, TfTin	Timer Input Rise and Fall Timer	3.3V		100		100		100		100	ns	[1]
			5.0V		100		100		100		100	ns	[1]
8	TwIL	Int. Request Input Low Time	3.3V	100		100		100		100		ns	[1,2]
			5.0V	70		70		70		70		ns	[1,2]
9	TwIH	Int. Request Input High Time	3.3V	3TpC		3TpC		3TpC		3TpC			[1,2]
			5.0V	3TpC		3TpC		3TpC		3TpC			[1,2]
10	Twsm	STOP Mode Recovery Width Spec	3.3V	12		12		12		12		ns	
			5.0V	12		12		12		12		ns	
11	Tost	Oscillator Startup Time	3.3V		5TpC		5TpC		5TpC		5TpC		Reg. [4]
			5.0V		5TpC		5TpC		5TpC		5TpC	ns	
	Twdt	Watchdog Timer Refresh Time	3.3V	15		15		12		12			[5]
			5.0V	5		5		3		3		ms	D0 = 0 [6] D1 = 0 [6]
			3.3V	30		30		25		25		ms	D0 = 1 [6]
			5.0V	16		16		12		12		ms	D1 = 0 [6]
			3.3V	60		60		50		50		ms	D0 = 0 [6]
			5.0V	25		25		30		30		ms	D1 = 1 [6]
			3.3V	250		250		200		200		ms	D0 = 1 [6]
			5.0V	120		120		100		100		ms	D1 = 1 [6]

Notes:
[1] Timing Reference uses 0.9 V_{CC} for a logic 1 and 0.1 V_{CC} for a logic 0.
[2] Interrupt request via Port 3 (P31-P33)
[3] 5.0V ± 0.5V, 3.3V ± 0.3V
[4] SMR-D5 = 0
[5] Reg. WDTMR
[6] Internal RC Oscillator only.

EXPANDED REGISTER FILE CONTROL REGISTERS

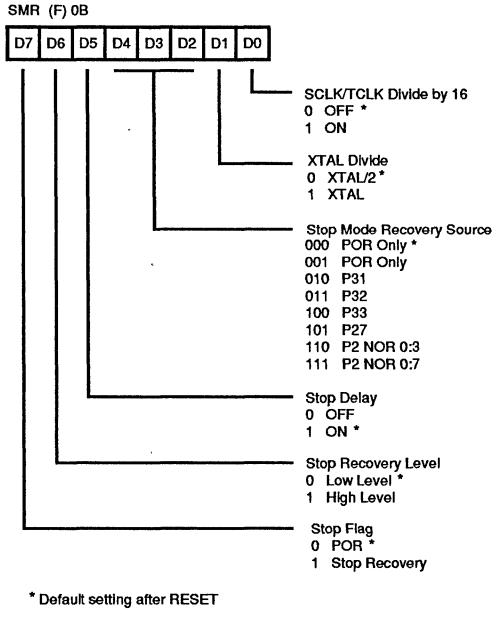


Figure 24. STOP Mode Recovery Register

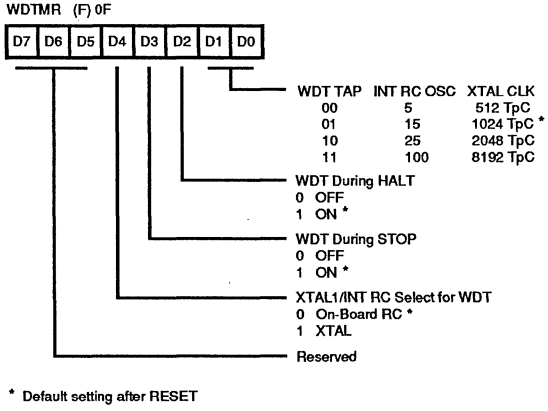


Figure 25. Watchdog Timer Mode Register

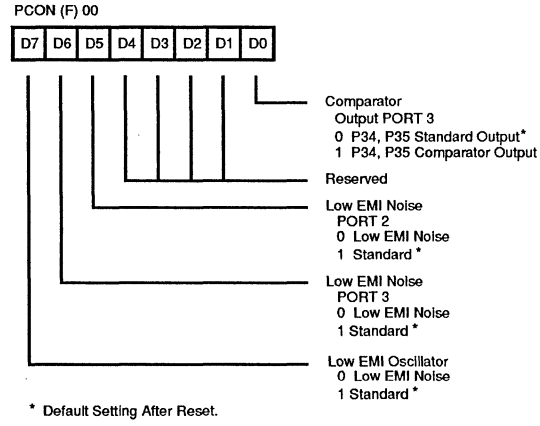


Figure 26. PORT Control Register

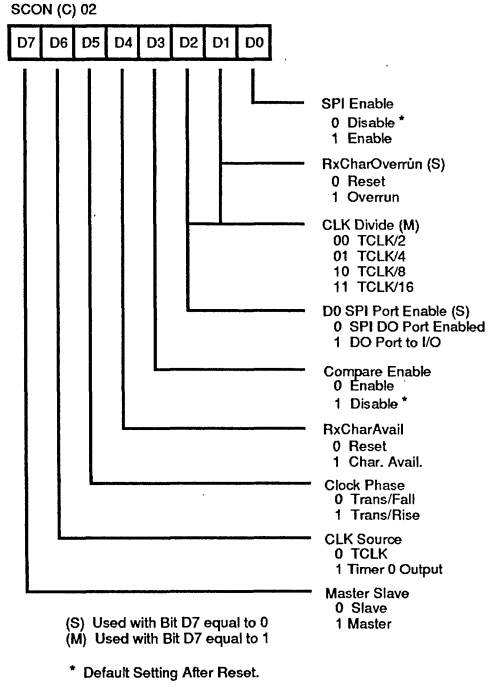


Figure 27. SPI Control Register

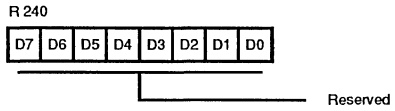


Figure 28. Reserved

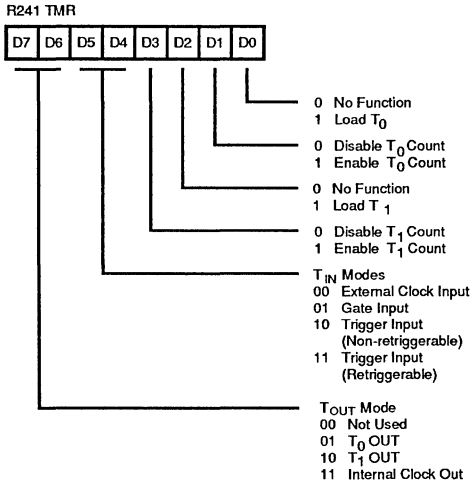


Figure 29. Timer Mode Register
($F1_H$: Read/Write)

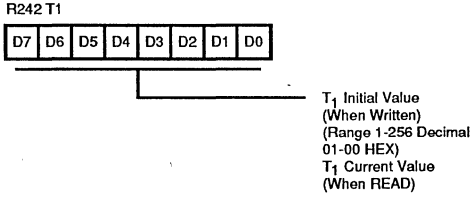


Figure 30. Counter Timer 1 Register
($F2_H$: Read/Write)

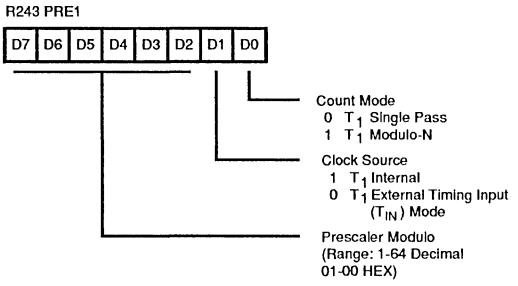


Figure 31. Prescaler 1 Register
($F3_H$: Write Only)

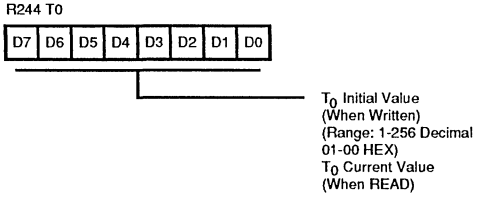


Figure 32. Counter/Timer 0 Register
($F4_H$: Read/Write)

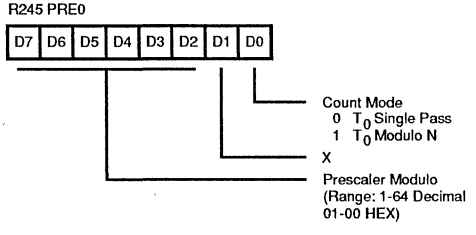


Figure 33. Prescaler 0 Register
($F5_H$: Write Only)

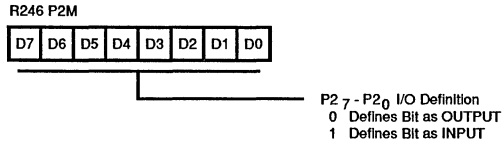


Figure 34. Port 2 Mode Register ($F6_H$: Write Only)

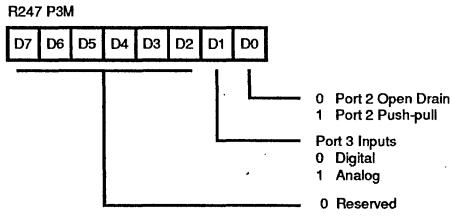


Figure 35. Port 3 Mode Register ($F7_H$: Write Only)

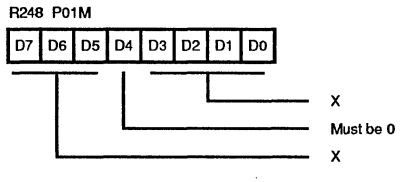


Figure 36. Port 0 and 1 Mode Register

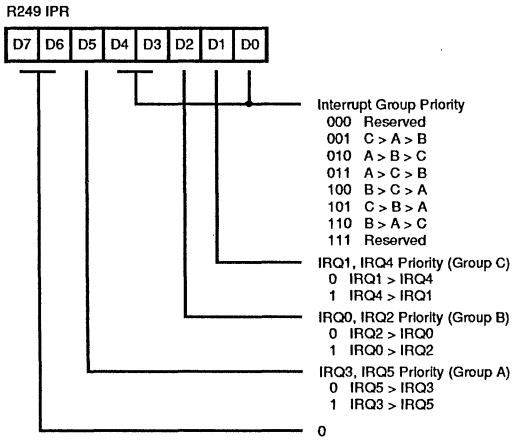


Figure 37. Interrupt Priority Register ($F9_H$: Write Only)

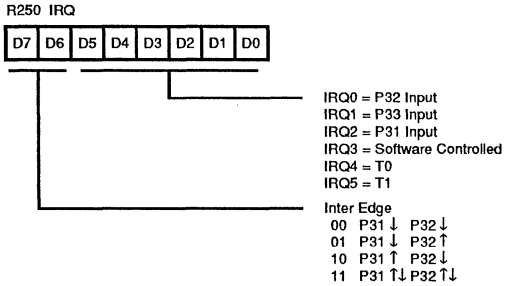


Figure 38. Interrupt Request Register (FA_H: Read/Write)

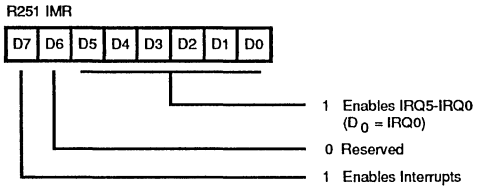


Figure 39. Interrupt Mask Register (FB_H: Read/Write)

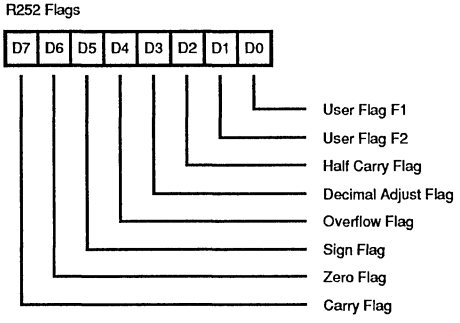


Figure 40. Flag Register (FC_H: Read/Write)

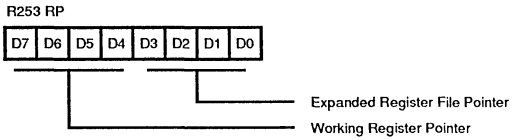


Figure 41. Register Pointer (FD_H: Read/Write)

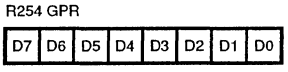


Figure 42. General Purpose Register (FE_H: Read/Write)

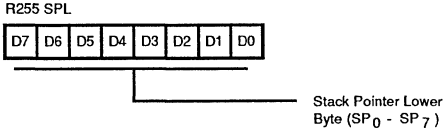


Figure 43. Stack Pointer (FF_H: Read/Write)

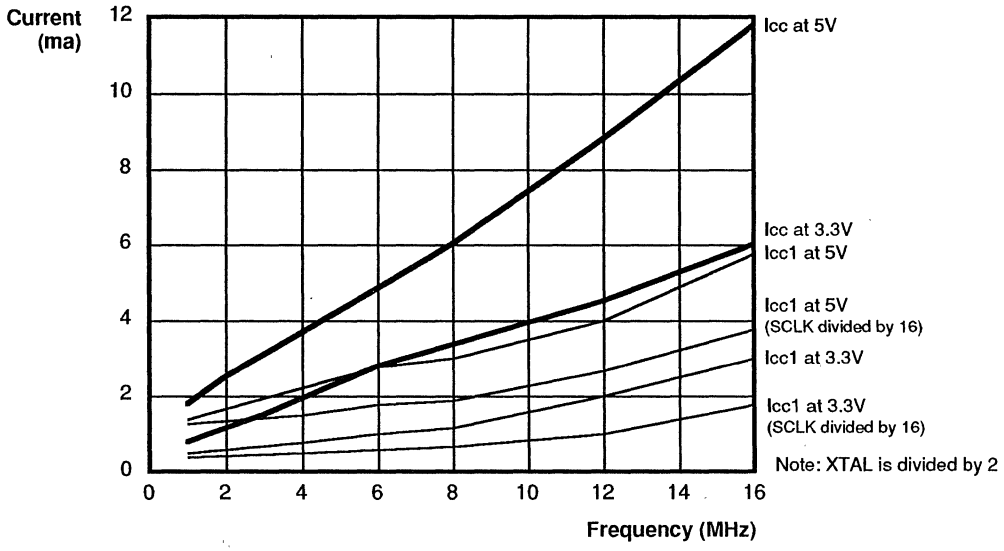


Figure 44. Typical I_{cc} vs Frequency

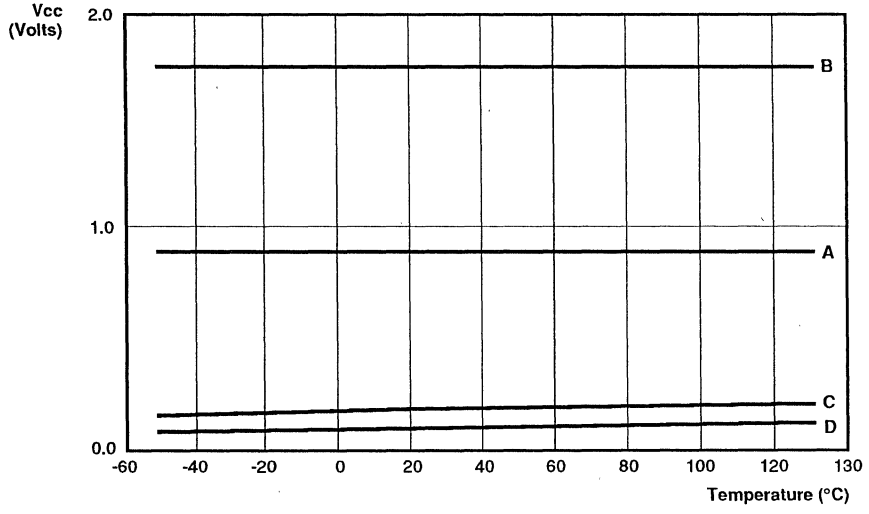


Figure 45. Typical V_{OL}, V_{IL} vs Temperature

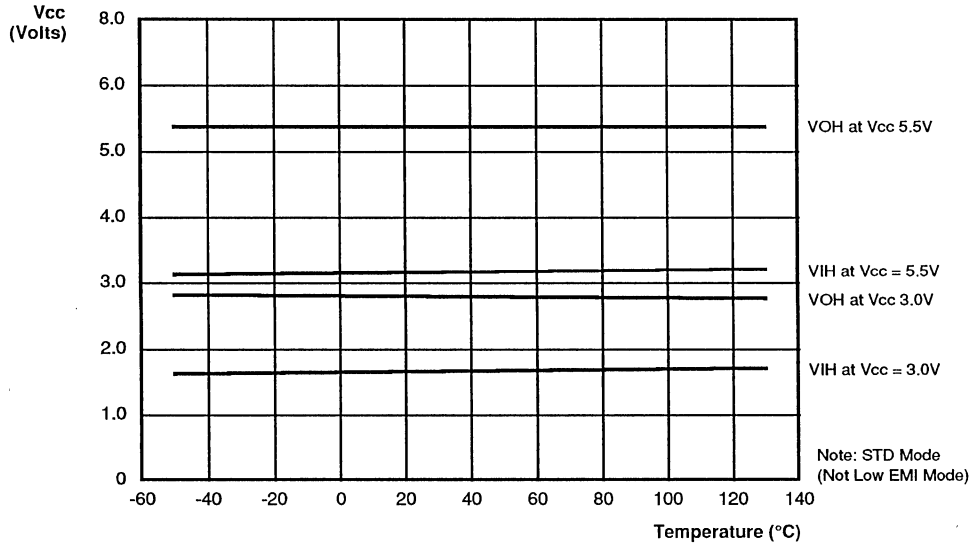


Figure 46. Typical V_{OH}, V_{IH} vs Temperature

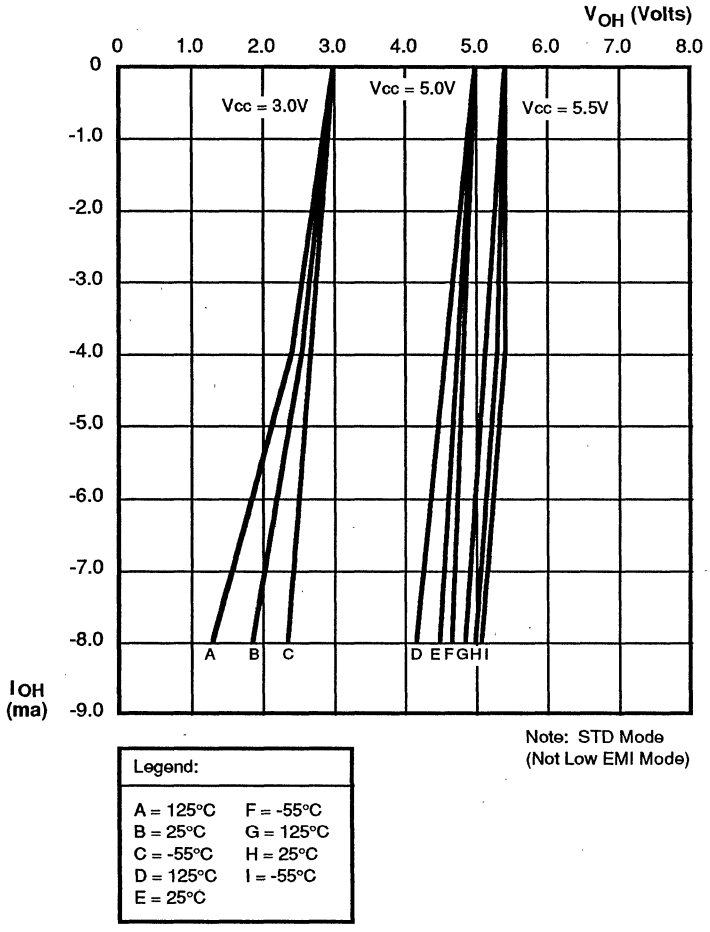


Figure 47. Typical V_{OH} vs I_{OH} Over Temperature

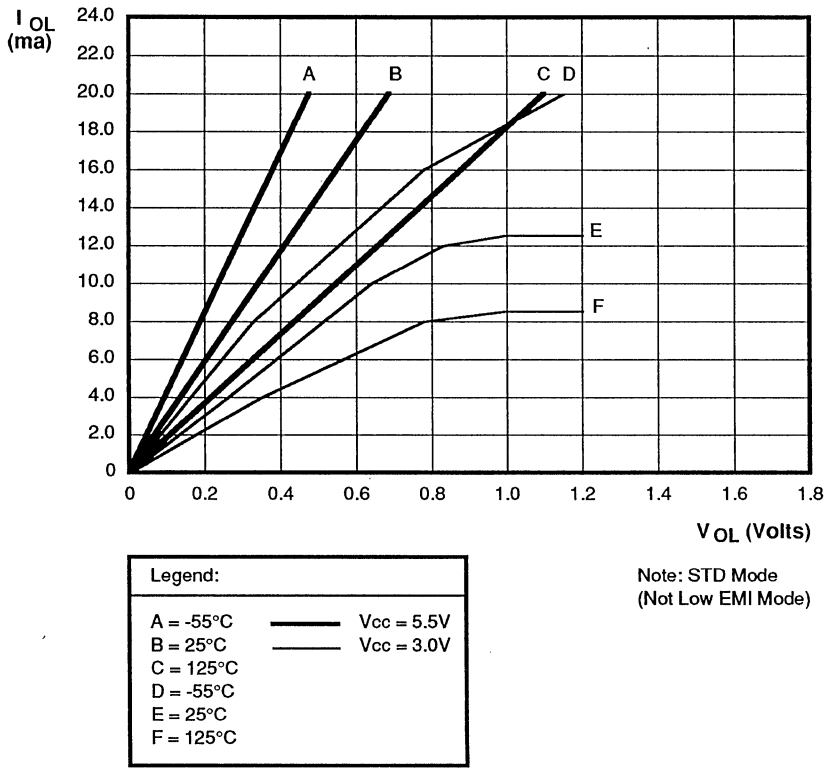


Figure 48. Typical I_{OL} vs V_{OL} Over Temperature

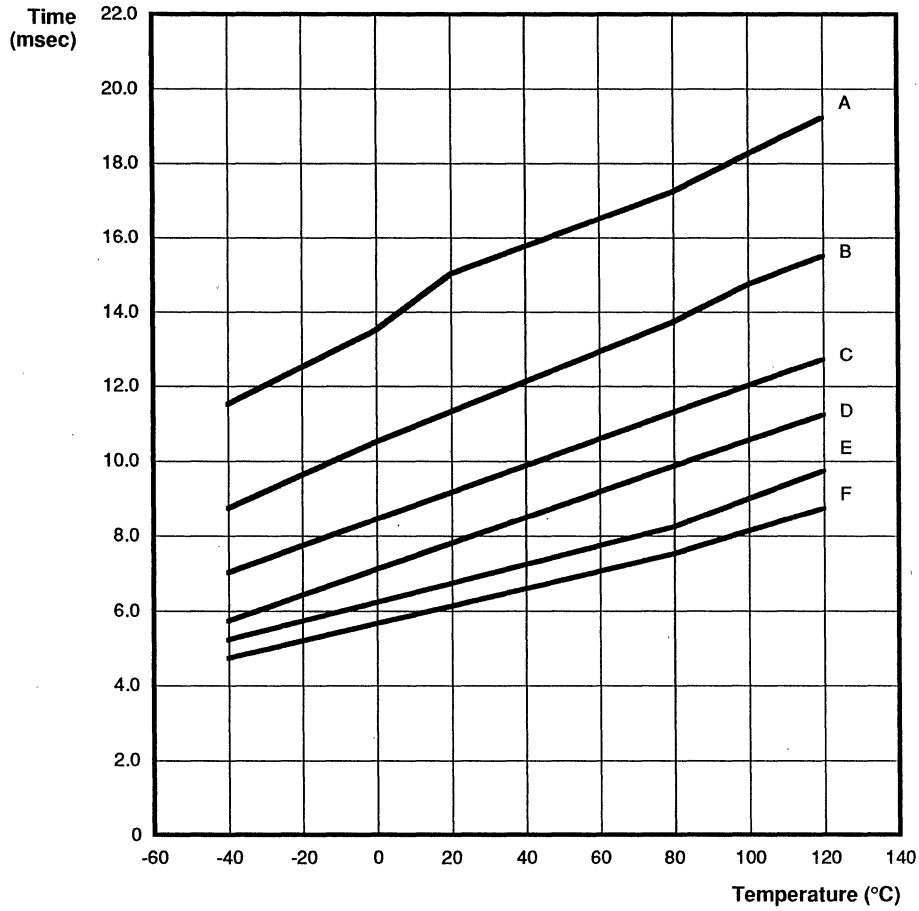


Figure 49. Typical Power-On Reset Time vs Temperature

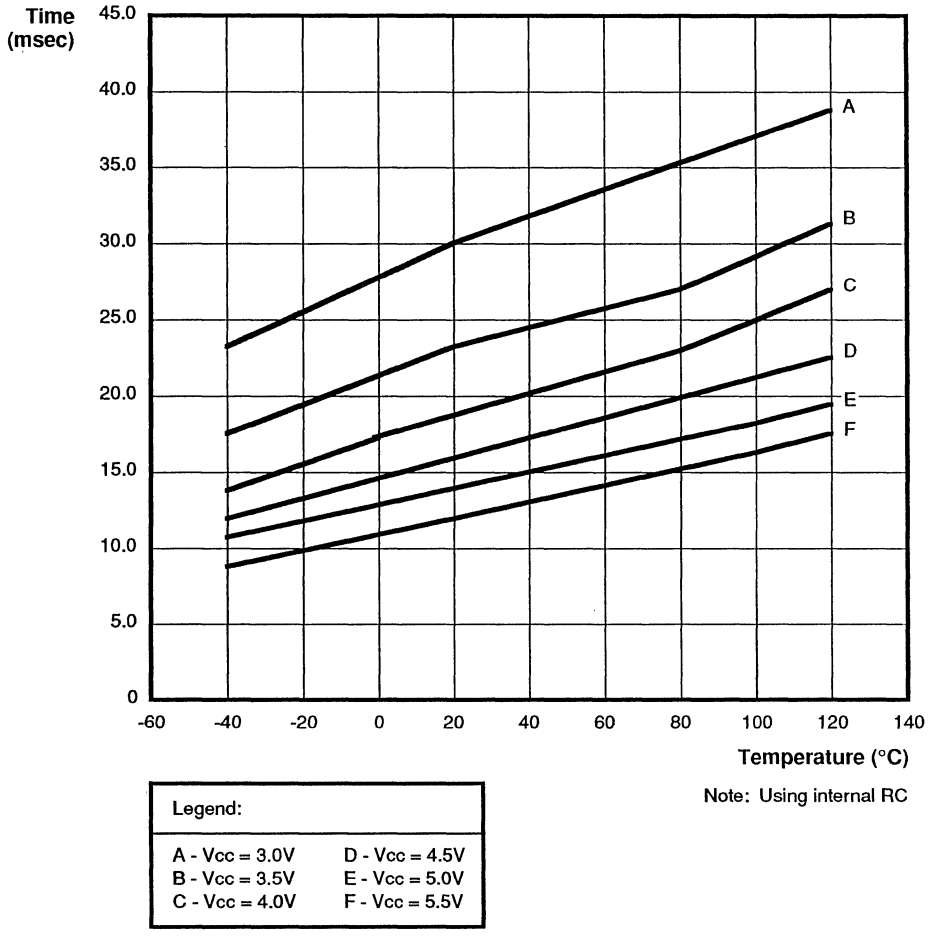
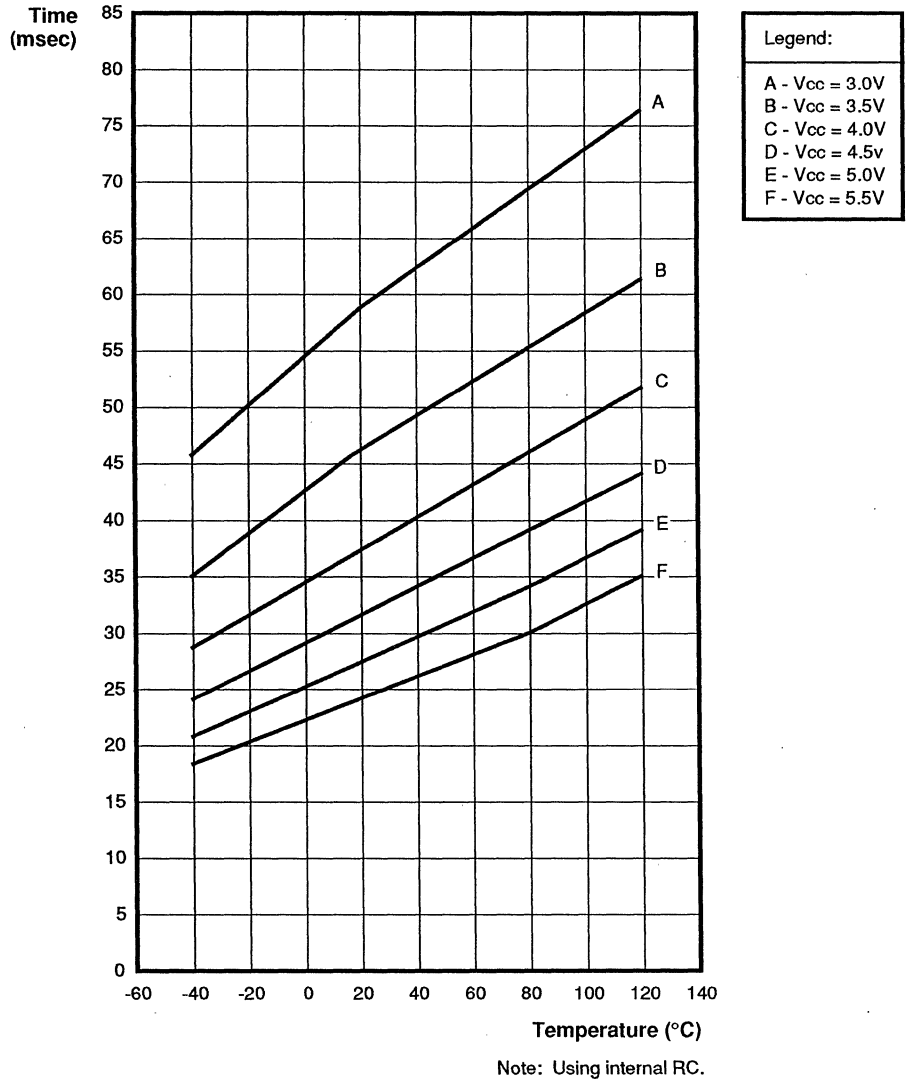


Figure 50. Typical 5 ms WDT Setting vs Temperature



Note: Using internal RC.

Figure 51. Typical 15 ms WDT Setting vs Temperature

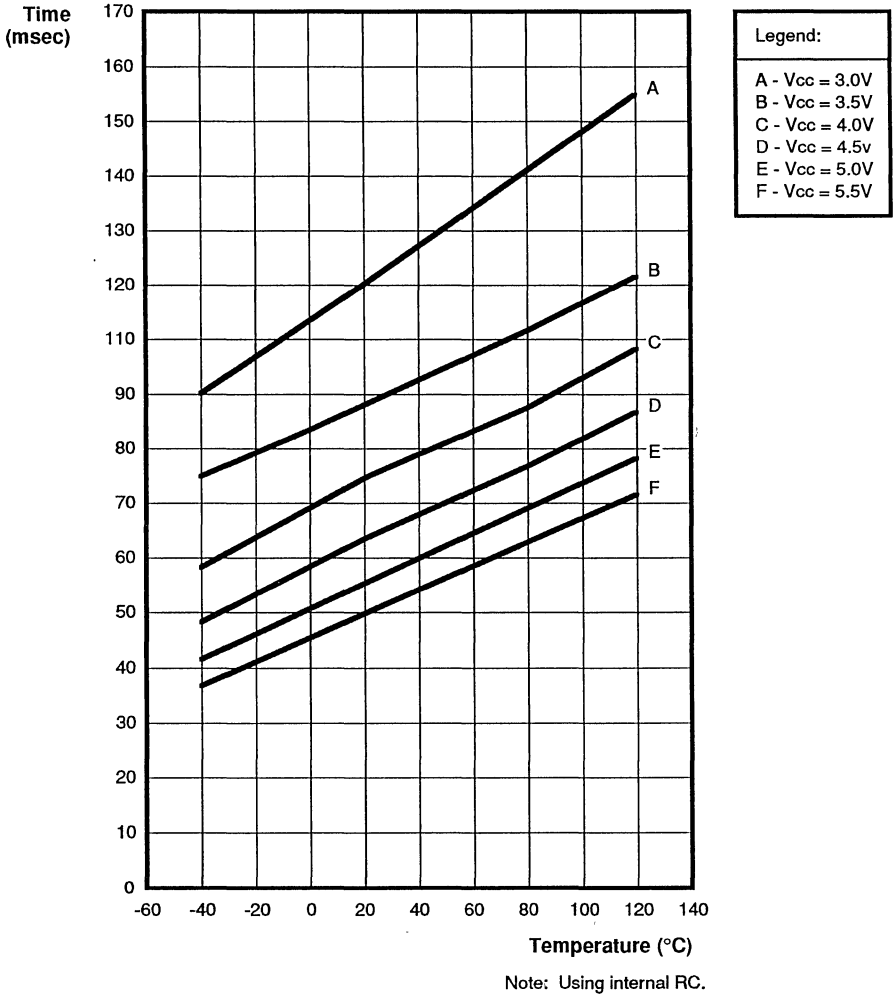


Figure 52. Typical 25 ms WDT Setting vs Temperature

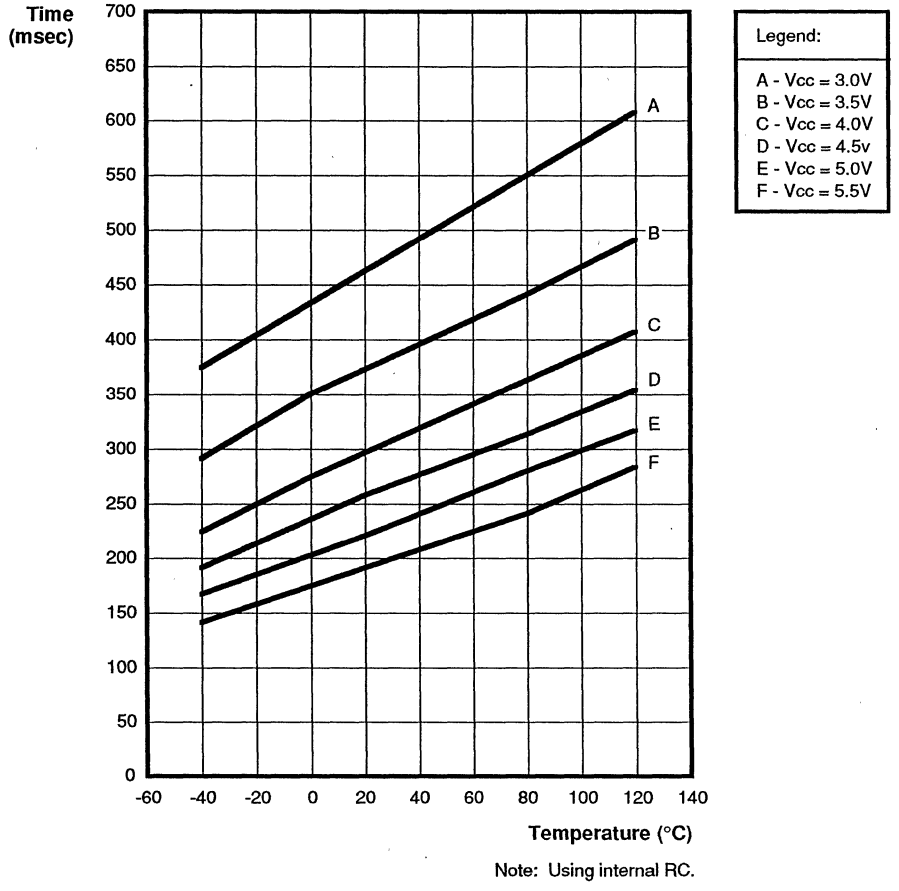
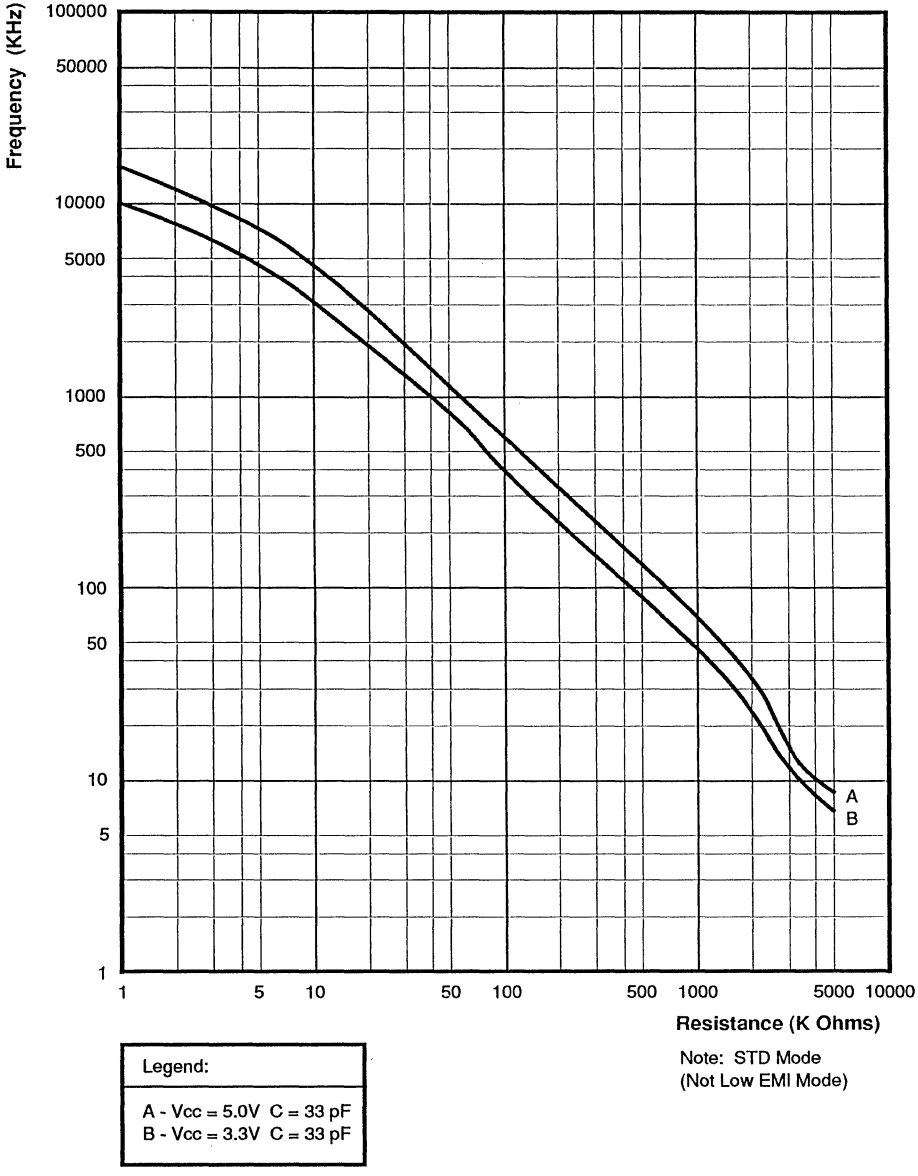
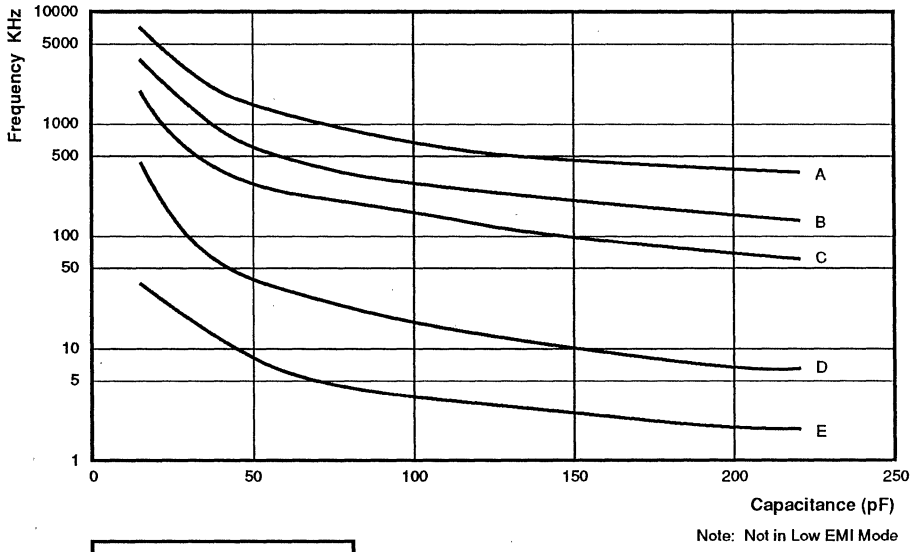


Figure 53. Typical 100 ms WDT Setting vs Temperature



Note: This chart for reference only. Each process will have a different characteristic curve.

Figure 54. Typical Frequency vs RC Resistance



Note: Not in Low EMI Mode

Legend:
A - Vcc = 5.0V R = 22 K Ohms B - Vcc = 5.0V R = 56 K Ohms C - Vcc = 5.0V R = 100 K Ohms D - Vcc = 5.0V R = 1 M Ohms C - Vcc = 5.0V R = 4 M Ohms

Figure 55. Typical RC Resistance/Capacitance vs Frequency

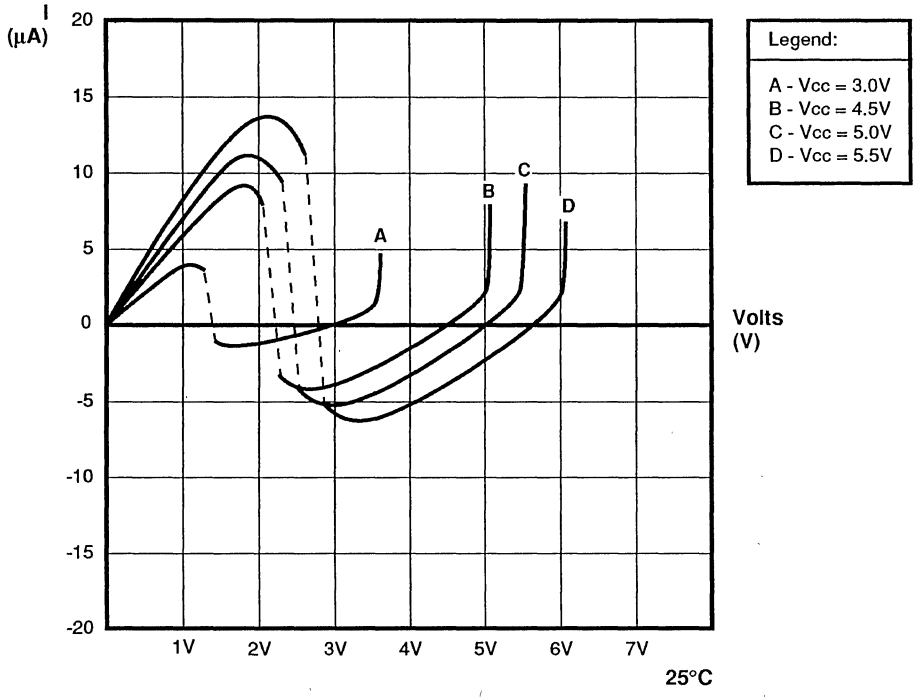


Figure 56. Auto Latch Characteristics

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

Symbol	Meaning
IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

Symbol	Meaning
dst	Destination location or contents
src	Source location or contents
cc	Condition code
@	Indirect address prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag register (Control Register 252)
RP	Register Pointer (R253)
IMR	Interrupt mask register (R251)

Flags. Control register (R252) contains the following six flags:

Symbol	Meaning
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

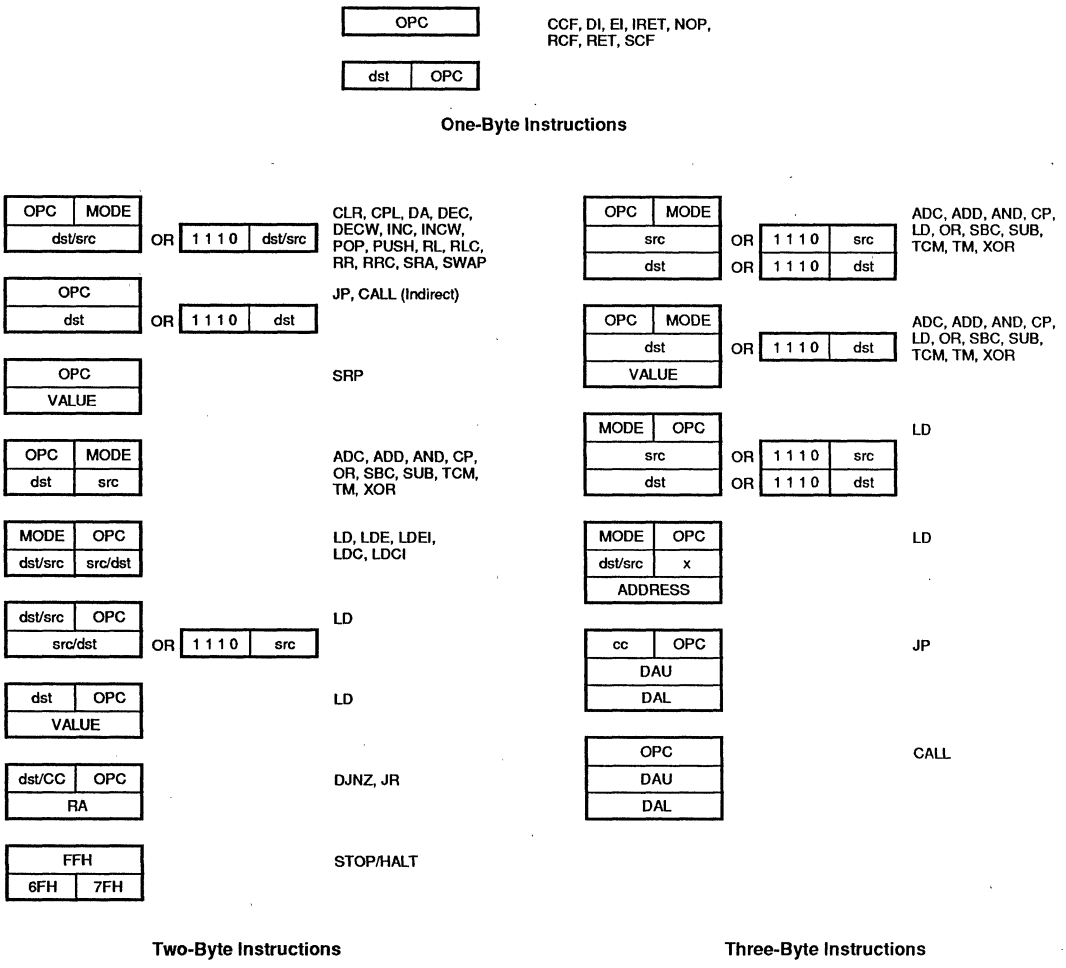
Affected flages are indicated by:

0	Clear to zero
1	Set to one
*	Set to clear according to operation
-	Unaffected
x	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000		Always True	
0111	C	Carry	C = 1
1111	NC	No Carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not Zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No Overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not Equal	Z = 0
1001	GE	Greater Than or Equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater Than	[Z OR (S XOR V)] = 0
0010	LE	Less Than or Equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned Greater Than or Equal	C = 0
0111	ULT	Unsigned Less Than	C = 1
1011	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned Less Than or Equal	(C OR Z) = 1
0000		Never True	

INSTRUCTION FORMATS



INSTRUCTION SUMMARY

Note: Assignment of a value is indicated by the symbol " ← ". For example:

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr (n)" is used to refer to bit (n) of a given operand location. For example:

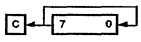
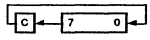
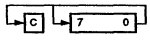
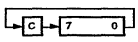
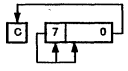
dst (7)

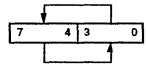
refers to bit 7 of the destination operand.

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst, src dst←dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst←dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst←dst AND src	†		5[ ]	-	*	*	0	-	-
CALL dst SP←SP - 2 @SP←PC, PC←dst	DA IRR		D6 D4	-	-	-	-	-	-
CCF C←NOT C			EF	*	-	-	-	-	-
CLR dst dst←0	R IR		B0 B1	-	-	-	-	-	-
COM dst dst←NOT dst	R IR		60 61	-	*	*	0	-	-
CP dst, src dst - src	†		A[ ]	*	*	*	*	-	-
DA dst dst←DA dst	R IR		40 41	*	*	*	X	-	-
DEC dst dst←dst - 1	R IR		00 01	-	*	*	*	-	-
DECW dst dst←dst - 1	RR IR		80 81	-	*	*	*	-	-
DI IMR(7)←0			8F	-	-	-	-	-	-
DJNZr, dst r←r - 1 if r ≠ 0 PC←PC + dst Range: +127, -128	RA		rA r = 0 - F	-	-	-	-	-	-
EI IMR(7)←1			9F	-	-	-	-	-	-
HALT			7F	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
INC dst dst←dst + 1	r R IR		rE r = 0 - F 20 21	-	*	*	*	-	-
INCW dst dst←dst + 1	RR IR		A0 A1	-	*	*	*	-	-
IRET FLAGS←@SP; SP←SP + 1 PC←@SP; SP←SP + 2; IMR(7)←1			BF	*	*	*	*	*	*
JP cc, dst if cc is true, PC←dst	DA IRR		cD c = 0 - F 30	-	-	-	-	-	-
JR cc, dst if cc is true, PC←PC + dst Range: +127, -128	RA		cB c = 0 - F	-	-	-	-	-	-
LD dst, src dst←src	r r R r X r r Ir R R R IR IR	Im R r X r Ir r R IR IM IM R	rC r8 r9 r = 0 - F C7 D7 E3 F3 E4 E5 E6 E7 F5	-	-	-	-	-	-
LDC dst, src dst←src	r	Irr	C2	-	-	-	-	-	-
LDCI dst, src dst←src r←r + 1;rr←rr + 1	Ir	Irr	C3	-	-	-	-	-	-
NOP			FF	-	-	-	-	-	-

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
OR dst, src dst←dst OR src	†		4[ ]	-	*	*	0	-	-
POP dst dst←@SP; SP←SP + 1	R IR		50 51	-	-	-	-	-	-
PUSH src SP←SP - 1; @SP←src	R IR		70 71	-	-	-	-	-	-
RCF C←0			CF	0	-	-	-	-	-
RET PC←@SP; SP←SP + 2			AF	-	-	-	-	-	-
RL dst 	R IR		90 91	*	*	*	*	-	-
RLC dst 	R IR		10 11	*	*	*	*	-	-
RR dst 	R IR		E0 E1	*	*	*	*	-	-
RRC dst 	R IR		C0 C1	*	*	*	*	-	-
SBC dst, src dst←dst←src←C	†		3[ ]	*	*	*	*	1	*
SCF C←1			DF	1	-	-	-	-	-
SRA dst 	R IR		D0 D1	*	*	*	0	-	-
SRP dst RP←src	Im		31	-	-	-	-	-	-
STOP			6F	1	-	-	-	-	-

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
SUB dst, src dst←dst←src	†		2[ ]	*	*	*	*	1	*
SWAP dst 	R IR		F0 F1	X	*	*	X	-	-
TCM dst, src (NOT dst) AND src	†		6[ ]	-	*	*	0	-	-
TM dst, src dst AND src	†		7[ ]	-	*	*	0	-	-
XOR dst, src dst←dst XOR src	†		B[ ]	-	*	*	0	-	-

† These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

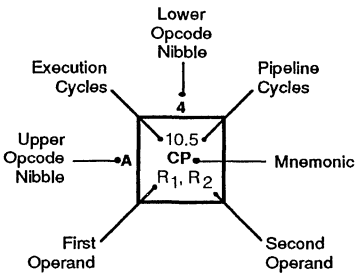
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address Mode dst	src	Lower Opcode Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

OPCODE MAP

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								6.0 **WDT**
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								6.0 **STOP**
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								7.0 **HALT**
8	10.5 **DECW** RR1	10.5 **DECW** IR1														6.1 **DI**
9	6.5 **RL** R1	6.5 **RL** IR1														6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14.0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2				10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1			20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6.5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2			3					2					3	1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_2 = Dst address
R_1 or r_2 = Src address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank are not defined.

* 2-byte instruction appears as a 3-byte instruction

PRELIMINARY PRODUCT SPECIFICATION

Z86C08

CMOS Z8® 8-BIT MICROCONTROLLER

FEATURES

- 8-bit CMOS microcontroller
- 18-pin DIP
- Low cost
- 3.0 to 5.5 Volt V_{CC} range
- Low power consumption; 50 mW (typical)
- Brown-Out protection
- Fast instruction pointer; 1 microsecond at 12 MHz
- Two standby modes - STOP and HALT
- 14 Input/Output lines
- All digital inputs at CMOS levels; Schmitt triggered
- 2 Kbytes of ROM
- 124 Bytes of RAM,
- Two programmable 8-bit counter/timers each with a 6-bit programmable prescaler.
- Six vectored, priority interrupts from six different sources
- Clock speeds 8 and 12 MHz
- Watchdog/Power-On Reset Timer
- Two Comparators with programmable interrupt polarity.
- On-chip oscillator that accepts a crystal, ceramic resonator, LC, or external clock drive.

GENERAL DESCRIPTION

The Z86C08 Microcontroller Unit (MCU) introduces a new level of sophistication to single-chip architecture. The Z86C08 is a member of the Z8 single-chip microcomputer family with 2 Kbytes of ROM and 124 bytes of general-purpose RAM. The device is housed in an 18-pin DIP, and is manufactured in CMOS technology. The Zilog Z86C08 offers all the outstanding features of the Z8 family architecture, and easy software/hardware system expansion along with low cost, low power consumption.

The Z86C08 is characterized by a flexible I/O scheme, an efficient register and address space structure, and a number of ancillary features that are useful in many consumer, industrial and advanced scientific applications.

The device applications demand powerful I/O capabilities. The Z86C08 fulfills this with 14 pins dedicated to input and output. These lines are grouped into three ports, and are configurable under software control to provide I/O, timing, and status signals.

There are two basic address spaces available to support this wide range of configurations, Program Memory, and 124 bytes of general-purpose registers.

To unburden the program from coping with real-time problems such as counting/timing and I/O data communications, the Z86C08 offers two on-chip counter/timers with a large number of user selectable modes. Also, there are two on-board comparators that can process analog signals with a common reference voltage (Figure 5).

Note: All Signals with a preceding front slash, "/", are active Low, e.g.: B/W (WORD is active Low); /B/W (BYTE is active Low, only).

GENERAL DESCRIPTION (Continued)

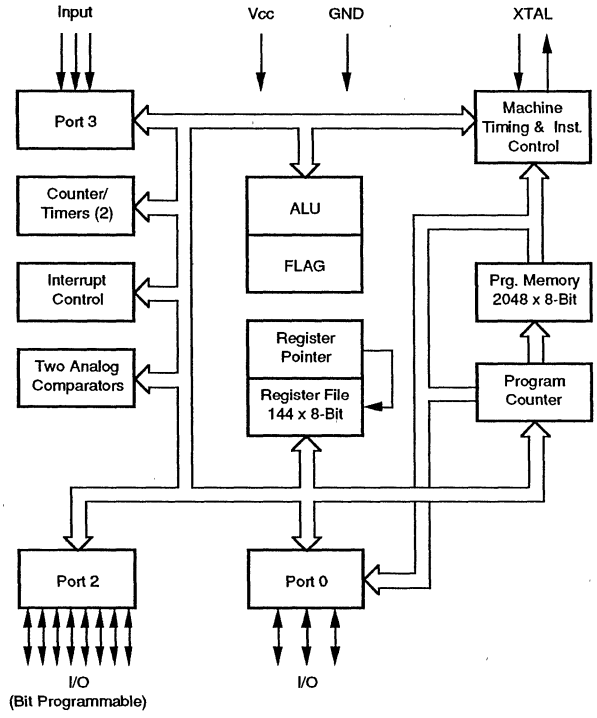


Figure 1. Functional Block Diagram

PIN DESCRIPTIONS AND SIGNAL FUNCTIONS

This Section describes the pin numbers and respective signals plus their functions (Figure 2 and Table 1).

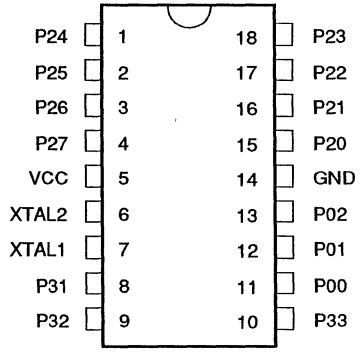


Figure 2. Pin Configuration

Table 1. Pin Identification

Pin #	Symbol	Function	Direction
1-4	P24-7	Port 2 pin 4,5,6,7	In/Output
5	V_{cc}	Power Supply, V_{DD}	Input
6	XTAL2	Crystal Oscillator Clock	Input
7	XTAL1	Crystal Oscillator Clock	Output
8	P31	Port 3 pin 1, AN1	Input
9	P32	Port 3 pin 2, AN2	Input
10	P33	Port 3 pin 3, REF	Input
11-13	P00-2	Port 0 pin 0,1,2	In/Output
14	GND	Ground, V_{SS}	Input
15-18	P20-3	Port 2 pin 0,1,2,3	In/Output

XTAL1, XTAL2. *Crystal in, Crystal Out* (time-based input and output, respectively). These pins connect a parallel-resonant crystal, LC, or an external single-phase clock (12 MHz max) to the on-chip clock oscillator and buffer.

Port 0 (P00-P02). Port 0 is a 3-bit I/O, nibble programmable, bidirectional, CMOS compatible I/O port. These 3 I/O lines can be configured under software control to be an input or output (Figure 3).

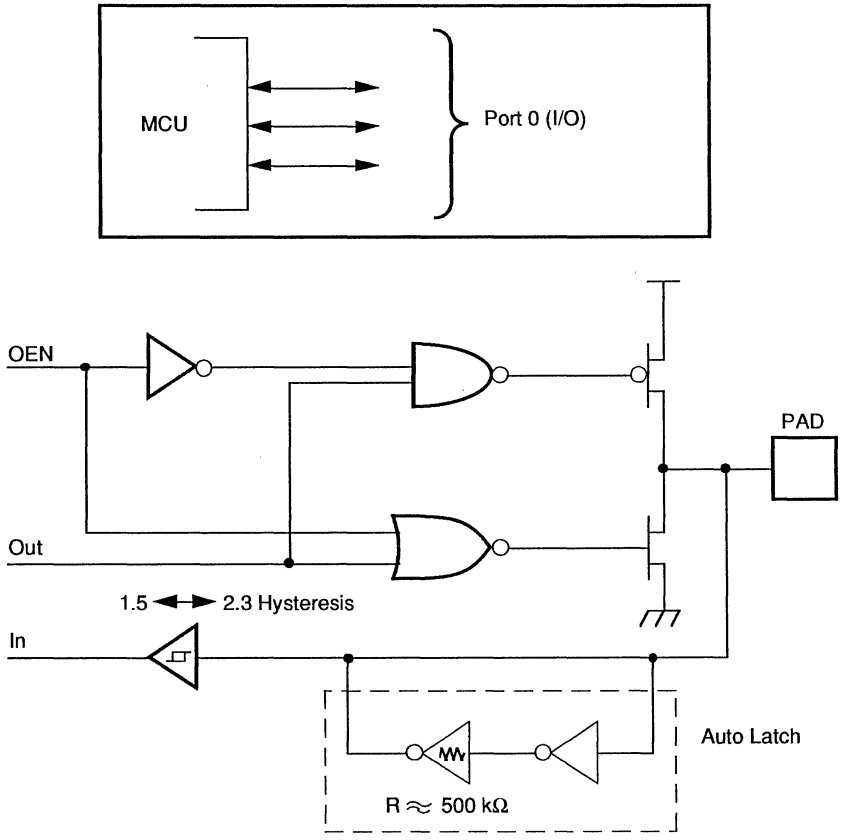


Figure 3. Port 0 Configuration

Port 2 (P20-P27). Port 2 is an 8-bit I/O, bit programmable, bidirectional, CMOS compatible I/O port. These 8 I/O lines can be configured under software control to be an input or output, independently. Bits programmed as outputs may be globally programmed as either push pull or open drain (Figure 4).

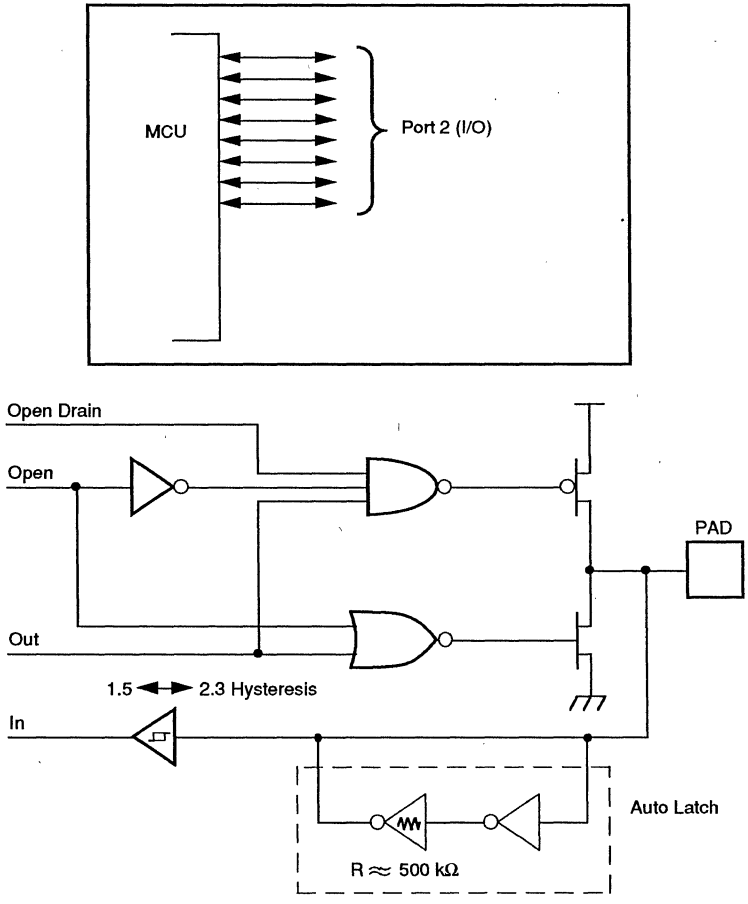


Figure 4. Port 2 Configuration

Port 3 (P31-P33). Port 3 is a 3-bit, CMOS compatible port with three fixed input (P32-P33) lines. These three input lines can be configured under software control as digital inputs or analog inputs. These three input lines can also be used as the interrupt sources IRQ0-IRQ3 and as the timer input signal (T_{IN}) (Figure 5).

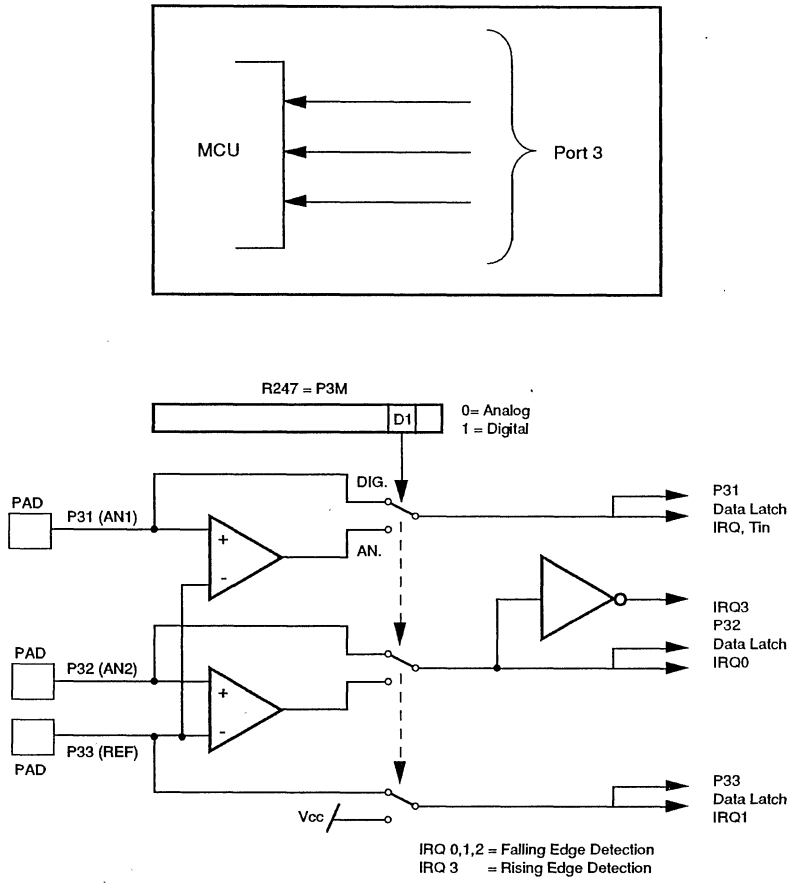


Figure 5. Port 3 Configuration

Comparator Inputs. Two analog comparators are added to Port 3 inputs for interface flexibility.

Typical applications for the on-board comparators are: Zero crossing detection, A/D conversion, voltage scaling, and threshold detection.

The dual comparator (common inverting terminal) features a single power supply which discontinues power in STOP Mode. The common voltage range is 0-4V; the power supply and common mode rejection ratios are 90dB and 60dB, respectively.

Interrupts are generated on either edge of comparator 2's output, or on the falling edge of comparator 1's output. The comparator output may be used for interrupt generation, Port 3 data inputs, or Tin through P31. Alternatively, the comparators may be disabled, freeing the reference input (P33) for use as IRQ1 and/or P33 input.

FUNCTIONAL DESCRIPTION

The Z8 MCU incorporates special functions to enhance the Z8's application in industrial, scientific research, an advanced technologies applications.

Reset. Upon power up the power-on reset circuit waits for 50 msec plus 18 crystal clocks and then starts program execution at address %000C (HEX) (Figure 6). Reference the Z86C08 control registers' Reset value (Table 2).

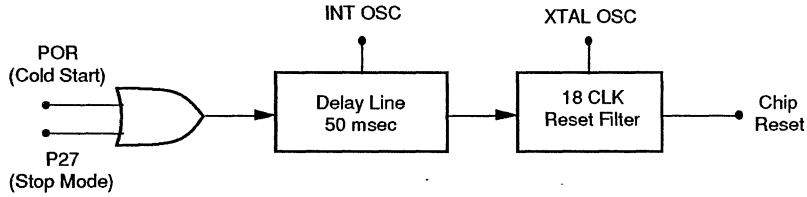


Figure 6. Internal Reset Configuration

Table 2. Z86C08 Control Registers

Addr.	Reg.	Reset Condition D7	D6	D5	D4	D3	D2	D1	D0	Comments
F1	TMR	0	0	0	0	0	0	0	0	
F2	T1	U	U	U	U	U	U	U	U	
F3	PRE1	U	U	U	U	U	U	0	0	
F4	T0	U	U	U	U	U	U	U	U	
F5	PRE0	U	U	U	U	U	U	U	0	
F6*	P2M	1	1	1	1	1	1	1	1	Inputs after reset
F7*	P3M	U	U	U	U	U	U	0	0	
F8*	P01M	U	U	U	0	U	U	0	1	
F9	IPR	U	U	U	U	U	U	U	U	
FA	IRQ	U	U	0	0	0	0	0	0	IRQ3 is used for positive edge detection
PB	IMR	0	U	U	U	U	U	U	U	
PC	FLAGS	U	U	U	U	U	U	U	U	
FD	RP	0	0	0	0	0	0	0	0	
FE	SPH	U	U	U	U	U	U	U	U	Not used, stack always internal
FF	SPL	U	U	U	U	U	U	U	U	

Note:
* A reset after a low on P27 to get out of stop mode may affect device reliability.

Program Memory. The Z86C08 can address up to 2 Kbytes of internal program memory (Figure 7). The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts. Bytes 0-2048 are on-chip mask-programmed ROM.

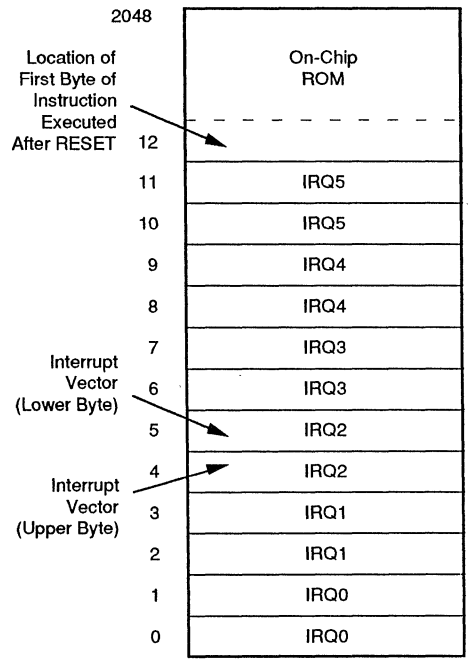


Figure 7. Program Memory Map

Register File. The Register File consists of three I/O port registers, 124 general purpose registers, and 15 control and status registers (R0-R3, R4-R127 and R241-R255, respectively - Figure 8). The Z86C08 instructions can access registers directly or indirectly via an 8-bit address field. This allows short 4-bit register addressing using the Register Pointer. In the 4-bit mode, the register file is divided into eight working register groups, each occupying 16 continuous locations. The Register Pointer (Figure 9) addresses the starting location of the active working-register group.

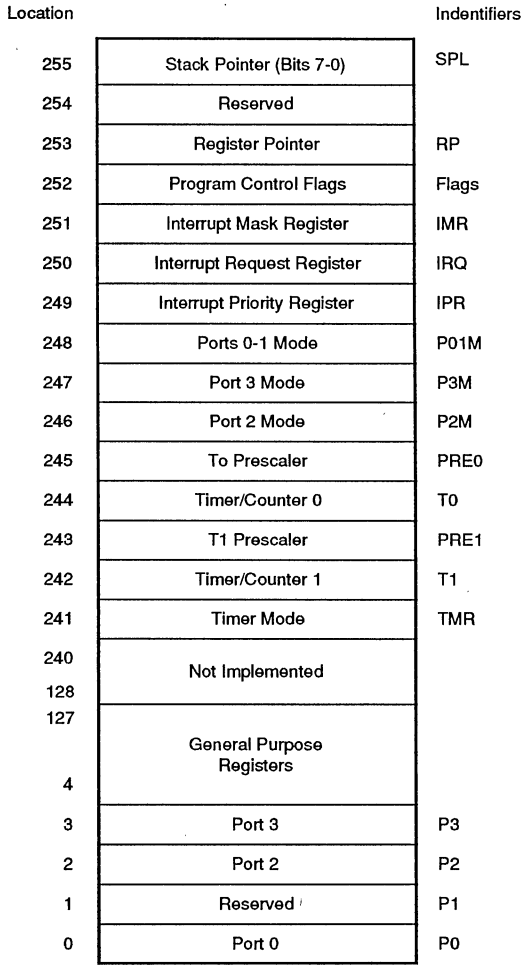


Figure 8. Register File

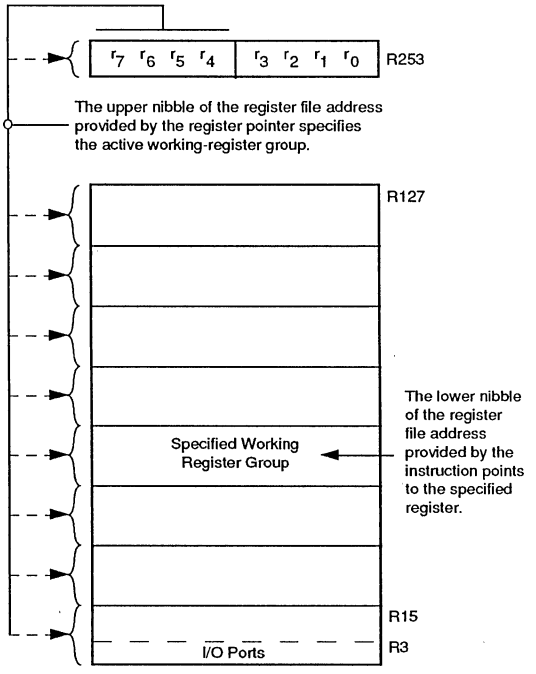


Figure 9. Register File

Stack Pointer. The Z86C08 has an 8-bit Stack Pointer (R255) used for the internal stack that resides within the 124 General-Purpose registers.

Counter/Timer. There are two 8-bit programmable counter/timers (T0 and T1), each driven by its own 6-bit programmable prescaler. The T1 prescaler can be driven by internal or external clock sources, however the T0 can be driven by the internal clock source only (Figure 10).

The 6-bit prescalers can divide the input frequency of the clock source by any integer number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When both counter and prescaler reach the end of count, a timer interrupt request IRQ4 (T0) or IRQ5 (T1) is generated.

The counter can be programmed to start, stop, restart to continue, or restart from the initial value. The counters can also be programmed to stop upon reaching zero (single pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode).

The counters, but not the prescalers are read at any time without disturbing their value or count mode. The clock source for T1 is user-definable and can be either the internal microprocessor clock divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input (P30) as an external clock, a trigger input that is retriggerable or not retriggerable, or as a gate input for the internal clock.

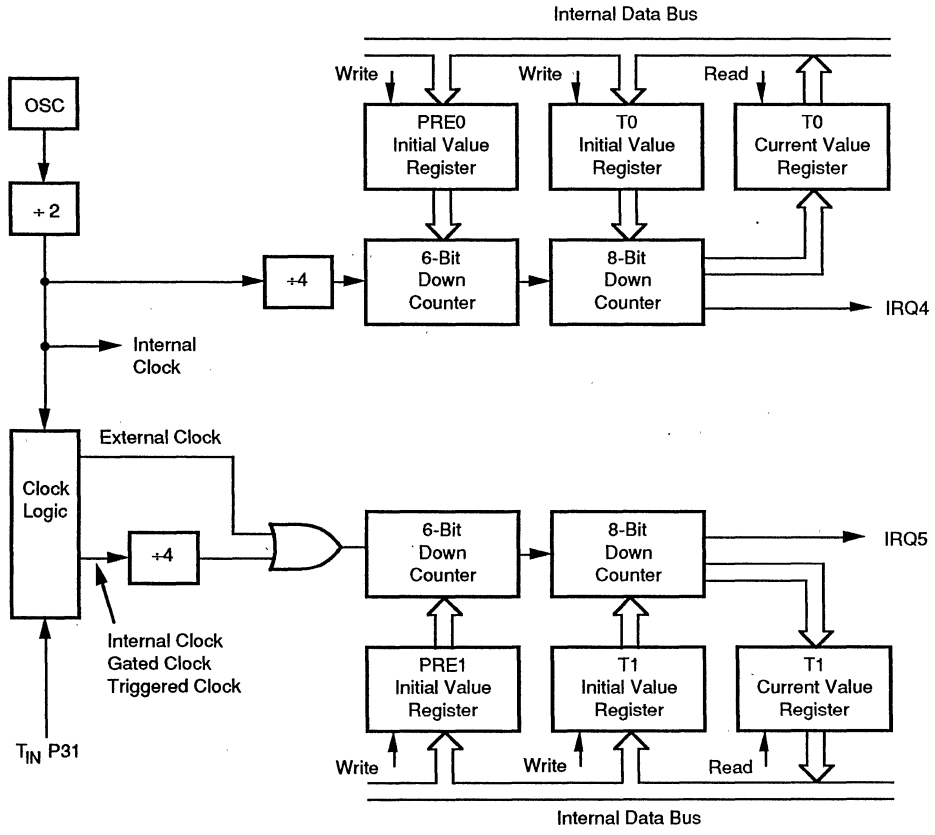


Figure 10. Counter/Timers Block Diagram

Interrupts. The Z86C08 has six interrupts from six different sources. These interrupts are maskable and prioritized (Figure 11). The six sources are divided as follows: the falling edge of P31 (AN1), P32 (AN2), P33 (REF), the rising edge of P32 (AN2), and the two counter/timers. The Interrupt Mask Register globally or individually enables or disables the six interrupt requests (Table 3).

When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register. All Z86C08 interrupts are vectored through locations in program memory. When an Interrupt machine cycle is activated, an interrupt request is granted. This disables all subsequent interrupts, saves the Program Counter and Status Flags, and then branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit starting address of the interrupt service routine for that particular interrupt request.

To accommodate polled interrupt systems, interrupt inputs are masked and the interrupt request register is polled to determine which of the interrupt requests needs service.

Table 3. Interrupt Types, Sources, and Vectors

Source	Name	Vector Location	Comments
AN2(P32)	IRQ0	0,1	External (F)Edge
REF(P33)	IRQ1	2,3	External (F)Edge
AN1(P31)	IRQ2	4,5	External (F)Edge
AN2(P32)	IRQ3	6,7	External (R)Edge
T0	IRQ4	8,9	Internal
T1	IRQ5	10,11	Internal

Notes:
F=Falling edge triggered
R=Rising edge triggered

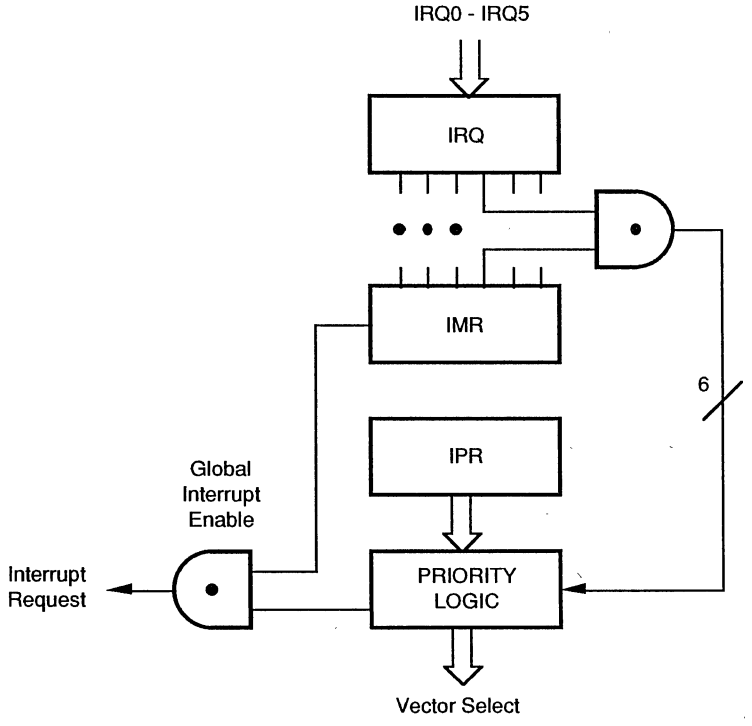


Figure 11. Interrupt Block Diagram

FUNCTIONAL DESCRIPTION (Continued)

Clock. The Z86C08 on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal, ceramic resonator, or any suitable external clock source (XTAL1 = Input, XTAL2 = Output). The crystal should be AT cut, 12 MHz max, with a series resistance (RS) less than or equal to 100 Ohms.

The crystal should be connected across XTAL1 and XTAL2 using the recommended capacitors (capacitance is between 10 pF to 250 pF which depends on the crystal manufacturer, ceramic resonator and PCB layout) from each pin to ground (Figure 12).

HALT Mode. Turns off the internal CPU clock but not the crystal oscillation. The counter/timers and external interrupts IRQ0, IRQ1, and IRQ2 remain active. The device can be recovered by interrupts, either externally or internally generated. The program execution begins at location 000C (HEX).

STOP Mode. This instruction turns off the internal clock and external crystal oscillation and reduces the standby current to 10 microamps. The STOP Mode can be released by two methods. The first method is a RESET of the device by removing VCC. The second method is if P27 is configured as an input line when the device executes the STOP instruction. A low input condition on P27 releases the STOP Mode.

Program execution under both conditions begins at location 000C (HEX). However, when P27 is used to release the STOP Mode, the I/O port mode registers are not reconfigured to their default power-on conditions. This prevents any I/O, configured as output when the STOP instruction was executed, from glitching to an unknown state. To use the P27 release approach with STOP Mode, use the following instruction:

```
OR      P2M, #80H
NOP
STOP
```

In order to enter STOP (or HALT) mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. To do this, the user must execute a NOP (opcode=FFH) immediately before the appropriate sleep instruction. i.e.:

```
FF    NOP     ; clear the pipeline
6F    STOP    ; enter STOP mode
              or
FF    NOP     ; clear the pipeline
7F    HALT    ; enter HALT mode
```

Watch Dog Timer (WDT). The Watch Dog Timer is enabled by instruction WDT. When the WDT is enabled, it cannot be stopped by the instruction. With the WDT instruction, the WDT should be refreshed once the WDT is enabled within every 15 msec; otherwise, the Z86C08 resets itself.

WDT=5F (HEX).

Opcode WDT (5FH). The first time opcode 5FH is executed, the WDT is enabled, and subsequent execution clears the WDT counter. This has to be done at least every 15 msec. Otherwise, the WDT times out and generates a reset. The generated reset is the same as a power on reset of 50 msec +18 XTAL clock cycles.

Opcode WDH (4FH). When this instruction is executed it will enable the WDT during HALT. If not, the WDT will stop when entering HALT. This instruction does not clear the counters, it just makes it possible to have the WDT function running during HALT Mode. A WDH instruction executed without executing WDT (5FH) has no effect.

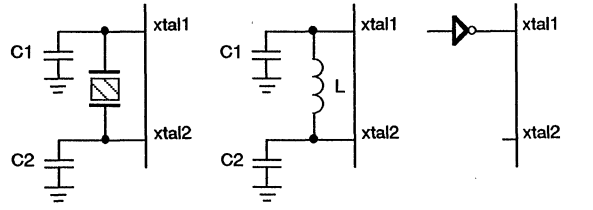


Figure 12. Oscillator Configuration

Brown-Out Protection (V_{BO}). The brown-out trip voltage (V_{BO}) is less than 3 volts and above 1.4 volts under the following conditions:

Maximum (V_{BO}) Conditions:

Case 1 T_A= -40°C, +105°C, Internal Clock Frequency equal or less than 1 MHz

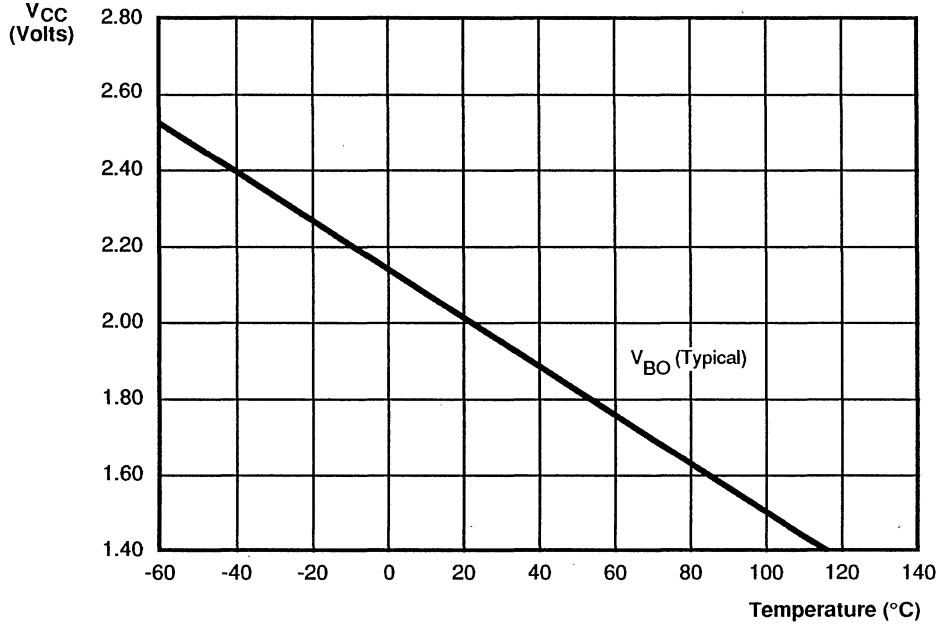
Case 2 T_A= -40°C, +85°C, Internal Clock Frequency equal or less than 2 MHz

Note: The internal clock frequency is one half the external clock frequency.

The device will function normally at or above 3.0V under all conditions. Below 3.0V, the device functions normally until the Brown-Out Protection trip point (V_{BO}) is reached. The device is guaranteed to function normally at supply voltages above the brown-out trip point for the temperatures and operating frequencies in Case 1 and Case 2 above. The actual brown-out trip point is a function of temperature and process parameters (Figure 13).

2 MHz (Typical)

Temp	-40°C	0°C	+25°C	+70°C	+105°C
V_{BO}	2.55	2.4	2.1	1.7	1.6



* Power-on Reset threshold for V_{CC} and 4 MHz V_{BO} overlap

Figure 13. Typical Z86C08 V_{BO} vs. Temperature

STANDARD TEST CONDITIONS

The characteristics listed below apply for standard test conditions as noted. All voltages are referenced to GND. Positive current flows into the referenced pin (Figure 14).

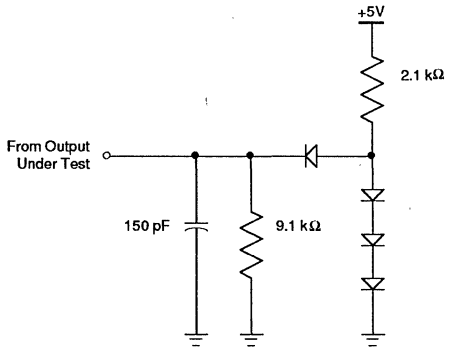


Figure 14. Test Load Diagram

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Min	Max	Units
V_{CC}	Supply Voltage*	-0.3	+7	V
T_{STG}	Storage Temp	-65°	+150°	C
T_A	Oper Ambient Temp	†	†	C

Notes:
*Voltages on all pins with respect to GND
† See Ordering Information

Stress greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

CAPACITANCE

T_A = GND = 0V, f = 1.0 MHz, unmeasured pins to GND

Parameter	Max
Input capacitance	10 pF
Output capacitance	20 pF
I/O capacitance	25 pF

V_{CC} SPECIFICATION

Low V_{CC} 3.3V ± 0.3V
High V_{CC} 5.0V ± 0.5V

DC ELECTRICAL CHARACTERISTICS

Symbol	Parameter	V_{CC}	T_A = 0°C to +70°C		T_A = -40°C to +105°C		Typical @ 25°C	Units	Conditions
			Min	Max	Min	Max			
	Max Input Voltage	3.0V		12		12		V	V_{IN} = 250 µA
		5.5V		12		12		V	V_{IN} = 250 µA
V_{CH}	Clock Input High Voltage	3.0V	0.7 V_{CC}	V_{CC}+0.3	0.7 V_{CC}	V_{CC}+0.3	1.7	V	Driven by External Clock Generator
		5.5V	0.7 V_{CC}	V_{CC}+0.3	0.7 V_{CC}	V_{CC}+0.3	2.75	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	3.0V	V_{SS}-0.3	0.2 V_{CC}	V_{SS}-0.3	0.2 V_{CC}	0.8	V	Driven by External Clock Generator
		5.5V	V_{SS}-0.3	0.2 V_{CC}	V_{SS}-0.3	0.2 V_{CC}	1.5	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	3.0V	0.7 V_{CC}	V_{CC}+0.3	0.7 V_{CC}	V_{CC}+0.3	1.8	V	
		5.5V	0.7 V_{CC}	V_{CC}+0.3	0.7 V_{CC}	V_{CC}+0.3	2.8	V	
V_{IL}	Input Low Voltage	3.0V	V_{SS}-0.3	0.2 V_{CC}	V_{SS}-0.3	0.2 V_{CC}	0.8	V	
		5.5V	V_{SS}-0.3	0.2 V_{CC}	V_{SS}-0.3	0.2 V_{CC}	1.5	V	
V_{OH}	Output High Voltge	3.0V	V_{CC}-0.4		V_{CC}-0.4		3.0	V	I_{OH} = -2.0 mA
		5.5V	V_{CC}-0.4		V_{CC}-0.4		4.8	V	I_{OH} = -2.0 mA
		3.0V	V_{CC}-0.4		V_{CC}-0.4			V	Low Noise @ 0.5 mA
		5.5V	V_{CC}-0.4		V_{CC}-0.4			V	Low Noise @ 0.5 mA
V_{OL1}	Output Low Voltage	3.0V		0.8		0.8	0.2	V	I_{OL} = +4.0 mA
		5.5V		0.4		0.4	0.1	V	I_{OL} = +4.0 mA
		3.0V		0.4		0.4		V	Low Noise @ 0.5 mA
		5.5V		0.4		0.4		V	Low Noise @ 0.5 mA
V_{OL2}	Output Low Voltage	3.0V		1.0		1.0	0.8	V	I_{OL} = +12 mA, 3 Pin Max
		5.5V		0.8		0.8	0.3	V	I_{OL} = +12 mA, 3 Pin Max
V_{OFFSET}	Comparator Input Offset Voltage	3.0V		25		25	10	mV	
		5.5V		25		25	10	mV	
V_{BO}	V_{CC} Brown Out Voltage		1.5	2.7	1.0	2.95	2.1	V	@ 2 MHz Max, Ext. CLK Freq
I_{IL}	Input Leakage (Input Bias Current of Comparator)	3.0V	-1.0	1.0	-1.0	1.0		µA	V_{IN} = 0V, V_{CC}
		5.5V	-1.0	1.0	-1.0	1.0		µA	V_{IN} = 0V, V_{CC}
I_{OL}	Output Leakage	3.0V	-1.0	1.0	-1.0	1.0		µA	V_{IN} = 0V, V_{CC}
		5.5V	-1.0	1.0	-1.0	1.0		µA	V_{IN} = 0V, V_{CC}
V_{REF}			0	V_{CC}-0.7	0	V_{CC} -1.0		V	

DC ELECTRICAL CHARACTERISTICS (Continued)

Symbol	Parameter	V_{CC}	T_A = 0°C to +70°C Min	Max	T_A = -40°C to +105°C Min	Max	Typical @ 25°C	Units	Conditions
I_{CC}	Supply Current	3.0V		3.5		3.5	1.5	mA	All Output and I/O Pins Floating @ 2 MHz
		5.5V		7.0		7.0	3.0	mA	All Output and I/O Pins Floating @ 2 MHz
		3.0V		8.0		8.0	3.0	mA	All Output and I/O Pins Floating @ 8 MHz
		5.5V		11.0		11.0	6.0	mA	All Output and I/O Pins Floating @ 8 MHz
		3.0V		10		10	3.6	mA	All Output and I/O Pins Floating @ 12 MHz
		5.5V		15		15	9.0	mA	All Output and I/O Pins Floating @ 12 MHz
I_{CC1}	Standby Current	3.0V		2.5		2.5	0.7	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 2 MHz
		5.5V		4.0		5.0	2.5	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 2 MHz
		3.0V		4.0		4.0	1.0	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 8 MHz
		5.5V		5.0		5.0	3.0	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 8 MHz
		3.0V		4.5		4.5	1.5	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 12 MHz
		5.5V		7.0		7.0	4.0	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 12 MHz
I_{CC}	Supply Current (Low Noise Mode)	3.0V		3.5		3.5	1.5	mA	All Output and I/O Pins Floating @ 1 MHz
		5.5V		7.0		7.0	4.2	mA	All Output and I/O Pins Floating @ 1 MHz
		3.0V		5.8		5.8	3.0	mA	All Output and I/O Pins Floating @ 2 MHz
		5.5V		9.0		9.0	6.0	mA	All Output and I/O Pins Floating @ 2 MHz
		3.0V		8.0		8.0	4.4	mA	All Output and I/O Pins Floating @ 4 MHz
		5.5V		11.0		11.0	9.0	mA	All Output and I/O Pins Floating @ 4 MHz

Symbol	Parameter	V_{CC}	T_A = 0°C to +70°C Min	Max	T_A = -40°C to +105°C Min	Max	Typical @ 25°C	Units	Conditions
I_{CC1}	Standby Current (Low Noise Mode)	3.0V		0.8		1.2	0.4	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 1 MHz
		5.5V		1		1.6	0.9	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 1 MHz
		3.0V		0.8		1.5	0.5	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 2 MHz
		5.5V		1		1.9	1	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 2 MHz
		3.0V		TBD		2.0	0.8	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 4 MHz
		5.5V		2.0		2.4	0.3	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 4 MHz
I_{CC2}	Standby Current	3.0V		10		20	1.0	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is not Running
		5.5V		10		20	1.0	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is not Running
I_{ALL}	Auto Latch Low Current	3.0V		6.0		8.0	3.0	µA	$0V < V_{IN} < V_{CC}$
		5.5V		22		30	16	µA	$0V < V_{IN} < V_{CC}$
I_{ALH}	Auto Latch High Current	3.0V		-4.0		-5.0	-1.5	µA	$0V < V_{IN} < V_{CC}$
		5.5V		-12.0		-20	-8.0	µA	$0V < V_{IN} < V_{CC}$

Notes:

[1] I_{CC1}

	Typ	Max	Unit	Freq
Clock Driven on Crystal	3.0	5.0	mA	8 MHz
or XTAL Resonator	0.3	50	mA	8 MHz

[2] V_{SS} = 0V = GND

[3] For 2.75V operating, the device operates down to V_{BO}. The minimum operational V_{CC} is determined on the value of the voltage V_{BO} at the ambient temperature. The V_{BO} increases as the temperature decreases.

†

DC ELECTRICAL CHARACTERISTICS (Continued)

Timing Diagrams

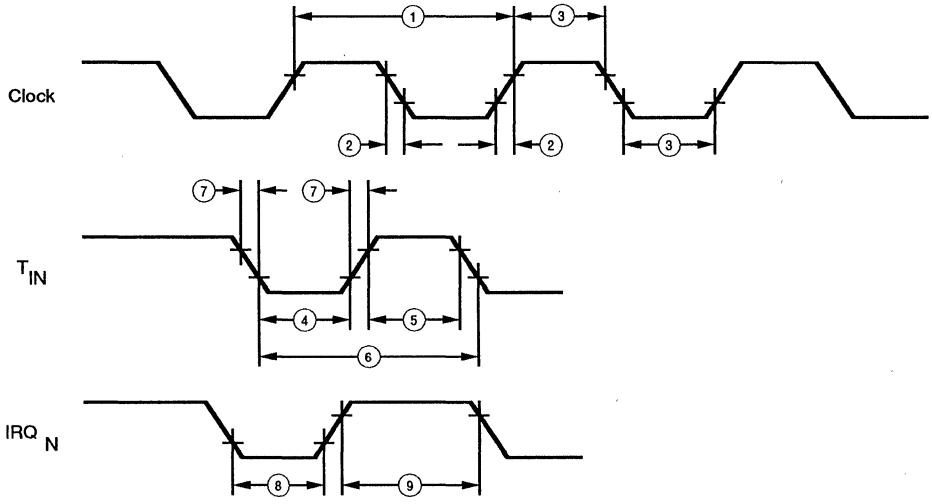


Figure 15. Electrical Timing Diagram

AC ELECTRICAL CHARACTERISTICS

Timing Table (Standard Mode)

No	Symbol	Parameter	V_{cc}	T_A = -40°C to +105°C				Units	Notes
				8 MHz		12 MHz			
				Min	Max	Min	Max		
1	TpC	Input Clock Period	3.0V	125	100,000	83	100,000	ns	[1]
			5.5V	125	100,000	83	100,000	ns	[1]
2	TrC,TfC	Clock Input Rise	3.0V		25		15	ns	[1]
		and Fall Times	5.5V		25		15	ns	
3	TwC	Input Clock Width	3.0V	37		26			[1]
			5.5V	37		26		ns	[1]
4	TwTinL	Timer Input Low Width	3.0V	100		100		ns	[1]
			5.5V	70		70		ns	[1]
5	TwTinH	Timer Input High Width	3.0V	5TpC		5TpC			[1]
			5.5V	5TpC		5TpC			[1]
6	TpTin	Timer Input Period	3.0V	8TpC		8TpC			[1]
			5.5V	8TpC		8TpC			[1]
7	TrTin,	Timer Input Rise	3.0V		100		100	ns	[1]
	TtTin	and Fall Timer	5.5V		100		100	ns	[1]
8	TwIL	Int. Request Input	3.0V	100		100		ns	[1,2]
		Low Time	5.5V	70		70		ns	[1,2]
9	TwIH	Int. Request Input	3.0V	5TpC		5TpC			[1]
		High Time	5.5V	5TpC		5TpC			[1,2]
10	Twdt	Watchdog Timer	3.0V		15		15	ms	[1]
		Delay Time	5.5V		10		10	ms	[1]
11	Tpor		3.0V		24		24	ms	[1]
			5.5V		12		12	ms	[1]

Notes:

[1] Timing Reference uses 0.9 V_{cc} for a logic 1 and 0.1 V_{cc} for a logic 0.

[2] Interrupt request via Port 3 (P31-P33).

Low Noise Version

Low EMI Emission

The Z86C08 can be programmed to operate in a low EMI emission mode by means of a mask ROM bit option. Use of this feature results in:

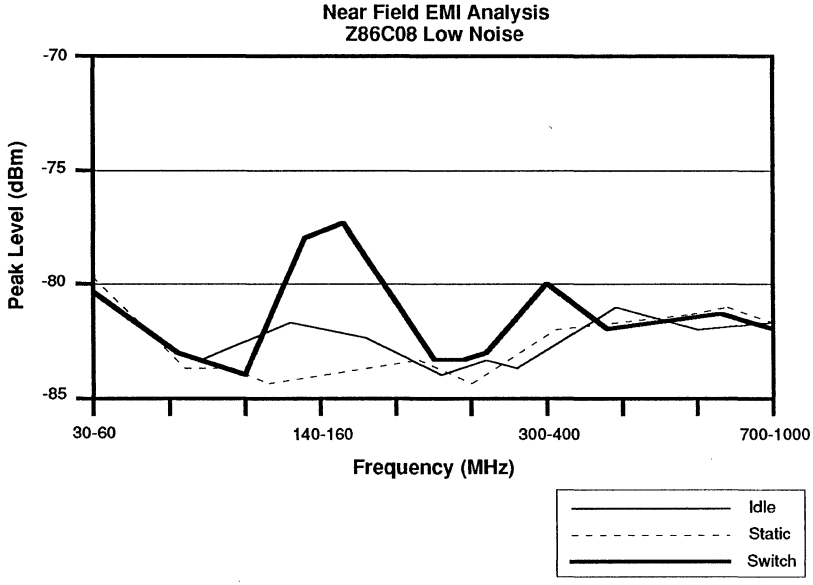
- Less than 1 mA consumed during HALT mode, -0°C to +70°C.
- All pre-driver slew rates reduced to 10 ns typical.
- Internal SLCK/TCLK operation limited to a maximum of 4 MHz - 250 ns cycle time.
- Output drivers have resistances of 200 ohms (typical).
- Oscillator divide-by-two circuitry eliminated.

The Low EMI mode is mask-programmable to be selected by the customer at the time the ROM code is submitted.

EMI Characteristics

The Z86C08 operating in the Low EMI mode generates EMI as measured in the following chart:

The measurements were made while operating the Z86C08 in three states: (1) Idle condition; (2) static output; (3) switched output.



AC ELECTRICAL CHARACTERISTICS
Low Noise Mode

No	Symbol	Parameter	V_{cc}	T_A= 0°C to +70°C				T_A= -40°C to +105°C				Units	Notes
				1 MHz		4 MHz		1 MHz		4 MHz			
				Min	Max	Min	Max	Min	Max	Min	Max		
5	TwTinH	Timer Input High Width	3.0V	2.5TpC		2.5TpC		2.5TpC		2.5TpC			[1]
			5.5V	2.5TpC		2.5TpC		2.5TpC		2.5TpC			[1]
6	TpTin	Timer Input Period	3.0V	4TpC		4TpC		4TpC		4TpC			[1]
			5.5V	4TpC		4TpC		4TpC		4TpC			[1]
7	TrTin,	Timer Input Rise	3.0V		100		100		100		100	ns	[1]
	TtTin	and Fall Timer	5.5V		100		100		100		100	ns	[1]
8	TwIL	Int. Request Input	3.0V	100		100		100		100		ns	[1,2]
		Low Time	5.5V	70		70		70		70		ns	[1,2]
9	TwIH	Int. Request Input	3.0V	2.5TpC		2.5TpC		2.5TpC		2.5TpC			[1]
		High Time	5.5V	2.5TpC		2.5TpC		2.5TpC		2.5TpC			[1,2]
10	Twdt	Watchdog Timer	3.0V		25		25		25		25	ms	[1]
		Delay Time	5.5V		15		15		10		10	ms	[1]

Notes:
[1] Timing Reference uses 0.9 V_{cc} for a logic 1 and 0.1 V_{cc} for a logic 0.
[2] Interrupt request via Port 3 (P31-P33)

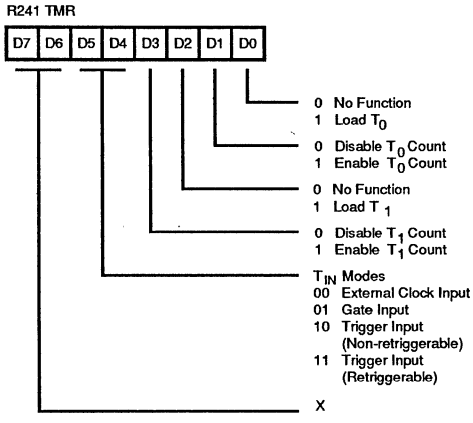


Figure 16. Timer Mode Register ($F1_H$: Read/Write)

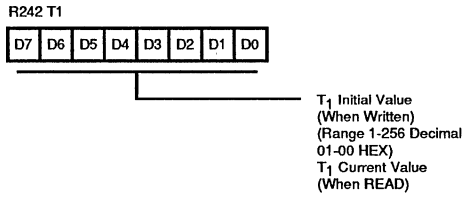


Figure 17. Counter Time 1 Register ($F2_H$: Read/Write)

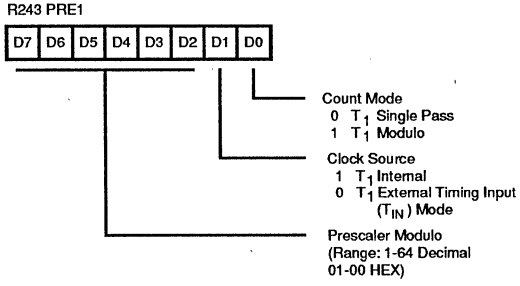


Figure 18. Prescaler 1 Register ($F3_H$: Write Only)

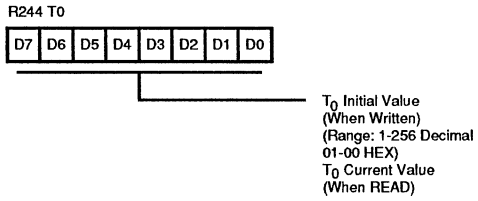


Figure 19. Counter/Timer 0 Register ($F4_H$: Read/Write)

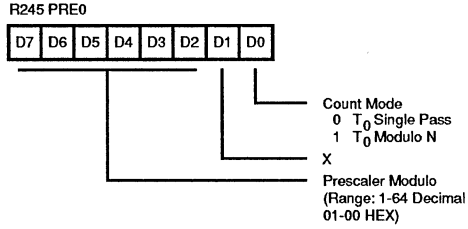


Figure 20. Prescaler 0 Register ($F5_H$: Write Only)

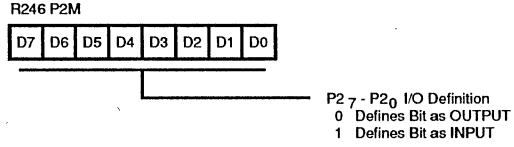


Figure 21. Port 2 Mode Register ($F6_H$: Write Only)

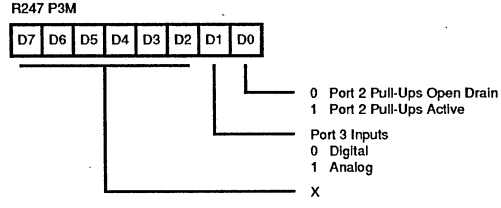


Figure 22. Port 3 Mode Register ($F7_H$: Write Only)

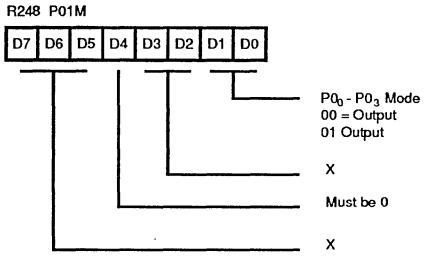


Figure 23. Port 0 and 1 Mode Register ($F8_H$: Write Only)

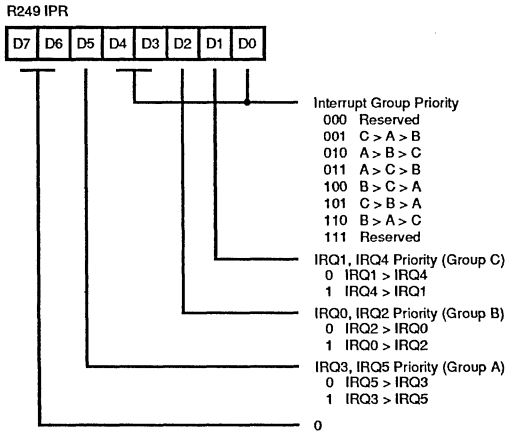


Figure 24. Interrupt Priority Register ($F9_H$: Write Only)

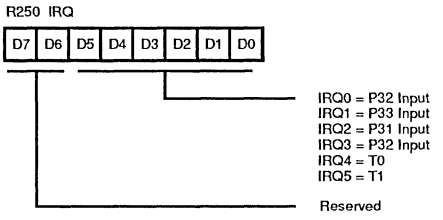


Figure 25. Interrupt Request Register (FA_H: Read/Write)

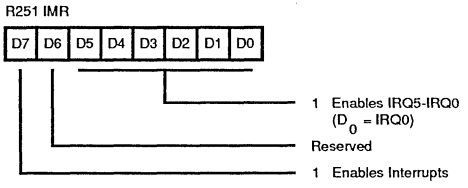


Figure 26. Interrupt Mask Register (FB_H: Read/Write)

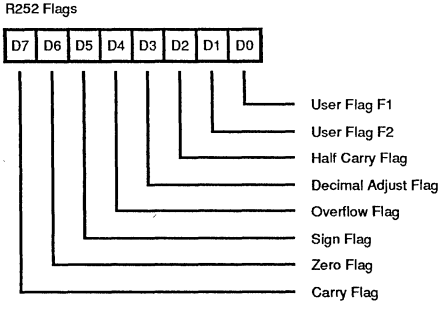


Figure 27. Flag Register (FC_H: Read/Write)

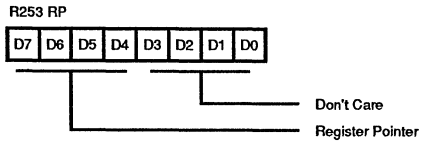


Figure 28. Register Pointer (FD_H: Read/Write)

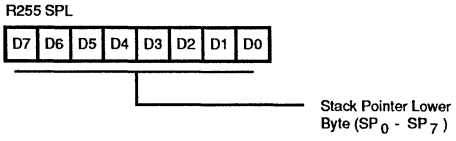


Figure 29. Stack Pointer (FF_H: Read/Write)

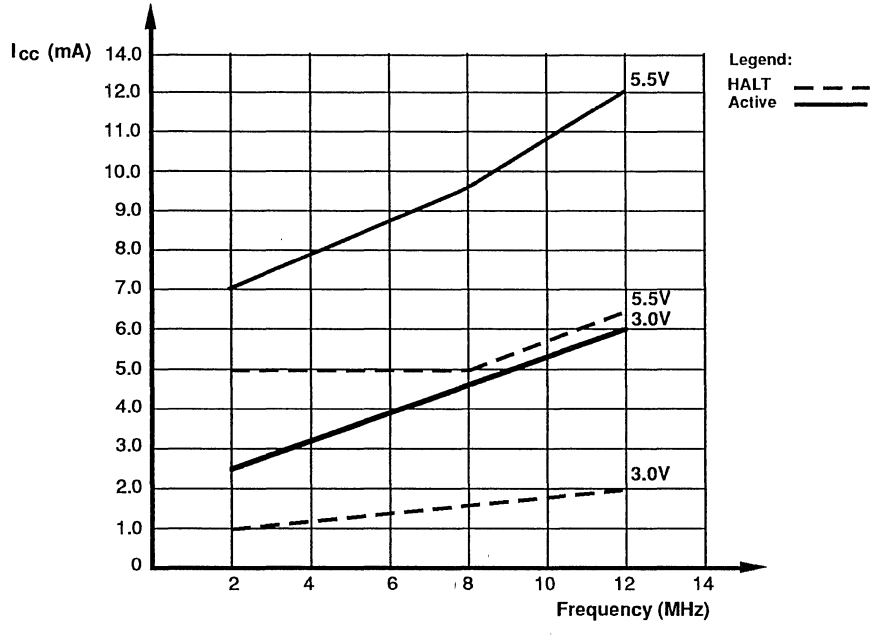


Figure 30. Maximum I_{cc} vs. Frequency

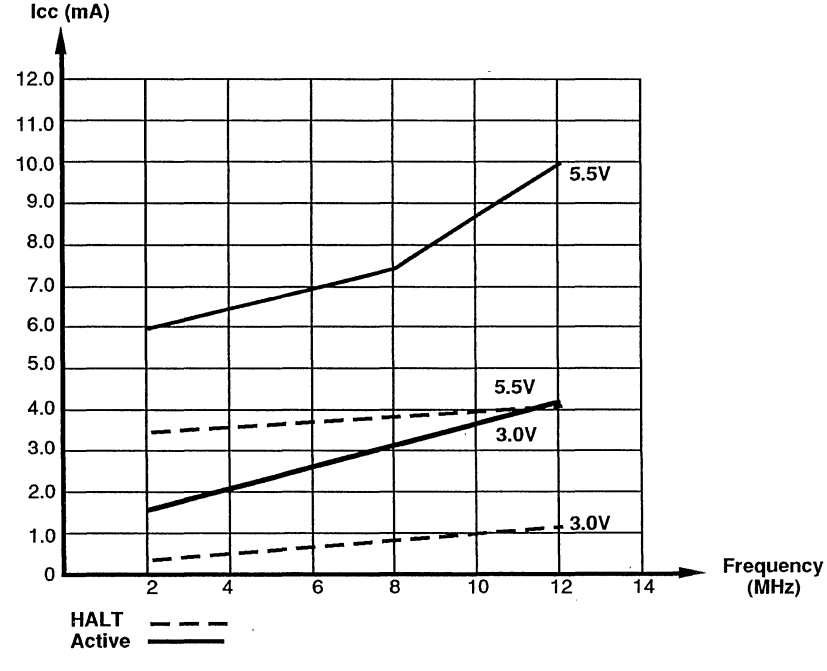


Figure 31. Typical I_{cc} vs. Frequency

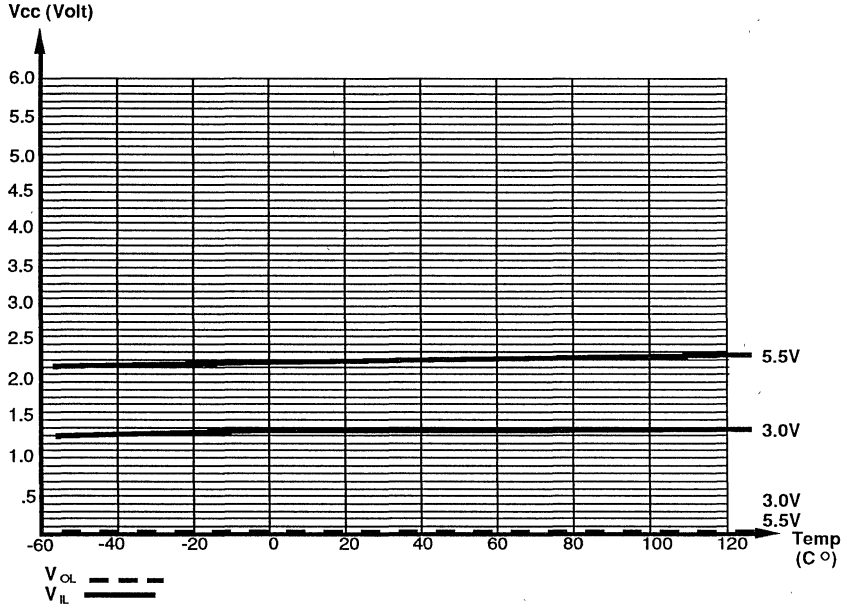


Figure 32. V_{IL}, V_{OL} vs. Temperature

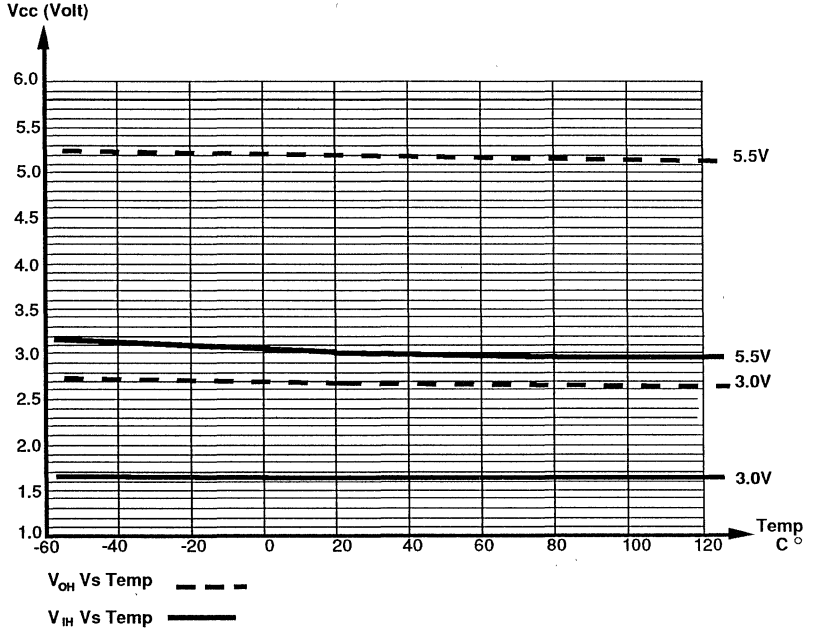


Figure 33. V_{IH}, V_{OH} vs. Temperature

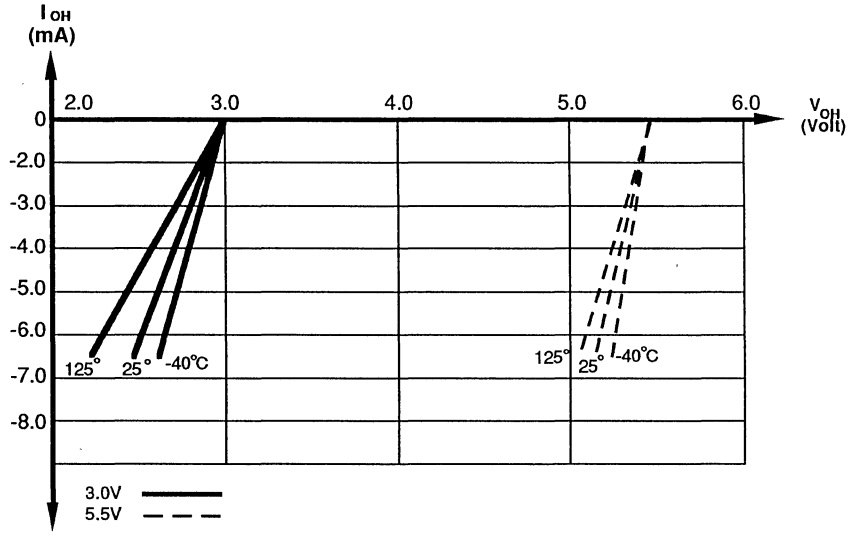


Figure 34. Typical I_{OH} vs. V_{OH}

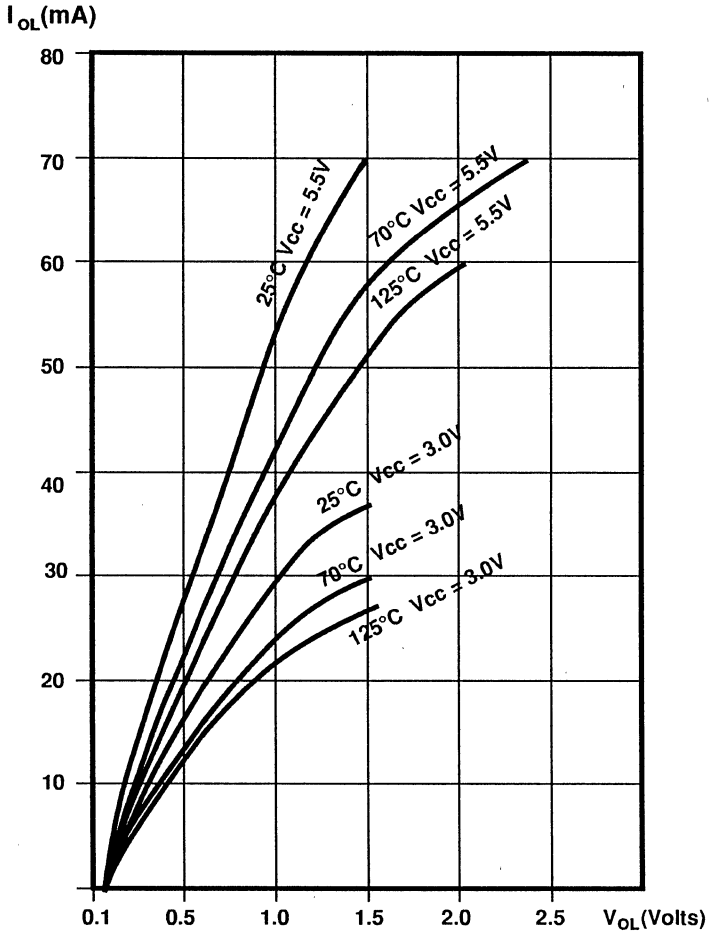


Figure 35. Typical I_{OL} vs. V_{OL}

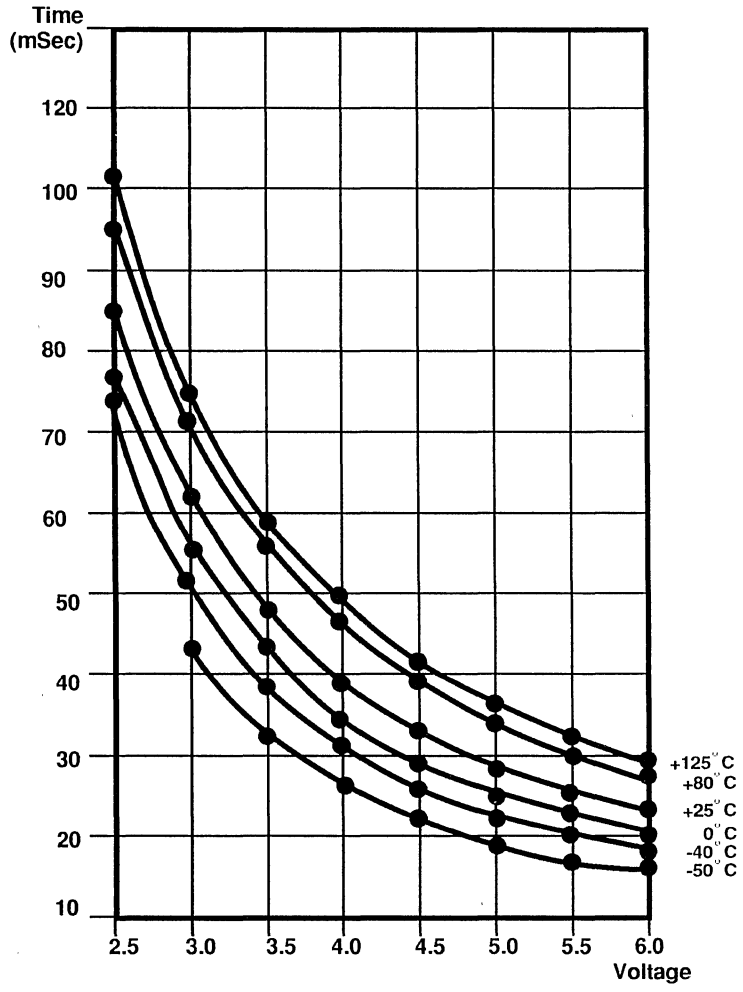


Figure 36. Typical WDT Time Out Period
vs. V_{cc} Over Temperature

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

Symbol	Meaning
IRR	Indirect register pair or indirect working-register pair address
Ir	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

Symbol	Meaning
dst	Destination location or contents
src	Source location or contents
cc	Condition code
@	Indirect address prefix
SP	Stack pointer
PC	Program counter
FLAGS	Flag register (Control Register 252)
RP	Register Pointer (R253)
IMR	Interrupt mask register (R251)

Flags. Control register (R252) contains the following six flags.

Symbol	Meaning
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

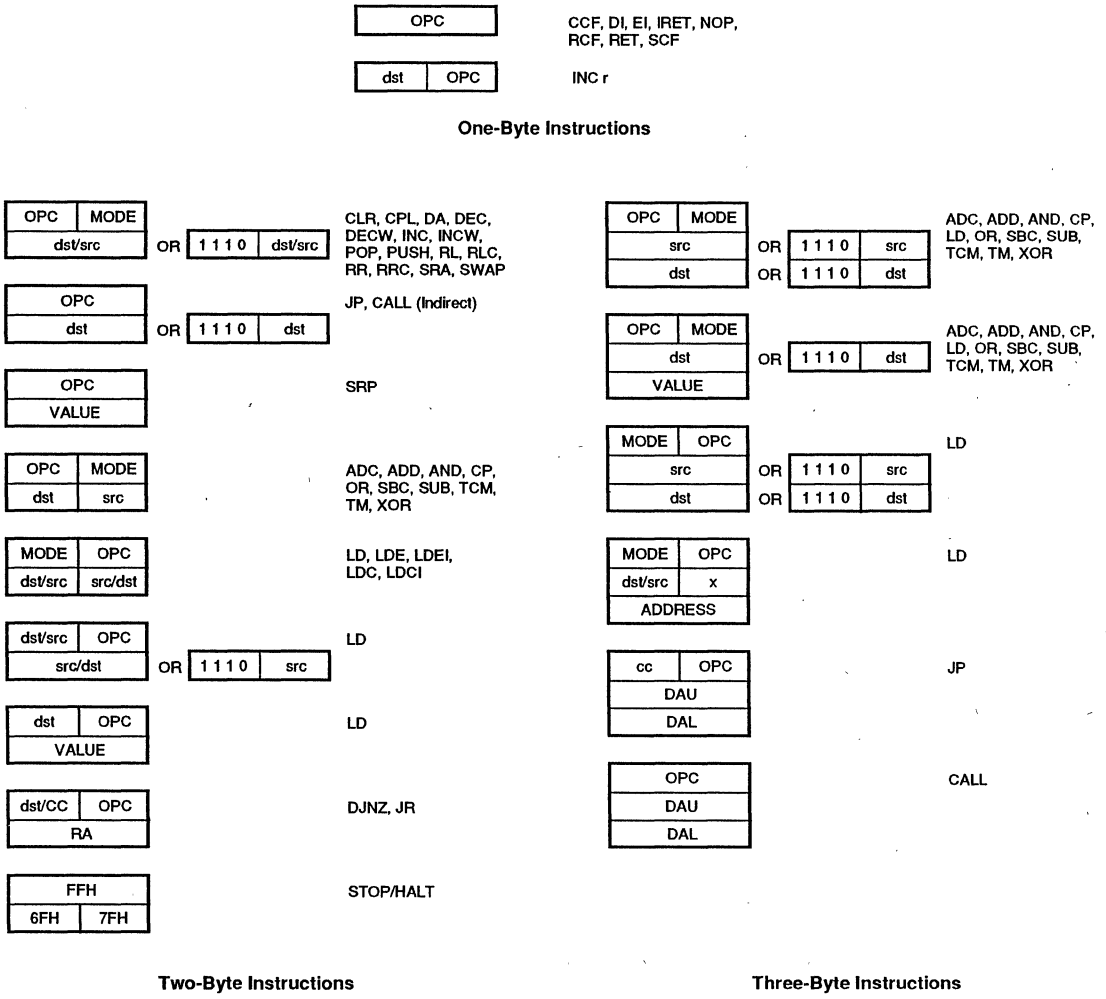
Affected flags are indicated by:

0	Clear to zero
1	Set to one
*	Set to clear according to operation
-	Unaffected
X	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000	---	Always true	---
0111	C	Carry	C=1
1111	NC	No Carry	C=0
0110	Z	Zero	Z=1
1110	NZ	Not zero	Z=0
1101	PL	Plus	S=0
0101	MI	Minus	S=1
0100	OV	Overflow	V=1
1100	NOV	No overflow	V=0
0110	EQ	Equal	Z=1
1110	NE	Not equal	Z=0
1001	GE	Greater than or equal	(S XOR V)=0
0001	LT	Less than	(S XOR V)=1
1010	GT	Greater than	[Z OR (S XOR V)]=0
0010	LE	Less than or equal	[Z OR (S XOR V)]=1
1111	UGE	Insigned greater than or equal	C=0
0111	ULT	Unsigned less than	C=1
1011	UGT	Unsigned greater than	(C=0 AND Z=0)=1
0011	ULE	Unsigned less than or equal	(C OR Z)=1
0000	---	Never true	---

INSTRUCTION FORMATS



INSTRUCTION SUMMARY

Note: Assignment of a value is indicated by the symbol "---". For example:

dst --- dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr (n)" is used to refer to bit (n) of a given operand location. For example:

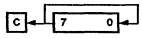
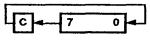
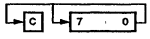
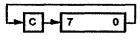
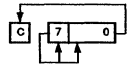
dst(7)

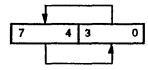
refers to bit 7 of the destination operand.

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst, src dst←dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst←dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst←dst AND src	†		5[ ]	-	*	*	0	-	-
CALL dst SP←SP - 2 @SP←PC, PC←dst	DA IRR		D6 D4	-	-	-	-	-	-
CCF C←NOT C			EF	*	-	-	-	-	-
CLR dst dst←0	R IR		B0 B1	-	-	-	-	-	-
COM dst dst←NOT dst	R IR		60 61	-	*	*	0	-	-
CP dst, src dst - src	†		A[ ]	*	*	*	*	-	-
DA dst dst←DA dst	R IR		40 41	*	*	*	X	-	-
DEC dst dst←dst - 1	R IR		00 01	-	*	*	*	-	-
DECW dst dst←dst - 1	RR IR		80 81	-	*	*	*	-	-
DI IMR(7)←0			8F	-	-	-	-	-	-
DJNZr, dst r←r - 1 if r ≠ 0 PC←PC + dst Range: +127, -128	RA		rA r = 0 - F	-	-	-	-	-	-
EI IMR(7)←1			9F	-	-	-	-	-	-
HALT			7F	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
INC dst dst←dst + 1	r R IR		rE r = 0 - F 20 21	-	*	*	*	-	-
INCW dst dst←dst + 1	RR IR		A0 A1	-	*	*	*	-	-
IRET FLAGS←@SP; SP←SP + 1 PC←@SP; SP←SP + 2; IMR(7)←1			BF	*	*	*	*	*	*
JP cc, dst if cc is true, PC←dst	DA IRR		cD c = 0 - F 30	-	-	-	-	-	-
JR cc, dst if cc is true, PC←PC + dst Range: +127, -128	RA		cB c = 0 - F	-	-	-	-	-	-
LD dst, src dst←src	r r R r X r Ir R R R IR IR	Im R r X r Ir r R IR IM IM R	rC r8 r9 r = 0 - F C7 D7 E3 F3 E4 E5 E6 E7 F5	-	-	-	-	-	-
LDC dst, src dst←src	r	Irr	C2	-	-	-	-	-	-
LDCI dst, src dst←src r←r + 1;rr←rr + 1	Ir	Irr	C3	-	-	-	-	-	-
NOP			FF	-	-	-	-	-	-

Instruction and Operation	Address Mode dst src	Opcode Byte (Hex)	Flags Affected C Z S V D H
OR dst, src dst←dst OR src	†	4[ ]	- * * 0 - -
POP dst dst←@SP; SP←SP + 1	R IR	50 51	- - - - - -
PUSH src SP←SP - 1; @SP←src	R IR	70 71	- - - - - -
RCF C←0		CF	0 - - - - -
RET PC←@SP; SP←SP + 2		AF	- - - - - -
RL dst 	R IR	90 91	* * * * - -
RLC dst 	R IR	10 11	* * * * - -
RR dst 	R IR	E0 E1	* * * * - -
RRC dst 	R IR	C0 C1	* * * * - -
SBC dst, src dst←dst←src←C	†	3[ ]	* * * * 1 *
SCF C←1		DF	1 - - - - -
SRA dst 	R IR	D0 D1	* * * 0 - -
SRP dst RP←src	Im	31	- - - - - -

Instruction and Operation	Address Mode dst src	Opcode Byte (Hex)	Flags Affected C Z S V D H
STOP		6F	1 - - - - -
SUB dst, src dst←dst←src	†	2[ ]	* * * * 1 *
SWAP dst 	R IR	F0 F1	X * * X - -
TCM dst, src (NOT dst) AND src	†	6[ ]	- * * 0 - -
TM dst, src dst AND src	†	7[ ]	- * * 0 - -
XOR dst, src dst←dst XOR src	†	B[ ]	- * * 0 - -

† These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

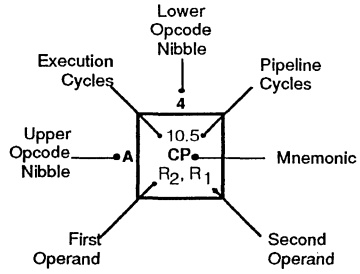
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address Mode dst	Address Mode src	Lower Opcode Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

OPCODE MAP

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								4.0 **WDH**
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								5.0 **WDT**
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								6.0 **STOP**
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								7.0 **HALT**
8	10.5 **DECW** RR1	10.5 **DECW** IR1														6.1 **DI**
9	6.5 **RL** R1	6.5 **RL** IR1														6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14.0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2				10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1			20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6.5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2			3					2					3	1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_1 = Dst address
R_2 or r_2 = Src address

Sequence:
Opcode, First Operand, Second Operand

Note: Blank areas not defined.

* 2-byte instruction appears as a 3-byte instruction

ZiLOG

PRODUCT SPECIFICATION

Z86E08

CMOS Z8® 8-BIT MICROCONTROLLER

FEATURES

- 8-bit CMOS microcontroller
- 18-pin DIP
- Low cost
- Low noise programmable
- ROM protect programmable
- 4.0 to 5.5 volt V_{cc} range
- Low power consumption - 50 mW (typical)
- Fast instruction pointer - 1 microsecond at 12 MHz
- Two standby modes - STOP and HALT
- 14 Input/Output lines
- All digital inputs, CMOS levels, Schmitt triggered.
- 2 Kbytes of one time PROM
- 144 bytes of RAM
- Two programmable 8-bit counter/timers each with a 6-bit programmable prescaler.
- Six vectored, priority interrupts from five different sources.
- Clock speeds - 8 and 12 MHz
- Watchdog Timer
- Power-On Reset
- Two Comparators
- On-chip oscillator that accepts a crystal, ceramic resonator, LC, or external clock drive.

GENERAL DESCRIPTION

The Z86E08 Microcontroller (MCU) introduces a new level of sophistication to single-chip architecture. The Z86E08 is a member of the Z8 single-chip microcontroller family with 2 Kbytes of one-time PROM. The device is housed in an 18-pin DIP, and is manufactured in CMOS technology. The device allows easy software development and debug, prototyping, and small production runs not economically desirable with a masked ROM version.

The Z86E08 has a flexible I/O scheme, an efficient register and address space structure. Also, it has a number of ancillary features that are useful in many consumer, industrial and advanced scientific applications.

The device applications demand powerful I/O capabilities. The Z86E08 fulfills this with 14 pins dedicated to input and output. These lines are grouped into three ports, and are configurable under software control to provide I/O, timing, and status signals.

There are two basic address spaces available to support this wide range of configurations; program memory and 124 bytes of general-purpose registers.

To unburden the program from coping with real-time problems such as counting/timing and I/O data communications, the Z86E08 offers two on-chip counter/timers with a large number of user selectable modes. Included, are two on-board comparators that process analog signals with a common reference voltage (Figures 1 and 2).

Note: All Signals with a preceding front slash, "/", are active Low, e.g.: B/W (WORD is active Low); /B/W (BYTE is active Low, only).

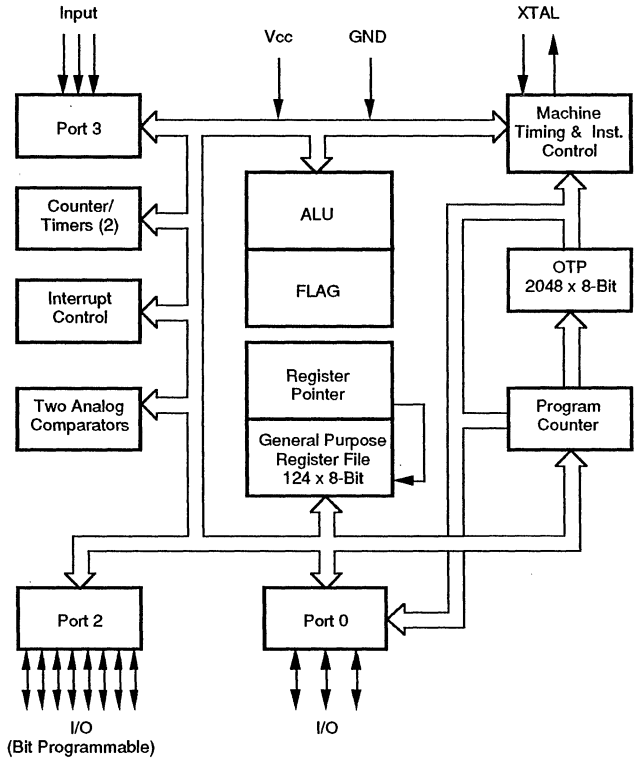


Figure 1. Functional Block Diagram

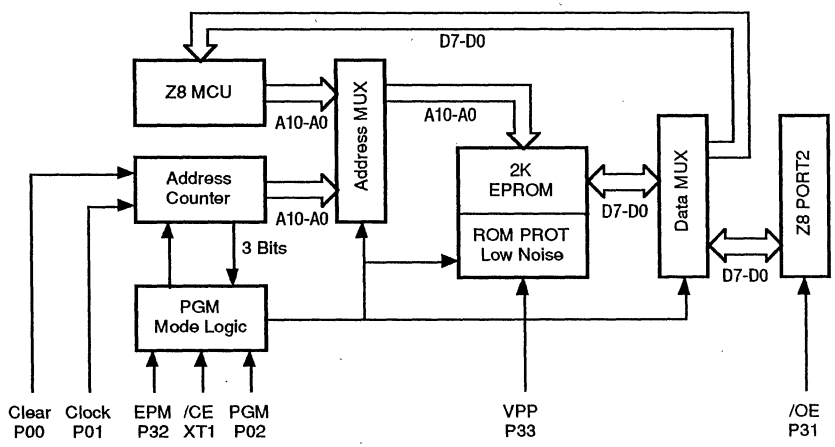
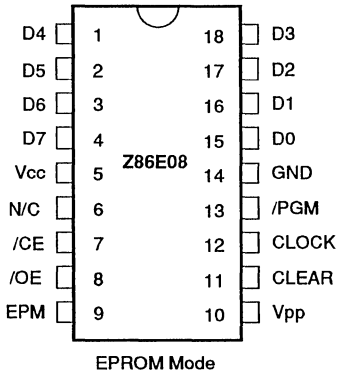


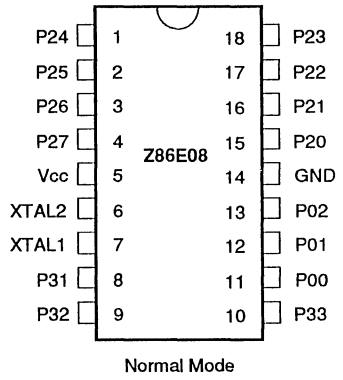
Figure 2. EPROM Mode Block Diagram

PIN DESCRIPTION



EPROM Mode

Figure 3. Pin Configuration



Normal Mode

Figure 4. Pin Configuration

Table 1. Pin Identification

OTP Programming Mode Pin #	Symbol	Function	Direction
1-4	D4-7	Data 4,5,6,7	In/Output
5	V_{CC}	Power Supply	Input
6	N/C	No connection	
7	/CE	Chip Enable	Input
8	/OE	Output Enable	Input
9	EPM	EPROM Prog Mode	Input
10	V_{PP}	Prog Voltage	Input
11	Clear	Clear Clock	Input
12	Clock	Address	Input
13	/PGM	Prog Mode	Input
14	GND	Ground	Input
15-18	D0-3	Data 0,1,2,3	In/Output

Table 2. Pin Identification

Z86E08 Standard Mode Pin #	Symbol	Function	Direction
1-4	P24-7	Port 2 pin 4,5,6,7	In/Output
5	V_{CC}	Power Supply	Input
6	XTAL2	Crystal Osc. Clock	Output
7	XTAL1	Crystal Osc. Clock	Input
8	P31	Port 3 pin 1	Input
9	P32	Port 3 pin 2	Input
10	P33	Port 3 pin 3	Input
11-13	P00-2	Port 0 pin 0,1,2	Input/Output
14	GND	Ground	Input
15-18	P20-3	Port 2 pin 0,1,2,3	In/Output

PIN FUNCTIONS

OTP Programming Mode

D7-D0. *Data Bus.* The data can be read from, or written to the EPROM through this data bus.

VCC. *Power Supply.* It is 5V during the EPROM Read mode and 6V during the other mode.

/CE. *Chip Enable* (Active Low). This pin is active during EPROM Read Mode, Program Mode, and Program Verify Mode.

/OE. *Output Enable* (Active Low). This pin drives the Data Bus direction. When this pin is Low, the Data Bus is output. When High, the Data Bus is input.

EPM. *EPROM Program Mode.* This pin controls the different EPROM Program Modes by applying different voltages.

V_{PP}. *Program Voltage.* This pin supplies the program voltage.

Clear. *Clear* (Active High). This pin resets the internal address counter at the High Level.

Clock. *Address Clock.* This pin is a clock input. The internal address counter increases by one with one clock signal.

/PGM. *Program Mode* (Active Low). Low Level at this pin programs the data to the EPROM through the Data Bus.

XTAL1, XTAL2. *Crystal In, Crystal Out* (time-based input and output, respectively). These pins connect a parallel-resonant crystal, LC, or an external single-phase clock (12 MHz max) to the on-chip clock oscillator and buffer.

Port 0 P00-P02. Port 0 is a 3-bit bi-directional, CMOS compatible I/O port. These 3 I/O lines can be globally configured under software control to be an input or output (Figure 5).

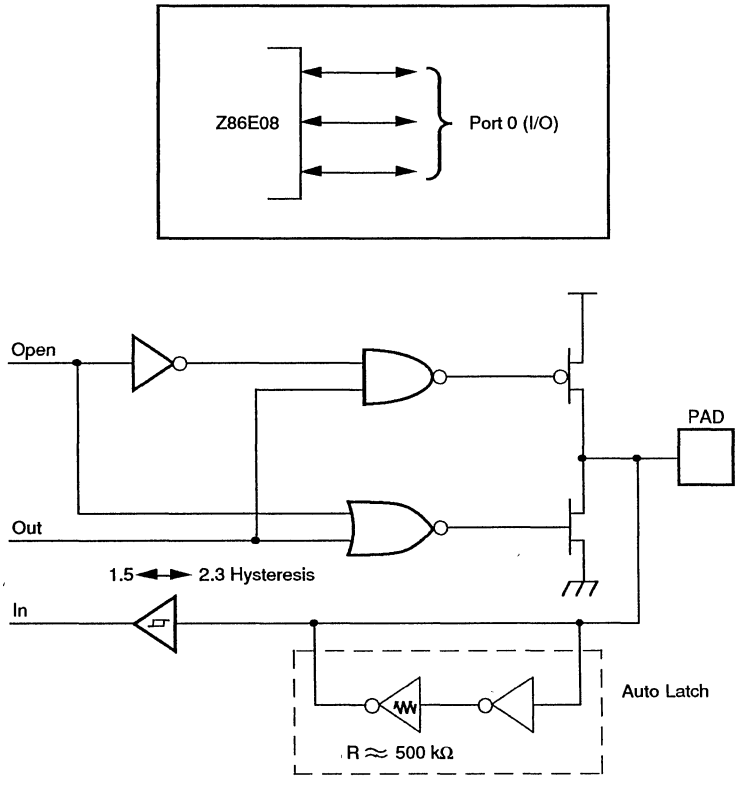


Figure 5. Port 0 Configuration

Port 2 P20-P27. Port 2 is an 8-bit, bit programmable, bidirectional, CMOS compatible I/O port. These eight I/O lines can be configured under software control to be an input or output, independently. Bits programmed as outputs can be globally programmed as either push-pull or open-drain (Figure 6).

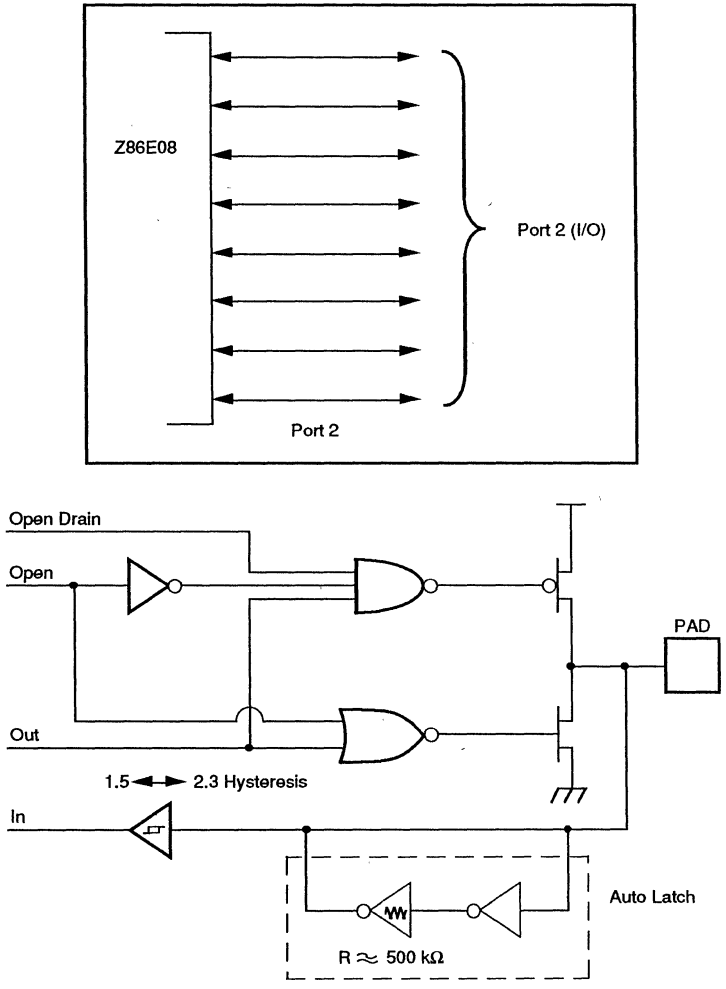


Figure 6. Port 2 Configuration

Port 3 P31-P33. Port 3 is a 3-bit, CMOS compatible port with three fixed input (P30-P32) lines. These three input lines can be configured under software control as digital inputs or analog inputs. These three input lines are also used as the interrupt sources IRQ0-IRQ3 and as the timer input signal (T_{IN} - Figure 7).

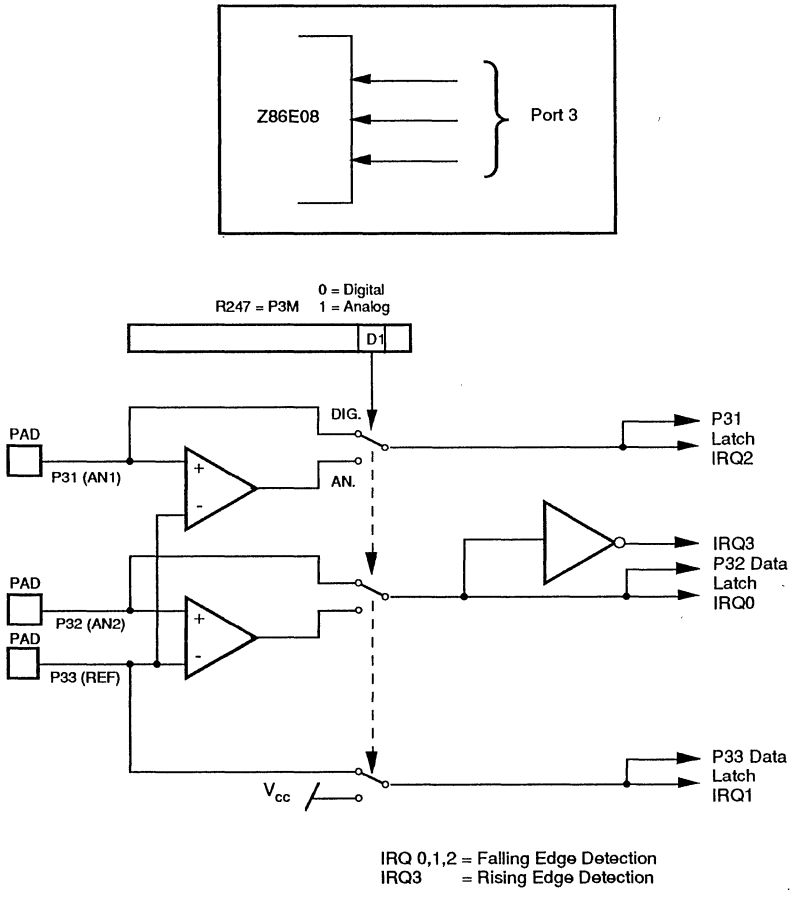


Figure 7. Port 3 Configuration

Comparator Inputs. Two analog comparators are added to input of Port 3, P31 and P32, for interface flexibility. The comparators reference voltage P3REF is common to both comparators.

Typical applications for the on-board comparators; Zero crossing detection, A/D conversion, voltage scaling, and threshold detection. In analog mode, P33 input functions serve as a reference voltage to the comparators.

The dual comparator (common inverting terminal) features a single power supply which discontinues power in STOP Mode. The common voltage range is 0-4V; the power supply and common mode rejection ratios are 90dB and 60dB, respectively.

Interrupts are generated on either edge of comparator 2's output, or on the falling edge of comparator 1's output. The comparator output is used for interrupt generation, Port 3 data inputs, or Tin through P31. Alternatively, the comparators can be disabled, freeing the reference input (P33) for use as IRQ1 and/or P33 input.

SPECIAL FUNCTIONS

The Z8 MCU incorporates special functions to enhance the Z8's application in industrial, scientific and advanced technologies applications.

RESET is accomplished through Power On or a watch-dog timer RESET. Upon Power Up, the power-on reset circuit waits for 50 msec plus 18 crystal clocks and then starts program execution at address 000C (HEX). Reference Table 3 for the Z86E08 control registers' reset values (Figure 8).

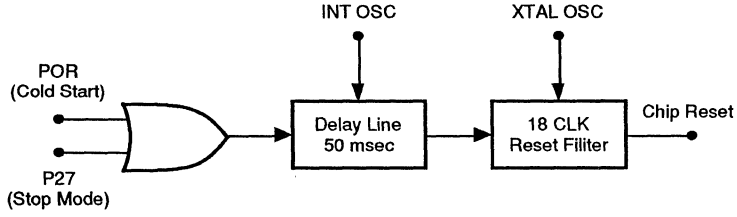


Figure 8. Internal Reset Configuration

Power-On Reset (POR). A timer circuit clocked by a dedicated on-board RC oscillator is used for a POR timer function. The POR time allows Vcc and the oscillator circuit to stabilize before instruction execution begins. The POR timer circuit is a one-shot timer triggered by one of the four following conditions:

- Power bad to power good status
- STOP Mode recovery
- WDT time out
- WDH time out

Watch Dog Timer Reset. The WDT is a retriggerable one-shot timer that resets the Z8 if it reaches its terminal count. The WDT is initially enabled by executing the WDT instruction and is retriggered on subsequent execution of the WDT instruction. The timer circuit is driven by an on-board RC oscillator.

Table 3. Z86E08 Control Registers

Addr.	Reg.	Reset Condition								Comments
		D7	D6	D5	D4	D3	D2	D1	D0	
F1	TMR	0	0	0	0	0	0	0	0	
F2	T1	U	U	U	U	U	U	U	U	
F3	PRE1	U	U	U	U	U	U	0	0	
F4	T0	U	U	U	U	U	U	U	U	
F5	PRE0	U	U	U	U	U	U	U	0	
F6*	P2M	1	1	1	1	1	1	1	1	Inputs after reset.
F7*	P3M	U	U	U	U	U	U	0	0	
F8*	P01M	U	U	U	0	U	U	0	1	
F9	IPR	U	U	U	U	U	U	U	U	
FA	IRQ	U	U	0	0	0	0	0	0	IRQ3 is used for positive edge detection.
PB	IMR	0	U	U	U	U	U	U	U	
PC	FLAGS	U	U	U	U	U	U	U	U	
FD	RP	0	0	0	0	0	0	0	0	
FF	SPL	U	U	U	U	U	U	U	U	

Note:
* Not reset after a low on P27 to get out of STOP Mode

Program Memory. The Z86E08 addresses up to 2 Kbytes of internal program memory (Figure 9). The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts. Bytes 0-2048 are on-chip one-time programmable ROM.

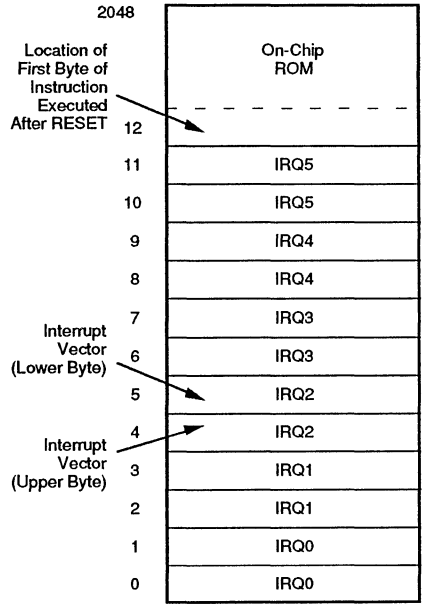


Figure 9. Program Memory Map

SPECIAL FUNCTIONS (Continued)

Register File. The Register File consists of three I/O port registers, 124 general purpose registers, and 14 control and status registers R0-R3, R4-R127 and R241-R255, respectively (Figure 10). General purpose registers occupy the 04H to 7FH address space. I/O ports are mapped as per the existing CMOS Z8. The Mode and Configuration Registers are the same as the Z86C08. The Z86E08 instructions can access registers directly or indirectly via an 8-bit address field. This allows short 4-bit register addressing using the Register Pointer. In the 4-bit mode, the register file is divided into eight working register groups, each occupying 16 continuous locations. The Register Pointer (Figure 11) addresses the starting location of the active working-register group.

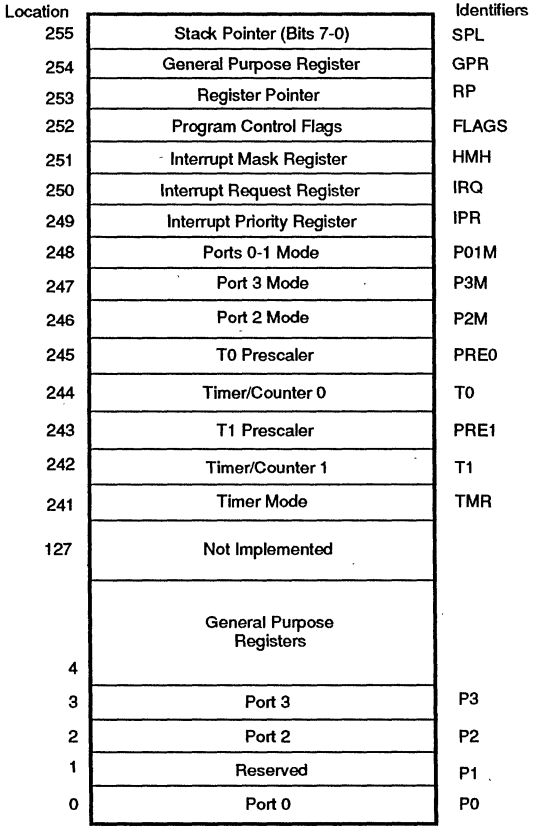


Figure 10. Register File

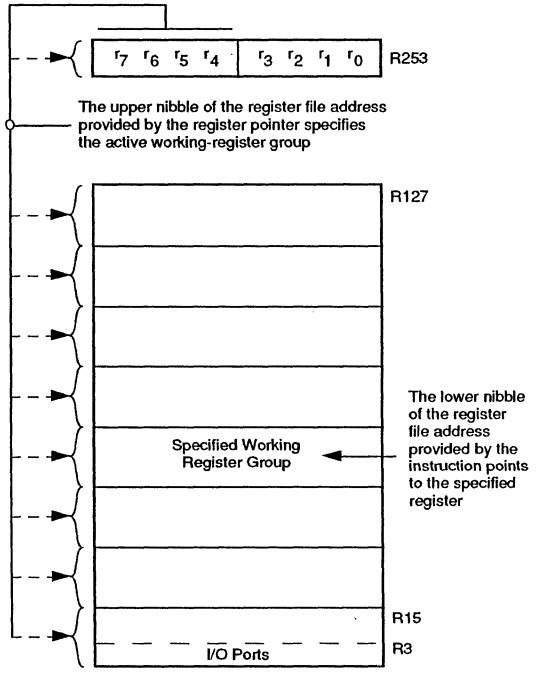


Figure 11. Register Pointer

Stack Pointer. The Z86E08 has an 8-bit Stack Pointer (R255) used for the internal stack that resides within the 124 general-purpose registers.

GPR. *(R254)* This register is a general-purpose register.

Counter/Timer. There are two 8-bit programmable counter/timers (T0 and T1), each driven by its own 6-bit programmable prescaler. The T1 prescaler is driven by internal or external clock sources; however, the T0 can be driven by the internal clock source only (Figure 12).

The 6-bit prescalers divides the input frequency of the clock source by any integer number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When both counter and prescaler reach the end of count, a timer interrupt request IRQ4 (T0) or IRQ5 (T1) is generated.

The counter can be programmed to start, stop, restart to continue, or restart from the initial value. The counters are also programmed to stop upon reaching zero (single pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode).

The counters, but not the prescalers, are read at any time without disturbing their value or count mode. The clock source for T1 is user-definable and is either the internal microprocessor clock divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input (P30) as an external clock, a trigger input that is retriggerable or not retriggerable, or used as a gate input for the internal clock.

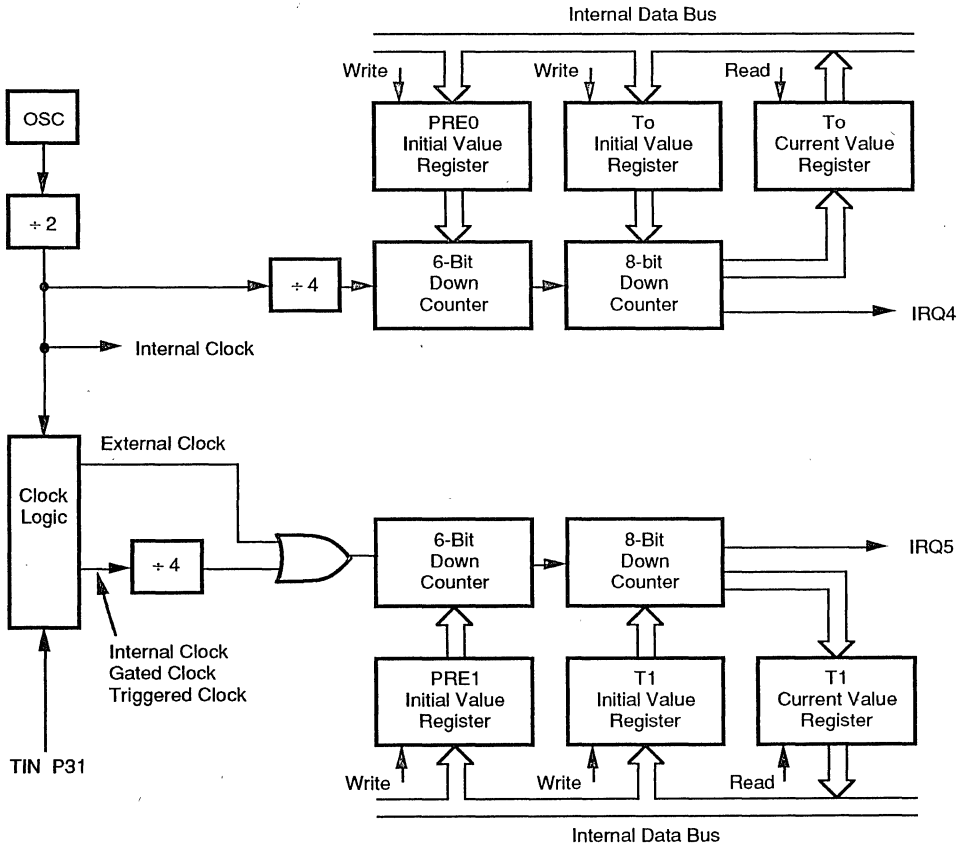


Figure 12. Counter/Timers Block Diagram

Interrupts. The Z86E08 has six interrupts from five different sources. These interrupts are maskable and prioritized (Figure 13). The five sources are divided as follows: the falling edge of P31 (AN1), P32 (AN2), P33 (REF), and two counter/timers. The Interrupt Mask Register globally or individually enables or disables the six interrupt requests (Table 4).

When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register. All Z86E08 interrupts are vectored through locations in program memory. When an Interrupt machine cycle is activated, an interrupt request is granted. This disables all subsequent interrupts, saves the Program Counter and Status Flags, and then branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit starting address of the interrupt service routine for that particular interrupt request.

To accommodate polled interrupt systems, interrupt inputs are masked and the interrupt request register is polled to determine which of the interrupt requests needs service.

Table 4. Interrupt Types, Sources, and Vectors

Source	Name	Vector Location	Comments
AN2(P32)	IRQ0	0,1	External (F)Edge
REF(P33)	IRQ1	2,3	External (F)Edge
AN1(P31)	IRQ2	4,5	External (F)Edge
AN2(P32)	IRQ3	6,7	External (R)Edge
T0	IRQ4	8,9	Internal
T1	IRQ5	10,11	Internal

Notes:
F=Falling edge triggered
R=Rising edge triggered

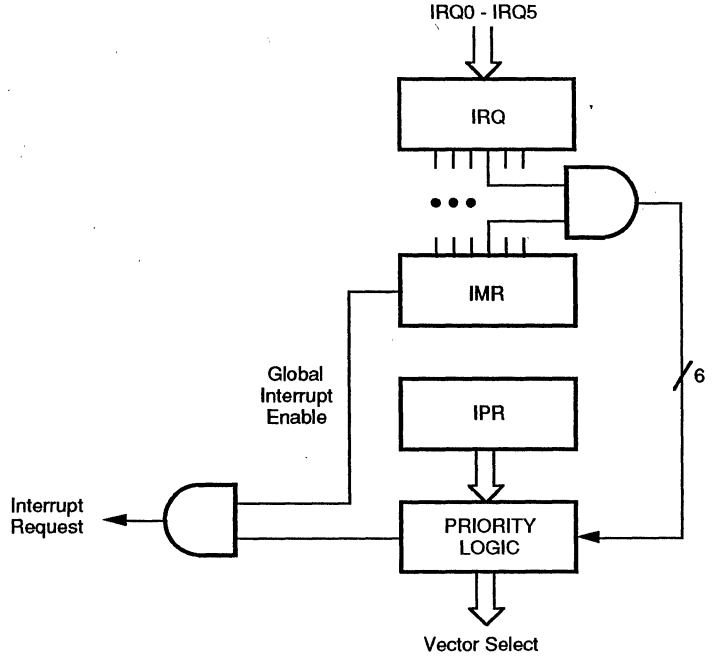


Figure 13. Interrupt Block Diagram

Clock. The Z86E08 on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal, ceramic resonator,or any suitable external clock source. The crystal should be AT cut, 12 MHz max, with a series resistance (RS) of less than or equal to 100 Ohms.

The crystal is connected across XTAL1 and XTAL2 using the recommended capacitors (capacitance is between 10 pF to 250 pF depending upon the crystal manufacturer, ceramic resonator and PCB layout) from each pin to ground (Figure 14).

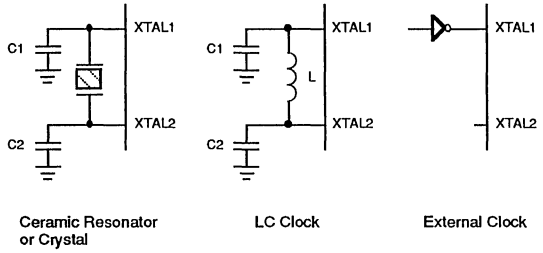


Figure 14. Oscillator Configuration

HALT Mode. Turns off the internal CPU clock but not the crystal oscillation. The counter/timers and external interrupts IRQ0, IRQ1, and IRQ2 remain active. The device is recovered by interrupts, either externally or internally generated. The I_{cc} in HALT state is I_{cc} (run mode) divided by 10.

STOP Mode. This instruction turns off the internal clock and external crystal oscillation and reduces the standby current to 10 µA. The STOP Mode is released by a RESET via a STOP Mode Recovery (pin P27. Program execution begins at location 000C (HEX). However, when P27 is used to release the STOP Mode, the I/O port mode registers are not reconfigured to their default power-on conditions. This prevents any I/O, configured as output when the STOP instruction was executed, from glitching to an unknown state. To use the P27 release approach with STOP Mode, use the following instruction:

```
OR      P2M, #80H
NOP
STOP
```

In order to enter STOP (or HALT) mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. To do this, the user must execute a NOP (opcode=FFH) immediately before the appropriate sleep instruction, i.e.:

```
FF    NOP;    clear the pipeline
6F    STOP;   enter STOP mode
              or
FF    NOP;    clear the pipeline
7F    HALT;   enter HALT mode
```

Watch Dog Timer (WDT). The Watch Dog Timer is enabled by instruction WDT. When the WDT is enabled, it cannot be stopped by the instruction. With the WDT instruction, the WDT is refreshed when it is enabled within every 15 msec; otherwise, the Z86E08 resets itself.

WDT=5F (HEX)

Opcode WDT (5FH). The first time opcode 5FH is executed, the WDT is enabled and subsequent execution clears the WDT counter. This has to be done at least every 15 msec. Otherwise, the WDT times out and generates a reset. The generated reset is the same as a power-on reset of 50 msec +18 XTAL clock cycles.

Opcode WDH (4FH). When this instruction is executed it enables the WDT during HALT. If not, the WDT stops when entering HALT. This instruction does not clear the counters, it just makes it possible to have the WDT running during HALT Mode. A WDH instruction executed without executing WDT (5FH) has no effect.

Auto Reset Voltage (V_{RST}). The Z86E08 has an auto-reset built-in. The auto-reset circuit resets the Z86E08 when it detects the V_{CC} below V_{RST}. Figure 15 shows the Auto Reset Voltage vs temperature.

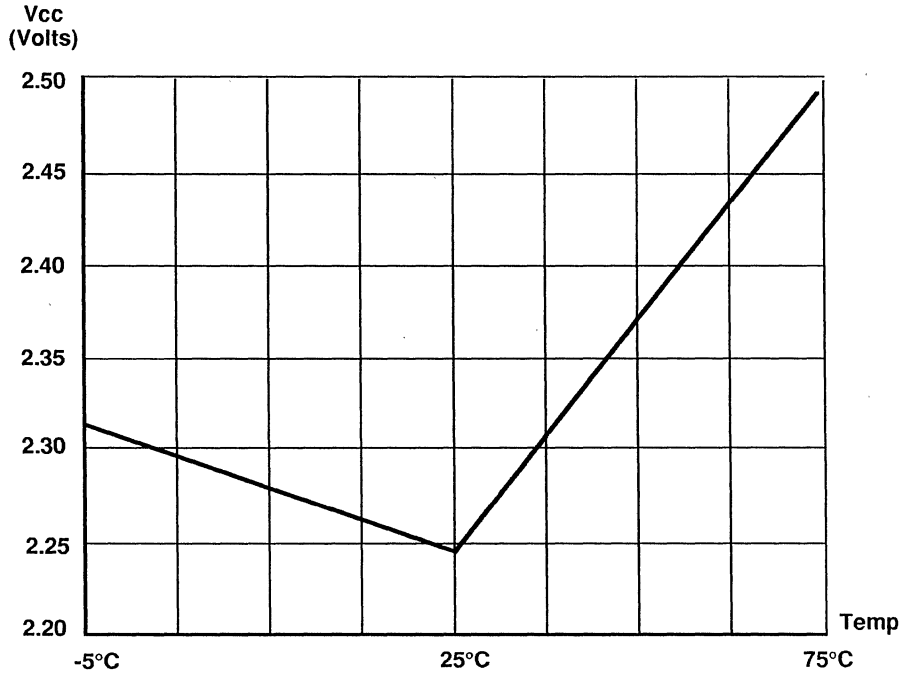


Figure 15. Typical Auto Reset Voltage (V_{RST}) vs Temperature

Low EMI Emission

The Z86E08 can be programmed to operate in a low EMI emission mode by means of an EPROM programmable bit option. Use of this feature results in:

- Less than 1 mA consumed during HALT mode.
- All drivers slew rates reduced to 10 ns typical.
- Internal SLCK/TCLK operation limited to a maximum of 4 MHz - 250 ns cycle time.
- Output drivers have resistances of 200 ohms (typical).
- Oscillator divide-by-two circuitry eliminated.

ROM Protect. ROM Protect fully protects the Z86E08 ROM code from being read externally. When ROM protect is selected, the Z86E08 will disable the instructions LDC and LDCI (Z86E08 and Z86C08 do not support the instructions of LDE and LDEI).

User Modes. Table 5 shows the programming voltage of each mode of Z86E08.

Table 5. OTP Programming Table

User Modes	V_{PP} (P33)	EPM (P32)	/CE (XTAL1)	/OE (P31)	/PGM (P02)	ADDR	DATA (Port2)	V_{CC}
EPROM Read	X	V_{IH}	V_{IL}	V_{IL}	V_{IH}	Addr	Data Out	5.0V
Program	V_{PP}	X	V_{IL}	V_{IH}	V_{IL}	Addr	Data In	6.0V
Program Verify	V_{PP}	X	V_{IL}	V_{IL}	V_{IH}	Addr	Data Out	6.0V
EPROM Protect	V_{PP}	V_H	V_H	V_{IH}	V_{IL}	X	X	6.0V
Low Noise	V_{PP}	V_{IH}	V_H	V_{IH}	V_{IL}	X	X	6.0V

Notes:
V_{PP} = 12.5V ± 0.5V
V_H = 12.5V ± 0.5V
X = TTL Level (irrelevant)
V_{IH} = 5.0V
V_{IL} = 0V

Internal Address Counter. The address of Z86E08 is generated internally with a counter clocked through pin P01 (Clock). Each clock signal increases the address by one and the "high" level of pin P00 (Clear) will reset the address to zero. Figure 16 shows the set-up time of the serial address input.

Programming Waveform. Figures 17, 18 and 19 show the programming waveforms of each mode. Table 6 shows the timing of programming waveforms.

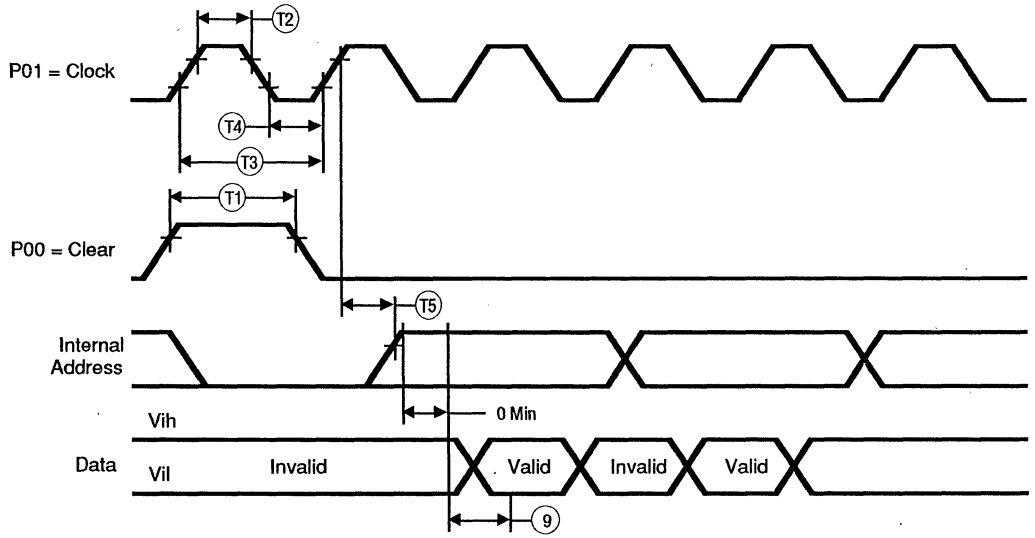
Programming Algorithm. Figure 20 shows the flow chart of the Z86E08 programming algorithm.

Table 6. Timing of Programming Waveform

Parameters	Name	Min	Max
1	Address Setup Time	2 µsec	
2	Data Setup Time	2 µsec	
3	V_{PP} Setup Time	2 µsec	
4	V_{CC} Setup Time	2 µsec	
5	Chip Enable SetupTime	2 µsec	
6	Program Pulse Width	0.95 msec	1.05 msec

Table 6. Timing of Programming Waveform (Continued)

Parameters	Name	Min	Max
7	Data Hold Time	2 µsec	
8	OE Setup Time	2 µsec	
9	Data Access Time		200 nsec
10	Data Output Float Time		100 nsec
11	Overprogram Pulse Width	2.85 msec	78.75 msec
12	EPM Setup Time	2 µsec	
13	OE Setup Time	2 µsec	
14	Address to OE Setup Time	2 µsec	
15	Option Bit Program Pulse Width	15 msec	78.75 msec



Legend:	
T1 Reset Clock Width	30 ns Min
T2 Input Clock High	30 ns Min
T3 Input Clock Period	70 ns Min
T4 Input Clock Low	30 ns Min
T5 Clock to Address Counter Out Delay	15 ns Max

Figure 16. Z86E08 Address Counter Waveform

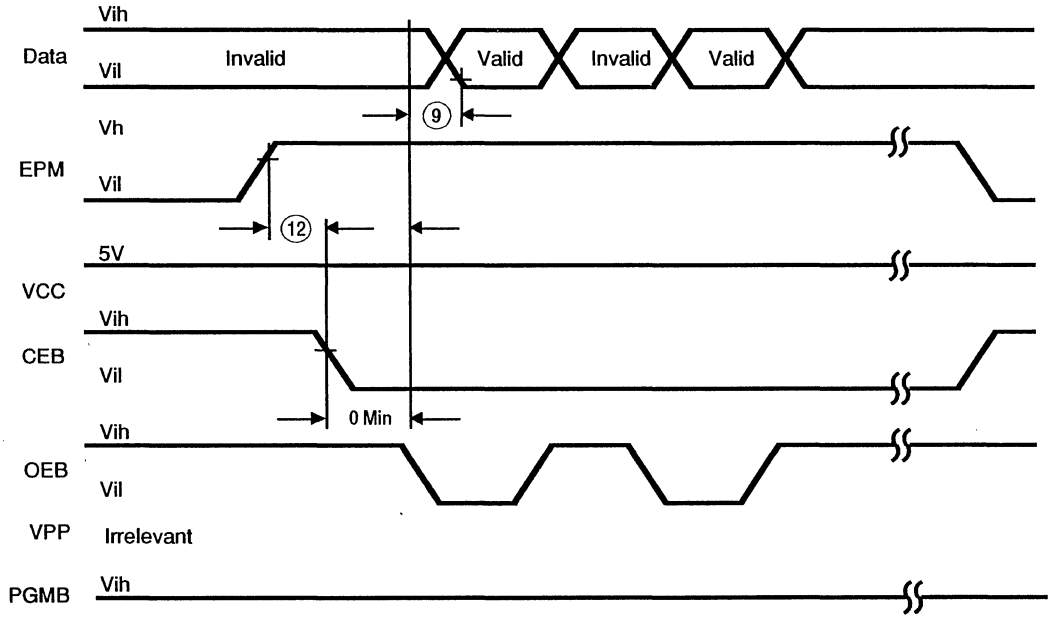


Figure 17. Z86E08 Programming Waveform (EPROM Read)

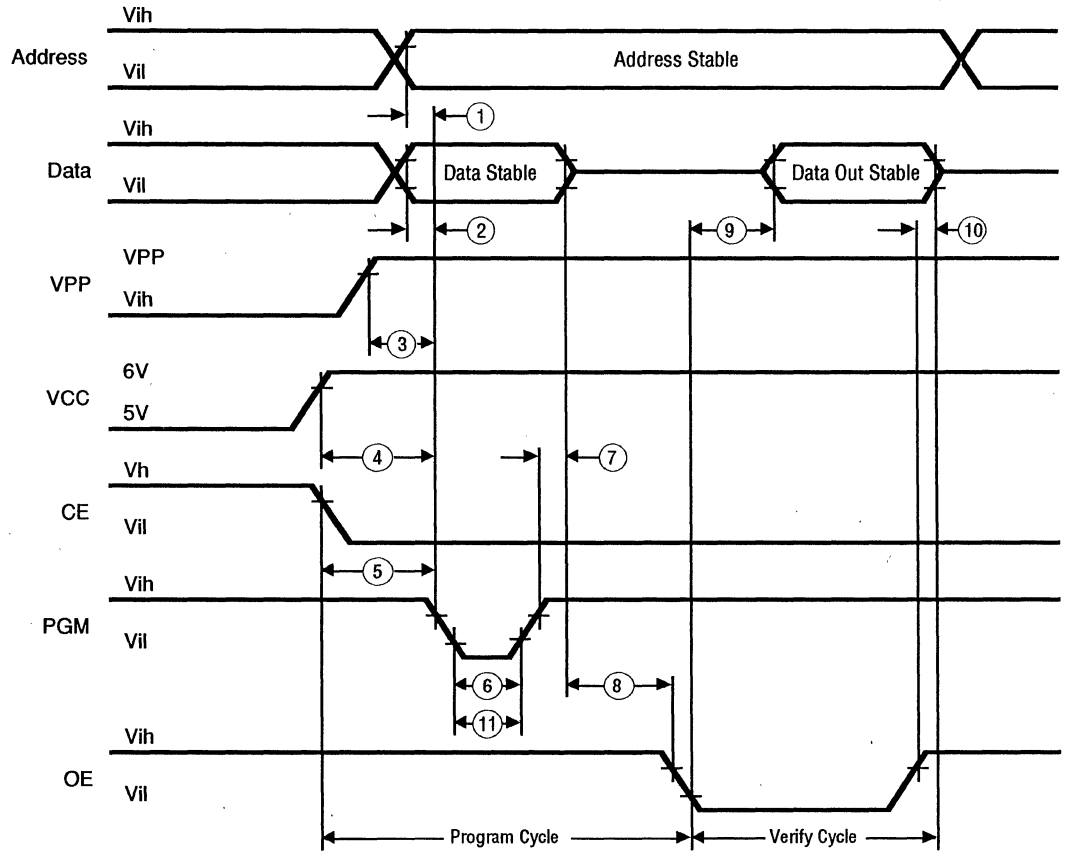


Figure 18. Z86E08 Programming Waveform (Program and Verify)

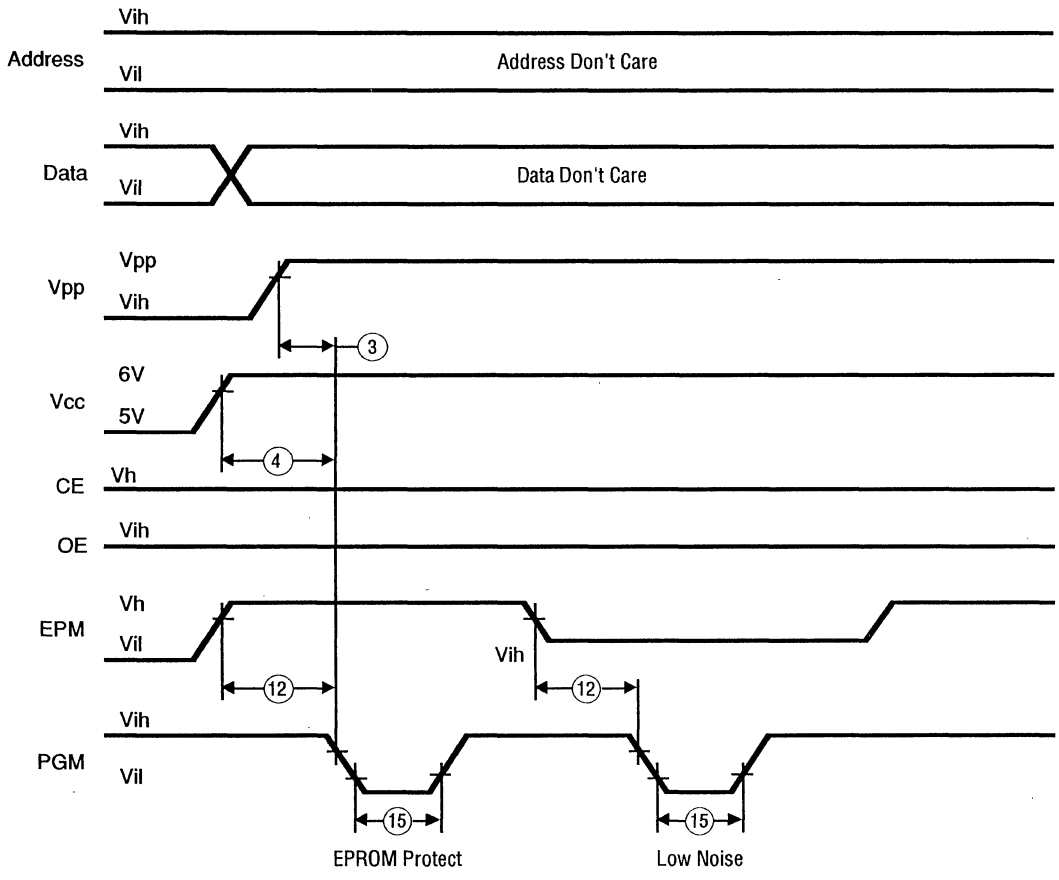


Figure 19. Z86E08 Programming Waveform (EPROM Protect and Low EMI Program)

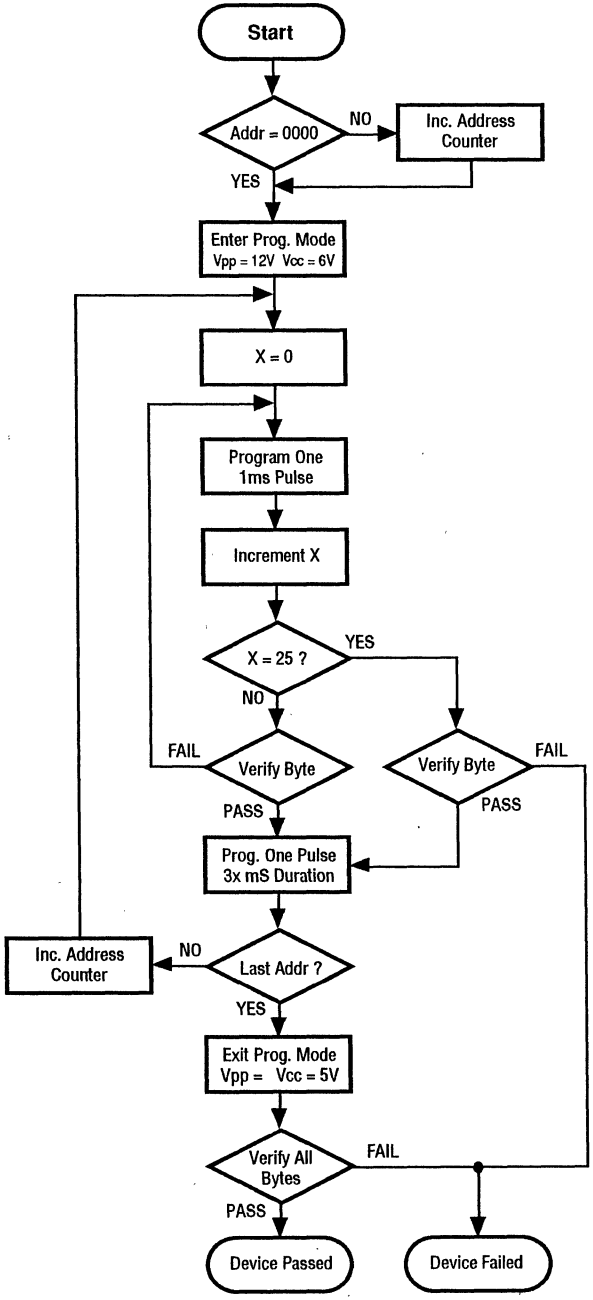


Figure 20. Z86E08 Programming Algorithm

STANDARD TEST CONDITIONS

The characteristics listed below apply for standard test conditions as noted. All voltages are referenced to GND. Positive current flows into the referenced pin (Figure 19).

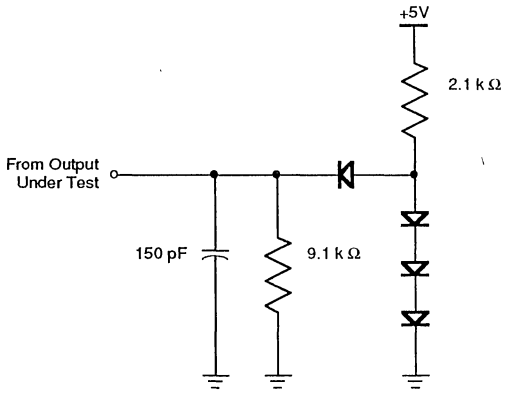


Figure 21. Test Load Diagram

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Min	Max	Units
V_{CC}	Supply Voltage*	-0.3	+7	V
T_{STG}	Storage Temp	-65	+150	C
T_A	Oper Ambient Temp	†	†	C

Notes:
* Voltages on all pins with respect to GND.
† See Ordering Information

Stress greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

CAPACITANCE

T_A = 25°C, V_{CC} = GND = 0V, f =1.0 MHz, unmeasured pins to GND.

Parameter	Max
Input Capacitance	10 pF
Output Capacitance	20 pF
I/O Capacitance	25 pF

V_{CC} SPECIFICATION

Low V_{CC} 4.4V ± 0.4V
High V_{CC} 5.0V ± 0.5V

DC ELECTRICAL CHARACTERISTICS

Symbol	Parameter	V_{CC}	T_A = 0°C to +70°C Min	Max	Typical @ 25°C	Units	Conditions
	Max Input Voltage	4.0V		12		V	V_{IN} = 250 µA
		5.5V		12		V	V_{IN} = 250 µA
V_{CH}	Clock Input High Voltage	4.0V	0.7 V_{CC}	V_{CC}+0.3	2.4	V	Driven by External Clock Generator
		5.5V	0.7 V_{CC}	V_{CC}+0.3	2.6	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	4.0V	V_{SS}-0.3	0.2 V_{CC}	1.6		Driven by External Clock Generator
		5.5V	V_{SS}-0.3	0.2 V_{CC}	2.3	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	4.0V	0.7 V_{CC}	V_{CC}+0.3	2.1	V	
		5.5V	0.7 V_{CC}	V_{CC}+0.3	2.7	V	
V_{IL}	Input Low Voltage	4.0V	V_{SS}-0.3	0.2 V_{CC}	1.2	V	
		5.5V	V_{SS}-0.3	0.2 V_{CC}	1.7	V	
V_{OH}	Output High Voltge	4.0V	V_{CC}-0.4		3.9	V	I_{OH} = -2.0 mA
		5.5V	V_{CC}-0.4		5.4	V	I_{OH} = -2.0 mA
V_{OL1}	Output Low Voltage	4.0V		0.8	0.2	V	I_{OL} = +4.0 mA
		5.5V		0.4	0.2	V	I_{OL} = +4.0 mA
V_{OL2}	Output Low Voltage	4.0V		TBD	0.7	V	I_{OL} = +12 mA, 3 Pin Max
		5.5V		0.8	0.5	V	I_{OL} = +12 mA, 3 Pin Max
V_{OFFSET}	Comparator Input Offset Voltage	4.0V		10	6	mV	
		5.5V		25	7	mV	
V_{RST}	Auto Reset Voltage		1.55	2.7	2.4	V	
I_{IL}	Input Leakage (Input Bias Current of Comparator)	4.0V	-1.0	1.0	1.0	µA	V_{IN} = 0V, V_{CC}
		5.5V	-1.0	1.0	1.0	µA	V_{IN} = 0V, V_{CC}
I_{OL}	Output Leakage	4.0V	-1.0	1.0	1.0	µA	V_{IN} = 0V, V_{CC}
		5.5V	-1.0	1.0	1.0	µA	V_{IN} = 0V, V_{CC}

Symbol	Parameter	V_{CC}	T_A = 0°C to +70°C Min	Max	Typical @ 25°C	Units	Conditions
I_{CC}	Supply Current (Standard Mode)	4.0V		4.0	2.2	mA	All Output and I/O Pins Floating @ 2 MHz
		5.5V		7.0	5.0	mA	All Output and I/O Pins Floating @ 2 MHz
		4.0V		9.0	4.5	mA	All Output and I/O Pins Floating @ 8 MHz
		5.5V		11.0	8.3	mA	All Output and I/O Pins Floating @ 8 MHz
		4.0V		10	6.1	mA	All Output and I/O Pins Floating @ 12 MHz
		5.5V		15	10.8	mA	All Output and I/O Pins Floating @ 12 MHz
I_{CC1}	Standby Current (Standard Mode)	4.0V		2.5	0.5	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 2 MHz
		5.5V		4.0	1.0	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 2 MHz
		4.0V		4.0	1.0	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 8 MHz
		5.5V		5.0	2.0	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 8 MHz
		4.0V		5.0	1.3	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 12 MHz
		5.5V		7.0	2.3	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 12 MHz
I_{CC}	Supply Current (Low Noise Mode)	4.0V		4.0	2.2	mA	All Output and I/O Pins Floating @ 1 MHz
		5.5V		7.0	4.2	mA	All Output and I/O Pins Floating @ 1 MHz
		4.0V		6.0	2.9	mA	All Output and I/O Pins Floating @ 2 MHz
		5.5V		9.0	5.5	mA	All Output and I/O Pins Floating @ 2 MHz
		4.0V		8.0	4.4	mA	All Output and I/O Pins Floating @ 4 MHz
		5.5V		11.0	7.9	mA	All Output and I/O Pins Floating @ 4 MHz

DC ELECTRICAL CHARACTERISTICS (Continued)

Symbol	Parameter	V_{CC}	T_A = 0°C to +70°C Min	Max	Typical @ 25°C	Units	Conditions
I_{CC1}	Standby Current (Low Noise Mode)	4.0V		1.2	0.4	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 1 MHz
		5.5V		1.6	0.9	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 1 MHz
		4.0V		1.5	0.5	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 2 MHz
		5.5V		1.9	1	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 2 MHz
		4.0V		2.0	0.8	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 4 MHz
		5.5V		2.4	1.3	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 4 MHz
I_{CC2}	Standby Current	4.0V		10	1.0	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is not Running
		5.5V		10	1.0	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is not Running
I_{ALL}	Auto Latch Low Current	4.0V		-7.0	-3.3	µA	$0V < V_{IN} < V_{CC}$
		5.5V		-7.0	-6.5	µA	$0V < V_{IN} < V_{CC}$
I_{ALH}	Auto Latch High Current	4.0V		10	-6.0	µA	$0V < V_{IN} < V_{CC}$
		5.5V		15	11.5	µA	$0V < V_{IN} < V_{CC}$

Notes:

[1] I_{CC1}

	Typ	Max	Unit	Freq
Clock Driven on Crystal	3.0	5.0	mA	8 MHz
or XTAL Resonator	0.3	5.0	mA	8 MHz

[2] V_{SS} = 0V = GND

AC ELECTRICAL CHARACTERISTICS

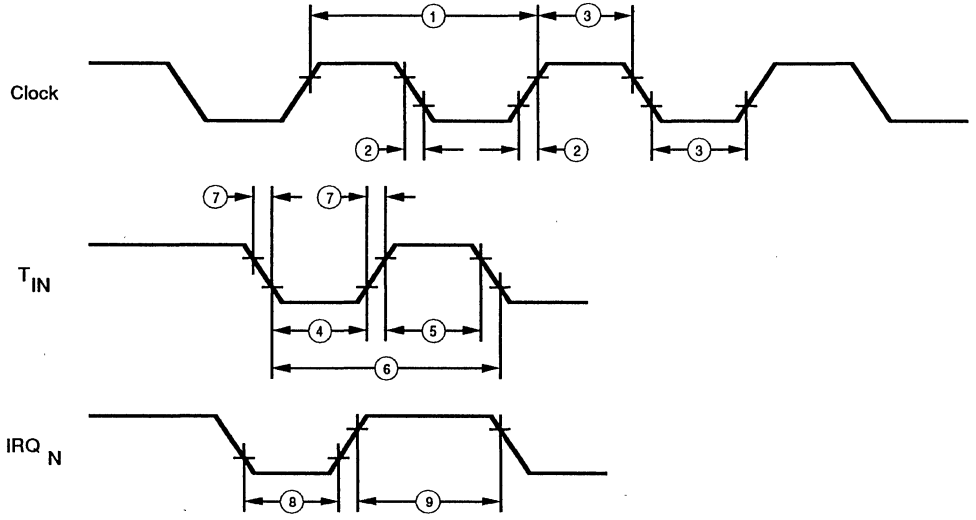


Figure 22. Electrical Timing Diagram

AC ELECTRICAL CHARACTERISTICS
Low Noise Mode

No	Symbol	Parameter	V_{cc}	T_A = 0°C to +70°C				Units	Notes
				1 MHz		4 MHz			
				Min	Max	Min	Max		
1	TpC	Input Clock Period	4.0V	500		125	100,000	ns	[1]
			5.5V	500		125	100,000	ns	[1]
2	TrC,TfC	Clock Input Rise and Fall Times	4.0V		25		25	ns	[1]
			5.5V		25		25	ns	
3	TwC	Input Clock Width	4.0V	225		37		ns	[1]
			5.5V	225		37		ns	[1]
4	TwTinL	Timer Input Low Width	4.0V	100		100		ns	[1]
			5.5V	70		70		ns	[1]

AC ELECTRICAL CHARACTERISTICS (Continued)
Low Noise Mode

No	Symbol	Parameter	V_{CC}	T_A = 0°C to +70°C				Units	Notes
				1 MHz		4 MHz			
				Min	Max	Min	Max		
5	TwTinH	Timer Input High Width	4.0V	1.5TpC		1.5TpC			[1]
			5.5V	1.5TpC		1.5TpC			[1]
6	TpTin	Timer Input Period	4.0V	4TpC		4TpC			[1]
			5.5V	4TpC		4TpC			[1]
7	TrTin, TtTin	Timer Input Rise and Fall Timer	4.0V		100		100	ns	[1]
			5.5V		100		100	ns	[1]
8	TwIL	Int. Request Input Low Time	4.0V	100		100		ns	[1,2]
			5.5V	70		70		ns	[1,2]
9	TwIH	Int. Request Input High Time	4.0V	1.5TpC		1.5TpC			[1]
			5.5V	1.5TpC		1.5TpC			[1,2]
10	Twdt	Watchdog Timer Delay Time	4.0V		20		20	ms	[1]
			5.5V		15		15	ms	[1]
11	TPOR	Power On Reset Time	4.0V		100		100	ms	[1]
			5.5V		90		90	ms	[1]

Notes:
[1] Timing Reference uses 0.9 V_{CC} for a logic 1 and 0.1 V_{CC} for a logic 0.
[2] Interrupt request via Port 3 (P31-P33)

AC ELECTRICAL CHARACTERISTICS
Standard Mode, Standard Temperature

No	Symbol	Parameter	V_{cc}	T_A = 0°C to +70°C						Units	Notes
				2 MHz		8 MHz		12 MHz			
				Min	Max	Min	Max	Min	Max		
1	TpC	Input Clock Period	4.0V	500		125	100,000	83	100,000	ns	[1]
			5.5V	500		125	100,000	83	100,000	ns	[1]
2	TrC,TfC	Clock Input Rise and Fall Times	4.0V		25		25		15	ns	[1]
			5.5V		25		25		15	ns	
3	TwC	Input Clock Width	4.0V	225		37		26		ns	[1]
			5.5V	225		37		26		ns	[1]
4	TwTinL	Timer Input Low Width	4.0V	100		100		100		ns	[1]
			5.5V	70		70		70		ns	[1]
5	TwTinH	Timer Input High Width	4.0V	3TpC		3TpC		3TpC			[1]
			5.5V	3TpC		3TpC		3TpC			[1]
6	TpTin	Timer Input Period	4.0V	8TpC		8TpC		8TpC			[1]
			5.5V	8TpC		8TpC		8TpC			[1]
7	TrTin, TtTin	Timer Input Rise and Fall Timer	4.0V		100		100		100	ns	[1]
			5.5V		100		100		100	ns	[1]
8	TwIL	Int. Request Input Low Time	4.0V	100		100		100		ns	[1,2]
			5.5V	70		70		70		ns	[1,2]
9	TwIH	Int. Request Input High Time	4.0V	3TpC		3TpC		3TpC			[1]
			5.5V	3TpC		3TpC		3TpC			[1,2]
10	Twdt	Watchdog Timer Delay Time	4.0V		50		50		50	ms	[1]
			5.5V		45		45		45	ms	[1]
11	TPOR	Power On Reset Time	4.0V		100		100		100	ms	[1]
			5.5V		90		90		90	ms	[1]

Notes:
[1] Timing Reference uses 0.9 V_{cc} for a logic 1 and 0.1 V_{cc} for a logic 0.
[2] Interrupt request via Port 3 (P31-P33)

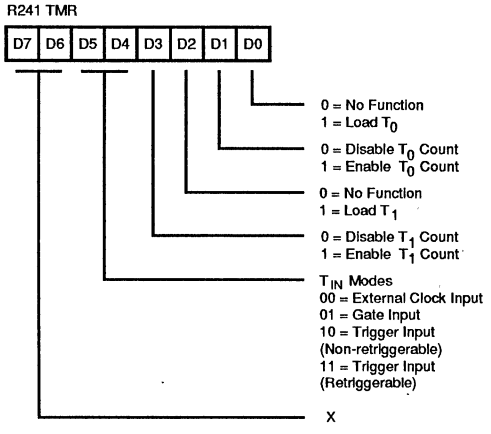


Figure 23. Timer Mode Register
(F1H: Read/Write)

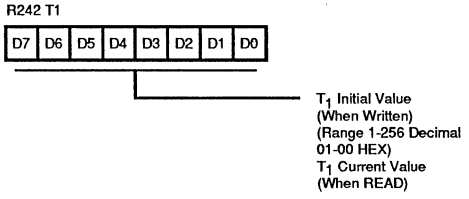


Figure 24. Counter Timer 1 Register
(F2H: Read/Write)

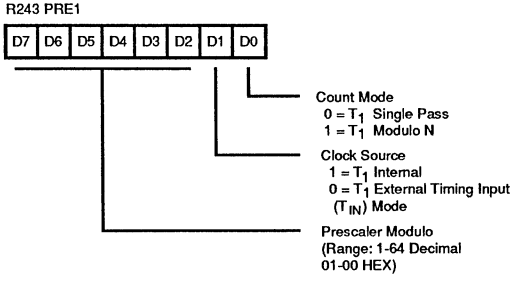
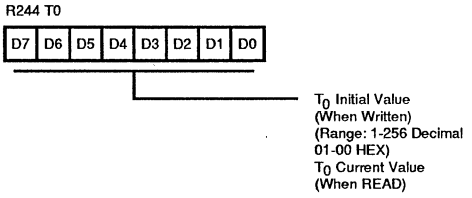


Figure 25. Prescaler 1 Register
(F3H: Write Only)



Figrue 26. Counter/Timer 0 Register
(F4H: Read/Write)

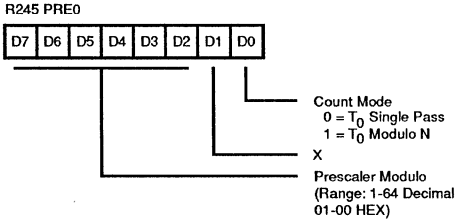


Figure 27. Prescaler 0 Register
(F5H: Write Only)

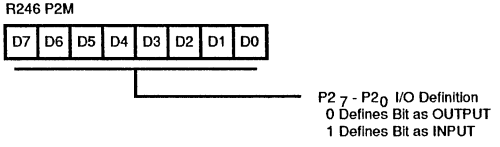


Figure 28. Port 2 Mode Register
(F6H: Write Only)

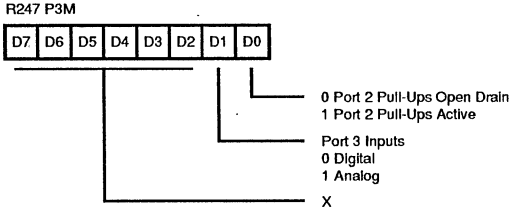


Figure 29. Port 3 Mode Register
(F7H: Write Only)

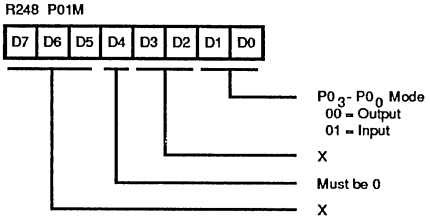


Figure 30. Port 0 and 1 Mode Register (F8H: Write Only)

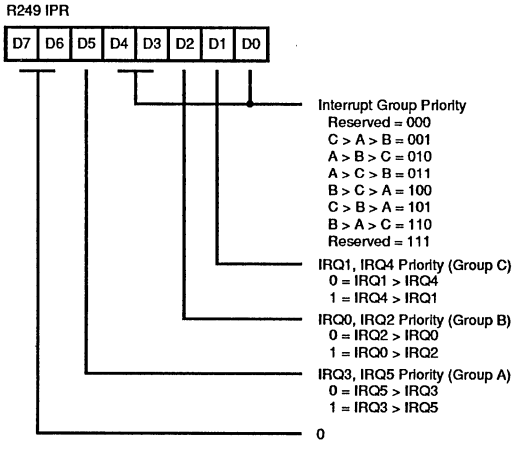


Figure 31. Interrupt Priority Register (F9H: Write Only)

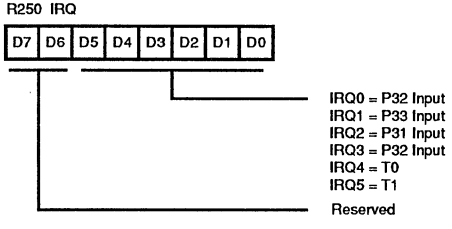


Figure 32. Interrupt Request Register (FAH: Read/Write)

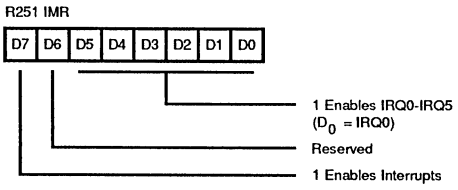


Figure 33. Interrupt Mask Register (FBH: Read/Write)

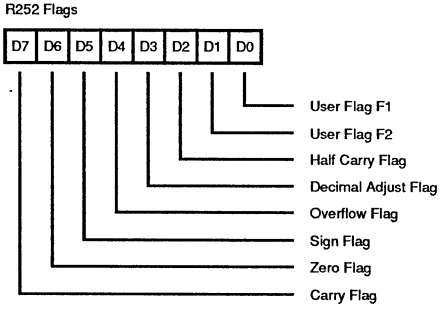


Figure 34. Flag Register (FCH: Read/Write)

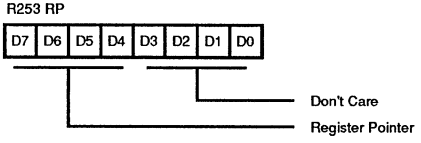


Figure 35. Register Pointer (FDH: Read/Write)

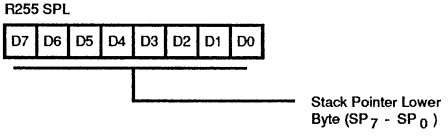


Figure 36. Stack Pointer (FFH: Read/Write)

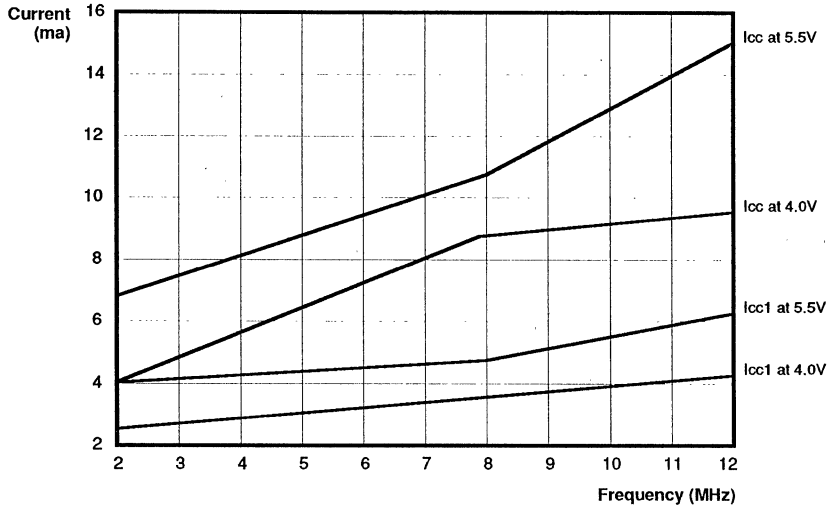


Figure 37. Maximum I_{CC} and I_{CC1} vs Frequency in Standard Mode

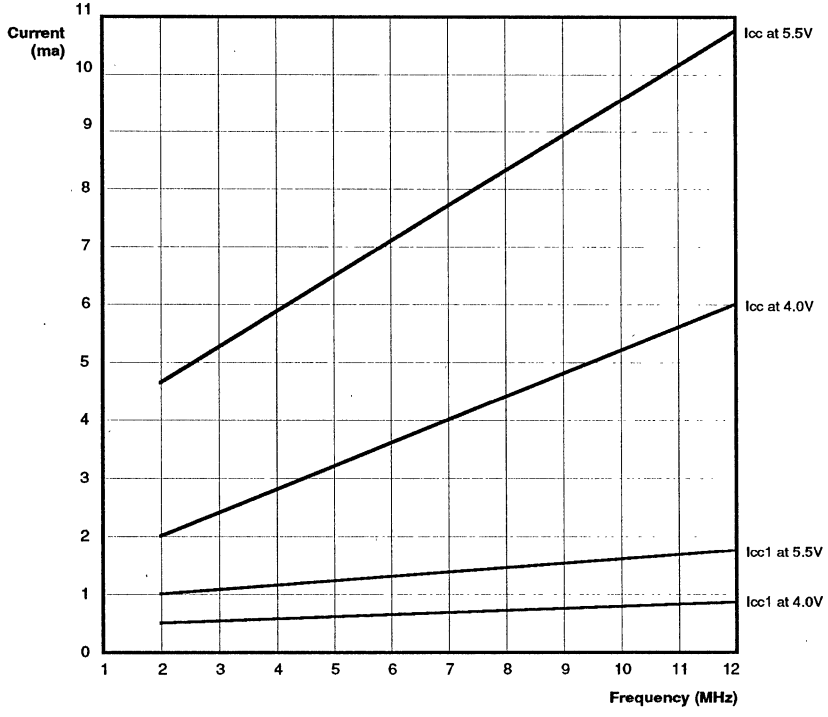


Figure 38. Typical I_{CC} and I_{CC1} vs Frequency in Standard Mode

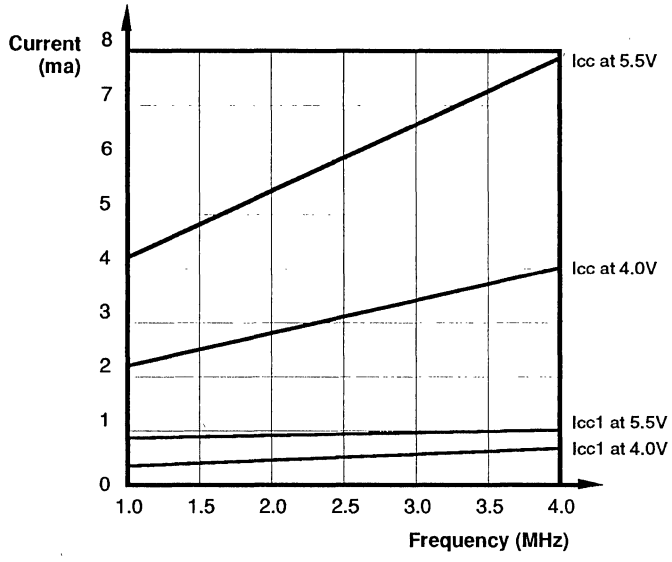


Figure 39. Typical I_{cc} and I_{cc1} vs Frequency in Low EMI Mode

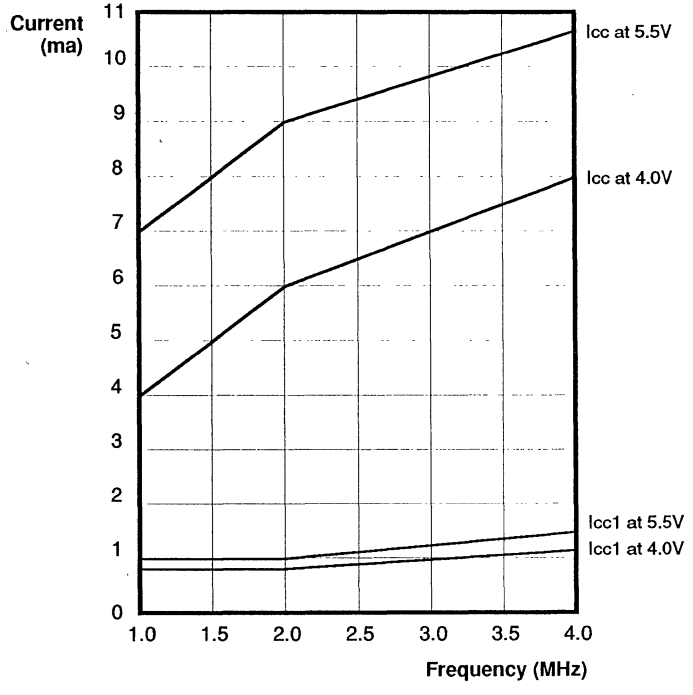


Figure 40. Maximum I_{cc} and I_{cc1} vs Frequency in Low EMI Mode

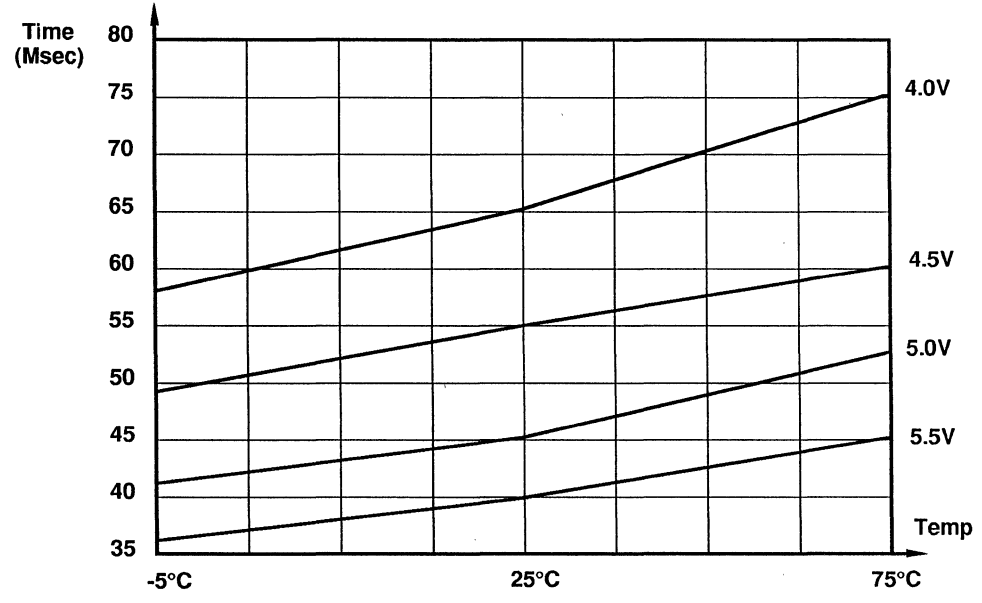


Figure 41. Typical POR Time Out Period vs Temperature

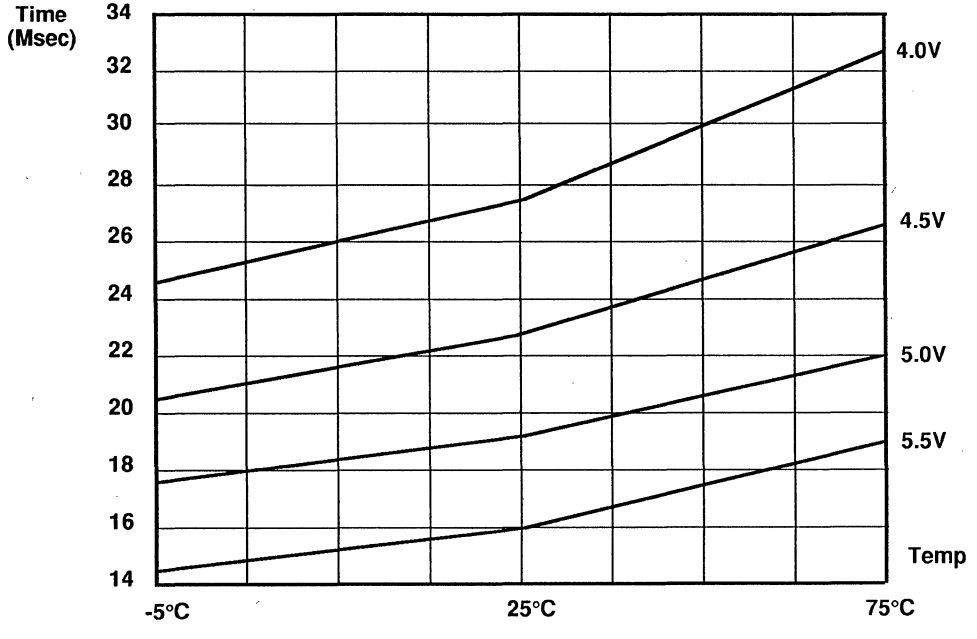


Figure 42. Typical WDT Time Out Period vs Temperature

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

Symbol	Meaning
IRR	Indirect register pair or indirect working-register pair address
Ir	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

Symbol	Meaning
dst	Destination location or contents
src	Source location or contents
cc	Condition code
@	Indirect address prefix
SP	Stack pointer
PC	Program counter
FLAGS	Flag register (Control Register 252)
RP	Register Pointer (R253)
IMR	Interrupt mask register (R251)

Flags. Control register (R252) contains the following six flags.

Symbol	Meaning
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

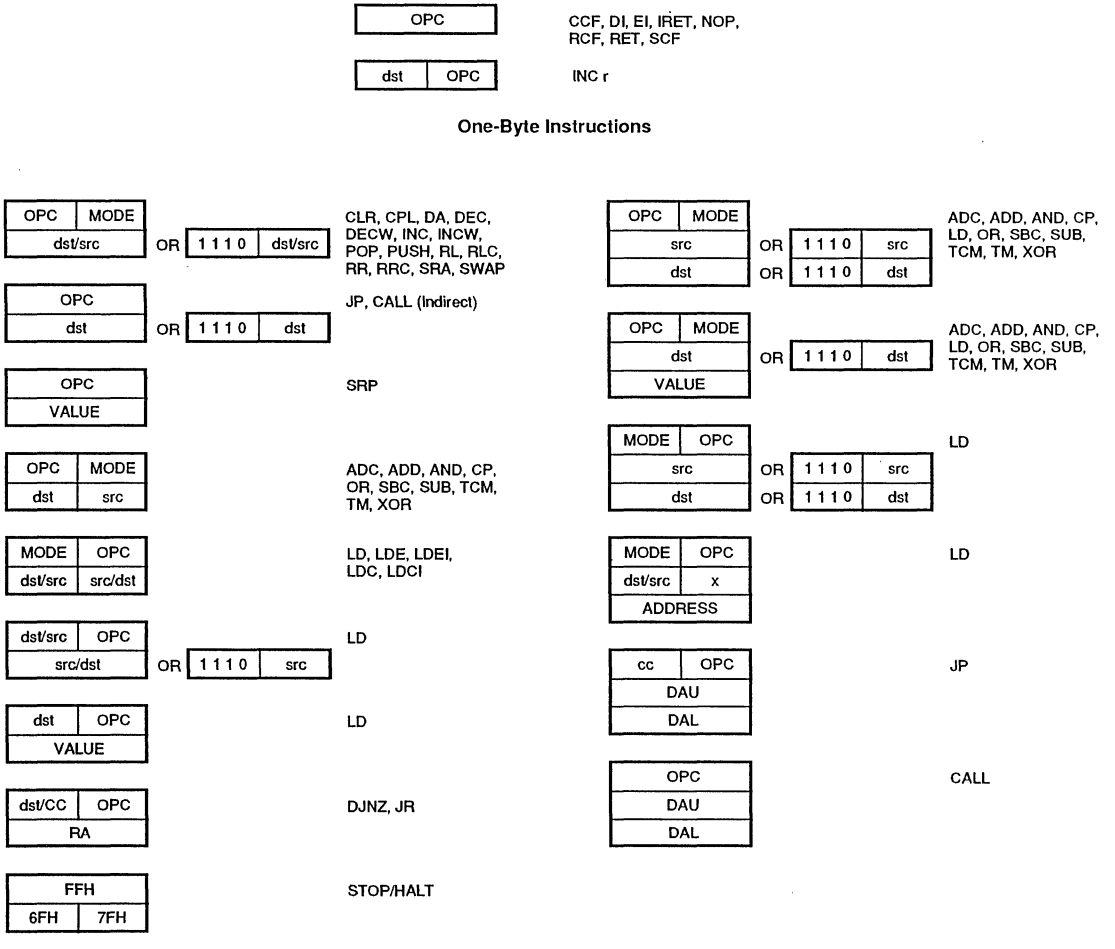
Affected flags are indicated by:

0	Clear to zero
1	Set to one
*	Set to clear according to operation
-	Unaffected
X	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000	---	Always true	---
0111	C	Carry	C=1
1111	NC	No Carry	C=0
0110	Z	Zero	Z=1
1110	NZ	Not zero	Z=0
1101	PL	Plus	S=0
0101	MI	Minus	S=1
0100	OV	Overflow	V=1
1100	NOV	No overflow	V=0
0110	EQ	Equal	Z=1
1110	NE	Not equal	Z=0
1001	GE	Greater than or equal	(S XOR V)=0
0001	LT	Less than	(S XOR V)=1
1010	GT	Greater than	[Z OR (S XOR V)]=0
0010	LE	Less than or equal	[Z OR (S XOR V)]=1
1111	UGE	Insigned greater than or equal	C=0
0111	ULT	Unsigned less than	C=1
1011	UGT	Unsigned greater than	(C=0 AND Z=0)=1
0011	ULE	Unsigned less than or equal	(C OR Z)=1
0000	---	Never true	---

INSTRUCTION FORMATS



INSTRUCTION SUMMARY

Note: Assignment of a value is indicated by the symbol "---". For example:

dst --- dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr (n)" is used to refer to bit (n) of a given operand location. For example:

dst(7)

refers to bit 7 of the destination operand.

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst, src dst←dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst←dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst←dst AND src	†		5[ ]	-	*	*	0	-	-
CALL dst SP←SP - 2 @SP←PC, PC←dst	DA IRR		D6 D4	-	-	-	-	-	-
CCF C←NOT C			EF	*	-	-	-	-	-
CLR dst dst←0	R IR		B0 B1	-	-	-	-	-	-
COM dst dst←NOT dst	R IR		60 61	-	*	*	0	-	-
CP dst, src dst - src	†		A[ ]	*	*	*	*	-	-
DA dst dst←DA dst	R IR		40 41	*	*	*	X	-	-
DEC dst dst←dst - 1	R IR		00 01	-	*	*	*	-	-
DECW dst dst←dst - 1	RR IR		80 81	-	*	*	*	-	-
DI IMR(7)←0			8F	-	-	-	-	-	-
DJNZr, dst r←r - 1 if r≠0 PC←PC + dst Range: +127, -128	RA		rA r = 0 - F	-	-	-	-	-	-
EI IMR(7)←1			9F	-	-	-	-	-	-
HALT			7F	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
INC dst dst←dst + 1	r R IR		rE r = 0 - F 20 21	-	*	*	*	-	-
INCW dst dst←dst + 1	RR IR		A0 A1	-	*	*	*	-	-
IRET FLAGS←@SP; SP←SP + 1 PC←@SP; SP←SP + 2; IMR(7)←1			BF	*	*	*	*	*	*
JP cc, dst if cc is true, PC←dst	DA IRR		cD c = 0 - F 30	-	-	-	-	-	-
JR cc, dst if cc is true, PC←PC + dst Range: +127, -128	RA		cB c = 0 - F	-	-	-	-	-	-
LD dst, src dst←src	r r R r X r r Ir R R IR IR	Im R r X r Ir r R IR IM IM R	rC r8 r9 r = 0 - F C7 D7 E3 F3 E4 E5 E6 E7 F5	-	-	-	-	-	-
LDC dst, src dst←src	r	Irr	C2	-	-	-	-	-	-
LDCI dst, src dst←src r←r + 1;rr←rr + 1	Ir	Irr	C3	-	-	-	-	-	-
NOP			FF	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
OR dst, src dst←dst OR src	†		4[ ]	-	*	*	0	-	-
POP dst dst←@SP; SP←SP + 1	R IR		50 51	-	-	-	-	-	-
PUSH src SP←SP - 1; @SP←src	R IR		70 71	-	-	-	-	-	-
RCF C←0			CF	0	-	-	-	-	-
RET PC←@SP; SP←SP + 2			AF	-	-	-	-	-	-
RL dst	R IR		90 91	*	*	*	*	-	-
RLC dst	R IR		10 11	*	*	*	*	-	-
RR dst	R IR		E0 E1	*	*	*	*	-	-
RRC dst	R IR		C0 C1	*	*	*	*	-	-
SBC dst, src dst←dst←src←C	†		3[ ]	*	*	*	*	1	*
SCF C←1			DF	1	-	-	-	-	-
SRA dst	R IR		D0 D1	*	*	*	0	-	-
SRP dst RP←src	Im		31	-	-	-	-	-	-
STOP			6F	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
SUB dst, src dst←dst←src	†		2[ ]	*	*	*	*	1	*
SWAP dst	R IR		F0 F1	X	*	*	X	-	-
TCM dst, src (NOT dst) AND src	†		6[ ]	-	*	*	0	-	-
TM dst, src dst AND src	†		7[ ]	-	*	*	0	-	-
WDH			4F	-	-	-	-	-	-
WDT			5F	-	-	-	-	-	-
XOR dst, src dst←dst XOR src	†		B[ ]	-	*	*	0	-	-

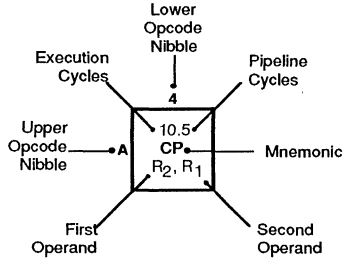
† These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address Mode dst	src	Lower Opcode Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

OPCODE MAP

Upper Nibble (Hex) \ Lower Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								4.0 **WDH**
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								5.0 **WDT**
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								6.0 **STOP**
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								7.0 **HALT**
8	10.5 **DECW** RR1	10.5 **DECW** IR1														6.1 **DI**
9	6.5 **RL** R1	6.5 **RL** IR1														6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14.0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2				10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1			20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6.5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2			3					2					3	1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_1 = Dst address
R_2 or r_2 = Src address

Sequence:
Opcode, First Operand,
Second Operand

Note: Blank areas not defined.

* 2-byte instruction appears as a 3-byte instruction

Z86C09/C19

CMOS Z8® 8-BIT MICROCONTROLLER

FEATURES

- 8-bit CMOS microcontroller, 18-pin DIP
- Low cost
- 3.0 to 5.5 volt operating range
- Low power consumption-50 mW (typical)
- Fast instruction pointer, 1.0 microseconds @ 12 MHz
- Two standby modes - STOP and HALT
- 14 input/output lines (2 with Comparator inputs)
- All digital inputs are CMOS levels and Schmitt triggered
- 2K, 4 Kbytes of ROM, Z86C09, Z86C19, respectively
- 124 bytes of RAM
- Two Expanded Register File Control Registers
- Two programmable 8-bit Counter/Timers
- 6-bit programmable prescaler
- 6 vectored, priority interrupts from five different sources
- Clock speeds 8 and 12 MHz
- Brown-out protection
- Watchdog/Power-On Reset Timer
- Two Comparators with programmable interrupt polarity
- On-chip oscillator that accepts a crystal, ceramic resonator, LC, RC, or external clock drive.

GENERAL DESCRIPTION

The Z86C09 and Z86C19 Consumer Controller Processors (CCP™) introduce a new level of sophistication to single-chip architecture. The Z86C09 and Z86C19 are members of the Z8 single-chip microcontroller family with 2K and 4K bytes of ROM, respectively, and 124 bytes of RAM. The devices are housed in a 18-pin DIP, and are CMOS compatible. Zilog's CMOS microcontroller offers fast execution, more efficient use of memory, more sophisticated interrupts, input/output bit manipulation capabilities, and easy hardware/software system expansion along with low cost and low power consumption.

The Z86C09/C19 architecture is characterized by Zilog's 8-bit microcontroller core with an Expanded Register File to allow access to register mapped peripheral and I/O circuits. The CCP offers a flexible I/O scheme, and a number of ancillary features that are useful in many industrial, automotive, and advanced scientific applications.

The device applications demand powerful I/O capabilities. The CCP fulfills this with 14 pins dedicated to input and output. These lines are grouped into two ports, and are configurable under software control to provide timing, status signals, or parallel I/O.

Three basic address spaces are available to support this wide range of configurations; Program Memory, Register File, and Expanded Register File. The Register File is composed of 124 bytes of General-Purpose Registers, two I/O Port registers and fifteen Control and Status registers. The Expanded Register File consists of two control registers.

To unburden the program from coping with real-time problems such as counting/timing and input/output data communication, the Z86C09/C19 offers two on-chip counter/timers with a large number of user selectable modes, and two on-board comparators that can process analog signals with a common reference voltage (Figure 1).

GENERAL DESCRIPTION (Continued)

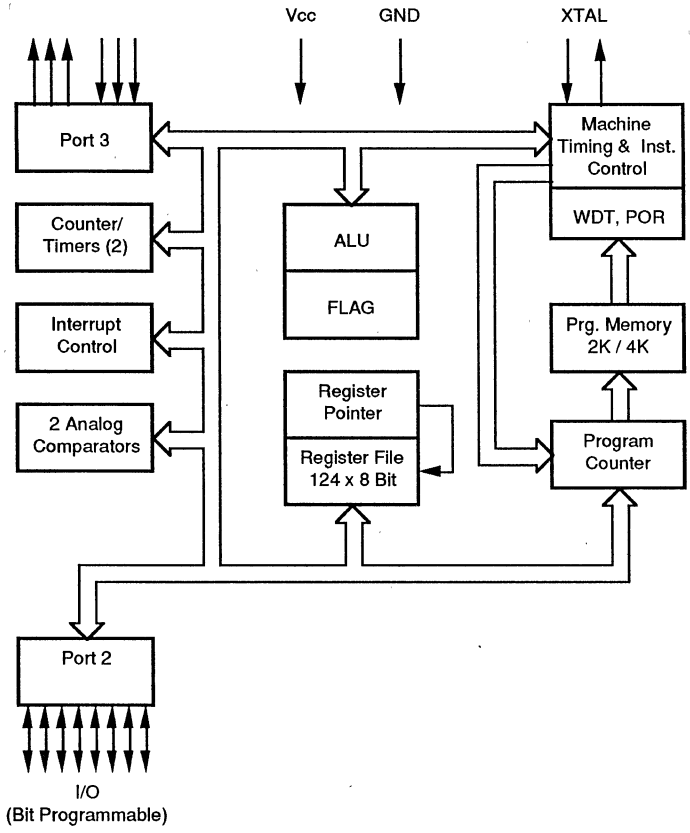


Figure 1. Functional Block Diagram

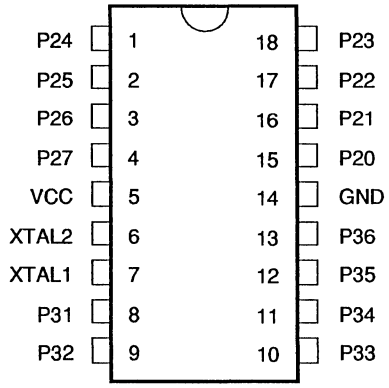


Figure 2. Pin Configuration

PIN DESCRIPTION

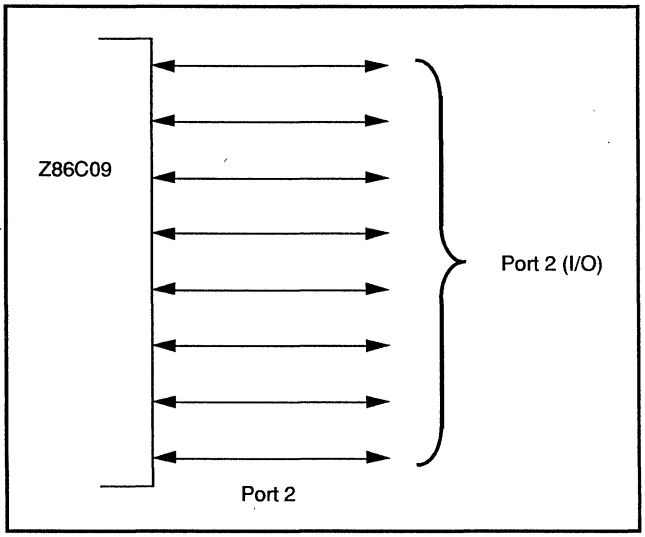
Table 1. Pin Identification

No	Symbol	Function	Direction
1-4	P24-7	Port 2 pin 4, 5, 6, 7	In/Output
5	V_{cc}	Power Supply	Input
6	XTAL2	Crystal Oscillator Clock	Output
7	XTAL1	Crystal Oscillator Clock	Input
8-10	P31-3	Port 3 pin 1, 2, 3	Fixed Input
11-13	P34-6	Port 3 pin 4, 5, 6	Fixed Output
14	GND	Ground	Input
15-18	P20-3	Port 2 pin 0, 1, 2, 3	In/Output

XTAL1. *Crystal 1* (time-based input). This pin connects a parallel-resonant crystal, ceramic resonator, LC or RC network or an external single-phase clock to the on-chip oscillator input.

XTAL2. *Crystal 2* (time-based output). This pin connects a parallel-resonant crystal, ceramic resonator, LC or RC network to the on-chip oscillator output.

Port 2 P20-P27. Port 2 is an 8-bit, bidirectional, CMOS compatible I/O port. These 8 I/O lines can be configured under software control to be an input or output, independently. Input buffers are Schmitt-triggered. Bits programmed as outputs may be globally programmed as either push-pull or open drain (Figure 3).



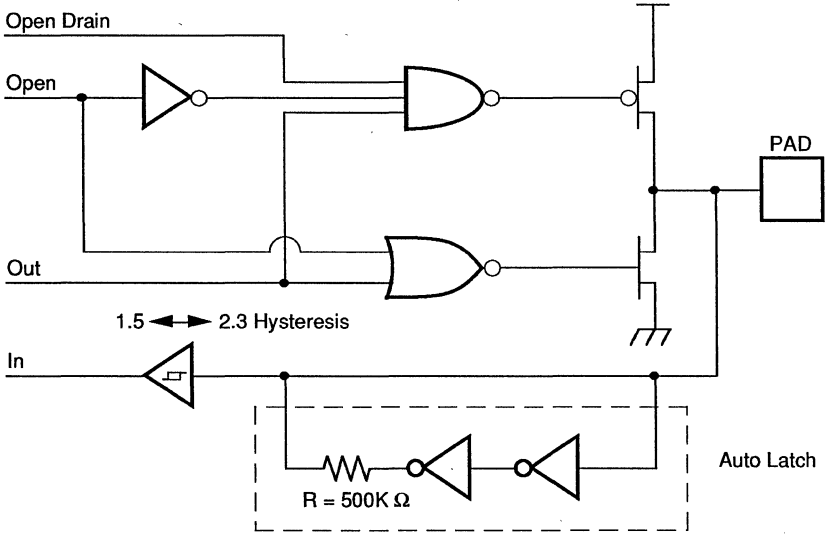


Figure 3. Port 2 Configuration

Auto-Latch. The auto-latch puts valid CMOS levels on all CMOS inputs that are not externally driven. This will reduce excessive supply current flow in the input buffer when it is not been driven by any source.

Port 3 P31-P36. Port 3 is a 6-bit, CMOS compatible port with three fixed input and three fixed output lines. These 6 lines consist of three fixed input (P31-P33) and three fixed output port (P34-P36) lines. Pins P31,P32 and P33 are standard CMOS inputs and pins P34,P35,and P36 are push-pull outputs. Two on-board comparators can process analog signals on P31 and P32 with reference to the voltage on P33. The analog function is enabled by programming Port 3 Mode Register (bit 1). Pins P31 and P32 are programmable as falling, rising, or both edge triggered interrupts (IRQ register bits 6 and 7). P33 is the comparator reference voltage input. Access to Counter/Timer 1 is made through P31 (Tin) and P36 (Tout), (Figure 4).

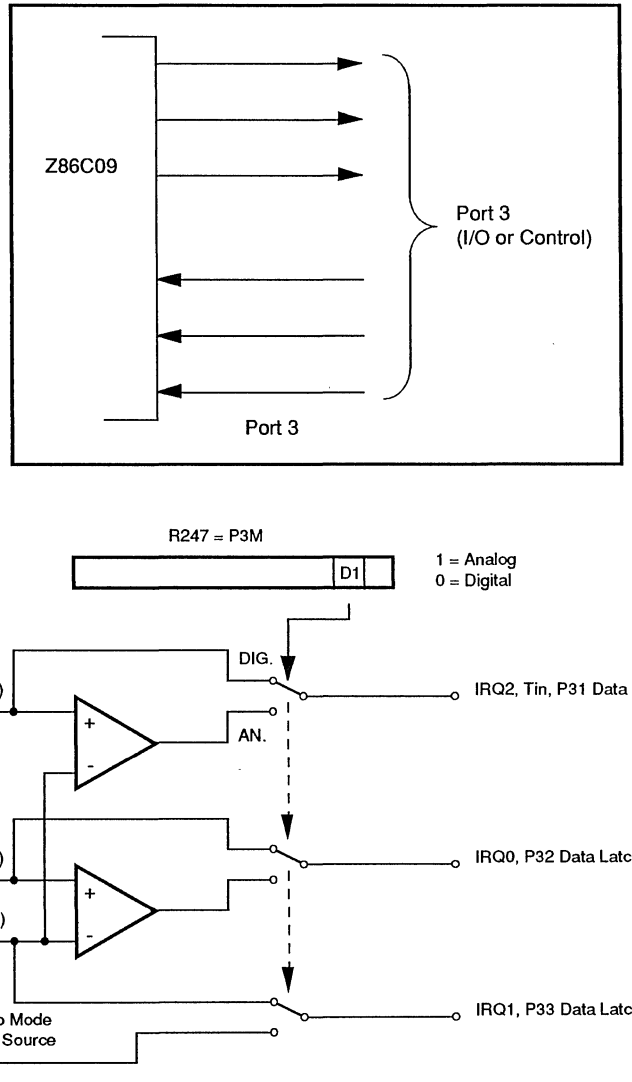


Figure 4. Port 3 Configuration

PIN DESCRIPTION (Continued)

Comparator Inputs. Port 3, Pin P31 and Pin P32 each have a comparator front end. The comparator reference voltage Pin P33 is common to both comparators. In analog mode, the P33 input functions as a reference voltage to the comparators. The internal P33 register and its corresponding IRQ1 is connected to the STOP Mode Recovery source selected by the SMR. In this mode, any of the STOP Mode Recovery sources can be used to toggle the P33 bit or generate IRQ1. In digital mode, Pin P33 can be used as a P33 register input or IRQ1 source (Figure 13).

Auto-Latch. The auto-latch puts valid CMOS levels on all CMOS inputs that are not externally driven. This reduces excessive supply current flow in the input buffer when it is not being driven by any source.

FUNCTIONAL DESCRIPTION

The Z8 CCP incorporates special functions to enhance the Z8's application in industrial, scientific research, and advanced technologies applications.

RESET. The device is reset in one of the following conditions:

- Power-On Reset
- Watch-Dog Timer
- STOP Mode Recovery Source

The device does not re-initialize the WDTMR, SMR, P2M, or P3M registers to their reset values on a STOP Mode Recovery operation.

Program Memory. Z86C09/C19 can address up to 2K/4K bytes of internal program memory respectively (Figure 5). The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts. Byte 13 to byte 2048/4096 consists of on-chip mask-programmed ROM.

The 2K/4K bytes of Program Memory is mask programmable. A ROM protect feature will prevent "dumping" of the ROM contents by inhibiting execution of LDC, LDCI, LDE, and LDEI instructions to program memory in all modes.

Expanded Register File. The register file has been expanded to allow for additional system control registers and for mapping of additional peripheral devices and input/output ports into the register address area. The Z8 register address space R0 through R15 is implemented as 16 groups of 16 registers per group (Figure 6). These register groups are known as the ERF (Expanded Register File). Bits 3:0 of the Register Pointer (RP) select the active ERF group. Bits 7:4 of register RP select the working register group (Figure 7). Two system configuration registers reside in the Expanded Register File address space at bank F. The rest of the Expanded Register addressing space is not physically implemented, and is open for future expansion.

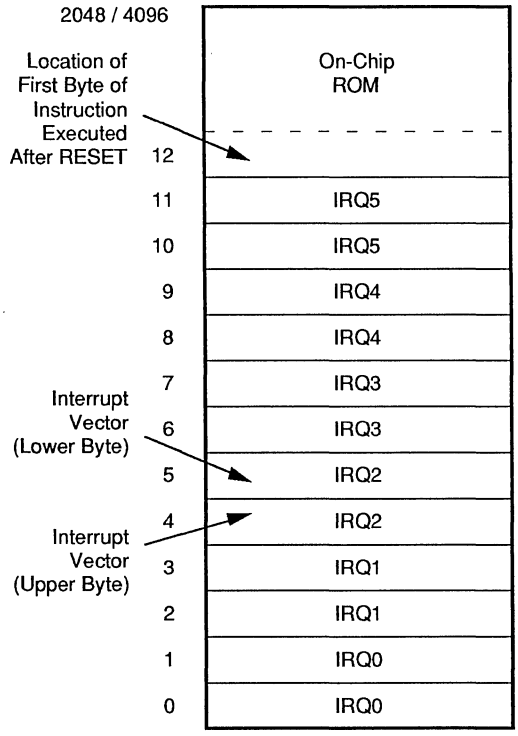


Figure 5. Program Memory Map

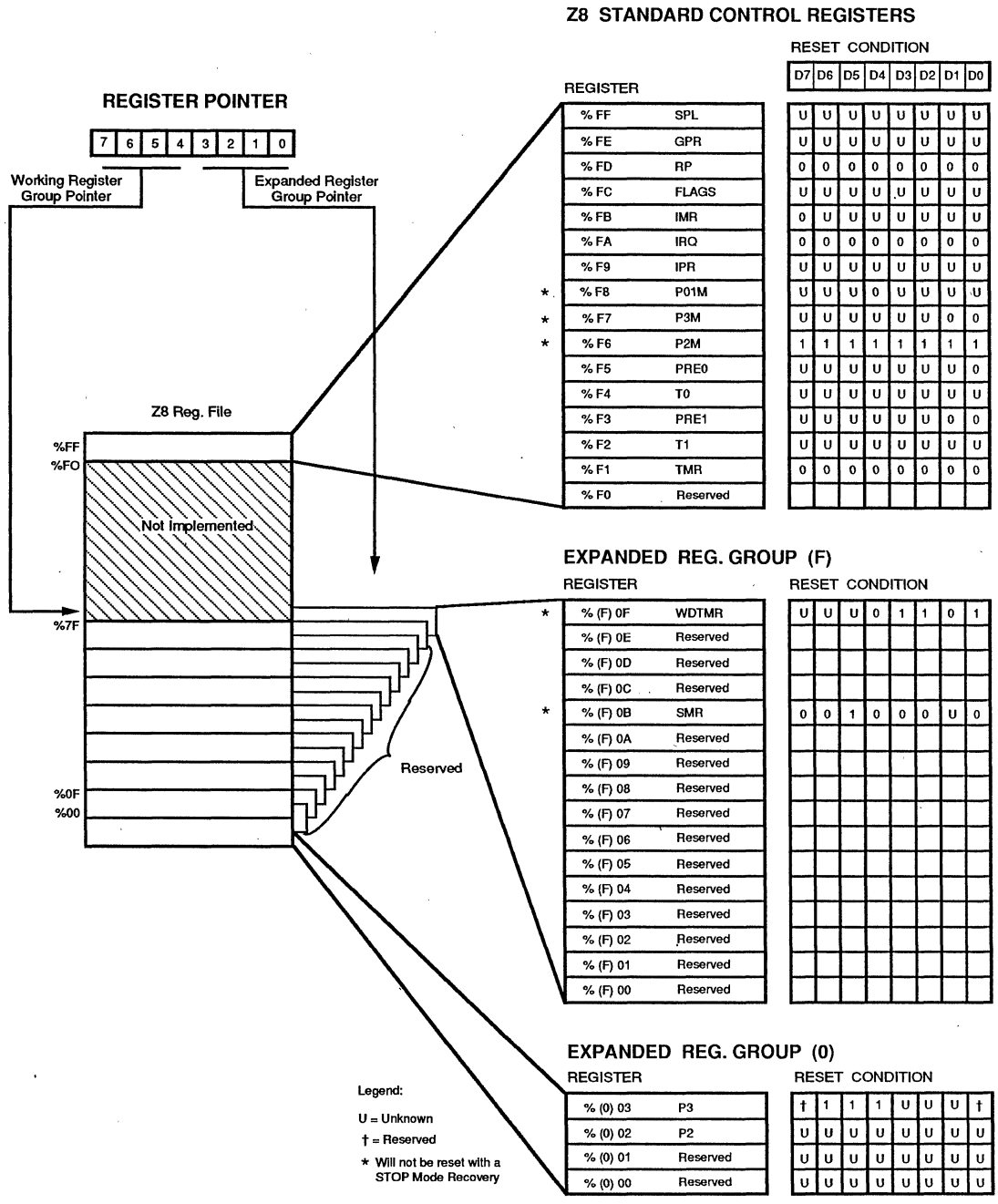
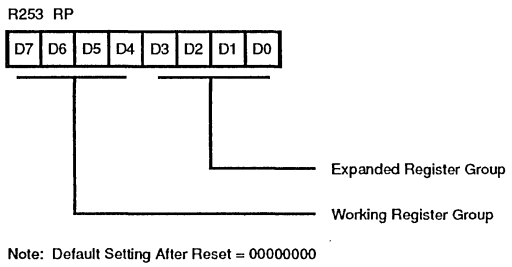


Figure 6. Expanded Register File Architecture



Note: Default Setting After Reset = 00000000

Figure 7. Register Pointer Register

Register File. The Register File consists of two I/O port registers, 124 general purpose registers, and 15 control and status registers, and two system configuration registers in the Expanded Register Group (Figure 6). The instructions can access registers directly or indirectly via an 8-bit address field. This allows a short 4-bit register address using the Register Pointer (Figure 8). In the 4-bit mode, the Register File is divided into 16 working register groups, each occupying 16 continuous locations. The Register Pointer addresses the starting location of the active working-register group.

Caution: D4 of Control Register P01M (R251) must be "0". If the Z86C09/19 is emulated by Z86C90, D4 of P01M has to change to "0" before submission to ROM code.

GPR. The Z86C09/C19 has one extra General Purpose Register located at %FE(R254).

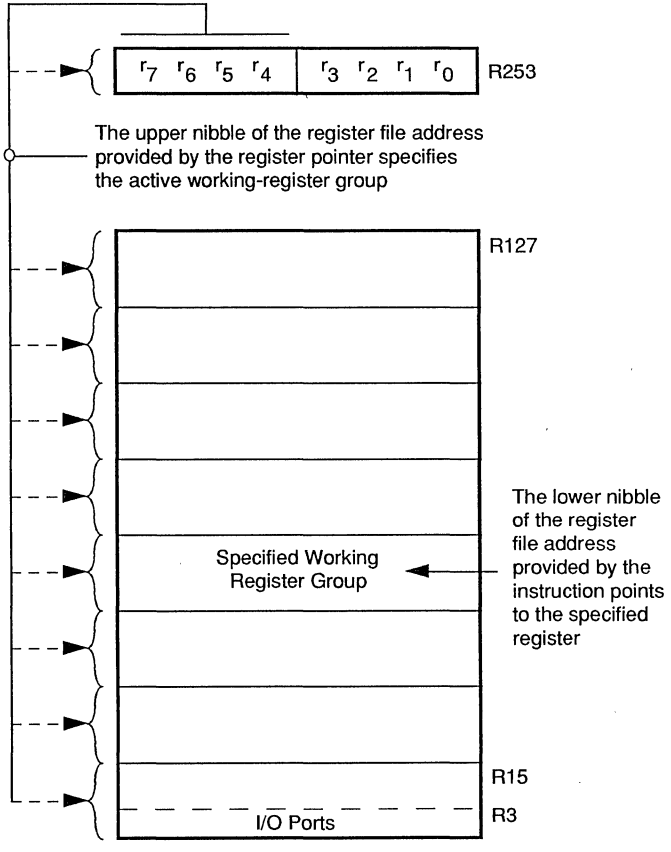


Figure 8. Register Pointer

Stack. The Z86C09/C19 has an 8-bit Stack Pointer (R255) used for the internal stack that resides within the 124 general-purpose registers.

Counter/Timers. There are two 8-bit programmable counter/timers (T0-T1), each driven by its own 6-bit programmable prescaler. The T1 prescaler can be driven by internal or external clock sources, however, the T0 prescaler is driven by the internal clock only (Figure 9).

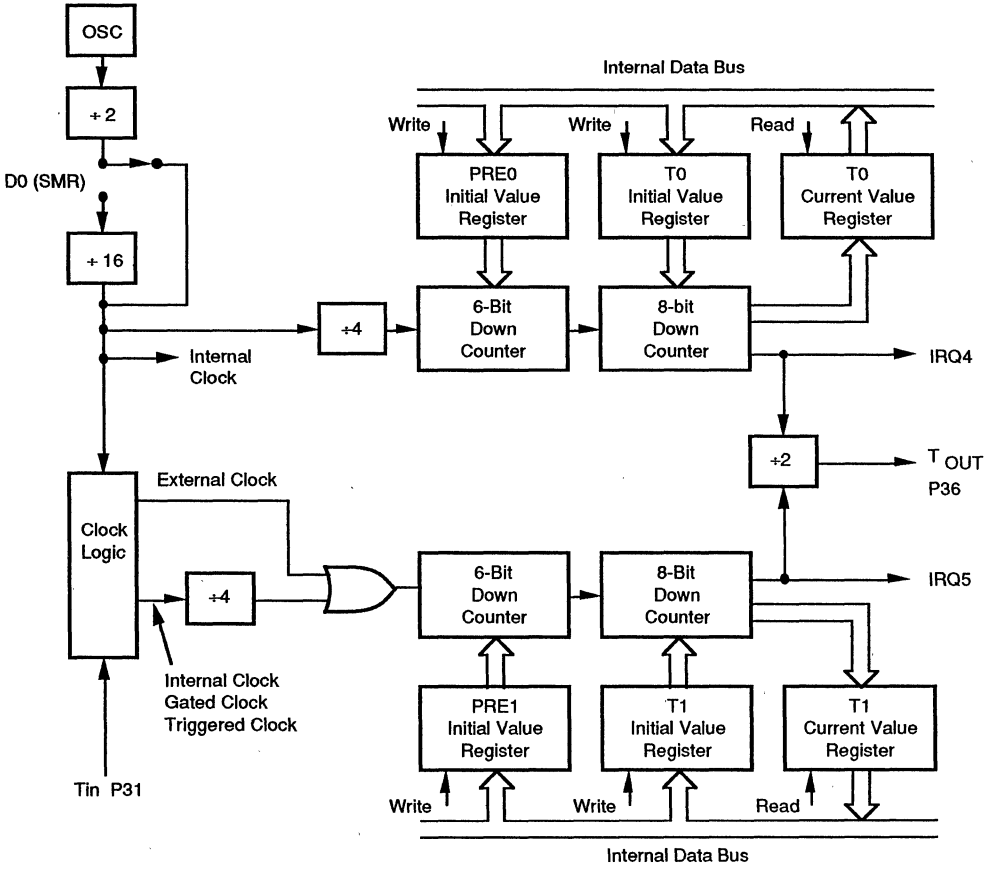


Figure 9. Counter/Timer Block Diagram

The 6-bit prescalers can divide the input frequency of the clock source by any integer number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request-IRQ4 (T0) or IRQ5 (T1), is generated.

The counters can be programmed to start, stop, restart to continue, or restart from the initial value. The counters can also be programmed to stop upon reaching zero (single-pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode).

The counters, but not the prescalers, can be read at any time without disturbing their value or count mode. The clock source for T1 is user-definable and can be either the internal microprocessor clock divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input (P31) as an external clock, a trigger input that can be retriggerable or not-retriggerable, or as a gate input for the internal clock. Port 3 line P36 serves as a timer output (Tout) through which T0, T1 or the internal clock can be output. The counter/timers can be cascaded by connecting the T0 output to the input of T1.

Interrupts. The Z86C09/Z86C19 has six different interrupts from five different sources. The interrupts are mask-able and prioritized (Figure 10). The five sources are divided as follows; three sources are claimed by Port 3 lines P31-P33, and two sources in counter/timers. The Interrupt Mask Register globally or individually enables or disables the six interrupt requests (Table 2).

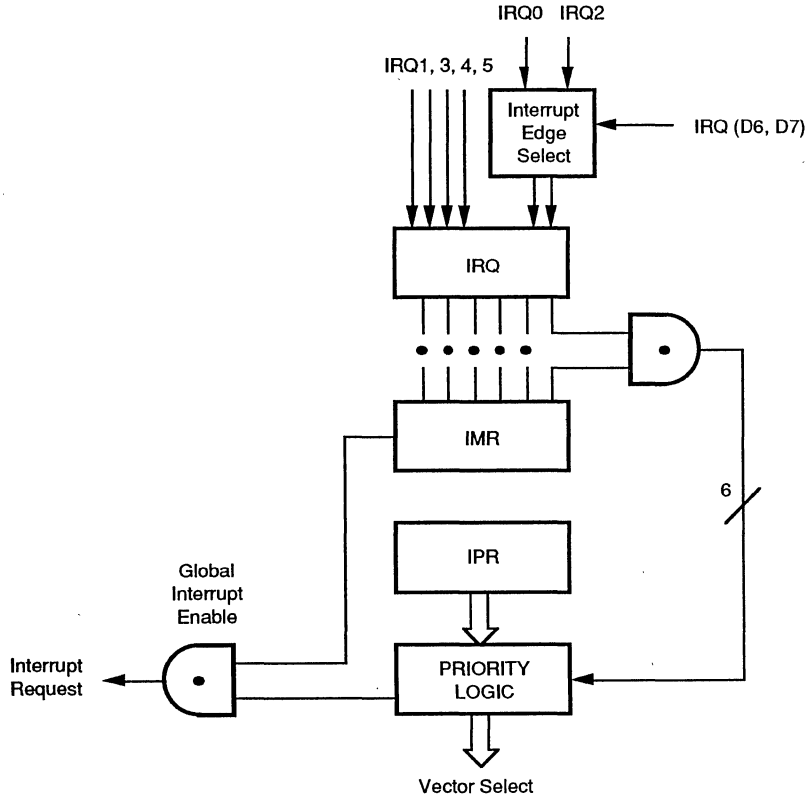


Figure 10. Interrupt Block Diagram

Table 2. Interrupt Types, Sources, and Vectors

Name	Source	Vector Location	Comments
IRQ 0	IRQ 0	0, 1	External (P32), ≠ Ø Edge Triggered
IRQ 1	IRQ 1	2, 3	External (P33), Ø Edge Triggered
IRQ 2	IRQ 2, TIN	4,5	External (P31), ≠ Ø Edge Triggered
IRQ 3		6, 7	Software Generated Only
IRQ 4	T0	8, 9	Internal
IRQ 5	TI	10, 11	Internal

When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register. An interrupt machine cycle is activated when an interrupt request is granted. This disables all subsequent interrupts, saves the Program Counter and Status Flags, and then branches to the program memory vector location reserved for that interrupt. All Z86C09/C19 interrupts are vectored through locations in the program memory. This memory location and the next byte contain the 16-bit starting address of the interrupt service routine for that particular interrupt request.

To accommodate polled interrupt systems, interrupt inputs are masked and the interrupt request register is polled to determine which of the interrupt requests needs services. IRQ3 has no hardware source but can be invoked by software (write to IRQ3 Register).

An interrupt resulting from AN1 is mapped into IRQ2, and an interrupt from AN2 is mapped into IRQ0. Interrupts IRQ2 and IRQ0 may be rising, falling, or both edge triggered, and are programmable by the user. The software may poll to identify the state of the pin.

The programming bits for the INTERRUPT EDGE SELECT are located in the IRQ register (R250), bits D7 and D6. The configuration is shown in Table 3.

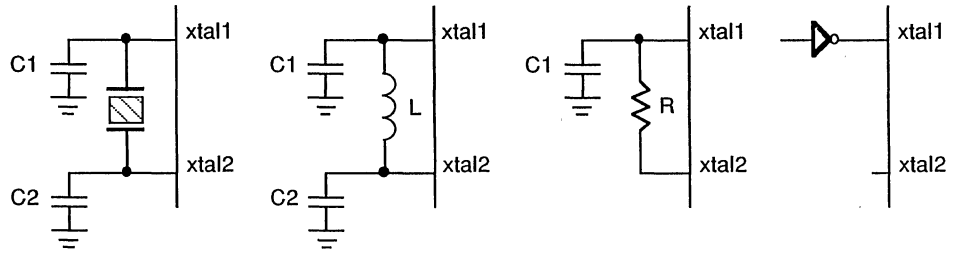
Table 3. IRQ Register

IRQ		Interrupt Edge	
D7	D6	P31	P32
0	0	F	F
0	1	F	R
1	0	R	F
1	1	R/F	R/F

Notes:
F = Falling Edge
R = Rising Edge

Clock. The Z86C09/C19 on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal, RC, ceramic resonator, or any suitable external clock source (XTAL1 = Input, XTAL2 = Output). The crystal should be AT cut, 10 KHz to 12 MHz max, with a series resistance (RS) less than or equal to 100 Ohms.

The crystal should be connected across XTAL1 and XTAL2 using the recommended capacitors (C1 is more than or equal to 22 pf) from each pin to ground. The RC oscillator option is mask-programmable, to be selected by the customer at the time the ROM code is submitted. The RC oscillator configuration must be an external resistor connected from XTAL1 to XTAL2, with a frequency-setting capacitor from XTAL1 to ground (Figure 11). The RC value vs Frequency curves are shown in Figure 48 and 49. (**Limitation:** The RC option is not avaiable in the 12 MHz part.)



Ceramic Resonator or Crystal	LC	RC	External Clock
C1, C2 = 47 pf TYP *	C1, C2 = 22 pf	@ 5V VCC (TYP)	
F = 8 MHz	L = 130 µH *	C1 = 33 pf	
	F = 3 MHz *	R = 1K	
		F = 16 MHz	

* Preliminary Value Including Pin Parasitics

Figure 11. Oscillator Configuration

Power-On Reset. A timer circuit clocked by a dedicated on-board RC oscillator or by the XTAL oscillator is used for the Power-On Reset (POR) timer function. The POR time allows Vcc and the oscillator circuit to stabilize before instruction execution begins. The POR timer circuit is a one-shot timer triggered by one of the three conditions:

1. Power fail to Power OK status
2. STOP mode recovery (If D5 of SMR=1)
3. WDT timeout

The POR time is a nominal 5mS. Bit 5 of the Stop Mode Register determines whether the POR timer is bypassed after STOP mode recovery (typical for external clock, and RC/LC oscillators with fast start up time).

HALT. Will turn off the internal CPU clock but not the XTAL oscillation. The counter/timers and external interrupt IRQ0, IRQ1, and IRQ2 remain active. The device is recovered by interrupts, either externally or internally generated.

STOP. This instruction turns off the internal clock and external crystal oscillation and reduces the standby current to 10 microamps or less. The Stop mode is terminated by a RESET only, either by WDT timeout, POR, or SMR recovery. This causes the processor to restart the application program at address 000C (HEX).

In order to enter STOP (or HALT) mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. To do this, the user must execute a NOP (opcode=FFH) immediately before the appropriate sleep instruction, i.e.:

```
FF  NOP; clear the pipeline
6F  STOP; enter STOP mode
```

or

```
FF  NOP; clear the pipeline
7F  HALT; enter HALT mode
```

Stop Mode Register (SMR). This register selects the clock divide value and determines the mode of STOP mode recovery (Figure 12). All bits are write only except Bit 7 which is Read only. Bit 7 is a flag bit that is hardware set on the condition of a STOP recovery and reset by a power-on cycle. Bit 6 controls whether a low level or high level is required from the recovery source. Bit 5 controls the reset delay after recovery. Bits 2,3, and 4 of the SMR specify the source of the STOP mode recovery signal. If the XTAL1 is used as a source to drive the POR counter, then the STOP Mode Recovery time is XTAL/512. The SMR is located in bank F of the Expanded Register Group at address 0BH.

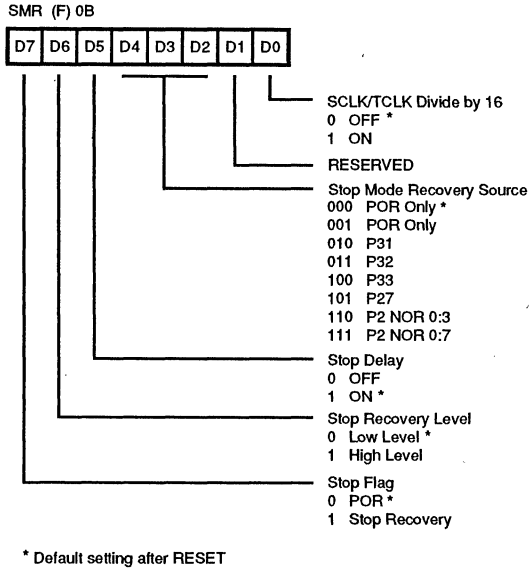


Figure 12. STOP Mode Recovery Register

SCLK/TCLK divide-by-16 select (D0). D0 of the SMR controls a divide-by-16 prescaler of SCLK/TCLK. The purpose of this control is to selectively reduce device power consumption during normal processor execution (SCLK control) and/or HALT mode (where TCLK sources the counter/timers and interrupt logic).

STOP Mode Recovery Source (D2, D3, and D4). These 3 bits of the SMR specify the wake-up source of the STOP Mode recovery (Figure 13 and Table 4).

Table 4. Stop Mode Recovery Source

SMR D4	SMR D3	SMR D2	Operation Description of action
0	0	0	POR recovery only
0	0	1	POR recovery only
0	1	0	P31 transition
0	1	1	P32 transition
1	0	0	P33 transition
1	0	1	P27 transition
1	1	0	Logical NOR of Port 2 bits 0:3
1	1	1	Logical NOR of Port 2 bits 0:7

P31-P33 can not wake up from STOP mode if the input lines are configured as analog input.

STOP Mode Recovery Delay Select (D5). This bit disables the 5mS RESET delay after STOP Mode Recovery. The default condition of this bit is 1.

STOP Mode Recovery Level Select (D6). A 1 in this bit position indicates that a high level on any one of the recovery sources wakes the device from STOP mode. A 0 indicates low level recovery. The default is 0 on POR. (See Figure 13).

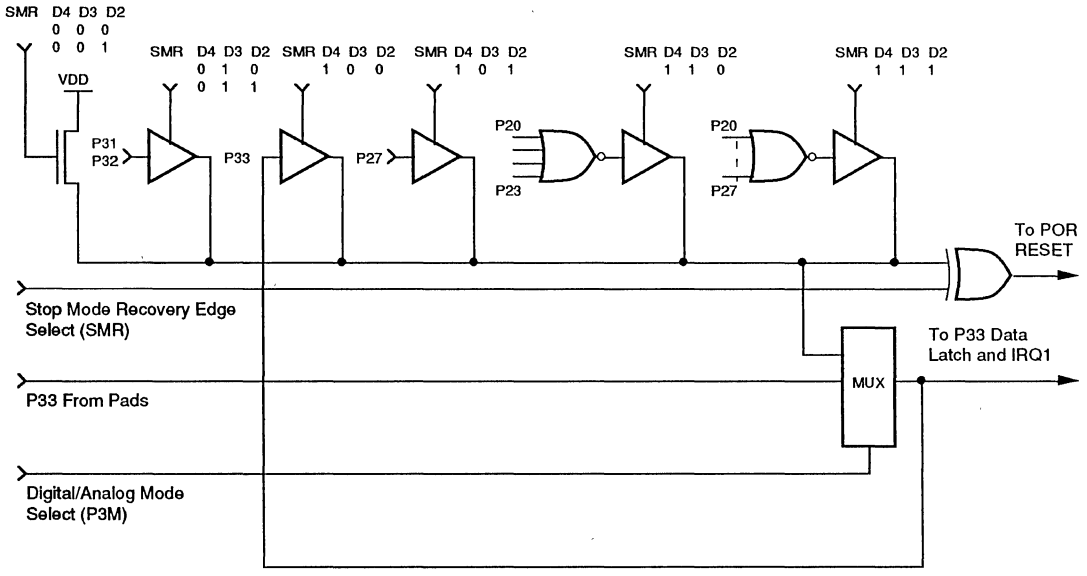


Figure 13. STOP Mode Recovery Source

Cold or Warm Start (D7). This bit is set by the device upon entering STOP mode. It is active high, and is 0 (cold) on POR/WDT RESET. This bit is a READ only. It is used to distinguish between cold or warm start.

Watch Dog Timer Mode Register (WDTMR). The WDT is a retriggerable one-shot timer that will reset the Z8 if it reaches its terminal count. The WDT is initially enabled by executing the WDT instruction and retriggered on subsequent executions of the WDT instruction. The timer circuit is driven by an on-board RC oscillator or external XTAL1 pin.

The POR clock source is selected with bit 4 of the WDT register. Bit 0 and 1 control a tap circuit that determines the timeout period. Bit 2 determines whether the WDT is active during HALT and Bit 3 determines WDT activity during STOP. Bits 5 through Bit 7 are reserved (Figure 14). This register is accessible only during the first 64 processor cycles (128 XTAL clocks) from the execution of the first instruction after Power-On-Reset, Watch Dog Reset or a Stop Mode Recovery (Figure 15). After this point, the register cannot be modified by any means, intentional or otherwise. The WDTMR cannot be read and is located in bank F of the Expanded Register Group at address location OFH. It is organized as follows:

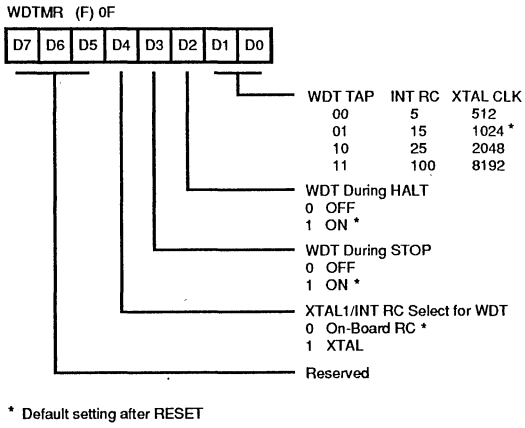


Figure 14. Watchdog Timer Mode Register

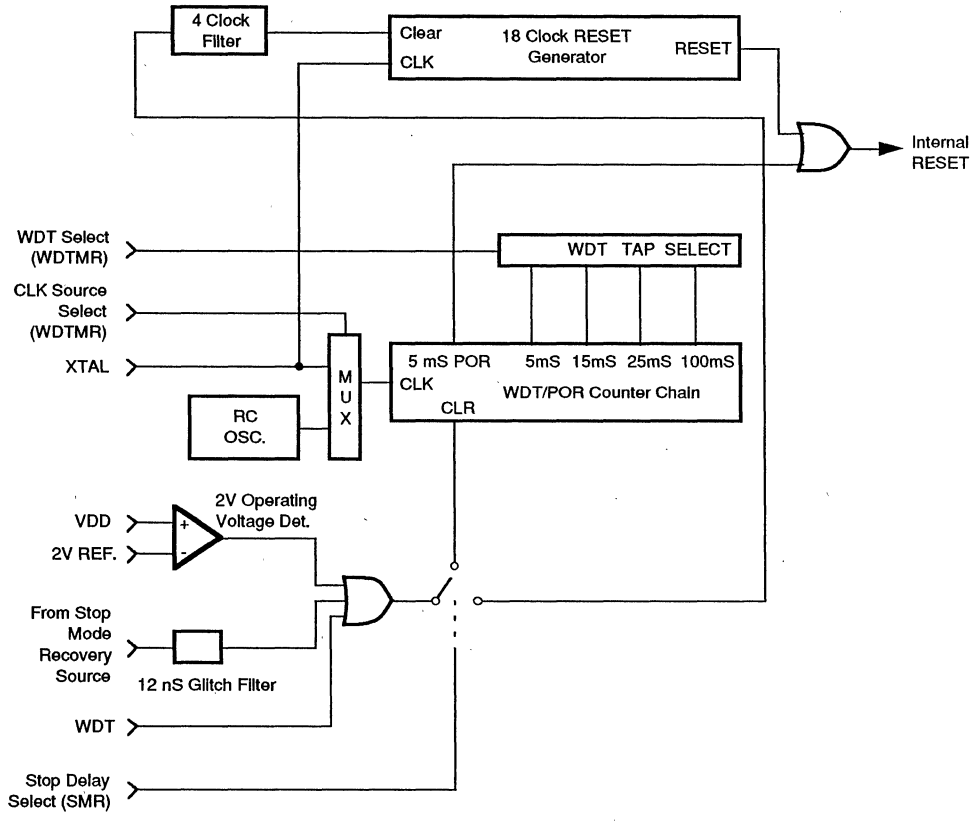


Figure 15. Resets and WDT

WDT Time Select (D1, D0). Selects the WDT time period. It is configured as shown in Table 5.

Table 5. WDT Time Select

D1	D0	Timeout Period (On-board RC) Clock Source	XTAL1 Clock Source
0	0	5mS min	XTAL1/512
0	1	15mS min	XTAL1/1024
1	0	25mS min	XTAL1/2048
1	1	100mS min	XTAL1/8192

Notes:
The default on a WDT initiated RESET is 15 mS.
See Figures 44 to 47 for details.

WDT During HALT (D2). This bit determines whether or not the WDT is active during HALT mode. A 1 indicates active during HALT. The default is 1.

WDT During STOP (D3). This bit determines whether or not the WDT is active during STOP mode. Since XTAL clock is stopped during STOP mode, the on-board RC has to be selected as the clock source to the POR counter. A 1 indicates active during STOP. The default is 1.

On-Board Power-On-Reset RC or External XTAL1 Oscillator Select (D4). This bit determines which oscillator source is used to clock the internal POR and WDT counter chain. If the bit is a 1, the internal RC oscillator is bypassed and the POR and WDT clock-source is driven from the external pin, XTAL1. The default configuration of this bit is 0, which selects the RC oscillator.

V_{CC} Voltage Comparator. An on-board Voltage Comparator checks that V_{CC} is at the required level to ensure correct operation of the device. Reset is globally driven if V_{CC} is below the specified voltage (typically 2.1V).

Brown Out Protection (V_{BO}). The brown out trip voltage (V_{BO}) will be less than 3 volts and above 1.4 volts under the following conditions.

Maximum (V_{BO}) Conditions:

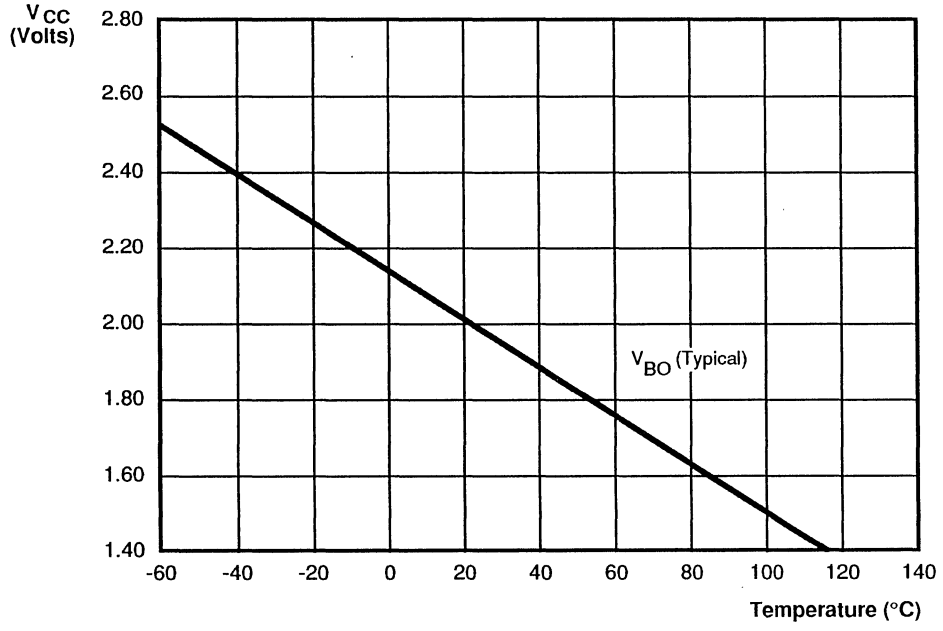
Case 1 T_A= -40, +105°C, Internal Clock Frequency equal or less than 1 MHz

Case 2 T_A= -40, +85°C, Internal Clock Frequency equal or less than 2 MHz

Note:
The internal clock frequency is one half the external clock frequency.

The device will function normally at or above 3.0V under all conditions. Below 3.0V, the device will function normally until the Brown Out Protection trip point (V_{BO}) is reached, for the temperatures and operating frequencies in case 1 and case 2 above. The device is guaranteed to function normally at supply voltages above the brown out trip point. The actual brown out trip point is a function of temperature and process parameters (Figure 16).

ROM Protect. ROM protect is mask-programmable. It is selected by the customer at the time the ROM code is submitted. **The selection of ROM protect will disable the LDC and LDCI instructions.**



* Power-on Reset threshold for V_{CC} and 4 MHz V_{BO} overlap

Figure 16. Typical Z86C19 V_{BO} Voltage Vs Temperature

ABSOLUTE MAXIMUM RATINGS

Symbol	Description	Min	Max	Units
V_{CC}	Supply Voltage *	-0.3	+7.0	V
TSTG	Storage Temp	-65	+150	C
T_A	Oper Ambient Temp	†		C

Notes:
* Voltage on all pins with respect to GND.
† See Ordering Information

Stress greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended period may affect device reliability.

STANDARD TEST CONDITIONS

The characteristics listed below apply for standard test conditions as noted. All voltages are referenced to ground. Positive current flows into the referenced pin (Figure 17).

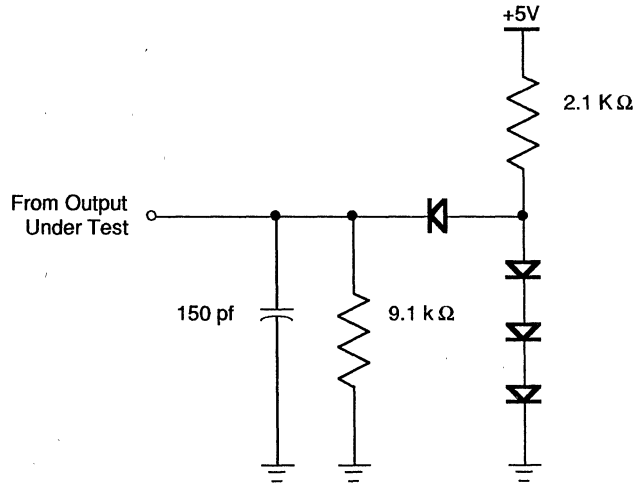


Figure 17. Test Load Configuration

DC ELECTRICAL CHARACTERISTICS
Z86C09/C19

Symbol	Parameter	V_{CC} Note [3]	T_A = 0°C to 70°C Min	T_A = 0°C to 70°C Max	T_A = -40°C to 105°C Min	T_A = -40°C to 105°C Max	Typical @ 25°C	Units	Conditions	Notes
	Max Input Voltage	3.3V		12		12		V	$I_{IN} \leq 250\ \mu A$	
		5.0V		12		12		V	$I_{IN} \leq 250\ \mu A$	
V_{CH}	Clock Input High Voltage	3.3V	0.9 V_{CC}	V_{CC}+0.3	0.9 V_{CC}	V_{CC}+0.3	2.4	V	Driven by External Clock Generator	
		5.0V	0.9 V_{CC}	V_{CC}+0.3	0.9 V_{CC}	V_{CC}+0.3	3.9	V	Driven by External Clock Generator	
V_{CL}	Clock Input Low Voltage	3.3V	V_{SS}-0.3	0.2 V_{CC}	V_{SS}-0.3	0.2 V_{CC}	1.6	V	Driven by External Clock Generator	
		5.0V	V_{SS}-0.3	0.2 V_{CC}	V_{SS}-0.3	0.2 V_{CC}	2.7	V	Driven by External Clock Generator	
V_{IH}	Input High Voltage	3.3V	0.7 V_{CC}	V_{CC}+0.3	0.7 V_{CC}	V_{CC}+0.3	1.8	V		
		5.0V	0.7 V_{CC}	V_{CC}+0.3	0.7 V_{CC}	V_{CC}+0.3	2.8	V		
V_{IL}	Input Low Voltage	3.3V	V_{SS}-0.3	0.2 V_{CC}	V_{SS}-0.3	0.2 V_{CC}	1.0	V		
		5.0V	V_{SS}-0.3	0.2 V_{CC}	V_{SS} 0.3	0.2 V_{CC}	1.5	V		
V_{OH}	Output High Voltage	3.3V	V_{CC}-0.4		V_{CC}-0.4		3.1	V	I_{OH} = -2.0 mA	
		5.0V	V_{CC}-0.4		V_{CC}-0.4		4.8	V	I_{OH} = -2.0 mA	
$V_{OL}1$	Output Low Voltage	3.3V		0.8		0.8	0.2	V	I_{OL} = +4.0 mA	
		5.0V		0.4		0.4	0.1	V	I_{OL} = +4.0 mA	
$V_{OL}2$	Output Low Voltage	3.3V		1.0		1.0	0.4	V	I_{OL} = 6 mA, 3 Pin Max	
		5.0V		1.0		1.0	0.5	V	I_{OL} = +12 mA, 3 Pin Max	
V_{OFFSET}	Comparator Input Offset Voltage	3.3V		25		25	10	mV		
		5.0V		25		25	10	mV		
I_{IL}	Input Leakage (Input bias current of comparator)	3.3V	-1.0	1.0	-1.0	1.0		µA	V_{IN} = 0V, V_{CC}	
		5.0V	-1.0	1.0	-1.0	1.0		µA	V_{IN} = 0V, V_{CC}	
I_{OL}	Output Leakage	3.3V	-1.0	1.0	-1.0	1.0		µA	V_{IN} = 0V, V_{CC}	
		5.0V	-1.0	1.0	-1.0	1.0		µA	V_{IN} = 0V, V_{CC}	
I_{CC}	Supply Current	3.3V		6		6	3.0	mA	@ 8 MHz	[4,5]
		5.0V		11.0		11.0	6.0	mA	@ 8 MHz	[4,5]
		3.3V		8.0		8.0	4.5	mA	@ 12 MHz	[4,5]
		5.0V		15		15	9.0	mA	@ 12 MHz	[4,5]

DC ELECTRICAL CHARACTERISTICS (Continued)
Z86C09/C19

Symbol	Parameter	V_{CC} Note [3]	T_A = 0°C to 70°C Min	Max	T_A = -40°C to 105°C Min	Max	Typical @ 25°C	Units	Conditions	Notes
$I_{CC}1$	Standby Current	3.3V		3.0		3.0	1.3	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 8 MHz	[4, 5]
		5.0V		5		5	3.0	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 8 MHz	[4, 5]
		3.3V		4.5		4.5	2.0	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 12 MHz	[4, 5]
		5.0V		7.0		7.0	4.0	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 12 MHz	[4, 5]
		3.3V		1.4		1.4	0.7	mA	Clock Divide by 16 @ 8 MHz	[4, 5]
		5.0V		3.5		3.5	2.0	mA	Clock Divide by 16 @ 8 MHz	[4, 5]
		3.3V		2.0		2.0	1.0	mA	Clock Divide by 16 @ 12 MHz	[4, 5]
		5.0V		4.5		4.5	2.5	mA	Clock Divide by 16 @ 12 MHz	[4, 5]
$I_{CC}2$	Standby Current	3.3V		10		20	1.0	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is not Running	[6]
		5.0V		10		20	3.0	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is not Running	[6]
		3.3V					TBD	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is Running	[6]
		5.0V		TBD		TBD	200	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is Running	[6]
I_{ALL}	Auto Latch Low Current	3.3V		7.0		14.0	4.0	µA	$0V < V_{IN} < V_{CC}$	
		5.0V		20.0		30.0	10	µA	$0V < V_{IN} < V_{CC}$	
I_{ALH}	Auto Latch High Current	3.3V		-4.0		-8.0	-2.0	µA	$0V < V_{IN} < V_{CC}$	
		5.0V		-9.0		-16.0	-5.0	µA	$0V < V_{IN} < V_{CC}$	
T_{POR}	Power On Reset	3.3V	7	24	6	25	13	mS		
		5.0V	3	13	2	14	7	mS		
V_{BO}	V_{CC} Brown Out Voltage		1.50	2.65	1.2	2.95	2.1	V	2 MHz max Ext. CLK Freq.	[3]

Notes:

[1] $I_{CC}1$

	Type	Max	Unit	Freq
Clock Driven on Crystal	3.0	5.0	mA	8 MHz
or XTAL Resonator	0.3	50	mA	8 MHz

[2] V_{SS} = 0V = GND
[3] 5.0V ± 0.5V, 3.0V ± 0.3V. The V_{BO} increases as the temperature decreases.
[4] All outputs unloaded, I/O pins floating, inputs at rail.
[5] $C_{L1} = C_{L2}$ = 100pf
[6] Same as note [4] except inputs at V_{CC}.

AC ELECTRICAL CHARACTERISTICS

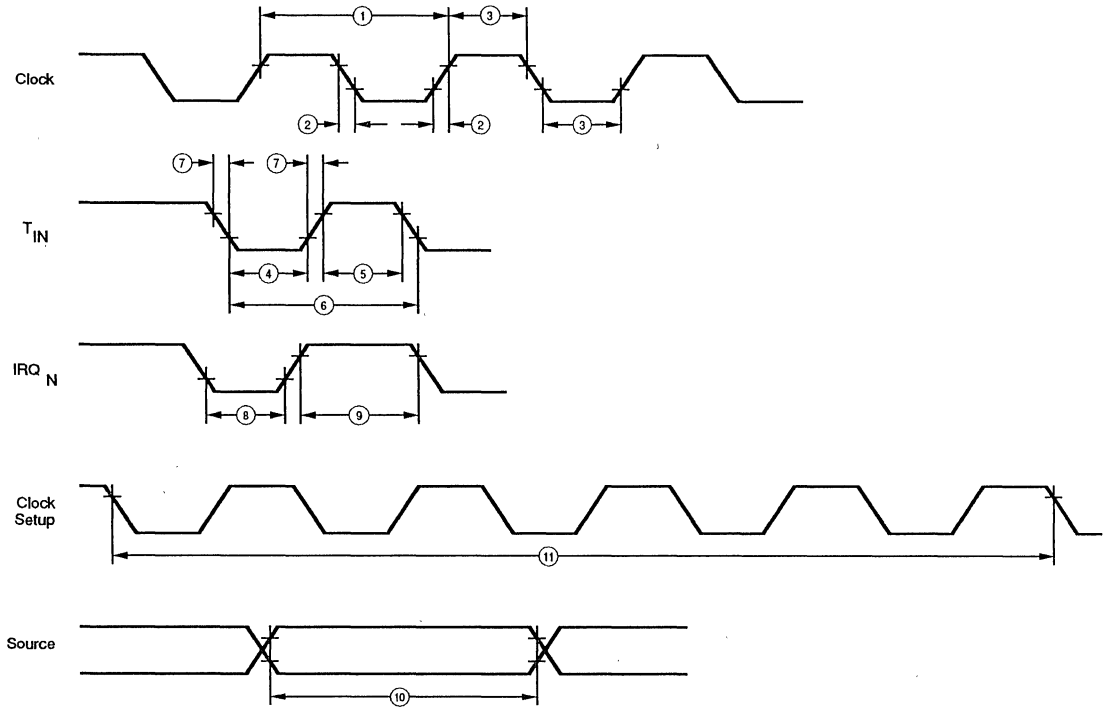


Figure 18. Additional Timing

AC ELECTRICAL CHARACTERISTICS
Z86C09/C19

No	Symbol	Parameter	V_{CC} Note[3]	T_A = 0°C TO 70°C 8 MHz Min	8 MHz Max	12 MHz Min	12 MHz Max	T_A = -40°C TO 105°C 8 MHz Min	8 MHz Max	12 MHz Min	12 MHz Max	Units	Notes
1	TpC	Input Clock Period	3.3V	125	100,000	83	100,000	125	100,000	83	100,000	ns	[1]
			5.0V	125	100,000	83	100,000	125	100,000	83	100,000	ns	[1]
2	TrC,TfC	Clock Input Rise	3.3V		25		15		25		15	ns	[1]
		and Fall Times	5.0V		25		15		25		15	ns	[1]
3	TwC	Input Clock Width	3.3V	37		26		37		26		ns	[1]
			5.0V	37		26		37		26		ns	[1]
4	TwTinL	Timer Input Low Width	3.3V	100		100		100		100		ns	[1]
			5.0V	70		70		70		70		ns	[1]
5	TwTinH	Timer Input High Width	3.3V	3TpC		3TpC		3TpC		3TpC			[1]
			5.0V	3TpC		3TpC		3TpC		3TpC			[1]

AC ELECTRICAL CHARACTERISTICS (Continued)
Z86C09/C19

No	Symbol	Parameter	V_{CC} Note[3]	T_A = 0°C TO 70°C 8 MHz Min	8 MHz Max	12 MHz Min	12 MHz Max	T_A = -40°C TO 105°C 8 MHz Min	8 MHz Max	12 MHz Min	12 MHz Max	Units	Notes
6	TpTin	Timer Input Period	3.3V	8TpC		8TpC		8TpC		8TpC			[1]
			5.0V	8TpC		8TpC		8TpC		8TpC			[1]
7	TrTin, TtTin	Timer Input Rise and Fall Timer	3.3V		100		100		100		100	ns	[1]
			5.0V		100		100		100		100	ns	[1]
8	TwIL	Int. Request Input Low Time	3.3V	100		100		100		100		ns	[1,2]
			5.0V	70		70		70		70		ns	[1,2]
9	TwIH	Int. Request Input High Time	3.3V	3TpC		3TpC		3TpC		3TpC			[1,2]
			5.0V	3TpC		3TpC		3TpC		3TpC			[1,2]
10	Twsm	STOP Mode Recovery Width Spec	3.3V	12		12		12		12		ns	
			5.0V	12		12		12		12		ns	
11	Tost	Oscillator Startup Time	3.3V		5TpC		5TpC		5TpC		5TpC		Reg. [4]
			5.0V		5TpC		5TpC		5TpC		5TpC	ns	
	Twdt	Watchdog Timer Refresh Time	3.3V	15		15		12		12			[5]
			5.0V	5		5		3		3		ms	D0 = 0 D1 = 0
			3.3V	30		30		25		25		ms	D0 = 0
			5.0V	16		16		12		12		ms	D1 = 1
			3.3V	60		60		50		50		ms	D0 = 0
			5.0V	25		25		30		30		ms	D1 = 1
			3.3V	250		250		200		200		ms	D0 = 1
			5.0V	120		120		100		100		ms	D1 = 1

Notes:

[1] Timing Reference uses 0.9 V_{CC} for a logic "1" and 0.1 V_{CC} for a logic "0".
[2] Interrupt request via Port 3 (P31-P33)
[3] 5.0V ± 0.5V, 3.3V ± 0.3V
[4] SMR-D5 = 0
[5] Reg. WDTMR

EXPANDED REGISTER FILE CONTROL REGISTERS

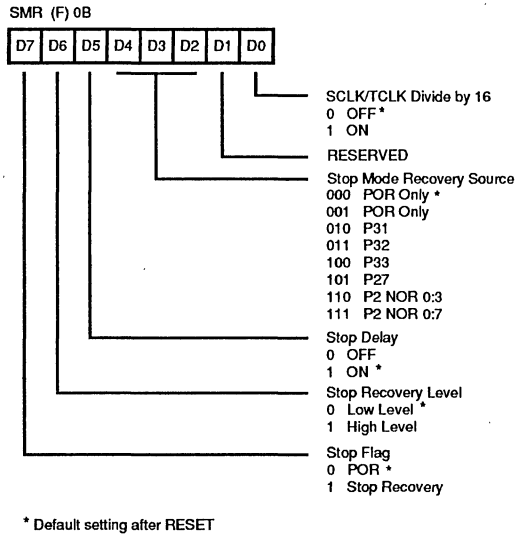


Figure 19. STOP Mode Recovery Register

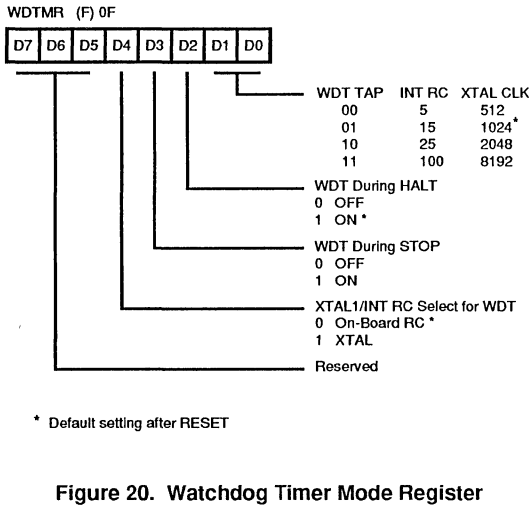


Figure 20. Watchdog Timer Mode Register

Z8 CONTROL REGISTER DIAGRAMS

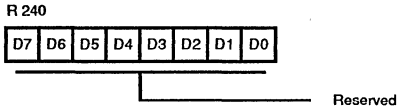


Figure 21. Reserved

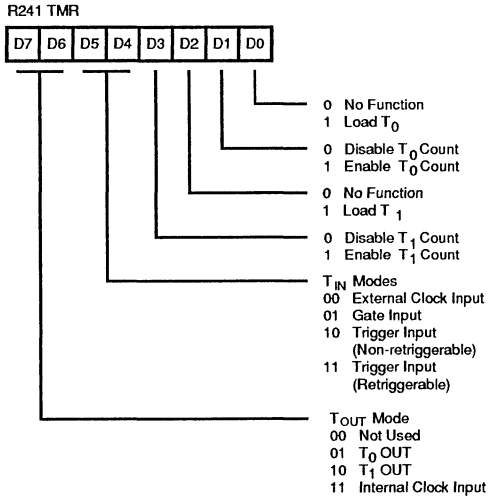


Figure 22. Timer Mode Register
(F1H: Read/Write)

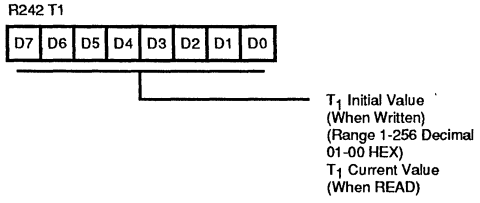


Figure 23. Counter Timer 1 Register
(F2H: Read/Write)

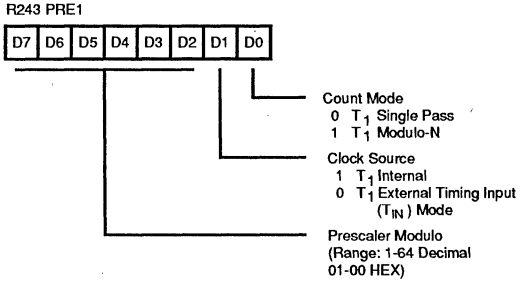


Figure 24. Prescaler 1 Register
(F3H: Write Only)

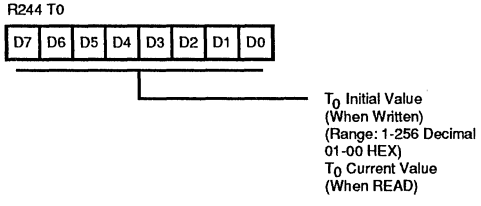


Figure 25. Counter/Timer 0 Register
(F4H: Read/Write)

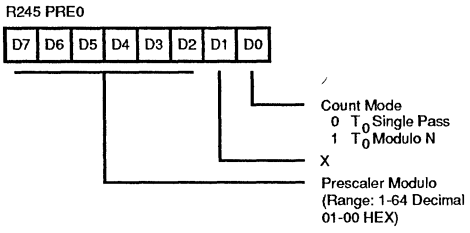


Figure 26. Prescaler 0 Register
(F5H: Write Only)

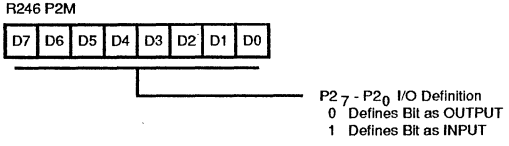


Figure 27. Port 2 Mode Register
(F6H: Write Only)

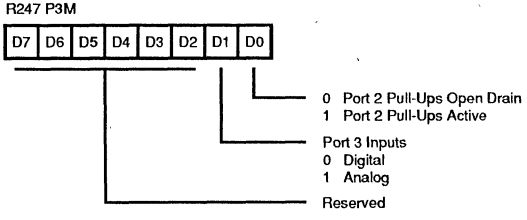


Figure 28. Port 3 Mode Register
(F7H: Write Only)

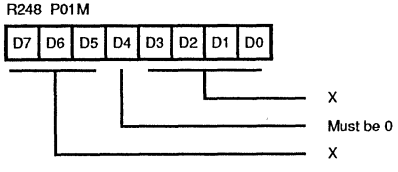


Figure 29. Port 0 and 1 Mode Register

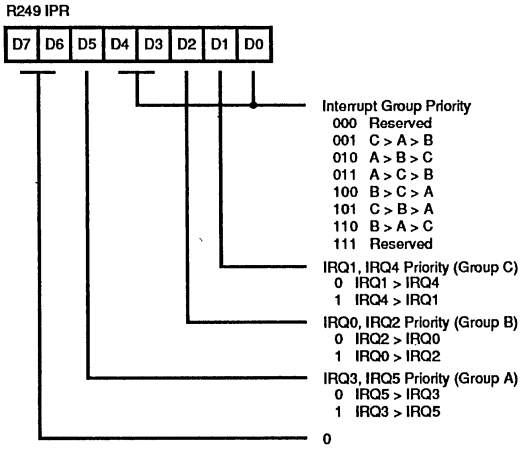


Figure 30. Interrupt Priority Register
(F9H: Write Only)

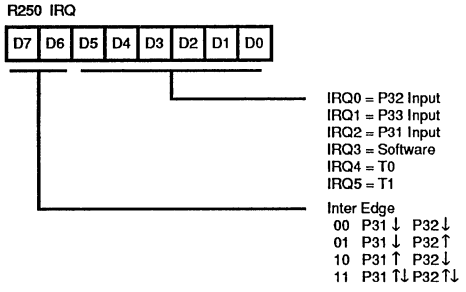


Figure 31. Interrupt Req Register
(FAH: Read/Write)

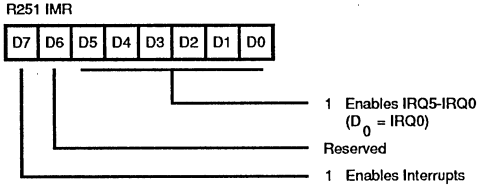


Figure 32. Interrupt Mask Register
(FBH: Read/Write)

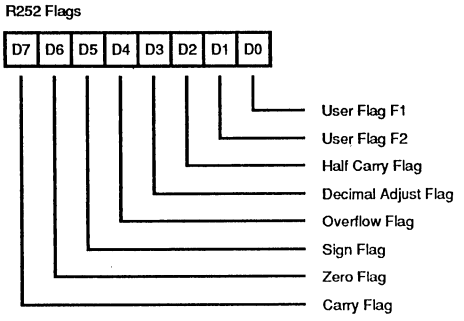


Figure 33. Flag Register
(FCH: Read/Write)

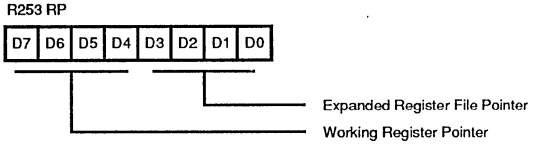


Figure 34. Register Pointer
(FDH: Read/Write)

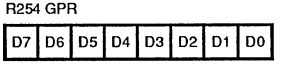


Figure 35. General Purpose Register
(FEH: Read/Write)

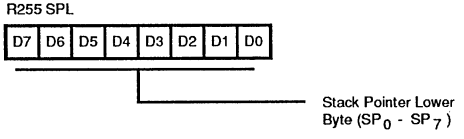


Figure 36. Stack Pointer
(FFH: Read/Write)

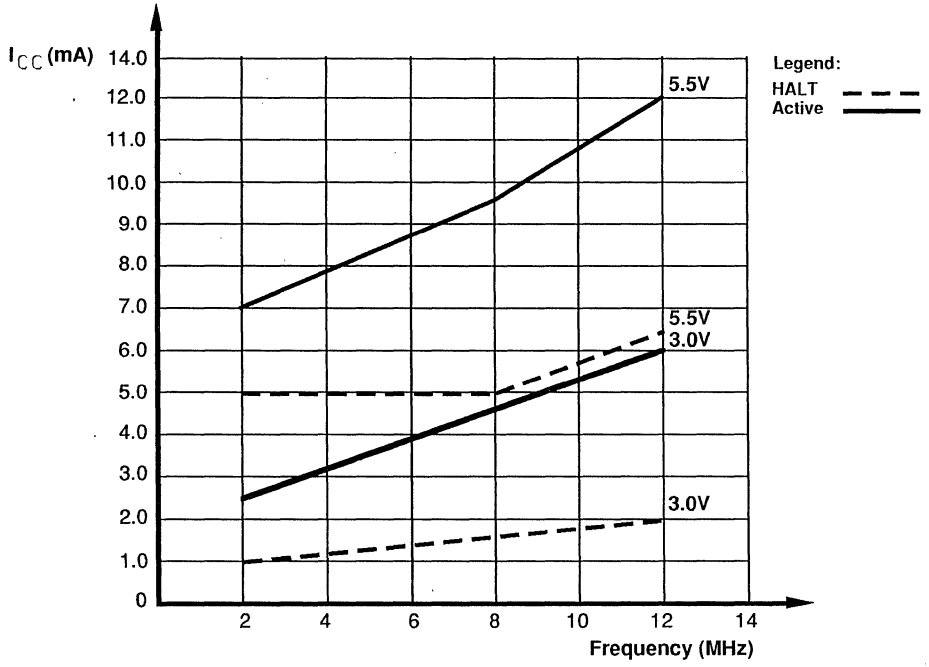


Figure 37. Maximum I_{cc} Vs Frequency

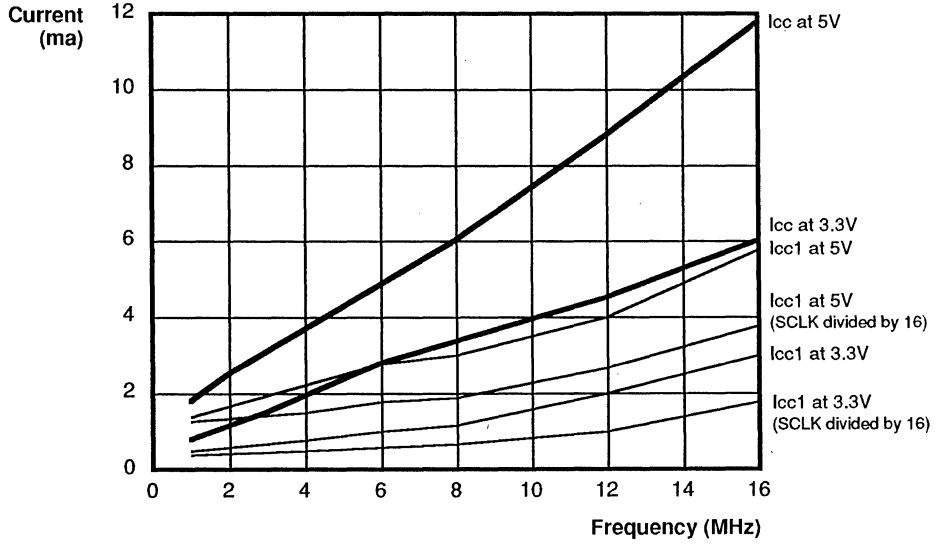


Figure 38. Typical I_{cc} Vs Frequency

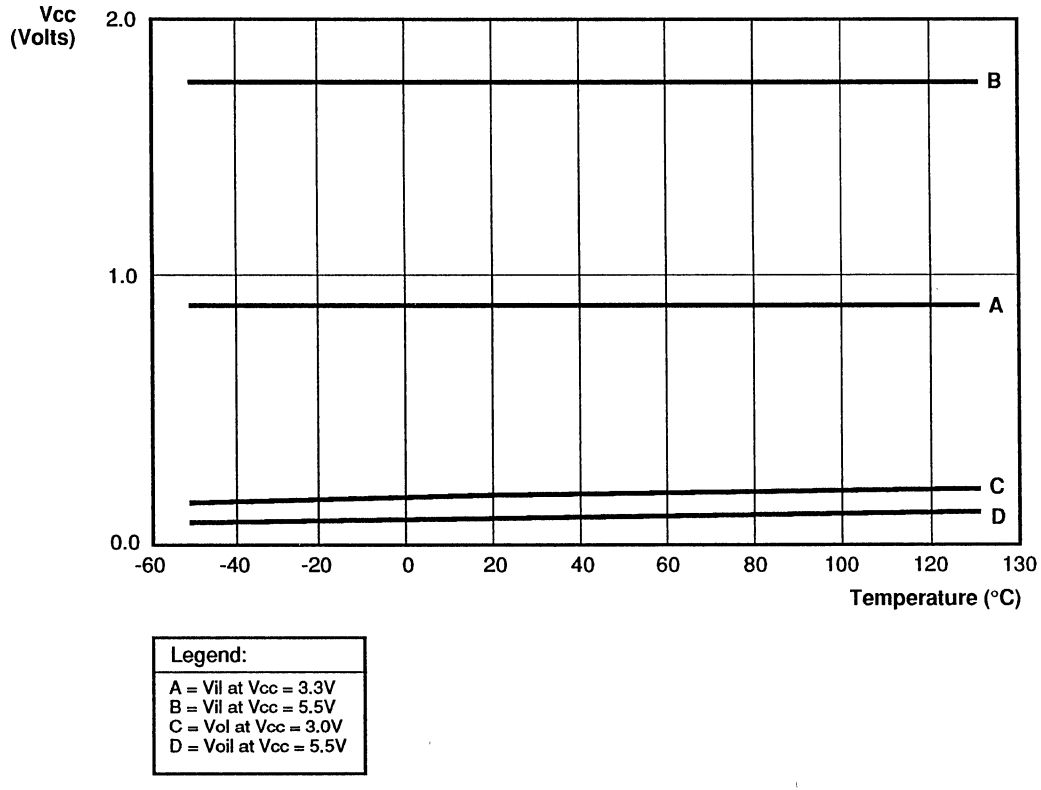


Figure 39. Typical V_{OL}, V_{IL} Vs Temperature

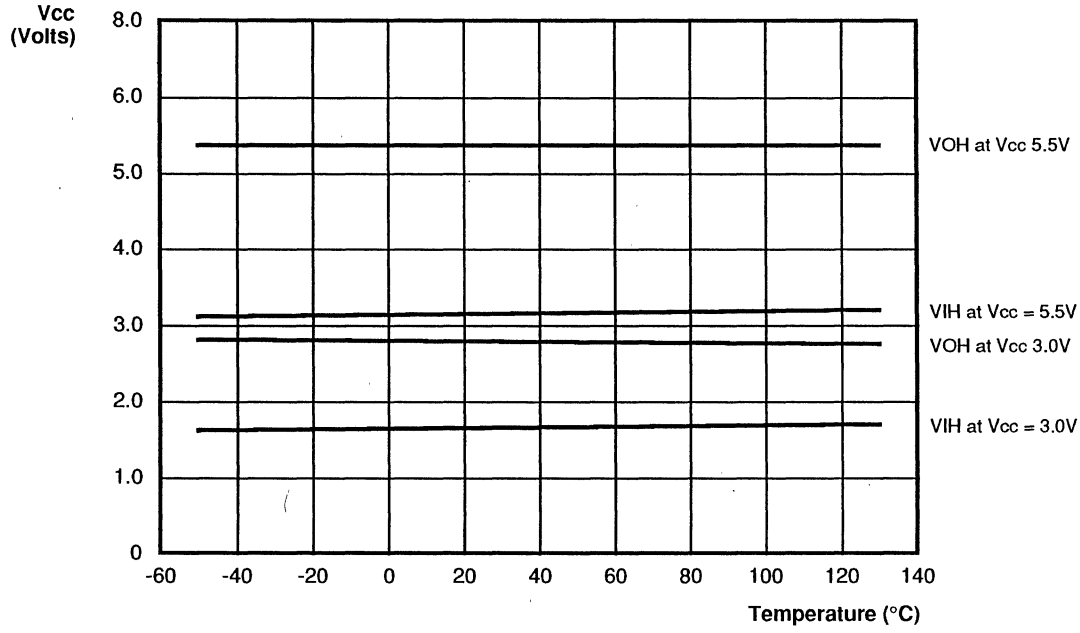


Figure 40. Typical V_{OH}, V_{IH} Vs Temperature

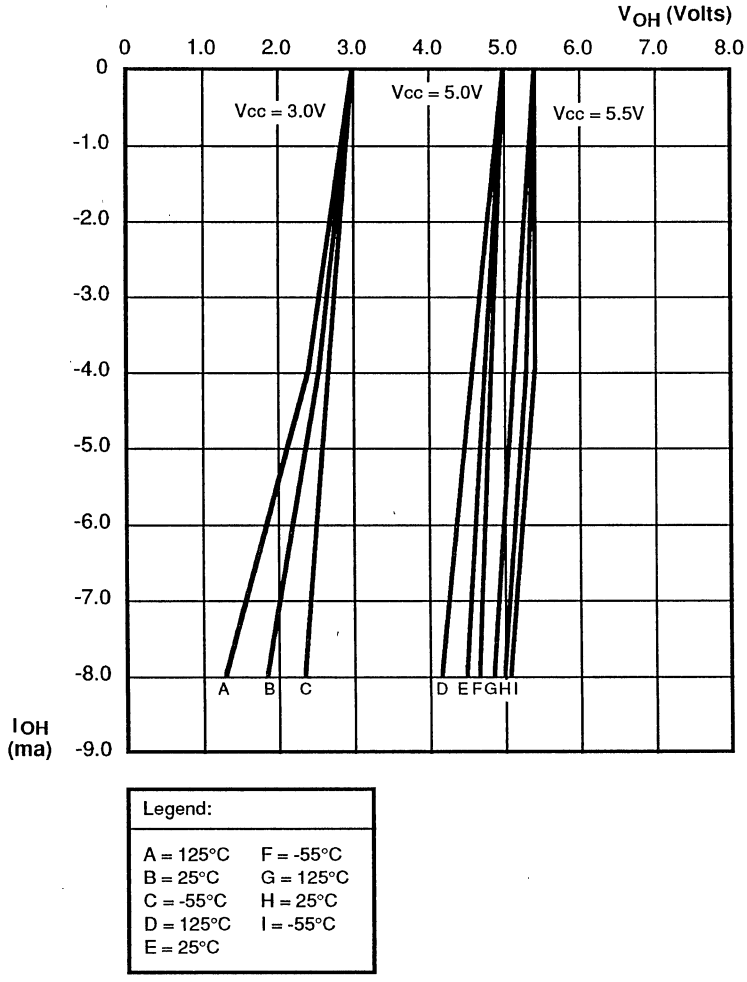


Figure 41. Typical V_{OH} Vs I_{OH} Over Temperature

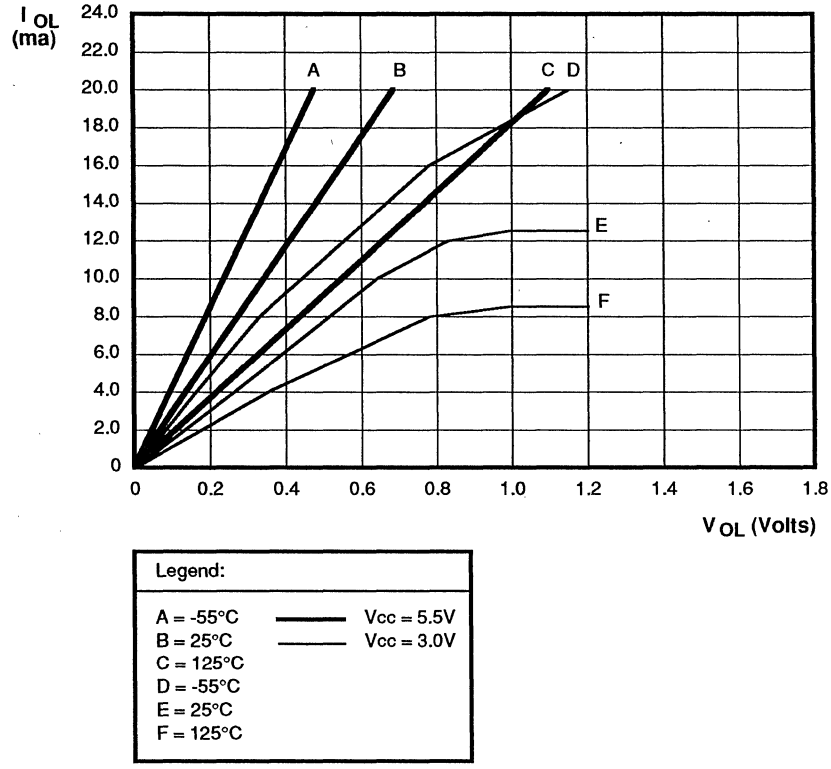
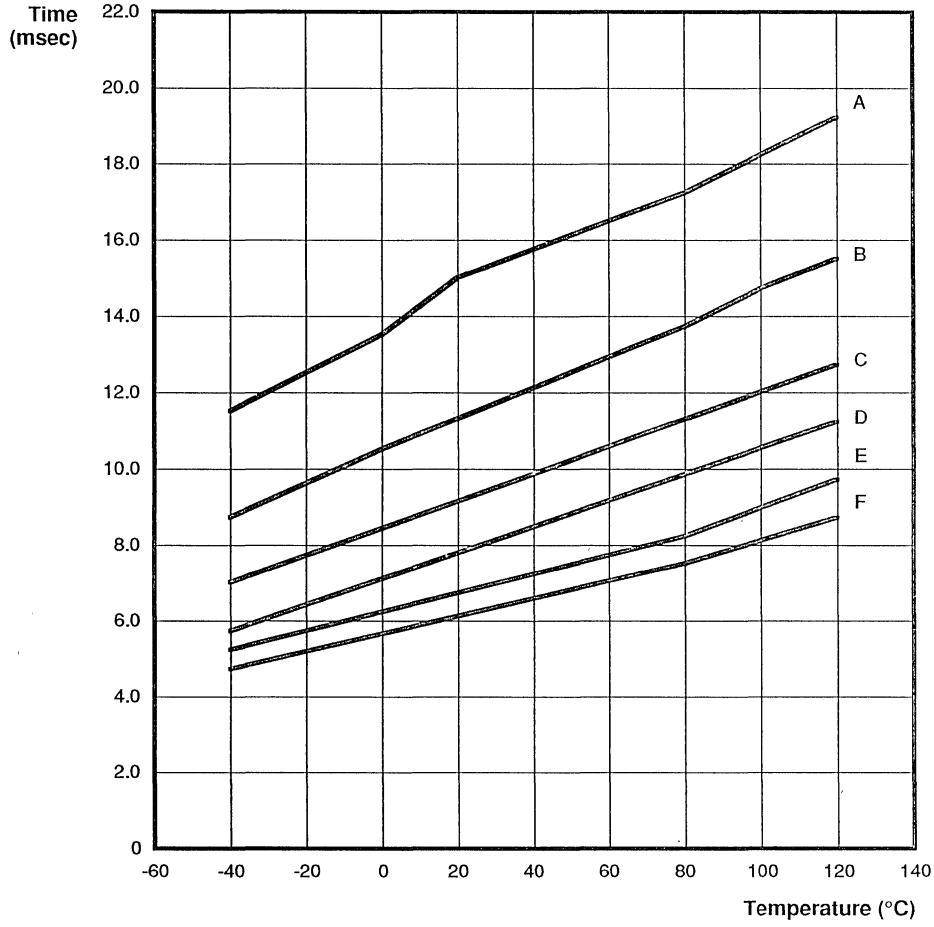


Figure 42. Typical I_{OL} Vs V_{OL} Over Temperature



Legend:	
A - Vcc = 3.0V	D - Vcc = 4.5v
B - Vcc = 3.5V	E - Vcc = 5.0V
C - Vcc = 4.0V	F - Vcc = 5.5V

Figure 43. Typical Power-On Reset Time Vs Temperature

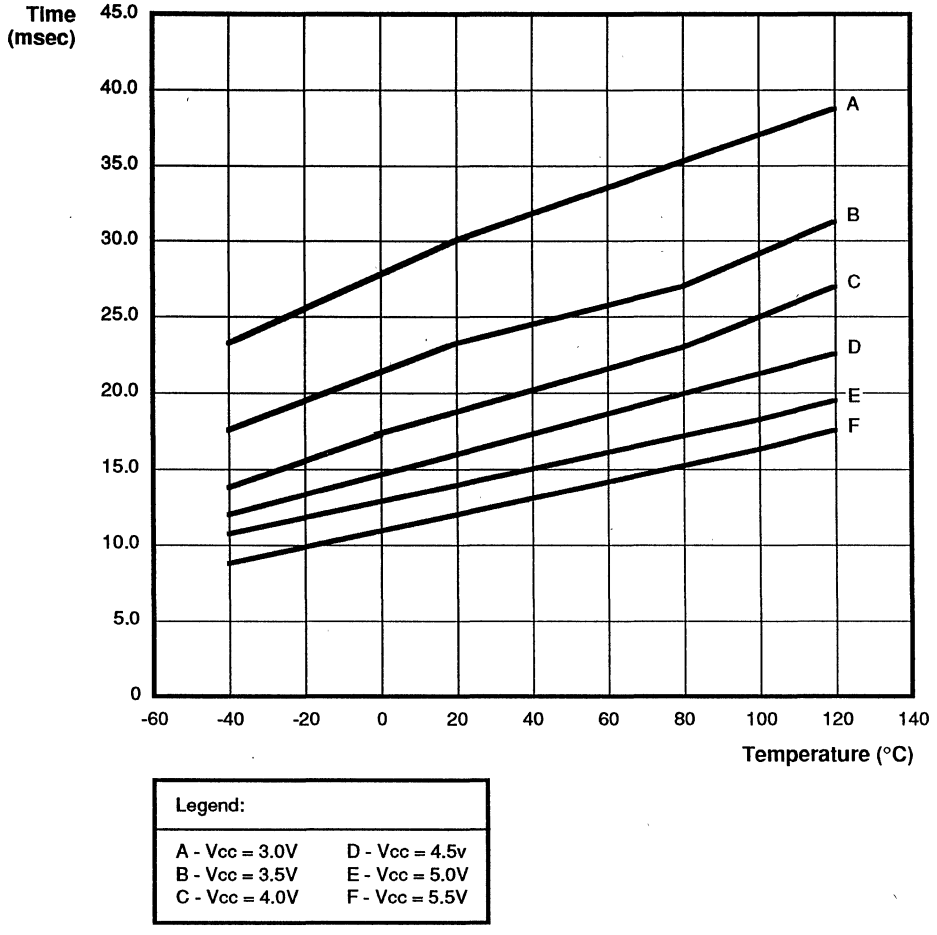


Figure 44. Typical 5 ms WDT Setting Vs Temperature

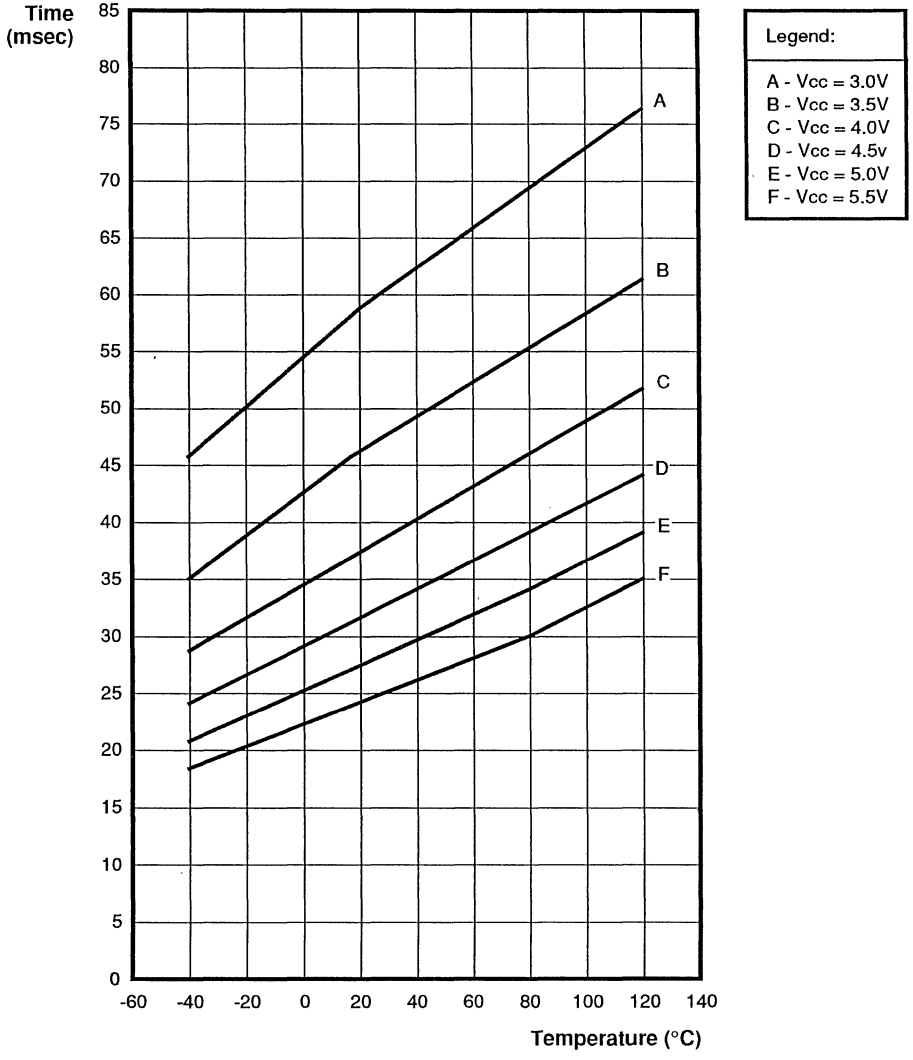


Figure 45. Typical 15 ms WDT Setting Vs Temperature

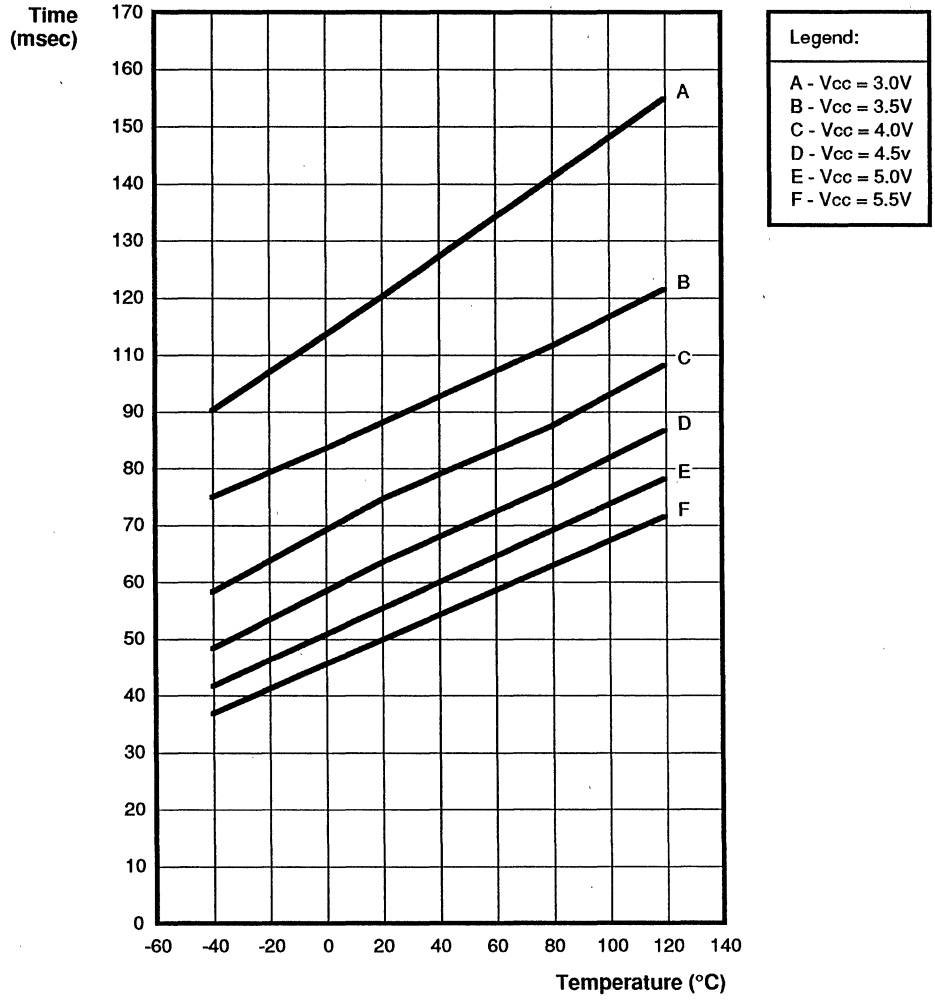


Figure 46. Typical 25 ms WDT Setting Vs Temperature

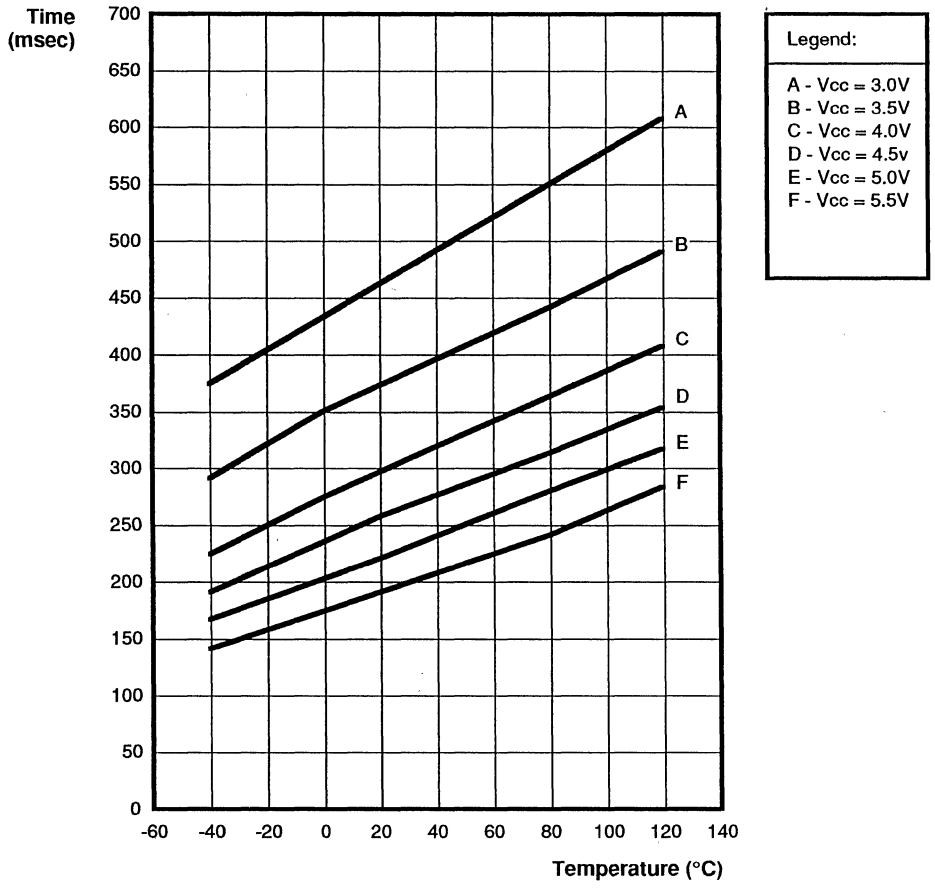
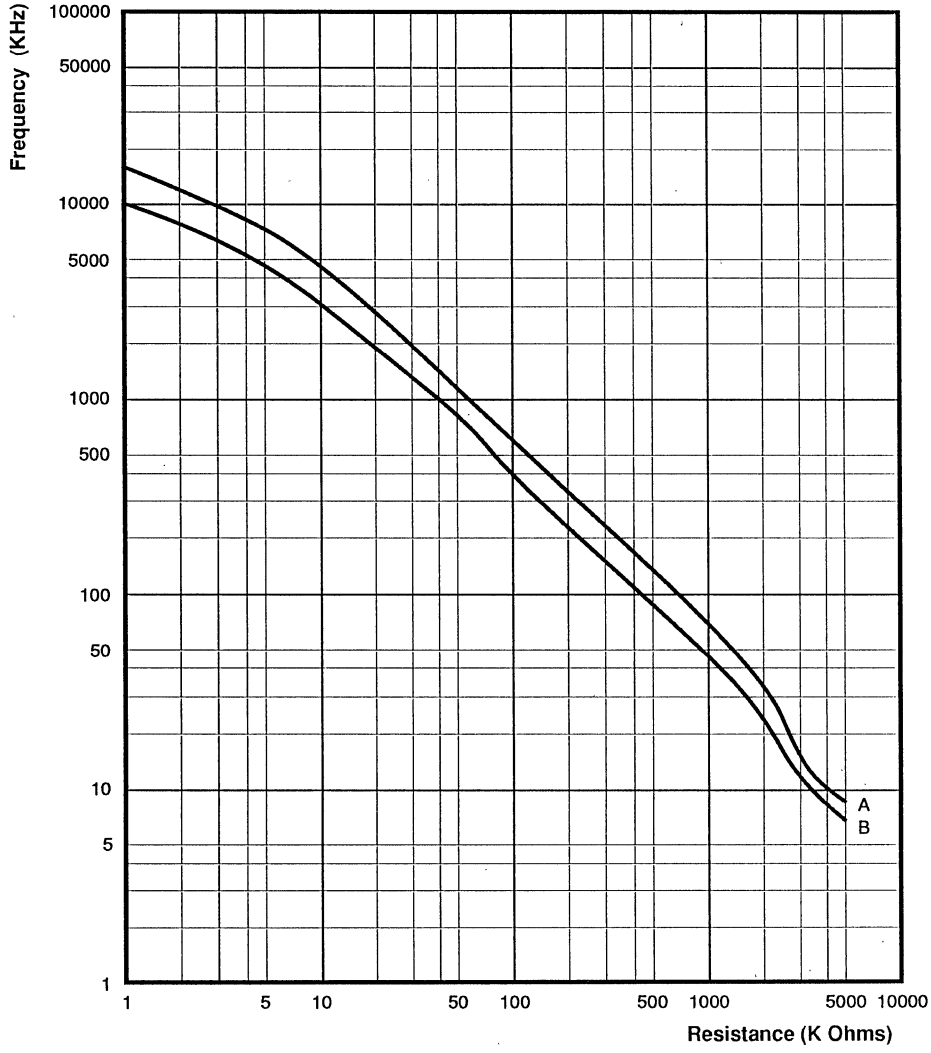


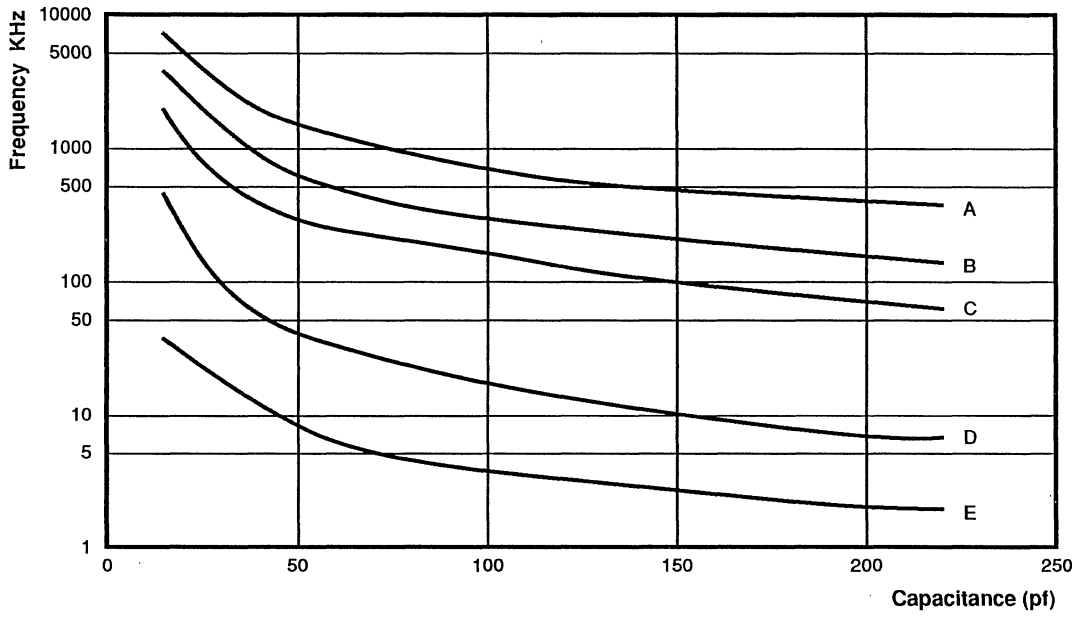
Figure 47. Typical 100 ms WDT Setting Vs Temperature



Legend:
A - Vcc = 5.0V C = 33pf B - Vcc = 3.3V C = 33pf

Note: This chart for reference only. Each process will have a different characteristc curve.

Figure 48. Typical Frequency Vs RC Resistance



Legend:
A - Vcc = 5.0V R = 22K Ohms B - Vcc = 5.0V R = 56K Ohms C - Vcc = 5.0V R = 100K Ohms D - Vcc = 5.0V R = 1M Ohms C - Vcc = 5.0V R = 4M Ohms

Figure 49. Typical RC Resistance/Capacitance Vs Frequency

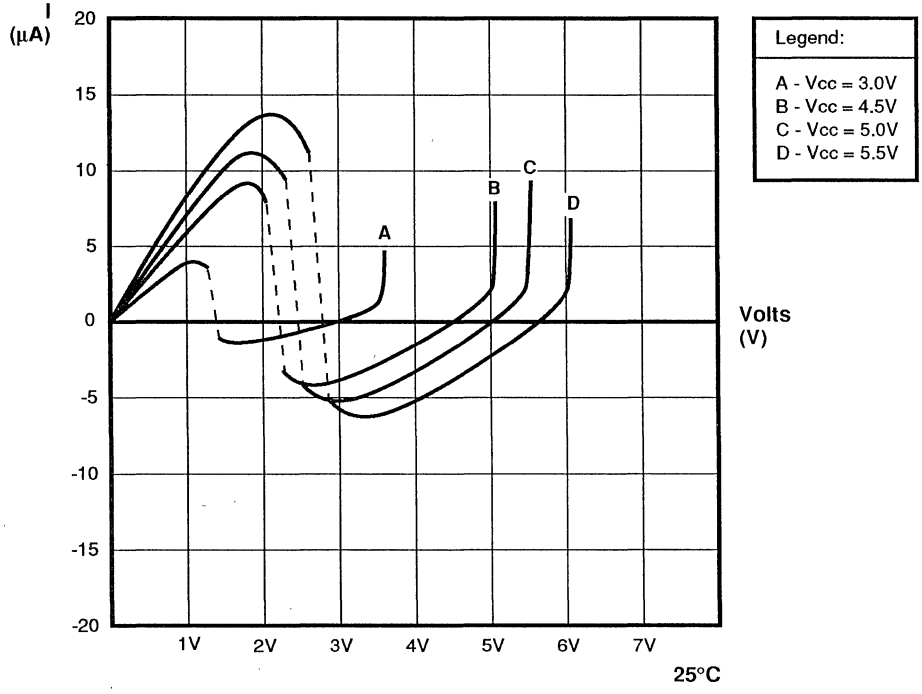


Figure 50. Auto Latch Characteristic

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

Symbol	Meaning
IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

Symbol	Meaning
dst	Destination location or contents
src	Source location or contents
cc	Condition code
@	Indirect address prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag register (Control Register 252)
RP	Register Pointer (R253)
IMR	Interrupt mask register (R251)

Flags. Control register (R252) contains the following six flags:

Symbol	Meaning
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

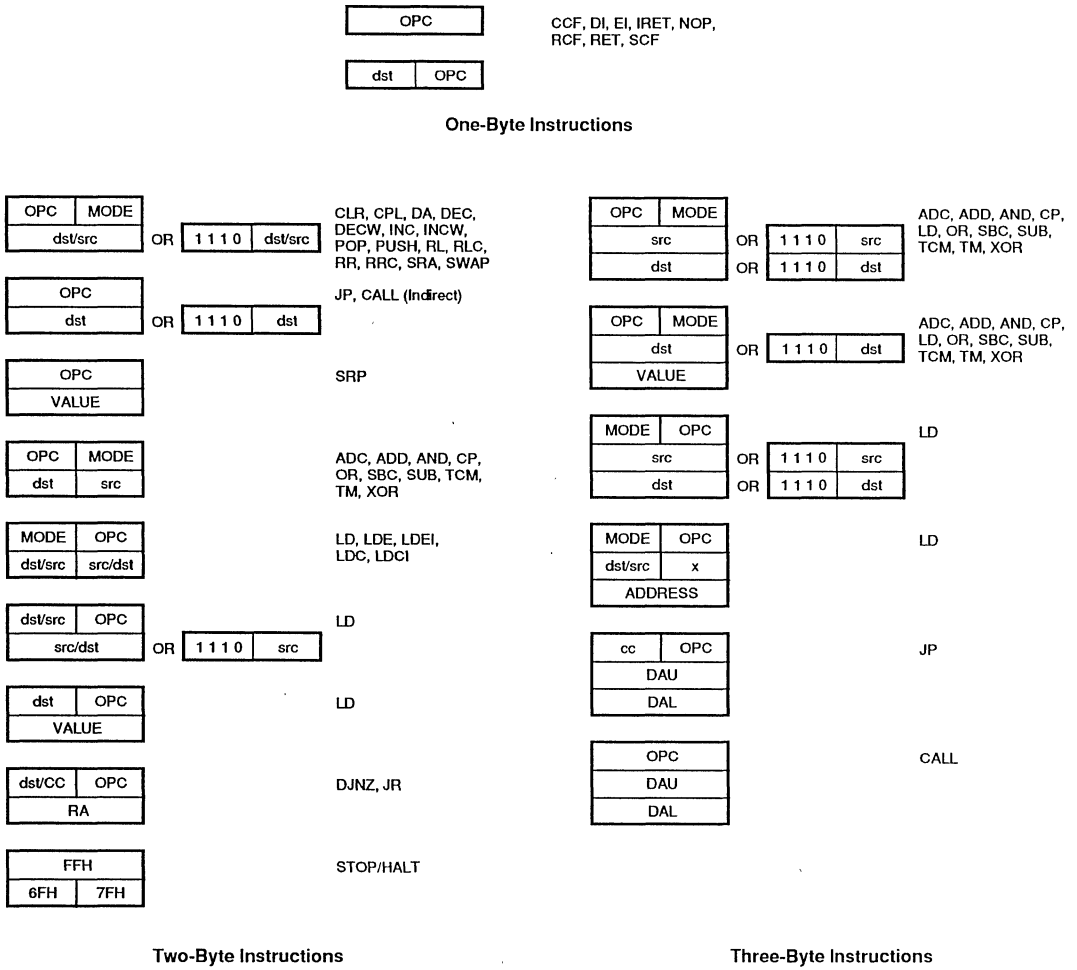
Affected flages are indicated by:

0	Clear to zero
1	Set to one
*	Set to clear according to operation
-	Unaffected
x	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000		Always True	
0111	C	Carry	C = 1
1111	NC	No Carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not Zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No Overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not Equal	Z = 0
1001	GE	Greater Than or Equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater Than	[Z OR (S XOR V)] = 0
0010	LE	Less Than or Equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned Greater Than or Equal	C = 0
0111	ULT	Unsigned Less Than	C = 1
1011	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned Less Than or Equal	(C OR Z) = 1
0000		Never True	

INSTRUCTION FORMATS



INSTRUCTION SUMMARY

Note: Assignment of a value is indicated by the symbol " ← ". For example:

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr (n)" is used to refer to bit (n) of a given operand location. For example:

dst (7)

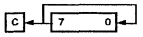
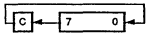
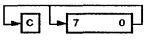
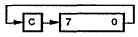
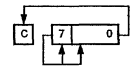
refers to bit 7 of the destination operand.

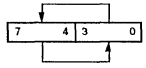
INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst, src dst←dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst←dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst←dst AND src	†		5[ ]	-	*	*	0	-	-
CALL dst SP←SP - 2 @SP←PC, PC←dst	DA IRR		D6 D4	-	-	-	-	-	-
CCF C←NOT C			EF	*	-	-	-	-	-
CLR dst dst←0	R IR		B0 B1	-	-	-	-	-	-
COM dst dst←NOT dst	R IR		60 61	-	*	*	0	-	-
CP dst, src dst - src	†		A[ ]	*	*	*	*	-	-
DA dst dst←DA dst	R IR		40 41	*	*	*	X	-	-
DEC dst dst←dst - 1	R IR		00 01	-	*	*	*	-	-
DECW dst dst←dst - 1	RR IR		80 81	-	*	*	*	-	-
DI IMR(7)←0			8F	-	-	-	-	-	-
DJNZr, dst r←r - 1 if r ≠ 0 PC←PC + dst Range: +127, -128	RA		rA r = 0 - F	-	-	-	-	-	-
EI IMR(7)←1			9F	-	-	-	-	-	-
HALT			7F	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
INC dst dst←dst + 1	r R IR		rE r = 0 - F 20 21	-	*	*	*	-	-
INCW dst dst←dst + 1	RR IR		A0 A1	-	*	*	*	-	-
IRET FLAGS←@SP; SP←SP + 1 PC←@SP; SP←SP + 2; IMR(7)←1			BF	*	*	*	*	*	*
JP cc, dst if cc is true PC←dst	DA IRR		cD c = 0 - F 30	-	-	-	-	-	-
JR cc, dst if cc is true, PC←PC + dst Range: +127, -128	RA		cB c = 0 - F	-	-	-	-	-	-
LD dst, src dst←src	r r R r X r r Ir R R R IR	Im R r X r Ir r R IR IM IM R	rC r8 r9 r = 0 - F C7 D7 E3 F3 E4 E5 E6 E7 F5	-	-	-	-	-	-
LDC dst, src	r	Irr	C2	-	-	-	-	-	-
LDCI dst, src dst←src r←r +1; rr←rr + 1	Ir	Irr	C3	-	-	-	-	-	-

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
NOP			FF	-	-	-	-	-	-
OR dst, src dst←dst OR src	†		4[ ]	-	*	*	0	-	-
POP dst dst←@SP; SP←SP + 1	R IR		50 51	-	-	-	-	-	-
PUSH src SP←SP - 1; @SP←src		R IR	70 71	-	-	-	-	-	-
RCF C←0			CF	0	-	-	-	-	-
RET PC←@SP; SP←SP + 2			AF	-	-	-	-	-	-
RL dst 	R IR		90 91	*	*	*	*	-	-
RLC dst 	R IR		10 11	*	*	*	*	-	-
RR dst 	R IR		E0 E1	*	*	*	*	-	-
RRC dst 	R IR		C0 C1	*	*	*	*	-	-
SBC dst, src dst←dst←src←C	†		3[ ]	*	*	*	*	1	*
SCF C←1			DF	1	-	-	-	-	-
SRA dst 	R IR		D0 D1	*	*	*	0	-	-
SRP src RP←src		Im	31	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
STOP			6F	-	-	-	-	-	-
SUB dst, src dst←dst←src	†		2[ ]	*	*	*	*	1	*
SWAP dst 	R IR		F0 F1	X	*	*	X	-	-
TCM dst, src (NOT dst) AND src	†		6[ ]	-	*	*	0	-	-
TM dst, src dst AND src	†		7[ ]	-	*	*	0	-	-
XOR dst, src dst←dst XOR src	†		B[ ]	-	*	*	0	-	-

† These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

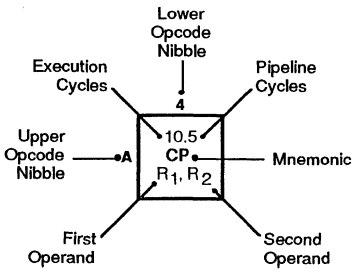
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address Mode dst	Address Mode src	Lower Opcode Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

OPCODE MAP

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								6.0 **WDT**
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								6.0 **STOP**
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								7.0 **HALT**
8	10.5 **DECW** RR1	10.5 **DECW** IR1														6.1 **DI**
9	6.5 **RL** R1	6.5 **RL** IR1														6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14.0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2				10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1			20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6.5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2			3					2				3		1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_2 = Dst address
R_1 or r_2 = Src address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank are not defined.

* 2-byte instruction appears as a 3-byte instruction

GENERAL DESCRIPTION

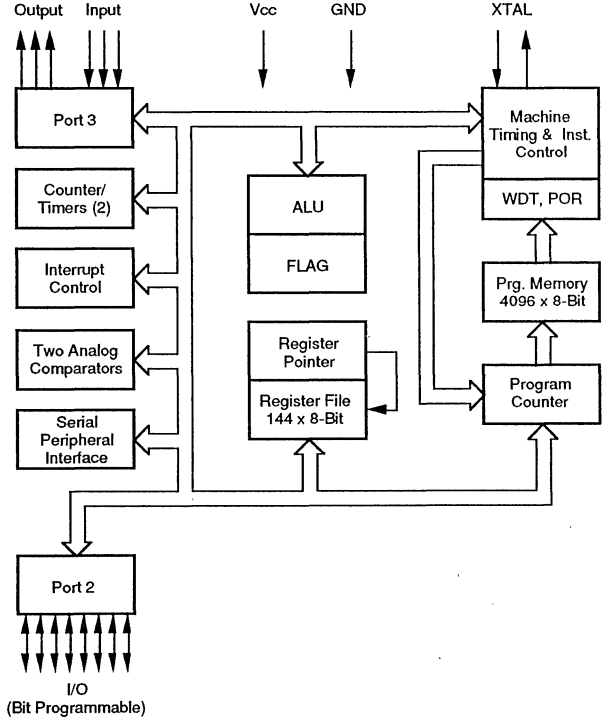
The Z86E19 is a member of the Z8 single-chip one-time-programmable microcontroller family. The device is housed in an18-pin DIP. Zilog's CMOS microcontroller offers fast execution, efficient use of memory, sophisticated interrupts, bit manipulation capabilities, and easy hardware/software system expansion along with low cost and low power consumption.

The Z86E19 architecture is based on Zilog's 8-bit microcontroller core with an Expanded Register File to allow access to register mapped peripheral and I/O circuits. The CCP offers a flexible I/O scheme and a number of ancillary features that are useful in many consumer, industrial and automotive applications.

With powerful peripheral features such as on-board comparators, counter/timers, watch dog timer, and serial peripheral interface, the Z86E19 meets the needs of most sophisticated controller applications.

FEATURES

- *8-bit CMOS microcontroller, 18-pin DIP*
- *Emulates the Z86C06/09/19*
- *Serial peripheral interface with compare feature*
- *Clock speeds 8, and 12 MHz*
- *"Brown-Out" protection*
- *Watchdog/Power-On Reset Timer*
- *Two Comparators with programmable interrupt polarity*
- *Six vectored, priority interrupts from six different sources*
- *On-chip oscillator that accepts a crystal, ceramic resonator, LC, RC, or external clock drive.*
- *14 input/output lines*
- *Low EMI Noise Mode*
- *Schmitt-triggered CMOS inputs*
- *4K bytes of OTP ROM*
- *124 bytes of RAM*
- *Standby modes: STOP and HALT*



PRODUCT SPECIFICATION

Z86C11
CMOS Z8® MICROCONTROLLER

FEATURES

- 8-bit CMOS microcontroller, 40- or 44-pin package
- 4.5 to 5.5 Voltage operating range
- Low power Consumption - 220 mW (max) @ 16 MHz
- Fast instruction pointer - 1.0 microseconds @ 12 MHz
- Two standby modes - STOP and HALT
- 32 input/output lines
- Full-Duplex UART
- All digital inputs are TTL levels
- Auto Latches
- RAM and ROM protect
- Low EMI option
- 4 Kbytes of ROM
- 236 bytes of RAM
- Two programmable 8-bit Counter/Timers each with 6-bit programmable prescaler.
- Six vectored, priority interrupts from eight different sources
- Clock speeds 12 and 16 MHz
- On-Chip oscillator that accepts a crystal, ceramic resonator, LC or external clock drive.

GENERAL DESCRIPTION

The Z86C11 microcontroller (MCU) introduces a new level of sophistication to single-chip architecture. The Z86C11 is a member of the Z8 single-chip microcontroller family with 4 Kbytes of ROM and 236 bytes of RAM.

The MCU is housed in a 40-pin DIP, 44-pin Leaded Chip-Carrier, or a 44-pin Quad Flat Pack, and is manufactured in CMOS technology. The ROMless pin option is available on the 44-pin versions only. Having the ROM/ROMless selectivity, the MCU offers both external memory and preprogrammed ROM. This enables the Z8 microcontroller to be used in high volume applications or where code flexibility is required.

Zilog's CMOS microcontroller offers fast execution, efficient use of memory, sophisticated interrupts, input/output bit manipulation capabilities, and easy hardware/software system expansion along with low cost and low power consumption.

The Z86C11 architecture is characterized by Zilog's 8-bit microcontroller core. The device offers a flexible I/O scheme, an efficient register and address space structure, multiplexed capabilities between address/data, I/O, and a number of ancillary features that are useful in many industrial and advanced scientific applications.

The device applications demand powerful I/O capabilities. The Z86C11 fulfills this with 32 pins dedicated to input and output. These lines are grouped into four ports. Each port consists of eight-lines, and is configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an address/data bus for interfacing external memory.

There are three basic address spaces available to support this wide range of configuration: Program Memory, Data Memory and 236 General-Purpose Registers.

GENERAL DESCRIPTION (Continued)

To unburden the program from coping with the real-time problems such as counting/timing and serial data communication, the Z86C11 offers two on-chip counter/timers with a large number of user selectable modes, and an asynchronous receiver/transmitter (UART - Figure 1).

Note: All Signals with a preceding front slash, "/", are active Low, e.g.: B//W (WORD is active Low); /B/W (BYTE is active Low, only).

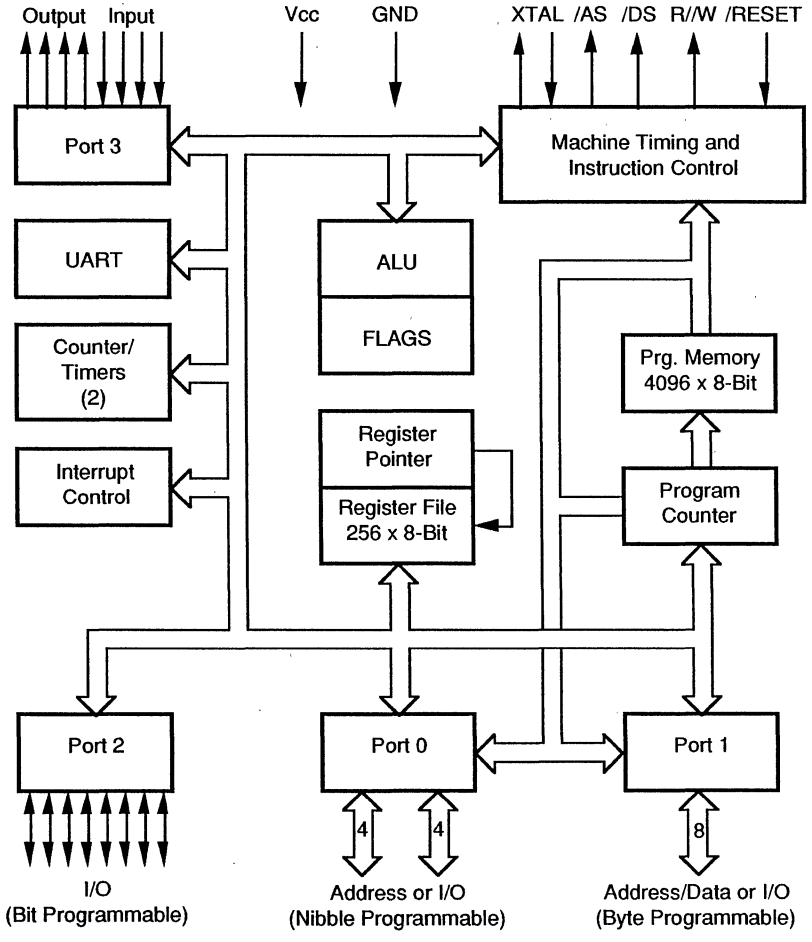


Figure 1. Functional Block Diagram

PIN DESCRIPTION

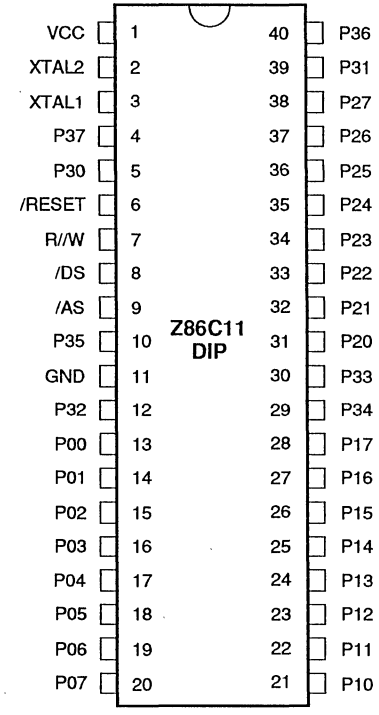


Figure 2. 40-Pin Plastic Dual In-Line Pin Assignments

Table 1. 40-Pin Plastic Dual In-Line Pin Identification

Pin #	Symbol	Function	Direction
1	V_{cc}	Power Supply	Input
2	XTAL2	Crystal, Oscillator Clock	Output
3	XTAL1	Crystal, Oscillator Clock	Input
4	P37	Port 3 pin 7	Output
5	P30	Port 3 pin 0	Input
6	/RESET	Reset	Input
7	R//W	Read/Write	Output
8	/DS	Data Strobe	Output
9	/AS	Address Strobe	Output
10	P35	Port 3 pin 5	Output
11	GND	Ground	Input
12	P32	Port 3 pin 2	Input
13-20	P00-P07	Port 0 pin 0,1,2,3,4,5,6,7	In/Output
21-28	P10-P17	Port 1 pin 0,1,2,3,4,5,6,7	In/Output
29	P34	Port 3 pin 4	Output
30	P33	Port 3 pin 3	Input
31-38	P20-P27	Port 2 pin 0,1,2,3,4,5,6,7	In/Output
39	P31	Port 3 pin 1	Input
40	P36	Port 3 pin 6	Output

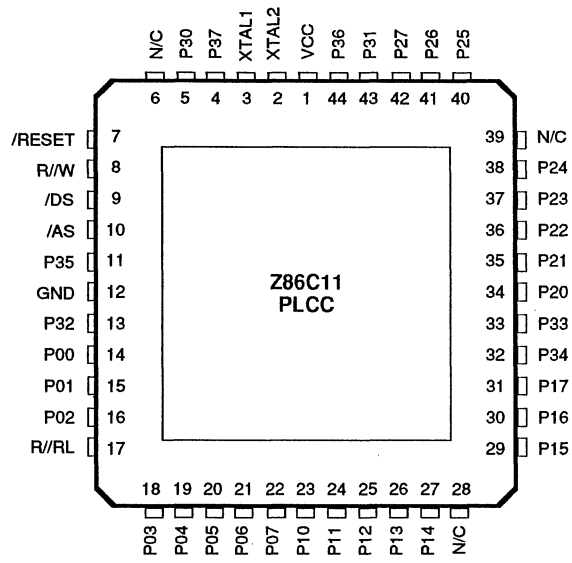


Figure 3. 44-Pin Plastic Leaded Chip Carrier Pin Assignments

Table 2. 44-Pin Plastic Leaded Chip Carrier Pin Identification

Pin #	Symbol	Function	Direction
1	V_{cc}	Power Supply	Input
2	XTAL2	Crystal, Oscillator Clock	Output
3	XTAL1	Crystal, Oscillator Clock	Input
4	P37	Port 3 pin 7	Output
5	P30	Port 3 pin 0	Input
6	N/C	Not Connected	Input
7	/RESET	Reset	Input
8	R//W	Read/Write	Output
9	/DS	Data Strobe	Output
10	/AS	Address Strobe	Output
11	P35	Port 3 pin 5	Output
12	GND	Ground, GND	Input
13	P32	Port 3 pin 2	Input
14-16	P00-P02	Port 0 pin 0,1,2	In/Output
17	R//RL	ROM/ROMless control	Input
18-22	P03-P07	Port 0 pin 3,4,5,6,7	In/Output
23-27	P10-P14	Port 1 pin 0,1,2,3,4	In/Output
28	N/C	Not Connected	Input
29-31	P15-P17	Port 1 pin 5,6,7	In/Output
32	P34	Port 3 pin 4	Output
33	P33	Port 3 pin 3	Input
34-38	P20-P24	Port 2 pin 0,1,2,3,4	In/Output
39	N/C	Not Connected	Input
40-42	P25-P27	Port 2 pin 5,6,7	In/Output
43	P31	Port 3 pin 1	Input
44	P36	Port 3 pin 6	Output

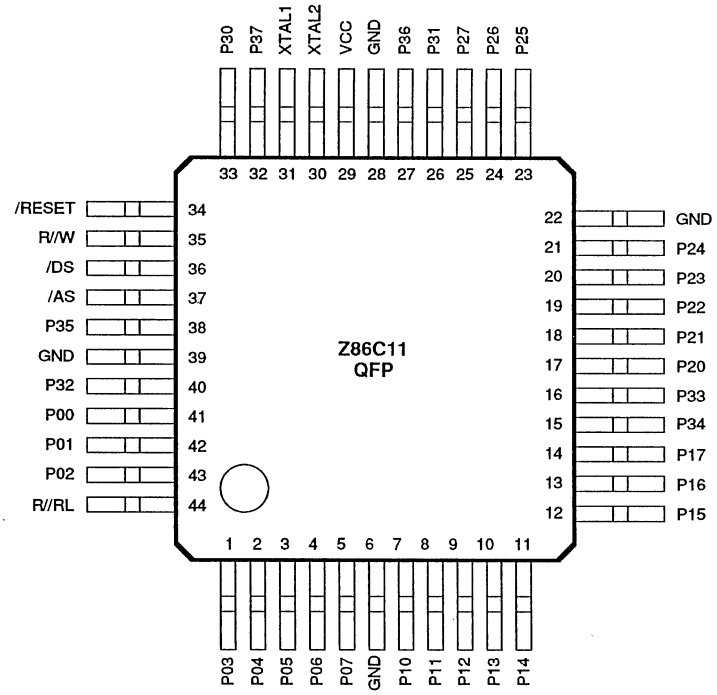


Figure 4. 44-Pin Quad Flat Pack Pin Assignments

Table 3. 44-Pin Quad Flat Pack Pin Identification

Pin #	Symbol	Function	Direction
1-5	P03-P07	Port 0 pin 3,4,5,6,7	In/Output
6	GND	Ground	Input
7-14	P10-P17	Port 1 pin 0,1,2,3,4,5,6,7	In/Output
15	P34	Port 3 pin 4	Output
16	P33	Port 3 pin 3	Input
17-21	P20-P24	Port 2 pin 0,1,2,3,4	In/Output
22	GND	Ground	Input
23-25	P25-P27	Port 2 pin 5,6,7	In/Output
26	P31	Port 3 pin 1	Input
27	P36	Port 3 pin 6	Output
28	GND	Ground	Input
29	V_{CC}	Power Supply	Input
30	XTAL2	Crystal, Oscillator Clock	Output

Pin #	Symbol	Function	Direction
31	XTAL1	Crystal, Oscillator Clock	Input
32	P37	Port 3 pin 7	Output
33	P30	Port 3 pin 0	Input
34	/RESET	Reset	Input
35	R//W	Read/Write	Output
36	/DS	Data Strobe	Output
37	/AS	Address Strobe	Output
38	P35	Port 3 pin 5	Output
39	GND	Ground	Input
40	P32	Port 3 pin 2	Input
41-43	P00-P02	Port 0 pin 0,1,2	In/Output
44	R//RL	ROM/ROMless control	Input

PIN FUNCTIONS

/ROMless. (input, active Low). This pin when connected to GND disables the internal ROM and forces the device to function as a Z86C91 ROMless Z8. (Note that, when left unconnected or pulled high to V_{CC}, the part functions as a normal Z86C11 ROM version). This pin is only available on the 44-pin versions of the Z86C11.

/DS. (output, active Low). Data Strobe is activated once for each external memory transfer. For a READ operation, data must be available prior to the trailing edge of /DS. For WRITE operations, the falling edge of /DS indicates that output data is valid.

/AS. (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Address output is via Port 1 for all external programs. Memory address transfers are valid at the trailing edge of /AS. Under program control, /AS can be placed in the high-impedance state along with Ports 0 and 1, Data Strobe, and Read/Write.

XTAL1, XTAL2. *Crystal 1, Crystal 2* (time-based input and output, respectively). These pins connect a parallel-resonant crystal, ceramic resonator, LC, or any external single-phase clock to the on-chip oscillator and buffer.

R/W. (output, write Low). The Read/Write signal is low when the MCU is writing to the external program or data memory.

/RESET. (input, active-Low). To avoid asynchronous and noisy reset problems, the Z86C11 is equipped with a reset filter of four external clocks (4TpC). If the external /RESET signal is less than 4TpC in duration, no reset occurs. On the 5th clock after the /RESET is detected, an internal RST signal is latched and held for an internal register count of 18 external clocks, or for the duration of the external /RESET, whichever is longer. During the reset cycle, /DS is held active low while /AS cycles at a rate of TpC/2. When /RESET is deactivated, program execution begins at location 000C (HEX). Power-up reset time must be held low for 50 mS, or until VCC is stable, whichever is longer.

Port 0. (P00-P07). Port 0 is an 8-bit, nibble programmable, bidirectional, TTL compatible port. These eight I/O lines can be configured under software control as a nibble I/O port, or as an address port for interfacing external memory. When used as an I/O port, Port 0 is placed under handshake control. In this configuration, Port 3, lines P32 and P35 are used as the handshake control /DAV0 and RDY0 (Data available and Ready). Handshake signal assignment is dictated by the I/O direction of the upper nibble P04-P07. The lower nibble must have the same direction as the upper nibble to be under handshake control. For the ROMless option, Port 0 comes up as A15-A8 Address lines after /RESET.

For external memory references, Port 0 can provide address bit A11-A8 (lower nibble) or A15-A8 (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. If one or both nibbles are needed for I/O operation, they must be configured by writing to the Port 0 Mode register. In ROMless mode, after a hardware reset, Port 0 lines are defined as address lines A15-A8, and extended timing is set to accommodate slow memory access. The initialization routine includes reconfiguration to eliminate this extended timing mode (Figure 5).

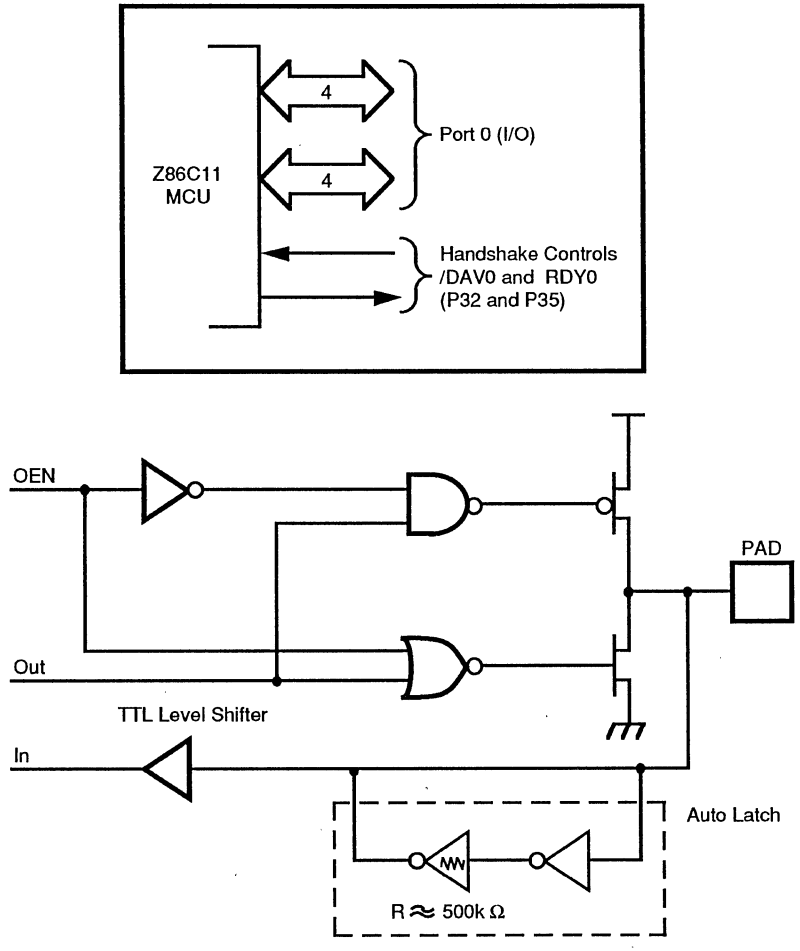


Figure 5. Port 0 Configuration

PIN FUNCTION (Continued)

Port 1. (P10-P17). Port 1 is an 8-bit, byte programmable, bidirectional, TTL compatible port. It has multiplexed Address (A7-A0) and Data (D7-D0) ports. For Z86C11, these eight I/O lines can be programmed as Input or Output lines or can be configured under software control as an address/data port for interfacing external memory. When used as an I/O port, Port 1 is placed under handshake control. In this configuration, Port 3 lines P33 and P34 are used as the handshake controls RDY1 and /DAV1.

Memory locations greater than 4096 are referenced through Port 1. To interface external memory, Port 1 is programmed for the multiplexed Address/Data mode. If more than 256 external locations are required, Port 0 must output the additional lines.

Port 1 can be placed in high-impedance state along with Port 0, /AS, /DS and R//W, allowing the MCU to share common resource in multiprocessor and DMA applications. Data transfers can be controlled by assigning P33 as a Bus Acknowledge input, and P34 as a Bus request output (Figure 6).

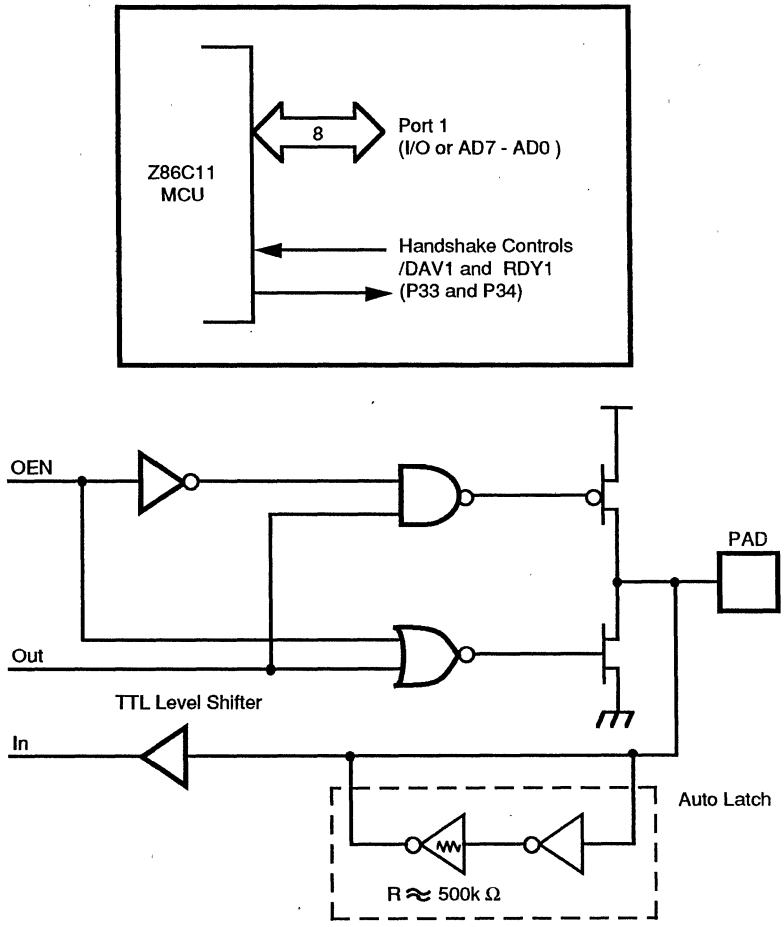


Figure 6. Port 1 Configuration

Port 2. (P20-P27). Port 2 is an 8-bit, bit programmable, bidirectional, TTL compatible port. Each of these eight I/O lines can be independently programmed as an input or output or globally as an open-drain output. Port 2 is always available for I/O operation. When used as an I/O port, Port 2 may be placed under handshake control. In this configuration, Port 3 lines P31 and P36 are used as the handshake controls lines /DAV2 and RDY2. The handshake signal assignment for Port 3 lines P31 and P36 is dictated by the direction (input or output) assigned to P27 (Figure 7).

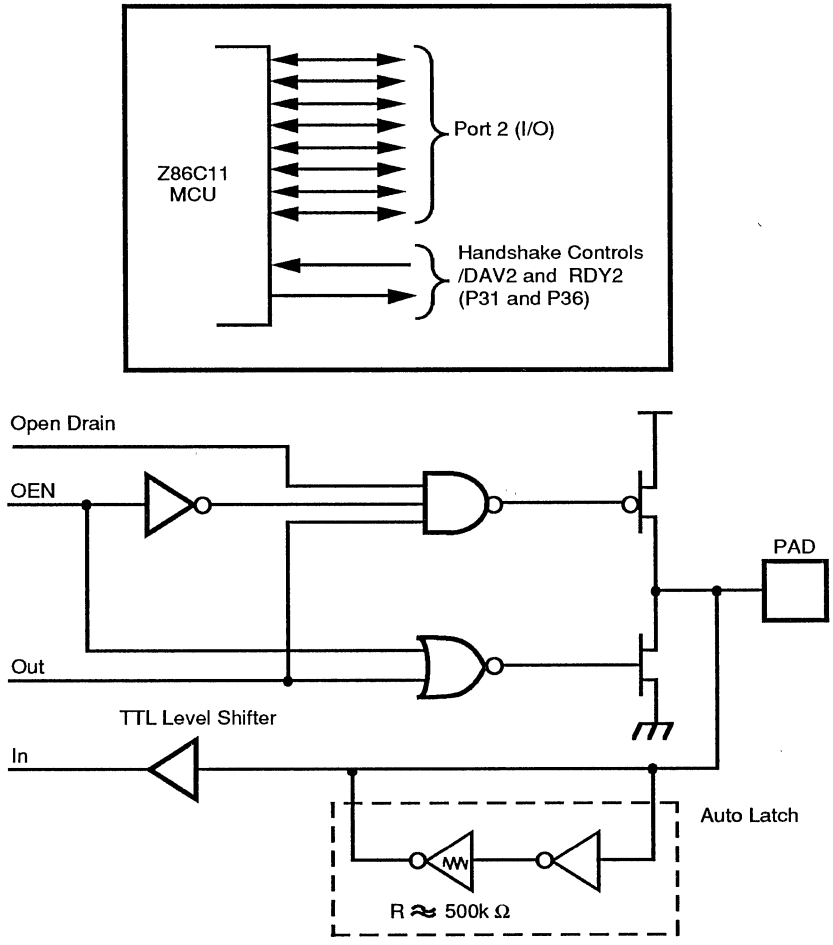


Figure 7. Port 2 Configuration

PIN FUNCTION (Continued)

Port 3. (P30-P37). Port 3 is an 8-bit, TTL compatible four-fixed input and four-fixed output port. These eight I/O lines have four-fixed (P30-P33) input and four fixed (P34-P37) output ports. Port 3 pins P30 and P37, when used as serial I/O, are programmed as serial in and serial out, respectively (Figure 8).

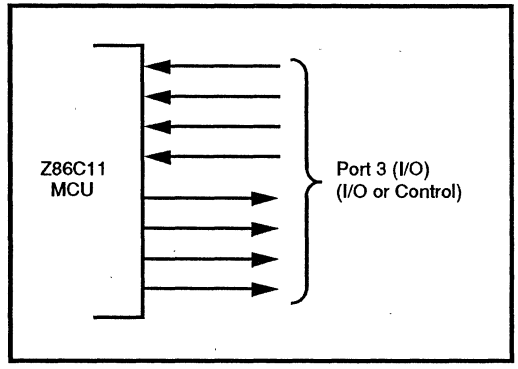


Figure 8. Port 3 Configuration

Port 3 is configured under software control to provide the following control functions: handshake for Ports 0, 1 and 2 (/DAV and RDY); four external interrupt request signals (IRQ0-IRQ3); timer input and output signals (T_{IN} and T_{OUT}), and Data Memory Select (/DM).

Table 4. Port 3 Pin Assignments

Pin	I/O	CTC1	Int.	P0 HS	P1 HS	P2 HS	UART	Ext
P30	IN		IRQ3				Serial In	
P31	IN	T_{IN}	IRQ2			D/R		
P32	IN		IRQ0	D/R				
P33	IN		IRQ1		D/R			
P34	OUT				R/D			DM
P35	OUT			R/D				
P36	OUT	T_{OUT}				R/D		
P37	OUT						Serial Out	

Notes:
HS = HANDSHAKE SIGNALS
D = Data Available
R = Ready

Port 3 lines P30 and P37, can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by the Counter/Timer 0.

The Z86C11 automatically adds a start bit and two stop bits to transmitted data (Figure 9). Odd parity is also available as an option. Eight data bits are always transmitted, regardless of parity selection. If parity is enabled, the eighth bit is the odd parity bit. An interrupt request (IRQ4) is generated on all transmitted characters.

Received data must have a start bit, eight data bits and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ3 interrupt request.

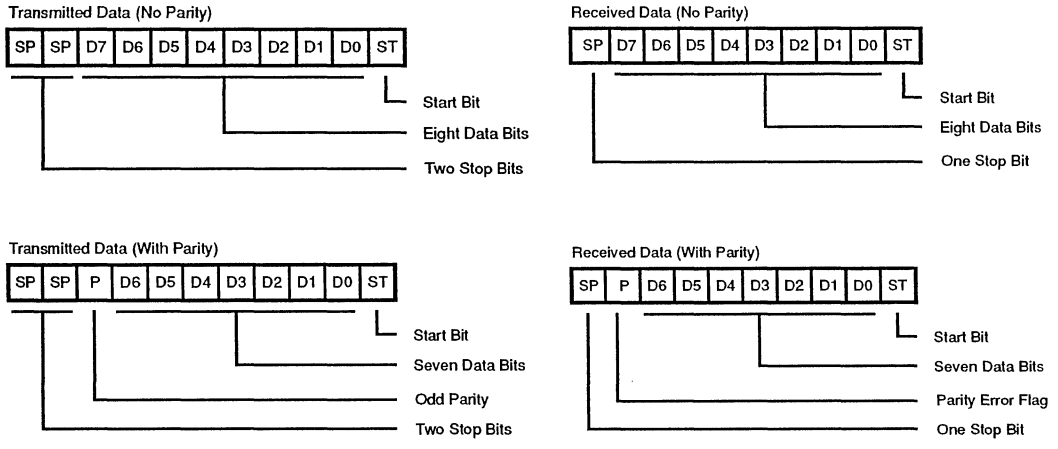


Figure 9. Serial Data Formats

Auto-Latch. The Auto-Latch puts valid CMOS levels on all CMOS inputs that are not externally driven. This will reduce excessive supply current flow in the input buffer when it is not been driven by any source.

Low EMI Option. The Z86C11 is available in a low EMI option. This option is mask-programmable, to be selected by the customer at the time when the ROM code is submitted. Use of this feature results in:

- Less than 1 mA current consumptions during HALT mode.
- The pre-drivers slew rate reduced to 10 ns typical.
- Low EMI output drivers have resistance of 200 ohms typical.
- Oscillator divide-by-two circuitry is eliminated.
- Internal SCLK/TCLK operation is limited to a maximum of 4 MHz (250 ns cycle time)

FUNCTIONAL DESCRIPTION

Address Space

Program Memory. The Z86C11 can address up to 60 K bytes of external program memory (Figure 10). The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts. For ROM mode, byte 13 to byte 4095 consists of on-chip ROM. At address 4096 and greater, the Z86C11 executes external program memory fetches. In the ROMless mode, the Z86C11 can address up to 64 Kbytes of external program memory. Program execution begins at external location 000C (HEX) after a reset.

Data Memory (/DM). The ROM version can address up to 60 Kbytes of external data memory space beginning at location 4096. The ROMless version can address up to 64 Kbytes of external data memory. External data memory may be included with, or separated from, the external program memory space. /DM, an optional I/O function that can be programmed to appear on pin P34, is used to distinguish between data and program memory space (Figure 11). The state of the /DM signal is controlled by the type instruction being executed. An LDC opcode references PROGRAM (/DM inactive) memory, and an LDE instruction references DATA (/DM active low) memory.

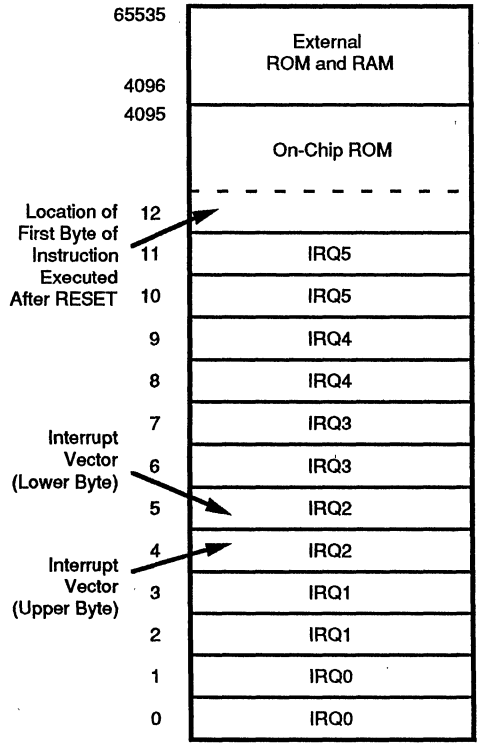


Figure 10. Program Memory Configuration

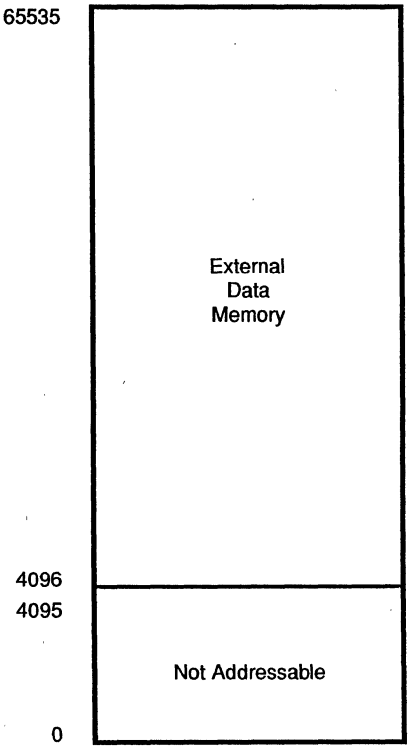


Figure 11. Data Memory Configuration

Register File. The Register File consists of four I/O port registers, 236 general-purpose registers and 16 control and status registers (Figure 12). The instructions can access registers directly or indirectly via an 8-bit address field. The Z86C11 also allows short 4-bit register addressing using the Register Pointer (Figure 13). In the 4-bit mode, the Register File is divided into 16 working register groups, each occupying 16 continuous locations. The Register Pointer addresses the starting location of the active working-register group.

Note: Register Bank E0-EF can only be accessed through working registers and indirect addressing modes.

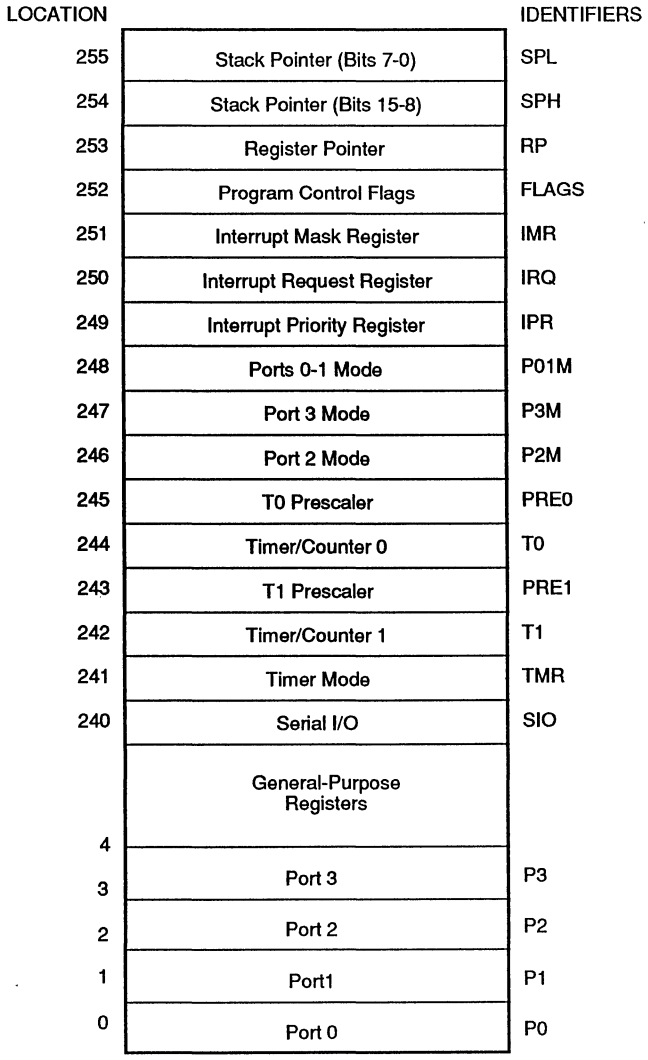


Figure 12. Register File

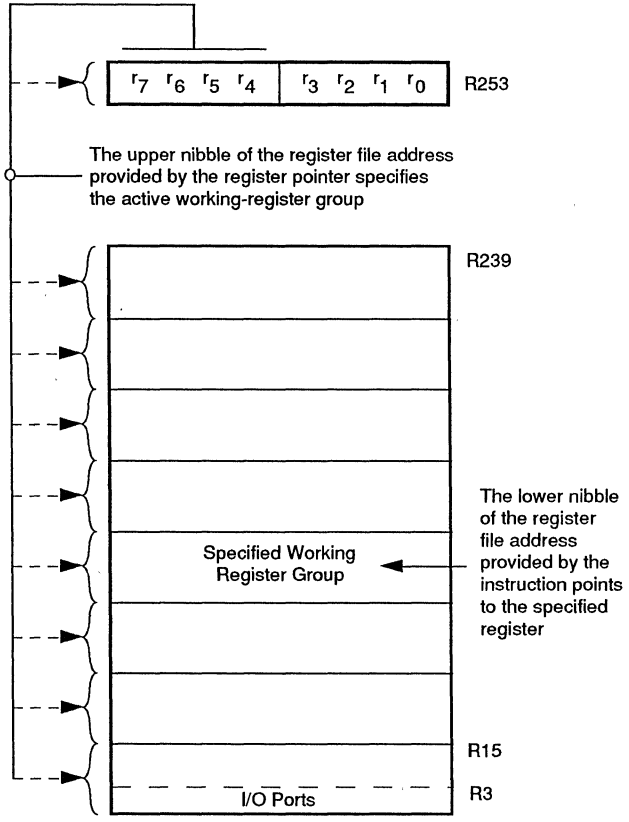


Figure 13. Register Pointer

RAM Protect. The upper portion of the RAM's address spaces 80H to EFH (excluding the control registers) can be protected from reading and writing. The RAM Protect bit option is mask-programmable and is selected by the customer when the ROM code is submitted. After the mask option is selected, the user can activate from the internal ROM code to turn off/on the RAM Protect by loading a bit D6 in the IMR register to either a 0 or a 1, respectively. A 1 in D6 indicates RAM Protect enabled.

ROM Protect. The first 4 Kbytes of program memory is mask programmable. A ROM protect feature prevents dumping of the ROM contents by inhibiting execution of LDC, LDCI, LDE, and LDEI instructions to Program Memory in all modes.

The ROM Protect option is mask-programmable, to be selected by the customer at the time when the ROM code is submitted.

Stack. The Z86C11 has a 16-bit Stack Pointer (R254-R255) used for external stack that resides anywhere in the data memory for the ROMless mode, but only from 4096 to 65535 in the ROM mode. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 236 general-purpose registers (R4-R239). The high byte of the Stack Pointer (SPH-Bit 8-15) is used as a general purpose register when using internal stack only.

Counter/Timers. There are two 8-bit programmable counter/timers (T0-T1), each driven by its own 6-bit programmable prescaler. The T1 prescaler can be driven by internal or external clock sources; however, the T0 prescaler is driven by the internal clock only (Figure 14).

The 6-bit prescalers can divide the input frequency of the clock source by any integer number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When both the counters and prescaler reach the end of the count, a timer interrupt request, IRQ4 (T0) or IRQ5 (T1), is generated.

The counter can be programmed to start, stop, restart to continue, or restart from the initial value. The counters can also be programmed to stop upon reaching zero (single pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode).

The counter, but not the prescalers, can be read at any time without disturbing their value or count mode. The clock source for T1 is user-definable and can be either the internal microprocessor clock divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input (P31) as an external clock, a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. Port 3 line P36 also serves as a timer output (TOUT) through which T0, T1 or the internal clock can be output. The counter/timers are cascaded by connecting the T0 output to the input of T1.

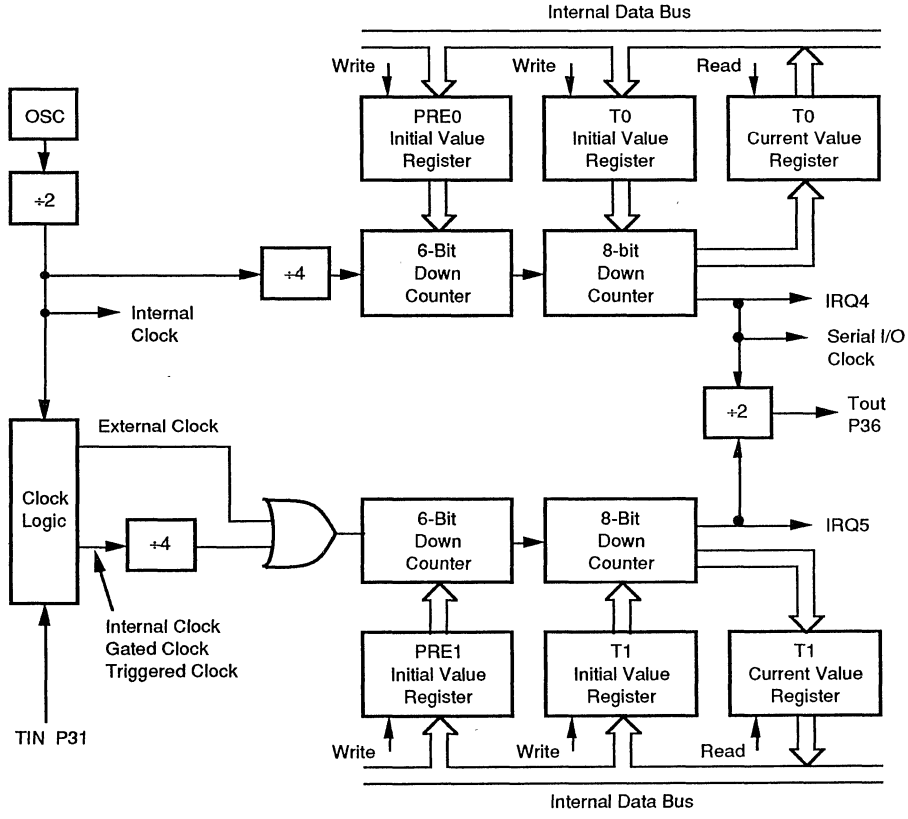


Figure 14. Counter/Timers Block Diagram

FUNCTIONAL DESCRIPTION (Continued)

Interrupts. The Z86C11 has six different interrupts from eight different sources. The interrupts are maskable and prioritized. The eight sources are divided as follows: four sources are claimed by Port 3 lines P30-P33, one in Serial Out, one in Serial In, and two in the counter/timers (Figure 15). The Interrupt Mask Register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All Z86C11 interrupts are vectored through locations in the program memory. When an interrupt machine cycle is activated, an interrupt request is granted. Thus, this disables all of the subsequent interrupts, saves the Program Counter and Status Flags, and then branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

To accommodate polled interrupt systems, interrupt inputs are masked and the Interrupt Request register is polled to determine which of the interrupt requests need service. Software initialed interrupts are supported by setting the appropriate bit in the Interrupt Request (IRQ) register.

Internal interrupt requests are sampled on the falling edge of the last cycle of every instruction, and the interrupt request must be valid 5TpC before the falling edge of the last clock cycle of the currently executing instruction.

For the ROMless mode, when the device samples a valid interrupt request, the next 48 (external) clock cycles are used to prioritize the interrupt, and push the two PC bytes and the FLAG register on the stack. The following nine cycles are used to fetch the interrupt vector from external memory. The first byte of the interrupt service routine is fetched beginning on the 58th TpC cycle following the internal sample point, which corresponds to the 63rd TpC cycle following the external interrupt sample point.

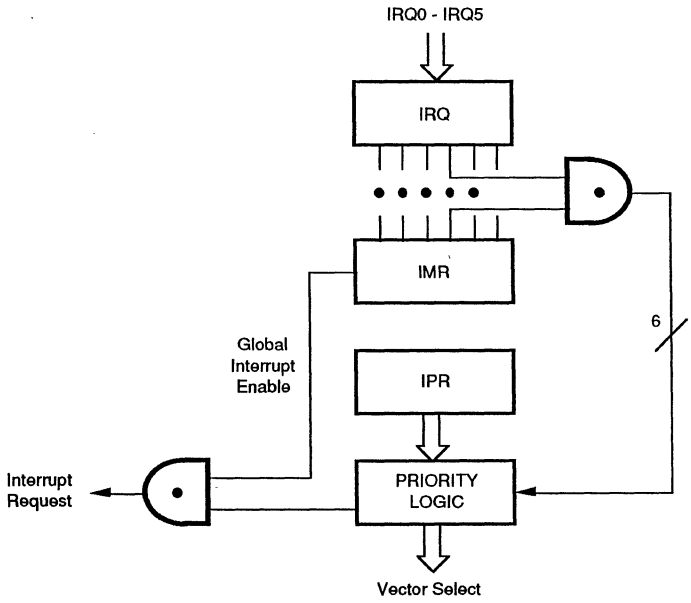


Figure 15. Interrupt Block Diagram

Clock. The Z86C11 on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal, LC, ceramic resonator, or any suitable external clock source (XTAL1=Input, XTAL2 = Output). The crystal should be AT cut, 1 MHz to 16 MHz max, and series resistance (RS) is less than or equal to 100 Ohms. The crystal should be connected across XTAL1 and XTAL2 using the recommended capacitors (10 pF < CL < 300 pF) from each pin to ground (Figure 16).

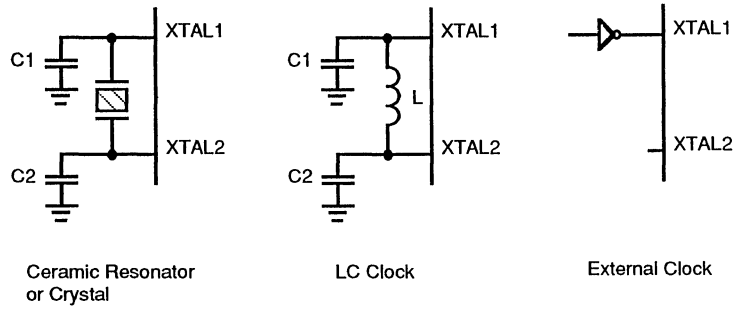


Figure 16. Oscillator Configuration

HALT. Will turn off the internal CPU clock but not the XTAL oscillation. The counter/timers and the external interrupts IRQ0, IRQ1, IRQ2 and IRQ3 remains active. The devices are recovered by interrupts, either externally or internally generated.

STOP. This instruction turns off the internal clock and external crystal oscillation and reduces the standby current to 10 microamperes or less. The Stop mode is terminated by a reset, which causes the processor to restart the application program at address 000C (HEX).

In order to enter STOP (or HALT) mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. To do this, the user executes a NOP (opcode=OFFH) immediately before the appropriate sleep instruction,. i.e.:

```
FF  NOP   ; clear the pipeline
6F  STOP  ; enter STOP mode
          or
FF  NOP   ; clear the pipeline
7F  HALT  ; enter HALT mode
```

ABSOLUTE MAXIMUM RATINGS

Symbol	Description	Min	Max	Units
V_{CC}	Supply Voltage*	-0.3	+7.0	V
T_{STG}	Storage Temp	-65	+150	°C
T_A	Oper Ambient Temp		†	°C

Notes:
* Voltages on all pins with respect to GND.
† See Ordering Information

Stress greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for an extended period may affect device reliability.

STANDARD TEST CONDITIONS

The characteristics listed below apply for standard test conditions as noted. All voltages are referenced to GND. Positive current flows into the referenced pin (Figure 17).

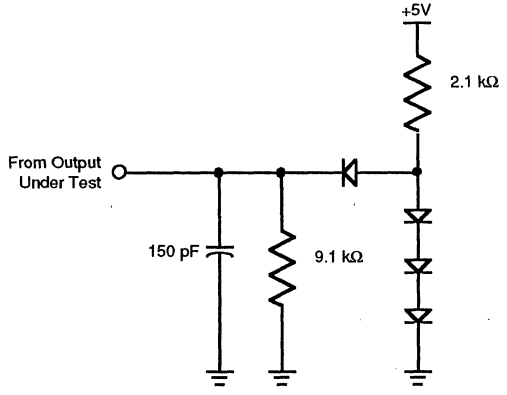


Figure 17. Test Load Diagram

DC CHARACTERISTICS

Sym	Parameter	T_A = 0°C to 70°C		T_A = -40°C to105°C		Typical at 25°C	Units	Conditions
		Min	Max	Min	Max			
	Max Input Voltage		7		7		V	I_{IN} 250µA
V_{CH}	Clock Input High Voltage	3.8	V_{CC}+0.3	3.8	V_{CC}+0.3		V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.03	0.8	-0.03	0.8		V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}+0.3	2.0	V_{CC}+0.3		V	
V_{IL}	Input Low Voltage	-0.3	0.8	-0.3	0.8		V	
V_{OH}	Output High Voltge	2.4		2.4			V	I_{OH} = -2.0 mA
V_{OH}	Output High Voltge	V_{CC}-100mV		V_{CC}-100mV			V	I_{OH} = -100 µA
V_{OL}	Output Low Voltage		0.4		0.4		V	I_{OL} = +5.0 mA
V_{RH}	Reset Input High Voltage	3.8	V_{CC}+0.3	3.8	V_{CC}+0.3		V	
V_{RI}	Reset Input Low Voltage	-0.03	0.8	-0.03	0.8		V	
I_{IL}	Input Leakage	-2	2	-2	2		µA	V_{IN} = 0V, V_{CC}
I_{OL}	Output Leakage	-2	2	-2	2		µA	V_{IN} = 0V, V_{CC}
I_{IR}	Reset Input Current		-80		-80		µA	V_{RL} = 0V
I_{CC}	Supply Current		30		30	20	mA	[1] @ 12 MHz
			35		35	24	mA	[1] @ 16 MHz
I_{CC1}	Standby Current		6.5		6.5	4	mA	[1] HALT Mode V_{IN} = 0V, V_{CC}@ 12 MHz
			7.0		7.0	4.5	mA	[1] HALT Mode V_{IN} = 0V,V_{CC}@ 16 MHz
I_{CC2}	Standby Current		10		20	5	µA	[1,2] STOP Mode V_{IN} = 0V,V_{CC}

Notes:
[1] All inputs driven to either 0V or V_{CC}, outputs floating.
[2] I_{CC2} requires loading TMR (F1H) with any value prior to STOP execution.
Use this sequence:

```
LD TMR,#00
NOP
STOP
```

AC CHARACTERISTICS

External I/O or Memory Read or Write Timing Diagram

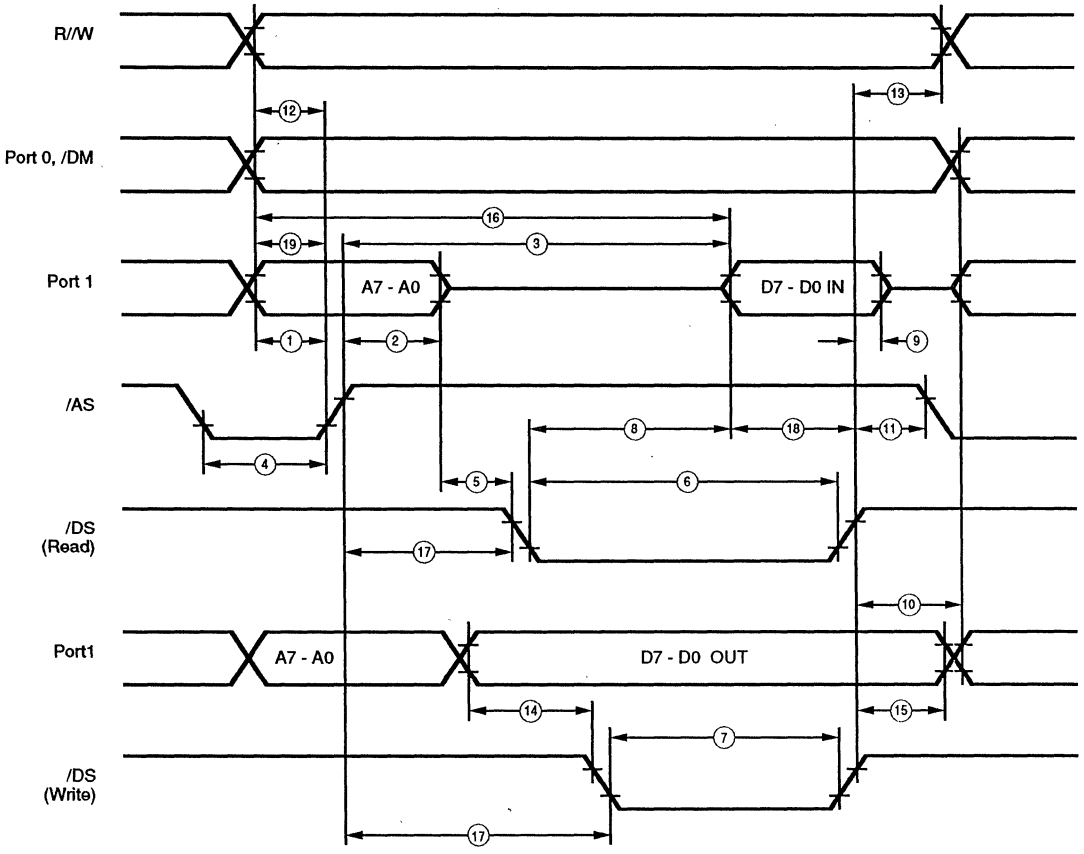


Figure 18. External I/O or Memory Read/Write Timing

AC CHARACTERISTICS

External I/O or Memory Read or Write Timing Table

No	Symbol	Parameter	T_A = 0°C to 70°C				T_A = -40°C to 105°C				Units	Notes
			12 MHz		16 MHz		12 MHz		16 MHz			
			Min	Max	Min	Max	Min	Max	Min	Max		
1	TdA(AS)	Address Valid to /AS Rise Delay	35		25		35		25		ns	[2,3]
2	TdAS(A)	/AS Rise to Address Float Delay	45		35		45		35		ns	[2,3]
3	TdAS(DR)	/AS Rise to Read Data Req'd Valid		250		180		250		180	ns	[1,2,3]
4	TwAS	/AS Low Width	55		40		55		40		ns	[2,3]
5	TdAZ(DS)	Address Float to /DS Fall	0		0		0		0		ns	
6	TwDSR	/DS (Read) Low Width	185		135		185		135		ns	[1,2,3]
7	TwDSW	/DS (Write) Low Width	110		80		110		80		ns	[1,2,3]
8	TdDSR(DR)	/DS Fall to Read Data Req'd Valid		130		75		130		75	ns	[1,2,3]
9	ThDR(DS)	Read Data to /DS Rise Hold Time	0		0		0		0		ns	[2,3]
10	TdDS(A)	/DS Rise to Address Active Delay	65		50		65		50		ns	[2,3]
11	TdDS(AS)	/DS Rise to /AS Fall Delay	45		35		45		35		ns	[2,3]
12	TdR/W(AS)	R/W Valid to /AS Rise Delay	30		20		33		25		ns	[2,3]
13	TdDS(R/W)	/DS Rise to R/W Not Valid	50		35		50		35		ns	[2,3]
14	TdDW(DSW)	Write Data Valid to /DS Fall (Write) Delay	35		25		35		25		ns	[2,3]
15	TdDS(DW)	/DS Rise to Write Data Not Valid Delay	55		35		55		35		ns	[2,3]
16	TdA(DR)	Address Valid to Read Data Req'd Valid		310		230		310		230	ns	[1,2,3]
17	TdAS(DS)	/AS Rise to /DS Fall Delay	65		45		65		45		ns	[2,3]
18	TdDI(DS)	Data Input Setup to /DS Rise	75		60		75		60		ns	[1,2,3]
19	TdDM(AS)	/DM Valid to /AS Rise Delay	50		30		50		30		ns	[2,3]

Notes:

[1] When using extended memory timing add 2 TpC.

[2] Timing numbers given are for minimum TpC.

[3] See clock cycle dependent characteristics table.

Standard Test Load

All timing references use 2.0V for a logic 1 and 0.8V for a logic 0.

Clock Dependent Formulas

Number	Symbol	Equation
1	TdA(AS)	0.40TpC + 0.32
2	TdAS(A)	0.59TpC - 3.25
3	TdAS(DR)	2.38TpC + 6.14
4	TwAS	0.66TpC - 1.65
6	TwDSR	2.33TpC - 10.56
7	TwDSW	1.27TpC + 1.67
8	TdDSR(DR)	1.97TpC - 42.5
10	TdDS(A)	0.8TpC
11	TdDS(AS)	0.59TpC - 3.14
12	TdR/W(AS)	0.4TpC
13	TdDS(R/W)	0.8TpC - 15
14	TdDW(DSW)	0.4TpC
15	TdDS(DW)	0.88TpC - 19
16	TdA(DR)	4TpC - 20
17	TdAS(DS)	0.91TpC - 10.7
18	TsDI(DS)	0.8TpC - 10
19	TdDM(AS)	0.9TpC - 26.3

AC CHARACTERISTICS
Additional Timing Diagram

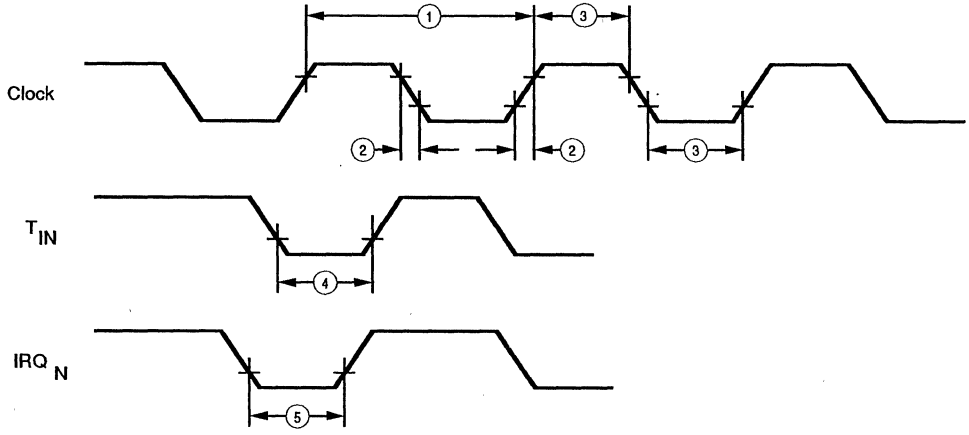


Figure 19. Additional Timing

AC CHARACTERISTICS
Additional Timing Table

No	Symbol	Parameter	T_A = 0°C to 70°C				T_A = -40°C to 105°C				Units	Notes
			12 MHz		16 MHz		12 MHz		16 MHz			
			Min	Max	Min	Max	Min	Max	Min	Max		
1	TpC	Input Clock Period	83	1000	62.5	1000	83	1000	62.5	1000	ns	[1]
2	TrC,TfC	Clock Input Rise & Fall Times		15		10		15		10	ns	[1]
3	TwC	Input Clock Width	35		25		35		25		ns	[1]
4	TwTinL	Timer Input Low Width	75		75		75		75		ns	[2]
5	TwTinH	Timer Input High Width	3TpC		3TpC		3TpC		3TpC			[2]
6	TpTin	Timer Input Period	8TpC		8TpC		8TpC		8TpC			[2]
7	TrTin,TfTin	Timer Input Rise & Fall Times	100		100		100		100		ns	[2]
8A	TwIL	Interrupt Request Input Low Times	70		70		70		50		ns	[2,4]
8B	TwIL	Interrupt Request Input Low Times	3TpC		3TpC		3TpC		3TpC			[2,5]
9	TwIH	Interrupt Request Input High Times	3TpC		3TpC		3TpC		3TpC			[2,3]

Notes:
[1] Clock timing references use 3.8V for a logic 1 and 0.8V for a logic 0.
[2] Timing references use 2.0V for a logic 1 and 0.8V for a logic 0.
[3] Interrupt references request via Port 3.
[4] Interrupt request via Port 3 (P31-P33).
[5] Interrupt request via Port 30.

AC CHARACTERISTICS
Handshake Timing Diagrams

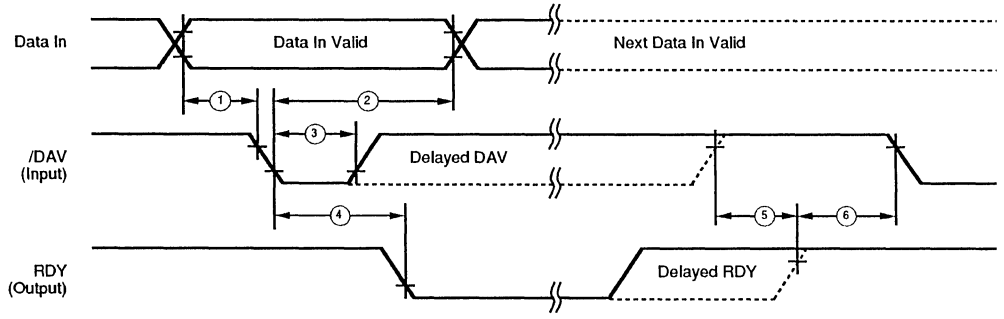


Figure 20. Input Handshake Timing

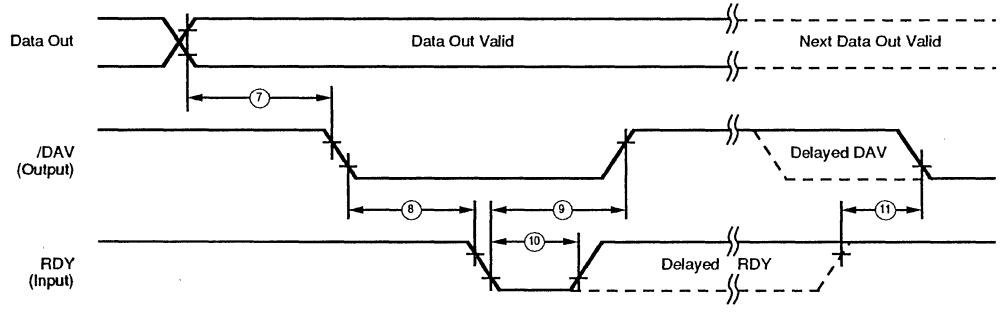


Figure 21. Output Handshake Timing

AC CHARACTERISTICS
Handshake Timing Table

No	Symbol	Parameter	T_A = 0°C to 70°C				T_A = -40°C to 105°C				Notes
			12 MHz		16 MHz		12 MHz		16 MHz		Data
			Min	Max	Min	Max	Min	Max	Min	Max	Direction
1	TsDI(DAV)	Data In Setup Time	0		0		0		0		IN
2	ThDI(DAV)	Data In Hold Time	145		145		145		145		IN
3	TwDAV	Data Available Width	110		110		110		110		IN
4	TdDAVI(RDY)	DAV Fall to RDY Fall Delay		115		115		115		115	IN
5	TdDAVId(RDY)	DAV Rise to RDY Rise Delay		115		115		115		115	IN
6	TdDO(DAV)	RDY Rise to DAV Fall Delay	0		0		0		0		IN
7	TcLDAV0(RDY)	Data Out to DAV Fall Delay		TpC		TpC		TpC		TpC	OUT
8	TcLDAV0(RDY)	DAV Fall to RDY Fall Delay	0		0		0		0		OUT
9	TdRDY0(DAV)	RDY Fall to DAV Rise Delay		115		115		115		115	OUT
10	TwRDY	RDY Width	110		110		110		110		OUT
11	TdRDY0d(DAV)	RDY Rise to DAV Fall Delay		115		115		115		115	OUT

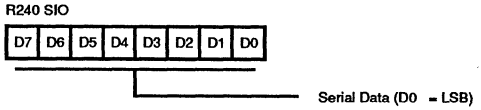


Figure 22. Serial I/O Register (F0H: Read/Write)

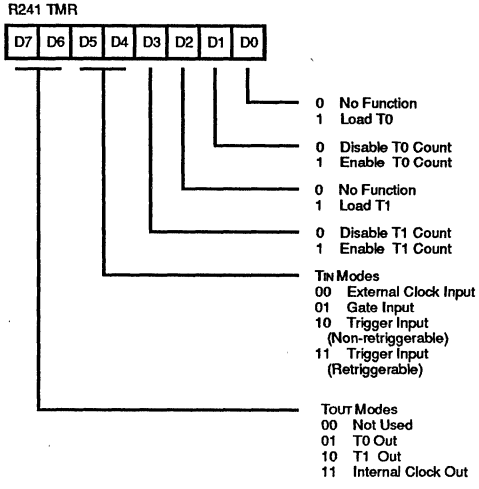


Figure 23. Timer Mode Register (F1H: Read/Write)

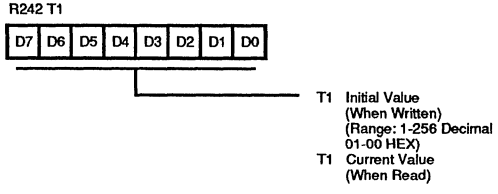


Figure 24. Counter/Timer 1 Register (F2H: Read/Write)

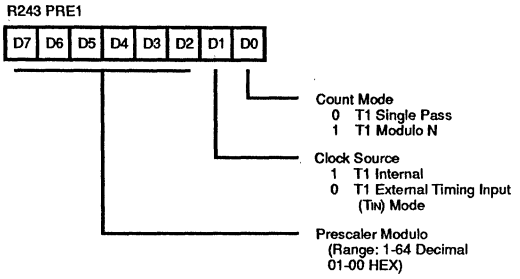


Figure 25. Prescaler 1 Register (F3H: Write Only)

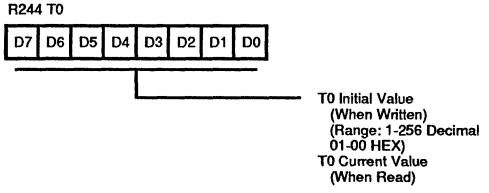


Figure 26. Counter/Timer 0 Register (F4H: Read/Write)

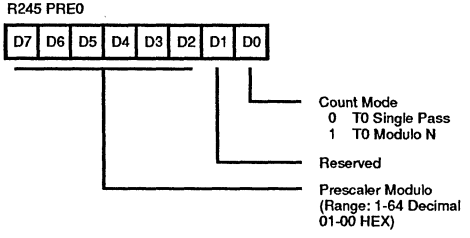


Figure 27. Prescaler 0 Register (F5H: Write Only)

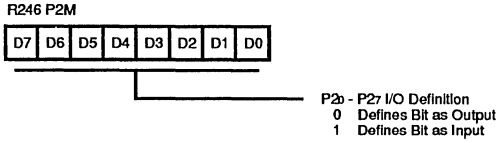


Figure 28. Port 2 Mode Register (F6H: Write Only)

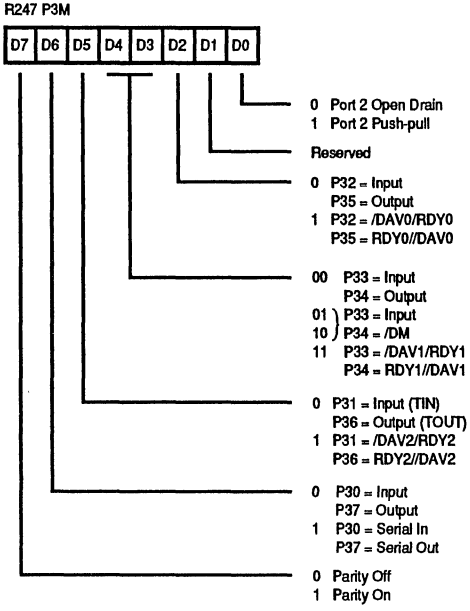


Figure 29. Port 3 Mode Register (F7H: Write Only)

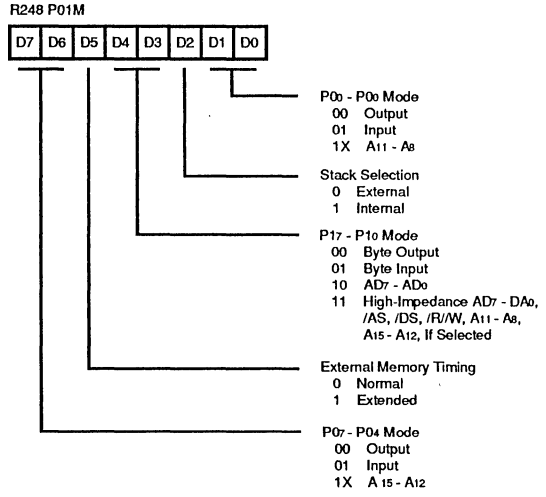


Figure 30. Port 0 and 1 Mode Register (F8H: Write Only)

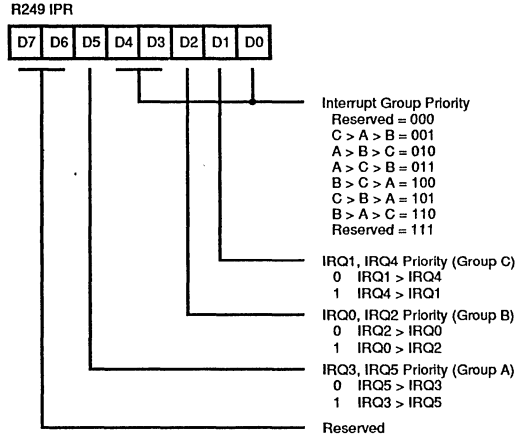


Figure 31. Interrupt Priority Register (F9H: Write Only)

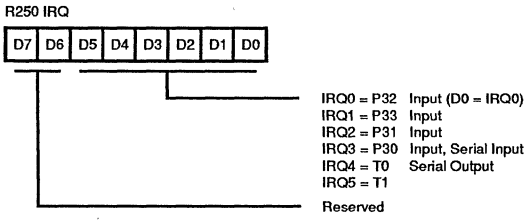


Figure 32. Interrupt Request Register (FAH: Read/Write)

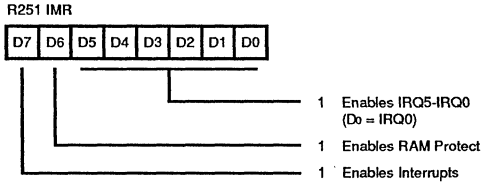


Figure 33. Interrupt Mask Register (FBH: Read/Write)

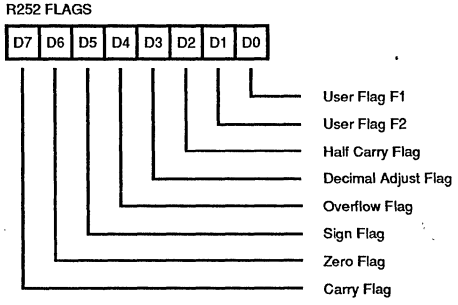


Figure 34. Flag Register (FCH: Read/Write)

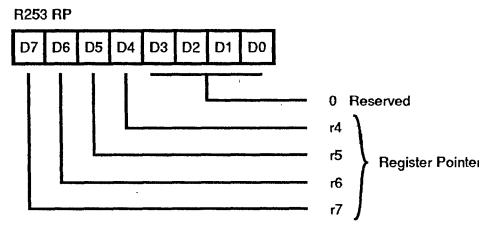


Figure 35. Register Pointer Register (FDH: Read/Write)

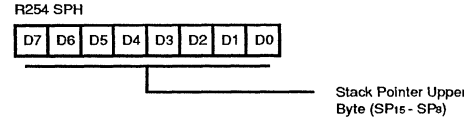


Figure 36. Stack Pointer Register (FEH: Read/Write)

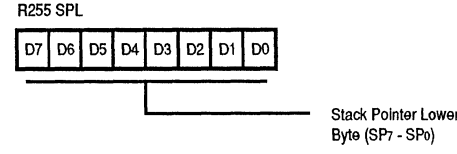


Figure 37. Stack Pointer Register (FFH: Read/Write)

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

Symbol	Meaning
IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

Symbol	Meaning
dst	Destination location or contents
src	Source location or contents
cc	Condition code
@	Indirect address prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag register (Control Register 252)
RP	Register Pointer (R253)
IMR	Interrupt mask register (R251)

Flags. Control register (R252) contains the following six flags:

Symbol	Meaning
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

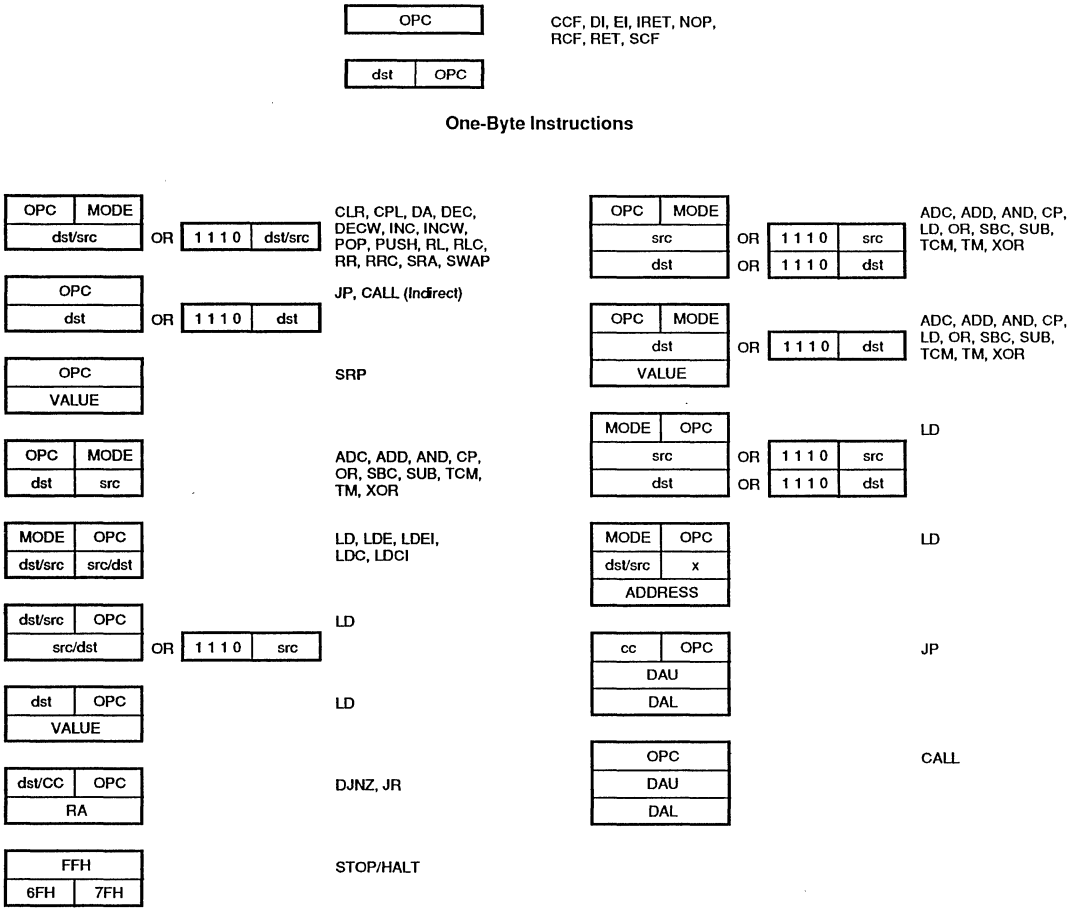
Affected flages are indicated by:

0	Clear to zero
1	Set to one
*	Set to clear according to operation
-	Unaffected
x	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000		Always True	
0111	C	Carry	C = 1
1111	NC	No Carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not Zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No Overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not Equal	Z = 0
1001	GE	Greater Than or Equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater Than	[Z OR (S XOR V)] = 0
0010	LE	Less Than or Equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned Greater Than or Equal	C = 0
0111	ULT	Unsigned Less Than	C = 1
1011	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned Less Than or Equal	(C OR Z) = 1
0000		Never True	

INSTRUCTION FORMATS



INSTRUCTION SUMMARY

Note: Assignment of a value is indicated by the symbol " ← ". For example:

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr (n)" is used to refer to bit (n) of a given operand location. For example:

dst (7)

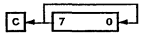
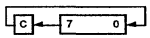
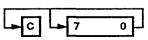
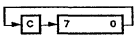
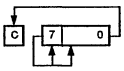
refers to bit 7 of the destination operand.

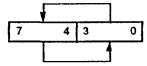
INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst, src dst←dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst←dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst←dst AND src	†		5[ ]	-	*	*	0	-	-
CALL dst SP←SP - 2 @SP←PC, PC←dst	DA IRR		D6 D4	-	-	-	-	-	-
CCF C←NOT C			EF	*	-	-	-	-	-
CLR dst dst←0	R IR		B0 B1	-	-	-	-	-	-
COM dst dst←NOT dst	R IR		60 61	-	*	*	0	-	-
CP dst, src dst - src	†		A[ ]	*	*	*	*	-	-
DA dst dst←DA dst	R IR		40 41	*	*	*	X	-	-
DEC dst dst←dst - 1	R IR		00 01	-	*	*	*	-	-
DECW dst dst←dst - 1	RR IR		80 81	-	*	*	*	-	-
DI IMR(7)←0			8F	-	-	-	-	-	-
DJNZr, dst r←r - 1 if r ≠ 0 PC←PC + dst Range: +127, -128	RA		rA r = 0 - F	-	-	-	-	-	-
EI IMR(7)←1			9F	-	-	-	-	-	-
HALT			7F	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
INC dst dst←dst + 1	r R IR		rE r = 0 - F 20 21	-	*	*	*	-	-
INCW dst dst←dst + 1	RR IR		A0 A1	-	*	*	*	-	-
IRET FLAGS←@SP; SP←SP + 1 PC←@SP; SP←SP + 2; IMR(7)←1			BF	*	*	*	*	*	*
JP cc, dst if cc is true PC←dst	DA IRR		cD c = 0 - F 30	-	-	-	-	-	-
JR cc, dst if cc is true, PC←PC + dst Range: +127, -128	RA		cB c = 0 - F	-	-	-	-	-	-
LD dst, src dst←src	r r R r X r r Ir R R IR IR	Im R r X r Ir r R IR IM IM R	rC r8 r9 r = 0 - F C7 D7 E3 F3 E4 E5 E6 E7 F5	-	-	-	-	-	-
LDC dst, src	r	Irr	C2	-	-	-	-	-	-
LDCI dst, src dst←src r←r +1; rr←rr + 1	Ir	Irr	C3	-	-	-	-	-	-

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
NOP			FF	-	-	-	-	-	-
OR dst, src dst←dst OR src	†		4[ ]	-	*	*	0	-	-
POP dst dst←@SP; SP←SP + 1	R IR		50 51	-	-	-	-	-	-
PUSH src SP←SP - 1; @SP←src		R IR	70 71	-	-	-	-	-	-
RCF C←0			CF	0	-	-	-	-	-
RET PC←@SP; SP←SP + 2			AF	-	-	-	-	-	-
RL dst 	R IR		90 91	*	*	*	*	-	-
RLC dst 	R IR		10 11	*	*	*	*	-	-
RR dst 	R IR		E0 E1	*	*	*	*	-	-
RRC dst 	R IR		C0 C1	*	*	*	*	-	-
SBC dst, src dst←dst←src←C	†		3[ ]	*	*	*	*	1	*
SCF C←1			DF	1	-	-	-	-	-
SRA dst 	R IR		D0 D1	*	*	*	0	-	-
SRP src RP←src		Im	31	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
STOP			6F	-	-	-	-	-	-
SUB dst, src dst←dst←src	†		2[ ]	*	*	*	*	1	*
SWAP dst 	R IR		F0 F1	X	*	*	X	-	-
TCM dst, src (NOT dst) AND src	†		6[ ]	-	*	*	0	-	-
TM dst, src dst AND src	†		7[ ]	-	*	*	0	-	-
XOR dst, src dst←dst XOR src	†		B[ ]	-	*	*	0	-	-

† These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

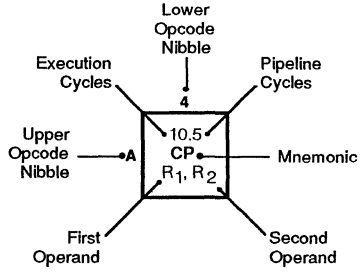
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address Mode dst	src	Lower Opcode Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

OPCODE MAP

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								6.0 **STOP**
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								7.0 **HALT**
8	10.5 **DECW** RR1	10.5 **DECW** IR1	12.0 **LDE** r1, Irr2	18.0 **LDEI** Ir1, Irr2												6.1 **DI**
9	6.5 **RL** R1	6.5 **RL** IR1	12.0 **LDE** r2, Irr1	18.0 **LDEI** Ir2, Irr1												6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14.0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2				10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1	12.0 **LDC** r2, Irr1	18.0 **LDCI** Ir2, Irr1	20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6.5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2				3				2					3	1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_2 = Dst address
R_1 or r_2 = Src address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank are not defined.

* 2-byte instruction appears as a 3-byte instruction

Z86C12

Z8® ICE IN-CIRCUIT EMULATOR

FEATURES

- 8-bit CMOS microcontroller emulator, 84-pin package
- 4.5 to 5.5 Volt operating range
- Low power consumption - 275 mW (max)
- Average instruction execution time of 1 s
- Fast instruction pointer - 0.6 us @ 16 MHz
- Two standby modes - STOP and HALT
- 32 input/output lines
- Full-Duplex UART
- All digital inputs are TTL levels
- Six Memory emulation modes
- 256 bytes of RAM
- Two programmable 8-bit Counter/Timers each with 6-bit programmable prescaler.
- Six vectored, priority interrupt from Eight different sources
- Clock speed 16 MHz
- On-chip oscillator that accepts a crystal, ceramic resonator, LC or external clock drive.

GENERAL DESCRIPTION

The Z86C12 ICE (In-Circuit-Emulator) introduces a new level of sophistication to single-chip architecture.

The ICE is housed in a 84-pin PGA, and is manufactured in CMOS technology.

The ICE development device allows users to prototype a system with an actual hardware device and to develop the code. This code is eventually mask-programmed into the on-chip ROM for any of the Z86CXX devices. Development devices are also useful in emulator applications where the final system configuration, memory configuration, I/O, interrupt inputs, etc., are unknown. The ICE development device is identical to its equivalent Z86C21 microcomputer with the following exceptions:

- No internal ROM is provided, so that code is developed in off-chip memory. Five size inputs configure the memory boundaries.
- The normally internal ROM address and data lines are buffered and brought out to external pins to interface with the external memory.
- Control lines (/MAS and /DAS) are added to interface with external program memory.
- The Timing and Control, I/O ports, and clock pins on the Z86C12 are identical in function to those on the Z86C21.

GENERAL DESCRIPTION (Continued)

The ICE architecture is characterized by Zilog's 8-bit microcontroller core. The device offers; fast execution, more efficient use of memory, more sophisticated interrupts, input/output bit manipulation capabilities, easy hardware/software system expansion, a flexible I/O scheme, an efficient register and address space structure, multiplexed capabilities between address/data, and a number of ancillary features that are useful in many industrial and advanced scientific applications.

Industrial applications demand powerful I/O capabilities. The ICE fulfills this with 32-pins dedicated to input and output. These lines are grouped into four ports. Each port consists of eight lines, and is configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an address/data bus for interfacing external memory.

There are three basic address spaces available to support this wide range of configuration: Program Memory, Data Memory, and 236 General Purpose Registers.

To unburden the program from coping with real-time problems such as counting/timing and serial data communication, the ICE offers two on-chip counter/timers with a large number of user selectable modes, and an asynchronous receiver/transmitter (UART-Figure 1).

Note: All Signals with a preceding front slash, "/", are active Low, e.g.: B/W (WORD is active Low); /B/W (BYTE is active Low, only).

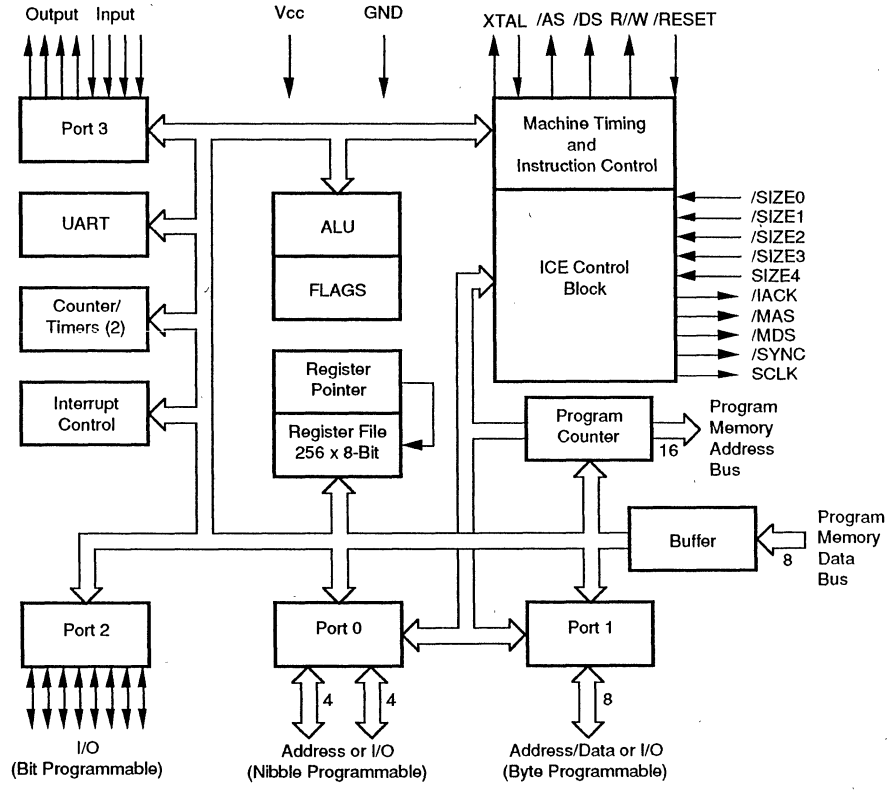


Figure 1. Functional Block Diagram

PIN DESCRIPTION

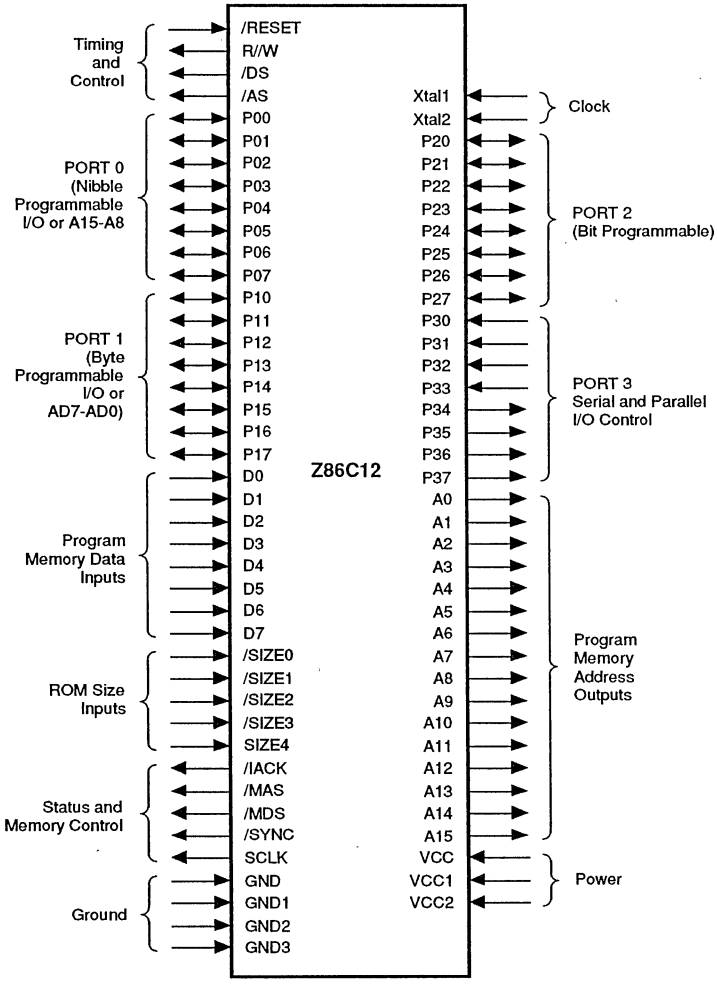


Figure 2. Z86C12 Pin Functions

PIN DESCRIPTION (Continued)

Table 1. Z86C12 Pin Assignments

Name	Pin	Name	Pin	Name	Pin	Name	Pin
/AS	B2	A5	K6	P02	D2	P27	C7
/DS	C4	A6	J6	P03	D1	P30	B4
/MAS	E1	A7	K8	P04	E3	P31	B7
/MDS	G3	A8	J5	P05	G1	P32	C2
/RST	B3	A9	K4	P06	H1	P33	D9
/SIZE0	A3	D0	H3	P07	J1	P34	E10
/SIZE1	C5	D1	K2	P10	G8	P35	B1
/SIZE2	A6	D2	J3	P11	G9	P36	A7
/SIZE3	C6	D3	K3	P12	G10	P37	A5
/SYNC	F1	D4	H8	P13	F8	R/W	A1
A0	J9	D5	J10	P14	D10	SCLK	G2
A1	H7	D6	H9	P15	C10	SIZE4	F10
A10	J4	D7	H10	P16	B10	V_{CC}	A4
A11	H4	/IACK	F2	P17	E9	V_{CC1}	B6
A12	K9	NC	J2	P20	C9	V_{CC2}	F9
A13	K7	NC	C3	P21	A10	GND	F3
A14	K5	NC	D8	P22	B9	GND1	E2
A15	H5	NC	H2	P23	C8	GND2	H6
A2	K10	NC	K1	P24	A9	GND3	E8
A3	J8	P00	C1	P25	B8	XTAL1	B5
A4	J7	P01	D3	P26	A8	XTAL2	A2

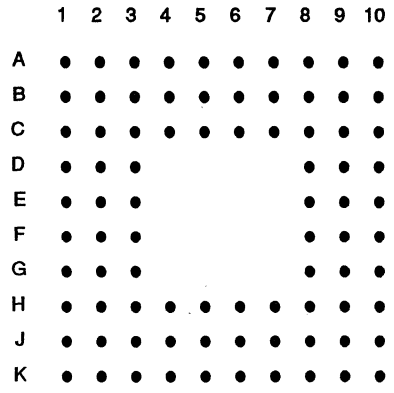


Figure 3. Z86C12 Pin Layout

PIN FUNCTIONS

/DS. (output, active Low). Data Strobe is activated once for each external memory transfer. For a READ operation, data must be available prior to the trailing edge of /DS. For WRITE operations, the falling edge of /DS indicates that output data is valid.

/AS. (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Address output is via Port 1 for all external program. Program or data memory address transfers are valid at the trailing edge of /AS. Under program control, /AS can be placed in the high-impedance state along with Ports 0 and 1, Data Strobe, and Read/Write.

XTAL1, XTAL2. *Crystal 1, Crystal 2* (time-based input and output, respectively). These pins connect a parallel-resonant crystal, ceramic resonator, LC, or any external single-phase clock to the on-chip oscillator and buffer.

R/W. (output, write Low). The Read/Write signal is low when the ICE is writing to external program or data memory.

/RESET. (input, active-Low). To avoid asynchronous and noisy reset problems, the ICE is equipped with a reset filter of four external clocks (4TpC). If the external /RESET signal is less than 4TpC in duration, no reset will occur. On the 5th clock after the /RESET is detected, an internal RST signal is latched and held for an internal register count of 18 external clocks, or for the duration of the external /RESET, whichever is longer. During the reset cycle, /DS is held active low while /AS cycles at a rate of TpC/2. When /RESET is deactivated, program execution begins at location 000C (HEX). Reset time is held low for 50 ms, or until Vcc is stable, whichever is longer.

D7-D0. (Inputs, TTL compatible) Data bus. These eight lines provide the input data bus to access external memory, which is emulating the on-chip ROM. During read cycles in the internal memory space the data on these lines is latched in just prior to the rise of the /MDS data strobe.

A15-A0. (Outputs TTL compatible) Address bus. During T1 these lines output the current memory address. All addresses, whether internal or external, are output.

/MAS. (Output, TTL compatible) Memory Address Strobe. This line is active during every T1 cycle. The rising edge of this signal is used to latch the current memory address on the lines A15 - A0. This line is always valid. It is not tri-stated when /AS is tri-stated.

/MDS. (Output, TTL compatible) Memory Data Strobe. This is a timing signal used to enable the external memory to emulate the on-chip ROM. It is active only during accesses to the on-chip ROM memory space as selected by the configuration of the SIZEn pins.

/SCLK. (Output, TTL compatible) System Clock. This line is the internal system clock.

/SYNC. (Output, TTL compatible) Sync signal. This signal indicates the last clock cycle of the currently executing instruction.

/IACK. (Output, TTL compatible) Interrupt acknowledge. This output, when low, indicates that the ICE is an interrupt cycle.

SIZE0, SIZE1, SIZE2, SIZE3, SIZE4. (Inputs, TTL compatible). The SIZEn lines control the emulation mode of the ICE. The functions are defined as shown in Table 2. The ICE need not be RESET when the state of these lines is changed.

Table 2. Memory Size Configuration

SIZE4	SIZE3	SIZE2	SIZE1	SIZE0	Memory
0	1	1	1	1	ROMless
0	1	1	1	0	2K ROM
0	1	1	0	1	4K ROM
0	1	0	1	1	8K ROM
0	0	1	1	1	16K ROM
1	1	1	1	1	32K ROM

Note: The SIZE pins can be configured to make the memory control signals (/MAS, /MDS, R//W, /AS, and /DS) look like the Z86C91 ROMless device. However, on power-up or reset, Ports 0 and 1 are configured as inputs, rather than A15 - A8 and AD7 - AD0, respectively. This means that if ROMless mode is desired, the device is powered up in ROM mode, and executes a few instructions via the ICE address/data ports. These instructions reconfigure the ports as required, and then the SIZE inputs can be set to ROMless mode - but without a RESET.

PIN FUNCTIONS (Continued)

I/O Ports

Port 0 (P00-P07). Port 0 is an 8-bit, nibble programmable, bidirectional, TTL compatible port. These eight I/O lines can be configured under software control as a nibble I/O port, or as an address port for interfacing external memory. When used as an I/O port, Port 0 may be placed under handshake control. In this configuration, Port 3, lines P32 and P35 are used as the handshake control /DAV0 and RDY0 (Data available and Ready). Handshake signal assignment is dictated by the I/O direction of the upper nibble P04-P07. The lower nibble must have the same direction as the upper nibble to be under handshake control. For the ROMless option, Port 0 appears as A15-A8 Address lines after reset.

For external memory references, Port 0 provides address bit A11-A8 (lower nibble) or A15-A8 (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O, while the lower nibble is used for addressing. If one or both nibbles are needed for I/O operation, they must be configured by writing to the Port 0 Mode register. In ROMless mode, after a hardware reset, Port 0 lines are defined as address lines A15-A8, and extended timing is set to accommodate slow memory access. The initialization routine can include reconfiguration to eliminate this extended timing mode (Figure 4).

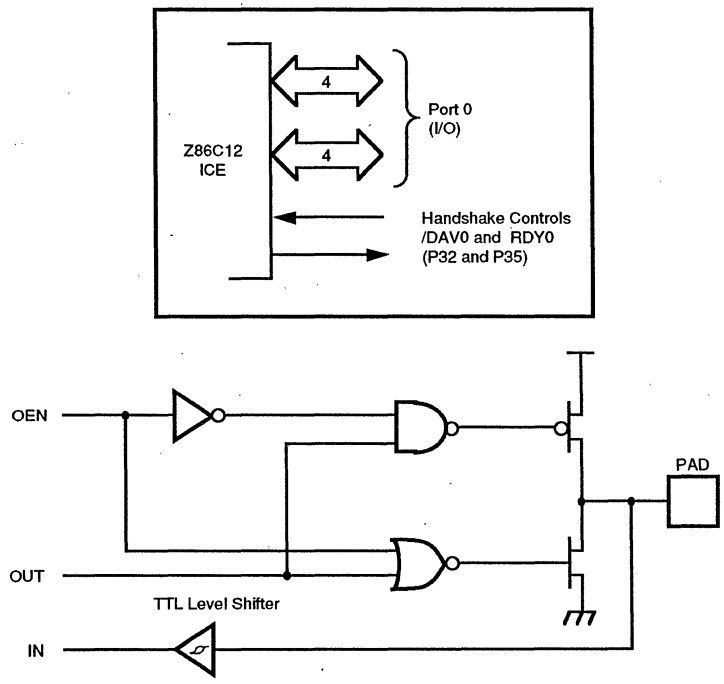


Figure 4. Port 0 Configuration

Port 1 (P10-P17). Port 1 is an 8-bit, byte programmable, bidirectional, TTL compatible port. It has multiplexed Address (A7-A0) and Data (D7-D0)ports. For the ICE, these eight I/O lines can be programmed as Input or Output lines or the port can be configured, under software control, as an address/data port for interfacing external memory. When used as an I/O port, Port 1 can be placed under handshake control. In this configuration, Port 3 lines P33 and P34 are used as the handshake controls RDY1 and /DAV1, respectively.

Memory locations greater than 8192 are referenced through Port 1. To interface external memory, Port 1 is programmed for the multiplexed Address/Data mode. If more than 256 external locations are required, Port 0 outputs the additional lines.

Port 1 can be placed in the high-impedance state along with Port 0, /AS, /DS and R//W, allowing the ICE to share common resource in multiprocessor and DMA applications. Data transfers can be controlled by assigning P33 as a Bus Acknowledge input, and P34 as a Bus request output (Figure 5).

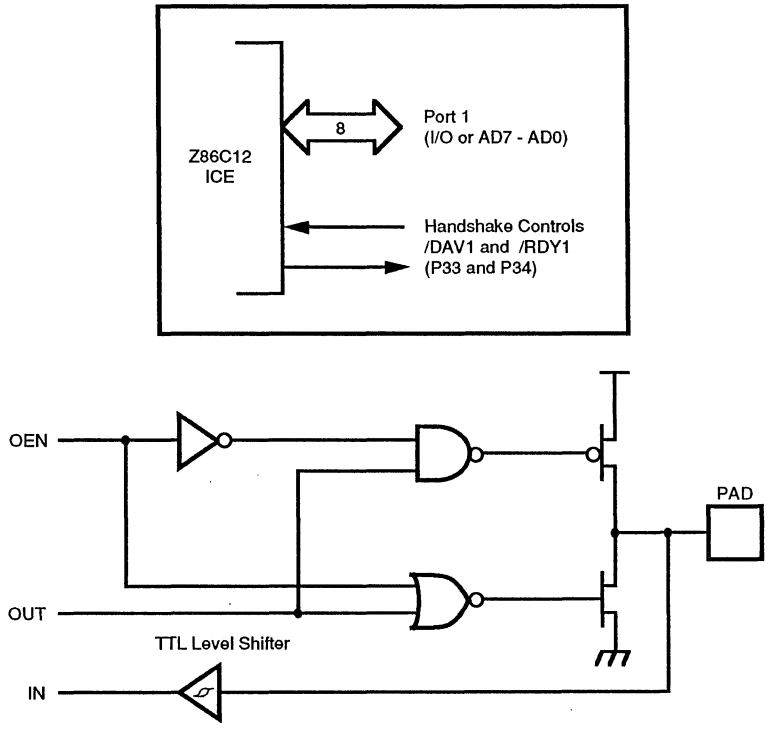


Figure 5. Port 1 Configuration

PIN FUNCTIONS (Continued)

Port 2 (P20-P27). Port 2 is an 8-bit, bit programmable, bidirectional, CMOS compatible port. Each of these eight I/O lines can be independently programmed as an input or output or globally as an open-drain output. Port 2 is always available for I/O operation. When used as an I/O port, Port 2 may be placed under handshake control. In this configuration, Port 3 lines P31 and P36 are used as the handshake controls lines /DAV2 and RDY2. The handshake signal assignment for Port 3 lines, P31 and P36, is dictated by the direction (input or output) assigned to P27 (Figure 6).

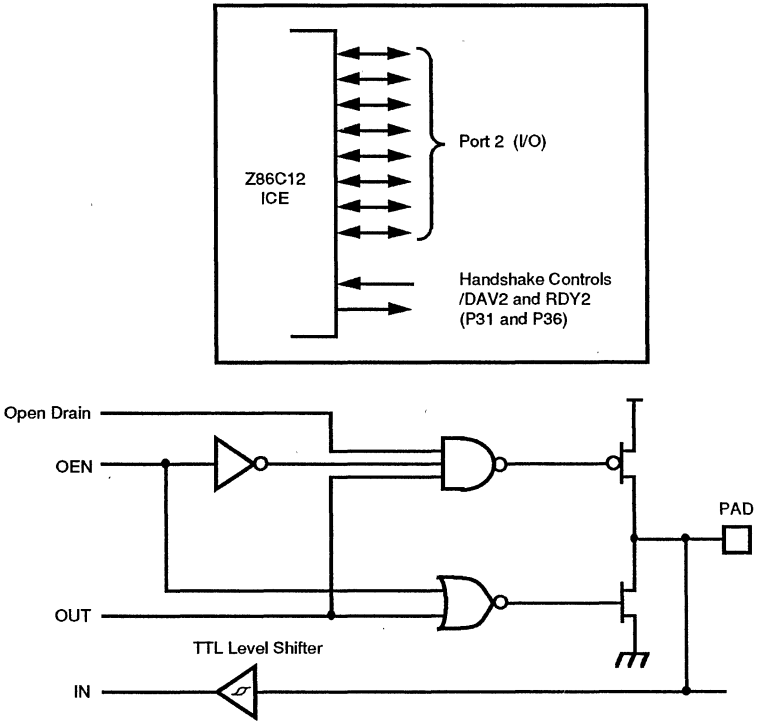


Figure 6. Port 2 Configuration

Port 3 (P30-P37). Port 3 is an 8-bit, CMOS compatible four-fixed-input and four-fixed output port. These eight I/O lines have four-fixed (P30-P33) input and four fixed (P34-P37) output ports. Port 3, when used as serial I/O, is programmed as serial in and serial out, respectively (Figure 7 and Table 3).

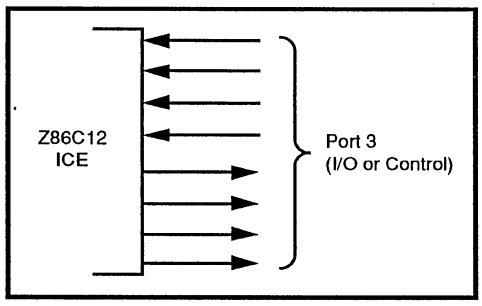


Figure 7. Port 3 Configuration

Port 3 is configured under software control to provide the following control functions: handshake for Ports 0 and 2 (/DAV and RDY); four external interrupt request signals (IRQ0-IRQ3); timer input and output signals (T_{IN} and T_{OUT}); Data Memory Select (/DM).

Table 3. Port 3 Pin Assignments

Pin	I/O	CTC1	Int.	P0 HS	P1 HS	P2 HS	UART	Ext
P30	IN		IRQ3				Serial In	
P31	IN	T_{IN}	IRQ2			D/R		
P32	IN		IRQ0	D/R				
P33	IN		IRQ1		D/R			
P34	OUT				R/D			DM
P35	OUT			R/D				
P36	OUT	T_{OUT}				R/D		
P37	OUT						Serial Out	

Notes:
HS = HANDSHAKE SIGNALS
D = Data Available
R = Ready

PIN FUNCTIONS (Continued)

Port 3 lines P30 and P37, can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by the Counter/Timer 0.

The ICE automatically adds a start bit and two stop bits to transmitted data (Figure 8). Odd parity is also available as an option. Eight data bits are always transmitted, regardless of parity selection. If parity is enabled, the eighth bit is the odd parity bit. An interrupt request (IRQ4) is generated on all transmitted characters.

Received data must have a start bit, eight data bits and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ3 interrupt request.

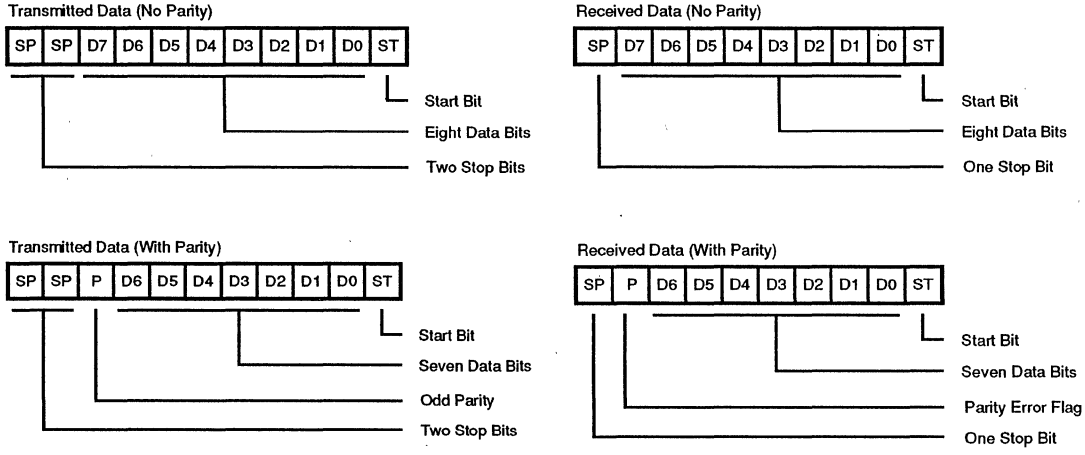


Figure 8. Serial Data Formats

PROGRAMMING

Address Space

Program Memory. The ICE can address up to 64K bytes of external program memory (Figure 9). The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts. The 5 SIZEn inputs dictate the amount of ROM being emulated, and for an 8K ROM the input is '01011'. Respectively, 000C to 8191 is the memory map for the emulated ROM , and 8192 to 65535 is the remaining program memory for which the ICE executes external memory fetches.

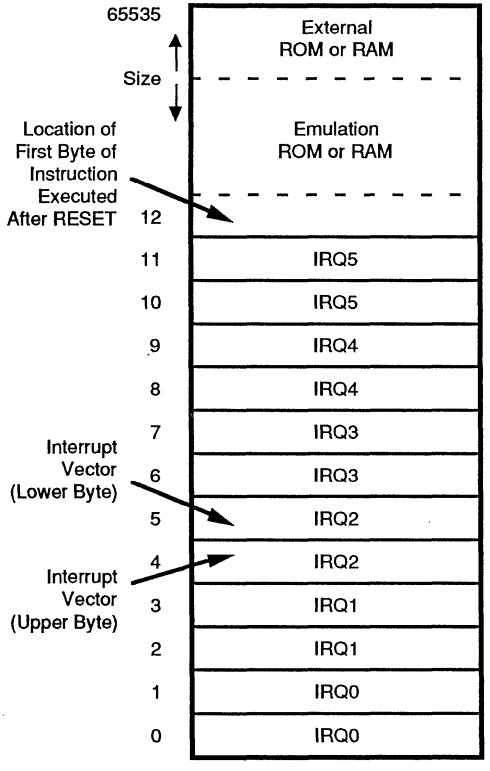


Figure 9. Program Memory Configuration

Data Memory (/DM). External data memory is included with, or separated from, the external program memory space. /DM, an optional I/O function that can be programmed to appear on pin P34, is used to distinguish between data and program memory space (Figure 10). The state of the /DM signal is controlled by the type instruction being executed. An LDC opcode references PROGRAM (/DM inactive) memory, and an LDE instruction references DATA (/DM active low) memory. The lower unaddressable part of the data memory is in fact addressable with the ICE chip's /MDS line (as /DS is not active for internal ROM reads), but there should be no need for this.

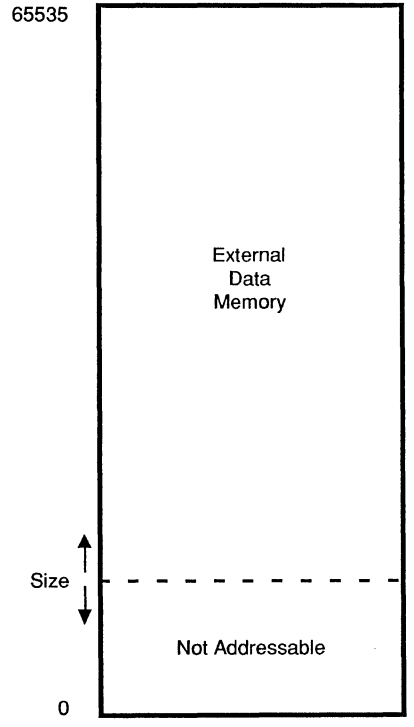


Figure 10. Data Memory Configuration

Register File. The Register File consists of four I/O port registers, 236 general-purpose registers and 16 control and status registers (Figure 11). The instructions can access registers directly or indirectly via an 8-bit address field. The ICE also allows short 4-bit register addressing using the Register Pointer (Figure 12). In the 4-bit mode, the Register File is divided into 16 working register groups, each occupying 16 continuous locations. The Register Pointer addresses the starting location of the active working-register group.

LOCATION		IDENTIFIERS
255	Stack Pointer (Bits 7-0)	SPL
254	Stack Pointer (Bits 15-8)	SPH
253	Register Pointer	RP
252	Program Control Flags	FLAGS
251	Interrupt Mask Register	IMR
250	Interrupt Request Register	IRQ
249	Interrupt Priority Register	IPR
248	Ports 0-1 Mode	P01M
247	Port 3 Mode	P3M
246	Port 2 Mode	P2M
245	T0 Prescaler	PRE0
244	Timer/Counter 0	T0
243	T1 Prescaler	PRE1
242	Timer/Counter 1	T1
241	Timer Mode	TMR
240	Serial I/O	SIO
	Not Implemented	
239	General-Purpose Registers	
4		
3	Port 3	P3
2	Port 2	P2
1	Port1	P1
0	Port 0	P0

Figure 11. Register File

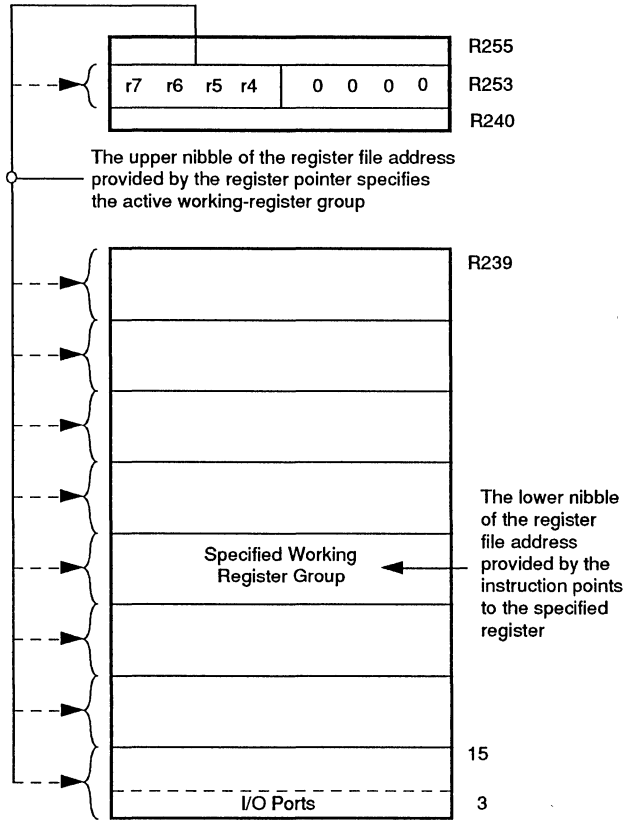


Figure 12. Register Pointer

Stack. The ICE has a 16-bit Stack Pointer (R254-R255) used for an external stack that resides anywhere in the data memory for the ROMless mode, but only from SIZEn to 65535 in ROM mode. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 236 general-purpose registers (R4-R239). The high byte of the Stack Pointer (SPH-Bit 8-15) can be use as a general purpose register when using internal stack only.

FUNCTIONAL DESCRIPTION

Counter/Timers. There are two 8-bit programmable counter/timers (T0-T1), each driven by its own 6-bit programmable prescaler. The T1 prescaler is driven by internal or external clock sources; the T0 prescaler is driven by the internal clock only (Figure 13).

The 6-bit prescalers divide the input frequency of the clock source by any integer number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When both the counters and prescalers reach the end of the count, a timer interrupt request, IRQ4 (T0) or IRQ5 (T1), is generated.

The counter can be programmed to start, stop, restart to continue, or restart from the initial value. The counters can also be programmed to stop upon reaching zero (single pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode).

The counter, but not the prescalers, is read at any time without disturbing its value or count mode. The clock source for T1 is user-definable and can be either the internal microprocessor clock divided-by-four, or an external signal input via Port 3. The Timer Mode register configures the external timer input (P31) as an external clock, a trigger input that is retriggerable or non-retriggerable, or as a gate input for the internal clock. Port 3, line P36, also serves as a timer output (Tout) through which T0, T1 or the internal clock is output. The counter/timers can be cascaded by connecting the T0 output to the input of T1.

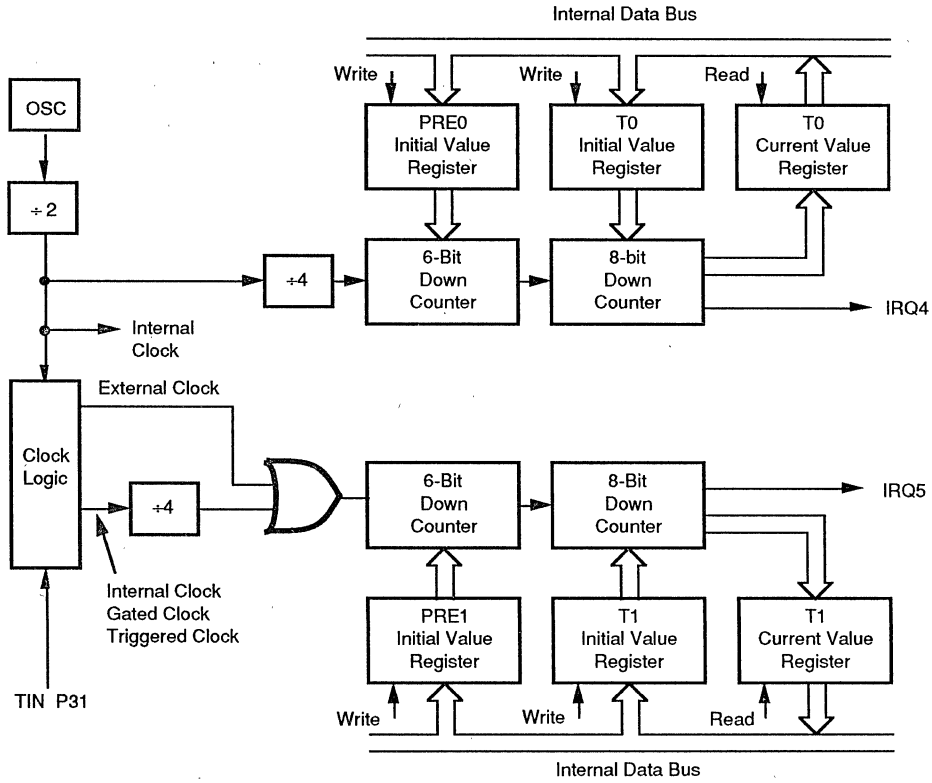


Figure 13. Counter/Timers Block Diagram

Interrupts. The ICE has six different interrupts from eight different sources. The interrupts are maskable and prioritized. The eight sources are divided as follows: four sources are claimed by Port 3 lines P30-P33, one in Serial Out, one in Serial In, and two in the counter/timers (Figure 14). The Interrupt Mask Register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register. All ICE interrupts are vectored through locations in the program memory. When an interrupt machine cycle is activated, an interrupt request is granted. Thus, this disables all of the subsequent interrupts, save the Program Counter and Status Flags, and then branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

To accommodate polled interrupt systems, interrupt inputs are masked and the Interrupt Request register is polled to determine which of the interrupt requests need service. Software initialed interrupts are supported by setting the appropriate bit in the Interrupt Request Register (IRQ).

Internal interrupt requests are sampled on the falling edge of the last cycle of every instruction, and the interrupt request is valid 5TpC before the falling edge of the last clock cycle of the currently executing instruction.

For the ROMless mode, when the device samples a valid interrupt request, the next 48 (external) clock cycles are used to prioritize the interrupt, and push the two PC bytes and the FLAG register on the stack. The following nine cycles are used to fetch the interrupt vector from external memory. The first byte of the interrupt service routine is fetched beginning on the 58th TpC cycle following the internal sample point. This corresponds to the 63rd TpC cycle following the external interrupt sample point.

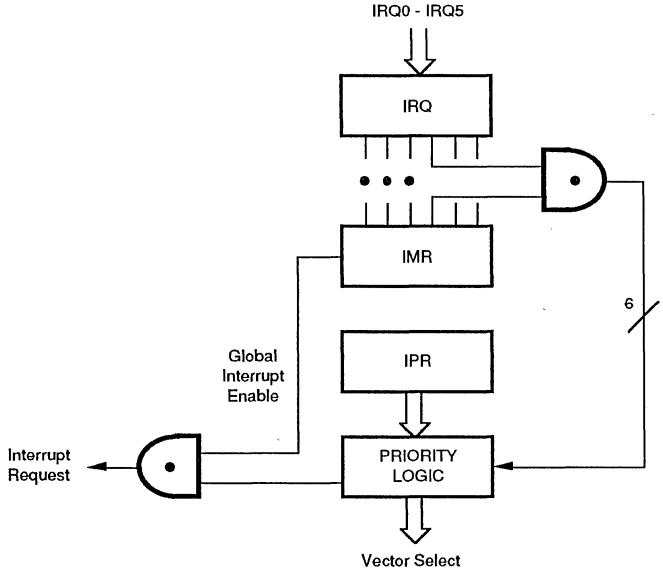


Figure 14. Interrupt Block Diagram

Clock. The ICE on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal, LC, ceramic resonator, or any suitable external clock source (XTAL1=Input, XTAL2=Output). The crystal should be AT cut, 1 MHz to 16 MHz max, and series resistance (RS) is less than or equal to 100 Ohms. The crystal should be connected across XTAL1 and XTAL2 using the recommended capacitors (10pF < CL < 100pF) from each pin to ground (Figure 15).

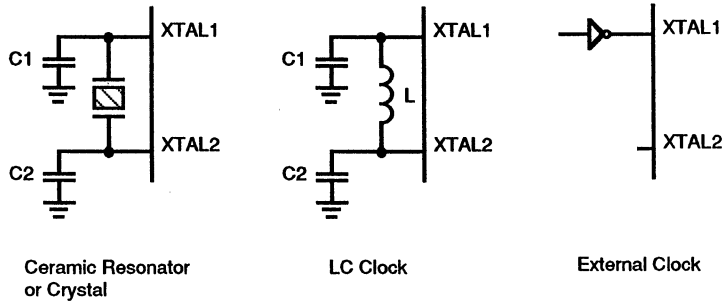


Figure 15. Oscillator Configuration

HALT. This turns off the internal CPU clock but not the XTAL oscillation. The counter/timers and the external interrupts IRQ0, IRQ1, IRQ2 and IRQ3 remain active. The devices are recovered by interrupts, either externally or internally generated.

STOP. This instruction turns off the internal clock and external crystal oscillation and reduces the standby current to 10 microamperes or less. The Stop mode is terminated by a reset, which causes the processor to restart the application program at address 000C (HEX).

To enter STOP (or HALT) mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. To do this, the user must execute a NOP (opcode=OFFH) immediately before the appropriate sleep instruction. i.e.:

```
FF  NOP   ; clear the pipeline
6F  STOP  ; enter STOP mode
          or
FF  NOP   ; clear the pipeline
7F  HALT  ; enter HALT mode
```

Instruction Cycle Timing

Figures 16 and 17 show instruction cycle timing for instructions fetched from external memory.

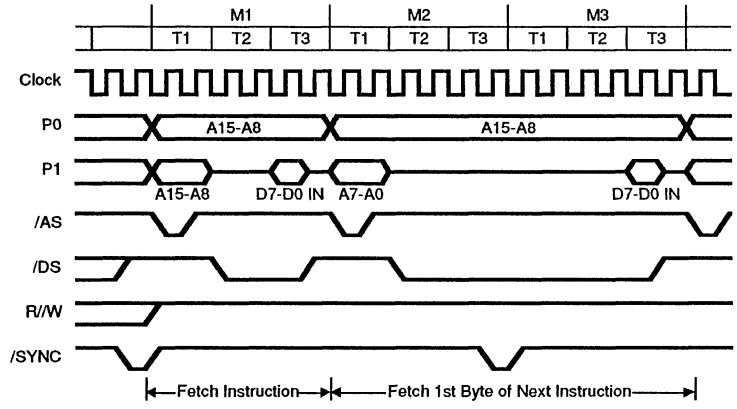


Figure 16. Instruction Cycle Timing (One-Byte Instructions)

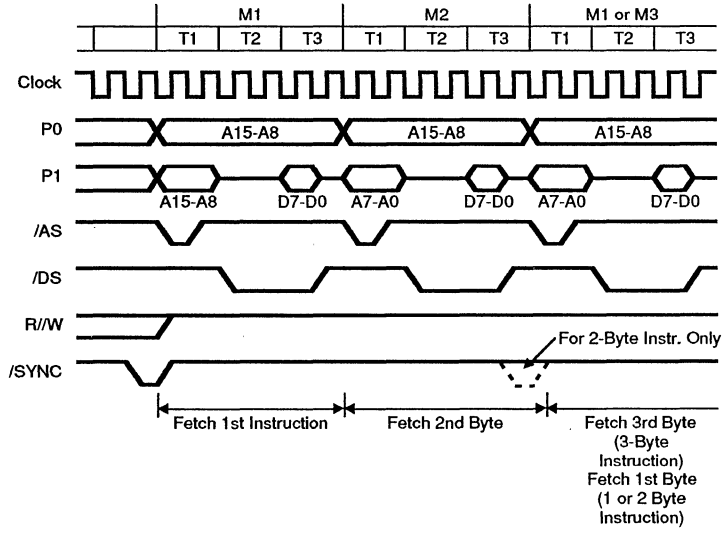


Figure 17. Instruction Cycle Timing (Two- and Three-Byte Instructions)

The addresses, Address Strobe (/AS) and Read Write (R//W) are output at the beginning of each machine cycle (Mn). The addresses output via Port 0 (if used) remain stable throughout the machine cycle. Addresses output via Port 1 remain valid only during MnT1. The addresses are guaranteed valid at the rising edge of /AS, which is used to latch the Port 1 output. Port 1 is placed in an input mode at the end of MnT1. The Data Strobe is output during MnT2 allowing data to be placed on the Port 1 bus. The Z8 accepts the data during MnT3 and /DS is terminated.

Instruction synchronization pulse /SYNC is output one clock pulse period prior to the beginning of an opcode fetch machine cycle (M1). This output is directly available on the 64-pin version of the Z8; whereas, on the 40-pin version, the Data Strobe pin outputs /SYNC only if external memory is not used.

Note that all instruction fetch cycles have the same machine timing regardless of whether the memory is internal or not. If configured for external memory, and internal memory is referenced, the addresses are still output via Ports 0 and 1; /DS and R//W are inactive. If configured for internal memory only, Ports 0 and 1 are used for I/O,/DS outputs, /SYNC; R//W is inactive.

The exception to the instruction fetch timing is during the opcode fetch of an instruction following the fetch of a one byte instruction. One-byte instructions require two machine cycles to execute. The pipelining causes the following opcode fetch to begin one machine cycle early.

External Memory or I/O Timing

When external memory is addressed, Ports 0 and 1 are configured to output the required number of address bits. Port 1 is used as a multiplexed address/data bus for AD7-AD0 and Port 0 outputs address bits A15-A8. The timing relationships for addressing external memory and I/O are illustrated in Figures 18, 19, 20 and 21. The main difference between these figures is that Figures 20 and 21 contain an added timing cycle (Tx) that extends external memory timing to allow for slower memory.

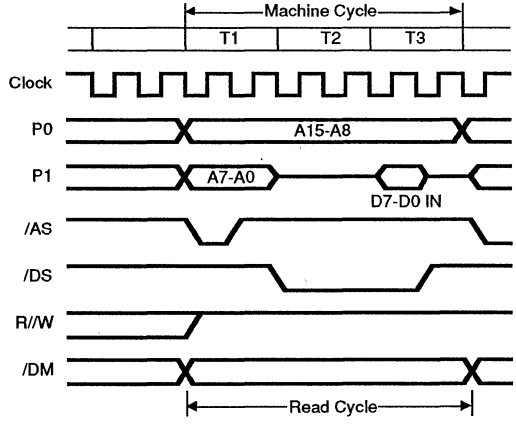


Figure 18. External Instruction Fetch, I/O or Memory Read Cycle

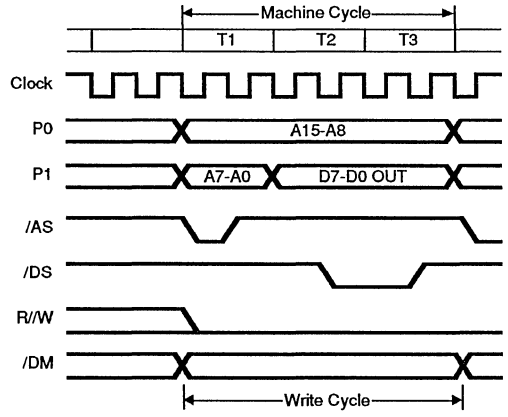


Figure 19. External I/O or Memory Write Cycle

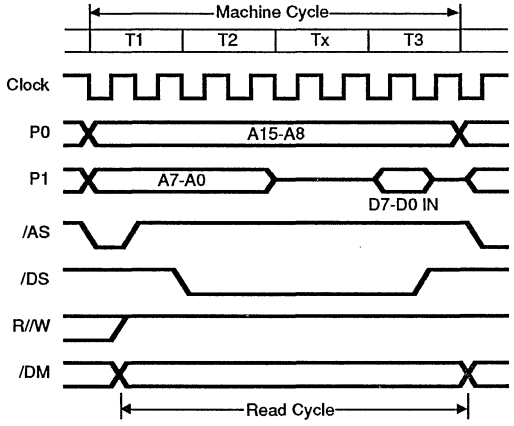


Figure 20. Extended External Instruction Fetch, I/O or Memory Read Cycle

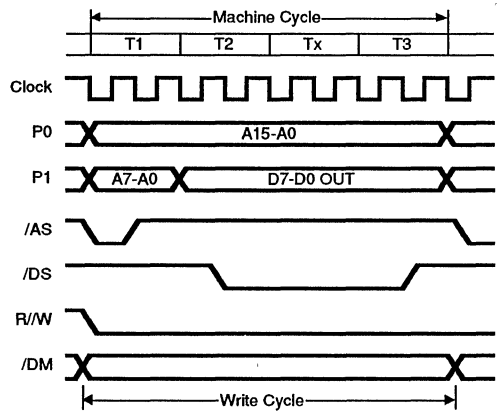


Figure 21. Extended External I/O or Memory Write Cycle

Address bits A15-A0 are valid on Ports 0 and 1 at the trailing edge of /AS for both the read and write memory cycles. Because Port 0 is not multiplexed, address bits A15-A8, if used, are present all through the read/write memory cycle.

During the read cycle, the input data must be valid on Port 1 at the trailing edge of the Data Strobe output (/DS). The Data Memory Select output (/DM) is used to select external data memory or external program memory. If selected, /DM is active during the execution of certain instructions.

During the write cycle, the address outputs follow the same timing relationships as for the read cycle. However, the output data is valid for the entire period /DS is active, and R/W is active (Low) during the entire write cycle.

Interrupt requests are sampled before each instruction fetch cycle (Figure 22). First, external interrupt requests are sampled four clock periods prior to the active /AS pulse that corresponds to an instruction fetch cycle. Then, internal interrupt requests are samples one clock period preceding /AS.

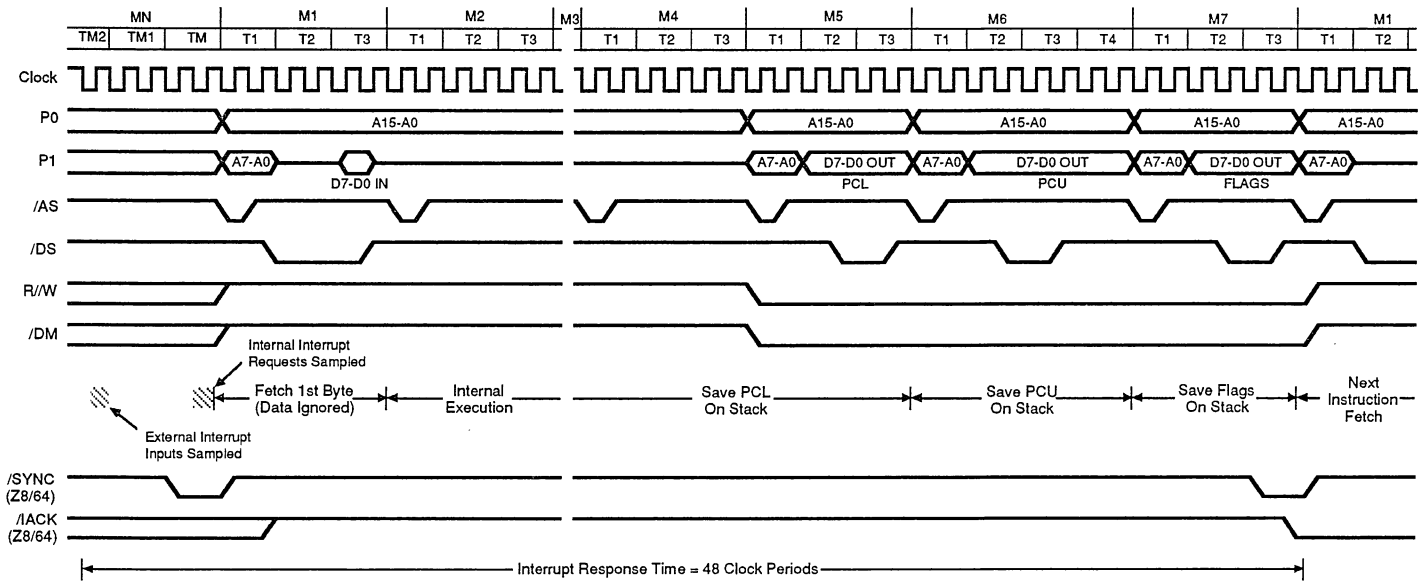


Figure 22. Interrupt Cycle Timing

If an interrupt request is set, the Z8 spends seven machine cycles (44 clock periods) resolving interrupt priorities, selecting the proper interrupt vector, and saving the program counter and flags on the stack. Although Figure 13 illustrates the timing for an external stack, the same timing is used for an internal stack. The total interrupt response time (including the external interrupt sample time) for an external interrupt is 48 clock periods. The first instruction of the interrupt service routine is fetched at this time. When an interrupt request is detected in the Z8/64 development device, /IACK is activated (Low) and remains active until the first instruction of the interrupt service routine is fetched.

Reset Timing

The internal logic is initialized during reset if the Reset input is held low for at least 18 clock periods (Figure 23). During the time /RESET is Low, /AS is output at the internal clock rate, /DS is forced Low, R//W is inactive and Ports 0, 1 and 2 are placed in an input mode. /AS and /DS both low is normally a mutually exclusive condition; therefore, the coincidence of /AS Low and /DS Low can be used as a reset condition for other devices. Zilog Z-Bus® peripherals take advantage of this reset condition.

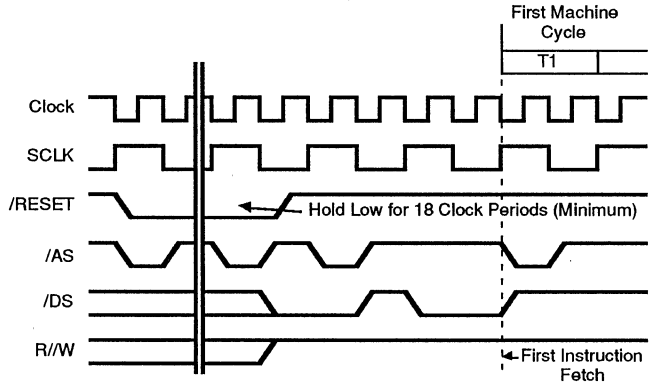


Figure 23. Reset Cycle Timing

Alternative Control Signal Uses

In addition to their uses in memory transfers, the control signals /AS, /DS and R//W can be used in the following interface applications:

/AS can be modified to provide the /RAS (Row Address Strobe) signal for dynamic memory interface. /RAS can be derived from the trailing edge of /DS to the trailing edge of /AS.

/DS has several alternative uses: as a /CAS (Column Address Strobe) for dynamic memory interface; as a Chip Enable for memory and other interface devices; as an Enable input for 3-state bus drivers/receivers for memory and interface devices.

R//W can be used as a Write input to memory interfaces, and as an Early Status output to switch the direction of 3-state bus drivers/receivers.

ABSOLUTE MAXIMUM RATINGS

Symbol	Description	Min	Max	Units
V_{CC}	Supply Voltage*	-0.3	+7.0	V
T_{STG}	Storage Temp	-65	+150	°C
T_A	Oper Ambient Temp**		C	

Notes:
* Voltages on all pins with respect to GND.
** See Ordering Information

Stress greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for an extended period may affect device reliability.

STANDARD TEST CONDITIONS

The characteristics listed below apply for standard test conditions as noted. All voltages are referenced to GND. Positive current flows into the referenced pin (Figure 24).

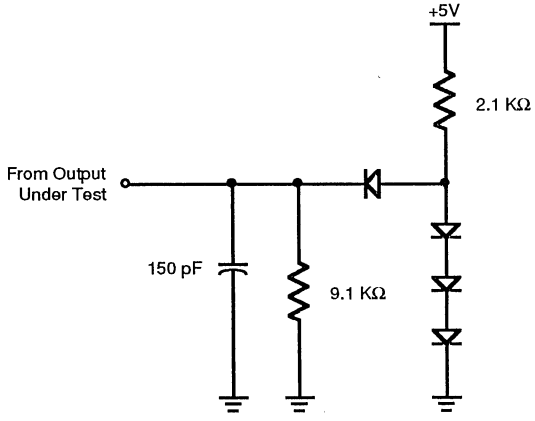


Figure 24. Test Load Diagram

DC CHARACTERISTICS

Sym	Parameter	T_A = 0°C to 70°C Min	Max	T_A = -40°C to 105°C Min	Max	Typical at 25°C	Units	Conditions
	Max Input Voltage		7		7		V	I_{IN} 250 µA
V_{CH}	Clock Input High Voltage	3.8	V_{CC}	3.8	V_{CC}		V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.03	0.8	-0.03	0.8		V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}	2.0	V_{CC}		V	
V_{IL}	Input Low Voltage	-0.3	0.8	-0.3	0.8		V	
V_{OH}	Output High Voltage	2.4		2.4			V	I_{OH} = -250 µA
V_{OL}	Output Low Voltage		0.4		0.4		V	I_{OL} = +2.0 mA
V_{RH}	Reset Input High Voltage	3.8	V_{CC}	3.8	V_{CC}		V	
V_{RI}	Reset Input Low Voltage	-0.03	0.8	-0.03	0.8		V	
I_{IL}	Input Leakage	-1	1	-10	10		µA	0V V_{IN} +5.25V
I_{OL}	Output Leakage	-1	1	-10	10		µA	0V V_{IN} +5.25V
I_{IR}	Reset Input Current		-80		-50		µA	V_{CC}= +5.25V, V_{RL} = 0V
I_{CC}	Supply Current		50		50	25	mA	@ 12 MHz
			60		60	35	mA	@ 16 MHz
I_{CC1}	Standby Current		15		15	5	mA	HALT Mode V_{IN} = 0V, V_{CC}@ 12 MHz
			20		20	10	mA	HALT Mode V_{IN} = 0V, V_{CC}@ 16 MHz
I_{CC2}	Standby Current		10		10	5	µA	STOP Mode V_{IN} = 0V, V_{CC}@ 12 MHz
			10		10	5	µA	STOP Mode V_{IN} = 0V, V_{CC}@ 16 MHz

Notes:

I_{CC2} requires loading TMR (%F1H) with any value prior to STOP execution.
Use this sequence:

```
LD TMR,#00
NOP
STOP
```

AC CHARACTERISTICS

External I/O or Memory Read or Write Timing Diagram

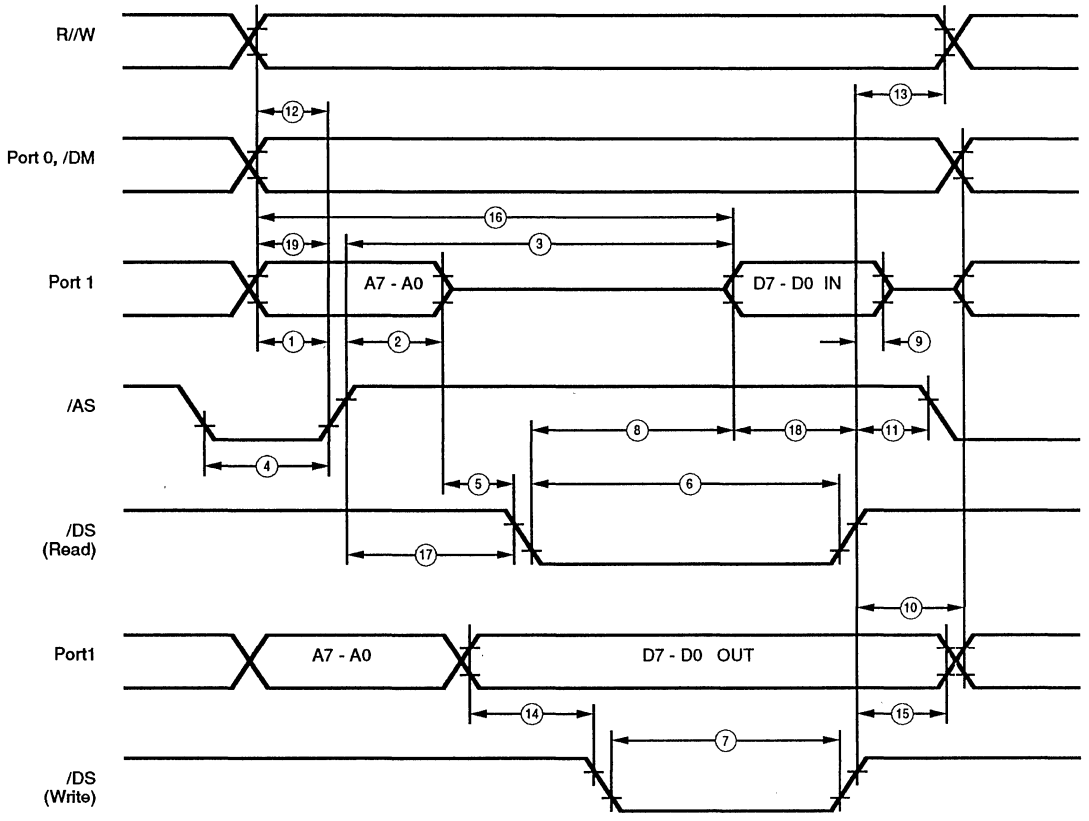


Figure 25. External I/O or Memory Read or Write Timing

AC CHARACTERISTICS
External I/O or Memory Read and Write Timing Table

No	Symbol	Parameter	T_A = 0°C to 70°C				T_A = -40°C to 105°C				Units	Notes
			12 MHz		16 MHz		12 MHz		16 MHz			
			Min	Max	Min	Max	Min	Max	Min	Max		
1	TdA(AS)	Address Valid to /AS Rise Delay	35		20		35		25		ns	[2,3]
2	TdAS(A)	/AS Rise to Address Float Delay	45		30		45		35		ns	[2,3]
3	TdAS(DR)	/AS Rise to Read Data Req'd Valid		220		180		250		180	ns	[1,2,3]
4	TwAS	/AS Low Width	55		35		55		40		ns	[2,3]
5	TdAZ(DS)	Address Float to /DS Fall	0		0		0		0		ns	
6	TwDSR	/DS (Read) Low Width	185		135		185		135		ns	[1,2,3]
7	TwDSW	/DS (Write) Low Width	110		80		110		80		ns	[1,2,3]
8	TdDSR(DR)	/DS Fall to Read Data Req'd Valid		130		75		130		75	ns	[1,2,3]
9	ThDR(DS)	Read Data to /DS Rise Hold Time	0		0		0		0		ns	[2,3]
10	TdDS(A)	/DS Rise to Address Active Delay	45		35		65		50		ns	[2,3]
11	TdDS(AS)	/DS Rise to /AS Fall Delay	55		30		45		35		ns	[2,3]
12	TdR/W(AS)	R/W Valid to /AS Rise Delay	30		20		33		25		ns	[2,3]
13	TdDS(R/W)	/DS Rise to R/W Not Valid	35		30		50		35		ns	[2,3]
14	TdDW(DSW)	Write Data Valid to /DS Fall (Write) Delay	35		25		35		25		ns	[2,3]
15	TdDS(DW)	/DS Rise to Write Data Not Valid Delay	35		30		55		35		ns	[2,3]
16	TdA(DR)	Address Valid to Read Data Req'd Valid		255		200		310		230	ns	[1,2,3]
17	TdAS(DS)	/AS Rise to /DS Fall Delay	55		40		65		45		ns	[2,3]
18	TdDI(DS)	Data Input Setup to /DS Rise	75		60		75		60		ns	[1,2,3]
19	TdDM(AS)	/DM Valid to /AS Fall Delay	50		30		50		30		ns	[2,3]

Notes:
[1] When using extended memory timing add 2 TpC.
[2] Timing numbers given are for minimum TpC.
[3] See clock cycle dependent characteristics table.

Standard Test Load
All timing references use 2.0V for a logic 1 and 0.8V for a logic 0.

Clock Dependent Formulas

Number	Symbol	Equation
1	TdA(AS)	0.40TpC + 0.32
2	TdAS(A)	0.59TpC - 3.25
3	TdAS(DR)	2.38TpC + 6.14
4	TwAS	0.66TpC - 1.65
6	TwDSR	2.33TpC - 10.56
7	TwDSW	1.27TpC + 1.67
8	TdDSR(DR)	1.97TpC - 42.5
10	TdDS(A)	0.8TpC

Number	Symbol	Equation
11	TdDS(AS)	0.59TpC - 3.14
12	TdR/W(AS)	0.4TpC
13	TdDS(R/W)	0.8TpC - 15
14	TdDW(DSW)	0.4TpC
15	TdDS(DW)	0.88TpC - 19
16	TdA(DR)	4TpC - 20
17	TdAS(DS)	0.91TpC - 10.7
18	TsDI(DS)	0.8TpC - 10
19	TdDM(AS)	0.9TpC - 26.3

AC CHARACTERISTICS
Additional Timing Diagram

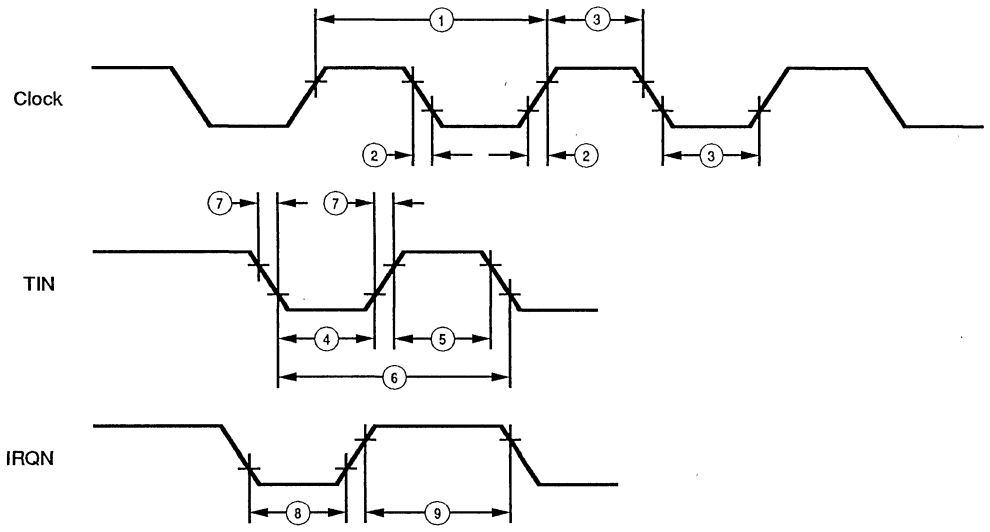


Figure 26. Additional Timing

AC CHARACTERISTICS
Additional Timing Table

No	Symbol	Parameter	T_A = 0°C to 70°C				T_A = -40°C to 105°C				Units	Notes
			12 MHz		16 MHz		12 MHz		16 MHz			
			Min	Max	Min	Max	Min	Max	Min	Max		
1	TpC	Input Clock Period	83	1000	62.5	1000	83	1000	62.5	1000	ns	[1]
2	TrC,TfC	Clock Input Rise & Fall Times		15		10		15		10	ns	[1]
3	TwC	Input Clock Width	37		21		37		21		ns	[1]
4	TwTinL	Timer Input Low Width	70		50		70		50		ns	[2]
5	TwTinH	Timer Input High Width	3TpC		3TpC		3TpC		3TpC			[2]
6	TpTin	Timer Input Period	8TpC		8TpC		8TpC		8TpC			[2]
7	TrTin,TfTin	Timer Input Rise & Fall Times	100		100		100		100		ns	[2]
8A	TwIL	Interrupt Request Input Low Times	70		50		70		50		ns	[2,4]
8B	TwIL	Interrupt Request Input Low Times	3TpC		3TpC		3TpC		3TpC			[2,5]
9	TwIH	Interrupt Request Input High Times	3TpC		3TpC		3TpC		3TpC			[2,3]

Notes:
[1] Clock timing references use 3.8V for a logic 1 and 0.8V for a logic 0.
[2] Timing references use 2.0V for a logic 1 and 0.8V for a logic 0.
[3] Interrupt references request via Port 3.
[4] Interrupt request via Port 3 (P31-P33)'.
[5] Interrupt request via Port 30.

AC CHARACTERISTICS
Handshake Timing Diagram

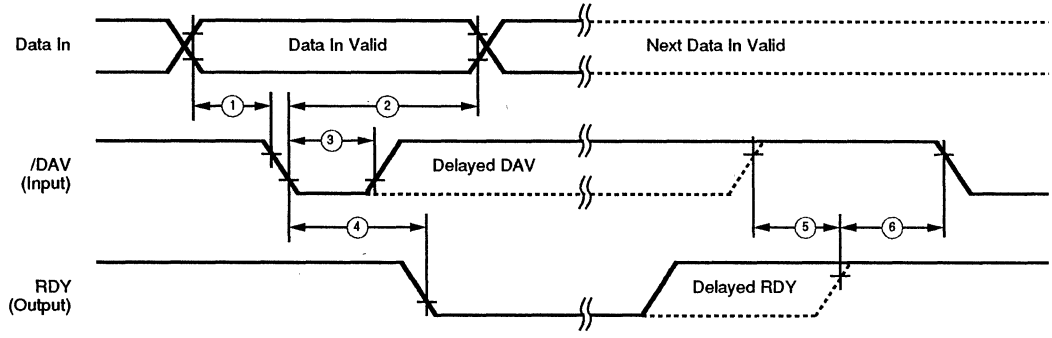


Figure 27. Input Handshake Timing

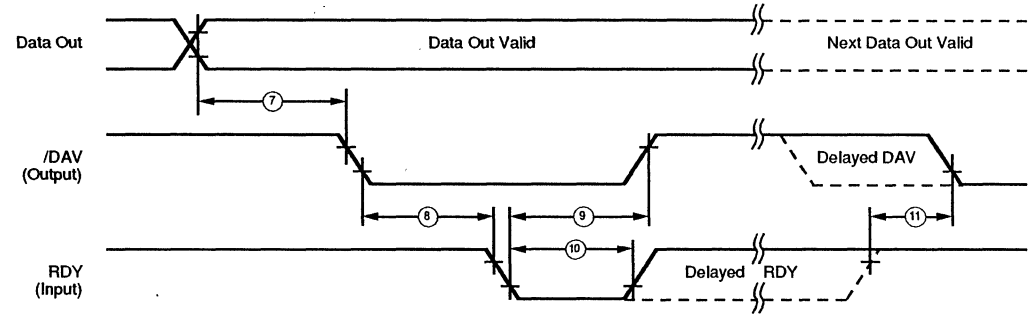


Figure 28. Output Handshake Timing

AC CHARACTERISTICS
Handshake Timing Table

No	Symbol	Parameter	T_A = 0°C to 70°C 12 MHz Min	12 MHz Max	16 MHz Min	16 MHz Max	T_A = -40°C to 105°C 12 MHz Min	12 MHz Max	16 MHz Min	16 MHz Max	Notes (Data Direction)
1	TsDI(DAV)	Data In Setup Time	0		0		0		0		IN
2	ThDI(DAV)	Data In Hold Time	145		145		145		145		IN
3	TwDAV	Data Available Width	110		110		110		110		IN
4	TdDAVI(RDY)	DAV Fall to RDY Fall Delay		115		115		115		115	IN
5	TdDAVId(RDY)	DAV Rise to RDY Rise Delay		115		115		115		115	IN
6	TdDO(DAV)	RDY Rise to DAV Fall Delay	0		0		0		0		IN
7	TcLDAV0(RDY)	Data Out to DAV Fall Delay		TpC		TpC		TpC		TpC	OUT
8	TcLDAV0(RDY)	DAV Fall to RDY Fall Delay	0		0		0		0		OUT
9	TdRDY0(DAV)	RDY Fall to DAV Rise Delay		115		115		115		115	OUT
10	TwRDY	RDY Width	110		110		110		110		OUT
11	TdRDY0d(DAV)	RDY Rise to DAV Fall Delay		115		115		115		115	OUT

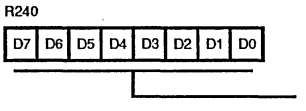


Figure 29. Serial I/O Register (F0H: Read/Write)

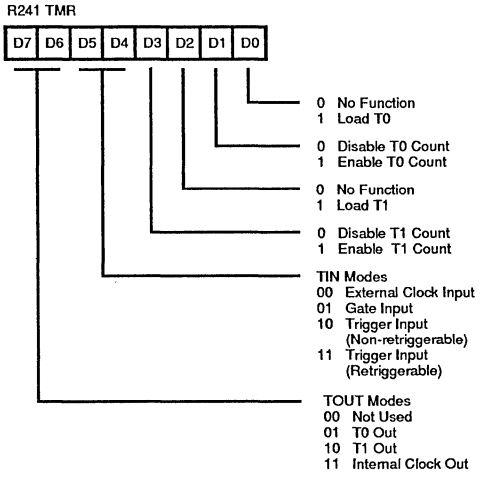


Figure 30. Timer Mode Register (F1H: Read/Write)

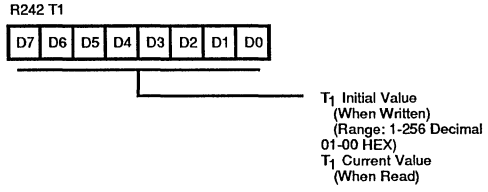


Figure 31. Counter/Timer 1 Register (F2H: Read/Write)

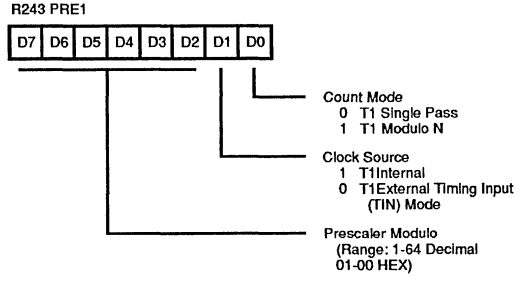


Figure 32. Prescaler 1 Register (F3H: Write Only)

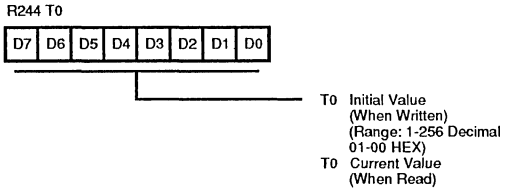


Figure 33. Counter/Timer 0 Register (F4H: Read/Write)

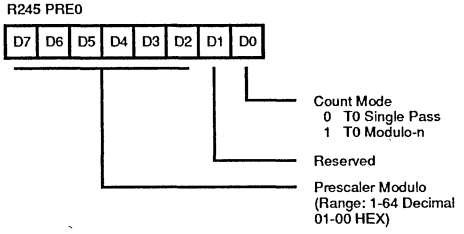


Figure 34. Prescaler 0 Register (F5H: Write Only)

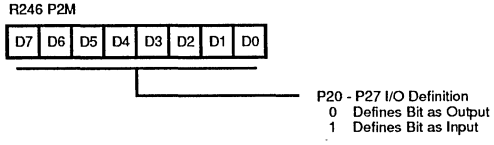


Figure 35. Port 2 Mode Register (F6H: Write Only)

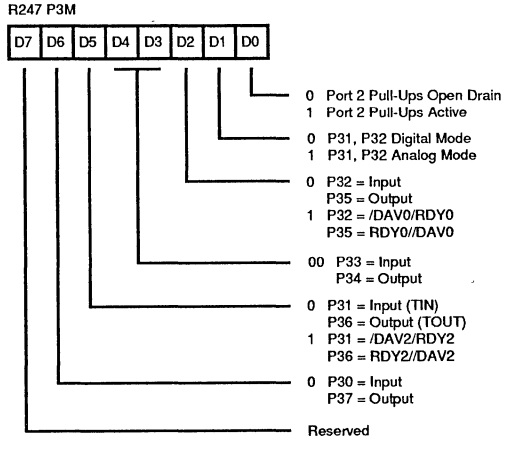


Figure 36. Port 3 Mode Register (F7H: Write Only)

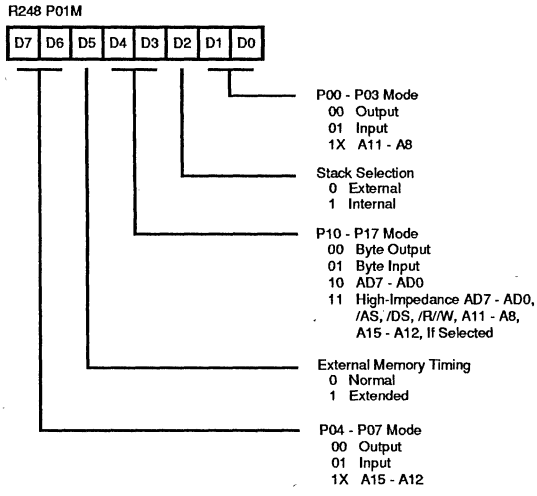


Figure 37. Ports 0 and 1 Mode Register (F8H: Write Only)

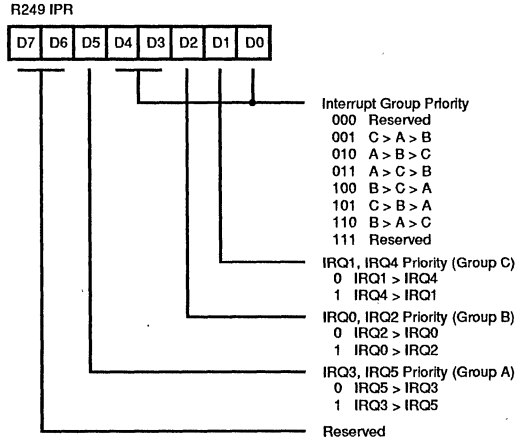


Figure 38. Interrupt Priority Registe (F9H: Write Only)

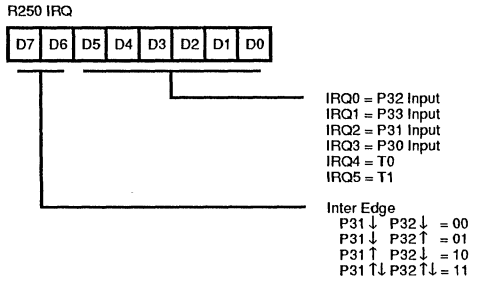


Figure 39. Interrupt Request Register (FAH: Read/Write)

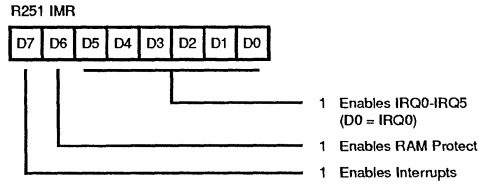


Figure 40. Interrupt Mask Register (FBH: Read/Write)

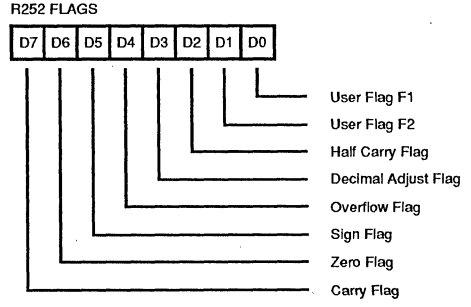


Figure 41. Flag Register (FCH: Read/Write)

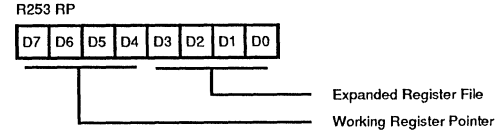


Figure 42. Register Pointer Register (FDH: Read/Write)

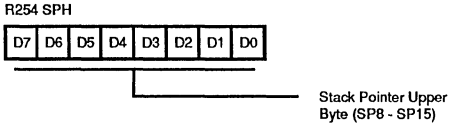


Figure 43. Stack Pointer Register (FEH: Read/Write)

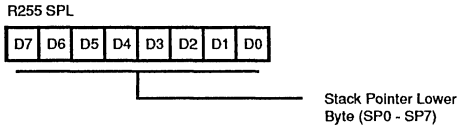


Figure 44. Stack Pointer Register (FFH: Read/Write)

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

Symbol	Meaning
IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

Symbol	Meaning
dst	Destination location or contents
src	Source location or contents
cc	Condition code
@	Indirect address prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag register (Control Register 252)
RP	Register Pointer (R253)
IMR	Interrupt mask register (R251)

Flags. Control register (R252) contains the following six flags:

Symbol	Meaning
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

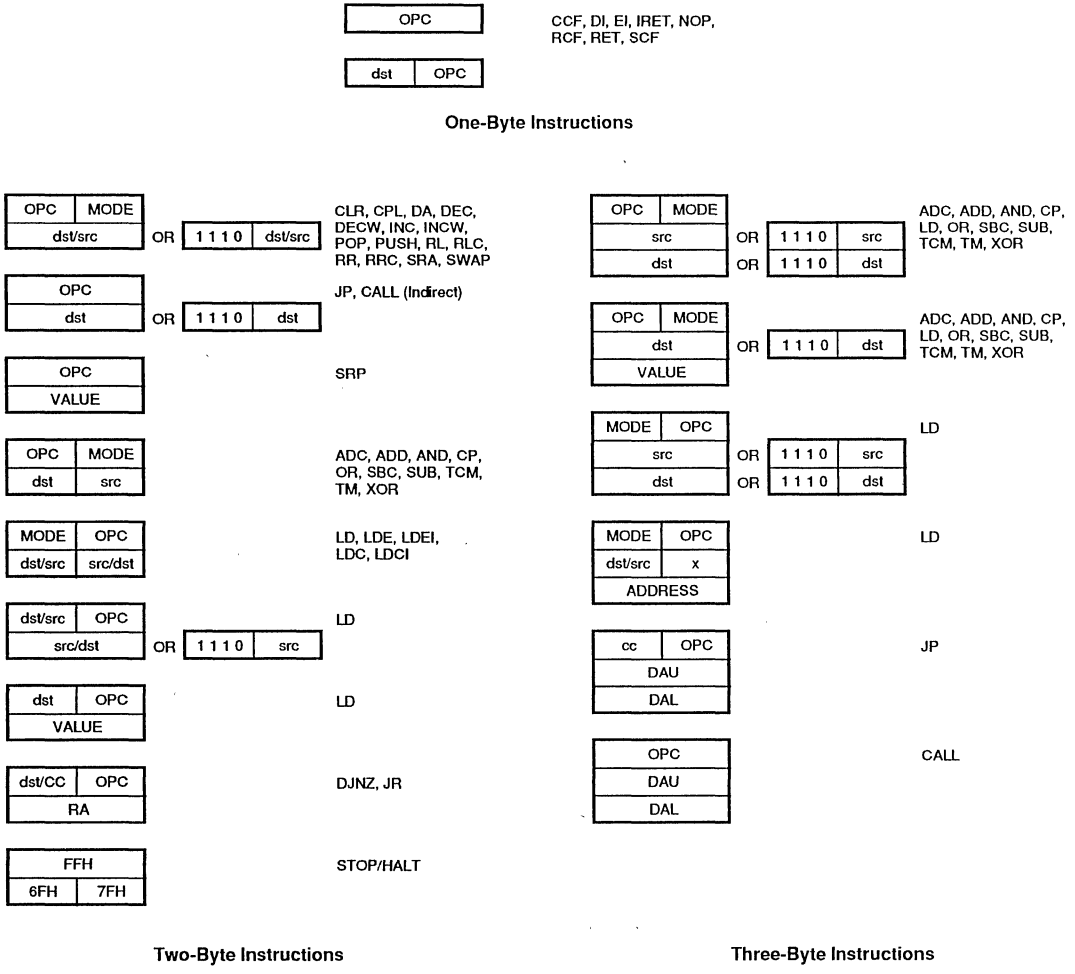
Affected flages are indicated by:

0	Clear to zero
1	Set to one
*	Set to clear according to operation
-	Unaffected
x	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000		Always True	
0111	C	Carry	C = 1
1111	NC	No Carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not Zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No Overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not Equal	Z = 0
1001	GE	Greater Than or Equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater Than	[Z OR (S XOR V)] = 0
0010	LE	Less Than or Equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned Greater Than or Equal	C = 0
0111	ULT	Unsigned Less Than	C = 1
1011	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned Less Than or Equal	(C OR Z) = 1
0000		Never True	

INSTRUCTION FORMATS



INSTRUCTION SUMMARY

Note: Assignment of a value is indicated by the symbol " ← ". For example:

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr (n)" is used to refer to bit (n) of a given operand location. For example:

dst (7)

refers to bit 7 of the destination operand.

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst, src dst←dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst←dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst←dst AND src	†		5[ ]	-	*	*	0	-	-
CALL dst SP←SP - 2 @SP←PC, PC←dst	DA IRR		D6 D4	-	-	-	-	-	-
CCF C←NOT C			EF	*	-	-	-	-	-
CLR dst dst←0	R IR		B0 B1	-	-	-	-	-	-
COM dst dst←NOT dst	R IR		60 61	-	*	*	0	-	-
CP dst, src dst - src	†		A[ ]	*	*	*	*	-	-
DA dst dst←DA dst	R IR		40 41	*	*	*	X	-	-
DEC dst dst←dst - 1	R IR		00 01	-	*	*	*	-	-
DECW dst dst←dst - 1	RR IR		80 81	-	*	*	*	-	-
DI IMR(7)←0			8F	-	-	-	-	-	-
DJNZr, dst r←r - 1 if r ≠ 0 PC←PC + dst Range: +127, -128	RA		rA r = 0 - F	-	-	-	-	-	-
EI IMR(7)←1			9F	-	-	-	-	-	-
HALT			7F	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
INC dst dst←dst + 1	r R IR		rE r = 0 - F 20 21	-	*	*	*	-	-
INCW dst dst←dst + 1	RR IR		A0 A1	-	*	*	*	-	-
IRET FLAGS←@SP; SP←SP + 1 PC←@SP; SP←SP + 2; IMR(7)←1			BF	*	*	*	*	*	*
JP cc, dst if cc is true PC←dst	DA IRR		cD c = 0 - F 30	-	-	-	-	-	-
JR cc, dst if cc is true, PC←PC + dst Range: +127, -128	RA		cB c = 0 - F	-	-	-	-	-	-
LD dst, src dst←src	r r R r X r Ir R R R IR IR	Im R r X r Ir r R IR IM IM R	rC r8 r9 r = 0 - F C7 D7 E3 F3 E4 E5 E6 E7 F5	-	-	-	-	-	-
LDC dst, src	r	Irr	C2	-	-	-	-	-	-
LDCI dst, src dst←src r←r +1; rr←rr + 1	Ir	Irr	C3	-	-	-	-	-	-

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
NOP			FF	-	-	-	-	-	-
OR dst, src dst←dst OR src	†		4[ ]	-	*	*	0	-	-
POP dst dst←@SP; SP←SP + 1	R IR		50 51	-	-	-	-	-	-
PUSH src SP←SP - 1; @SP←src		R IR	70 71	-	-	-	-	-	-
RCF C←0			CF	0	-	-	-	-	-
RET PC←@SP; SP←SP + 2			AF	-	-	-	-	-	-
RL dst	R IR		90 91	*	*	*	*	-	-
RLC dst	R IR		10 11	*	*	*	*	-	-
RR dst	R IR		E0 E1	*	*	*	*	-	-
RRC dst	R IR		C0 C1	*	*	*	*	-	-
SBC dst, src dst←dst←src←C	†		3[ ]	*	*	*	*	1	*
SCF C←1			DF	1	-	-	-	-	-
SRA dst	R IR		D0 D1	*	*	*	0	-	-
SRP src RP←src		Im	31	-	-	-	-	-	-
STOP			6F	-	-	-	-	-	-
SUB dst, src dst←dst←src	†		2[ ]	*	*	*	*	1	*
SWAP dst	R IR		F0 F1	X	*	*	X	-	-
TCM dst, src (NOT dst) AND src	†		6[ ]	-	*	*	0	-	-
TM dst, src dst AND src	†		7[ ]	-	*	*	0	-	-
XOR dst, src dst←dst XOR src	†		B[ ]	-	*	*	0	-	-

† These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

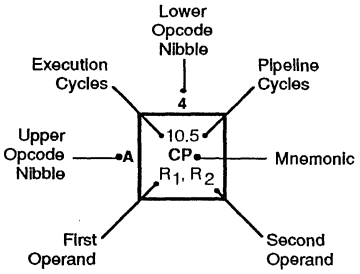
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address Mode dst	Address Mode src	Lower Opcode Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

OPCODE MAP

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								6.0 **STOP**
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								7.0 **HALT**
8	10.5 **DECW** RR1	10.5 **DECW** IR1	12.0 **LDE** r1, Irr2	18.0 **LDEI** Ir1, Irr2		**J85**										6.1 **DI**
9	6.5 **RL** R1	6.5 **RL** IR1	12.0 **LDE** r2, Irr1	18.0 **LDEI** Ir2, Irr1												6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14.0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2				10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1	12.0 **LDC** r2, Irr1	18.0 **LDCI** Ir2, Irr1	20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6.5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2			3					2					3	1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_2 = Dst address
R_1 or r_2 = Src address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank are not defined.

* 2-byte instruction appears as a 3-byte instruction

PRODUCT SPECIFICATION

Z86C21

CMOS Z8®
MICROCONTROLLER

FEATURES

- 8-bit CMOS microcontroller, 40- or 44-pin package
- 4.5 to 5.5 Voltage operating range
- Low power Consumption - 220 mW (max) @ 16 MHz
- Fast instruction pointer - 1.0 microsecond @ 12 MHz
- Two standby modes - STOP and HALT
- 32 input/output lines
- Full-Duplex UART
- All digital inputs are TTL levels
- Auto Latches
- RAM and ROM protect
- 8 Kbytes of ROM
- 236 bytes of RAM
- Two programmable 8-bit Counter/Timers each with 6-bit programmable prescaler.
- Six vectored, priority interrupts from eight different sources
- Clock speeds 12 and 16 MHz
- On-Chip oscillator that accepts a crystal, ceramic resonator, LC or external clock drive.

GENERAL DESCRIPTION

The Z86C21 microcontroller introduces a new level of sophistication to single-chip architecture. The Z86C21 is a member of the Z8 single-chip microcontroller family with 8 Kbytes of ROM and 236 bytes of RAM.

The MCU is housed in a 40-pin DIP, 44-pin Leaded Chip-Carrier, or a 44-pin Quad Flat Pack, and is manufactured in CMOS technology. The ROMless pin option is available on the 44-pin versions only. Having the ROM/ROMless selectively, the MCU offers both external memory and preprogrammed ROM which enables this Z8 microcontroller to be used in high volume applications or where code flexibility is required.

Zilog's CMOS microcontroller offers fast execution, efficient use of memory, sophisticated interrupts, input/output bit manipulation capabilities, and easy hardware/software system expansion along with low cost and low power consumption.

The Z86C21 architecture is characterized by Zilog's 8-bit microcontroller core. The device offers a flexible I/O scheme, an efficient register and address space structure, multiplexed capabilities between address/data, I/O, and a number of ancillary features that are useful in many industrial and advanced scientific applications.

The device applications demand powerful I/O capabilities. The Z86C21 fulfills this with 32 pins dedicated to input and output. These lines are grouped into four ports. Each port consists of eight lines, and is configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an address/data bus for interfacing external memory.

There are three basic address spaces available to support this wide range of configuration: Program Memory, Data Memory and 236 General-Purpose Registers.

GENERAL DESCRIPTION (Continued)

To unburden the program from coping with the real-time problems such as counting/timing and serial data communication, the Z86C21 offers two on-chip counter/timers with a large number of user selectable modes, and a Asynchronous Receiver/Transmitter (UART-Figure 1).

Note: All Signals with a preceding front slash, "/", are active Low, e.g.: B//W (WORD is active Low); /B/W (BYTE is active Low, only).

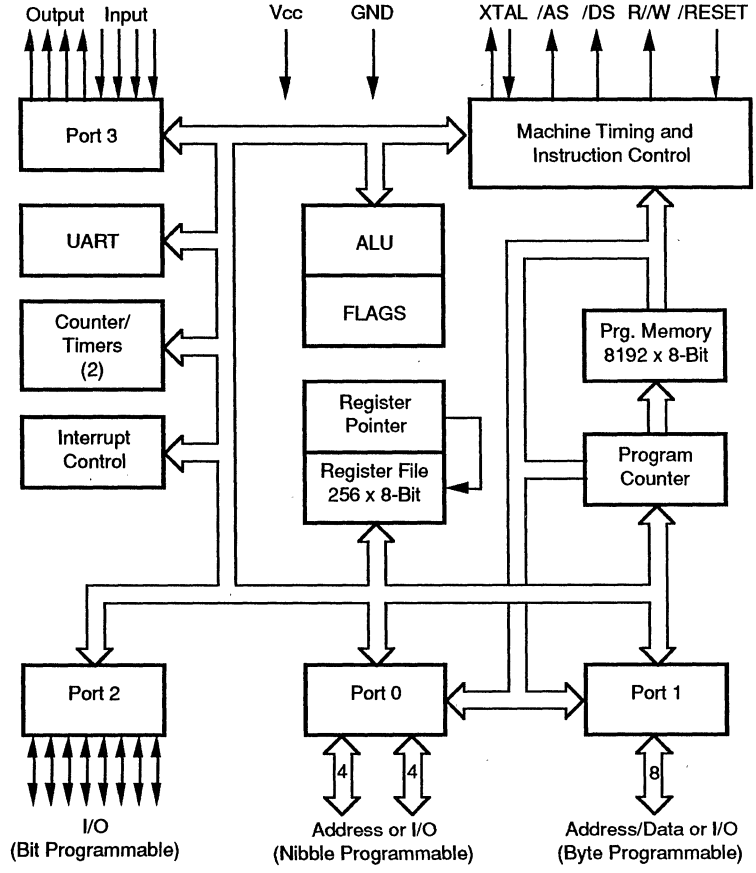


Figure 1. Functional Block Diagram

PIN DESCRIPTION

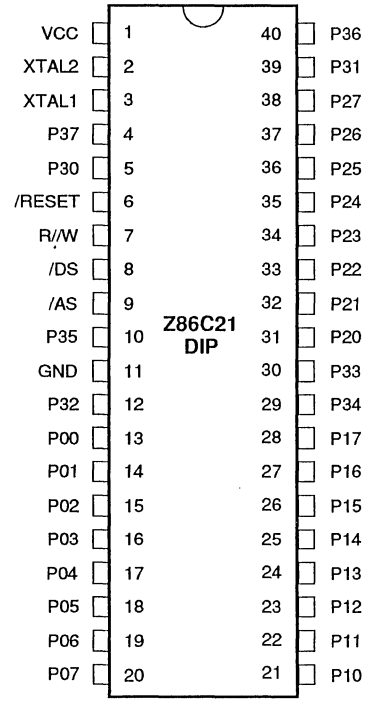


Figure 2. 40-Pin Dual In-Line Plastic Pin Assignments

Table 1. 40-Pin Dual In-Line Plastic Pin Identification

Pin #	Symbol	Function	Direction
1	V_{CC}	Power Supply	Input
2	XTAL2	Crystal, Oscillator Clock	Output
3	XTAL1	Crystal, Oscillator Clock	Input
4	P37	Port 3 pin 7	Output
5	P30	Port 3 pin 0	Input
6	/RESET	Reset	Input
7	R/W	Read/Write	Output
8	/DS	Data Strobe	Output
9	/AS	Address Strobe	Output
10	P35	Port 3 pin 5	Output
11	GND	Ground	Input
12	P32	Port 3 pin 2	Input
13-20	P00-P07	Port 0 pin 0,1,2,3,4,5,6,7	In/Output
21-28	P10-P17	Port 1 pin 0,1,2,3,4,5,6,7	In/Output
29	P34	Port 3 pin 4	Output
30	P33	Port 3 pin 3	Input
31-38	P20-P27	Port 2 pin 0,1,2,3,4,5,6,7	In/Output
39	P31	Port 3 pin 1	Input
40	P36	Port 3 pin 6	Output

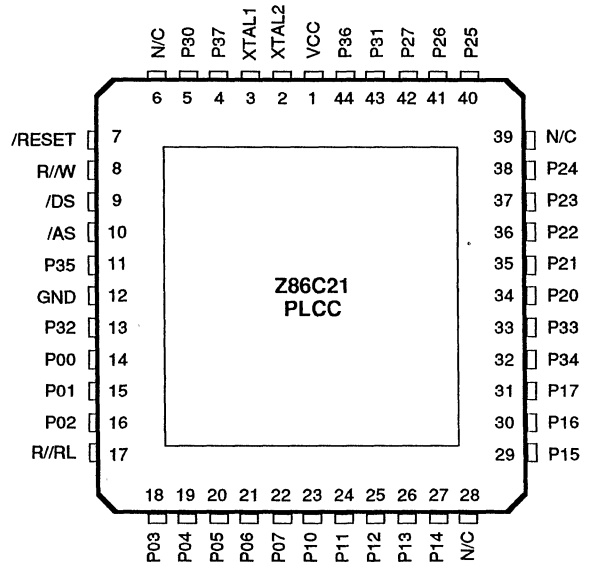


Figure 3. 44-Pin Plastic Leaded Chip Carrier Pin Assignments

Table 2. 44-Pin Plastic Leaded Chip Carrier Pin Identification

Pin #	Symbol	Function	Direction
1	V_{cc}	Power Supply	Input
2	XTAL2	Crystal, Oscillator Clock	Output
3	XTAL1	Crystal, Oscillator Clock	Input
4	P37	Port 3 pin 7	Output
5	P30	Port 3 pin 0	Input
6	N/C	Not Connected	Input
7	/RESET	Reset	Input
8	R//W	Read/Write	Output
9	/DS	Data Strobe	Output
10	/AS	Address Strobe	Output
11	P35	Port 3 pin 5	Output
12	GND	Ground	Input
13	P32	Port 3 pin 2	Input
14-16	P00-P02	Port 0 pin 0,1,2	In/Output
17	R//RL	ROM/ROMless control	Input
18-22	P03-P07	Port 0 pin 3,4,5,6,7	In/Output
23-27	P10-P14	Port 1 pin 0,1,2,3,4	In/Output
28	N/C	Not Connected	Input
29-31	P15-P17	Port 1 pin 5,6,7	In/Output
32	P34	Port 3 pin 4	Output
33	P33	Port 3 pin 3	Input
34-38	P20-P24	Port 2 pin 0,1,2,3,4	In/Output
39	N/C	Not Connected	Input
40-42	P25-P27	Port 2 pin 5,6,7	In/Output
43	P31	Port 3 pin 1	Input
44	P36	Port 3 pin 6	Output

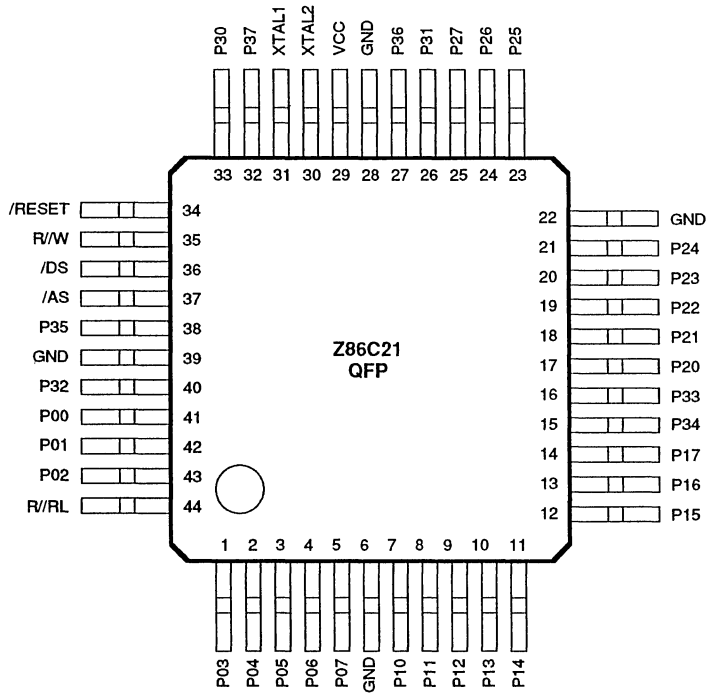


Figure 4. 44-Pin Quad Flat Pack Pin Assignments

Table 3. 44-Pin Quad Flat Pack Pin Identification

Pin #	Symbol	Function	Direction
1-5	P03-P07	Port 0 pin 3,4,5,6,7	In/Output
6	GND	Ground	Input
7-14	P10-P17	Port 1 pin 0,1,2,3,4,5,6,7	In/Output
15	P34	Port 3 pin 4	Output
16	P33	Port 3 pin 3	Input
17-21	P20-P24	Port 2 pin 0,1,2,3,4	In/Output
22	GND	Ground	Input
23-25	P25-P27	Port 2 pin 5,6,7	In/Output
26	P31	Port 3 pin 1	Input
27	P36	Port 3 pin 6	Output
28	GND	Ground	Input
29	V_{CC}	Power Supply	Input
30	XTAL2	Crystal, Oscillator Clock	Output

Pin #	Symbol	Function	Direction
31	XTAL1	Crystal, Oscillator Clock	Input
32	P37	Port 3 pin 7	Output
33	P30	Port 3 pin 0	Input
34	/RESET	Reset	Input
35	R//W	Read/Write	Output
36	/DS	Data Strobe	Output
37	/AS	Address Strobe	Output
38	P35	Port 3 pin 5	Output
39	GND	Ground	Input
40	P32	Port 3 pin 2	Input
41-43	P00-P02	Port 0 pin 0,1,2	In/Output
44	R//RL	ROM/ROMless control	Input

PIN FUNCTIONS

/ROMless. (input, active Low). This pin, when connected to GND, disables the internal ROM and forces the device to function as a Z86C91 ROMless Z8. (Note that, when left unconnected or pulled high to V_{CC}, the part functions as a normal Z86C21 ROM version). This pin is only available on the 44-pin versions of the Z86C21.

/DS. (output, active Low). Data Strobe is activated once for each external memory transfer. For a READ operation, data must be available prior to the trailing edge of /DS. For WRITE operations, the falling edge of /DS indicates that output data is valid.

/AS. (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Address output is via Port 1 for all external programs. Memory address transfers are valid at the trailing edge of /AS. Under program control, /AS is placed in the high-impedance state along with Ports 0 and 1, Data Strobe, and Read/ Write.

XTAL1, XTAL2. *Crystal 1, Crystal 2* (time-based input and output, respectively). These pins connect a parallel-resonant crystal, ceramic resonator, LC, or any external single-phase clock to the on-chip oscillator and buffer.

R/W. (output, write Low). The Read/Write signal is low when the MCU is writing to the external program or data memory.

/RESET. (input, active-Low). To avoid asynchronous and noisy reset problems, the Z86C21 is equipped with a reset filter of four external clocks (4TpC). If the external /RESET signal is less than 4TpC in duration, no reset occurs on the 5th clock after the /RESET is detected, an internal RST signal is latched and held for an internal register count of 18 external clocks, or for the duration of the external /RESET, whichever is longer. During the reset cycle, /DS is held active low while /AS cycles at a rate of TpC/2. When /RESET is deactivated, program execution begins at location 000C (HEX). Reset time must be held low for 50mS, or until V_{CC} is stable, whichever is longer.

Port 0. (P00-P07). Port 0 is an 8-bit, nibble programmable, bidirectional, TTL compatible port. These eight I/O lines can be configured under software control as a nibble I/O port, or as an address port for interfacing external memory. When used as an I/O port, Port 0 may be placed under handshake control. In this configuration, Port 3, lines P32 and P35 are used as the handshake control /DAV0 and RDY0 (Data Available and Ready). Handshake signal assignment is dictated by the I/O direction of the upper nibble P04-P07. The lower nibble must have the same direction as the upper nibble to be under handshake control. For the ROMless option, Port 0 comes up as A15-A8 Address lines after /RESET.

For external memory references, Port 0 can provide address bits A11-A8 (lower nibble) or A15-A8 (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 is programmed independently as I/O while the lower nibble is used for addressing. If one or both nibbles are needed for I/O operation, they must be configured by writing to the Port 0 Mode register. In ROMless mode, after a hardware reset, Port 0 lines are defined as address lines A15-A8, and extended timing is set to accommodate slow memory access. The initialization routine can include reconfiguration to eliminate this extended timing mode (Figure 5).

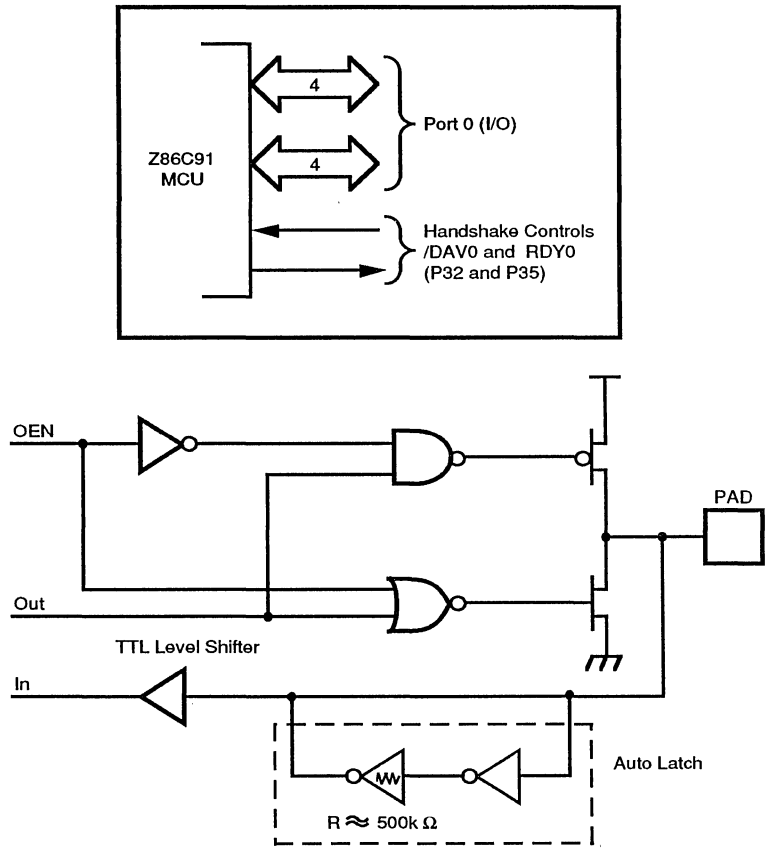


Figure 5. Port 0 Configuration

PIN FUNCTIONS (Continued)

Port 1. (P10-P17). Port 1 is an 8-bit, byte programmable, bidirectional, TTL compatible port. It has multiplexed Address (A7-A0) and Data (D7-D0) ports. For Z86C21, these eight I/O lines can be programmed as Input or Output lines or can be configured under software control as an address/data port for interfacing external memory. When used as an I/O port, Port 1 can be placed under handshake control. In this configuration, Port 3 line P33 and P34 are used as the handshake controls RDY1 and /DAV1.

Memory locations greater than 8192 are referenced through Port 1. To interface external memory, Port 1 is programmed for the multiplexed Address/Data mode. If more than 256 external locations are required, Port 0 must output the additional lines.

Port 1 can be placed in a high-impedance state along with Port 0, /AS, /DS and R//W, allowing the MCU to share common resource in multiprocessor and DMA applications. Data transfers are controlled by assigning P33 as a Bus Acknowledge input, and P34 as a Bus request output (Figure 6).

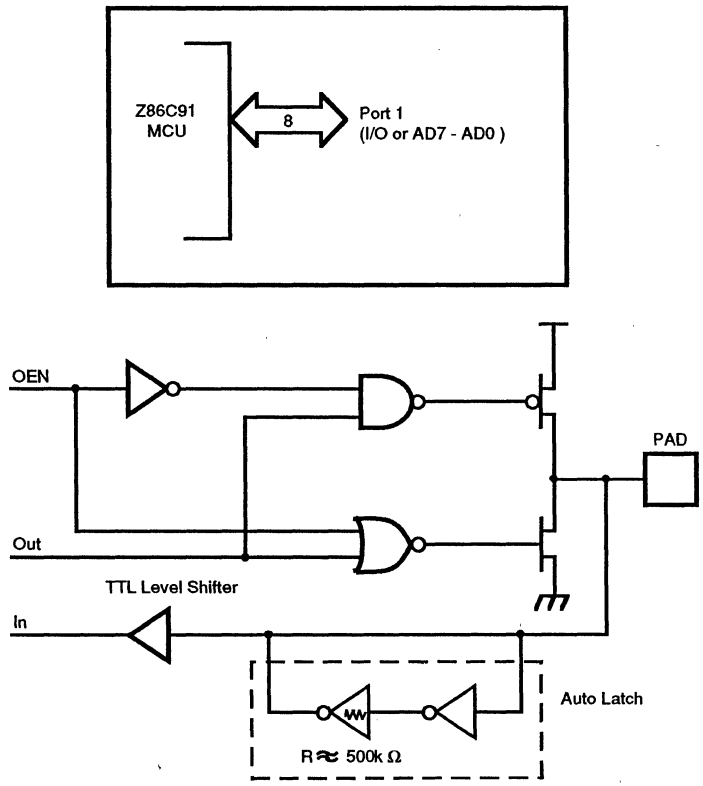


Figure 6. Port 1 Configuration

Port 2. (P20-P27). Port 2 is an 8-bit, bit programmable, bidirectional, CMOS compatible port. Each of these eight I/O lines can be independently programmed as an input or output or globally as an open-drain output. Port 2 is always available for I/O operation. When used as an I/O port, Port 2 may be placed under handshake control. In this configuration, Port 3 lines P31 and P36 are used as the handshake control lines /DAV2 and RDY2. The handshake signal assignment for Port 3 lines P31 and P36 is dictated by the direction (input or output) assigned to P27 (Figure 7).

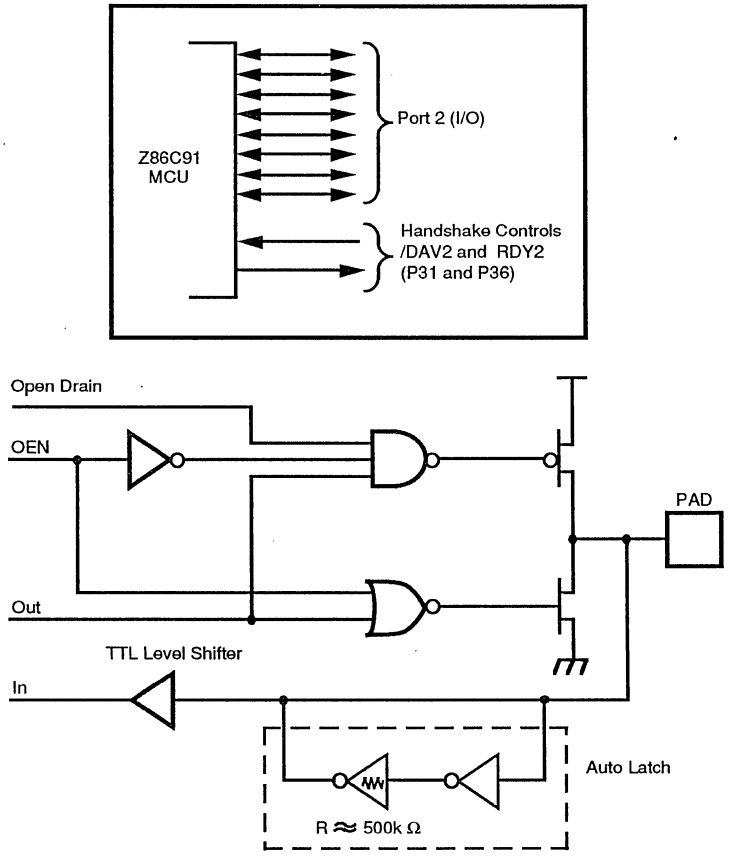


Figure 7. Port 2 Configuration

Port 3. (P30-P37). Port 3 is an 8-bit, CMOS compatible four-fixed-input and four-fixed-output port. These eight I/O lines have four-fixed input (P30-P33) and four fixed output (P34-P37) ports. Port 3, when used as serial I/O, is programmed as serial in and serial out, respectively (Figure 8 and Table 4).

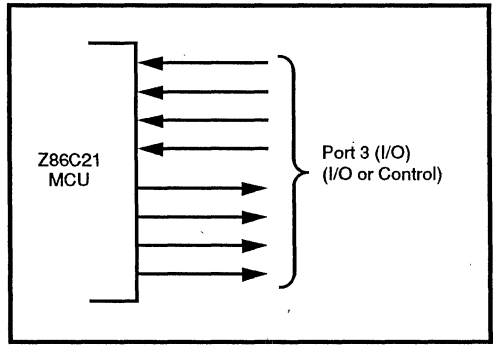


Figure 8. Port 3 Configuration

Port 3 is configured under software control to provide the following control functions: handshake for Ports 0 and 2 (/DAV and RDY); four external interrupt request signals (IRQ0-IRQ3); timer input and output signals (T_{IN} and T_{OUT}), and Data Memory Select (/DM).

Table 4. Port 3 Pin Assignments

Pin	I/O	CTC1	Int.	P0 HS	P1 HS	P2 HS	UART	Ext
P30	IN		IRQ3				Serial In	
P31	IN	T_{IN}	IRQ2			D/R		
P32	IN		IRQ0	D/R				
P33	IN		IRQ1		D/R			
P34	OUT				R/D			DM
P35	OUT			R/D				
P36	OUT	T_{OUT}				R/D		
P37	OUT						Serial Out	

Notes:
HS = HANDSHAKE SIGNALS
D = Data Available
R = Ready

Port 3 lines P30 and P37, are be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by the Counter/Timer 0.

The Z86C21 automatically adds a start bit and two stop bits to transmitted data (Figure 9). Odd parity is also available as an option. Eight data bits are always transmitted, regardless of parity selection. If parity is enabled, the eighth bit is the odd parity bit. An interrupt request (IRQ4) is generated on all transmitted characters.

Received data must have a start bit, 8 data bits and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ3 interrupt request.

Auto-Latch. The Auto-Latch puts valid CMOS levels on all CMOS inputs that are not externally driven. This reduces excessive supply current flow in the input buffer when it is not been driven by any source.

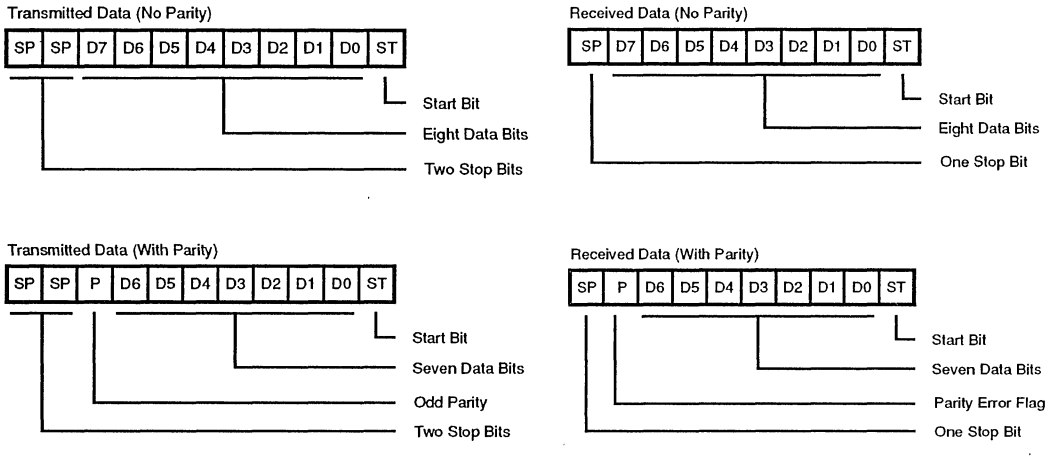


Figure 9. Serial Data Formats

Low EMI Option. The Z86C21 is available in a low EMI option. This option is mask-programmable, to be selected by the customer at the time when the ROM code is submitted. Use of this feature results in:

- Less than 1 mA current consumptions during HALT mode.
- The pre-drivers slew rate reduced to 10 ns typical.
- Low EMI output drivers have resistance of 200 ohms typical.
- Oscillator divide-by-two circuitry is eliminated.
- Internal SCLK/TCLK operation is limited to a maximum of 4 MHz (250 ns cycle time)

FUNCTIONAL DESCRIPTION

Address Space

Program Memory. The Z86C21 can address up to 56K bytes of external program memory (Figure 10). The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts. For ROM mode, byte 13 to byte 8191 consists of on-chip ROM. At addresses 8192 and greater, the Z86C21 executes external program memory fetches. In the ROMless mode, the Z86C21 can address up to 64K bytes of external program memory. Program execution begins at external location 000C (HEX) after a reset.

Data Memory (/DM). The ROM version can address up to 56K bytes of external data memory space beginning at location 8192. The ROMless version can address up to 64K bytes of external data memory. External data memory can be included with, or separated from, the external program memory space. /DM, an optional I/O function that can be programmed to appear on pin P34, is used to distinguish between data and program memory space (Figure 11). The state of the /DM signal is controlled by the type instruction being executed. An LDC opcode references PROGRAM (/DM inactive) memory, and an LDE instruction references DATA (/DM active low) memory.

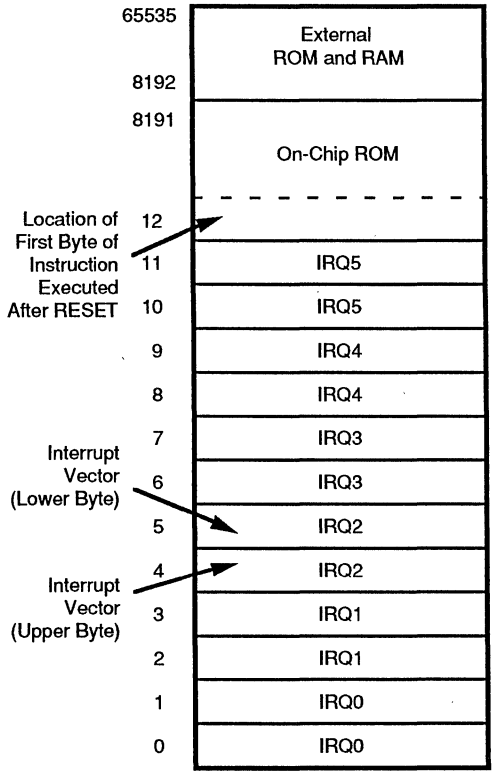


Figure 10. Program Memory Configuration

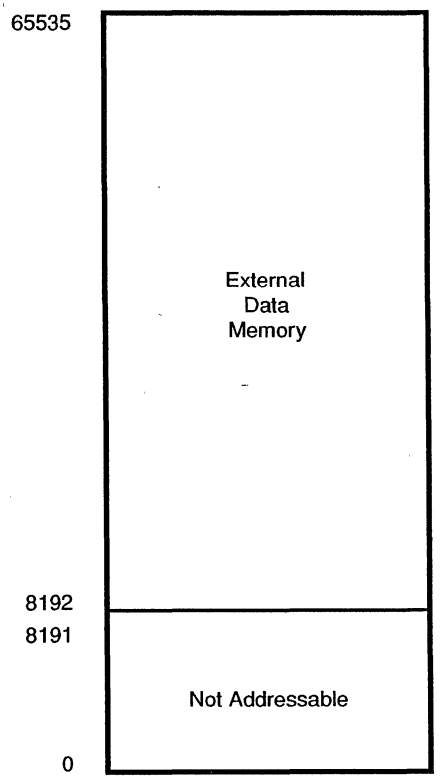


Figure 11. Data Memory Configuration

Register File. The Register File consists of four I/O port registers, 236 general-purpose registers and 16 control and status registers (Figure 12). The instructions can access registers directly or indirectly via an 8-bit address field. The Z86C21 also allows short 4-bit register addressing using the Register Pointer (Figure 13). In the 4-bit mode, the Register File is divided into 16 working register groups, each occupying 16 continuous locations. The Register Pointer addresses the starting location of the active working-register group.

Note: Register Bank E0-EF can only be accessed through working registers and indirect addressing modes.

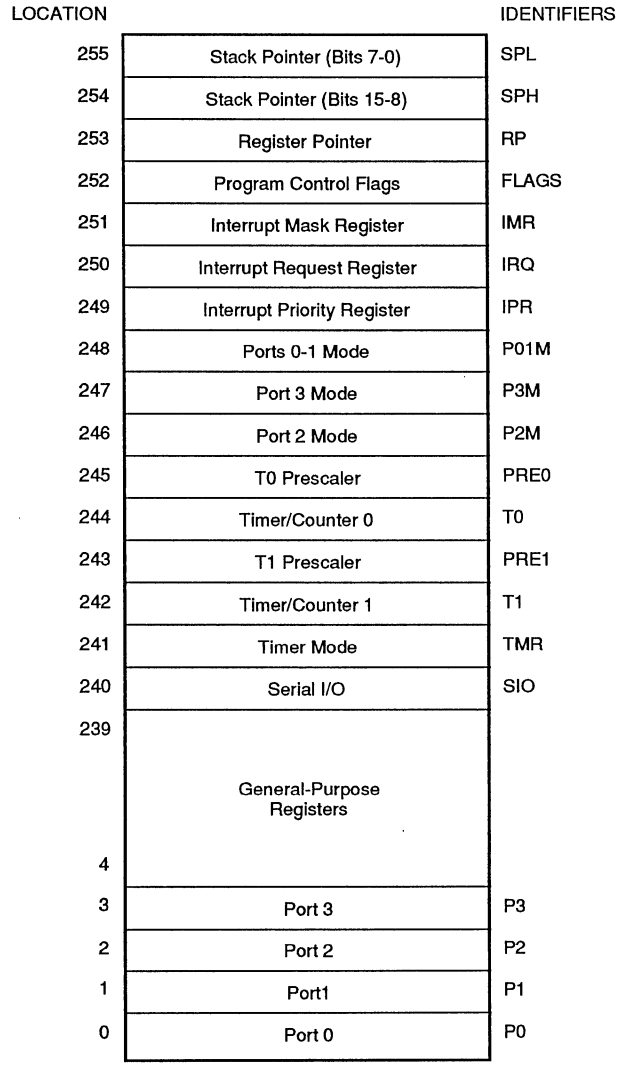


Figure 12. Register File

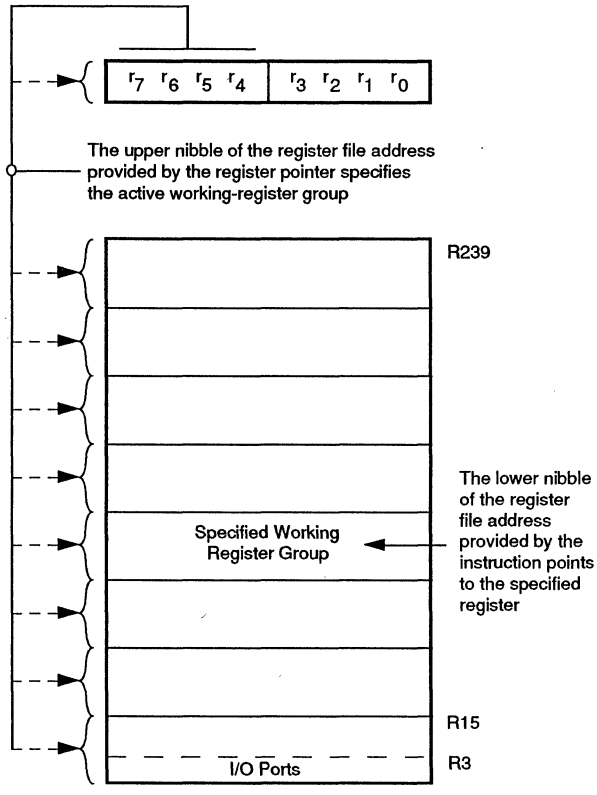


Figure 13. Register Pointer

RAM Protect. The upper portion of the RAM's address spaces 80FH to EFH (excluding the control registers) can be protected from reading and writing. The RAM Protect bit option is mask-programmable and is selected by the customer when the ROM code is submitted. After the mask option is selected, the user activates from the internal ROM code to turn off/on the RAM Protect by loading a bit D6 in the IMR register to either a 0 or a 1, respectively. A 1 in D6 indicates RAM Protect enabled.

ROM Protect. The first 8 Kbytes of program memory is mask programmable. A ROM protect feature prevent dumping of the ROM contents by inhibiting execution of LDC, LDCI, LDE, and LDEI instructions to Program Memory in all modes.

The ROM Protect option is mask-programmable, to be selected by the customer at the time when the ROM code is submitted.

Stack. The Z86C21 has a 16-bit Stack Pointer (R254-R255) used for external stack that resides anywhere in the data memory for the ROMless mode, but only from 8192 to 65535 in the ROM mode. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 236 general-purpose registers (R4-R239). The high byte of the Stack Pointer (SPH-Bit 8-15) is used as a general-purpose register when using internal stack only.

Counter/Timers. There are two 8-bit programmable counter/timers (T0-T1), each driven by its own 6-bit programmable prescaler. The T1 prescaler is driven by internal or external clock sources; however, the T0 prescaler is driven by the internal clock only (Figure 14).

The 6-bit prescalers divides the input frequency of the clock source by any integer number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When both the counter and prescaler reach the end of the count, a timer interrupt request, IRQ4 (T0) or IRQ5 (T1), is generated.

The counter can be programmed to start, stop, restart to continue, or restart from the initial value. The counters can also be programmed to stop upon reaching zero (single pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode).

The counter, but not the prescalers, can be read at any time without disturbing their value or count mode. The clock source for T1 is user-definable and can be either the internal microprocessor clock divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input (P31) as an external clock, a trigger input that is retriggerable or non-retriggerable, or as a gate input for the internal clock. Port 3, line P36, also serves as a timer output (T_{OUT}) through which T0, T1 or the internal clock is output. The counter/timers are cascaded by connecting the T0 output to the input of T1.

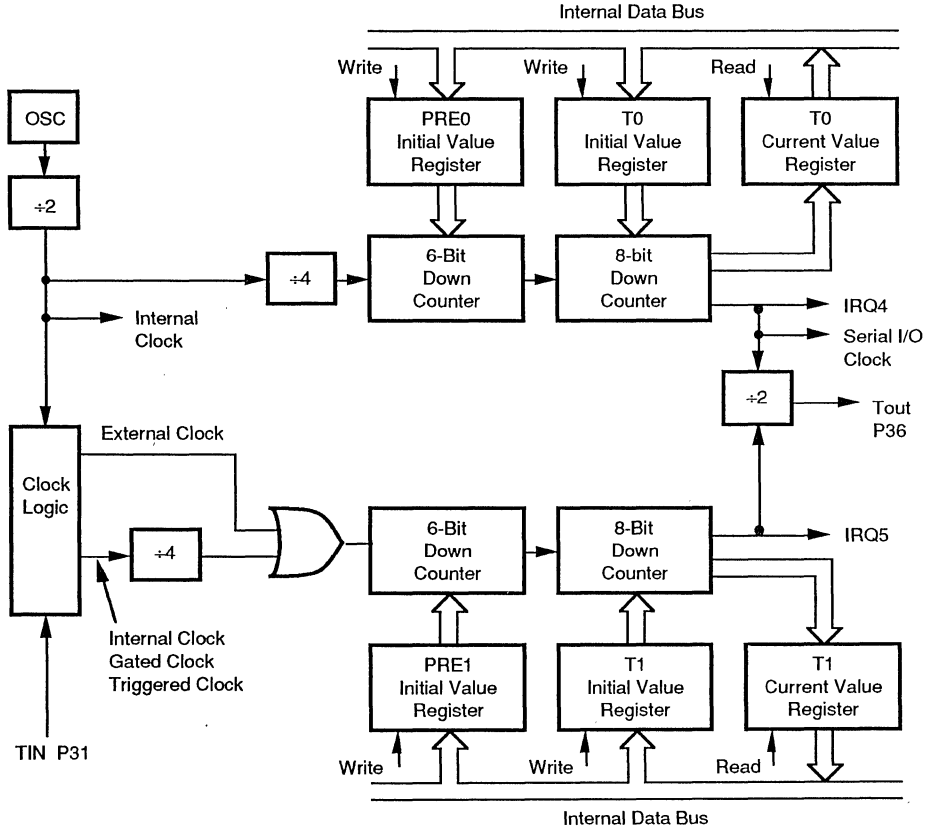


Figure 14. Counter/Timers Block Diagram

Interrupts. The Z86C21 has six different interrupts from eight different sources. The interrupts are maskable and prioritized. The eight sources are divided as follow: four sources are claimed by Port 3, lines P30-P33; one in Serial Out, one in Serial In, and two in the counter/timers (Figure 15). The Interrupt Mask Register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All Z86C21 interrupts are vectored through locations in the program memory. When an interrupt machine cycle is activated, an interrupt request is granted. Thus, this disables all of the subsequent interrupts, save the Program Counter and Status Flags, and then branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

To accommodate polled interrupt systems, interrupt inputs are masked and the Interrupt Request register is polled to determine which of the interrupt requests need service. Software initialed interrupts are supported by setting the appropriate bit in the Interrupt Request Register (IRQ).

Internal interrupt requests are sampled on the falling edge of the last cycle of every instruction, and the interrupt request must be valid 5TpC before the falling edge of the last clock cycle of the currently executing instruction.

For the ROMless mode, when the device samples a valid interrupt request, the next 48 (external) clock cycles are used to prioritize the interrupt, and push the two PC bytes and the FLAG register on the stack. The following nine cycles are used to fetch the interrupt vector from external memory. The first byte of the interrupt service routine is fetched beginning on the 58th TpC cycle following the internal sample point, which corresponds to the 63rd TpC cycle following the external interrupt sample point.

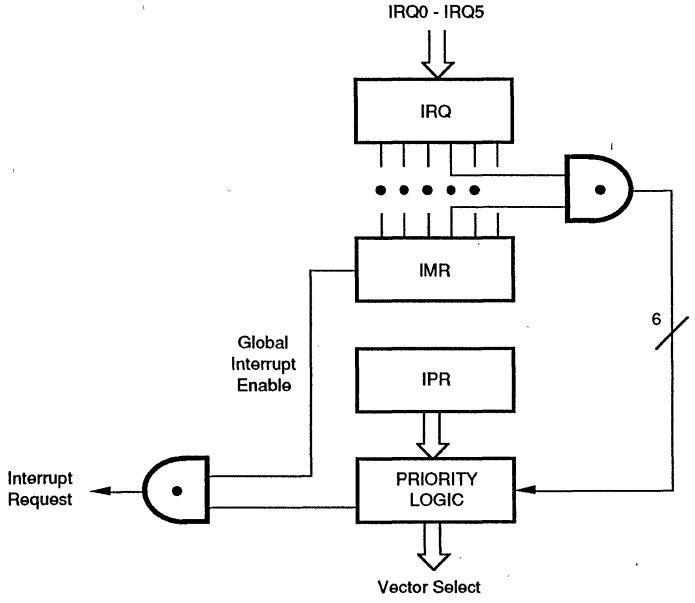


Figure 15. Interrupt Block Diagram

Clock. The Z86C21 on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal, LC, ceramic resonator, or any suitable external clock source (XTAL1=Input, XTAL2=Output). The crystal should be AT cut, 1 MHz to 16 MHz max, and series resistance (RS) is less than or equal to 100 Ohms. The crystal should be connected across XTAL1 and XTAL2 using the recommended capacitors (10 pF < CL < 300 pF) from each pin to ground (Figure 16).

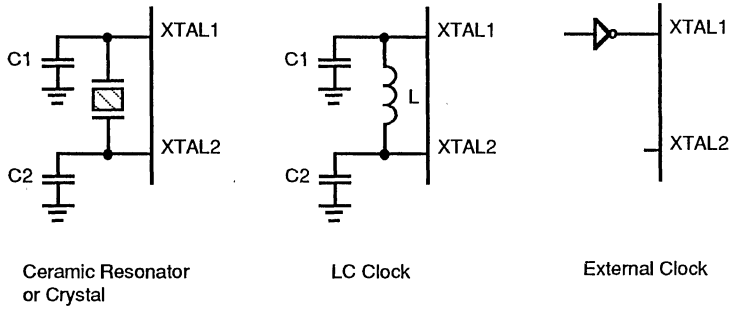


Figure 16. Oscillator Configuration

HALT. Turns off the internal CPU clock but not the XTAL oscillation. The counter/timers and the external interrupts IRQ0, IRQ1, IRQ2 and IRQ3 remain active. The devices are recovered by interrupts, either externally or internally generated.

STOP. This instruction turns off the internal clock and external crystal oscillation and reduces the standby current to 10 microamperes or less. The Stop mode is terminated by a reset which causes the processor to restart the application program at address 000C (HEX).

In order to enter STOP (or HALT) mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. To do this, the user must execute a NOP (opcode=0FFH) immediately before the appropriate sleep instruction. i.e.:

```
FF  NOP   ; clear the pipeline
6F  STOP  ; enter STOP mode
          or
FF  NOP   ; clear the pipeline
7F  HALT  ; enter HALT mode
```

ABSOLUTE MAXIMUM RATINGS

Symbol	Description	Min	Max	Units
V_{CC}	Supply Voltage*	-0.3	+7.0	V
T_{STG}	Storage Temp	-65	+150	C
T_A	Oper Ambient Temp		†	C

Notes:
* Voltages on all pins with respect to GND.
† See Ordering Information

Stress greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for an extended period may affect device reliability.

STANDARD TEST CONDITIONS

The characteristics listed below apply for standard test conditions as noted. All voltages are referenced to GND. Positive current flows into the referenced pin (Figure 17).

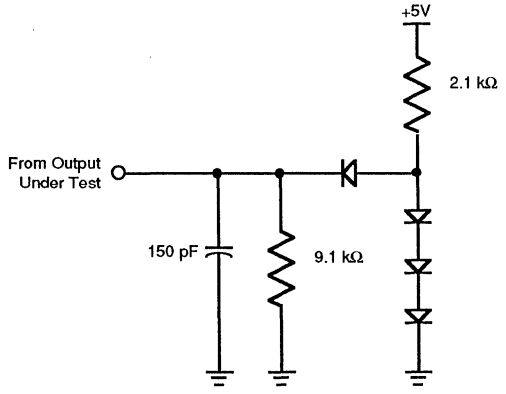


Figure 17. Test Load Diagram

DC CHARACTERISTICS

Sym	Parameter	T_A = 0°C to 70°C		T_A = -40°C to105°C		Typical at 25°C	Units	Conditions
		Min	Max	Min	Max			
	Max Input Voltage		7		7		V	I_{IN} 250µA
V_{CH}	Clock Input High Voltage	3.8	V_{CC}+0.3	3.8	V_{CC}+0.3		V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.03	0.8	-0.03	0.8		V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}+0.3	2.0	V_{CC}+0.3		V	
V_{IL}	Input Low Voltage	-0.3	0.8	-0.3	0.8		V	
V_{OH}	Output High Voltge	2.4		2.4			V	I_{OH} = -2.0 mA
V_{OH}	Output High Voltge	V_{CC}-100mV		V_{CC}-100mV			V	I_{OH} = -100 µA
V_{OL}	Output Low Voltage		0.4		0.4		V	I_{OL} = +5.0 mA
V_{RH}	Reset Input High Voltage	3.8	V_{CC}+0.3	3.8	V_{CC}+0.3		V	
V_{RI}	Reset Input Low Voltage	-0.03	0.8	-0.03	0.8		V	
I_{IL}	Input Leakage	-2	2	-2	2		µA	V_{IN} = 0V, V_{CC}
I_{OL}	Output Leakage	-2	2	-2	2		µA	V_{IN} = 0V, V_{CC}
I_{IR}	Reset Input Current		-80		-80		µA	V_{RL} = 0V
I_{CC}	Supply Current		30		30	20	mA	[1] @ 12 MHz
			35		35	24	mA	[1] @ 16 MHz
I_{CC1}	Standby Current		6.5		6.5	4	mA	[1] HALT Mode V_{IN} = 0V, V_{CC}@ 12 MHz
			7.0		7.0	4.5	mA	[1] HALT Mode V_{IN} = 0V,V_{CC}@ 16 MHz
I_{CC2}	Standby Current		10		20	5	µA	[1,2] STOP Mode V_{IN} = 0V,V_{CC}

Notes:

[1] All inputs driven to either 0V or V_{CC}, outputs floating.

[2] I_{CC2} requires loading TMR (%F1H) with any value prior to STOP execution.

Use this sequence:

```
LD TMR,#00
NOP
STOP
```

AC CHARACTERISTICS

External I/O or Memory Read or Write Timing Diagram

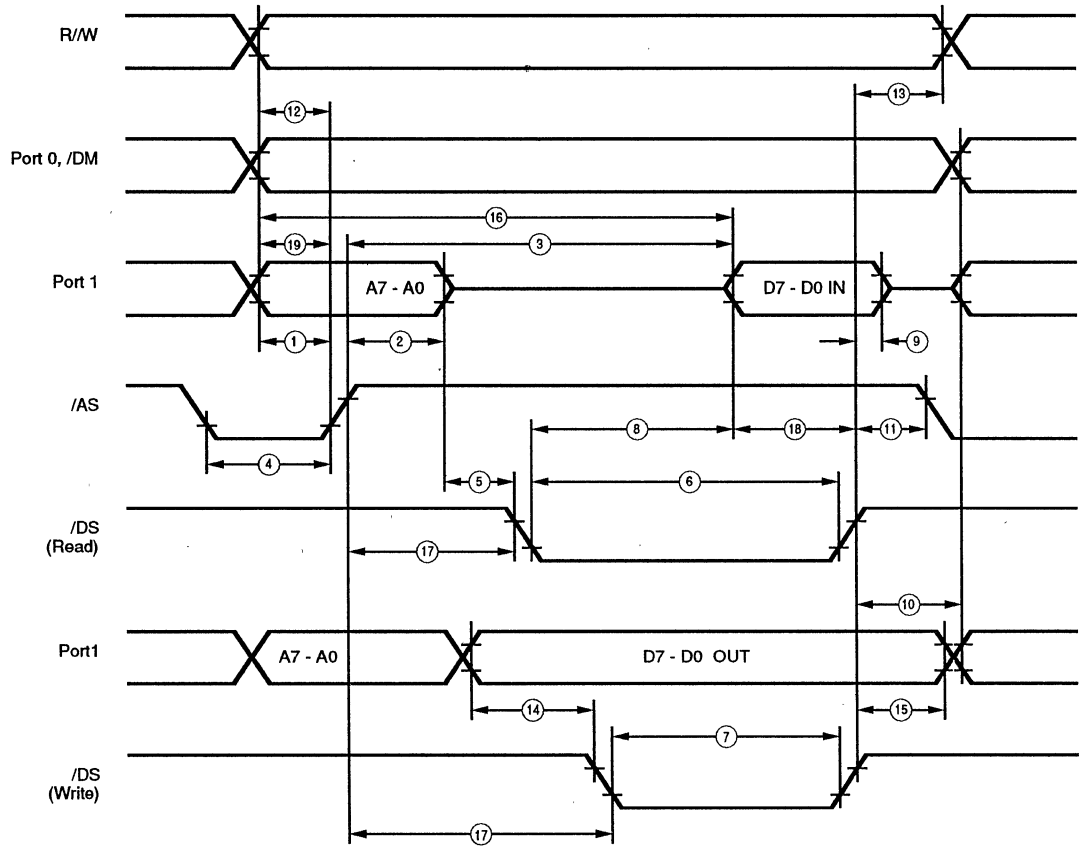


Figure 18. External I/O or Memory Read/Write Timing

AC CHARACTERISTICS

External I/O or Memory Read or Write Timing Table

No	Symbol	Parameter	T_A = 0°C to 70°C				T_A = -40°C to 105°C				Units	Notes
			12 MHz		16 MHz		12 MHz		16 MHz			
			Min	Max	Min	Max	Min	Max	Min	Max		
1	TdA(AS)	Address Valid to /AS Rise Delay	35		25		35		25		ns	[2,3]
2	TdAS(A)	/AS Rise to Address Float Delay	45		35		45		35		ns	[2,3]
3	TdAS(DR)	/AS Rise to Read Data Req'd Valid		250		180		250		180	ns	[1,2,3]
4	TwAS	/AS Low Width	55		40		55		40		ns	[2,3]
5	TdAZ(DS)	Address Float to /DS Fall	0		0		0		0		ns	
6	TwDSR	/DS (Read) Low Width	185		135		185		135		ns	[1,2,3]
7	TwDSW	/DS (Write) Low Width	110		80		110		80		ns	[1,2,3]
8	TdDSR(DR)	/DS Fall to Read Data Req'd Valid		130		75		130		75	ns	[1,2,3]
9	ThDR(DS)	Read Data to /DS Rise Hold Time	0		0		0		0		ns	[2,3]
10	TdDS(A)	/DS Rise to Address Active Delay	65		50		65		50		ns	[2,3]
11	TdDS(AS)	/DS Rise to /AS Fall Delay	45		35		45		35		ns	[2,3]
12	TdR/W(AS)	R//W Valid to /AS Rise Delay	30		20		33		25		ns	[2,3]
13	TdDS(R/W)	/DS Rise to R/W Not Valid	50		35		50		35		ns	[2,3]
14	TdDW(DSW)	Write Data Valid to /DS Fall (Write) Delay	35		25		35		25		ns	[2,3]
15	TdDS(DW)	/DS Rise to Write Data Not Valid Delay	55		35		55		35		ns	[2,3]
16	TdA(DR)	Address Valid to Read Data Req'd Valid		310		230		310		230	ns	[1,2,3]
17	TdAS(DS)	/AS Rise to /DS Fall Delay	65		45		65		45		ns	[2,3]
18	TdDI(DS)	Data Input Setup to /DS Rise	75		60		75		60		ns	[1,2,3]
19	TdDM(AS)	/DM Valid to /AS Rise Delay	50		30		50		30		ns	[2,3]

Notes:
[1] When using extended memory timing add 2 TpC.
[2] Timing numbers given are for minimum TpC.
[3] See clock cycle dependent characteristics table.

Standard Test Load
All timing references use 2.0V for a logic 1 and 0.8V for a logic 0.

Clock Dependent Formulas

Number	Symbol	Equation
1	TdA(AS)	0.401pC + 0.32
2	TdAS(A)	0.591pC - 3.25
3	TdAS(DR)	2.381pC + 6.14
4	TwAS	0.661pC - 1.65
6	TwDSR	2.331pC - 10.56
7	TwDSW	1.271pC + 1.67
8	TdDSR(DR)	1.971pC - 42.5
10	TdDS(A)	0.81pC
11	TdDS(AS)	0.591pC - 3.14
12	TdR/W(AS)	0.4TpC
13	TdDS(R/W)	0.8TpC - 15
14	TdDW(DSW)	0.4TpC
15	TdDS(DW)	0.881pC - 19
16	TdA(DR)	4TpC - 20
17	TdAS(DS)	0.911pC - 10.7
18	TsDI(DS)	0.8TpC - 10
19	TdDM(AS)	0.9TpC - 26.3

AC CHARACTERISTICS
Additional Timing Diagram

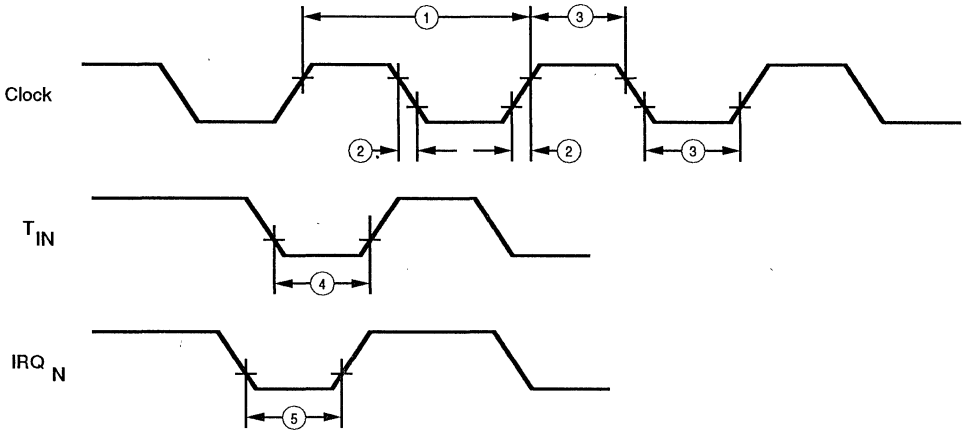


Figure 19. Additional Timing

AC CHARACTERISTICS
Additional Timing Table

No	Symbol	Parameter	T_A = 0°C to 70°C				T_A = -40°C to 105°C				Units	Notes
			12 MHz		16 MHz		12 MHz		16 MHz			
			Min	Max	Min	Max	Min	Max	Min	Max		
1	TpC	Input Clock Period	83	1000	62.5	1000	83	1000	62.5	1000	ns	[1]
2	TrC,TfC	Clock Input Rise & Fall Times		15		10		15		10	ns	[1]
3	TwC	Input Clock Width	35		25		35		25		ns	[1]
4	TwTinL	Timer Input Low Width	75		75		75		75		ns	[2]
5	TwTinH	Timer Input High Width	3TpC		3TpC		3TpC		3TpC			[2]
6	TpTin	Timer Input Period	8TpC		8TpC		8TpC		8TpC			[2]
7	TrTin,TfTin	Timer Input Rise & Fall Times	100		100		100		100		ns	[2]
8A	TwIL	Interrupt Request Input Low Times	70		70		70		50		ns	[2,4]
8B	TwIL	Interrupt Request Input Low Times	3TpC		3TpC		3TpC		3TpC			[2,5]
9	TwIH	Interrupt Request Input High Times	3TpC		3TpC		3TpC		3TpC			[2,3]

Notes:
[1] Clock timing references use 3.8V for a logic 1 and 0.8V for a logic 0.
[2] Timing references use 2.0V for a logic 1 and 0.8V for a logic 0.
[3] Interrupt references request via Port 3.
[4] Interrupt request via Port 3 (P31-P33).
[5] Interrupt request via Port 30.

AC CHARACTERISTICS
Handshake Timing Diagrams

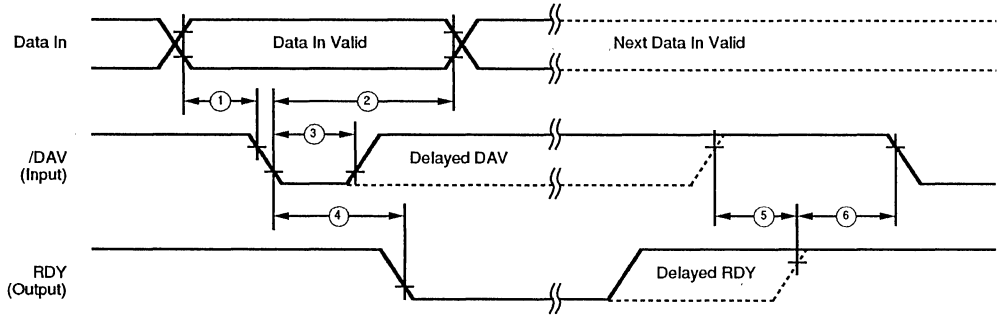


Figure 20. Input Handshake Timing

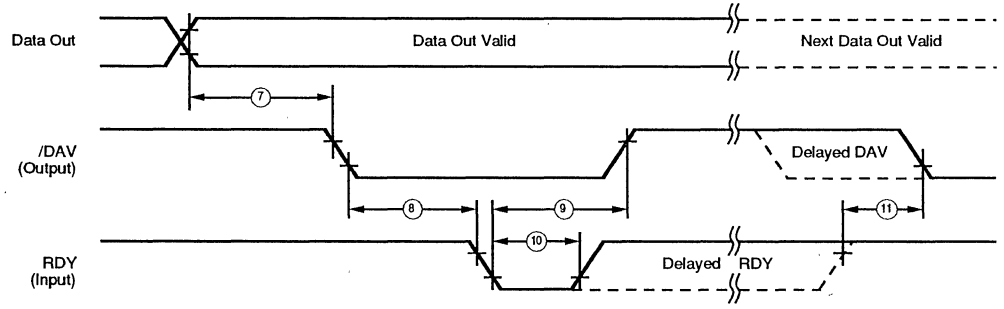


Figure 21. Output Handshake Timing

AC CHARACTERISTICS
Handshake Timing Table

No	Symbol	Parameter	T_A = 0°C to 70°C				T_A = -40°C to 105°C				Notes
			12 MHz		16 MHz		12 MHz		16 MHz		Data
			Min	Max	Min	Max	Min	Max	Min	Max	Direction
1	TsDI(DAV)	Data In Setup Time	0		0		0		0		IN
2	ThDI(DAV)	Data In Hold Time	145		145		145		145		IN
3	TwDAV	Data Available Width	110		110		110		110		IN
4	TdDAVI(RDY)	DAV Fall to RDY Fall Delay		115		115		115		115	IN
5	TdDAVId(RDY)	DAV Rise to RDY Rise Delay		115		115		115		115	IN
6	TdDO(DAV)	RDY Rise to DAV Fall Delay	0		0		0		0		IN
7	TcLDAV0(RDY)	Data Out to DAV Fall Delay		TpC		TpC		TpC		TpC	OUT
8	TcLDAV0(RDY)	DAV Fall to RDY Fall Delay	0		0		0		0		OUT
9	TdRDY0(DAV)	RDY Fall to DAV Rise Delay		115		115		115		115	OUT
10	TwRDY	RDY Width	110		110		110		110		OUT
11	TdRDY0d(DAV)	RDY Rise to DAV Fall Delay		115		115		115		115	OUT

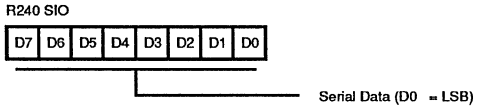


Figure 22. Serial I/O Register (F0H: Read/Write)

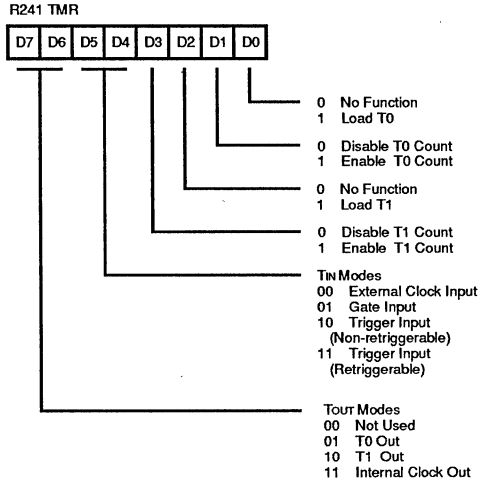


Figure 23. Timer Mode Register (F1H: Read/Write)

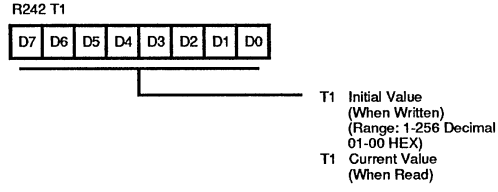


Figure 24. Counter/Timer 1 Register (F2H: Read/Write)

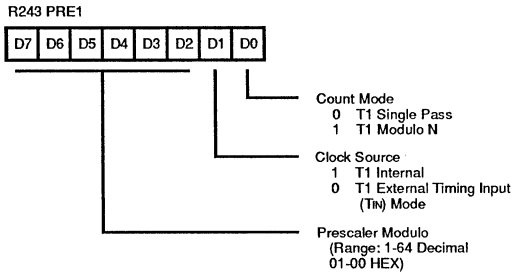


Figure 25. Prescaler 1 Register (F3H: Write Only)

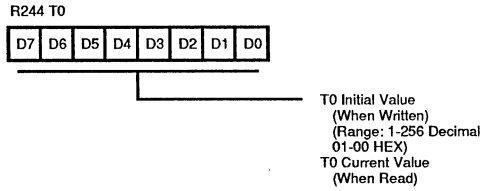


Figure 26. Counter/Timer 0 Register (F4H: Read/Write)

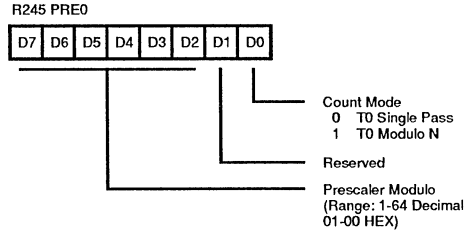


Figure 27. Prescaler 0 Register (F5H: Write Only)

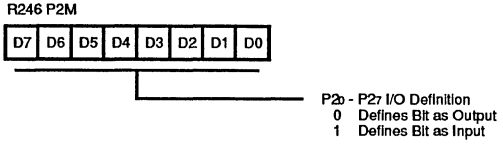


Figure 28. Port 2 Mode Register (F6H: Write Only)

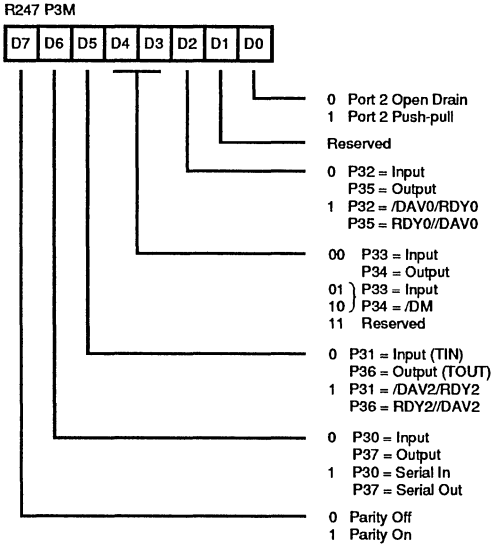


Figure 29. Port 3 Mode Register (F7H: Write Only)

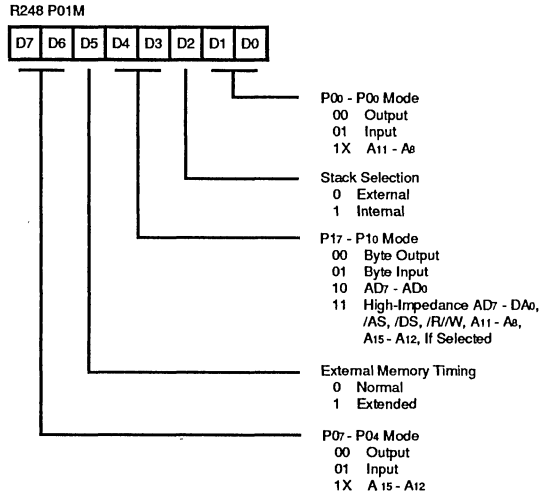


Figure 30. Port 0 and 1 Mode Register (F8H: Write Only)

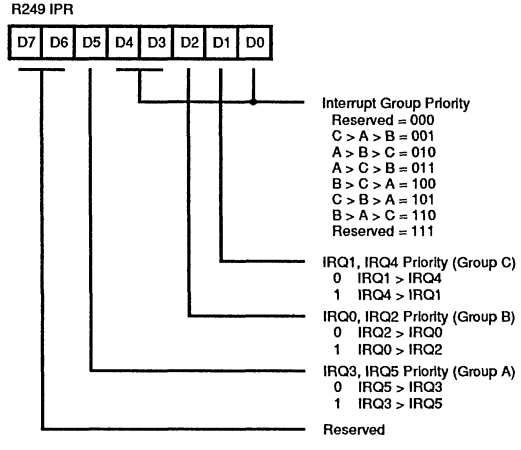


Figure 31. Interrupt Priority Register (F9H: Write Only)

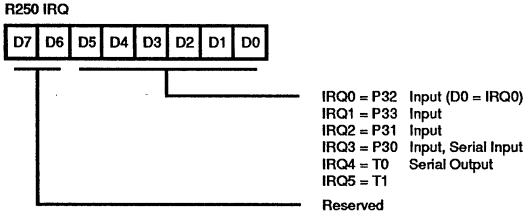


Figure 32. Interrupt Request Register (FAH: Read/Write)

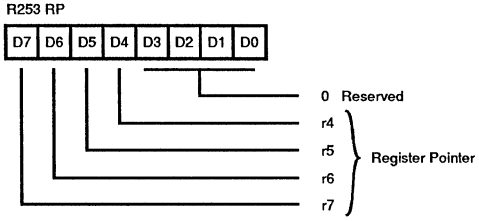


Figure 35. Register Pointer Register (FDH: Read/Write)

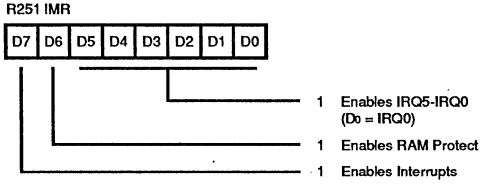


Figure 33. Interrupt Mask Register (FBH: Read/Write)

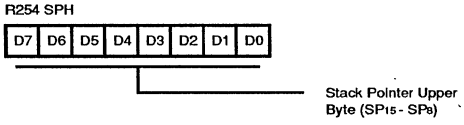


Figure 36. Stack Pointer Register (FEH: Read/Write)

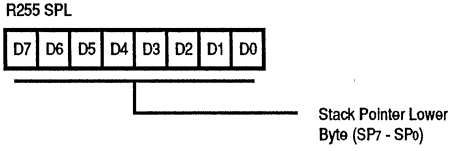


Figure 37. Stack Pointer Register (FFH: Read/Write)

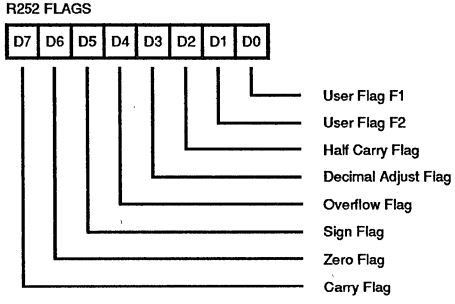


Figure 34. Flag Register (FCH: Read/Write)

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

Symbol	Meaning
IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

Symbol	Meaning
dst	Destination location or contents
src	Source location or contents
cc	Condition code
@	Indirect address prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag register (Control Register 252)
RP	Register Pointer (R253)
IMR	Interrupt mask register (R251)

Flags. Control register (R252) contains the following six flags:

Symbol	Meaning
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

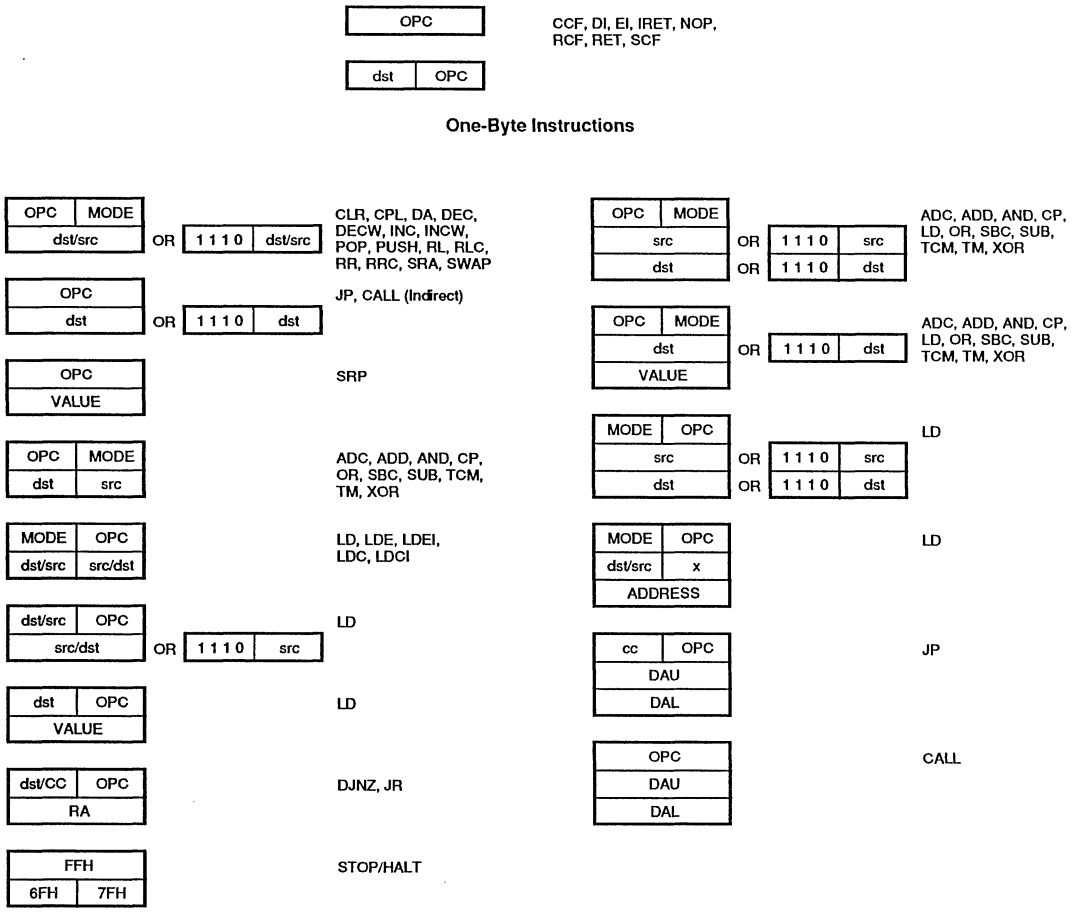
Affected flages are indicated by:

0	Clear to zero
1	Set to one
*	Set to clear according to operation
-	Unaffected
x	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000		Always True	
0111	C	Carry	C = 1
1111	NC	No Carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not Zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No Overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not Equal	Z = 0
1001	GE	Greater Than or Equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater Than	[Z OR (S XOR V)] = 0
0010	LE	Less Than or Equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned Greater Than or Equal	C = 0
0111	ULT	Unsigned Less Than	C = 1
1011	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned Less Than or Equal	(C OR Z) = 1
0000		Never True	

INSTRUCTION FORMATS



INSTRUCTION SUMMARY

Note: Assignment of a value is indicated by the symbol " ← ". For example:

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr (n)" is used to refer to bit (n) of a given operand location. For example:

dst (7)

refers to bit 7 of the destination operand.

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst, src dst←dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst←dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst←dst AND src	†		5[ ]	-	*	*	0	-	-
CALL dst SP←SP - 2 @SP←PC, PC←dst	DA IRR		D6 D4	-	-	-	-	-	-
CCF C←NOT C			EF	*	-	-	-	-	-
CLR dst dst←0	R IR		B0 B1	-	-	-	-	-	-
COM dst dst←NOT dst	R IR		60 61	-	*	*	0	-	-
CP dst, src dst - src	†		A[ ]	*	*	*	*	-	-
DA dst dst←DA dst	R IR		40 41	*	*	*	X	-	-
DEC dst dst←dst - 1	R IR		00 01	-	*	*	*	-	-
DECW dst dst←dst - 1	RR IR		80 81	-	*	*	*	-	-
DI IMR(7)←0			8F	-	-	-	-	-	-
DJNZr, dst r←r - 1 if r ≠ 0 PC←PC + dst Range: +127, -128	RA		rA r = 0 - F	-	-	-	-	-	-
EI IMR(7)←1			9F	-	-	-	-	-	-
HALT			7F	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
INC dst dst←dst + 1	r R IR		rE r = 0 - F 20 21	-	*	*	*	-	-
INCW dst dst←dst + 1	RR IR		A0 A1	-	*	*	*	-	-
IRET FLAGS←@SP; SP←SP + 1 PC←@SP; SP←SP + 2; IMR(7)←1			BF	*	*	*	*	*	*
JP cc, dst if cc is true PC←dst	DA IRR		cD c = 0 - F 30	-	-	-	-	-	-
JR cc, dst if cc is true, PC←PC + dst Range: +127, -128	RA		cB c = 0 - F	-	-	-	-	-	-
LD dst, src dst←src	r r R r X r r Ir R R R IR IR	Im R r X r Ir r R IR IM IM R	rC r8 r9 r = 0 - F C7 D7 E3 F3 E4 E5 E6 E7 F5	-	-	-	-	-	-
LDC dst, src	r	Irr	C2	-	-	-	-	-	-
LDCI dst, src dst←src r←r +1; rr←rr + 1	Ir	Irr	C3	-	-	-	-	-	-

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
NOP			FF	-	-	-	-	-	-
OR dst, src dst←dst OR src	†		4[ ]	-	*	*	0	-	-
POP dst dst←@SP; SP←SP + 1	R IR		50 51	-	-	-	-	-	-
PUSH src SP←SP - 1; @SP←src		R IR	70 71	-	-	-	-	-	-
RCF C←0			CF	0	-	-	-	-	-
RET PC←@SP; SP←SP + 2			AF	-	-	-	-	-	-
RL dst	R IR		90 91	*	*	*	*	-	-
RLC dst	R IR		10 11	*	*	*	*	-	-
RR dst	R IR		E0 E1	*	*	*	*	-	-
RRC dst	R IR		C0 C1	*	*	*	*	-	-
SBC dst, src dst←dst←src←C	†		3[ ]	*	*	*	*	1	*
SCF C←1			DF	1	-	-	-	-	-
SRA dst	R IR		D0 D1	*	*	*	0	-	-
SRP src RP←src		Im	31	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
STOP			6F	-	-	-	-	-	-
SUB dst, src dst←dst←src	†		2[ ]	*	*	*	*	1	*
SWAP dst	R IR		F0 F1	X	*	*	X	-	-
TCM dst, src (NOT dst) AND src	†		6[ ]	-	*	*	0	-	-
TM dst, src dst AND src	†		7[ ]	-	*	*	0	-	-
XOR dst, src dst←dst XOR src	†		B[ ]	-	*	*	0	-	-

† These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

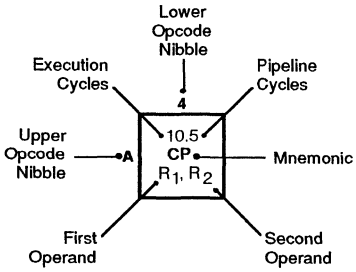
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address Mode dst	Address Mode src	Lower Opcode Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

OPCODE MAP

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								6.0 **STOP**
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								7.0 **HALT**
8	10.5 **DECW** RR1	10.5 **DECW** IR1	12.0 **LDE** r1, Irr2	18.0 **LDEI** Ir1, Irr2												6.1 **DI**
9	6.5 **RL** R1	6.5 **RL** IR1	12.0 **LDE** r2, Irr1	18.0 **LDEI** Ir2, Irr1												6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14.0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2				10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1	12.0 **LDC** r2, Irr1	18.0 **LDCI** Ir2, Irr1	20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6.5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2			3					2					3	1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_2 = Dst address
R_1 or r_2 = Src address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank are not defined.

* 2-byte instruction appears as a 3-byte instruction

PRODUCT SPECIFICATION

Z86E21

CMOS Z8® OTP MICROCONTROLLER

FEATURES

- 8-bit CMOS microcontroller, 40- or 44-pin package
- 4.5 to 5.5 Voltage operating range
- Low Power consumption - 275 mW (max)
- Fast instruction pointer - 1.0 microseconds @ 12 MHz
- Two standby modes - STOP and HALT
- 32 input/output lines
- Full-Duplex UART
- All digital inputs are TTL levels
- Auto latches
- High voltage protection on high voltage inputs
- RAM and EPROM protect
- 8 Kbytes of EPROM
- 256 bytes of RAM (236 for general purpose)
- Two programmable 8-bit Counter/Timers each with 6-bit programmable prescaler.
- Six vectored, priority interrupts from eight different sources.
- Clock speeds 12 and 16 MHz
- On-chip oscillator that accepts a crystal, ceramic resonator, LC or external clock drive.

GENERAL DESCRIPTION

The Z86E21 microcontroller (MCU) introduces the next level of sophistication to single-chip architecture. The Z86E21 is a member of the Z8 single-chip microcontroller family with 8 Kbytes of EPROM and 236 bytes of general purpose RAM.

The Z86E21 is a pin compatible, One-Time-Programmable (OTP) version of the Z86C21. The Z86E21 contains 8 Kbytes of EPROM memory in place of the 8 Kbyte of ROM on the Z86C21.

The MCU is housed in a 40-pin DIP, 44-pin Leaded Chip-Carrier, or a 44-pin Quad Flat Pack, and is manufactured in CMOS technology. The ROMless pin option is available on the 44-pin versions only. The MCU can address both external memory and preprogrammed ROM which enables this Z8 microcomputer to be used in high volume applications or where code flexibility is required.

Zilog's CMOS microcontroller offers fast execution, efficient use of memory, sophisticated interrupts, input/output bit manipulation capabilities, and easy hardware/software system expansion along with low cost and low power consumption.

The Z86E21 architecture is based on Zilog's 8-bit microcontroller core. The device offers a flexible I/O scheme, an efficient register and address space structure, multiplexed capabilities between address/data, I/O, and a number of ancillary features that are useful in many industrial and advanced scientific applications.

The device applications demand powerful I/O capabilities. The Z86E21 fulfills this with 32-pin dedicated to input and output. These lines are grouped into four ports. Each port consists of eight lines, and is configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an address/data bus for interfacing external memory.

GENERAL DESCRIPTION (Continued)

There are three basic address spaces available to support this wide range of configuration: Program Memory, Data Memory and 236 General-Purpose registers.

To unburden the program from coping with real-time problems such as counting/timing and serial data communication, the Z86E21 offers two on-chip counter/timers with a large number of user selectable modes, and an asynchronous receiver/transmitter (UART) (Figure 1).

Note: All Signals with a preceding front slash, "/", are active Low, e.g.: B/W (WORD is active Low); /B/W (BYTE is active Low, only); /N//S (NORMAL and SYSTEM are both active Low).

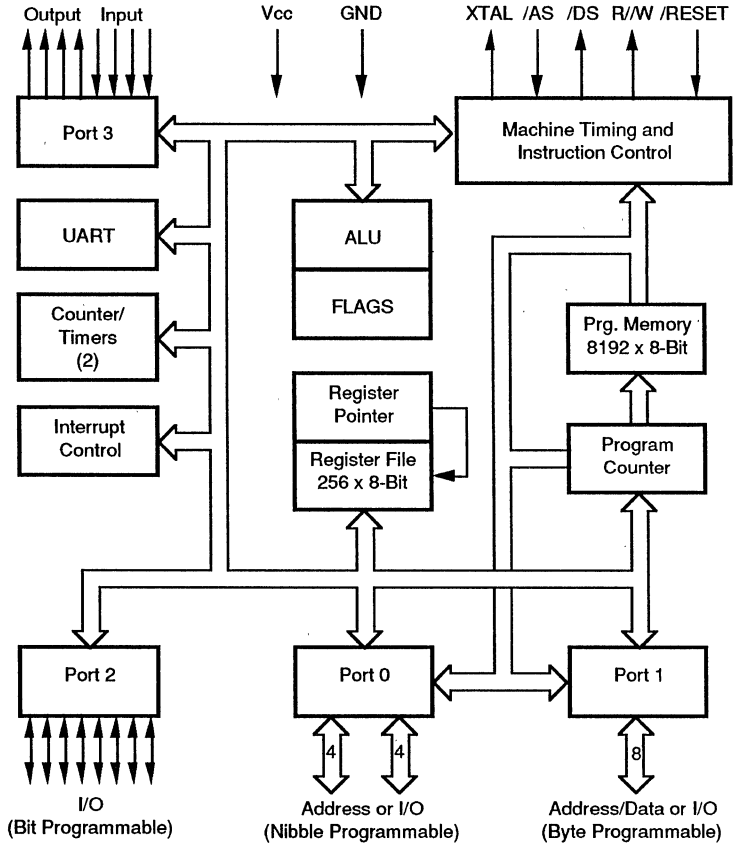


Figure 1. Functional Block Diagram

PIN DESCRIPTION
Standard Mode

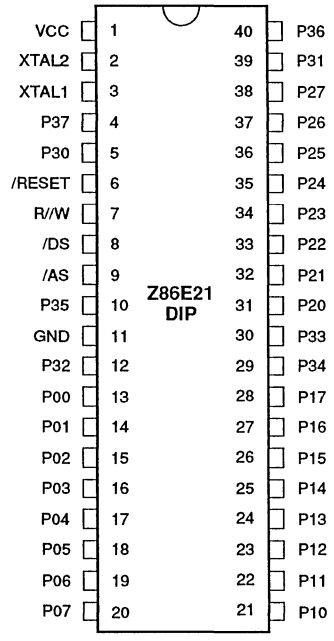


Figure 2. 40-Pin Dual-In-Line Pin Assignments

Table 1. 40-Pin Dual-In-Line Pin Identification (Standard Mode)

Pin #	Symbol	Function	Direction
1	V_{cc}	Power Supply	Input
2	XTAL1	Crystal, Oscillator Clock	Output
3	XTAL2	Crystal, Oscillator Clock	Input
4	P37	Port 3 pin 7	Output
5	P30	Port 3 pin 0	Input
6	/RESET	Reset	Input
7	R//W	Read/Write	Output
8	/DS	Data Strobe	Output
9	/AS	Address Strobe	Output
10	P35	Port 3 pin 5	Output

Pin #	Symbol	Function	Direction
11	GND	Ground, GND	Input
12	P32	Port 3 pin 2	Input
13-20	P00-P07	Port 0 pin 0,1,2,3,4,5,6,7	In/Output
21-28	P10-P17	Port 1 pin 0,1,2,3,4,5,6,7	In/Output
29	P34	Port 3 pin 4	Output
30	P33	Port 3 pin 3	Input
31-38	P20-P27	Port 2 pin 0,1,2,3,4,5,6,7	In/Output
39	P31	Port 3 pin 1	Input
40	P36	Port 3 pin 6	Output

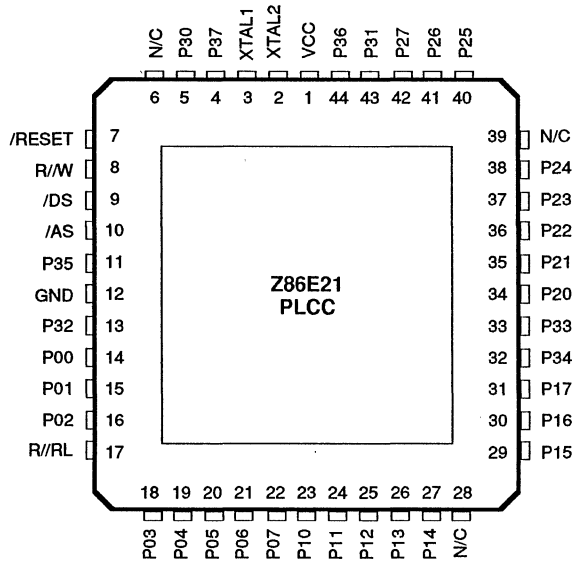


Figure 3. 44-Pin Leaded Chip Carrier Pin Assignments

Table 2. 44-Pin Leaded Chip Carrier Pin Identification (Standard Mode)

Pin #	Symbol	Function	Direction
1	V_{CC}	Power Supply	Input
2	XTAL1	Crystal, Oscillator Clock	Output
3	XTAL2	Crystal, Oscillator Clock	Input
4	P37	Port 3 pin 7	Output
5	P30	Port 3 pin 0	Input
6	N/C	Not Connected	Input
7	/RESET	Reset	Input
8	R/W	Read/Write	Output
9	/DS	Data Strobe	Output
10	/AS	Address Strobe	Output
11	P35	Port 3 pin 5	Output
12	GND	Ground, GND	Input
13	P32	Port 3 pin 2	Input

Pin #	Symbol	Function	Direction
14-16	P00-P02	Port 0 pin 0,1,2	In/Output
17	R//RL	ROM/ROMless control	Input
18-22	P03-P07	Port 0 pin 3,4,5,6,7	In/Output
23-27	P10-P14	Port 1 pin 0,1,2,3,4	In/Output
28	N/C	Not Connected	Input
29-31	P15-P17	Port 1 pin 5,6,7	In/Output
32	P34	Port 3 pin 4	Output
33	P33	Port 3 pin 3	Input
34-38	P20-P24	Port 2 pin 0,1,2,3,4	In/Output
39	N/C	Not Connected	Input
40-42	P25-P27	Port 2 pin 5,6,7	In/Output
43	P31	Port 3 pin 1	Input
44	P36	Port 3 pin 6	Output

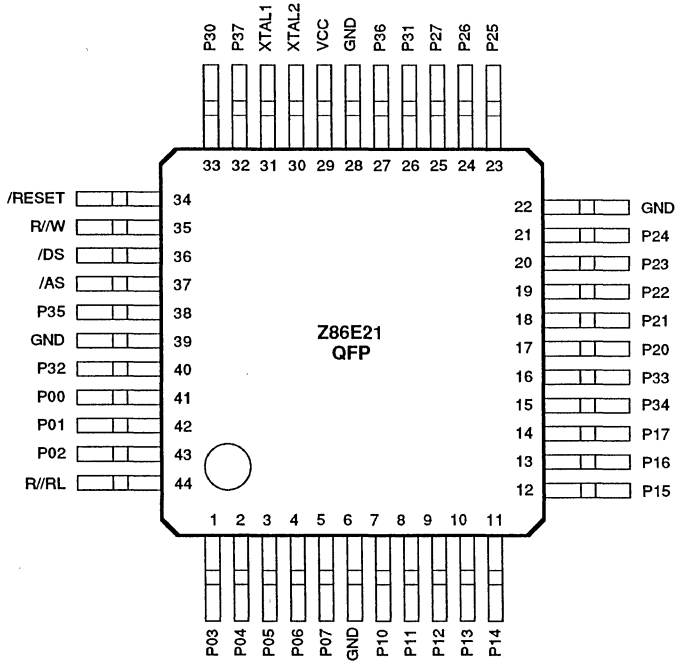


Figure 4. 44-Pin Quad Flat Pack Pin Assignments

Table 3. 44-Pin Quad Flat Pack Pin Identification (Standard Mode)

Pin #	Symbol	Function	Direction
1-5	P03-P07	Port 0 pin 3,4,5,6,7	In/Output
6	GND	Ground, GND	Input
7-14	P10-P17	Port 1 pin 0,1,2,3,4,5,6,7	In/Output
15	P34	Port 3 pin 4	Output
16	P33	Port 3 pin 3	Input
17-21	P20-P24	Port 2 pin 0,1,2,3,4	In/Output
22	GND	Ground, GND	Input
23-25	P25-P27	Port 2 pin 5,6,7	In/Output
26	P31	Port 3 pin 1	Input
27	P36	Port 3 pin 6	Output
28	GND	Ground, GND	Input
29	V_{CC}	Power Supply	Input
30	XTAL1	Crystal, Oscillator Clock	Output

Pin #	Symbol	Function	Direction
31	XTAL2	Crystal, Oscillator Clock	Input
32	P37	Port 3 pin 7	Output
33	P30	Port 3 pin 0	Input
34	/RESET	Reset	Input
35	R//W	Read/Write	Output
36	/DS	Data Strobe	Output
37	/AS	Address Strobe	Output
38	P35	Port 3 pin 5	Output
39	GND	Ground, GND	Input
40	P32	Port 3 pin 2	Input
41-43	P00-P02	Port 0 pin 0,1,2	In/Output
44	R//RL	ROM/ROMless control	Input

PIN DESCRIPTION (Continued)
EPROM Mode

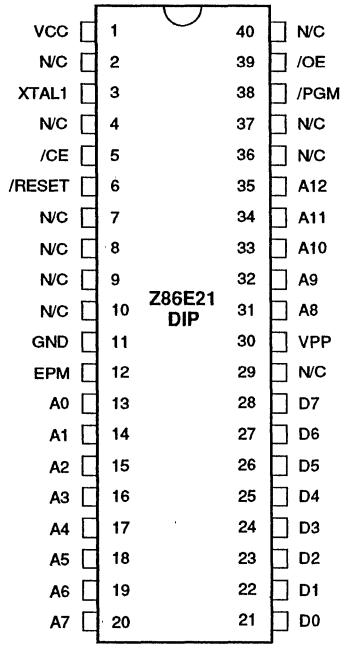


Figure 5. 40-Pin Dual-In-Line Pin Assignments

Table 4. 40-Pin Dual-In-Line Pin Identification (EPROM Mode)

Pin #	Symbol	Function	Direction
1	V_{CC}	Power Supply	Input
2	N/C	Not Connected	Input
3	XTAL1	Crystal, Oscillator Clock	Input
4	N/C	Not Connected	Input
5	/CE	Chip Enable	Input
6	/RESET	Reset	Input
7-10	N/C	Not Connected	Input
11	GND	Ground, GND	Input
12	EPM	EPROM Prog Mode	Input

Pin #	Symbol	Function	Direction
13-20	A0-A7	Address 0,1,2,3,4,5,6,7	Input
21-28	D0-D7	Data 0,1,2,3,4,5,6,7	In/Output
29	N/C	Not Connected	Input
30	V_{PP}	Prog Voltage	Input
31-35	A8-A12	Address 8,9,10,11,12	Input
36-37	N/C	Not Connected	Input
38	/PGM	Prog Mode	Input
39	/OE	Output Enable	Input
40	N/C	Not Connected	Input

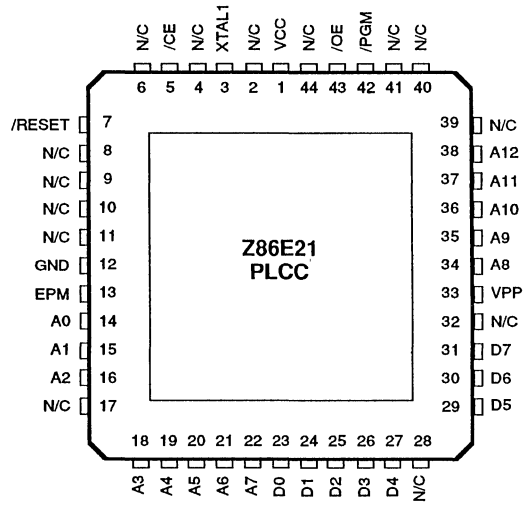


Figure 6. 44-Pin Leaded Chip Carrier Pin Assignments

Table 5. 44-Pin Leaded Chip Carrier Pin Identification (EPROM Mode)

Pin #	Symbol	Function	Direction
1	V_{CC}	Power Supply	Input
2	N/C	Not Connected	Input
3	XTAL1	Crystal, Oscillator Clock	Input
4	N/C	Not Connected	Input
5	/CE	Chip Enable	Input
6	N/C	Not Connected	Input
7	/RESET	Reset	Input
8-11	N/C	Not Connected	Input
12	GND	Ground, GND	Input
13	EPM	EPROM Prog Mode	Input
14-16	A0-A2	Address 0,1,2	Input
17	N/C	Not Connected	Input
18-22	A3-A7	Address 3,4,5,6,7	Input
23-27	D0-D4	Data 0,1,2,3,4	In/Output
28	N/C	Not Connected	Input
29-31	D5-D7	Data 5,6,7	In/Output
32	N/C	Not Connected	Input
33	V_{PP}	Prog Voltage	Input
34-38	A8-A12	Address 8,9,10,11,12	Input
39-41	N/C	Not Connected	Input
42	/PGM	Prog Mode	Input
43	/OE	Output Enable	Input
44	N/C	Not Connected	Input

PIN DESCRIPTION (Continued)
EPROM Mode

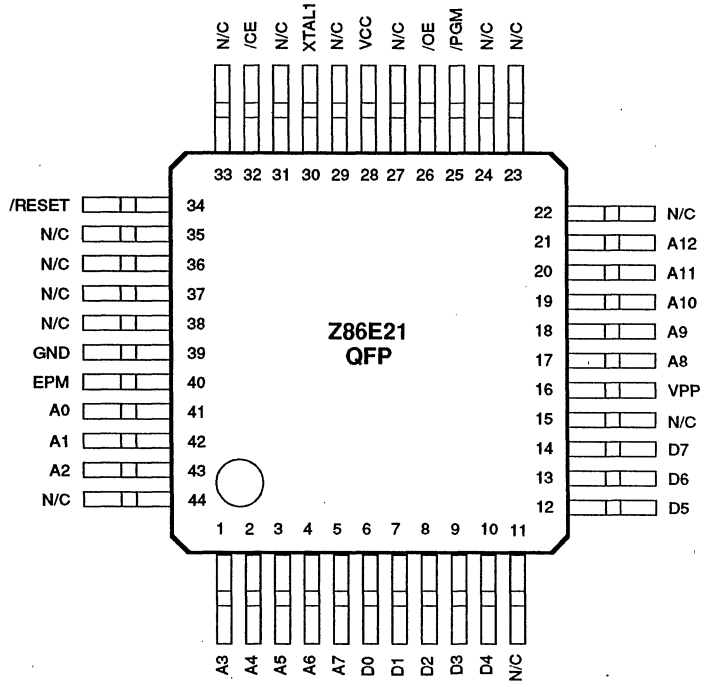


Figure 7. 44-Pin Quad Flat Pack Pin Assignments

Table 6. 44-Pin Quad Flat Pack Pin Identification (EPROM Mode)

Pin #	Symbol	Function	Direction
1-5	A3-A7	Address 3,4,5,6,7	Input
6-10	D0-D4	Data 0,1,2,3,4	In/Output
11	N/C	Not Connected	Input
12-14	D5-D7	Data 5,6,7	In/Output
15	N/C	Not Connected	Input
16	V_{PP}	Prog Voltage	Input
17-21	A8-A12	Address 8,9,10,11,12	Input
22-24	N/C	Not Connected	Input
25	/PGM	Prog Mode	Input
26	/OE	Output Enable	Input
27	N/C	Not Connected	Input
28	V_{CC}	Power Supply	Input

Pin #	Symbol	Function	Direction
29	N/C	Not Connected	Input
30	XTAL1	Crystal, Oscillator Clock	Input
31	N/C	Not Connected	Input
32	/CE	Chip Enable	Input
33	N/C	Not Connected	Input
34	/RESET	Reset	Input
35-38	N/C	Not Connected	Input
39	GND	Ground, GND	Input
40	EPM	EPROM Prog Mode	Input
41-43	A0-A2	Address 0,1,2	Input
44	N/C	Not Connected	Input

PIN FUNCTIONS

ROMless. (input, active Low). This pin when connected to GND disables the internal ROM and forces the device to function as a Z86C91 ROMless Z8 (note that, when left unconnected or pulled high to Vcc, the part will function as a normal Z86E21 EPROM version). This pin is only available on the 44-pin version of the Z86E21.

/DS. (output, active Low). Data Strobe is activated once for each external memory transfer. For a READ operation, data must be available prior to the trailing edge of /DS. For WRITE operations, the falling edge of /DS indicates that output data is valid.

/AS. (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Address output is via Port 1 for all external programs. Memory address transfers are valid at the trailing edge of /AS. Under program control, /AS can be placed in the high-impedance state along with Ports 0 and 1, Data Strobe, and Read/Write.

XTAL1, XTAL2. *Crystal 1, Crystal 2* (time-based input and output, respectively). These pins connect a parallel-resonant crystal, ceramic resonator, LC, or any external single-phase clock to the on-chip oscillator and buffer.

R/W. (output, write Low). The Read/Write signal is low when the MCU is writing to the external program or data memory.

/RESET. (input, active-Low). To avoid asynchronous and noisy reset problems, the Z86E21 is equipped with a reset filter of four external clocks (4TpC). If the external /RESET signal is less than 4TpC in duration, no reset occurs.

On the 5th clock after the /RESET is detected, an internal RST signal is latched and held for an internal register count of 18 external clocks, or for the duration of the external /RESET, whichever is longer. During the reset cycle, /DS is held active low while /AS cycles at a rate of TpC/2. When /RESET is deactivated, program execution begins at location 000C (HEX). Reset time must be held low for 50 mS, or until Vcc is stable, whichever is longer.

Port 0 P00-P07. Port 0 is an 8-bit, nibble programmable, bidirectional, TTL compatible port. These eight I/O lines can be configured under software control as a nibble I/O port, or as an address port for interfacing external memory. When used as an I/O port, Port 0 may be placed under handshake control. In this configuration, Port 3, lines P32 and P35 are used as the handshake control /DAV0 and RDY0 (Data Available and Ready). Handshake signal assignment is dictated by the I/O direction of the upper nibble P04-P07. The lower nibble must have the same direction as the upper nibble to be under handshake control. For the ROMless option, Port 0 comes up as A15-A8 Address lines after /RESET.

For external memory references, Port 0 can provide address bits A11-A8 (lower nibble) or A15-A8 (lower and upper nibbles) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. If one or both nibbles are needed for I/O operation, they must be configured by writing to the Port 0 Mode register. In ROMless mode, after a hardware reset, Port 0 lines are defined as address lines A15-A8, and extended timing is set to accommodate slow memory access. The initialization routine can include reconfiguration to eliminate this extended timing mode (Figure 8).

Port 1 P10-P17. Port 1 is an 8-bit, byte programmable, bidirectional, TTL compatible port. It has multiplexed Address (A7-A0) and Data (D7-D0) ports. For Z86E21, these eight I/O lines can be programmed as Input or Output lines or are configured under software control as an address/data port for interfacing external memory. When used as an I/O port, Port 1 can be placed under handshake control. In this configuration, Port 3 lines, P33 and P34, are used as the handshake controls RDY1 and /DAV1.

Memory locations greater than 8192 are referenced through Port 1. To interface external memory, Port 1 must be programmed for the multiplexed Address/Data mode. If more than 256 external locations are required, Port 0 must output the additional lines.

Port 1 can be placed in high-impedance state along with Port 0, /AS, /DS and R//W, allowing the MCU to share common resources in multiprocessor and DMA applications. Data transfers are controlled by assigning P33 as a Bus Acknowledge input, and P34 as a Bus request output (Figure 9).

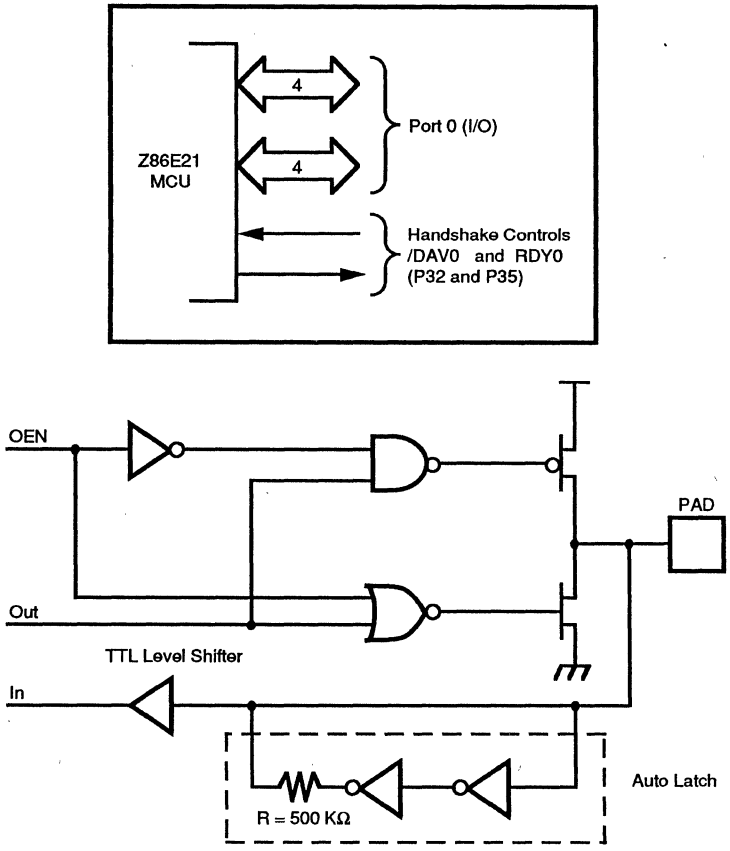


Figure 8. Port 0 Configuration

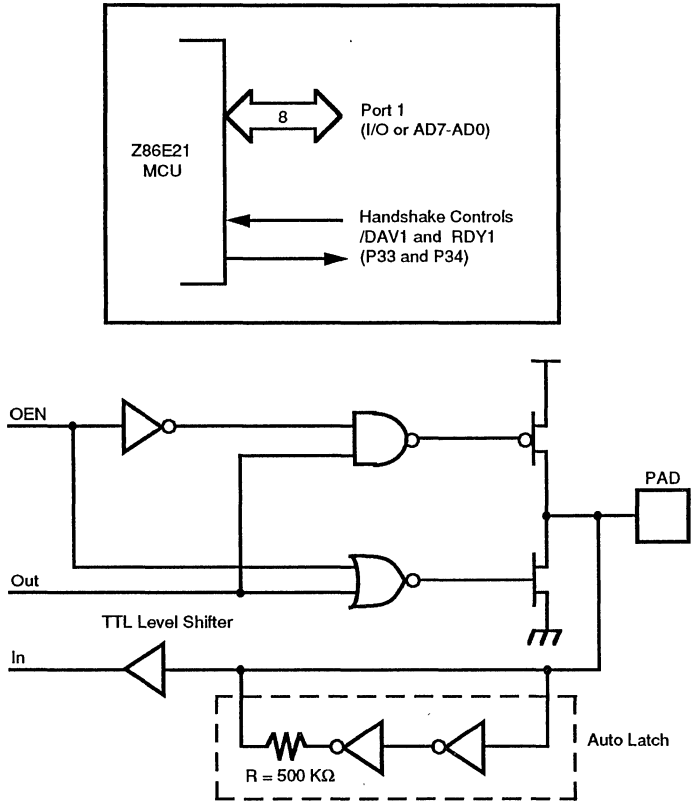


Figure 9. Port 1 Configuration

Port 2 P20-P27. Port 2 is an 8-bit, bit programmable, bidirectional, CMOS compatible port. Each of these eight I/O lines can be independently programmed as an input or output, or globally as an open-drain output. Port 2 is always available for I/O operation. When used as an I/O port, Port 2 can be placed under handshake control. In this configuration, Port 3 lines P31 and P36 are used as the handshake control lines /DAV2 and RDY2. The handshake signal assignment for Port 3 lines, P31 and P36, is dictated by the direction (input or output) assigned to P27 (Figure 10 and Table 7).

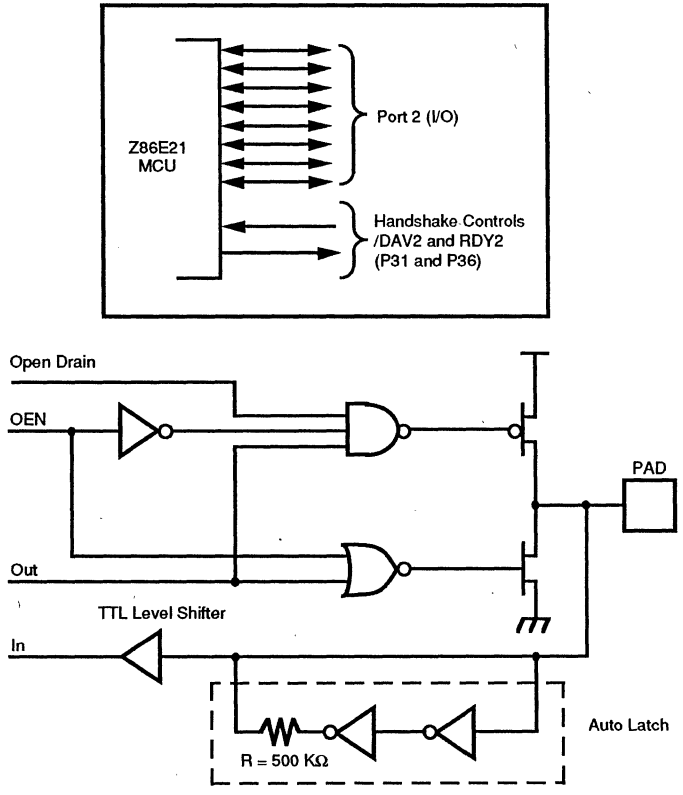


Figure 10. Port 2 Configuration

Port 3 P30-P37. Port 3 is an 8-bit, CMOS compatible four fixed input and four fixed output port. These 8 I/O lines have four-fixed (P33-P30) input and four fixed (P37-P34) output ports. Port 3, when used as serial I/O, is programmed as serial in and serial out, respectively (Figure 11).

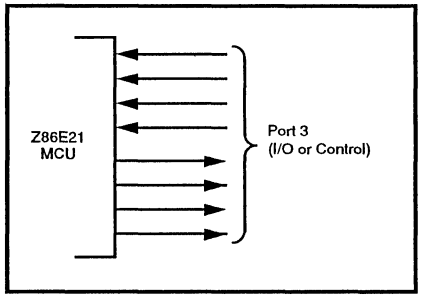


Figure 11. Port 3 Configuration

Port 3 is configured under software control to provide the following control functions: handshake for Ports 0 and 2 (/DAV and RDY); four external interrupt request signals (IRQ0-IRQ3); timer input and output signals (T_{IN} and T_{OUT}), Data Memory Select (/DM) and EPROM control signals (P30=/CE, P31=/OE, P32=EPM and P33=V_{PP}).

Table 7. Port 3 Pin Assignments

Pin	I/O	CTC1	Int.	P0 HS	P1 HS	P2 HS	UART	Ext	EPROM
P30	IN		IRQ3				Serial In		CE
P31	IN	T_{IN}	IRQ2			D/R			OE
P32	IN		IRQ0	D/R					EPM
P33	IN		IRQ1		D/R				V_{PP}
P34	OUT				R/D			DM	
P35	OUT			R/D					
P36	OUT	T_{OUT}				R/D			
P37	OUT						Serial Out		

Notes:
HS = Handshake Signals
D = Data Available
R = Ready

PIN FUNCTIONS (Continued)

Port 3 lines P30 and P37, are programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by Counter/Timer 0.

The Z86E21 automatically adds a start bit and two stop bits to transmitted data (Figure 12). Odd parity is also available as an option. Eight data bits are always transmitted, regardless of parity selection. If parity is enabled, the eighth bit is the odd parity bit. An interrupt request (IRQ4) is generated on all transmitted characters.

Received data must have a start bit, 8 data bits, and at least one stop bit. If parity is on, bit-7 of the received data is replaced by a parity error flag. Received characters generate the IRQ3 interrupt request.

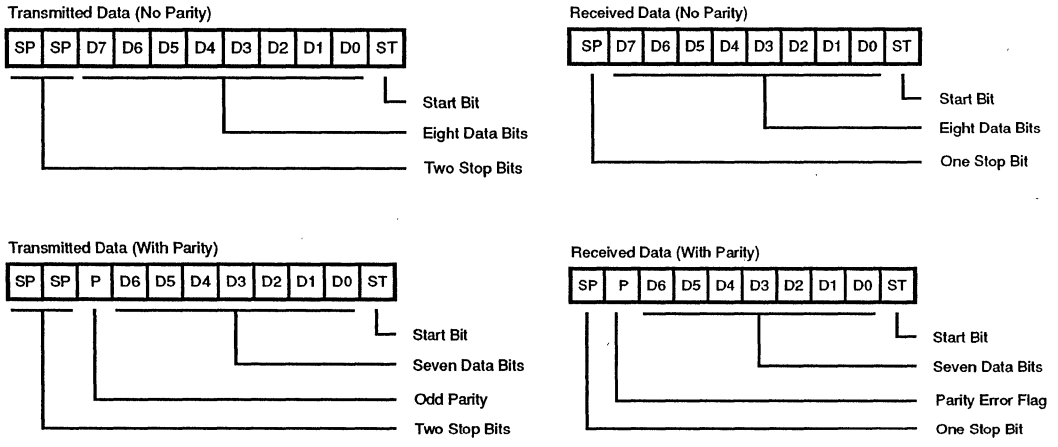


Figure 12. Serial Data Formats

Auto-Latch. The auto-latch puts valid CMOS levels on all CMOS inputs that are not externally driven. This reduces excessive supply current flow in the input buffer when it is not driven by any source.

Note: P30-P33 inputs differ from the Z86C21 because there is no clamping diode to V_{CC} due to the EPROM high voltage detection circuits. Exceeding the V_{IH} maximum specification during standard operating mode may cause the device to enter EPROM mode

ADDRESS SPACE

Program Memory. The Z86E21 can address 56 Kbytes of external program memory (Figure 13). The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts. For EPROM mode, byte 13 to byte 8191 consists of on-chip EPROM. At addresses 8192 and above, the Z86E21 executes external program memory fetches. In ROMless mode, the Z86E21 can address up to 64K bytes of program memory. Program execution begins at external location 000C (HEX) after a reset.

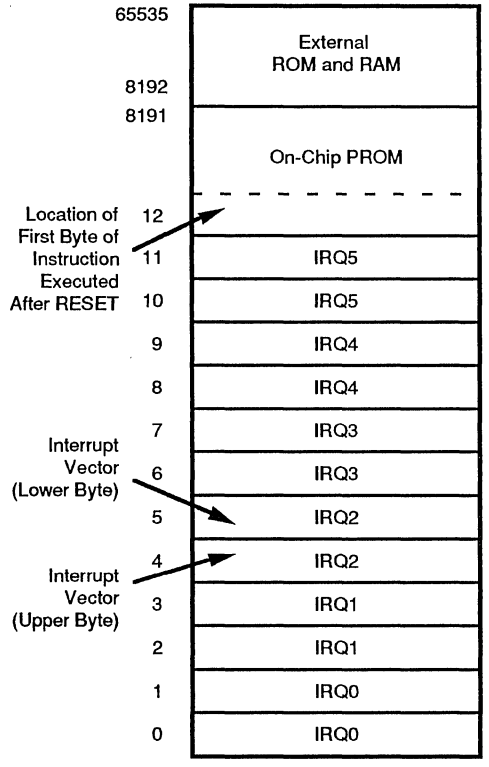


Figure 13. Program Memory Configuration

Data Memory (/DM). The EPROM version can address up to 56 Kbytes of external data memory space beginning at location 8192. The ROMless version can address up to 64 Kbytes of external data memory. External data memory may be included with, or separated from, the external program memory space. /DM, an optional I/O function that can be programmed to appear on pin P34, is used to distinguish between data and program memory space (Figure 14). The state of the /DM signal is controlled by the type instruction being executed. An LDC opcode references PROGRAM (/DM inactive) memory, and an LDE instruction references DATA (/DM active low) memory.

Register File. The Register File consists of four I/O port registers, 236 general-purpose registers and 16 control and status registers (Figure 15). The instructions can access registers directly or indirectly via an 8-bit address field. The Z86E21 also allows short 4-bit register addressing using the Register Pointer (Figure 16). In the 4-bit mode, the Register File is divided into 16 working register groups, each occupying 16 continuous locations. The Register Pointer addresses the starting location of the active working-register group.

Stack. The Z86E21 has a 16-bit Stack Pointer (R254-R255) used for external stacks that reside anywhere in the data memory for the ROMless mode, but only from 8192 to 65535 in the EPROM mode. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 236 general-purpose registers (R4-R239). The high byte of the Stack Pointer (SPH Bits 8-15) can be use as a general purpose register when using internal stack only.

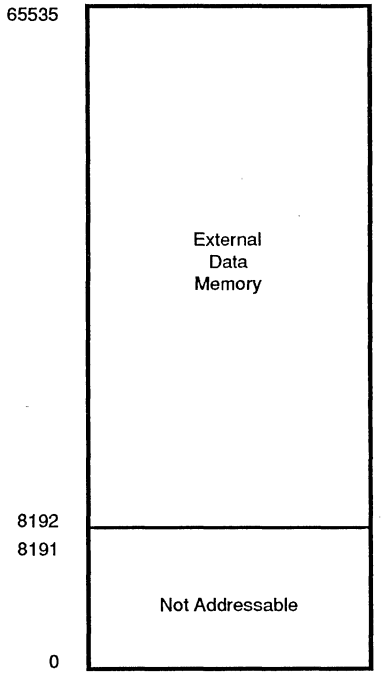


Figure 14. Data Memory Configuration

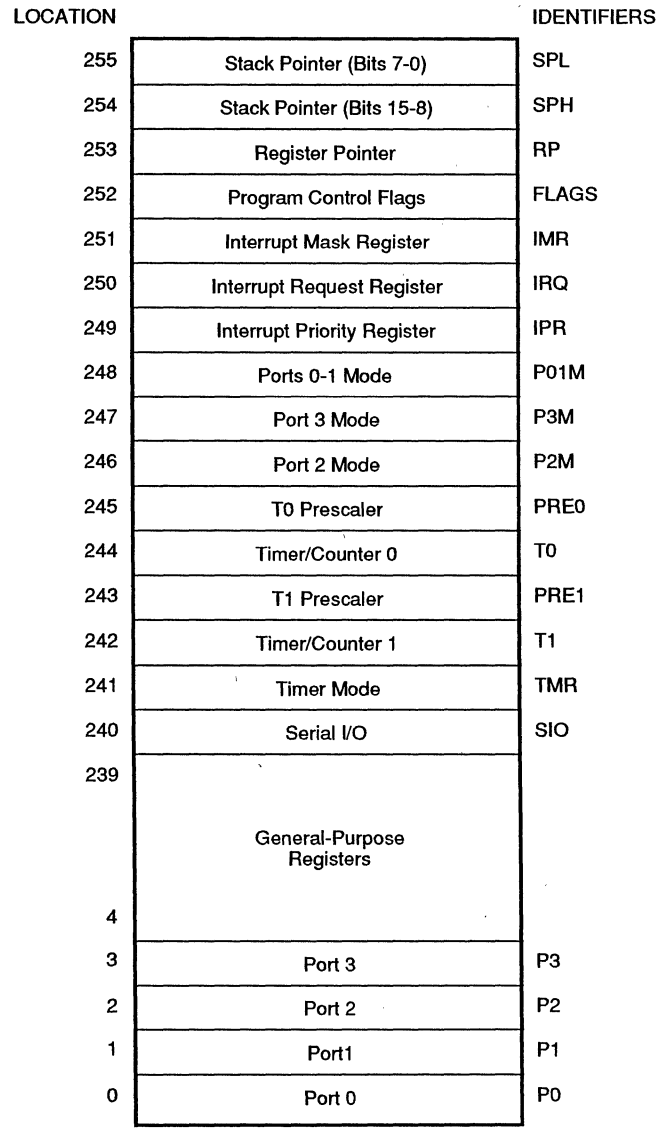


Figure 15. Register File

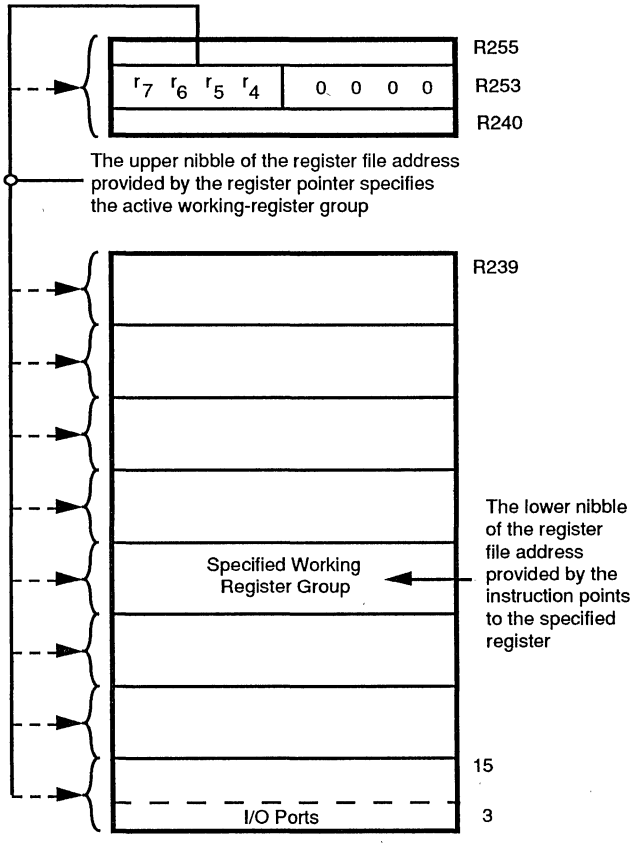


Figure 16. Register Pointer

FUNCTIONAL DESCRIPTION

Counter/Timers. There are two 8-bit programmable counter/timers (T0-T1), each driven by its own 6-bit programmable prescaler. The T1 prescaler is driven by internal or external clock sources; however, the T0 prescaler is driven by the internal clock only (Figure 17).

The 6-bit prescalers can divide the input frequency of the clock source by any integer number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When both the counters and prescalers reach the end of the count, a timer interrupt request, IRQ4 (T0) or IRQ5 (T1), is generated.

The counter is programmed to start, stop, restart to continue, or restart from the initial value. The counters can also be programmed to stop upon reaching zero (single pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode).

The counter, but not the prescalers, are read at any time without disturbing their value or count mode. The clock source for T1 is user-definable and is either the internal microprocessor clock divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input (P31) as an external clock, a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. Port 3 line P36 also serves as a timer output (T_{OUT}) through which T0, T1, or the internal clock can be output. The counter/timers are cascaded by connecting the T0 output to the input of T1.

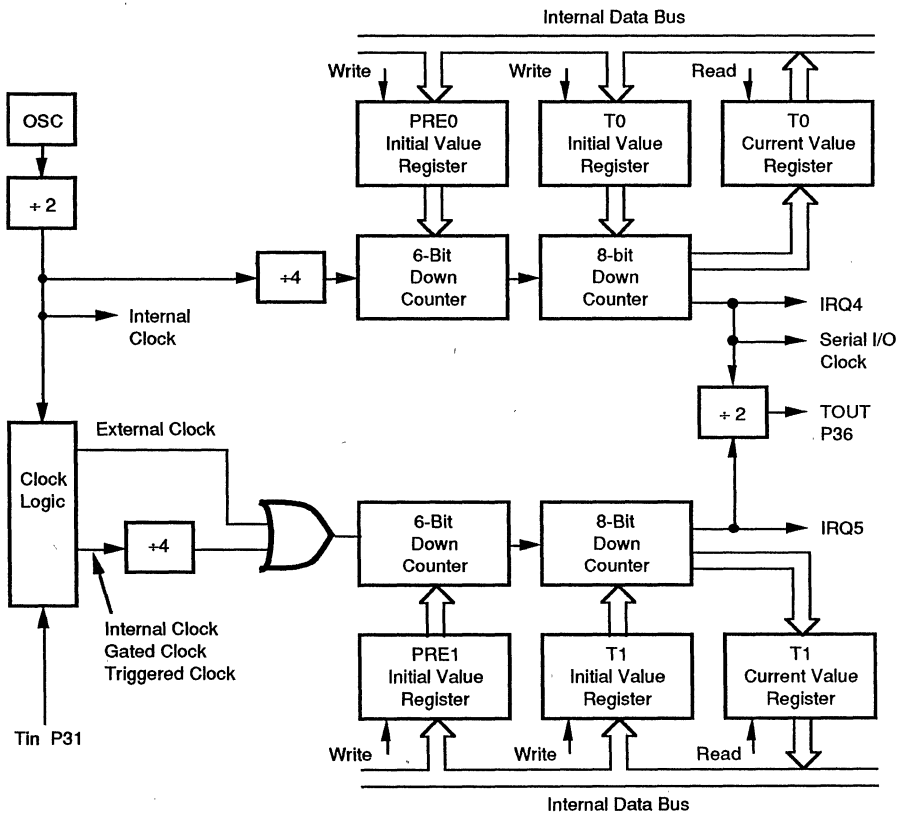


Figure 17. Counter/Timers Block Diagram

Interrupts. The Z86E21 has six different interrupts from eight different sources. The interrupts are maskable and prioritized. The 8 sources are divided as follows: four sources are claimed by Port 3 lines P30-P33, one in Serial Out, one in Serial In, and two in the counter/timers (Figure 18). The Interrupt Mask Register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All Z86E21 interrupts are vectored through locations in the program memory. When an interrupt machine cycle is activated, an interrupt request is granted. Thus, this disables all of the subsequent interrupts, saves the Program Counter and Status Flags, and then branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

To accommodate polled interrupt systems, interrupt inputs are masked and the Interrupt Request register is polled to determine which of the interrupt requests need service. Software initialed interrupts are supported by setting the appropriate bit in the Interrupt Request Register (IRQ).

Internal interrupt requests are sampled on the falling edge of the last cycle of every instruction, and the interrupt request must be valid 5TpC before the falling edge of the last clock cycle of the currently executing instruction.

For the ROMless mode, when the device samples a valid interrupt request, the next 48 (external) clock cycles are used to prioritize the interrupt, and push the two PC bytes and the FLAG register on the stack. The following nine cycles are used to fetch the interrupt vector from external memory. The first byte of the interrupt service routine is fetched beginning on the 58th TpC cycle following the internal sample point, which corresponds to the 63rd TpC cycle following the external interrupt sample point.

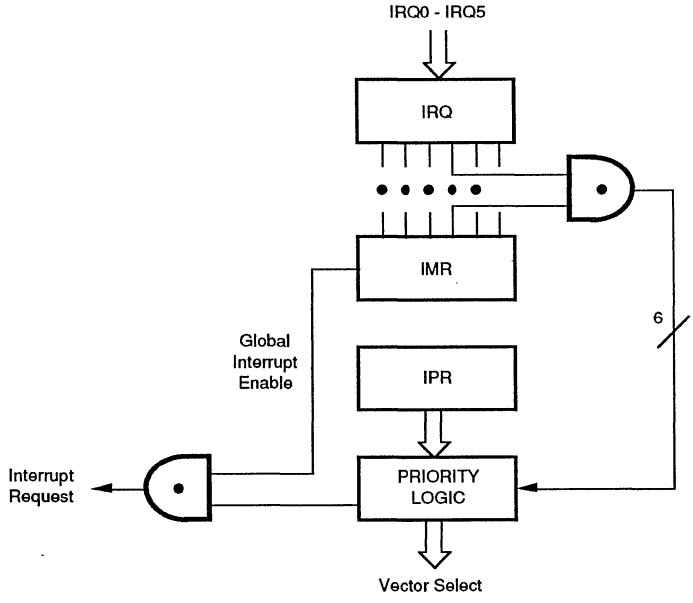


Figure 18. Interrupt Block Diagram

FUNCTIONAL DESCRIPTION (Continued)

Clock. The Z86E21 on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal, LC, ceramic resonator, or any suitable external clock source (XTAL1=Input, XTAL2=Output). The crystal should be AT cut, 1 MHz to 16 MHz max; series resistance (RS) is less than or equal to 100 Ohms. The crystal should be connected across XTAL1 and XTAL2 using the recommended capacitors (10 pF < CL < 100 pF) from each pin to ground (Figure 19).

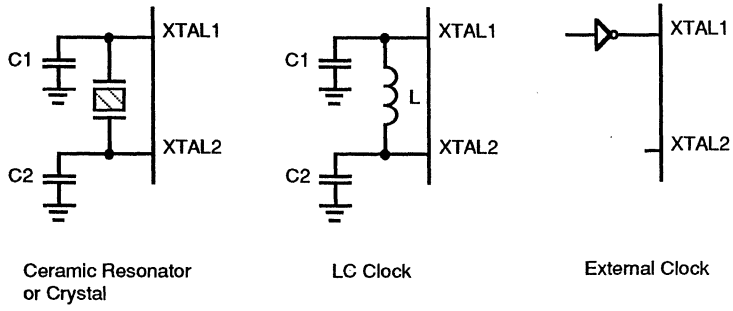


Figure 19. Oscillator Configuration

HALT. Turns off the internal CPU clock but not the XTAL oscillation. The counter/timers and external interrupts IRQ0, IRQ1, IRQ2 and IRQ3 remain active. The devices are recovered by interrupts, either externally or internally generated.

STOP. This instruction turns off the internal clock and external crystal oscillation, and reduces the standby current to 10 microamperes or less. The Stop mode is terminated by a reset, which cause the processor to restart the application program at address 000C (HEX).

In order to enter STOP (or HALT) mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. To do this, the user must execute a NOP (opcode=OFFH) immediately before the appropriate sleep instruction. i.e.:

```
FF  NOP   ; clear the pipeline
6F  STOP  ; enter STOP mode
    or
FF  NOP   ; clear the pipeline
7F  HALT  ; enter HALT mode
```

PROGRAMMING

Z86E21 User Modes

The Z86E21 uses separate AC timing cycles for the different User Modes available. Table 8 shows the Z86E21 User Modes. Table 9 shows the timing of the programming waveforms.

User MODE 1 EPROM Read

The Z86E21 EPROM read cycle is provided so that the user may read the Z86E21 as a standard 2764A EPROM. This is accomplished by driving the /EPM pin (P32) to V_H and activating /CE and /OE. /PGM remains inactive. This mode is not valid after execution of an EPROM protect cycle. Timing for the EPROM read cycle is shown in Figure 20.

User MODE 2 EPROM Program

The Z86E21 Program function conforms to the Intelligent programming algorithm. The device is programmed with V_{CC} at 6.0V and V_{PP} = 12.5V. Programming pulses are applied in 1 msec increments to a maximum of 25 pulses before proper verification. After verification, a programming pulse of three times the duration of the cycles necessary to program the device is issued to insure proper programming. After all addresses are programmed, a final data comparison is executed and the programming cycle is complete. Timing for the Z86E21 programming cycle is shown in Figure 21.

User MODE 3 EPROM Verify

The Program Verify cycle is used as part of the Intelligent programming algorithm to insure data integrity under worst-case conditions. It differs from the EPROM read cycle in that V_{PP} is active and V_{CC} must be driven to 6.0V. Timing is shown in Figure 21.

User MODES 4 AND 5 EPROM and RAM Protect

To extend program security, EPROM and RAM protect cycles are provided for the Z86E21. Execution of the EPROM protect cycle prohibits proper execution of the EPROM Read, EPROM Verify, and EPROM programming cycles. Execution of the RAM protect cycle disables accesses to the upper 128 bytes of register memory (excluding mode and configuration registers), but first the user's program must set bit-6 of the IMR (R251). Timing is shown in Figure 22.

User MODE 6 4K/8K Size Selection

The Z86E21 allows the user to select the internal ROM size. This feature is useful in that once programmed, the Z86E21 knows at which address boundary to "go external." The Z8 distinguishes internal and external fetches using the data strobe (/DS). If programmed for 4K ROM, fetch cycles include /DS beginning at location 4096 (indicating an external memory fetch). If programmed for 8K ROM, /DS remains inactive until location 8192 is reached. Once the 4K ROM size option is selected, the upper 4K of address space is unusable in the Z86E21.

The timing of the 4K/8K size selection cycle is similar to the EPROM and RAM protect cycles. Note that the 4K/8K size selection cycle requires that address 03 be indicated on the address bus during execution. Timing is shown in Figure 22.

Table 8. OTP Programming Table

Mode	V_{PP}	EPM	CE	OE	PGM	V_{CC}	ADDR	Data
1. EPROM Read	X	V_{IH}	V_{IL}	V_{IL}	V_{IH}	5.0	ADDR	DATA OUT
2. Program	V_{PP}	X	V_{IL}	V_{IH}	V_{IL}	6.0	ADDR	DATA IN
3. Program Verify	V_{PP}	X	V_{IL}	V_{IL}	V_{IH}	6.0	ADDR	DATA OUT
4. EPROM Protect	V_{PP}	V_H	V_H	V_{IH}	V_{IL}	6.0	X	X
5. RAM Protect	V_{PP}	X	V_H	V_{IH}	V_{IL}	6.0	X	X
6. 4K/8K Size Selection*	V_{PP}	V_H	V_H	V_H	V_{IL}	6.0	03	X

Notes:
V_{PP} = 12.5 ± 0.5 V
V_{CC} = 6.0 during programming
V_H = 12.5 ± 0.5 V
X = irrelevant
V_{IH} = 5.0 V
V_{IL} = 0 V
* If not programmed, the EPROM size is 8K.

Table 9. Timing of Programming Waveforms

Parameters	Name	Min	Max	Units
1	Address Setup Time	2		µs
2	Data Setup Time	2		µs
3	V_{PP} Setup	2		µs
4	V_{CC} Setup Time	2		µs
5	Chip Enable Setup Time	2		µs
6	Program Pulse Width	0.95	1.05	ms
7	Data Hold Time	2		µs
8	/OE Setup Time	2		µs
9	Data Access Time		200	ns
10	Data Output Float Time		100	ns
11	Overprogram Pulse Width	2.85	78.75	ms
12	EPM Setup Time	2		µs
13	/OE Setup Time	2		µs
14	Address to /OE Setup Time	2		µs
15	Option Program Pulse Width		78.75	ms

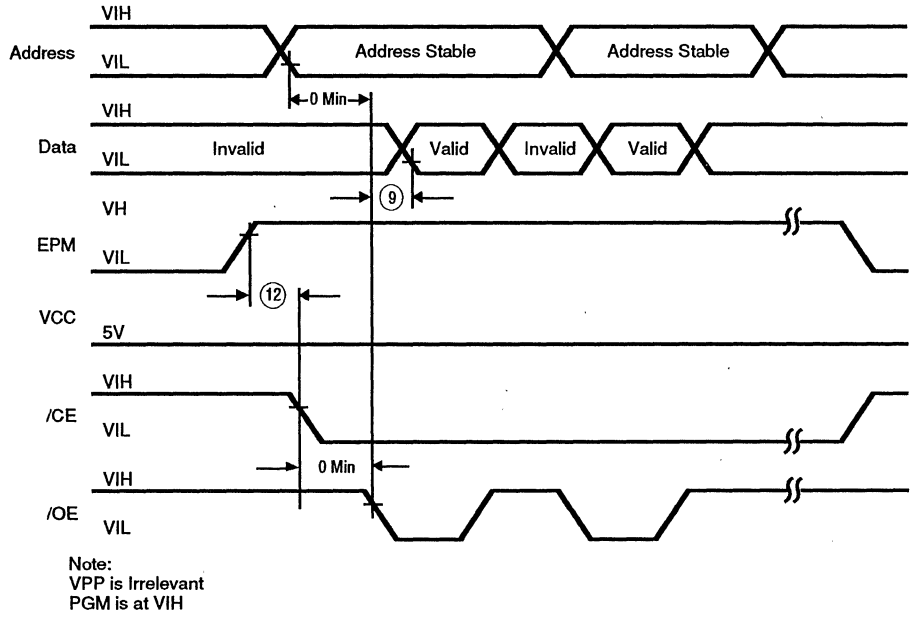


Figure 20. Programming Waveform (User Mode 1)

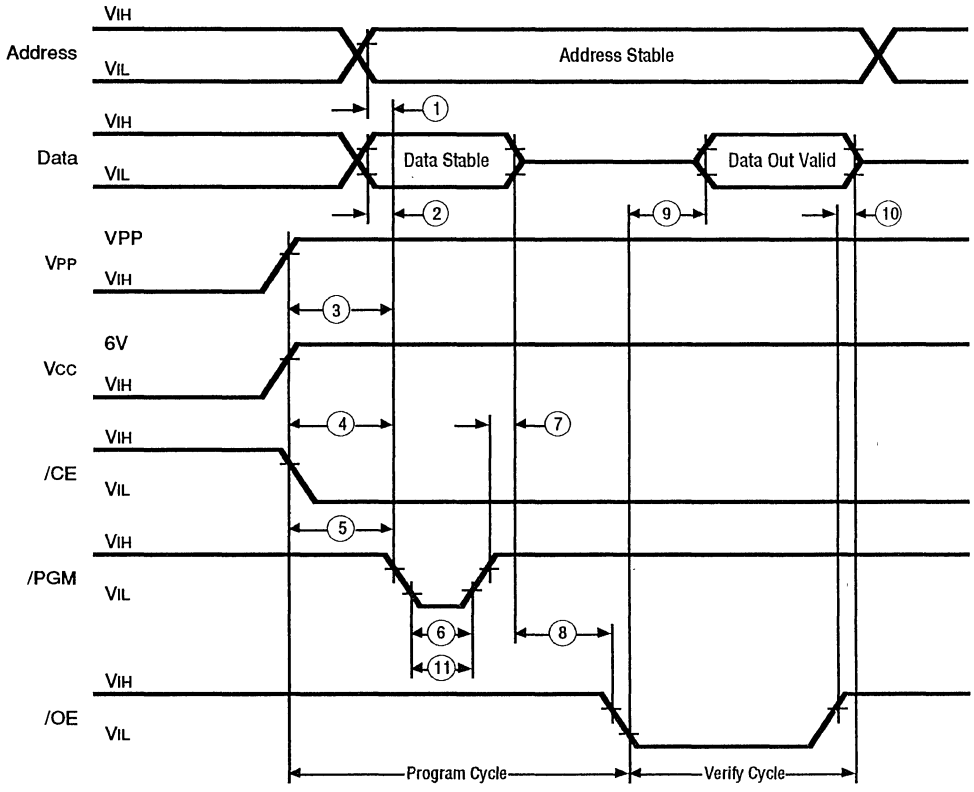


Figure 21. Programming Waveform (User Mode 2, 3)

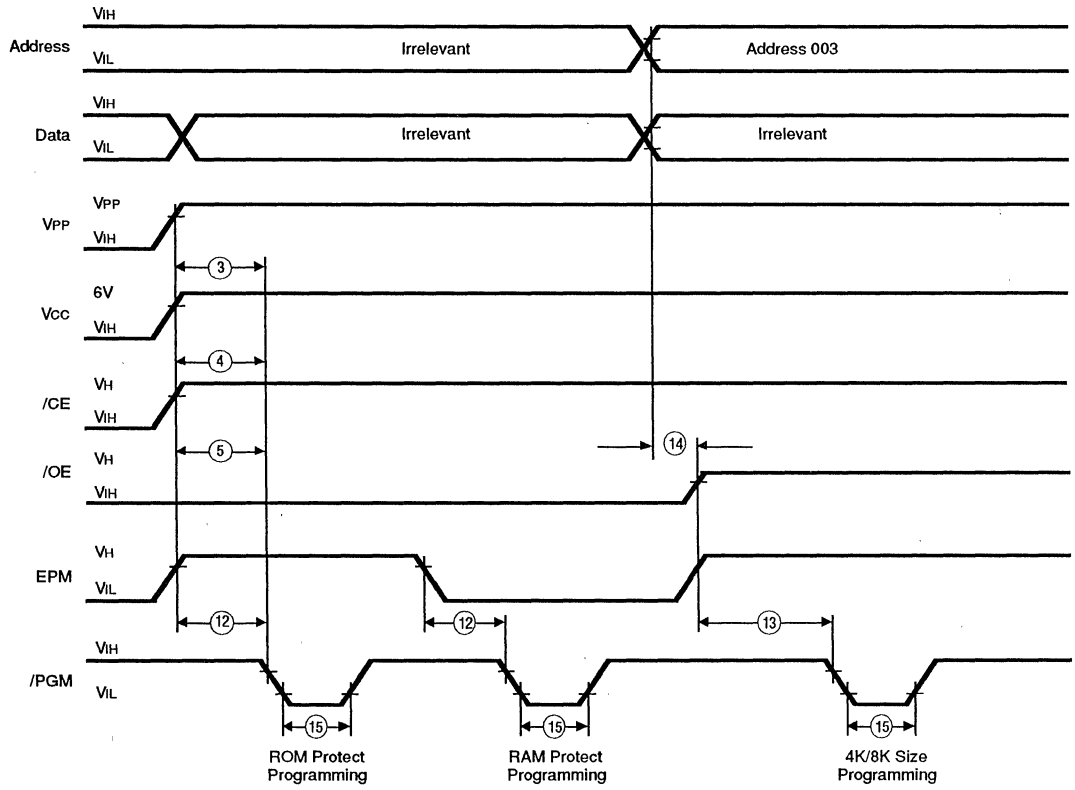


Figure 22. Programming Waveform (User Mode 4,5,6)

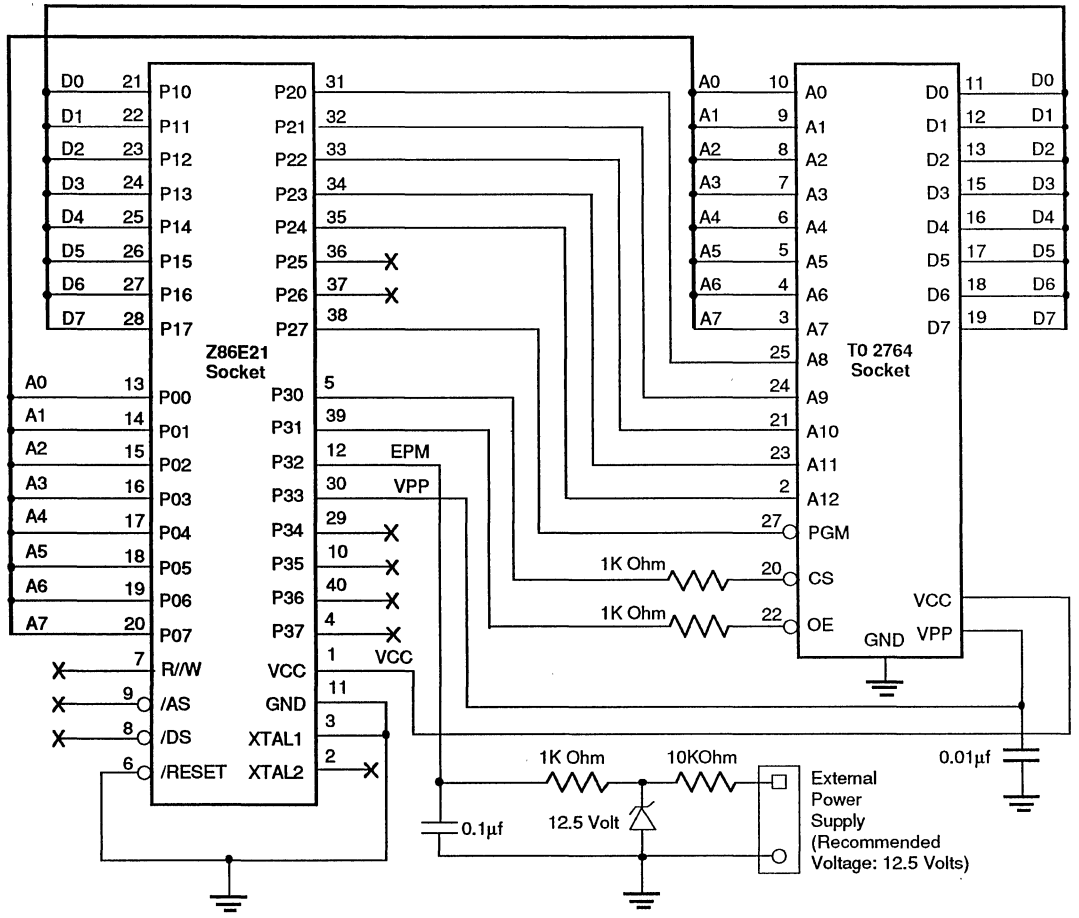


Figure 23. Z86E21 Z8 OTP Programming Adapter

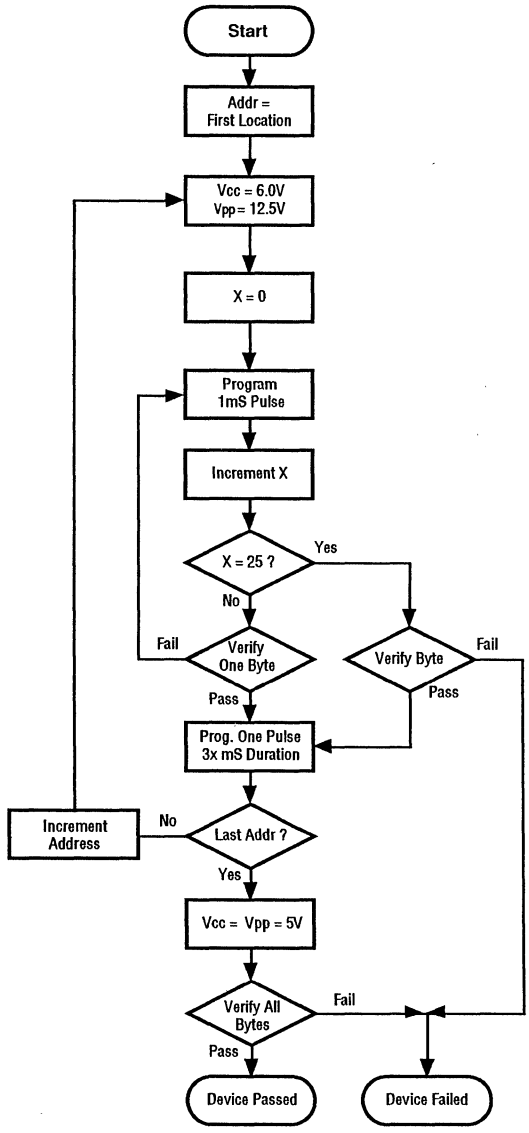


Figure 24. Intelligent Programming Flowchart

ABSOLUTE MAXIMUM RATINGS

Symbol	Description	Min	Max	Units
V_{CC}	Supply Voltage*	-0.3	+7.0	V
T_{STG}	Storage Temp	-65	+150	C
T_A	Oper Ambient Temp		†	C

Notes:
* Voltages on all pins with respect to GND. 13.0 V Maximum on P30-P33.
† See Ordering Information

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for an extended period may affect device reliability.

STANDARD TEST CONDITIONS

The characteristics listed below apply for standard test conditions as noted. All voltages are referenced to GND. Positive current flows into the referenced pin (Figure 25).

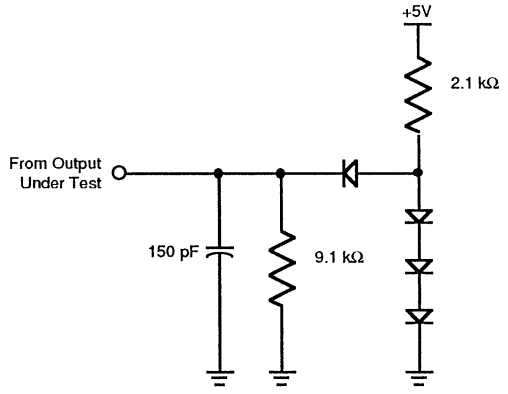


Figure 25. Test Load Diagram

DC CHARACTERISTICS

Sym	Parameter	T_A = 0°C to 70°C Min	T_A = 0°C to 70°C Max	T_A = -40°C to105°C Min	T_A = -40°C to105°C Max	Typical at 25°C	Units	Conditions
	Max Input Voltage		7		7		V	I_{IN} 250 µA
	Max Input Voltage		13		13		V	P30-P33 Only
V_{CH}	Clock Input High Voltage	3.8	V_{CC}	3.8	V_{CC}		V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.03	0.8	-0.03	0.8		V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}	2.0	V_{CC}		V	
V_{IL}	Input Low Voltage	-0.3	0.8	-0.3	0.8		V	
V_{OH}	Output High Voltage	2.4		2.4			V	I_{OH} = -2.0 mA
V_{OL}	Output Low Voltage		0.4		0.4		V	I_{OL} = +2.0 mA
V_{RH}	Reset Input High Voltage	3.8	V_{CC}	3.8	V_{CC}		V	
V_{RI}	Reset Input Low Voltage	-0.03	0.8	-0.03	0.8		V	
I_{IL}	Input Leakage	-10	10	-10	10		µA	0V V_{IN} +5.25V
I_{OL}	Output Leakage	-10	10	-10	10		µA	0V V_{IN} +5.25V
I_{IR}	Reset Input Current		-50		-50		µA	V_{CC}= +5.25V, V_{RL} = 0V
I_{CC}	Supply Current		50		50	25	mA	@ 12 MHz
			60		60	35	mA	@ 16 MHz
I_{CC1}	Standby Current		15		15	5	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 12 MHz
			20		20	10	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 16 MHz
I_{CC2}	Standby Current		20		20	5	µA	STOP Mode V_{IN} = 0V, V_{CC} @ 12 MHz
			20		20	5	µA	STOP Mode V_{IN} = 0V, V_{CC} @ 16 MHz

Notes:

I_{CC2} requires loading TMR (%F1H) with any value prior to STOP execution.
Use this sequence:

```
LD TMR,#00
NOP
STOP
```

AC CHARACTERISTICS

External I/O or Memory Read or Write Timing Diagram

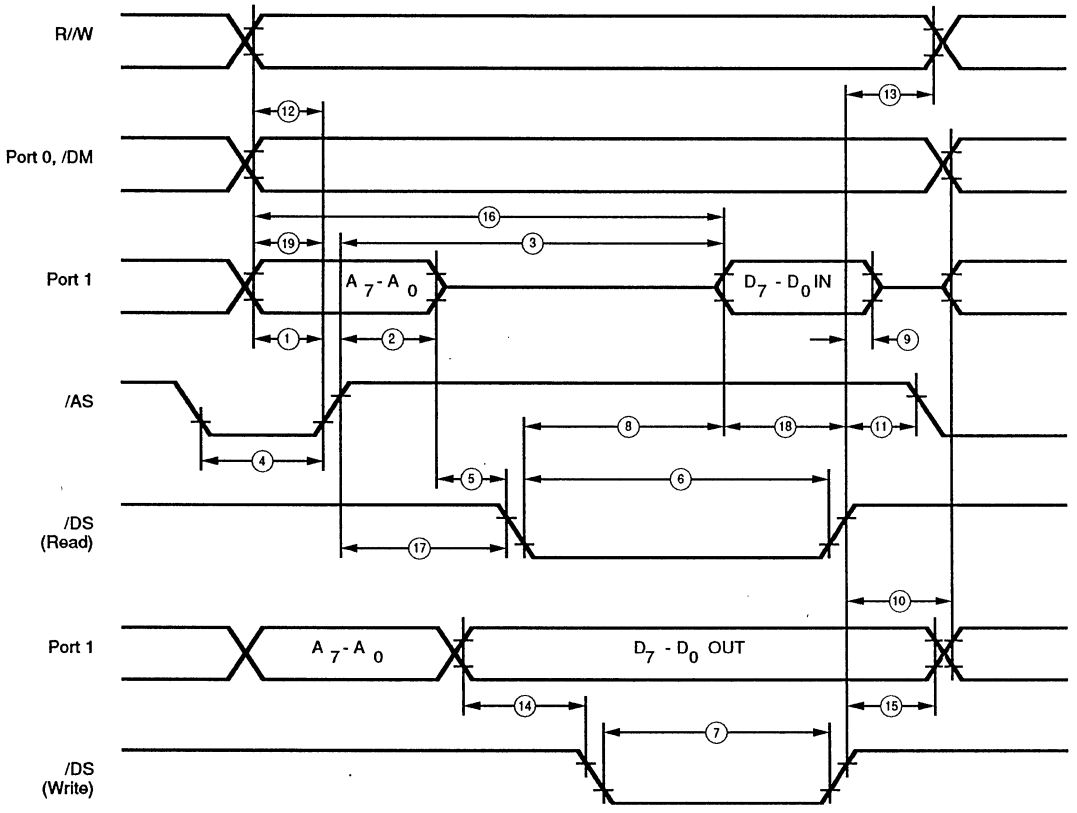


Figure 26. External I/O or Memory Read/Write Timing

AC CHARACTERISTICS

External I/O or Memory Read and Write Timing Table

No	Symbol	Parameter	T_A = 0°C to 70°C				T_A = -40°C to 105°C				Units	Notes
			12 MHz		16 MHz		12 MHz		16 MHz			
			Min	Max	Min	Max	Min	Max	Min	Max		
1	TdA(AS)	Address Valid to /AS Rise Delay	35		20		35		25		ns	[2,3]
2	TdAS(A)	/AS Rise to Address Float Delay	45		30		45		35		ns	[2,3]
3	TdAS(DR)	/AS Rise to Read Data Req'd Valid		220		180		250		180	ns	[1,2,3]
4	TwAS	/AS Low Width	55		35		55		40		ns	[2,3]
5	TdAZ(DS)	Address Float to /DS Fall	0		0		0		0		ns	
6	TwDSR	/DS (Read) Low Width	185		135		185		135		ns	[1,2,3]
7	TwDSW	/DS (Write) Low Width	110		80		110		80		ns	[1,2,3]
8	TdDSR(DR)	/DS Fall to Read Data Req'd Valid		130		75		130		75	ns	[1,2,3]
9	ThDR(DS)	Read Data to /DS Rise Hold Time	0		0		0		0		ns	[2,3]
10	TdDS(A)	/DS Rise to Address Active Delay	45		35		65		50		ns	[2,3]
11	TdDS(AS)	/DS Rise to /AS Fall Delay	55		30		45		35		ns	[2,3]
12	TdR/W(AS)	R//W Valid to /AS Rise Delay	30		20		33		25		ns	[2,3]
13	TdDS(R/W)	/DS Rise to R//W Not Valid	35		30		50		35		ns	[2,3]
14	TdDW(DSW)	Write Data Valid to /DS Fall (Write) Delay	35		25		35		25		ns	[2,3]
15	TdDS(DW)	/DS Rise to Write Data Not Valid Delay	35		30		55		35		ns	[2,3]
16	TdA(DR)	Address Valid to Read Data Req'd Valid		255		200		310		230	ns	[1,2,3]
17	TdAS(DS)	/AS Rise to /DS Fall Delay	55		40		65		45		ns	[2,3]
18	TdDI(DS)	Data Input Setup to /DS Rise	75		60		75		60		ns	[1,2,3]
19	TdDM(AS)	/DM Valid to /AS Fall Delay	50		30		50		30		ns	[2,3]

Notes:

[1] When using extended memory timing add 2 TpC.
[2] Timing numbers given are for minimum TpC.
[3] See clock cycle dependent characteristics table.

Standard Test Load

All timing references use 2.0V for a logic 1 and 0.8V for a logic 0.

Clock Dependent Formulas

Number	Symbol	Equation
1	TdA(AS)	0.40TpC + 0.32
2	TdAS(A)	0.59TpC - 3.25
3	TdAS(DR)	2.38TpC + 6.14
4	TwAS	0.66TpC - 1.65
6	TwDSR	2.33TpC - 10.56
7	TwDSW	1.27TpC + 1.67
8	TdDSR(DR)	1.97TpC - 42.5
10	TdDS(A)	0.8TpC
11	TdDS(AS)	0.59TpC - 3.14
12	TdR/W(AS)	0.4TpC
13	TdDS(R/W)	0.8TpC - 15
14	TdDW(DSW)	0.4TpC
15	TdDS(DW)	0.88TpC - 19
16	TdA(DR)	4TpC - 20
17	TdAS(DS)	0.91TpC - 10.7
18	TsDI(DS)	0.8TpC - 10
19	TdDM(AS)	0.9TpC - 26.3

AC CHARACTERISTICS
Additional Timing Diagram

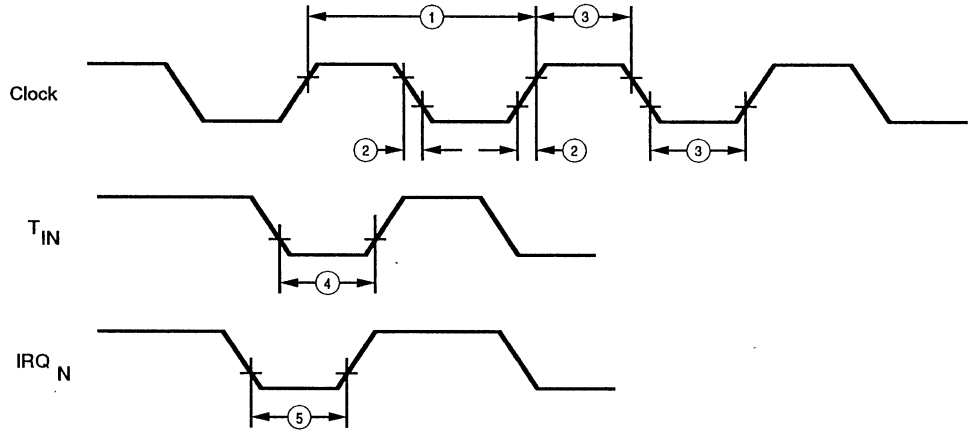


Figure 27. Additional Timing

AC CHARACTERISTICS
Additional Timing Table

No	Symbol	Parameter	T_A = 0°C to 70°C				T_A = -40°C to 105°C				Units	Notes
			12 MHz		16 MHz		12 MHz		16 MHz			
			Min	Max	Min	Max	Min	Max	Min	Max		
1	TpC	Input Clock Period	83	1000	62.5	1000	83	1000	62.5	1000	ns	[1]
2	TrC,TfC	Clock Input Rise & Fall Times		15		10		15		10	ns	[1]
3	TwC	Input Clock Width	37		21		37		21		ns	[1]
4	TwTinL	Timer Input Low Width	75		50		75		50		ns	[2]
5	TwTinH	Timer Input High Width	3TpC		3TpC		3TpC		3TpC			[2]
6	TpTin	Timer Input Period	8TpC		8TpC		8TpC		8TpC			[2]
7	TrTin,TfTin	Timer Input Rise & Fall Times	100		100		100		100		ns	[2]
8A	TwIL	Interrupt Request Input Low Times	70		50		70		50		ns	[2,4]
8B	TwIL	Interrupt Request Input Low Times	3TpC		3TpC		3TpC		3TpC			[2,5]
9	TwIH	Interrupt Request Input High Times	3TpC		3TpC		3TpC		3TpC			[2,3]

Notes:
[1] Clock timing references use 3.8V for a logic 1 and 0.8V for a logic 0.
[2] Timing references use 2.0V for a logic 1 and 0.8V for a logic 0.
[3] Interrupt references request via Port 3.
[4] Interrupt request via Port 3 (P31-P33).
[5] Interrupt request via Port 30.

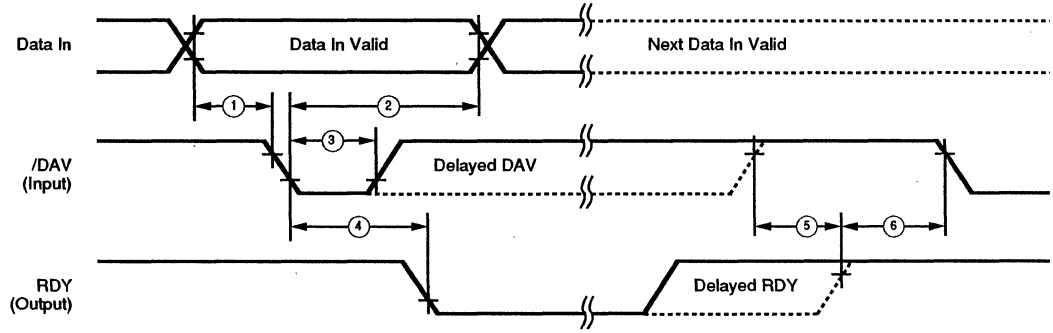


Figure 28. Input Handshake Timing

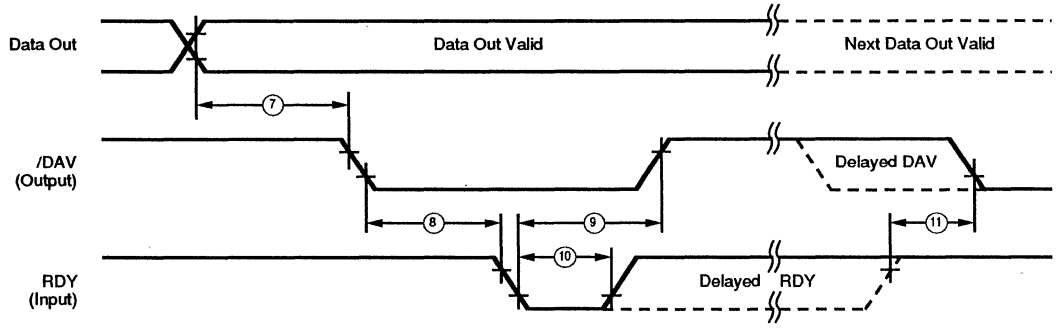


Figure 29. Output Handshake Timing

AC CHARACTERISTICS
Handshake Timing Table

No	Symbol	Parameter	T_A = 0°C to 70°C				T_A = -40°C to 105°C				Notes
			12 MHz		16 MHz		12 MHz		16 MHz		Data
			Min	Max	Min	Max	Min	Max	Min	Max	Direction
1	TsDI(DAV)	Data In Setup Time	0		0		0		0		IN
2	ThDI(DAV)	Data In Hold Time	145		145		145		145		IN
3	TwDAV	Data Available Width	110		110		110		110		IN
4	TdDAVI(RDY)	DAV Fall to RDY Fall Delay		115		115		115		115	IN
5	TdDAVId(RDY)	DAV Rise to RDY Rise Delay		115		115		115		115	IN
6	TdDO(DAV)	RDY Rise to DAV Fall Delay	0		0		0		0		IN
7	TcLDAV0(RDY)	Data Out to DAV Fall Delay		TpC		TpC		TpC		TpC	OUT
8	TcLDAV0(RDY)	DAV Fall to RDY Fall Delay	0		0		0		0		OUT
9	TdRDY0(DAV)	RDY Fall to DAV Rise Delay		115		115		115		115	OUT
10	TwRDY	RDY Width	110		110		110		110		OUT
11	TdRDY0d(DAV)	RDY Rise to DAV Fall Delay		115		115		115		115	OUT

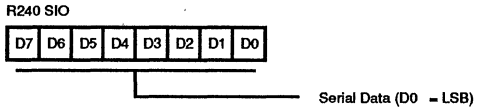


Figure 30. Serial I/O Register (F0H: Read/Write)

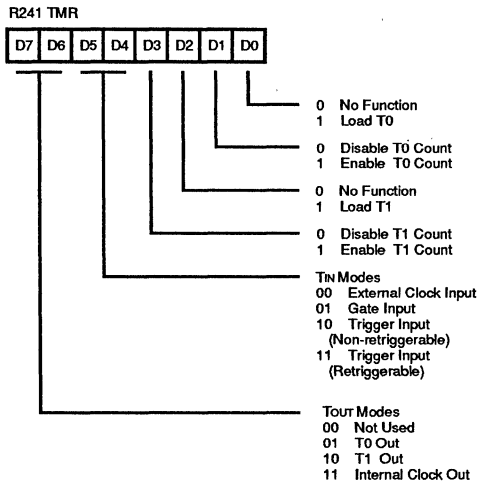


Figure 31. Timer Mode Register (F1H: Read/Write)

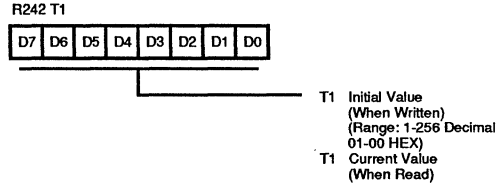


Figure 32. Counter/Timer 1 Register (F2H: Read/Write)

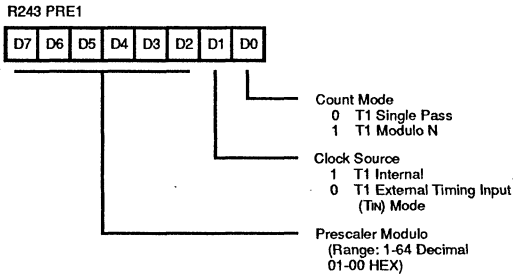


Figure 33. Prescaler 1 Register (F3H: Write Only)

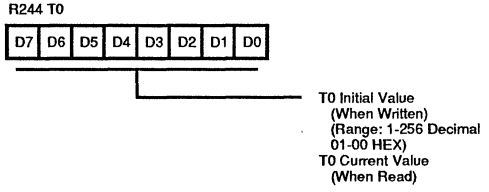


Figure 34. Counter/Timer 0 Register (F4H: Read/Write)

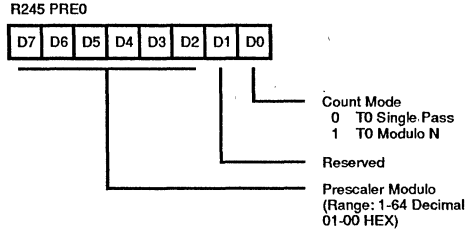


Figure 35. Prescaler 0 Register (F5H: Write Only)

R246 P2M

D7	D6	D5	D4	D3	D2	D1	D0

P2b - P27 I/O Definition
0 Defines Bit as Output
1 Defines Bit as Input

Figure 36. Port 2 Mode Register (F6H: Write Only)

R247 P3M

D7	D6	D5	D4	D3	D2	D1	D0

0 Port 2 Pull-Ups Open Drain
1 Port 2 Pull-Ups Active

Reserved

0 P32 = Input
P35 = Output
1 P32 = /DAV0/RDY0
P35 = RDY0//DAV0

00 P33 = Input
P34 = Output
01 } P33 = Input
10 } P34 = /DM
11 P33 = /DAV1/RDY1
P34 = RDY1//DAV1

0 P31 = Input (TIN)
P36 = Output (TOUT)
1 P31 = /DAV2/RDY2
P36 = RDY2//DAV2

0 P30 = Input
P37 = Output
1 P30 = Serial In
P37 = Serial Out

0 Parity Off
1 Parity On

Figure 37. Port 3 Mode Register (F7H: Write Only)

R248 P01M

D7	D6	D5	D4	D3	D2	D1	D0

P00 - P00 Mode
00 Output
01 Input
1X A11 - A8

Stack Selection
0 External
1 Internal

P17 - P10 Mode
00 Byte Output
01 Byte Input
10 AD7 - AD0
11 High-Impedance AD7 - DA0, /AS, /DS, /R//W, A11 - A8, A15 - A12, If Selected

External Memory Timing
0 Normal
1 Extended

P07 - P04 Mode
00 Output
01 Input
1X A 15 - A12

Figure 38. Port 0 and 1 Mode Register (F8H: Write Only)

R249 IPR

D7	D6	D5	D4	D3	D2	D1	D0

Interrupt Group Priority
Reserved = 000
C > A > B = 001
A > B > C = 010
A > C > B = 011
B > C > A = 100
C > B > A = 101
B > A > C = 110
Reserved = 111

IRQ1, IRQ4 Priority (Group C)
0 IRQ1 > IRQ4
1 IRQ4 > IRQ1

IRQ0, IRQ2 Priority (Group B)
0 IRQ2 > IRQ0
1 IRQ0 > IRQ2

IRQ3, IRQ5 Priority (Group A)
0 IRQ5 > IRQ3
1 IRQ3 > IRQ5

Reserved

Figure 39. Interrupt Priority Register (F9H: Write Only)

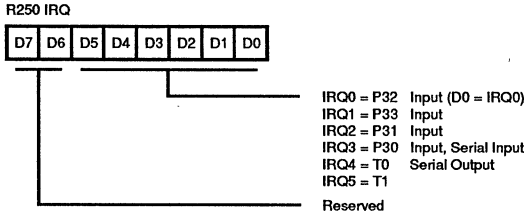


Figure 40. Interrupt Request Register (FAH: Read/Write)

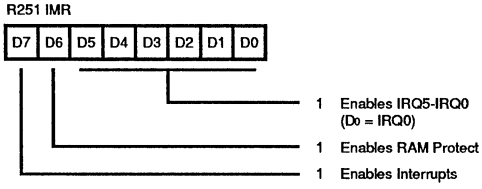


Figure 41. Interrupt Mask Register (FBH: Read/Write)

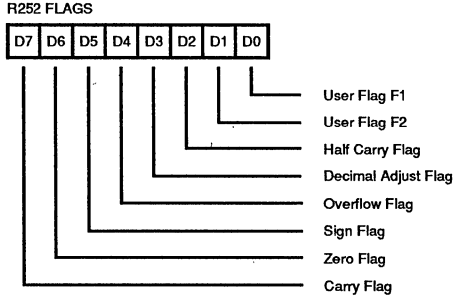


Figure 42. Flag Register (FCH: Read/Write)

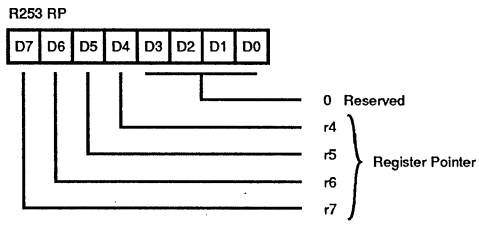


Figure 43. Register Pointer Register (FDH: Read/Write)

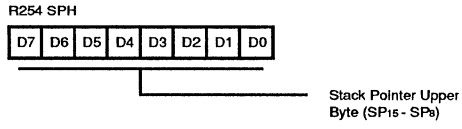


Figure 44. Stack Pointer Register (FEH: Read/Write)

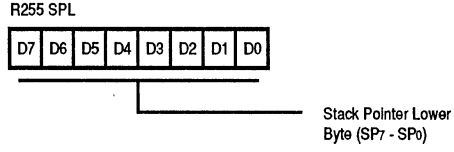


Figure 45. Stack Pointer Register (FFH: Read/Write)

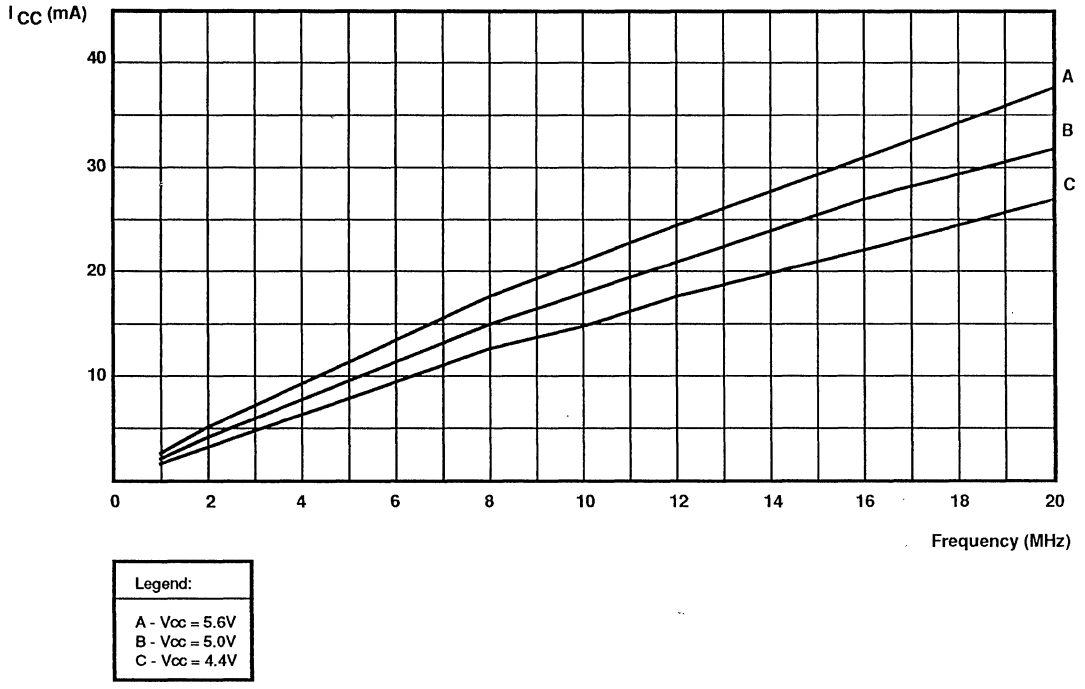


Figure 46. Typical I_{cc} vs Frequency

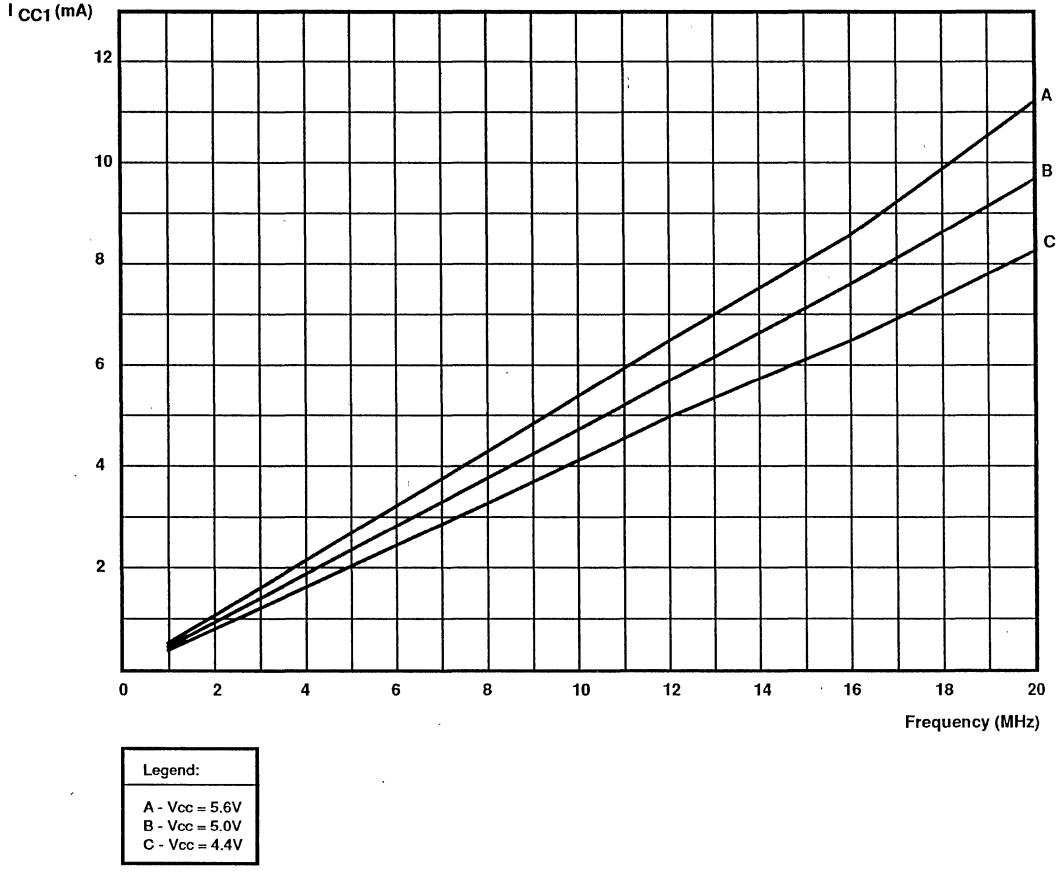


Figure 47. Typical I_{CC1} vs Frequency

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

Symbol	Meaning
IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

Symbol	Meaning
dst	Destination location or contents
src	Source location or contents
cc	Condition code
@	Indirect address prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag register (Control Register 252)
RP	Register Pointer (R253)
IMR	Interrupt mask register (R251)

Flags. Control register (R252) contains the following six flags:

Symbol	Meaning
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

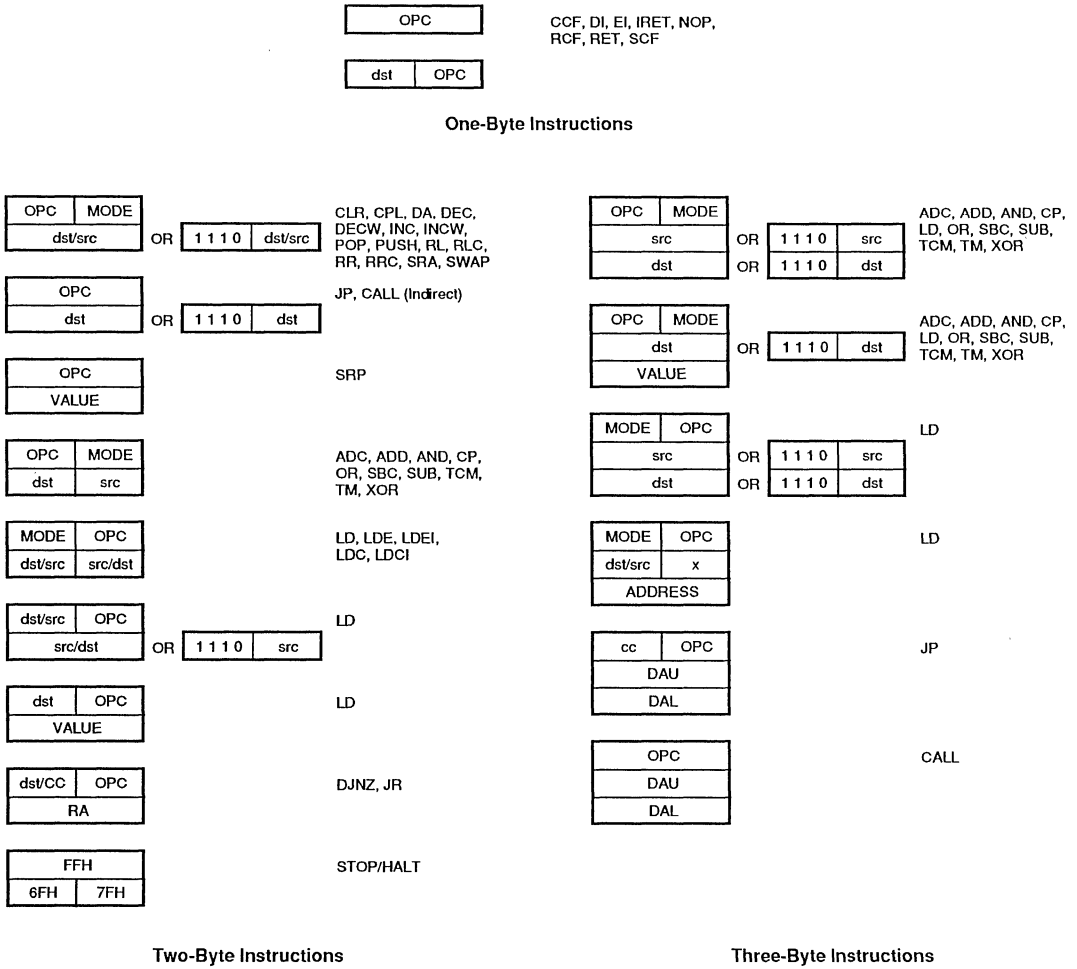
Affected flages are indicated by:

0	Clear to zero
1	Set to one
*	Set to clear according to operation
-	Unaffected
x	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000		Always True	
0111	C	Carry	C = 1
1111	NC	No Carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not Zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No Overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not Equal	Z = 0
1001	GE	Greater Than or Equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater Than	[Z OR (S XOR V)] = 0
0010	LE	Less Than or Equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned Greater Than or Equal	C = 0
0111	ULT	Unsigned Less Than	C = 1
1011	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned Less Than or Equal	(C OR Z) = 1
0000		Never True	

INSTRUCTION FORMATS



INSTRUCTION SUMMARY

Note: Assignment of a value is indicated by the symbol " ← ". For example:

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr (n)" is used to refer to bit (n) of a given operand location. For example:

dst (7)

refers to bit 7 of the destination operand.

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst, src dst←dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst←dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst←dst AND src	†		5[ ]	-	*	*	0	-	-
CALL dst SP←SP - 2 @SP←PC, PC←dst	DA IRR		D6 D4	-	-	-	-	-	-
CCF C←NOT C			EF	*	-	-	-	-	-
CLR dst dst←0	R IR		B0 B1	-	-	-	-	-	-
COM dst dst←NOT dst	R IR		60 61	-	*	*	0	-	-
CP dst, src dst - src	†		A[ ]	*	*	*	*	-	-
DA dst dst←DA dst	R IR		40 41	*	*	*	X	-	-
DEC dst dst←dst - 1	R IR		00 01	-	*	*	*	-	-
DECW dst dst←dst - 1	RR IR		80 81	-	*	*	*	-	-
DI IMR(7)←0			8F	-	-	-	-	-	-
DJNZr, dst r←r - 1 if r ≠ 0 PC←PC + dst Range: +127, -128	RA		rA r = 0 - F	-	-	-	-	-	-
EI IMR(7)←1			9F	-	-	-	-	-	-
HALT			7F	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
INC dst dst←dst + 1	r R IR		rE r = 0 - F 20 21	-	*	*	*	-	-
INCW dst dst←dst + 1	RR IR		A0 A1	-	*	*	*	-	-
IRET FLAGS←@SP; SP←SP + 1 PC←@SP; SP←SP + 2; IMR(7)←1			BF	*	*	*	*	*	*
JP cc, dst if cc is true PC←dst	DA IRR		cD c = 0 - F 30	-	-	-	-	-	-
JR cc, dst if cc is true, PC←PC + dst Range: +127, -128	RA		cB c = 0 - F	-	-	-	-	-	-
LD dst, src dst←src	r r R r X r r Ir R R IR IR	Im R r X r Ir r R IR IM IM R	rC r8 r9 r = 0 - F C7 D7 E3 F3 E4 E5 E6 E7 F5	-	-	-	-	-	-
LDC dst, src	r	Irr	C2	-	-	-	-	-	-
LDCI dst, src dst←src r←r +1; rr←rr + 1	Ir	Irr	C3	-	-	-	-	-	-

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
NOP			FF	-	-	-	-	-	-
OR dst, src dst←dst OR src	†		4[ ]	-	*	*	0	-	-
POP dst dst←@SP; SP←SP + 1	R IR		50 51	-	-	-	-	-	-
PUSH src SP←SP - 1; @SP←src		R IR	70 71	-	-	-	-	-	-
RCF C←0			CF	0	-	-	-	-	-
RET PC←@SP; SP←SP + 2			AF	-	-	-	-	-	-
RL dst	R IR		90 91	*	*	*	*	-	-
RLC dst	R IR		10 11	*	*	*	*	-	-
RR dst	R IR		E0 E1	*	*	*	*	-	-
RRC dst	R IR		C0 C1	*	*	*	*	-	-
SBC dst, src dst←dst←src←C	†		3[ ]	*	*	*	*	1	*
SCF C←1			DF	1	-	-	-	-	-
SRA dst	R IR		D0 D1	*	*	*	0	-	-
SRP src RP←src		Im	31	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
STOP			6F	-	-	-	-	-	-
SUB dst, src dst←dst←src	†		2[ ]	*	*	*	*	1	*
SWAP dst	R IR		F0 F1	X	*	*	X	-	-
TCM dst, src (NOT dst) AND src	†		6[ ]	-	*	*	0	-	-
TM dst, src dst AND src	†		7[ ]	-	*	*	0	-	-
XOR dst, src dst←dst XOR src	†		B[ ]	-	*	*	0	-	-

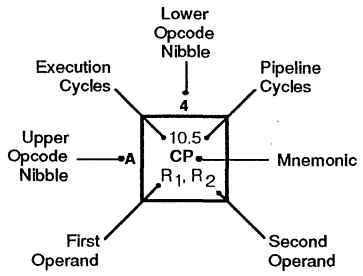
† These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address Mode dst	src	Lower Opcode Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

OPCODE MAP

Upper Nibble (Hex)	Lower Nibble (Hex) 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								6.0 **STOP**
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								7.0 **HALT**
8	10.5 **DECW** RR1	10.5 **DECW** IR1	12.0 **LDE** r1, Irr2	18.0 **LDEI** Ir1, Irr2												6.1 **DI**
9	6.5 **RL** R1	6.5 **RL** IR1	12.0 **LDE** r2, Irr1	18.0 **LDEI** Ir2, Irr1												6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14.0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2				10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1	12.0 **LDC** r2, Irr1	18.0 **LDCI** Ir2, Irr1	20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6.5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2			3					2					3	1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_2 = Dst address
R_1 or r_2 = Src address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank are not defined.

* 2-byte instruction appears as a 3-byte instruction

PRODUCT SPECIFICATION

Z86C30

CMOS Z8® 8-BIT MICROCONTROLLER

FEATURES

- 8-bit CMOS microcontroller, 28-pin DIP
- Low cost
- 3.0 to 5.5 volt operating range
- Low power consumption - 50 mW (Typical)
- Fast instruction pointer - 1.0 microsecond @ 12 MHz
- Two standby modes (STOP and HALT)
- 24 input/output lines (two with comparator inputs)
- All digital inputs CMOS levels, Schmitt-triggered
- 4 Kbytes of ROM
- 236 bytes of RAM
- Two Expanded Register File control registers
- Two programmable 8-bit Counter/Timers
- 6-bit programmable prescaler on each Counter/Timer
- Six vectored, priority interrupts from six different sources
- Clock speeds 8 and 12 MHz
- "Brown-Out" protection
- Watchdog/Power-On Reset Timer
- Two comparators with programmable interrupt polarity
- On-chip oscillator that accepts a crystal, ceramic resonator, LC, RC or external clock drive.
- ROM and RAM protect

GENERAL DESCRIPTION

The Z86C30 CCP™ Consumer Controller Processor introduces a new level of sophistication to single-chip architecture. The Z86C30 is a member of the Z8 single-chip microcontroller family with 4 Kbytes of ROM and 236 bytes of RAM. The device is housed in a 28-pin DIP, and is manufactured in CMOS technology. Zilog's CMOS microcontroller offers fast execution, efficient use of memory, sophisticated interrupts, input/output bit manipulation capabilities, and easy hardware/software system expansion along with low cost and low power consumption.

The Z86C30 architecture is characterized by Zilog's 8-bit microcontroller core with an Expanded Register File to allow easy access to register mapped peripheral and I/O circuits. The CCP offers a flexible I/O scheme, an efficient register and address space structure, and a number of ancillary features that are useful in many industrial, automotive, and advanced scientific applications.

The device applications demand powerful I/O capabilities. The CCP fulfills this with 24 pins dedicated to input and output. These lines are grouped into three ports, eight lines per port, and are configurable under software control to provide timing, status signals, and parallel I/O with or without handshake.

There are three basic address spaces available to support this wide range of configurations: Program Memory, Register File, and Expanded Register File. The Register File is composed of 236 bytes of general purpose registers, three I/O port registers and 15 control and status registers. The Expanded Register File consists of two Control registers.

GENERAL DESCRIPTION (Continued)

To unburden the program from coping with the real-time problems such as counting/timing and input/output data communication, the Z86C30 offers two on-chip counter/timers with a large number of user selectable modes, and on-board comparators to process analog signals with a common reference voltage (Figure 1).

Note: All Signals with a preceding front slash, "/", are active Low, e.g.: B//W (WORD is active Low); /B/W (BYTE is active Low, only); /N//S (NORMAL and SYSTEM are both active Low).

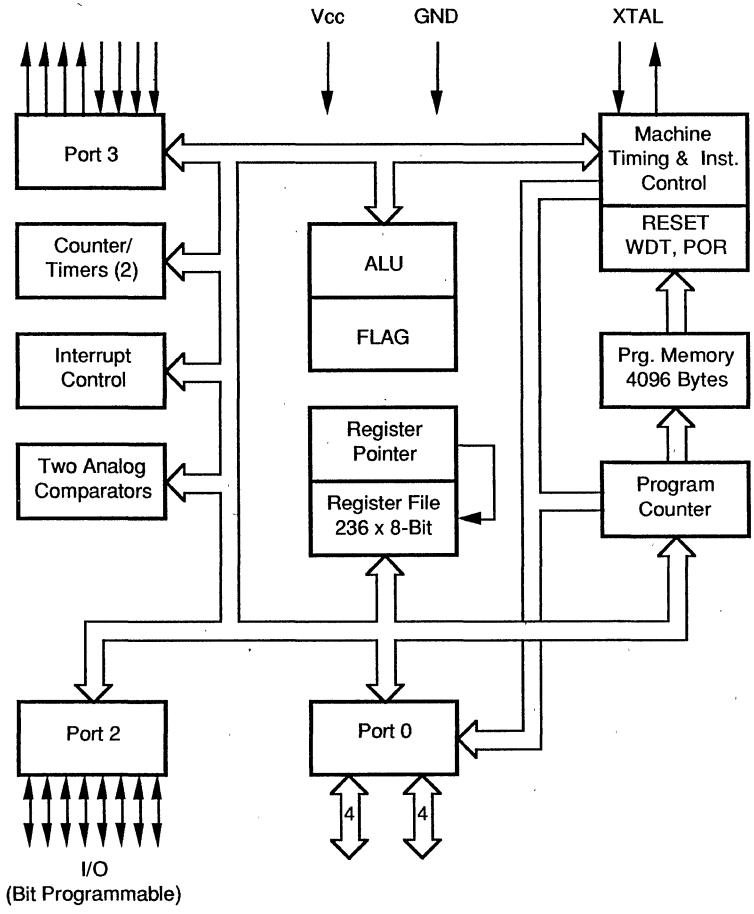


Figure 1. Functional Block Diagram

PIN DESCRIPTION

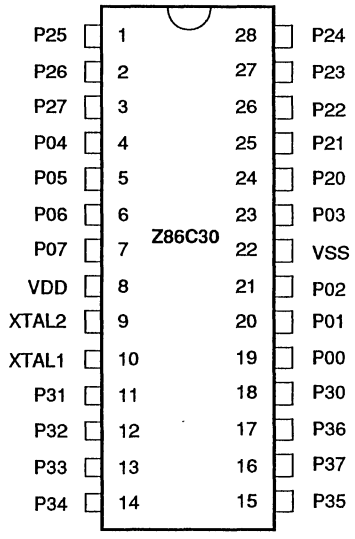


Figure 2. Pin Configuration

Table 1. Pin Identification

Pin #	Symbol	Function	Direction
1-3	P25-7	Port 2 pin 5,6,7	In/Output
4-7	P04-7	Port 0 pin 4,5,6,7	In/Output
8	V_{CC}	Power Supply	Input
9	XTAL2	Crystal Oscillator	Output
10	XTAL1	Crystal Oscillator	Input
11-13	P31-3	Port 3 pin 1,2,3	Fixed Input
14-15	P34-5	Port 3 pin 4,5	Fixed Output
16	P37	Port 3 pin 7	Fixed Output
17	P36	Port 3 pin 6	Fixed Output
18	P30	Port 3 pin 0	Fixed Input
19-21	P00-2	Port 0 pin 0,1,2	In/Output
22	GND	Ground, V_{SS}	Input
23	P03	Port 0 pin 3	In/Output
24-28	P20-4	Port 2 pin 0,1,2,3,4	In/Output

Note: Power connections follow Conventional descriptions below

Connection	Circuit	Device
Power	V_{CC}	V_{DD}
Ground	GND	V_{SS}

XTAL1. *Crystal 1* (time-based input). This pin connects a parallel-resonant crystal, ceramic resonator, LC or RC network or external single-phase clock to the on-chip oscillator input.

XTAL2. *Crystal 2* (time-based output). This pin connects a parallel-resonant crystal, ceramic resonator, LC or RC network to the on-chip oscillator output.

Port 0 P00-P07. Port 0 is an 8-bit, bidirectional, CMOS compatible I/O port. These eight I/O lines can be nibble programmed as P00-P03 input/output and P04-P07 input/output, separately. All input buffers are Schmitt-triggered and output drivers are push-pull. It can also be used as a handshake I/O port.

Port 3 lines P32 and P35 are used as handshake control lines. The handshake direction is determined by the configuration (input or output) assigned to port 0's upper nibble. The lower nibble must have the same direction as the upper nibble (Figure 3).

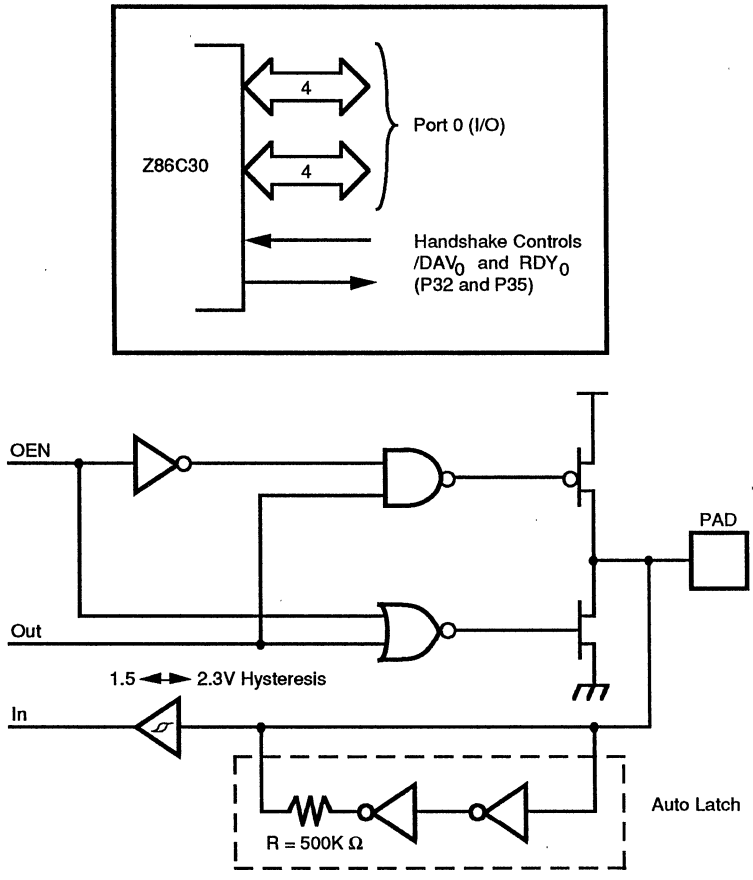


Figure 3. Port 0 Configuration

Port 2 P20-P27. Port 2 is an 8-bit, bidirectional, CMOS compatible I/O port. These eight I/O lines can be configured under software control as an input or output, independently. Input buffers are Schmitt-triggered. Bits programmed as outputs may be globally programmed as either push-pull or open drain. When used as an I/O port, Port 2 may be placed under handshake control. In this configuration, Port 3 lines P31 and P36 are used as the handshake control lines. The handshake signal assignment for Port 3 lines P31 and P36 is dictated by the direction (input or output) assigned to bit 7, Port 2 (Figure 4).

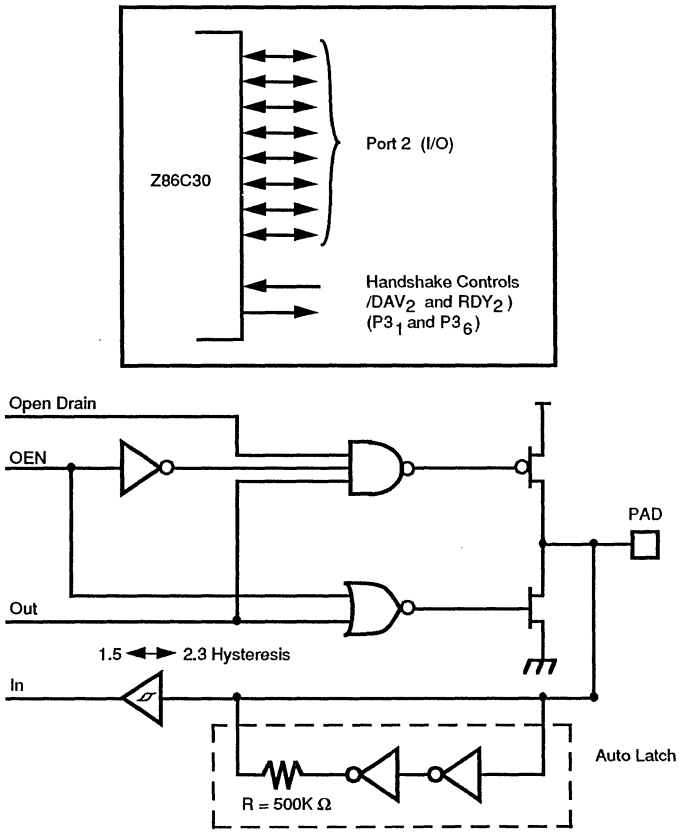


Figure 4. Port 2 Configuration

Auto-Latch. The Auto-Latch puts valid CMOS levels on all CMOS inputs that are not externally driven. This reduces excessive supply current flow in the input buffer when not being driven by any source (Figure 46).

Port 3 P30-P37. Port 3 is an 8-bit, CMOS compatible port with four fixed input and four fixed output. Port 3 consists of four fixed inputs (P30-P33) and four fixed outputs (P34-P37), and can be configured under software for interrupt and port handshake functions. Port 3, pin 0 input is Schmitt-triggered. Pins P31, P32 and P33 are standard CMOS inputs and the outputs are push-pull. Two on-board comparators can process analog signals on P31 and P32 with reference to the voltage on P33. The analog function is enabled by programming the Port 3 Mode Register (bit 1). P30 and P33 are falling edge interrupt inputs. P31 and P32 are programmable as falling, rising, or both edge triggered interrupts (IRQ register bits 6 and 7). P33 is the comparator reference voltage input.

Access to Counter/Timer 1 is made through P31 (T_{IN} and P36 (T_{OUT}). Handshake lines for Ports 0 and 2 are available on P3 pin 1 through 6 (Figure 5).

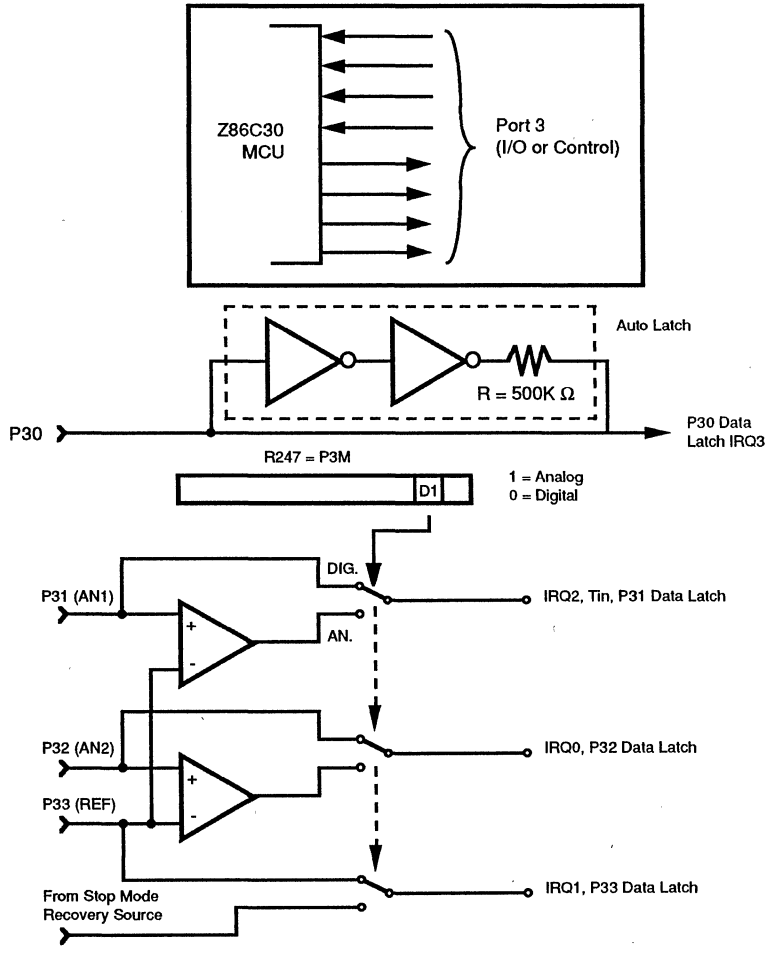


Figure 5. Port 3 Configuration

Table 2. Pin Assignments

Pin	I/O	CTC1	AN In	Int.	P0 HS	P2 HS
P30	IN			IRQ3		
P31	IN	Tin	AN1	IRQ2		D/R
P32	IN		AN2	IRQ0	D/R	
P33	IN		REF	IRQ1		
P34	OUT					
P35	OUT				R/D	
P36	OUT	Tout				R/D
P37	OUT					

Notes:
HS = Handshake Signals
D = DAV
R = RDY

Comparator Inputs. Port 3 Pin P31 and P32 each have a comparator front end. The comparator reference voltage Pin P33 is common to both comparators. In analog mode, the P31 and P32 are the positive inputs to the comparators and P33 is the reference voltage supplied to both comparators. In digital mode, pin P33 can be used as a P33 register input or IRQ1 source.

FUNCTIONAL DESCRIPTION

The Z8 CCP incorporates special functions to enhance the Z8's application in industrial, scientific research and advanced technologies applications.

Reset. The device is reset in one of the following conditions:

- Power-On Reset
- Watch-Dog Timer
- STOP Mode Recovery Source
- Brown-Out Recovery

The Z86C30 does not re-initialize WDTMR, SMR, P2M, and P3M registers to their reset values on a STOP Mode Recovery operation.

Program Memory. The Z86C30 can address up to 4K bytes of internal program memory (Figure 6). The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six, 16-bit vectors that correspond to the six available interrupts. Byte 13 to byte 4095 consists of on-chip mask programmed ROM.

The 4K bytes of program memory is mask programmable. **A ROM protect feature prevents "dumping" of the ROM contents by inhibiting execution of LDC and LDCI instructions to program memory in ALL modes.**

The ROM protect option is mask-programmable, and is selected by the customer when the ROM code is submitted.

Expanded Register File. The register file has been expanded to allow for additional system control registers, mapping of additional peripheral devices and input/output ports into the register address area. The Z8 register address space R0 through R15 is implemented as 16 groups of 16 registers per group (Figure 7). These register groups are known as the ERF (Expanded Register File).

Bits 3:0 of the Register Pointer (RP) select the active ERF group. Bits 7:4 of register RP select the working register group (Figure 8). Two system configuration registers reside in the Expanded Register File at bank F. The rest of the Expanded Register is not physically implemented and is open for future expansion.

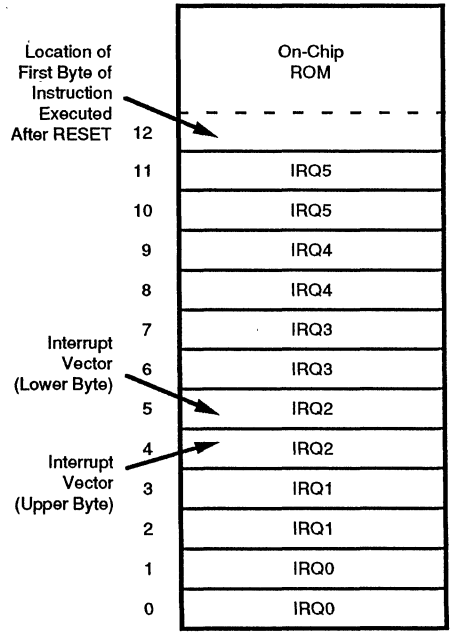


Figure 6. Program Memory Map

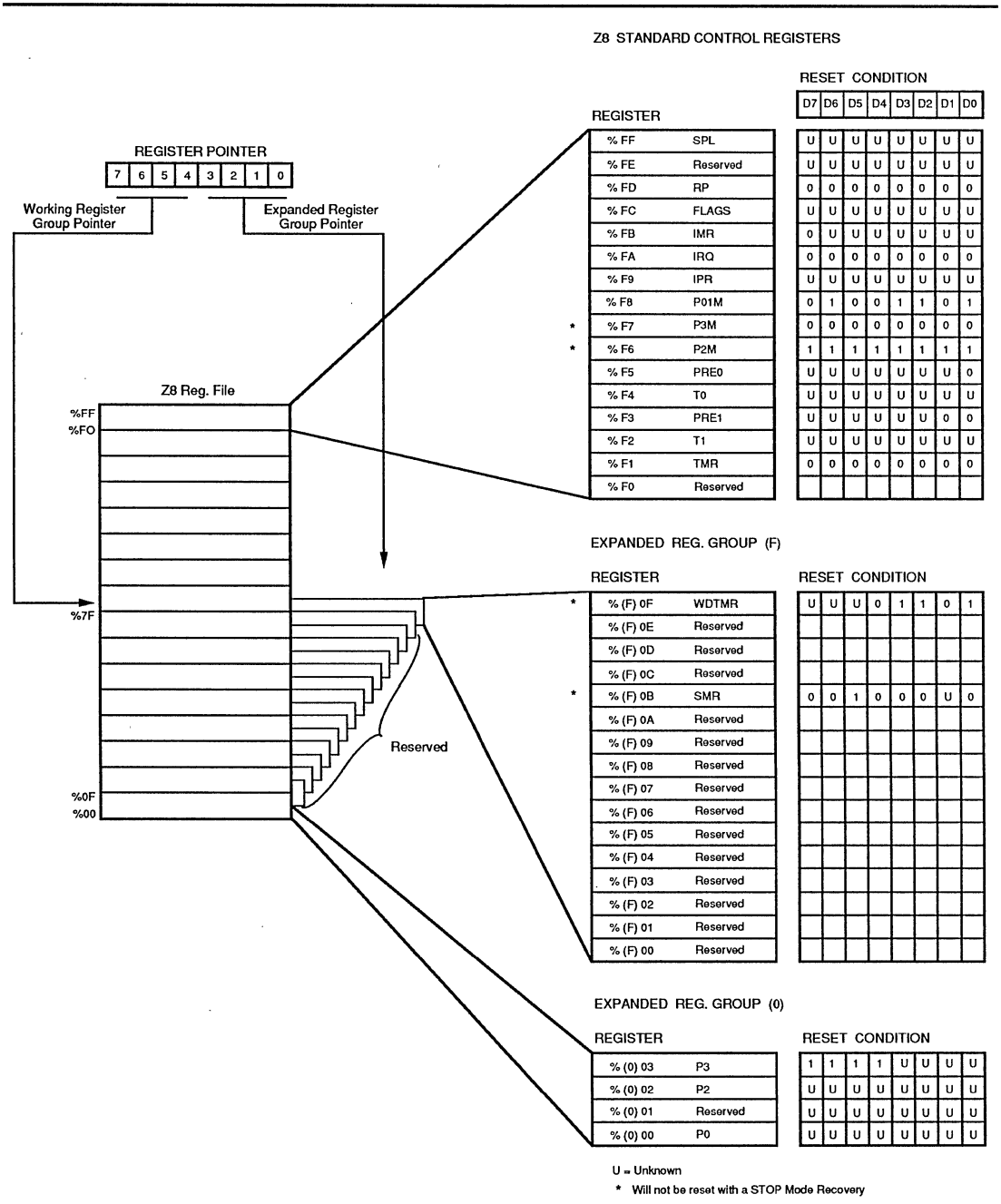


Figure 7. Expanded Register File Architecture

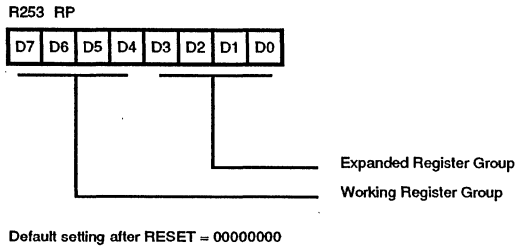


Figure 8. Register Pointer Register

Register File. The register file consists of three I/O port registers, 236 general purpose registers and 15 control and status registers (R0-R3, R4-239 and R240-R255, respectively), and two system configuration registers in the expanded register group (See Figure 7). The instructions can access registers directly or indirectly via an 8-bit address field. This allows a short 4-bit register address using the Register Pointer (Figure 9). In the 4-bit mode, the register file is divided into 16 working register groups, each occupying 16 continuous locations. The Register Pointer addresses the starting location of the active working register group.

Note: Register Bank E0-EF can only be accessed through working register and indirect addressing modes.

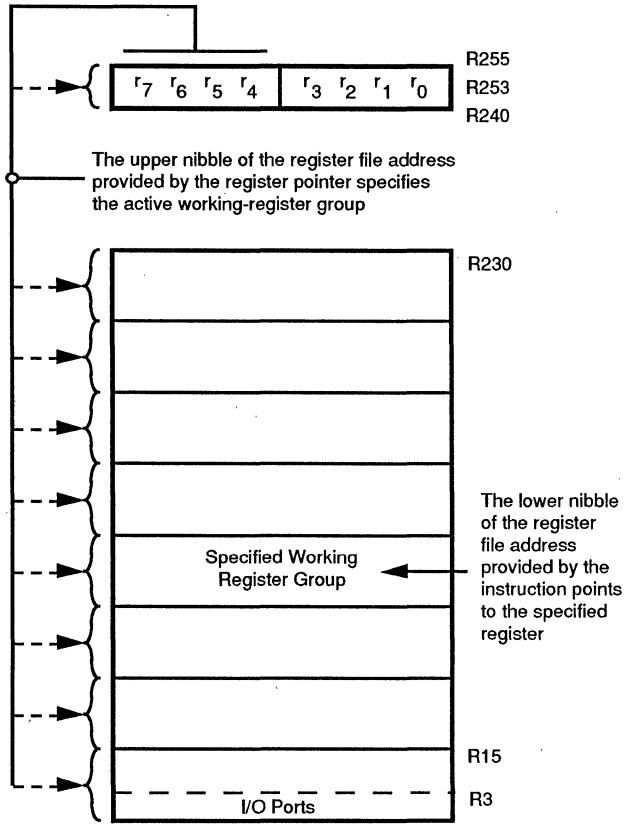


Figure 9. Register Pointer

RAM Protect. The upper portion of the RAM's address spaces %7F to %EF (excluding the control registers) can be protected from reading and writing. The RAM Protect bit option is mask-programmable and is selected by the customer when the ROM code is submitted. After the mask option is selected, the user can activate from the internal ROM code to turn off/on the RAM Protect by loading a bit D6 in the IMR register to either a 0 or a 1, respectively. A 1 in D6 indicates RAM Protect enabled.

Stack. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 256 general purpose registers.

Counter/Timers. There are two 8-bit programmable counter/timers (T0-T1), each driven by its own 6-bit programmable prescaler. The T1 prescaler can be driven by internal or external clock sources; however, the T0 prescaler is driven by the internal clock only (Figure 10).

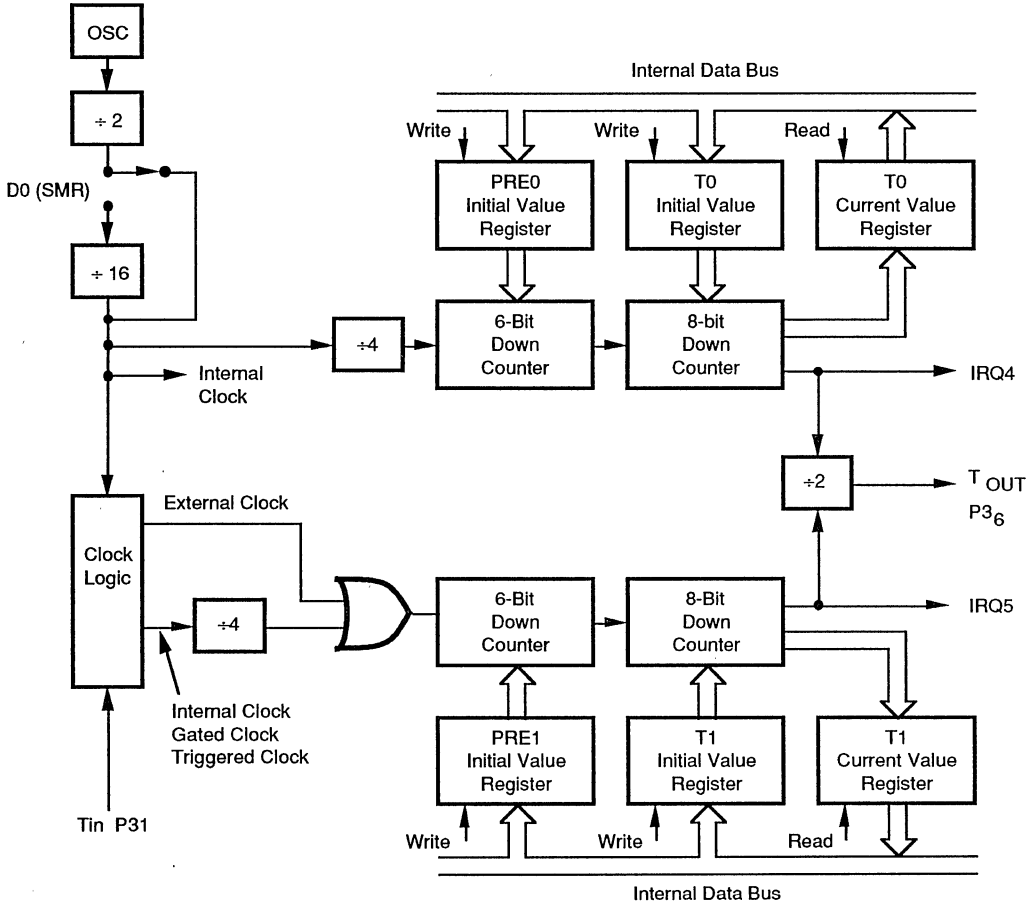


Figure 10. Counter/Timer Block Diagram

The 6-bit prescalers can divide the input frequency of the clock source by any integer number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request-IRQ4 (T0) or IRQ5 (T1) is generated.

The counters can be programmed to start, stop, restart to continue, or restart from the initial value. The counters can also be programmed to stop upon reaching zero (single pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode).

The counters, but not the prescalers, can be read at any time without disturbing their value or count mode. The clock source for T1 is user-definable and can be either the internal microprocessor clock divided-by-four, or an external signal input via Port 3. The Timer Mode register configures the external timer input (P31) as an external clock; a trigger input that can be retriggerable or not-retriggerable; or as a gate input for the internal clock. Port 3 line P36 serves as a timer output (Tout) through which T0, T1 or the internal clock are output. The counter/timers can be cascaded by connecting the T0 output to the input of T1.

Interrupts. The Z86C30 has six different interrupts from six different sources. The interrupts are maskable and prioritized (Figure 11). The six sources are divided as follows; four sources are claimed by Port 3 lines P30-P33 and two in counter/timers. The Interrupt Mask Register globally or individually enables or disables the six interrupt requests (Table 3).

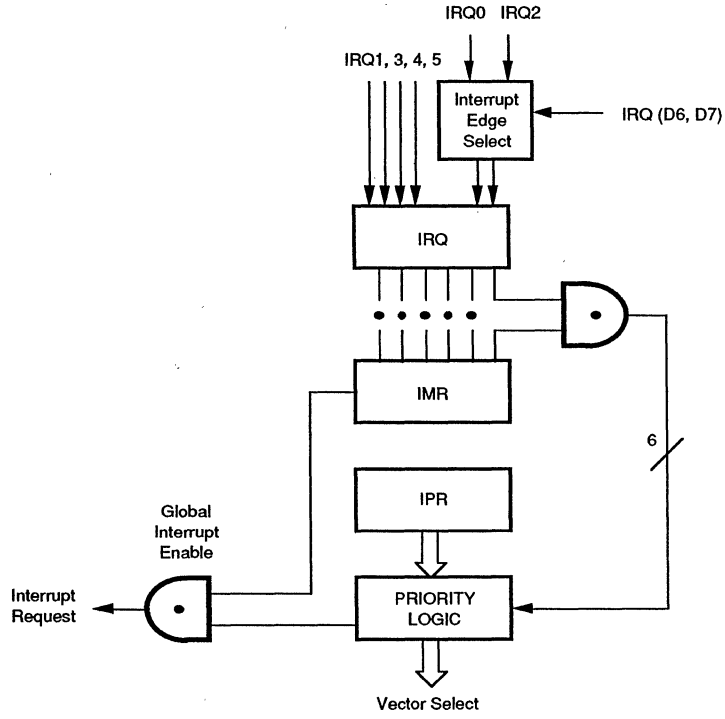


Figure 11. Interrupt Block Diagram

Table 3. Interrupt Types, Sources, and Vectors

Name	Source	Vector Location	Comments
IRQ 0	/DAV 0, IRQ 0	0, 1	External (P32), Rising/Falling Edge Triggered
IRQ 1,	IRQ 1	2, 3	External (P33), Falling Edge Triggered
IRQ 2	/DAV 2, IRQ 2, TIN	4, 5	External (P31), Rising/Falling Edge Triggered
IRQ 3	IRQ3	6, 7	External (P30), Falling Edge Triggered
IRQ 4	T0	8, 9	Internal
IRQ 5	T1	10, 11	Internal

When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register. An interrupt machine cycle is activated when an interrupt request is granted. Thus, disabling all subsequent interrupts saves the Program Counter and Status Flags, and then branches to the program memory vector location reserved for that interrupt. All Z86C30 interrupts are vectored through locations in the program memory. This memory location and the next byte contain the 16-bit starting address of the interrupt service routine for that particular interrupt request.

To accommodate polled interrupt systems, interrupt inputs are masked and the interrupt request register is polled to determine which of the interrupt requests need service.

An interrupt resulting from AN1 is mapped into IRQ2, and an interrupt from AN2 is mapped into IRQ0. Interrupts IRQ2 and IRQ0 may be rising, falling or both edge triggered, and are programmable by the user. The software can poll to identify the state of the pin.

Programming bits for the Interrupt Edge Select are located in the IRQ Register (R250), bits D7 and D6. The configuration is shown in Table 4.

Table 4. IRQ Register

IRQ		Interrupt Edge	
D7	D6	P31	P32
0	0	F	F
0	1	F	R
1	0	R	F
1	1	R/F	R/F

Notes:
F = Falling Edge
R = Rising Edge

Clock. The Z86C30 on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal, RC, LC, ceramic resonator, or any suitable external clock source (XTAL1 = Input, XTAL2 = Output). The crystal should be AT cut, 10 KHz to 12 MHz max., with a series resistance (RS) less than, or equal to, 100 Ohms.

The crystal should be connected across XTAL1 and XTAL2 using the recommended capacitors (capacitance is more than or equal to 22pf) from each pin to ground. The RC oscillator option is mask-programmable, to be selected by the customer at the time ROM code is submitted. The RC oscillator configuration must be an external resistor connected from XTAL1 to XTAL2, with a frequency-setting capacitor from XTAL1 to ground (Figure 12). The RC value vs. Frequency curves are shown in Figures 46 to 48. **(Note: The RC option is not available in the 12 MHz part.)**

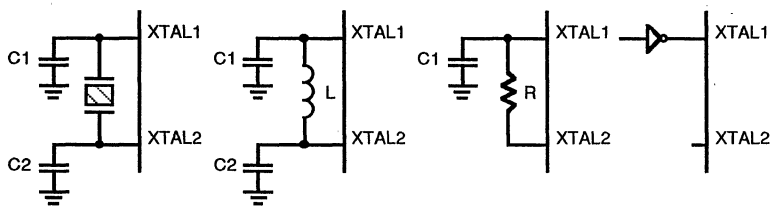
Power-On Reset (POR). A timer circuit clocked by a dedicated on-board RC oscillator is used for the Power-On Reset (POR) timer function. The POR timer allows V_{CC} and the oscillator circuit to stabilize before instruction execution begins.

The POR timer circuit is a one-shot timer triggered by one of three conditions:

1. Power-fail to Power-OK status
2. STOP mode recovery (if D5 of SMR=1)
3. WDT timeout

The POR time is a nominal 5 mS. Bit 5 of the Stop Mode register determines whether the POR timer is bypassed after STOP mode recovery (typical for external clock, and RC/LC oscillators with fast start up time).

HALT. Turns off the internal CPU clock, but not the XTAL oscillation. The counter/timers and external interrupt IRQ0, IRQ1, and IRQ2, remain active. The device may be recovered by interrupts, either external or internal generated.



Ceramic Resonator or Crystal	LC	RC	External Clock
C1, C2 = 47 pf TYP *	C1, C2 = 22 pf	@ 5V VCC (TYP)	
F = 8 MHz	L = 130 uH *	C1 = 33 pf *	
	F = 3 MHz *	R = 1K *	
		F = 9.5 MHz *	

* Preliminary Value Including Pin Parasitics

Figure 12. Oscillator Configuration

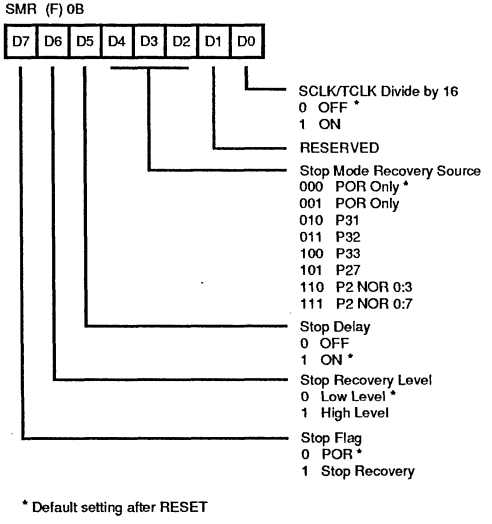
STOP. This instruction turns off the internal clock and external crystal oscillation and reduces the standby current to 10 microamps or less. The Stop mode is terminated by a RESET only, either by WDT timeout, POR, or SMR recovery. This causes the processor to restart the application program at address 000C (HEX). In order to enter STOP (or HALT) mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. To do this, the user must execute a NOP (opcode=FFH) immediately before the appropriate sleep instruction. i.e.:

```
FF      NOP     ; clear the pipeline
6F      STOP    ; enter STOP mode
          or
FF      NOP     ; clear the pipeline
7F      HALT    ; enter HALT mode
```

Stop Mode Recovery Register (SMR). This register selects the clock divide value and determines the mode of STOP mode recovery (Figure 13). All bits are Write only, except Bit 7 which is a Read only. Bit 7 is a flag bit that is hardware set on the condition of STOP Recovery and reset by a power-on cycle. Bit 6 controls whether a low level or high level is required from the recovery source. Bit 5 controls the reset delay after recovery. Bits 2, 3, and 4 of the SMR register specify the source of the STOP Mode Recovery signal. Bits 0 and 1 determine the timeout period of the WDT (Table 6). The SMR is located in bank F of the Expanded Register Group at address OBH.

SCLK/TCLK Divide-by-16 Select (D0). D0 of the SMR controls a divide-by-16 prescaler of SCLK/TCLK. The purpose of this control is to selectively reduce device power consumption during normal processor execution (SCLK control) and/or HALT mode (TCLK sources, counter/timers, and interrupt logic).

STOP Mode Recovery Source (D2, D3, and D4). These three bits of the SMR specify the wake up source of the STOP Mode recovery (Figure 14).



* Default setting after RESET

Figure 13. STOP Mode Recovery Register

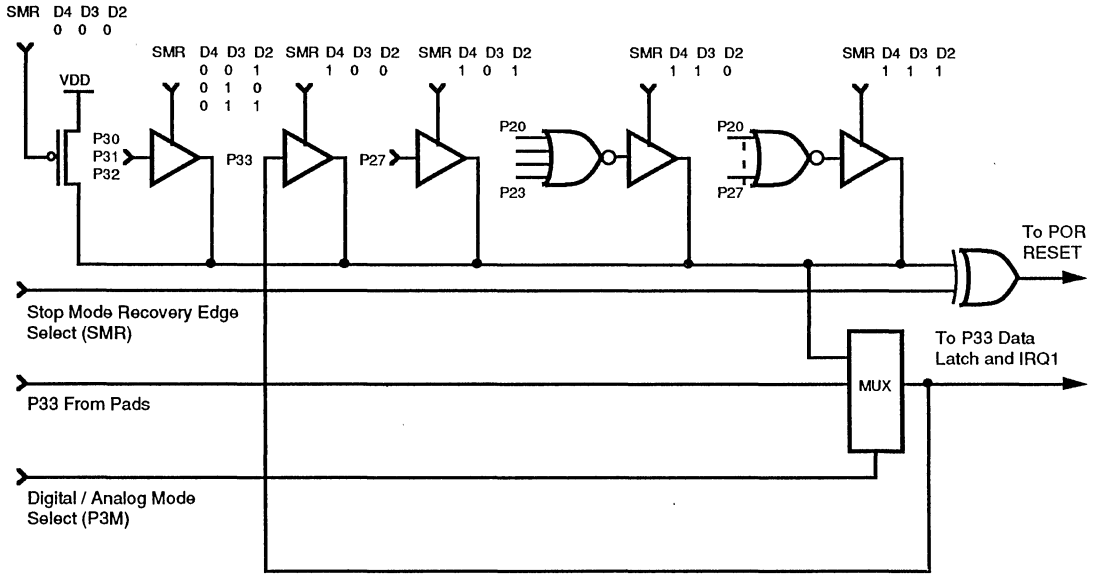


Figure 14. STOP Mode Recovery Source

Table 5. Stop Mode Recovery Source

SMR D4	SMR D3	SMR D2	Operation Description of action
0	0	0	POR recovery only
0	0	1	P30 transition
0	1	0	P31 transition
0	1	1	P32 transition
1	0	0	P33 transition
1	0	1	P27 transition
1	1	0	Logical NOR of Port 2 bits 0:3
1	1	1	Logical NOR of Port 2 bits 0:7

STOP Mode Recovery Delay Select (D5). This bit disables the 5 mS RESET delay after STOP Mode Recovery. The default condition of this bit is 1. If the "fast" wake up is selected, the STOP Mode Recovery source needs to be kept active for at least 5 TpC.

STOP Mode Recovery Level Select (D6). A 1 in this bit position indicates that a high level on any one of the recovery sources wakes the Z86C30 from STOP Mode. A 0 indicates low level recovery. The default is 0 on POR (Figure 14).

Cold or Warm Start (D7). This bit is set by the device upon entering STOP Mode. A 0 in this bit (cold) indicates that the device will be reset by POR RESET. A 1 in this bit (warm) indicates that the device awakens by a SMR source.

Watch Dog Timer Mode Register (WDTMR). The WDT is a retriggerable one-shot timer that will reset the Z8 if it reaches its terminal count. The WDT is initially enabled by executing the WDT instruction and refreshed on subsequent executions of the WDT instruction. The WDT cannot be disabled after it has been initially enabled. The WDT circuit is driven by an on-board RC oscillator or external oscillator from XTAL1 pin. The POR clock source is selected with bit 4 of the WDT register.

WDT Time Select (D0, D1). Bits 0 and 1 control a tap circuit that determines the timeout period. Table 6 shows the different values that can be obtained. The default value of D0 and D1 are 1 and 0, respectively.

FUNCTIONAL DESCRIPTION (Continued)

Table 6. Timeout Period of the WDT

D1	D0	Timeout of Internal RC OSC	Timeout of XTAL clock
0	0	5 ms min	256 TpC
0	1	15 ms min	512 TpC
1	0	25 ms min	1024 TpC
1	1	100 ms min	4096 TpC

Notes:
TpC = XTAL clock cycle
The default on reset is 15 ms.
See Figures 50 to 53 for details.

WDT During HALT (D2). This bit determines whether or not the WDT is active during HALT Mode. A 1 indicates active during HALT. The default is 1.

WDT During STOP (D3). This bit determines whether or not the WDT is active during STOP mode. A 1 indicates active during STOP. A 0 will disable the WDT during STOP mode.

Since the on-board OSC is stopped during STOP mode, the WDT clock source has to select the on-board RC OSC for the WDT to recover from STOP mode. The default is 1.

Clock source for WDT (D4). This bit determines which oscillator source is used to clock the internal POR and WDT counter chain. If the bit is a 1, the internal RC oscillator is bypassed and the POR and WDT clock source is driven from the external pin, XTAL1. The default configuration of this bit is 0, which selects the RC oscillator.

Bits 5 through 7 are reserved. The WDTMR register is accessible only during the first 64 processor cycles (128 XTAL clock cycles) from the execution of the first instruction after Power-On Reset, watch dog reset or a STOP mode recovery. After this point, the register cannot be modified by any means, intentional or otherwise. The WDTMR cannot be read and is located in bank F of the Expanded Register Group at address location 0FH (Figure 15).

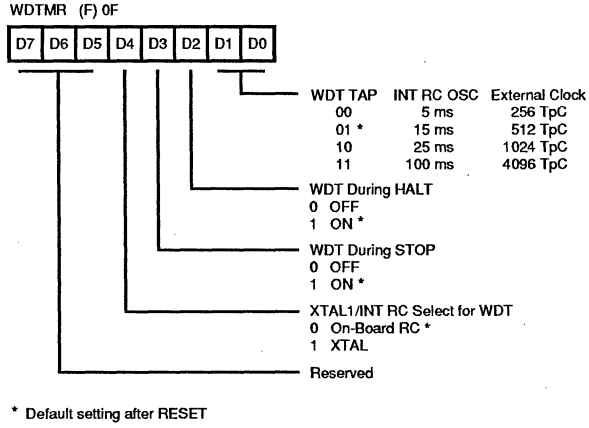


Figure 15. Watchdog Timer Mode Register

Brown Out Protection. An on-board Voltage Comparator checks that V_{CC} is at the required level to ensure correct operation of the device. Reset is globally driven if V_{CC} is below the referenced voltage (Brown Out Voltage). The minimum operating voltage varies with temperature and operating frequency, while the brown out voltage (V_{BO}) varies with temperature only.

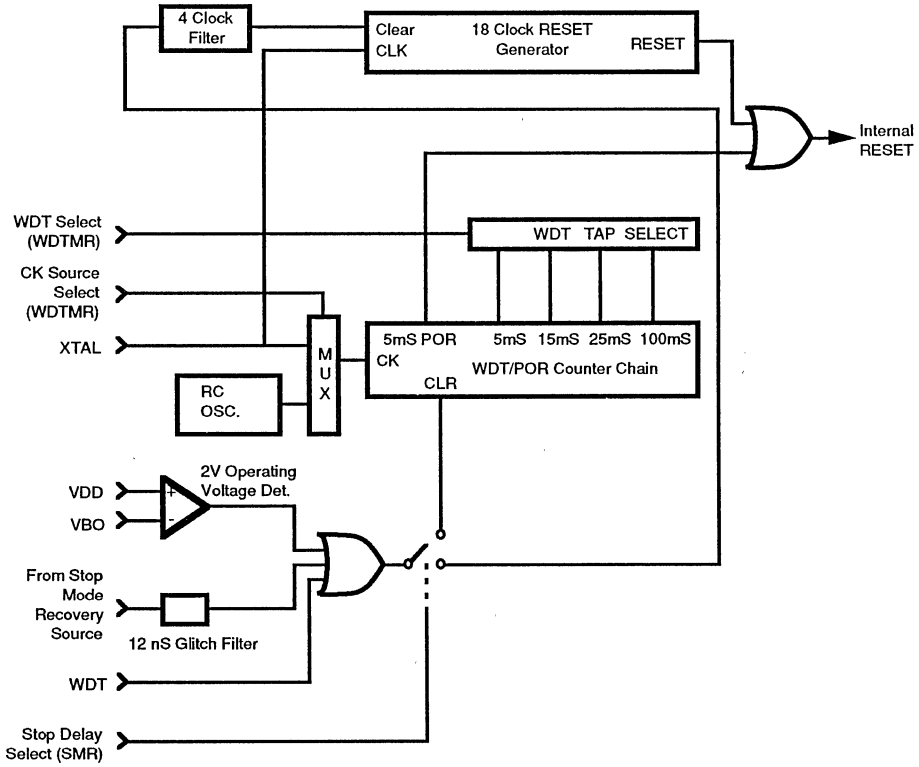


Figure 16. Resets and WDT

The brown-out trip voltage (V_{BO}) is less than 3 volts and above 1.4 volts under the following conditions.

Maximum (V_{BO}) Conditions:

Case1 T_A = -40°C to +105°C, Internal Clock Frequency equal or less than 1 MHz

Case 2 T_A = -40°C to + 85°C, Internal Clock Frequency equal or less than 2 MHz

Note: The internal clock frequency runs at half the external clock frequency.

The device functions normally at or above 3.0V under all conditions. Below 3.0V, the device is guaranteed to function normally until the Brown-Out Protection trip point (V_{BO}) is reached for the temperatures and operating frequencies in case 1 and case 2 above. The actual brown-out trip point is a function of temperature and process parameters (Figure 17).

ROM Protect. ROM protect is mask-programmable. It is selected by the customer at the time the ROM code is submitted. **The selection of ROM protect will disable the LDC and LDCI instructions in ALL modes.**

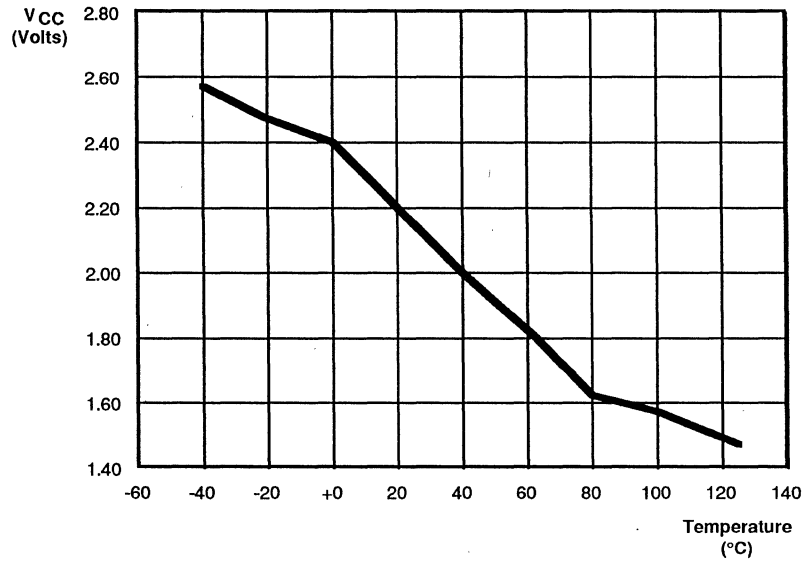


Figure 17. Typical Z86C30 V_{BO} Voltage vs Temperature

STANDARD TEST CONDITIONS

The characteristics listed below apply for standard test conditions, as noted. All voltages are referenced to GND. Positive current flows into the referenced pin (Figure 18).

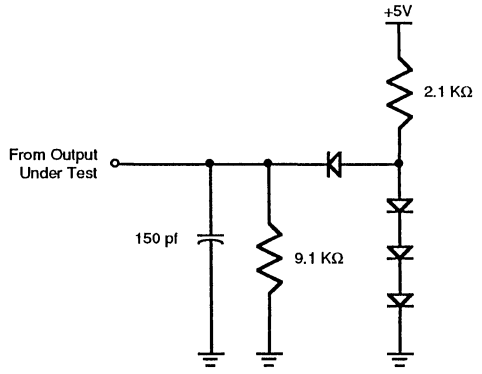


Figure 18. Test Load Configuration

ABSOLUTE MAXIMUM RATINGS

Symbol	Description	Min	Max	Units
Vcc	Supply Voltage (*)	-0.3	+7.0	V
TSTG	Storage Temp	-65	+150	C
T_A	Oper Ambient Temp		†	C
	Power Dissipation		2.2	W

Notes:
* Voltage on all pins with respect to GND.
† See Ordering Information.

Stress greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended period may affect device reliability.

CAPACITANCE

T_A = 25°C, V_{CC} = GND = 0V, f = 1.0 MHz, Unmeasured pins to GND.

Parameter	Max
Input capacitance	12 pF
Output capacitance	12 pF
I/O capacitance	12 pF

DC ELECTRICAL CHARACTERISTICS
Z86C30

Symbol	Parameter	V_{CC} Note [3]	T_A = 0°C to 70°C Min	Max	T_A = -40°C to 105°C Min	Max	Typical at 25°C	Units	Conditions	Notes
	Max Input Voltage	3.3V		7		7		V	I_{IN} 250µA	
		5.0V		7		7		V	I_{IN} 250µA	
V_{CH}	Clock Input High Voltage	3.3V	0.7 V_{CC}	V_{CC}+0.3	0.7 V_{CC}	V_{CC}+0.3	1.3	V	Driven by External Clock Generator	
		5.0V	0.7 V_{CC}	V_{CC}+0.3	0.7 V_{CC}	V_{CC}+0.3	2.5	V	Driven by External Clock Generator	
V_{CL}	Clock Input Low Voltage	3.3V	V_{SS}-0.3	0.2 V_{CC}	V_{SS}-0.3	0.2 V_{CC}	0.7	V	Driven by External Clock Generator	
		5.0V	V_{SS}-0.3	0.2 V_{CC}	V_{SS}-0.3	0.2 V_{CC}	1.5	V	Driven by External Clock Generator	
V_{IH}	Input High Voltage	3.3V	0.7 V_{CC}	V_{CC}+0.3	0.7 V_{CC}	V_{CC}+0.3	1.3	V		
		5.0V	0.7 V_{CC}	V_{CC}+0.3	0.7 V_{CC}	V_{CC}+0.3	2.5	V		
V_{IL}	Input Low Voltage	3.3V	V_{SS}-0.3	0.2 V_{CC}	V_{SS}-0.3	0.2 V_{CC}	0.7	V		
		5.0V	V_{SS}-0.3	0.2 V_{CC}	V_{SS}-0.3	0.2 V_{CC}	1.5	V		
V_{OH}	Output High Voltage	3.3V	V_{CC}-0.4		V_{CC}-0.4		3.1	V	I_{OL} = -2.0 mA	
		5.0V	V_{CC}-0.4		V_{CC}-0.4		4.8	V	I_{OL} = -2.0 mA	
$V_{OL}1$	Output Low Voltage	3.3V		0.6		0.6	0.2	V	I_{OL} = +4.0 mA	
		5.0V		0.4		0.4	0.1	V	I_{OL} = +4.0 mA	
$V_{OL}2$	Output Low Voltage	3.3V		1.2		1.2	0.3	V	I_{OL} = +6 mA, 3 Pin Max	
		5.0V		1.2		1.2	0.3	V	I_{OL} = +12 mA, 3 Pin Max	
V_{RH}	Reset Input High Voltage	3.3V	.8 V_{CC}	V_{CC}	.8 V_{CC}	V_{CC}	1.5	V		
		5.0V	.8 V_{CC}	V_{CC}	.8 V_{CC}	V_{CC}	2.1	V		
V_{RI}	Reset Input Low Voltage	3.3V	V_{SS}-0.3	0.2 V_{CC}	V_{SS}-0.3	0.2 V_{CC}	1.1			
		5.0V	V_{SS}-0.3	0.2 V_{CC}	V_{SS}-0.3	0.2 V_{CC}	1.7			
V_{OFFSET}	Comparator Input Offset Voltage	3.3V		25		25	10	mV		
		5.0V		25		25	10	mV		
I_{IL}	Input Leakage	3.3V	-1	1	-1	2	<1	µA	V_{IN} = 0V, V_{CC}	
		5.0V	-1	1	-1	2	<1	µA	V_{IN} = 0V, V_{CC}	
I_{OL}	Output Leakage	3.3V	-1	1	-1	2	<1	µA	V_{IN} = 0V, V_{CC}	
		5.0V	-1	1	-1	2	<1	µA	V_{IN} = 0V, V_{CC}	
I_{IR}	Reset Input Current	3.3V		-45		-60	-20	µA		
		5.0V		-55		-70	-30	µA		
I_{CC}	Supply Current	3.3V		10		10	4	mA	@ 8 MHz	[4,5]
		5.0V		15		15	10	mA	@ 8 MHz	[4,5]
		3.3V		15		15	5	mA	@ 12 MHz	[4,5]
		5.0V		20		20	15	mA	@ 12 MHz	[4,5]

DC ELECTRICAL CHARACTERISTICS (Continued)
Z86C30

Symbol	Parameter	V_{CC} Note [3]	T_A = 0°C to 70°C Min	Max	T_A = -40°C to 105°C Min	Max	Typical at 25°C	Units	Conditions	Notes
$I_{CC}1$	Standby Current	3.3V		3		3	1	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 8 MHz	[4,5]
		5.0V		5		5	2.4	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 8 MHz	[4,5]
		3.3V		4		4	1.5	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 12 MHz	[4,5]
		5.0V		6		6	3.2	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 12 MHz	[4,5]
		3.3V		2		2	0.8	mA	Clock Divide by 16 @ 8 MHz	[4,5]
		5.0V		4		4	1.8	mA	Clock Divide by 16 @ 8 MHz	[4,5]
		3.3V		3		3	1.2	mA	Clock Divide by 16 @ 12 MHz	[4,5]
		5.0V		5		5	2.5	mA	Clock Divide by 16 @ 12 MHz	[4,5]
$I_{CC}2$	Standby Current	3.3V		8		15	1	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is not Running	[6]
		5.0V		10		20	2	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is not Running	[6]
		3.3V		500		600	310	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is Running	[6]
		5.0V		800		1000	600	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is Running	[6]
I_{ALL}	Auto Latch Low Current	3.3V		8		10	5	µA	$0V < V_{IN} < V_{CC}$	
		5.0V		15		20	11	µA	$0V < V_{IN} < V_{CC}$	
I_{ALH}	Auto Latch High Current	3.3V		-5		-7	-3	µA	$0V < V_{IN} < V_{CC}$	
		5.0V		-8		-10	-6	µA	$0V < V_{IN} < V_{CC}$	
T_{POR}	Power On Reset	3.3V	7	24	8	25	13	mS		
		5.0V	3	13	4	14	7	mS		
V_{BO}	V_{CC} Brown Out Voltage		1.5	2.65	1.2	2.95	2.1	V	2 MHz max Ext. CLK Freq.	[3]

Notes:

[1]

ICC1	Type	Max	Unit	Freq
Clock Driven on Crystal	3.0 mA	5	mA	8 MHz
or XTAL Resonator	0.3 mA	50	mA	8 MHz

[2] V_{SS} =0V=GND.
[3] 5.0V ± 0.5V, 3.3 V ± 0.3V. The V_{BO} increases as the temperature decreases.
[4] All outputs unloaded, I/O pins floating, inputs at rail.
[5] C_L1=C_L2=100 pF.
[6] Same as note [4] except inputs at V_{CC}.

AC ELECTRICAL CHARACTERISTICS
Additional Timing Diagram

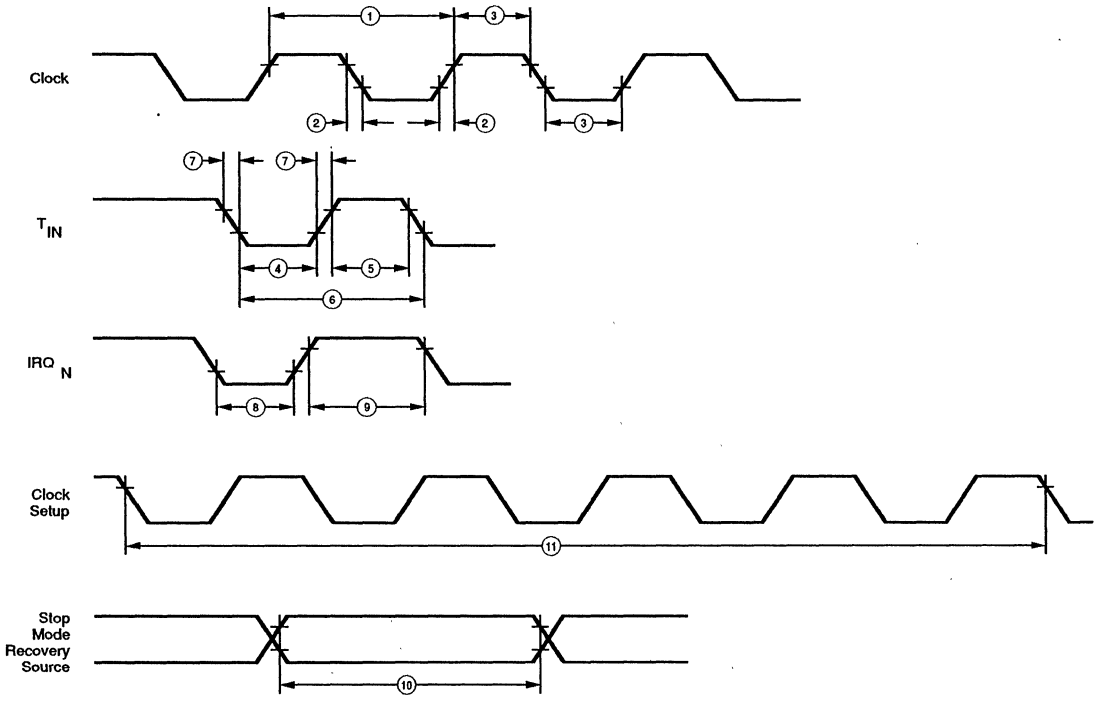


Figure 19. Additional Timing

AC ELECTRICAL CHARACTERISTICS
Additional Timing Table

No	Symbol	Parameter	V_{CC} Note[6]	T_A = 0°C to 70°C				T_A = -40°C to 105°C				Units	Notes
				8 MHz		12 MHz		8 MHz		12 MHz			
				Min	Max	Min	Max	Min	Max	Min	Max		
1	TpC	Input Clock Period	3.3V	125	100000	83	100000	125	100000	83	100000	ns	[1]
			5.0V	125	100000	83	100000	125	100000	83	100000	ns	[1]
2	TrC,TfC	Clock Input Rise & Fall Times	3.3V		25		15		25		15	ns	[1]
			5.0V		25		15		25		15	ns	[1]
3	TwC	Input Clock Width	3.3V	37		26		37		26		ns	[1]
			5.0V	37		26		37		26		ns	[1]
4	TwTinL	Timer Input Low Width	3.3V	100		100		100		100		ns	[1]
			5.0V	70		70		70		70		ns	[1]
5	TwTinH	Timer Input High Width	3.3V	3TpC		3TpC		3TpC		3TpC			[1]
6	TpTin	Timer Input Period	3.3V	8TpC		8TpC		8TpC		8TpC			[1]
			5.0V	8TpC		8TpC		8TpC		8TpC			[1]
7	TrTin	Timer Input Rise & Fall Timers	3.3V		100		100		100		100	ns	[1]
			5.0V		100		100		100		100	ns	[1]
8A	TwIL	Int. Request Low Time	3.3V	100		100		100		100		ns	[1,2]
			5.0V	70		70		70		70		ns	[1,2]
8B	TwIL	Int. Request Low Time	3.3V	3TpC		3TpC		3TpC		3TpC			[1,3]
			5.0V	3TpC		3TpC		3TpC		3TpC			[1,3]
9	TwIH	Int. Request Input High Time	3.3V	3TpC		3TpC		3TpC		3TpC			[1,2]
			5.0V	3TpC		3TpC		3TpC		3TpC			[1,2]
10	Twsm	STOP Mode Recovery Width Spec	3.3V	12		12		12		12		ns	
			5.0V	12		12		12		12		ns	
			3.3V	5TpC									Reg. SMR - D5=0 No Delay
			5.0V	5TpC									Reg. SMR - D5=1 with Delay
11	Tost	Oscillator Startup Time	3.3V		5TpC		5TpC		5TpC		5TpC		[4]
			5.0V		5TpC		5TpC		5TpC		5TpC		[4]
12	Twdt	Watchdog Timer Delay Time	3.3V	10		10		10		10		ms	D0 = 0 [5]
			5.0V	5		5		5		5		ms	D1 = 0 [5]
			3.3V	30		30		30		30		ms	D0 = 1 [5]
			5.0V	15		15		15		15		ms	D1 = 0 [5]
			3.3V	50		50		50		50		ms	D0 = 0 [5]
			5.0V	25		25		25		25		ms	D1 = 1 [5]
			3.3V	200		200		200		200		ms	D0 = 1 [5]
			5.0V	100		100		100		100		ms	D1 = 1 [5]

Notes:
[1] Timing Reference uses 0.9 V_{CC} for a logic "1" and 0.1 V_{CC} for a logic "0".
[2] Interrupt request via Port 3 (P31-P33).
[3] Interrupt request via Port 3 (P30).
[4] SMR-D5 = 0.
[5] Reg. WDTMR.
[6] 5.0V ± 0.5V, 3.3V ± 0.3V.

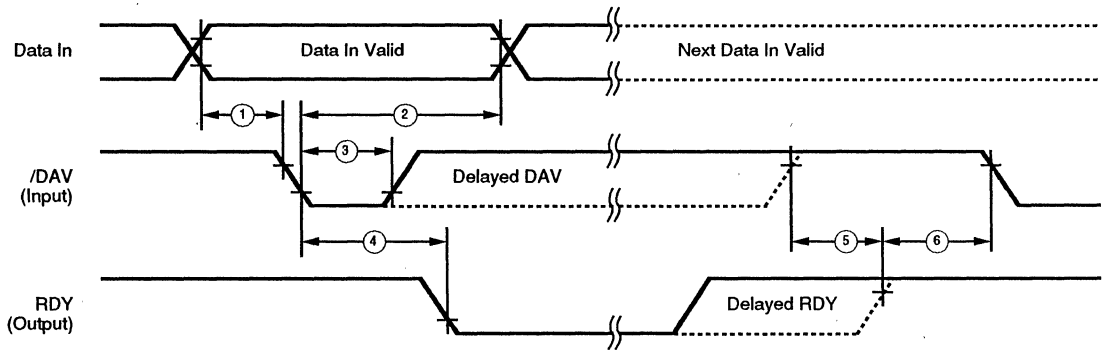


Figure 20. Input Handshake Timing

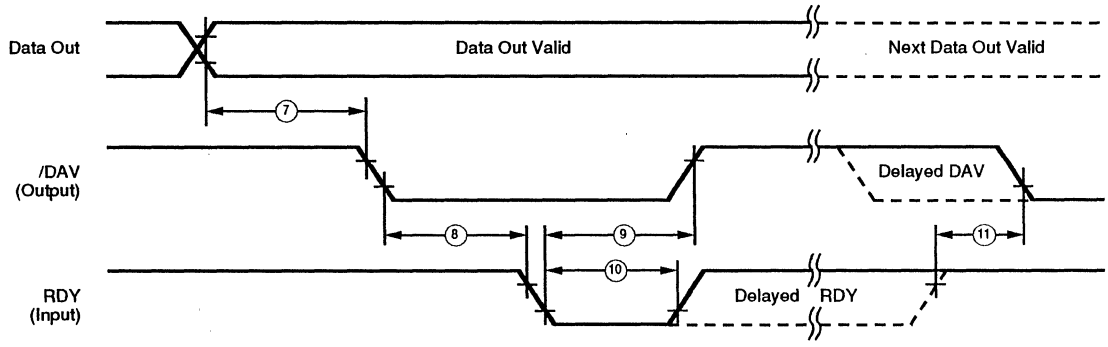
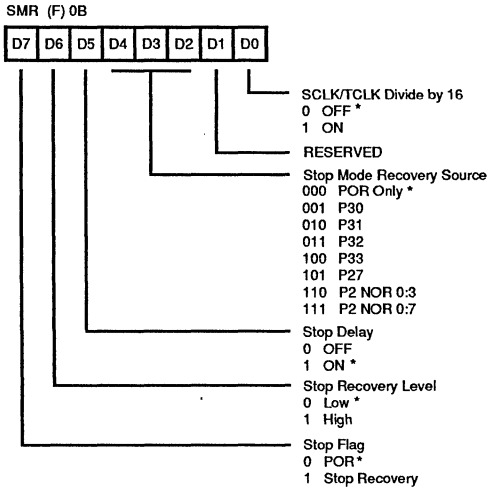


Figure 21. Output Handshake Timing

AC ELECTRICAL CHARACTERISTICS (Continued)
Handshake Timing Table

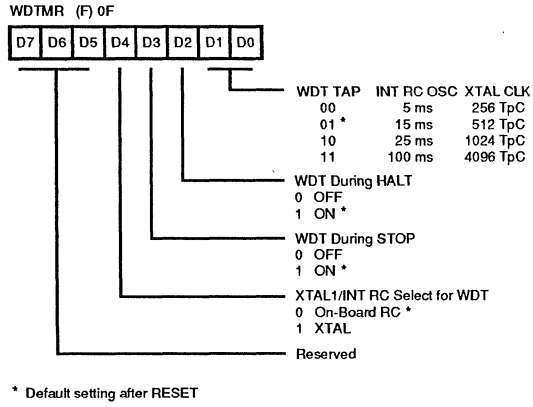
No	Symbol	Parameter	V_{cc} Note[1]	T_A = 0°C to 70°C				T_A = -40°C to 105°C				Notes
				8 MHz		12 MHz		8 MHz		12 MHz		
				Min	Max	Min	Max	Min	Max	Min	Max	
1	TsDI(DAV)	Data In Setup Time	3.3V	0		0		0		0		IN
			5.0V	0		0		0		0		IN
2	ThDI(DAV)	Data In Hold Time	3.3V	160		160		160		160		IN
			5.0V	115		115		115		115		IN
3	TwDAV	Data Available Width	3.3V	155		155		155		155		IN
			5.0V	110		110		110		110		IN
4	TdDAVI(RDY)	DAV Fall to RDY Fall Delay	3.3V		160		160		160		160	IN
			5.0V		115		115		115		115	IN
5	TdDAVId(RDY)	DAV Rise to RDY Rise Delay	3.3V		120		120		120		120	IN
			5.0V		80		80		80		80	IN
6	TdDO(DAV)	RDY Rise to DAV Fall Delay	3.3V	0		0		0		0		IN
			5.0V	0		0		0		0		IN
7	TcLDAV0(RDY)	Data Out to	3.3V	63		42		63		42		OUT
		DAV Fall Delay	5.0V	63		42		63		42		OUT
8	TcLDAV0(RDY)	DAV Fall to RDY Fall Delay	3.3V	0		0		0		0		OUT
			5.0V	0		0		0		0		OUT
9	TdRDY0(DAV)	RDY Fall to DAV Rise Delay	3.3V		160		160		160		160	OUT
			5.0V		115		115		115		115	OUT
10	TwRDY	RDY Width	3.3V	110		110		110		110		OUT
			5.0V	80		80		80		80		OUT
11	TdRDY0d(DAV)	RDY Rise to DAV Fall Delay	3.3V		110		110		110		110	OUT
			5.0V		80		80		80		80	OUT

Note:
[1] 5.0 V ± 0.5V, 3.3V ± 0.3V.



* Default setting after RESET

Figure 22. Stop Mode Recovery Register



* Default setting after RESET

Figure 23. Watchdog Timer Mode Register

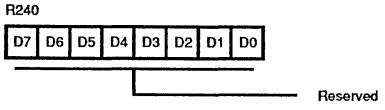


Figure 24. Reserved

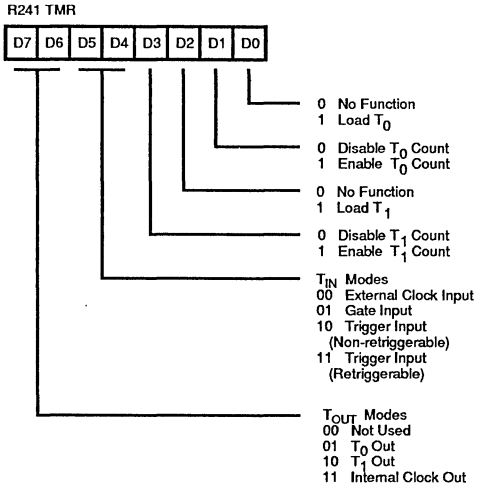


Figure 25. Timer Mode Register
(F1H: Read/Write)

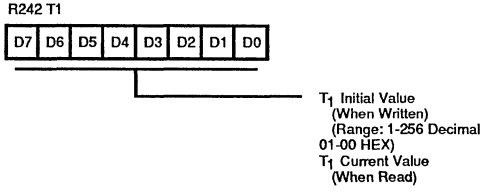


Figure 26. Counter Timer 1 Register
(F2H: Read/Write)

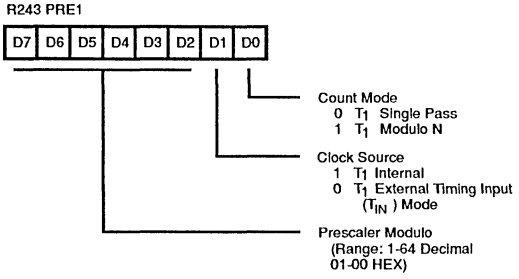


Figure 27. Prescaler 1 Register
(F3H: Write Only)

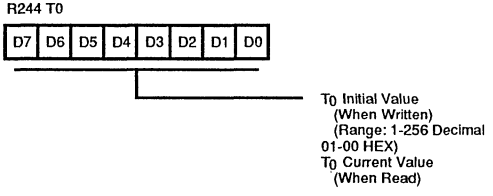


Figure 28. Counter/Timer 0 Register
(F4H: Read/Write)

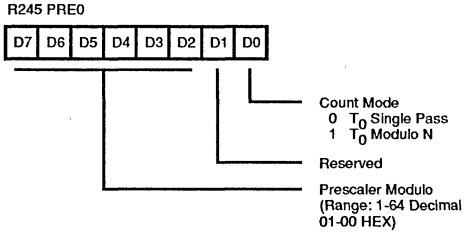


Figure 29. Prescaler 0 Register
(F5H: Write Only)

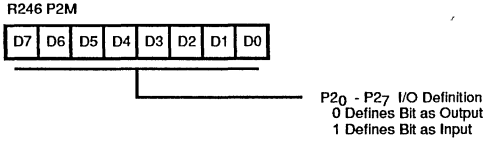


Figure 30. Port 2 Mode Register
(F6H: Write Only)

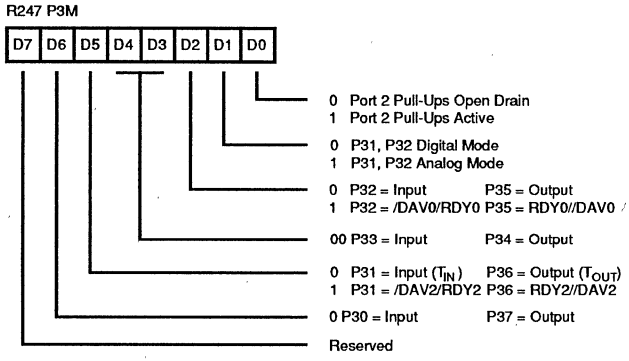


Figure 31. Port 3 Mode Register (F7H: Write Only)

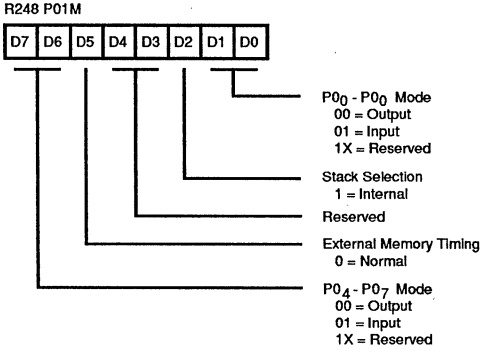


Figure 32. Port 0 and 1 Mode Register (F8H: Write Only)

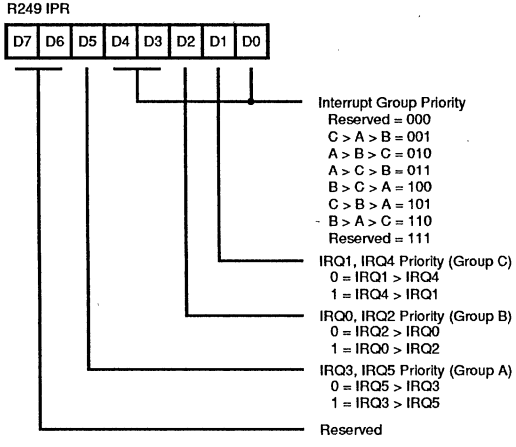


Figure 33. Interrupt Priority Register (F9H: Write Only)

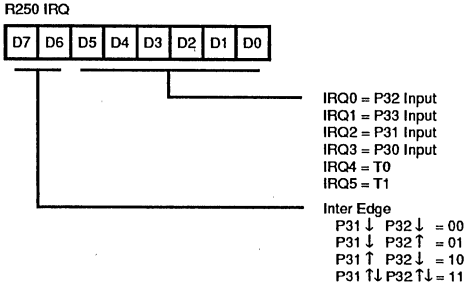


Figure 34. Interrupt Request Register (FAH: Read/Write)

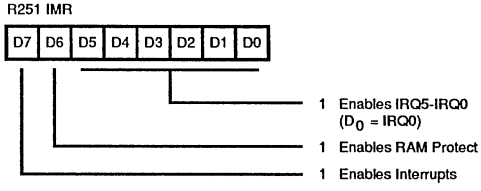


Figure 35. Interrupt Mask Register (FBH: Read/Write)

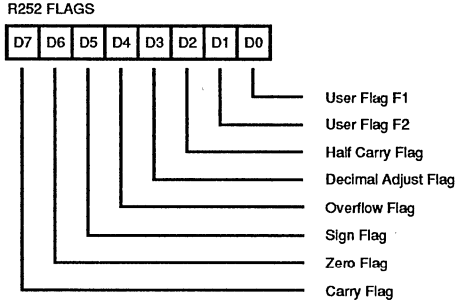


Figure 36. Flag Register (FCH: Read/Write)

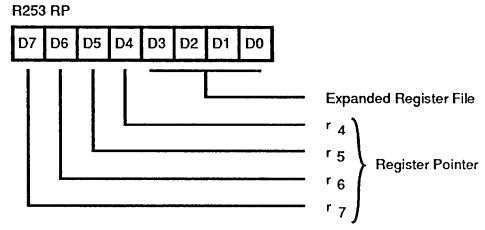


Figure 37. Register Pointer (FDH: Read/Write)

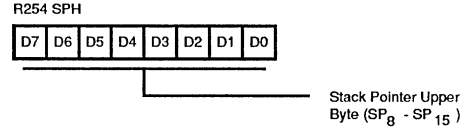


Figure 38. Reserved

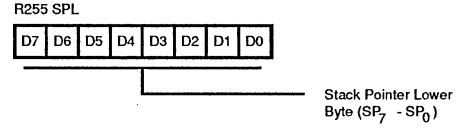


Figure 39. Stack Pointer (FFH: Read/Write)

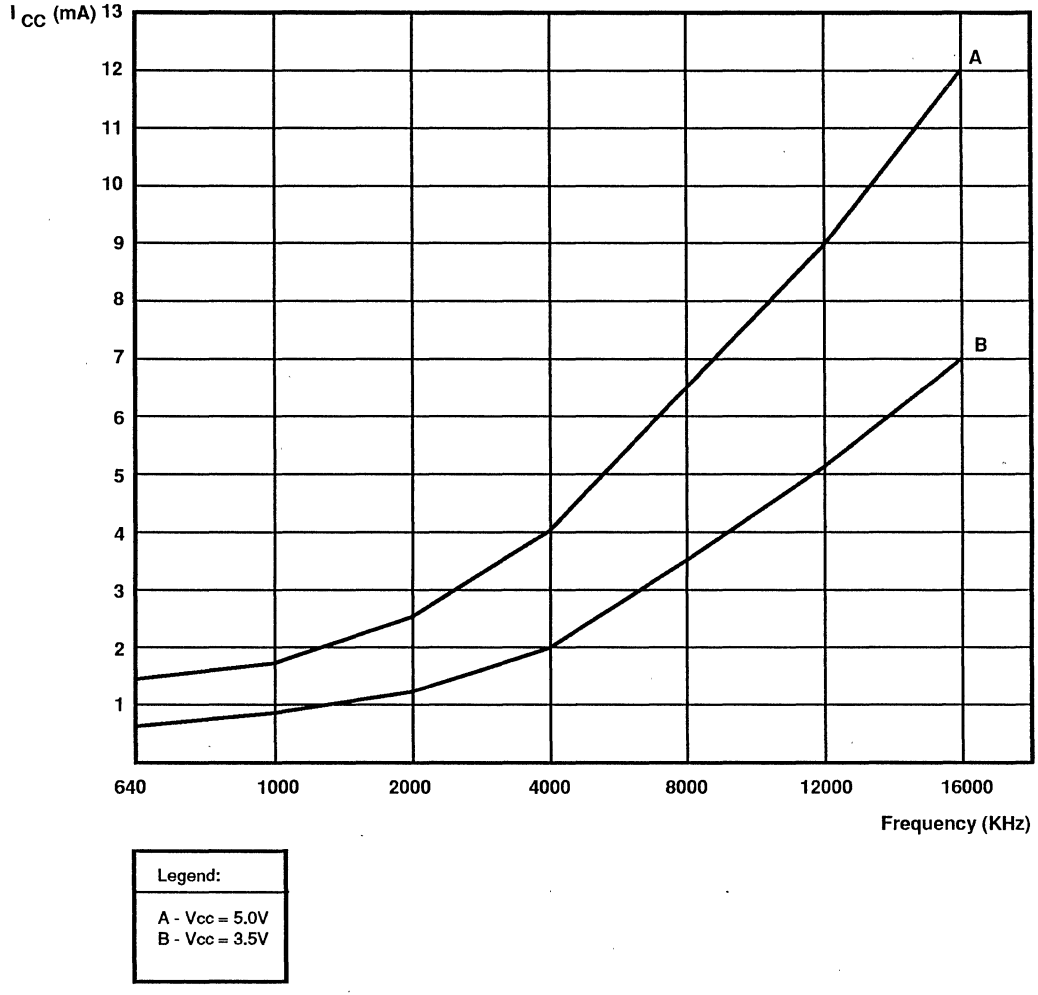
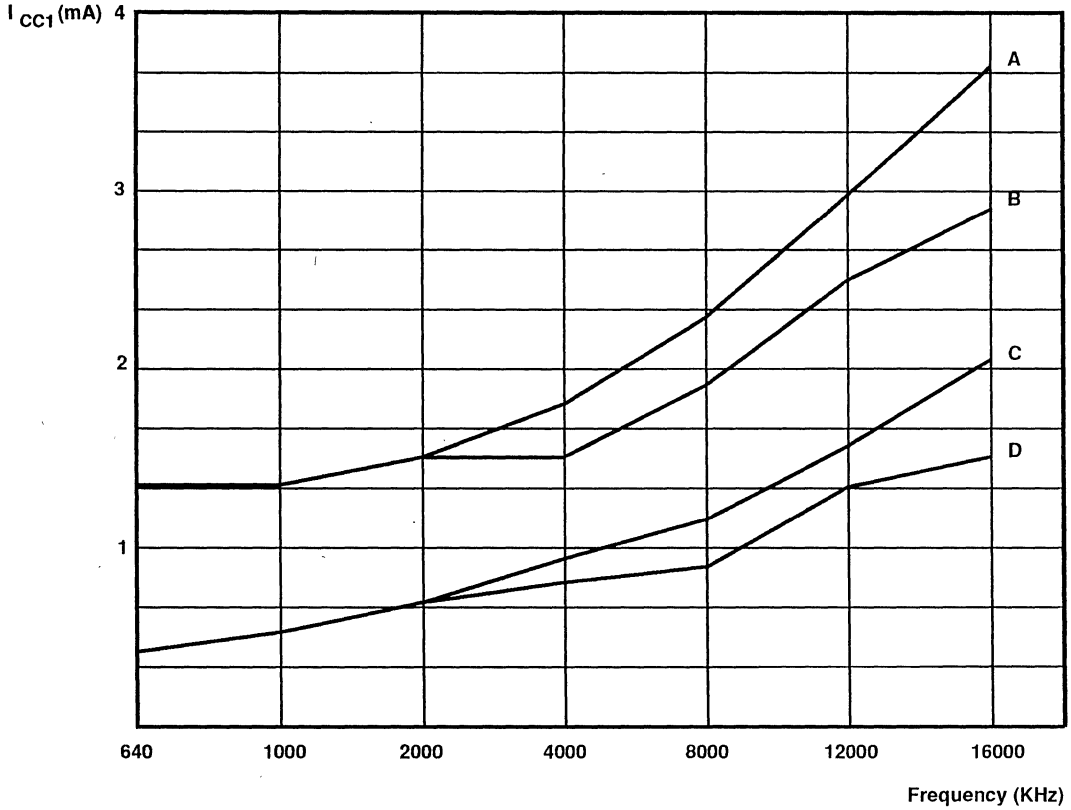
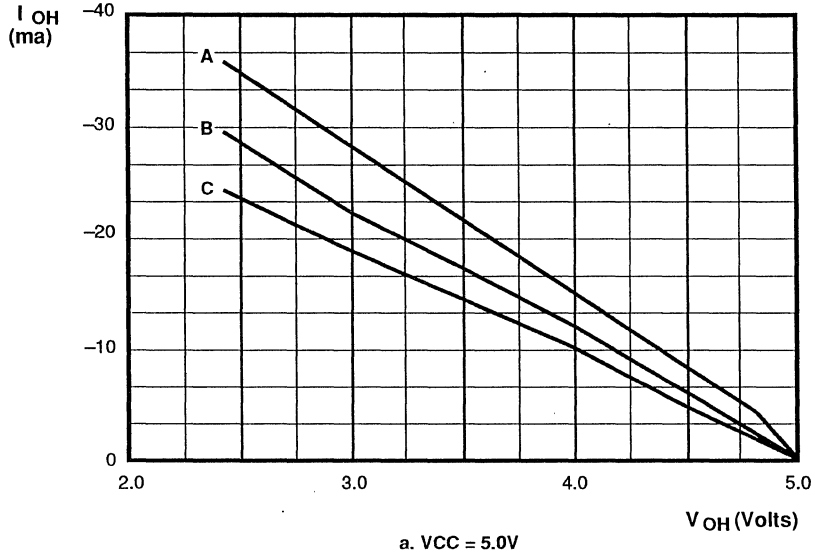


Figure 40. Typical I_{cc} vs Frequency

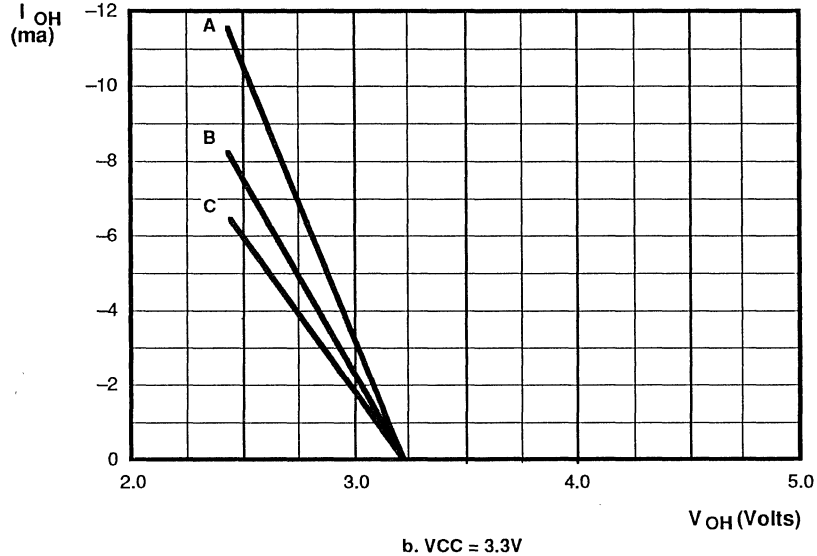


Legend:
A - Vcc1 = 5.0V B - Vcc1 = 5.0V (SCLK Divided by 16) C - Vcc1 = 3.5V D - Vcc1 = 3.5V (SCLK Divided by 16)

Figure 41. Typical I_{CC1} vs Frequency



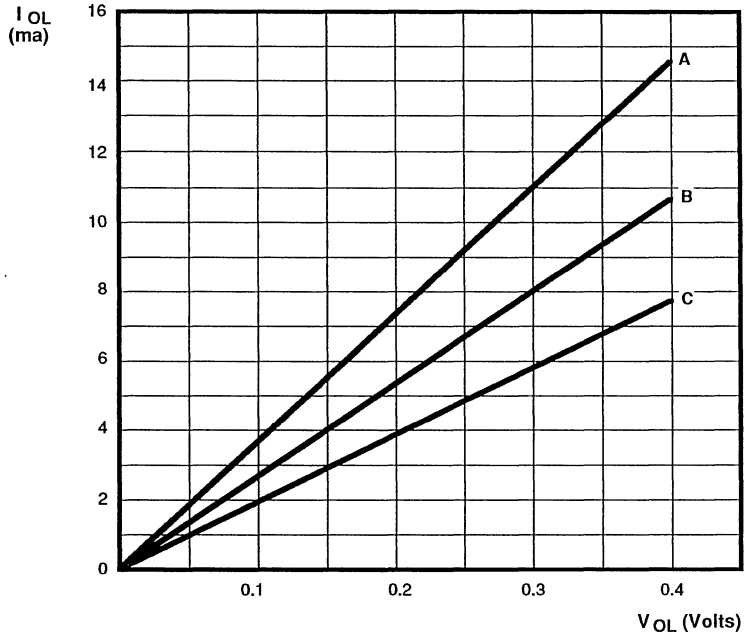
a. VCC = 5.0V



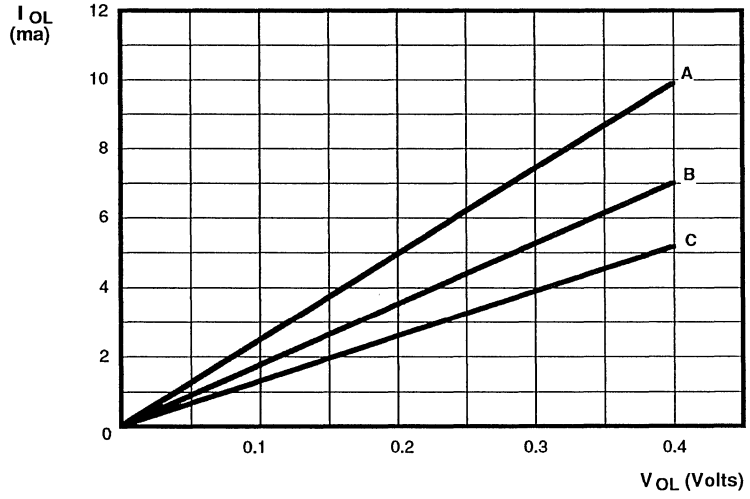
b. VCC = 3.3V

Legend:
A = –55°C B = 25°C C = 125°C

Figure 42. Typical I_{OH} vs V_{OH} Over Temperature



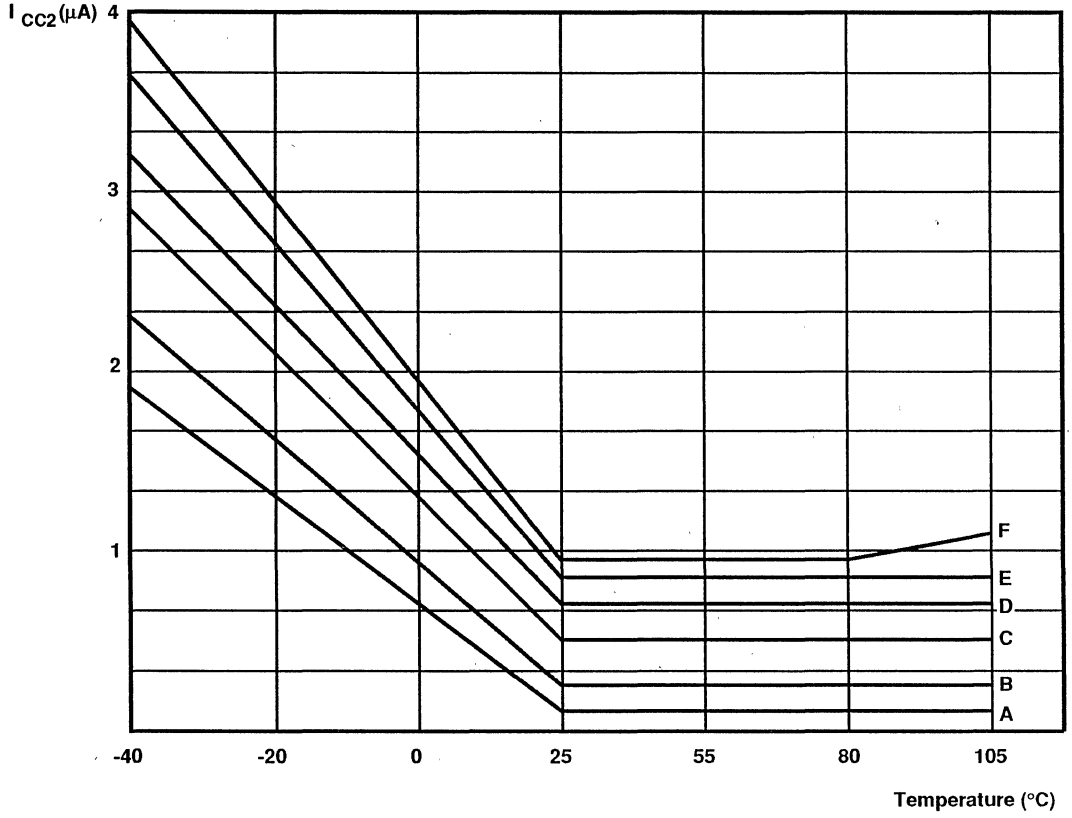
a. VCC = 5.0V



b. VCC = 3.3V

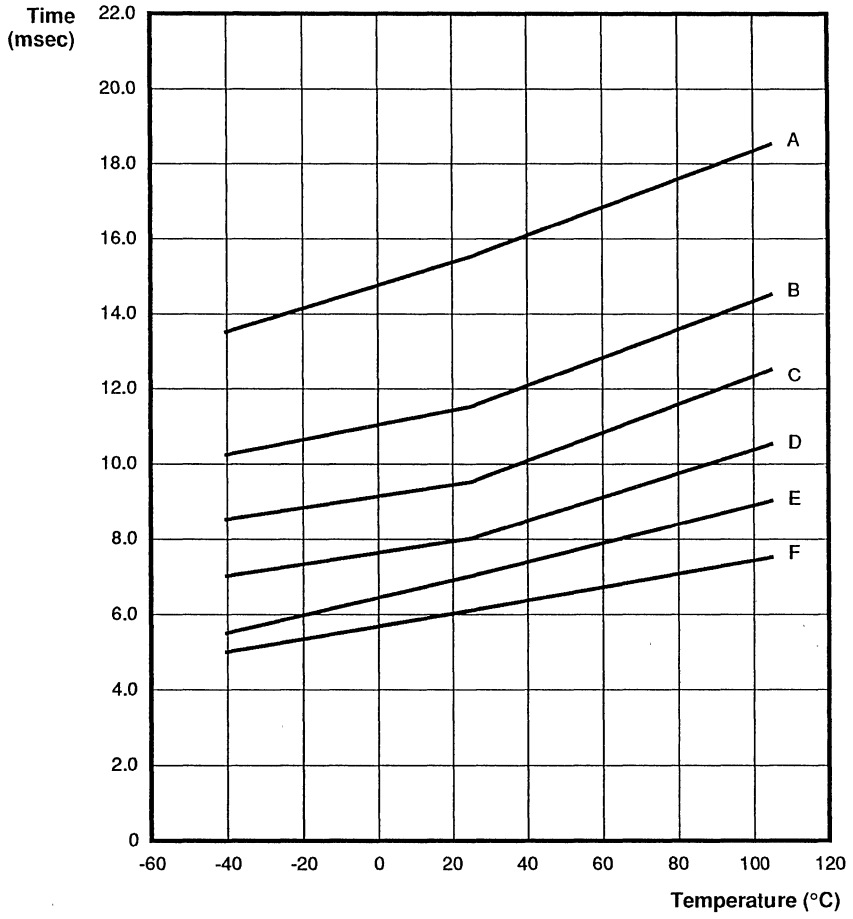
Legend:
A = –55°C B = 25°C C = 125°C

Figure 43. Typical I_{OL} vs V_{OL} Over Temperature



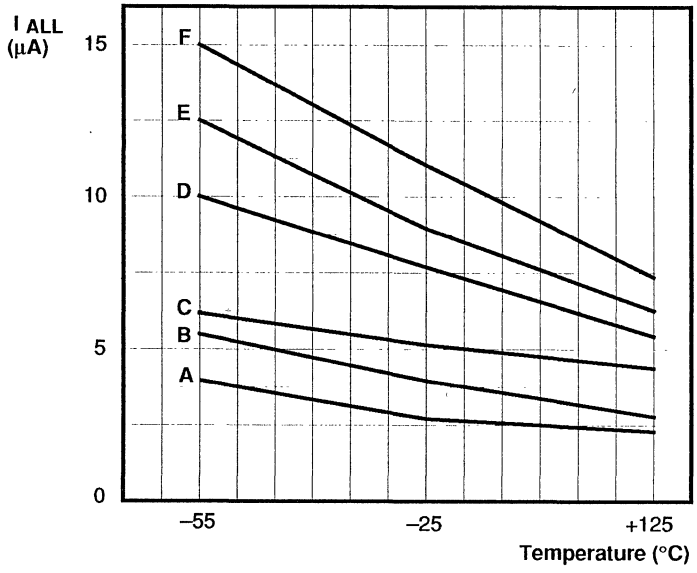
Legend:	
A - Vcc = 3.0V	D - Vcc = 4.5V
B - Vcc = 3.5V	E - Vcc = 5.0V
C - Vcc = 4.0V	F - Vcc = 5.5V

Figure 44. Typical I_{CC2} vs Temperature

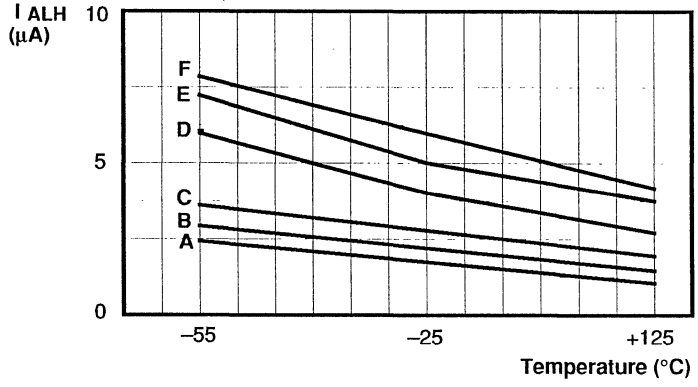


Legend:	
A - Vcc = 3.0V	D - Vcc = 4.5V
B - Vcc = 3.5V	E - Vcc = 5.0V
C - Vcc = 4.0V	F - Vcc = 5.5V

Figure 45. Typical Power-On Reset Time vs Temperature



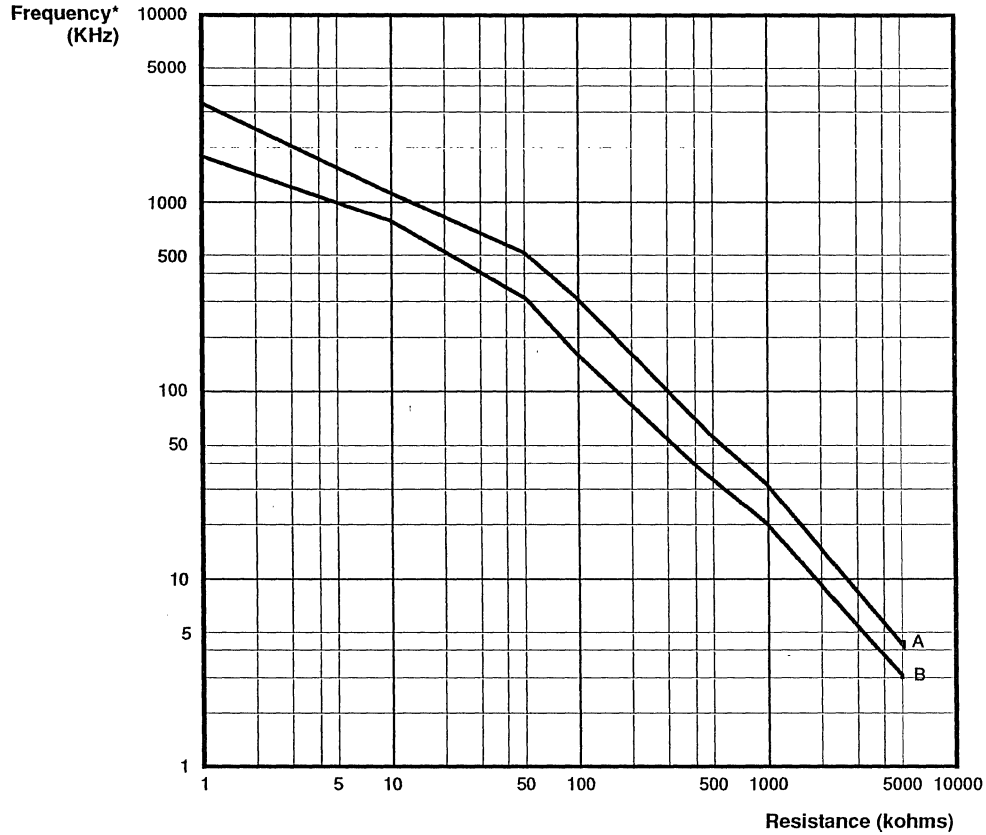
a. Typical Auto Latch Low Current vs Temperature



b. Typical Auto Latch High Current vs Temperature

Legend:	
A - Vcc = 3.0V	D - Vcc = 4.5V
B - Vcc = 3.3V	E - Vcc = 5.0V
C - Vcc = 3.6V	F - Vcc = 5.5V

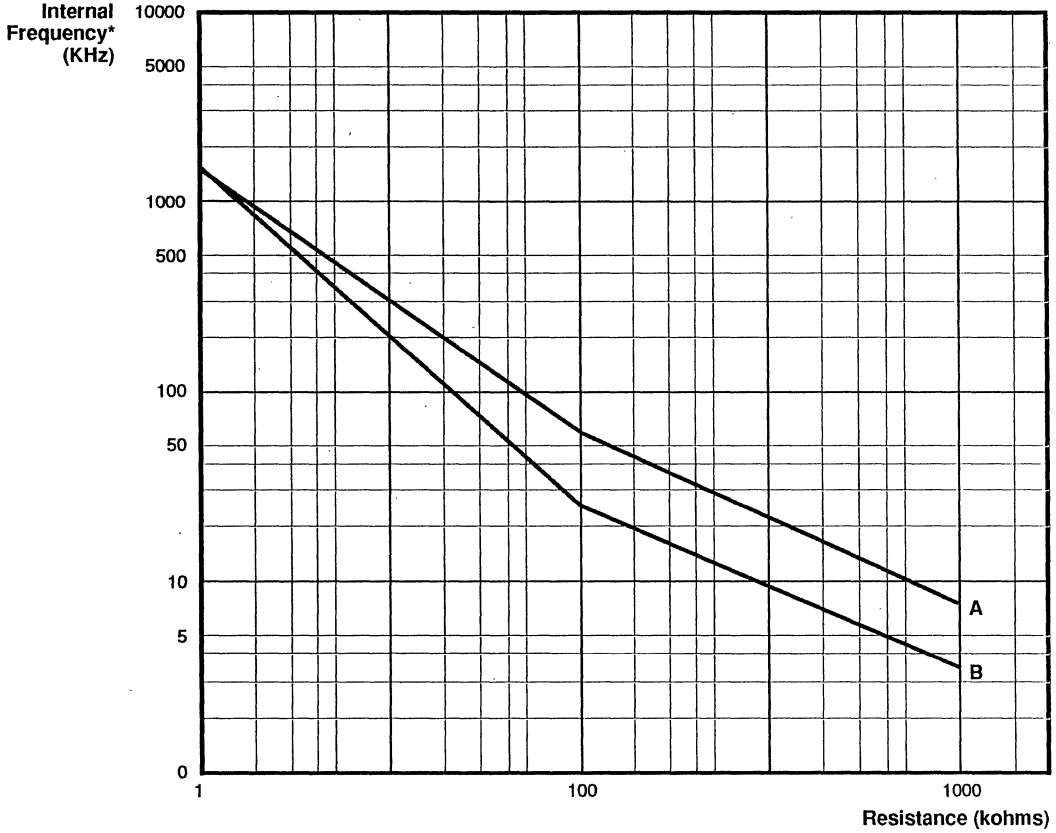
Figure 46. Typical Auto-Latch Current vs Temperature



Legend:
A - Vcc = 5.0V C = 33 pF B - Vcc = 3.3V C = 33 pF

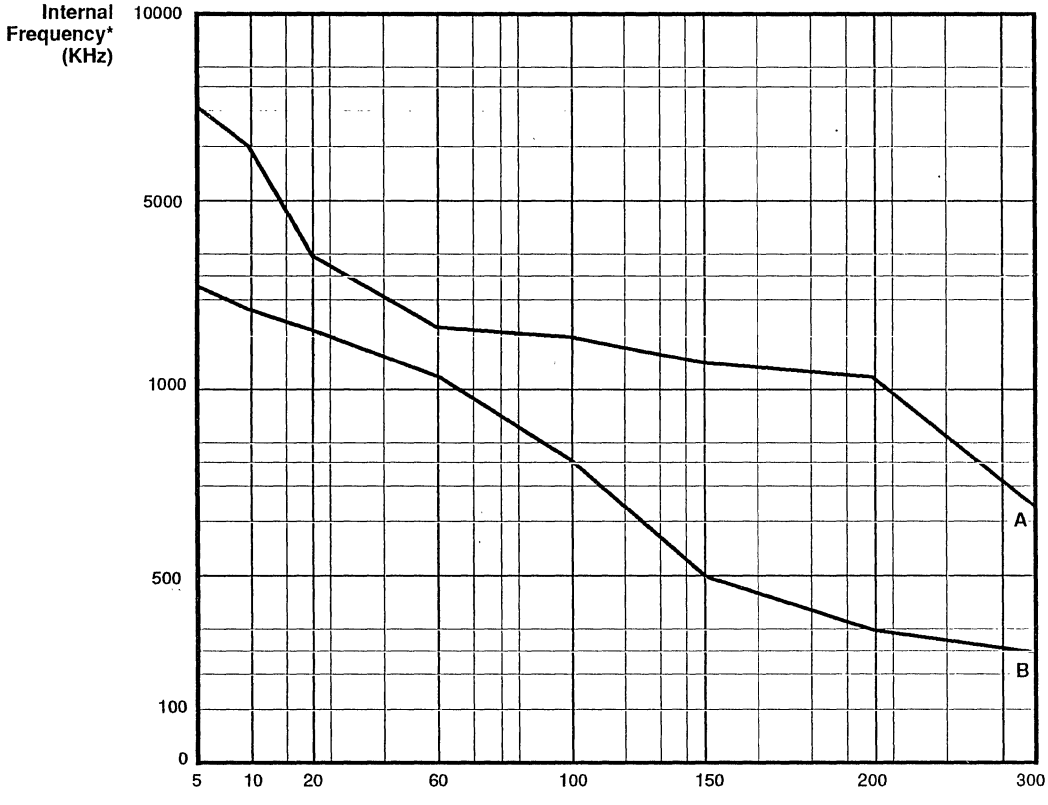
Note: * The internal clock frequency is one half the external clock frequency.
This chart for reference only. Each process will have a different characteristc curve.

Figure 47. Typical Internal Frequency vs RC Resistance



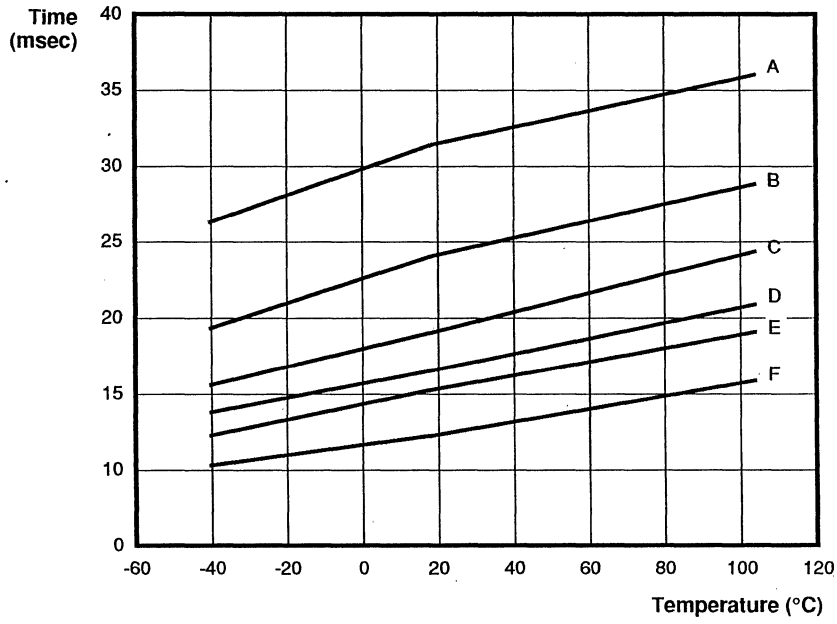
Note: * The internal clock frequency is one half the external clock frequency.
This chart for reference only. Each process will have a different characteristc curve.

Figure 48. Typical Internal Frequency vs Resistance



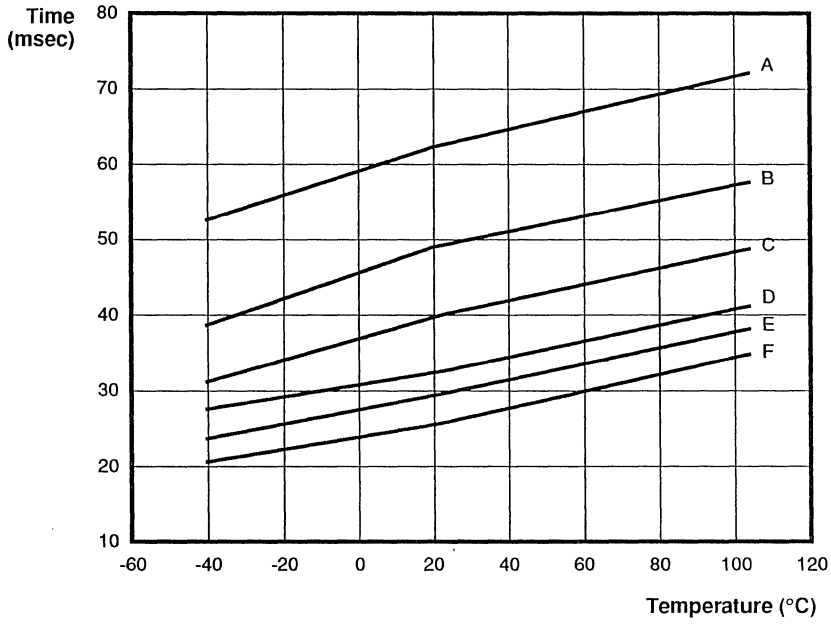
Note: * The internal clock frequency is one half the external clock frequency.
This chart for reference only. Each process will have a different characteristc curve.
R = 1 kohm

Figure 49. Typical Internal Frequency vs RC Capacitance



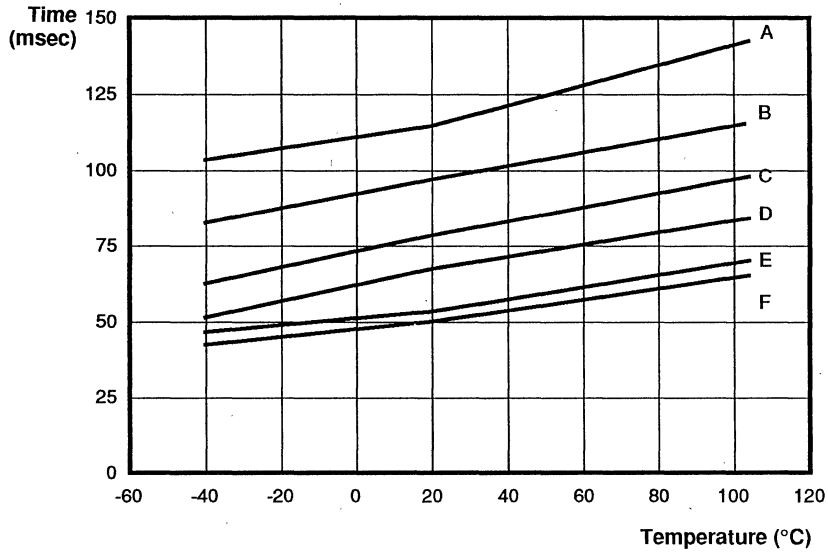
Legend:	
A - Vcc = 3.0V	D - Vcc = 4.5V
B - Vcc = 3.5V	E - Vcc = 5.0V
C - Vcc = 4.0V	F - Vcc = 5.5V

Figure 50. Typical 5 ms WDT Setting vs Temperature



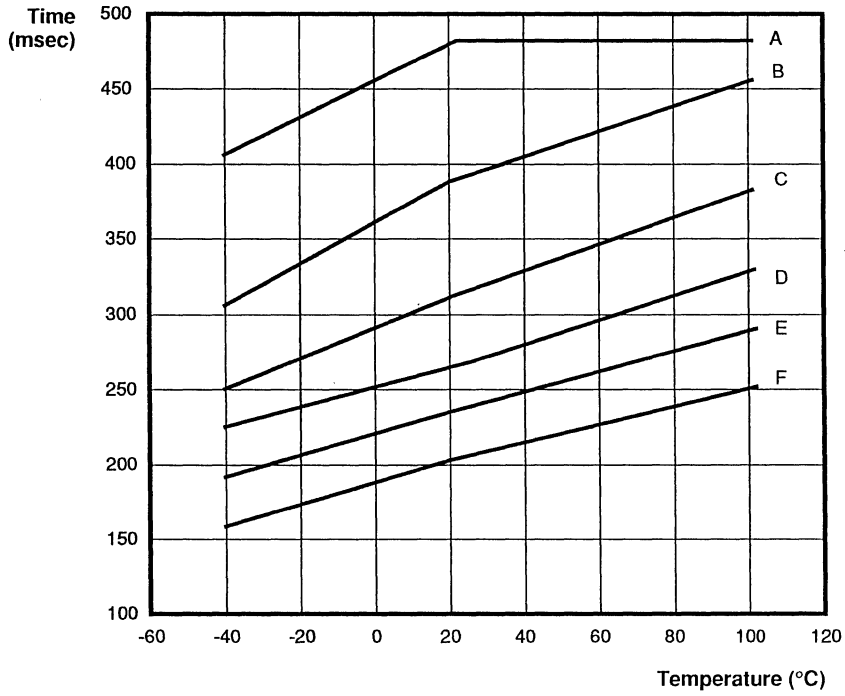
Legend:	
A - Vcc = 3.0V	D - Vcc = 4.5V
B - Vcc = 3.5V	E - Vcc = 5.0V
C - Vcc = 4.0V	F - Vcc = 5.5V

Figure 51. Typical 15 ms WDT Setting vs Temperature



Legend:	
A - Vcc = 3.0V	D - Vcc = 4.5V
B - Vcc = 3.5V	E - Vcc = 5.0V
C - Vcc = 4.0V	F - Vcc = 5.5V

Figure 52. Typical 25 ms WDT Setting vs Temperature



Legend:	
A - Vcc = 3.0V	D - Vcc = 4.5V
B - Vcc = 3.5V	E - Vcc = 5.0V
C - Vcc = 4.0V	F - Vcc = 5.5V

Figure 53. Typical 100 ms WDT Setting vs Temperature

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

Symbol	Meaning
IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

Symbol	Meaning
dst	Destination location or contents
src	Source location or contents
cc	Condition code
@	Indirect address prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag register (Control Register 252)
RP	Register Pointer (R253)
IMR	Interrupt mask register (R251)

Flags. Control register (R252) contains the following six flags:

Symbol	Meaning
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

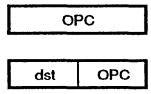
Affected flages are indicated by:

0	Clear to zero
1	Set to one
*	Set to clear according to operation
-	Unaffected
x	Undefined

CONDITION CODES

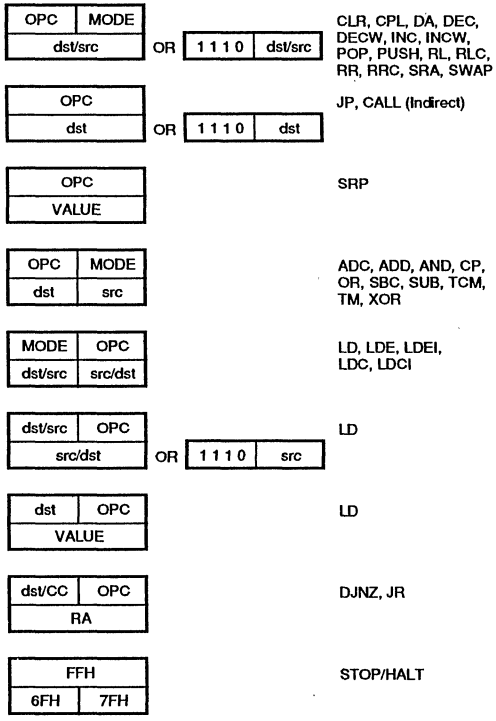
Value	Mnemonic	Meaning	Flags Set
1000		Always True	
0111	C	Carry	C = 1
1111	NC	No Carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not Zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No Overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not Equal	Z = 0
1001	GE	Greater Than or Equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater Than	[Z OR (S XOR V)] = 0
0010	LE	Less Than or Equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned Greater Than or Equal	C = 0
0111	ULT	Unsigned Less Than	C = 1
1011	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned Less Than or Equal	(C OR Z) = 1
0000		Never True	

INSTRUCTION FORMATS

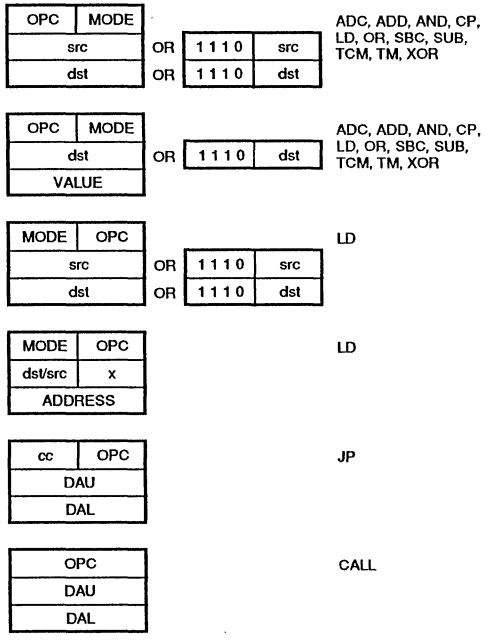


CCF, DI, EI, IRET, NOP, RCF, RET, SCF

One-Byte Instructions



Two-Byte Instructions



Three-Byte Instructions

INSTRUCTION SUMMARY

Note: Assignment of a value is indicated by the symbol " ← ". For example:

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location.

The notation "addr (n)" is used to refer to bit (n) of a given operand location. For example:

dst (7)

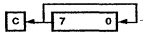
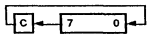
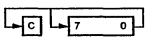
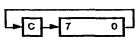
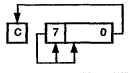
refers to bit 7 of the destination operand.

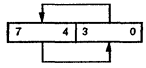
INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst, src dst←dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst←dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst←dst AND src	†		5[ ]	-	*	*	0	-	-
CALL dst SP←SP - 2 @SP←PC, PC←dst	DA IRR		D6 D4	-	-	-	-	-	-
CCF C←NOT C			EF	*	-	-	-	-	-
CLR dst dst←0	R IR		B0 B1	-	-	-	-	-	-
COM dst dst←NOT dst	R IR		60 61	-	*	*	0	-	-
CP dst, src dst - src	†		A[ ]	*	*	*	*	-	-
DA dst dst←DA dst	R IR		40 41	*	*	*	X	-	-
DEC dst dst←dst - 1	R IR		00 01	-	*	*	*	-	-
DECW dst dst←dst - 1	RR IR		80 81	-	*	*	*	-	-
DI IMR(7)←0			8F	-	-	-	-	-	-
DJNZr, dst r←r - 1 if r ≠ 0 PC←PC + dst Range: +127, -128	RA		rA r = 0 - F	-	-	-	-	-	-
EI IMR(7)←1			9F	-	-	-	-	-	-
HALT			7F	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
INC dst dst←dst + 1	r R IR		rE r = 0 - F 20 21	-	*	*	*	-	-
INCW dst dst←dst + 1	RR IR		A0 A1	-	*	*	*	-	-
IRET FLAGS←@SP; SP←SP + 1 PC←@SP; SP←SP + 2; IMR(7)←1			BF	*	*	*	*	*	*
JP cc, dst if cc is true PC←dst	DA IRR		cD c = 0 - F 30	-	-	-	-	-	-
JR cc, dst if cc is true, PC←PC + dst Range: +127, -128	RA		cB c = 0 - F	-	-	-	-	-	-
LD dst, src dst←src	r r R r X r Ir R R R IR	Im R r X r Ir r R IR IM IM R	rC r8 r9 r = 0 - F C7 D7 E3 F3 E4 E5 E6 E7 F5	-	-	-	-	-	-
LDC dst, src	r	Irr	C2	-	-	-	-	-	-
LDCI dst, src dst←src r←r +1; rr←rr + 1	Ir	Irr	C3	-	-	-	-	-	-

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
NOP			FF	-	-	-	-	-	-
OR dst, src dst←dst OR src	†		4[ ]	-	*	*	0	-	-
POP dst dst←@SP; SP←SP + 1	R IR		50 51	-	-	-	-	-	-
PUSH src SP←SP - 1; @SP←src		R IR	70 71	-	-	-	-	-	-
RCF C←0			CF	0	-	-	-	-	-
RET PC←@SP; SP←SP + 2			AF	-	-	-	-	-	-
RL dst 	R IR		90 91	*	*	*	*	-	-
RLC dst 	R IR		10 11	*	*	*	*	-	-
RR dst 	R IR		E0 E1	*	*	*	*	-	-
RRC dst 	R IR		C0 C1	*	*	*	*	-	-
SBC dst, src dst←dst←src←C	†		3[ ]	*	*	*	*	1	*
SCF C←1			DF	1	-	-	-	-	-
SRA dst 	R IR		D0 D1	*	*	*	0	-	-
SRP src RP←src		Im	31	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
STOP			6F	-	-	-	-	-	-
SUB dst, src dst←dst←src	†		2[ ]	*	*	*	*	1	*
SWAP dst 	R IR		F0 F1	X	*	*	X	-	-
TCM dst, src (NOT dst) AND src	†		6[ ]	-	*	*	0	-	-
TM dst, src dst AND src	†		7[ ]	-	*	*	0	-	-
XOR dst, src dst←dst XOR src	†		B[ ]	-	*	*	0	-	-

† These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

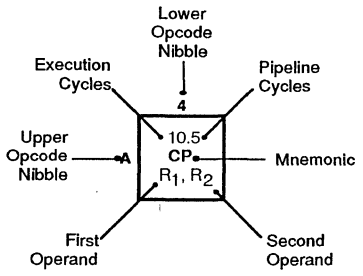
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address Mode dst	src	Lower Opcode Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

OPCODE MAP

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								6.0 **WHD**
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								6.0 **WDT**
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								6.0 **STOP**
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								7.0 **HALT**
8	10.5 **DECW** RR1	10.5 **DECW** IR1														6.1 **DI**
9	6.5 **RL** R1	6.5 **RL** IR1														6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14.0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2				10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1			20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6.5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2			3					2					3	1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_2 = Dst address
R_1 or r_2 = Src address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank are not defined.

* 2-byte instruction appears as a 3-byte instruction

Z86E30

CMOS Z8® OTP CCP™ MICROCONTROLLER

FEATURES

- 8-bit CMOS microcomputer, 28-pin DIP
- Low cost
- 4.0 to 5.5 volt operating range
- Software programmable low EMI mode
- Pull-Up Active/Open Drain programmable on Ports 0 and 2
- EPROM protect programmable
- RAM protect programmable
- RC oscillator programmable
- Low power consumption - 60 mW
- Fast instruction pointer - 0.6 microseconds
- Two standby modes - STOP and HALT
- 24 input/output lines
- All digital inputs, CMOS levels, Schmitt triggered
- 4 Kbytes of one-time PROM
- 236 bytes of RAM
- Three Expanded Register File control registers
- Two programmable 8-bit Counter/Timers each with a 6-bit programmable prescaler.
- Six vectored, priority interrupts from six different sources
- Clock speeds up to 12 MHz
- Watchdog Timer
- Auto Power-On Reset
- Two Comparators
- On-chip oscillator that accepts a crystal, ceramic resonator, LC, RC or external clock drive.

GENERAL DESCRIPTION

The Z86E30 CCP (Consumer Controller Processor) introduces the next level of sophistication to single-chip architecture. The Z86E30 is a member of the Z8 single-chip microcontroller family with 4 Kbytes of EPROM and 236 bytes of RAM. The device is housed in a 28-pin DIP, and is manufactured in CMOS technology. The device offers easy software development and debug, prototyping, and small production runs not economically desirable with a masked ROM version.

The Z86E30 architecture is characterized by Zilog's 8-bit microcontroller core with an expanded register file to allow easy access to register mapped peripheral and I/O circuits. The CCP offers a flexible I/O scheme, an efficient register and address space structure, and a number of ancillary features that are useful in many industrial, high volume, automotive, peripheral types, and advanced scientific applications.

The device applications demand powerful I/O capabilities. The CCP fulfills this with 24 pins dedicated to input and output. These lines are grouped into three ports, eight lines per port, and are configurable under software control to provide timing, status signals, and parallel I/O with or without handshake.

GENERAL DESCRIPTION (Continued)

There are three basic address spaces available to support this wide range of configurations: Program Memory, Register File, and Expanded Register File (ERF). The Register File is composed of 236 bytes of general-purpose registers, three I/O port registers and 15 control and status registers. The Expanded Register File consists of three control resisters.

To unburden the program from coping with the real-time problems such as counting/timing and input/output data communication, the Z86E30 offers two on-chip counter/timers with a large number of user selectable modes, and two on-board comparators to process analog signals with a common reference voltage (Figures 1 and 2).

Note: All Signals with a preceding front slash "/", are active Low, e.g.: B//W (Word is active Low); /B/W (BYTE is active Low, only).

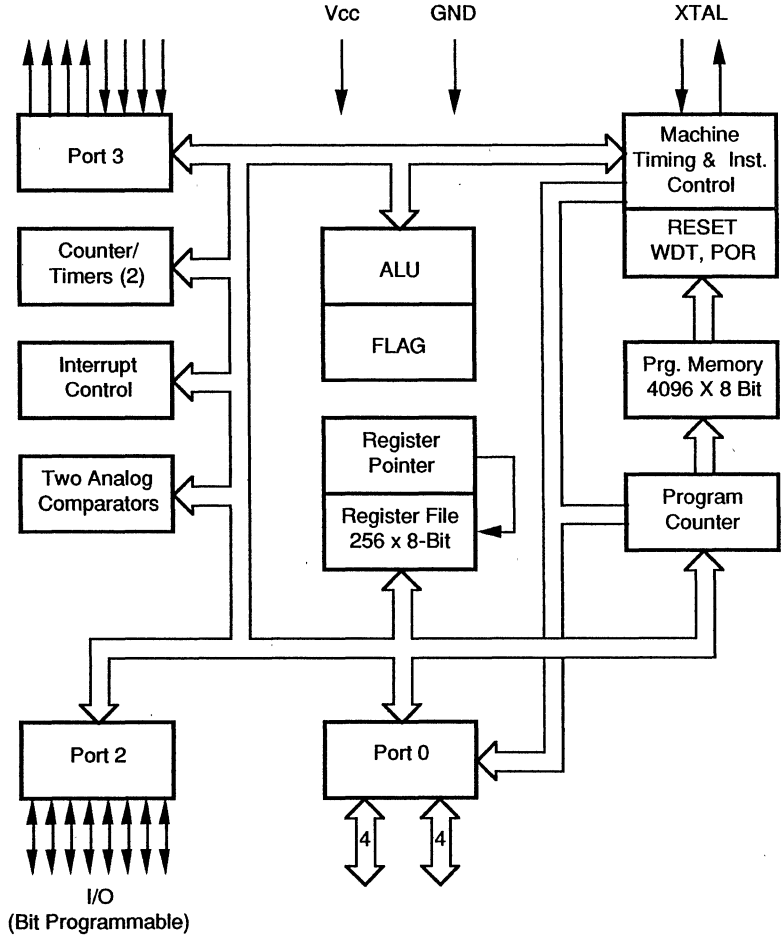


Figure 1. Functional Block Diagram

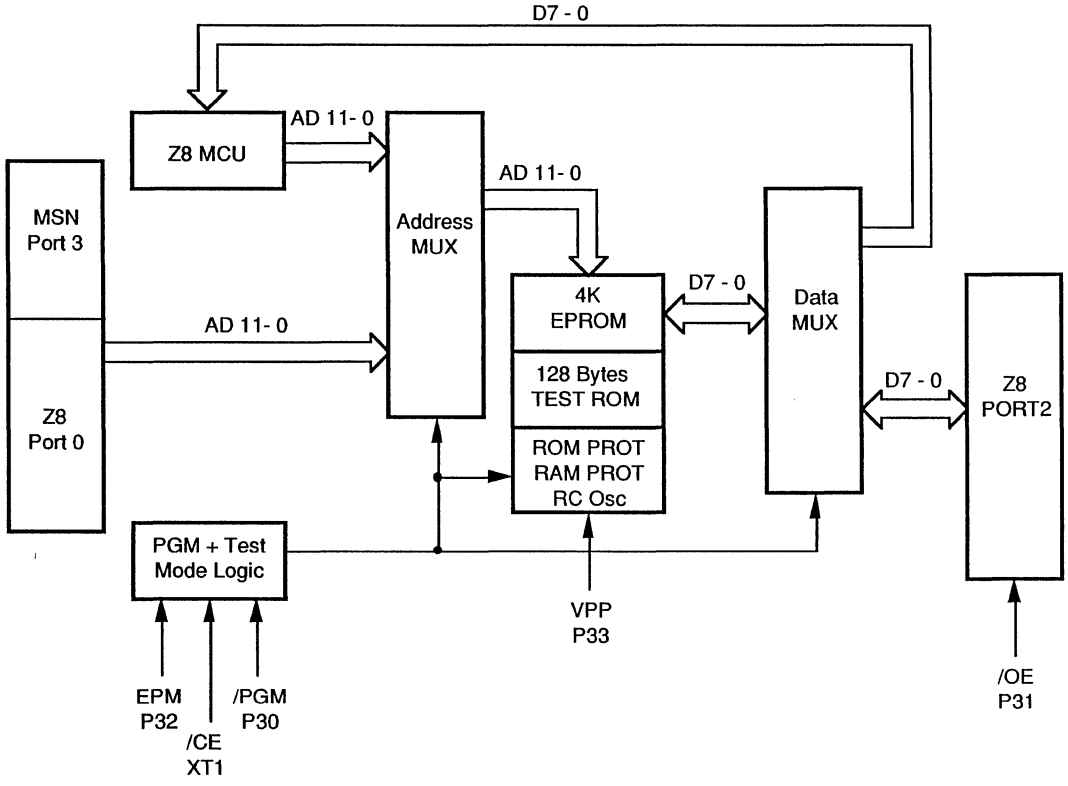


Figure 2. EPROM Programming Block Diagram

PIN DESCRIPTION

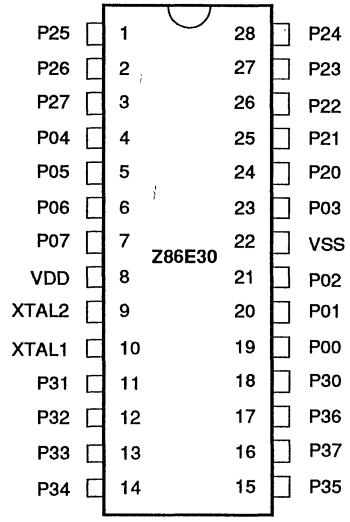
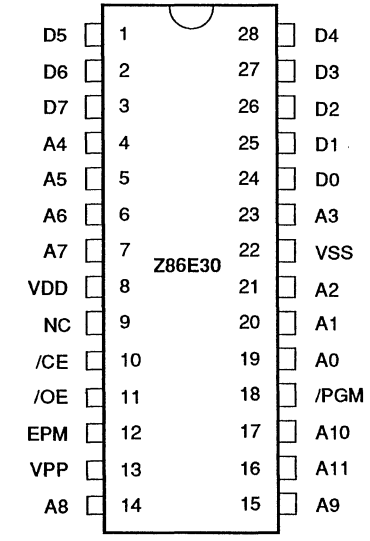


Figure 3. Z86E30 Standard Mode Pin Configuration

Table 1. Z86E30 Standard Mode

Pin #	Symbol	Function	Direction
1-3	P25-7	Port 2 pin 5,6,7	In/Output
4-7	P04-7	Port 0 pin 4,5,6,7	In/Output
8	V_{DD}	Crystal Oscillator	Output
10	XTAL1	Crystal Oscillator	Input
11-13	P31-3	Port 3 pin 1,2,3	Input
14-15	P34-5	Port 3 pin 4,5	Output
16	P37	Port 3 pin 7	Output
17	P36	Port 3 pin 6	Output
18	P30	Port 3 pin 0	Input
19-21	P00-2	Port 0 pin 0,1,2	In/Output
22	V_{SS}	Ground	Input
23	P03	Port 0 pin 3	In/Output
24-28	P20-4	Port 2 pin 0,1,2,3,4	In/Output



Z86E30 EPROM Programming Mode

Figure 4. Z86E30 EPROM Programming Mode Pin Configuration

Table 2. EPROM Programming Mode

Pin #	Symbol	Function	Direction
1-3	D5-7	Data 5,6,7	In/Output
4-7	A4-7	Address 4,5,6,7	Input
8	V_{DD}	Power Supply	Input
9	N/C	No connection	
10	/CE	Chip Select	Input
11	/OE	Output Enable	Input
12	EPM	EPROM Prog. Mode	Input
13	V_{PP}	Prog. Voltage	Input
14-15	A8-9	Address 8,9	Input
16	A11	Address 11	Input
17	A10	Address 10	Input
18	/PGM	Prog. Mode	Input
19-21	A0-2	Address 0,1,2	Input
22	V_{SS}	Ground	Input
23	A3	Address 3	Input
24-28	D0-4	Data 0,1,2,3,4	In/Output

Note: Power connections follow Conventional descriptions below

Connection	Circuit	Device
Power	V_{CC}	V_{DD}
Ground	GND	V_{SS}

PIN FUNCTIONS

EPROM Programming Mode

D7-D0. *Data Bus.* The data can be read from or written to the EPROM through the data bus.

A11-A0. *Address Bus.* During programming, the EPROM address is written to the address bus.

V_{CC}. *Power Supply.* This pin has to supply 5V during the EPROM read mode and 6V during other modes.

/CE. *Chip Enable* (active Low). This pin is active during EPROM Read Mode, Program Mode and Program Verify Mode.

/OE. *Output Enable* (active Low). This pin drives the direction of the Data Bus. When this pin is low, the Data Bus is output. When high, the Data Bus is input.

EPM. *EPROM Program Mode.* This pin controls the different EPROM Program Mode by applying different voltages.

V_{PP}. *Program Voltage.* This pin supplies the program voltage.

/PGM. *Program Mode* (active Low). When this pin is low, the data is programmed to the EPROM through the Data Bus.

Z86E30 Standard Mode

XTAL1. *Crystal 1* (time-based input). This pin connects a parallel-resonant crystal, ceramic resonator, LC, RC network or external single-phase clock to the on-chip oscillator input.

XTAL2. *Crystal 2* (time-based output). This pin connects a parallel-resonant crystal, ceramic resonator, LC or RC network to the on-chip oscillator output.

Port 0 (P00-P07). Port 0 is an 8-bit, bidirectional, CMOS compatible I/O port. These eight I/O lines can be nibble programmed as P00-P03 input/output and P04-P07 input/output, separately. The input buffers are Schmitt triggered and nibbles programmed as outputs can be globally programmed as either push-pull or open drain. Low EMI output buffers can be globally programmed by the software. Port 0 can also be used as a handshake I/O port.

In Handshake Mode, Port 3 lines P32 and P35 are used as handshake control lines. The handshake direction is determined by the configuration (input or output) assigned to Port 0's upper nibble. The lower nibble must have the same direction as the upper nibble (Figure 5).

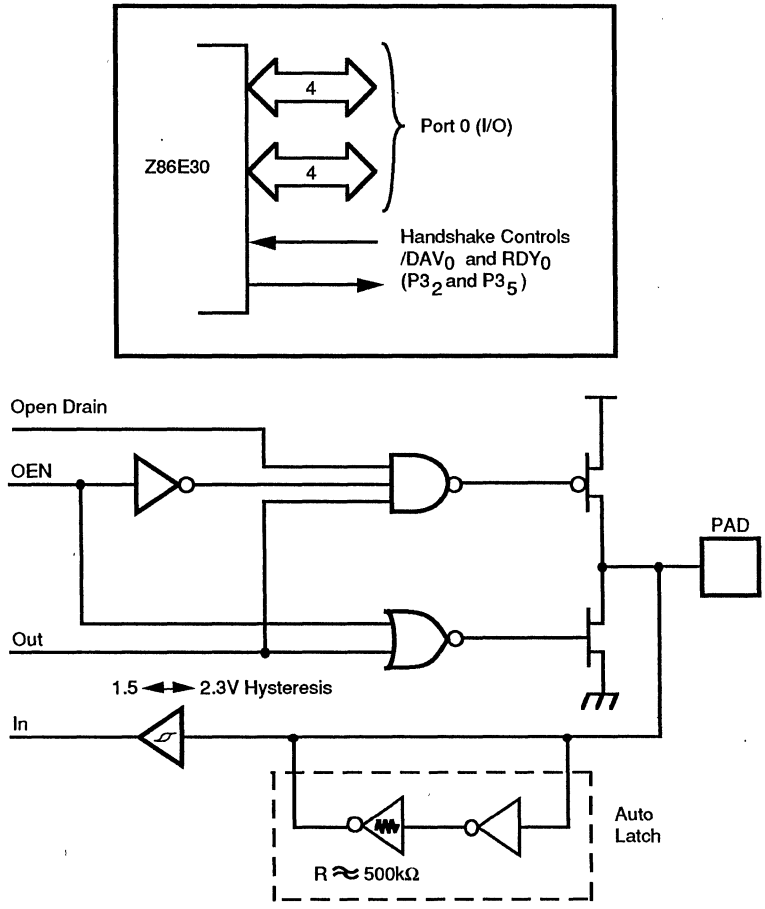


Figure 5. Port 0 Configuration

Port 2 (P20-P27). Port 2 is an 8-bit, bidirectional, CMOS compatible I/O port. These eight I/O lines can be configured under software control as an input or output, independently. All input buffers are Schmitt triggered. Bits programmed as outputs can be globally programmed as either push-pull or open drain. Low EMI output buffers can be globally programmed by the software. When used as an I/O port, Port 2 can be placed under handshake control.

In Handshake Mode, Port 3 lines P31 and P36 are used as handshake control lines. The handshake direction is determined by the configuration (input or output) assigned to bit 7 of Port 2 (Figure 6).

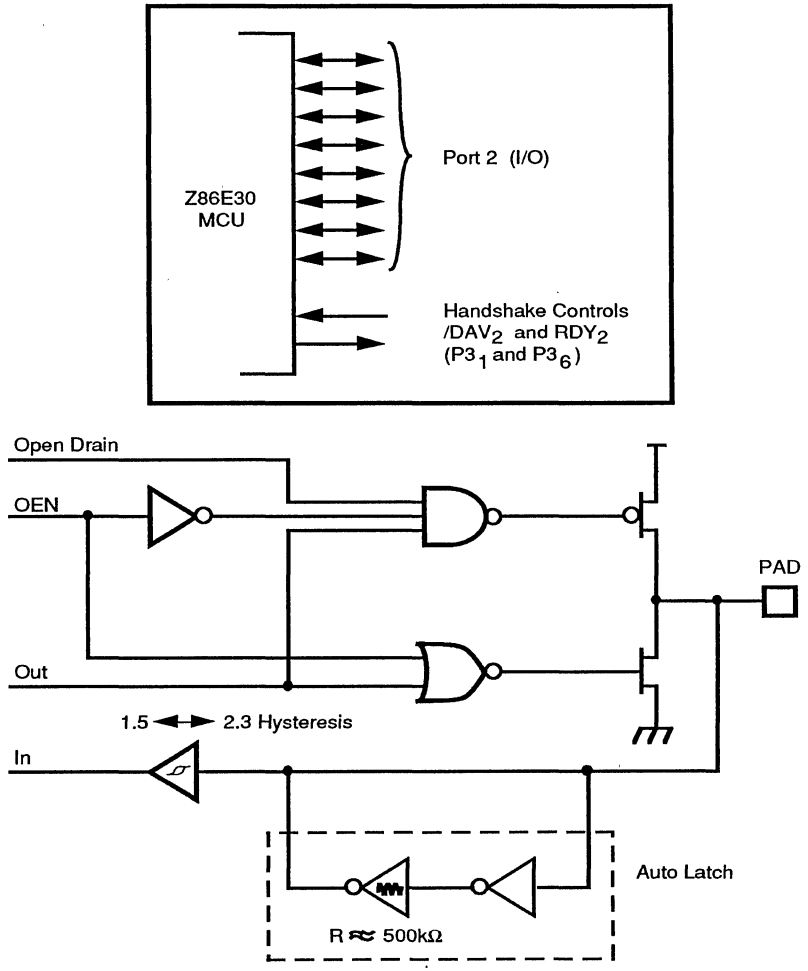


Figure 6. Port 2 Configuration

Port 3 (P30-P37). Port 3 is an 8-bit, CMOS compatible port with four fixed inputs and four fixed outputs. Port 3 consists of four fixed inputs (P30-P33) and four fixed outputs (P34-P37), and can be configured under software for interrupt and handshake control functions. Port 3, Pin 0 is Schmitt triggered. Pins P31, P32 and P33 are standard CMOS inputs (no Auto-Latches) and Pins P34, P35, P36 and P37 are push-pull output lines. Low EMI output buffers can be globally programmed by software. Two on-board comparators can process analog signals on P31 and P32 with reference to the voltage on P33.

The analog function is enabled by setting the D1 of Port 3 Mode Register (P3M). For the interrupt function, P30 and P33 are falling edge triggered interrupt inputs. P31 and P32 can be programmed as falling, rising or both edge triggered interrupt inputs (Figure 7). Access to Counter/Timer 1 is made through P31 (T_{IN}) and P36 (T_{OUT}). Handshake lines for Ports 0 and 2 are also available on Port 3 (Table 3).

Note: P30-P33 inputs differ from the Z86C30 because there is no clamping diode to V_{CC} due to the EPROM high voltage detection circuits. Exceeding the V_{IH} maximum specification during standard operating mode may cause the device to enter EPROM mode.

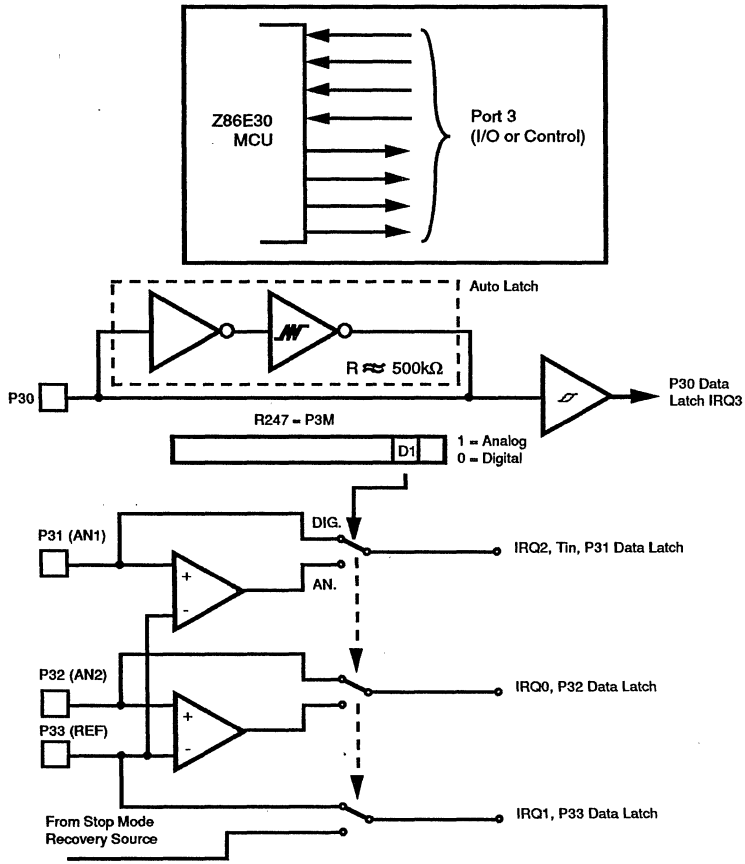


Figure 7. Port 3 Configuration

Table 3. Pin Assignments of Port 3

Pin	I/O	CTC1	AN IN	Int.	P0 HS	P2 HS
P30	IN			IRQ3		
P31	IN	T_{IN}	AN1	IRQ2		D/R
P32	IN		AN2	IRQ0	D/R	
P33	IN		REF	IRQ1		
P34	OUT					
P35	OUT				R/D	
P36	OUT	T_{OUT}				R/D
P37	OUT					

Comparator Inputs. Port 3, pins P31 and P32 each have a comparator front end. The comparator reference voltage (Pin P33) is common to both comparators. In analog mode, P31 and P32 are the positive inputs of the comparators and P33 is the reference voltage of the comparators.

Auto-Latch. The Auto-Latch puts valid CMOS levels on all CMOS inputs (except P31-P33) that are not externally driven. Whether this is zero or one, cannot be determined. A valid CMOS level, rather than a floating node, reduces excessive supply current flow in the input buffer.

Low EMI Emission. The Z86E30 can be programmed to operate in a low EMI emission mode in the PCON register. The oscillator and all I/O ports can be programmed as low EMI emission mode independently. Use of this feature results in:

- Less than 1.5 mA (typical) current consumption during HALT mode.
- The pre-drivers slew rate reduced to 10 ns typical.
- Low EMI output drivers have resistance of 200 ohms (typical).
- Oscillator divide-by-two circuitry is eliminated.
- Internal SLCK/TCLK operation limited to a maximum of 4 MHz - 250 ns cycle time.

FUNCTIONAL DESCRIPTION

The Z8 CCP incorporates special functions to enhance the Z8's applications in industrial, scientific research, and advanced technologies.

RESET. The device is reset in one of the following conditions:

- Power-On Reset
- Watch-Dog Timer
- STOP Mode Recovery Source

Having the Auto Power-on Reset circuitry built in, the Z86E30 does not need to be connected to an external power-on reset circuit. The reset time is 5 ms (typical) plus 18 clock cycles. The Z86E30 does not re-initialize WDTMR, SMR, P2M, and P3M registers to their reset values on a STOP Mode Recovery operation.

Program Memory. The Z86E30 can address up to 4 Kbytes of internal program memory (Figure 8). The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts. Address 12 (000CH) to address 4095 (0FFFH) are reserved for the user program. After reset, the program counter points at the address 000CH which is the starting address of the user program.

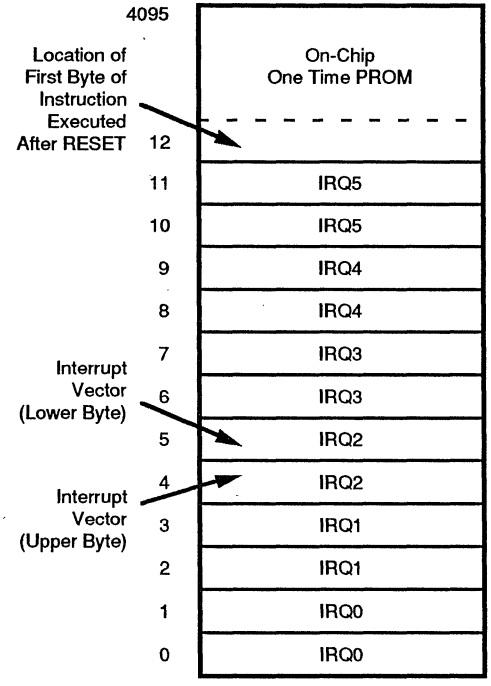


Figure 8. Program Memory Map

EPROM Protect. The 4 Kbytes program memory is a one time PROM. An EPROM protect feature prevents "dumping" of the ROM contents by inhibiting execution of LDC and LDCI instructions (LDE and LDEI instructions are not available in Z86E30) to program memory in all modes. In EPROM protect mode, the instructions of LDC and LDCI are disabled.

Expanded Register File. The register file has been expanded to allow for additional system control registers, mapping of additional peripheral devices, and input/output ports into the register address area. The Z8 register address space R0 through R15 is implemented as 16 groups of 16 registers per group (Figure 9). These register groups are known as the ERF (Expanded Register File). The low nibble (D0-D3) of the Register Pointer (RP) selects the active ERF group, and the high nibble (D4-D7) of register RP selects the working register group (Figure 10).

Three system configuration registers reside in the Expanded Register File at bank FH: PCON, SMR, and WDTMR. The rest of the Expanded Register is not physically implemented and is reserved for future expansion.

Register File. The 256 byte register file consists of 3 I/O port registers, 236 general-purpose registers, and 15 control and status registers (R3 and R240 are reserved), and three system configuration registers in the expanded register group (Figure 9). The instructions can access registers directly or indirectly via an 8-bit address field. This allows a short 4-bit register address using the Register Pointer (Figures 10, 11). In the 4-bit mode, the register file is divided into 16 working register groups, each occupying 16 continuous locations. The Register Pointer addresses the starting location of the active working-register group.

Note: Register Bank E0H-EFH can only be accessed through working registers and indirect addressing modes.

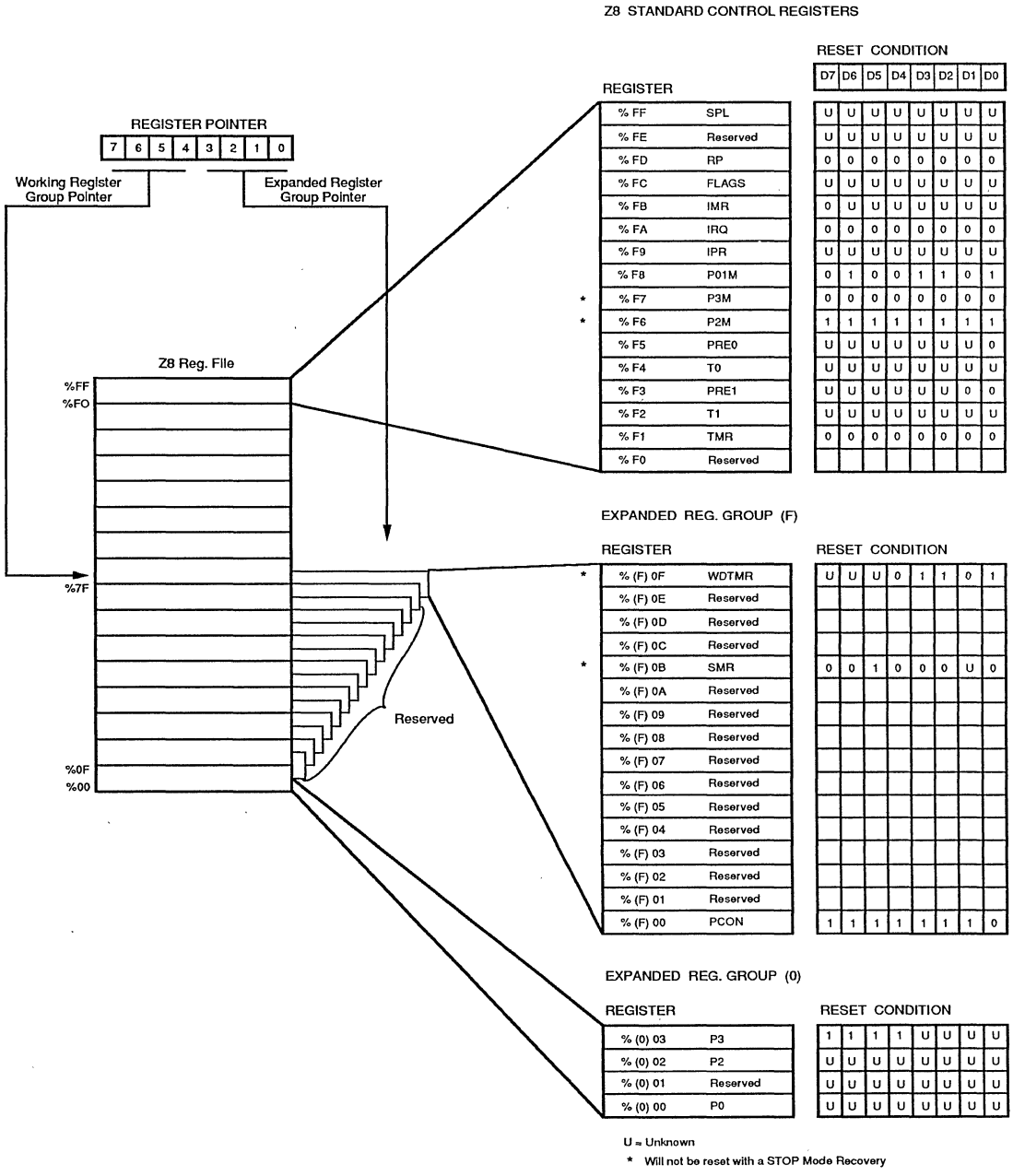


Figure 9. Expanded Register File Architecture

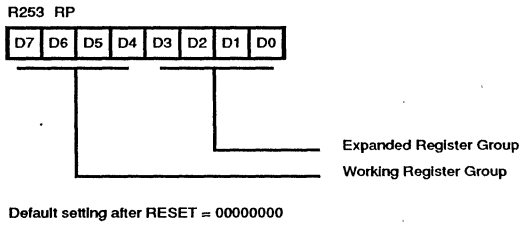


Figure 10. Register Pointer Register

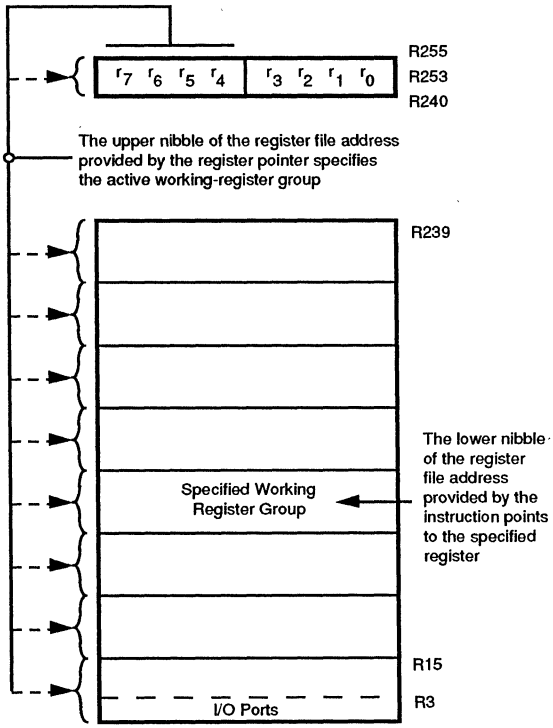


Figure 11. Register Pointer

RAM Protect. The upper portion of the RAM's address space 80H to EFH (excluding the control registers) can be protected from reading and writing. This option can be selected in EPROM Programming Mode. D6 of the IMR Control Register (R251) is used to turn on or turn off the RAM protect. The RAM protect is turned on by setting the D6 of the IMR Control Register (D6=1) and turned off by resetting this bit (D6=0).

Stack. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 256 general-purpose registers.

Counter/Timers. There are two 8-bit programmable counter/timers (T0 and T1), each driven by its own 6-bit programmable prescaler. The T1 prescaler can be driven by internal or external clock sources; however, the T0 prescaler is driven by the internal clock only (Figure 12).

The 6-bit prescalers can divide the input frequency of the clock source by any integer number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request-IRQ4 (T0) or IRQ5 (T1) is generated.

The counters can be programmed to start, stop, restart to continue, or restart from the initial value. The counters can also be programmed to stop upon reaching zero (single pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode).

The counters, but not the prescalers, can be read at any time without disturbing their value or count mode. The clock source for T1 is user-definable and can be either the internal microprocessor clock divided-by-four, or an external signal input via Port 3. The Timer Mode register configures the external timer input (P30) as an external clock, a trigger input that can be retriggerable or not-retriggerable, or as a gate input for the internal clock. Port 3 line P36 serves as a timer output (T_{OUT}) through which T0, T1 or the internal clock is output. The counter/timers are cascaded by connecting the T0 output to the input of T1.

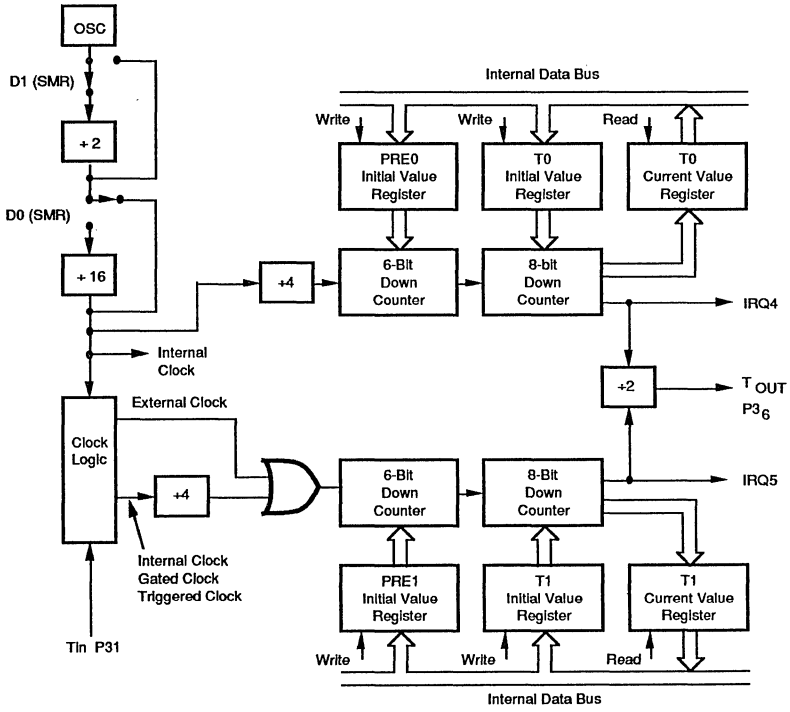


Figure 12. Counter/Timer Block Diagram

FUNCTIONAL DESCRIPTION (Continued)

Interrupts. The Z86E30 has six different interrupts from six different sources. The interrupts are maskable and prioritized (Figure 13). The six sources are divided as follow: four sources are claimed by Port 3 lines P30-P33, and two in counter/timers. The Interrupt Mask Register globally or individually enables or disables the six interrupt requests (Table 4).

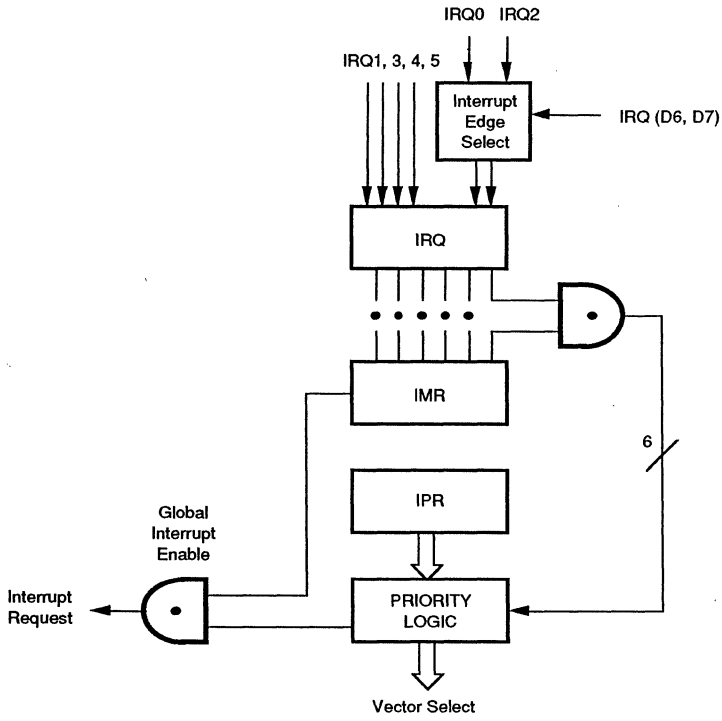


Figure 13. Interrupt Block Diagram

Table 4. Interrupt Types, Sources, and Vectors

Name	Source	Vector Location	Comments
IRQ 0	/DAV 0, IRQ 0	0, 1	External (P32), Rising/Falling Edge Triggered
IRQ 1	IRQ 1	2, 3	External (P33), Falling Edge Triggered
IRQ 2	/DAV 2, IRQ 2, T_{IN}	4, 5	External (P31), Rising/Falling Edge Triggered
IRQ 3	IRQ3	6, 7	External (P30), Falling Edge Triggered
IRQ 4	T0	8, 9	Internal
IRQ 5	T1	10, 11	Internal

When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register. An interrupt machine cycle is activated when an interrupt request is granted. Thus, disabling all subsequent interrupts, saves the Program Counter and Status Flags, and then branches to the program memory vector location reserved for that interrupt. All Z86E30 interrupts are vectored through locations in the program memory. This memory location and the next byte contain the 16-bit starting address of the interrupt service routine for that particular interrupt request.

To accommodate polled interrupt systems, interrupt inputs are masked and the interrupt request register is polled to determine which of the interrupt requests need service.

An interrupt resulting from AN1 is mapped into IRQ2, and an interrupt from AN2 is mapped into IRQ0. Interrupts IRQ2 and IRQ0 may be rising, falling or both edge triggered, and are programmable by the user. The software may poll to identify the state of the pin.

Programming bits for the Interrupt Edge Select are located in bits D7 and D6 of the IRQ Register (R250). The configuration is shown in Table 5.

Table 5. IRQ Register Configuration

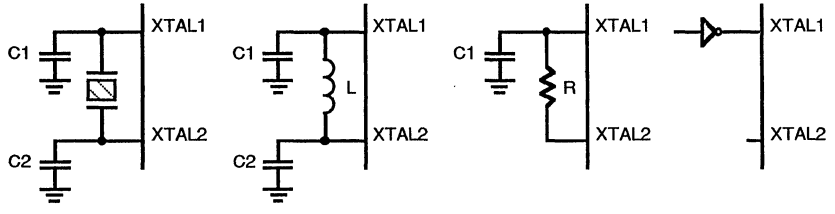
IRQ		Interrupt Edge	
D7	D6	P31	P32
0	0	F	F
0	1	F	R
1	0	R	F
1	1	R/F	R/F

Notes:
F=Falling Edge
R=Rising Edge

Clock. The Z86E30 on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal, RC, ceramic resonator, or any suitable external clock source (XTAL1 = Input, XTAL2 = Output). The crystal should be AT cut, 10 kHz to 12 MHz max, with a series resistance (RS) less than or equal to 100 Ohms.

The crystal should be connected across XTAL1 and XTAL2 using the recommended capacitors (10 pF to 100 pF) from each pin to ground. The RC oscillator option is selected in the programming mode. The RC oscillator configuration must be an external resistor connected from XTAL1 to XTAL2, with a frequency-setting capacitor from XTAL1 to ground (Figure 14).

Note: RC OSC may not reach to 12 MHz



Ceramic Resonator or Crystal
C1, C2 = 47 pF TYP *
f = 8 MHz

LC
C1, C2 = 22 pF

L = 130 uH *
f = 3 MHz *

RC
@ 5V VCC (TYP)

C1 = 33 pF *
R = 1k *
f = 6.6 MHz *

External Clock

* Typical value including pin parasitics

Figure 14. Oscillator Configuration

Power-On Reset (POR). A timer circuit clocked by a dedicated on-board RC oscillator is used for the Power-On Reset (POR) timer function. The POR timer allows V_{CC} and the oscillator circuit to stabilize before instruction execution begins.

The POR timer circuit is a one-shot timer triggered by one of three conditions:

1. Power bad to Power OK status
2. STOP mode recovery (if D5 of SMR=0)
3. WDT timeout

The POR time is a nominal 5 ms. Bit 5 of the Stop Mode Register (SMR) determines whether the POR timer is bypassed after STOP mode recovery (typical for external clock, and RC/LC oscillators with fast start up time).

HALT. Turns off the internal CPU clock, but not the XTAL oscillation. The counter/timers and external interrupts IRQ0, IRQ1, and IRQ2 remain active. The device is recovered by interrupts, either externally or internally generated.

STOP. This instruction turns off the internal clock and external crystal oscillation and reduces the standby current to 10 microamperes or less. The Stop mode is terminated by one of the following resets: WDT timeout, POR, or Stop Mode Recovery Source which is defined by SMR register . This causes the processor to restart the application program at address 000C (HEX). In order to enter STOP (or HALT) mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. To do this, the user must execute a NOP instruction (opcode=FFH) immediately before the appropriate sleep instruction. For example:

```
FF  NOP   ; clear the pipeline
6F  STOP  ; enter STOP mode
or
FF  NOP   ; clear the pipeline
7F  HALT  ; enter HALT mode
```

Port Configuration Register (PCON). The PORT Configuration Register (PCON) configures the port's individually for comparator output on Port 3, Open Drain on Port 0, low EMI noise on Port's 0, 2, and 3, and low EMI noise oscillator. The PCON Register is located in the Expanded Register File at bank F, location 00 (Figure 15).

Note: PCON is not available in Z86C30

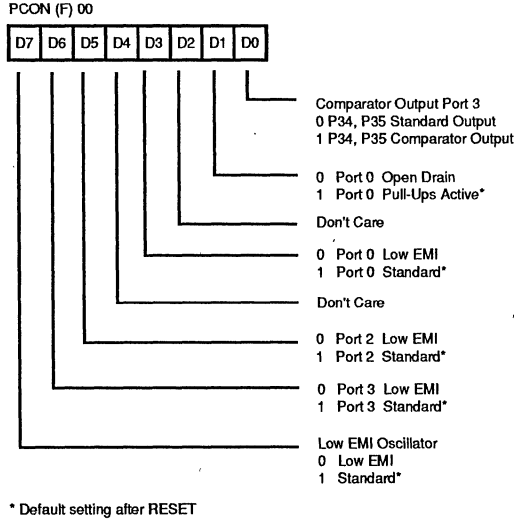


Figure 15. Port Configuration Register (PCON)

Comparator Output Port 3 (D0). Bit 0 controls the comparator use in Port 3. A 1 in this location brings the comparator outputs to P34 and P35 and a 0 releases the Port to its standard I/O configuration.

Port 0 Open Drain (D1). Port 0 is configured as an Open-drain by resetting this bit (D1 = 0) and configured as Pull-up Active by setting D1 = 1. The default value is 1.

Low EMI Port 0 (D3). Port 0 is configured as a Low EMI Port by resetting this bit (D3 =0) and configured as a Standard Port by setting D3 =1. The default value is 1.

Low EMI Port 2 (D5). Port 2 is configured as a Low EMI Port by resetting this bit (D5 = 0) and configured as a Standard Port by setting D5 = 1. The default value is 1.

Low EMI Port 3 (D6). Port 3 is configured as a Low EMI Port by resetting this bit (D6 = 0) and configured as a Standard Port by setting D6 = 1. The default value is 1.

Low EMI OSC (D7). This bit of the PCON Register controls the low EMI noise oscillator. A1 in this location configures the oscillator with standard drive, while a 0 configures the oscillator with low noise drive.

Stop Mode Recovery Register (SMR). This register selects the clock divide value and determines the mode of STOP mode recovery (Figure 16). All bits are Write only except Bit 7 which is a Read only. Bit 7 is a flag bit that is hardware set on the condition of STOP Recovery and reset by a power-on cycle. Bit 6 controls whether a low or high level is required from the recovery source. Bit 5 controls the reset delay after recovery. Bits 2, 3, and 4 of the SMR register specify the STOP Mode Recovery Source. (Table 7). The SMR is located in bank F of the Expanded Register Group at address OBH.

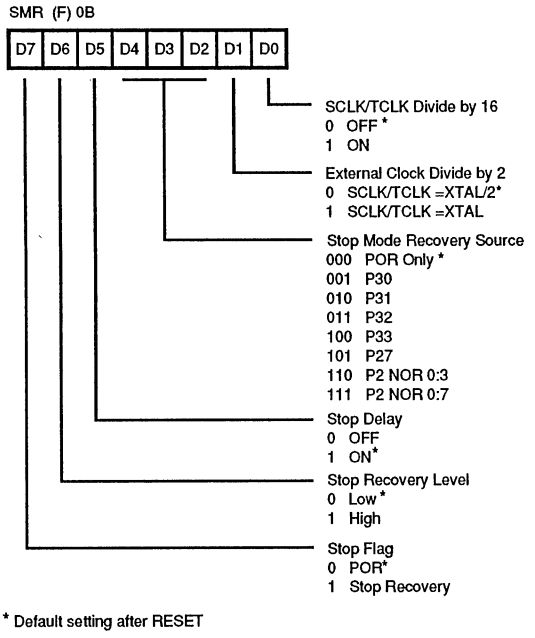


Figure 16. Stop Mode Recovery Register

SCLK/TCLK divide-by-16 Select (D0). This bit of the SMR controls a divide-by-16 prescaler of SCLK/TCLK. The purpose of this control is to selectively reduce device power consumption during normal processor execution (SCLK control) and/or HALT mode (where TCLK sources counter/timers and interrupt logic).

External Clock Divide By 2 (D1). This bit can eliminate the oscillator divide-by-two circuitry. When this bit is 0, SCLK (System Clock) and TCLK (Timer Clock) are equal to the external clock frequency divided by two. The SCLK/TCLK is equal to the external clock frequency when this bit is set (D1 = 1). Using this bit, together with D7 of PCON, further helps lower EMI [i.e., D7 (PCON) = 0, D1 (SMR) = 1]. The default setting is 0.

STOP Mode Recovery Source (D2, D3, and D4). These 3 bits of the SMR register specify the wake-up source of the STOP Mode recovery (Figure 17). Table 6 shows the SMR source selected with the setting of D2 to D4. P31-P33 cannot be used to wake up from STOP mode when programmed as analog inputs.

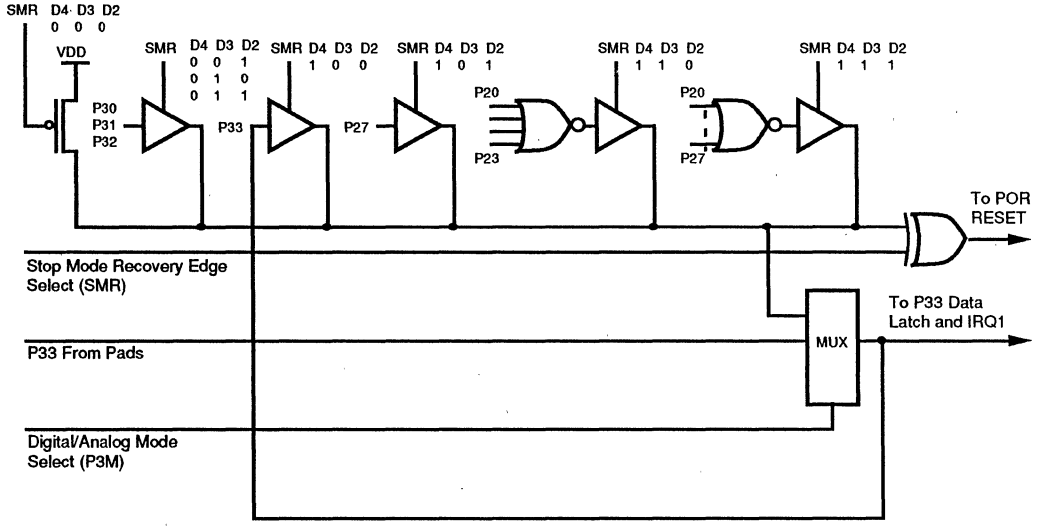


Figure 17. Stop Mode Recovery Source

Table 6. Stop Mode Recovery Source

D4	D3	D2	SMR Source selection
0	0	0	POR recovery only
0	0	1	P30 transition
0	1	0	P31 transition (Not in analog mode)
0	1	1	P32 transition (Not in analog mode)
1	0	0	P33 transition (Not in analog mode)
1	0	1	P27 transition
1	1	0	Logical NOR of Port 2 bits 0:3
1	1	1	Logical NOR of Port 2 bits 0:7

STOP Mode Recovery Delay Select (D5). The 5 ms RESET delay after STOP Mode Recovery is disabled by programming this bit to a zero. A1 in this bit causes a 5 ms RESET delay after STOP Mode Recovery. The default condition of this bit is 1. If the fast wake up mode is selected, the STOP Mode Recovery source must be kept active for at least 5 TpC.

STOP Mode Recovery Level Select (D6). A 1 in this bit defines that a high level on any one of the recovery sources wakes the Z86E30 from STOP Mode. A 0 defines the low level recovery. The default value is 0.

Cold or Warm Start (D7). This bit is set by the device upon entering STOP Mode. A 0 in this bit indicates that the device has been reset by POR (cold). A 1 in this bit indicates the device was awakened by a SMR source (warm).

Watch Dog Timer Mode Register (WDTMR). The WDT is a retriggerable one-shot timer that resets the Z8 if it reaches terminal count. The WDT is disabled after Power-On Reset and initially enabled by executing the WDT instruction. It is refreshed on subsequent executions of the WDT instruction. The WDT cannot be disabled when it has been enabled. The WDT is driven either by an on-board RC oscillator or external oscillator from XTAL1 pin. The POR clock source is selected with bit-4 of the WDT register.

WDT Timeout Period (D0 and D1). Bits 0 and 1 control a tap circuit that determines the timeout periods that can be obtained. Table 7 shows the timeout period. The default value of D0 and D1 are 1 and 0, respectively.

Table 7. Timeout Period of the WDT

D1	D0	Timeout of Internal RC OSC	Timeout of the Crystal Clock
0	0	5 ms	256TpC
0	1	15 ms	512TpC
1	0	25 ms	1024TpC
1	1	100 ms	4096TpC

Notes:
TpC = crystal clock cycle
The default setting is 15 ms.

WDT During the HALT Mode (D2). This bit determines whether or not the WDT is active during HALT mode. A 1 indicates that the WDT is active during HALT. A 0 disables the WDT in HALT mode. The default value is 1.

WDT During STOP Mode (D3). This bit determines whether or not the WDT is active during STOP mode. A 1 indicates active during STOP. A 0 disables the WDT during STOP mode. Since the on-board OSC is stopped during STOP mode, the WDT clock source has to select the on-board RC OSC for the WDT to recover from STOP mode. The default is 1.

Clock Source For WDT (D4). This bit determines which oscillator source is used to clock the internal POR and WDT counter chain. If the bit is a 1, the internal RC oscillator is bypassed and the POR and WDT clock source is driven from the external pin, XTAL1. The default configuration of the bit is 0, which selects the RC oscillator.

Bits 5 through 7 are reserved. The WDTMR register is accessible only during the first 64 processor cycles (128 XTAL clock cycles) from the execution of the first instruction after Power-On Reset, watch dog reset, or a STOP mode recovery. After this point, the register cannot be modified by any means, intentional or otherwise. The WDTMR cannot be read and is located in bank F of the Expanded Register Group at address location OFH.

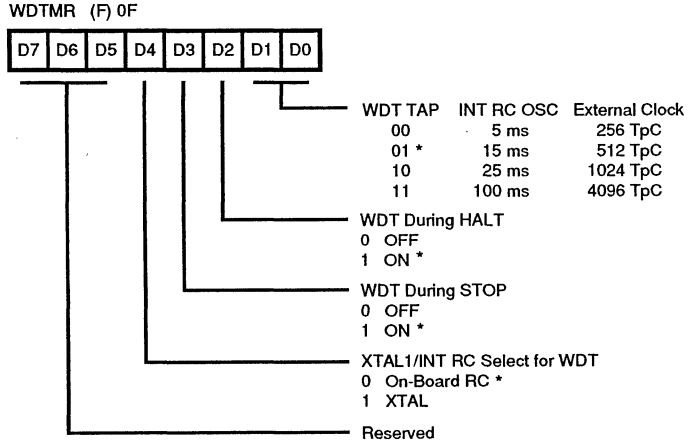


Figure 18. Watchdog Timer Mode Register

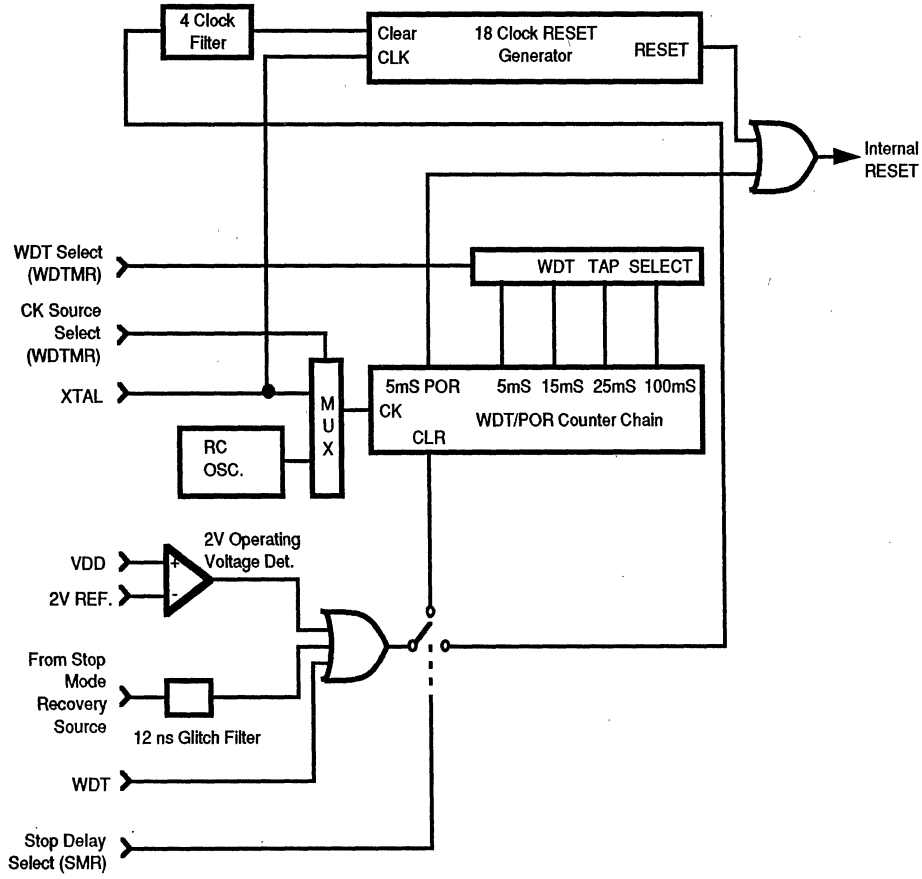


Figure 19. Resets and WDT

Auto Reset Voltage. An on-board Voltage Comparator checks that V_{cc} is at the required level to ensure correct operation of the device. Reset is globally driven if V_{cc} is below V_{RST} (Auto Reset Voltage - Figure 26).

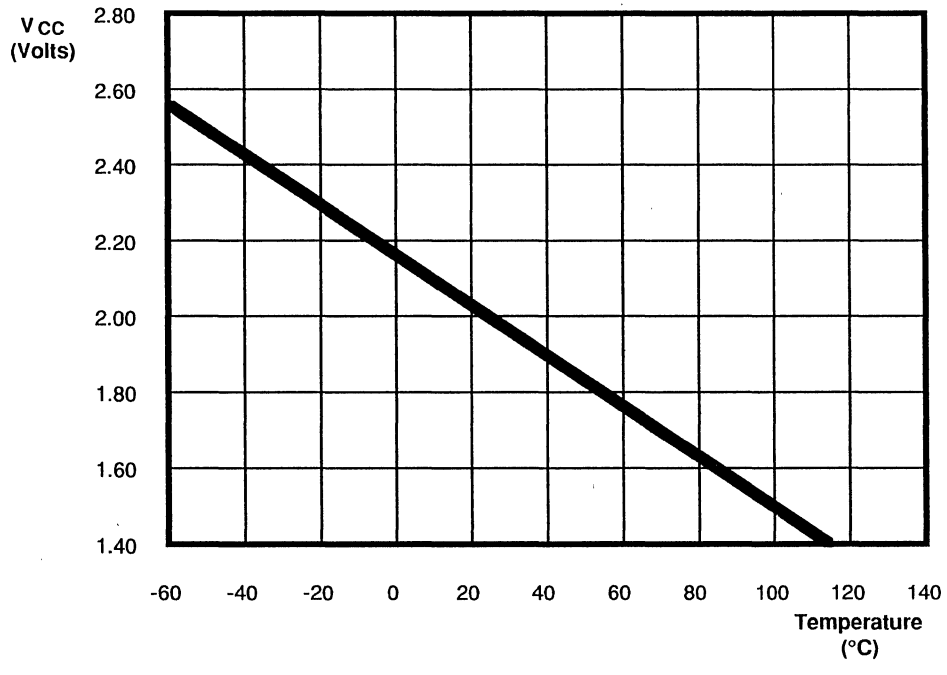


Figure 20. Typical Z86E30 V_{RST} Voltage vs. Temperature

FUNCTIONAL DESCRIPTION (Continued)

EPROM Programming Mode

Table 8 shows the programming voltages of each programming mode. Table 9, Figures 21, 22, and 23 show the programming timing of each programming mode. Figure 24 shows the flow-chart of an Intelligent Programming Algorithm, which is compatible with a 2764A EPROM (Z86E30 is 4K EPROM, 2764A is 8K EPROM). Figure 25 shows the circuit diagram of the Z86E30 programming adaptor which adapts from 2764A to Z86E30. Since the EPROM size of Z86E30 differs from 2764A, the programming address range should be set from 0000H to 0FFFH. Otherwise, the upper 4K of data (1000H-1FFFH) will overwrite the lower 4K of data.

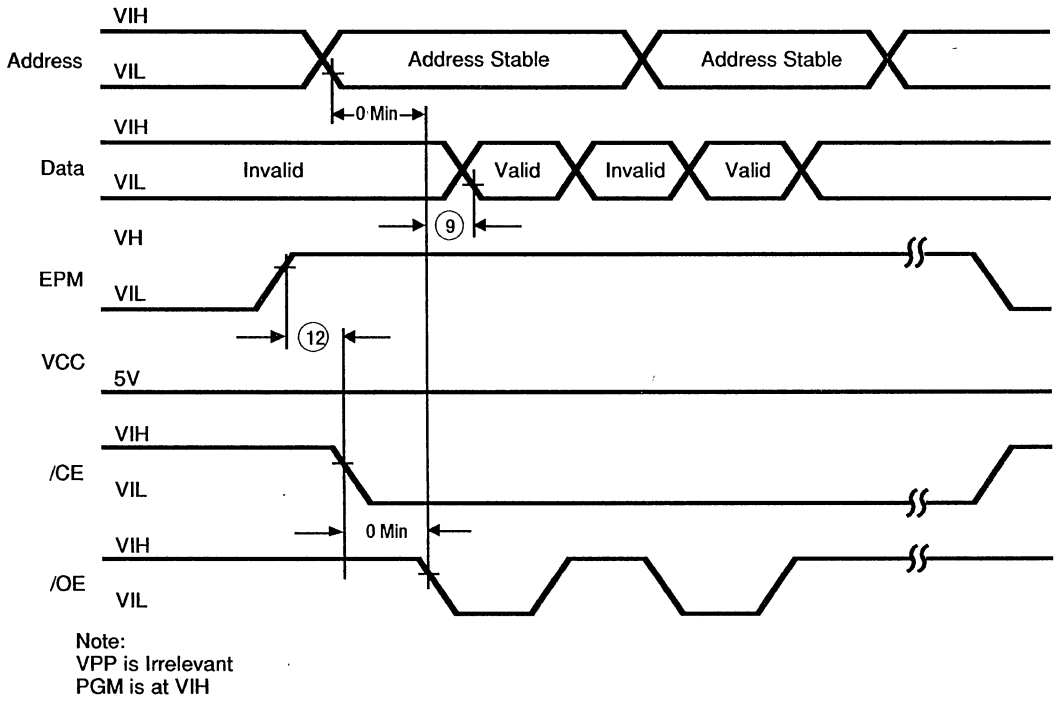
Table 8. EPROM Programming Table

Programming Modes	V_{PP} (P33)	EPM (P32)	/CE (XTAL1)	/OE (P31)	/PGM (P30)	ADDR	DATA (PORT2)	V_{CC}
EPROM READ	X	V_H	V_{IL}	V_{IL}	V_{IH}	Addr	Data Out	5.0V
PROGRAM	V_{PP}	X	V_{IL}	V_{IH}	V_{IL}	Addr	Data In	6.0V
PROGRAM VERIFY	V_{PP}	X	V_{IL}	V_{IL}	V_{IH}	Addr	Data Out	6.0V
EPROM PROTECT	V_{PP}	V_H	V_H	V_{IH}	V_{IL}	X	X	6.0V
RAM PROTECT	V_{PP}	V_{IH}	V_H	V_{IL}	V_{IL}	X	X	6.0V
RC OSCILLATOR	V_{PP}	V_{IL}	V_H	V_{IH}	V_{IL}	X	X	6.0V

Notes:
V_{PP} = 12.5V ± 0.5V
V_H = 12.5V ± 0.5V
X = TTL Level (irrelevant)
V_{IH} = 5.0V
V_{IL} = 0V

Table 9. EPROM Programming Timing

Parameters	Name	Min	Max	Units
1	Address Setup Time	2		µs
2	Data Setup Time	2		µs
3	V_{PP} Setup Time	2		µs
4	V_{CC} Setup Time	2		µs
5	Chip Enable Setup	2		µs
6	Program Pulse Width	0.95	1.05	ms
7	Data Hold Time	2		µs
8	OE Setup Time	2		µs
9	Data Access Time		200	ns
10	Data Output Float Time		100	ns
11	Overprogram Pulse Width	2.85	78.75	ms
12	EPM Setup Time	2		µs
13	OE Setup Time	2		µs
14	Address to OE Setup Time	2		µs
15	Option Bit Program Pulse Width		78.75	ms



Note:
VPP is Irrelevant
PGM is at VIH

Figure 21. EPROM READ Mode Timing Diagram

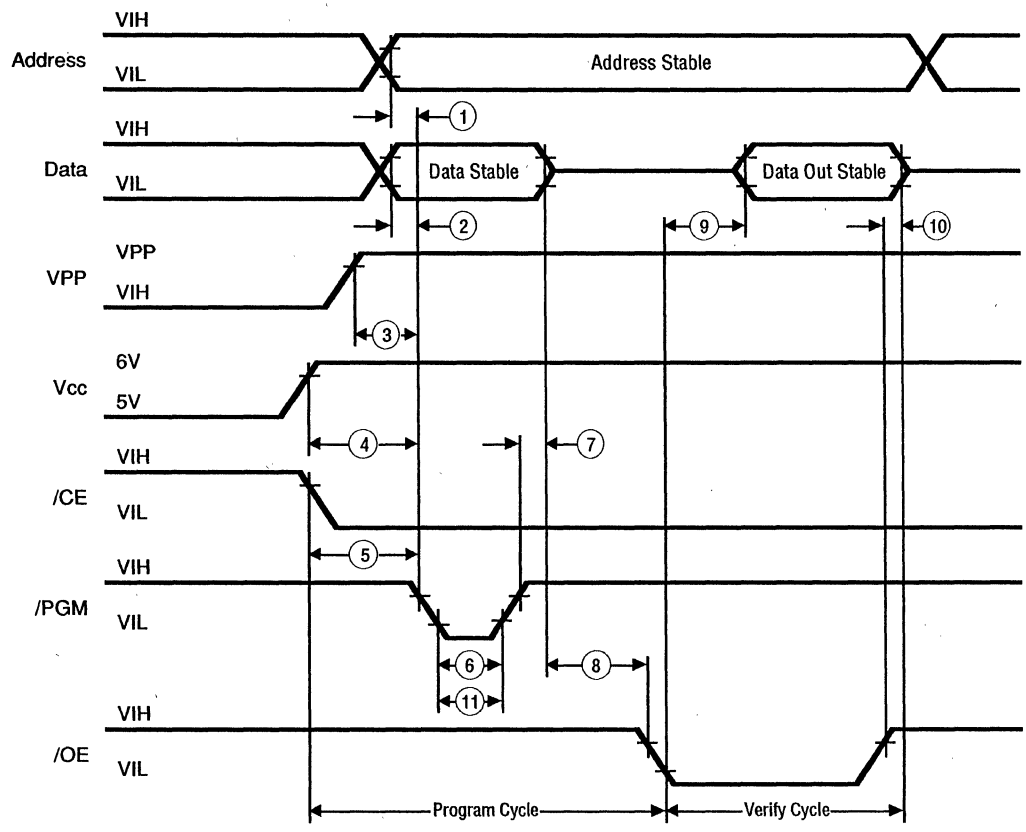


Figure 22. Timing Diagram of EPROM Program and Verify Modes

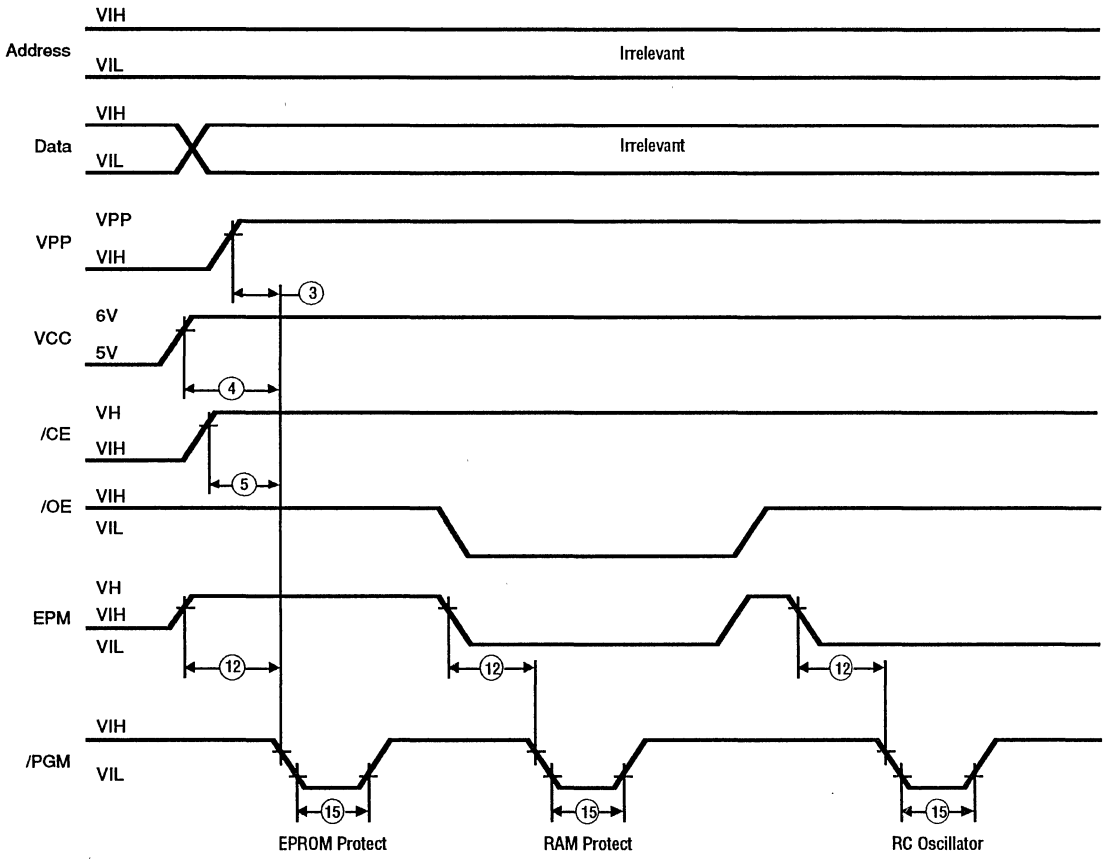


Figure 23. Timing Diagram of EPROM Protect, RAM Protect, and RC OSC Modes

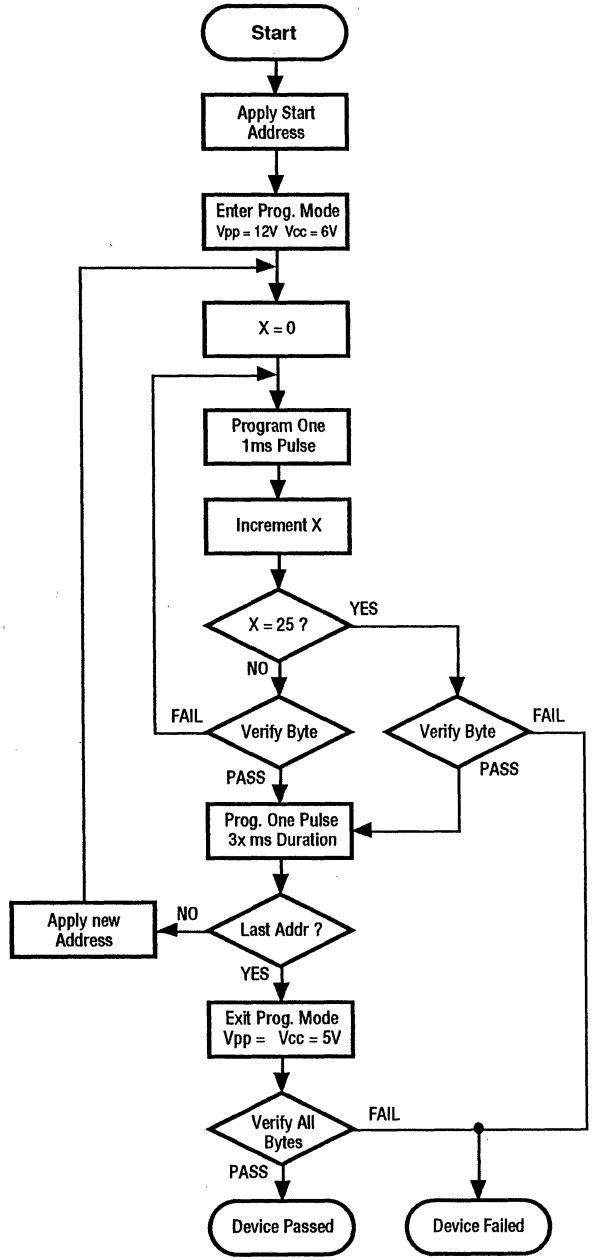
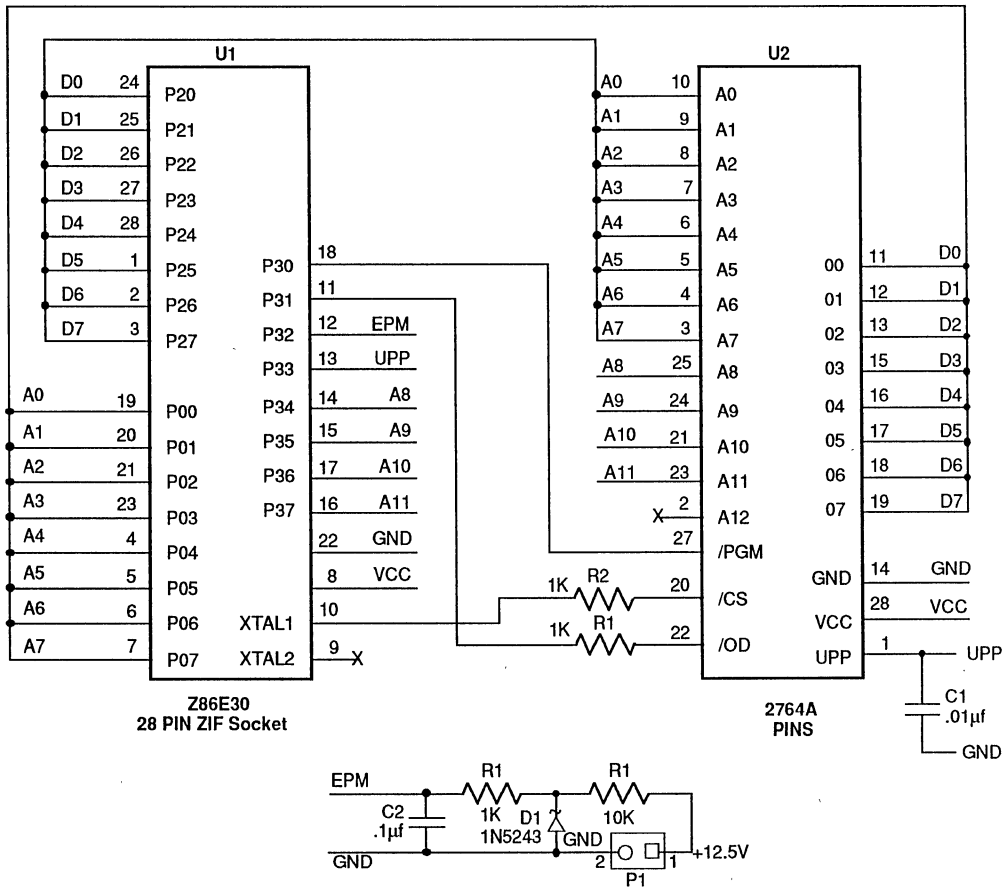


Figure 24. Z86E30 Programming Algorithm



Note: The programming address has to be set to 0000H -0FFFH (lower 4K byte memory)

Figure 25. Z86E30 Programming Adaptor Circuitry

STANDARD TEST CONDITIONS

The characteristics listed below apply for standard test conditions as noted. All voltages are referenced to GND. Positive current flows into the referenced pin (Figure 26).

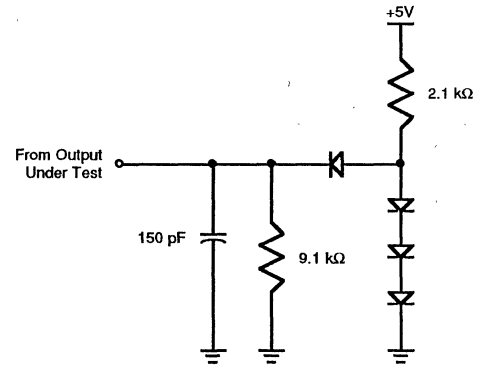


Figure 26. Test Load Configuration

ABSOLUTE MAXIMUM RATINGS

Symbol	Description	Min	Max	Units
V_{CC}	Supply Voltage (*)	-0.3	+7.0	V
T_{STG}	Storage Temp	-65	+150	C
T_A	Oper Ambient Temp		†	C
	Power Dissipation		2.2	W

Notes:
* Voltage on all pins with respect to GND.
† See Ordering Information.

Stress greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

CAPACITANCE

T_A = 25°C; V_{CC} = GND = 0V; f = 1.0 MHz; unmeasured pins to GND.

Parameter	Max
Input capacitance	12 pF
Output capacitance	12 pF
I/O capacitance	12 pF

V_{CC} SPECIFICATION

Low V_{CC} 4.3V ± 0.3V
High V_{CC} 5.0V ± 0.5V

DC ELECTRICAL CHARACTERISTICS

Symbol	Parameter	V_{CC} Note[3]	T_A = 0°C to +70°C Min	Max	Typical at 25°C	Units	Conditions	Notes
	Max Input Voltage	4.0V		V_{CC}+ 0.5V		V	I_{IN} 250 µA	
		5.0V		V_{CC} +0.5V		V	I_{IN} 250 µA	
V_{CH}	Clock Input High Voltage	4.0V	0.7 V_{CC}	V_{CC}+0.3V	1.3	V	Driven by External Clock Generator	
		5.0V	0.7 V_{CC}	V_{CC}+0.3V	2.5	V	Driven by External Clock Generator	
V_{CL}	Clock Input Low Voltage	4.0V	V_{SS}-0.3	0.2 V_{CC}	0.7	V	Driven by External Clock Generator	
		5.0V	V_{SS}-0.3	0.2 V_{CC}	1.5	V	Driven by External Clock Generator	
V_{IH}	Input High Voltage	4.0V	0.7 V_{CC}	V_{CC}+0.3	1.3	V		
		5.0V	0.7 V_{CC}	V_{CC}+0.3	2.5	V		
V_{IL}	Input Low Voltage	4.0V	V_{SS}-0.3	0.2 V_{CC}	0.7	V		
		5.0V	V_{SS}-0.3	0.2 V_{CC}	1.5	V		
V_{OH}	Output High Voltge	4.0V	V_{CC}-0.4		3.8	V	I_{OH} = -2.0 mA	
		5.0V	V_{CC}-0.4		4.8	V	I_{OH} = -2.0 mA	
V_{OL1}	Output Low Voltage	4.0V		0.4	0.2	V	I_{OH} = +4.0 mA	
		5.0V		0.4	0.1	V	I_{OL} = +4.0 mA	
V_{OL2}	Output Low Voltage	4.0V		1.5	0.3	V		
		5.0V		1.5	0.3	V	I_{OL} = +12 mA, 3 Pin Max	
V_{RH}	Reset Input High Voltage	4.0V	0.7 V_{CC}	V_{CC}+0.3	1.5	V		
		5.0V	0.7 V_{CC}	V_{CC}+0.3	2.1	V		
V_{RI}	Reset Input Low Voltage	4.0V	V_{SS}-0.3	0.2V_{CC}	1.1	V		
		5.0V	V_{SS}-0.3	0.2 V_{CC}	1.7	V		
V_{OFFSET}	Comparator Input Offset Voltage	4.0V		50	10	mV		
		5.0V		50	10	mV		
I_{IL}	Input Leakage	4.0V	-10	+10	< 1	µA	V_{IN} = 0V, V_{CC}	
		5.0V	-10	+10	< 1	µA	V_{IN} = 0V, V_{CC}	
I_{OL}	Output Leakage	4.0V	-10	+10	< 1	µA	V_{IN} = 0V, V_{CC}	
		5.0V	-10	+10	< 1	µA	V_{IN} = 0V, V_{CC}	
I_{IR}	Reset Input Current	4.0V		50	40	µA	V_{CC} =5.0V, RV_{RL} = 0	
		5.0V		60	45	µA		
I_{CC}	Supply Current (Standard Mode)	4.0V		12	8.5	mA	@ 8 MHz	[4,5]
		5.0V		16	15.0	mA	@ 8 MHz	[4,5]
		4.0V		15	11.5	mA	@ 12 MHz	[4,5]
		5.0V		20	18.0	mA	@ 12 MHz	[4,5]

DC ELECTRICAL CHARACTERISTICS (Continued)

Symbol	Parameter	V_{CC} Note[3]	T_A = 0°C to +70°C Min	Max	Typical at 25°C	Units	Conditions	Notes
I_{CC1}	Standby Current (Standard Mode)	4.0V		4.0	2.0	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 8 MHz	[4,5]
		5.0V		6.0	3.5	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 8 MHz	[4,5]
		4.0V		5.0	2.5	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 12 MHz	[4,5]
		5.0V		7.5	4.5	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 12 MHz	[4,5]
		4.0V		2.0	1.25	mA	Clock Divide by 16 @ 8 MHz	[4,5]
		5.0V		3.0	1.50	mA	Clock Divide by 16 @ 8 MHz	[4,5]
		4.0V		2.0	1.35	mA	Clock Divide by 16 @ 12 MHz	[4,5]
		5.0V		3.0	1.70	mA	Clock Divide by 16 @ 12 MHz	[4,5]
I_{CC}	Supply Current (Low EMI Mode)	4.0V		6.0	4.0	mA	@ 2 MHz	[4,5]
		5.0V		7.5	5.0	mA	@ 2 MHz	[4,5]
		4.0V		9.5	6.0	mA	@ 4 MHz	[4,5]
		5.0V		12.0	8.0	mA	@ 4 MHz	[4,5]
I_{CC1}	Standby Current (Low EMI Mode)	4.0V		1.6	0.8	mA	@ 2 MHz	[4,5]
		5.0V		2.0	1.0	mA	@ 2 MHz	[4,5]
		4.0V		2.4	1.2	mA	@ 4 MHz	[4,5]
		5.0V		3.0	1.5	mA	@ 4 MHz	[4,5]
		4.0V		1.0	0.50	mA	Clock Divide by 16 @ 2 MHz	[4,5]
		5.0V		2.0	0.75	mA	Clock Divide by 16 @ 2 MHz	[4,5]
		4.0V		1.0	0.75	mA	Clock Divide by 16 @ 4 MHz	[4,5]
		5.0V		2.0	1.0	mA	Clock Divide By 16 @ 4 MHz	[4,5]
I_{CC2}	Standby Current	4.0V		10	2	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is not Running	[6]
		5.0V		10	2	µA	STOP Mode V_{IN} = W_{DT} is not Running	[6]
		4.0V		400	250	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is Running	[6]
		5.0V		800	450	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is Running	[6]

Symbol	Parameter	V_{CC} Note[3]	T_A = 0°C to +70°C Min	Max	Typical at 25°C	Units	Conditions	Notes
I_{ALL}	Auto Latch Low Current	4.0V		-10	-5	µA	$0V < V_{IN} < V_{CC}$	
		5.0V		-10	-5	µA	$0V < V_{IN} < V_{CC}$	
I_{ALH}	Auto Latch High Current	4.0V		20	10	µA	$0V < V_{IN} < V_{CC}$	
		5.0V		20	10	µA	$0V < V_{IN} < V_{CC}$	
I_{POR}	Power On Reset	4.0V	4		7.5	ms		
		5.0V	2.5		4.5	ms		
V_{RST}	Auto Reset Voltatge			3.0	2.5	V	2 MHz max Ext. CLK Freq.	[3]

Notes:

[1] I_{CC1}

	Typ	Max	Unit	Freq
Clock Driven on Crystal	3.0 mA	5.0	mA	8 MHz
or XTAL Resonator	0.3 mA	5.0	mA	8 MHz

[2] V_{SS}=0V=GND.

[3] 5.0V ± 0.5V, 4.0V

[4] All outputs unloaded, I/O pins floating, inputs at rail.

[5] CL1=CL2=100 pF.

[6] Same as note [4] except inputs at V_{CC}.

AC ELECTRICAL CHARACTERISTICS
Additional Timing Diagram (Standard Mode)

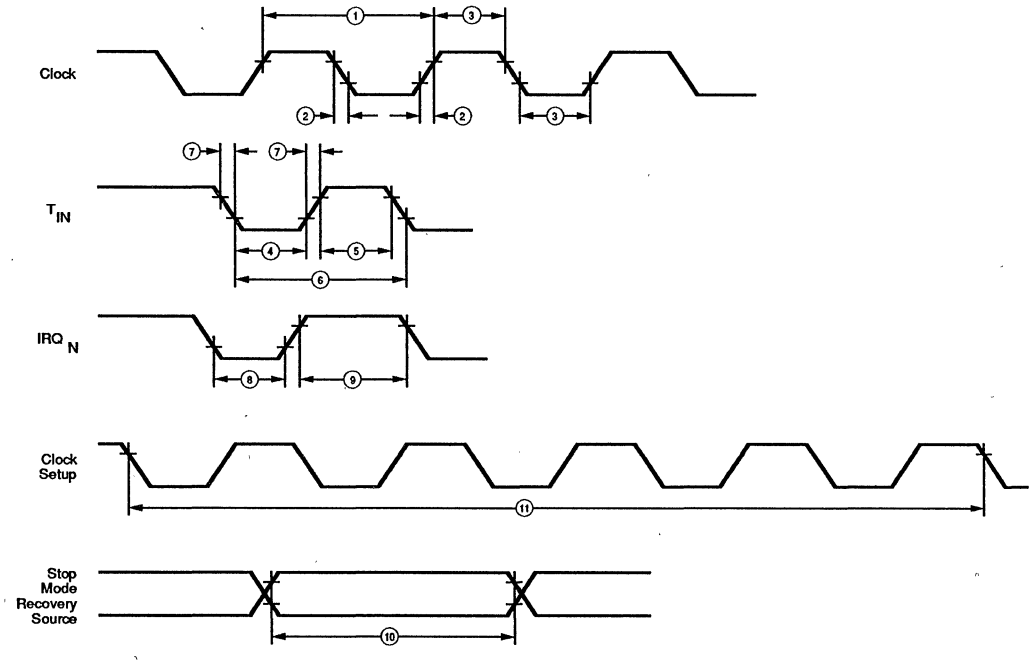


Figure 27. Additional Timing

AC ELECTRICAL CHARACTERISTICS
Additional Timing Table (Standard Mode)

No	Symbol	Parameter	V_{CC} Note [6]	T_A = 0°C to +70°C 8 MHz Min	8 MHz Max	12 MHz Min	12 MHz Max	Units	Notes
1	TpC	Input Clock Period	4.0V	125	100000	83	100000	ns	[1]
			5.0V	125	100000	83	100000	ns	[1]
2	TrC,TfC	Clock Input Rise & Fall Times	4.0V		25		15	ns	[1]
			5.0V		25		15	ns	[1]
3	TwC	Input Clock Width	4.0V	37		26		ns	[1]
			5.0V	37		26		ns	[1]
4	TwTinL	Timer Input Low Width	4.0V	100		100		ns	[1]
			5.0V	70		70		ns	[1]
5	TwTinH	Timer Input High Width	4.0V	3TpC		3TpC			[1]
			5.0V	3TpC		3TpC			[1]

AC ELECTRICAL CHARACTERISTICS

Additional Timing Table (Standard Mode – Continued)

No	Symbol	Parameter	V_{CC} Note [6]	T_A = 0°C to +70°C 8 MHz Min	8 MHz Max	12 MHz Min	12 MHz Max	Units	Notes
6	TpTin	Timer Input Period	4.0V	8TpC		8TpC			[1]
			5.0V	8TpC		8TpC			[1]
7	TrTin, TfTin	Timer Input Rise & Fall Timers	4.0V		100		100	ns	[1]
			5.0V		100		100	ns	[1]
8A	TwIL	Int. Request Low Time	4.0V	100		100		ns	[1,2]
			5.0V	70		70		ns	[1,2]
8B	TwIL	Int. Request Low Time	4.0V	3TpC		3TpC			[1,3]
			5.0V	3TpC		3TpC			[1,3]
9	TwIH	Int. Request Input High Time	4.0V	3TpC		3TpC			[1,2]
			5.0V	3TpC		3TpC			[1,2]
10	Twsm	STOP Mode Recovery Width Spec	4.0V	12		12		ns	
			5.0V	12		12		ns	
			4.0V	5TpC					Reg. SMR - D5=0
			5.0V	5TpC					No Delay
			4.0V	5TpC					Reg. SM - D5=1
			5.0V	5TpC					with Delay
11	Tost	Oscillator Startup Time	4.0V		5TpC		5TpC		[4]
			5.0V		5TpC		5TpC		[4]
12	Twdt	Watchdog Timer Delay Time	4.0V	10		10		ms	D0 = 0 [5] [7]
			5.0V	5		5		ms	D1 = 0 [5] [7]
			4.0V	20		20		ms	D0 = 1 [5] [8]
			5.0V	15		15		ms	D1 = 0 [5] [8]
			4.0V	35		35		ms	D0 = 0 [5] [9]
			5.0V	25		25		ms	D1 = 1 [5] [9]
			4.0V	175		175		ms	D0 = 1 [5] [10]
			5.0V	100		100		ms	D1 = 1 [5] [10]

Notes:

[1] Timing Reference uses 0.9 V_{CC} for a logic 1 and 0.1 V_{CC} for a logic 0.
[2] Interrupt request via Port 3 (P31-P33).
[3] Interrupt request via Port 3 (P30).
[4] SMR-D5 = 0.
[5] Reg. WDTMR.
[6] 5.0V ± 0.5V, 4.0V
[7] Reg.WDTMR D1=0, D0=0
[8] Reg. WDTMR D1=0, D0=1
[9] Reg. WDTMR D1=1, D0=0
[10] Reg.WDTMR D1=1, D0=1

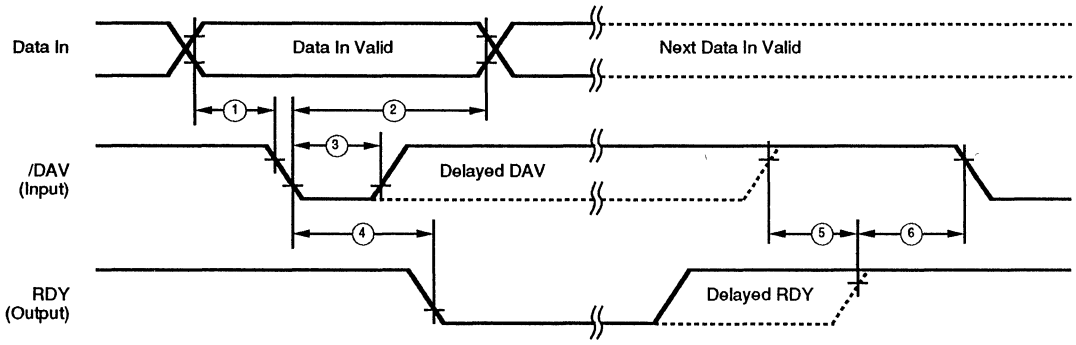


Figure 28. Input Handshake Timing

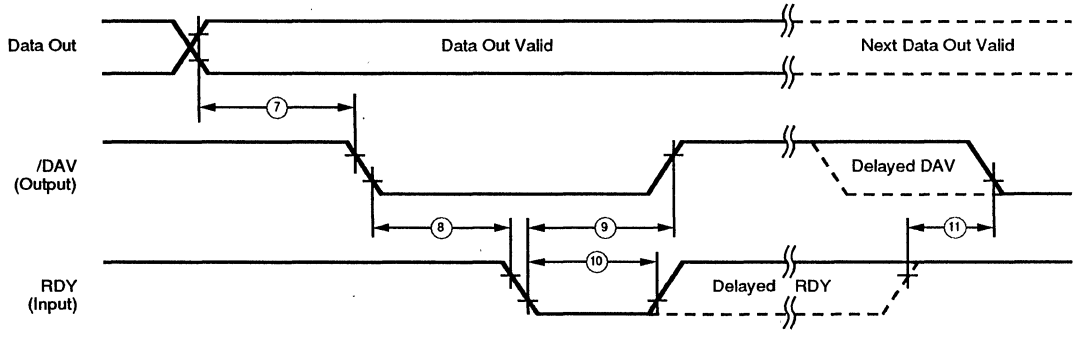


Figure 29. Output Handshake Timing

AC ELECTRICAL CHARACTERISTICS

Handshake Timing Table - Standard Mode

No	Symbol	Parameter	V_{cc} Note [1]	Standard Mode 8 MHz Min	8 MHz Max	12 MHz Min	12 MHz Max	Data Direction
1	TsDI(DAV)	Data In Setup Time	4.0V	0	0			IN
			5.0V	0	0			IN
2	ThDI(DAV)	Data In Hold Time	4.0V	160	160			IN
			5.0V	115	115			IN
3	TwDAV	Data Available Width	4.0V	155	155			IN
			5.0V	110	110			IN
4	TdDAVI (RDY)	DAV Fall to RDY Fall Delay	4.0V		160		160	IN
			5.0V		115		115	IN
5	TdDAVId (RDY)	DAV Rise to RDY Rise Delay	4.0V		120		120	IN
			5.0V		80		80	IN
6	TdDO(DAV)	RDY Rise to DAV Fall Delay	4.0V	0		0		IN
			5.0V	0		0		IN
7	TcLDAV0 (RDY)	Data Out to DAV Fall Delay	4.0V	63		42		OUT
			5.0V	63		42		OUT
8	TcLDAV0 (RDY)	DAV Fall to RDY Fall Delay	4.0V	0		0		OUT
			5.0V	0		0		OUT
9	TdRDY0 (DAV)	RDY Fall to DAV Rise Delay	4.0V		160		160	OUT
			5.0V		115		115	OUT
10	TwRDY	RDY Width	4.0V	110		110		OUT
			5.0V	80		80		OUT
11	TdRDY0d (DAV)	RDY Rise to DAV Fall Delay	4.0V	110			110	OUT
			5.0V	80			80	OUT

Note:
[1] 5.0V ± 0.5V, 4.0V.
Standard operating temperature range 0°C to +70°C.

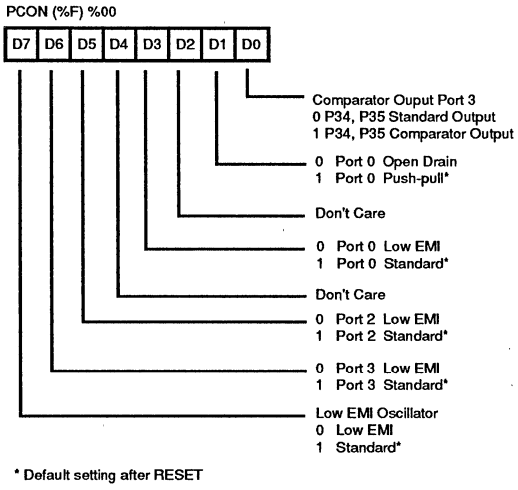


Figure 30. Port Configuration Register

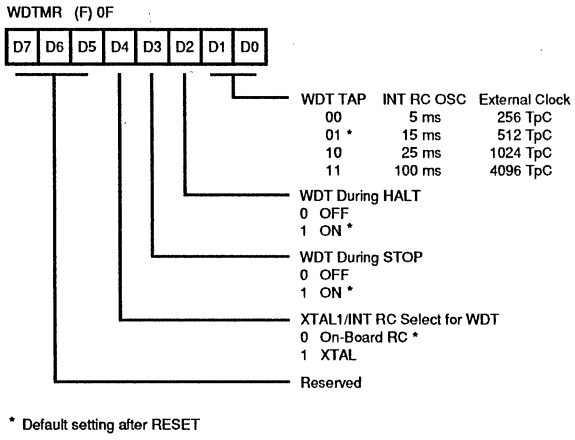


Figure 32. Watchdog Timer Mode Register Z8 Control Register Diagrams

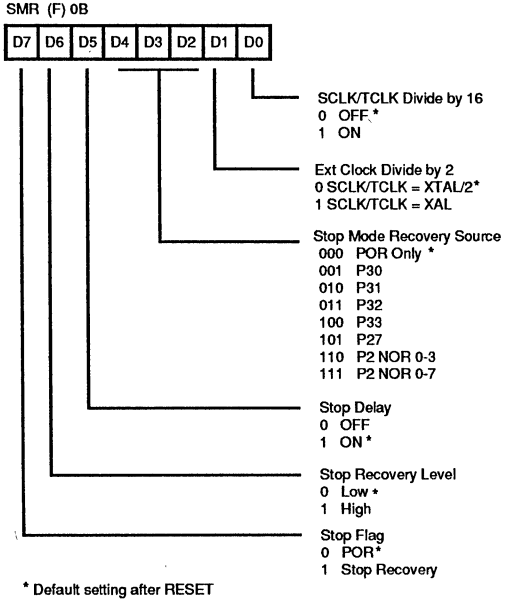


Figure 31. Stop Mode Recovery Register

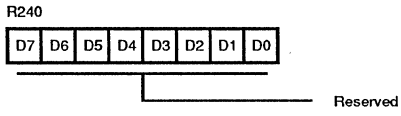


Figure 33. Reserved

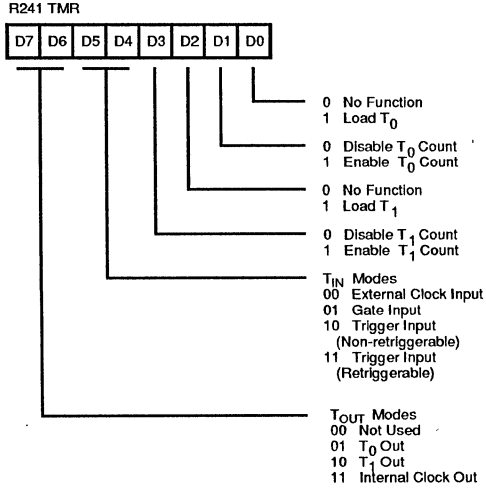


Figure 34. Timer Mode Register (F1H: Read/Write)

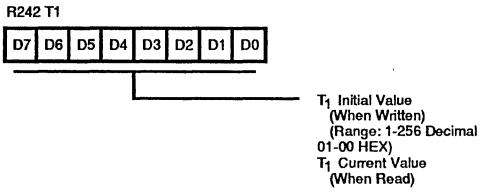


Figure 35. Counter Timer 1 Register (F2H: Read/Write)

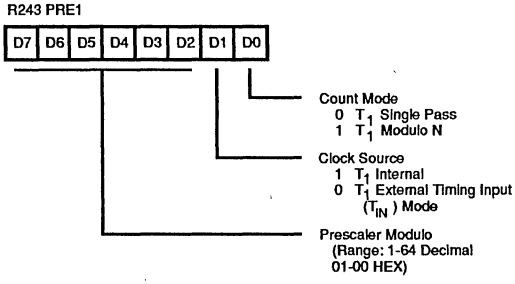


Figure 36. Prescaler 1 Register (F3H: Write Only)

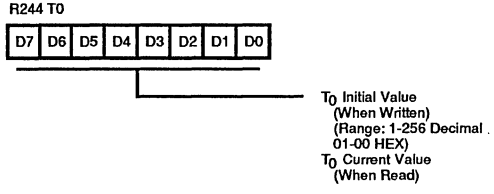


Figure 37. Counter/Timer 0 Register (F4H: Read/Write)

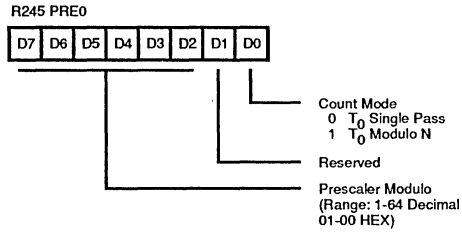


Figure 38. Prescaler 0 Register (F5H: Write Only)

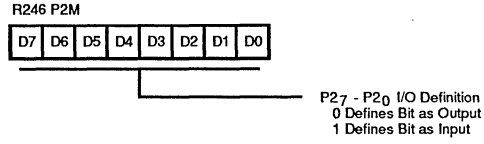


Figure 39. Port 2 Mode Register (F6H: Write Only)

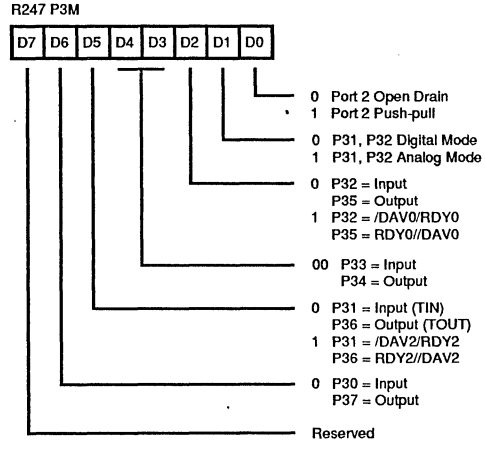


Figure 40. Port 3 Mode Register (F7H: Write Only)

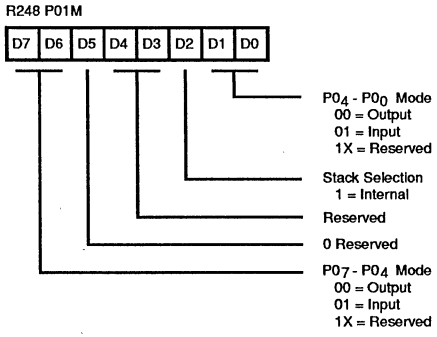


Figure 41. Port 0 and 1 Mode Register (F8H: Write Only)

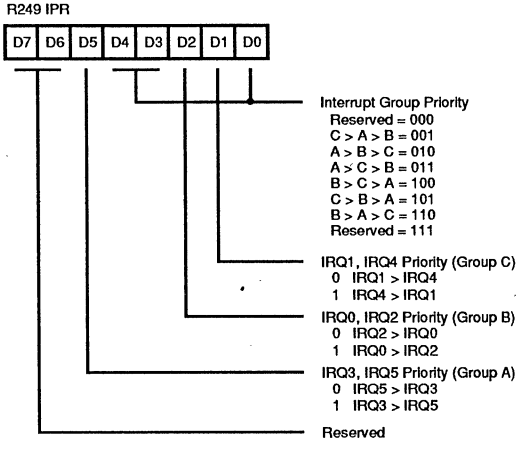


Figure 42. Interrupt Priority Register (F9H: Write Only)

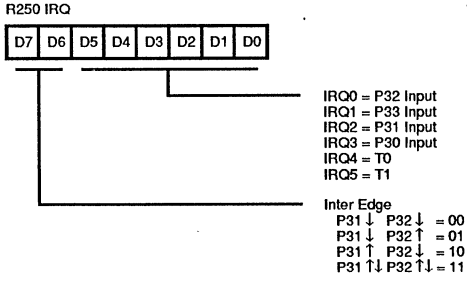


Figure 43. Interrupt Request Register (FAH: Read/Write)

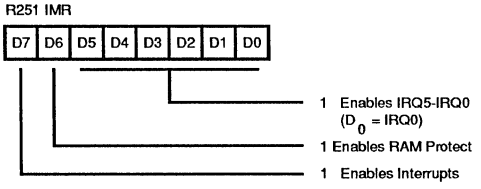


Figure 44. Interrupt Mask Register (FBH: Read/Write)

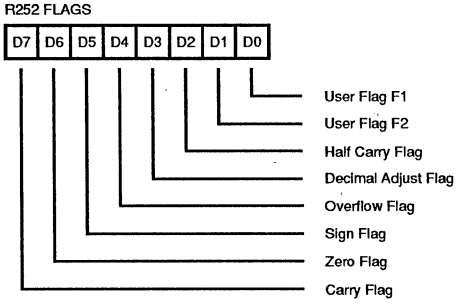


Figure 45. Flag Register (FCH: Read/Write)

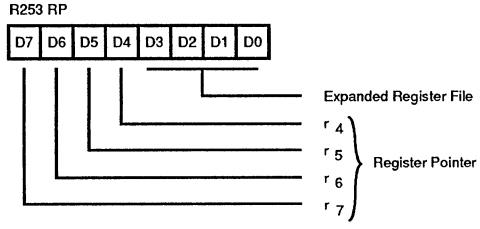


Figure 46. Register Pointer (FDH: Read/Write)

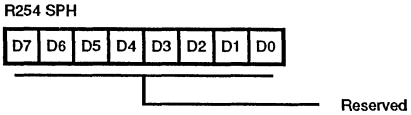


Figure 47. Reserved

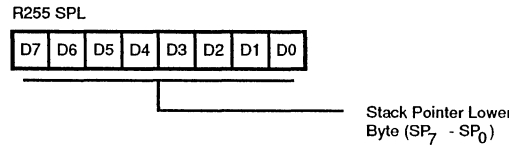


Figure 48. Stack Pointer (FFH: Read/Write)

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

Symbol	Meaning
IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

Symbol	Meaning
dst	Destination location or contents
src	Source location or contents
cc	Condition code
@	Indirect address prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag register (Control Register 252)
RP	Register Pointer (R253)
IMR	Interrupt mask register (R251)

Flags. Control register (R252) contains the following six flags:

Symbol	Meaning
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

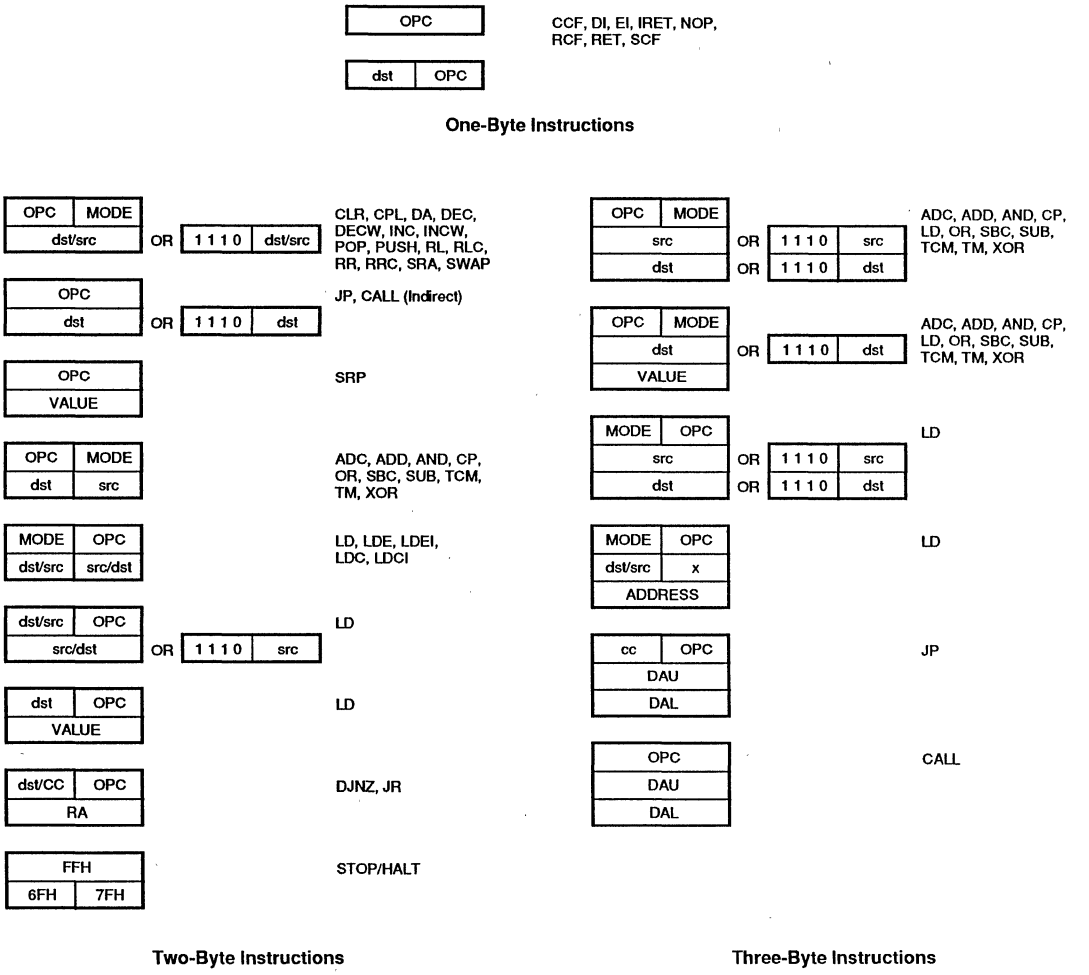
Affected flages are indicated by:

0	Clear to zero
1	Set to one
*	Set to clear according to operation
-	Unaffected
x	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000		Always True	
0111	C	Carry	C = 1
1111	NC	No Carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not Zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No Overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not Equal	Z = 0
1001	GE	Greater Than or Equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater Than	[Z OR (S XOR V)] = 0
0010	LE	Less Than or Equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned Greater Than or Equal	C = 0
0111	ULT	Unsigned Less Than	C = 1
1011	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned Less Than or Equal	(C OR Z) = 1
0000		Never True	

INSTRUCTION FORMATS



INSTRUCTION SUMMARY

Note: Assignment of a value is indicated by the symbol " ← ". For example:

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr (n)" is used to refer to bit (n) of a given operand location. For example:

dst (7)

refers to bit 7 of the destination operand.

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst, src dst←dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst←dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst←dst AND src	†		5[ ]	-	*	*	0	-	-
CALL dst SP←SP - 2 @SP←PC, PC←dst	DA IRR		D6 D4	-	-	-	-	-	-
CCF C←NOT C			EF	*	-	-	-	-	-
CLR dst dst←0	R IR		B0 B1	-	-	-	-	-	-
COM dst dst←NOT dst	R IR		60 61	-	*	*	0	-	-
CP dst, src dst - src	†		A[ ]	*	*	*	*	-	-
DA dst dst←DA dst	R IR		40 41	*	*	*	X	-	-
DEC dst dst←dst - 1	R IR		00 01	-	*	*	*	-	-
DECW dst dst←dst - 1	RR IR		80 81	-	*	*	*	-	-
DI IMR(7)←0			8F	-	-	-	-	-	-
DJNZr, dst r←r - 1 if r ≠ 0 PC←PC + dst Range: +127, -128	RA		rA r = 0 - F	-	-	-	-	-	-
EI IMR(7)←1			9F	-	-	-	-	-	-
HALT			7F	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
INC dst dst←dst + 1	r R IR		rE r = 0 - F 20 21	-	*	*	*	-	-
INCW dst dst←dst + 1	RR IR		A0 A1	-	*	*	*	-	-
IRET FLAGS←@SP; SP←SP + 1 PC←@SP; SP←SP + 2; IMR(7)←1			BF	*	*	*	*	*	*
JP cc, dst if cc is true PC←dst	DA IRR		cD c = 0 - F 30	-	-	-	-	-	-
JR cc, dst if cc is true, PC←PC + dst Range: +127, -128	RA		cB c = 0 - F	-	-	-	-	-	-
LD dst, src dst←src	r r R r X r r Ir R R R IR IR	Im R r X r Ir r R IR IM IM R	rC r8 r9 r = 0 - F C7 D7 E3 F3 E4 E5 E6 E7 F5	-	-	-	-	-	-
LDC dst, src	r	Irr	C2	-	-	-	-	-	-
LDCI dst, src dst←src r←r +1; rr←rr + 1	Ir	Irr	C3	-	-	-	-	-	-

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
NOP			FF	-	-	-	-	-	-
OR dst, src dst←dst OR src	†		4[ ]	-	*	*	0	-	-
POP dst dst←@SP; SP←SP + 1	R IR		50 51	-	-	-	-	-	-
PUSH src SP←SP - 1; @SP←src		R IR	70 71	-	-	-	-	-	-
RCF C←0			CF	0	-	-	-	-	-
RET PC←@SP; SP←SP + 2			AF	-	-	-	-	-	-
RL dst	R IR		90 91	*	*	*	*	-	-
RLC dst	R IR		10 11	*	*	*	*	-	-
RR dst	R IR		E0 E1	*	*	*	*	-	-
RRC dst	R IR		C0 C1	*	*	*	*	-	-
SBC dst, src dst←dst←src←C	†		3[ ]	*	*	*	*	1	*
SCF C←1			DF	1	-	-	-	-	-
SRA dst	R IR		D0 D1	*	*	*	0	-	-
SRP src RP←src		Im	31	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
STOP			6F	-	-	-	-	-	-
SUB dst, src dst←dst←src	†		2[ ]	*	*	*	*	1	*
SWAP dst	R IR		F0 F1	X	*	*	X	-	-
TCM dst, src (NOT dst) AND src	†		6[ ]	-	*	*	0	-	-
TM dst, src dst AND src	†		7[ ]	-	*	*	0	-	-
WDT			5F	-	-	-	-	-	-
XOR dst, src dst←dst XOR src	†		B[ ]	-	*	*	0	-	-

† These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

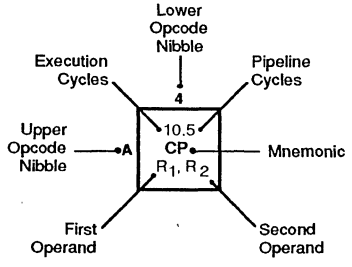
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address Mode dst	Address Mode src	Lower Opcode Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

OPCODE MAP

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								6.0 **STOP**
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								7.0 **HALT**
8	10.5 **DECW** RR1	10.5 **DECW** IR1	12.0 **LDE** r1, Irr2	18.0 **LDEI** Ir1, Irr2												6.1 **DI**
9	6.5 **RL** R1	6.5 **RL** IR1	12.0 **LDE** r2, Irr1	18.0 **LDEI** Ir2, Irr1												6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14.0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2				10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1	12.0 **LDC** r2, Irr1	18.0 **LDCI** Ir2, Irr1	20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6.5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2				3				2					3	1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_2 = Dst address
R_1 or r_2 = Src address

Sequence:
Opcode, First Operand,
Second Operand

Note: The blank are not defined.

* 2-byte instruction appears as a 3-byte instruction

PRODUCT SPECIFICATION

Z86C40

CMOS Z8® CCP™
CONSUMER CONTROLLER PROCESSOR

FEATURES

- 8-bit CMOS microcntroller, 40- or 44-pin package
- Low cost
- 3.0 to 5.5 volt operating range
- Low power consumption - 50 mW (Typical)
- Fast instruction pointer - 1.0 microsecond @ 12 MHz
- Two standby modes - STOP and HALT
- 32 input/output lines (two with comparator inputs)
- All digital inputs are CMOS levels, Schmitt triggered
- 4 Kbytes of ROM
- 256 bytes of RAM (236 for general purpose)
- Two Expanded Register File control registers
- Two programmable 8-bit Counter/Timers
- 6-bit programmable prescaler
- Six vectored, priority interrupts from six different sources.
- Clock speeds 8 and 12 MHz
- Brown-Out protection
- Watch Dog/Power-On Reset Timer
- Two Comparators with programmable interrupt polarity
- On-chip oscillator that accepts a crystal, ceramic resonator, LC, RC or external clock drive.
- RAM and ROM Protect

GENERAL DESCRIPTION

The Z86C40 CCP (Consumer Controller Processor) introduces a new level of sophistication to single-chip architecture. The Z86C40 is a member of the Z8 single-chip microcontroller family with 4 Kbytes of ROM (Z86C40) and 236 bytes of general purpose RAM. The CCP is housed in a 40-pin DIP, 44-pin Leaded Chip Carrier, and a 44-pin Quad Flat Pack, and is CMOS compatible. Having the ROM/ROMless selectively, the CCP offers both external memory and pre-programmed ROM which enables this Z8 microcomputer to be used in high volume applications or where code flexibility is required. Zilog's CMOS microcomputer offers fast execution, efficient use of memory, sophisticated interrupts, input/output bit manipulation capabilities, and easy hardware/software system expansion along with low cost and low power consumption.

The Z86C40 architecture is characterized by Zilog's 8-bit microcontroller core with an Expanded Register File to allow access to register mapped peripheral and I/O circuits. The CCP offers a flexible I/O scheme, an efficient register and address space structure, and a number of ancillary features that are useful in many industrial, automotive, computer peripherals, and advanced scientific applications.

The CCP applications demand powerful I/O capabilities. The Z86C40 fulfills this with 32 pins dedicated to input and output. These lines are grouped into four ports. Each port consists of eight lines, and is configurable under software control to provide timing, status signals, parallel I/O with or without handshake, and address/data bus for interfacing external memory.

GENERAL DESCRIPTION (Continued)

There are four basic address spaces available to support this wide range of configurations: Program Memory, Register File, Data Memory, and Expanded Register File. The Register File is composed of 236 bytes of general purpose registers, four I/O port registers, and 15 control and status registers. The Expanded Register File consists of two control registers.

To unburden the program from coping with the real-time problems, such as counting/timing and data communication, the Z86C40 offers two on-chip counter/timers with a large number of user selectable modes. Also, two on-board comparators which process analog signals with a common reference voltage (Figure 1).

Note: All Signals with a preceding front slash, "/", are active Low, e.g.: B/W (WORD is active Low); /B/W (BYTE is active Low, only).

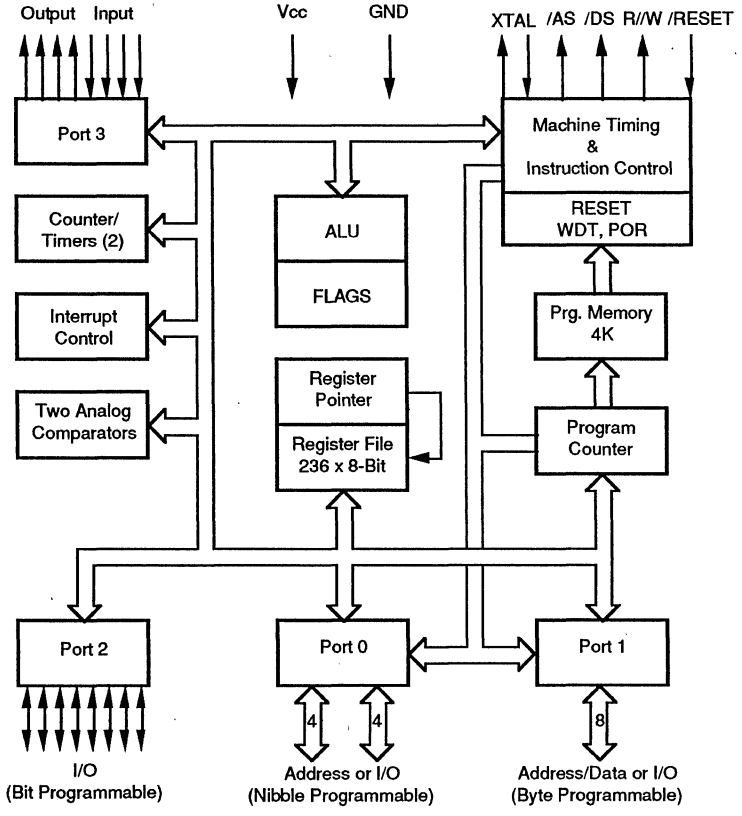


Figure 1. Functional Block Diagram

PIN DESCRIPTION

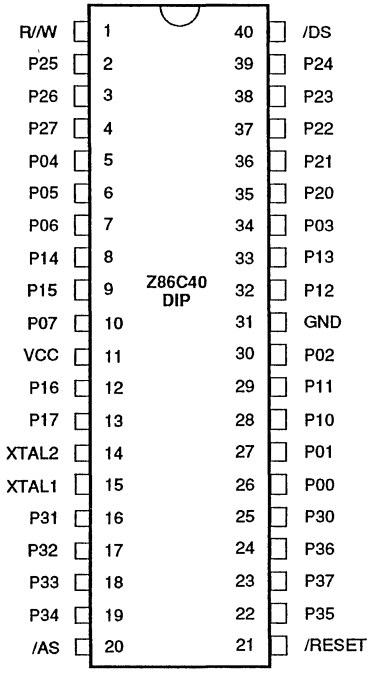


Figure 2. 40-Pin Dual In Line, Pin Assignments

Table 1. 40-Pin Dual-In-Line Package, Pin Identification

Pin #	Symbol	Function	Direction
1	R//W	Read/Write	Output
2-4	P25-7	Port 2 pins 5,6,7	In/Output
5-7	P04-6	Port 0 pins 4,5,6	In/Output
8-9	P14-5	Port 1 pins 4,5	In/Output
10	P07	Port 0 pin 7	In/Output
11	V_{cc}	Power Supply	Input
12-13	P16-7	Port 1 pins 6,7	In/Output
14	XTAL2	Crystal, Oscillator Clock	Output
15	XTAL1	Crystal, Oscillator Clock	Input
16-18	P31-3	Port 3 pins 1,2,3	Input
19	P34	Port 3 pin 4	Output
20	/AS	Address Strobe	Output
21	/RESET	Reset	Input
22	P35	Port 3 pin 5	Output
23	P37	Port 3 pin 7	Output
24	P36	Port 3 pin 6	Output
25	P30	Port 3 pin 0	Input
26-27	P00-1	Port 0 pin 0,1	In/Output
28-29	P10-1	Port 1 pin 0,1	In/Output
30	P02	Port 0 pin 2	In/Output
31	GND	Ground, GND	Input
32-33	P12-3	Port 1 pin 2,3	In/Output
34	P03	Port 0 pin 3	In/Output
35-39	P20-4	Port 2 pin 0,1,2,3,4	In/Output
40	/DS	Data Strobe	Output

PIN DESCRIPTION (Continued)

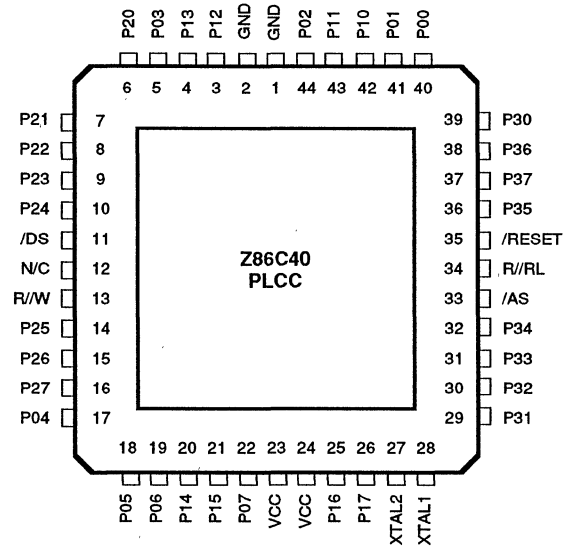


Figure 3. 44-Pin Leaded Chip Carrier, Pin Assignments

Table 2. 44-pin Leaded Chip Carrier, Pin Identification

Pin #	Symbol	Function	Direction
1-2	GND	Ground, GND	Input
3-4	P12-3	Port 1 pins 2,3	In/Output
5	P03	Port 0 pin 3	In/Output
6-10	P20-4	Port 2 pins 0,1,2,3,4	In/Output
11	/DS	Data Strobe	Output
12	NC	Not Connected	
13	R//W	Read/Write	Output
14-16	P25-7	Port 2 pins 5,6,7	In/Output
17-19	P04-6	Port 0 pins 4,5,6	In/Output
20-21	P14-5	Port 1 pins 4,5	In/Output
22	P07	Port 0 pin 7	In/Output
23-24	V_{cc}	Power Supply	Input
25-26	P16-7	Port 1 pins 6,7	In/Output
27	XTAL2	Crystal, Oscillator Clock	Output

Pin #	Symbol	Function	Direction
28	XTAL1	Crystal, Oscillator Clock	Input
29-31	P31-3	Port 3 pins 1,2,3	Input
32	P34	Port 3 pin 4	Output
33	/AS	Address Strobe	Output
34	R//RL	ROM/ROMless Control	Input
35	/RESET	Reset	Input
36	P35	Port 3 pin 5	Output
37	P37	Port 3 pin 7	Output
38	P36	Port 3 pin 6	Output
39	P30	Port 3 pin 0	Input
40-41	P00-1	Port 0 pins 0,1	In/Output
42-43	P10-1	Port 1 pins 0,1	In/Output
44	P02	Port 0 pin 2	In/Output

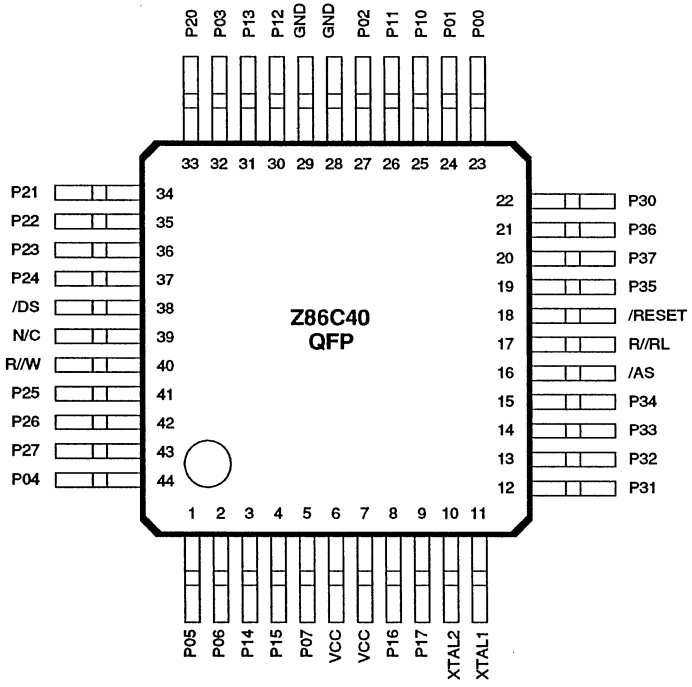


Figure 4. 44-Pin Quad Flat Pack, Pin Assignments

Table 3. 44-Pin Quad Flat Pack, Pin Identification

Pin #	Symbol	Function	Direction
1-2	P05-6	Port 0 pins 5,6	In/Output
3-4	P14-5	Port 1 pins 4,5	In/Output
5	P07	Port 0 pin 7	In/Output
6-7	V_{CC}	Power Supply	Input
8-9	P16-7	Port 1 pins 6,7	In/Output
10	XTAL2	Crystal, Oscillator Clock	Output
11	XTAL1	Crystal, Oscillator Clock	Input
12-14	P31-3	Port 3 pins 1,2,3	Input
15	P34	Port 3 pin 4	Output
16	/AS	Address Strobe	Output
17	R//RL	ROM/ROMless Control	Input
18	/RESET	Reset	Input
19	P35	Port 3 pin 5	Output
20	P37	Port 3 pin 7	Output

Pin #	Symbol	Function	Direction
21	P36	Port 3 pin 6	Output
22	P30	Port 3 pin 0	Input
23-24	P00-1	Port 0 pins 0,1	In/Output
25-26	P10-1	Port 1 pins 0,1	In/Output
27	P02	Port 0 pin 2	In/Output
28-29	GND	Ground, GND	Input
30-31	P12-3	Port 1 pins 2,3	In/Output
32	P03	Port 0 pin 3	In/Output
33-37	P20-4	Port 2 pins 0,1,2,3,4	In/Output
38	/DS	Data Strobe	Output
39	NC	Not Connected	
40	R//W	Read/Write	Output
41-43	P25-7	Port 2 pins 5,6,7	In/Output
44	P04	Port 0 pin 4	In/Output

PIN FUNCTIONS

/ROMless. (input, active Low). This pin, when connected to GND, disables the internal ROM and forces the device to function as a Z86C90/C89 ROMless Z8. (Note that, when left unconnected or pulled high to V_{cc} the part functions normally as a Z8 ROM version).

/DS. (output, active Low). Data Strobe is activated once for each external memory transfer. For a READ operation, data must be available prior to the trailing edge of /DS. For WRITE operations, the falling edge of /DS indicates that output data is valid.

/AS. (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Address output is via Port 0/Port 1 for all external programs. Memory address transfers are valid at the trailing edge of /AS. Under program control, /AS is placed in the high-impedance state along with Ports 0 and 1, Data Strobe, and Read/Write.

XTAL1. *Crystal 1* (time-based input). This pin connects a parallel-resonant crystal, ceramic resonator, LC, or RC network, or an external single-phase clock to the on-chip oscillator input.

XTAL2. *Crystal 2* (time-based output). This pin connects a parallel-resonant, crystal, ceramic resonant, LC, or RC network to the on-chip oscillator output.

R/W. (output, write Low). Read/Write, the R/W signal is low when the CCP is writing to the external program or data memory.

Port 0. (P00-P07). Port 0 is an 8-bit, bidirectional, CMOS compatible port. These eight I/O lines are configured under software control as a nibble I/O port, or as an address port for interfacing external memory. The input buffers are Schmitt triggered and output drivers are push-pull. Port 0 is placed under handshake control. In this configuration, Port 3, lines P32 and P35 are used as the handshake control /DAV0 and RDY0. Handshake signal direction is dictated by the I/O direction to Port 0 of the upper nibble P04-P07. The lower nibble must have the same direction as the upper nibble.

For external memory references, Port 0 provides address bits A11-A8 (lower nibble) or A15-A8 (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. If one or both nibbles are needed for I/O operation, they are configured by writing to the Port 0 mode register. In ROMless mode, after a hardware reset, Port 0 is configured as address lines A15-A8, and extended timing is set to accommodate slow memory access. The initialization routine can include reconfiguration to eliminate this extended timing mode. (In ROM mode, Port 0 is defined as input after reset.)

Port 0 is set in the high-impedance mode if selected as an address output state along with Port 1 and the control signals /AS, /DS and R/W (Figure 5).

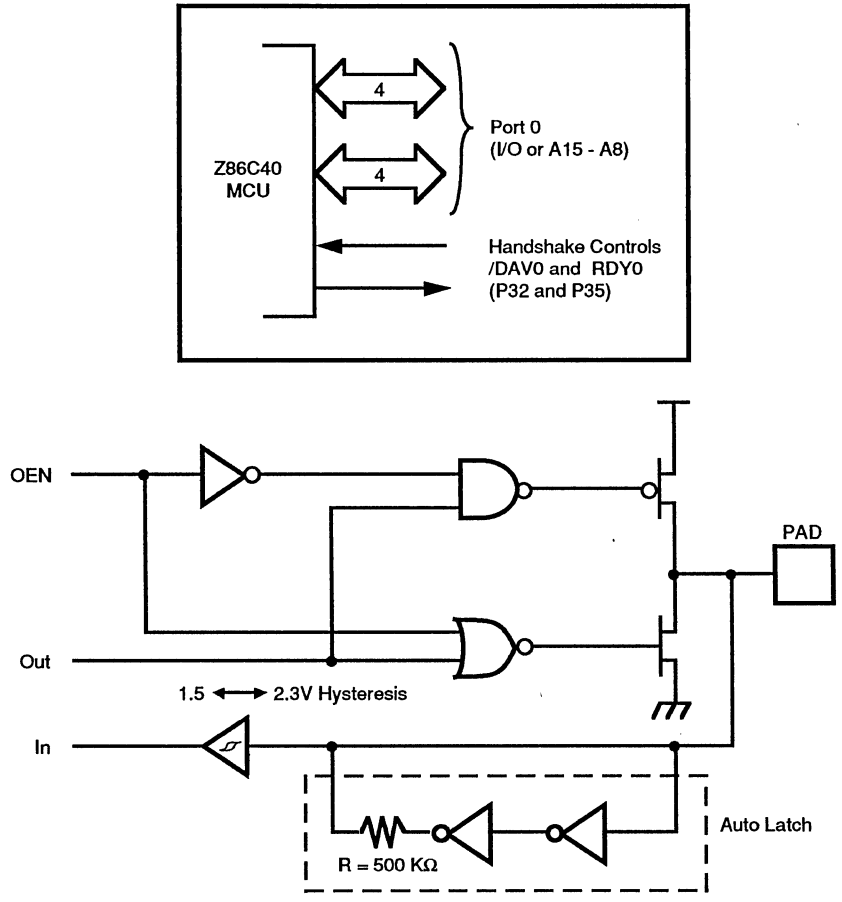


Figure 5. Port 0 Configuration

PIN FUNCTIONS (Continued)

Port 1. (P10-P17). Port 1 is an 8-bit, bidirectional, CMOS compatible port (Figure 6). It has multiplexed Address (A7-A0) and Data (D7-D0) ports. For the Z86C40 ROM device, these eight I/O lines are programmed as inputs or outputs, or can be configured under software control as an Address/Data port for interfacing external memory. The input buffers are Schmitt triggered and the output drivers are push-pull.

Port 1 may be placed under handshake control. In this configuration, Port 3, lines P33 and P34 are used as the handshake controls RDY1 and /DAV1 (Ready and Data Available). Memory locations greater than 4096 are referenced through Port 1. To interface external memory, Port 1 must be programmed for the multiplexed Address/Data mode. If more than 256 external locations are required, Port 0 outputs the additional lines.

Port 1 can be placed in the high-impedance state along with Port 0, /AS, /DS and R/W, allowing the Z86C40 to share common resources in multiprocessor and DMA applications.

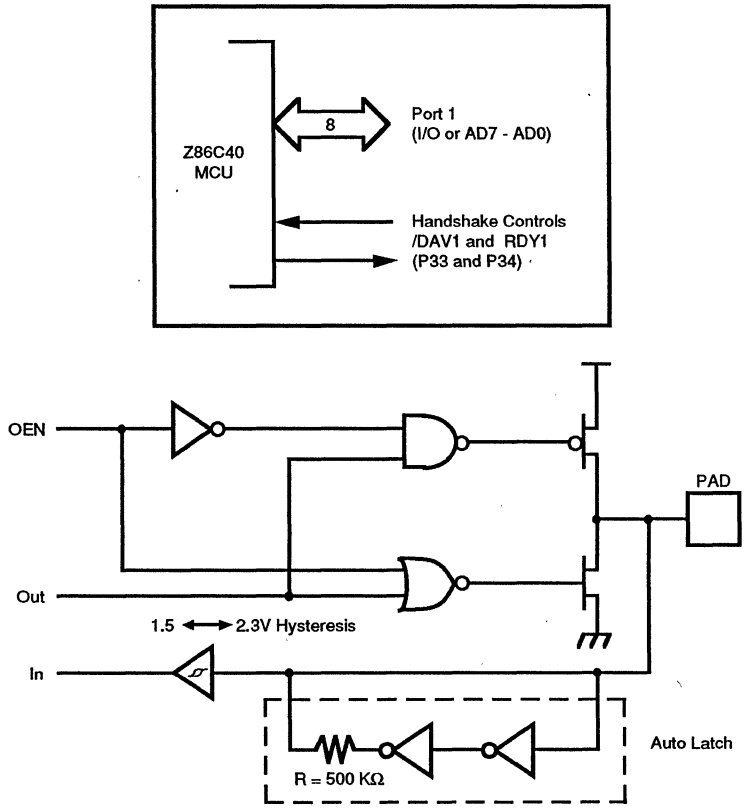


Figure 6. Port 1 Configuration

Port 2. (P20-P27). Port 2 is an 8-bit, bidirectional, CMOS compatible I/O port. These eight I/O lines are configured under software control as an input or output, independently. Port 2 is always available for I/O operation. The input buffers are Schmitt triggered. Bits programmed as outputs may be globally programmed as either push-pull or open-drain.

Port 2 may be placed under handshake control. In this configuration, Port 3 lines P31 and P36 are used as the handshake controls lines /DAV2 and RDY2. The handshake signal assignment for Port 3 lines P31 and P36 is dictated by the direction (input or output) assigned to bit 7, Port 2 (Figure 7).

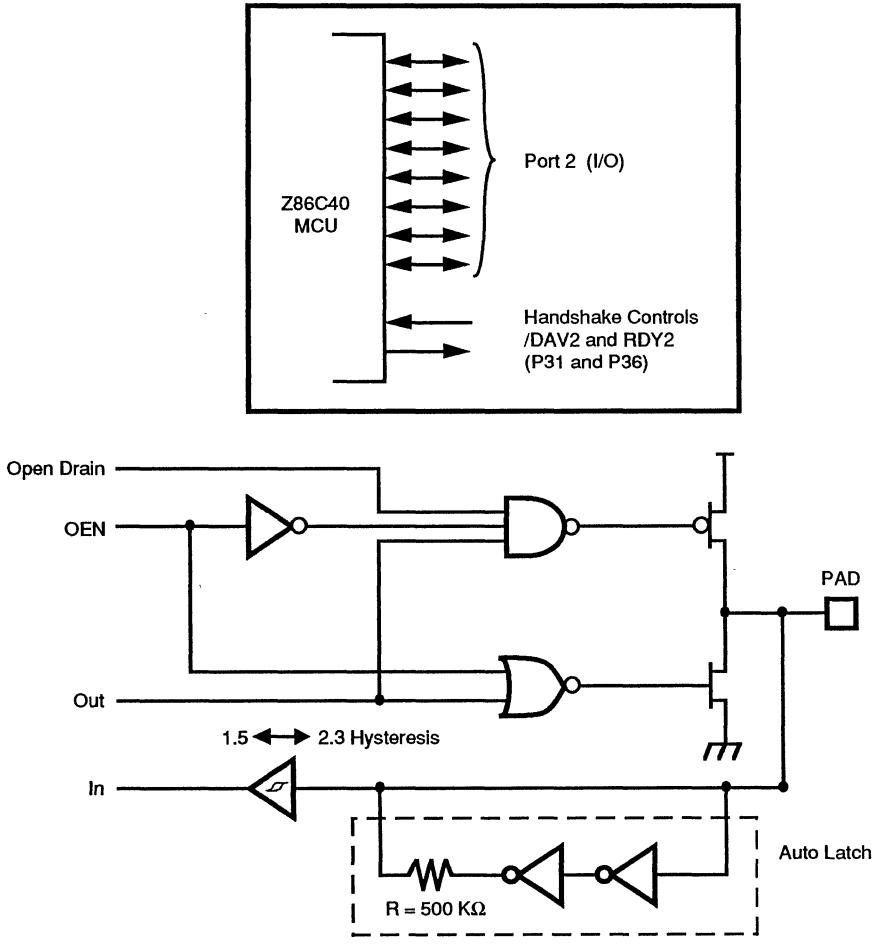


Figure 7. Port 2 Configuration

PIN FUNCTIONS (Continued)

Port 3. (P30-P37). Port 3 is an 8-bit, CMOS compatible four fixed input and four fixed output. Port 3 consists of four fixed inputs (P30-P33) and four fixed outputs (P34-P37). It is configured under software control for Input/Output, Counter/Timers, interrupt, port handshake and Data Memory functions. Port 3, pin 0 input is Schmitt triggered, and pins P31, P32, and P33 are standard CMOS inputs; outputs are push-pull.

Two on-board comparators can process analog signals on P31 and P32 with reference to the voltage on P33. The analog function is enabled by programming Port 3 Mode Register (bit 1). Port 3 pin 0 and pin 3 are falling edge interrupt inputs. P31 and P32 are programmable as rising, falling or both edge triggered interrupts (IRQ register bits 6 and bit 7). P33 is the comparator reference voltage input.

Access to Counter/Timers 1 is made through P31 (T_{IN}) and P36 (T_{OUT}). Handshake lines for ports 0, 1 and 2 are available on P31 through P36.

Port 3 also provides the following control functions: handshake for Ports 0, 1 and 2 (/DAV and RDY); four external interrupt request signals (IRQ0-IRQ3); timer input and output signals (T_{IN} and T_{OUT}); Data Memory Select (/DM - Figure 8).

Auto-Latch. The Auto-Latch puts valid CMOS levels on all CMOS inputs (except P31-P33) that are not externally driven. Whether this level is zero or one, cannot be determined. A valid CMOS level, rather than a floating node, reduces excessive supply current flow in the input buffer.

Table 4. Pin Assignments

Pin	I/O	CTC1	AN IN	Int.	P0 HS	P1 HS	P2 HS	Ext
P30	IN			IRQ3				
P31	IN	Tin	AN1	IRQ2			D/R	
P32	IN		AN2	IRQ0	D/R			
P33	IN		REF	IRQ1		D/R		
P34	OUT					R/D		DM
P35	OUT				R/D			
P36	OUT	Tout					R/D	
P37	OUT							

Notes:
HS = Handshake Signals
D = DAV
R = RDY

Comparator Inputs. Port 3, Pins P31 and P32 each have a comparator front end. The comparator reference voltage, Pin P33, is common to both comparators. In analog mode, the P31 and P32 are the positive inputs to the comparators and P33 is the reference voltage supplied to both comparators. In digital mode, pin P33 can be used as a P33 register input or IRQ1 source.

/RESET. (input, active-Low). Initializes the MCU. Reset is accomplished either through Power-On, Watch Dog Timer reset, STOP Mode Recovery , or external reset. During Power-On Reset and Watch Dog Reset, the internally generated reset is driving the reset pin low for the POR time. Any devices driving the reset line must be open-drain to avoid damage from a possible conflict during reset conditions. Pull-up is provided internally.

After the POR time, /RESET is a Schmitt triggered input. To avoid asynchronous and noisy reset problems, the Z86C40 is equipped with a reset filter of four external clocks (4TpC). If the external reset signal is less than 4TpC in duration, no reset occurs. On the fifth clock after the reset is detected, an internal RST signal is latched and held for an internal register count of 18 external clocks, or for the duration of the external reset, whichever is longer. During the reset cycle, /DS is held active low while /AS cycles at a rate of TpC/2. Program execution begins at location 000C (HEX), 5-10 TpC cycles after the RST is released. For Power-On Reset, the reset output time is 5 ms. The Z86C40 does not reset WDTMR, SMR, P2M, and P3M registers on a STOP Mode Recovery operation.

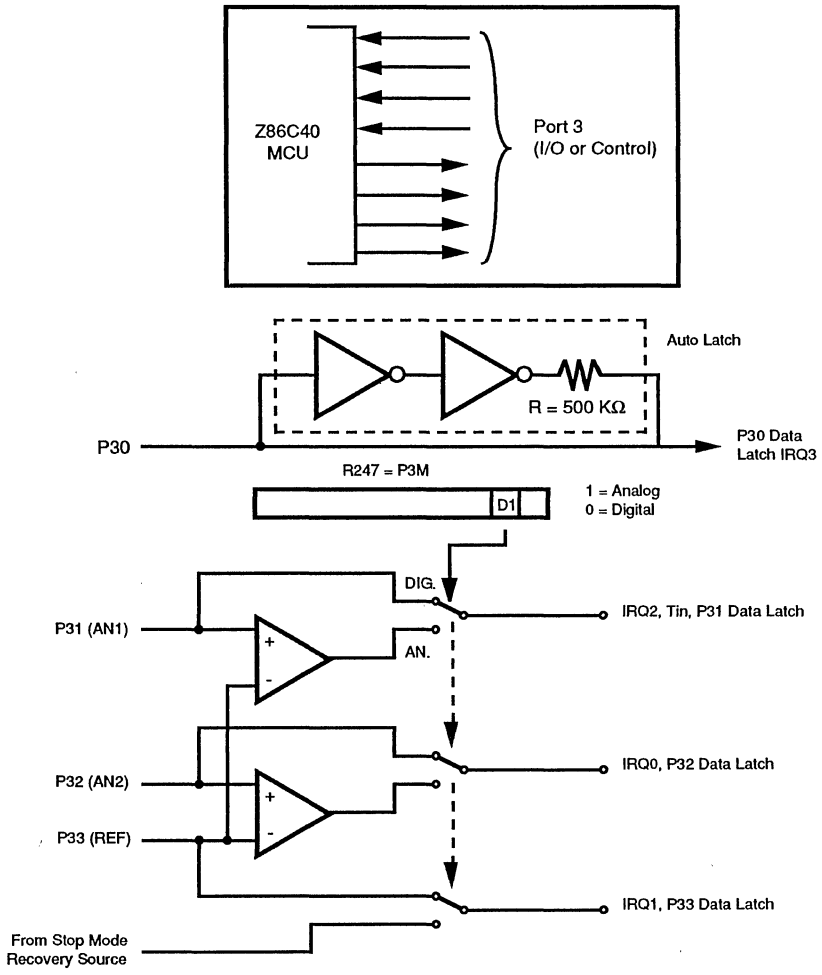


Figure 8. Port 3 Configuration

FUNCTIONAL DESCRIPTION

The Z8 CCP incorporates special functions to enhance the Z8's application in industrial, scientific research and advanced technologies applications.

Reset. The device is reset in one of the following conditions:

- Power-On Reset
- Watch-Dog Timer
- STOP Mode Recovery Source
- Brown-out Recovery
- External Reset

Program Memory. The Z86C40 addresses up to 4 Kbytes internal program memory and 60 Kbytes external memory (Figure 9). The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts. For ROM mode, byte 13 to byte 4095 consists of on-chip mask-programmed ROM. At addresses 4096 and greater, the Z86C40 executes external program memory fetches.

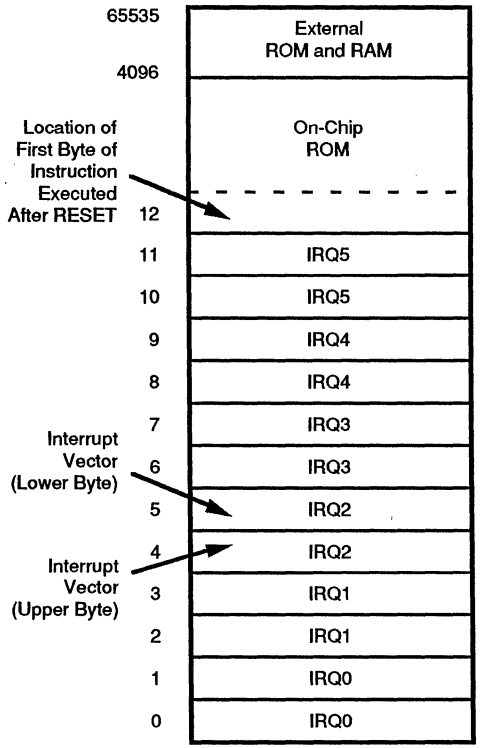


Figure 9. Program Memory Map

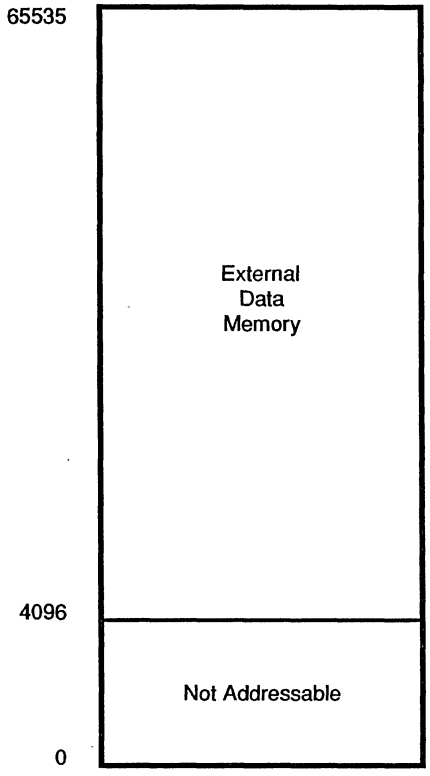


Figure 10. Data Memory Map

The 4 Kbyte program memory is mask programmable. A ROM protect feature prevents "dumping" of the ROM contents by inhibiting execution of LDC, LDCI, LDE, and LDEI instructions to Program Memory in all modes.

The ROM Protect option is mask-programmable, to be selected by the customer at the time when the ROM code is submitted.

Data Memory (/DM). The Z86C40 ROM version can address up to 60 Kbytes of external data memory beginning at location 4096 (Figure 10). External data memory may be included with, or separated from, the external program memory space. /DM, an optional I/O function that can be programmed to appear on pin P34, is used to distinguish between data and program memory space (Figure 8). The state of the /DM signal is controlled by the type of instruction being executed. An LDC opcode references PROGRAM (/DM inactive) memory, and an LDE instruction references data (/DM active Low) memory.

Expanded Register File. The register file has been expanded to allow for additional system control registers, and for mapping of additional peripheral devices along with I/O ports into the register address area. The Z8 register address space R0 through R15 has now been implemented as 16 groups of 16 registers per group (Figure 11). These register groups are known as the ERF (Expanded Register File). Bits 7-4 of register RP select the working register group. Bits 3-0 of register RP select the expanded register group (Figure 12). Two system configuration registers reside in the Expanded Register File at Bank F. The rest of the Expanded Register is not physically implemented and is open for future expansion.

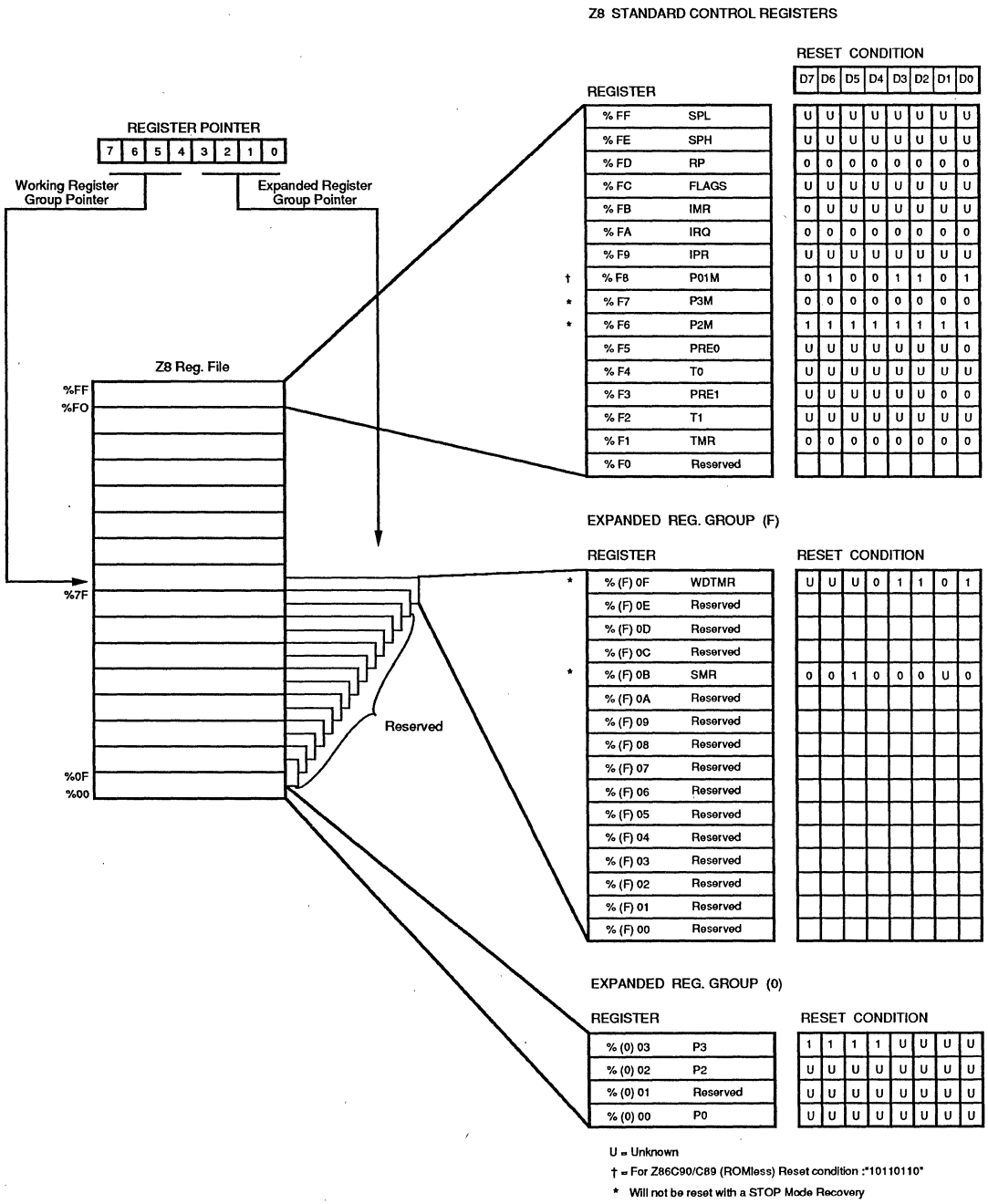


Figure 11. Expanded Register File Architecture

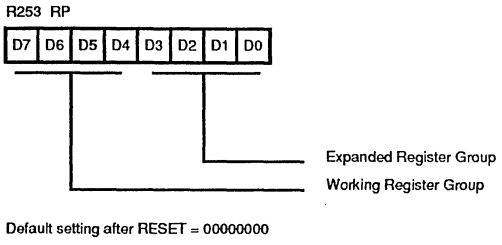


Figure 12. Register Pointer Register

Register File. The register file consists of four I/O port registers, 236 general purpose registers and 15 control and status registers (R0-R3, R4-239 and R240-R255, respectively), plus two system configuration registers in the expanded register group. The instructions access registers directly or indirectly via an 8-bit address field. This allows a short, 4-bit register address using the Register Pointer (Figure 13). In the 4-bit mode, the register file is divided into 16 working register groups, each occupying 16 continuous locations. The Register Pointer addresses the starting location of the active working register group.

Note: Register Bank E0-EF is only accessed through working register and indirect addressing modes.

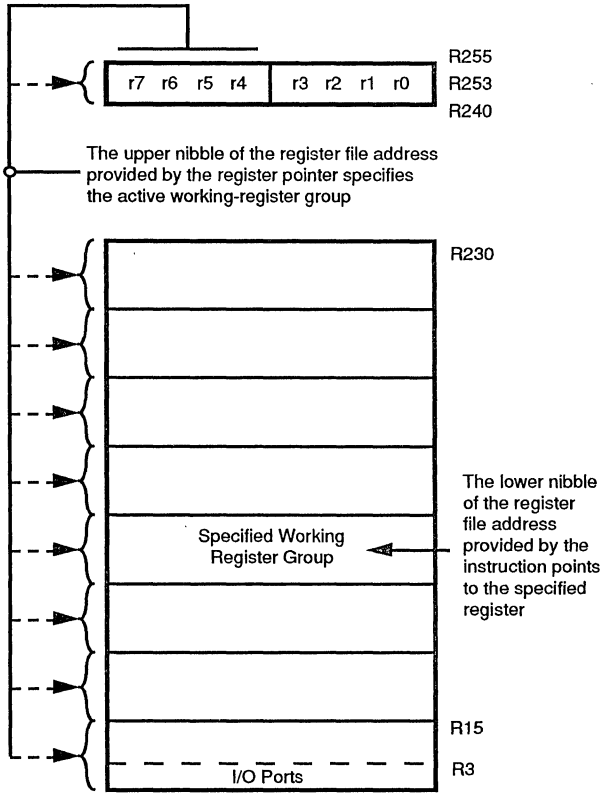


Figure 13. Register Pointer

RAM Protect. The upper portion of the RAM's address spaces %80F to %EF (excluding the control registers) are protected from reading and writing. The RAM Protect bit option is mask-programmable and is selected by the customer when the ROM code is submitted. After the mask option is selected, the user activates from the internal ROM code to turn off/on the RAM Protect by loading a bit D6 in the IMR register to either a 0 or a 1, respectively. A 1 in D6 indicates RAM Protect enabled.

Stack. The Z86C40 external data memory or the internal register file is used for the stack. The 16-bit Stack Pointer (R254-R255) is used for the external stack which can reside anywhere in the data memory for ROMless mode, but only from 4096 to 65535 in ROM mode. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 236 general purpose registers (R4-R239). SPH is used as a general purpose register when using internal stack only.

Counter/Timers. There are two 8-bit programmable counter/timers (T0-T1), each driven by its own 6-bit programmable prescaler. The T1 prescaler is driven by internal or external clock sources; however, the T0 prescaler is driven by the internal clock only (Figure 14).

The 6-bit prescalers can divide the input frequency of the clock source by any integer number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of the count, a timer interrupt request, IRQ4 (T0) or IRQ5 (T1), is generated.

The counters can be programmed to start, stop, restart to continue, or restart from the initial value. The counters can also be programmed to stop upon reaching zero (single pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode).

The counters, but not the prescalers, are read at any time without disturbing their value or count mode. The clock source for T1 is user-definable and is either the internal microprocessor clock divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input (P31) as an external clock, a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. The counter/timers can be cascaded by connecting the T0 output to the input of T1.

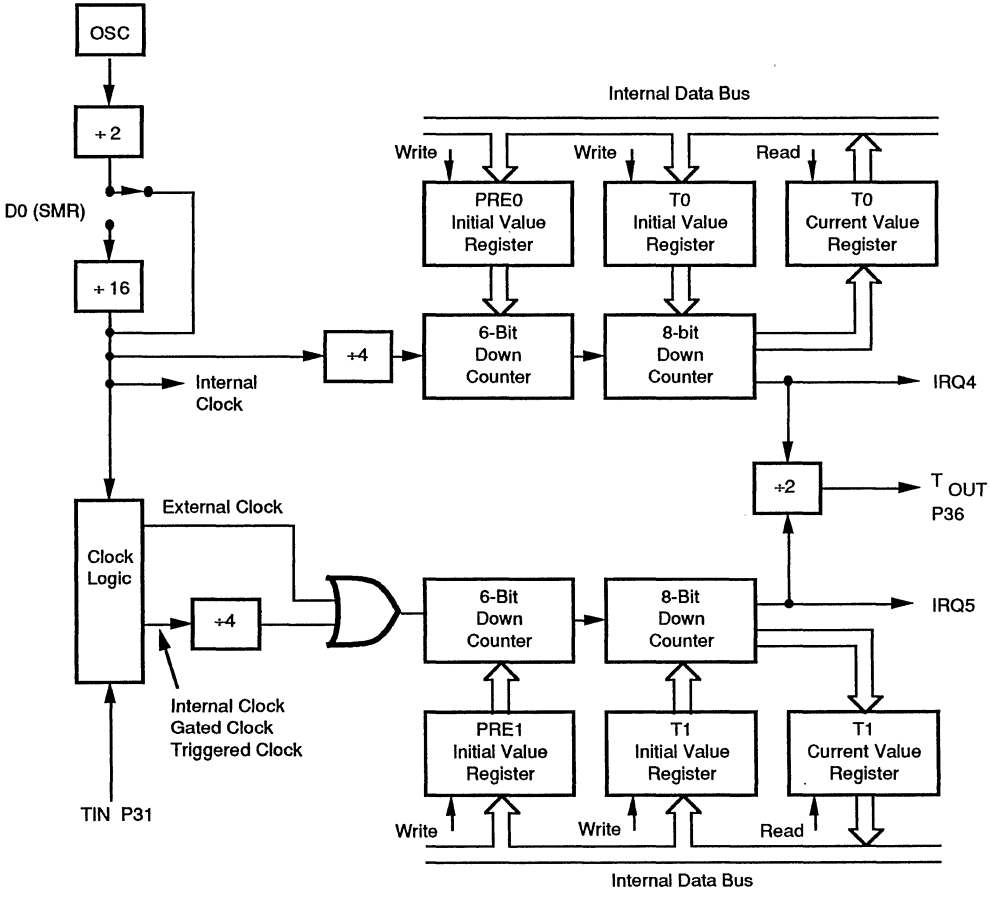


Figure 14. Counter/Timer Block Diagram

Interrupts. The Z86C40 has six different interrupts from six different sources. The interrupts are maskable and prioritized (Figure 15). The six sources are divided as follows; four sources are claimed by Port 3 lines P30-P33, and two in counter/timers (Table 5). The Interrupt Mask Register globally or individually enables or disables the six interrupt requests.

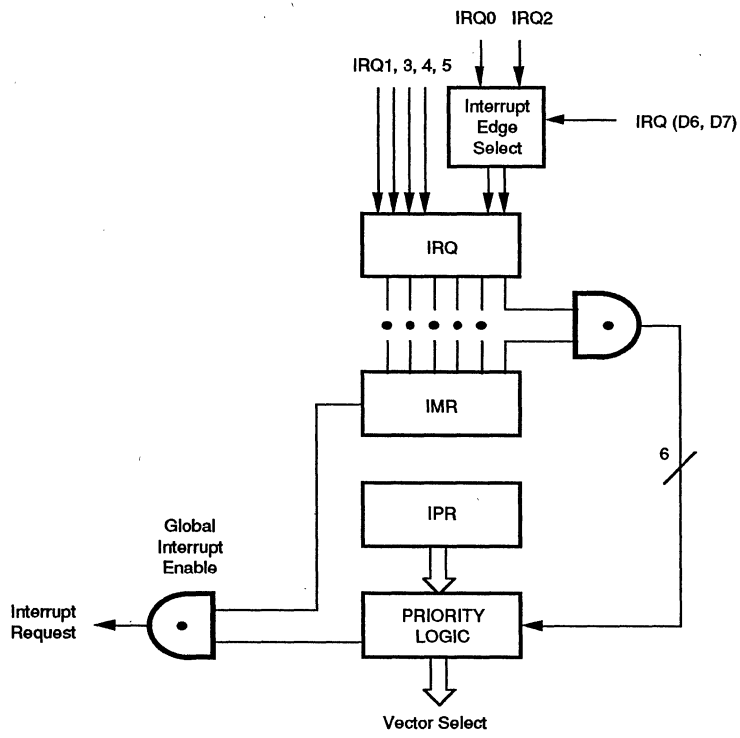


Figure 15. Interrupt Block Diagram

Table 5. Interrupt Types, Sources, and Vectors

Name	Source	Vector Location	Comments
IRQ 0	/DAV 0, IRQ 0	0, 1	External (P32), Rise Fall Edge Triggered
IRQ 1,	IRQ 1	2, 3	External (P33), Fall Edge Triggered
IRQ 2	/DAV 2, IRQ 2, TIN	4, 5	External (P31), Rise Fall Edge Triggered
IRQ 3	IRQ3	6, 7	External (P30), Fall Edge Triggered
IRQ 4	T0	8, 9	Internal
IRQ 5	TI	10, 11	Internal

When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register. An interrupt machine cycle is activated when an interrupt request is granted. Thus, this disables all subsequent interrupts, saves the Program Counter and Status Flags, and then branches to the program memory vector location reserved for that interrupt.

All Z86C40 interrupts are vectored through locations in the program memory. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request. To accommodate polled interrupt systems, interrupt inputs are masked and the Interrupt Request register is polled to determine which of the interrupt requests need service.

An interrupt resulting from AN1 is mapped into IRQ2, and an interrupt from AN2 is mapped into IRQ0. Interrupts IRQ2 and IRQ0 may be rising, falling or both edge triggered, and are programmable by the user. The software may poll to identify the state of the pin.

Programming bits for the Interrupt Edge Select is located in the IRQ Register (R250), bits D7 and D6. The configuration is shown in Table 6.

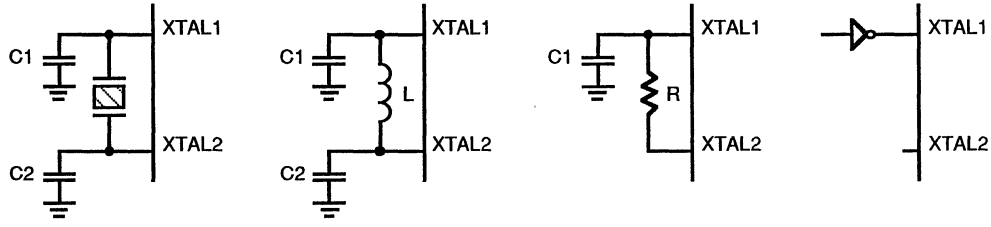
Table 6. IRQ Register

IRQ		Interrupt Edge	
D7	D6	P31	P32
0	0	F	F
0	1	F	R
1	0	R	F
1	1	R/F	R/F

Notes:
F=Falling Edge
R=Rising Edge

Clock. The Z86C40 on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal, LC, RC, ceramic resonator, or any suitable external clock source (XTAL1 = Input, XTAL2 = Output). The crystal should be AT cut, 12 MHz max., with a series resistance (RS) less than or equal to 100 Ohms.

The crystal is connected across XTAL1 and XTAL2 using the recommended capacitors (capacitance is more than or equal to 22 pF) from each pin to ground. The RC oscillator option is mask-programmable on the Z86C40 and is selected by the customer at the time when the ROM code is submitted. (Note that the RC option is not available on the 12 MHz part). The RC oscillator configuration must be an external resistor connected from XTAL1 to XTAL2, with a frequency-setting capacitor from XTAL1 to ground (Figure 16). See Figures 52-54 for typical characteristics.



Ceramic Resonator or Crystal
C1, C2 = 47 pF TYP *
f = 8 MHz

LC
C1, C2 = 22 pF

L = 130 uH *
f = 3 MHz *

RC
@ 5V VCC (TYP)

C1 = 33 pF *
R = 1K *
f = 6 MHz *

External Clock

* Preliminary value including pin parasitics

Figure 16. Oscillator Configuration

FUNCTIONAL DESCRIPTION (Continued)

Power-On-Reset (POR). A timer circuit clocked by a dedicated on-board RC oscillator is used for the Power-On Reset (POR) timer function. The POR time allows V_{CC} and the oscillator circuit to stabilize before instruction execution begins.

The POR timer circuit is a one-shot timer triggered by one of three conditions:

1. Power fail to Power OK status.
2. STOP mode recovery (if D5 of SMR=1).
3. WDT timeout.

The POR time is a nominal 5 ms. Bit 5 of the Stop Mode Register determines whether the POR timer is bypassed after STOP mode recovery (typical for external clock, RC/LC oscillators).

HALT. HALT turns off the internal CPU clock, but not the XTAL oscillation. The counter/timers and external interrupts IRQ0, IRQ1, IRQ2, and IRQ3 remain active. The devices are recovered by interrupts, either externally or internally generated.

STOP. This instruction turns off the internal clock and external crystal oscillation. It reduces the standby current to 10 microamperes or less. The Stop mode is terminated by a reset only, either by WDT timeout, POR, SMR recovery or external reset. This causes the processor to restart the application program at address 000C (HEX). In order to enter STOP (or HALT) mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. To do this, the user must execute a NOP (opcode=FFH) immediately before the appropriate sleep instruction, i.e.:

```
FF  NOP   ; clear the pipeline
6F  STOP  ; enter STOP mode
          or
FF  NOP   ; clear the pipeline
7F  HALT  ; enter HALT mode
```

Stop Mode Recovery Register (SMR). This register selects the clock divide value and determines the mode of STOP Mode Recovery (Figure 17). All bits are write only, except Bit 7 which is read only. Bit-7 is a flag bit that is hardware set on the condition of STOP recovery and reset by a power-on cycle. Bit-6 controls whether a low level or a high level is required from the recovery source. Bit-5 controls the reset delay after recovery. Bits 2, 3, and 4, or the SMR register, specify the source of the STOP Mode Recovery signal. Bits 0 and 1 determine the timeout period of the WDT. The SMR is located in Bank F of the Expanded Register Group at address OBH.

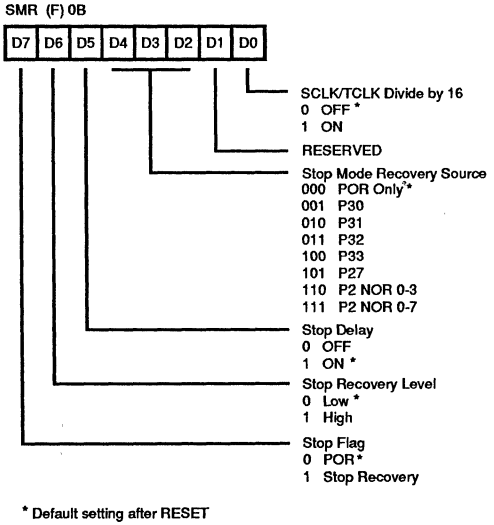


Figure 17. STOP Mode Recovery Register

SCLK/TCLK divide-by-16 Select (D0). D0 of the SMR controls a divide-by-16 prescaler of SCLK/TCLK. The purpose of this control is to selectively reduce device power consumption during normal processor execution (SCLK control) and/or HALT mode (where TCLK sources counter/timers and interrupt logic).

STOP Mode Recovery Source (D2, D3, and D4). These three bits of the SMR specify the wake-up source of the STOP recovery (Figure 18 and Table 7).

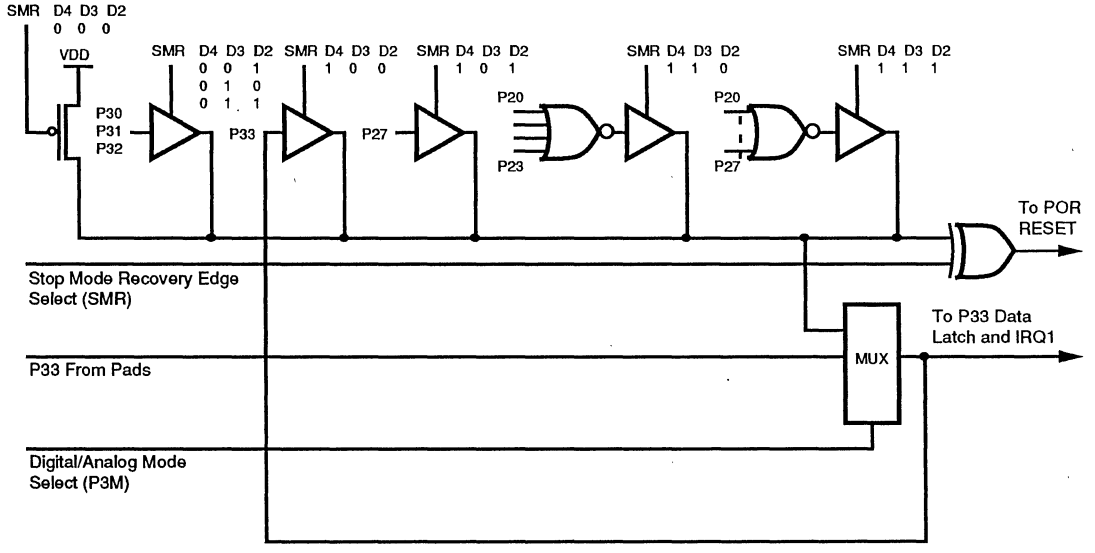


Figure 18. STOP Mode Recovery Source

Table 7. STOP Mode Recovery Source

SMR:432			Operation
D4	D3	D2	Description of Action
0	0	0	POR and/or external reset recovery
0	0	1	P30 transition
0	1	0	P31 transition
0	1	1	P32 transition
1	0	0	P33 transition
1	0	1	P27 transition
1	1	0	Logical NOR of P20 through P23
1	1	1	Logical NOR of P20 through P27

STOP Mode Recovery Delay Select (D5). This bit, if high, disables the 5 ms /RESET delay after STOP Mode Recovery. The default configuration of this bit is one. If the "fast" wake up is selected the STOP Mode Recovery source is kept active for at least 5 TpC.

STOP Mode Recovery Edge Select (D6). A 1 in this bit position indicates that a high level on any one of the recovery sources wakes the Z86C40 from STOP Mode. A 0 indicates low level recovery. The default is 0 on POR (Figure 18).

Cold or Warm Start (D7). This bit is set by the device upon entering STOP Mode. A 0 in this bit (cold) indicates that the device resets by POR/WDT RESET. A 1 in this bit (warm) indicates that the device awakens by a SMR source.

Watch-Dog-Timer Mode Register (WDTMR). The WDT is a retriggerable one-shot timer that resets the Z8 if it reaches its terminal count. The WDT is initially enabled by executing the WDT instruction and refreshed on subsequent executions of the WDT instruction. The WDT circuit is driven by an on-board RC oscillator or external oscillator from the XTAL1 pin. The POR clock source is selected with bit-4 of the WDT register (Figure 19).

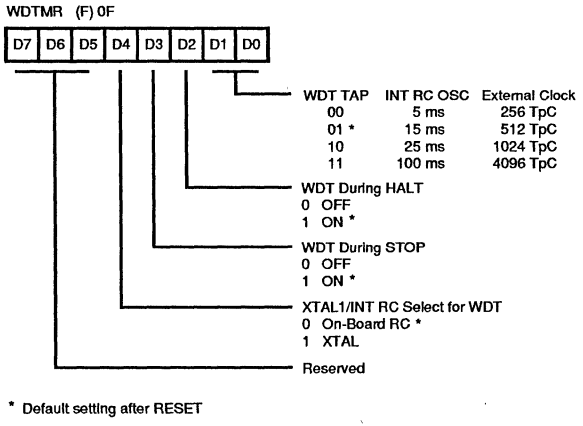


Figure 19. Watch-Dog Timer Mode Register

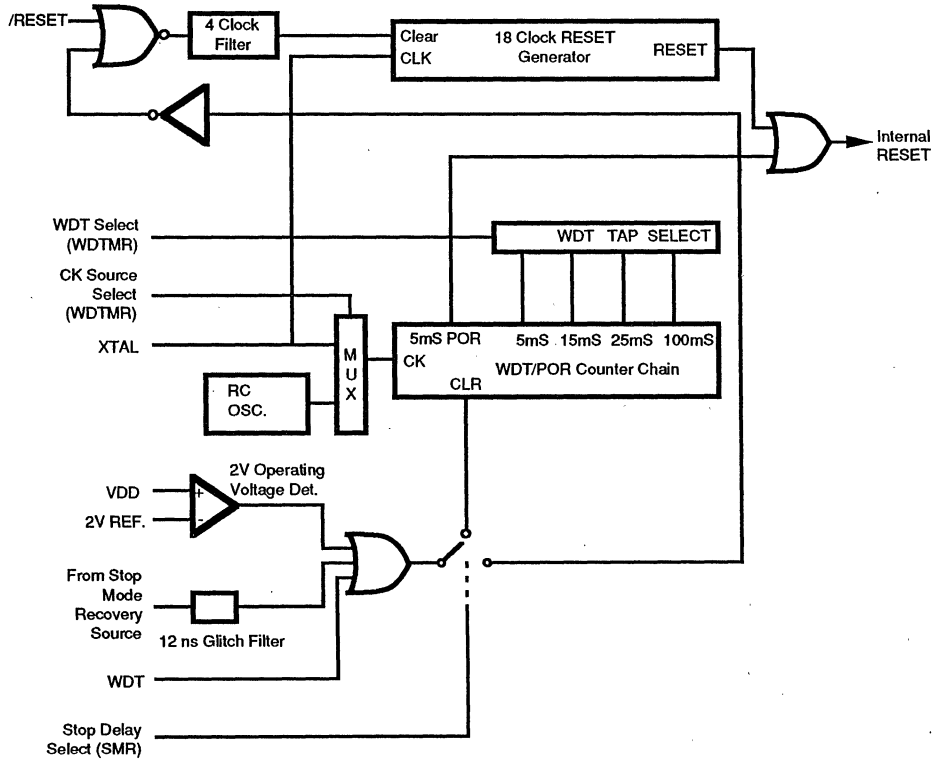


Figure 20. Resets and WDT

WDT Time Select. (D0,D1). Selects the WDT time period. It is configured as shown in Table 8.

Table 8. WDT Time Select

D1	D0	Timeout of Internal RC OSC	Timeout of XTAL clock
0	0	5 ms min	256 TpC
0	1	15 ms min	512 TpC
1	0	25 ms min	1024 TpC
1	1	100 ms min	4096 TpC

Notes:
TpC = XTAL clock cycle
The default on reset is 15 ms.
See Figures 55 to 58 for details.

WDTMR During HALT (D2). This bit determines whether or not the WDT is active during HALT Mode. A 1 indicates active during HALT. The default is 1.

WDTMR During STOP (D3). This bit determines whether or not the WDT is active during STOP Mode. Since XTAL clock is stopped during STOP Mode, the on-board RC has to be selected as the clock source to the POR counter. A 1 indicates active during STOP. The default is 1.

Clock Source for WDT (D4). This bit determines which oscillator source is used to clock the internal POR and WDT counter chain. If the bit is a 1, the internal RC oscillator is bypassed and the POR and WDT clock source is driven from the external pin, XTAL1. The default configuration of this bit is 0 which selects the RC oscillator.

Brown Out Protection. An on-board Voltage Comparator checks that V_{CC} is at the required level to ensure correct operation of the device. Reset is globally driven if V_{CC} is below the specified voltage (Brown Out Voltage). The minimum operating voltage is varying with the temperature and operating frequency, while the brown out voltage (V_{BO}) varies with temperature only.

The brown out trip voltage (V_{BO}) is less then 3 volts and above 1.4 volts under the following conditions.

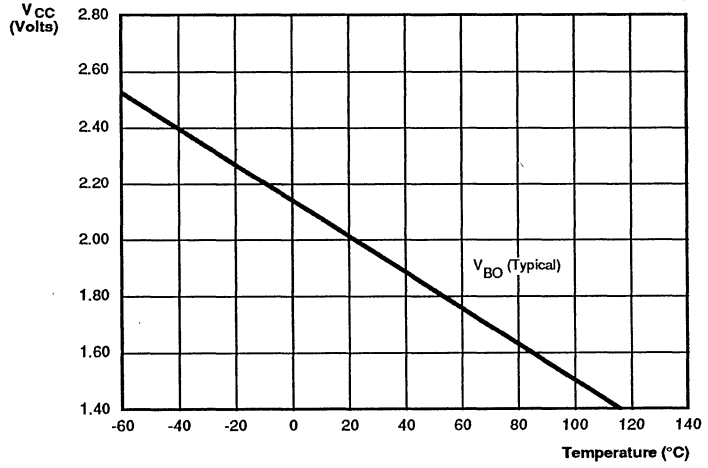
Maximum (V_{BO}) Conditions:

Case 1: T_A = -40°C, + 105°C, Internal Clock Frequency equal or less than 1 MHz

Case 2: T_A = -40°C, + 85°C, Internal Clock Frequency equal or less than 2 MHz

Note: The internal clock frequency is one-half the external clock frequency.

The device functions normally at or above 3.0V under all conditions. Below 3.0V, the device functions normally until the Brown Out Protection trip point (V_{BO}) is reached, for the temperatures and operating frequencies in case 1 and case 2, above. The device is guaranteed to function normally at supply voltages above the brown out trip point. The actual brown out trip point is a function of temperature and process parameters (Figure 21).



* Power-on Reset threshold for V_{CC} and 4 MHz V_{BO} overlap

Figure 21. Typical Z86C40 Brown Out Voltage vs Temperature At 4 MHz

ABSOLUTE MAXIMUM RATINGS

Symbol	Description	Min	Max	Units
V_{CC}	Supply Voltage (*)	-0.3	+7.0	V
T_{STG}	Storage Temp	-65	+150	C
T_A	Oper Ambient Temp			C
	Power Dissipation		2.2	W

Notes:
* Voltage on all pins with respect to GND.
See Ordering Information.

Stress greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for an extended period may affect device reliability.

STANDARD TEST CONDITIONS

The characteristics listed below apply for standard test conditions as noted. All voltages are referenced to GND. Positive current flows into the referenced pin (Figure 22).

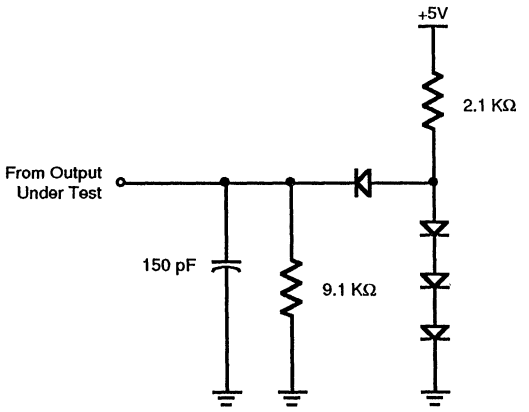


Figure 22. Test Load Diagram

CAPACITANCE

T_A = 25°C, V_{CC} = GND = 0V, f = 1.0 MHz, Unmeasured pins to GND

Parameter	Max
Input capacitance	12 pF
Output capacitance	12 pF
I/O capacitance	12 pF

DC ELECTRICAL CHARACTERISTICS

Sym	Parameter	V_{CC} Note [3]	T_A = 0° C to 70°C Min	Max	T_A = -40°C to 105°C Min	Max	Typical at 25°C	Units	Conditions	Notes
	Max Input Voltage	3.3V		7		7		V	I_{IN} 250 uA	
		5.0V		7		7		V	I_{IN} 250 uA	
V_{CH}	Clock Input High Voltage	3.3V	0.7 V_{CC}	V_{CC}+0.3	0.7 V_{CC}	V_{CC}+0.3	1.3	V	Driven by External Clock Generator	
		5.0V	0.7 V_{CC}	V_{CC}+0.3	0.7 V_{CC}	V_{CC}+0.3	2.5	V	Driven by External Clock Generator	
V_{CL}	Clock Input Low Voltage	3.3V	GND-0.3	0.2 V_{CC}	GND-0.3	0.2 V_{CC}	0.7	V	Driven by External Clock Generator	
		5.0V	GND-0.3	0.2 V_{CC}	GND-0.3	0.2 V_{CC}	1.5	V	Driven by External Clock Generator	
V_{IH}	Input High Voltage	3.3V	0.7 V_{CC}	V_{CC}+0.3	0.7 V_{CC}	V_{CC}+0.3	1.3	V		
		5.0V	0.7 V_{CC}	V_{CC}+0.3	0.7 V_{CC}	V_{CC}+0.3	2.5	V		
V_{IL}	Input Low Voltage	3.3V	GND-0.3	0.2 V_{CC}	GND-0.3	0.2 V_{CC}	0.7	V		
		5.0V	GND-0.3	0.2 V_{CC}	GND-0.3	0.2 V_{CC}	1.5	V		
V_{OH}	Output High Voltge	3.3V	V_{CC}-0.4		V_{CC}-0.4		3.1	V	I_{OH} = -2.0 mA	
		5.0V	V_{CC}-0.4		V_{CC}-0.4		4.8	V	I_{OH} = -2.0 mA	
V_{OL1}	Output Low Voltage	3.3V		0.6		0.6	0.2	V	I_{OH} = +4.0 mA	
		5.0V		0.4		0.4	0.1	V	I_{OL} = +4.0 mA	
V_{OL2}	Output Low Voltage	3.3V		1.2		1.2	0.3	V	I_{OL} = +6 mA, 3 Pin Max	
		5.0V		1.2		1.2	0.3	V	I_{OL} = +12 mA, 3 Pin Max	
V_{RH}	Reset Input High Voltage	3.3V	.8 V_{CC}	V_{CC}	.8 V_{CC}	V_{CC}	1.5	V		
		5.0V	.8 V_{CC}	V_{CC}	.8 V_{CC}	V_{CC}	2.1	V		
V_{RI}	Reset Input Low Voltage	3.3V	GND-0.3	0.2 V_{CC}	GND-0.3	0.2 V_{CC}	1.1			
		5.0V	GND-0.3	0.2 V_{CC}	GND-0.3	0.2 V_{CC}	1.7			
V_{OFFSET}	Comparator Input Offset	3.3V		25		25	10	mV		
	Voltage	5.0V		25		25	10	mV		
I_{IL}	Input Leakage	3.3V	-1	1	-1	2	<1	µA	V_{IN} = 0V, V_{CC}	
		5.0V	-1	1	-1	2	<1	µA	V_{IN} = 0V, V_{CC}	
I_{OL}	Output Leakage	3.3V	-1	1	-1	2	<1	µA	V_{IN} = 0V, V_{CC}	
		5.0V	-1	1	-1	2	<1	µA	V_{IN} = 0V, V_{CC}	
I_{IR}	Reset Input Current	3.3V		-45		-60	-20	µA		
		5.0V		-55		-70	-30	µA		
I_{CC}	Supply Current	3.3V		10		10	4	mA	@ 8 MHz	[4,5]
		5.0V		15		15	10	mA	@ 8 MHz	[4,5]
		3.3V		15		15	5	mA	@ 12 MHz	[4,5]
		5.0V		20		20	15	mA	@ 12 MHz	[4,5]
I_{CC1}	Standby Current	3.3V		3		3	1	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 8 MHz	[4,5]
		5.0V		5		5	2.4	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 8 MHz	[4,5]
		3.3V		4		4	1.5	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 12 MHz	[4,5]
		5.0V		6		6	3.2	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 12 MHz	[4,5]
		3.3V		2		2	0.8	mA	Clock Divide by 16 @ 8 MHz	[4,5]
		5.0V		4		4	1.8	mA	Clock Divide by 16 @ 8 MHz	[4,5]
		3.3V		3		3	1.2	mA	Clock Divide by 16 @ 12 MHz	[4,5]
		5.0V		5		5	2.5	mA	Clock Divide by 16 @ 12 MHz	[4,5]
I_{CC2}	Standby Current	3.3V		8		15	1	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is not Running	[6]
		5.0V		10		20	2	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is not Running	[6]

DC ELECTRICAL CHARACTERISTICS (Continued)

Sym	Parameter	V_{CC} Note [3]	T_A = 0° C to 70°C Min	Max	T_A = -40°C to 105°C Min	Max	Typical at 25°C	Units	Conditions	Notest
		3.3V		500		600	310	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is Running	[6]
		5.0V		800		1000	600	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is Running	[6]
I_{ALL}	Auto Latch Low Current	3.3V		8		10	5	µA	$0V < V_{IN} < V_{CC}$	
		5.0V		15		20	11	µA	$0V < V_{IN} < V_{CC}$	
I_{ALH}	Auto Latch High Current	3.3V		-5		-7	-3	µA	$0V < V_{IN} < V_{CC}$	
		5.0V		-8		-10	-6	µA	$0V < V_{IN} < V_{CC}$	
T_{POR}	Power On Reset	3.3V	7	24	7	25	13	mS		
		5.0V	3	13	3	14	7	mS		
V_{BO}	V_{CC} Brown Out Voltage		1.5	2.65	1.2	2.95	2.1	V	2 MHz max Ext. CLK Freq.	[7]

Note:

[1]

I_{CC1}	Typ	Max	Unit	Freq
Clock Driven on Crystal or	3.0 mA	5	mA	8 MHz
XTAL Resonator	0.3 mA	50	mA	8 MHz

[2] GND=0V.
[3] 5.0V ± 0.5V,3.3V ± 0.3V.
[4] All outputs unloaded, I/O pins floating, inputs at rail.
[5] CL1=CL2=100pF
[6] Same as note [4] except inputs at V_{CC}.
[7] The V_{BO} increases as the temperature decreases.

AC CHARACTERISTICS

External I/O or Memory Read and Write Timing Diagram

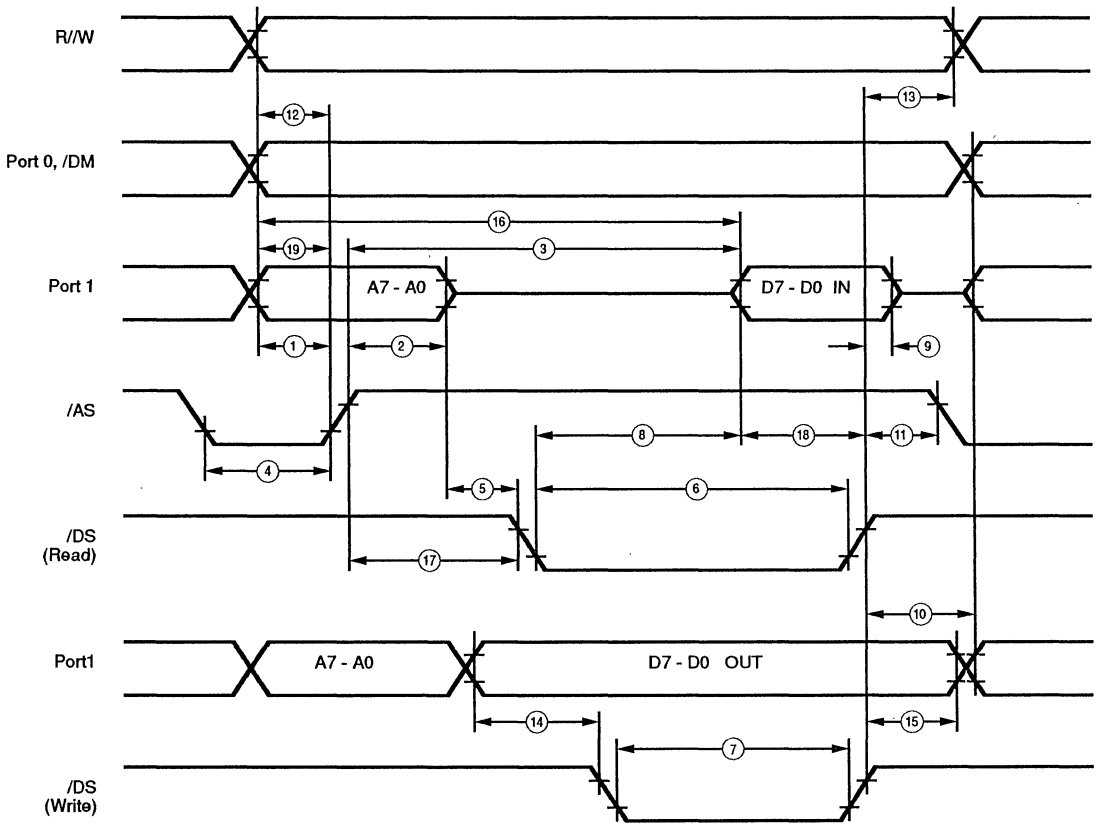


Figure 23. External I/O or Memory Read/Write Timing

AC CHARACTERISTICS

External I/O or Memory Read and Write Timing Table

No	Symbol	Parameter	V_{CC} [3]	T_A = 0°C to 70°C				T_A = -40°C to 105°C				Units	Notes
				8 MHz		12 MHz		8 MHz		12 MHz			
				Min	Max	Min	Max	Min	Max	Min	Max		
1	TdA(AS)	Address Valid to /AS Rise Delay	3.3	55		35		55		35		ns	[2]
			5.0	55		35		55		35			
2	TdAS(A)	/AS Rise to Address Float Delay	3.3	70		45		70		45		ns	[2]
			5.0	70		45		70		45		ns	
3	TdAS(DR)	/AS Rise to Read Data Req'd Valid	3.3		400		250		400		250	ns	[1,2]
			5.0		400		250		400		250	ns	
4	TwAS	/AS Low Width	3.3	80		55		80		55		ns	[2]
			5.0	80		55		80		55		ns	
5	Td	Address Float to /DS Fall	3.3	0		0		0		0		ns	
			5.0	0		0		0		0		ns	
6	TwDSR	/DS (Read) Low Width	3.3	300		200		300		200		ns	[1,2]
			5.0	300		200		300		200		ns	
7	TwDSW	/DS (Write) Low Width	3.3	165		110		165		110		ns	[1,2]
			5.0	165		110		165		110		ns	
8	TdDSR(DR)	/DS Fall to Read Data Req'd Valid	3.3		260		150		260		150	ns	[1,2]
			5.0		260		160		260		160	ns	
9	ThDR(DS)	Read Data to /DS Rise Hold Time	3.3	0		0		0		0		ns	[2]
			5.0	0		0		0		0		ns	
10	TdDS(A)	/DS Rise to Address Active Delay	3.3	85		45		85		45		ns	[2]
			5.0	95		55		95		55		ns	
11	TdDS(AS)	/DS Rise to /AS Fall Delay	3.3	60		30		60		30		ns	[2]
			5.0	70		45		70		45		ns	
12	TdR/W(AS)	R/W Valid to /AS Rise Delay	3.3	70		45		70		45		ns	[2]
			5.0	70		45		70		45		ns	
13	TdDS(R/W)	/DS Rise to R/W Not Valid	3.3	70		45		70		45		ns	[2]
			5.0	70		45		70		45		ns	
14	TdDW(DSW)	Write Data Valid to /DS Fall (Write) Delay	3.3	80		55		80		55		ns	[2]
			5.0	80		55		80		55		ns	
15	TdDS(DW)	/DS Rise to Write Data Not Valid Delay	3.3	70		45		70		45		ns	[2]
			5.0	80		55		80		55		ns	
16	TdA(DR)	Address Valid to Read Data Req'd Valid	3.3		475		310		475		310	ns	[1,2]
			5.0		475		310		475		310	ns	
17	TdAS(DS)	/AS Rise to /DS Fall Delay	3.3	100		65		100		65		ns	[2]
			5.0	100		65		100		65		ns	
18	TdDI(DS)	Data Input Setup to /DS Rise	0.0	115		115		115		115		ns	[1,2]
			5.0	75		75		75		75		ns	

AC CHARACTERISTICS

External I/O or Memory Read and Write Timing Table (Continued)

No	Symbol	Parameter	V_{cc} [3]	T_A = 0°C to 70°C				T_A = -40°C to 105°C				Units	Notes
				8 MHz		12 MHz		8 MHz		12 MHz			
				Min	Max	Min	Max	Min	Max	Min	Max		
19	TdDM(AS)	/DM Valid to /AS Fall Delay	3.3	55		35		55		35		ns	[2]
			5.0	55		35		55		35		ns	

Notes:
[1] When using extended memory timing add 2 TpC.
[2] Timing numbers given are for minimum TpC.
[3] 5.0V ± 0.5V, 3.3V ± 0.3V.

Standard Test Load
All timing references use 0.9 V_{cc} for a logic 1 and 0.1 V_{cc} for a logic0.

AC ELECTRICAL CHARACTERISTICS
Additional Timing Diagram

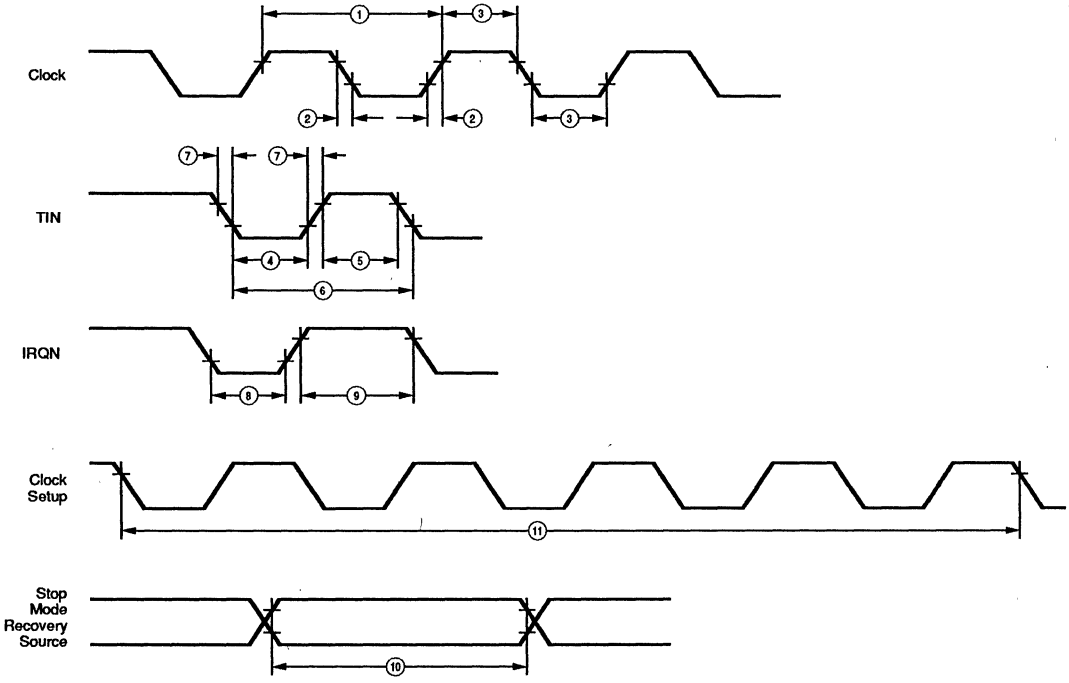


Figure 24. Additional Timing

AC ELECTRICAL CHARACTERISTICS
Additional Timing Table

No	Symbol	Parameter	V_{CC} Note[6]	T_A = 0°C to 70°C 8 MHz Min	8 MHz Max	12 MHz Min	12 MHz Max	T_A = -40°C to 105°C 8 MHz Min	8 MHz Max	12 MHz Min	12 MHz Max	Units	Notes
1	TpC	Input Clock Period	3.3V	125	100000	83	100000	125	100000	83	100000	ns	[1]
			5.0V	125	100000	83	100000	125	100000	83	100000	ns	[1]
2	TrC,TfC	Clock Input Rise & Fall Times	3.3V		25		15		25		15	ns	[1]
			5.0V		25		15		25		15	ns	[1]
3	TwC	Input Clock Width	3.3V	37		26		37		26		ns	[1]
			5.0V	37		26		37		26		ns	[1]
4	TwTinL	Timer Input Low Width	3.3V	100		100		100		100		ns	[1]
			5.0V	70		70		70		70		ns	[1]
5	TwTinH	Timer Input High Width	3.3V	3TpC		3TpC		3TpC		3TpC			[1]
			5.0V	3TpC		3TpC		3TpC		3TpC			[1]
6	TpTin	Timer Input Period	3.3V	8TpC		8TpC		8TpC		8TpC			[1]
			5.0V	8TpC		8TpC		8TpC		8TpC			[1]
7	TrTin, TfTin	Timer Input Rise & Fall Timer	3.3V		100		100		100		100	ns	[1]
			5.0V		100		100		100		100	ns	[1]
8A	TwIL	Int. Request Low Time	3.3V	100		100		100		100		ns	[1,2]
			5.0V	70		70		70		70		ns	[1,2]
8B	TwIL	Int. Request Low Time	3.3V	3TpC		3TpC		3TpC		3TpC			[1,3]
			5.0V	3TpC		3TpC		3TpC		3TpC			[1,3]
9	TwIH	Int. Request Input High Time	3.3V	3TpC		3TpC		3TpC		3TpC			[1,2]
			5.0V	3TpC		3TpC		3TpC		3TpC			[1,2]
10	Twsm	STOP Mode Recovery Width Spec	3.3V	12		12		12		12		ns	
			5.0V	12		12		12		12		ns	
			3.3V	5TpC									Reg. SMR - D5=0
			5.0V	5TpC									Reg. SMR - D5=1
11	Tost	Oscillator Startup Time	3.3V		5TpC		5TpC		5TpC		5TpC		[4]
			5.0V		5TpC		5TpC		5TpC		5TpC		[4]
12	Twdt	Watchdog Timer Delay Time	3.3V	10		10		10		10		ms	D0 = 0 [5]
			5.0V	5		5		5		5		ms	D1 = 0 [5]
			3.3V	30		30		30		30		ms	D0 = 1 [5]
			5.0V	15		15		15		15		ms	D1 = 0 [5]
			3.3V	50		50		50		50		ms	D0 = 0 [5]
			5.0V	25		25		25		25		ms	D1 = 1 [5]
			3.3V	200		200		200		200		ms	D0 = 1 [5]
			5.0V	100		100		100		100		ms	D1 = 1 [5]

Notes:
[1] Timing Reference uses 0.9 V_{CC} for a logic 1 and 0.1 V_{CC} for a logic 0.
[2] Interrupt request via Port 3 (P31-P33).
[3] Interrupt request via Port 3 (P30).
[4] SMR-D5 = 0.
[5] Reg. WDTMR.
[6] 5.0V ± 0.5V, 3.3V ± 0.3V.

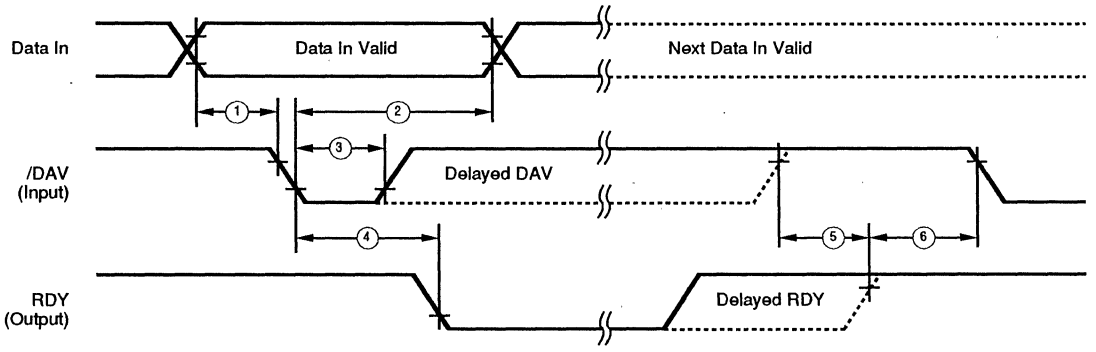


Figure 25. Input Handshake Timing

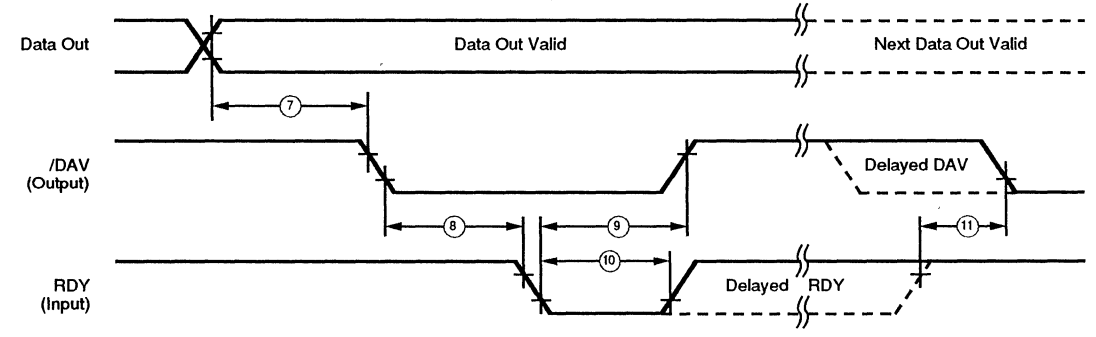


Figure 26. Output Handshake Timing

AC ELECTRICAL CHARACTERISTICS
Handshake Timing Table

No	Symbol	Parameter	V_{cc} Note[1]	T_A = 0°C to 70°C				T_A = -40°C to 105°C				Notes Data Direction
				8 MHz		12 MHz		8 MHz		12 MHz		
				Min	Max	Min	Max	Min	Max	Min	Max	
1	TsDI(DAV)	Data In Setup Time	3.3V	0		0		0		0		IN
			5.0V	0		0		0		0		IN
2	ThDI(DAV)	Data In Hold Time	3.3V	160		160		160		160		IN
			5.0V	115		115		115		115		IN
3	TwDAV	Data Available Width	3.3V	155		155		155		155		IN
			5.0V	110		110		110		110		IN
4	TdDAVl(RDY)	DAV Fall to RDY Fall Delay	3.3V		160		160		160		160	IN
			5.0V		115		115		115		115	IN
5	TdDAVld(RDY)	DAV Rise to RDY Rise Delay	3.3V		120		120		120		120	IN
			5.0V		80		80		80		80	IN
6	TdDO(DAV)	RDY Rise to DAV Fall Delay	3.3V	0		0		0		0		IN
			5.0V	0		0		0		0		IN
7	TcLDAV0(RDY)	Data Out to DAV Fall Delay	3.3V	63		42		63		42		OUT
			5.0V	63		42		63		42		OUT
8	TcLDAV0(RDY)	DAV Fall to RDY Fall Delay	3.3V	0		0		0		0		OUT
			5.0V	0		0		0		0		OUT
9	TdRDY0(DAV)	RDY Fall to DAV Rise Delay	3.3V		160		160		160		160	OUT
			5.0V		115		115		115		115	OUT
10	TwRDY	RDY Width	3.3V	110		110		110		110		OUT
			5.0V	80		80		80		80		OUT
11	TdRDY0d(DAV)	RDY Rise to DAV Fall Delay	3.3V		110		110		110		110	OUT
			5.0V		80		80		80		80	OUT

Notes:
[1] 5.0 V ± 0.5V, 3.3V ± 0.3V.

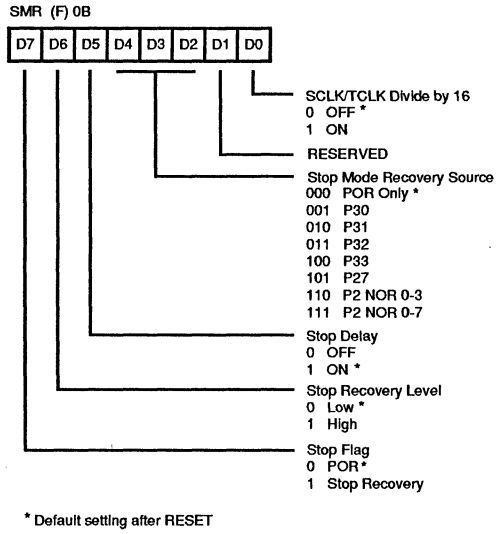


Figure 27. Stop Mode Recovery Register

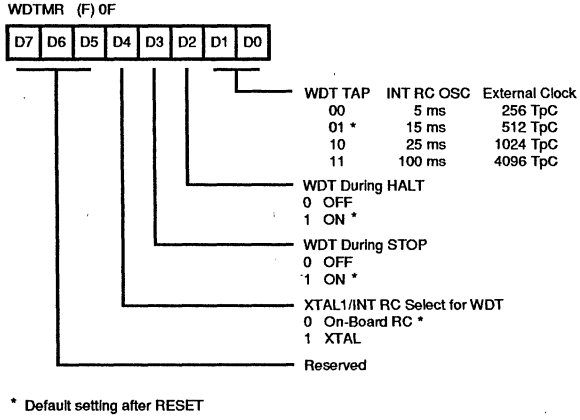


Figure 28. Watchdog Timer Mode Register

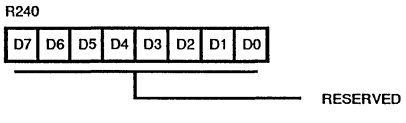


Figure 29. Reserved

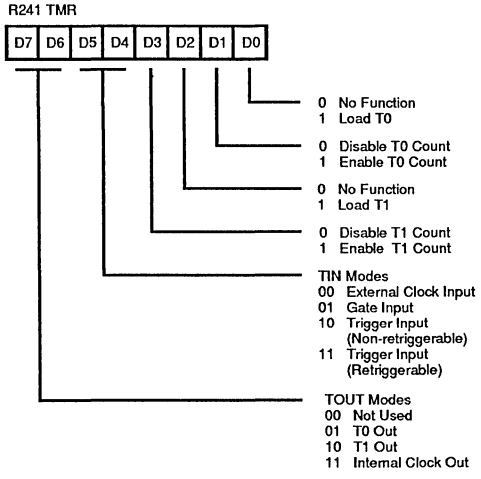


Figure 30. Timer Mode Register ($F1_H$:Read/Write)

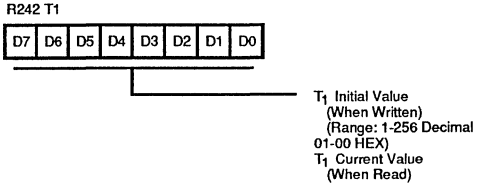


Figure 31. Counter/Timer 1 Register ($F2_H$:Read/Write)

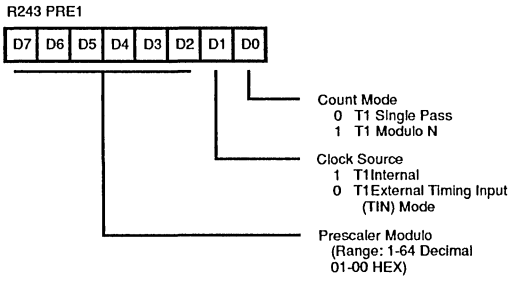


Figure 32. Prescaler 1 Register ($F3_H$:Write Only)

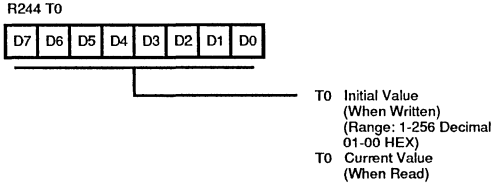


Figure 33. Counter/Timer 0 Register ($F4_H$:Read/Write)

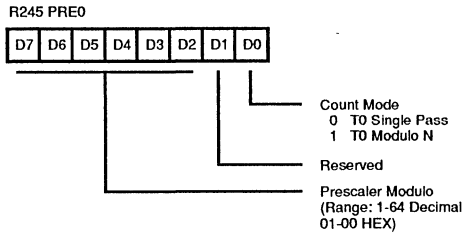


Figure 34. Prescaler 0 Register ($F5_H$:Write Only)

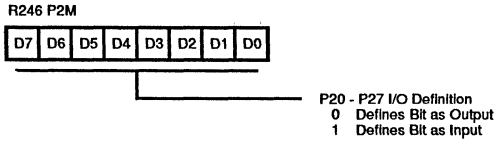


Figure 35. Port 2 Mode Register ($F6_H$: Write Only)

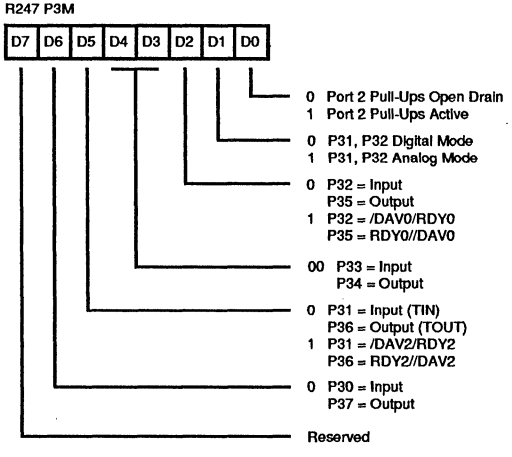


Figure 36. Port 3 Mode Register ($F7_H$:Write Only)

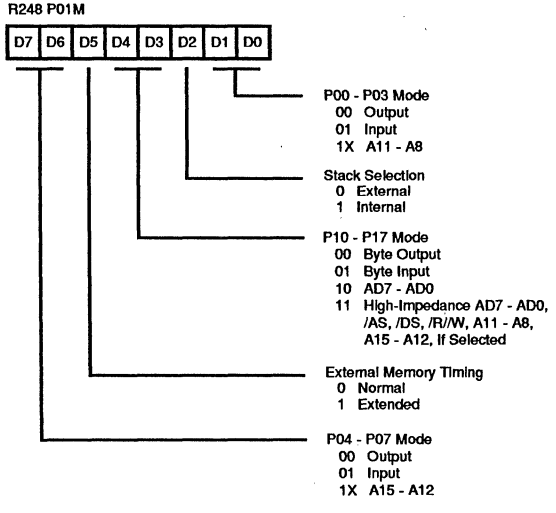


Figure 37. Port 0 and 1 Mode Register ($F8_H$:Write Only)

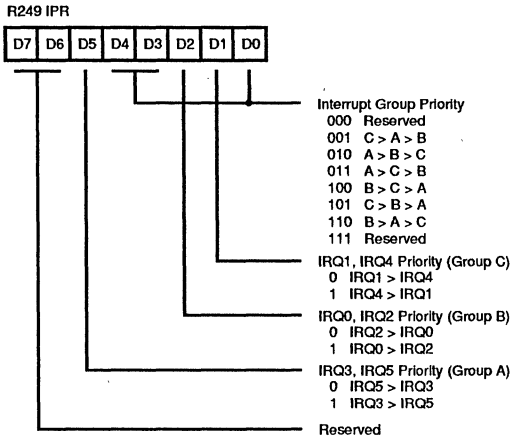


Figure 38. Interrupt Priority Register ($F9_H$:Write Only)

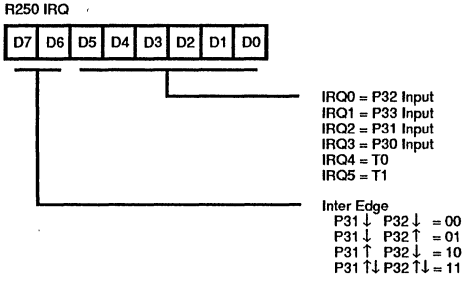


Figure 39. Interrupt Request Register (FA_H:Read/Write)

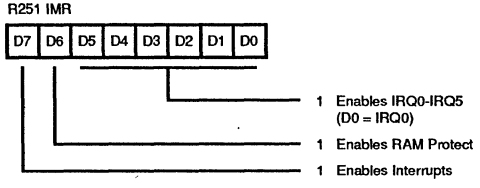


Figure 40. Interrupt Mask Register (FB_H:Read/Write)

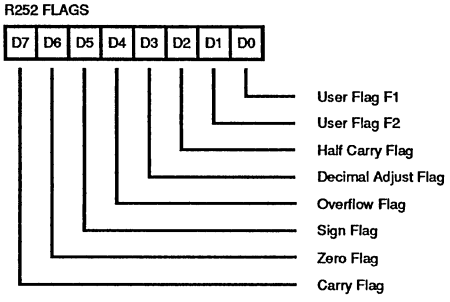


Figure 41. Flag Register (FC_H:Read/Write)

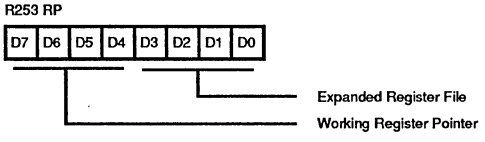


Figure 42. Register Pointer (FD_H:Read/Write)

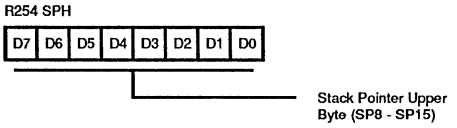


Figure 43. Stack Pointer High (FE_H:Read/Write)

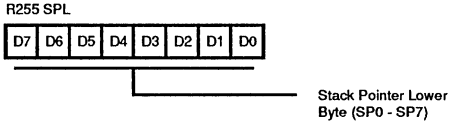


Figure 44. Stack Pointer Low (FF_H:Read/Write)

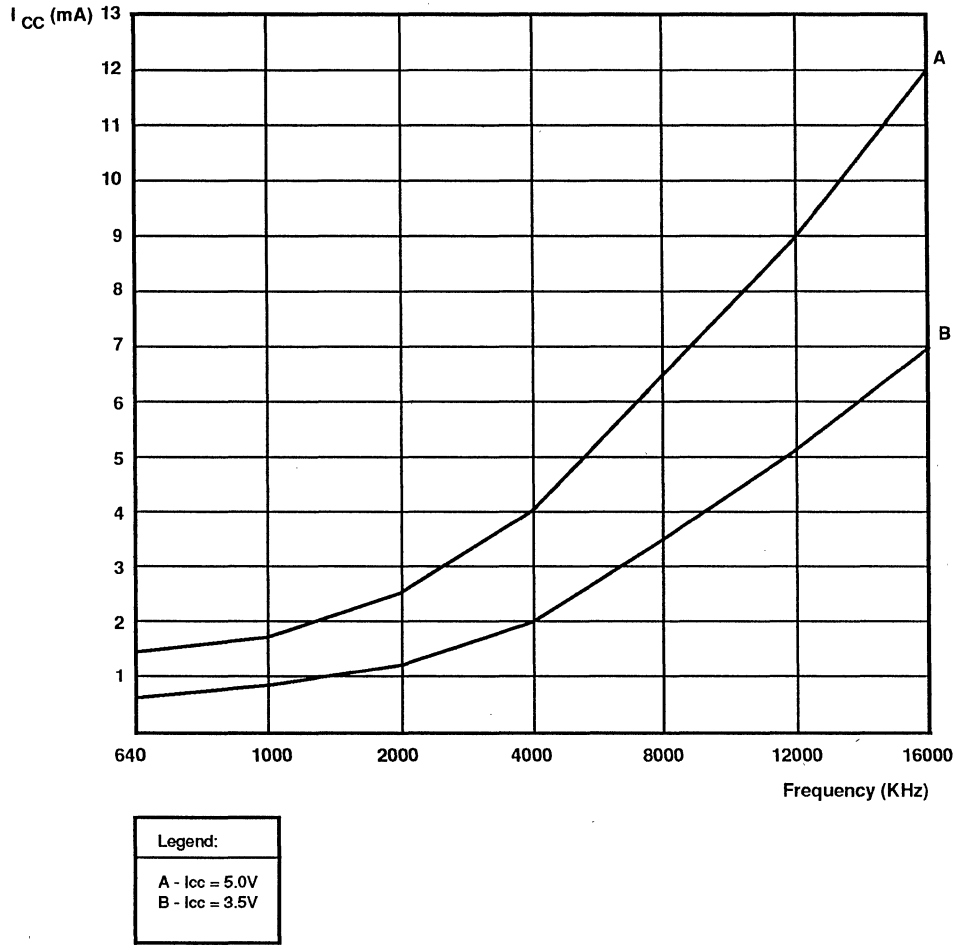


Figure 45. Typical I_{cc} vs Frequency

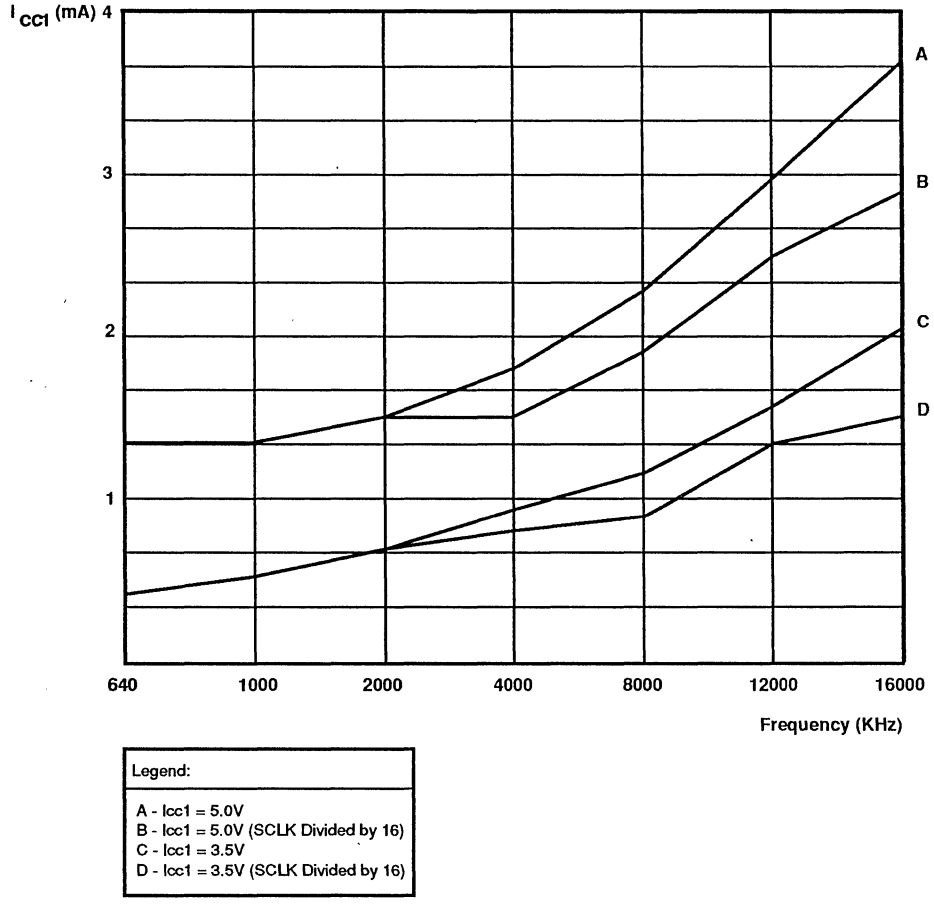


Figure 46. Typical I_{CC1} vs Frequency

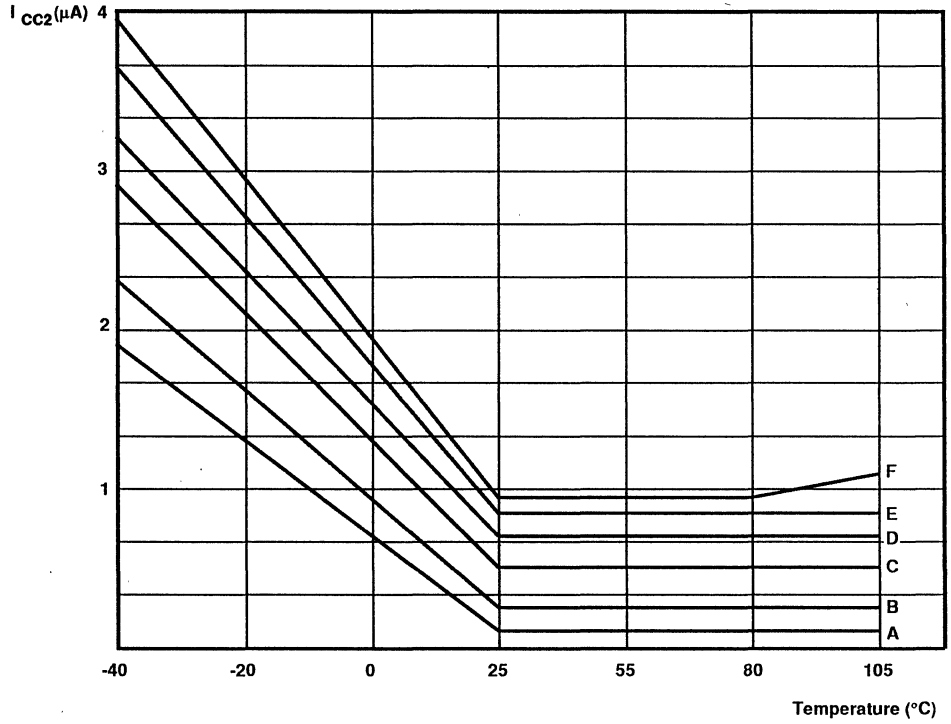
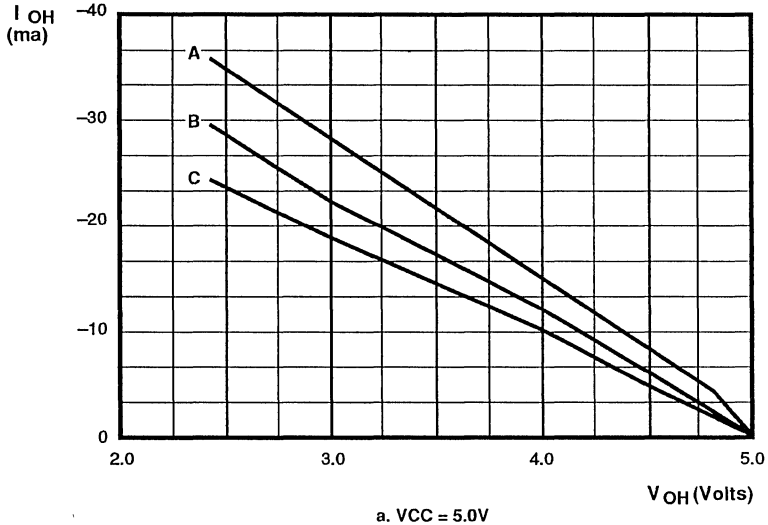
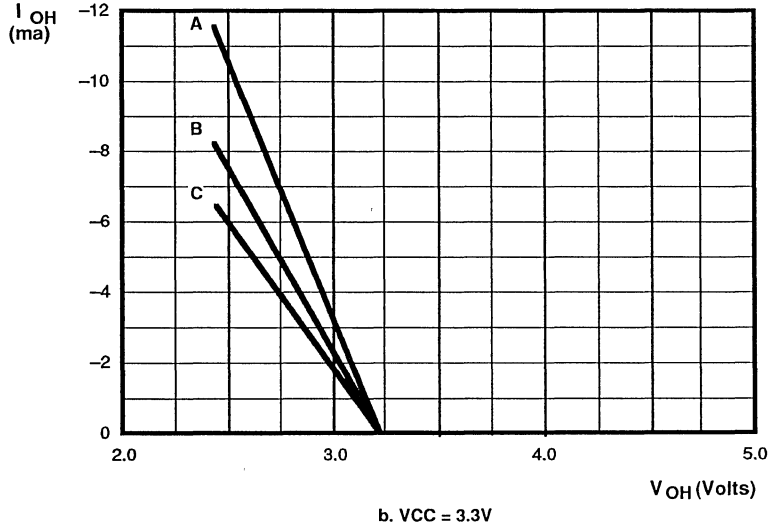


Figure 47. Typical I_{CC2} vs Frequency



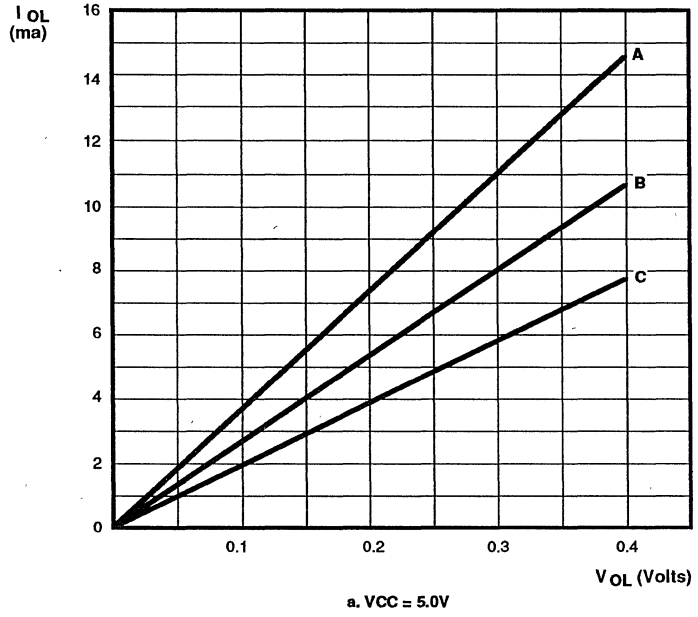
a. VCC = 5.0V



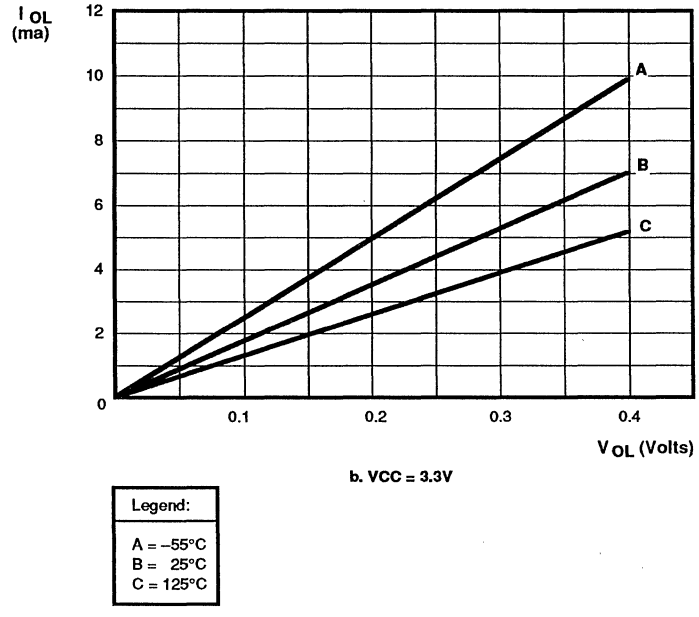
b. VCC = 3.3V

Legend:
A = –55°C B = 25°C C = 125°C

Figure 48. Typical I_{OH} vs V_{OH} Over Temperature



a. VCC = 5.0V



b. VCC = 3.3V

Legend:
A = –55°C
B = 25°C
C = 125°C

Figure 49. Typical I_{OL} vs V_{OL} Over Temperature

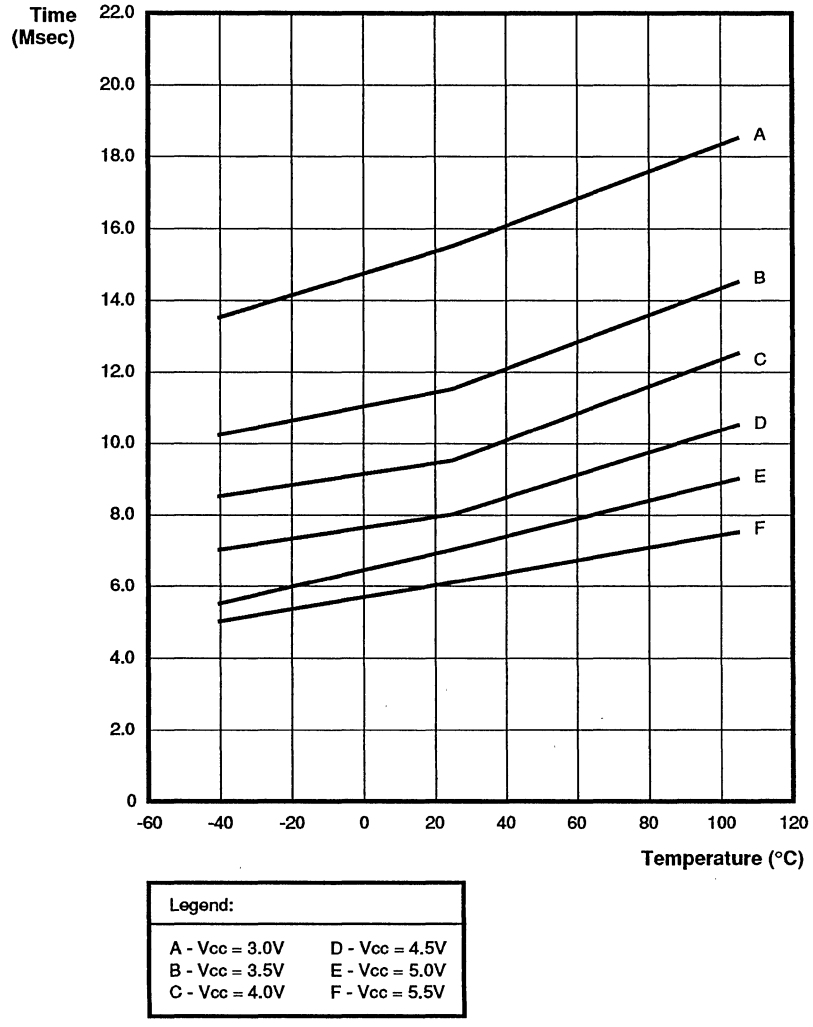
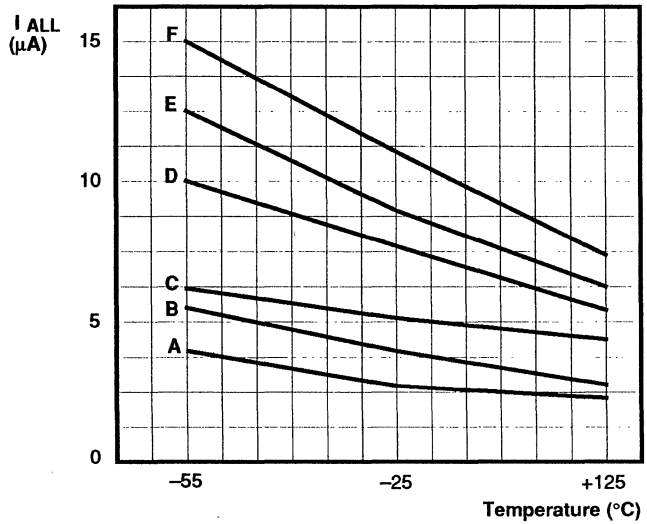
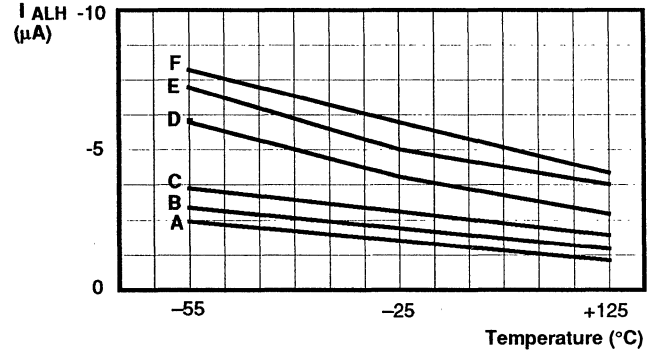


Figure 50. Typical Power-On Reset Time vs Temperature



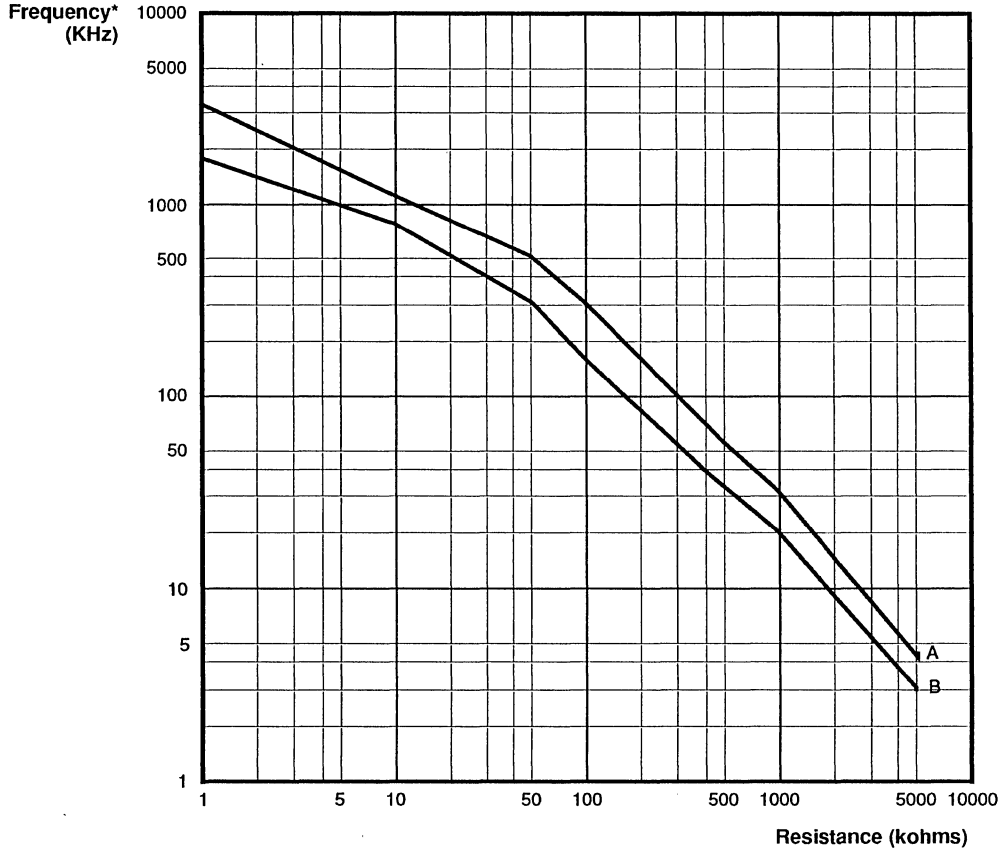
a. Typical Auto Latch Low Current vs Temperature



b. Typical Auto Latch High Current vs Temperature

Legend:	
A - Vcc = 3.0V	D - Vcc = 4.5V
B - Vcc = 3.3V	E - Vcc = 5.0V
C - Vcc = 3.6V	F - Vcc = 5.5V

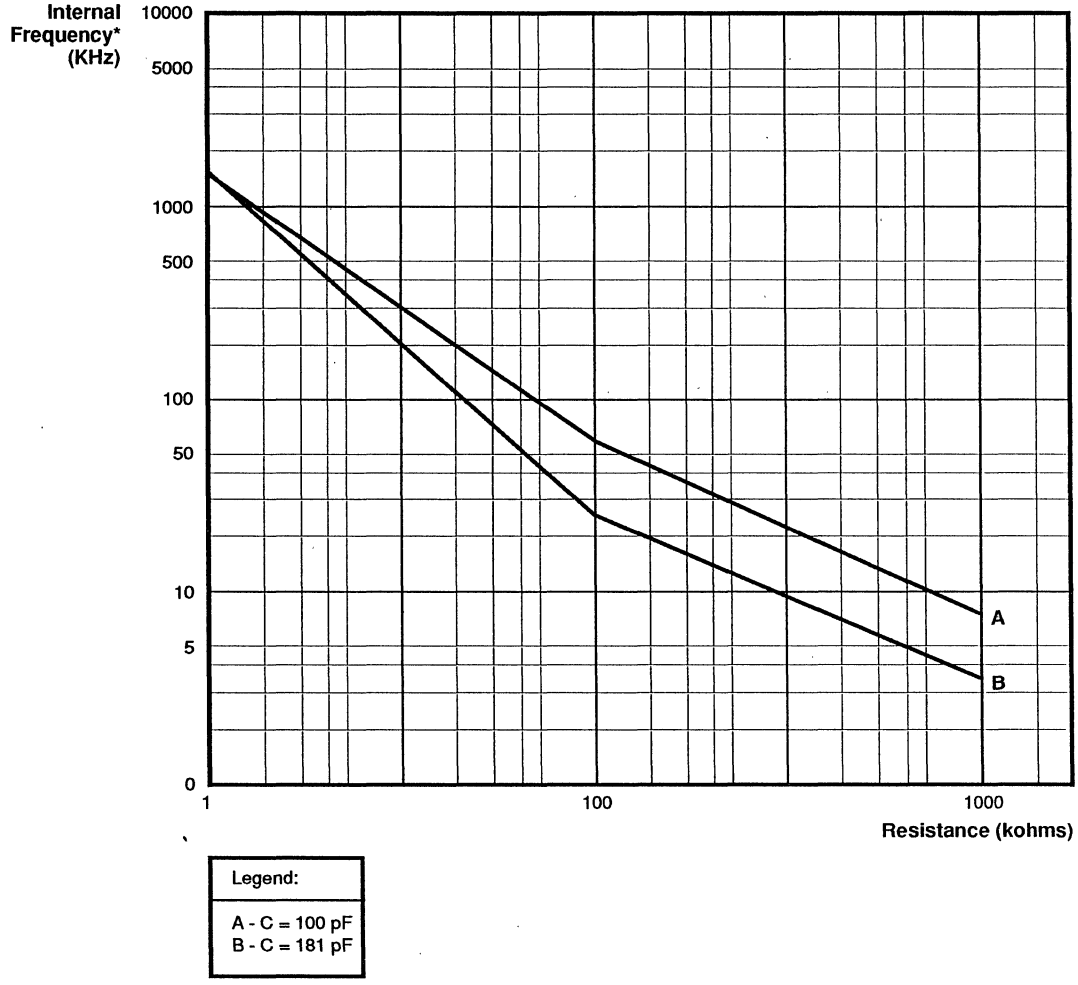
Figure 51. Typical Auto-Latch Current vs Temperature



Legend:
A - Vcc = 5.0V C = 33 pF B - Vcc = 3.3V C = 33 pF

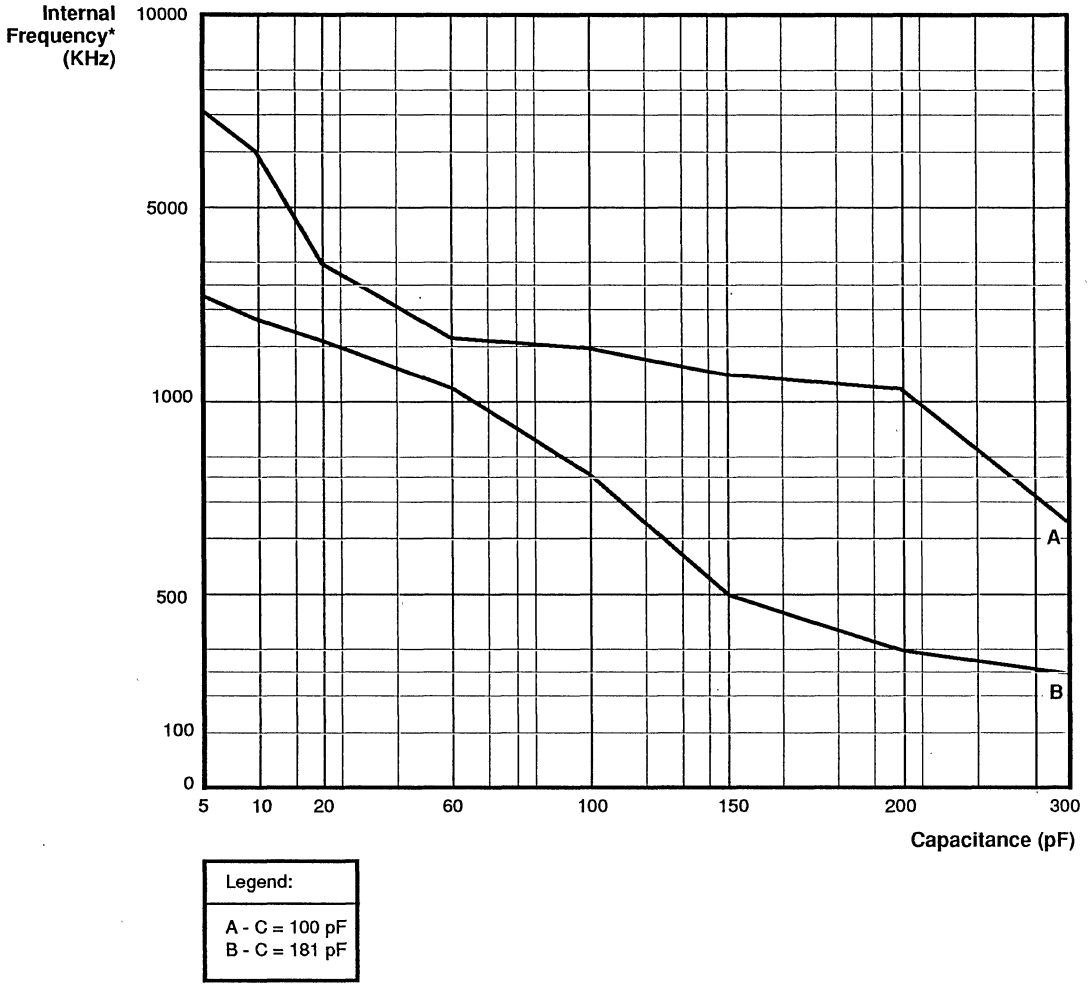
Note: * The internal clock frequency is one half the external clock frequency.
This chart for reference only. Each process will have a different characteristc curve.

Figure 52. Typical Internal Frequency vs RC Resistance



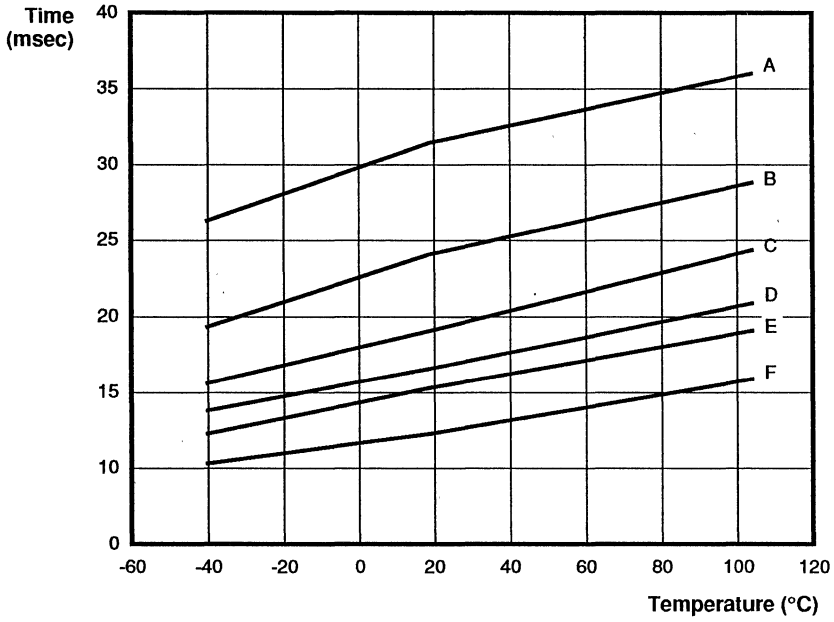
Note: * The internal clock frequency is one half the external clock frequency.
This chart for reference only. Each process will have a different characteristc curve.

Figure 53. Typical Internal Frequency vs Resistance



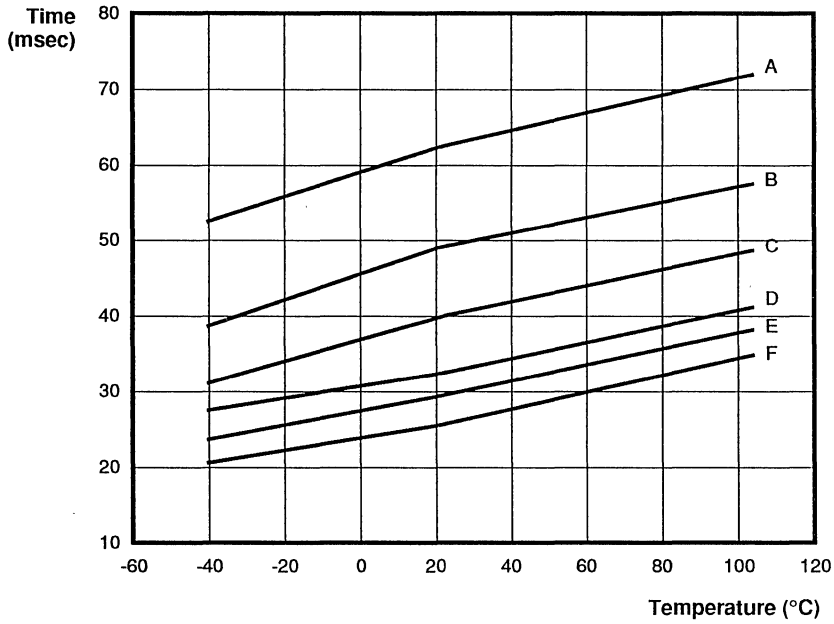
Note: * The internal clock frequency is one half the external clock frequency.
This chart for reference only. Each process will have a different characteristc curve.
R = 1 kohm

Figure 54. Typical Internal Frequency vs RC Capacitance



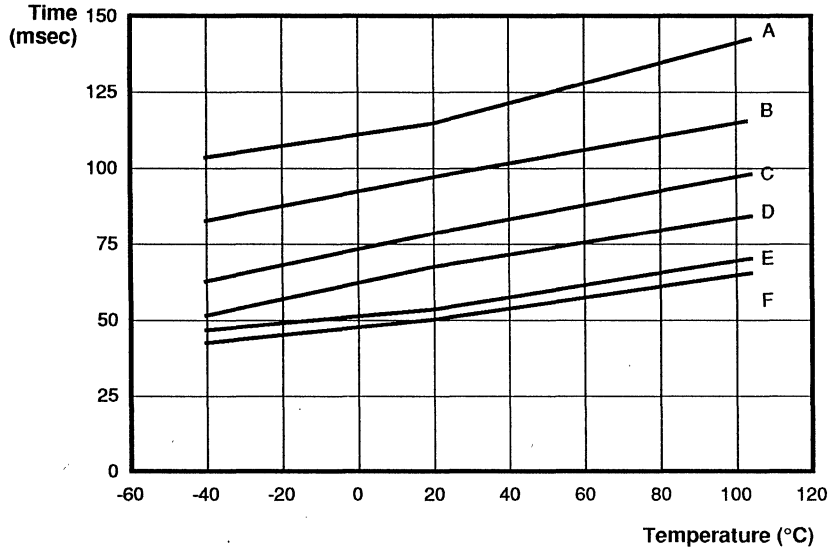
Legend:	
A - Vcc = 3.0V	D - Vcc = 4.5V
B - Vcc = 3.5V	E - Vcc = 5.0V
C - Vcc = 4.0V	F - Vcc = 5.5V

Figure 55. Typical 5 ms WDT Setting vs Temperature



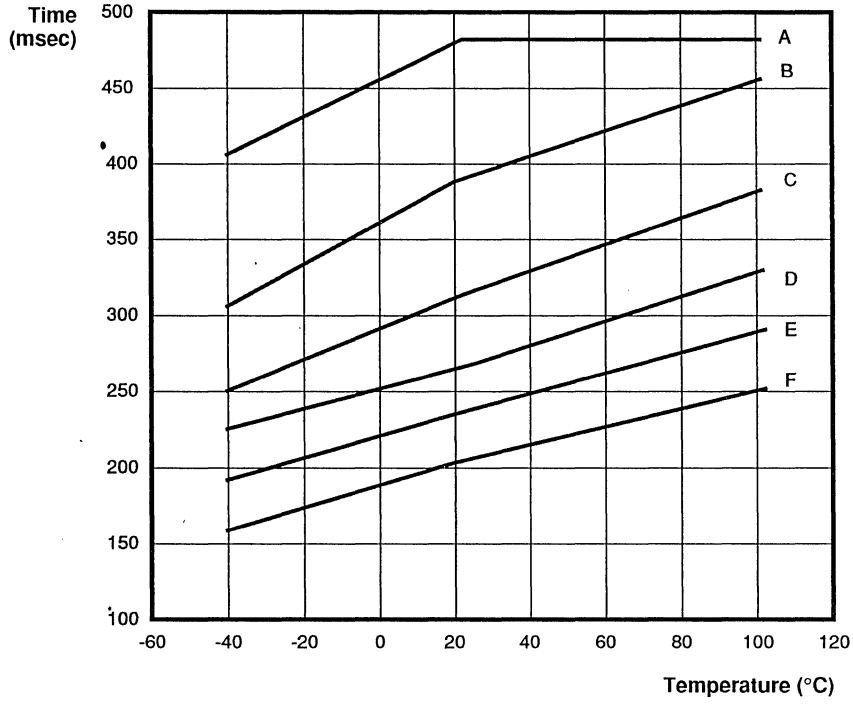
Legend:	
A - Vcc = 3.0V	D - Vcc = 4.5V
B - Vcc = 3.5V	E - Vcc = 5.0V
C - Vcc = 4.0V	F - Vcc = 5.5V

Figure 56. Typical 15 ms WDT Setting vs Temperature



Legend:	
A - Vcc = 3.0V	D - Vcc = 4.5V
B - Vcc = 3.5V	E - Vcc = 5.0V
C - Vcc = 4.0V	F - Vcc = 5.5V

Figure 57. Typical 25 ms WDT Setting vs Temperature



Legend:	
A - Vcc = 3.0V	D - Vcc = 4.5V
B - Vcc = 3.5V	E - Vcc = 5.0V
C - Vcc = 4.0V	F - Vcc = 5.5V

Figure 58. Typical 100 ms WDT Setting vs Temperature

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

Symbol	Meaning
IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

Symbol	Meaning
dst	Destination location or contents
src	Source location or contents
cc	Condition code
@	Indirect address prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag register (Control Register 252)
RP	Register Pointer (R253)
IMR	Interrupt mask register (R251)

Flags. Control register (R252) contains the following six flags:

Symbol	Meaning
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

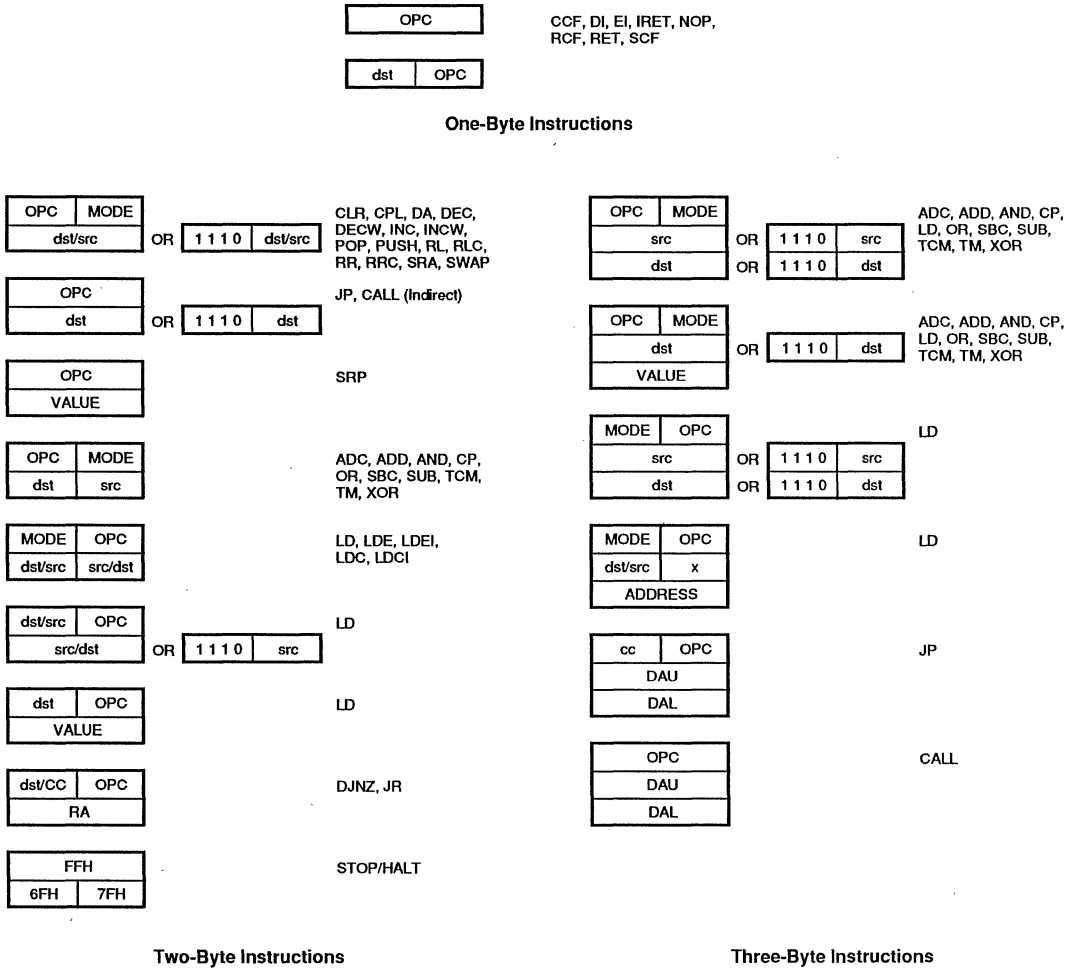
Affected flages are indicated by:

0	Clear to zero
1	Set to one
*	Set to clear according to operation
-	Unaffected
x	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000		Always True	
0111	C	Carry	C = 1
1111	NC	No Carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not Zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No Overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not Equal	Z = 0
1001	GE	Greater Than or Equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater Than	[Z OR (S XOR V)] = 0
0010	LE	Less Than or Equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned Greater Than or Equal	C = 0
0111	ULT	Unsigned Less Than	C = 1
1011	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned Less Than or Equal	(C OR Z) = 1
0000		Never True	

INSTRUCTION FORMATS



INSTRUCTION SUMMARY

Note: Assignment of a value is indicated by the symbol " ← ". For example:

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr (n)" is used to refer to bit (n) of a given operand location. For example:

dst (7)

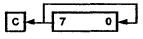
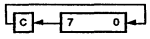
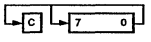
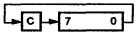
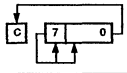
refers to bit 7 of the destination operand.

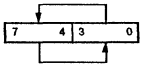
INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst, src dst←dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst←dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst←dst AND src	†		5[ ]	-	*	*	0	-	-
CALL dst SP←SP - 2 @SP←PC, PC←dst	DA IRR		D6 D4	-	-	-	-	-	-
CCF C←NOT C			EF	*	-	-	-	-	-
CLR dst dst←0	R IR		B0 B1	-	-	-	-	-	-
COM dst dst←NOT dst	R IR		60 61	-	*	*	0	-	-
CP dst, src dst - src	†		A[ ]	*	*	*	*	-	-
DA dst dst←DA dst	R IR		40 41	*	*	*	X	-	-
DEC dst dst←dst - 1	R IR		00 01	-	*	*	*	-	-
DECW dst dst←dst - 1	RR IR		80 81	-	*	*	*	-	-
DI IMR(7)←0			8F	-	-	-	-	-	-
DJNZr, dst r←r - 1 if r ≠ 0 PC←PC + dst Range: +127, -128	RA		rA r = 0 - F	-	-	-	-	-	-
EI IMR(7)←1			9F	-	-	-	-	-	-
HALT			7F	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
INC dst dst←dst + 1	r R IR		rE r = 0 - F 20 21	-	*	*	*	-	-
INCW dst dst←dst + 1	RR IR		A0 A1	-	*	*	*	-	-
IRET FLAGS←@SP; SP←SP + 1 PC←@SP; SP←SP + 2; IMR(7)←1			BF	*	*	*	*	*	*
JP cc, dst if cc is true PC←dst	DA IRR		cD c = 0 - F 30	-	-	-	-	-	-
JR cc, dst if cc is true, PC←PC + dst Range: +127, -128	RA		cB c = 0 - F	-	-	-	-	-	-
LD dst, src dst←src	r r R r X r r Ir R R R IR IR	Im R r X r Ir r R IR IM IM R	rC r8 r9 r = 0 - F C7 D7 E3 F3 E4 E5 E6 E7 F5	-	-	-	-	-	-
LDC dst, src	r	Irr	C2	-	-	-	-	-	-
LDCI dst, src dst←src r←r +1; rr←rr + 1	Ir	Irr	C3	-	-	-	-	-	-

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
NOP			FF	-	-	-	-	-	-
OR dst, src dst←dst OR src	†		4[ ]	-	*	*	0	-	-
POP dst dst←@SP; SP←SP + 1	R IR		50 51	-	-	-	-	-	-
PUSH src SP←SP - 1; @SP←src		R IR	70 71	-	-	-	-	-	-
RCF C←0			CF	0	-	-	-	-	-
RET PC←@SP; SP←SP + 2			AF	-	-	-	-	-	-
RL dst 	R IR		90 91	*	*	*	*	-	-
RLC dst 	R IR		10 11	*	*	*	*	-	-
RR dst 	R IR		E0 E1	*	*	*	*	-	-
RRC dst 	R IR		C0 C1	*	*	*	*	-	-
SBC dst, src dst←dst←src←C	†		3[ ]	*	*	*	*	1	*
SCF C←1			DF	1	-	-	-	-	-
SRA dst 	R IR		D0 D1	*	*	*	0	-	-
SRP src RP←src		Im	31	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
STOP			6F	-	-	-	-	-	-
SUB dst, src dst←dst←src	†		2[ ]	*	*	*	*	1	*
SWAP dst 	R IR		F0 F1	X	*	*	X	-	-
TCM dst, src (NOT dst) AND src	†		6[ ]	-	*	*	0	-	-
TM dst, src dst AND src	†		7[ ]	-	*	*	0	-	-
XOR dst, src dst←dst XOR src	†		B[ ]	-	*	*	0	-	-

† These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

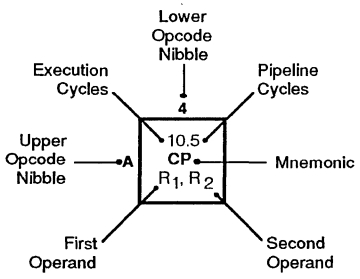
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address Mode dst	Address Mode src	Lower Opcode Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

OPCODE MAP

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								6.0 **STOP**
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								7.0 **HALT**
8	10.5 **DECW** RR1	10.5 **DECW** IR1	12.0 **LDE** r1, Irr2	18.0 **LDEI** Ir1, Irr2												6.1 **DI**
9	6.5 **RL** R1	6.5 **RL** IR1	12.0 **LDE** r2, Irr1	18.0 **LDEI** Ir2, Irr1												6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14.0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2				10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1	12.0 **LDC** r2, Irr1	18.0 **LDCI** Ir2, Irr1	20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6.5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2			3					2				3		1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_2 = Dst address
R_1 or r_2 = Src address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank are not defined.

* 2-byte instruction appears as a 3-byte instruction

PRODUCT SPECIFICATION

Z86E40

CMOS Z8® OTP CCP™ MICROCONTROLLER

FEATURES

- 8-bit CMOS microcontroller
- 40- or 44-pin package
- Low cost
- 4.0 to 5.5 volt operating range
- Software programmable low EMI mode.
- Pull-up Active/Open Drain programmable on Port 0, Port 1 and Port 2
- EPROM protect programmable
- RAM protect programmable
- RC oscillator programmable
- Low power consumption - 60 mW
- Fast instruction pointer - 0.6 microseconds
- Two standby modes - STOP and HALT
- 32 input/output lines
- Digital inputs CMOS levels, Schmitt triggered
- 4 Kbytes of one-time PROM
- 236 bytes of RAM
- Three Expanded Register File control Registers
- Two programmable 8-bit Counter/Timers each with a 6-bit programmable prescaler
- Six vectored, priority interrupts from six different sources
- Clock speeds up to 12 MHz
- Watchdog Timer
- Auto Power On Reset
- Two Comparators
- On-chip oscillator that accepts a crystal, ceramic resonator, LC, RC or external clock drive.

GENERAL DESCRIPTION

The Z86E40 Consumer Controller Processor (CCP) introduces a new level of sophistication to single-chip architecture. The Z86E40 is a member of the Z8 single-chip microcontroller family with 4 Kbytes of EPROM and 236 bytes of RAM. The device is housed in a 40-pin DIP, 44-pin Plastic Leaded Chip Carrier, and a 44-pin Quad Flat Pack, and is manufactured in CMOS technology. The device offers easy software development and debug, prototyping, and small production runs not economically desirable with a masked ROM version.

The Z86E40 architecture is characterized by Zilog's 8-bit microcontroller core with an expanded register file to allow easy access to register mapped peripheral and I/O circuits. The CCP offers a flexible I/O scheme, an efficient register and address space structure, and a number of ancillary features that are useful in many industrial, consumer, automotive, peripheral types, and advanced scientific applications.

The device applications demand powerful I/O capabilities. The CCP fulfills this with 32-pins dedicated to input and output. These lines are grouped into four ports, eight lines per port, and are configurable under software control to provide timing, status signals, and parallel I/O with or without handshake, and address/data bus for interfacing external memory.

GENERAL DESCRIPTION (Continued)

There are four basic address spaces available to support this wide range of configurations: Program Memory, Data Memory, Register File, and Expanded Register File (ERF). The Register File is composed of 236 bytes of general-purpose registers, four I/O port registers and 15 control and status registers. The Expanded Register File consists of three control resisters.

To unburden the program from coping with the real-time problems such as counting/timing and input/output data communication, the Z86E40 offers two on-chip counter/timers with a large number of user selectable modes, and two on-board comparators to process analog signals with a common reference voltage (Figures1 and 2).

Note: All Signals with a preceding front slash, "/", are active Low, e.g.: B//W (WORD is active Low); /B/W (BYTE is active Low, only).

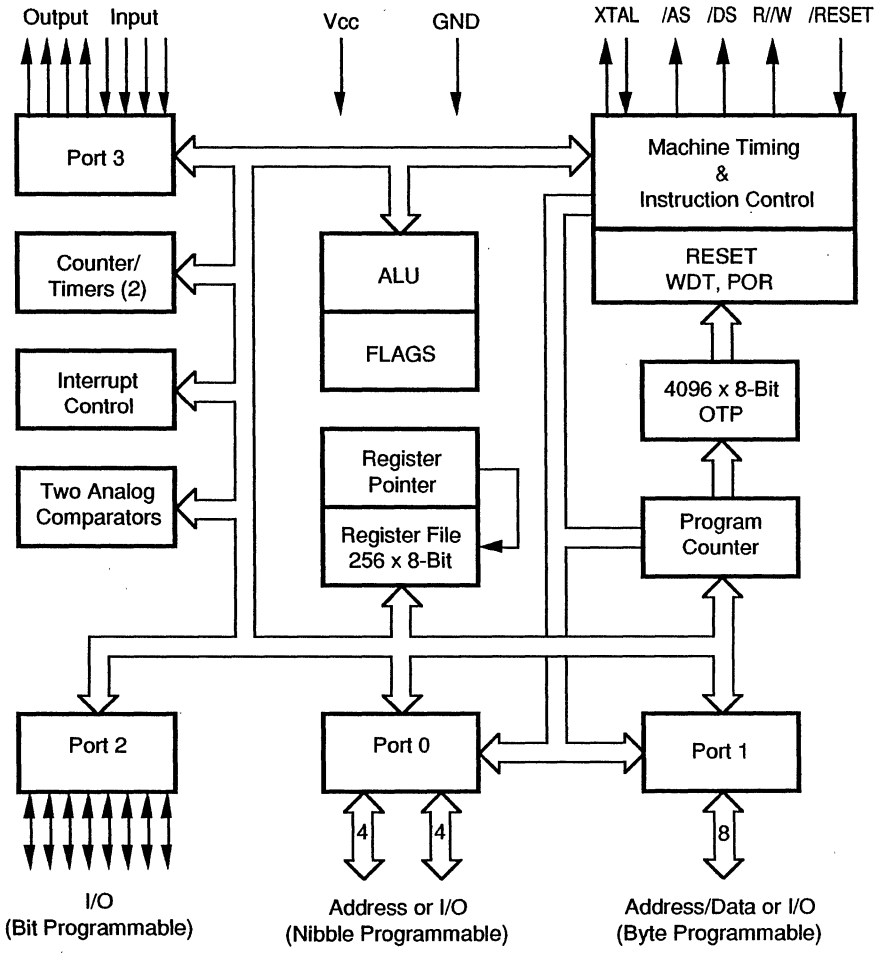


Figure 1. Functional Block Diagram

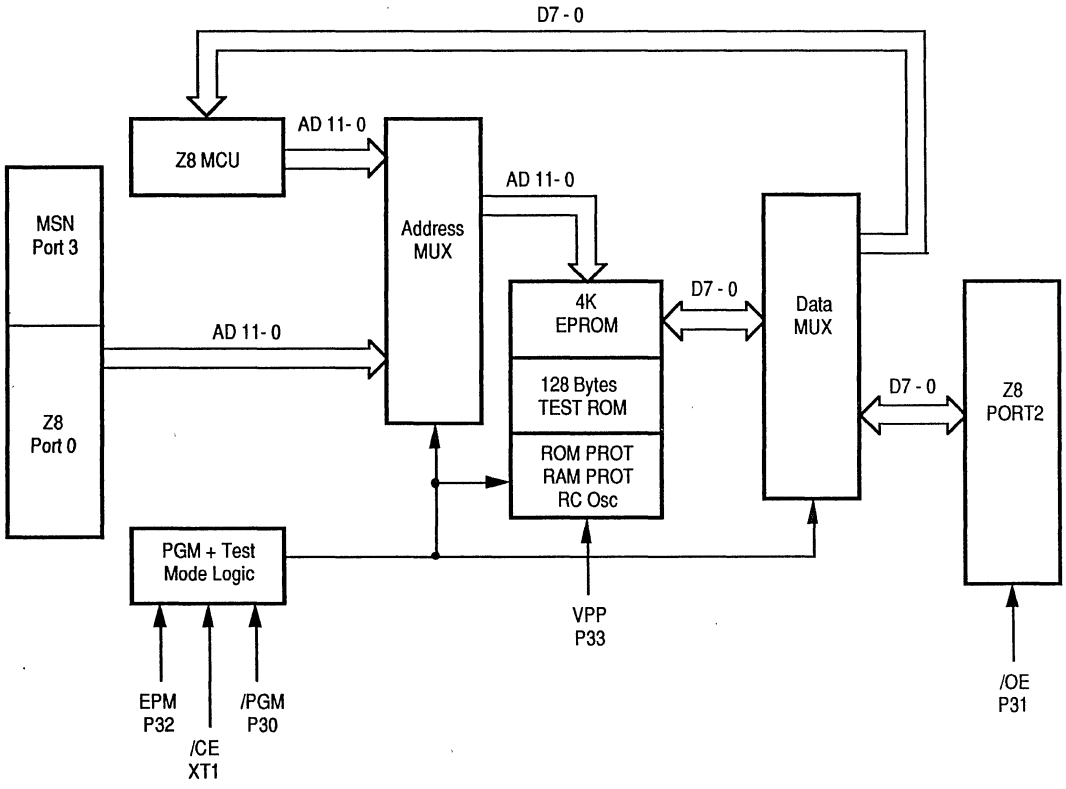


Figure 2. EPROM Programming Block Diagram

PIN IDENTIFICATION

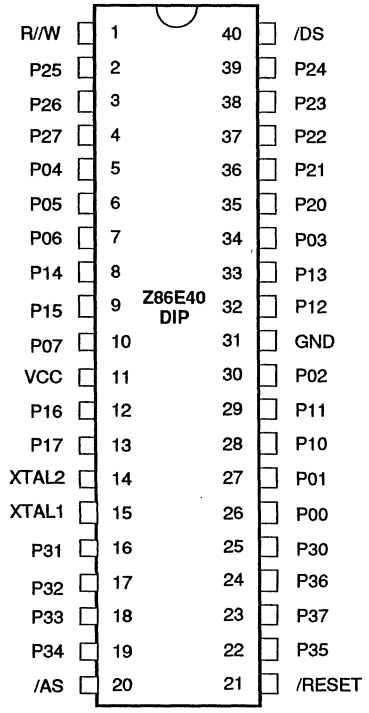


Figure 3. 40-Pin Dual In-Line, Pin Assignments (Standard Mode)

Table 1. 40-Pin Dual In-Line Package Pin Identification (Standard Mode)

Pin #	Symbol	Function	Direction
1	R//W	Read/Write	Output
2-4	P25-7	Port 2 pin 5,6,7	In/Output
5-7	P04-6	Port 0 pin 4,5,6	In/Output
8-9	P14-5	Port 1 pin 4,5	In/Output
10	P07	Port 0 pin 7	In/Output
11	V_{cc}	Power Supply	Input
12-13	P16-7	Port 1 pin 6,7	In/Output
14	XTAL2	Crystal Oscillator	Output
15	XTAL1	Crystal Oscillator	Input
16-18	P31-3	Port 3 pin 1,2,3	Input
19	P34	Port 3 pin 4	Output
20	/AS	Address Strobe	Output
21	/RESET	Reset	Input
22	P35	Port 3 pin 5	Output
23	P37	Port 3 pin 7	Output
24	P36	Port 3 pin 6	Output
25	P30	Port 3 pin 0	Input
26-27	P00-1	Port 0 pin 0,1	In/Output
28-29	P10-1	Port 1 pin 0,1	In/Output
30	P02	Port 0 pin 2	In/Output
31	GND	Ground, V_{ss}	Input
32-33	P12-3	Port 1 pin 2,3	In/Output
34	P03	Port 0 pin 3	In/Output
35-39	P20-4	Port 2 pin 0,1,2,3,4	In/Output
40	/DS	Data Strobe	Output

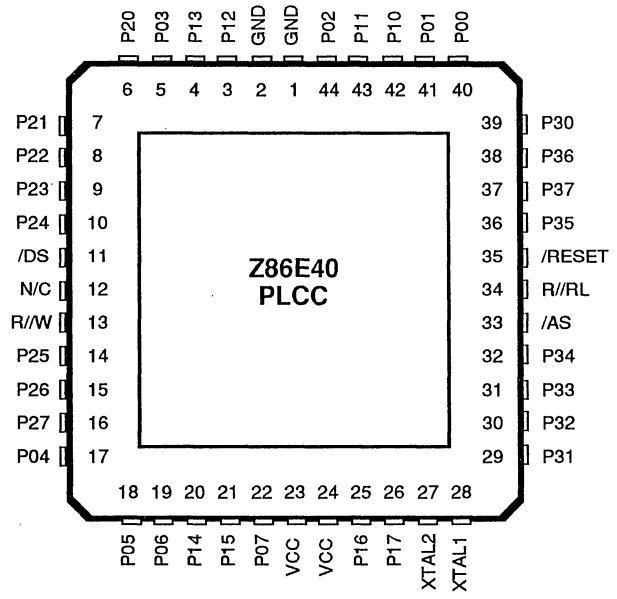


Figure 4. 44-Pin PLCC Pin Assignments (Standard Mode)

Table 2. 44-Pin Plastic Leaded Chip Carrier Pin Identification (Standard Mode)

Pin #	Symbol	Function	Direction
1-2	GND	Ground , V_{SS}	Input
3-4	P12-3	Port 1 pin 2,3	In/Output
5	P03	Port 0 pin 3	In/Output
6-10	P20-4	Port 2 pin 0,1,2,3,4	In/Output
11	/DS	Data Strobe	Output
12	N/C	No Connection	
13	R//W	Read/Write	Output
14-16	P25-7	Port 2 pin 5,6,7	In/Output
17-19	P04-6	Port 0 pin 4,5,6	In/Output
20-21	P14-5	Port 1 pin 4,5	In/Output
22	P07	Port 0 pin 7	In/Output
23-24	V_{CC}	Power Supply	Input
25-26	P16-7	Port 1 pin 6,7	In/Output
27	XTAL2	Crystal Oscillator	Output
28	XTAL1	Crystal Oscillator	Input
29-31	P31-3	Port 3 pin 1,2,3	Input
32	P34	Port 3 pin 4	Output
33	/AS	Address Strobe	Output
34	R//RL	ROM/ROMless select	Input
35	/RESET	Reset	Input
36	P35	Port 3 pin 5	Output
37	P37	Port 3 pin 7	Output
38	P36	Port 3 pin 6	Output
39	P30	Port 3 pin 0	Input
40-41	P00-1	Port 0 pin 0,1	In/Output
42-43	P10-1	Port 1 pin 0,1	In/Output
44	P02	Port 0 pin 2	In/Output

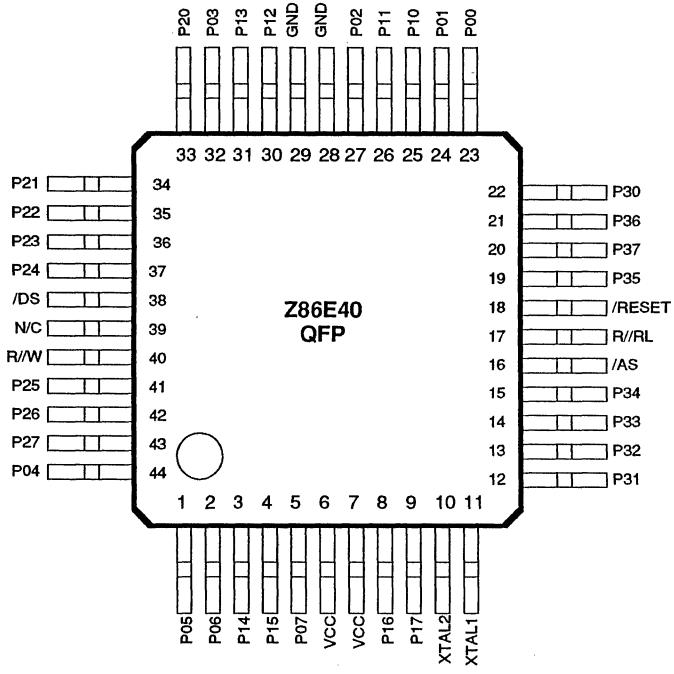


Figure 5. 44-Pin QFP Pin Assignments (Standard Mode)

Table 3. 44-Pin Quad Flat Pack Pin Identification (Standard Mode)

Pin #	Symbol	Function	Direction
1-2	P05-6	Port 0 pin 5,6	In/Output
3-4	P14-5	Port 1 pin 4,5	In/Output
5	P07	Port 0 pin 7	In/Output
6-7	V_{cc}	Power Supply	Input
8-9	P16-7	Port 1 pin 6,7	In/Output
10	XTAL2	Crystal Oscillator	Output
11	XTAL1	Crystal Oscillator	Input
12-14	P31-3	Port 3 pin 1,2,3	Input
15	P34	Port 3 pin 4	Output
16	/AS	Address Strobe	Output
17	R//RL	ROM/ROMless select	Input
18	/RESET	Reset	Input
19	P35	Port 3 pin 5	Output
20	P37	Port 3 pin 7	Output

Pin #	Symbol	Function	Direction
21	P36	Port 3 pin 6	Output
22	P30	Port 3 pin 0	Input
23-24	P00-1	Port 0 pin 0,1	In/Output
25-26	P10-1	Port 1 pin 0,1	In/Output
27	P02	Port 0 pin 2	In/Output
28-29	GND	Ground, V_{ss}	Input
30-31	P12-3	Port 1 pin 2,3	In/Output
32	P03	Port 0 pin 3	In/Output
33-37	P20-4	Port 2 pin 0,1,2,3,4	In/Output
38	/DS	Data Strobe	Output
39	N/C	No Connection	
40	R//W	Read/Write	Output
41-43	P25-7	Port 2 pin 5,6,7	In/Output
44	P04	Port 0 pin 4	In/Output

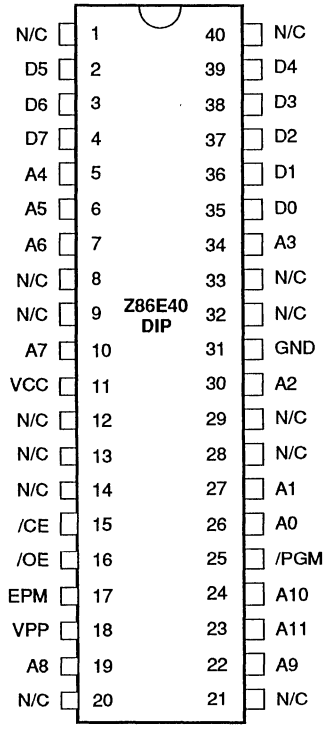


Figure 6. 40-Pin DIP Pin Assignments (EPROM Programming Mode)

Table 4. 40-Pin Dual In-Line Package Pin Identification (EPROM Programming Mode)

Pin #	Symbol	Function	Direction
1	N/C	No Connection	
2-4	D5-7	Data 5,6,7	In/Output
5-7	A4-6	Address 4,5,6	Input
8-9	N/C	No Connection	
10	A7	Address 7	Input
11	V_{CC}	Power Supply	Input
12-14	N/C	No Connection	
15	/CE	Chip Select	Input
16	/OE	Output Enable	Input
17	EPM	EPROM Prog. Mode	Input
18	V_{PP}	Prog. Voltage	Input
19	A8	Address 8	Input
20-21	N/C	No Connection	
22	A9	Address 9	Input
23	A11	Address 11	Input
24	A10	Address 10	Input
25	/PGM	Prog. Mode	Input
26-27	A0-1	Address 0,1	Input
28-29	N/C	No connection	
30	A2	Address 2	Input
31	GND	Ground	Input
32-33	N/C	No connection	
34	A3	Address 3	Input
35-39	D0-4	Data 0,1,2,3,4	In/Output
40	N/C	No connection	

PIN IDENTIFICATION (Continued)

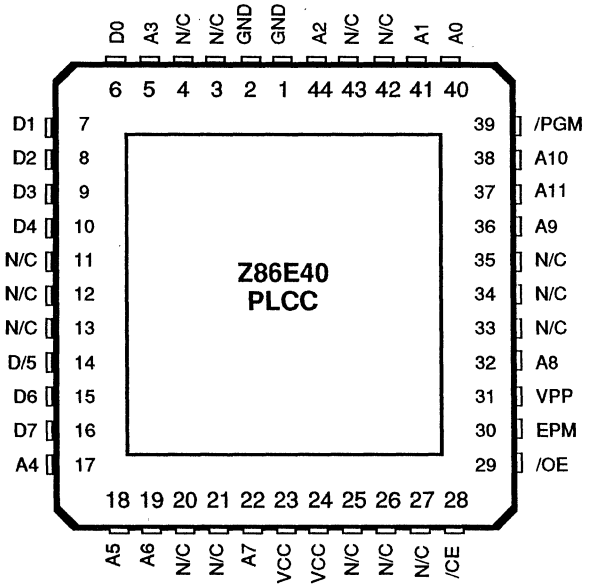


Figure 7. 44-Pin PLCC Pin Assignments (EPROM programming mode)

Table 5. 44-Pin Leaded Chip Carrier Pin Identification (EPROM Programming Mode)

Pin #	Symbol	Function	Direction
1	N/C	No Connection	
2-4	D5-7	Data 5,6,7	In/Output
5-7	A4-6	Address 4,5,6	Input
8-9	N/C	No Connection	
10	A7	Address 7	Input
11	V_{CC}	Power Supply	Input
12-14	N/C	No Connection	
15	/CE	Chip Select	Input
16	/OE	Output Enable	Input
17	EPM	EPROM Prog. Mode	Input
18	V_{PP}	Prog. Voltage	Input
19	A8	Address 8	Input
20-21	N/C	No Connection	
22	A9	Address 9	Input
23	A11	Address 11	Input
24	A10	Address 10	Input
25	/PGM	Prog. Mode	Input
26-27	A0-1	Address 0,1	Input
28-29	N/C	No connection	
30	A2	Address 2	Input
31	GND	Ground	Input
32-33	N/C	No connection	
34	A3	Address 3	Input
35-39	D0-4	Data 0,1,2,3,4	In/Output
40	N/C	No connection	

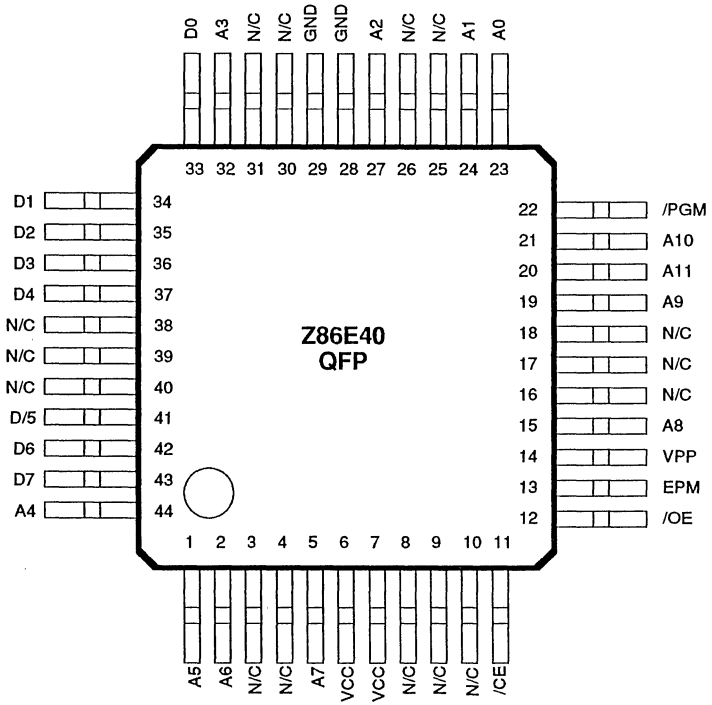


Figure 8. 44-Pin QFP Pin Assignments (EPROM programming mode)

Table 6. 44-pin Quad Flat Pack (EPROM Programming Mode)

Pin #	Symbol	Function	Direction
1	N/C	No Connection	
2-4	D5-7	Data 5,6,7	In/Output
5-7	A4-6	Address 4,5,6	Input
8-9	N/C	No Connection	
10	A7	Address 7	Input
11	V_{CC}	Power Supply	Input
12-14	N/C	No Connection	
15	/CE	Chip Select	Input
16	/OE	Output Enable	Input
17	EPM	EPROM Prog. Mode	Input
18	V_{PP}	Prog. Voltage	Input
19	A8	Address 8	Input
20-21	N/C	No Connection	
22	A9	Address 9	Input
23	A11	Address 11	Input
24	A10	Address 10	Input
25	/PGM	Prog. Mode	Input
26-27	A0-1	Address 0,1	Input
28-29	N/C	No connection	
30	A2	Address 2	Input
31	GND	Ground	Input
32-33	N/C	No connection	
34	A3	Address 3	Input
35-39	D0-4	Data 0,1,2,3,4	In/Output
40	N/C	No connection	

PIN FUNCTIONS

EPROM Programming Mode

D7-D0. *Data Bus.* The data can be read from or written to external memory through the data bus.

A11-A0. *Address Bus.* During programming, the EPROM address is written to the address bus.

V_{CC}. *Power Supply.* This pin has to supply 5V during the EPROM read mode and 6V during other modes.

/CE. *Chip Enable* (active Low). This pin is active during EPROM Read Mode, Program Mode and Program Verify Mode.

/OE. *Output Enable* (active Low). This pin drives the direction of the Data Bus. When this pin is low, the Data Bus is output. When high, the Data Bus is input.

EPM. *EPROM Program Mode.* This pin controls the different EPROM Program Mode by applying different voltages.

V_{PP}. *Program Voltage.* This pin supplies the program voltage.

/PGM. *Program Mode* (active Low). When this pin is low, the data is programmed to the EPROM through the Data Bus.

Z86E40 Standard Mode

XTAL. *Crystal 1* (time-based input). This pin connects a parallel-resonant crystal, ceramic resonator, LC, RC network or external single-phase clock to the on-chip oscillator input.

XTAL2. *Crystal 2* (time-based output). This pin connects a parallel-resonant crystal, ceramic resonator, LC or RC network to the on-chip oscillator output.

R/W. *Read/Write* (output, write Low). The R/W signal is low when the CCP is writing to the external program or data memory.

/RESET. *Reset* (input, active-Low). Reset will initialize the MCU. Reset is accomplished either through Power-On, Watch Dog Timer reset, STOP Mode Recovery, or external reset. During Power-On Reset and Watch Dog Timer Reset, the internally generated reset drives the reset pin low for the POR time. Any devices driving the reset line must be open-drain in order to avoid damage from a possible conflict during reset conditions. Pull-up is provided internally. After the POR time, /RESET is a Schmitt triggered input.

To avoid asynchronous and noisy reset problems, the Z86E40 is equipped with a reset filter of four external clocks (4TpC). If the external reset signal is less than 4TpC in duration, no reset occurs. On the fifth clock after the reset is detected, an internal RST signal is latched and held for an internal register count of 18 external clocks, or for the duration of the external reset, whichever is longer. During the reset cycle, /DS is held active low while /AS cycles at a rate of TpC/2. Program execution begins at location 000CH, 5-10 TpC cycles after /RESET is released. For Power-On Reset, the reset output time is 5 ms. The Z86E40 does not reset WDTMR, SMR, P2M, and P3M registers on a STOP Mode Recovery operation.

Port 0 (P00-P07). Port 0 is an 8-bit, bidirectional, CMOS compatible I/O port. These eight I/O lines can be configured under software control as a nibble I/O port, or as an address port for interfacing external memory. The input buffers are Schmitt triggered and nibble programmed. Either nibble output that can be globally programmed as push-pull or open drain. Low EMI output buffers can be globally programmed by the software. Port 0 can be placed under handshake control. In Handshake Mode, Port 3 lines P32 and P35 are used as handshake control lines. The handshake direction is determined by the configuration (input or output) assigned to Port 0's upper nibble. The lower nibble must have the same direction as the upper nibble.

For external memory references, Port 0 provides address bits A11-A8 (lower nibble) or A15-A8 (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. If one or both nibbles are needed for I/O operation, they must be configured by writing to the Port 0 mode register. In ROMless mode, after a hardware reset, Port 0 is configured as address lines A15-A8, and extended timing is set to accommodate slow memory access. The initialization routine can include reconfiguration to eliminate this extended timing mode. In ROM mode, Port 0 is defined as input after reset.

Port 0 can be set in the high-impedance mode if selected as an address output state, along with Port 1 and the control signals /AS, /DS and R/W (Figure 9).

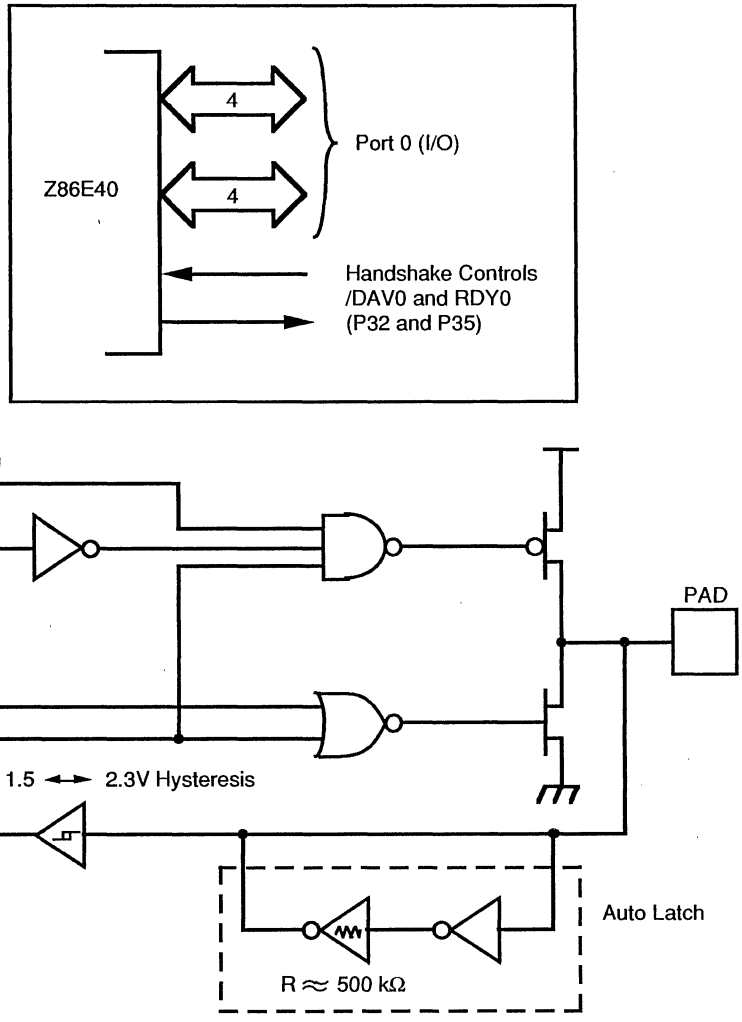


Figure 9. Port 0 Configuration

PIN FUNCTIONS (Continued)

Port 1 (P10-P17). Port 1 is an 8-bit, bidirectional, CMOS compatible port. It has multiplexed Address (A7-A0) and Data (D7-D0) ports. These eight I/O lines can be programmed as inputs or outputs or can be configured under software control as an Address/Data port for interfacing external memory. The input buffers are Schmitt triggered and the output buffers can be globally programmed as either push-pull or open drain. Low EMI output buffers can be globally programmed by the software. Port 1 can be placed under handshake control. In this configuration, Port 3, lines P33 and P34 are used as the handshake controls RDY1 and /DAV1 (Ready and Data Available). To interface external memory, Port 1 must be programmed for the multiplexed Address/Data mode. If more than 256 external locations are required, Port 0 outputs the additional lines (Figure 10).

Port 1 can be placed in the high-impedance state along with Port 0, /AS, /DS and R/W, allowing the Z86E40 to share common resources in multiprocessor and DMA applications.

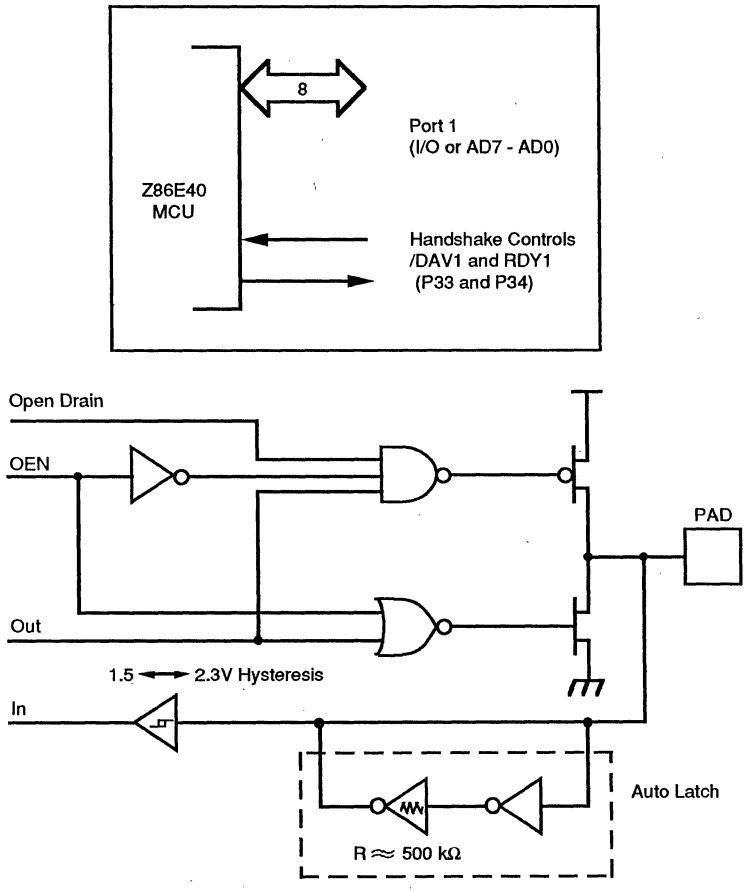


Figure 10. Port 1 Configuration

Port 2 (P20-P27). Port 2 is an 8-bit, bidirectional, CMOS compatible I/O port. These eight I/O lines can be configured under software control as an input or output, independently. All input buffers are Schmitt triggered. Bits programmed as outputs can be globally programmed as either push-pull or open drain. Low EMI output buffers can be globally programmed by the software. When used as an I/O port, Port 2 can be placed under handshake control.

In Handshake Mode, Port 3 lines P31 and P36 are used as handshake control lines. The handshake direction is determined by the configuration (input or output) assigned to bit 7 of Port 2 (Figure 11).

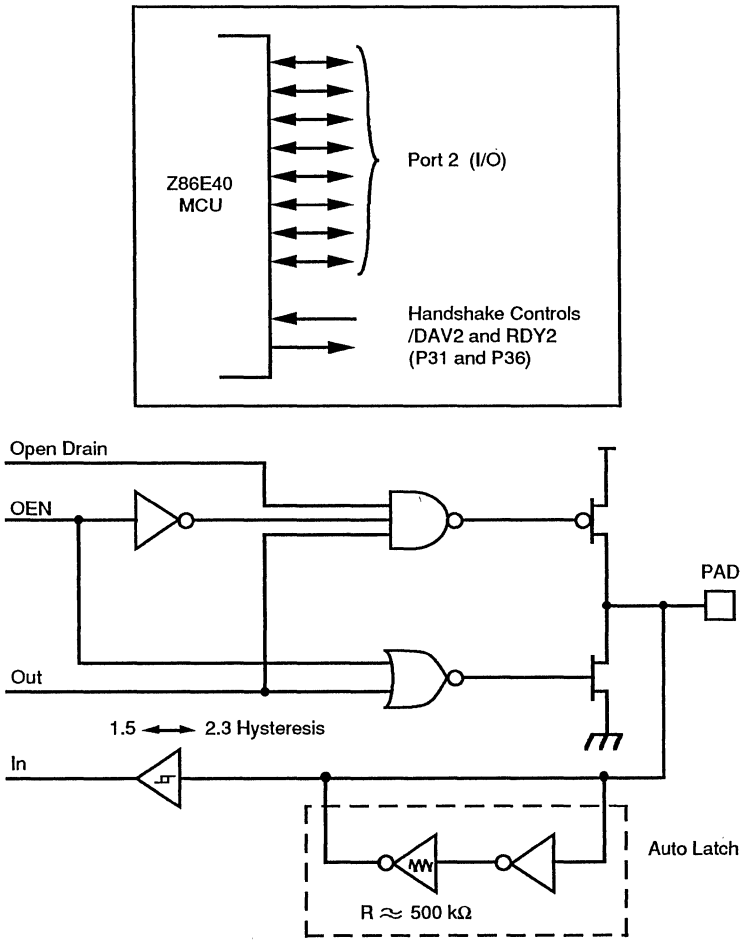


Figure 11. Port 2 Configuration

Port 3 (P30-P37). Port 3 is an 8-bit, CMOS compatible port with four fixed input and four fixed output. Port 3 consists of four fixed input (P30-P33) and four fixed output (P34-P37), and can be configured by software for interrupt and handshake control functions. Port 3, Pin 0 is Schmitt triggered. Pins P31, P32 and P33 are standard CMOS inputs (no Auto-Latches) and Pins P34, P35, P36 and P37 are push-pull output lines. Low EMI output buffers can be globally programmed by the software. Two on-board comparators can process analog signals on P31 and P32 with reference to the voltage on P33. The analog function is enabled by setting the D1 of Port 3 Mode Register (P3M).

For the interrupt function, P30 and P33 are falling edge triggered interrupt inputs. P31 and P32 can be programmed as falling, rising or both edges triggered interrupt inputs (Figure 12). Access to Counter/Timer 1 is made through P31 (T_{IN}) and P36 (T_{OUT}). Handshake lines for Port 0, Port 1, and Port 2 are also available on Port 3 (Table 7).

Note: P30-P33 differs from the Z86C40 because there is no clamping diode to V_{CC} due to the EPROM high voltage circuits. Exceeding the V_{IH} maximum specification during standard operating mode may cause the device to enter EPROM mode.

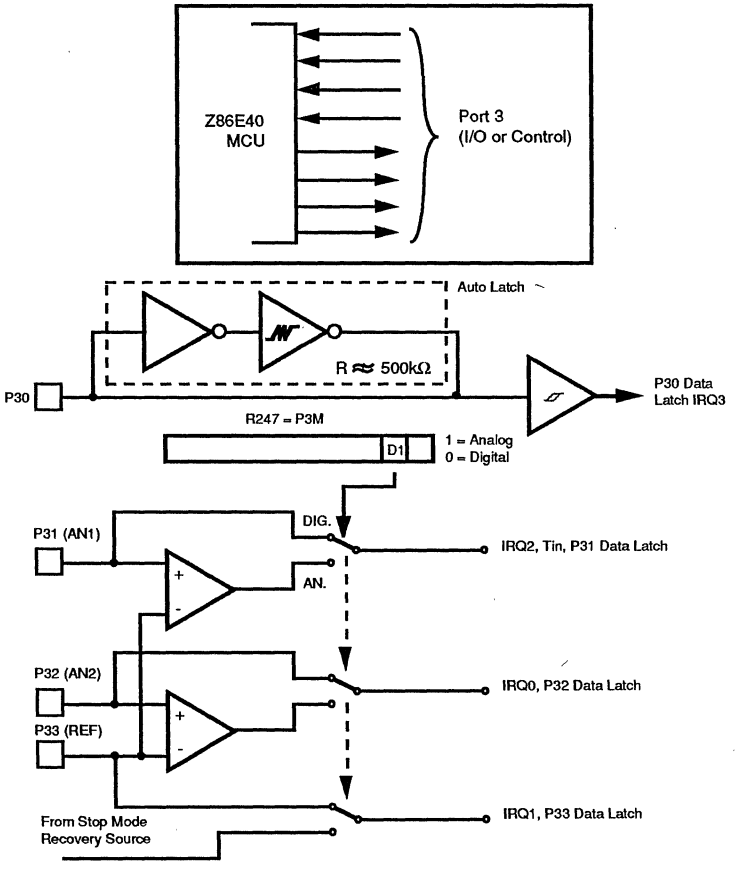


Figure 12. Port 3 Configuration

Table 7. Pin Assignments of Port 3

Pin	I/O	CTC1	AN IN	Interrupt	P0 HS	P1 HS	P2 HS	Ext
P30	IN			IRQ3				
P31	IN	T_{IN}	AN1	IRQ2		D/R		
P32	IN		AN2	IRQ0	D/R			
P33	IN		REF	IRQ1		D/R		
P34	OUT					R/D		DM
P35	OUT				R/D			
P36	OUT	T_{OUT}				R/D		
P37	OUT							

Comparator Inputs. Port 3 pins P31 and P32 each has a comparator front end. The comparator reference voltage Pin P33 is common to both comparators. In analog mode, P31 and P32 are the positive input of the comparators and P33 is the reference voltage of the comparators.

Auto-Latch. The Auto-Latch puts valid CMOS levels on all CMOS inputs (except P31-P33) that are not externally driven. Whether this level is zero or one, cannot be determined. A valid CMOS level, rather than a floating node, reduces excessive supply current flow in the input buffer. Auto-Latches are available on Port 0, Port 2 and P30. There are no Auto-Latches on P31, P32, and P33.

Low EMI Emission. The Z86E40 can be programmed to operate in a low EMI emission mode in the PCON register. The oscillator and all I/O ports can be programmed as low EMI emission mode independently. Use of this feature results in:

- Less than 1.5 mA typical consumption during HALT mode.
- The pre-drivers slew rate is reduced to 10 ns typical.
- Low EMI output drivers have resistance of 200 ohms (typical).
- Oscillator divide-by-two circuitry is eliminated.
- Internal SLCK/TCLK operation limited to a maximum of 4 MHz - 250 ns cycle time.

FUNCTIONAL DESCRIPTION

The Z8 CCP incorporates special functions to enhance the Z8's applications in industrial, scientific research, and advanced technologies applications.

RESET. The device is reset in one of the following conditions:

- Power-On Reset
- Watch-Dog Timer
- STOP Mode Recovery Source

Having the Auto Power-on Reset circuitry built-in, the Z86E40 does not need to be connected to an external power-on reset circuit. The reset time is 5 ms (typical) plus 18 clock cycles. The Z86E40 does not re-initialize WDTMR, SMR, P2M, and P3M registers to their reset values on a STOP Mode Recovery operation.

Program Memory. The Z86E40 can address up to 4 Kbytes of internal program memory (Figure 13). The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts. For EPROM mode, byte 12 (000CH) to address 4095 (0FFFH) consists of programmable EPROM. After reset, the program counter points at the address 000CH which is the starting address of the user program. In ROMless mode, the Z86E40 can address up to 64 Kbytes of external program memory. The ROM/ROMless option is only available on the 44-pin devices.

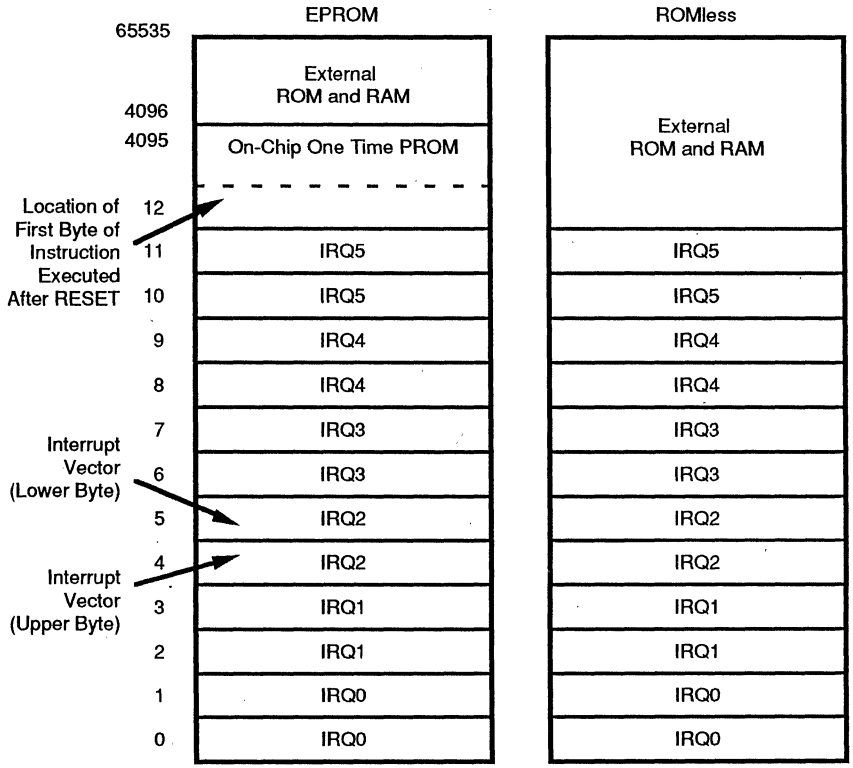


Figure 13. Program Memory Map

EPROM Protect. The 4 Kbytes of program memory is a one-time PROM. An EPROM protect feature prevents dumping of the ROM contents by inhibiting execution of LDC, LDCI, LDE, and LDEI instructions to program memory in all modes.

Data Memory (/DM). In EPROM mode, the Z86E40 can address up to 60 Kbytes of external data memory beginning at location 4096. In ROMless mode, the Z86E40 can address up to 64 Kbytes of data memory. External data memory may be included with, or separated from, the external program memory space. /DM, an optional I/O function that can be programmed to appear on pin P34, is used to distinguish between data and program memory space (Figure 14). The state of the /DM signal is controlled by the type of instruction being executed. An LDC opcode references PROGRAM (/DM inactive) memory, and an LDE instruction references data (/DM active Low) memory.

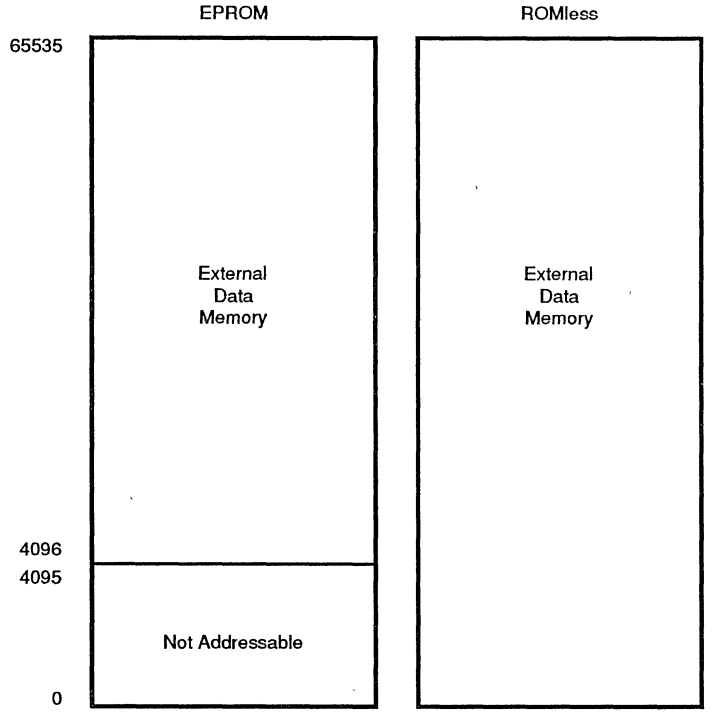


Figure 14. Data Memory Map

Expanded Register File. The register file has been expanded to allow for additional system control registers, mapping of additional peripheral devices and input/output ports into the register address area. The Z8 register address space R0 through R15 is implemented as 16 groups of 16 registers per group (Figure 15). These register groups are known as the Expanded Register File (ERF). The low nibble (D3-D0) of the Register Pointer (RP) select the active ERF group, and the high nibble (D7-D4) of register RP select the working register group (Figure 16). Three system configuration registers reside in the Expanded Register File at bank FH: PCON, SMR, and WDTMR. The rest of the Expanded Register is not physically implemented and is reserved for future expansion.

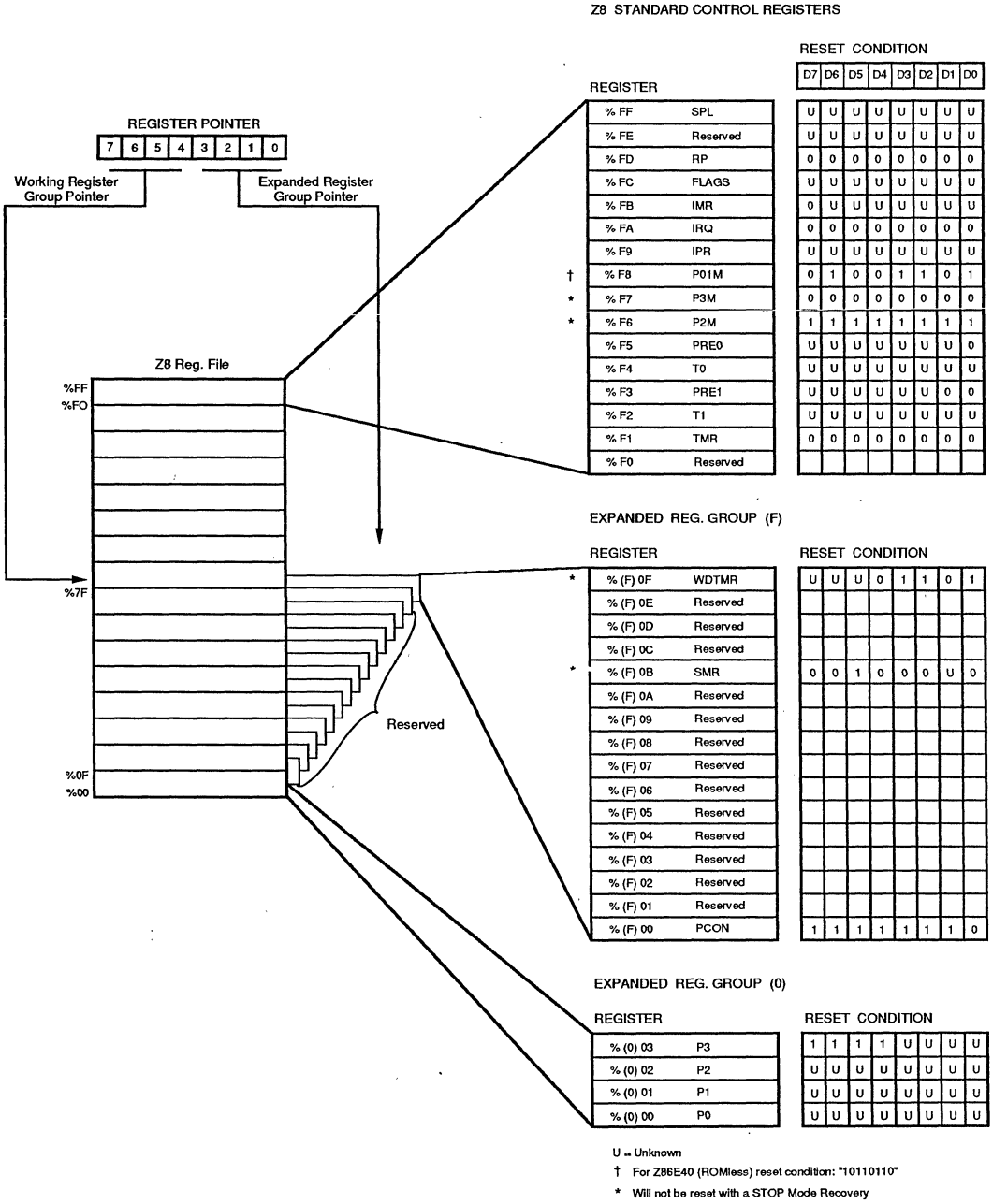


Figure 15. Expanded Register File Architecture

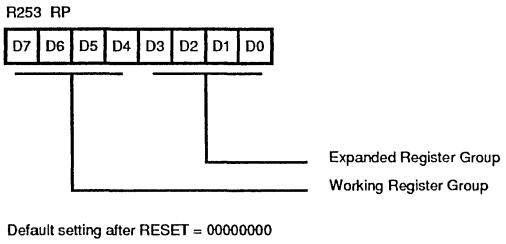


Figure 16. Register Pointer Register

Register File. The 256 byte register file consists of 4 I/O port registers, 236 general-purpose registers and 15 control and status registers (R240 is reserved). The instructions can access registers directly or indirectly via an 8-bit address field. This allows short 4-bit register addresses using the Register Pointer (Figures 16 and 17). In the 4-bit mode, the register file is divided into 16 working register groups, each occupying 16 continuous locations. The Register Pointer addresses the starting location of the active working-register group.

Note: Register Bank E0H-EFH can only be accessed through working registers and indirect addressing modes.

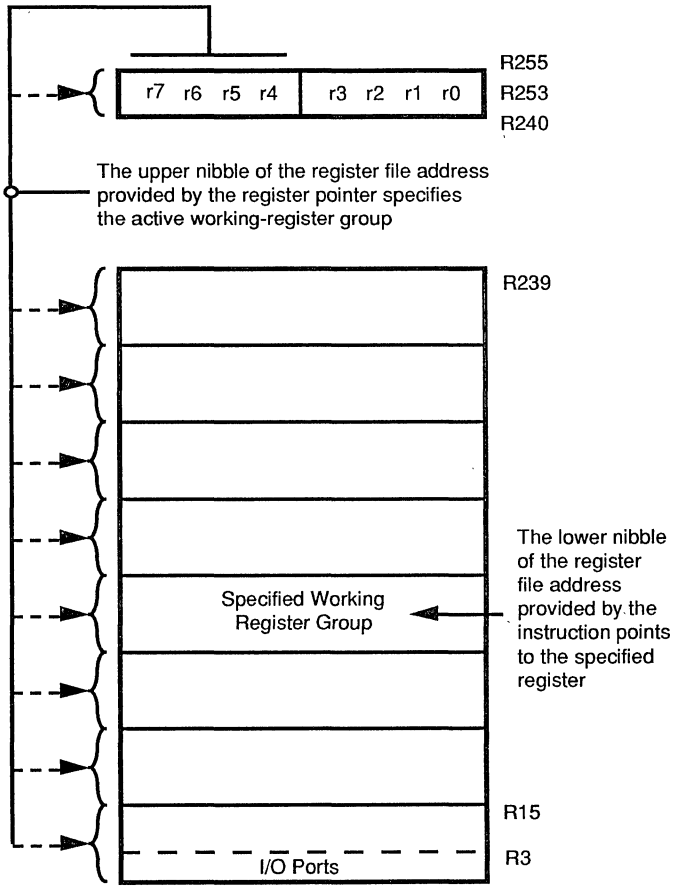


Figure 17. Register Pointer

RAM Protect. The upper portion of the RAM's address spaces 80H to EFH (excluding the control registers) can be protected from reading and writing. This option can be selected during the EPROM Programming Mode. After this option is selected, the user can activate this feature from the internal EPROM. D6 of the IMR control register (R251) is used to turn off/on the RAM protect by loading a 0 or 1, respectively. A 1 in D6 indicates RAM Protect enabled.

Stack. The Z86E40 external data memory or the internal register file can be used for the stack. The 16-bit Stack Pointer (R254-R255) is used for the external stack which can reside anywhere in the data memory for ROMless mode, but only from 4096 to 65535 in ROM mode. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 236 general purpose registers (R4-R239). SPH can be used as a general purpose register when using internal stack only.

Counter/Timers. There are two 8-bit programmable counter/timers (T0 and T1), each driven by its own 6-bit programmable prescaler. The T1 prescaler is driven by internal or external clock sources; however, the T0 prescaler is driven by the internal clock only (Figure 18).

The 6-bit prescalers can divide the input frequency of the clock source by any integer number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request, IRQ4 (T0) or IRQ5 (T1), is generated.

The counters can be programmed to start, stop, restart to continue, or restart from the initial value. The counters can also be programmed to stop upon reaching zero (single pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode).

The counters, but not the prescalers, can be read at any time without disturbing their value or count mode. The clock source for T1 is user-definable and can be either the internal microprocessor clock divided-by-four, or an external signal input via Port 3. The Timer Mode register configures the external timer input (P31) as an external clock, a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. Port 3 line P36 serves as a timer output (T_{OUT}) through which T0, T1 or the internal clock can be output. The counter/timers can be cascaded by connecting the T0 output to the input of T1.

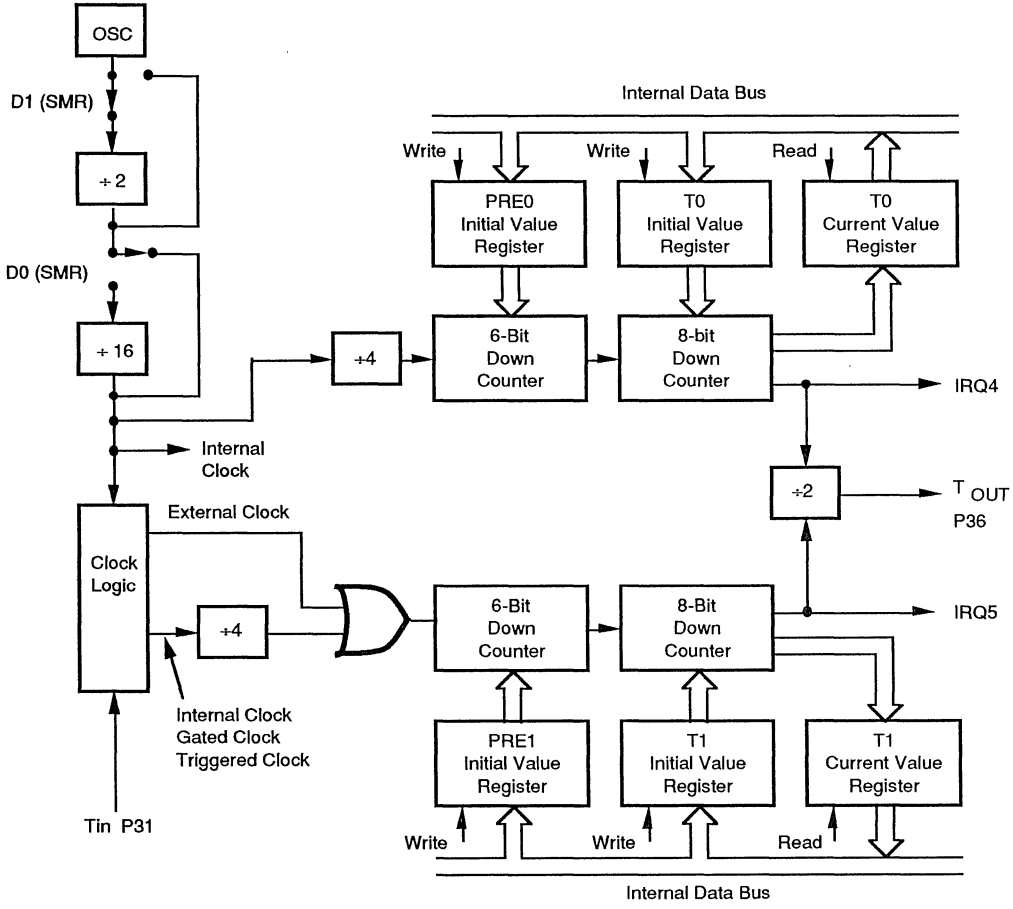


Figure 18. Counter/Timer Block Diagram

Interrupts. The Z86E40 has six different interrupts from six different sources. The interrupts are maskable and prioritized (Figure 19). The six sources are divided as follows: four sources are claimed by Port 3 lines P30-P33, and two in counter/timers. The Interrupt Mask Register globally or individually enables or disables the six interrupt requests (Table 8).

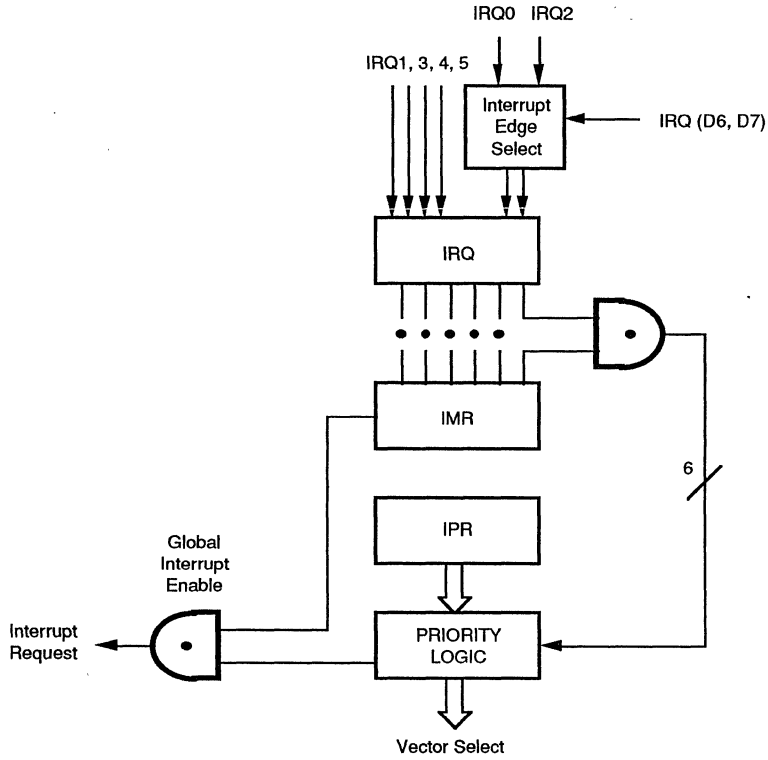


Figure 19. Interrupt Block Diagram

Table 8. Interrupt Types, Sources, and Vectors

Name	Source	Vector Location	Comments
IRQ 0	/DAV 0, IRQ 0	0, 1	External (P32), Rising/Falling Edge Triggered
IRQ 1	IRQ 1	2, 3	External (P33), Falling Edge Triggered
IRQ 2	/DAV 2, IRQ 2, T_{IN}	4, 5	External (P31), Rising/Falling Edge Triggered
IRQ 3	IRQ3	6, 7	External (P30), Falling Edge Triggered
IRQ 4	T0	8, 9	Internal
IRQ 5	TI	10, 11	Internal

When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority Register (IPR). An interrupt machine cycle is activated when an interrupt request is granted. Thus, disabling all subsequent interrupts, saves the Program Counter and Status Flags, and then branches to the program memory vector location reserved for that interrupt. All Z86E40 interrupts are vectored through locations in the program memory. This memory location and the next byte contain the 16-bit starting address of the interrupt service routine for that particular interrupt request.

To accommodate polled interrupt systems, interrupt inputs are masked and the interrupt request register is polled to determine which of the interrupt requests need service.

An interrupt resulting from AN1 is mapped into IRQ2, and an interrupt from AN2 is mapped into IRQ0. Interrupts IRQ2 and IRQ0 may be rising, falling or both edge triggered, and are programmable by the user. The software may poll to identify the state of the pin.

Programming bits for the Interrupt Edge Select are located in bits D7 and D6 of the IRQ Register (R250). The configuration is shown in Table 9.

Table 9. IRQ Register Configuration

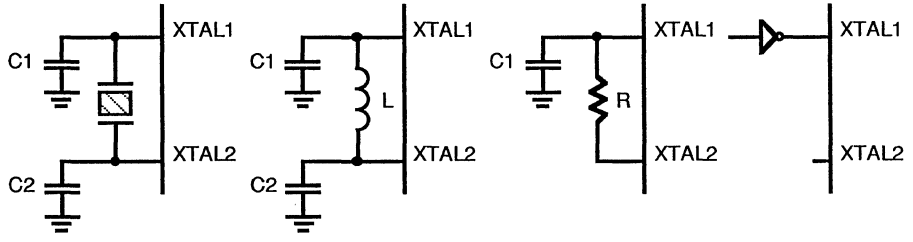
IRQ		Interrupt Edge	
D7	D6	P31	P32
0	0	F	F
0	1	F	R
1	0	R	F
1	1	R/F	R/F

Notes:
F=Falling Edge
R=Rising Edge

Clock. The Z86E40 on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal, RC, ceramic resonator, or any suitable external clock source (XTAL1 = Input, XTAL2 = Output). The crystal should be AT cut, 10 kHz to 12 MHz max, with a series resistance (RS) less than or equal to 100 Ohms.

The crystal should be connected across XTAL1 and XTAL2 using the recommended capacitors (capacitance is from 10 pF to 100 pF) from each pin to ground. The RC oscillator option can be selected in the programming mode. The RC oscillator configuration must be an external resistor connected from XTAL1 to XTAL2, with a frequency-setting capacitor from XTAL1 to ground (Figure 20).

Note: RC OSC may not reach to 12 MHz



Ceramic Resonator or Crystal
C1, C2 = 47 pF TYP *
F = 8 MHz

LC
C1, C2 = 22 pF

L = 130 µH *
F = 3 MHz *

RC
@ 5V Vcc (TYP)

C1 = 33 pF *
R = 1K *
F = 6.6 MHz *

External Clock

* Typical value including pin parasitics

Figure 20. Oscillator Configuration

Power-On Reset (POR). A timer circuit clocked by a dedicated on-board RC oscillator is used for the Power-On Reset (POR) timer function. The POR timer allows V_{CC} and the oscillator circuit to stabilize before instruction execution begins.

The POR timer circuit is a one-shot timer triggered by one of three conditions:

1. Power fail to Power OK status
2. STOP mode recovery (if D5 of SMR=0)
3. WDT timeout

The POR time is a nominal 5 ms. Bit 5 of the Stop Mode Register (SMR) determines whether the POR timer is bypassed after STOP mode recovery (typical for an external clock and RC/LC oscillators with fast start up times).

HALT. Turns off the internal CPU clock, but not the XTAL oscillation. The counter/timers and external interrupt IRQ0, IRQ1, and IRQ2 remain active. The device is recovered by interrupts, either externally or internally generated.

STOP. This instruction turns off the internal clock and external crystal oscillation and reduces the standby current to 10 microamperes or less. STOP mode is terminated by one of the following resets: either by WDT timeout, POR, a Stop Mode Recovery Source which is defined by the SMR register or external reset.. This causes the processor to restart the application program at address 000CH. In order to enter STOP (or HALT) mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. To do this, the user executes a NOP instruction (opcode=FFH) immediately before the appropriate sleep instruction. For example:

```
FF      NOP   ;      clear the pipeline
6F      STOP  ;      enter STOP mode
or
FF      NOP   ;      clear the pipeline
7F      HALT  ;      nter HALT mode
```

Port Configuration Register (PCON). The PCON register configures the ports individually; comparator output on Port 3, Open Drain on Port 0 and Port 1, low EMI on Port's 0, 1, 2 and 3, and low EMI oscillator. The PCON register is located in the expanded register file at bank F, location 00 (Figure 21).

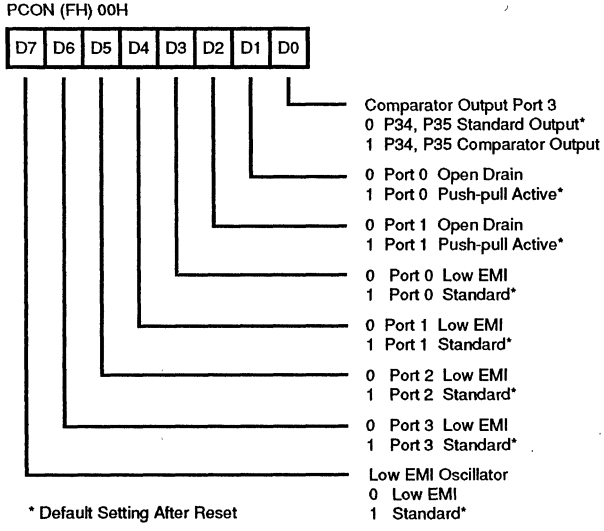


Figure 21. Port Configuration Register (PCON)

Comparator Output Port 3 (D0). Bit 0 controls the comparator use in Port 3. A 1 in this location brings the comparator outputs to P34 and P35, and a 0 releases the Port to its standard I/O configuration.

Port 0 Open Drain (D1). Port 0 can be configured as an Open-drain by resetting this bit (D1=0) or configured as Push-pull Active by setting this bit (D1=1). The default value is 1.

Port 1 Open Drain (D2). Port 1 can be configured as an Open-drain by resetting this bit (D2=0) or configured as Push-pull Active by setting this bit (D2=1). The default value is 1.

Low EMI Port 0 (D3). Port 0 can be configured as a Low EMI Port by resetting this bit (D3=0) or configured as a Standard Port by setting this bit (D3=1). The default value is 1.

Low EMI Port 1 (D4). Port 1 can be configured as a Low EMI Port by resetting this bit (D4=0) or configured as a Standard Port by setting this bit (D4=1). The default value is 1.

Low EMI Port 2 (D5). Port 2 can be configured as a Low EMI Port by resetting this bit (D5=0) or configured as a Standard Port by setting this bit (D5=1). The default value is 1.

Low EMI Port 3 (D6). Port 3 can be configured as a Low EMI Port by resetting this bit (D6=0) or configured as a Standard Port by setting this bit (D6=1). The default value is 1.

Low EMI OSC (D7). This bit of the PCON Register controls the low EMI noise oscillator. A 1 in this location configures the oscillator with standard drive, while a 0 configures the oscillator with low noise drive. The low EMI mode will reduce the drive of the oscillator (OSC).

Stop Mode Recovery Register (SMR). This register selects the clock divide value and determines the mode of STOP mode recovery (Figure 22). All bits are Write only except Bit 7 which is a Read only. Bit 7 is a flag bit that is hardware set on the condition of STOP Recovery and reset by a power on cycle. Bit 6 controls whether a low or high level is required from the recovery source. Bit 5 controls the reset delay after recovery. Bits 2, 3, and 4 of the SMR register specify the STOP Mode Recovery Source. The SMR is located in bank F of the Expanded Register Group at address OBH.

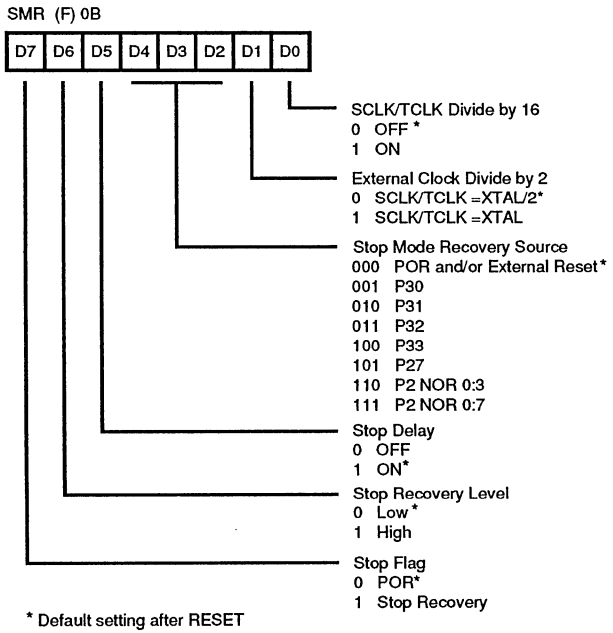


Figure 22. Stop Mode Recovery Register

SCLK/TCLK divide-by-16 Select (D0). This bit of the SMR controls a divide-by-16 prescaler of SCLK/TCLK. The purpose of this control is to selectively reduce device power consumption during normal processor execution (SCLK control) and/or HALT mode (where TCLK sources counter/timers and interrupt logic).

External Clock Divide By 2 (D1). This bit can eliminate the oscillator divide-by-two circuitry. When this bit is 0, the System Clock (SCLK) and Timer Clock (TCLK) are equal to the external clock frequency divided by two. The SCLK/TCLK is equal to the external clock frequency when this bit is set (D1=1). Using this bit together with D7 of PCON further helps lower EMI (i.e. D7 (PCON) = 0, D1 (SMR) = 1). The default setting is zero.

STOP Mode Recovery Source (D2, D3, and D4). These three bits of the SMR register specify the wake up source of the STOP Mode recovery (Figure 23). Table 10 shows the SMR source selected with the setting of D2 to D4. P31-P33 cannot be used to wake up from STOP mode when programmed as analog inputs.

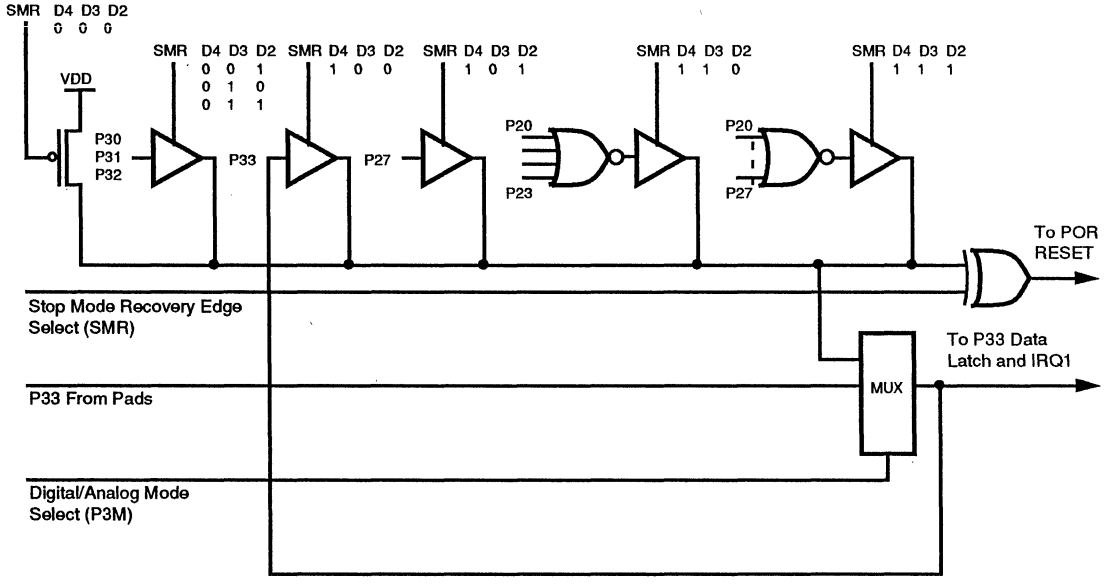


Figure 23. Stop Mode Recovery Source

Table 10. Stop Mode Recovery Source

D4	D3	D2	SMR Source selection
0	0	0	POR recovery only
0	0	1	P30 transition
0	1	0	P31 transition (Not in analog mode)
0	1	1	P32 transition (Not in analog mode)
1	0	0	P33 transition (Not in analog mode)
1	0	1	P27 transition
1	1	0	Logical NOR of Port 2 bits 0:3
1	1	1	Logical NOR of Port 2 bits 0:7

STOP Mode Recovery Delay Select (D5). The 5 ms RESET delay after STOP Mode Recovery is disabled by programming this bit to a zero. A 1 in this bit will cause a 5 ms RESET delay after STOP Mode Recovery. The default condition of this bit is 1. If the fast wake up mode is selected, the STOP Mode Recovery source needs to be kept active for at least 5 TpC.

STOP Mode Recovery Level Select (D6). A 1 in this bit defines that a high level on any one of the recovery sources wakes the Z86E40 from STOP Mode. A 0 defines low level recovery. The default value is 0.

Cold or Warm Start (D7). This bit is set by the device upon entering STOP Mode. A 0 in this bit indicates that the device has been reset by POR (cold). A 1 in this bit indicates the device was awakened by a SMR source (warm).

Watch Dog Timer Mode Register (WDTMR). The WDT is a retriggerable one-shot timer that resets the Z8 if it reaches its terminal count. The WDT is disabled after Power On Reset and initially enabled by executing the WDT instruction and refreshed on subsequent executions of the WDT instruction. The WDT cannot be disabled when it has been enabled. The WDT is driven either by an on-board RC oscillator or an external oscillator from XTAL1 pin. The POR clock source is selected with bit 4 of the WDT register.

WDT Timeout Period (D0 and D1). Bits 0 and 1 control a tap circuit that determines the timeout periods that can be obtained (Table 11). The default value of D0 and D1 are 1 and 0, respectively.

Table 11. Timeout Period of the WDT

D1	D0	Timeout of the internal RC OSC	Timeout of the external clock
0	0	5 ms	256TpC
0	1	15 ms	512TpC
1	0	25 ms	1024TpC
1	1	100 ms	4096TpC

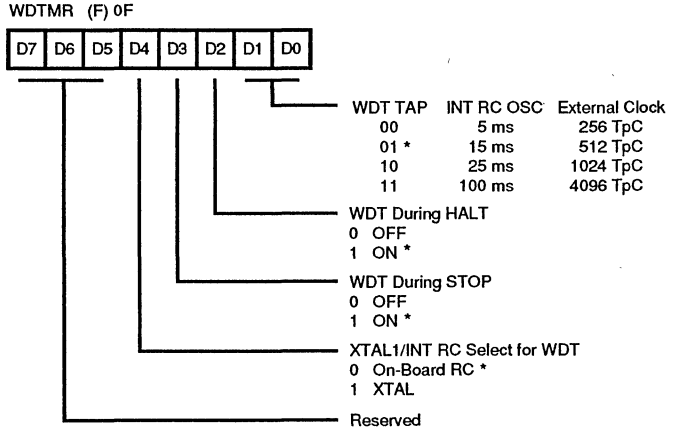
Notes:
TpC = External clock cycle.
The default setting is 15 ms.

WDT During HALT Mode (D2). This bit determines whether or not the WDT is active during HALT mode. A 1 indicates that the WDT is active during HALT. A 0 disables the WDT in HALT mode. The default value is 1.

WDT During STOP Mode (D3). This bit determines whether or not the WDT is active during STOP mode. A 1 indicates active during STOP. A 0 disables the WDT during STOP mode.

Clock Source For WDT (D4). This bit determines which oscillator source is used to clock the internal POR and WDT counter chain. If the bit is a 1, the internal RC oscillator is bypassed and the POR and WDT clock source is driven from the external pin, XTAL1. The default configuration of this bit is 0, which selects the RC oscillator.

Bit 5 through Bit 7 are reserved. The WDTMR register is accessible only during the first 64 processor cycles (128 external XTAL clock cycles) from the execution of the first instruction after Power-On Reset, Watch Dog reset or a STOP Mode Recovery (Figures 24 and 25). After this point, the register cannot be modified by any means, intentional or otherwise. The WDTMR cannot be read and is located in bank F of the Expanded Register Group at address location OFH.



* Default setting after RESET

Figure 24. Watchdog Timer Mode Register

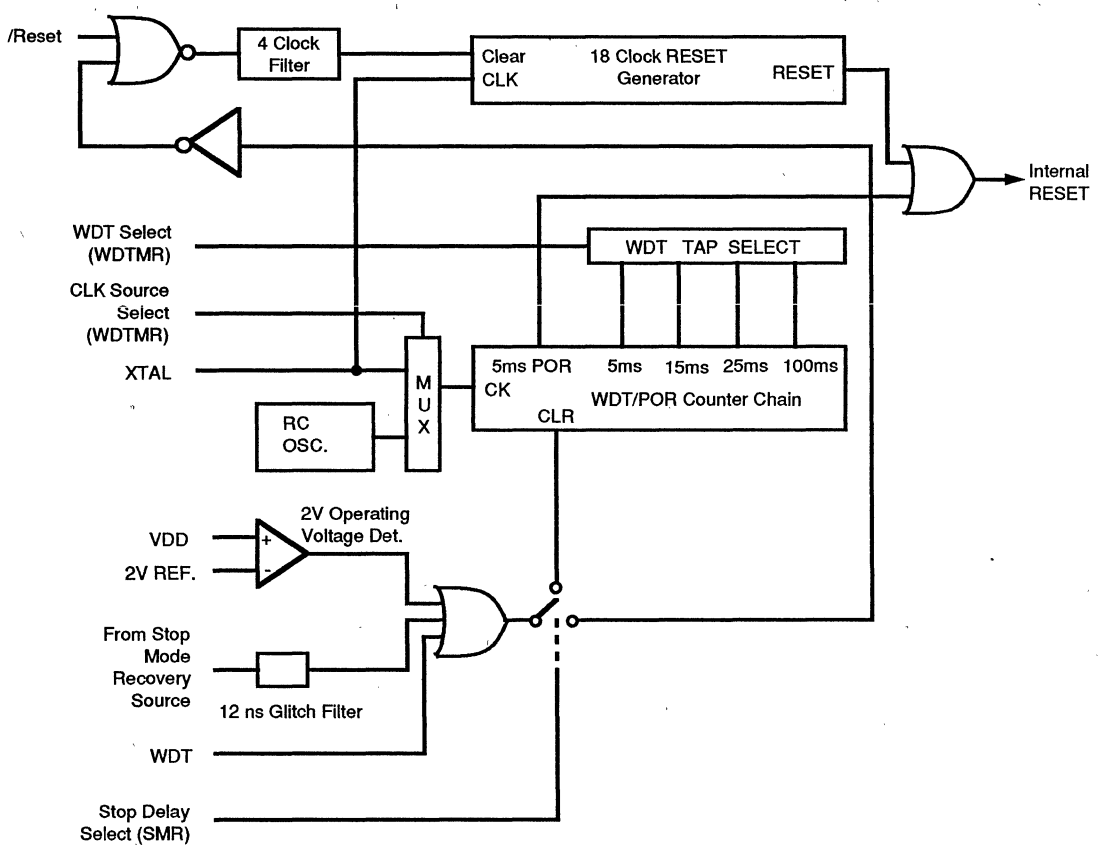


Figure 25. Resets and WDT

Auto Reset Voltage. An on-board Voltage Comparator checks that V_{CC} is at the required level to ensure correct operation of the device. Reset is globally driven if V_{CC} is below V_{RST} (Auto Reset Voltage - Figure 26).

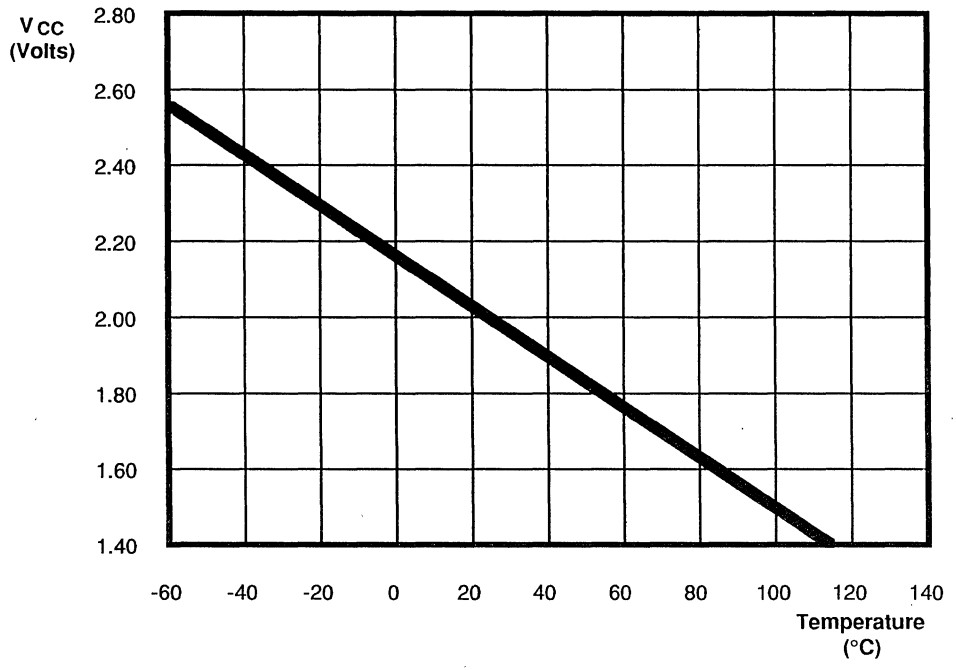


Figure 26. Typical Z86E40 V_{RST} Voltage vs Temperature

EPROM Programming Mode

Table 12 shows the programming voltages of each programming mode. Table 13, Figures 27, 28 and 29 show the programming timing of each programming mode. Figure 30 shows the circuit diagram of a Z86E40 programming adaptor which adapts from 2764A to Z86E40. Figure 31, shows the flow-chart of an Intelligent Programming Algorithm, which is compatible with 2764A EPROM (Z86E40 is 4K EPROM, 2764A is 8K EPROM). Since the EPROM size of Z86E40 differs from 2764A, the programming address range has to be set from 0000H to 0FFFH. Otherwise, the upper 4K of data (1000H-1FFFH) will overwrite the lower 4K of data.

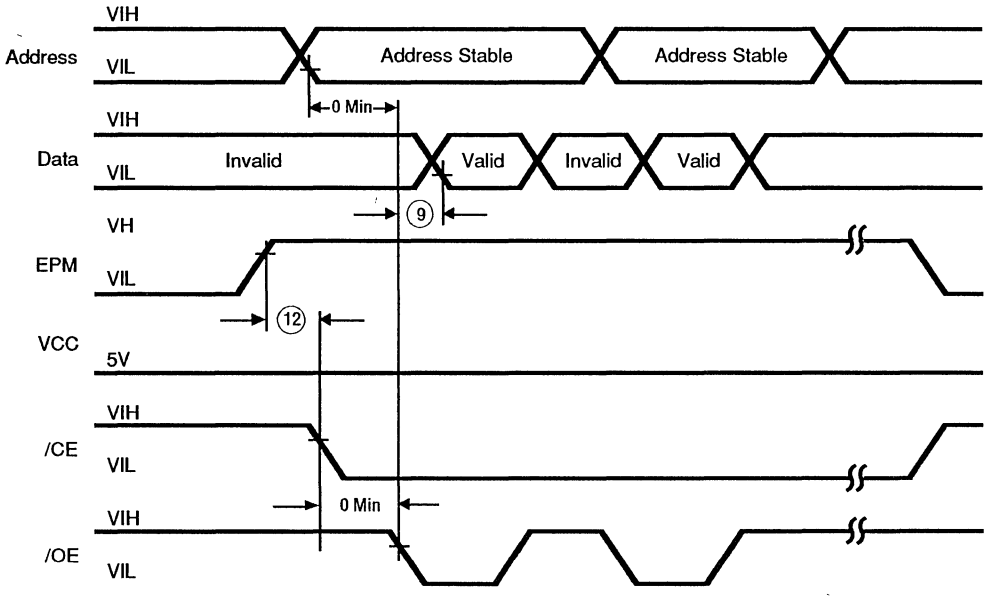
Table 12. EPROM Programming Table

PROGRAMMING MODES	V_{PP} (P33)	EPM (P32)	/CE (XTAL1)	/OE (P31)	/PGM (P30)	ADDR (PORT 2)	DATA	V_{CC}
EPROM READ	X	V_H	V_{IL}	V_{IL}	V_{IH}	Addr	Data Out	5.0V
PROGRAM	V_{PP}	X	V_{IL}	V_{IH}	V_{IL}	Addr	Data In	6.0V
PROGRAM VERIFY	V_{PP}	X	V_{IL}	V_{IL}	V_{IH}	Addr	Data Out	6.0V
EPROM PROTECT	V_{PP}	V_H	V_H	V_{IH}	V_{IL}	X	X	6.0V
RAM PROTECT	V_{PP}	V_{IH}	V_H	V_{IL}	V_{IL}	X	X	6.0V
RC OSCILLATOR	V_{PP}	V_{IL}	V_H	V_{IH}	V_{IL}	X	X	6.0V

Notes:
V_{PP} = 12.5V ± 0.5V
V_H = 12.5V ± 0.5V
X = TTL Level (irrelevant)
V_{IH} = 5.0V
V_{IL} = 0V

Table 13. EPROM Programming Timing

Parameters	Name	Min	Max	Units
1	Address Setup Time	2		µs
2	Data Setup Time	2		µs
3	V_{PP} Setup Time	2		µs
4	V_{CC} Setup Time	2		µs
5	Chip Enable SetupTime	2		µs
6	Program Pulse Width	0.95	1.05	ms
7	Data Hold Time	2		µs
8	OE Setup Time	2		µs
9	Data Access Time		200	ns
10	Data Output Float Time		100	ns
11	Overprogram Pulse Width	2.85	78.75	ms
12	EPM Setup Time	2		µs
13	OE Setup Time	2		µs
14	Address to OE Setup Time	2		µs
15	Option Bit Program Pulse Width		78.75	ms



Note:
Vpp is Irrelevant
PGM is at VIH

Figure 27. EPROM READ Mode Timing Diagram

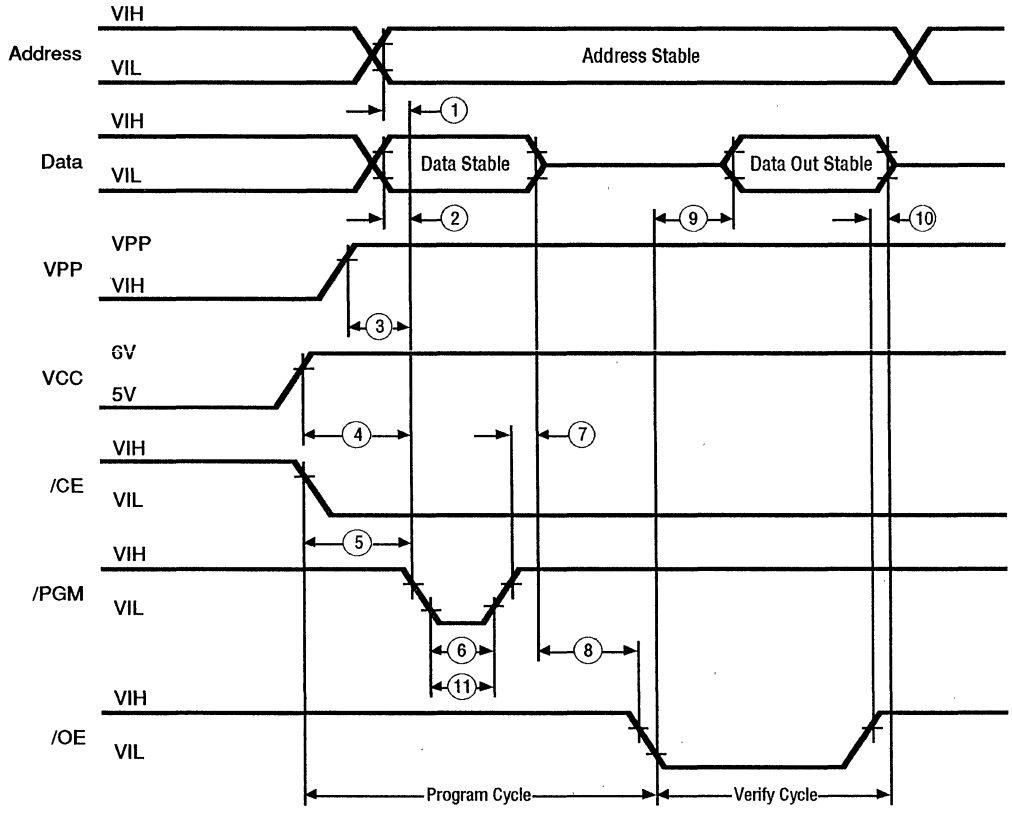


Figure 28. Timing Diagram of EPROM Program and Verify Modes

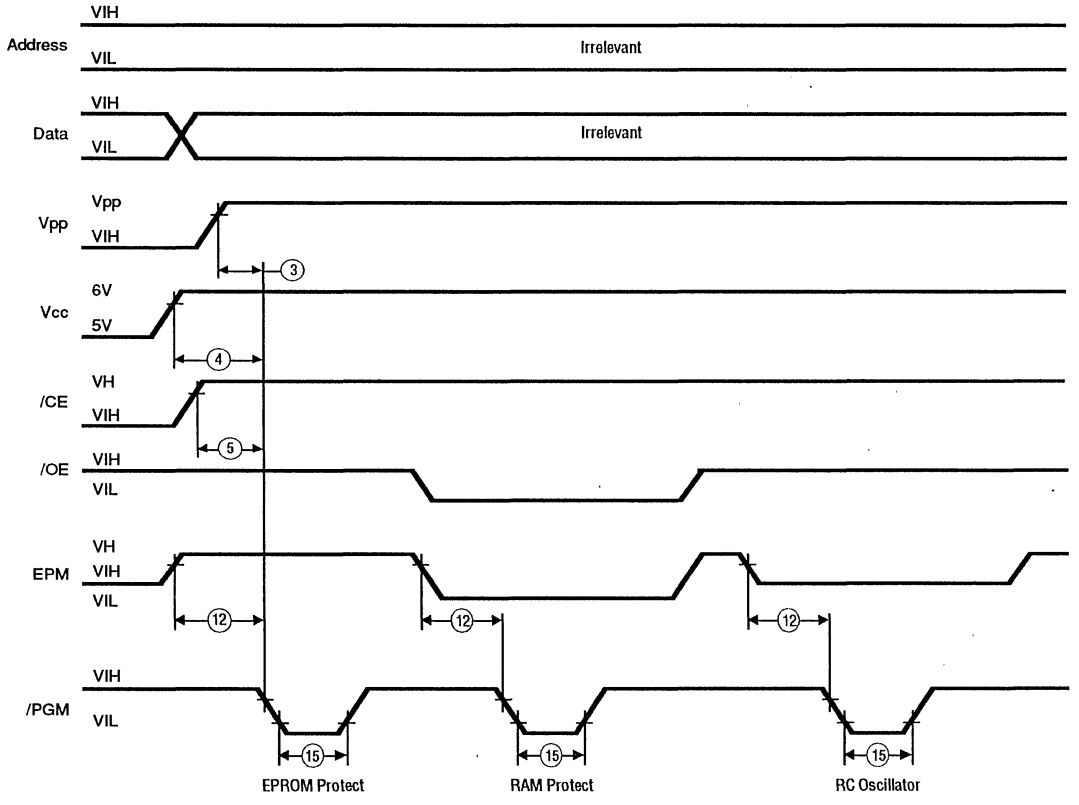
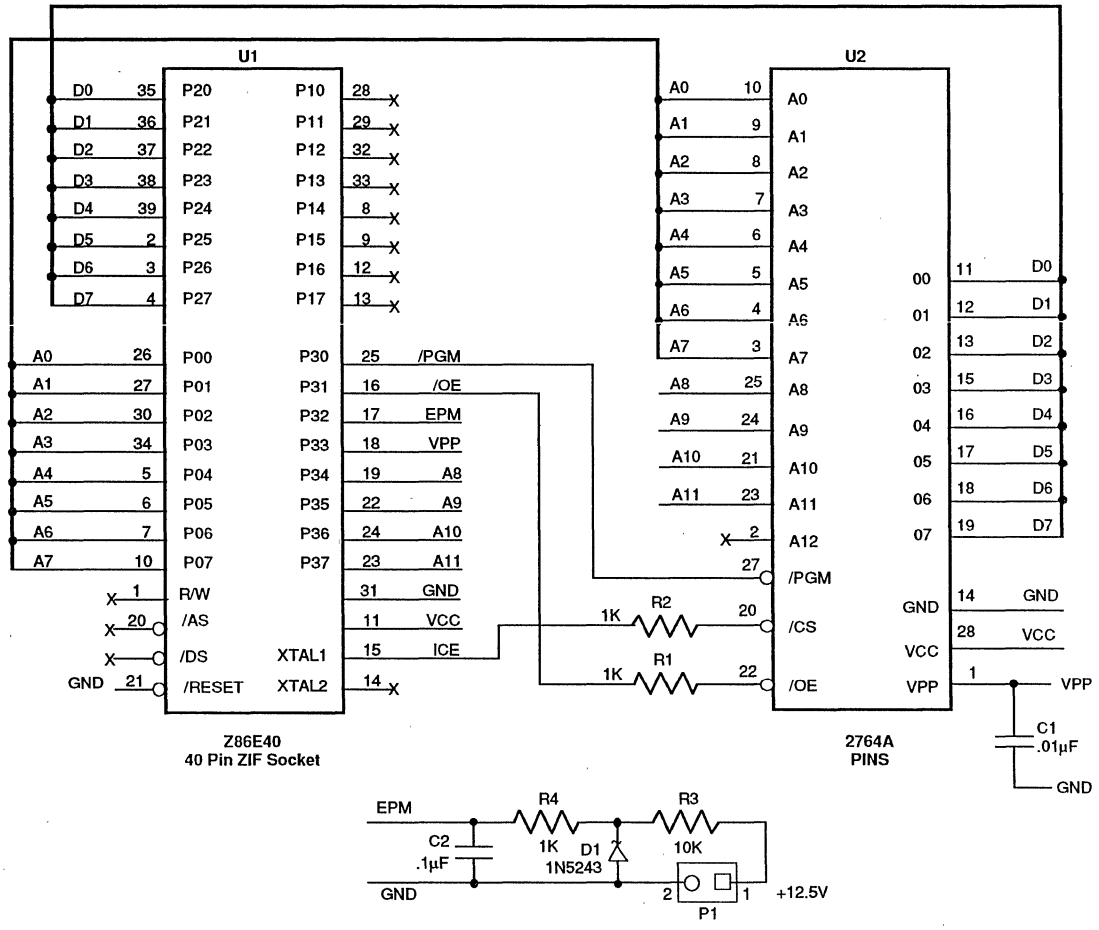


Figure 29. Timing Diagram of EPROM Protect, RAM Protect and RC OSC Modes



Note: The programming address has to be set to 0000H -0FFFH (lower 4 Kbyte memory)

Figure 30. Z86E40 Z8 OTP Programming Adapter

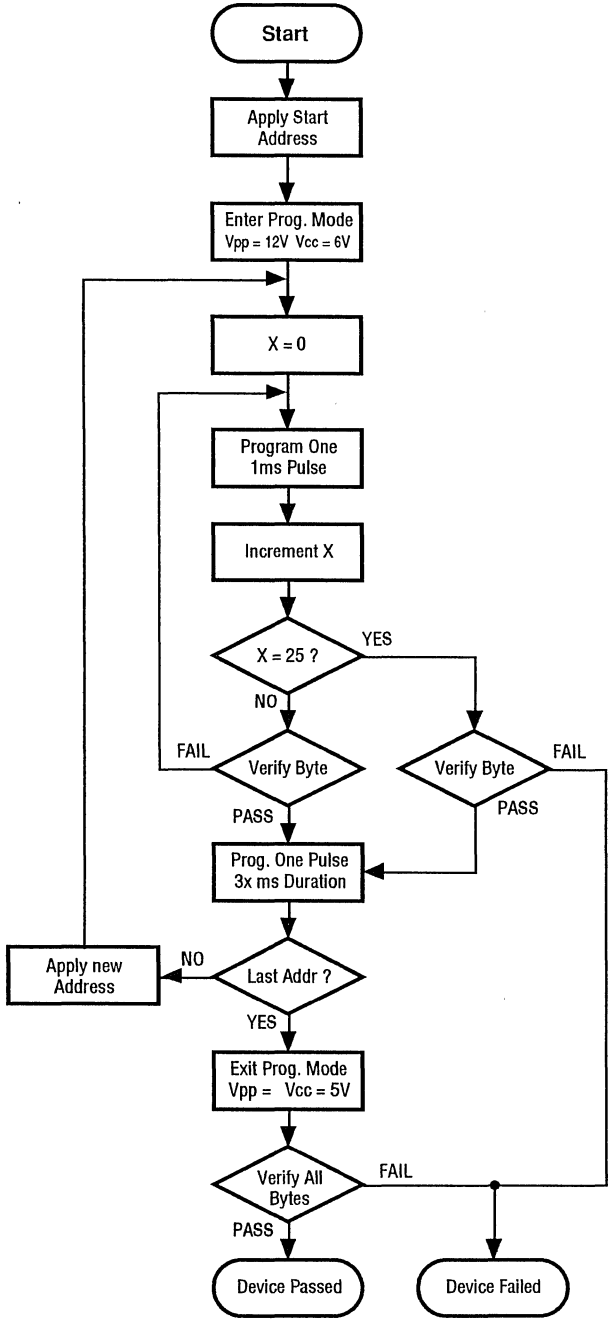


Figure 31. Z86E40 Programming Algorithm

ABSOLUTE MAXIMUM RATINGS

Symbol	Description	Min	Max	Units
V_{CC}	Supply Voltage (*)	-0.3	+7.0	V
T_{STG}	Storage Temp	-65°	+150°	C
T_A	Oper Ambient Temp	†	†	C
	Power Dissipation		2.2	W

Notes:
* Voltage on all pins with respect to GND.
† See Ordering Information.

Stress greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for an extended period may affect device reliability.

STANDARD TEST CONDITIONS

The characteristics listed below apply for standard test conditions as noted. All voltages are referenced to GND. Positive current flows into the referenced pin (Figure 32).

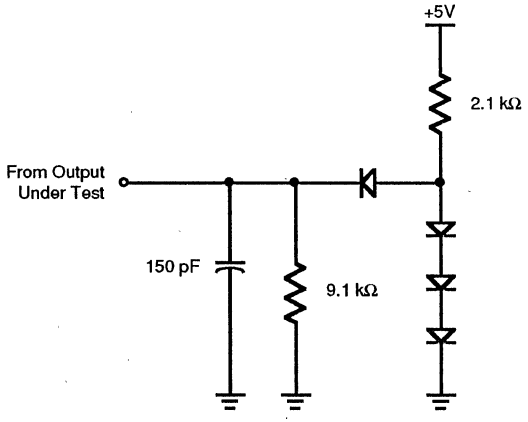


Figure 32. Test Load Diagram

CAPACITANCE

T_A = 25°C; V_{CC} = GND = 0V; f = 1.0 MHz; unmeasured pins to GND.

Parameter	Max
Input capacitance	12 pF
Output capacitance	12 pF
I/O capacitance	12 pF

V_{CC} SPECIFICATION

Low V_{CC}	4.3V ± 0.3V
High V_{CC}	5.0V ± 0.5V

DC ELECTRICAL CHARACTERISTICS

Sym	Parameter	V_{CC} Note [3]	T_A = 0°C to +70°C Min	Max	Typical at 25°C	Units	Conditions	Notes
	Max Input Voltage	4.0V		V_{CC}+0.5		V	I_{IN} 250 µA	
		5.0V		V_{CC}+0.5		V	I_{IN} 250 µA	
V_{CH}	Clock Input High Voltage	4.0V	0.7 V_{CC}	V_{CC}+0.3	1.3	V	Driven by External Clock Generator	
		5.0V	0.7 V_{CC}	V_{CC}+0.3	2.5	V	Driven by External Clock Generator	
V_{CL}	Clock Input Low Voltage	4.0V	V_{SS}-0.3	0.2 V_{CC}	0.7	V	Driven by External Clock Generator	
		5.0V	V_{SS}-0.3	0.2 V_{CC}	1.5	V	Driven by External Clock Generator	
V_{IH}	Input High Voltage	4.0V	0.7 V_{CC}	V_{CC}+0.3	1.3	V		
		5.0V	0.7 V_{CC}	V_{CC}+0.3	2.5	V		
V_{IL}	Input Low Voltage	4.0V	V_{SS}-0.3	0.2 V_{CC}	0.7	V		
		5.0V	V_{SS}-0.3	0.2 V_{CC}	1.5	V		
V_{OH}	Output High Voltge	4.0V	V_{CC}-0.4		3.8	V	I_{OH} = -2.0 mA	
		5.0V	V_{CC}-0.4		4.8	V	I_{OH} = -2.0 mA	
V_{OL1}	Output Low Voltage	4.0V		0.4	0.2	V	I_{OH} = +4.0 mA	
		5.0V		0.4	0.1	V	I_{OL} = +4.0 mA	
V_{OL2}	Output Low Voltage	4.0V		1.5	0.3	V		
		5.0V		1.5	0.3	V	I_{OL} = +12 mA, 3 Pin Max	
V_{RH}	Reset Input High Voltage	4.0V	.7V_{CC}	V_{CC}+0.3	1.5	V		
		5.0V	.7V_{CC}	V_{CC}+0.3	2.1	V		
V_{RI}	Reset Input Low Voltage	4.0V	V_{SS}-0.3	0.2V_{CC}	1.1			
		5.0V	V_{SS}-0.3	0.2V_{CC}	1.7			
V_{OFFSET}	Comparator Input Offset Voltage	4.0V		50	10	mV		
		5.0V		50	10	mV		
I_{IL}	Input Leakage	4.0V	-10	10	<1	µA	V_{IN} = 0V, V_{CC}	
		5.0V	-10	10	<1	µA	V_{IN} = 0V, V_{CC}	
I_{OL}	Output Leakage	4.0V	-10	10	<1	µA	V_{IN} = 0V, V_{CC}	
		5.0V	-10	10	<1	µA	V_{IN} = 0V, V_{CC}	
I_{IR}	Reset Input Current	4.0V		50	40	µA	V_{CC} = 5.0V, V_{RL} = 0	
		5.0V		60	45	µA	V_{CC} = 5.0V, V_{RL} = 0	
I_{CC}	Supply Current (Standard Mode)	4.0V		12	8.5	mA	@ 8 MHz	[4,5]
		5.0V		16	15.0	mA	@ 8 MHz	[4,5]
		4.0V		15	11.5	mA	@ 12 MHz	[4,5]
		5.0V		20	18.0	mA	@ 12 MHz	[4,5]

DC ELECTRICAL CHARACTERISTICS (Continued)

Sym	Parameter	V_{CC} Note [3]	T_A = 0°C to +70°C Min	Max	Typical at 25°C	Units	Conditions	Notes
I_{CC1}	Standby Current (Standard Mode)	4.0V		4.0	2	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 8 MHz	[4,5]
		5.0V		6.0	3.5	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 8 MHz	[4,5]
		4.0V		5.0	2.5	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 12 MHz	[4,5]
		5.0V		7.5	4.5	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 12 MHz	[4,5]
		4.0V		2.0	1.25	mA	Clock Divide by 16 @ 8 MHz	[4,5]
		5.0V		3.0	1.50	mA	Clock Divide by 16 @ 8 MHz	[4,5]
		4.0V		2.0	1.35	mA	Clock Divide by 16 @ 12 MHz	[4,5]
		5.0V		3.0	1.70	mA	Clock Divide by 16 @ 12 MHz	[4,5]
I_{CC}	Supply Current (Low EMI Mode)	4.0V		6.0	4.0	mA	@ 2 MHz	[4,5]
		5.0V		7.5	5.0	mA	@ 2 MHz	[4,5]
		4.0V		9.5	6.0	mA	@ 4 MHz	[4,5]
		5.0V		12	8.0	mA	@ 4 MHz	[4,5]
I_{CC1}	Standby Current (Low EMI Mode)	4.0V		1.6	0.8	mA	@ 2 MHz	[4,5]
		5.0V		2.0	1.0	mA	@ 2 MHz	[4,5]
		4.0V		2.4	1.2	mA	@ 4 MHz	[4,5]
		5.0V		3.0	1.5	mA	@ 4 MHz	[4,5]
		4.0V		1.0	0.5	mA	Clock Divide by 16 @ 2 MHz	[4,5]
		5.0V		2.0	.75	mA	Clock Divide by 16 @ 2 MHz	[4,5]
		4.0V		1.0	.75	mA	Clock Divide by 16 @ 4 MHz	[4,5]
		5.0V		2.0	1.0	mA	Clock Divide by 16 @ 4 MHz	[4,5]
I_{CC2}	Standby Current	4.0V		10	2	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is not Running	[6]
		5.0V		10	2	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is not Running	[6]
		4.0V		400	250	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is Running	[6]
		5.0V		800	450	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is Running	[6]
I_{ALL}	Auto Latch Low Current	4.0V		-10	-5	µA	$0V < V_{IN} < V_{CC}$	
		5.0V		-10	-5	µA	$0V < V_{IN} < V_{CC}$	
I_{ALH}	Auto Latch High Current	4.0V		20	10	µA	$0V < V_{IN} < V_{CC}$	
		5.0V		20	10	µA	$0V < V_{IN} < V_{CC}$	
I_{POR}	Power On Reset	4.0V	4.0		7.5	ms		
		5.0V	2.5		4.5	ms		
V_{RST}	Auto Reset Voltage			3.0	2.5	V	2 MHz Max Ext. CLK Freq.	

Notes:

[1] I_{CC1}

	Typ	Max	Unit	Freq
Clock Driven on Crystal	3.0	5.0	mA	8 MHz
or XTAL Resonator	0.3	5.0	mA	8 MHz

[2] V_{SS}=0V=GND

[3] 5.0V ± 0.5V,4.0V

[4] All outputs unloaded, I/O pins floating, inputs at rail.

[5] CL1=CL2=100 pF

[6] Same as note [4] except inputs at V_{CC}.

AC ELECTRICAL CHARACTERISTICS

External I/O or Memory Read/Write Timing Diagrams (Standard Mode)

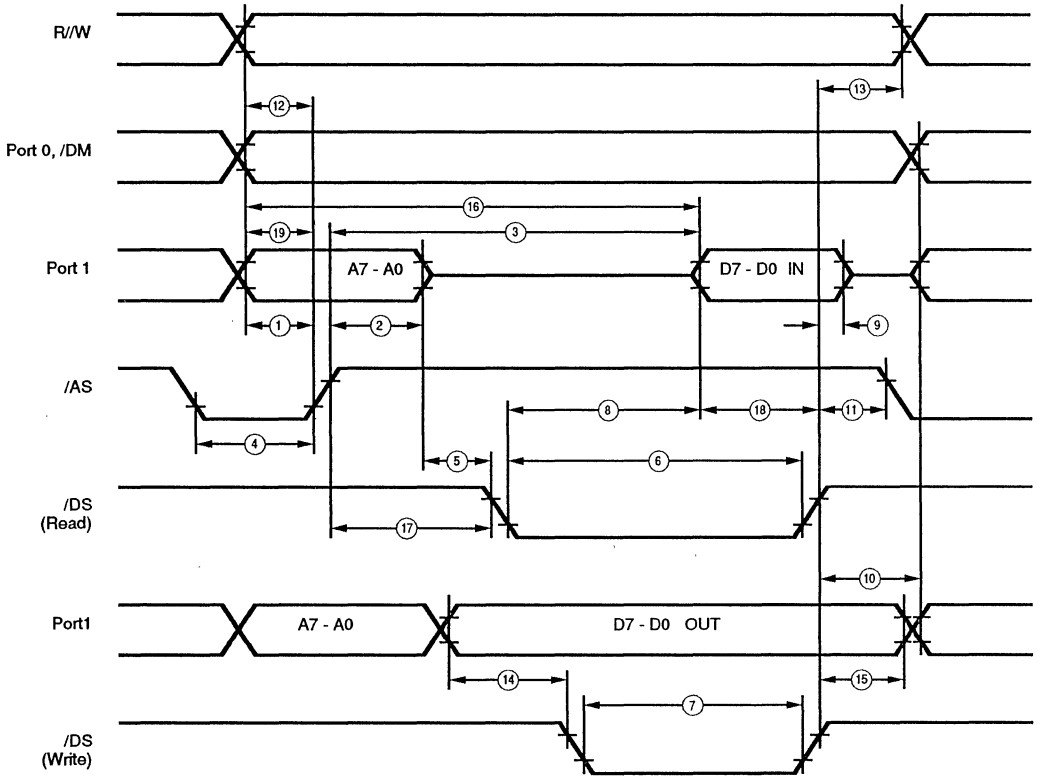


Figure 33. External I/O or Memory Read/Write Timing

AC ELECTRICAL CHARACTERISTICS

External I/O or Memory Read/Write Timing (Standard Mode)

No	Sym	Parameter	V_{cc} Note[3]	Standard Mode 8 MHz Min	8 MHz Max	12 MHz Min	12 MHz Max	Units	Notes
1	TdA(AS)	Address Valid to /AS Rise Delay	4.0V	55		35		ns	[2]
			5.0V	35		35		ns	[2]
2	TdAS(A)	/AS Rise to Address Float Delay	4.0V	70		45		ns	[2]
			5.0V	70		45		ns	[2]
3	TdAS(DR)	/AS Rise to Read Data Req'd Valid	4.0V		400		250	ns	[1,2]
			5.0V		400		250	ns	[1,2]
4	TwAS	/AS Low Width	4.0V	80		55		ns	[2]
			5.0V	80		55		ns	[2]
5	TdAS(DS)	Address Float to /DS Fall	4.0V	0		0		ns	
			5.0V	0		0		ns	
6	TwDSR	/DS (Read) Low Width	4.0V	300		200		ns	[1,2]
			5.0V	300		200		ns	[1,2]
7	TwDSW	/DS (Write) Low Width	4.0V	165		110		ns	[1,2]
			5.0V	165		110		ns	[1,2]
8	TdDSR(DR)	/DS Fall to Read Data Req'd Valid	4.0V		260		150	ns	[1,2]
			5.0V		260		160	ns	[1,2]
9	ThDR(DS)	Read Data /DS Rise Hold Time	4.0V	0		0		ns	[2]
			5.0V	0		0		ns	[2]
10	TdDS(A)	/DS Rise to Address Active Delay	4.0V	85		45		ns	[2]
			5.0V	95		55		ns	[2]
11	TdDS(AS)	/DS Rise to /AS Fall Delay	4.0V	60		30		ns	[2]
			5.0V	70		45		ns	[2]
12	TdR/W(AS)	R/W Valid to /AS Rise Delay	4.0V	70		45		ns	[2]
			5.0V	70		45		ns	[2]
13	TdDS(R/W)	/DS Rise to R/W Not Valid	4.0V	70		45		ns	[2]
			5.0V	70		45		ns	[2]
14	TdDW(DSW)	Write Data Valid to /DS Fall (Write) Delay	4.0V	80		55		ns	[2]
			5.0V	80		55		ns	[2]
15	TdDS(DW)	/DS Rise to Write Data Not Valid Delay	4.0V	70		45		ns	[2]
			5.0V	80		55		ns	[2]
16	TdA(DR)	Address Valid to Read Data Req'd Valid	4.0V		475		310	ns	[1,2]
			5.0V		475		310	ns	[1,2]
17	TdAS(DS)	/AS Rise to /DS Fall Delay	4.0V	100		65		ns	[2]
			5.0V	100		65		ns	[2]
18	TdDI(DS)	Data Output Setup to /DS Rise	4.0V	115		115		ns	[1,2]
			5.0V	75		75		ns	[1,2]
19	TdDM(AS)	/DM Valid to /AS Fall Delay	4.0V	55		35		ns	[2]
			5.0V	55		35		ns	[2]

Notes:

[1] When using extended memory timing add 2TpC.

[2] Timing numbers given are for minimum TpC.

[3] 5.0V ± 0.5V, 4.0V

Standard Test Load

All timining references use 0.9V_{cc} for a logic 1 and 0.1 V_{cc} for a logic 0.

Standard operating temperature range 0°C to 70°C.

AC ELECTRICAL CHARACTERISTICS
Additional Timing Diagrams (Standard Mode)

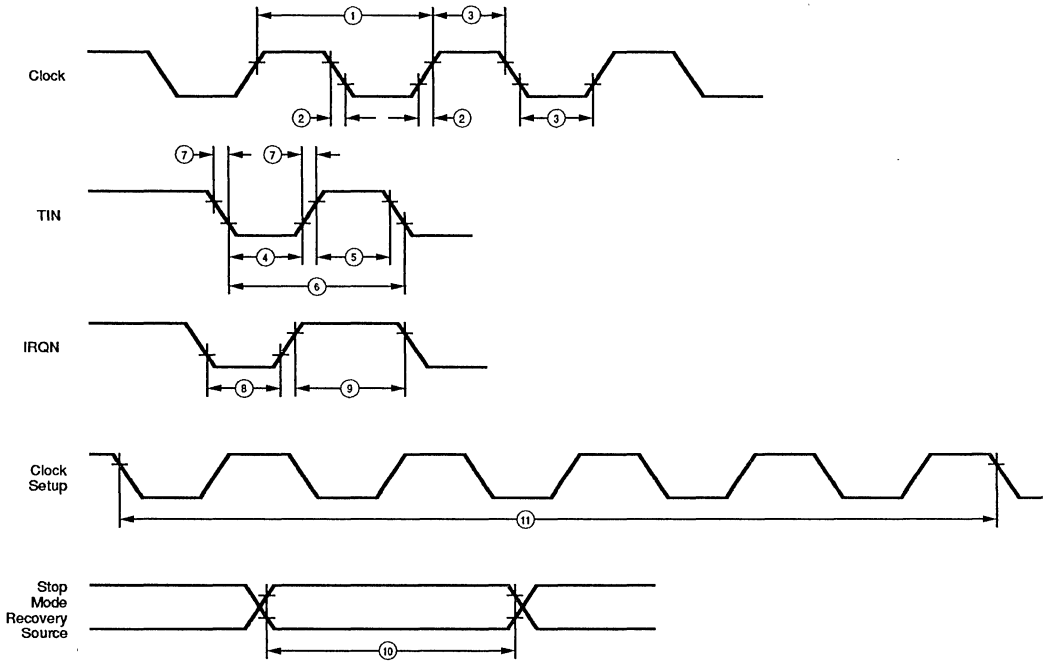


Figure 34. Additional Timing

AC ELECTRICAL CHARACTERISTICS
Additional Timing Table (Standard Mode)

No	Symbol	Parameter	V_{cc} Note[5]	Standard Mode 8 MHz Min	8 MHz Max	12 MHz Min	12 MHz Max	Units	Notes
1	TpC	Input Clock Period	4.0V	125	100000	83	100000	ns	[1]
			5.0V	125	100000	83	100000	ns	[1]
2	TrC,TfC	Clock Input Rise & Fall Times	4.0V		25		15	ns	[1]
			5.0V		25		15	ns	[1]
3	TwC	Input Clock Width	4.0V	37		26		ns	[1]
			5.0V	37		26		ns	[1]
4	TwTinL	Timer Input Low Width	4.0V	100		100		ns	[1]
			5.0V	70		70		ns	[1]
5	TwTinH	Timer Input High Width	4.0V	3TpC		3TpC			[1]
			5.0V	3TpC		3TpC			[1]
6	TpTin	Timer Input Period	4.0V	8TpC		8TpC			[1]
			5.0V	8TpC		8TpC			[1]
7	TrTin,TfTin	Timer Input Rise & Fall Timers	4.0V		100		100	ns	[1]
			5.0V		100		100	ns	[1]
8A	TwIL	Int. Request Low Time	4.0V	100		100		ns	[1,2]
			5.0V	70		70		ns	[1,2]
8B	TwIL	Int. Request Low Time	4.0V	3TpC		3TpC			[1,3]
			5.0V	3TpC		3TpC			[1,3]
9	TwIH	Int. Request Input High Time	4.0V	3TpC		3TpC			[1,2]
			5.0V	3TpC		3TpC			[1,2]
10	Twsm	STOP Mode Recovery Width Spec	4.0V	12		12		ns	
			5.0V	12		12		ns	
			4.0V	5TpC		5TpC			[10]
			5.0V	5TpC		5TpC			[11]
11	Tost	Oscillator Startup Time	4.0V		5TpC		5TpC		[4]
			5.0V		5TpC		5TpC		[4]
12	Twdt	Watchdog Timer Delay Time	4.0V	10		10		ms	[6]
			5.0V	5		5		ms	[6]
			4.0V	20		20		ms	[7]
			5.0V	15		15		ms	[7]
			4.0V	35		35		ms	[8]
			5.0V	25		25		ms	[8]
			4.0V	175		175		ms	[9]
			5.0V	100		100		ms	[9]

Notes:
[1] Timing Reference uses 0.9 V_{cc} for a logic 1 and 0.1 V_{cc} for a logic 0.
[2] Interrupt request via Port 3 (P31-P33)
[3] Interrupt request via Port 3 (P30)
[4] SMR-D5 = 0
[5] 5.0V ± 0.5V, 4.0V
[6] Reg. WDTMR D1=0, D0=0
[7] Reg. WDTMR D1=0, D0=1
[8] Reg. WDTMR D1=1, D0=0
[9] Reg. WDTMR D1=1, D0=1
[10] Reg. SMR-D5=0. No Delay
[11] Reg. SMR-D5=1. With Delay

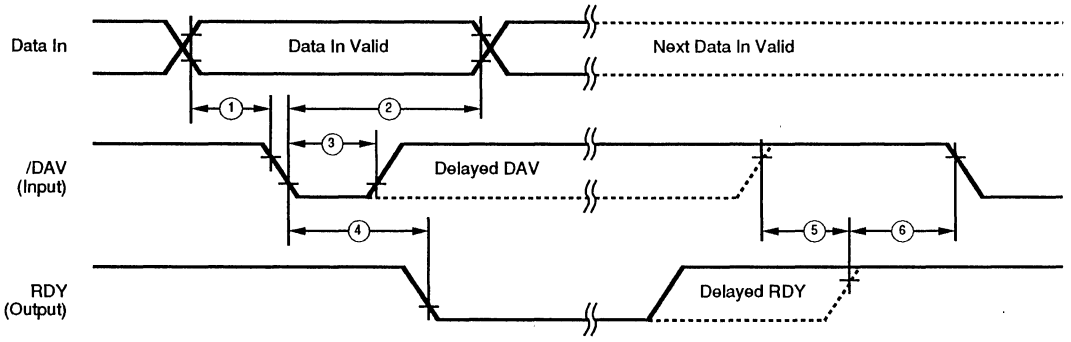


Figure 34. Input Handshake Timing

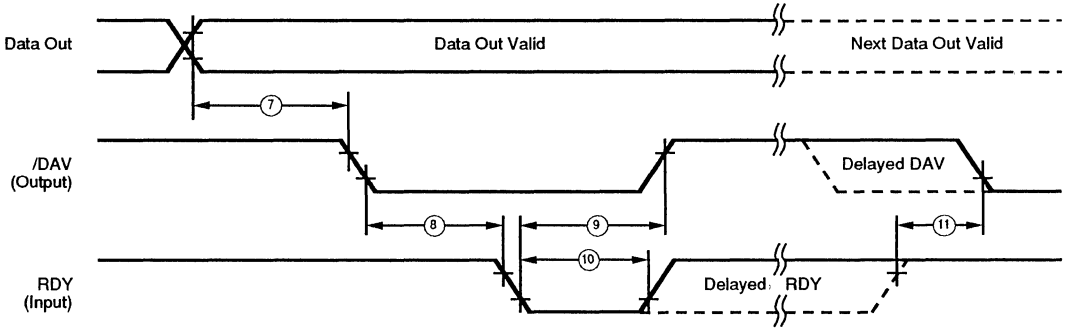


Figure 35. Output Handshake Timing

AC ELECTRICAL CHARACTERISTICS

Handshake Timing Table - (Standard Modes)

No	Sym	Parameter	V_{cc} Note[1]	Standard Mode 8 MHz Min	8 MHz Max	12 MHz Min	12 MHz Max	Data Direction
1	TsDI(DAV)	Data In Setup Time	4.0V	0		0		IN
			5.0V	0		0		IN
2	ThDI(DAV)	Data In Hold Time	4.0V	160		160		IN
			5.0V	115		115		IN
3	TwDAV	Data Available Width	4.0V	155		155		IN
			5.0V	110		110		IN
4	TdDAVI(RDY)	DAV Fall to RDY Fall Delay	4.0V		160		160	IN
			5.0V		115		115	IN
5	TdDAVId(RDY)	DAV Rise to RDY Rise Delay	4.0V		120		120	IN
			5.0V		80		80	IN
6	TdDO(DAV)	RDY Rise to DAV Fall Delay	4.0V	0		0		IN
			5.0V	0		0		IN
7	TcLDAV0(RDY)	Data Out to DAV Fall Delay	4.0V	63		42		OUT
			5.0V	63		42		OUT
8	TcLDAV0(RDY)	DAV Fall to RDY Fall Delay	4.0V	0		0		OUT
			5.0V	0		0		OUT
9	TdRDY0(DAV)	RDY Fall to DAV Rise Delay	4.0V		160		160	OUT
			5.0V		115		115	OUT
10	TwRDY	RDY Width	4.0V	110		110		OUT
			5.0V	80		80		OUT
11	TdRDY0d(DAV)	RDY Rise to DAV Fall Delay	4.0V		110		110	OUT
			5.0V		80		80	OUT

Notes:
[1] 5.0 V ± 0.5V, 4.0V
Standard operating temperature range 0°C to +70°C

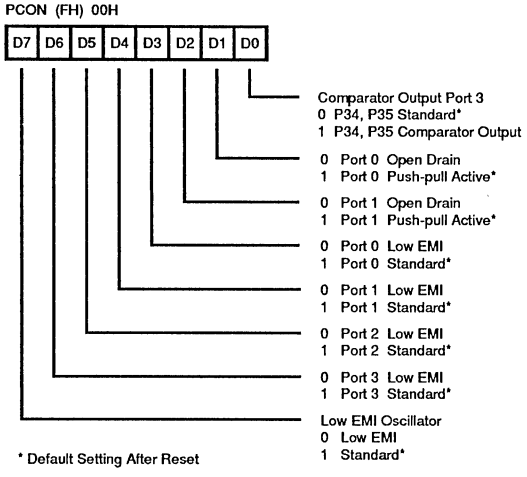


Figure 36. Port Configuration Register

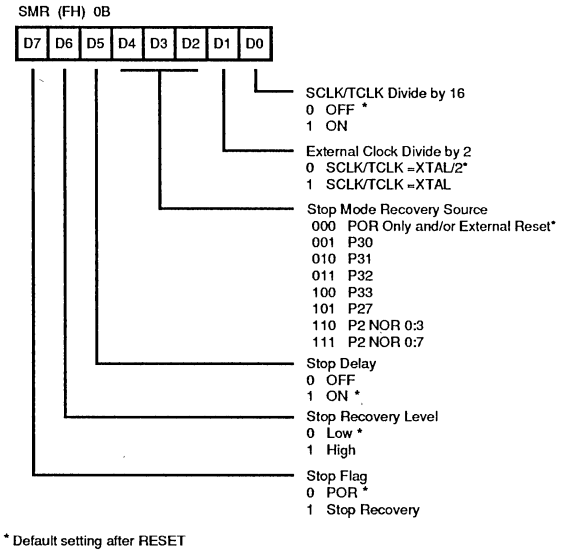


Figure 37. Stop Mode Recovery Register

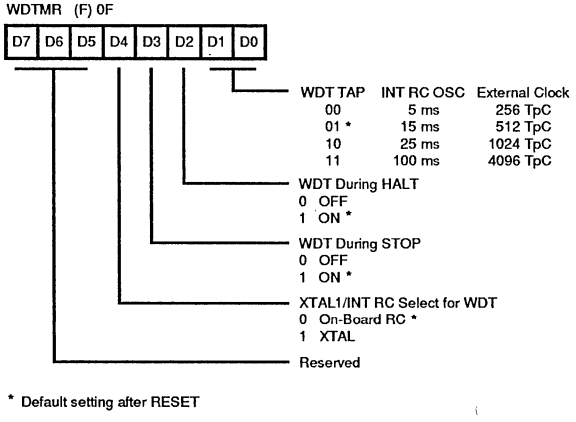


Figure 38. Watchdog Timer Mode Register

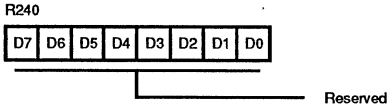


Figure 39. Reserved

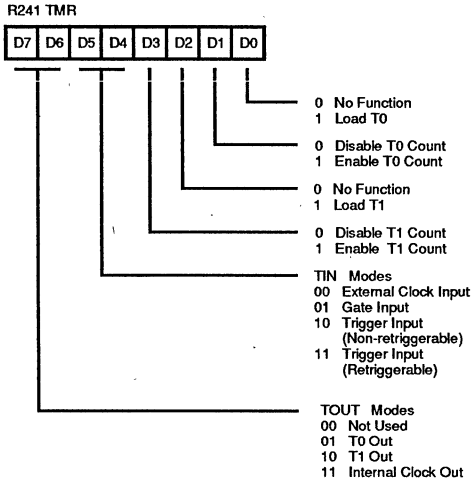


Figure 40. Timer Mode Register
($F1_H$:Read/Write)

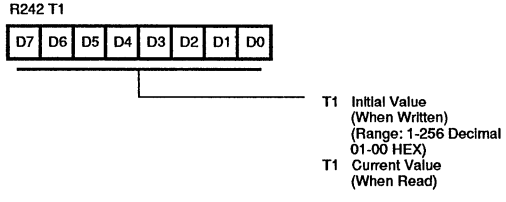


Figure 41. Counter/Timer 1 Register
($F2_H$:Read/Write)

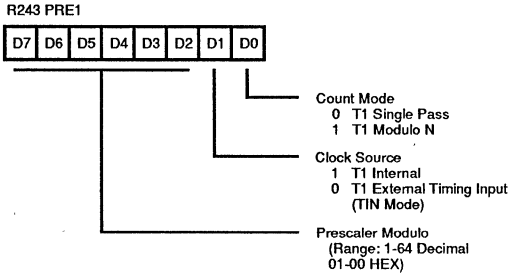


Figure 42. Prescaler 1 Register
($F3_H$:Write Only)

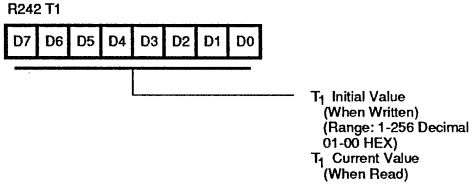


Figure 43. Counter/Timer 0 Register
($F4_H$:Read/Write)

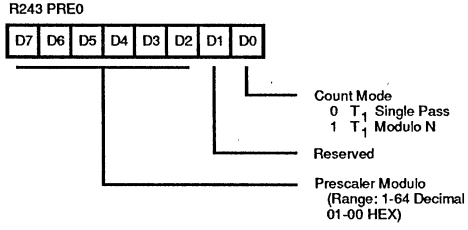


Figure 44. Prescaler 0 Register
($F5_H$:Write Only)

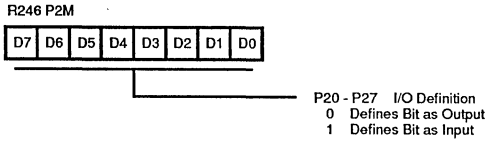


Figure 45. Port 2 Mode Register ($F6_H$: Write Only)

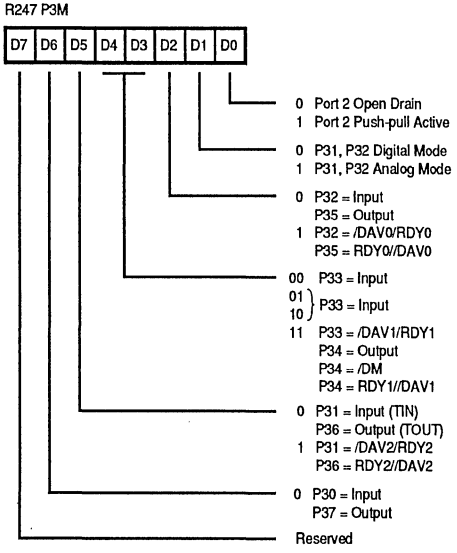


Figure 46. Port 3 Mode Register ($F7_H$:Write Only)

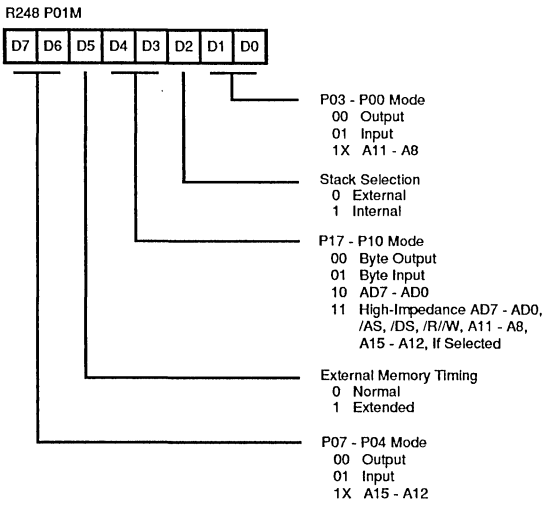


Figure 47. Port 0 and 1 Mode Register ($F8_H$:Write Only)

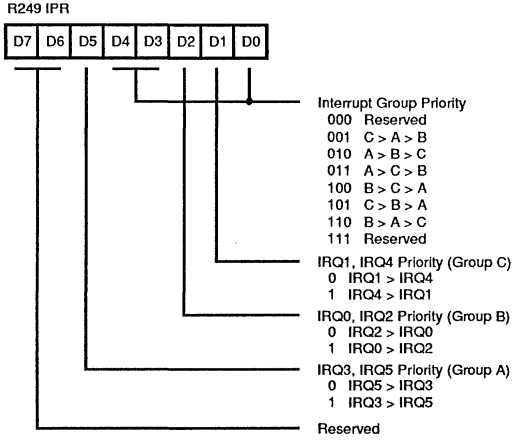


Figure 48. Interrupt Priority Register ($F9_H$:Write Only)

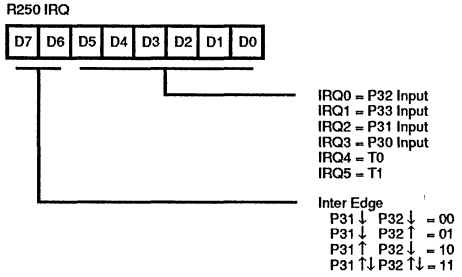


Figure 49. Interrupt Request Register (FA_H:Read/Write)

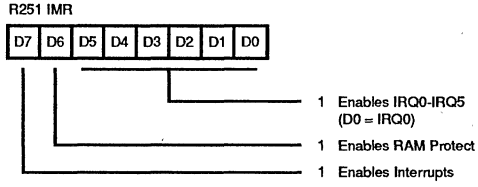


Figure 50. Interrupt Mask Register (FB_H:Read/Write)

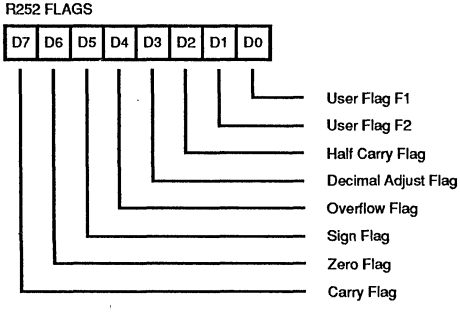


Figure 51. Flag Register (FC_H:Read/Write)

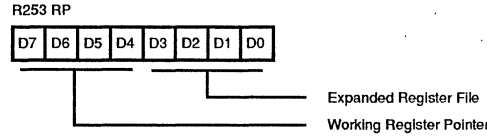


Figure 52. Register Pointer (FD_H:Read/Write)

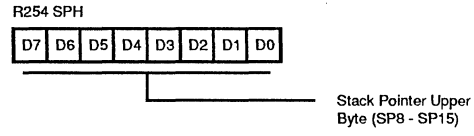


Figure 53. Stack Pointer High (FE_H:Read/Write)

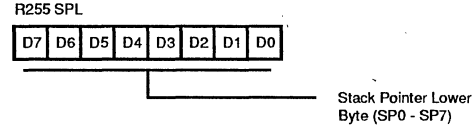


Figure 54. Stack Pointer Low (FF_H:Read/Write)

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

Symbol	Meaning
IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

Symbol	Meaning
dst	Destination location or contents
src	Source location or contents
cc	Condition code
@	Indirect address prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag register (Control Register 252)
RP	Register Pointer (R253)
IMR	Interrupt mask register (R251)

Flags. Control register (R252) contains the following six flags:

Symbol	Meaning
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

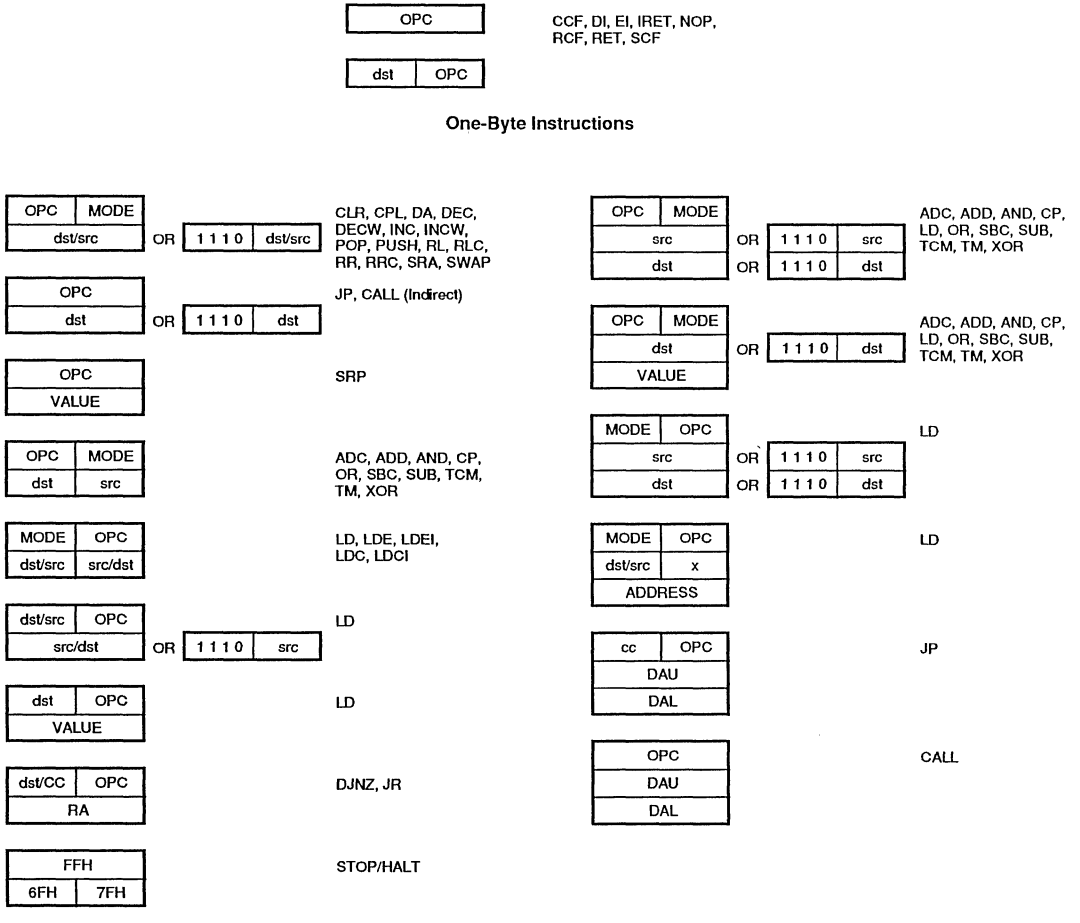
Affected flags are indicated by:

0	Clear to zero
1	Set to one
*	Set to clear according to operation
-	Unaffected
x	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000		Always True	
0111	C	Carry	C = 1
1111	NC	No Carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not Zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No Overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not Equal	Z = 0
1001	GE	Greater Than or Equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater Than	[Z OR (S XOR V)] = 0
0010	LE	Less Than or Equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned Greater Than or Equal	C = 0
0111	ULT	Unsigned Less Than	C = 1
1011	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned Less Than or Equal	(C OR Z) = 1
0000		Never True	

INSTRUCTION FORMATS



INSTRUCTION SUMMARY

Note: Assignment of a value is indicated by the symbol " ← ". For example:

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr (n)" is used to refer to bit (n) of a given operand location. For example:

dst (7)

refers to bit 7 of the destination operand.

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst, src dst←dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst←dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst←dst AND src	†		5[ ]	-	*	*	0	-	-
CALL dst SP←SP - 2 @SP←PC, PC←dst	DA IRR		D6 D4	-	-	-	-	-	-
CCF C←NOT C			EF	*	-	-	-	-	-
CLR dst dst←0	R IR		B0 B1	-	-	-	-	-	-
COM dst dst←NOT dst	R IR		60 61	-	*	*	0	-	-
CP dst, src dst - src	†		A[ ]	*	*	*	*	-	-
DA dst dst←DA dst	R IR		40 41	*	*	*	X	-	-
DEC dst dst←dst - 1	R IR		00 01	-	*	*	*	-	-
DECW dst dst←dst - 1	RR IR		80 81	-	*	*	*	-	-
DI IMR(7)←0			8F	-	-	-	-	-	-
DJNZr, dst r←r - 1 if r ≠ 0 PC←PC + dst Range: +127, -128	RA		rA r = 0 - F	-	-	-	-	-	-
EI IMR(7)←1			9F	-	-	-	-	-	-
HALT			7F	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
INC dst dst←dst + 1	r R IR		rE r = 0 - F 20 21	-	*	*	*	-	-
INCW dst dst←dst + 1	RR IR		A0 A1	-	*	*	*	-	-
IRET FLAGS←@SP; SP←SP + 1 PC←@SP; SP←SP + 2; IMR(7)←1			BF	*	*	*	*	*	*
JP cc, dst if cc is true, PC←dst	DA IRR		cD c = 0 - F 30	-	-	-	-	-	-
JR cc, dst if cc is true, PC←PC + dst Range: +127, -128	RA		cB c = 0 - F	-	-	-	-	-	-
LD dst, src dst←src	r r R r X r Ir R R R R IR	Im R r X r Ir r R IR IM IM R	rC r8 r9 r = 0 - F C7 D7 E3 F3 E4 E5 E6 E7 F5	-	-	-	-	-	-
LDC dst, src dst←src	r	Irr	C2	-	-	-	-	-	-
LDCI dst, src dst←src r←r + 1;rr←rr + 1	Ir	Irr	C3	-	-	-	-	-	-
NOP			FF	-	-	-	-	-	-

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
OR dst, src dst←dst OR src	†		4[ ]	-	*	*	0	-	-
POP dst dst←@SP; SP←SP + 1	R IR		50 51	-	-	-	-	-	-
PUSH src SP←SP - 1; @SP←src	R IR		70 71	-	-	-	-	-	-
RCF C←0			CF	0	-	-	-	-	-
RET PC←@SP; SP←SP + 2			AF	-	-	-	-	-	-
RL dst	R IR		90 91	*	*	*	*	-	-
RLC dst	R IR		10 11	*	*	*	*	-	-
RR dst	R IR		E0 E1	*	*	*	*	-	-
RRC dst	R IR		C0 C1	*	*	*	*	-	-
SBC dst, src dst←dst←src←C	†		3[ ]	*	*	*	*	1	*
SCF C←1			DF	1	-	-	-	-	-
SRA dst	R IR		D0 D1	*	*	*	0	-	-
SRP dst RP←src	Im		31	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
STOP			6F	1	-	-	-	-	-
SUB dst, src dst←dst←src	†		2[ ]	*	*	*	*	1	*
SWAP dst	R IR		F0 F1	X	*	*	X	-	-
TCM dst, src (NOT dst) AND src	†		6[ ]	-	*	*	0	-	-
TM dst, src dst AND src	†		7[ ]	-	*	*	0	-	-
XOR dst, src dst←dst XOR src	†		B[ ]	-	*	*	0	-	-

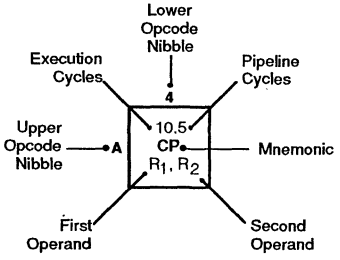
† These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address Mode dst	src	Lower Opcode Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

OPCODE MAP

Upper Nibble (Hex) \ Lower Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								6.0 **WDT**
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								6.0 **STOP**
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								7.0 **HALT**
8	10.5 **DECW** RR1	10.5 **DECW** IR1	12.0 **LDE** r1, Irr2	18.0 **LDEI** Ir1, Irr2												6.1 **DI**
9	6.5 **RL** R1	6.5 **RL** IR1	12.0 **LDE** r2, Irr1	18.0 **LDEI** Ir2, Irr1												6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14.0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2				10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1	12.0 **LDC** r2, Irr1	18.0 **LDCI** Ir2, Irr1	20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6.5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2			3						2				3	1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_2 = Dst address
R_1 or r_2 = Src address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank are not defined.

* 2-byte instruction appears as a 3-byte instruction

ZiLOG

PRODUCT SPECIFICATION

Z86C27-ROM
Z86C97-ROMLESS
CMOS Z8® 8-BIT MICROCONTROLLER

FEATURES

- 8-bit CMOS microcontroller for consumer television applications, 64-pin DIP package.
- Low cost
- Low power consumption
- Fast instruction pointer-1.5 microseconds @ 4 MHz
- Two standby modes-STOP and HALT
- Low voltage detection/voltage sensitive reset
- 35 input/output lines
- On Screen Display Controller
- All digital CMOS levels Schmitt triggered
- 8 Kbytes of ROM
- 236 bytes of RAM
- Two programmable 8-bit Counter/Timers each with 6-bit programmable prescaler.
- Six vectored, priority interrupts from six different sources
- Clock speed up to 4 MHz
- Watch Dog/Power-On Reset Timer
- 4K x 6-bit character generator ROM
- 160 x 7-bit video RAM
- On-chip oscillator that accepts a crystal, ceramic resonator, LC or external clock drive.
- Mask programmable 128 character set displayed in an 8-row by 20-column format, 12 by 15 pixel character cell, capable of supporting English, Korean, Chinese and Japanese high resolution characters.
- Fully programmable color attributes including row character, row background/fringes, frame background/position, bar graph color change, and character size.
- Programmable display position and character size control.
- One Pulse Width Modulator (14-bit resolution) for voltage synthesis tuner control.
- Five Pulse Width Modulators (8-bit resolution) for picture control.
- Seven Pulse Width Modulators (6-bit resolution) for audio control.
- Port 2 (8-bit programmable I/O) and Port 3 (2-bit input, 3-bit output) register mapped ports.
- Port 4 (8-bit output), Port 5 (8-bit LED drive output) and Port 6 (6-bit input and tri-state comparator AFC input) memory mapped I/O ports.

GENERAL DESCRIPTION

The Z86C27 and Z86C97 Digital Television Controller (DTC) introduce a new level of sophistication to single-chip architecture. The Z86C27/C97 are members of the Z8 single-chip microcontroller family with 8 Kbytes of ROM (Z86C27), ROMless (Z86C97) and 236 bytes of RAM. Both devices are housed in a 64-pin DIP package, and are CMOS compatible. Having the ROM/ROMless selectivity, the DTC offers both external memory and pre-programmed ROM which enables the Z8 microcontroller to be used in a high volume production application device embedded with a custom program (customer supplied program). The Z86C97 ROMless offers the use of external memory rather

GENERAL DESCRIPTION (Continued)

than a preprogrammed ROM. This enables the Z8 microcontroller to be used in prototyping, low volume applications or where code flexibility is required. Zilog's DTC offers fast execution, efficient use of memory, sophisticated interrupts, input/output bit manipulation capabilities, and easy hardware/software system expansion along with low cost and low power consumption. The device provides an ideal performance and reliability solution for consumer and industrial television applications.

The Z86C27/C97 architecture is characterized by utilizing Zilog's advanced Superintegration™ design methodology. The devices have an 8-bit internal data path controlled by a Z8 microcontroller, and On Screen Display (OSD) logic circuits/Pulse Width Modulators (PWM). On-chip peripherals include two register mapped I/O ports (Ports 2 and Port 3), Interrupt control logic (1 software, 2 external and 3 internal interrupts) and a standby mode recovery input port (Port 3, pin P30).

The OSD control circuits support 8 rows by 20 columns for 128 kinds of characters. The character color is specified by row. One of the 8 rows is assigned to show two kinds of colors for bar type displays such as volume control. The OSD is capable of displaying either low resolution (5x7 dot pattern) or high resolution (11x15 dot pattern) characters. The Z86C97 currently supports high resolution characters only.

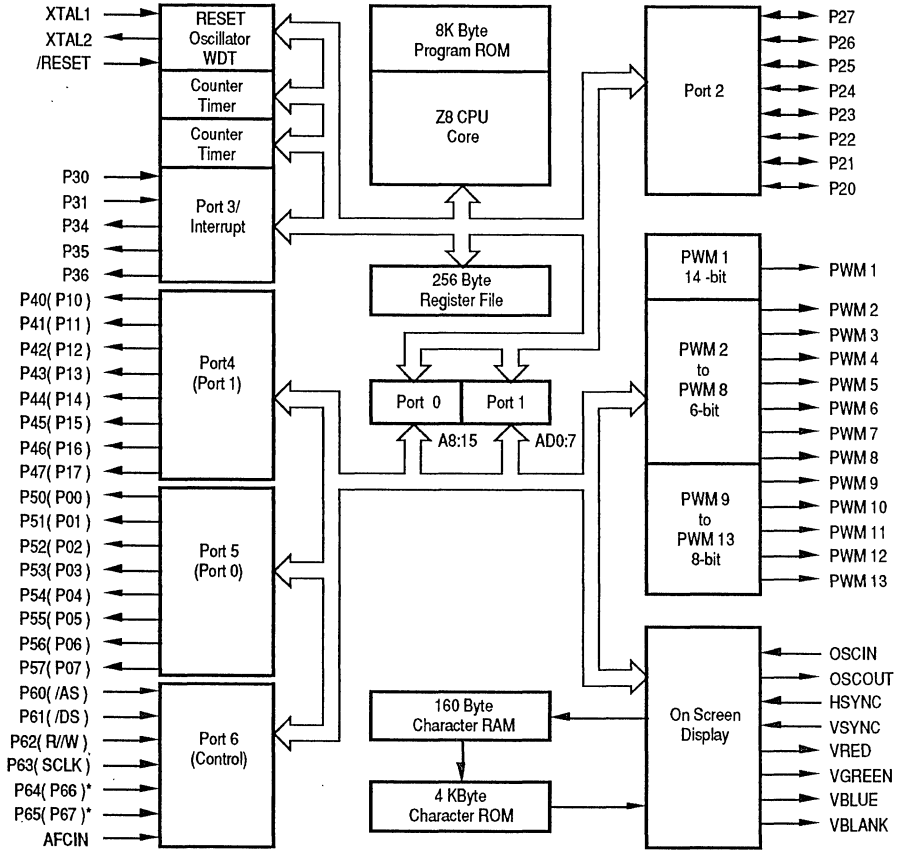
A 14-bit PWM port provides enough voltage resolution for a voltage synthesizer tuning system. Seven 6-bit PWM ports are used for controlling audio signal level. Five 8-bit PWM ports are used to vary picture levels.

The DTC applications demand powerful I/O capabilities. The Z86C27/C97 fulfills this with 35 I/O pins dedicated to input and output. These lines are grouped into five ports, and are configurable under software control to provide timing, status signals, parallel I/O and an address/data bus for interfacing to external memory.

There are three basic address spaces available to support this wide range of configurations: Program Memory, Register File and Data Memory. The Register File is composed of 236 bytes of general purpose register, two I/O Port registers and 15 control and status registers.

To unburden the program from coping with the real-time problems such as counting/timing and data communication, the DTC's offer two on-chip counter/timers with a large number of user selectable modes (Figure 1).

Note: All Signals with a preceding front slash, "/", are active Low, e.g.: B//W (WORD is active Low); /B/W (BYTE is active Low, only).



* ( ) Denotes Z86C97 signal differences.

Figure 1. Functional Block Diagram

PIN CONFIGURATION

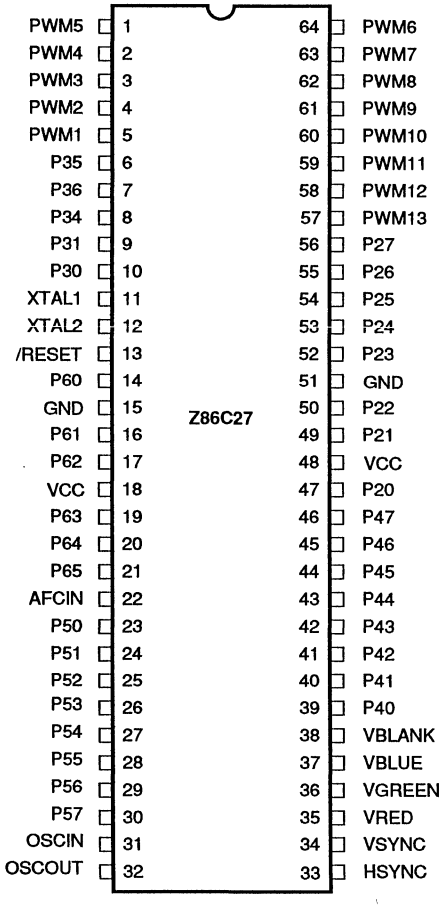


Figure 2. Z86C27 Mask-ROM Plastic DIP

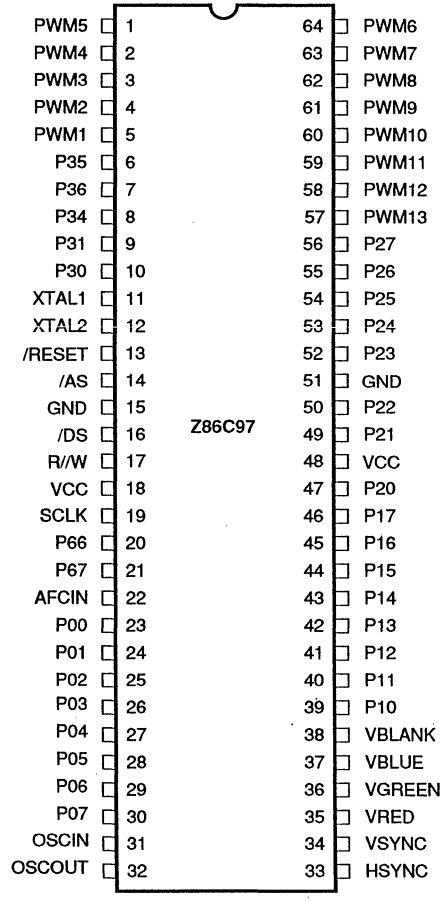


Figure 3. Z86C97 ROMless Plastic DIP

PIN IDENTIFICATION
64-pin DIP Z86C27

Pin	Name	Function	Direction
1	PWM5	Pulse Width Modulator 5	Output
2	PWM4	Pulse Width Modulator 4	Output
3	PWM3	Pulse Width Modulator 3	Output
4	PWM2	Pulse Width Modulator 2	Output
5	PWM1	Pulse Width Modulator 1	Output
6, 7	P35-6	Port 3 pin 5, 6	Output
8	P34	Port 3 pin 4	Output
9	P31	Port 3 pin 1	Input
10	P30	Port 3 pin 0	Input
11	XTAL1	Crystal Oscillator	Input
12	XTAL2	Crystal Oscillator	Output
13	/RESET	System Reset	Input
14	P60	Port 6 pin 0	Input
15	GND	Ground,GND	Input
16	P61	Port 6 pin 1	Input
17	P62	Port 6 pin 2	Input
18	V_{cc}	Power Supply	Input
19-21	P63-5	Port 6 pin 3, 4, 5	Input
22	AFCIN	AFC Voltage Level	Input
23-30	P50-7	Port 5 pin 0, 1, 2, 3, 4, 5, 6, 7	Output
31	OSCIN	Video Dot Clock Osc	Input
32	OSCOUT	Video Dot Clock Osc	Output
33	HSYNC	Horizontal Sync	Input
34	VSYNC	Vertical Sync	Input
35	Vred	Video Red	Output
36	Vgreen	Video Green	Output
37	Vblue	Video Blue	Output
38	Vblank	Video Blank	Output
39-46	P40-7	Port 4 pin 0, 1, 2, 3, 4, 5, 6, 7	Output
47	P20	Port 2 pin 0	In/Output
48	V_{cc}	Power Supply	Input
49,50	P21-2	Port 2 pin 1, 2	In/Output
51	GND	Ground,GND	Input
52-56	P23-7	Port 2 pin 3, 4, 5, 6, 7	In/Output
57	PWM13	Pulse Width Modulator 13	Output
58	PWM12	Pulse Width Modulator 12	Output
59	PWM11	Pulse Width Modulator 11	Output
60	PWM10	Pulse Width Modulator 10	Output
61	PWM9	Pulse Width Modulator 9	Output
62	PWM8	Pulse Width Modulator 8	Output
63	PWM7	Pulse Width Modulator 7	Output
64	PWM6	Pulse Width Modulator 6	Output

PIN IDENTIFICATION (Continued)
64-pin DIP Z86C97

Pin	Name	Function	Direction
1	PWM5	Pulse Width Modulator 5	Output
2	PWM4	Pulse Width Modulator 4	Output
3	PWM3	Pulse Width Modulator 3	Output
4	PWM2	Pulse Width Modulator 2	Output
5	PWM1	Pulse Width Modulator 1	Output
6, 7	P35-6	Port 3 pin 5, 6	Output
8	P34	Port 3 pin 4	Output
9	P31	Port 3 pin 1	Input
10	P30	Port 3 pin 0	Input
11	XTAL1	Crystal Oscillator	Input
12	XTAL2	Crystal Oscillator	Output
13	/RESET	System Reset	Input
14	/AS	Address Strobe	Output
15	GND	Ground, GND	Input
16	/DS	Data Strobe	Output
17	R//W	Read/Write	Output
18	V_{cc}	Power Supply	Input
19	SCLK	System Clock	Output
20-21	P66-7	Port 6 pin 6, 7	Output
22	AFCIN	AFC Analog	Input
23-30	P00-7	Port 0 pin 0,1, 2, 3, 4, 5, 6, 7	Output
31	OSCIN	Video Dot Clock Oscillator	Input
32	OSCOUT	Video Dot Clock Oscillator	Output
33	Hsync	Horizontal Sync	Input
34	Vsync	Vertical Sync	Input
35	Vred	Video Red	Output
36	Vgreen	Video Green	Output
37	Vblue	Video Blue	Output
38	Vblank	Video Blank	Output
39-46	P10-7	Port 1 pin 0,1, 2, 3, 4, 5, 6, 7	Output
47	P20	Port 2 pin 0	In/Output
48	V_{cc}	Power Supply	Input
49-50	P21-2	Port 2 pin 1, 2	In/Output
51	GND	Ground,GND	Input
52-56	P23-7	Port 2 pin 3, 4, 5, 6, 7	In/Output
57	PWM13	Pulse Width Modulator 13	Output
58	PWM12	Pulse Width Modulator 12	Output
59	PWM11	Pulse Width Modulator 11	Output
60	PWM10	Pulse Width Modulator 10	Output
61	PWM9	Pulse Width Modulator 9	Output
62	PWM8	Pulse Width Modulator 8	Output
63	PWM7	Pulse Width Modulator 7	Output
64	PWM6	Pulse Width Modulator 6	Output

PIN DESCRIPTION

XTAL1, XTAL2. (Time-based input, output, respectively). These pins connect to the internal parallel-resonant clock crystal (4MHz max) oscillator circuit with 2 capacitors to GND. XTAL1 is also used as an external clock input.

/AS. *Address Strobe* (output, active Low) is pulsed once at the beginning of each machine cycle. Address output is via Port 0 and Port 1 for all external programs. Memory address transfers are valid at the trailing edge of /AS. Under program control, /AS can be placed in the high impedance state along with Port 0 and Port 1, Data Strobe and Read/Write.

/DS. *Data Strobe* (output, active Low) is active once for each external memory transfer. For READ operations, data must be available prior to the trailing edge of /DS. For WRITE operations, the falling edge of /DS indicates the output data is valid.

R/W. *Read/Write* (output, Write active Low) signal is low when the DTC is writing to the external program or data memory.

SCLK. *System Clock.* SCLK is the internal system clock. It can be used to clock external glue logic.

HSYNC. (input Schmitt triggered, CMOS level). Horizontal Sync is an input pin that accepts an externally generated Horizontal Sync signal of either negative or positive polarity.

VSYNC. (input Schmitt triggered, CMOS level). Vertical Sync is an input pin that accepts an externally generated Vertical Sync signal of either negative or positive polarity.

OSCIN, OSCOUT. (Video Oscillator input, output, respectively). Oscillator input and output pins for on-screen display circuits. These pins connect to an inductor and two capacitors to generate the character dot clock (typically around 6MHz). The dot clock frequency determines the character pixel width and phase synchronized to HSYNC.

Vblank. *Video Blank* (output). CMOS output, programmable polarity. Used as a superimpose control port to display characters from video RAM. The signal controls Y signal output of the CRT and turns off the incoming video display while the characters in video RAM are superimposed on the screen. The red, green, and blue outputs drive the three electron guns on the CRT directly, while the blank output turns off the Y signal.

Vblue. *Video Blue* (output). CMOS Output of the Blue video signal (B-Y) and is programmable for either polarity.

Vgreen. *Video Green* (output). CMOS Output of the Green video signal (G-Y) and is programmable for either polarity.

Vred. *Video Red* (output). CMOS Output of the Red video signal (R-Y) and is programmable for either polarity.

PIN DESCRIPTION (Continued)

Port 0 (P00-P07). Port 0 is an 8-bit, CMOS compatible, High Address Bus (A15-A8). In the ROMless mode this port is used to output the high order address (A15-A8) during an external memory cycle (Figure 4).

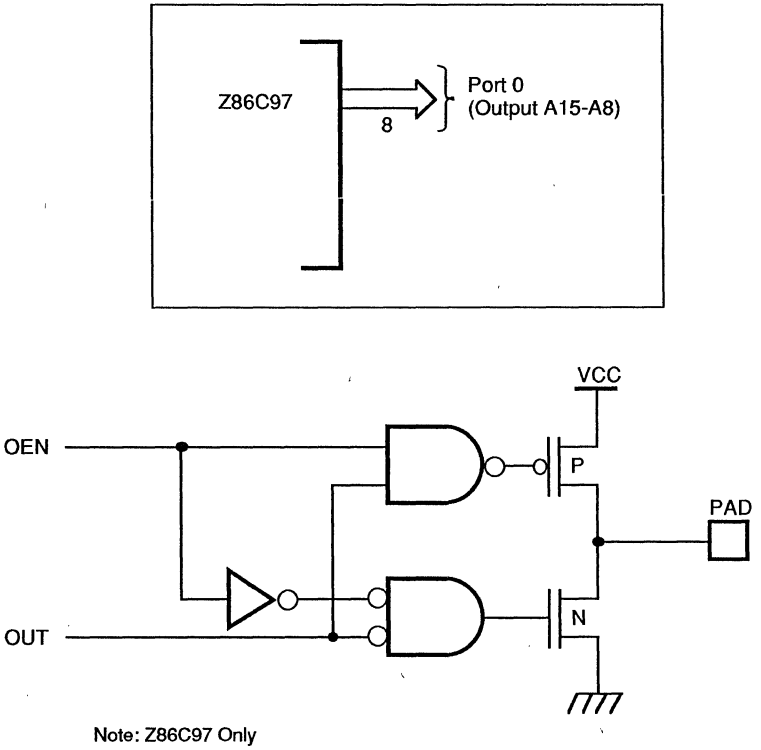


Figure 4. Port 0 Configuration

Port 1 (P10-P17). Port 1 is an 8-bit, CMOS compatible, Multiplexed Address/Data Bus (A7-A0)/(D7-D0). In the ROMless mode this port multiplexes the low order address (A7-A0 during /AS) and data (D7-D0 during /DS) for an external memory cycle (Figure 5).

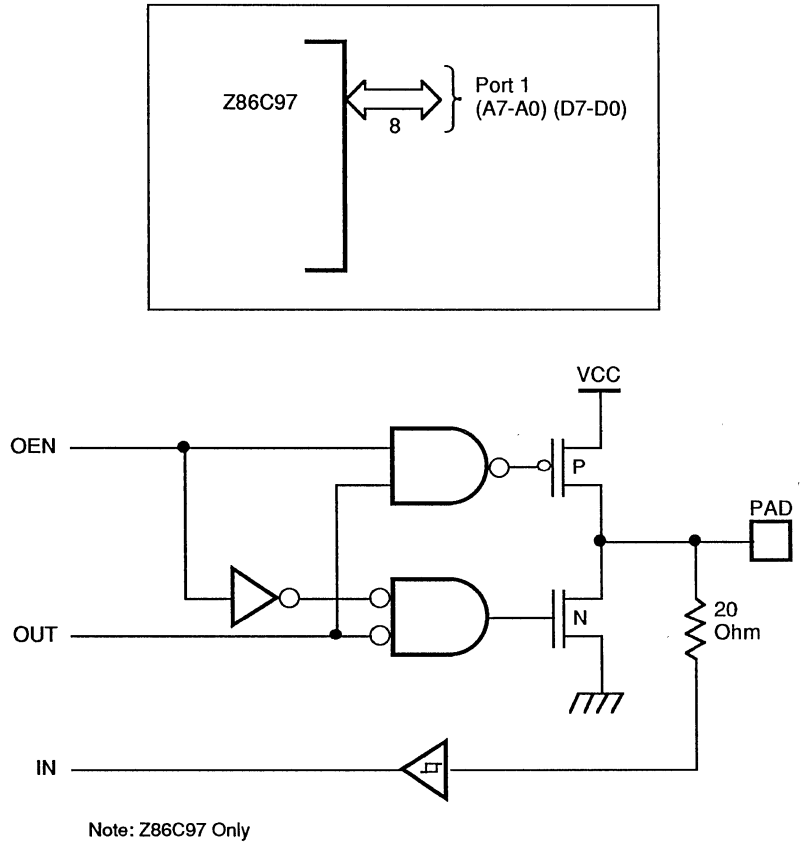
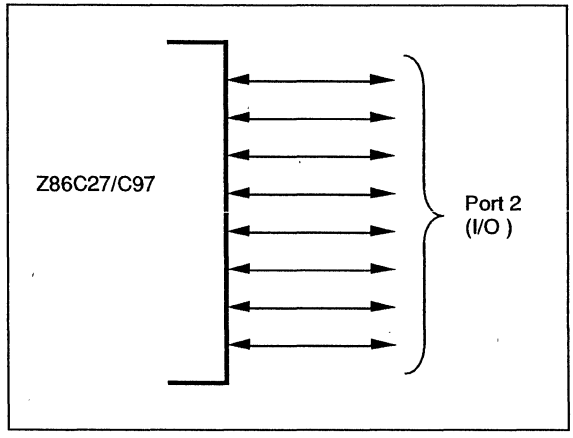
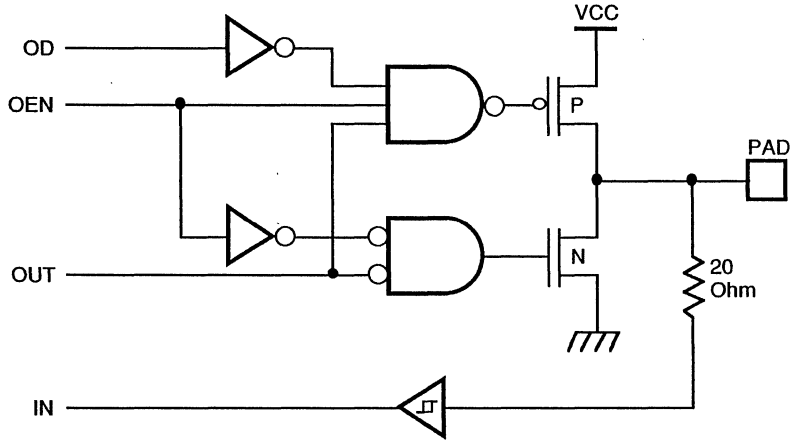


Figure 5. Port 1 Configuration

PIN DESCRIPTION (Continued)

Port 2 (P20-P27). Port 2 is an 8-bit port, CMOS compatible, bit programmable for either input or output. Input buffers are Schmitt triggered. Bits programmed as outputs may be globally programmed as either push-pull or open-drain (Figure 6).

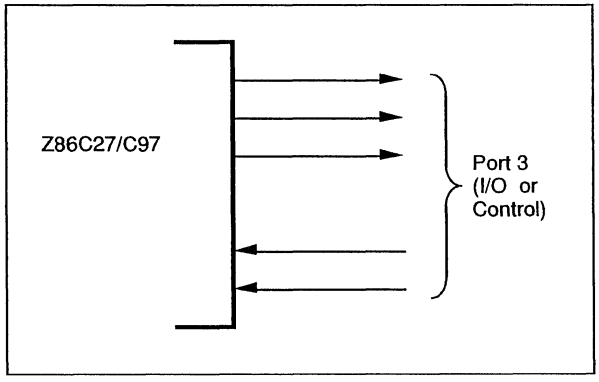


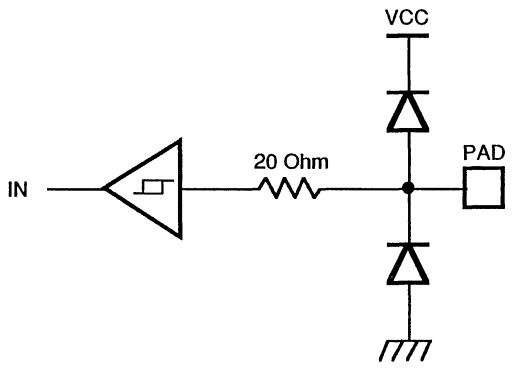


Note: Input/Output, 3-State, Open Drain, Pad Type 5

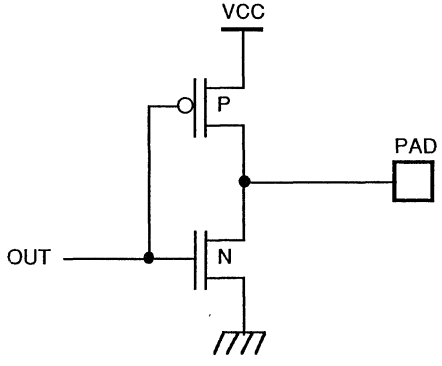
Figure 6. Port 2 Configuration

Port 3 (P30-1, P34-5 and P36). Port 3 Pin P30 input, is read directly. A negative edge event is latched in IRQ3 to initiate an IRQ3 vectored interrupt if appropriately enabled. An application could place the device in STOP mode when P30 goes low (in the IRQ3 interrupt routine). P30 initiates a STOP mode recovery when it subsequently goes high. Port 3, Pin P31 is read directly. A negative edge event is latched in IRQ2 to initiate an IRQ2 vectored interrupt if appropriately enabled. P31 high is signified as the T_{IN} signal to Timer1. Port 3, Pin P34 and Pin P35 are general purpose output lines. Port 3, Pin P36 can be used as a general purpose output or as an output for T_{OUT} (from Timer1 or Timer2) or SCLK (Figure 7).





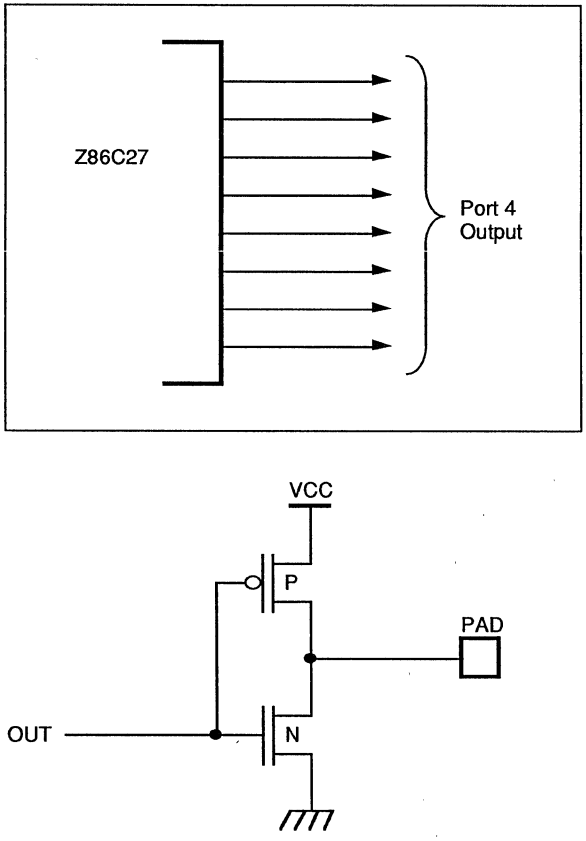
Note: Input Only, Schmitt-triggered, Pad Type 2



Note: Output Only, Pad Type 3

Figure 7. Port 3 Configuration

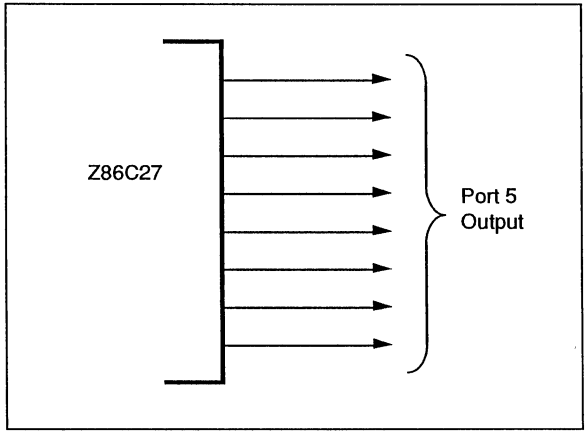
Port 4 (P40-P47). Port 4 is an 8-bit, CMOS compatible, Output Port (Figure 8).

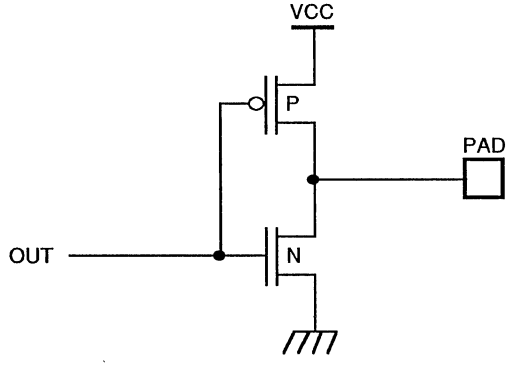


Note: Z86C27 Only

Figure 8. Port 4 Configuration

Port 5 (P50-P57). Port 5 is an 8-bit, CMOS compatible, Output Port. The output ports can directly sink 10 mA at 1.5 Volt V_{OL}. They are typically used to drive multiplexed LED displays (Figure 9).





Note: Z86C27 Only

Figure 9. Port 5 Configuration

PIN DESCRIPTION (Continued)

Port 6 (P60-P65). Port 6 is a 6-bit, Schmitt triggered CMOS compatible, input port. The outputs of the AFC comparators internally feed into the Port 6, bit-6 and bit-7 inputs in ROM mode. In ROMless mode, pins 20 and 21 bring out the internal comparator outputs for Port 6, bit-6 and bit-7 emulation (Figure 10).

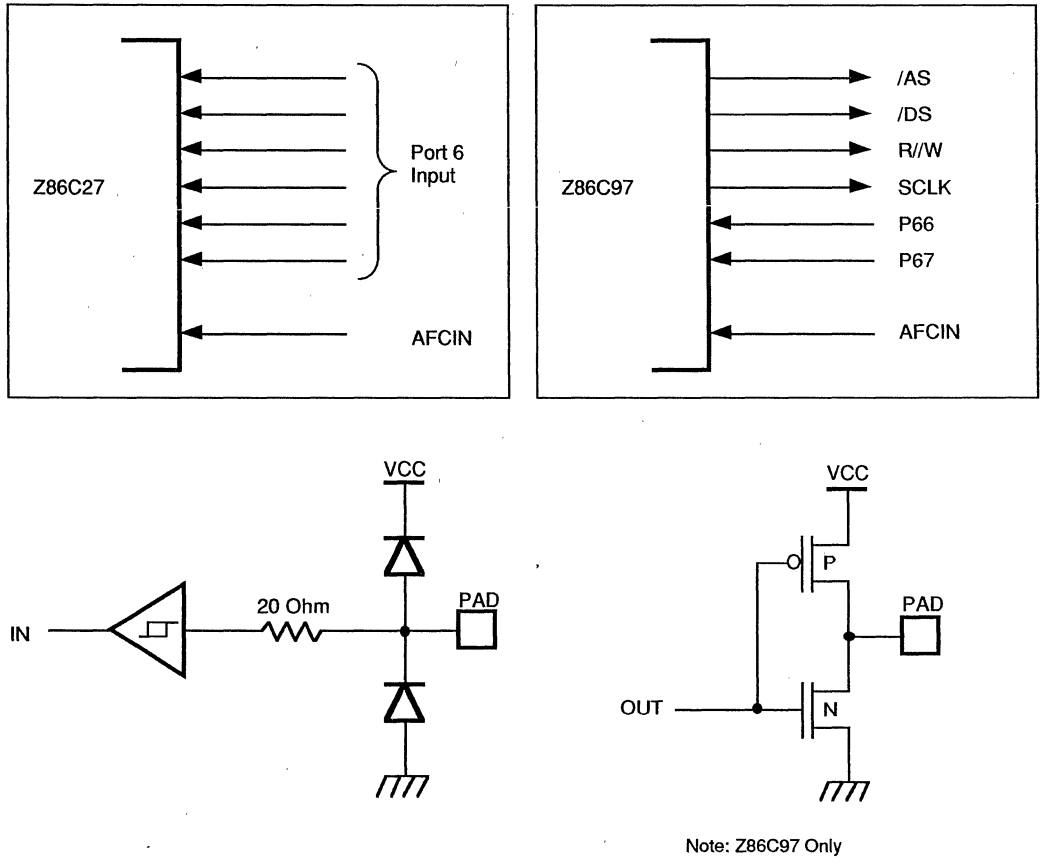


Figure 10. Port 6 Configuration

AFCIN. (Comparator input port, memory mapped). The input signal is supplied to two comparators with VTH1=2/5 V_{cc} and VTH2=3/5 V_{cc} typical threshold voltage. The comparator outputs are internally connected to Port 6, bit-6 and bit-7. AFCIN is typically used to detect AFC voltage level to accommodate digital automatic fine tuning functions. For Z86C97 Port 6, bit-6 and bit-7 are external outputs through pin 20 and pin 21 (Figure 11).

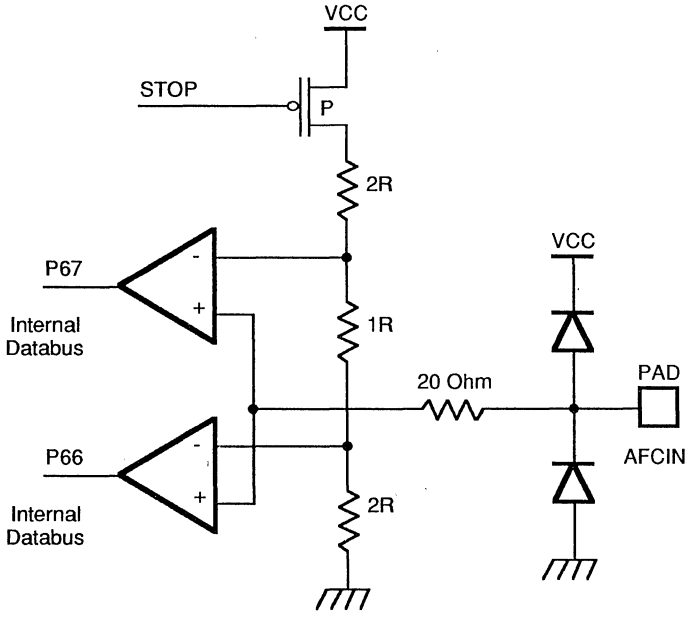


Figure 11. AFCIN Comparator Circuits

PIN DESCRIPTION (Continued)

Pulse Width Modulator 1 (PWM). PWM1 is typically used as the D/A converter for Voltage Synthesis Tuning systems.

Pulse Width Modulator 2-8 (PWM). PWM2-PWM8 are Pulse Width Modulators with 6-bit resolution.

Pulse Width Modulator 9-13 (PWM). PWM9-PWM13 are Pulse Width Modulator circuits with 8-bit resolution or individually programmed as general purpose outputs. In either case, the output drivers are 12-volt open-drain circuits.

/RESET. System Reset. Code is executed from memory address 000C (HEX) after the /RESET pin is set to a high level. The reset function is also carried out by detecting a V_{CC} transition state (automatic power on reset) so that the external reset pin can be permanently tied to V_{CC}. A low level on /RESET forces a restart of the device.

SPECIAL FUNCTIONS

The Z8 DTC incorporates special functions to enhance the Z8's application in consumer, industrial and television control applications.

Pulse Width Modulator (PWM). The DTC has thirteen PWM channels (Figure 12). There are three types of PWM circuits: PWM1 (1 channel of 14-bit resolution) typically used for Voltage Synthesis Tuning, PWM2-PWM8 (7 channels of 6-bit resolution) typically used for audio level control, and PWM9-PWM13 (5 channels of 8-bit resolution) typically used for picture level control. The PWM control registers are mapped into external memory and are accessed via LDE and LDEI instructions.

On Screen Display (OSD). The OSD has a capability of displaying 8 rows by 20 columns of 128 kinds of characters for either high resolution (11x15 dots) or low resolution (5x7 dots) pattern (Figure 13).

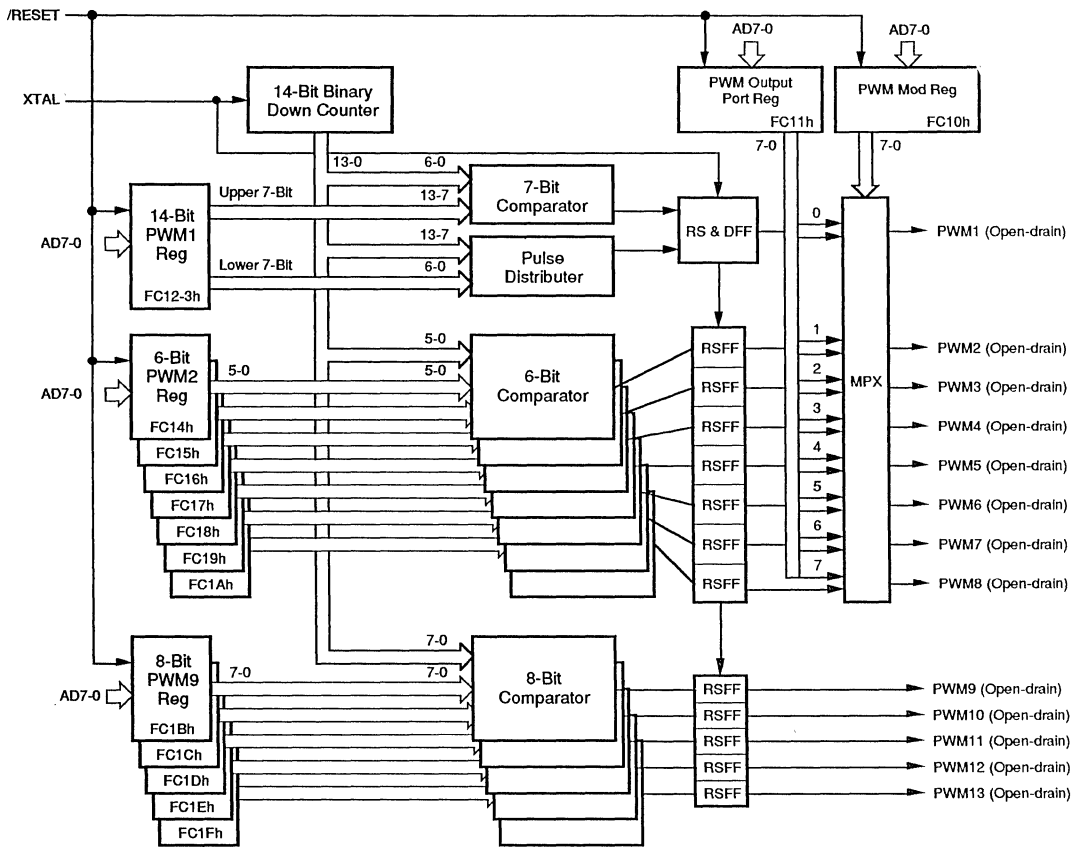


Figure 12. Pulse Width Modulator Block Diagram

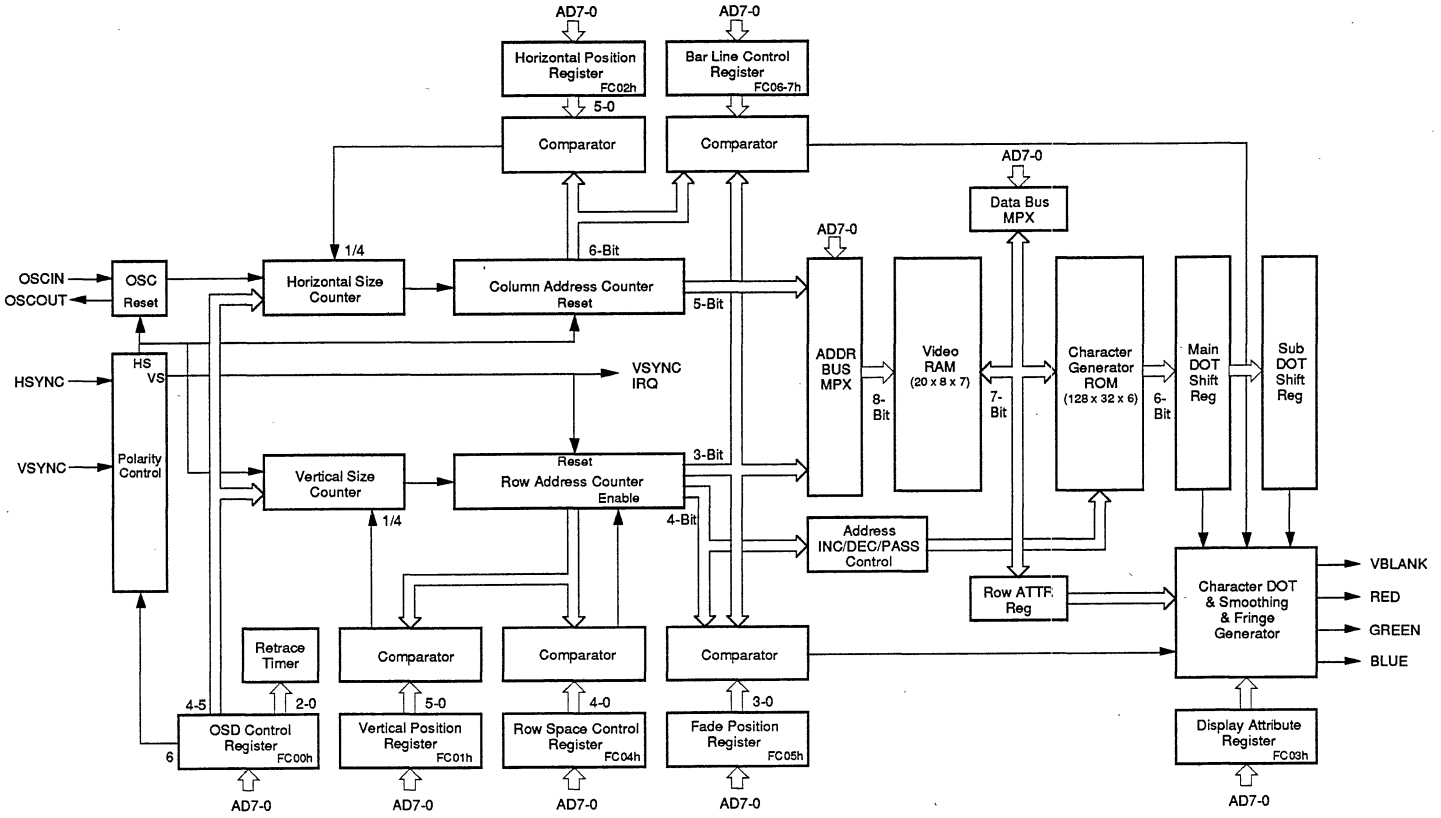


Figure 13. On-Screen-Display Block Diagram

The OSD features are as follows:

- Character Color: Seven kinds of color are specified on a row basis.
- Character Pixel Size: Four character pixel sizes are selected for a low resolution (2HL, 4HL, 6HL and 8HL) and high resolution (1HL, 2HL, 3HL and 4HL) Horizontal Line (HL).
- Polarity Selections: Can select active low or high for horizontal/vertical sync input and RGB outputs.
- Display Position: Can display 64 vertical positions by 4HL units and 64 horizontal positions by a 4 dot clock.
- Inter Row Spacing: Inter row vertical line spacing is set from 2HL to 25HL (17HL for high resolution).
- Fade In/Out Control: Fade position can be determined in vertical direction.
- Bar Line Type Display: One of the rows is selected to display an analog bar line every half column by setting second color with proper character set.
- Fringe Function: Fringe off/on and the color selected.
- Background Color: Eight kinds of color including black background color.
- ON/OFF Control: Character display, backgrounds are turned on and off.
- Number of Display Characters: 8 rows x 20 columns.
- Character Set: 128 (5x7 dots or 11x15 dots).

Character Generator ROM. The character generator ROM is organized as 4 Kbytes of 6 bits. The ROM defines either 11x15 dot (high resolution) or 5x7 (low resolution) characters (Figure 14).

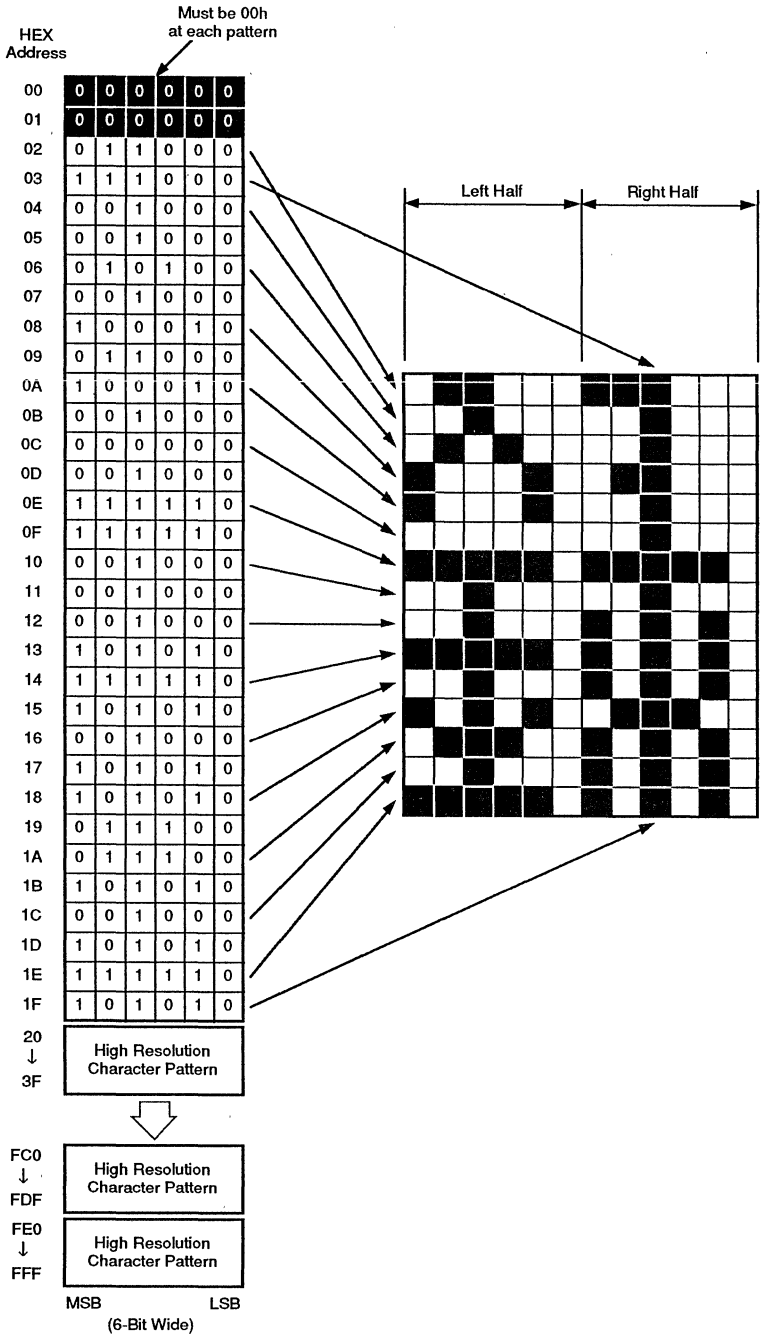


Figure 14a. High and Low Resolution Character ROM Configuration

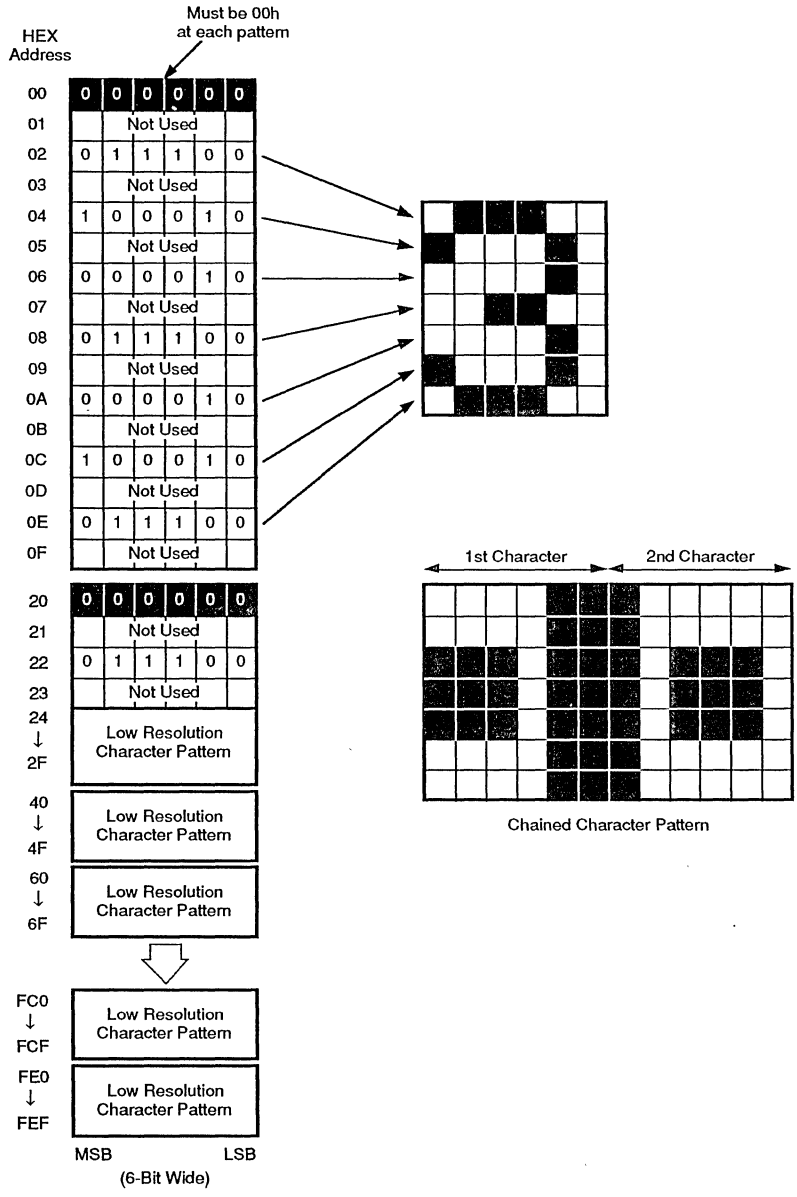


Figure 14b. High and Low Resolution Character ROM Configuration

SPECIAL FUNCTIONS (Continued)

Program Memory. The program ROM size is 8K bytes (Figure 15). The IRQ vector table is located in the lower address space. The vector address is fetched after the corresponding interrupt and program control is passed to the specified vector address. IRQ1 vector is fixed to VSYNC interrupt request and occurs at the leading edge of the filtered VSYNC input. Program memory start at address 000C (HEX) after reset.

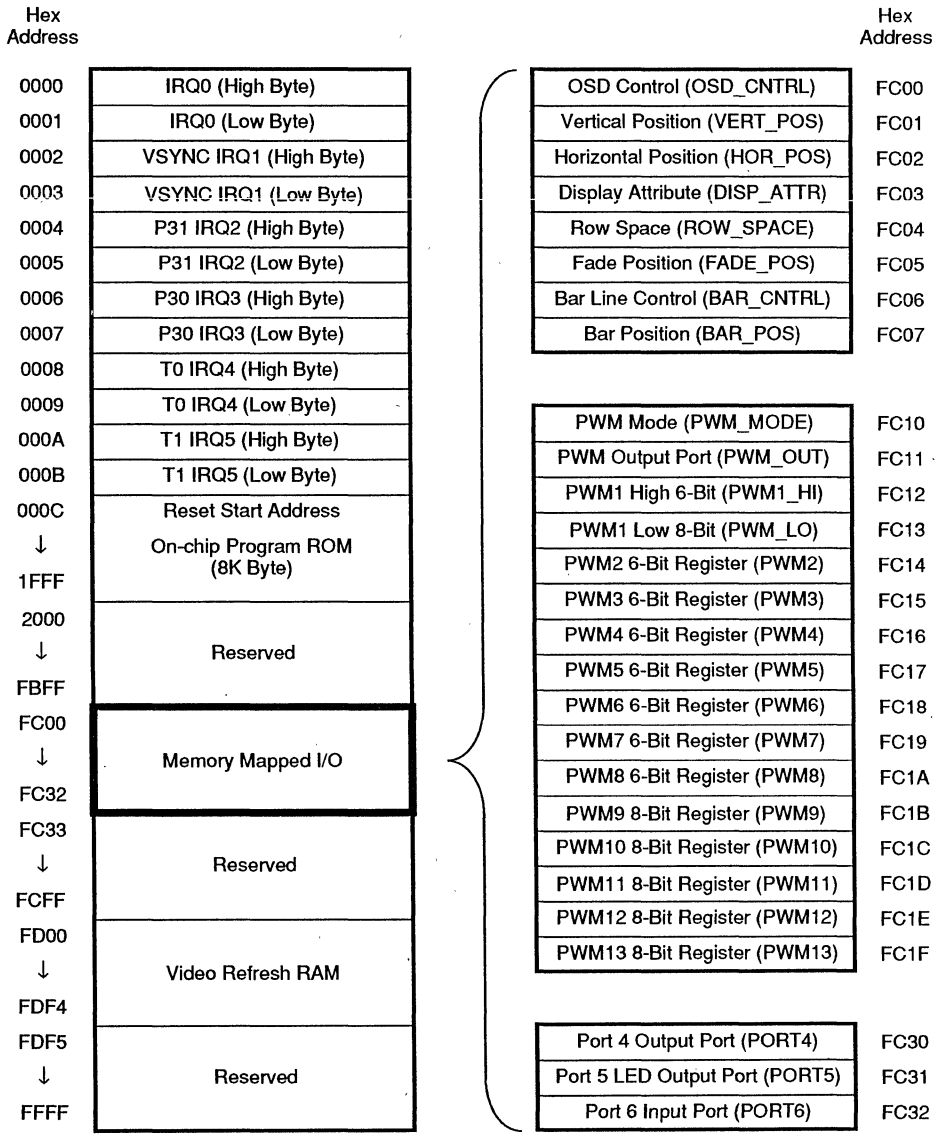


Figure 15. Program Memory

Memory Mapped Register. All control registers and I/O ports (except Port 2 and Port 3) are assigned to program memory space. Address space FC00 (HEX) contains OSD control registers, PWM output registers and Ports 4, 5 and 6 I/O registers. Two bits of the decoded AFCIN port are assigned to Port 6 input port. LDE and LDEI instructions are required to transfer data between the Register File and the Memory Mapped Registers.

Data Memory (/DM). The Z86C27/C97 can address up to 64K bytes of program memory, and 56K bytes of external data memory. External data memory may be included with or separated from the external program memory space. /DM, an optional I/O signal that can be programmed to appear on Port 3 Pin P34, distinguishes between data and program memory space.

Register File. A total of 253 byte registers are implemented in the Z8 core. Address 00 (HEX), 01 (HEX) and FO (HEX) are reserved. The register file consists of 2 I/O Port registers, 236 general-purpose registers and 15 control and status registers (Figure 16). The instructions can access registers directly or indirectly with an 8-bit address field. This also allows short 4-bit register addressing using the Register Pointer. In the 4-bit mode, the register file is divided into sixteen working-register groups, each occupying 16 continuous locations. The Register Pointer addresses the starting location of the active working-register group (Figure 17).

Note: Register Bank E0-EF is only accessed through a working register and indirect addressing modes.

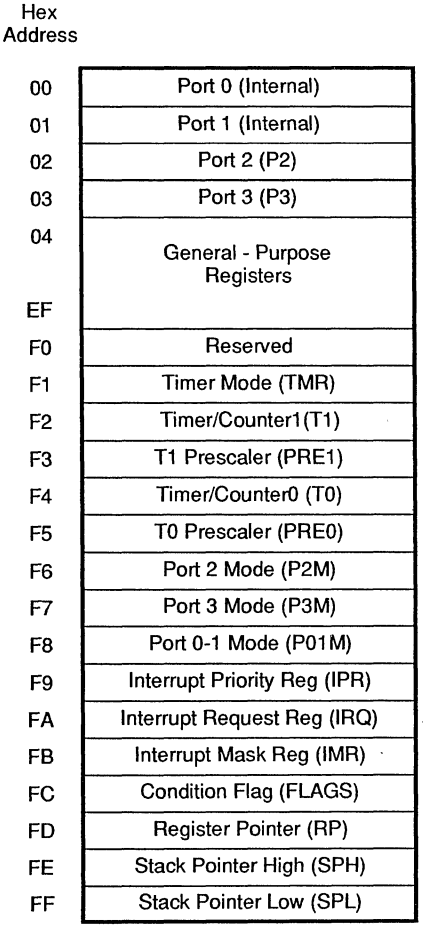


Figure 16. Register File Configuration

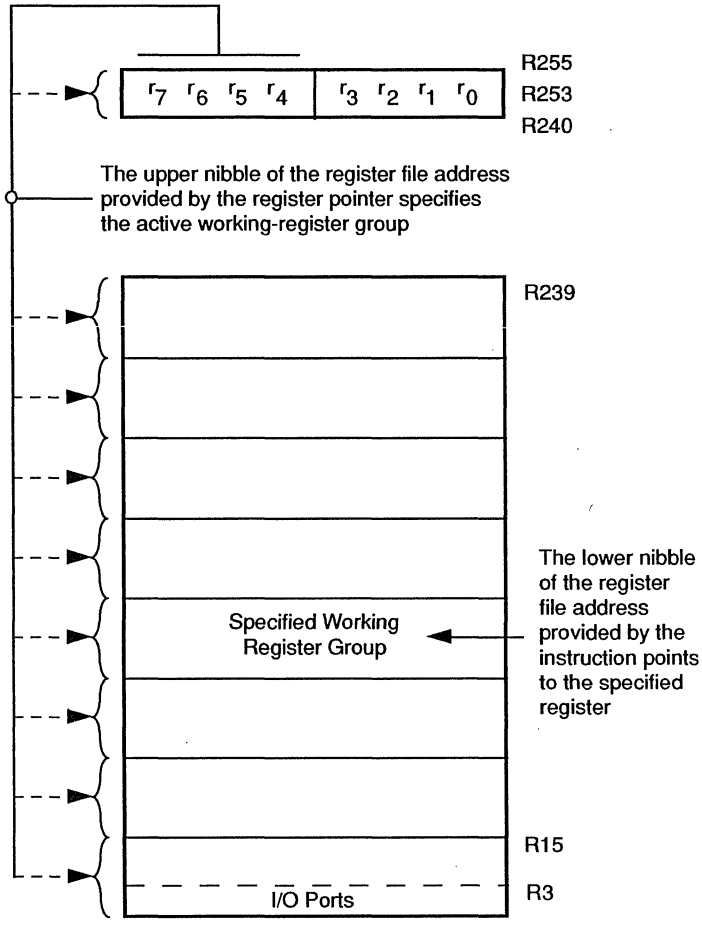


Figure 17. Register Pointer

Stack. Either the internal register file or the external data memory is used for the stack. A 16-bit Stack Pointer is used for the external stack, which can reside anywhere in data memory. An 8-bit Stack Pointer is used for the internal stack that resides within the 236 general-purpose registers.

Counter/Timers. There are two 8-bit programmable counter/timers (T0-T1), each driven by its own 6-bit programmable prescaler (PRE0 and PRE1). The T1 prescaler can be driven by internal or external clock sources; however, the T0 prescaler is driven by the internal clock only (Figure 18).

The counter, but not the prescalers, are read at any time without disturbing their value or count mode. The clock source for T1 is user-definable and is the internal microprocessor clock (XTAL clock/4), or an external signal input via Port 3, P31. The counter/timers are programmably cascaded by connecting the T0 output to the input of T1.

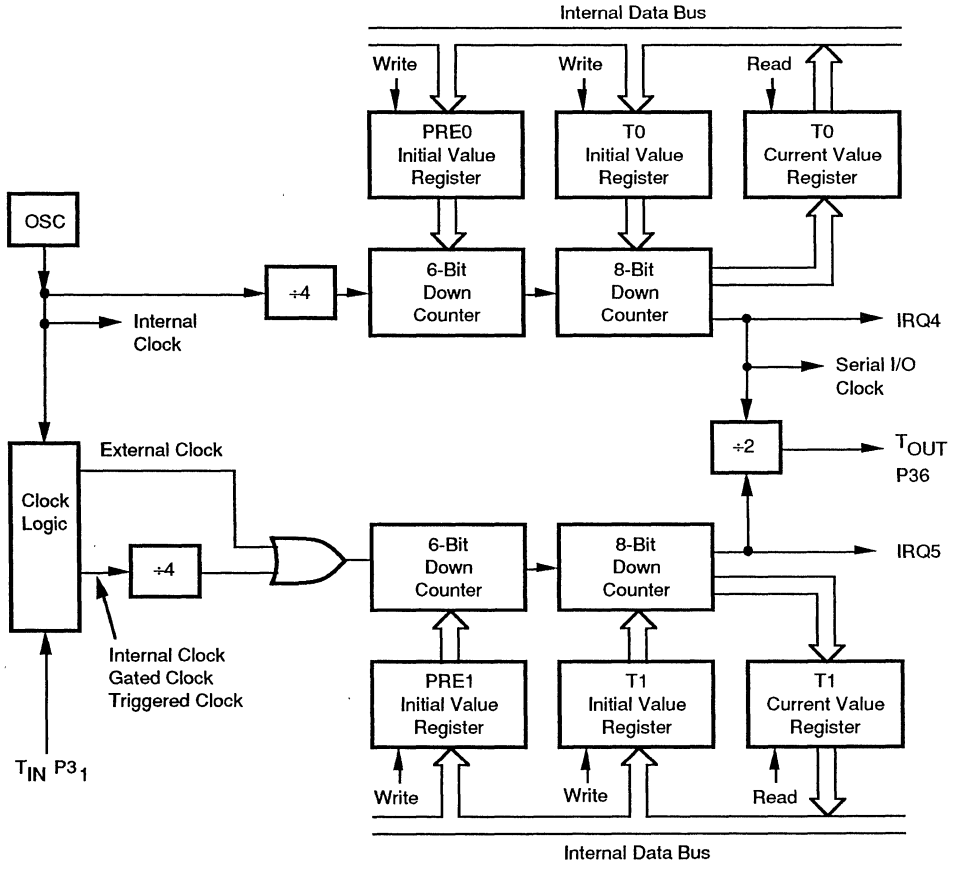


Figure 18. Counter/Timer Block Diagram

Interrupts. The DTC has six different interrupts from six different sources. These interrupts are maskable and prioritized (Figure 19). The six sources are divided as follows: two sources are claimed by Port 3 (P30, P31), one by VSYNC, two by the counter/timers, and one is software triggered only.

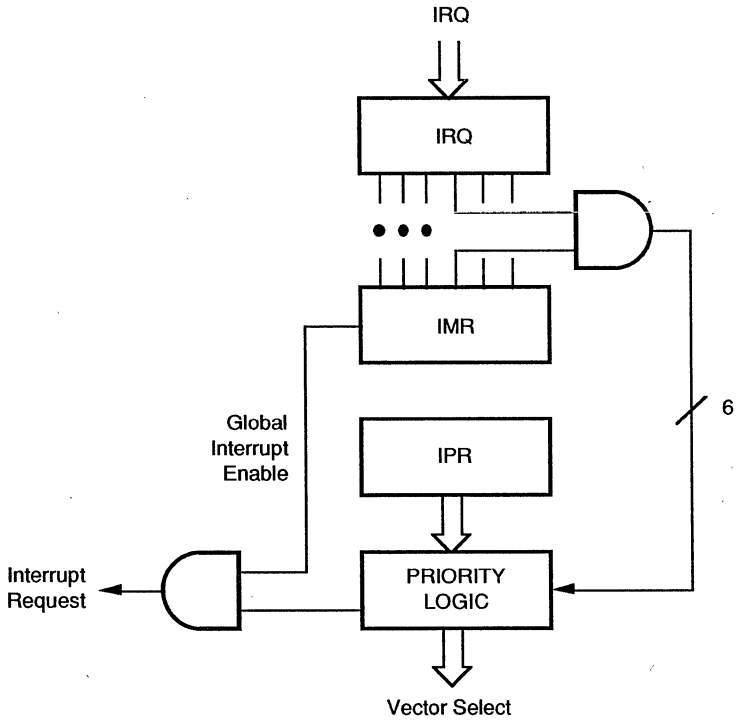


Figure 19. Interrupt Block Diagram

HALT Mode. The Z86C27/C97 is driven by two internal clocks, TCLK and SCLK, They both oscillate at the crystal frequency. TCLK provides the clock signal for the counter-timers and the interrupt block. SCLK provides the clock signal for all other CPU blocks. Halt mode turns off the internal CPU clock (SCLK), but not the XTAL oscillation. The counter/timers and external interrupts remain active. The device may be recovered by interrupts, either external or internally generated.

STOP Mode. The STOP instruction stops crystal oscillation, thereby stopping both SCLK and TCLK. The device ceases to operate. The STOP mode can be released by two methods. The first method is to reset the device. A high input condition on Port 3 Pin P30 is the second method. After releasing the STOP mode by using either one of the two methods, program execution begins at location %000C (HEX). To complete an instruction prior to entering the standby modes, a NOP instruction has to be placed before the HALT or STOP instructions. This is required because of instruction pipelining. i.e.:

```
FF NOP    ; clear the pipeline
6F STOP   ; enter STOP mode
          or
FF NOP    ; clear the pipeline
7F HALT   ; enter HALT mode
```

Notes:
In STOP mode, XTAL2 pin has an internal pull-up on it and OSCOUT has an internal pull-down.

Clock. The Z86C27/C97 on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal, ceramic resonator, or any suitable external clock source (XTAL1 = Input, XTAL2 = Output). The crystal is an AT cut, parallel resonant, 4 MHz max with a series resistance (RS) less than or equal to 100 Ohms.

The crystal source is connected across XTAL1 AND XTAL2 using the recommended capacitors (10 pF < CL < 300 pF, where C1=C2=CL) from each pin to ground (Figure 20).

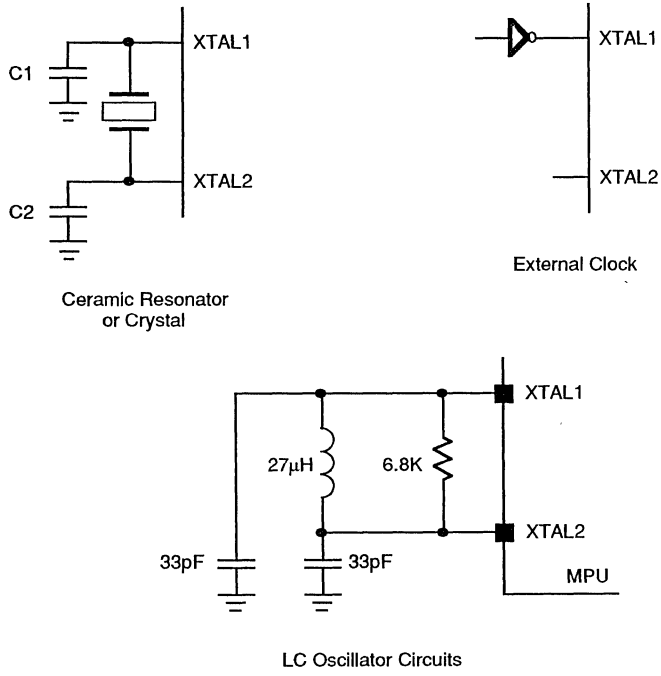


Figure 20. Oscillator Configuration

Watch Dog Timer (WDT). The Z86C27/C97 is equipped with a watch dog timer which should be refreshed within 12 ms. Failure to refresh the timer results in a reset of the device. The WDT is enabled the first time that a WDT 5F (HEX) instruction is executed. Every subsequent WDT instruction retriggers the timer. The watch dog timer may or may not be enabled during the HALT mode. The instruction WDH 4F (HEX) enables the timer during HALT mode. If the HALT mode is not released and the watch dog timer is not retriggered (by the WDT instruction) within 12 ms, a device reset occurs.

V_{cc} Voltage Sensitive Reset (VSR). Reset is globally driven if V_{cc} is below the specified voltage (Figure 21).

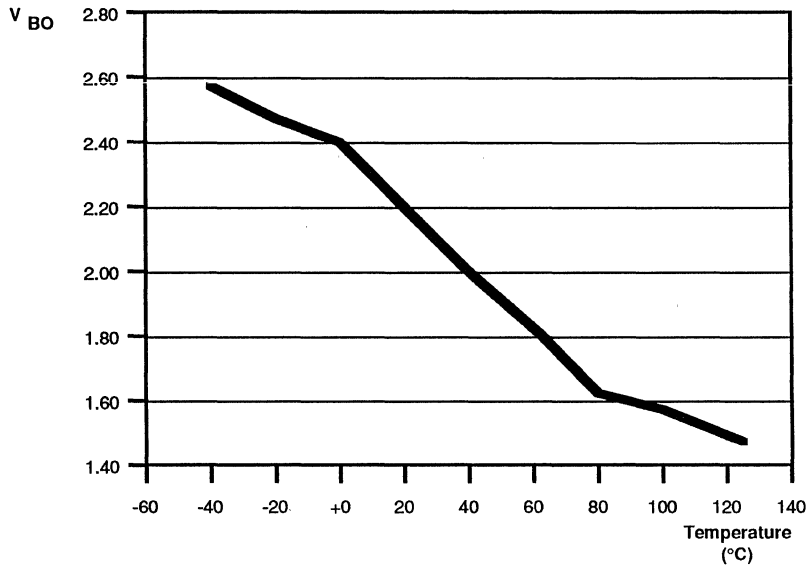


Figure 21. Voltage Sensitive Reset Vs Temperature

ABSOLUTE MAXIMUM RATINGS

Stress greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Symbol	Parameters	Min	Max	Units	Notes
V_{CC}	Power Supply Voltage †	-0.3	+7	V	
V_I	Input Voltage	-0.3	V_{CC}+0.3	V	
V_I	Input Voltage	-0.3	V_{CC}+0.3	V	[1]
V_O	Output Voltage	-0.3	V_{CC}+8.0	V	[2]
I_{OH}	Output Current High		-10	mA	1 pin
I_{OH}	Output Current High		-100	mA	all total
I_{OL}	Output Current Low		20	mA	1 pin
I_{OL}	Output Current Low		40	mA	[3] (1 pin)
I_{OL}	Output Current Low,all total		200	mA	
T_A	Operating Temperature	††			
T_{STG}	Storage Temperature	-65	+150	C	

Notes:
[1] Port 2 open-drain
[2] PWM open drain outputs
[3] Port 5

† Voltage on all pins with respect to GND.
†† See Ordering Information

STANDARD TEST CONDITIONS

The characteristics listed below apply for standard test conditions as noted. All voltages are referenced to GND. Positive current flows into the referenced pin (Figure 22).

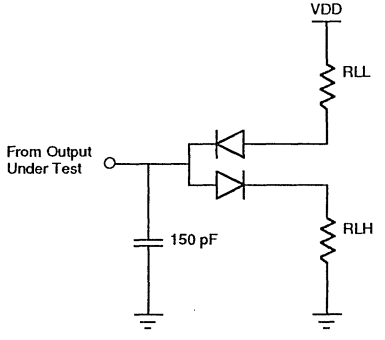


Figure 22. Test Load Diagram

CAPACITANCE

T_A=25°C, V_{CC}=GND=0V, Freq=1.0 MHz, unmeasured pins to GND.

Parameter	Max	Units
Input capacitance	10	pF
Output capacitance	20	pF
I/O capacitance	25	pF
AFCin input capacitance	10	pF

DC CHARACTERISTICS

T_A=0°C to +70°C; V_{CC}=+4.5V to +5.5V; F_{OSC}=4 MHz

Symbol	Parameter	T_A=0°C to 70°C Min	Max	Typical @ 25°C	Units	Conditions
V_{IL}	Input Voltage Low	0	0.2 V_{CC}	1.48	V	
V_{ILC}	Input XTAL/Osc In Low		0.07 V_{CC}	0.98	V	External Clock Generator Driven
V_{IH}	Input Voltage XTAL/Osc In High	0.7 V_{CC}	V_{CC}	3.2	V	External Clock Generator Driven
V_{IHC}	Input XTAL/Osc in High	0.8 V_{CC}	V_{CC}	3.0	V	External Clock Generator Driven
V_{HY}	Schmitt Hysteresis	0.1 V_{CC}		0.8	V	
V_{PU}	Maximum Pull-up Voltage		12		V	[2]
V_{OL}	Output Voltage Low		0.4	0.16	V	I_{OL}=1.00mA
			0.4	0.19	V	I_{OL}=3.2mA, [1]
			0.4	0.19	V	I_{OL}=0.75mA [2]
			1.5	1.00	V	I_{OL}=10mA [1]
V_{00-01}	AFC Level 01 In		0.45 V_{CC}	1.9	V	
V_{01-11}	AFC Level 11 In	0.5 V_{CC}	0.75 V_{CC}	3.12	V	
V_{OH}	Output Voltage High	V_{CC}-0.4		4.75	V	I_{OH}=-0.75mA
I_{IR}	Reset Input Current		-80	-46	µA	V_{RL}=0V
I_{IL}	Input Leakage	-3.0	3.0	0.01	µA	0V,V_{CC}
I_{OL}	Tri-State Leakage	-3.0	3.0	0.02	µA	0V,V_{CC}
I_{CC}	Supply Current		20	13.2	mA	All inputs at rail
I_{CC1}			6	3.2	mA	All inputs at rail
I_{CC2}			10	0	µA	All inputs at rail

Notes:
[1] Port 5
[2] PWM Open Drain

AC CHARACTERISTICS

Timing Diagrams

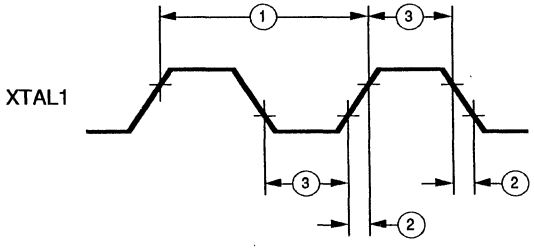


Figure 23. External Clock

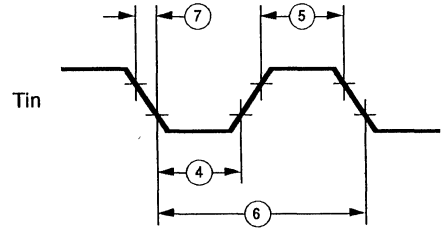


Figure 24. Counter Timer

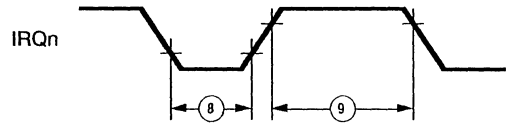


Figure 25. Interrupt Request

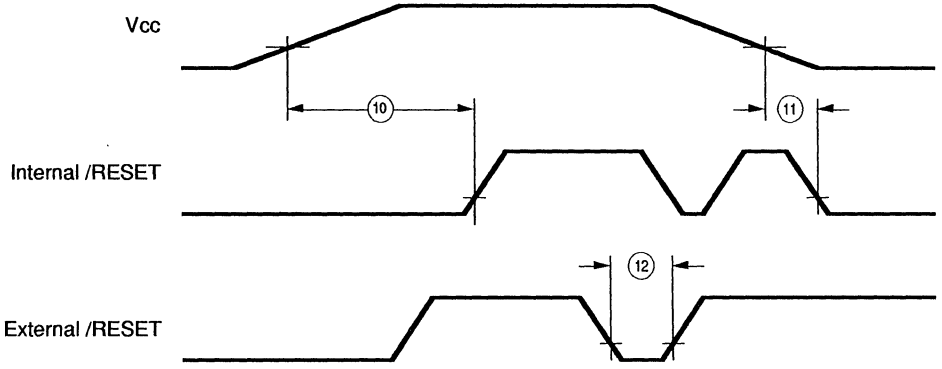


Figure 26. Power On Reset

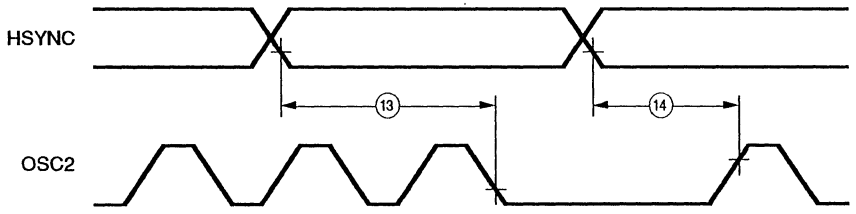


Figure 27. On Screen Display

AC CHARACTERISTICS

T_A=0° C to 70° C; V_{CC}=+4.5V to +5.5V; F_{OSC}=4 MHz,

No	Symbol	Parameter	Min	Max	Unit
1	TpC	Input clock period	250	1000	ns
2	TrC,TfC	Clock input raise and fall		15	ns
3	TwC	Input clock width	70		ns
4	TwTinL	Timer input low width	70		ns
5	TwTinH	Timer input high width	3TpC		
6	TpTin	Timer input period	8TpC		
7	TrTin,TfTin	Timer input raise and fall		100	ns
8A	TwIL	Int req input low	70		ns
8B	TwIL		3TpC		
9	TwIH	Int request input high	3TpC		
10	TdPOR	Power On Reset delay	25	100	ms
11	TdLVIRES	Low voltage detect to In-Internal RESET condition	200		ns
12	TwRES	Reset minimum width	5TpC		
13	TdHsOl	Hsync start to Vosc stop	2TpV	3TpV	
14	TdHsOh	Hsync end to Vosc start		1TpV	
15	TdWDT	WDT Refresh Time		12	ms

Notes:
[1] Refer to DC Characteristics for details on switching levels.

* Units in nanoseconds

AC CHARACTERISTICS

Unique to Z86C97 External Memory Read/Write Timing Diagram

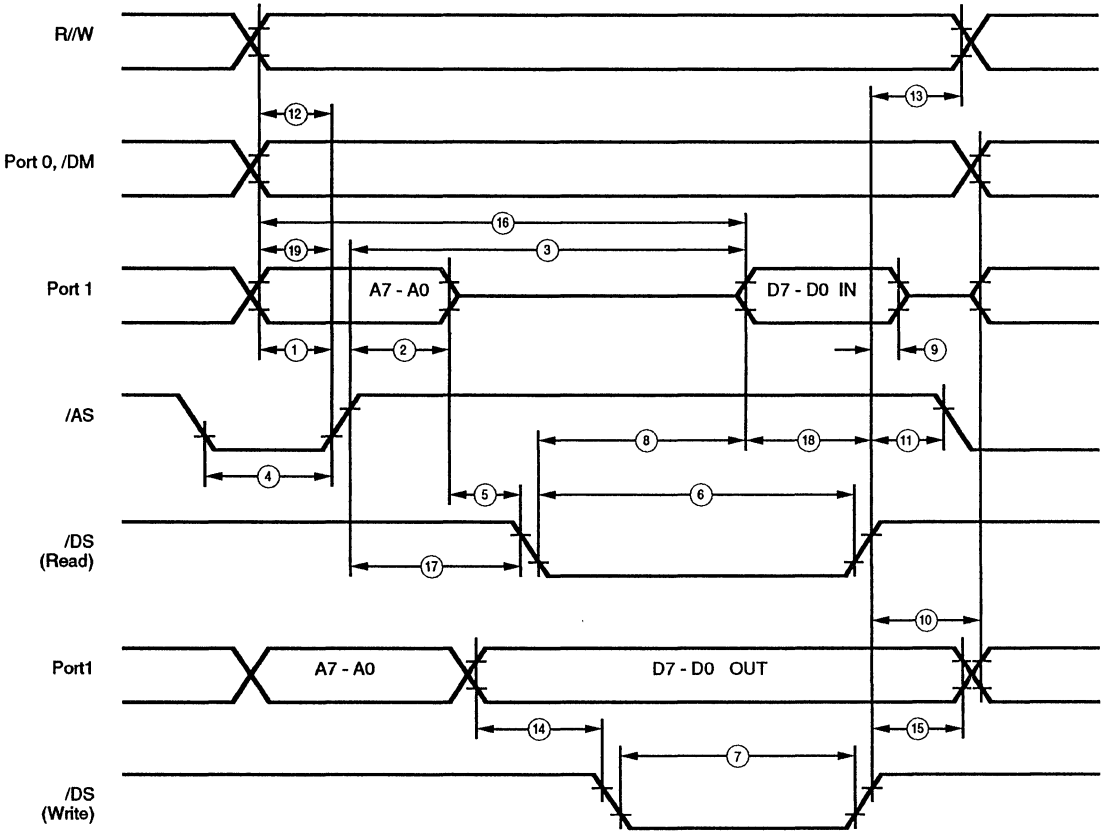


Figure 28. Z86C97 External Memory Read/Write Timing

AC CHARACTERISTICS

Unique to Z86C97, T_A=0°C to 70°C; V_{cc}=+4.5V to +5.5V; F_{osc} = 4 MHz

No	Symbol	Parameter	Min	Max	Unit	Notes
1	TdA(AS)	Address Valid to /AS High Delay	35		ns	[2]
2	TdAS(AS)	/AS High to Address Float Delay	45		ns	[2]
3	TdAS(DR)	/AS High to Read Data Required Valid		250	ns	[1,2]
4	TwAS	/AS Low Width	55		ns	[2]
5	TdAZ(DS)	Address Float to /DS Low	0		ns	[2]
6	TwDSR	/DS (Read) Low Width	185		ns	[1,2]
7	TwDSW	DS (Write) Low Width	110		ns	[1,2]
8	TdDSR(DR)	/DS Low to Read Data Required Valid		130	ns	[1,2]
9	ThDR(DS)	Read Data to /DS High Hold		5	ns	
10	TdDS(A)	/DS High to Address Active Delay	55		ns	[2]
11	TdDS(AS)	/DS High to /AS Low Delay	55		ns	[2]
12	TdR/W(AS)	R//W Valid to /AS High Delay	35		ns	[2]
13	TdDS(R/W)	/DS High to R//W Not Valid	55		ns	[2]
14	TdDW(DSW)	Write Data Valid to /DS Low Delay	35		ns	[2]
15	TdDS(DW)	/DS High to Write Data Not Valid	55		ns	[2]
16	TdA(DR)	Address Valid to Read Data Required Valid		330	ns	[1,2]
17	TdAS(DS)	/AS High to /DS Low Delay	65		ns	[2]
18	TdDI(DS)	Data Input Setup to /DS High	75		ns	[1]

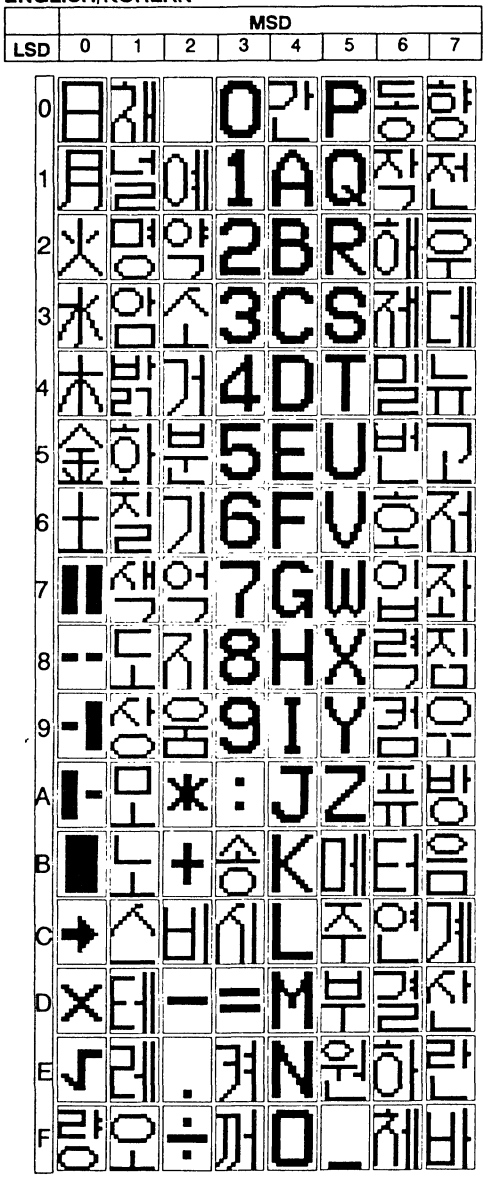
Notes:

[1] When using extended memory timing, for parameters 3, 6, 7, 8, and 16, add 2TpC (250 ns @ 4.0 MHz).

[2] Min and Max times are in nanoseconds unless otherwise noted.

STANDARD CHARACTER SETS

ENGLISH/KOREAN



SUMMARY

Input/Output Circuits

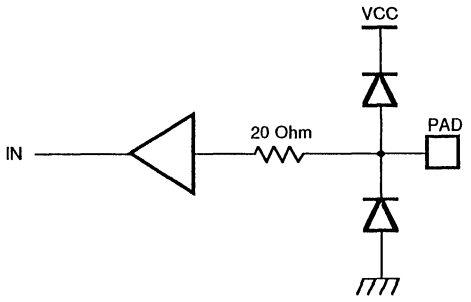


Figure 29. Input Only (Pad Type 1)

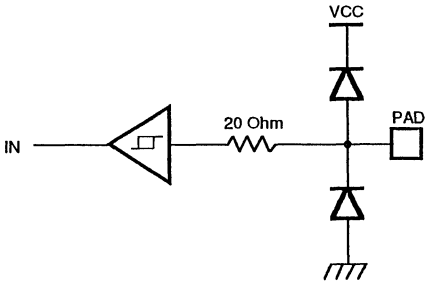


Figure 30. Input Only, Schmitt Triggered (Pad Type 2)

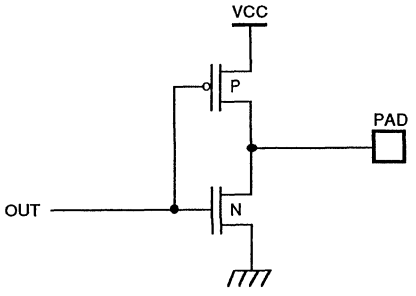


Figure 31. Output Only (Pad Type 3)

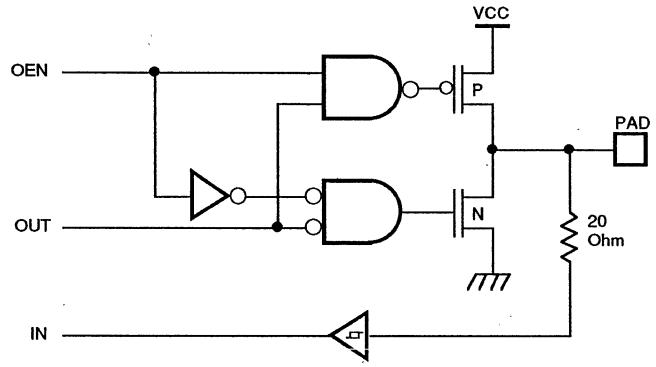


Figure 32. Input/Output 3-State (Pad Type 4)

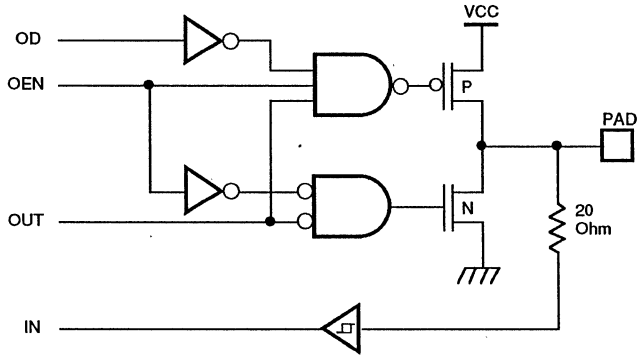


Figure 33. Input/Output, 3-state, Open Drain (Pad Type 5)

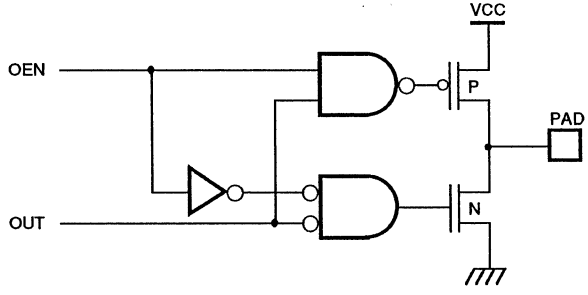


Figure 34. Output Only, 3-State (Pad Type 6)

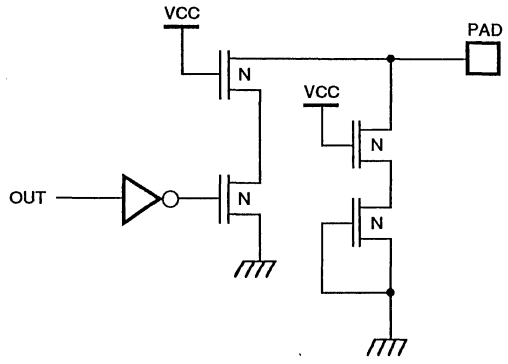


Figure 35. Output Only, 12-Volt Open Drain (Pad Type 7)

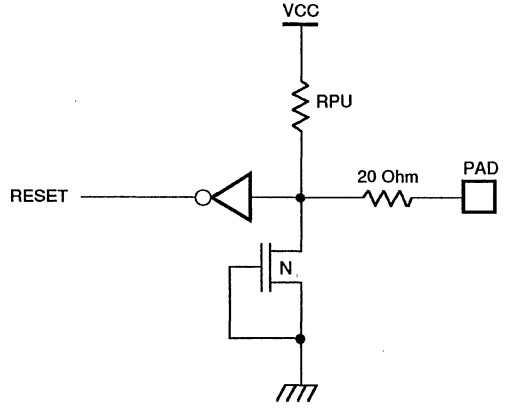


Figure 36. Reset Input Circuit (Pad Type 8)

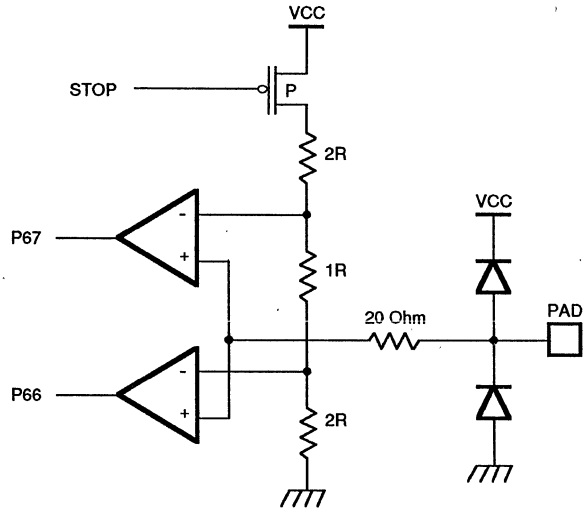


Figure 37. AFC Input Circuit (Pad Type 9)

Mapping of Symbolic Pad Types to Pin Functions

Pin Name	Pad Type	Notes
XTAL1, OSC_{IN} XTAL2, OSC_{OUT}	1	High gain start, low gain run amplifier circuit
/RESET	8	
P00-07	6	Z86C97 only
P10-17	4	Z86C97 only
P20-P27	5	
P30-P31	2	
P34-P36	3	
P40-P47	3	Z86C27 only
P50-P57	3	Z86C27 only
P60-P65	2	Z86C27 only
P66-P67	3	Z86C97 only
/AS, /DS, R/W, SCLK	3	Z86C97 only
AFCIN	9	
PWM1-PWM13	7	
HSYNC, VSYNC	2	
VRED, VBLUE, VGREEN, VBLANK	3	

DTC CONTROL REGISTER DIAGRAMS
Port Registers

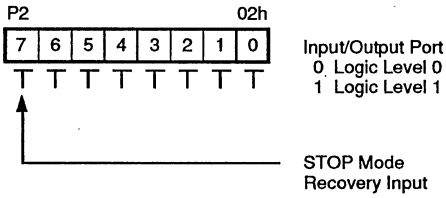


Figure 38. Port 2 Register

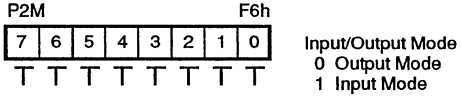


Figure 39. Port 2 Mode Register

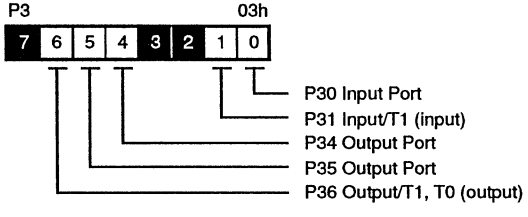


Figure 40. Port 3 Register

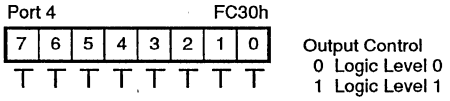


Figure 41. Port 4 Register

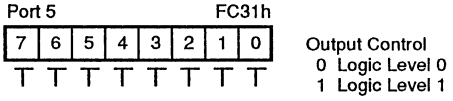


Figure 42. Port 5 Register

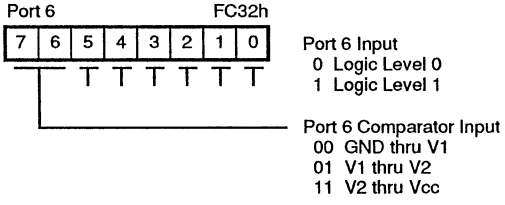


Figure 43. Port 6 Register

DTC CONTROL REGISTER DIAGRAMS
PWM Registers

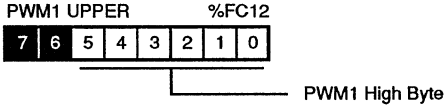


Figure 44. PWM 1 High Value

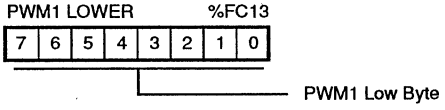


Figure 45. PWM 1 Low Value

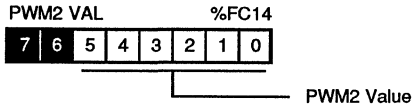


Figure 46. PWM 2 Value

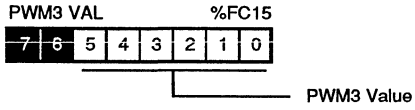


Figure 47. PWM 3 Value

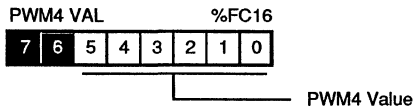


Figure 48. PWM 4 Value

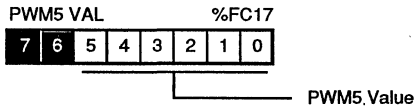


Figure 49. PWM 5 Value

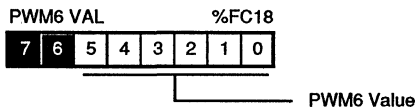


Figure 50. PWM 6 Value

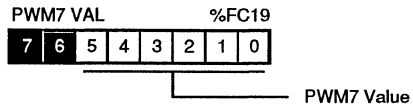


Figure 51. PWM 7 Value

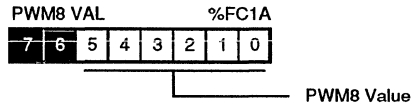


Figure 52. PWM 8 Value

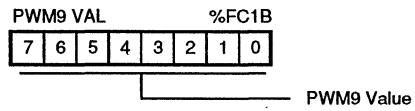


Figure 53. PWM 9 Value

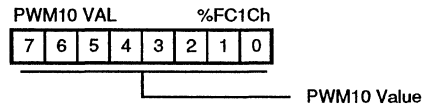


Figure 54. PWM 10 Value

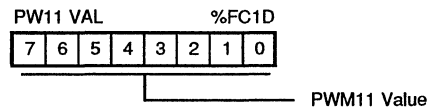


Figure 55. PWM 11 Value

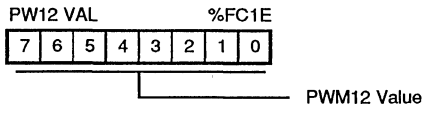


Figure 56. PWM 12 Value

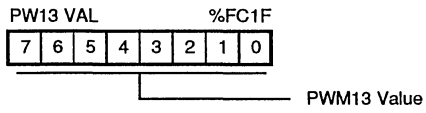


Figure 57. PWM 13 Value Register

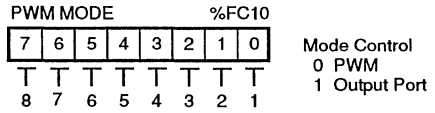


Figure 58. PWM Mode Register

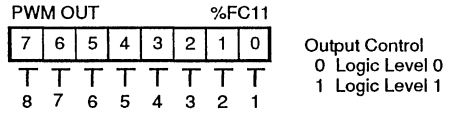


Figure 59. PWM Port Output Register

DTC CONTROL REGISTER DIAGRAMS

OSD Registers

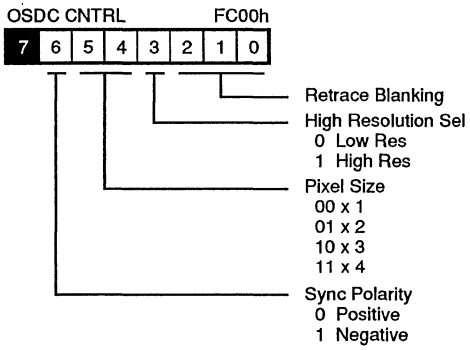


Figure 60. OSD Control Register

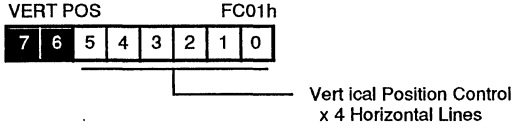


Figure 61. OSD Vertical Position Register

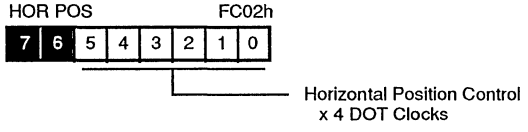


Figure 62. OSD Horizontal Position Register

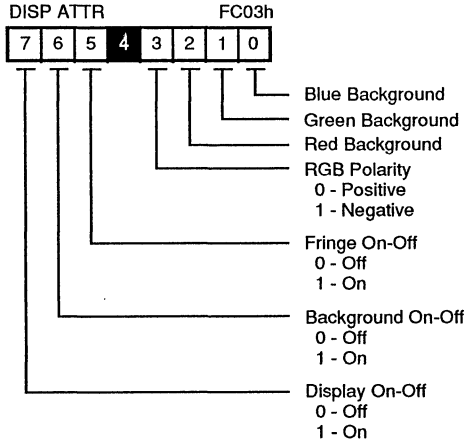


Figure 63. OSD Display Attribute Register

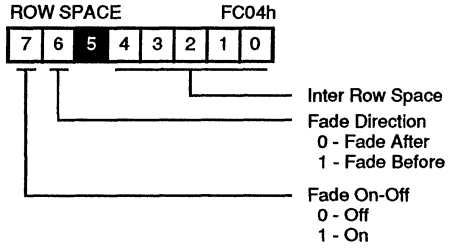


Figure 64. OSD Row Space Register

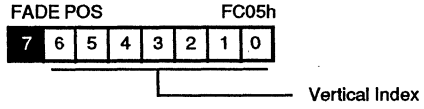


Figure 65. OSD Fade Position Register

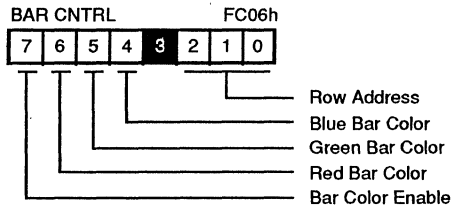


Figure 66. OSD Bar Control Register

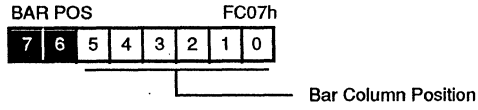


Figure 67. OSD Bar Position Register

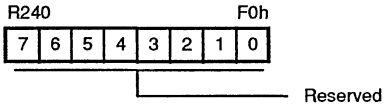


Figure 68. Reserved (F0H)

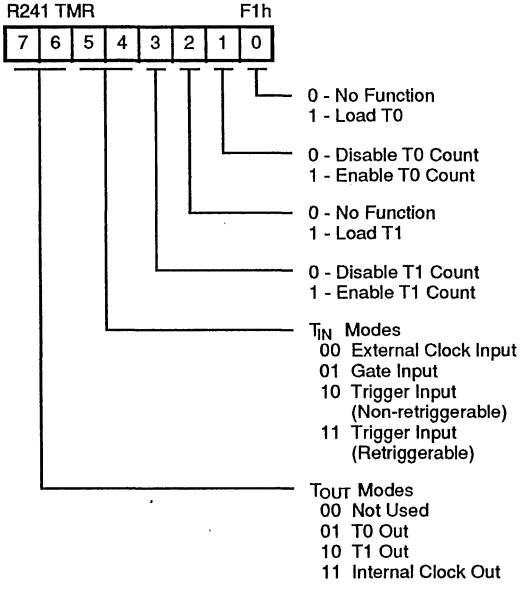


Figure 69. Timer Mode Register (F1H; Read/Write)

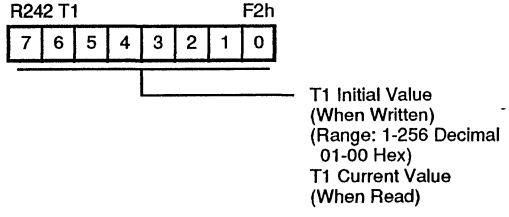


Figure 70. Counter Timer 1 Register (F1H; Read/Write)

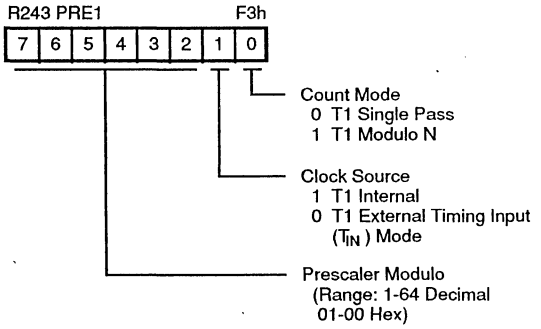


Figure 71. Prescaler 1 Register (F3H; Write Only)

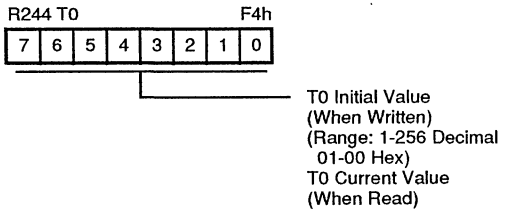


Figure 72. Counter/Timer 0 Register (F4H; Read/Write)

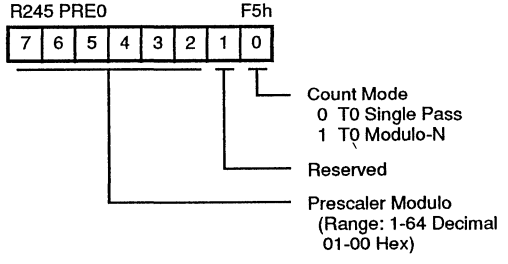


Figure 73 Prescaler 0 Register (F5H; Write Only)

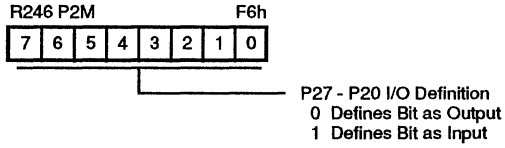


Figure 74. Port 2 Mode Register (F6H; Write Only)

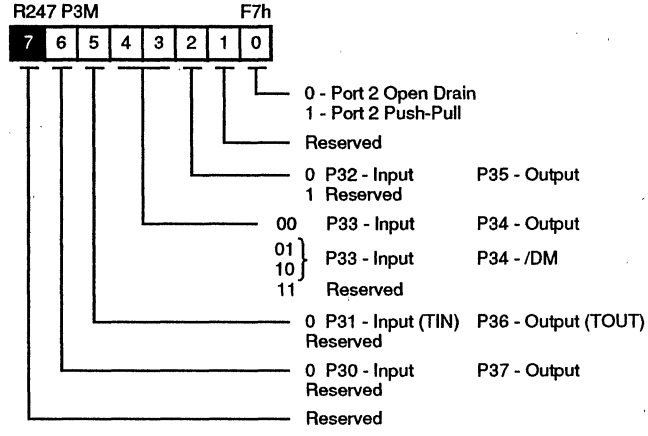


Figure 75. Port 3 Mode Register (F7H; Write Only)

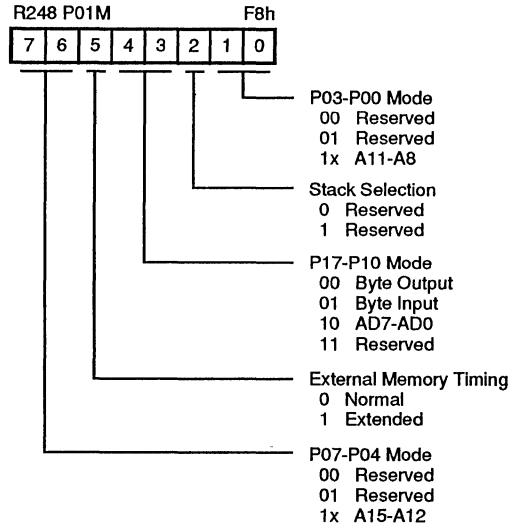


Figure 76. Port 0 and 1 Mode Register (F8H; Write Only)

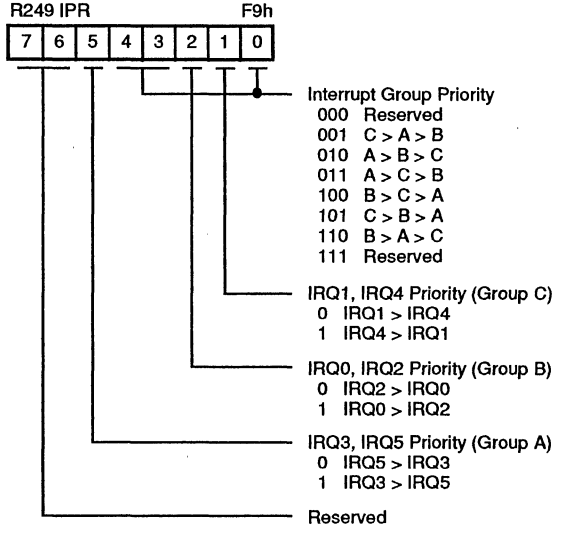


Figure 77. Interrupt Priority Register (F9H; Write Only)

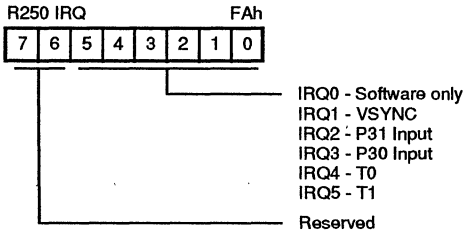


Figure 78. Interrupt Request Register (FAH; Read/Write)

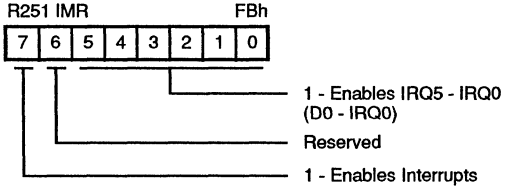


Figure 79. Interrupt Mask Register (FBH; Read/Write)

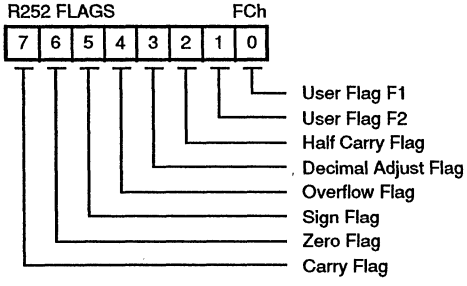


Figure 80. Flag Register (FCH; Read/Write)

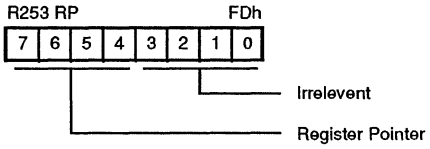


Figure 81. Register Pointer (FDH; Read/Write)

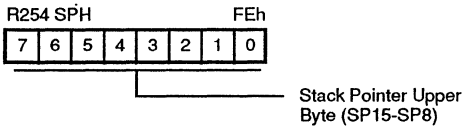


Figure 82. Stack Pointer (FEH; Read/Write)

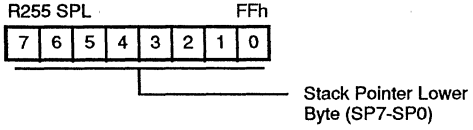


Figure 83. Stack Pointer (FFH; Read/Write)

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

Symbol	Meaning
IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

Symbol	Meaning
dst	Destination location or contents
src	Source location or contents
cc	Condition code
@	Indirect address prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag register (Control Register 252)
RP	Register Pointer (R253)
IMR	Interrupt mask register (R251)

Flags. Control register (R252) contains the following six flags:

Symbol	Meaning
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

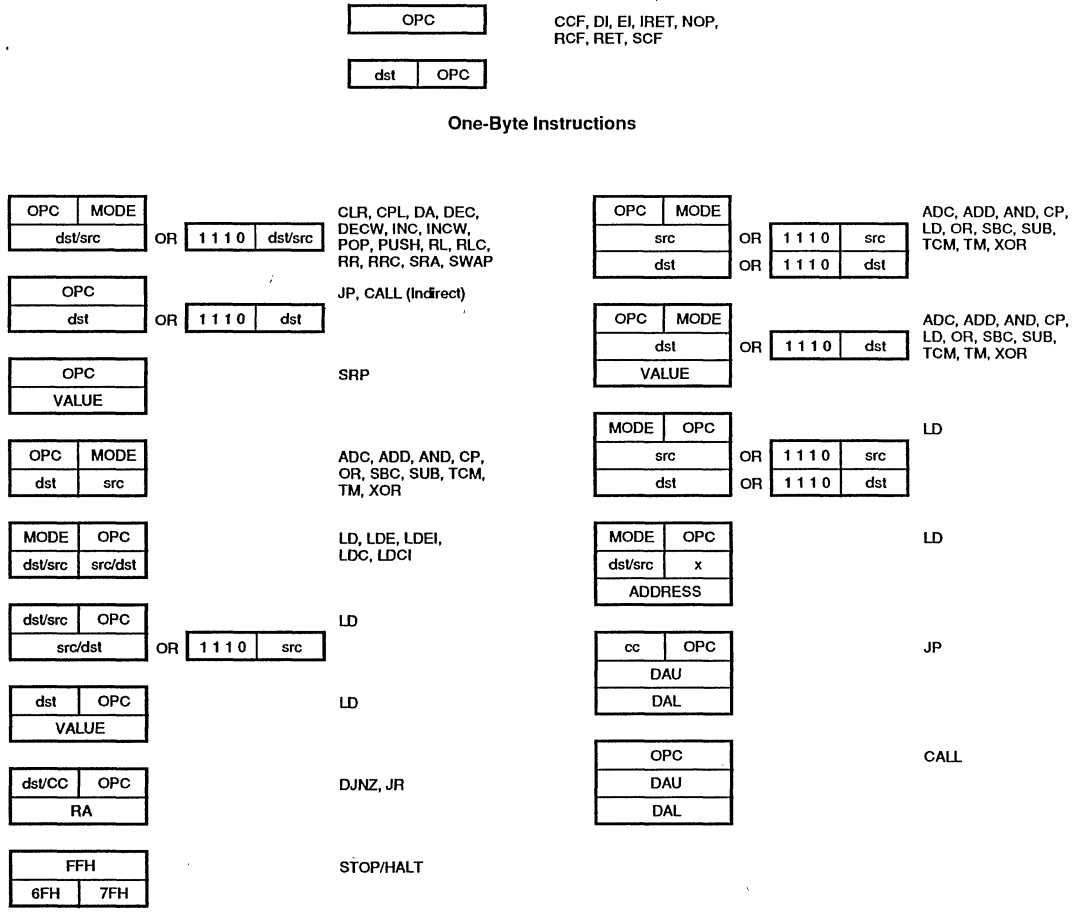
Affected flages are indicated by:

0	Clear to zero
1	Set to one
*	Set to clear according to operation
-	Unaffected
x	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000		Always True	
0111	C	Carry	C = 1
1111	NC	No Carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not Zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No Overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not Equal	Z = 0
1001	GE	Greater Than or Equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater Than	[Z OR (S XOR V)] = 0
0010	LE	Less Than or Equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned Greater Than or Equal	C = 0
0111	ULT	Unsigned Less Than	C = 1
1011	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned Less Than or Equal	(C OR Z) = 1
0000		Never True	

INSTRUCTION FORMATS



INSTRUCTION SUMMARY

Note: Assignment of a value is indicated by the symbol " ← ". For example:

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr (n)" is used to refer to bit (n) of a given operand location. For example:

dst (7)

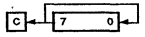
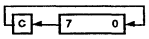
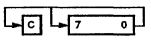
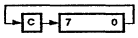
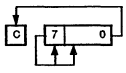
refers to bit 7 of the destination operand.

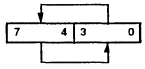
INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst, src dst←dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst←dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst←dst AND src	†		5[ ]	-	*	*	0	-	-
CALL dst SP←SP - 2 @SP←PC, PC←dst	DA IRR		D6 D4	-	-	-	-	-	-
CCF C←NOT C			EF	*	-	-	-	-	-
CLR dst dst←0	R IR		B0 B1	-	-	-	-	-	-
COM dst dst←NOT dst	R IR		60 61	-	*	*	0	-	-
CP dst, src dst - src	†		A[ ]	*	*	*	*	-	-
DA dst dst←DA dst	R IR		40 41	*	*	*	X	-	-
DEC dst dst←dst - 1	R IR		00 01	-	*	*	*	-	-
DECW dst dst←dst - 1	RR IR		80 81	-	*	*	*	-	-
DI IMR(7)←0			8F	-	-	-	-	-	-
DJNZr, dst r←r - 1 if r ≠ 0 PC←PC + dst Range: +127, -128	RA		rA r = 0 - F	-	-	-	-	-	-
EI IMR(7)←1			9F	-	-	-	-	-	-
HALT			7F	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
INC dst dst←dst + 1	r R IR		rE r = 0 - F 20 21	-	*	*	*	-	-
INCW dst dst←dst + 1	RR IR		A0 A1	-	*	*	*	-	-
IRET FLAGS←@SP; SP←SP + 1 PC←@SP; SP←SP + 2; IMR(7)←1			BF	*	*	*	*	*	*
JP cc, dst if cc is true PC←dst	DA IRR		cD c = 0 - F 30	-	-	-	-	-	-
JR cc, dst if cc is true, PC←PC + dst Range: +127, -128	RA		cB c = 0 - F	-	-	-	-	-	-
LD dst, src dst←src	r r R r X r r Ir R R IR IR	Im R r X r Ir r R IR IM IM R	rC r8 r9 r = 0 - F C7 D7 E3 F3 E4 E5 E6 E7 F5	-	-	-	-	-	-
LDC dst, src	r	Irr	C2	-	-	-	-	-	-
LDCI dst, src dst←src r←r +1; rr←rr + 1	Ir	Irr	C3	-	-	-	-	-	-

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
NOP			FF	-	-	-	-	-	-
OR dst, src dst←dst OR src	†		4[ ]	-	*	*	0	-	-
POP dst dst←@SP; SP←SP + 1	R IR		50 51	-	-	-	-	-	-
PUSH src SP←SP - 1; @SP←src		R IR	70 71	-	-	-	-	-	-
RCF C←0			CF	0	-	-	-	-	-
RET PC←@SP; SP←SP + 2			AF	-	-	-	-	-	-
RL dst 	R IR		90 91	*	*	*	*	-	-
RLC dst 	R IR		10 11	*	*	*	*	-	-
RR dst 	R IR		E0 E1	*	*	*	*	-	-
RRC dst 	R IR		C0 C1	*	*	*	*	-	-
SBC dst, src dst←dst←src←C	†		3[ ]	*	*	*	*	1	*
SCF C←1			DF	1	-	-	-	-	-
SRA dst 	R IR		D0 D1	*	*	*	0	-	-
SRP src RP←src		Im	31	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
STOP			6F	-	-	-	-	-	-
SUB dst, src dst←dst←src	†		2[ ]	*	*	*	*	1	*
SWAP dst 	R IR		F0 F1	X	*	*	X	-	-
TCM dst, src (NOT dst) AND src	†		6[ ]	-	*	*	0	-	-
TM dst, src dst AND src	†		7[ ]	-	*	*	0	-	-
XOR dst, src dst←dst XOR src	†		B[ ]	-	*	*	0	-	-

† These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

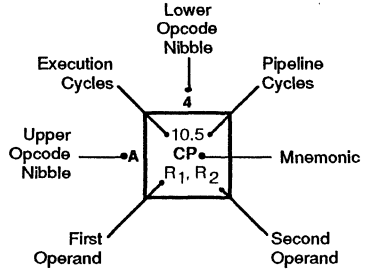
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address Mode dst	Address Mode src	Lower Opcode Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

OPCODE MAP

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								6.0 **WDH**
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								6.0 **WDT**
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								6.0 **STOP**
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								7.0 **HALT**
8	10.5 **DECW** RR1	10.5 **DECW** IR1	12.0 **LDE** r1, Irr2	18.0 **LDEI** Ir1, Irr2												6.1 **DI**
9	6.5 **RL** R1	6.5 **RL** IR1	12.0 **LDE** r2, Irr1	18.0 **LDEI** Ir2, Irr1												6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14.0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2				10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1	12.0 **LDC** r2, Irr1	18.0 **LDCI** Ir2, Irr1	20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6.5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2			3					2					3	1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_2 = Dst address
R_1 or r_2 = Src address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank are not defined.

* 2-byte instruction appears as a 3-byte instruction

ZiLOG

ADVANCE INFORMATION SPECIFICATION

Z86127

LOW-COST DIGITAL TELEVISION CONTROLLER (LDTC)

FEATURES

8-bit CMOS microcontroller for consumer television applications, 64-pin DIP package.

- Low cost
- Low power consumption
- Fast instruction pointer -1.5 microseconds @ 4 MHz
- Two standby modes - STOP and HALT
- Low voltage detection/voltage sensitive reset
- 35 input/output lines
- Port 2 (8-bit programmable I/O) and Port 3 (2-bit input, 3-bit output) register mapped ports.
- Port 5 (8-bit LED drive output) and Port 6 (6-bit input and tri-state comparator AFC input) memory mapped I/O ports.
- All digital CMOS levels Schmitt triggered
- 8 Kbytes of ROM
- 236 bytes of RAM
- Two programmable 8-bit Counter/Timers each with 6-bit programmable prescaler.
- Six vectored, priority interrupts from six different sources
- Clock speed up to 4 MHz
- On-chip oscillator that accepts a crystal, ceramic resonator, LC or external clock drive.
- Watch Dog/Power-On Reset Timer

On Screen Display Controller

- 4K x 6-bit character generator ROM
- 160 x 7-bit video RAM
- Mask programmable 128 character set displayed in an 8-row by 20-column format, 12 by 15 pixel character cell, capable of supporting English, Korean, Chinese and Japanese high resolution characters.
- Fully programmable color attributes including row character, row background/fringes, frame background/position, bar graph color change, and character size.
- Programmable display position and character size control.
- One Pulse Width Modulator (14-bit resolution) for voltage synthesis tuner control.
- Five Pulse Width Modulators (8-bit resolution) for picture control.
- Three Pulse Width Modulators (6-bit resolution) for audio control.

GENERAL DESCRIPTION

The Z86127 Low-Cost Digital Television Controller (LDTC) introduce a new level of sophistication to single-chip architecture. The Z86127 is a member of the Z8® single-chip microcontroller family with 8 Kbytes of ROM and 236 bytes of RAM. The device is housed in a 64-pin DIP package, in which only 52 are active, and are CMOS compatible. The LDTC offers mask programmed ROM which enables the Z8 microcontroller to be used in a high volume production application device embedded with a custom program (customer supplied program).

GENERAL DESCRIPTION (Continued)

Zilog's LDTC offers fast execution, efficient use of memory, sophisticated interrupts, input/output bit manipulation capabilities, and easy hardware/software system expansion along with low cost and low power consumption. The device provides an ideal performance and reliability solution for consumer and industrial television applications.

The Z86127 architecture is characterized by utilizing Zilog's advanced Superintegration™ design methodology. The devices have an 8-bit internal data path controlled by a Z8 microcontroller, and On Screen Display (OSD) logic circuits/Pulse Width Modulators (PWM). On-chip peripherals include two register mapped I/O ports (Ports 2 and Port 3), Interrupt control logic (1 software, 2 external and 3 internal interrupts) and a standby mode recovery input port (Port 3, pin P30).

The OSD control circuits support 8 rows by 20 columns of characters. The character color is specified by row. One of the 8 rows is assigned to show two kinds of colors for bar type displays such as volume control. The OSD is capable of displaying either low resolution (5x7 dot pattern) or high resolution (11x15 dot pattern) characters. The Z86C97 currently supports high resolution characters only.

A 14-bit PWM port provides enough voltage resolution for a voltage synthesizer tuning system. Three 6-bit PWM ports are used for controlling audio signal level. Five 8-bit PWM ports are used to vary picture levels.

The LDTC applications demand powerful I/O capabilities. The Z86127 fulfills this with 27 I/O pins dedicated to input and output. These lines are grouped into four ports, and are configurable under software control to provide timing, status signals, parallel I/O and an address/data bus for interfacing to external memory.

There are three basic address spaces available to support this wide range of configurations: Program Memory, Video RAM, and Register File .The Register File is composed of 236 bytes of general purpose registers, two I/O Port registers, 15 control and status registers and 3 reserved registers.

To unburden the program from coping with the real-time problems such as counting/timing and data communication, the LDTC's offer's two on-chip counter/timers with a large number of user selectable modes (Figure 1).

Note: All Signals with a preceding front slash, "/", are active Low, e.g.: B/W (WORD is active Low); /B/W (BYTE is active Low, only)

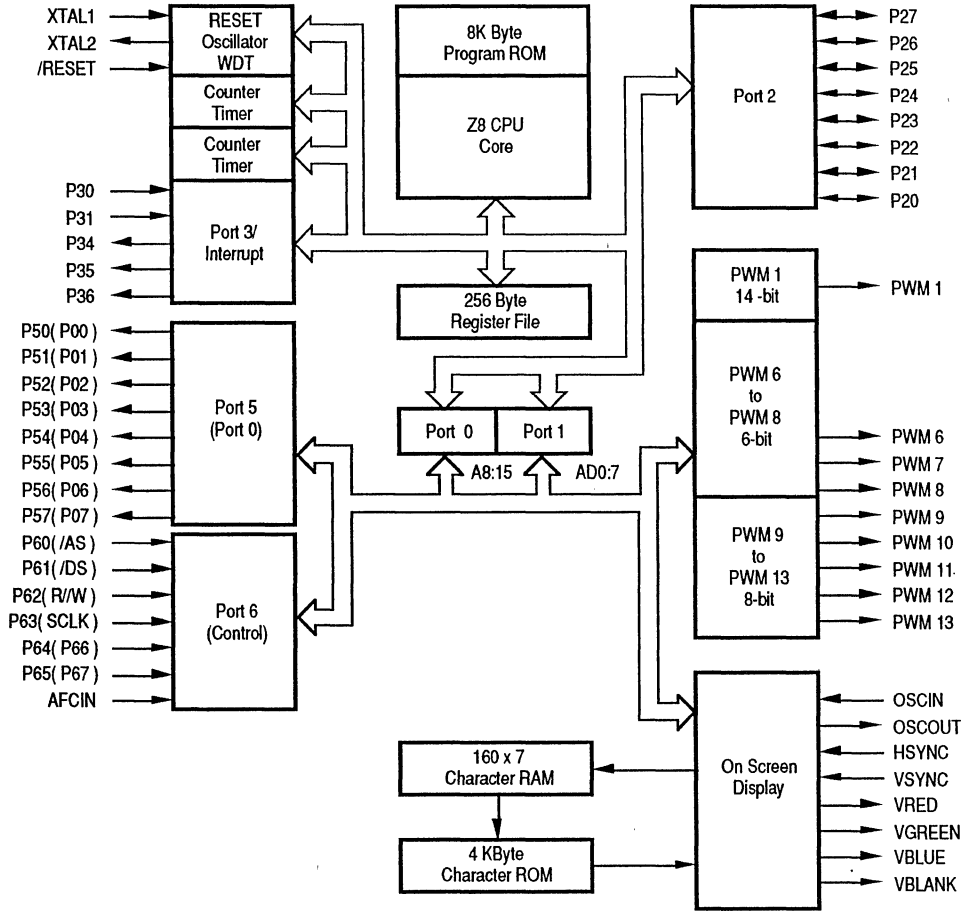


Figure 1. Functional Block Diagram

PIN CONFIGURATION

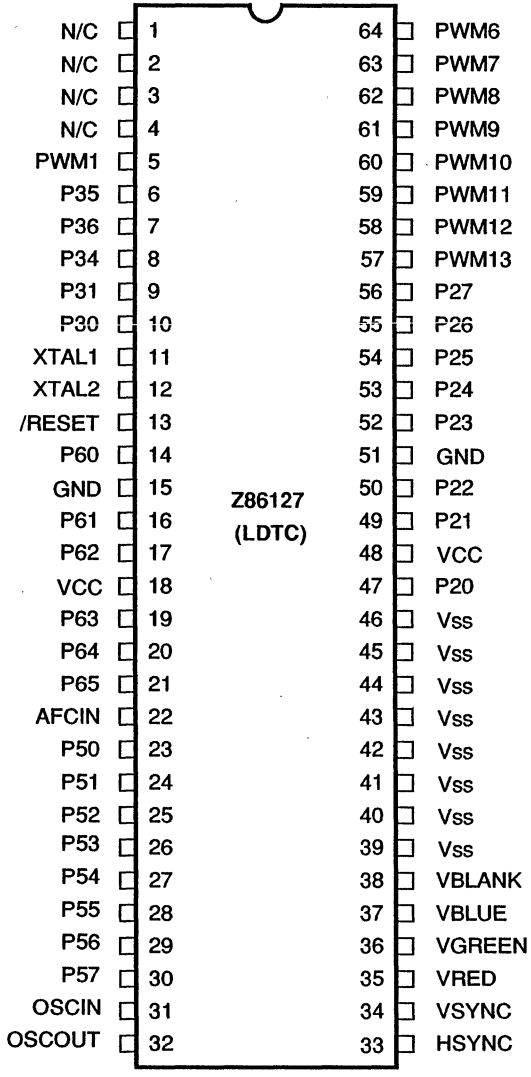


Figure 2. Z86127 Mask-ROM Plastic DIP

PIN IDENTIFICATION
64-pin DIP Z86127

Pin	Name	Function	Direction
1	N/C	No Connection	
2	N/C	No Connection	
3	N/C	No Connection	
4	N/C	No Connection	
5	PWM1	Pulse Width Modulator 1	Output
6, 7	P35-6	Port 3 pin 5, 6	Output
8	P34	Port 3 pin 4	Output
9	P31	Port 3 pin 1	Input
10	P30	Port 3 pin 0	Input
11	XTAL1	Crystal Oscillator	Input
12	XTAL2	Crystal Oscillator	Output
13	/RESET	System Reset	Input
14	P60	Port 6 pin 0	Input
15	GND	Ground	Input
16	P61	Port 6 pin 1	Input
17	P62	Port 6 pin 2	Input
18	V_{CC}	Power Supply	Input
19-21	P63-5	Port 6 pin 3, 4, 5	Input
22	AFC_{IN}	AFC Voltage Level	Input
23-30	P50-7	Port 5 pin 0, 1, 2, 3, 4, 5, 6, 7	Output
31	OSC_{IN}	Video Dot Clock Osc	Input
32	OSC_{OUT}	Video Dot Clock Osc	Output
33	HSYNC	Horizontal Sync	Input
34	VSYNC	Vertical Sync	Input
35	Vred	Video Red	Output
36	Vgreen	Video Green	Output
37	Vblue	Video Blue	Output
38	Vblank	Video Blank	Output
39-46	V_{SS}	Pull high to V_{CC}	In
47	P20	Port 2 pin 0	In/Output
48	V_{CC}	Power Supply	Input
49,50	P21-2	Port 2 pin 1, 2	In/Output
51	GND	Ground	Input
52-56	P23-7	Port 2 pin 3, 4, 5, 6, 7	In/Output
57	PWM13	Pulse Width Modulator 13	Output
58	PWM12	Pulse Width Modulator 12	Output
59	PWM11	Pulse Width Modulator 11	Output
60	PWM10	Pulse Width Modulator 10	Output
61	PWM9	Pulse Width Modulator 9	Output
62	PWM8	Pulse Width Modulator 8	Output
63	PWM7	Pulse Width Modulator 7	Output
64	PWM6	Pulse Width Modulator 6	Output

PIN DESCRIPTION

XTAL1, XTAL2. (Time-based input, output, respectively). These pins connect to the internal parallel-resonant clock crystal (4 MHz max) oscillator circuit with 2 capacitors to GND. XTAL1 is also used as an external clock input.

/AS. *Address Strobe* (output, active Low) is pulsed once at the beginning of each machine cycle. Address output is via Port 0 and Port 1 for all external programs. Memory address transfers are valid at the trailing edge of /AS. Under program control, /AS can be placed in the high impedance state along with Port 0 and Port 1, Data Strobe and Read/Write.

/DS. *Data Strobe* (output, active Low) is active once for each external memory transfer. For READ operations, data must be available prior to the trailing edge of /DS. For WRITE operations, the falling edge of /DS indicates the output data is valid.

R/W. *Read/Write* (output, Write active Low) signal is low when the DTC is writing to the external program or data memory.

SCLK. *System Clock.* SCLK is the internal system clock. It can be used to clock external glue logic.

HSYNC. (input Schmitt triggered, CMOS level). Horizontal Sync is an input pin that accepts an externally generated Horizontal Sync signal of either negative or positive polarity.

VSYNC. (input Schmitt triggered, CMOS level). Vertical Sync is an input pin that accepts an externally generated Vertical Sync signal of either negative or positive polarity.

OSC_{IN}, OSC_{OUT}. (Video Oscillator input, output, respectively). Oscillator input and output pins for on-screen display circuits. These pins connect to an inductor and two capacitors to generate the character dot clock (typically around 6MHz). The dot clock frequency determines the character pixel width and phase synchronized to HSYNC.

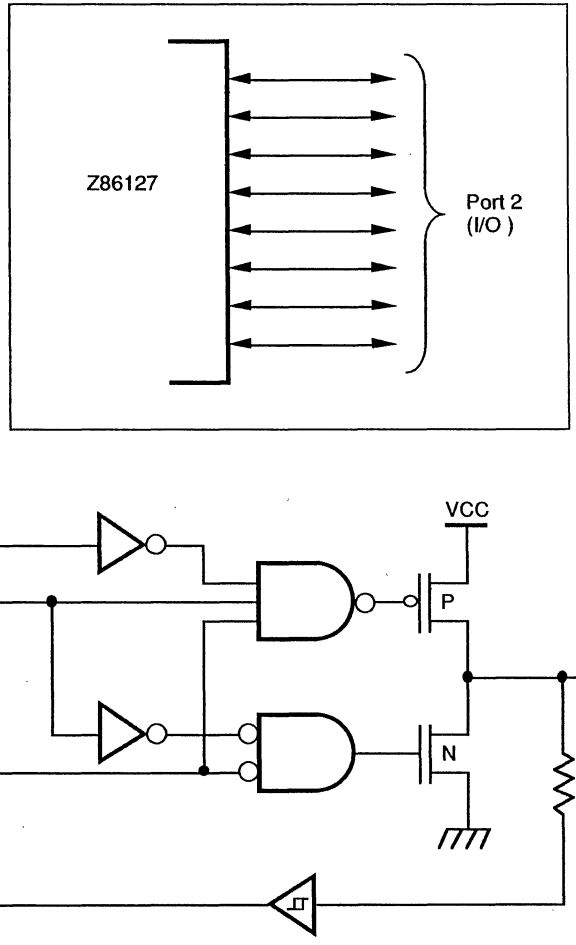
Vblank. *Video Blank* (output). CMOS output, programmable polarity. Used as a superimpose control port to display characters from video RAM. The signal controls Y signal output of the CRT and turns off the incoming video display while the characters in video RAM are superimposed on the screen. The red, green, and blue outputs drive the three electron guns on the CRT directly, while the blank output turns off the Y signal.

Vblue. *Video Blue* (output). CMOS Output of the Blue video signal (B-Y) and is programmable for either polarity.

Vgreen. *Video Green* (output). CMOS Output of the Green video signal (G-Y) and is programmable for either polarity.

Vred. *Video Red* (output). CMOS Output of the Red video signal (R-Y) and is programmable for either polarity.

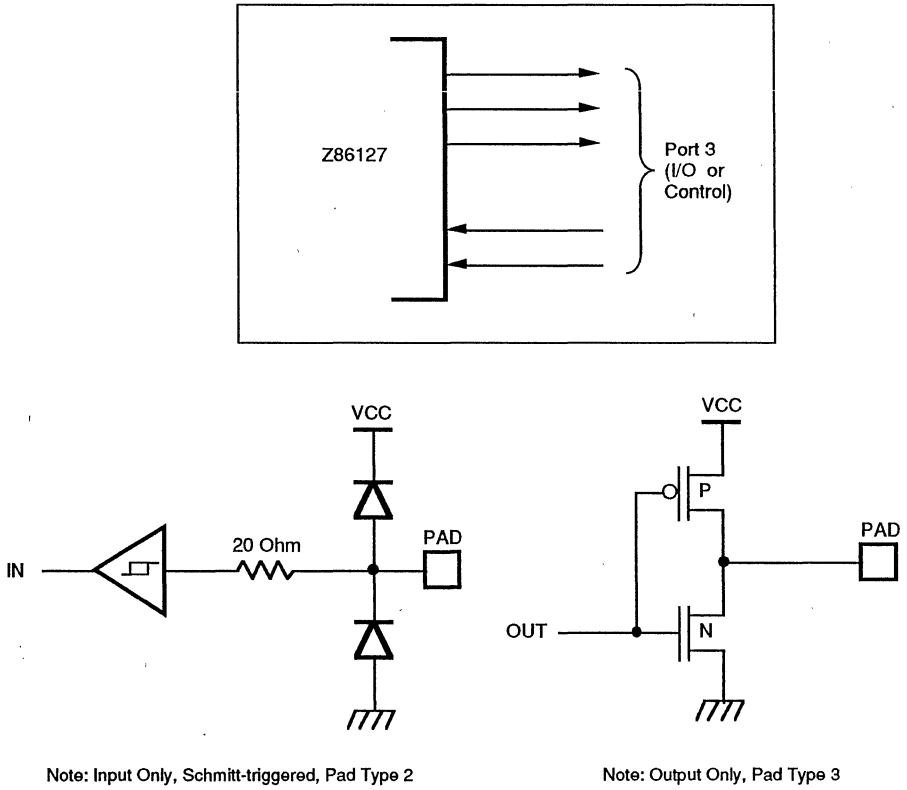
Port 2 (P20-P27). Port 2 is an 8-bit port, CMOS compatible, bit programmable for either input or output. Input buffers are Schmitt triggered. Bits programmed as outputs may be globally programmed as either push-pull or open-drain (Figure 3).



Note: Input/Output, 3-State, Open Drain, Pad Type 5

Figure 3. Port 2 Configuration

Port 3 (P30-1, P34-5 and P36). Port 3 Pin P30 input, is read directly. A negative edge event is latched in IRQ3 to initiate an IRQ3 vectored interrupt if appropriately enabled. An application could place the device in STOP mode when P30 goes low (in the IRQ3 interrupt routine). P30 initiates a STOP mode recovery when it subsequently goes high. Port 3, Pin P31 is read directly. A negative edge event is latched in IRQ2 to initiate an IRQ2 vectored interrupt if appropriately enabled. P31 high is signified as the T_{IN} signal to Timer1. Port 3, Pin P34 and Pin P35 are general purpose output lines. Port 3, Pin P36 can be used as a general purpose output or as an output for T_{OUT} (from Timer1 or Timer2) or SCLK (Figure 4).



Note: Input Only, Schmitt-triggered, Pad Type 2

Note: Output Only, Pad Type 3

Figure 4. Port 3 Configuration

Port 5 (P50-P57). Port 5 is an 8-bit, CMOS compatible, Output Port. The output ports can directly sink 10 mA at 1.5 Volt V_{OL}. They are typically used to drive multiplexed LED displays (Figure 5).

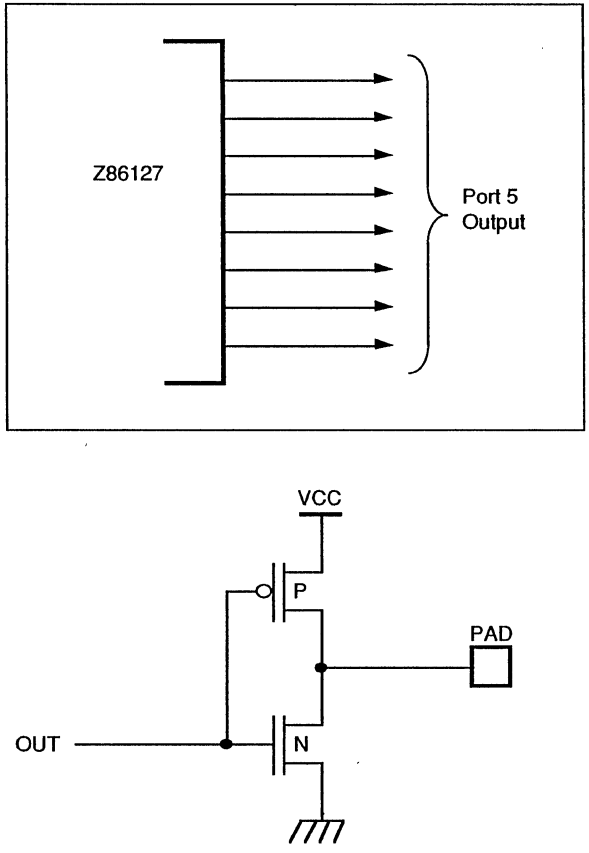


Figure 5. Port 5 Configuration

PIN DESCRIPTION (Continued)

Port 6 (P60-P65). Port 6 is a 6-bit, Schmitt triggered CMOS compatible, input port. The outputs of the AFC comparators internally feed into the Port 6, bit-6 and bit-7 inputs (Figure 6).

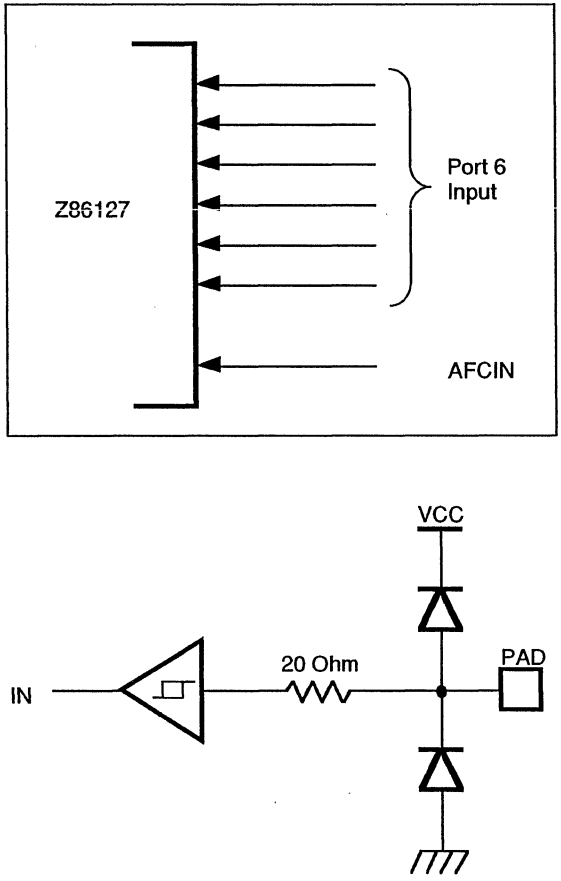


Figure 6. Port 6 Configuration

AFCIN. (Comparator input port, memory mapped). The input signal is supplied to two comparators with VTH1= 2/5 V_{cc} and VTH2=3/5 V_{cc} typical threshold voltage. The comparator outputs are internally connected to Port 6, bit-6 and bit-7. AFC_{IN} is typically used to detect AFC voltage level to accommodate digital automatic fine tuning functions (Figure 7).

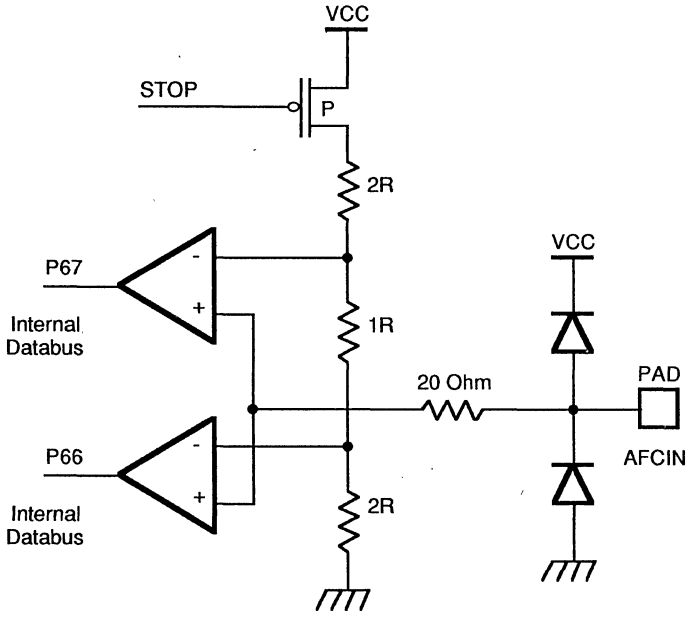


Figure 7. AFC_{IN} Comparator Circuits

PIN DESCRIPTION (Continued)

Pulse Width Modulator 1 (PWM). PWM1 is typically used as the D/A converter for Voltage Synthesis Tuning systems.

Pulse Width Modulator 6-8 (PWM). PWM6-PWM8 are Pulse Width Modulators with 6-bit resolution.

Pulse Width Modulator 9-13 (PWM). PWM9-PWM13 are Pulse Width Modulator circuits with 8-bit resolution or individually programmed as general purpose outputs. In either case, the output drivers are 12-volt open-drain circuits.

/RESET. System Reset. Code is executed from memory address 000C (HEX) after the /RESET pin is set to a high level. The reset function is also carried out by detecting a V_{CC} transition state (automatic power on reset) so that the external reset pin can be permanently tied to V_{CC}. A low level on /RESET forces a restart of the device.

SPECIAL FUNCTIONS

The Z8 LDTC incorporates special functions to enhance the Z8's application in consumer, industrial and television control applications.

Pulse Width Modulator (PWM). The LDTC has nine PWM channels (Figure 12). There are three types of PWM circuits: PWM1 (1 channel of 14-bit resolution) typically used for Voltage Synthesis Tuning, PWM6-PWM8 (3 channels of 6-bit resolution) typically used for audio level control, and PWM9-PWM13 (5 channels of 8-bit resolution) typically used for picture level control. The PWM control registers are mapped into external memory and are accessed via LDE and LDEI instructions.

On Screen Display (OSD). The OSD has a capability of displaying 8 rows by 20 columns of 128 kinds of characters for either high resolution (11x15 dots) or low resolution (5x7 dots) pattern (Figures 8 and 9).

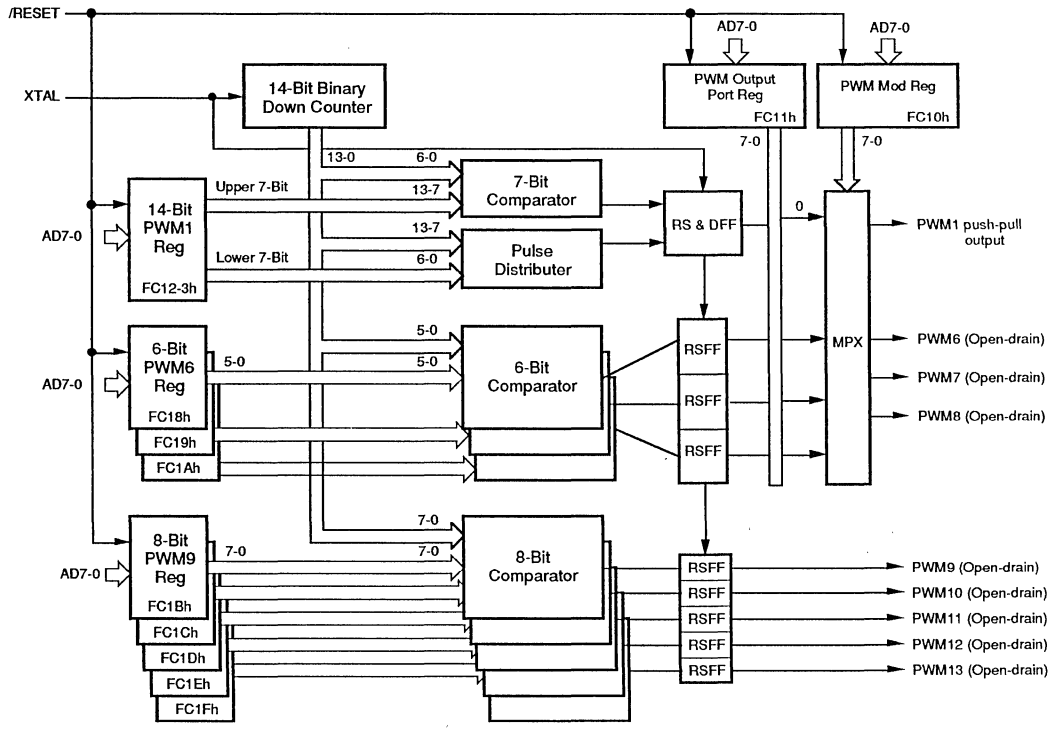


Figure 8. Pulse Width Modulator Block Diagram

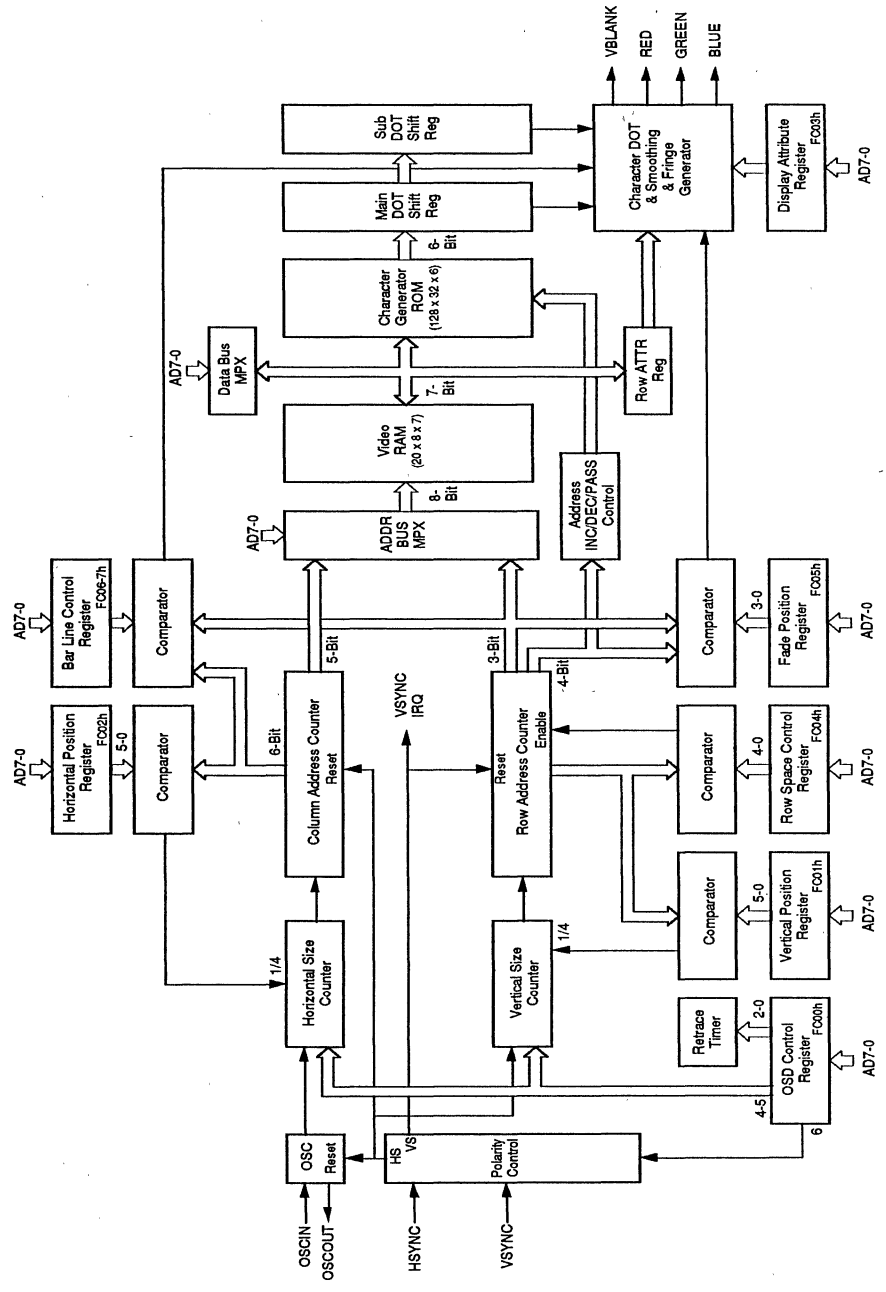


Figure 9. On-Screen-Display Block Diagram

The OSD features are as follows:

- Character Color: Seven kinds of color are specified on a row basis.
- Character Pixel Size: Four character pixel sizes are selected for a low resolution (2HL, 4HL, 6HL and 8HL) and high resolution (1HL, 2HL, 3HL and 4HL) Horizontal Line (HL).
- Polarity Selections: Can select active low or high for horizontal/vertical sync input and RGB outputs.
- Display Position: Can display 64 vertical positions by 4HL units and 64 horizontal positions by a 4 dot clock.
- Inter Row Spacing: Inter row vertical line spacing is set from 2HL to 25HL (17HL for high resolution).
- Fade In/Out Control: Fade position can be determined in vertical direction.
- Bar Line Type Display: One of the rows is selected to display an analog bar line every half column by setting second color with proper character set.
- Fringe Function: Fringe off/on and the color selected.
- Background Color: Eight kinds of color including black background color.
- ON/OFF Control: Character display, backgrounds are turned on and off.
- Number of Display Characters: 8 rows x 20 columns.
- Character Set: 128 (5x7 dots or 11x15 dots).

Character Generator ROM. The character generator ROM is organized as 4 Kbytes of 6 bits. The ROM defines either 11x15 dot (high resolution) or 5x7 (low resolution) characters (Figure 10).

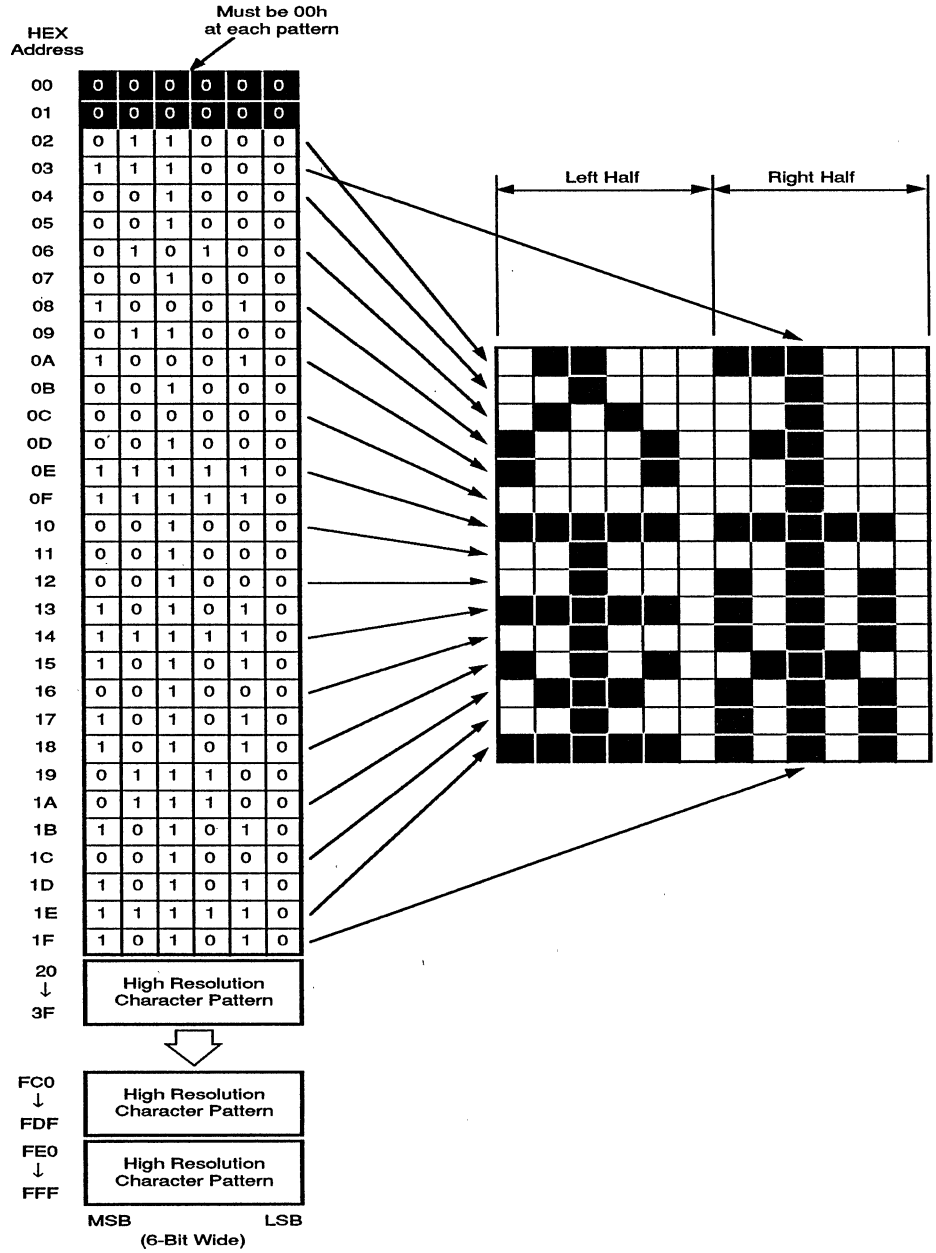


Figure 10a. High and Low Resolution Character ROM Configuration

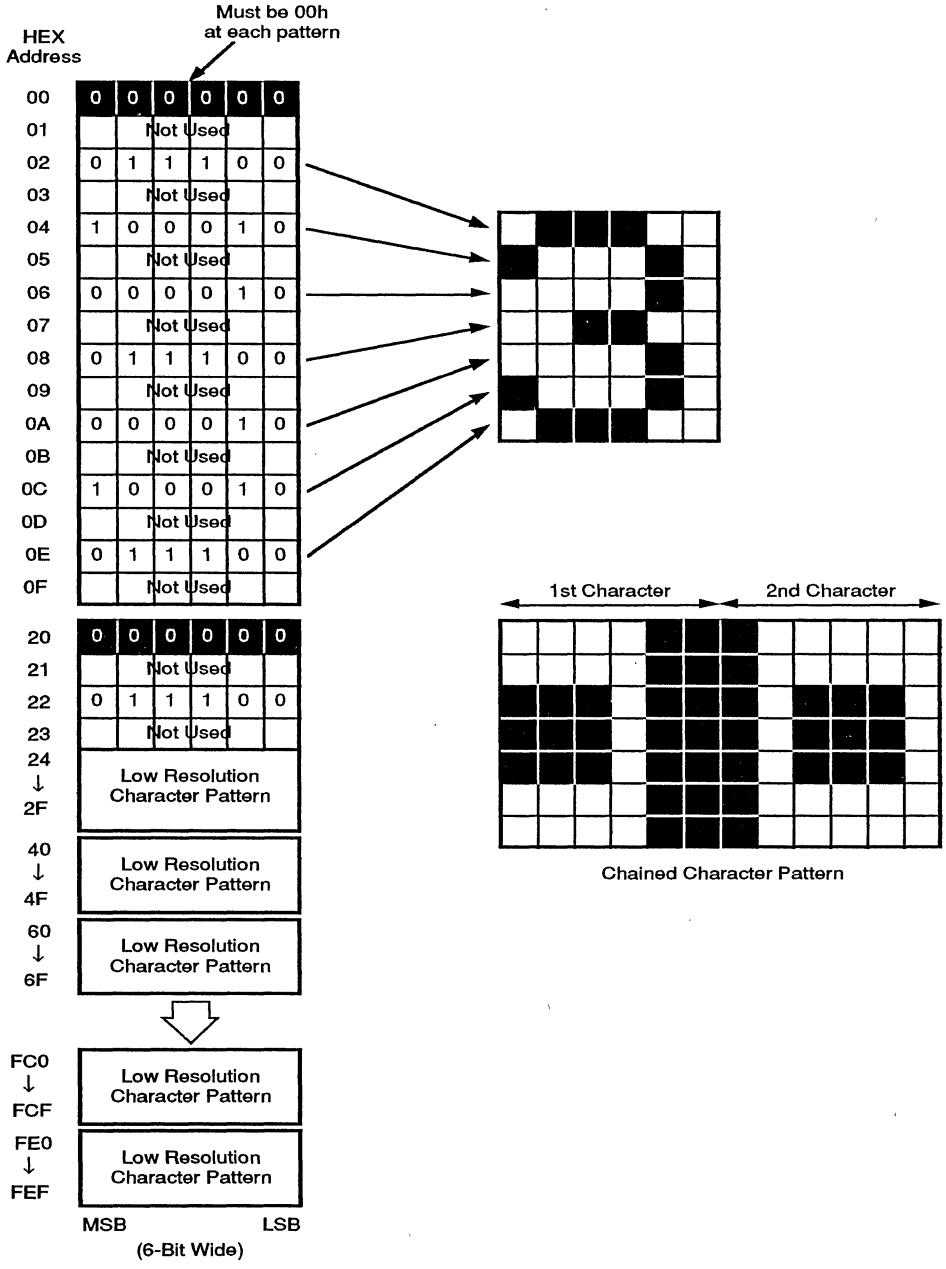


Figure 10b. High and Low Resolution Character ROM Configuration

SPECIAL FUNCTIONS (Continued)

Program Memory. The program ROM size is 8-Kbytes (Figure 11). The IRQ vector table is located in the lower address space. The vector address is fetched after the corresponding interrupt and program control is passed to the specified vector address. IRQ1 vector is fixed to VSYNC interrupt request and occurs at the leading edge of the filtered VSYNC input. Program memory start at address 000C (HEX) after reset.

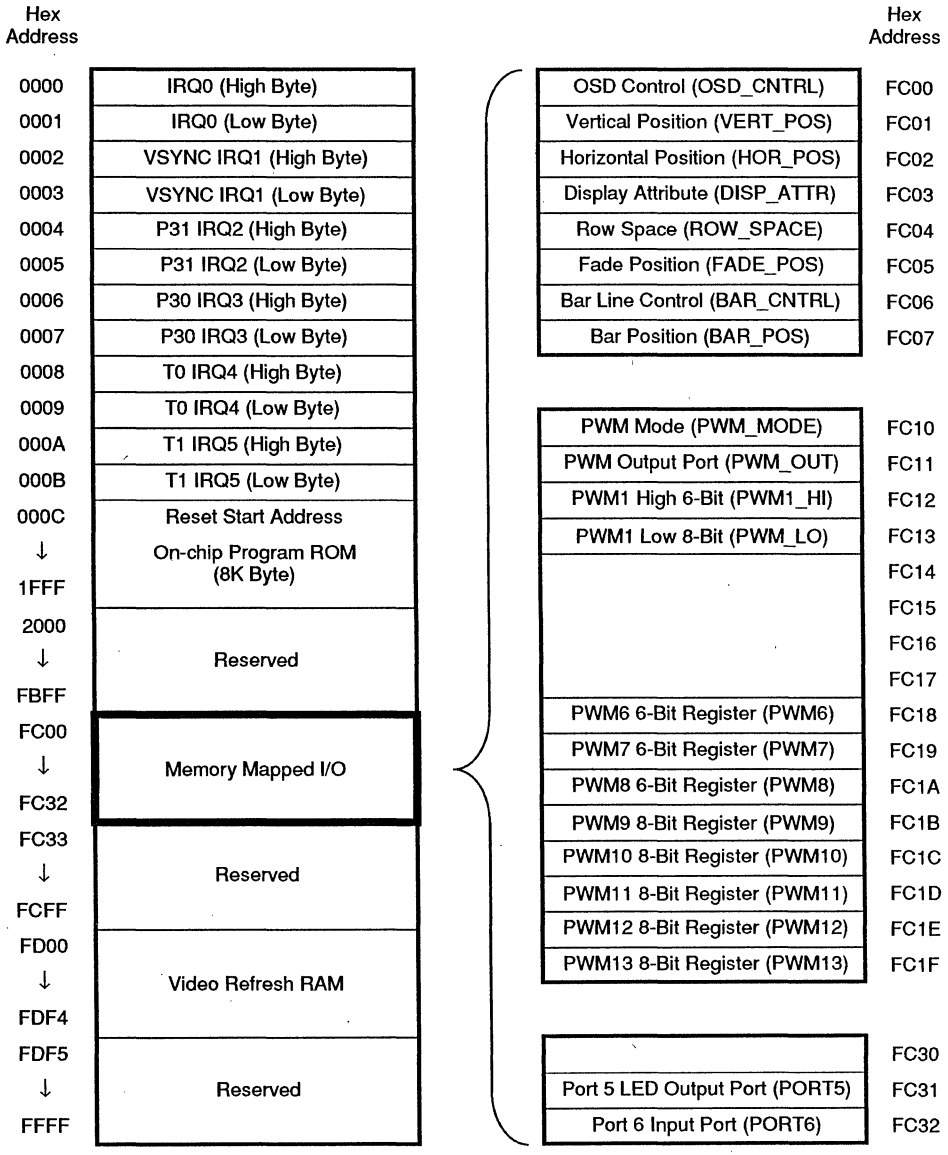


Figure 11. Program Memory

Memory Mapped Register. All control registers and I/O ports (except Port 2 and Port 3) are assigned to program memory space. Address space FC00 (HEX) contains OSD control registers, PWM output registers and Ports 5 and 6 I/O registers. Two bits of the decoded AFCIN port are assigned to Port 6 input port. LDE and LDEI instructions are required to transfer data between the Register File and the Memory Mapped Registers.

Register File. A total of 253 byte registers are implemented in the Z8 core. Address 00 (HEX), 01 (HEX) and FO (HEX) are reserved. The register file consists of 2 I/O Port registers, 236 general-purpose registers and 15 control and status registers (Figure 12). The instructions can access registers directly or indirectly with an 8-bit address field. This also allows short 4-bit register addressing using the Register Pointer. In the 4-bit mode, the register file is divided into sixteen working-register groups, each occupying 16 continuous locations. The Register Pointer addresses the starting location of the active working-register group (Figure 13).

Note: Register Bank E0-EF is only accessed through a working register and indirect addressing modes.

Hex Address	
02	Port 2 (P2)
03	Port 3 (P3)
04	General - Purpose Registers
EF	
F0	Reserved
F1	Timer Mode (TMR)
F2	Timer/Counter1 (T1)
F3	T1 Prescaler (PRE1)
F4	Timer/Counter0 (T0)
F5	T0 Prescaler (PRE0)
F6	Port 2 Mode (P2M)
F7	Port 3 Mode (P3M)
F8	Port 0-1 Mode (P01M)
F9	Interrupt Priority Reg (IPR)
FA	Interrupt Request Reg (IRQ)
FB	Interrupt Mask Reg (IMR)
FC	Condition Flag (FLAGS)
FD	Register Pointer (RP)
FE	Stack Pointer High (SPH)
FF	Stack Pointer Low (SPL)

Figure 12. Register File Configuration

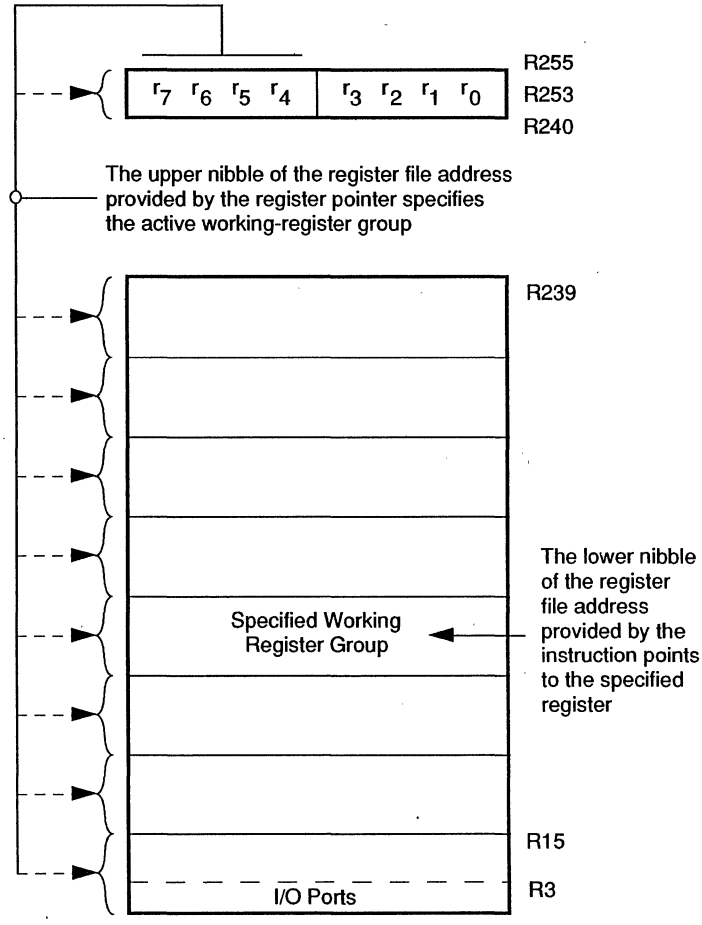


Figure 13. Register Pointer

Stack. Either the internal register file or the external data memory is used for the stack. A 16-bit Stack Pointer is used for the external stack, which can reside anywhere in data memory. An 8-bit Stack Pointer is used for the internal stack that resides within the 236 general-purpose registers.

Counter/Timers. There are two 8-bit programmable counter/timers (T0-T1), each driven by its own 6-bit programmable prescaler (PRE0 and PRE1). The T1 prescaler can be driven by internal or external clock sources; however, the T0 prescaler is driven by the internal clock only (Figure 14).

The counter, but not the prescalers, are read at any time without disturbing their value or count mode. The clock source for T1 is user-definable and is the internal microprocessor clock (XTAL clock/4), or an external signal input via Port 3, P31. The counter/timers are programmably cascaded by connecting the T0 output to the input of T1.

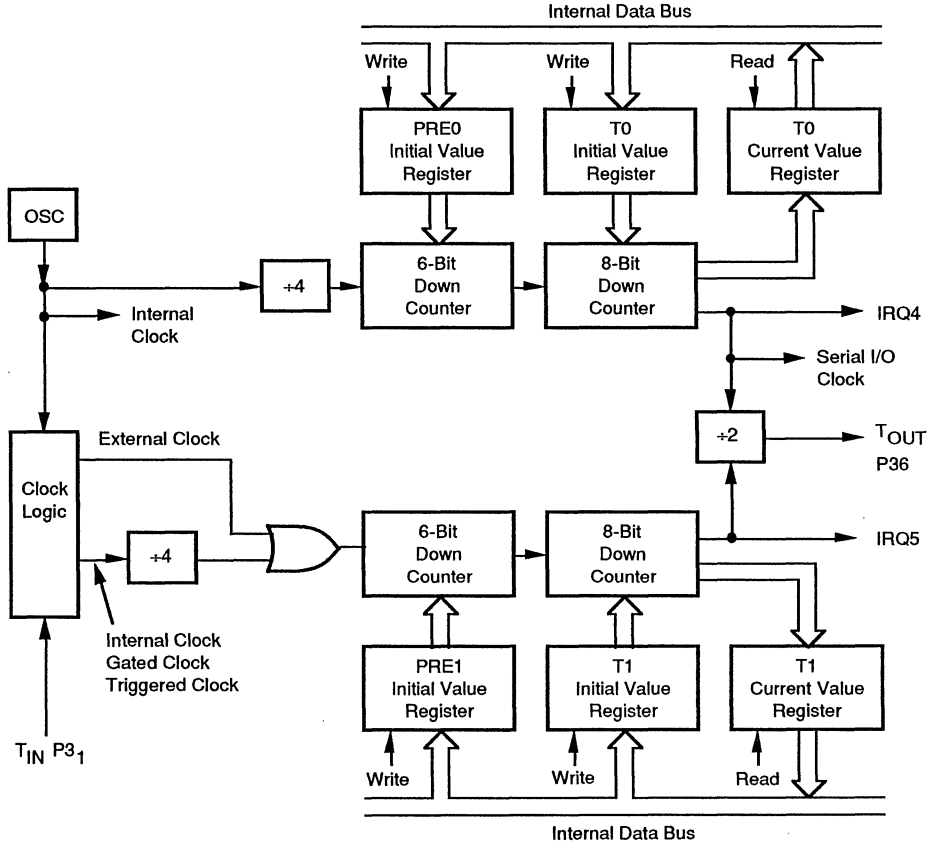


Figure 14. Counter/Timer Block Diagram

SPECIAL FUNCTIONS (Continued)

Interrupts. The LDTC has six different interrupts from six different sources. These interrupts are maskable and prioritized (Figure 15). The six sources are divided as follows: two sources are claimed by Port 3 (P30, P31), one by VSYNC, two by the counter/timers, and one is software triggered only.

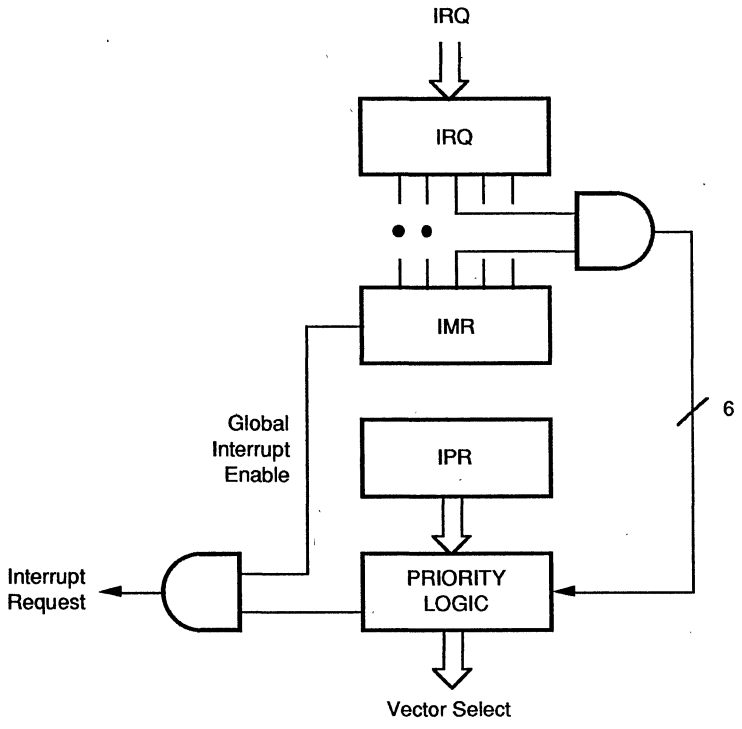


Figure 15. Interrupt Block Diagram

HALT Mode. The Z86127 is driven by two internal clocks, TCLK and SCLK, They both oscillate at the crystal frequency. TCLK provides the clock signal for the counter-timers and the interrupt block. SCLK provides the clock signal for all other CPU blocks. Halt mode turns off the internal CPU clock (SCLK), but not the XTAL oscillation. The counter/timers and external interrupts remain active. The device may be recovered by interrupts, either external or internally generated.

STOP Mode. The STOP instruction stops crystal oscillation, thereby stopping both SCLK and TCLK. The device ceases to operate. The STOP mode can be released by two methods. The first method is to reset the device. A high input condition on Port 3 Pin P30 is the second method. After releasing the STOP mode by using either one of the two methods, program execution begins at location 000C (HEX). To complete an instruction prior to entering the standby modes, a NOP instruction has to be placed before the HALT or STOP instructions. This is required because of instruction pipelining. i.e.:

```
FF NOP    ; clear the pipeline
6F STOP   ; enter STOP mode
          or
FF NOP    ; clear the pipeline
7F HALT   ; enter HALT mode
```

Note:
In STOP mode, XTAL2 pin has an internal pull-up on it and OSCOUT has an internal pull-down.

Clock. The Z86127 on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal, ceramic resonator, or any suitable external clock source (XTAL1 = Input, XTAL2 = Output). The crystal is an AT cut, parallel resonant, 4 MHz max with a series resistance (RS) less than or equal to 100 Ohms.

The crystal source is connected across XTAL1 AND XTAL2 using the recommended capacitors (10 pF < CL < 300 pF, where C1=C2=CL) from each pin to ground (Figure 16).

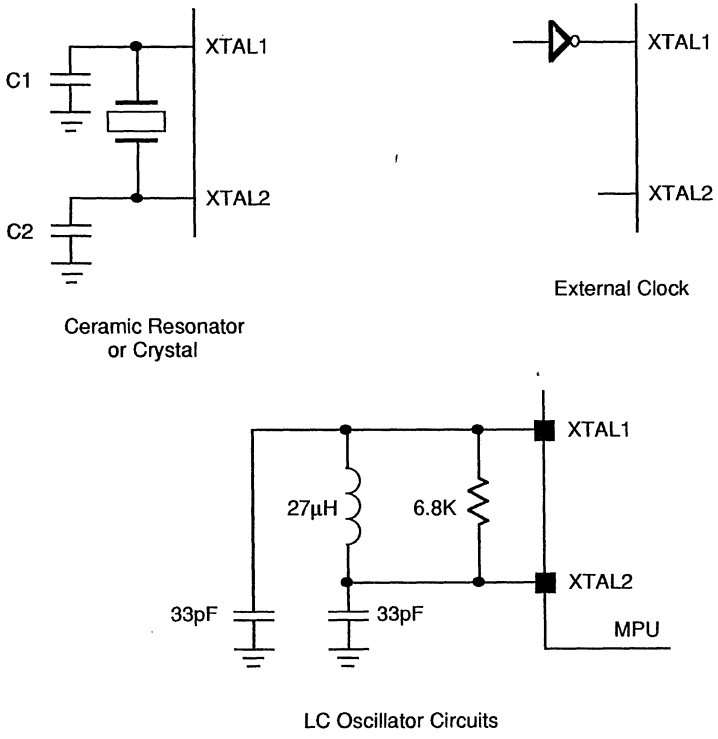


Figure 16. Oscillator Configuration

SPECIAL FUNCTIONS (Continued)

Watch Dog Timer (WDT). The Z86127 is equipped with a watch dog timer which should be refreshed within 12 ms. Failure to refresh the timer results in a reset of the device. The WDT is permanently enabled.

V_{cc} Voltage Sensitive Reset (VSR). Reset is globally driven if V_{cc} is below the specified voltage (Figure 17).

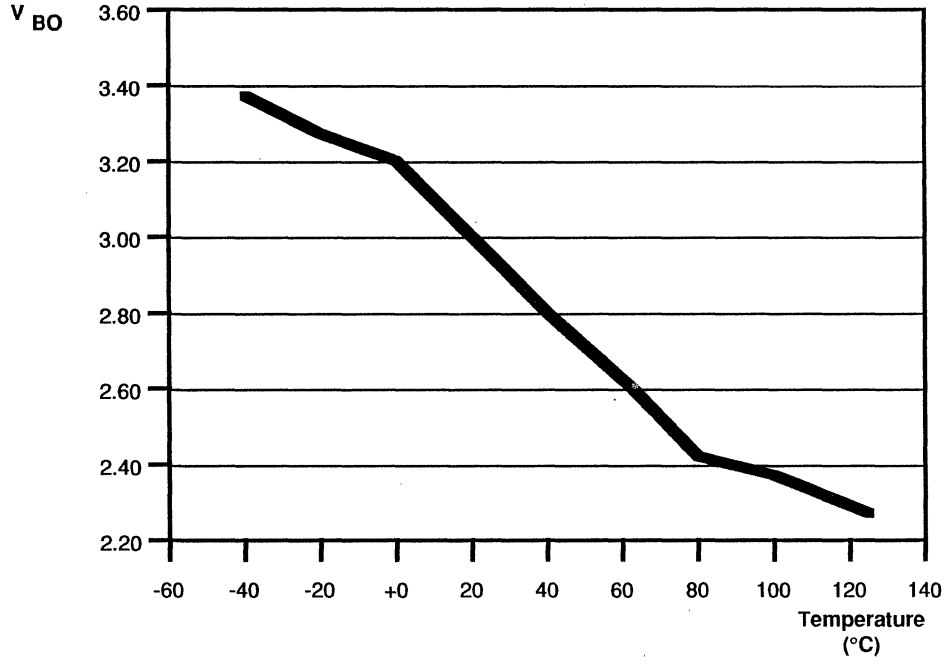


Figure 17. Voltage Sensitive Reset vs Temperature

ABSOLUTE MAXIMUM RATINGS

Stress greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Symbol	Parameters	Min	Max	Units	Notes
V_{CC}	Power Supply Voltage †	-0.3	+7	V	
V_I	Input Voltage	-0.3	V_{CC}+0.3	V	
V_I	Input Voltage	-0.3	V_{CC}+0.3	V	[1]
V_O	Output Voltage	-0.3	V_{CC}+8.0	V	[2]
I_{OH}	Output Current High		-10	mA	1 pin
I_{OH}	Output Current High		-100	mA	all total
I_{OL}	Output Current Low		20	mA	1 pin
I_{OL}	Output Current Low		40	mA	[3] (1 pin)
I_{OL}	Output Current Low, all total		200	mA	
T_A	Operating Temperature	††			
T_{STG}	Storage Temperature	-65	+150	C	

Notes:
[1] Port 2 open-drain
[2] PWM open drain outputs
[3] Port 5
† Voltage on all pins with respect to GND.
†† See Ordering Information

STANDARD TEST CONDITIONS

The characteristics listed below apply for standard test conditions as noted. All voltages are referenced to GND. Positive current flows into the referenced pin (Figure 18).

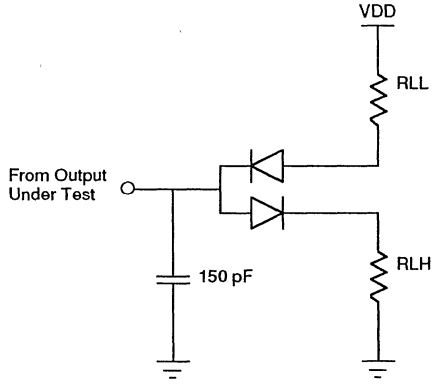


Figure 18. Test Load Diagram

CAPACITANCE

T_A=25°C; V_{CC}=GND=0V; Freq=1.0 MHz; unmeasured pins to GND.

Parameter	Max	Units
Input capacitance	10	pF
Output capacitance	20	pF
I/O capacitance	25	pF
AFCin input capacitance	10	pF

DC CHARACTERISTICS

T_A=0°C to +70°C; V_{CC}=+4.5V to +5.5V; F_{OSC}=4 MHz

Sym	Parameter	T_A=0°C to 70°C Min	Max	Typical @ 25°C	Units	Conditions
V_{IL}	Input Voltage Low	0	0.2 V_{CC}	1.48	V	
V_{ILC}	Input XTAL/Osc In Low		0.07 V_{CC}	0.98	V	External Clock Generator Driven
V_{IH}	Input Voltage XTAL/Osc In High	0.7 V_{CC}	V_{CC}	3.2	V	External Clock Generator Driven
V_{IHC}	Input XTAL/Osc in High	0.8 V_{CC}	V_{CC}	3.0	V	External Clock Generator Driven
V_{HY}	Schmitt Hysteresis	0.1 V_{CC}		0.8	V	
V_{PU}	Maximum Pull-up Voltage		12		V	[2]
V_{OL}	Output Voltage Low		0.4	0.16	V	I_{OL}=1.00mA
			0.4	0.19	V	I_{OL}=3.2mA, [1]
			0.4	0.19	V	I_{OL}=0.75mA [2]
			1.5	1.00	V	I_{OL}=10mA [1]
V_{00-01}	AFC Level 01 In		0.45 V_{CC}	1.9	V	
V_{01-11}	AFC Level 11 In	0.5 V_{CC}	0.75 V_{CC}	3.12	V	
V_{OH}	Output Voltage High	V_{CC}-0.4		4.75	V	I_{OH}=-0.75mA
I_{IR}	Reset Input Current		-80	-46	µA	V_{RL}=0V
I_{IL}	Input Leakage	-3.0	3.0	0.01	µA	0V,V_{CC}
I_{OL}	Tri-State Leakage	-3.0	3.0	0.02	µA	0V,V_{CC}
I_{CC}	Supply Current		20	13.2	mA	All inputs at rail
I_{CC1}			6	3.2	mA	All inputs at rail
I_{CC2}			10	0	µA	All inputs at rail

Notes:
[1] Port 5
[2] PWM Open Drain

AC CHARACTERISTICS

Timing Diagrams (Figures 19-23)

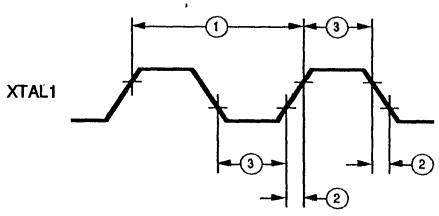


Figure 19. External Clock

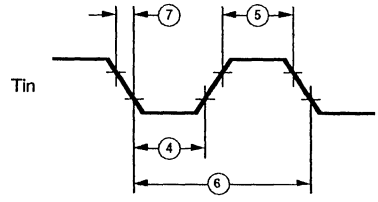


Figure 20. Counter Timer

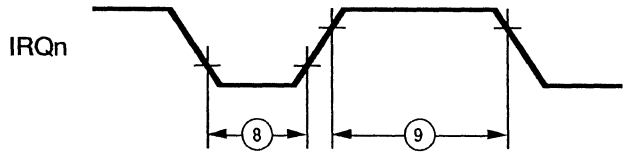


Figure 21. Interrupt Request

AC CHARACTERISTICS
Timing Diagrams (Continued)

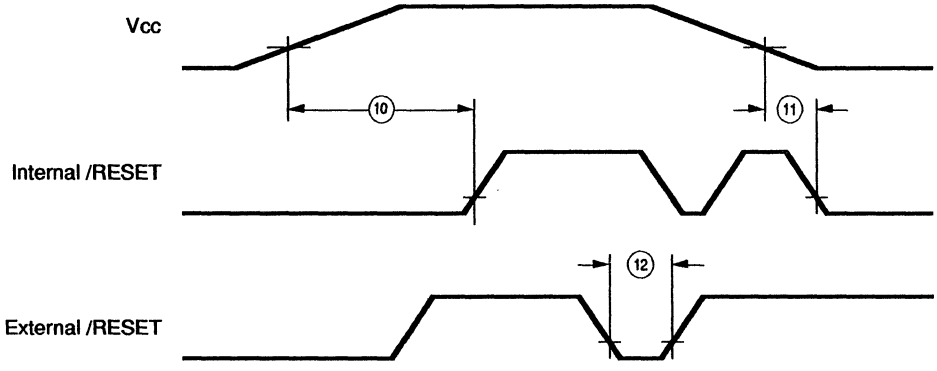


Figure 22. Power On Reset

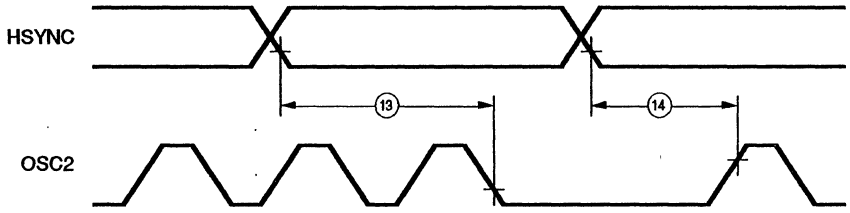


Figure 23. On Screen Display

AC CHARACTERISTICS

T_A=0° C to 70° C; V_{CC}=+4.5V to +5.5V; F_{osc}=4 MHz,

No	Symbol	Parameter	Min	Max	Unit
1	TpC	Input clock period	250	1000	ns
2	TrC,TfC	Clock input raise and fall		15	ns
3	TwC	Input clock width	70		ns
4	TwTinL	Timer input low width	70		ns
5	TwTinH	Timer input high width	3TpC		
6	TpTin	Timer input period	8TpC		
7	TrTin,TfTin	Timer input raise and fall		100	ns
8A	TwIL	Int req input low	70		ns
8B	TwIL		3TpC		
9	TwIH	Int request input high	3TpC		
10	TdPOR	Power On Reset delay	25	100	ms
11	TdLVIRES	Low voltage detect to In-Internal RESET condition	200		ns
12	TwRES	Reset minimum width	5TpC		
13	TdHsOl	Hsync start to Vosc stop	2TpV	3TpV	
14	TdHsOh	Hsync end to Vosc start		1TpV	
15	TdWDT	WDT Refresh Time		12	ms

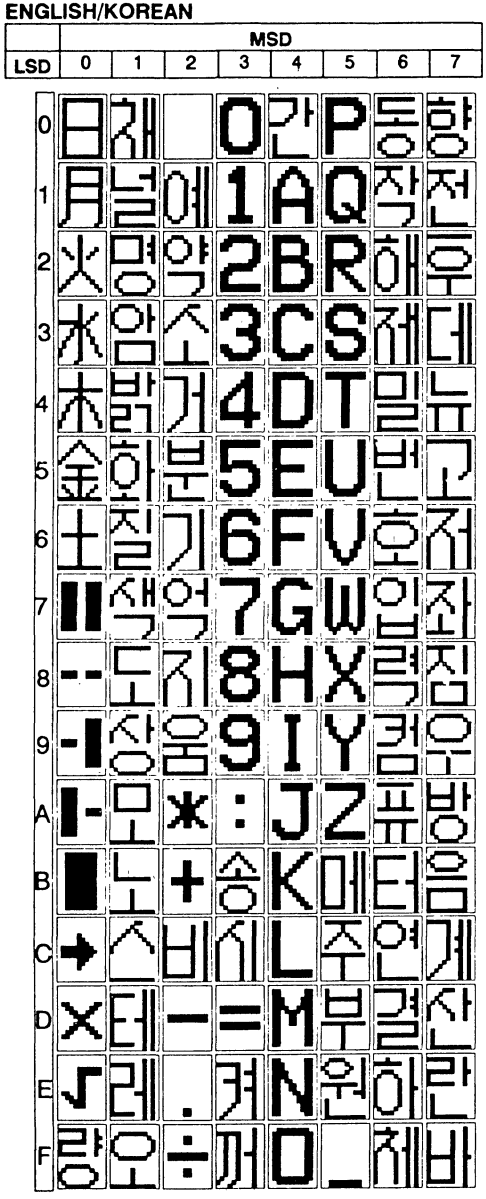
Notes:

[1] Refer to DC Characteristics for details on switching levels.

* Units in nanoseconds

STANDARD CHARACTER SETS

ENGLISH/KOREAN



SUMMARY

Input/Output Circuits (Figures 24-32)

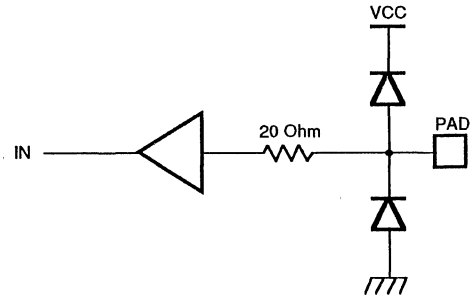


Figure 24. Input Only (Pad Type 1)

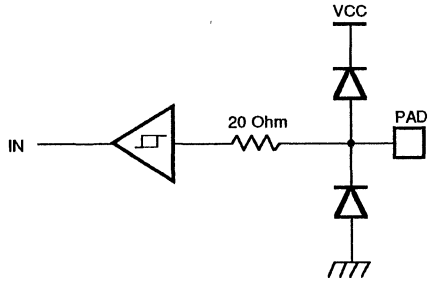


Figure 25. Input Only, Schmitt Triggered (Pad Type 2)

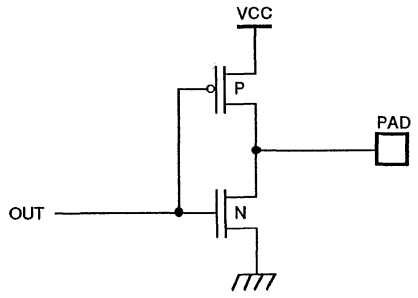


Figure 26. Output Only (Pad Type 3)

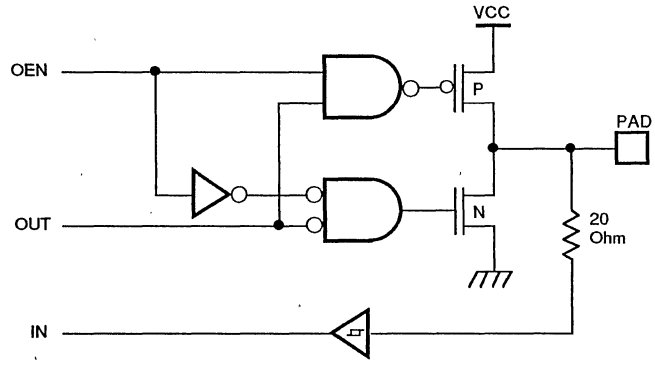


Figure 27. Input/Output 3-State
(Pad Type 4)

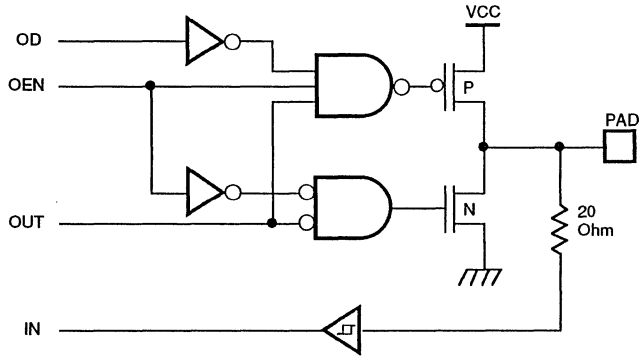


Figure 28. Input/Output, 3-state, Open Drain
(Pad Type 5)

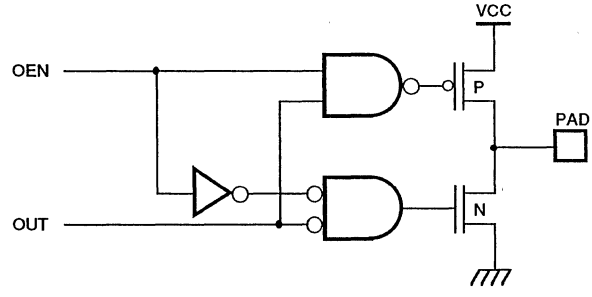


Figure 29. Output Only, 3-State
(Pad Type 6)

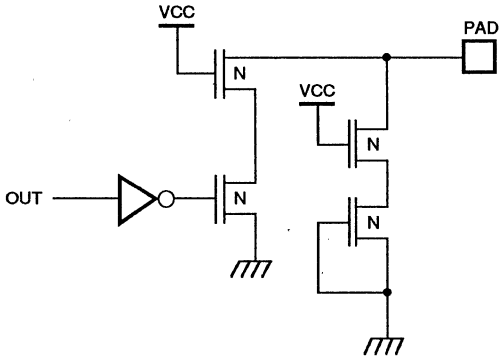


Figure 30. Output Only, 12-Volt Open Drain (Pad Type 7)

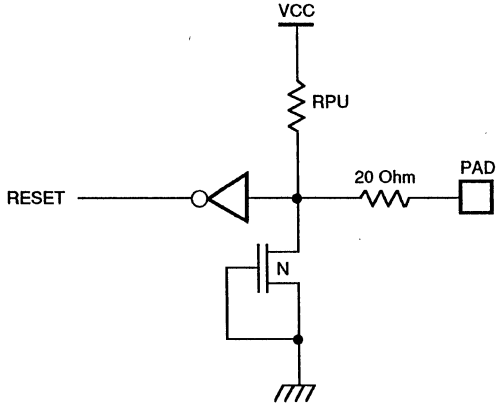


Figure 31. Reset Input Circuit (Pad Type 8)

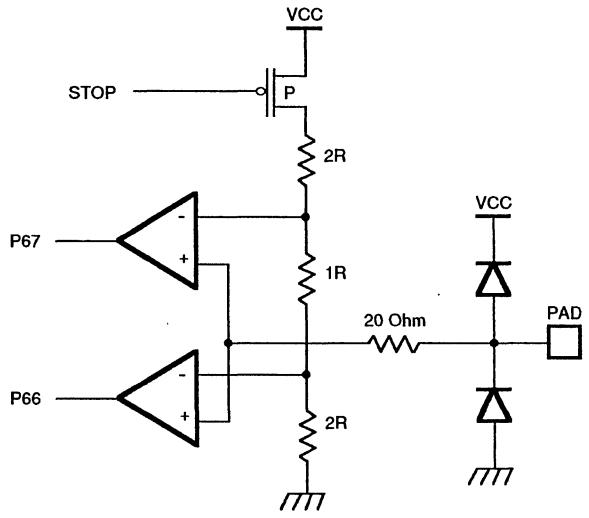


Figure 32. AFC Input Circuit (Pad Type 9)

Mapping of Symbolic Pad Types to Pin Functions

Pin Name	Pad Type	Notes
XTAL1, OSC_{IN} XTAL2, OSC_{OUT}	1	High gain start, low gain run amplifier circuit
/RESET	8	
P20-P27	5	
P30-P31	2	
P34-P36	3	
P50-P57	3	
P60-P65	2	
AFCIN	9	
HSYNC, VSYNC	2	
VRED, VBLUE, VGREEN,	3	
VBLANK	3	
PWM1	3	
PWM [2, 6 -13]	7	

DTC CONTROL REGISTER DIAGRAMS
Port Registers (Figures 33-49)

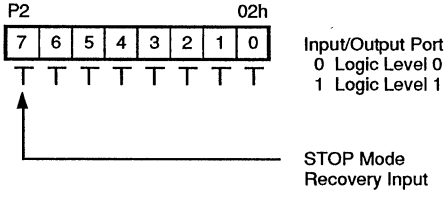


Figure 33. Port 2 Register

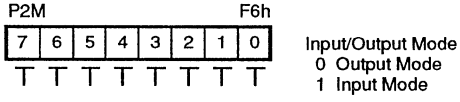


Figure 34. Port 2 Mode Register

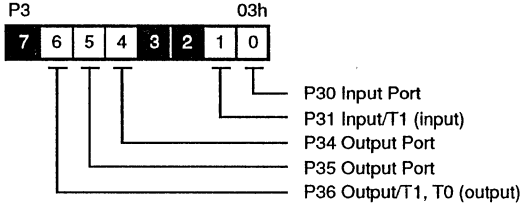


Figure 35. Port 3 Register

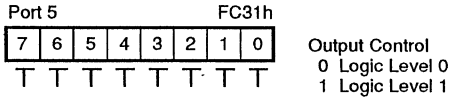


Figure 36. Port 5 Register

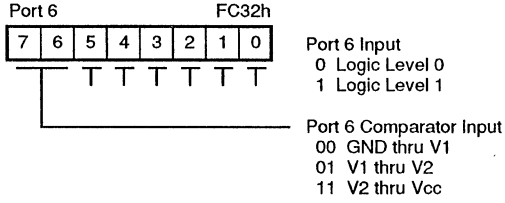


Figure 37. Port 6 Register

DTC CONTROL REGISTER DIAGRAMS
PWM Registers

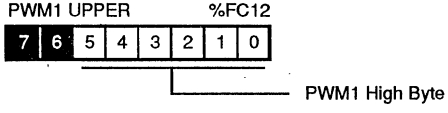


Figure 38. PWM 1 High Value

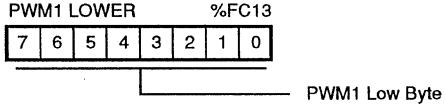


Figure 39. PWM 1 Low Value

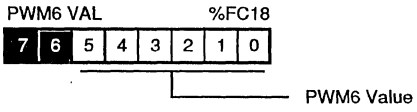


Figure 40. PWM 6 Value

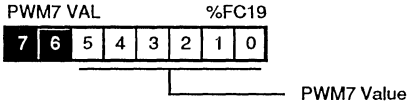


Figure 41. PWM 7 Value

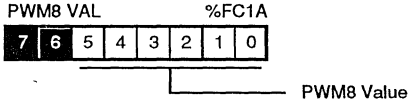


Figure 42. PWM 8 Value

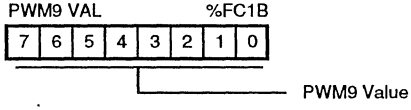


Figure 43. PWM 9 Value

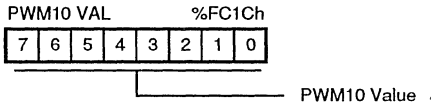


Figure 44. PWM 10 Value

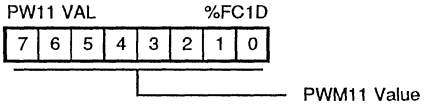


Figure 45. PWM 11 Value

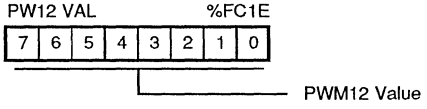


Figure 46. PWM 12 Value

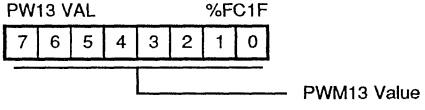


Figure 47. PWM 13 Value Register

DTC CONTROL REGISTER DIAGRAMS
PWM Registers (Continued)

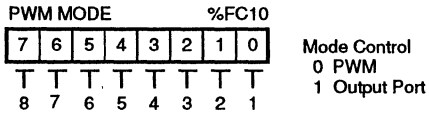


Figure 48. PWM Mode Register

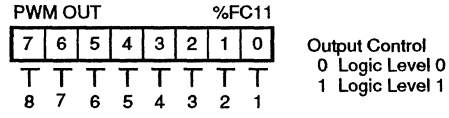


Figure 49. PWM Port Output Register

DTC CONTROL REGISTER DIAGRAMS
OSD Registers (Figures 50-57)

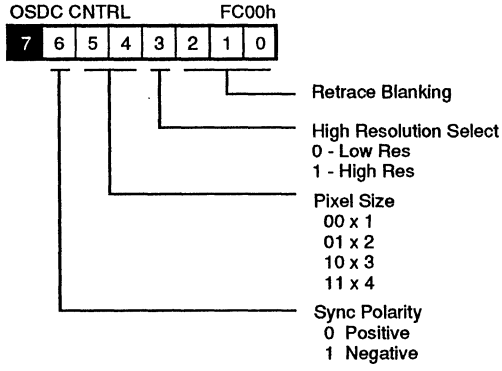


Figure 50. OSD Control Register

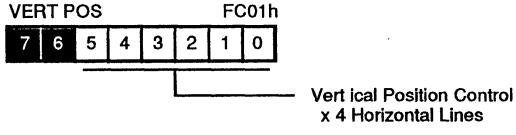


Figure 51. OSD Vertical Position Register

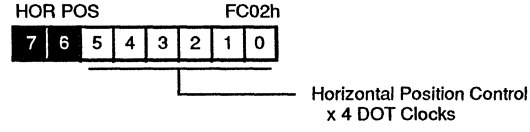


Figure 52. OSD Horizontal Position Register

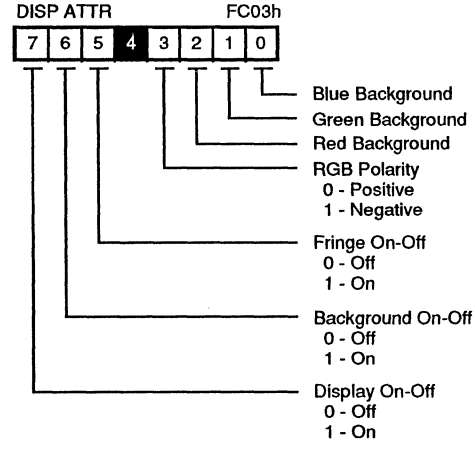


Figure 53. OSD Display Attribute Register

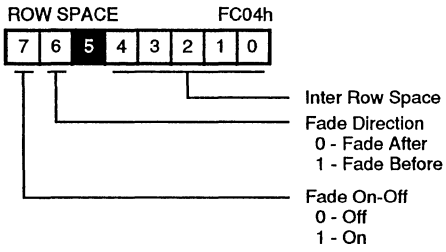


Figure 54. OSD Row Space Register

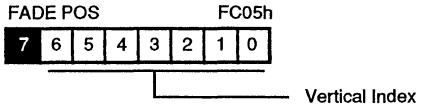


Figure 55. OSD Fade Position Register

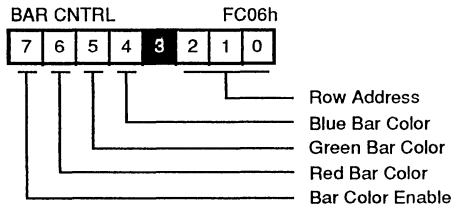


Figure 56. OSD Bar Control Register

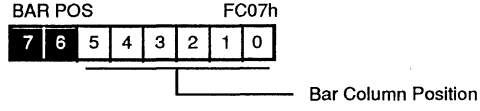


Figure 57. OSD Bar Position Register

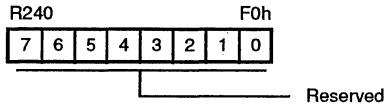


Figure 58. Reserved (F0H)

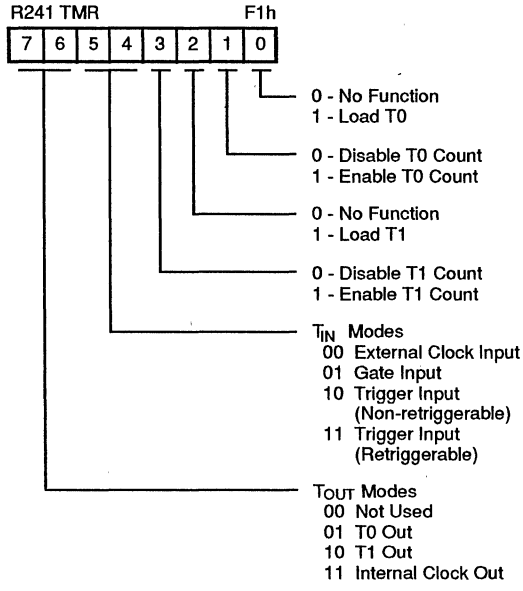


Figure 59. Timer Mode Register (F1H; Read/Write)

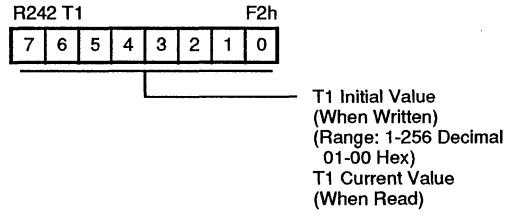


Figure 60. Counter Timer 1 Register (F1H; Read/Write)

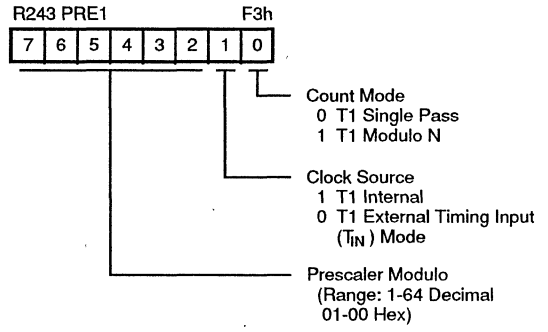


Figure 61. Prescaler 1 Register (F3H; Write Only)

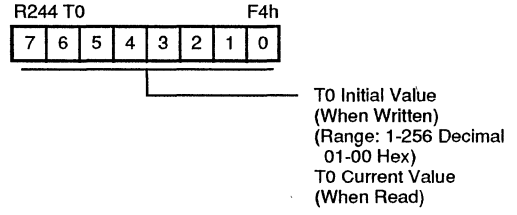


Figure 62. Counter/Timer 0 Register (F4H; Read/Write)

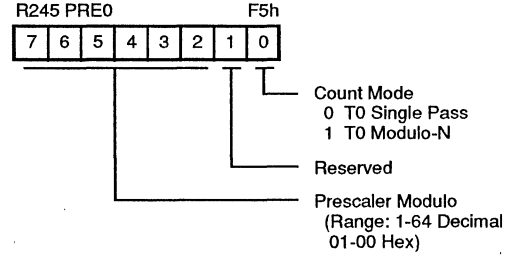


Figure 63 Prescaler 0 Register (F5H; Write Only)

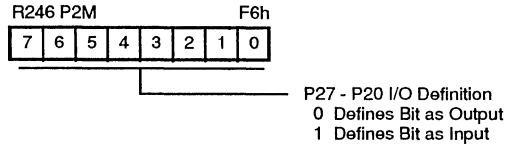


Figure 64. Port 2 Mode Register (F6H; Write Only)

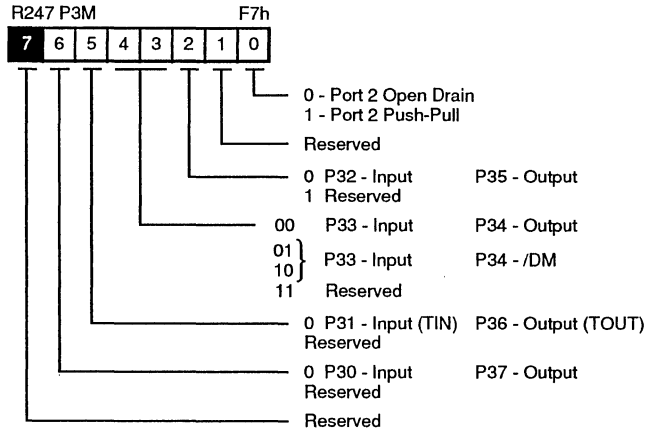


Figure 65. Port 3 Mode Register (F7H; Write Only)

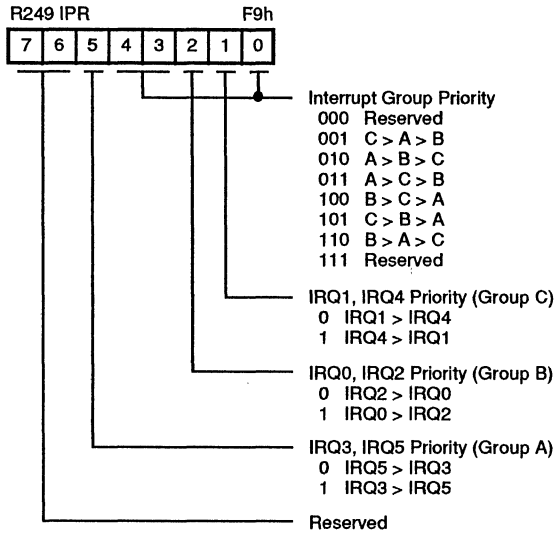


Figure 66. Interrupt Priority Register (F9H; Write Only)

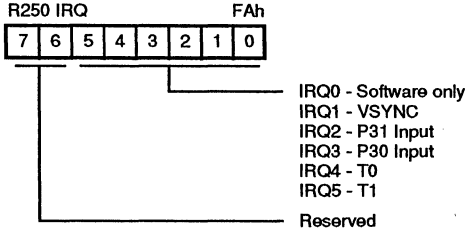


Figure 67. Interrupt Request Register (FAH; Read/Write)

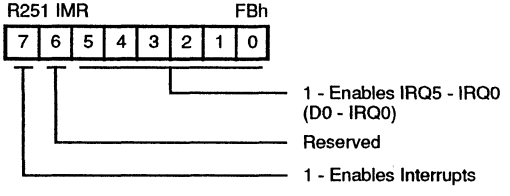


Figure 68. Interrupt Mask Register (FBH; Read/Write)

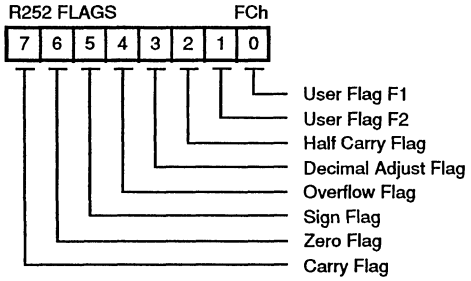


Figure 69. Flag Register (FCH; Read/Write)

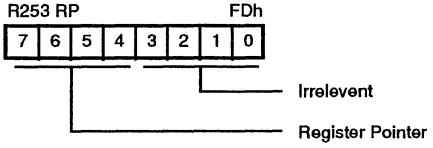


Figure 70. Register Pointer (FDH; Read/Write)

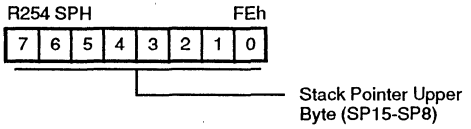


Figure 71. Stack Pointer (FEH; Read/Write)

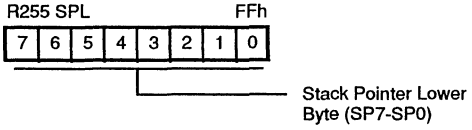


Figure 72. Stack Pointer (FFH; Read/Write)

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

Symbol	Meaning
IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

Symbol	Meaning
dst	Destination location or contents
src	Source location or contents
cc	Condition code
@	Indirect address prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag register (Control Register 252)
RP	Register Pointer (R253)
IMR	Interrupt mask register (R251)

Flags. Control register (R252) contains the following six flags:

Symbol	Meaning
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

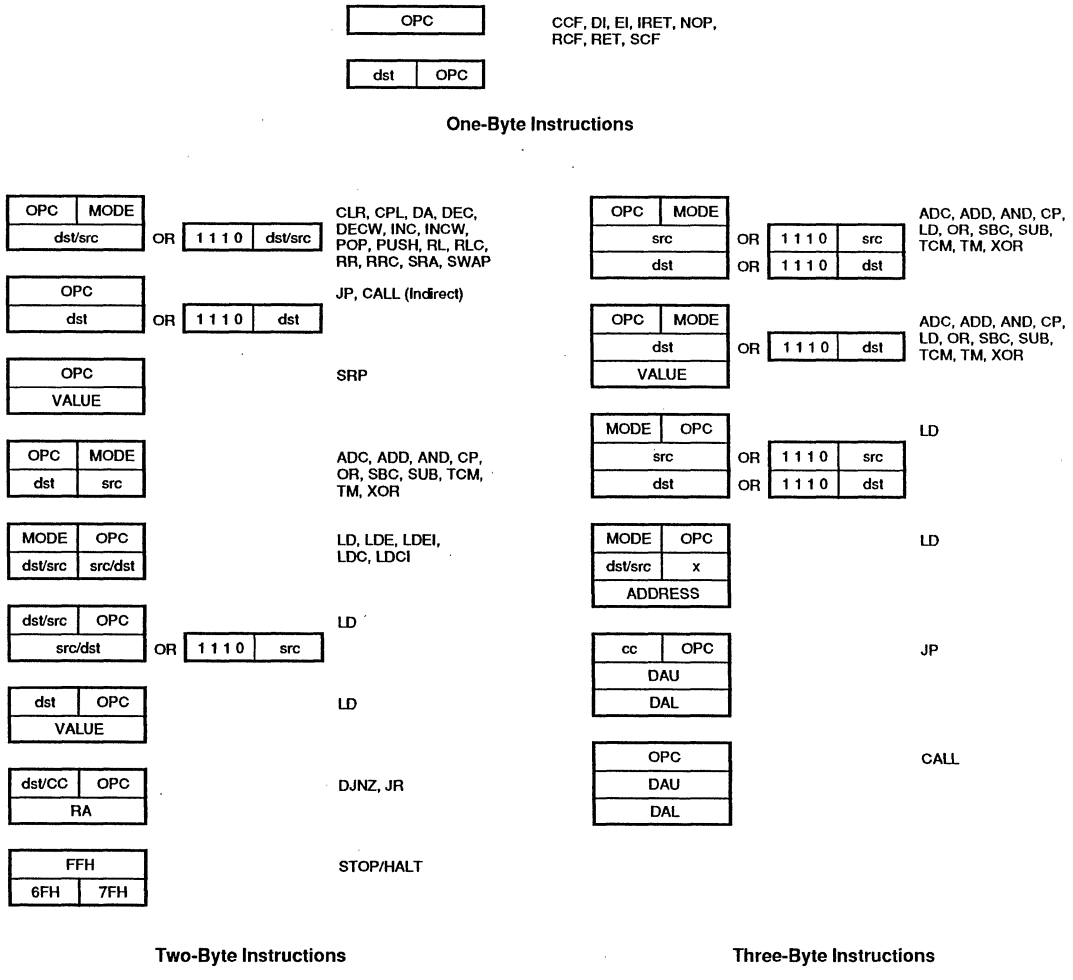
Affected flages are indicated by:

0	Clear to zero
1	Set to one
*	Set to clear according to operation
-	Unaffected
x	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000		Always True	
0111	C	Carry	C = 1
1111	NC	No Carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not Zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No Overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not Equal	Z = 0
1001	GE	Greater Than or Equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater Than	[Z OR (S XOR V)] = 0
0010	LE	Less Than or Equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned Greater Than or Equal	C = 0
0111	ULT	Unsigned Less Than	C = 1
1011	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned Less Than or Equal	(C OR Z) = 1
0000		Never True	

INSTRUCTION FORMATS



INSTRUCTION SUMMARY

Note: Assignment of a value is indicated by the symbol " ← ". For example:

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr (n)" is used to refer to bit (n) of a given operand location. For example:

dst (7)

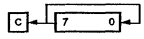
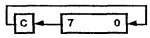
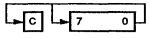
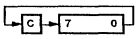
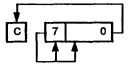
refers to bit 7 of the destination operand.

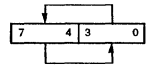
INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst, src dst←dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst←dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst←dst AND src	†		5[ ]	-	*	*	0	-	-
CALL dst SP←SP - 2 @SP←PC, PC←dst	DA IRR		D6 D4	-	-	-	-	-	-
CCF C←NOT C			EF	*	-	-	-	-	-
CLR dst dst←0	R IR		B0 B1	-	-	-	-	-	-
COM dst dst←NOT dst	R IR		60 61	-	*	*	0	-	-
CP dst, src dst - src	†		A[ ]	*	*	*	*	-	-
DA dst dst←DA dst	R IR		40 41	*	*	*	X	-	-
DEC dst dst←dst - 1	R IR		00 01	-	*	*	*	-	-
DECW dst dst←dst - 1	RR IR		80 81	-	*	*	*	-	-
DI IMR(7)←0			8F	-	-	-	-	-	-
DJNZr, dst r←r - 1 if r ≠ 0 PC←PC + dst Range: +127, -128	RA		rA r = 0 - F	-	-	-	-	-	-
EI IMR(7)←1			9F	-	-	-	-	-	-
HALT			7F	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
INC dst dst←dst + 1	r R IR		rE r = 0 - F 20 21	-	*	*	*	-	-
INCW dst dst←dst + 1	RR IR		A0 A1	-	*	*	*	-	-
IRET FLAGS←@SP; SP←SP + 1 PC←@SP; SP←SP + 2; IMR(7)←1			BF	*	*	*	*	*	*
JP cc, dst if cc is true PC←dst	DA IRR		cD c = 0 - F 30	-	-	-	-	-	-
JR cc, dst if cc is true, PC←PC + dst Range: +127, -128	RA		cB c = 0 - F	-	-	-	-	-	-
LD dst, src dst←src	r r R r X r Ir R R IR IR	Im R r X r Ir r R IR IM IM R	rC r8 r9 r = 0 - F C7 D7 E3 F3 E4 E5 E6 E7 F5	-	-	-	-	-	-
LDC dst, src	r	Irr	C2	-	-	-	-	-	-
LDCI dst, src dst←src r←r +1; rr←rr + 1	Ir	Irr	C3	-	-	-	-	-	-

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
NOP			FF	-	-	-	-	-	-
OR dst, src dst←dst OR src	†		4[ ]	-	*	*	0	-	-
POP dst dst←@SP; SP←SP + 1	R IR		50 51	-	-	-	-	-	-
PUSH src SP←SP - 1; @SP←src		R IR	70 71	-	-	-	-	-	-
RCF C←0			CF	0	-	-	-	-	-
RET PC←@SP; SP←SP + 2			AF	-	-	-	-	-	-
RL dst 	R IR		90 91	*	*	*	*	-	-
RLC dst 	R IR		10 11	*	*	*	*	-	-
RR dst 	R IR		E0 E1	*	*	*	*	-	-
RRC dst 	R IR		C0 C1	*	*	*	*	-	-
SBC dst, src dst←dst←src←C	†		3[ ]	*	*	*	*	1	*
SCF C←1			DF	1	-	-	-	-	-
SRA dst 	R IR		D0 D1	*	*	*	0	-	-
SRP src RP←src		Im	31	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
STOP			6F	-	-	-	-	-	-
SUB dst, src dst←dst←src	†		2[ ]	*	*	*	*	1	*
SWAP dst 	R IR		F0 F1	X	*	*	X	-	-
TCM dst, src (NOT dst) AND src	†		6[ ]	-	*	*	0	-	-
TM dst, src dst AND src	†		7[ ]	-	*	*	0	-	-
XOR dst, src dst←dst XOR src	†		B[ ]	-	*	*	0	-	-

† These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

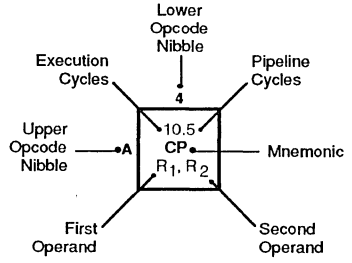
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address Mode dst	Address Mode src	Lower Opcode Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

OPCODE MAP

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								6.0 **WDH**
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								6.0 **WDT**
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								6.0 **STOP**
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								7.0 **HALT**
8	10.5 **DECW** RR1	10.5 **DECW** IR1	12.0 **LDE** r1, Irr2	18.0 **LDEI** Ir1, Irr2												6.1 **DI**
9	6.5 **RL** R1	6.5 **RL** IR1	12.0 **LDE** r2, Irr1	18.0 **LDEI** Ir2, Irr1												6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14.0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2				10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1	12.0 **LDC** r2, Irr1	18.0 **LDCI** Ir2, Irr1	20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6.5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2			3					2					3	1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_2 = Dst address
R_1 or r_2 = Src address

Sequence:
Opcode, First Operand,
Second Operand

Note: The blank are not defined.

* 2-byte instruction appears as a
3-byte instruction

ADVANCE INFORMATION SPECIFICATION

Z86C50

CMOS Z8® CCP™ ICE
IN-CIRCUIT EMULATOR

FEATURES

- 8-bit CMOS Z8 CCP base In-Circuit Emulation chip
- 124-pin PGA package
- Full Z8 CCP family instruction set
- 3.0 to 5.5 volt operation range
- Clock speed 20 MHz
- 236-byte general-purpose register
- Low EMI mode programmable
- Internal register read bus and write bus interface
- Register file bank pointer and register file address bus interface for accessing the Expanded Register File (ERF) externally.
- Expanded Register control signals (/REGRD, /REGWR and /CE_ERF) interface
- All internal control signals interface (i.e., /SYNC, /IACK, /MAS, /MDS, etc.)
- Disable Timers control signal
- Hold internal clock control signal
- Wait control signal
- STOP and HALT modes signal status outputs
- Two on-board analog comparators
- Two programmable 8-bit Counter/Timers, each with a 6-bit programmable prescaler
- Six vectored, priority interrupts from six different sources
- Internal program memory size select interface

GENERAL DESCRIPTION

The Z86C50 CCP In-Circuit Emulation (ICE) chip introduces a new level of sophistication to ICE architecture. The Z86C50 not only provides all the control signal interfaces, but also the internal register bus interface. The Expanded Register File (ERF) can be external and interface with the internal register bus and ERF control signals.

The Z86C50 allows users to prototype a system with an actual hardware device and to develop the code. Also, this device is very useful in emulator applications.

Figure 1 is the functional block diagram of Z86C50. This device is housed in a 124-pin PGA package (Figure 2). Table 1 is the pin identification of the Z86C50.

Note: All Signals with a preceding front slash, "/", are active Low, e.g.: B//W (WORD is active Low); /B/W (BYTE is active Low, only).

GENERAL DESCRIPTION (Continued)

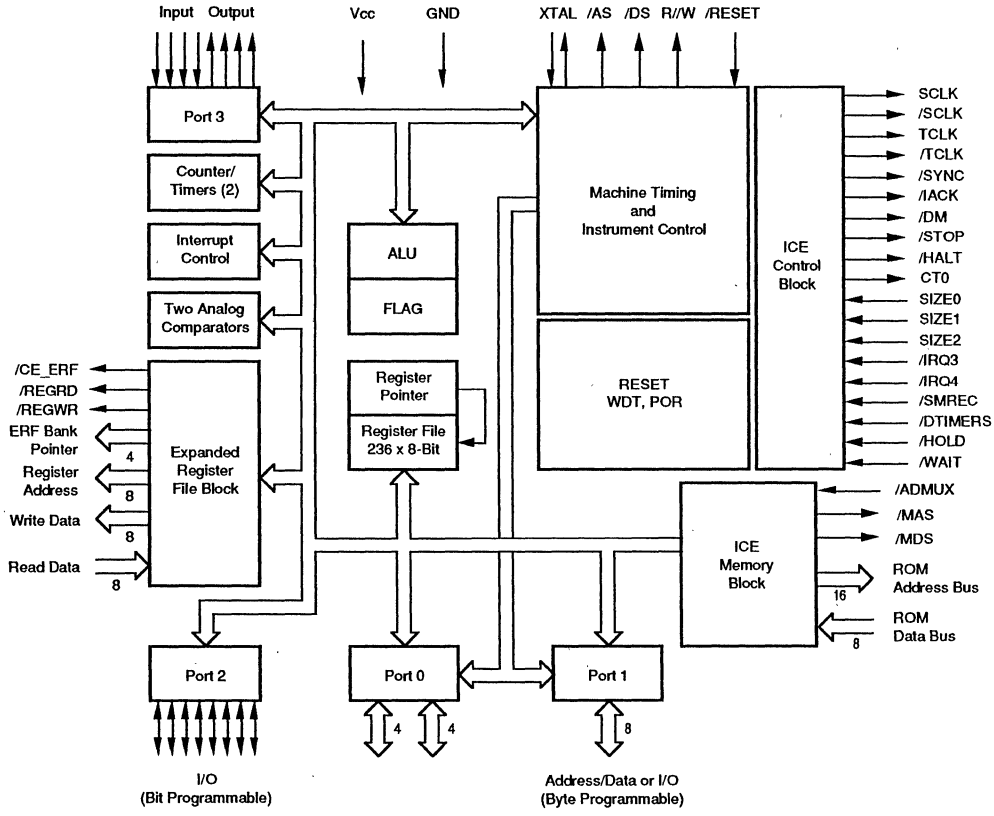


Figure 1. Functional Block Diagram

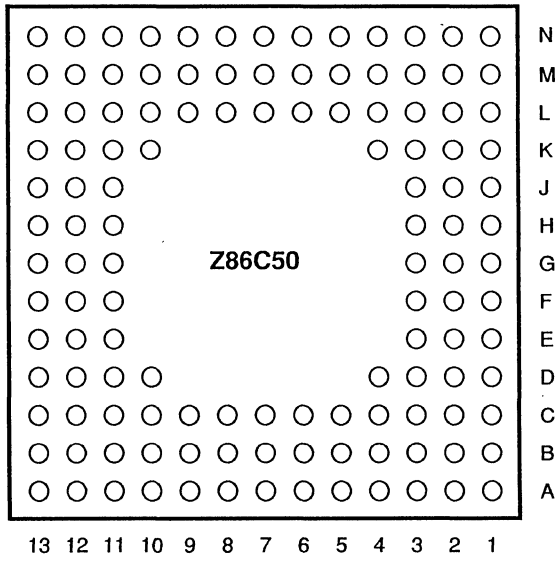


Figure 2. 124-Pin Grid Array Pin Assignments (Top View)

PIN DESCRIPTION

Table 1. Pin Identification

Pin #	Symbol	Function	Direction
C3	P30	Port 3 pin 0	Input
B2	P31	Port 3 pin 1	Input
B1	P32	Port 3 pin 2	Input
D3	P33	Port 3 pin 3	Input
C2	P34	Port 3 pin 4	Output
C1	P35	Port 3 pin 5	Output
D2	P36	Port 3 pin 6	Output
E3	V_{cc}	Power Supply	Input
D1	P37	Port 3 pin 7	Output
E2	P10	Port 1 pin 0	In/Output
E1	P11	Port 1 pin 1	In/Output
F3	P12	Port 1 pin 2	In/Output
F2	P13	Port 1 pin 3	In/Output
F1	GND	Ground, V_{ss}	Input
G2	P14	Port 1 pin 4	In/Output
G3	P15	Port 1 pin 5	In/Output
G1	P16	Port 1 pin 6	In/Output
H1	P17	Port 1 pin 7	In/Output
H2	/DM	Data Memory Strobe	Output
H3	GND	Ground	Input
J1	/RESET	Reset	Input
J2	/DTIMERS	Disable Timers	Input
K1	/HOLD	Hold Control	Input
J3	R//W	Read/Write	Output
K2	/DS	Data Strobe	Output
L1	/AS	Address Strobe	Output
M1	SCLK	System Clock	Output
K3	/SYNC	Instruction Sync. Signal	Output
L2	/IACK	Interrupt Acknowledge	Output
N1	/MAS	Interrupt Memory Address strobe	Output
K4	/WAIT	Wait Control	Input
L3	/MDS	Interrupt Memory Data Strobe	Output
M2	A15	Internal Address Line 15	Output
N2	A14	Internal Address Line 14	Output
L4	A13	Internal Address Line 13	Output
M3	A12	Internal Address Line 12	Output
N3	A11	Internal Address Line 11	Output
M4	A10	Internal Address Line 10	Output
L5	/SMR	Stop mode recovery	Input
N4	A9	Internal Address Line 9	Output
M5	A8	Internal Address Line 8	Output
N5	A7	Internal Address Line 7	Output
L6	A6	Internal Address Line 6	Output
M6	A5	Internal Address Line 5	Output
N6	A4	Internal Address Line 4	Output
M7	A3	Internal Address Line 3	Output

Pin #	Symbol	Function	Direction
L7	A2	Internal Address Line 2	Output
N7	GND	Ground, V_{SS}	Input
N8	A	Internal Address Line 1	Output
M8	A0	Internal Address Line 0	Output
L8	/ADMUX	Address/Data Bus Multiplex	Input
N9	GND	Ground	Input
M9	XTAL2	Oscillator Output	Output
N10	V_{CC}	Power Supply	Input
L9	/IRQ4	Interrupt Request 4	Input
M10	/IRQ3	Interrupt Request 3	Input
N11	XTAL1	Oscillator Input	Input
N12	RBP0	Reg. Bank Pointer bit 0	Output
L10	RBP1	Reg. Bank Pointer bit 1	Output
M11	RBP2	Reg. Bank Pointer bit 2	Output
N13	RBP3	Reg. Bank Pointer bit 3	Output
K10	/SCLK	System Clock (inverted)	Output
L11	IWD0	Internal Reg. Write Bus D0	Output
M12	IWD1	Internal Reg. Write Bus D1	Output
M13	IWD2	Internal Reg. Write Bus D2	Output
L12	IWD4	Internal Reg. Write Bus D4	Output
L13	IWD5	Internal Reg. Write Bus D5	Output
K12	IWD6	Internal Reg. Write Bus D6	Output
J11	IWD7	Internal Reg. Write Bus D7	Output
K13	/REGWR	Register Write	Output
J12	/REGRD	Register Read	Output
J13	GND	Ground, V_{SS}	Input
H11	RA0	Register Address Line 0	Output
H12	RA1	Register Address Line 1	Output
H13	RA2	Register Address Line 2	Output
G12	RA3	Register Address Line 3	Output
G11	RA4	Register Address Line 4	Output
G13	RA5	Register Address Line 5	Output
F13	RA6	Register Address Line 6	Output
F12	RA7	Register Address Line 7	Output
F11	CT0	Counter/Timer 0 Output	Output
E13	IRD0	Internal Reg. Read Bus D0	Input
E12	IRD1	Internal Reg. Read Bus D1	Input
D13	V_{CC}	Power Supply	Input
E11	IRD2	Internal Reg. Read Bus D2	Input
D12	IRD3	Internal Reg. Read Bus D3	Input
C13	IRD4	Internal Reg. Read Bus D4	Input
B13	IRD5	Internal Reg. Read Bus D5	Input
D11	IRD6	Internal Reg. Read Bus D6	Input
C12	IRD7	Internal Reg. Read Bus D7	Input
A13	TCLK	Timer Clock Output	Output
D10	/STOP	STOP Signal	Output
C11	/CE_ERF	ERF Chip Select	Output
B12	D0	Internal Memory Data Line 0	In/Output

PIN DESCRIPTION

Table 1. Pin Identification (Continued)

Pin #	Symbol	Function	Direction
A12	D1	Internal Memory Data Line 1	In/Output
C10	D2	Internal Memory Data Line 2	In/Output
B11	D3	Internal Memory Data Line 3	In/Output
A11	D4	Internal Memory Data Line 4	In/Output
B10	D5	Internal Memory Data Line 5	In/Output
C9	D6	Internal Memory Data Line 6	In/Output
A10	D7	Internal Memory Data Line 7	In/Output
B9	SIZE0	Size Select 0	Input
A9	SIZE1	Size Select 1	Input
C8	SIZE2	Size Select 2	Input
B8	P00	Port 0 Pin 0	In/Output
A8	P0	Port 0 Pin 1	In/Output
B7	P0	Port 0 Pin 2	In/Output
C7	P0	Port 0 Pin 3	In/Output
A7	GND	Ground, V_{SS}	Input
A6	P0	Port 0 Pin 4	In/Output
B6	P0	Port 0 Pin 5	In/Output
C6	P0	Port 0 Pin 6	In/Output
B5	P2	Port 2 Pin 0	In/Output
A4	V_{CC}	Power Supply	Input
C5	P2	Port 2 Pin 1	In/Output
B4	P2	Port 2 Pin 2	In/Output
A3	P2	Port 2 Pin 3	In/Output
A2	P2	Port 2 Pin 4	In/Output
C4	P2	Port 2 Pin 5	In/Output
B3	P2	Port 2 Pin 6	In/Output
A1	P2	Port 2 Pin 7	In/Output
D4	/HALT	HALT Signal	Output

PIN FUNCTIONS

P00-P07. Port 0, nibble programmable. Port 0 can be configured as input, output or address lines.

P10-P17. Port 1, byte programmable. Port 1 can be configured as input, output or multiplexed address/data lines.

P20-P27. Port 2, bit programmable. Port 2 can be configured as input or output lines.

P30-P33. Port 3 input lines, P31-P33 can be configured as analog input.

P34-P37. Port 3 output lines.

A0-A15. *Internal Memory Address Bus* (Output). The internal memory can address up to 32 Kbyte.

/ADMUX. *Address/Data Bus Multiplexed* (Input). When this pin is low, A7-A0 is address and data bus multiplexing (AD7-AD0). When this pin is high, A15-A0 outputs the address only. This pin has an internal pull-up resistor.

D7-D0 *(In/Output).* Internal Memory Data Bus.

R/W. Read/Write Signal output line.

SIZE0-SIZE2. *Internal Memory Size* (Input). These three pins select the internal memory size (Table 2).

Table 2. Memory Size Table

SIZE2	SIZE1	SIZE0	Memory Size
0	0	0	0 Kbyte
0	0	1	2 Kbytes
0	1	0	4 Kbytes
0	1	1	8 Kbytes
1	0	0	16 Kbytes
1	0	1	32 Kbytes

RBP0-RBP3. *Register Bank Pointer* (Output). The RBP points to the current register bank. The ERF is located in bank %1 to bank %F.

RA7-RA0. *Register Address Bus* (Output). This bus is used to access each of 256 byte registers of the current register bank.

IWD7-IWD0. *Register Write Bus* (Output). This bus is used to write the data to the register.

/REGWR. *Register Write signal* (Output). This pin goes low during each register write cycle.

IRD7-IRD0. *Register Read Bus* (Input). This bus is used to read the data from the register.

/REGRD. *Register Read signal* (Output). This pin goes low during each register read cycle.

/CE_ERF. *ERF Chip Enable* (Output). This pin goes active when the ERF is accessed and inactive when the on-board register file is accessed.

/DM. *Data Memory Strobe* (Output). This pin goes low during data memory access.

/MDS. *Internal Memory Data Strobe* (Output). This pin goes low during each internal memory fetch cycle.

/MAS. *Internal Memory Address Strobe* (Output). This pin goes low during each T1 cycle.

/DS. *Data Strobe* (Output). This pin goes low during each external memory fetch cycle.

/AS. *Address Strobe* (Output). This pin goes low during each T1 cycle.

PIN FUNCTIONS (Continued)

SCLK. System Clock output pin.

/SCLK. Inverse System Clock output pin.

/SYNC. *Instruction Sync Signal* (Output). This signal indicates the last clock of the current executing instruction.

/IACK. *Interrupt Acknowledge* (Output). This pin goes low during an interrupt cycle.

/IRQ3. Interrupt Request 3 input line.

/IRQ4. Interrupt Request 4 input line.

/DTIMERS. *Disable Timers* (Input). All timers (including the WDT) are stopped by the low level at this pin. This pin has an internal pull-up resistor.

/HOLD. *Hold* (Input). The internal clock is stopped by the low level at this pin. The on-board oscillator keeps on oscillating. This pin has an internal pull-up resistor.

/WAIT. *Wait* (Input). The code execution is stopped after completion of the currently executing instruction by the low level at this pin. The address of the fetching byte is latched. /AS goes high and /DS goes low. The high level at this pin releases from WAIT. This pin has an internal pull-up resistor.

/HALT. *HALT Mode Indication* (Output). The low level of this pin indicates the ICE chip in HALT mode.

/STOP. *STOP Mode Indication* (Output). The low level of this pin indicates the ICE chip in STOP mode.

/SMR. *STOP Mode Recovery* (Input). Low level at this pin wakes up the ICE chip from STOP mode.

TCLK. *Timer Clock Output lines.* The frequency of TCLK is the same as SCLK. TCLK drives the timer and interrupt logic.

CT0. Counter/Timer 0 output line.

TIMING DIAGRAMS

Figure 3 shows the configuration of ICE Control Register (ICECON) which selects the access timing of the Expanded Register File.

Figures 4 through 9 are the preliminary access timing diagrams of the register file.

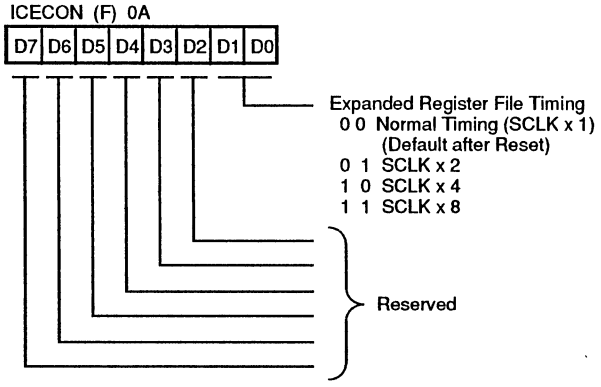


Figure 3. ICE Control Register

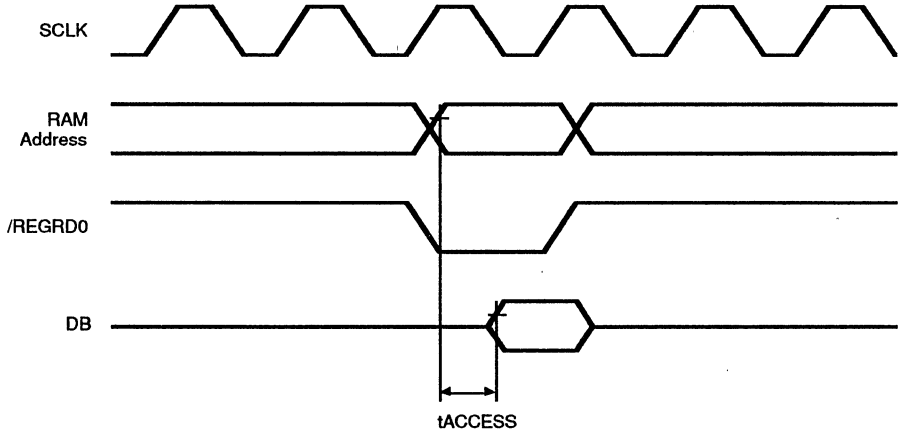


Figure 4. Internal Register File Read Cycle For Indirect Addressing

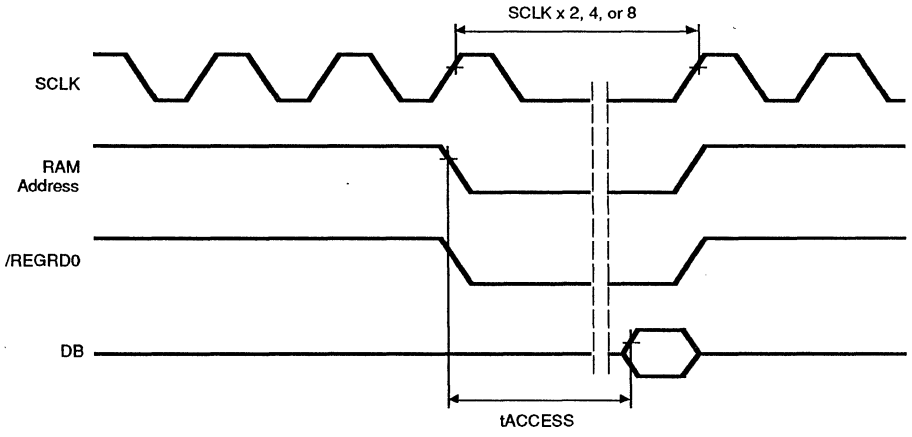


Figure 5. Internal Register File Read Cycle For Indirect Addressing (Extended Timing)

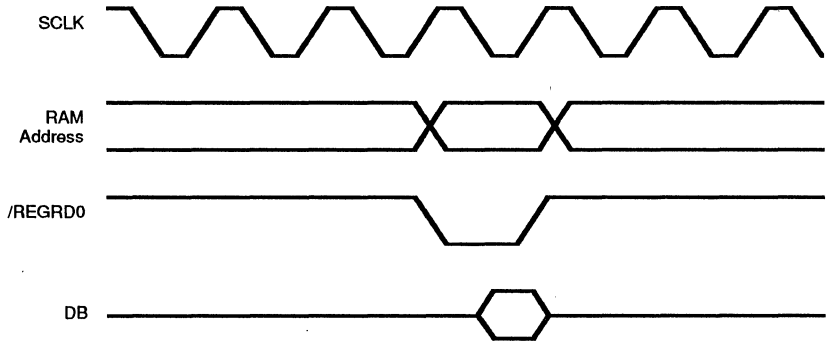


Figure 6. Internal Register File Write Cycle

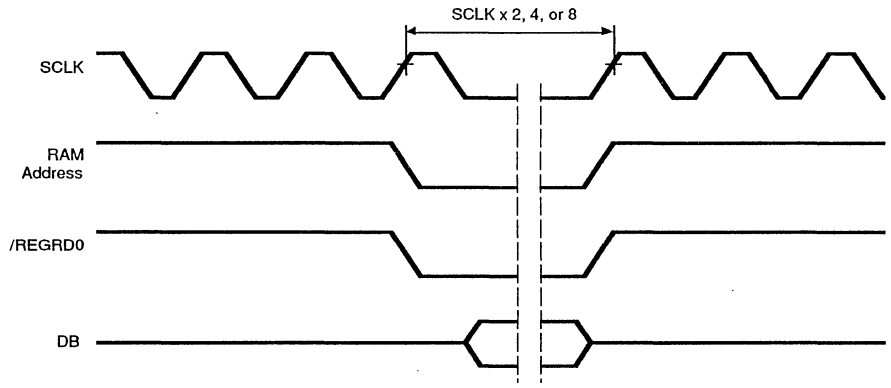


Figure 7. Internal Register File Write Cycle (Extended Timing)

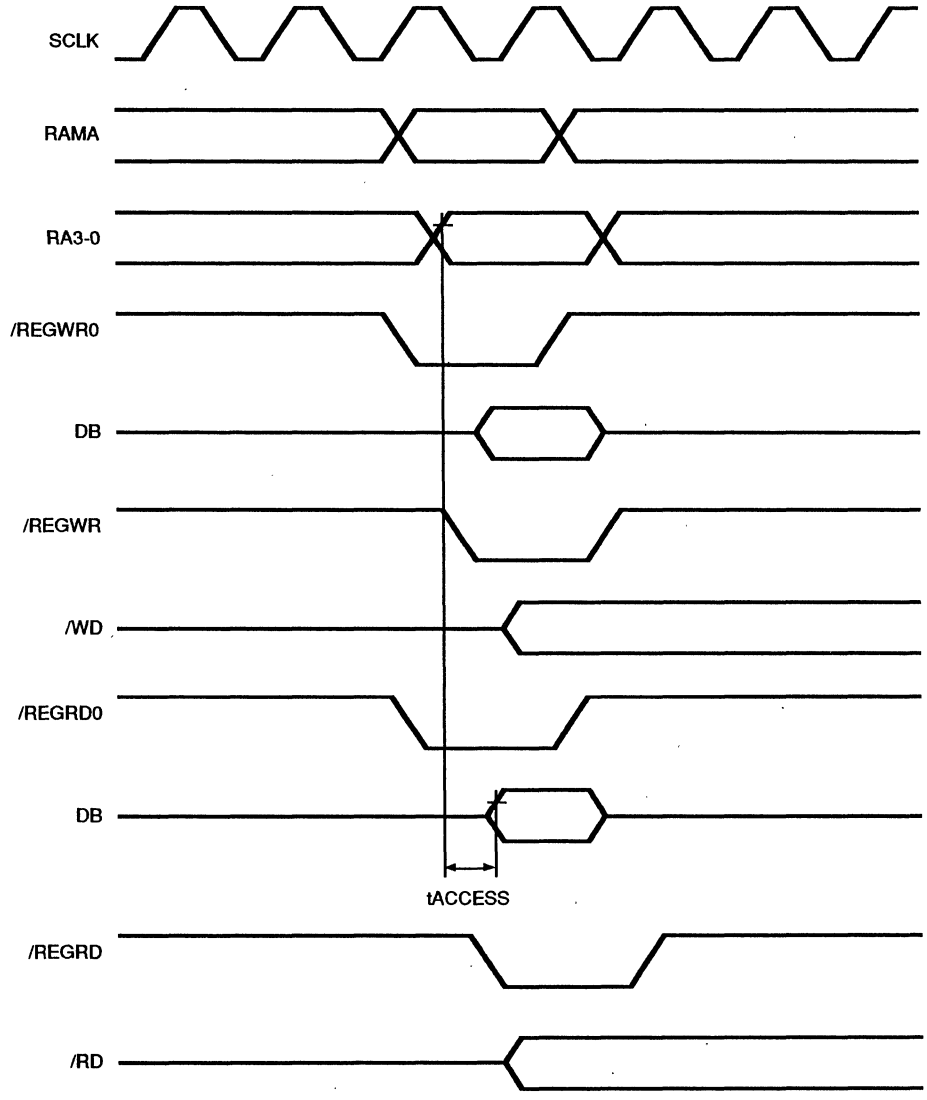


Figure 8. External Read And Write Timing (Normal Timing)

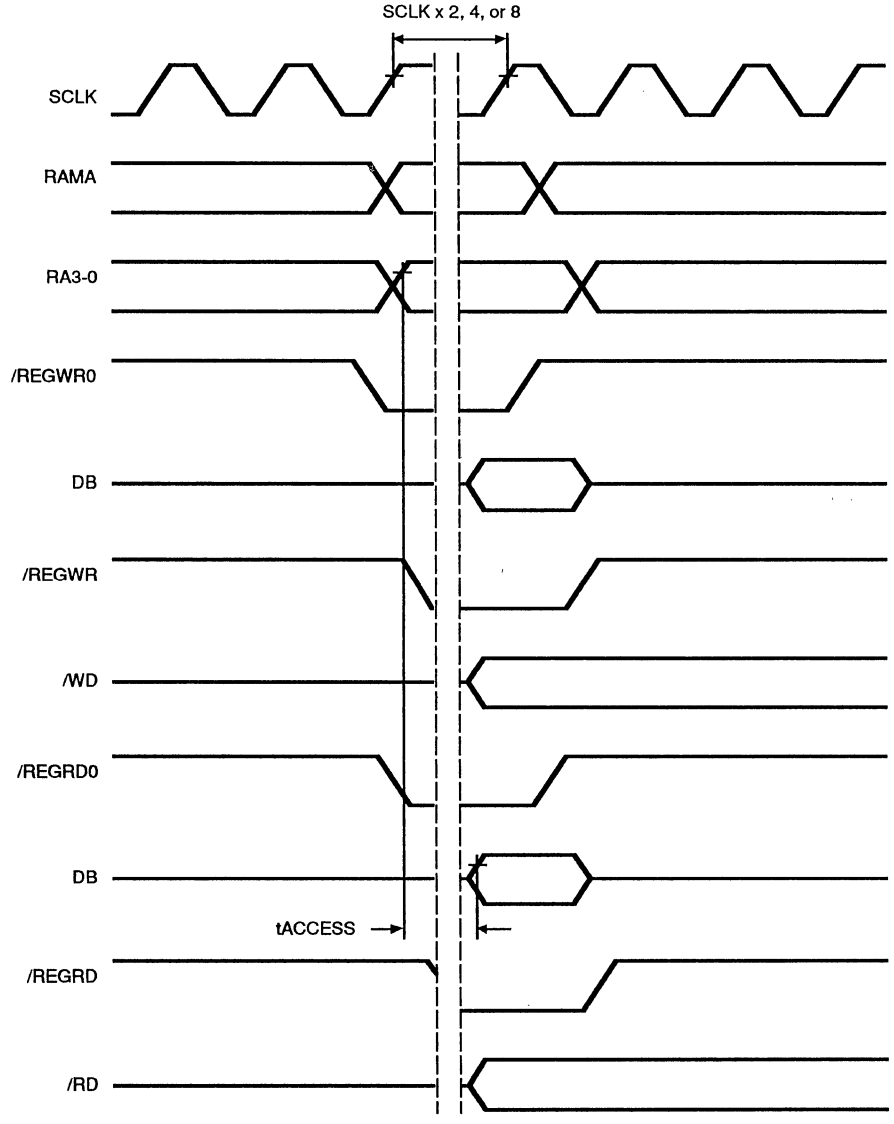


Figure 9. External Read And Write Timing (Extended Timing)

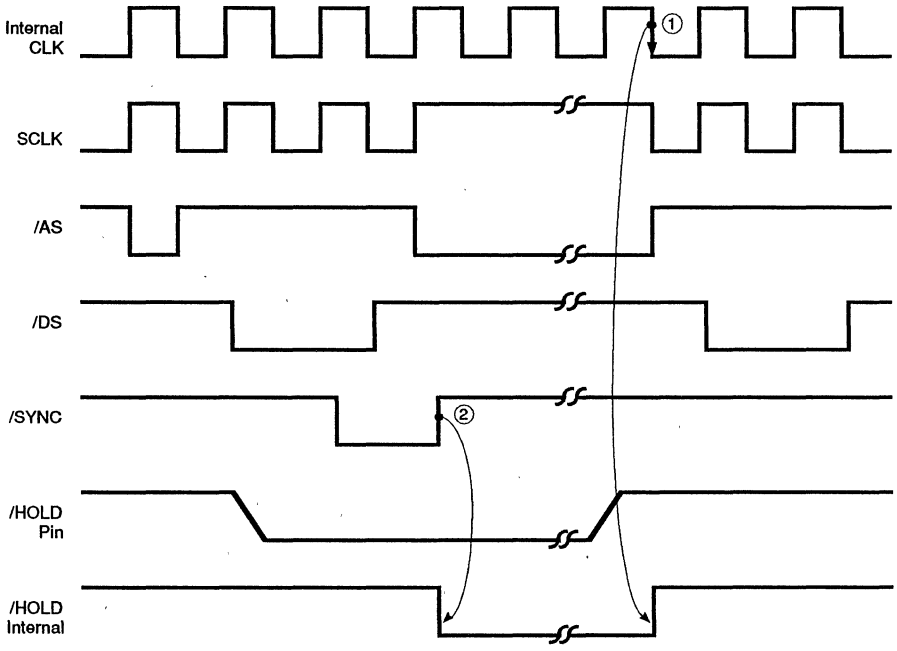
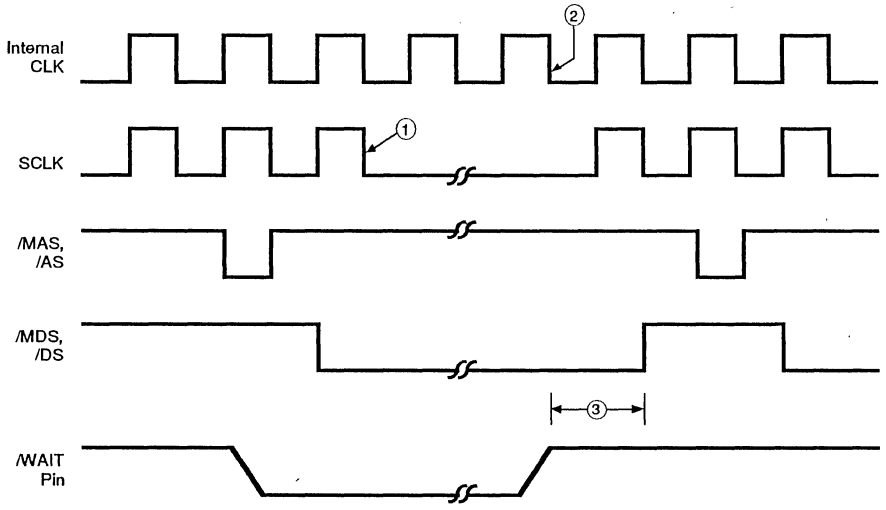
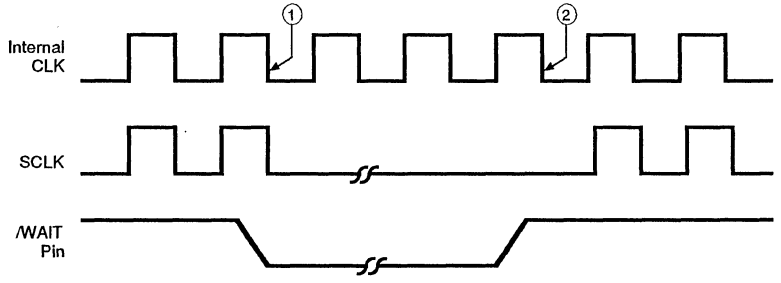


Figure 10. HOLD Timing Diagram



Notes:
1. WAIT Pin sampled during /MDS, /DS Low
2. WAIT Released
3. Remainder of /DS
TCLK not affected by /WAIT function

Figure 11. WAIT Timing Diagram



Notes:
1. /DTIMERS Pin Samples
2. /DTIMERS Released

Figure 12. DTIMERS Timing Diagram

GENERAL DESCRIPTION

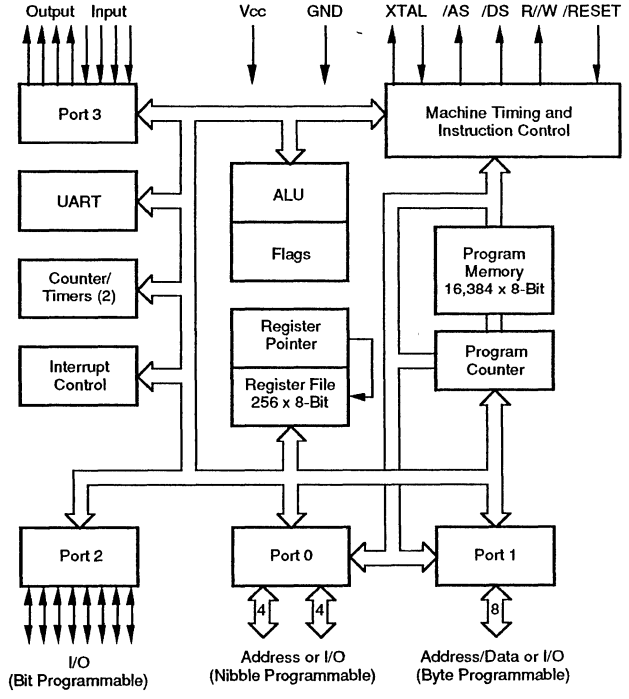
The Z86C61 is a member of the Z8 single-chip microcontroller family. It is pin compatible with the Z86C21 but has twice the on-board memory with 16K bytes of ROM. The device is housed in a 40-pin DIP, 44-pin Leaded Chip Carrier, or a 44-pin Quad Flat Pack. It offers all the outstanding features of the Z8 family architecture.

The Z86C61 provides up to 16 output address lines permitting an address space of up to 48K bytes of external program and data memory each. The 256-byte Register File consists of 236 general purpose registers, four I/O port registers, and 16 status and control registers.

The Z86C61 architecture is based on Zilog's 8-bit microcontroller core. The device has instruction compatibility with the entire Z8 family for easy software/hardware system expansion.

FEATURES

- *Complete microcomputer with 40-pin DIP, 44-pin PLCC, or 44-pin QFP, 32 I/O lines and 16K bytes of on-chip ROM.*
- *The register file is composed of 236 General Purpose registers, four I/O port registers and 16 control and status registers.*
- *Full duplex UART and two programmable 8-bit counter/timers, each with a 6-bit programmable prescaler.*
- *On-chip oscillator which accepts crystal, ceramic resonator, LC or external clock drive.*
- *RAM/ROM protect option*
- *Fast instruction pointer - 1 microsecond at 12 MHz*
- *Standby modes: STOP and HALT*
- *Clock speed 16 MHz*
- *1.2 micron CMOS technology*
- *3.0 to 5.5 volts operating range*
- *Six vectored, priority interrupts from eight different sources*



GENERAL DESCRIPTION

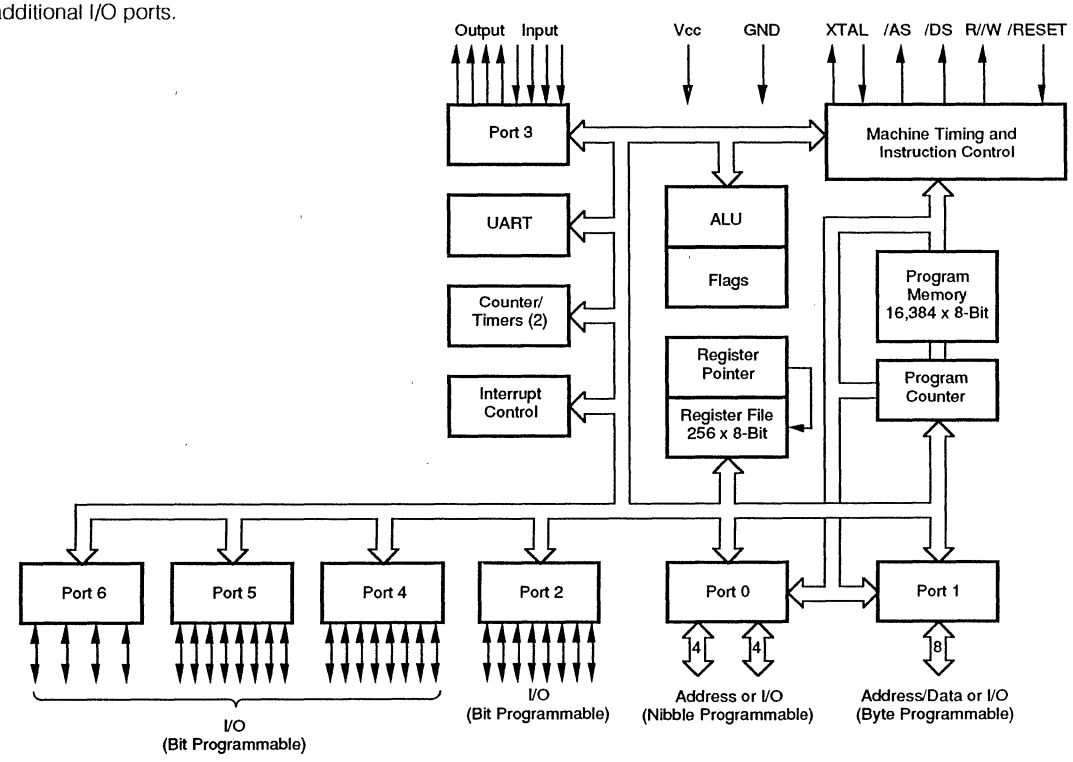
The Z86C62 is a member of the Z8 single-chip microcontroller family with 16K bytes of ROM. The device is housed in a 64-pin DIP or a 68-pin Leaded Chip Carrier. It offers all the outstanding features of the Z8 family architecture.

The Z86C62 provides up to 16 output address lines permitting an address space of up to 48K bytes of external program and data memory each. The 256-byte Register File consists of 236 general purpose registers, four I/O port registers, and 16 status and control registers. Three additional I/O port registers and five status and control registers reside in the Expanded Register File.

The Z86C62 architecture is based on Zilog's 8-bit microcontroller core. The device has instruction compatibility with the entire Z8 family for easy software/hardware system expansion. The Z86C62 is similar to the Z86C61, but with three additional I/O ports.

FEATURES

- *Complete microcontroller with 68-pin PLCC, or 64-pin DIP, 56 I/O lines and 16K bytes of on-chip ROM.*
- *The register file is composed of 236 General Purpose registers, four I/O port registers and 16 control and status registers.*
- *Three Expanded Register File I/O port registers and five control registers.*
- *Full duplex UART and two programmable 8-bit counter/timers, each with a 6-bit programmable prescaler.*
- *On-chip oscillator which accepts crystal, ceramic resonator, LC or external clock drive.*
- *Six vectored, priority interrupts from eight different sources*
- *RAM/ROM protect option*
- *Standby modes: STOP and HALT*
- *Clock speeds 16 MHz*
- *1.2 micron CMOS technology*
- *3.0 to 5.5 volts operating range*
- *All pins TTL compatible*



PRODUCT SPECIFICATION

Z86C90/C89

ROMLESS CMOS Z8® 8-BIT MICROCONTROLLER

FEATURES

- 8-bit CMOS microcomputer
- 40- or 44-pin package
- Low cost
- 3.0 to 5.5 volt operating range
- Low power consumption - 50 mW (Typical)
- Fast instruction pointer - 1.0 microsecond @ 12 MHz
- Two standby modes - STOP and HALT
- 32 input/output lines (two with comparator inputs)
- All digital inputs are CMOS levels, Schmitt triggered
- ROMless
- 256 bytes of RAM (236 for general purpose)
- Two Expanded Register File control registers
- Two programmable 8-bit Counter/Timers
- 6-bit programmable prescaler
- Six vectored, priority interrupts from six different sources
- Clock speeds 8 and 12 MHz for Z86C90, 8 Mhz only for Z86C89
- Brown-Out protection
- Programmable Watch Dog/Power-On Reset Timer
- Two Comparators with programmable interrupt polarity
- On-chip oscillator that accepts a crystal, ceramic resonator, LC, or external clock drive (Z86C90).
- On-chip oscillator that accepts an RC network or external clock drive (Z86C89).

GENERAL DESCRIPTION

The Z86C90/C89 CCP™ (Consumer Controller Processor) introduces a new level of sophistication to single-chip architecture. The Z86C90/C89 are ROMless members of the Z8 single-chip microcontroller family with 236 bytes of general purpose RAM. The only difference that exists between the Z86C89 and the Z86C90 is that the on-chip oscillator of the Z86C89 can accept an external RC network or other external clock source, while the Z86C90's on-chip oscillator accepts a crystal, ceramic resonator, LC, or external clock source drive. The CCP controllers are housed in a 40-pin DIP, 44-pin Leaded Chip Carrier, or a 44-pin Quad Flat Pack, and are CMOS compatible. The CCP offers the use of external memory which enables this Z8 microcomputer to be used where code flexibility is required. Zilog's CMOS microcomputer offers fast execution, efficient use of memory, sophisticated interrupts, input/output bit manipulation capabilities, and easy hardware/software system expansion along with low cost and low power consumption.

The Z86C90/C89 architecture is based on Zilog's 8-bit microcontroller core with an Expanded Register File to allow access to register mapped peripheral and I/O circuits. The CCP offers a flexible I/O scheme, an efficient register and address space structure, and a number of ancillary features that are useful in many industrial, automotive, computer peripherals, and advanced scientific applications.

The CCP applications demand powerful I/O capabilities. The Z86C90/C89 fulfills this with 32 pins dedicated to input and output. These lines are grouped into four ports. Each port consists of eight lines, and is configurable under software control to provide timing, status signals, parallel I/O with or without handshake, and an address/data bus for interfacing external memory.

GENERAL DESCRIPTION (Continued)

There are four basic address spaces available to support this wide range of configurations: Program Memory, Register File, Data Memory, and Expanded Register File. The Register File is composed of 236 bytes of general purpose registers, four I/O port registers, and fifteen control and status registers. The Expanded Register File consists of two control registers.

To unburden the program from coping with the real-time problems, such as counting/timing and data communication, the Z86C90/C89 offers two on-chip counter/timers. Included are a large number of user selectable modes, and two on-board comparators to process analog signals with a common reference voltage (Figure 1).

Note: All Signals with a preceding front slash, "/", are active Low, e.g.: B//W (WORD is active Low); /B/W (BYTE is active Low, only); /N//S (NORMAL and SYSTEM are both active Low).

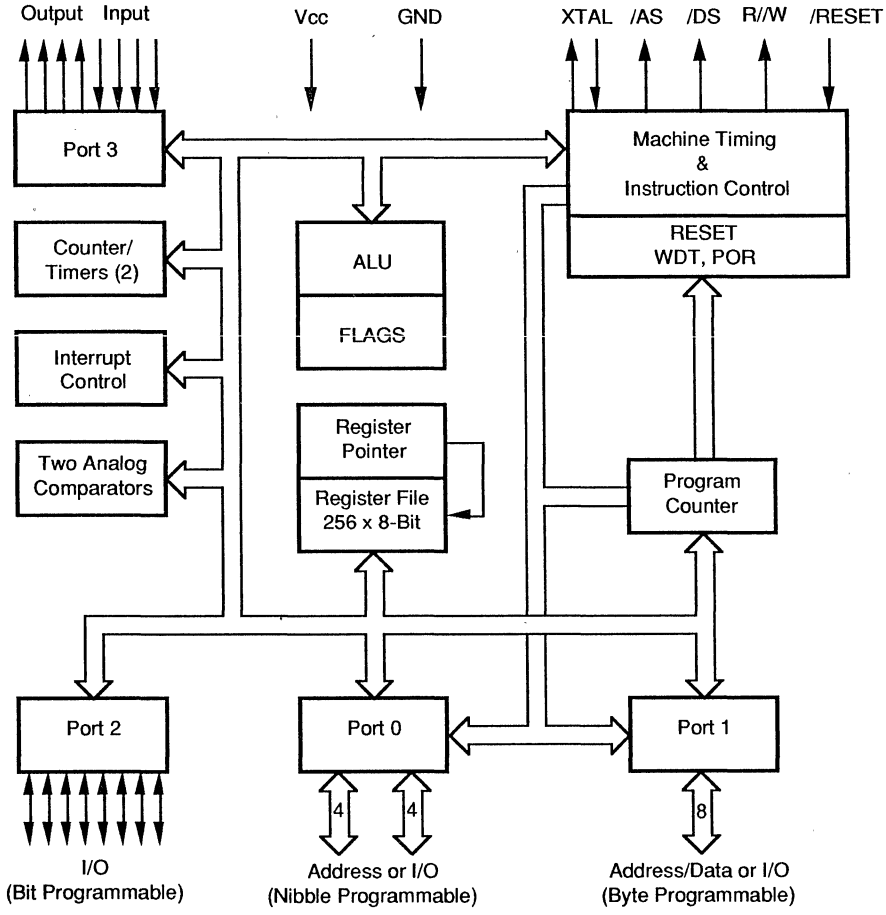


Figure 1. Functional Block Diagram

PIN DESCRIPTION

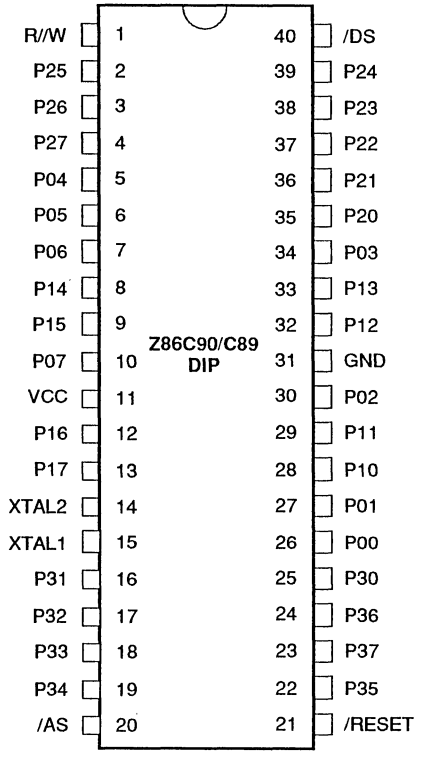


Figure 2. 40-Pin Dual-In-Line Pin Assignments

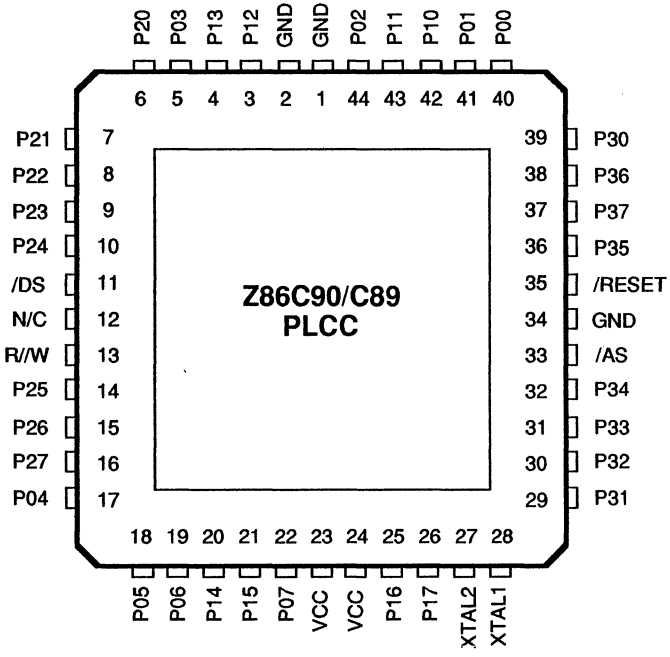


Figure 3. 44-Pin Leaded Chip Carrier Pin Assignments

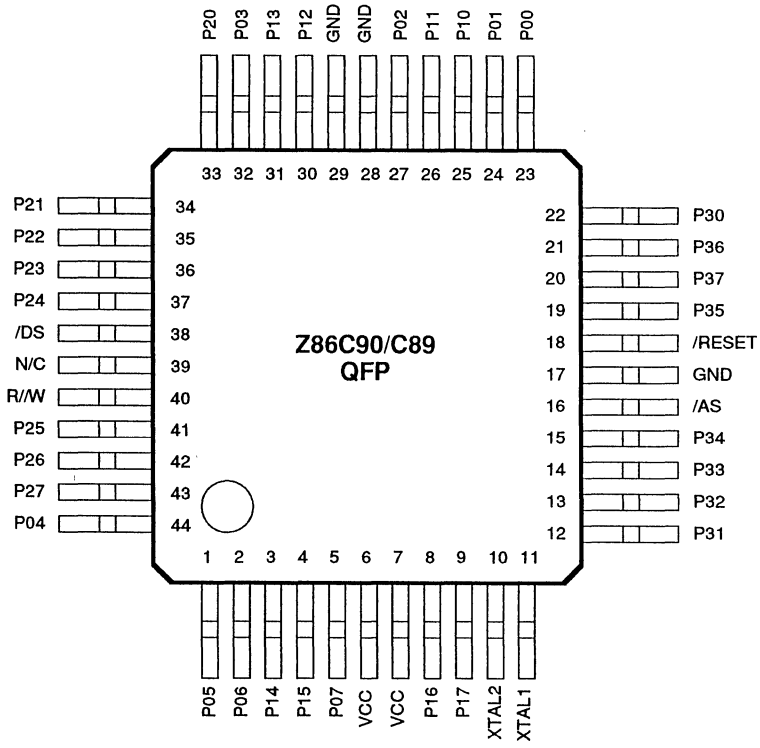


Figure 4. 44-Pin Quad Flat Pack Pin Assignments

PIN DESCRIPTION (Continued)

Table 1. 40-Pin Dual-In-Line Pin Identification

Pin #	Symbol	Function	Direction
1	R//W	Read/Write	Output
2-4	P25-27	Port 2 pin 5,6,7	In/Output
5-7	P04-P06	Port 0 pin 4,5,6	In/Output
8-9	P14-P15	Port 1 pin 4,5	In/Output
10	P07	Port 0 pin 7	In/Output
11	V_{cc}	Power Supply	Input
12-13	P16-P17	Port 1 pin 6,7	In/Output
14	XTAL2	Crystal, Oscillator Clock	Output
15	XTAL1	Crystal, Oscillator Clock	Input
16-18	P31-P33	Port 3 pin 1,2,3	Input
19	P34	Port 3 pin 4	Output
20	/AS	Address Strobe	Output
21	/RESET	Reset	Input
22	P35	Port 3 pin 5	Output
23	P37	Port 3 pin 7	Output
24	P36	Port 3 pin 6	Output
25	P30	Port 3 pin 0	Input
26-27	P00-P01	Port 0 pin 0,1	In/Output
28-29	P10-P11	Port 1 pin 0,1	In/Output
30	P02	Port 0 pin 2	In/Output
31	GND	Ground	Input
32-33	P12-P13	Port 1 pin 2,3	In/Output
34	P03	Port 0 pin 3	In/Output
35-39	P20-P24	Port 2 pin 0,1,2,3,4	In/Output
40	/DS	Data Strobe	Output

Table 2. 44-Pin Leaded Chip Carrier Pin Identification

Pin #	Symbol	Function	Direction
1-2	GND	Ground	Input
3-4	P12-P13	Port 1 pin 2,3	In/Output
5	P03	Port 0 pin 3	In/Output
6-10	P20-P24	Port 2 pin 0,1,2,3,4	In/Output
11	/DS	Data Strobe	Output
12	N/C	Not Connected	Input
13	R//W	Read/Write	Output
14-16	P25-27	Port 2 pin 5,6,7	In/Output
17-19	P04-P06	Port 0 pin 4,5,6	In/Output
20-21	P14-P15	Port 1 pin 4,5	In/Output
22	P07	Port 0 pin 7	In/Output
23-24	V_{cc}	Power Supply	Input
25-26	P16-P17	Port 1 pin 6,7	In/Output
27	XTAL2	Crystal, Oscillator Clock	Output
28	XTAL1	Crystal, Oscillator Clock	Input
29-31	P31-P33	Port 3 pin 1,2,3	Input
32	P34	Port 3 pin 4	Output
33	/AS	Address Strobe	Output
34	GND	Ground	Input
35	/RESET	Reset	Input
36	P35	Port 3 pin 5	Output
37	P37	Port 3 pin 7	Output
38	P36	Port 3 pin 6	Output
39	P30	Port 3 pin 0	Input
40-41	P00-P01	Port 0 pin 0,1	In/Output
42-43	P10-P11	Port 1 pin 0,1	In/Output
44	P02	Port 0 pin 2	In/Output

PIN DESCRIPTION (Continued)

Table 3. 44-Pin Quad Flat Pack Pin Identification

Pin #	Symbol	Function	Direction
1-2	P05-P06	Port 0 pin 5,6	In/Output
3-4	P14-P15	Port 1 pin 4,5	In/Output
5	P07	Port 0 pin 7	In/Output
6-7	V_{CC}	Power Supply	Input
8-9	P16-P17	Port 1 pin 6,7	In/Output
10	XTAL2	Crystal, Oscillator Clock	Output
11	XTAL1	Crystal, Oscillator Clock	Input
12-14	P31-P33	Port 3 pin 1,2,3	Input
15	P34	Port 3 pin 4	Output
16	/AS	Address Strobe	Output
17	GND	Ground	Input
18	/RESET	Reset	Input
19	P35	Port 3 pin 5	Output
20	P37	Port 3 pin 7	Output
21	P36	Port 3 pin 6	Output
22	P30	Port 3 pin 0	Input
23-24	P00-P01	Port 0 pin 0,1	In/Output
25-26	P10-P11	Port 1 pin 0,1	In/Output
27	P02	Port 0 pin 2	In/Output
28-29	GND	Ground	Input
30-31	P12-P13	Port 1 pin 2,3	In/Output
32	P03	Port 0 pin 3	In/Output
33-37	P20-P24	Port 2 pin 0,1,2,3,4	In/Output
38	/DS	Data Strobe	Output
39	N/C	Not Connected	Input
40	R//W	Read/Write	Output
41-43	P25-27	Port 2 pin 5,6,7	In/Output
44	P04	Port 0 pin 4	In/Output

PIN FUNCTIONS

/DS. *(output, active Low).* Data Strobe is activated once for each external memory transfer. For a READ operation, data must be available prior to the trailing edge of /DS. For WRITE operations, the falling edge of /DS indicates that output data is valid.

/AS. *(output, active Low).* Address Strobe is pulsed once at the beginning of each machine cycle. Address output is via Port 0/Port 1 for all external programs. Memory address transfers are valid at the trailing edge of /AS. Under program control, /AS is placed in the high-impedance state along with Ports 0 and 1, Data Strobe, and Read/Write.

XTAL1. *Crystal 1* (time-based input). This pin connects a parallel-resonant crystal, ceramic resonator, LC, or RC network or an external single-phase clock to the on-chip oscillator input.

XTAL2. *Crystal 2* (time-based output). This pin connects a parallel-resonant, crystal, ceramic resonant, LC, or RC network to the on-chip oscillator output.

R/W. *(output, write Low).* Read/Write, the R/W signal is low when the CCP is writing to the external program or data memory.

Port 0 (P00-P07). Port 0 is an 8-bit, bidirectional, CMOS compatible port. These eight I/O lines are configured under software control as a nibble I/O port, or as an address port for interfacing external memory. The input buffers are Schmitt triggered and the output drivers are push-pull. Port 0 is placed under handshake control. In this configuration, Port 3, lines P32 and P35 are used as the handshake control /DAV0 and RDY0. Handshake signal direction is dictated by the I/O direction to Port 0 of the upper nibble P04-P07. The lower nibble must have the same direction as the upper nibble.

For external memory references, Port 0 can provide address bits A11-A8 (lower nibble) or A15-A8 (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. If one or both nibbles are needed for I/O operation, they must be configured by writing to the Port 0 mode register. After a hardware reset, Port 0 is configured as address lines A15-A8, and extended timing is set to accommodate slow memory access. The initialization routine can include reconfiguration to eliminate this extended timing mode.

Port 0 is set in the high-impedance mode if selected as an address output state along with Port 1 and the control signals /AS, /DS and R/W (Figure 5).

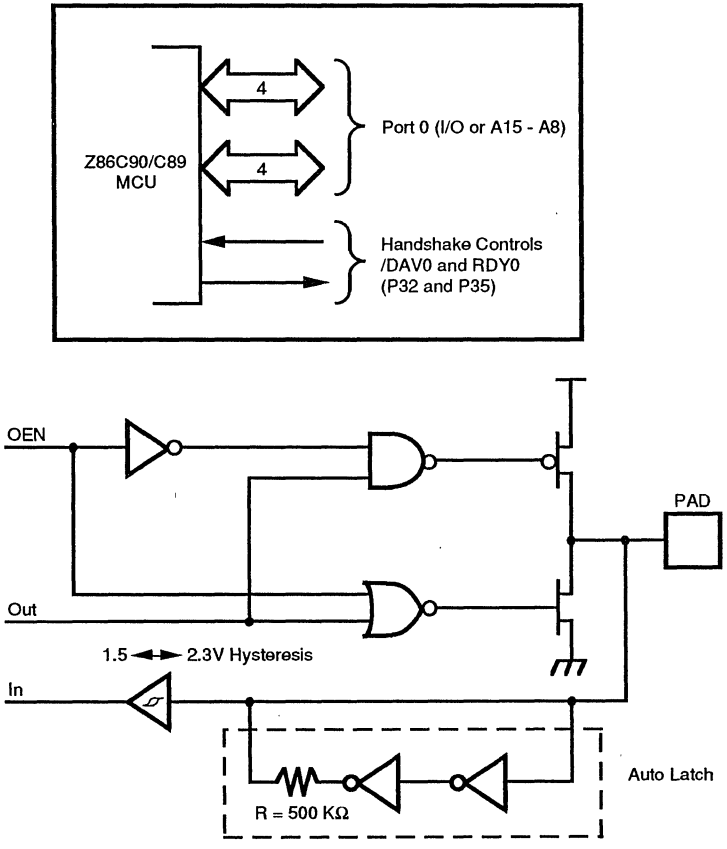


Figure 5. Port 0 Configuration

PIN FUNCTIONS (Continued)

Port 1 (P10-P17). Port 1 is a multiplexed Address (A7-A0) and Data (D7-D0), CMOS compatible port. Port 1 is dedicated to the Zilog ZBUS®-compatible memory interface. The operations of Port 1 are supported by the Address Strobe (/AS) and Data Strobe (/DS) lines, and by the Read/Write (R//W) and Data Memory (/DM) control lines. Data memory read/write operations are done through this port (Figure 6). If more than 256 external locations are required, Port 0 outputs the additional lines.

Port 1 can be placed in the high-impedance state along with Port 0, /AS, /DS and R//W, allowing the Z86C90/C89 to share common resources in multiprocessor and DMA applications.

The CCP wakes up with the 8 bits of Port 1 configured as address outputs for external memory.

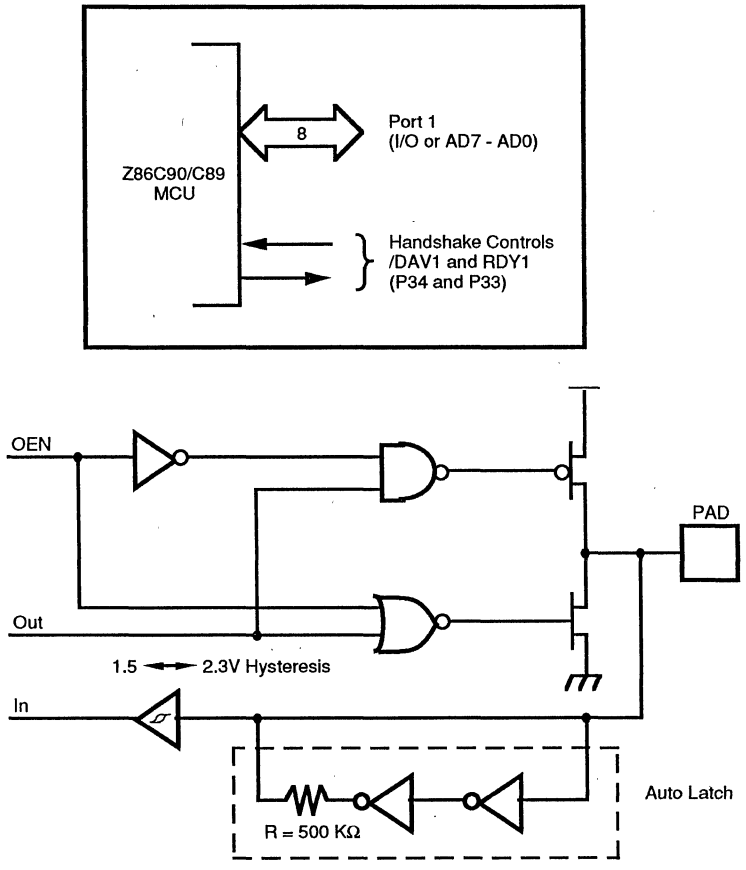


Figure 6. Port 1 Configuration

Port 2 (P20-P27). Port 2 is an 8-bit, bidirectional, CMOS compatible I/O port. These eight I/O lines can be configured under software control as an input or output, independently. Port 2 is always available for I/O operation. The input buffers are Schmitt triggered. Bits programmed as outputs are globally programmed as either push-pull or open-drain. Port 2 may be placed under handshake control. In this configuration, Port 3 lines, P31 and P36 are used as the handshake controls lines /DAV2 and RDY2. The handshake signal assignment for Port 3, lines P31 and P36 is dictated by the direction (input or output) assigned to bit-7 Port 2 (Figure 7).

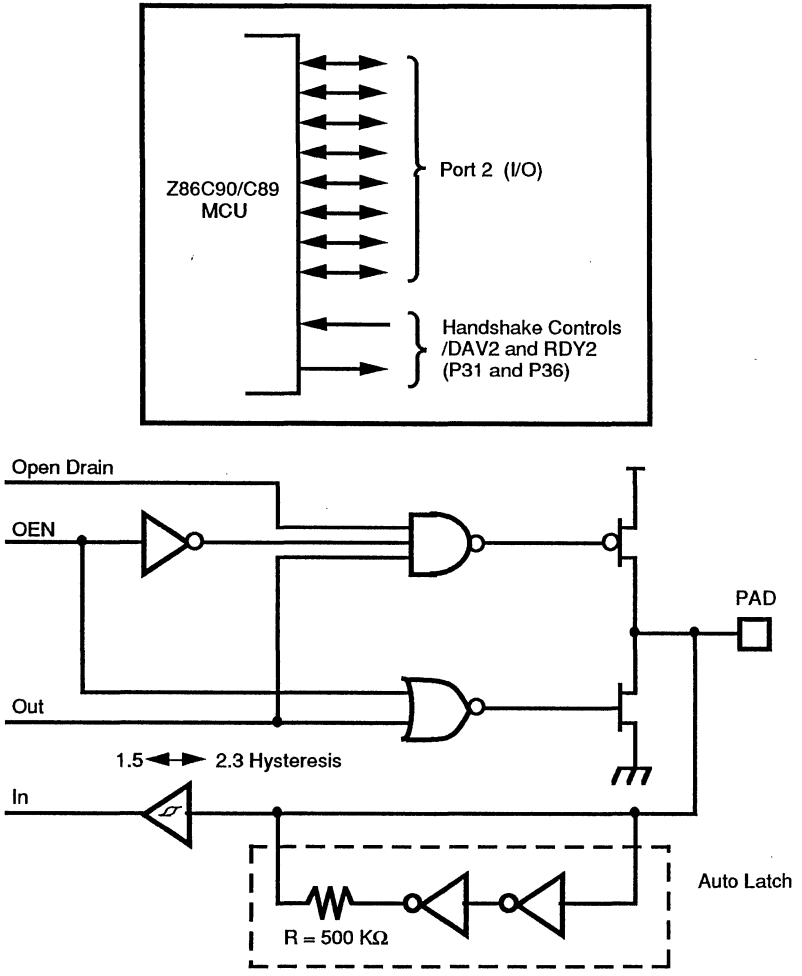


Figure 7. Port 2 Configuration

Port 3 (P30-P37). Port 3 is an 8-bit, CMOS compatible four-fixed input and four-fixed output. Port 3 consists of four-fixed input (P30-P33) and four-fixed output (P34-P37), and can be configured under software control for Input/Output, Counter/Timers, interrupt, Port handshake and Data Memory functions. Port 3, Pin-0 input is Schmitt triggered, and pins P31, P32, and P33 are standard CMOS inputs; outputs are push-pull.

PIN FUNCTIONS (Continued)

Two on-board comparators process analog signals on P31 and P32 with reference to the voltage on P33. The analog function is enabled by programming the Port 3 Mode Register (bit-1). Port 3, pins 0 and 3 are falling edge interrupt inputs. P31 and P32 are programmable as rising, falling, or both edge triggered interrupts (IRQ register bits 6 and 7). P33 is the comparator reference voltage input. Access to Counter/Timer 1 is made through P31 (T_{IN}) and P36 (T_{OUT}). Handshake lines Ports 0, 1 and 2 are available on P31 through P36.

Port 3 also provides the following control functions: handshake for Ports 0, 1 and 2 (/DAV and RDY); four external interrupt request signals (IRQ0-IRQ3); timer input and output signals (T_{IN} and T_{OUT}) and Data Memory Select [(/DM) (Figure 8)].

Auto-Latch. The Auto-Latch puts valid CMOS levels on all CMOS inputs (except P31-P33) that are not externally driven. Whether this level is zero or one, cannot be determined. A valid CMOS level, rather than a floating node, reduces excessive supply current flow in the input buffer.

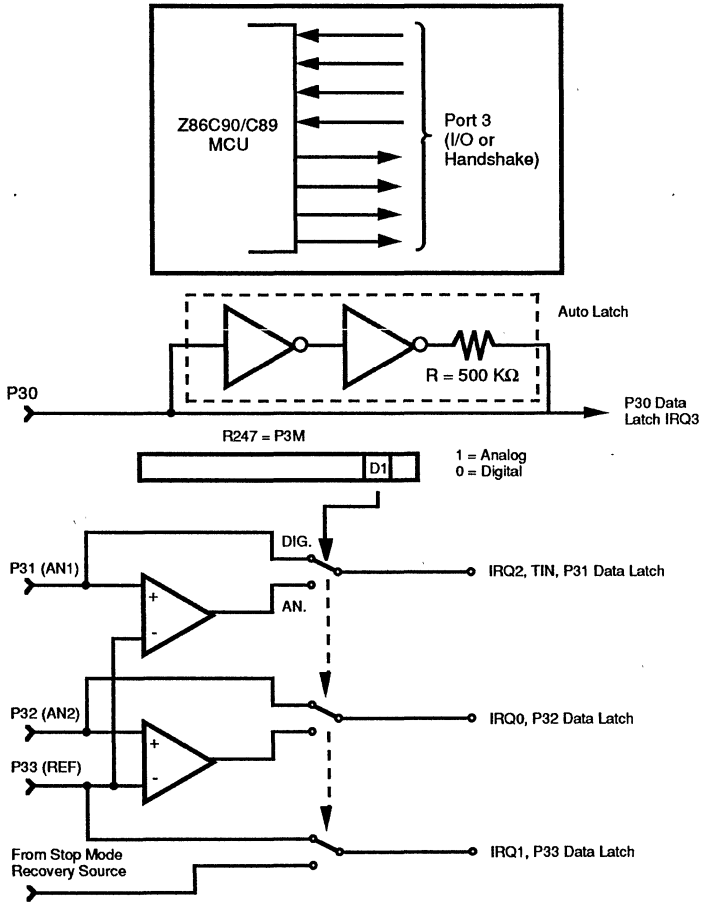


Figure 8. Port 3 Configuration

Table 4. Pin Assignments

Pin	I/O	CTC1	AN IN	Int.	P0 HS	P1 HS	P2 HS	Ext
P30	IN			IRQ3				
P31	IN	T_{IN}	AN1	IRQ2			D/R	
P32	IN		AN2	IRQ0	D/R			
P33	IN		REF	IRQ1		D/R		
P34	OUT					R/D		DM
P35	OUT				R/D			
P36	OUT	T_{OUT}					R/D	
P37	OUT							

Notes:
HS = Handshake Signals
D = DAV
R = RDY

Comparator Inputs. Port 3, Pins P31 and P32 each have a comparator front end. The comparator reference voltage (Pin P33) is common to both comparators. In analog mode, P31 and P32 are the positive inputs to the comparators and P33 is the reference voltage supplied to both comparators. In digital mode, Pin P33 can be used as a P33 register input or IRQ1 source.

/RESET. *(input, active-Low).* Initializes the MCU. Reset is accomplished either through Power-On, Watch Dog Timer reset, STOP Mode Recovery, or external reset. During Power-On Reset and Watch Dog Reset, the internally generated reset drives the reset pin low for the POR time. Any devices driving the reset line should be open-drain in order to avoid damage from a possible conflict during reset conditions. Pull-up is provided internally.

After the POR time, /RESET is a Schmitt triggered input. To avoid asynchronous and noisy reset problems, the Z86C90/C89 is equipped with a reset filter of four external clocks (4TpC). If the external reset signal is less than 4TpC in duration, no reset occurs. On the fifth clock after the reset is detected, an internal RST signal is latched and held for an internal register count of 18 external clocks, or for the duration of the external reset, whichever is longer.

During the reset cycle, /DS is held active low while /AS cycles at a rate of TpC/2. Program execution begins at location 000C (HEX), 5-10 TpC cycles after the RST is released. For Power-On Reset, the typical reset output time is 5 ms. The Z86C90/C89 does not reset WDTMR, SMR, P2M, or P3M registers on a STOP Mode Recovery operation.

FUNCTIONAL DESCRIPTION

The Z8 CCP incorporates special functions to enhance the Z8's functionality in industrial, scientific research and advanced technologies applications.

Reset. The device is reset in one of the following conditions:

- Power-On Reset
- Watch-Dog Timer
- STOP Mode Recovery Source
- Brown-out Recovery
- External Reset

Program Memory. The Z86C90/C89 addresses up to 64K external program memory (Figure 9). The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts. Program execution begins at location 000C (HEX) after a reset.

FUNCTIONAL DESCRIPTION

Address Space

The following subsections define Program Memory, Data Memory, Register Files, and Stack Pointers.

Program Memory. The Z86C91 can address up to 64 Kbytes of external program memory (Figure 10). The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts. Program execution begins at external location 000CH after a reset.

Data Memory (/DM). The Z86C91 addresses up to 64 Kbytes of external data memory space. External data memory is included with, or separated from, the external program memory space. /DM, an optional I/O function that can be programmed to appear on pin P34, is used to distinguish between data and program memory space (Figure 11). The state of the /DM signal is controlled by the type instruction being executed. An LDC opcode references PROGRAM (/DM inactive) memory, and an LDE instruction references DATA (/DM active low) memory.

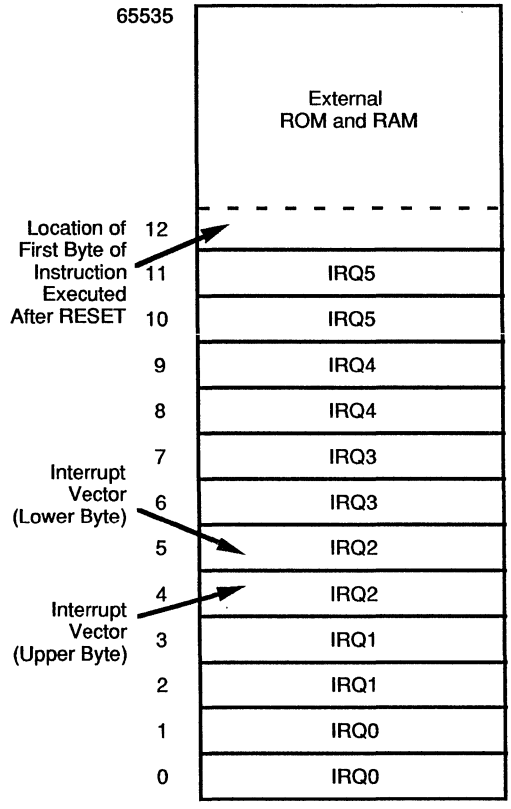


Figure 10. Program Memory Configuration

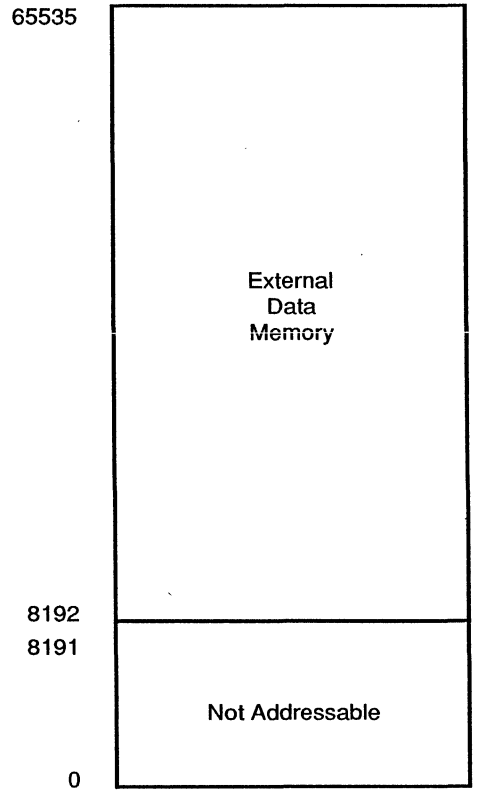


Figure 11. Data Memory Configuration

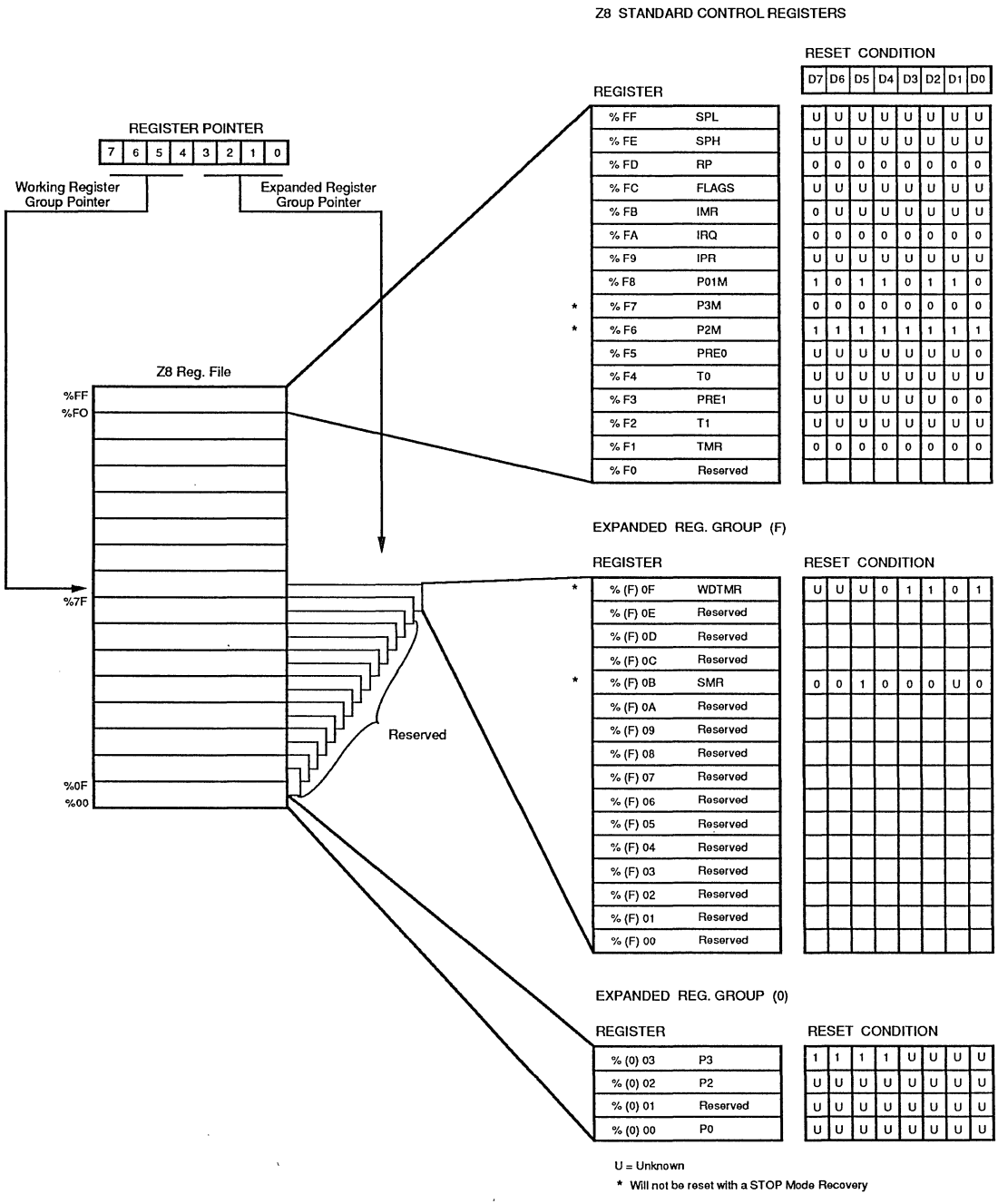


Figure 11. Expanded Register File Architecture

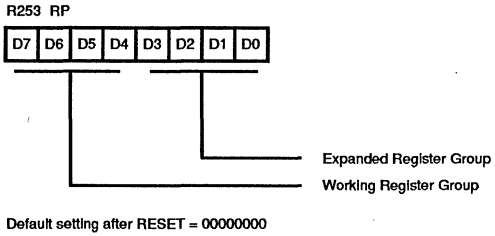


Figure 12. Register Pointer Register

Register File. The register file consists of four I/O port registers, 236 general purpose registers and 15 control and status registers (R0-R3, R4-239 and R240-R255, respectively), plus two system configuration registers in the expanded register group. The instructions can access registers directly or indirectly via an 8-bit address field. This allows a short, 4-bit register address using the Register Pointer (Figure 13). In the 4-bit mode, the register file is divided into 16 working register groups, each occupying 16 continuous locations. The Register Pointer addresses the starting location of the active working register group.

Note: Register Bank E0-EF is only accessed through working registers and indirect addressing modes.

Stack. The Z86C90/C89 external data memory or the internal register file is used for the stack. The 16-bit Stack Pointer (R254-R255) is used for the external stack residing anywhere in the data memory for ROMless mode. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 236 general purpose registers (R4-R239). SPH is used as a general purpose register only when using internal stacks.

Counter/Timers. There are two 8-bit programmable counter/timers (T0-T1), each driven by its own 6-bit programmable prescaler. The T1 prescaler is driven by internal or external clock sources; however, the T0 prescaler is driven by the internal clock only (Figure 14).

The 6-bit prescaler divides the input frequency of the clock source by any integer number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of the count, a timer interrupt request, IRQ4 (T0) or IRQ5 (T1), is generated.

The counters can be programmed to start, stop, restart to continue, or restart from the initial value. The counters can also be programmed to stop upon reaching zero (single pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode).

The counters, but not the prescalers, are read at any time without disturbing their value or count mode. The clock source for T1 is user-definable and can be either the internal microprocessor clock divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input (P31) as an external clock, a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. The counter/timers can be cascaded by connecting the T0 output to the input of T1.

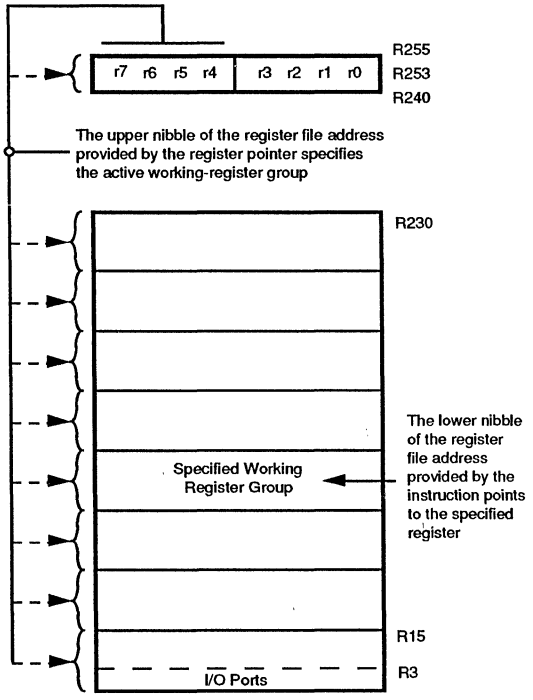


Figure 13. Register Pointer

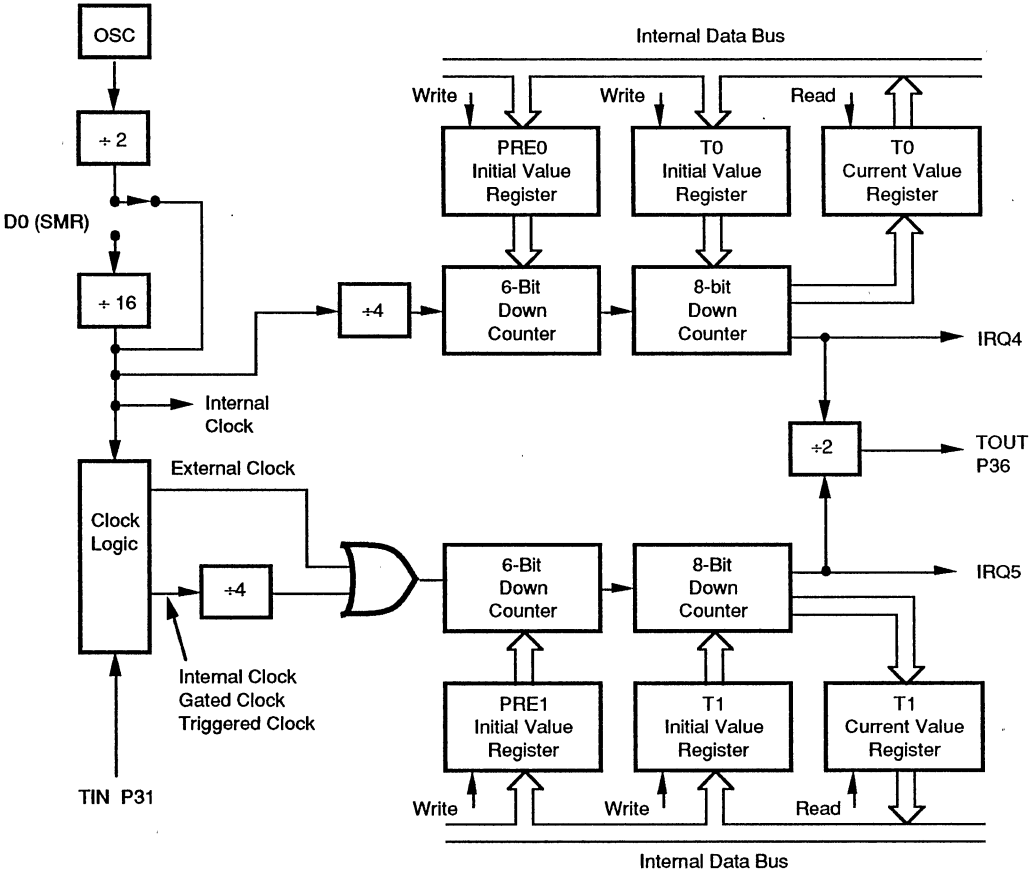


Figure 14. Counter/Timer Block Diagram

Interrupts. The Z86C90/C89 has six different interrupts from six different sources. The interrupts are maskable and prioritized (Figure 15). The six sources are divided as follows; four sources are claimed by Port 3 lines P30-P33, and two in counter/timers (Table 5). The Interrupt Mask Register globally or individually enables or disables the six interrupt requests.

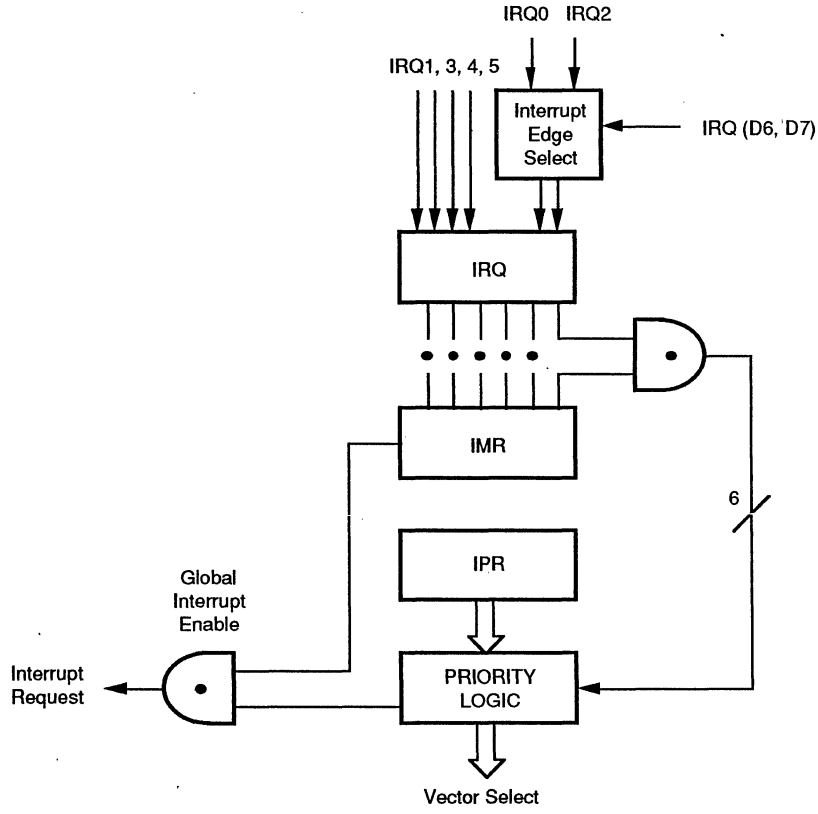


Figure 15. Interrupt Block Diagram

Table 5. Interrupt Types, Sources, and Vectors

Name	Source	Vector Location	Comments
IRQ 0	/DAV 0, IRQ 0	0, 1	External (P32), Rising Falling Edge Triggered
IRQ 1,	IRQ 1	2, 3	External (P33), Falling Edge Triggered
IRQ 2	/DAV 2, IRQ 2, T_{IN}	4, 5	External (P31), Rising Falling Edge Triggered
IRQ 3	IRQ3	6, 7	External (P30), Falling Edge Triggered
IRQ 4	T0	8, 9	Internal
IRQ 5	TI	10, 11	Internal

When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register. An interrupt machine cycle is activated when an interrupt request is granted. Thus, this disables all subsequent interrupts, saves the Program Counter and Status Flags, and then branches to the program memory vector location reserved for that interrupt. All Z86C90/C89 interrupts are vectored through locations in the program memory. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request. To accommodate polled interrupt systems, interrupt inputs are masked and the Interrupt Request register is polled to determine which of the interrupt requests need service.

An interrupt resulting from AN1 is mapped into IRQ2, and an interrupt from AN2 is mapped into IRQ0. Interrupts IRQ2 and IRQ0 may be rising, falling or both edge triggered, and are programmable by the user. The software can poll to identify the state of the pin.

Programming bits for the Interrupt Edge Select are located in the IRQ Register (R250), bits D7 and D6. The configuration is shown in Table 6.

Table 6. IRQ Register

IRQ		Interrupt Edge	
D7	D6	P31	P32
0	0	F	F
0	1	F	R
1	0	R	F
1	1	R/F	R/F

Notes:
F=Falling Edge
R=Rising Edge

Clock. The Z86C90 on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal, LC, ceramic resonator, or any suitable external clock source (XTAL1 = Input, XTAL2 = Output). The crystal should be AT cut, 1 MHz to 12 MHz max., with a series resistance (RS) less than or equal to 100 Ohms. The Z86C89 on-chip oscillator may be driven with a low cost RC network or other suitable external clock source. (Note: The RC option is not available in the 12 MHz part).

The crystal should be connected across XTAL1 and XTAL2 using the recommended capacitors (capacitance is more than or equal to 22 pF) from each pin to ground. The RC oscillator configuration is an external resistor connected from XTAL1 to XTAL2, with a frequency-setting capacitor from XTAL1 to ground (Figure 16). See Figures 52-54 for typical characteristics.

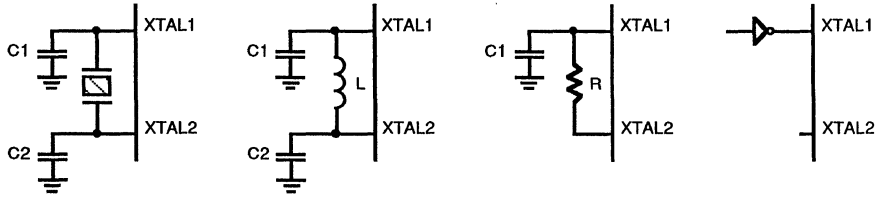
Power-On-Reset (POR). A timer circuit clocked by a dedicated on-board RC oscillator is used for the Power-On Reset (POR) timer function. The POR time allows V_{CC} and the oscillator circuit to stabilize before instruction execution begins.

The POR timer circuit is a one-shot timer triggered by one of three conditions:

1. Power fail to Power OK status.
2. STOP mode recovery (if D5 of SMR=1).
3. WDT timeout.

The POR time is a nominal 5 ms. Bit-5 of the Stop Mode Register determines whether the POR timer is bypassed after STOP mode recovery (typical for external clock, RC/LC oscillators).

HALT. HALT turns off the internal CPU clock, but not the XTAL oscillation. The counter/timers and external interrupts IRQ0, IRQ1, IRQ2, and IRQ3, remain active. The devices are recovered by interrupts, either externally or internally generated.



Ceramic Resonator or Crystal
C1, C2 = 47 pF TYP *
f = 8 MHz

LC
C1, C2 = 22 pF

L = 130 µH *
f = 3 MHz *

RC
@ 5V VCC (TYP)

C1 = 33 pF *
R = 1K *
f = 6 MHz *

External Clock

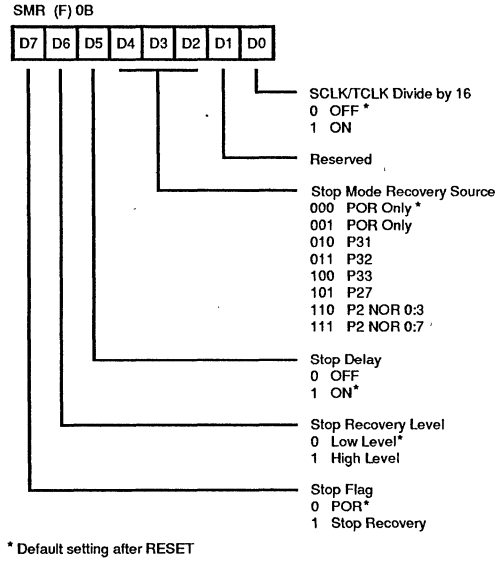
* Preliminary value including pin parasitics

Figure 16. Oscillator Configuration

STOP. This instruction turns off the internal clock and external crystal oscillation and reduces the standby current to 10 microamperes or less. The STOP Mode is terminated by a reset only, either by WDT timeout, POR, SMR recovery or external reset. This causes the processor to restart the application program at address 000C (HEX). In order to enter STOP (or HALT) mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. To do this, the user must execute a NOP (opcode=FFH) immediately before the appropriate sleep instruction, i.e.:

```
FF    NOP     ; clear the pipeline
6F    STOP    ; enter STOP mode
              or
FF    NOP     ; clear the pipeline
7F    HALT    ; enter HALT mode
```

Stop Mode Recovery Register (SMR). This register selects the clock divide value and determines the mode of STOP Mode Recovery (Figure 17). All bits are write only, except Bit 7 which is read only. Bit 7 is a flag bit that is hardware set on the condition of STOP recovery and reset by a power-on cycle. Bit 6 controls whether a low level or a high level is required from the recovery source. Bit 5 controls the reset delay after recovery. Bits 2, 3, and 4, or the SMR register, specify the source of the STOP Mode Recovery signal. Bits 0 and 1 determine the timeout period of the WDT. The SMR is located in Bank F of the Expanded Register Group at address OBH.



* Default setting after RESET

Figure 17. STOP Mode Recovery Register

SCLK/TCLK divide-by-16 Select (D0). D0 of the SMR control a divide-by-16 prescaler of SCLK/TCLK. The purpose of this control is to selectively reduce device power consumption during normal processor execution (SCLK control) and/or HALT mode (where TCLK sources counter/timers and interrupt logic).

STOP Mode Recovery Source (D2, D3, and D4). These three bits of the SMR specify the wake up source of the STOP recovery (Figure 18 and Table 7).

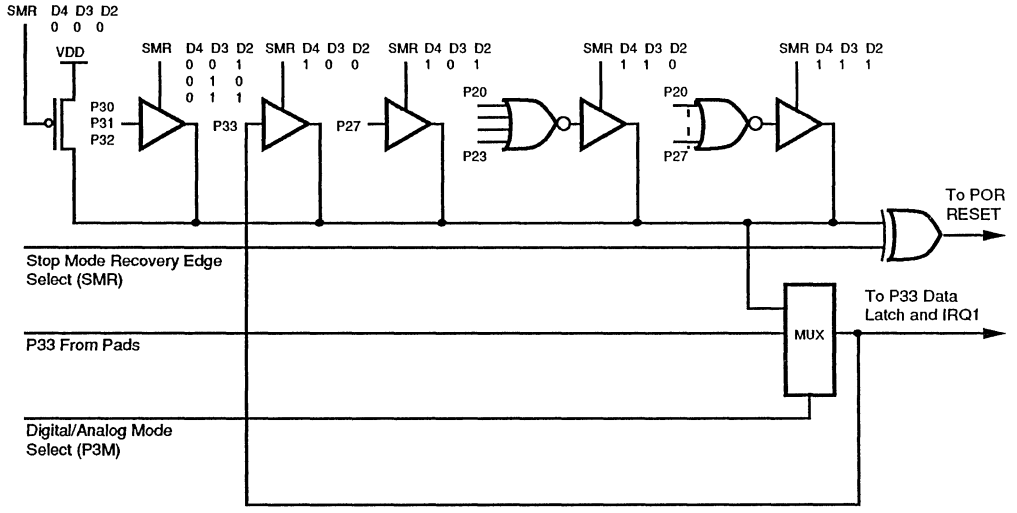


Figure 18. STOP Mode Recovery Source

Table 7. STOP Mode Recovery Source

SMR:432			Operation
D4	D3	D2	Description of Action
0	0	0	POR and/or external reset recovery
0	0	1	P30 transition
0	1	0	P31 transition
0	1	1	P32 transition
1	0	0	P33 transition
1	0	1	P27 transition
1	1	0	Logical NOR of P20 through P23
1	1	1	Logical NOR of P20 through P27

STOP Mode Recovery Delay Select (D5). This bit, if high, disables the 5 ms /RESET delay after STOP Mode Recovery. The default configuration of this bit is one. If the "fast" wake up is selected, the STOP Mode Recovery source needs to be kept active for at least 5 TpC.

STOP Mode Recovery Edge Select (D6). A 1 in this bit position indicates that a high level on any one of the recovery sources wakes the Z86C90/C89 from STOP Mode. A 0 indicates low level recovery. The default is 0 on POR (Figure 18).

Cold or Warm Start (D7). This bit is set by the device upon entering STOP Mode. A 0 in this bit (cold) indicates that the device will be reset by POR/WDT RESET. A 1 in this bit (warm) indicates that the device awakens by a SMR source.

Watch-Dog-Timer Mode Register (WDTMR). The WDT is a retriggerable one-shot timer that resets the Z8 if it reaches its terminal count. The WDT is initially enabled by executing the WDT instruction and refreshed on subsequent executions of the WDT instruction. The WDT circuit is driven by an on-board RC oscillator or external oscillator from the XTAL1 pin. The POR clock source is selected with bit 4 of the WDT register (Figures 19 and 20).

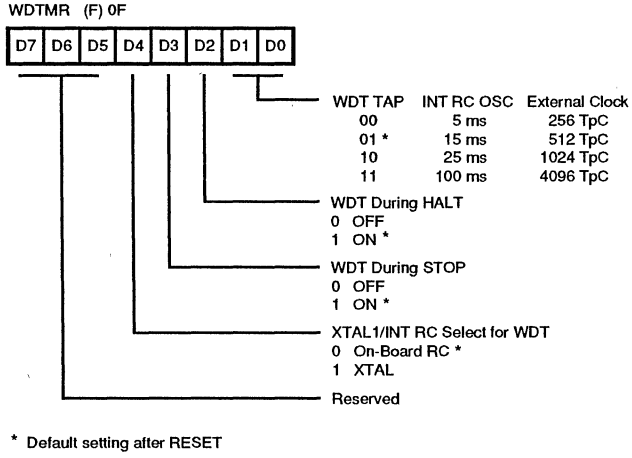


Figure 19. Watch-Dog Timer Mode Register

WDT Time Select (D0,D1). Selects the WDT time period. It is configured as shown in Table 8.

Table 8. WDT Time Select

D1	D0	Timeout of internal RC OSC	Timeout of XTAL clock
0	0	5 ms min	256TpC
0	1	15 ms min	512TpC
1	0	25 ms min	1024TpC
1	1	100 ms min	4096TpC

Notes:
TpC = XTAL clock cycle
The default on reset is 15 ms.
See Figures 55 to 58 for details.

WDTMR During HALT (D2). This bit determines whether or not the WDT is active during HALT Mode. A 1 indicates active during HALT. The default is 1.

WDTMR During STOP (D3). This bit determines whether or not the WDT is active during STOP Mode. Since XTAL clock is stopped during STOP Mode, the on-board RC has to be selected as the clock source to the POR counter. A 1 indicates active during STOP. The default is 1.

Clock source for WDT (D4). This bit determines which oscillator source is used to clock the internal POR and WDT counter chain. If the bit is a 1, the internal RC oscillator is bypassed and the POR and WDT clock source is driven from the external pin, XTAL1. The default configuration of this bit is 0, which selects the RC oscillator.

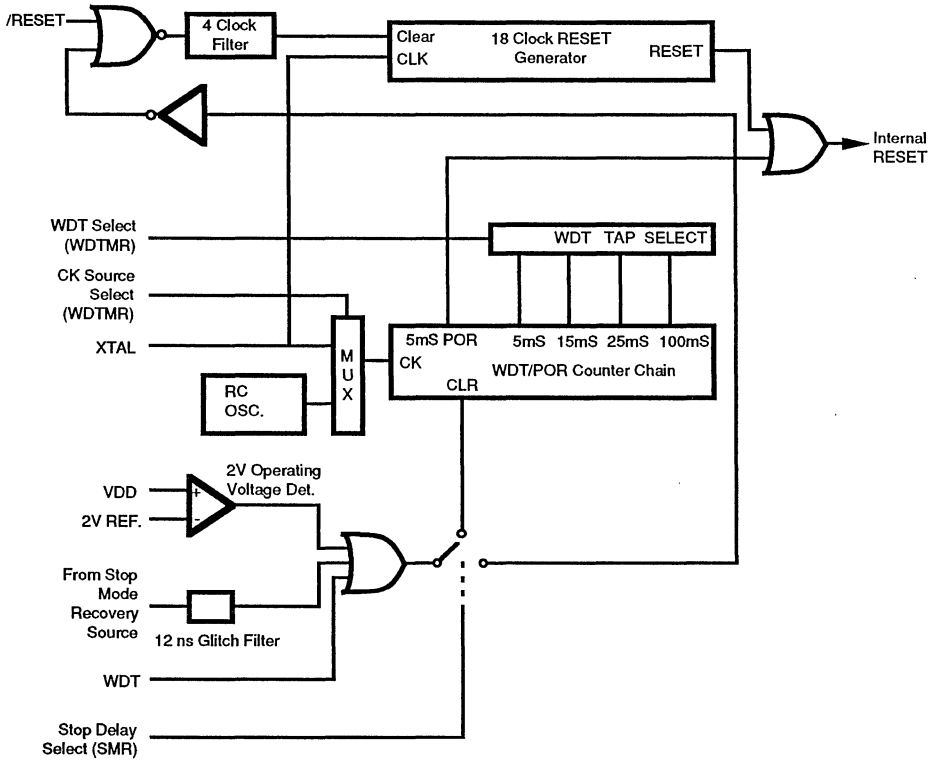


Figure 20. Resets and WDT

Brown-Out Protection. An on-board Voltage Comparator checks that V_{CC} is at the required level to ensure correct operation of the device. Reset is globally driven if V_{CC} is below V_{BO} (Brown-Out Voltage). The minimum operating voltage varies with the temperature and operating frequency, while V_{BO} varies with temperature only.

The brown-out trip voltage (V_{BO}) is less than 3 volts and above 1.4 volts under the following conditions.

Maximum (V_{BO}) Conditions:

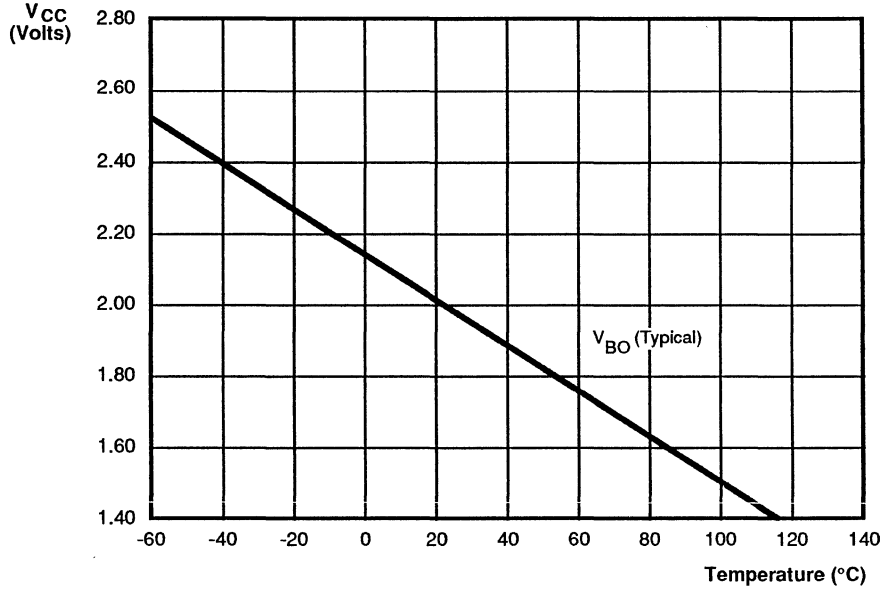
Case 1 T_A = -40°C, +105°C, Internal Clock Frequency equal or less than 1 MHz

Case 2 T_A = -40°C, +85°C, Internal Clock Frequency equal or less than 2 MHz

Note: The internal clock frequency is one-half the external clock frequency.

FUNCTIONAL DESCRIPTION (Continued)

The device functions normally at or above 3.0 V under all conditions. Below 3.0 V, the device functions normally until the Brown-Out Protection trip point is reached, below which reset is globally driven. The device is guaranteed to function normally at supply voltages above the brown out trip point for the temperatures and operating frequencies in cases 1 and 2. The actual brown out trip point is a function of temperature and process parameters (Figure 21).



* Power-on Reset threshold for V_{CC} and 4 MHz V_{BO} overlap

Figure 21. Typical Z86C90/C89 Brown-Out Voltage vs Temperature at 4 MHz

STANDARD TEST CONDITIONS

The characteristics listed below apply for standard test conditions as noted. All voltages are referenced to GND. Positive current flows into the referenced pin (Figure 22).

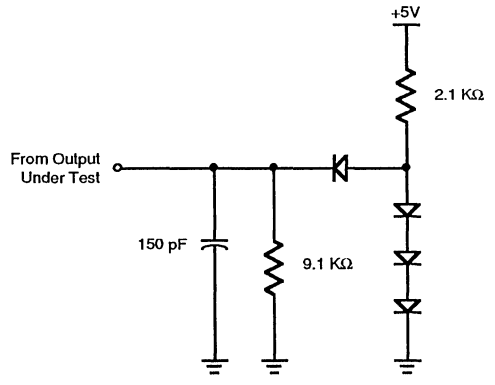


Figure 22. Test Load Diagram

ABSOLUTE MAXIMUM RATINGS

Symbol	Description	Min	Max	Units
V_{CC}	Supply Voltage (*)	-0.3	+7.0	V
T_{STG}	Storage Temp	-65°	+150°	C
T_A	Oper Ambient Temp		†	C
	Power Dissipation		2.2	W

Notes:
* Voltage on all pins with respect to GND.
† See Ordering Information.

Stress greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for an extended period may affect device reliability.

CAPACITANCE

T_A = 25°C, V_{CC} = GND = 0V, f = 1.0 MHz, unmeasured pins to GND

Parameter	Max
Input capacitance	12 pF
Output capacitance	12 pF
I/O capacitance	12 pF

DC CHARACTERISTICS

Sym	Parameter	V_{CC} Note [3]	T_A = 0°C to 70°C Min	Max	T_A = -40° C to 105° C Min	Max	Typ @ 25°C	Units	Conditions	Notes
	Max Input Voltage	3.3V		7		7		V	I_{IN} 250µA	
		5.0V		7		7		V	I_{IN} 250µA	
V_{CH}	Clock Input High Voltage	3.3V	0.7 V_{CC}	V_{CC}+0.3	0.7 V_{CC}	V_{CC}+0.3	1.3	V	Driven by External Clock Generator	
		5.0V	0.7 V_{CC}	V_{CC}+0.3	0.7 V_{CC}	V_{CC}+0.3	2.5	V	Driven by External Clock Generator	
V_{CL}	Clock Input Low Voltage	3.3V	GND -0.3	0.2 V_{CC}	GND-0.3	0.2 V_{CC}	0.7	V	Driven by External Clock Generator	
		5.0V	GND-0.3	0.2 V_{CC}	GND-0.3	0.2 V_{CC}	1.5	V	Driven by External Clock Generator	
V_{IH}	Input High Voltage	3.3V	0.7 V_{CC}	V_{CC}+0.3	0.7 V_{CC}	V_{CC}+0.3	1.3	V		
		5.0V	0.7 V_{CC}	V_{CC}+0.3	0.7 V_{CC}	V_{CC}+0.3	2.5	V		
V_{IL}	Input Low Voltage	3.3V	GND-0.3	0.2 V_{CC}	GND-0.3	0.2 V_{CC}	0.7	V		
		5.0V	GND-0.3	0.2 V_{CC}	GND-0.3	0.2 V_{CC}	1.5	V		
V_{OH}	Output High Voltge	3.3V	V_{CC}-0.4		V_{CC}-0.4		3.1	V	I_{OH} = -2.0 mA	
		5.0V	V_{CC}-0.4		V_{CC}-0.4		4.8	V	I_{OH} = -2.0 mA	
V_{OL1}	Output Low Voltage	3.3V		0.6		0.6	0.2	V	I_{OH} = +4.0 mA	
		5.0V		0.4		0.4	0.1	V	I_{OL} = +4.0 mA	
V_{OL2}	Output Low Voltage	3.3V		1.2		1.2	0.3	V	I_{OL} = +6 mA, 3 Pin Max	
		5.0V		1.2		1.2	0.3	V	I_{OL} = +12 mA, 3 Pin Max	
V_{RH}	Reset Input	3.3V	.8 V_{CC}	V_{CC}	.8 V_{CC}	V_{CC}	1.5	V		
	High Voltage	5.0V	.8 V_{CC}	V_{CC}	.8 V_{CC}	V_{CC}	2.1	V		
V_{RI}	Reset Input	3.3V	GND-0.3	0.2 V_{CC}	GND-0.3	0.2 V_{CC}	1.1			
	Low Voltage	5.0V	GND-0.3	0.2 V_{CC}	GND-0.3	0.2 V_{CC}	1.7			
V_{OFFSET}	Comparator Input	3.3V		25		25	10	mV		
	Offset Voltage	5.0V		25		25	10	mV		
I_{IL}	Input Leakage	3.3V	-1	1	-1	2	< 1	µA	V_{IN} = 0V, V_{CC}	
		5.0V	-1	1	-1	2	< 1	µA	V_{IN} = 0V, V_{CC}	
I_{OL}	Output Leakage	3.3V	-1	1	-1	2	< 1	µA	V_{IN} = 0V, V_{CC}	
		5.0V	-1	1	-1	2	< 1	µA	V_{IN} = 0V, V_{CC}	
I_{IR}	Reset Input Current	3.3V		-45		-60	-20	µA		
		5.0V		-55		-70	-30	µA		
I_{CC}	Supply Current	3.3V		10		10	4	mA	@ 8 MHz	[4,5]
		5.0V		15		15	10	mA	@ 8 MHz	[4,5]
		3.3V		15		15	5	mA	@ 12 MHz	[4,5]
		5.0V		20		20	15	mA	@ 12 MHz	[4,5]

Sym	Parameter	V_{CC} Note [3]	T_A = 0°C to 70°C Min	Max	T_A = -40° C to 105° C Min	Max	Typ @ 25°C	Units	Conditions	Notes
I_{CC1}	Standby Current	3.3V		3		3	1	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 8 MHz	[4,5]
		5.0V		5		5	2.4	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 8 MHz	[4,5]
		3.3V		4		4	1.5	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 12 MHz	[4,5]
		5.0V		6		6	3.2	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 12 MHz	[4,5]
		3.3V		2		2	0.8	mA	Clock Divide by 16 @ 8 MHz	[4,5]
		5.0V		4		4	1.8	mA	Clock Divide by 16 @ 8 MHz	[4,5]
		3.3V		3		3	1.2	mA	Clock Divide by 16 @ 12 MHz	[4,5]
		5.0V		5		5	2.5	mA	Clock Divide by 16 @ 12 MHz	[4,5]
I_{CC2}	Standby Current	3.3V		8		15	1	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is not Running	[6]
		5.0V		10		20	2	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is not Running	[6]
		3.3V		500		600	310	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is Running	[6]
		5.0V		800		1000	600	µA	STOP Mode V_{IN} = 0V, V_{CC} WDT is Running	
I_{ALL}	Auto Latch Low Current	3.3V		8		10	5	µA	0V < V_{IN} < V_{CC}	
		5.0V		15		20	11	µA	0V < V_{IN} < V_{CC}	
I_{ALH}	Auto Latch High Current	3.3V		-5		-7	-3	µA	0V < V_{IN} < V_{CC}	
		5.0V		-8		-10	-6	µA	0V < V_{IN} < V_{CC}	
I_{POR}	Power On Reset	3.3V	7	24	7	25	13	ms		
		5.0V	3	13	3	14	7	ms		
V_{BO}	V_{CC} Brown-Out Voltage		1.5	2.65	1.2	2.95	2.1	V	2 MHz max Ext. CLK Freq.	[7]

Notes:

[1] I_{CC1}

	Typ	Max	Unit	Frequency
Crystal/Resonator	3.0 mA	5	mA	8 MHz
External Clock Drive	0.3 mA	5	mA	8 MHz

[2] GND=0V

[3] 5.0V ±0.5V,3.3V ±0.3V.

[4] All outputs unloaded, I/O pins floating, inputs at rail.

[5] CL1=CL2=100 pF

[6] Same as note [4] except inputs at V_{CC}.

[7] The V_{BO} increases as the temperature decreases.

AC CHARACTERISTICS

External I/O or Memory Read and Write Timing Diagram

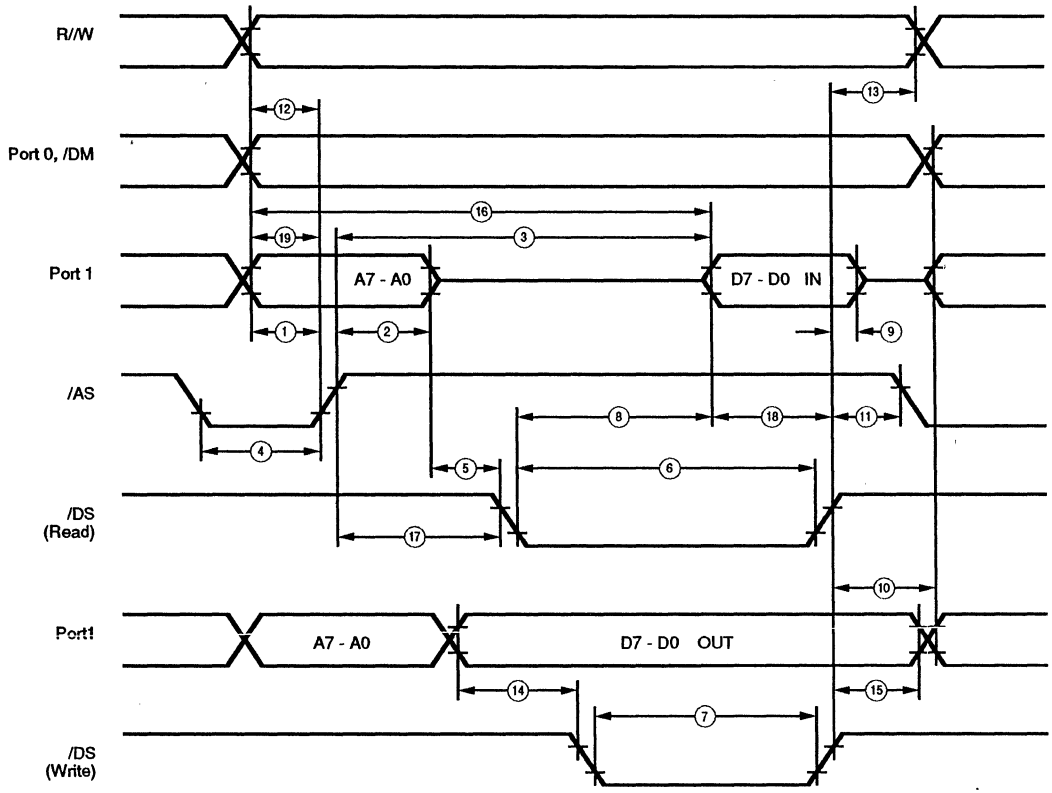


Figure 23. External I/O or Memory Read/Write Timing

AC CHARACTERISTICS

External I/O or Memory Read and Write Timing Table

No	Symbol	Parameter	V_{CC} Note[3]	T_A = 0°C to 70°C				T_A = -40°C to 105°C				Units	Notes
				8 MHz		12 MHz		8 MHz		12 MHz			
				Min	Max	Min	Max	Min	Max	Min	Max		
1	TdA(AS)	Address Valid to	3.3	55		35		55		35		ns	[2]
		/AS Rising Delay	5.0	55		35		55		35		ns	
2	TdAS(A)	/AS Rising to Address	3.3	70		45		70		45		ns	[2]
		Float Delay	5.0	70		45		70		45		ns	
3	TdAS(DR)	/AS Rising to Read	3.3		400		250		400		250	ns	[1,2]
		Data Required Valid	5.0		400		250		400		250	ns	
4	TwAS	/AS Low Width	3.3	80		55		80		55		ns	[2]
			5.0	80		55		80		55		ns	
5	Td	Address Float to	3.3	0		0		0		0		ns	
		/DS Falling	5.0	0		0		0		0		ns	
6	TwDSR	/DS (Read) Low Width	3.3	300		200		300		200		ns	[1,2]
			5.0	300		200		300		200		ns	
7	TwDSW	/DS (Write) Low Width	3.3	165		110		165		110		ns	[1,2]
			5.0	165		110		165		110		ns	
8	TdDSR(DR)	/DS Falling to Read	3.3		260		150		260		150	ns	[1,2]
		Data Required Valid	5.0		260		160		260		160	ns	
9	ThDR(DS)	Read Data to	3.3	0		0		0		0		ns	[2]
		/DS Rising Hold Time	5.0	0		0		0		0		ns	
10	TdDS(A)	/DS Rising to Address	3.3	85		45		85		45		ns	[2]
		Active Delay	5.0	95		55		95		55		ns	
11	TdDS(AS)	/DS Rising to /AS	3.3	60		30		60		30		ns	[2]
		Falling Delay	5.0	70		45		70		45		ns	
12	TdR/W(AS)	R/W Valid to /AS	3.3	70		45		70		45		ns	[2]
		Rising Delay	5.0	70		45		70		45		ns	
13	TdDS(R/W)	/DS Rising to	3.3	70		45		70		45		ns	[2]
		R/W Not Valid	5.0	70		45		70		45		ns	
14	TdDW(DSW)	Write Data Valid to /DS	3.3	80		55		80		55		ns	[2]
		Falling (Write) Delay	5.0	80		55		80		55		ns	
15	TdDS(DW)	/DS Rising to Write	3.3	70		45		70		45		ns	[2]
		Data Not Valid Delay	5.0	80		55		80		55		ns	
16	TdA(DR)	Address Valid to Read	3.3		475		310		475		310	ns	[1,2]
		Data Required Valid	5.0		475		310		475		310	ns	
17	TdAS(DS)	/AS Rising to	3.3	100		65		100		65		ns	[2]
		/DS Falling Delay	5.0	100		65		100		65		ns	
18	TdDI(DS)	Data Input Setup to	0.0	115		115		115		115		ns	[1,2]
		/DS Rising	5.0	75		75		75		75		ns	
19	TdDM(AS)	/DM Valid to /AS	3.3	55		35		55		35		ns	[2]
		Falling Delay	5.0	55		35		55		35		ns	

Notes:

[1] When using extended memory timing add 2 TpC.

[2] Timing numbers given are for minimum TpC.

[3] 5.0V ±0.5V, 3.3V ±0.3V.

† Standard Test Load

†† All timing references use 0.9 V_{CC} for a logic 1 and 0.1 V_{CC} for a logic 0.

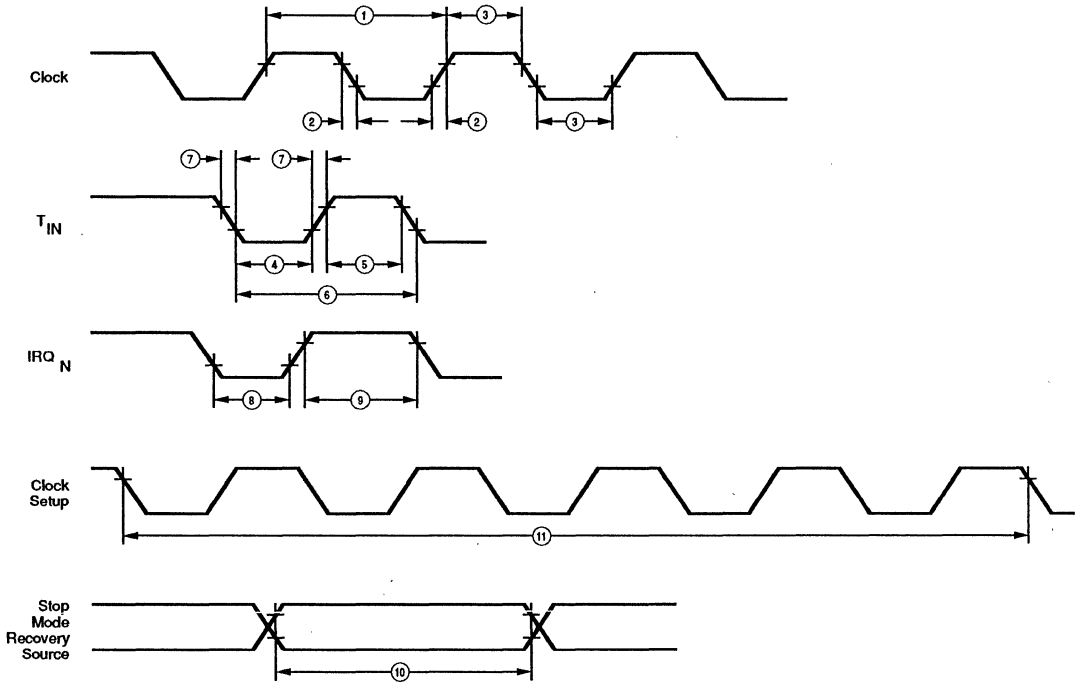


Figure 24. Additional Timing

AC CHARACTERISTICS
Additional Timing Table

No	Symbol	Parameter	V_{CC} Note[6]	T_A = 0°C to 70°C 8 MHz Min	8 MHz Max	12 MHz Min	12 MHz Max	T_A = -40°C to 105°C 8 MHz Min	8 MHz Max	12 MHz Min	12 MHz Max	Units	Notes
1	TpC	Input Clock Period	3.3V	125	100000	83	100000	125	100000	83	100000	ns	[1]
			5.0V	125	100000	83	100000	125	100000	83	100000	ns	[1]
2	TrC,TfC	Clock Input Rise	3.3V		25		15		25		15	ns	[1]
		and Fall Times	5.0V		25		15		25		15	ns	[1]
3	TwC	Input Clock Width	3.3V	37		26		37		26		ns	[1]
			5.0V	37		26		37		26		ns	[1]
4	TwTinL	Timer Input	3.3V	100		100		100		100		ns	[1]
		Low Width	5.0V	70		70		70		70		ns	[1]
5	TwTinH	Timer Input	3.3V	3TpC		3TpC		3TpC		3TpC			[1]
		High Width	5.0V	3TpC		3TpC		3TpC		3TpC			[1]
6	TpTin	Timer Input Period	3.3V	8TpC		8TpC		8TpC		8TpC			[1]
			5.0V	8TpC		8TpC		8TpC		8TpC			[1]
7	TrTin,TfTin	Timer Input Rise	3.3V		100		100		100		100	ns	[1]
		and Fall Timers	5.0V		100		100		100		100	ns	[1]
8A	TwIL	Interrupt Request	3.3V	100		100		100		100		ns	[1,2]
		Low Time	5.0V	70		70		70		70		ns	[1,2]
8B	TwIL	Int. Request	3.3V	3TpC		3TpC		3TpC		3TpC			[1,3]
		Low Time	5.0V	3TpC		3TpC		3TpC		3TpC			[1,3]
9	TwIH	Interrupt Request	3.3V	3TpC		3TpC		3TpC		3TpC			[1,2]
		Input High Time	5.0V	3TpC		3TpC		3TpC		3TpC			[1,2]
10	Twsm	STOP Mode	3.3V	12		12		12		12		ns	
		Recovery Width Spec	5.0V	12		12		12		12		ns	
			3.3V	5TpC									[7]
			5.0V	5TpC									[8]

AC CHARACTERISTICS
Additional Timing Table (Continued)

No	Symbol	Parameter	V_{CC} Note[6]	T_A = 0°C to 70°C				T_A = -40°C to 105°C				Units	Notes
				8 MHz		12 MHz		8 MHz		12 MHz			
				Min	Max	Min	Max	Min	Max	Min	Max		
11	Tost	Oscillator	3.3V		5TpC		5TpC		5TpC		5TpC		[4]
		Startup Time	5.0V		5TpC		5TpC		5TpC		5TpC		[4]
12	Twdt	Watchdog Timer	3.3V	10		10		10		10		ms	D0 = 0 [5]
		Delay Time	5.0V	5		5		5		5		ms	D1 = 0 [5]
			3.3V	30		30		30		30		ms	D0 = 1 [5]
			5.0V	15		15		15		15		ms	D1 = 0 [5]
			3.3V	50		50		50		50		ms	D0 = 0 [5]
			5.0V	25		25		25		25		ms	D1 = 1 [5]
			3.3V	200		200		200		200		ms	D0 = 1 [5]
			5.0V	100		100		100		100		ms	D1 = 1 [5]

Notes:
[1] Timing Reference uses 0.9 V_{CC} for a logic 1 and 0.1 V_{CC} for a logic 0.
[2] Interrupt request via Port 3 (P31-P33).
[3] Interrupt request via Port 3 (P30).
[4] SMR-D5 = 0
[5] Reg. WDTMR
[6] 5.0V ±0.5V, 3.3V ±0.3V
[7] Reg. SMR - D5=0
[8] Reg. SMR - D5=1

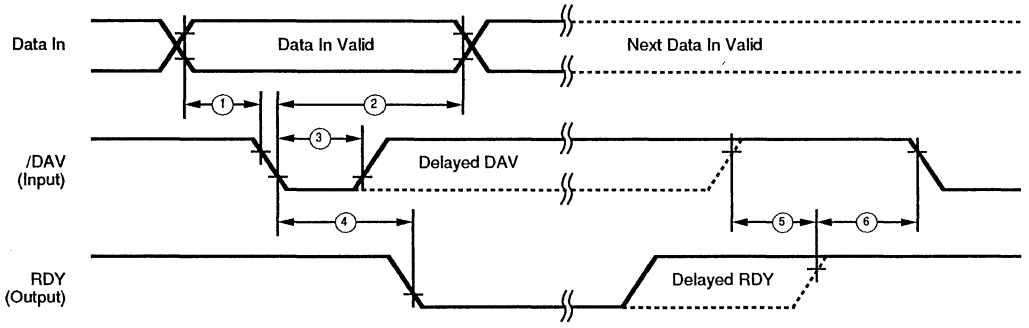


Figure 25. Input Handshake Timing

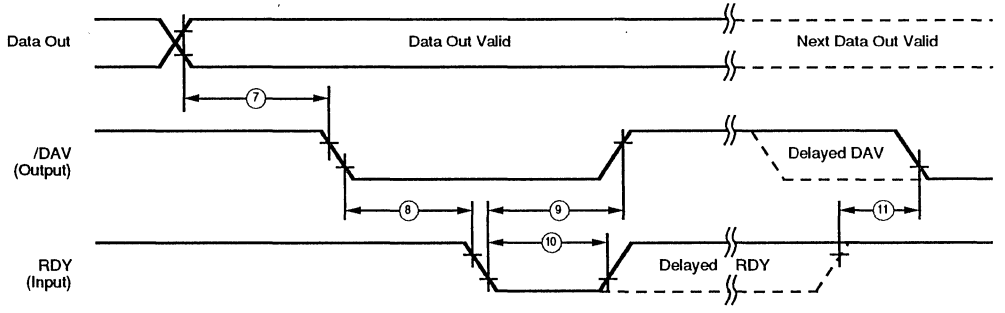
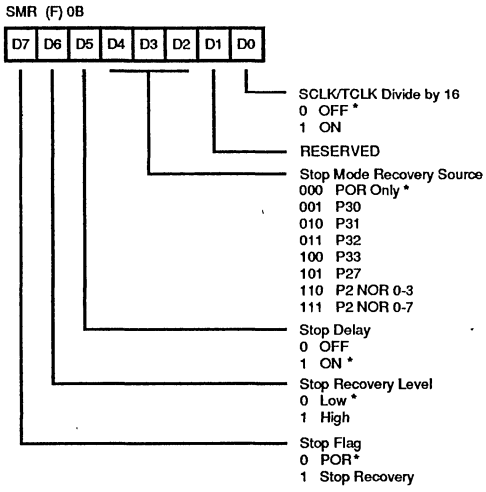


Figure 26. Output Handshake Timing

AC CHARACTERISTICS
Handshake Timing Table

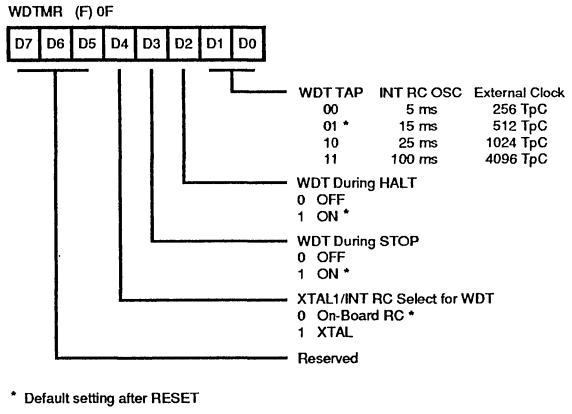
No	Symbol	Parameter	V_{CC} Note[1]	T_A = 0°C To 70° C				T_A = -40° C To 105° C				Data Direction
				8 MHz		12 MHz		8 MHz		12 MHz		
				Min	Max	Min	Max	Min	Max	Min	Max	
1	TsDI(DAV)	Data In Setup Time	3.3V	0		0		0		0		IN
			5.0V	0		0		0		0		IN
2	ThDI(DAV)	Data In Hold Time	3.3V	160		160		160		160		IN
			5.0V	115		115		115		115		IN
3	TwDAV	Data Available Width	3.3V	155		155		155		155		IN
			5.0V	110		110		110		110		IN
4	TdDAVl(RDY)	DAV Falling to RDY	3.3V		160		160		160		160	IN
		Falling Delay	5.0V		115		115		115		115	IN
5	TdDAVld(RDY)	DAV Rising to RDY	3.3V		120		120		120		120	IN
		Falling Delay	5.0V		80		80		80		80	IN
6	TdDO(DAV)	RDY Rising to DAV	3.3V	0		0		0		0		IN
		Falling Delay	5.0V	0		0		0		0		IN
7	TcLDAV0(RDY)	Data Out to DAV	3.3V	63		42		63		42		OUT
		Falling Delay	5.0V	63		42		63		42		OUT
8	TcLDAV0(RDY)	DAV Falling to RDY	3.3V	0		0		0		0		OUT
		Falling Delay	5.0V	0		0		0		0		OUT
9	TdRDY0(DAV)	RDY Falling to DAV	3.3V		160		160		160		160	OUT
		Rising Delay	5.0V		115		115		115		115	OUT
10	TwRDY	RDY Width	3.3V	110		110		110		110		OUT
			5.0V	80		80		80		80		OUT
11	TdRDY0d(DAV)	RDY Rising to DAV	3.3V		110		110		110		110	OUT
		Falling Delay	5.0V		80		80		80		80	OUT

Note:
[1] 5.0 V ±0.5V, 3.3V ±0.3V



* Default setting after RESET

Figure 27. Stop Mode Recovery Register



* Default setting after RESET

Figure 28. Watchdog Timer Mode Register

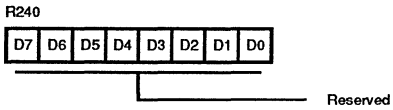


Figure 29. Reserved

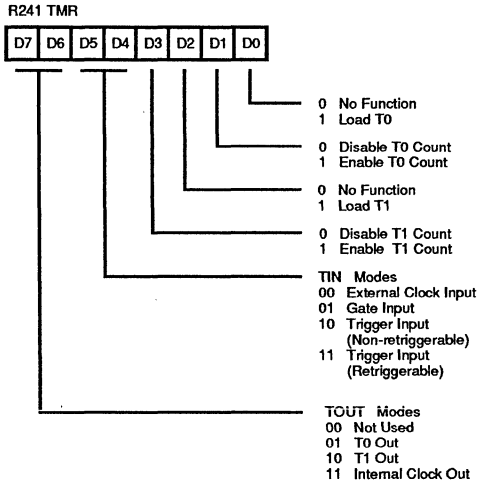


Figure 30. Timer Mode Register ($F1_H$:Read/Write)

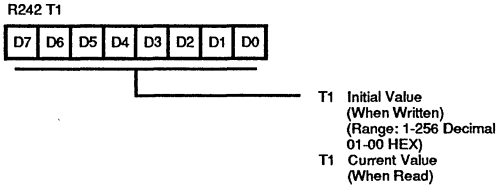


Figure 31. Counter/Timer 1 Register ($F2_H$:Read/Write)

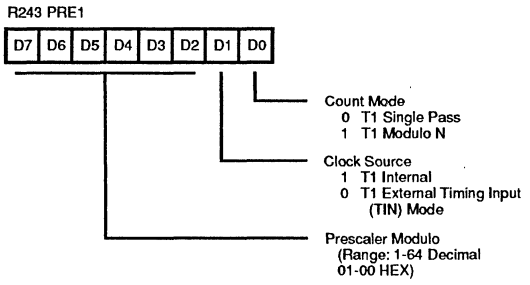


Figure 32. Prescaler 1 Register ($F3_H$:Write Only)

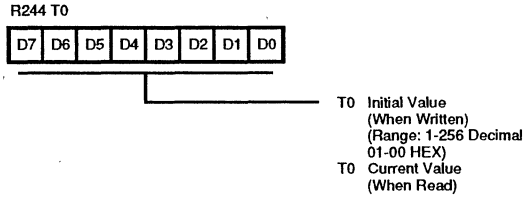


Figure 33. Counter/Timer 0 Register ($F4_H$:Read/Write)

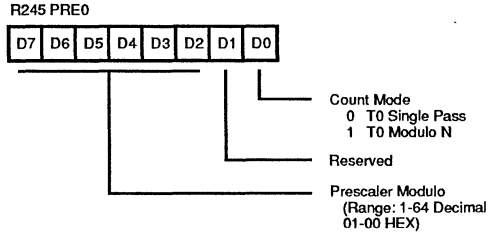


Figure 34. Prescaler 0 Register ($F5_H$:Write Only)

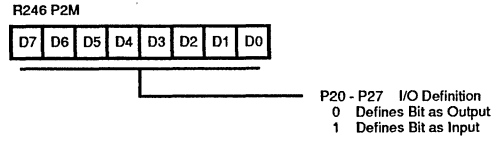


Figure 35. Port 2 Mode Register ($F6_H$: Write Only)

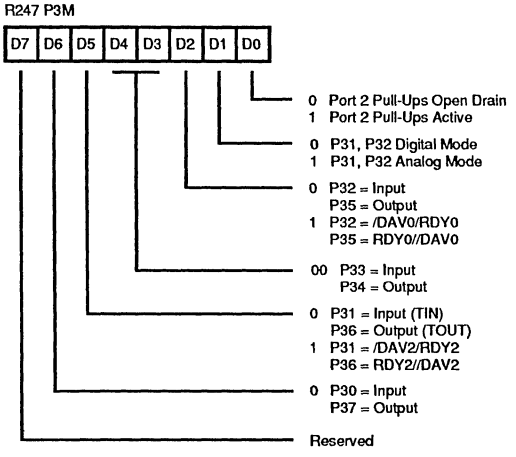


Figure 36. Port 3 Mode Register ($F7_H$:Write Only)

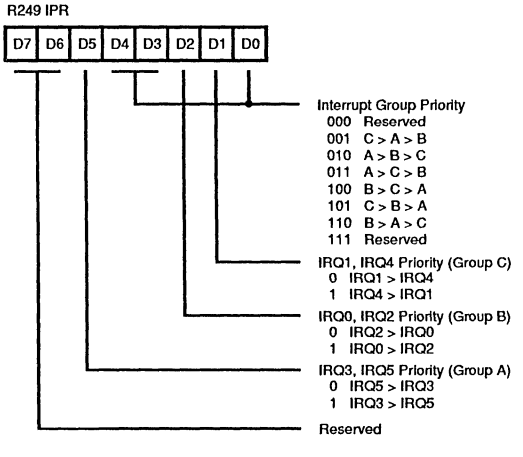


Figure 38. Interrupt Priority Register ($F9_H$:Write Only)

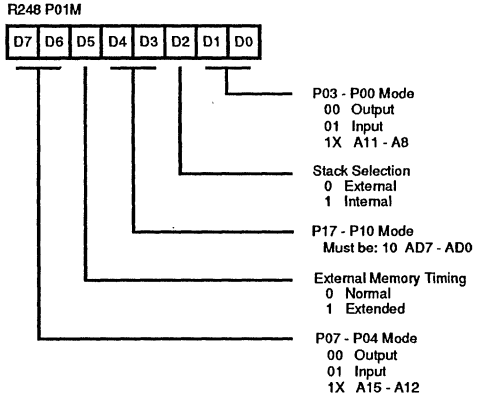


Figure 37. Ports 0 and 1 Mode Registers ($F8_H$:Write Only)

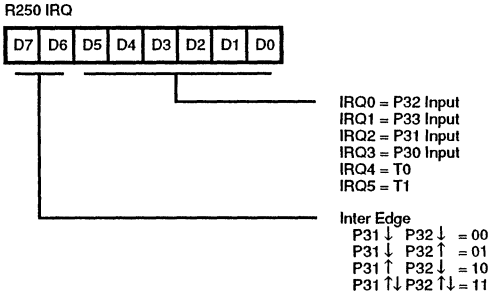


Figure 39. Interrupt Request Register (FA_H:Read/Write)

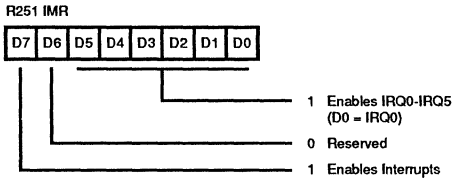


Figure 40. Interrupt Mask Register (FB_H:Read/Write)

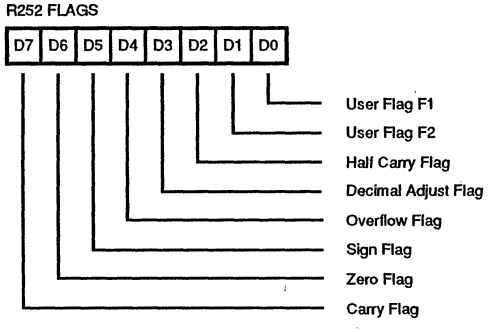


Figure 41. Flag Register (FC_H:Read/Write)

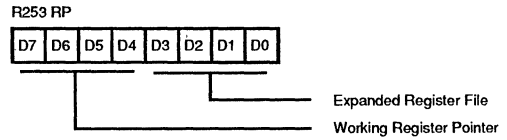


Figure 42. Register Pointer (FD_H:Read/Write)

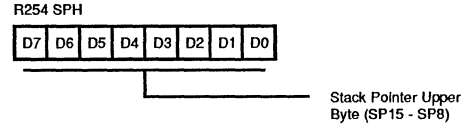


Figure 43. Stack Pointer High (FE_H:Read/Write)

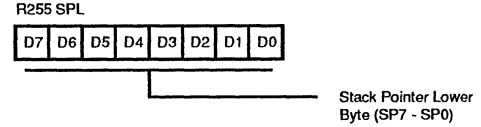


Figure 44. Stack Pointer Low (FF_H:Read/Write)

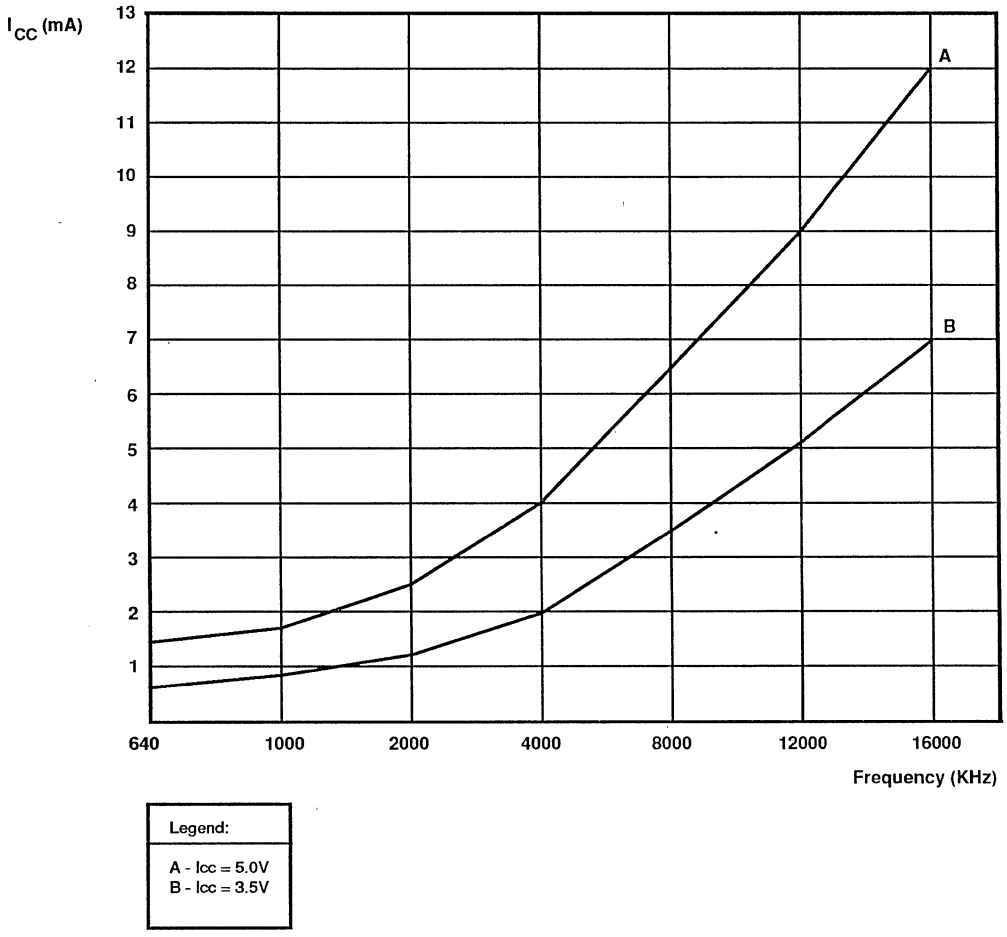


Figure 45. Typical I_{cc} vs Frequency

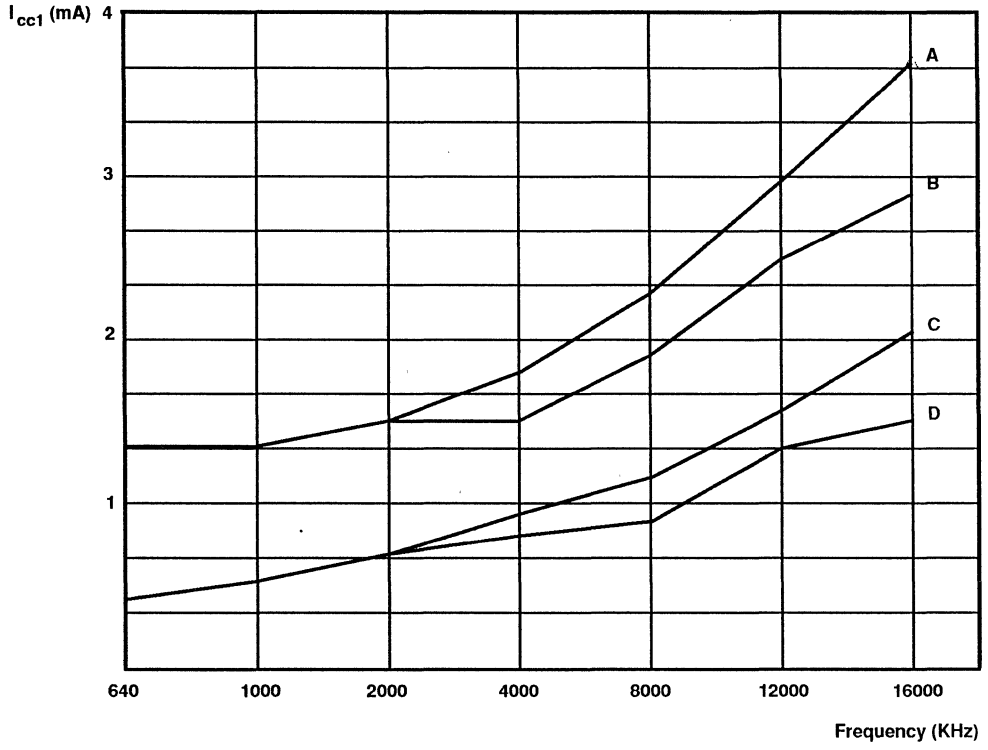
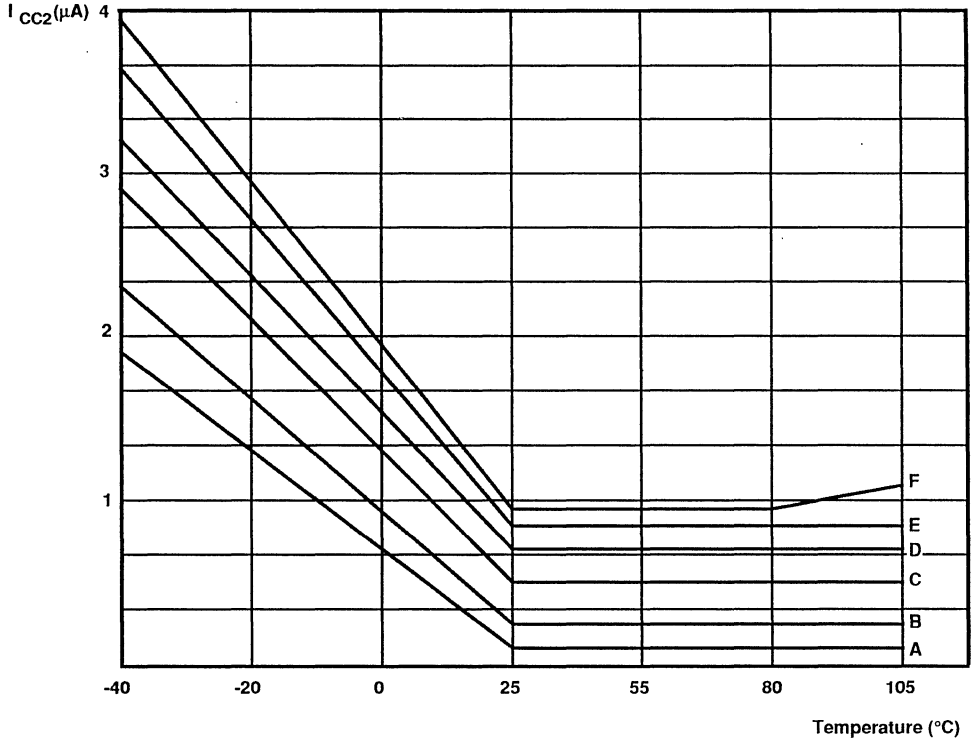
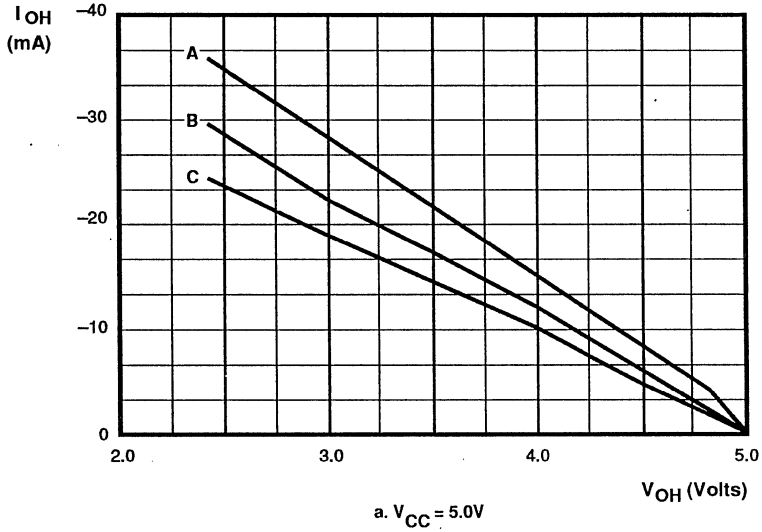


Figure 46. Typical I_{cc1} vs Frequency

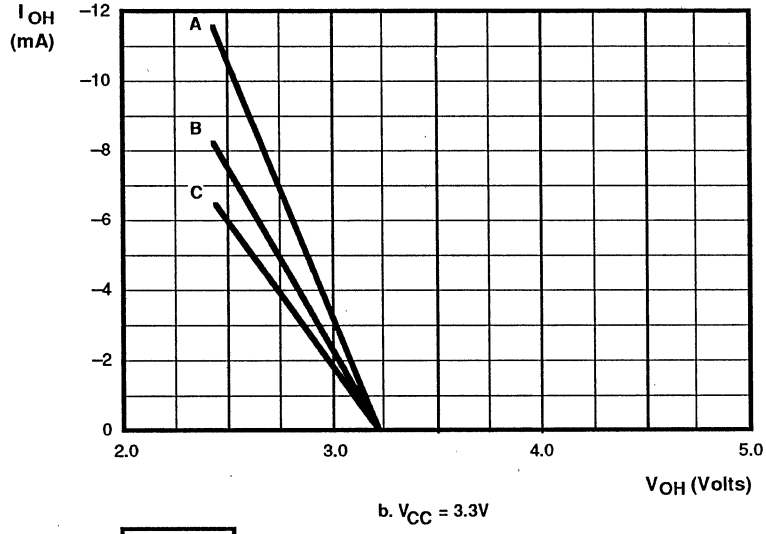


Legend:	
A - Vcc = 3.0V	D - Vcc = 4.5V
B - Vcc = 3.5V	E - Vcc = 5.0V
C - Vcc = 4.0V	F - Vcc = 5.5V

Figure 47. Typical I_{CC2} vs Frequency



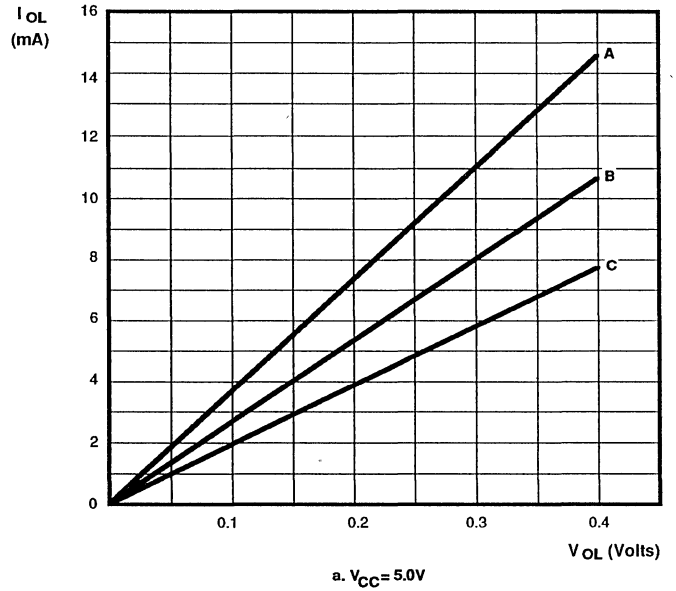
a. V_{CC} = 5.0V



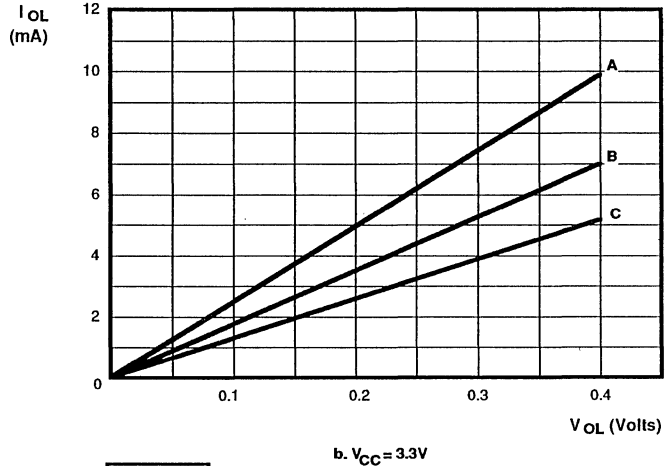
b. V_{CC} = 3.3V

Legend:
A = −55°C
B = 25°C
C = 125°C

Figure 48. Typical I_{OH} vs V_{OH} Over Temperature



a. V_{CC} = 5.0V



b. V_{CC} = 3.3V

Legend:
A = –55°C B = 25°C C = 125°C

Figure 49. Typical I_{OL} vs V_{OL} Over Temperature

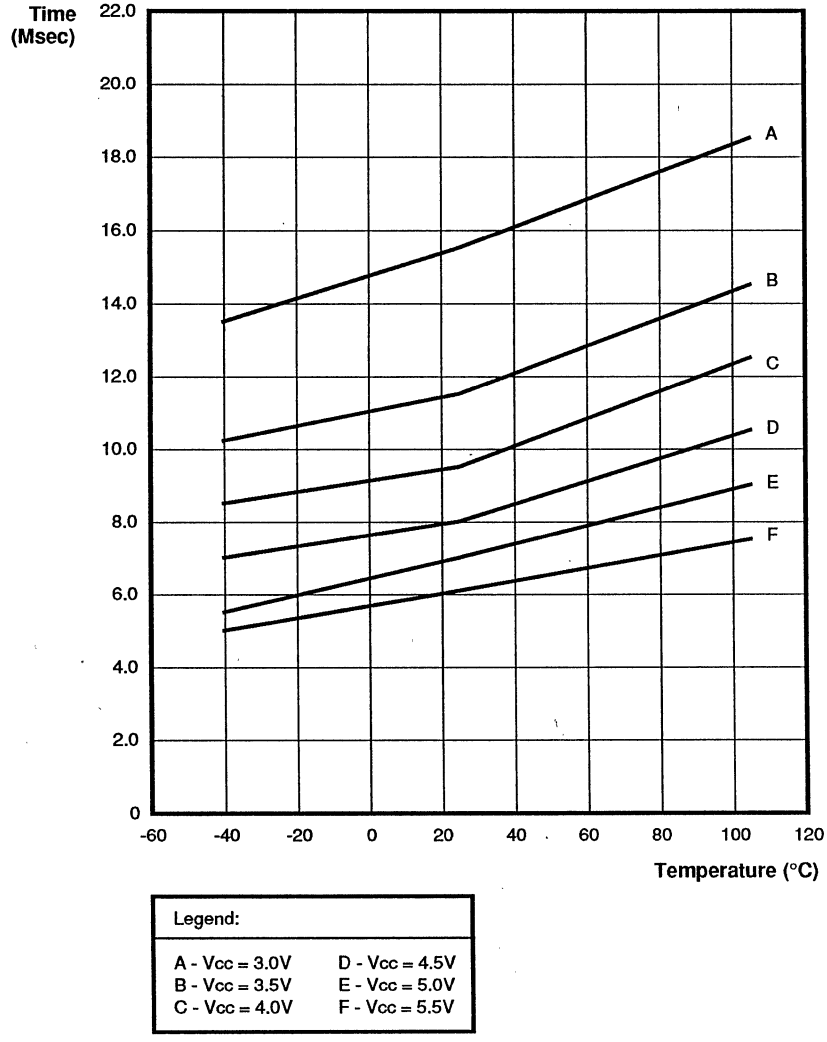
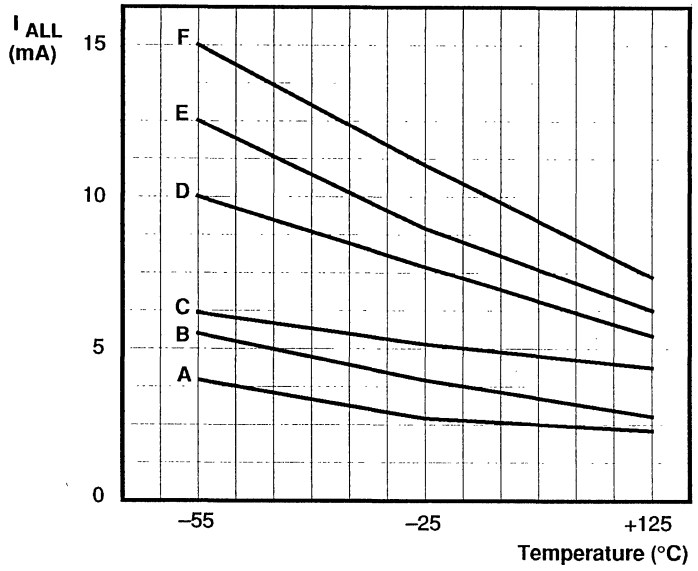
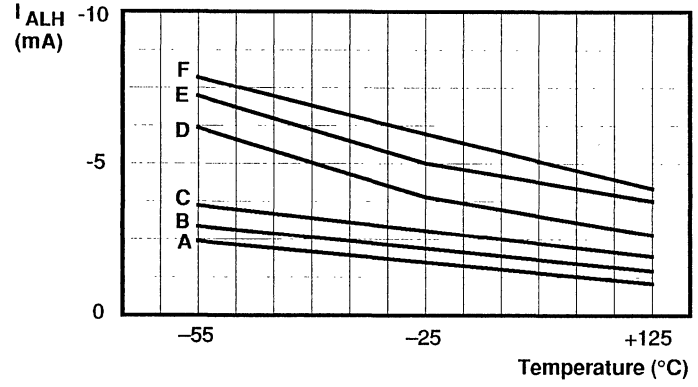


Figure 50. Typical Power-On Reset Time vs Temperature



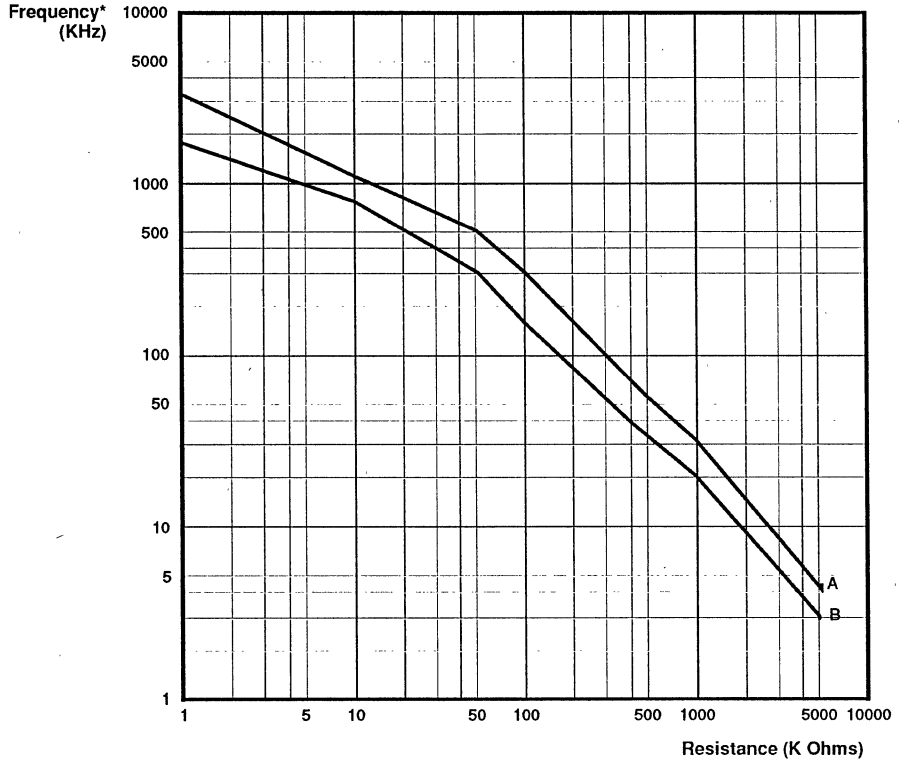
a. Typical Auto Latch Low Current vs Temperature



b. Typical Auto Latch High Current vs Temperature

Legend:	
A - Vcc = 3.0V	D - Vcc = 4.5V
B - Vcc = 3.3V	E - Vcc = 5.0V
C - Vcc = 3.6V	F - Vcc = 5.5V

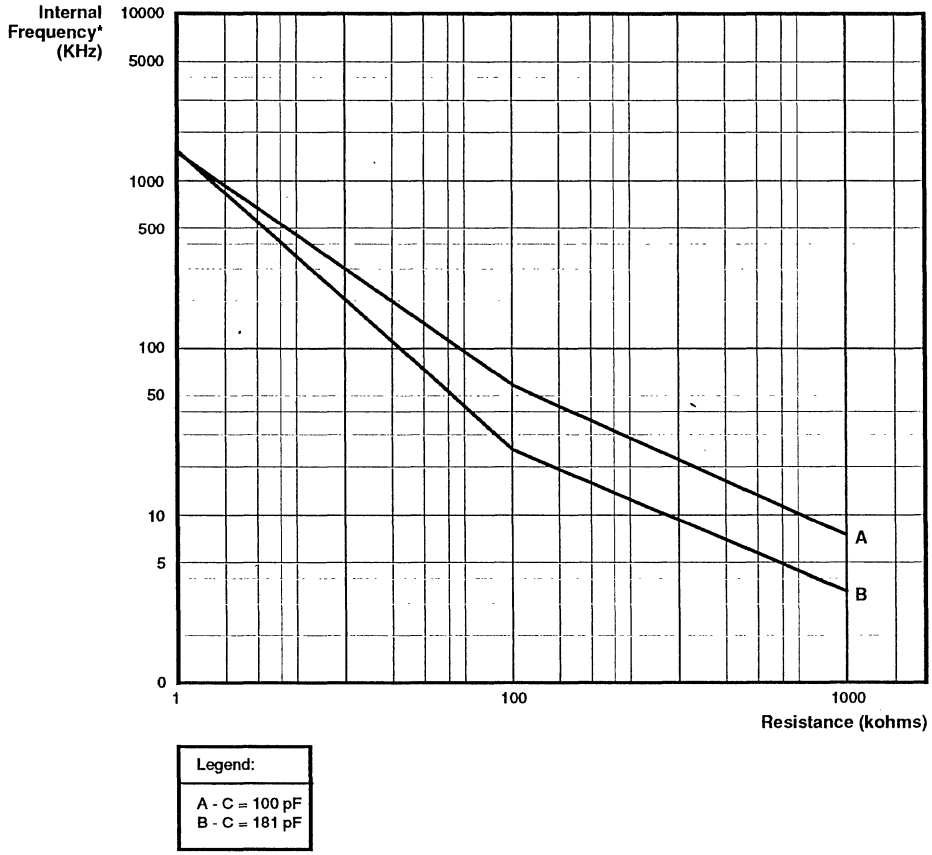
Figure 51. Typical Auto-Latch Current vs Temperature



Legend:
A - Vcc = 5.0V C = 33pF B - Vcc = 3.3V C = 33pF

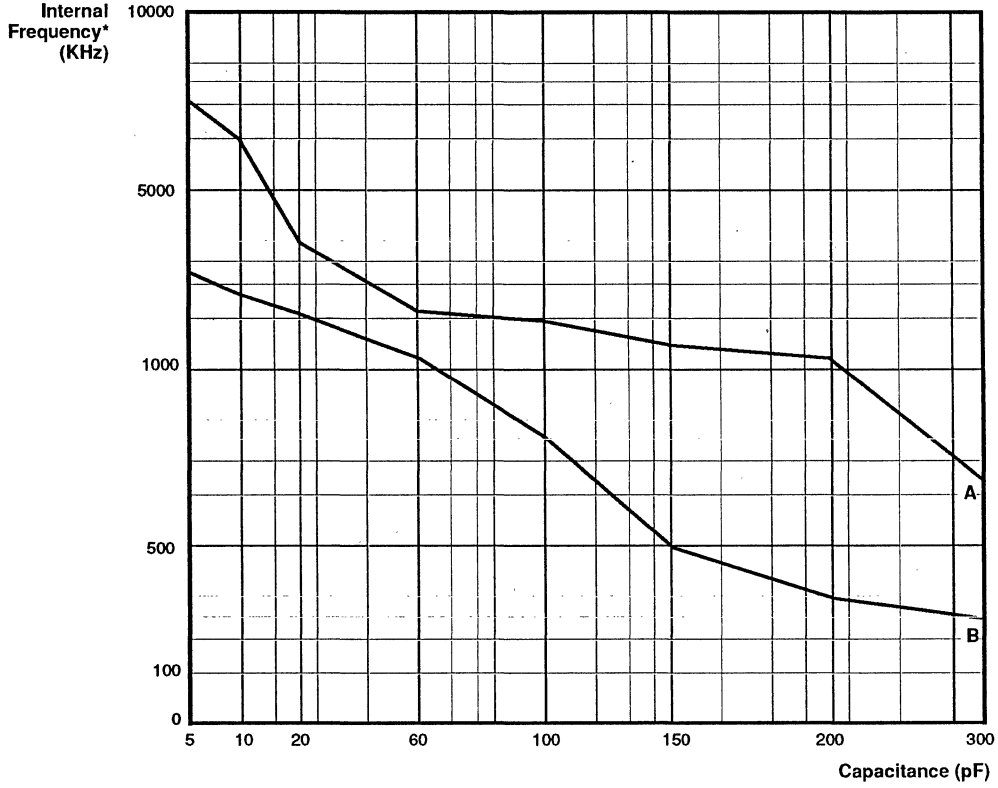
Note: * The internal clock frequency is one half the external clock frequency.
This chart for reference only. Each process will have a different characteristic curve.

Figure 52. Typical Internal Frequency vs RC Resistance



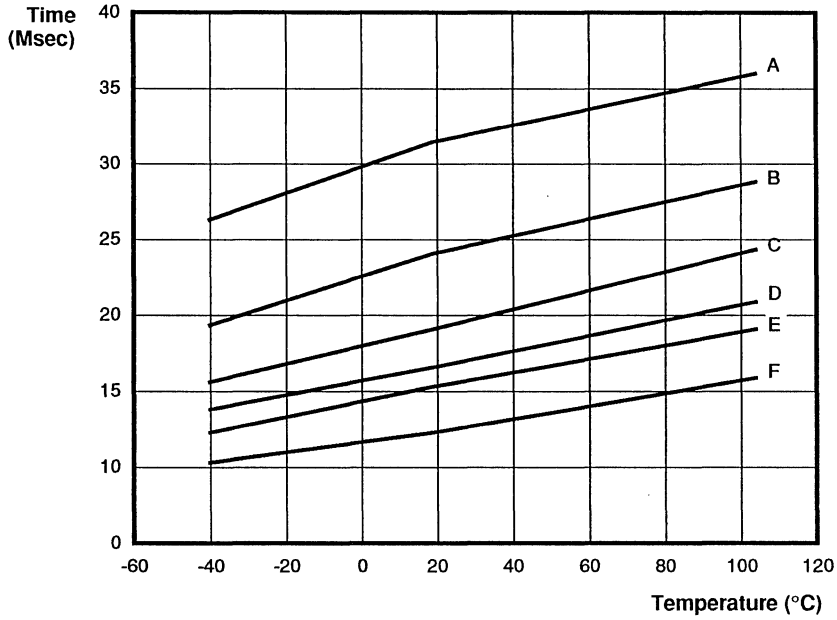
Note: * The internal clock frequency is one half the external clock frequency.
This chart for reference only. Each process will have a different characteristc curve.

Figure 53. Typical Internal Frequency vs Resistance



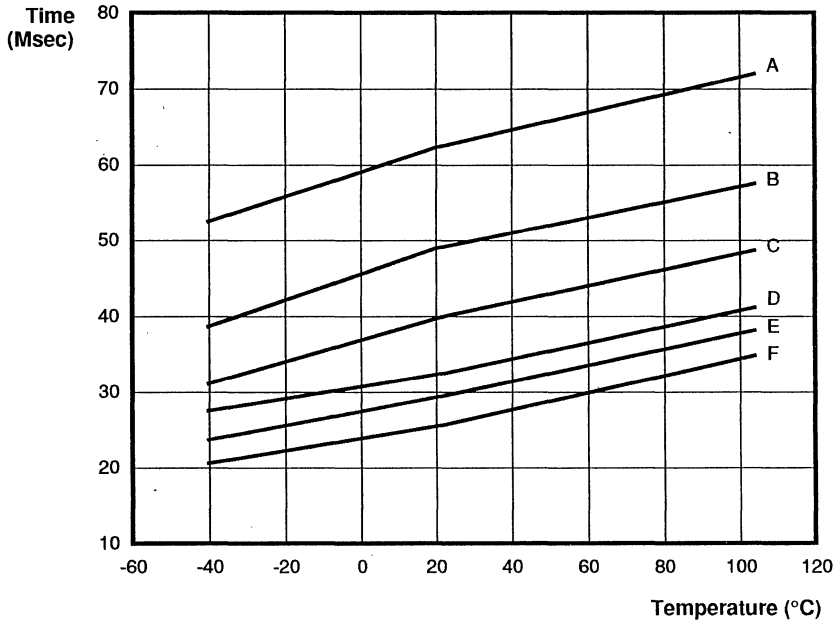
Note: * The internal clock frequency is one half the external clock frequency.
This chart for reference only. Each process will have a different characteristc curve.
R = 1 kohm

Figure 54. Typical Internal Frequency vs RC Capacitance



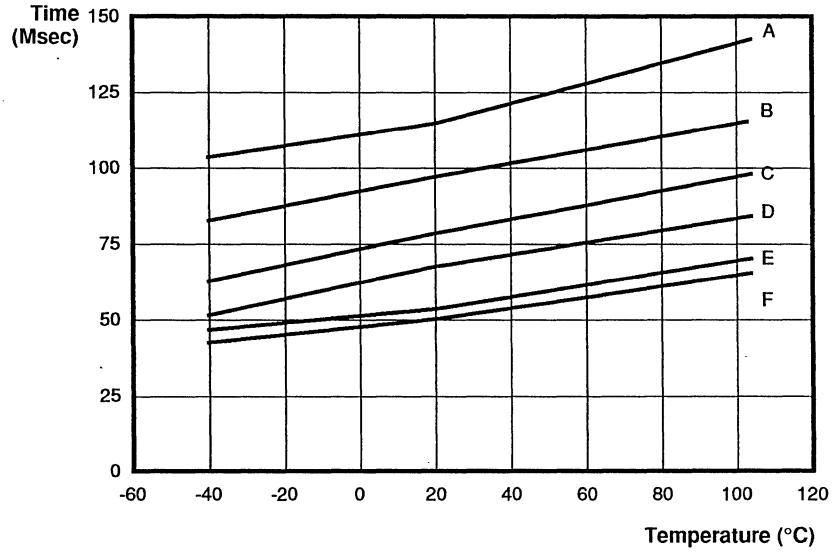
Legend:	
A - Vcc = 3.0V	D - Vcc = 4.5V
B - Vcc = 3.5V	E - Vcc = 5.0V
C - Vcc = 4.0V	F - Vcc = 5.5V

Figure 55. Typical 5 ms WDT Setting vs Temperature



Legend:	
A - Vcc = 3.0V	D - Vcc = 4.5V
B - Vcc = 3.5V	E - Vcc = 5.0V
C - Vcc = 4.0V	F - Vcc = 5.5V

Figure 56. Typical 15 ms WDT Setting vs Temperature



Legend:	
A - Vcc = 3.0V	D - Vcc = 4.5V
B - Vcc = 3.5V	E - Vcc = 5.0V
C - Vcc = 4.0V	F - Vcc = 5.5V

Figure 57. Typical 25 ms WDT Setting vs Temperature

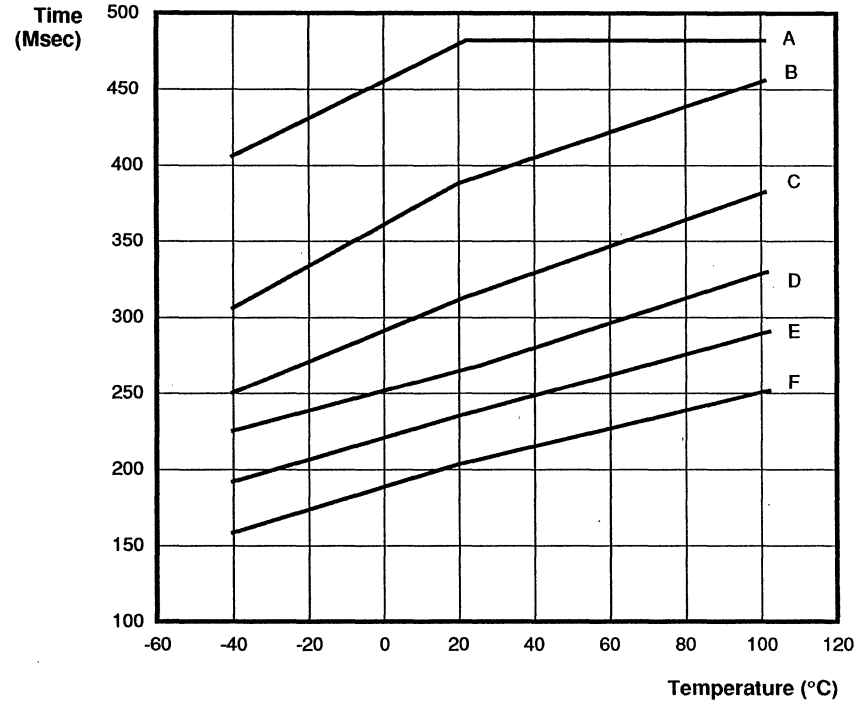


Figure 58. Typical 100 ms WDT Setting vs Temperature

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

Symbol	Meaning
IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

Symbol	Meaning
dst	Destination location or contents
src	Source location or contents
cc	Condition code
@	Indirect address prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag register (Control Register 252)
RP	Register Pointer (R253)
IMR	Interrupt mask register (R251)

Flags. Control register (R252) contains the following six flags:

Symbol	Meaning
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

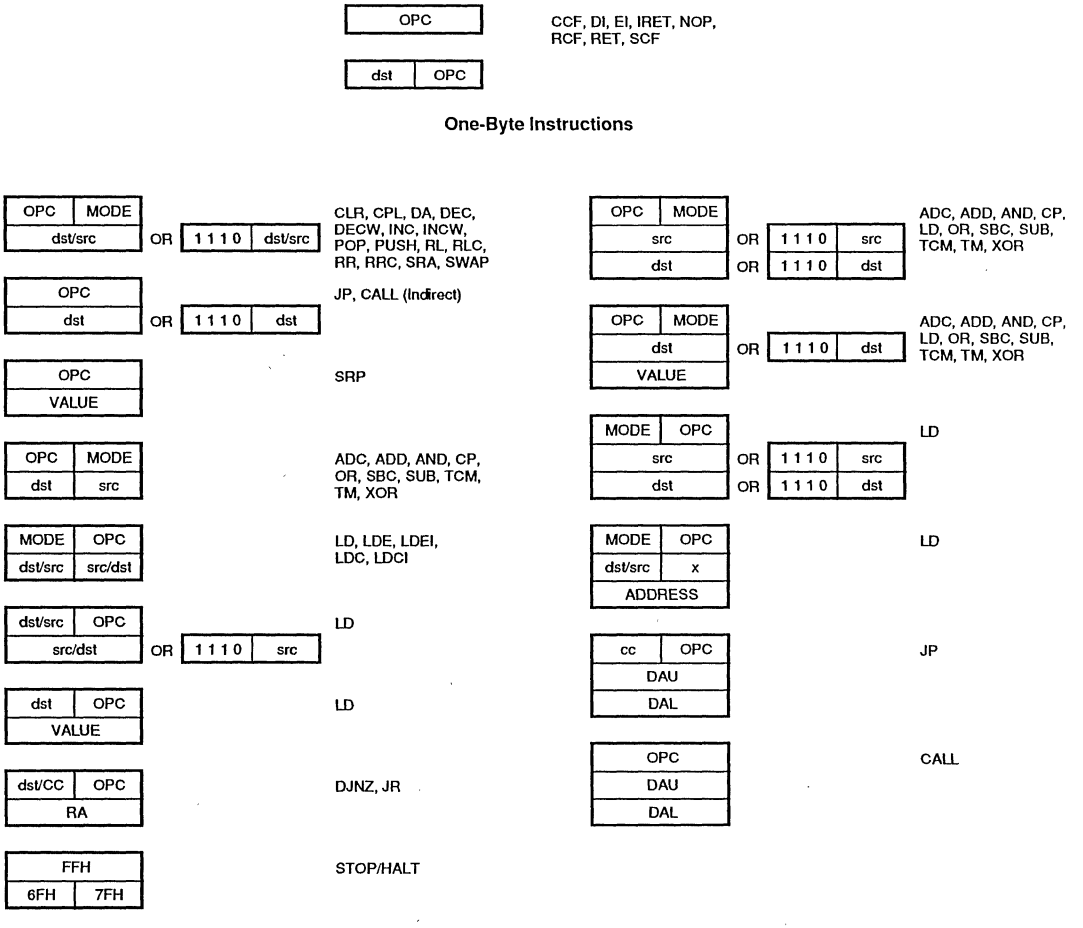
Affected flages are indicated by:

0	Clear to zero
1	Set to one
*	Set to clear according to operation
-	Unaffected
x	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000		Always True	
0111	C	Carry	C = 1
1111	NC	No Carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not Zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No Overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not Equal	Z = 0
1001	GE	Greater Than or Equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater Than	[Z OR (S XOR V)] = 0
0010	LE	Less Than or Equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned Greater Than or Equal	C = 0
0111	ULT	Unsigned Less Than	C = 1
1011	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned Less Than or Equal	(C OR Z) = 1
0000		Never True	

INSTRUCTION FORMATS



INSTRUCTION SUMMARY

Note: Assignment of a value is indicated by the symbol " ← ". For example:

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr (n)" is used to refer to bit (n) of a given operand location. For example:

dst (7)

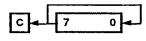
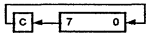
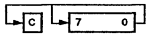
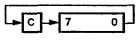
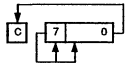
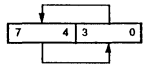
refers to bit 7 of the destination operand.

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst, src dst←dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst←dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst←dst AND src	†		5[ ]	-	*	*	0	-	-
CALL dst SP←SP - 2 @SP←PC, PC←dst	DA IRR		D6 D4	-	-	-	-	-	-
CCF C←NOT C			EF	*	-	-	-	-	-
CLR dst dst←0	R IR		B0 B1	-	-	-	-	-	-
COM dst dst←NOT dst	R IR		60 61	-	*	*	0	-	-
CP dst, src dst - src	†		A[ ]	*	*	*	*	-	-
DA dst dst←DA dst	R IR		40 41	*	*	*	X	-	-
DEC dst dst←dst - 1	R IR		00 01	-	*	*	*	-	-
DECW dst dst←dst - 1	RR IR		80 81	-	*	*	*	-	-
DI IMR(7)←0			8F	-	-	-	-	-	-
DJNZr, dst r←r - 1 if r ≠ 0 PC←PC + dst Range: +127, -128	RA		rA r = 0 - F	-	-	-	-	-	-
EI IMR(7)←1			9F	-	-	-	-	-	-
HALT			7F	-	-	-	-	-	-
INC dst dst←dst + 1	r R IR		rE r = 0 - F 20 21	-	*	*	*	-	-

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
INCW dst dst←dst + 1	RR IR		A0 A1	-	*	*	*	-	-
IRET FLAGS←@SP; SP←SP + 1 PC←@SP; SP←SP + 2; IMR(7)←1			BF	*	*	*	*	*	*
JP cc, dst if cc is true, PC←dst	DA IRR		cD c = 0 - F 30	-	-	-	-	-	-
JR cc, dst if cc is true, PC←PC + dst Range: +127, -128	RA		cB c = 0 - F	-	-	-	-	-	-
LD dst, src dst←src	r r R r X r Ir R R R IR IR	Im R r X r Ir r R IR IM IM R	rC r8 r9 r = 0 - F C7 D7 E3 F3 E4 E5 E6 E7 F5	-	-	-	-	-	-
LDC dst, src dst←src	r Irr	Irr r	C2 D2	-	-	-	-	-	-
LDCI dst, src dst←src r←r + 1;rr←rr + 1	Ir Irr	Irr Ir	C3 D3	-	-	-	-	-	-
LDE dst, src dst←src	r Irr	Irr Ir	82 92	-	-	-	-	-	-
LDEI dst, src dst←src r←r + 1;rr←rr + 1	Ir Irr	Irr Ir	83 93	-	-	-	-	-	-
NOP			FF	-	-	-	-	-	-

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
OR dst, src dst←dst OR src	†		4[ ]	-	*	*	0	-	-
POP dst dst←@SP; SP←SP + 1	R IR		50 51	-	-	-	-	-	-
PUSH src SP←SP - 1; @SP←src	R IR		70 71	-	-	-	-	-	-
RCF C←0			CF	0	-	-	-	-	-
RET PC←@SP; SP←SP + 2			AF	-	-	-	-	-	-
RL dst 	R IR		90 91	*	*	*	*	-	-
RLC dst 	R IR		10 11	*	*	*	*	-	-
RR dst 	R IR		E0 E1	*	*	*	*	-	-
RRC dst 	R IR		C0 C1	*	*	*	*	-	-
SBC dst, src dst←dst←src←C	†		3[ ]	*	*	*	*	1	*
SCF C←1			DF	1	-	-	-	-	-
SRA dst 	R IR		D0 D1	*	*	*	0	-	-
SRP dst RP←src	Im		31	-	-	-	-	-	-
STOP			6F	-	-	-	-	-	-
SUB dst, src dst←dst←src	†		2[ ]	*	*	*	*	1	*
SWAP dst 	R IR		F0 F1	X	*	*	X	-	-
TCM dst, src (NOT dst) AND src	†		6[ ]	-	*	*	0	-	-
TM dst, src dst AND src	†		7[ ]	-	*	*	0	-	-
XOR dst, src dst←dst XOR src	†		B[ ]	-	*	*	0	-	-

† These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

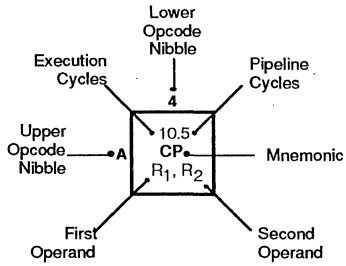
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address Mode dst	Address Mode src	Lower Opcode Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

OPCODE MAP

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								4.0 **WDH**
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								5.0 **WDT**
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								6.0 **STOP**
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								7.0 **HALT**
8	10.5 **DECW** RR1	10.5 **DECW** IR1	12.0 **LDE** r1, Irr2	18.0 **LDEI** Ir1, Irr2												6.1 **DI**
9	6.5 **RL** R1	6.5 **RL** IR1	12.0 **LDE** r2, Irr1	18.0 **LDEI** Ir2, Irr1												6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14.0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2				10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2	20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6.5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2		3		2		3		1							



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_2 = Dst address
R_1 or r_2 = Src address

Sequence:
Opcode, First Operand, Second Operand

Note: Blank areas not defined.

* 2-byte instruction appears as a 3-byte instruction

PRODUCT SPECIFICATION

Z86C91

CMOS Z8® ROMLESS MICROCONTROLLER

FEATURES

- 8-bit CMOS microcontroller, 40-pin DIP or 44-pin PLCC and QFP package
- 4.5 to 5.5 Voltage operating range
- Low power Consumption - 275 mW (max) @ 20 MHz
- Fast instruction pointer - 1.0 microsecond @ 12 MHz
- Two standby modes - STOP and HALT
- 32 input/output lines
- Full-Duplex UART
- All digital inputs are TTL levels
- Auto Latches
- ROMless
- 236 bytes of RAM
- Two programmable 8-bit Counter/Timers each with 6-bit programmable prescaler.
- Six vectored, priority interrupts from eight different sources
- Clock speeds 12, 16 and 20 MHz
- On-chip oscillator that accepts a crystal, ceramic resonator, LC or external clock drive.

GENERAL DESCRIPTION

The Z86C91 microcontroller (MCU) introduces a new level of sophistication to single-chip architecture. The Z86C91 is a member of the ROMless Z8 single-chip microcontroller family with 236 bytes of RAM.

The MCU is housed in a 40-pin DIP, 44-pin Leaded Chip-Carrier, or a 44-pin Quad Flat Pack, and is manufactured in CMOS technology. The Z86C91 is a ROMless part and offers the use of external memory which enables this Z8 microcontroller to be used where code flexibility is required.

Zilog's CMOS microcontroller offers fast execution, efficient use of memory, sophisticated interrupts, input/output bit manipulation capabilities, and easy hardware/software system expansion along with low cost and low power consumption.

The Z86C91 architecture is characterized by Zilog's 8-bit microcontroller core. The device offers a flexible I/O scheme, an efficient register and address space structure, multiplexed capabilities between address/data, I/O, and a number of ancillary features that are useful in many industrial and advanced scientific applications.

The device applications demand powerful I/O capabilities. The Z86C91 fulfills this with 24-pin dedicated to input and output. These lines are grouped into four ports. Each port consists of eightlines, and is configurable under software control to provided timing, status signals, serial or parallel I/O with or without handshake, and an address/data bus for interfacing external memory.

There are three basic address spaces available to support this wide range of configuration: Program Memory, Data Memory and 236 General-Purpose Registers.

To unburden the program from coping with the real-time problems such as counting/timing and serial data communication, the Z86C91 offers two on-chip counter/timers with a large number of user selectable modes, and an asynchronous receiver/transmitter (UART-Figure 1).

Note: All Signals with a preceding front slash, "/", are active Low, e.g.: B//W (WORD is active Low); /B/W (BYTE is active Low, only)

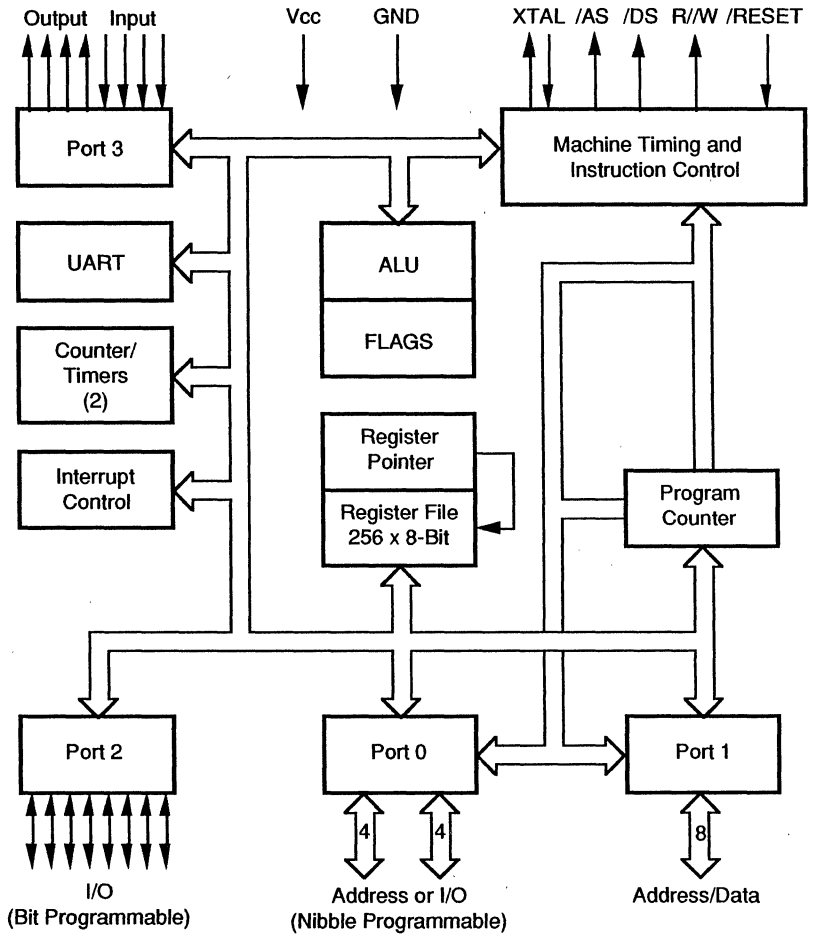


Figure 1. Functional Block Diagram

PIN DESCRIPTION

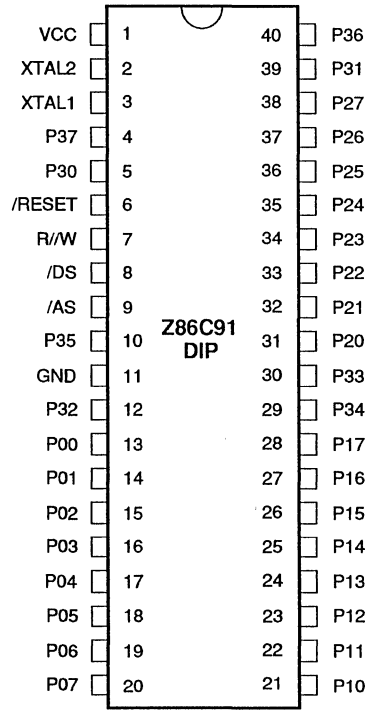


Figure 2. 40-Pin Dual In-Line Plastic Pin Assignments

Table 1. 40-Pin Dual In-Line Plastic Pin Identification

Pin #	Symbol	Function	Direction
1	V_{cc}	Power Supply	Input
2	XTAL2	Crystal, Oscillator Clock	Output
3	XTAL1	Crystal, Oscillator Clock	Input
4	P37	Port 3 pin 7	Output
5	P30	Port 3 pin 0	Input
6	/RESET	Reset	Input
7	R//W	Read/Write	Output
8	/DS	Data Strobe	Output
9	/AS	Address Strobe	Output
10	P35	Port 3 pin 5	Output
11	GND	Ground, GND	Input
12	P32	Port 3 pin 2	Input
13-20	P00-P07	Port 0 pin 0,1,2,3,4,5,6,7	In/Output
21-28	P10-P17	Port 1 pin 0,1,2,3,4,5,6,7	In/Output
29	P34	Port 3 pin 4	Output
30	P33	Port 3 pin 3	Input
31-38	P20-P27	Port 2 pin 0,1,2,3,4,5,6,7	In/Output
39	P31	Port 3 pin 1	Input
40	P36	Port 3 pin 6	Output

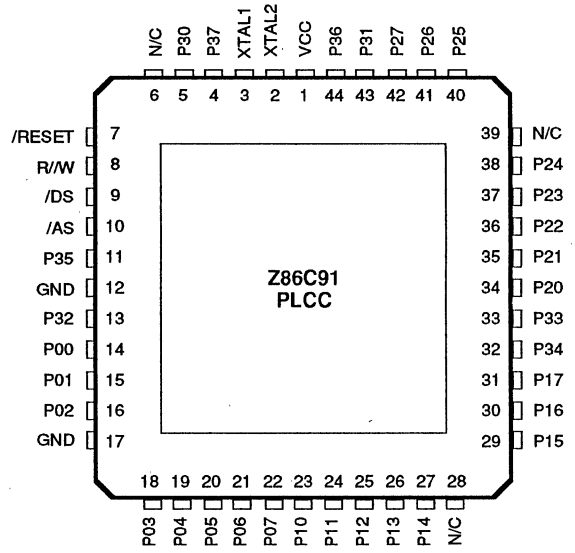


Figure 3. 44-Pin Plastic Leaded Chip Carrier Pin Assignments

Table 2. 44-Pin Plastic Leaded Chip Carrier Pin Identification

Pin #	Symbol	Function	Direction
1	V_{cc}	Power Supply	Input
2	XTAL2	Crystal, Oscillator Clock	Output
3	XTAL1	Crystal, Oscillator Clock	Input
4	P37	Port 3 pin 7	Output
5	P30	Port 3 pin 0	Input
6	N/C	Not Connected	Input
7	/RESET	Reset	Input
8	R//W	Read/Write	Output
9	/DS	Data Strobe	Output
10	/AS	Address Strobe	Output
11	P35	Port 3 pin 5	Output
12	GND	Ground	Input
13	P32	Port 3 pin 2	Input

Pin #	Symbol	Function	Direction
14-16	P00-P02	Port 0 pin 0,1,2	In/Output
17	GND	Ground	Input
18-22	P03-P07	Port 0 pin 3,4,5,6,7	In/Output
23-27	P10-P14	Port 1 pin 0,1,2,3,4	In/Output
28	N/C	Not Connected	Input
29-31	P15-P17	Port 1 pin 5,6,7	In/Output
32	P34	Port 3 pin 4	Output
33	P33	Port 3 pin 3	Input
34-38	P20-P24	Port 2 pin 0,1,2,3,4	In/Output
39	N/C	Not Connected	Input
40-42	P25-P27	Port 2 pin 5,6,7	In/Output
43	P31	Port 3 pin 1	Input
44	P36	Port 3 pin 6	Output

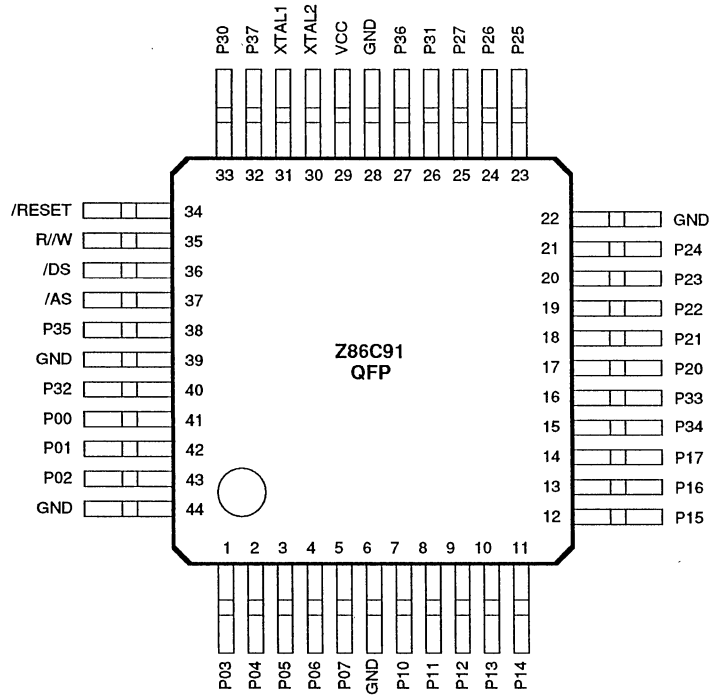


Figure 4. 44-Pin Quad Flat Pack Pin Assignments

Table 3. 44-Pin Quad Flat Pack Pin Identification

Pin #	Symbol	Function	Direction
1-5	P03-P07	Port 0 pin 3,4,5,6,7	In/Output
6	GND	Ground	Input
7-14	P10-P17	Port 1 pin 0,1,2,3,4,5,6,7	In/Output
15	P34	Port 3 pin 4	Output
16	P33	Port 3 pin 3	Input
17-21	P20-P24	Port 2 pin 0,1,2,3,4	In/Output
22	GND	Ground	Input
23-25	P25-P27	Port 2 pin 5,6,7	In/Output
26	P31	Port 3 pin 1	Input
27	P36	Port 3 pin 6	Output
28	GND	Ground	Input
29	V_{cc}	Power Supply	Input
30	XTAL2	Crystal, Oscillator Clock	Output

Pin #	Symbol	Function	Direction
31	XTAL1	Crystal, Oscillator Clock	Input
32	P37	Port 3 pin 7	Output
33	P30	Port 3 pin 0	Input
34	/RESET	Reset	Input
35	R//W	Read/Write	Output
36	/DS	Data Strobe	Output
37	/AS	Address Strobe	Output
38	P35	Port 3 pin 5	Output
39	GND	Ground	Input
40	P32	Port 3 pin 2	Input
41-43	P00-P02	Port 0 pin 0,1,2	In/Output
44	GND	Ground	Input

PIN FUNCTIONS

/DS. (output, active Low). Data Strobe is activated once for each external memory transfer. For a READ operation, data must be available prior to the trailing edge of /DS. For WRITE operations, the falling edge of /DS indicates that output data is valid.

/AS. (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Address output is via Port 1 for all external program. Memory address transfers are valid at the trailing edge of /AS. Under program control, /AS can be placed in the high-impedance state along with Ports 0 and 1, Data Strobe, and Read/Write.

XTAL1, XTAL2. *Crystal 1, Crystal 2* (time-based input and output, respectively). These pins connect a parallel-resonant crystal, ceramic resonator, LC, or any external single-phase clock to the on-chip oscillator and buffer.

R/W. (output, write Low). The Read//Write signal is low when the MCU is writing to the external program or data memory.

/RESET. (input, active-Low). To avoid asynchronous and noisy reset problems, the Z86C91 is equipped with a reset filter of four external clocks (4TpC). If the external /RESET signal is less than 4TpC in duration, no reset occurs.

On the 5th clock after the /RESET is detected, an internal RST signal is latched and held for an internal register count of 18 external clocks, or for the duration of the external /RESET, whichever is longer. During the reset cycle, /DS is held active low while /AS cycles at a rate of TpC/2. When /RESET is deactivated program execution begins at location 000C. Power-up reset time is held low for 50 mS, or until VCC is stable, whichever is longer.

Port 0. P00-P07. Port 0 is an 8-bit, nibble programmable, bidirectional, TTL compatible port. These eight I/O lines are configured under software control as a nibble I/O port, or as an address port for interfacing external memory. When used as an I/O port, Port 0 may be placed under handshake control. In this configuration, Port 3, lines P32 and P35 are used as the handshake control /DAV0 and RDY0 (Data available and Ready). Handshake signal assignment is dictated by the I/O direction of the upper nibble P04-P07. The lower nibble must have the same direction as the upper nibble to be under handshake control.

For external memory references, Port 0 provides Address bits A11-A8 (lower nibble) or A15-A8 (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. If one or both nibbles are needed for I/O operation, they must be configured by writing to the Port 0 Mode register. After a hardware reset, Port 0 lines are defined as address lines A15-A8, and extended timing is set to accommodate slow memory access. The initialization routine includes reconfiguration to eliminate this extended timing mode (Figure 5).

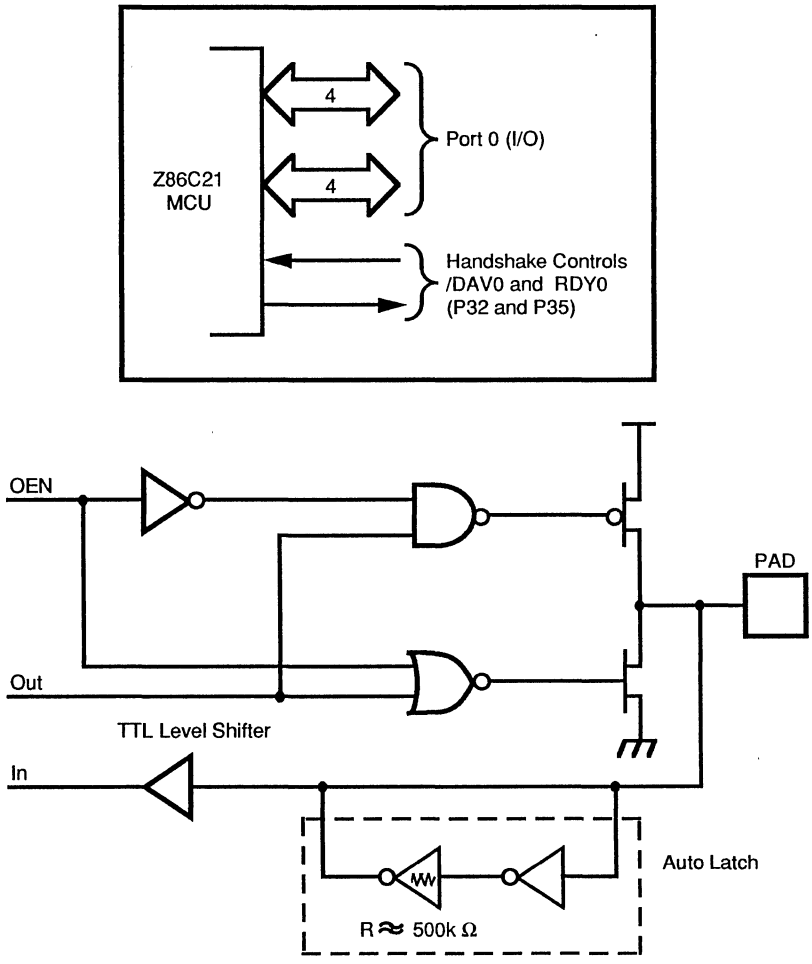


Figure 5. Port 0 Configuration

PIN FUNCTIONS (Continued)

Port 1. (P10-P17). Port 1 is an 8-bit, TTL compatible port. It has multiplexed Address (A7-A0) and Data (D7-D0) ports for interfacing external memory.

If more than 256 external locations are required, Port 0 must output the additional lines.

Port 1 can be placed in a high-impedance state along with Port 0, /AS, /DS and R//W, allowing the MCU to share common resources in multiprocessor and DMA applications. Data transfers are controlled by assigning P33 as a Bus Acknowledge input, and P34 as a Bus request output (Figure 6).

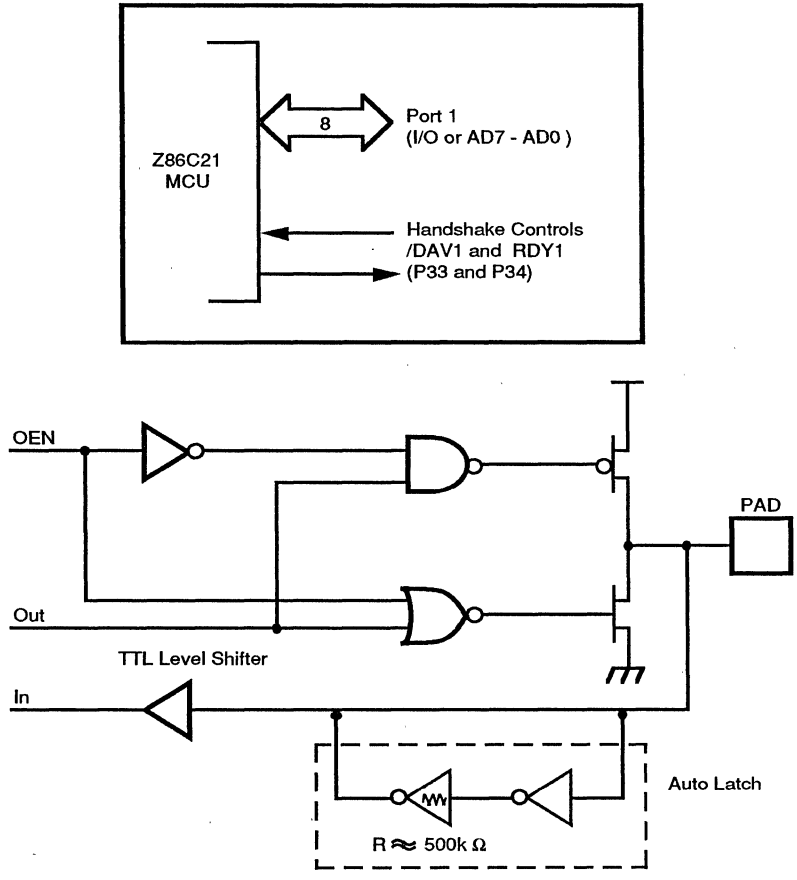


Figure 6. Port 1 Configuration

Port 2. (P20-P27). Port 2 is an 8-bit, bit programmable, bidirectional, TTL compatible port. Each of these eight I/O lines are independently programmed as an input or output or globally as an open-drain output. Port 2 is always available for I/O operation. When used as an I/O port, Port 2 is placed under handshake control. In this configuration, Port 3 lines P31 and P36 are used as the handshake control lines /DAV2 and RDY2. The handshake signal assignment for Port 3 lines P31 and P36 is dictated by the direction (input or output) assigned to P27 (Figure 7).

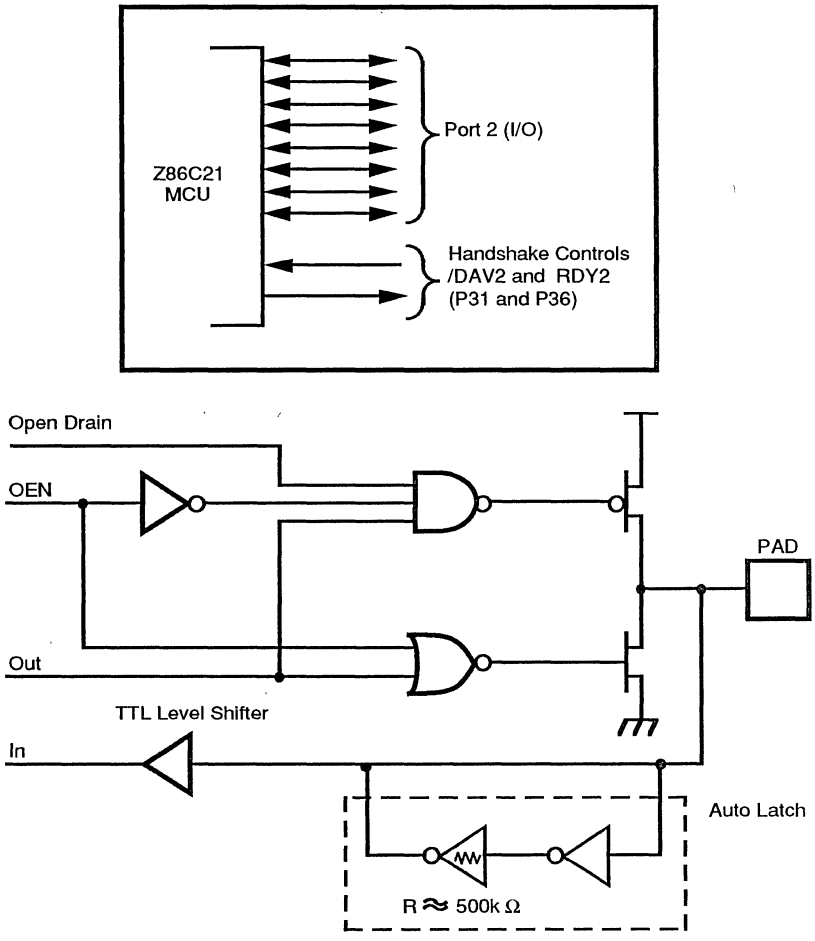


Figure 7. Port 2 Configuration

PIN FUNCTION (Continued)

Port 3. (P30-P37). Port 3 is an 8-bit, TTL compatible four fixed input and four fixed output port. These eight I/O lines have four-fixed (P33-P30) input and four fixed (P37-P34) output ports. Port 3, when used as serial I/O, are programmed as serial in and serial out, respectively (Figure 8).

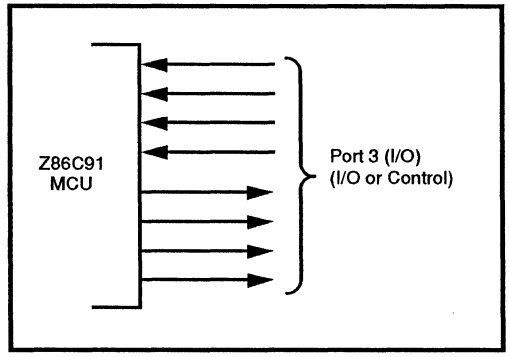


Figure 8. Port 3 Configuration

Port 3 is configured under software control to provide the following control functions: handshake for Ports 0 and 2 (/DAV and RDY); four external interrupt request signals (IRQ0-IRQ3); timer input and output signals (T_{IN} and T_{OUT}), and Data Memory Select (/DM).

Table 4. Port 3 Pin Assignments

Pin	I/O	CTC1	Int.	P0 HS	P1 HS	P2 HS	UART	Ext
P30	IN		IRQ3				Serial In	
P31	IN	T_{IN}	IRQ2			D/R		
P32	IN		IRQ0	D/R				
P33	IN		IRQ1		D/R			
P34	OUT				R/D			DM
P35	OUT			R/D				
P36	OUT	T_{OUT}				R/D		
P37	OUT						Serial Out	

Notes:
HS = HANDSHAKE SIGNALS
D = Data Available
R = Ready

Port 3 lines P30 and P37, are programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by the Counter/Timer 0.

The Z86C91 automatically adds a start bit and two stop bits to transmitted data (Figure 9). Odd parity is also available as an option. Eight data bits are always transmitted, regardless of parity selection. If parity is enabled, the eighth bit is the odd parity bit. An interrupt request (IRQ4) is generated on all transmitted characters.

Received data must have a start bit, 8 data bits and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ3 interrupt request.

Auto-Latch. The Auto-Latch puts valid CMOS levels on all CMOS inputs that are not externally driven. This reduces excessive supply current flow in the input buffer when it is not driven by any source.

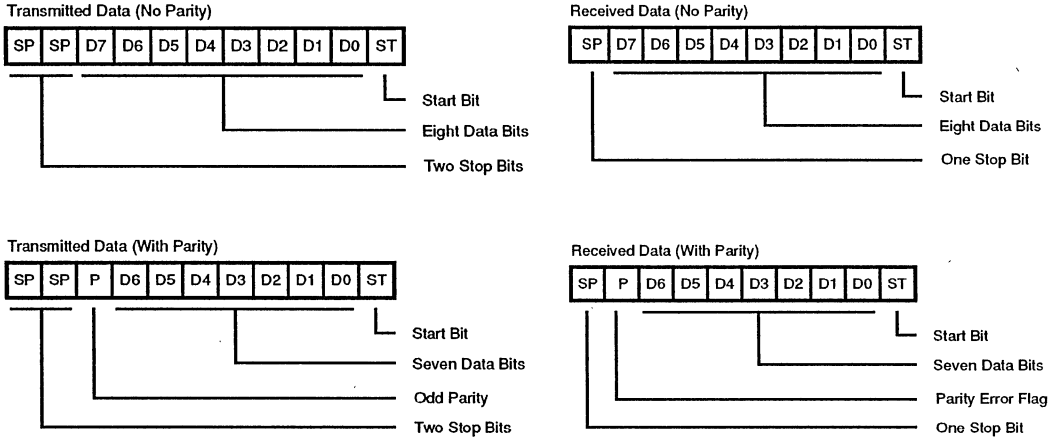


Figure 9. Serial Data Formats

FUNCTIONAL DESCRIPTION

Address Space

The following subsections define Program Memory, Data Memory, Register Files, and Stack Pointers.

Program Memory. The Z86C91 can address up to 64 Kbytes of external program memory (Figure 10). The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts. Program execution begins at external location 000CH after a reset.

Data Memory (/DM). The Z86C91 addresses up to 64 Kbytes of external data memory space. External data memory is included with, or separated from, the external program memory space. /DM, an optional I/O function that can be programmed to appear on pin P34, is used to distinguish between data and program memory space (Figure 11). The state of the /DM signal is controlled by the type instruction being executed. An LDC opcode references PROGRAM (/DM inactive) memory, and an LDE instruction references DATA (/DM active low) memory.

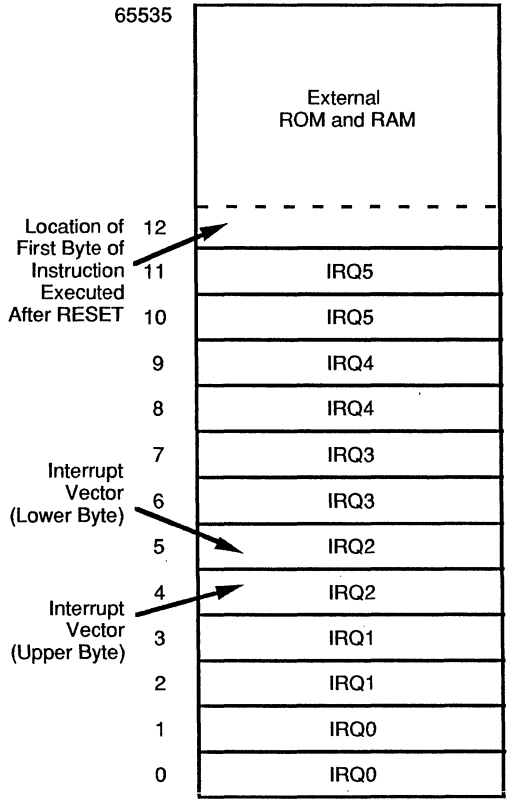


Figure 10. Program Memory Configuration

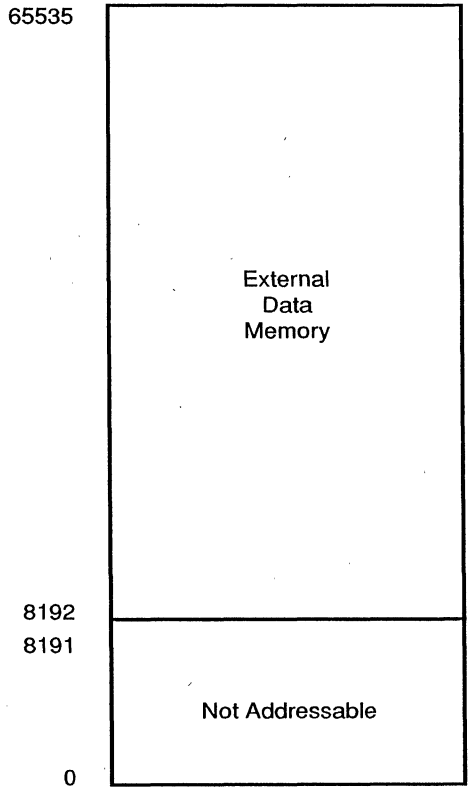


Figure 11. Data Memory Configuration

Register File. The Register File consists of three I/O port registers, 236 general-purpose registers and 16 control and status registers (Figure 12). The instructions can access registers directly or indirectly via an 8-bit address field. The Z86C91 also allows short 4-bit register addressing using the Register Pointer (Figure 13). In the 4-bit mode, the Register File is divided into 16 working register groups, each occupying 16 continuous locations. The Register Pointer addresses the starting location of the active working-register group.

Note: Register Bank E0-EF is only accessed through working register and indirect addressing modes.

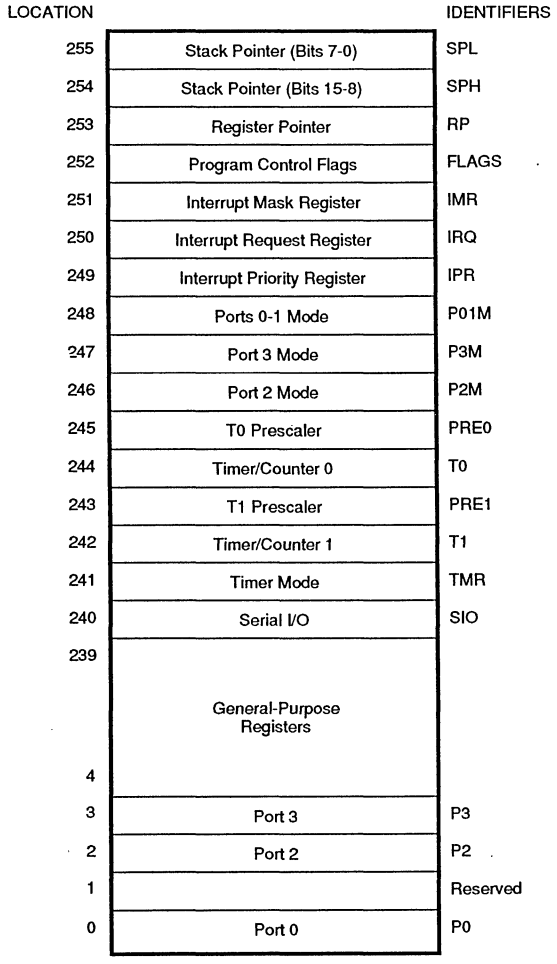


Figure 12. Register File

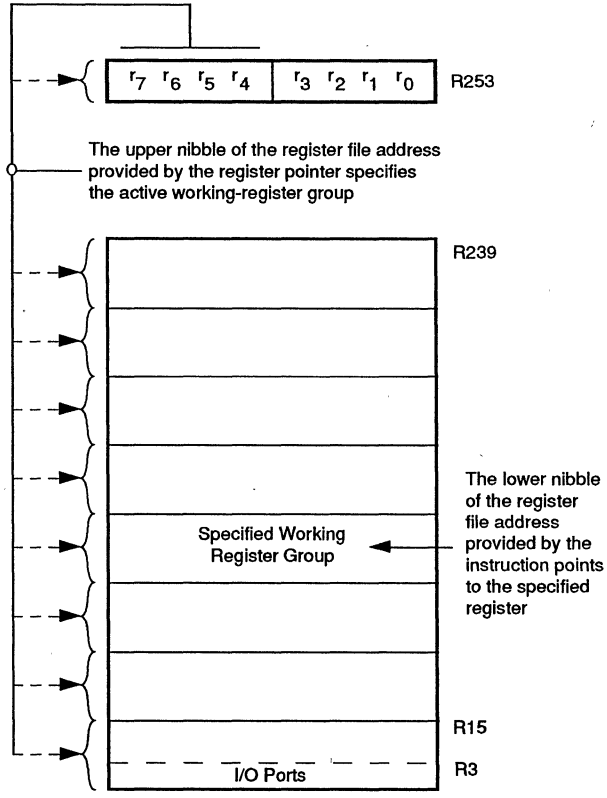


Figure 13. Register Pointer

Stack. The Z86C91 has a 16-bit Stack Pointer (R254-R255) used for external stack that resides anywhere in the data memory. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 236 general-purpose registers (R4-R239). The high byte of the Stack Pointer (SPH-Bit 8-15) is used as a general purpose register when using internal stack only.

Counter/Timers. There are two 8-bit programmable counter/timers (T0-T1), each driven by its own 6-bit programmable prescaler. The T1 prescaler is driven by internal or external clock sources; however, the T0 prescaler is driven by the internal clock only (Figure 14).

The 6-bit prescalers divide the input frequency of the clock source by any integer number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When both the counter and prescaler reach the end of the count, a timer interrupt request, IRQ4 (T0) or IRQ5 (T1), is generated.

The counter can be programmed to start, stop, restart to continue, or restart from the initial value. The counters can also be programmed to stop upon reaching zero (single pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode).

The counter, but not the prescalers, are read at any time without disturbing their value or count mode. The clock source for T1 is user-definable and can be either the internal microprocessor clock divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input (P31) as an external clock, a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. Port 3 line P36 also serves as a timer output (T_{OUT}) through which T0, T1 or sub the internal clock is output. The counter/timers are cascaded by connecting the T0 output to the input of T1

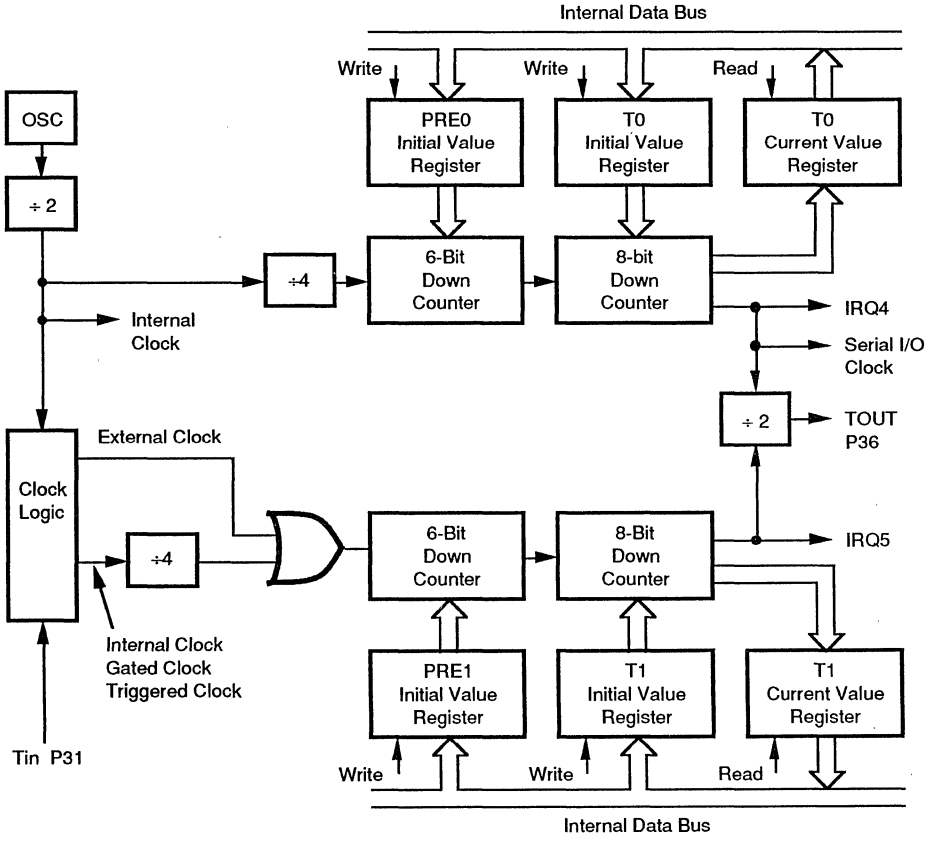


Figure 14. Counter/Timers Block Diagram

Interrupts. The Z86C91 has six different interrupts from eight different sources. The interrupts are maskable and prioritized. The eight sources are divided as follow: four sources are claimed by Port 3 lines P30-P33, one in Serial Out, one in Serial In, and two in the counter/timers (Figure 15). The Interrupt Mask Register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All Z86C91 interrupts are vectored through locations in the program memory. When an interrupt machine cycle is activated, an interrupt request is granted. Thus, this disables all of the subsequent interrupts, save the Program Counter and Status Flags, and then branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

To accommodate polled interrupt systems, interrupt inputs are masked and the Interrupt Request register is polled to determine which of the interrupt requests need service. Software initialed interrupts are supported by setting the appropriate bit in the Interrupt Request register (IRQ).

Internal interrupt requests are sampled on the falling edge of the last cycle of every instruction, and the interrupt request must be valid 5TpC before the falling edge of the last clock cycle of the currently executing instruction.

When the device samples a valid interrupt request, the next 48 (external) clock cycles are used to prioritize the interrupt, and push the two PC bytes and the FLAG register on the stack. The following nine cycles are used to fetch the interrupt vector from external memory. The first byte of the interrupt service routine is fetched beginning on the 58th TpC cycle following the internal sample point, which corresponds to the 63rd TpC cycle following the external interrupt sample point.

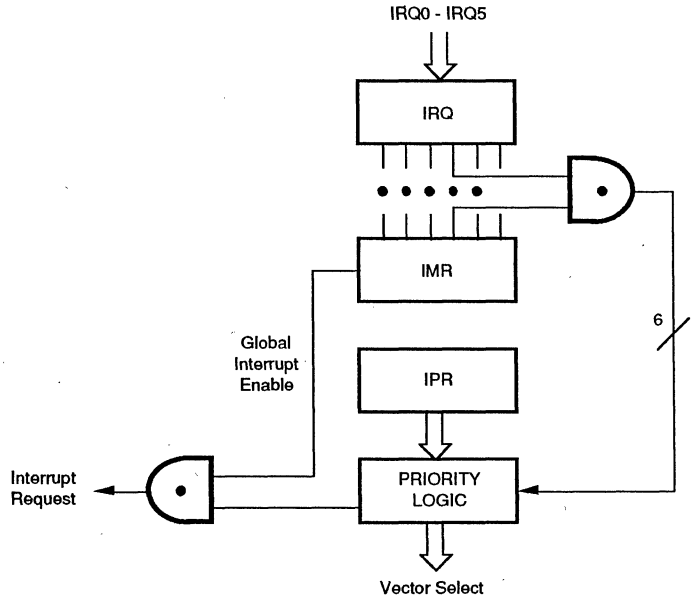


Figure 15. Interrupt Block Diagram

Clock. The Z86C91 on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal, LC, ceramic resonator, or any suitable external clock source (XTAL1=Input, XTAL2=Output). The crystal should be AT cut, 1 MHz to 20 MHz max, and series resistance (RS) less than or equal to 100 Ohms. The crystal should be connected across XTAL1 and XTAL2 using the recommended capacitors (10 pF < CL < 300 pF) from each pin to ground (Figure 16).

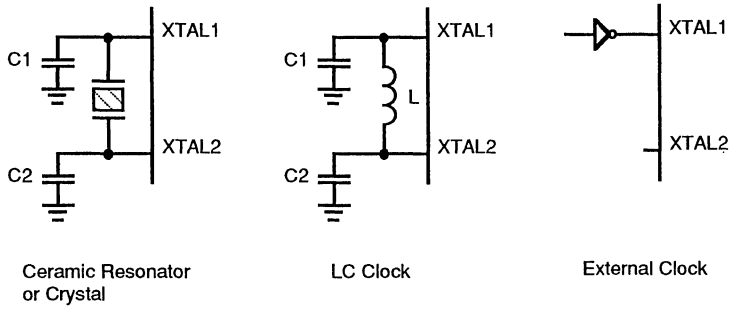


Figure 16. Oscillator Configuration

HALT. Turns off the internal CPU clock but not the XTAL oscillation. The counter/timers and the external interrupts IRQ0, IRQ1, IRQ2 and IRQ3 remain active. The devices are recovered by interrupts, either externally or internally generated.

STOP. This instruction turns off the internal clock and external crystal oscillation and reduces the standby current to 10 microamperes or less. The Stop mode is terminated by a reset, which cause the processor to restart the application program at address 000CH.

In order to enter STOP (or HALT) mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. To do this, the user must execute a NOP (opcode=OFFH) immediately before the appropriate sleep instruction,. i.e.:

```
FF  NOP   ; clear the pipeline
6F  STOP  ; enter STOP mode
          or
FF  NOP   ; clear the pipeline
7F  HALT  ; enter HALT mode
```

ABSOLUTE MAXIMUM RATINGS

Symbol	Description	Min	Max	Units
V_{CC}	Supply Voltage*	-0.3	+7.0	V
T_{STG}	Storage Temp	-65	+150	C
T_A	Oper Ambient Temp		†	C

Notes:
* Voltages on all pins with respect to GND.
† See Ordering Information

Stress greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for an extended period may affect device reliability.

STANDARD TEST CONDITIONS

The characteristics listed below apply for standard test conditions as noted. All voltages are referenced to GND. Positive current flows into the referenced pin (Figure 17).

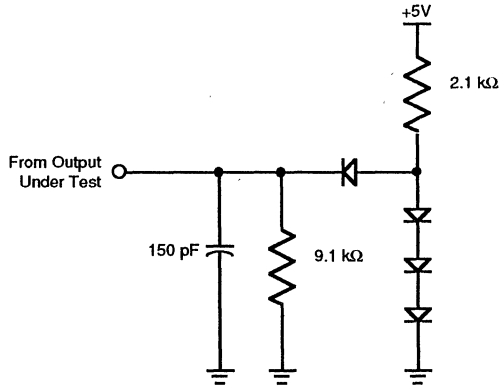


Figure 17. Test Load Diagram

DC CHARACTERISTICS

Sym	Parameter	T_A = 0°C to 70°C		T_A = -40°C to105°C		Typical at 25°C	Units	Conditions
		Min	Max	Min	Max			
	Max Input Voltage		7		7		V	I_{IN} 250µA
V_{CH}	Clock Input High Voltage	3.8	V_{CC}	3.8	V_{CC}		V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.03	0.8	-0.03	0.8		V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}	2.0	V_{CC}		V	
V_{IL}	Input Low Voltage	-0.3	0.8	-0.3	0.8		V	
V_{OH}	Output High Voltge	2.4		2.4			V	I_{OH} = -2.0 mA
V_{OH}	Output High Voltge	V_{CC} - 100mV		V_{CC} - 100mV			V	I_{OH} = -100 µA
V_{OL}	Output Low Voltage		0.4		0.4		V	I_{OH} = +2.0 mA
V_{RH}	Reset Input High Voltage	3.8	V_{CC}	3.8	V_{CC}		V	
V_{RI}	Reset Input Low Voltage	-0.03	0.8	-0.03	0.8		V	
I_{IL}	Input Leakage	-2	2	-2	2		µA	Test at 0V, V_{CC}
I_{OL}	Output Leakage	-2	2	-2	2		µA	Test at 0V, V_{CC}
I_{IR}	Reset Input Current		-80		-80		µA	V_{RL} = 0V
I_{CC}	Supply Current		30		30	20	mA	@ 12 MHz [1]
			35		35	24	mA	@ 16 MHz [1]
			50		50		mA	@ 20 MHz [1]
I_{CC1}	Standby Current		6.5		6.5	4	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 12 MHz [1]
			7		7	4.5	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 16 MHz [1]
			8.5		8.5		mA	HALT Mode V_{IN} = 0V, V_{CC} @ 20 MHz [1]
I_{CC2}	Standby Current		10		10	1	µA	STOP Mode V_{IN} = 0V, V_{CC} [1]
I_{ALL}	Auto Latch Low Current	-10	10	-14	14	5	µA	

Note:
[1] All inputs driven to 0V, V_{CC} and outputs floating.

AC CHARACTERISTICS

External I/O or Memory Read/Write Timing Diagram

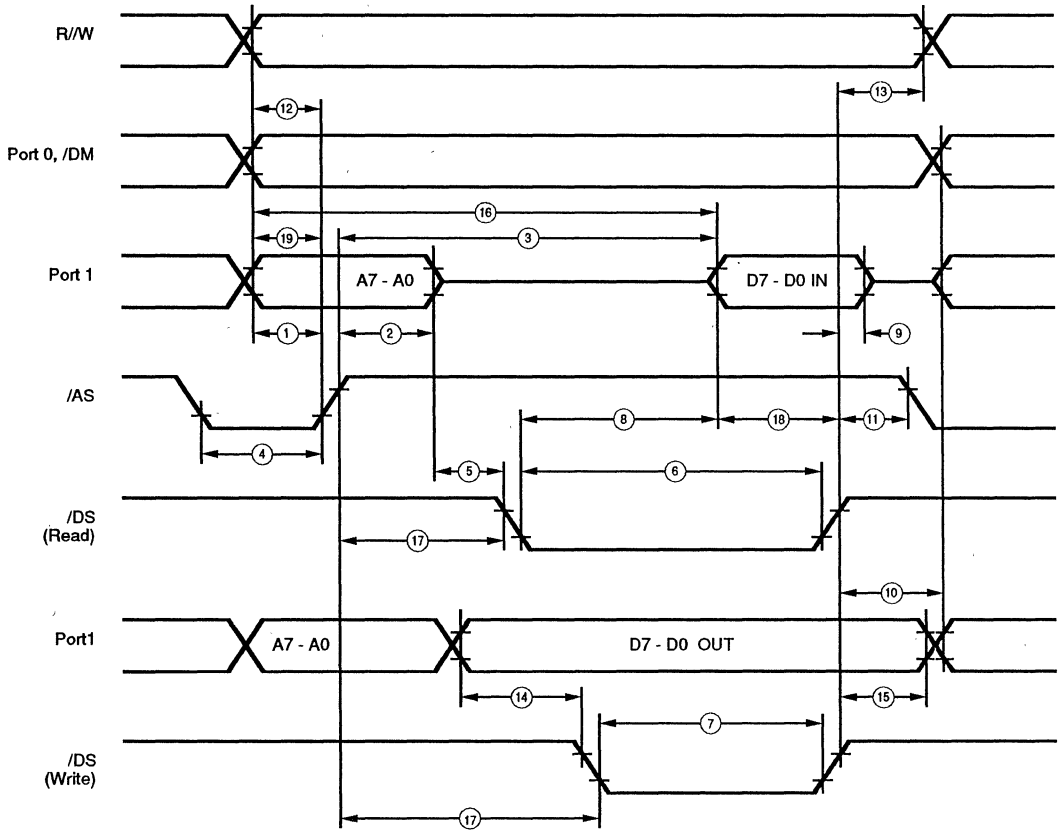


Figure 18. External I/O or Memory Read/Write Timing

AC CHARACTERISTICS

External I/O or Memory Read or Write Timing Table

No	Symbol	Parameter	T_A = 0°C to 70°C						T_A = -40°C to 105°C						Units	Notes
			12 MHz		16 MHz		20 MHz		12 MHz		16 MHz		20 MHz			
			Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max		
1	TdA(AS)	Address Valid to /AS Rise Delay	35		25		20		35		25		20		ns	[2,3]
2	TdAS(A)	/AS Rise to Address Float Delay	45		35		25		45		35		25		ns	[2,3]
3	TdAS(DR)	/AS Rise to Read Data Req'd Valid		250		180		150		250		180		150	ns	[1,2,3]
4	TwAS	/AS Low Width	55		40		30		55		40		30		ns	[2,3]
5	TdAZ(DS)	Address Float to /DS Fall	0		0		0		0		0		0		ns	
6	TwDSR	/DS (Read) Low Width	185		135		105		185		135		105		ns	[1,2,3]
7	TwDSW	/DS (Write) Low Width	110		80		65		110		80		65		ns	[1,2,3]
8	TdDSR(DR)	/DS Fall to Read Data Req'd Valid		130		75		55		130		75		55	ns	[1,2,3]
9	ThDR(DS)	Read Data to /DS Rise Hold Time	0		0		0		0		0		0		ns	[2,3]
10	TdDS(A)	/DS Rise to Address Active Delay	65		50		40		65		50		40		ns	[2,3]
11	TdDS(AS)	/DS Rise to /AS Fall Delay	45		35		25		45		35		25		ns	[2,3]
12	TdR/W(AS)	R/W Valid to /AS Rise Delay	30		25		20		33		25		20		ns	[2,3]
13	TdDS(R/W)	/DS Rise to R/W Not Valid	50		35		25		50		35		25		ns	[2,3]
14	TdDW(DSW)	Write Data Valid to /DS Fall (Write) Delay	35		25		20		35		25		20		ns	[2,3]
15	TdDS(DW)	/DS Rise to Write Data Not Valid Delay	55		35		25		55		35		25		ns	[2,3]
16	TdA(DR)	Address Valid to Read Data Req'd Valid		310		230		180		310		230		180	ns	[1,2,3]
17	TdAS(DS)	/AS Rise to /DS Fall Delay	65		45		35		65		45		35		ns	[2,3]
18	TdDI(DS)	Data Input Setup to /DS Rise	75		60		50		75		60		50		ns	[1,2,3]
19	TdDM(AS)	/DM Valid to /AS Rise Delay	50		30		20		50		30		20		ns	[2,3]

Notes:

[1] When using extended memory timing add 2 TpC.
[2] Timing numbers given are for minimum TpC.
[3] See clock cycle dependent characteristics table.

Standard Test Load
All timing references use 2.0V for a logic 1 and 0.8V for a logic 0.

Clock Dependent Formulas

Number	Symbol	Equation
1	TdA(AS)	0.40TpC + 0.32
2	TdAS(A)	0.59TpC - 3.25
3	TdAS(DR)	2.38TpC + 6.14
4	TwAS	0.66TpC - 1.65
6	TwDSR	2.33TpC - 10.56
7	TwDSW	1.27TpC + 1.67
8	TdDSR(DR)	1.97TpC - 42.5
10	TdDS(A)	0.8TpC
11	TdDS(AS)	0.59TpC - 3.14
12	TdR/W(AS)	0.4TpC
13	TdDS(R/W)	0.8TpC - 15
14	TdDW(DSW)	0.4TpC
15	TdDS(DW)	0.88TpC - 19
16	TdA(DR)	4TpC - 20
17	TdAS(DS)	0.91TpC - 10.7
18	TsDI(DS)	0.8TpC - 10
19	TdDM(AS)	0.9TpC - 26.3

AC CHARACTERISTICS
Additional Timing Diagram

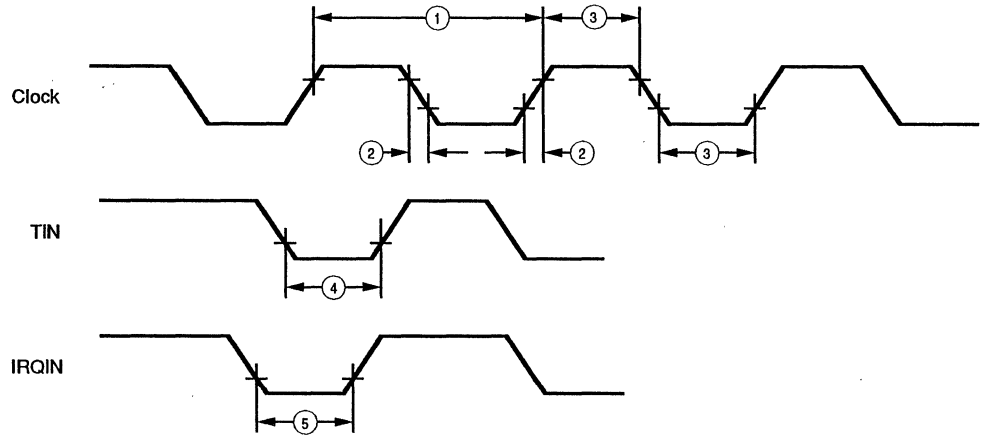


Figure 19. Additional Timing

AC CHARACTERISTICS
Additional Timing Table

No	Symbol	Parameter	T_A = 0°C to 70°C						T_A = -40°C to 105°C						Units	Notes
			12 MHz		16 MHz		20 MHz		12 MHz		16 MHz		20 MHz			
			Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max		
1	TpC	Input Clock Period	83	1000	62.5	1000	50	1000	83	1000	62.5	1000	50	1000	ns	[1]
2	TrC,TfC	Clock Imput Rise & Fall Times		15		10		10		15		10		10	ns	[1]
3	TwC	Input Clock Width	35		25		15		35		25		15		ns	[1]
4	TwTinL	Timer Input Low Width	75		75		75		75		75		75		ns	[2]
5	TwTinH	Timer Input High Width	3TpC		3TpC		3TpC		3TpC		3TpC		3TpC			[2]
6	TpTin	Timer Input Period	8TpC		8TpC		8TpC		8TpC		8TpC		8TpC			[2]
7	TrTin,TfTin	Timer Input Rise & Fall Times	100		100		100		100		100		100		ns	[2]
8A	TwIL	Interrupt Request Input Low Times	70		70		70		70		70		70		ns	[2,4]
8B	TwIL	Interrupt Request Input Low Times	3TpC		3TpC		3TpC		3TpC		3TpC		3TpC			[2,5]
9	TwIH	Interrupt Request Input High Times	3TpC		3TpC		3TpC		3TpC		3TpC		3TpC			[2,3]

Notes:
[1] Clock timing references use 3.8V for a logic 1 and 0.8V for a logic 0.
[2] Timing references use 2.0V for a logic 1 and 0.8V for a logic 0.
[3] Interrupt references request via Port 3.
[4] Interrupt request via Port 3 (P31-P33).
[5] Interrupt request via Port 30.

AC CHARACTERISTICS
Handshake Timing Diagrams

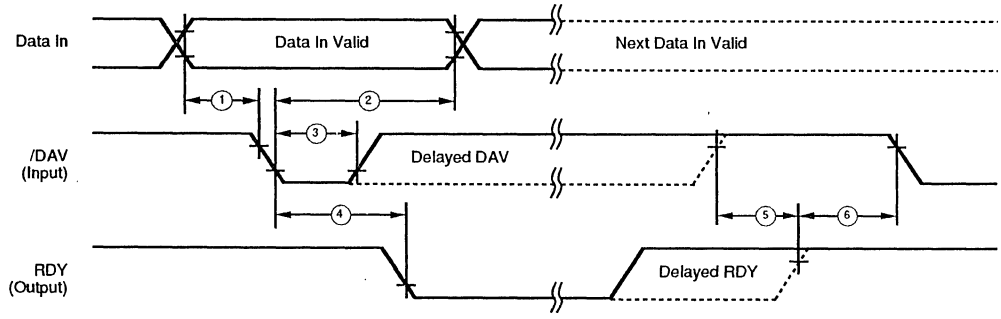


Figure 20. Input Handshake Timing

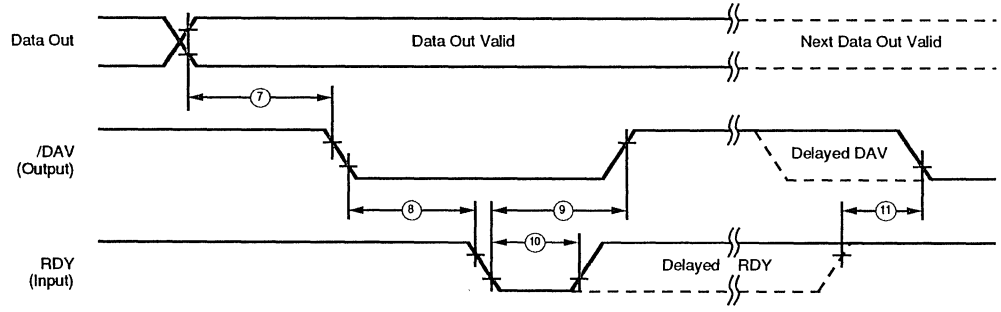


Figure 21. Output Handshake Timing

AC CHARACTERISTICS
Handshake Timing Table

No	Symbol	Parameter	T_A = 0°C TO 70°C 12, 16, and 20 MHz Min	Max	T_A = -40°C TO 105°C 12, 16, and 20 MHz Min	Max	Notes (Data Direction)
1	TsDI(DAV)	Data In Setup Time	0		0		IN
2	ThDI(DAV)	Data In Hold Time	145		145		IN
3	TwDAV	Data Available Width	110		110		IN
4	TdDAVI(RDY)	DAV fall to RDY fall Delay		115		115	IN
5	TdDAVId(RDY)	DAV rise to RDY rise Delay		115		115	IN
6	TdDO(DAV)	RDY rise to DAV fall Delay	0		0		IN
7	TcLDAV0(RDY)	Data Out to DAV fall Delay		TpC		TpC	OUT
8	TcLDAV0(RDY)	DAV fall to RDY fall Delay	0		0		OUT
9	TdRDY0(DAV)	RDY fall to DAV rise Delay		115		115	OUT
10	TwRDY	RDY Width	110		110		OUT
11	TdRDY0d(DAV)	RDY rise to DAV fall Delay		115		115	OUT

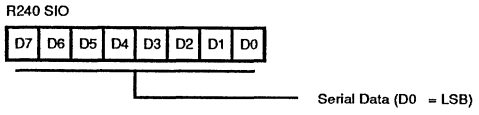


Figure 22. Serial I/O Register (F0H: Read/Write)

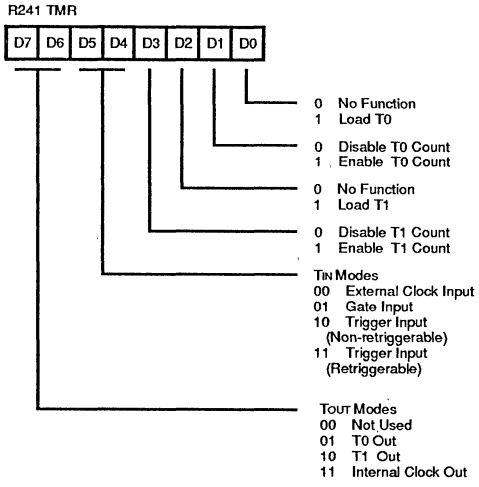


Figure 23. Timer Mode Register (F1H: Read/Write)

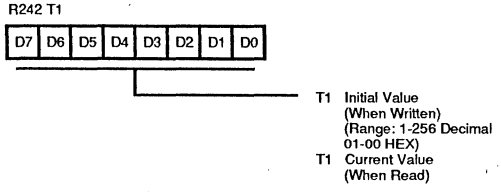


Figure 24. Counter/Timer 1 Register (F2H: Read/Write)

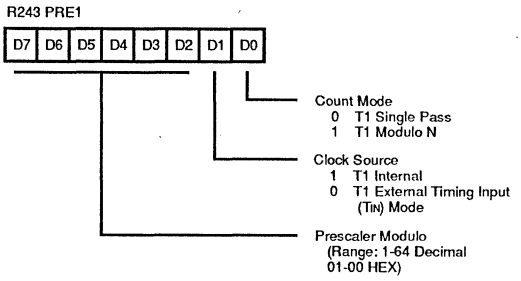


Figure 25. Prescaler 1 Register (F3H: Write Only)

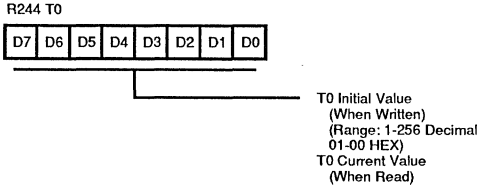


Figure 26. Counter/Timer 0 Register (F4H: Read/Write)

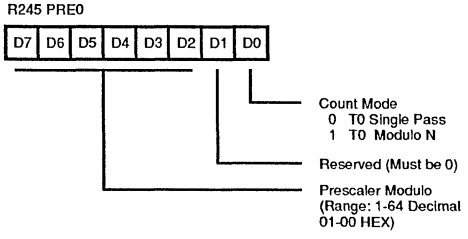


Figure 27. Prescaler 0 Register (F5H: Write Only)

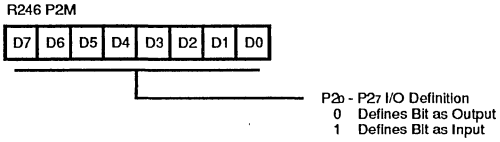


Figure 28. Port 2 Mode Register (F6H: Write Only)

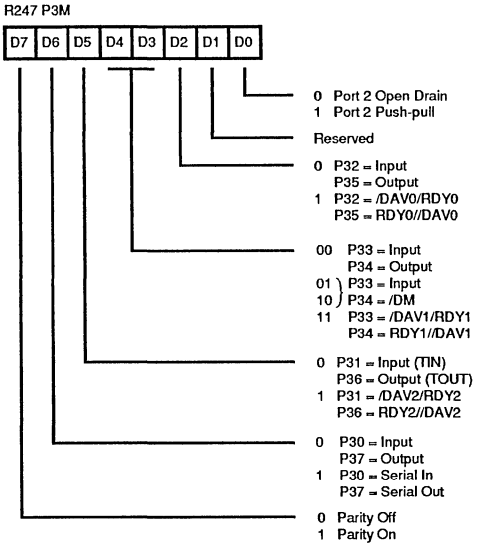


Figure 29. Port 3 Mode Register (F7H: Write Only)

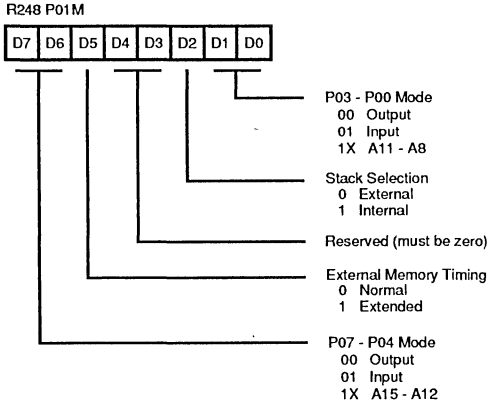


Figure 30. Port 0 and 1 Mode Register (F8H: Write Only)

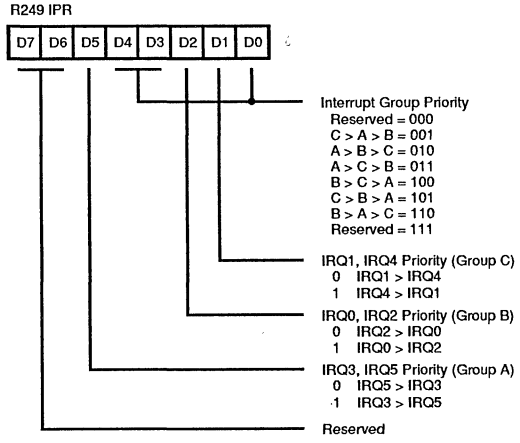


Figure 31. Interrupt Priority Register (F9H: Write Only)

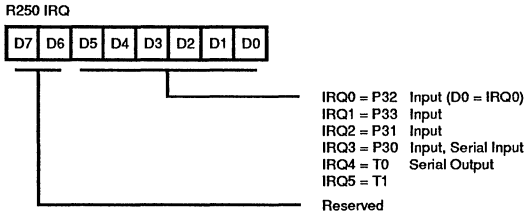


Figure 32. Interrupt Request Register (FAH: Read/Write)

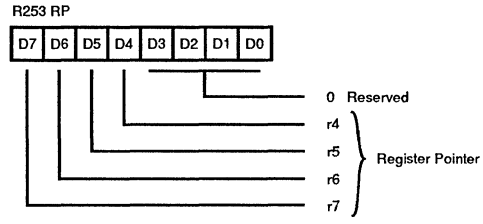


Figure 35. Register Pointer Register (FDH: Read/Write)

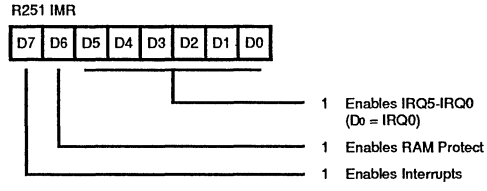


Figure 33. Interrupt Mask Register (FBH: Read/Write)

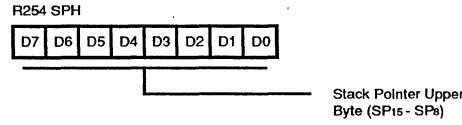


Figure 36. Stack Pointer Register (FEH: Read/Write)

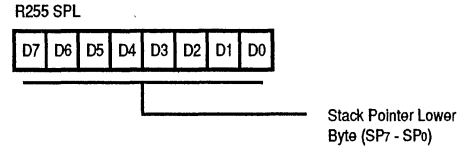


Figure 37. Stack Pointer Register (FFH: Read/Write)

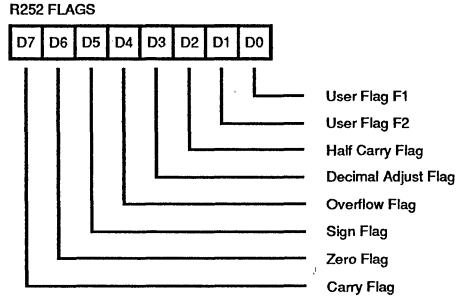


Figure 34. Flag Register (FCH: Read/Write)

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

Symbol	Meaning
IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

Symbol	Meaning
dst	Destination location or contents
src	Source location or contents
cc	Condition code
@	Indirect address prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag register (Control Register 252)
RP	Register Pointer (R253)
IMR	Interrupt mask register (R251)

Flags. Control register (R252) contains the following six flags:

Symbol	Meaning
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

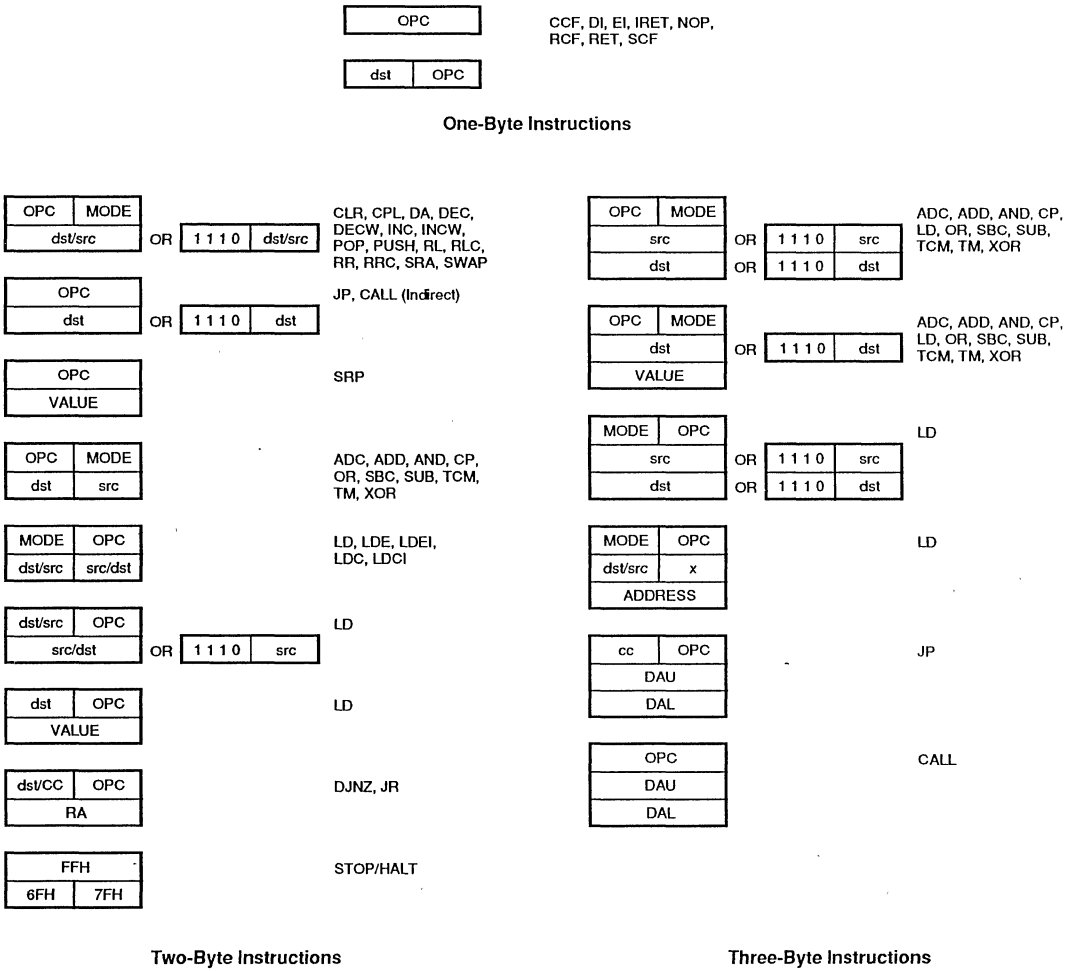
Affected flages are indicated by:

0	Clear to zero
1	Set to one
*	Set to clear according to operation
-	Unaffected
x	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000		Always True	
0111	C	Carry	C = 1
1111	NC	No Carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not Zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No Overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not Equal	Z = 0
1001	GE	Greater Than or Equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater Than	[Z OR (S XOR V)] = 0
0010	LE	Less Than or Equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned Greater Than or Equal	C = 0
0111	ULT	Unsigned Less Than	C = 1
1011	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned Less Than or Equal	(C OR Z) = 1
0000		Never True	

INSTRUCTION FORMATS



INSTRUCTION SUMMARY

Note: Assignment of a value is indicated by the symbol " ← ". For example:

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr (n)" is used to refer to bit (n) of a given operand location. For example:

dst (7)

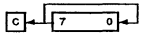
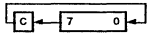
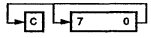
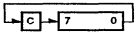
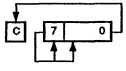
refers to bit 7 of the destination operand.

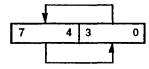
INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst, src dst←dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst←dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst←dst AND src	†		5[ ]	-	*	*	0	-	-
CALL dst SP←SP - 2 @SP←PC, PC←dst	DA IRR		D6 D4	-	-	-	-	-	-
CCF C←NOT C			EF	*	-	-	-	-	-
CLR dst dst←0	R IR		B0 B1	-	-	-	-	-	-
COM dst dst←NOT dst	R IR		60 61	-	*	*	0	-	-
CP dst, src dst - src	†		A[ ]	*	*	*	*	-	-
DA dst dst←DA dst	R IR		40 41	*	*	*	X	-	-
DEC dst dst←dst - 1	R IR		00 01	-	*	*	*	-	-
DECW dst dst←dst - 1	RR IR		80 81	-	*	*	*	-	-
DI IMR(7)←0			8F	-	-	-	-	-	-
DJNZr, dst r←r - 1 if r ≠ 0 PC←PC + dst Range: +127, -128	RA		rA r = 0 - F	-	-	-	-	-	-
EI IMR(7)←1			9F	-	-	-	-	-	-
HALT			7F	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
INC dst dst←dst + 1	r		rE r = 0 - F	-	*	*	*	-	-
	R		20						
	IR		21						
INCW dst dst←dst + 1	RR IR		A0 A1	-	*	*	*	-	-
IRET FLAGS←@SP; SP←SP + 1 PC←@SP; SP←SP + 2; IMR(7)←1			BF	*	*	*	*	*	*
JP cc, dst if cc is true PC←dst	DA IRR		cD c = 0 - F 30	-	-	-	-	-	-
JR cc, dst if cc is true, PC←PC + dst Range: +127, -128	RA		cB c = 0 - F	-	-	-	-	-	-
LD dst, src dst←src	r	Im	rC	-	-	-	-	-	-
	r	R	r8						
	R	r	r9						
			r = 0 - F						
	r	X	C7						
	X	r	D7						
	r	Ir	E3						
	Ir	r	F3						
	R	R	E4						
	R	IR	E5						
	R	IM	E6						
	IR	IM	E7						
	IR	R	F5						
LDC dst, src	r	Irr	C2	-	-	-	-	-	-
LDCI dst, src dst←src r←r +1; rr←rr + 1	Ir	Irr	C3	-	-	-	-	-	-

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
NOP			FF	-	-	-	-	-	-
OR dst, src dst←dst OR src	†		4[ ]	-	*	*	0	-	-
POP dst dst←@SP; SP←SP + 1	R IR		50 51	-	-	-	-	-	-
PUSH src SP←SP - 1; @SP←src		R IR	70 71	-	-	-	-	-	-
RCF C←0			CF	0	-	-	-	-	-
RET PC←@SP; SP←SP + 2			AF	-	-	-	-	-	-
RL dst 	R IR		90 91	*	*	*	*	-	-
RLC dst 	R IR		10 11	*	*	*	*	-	-
RR dst 	R IR		E0 E1	*	*	*	*	-	-
RRC dst 	R IR		C0 C1	*	*	*	*	-	-
SBC dst, src dst←dst←src←C	†		3[ ]	*	*	*	*	1	*
SCF C←1			DF	1	-	-	-	-	-
SRA dst 	R IR		D0 D1	*	*	*	0	-	-
SRP src RP←src		Im	31	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
STOP			6F	-	-	-	-	-	-
SUB dst, src dst←dst←src	†		2[ ]	*	*	*	*	1	*
SWAP dst 	R IR		F0 F1	X	*	*	X	-	-
TCM dst, src (NOT dst) AND src	†		6[ ]	-	*	*	0	-	-
TM dst, src dst AND src	†		7[ ]	-	*	*	0	-	-
XOR dst, src dst←dst XOR src	†		B[ ]	-	*	*	0	-	-

† These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

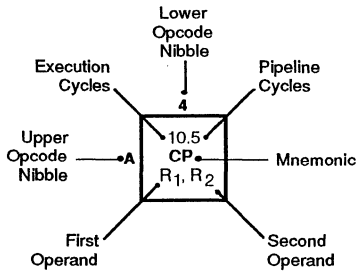
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address Mode dst	src	Lower Opcode Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

OPCODE MAP

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								6.0 **STOP**
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								7.0 **HALT**
8	10.5 **DECW** RR1	10.5 **DECW** IR1	12.0 **LDE** r1, Irr2	18.0 **LDEI** Ir1, Irr2												6.1 **DI**
9	6.5 **RL** R1	6.5 **RL** IR1	12.0 **LDE** r2, Irr1	18.0 **LDEI** Ir2, Irr1												6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14.0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2				10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1	12.0 **LDC** r2, Irr1	18.0 **LDCI** Ir2, Irr1	20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6.5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2			3					2				3		1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_2 = Dst address
R_1 or r_2 = Src address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank are not defined.

* 2-byte instruction appears as a 3-byte instruction

PRELIMINARY PRODUCT SPECIFICATION

Z86C93

CMOS Z8® MULT/DIV MICROCONTROLLER

FEATURES

- Complete microcontroller, 24 I/O lines, and up to 64 Kbytes of addressable external space each for program and data memory.
- 16-bit x 16-bit hardwired multiplier with 32-bit product in 17 clock cycles.
- 32-bit by 16-bit hardwired divider with 16-bit quotient and 16-bit remainder in 20 clock cycles.
- 256-byte register file, including 236 general registers, four I/O port registers and 16 status and control registers.
- 17-byte Expanded Register File, including two general-purpose registers and 15 status and control registers.
- Vectored, priority interrupts for I/O, counter/timers and UART.
- On-chip oscillator that accepts crystal or external clock drive.
- Full-duplex UART and two 16-bit counter timers with 6-bit prescalers.
- Third 16-bit counter/timer with 4-bit prescaler, one capture register and a fast decrement mode.
- Register Pointer so that short, fast instructions can access any one of the sixteen working register groups.
- Additional emulation signals SCLK, IACK, and /SYNC are made available.
- Single +5V power supply, all I/O pins TTL compatible
- 1.2 micron CMOS technology
- Clock speed - 20 MHz
- Two low power standby modes, STOP and HALT

GENERAL DESCRIPTION

The Z86C93 is a CMOS ROMless Z8 microcontroller enhanced with a hardwired 16-bit x 16-bit multiplier and 32-bit/16-bit divider and three 16-bit counter timers (Figure 1). A capture register and a fast decrement mode is also provided. It is fabricated using 1.2 micron CMOS technology. It is offered in 40-pin Plastic Dual-In-Line, 44-pin Leaded Chip Carrier and 44-pin Plastic Quad Flat Pack (Figures 2,3, 4, and 5). Besides the three additional emulation signals (SCLK, IACK, and /SYNC), the Z86C93 is fully pin compatible with Z86C91, yet it offers a much more powerful mathematical capability.

The Z86C93 provides up to 16 output address lines thus permitting an address space of up to 64 K bytes of data and program memory each. Eight address outputs (AD7-AD0) are provided by a multiplexed, 8-bit, Address/Data bus. The remaining 8-bits can be provided by the software configuration of Port 0 to output address bits A15-A8.

There are 256 registers located on-chip organized as 236 general purpose registers, 16 control and status registers, and four I/O port registers. The register file can be divided into sixteen groups of 16 working registers each. Configuration of the registers in this manner allows the use of short format instructions; in addition, any of the individual registers can be accessed directly. There are an additional 17 registers implemented in the Expanded Register File in Banks D and E. Two of the registers may be used as general purpose registers, while 15 registers supply the data and control functions for the Multiply/Divide Unit and Counter/Timer blocks.

Note: All Signals with a preceding front slash, "/", are active Low, e.g.: B//W (WORD is active Low); /B/W (BYTE is active Low, only); /N//S (NORMAL and SYSTEM are both active Low).

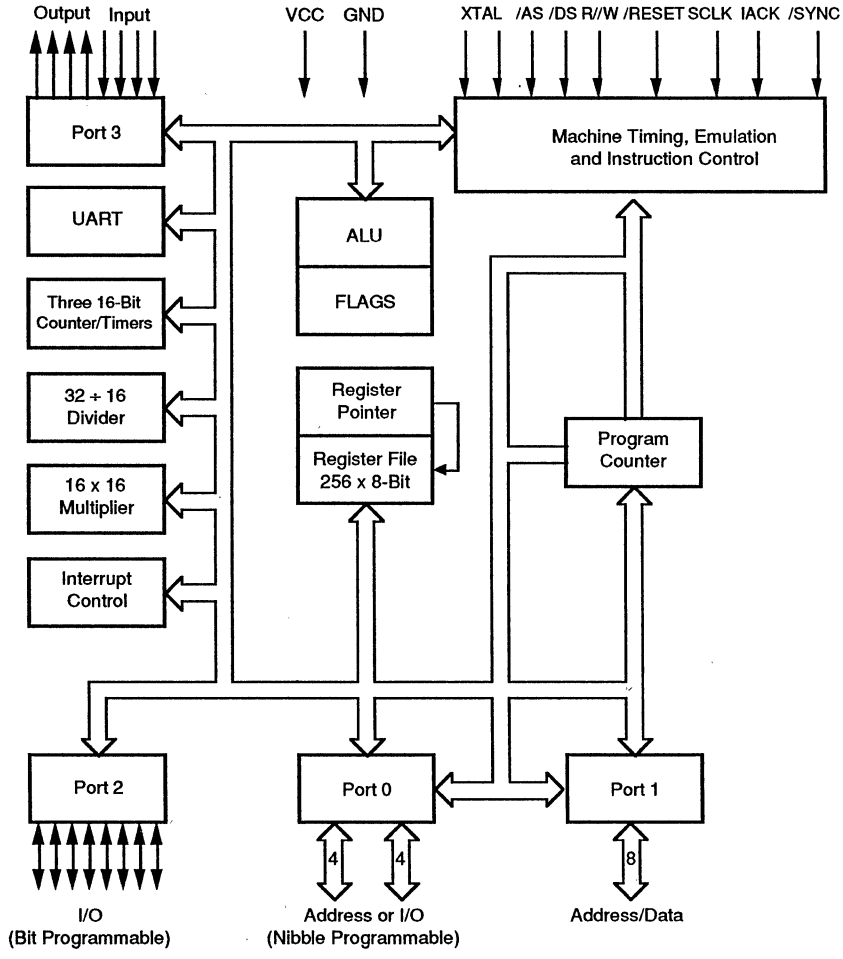


Figure 1. Functional Block Diagram

PIN DESCRIPTION

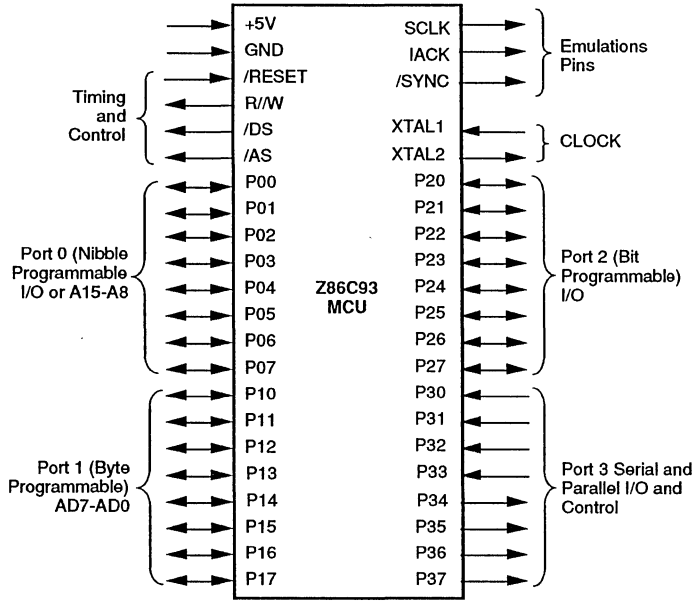


Figure 2. Pin Functions

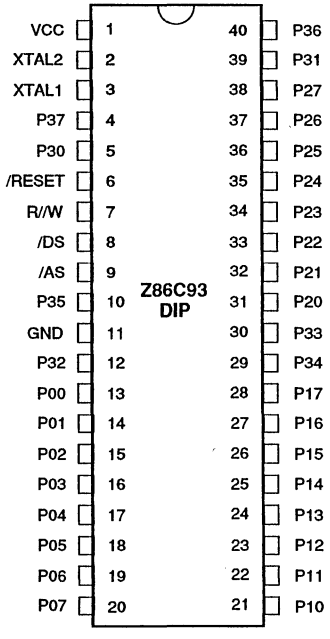


Figure 3. 40-Pin Dual-In-Line Package

Table 1. 40-Pin Dual-In-Line Pin Identification

Pin #	Symbol	Function	Direction
1	V_{CC}	Power Supply	Input
2	XTAL1	Crystal, Oscillator Clock	Input
3	XTAL2	Crystal, Oscillator Clock	Output
4	P37	Port 3 pin 7	Output
5	P30	Port 3 pin 0	Input
6	/RESET	Reset	Input
7	R//W	Read/Write	Output
8	/DS	Data Strobe	Output
9	/AS	Address Strobe	Output
10	P35	Port 3 pin 5	Output
11	GND	Ground, GND	Input
12	P32	Port 3 pin 2	Input
13-20	P00-P07	Port 0 pin 0,1,2,3,4,5,6,7	In/Output
21-28	P10-P17	Port 1 pin 0,1,2,3,4,5,6,7	In/Output
29	P34	Port 3 pin 4	Output
30	P33	Port 3 pin 3	Input
31-38	P20-P27	Port 2 pin 0,1,2,3,4,5,6,7	In/Output
39	P31	Port 3 pin 1	Input
40	P36	Port 3 pin 6	Output

PIN DESCRIPTION (Continued)

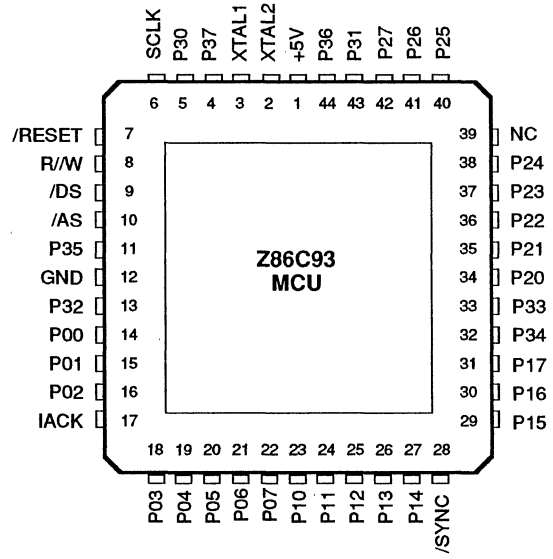


Figure 4. 44-Pin Leaded Chip Carrier

Table 2. 44-Pin Leaded Chip Carrier Pin Identification

No	Symbol	Function	Direction
1	V_{cc}	Power Supply	Input
2	XTAL2	Crystal, Osc. Clock	Output
3	XTAL1	Crystal, Osc. Clock	Input
4	P37	Port 3 pin 7	Output
5	P30	Port 3 pin 0	Input
6	SCLK	System Clock	Output
7	/RESET	Reset	Input
8	R//W	Read/Write	Output
9	/DS	Data Strobe	Output
10	/AS	Address Strobe	Output
11	P35	Port 3 pin 5	Output
12	GND	Ground GND	Input
13	P32	Port 3 pin 2	Input

No	Symbol	Function	Direction
14-16	P00-P02	Port 0 pin 0,1,2	In/Output
17	IACK	Int. Acknowledge	Output
18-22	P03-P07	Port 0 pin 3,4,5,6,7	In/Output
23-27	P10-P14	Port 1 pin 0,1,2,3,4	In/Output
28	/SYNC	Synchronize Pin	Output
29-31	P15-P17	Port 1 pin 5,6,7	In/Output
32	P34	Port 3 pin 4	Output
33	P33	Port 3 pin 3	Input
34-38	P20-P24	Port 2 pin 0,1,2,3,4	In/Output
39	N/C	Not Connected	Input
40-42	P25-P27	Port 2 pin 5,6,7	In/Output
43	P31	Port 3 pin 1	Input
44	P36	Port 3 pin 6	Output

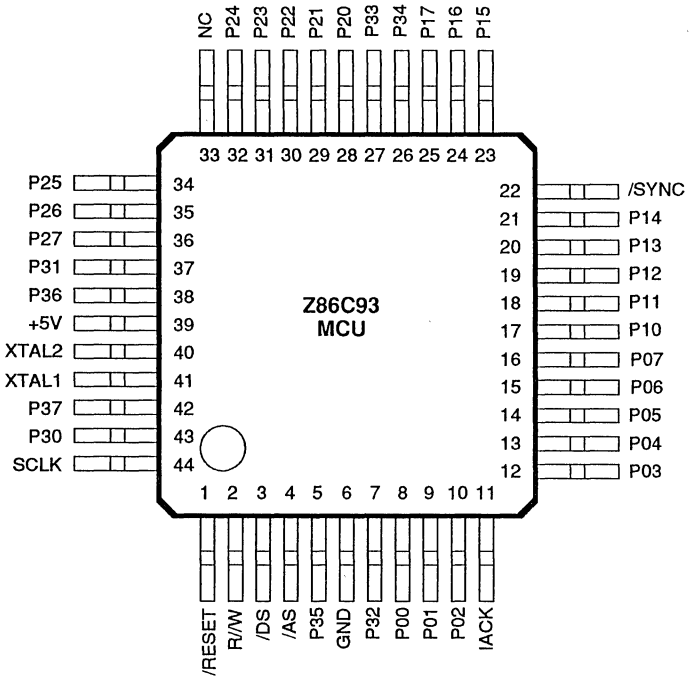


Figure 5. 44-Pin Quad Flat Pack

Table 3. 44-Pin Quad Flat Pack Pin Identification

No	Symbol	Function	Direction
1	/RESET	Reset	Input
2	R/W	Read/Write	Output
3	/DS	Data Strobe	Output
4	/AS	Address Strobe	Output
5	P35	Port 3 pin 5	Input
6	GND	Ground GND	Input
7	P32	Port 3 pin 2	Input
8-10	P00-P02	Port 0 pin 0,1,2	In/Output
11	IACK	Int. Acknowledge	Output
12-16	P03-P07	Port 0 pin 3,4,5,6,7	In/Output
17-21	P10-P14	Port 1 pin 0,1,2,3,4	In/Output
22	/SYNC	Synchronize Pin	Output
23-25	P15-P17	Port 1 pin 5,6,7	In/Output

No	Symbol	Function	Direction
26	P34	Port 3 pin 4	Output
27	P33	Port 3 pin 3	Input
28-32	P20-P24	Port 2 pin 0,1,2,3,4	In/Output
33	N/C	Not Connected	Input
34-36	P25-P27	Port 2 pin 5,6,7	In/Output
37	P31	Port 3 pin 1	Input
38	P36	Port 3 pin 6	Output
39	V_{CC}	Power Supply	Input
40	XTAL2	Crystal, Osc. Clock	Output
41	XTAL1	Crystal, Osc. Clock	Input
42	P37	Port 3 pin 7	Output
43	P30	Port 3 pin 0	Input
44	SCLK	System Clock	Output

PIN FUNCTIONS

/DS. (output, active Low). Data Strobe is activated once for each external memory transfer. For a READ operation, data must be available prior to the trailing edge of /DS. For WRITE operations, the falling edge of /DS indicates that output data is valid.

/AS. (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Address output is via Port 1 for all external programs. When /RESET is asserted, /AS toggles. Memory address transfers are valid at the trailing edge of /AS. Under program control, /AS can be placed in the high-impedance state along with Ports 0 and 1, Data Strobe, and Read//Write.

XTAL1, XTAL2. *Crystal 1, Crystal 2* (time-based input and output, respectively). These pins connect a parallel-resonant crystal, ceramic resonator, LC, or any external single-phase clock to the on-chip oscillator and buffer.

R//W. (output, read High/write Low). The Read/Write signal is low when the MCU is writing to the external program or data memory. It is high when the MCU is reading from the external program or data memory.

/RESET. (input, active-Low). To avoid asynchronous and noisy reset problems, the Z86C93 is equipped with a reset filter of four external clocks (4TpC). If the external /RESET signal is less than 4TpC in duration, no reset occurs.

On the 5th clock after the /RESET is detected, an internal RST signal is latched and held for an internal register count of 18 external clocks, or for the duration of the external /RESET, whichever is longer. During the reset cycle, /DS is held active low while /AS cycles at a rate of TpC/2. When /RESET is deactivated, program execution begins at location 000C (HEX). Reset time must be held low for 50 ms or until VCC is stable, whichever is longer.

SCLK. *System Clock* (output). The internal system clock is available at this pin.

IACK. *Interrupt Acknowledge* (output, active-High). This output, when high, indicates that the Z86C93 is in an interrupt cycle.

/SYNC. (output, active-Low). This signal indicates the last clock cycle of the currently executing instruction.

Port 0 P00-P07. Port 0 is an 8-bit, nibble programmable, bidirectional, TTL compatible port. These eight I/O lines can be configured under software control as a nibble I/O port, or as an address port for interfacing external memory. When used as an I/O port, Port 0 may be placed under handshake control. In this configuration, Port 3, lines P32 and P35 are used as the handshake control /DAV0 and RDY0 (Data Available and Ready). Handshake signal assignment is dictated by the I/O direction of the upper nibble P04-P07. The lower nibble must have the same direction as the upper nibble to be under handshake control. Port 0 comes up as A15-A8 Address lines after /RESET.

For external memory references, Port 0 can provide address bits A11-A8 (lower nibble) or A15-A8 (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. If one or both nibbles are needed for I/O operation, they must be configured by writing to the Port 0 Mode register. After a hardware reset, Port 0 lines are defined as address lines A15-A8, and extended timing is set to accommodate slow memory access. The initialization routine can include reconfiguration to eliminate this extended timing mode (Figure 6). The /OEN (Output Enable) signal in Figure 6 is an internal signal.

Port 1 P10-P17. Port 1 is an 8-bit, byte programmable, bidirectional, TTL compatible port. It has multiplexed Address (A7-A0) and Data (D7-D0) ports.

To interface external memory, Port 1 is programmed for the multiplexed Address/Data mode. If more than 256 external locations are required, Port 0 must output the additional lines (Figure 7).

Port 2 P20-P27. Port 2 is an 8-bit, bit programmable, bidirectional, CMOS compatible port. Each of these eight I/O lines are independently programmed as an input or output, or globally as an open-drain output. Port 2 is always available for I/O operation. When used as an I/O port, Port 2 is placed under handshake control. In this configuration, Port 3 lines P31 and P36 are used as the handshake controls lines /DAV2 and RDY2. The handshake signal assignment for Port 3 lines P31 and P36 is dictated by the direction (input or output) assigned to P27 (Figure 8).

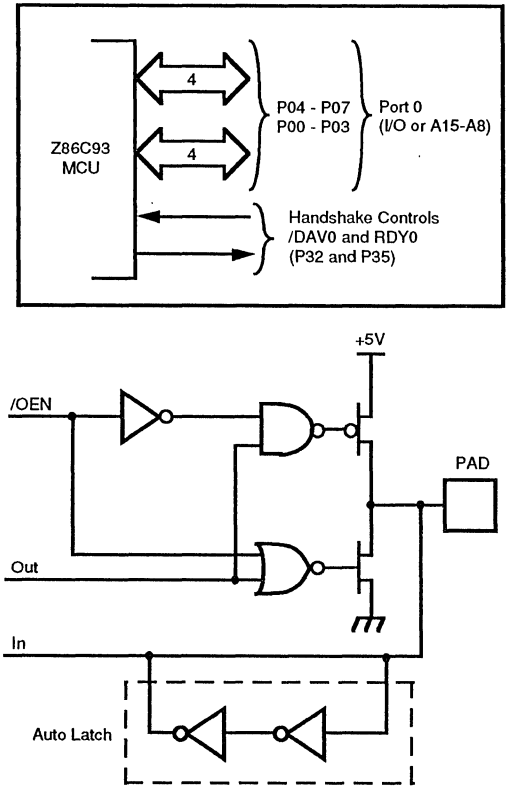


Figure 6. Port 0 Configuration

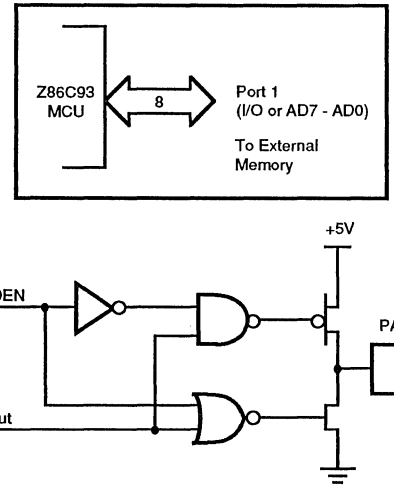


Figure 7. Port 1 Configuration

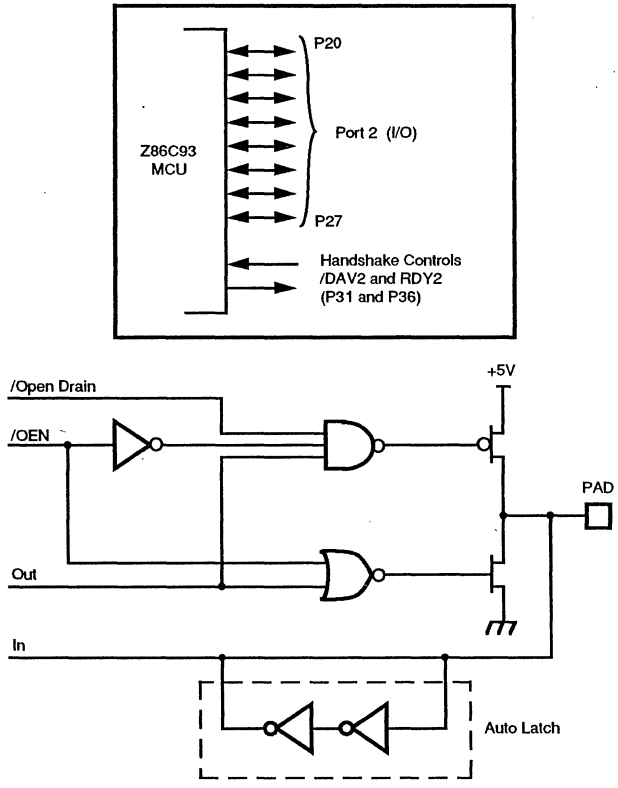


Figure 8. Port 2 Configuration

Port 3 P30-P37. Port 3 is an 8-bit, CMOS compatible four fixed input and four fixed output port. These 8 I/O lines have four-fixed (P30-P33) input and four fixed (P34-P37) output ports. Port 3 pins P30 and P37, when used as serial I/O, are programmed as serial in and serial out, respectively (Figure 9 and Table 4).

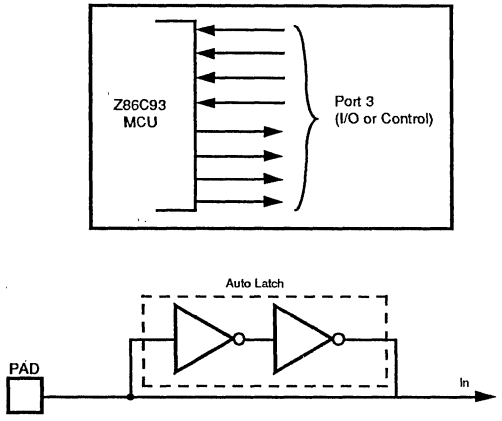


Figure 9. Port 3 Configuration

Table 4. Port 3 Pin Assignments

Pin #	I/O	CTC1	Int.	P0HS	P1HS	P2HS	UART	Ext.
P30	In		IRQ3				Serial In	
P31	In	Tin	IRQ2			D/R		
P32	In		IRQ0	D/R				
P33	In		IRQ1		D/R			
P34	Out				R/D			DM
P35	Out			R/D				
P36	Out	Tout				R/D		
P37	Out						Serial Out	

Port 3 is configured under software control to provide the following control functions: handshake for Ports 0 and 2 (/DAV and RDY); four external interrupt request signals (IRQ0-IRQ3); timer input and output signals (Tin and Tout), and Data Memory Select (/DM).

Port 3 lines P30 and P37, can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by the Counter/Timer 0.

The Z86C93 automatically adds a start bit and two stop bits to transmitted data (Figure 10). Odd parity is also available as an option. Eight data bits are always transmitted, regardless of parity selection. If parity is enabled, the eighth bit is the odd parity bit. An interrupt request (IRQ4) is generated on all transmitted characters.

Received data must have a start bit, 8 data bits and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ3 interrupt request.

Auto-Latch. The auto-latch puts a valid CMOS level on all CMOS inputs that are not externally driven. Whether this level is zero or one, cannot be determined. A valid CMOS level rather than a floating node reduces excessive supply current flow in the input buffer.

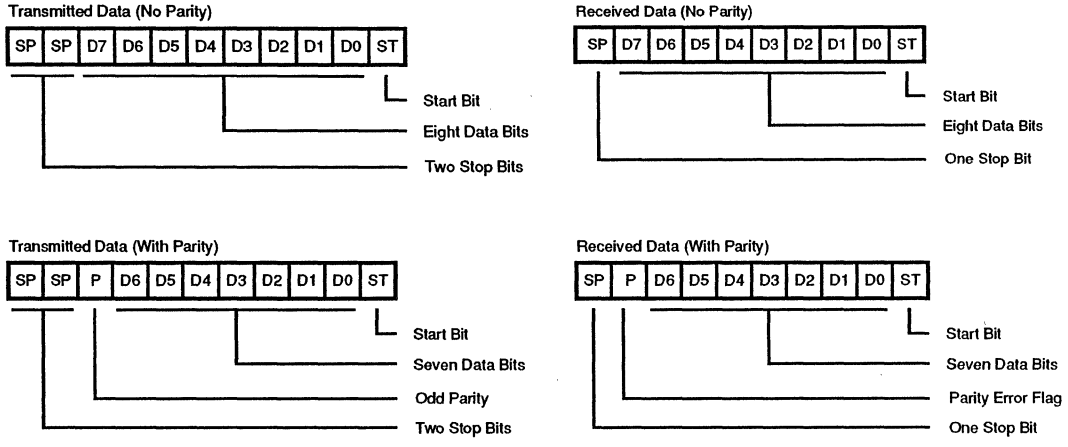


Figure 10. Serial Data Formats

ADDRESS SPACE

Program Memory. The Z86C93 can address up to 64 Kbytes of external program memory. Program execution begins at external location 000C (HEX) after a reset.

Data Memory. The Z96C93 can address up to 64 Kbytes of external data memory. External data memory is included with, or separated from, the external program memory space. /DM, an optional I/O function that can be programmed to appear on pin P34, is used to distinguish between data and program memory space (Figure 11). The state of the /DM signal is controlled by the type instruction being executed. An "LDC" opcode references PROGRAM (/DM inactive) memory, and an "LDE" instruction references DATA (/DM active low) memory.

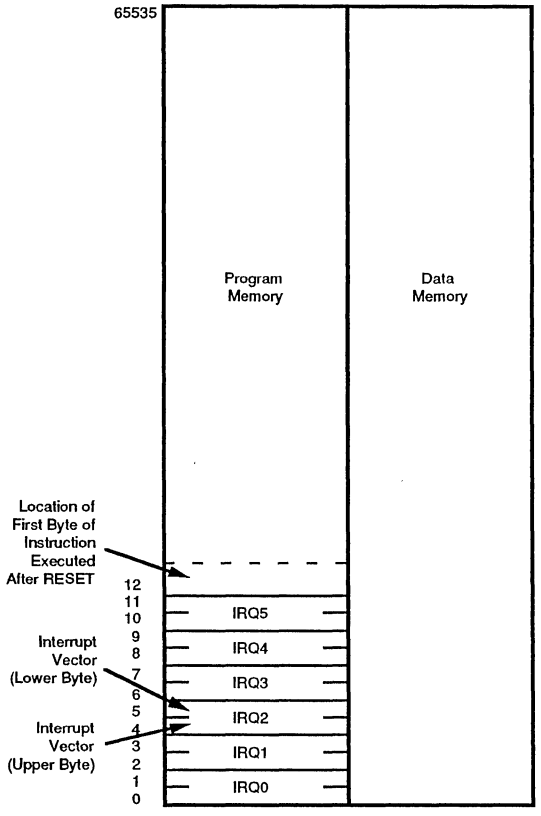


Figure 11. Program and Data Memory Configuration

Expanded Register File. The register file has been expanded to allow for additional system control registers, and for mapping of additional peripheral devices along with I/O ports into the register address area (Figure 12). The Z8 register address space R0 through R15 has now been implemented as 16 groups of 16 registers per group. These register groups are known as the ERF (Expanded Register File). Bits 7-4 of register RP select the working register group. Bits 3-0 of register RP select the expanded register group (Figure 13). The registers that are used in the multiply/divide unit reside in the Expanded Register File at Bank E and those for the additional timer control words reside in Bank D. The rest of the Expanded Register is not physically implemented and is open for future expansion.

Register File. The Register File consists of four I/O port registers, 236 general-purpose registers and 16 control and status registers. The instructions can access registers directly or indirectly via an 8-bit address field. The Z86C93 also allows short 4-bit register addressing using the Register Pointer (Figure 14). In the 4-bit mode, the Register File is divided into 16 working register groups, each occupying 16 continuous locations. The Register Pointer addresses the starting location of the active working-register group.

Note: Register Bank E0-EF can only be accessed through working register and indirect addressing modes.

Stack. The Z86C93 has a 16-bit Stack Pointer (R254-R255), used for external stack, that resides anywhere in the data memory. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 236 general-purpose registers (R4-R239). The high byte of the Stack Pointer (SPH, Bits 8-15) can be used as a general purpose register when using internal stack only.

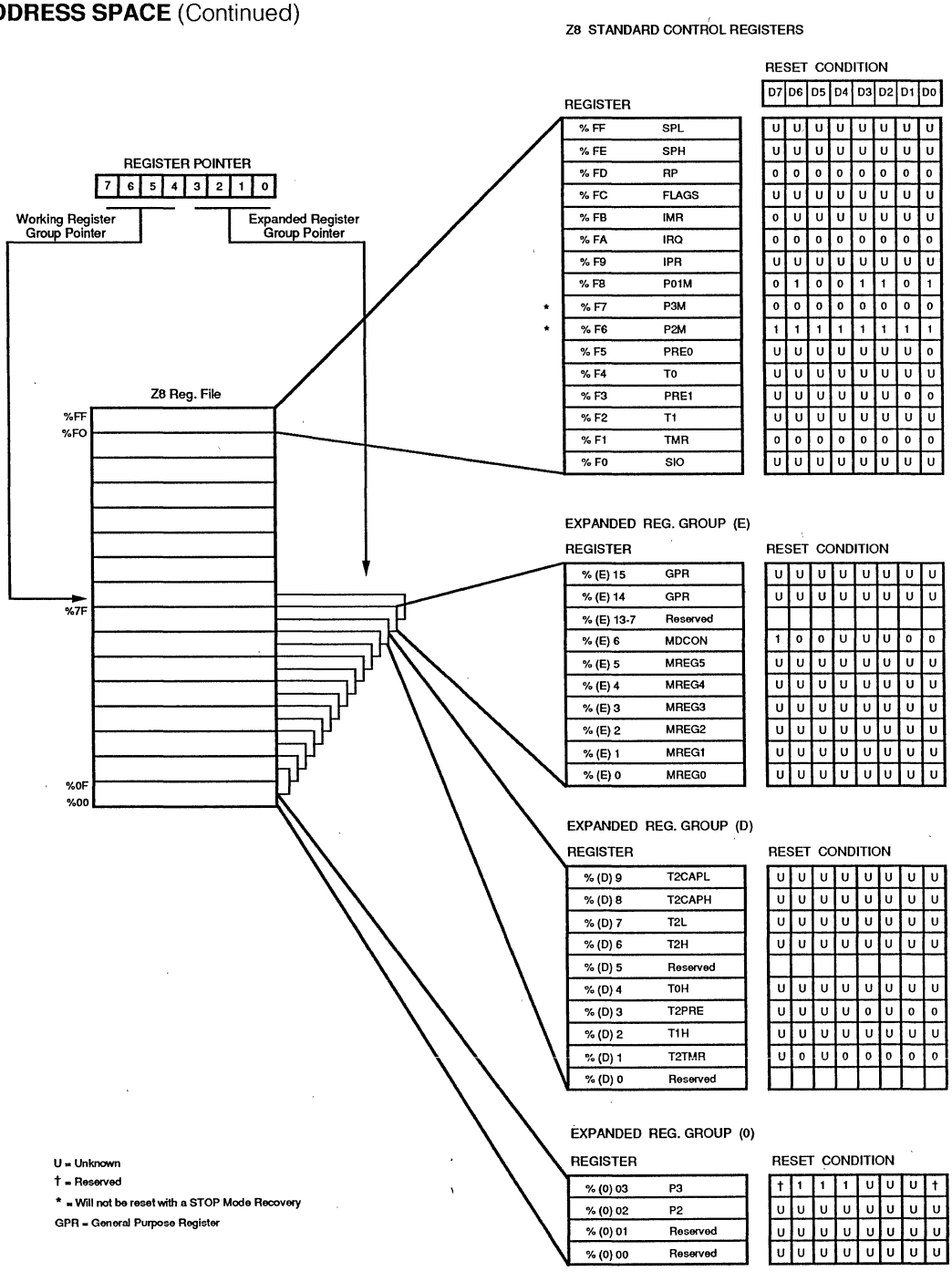


Figure 12. Register File

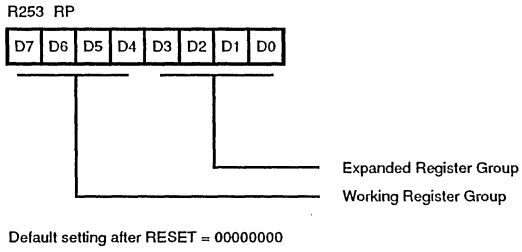


Figure 13. Register Pointer Register

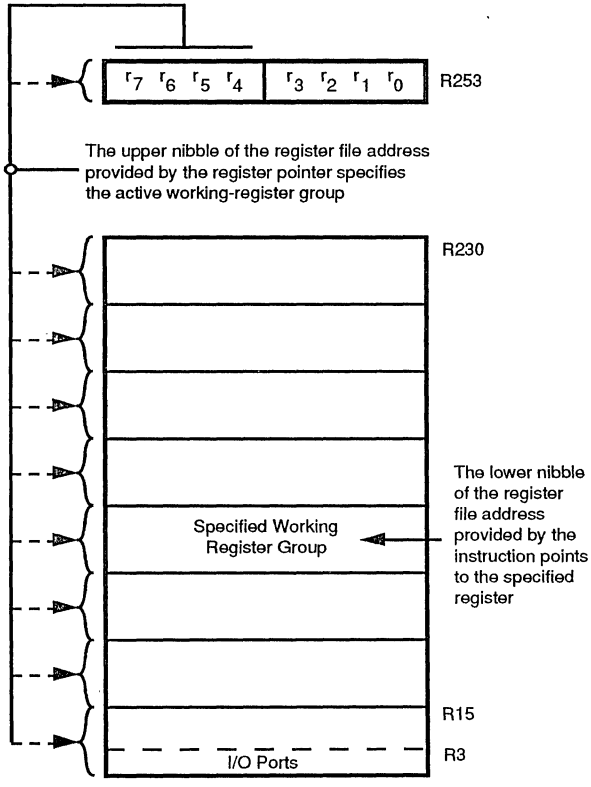


Figure 14. Register Pointer

FUNCTIONAL DESCRIPTION

This section breaks down the Z86C93 into its main functional parts.

Multiply/Divide Unit

This section describes the basic features, implementation details of the interface between the Z8 and the multiply/divide unit.

Basic features:

- 16-bit by 16-bit multiply with 32-bit product
- 32-bit by 16-bit divide with 16-bit quotient and 16-bit remainder
- Unsigned integer data format
- Simple interface to Z8

Interface to Z8. The following is a brief description of the register mapping in the multiply/divide unit and its interface to Z8 (Figure 15).

The multiply/divide unit is interfaced like a peripheral. The only addressing mode available with the peripheral interface is register addressing. In other words, all of the operands are in the respective registers before a multiplication/division can start.

Register mapping. The registers used in the multiply/divide unit are mapped onto the expanded register file in Bank E. The exact register locations used are shown below.

REGISTER	ADDRESS
MREG0	% (E) 00
MREG1	% (E) 01
MREG2	% (E) 02
MREG3	% (E) 03
MREG4	% (E) 04
MREG5	% (E) 05
MDCON	% (E) 06
GPR	% (E) 14
GPR	% (E) 15

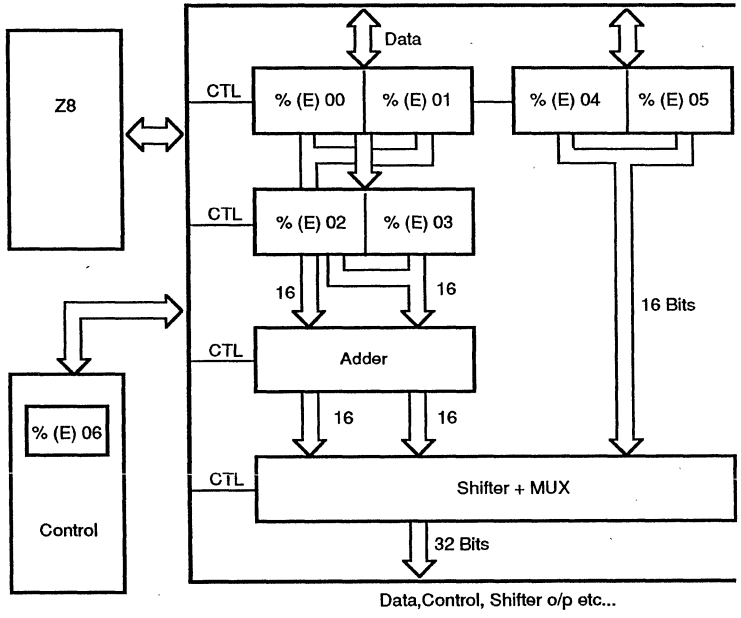


Figure 15. Multiply/Divide Unit Block Diagram

Register allocation. The following is the register allocation during multiplication.

Multiplier high byte	MREG2
Multiplier low byte	MREG3
Multiplicand high byte	MREG4
Multiplicand low byte	MREG5
Result high byte of high word	MREG0
Result low byte of high word	MREG1
Result high byte of low word	MREG2
Result low byte of low word	MREG3
Multiply/Divide Control register	MDCON

The following is the register allocation during division.

High byte of high word of dividend	MREG0
Low byte of high word of dividend	MREG1
High byte of low word of dividend	MREG2
Low byte of low word of dividend	MREG3
High byte of divisor	MREG4
Low byte of divisor	MREG5
High byte of remainder	MREG0
Low byte of remainder	MREG1
High byte of quotient	MREG2
Low byte of quotient	MREG3
Multiply/Divide Control register	MDCON

Control register. The MDCON (Multiply/Divide Control Register) is used to interface with the multiply/divide unit (Figure 16). Specific functions of various bits in the control register are given below.

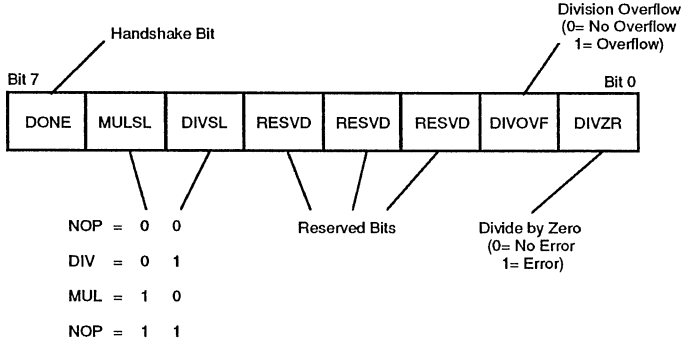


Figure 16. Multiply/Divide Control Register (MDCON)

DONE bit (D7). This bit is a handshake bit between the math unit and the external world. On power up, this bit is set to one to indicate that the math unit has completed the previous operation and is ready to perform the next operation.

Before starting a new multiply/divide operation, this bit should be reset to zero by the processor/programmer. This indicates that all the data registers have been loaded and the math unit can now begin a multiply/divide operation. During the process of multiplication or division, this bit is write-protected. Once the math unit completes its operation it sets this bit to indicate the completion of operation. The processor/programmer can then read the result.

MULSL. *Multiply Select* (D6). If this bit is set to one, it indicates a multiply operation directive. Like the DONE bit, this bit is also write-protected during math unit operation and is reset to zero by the math unit upon starting of the multiply/divide operation.

DIVSL. *Division Select* (D5). Similar to D6, D5 starts a division operation.

D4-D2. Reserved.

DIVOVF. *Division Overflow* (D1). This bit indicated an overflow during the division process. Division overflow occurs when the high word of the dividend is greater than or equal to the divisor. This bit is read only. When set to one, it indicates overflow error.

DIVZR. *Division by Zero* (D0). When set to one, this indicates an error of division by zero. This bit is read only.

Example:

Upon reset, the status of the MDCON register is 100uuu00b (D7 to D0).

u = Undefined
x = Irrelevant
b = Binary

If multiplication operation is desired, the MDCON register is set to 010xxxxxb.

If the MDCON register is READ during multiplication, it would have a value of 000uuu00b.

On completion of multiplication, the result of the MDCON register is 100uuu00b.

If division operation is desired, the MDCON register is set to 001xxxxxb.

During division operation, the register would contain 000uu??b (?- value depends on the DIVIDEND, DIVISOR).

Upon completion of division operation, the MDCON register contains 100uuu??b.

Note that once the multiplication/division operation starts, all data registers (MREG5 thru MREG0) are write-protected and so are the writable bits of the MDCON register. The write protection is released once the math unit operation is complete. However, the registers may be read at any time.

A multiplication sequence would look like:

1. Load multiplier and multiplicand.
2. Load MDCON register to start multiply operation.
3. Wait for the DONE bit of the MDCON register to be set to ONE and then read results.

Note that while the multiply/divide operation is in progress, the programmer can use the Z8 to do other things. Also, since the multiplication/division takes a fixed number of cycles, he can start reading the results before the DONE bit is set.

During a division operation, the error flag bits are set at the beginning of the division operation which means the flag bits can be checked by the Z8 while the division operation is being done.

The two General Purpose Registers (GPR) can be used as scratch pad registers or as external data memory address pointers during an LDE instruction. MREG0 thru MREG5, if not used for multiplication or division, can be used as general purpose registers.

Performance of multiplication. The actual multiplication takes 17 internal clock cycles. It is expected that the chip would run at a 10 MHz internal clock frequency (external clock divided by two). This results in an actual multiplication time (16-bit x 16-bit) of 1.7 us. If the time to load operands and read results is included:

No. of internal clock cycles to load 5 registers: 30
No. of internal clock cycles to read 4 registers: 24

The total internal clock cycles to perform a multiplication is 71. This results in a net multiplication time of 7.1 us. Note that this would be the worst case. This assumes that all of the operands are loaded from the external world as opposed to some of the operands being already in place as a result of a previous operation whose destination register is one of the math unit registers.

Performance of division. The actual division needs 20 internal clock cycles. This translates to 2.0 us for the actual division at 10 Mhz (internal clock speed). If the time to load operands and read results is included:

Number of internal clock cycles to load operands: 42
Number of internal clock cycles to read results: 24

The total internal clock cycles to perform a division is 86. This translates to 8.6 us at 10 Mhz.

Counter/Timers

This section describes the enhanced features of the counter/timers (CTC) on the Z86C93. It contains the register mapping of CTC registers and the bit functions of the newly added Timer2 control register.

In a standard Z8, there are two 8-bit programmable counter/timers (T0 and T1), each driven by its own 6-bit programmable prescaler. The T1 prescaler is driven by internal or external clock sources; however, the T0 prescaler is driven by the internal clock only.

The 6-bit prescalers divide the input frequency of the clock source by any integer number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of the count, a timer interrupt request IRQ4 (T0) or IRQ5 (T1), is generated.

The counters are programmed to start, stop, restart to continue, or restart from the initial value. The counters can also be programmed to stop upon reaching zero (single pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode).

The counters, but not the prescalers, are read at any time without disturbing their value or count mode. The clock source for T1 is user-definable and is either the internal microprocessor clock divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input (P31) as an external clock, a trigger input that is retriggerable or non-retriggerable, or as a gate input for the internal clock. The counter/timers are cascaded by connecting the T0 output to the input of T1. Either T0 or T1 can be outputted via P36.

The following are the enhancements made to the counter/timer block on the Z86C93 (Figure 17):

- T0 counter length is extended to 16-bits. For example, T0 now has a 6-bit prescaler and 16-bit down counter.
- T1 counter length is extended to 16-bits. For example, T1 now has a 6-bit prescaler and 16-bit down counter.
- A new counter/timer T2 is added. T2 has a 4-bit prescaler and a 16-bit down counter with capture register.

These three counters are cascadable as shown in Table 5. The result is that T2 may be extendable to 32 bits and T1 extendable to 24 bits. Bits 1 and 0 (CAS1 AND CAS0) of the T2 Prescaler Register (PRE2) determine the counter length.

Table 5. Counter Length Configurations

CAS 1	CAS0	T0	T1	T2
0	0	8	8	32
0	1	16	16	16
1	0	8	24	16
1	1	8	16	24

The controlling clock input to T2 is programmed to XTAL/2 or XTAL/8 (only when T2 counter length is 16 bits), which results in a resolution of 100 ns at an external XTAL clock speed of 20 Mhz.

T2 has a 16-bit capture register associated with T2 HIGH BYTE and T2 LOW BYTE registers. The negative going transition on pin P33 enables the latching of the current T2 value (16 bits) into the capture register. The register mapping of the capture register is in Bank D (Figure 12). Note that the negative transition on P33 is capable of generating an interrupt. Also, the negative transition on P33 always latches the current T2 value into the capture register. There is no need for a control bit to enable/disable the latching; the capture register is read only.

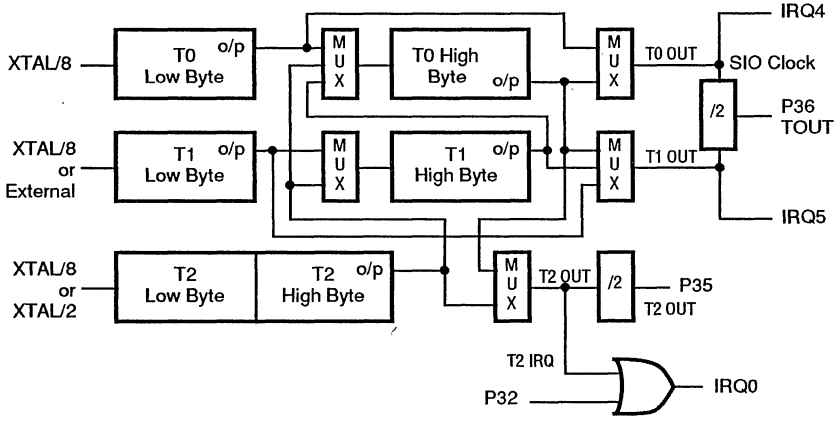


Figure 17. Counter/Timer Block Diagram

Observations

Except for the programmable down counter length and clock input, T2 is identical to T0.

T0 and T1 retain all their features except that now they are extendable interims of the down counter length.

The output of T2, under program control, goes to an output pin (P35). Also, the interrupt generated by T2 is ORed with the interrupt request generated by P32. Note that the service routine then has to poll the T2 flag bit and also clear it (Bit 7 of T2 Timer Mode Register).

On power up, T0 and T1 are configured in the 8-bit down counter length mode (to be compatible with Z86C91) and T2 is in the 32-bit mode with its output disabled (no interrupt is generated and T2 output does NOT go to port pin P35).

The UART uses T0 for generating the bit clock. This means, while using UART, T0 should be in 8-bit mode. So, while using the UART there are only two independent timer/counters.

The counters are configured in the following manner:

Timer	Mode	Byte
T0	8-bit	low byte (T0)
T0	16-bit	high byte (T0) + low byte (T0)
T1	8-bit	low byte (T1)
T1	16-bit	high byte (T1)+ low byte (T1)
T1	24-bit	high byte (T0) + high byte (T1) + low byte (T1)
T2	16-bit	high byte (T2) + low byte (T2)
T2	24-bit	high byte (T0) + high byte (T2) + low byte (T2)
T2	32-bit	high byte (T0) + high byte (T1) + high byte (T2) + low byte (T2)

Note that the T2 interrupt is logically ORed with P32 to generate IRQ0.

The T2 Timer Mode register is shown in Figure 18. Upon reaching end of count, bit 7 of this register is set to one. This bit is NOT reset in hardware and it has to be cleared by the interrupt service routine.

The register map of the new CTC registers is shown in Figure 12. T0 high byte and T1 high byte are at the same relative locations as their respective low bytes, but in a different register bank.

The T2 prescaler register is shown in Figure 19. Bits 1 and 0 of this register control the various cascade modes of the counters.

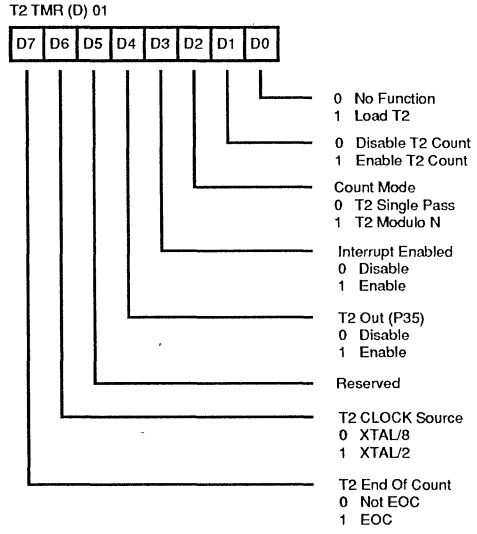


Figure 18. T2 Timer Mode Register (T2)

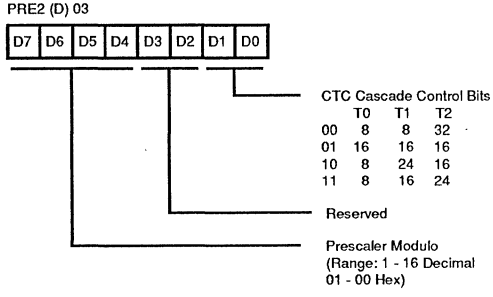


Figure 19. T2 Prescaler Register (PRE2)

Interrupts

The Z86C93 has six different interrupts from eight different sources. The interrupts are maskable and prioritized. The eight sources are divided as follow: four sources are claimed by Port 3 lines P30-P33, one in Serial Out, one in

Serial In, and two in the counter/timers. The Interrupt Mask Register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register. All Z86C93 interrupts are vectored through locations in the program memory. When an interrupt machine cycle is activated an interrupt request is granted. Thus, this disables all of the subsequent interrupts, save the Program Counter and Status Flags, and then branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

To accommodate polled interrupt systems, interrupt inputs are masked and the Interrupt Request register is polled to determine which of the interrupt requests need service. Software initiated interrupts are supported by setting the appropriate bit in the Interrupt Request Register (IRQ).

Internal interrupt requests are sampled on the falling edge of the last cycle of every instruction, and the interrupt request must be valid 5TpC before the falling edge of the last clock cycle of the currently executing instruction.

When the device samples a valid interrupt request, the next 48 (external) clock cycles are used to prioritize the interrupt, and push the two PC bytes and the FLAG register on the stack. The following nine cycles are used to fetch the interrupt vector from external memory. The first byte of the interrupt service routine is fetched beginning on the 58th TpC cycle following the internal sample point, which corresponds to the 63rd TpC cycle following the external interrupt sample point.

Clock

The Z86C93 on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal, LC, ceramic resonator, or any suitable external clock source (XTAL1=Input, XTAL2=Output). The external clock levels are not TTL. The crystal should be AT cut, 1 MHz to 20 MHz max, and series resistance (RS) is less than or equal to 100 Ohms. The crystal should be connected across XTAL1 and XTAL2 using the recommended capacitors (10 pF<CL<100 pF) from each pin to ground (Figure 20).

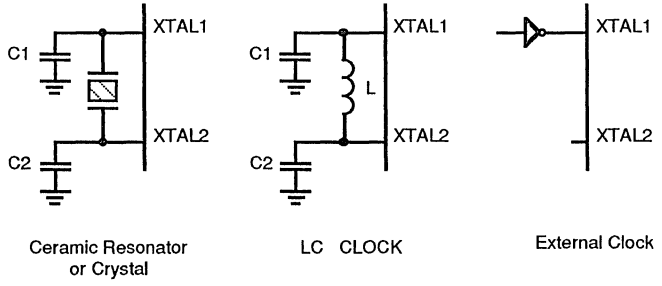


Figure 20. Oscillator Configuration

Power Down Modes

Halt. Turns off the internal CPU clock but not the XTAL oscillation. The counter/timers and the external interrupts IRQ0, IRQ1, IRQ2 and IRQ3 remain active. The devices are recovered by interrupts, either externally or internally generated.

STOP. This instruction turns off the internal clock and external crystal oscillation and reduces the standby current to 10 microamperes or less. The Stop mode is terminated by a /RESET, which causes the processor to restart the application program at address 000C (HEX).

In order to enter STOP (or HALT) mode, it is necessary to first flush the instruction pipeline to avoid suspending execution in mid-instruction. To do this, the user executes a NOP (opcode=OFFH) immediately before the appropriate sleep instruction, i.e.:

```
FF  NOP   ; clear the pipeline
6F  STOP  ; enter STOP mode
          or
FF  NOP   ; clear the pipeline
7F  HALT  ; enter HALT mode
```

ABSOLUTE MAXIMUM RATINGS

Symbol	Description	Min	Max	Units
V_{CC}	Supply Voltage*	-0.3	+7.0	V
T_{STG}	Storage Temp	-65	+150	C
T_A	Oper Ambient Temp		†	C

* Voltages on all pins with respect to GND.
† See Ordering Information

Stress greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for an extended period may affect device reliability.

STANDARD TEST CONDITIONS

The characteristics listed below apply for standard test conditions as noted. All voltages are referenced to GND. Positive current flows into the referenced pin (Figure 21).

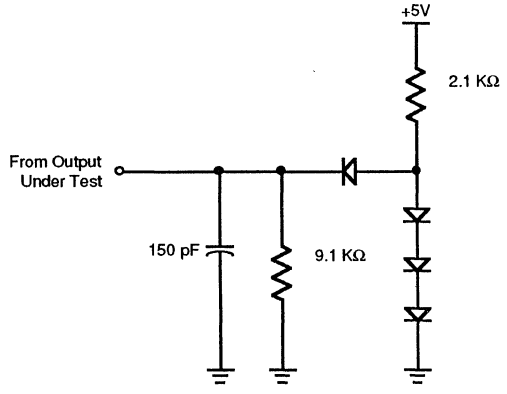


Figure 21. Test Load Diagram

DC ELECTRICAL CHARACTERISTICS

Sym	Parameter	T_A = 0°C to 70°C Min	Max	T_A = -40°C to 105°C Min	Max	Typical at 25°C	Units	Conditions
	Max Input Voltage		7		7		V	I_{IN} 250 µA
V_{CH}	Clock Input High Voltage	3.8	V_{CC}	3.8	V_{CC}		V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.03	0.8	-0.03	0.8		V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}	2.0	V_{CC}		V	
V_{IL}	Input Low Voltage	-0.3	0.8	-0.3	0.8		V	
V_{OH}	Output High Voltge	2.4		2.4			V	I_{OH} = -2.0 mA
V_{OH}	Output High Voltge	V_{CC} - 100mV		V_{CC} - 100mV			V	I_{OH} = -100 µA
V_{OL}	Output Low Voltage		0.4		0.4		V	I_{OH} = +2.0 mA
V_{RH}	Reset Input High Voltage	3.8	V_{CC}	3.8	V_{CC}		V	
V_{RI}	Reset Input Low Voltage	-0.03	0.8	-0.03	0.8		V	
I_{IL}	Input Leakage	-2	2	-2	2		µA	Test at 0V, V_{CC}
I_{OL}	Output Leakage	-2	2	-2	2		µA	Test at 0V, V_{CC}
I_{IR}	Reset Input Current		-80		-80		µA	V_{RL} = 0V
I_{CC}	Supply Current		25		25	16	mA	@ 16 MHz [1]
			30		30	20	mA	@ 20 MHz [1]
I_{CC1}	Standby Current		10		10	8	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 16 MHz [1]
			12		12	9	mA	HALT Mode V_{IN} = 0V, V_{CC} @ 20 MHz [1]
I_{CC2}	Standby Current		10		20	1	µA	STOP Mode V_{IN} = 0V, V_{CC} [1]
I_{ALL}	Auto Latch Low Current	-10	10	-14	14	5	µA	

Note:
[1] All inputs driven to 0V, V_{cc} and outputs floating.

AC CHARACTERISTICS

External I/O or Memory Read/Write Timing Diagram

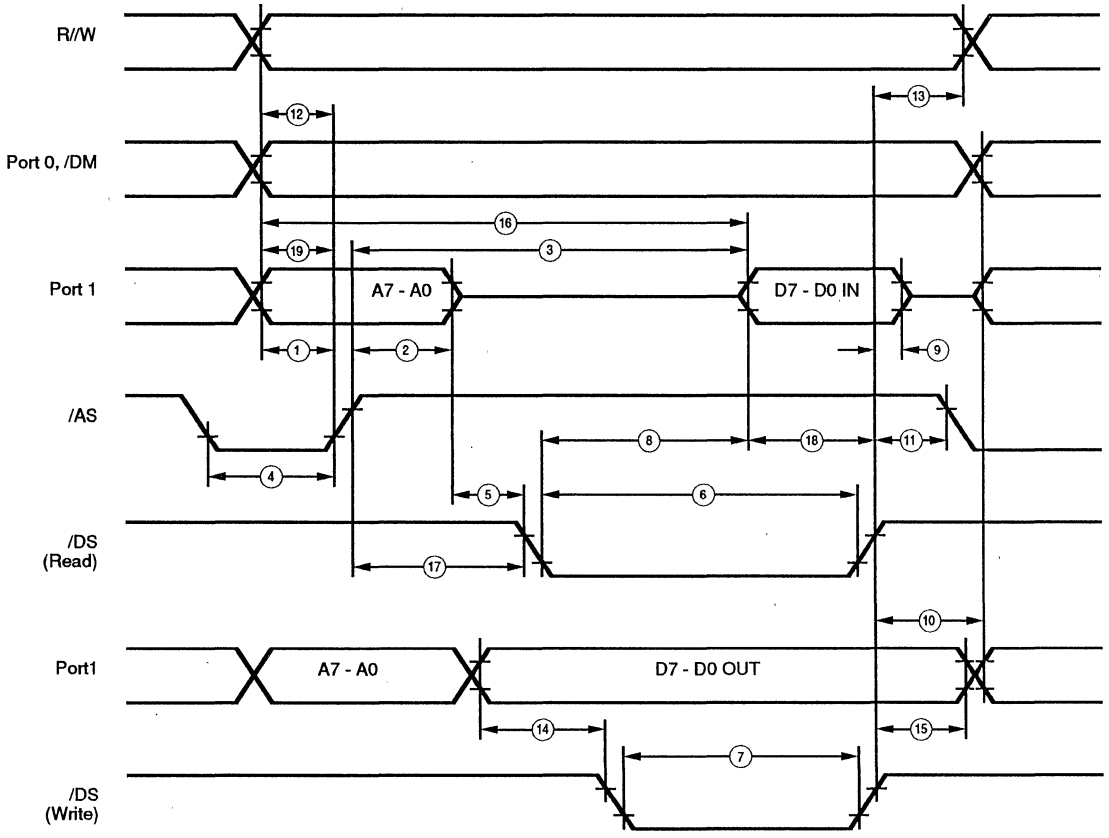


Figure 22. External I/O or Memory Read/Write Timing

AC CHARACTERISTICS

External I/O or Memory Read and Write Timing Table

No	Symbol	Parameter	T_A = 0°C to 70°C				T_A = -40°C to 105°C				Units	Notes
			16 MHz		20 MHz		16 MHz		20 MHz			
			Min	Max	Min	Max	Min	Max	Min	Max		
1	TdA(AS)	Address Valid to /AS Rise Delay	25		20		25		20		ns	[2,3]
2	TdAS(A)	/AS Rise to Address Float Delay	35		25		35		25		ns	[2,3]
3	TdAS(DR)	/AS Rise to Read Data Req'd Valid		180		150		180		150	ns	[1,2,3]
4	TwAS	/AS Low Width	40		30		40		30		ns	[2,3]
5	TdAZ(DS)	Address Float to /DS Fall	0		0		0		0		ns	
6	TwDSR	/DS (Read) Low Width	135		105		135		105		ns	[1,2,3]
7	TwDSW	/DS (Write) Low Width	80		65		80		65		ns	[1,2,3]
8	TdDSR(DR)	/DS Fall to Read Data Req'd Valid		75		55		75		55	ns	[1,2,3]
9	ThDR(DS)	Read Data to /DS Rise Hold Time	0		0		0		0		ns	[2,3]
10	TdDS(A)	/DS Rise to Address Active Delay	50		40		50		40		ns	[2,3]
11	TdDS(AS)	/DS Rise to /AS Fall Delay	35		25		35		25		ns	[2,3]
12	TdR/W(AS)	R/W Valid to /AS Rise Delay	25		20		25		20		ns	[2,3]
13	TdDS(R/W)	/DS Rise to R/W Not Valid	35		25		35		25		ns	[2,3]
14	TdDW(DSW)	Write Data Valid to /DS Fall (Write) Delay	25		20		25		20		ns	[2,3]
15	TdDS(DW)	/DS Rise to Write Data Not Valid Delay	35		25		35		25		ns	[2,3]
16	TdA(DR)	Address Valid to Read Data Req'd Valid		230		180		230		180	ns	[1,2,3]
17	TdAS(DS)	/AS Rise to /DS Fall Delay	45		35		45		35		ns	[2,3]
18	TdDI(DS)	Data Input Setup to /DS Rise	60		50		60		50		ns	[1,2,3]
19	TdDM(AS)	/DM Valid to /AS Rise Delay	30		20		30		20		ns	[2,3]

Notes:
[1] When using extended memory timing add 2 TpC.
[2] Timing numbers given are for minimum TpC.
[3] See clock cycle dependent characteristics table 5.

Standard Test Load
All timing references use 2.0V for a logic 1 and 0.8V for a logic 0.

Clock Dependent Equations

No	Symbol	Equation
1	TdA(AS)	0.40TpC + 0.32
2	TdAS(A)	0.59TpC - 3.25
3	TdAS(DR)	2.38TpC + 6.14
4	TwAS	0.66TpC - 1.65
6	TwDSR	2.33TpC - 10.56
7	TwDSW	1.27TpC + 1.67
8	TdDSR(DR)	1.97TpC - 42.5
10	TdDS(A)	0.8TpC
11	TdDS(AS)	0.59TpC - 3.14

No	Symbol	Equation
12	TdR/W(AS)	0.4TpC
13	TdDS(R/W)	0.8TpC - 15
14	TdDW(DSW)	0.4TpC
15	TdDS(DW)	0.88TpC - 19
16	TdA(DR)	4TpC - 20
17	TdAS(DS)	0.91TpC - 10.7
18	TsDI(DS)	0.8TpC - 10
19	TdDM(AS)	0.9TpC - 26.3

Note:
Equation units are in nanoseconds

AC CHARACTERISTICS
Additional Timing Diagram

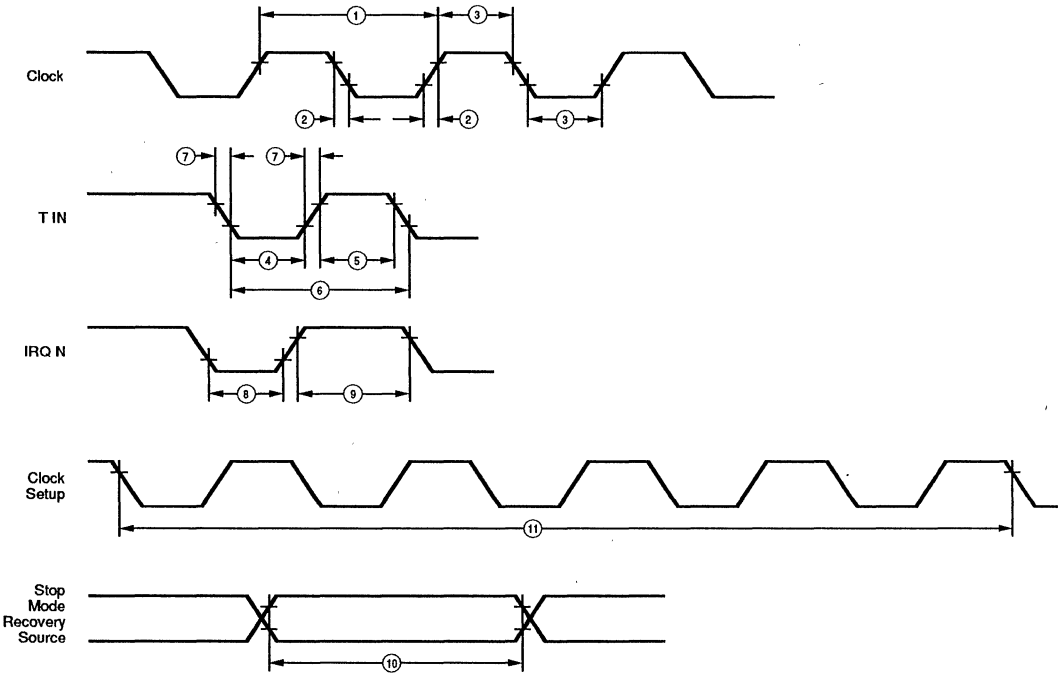


Figure 23. Additional Timing

AC CHARACTERISTICS
Additional Timing Table

No	Symbol	Parameter	T_A = 0°C to 70°C				T_A = -40°C to 105°C				Units	Notes
			16 MHz		20 MHz		16 MHz		20 MHz			
			Min	Max	Min	Max	Min	Max	Min	Max		
1	TpC	Input Clock Period	62.5	1000	50	1000	62.5	1000	50	1000	ns	[1]
2	TrC,TfC	Clock Imput Rise & Fall Times		10		10		10		10	ns	[1]
3	TwC	Input Clock Width	25		15		25		15		ns	[1]
4	TwTinL	Timer Input Low Width	75		75		75		75		ns	[2]
5	TwTinH	Timer Input High Width	3TpC		3TpC		3TpC		3TpC			[2]
6	TpTin	Timer Input Period	8TpC		8TpC		8TpC		8TpC			[2]
7	TrTin,TfTin	Timer Input Rise & Fall Times	100		100		100		100		ns	[2]
8A	TwIL	Interrupt Request Input Low Times	70		70		70		70		ns	[2,4]
8B	TwIL	Interrupt Request Input Low Times	5TpC		5TpC		5TpC		5TpC			[2,5]
9	TwIH	Interrupt Request Input High Times	3TpC		3TpC		3TpC		3TpC			[2,3]

Notes:
[1] Clock timing references use 3.8V for a logic 1 and 0.8V for a logic 0.
[2] Timing references use 2.0V for a logic 1 and 0.8V for a logic 0.
[3] Interrupt references request via Port 3.
[4] Interrupt request via Port 3 (P31-P33).
[5] Interrupt request via Port 30.

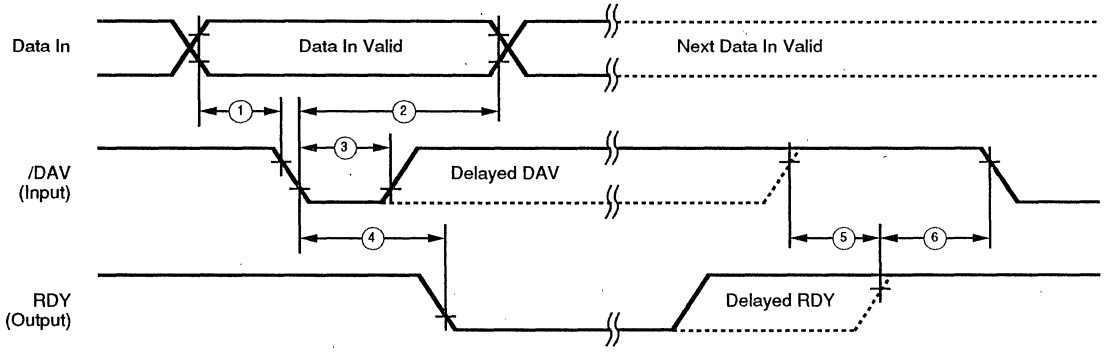


Figure 24. Input Handshake Timing

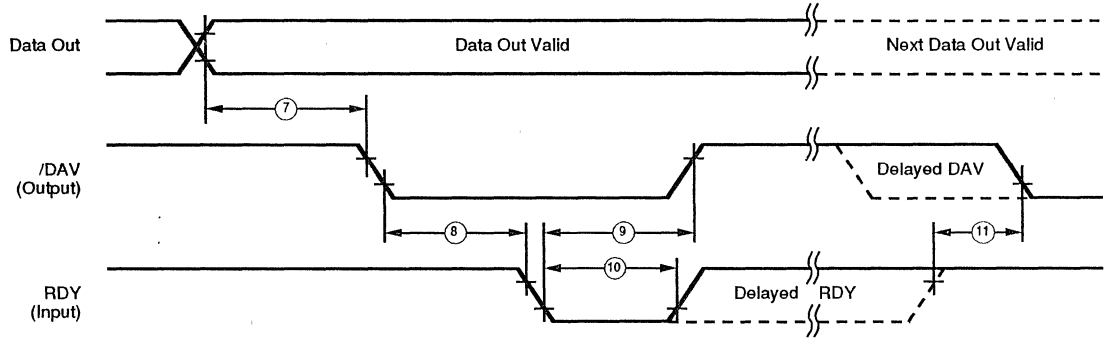


Figure 25. Output Handshake Timing

AC CHARACTERISTICS

Handshake Timing Table

No	Symbol	Parameter	T_A = 0°C to 70°C				T_A = -40°C to 105°C				Units	Notes
			16 MHz		20 MHz		16 MHz		20 MHz			
			Min	Max	Min	Max	Min	Max	Min	Max		
1	TsDI(DAV)	Data In Setup Time	0		0		0		0		ns	IN
2	ThDI(DAV)	Data In Hold Time	145		145		145		145		ns	IN
3	TwDAV	Data Available Width	110		110		110		110		ns	IN
4	TdDAVI(RDY)	DAV Fall to RDY Fall Delay		115		115		115		115	ns	IN
5	TdDAVId(RDY)	DAV Rise to RDY Rise Delay		115		115		115		115	ns	IN
6	TdDO(DAV)	RDY Rise to DAV Fall Delay	0		0		0		0		ns	IN
7	TcLDAV0(RDY)	Data Out to DAV Fall Delay		TpC		TpC		TpC		TpC		OUT
8	TcLDAV0(RDY)	DAV Fall to RDY Fall Delay	0		0		0		0		ns	OUT
9	TdRDY0(DAV)	RDY Fall to DAV Rise Delay		115		115		115		115	ns	OUT
10	TwRDY	RDY Width	110		110		110		110		ns	OUT
11	TdRDY0d(DAV)	RDY Rise to DAV Fall Delay		115		115		115		115	ns	OUT

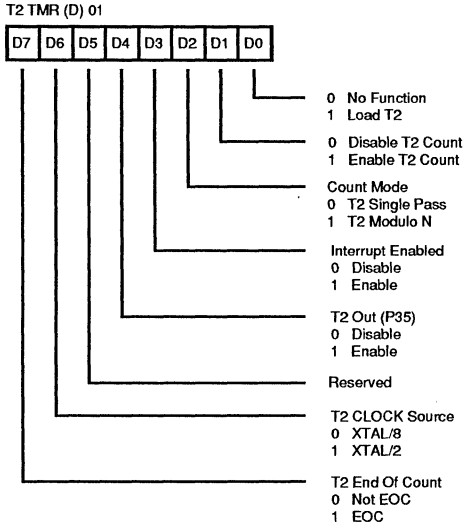


Figure 26. Timer 2 Mode Register [%(D) 01: Read/Write]

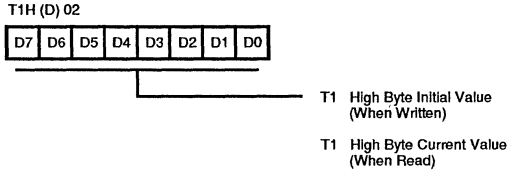


Figure 27. Counter Timer 1 Register High Byte [%(D) 02: Read/Write]

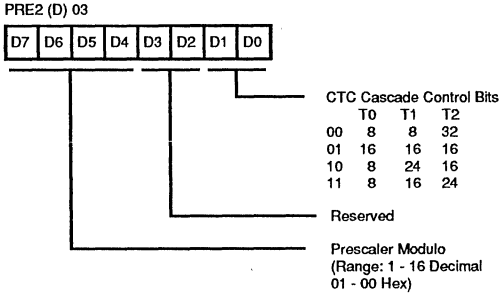


Figure 28. Prescaler 2 Register High Byte [%(D) 03: Write Only]

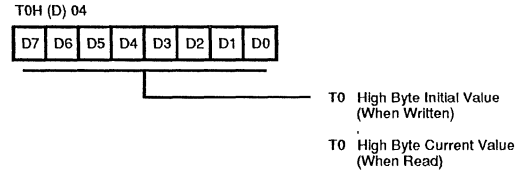


Figure 29. Counter Timer 0 Register High Byte [%(D) 04: Read/Write]

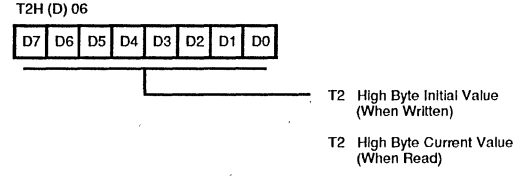


Figure 30. Counter Timer 2 Register High Byte [%(D) 06: Read/Write]

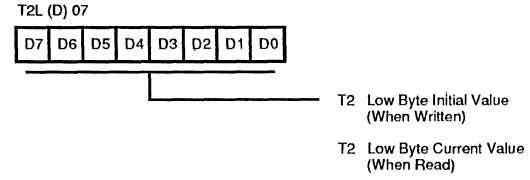


Figure 31. Counter Timer 2 Register Low Byte [%(D) 07: Read/Write]

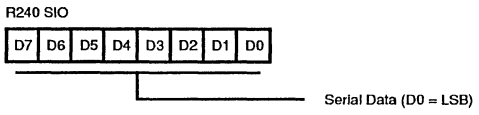


Figure 32. Serial I/O Register (F0H: Read/Write)

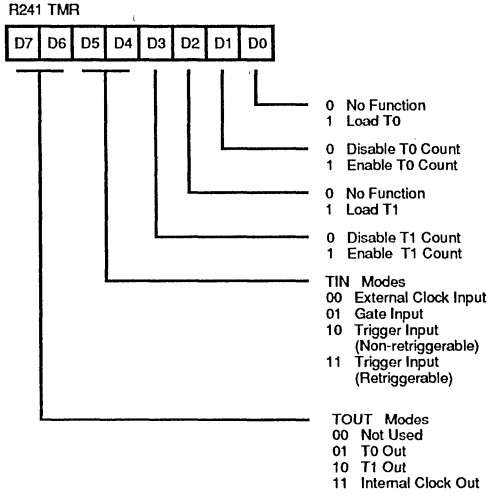


Figure 33. Timer Mode Register (F1H:Read/Write)

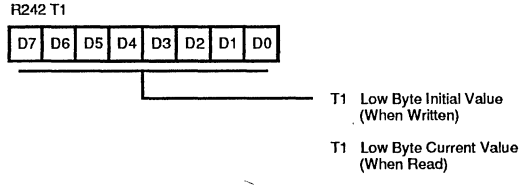


Figure 34. Counter/Timer 1 Register (F2H:Read/Write)

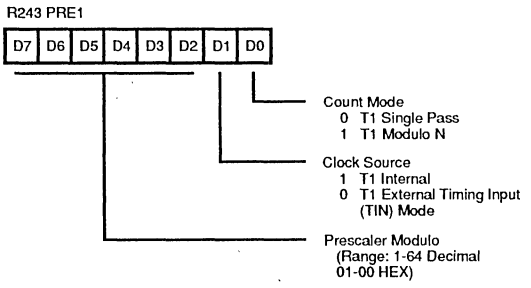


Figure 35. Prescaler 1 Register (F3H:Write Only)

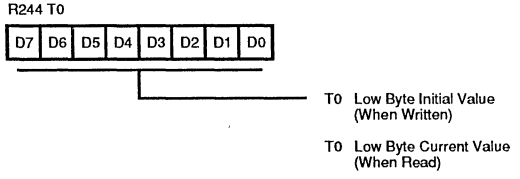


Figure 36. Counter/Timer 0 Register (F4H:Read/Write)

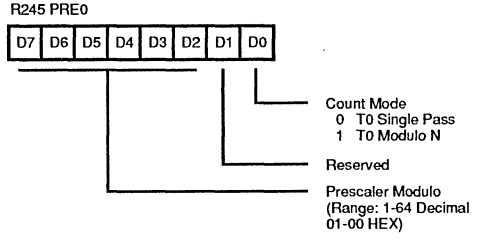


Figure 37. Prescaler 0 Register (F5H:Write Only)

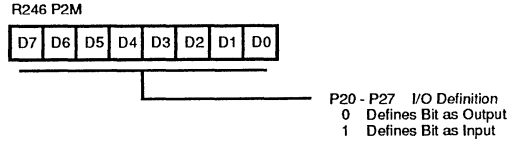


Figure 38. Port 2 Mode Register (F6H: Write Only)

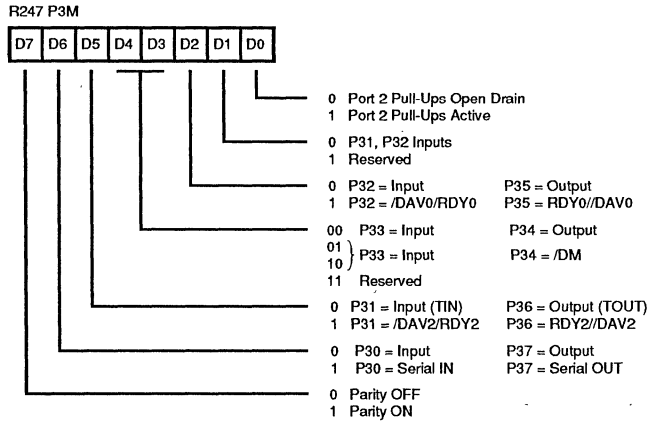


Figure 39. Port 3 Mode Register (F7H:Write Only)

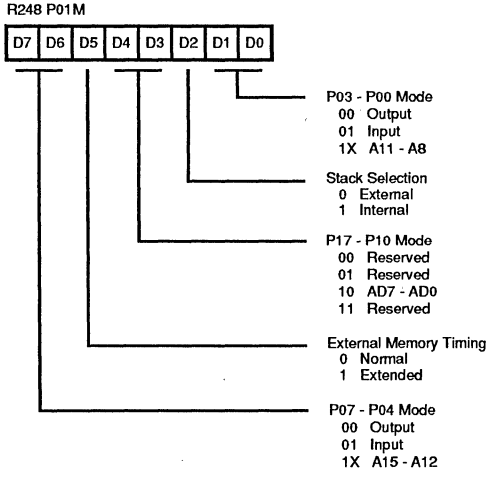


Figure 40. Ports 0 and 1 Mode Registers (F8H:Write Only)

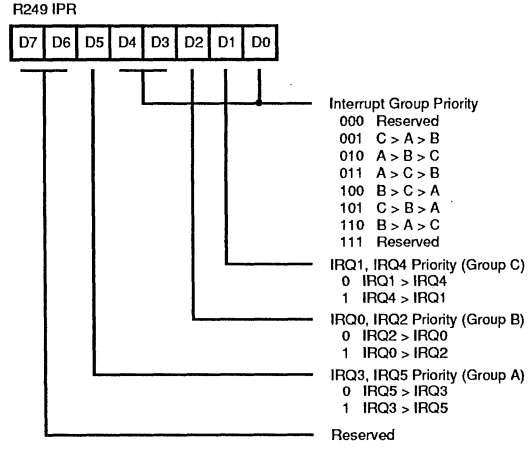


Figure 41. Interrupt Priority Register (F9H:Write Only)

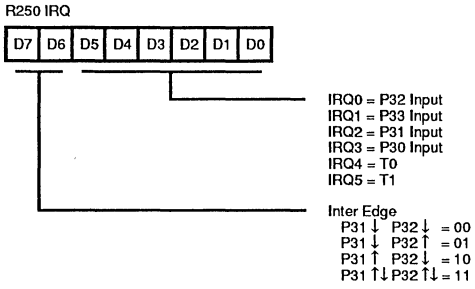


Figure 42. Interrupt Request Register (FAH:Read/Write)

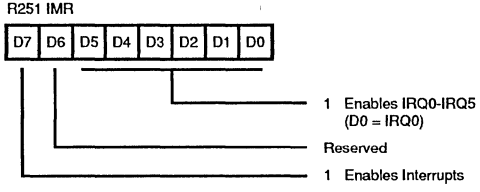


Figure 43. Interrupt Mask Register (FBH:Read/Write)

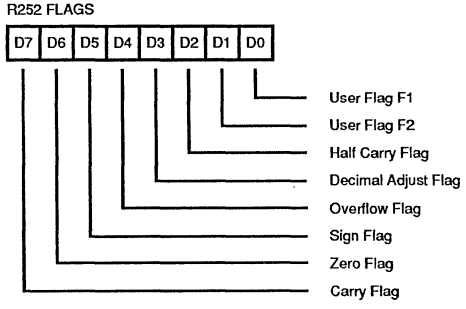


Figure 44. Flag Register (FCH:Read/Write)

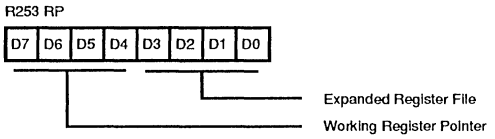


Figure 45. Register Pointer (FDH:Read/Write)

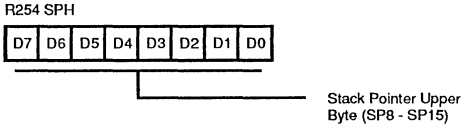


Figure 46. Stack Pointer High (FEH:Read/Write)

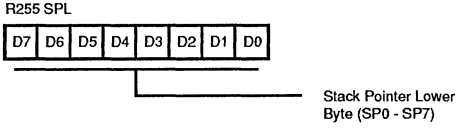


Figure 47. Stack Pointer Low (FFH:Read/Write)

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

Symbol	Meaning
IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

Symbol	Meaning
dst	Destination location or contents
src	Source location or contents
cc	Condition code
@	Indirect address prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag register (Control Register 252)
RP	Register Pointer (R253)
IMR	Interrupt mask register (R251)

Flags. Control register (R252) contains the following six flags:

Symbol	Meaning
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

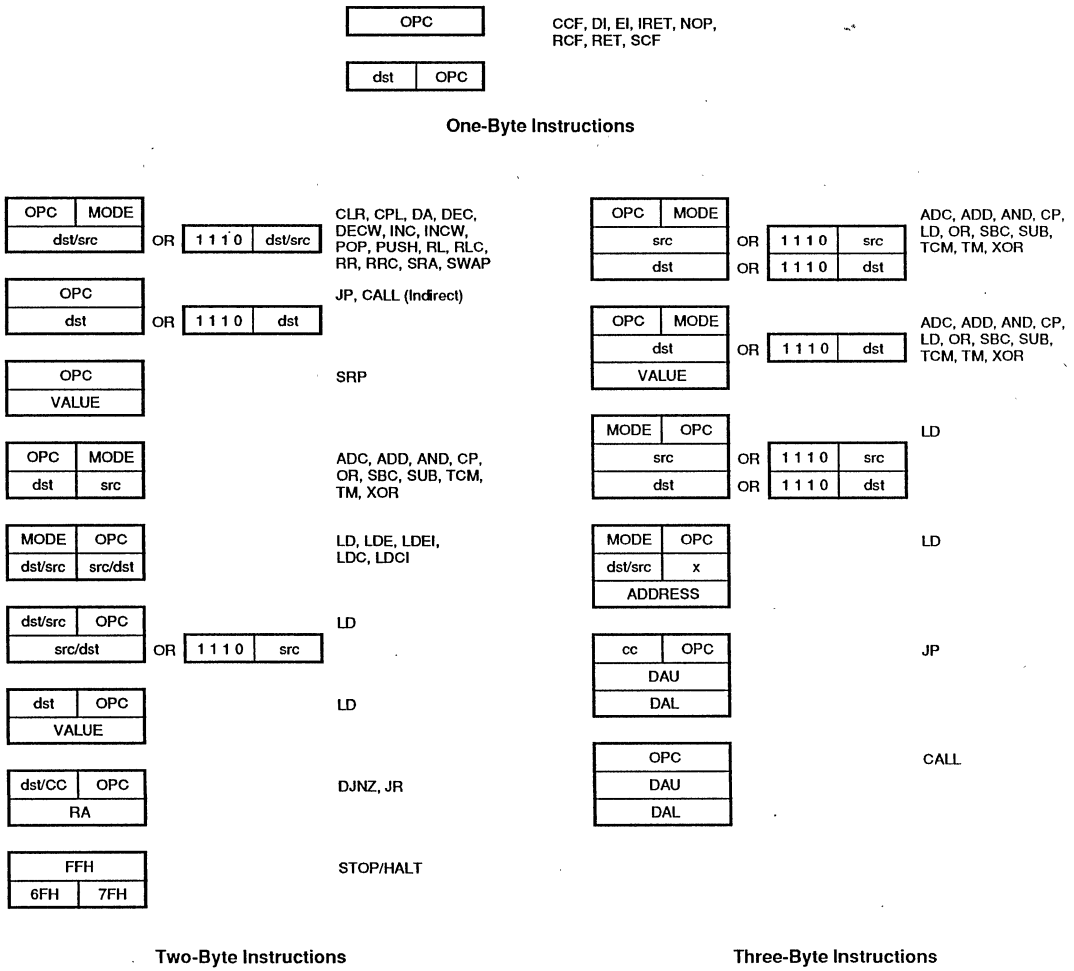
Affected flages are indicated by:

0	Clear to zero
1	Set to one
*	Set to clear according to operation
-	Unaffected
x	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000		Always True	
0111	C	Carry	C = 1
1111	NC	No Carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not Zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No Overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not Equal	Z = 0
1001	GE	Greater Than or Equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater Than	[Z OR (S XOR V)] = 0
0010	LE	Less Than or Equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned Greater Than or Equal	C = 0
0111	ULT	Unsigned Less Than	C = 1
1011	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned Less Than or Equal	(C OR Z) = 1
0000		Never True	

INSTRUCTION FORMATS



INSTRUCTION SUMMARY

Note: Assignment of a value is indicated by the symbol " ← ". For example:

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr (n)" is used to refer to bit (n) of a given operand location. For example:

dst (7)

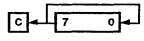
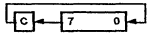
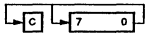
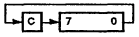
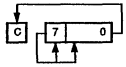
refers to bit 7 of the destination operand.

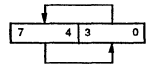
INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst, src dst←dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst←dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst←dst AND src	†		5[ ]	-	*	*	0	-	-
CALL dst SP←SP - 2 @SP←PC, PC←dst	DA IRR		D6 D4	-	-	-	-	-	-
CCF C←NOT C			EF	*	-	-	-	-	-
CLR dst dst←0	R IR		B0 B1	-	-	-	-	-	-
COM dst dst←NOT dst	R IR		60 61	-	*	*	0	-	-
CP dst, src dst - src	†		A[ ]	*	*	*	*	-	-
DA dst dst←DA dst	R IR		40 41	*	*	*	X	-	-
DEC dst dst←dst - 1	R IR		00 01	-	*	*	*	-	-
DECW dst dst←dst - 1	RR IR		80 81	-	*	*	*	-	-
DI IMR(7)←0			8F	-	-	-	-	-	-
DJNZr, dst r←r - 1 if r ≠ 0 PC←PC + dst Range: +127, -128	RA		rA r = 0 - F	-	-	-	-	-	-
EI IMR(7)←1			9F	-	-	-	-	-	-
HALT			7F	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
INC dst dst←dst + 1	r R IR		rE r = 0 - F 20 21	-	*	*	*	-	-
INCW dst dst←dst + 1	RR IR		A0 A1	-	*	*	*	-	-
IRET FLAGS←@SP; SP←SP + 1 PC←@SP; SP←SP + 2; IMR(7)←1			BF	*	*	*	*	*	*
JP cc, dst if cc is true PC←dst	DA IRR		cD c = 0 - F 30	-	-	-	-	-	-
JR cc, dst if cc is true, PC←PC + dst Range: +127, -128	RA		cB c = 0 - F	-	-	-	-	-	-
LD dst, src dst←src	r r R r X r Ir R R IR IR	Im R r X r Ir r R IR IM R	rC r8 r9 r = 0 - F C7 D7 E3 F3 E4 E5 E6 E7 F5	-	-	-	-	-	-
LDC dst, src	r	Irr	C2	-	-	-	-	-	-
LDCI dst, src dst←src r←r +1; rr←rr + 1	Ir	Irr	C3	-	-	-	-	-	-

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
NOP			FF	-	-	-	-	-	-
OR dst, src dst←dst OR src	†		4[ ]	-	*	*	0	-	-
POP dst dst←@SP; SP←SP + 1	R IR		50 51	-	-	-	-	-	-
PUSH src SP←SP - 1; @SP←src		R IR	70 71	-	-	-	-	-	-
RCF C←0			CF	0	-	-	-	-	-
RET PC←@SP; SP←SP + 2			AF	-	-	-	-	-	-
RL dst 	R IR		90 91	*	*	*	*	-	-
RLC dst 	R IR		10 11	*	*	*	*	-	-
RR dst 	R IR		E0 E1	*	*	*	*	-	-
RRC dst 	R IR		C0 C1	*	*	*	*	-	-
SBC dst, src dst←dst←src←C	†		3[ ]	*	*	*	*	1	*
SCF C←1			DF	1	-	-	-	-	-
SRA dst 	R IR		D0 D1	*	*	*	0	-	-
SRP src RP←src		Im	31	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
STOP			6F	-	-	-	-	-	-
SUB dst, src dst←dst←src	†		2[ ]	*	*	*	*	1	*
SWAP dst 	R IR		F0 F1	X	*	*	X	-	-
TCM dst, src (NOT dst) AND src	†		6[ ]	-	*	*	0	-	-
TM dst, src dst AND src	†		7[ ]	-	*	*	0	-	-
XOR dst, src dst←dst XOR src	†		B[ ]	-	*	*	0	-	-

† These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

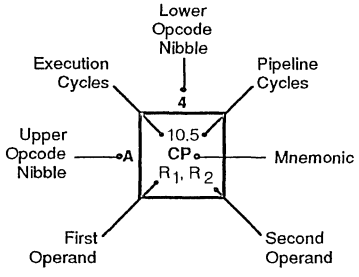
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address Mode dst	Address Mode src	Lower Opcode Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

OPCODE MAP

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								6.0 **STOP**
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								7 0 **HALT**
8	10.5 **DECW** RR1	10.5 **DECW** IR1	12.0 **LDE** r1, Irr2	18.0 **LDEI** Ir1, Irr2												6.1 **DI**
9	6.5 **RL** R1	6.5 **RL** IR1	12.0 **LDE** r2, Irr1	18.0 **LDEI** Ir2, Irr1												6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14 0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2				10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1	12.0 **LDC** r2, Irr1	18.0 **LDCI** Ir2, Irr1	20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6 5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2			3					2				3		1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_2 = Dst address
R_1 or r_2 = Src address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank are not defined.

* 2-byte instruction appears as a 3-byte instruction

GENERAL DESCRIPTION

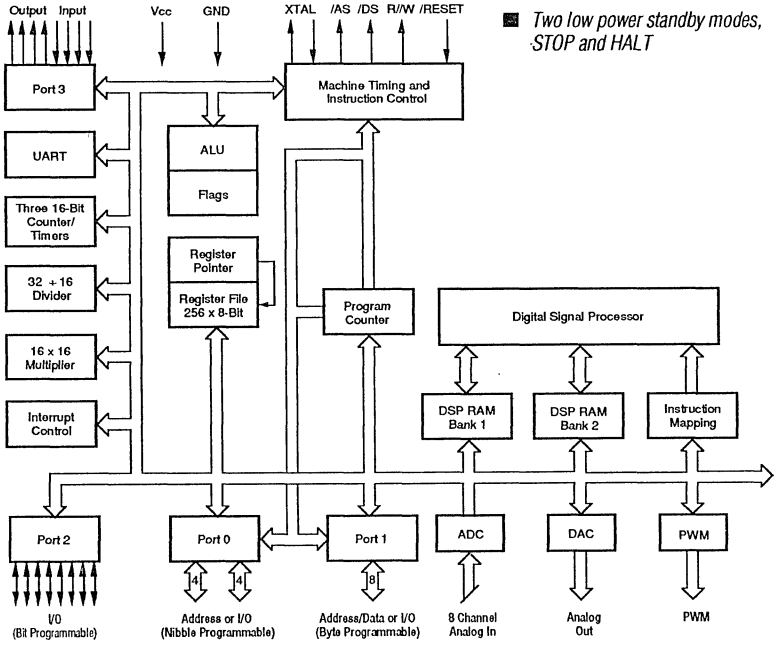
The Z86C94 is a CMOS ROMless Z8 microcontroller integrated with a digital signal processor as a slave processor. With the DSP slave processor, a 16-bit x 16-bit multiplication and accumulation can be accomplished in one clock cycle. In addition, it is further enhanced with a hardwired 16-bit x 16-bit multiplier and 32-bit/16-bit divider, three 16-bit counter timers with capture and compare registers, a fast 8-bit A/D converter, an 8-bit DAC with 4x programmable gain stage, UART and a PWM output channel. It is fabricated using 1.2 micron CMOS technology and offered in a 80-pin Plastic Quad Flat Pack.

The Z86C94 provides up to 16 output address lines permitting an address space of up to 64K bytes of data and program memory each. Eight address outputs (AD7-AD0) are provided by a multiplexed, 8-bit, Address/Data bus. The remaining 8 bits can be provided by the software configuration of Port 0 to output address bits A8-A15.

There are 256 registers located on-chip organized as 236 general purpose registers, 16 control and status registers, and three I/O port registers. Register file can be divided into sixteen groups of 16 working registers each. Configuring the registers in this manner allows the use of short format instructions; in addition, any of the individual registers can be accessed directly.

FEATURES

- *Complete microcomputer, 24 I/O lines, and up to 64K bytes of addressable external space each for program and data memory.*
- *DSP slave processor capable of 16-bit x 16-bit multiplication and accumulation in one clock cycle.*
- *16-bit x 16-bit hardwired divider with 16-bit quotient and 16-bit remainder in 20 clock cycles.*
- *An 8-channel 8-bit A/D converter with sample and hold. The maximum single conversion time is 2 µs.*
- *An 8-bit D/A converter with 4x programmable gain stage.*
- *A pulse width modulator output at 40-80 KHz*
- *256-byte register file, including 236 general purpose registers, three I/O port registers and 16 status and control registers.*
- *Vectored, priority interrupts for I/O, counter/timers and UART.*
- *On-chip oscillator that accepts crystal or external clock drive*
- *Full-duplex UART*
- *Three 16-bit counter timers with capture and compare registers*
- *Register Pointer so that short, fast instructions can access any one of the sixteen working register groups.*
- *Single +5V power supply, all I/O pins TTL compatible*
- *1.2 micron CMOS technology*
- *Clock speed - 20 MHz*
- *Two low power standby modes, STOP and HALT*



GENERAL DESCRIPTION

The Z86C96 is a CMOS ROMless version of the Z8 single-chip microcomputer. It offers all the outstanding features of the Z8 family architecture except an on-chip program ROM. Use of external memory rather than a preprogrammed ROM enables this Z8 microcomputer to be used in applications where code flexibility is required.

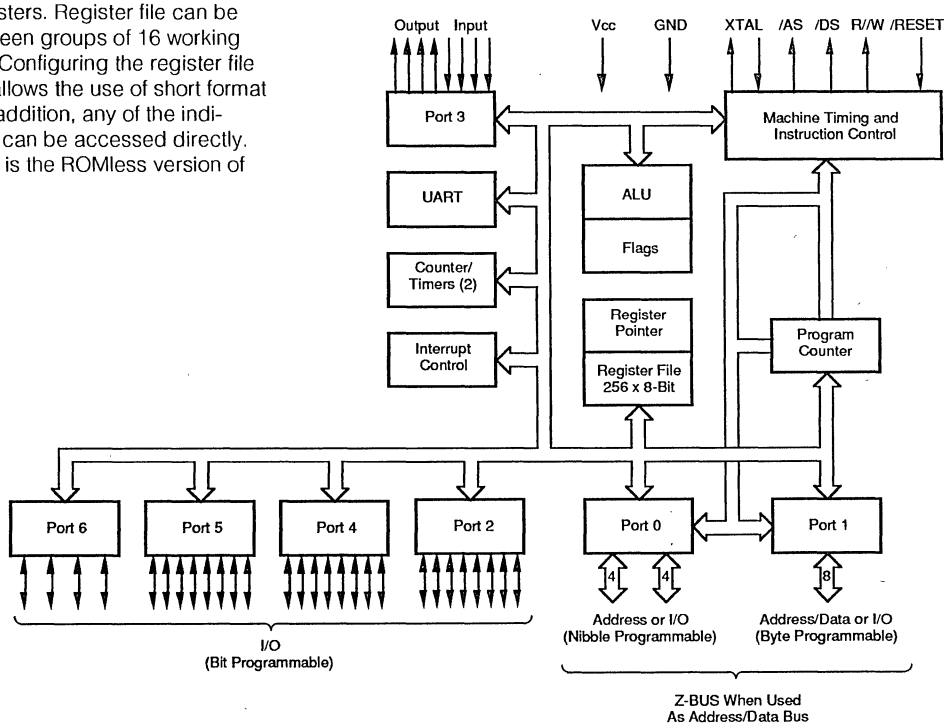
The Z86C96 can provide up to 16 output address lines, thus permitting an address space of up to 64K bytes of data or program memory. Eight address outputs (AD7-AD0) are provided by a multiplexed, 8-bit, Address/Data bus. The remaining 8 bits can be provided by the software configuration of Port 0 to output address bits A15-A8.

There are 256 registers located on-chip organized as 236 general purpose registers, 16 control and status registers, and six I/O port registers. Register file can be divided into sixteen groups of 16 working registers each. Configuring the register file in this manner allows the use of short format instructions; in addition, any of the individual registers can be accessed directly.

The Z86C96 is the ROMless version of the Z86C62.

FEATURES

- *Complete microcomputer, 56 I/O lines, and up to 64K bytes of addressable external space each for program and data memory.*
- *256-byte register file, including 236 general purpose registers, six I/O port registers, and 16 status and control registers.*
- *Three Expanded Register File I/O port registers and five control registers*
- *Vectored, priority interrupts for I/O, counter/timers, and UART.*
- *On-chip oscillator that accepts crystal, ceramic resonator, LC or external clock drive.*
- *Full-duplex UART and two programmable 8-bit counter/timers, each with a 6-bit programmable prescaler.*
- *Register Pointer so that short, fast instructions can access any one of the sixteen working-register groups.*
- *3.0 to 5.5 volts operating range*
- *Clock speed 20 MHz*
- *1.2 micron CMOS technology*
- *Two low-power standby modes, STOP and HALT*



ADVANCE INFORMATION SPECIFICATION

Z88C00

CMOS SUPER8™ ROMLESS MCU

FEATURES

- Full Super8 Instruction Set
- CMOS Technology
- Available in 48- and 68-pin packages
- Multiply and Divide instructions, Boolean and BCD operations
- 325 byte registers, including 272 general-purpose registers, and 53 mode and control registers
- Addresses up to 128 Kbytes of external program and data memory
- Two register pointers allow 600 nsec access time
- Direct Memory Access (DMA)
- Two 16-bit counter/timers with 8-bit prescalers
- Up to 32 bit-programmable and byte-programmable I/O lines, with two handshake channels
- Interrupt structure supports:
  - 27 interrupt sources
  - 16 interrupt vectors
  - 8 interrupt levels
  - Servicing capabilities in 600 nsec
- Full-Duplex UART with special features
- On-Chip oscillator
- DC to 25 MHz operating frequency
- STOP Mode
- HALT Mode with the use of WFI instruction
- TEST word feature
- Low power consumption
- Full pin-for-pin compatibility with NMOS Super8

GENERAL DESCRIPTION

The CMOS Super8 offers new flexibility and sophistication in 8-bit microcontrollers. The Super8 offers all the features necessary for industrial, consumer, and automotive applications with an enhanced feature set in CMOS technology. At the same time, the CMOS Super8 retains full pin-for-pin compatibility with the NMOS Super8. Available in 48-pin Dual In-line Plastic (DIP) and 64-pin Plastic Leaded Chip Carrier (PLCC), the CMOS Super8 is the last word in general purpose controllers.

The Super8 features a full-duplex, Universal Asynchronous Receiver/Transmitter (UART) with on-chip baud rate generator, on-chip oscillator, 2 16-bit counter/timers each with an 8-bit prescaler, a Direct Memory Access controller (DMA), Watch Dog Timer (WDT), and STOP mode.

Incorporated in the new CMOS Super8 is also a bonding option for a de-multiplexed external memory interface bus. In de-muxed mode, PORTs 0 and 4 function as address ports, with PORT 1 as data bus. PORT 4 drives the lower address bits and PORT 0 drives the upper address bits, when both ports are configured as external memory interface. This gives the user more addressing flexibility.

Finally, by adding the enhanced features of WDT, STOP and HALT modes, and more versatile counter/timers, the CMOS Super8 can fit easily into more complex function applications where a general-purpose microcontroller is a necessity.

GENERAL DESCRIPTION (Continued)

Figure 1 is the functional block diagram of Z88C00. The device is housed in a 48-pin DIP (Figure 2), and a 68-pin PLCC package (Figure 3). The pin functions of the Z88C00 are shown in Figure 4.

Note: All Signals with a preceding front slash, "/", are active Low, e.g.: B/W (WORD is active Low); /B/W (BYTE is active Low, only); /N//S (NORMAL and SYSTEM are both active Low).

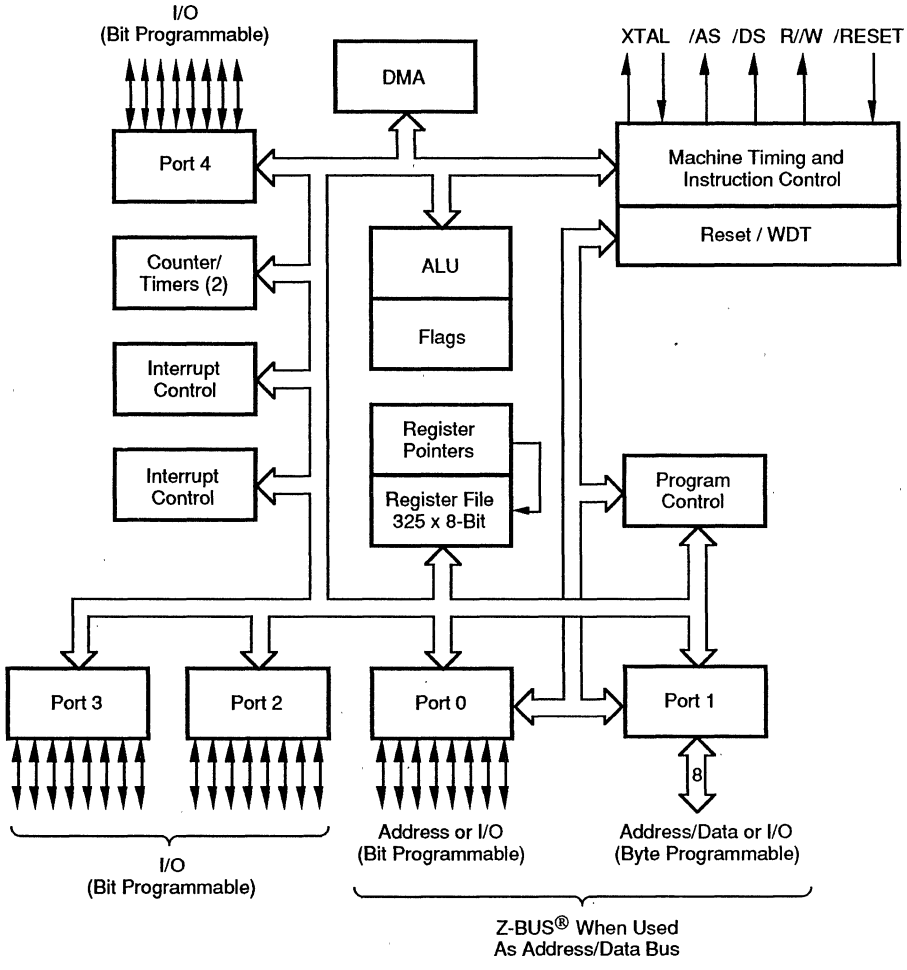


Figure 1. Functional Block Diagram

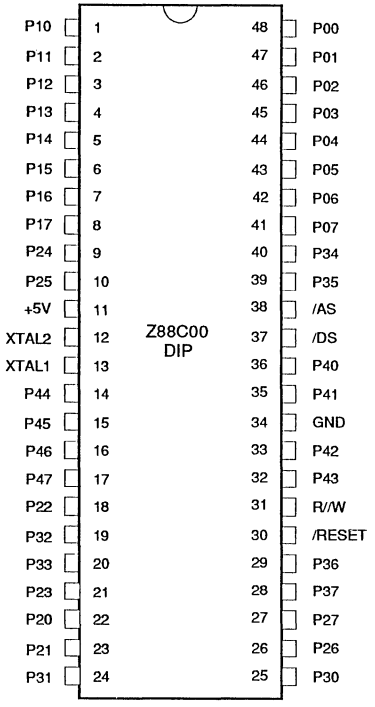


Figure 2. 48-Pin Plastic Dual In-Line (PDIP) Package

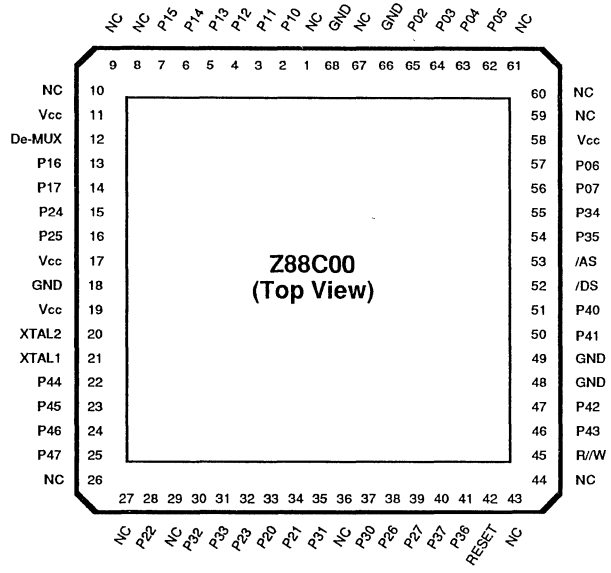


Figure 3. 68-Pin Plastic Leaded Chip Carrier (PLCC) Package

PIN FUNCTIONS

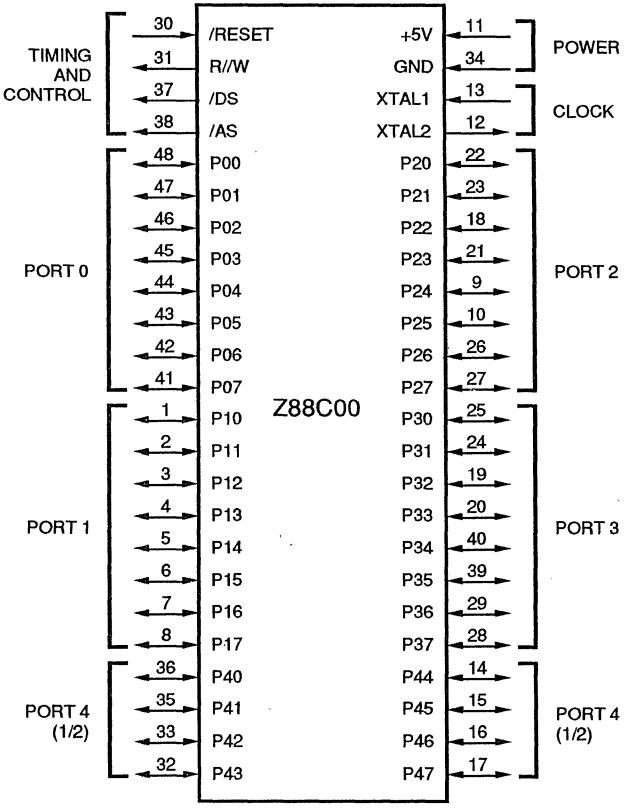


Figure 4. Pin Functions

NMOS PRODUCTS

Z8600 Z8®
Microcomputer

FEATURES

- Complete microcomputer, 2K bytes of ROM, 128 bytes of RAM, and 22 I/O lines.
- 144-byte register file, including 124 general-purpose registers, four I/O port registers, and 14 status and control registers.
- Vectored, priority interrupts for I/O and counter/timers.
- Two programmable 8-bit counter/timers, each with a 6-bit programmable prescaler.
- Register Pointer so that short, fast instructions can access any one of the nine working register groups.
- On-chip oscillator that accepts crystal or external clock drive.
- 8 MHz
- Single +5 power supply—all pins TTL-compatible.
- Average instruction execution time of 2.2 μs, maximum 1.5 μs.

GENERAL DESCRIPTION

The Z8600 microcomputer introduces a new level of sophistication to single-chip architecture. Compared to earlier single-chip microcomputers, the Z8600 offers:

- faster execution
- more efficient use of memory
- more sophisticated interrupt, input/output, and bit manipulation capabilities
- easier system expansion

Under program control, the MCU can be tailored to the needs of its user. It can be configured as a stand-alone microcomputer with 2K bytes of internal ROM. In all configurations, a large number of pins remain available for I/O.

The MCU is offered in a 28 pin Dual-In-Line-Package (DIP) (Figures 1 and 2).

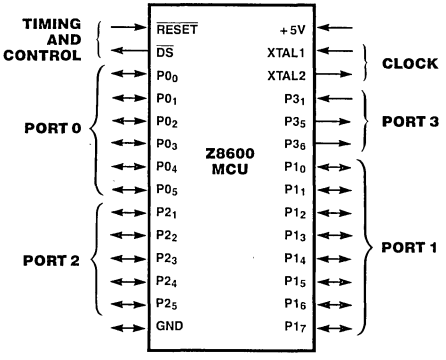


Figure 1. Pin Functions

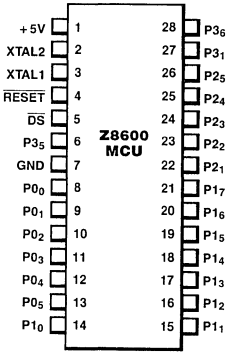


Figure 2. Pin Assignments

PIN DESCRIPTIONS

$\overline{DS}$. *Data Strobe* (output, active Low). Data Strobe is activated once for each memory transfer.

$P0_0$-$P0_5$, $P1_0$-$P1_7$, $P2_1$-$P2_5$, $P3_1$, $P3_5$, $P3_6$. *I/O Port lines* (bidirectional, TTL-compatible). These 22 I/O lines are grouped in four ports that can be configured under program control for I/O.

$\overline{RESET}$. *Reset* (input, active Low). $\overline{RESET}$ initializes the MCU. When $\overline{RESET}$ is deactivated, program execution begins from internal program location $000C_H$.

XTAL1, XTAL2. *Crystal 1, Crystal 2* (time-base input and output). These pins connect a parallel-resonant 8 MHz crystal to the on-chip clock oscillator and buffer.

ARCHITECTURE

The MCU's architecture is characterized by a flexible I/O scheme, an efficient register and address space structure, and a number of ancillary features that are helpful in many applications. (Figure 3).

Microcomputer applications demand powerful I/O capabilities. The MCU fulfills this with 22 pins dedicated to input and output. These lines are grouped in four ports and are configurable under software control to provide timing, status signals, and parallel I/O.

Two basic internal address spaces are available to support this wide range of configurations: program memory and the register file. The 144-byte random-access register file is composed of 124 general-purpose registers, four I/O port registers, and 14 control and status registers.

To unburden the program from coping with real-time problems such as counting/timing, two counter/timers with a large number of user-selectable modes are offered on-chip.

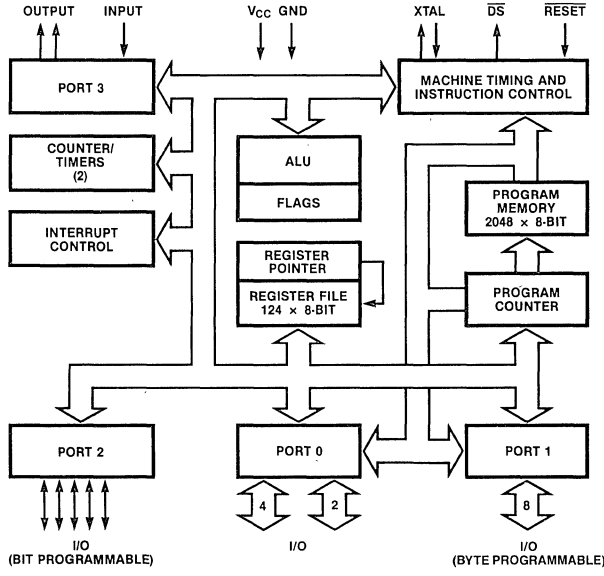


Figure 3. Functional Block Diagram

ADDRESS SPACES

Program Memory. The 16-bit program counter addresses 2K bytes of program memory space as shown in Figure 4.

The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain three 16-bit vectors that correspond to the three available interrupts.

Register File. The 144-byte register file includes four I/O port registers (R_0-R_3), 124 general-purpose registers (R_4-R_{127}) and **14** control and status registers (R_{241}-R_{255}). These registers are assigned the address locations shown in Figure 5.

Instructions can access registers directly or indirectly with an 8-bit address field. The MCU also allows short 4-bit register addressing using the Register Pointer (one of the control registers). In the 4-bit mode, the register file is divided into nine working-register groups, each occupying 16 contiguous locations (Figure 6). The Register Pointer addresses the starting location of the active working-register group.

Stacks. An 8-bit Stack Pointer (R_{255}) is used for the internal stack that resides within the 124 general-purpose registers (R_4-R_{127}).

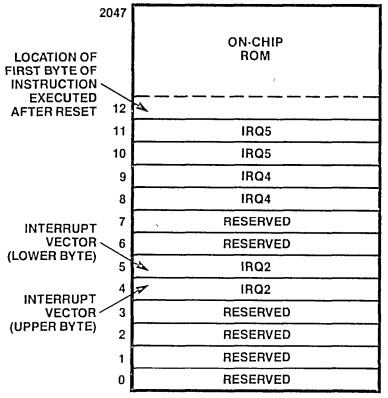


Figure 4. Program Memory Map

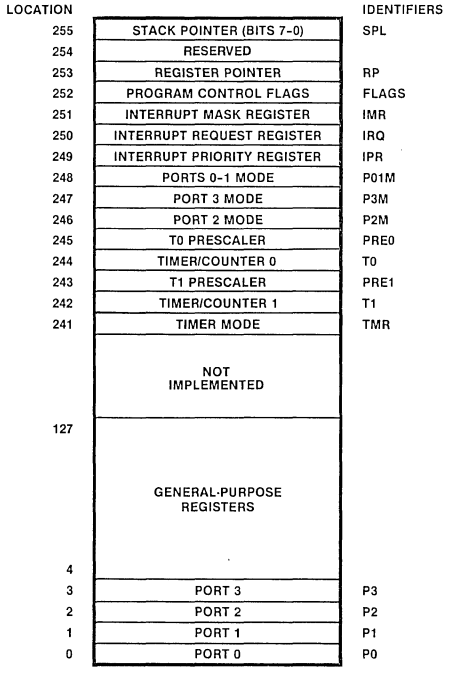


Figure 5. Register File

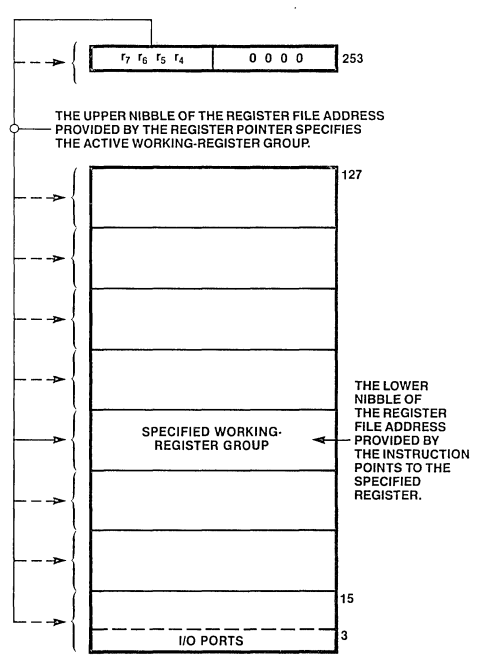


Figure 6. Register Pointer

COUNTER/TIMERS

The MCU contains two 8-bit programmable counter/timers (T_0 and T_1), each driven by its own 6-bit programmable prescaler. The T_1 prescaler can be driven by internal or external clock sources; however, the T_0 prescaler is driven by the internal clock only.

The 6-bit prescalers can divide the input frequency of the clock source by any number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request—IRQ_4 (T_0) or IRQ_5 (T_1)—is generated.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. The counters can also be programmed to stop upon reaching zero (single-pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode). The counters, but not the prescalers, can be read any time without disturbing their value or count mode.

The clock source for T_1 is user-definable and can be the internal microprocessor clock (4 MHz maximum) divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input as an external clock (1 MHz maximum), a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. The counter/timers can be programmably cascaded by connecting the T_0 output to the input of T_1. Port 3 line $P3_6$ also serves as a timer output (T_{OUT}) through which T_0, T_1 or the internal clock can be output.

I/O PORTS

The MCU has 22 lines dedicated to input and output grouped in four ports. Under software control, the ports can be programmed to provide address outputs, timing, status signals, and parallel I/O. All ports have active pull-ups and pull-downs compatible with TTL loads.

Port 0 can be programmed as an I/O port.

Port 1 can be programmed as a byte I/O port.

Port 2 can be programmed independently as input or output and is always available for I/O operations. In addition, Port 2 can be configured to provide open-drain outputs.

Port 3 can be configured as I/O or control lines. $P3_1$ is a general purpose input or can be used for an external interrupt request signal (IRQ_2). $P3_5$ and $P3_6$ are general purpose outputs. $P3_6$ is also used for timer input (T_{IN}) and output (T_{OUT}) signals.

INTERRUPTS

The MCU allows three different interrupts from three sources, the Port 3 line $P3_1$ and the two counter/timers. These interrupts are both maskable and prioritized. The Interrupt Mask register globally or individually enables or disables the three interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All interrupts are vectored. When an interrupt request is granted, an interrupt machine cycle is entered. This disables all subsequent interrupts, saves the Program Counter and status flags, and branches to the program memory vector locations reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

Polled interrupt systems are also supported. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

CLOCK

The on-chip oscillator has a high-gain parallel-resonant amplifier for connection to a crystal or to any suitable external clock source (XTAL1 = Input, XTAL2 = Output).

Crystal source is connected across XTAL1 and XTAL2 using the recommended capacitors (C1 $\leqslant$ 15 pf) from each pin to ground. The specifications are as follows:

- AT cut, parallel resonant
- Fundamental type, 8 MHz maximum
- Series resistance, Rs $\leqslant 100\Omega$

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

dst	Destination location or contents
src	Source location or contents
cc	Condition code (see list)
@	Indirect address prefix
SP	Stack pointer (control registers 254-255)
PC	Program counter
FLAGS	Flag register (control register 252)
RP	Register pointer (control register 253)
IMR	Interrupt mask register (control register 251)

Assignment of a value is indicated by the symbol "←". For example,

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

dst (7)

refers to bit 7 of the destination operand.

Flags. Control Register R252 contains the following six flags:

C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

Affected flags are indicated by:

0	Cleared to zero
1	Set to one
*	Set or cleared according to operation
—	Unaffected
X	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000		Always true	—
0111	C	Carry	C = 1
1111	NC	No carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	[Z OR (S XOR V)] = 0
0010	LE	Less than or equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned greater than or equal	C = 0
0111	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1
0000		Never true	—

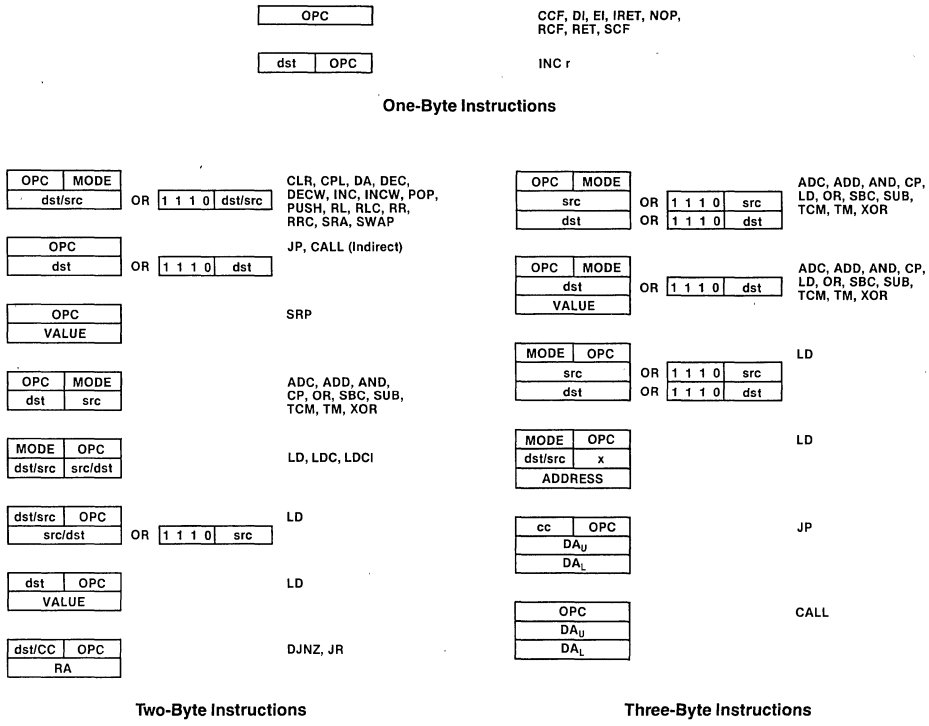


Figure 7. Instruction Formats

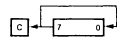
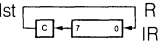
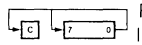
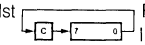
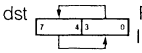
INSTRUCTION SUMMARY

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst,src dst ← dst + src + C	(Note 1)		1□	*	*	*	*	0	*
ADD dst,src dst ← dst + src	(Note 1)		0□	*	*	*	*	0	*
AND dst,src dst ← dst AND src	(Note 1)		5□	—	*	*	0	—	—
CALL dst SP ← SP − 2 @SP ← PC; PC ← dst	DA IRR		D6 D4	—	—	—	—	—	—
CCF C ← NOT C			EF	*	—	—	—	—	—
CLR dst dst ← 0	R IR		B0 B1	—	—	—	—	—	—
COM dst dst ← NOT dst	R IR		60 61	—	*	*	0	—	—

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
CP dst,src dst − src	(Note 1)		A□	*	*	*	*	—	—
DA dst dst ← DA dst	R IR		40 41	*	*	*	X	—	—
DEC dst dst ← dst − 1	R IR		00 01	—	*	*	*	—	—
DECW dst dst ← dst − 1	RR IR		80 81	—	*	*	*	—	—
DI IMR (7) ← 0			8F	—	—	—	—	—	—
DJNZ r,dst r ← r − 1 if r ≠ 0 PC ← PC + dst Range: +127, −128	RA		rA r = 0 − F	—	—	—	—	—	—

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
EI IMR (7) ← 1			9F	—	—	—	—	—	—
INC dst dst ← dst + 1	r		rE r = 0 – F	—	*	*	*	—	—
	R		20						
	IR		21						
INCW dst dst ← dst + 1	RR		A0	—	*	*	*	—	—
	IR		A1						
IRET FLAGS ← @SP; SP ← SP + 2 PC ← @SP; SP ← SP + 2; IMR (7) ← 1			BF	*	*	*	*	*	*
JP cc,dst if cc is true PC ← dst	DA		cD c = 0 – F	—	—	—	—	—	—
	IRR		30						
JR cc,dst if cc is true, PC ← PC + dst Range: +127, –128	RA		cB c = 0 – F	—	—	—	—	—	—
LD dst,src dst ← src	r	Im	rC	—	—	—	—	—	—
	r	R	r8						
	R	r	r9						
			r = 0 – F						
	r	X	C7						
	X	r	D7						
	r	Ir	E3						
	Ir	r	F3						
	R	R	E4						
	R	IR	E5						
	R	IM	E6						
	IR	IM	E7						
	IR	R	F5						
LDC dst,src dst ← src	r	Irr	C2	—	—	—	—	—	—
	Irr	r	D2						
LDCI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir	Irr	C3	—	—	—	—	—	—
	Irr	Ir	D3						
NOP			FF	—	—	—	—	—	—
OR dst,src dst ← dst OR src	(Note 1)		4□	—	*	*	0	—	—
POP dst dst ← @SP; SP ← SP + 1		R	50	—	—	—	—	—	—
		IR							
PUSH src SP ← SP – 1; @SP ← src		R	70	—	—	—	—	—	—
		IR	71						
RCF C ← 0			CF	0	—	—	—	—	—
RET PC ← @SP; SP ← SP + 2			AF	—	—	—	—	—	—

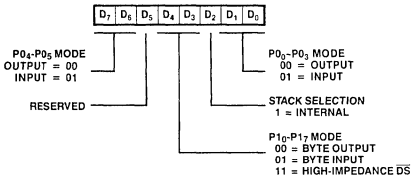
Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
RL dst 	R		90	*	*	*	*	—	—
	IR		91						
RLC dst 	R		10	*	*	*	*	—	—
	IR		11						
RR dst 	R		E0	*	*	*	*	—	—
	IR		E1						
RRC dst 	R		C0	*	*	*	*	—	—
	IR		C1						
SBC dst,src dst ← dst ← src ← C	(Note 1)		3□	*	*	*	*	1	*
SCF C ← 1			DF	1	—	—	—	—	—
SRA dst	R		D0	*	*	*	0	—	—
	IR		D1						
SRP src RP ← src		Im	31	—	—	—	—	—	—
SUB dst,src dst ← dst ← src	(Note 1)		2□	*	*	*	*	1	*
SWAP dst 	R		F0	X	*	*	X	—	—
	IR		F1						
TCM dst,src (NOT dst) AND src	(Note 1)		6□	—	*	*	0	—	—
TM dst,src dst AND src	(Note 1)		7□	—	*	*	0	—	—
XOR dst,src dst ← dst XOR src	(Note 1)		B□	—	*	*	0	—	—

NOTE 1: These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a □ in this table, and its value is found in the following table to the right of the applicable addressing mode pair.

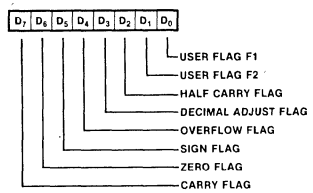
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Addr Mode dst	Addr Mode src	Lower Opcode Nibble
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7

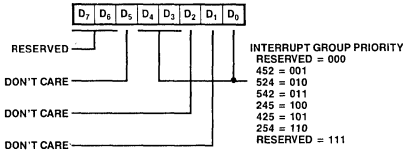
R248 P01M
PORT 0 AND 1 MODE REGISTER
($F8_H$; Write Only)



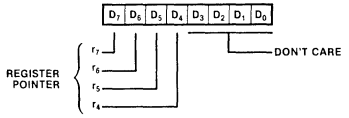
R252 FLAGS
FLAG REGISTER
(FC_H; Read/Write)



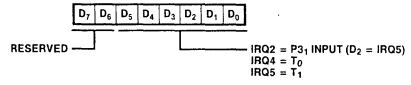
R249 IPR
INTERRUPT PRIORITY REGISTER
($F9_H$; Write Only)



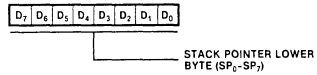
R253 RP
REGISTER POINTER
(FD_H; Read/Write)



R250 IRQ
INTERRUPT REQUEST REGISTER
(FA_H; Read/Write)



R255 SPL
STACK POINTER
(FF_H; Read/Write)



R251 IMR
INTERRUPT MASK REGISTER
(FB_H; Read/Write)

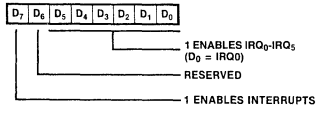
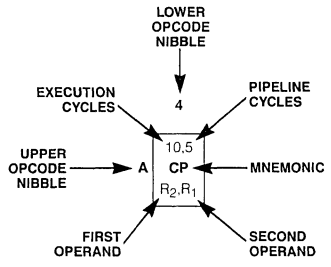


Figure 8. Control Registers (Continued)

OPCODE MAP

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R_1	6.5 **DEC** IR_1	6.5 **ADD** r_1,r_2	6.5 **ADD** r_1,Ir_2	10.5 **ADD** R_2,R_1	10.5 **ADD** IR_2,R_1	10.5 **ADD** R_1,IM	10.5 **ADD** IR_1,IM	6.5 **LD** r_1,R_2	6.5 **LD** r_2,R_1	12/10.5 **DJNZ** r_1,RA	12/10.0 **JR** cc,RA	6.5 **LD** r_1,IM	12/10.0 **JP** cc,DA	6.5 **INC** r1	
1	6.5 **RLC** R_1	6.5 **RLC** IR_1	6.5 **ADC** r_1,r_2	6.5 **ADC** r_1,Ir_2	10.5 **ADC** R_2,R_1	10.5 **ADC** IR_2,R_1	10.5 **ADC** R_1,IM	10.5 **ADC** IR_1,IM								
2	6.5 **INC** R_1	6.5 **INC** IR_1	6.5 **SUB** r_1,r_2	6.5 **SUB** r_1,Ir_2	10.5 **SUB** R_2,R_1	10.5 **SUB** IR_2,R_1	10.5 **SUB** R_1,IM	10.5 **SUB** IR_1,IM								
3	8.0 **JP** IRR_1	6.1 **SRP** IM	6.5 **SBC** r_1,r_2	6.5 **SBC** r_1,Ir_2	10.5 **SBC** R_2,R_1	10.5 **SBC** IR_2,R_1	10.5 **SBC** R_1,IM	10.5 **SBC** IR_1,IM								
4	8.5 **DA** R_1	8.5 **DA** IR_1	6.5 **OR** r_1,r_2	6.5 **OR** r_1,Ir_2	10.5 **OR** R_2,R_1	10.5 **OR** IR_2,R_1	10.5 **OR** R_1,IM	10.5 **OR** IR_1,IM								
5	10.5 **POP** R_1	10.5 **POP** IR_1	6.5 **AND** r_1,r_2	6.5 **AND** r_1,Ir_2	10.5 **AND** R_2,R_1	10.5 **AND** IR_2,R_1	10.5 **AND** R_1,IM	10.5 **AND** IR_1,IM								
6	6.5 **COM** R_1	6.5 **COM** IR_1	6.5 **TCM** r_1,r_2	6.5 **TCM** r_1,Ir_2	10.5 **TCM** R_2,R_1	10.5 **TCM** IR_2,R_1	10.5 **TCM** R_1,IM	10.5 **TCM** IR_1,IM								
7	10/12.1 **PUSH** R_2	12/14.1 **PUSH** IR_2	6.5 **TM** r_1,r_2	6.5 **TM** r_1,Ir_2	10.5 **TM** R_2,R_1	10.5 **TM** IR_2,R_1	10.5 **TM** R_1,IM	10.5 **TM** IR_1,IM								
8	10.5 **DECW** RR_1	10.5 **DECW** IR_1														6.1 **DI**
9	6.5 **RL** R_1	6.5 **RL** IR_1														6.1 **EI**
A	10.5 **INCW** RR_1	10.5 **INCW** IR_1	6.5 **CP** r_1,r_2	6.5 **CP** r_1,Ir_2	10.5 **CP** R_2,R_1	10.5 **CP** IR_2,R_1	10.5 **CP** R_1,IM	10.5 **CP** IR_1,IM								14.0 **RET**
B	6.5 **CLR** R_1	6.5 **CLR** IR_1	6.5 **XOR** r_1,r_2	6.5 **XOR** r_1,Ir_2	10.5 **XOR** R_2,R_1	10.5 **XOR** IR_2,R_1	10.5 **XOR** R_1,IM	10.5 **XOR** IR_1,IM								16.0 **IRET**
C	6.5 **RRC** R_1	6.5 **RRC** IR_1	12.0 **LDC** r_1,Irr_2	18.0 **LDCI** Ir_1,Irr_2				10.5 **LD** r_1,x,R_2								6.5 **RCF**
D	6.5 **SRA** R_1	6.5 **SRA** IR_1	12.0 **LDC** r_2,Irr_1	18.0 **LDCI** Ir_2,Irr_1	20.0 **CALL*** IRR_1		20.0 **CALL** DA	10.5 **LD** r_2,x,R_1								6.5 **SCF**
E	6.5 **RR** R_1	6.5 **RR** IR_1		6.5 **LD** r_1,IR_2	10.5 **LD** R_2,R_1	10.5 **LD** IR_2,R_1	10.5 **LD** R_1,IM	10.5 **LD** IR_1,IM								6.5 **CCF**
F	8.5 **SWAP** R_1	8.5 **SWAP** IR_1		6.5 **LD** Ir_1,r_2		10.5 **LD** R_2,IR_1										6.0 **NOP**
Bytes per Instruction	2			3				2				3			1	



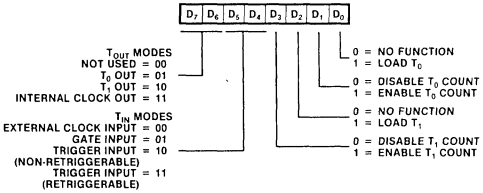
Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_1 = Dst address
R_2 or r_2 = Src address

Sequence:
Opcode, First Operand, Second Operand

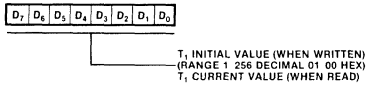
NOTE: The blank areas are not defined.

*2-byte instruction, fetch cycle appears as a 3-byte instruction

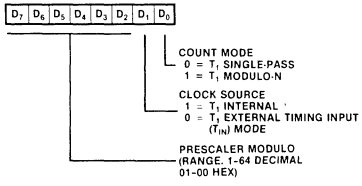
R241 TMR
TIMER MODE REGISTER
($F1_H$; Read/Write)



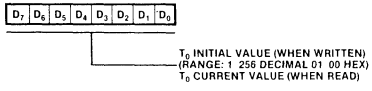
R242 T1
COUNTER TIMER 1 REGISTER
($F2_H$; Read/Write)



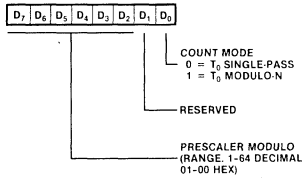
R243 PRE1
PRESCALER 1 REGISTER
($F3_H$; Write Only)



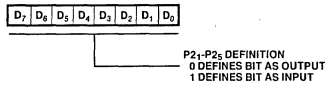
R244 T0
COUNTER/TIMER 0 REGISTER
($F4_H$; Read/Write)



R245 PRE0
PRESCALER 0 REGISTER
($F5_H$; Write Only)



R246 P2M
PORT 2 MODE REGISTER
($F6_H$; Write Only)



R247 P3M
PORT 3 MODE REGISTER
($F7_H$; Write Only)

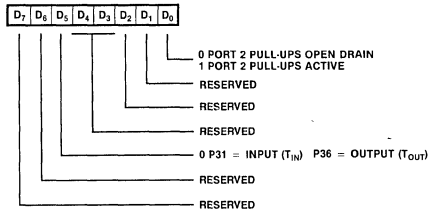


Figure 8. Control Registers

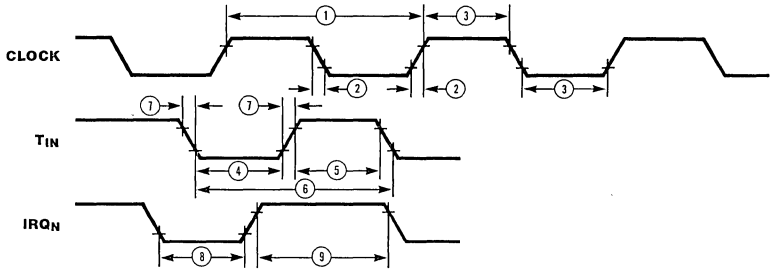


Figure 9. Timing

AC CHARACTERISTICS

Timing Table

Number	Symbol	Parameter	Z8600 Min	Z8600 Max	Notes*
1	TpC	Input Clock Period	125	1000	1
2	TrC,TfC	Clock Input Rise and Fall Times		25	1
3	TwC	Input Clock Width	37		1
4	TwTinL	Timer Input Low Width	100		2
5	TwTinH	Timer Input High Width	3TpC		2
6	TpTin	Timer Input Period	8TpC		2
7	TrTin,TfTin	Timer Input Rise and Fall Times		100	2
8	TwIL	Interrupt Request Input Low Time	100		2,3
9	TwIH	Interrupt Request Input High Time	3TpC		2.3

NOTES:
1. Clock timing references use 3.8V for a logic "1" and 0.8V for a logic "0".
2. Timing references use 2.0V for a logic "1" and 0.8V for a logic "0".
3. Interrupt request via Port 3 ($P3_1$-$P3_3$).
* Units in nanoseconds (ns).

ABSOLUTE MAXIMUM RATINGS

Voltages on all pins with respect to GND . . . . . −0.3V to +7.0V
Operating Ambient Temperature . . . . . .See Ordering Information
Storage Temperature . . . . . −65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

STANDARD TEST CONDITIONS

The DC characteristics listed below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin.

Standard conditions are:

- $+4.75V \leqslant V_{CC} \leqslant +5.25V$
- GND = 0V
- $0°C \leqslant T_A \leqslant +70°C$

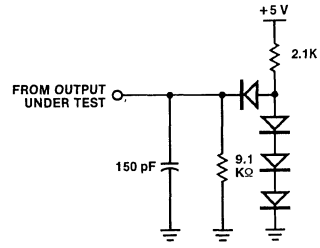


Figure 10. Test Load 1

DC CHARACTERISTICS

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	3.8	V_{CC}	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	−0.3	0.8	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{IL}	Input Low Voltage	−0.3	0.8	V	
V_{RH}	Reset Input High Voltage	3.8	V_{CC}	V	
V_{RL}	Reset Input Low Voltage	−0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\,\mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0\,mA$
I_{IL}	Input Leakage	−10	10	μA	$0V \leqslant V_{IN} \leqslant +5.25V$
I_{OH}	**Output Drive Current**		**1.5** **2.50**	**mA** **μA**	**$V_{OH} = +2.4V$** **$V_{OH} = +4.0V$**
I_{OL}	Output Leakage	−10	10	μA	$0V \leqslant V_{IN} \leqslant +5.25V$
I_{IR}	Reset Input Current		−50	μA	$V_{CC} = +5.25V$, $V_{RL} = 0V$
I_{CC}	V_{CC} Supply Current		150	mA	

Z8601/Z8603
Z8611/Z8613 Z8®

Z8601 Single-Chip MCU with 2K ROM
Z8603 Prototyping Device with 2K EPROM Interface
Z8611 Single-Chip MCU with 4K ROM
Z8613 Prototyping Device with 4K EPROM Interface

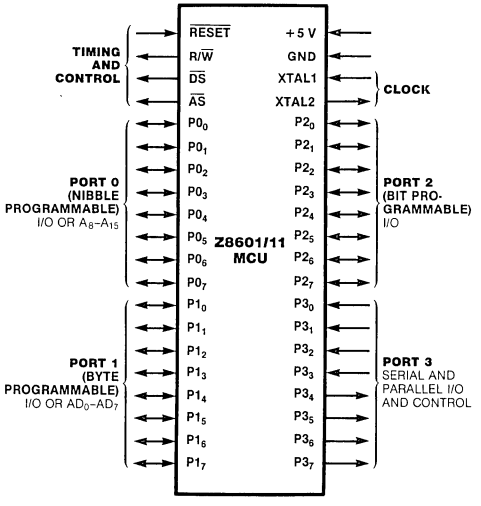
Features

- Complete microcomputer, 2K (8601) or 4K (8611) bytes of ROM, 128 bytes of RAM, 32 I/O lines, and up to 62K (8601) or 60K (8611) bytes addressable external space each for program and data memory.
- 144-byte register file, including 124 general-purpose registers, four I/O port registers, and 16 status and control registers.
- Average instruction execution time of 1.5 μs, maximum of 1 μs.
- Vectored, priority interrupts for I/O, counter/timers, and UART.
- Full-duplex UART and two programmable 8-bit counter/timers, each with a 6-bit programmable prescaler.
- Register Pointer so that short, fast instructions can access any of nine working register groups in 1 μs.
- On-chip oscillator which accepts crystal or external clock drive.
- Single +5 V power supply—all pins TTL compatible.
- 12.5 MHz.

General Description

The Z8 microcomputer introduces a new level of sophistication to single-chip architecture. Compared to earlier single-chip microcomputers, the Z8 offers faster execution; more efficient use of memory; more sophisticated interrupt, input/output and bit-manipulation capabilities; and easier system expansion.

Under program control, the Z8 can be tailored to the needs of its user. It can be configured as a stand-alone microcomputer with 2K or 4K bytes of internal ROM, a traditional microprocessor that manages up to 124K bytes of external memory, or a parallel-processing element in a system with other processors and peripheral controllers linked by the Z-BUS® bus. In all configurations, a large number of pins remain available for I/O.



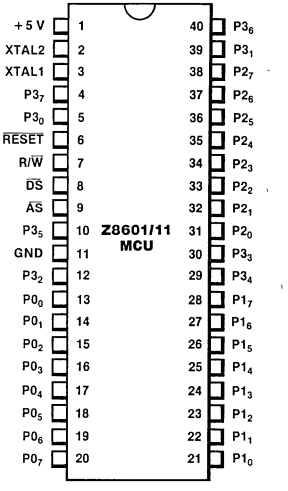


Figure 2a. 40-pin Dual-In-Line Package (DIP), Pin Assignments

Pin Description

$\overline{AS}$. *Address Strobe* (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Addresses output via Port 1 for all external program or data memory transfers are valid at the trailing edge of $\overline{AS}$. Under program control, $\overline{AS}$ can be placed in the high-impedance state along with Ports 0 and 1, Data Strobe and Read/Write.

$\overline{DS}$. *Data Strobe* (output, active Low). Data Strobe is activated once for each external memory transfer.

$P0_0$-$P0_7$, $P1_0$-$P1_7$, $P2_0$-$P2_7$, $P3_0$-$P3_7$. *I/O Port Lines* (input/outputs, TTL-compatible). These 32 lines are divided into four 8-bit I/O ports that can be configured under program control for I/O or external memory interface.

$\overline{RESET}$. *Reset* (input, active Low). $\overline{RESET}$ initializes the Z8. When $\overline{RESET}$ is deactivated, program execution begins from internal program location $000C_H$.

$\overline{ROMless}$. (input, active LOW). This pin is only available on the 44 pin version of the Z8611. When connected to GND disables the internal ROM and forces the part to function as a Z8681 ROMless Z8. When left unconnected or pulled high to V_{CC} the part will function normally as a Z8611.

R/$\overline{W}$. *Read/Write* (output). R/$\overline{W}$ is Low when the Z8 is writing to external program or data memory.

XTAL1, XTAL2. *Crystal 1, Crystal 2* (time-base input and output). These pins connect a parallel resonant 12.5 MHz crystal or an external single-phase 12.5 MHz clock to the on-chip clock oscillator and buffer.

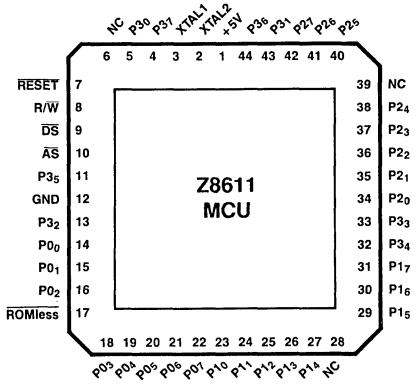


Figure 2b. 44-pin Chip Carrier, Pin Assignments

Architecture

Z8 architecture is characterized by a flexible I/O scheme, an efficient register and address space structure and a number of ancillary features that are helpful in many applications.

Microcomputer applications demand powerful I/O capabilities. The Z8 fulfills this with 32 pins dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an address/data bus for interfacing external memory.

Because the multiplexed address/data bus is merged with the I/O-oriented ports, the Z8 can assume many different memory and I/O configurations. These configurations range from a self-contained microcomputer to a microprocessor that can address 124K (Z8601) or 120K (Z8611) bytes of external memory.

Three basic address spaces are available to support this wide range of configurations: program memory (internal and external), data memory (external) and the register file (internal). The 144-byte random-access register file is composed of 124 general-purpose registers, four I/O port registers, and 16 control and status registers.

To unburden the program from coping with real-time problems such as serial data communication and counting/timing, an asynchronous receiver/transmitter (UART) and two counter/timers with a large number of userselectable modes are offered on-chip. Hardware support for the UART is minimized because one of the on-chip timers supplies the bit rate.

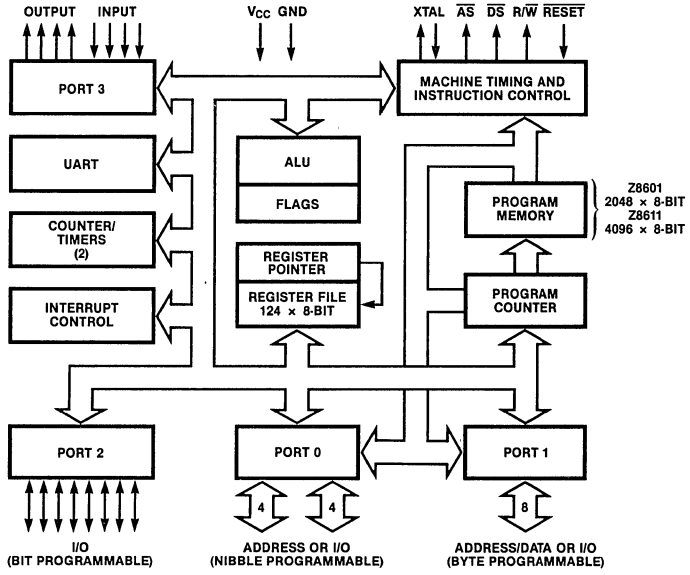


Figure 3. Functional Block Diagram

Address Spaces

Program Memory. The 16-bit program counter addresses 64K bytes of program memory space. Program memory can be located in two areas: one internal and the other external (Figure 4). The first 2048 (Z8601) or 4096 (Z8611) bytes consist of on-chip mask-programmed ROM. At addresses 2048 (Z8601) or 4096 (Z8611) and greater, the Z8 executes external program memory fetches.

The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts.

Data Memory. The Z8 can address 62K (Z8601) or 60K (Z8611) bytes of external data memory beginning at location 2048 (Z8601) or 4096 (Z8611) (Figure 5). External data memory may be included with or separated from the external program memory space. $\overline{DM}$, an optional I/O function that can be programmed to appear on pin $P3_4$, is used to distinguish between data and program memory space.

Register File. The 144-byte register file includes four I/O port registers (R0–R3), 124 general-purpose registers (R4–R127) and 16 control and status registers (R240–R255). These registers are assigned the address locations shown in Figure 6.

Z8 instructions can access registers directly or indirectly with an 8-bit address field. The Z8 also allows short 4-bit register addressing using the Register Pointer (one of the control registers). In the 4-bit mode, the register file is

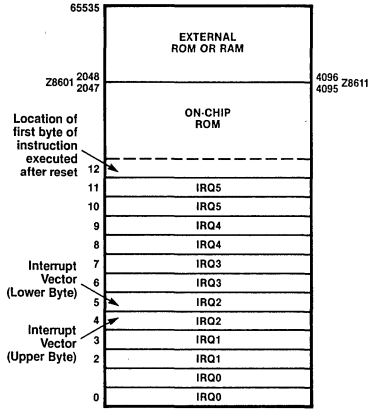


Figure 4. Program Memory Map

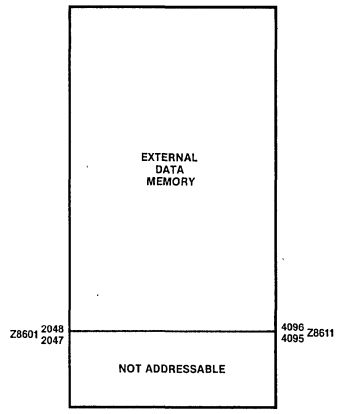


Figure 5. Data Memory Map

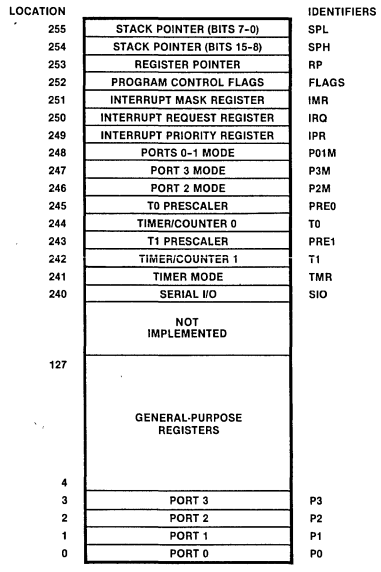


Figure 6. The Register File

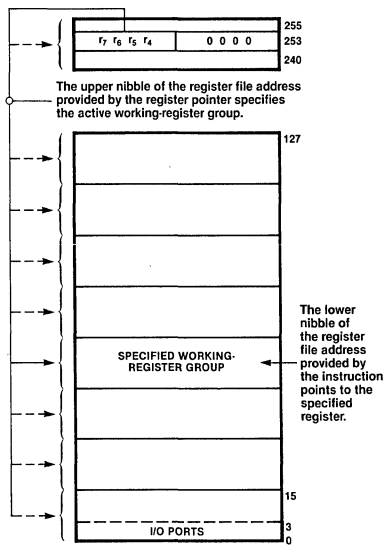


Figure 7. The Register Pointer

divided into nine working-register groups, each occupying 16 continguous locations (Figure 6). The Register Pointer addresses the starting location of the active working-register group.

Stacks. Either the internal register file or the external data memory can be used for the stack. A 16-bit Stack Pointer (R254 and R255) is used for the external stack, which can reside anywhere in data memory between locations 2048 (8601) or 4096 (8611) and 65535. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 124 general-purpose registers (R4–R127).

Serial Input/ Output

Port 3 lines $P3_0$ and $P3_7$ can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by Counter/Timer 0, at 12 MHz.

The Z8 automatically adds a start bit and two stop bits to transmitted data (Figure 8). Odd parity is also available as an option. Eight data bits are always transmitted, regardless of parity selection. If parity is enabled, the eighth bit is the odd parity bit. An interrupt request (IRQ_4) is generated on all transmitted characters.

Received data must have a start bit, eight data bits and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ_3 interrupt request.

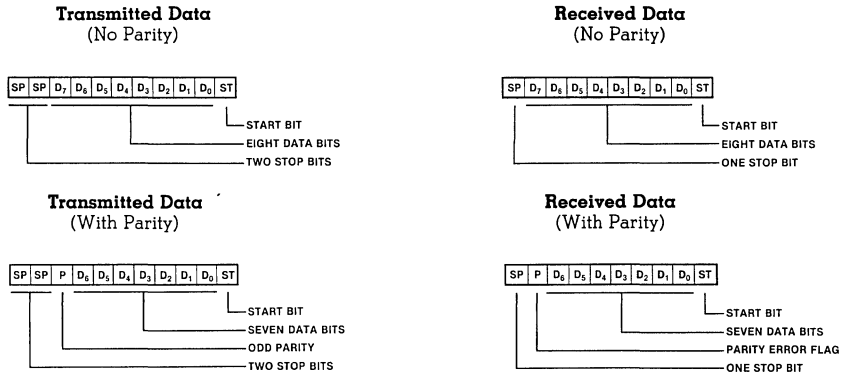


Figure 8. Serial Data Formats

Counter/ Timers

The Z8 contains two 8-bit programmable counter/timers (T_0 and T_1), each driven by its own 6-bit programmable prescaler. The T_1 prescaler can be driven by internal or external clock sources; however, the T_0 prescaler is driven by the internal clock only.

The 6-bit prescalers can divide the input frequency of the clock source by any number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request—IRQ_4 (t_0) or IRQ_5 (T_1)—is generated.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. The counters can also be programmed to stop upon reaching zero (single-pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode). The counters, but not the prescalers, can be read any time without disturbing their value or count mode.

The clock source for T_1 is user-definable and can be the internal microprocessor clock divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input as an external clock, a trigger input that can be retriggerable or nonretriggerable, or as a gate input for the internal clock. The counter/timers can be programmably cascaded by connecting the T_0 output to the input of T_1. Port 3 line $P3_6$ also serves as a timer output (T_{OUT}) through which T_0, T_1 or the internal clock can be output.

I/O Ports

The Z8 has 32 lines dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable as input, output or address/data. Under software control, the ports can be programmed to provide address outputs, timing, status signals, serial I/O, and parallel I/O with or without handshake. All ports have active pull-ups and pull-downs compatible with TTL loads.

Port 1 can be programmed as a byte I/O port or as an address/data port for interfacing external memory. When used as an I/O port, Port 1 may be placed under handshake control. In this configuration, Port 3 lines $P3_3$ and $P3_4$ are used as the handshake controls RDY_1 and $\overline{DAV}_1$ (Ready and Data Available).

Memory locations greater than 2048 (Z8601) or 4096 (Z8611) are referenced through Port 1. To interface external memory, Port 1 must be programmed for the multiplexed Address/Data mode. If more than 256 external locations are required, Port 0 must output the additional lines.

Port 1 can be placed in the high-impedance state along with Port 0, $\overline{AS}$, $\overline{DS}$ and R/$\overline{W}$, allowing the Z8 to share common resources in multiprocessor and DMA applications. Data transfers can be controlled by assigning $P3_3$ as a Bus Acknowledge input and $P3_4$ as a Bus Request output.

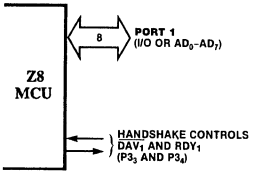


Figure 9a. Port 1

Port 0 can be programmed as a nibble I/O port, or as an address port for interfacing external memory. When used as an I/O port, Port 0 may be placed under handshake control. In this configuration, Port 3 lines $P3_2$ and $P3_5$ are used as the handshake controls $\overline{DAV}_0$ and RDY_0. Handshake signal assignment is dictated by the I/O direction of the upper nibble $P0_4$–$P0_7$.

For external memory references, Port 0 can provide address bits A_8–A_{11} (lower nibble) or A_8–A_{15} (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. When Port 0 nibbles are defined as address bits, they can be set to the highimpedance state along with Port 1 and the control signals $\overline{AS}$, $\overline{DS}$ and R/$\overline{W}$.

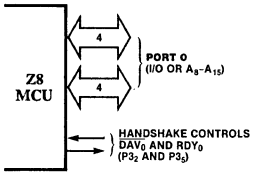


Figure 9b. Port 0

Port 2 bits can be programmed independently as input or output. The port is always available for I/O operations. In addition, Port 2 can be configured to provide open-drain outputs.

Like Ports 0 and 1, Port 2 may also be placed under handshake control. In this configuration, Port 3 lines $P3_1$ and $P3_6$ are used as the handshake controls lines $\overline{DAV}_2$ and RDY_2. The handshake signal assignment for Port 3 lines $P3_1$ and $P3_6$ is dictated by the direction (input or output) assigned to bit 7 of Port 2.

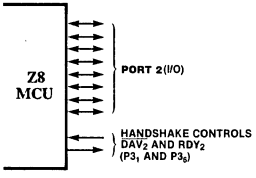


Figure 9c. Port 2

Port 3 lines can be configured as I/O or control lines. In either case, the direction of the eight lines is fixed as four input ($P3_0$–$P3_3$) and four output ($P3_4$–$P3_7$). For serial I/O, lines $P3_0$ and $P3_7$ are programmed as serial in and serial out respectively.

Port 3 can also provide the following control functions: handshake for Ports 0, 1 and 2 ($\overline{DAV}$ and RDY); four external interrupt request signals (IRQ_0–IRQ_3); timer input and output signals (T_{IN} and T_{OUT}) and Data Memory Select ($\overline{DM}$).

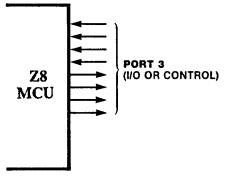


Figure 9d. Port 3

Interrupts

The Z8 allows six different interrupts from eight sources: the four Port 3 lines $P3_0$–$P3_3$, Serial In, Serial Out, and the two counter/timers. These interrupts are both maskable and prioritized. The Interrupt Mask register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All Z8 interrupts are vectored. When an interrupt request is granted, an interrupt machine cycle is entered. This disables all subsequent interrupts, saves the Program Counter and status flags, and branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

Polled interrupt systems are also supported. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

Clock

The on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal or to any suitable external clock source (XTAL1 = Input, XTAL2 = Output).

The crystal source is connected across XTAL1 and XTAL2, using the recommended capacitors ($C_1 \leq 15$ pF) from each pin to ground. The specifications for the crystal are as follows:

- AT cut, parallel resonant
- Fundamental type, 12.5 MHz maximum
- Series resistance, $R_s \leq 100\,\Omega$

Z8603/13 Protopack Emulator

The Z8 Protopack is used for prototype development and preproduction of mask-programmed applications. The Protopack is a ROMless version of the standard Z8601 or Z8611 housed in a pin-compatible 40-pin package (Figure 11).

To provide pin compatibility and interchangeability with the standard maskprogrammed device, the Protopack carries piggy-back a 24-pin socket for a direct interface to program memory (Figure 1). The Z8603 24-pin socket is equipped with 11 ROM address lines, 8 ROM data lines and necessary control lines for interface to 2716 EPROM for the first 2K bytes of program memory. The Z8613 24-pin socket is equipped with 12 ROM address lines, 8 ROM data lines and necessary control lines for interface to 2732 EPROM for the first 4K bytes of program memory.

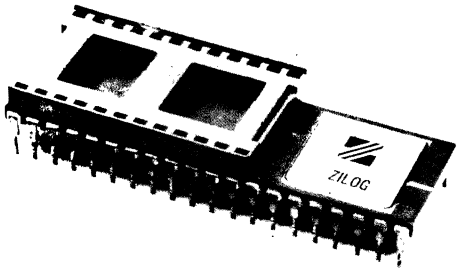


Figure 11. The Z8 Microcomputer Protopack Emulator

Pin compatibility allows the user to design the pc board for a final 40-pin maskprogrammed Z8, and, at the same time, allows the use of the Protopack to build the prototype and pilot production units. When the final program is established, the user can then switch over to the 40-pin mask-programmed Z8 for large volume production. The Protopack is also useful in small volume applica tions where masked ROM setup time, mask charges, etc., are prohibitive and program flexibility is desired.

Compared to the conventional EPROM versions of the single-chip microcomputers, the Protopack approach offers two main advantages:

- Ease of developing various programs during the prototyping stage. For instance, in applications where the same hardware configuration is used with more than one program, the Protopack allows economical program storage in separate EPROMs (or PROMs), whereas the use of separate EPROM-based single-chip microcomputers is more costly.
- Elimination of long lead time in procuring EPROM-based microcomputers.

Instruction Set Notation

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

dst	Destination location or contents
src	Source location or contents
cc	Condition code (see list)
@	Indirect address prefix
SP	Stack pointer (control registers 254–255)
PC	Program counter
FLAGS	Flag register (control register 252)
RP	Register pointer (control register 253)
IMR	Interrupt mask register (control register 251)

Assignment of a value is indicated by the symbol "←". For example,

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

dst (7)

refers to bit 7 of the destination operand.

Flags. Control Register R252 contains the following six flags:

C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

Affected flags are indicated by:

0	Cleared to zero
1	Set to one
*	Set or cleared according to operation
–	Unaffected
X	Undefined

Condition Codes

Value	Mnemonic	Meaning	Flags Set
1000		Always true	---
0111	C	Carry	C = 1
1111	NC	No carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	[Z OR (S XOR V)] = 0
0010	LE	Less than or equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned greater than or equal	C = 0
0111	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1
0000		Never true	---

Instruction Formats

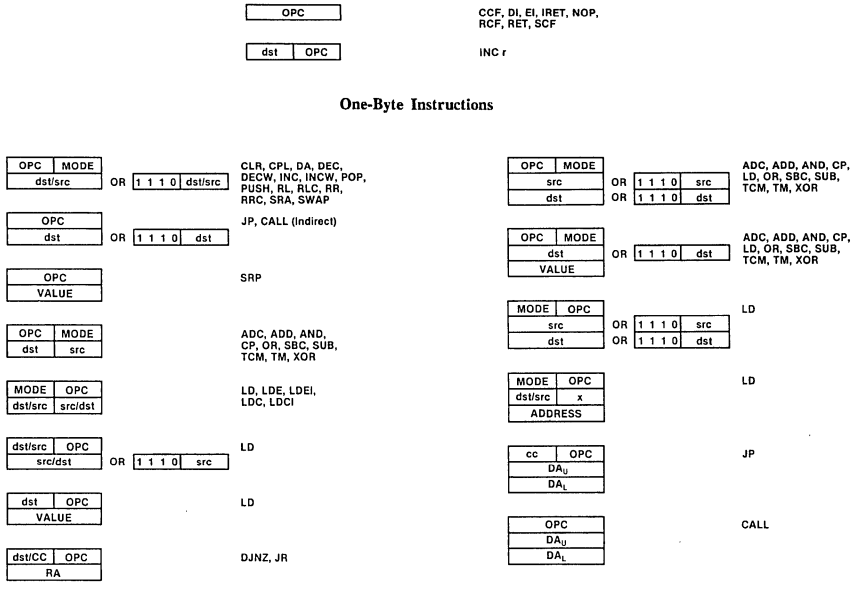
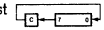
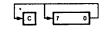
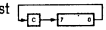
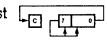


Figure 12. Instruction Formats

Instruction Summary

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
ADC dst,src dst ← dst + src + C	(Note 1)		1□	* * * * 0 *
ADD dst,src dst ← dst + src	(Note 1)		0□	* * * * 0 *
AND dst,src dst ← dst AND src	(Note 1)		5□	- * * 0 - -
CALL dst SP ← SP - 2 @SP ← PC; PC ← dst	DA IRR		D6 D4	- - - - - -
CCF C ← NOT C			EF	* - - - - -
CLR dst dst ← 0	R IR		B0 B1	- - - - - -
COM dst dst ← NOT dst	R IR		60 61	- * * 0 - -
CP dst,src dst - src	(Note 1)		A□	* * * * - -
DA dst dst ← DA dst	R IR		40 41	* * * X - -
DEC dst dst ← dst - 1	R IR		00 01	- * * * - -
DECW dst dst ← dst - 1	RR IR		80 81	- * * * - -
DI IMR (7) ← 0			8F	- - - - - -
DJNZ r,dst r ← r - 1 if r ≠ 0 PC ← PC + dst Range: +127, -128	RA		rA r=0-F	- - - - - -
EI IMR (7) ← 1			9F	- - - - - -
INC dst dst ← dst + 1	r R IR		rE r=0-F 20 21	- * * * - -
INCW dst dst ← dst + 1	RR IR		A0 A1	- * * * - -
IRET FLAGS ← @SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; IMR (7) ← 1			BF	* * * * * *
JP cc,dst if cc is true PC ← dst	DA IRR		cD c=0-F 30	- - - - - -
JR cc,dst if cc is true, PC ← PC + dst Range: +127, -128	RA		cB c=0-F	- - - - - -
LD dst,src dst ← src	r r R r X r Ir R R R IR IR	Im R r X r Ir r R IR Im Im R	rC r8 r9 r=0-F C7 D7 E3 F3 E4 E5 E6 E7 F5	- - - - - -
LDC dst,src dst ← src	r Irr	Irr r	C2 D2	- - - - - -
LDCI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	C3 D3	- - - - - -

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
LDE dst,src dst ← src	r Irr	Irr r	82 92	- - - - - -
LDEI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	83 93	- - - - - -
NOP			FF	- - - - - -
OR dst,src dst ← dst OR src	(Note 1)		4□	- * * 0 - -
POP dst dst ← @SP SP ← SP + 1	R IR		50 51	- - - - - -
PUSH src SP ← SP - 1; @SP ← src		R IR	70 71	- - - - - -
RCF C ← 0			CF	0 - - - - -
RET PC ← @SP; SP ← SP + 2			AF	- - - - - -
RL dst 	R IR		90 91	* * * * - -
RLC dst 	R IR		10 11	* * * * - -
RR dst 	R IR		E0 E1	* * * * - -
RRC dst 	R IR		C0 C1	* * * * - -
SBC dst,src dst ← dst - src - C	(Note 1)		3□	* * * * 1 *
SCF C ← 1			DF	1 - - - - -
SRA dst 	R IR		D0 D1	* * * 0 - -
SRP src RP ← src		Im	31	- - - - - -
SUB dst,src dst ← dst - src	(Note 1)		2□	* * * * 1 *
SWAP dst 	R IR		F0 F1	X * * X - -
TCM dst,src (NOT dst) AND src	(Note 1)		6□	- * * 0 - -
TM dst, src dst AND src	(Note 1)		7□	- * * 0 - -
XOR dst,src dst ← dst XOR src	(Note 1)		B□	- * * 0 - -

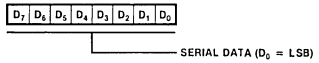
Note 1

These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a □ in this table, and its value is found in the following table to the right of the applicable addressing mode pair.

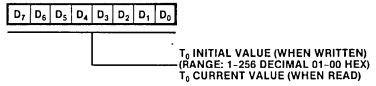
For example, to determine the opcode of a ADC instruction use the addressing modes r (destination) and Ir (source). The result is 13.

Addr Mode dst	Addr Mode src	Lower Opcode Nibble
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7

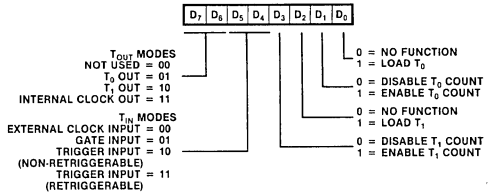
R240 SIO
Serial I/O Register
($F0_H$; Read/Write)



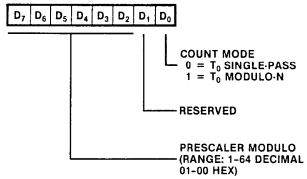
R244 T0
Counter/Timer 0 Register
($F4_H$; Read/Write)



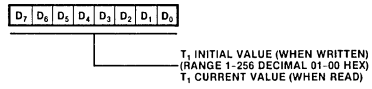
R241 TMR
Timer Mode Register
($F1_H$; Read/Write)



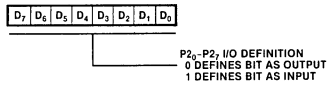
R245 PRE0
Prescaler 0 Register
($F5_H$; Write Only)



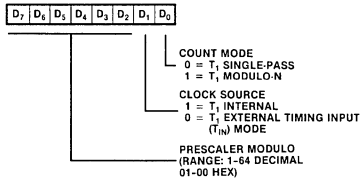
R242 T1
Counter Timer 1 Register
($F2_H$; Read/Write)



R246 P2M
Port 2 Mode Register
($F6_H$; Write Only)



R243 PRE1
Prescaler 1 Register
($F3_H$; Write Only)



R247 P3M
Port 3 Mode Register
($F7_H$; Write Only)

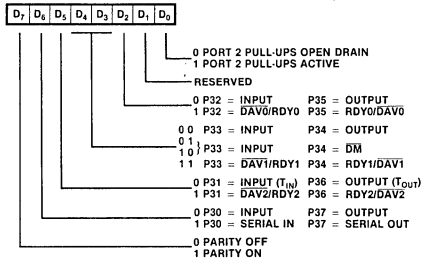
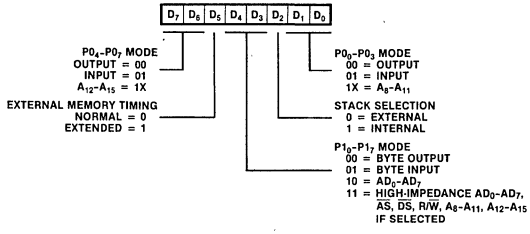
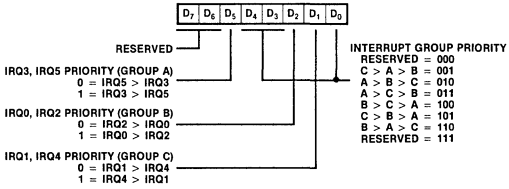


Figure 13. Control Registers

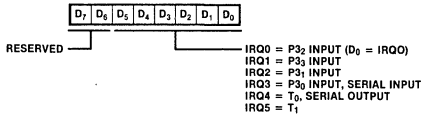
R248 P01M
Port 0 and 1 Mode Register
($F8_H$; Write Only)



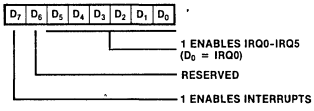
R249 IPR
Interrupt Priority Register
($F9_H$; Write Only)



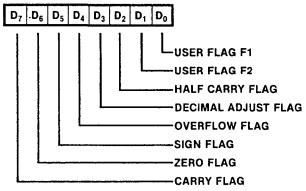
R250 IRQ
Interrupt Request Register
(FA_H; Read/Write)



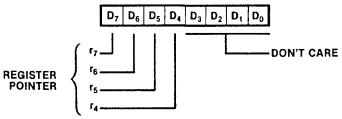
R251 IMR
Interrupt Mask Register
(FB_H; Read/Write)



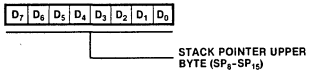
R252 FLAGS
Flag Register
(FC_H; Read/Write)



R253 RP
Register Pointer
(FD_H; Read/Write)



R254 SPH
Stack Pointer
(FE_H; Read/Write)



R255 SPL
Stack Pointer
(FF_H; Read/Write)

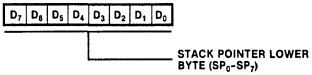
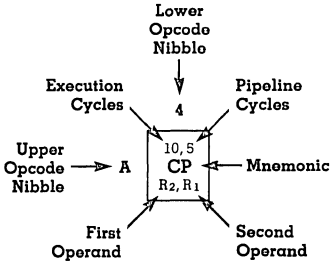


Figure 13. Control Registers (Continued)

Opcode Map

Upper Nibble (Hex) \ Lower Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6,5 **DEC** R1	6,5 **DEC** IR1	6,5 **ADD** r1, r2	6,5 **ADD** r1, Ir2	10,5 **ADD** R2, R1	10,5 **ADD** IR2, R1	10,5 **ADD** R1, IM	10,5 **ADD** IR1, IM	6,5 **LD** r1, R2	6,5 **LD** r2, R1	12/10,5 **DJNZ** r1, RA	12/10,0 **JR** cc, RA	6,5 **LD** r1, IM	12/10,0 **JP** cc, DA	6,5 **INC** r1	
1	6,5 **RLC** R1	6,5 **RLC** IR1	6,5 **ADC** r1, r2	6,5 **ADC** r1, Ir2	10,5 **ADC** R2, R1	10,5 **ADC** IR2, R1	10,5 **ADC** R1, IM	10,5 **ADC** IR1, IM								
2	6,5 **INC** R1	6,5 **INC** IR1	6,5 **SUB** r1, r2	6,5 **SUB** r1, Ir2	10,5 **SUB** R2, R1	10,5 **SUB** IR2, R1	10,5 **SUB** R1, IM	10,5 **SUB** IR1, IM								
3	8,0 **JP** IRR1	6,1 **SRP** IM	6,5 **SBC** r1, r2	6,5 **SBC** r1, Ir2	10,5 **SBC** R2, R1	10,5 **SBC** IR2, R1	10,5 **SBC** R1, IM	10,5 **SBC** IR1, IM								
4	8,5 **DA** R1	8,5 **DA** IR1	6,5 **OR** r1, r2	6,5 **OR** r1, Ir2	10,5 **OR** R2, R1	10,5 **OR** IR2, R1	10,5 **OR** R1, IM	10,5 **OR** IR1, IM								
5	10,5 **POP** R1	10,5 **POP** IR1	6,5 **AND** r1, r2	6,5 **AND** r1, Ir2	10,5 **AND** R2, R1	10,5 **AND** IR2, R1	10,5 **AND** R1, IM	10,5 **AND** IR1, IM								
6	6,5 **COM** R1	6,5 **COM** IR1	6,5 **TCM** r1, r2	6,5 **TCM** r1, Ir2	10,5 **TCM** R2, R1	10,5 **TCM** IR2, R1	10,5 **TCM** R1, IM	10,5 **TCM** IR1, IM								
7	10/12, 1 **PUSH** R2	12/14, 1 **PUSH** IR2	6,5 **TM** r1, r2	6,5 **TM** r1, Ir2	10,5 **TM** R2, R1	10,5 **TM** IR2, R1	10,5 **TM** R1, IM	10,5 **TM** IR1, IM								
8	10,5 **DECW** RR1	10,5 **DECW** IR1	12,0 **LDE** r1, Irr2	18,0 **LDEI** Ir1, Irr2												6,1 **DI**
9	6,5 **RL** R1	6,5 **RL** IR1	12,0 **LDE** Irr1	18,0 **LDEI** Ir2, Irr1												6,1 **EI**
A	10,5 **INCW** RR1	10,5 **INCW** IR1	6,5 **CP** r1, r2	6,5 **CP** r1, Ir2	10,5 **CP** R2, R1	10,5 **CP** IR2, R1	10,5 **CP** R1, IM	10,5 **CP** IR1, IM								14,0 **RET**
B	6,5 **CLR** R1	6,5 **CLR** IR1	6,5 **XOR** r1, r2	6,5 **XOR** r1, Ir2	10,5 **XOR** R2, R1	10,5 **XOR** IR2, R1	10,5 **XOR** R1, IM	10,5 **XOR** IR1, IM								16,0 **IRET**
C	6,5 **RRC** R1	6,5 **RRC** IR1	12,0 **LDC** r1, Irr2	18,0 **LDCI** Ir1, Irr2				10,5 **LD** r1, x, R2								6,5 **RCF**
D	6,5 **SRA** R1	6,5 **SRA** IR1	12,0 **LDC** r2, Irr1	18,0 **LDCI** Ir2, Irr1	20,0 **CALL*** IRR1		20,0 **CALL** DA	10,5 **LD** r2, x, R1								6,5 **SCF**
E	6,5 **RR** R1	6,5 **RR** IR1		6,5 **LD** r1, Ir2	10,5 **LD** R2, R1	10,5 **LD** IR2, R1	10,5 **LD** R1, IM	10,5 **LD** IR1, IM								6,5 **CCF**
F	8,5 **SWAP** R1	8,5 **SWAP** IR1		6,5 **LD** Ir1, r2		10,5 **LD** R2, IR1										6,0 **NOP**
Bytes per Instruction	2				3				2					3	1	



Legend:
R = 8-Bit Address
r = 4-Bit Address
R1 or r1 = Dst Address
R2 or r2 = Src Address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank areas are not defined.

*2-byte instruction; fetch cycle appears as a 3-byte instruction

Absolute Maximum Ratings

Voltages on all pins with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . See Ordering Information

Storage Temperature . . . . . -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

The DC characteristics listed below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the reference pin.

Standard conditions are:

- □ $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- □ GND = 0 V
- □ $0°C \leq T_A \leq +70°C$

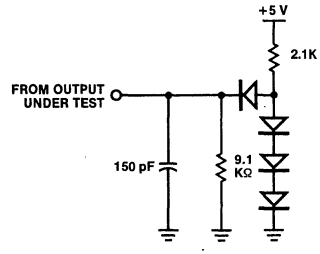


Figure 14. Test Load 1

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	3.8	V_{CC}	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.3	0.8	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{RH}	Reset Input High Voltage	3.8	V_{CC}	V	
V_{RL}	Reset Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0$ mA
I_{IL}	Input Leakage	-10	10	μA	$0\ V \leq V_{IN} \leq +5.25\ V$
I_{OL}	Output Leakage	-10	10	μA	$0\ V \leq V_{IN} \leq +5.25\ V$
I_{IR}	Reset Input Current		-50	μA	$V_{CC} = +5.25\ V$, $V_{RL} = 0\ V$
I_{CC}	V_{CC} Supply Current		150	mA	

External I/O or Memory Read and Write Timing

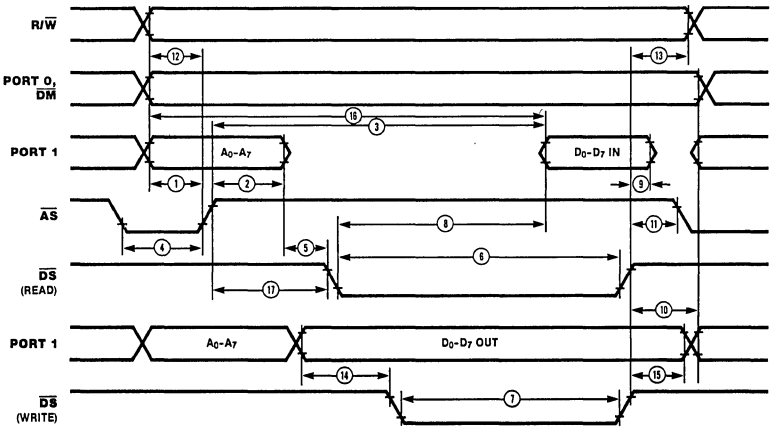


Figure 15. External I/O or Memory Read/Write

No.	Symbol	Parameter	8 MHz Min	8 MHz Max	12.5 MHz Min	12.5 MHz Max	Notes*†°
1	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	50		35		2,3
2	TdAS(A)	$\overline{AS}$ ↑ to Address Float Delay	60		45		2,3
3	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Required Valid		320		220	1,2,3
4	TwAS	$\overline{AS}$ Low Width	80		55		1,2,3
5	TdAz(DS)	Address Float to $\overline{DS}$ ↓	0		0		
6	TwDSR	$\overline{DS}$ (Read) Low Width	250		185		1,2,3
7	TwDSW	$\overline{DS}$ (Write) Low Width	160		110		1,2,3
8	TdDSR(DR)	$\overline{DS}$ ↓ to Read Data Required Valid		200		130	1,2,3
9	ThDR(DS)	Read Data to $\overline{DS}$ ↑ Hold Time	0		0		
10	TdDS(A)	$\overline{DS}$ ↑ to Address Active Delay	80		45		2,3
11	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	70		55		2.3
12	TdR/W(AS)	R/$\overline{W}$ Valid to $\overline{AS}$ ↑ Delay	50		30		2,3
13	TdDS(R/W)	$\overline{DS}$ ↑ to R/$\overline{W}$ Not Valid	60		35		2,3
14	TdDW(DSW)	Write Data Valid to $\overline{DS}$ (Write) ↓ Delay	50		35		2,3
15	TdDS(DW)	$\overline{DS}$ ↑ to Write Data Not Valid Delay	80		45		2,3
16	TdA(DR)	Address Valid to Read Data Required Valid		410		255	1,2,3
17	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	80		55		2,3

NOTES:
1. When using extended memory timing add 2 TpC.
2. Timing numbers given are for minimum TpC.
3. See clock cycle time dependent characteristics table.

† Test Load 1.
° All timing references use 2.0 V for a logic "1" and 0.8 V for a logic "0".
* All units in nanoseconds (ns).

Additional Timing Table

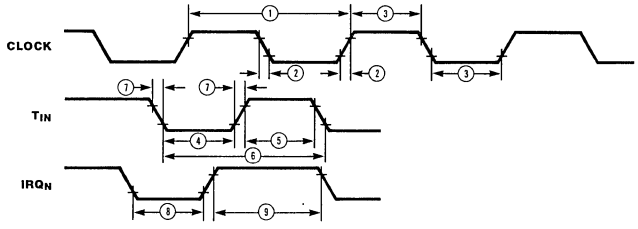


Figure 16. Additional Timing

No.	Symbol	Parameter	8 MHz Min	8 MHz Max	12.5 MHz Min	12.5 MHz Max	Notes*
1	TpC	Input Clock Period	125	1000	80	1000	1
2	TrC,TfC	Clock Input Rise And Fall Times		25		15	1
3	TwC	Input Clock Width	37		26		1
4	TwTinL	Time Input Low Width	100		70		2
5	TwTinH	Timer Input High Width	3TpC		3TpC		2
6	TpTin	Timer Input Period	8TpC		8TpC		2
7	TrTin,TfTin	Timer Input Rise And Fall Times		100		100	2
8a	TwIL	Interrupt Request Input Low Time	100		70		2,3
8b	TwIL	Interrupt Request Input Low Time		3TpC	3TpC		2,4
9	TwIH	Interrupt Request Input High Time		3TpC	3TpC		2,3

NOTES:
1. Clock timing references uses 3.8 V for a logic "1" and 0.8 V for a logic "0".
2. Timing reference uses 2.0 V for a logic "1" and 0.8 V for a logic "0".
3. Interrupt request via Port 3 ($P3_1$-$P3_3$).
4. Interrupt request via Port 3 ($P3_0$).
* Units in nanoseconds (ns).

Memory Port Timing

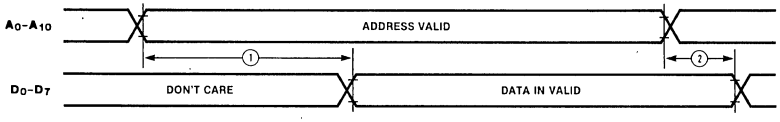


Figure 17. Memory Port Timing

No.	Symbol	Parameter	Min	Max	Notes*
1	TdA(DI)	Address Valid to Data Input Delay		320	1,2
2	ThDI(A)	Data In Hold time	0		1

NOTES:
1. Test Load 2.
2. This is a Clock-Cycle-Dependent parameter. For clock frequencies other than the maximum, use the following formula: 5 TpC – 95

*Units are nanoseconds unless otherwise specified.

Handshake Timing

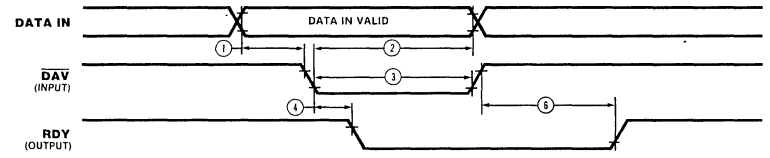


Figure 18a. Input Handshake

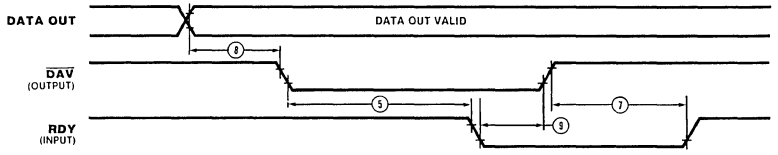


Figure 18b. Output Handshake

No.	Symbol	Parameter	Min	Max	Notes*
1	TsDI(DAV)	Data In Setup Time	0		
2	ThDI(DAV)	Data In Hold time	160		
3	TwDAV	Data Available Width	120		
4	TdDAVIf(RDY)	$\overline{DAV}$ ↓ Input to RDY ↓ Delay		120	1,2
5	TdDAVOf(RDY)	$\overline{DAV}$ ↓ Output to RDY ↓ Delay	0		1,3
6	TdDAVIr(RDY)	$\overline{DAV}$ ↑ Input to RDY ↑ Delay		120	1,2
7	TdDAV0r(RDY)	$\overline{DAV}$ ↑ Output to RDY ↑ Delay	0		1,3
8	TdDO(DAV)	Data Out to $\overline{DAV}$ ↓ Delay	30		1
9	TdRDY(DAV)	Rdy ↓ Input to $\overline{DAV}$ ↑ Delay	0	140	1

NOTES:
1. Test load 1
2. Input handshake
3. Output handshake

† All timing references use 2.0 V for a logic "1" and 0.8 V for a logic "0".

* Units in nanoseconds (ns).

Clock-Cycle-Time-Dependent Characteristics

Number	Symbol	Equation
1	TdA(AS)	TpC-50
2	TdAS(A)	TpC-40
3	TdAS(DR)	4TpC-110*
4	TwAS	TpC-30
5	TwDSR	3TpC-65*
7	TwDSW	2TpC-55*
8	TdDSR(DR)	3TpC-120*
10	Td(DS)A	TpC-40
11	TdDS(AS)	TpC-30
12	TdR/W(AS)	TpC-55
13	TdDS(R/W)	TpC-50
14	TdDW(DSW)	TpC-50
15	TdDS(DW)	TpC-40
16	TdA(DR)	5TpC-160*
17	TdAS(DS)	TpC-30

*Add 2TpC when using extended memory timing.

PRELIMINARY PRODUCT SPECIFICATION

Z8602

NMOS Z8® 8-BIT MCU KEYBOARD CONTROLLER

FEATURES

- 8-bit microcontroller, 40-pin DIP package
- Low cost
- +4.75 to +5.25 Vcc range
- Low power consumption - 750mW
- Fast instruction pointer - 1.5 microsecond @ 4 MHz
- 32 input/output lines
- All digital inputs NMOS levels
- 2K bytes ROM
- 124 bytes of RAM
- Full Duplex Serial/Parallel Port
- Two programmable 8-bit Counter/Timers each with 6-bit programmable prescaler
- Six vectored, priority interrupts from eight different sources
- Clock speed up to 4 MHz
- On-chip oscillator that accepts a crystal, ceramic resonator or external clock drive
- Low EMI emission

DESCRIPTION

The Z8602 Keyboard Controller (KBC) introduces a new level of sophistication to single-chip architecture. The Z8602 is a member of the Z8 single-chip microcomputer family with 2K bytes of ROM.

The Z8602 KBC is housed in a 40-pin DIP, and is manufactured in NMOS technology. Zilog's microcontroller offers fast execution, more efficient use of memory, more sophisticated interrupt, input/output bit-manipulation capabilities, and easy hardware/software system expansion along with low cost and low power consumption.

The KBC architecture is characterized by a flexible I/O scheme, an efficient register, I/O, and a number of ancillary features that are useful in many industrial and advanced scientific applications.

The device applications demand powerful I/O capabilities. The KBC fulfills this with 32-pins dedicated to input and output. These lines are grouped into four ports, each port consists of 8 lines, and are configurable under software control to provide timing, status signals, and serial or parallel I/O ports.

The Z8602 offers Low EMI emission and is achieved by means of several modifications in the output drivers and clock circuitry of the device.

There are two basic address spaces which are available to support this wide range of configurations: Program Memory and 124 General-Purpose Registers.

The KBC offers two on-chip counter/timers with a large number of user selectable modes, and an asynchronous receiver/transmitter (UART). This unburdens the program from coping with real-time problems such as counting/timing and serial data communications. Hardware support for the UART is minimized because one of the on-chip timers supplies the bit rate (Figure 1).

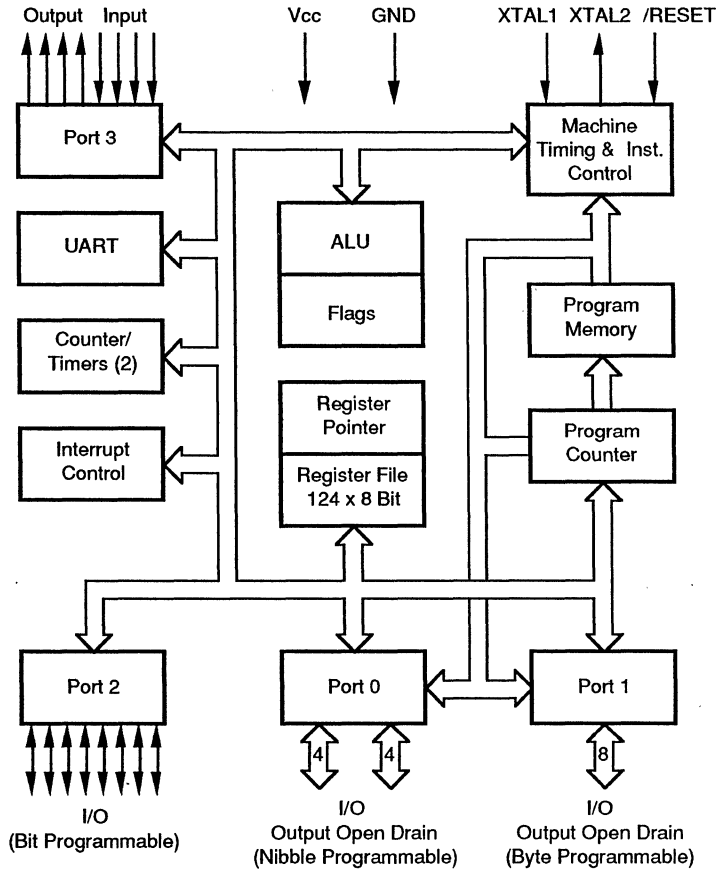


Figure 1. Functional Block Diagram

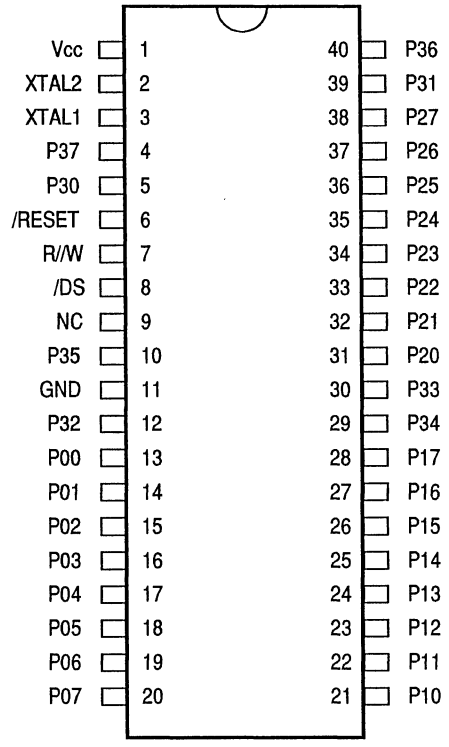


Figure 2. Pin Configuration

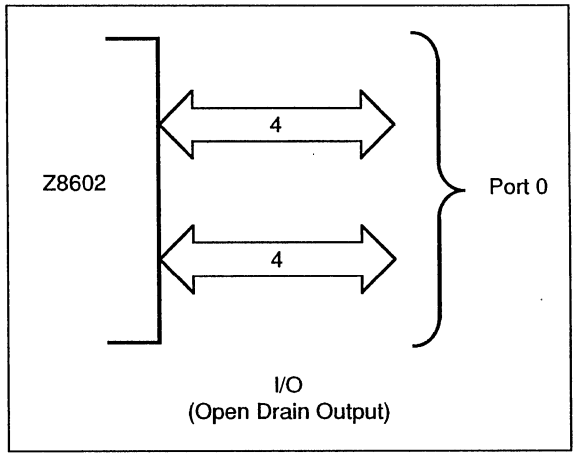
PIN IDENTIFICATION

Pin #	Symbol	Function	Direction
1	Vcc	Power Supply	Input
2	XTAL2	Crystal Oscillator Clock	Output
3	XTAL1	Crystal Oscillator Clock	Input
4	P37	Port 3 pin 7	Output
5	P30	Port 3 pin 0	Input
6	/RESET	Reset	Input
7	NC	Not Connecting	
8	NC	Not Connecting	
9	NC	Not Connecting	
10	P35	Port 3 pin 5	Output
11	GND	Ground, V_{ss}	Input
12	P32	Port 3 pin 2	Input
13-20	P00-7	Port 0 pin 0,1,2,3,4,5,6,7	In/Output
21-28	P10-7	Port 1 pin 0,1,2,3,4,5,6,7	In/Output
29	P34	Port 3 pin 4	Output
30	P33	Port 3 pin 3	Input
31-38	P20-7	Port 2 pin 0,1,2,3,4,5,6,7	In/Output
39	P31	Port 3 pin 1	Input
40	P36	Port 3 pin 6	Output

PIN FUNCTIONS

XTAL1, XTAL2. *Crystal 1, Crystal 2* (time-based input and output, respectively). These pins connect a parallel-resonant crystal or an external single-phase clock to the on-chip clock oscillator and buffer.

Port 0 P07-P00. Port 0 is an 8-bit, nibble programmable, bidirectional, NMOS compatible I/O port. These 8 I/O lines can be configured under software control as a nibble input port, or as a nibble open drain output port. When used as an I/O port, inputs are standard NMOS and outputs are open drain (Figure 3).



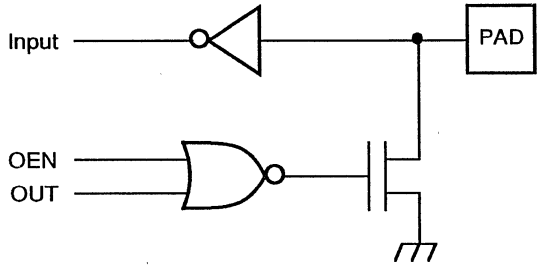


Figure 3. Port 0 Configuration

Port 1 P17-P10. Port 1 is an 8-bit, byte programmable, bidirectional, NMOS compatible I/O port. These 8 I/O lines are configured under software control program as byte input port or as an open drain output port. When used as an I/O port, inputs are standard NMOS and outputs are open drain (Figure 4).

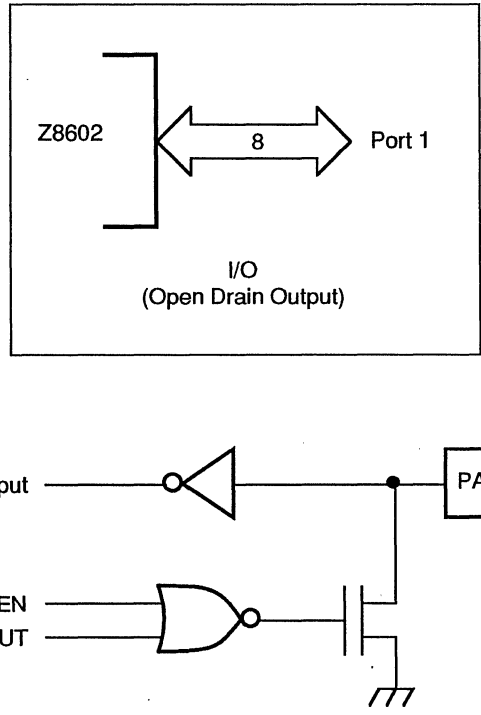


Figure 4. Port 1 Configuration

Port 2 P27-P20. Port 2 is an 8-bit, bit programmable, bidirectional, NMOS compatible I/O port. These 8 I/O lines are configured under the software control program for I/O. Port 2 can be programmed as bit-by-bit independently, as input or output, or configured to provide open-drain outputs (Figure 5).

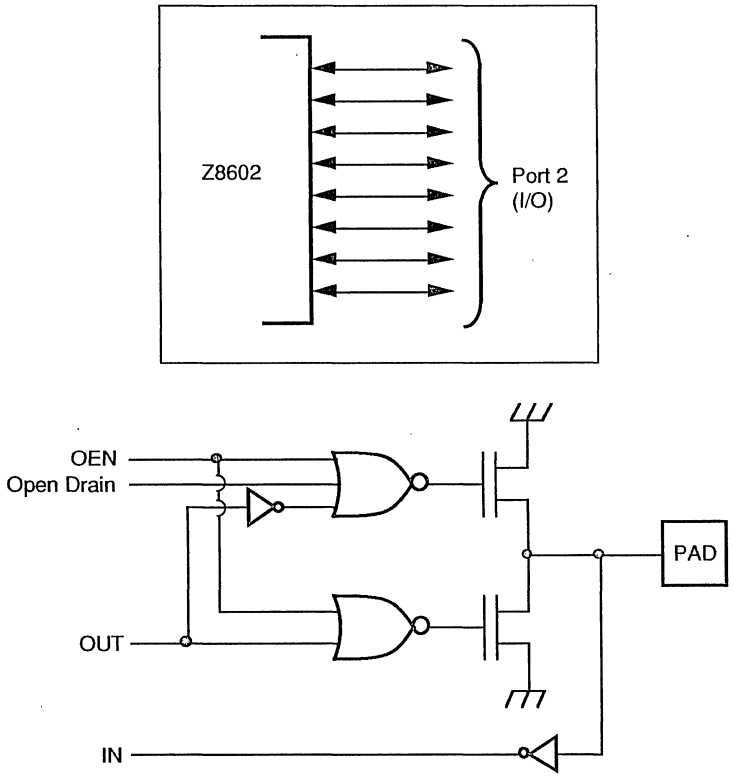
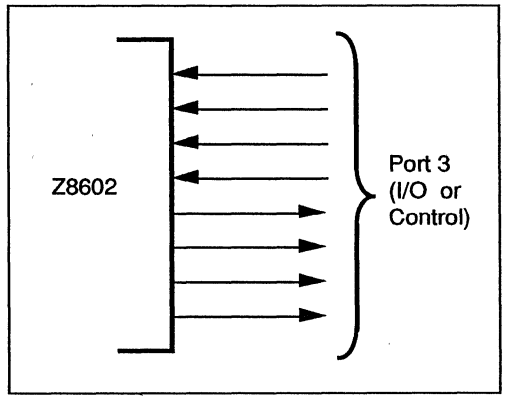
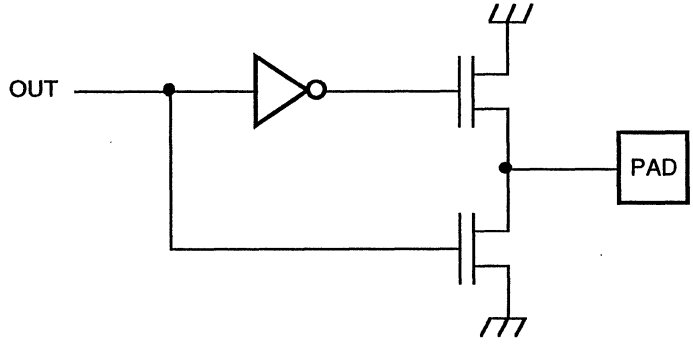


Figure 5. Port 2 Configuration

Port 3 P37-P30. Port 3 is an 8-bit, NMOS compatible four-fixed-input and four-fixed-output I/O port. These 8 I/O lines have four-fixed-input (P33 P30) and four-fixed-output (P37-P34) ports. Port 3, when used as serial I/O, are programmed as serial in and serial out, respectively. Port 3 outputs have the capability of driving LED's directly with a pull-up resistor (output voltage or port 3 is 0.8V @ 10mA).

Port 3 is configured under software control to provide the following control functions: four external interrupt request signals (IRQ3-IRQ0); timer input and output signals (Tin and Tout - Figure 6).





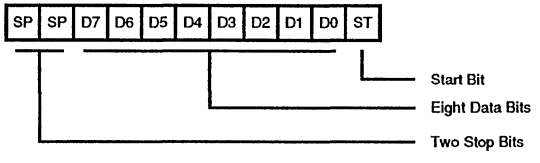
(a) Port 3 P34-P37

(b) Port 3 P30-P33

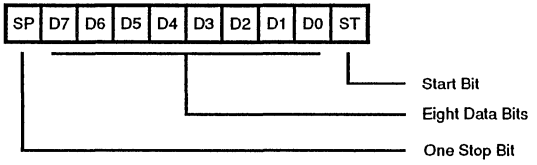
Figure 6. Port 3 Configuration

Port 3 lines, P30 and P37, are programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by Counter/Timer 0, with a maximum rate of 60K bits/second at 4MHz.

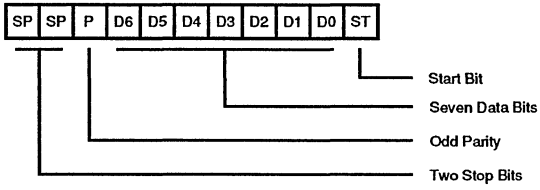
The KBC automatically adds a start bit and two stop bits to transmitted data (Figure 7). Odd parity is also available as an option. 8 data bits are always transmitted, regardless of parity selection. If parity is enabled, the 8th bit is the odd parity bit. An interrupt request (IRQ4) is generated on all transmitted characters.



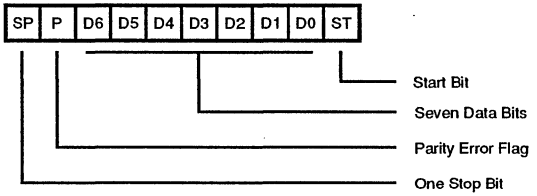
(a) Transmitted Data (No Parity)



(c) Received Data (No Parity)



(b) Transmitted Data (With Parity)



(d) Received Data (With Parity)

Figure 7. Serial Data Formats

Receive data must have a start bit, 8 data bits and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ3 interrupt request.

/RESET (input, active Low). When activated, /RESET initializes the 8602. When /RESET is deactivated, program execution begins from the internal program location at 000C (HEX).

SPECIAL FUNCTIONS

The device incorporates special functions to enhance Zilog's Z8 applications as a keyboard controller, scientific research and advanced technologies applications.

Program Memory. The 16-bit program counter address 64K bytes of program memory space at internal locations (Figure 8).

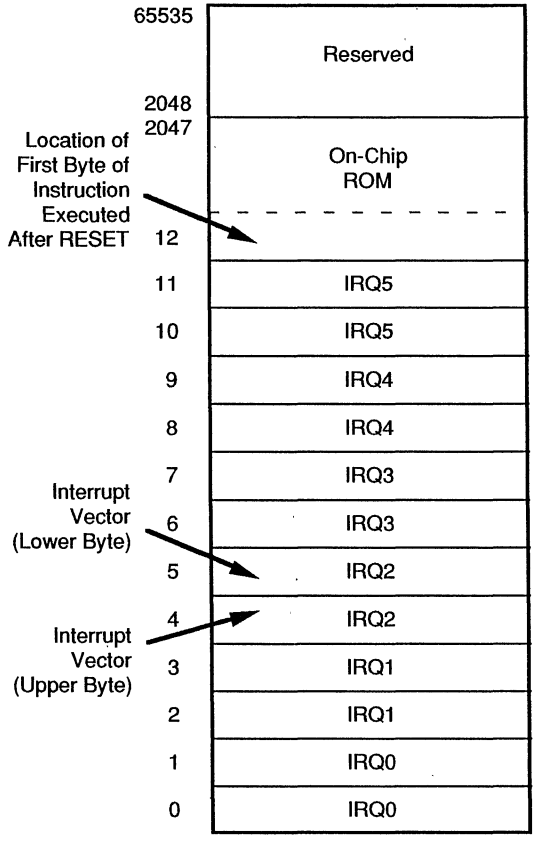


Figure 8. Program Memory Map

The first 12 bytes of program memory are reserved for the interrupt vectors. These locations have six 16-bit vectors that correspond to the six available interrupts.

Byte 13 to byte 2047 consists of on-chip, mask-programmed ROM. Addresses 2048 and greater are reserved.

Register File. The register file (Figure 9) consists of 4 I/O port registers, 124 general-purpose registers and 16 control and status registers (R3-R0, R127-R4, and R255-R240 respectively). The instructions can access registers directly or indirectly via an 8-bit address field. This allows short, 4-bit register addressing using the Register Pointer (Figure 10). In the 4-bit mode, the register file is divided into 9 working-register groups, each occupying 16 continuous locations. The Register Pointer addresses the starting location of the active working-register group.

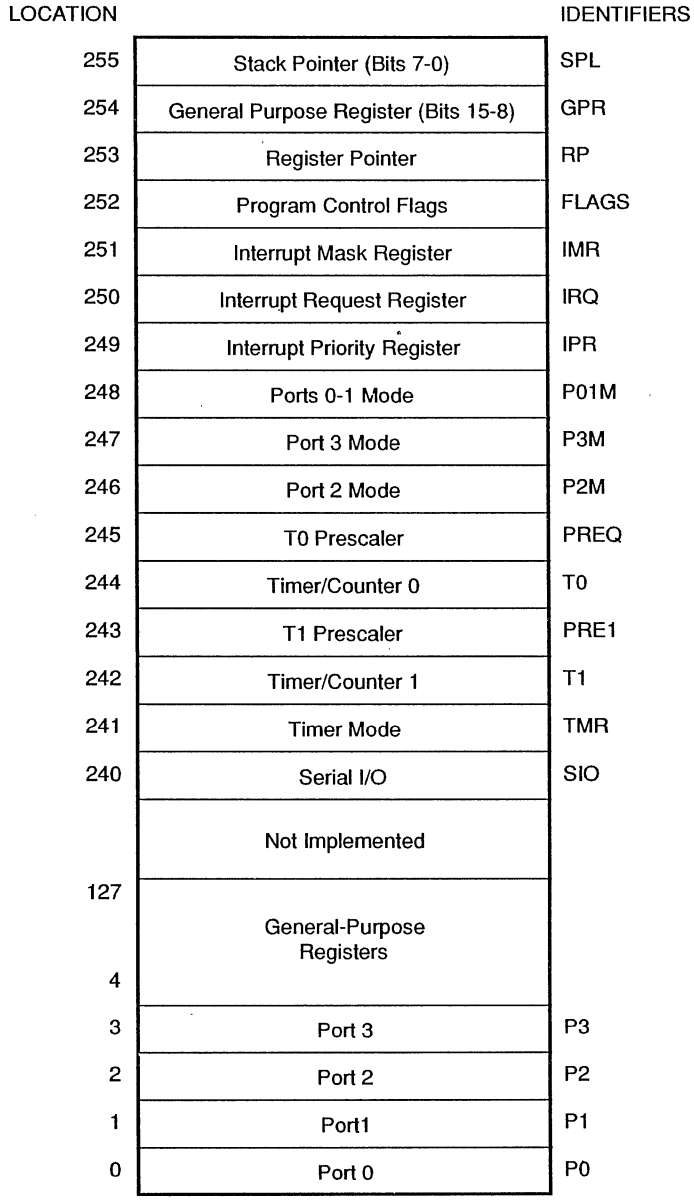


Figure 9. Register File Configuration

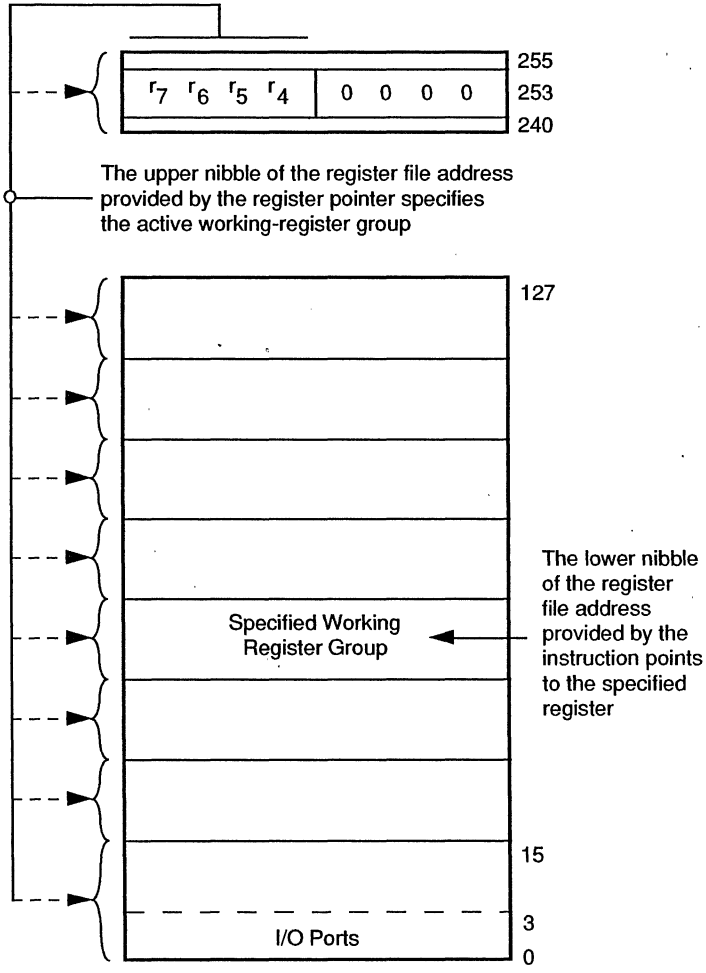


Figure 10. Register Pointer Configuration

Stack. The KBC internal register files are used for the stack. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 124 general-purpose registers.

Counter/Timers. There are two 8-bit programmable counter/timers (T0-T1), each driven by its own 6-bit programmable prescaler. The T1 prescaler can be driven by internal or external clock sources however, the T0 prescaler is driven by the internal clock only (Figure 11).

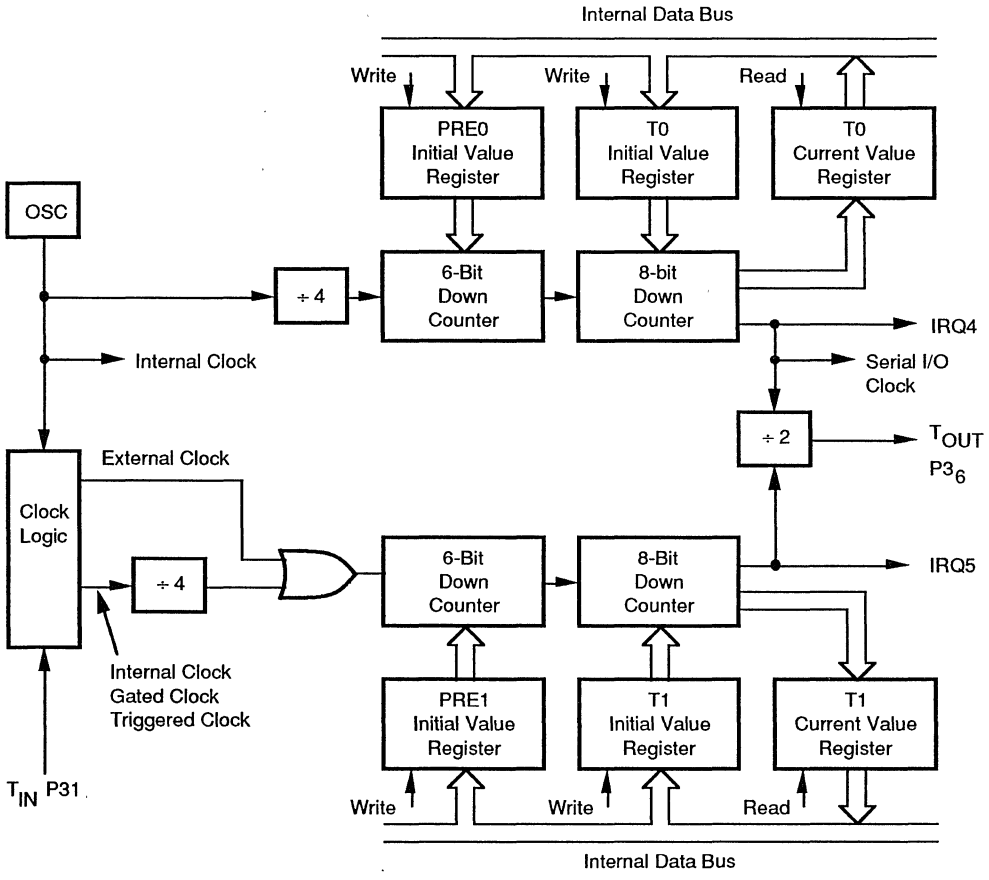


Figure 11. Counter/Timers Block Diagram

The 6-bit prescalers can divide the input frequency of the clock source by any integer number from 1 to 64. Each prescaler drives its own counter, which decrements the value (1 to 256) that has been loaded into the counter. When both the counter and prescaler reach the end of count, a timer interrupt request, IRQ4 (T0) or IRQ5 (T1), is generated.

The counter can be programmed to start, stop, restart to continue, or restart from the initial value. The counters can also be programmed to stop upon reaching zero (single pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode).

The counters, but not the prescalers, are read at any time without disturbing their value or count mode. The clock source for T1 is user-definable and are either the internal microprocessor clock divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input as an external clock, a trigger input that can be retriggerable or non-retrigerable, or as a gate input for the internal clock. The counter/timers can be programmably cascaded by connecting the T0 output to the input of T1. Port 3 line P36 also serves as a timer output (Tout) through which T0, T1 or the internal clock are output.

Interrupts. The KBC have six different interrupts from eight different sources. These interrupts are maskable and prioritized (Figure 12). The 8 sources are divided as follow: 4 sources are claimed by Port 3 lines P33-P30, one in Serial Out, one in Serial In, and 2 in counter/timers. The Interrupt Masked Register globally or individually enables or disables the six interrupts requests.

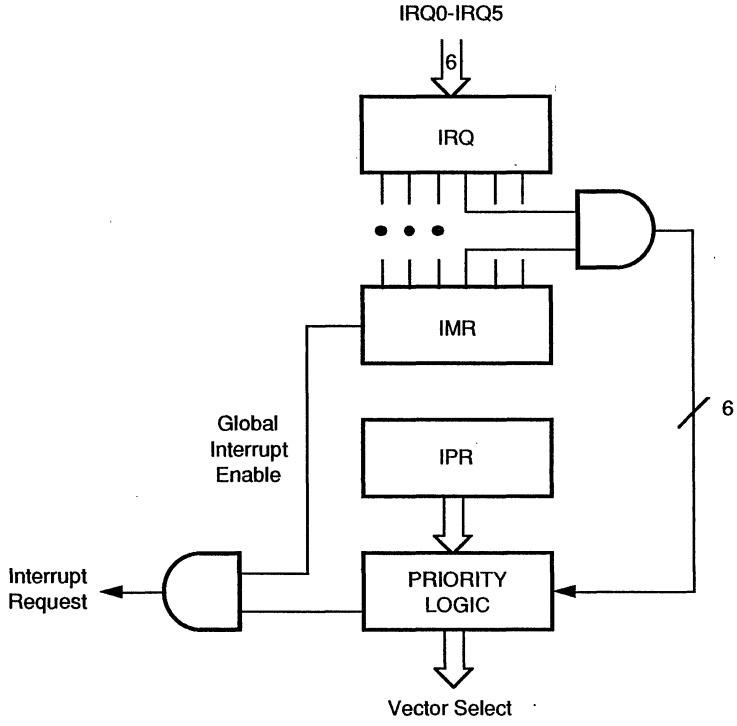


Figure 12. Interrupt Block Diagram

When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register. All interrupts are vectored through locations in the program memory. When an interrupt machine cycle is activated an interrupt request is granted. Thus, this disables all of the subsequent interrupts, saves the Program Counter and status flags, and then branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

To accommodate polled interrupt systems, interrupt inputs are masked and the interrupt request register is polled to determine which of the interrupt requests needs service.

Clock. The KBC on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal, ceramic resonator, or any suitable external clock source (XTAL1= Input, XTAL2= Out Put). The internal clock oscillates at the crystal frequency. The crystal should be AT cut, 4MHz max, with a series resistance (RS) less than or equal to 100 Ohms.

The crystal should be connected across XTAL1 and XTAL2 using the recommended capacitors from each pin to ground (Figure 13). Capacitance is between 15pF to 40pF depending upon the manufacturer of crystal, ceramic resonator and PCB layout.

EMI. The Z8602 offers low EMI emission due to circuit modifications to improve EMI performance. The internal divide-by-two circuit has been removed to improve EMI performance.

The EMI measurements for the KBC are done in a closed field environment (e.g., indoor, at room temperature) with 5.1V applied to the Vcc, where the EMI probe is positioned directly above the Z8602. Far field EMI measurements should be conducted at an enclosed, shielded facility.

Figures 14 through 17 show the EMI plots for the Z8602 running at 2 and 4 MHz crystals with 40 and 100 MHz frequency span.

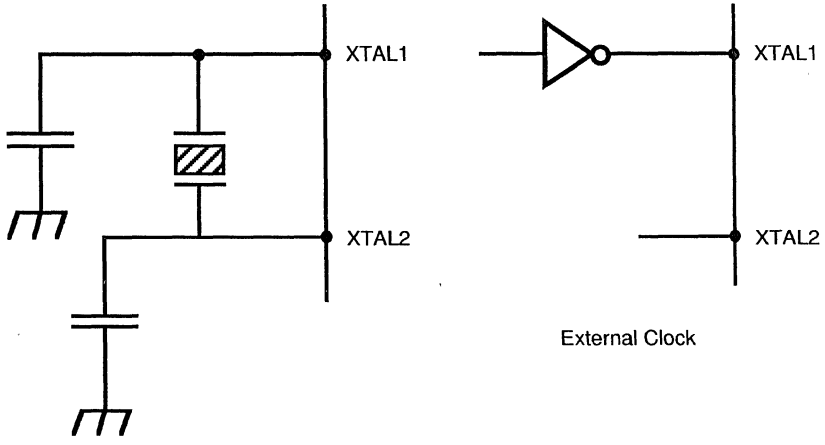


Figure 13. Oscillator Configuration

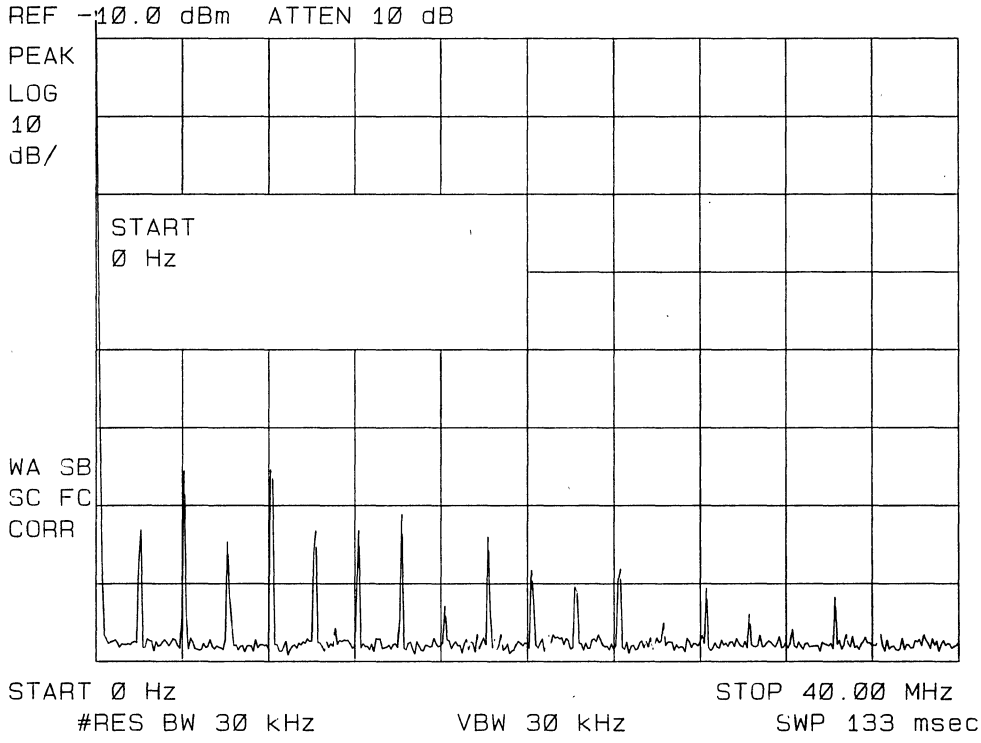


Figure 14. Zilog PC Board W/2 MHz Crystal

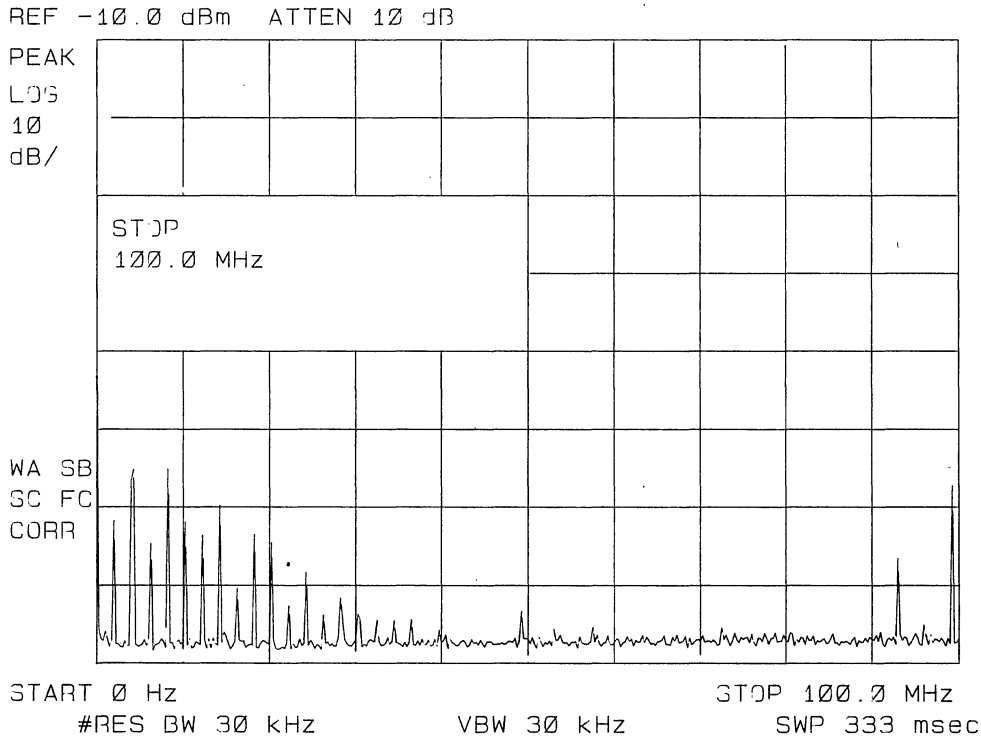


Figure 15. Z8602 PC Board W/2 Mhz Crystal

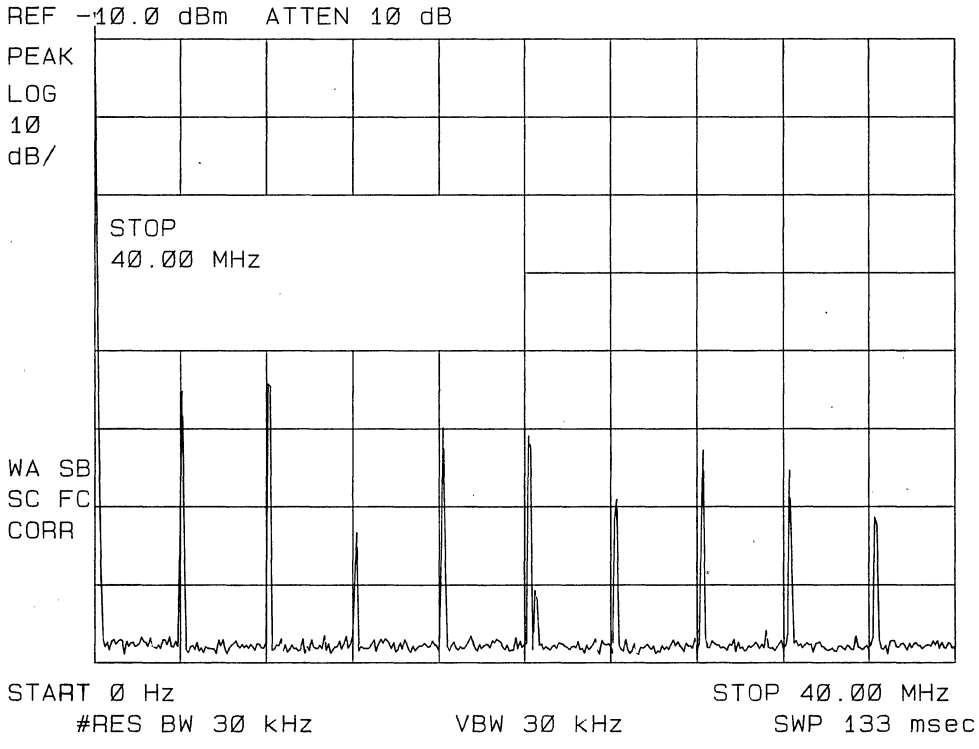


Figure 16. Z8602 PC Board W/4 MHz Crystal

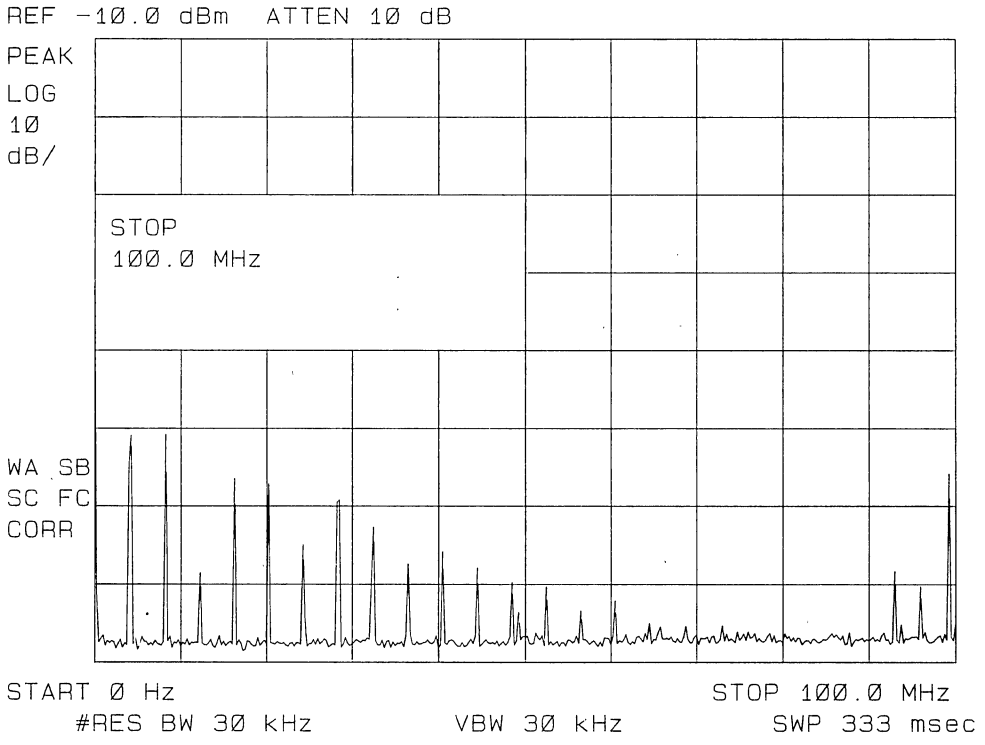


Figure 17. Z8602 W/4 MHz Crystal

STANDARD TEST CONDITIONS

Standard Test Conditions. The characteristics listed here apply for standard test conditions as noted. All voltages are referenced to GND. Positive current flows into the referenced pin (Figure 18).

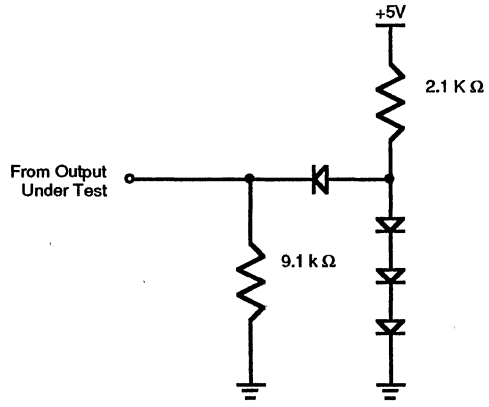


Figure 18. Test Load Diagram

ABSOLUTE MAXIMUM RATINGS

Symbol	Description	Min	Max	Units
V_{cc}	Supply Voltage(*)	-0.3	+7.0	V
TSTG	Storage Temp	-65	+150	C
TA	Oper Ambient Temp	See ordering information		

(*) Voltage on all pins with respect to GND.

Stress greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC CHARACTERISTICS
V_{CC} = 4.75 V to 5.25V @ 0°C to +70°C

Symbol	Parameter	Min	Max	Typ†	Unit	Condition
V_{CH}	Clock Input High Voltage	3.8	V_{CC}		V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.3	0.8		V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}		V	
V_{IL}	Input Low Voltage	-0.3	0.8		V	
V_{RH}	Reset Input High Voltage	3.8	V_{CC}		V	
V_{RL}	Reset Input Low Voltage	-0.3	0.8		V	
V_{OH}	Output High Voltage	2.0			V	I_{OH}= -250 µA (Port 2 only)
	Output High Voltage	2.4			V	I_{OH}= -250 µA (Port 3 only)
V_{OL}	Output Low Voltage		0.8		V	I_{OL}= +2.0 mA (Port 0,1,2)
	Output Low Voltage		0.4		V	I_{OL}= +2.0 mA (Port 3 only)
I_{IL}	Input Leakage	-10	10		µA	V_{IN} = 0V, 5.25V
I_{OL}	Output Leakage	-10	10		µA	V_{IN} = 0V, 5.25V
I_{IR}	Reset Input Current		-50		µA	V_{IN} = 0V, 5.25V
I_{CC}	V_{CC}Supply Current		150	135	mA	

† Typical @ 25°C

CONTROL REGISTERS

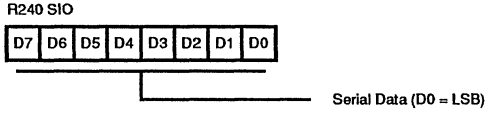


Figure 19. Serial I/O Register
(F0H; Read/Write)

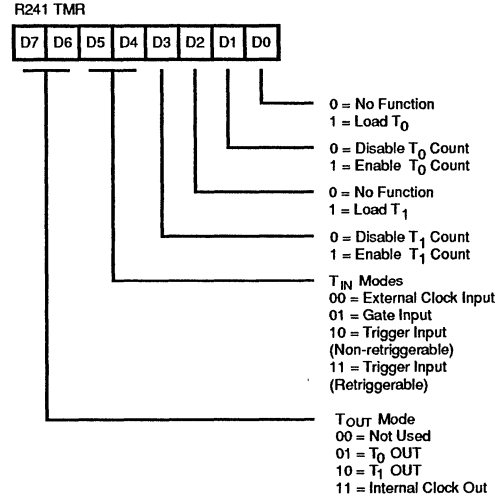


Figure 20. Timer Mode Register
(F1H; Read/Write)

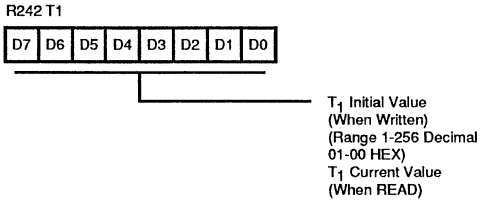


Figure 21. Counter Timer 1 Register
(F2H; Read/Write)

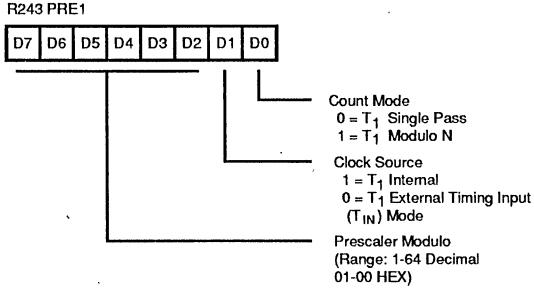


Figure 22. Prescaler 1 Register
(F3H; Write Only)

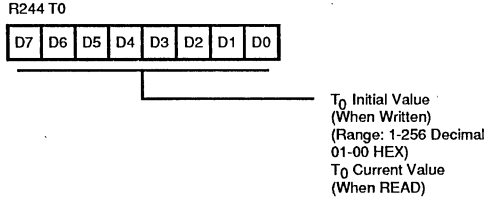


Figure 23. Counter/Timer 0 Register
(F4H; Read/Write)

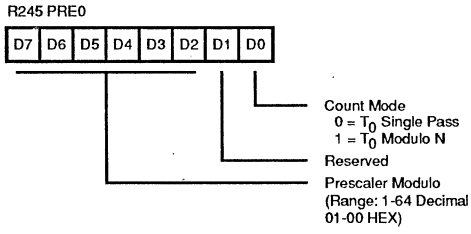


Figure 24. Prescaler 0 Register
(F5H; Write Only)

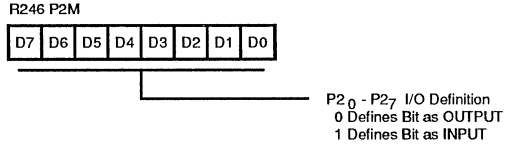


Figure 25. Port 2 Mode Register
(F6H; Write Only)

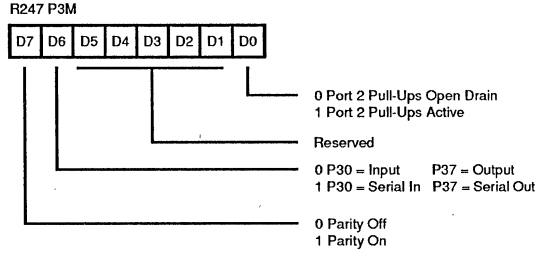


Figure 26. Port 3 Mode Register
(F7H; Write Only)

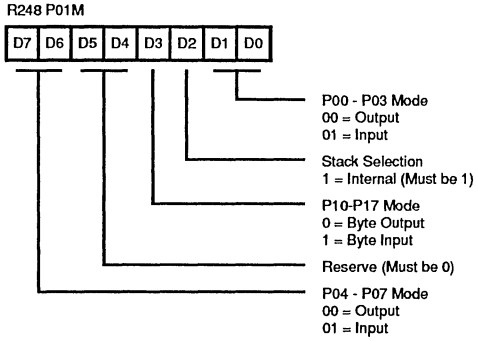


Figure 27. Port 0 and 1 Mode Register
(F8H; Write Only)

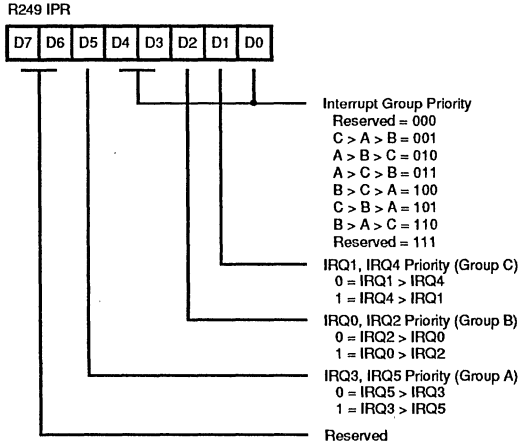


Figure 28. Interrupt Priority Register
(F9H; Write Only)

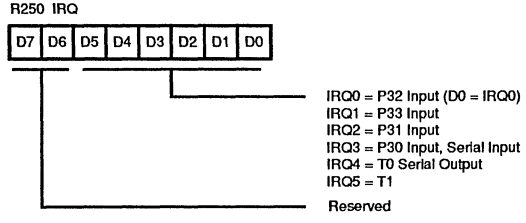


Figure 29. Interrupt Request Register
(FAH; Read/Write)

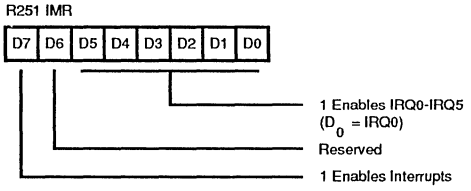


Figure 30. Interrupt Mask Register
(FBH; Read/Write)

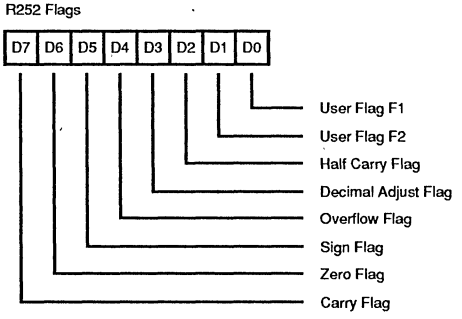


Figure 31. Flag Register
(FCH; Read/Write)

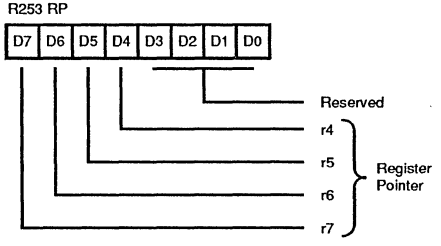


Figure 32. Register Pointer
(FDH; Read/Write)

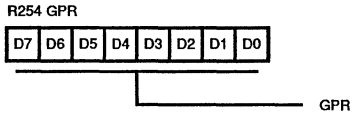


Figure 33. General Purpose Register
(FEH; Read/Write)

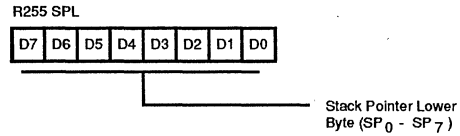


Figure 34. Stack Pointer
(FFH; Read/Write)

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

Symbol	Meaning
IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

Symbol	Meaning
dst	Destination location or contents
src	Source location or contents
cc	Condition code
@	Indirect address prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag register (Control Register 252)
RP	Register Pointer (R253)
IMR	Interrupt mask register (R251)

Flags. Control register (R252) contains the following six flags:

Symbol	Meaning
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

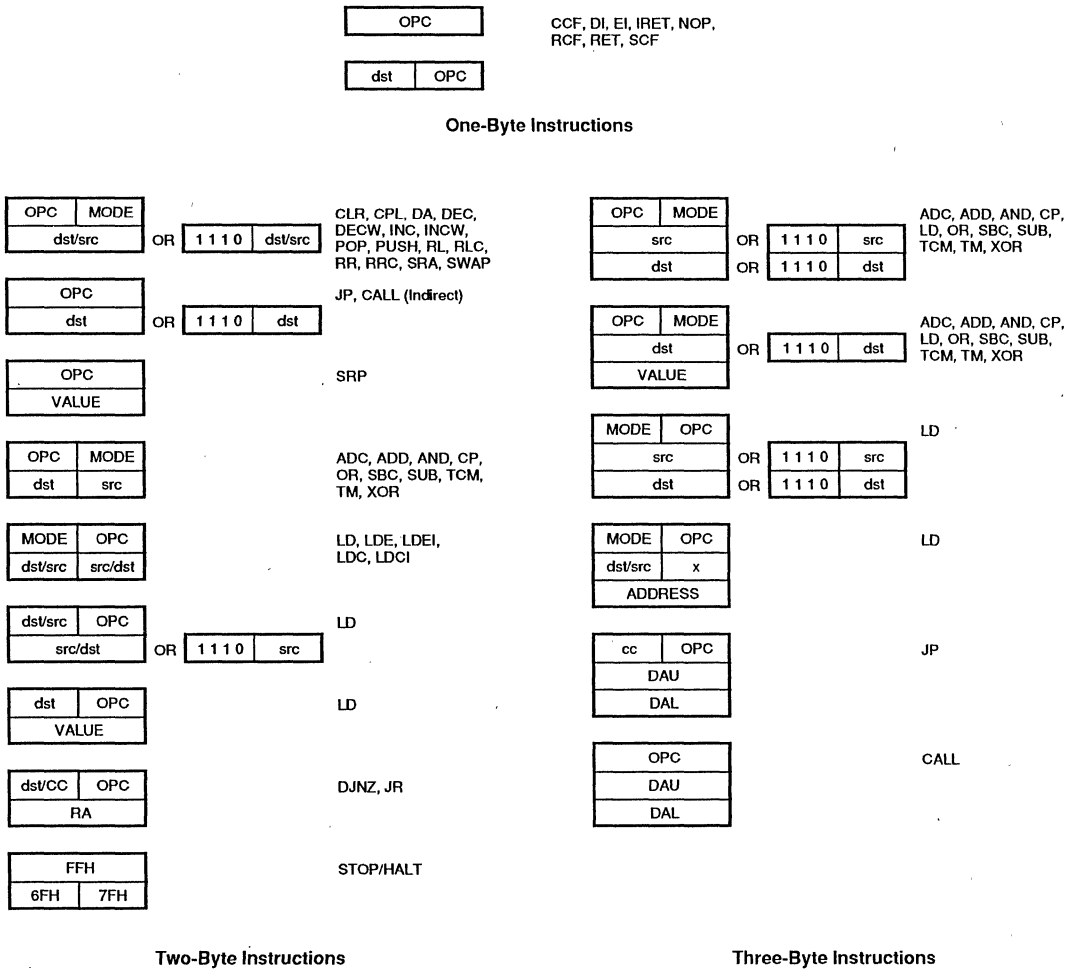
Affected flags are indicated by:

0	Clear to zero
1	Set to one
*	Set to clear according to operation
-	Unaffected
x	Undefined

Condition Codes

Value	Mnemonic	Meaning	Flags Set
1000		Always True	
0111	C	Carry	C = 1
1111	NC	No Carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not Zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No Overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not Egual	Z = 0
1001	GE	Greater Than or Equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater Than	[Z OR (S XOR V)] = 0
0010	LE	Less Than or Equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned Greater Than or Equal	C = 0
0111	ULT	Unsigned Less Than	C = 1
1011	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned Less Than or Equal	(C OR Z) = 1
0000		Never True	

INSTRUCTION FORMATS



INSTRUCTION SUMMARY

Note: Assignment of a value is indicated by the symbol "—". For example:

dst — dst + src

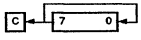
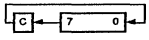
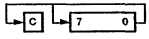
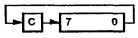
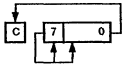
indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr (n)" is used to refer to bit (n) of a given operand location.

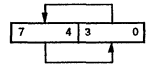
INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst, src dst←dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst←dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst←dst AND src	†		5[ ]	-	*	*	0	-	-
CALL dst SP←SP - 2 @SP←PC, PC←dst	DA IRR		D6 D4	-	-	-	-	-	-
CCF C←NOT C			EF	*	-	-	-	-	-
CLR dst dst←0	R IR		B0 B1	-	-	-	-	-	-
COM dst dst←NOT dst	R IR		60 61	-	*	*	0	-	-
CP dst, src dst - src	†		A[ ]	*	*	*	*	-	-
DA dst dst←DA dst	R IR		40 41	*	*	*	X	-	-
DEC dst dst←dst - 1	R IR		00 01	-	*	*	*	-	-
DECW dst dst←dst - 1	RR IR		80 81	-	*	*	*	-	-
DI IMR(7)←0			8F	-	-	-	-	-	-
DJNZr, dst r←r - 1 if r ≠ 0 PC←PC + dst Range: +127, -128	RA		rA r = 0 - F	-	-	-	-	-	-
EI IMR(7)←1			9F	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
INC dst dst←dst + 1	r R IR		rE r = 0 - F 20 21	-	*	*	*	-	-
INCW dst dst←dst + 1	RR IR		A0 A1	-	*	*	*	-	-
IRET FLAGS←@SP; SP←SP + 1 PC←@SP; SP←SP + 2; IMR(7)←1			BF	*	*	*	*	*	*
JP cc, dst if cc is true PC←dst	DA IRR		cD c = 0 - F 30	-	-	-	-	-	-
JR cc, dst if cc is true, PC←PC + dst Range: +127, -128	RA		cB c = 0 - F	-	-	-	-	-	-
LD dst, src dst←src	r r R r X r r Ir R R IR IR	Im R r X r Ir r R IR IM IM R	rC r8 r9 r = 0 - F C7 D7 E3 F3 E4 E5 E6 E7 F5	-	-	-	-	-	-
LDC dst, src	r	Irr	C2	-	-	-	-	-	-
LDCI dst, src r←r +1; rr←rr + 1	Ir	Irr	C3	-	-	-	-	-	-

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
NOP			FF	-	-	-	-	-	-
OR dst, src dst←dst OR src	†		4[ ]	-	*	*	0	-	-
POP dst dst←@SP; SP←SP + 1	R IR		50 51	-	-	-	-	-	-
PUSH src SP←SP - 1; @SP←src		R IR	70 71	-	-	-	-	-	-
RCF C←0			CF	0	-	-	-	-	-
RET PC←@SP; SP←SP + 2			AF	-	-	-	-	-	-
RL dst 	R IR		90 91	*	*	*	*	-	-
RLC dst 	R IR		10 11	*	*	*	*	-	-
RR dst 	R IR		E0 E1	*	*	*	*	-	-
RRC dst 	R IR		C0 C1	*	*	*	*	-	-
SBC dst, src dst←dst←src←C	†		3[ ]	*	*	*	*	1	*
SCF C←1			DF	1	-	-	-	-	-
SRA dst 	R IR		D0 D1	*	*	*	0	-	-

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
SRP src RP←src		Im	31	-	-	-	-	-	-
SUB dst, src dst←dst←src	†		2[ ]	*	*	*	*	1	*
SWAP dst 	R IR		F0 F1	X	*	*	X	-	-
TCM dst, src (NOT dst) AND src	†		6[ ]	-	*	*	0	-	-
TM dst, src dst AND src	†		7[ ]	-	*	*	0	-	-
XOR dst, src dst←dst XOR src	†		B[ ]	-	*	*	0	-	-

† These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

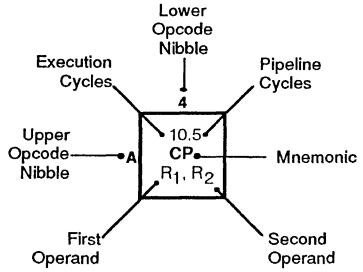
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address Mode dst	Address Mode src	Lower Opcode Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

OPCODE MAP

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								
8	10.5 **DECW** RR1	10.5 **DECW** IR1														DI
9	6.5 **RL** R1	6.5 **RL** IR1														6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14.0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2				10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1			20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6.5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2			3					2					3	1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_2 = Dst address
R_1 or r_2 = Src address

Sequence:
Opcode, First Operand, Second Operand

Note: The blanks are not defined.

* 2-byte instruction appears as a 3-byte instruction

PRELIMINARY PRODUCT SPECIFICATION

Z8604

NMOS Z8® 8-BIT MICROCONTROLLER

FEATURES

- 8-bit NMOS Microcomputer, 18-pin DIP
- Low Cost
- 4.5 to 5.5 Volt Operating Range
- Low Power Consumption—600 mW (typical)
- Fast instruction pointer—1.5 microseconds at 8MHz
- 14 input/output lines
- All inputs are Schmitt triggered
- 1K byte of ROM
- Two programmable 8-bit Counter/Timers each with 6-bit programmable prescaler
- 6 vectored, priority interrupts from 5 different sources
- Clock speed 1 to 8MHz
- Watchdog/Power-On Reset Timer
- Bit Programmable RC Oscillator
- On-chip oscillator that accepts a crystal, ceramic resonator, RC or external clock drive.

GENERAL DESCRIPTION

The Z8604 microcontroller (MCU) introduces a new level of sophistication to single-chip architecture. The Z8604 is a member of the Z8 single-chip microcontroller family with 1K of ROM. The device is housed in a 18-pin DIP, and is NMOS compatible. Zilog's NMOS microcontroller offers fast execution, more efficient use of memory, more sophisticated interrupts, input/output bit manipulation capabilities, and easy hardware/software system expansion along with low cost and low power consumption.

The Z8604 architecture is characterized by Zilog's 8-bit microcontroller core. The MCU offers a flexible I/O scheme, an efficient register, I/O, and a number of ancillary features that are useful in many industrial, high volume, peripheral types, and advanced scientific applications.

The device applications demand powerful I/O capabilities. The MCU fulfills this with 14 pins dedicated to input and output. These lines are grouped into two ports, and are configurable under software control to provide timing, status signals, or parallel I/O.

There are two basic address spaces available to support the wide range of configurations: Program Memory and 76 bytes of General Purpose Registers.

To unburden the program from coping with the real-time problems such as counting/timing and input/output data communication, the Z8604 offers two on-chip counter/timers with a large number of user selectable modes (Figure 1).

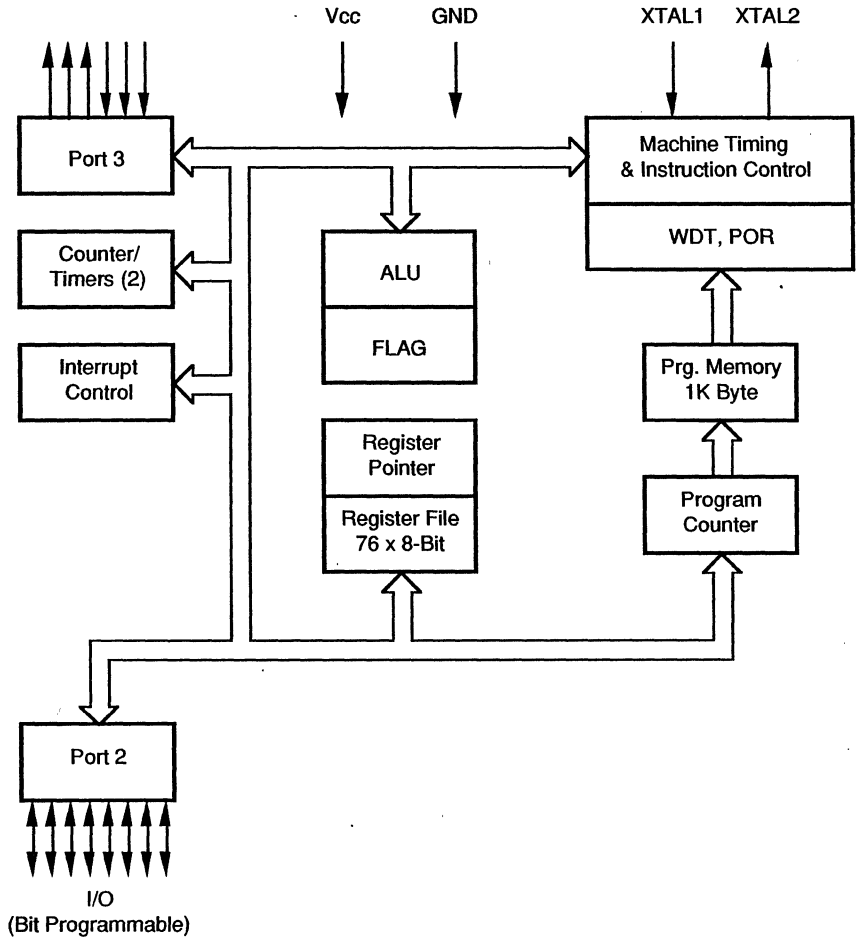


Figure 1. Functional Block Diagram

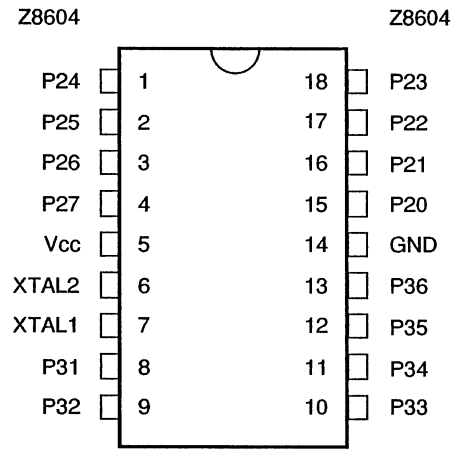


Figure 2. Pin Configuration

PIN DESCRIPTION

Table 1. Pin Description

Pin #	Symbol	Function	Direction
1-4	P24-7	Port 2 pin 4,5,6,7	In/Output
5	Vcc	Power Supply	Input
6	XTAL2	Crystal Oscillator Clock	Output
7	XTAL1	Crystal Oscillator Clock	Input
8-10	P31-3	Port 3 pin 1,2,3	Fixed Input
11-13	P34-6	Port 3 pin 4,5,6	Fixed Output
14	GND	Ground, V_{SS}	Input
15-18	P20-3	Port 2 pin 0,1,2,3	In/Output

PIN FUNCTIONS

XTAL1. *Crystal 1* (time-based input).
This pin connects a parallel-resonant crystal, ceramic resonator or an external single-phase clock to the on-chip oscillator input.

XTAL2. *Crystal 2* (time-based output).
This pin connects a parallel-resonant crystal, ceramic resonator to the on-chip oscillator output.

Port 2 P20-P27. Port 2 is an 8-bit, bidirectional, NMOS compatible I/O port. These 8 I/O lines can be configured under software control to be an input or output, independently. Input buffers are Schmitt triggered. Bits programmed as outputs are globally programmed as either push-pull or open drain (Figure 3).

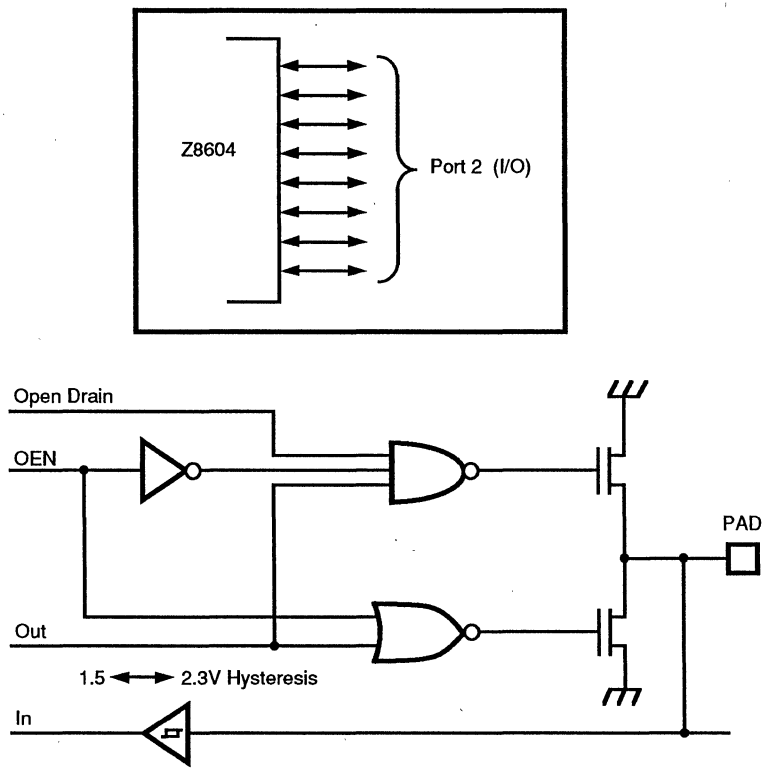


Figure 3. Port 2 Configuration

Port 3 P31-P36. Port 3 is a 6-bit port, NMOS compatible with three fixed input and three fixed output lines. These six lines consist of three fixed input (P31-P33) and three fixed output port (P34-P36) lines. Pins P31, P32 and P33 are standard Schmitt triggered NMOS inputs. Pins P34, P35, and P36 are push-pull outputs. Access to counter/timer 1 is made through P31 (Tin) and P36 (Tout) (Figure 4).

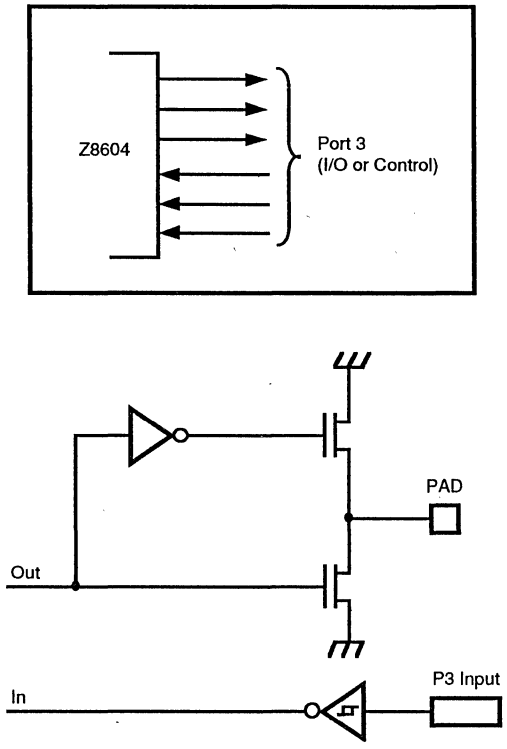


Figure 4. Port 3 Configuration

FUNCTIONAL DESCRIPTION

The Z8 MCU incorporates special functions to enhance the Z8's application in industrial, scientific research and advanced technologies applications.

Reset

The device resets in one of the following conditions:

Power-On Reset
Watch-Dog Timer

Program Memory

The Z8604 can address up to 1K byte of internal program memory (Figure 5). This 1K byte Program Memory is mask programmable. The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts. Byte 13 to byte 1024 consists of on-chip mask-programmed ROM.

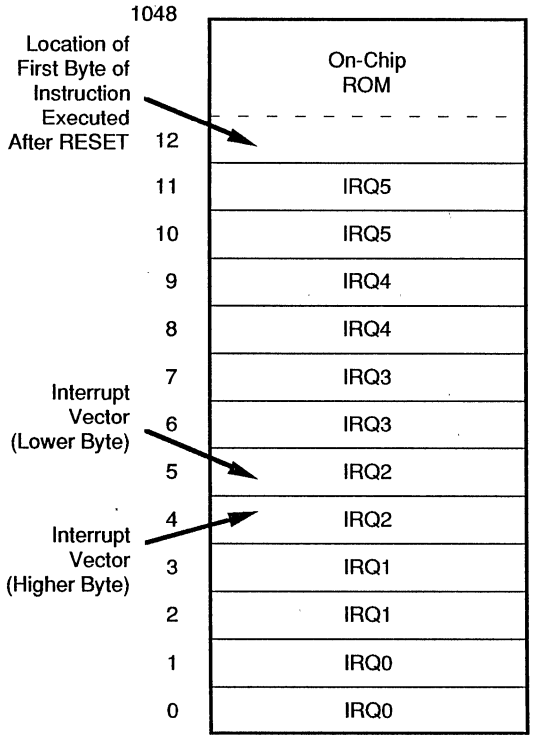


Figure 5. Program Memory Map

Register File

The Register File consists of two I/O port registers, 76 general-purpose registers and 15 control and status registers (Figure 6). The instructions can access registers directly or indirectly via an 8-bit address field. This allows a short 4-bit register address using the Register Pointer (Figure 7). In the 4-bit mode, the Register File is divided into 16 working register groups, each occupying 16 continuous locations. The Register Pointer addresses the starting location of the active working-register group.

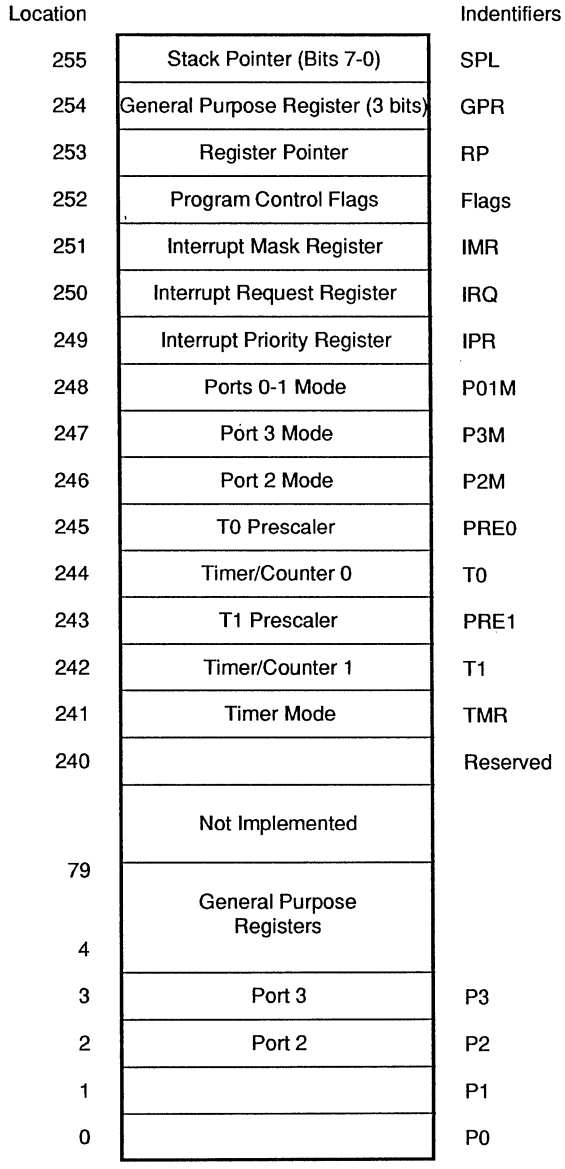


Figure 6. Register File

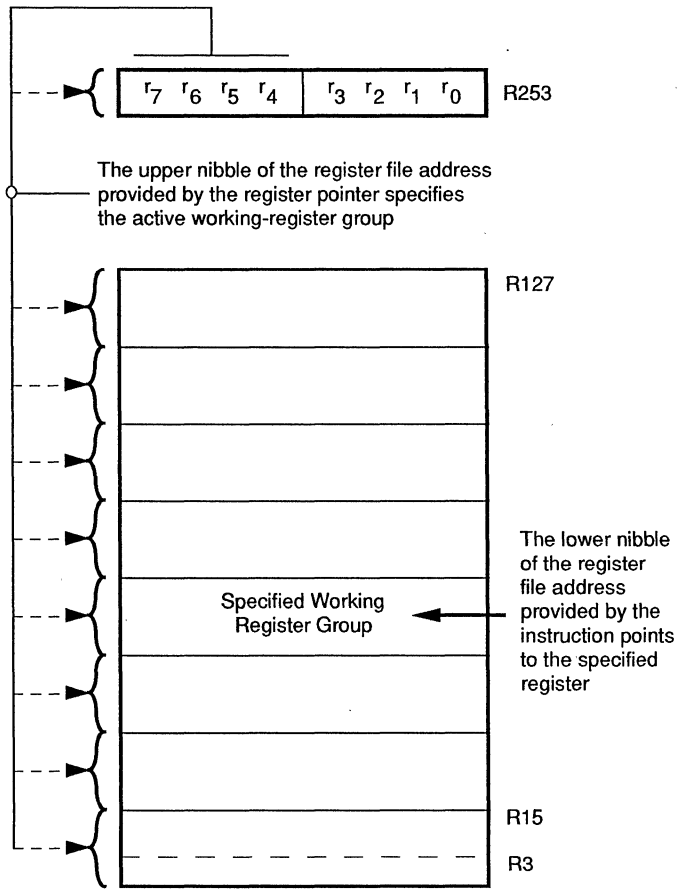


Figure 7. Register Pointer

Stack

The Z8604 has an 8-bit Stack Pointer (R255) that is used for the internal stack that resides within the 76 general purpose registers.

Table 1. Control Registers

Addr	Register	Reset Condition	D7	D6	D5	D4	D3	D2	D1	D0	Comments
F0		Not Implemented									
F1	TMR	Unchanged	0	0	0	0	0	0	0	0	
F2	Timer/CNTR1	Unchanged	U	U	U	U	U	U	U	U	
F3	PRE1	Unchanged	U	U	U	U	U	U	0	0	
F4	Timer/CNTR0	Unchanged	U	U	U	U	U	U	U	U	
F5	PRE0	Unchanged	U	U	U	U	U	U	U	0	
F6	Port 2 MDE	Unchanged	1	1	1	1	1	1	1	1	
F7	Port 3 MDE	X X X X X X X 0 D0: 0=P2 Open drain, 1=Push pull	U	U	U	U	U	U	U	0	
F8	Port 01 MDE	X X X 4 X 2 X X	U	U	U	0	U	1	U	U	Reserved
F9	IR Priority	Unchanged	U	U	U	U	U	U	U	U	
FA	IR Request	X X 5 4 3 2 1 0 D0=IRQ0=P32 Input D1=IRQ1=P33 Input D2=IRQ2=P31 Input	U	U	0	0	0	0	0	0	
		D3=IRQ3=P32 Input Inverted D4=T0 D5=T1									IRQ3 is used for positive edge detection.
FB	IR Mask	Unchanged	0	U	U	U	U	U	U	U	
FC	Flags	Unchanged	U	U	U	U	U	U	U	U	
FD	RP	RP Bank	U	U	U	U	U	U	U	U	
FE	SPH	X X X X X 2 1 0	U	U	U	U	U	U	U	U	
FF	SPL	Unchanged	U	U	U	U	U	U	U	U	

Counter/Timers

There are two 8-bit programmable counter/timers (T0-T1), each driven by its own 6-bit programmable prescaler. The T1 prescaler is driven by internal or external clock sources, however, the T0 prescaler is driven by the internal clock only (Figure 8).

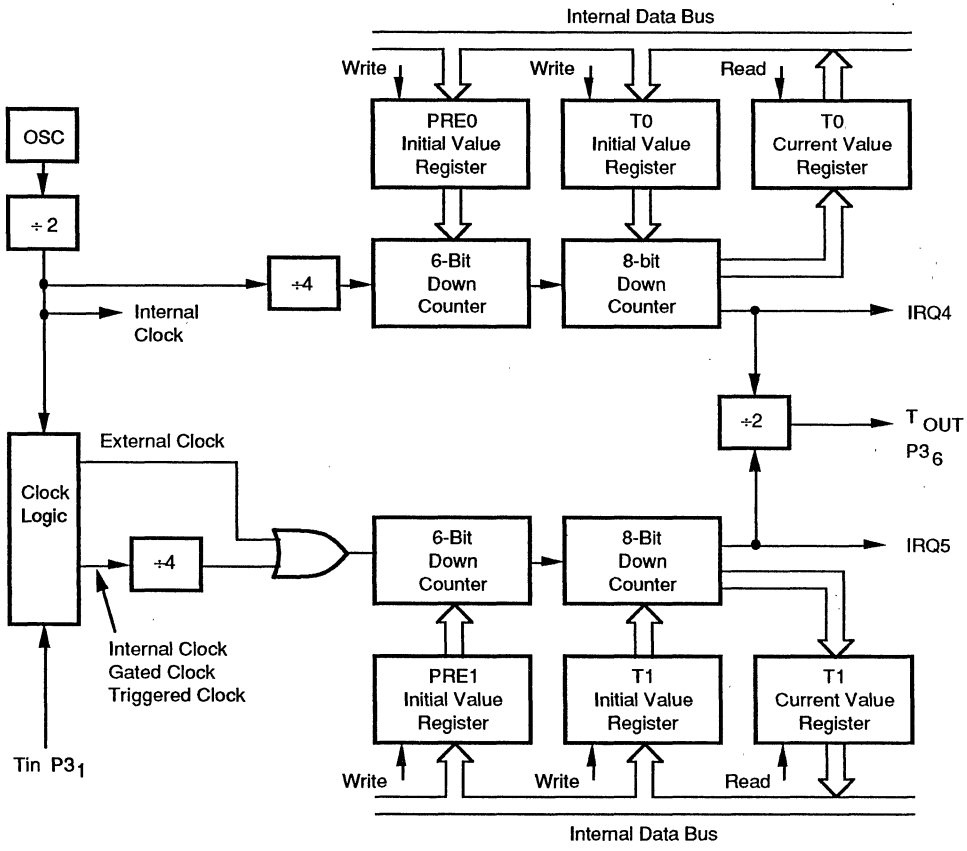


Figure 8. Counter/Timer Block Diagram

The 6-bit prescaler divides the input frequency of the clock source by any integer number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When both the counter and prescaler reach the end of count, a timer interrupt request-IRQ4 (T0) or IRQ5 (T1) is generated.

The counters are programmed to start, stop, restart to continue, or restart from the initial value. The counters are also programmed to stop upon reaching zero (single pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode).

The counters, but not the prescalers, are read at any time without disturbing their value or count mode. The clock source for T1 is user-definable and is either the internal microprocessor clock divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input (P31) as an external clock, a trigger input that is retriggerable or not-retriggerable, or as a gate input for the internal clock. Port 3 line P36 serves as a timer output (Tout) through which T0, T1 or the internal clock are output. The counter/timers can be cascaded by connecting the T0 output to the input of T1.

Interrupts

The Z8604 has six different interrupts from five different sources. The interrupts are maskable and prioritized (Figure 9). The five sources are divided as follow: three sources are claimed by Port 3 lines P31-P33, and two in counter/timers. The Interrupt Mask Register globally or individually enables or disables the six interrupt requests, (Table 2).

Table 2. Interrupt Types, Sources and Vectors

Name	Source	Vector Location	Edge triggered	Comment
IRQ0	IRQ0	0,1	Falling	Ext (P32)
IRQ1	IRQ1	2,3	Falling	Ext (P33)
IRQ2	IRQ2, Tin	4,5	Falling	Ext (P31)
IRQ3		6,7	Rising	Ext (P32)
IRQ4	T0	8,9		Internal
IRQ5	T1	10,11		Internal

When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority Register. All Z8604 interrupts are vectored through locations in the program memory. When an interrupt machine cycle is activated, an interrupt request is granted. Thus, this disables all subsequent interrupts, saves the Program Counter and Status Flags, and then branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit starting address of the interrupt service routine for that particular interrupt request.

To accommodate polled interrupt systems, interrupt inputs are masked and the interrupt request register is polled. This determines which of the interrupt requests needs services.

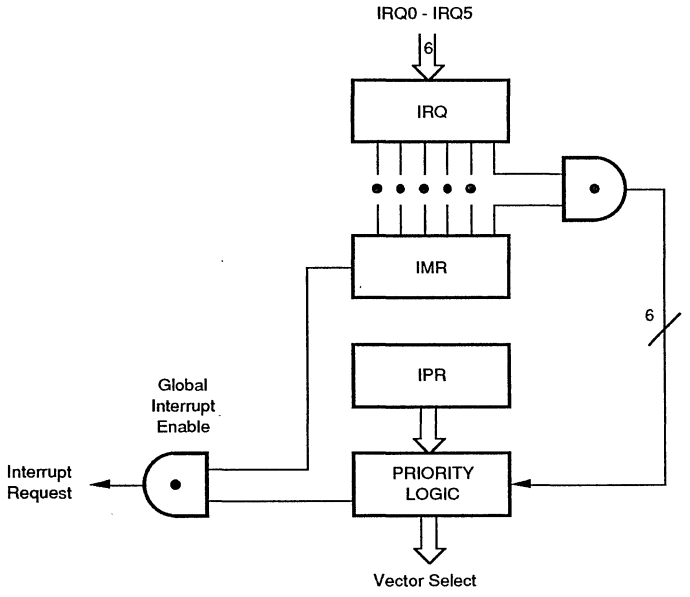


Figure 9. Interrupt Block Diagram

Clock

The Z8604 on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal, RC, ceramic resonator, or any suitable external clock source (XTAL1 = Input, XTAL2 = Output). The crystal should be AT cut, 1 to 8 MHz max, with a series resistance (RS) less than or equal to 100 Ohms (Figure 10a).

The Z8604 has an on-chip bit programmable RC Oscillator. The RC oscillator uses an internal capacitor and an external resistor to determine its operation frequency. The external resisitor is connected between VCC and XTAL1. Resistor values range from 0 to 100K. By connecting XTAL1 to VCC the maximum frequency is obtained (Figure 10b).

The crystal should be connected across XTAL1 and XTAL2 using the recommended capacitors (capacitance is between 15pf to 25pf which depends on the manufacturer of crystal, ceramic resonator and PCB layout) from each pin to ground.

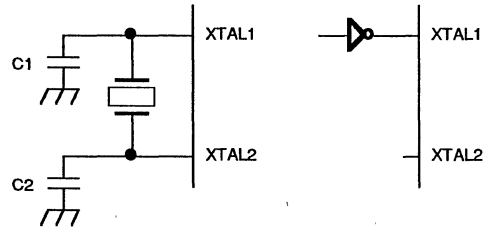


Figure 10a. Crystal Oscillator Configuration

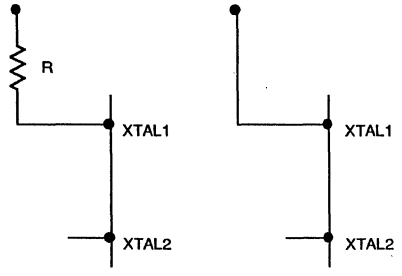


Figure 10b. RC Oscillator Configuration

Power-On Reset

A timer circuit clocked by a dedicated on-board RC oscillator and by the XTAL oscillator is used for the Power-On Reset (POR) timer function. The POR time allows Vcc and the oscillator circuit to stabilize before instruction execution begins. The POR timer circuit is a one-shot timer triggered by WDT timeout. The POR time is a nominal 40mS.

Watch Dog Timer (WDT). The WDT is enabled by instruction WDT. When the WDT is enabled, it cannot be stopped,and must be refreshed by executing the WDT instruction every 10 ms; otherwise the Z8604 will reset itself.

WDT=5F (HEX)

Opcode WDT (5F%). The first time opcode %5F is executed, the WDT is enabled, subsequent execution clears the WDT counter. This has to be done at least every 10 ms. Otherwise, the WDT will time out and generate a reset. The generated reset is the same as a power on reset of 40 ms+18 XTALK clock cycles.

STANDARD TEST CONDITIONS

The characteristics listed below apply for standard test conditions as noted. All voltages are referenced to ground. Positive current flows into the referenced pin (Figure 11).

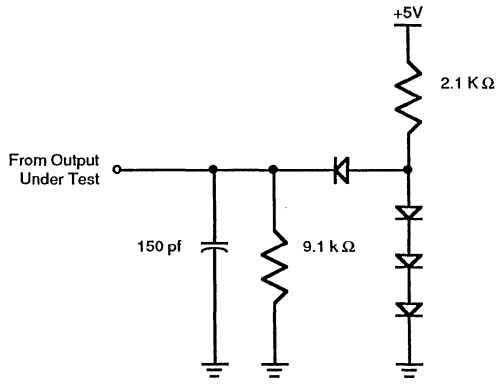


Figure 11. Test Load Diagram

ABSOLUTE MAXIMUM RATINGS

Symbol	Description	Min	Max	Units
V_{CC}	Supply Voltage*	-0.3	+7.0	V
TSTG	Storage Temp	-65	+150	C
TA	Oper Ambient Temp	†		C

† See Ordering Information

Note (*). Voltage on all pins with respect to GND Stress greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC CHARACTERISTICS

V_{CC}=+4.5 to +5.5V

		TA=0°C to 70°C				
Sym	Parameter	Min	Max	Typ	Unit	Condition
V_{CH}	Clock Input High Voltage	3.8	V_{CC}		V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	V_{SS}-0.3	8.0		V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.75	V_{CC}		V	
V_{IL}	Input Low Voltage	0.3	1.5		V	
V_{OH}	Output High Voltage	2.4			V	I_{OH}= -250 µA
V_{OL}	Output Low Voltage	0.4			V	I_{OL}= +2.0 mA
I_{IL}	Input Leakage	-10	10		µA	V_{IN}= 0V, V_{CC}
I_{OL}	Output Leakage	-10	10		µA	V_{IN}= 0V, V_{CC}
I_{CC}	Supply Current		120		mA	

REGISTER DIAGRAMS

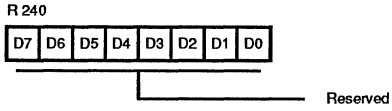


Figure 12. Reserved

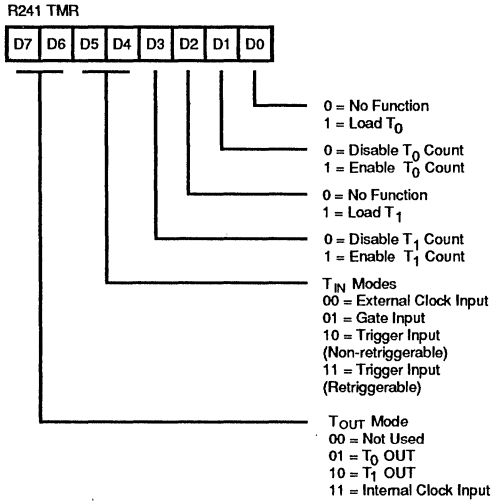


Figure 13. Timer Mode Register
(F1H; Read/Write)

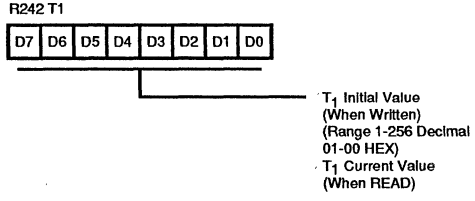


Figure 14. Counter Timer 1 Register
(F2H; Read/Write)

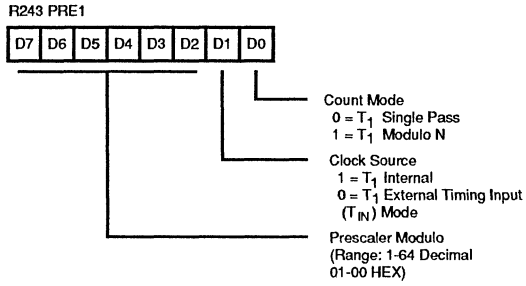


Figure 15. Prescaler 1 Register
(F3H; Write Only)

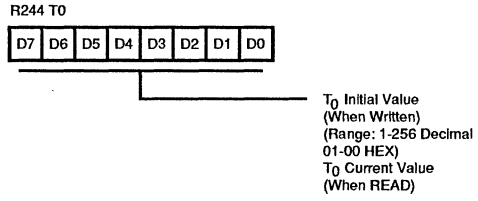


Figure 16. Counter/Timer 0 Register
(F4H; Read/Write)

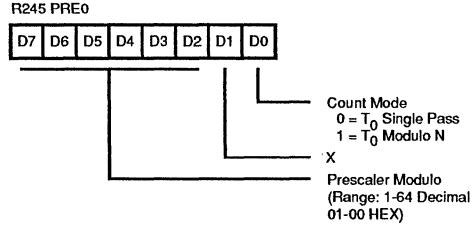


Figure 17. Prescaler 0 Register
(F5H; Write Only)

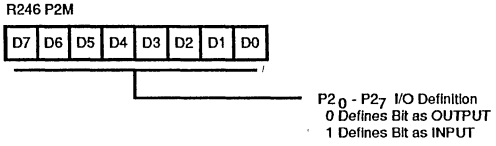


Figure 18. Port 2 Mode Register
(F6H; Write Only)

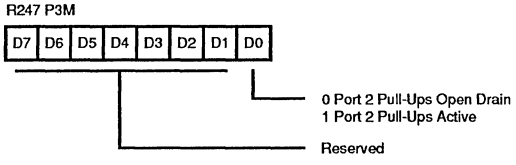


Figure 19. Port 3 Mode Register
(F7H; Write Only)

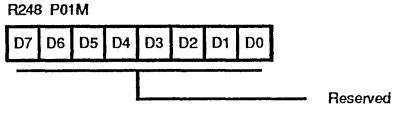


Figure 20. Port 0 and 1 Mode Register

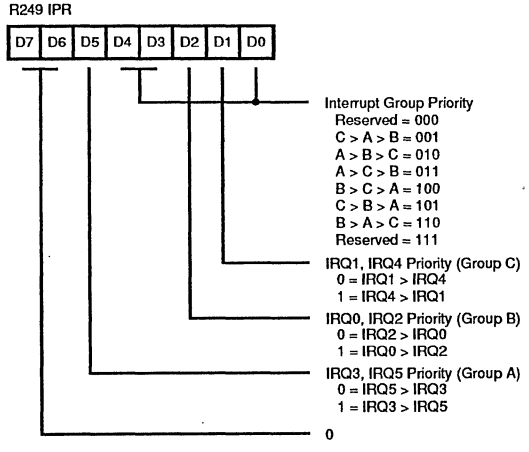


Figure 21. Interrupt Priority Register
(F9H; Write Only)

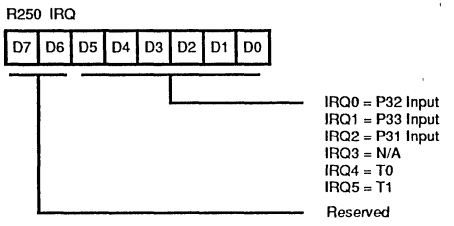


Figure 22. Interrupt Req Register
(FAH; Read/Write)

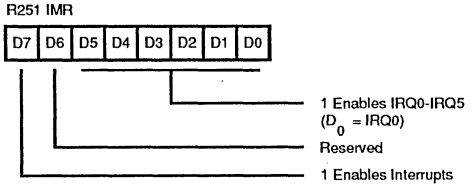


Figure 23. Interrupt Mask Register
(FBH; Read/Write)

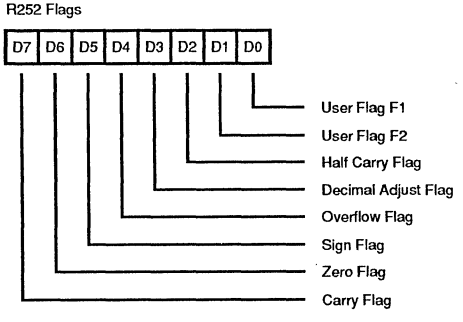


Figure 24. Flag Register
(FCH; Read/Write)

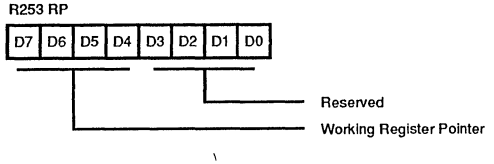


Figure 25. Register Pointer
(FDH; Read/Write)

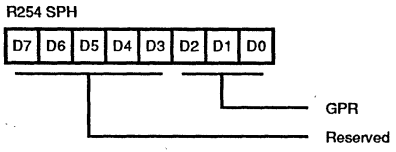


Figure 26. General Purpose Register
(FEH; Read/Write)

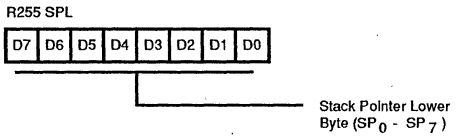


Figure 27. Stack Pointer
(FFH; Read/Write)

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

Symbol	Meaning
IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed Address
DA	Direct Address
RA	Relative Address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-Register or indirect working-egister address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

Symbol	Meaning
dst	Destination location or contents
src	Source location or contents
cc	Condition Code
@	Indirect address prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag register (Control Register 252)
RP	Register Pointer (R253)
IMR	Interrupt Mask Register (R251)

Flags. Control register (R252) contains the following six flags:

Symbol	Meaning
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

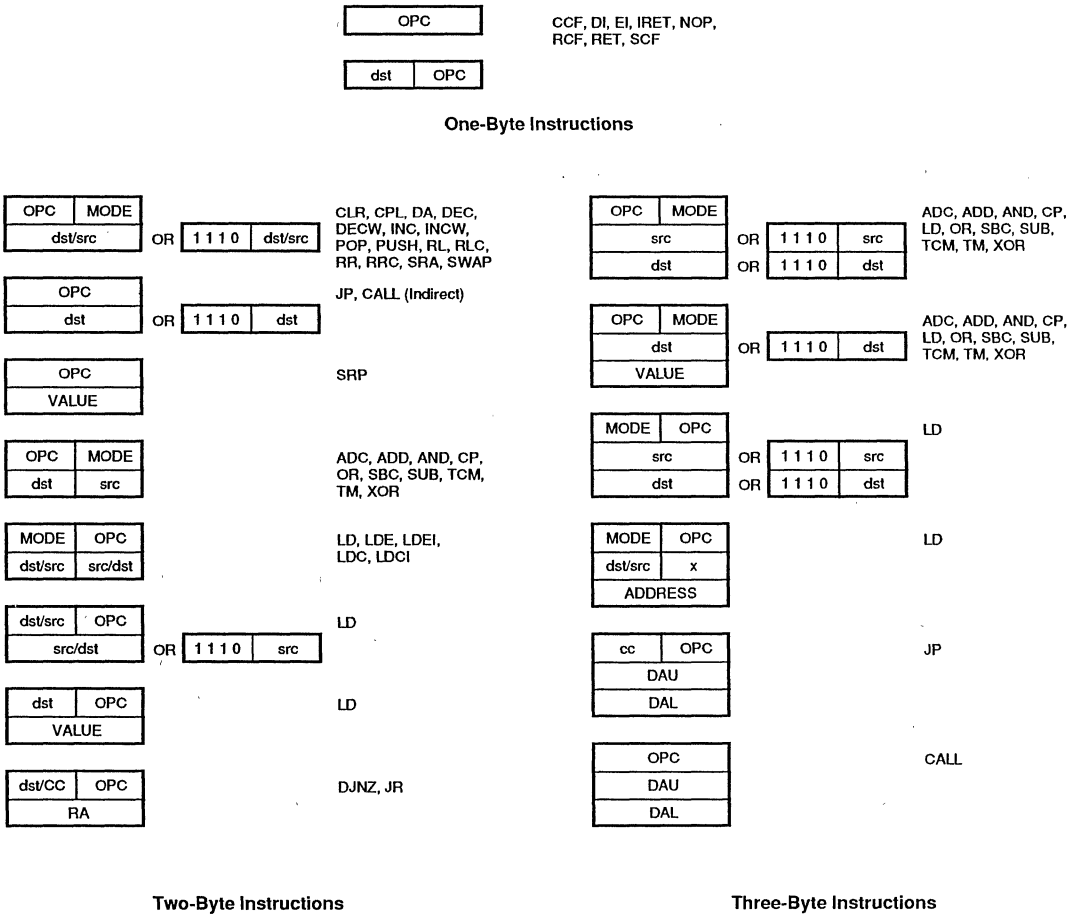
Affected flags are indicated by:

0	Clear to zero
1	Set to one
*	Set to clear according to operation
-	Unaffected
x	Undefined

Table 3. Condition Codes

Value	Mnemonic	Meaning	Flags Set
1000		Always True	
0111	C	Carry	C = 1
1111	NC	No Carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not Zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No Overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not Equal	Z = 0
1001	GE	Greater Than or Equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater Than	[Z OR (S XOR V)] = 0
0010	LE	Less Than or Equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned Greater Than or Equal	C = 0
0111	ULT	Unsigned Less Than	C = 1
1011	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned Less Than or Equal	(C OR Z) = 1
0000		Never True	

INSTRUCTION FORMATS



INSTRUCTION SUMMARY

Note: Assignment of a value is indicated by the symbol "—". For example:

dst — dst + src

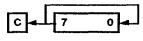
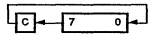
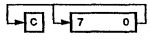
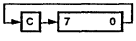
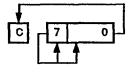
indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr (n)" is used to refer to bit (n) of a given operand location.

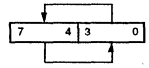
INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst, src dst←dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst←dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst←dst AND src	†		5[ ]	-	*	*	0	-	-
CALL dst SP←SP - 2 @SP←PC, PC←dst	DA IRR		D6 D4	-	-	-	-	-	-
CCF C←NOT C			EF	*	-	-	-	-	-
CLR dst dst←0	R IR		B0 B1	-	-	-	-	-	-
COM dst dst←NOT dst	R IR		60 61	-	*	*	0	-	-
CP dst, src dst - src	†		A[ ]	*	*	*	*	-	-
DA dst dst←DA dst	R IR		40 41	*	*	*	X	-	-
DEC dst dst←dst - 1	R IR		00 01	-	*	*	*	-	-
DECW dst dst←dst - 1	RR IR		80 81	-	*	*	*	-	-
DI IMR(7)←0			8F	-	-	-	-	-	-
DJNZr, dst r←r - 1 if r ≠ 0 PC←PC + dst Range: +127, -128	RA		rA r = 0 - F	-	-	-	-	-	-
EI IMR(7)←1			9F	-	-	-	-	-	-
WDT			5F	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
INC dst dst←dst + 1	r R IR		rE r = 0 - F 20 21	-	*	*	*	-	-
INCW dst dst←dst + 1	RR IR		A0 A1	-	*	*	*	-	-
IRET FLAGS←@SP; SP←SP + 1 PC←@SP; SP←SP + 2; IMR(7)←1			BF	*	*	*	*	*	*
JP cc, dst if cc is true PC←dst	DA IRR		cD c = 0 - F 30	-	-	-	-	-	-
JR cc, dst if cc is true, PC←PC + dst Range: +127, -128	RA		cB c = 0 - F	-	-	-	-	-	-
LD dst, src dst←src	r r R r X r Ir R R IR IR	Im R r X r Ir r R IR IM IM R	rC r8 r9 r = 0 - F C7 D7 E3 F3 E4 E5 E6 E7 F5	-	-	-	-	-	-
LDC dst, src	r	Irr	C2	-	-	-	-	-	-
LDCI dst, src r←r +1; rr←rr + 1	Ir	Irr	C3	-	-	-	-	-	-

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
NOP			FF	-	-	-	-	-	-
OR dst, src dst←dst OR src	†		4[ ]	-	*	*	0	-	-
POP dst dst←@SP; SP←SP + 1	R IR		50 51	-	-	-	-	-	-
PUSH src SP←SP - 1; @SP←src		R IR	70 71	-	-	-	-	-	-
RCF C←0			CF	0	-	-	-	-	-
RET PC←@SP; SP←SP + 2			AF	-	-	-	-	-	-
RL dst 	R IR		90 91	*	*	*	*	-	-
RLC dst 	R IR		10 11	*	*	*	*	-	-
RR dst 	R IR		E0 E1	*	*	*	*	-	-
RRC dst 	R IR		C0 C1	*	*	*	*	-	-
SBC dst, src dst←dst←src←C	†		3[ ]	*	*	*	*	1	*
SCF C←1			DF	1	-	-	-	-	-
SRA dst 	R IR		D0 D1	*	*	*	0	-	-

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
SRP src RP←src		Im	31	-	-	-	-	-	-
SUB dst, src dst←dst←src	†		2[ ]	*	*	*	*	1	*
SWAP dst 	R IR		F0 F1	X	*	*	X	-	-
TCM dst, src (NOT dst) AND src	†		6[ ]	-	*	*	0	-	-
TM dst, src dst AND src	†		7[ ]	-	*	*	0	-	-
XOR dst, src dst←dst XOR src	†		B[ ]	-	*	*	0	-	-

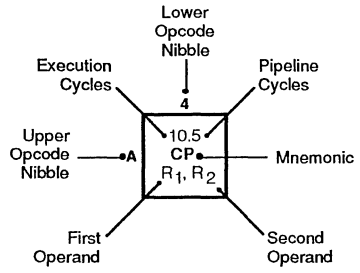
† These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address Mode dst	Address Mode src	Lower Opcode Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								5.0 **WDT**
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								
8	10.5 **DECW** RR1	10.5 **DECW** IR1														**DI**
9	6.5 **RL** R1	6.5 **RL** IR1														6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14.0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2					10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1	**LDCI** Ir1,Irr2		20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6.5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2			3					2					3	1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_2 = Dst address
R_1 or r_2 = Src address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank are not defined.

* 2-byte instruction appears as a 3-byte instruction

PRODUCT SPECIFICATION

Z8612

NMOS Z8® ICE
IN-CIRCUIT EMULATOR

FEATURES

- 8-bit microcontroller emulator, 64-pin shrink DIP or 68-pin PLCC package
- 4.5 to 5.5 Voltage operating range
- Average instruction execution time of 1.5 µs
- Fast instruction pointer - 1 µs @ 12 MHz
- 32 input/output lines
- Full-duplex UART
- 4K Internal Program Emulation
- All digital inputs are TTL levels
- 124 bytes of RAM
- Two programmable 8-bit Counter/Timers each with 6-bit programmable prescaler
- Six vectored, priority interrupt from eight different sources
- Clock speeds 8 and 12 MHz
- On-chip oscillator that accepts a crystal, ceramic resonator, LC or external clock drive

GENERAL DESCRIPTION

The Z8612 ICE (In-Circuit Emulator) introduces a new level of sophistication to single-chip architecture.

The ICE is housed in a 64-pin shrink DIP or 68-pin PLCC package, and is manufactured in NMOS technology.

The ICE development device allows users to prototype a system with an actual hardware device and to develop the code. This code is eventually mask-programmed into the on-chip ROM for any of the Z86XX devices. Development devices are also useful in emulator applications where the final system configuration, I/O, interrupt inputs, etc. are unknown. The ICE development device is identical to its equivalent Z8611 microcontroller with the following exceptions:

- No internal ROM is provided, so that code is developed in off-chip memory.
- The normally internal ROM address and data lines are buffered and brought out to external pins to interface with the external memory.
- Control line (/DAS) is added to interface with external program memory.
- The Timing and Control, I/O ports, and clock pins on the Z8612 are identical in function to those on the Z8611.

The ICE architecture is characterized by Zilog's 8-bit microcontroller core. The device offers; fast execution, more efficient use of memory, more sophisticated interrupts, input/output bit manipulation capabilities, easy hardware/software system expansion, a flexible I/O scheme, an efficient register and address space structure, multiplexed capabilities between address/data, and a number of ancillary features that are useful in many industrial and advanced scientific applications.

Industrial applications demand powerful I/O capabilities. The ICE fulfills this with 32 pins dedicated to input and output. These lines are grouped into four ports. Each port consists of eight lines and is configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an address/data bus for interfacing external memory.

GENERAL DESCRIPTION (Continued)

There are three basic address spaces available to support this wide range of configuration: Program Memory, Data Memory, and 124 General-Purpose Registers.

To unburden the program from coping with the real-time problems such as counting/timing and serial data communication, the ICE offers two on-chip counter/timers with a large number of user selectable modes, and an asynchronous receiver/transmitter (UART – Figure 1).

Note: All Signals with a preceding front slash, "/", are active Low, e.g.: B//W (WORD is active Low); /B/W (BYTE is active Low, only).

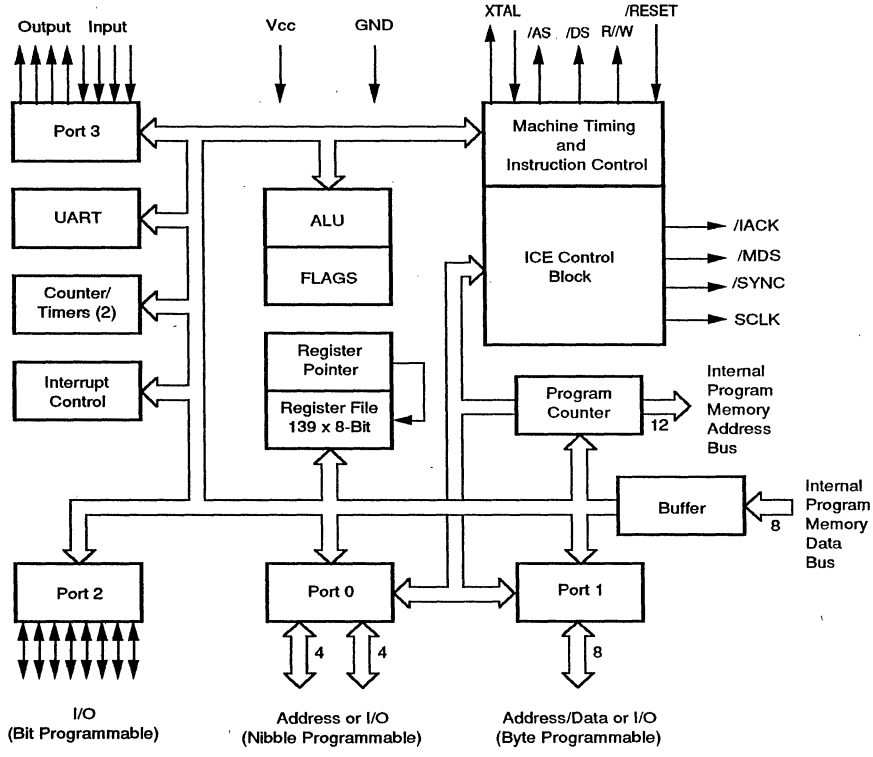


Figure 1. Functional Block Diagram

PIN DESCRIPTION

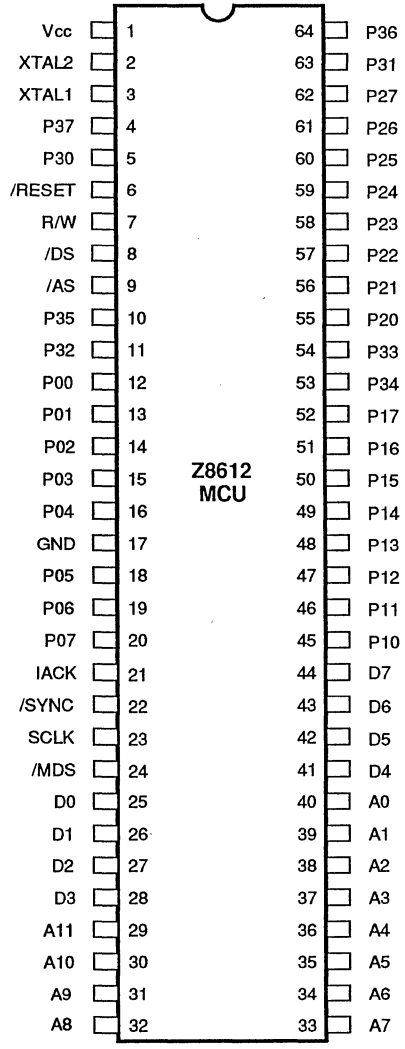


Figure 2. 64-Pin Dual In-Line Plastic (DIP) Pin Assignments

PIN DESCRIPTION (Continued)

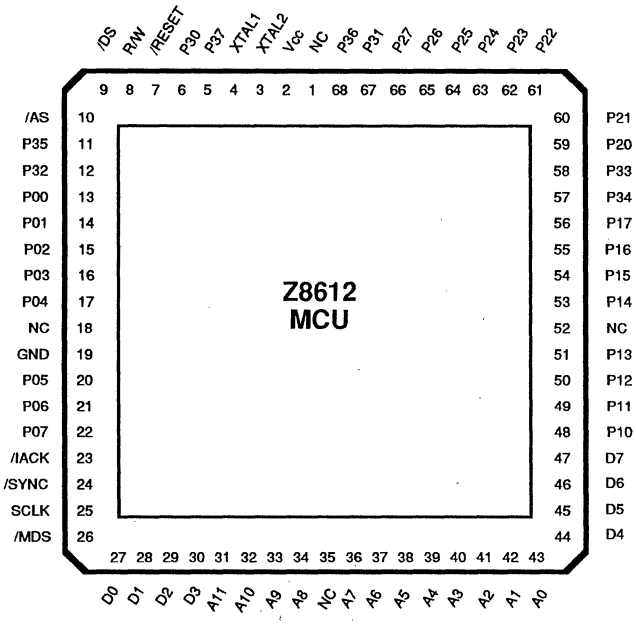


Figure 3. 68-Pin Plastic Chip Carrier Pin Assignments

/DS. (output, active Low). Data Strobe is activated once for each external memory transfer. For a READ operation, data must be available prior to the trailing edge of /DS. For WRITE operations, the falling edge of /DS indicates that output data is valid.

/AS. (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Address output is via Port 1 for all external programs. Program or data memory address transfers are valid at the trailing edge of /AS. Under program control, /AS can be placed in the high-impedance state along with Ports 0 and 1, Data Strobe, and Read/Write.

XTAL1, XTAL2. *Crystal 1, Crystal 2* (time-based input and output, respectively). These pins connect a parallel-resonant crystal, ceramic resonator, LC, or any external single-phase clock to the on-chip oscillator and buffer.

R/W. (output, write Low). The Read/Write signal is low when the ICE is writing to external program or data memory.

/RESET. (input, active-Low). To avoid asynchronous and noisy reset problems, the ICE is equipped with a reset filter of four external clocks (4TpC). If the external /RESET signal is less than 4TpC in duration, no reset occurs. On the 5th clock after /RESET is detected, an internal RST signal is latched and held for an internal register count of 18 external clocks, or for the duration of the external /RESET, whichever is longer. During the reset cycle, /DS is held active low while /AS cycles at a rate of TpC/2. When /RESET is deactivated, program execution begins at location 000C (HEX). Reset time must be held low for 50 ms, or until Vcc is stable, whichever is longer.

D7 - D0. (I/O, TTL compatible) Internal Data bus. These eight lines provide the data bus to access external memory emulating the on-chip ROM. During read cycles in the internal memory space, the data on these lines is latched in just prior to the rise of the /MDS data strobe.

A11 - A0. (outputs TTL compatible) Internal Address bus. During T1 these lines output the current memory address. All addresses, whether internal or external, are output.

/MDS. (output, TTL compatible) Memory Data Strobe. This is a timing signal used to enable the external memory to emulate the on-chip ROM. It is active only

during accesses to the on-chip ROM memory space (the bottom 4K of program memory).

/SCLK. (output, TTL compatible) System Clock. This line is the internal system clock.

/SYNC. (output, TTL compatible) Sync signal. This signal indicates the last clock cycle of the currently executing instruction.

/IACK. (output, TTL compatible) Interrupt Acknowledge. This output, when low, indicates that the ICE is an interrupt cycle.

I/O PORTS

Port 0 P00-P07. Port 0 is an 8-bit, nibble programmable, bidirectional, TTL compatible port. These eight I/O lines can be configured under software control as a nibble I/O port, or as an address port for interfacing external memory. When used as an I/O port, Port 0 may be placed under handshake control. In this configuration, Port 3, lines P32 and P35 are used as the handshake control /DAV0 and RDY0 (Data available and Ready). Handshake signal assignment is dictated by the I/O direction of the upper nibble P04-P07. The lower nibble must have the same direction as the upper nibble to be under handshake control. For the ROMless option, Port 0 appears as A15-A8 Address lines after reset.

For external memory references, Port 0 can provide address bits A11-A8 (lower nibble) or A15-A8 (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. If one or both nibbles are needed for I/O operation, they are configured by writing to the Port 0 Mode register. In ROMless mode, after a hardware reset, Port 0 lines are defined as address lines A15-A8, and extended timing is set to accommodate slow memory access. The initialization routine can include reconfiguration to eliminate this extended timing mode (Figure 4).

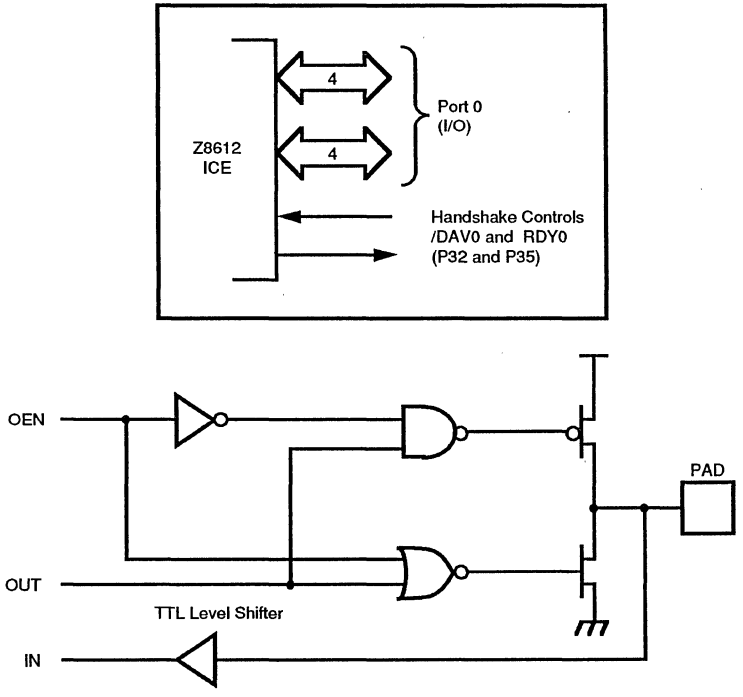


Figure 4. Port 0 Configuration

PIN FUNCTIONS (Continued)

Port 1 P10-P17. Port 1 is an 8-bit, byte programmable, bidirectional, TTL compatible port. It has multiplexed Address (A7-A0) and Data (D7-D0) ports. For the ICE, these eight I/O lines are programmed as Input or Output lines or the port can be configured, under software control, as an address/data port for interfacing external memory. When used as an I/O port, Port 1 may be placed under handshake control. In this configuration, Port 3, lines P33 and P34 are used as the handshake controls RDY1 and /DAV1.

Memory locations greater than 4096 are referenced through Port 1. To interface external memory, Port 1 must be programmed for the multiplexed Address/Data mode. If more than 256 external locations are required, Port 0 must output the additional lines.

Port 1 can be placed in high-impedance state along with Port 0, /AS, /DS and R//W, allowing the ICE to share common resource in multiprocessor and DMA applications. Data transfers can be controlled by assigning P33 as a Bus Acknowledge input, and P34 as a Bus Request output (Figure 5).

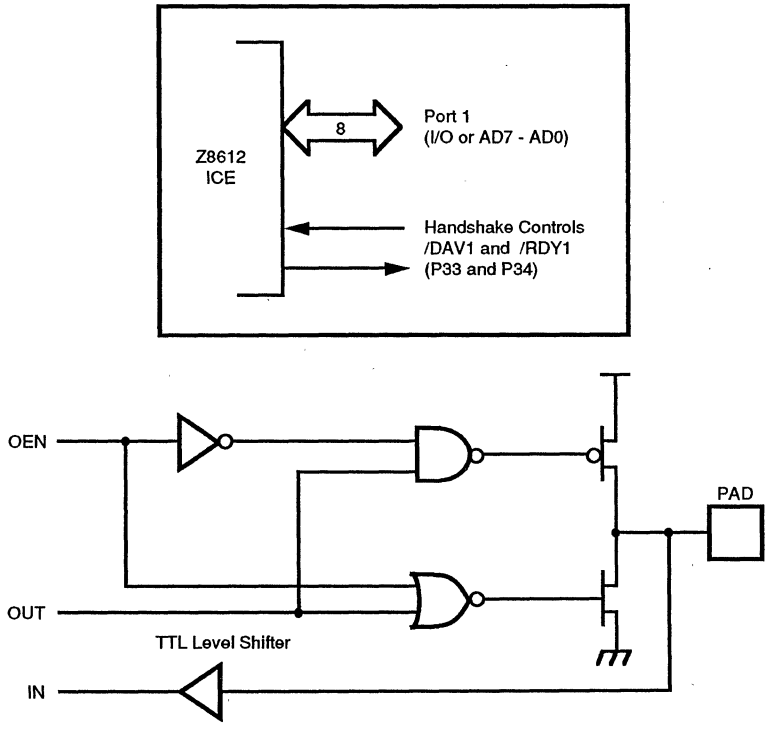


Figure 5. Port 1 Configuration

Port 2 P20-P27. Port 2 is an 8-bit, bit programmable, bidirectional, CMOS compatible port. Each of these eight I/O lines can be independently programmed as an input or output or globally as an open-drain output. Port 2 is always available for I/O operation. When used as an I/O port, Port2 may be placed under handshake control. In this configuration, Port 3, lines P31 and P36 are used as the handshake controls lines /DAV2 and RDY2. The handshake signal assignment for Port 3 lines P31 and P36 is dictated by the direction (input or output) assigned to P27 (Figure 6).

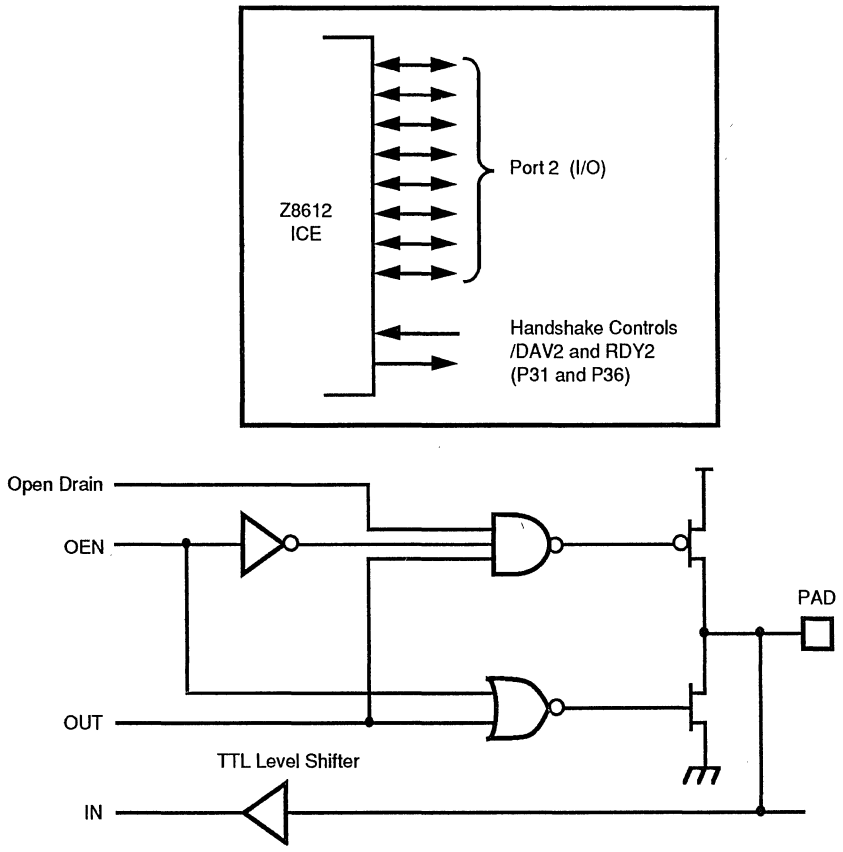


Figure 6. Port 2 Configuration

PIN FUNCTIONS (Continued)

Port 3 P30-P37. Port 3 is an 8-bit, CMOS compatible four-fixed input and four-fixed output port. These eight I/O lines have four-fixed (P33-P30) input and four fixed (P37-P34) output ports. Port 3, when used as serial I/O, are programmed as serial in and serial out, respectively (Figure 7 and Table 1).

Port 3 is configured under software control to provide the following control functions: handshake for Ports 0 and 2 (/DAV and RDY); four external interrupt request signals (IRQ0-IRQ3); timer input and output signals (T_{IN} and T_{OUT}); Data Memory Select (/DM).

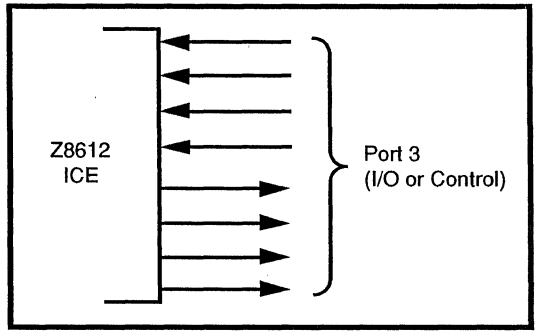


Figure 7. Port 3 Configuration

Table 1. Port 3 Pin Assignments

Pin	I/O	CTC1	Int.	P0 HS	P1 HS	P2 HS	UART	Ext
P30	IN		IRQ3				Serial In	
P31	IN	T_{IN}	IRQ2			D/R		
P32	IN		IRQ0	D/R				
P33	IN		IRQ1		D/R			
P34	OUT				R/D			DM
P35	OUT			R/D				
P36	OUT	T_{OUT}				R/D		
P37	OUT						Serial Out	

Notes
HS = Handshake Signal
D = Data Available
R = Ready

Port 3, lines P30 and P37, can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by the Counter/Timer 0.

The ICE automatically adds a start bit and two stop bits to transmitted data (Figure 8). Odd parity is also available as an option. Eight data bits are always transmitted, regard less of parity selection. If parity is enabled, the eighth bit is the odd parity bit. An interrupt request (IRQ4) is generated on all transmitted characters.

Received data must have a start bit, eight data bits and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ3 interrupt request.

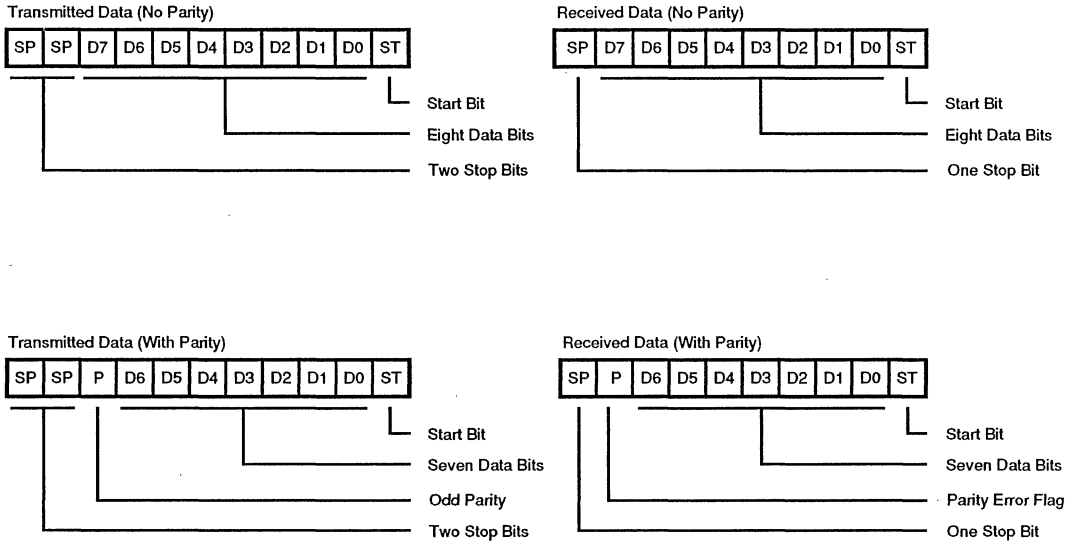


Figure 8. Serial Data Formats

Address Space

Program Memory. The ICE can address up to 64 Kbytes of external program memory (Figure 9). The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts. 000C to 4095 is the memory map for the Internal ROM to be emulated, and 4096 to 65535, the remaining program memory for which the ICE executes external memory fetches.

Data Memory (/DM). External data memory is included with, or separated from, the external program memory space. /DM, an optional I/O function that can be programmed to appear on pin P34, is used to distinguish between data and program memory space (Figure 10). The state of the /DM signal is controlled by the type instruction being executed. An LDC opcode references PROGRAM (/DM inactive) memory, and an LDE instruction references DATA (/DM active low) memory. The lower unaddressable part of the data memory is in fact addressable with the ICE chip's /MDS line (as /DS is not active for internal ROM reads), but there is no need for this.

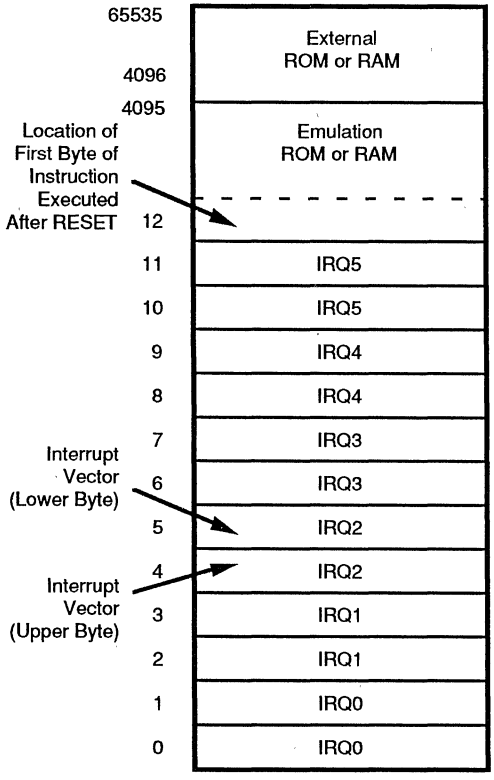


Figure 9. Program Memory Configuration

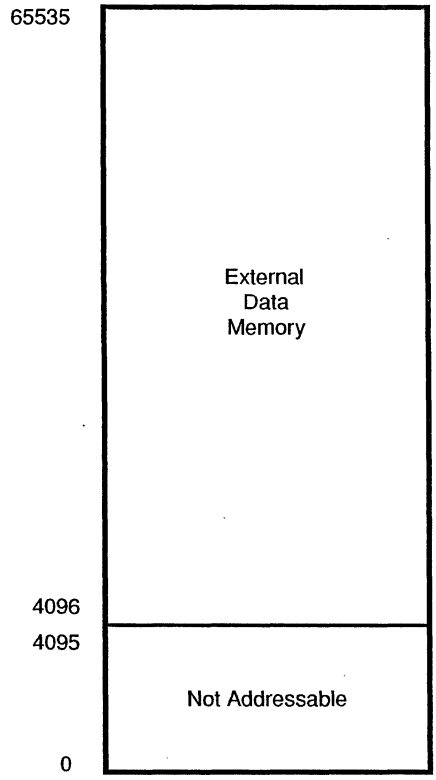


Figure 10. Data Memory Configuration

Register File. The Register File consists of four I/O port registers, 144 general-purpose registers and 16 control and status registers (Figure 11). The instructions can access registers directly or indirectly via an 8-bit address field. The ICE also allows short 4-bit register addressing using the Register Pointer (Figure 12). In the 4-bit mode, the Register File is divided into 16 working register groups, each occupying 16 continuous locations. The Register Pointer addresses the starting location of the active working-register group.

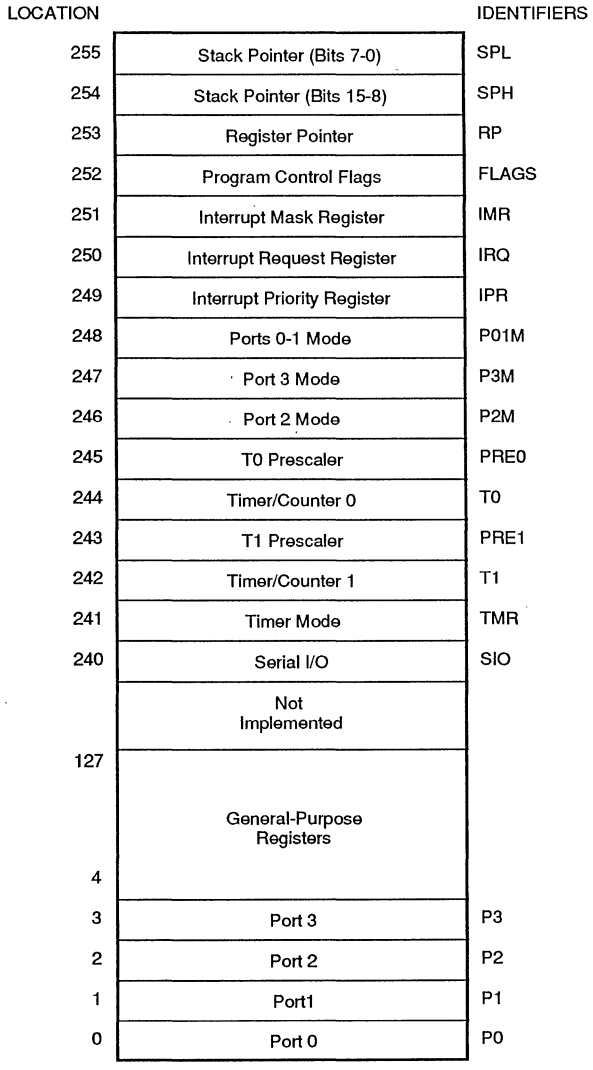


Figure 11. Register File

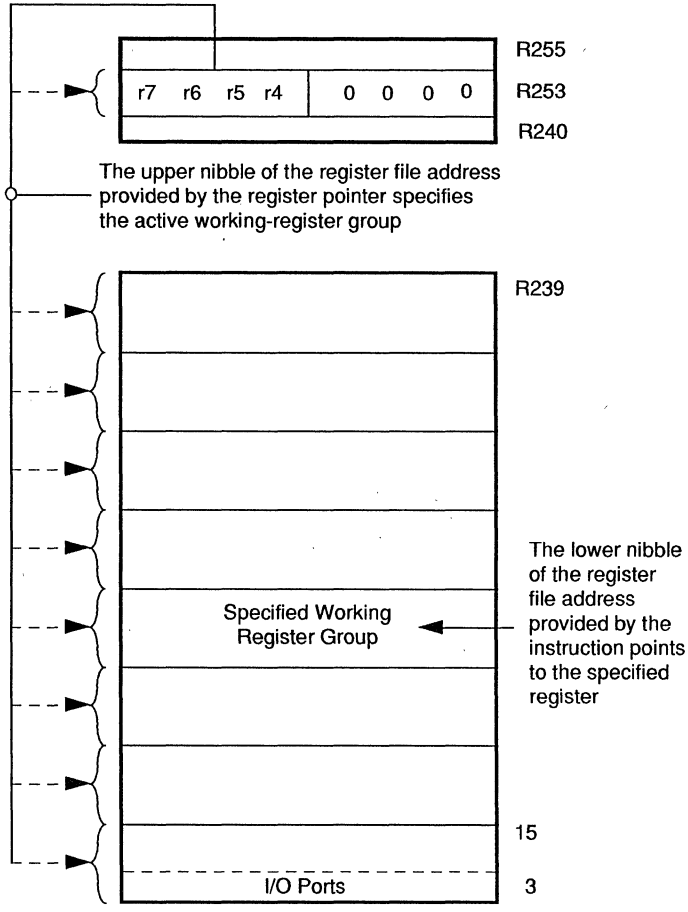


Figure 12. Register Pointer

Stack. The ICE has a 16-bit Stack Pointer (R254-R255) used for external stack that resides anywhere in the data memory for the ROMless mode, but only from 4096 to 65535 in ROM mode.

An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 124 general-purpose registers (R4-R127). The high byte of the Stack Pointer (SPH-Bit 8-15) can be used as a general purpose register when using internal stack only.

FUNCTIONAL DESCRIPTION

Counter/Timers. There are two 8-bit programmable counter/timers (T0-T1), each driven by its own 6-bit programmable prescaler. The T1 prescaler is driven by internal or external clock sources; however, the T0 prescaler is driven by the internal clock only (Figure 13).

The 6-bit prescalers can divide the input frequency of the clock source by any integer number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When both the counters and prescaler reach the end of the count, a timer interrupt request, IRQ4 (T0) or IRQ5 (T1), is generated.

The counter can be programmed to start, stop, restart to continue, or restart from the initial value. The counters can also be programmed to stop upon reaching zero (single pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode).

The counter, but not the prescalers, can be read at any time without disturbing their value or count mode. The clock source for T1 is user-definable and can be either the internal microprocessor clock divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input (P31) as an external clock, a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. Port 3 line P36 also serves as a timer output (Tout) through which T0, T1 or the internal clock can be output. The counter/timers can be cascaded by connecting the T0 output to the input of T1

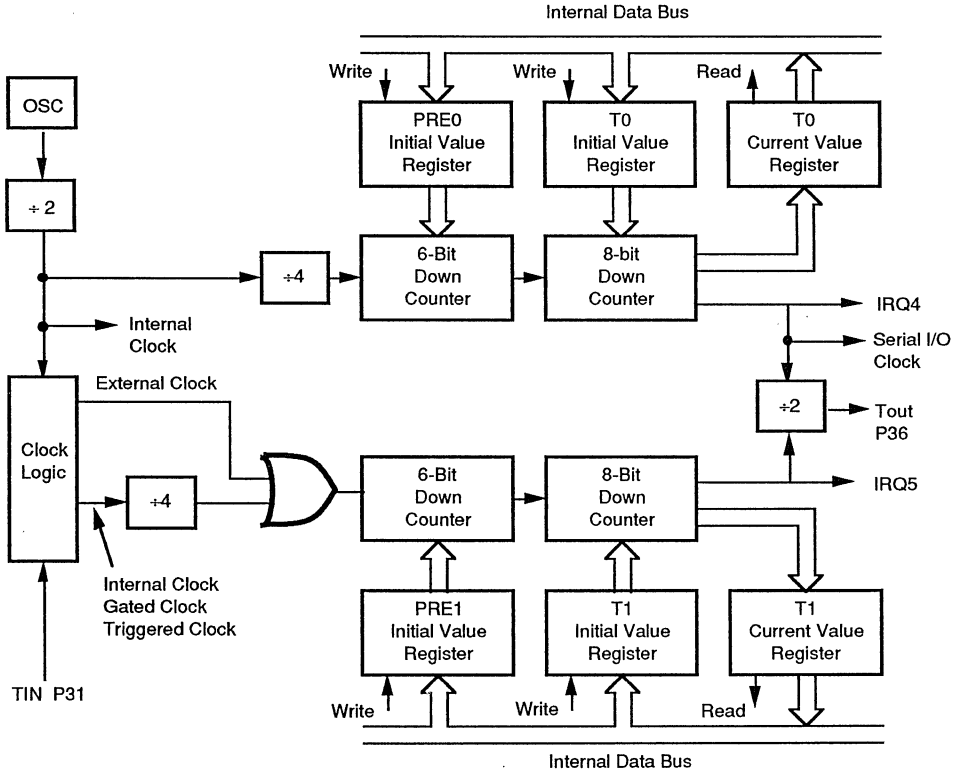


Figure 13. Counter/Timers Block Diagram

FUNCTIONAL DESCRIPTION (Continued)

Interrupts. The ICE has six different interrupts from eight different sources. The interrupts are maskable and prioritized. The eight sources are divided as follows: four sources are claimed by Port 3 lines P30-P33, one in Serial Out, one in Serial In, and two in the counter/timers (Figure 14). The Interrupt Mask Register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register. All ICE interrupts are vectored through locations in the program memory. When an interrupt machine cycle is activated, an interrupt request is granted. Thus, this disables all of the subsequent interrupts, saves the Program Counter and Status Flags, and then branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

To accommodate polled interrupt systems, interrupt inputs are masked and the Interrupt Request register is polled to determine which of the interrupt requests need service. Software initialed interrupts are supported by setting the appropriate bit in the Interrupt Request Register (IRQ).

Internal interrupt requests are sampled on the falling edge of the last cycle of every instruction, and the interrupt request must be valid 5TpC before the falling edge of the last clock cycle of the currently executing instruction.

For the ROMless mode, when the device samples a valid interrupt request, the next 48 (external) clock cycles are used to prioritize the interrupt, and push the two PC bytes and the FLAG register on the stack. The following nine cycles are used to fetch the interrupt vector from external memory. The first byte of the interrupt service routine is fetched beginning on the 58th TpC cycle following the internal sample point, which corresponds to the 63rd TpC cycle following the external interrupt sample point.

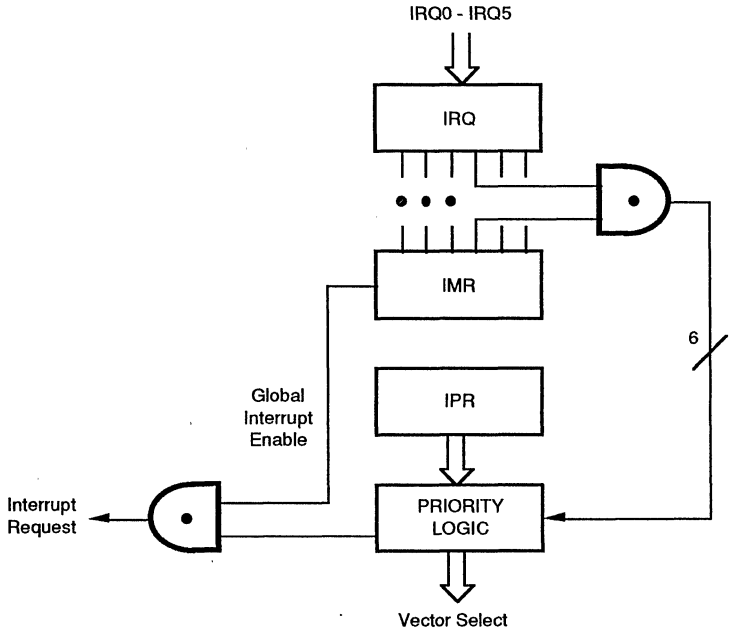


Figure 14. Interrupt Block Diagram

Clock. The ICE on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal, LC, ceramic resonator, or any suitable external clock source (XTAL1=Input, XTAL2=Output). The crystal should be AT cut, 1 MHz to 12 MHz max, and series resistance (RS) is less than or equal to 100 Ohms. The crystal should be connected across XTAL1 and XTAL2 using the recommended capacitors (10 pF < CL < 100 pF) from each pin to ground (Figure 15).

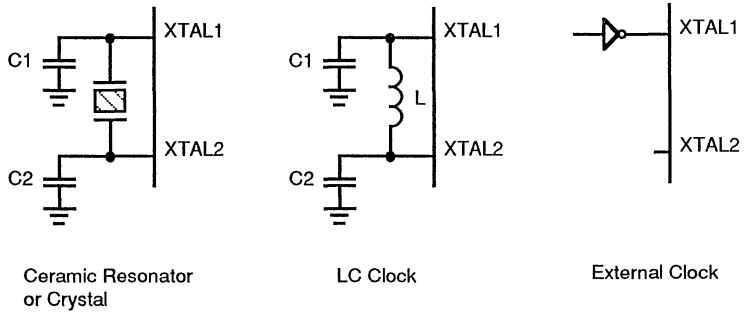


Figure 15. Oscillator Configuration

INSTRUCTION CYCLE TIMING

Figures 16 and 17 show instruction cycle timing for instruction fetched from external memory.

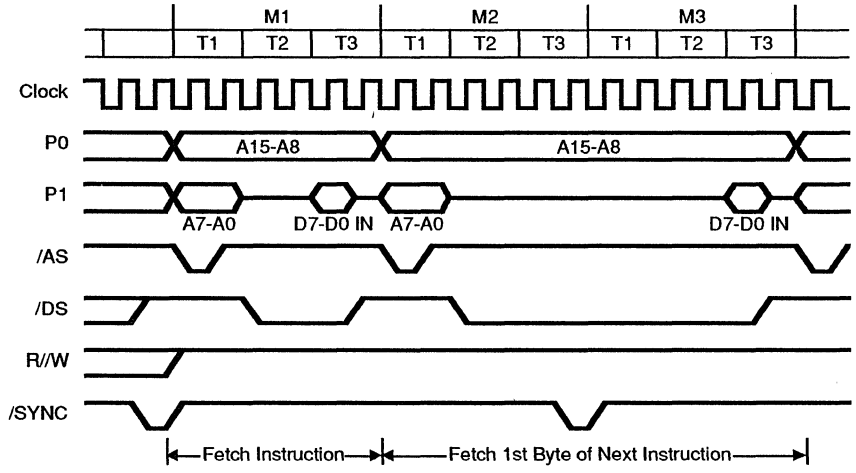


Figure 16. Instruction Cycle Timing (One-Byte Instructions)

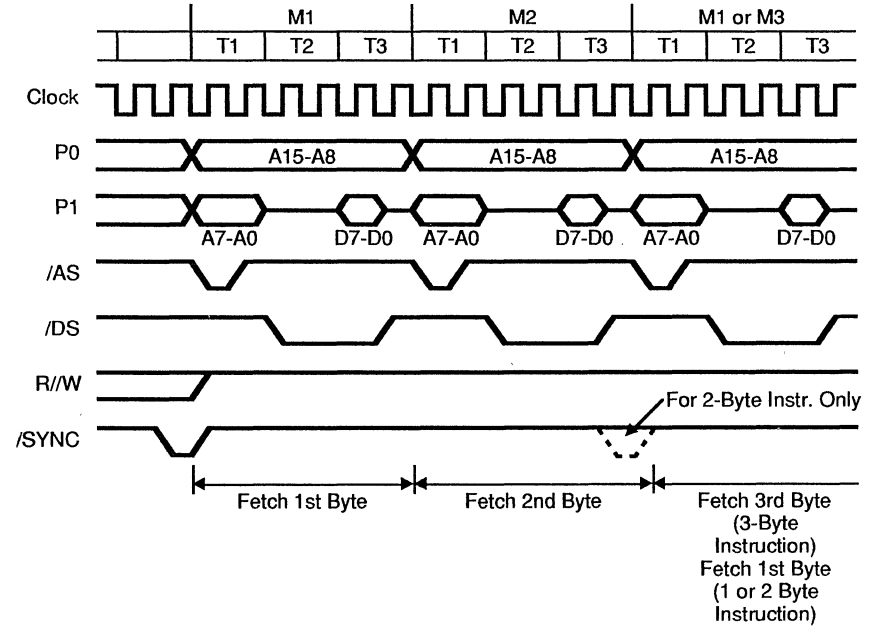


Figure 17. Instruction Cycle Timing (Two- and Three-Byte Instructions)

The addresses. Address Strobe (/AS) and Read Write (R//W) are output at the beginning of each machine cycle (Mn). The addresses output via Port) (if used) remain stable throughout the machine cycle, whereas addresses output via Port 1 remain valid only during MnT1. The addresses are guaranteed valid at the rising edge of /AS, which is used to latch the Port 1 output. Port 1 is placed in an input mode at the end of MnT1. The Data Strobe is output during MnT2, allowing data to be placed on the Port 1 bus. The Z8 accepts the data during MnT3 and /DS is terminated.

An instruction synchronization pulse /SYNC is output one clock pulse period prior to the beginning of an opcode fetch matching cycle (M1). This output is directly available on the 64-pin version of the Z8; whereas, on the 40-pin version, the Data Strobe pin outputs /SYNC only if external memory is not used.

Note that all instruction fetch cycles have the same machine timing regardless of whether the memory is internal or not. If configured for external memory and internal memory is referenced, the addresses are still output via Ports 0 and 1; however, /DS and R//W are inactive. If configured for internal memory only, Ports 0 and 1 are used for I/O, /DS outputs, and /SYNC; R//W is inactive.

The exception to the instruction fetch timing is during the opcode fetch of an instruction following the fetch of a one byte instruction. One-byte instructions require two machine cycles of execute. The pipeline causes the following opcode fetch to begin one machine cycle early.

External Memory or I/O Timing

When external memory is addressed, Ports 0 and 1 are configured to output the required number of address bits. Port 1 is used as a multiplexed address/data bus for AD7-AD0 and Port 0 outputs address bits A15-A8. The timing relationships for addressing external memory and I/O are illustrated in Figures 18, 19, 20 and 21. The main difference between these figures is that Figures 20/21 contain an added timing cycle (Tx) that extends external memory timing to allow for slower memory.

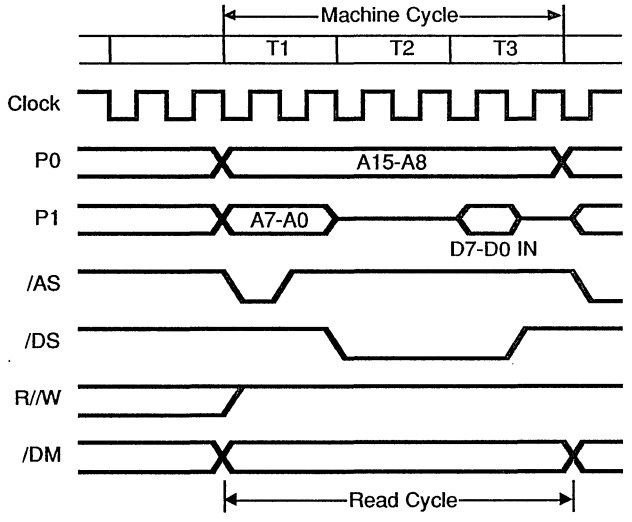


Figure 18. External Instruction Fetch, I/O, or Memory Read Cycle

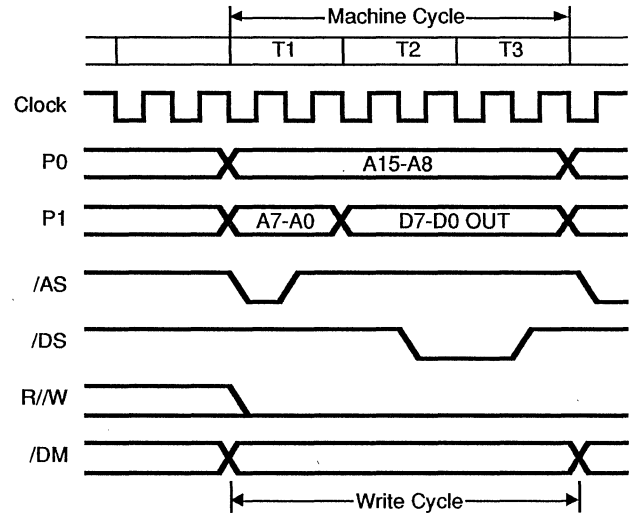


Figure 19. External I/O or Memory Write Cycle

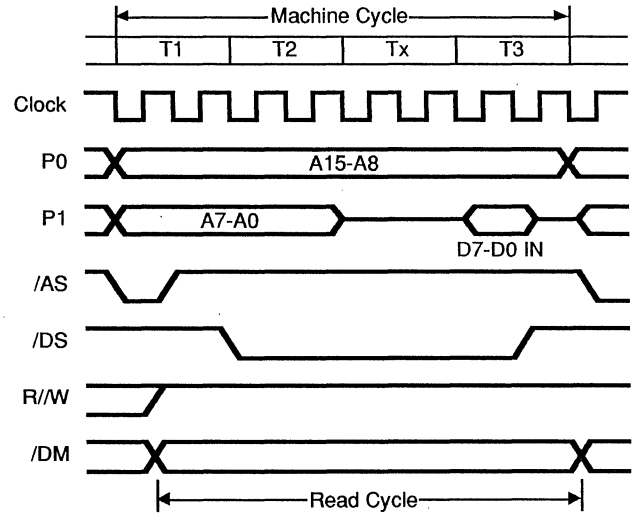


Figure 20. Extended External Instruction Fetch, I/O, or Memory Read Cycle

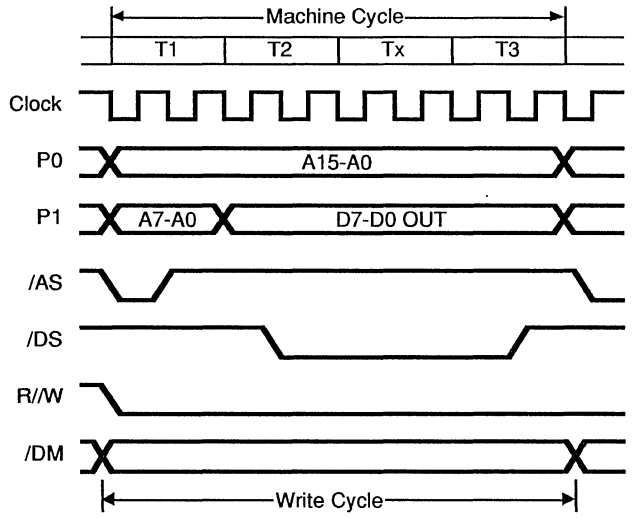


Figure 21. Extended External I/O or Memory Write Cycle

Address bits A15-A0 are valid on Ports 0 and 1 at the trailing edge of /AS for both the read and write memory cycles. Because Port 0 is not multiplexed, address bits A15-A8, if used, are present all through the read/write memory cycles.

During the read cycle, the input data must be valid on Port 1 at the trailing edge of the Data Strobe output (/DS). The Data Memory Select output (/DM) is used to select external data memory or external program memory. If selected, /DM is active during the execution of certain instructions.

During the write cycle, the address outputs follow the same timing relationships as for the read cycle. However, the output data is valid for the entire period /DS is active, and R//W is active (low) during the entire write cycle.

Interrupt requests are sampled before each instruction fetch cycle (Figure 22). First, external interrupt requests are sampled four clock periods prior to the active /AS pulse that corresponds to an instruction fetch cycle. Then, internal interrupt requests are samples one clock period preceding /AS.

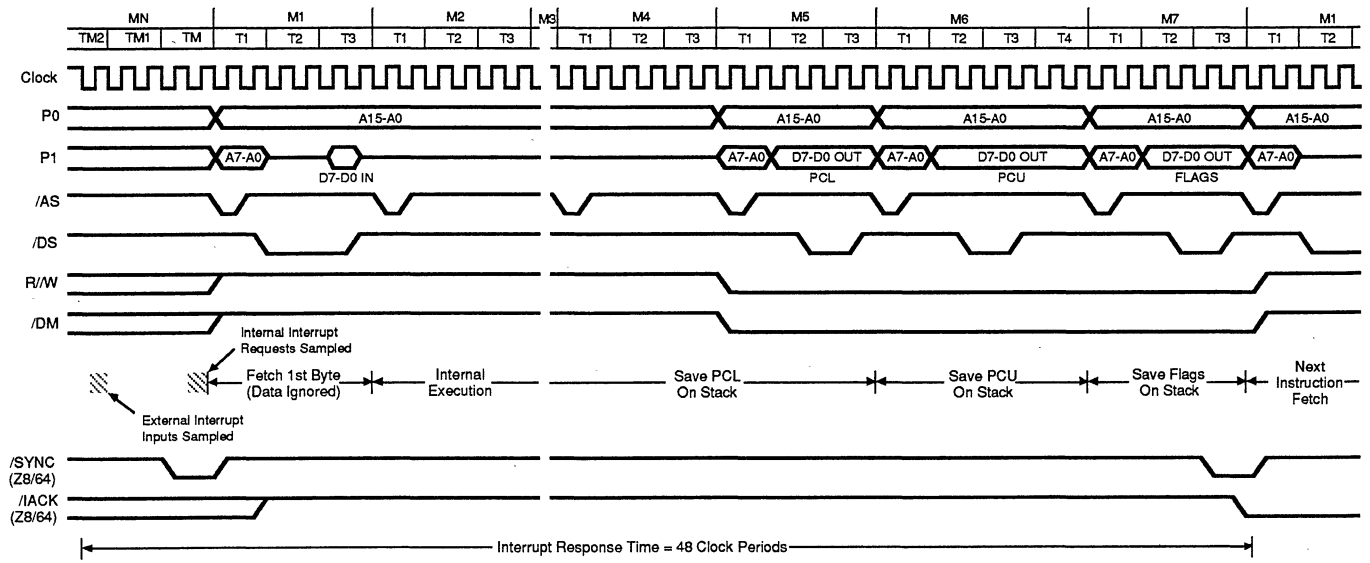


Figure 22. Interrupt Cycle Timing

If an interrupt request is set, the Z8 spends seven machine cycles (44 clock periods) resolving interrupt priorities, selecting the proper interrupt vector, and saving the program counter and flags on the stack. Although Figures 1-13 illustrate the timing for an external stack, the same timing is used for an internal stack. The total interrupt response time (including the external interrupt sample time) for an external interrupt is 48 clock periods, at which the first instruction of the interrupt service routine is fetched. When an interrupt request is detected in the Z8/64 development device, /IACK is activated (Low) and remains active until the first instruction of the interrupt service routine is fetched.

Reset Timing

The internal logic is initialized during reset if the Reset input is held low for at least 18 clock periods (Figure 23). During the time /RESET is Low, /AS is output at the internal clock rate, /DS is forced Low, R//W is inactive and Ports 0, 1 and 2 are placed in an input mode. /AS and /DS both low is normally a mutually exclusive condition; therefore, the coincidence of /AS Low and /DS Low can be used as a reset condition for other devices. Zilog Z-Bus® peripherals take advantage of this reset condition.

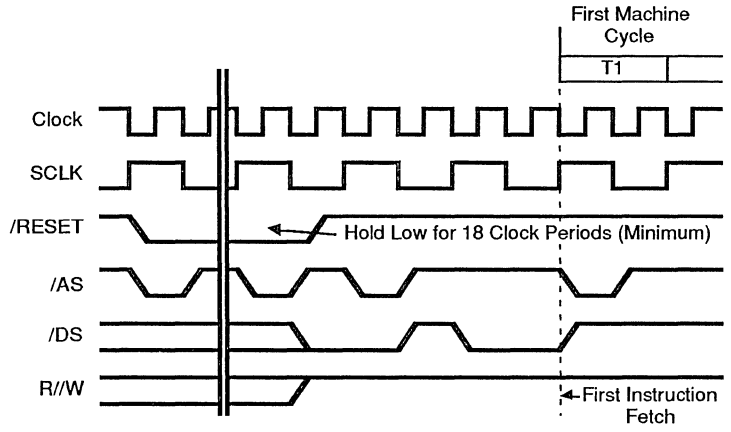


Figure 23. Reset Cycle Timing

Alternative Control Signal Uses

In addition to their uses in memory transfers, the control signals /AS, /DS and R//W are used in the following interface applications.

/AS can be modified to provide the /RAS (Row Address Strobe) signal for dynamic memory interface. /RAS can be derived from the trailing edge of /DS to the trailing edge of /AS.

/DS has several alternative uses: as a /CAS (Column Address Strobe) for dynamic memory interface, as a Chip Enable for memory and other interface devices, and as an Enable input for 3-state bus drivers/receivers for memory and interface devices.

R//W is used as a Write input to memory interfaces, and as an Early Status output to switch the direction of 3-state bus drivers/receivers.

ABSOLUTE MAXIMUM RATINGS

Symbol	Description	Min	Max	Units
V_{cc}	Supply Voltage*	-0.3	+7.0	V
T_{STG}	Storage Temp	-65°C	+150°	C
T_A	Oper Ambient Temp**			C

Notes:
* Voltages on all pins with respect to GND.
** See Ordering Information

Stress greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for an extended period may affect device reliability.

STANDARD TEST CONDITIONS

The characteristics listed below apply for standard test conditions as noted. All voltages are referenced to GND. Positive current flows into the referenced pin (Figure 24).

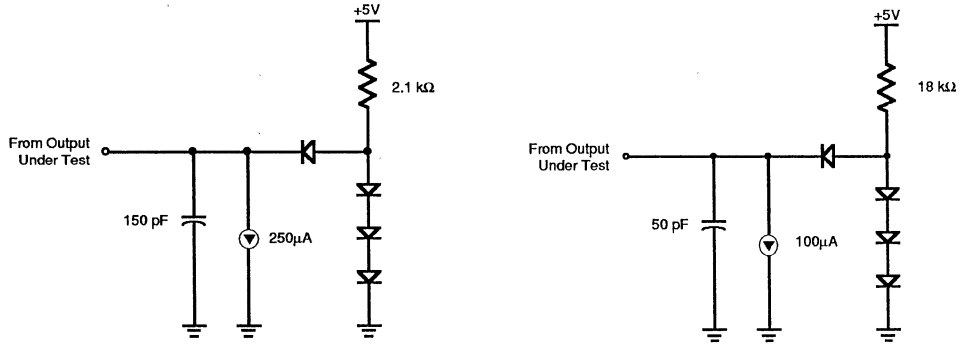


Figure 24. Test Load Diagrams

DC CHARACTERISTICS

Symbol	Parameter	TA = 0° C TO +70° C Min	Max	Units	Notes
	Max Input Voltage		7	V	I_{IN} 250 µA
V_{CH}	Clock Input High Voltage	$3.8V_{CC}$		V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.3	0.8	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	$2.0V_{CC}$		V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltge	2.4		V	I_{OH} = -250 uA
V_{OL}	Output Low Voltage		0.4	V	I_{OH} = +2.0 mA
V_{RH}	Reset Input High Voltage	$3.8V_{CC}$		V	
V_{RI}	Reset Input Low Voltage	-0.3	0.8	V	
I_{IL}	Input Leakage	-10	10	µA	0V V_{IN} +5.25V
I_{OL}	Output Leakage	-10	10	µA	0V V_{IN} +5.25V
I_{IR}	Reset Input Current		-50	µA	V_{CC} = +5.25V, V_{RL} = 0V
I_{CC}	Supply Current		180	mA	@ 12 MHz

Note:
For A11-A0, /MDS, /SYNC, /IACK and SCLK, IOH = -100 µA and I_{OL} = 1 mA

AC CHARACTERISTICS

External I/O or Memory Read or Write Timing Diagram

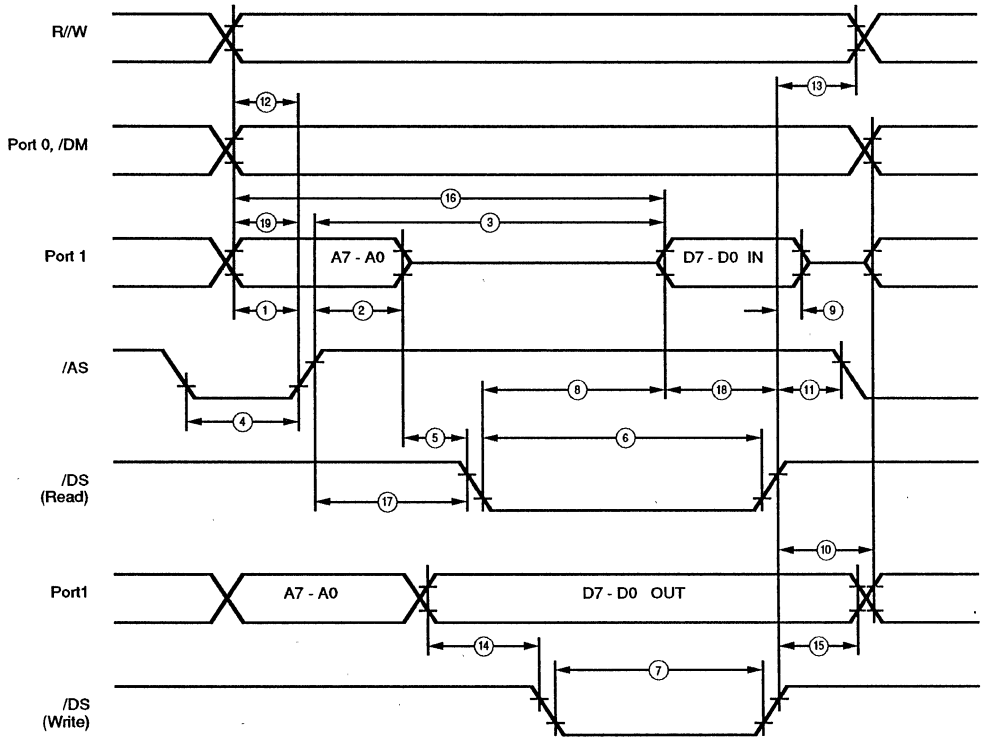


Figure 25. External I/O or Memory Read or Write Timing

AC CHARACTERISTICS

External I/O or Memory Read or Write Timing Table

No	Symbol	Parameter	TA = 0° C to +70° C 8 MHz Min	8 MHz Max	12 MHz Min	12 MHz Max	Units	Notes
1	TdA(AS)	Address Valid to /AS rise Delay	50		35		ns	[2,3]
2	TdAS(A)	/AS rise to Address Float Delay	70		45		ns	[2,3]
3	TdAS(DR)	/AS rise to Read Data Req'd Valid		360		220	ns	[1,2,3]
4	TwAS	/AS Low Width	80		55		ns	[2,3]
5	TdAZ(DS)	Address Float to /DS fall	0		0		ns	
6	TwDSR	/DS (Read) Low Width	250		186		ns	[1,2,3]
7	TwDSW	/DS (Write) Low Width	160		110		ns	[1,2,3]
8	TdDSR(DR)	/DS fall to Read Data Req'd Valid		200		130	ns	[1,2,3]
9	ThDR(DS)	Read Data to /DS rise Hold Time	0		0		ns	[2,3]
10	TdDS(A)	/DS rise to Address Active Delay	70		45		ns	[2,3]
11	TdDS(AS)	/DS rise to /AS fall Delay	70		55		ns	[2,3]
12	TdR/W(AS)	R//W Valid to /AS rise Delay	50		30		ns	[2,3]
13	TdDS(R/W)	/DS rise to R//W Not Valid	60		35		ns	[2,3]
14	TdDW(DSW)	Write Data Valid to /DS fall (Write) Delay	50		35		ns	[2,3]
15	TdDS(DW)	/DS rise to Write Data Not Valid Delay	50		35		ns	[2,3]
16	TdA(DR)	Address Valid to Read Data Req'd Valid		410		255	ns	[1,2,3]
17	TdAS(DS)	/AS rise to /DS fall Delay	80		55		ns	[2,3]

Notes:

[1] When using extended memory timing add 2 TpC.

[2] Timing numbers given are for minimum TpC.

[3] See clock cycle dependent characteristics table.

Standard Test Load

All timing references use 2.0V for a logic 1 and 0.8V for a logic 0.

AC CHARACTERISTICS
Additonal Timing Diagram

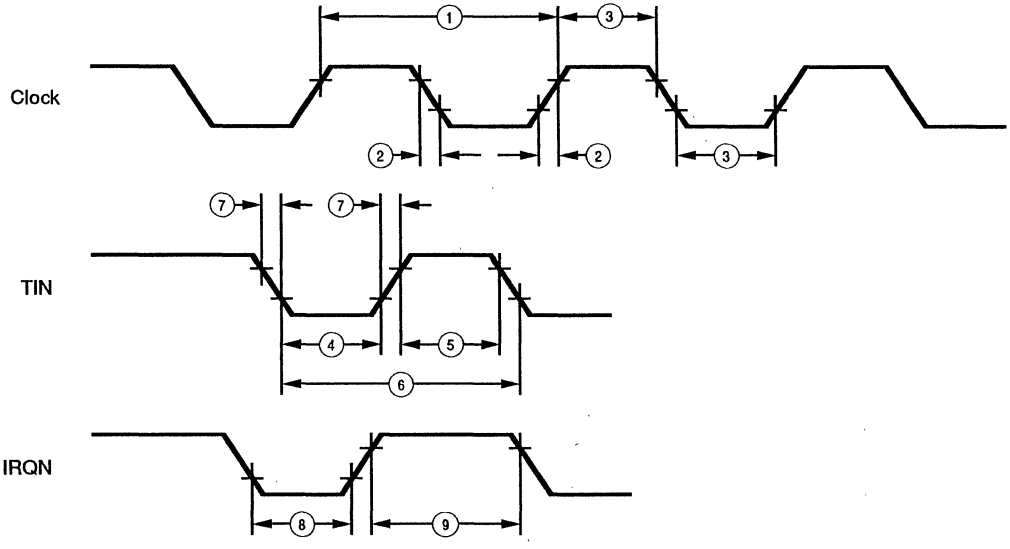


Figure 26. Additional Timing

AC CHARACTERISTICS
Additional Timing Table

No	Symbol	Parameter	T_A = 0° C to +70° C 8 MHz Min	8 MHz Max	12 MHZ Min	12 MHZ Max	Units	Notes
1	TpC	Input Clock Period	125	1000	83	1000	ns	[1]
2	TrC,TfC	Clock Imput Rise & Fall Times		25		15	ns	[1]
3	TwC	Input Clock Width	37		26		ns	[1]
4	TwTinL	Timer Input Low Width	100		70		ns	[2]
5	TwTinH	Timer Input High Width	3TpC		3TpC			[2]
6	TpTin	Timer Input Period	8TpC		8TpC			[2]
7	TrTin,TfTin	Timer Input Rise & Fall Times		100		100	ns	[2]
8A	TwIL	Interrupt Request Input Low Times		100		70	ns	[2,4]
8B	TwIL	Interrupt Request Input Low Times	3TpC		3TpC			[2,5]
9	TwIH	Interrupt Request Input High Times	3TpC		3TpC			[2,3]

Notes:
[1] Clock timing references use 3.8V for a logic 1 and 0.8V for a logic 0.
[2] Timing references use 2.0V for a logic 1 and 0.8V for a logic 0.
[3] Interrupt references request via Port 3.
[4] Interrupt request via Port 3 (P31-P33)`.
[5] Interrupt request via Port 30.

AC CHARACTERISTICS
Memory Port Timing Table

No	Symbol	Parameter	T_A = 0° C TO +70° C 8 MHz Min	8 MHz Max	12 MHz Min	12 MHz Max	Units	Notes
1	TdA(DI)	Address Valid to Data Input Delay		460		320	ns	1,2
2	ThDI(A)	Data In Hold Time	0		0		ns	1

Notes:
1. Test load 2
2. This is a clock cycle dependent parameter. For frequencies other than the maximum, use the following formula: 5TpC - 95.

AC CHARACTERISTICS
Memory Port Timing Diagram

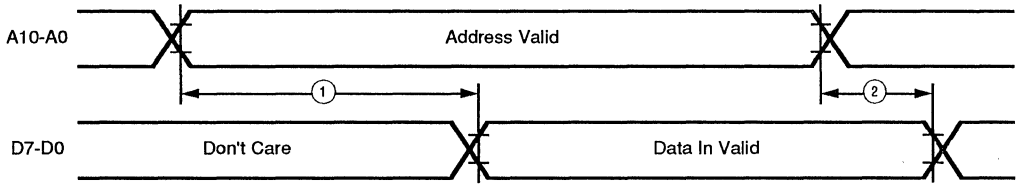


Figure 27. Memory Port Timing

AC CHARACTERISTICS
Handshake Timing Diagram

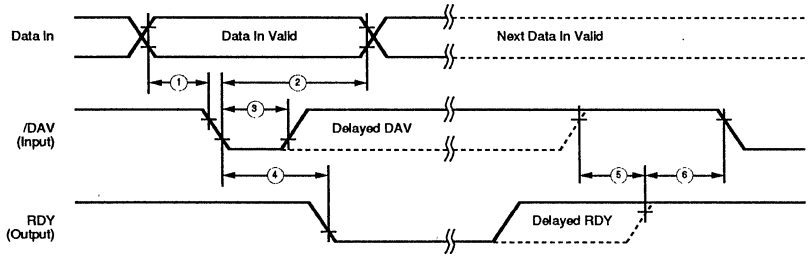


Figure 28. Input Handshake Timing

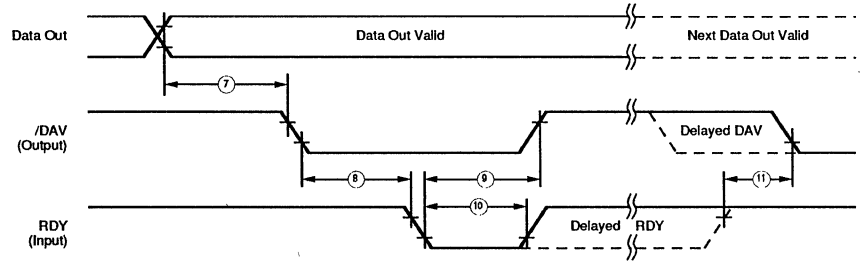


Figure 29. Output Handshake Timing

AC CHARACTERISTICS
Handshake Timing Table

No	Symbol	Parameter	T_A = 0° C TO +70° C 8 MHz Min	8 MHz Max	12 MHz Min	12 MHz Max	Notes
1	TsDI(DAV)	Data In Setup Time	0		0		
2	ThDI(DAV)	Data In Hold Time	230		160		
3	TwDAV	Data Available Width	175		120		
4	TdDAVI(RDY)	DAV fall to RDY fall Delay		175		120	1,2
5	TcLDAVO(RDY)	DAV fall to RDY fall Delay	0		0		1,3
6	TdDAVIr(RDY)	DAV rise to RDY rise Delay		175		120	1,2
7	TdDAVOr(RDY)	DAV rise to RDY rise Delay	0		0		1,3
8	TdDO(DAV)	Data Out to DAV fall Delay	50		30		1
9	TdRDYI(DAV)	RDY fall to DAV rise Delay	0	200	0	140	1

Notes:
1. Test load 1
2. Input handshake
3. Output handshake

All timing references use 2.0 V for a logic 1 and 0.8 V for logic 0.
Units in nanoseconds

CLOCK DEPENDENT
AC Characteristics

No	Symbol	Equation
1	TdA(AS)	0.40TpC + 0.32
2	TdAS(A)	0.59TpC - 3.25
3	TdAS(DR)	2.38TpC + 6.14
4	TwAS	0.66TpC - 1.65
6	TwDSR	2.33TpC - 10.56
7	TwDSW	1.27TpC + 1.67
8	TdDSR(DR)	1.97TpC - 42.5
10	TdDS(A)	0.8TpC
11	TdDS(AS)	0.59TpC - 3.14
12	TdR/W(AS)	0.4TpC
13	TdDS(R/W)	0.8TpC - 15
14	TdDW(DSW)	0.4TpC
15	TdDS(DW)	0.88TpC - 19
16	TdA(DR)	4TpC - 20
17	TdAS(DS)	0.91TpC - 10.7
18	TsDI(DS)	0.8TpC - 10
19	TdDM(AS)	0.9TpC - 26.3

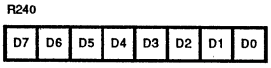


Figure 30. Serial I/O Register (F0H: Read/Write)

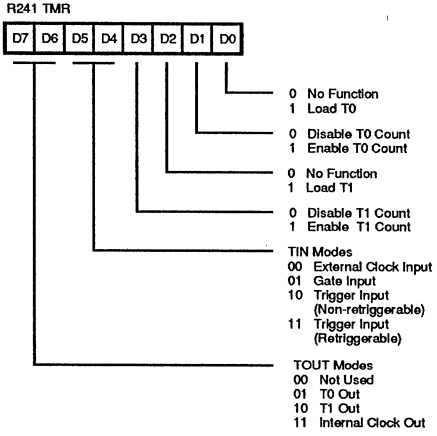


Figure 31. Timer Mode Register (F1H: Read/Write)

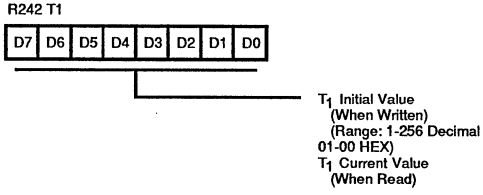


Figure 32. Counter/Timer 1 Register (F2H: Read/Write)

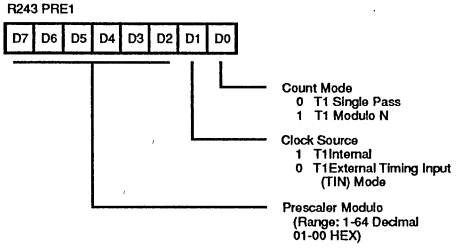


Figure 33. Prescaler 1 Register (F3H: Write Only)

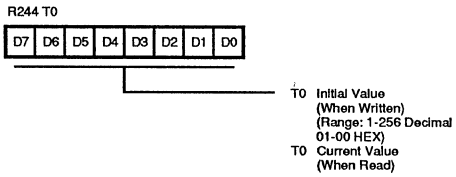


Figure 34. Counter/Timer 0 Register (F4H: Read/Write)

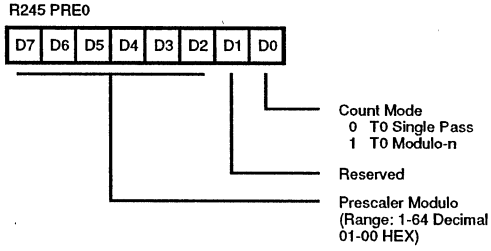


Figure 35. Prescaler 0 Register (F5H: Write Only)

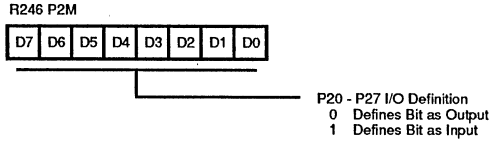


Figure 36. Port 2 Mode Register (F6H: Write Only)

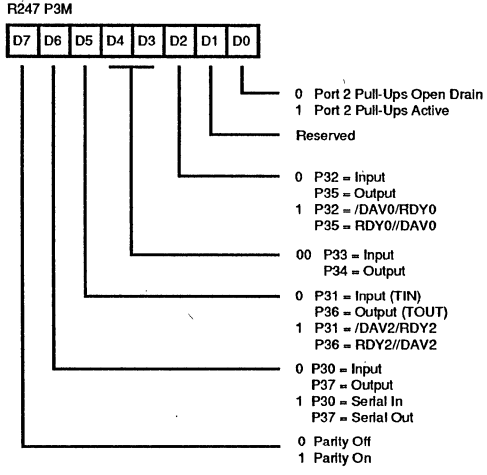


Figure 37. Port 3 Mode Register (F7H: Write Only)

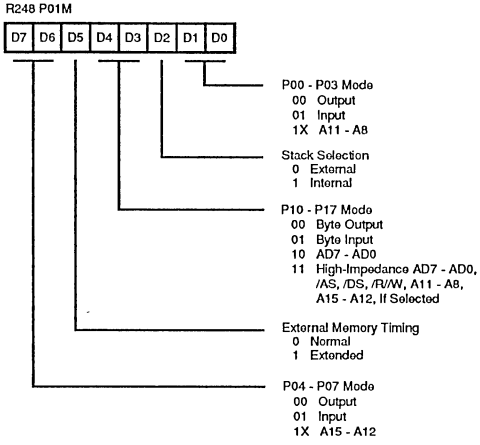


Figure 38. Port 0 and 1 Mode Register (F8H: Write Only)

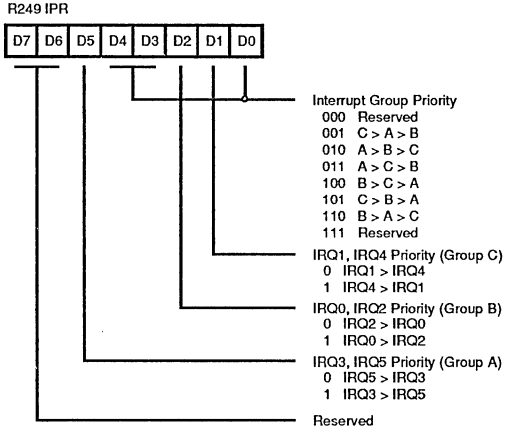


Figure 39. Interrupt Priority Register (F9H: Write Only)

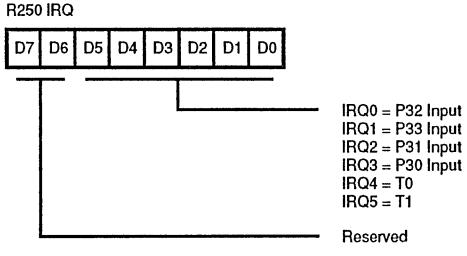


Figure 40. Interrupt Request Register (FAH: Read/Write)

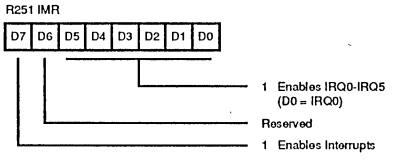


Figure 41. Interrupt Mask Register (FBH: Read/Write)

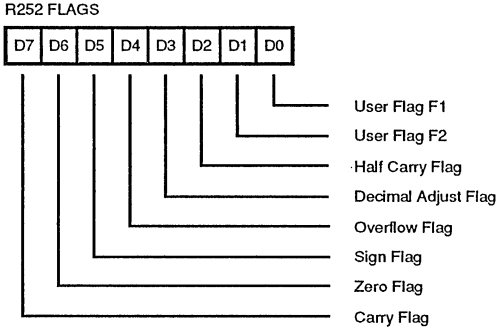


Figure 42. Flag Register (FCH: Read/Write)

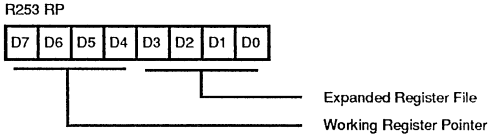


Figure 43. Register Pointer Register (FDH: Read/Write)

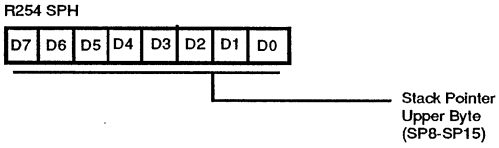


Figure 44. Stack Pointer Register (FEH: Read/Write)

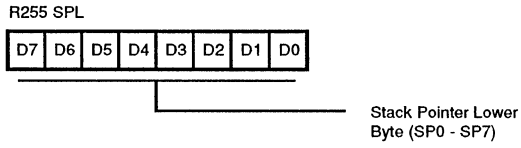


Figure 45. Stack Pointer Register (FFH: Read/Write)

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

Symbol	Meaning
IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

Symbol	Meaning
dst	Destination location or contents
src	Source location or contents
cc	Condition code
@	Indirect address prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flag register (Control Register 252)
RP	Register Pointer (R253)
IMR	Interrupt mask register (R251)

Flags. Control register (R252) contains the following six flags:

Symbol	Meaning
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

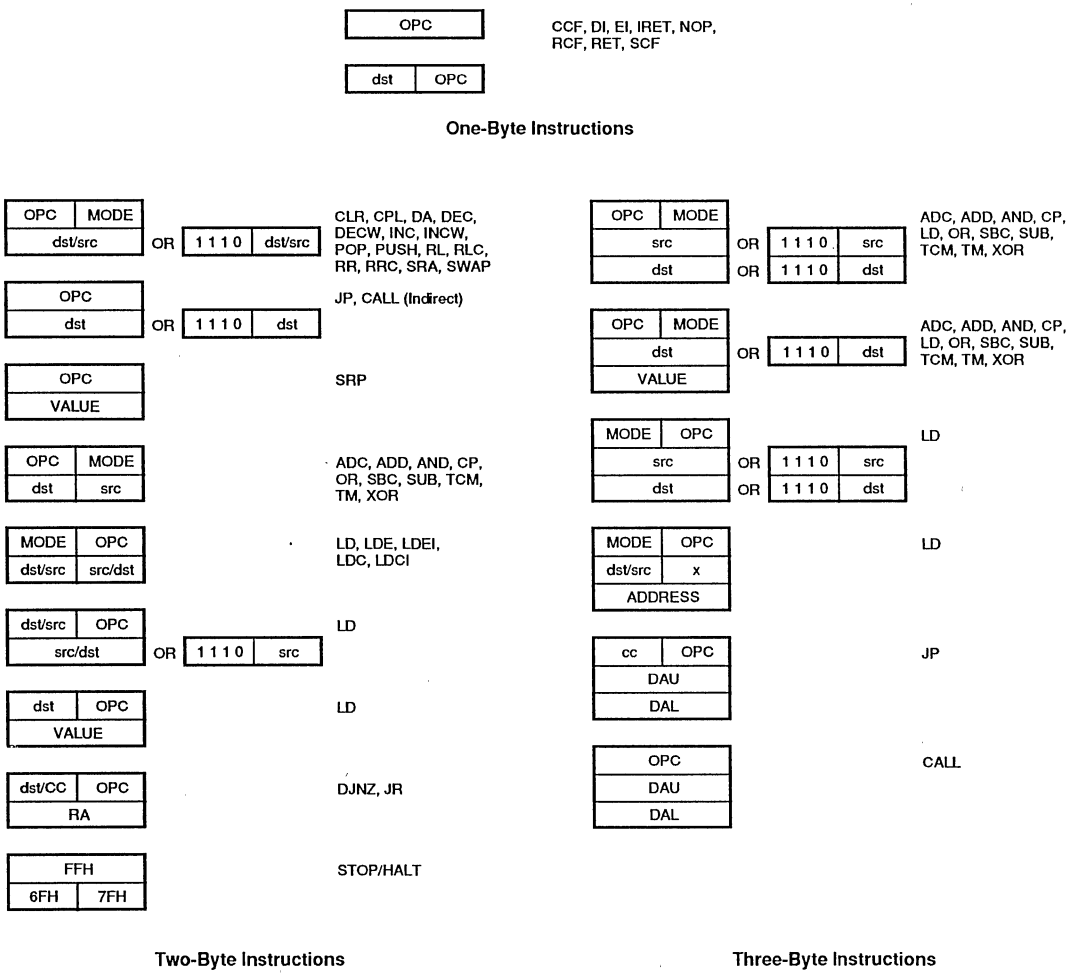
Affected flags are indicated by:

0	Clear to zero
1	Set to one
*	Set to clear according to operation
-	Unaffected
x	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000		Always True	
0111	C	Carry	C = 1
1111	NC	No Carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not Zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No Overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not Equal	Z = 0
1001	GE	Greater Than or Equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater Than	[Z OR (S XOR V)] = 0
0010	LE	Less Than or Equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned Greater Than or Equal	C = 0
0111	ULT	Unsigned Less Than	C = 1
1011	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned Less Than or Equal	(C OR Z) = 1
0000		Never True	

INSTRUCTION FORMATS



INSTRUCTION SUMMARY

Note: Assignment of a value is indicated by the symbol " ← ". For example:

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr (n)" is used to refer to bit (n) of a given operand location. For example:

dst (7)

refers to bit 7 of the destination operand.

INSTRUCTION SUMMARY

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst, src dst←dst + src +C	†		1[ ]	*	*	*	*	0	*
ADD dst, src dst←dst + src	†		0[ ]	*	*	*	*	0	*
AND dst, src dst←dst AND src	†		5[ ]	-	*	*	0	-	-
CALL dst SP←SP - 2 @SP←PC, PC←dst	DA IRR		D6 D4	-	-	-	-	-	-
CCF C←NOT C			EF	*	-	-	-	-	-
CLR dst dst←0	R IR		B0 B1	-	-	-	-	-	-
COM dst dst←NOT dst	R IR		60 61	-	*	*	0	-	-
CP dst, src dst - src	†		A[ ]	*	*	*	*	-	-
DA dst dst←DA dst	R IR		40 41	*	*	*	X	-	-
DEC dst dst←dst - 1	R IR		00 01	-	*	*	*	-	-
DECW dst dst←dst - 1	RR IR		80 81	-	*	*	*	-	-
DI IMR(7)←0			8F	-	-	-	-	-	-
DJNZr, dst r←r - 1 if r ≠ 0 PC←PC + dst Range: +127, -128	RA		rA r = 0 - F	-	-	-	-	-	-
EI IMR(7)←1			9F	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
INC dst dst←dst + 1	r R IR		rE r = 0 - F 20 21	-	*	*	*	-	-
INCW dst dst←dst + 1	RR IR		A0 A1	-	*	*	*	-	-
IRET FLAGS←@SP; SP←SP + 1 PC←@SP; SP←SP + 2; IMR(7)←1			BF	*	*	*	*	*	*
JP cc, dst if cc is true PC←dst	DA IRR		cD c = 0 - F 30	-	-	-	-	-	-
JR cc, dst if cc is true, PC←PC + dst Range: +127, -128	RA		cB c = 0 - F	-	-	-	-	-	-
LD dst, src dst←src	r r R r X r Ir R R IR IR	Im R r X r Ir r R IR IM IM R	rC r8 r9 r = 0 - F C7 D7 E3 F3 E4 E5 E6 E7 F5	-	-	-	-	-	-
LDC dst, src	r	Irr	C2	-	-	-	-	-	-
LDCI dst, src dst←src r←r +1; rr←rr + 1	Ir	Irr	C3	-	-	-	-	-	-

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
NOP			FF	-	-	-	-	-	-
OR dst, src dst←dst OR src	†		4[ ]	-	*	*	0	-	-
POP dst dst←@SP; SP←SP + 1	R IR		50 51	-	-	-	-	-	-
PUSH src SP←SP - 1; @SP←src		R IR	70 71	-	-	-	-	-	-
RCF C←0			CF	0	-	-	-	-	-
RET PC←@SP; SP←SP + 2			AF	-	-	-	-	-	-
RL dst	R IR		90 91	*	*	*	*	-	-
RLC dst	R IR		10 11	*	*	*	*	-	-
RR dst	R IR		E0 E1	*	*	*	*	-	-
RRC dst	R IR		C0 C1	*	*	*	*	-	-
SBC dst, src dst←dst←src←C	†		3[ ]	*	*	*	*	1	*
SCF C←1			DF	1	-	-	-	-	-
SRA dst	R IR		D0 D1	*	*	*	0	-	-
SRP src RP←src		Im	31	-	-	-	-	-	-

Instruction and Operation	Address Mode dst	Address Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
SUB dst, src dst←dst←src	†		2[ ]	*	*	*	*	1	*
SWAP dst	R IR		F0 F1	X	*	*	X	-	-
TCM dst, src (NOT dst) AND src	†		6[ ]	-	*	*	0	-	-
TM dst, src dst AND src	†		7[ ]	-	*	*	0	-	-
XOR dst, src dst←dst XOR src	†		B[ ]	-	*	*	0	-	-

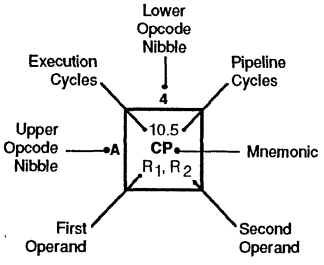
† These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a '[ ]' in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Address Mode dst	Address Mode src	Lower Opcode Nibble
r	r	[2]
r	Ir	[3]
R	R	[4]
R	IR	[5]
R	IM	[6]
IR	IM	[7]

OPCODE MAP

Upper Nibble (Hex) \ Lower Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R1	6.5 **DEC** IR1	6.5 **ADD** r1, r2	6.5 **ADD** r1, Ir2	10.5 **ADD** R2, R1	10.5 **ADD** IR2, R1	10.5 **ADD** R1, IM	10.5 **ADD** IR1, IM	6.5 **LD** r1, R2	6.5 **LD** r2, R1	12/10.5 **DJNZ** r1, RA	12/10.0 **JR** cc, RA	6.5 **LD** r1, IM	12.10.0 **JP** cc, DA	6.5 **INC** r1	
1	6.5 **RLC** R1	6.5 **RLC** IR1	6.5 **ADC** r1, r2	6.5 **ADC** r1, Ir2	10.5 **ADC** R2, R1	10.5 **ADC** IR2, R1	10.5 **ADC** R1, IM	10.5 **ADC** IR1, IM								
2	6.5 **INC** R1	6.5 **INC** IR1	6.5 **SUB** r1, r2	6.5 **SUB** r1, Ir2	10.5 **SUB** R2, R1	10.5 **SUB** IR2, R1	10.5 **SUB** R1, IM	10.5 **SUB** IR1, IM								
3	8.0 **JP** IRR1	6.1 **SRP** IM	6.5 **SBC** r1, r2	6.5 **SBC** r1, Ir2	10.5 **SBC** R2, R1	10.5 **SBC** IR2, R1	10.5 **SBC** R1, IM	10.5 **SBC** IR1, IM								
4	8.5 **DA** R1	8.5 **DA** IR1	6.5 **OR** r1, r2	6.5 **OR** r1, Ir2	10.5 **OR** R2, R1	10.5 **OR** IR2, R1	10.5 **OR** R1, IM	10.5 **OR** IR1, IM								
5	10.5 **POP** R1	10.5 **POP** IR1	6.5 **AND** r1, r2	6.5 **AND** r1, Ir2	10.5 **AND** R2, R1	10.5 **AND** IR2, R1	10.5 **AND** R1, IM	10.5 **AND** IR1, IM								
6	6.5 **COM** R1	6.5 **COM** IR1	6.5 **TCM** r1, r2	6.5 **TCM** r1, Ir2	10.5 **TCM** R2, R1	10.5 **TCM** IR2, R1	10.5 **TCM** R1, IM	10.5 **TCM** IR1, IM								
7	10/12.1 **PUSH** R2	12/14.1 **PUSH** IR2	6.5 **TM** r1, r2	6.5 **TM** r1, Ir2	10.5 **TM** R2, R1	10.5 **TM** IR2, R1	10.5 **TM** R1, IM	10.5 **TM** IR1, IM								
8	10.5 **DECW** RR1	10.5 **DECW** IR1	12.0 **LDE** r1, Irr2	18.0 **LDEI** Ir1, Irr2		**J85**										6.1 **DI**
9	6.5 **RL** R1	6.5 **RL** IR1	12.0 **LDE** r2, Irr1	18.0 **LDEI** Ir2, Irr1												6.1 **EI**
A	10.5 **INCW** RR1	10.5 **INCW** IR1	6.5 **CP** r1, r2	6.5 **CP** r1, Ir2	10.5 **CP** R2, R1	10.5 **CP** IR2, R1	10.5 **CP** R1, IM	10.5 **CP** IR1, IM								14.0 **RET**
B	6.5 **CLR** R1	6.5 **CLR** IR1	6.5 **XOR** r1, r2	6.5 **XOR** r1, Ir2	10.5 **XOR** R2, R1	10.5 **XOR** IR2, R1	10.5 **XOR** R1, IM	10.5 **XOR** IR1, IM								16.0 **IRET**
C	6.5 **RRC** R1	6.5 **RRC** IR1	12.0 **LDC** r1, Irr2	18.0 **LDCI** Ir1, Irr2				10.5 **LD** r1,x,R2								6.5 **RCF**
D	6.5 **SRA** R1	6.5 **SRA** IR1	12.0 **LDC** r2, Irr1	18.0 **LDCI** Ir2, Irr1	20.0 **CALL*** IRR1		20.0 **CALL** DA	10.5 **LD** r2,x,R1								6.5 **SCF**
E	6.5 **RR** R1	6.5 **RR** IR1		6.5 **LD** r1, IR2	10.5 **LD** R2, R1	10.5 **LD** IR2, R1	10.5 **LD** R1, IM	10.5 **LD** IR1, IM								6.5 **CCF**
F	8.5 **SWAP** R1	8.5 **SWAP** IR1		6.5 **LD** Ir1, r2		10.5 **LD** R2, IR1										6.0 **NOP**
Bytes per Instruction	2			3					2					3	1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_2 = Dst address
R_1 or r_2 = Src address

Sequence:
Opcode, First Operand,
Second Operand

Note: The blank are not defined.

* 2-byte instruction appears as a 3-byte instruction

Z8671 Z8® MCU with BASIC/Debug Interpreter

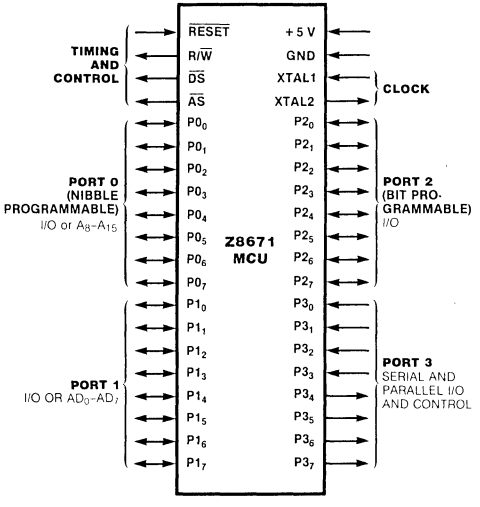
FEATURES

- The Z8671 MCU is a complete microcomputer preprogrammed with a BASIC/Debug interpreter. Interaction between the interpreter and its user is provided through an on-board UART.
- BASIC/Debug can directly address the Z8671's internal registers and all external memory. It provides quick examination and modification of any external memory location or I/O port.
- The BASIC/Debug interpreter can call machine language subroutines to increase execution speed.
- The Z8671's auto start-up capability allows a program to be executed on power-up or Reset without operator intervention.
- Single +5V power supply—all I/O pins TTL-compatible.
- 8 MHz

GENERAL DESCRIPTION

The Z8671 Single-Chip Microcomputer (MCU) is one of a line of preprogrammed chips—in this case with a BASIC/Debug interpreter in ROM—offered by Zilog. As a member of the Z8 Family of microcomputers, it offers the same abundance of resources as the other Z8 microcomputers.

Because the BASIC/Debug interpreter is already part of the chip circuit, programming is made much easier. The Z8671 MCU thus offers a combination of software and hardware that is ideal for many industrial control applications. The Z8671 MCU allows fast hardware tests and bit-by-bit examination and modification of memory location, I/O ports,



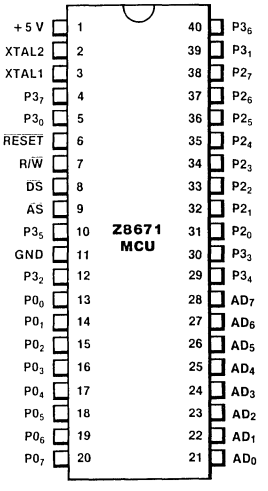


Figure 2a. 40-pin Dual-In-Line Package (DIP), Pin Assignments

or registers. It also allows bit manipulation and logical operations. A self-contained line editor supports interactive debugging, further speeding up program development.

The BASIC/Debug interpreter, a subset of Dartmouth BASIC, operates with three kinds of memory: on-chip registers and external ROM or RAM. The BASIC/Debug interpreter is located in the 2K bytes of on-chip ROM.

Additional features of the Z8671 MCU include the ability to call machine language subroutines to increase execution speed and the ability to have a program execute on power-up or Reset, without operator intervention.

Maximum memory addressing capabilities include 62K bytes of external program memory and 62K bytes of data memory with program storage beginning at location 800_H. This provides up to 124K bytes of useable memory space. Very few 8-bit microcomputers can directly access this amount of memory.

Each Z8671 Microcomputer has 32 I/O lines, a 144-byte register file, an on-board UART, and two counter/timers.

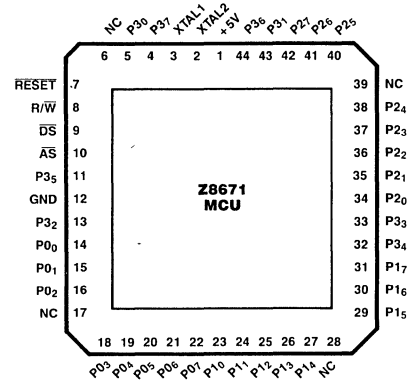


Figure 2b. 44-pin Chip Carrier, Pin Assignments

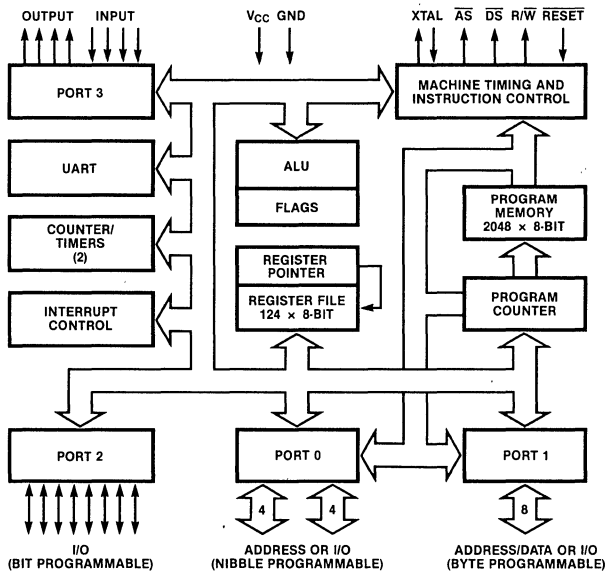


Figure 3. Functional Block Diagram

ARCHITECTURE

Z8671 architecture is characterized by a flexible I/O scheme, an efficient register and address space structure, and a number of ancillary features that are helpful in many applications.

Microcomputer applications demand powerful I/O capabilities. The Z8671 fulfills this with 32 pins dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an address/data bus for interfacing external memory.

Because the multiplexed address/data bus is merged with the I/O-oriented ports, the Z8671 can assume many different memory and I/O configurations. These configurations range from a self-contained microcomputer to a microprocessor that can address 124K bytes of external memory.

Three basic address spaces are available to support this wide range of configurations: program memory (internal and external), data memory (external) and the register file (internal). The 144-byte random-access register file is composed of 124 general-purpose registers, four I/O port registers, and 16 control and status registers.

To unburden the program from coping with real-time problems such as serial data communication and counting/timing, an asynchronous receiver/transmitter (UART) and two counter/timers with a large number of userselectable modes are offered on-chip. Hardware support for the UART is minimized because one of the on-chip timers supplies the bit rate.

PIN DESCRIPTION

$\overline{AS}$. *Address Strobe* (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Addresses output via Port 1 for all external program or data memory transfers are valid at the trailing edge of $\overline{AS}$. Under program control, $\overline{AS}$ can be placed in the high-impedance state along with Ports 0 and 1, Data Strobe, and Read/Write.

$\overline{DS}$. *Data Strobe* (output, active Low). Data Strobe is activated once for each external memory transfer.

$P0_0$-$P0_7$, $P1_0$-$P1_7$, $P2_0$-$P2_7$, $P3_0$-$P3_7$. *I/O Port Lines* (input/outputs, TTL-compatible). These 32 lines are divided into four 8-bit I/O ports that can be configured under program control for I/O or external memory interface.

$\overline{RESET}$. *Reset* (input, active Low). $\overline{RESET}$ initializes the Z8671. When $\overline{RESET}$ is deactivated, program execution begins from internal program location $000C_H$.

R/$\overline{W}$. *Read/Write* (output). R/$\overline{W}$ is Low when the Z8671 is writing to external program or data memory.

XTAL1, XTAL2. *Crystal 1, Crystal 2* (time-base input and output). These pins connect a parallel-resonant crystal (8 MHz maximum) or an external single-phase clock (8 MHz maximum) to the on-chip clock oscillator and buffer.

ADDRESS SPACES

Program Memory. The Z8671's 16-bit program counter can address 64K bytes of program memory space. Program memory consists of 2K bytes of internal ROM and up to 62K bytes of external ROM, EPROM, or RAM. The first 12 bytes of program memory are reserved for interrupt vectors (Figure 4). These locations contain six 16-bit vectors that correspond to the six available interrupts. The BASIC/Debug interpreter is located in the 2K bytes of internal ROM. The interpreter begins at address 12 and extends to 2047.

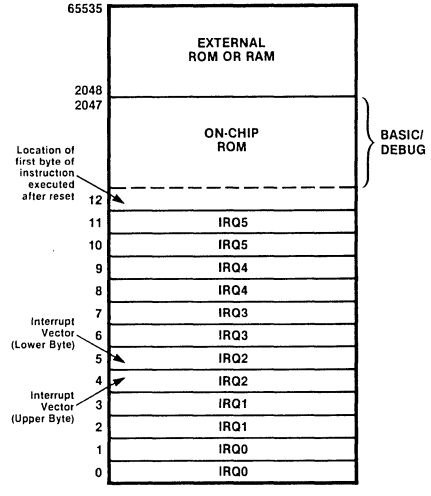


Figure 4. Program Memory Map

Data Memory. The Z8671 can address up to 62K bytes of external data memory beginning at location 2048 (Figure 5). External data memory may be included with, or separated from, the external program memory space. DM, an optional I/O function that can be programmed to appear on pin $P3_4$, is used to distinguish data and program memory space.

Register File. The 144-byte register file may be accessed by BASIC programs as memory locations 0-127 and 240-255. The register file includes four I/O port registers (R0-R3), 124 general-purpose registers (R4-R127), and 16 control and status registers (Figure 6).

The BASIC/Debug Interpreter uses many of the general-purpose registers as pointers, scratch workspace, and internal variables. Consequently, these registers cannot be used by a machine language subroutine or other user programs. On power-up/Reset, BASIC/Debug searches for external RAM memory and checks for an auto start-up program. In a non-destructive method, memory is tested at relative location $xxFD_H$. When BASIC/Debug discovers RAM in the system, it initializes the pointer registers to mark the boundaries between areas of memory that are assigned specific uses. The top page of RAM is allocated for the line buffer, variable storage, and the GOSUB stack. Figure 7a illustrates the contents of the general-purpose registers in the Z8671 system with external RAM. When BASIC/Debug tests memory and finds no RAM, it uses an internal stack and shares register space with the input line buffer and variables. Figure 7b illustrates the contents of the general-purpose registers in the Z8671 system without external RAM.

Stacks. Either the internal register file or the external data memory can be used for the stack. A 16-bit Stack Pointer (R254 and R255) is used for the external stack, which can reside anywhere in data memory between location 2048 and 65535. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 124 general-purpose registers (R4-R127).

Register Addressing. Z8671 instructions can directly or indirectly access registers with an 8-bit address field. The Z8671 also allows short 4-bit register addressing using the Register Pointer, which is one of the control registers. In the 4-bit mode, the register file is divided into nine working-register groups, each group consisting of 16 contiguous registers (Figure 8). The Register Pointer addresses the starting location of the active working-register group.

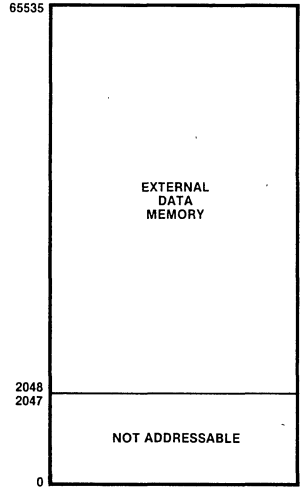


Figure 5. Data Memory Map

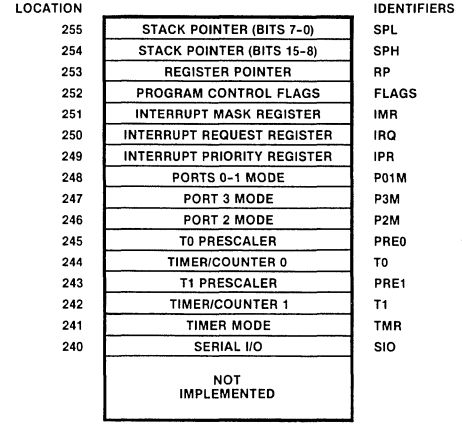


Figure 6. Control and Status Registers

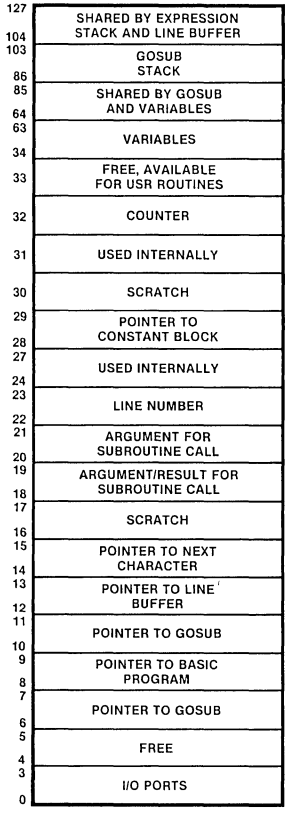


Figure 7a. General-Purpose Registers with External RAM

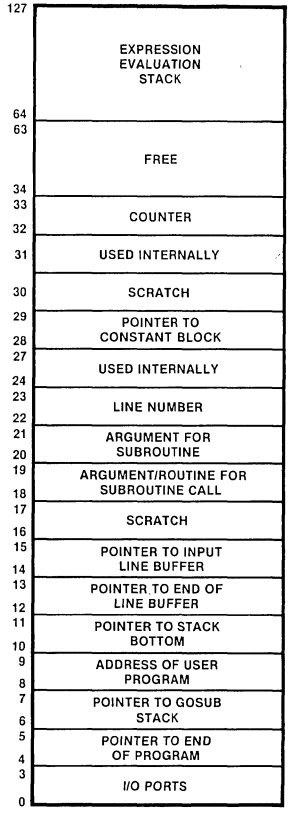


Figure 7b. General-Purpose Registers without External RAM

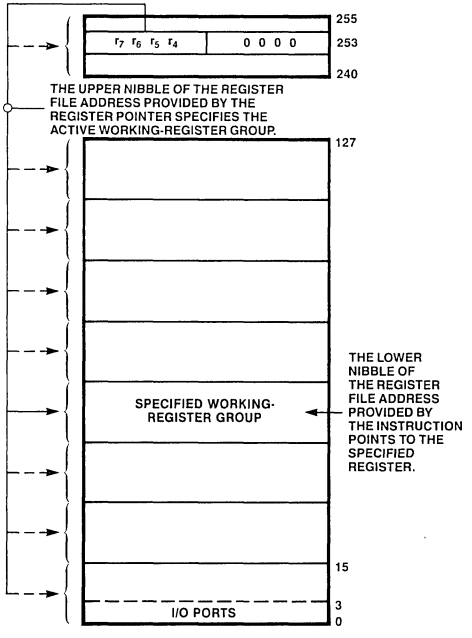


Figure 8. The Register Pointer

PROGRAM EXECUTION

Automatic Start-up. The Z8671 has an automatic start-up capability which allows a program stored in ROM to be executed without operator intervention. Automatic execution occurs on power-on or Reset when the program is stored at address 1020_H.

Execution Modes. The Z8671's BASIC/Debug Interpreter operates in two execution modes: Run and Immediate. Programs are edited and interactively debugged in the Immediate mode. Some BASIC/Debug commands are used almost exclusively in this mode. The Run mode is entered from the Immediate mode by entering the command RUN. If there is a program in RAM, it is executed. The system returns to the Immediate mode when program execution is complete or interrupted by an error.

INTERACTIVE DEBUGGING

Interactive debugging is accomplished with the self-contained line editor which operates in the Immediate mode. In addition to changing program lines, the editor can correct an immediate command before it is executed. It also allows the correction of typing and other errors as a program is entered.

BASIC/Debug allows interruptions and changes during a program run to correct errors and add new instructions without disturbing the sequential execution of the program. A program run is interrupted with the use of the escape key. The run is restarted with a GOTO command, followed by the appropriate line number, after the desired changes are entered. The same procedure is used to enter corrections after BASIC/Debug returns an error.

COMMANDS

BASIC/Debug recognizes 15 command keywords. For detailed instructions of command usage, refer to the *BASIC/Debug Software Reference Manual* (#03-3149-02).

FO — The GO command unconditionally branches to a machine language subroutine. This statement is similar to the USR function except that no value is returned by the assembly language routine.

GOSUB — GOSUB unconditionally branches to a subroutine at a line number specified by the user.

GOTO — GOTO unconditionally changes the sequence of program execution (branches to a line number).

IF/THEN — This command is used for conditional operations and branches.

INPUT/IN — These commands request information from the user with the prompt "?", then read the input values (which must be separated by commas) from the keyboard, and store them in the indicated variables. INPUT discards any values remaining in the buffer from previous IN, INPUT, or RUN statements, and requests new data from the operator. IN uses any values left in the buffer first, then requests new data.

LET — LET assigns the value of an expression to a variable or memory location.

LIST — This command is used in the interactive mode to generate a listing of program lines stored in memory on the terminal device.

NEW — The NEW command resets pointer R10-11 to the beginning of user memory, thereby marking the space as empty and ready to store a new program.

PRINT — PRINT lists its arguments, which may be text messages or numerical values, on the output terminal.

REM — This command is used to insert explanatory messages into the program.

RETURN — This command returns control to the line following a GOSUB statement.

RUN — RUN initiates sequential execution of all instructions in the current program.

STOP — STOP ends program execution and clears the GOSUB stack.

FUNCTIONS

BASIC/Debug supports two functions: AND and USR.

The AND function performs a logical AND. It can be used to mask, turn off, or isolate bits. This function is used in the following format:

AND (expression, expression)

The two expressions are evaluated, and their bit patterns are ANDed together. If only one value is included in the parentheses, it is ANDed with itself. A logical OR can also be performed by complementing the AND function. This is accomplished by subtracting each expression from -1. For example, the function below is equivalent to the OR of A and B.

-1-AND(-1-A, -1-B)

The USR function calls a machine language subroutine and returns a value. This is useful for applications in which a subroutine can be performed more quickly and efficiently in machine language than in BASIC/Debug.

The address of the first instruction of the subroutine is the first argument of the USR function. The address can be followed by one or two values to be processed by the subroutine. In the following example, BASIC/Debug executes the subroutine located at address 2000 using values literal 256 and variable C.

USR(%2000,256,C)

The resulting value is stored in Registers 18-19.

SERIAL INPUT/OUTPUT

Port 3 lines $P3_0$ and $P3_7$ can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by Counter/Timer 0, with a maximum rate of 62.5K bits/second.

The Z8671 automatically adds a start bit and two stop bits to transmitted data (Figure 9). Odd parity is also available as an option. Eight data bits are always transmitted, regardless of parity selection. If parity is enabled, the eighth data bit is used as the odd parity bit. An interrupt request (IRQ4) is generated on all transmitted characters.

Received data must have a start bit, eight data bits, and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ3 interrupt request.

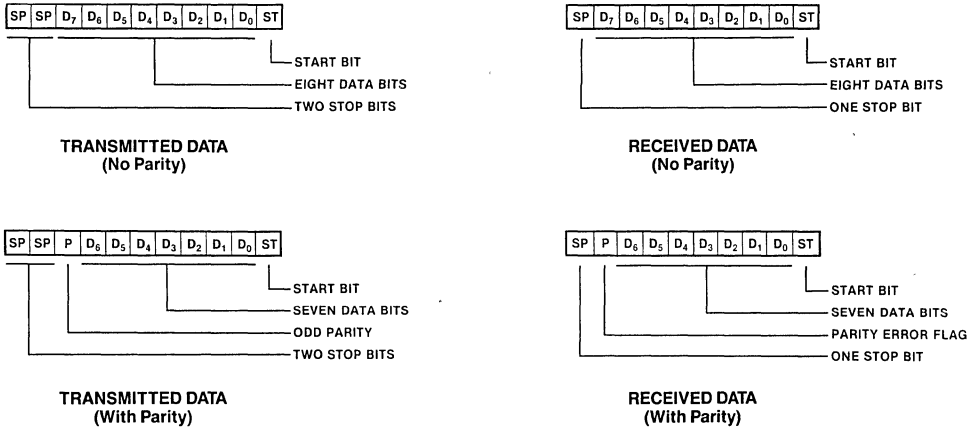


Figure 9. Serial Data Formats

I/O PORTS

The Z8671 has 32 lines dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable as input, output or address/data. Under software control, the ports can be programmed to provide address outputs, timing, status signals, serial I/O, and parallel I/O with or without handshake. All ports have active pull-ups and pull-downs compatible with TTL loads.

Port 1 can be programmed as a byte I/O port or as an address/data port for interfacing external memory. When used as an I/O port, Port 1 may be placed under handshake control. In this configuration, Port 3 lines $P3_3$ and $P3_4$ are used as the handshake controls RDY1 and $\overline{DAV1}$ (Ready and Data Available).

Memory locations greater than 2048 are referenced through Port 1. To interface external memory, Port 1 must be programmed for the multiplexed Address/Data mode. If more than 256 external locations are required, Port 0 must output the additional lines.

Port 1 can be placed in the high-impedance state along with Port 0, $\overline{AS}$, $\overline{DS}$ and R/$\overline{W}$, allowing the Z8671 to share common resources in multiprocessor and DMA applications. Data transfers can be controlled by assigning $P3_3$ as a Bus Acknowledge input and $P3_4$ as a Bus Request output.

Port 0 can be programmed as a nibble I/O port, or as an address port for interfacing external memory. When used as an I/O port, Port 0 may be placed under handshake control. In this configuration, Port 3 lines $P3_2$ and $P3_5$ are used as the handshake controls $\overline{DAV0}$ and RDY0. Handshake signal assignment is dictated by the I/O direction of the upper nibble $P0_4$-$P0_7$.

For external memory references, Port 0 can provide address bits A_8-A_{11} (lower nibble) or A_8-A_{15} (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. When Port 0 nibbles are defined as address bits, they can be set to the high-impedance state along with Port 1 and the control signals $\overline{AS}$, $\overline{DS}$ and R/$\overline{W}$.

Port 2 bits can be programmed independently as input or output. The port is always available for I/O operations. In addition, Port 2 can be configured to provide open-drain outputs.

Like Ports 0 and 1, Port 2 may also be placed under handshake control. In this configuration, Port 3 lines $P3_1$ and $P3_6$ are used as the handshake controls lines $\overline{DAV2}$ and RDY2. The handshake signal assignment for Port 3 lines $P3_1$ and $P3_6$ is dictated by the direction (input or output) assigned to bit 7 of Port 2.

Port 3 lines can be configured as I/O or control lines. In either case, the direction of the eight lines is fixed as four input ($P3_0$-$P3_3$) and four output ($P3_4$-$P3_7$). For serial I/O, lines $P3_0$ and $P3_7$ are programmed as serial in and serial out respectively.

Port 3 can also provide the following control functions: handshake for Ports 0, 1 and 2 ($\overline{DAV}$ and RDY); four external interrupt request signals (IRQ0-IRQ3); timer input and output signals (T_{IN} and T_{OUT}) and Data Memory Select ($\overline{DM}$).

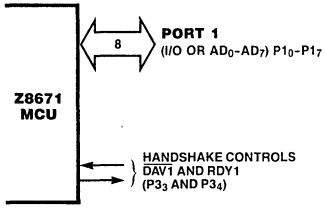


Figure 10a. Port 1

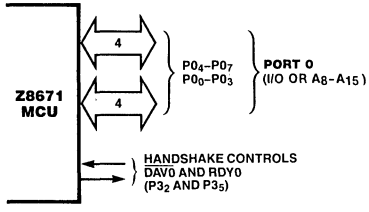


Figure 10b. Port 0

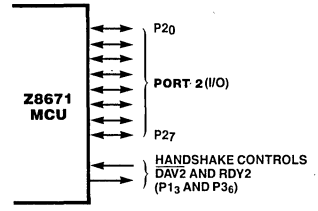


Figure 10c. Port 2

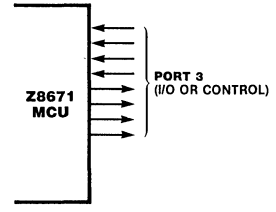


Figure 10d. Port 3

COUNTER/TIMERS

The Z8671 contains two 8-bit programmable counter/timers (T0 and T1), each driven by its own 6-bit programmable prescaler. The T1 prescaler can be driven by internal or external clock sources; however, the T0 prescaler is driven by the internal clock only.

The 6-bit prescalers can divide the input frequency of the clock source by any number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request—IRQ4 (T_0) or IRQ5 (T_1)—is generated.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. The counters can also be programmed to stop upon reaching zero (single-pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode). The counters, but not the prescalers, can be read any time without disturbing their value or count mode.

The clock source for T1 is user-definable; it can be either the internal microprocessor clock (4 MHz maximum) divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input as an external clock, a trigger input that can be retriggerable or nonretriggerable, or as a gate input for the internal clock. The counter/timers can be programmably cascaded by connecting the T0 output to the input of T1. Port 3 line $P3_6$ also serves as a timer output (T_{OUT}) through which T0, T1 or the internal clock can be output.

INTERRUPTS

The Z8671 allows six different interrupts from eight sources: the four Port 3 lines $P3_0$-$P3_3$, Serial In, Serial Out, and the two counter/timers. These interrupts are both maskable and prioritized. The Interrupt Mask register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All Z8671 interrupts are vectored; however, the internal UART operates in a polling fashion. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

The BASIC/Debug Interpreter does not process interrupts. Interrupts are vectored through locations in internal ROM which point to addresses $1000\text{-}1011_H$. To process interrupts, jump instructions can be entered to the interrupt handling routines at the appropriate addresses as shown in Table 1

Table 1. Interrupt Jump Instructions

Hex Address	Contains Jump Instruction and Subroutine Address for:
1000-1002	IRQ0
1003-1005	IRQ1
1006-1008	IRQ2
1009-100B	IRQ3
100C-100E	IRQ4
100F-1011	IRQ5

CLOCK

The on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal or to any suitable external clock source (XTAL1 = Input, XTAL2 = Output).

The crystal source is connected across XTAL1 and XTAL2, using the recommended capacitance (C_L = 15 pf maximum) from each pin to ground. The specifications for the crystal are as follows:

- AT cut, parallel resonant
- Fundamental type, 8 maximum
- Series resistance, R ⩽ 100 Ω
- 8 MHz maximum

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

dst	Destination location or contents
src	Source location or contents
cc	Condition code (see list)
@	Indirect address prefix
SP	Stack pointer (control registers 254-255)
PC	Program counter
FLAGS	Flag register (control register 252)
RP	Register pointer (control register 253)
IMR	Interrupt mask register (control register 251)

Assignment of a value is indicated by the symbol "←". For example,

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

dst (7)

refers to bit 7 of the destination operand.

Flags. Control Register R252 contains the following six flags:

C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

Affected flags are indicated by:

0	Cleared to zero
1	Set to one
*	Set or cleared according to operation
—	Unaffected
X	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000		Always true	—
0111	C	Carry	C = 1
1111	NC	No carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	[Z OR (S XOR V)] = 0
0010	LE	Less than or equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned greater than or equal	C = 0
0111	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1
0000		Never true	—

INSTRUCTION FORMATS

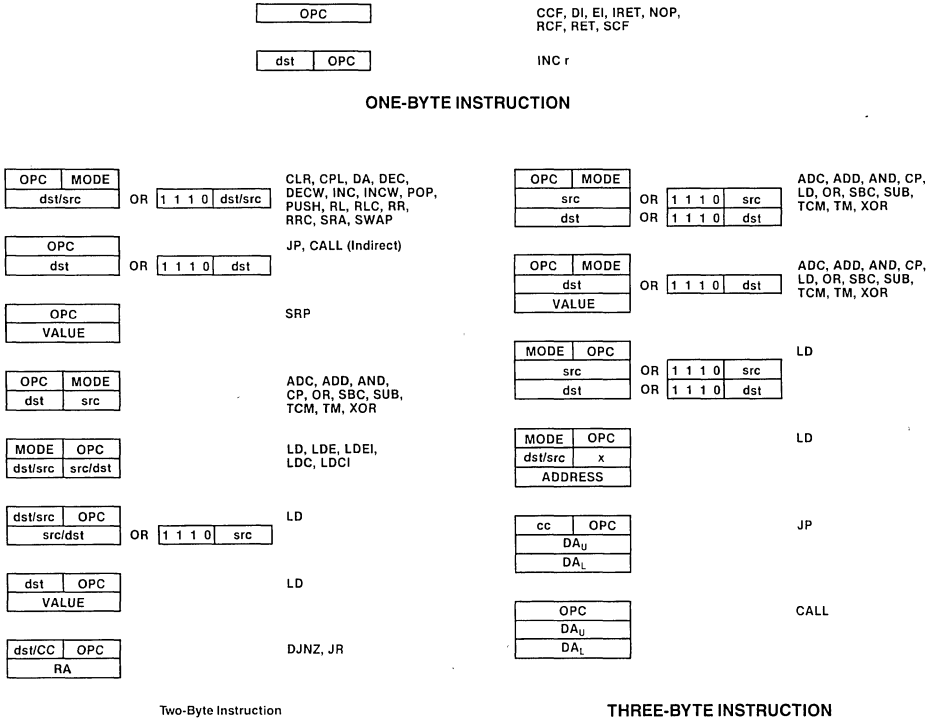
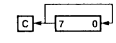
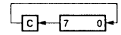
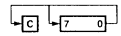


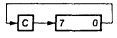
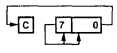
Figure 11. Instruction Formats

INSTRUCTION SUMMARY

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst,src dst ← dst + src + C	(Note 1)		1□	*	*	*	*	0	*
ADD dst,src dst ← dst + src	(Note 1)		0□	*	*	*	*	0	*
AND dst,src dst ← dst AND src	(Note 1)		5□	—	*	*	0	—	—
CALL dst SP ← SP − 2 @SP ← PC; PC ← dst	DA IRR		D6 D4	—	—	—	—	—	—
CCF C ← NOT C			EF	*	—	—	—	—	—
CLR dst dst ← 0	R IR		B0 B1	—	—	—	—	—	—
COM dst dst ← NOT dst	R IR		60 61	—	*	*	0	—	—
CP dst,src dst − src	(Note 1)		A□	*	*	*	*	—	—
DA dst dst ← DA dst	R IR		40 41	*	*	*	X	—	—
DEC dst dst ← dst − 1	R IR		00 01	—	*	*	*	—	—
DECW dst dst ← dst − 1	RR IR		80 81	—	*	*	*	—	—
DI IMR (7) ← 0			8F	—	—	—	—	—	—
DJNZ r,dst r ← r − 1 if r ≠ 0 PC ← PC + dst Range: +127, −128	RA		rA r = 0 − F	—	—	—	—	—	—
EI IMR (7) ← 1			9F	—	—	—	—	—	—
INC dst dst ← dst + 1	r R IR		rE r = 0 − F 20 21	—	*	*	*	—	—
INCW dst dst ← dst + 1	RR IR		A0 A1	—	*	*	*	—	—
IRET FLAGS ← @SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; IMR (7) ← 1			BF	*	*	*	*	*	*
JP cc,dst if cc is true PC ← dst	DA IRR		cD c = 0 − F 30	—	—	—	—	—	—

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
JR cc,dst if cc is true, PC ← PC + dst Range: +127, −128	RA		cB c = 0 − F	—	—	—	—	—	—
LD dst,src dst ← src	r r R r r X r r Ir R R R IR IR	Im R r X r Ir r R IR IM IM IR R	rC r8 r9 r = 0 − F C7 D7 E3 F3 E4 E5 E6 E7 F5	—	—	—	—	—	—
LDC dst,src dst ← src	r Irr	Irr r	C2 D2	—	—	—	—	—	—
LDCI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	C3 D3	—	—	—	—	—	—
LDE dst,src dst ← src	r Irr	Irr r	82 92	—	—	—	—	—	—
LDEI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	83 93	—	—	—	—	—	—
NOP			FF	—	—	—	—	—	—
OR dst,src dst ← dst OR src	(Note 1)		4□	—	*	*	0	—	—
POP dst dst ← @SP; SP ← SP + 1	R IR		50 51	—	—	—	—	—	—
PUSH src SP ← SP − 1; @SP ← src		R IR	70 71	—	—	—	—	—	—
RCF C ← 0			CF	0	—	—	—	—	—
RET PC ← @SP; SP ← SP + 2			AF	—	—	—	—	—	—
RL dst 		R IR	90 91	*	*	*	*	—	—
RLC dst 		R IR	10 11	*	*	*	*	—	—
RR dst 		R IR	E0 E1	*	*	*	*	—	—

INSTRUCTION SUMMARY (Continued)

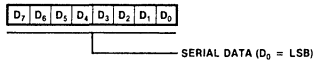
Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
RRC dst 	R		C0	*	*	*	*	—	—
	IR		C1						
SBC dst,src dst ← dst ← src ← C	(Note 1)		3☐	*	*	*	*	1	*
SCF C ← 1			DF	1	—	—	—	—	—
SRA dst 	R		D0	*	*	*	0	—	—
	IR		D1						
SRP src RP ← src		Im	31	—	—	—	—	—	—
SUB dst,src dst ← dst ← src	(Note 1)		2☐	*	*	*	*	1	*
SWAP dst 	R		F0	X	*	*	X	—	—
	IR		F1						
TCM dst,src (NOT dst) AND src	(Note 1)		6☐	—	*	*	0	—	—
TM dst,src dst AND src	(Note 1)		7☐	—	*	*	0	—	—
XOR dst,src dst ← dst XOR src	(Note 1)		B☐	—	*	*	0	—	—

NOTE: These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a ☐ in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

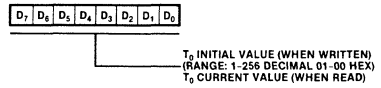
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Addr Mode dst	Addr Mode src	Lower Opcode Nibble
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7

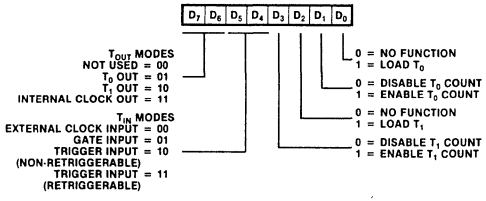
R240 SIO
Serial I/O Register
($F0_H$; Read/Write)



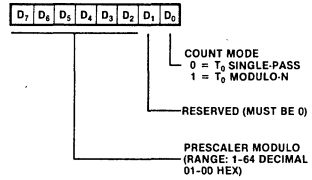
R244 TO
Counter/Timer 0 Register
($F4_H$; Read/Write)



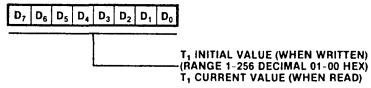
R241 TMR
Time Mode Register
($F1_H$; Read/Write)



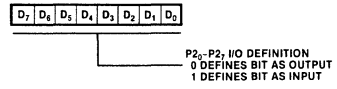
R245 PRE0
Prescaler 0 Register
($F5_H$; Write Only)



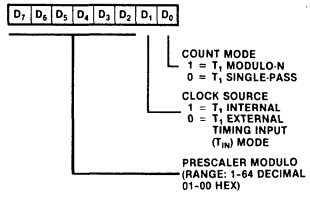
R242 T1
Counter Timer 1 Register
($F2_H$; Read/Write)



R246 P2M
Port 2 Mode Register
($F6_H$; Write Only)



R243 PRE1
Prescaler 1 Register
($F3_H$; Write Only)



R247 P3M
Port 3 Mode Register
($F7_H$; Write Only)

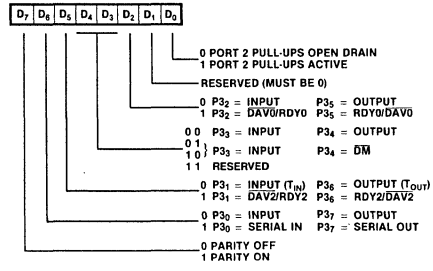
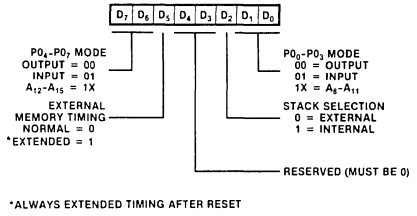


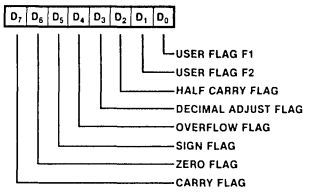
Figure 12. Control Registers

R248 P01M
Port 0 Register
($F8_H$; Write Only)

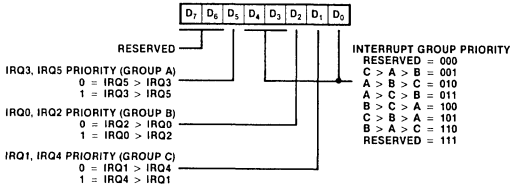


*ALWAYS EXTENDED TIMING AFTER RESET

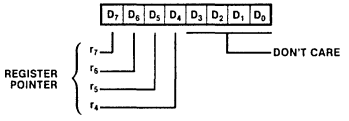
R252 FLAGS
Flag Register
(FC_H; Read/Write)



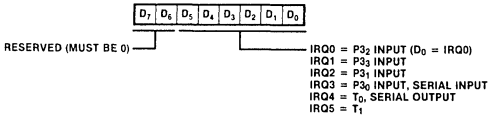
R249 IPR
Interrupt Priority Register
($F9_H$; Write Only)



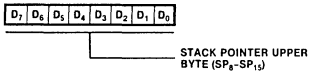
R253 RP
Register Pointer
(FD_H; Read/Write)



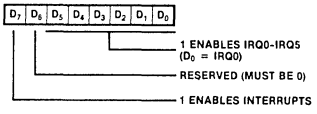
R250 IRQ
Interrupt Request Register
(FA_H; Read/Write)



R254 SPH
Stack Pointer
(FE_H; Read/Write)



R251 IMR
Interrupt Mask Register
(FB_H; Read/Write)



R255 SPL
Stack Pointer
(FF_H; Read/Write)

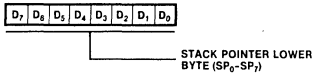
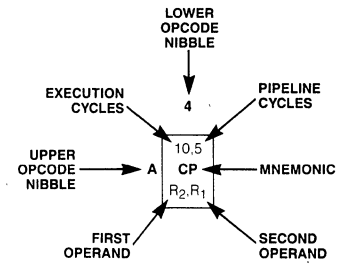


Figure 12. Control Registers (Continued)

OPCODE MAP

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6,5 **DEC** R_1	6,5 **DEC** IR_1	6,5 **ADD** r_1,r_2	6,5 **ADD** r_1,Ir_2	10,5 **ADD** R_2,R_1	10,5 **ADD** IR_2,R_1	10,5 **ADD** R_1,IM	10,5 **ADD** IR_1,IM	6,5 **LD** r_1,R_2	6,5 **LD** r_2,R_1	12/10,5 **DJNZ** r_1,RA	12/10,0 **JR** cc,RA	6,5 **LD** r_1,IM	12/10,0 **JP** cc,DA	6,5 **INC** r1	
1	6,5 **RLC** R_1	6,5 **RLC** IR_1	6,5 **ADC** r_1,r_2	6,5 **ADC** r_1,Ir_2	10,5 **ADC** R_2,R_1	10,5 **ADC** IR_2,R_1	10,5 **ADC** R_1,IM	10,5 **ADC** IR_1,IM								
2	6,5 **INC** R_1	6,5 **INC** IR_1	6,5 **SUB** r_1,r_2	6,5 **SUB** r_1,Ir_2	10,5 **SUB** R_2,R_1	10,5 **SUB** IR_2,R_1	10,5 **SUB** R_1,IM	10,5 **SUB** IR_1,IM								
3	8,0 **JP** IRR_1	6,1 **SRP** IM	6,5 **SBC** r_1,r_2	6,5 **SBC** r_1,Ir_2	10,5 **SBC** R_2,R_1	10,5 **SBC** IR_2,R_1	10,5 **SBC** R_1,IM	10,5 **SBC** IR_1,IM								
4	8,5 **DA** R_1	8,5 **DA** IR_1	6,5 **OR** r_1,r_2	6,5 **OR** r_1,Ir_2	10,5 **OR** R_2,R_1	10,5 **OR** IR_2,R_1	10,5 **OR** R_1,IM	10,5 **OR** IR_1,IM								
5	10,5 **POP** R_1	10,5 **POP** IR_1	6,5 **AND** r_1,r_2	6,5 **AND** r_1,Ir_2	10,5 **AND** R_2,R_1	10,5 **AND** IR_2,R_1	10,5 **AND** R_1,IM	10,5 **AND** IR_1,IM								
6	6,5 **COM** R_1	6,5 **COM** IR_1	6,5 **TCM** r_1,r_2	6,5 **TCM** r_1,Ir_2	10,5 **TCM** R_2,R_1	10,5 **TCM** IR_2,R_1	10,5 **TCM** R_1,IM	10,5 **TCM** IR_1,IM								
7	10/12,1 **PUSH** R_2	12/14,1 **PUSH** IR_2	6,5 **TM** r_1,r_2	6,5 **TM** r_1,Ir_2	10,5 **TM** R_2,R_1	10,5 **TM** IR_2,R_1	10,5 **TM** R_1,IM	10,5 **TM** IR_1,IM								
8	10,5 **DECW** RR_1	10,5 **DECW** IR_1	12,0 **LDE** r_1,Irr_2	18,0 **LDEI** Ir_1,Ir_2												6,1 **DI**
9	6,5 **RL** R_1	6,5 **RL** IR_1	12,0 **LDE** r_2,Irr_1	18,0 **LDEI** Ir_2,Irr_1												6,1 **EI**
A	10,5 **INCW** RR_1	10,5 **INCW** IR_1	6,5 **CP** r_1,r_2	6,5 **CP** r_1,Ir_2	10,5 **CP** R_2,R_1	10,5 **CP** IR_2,R_1	10,5 **CP** R_1,IM	10,5 **CP** IR_1,IM								14,0 **RET**
B	6,5 **CLR** R_1	6,5 **CLR** IR_1	6,5 **XOR** r_1,r_2	6,5 **XOR** r_1,Ir_2	10,5 **XOR** R_2,R_1	10,5 **XOR** IR_2,R_1	10,5 **XOR** R_1,IM	10,5 **XOR** IR_1,IM								16,0 **IRET**
C	6,5 **RRC** R_1	6,5 **RRC** IR_1	12,0 **LDC** r_1,Irr_2	18,0 **LDCI** Ir_1,Ir_2				10,5 **LD** r_1,x,R_2								6,5 **RCF**
D	6,5 **SRA** R_1	6,5 **SRA** IR_1	12,0 **LDC** r_2,Irr_1	18,0 **LDCI** Ir_2,Irr_1	20,0 **CALL*** IRR_1		20,0 **CALL** DA	10,5 **LD** r_2,x,R_1								6,5 **SCF**
E	6,5 **RR** R_1	6,5 **RR** IR_1		6,5 **LD** r_1,IR_2	10,5 **LD** R_2,R_1	10,5 **LD** IR_2,R_1	10,5 **LD** R_1,IM	10,5 **LD** IR_1,IM								6,5 **CCF**
F	8,5 **SWAP** R_1	8,5 **SWAP** IR_1		6,5 **LD** Ir_1,r_2		10,5 **LD** R_2,IR_1										6,0 **NOP**
Bytes per Instruction	2				3				2					3	1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_1 = Dst address
R_2 or r_2 = Src address

Sequence:
Opcode, First Operand, Second Operand

NOTE: The blank areas are not defined.

*2-byte instruction, fetch cycle appears as a 3-byte instruction

ABSOLUTE MAXIMUM RATINGS

Voltages on all pins with respect to GND . . . . . −0.3V to +7.0V
Operating Ambient Temperature . . . . . See Ordering Information
Storage Temperature . . . . . −65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

STANDARD TEST CONDITIONS

The DC characteristics listed below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin.

Standard conditions are:

- $+4.75V \leqslant V_{CC} \leqslant +5.25V$
- GND = 0V
- $0°C \leqslant T_A \leqslant +70°C$

The Ordering Information section lists package temperature ranges and product numbers. Package drawings are in the Package Information section. Refer to the Literature List for additional documentation.

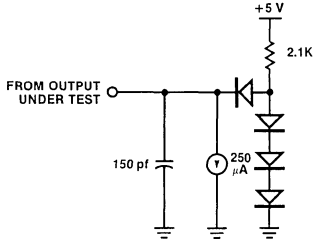


Figure 13. Test Load 1

DC CHARACTERISTICS

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	3.8	V_{CC}	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	−0.3	0.8	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{IL}	Input Low Voltage	−0.3	0.8	V	
V_{RH}	Reset Input High Voltage	3.8	V_{CC}	V	
V_{RL}	Reset Input Low Voltage	−0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −250 μA
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA
I_{IL}	Input Leakage	−10	10	μA	$0V \leqslant V_{IN} \leqslant +5.25V$
I_{OL}	Output Leakage	−10	10	μA	$0V \leqslant V_{IN} \leqslant +5.25V$
I_{IR}	Reset Input Current		−50	μA	V_{CC} = +5.25V, V_{RL} = 0V
I_{CC}	V_{CC} Supply Current		180	mA	

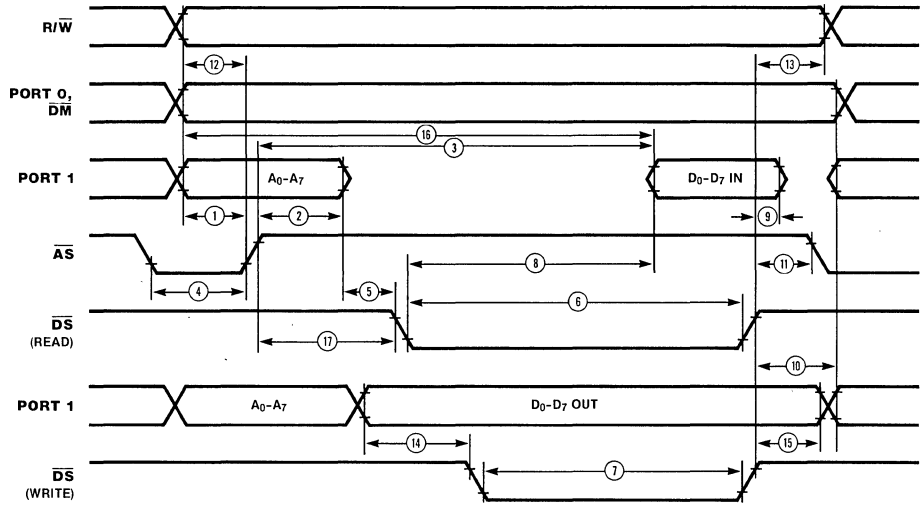


Figure 16. External I/O or Memory Read/Write

AC CHARACTERISTICS

External I/O or Memory Read/Write Timing

No.	Symbol	Parameter	Min	Max	Notes*†°
1	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	35		2,3
2	TdAS(A)	$\overline{AS}$ ↑ to Address Float Delay	45		2,3
3	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Required Valid		220	1,2,3
4	TwAS	$\overline{AS}$ Low Width	55		1,2,3
5	TdAz(DS)	Address Float to $\overline{DS}$ ↓	0		
6	TwDSR	$\overline{DS}$ (Read) Low Width	185		1,2,3
7	TwDSW	$\overline{DS}$ (Write) Low Width	110		1,2,3
8	TdDSR(DR)	$\overline{DS}$ ↓ to Read Data Required Valid		130	1,2,3
9	ThDR(DS)	Read Data to $\overline{DS}$ ↑ Hold Time	0		
10	TdDS(A)	$\overline{DS}$ ↑ to Address Active Delay	45		2,3
11	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	55		2.3
12	TdR/W(AS)	R/$\overline{W}$ Valid to $\overline{AS}$ ↑ Delay	30		2,3
13	TdDS(R/W)	$\overline{DS}$ ↑ to R/$\overline{W}$ Not Valid	35		2,3
14	TdDW(DSW)	Write Data Valid to $\overline{DS}$ (Write) ↓ Delay	35		2,3
15	TdDS(DW)	$\overline{DS}$ ↑ to Write Data Not Valid Delay	45		2,3
16	TdA(DR)	Address Valid to Read Data Required Valid		255	1,2,3
17	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	55		2,3

NOTES:
1. When using extended memory timing add 2 TpC.
2. Timing numbers given are for minimum TpC.
3. See clock cycle time dependent characteristics table.

† Test Load 1.
° All timing references use 2.0 V for a logic "1" and 0.8 V for a logic "0".
* All units in nanoseconds (ns).

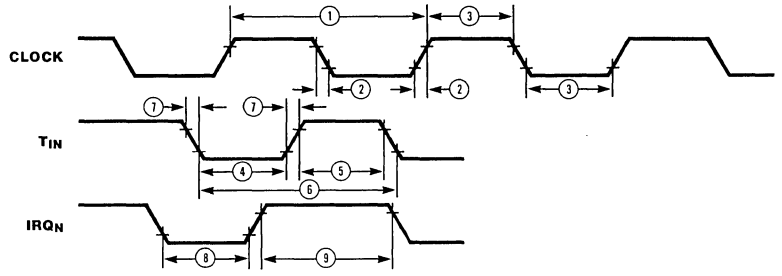


Figure 17. Additional Timing

AC CHARACTERISTICS

Additional Timing

No.	Symbol	Parameter	Min	Max	Notes*
1	TpC	Input Clock Period	80	1000	1
2	TrC,TfC	Clock Input Rise And Fall Times		15	1
3	TwC	Input Clock Width	26		1
4	TwTinL	Time Input Low Width	70		2
5	TwTinH	Timer Input High Width	3TpC		2
6	TpTin	Timer Input Period	8TpC		2
7	TrTin,TfTin	Timer Input Rise And Fall Times		100	2
8a	TwIL	Interrupt Request Input Low Time	70		2,3
8b	TwIL	Interrupt Request Input Low Time	3TpC		2,4
9	TwIH	Interrupt Request Input High Time	3TpC		2,3

NOTES:
1. Clock timing references uses 3.8 V for a logic "1" and 0.8 V for a logic "0".
2. Timing reference uses 2.0 V for a logic "1" and 0.8 V for a logic "0".
3. Interrupt request via Port 3 ($P3_1$-$P3_3$).
4. Interrupt request via Port 3 ($P3_0$).
* Units in nanoseconds (ns).

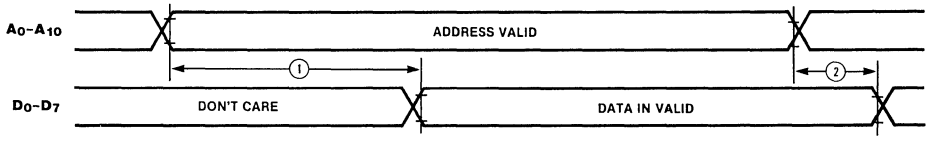


Figure 18. Memory Port Timing

AC CHARACTERISTICS

Memory Port Timing

No.	Symbol	Parameter	Min	Max	Notes*
1	TdA(DI)	Address Valid to Data Input Delay		320	1,2
2	ThDI(A)	Data In Hold time	0		1

NOTES:
1. Test Load 2.
2. This is a Clock-Cycle-Dependent parameter. For clock frequencies other than the maximum, use the following formula: 5 TpC – 95

*Units are nanoseconds unless otherwise specified.

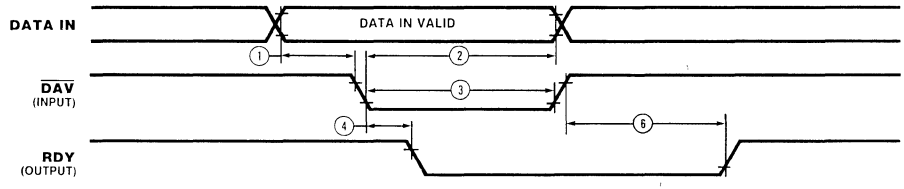


Figure 18a. Input Handshake

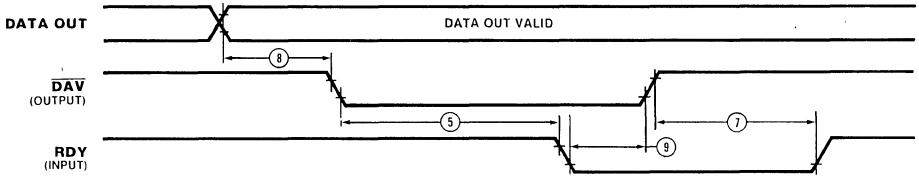


Figure 18b. Output Handshake

AC CHARACTERISTICS

Handshake Timing

No.	Symbol	Parameter	Min	Max	Notes*
1	TsDI(DAV)	Data In Setup Time	0		
2	ThDI(DAV)	Data In Hold time	160		
3	TwDAV	Data Available Width	120		
4	TdDAVIf(RDY)	$\overline{DAV}$ ↓ Input to RDY ↓ Delay		120	1,2
5	TdDAVOf(RDY)	$\overline{DAV}$ ↓ Output to RDY ↓ Delay	0		1,3
6	TdDAVIr(RDY)	$\overline{DAV}$ ↑ Input to RDY ↑ Delay		120	1,2
7	TdDAV0r(RDY)	$\overline{DAV}$ ↑ Output to RDY ↑ Delay	0		1,3
8	TdDO(DAV)	Data Out to $\overline{DAV}$ ↓ Delay	30		1
9	TdRDY(DAV)	Rdy ↓ Input to $\overline{DAV}$ ↑ Delay	0	140	1

NOTES:
1. Test load 1
2. Input handshake
3. Output handshake
† All timing references use 2.0 V for a logic "1" and 0.8 V for a logic "0".

* Units in nanoseconds (ns).

CLOCK CYCLE TIME-DEPENDENT CHARACTERISTICS

Number	Symbol	Z8671-8 Equation	Number	Symbol	Z8671-8 Equation
1	TdA(AS)	TpC – 75	13	TdDS(R/W)	TpC – 65
2	TdAS(A)	TpC – 55	14	TdDW(DSW)	TpC – 75
3	TdAS(DR)	4TpC – 140*	15	TdDS(DW)	TpC – 55
4	TwAS	TpC – 45	16	TdA(DR)	5TpC – 215*
6	TwDSR	3TpC – 125*	17	TdAS(DS)	TpC – 45
7	TwDSW	2TpC – 90*			
8	TdDSR(DR)	3TpC – 175*			
10	Td(DS)A	TpC – 55			
11	TdDS(AS)	TpC – 55			
12	TdR/W(AS)	TpC – 75			

* Add 2TpC when using extended memory timing

Z8681/82 Z8® ROMless MCU

FEATURES

- Complete microcomputer, 24 I/O lines, and up to 64K bytes of addressable external space each for program and data memory.
- 143-byte register file, including 124 general-purpose registers, 3 I/O port registers, and 16 status and control registers.
- Vectored, priority interrupts for I/O, counter/timers, and UART.
- On-chip oscillator that accepts crystal or external clock drive.
- Full-duplex UART and two programmable 8-bit counter/timers, each with a 6-bit programmable prescaler.
- Register Pointer so that short, fast instructions can access any one of the nine working-register groups.
- Single +5V power supply—all I/O pins TTL compatible.
- **Z8681/82 available in 8 MHz. Z8681 also available in 12 MHz.**

GENERAL DESCRIPTION

The Z8681 and Z8682 are ROMless versions of the Z8 single-chip microcomputer. The Z8682 is usually more cost effective. These products differ only slightly and can be used interchangeably with proper system design to provide maximum flexibility in meeting price and delivery needs.

The Z8681/82 offers all the outstanding features of the Z8 family architecture except an on-chip program ROM. Use of external memory rather than a preprogrammed ROM enables this Z8 microcomputer to be used in low volume applications or where code flexibility is required.

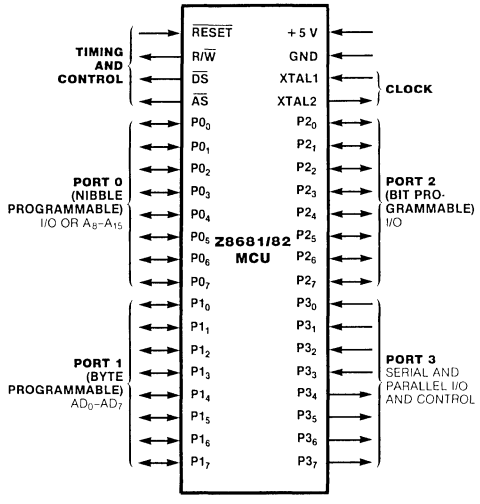


Figure 1. Pin Functions

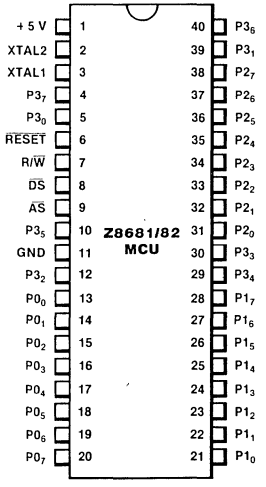


Figure 2a. 40-pin Dual-In-Line Package (DIP), Pin Assignments

The Z8681/82 can provide up to 16 output address lines, thus permitting an address space of up to 64K bytes of data or program memory. Eight address outputs (AD_0-AD_7) are provided by a multiplexed, 8-bit, Address/Data bus. The remaining 8 bits can be provided by the software configuration of Port 0 to output address bits A_8-A_{15}.

Available address space can be doubled (up to 128K bytes for the Z8681 and 124K bytes for the Z8682) by programming bit 4 of Port 3 ($P3_4$) to act as a data memory select output ($\overline{DM}$). The two states of $\overline{DM}$ together with the 16 address outputs can define separate data and memory address spaces of up to 64K/62Kbytes each.

There are 143 bytes of RAM located on-chip and organized as a register file of 124 general-purpose registers, 16 control and status registers, and three I/O port registers. This register file can be divided into nine groups of 16 working registers each. Configuring the register file in this manner allows the use of short format instructions; in addition, any of the individual registers can be accessed directly.

The pin functions and the pin assignments of the Z8681/82 40– and 44–pin packages are illustrated in Figures 1 and 2, respectively.

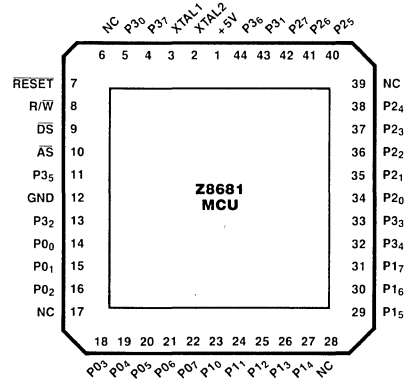


Figure 2b. 44-pin Chip Carrier, Pin Assignments

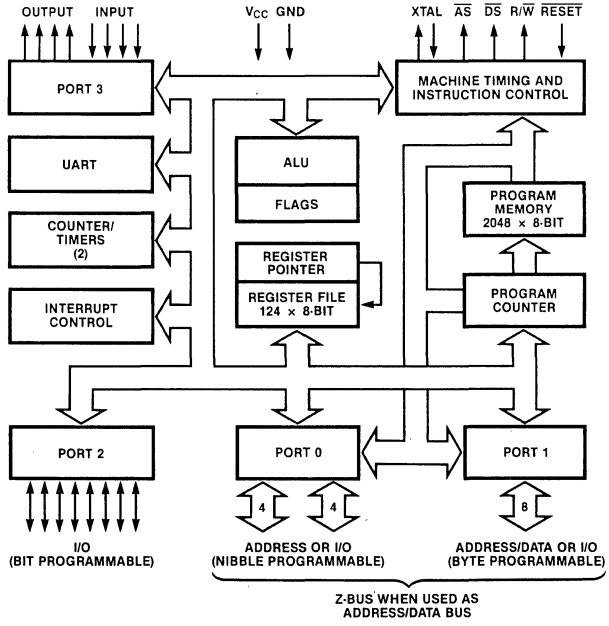


Figure 3. Functional Block Diagram

ARCHITECTURE

Z8681/82 architecture is characterized by a flexible I/O scheme, an efficient register and address space structure and a number of ancillary features that are helpful in many applications.

Microcomputer applications demand powerful I/O capabilities. The Z8681/82 fulfills this with 24 pins available for input and output. These lines are grouped into three ports of eight lines each and are configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an Address bus for interfacing external memory.

Three basic address spaces are available: program memory, data memory and the register file (internal). The 143-byte random-access register file is composed of 124 general-purpose registers, three I/O port registers, and 16 control and status registers.

To unburden the program from coping with real-time problems such as serial data communication and counting/timing, an asynchronous receiver/transmitter (UART) and two counter/timers with a large number of user-selectable modes are offered on-chip. Hardware support for the UART is minimized because one of the on-chip timers supplies the bit rate. Figure 3 shows the Z8681/82 block diagram.

PIN DESCRIPTION

$\overline{AS}$. *Address Strobe* (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Addresses output via Port 1 for all external program or data memory transfers are valid at the trailing edge of $\overline{AS}$.

$\overline{DS}$. *Data Strobe* (output, active Low). Data Strobe is activated once for each external memory transfer.

$P0_0$-$P0_7$, $P2_0$-$P2_7$, $P3_0$-$P3_7$. *I/O Port Lines* (input/outputs, TTL-compatible). These 24 lines are divided into three 8-bit I/O ports that can be configured under program control for I/O or external memory interface (Figure 3)

$P1_0$-$P1_7$. *Address/Data Port* (bidirectional). Multiplexed address (A_0-A_7) and data (D_0-D_7) lines used to interface with program and data memory.

$\overline{RESET}$. *Reset* (input, active Low). $\overline{RESET}$ initializes the Z8681/82. After $\overline{RESET}$ the Z8681 is in the extended memory mode. When $\overline{RESET}$ is deactivated, program execution begins from program location $000C_H$ for the Z8681 and 0812_H for the Z8682.

R/$\overline{W}$. *Read/Write* (output). R/$\overline{W}$ is Low when the Z8681/82 is writing to external program or data memory.

XTAL1, XTAL2. *Crystal 1, Crystal 2* (time-base input and output). These pins connect a parallel-resonant crystal to the on-chip clock oscillator and buffer.

SUMMARY OF Z8681 AND Z8682 DIFFERENCES

Feature	Z8681	Z8682
Address of first instruction executed after Reset	12	2066
Addressable memory space	0–64K	2K–64K
Address of interrupt vectors	0–11	2048–2065
Reset input high voltage	TTL levels *	7.35–8.0V
Port 0 configuration after Reset	Input, float after reset. Can be programmed as Address bits.	Output, configured as Address bit A_8–A_{15}.
External memory timing start-up configurations	Extended Timing	Normal Timing
Interrupt vectors	2 byte vectors point directly to service routines.	2 byte vectors in internal ROM point to 3 byte Jump instructions, which point to service routines.
Interrupt response time	26 clocks	36 clocks

* 8.0V V_{IN} max.

ADDRESS SPACES

Program Memory*. The Z8681/82 addresses 64K/62K bytes of external program memory space (Figure 4).

For the Z8681, the first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts. Program execution begins at location $000C_H$ after a reset.

The Z8682 has six 24-bit interrupt vectors beginning at address 0800_H. The vectors consist of Jump Absolute instructions. After a reset, program execution begins at location 0812_H for the Z8682.

Data Memory*. The Z8681/82 can address 64K/62K bytes of external data memory. External data memory may be included with or separated from the external program memory space. $\overline{DM}$, an optional I/O function that can be programmed to appear on pin $P3_4$, is used to distinguish between data and program memory space.

Register File. The 143-byte register file includes three I/O port registers (R0, R2, R3), 124 general-purpose registers (R4-R127) and 16 control and status registers (R240-R255). These registers are assigned the address locations shown in Figure 5.

Z8681/82 instructions can access registers directly or indirectly with an 8-bit address field. This also allows short 4-bit register addressing using the Register Pointer (one of the control registers). In the 4-bit mode, the register file is divided into nine working-register groups, each occupying 16 contiguous locations (Figure 5). The Register Pointer addresses the starting location of the active working-register group (Figure 6).

Stacks. Either the internal register file or the external data memory can be used for the stack. A 16-bit Stack Pointer (R254 and R255) is used for the external stack, which can reside anywhere in data memory. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 124 general-purpose registers (R4-R127).

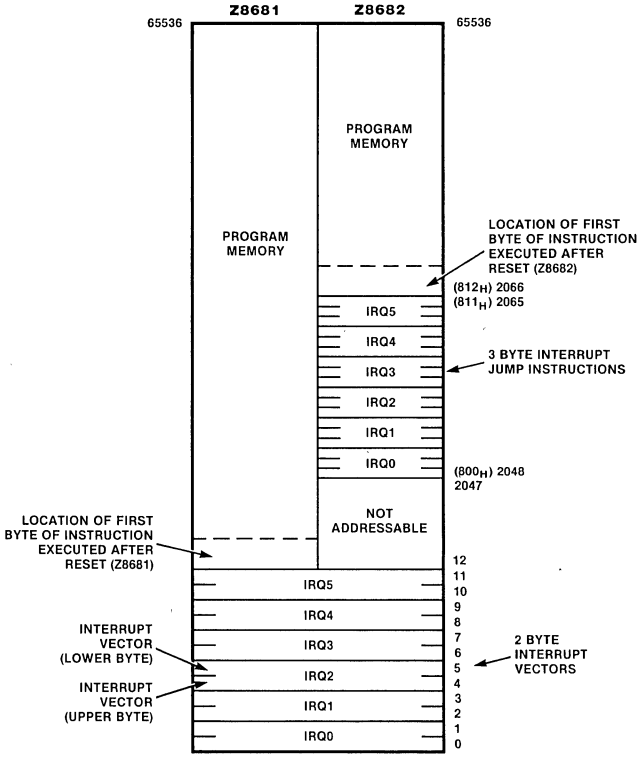


Figure 4. Z8681/82 Program Memory Map

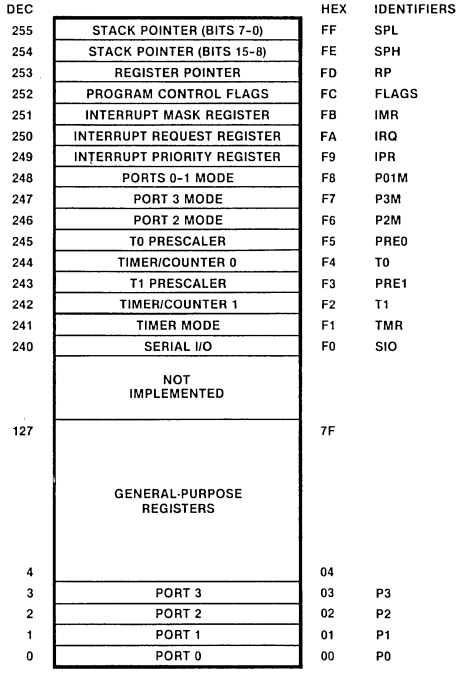


Figure 5. The Register File

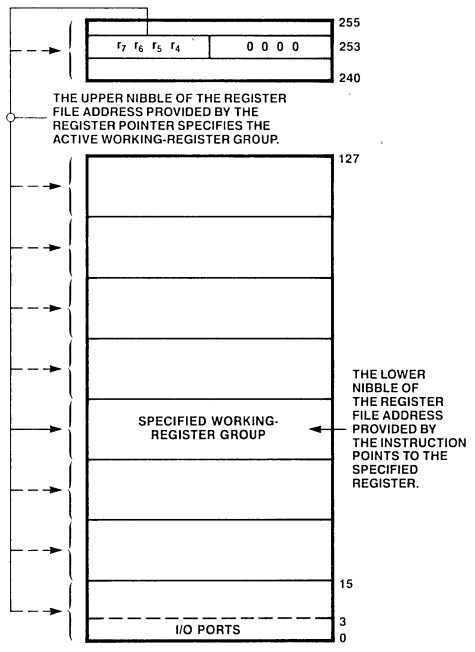


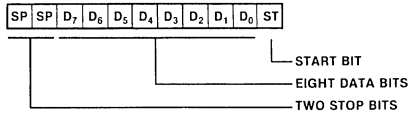
Figure 6. The Register Pointer

SERIAL INPUT/OUTPUT

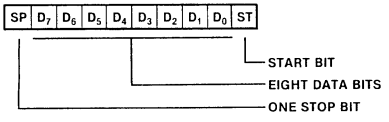
Port 3 lines $P3_0$ and $P3_7$ can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by Counter/Timer 0.

The Z8681/82 automatically adds a start bit and two stop bits to transmitted data (Figure 7). Odd parity is also available as an option. Eight data bits are always transmitted, regardless of parity selection. If parity is enabled, the eighth data bit is used as the odd parity bit. An interrupt request (IRQ4) is generated on all transmitted characters.

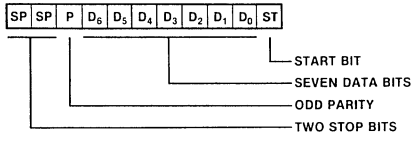
Received data must have a start bit, eight data bits, and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ3 interrupt request.



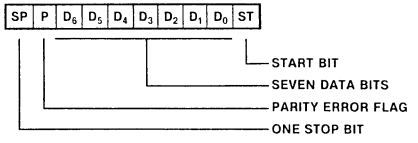
Transmitted Data (No Parity)



Received Data (No Parity)



Transmitted Data (With Parity)



Received Data (With Parity)

Figure 7. Serial Data Formats

COUNTER/TIMERS

The Z8681/82 contains two 8-bit programmable counter/timers (T_0 and T_1), each driven by its own 6-bit programmable prescaler. The T_1 prescaler can be driven by internal or external clock sources; however, the T_0 prescaler is driven by the internal clock only.

The 6-bit prescalers can divide the input frequency of the clock source by any number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request—IRQ4 (T_0) or IRQ5 (T_1)—is generated.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. The counters can also be programmed to stop upon reaching zero (single-pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode). The counters, but not the prescalers, can be read any time without disturbing their value or count mode.

The clock source for T_1 is user-definable; it can be either the internal microprocessor clock divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input as an external clock, a trigger input that can be retriggerable or nonretriggerable, or as a gate input for the internal clock. The counter/timers can be programmably cascaded by connecting the T_0 output to the input of T_1. Port 3 line $P3_6$ also serves as a timer output (T_{OUT}) through which T_0, T_1 or the internal clock can be output.

I/O PORTS

The Z8681/82 has 24 lines available for input and output. These lines are grouped into three ports of eight lines each and are configurable as input, output or address. Under software control, the ports can be programmed to provide address outputs, timing, status signals, serial I/O, and parallel I/O with or without handshake. All ports have active pull-ups and pull-downs compatible with TTL loads.

Port 1 is a dedicated Z-BUS compatible memory interface. The operations of Port 1 are supported by the Address Strobe ($\overline{AS}$) and Data Strobe ($\overline{DS}$) lines, and by the Read/Write (R/$\overline{W}$) and Data Memory ($\overline{DM}$) control lines. The low-order program and data memory addresses (A_0-A_7) are output through Port 1 (Figure 8) and are multiplexed with data in/out (D_0-D_7). Instruction fetch and data memory read/write operations are done through this port.

Port 1 cannot be used as a register nor can a handshake mode be used with this port.

Both the Z8681 and Z8682 wake up with the 8 bits of Port 1 configured as address outputs for external memory. If more than eight address lines are required with the Z8681, additional lines can be obtained by programming Port 0 bits as address bits. The least-significant four bits of Port 0 can be configured to supply address bits A_8-A_{11} for 4K byte addressing or both nibbles of Port 0 can be configured to supply address bits A_8-A_{15} for 64K byte addressing.

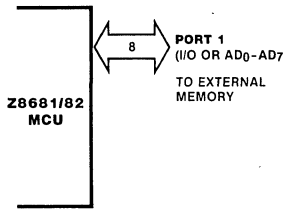


Figure 8. Port 1

Port 0* can be programmed as a nibble I/O port, or as an address port for interfacing external memory (Figure 9). When used as an I/O port, Port 0 can be placed under handshake control. In this configuration, Port 3 lines $P3_2$ and $P3_5$ are used as the handshake controls DAV_0 and RDY_0. Handshake signal assignment is dictated by the I/O direction of the upper nibble $P0_4$-$P0_7$.

For external memory references, Port 0 can provide address bits A_8-A_{11} (lower nibble) or A_8-A_{15} (lower and upper nibbles) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing.

*In the Z8681 **, Port 0 lines float after reset; their logic state is unknown until the execution of an initialization routine that configures Port 0.

Such an initialization routine must reside within the first 256 bytes of executable code and must be physically mapped into memory by forcing the Port 0 address lines to a known state (Figure 10). The proper port initialization sequence is:

1. Write initial address (A_8-A_{15}) of initialization routine to Port 0 address lines.
2. Configure Port 0 Mode register to output A_8-A_{15} (or A_8-A_{11}).

To permit the use of slow memory, an automatic wait mode of two oscillator clock cycles is configured for the bus timing of the Z8681 after each reset. The initialization routine could include reconfiguration to eliminate this extended timing mode.

*This feature differs in the Z8681 and Z8682.

The following example illustrates the manner in which an initialization routine can be mapped in a Z8681 system with 4K of memory.

Example. In Figure 10, the initialization routine is mapped to the first 256 bytes of program memory. Pull-down resistors maintain the address lines at a logic 0 level when these lines are floating. The leakage current caused by fanout must be taken into consideration when selecting the value of the pulldown resistors. The resistor value must be large enough to allow the Port 0 output driver to pull the line to a logic 1. Generally, pulldown resistors are incompatible with TTL loads. If Port 0 drives into TTL input loads (I_{LOW} = 1.6 mA) the external resistors should be tied to V_{CC} and the initialization routine put in address space $FF00_H$–$FFFF_H$.

*In the Z8682**, Port 0 lines are configured as address lines A_8-A_{15} after a Reset. If one or both nibbles are needed for I/O operation, they must be configured by writing to the Port 0 Mode register. The Z8682 is in the fast memory timing mode after Reset, so the initialization routine must be in fast memory.

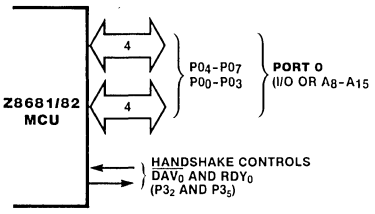


Figure 9. Port 0

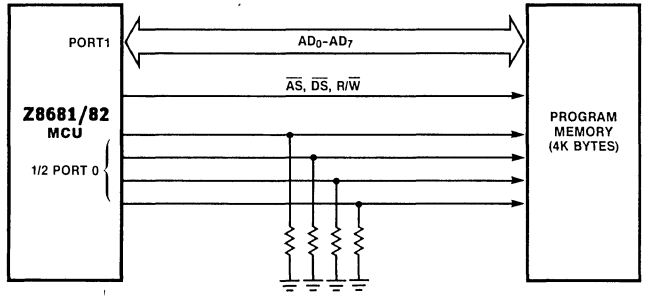


Figure 10. Port 0 Address Lines Tied to Logic 0

Port 2 bits can be programmed independently as input or output (Figure 11). This port is always available for I/O operations. In addition, Port 2 can be configured to provide open-drain outputs.

Like Port 0, Port 2 may also be placed under handshake control. In this configuration, Port 3 lines $P3_1$ and $P3_6$ are used as the handshake controls lines $\overline{DAV}_2$ and RDY_2. The handshake signal assignment for Port 3 lines $P3_1$ and $P3_6$ is dictated by the direction (input or output) assigned to bit 7 of Port 2.

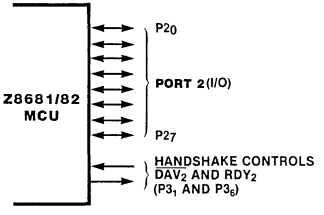


Figure 11. Port 2

Port 3 lines can be configured as I/O or control lines (Figure 12). In either case, the direction of the eight lines is fixed as four input ($P3_0$-$P3_3$) and four output ($P3_4$-$P3_7$). For serial I/O, lines $P3_0$ and $P3_7$ are programmed as serial in and serial out, respectively.

Port 3 can also provide the following control functions: handshake for Ports 0 and 2 ($\overline{DAV}$ and RDY); four external interrupt request signals (IRQ0-IRQ3); timer input and output signals (T_{IN} and T_{OUT}) and Data Memory Select ($\overline{DM}$).

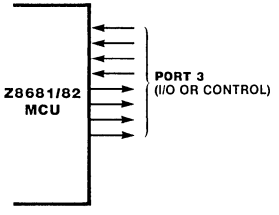


Figure 12. Port 3

*This feature differs in the Z8681 and Z8682.

INTERRUPTS*

The Z8681/82 allows six different interrupts from eight sources: the four Port 3 lines $P3_0$-$P3_3$, Serial In, Serial Out, and the two counter/timers. These interrupts are both maskable and prioritized. The Interrupt Mask register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All Z8681 and Z8682 interrupts are vectored through locations in program memory. When an interrupt request is granted, an interrupt machine cycle is entered. This disables all subsequent interrupts, saves the Program Counter and status flags, and accesses the program memory vector location reserved for that interrupt. In the Z8681, this memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request. The Z8681 takes 63 crystal cycles to enter an interrupt subroutine.

The Z8682 has a small internal ROM that contains six 2-byte interrupt vectors pointing to addresses 2048-2065, where 3-byte jump absolute instructions are located (Figure 4 and Table 1). These jump instructions each contain a 1-byte opcode and a 2-byte starting address for the interrupt service routine.

Table 1. Z8682 Interrupt Processing

Hex Address	Contains Jump Instruction and Subroutine Address For
800-802	IRQ0
803-805	IRQ1
806-808	IRQ2
809-80B	IRQ3
80C-80E	IRQ4
80F-811	IRQ5

Polled interrupt systems are also supported. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

CLOCK

The on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal or to any suitable external clock source (XTAL1 = Input, XTAL2 = Output).

The crystal source is connected across XTAL1 and XTAL2, using the recommended capacitance (C_L = 15 pf maximum) from each pin to ground. The specifications for the crystal are as follows:

- AT cut, parallel-resonant
- Fundamental type
- Series resistance, $R_S \leq 100\Omega$
- For Z8682, 8 MHz maximum
- **For Z8681–12, 16 MHz maximum**

Z8681/Z8682 INTERCHANGEABILITY

Although the Z8681 and Z8682 have minor differences, a system can be designed for compatibility with both ROMless versions. To achieve interchangeability, the design must take into account the special requirements of each device in the external interface, initialization, and memory mapping.

External Interface. The Z8682 requires a 7.5V positive logic level on the $\overline{\text{RESET}}$ pin for at least 6 clock periods immediately following reset, as shown in Figure 13. The Z8681 requires a 3.8V or higher positive logic level, but is compatible with the Z8682 $\overline{\text{RESET}}$ waveform. Figure 14 shows a simple circuit for generating the 7.5V level.

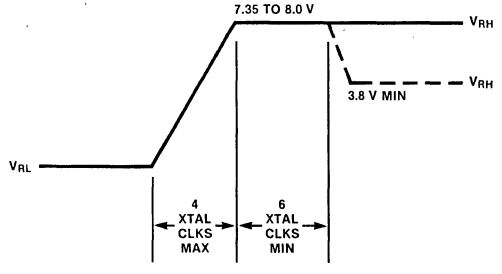


Figure 13. Z8682 RESET Pin Input Waveform

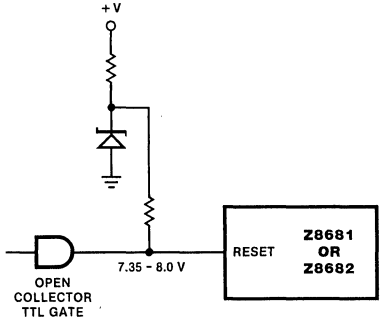


Figure 14. RESET Circuit

*This feature differs in the Z8681 and Z8682.

Initialization. The Z8681 wakes up after reset with Port 0 configured as an input, which means Port 0 lines are floating in a high-impedance state. Because of this pullup or pulldown, resistors must be attached to Port 0 lines to force them to a valid logic level until Port 0 is configured as an address port.

Port 0 initialization is discussed in the section on ports. An example of an initialization routine for Z8681/Z8682 compatibility is shown in Table 2. Only the Z8681 need execute this program.

Table 2. Initialization Routine

Address	Opcodes	Instruction	Comments
000C	E6 00 00	LD PO #%00	Set A_8-A_{15} to 0.
000F	E6 F8 96	LD P01M #%96	Configure Port 0 as A_8-A_{15}. Eliminate extended memory timing.
0012	8D 08 12	JP START ADDRESS	Execute application program.

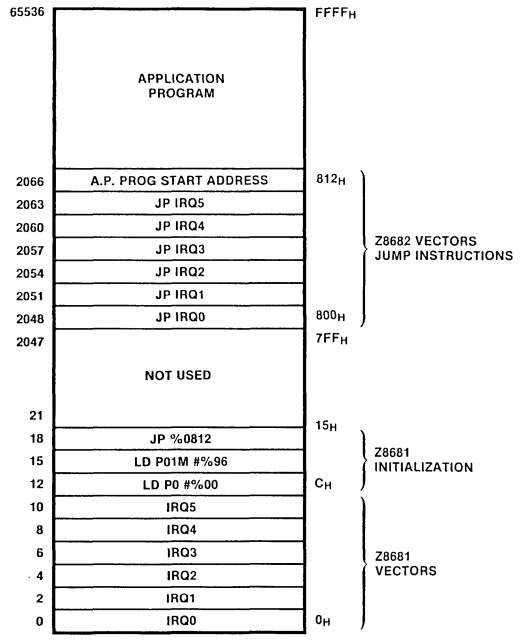
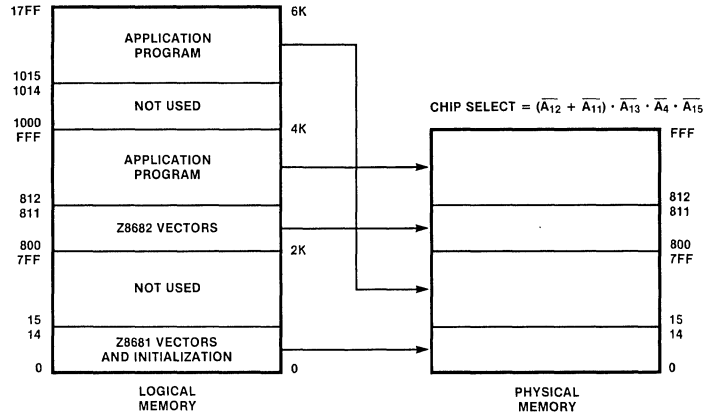
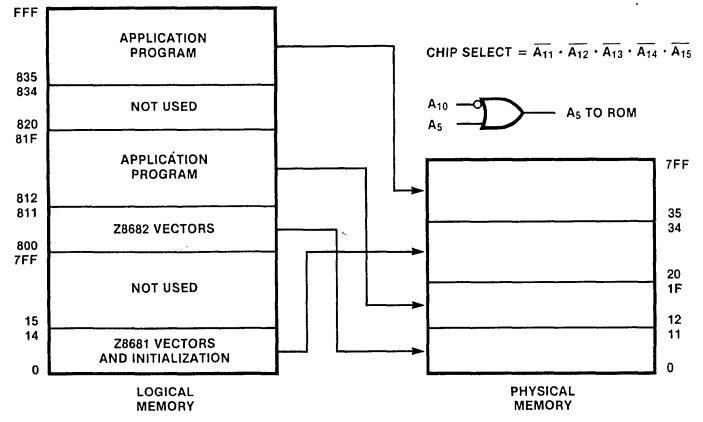


Figure 15. Z8681/82 Logical Program Memory Mapping

Memory Mapping. The Z8681 and Z8682 lower memory boundaries are located at 0 and 2048, respectively. A single program ROM can be used with either product if the logical program memory map shown in Figure 15 is followed. The Z8681 vectors and initialization routine must be starting at address 0 and the Z8682 3-byte vectors (jump instructions) must be at address 2048 and higher. Addresses in the range 21-2047 are not used. Figure 16 shows practical schemes for implementing this memory map using 4K and 2K ROMs.



a. Logical to Physical Memory Mapping for 4K ROM



b. Logical to Physical Memory Mapping for 2K ROM

Figure 16. Practical Schemes for Implementing Z8681 and Z8682 Compatible Memory Map

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

dst	Destination location or contents
src	Source location or contents
cc	Condition code (see list)
@	Indirect address prefix
SP	Stack pointer (control registers 254-255)
PC	Program counter
FLAGS	Flag register (control register 252)
RP	Register pointer (control register 253)
IMR	Interrupt mask register (control register 251)

Assignment of a value is indicated by the symbol "←". For example,

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

dst (7)

refers to bit 7 of the destination operand.

Flags. Control Register R252 contains the following six flags:

C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

Affected flags are indicated by:

0	Cleared to zero
1	Set to one
*****	Set or cleared according to operation
—	Unaffected
X	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000		Always true	—
0111	C	Carry	C = 1
1111	NC	No carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	[Z OR (S XOR V)] = 0
0010	LE	Less than or equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned greater than or equal	C = 0
0111	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1
0000		Never true	—

INSTRUCTION FORMATS

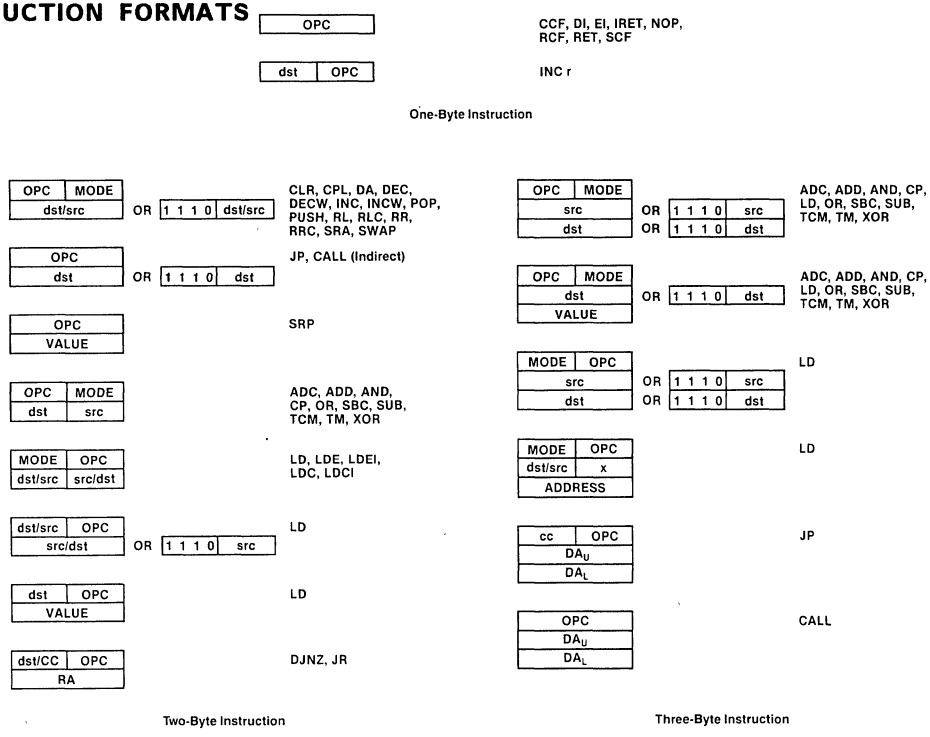
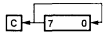
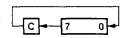
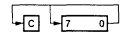
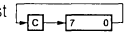
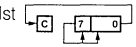
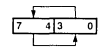


Figure 17. Instruction Formats

INSTRUCTION SUMMARY

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst,src dst ← dst + src + C	(Note 1)		1□	*	*	*	*	0	*
ADD dst,src dst ← dst + src	(Note 1)		0□	*	*	*	*	0	*
AND dst,src dst ← dst AND src	(Note 1)		5□	—	*	*	0	—	—
CALL dst SP ← SP − 2 @SP ← PC; PC ← dst	DA IRR		D6 D4	—	—	—	—	—	—
CCF C ← NOT C			EF	*	—	—	—	—	—
CLR dst dst ← 0	R IR		B0 B1	—	—	—	—	—	—
COM dst dst ← NOT dst	R IR		60 61	—	*	*	0	—	—
CP dst,src dst − src	(Note 1)		A□	*	*	*	*	—	—
DA dst dst ← DA dst	R IR		40 41	*	*	*	X	—	—
DEC dst dst ← dst − 1	R IR		00 01	—	*	*	*	—	—
DECW dst dst ← dst − 1	RR IR		80 81	—	*	*	*	—	—
DI IMR (7) ← 0			8F	—	—	—	—	—	—
DJNZ r,dst r ← r − 1 if r ≠ 0 PC ← PC + dst Range: +127, −128	RA		rA r = 0 − F	—	—	—	—	—	—
EI IMR (7) ← 1			9F	—	—	—	—	—	—
INC dst dst ← dst + 1	r R IR		rE r = 0 − F 20 21	—	*	*	*	—	—
INCW dst dst ← dst + 1	RR IR		A0 A1	—	*	*	*	—	—

INSTRUCTION SUMMARY (Continued)

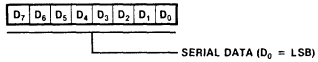
Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
IRET FLAGS ← @SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; IMR (7) ← 1			BF	*	*	*	*	*	*
JP cc,dst if cc is true PC ← dst	DA IRR		cD c = 0 – F 30	—	—	—	—	—	—
JR cc,dst if cc is true, PC ← PC + dst Range: +127, –128	RA		cB c = 0 – F	—	—	—	—	—	—
LD dst,src dst ← src	r r R r X r Ir R R IR IR	Im R r r = 0 – F X r Ir r R IR IM R	rC r8 r9 C7 D7 E3 F3 E4 E5 E6 E7 F5	—	—	—	—	—	—
LDC dst,src dst ← src	r Irr	Irr r	C2 D2	—	—	—	—	—	—
LDCI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	C3 D3	—	—	—	—	—	—
LDE dst,src dst ← src	r Irr	Irr r	82 92	—	—	—	—	—	—
LDEI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	83 93	—	—	—	—	—	—
NOP			FF	—	—	—	—	—	—
OR dst,src dst ← dst OR src	(Note 1)		4□	—	*	*	0	—	—
POP dst dst ← @SP; SP ← SP + 1	R IR		50 51	—	—	—	—	—	—
PUSH src SP ← SP – 1; @SP ← src		R IR	70 71	—	—	—	—	—	—
RCF C ← 0			CF	0	—	—	—	—	—
RET PC ← @SP; SP ← SP + 2			AF	—	—	—	—	—	—
RL dst 	R IR		90 91	*	*	*	*	—	—
RLC dst 	R IR		10 11	*	*	*	*	—	—
RR dst 	R IR		E0 E1	*	*	*	*	—	—
RRC dst 	R IR		C0 C1	*	*	*	*	—	—
SBC dst,src dst ← dst ← src ← C	(Note 1)		3□	*	*	*	*	1	*
SCF C ← 1			DF	1	—	—	—	—	—
SRA dst 	R IR		D0 D1	*	*	*	0	—	—
SRP src RP ← src		Im	31	—	—	—	—	—	—
SUB dst,src dst ← dst ← src	(Note 1)		2□	*	*	*	*	1	*
SWAP dst 	R IR		F0 F1	X	*	*	X	—	—
TCM dst,src (NOT dst) AND src	(Note 1)		6□	—	*	*	0	—	—
TM dst,src dst AND src	(Note 1)		7□	—	*	*	0	—	—
XOR dst,src dst ← dst XOR src	(Note 1)		B□	—	*	*	0	—	—

NOTE: These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a □ in this table, and its value is found in the following table to the left of the applicable addressing mode pair

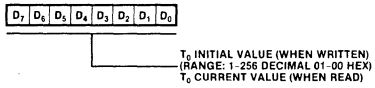
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13

Addr Mode dst	Addr Mode src	Lower Opcode Nibble
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7

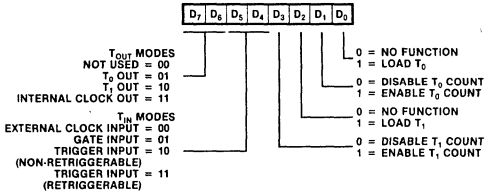
R240 SIO
Serial I/O Register
($F0_H$; Read/Write)



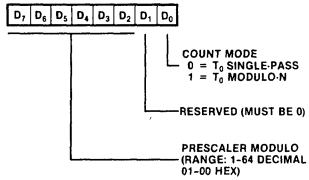
R244 TO
Counter/Timer 0 Register
($F4_H$; Read/Write)



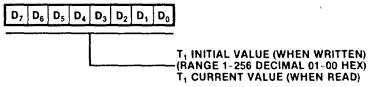
R241 TMR
Time Mode Register
($F1_H$; Read/Write)



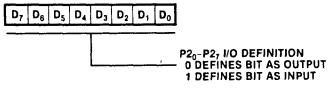
R245 PRE0
Prescaler 0 Register
($F5_H$; Write Only)



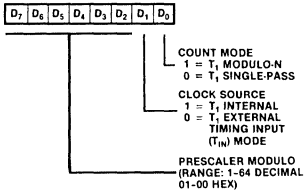
R242 T1
Counter Timer 1 Register
($F2_H$; Read/Write)



R246 P2M
Port 2 Mode Register
($F6_H$; Write Only)



R243 PRE1
Prescaler 1 Register
($F3_H$; Write Only)



R247 P3M
Port 3 Mode Register
($F7_H$; Write Only)

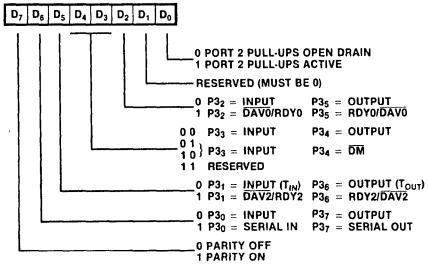
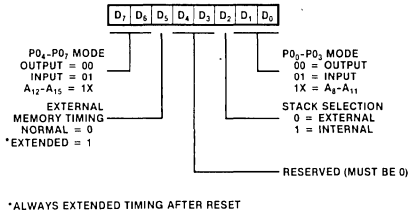


Figure 18. Control Registers

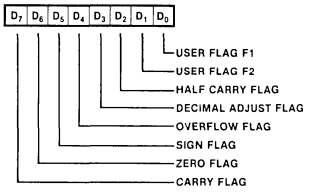
REGISTERS
(Continued)

R248 P01M
Port 0 Register
(F8H; Write Only)

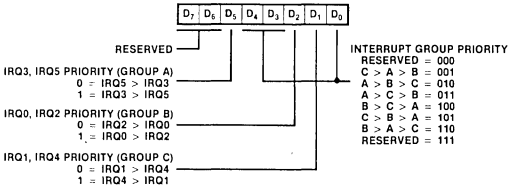


*ALWAYS EXTENDED TIMING AFTER RESET

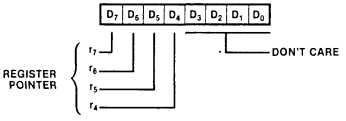
R252 FLAGS
Flag Register
(FCH; Read/Write)



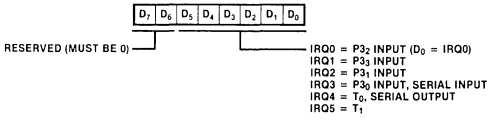
R249 IPR
Interrupt Priority Register
(F9H; Write Only)



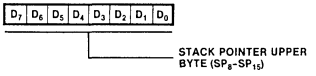
R253 RP
Register Pointer
(FDH, Read/Write)



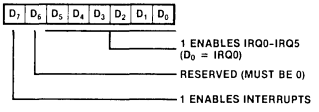
R250 IRQ
Interrupt Request Register
(FAH; Read/Write)



R254 SPH
Stack Pointer
(FEH; Read/Write)



R251 IMR
Interrupt Mask Register
(FBH; Read/Write)



R255 SPL
Stack Pointer
(FFH; Read/Write)

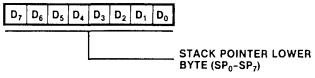
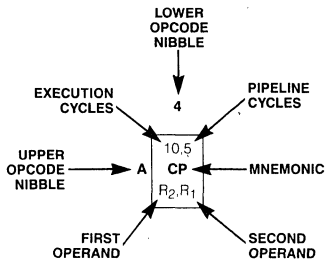


Figure 18. Control Registers (Continued)

Z8681/82 OPCODE MAP

Upper Nibble (Hex) \ Lower Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6,5 **DEC** R_1	6,5 **DEC** IR_1	6,5 **ADD** r_1,r_2	6,5 **ADD** r_1,Ir_2	10,5 **ADD** R_2,R_1	10,5 **ADD** IR_2,R_1	10,5 **ADD** R_1,IM	10,5 **ADD** IR_1,IM	6,5 **LD** r_1,R_2	6,5 **LD** r_2,R_1*	12/10,5 **DJNZ** r_1,RA	12/10,0 **JR** cc,RA	6,5 **LD** r_1,IM	12/10,0 **JP** cc,DA	6,5 **INC** r1	
1	6,5 **RLC** R_1	6,5 **RLC** IR_1	6,5 **ADC** r_1,r_2	6,5 **ADC** r_1,Ir_2	10,5 **ADC** R_2,R_1	10,5 **ADC** IR_2,R_1	10,5 **ADC** R_1,IM	10,5 **ADC** IR_1,IM								
2	6,5 **INC** R_1	6,5 **INC** IR_1	6,5 **SUB** r_1,r_2	6,5 **SUB** r_1,Ir_2	10,5 **SUB** R_2,R_1	10,5 **SUB** IR_2,R_1	10,5 **SUB** R_1,IM	10,5 **SUB** IR_1,IM								
3	8,0 **JP** IRR_1	6,1 **SRP** IM	6,5 **SBC** r_1,r_2	6,5 **SBC** r_1,Ir_2	10,5 **SBC** R_2,R_1	10,5 **SBC** IR_2,R_1	10,5 **SBC** R_1,IM	10,5 **SBC** IR_1,IM								
4	8,5 **DA** R_1	8,5 **DA** IR_1	6,5 **OR** r_1,r_2	6,5 **OR** r_1,Ir_2	10,5 **OR** R_2,R_1	10,5 **OR** IR_2,R_1	10,5 **OR** R_1,IM	10,5 **OR** IR_1,IM								
5	10,5 **POP** R_1	10,5 **POP** IR_1	6,5 **AND** r_1,r_2	6,5 **AND** r_1,Ir_2	10,5 **AND** R_2,R_1	10,5 **AND** IR_2,R_1	10,5 **AND** R_1,IM	10,5 **AND** IR_1,IM								
6	6,5 **COM** R_1	6,5 **COM** IR_1	6,5 **TCM** r_1,r_2	6,5 **TCM** r_1,Ir_2	10,5 **TCM** R_2,R_1	10,5 **TCM** IR_2,R_1	10,5 **TCM** R_1,IM	10,5 **TCM** IR_1,IM								
7	10/12,1 **PUSH** R_2	12/14,1 **PUSH** IR_2	6,5 **TM** r_1,r_2	6,5 **TM** r_1,Ir_2	10,5 **TM** R_2,R_1	10,5 **TM** IR_2,R_1	10,5 **TM** R_1,IM	10,5 **TM** IR_1,IM								
8	10,5 **DECW** RR_1	10,5 **DECW** IR_1	12,0 **LDE** r_1,Irr_2	18,0 **LDEI** Ir_1,Irr_2												6,1 **DI**
9	6,5 **RL** R_1	6,5 **RL** IR_1	12,0 **LDE** r_2,Irr_1	18,0 **LDEI** Ir_2,Irr_1												6,1 **EI**
A	10,5 **INCW** RR_1	10,5 **INCW** IR_1	6,5 **CP** r_1,r_2	6,5 **CP** r_1,Ir_2	10,5 **CP** R_2,R_1	10,5 **CP** IR_2,R_1	10,5 **CP** R_1,IM	10,5 **CP** IR_1,IM								14,0 **RET**
B	6,5 **CLR** R_1	6,5 **CLR** IR_1	6,5 **XOR** r_1,r_2	6,5 **XOR** r_1,Ir_2	10,5 **XOR** R_2,R_1	10,5 **XOR** IR_2,R_1	10,5 **XOR** R_1,IM	10,5 **XOR** IR_1,IM								16,0 **IRET**
C	6,5 **RRC** R_1	6,5 **RRC** IR_1	12,0 **LDC** r_1,Irr_2	18,0 **LDCI** Ir_1,Irr_2				10,5 **LD** r_1,x,R_2								6,5 **RCF**
D	6,5 **SRA** R_1	6,5 **SRA** IR_1	12,0 **LDC** r_2,Irr_1	18,0 **LDCI** Ir_2,Irr_1	20,0 **CALL*** IRR_1		20,0 **CALL** DA	10,5 **LD** r_2,x,R_1								6,5 **SCF**
E	6,5 **RR** R_1	6,5 **RR** IR_1		6,5 **LD** r_1,IR_2	10,5 **LD** R_2,R_1	10,5 **LD** IR_2,R_1	10,5 **LD** R_1,IM	10,5 **LD** IR_1,IM								6,5 **CCF**
F	8,5 **SWAP** R_1	8,5 **SWAP** IR_1		6,5 **LD** Ir_1,r_2		10,5 **LD** R_2,IR_1										6,0 **NOP**
Bytes per Instruction	2				3				2					3	1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_1 = Dst address
R_2 or r_2 = Src address

Sequence:
Opcode, First Operand, Second Operand

NOTE: The blank areas are not defined

*2-byte instruction, fetch cycle appears as a 3-byte instruction

ABSOLUTE MAXIMUM RATINGS

Voltages on all pins except $\overline{\text{RESET}}$
with respect to GND . . . . . −0.3V to +7.0V
Operating Ambient
Temperature . . . . . .See Ordering Information
Storage Temperature . . . . . −65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

STANDARD TEST CONDITIONS

The DC characteristics listed below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin.

Standard conditions are as follows:

- $+4.75V \leqslant V_{CC} \leqslant +5.25V$
- GND = 0V
- $0°C \leqslant T_A \leqslant +70°C$ for S (Standard temperature)
- $-40°C \leqslant T_A \leqslant +100°C$ for E (Extended temperature)

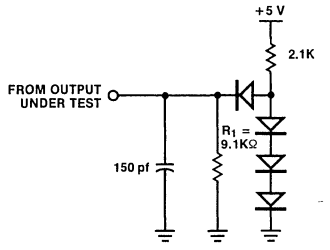


Figure 19. Test Load 1

DC CHARACTERISTICS

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	3.8	V_{CC}	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	−0.3	0.8	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{IL}	Input Low Voltage	−0.3	0.8	V	
V_{RH}	Reset Input High Voltage	3.8	V_{CC}	V	See Note
V_{RL}	Reset Input Low Voltage	−0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\,\mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0$ mA
I_{IL}	Input Leakage	−10	10	μA	$0V \leqslant V_{IN} \leqslant +5.25V$
I_{OL}	Output Leakage	−10	10	μA	$0V \leqslant V_{IN} \leqslant +5.25V$
I_{IR}	Reset Input Current		−50	μA	$V_{CC} = +5.25V$, $V_{RL} = 0V$
I_{CC}	V_{CC} Supply Current		**150**	mA	All outputs and I/O pins floating

*The Reset line (pin 6) is used to place the Z8682 in external memory mode. This is accomplished as shown in Figure 13.

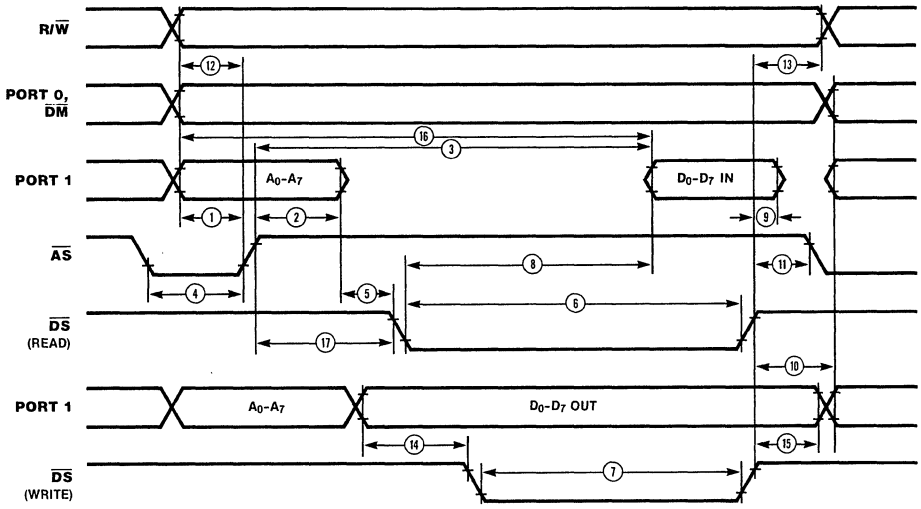


Figure 20. External I/O or Memory Read/Write Timing

AC CHARACTERISTICS

External I/O or Memory Read and Write Timing

Number	Symbol	Parameter	Z8681/82 8 MHz Min	Z8681/82 8 MHz Max	Z8681 12 MHz Min	Z8681 12 MHz Max	Z8681 16 MHz Min	Z8681 16 MHz Max	Notes
1	TdA(AS)	Address Valid to $\overline{AS}$ ↑Delay	50		35		20		2,3
2	TdAS(A)	$\overline{AS}$ ↑ to Address Float Delay	70		45		30		2,3
3	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Required Valid		360		220		180	1,2,3
4	TwAS	$\overline{AS}$ Low Width	80		55		35		2,3
5	TdAz(DS)	Address Float to $\overline{DS}$ ↓	0		0		0		
6	TwDSR	$\overline{DS}$ (Read) Low Width	250		185		135		1,2,3
7	TwDSW	$\overline{DS}$ (Write) Low Width	160		110		80		1,2,3
8	TdDSR(DR)	$\overline{DS}$ ↓ to Read Data Required Valid		200		130		75	1,2,3
9	ThDR(DS)	Read Data to $\overline{DS}$ ↑ Hold Time	0		0		0		2,3
10	TdDS(A)	$\overline{DS}$ ↑ to Address Active Delay	70		45		-		2,3
11	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓Delay	70		55		30		2,3
12	TdR/W(AS)	R/W Valid to $\overline{AS}$ ↑ Delay	50		30		20		2,3
13	TdDS(R/W)	$\overline{DS}$ ↑ to R/W Not Valid	60		35		30		2,3
14	TdDW(DSW)	Write Data Valid to $\overline{DS}$ (Write) ↓ Delay	50		35		25		2,3
15	TdDS(DW)	$\overline{DS}$ ↑ to Write Data Not Valid Delay	60		35		30		2,3
16	TdA(DR)	Address Valid to Read Data Required Valid		410		255		200	1,2,3
17	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	80		55		40		2,3

NOTES:
1. When using extended memory timing add 2 TpC.
2. Timing numbers given are for minimum TpC.
3. See clock cycle time dependent characteristics table.
4. **16 MHz timing is preliminary and subject to change.**

* All units in nanoseconds (ns).
† Test Load 1
° All timing references use 2.0V for a logic "1" and 0.8V for a logic "0".

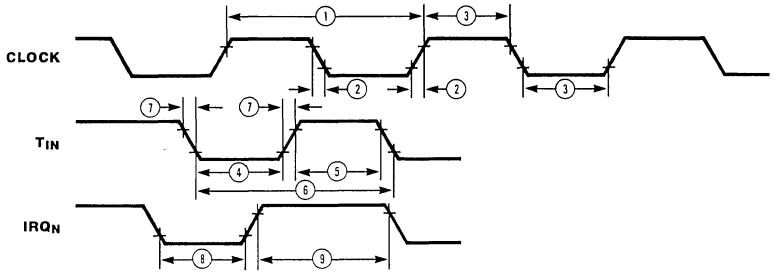


Figure 21. Additional Timing

AC CHARACTERISTICS

Additional Timing Table

Number	Symbol	Parameter	Z8681/82 8 MHz Min	Z8681/82 8 MHz Max	Z8681 12 MHz Min	Z8681 12 MHz Max	Z8681 16 MHz Min	Z8681 16 MHz Max	Notes
1	TpC	Input Clock Period	125	1000	83	1000	62.5	1000	1
2	TrC,TfC	Clock Input Rise and Fall Times		25		15		10	1
3	TwC	Input Clock Width	37		70		21		1
4	TwTinL	Timer Input Low Width	100		70		50		2
5	TwTinH	Timer Input High Width	3TpC		3TpC		3TpC		2
6	TpTin	Timer Input Period	8TpC		8TpC		8TpC		2
7	TrTin,TfTin	Timer Input Rise and Fall Times		100		100		100	2
8A	TwIL	Interrupt Request Input Low Time	100		70		50		2,4
8B	TwIL	Interrupt Request Input Low Time	3TpC		3TpC		3TpC		2,5
9	TwIH	Interrupt Request Input High Time	3TpC		3TpC		3TpC		2,3

NOTES:
1. Clock timing references use 3.8V for a logic "1" and 0.8V for a logic "0".
2. Timing references use 2.0V for a logic "1" and 0.8V for a logic "0".
3. Interrupt request via Port 3.
4. Interrupt request via Port 3 ($P3_1$-$P3_3$)
5. Interrupt request via Port 3 ($P3_0$)
6. **16 MHz timing is preliminary and subject to change.**

* Units in nanoseconds (ns).

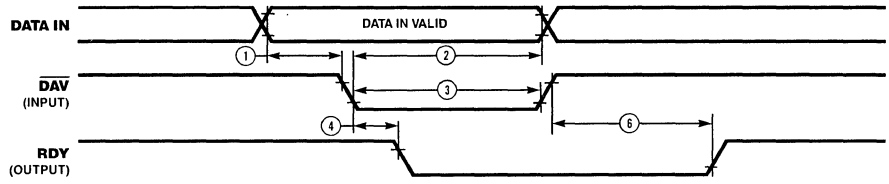


Figure 22a. Input Handshake Timing

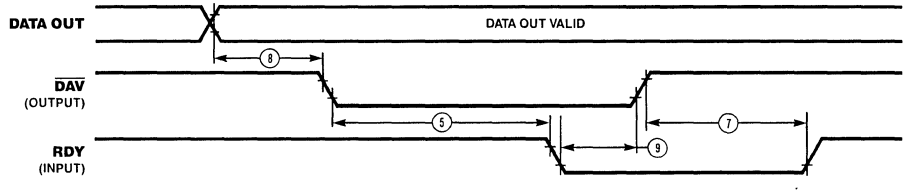


Figure 22b. Output Handshake Timing

AC CHARACTERISTICS

Handshake Timing

Number	Symbol	Parameter	Z8681/82 8 MHz Min	Z8681/82 8 MHz Max	Z8681 12 MHz Min	Z8681 12 MHz Max	Z8681 16 MHz Min	Z8681 16 MHz Max	Notes
1	TsDI(DAV)	Data In Setup Time	0		0		0		
2	ThDI(DAV)	Data In Hold Time	230		160		145		
3	TwDAV	Data Available Width	175		120		110		
4	TdDAVlf(RDY)	DAV ↓ Input to RDY ↓ Delay		175		120		115	1,2
5	TdDAVOf(RDY)	DAV ↓ Output to RDY ↓ Delay	0		0		0		1,3
6	TdDAVlr(RDY)	DAV ↑ Input to RDY ↑ Delay		175		120		115	1,2
7	TdDAVOr(RDY)	DAV ↑ Output to RDY ↑ Delay	0		0		0		1,3
8	TdDO(DAV)	Data Out to DAV ↓ Delay	50		30		30		1
9	TdRDY(DAV)	Rdy ↓ Input to DAV ↑ Delay	0	200	0	140	0	130	1

NOTES:
1. Test load 1
2. Input handshake
3. Output handshake
4. **16 MHz timing is preliminary and subject to change.**

† All timing references use 2.0V for a logic "1" and 0.8V for a logic "0".
* Units in nanoseconds (ns).

CLOCK CYCLE TIME-DEPENDENT CHARACTERISTICS

Number	Symbol	Z8681/82 8 MHz Equation	Z8681/82 12 MHz Equation
1	TdA(AS)	TpC-75	TpC-50
2	TdAS(A)	TpC-55	TpC-40
3	TdAS(DR)	4TpC-140 *	4TpC-110 *
4	TwAS	TpC-45	TpC-30
6	TwDSR	3TpC-125 *	3TpC-65 *
7	TwDSW	2TpC-90 *	2TpC-55 *
8	TdDSR(DR)	3TpC-175 *	3TpC-120 *
10	Td(DS)A	TpC-55	TpC-40
11	TdDS(AS)	TpC-55	TpC-30
12	TdR/W(AS)	TpC-75	TpC-55
13	TdDS(R/W)	TpC-65	TpC-50
14	TdDW(DSW)	TpC-75	TpC-50
15	TdDS(DW)	TpC-55	TpC-40
16	TdA(DR)	5TpC-215 *	5TpC-160 *
17	TdAS(DS)	TpC-45	TpC-30

* Add 2TpC when using extended memory timing

Z8691 Z8®

ROMLESS MICROCONTROLLER

FEATURES

- Complete microcomputer, 24 I/O lines, and up to 64K bytes of addressable external space each for program and data memory.
- 143-byte register file, including 124 general-purpose registers, 3 I/O port registers, and 16 status and control registers.
- Vectored, priority interrupts for I/O, counter/timers, and UART.
- On-chip oscillator that accepts crystal or external clock drive.
- Full-duplex UART and two programmable 8-bit counter/timers, each with a 6-bit programmable prescaler.
- Register Pointer so that short, fast instructions can access any one of the nine working-register groups.
- Single +5V power supply—all I/O pins TTL compatible.
- 8 MHz/12 MHz versions.

GENERAL DESCRIPTION

The Z8691 is a ROMless version of the Z8 single-chip microcomputer. The Z8691 offers all the outstanding features of the Z8 family architecture except an on-chip program ROM. Use of external memory rather than a preprogrammed ROM enables this Z8 microcomputer to be used in low volume applications or where code flexibility is required.

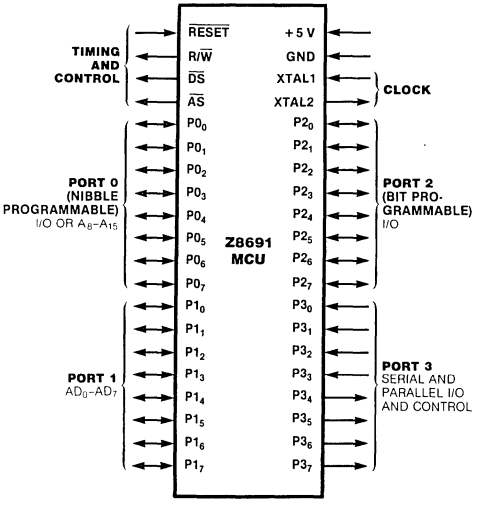


Figure 1. Pin Functions

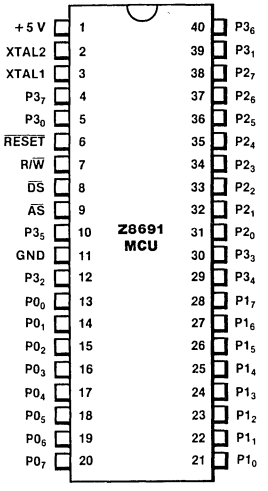


Figure 2a. 40-pin Dual-In-Line Package (DIP), Pin Assignments

The Z8691 can provide up to 16 output address lines, thus permitting an address space of up to 64K bytes of data or program memory. Eight address outputs (AD_0-AD_7) are provided by a multiplexed, 8-bit, Address/Data bus. The remaining 8 bits can be provided by the software configuration of Port 0 to output address bits A_8-A_{15}.

Available address space can be doubled (up to 128K bytes) by programming bit 4 of Port 3 ($P3_4$) to act as a data memory select output ($\overline{DM}$). The two states of $\overline{DM}$ together with the 16 address outputs can define separate data and memory address spaces of up to 64K bytes each.

There are 143 bytes of RAM located on-chip and organized as a register file of 124 general-purpose registers, 16 control and status registers, and three I/O port registers. This register file can be divided into nine groups of 16 working registers each. Configuring the register file in this manner allows the use of short format instructions; in addition, any of the individual registers can be accessed directly.

The pin functions and the pin assignments of the Z8691 40-pin and 44-pin packages are illustrated in Figures 1 and 2, respectively.

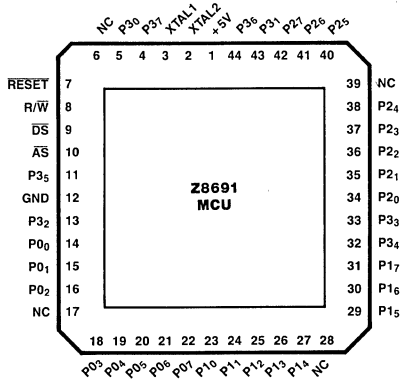


Figure 2b. 44-pin Chip Carrier, Pin Assignments

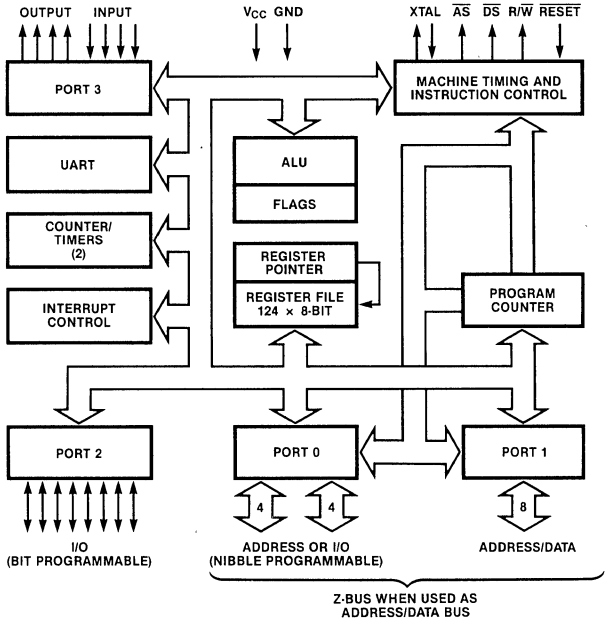


Figure 3. Functional Block Diagram

ARCHITECTURE

Z8691 architecture is characterized by a flexible I/O scheme, an efficient register and address space structure and a number of ancillary features that are helpful in many applications.

Microcomputer applications demand powerful I/O capabilities. The Z8691 fulfills this with 24 pins available for input and output. These lines are grouped into three ports of eight lines each and are configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an Address bus for interfacing external memory.

Three basic address spaces are available: program memory, data memory and the register file (internal). The 143-byte random-access register file is composed of 124 general-purpose registers, three I/O port registers, and 16 control and status registers.

To unburden the program from coping with real-time problems such as serial data communication and counting/timing, an asynchronous receiver/transmitter (UART) and two counter/timers with a large number of user-selectable modes are offered on-chip. Hardware support for the UART is minimized because one of the on-chip timers supplies the bit rate. Figure 3 shows the Z8691 block diagram.

PIN DESCRIPTION

$\overline{AS}$. *Address Strobe* (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Addresses output via Port 1 for all external program or data memory transfers are valid at the trailing edge of $\overline{AS}$.

$\overline{DS}$. *Data Strobe* (output, active Low). Data Strobe is activated once for each external memory transfer.

$P0_0$-$P0_7$, $P2_0$-$P2_7$, $P3_0$-$P3_7$. *I/O Port Lines* (input/outputs, TTL-compatible). These 24 lines are divided into three 8-bit I/O ports that can be configured under program control for I/O or external memory interface (Figure 3).

$P1_0$-$P1_7$. *Address/Data Port* (bidirectional). Multiplexed address (A_0-A_7) and data (D_0-D_7) lines used to interface with program and data memory.

$\overline{RESET}$. *Reset* (input, active Low). $\overline{RESET}$ initializes the Z8691. After $\overline{RESET}$ the Z8691 is in the extended memory mode. When $\overline{RESET}$ is deactivated, program execution begins from program location $000C_H$.

R/$\overline{W}$. *Read/Write* (output). R/$\overline{W}$ is Low when the Z8691 is writing to external program or data memory.

XTAL1, XTAL2. *Crystal 1, Crystal 2* (time-base input and output). These pins connect a parallel-resonant crystal to the on-chip clock oscillator and buffer.

ADDRESS SPACES

Program Memory. The Z8691 addresses 64K/62K bytes of external program memory space (Figure 4).

The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts. Program execution begins at location $000C_H$ after a reset.

Data Memory. The Z8691 can address 64K bytes of external data memory. External data memory may be included with or separated from the external program memory space. $\overline{DM}$, an optional I/O function that can be programmed to appear on pin $P3_4$, is used to distinguish between data and program memory space.

Register File. The 143-byte register file includes three I/O port registers (R0, R2, R3), 124 general-purpose registers (R4-R127) and 16 control and status registers (R240-R255). These registers are assigned the address locations shown in Figure 5.

Z8691 instructions can access registers directly or indirectly with an 8-bit address field. This also allows short 4-bit register addressing using the Register Pointer (one of the control registers). In the 4-bit mode, the register file is divided into nine working-register groups, each occupying 16 contiguous locations (Figure 5). The Register Pointer addresses the starting location of the active working-register group (Figure 6).

Stacks. Either the internal register file or the external data memory can be used for the stack. A 16-bit Stack Pointer (R254 and R255) is used for the external stack, which can reside anywhere in data memory. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 124 general-purpose registers (R4-R127).

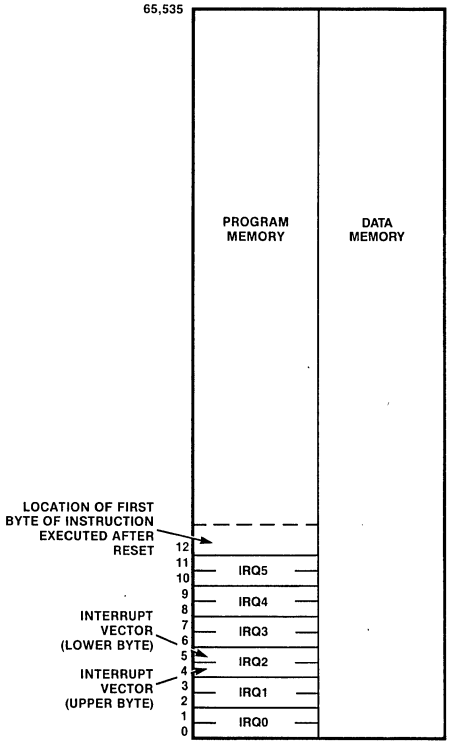


Figure 4. Program Memory Map

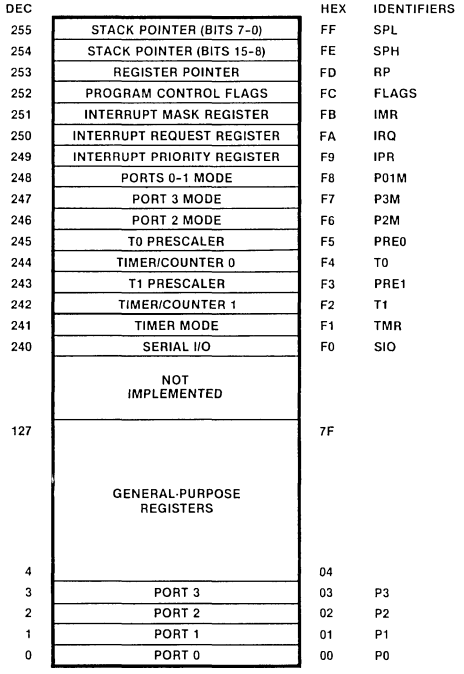


Figure 5. The Register File

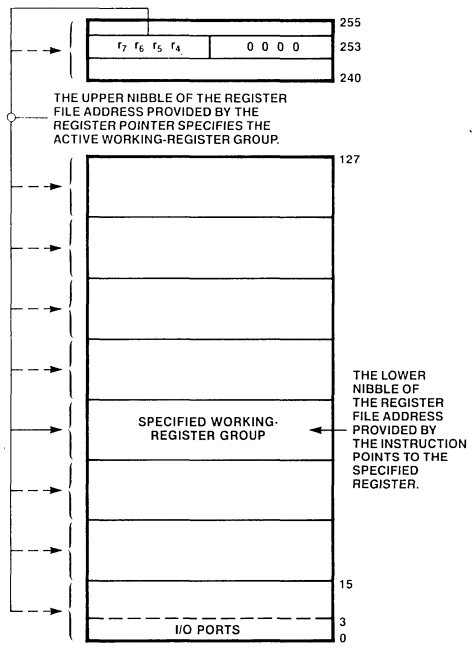


Figure 6. The Register Pointer

SERIAL INPUT/OUTPUT

Port 3 lines $P3_0$ and $P3_7$ can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by Counter/Timer 0, with a maximum rate of 62.5K bits/second at 8 MHz or 93.75K bits/second at 12 MHz on the Z8691.

The Z8691 automatically adds a start bit and two stop bits to transmitted data (Figure 7). Odd parity is also available as an option. Eight data bits are always transmitted, regardless of parity selection. If parity is enabled, the eighth data bit is used as the odd parity bit. An interrupt request (IRQ4) is generated on all transmitted characters.

Received data must have a start bit, eight data bits, and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ3 interrupt request.

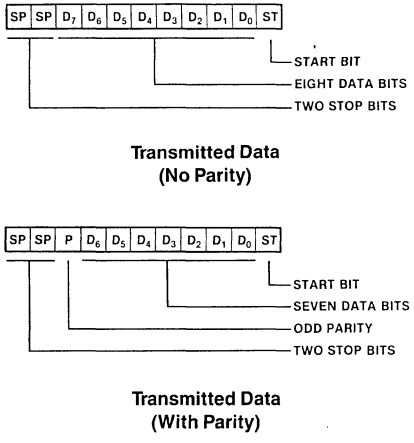


Figure 7. Serial Data Formats

COUNTER/TIMERS

The Z8691 contains two 8-bit programmable counter/timers (T_0 and T_1), each driven by its own 6-bit programmable prescaler. The T_1 prescaler can be driven by internal or external clock sources; however, the T_0 prescaler is driven by the internal clock only.

The 6-bit prescalers can divide the input frequency of the clock source by any number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request—IRQ4 (T_0) or IRQ5 (T_1)—is generated.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. The counters can also be programmed to stop upon reaching zero (single-pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode). The counters, but not the prescalers, can be read any time without disturbing their value or count mode.

The clock source for T_1 is user-definable; it can be either the internal microprocessor clock divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input as an external clock, a trigger input that can be retriggerable or nonretriggerable, or as a gate input for the internal clock. The counter/timers can be programmably cascaded by connecting the T_0 output to the input of T_1. Port 3 line $P3_6$ also serves as a timer output (T_{OUT}) through which T_0, T_1 or the internal clock can be output.

I/O PORTS

The Z8691 has 24 lines available for input and output. These lines are grouped into three ports of eight lines each and are configurable as input, output or address. Under software control, the ports can be programmed to provide address outputs, timing, status signals, serial I/O, and parallel I/O with or without handshake. All ports have active pull-ups and pull-downs compatible with TTL loads.

Port 1 is a dedicated Z-BUS compatible memory interface. The operations of Port 1 are supported by the Address Strobe ($\overline{AS}$) and Data Strobe ($\overline{DS}$) lines, and by the Read/Write (R/$\overline{W}$) and Data Memory ($\overline{DM}$) control lines. The low-order program and data memory addresses (A_0-A_7) are output through Port 1 (Figure 8) and are multiplexed with data in/out (D_0-D_7). Instruction fetch and data memory read/write operations are done through this port.

Port 1 cannot be used as a register nor can a handshake mode be used with this port.

The Z8691 wakes up with the 8 bits of Port 1 configured as address outputs for external memory. If more than eight address lines are required, additional lines can be obtained by programming Port 0 bits as address bits. The least-significant four bits of Port 0 can be configured to supply address bits A_8-A_{11} for 4K byte addressing or both nibbles of Port 0 can be configured to supply address bits A_8-A_{15} for 64K byte addressing.

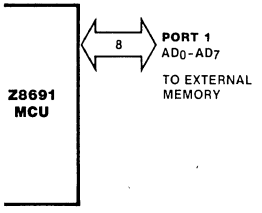


Figure 8. Port 1

Port 0 can be programmed as a nibble I/O port, or as an address port for interfacing external memory (Figure 9). When used as an I/O port, Port 0 can be placed under handshake control. In this configuration, Port 3 lines $P3_2$ and $P3_5$ are used as the handshake controls DAV_0 and RDY_0. Handshake signal assignment is dictated by the I/O direction of the upper nibble $P0_4$-$P0_7$.

For external memory references, Port 0 can provide address bits A_8-A_{11} (lower nibble) or A_8-A_{15} (lower and upper nibbles) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing.

Port 0 lines are configured as address lines A_8-A_{15} after a reset. If one or both nibbles are needed for I/O operation, they must be configured by writing to the Port 0 Mode register.

To permit the use of slow memory, an automatic wait mode of two oscillator clock cycles is configured for the bus timing of the Z8691 after each reset. The initialization routine could include reconfiguration to eliminate this extended timing mode.

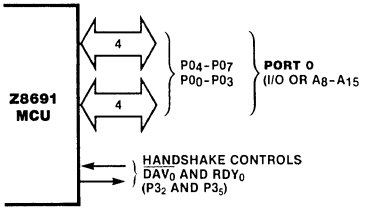


Figure 9. Port 0

Port 2 bits can be programmed independently as input or output (Figure 10). This port is always available for I/O operations. In addition, Port 2 can be configured to provide open-drain outputs.

Port 2 may also be placed under handshake control. In this configuration, Port 3 lines $P3_1$ and $P3_6$ are used as the handshake controls lines $\overline{DAV}_2$ and RDY_2. The handshake signal assignment for Port 3 lines $P3_1$ and $P3_6$ is dictated by the direction (input or output) assigned to bit 7 of Port 2.

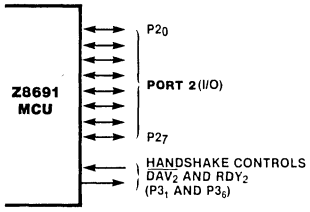


Figure 10. Port 2

Port 3 lines can be configured as I/O or control lines (Figure 11). In either case, the direction of the eight lines is fixed as four input ($P3_0$-$P3_3$) and four output ($P3_4$-$P3_7$). For serial I/O, lines $P3_0$ and $P3_7$ are programmed as serial in and serial out, respectively.

Port 3 can also provide the following control functions: handshake for Ports 0 and 2 ($\overline{DAV}$ and RDY); four external interrupt request signals (IRQ0-IRQ3); timer input and output signals (T_{IN} and T_{OUT}) and Data Memory Select ($\overline{DM}$).

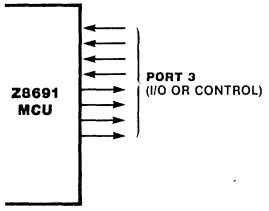


Figure 11. Port 3

INTERRUPTS

The Z8691 allows six different interrupts from eight sources: the four Port 3 lines $P3_0$-$P3_3$, Serial In, Serial Out, and the two counter/timers. These interrupts are both maskable and prioritized. The Interrupt Mask register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All interrupts are vectored through locations in program memory. When an interrupt request is granted, an interrupt machine cycle is entered. This disables all subsequent interrupts, saves the Program Counter and status flags, and accesses the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request. The Z8691 takes 63 crystal cycles to enter an interrupt subroutine.

Polled interrupt systems are also supported. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

CLOCK

The on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal or to any suitable external clock source (XTAL1 = Input, XTAL2 = Output).

The crystal source is connected across XTAL1 and XTAL2, using the recommended capacitance (C_L = 15 pf maximum) from each pin to ground. The specifications for the crystal are as follows:

- AT cut, parallel-resonant
- Fundamental type
- Series resistance, $R_S \leq 100\ \Omega$
- 8 or 12 MHz maximum

INSTRUCTION SET NOTATION

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

dst	Destination location or contents
src	Source location or contents
cc	Condition code (see list)
@	Indirect address prefix
SP	Stack pointer (control registers 254-255)
PC	Program counter
FLAGS	Flag register (control register 252)
RP	Register pointer (control register 253)
IMR	Interrupt mask register (control register 251)

Assignment of a value is indicated by the symbol "←". For example,

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

dst (7)

refers to bit 7 of the destination operand.

Flags. Control Register R252 contains the following six flags:

C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

Affected flags are indicated by:

0	Cleared to zero
1	Set to one
*****	Set or cleared according to operation
—	Unaffected
X	Undefined

CONDITION CODES

Value	Mnemonic	Meaning	Flags Set
1000		Always true	—
0111	C	Carry	C = 1
1111	NC	No carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	[Z OR (S XOR V)] = 0
0010	LE	Less than or equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned greater than or equal	C = 0
0111	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1
0000		Never true	—

INSTRUCTION FORMATS

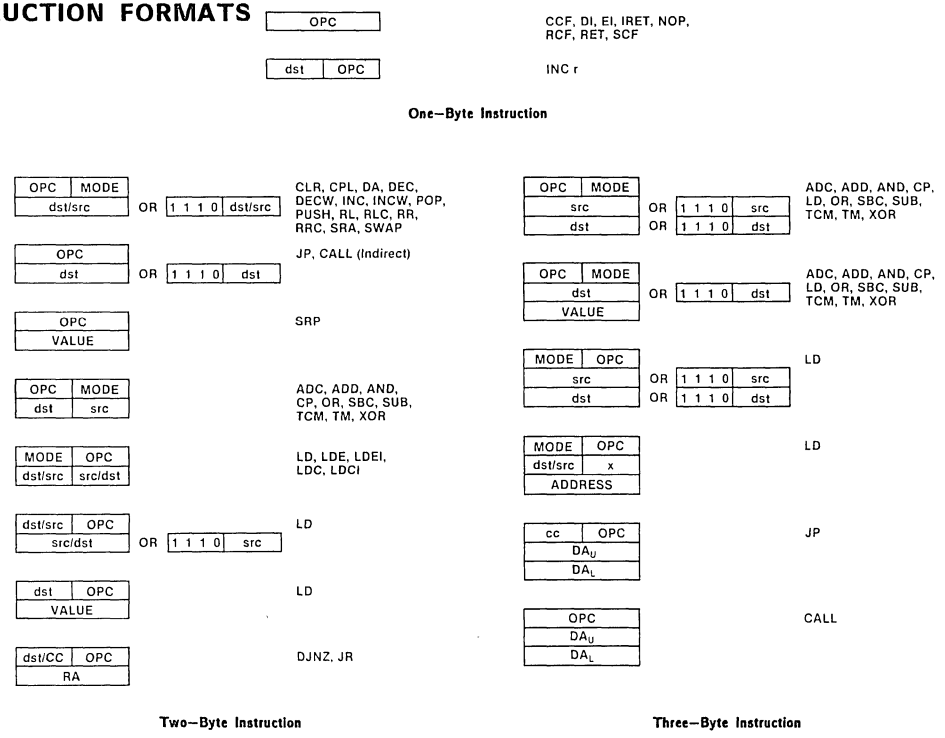


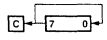
Figure 12. Instruction Formats

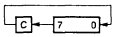
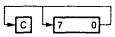
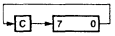
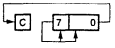
INSTRUCTION SUMMARY

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst,src dst ← dst + src + C	(Note 1)		1□	*	*	*	*	0	*
ADD dst,src dst ← dst + src	(Note 1)		0□	*	*	*	*	0	*
AND dst,src dst ← dst AND src	(Note 1)		5□	—	*	*	0	—	—
CALL dst SP ← SP − 2 @SP ← PC; PC ← dst	DA IRR		D6 D4	—	—	—	—	—	—
CCF C ← NOT C			EF	*	—	—	—	—	—
CLR dst dst ← 0	R IR		B0 B1	—	—	—	—	—	—
COM dst dst ← NOT dst	R IR		60 61	—	*	*	0	—	—
CP dst,src dst − src	(Note 1)		A□	*	*	*	*	—	—
DA dst dst ← DA dst	R IR		40 41	*	*	*	X	—	—

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
DEC dst dst ← dst − 1	R IR		00 01	—	*	*	*	—	—
DECW dst dst ← dst − 1	RR IR		80 81	—	*	*	*	—	—
DI IMR (7) ← 0			8F	—	—	—	—	—	—
DJNZ r,dst r ← r − 1 if r ≠ 0 PC ← PC + dst Range: +127, −128	RA		rA r = 0 − F	—	—	—	—	—	—
EI IMR (7) ← 1			9F	—	—	—	—	—	—
INC dst dst ← dst + 1	r R IR		rE r = 0 − F 20 21	—	*	*	*	—	—
INCW dst dst ← dst + 1	RR IR		A0 A1	—	*	*	*	—	—

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
IRET FLAGS ← @SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; IMR (7) ← 1			BF	*	*	*	*	*	*
JP cc,dst if cc is true PC ← dst	DA IRR		cD c = 0 – F 30	—	—	—	—	—	—
JR cc,dst if cc is true, PC ← PC + dst Range: +127, –128	RA		cB c = 0 – F	—	—	—	—	—	—
LD dst,src dst ← src	r r R r X r Ir R R R IR IR	Im R r X r Ir r R IR IM IM R	rC r8 r9 r = 0 – F C7 D7 E3 F3 E4 E5 E6 E7 F5	—	—	—	—	—	—
LDC dst,src dst ← src	r Irr	Irr r	C2 D2	—	—	—	—	—	—
LDCI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	C3 D3	—	—	—	—	—	—
LDE dst,src dst ← src	r Irr	Irr r	82 92	—	—	—	—	—	—
LDEI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	83 93	—	—	—	—	—	—
NOP			FF	—	—	—	—	—	—
OR dst,src dst ← dst OR src	(Note 1)		4□	—	*	*	0	—	—
POP dst dst ← @SP; SP ← SP + 1	R IR		50 51	—	—	—	—	—	—
PUSH src SP ← SP – 1; @SP ← src		R IR	70 71	—	—	—	—	—	—
RCF C ← 0			CF	0	—	—	—	—	—
RET PC ← @SP; SP ← SP + 2			AF	—	—	—	—	—	—
RL dst 	R IR		90 91	*	*	*	*	—	—

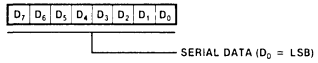
Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
RLC dst 	R IR		10 11	*	*	*	*	—	—
RR dst 	R IR		E0 E1	*	*	*	*	—	—
RRC dst 	R IR		C0 C1	*	*	*	*	—	—
SBC dst,src dst ← dst ← src ← C	(Note 1)		3□	*	*	*	*	1	*
SCF C ← 1			DF	1	—	—	—	—	—
SRA dst 	R IR		D0 D1	*	*	*	0	—	—
SRP src RP ← src		Im	31	—	—	—	—	—	—
SUB dst,src dst ← dst ← src	(Note 1)		2□	*	*	*	*	1	*
SWAP dst 	R IR		F0 F1	X	*	*	X	—	—
TCM dst,src (NOT dst) AND src	(Note 1)		6□	—	*	*	0	—	—
TM dst,src dst AND src	(Note 1)		7□	—	*	*	0	—	—
XOR dst,src dst ← dst XOR src	(Note 1)		B□	—	*	*	0	—	—

NOTE These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a □ in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

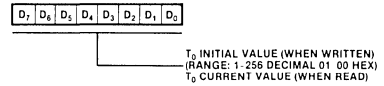
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Addr Mode dst	Addr Mode src	Lower Opcode Nibble
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7

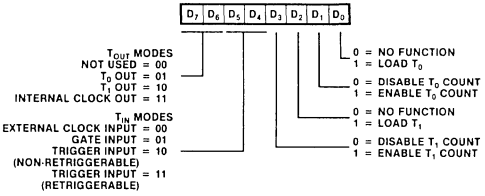
R240 SIO
Serial I/O Register
($F0_H$, Read/Write)



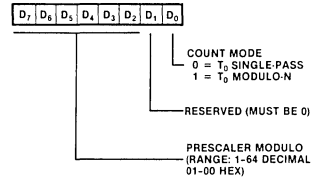
R244 TO
Counter/Timer 0 Register
($F4_H$; Read/Write)



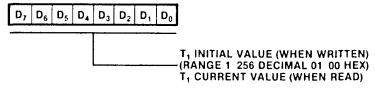
R241 TMR
Time Mode Register
($F1_H$; Read/Write)



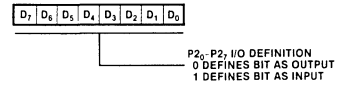
R245 PRE0
Prescaler 0 Register
($F5_H$; Write Only)



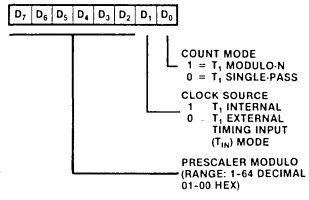
R242 T1
Counter Timer 1 Register
($F2_H$; Read/Write)



R246 P2M
Port 2 Mode Register
($F6_H$; Write Only)



R243 PRE1
Prescaler 1 Register
($F3_H$; Write Only)



R247 P3M
Port 3 Mode Register
($F7_H$; Write Only)

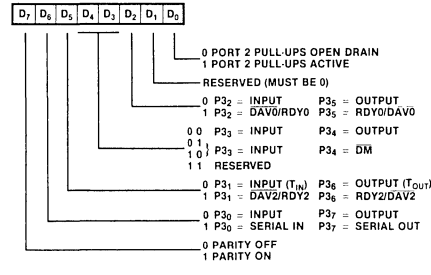
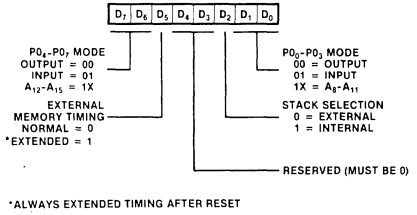


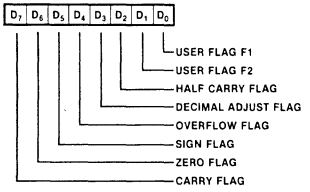
Figure 13. Control Registers

R248 P01M
Port 0 Mode Register
($F8_H$; Write Only)

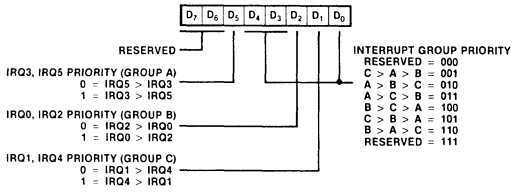


*ALWAYS EXTENDED TIMING AFTER RESET

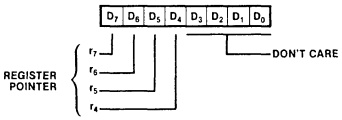
R252 FLAGS
Flag Register
(FC_H; Read/Write)



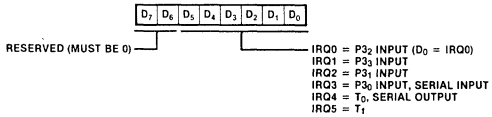
R249 IPR
Interrupt Priority Register
($F9_H$; Write Only)



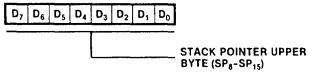
R253 RP
Register Pointer
(FD_H; Read/Write)



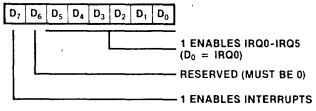
R250 IRQ
Interrupt Request Register
(FA_H; Read/Write)



R254 SPH
Stack Pointer
(FE_H; Read/Write)



R251 IMR
Interrupt Mask Register
(FB_H; Read/Write)



R255 SPL
Stack Pointer
(FF_H; Read/Write)

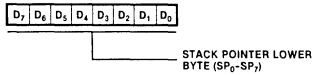
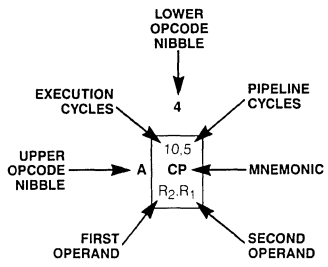


Figure 13. Control Registers (Continued)

OPCODE MAP

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6.5 **DEC** R_1	6.5 **DEC** IR_1	6.5 **ADD** r_1,r_2	6.5 **ADD** r_1,Ir_2	10.5 **ADD** R_2,R_1	10.5 **ADD** IR_2,R_1	10.5 **ADD** R_1,IM	10.5 **ADD** IR_1,IM	6.5 **LD** r_1,R_2	6.5 **LD** r_2,R_1	12/10.5 **DJNZ** r_1,RA	12/10.0 **JR** cc,RA	6.5 **LD** r_1,IM	12/10.0 **JP** cc,DA	6.5 **INC** r1	
1	6.5 **RLC** R_1	6.5 **RLC** IR_1	6.5 **ADC** r_1,r_2	6.5 **ADC** r_1,Ir_2	10.5 **ADC** R_2,R_1	10.5 **ADC** IR_2,R_1	10.5 **ADC** R_1,IM	10.5 **ADC** IR_1,IM								
2	6.5 **INC** R_1	6.5 **INC** IR_1	6.5 **SUB** r_1,r_2	6.5 **SUB** r_1,Ir_2	10.5 **SUB** R_2,R_1	10.5 **SUB** IR_2,R_1	10.5 **SUB** R_1,IM	10.5 **SUB** IR_1,IM								
3	8.0 **JP** IRR_1	6.1 **SRP** IM	6.5 **SBC** r_1,r_2	6.5 **SBC** r_1,Ir_2	10.5 **SBC** R_2,R_1	10.5 **SBC** IR_2,R_1	10.5 **SBC** R_1,IM	10.5 **SBC** IR_1,IM								
4	8.5 **DA** R_1	8.5 **DA** IR_1	6.5 **OR** r_1,r_2	6.5 **OR** r_1,Ir_2	10.5 **OR** R_2,R_1	10.5 **OR** IR_2,R_1	10.5 **OR** R_1,IM	10.5 **OR** IR_1,IM								
5	10.5 **POP** R_1	10.5 **POP** IR_1	6.5 **AND** r_1,r_2	6.5 **AND** r_1,Ir_2	10.5 **AND** R_2,R_1	10.5 **AND** IR_2,R_1	10.5 **AND** R_1,IM	10.5 **AND** IR_1,IM								
6	6.5 **COM** R_1	6.5 **COM** IR_1	6.5 **TCM** r_1,r_2	6.5 **TCM** r_1,Ir_2	10.5 **TCM** R_2,R_1	10.5 **TCM** IR_2,R_1	10.5 **TCM** R_1,IM	10.5 **TCM** IR_1,IM								
7	10/12.1 **PUSH** R_2	12/14.1 **PUSH** IR_2	6.5 **TM** r_1,r_2	6.5 **TM** r_1,Ir_2	10.5 **TM** R_2,R_1	10.5 **TM** IR_2,R_1	10.5 **TM** R_1,IM	10.5 **TM** IR_1,IM								
8	10.5 **DECW** RR_1	10.5 **DECW** IR_1	12.0 **LDE** r_1,Irr_2	18.0 **LDEI** Ir_1,Irr_2												6.1 **DI**
9	6.5 **RL** R_1	6.5 **RL** IR_1	12.0 **LDE** r_2,Irr_1	18.0 **LDEI** Ir_2,Irr_1												6.1 **EI**
A	10.5 **INCW** RR_1	10.5 **INCW** IR_1	6.5 **CP** r_1,r_2	6.5 **CP** r_1,Ir_2	10.5 **CP** R_2,R_1	10.5 **CP** IR_2,R_1	10.5 **CP** R_1,IM	10.5 **CP** IR_1,IM								14.0 **RET**
B	6.5 **CLR** R_1	6.5 **CLR** IR_1	6.5 **XOR** r_1,r_2	6.5 **XOR** r_1,Ir_2	10.5 **XOR** R_2,R_1	10.5 **XOR** IR_2,R_1	10.5 **XOR** R_1,IM	10.5 **XOR** IR_1,IM								16.0 **IRET**
C	6.5 **RRC** R_1	6.5 **RRC** IR_1	12.0 **LDC** r_1,Irr_2	18.0 **LDCI** Ir_1,Irr_2				10.5 **LD** r_1,x,R_2								6.5 **RCF**
D	6.5 **SRA** R_1	6.5 **SRA** IR_1	12.0 **LDC** r_2,Irr_1	18.0 **LDCI** Ir_2,Irr_1	20.0 **CALL*** IRR_1		20.0 **CALL** DA	10.5 **LD** r_2,x,R_1								6.5 **SCF**
E	6.5 **RR** R_1	6.5 **RR** IR_1		6.5 **LD** r_1,IR_2	10.5 **LD** R_2,R_1	10.5 **LD** IR_2,R_1	10.5 **LD** R_1,IM	10.5 **LD** IR_1,IM								6.5 **CCF**
F	8.5 **SWAP** R_1	8.5 **SWAP** IR_1		6.5 **LD** Ir_1,r_2		10.5 **LD** R_2,IR_1										6.0 **NOP**
Bytes per Instruction	2				3				2					3	1	



Legend:
R = 8-bit address
r = 4-bit address
R_1 or r_1 = Dst address
R_2 or r_2 = Src address

Sequence:
Opcode, First Operand Second Operand

NOTE The blank areas are not defined

*2-byte instruction; fetch cycle appears as a 3-byte instruction

ABSOLUTE MAXIMUM RATINGS

Voltages on all pins except RESET
with respect to GND . . . . . −0.3V to +7.0V
Operating Ambient
Temperature. . . . . .See Ordering Information
Storage Temperature . . . . . −65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

STANDARD TEST CONDITIONS

The DC characteristics listed below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin.

Standard conditions are as follows:

- $+4.75V \leqslant V_{CC} \leqslant +5.25V$
- GND = 0V
- $0°C \leqslant T_A \leqslant +70°C$ for S (Standard temperature)
- $-40°C \leqslant T_A \leqslant +100°C$ for E (Extended temperature)

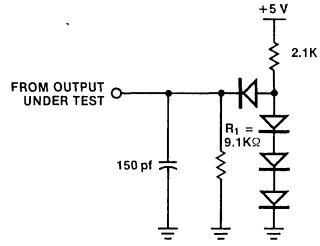


Figure 14. Test Load 1

DC CHARACTERISTICS

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	3.8	V_{CC}	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	−0.3	0.8	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{IL}	Input Low Voltage	−0.3	0.8	V	
V_{RH}	Reset Input High Voltage	3.8	V_{CC}	V	
V_{RL}	Reset Input Low Voltage	−0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\,\mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0$ mA
I_{IL}	Input Leakage	−10	10	μA	V_{IN} = 0V, 5.25V
I_{OL}	Output Leakage	−10	10	μA	V_{IN} = 0V, 5.25V
I_{IR}	Reset Input Current		−50	μA	$V_{CC} = +5.25V$, V_{RL} = 0V
I_{CC}	V_{CC} Supply Current		180	mA	All outputs and I/O pins floating

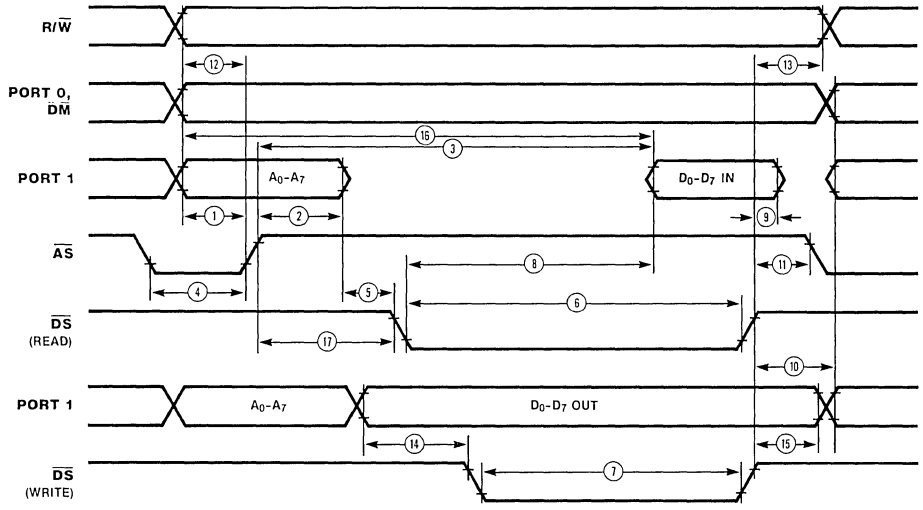


Figure 15. External I/O or Memory Read/Write Timing

AC CHARACTERISTICS

External I/O or Memory Read and Write Timing

			8 MHz		12 MHz		
Number	**Symbol**	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Notes*†°**
1	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	50		35		2,3
2	TdAS(A)	$\overline{AS}$ ↑ to Address Float Delay	70		45		2,3
3	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Required Valid		360		220	1,2,3
4	TwAS	$\overline{AS}$ Low Width	80		55		2,3
5	TdAz(DS)	Address Float to $\overline{DS}$ ↓	0		0		
6	TwDSR	$\overline{DS}$ (Read) Low Width	250		185		1,2,3
7	TwDSW	$\overline{DS}$ (Write) Low Width	160		110		1,2,3
8	TdDSR(DR)	$\overline{DS}$ ↓ to Read Data Required Valid		200		130	1,2,3
9	ThDR(DS)	Read Data to $\overline{DS}$ ↑ Hold Time	0		0		
10	TdDS(A)	$\overline{DS}$ ↑ to Address Active Delay	70		45		2,3
11	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	70		55		2,3
12	TdR/W(AS)	R/$\overline{W}$ Valid to $\overline{AS}$ ↑ Delay	50		30		2,3
13	TdDS(R/W)	$\overline{DS}$ ↑ to R/W Not Valid	60		35		2,3
14	TdDW(DSW)	Write Data Valid to $\overline{DS}$ (Write) ↓ Delay	50		35		2,3
15	TdDS(DW)	$\overline{DS}$ ↑ to Write Data Not Valid Delay	60		35		2,3
16	TdA(DR)	Address Valid to Read Data Required Valid		410		255	1,2,3
17	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	80		55		2,3

NOTES:
1. When using extended memory timing add 2 TpC.
2. Timing numbers given are for minimum TpC.
3. See clock cycle time dependent characteristics table.

* All units in nanoseconds (ns).
† Test Load 1
° All timing references use 2.0V for a logic "1" and 0.8V for a logic "0".

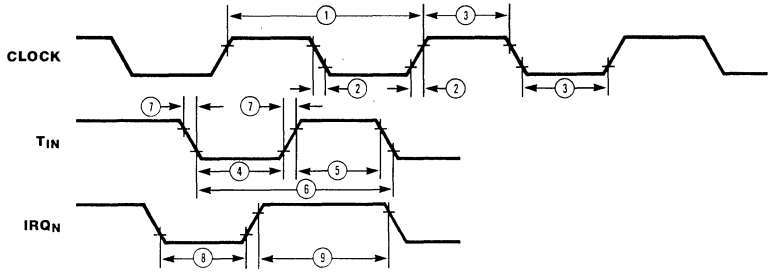


Figure 16. Additional Timing

AC CHARACTERISTICS

Additional Timing Table

Number	Symbol	Parameter	8 MHz Min	8 MHz Max	12 MHz Min	12 MHz Max	Notes*
1	TpC	Input Clock Period	125	1000	83	1000	1
2	TrC,TfC	Clock Input Rise and Fall Times		25		15	1
3	TwC	Input Clock Width	37		70		1
4	TwTinL	Timer Input Low Width	100		70		2
5	TwTinH	Timer Input High Width	3TpC		3TpC		2
6	TpTin	Timer Input Period	8TpC		8TpC		2
7	TrTin,TfTin	Timer Input Rise and Fall Times		100		100	2
8A	TwIL	Interrupt Request Input Low Time	100		70		2,4
8B	TwIL	Interrupt Request Input Low Time	3TpC		3TpC		2,5
9	TwIH	Interrupt Request Input High Time	3TpC		3TpC		2,3

NOTES:
1. Clock timing references use 3.8V for a logic "1" and 0.8V for a logic "0".
2. Timing references use 2.0V for a logic "1" and 0.8V for a logic "0".
3. Interrupt request via Port 3.
4. Interrupt request via Port 3 ($P3_1$-$P3_3$)
5. Interrupt request via Port 3 ($P3_0$)
* Units in nanoseconds (ns).

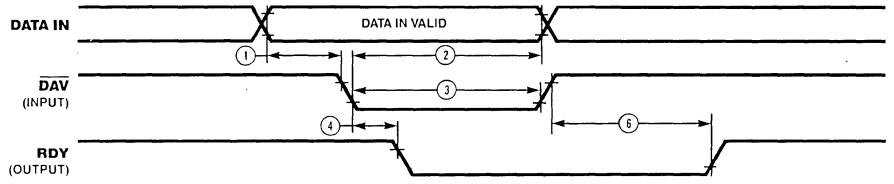


Figure 17a. Input Handshake Timing

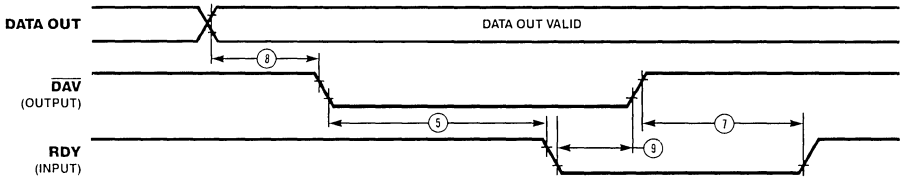


Figure 17b. Output Handshake Timing

AC CHARACTERISTICS

Handshake Timing

Number	Symbol	Parameter	8 MHz Min	8 MHz Max	12 MHz Min	12 MHz Max	Notes†*
1	TsDI(DAV)	Data In Setup Time	0		0		
2	ThDI(DAV)	Data In Hold Time	230		160		
3	TwDAV	Data Available Width	175		120		
4	TdDAVIf(RDY)	$\overline{DAV}$ ↓ Input to RDY ↓ Delay		175		120	1,2
5	TdDAVOf(RDY)	$\overline{DAV}$ ↓ Output to RDY ↓ Delay	0		0		1,3
6	TdDAVIr(RDY)	$\overline{DAV}$ ↑ Input to RDY ↑ Delay		175		120	1,2
7	TdDAVOr(RDY)	$\overline{DAV}$ ↑ Output to RDY ↑ Delay	0		0		1,3
8	TdDO(DAV)	Data Out to $\overline{DAV}$ ↓ Delay	50		30		1
9	TdRDY(DAV)	Rdy ↓ Input to $\overline{DAV}$ ↑ Delay	0	200	0	140	1

NOTES:
1. Test load 1
2. Input handshake
3. Output handshake

† All timing references use 2.0V for a logic "1" and 0.8V for a logic "0".

* Units in nanoseconds (ns).

CLOCK CYCLE TIME-DEPENDENT CHARACTERISTICS

Number	Symbol	8 MHz Equation	12 MHz Equation
1	TdA(AS)	TpC-75	TpC-50
2	TdAS(A)	TpC-55	TpC-40
3	TdAS(DR)	4TpC-140*	4TpC-110*
4	TwAS	TpC-45	TpC-30
6	TwDSR	3TpC-125*	3TpC-65*
7	TwDSW	2TpC-90*	2TpC-55*
8	TdDSR(DR)	3TpC-175*	3TpC-120*
10	Td(DS)A	TpC-55	TpC-40
11	TdDS(AS)	TpC-55	TpC-30
12	TdR/W(AS)	TpC-75	TpC-55
13	TdDS(R/W)	TpC-65	TpC-50
14	TdDW(DSW)	TpC-75	TpC-50
15	TdDS(DW)	TpC-55	TpC-40
16	TdA(DR)	5TpC-215*	5TpC-160*
17	TdAS(DS)	TpC-45	TpC-30

*Add 2TpC when using extended memory timing

Super8™ MCU ROMless, ROM, and Prototyping Device with EPROM Interface

Z8800, Z8801, Z8820, Z8822

FEATURES

- Improved Z8® instruction set includes multiply and divide instructions, Boolean and BCD operations.
- Additional instructions support threaded-code languages, such as "Forth."
- 325 byte registers, including 272 general-purpose registers, and 53 mode and control registers.
- Addressing of up to 128K bytes of memory.
- Two register pointers allow use of short and fast instructions to access register groups within 600 nsec.
- Direct Memory Access controller (DMA).
- Two 16-bit counter/timers.
- Up to 32 bit-programmable and 8 byte-programmable I/O lines, with 2 handshake channels.
- Interrupt structure supports:
  - 27 interrupt sources
  - 16 interrupt vectors (2 reserved for future versions)
  - 8 interrupt levels
  - Servicing in 600 nsec. (1 level only)
- Full-duplex UART with special features.
- On-chip oscillator.
- 20 MHz clock.
- 8K byte ROM for Z8820

GENERAL DESCRIPTION

The Zilog Super8 single-chip MCU can be used for development and production. It can be used as I/O- or memory-intensive computers, or configured to address external memory while still supporting many I/O lines.

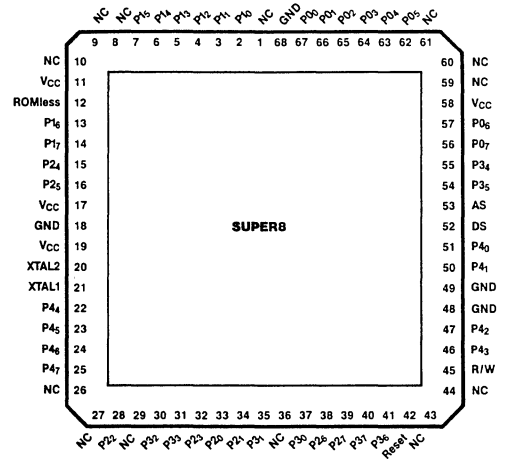
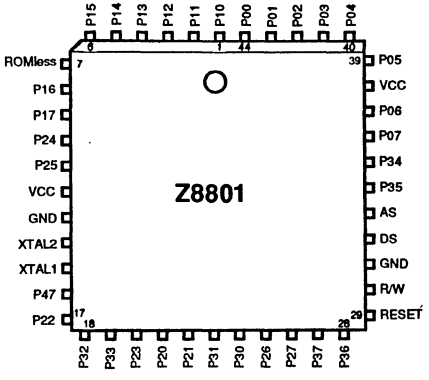


Figure 1a. Pin Assignments — 68-pin PLCC

The Super8 features a full-duplex universal asynchronous receiver/transmitter (UART) with on-chip baud rate generator, two programmable counter/timers, a direct memory access (DMA) controller, and an on-chip oscillator.

The Super8 is also available as a 48-pin and 68-pin ROMless microcomputer with four byte-wide I/O ports plus a byte-wide address/data bus. Additional address bits can be configured, up to a total of 16.



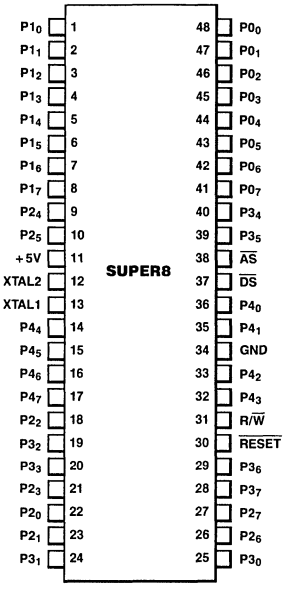


Figure 1b. Pin Assignments — 48-pin DIP

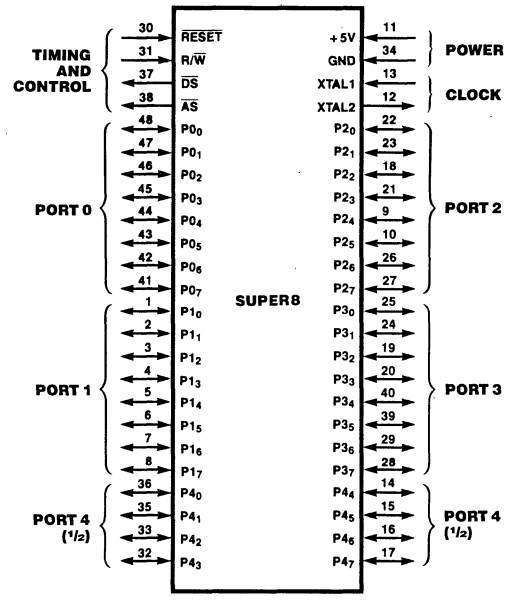


Figure 2. Pin Functions

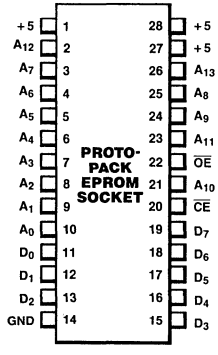


Figure 3. Pin Assignments—28-Pin Piggyback Socket

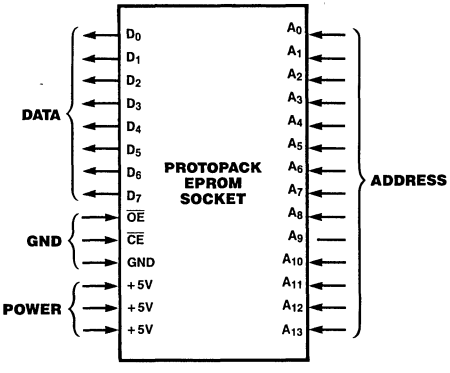


Figure 4. Pin Functions—28-Pin Piggyback Socket

Protopack

This part functions as an emulator for the basic microcomputer. It uses the same package and pin-out as the basic microcomputer but also has a 28-pin "piggy back" socket on the top into which a ROM or EPROM can be installed. The socket is designed to accept a type 2764 EPROM.

This package permits the protopack to be used in prototype and final PC boards while still permitting user program development. When a final program is developed, it can be mask-programmed into the production microcomputer device, directly replacing the emulator. The protopack part is also useful in situations where the cost of mask-programming is prohibitive or where program flexibility is desired.

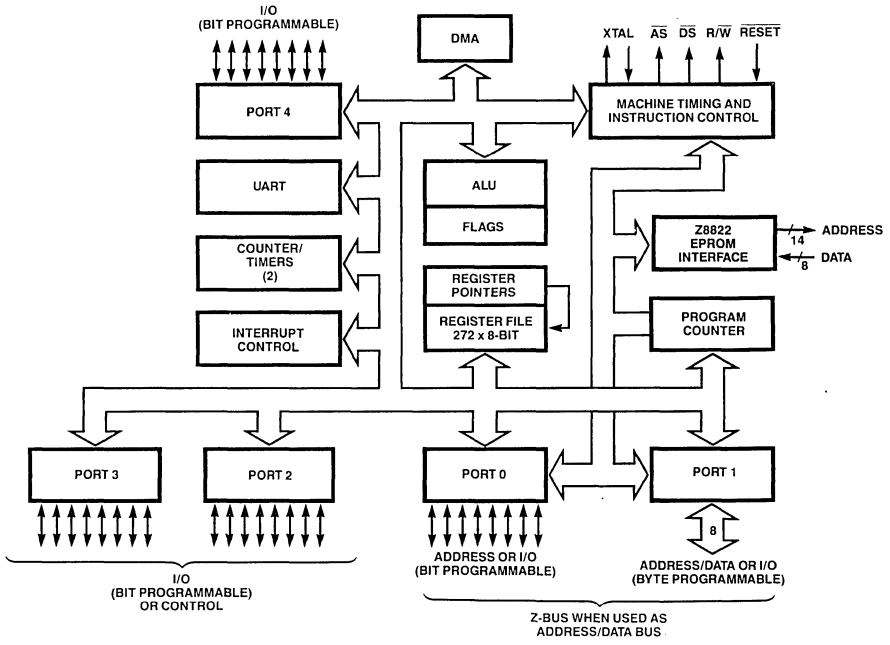


Figure 5. Functional Block Diagram

ARCHITECTURE

The Super8 architecture includes 325 byte-wide internal registers. 272 of these are available for general purpose use; the remaining 53 provide control and mode functions.

The instruction set is specially designed to deal with this large register set. It includes a full complement of 8-bit arithmetic and logical operations, including multiply and divide instructions and provisions for BCD operations. Addresses and counters can be incremented and decremented as 16-bit quantities. Rotate, shift, and bit manipulation instructions are provided. Three new instructions support threaded-code languages.

The UART is a full-function multipurpose asynchronous serial channel with many premium features.

The 16-bit counters can operate independently or be cascaded to perform 32-bit counting and timing operations. The DMA controller handles transfers to and from the register file or memory. DMA can use the UART or one of two ports with handshake capability.

The architecture appears in the block diagram (Figure 5).

PIN DESCRIPTIONS

The Super8 connects to external devices via the following TTL-compatible pins:

$\overline{AS}$. *Address Strobe* (output, active Low). $\overline{AS}$ is pulsed Low once at the beginning of each machine cycle. The rising edge indicates that addresses R/$\overline{W}$ and $\overline{DM}$, when used, are valid.

$\overline{DS}$. *Data Strobe* (output, active Low). $\overline{DS}$ provides timing for data movement between the address/data bus and external memory. During write cycles, data output is valid at the leading edge of $\overline{DS}$. During read cycles, data input must be valid prior to the trailing edge of $\overline{DS}$.

$P0_0$-$P0_7$, $P1_0$-$P1_7$, $P2_0$-$P2_7$, $P3_0$-$P3_7$, $P4_0$-$P4_7$. *Port I/O Lines* (input/output). These 40 lines are divided into five 8-bit I/O ports that can be configured under program control for I/O or external memory interface.

In the ROMless devices, Port 1 is dedicated as a multiplexed address/data port, and Port 0 pins can be assigned as additional address lines; Port 0 non-address pins may be assigned as I/O. In the ROM and protopack, Port 1 can be assigned as input or output, and Port 0 can be assigned as input or output on a bit by bit basis.

Ports 2 and 3 can be assigned on a bit-for-bit basis as general I/O or interrupt lines. They can also be used as special-purpose I/O lines to support the UART, counter/timers, or handshake channels.

Port 4 is used for general I/O.

During reset, all port pins are configured as inputs (high impedance) except for Port 1 and Port 0 in the ROMless devices. In these, Port 1 is configured as a multiplexed address/data bus, and Port 0 pins $P0_0$-$P0_4$ are configured as address out, while pins $P0_5$-$P0_7$ are configured as inputs.

$\overline{\text{RESET}}$. *Reset* (input, active Low). Reset initializes and starts the Super8. When it is activated, it halts all processing; when it is deactivated, the Super8 begins processing at address 0020_H.

ROMless. (input, active High). This input controls the operation mode of a 68-pin Super8. When connected to V_{CC}, the part will function as a ROMless Z8800. When connected to GND, the part will function as a Z8820 ROM part.

R/$\overline{\text{W}}$. *Read/Write* (output). R/$\overline{\text{W}}$ determines the direction of data transfer for external memory transactions. It is Low when writing to program memory or data memory, and High for everything else.

XTAL1, XTAL2. (Crystal oscillator input.) These pins connect a parallel resonant crystal or an external clock source to the on-board clock oscillator and buffer.

REGISTERS

The Super8 contains a 256-byte internal register space. However, by using the upper 64 bytes of the register space more than once, a total of 325 registers are available.

Registers from 00 to BF are used only once. They can be accessed by any register command. Register addresses C0 to FF contain two separate sets of 64 registers. One set, called control registers, can only be accessed by register direct commands. The other set can only be addressed by register indirect, indexed, stack, and DMA commands.

The uppermost 32 register direct registers (E0 to FF) are further divided into two banks (0 and 1), selected by the Bank Select bit in the Flag register. When a Register Direct command accesses a register between E0 and FF, it looks at the Bank Select bit in the Flag register to select one of the banks.

The register space is shown in Figure 6.

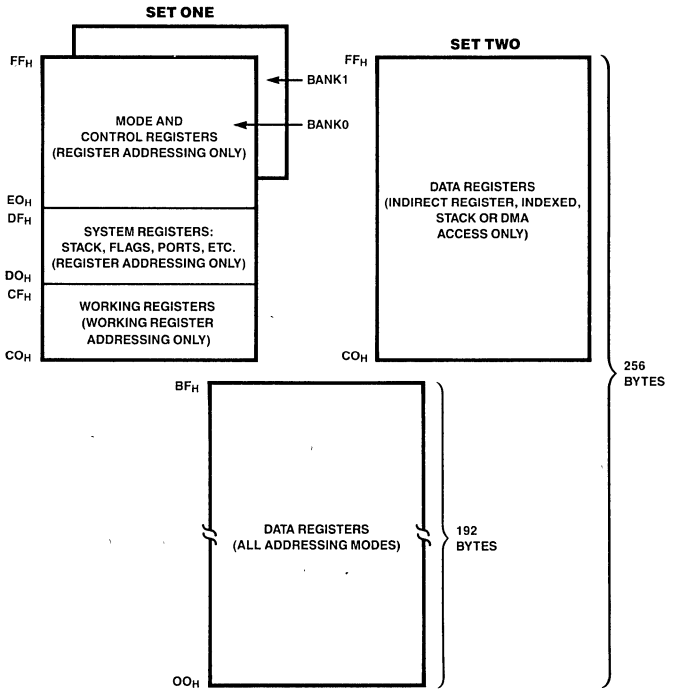


Figure 6. Super8 Registers

Working Register Window

Control registers R214 and R215 are the register pointers, RP0 and RP1. They each define a moveable, 8-register section of the register space. The registers within these spaces are called working registers.

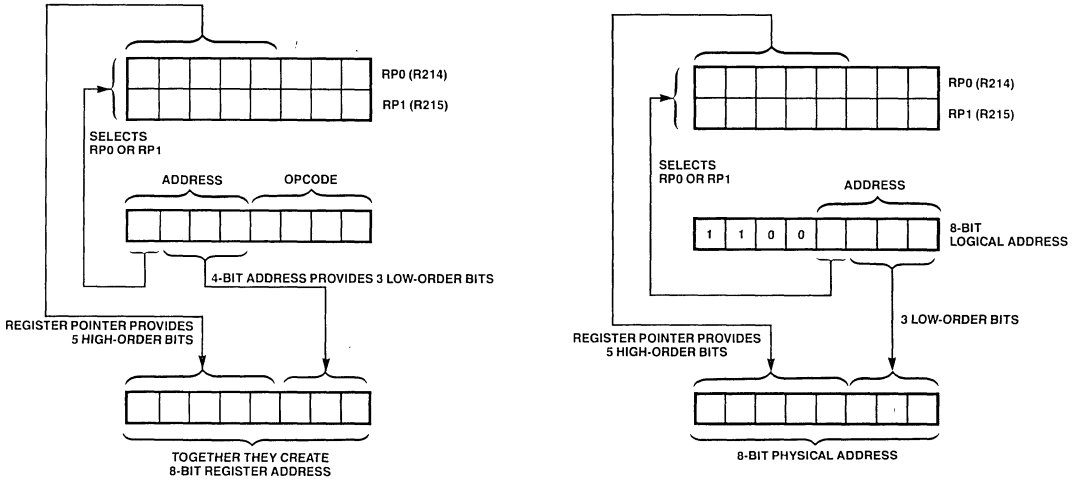
Working registers can be accessed using short 4-bit addresses. The process, shown in section a of Figure 4, works as follows:

- The high-order bit of the 4-bit address selects one of the two register pointers (0 selects RP0; 1 selects RP1).
- The five high-order bits in the register pointer select an 8-register (contiguous) slice of the register space.
- The three low-order bits of the 4-bit address select one of the eight registers in the slice.

The net effect is to concatenate the five bits from the register pointer to the three bits from the address to form an 8-bit address. As long as the address in the register pointer remains unchanged, the three bits from the address will always point to an address within the same eight registers.

The register pointers can be moved by changing the five high bits in control registers R214 for RP0 and R215 for RP1.

The working registers can also be accessed by using full 8-bit addressing. When an 8-bit logical address in the range 192 to 207 (C0 to CF) is specified, the lower nibble is used similarly to the 4-bit addressing described above. This is shown in section b of Figure 7.



a. 4-Bit Addressing

b. 8-Bit Addressing

Figure 7. Working Register Window

Since any direct access to logical addresses 192 to 207 involves the register pointers, the physical registers 192 to 207 can be accessed only when selected by a register pointer. After a reset, RP0 points to R192 and RP1 points to R200.

Register List

Table 1 lists the Super8 registers. For more details, see Figure 8.

Table 1. Super-8 Registers

Address Decimal	Hexadecimal		Mnemonic	Function
General-Purpose Registers				
000-192	00-BF		—	General purpose (all address modes)
192-207	C0-CF		—	Working register (direct only)
192-255	C0-FF		—	General purpose (indirect only)
Mode and Control Registers				
208	D0		P0	Port 0 I/O bits
209	D1		P1	Port 1 (I/O only)
210	D2		P2	Port 2
211	D3		P3	Port 3
212	D4		P4	Port 4
213	D5		FLAGS	System Flags Register
214	D6		RP0	Register Pointer 0
215	D7		RP1	Register Pointer 1
216	D8		SPH	Stack Pointer High Byte
217	D9		SPL	Stack Pointer Low Byte
218	DA		IPH	Instruction Pointer High Byte
219	DB		IPL	Instruction Pointer Low Byte
220	DC		IRQ	Interrupt Request
221	DD		IMR	Interrupt Mask Register
222	DE		SYM	System Mode
224	E0	Bank 0	C0CT	CTR 0 Control
		Bank 1	COM	CTR 0 Mode
225	E1	Bank 0	C1CT	CTR 1 Control
		Bank 1	C1M	CTR 1 Mode
226	E2	Bank 0	C0CH	CTR 0 Capture Register, bits 8-15
		Bank 1	CTCH	CTR 0 Timer Constant, bits 8-15
227	E3	Bank 0	C0CL	CTR 0 Capture Register, bits 0-7
		Bank 1	CTCL	CTR 0 Time Constant, bits 0-7
228	E4	Bank 0	C1CH	CTR 1 Capture Register, bits 8-15
		Bank 1	C1TCH	CTR 1 Time Constant, bits 8-15
229	E5	Bank 0	C1CL	CTR 1 Capture Register, bits 0-7
		Bank 1	C1TCL	CTR 1 Time Constant, bits 0-7
235	EB	Bank 0	UTC	UART Transmit Control
236	EC	Bank 0	URC	UART Receive Control
237	ED	Bank 0	UIE	UART Interrupt Enable
239	EF	Bank 0	UIO	UART Data
240	F0	Bank 0	P0M	Port 0 Mode
		Bank 1	DCH	DMA Count, bits 8-15
241	F1	Bank 0	PM	Port Mode Register
		Bank 1	DCL	DMA Count, bits 0-7
244	F4	Bank 0	H0C	Handshake Channel 0 Control
245	F5	Bank 0	H1C	Handshake Channel 1 Control
246	F6	Bank 0	P4D	Port 4 Direction
247	F7	Bank 0	P4OD	Port 4 Open Drain
248	F8	Bank 0	P2AM	Port 2/3 A Mode
		Bank 1	UBGH	UART Baud Rate Generator, bits 8-15

Table 1. Super-8 Registers (Continued)

Address Decimal	Hexadecimal		Mnemonic	Function
Mode and Control Registers (Continued)				
249	F9	Bank 0	P2BM	Port 2/3 B Mode
		Bank 1	UBGL	UART Baud Rate Generator, bits 0-7
250	FA	Bank 0	P2CM	Port 2/3 C Mode
		Bank 1	UMA	UART Mode A
251	FB	Bank 0	P2DM	Port 2/3 D Mode
		Bank 1	UMB	UART Mode B
252	FC	Bank 0	P2AIP	Port 2/3 A Interrupt Pending
253	FD	Bank 0	P2BIP	Port 2/3 B Interrupt Pending
254	FE	Bank 0	EMT	External Memory Timing
		Bank 1	WUMCH	Wakeup Match Register
255	FF	Bank 0	IPR	Interrupt Priority Register
		Bank 1	WUMSK	Wakeup Mask Register

MODE AND CONTROL REGISTERS

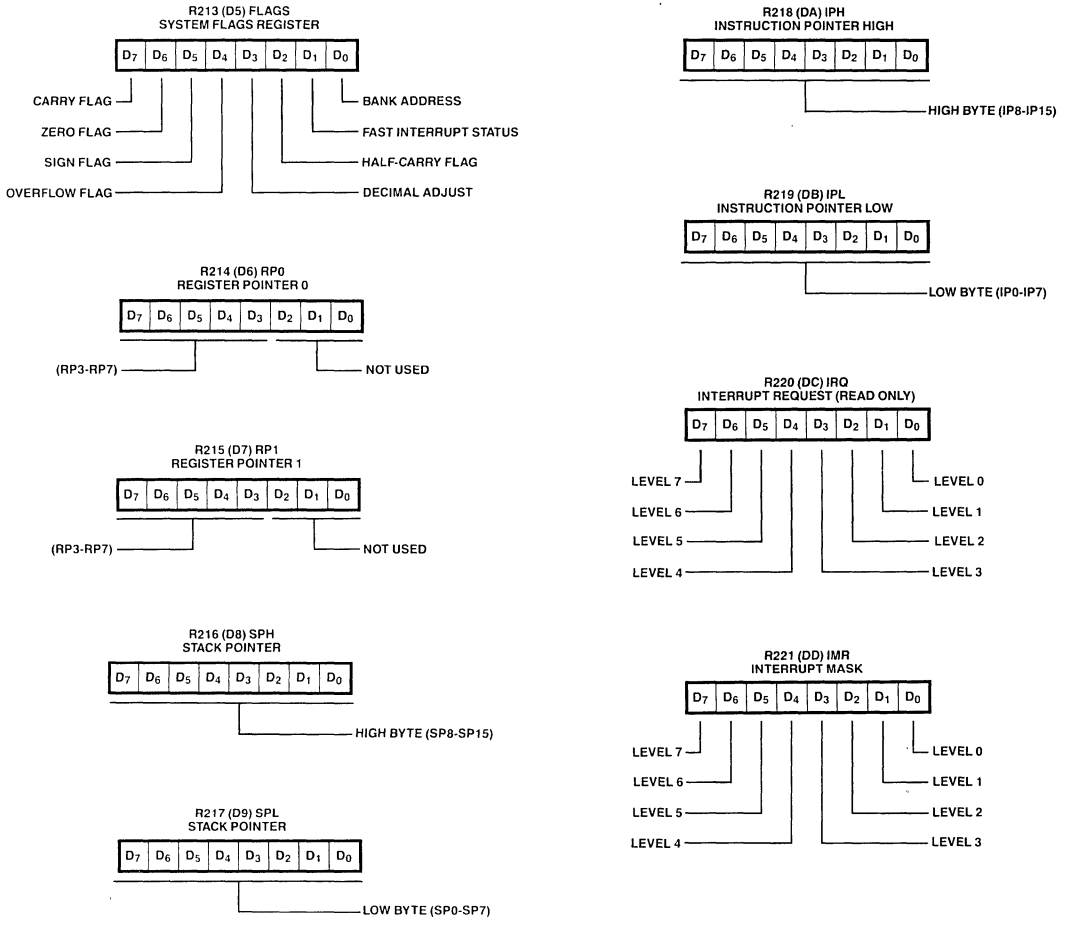
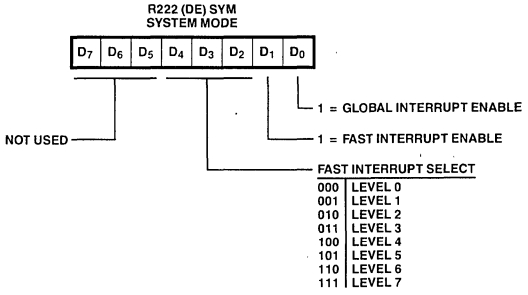
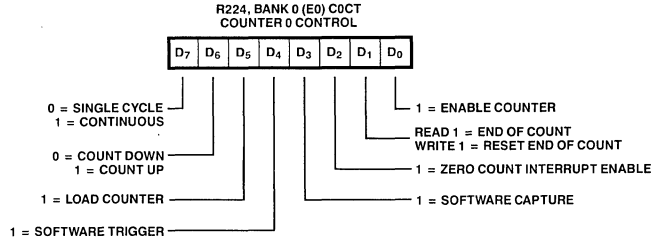
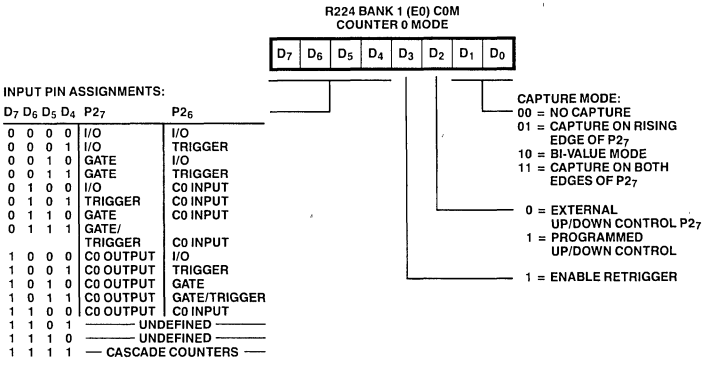


Figure 8. Mode and Control Registers







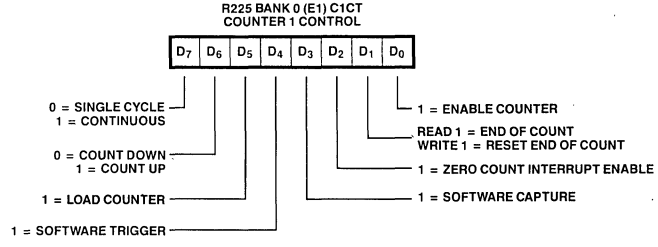


Figure 8. Mode and Control Registers (Continued)

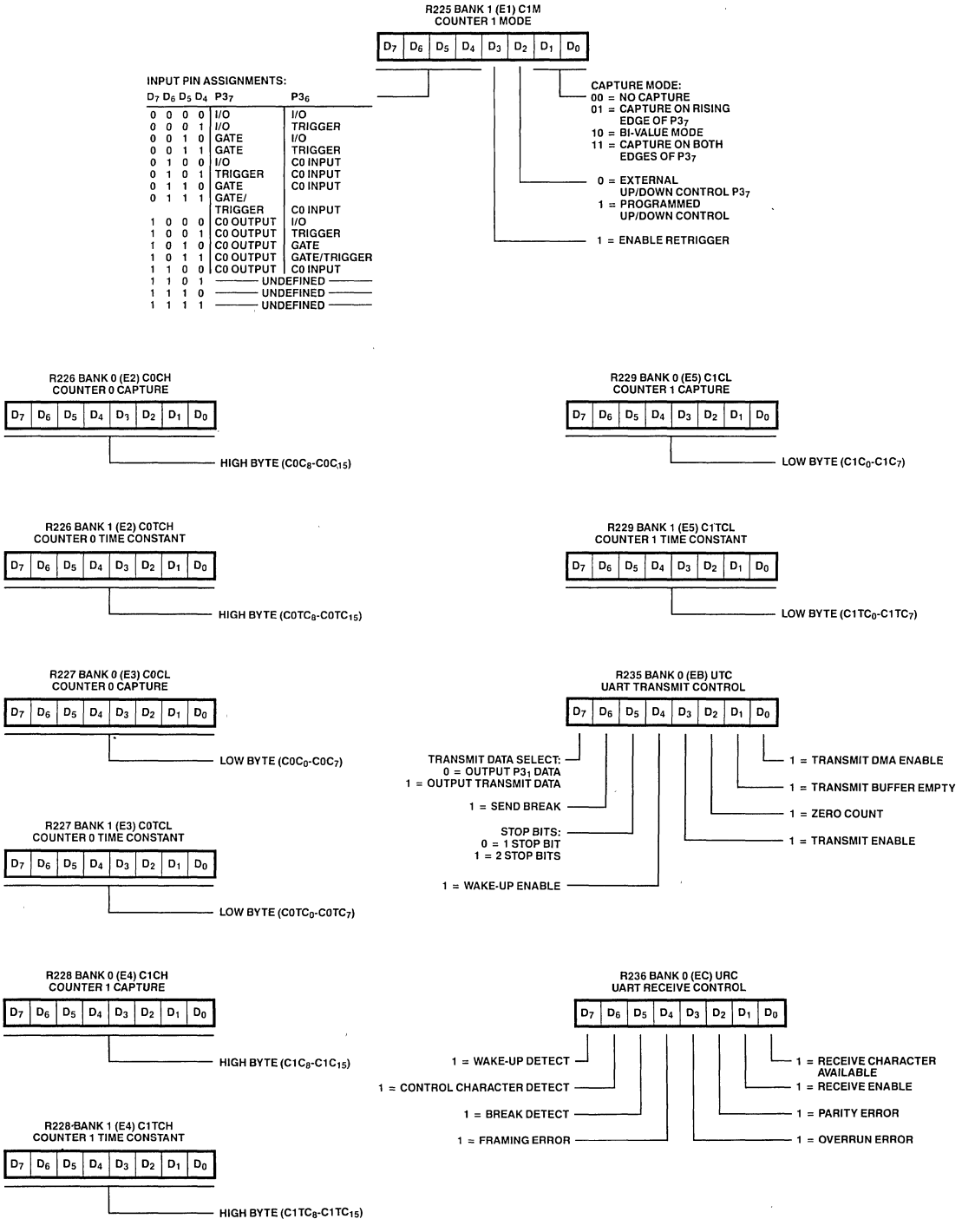


Figure 8. Mode and Control Registers (Continued)

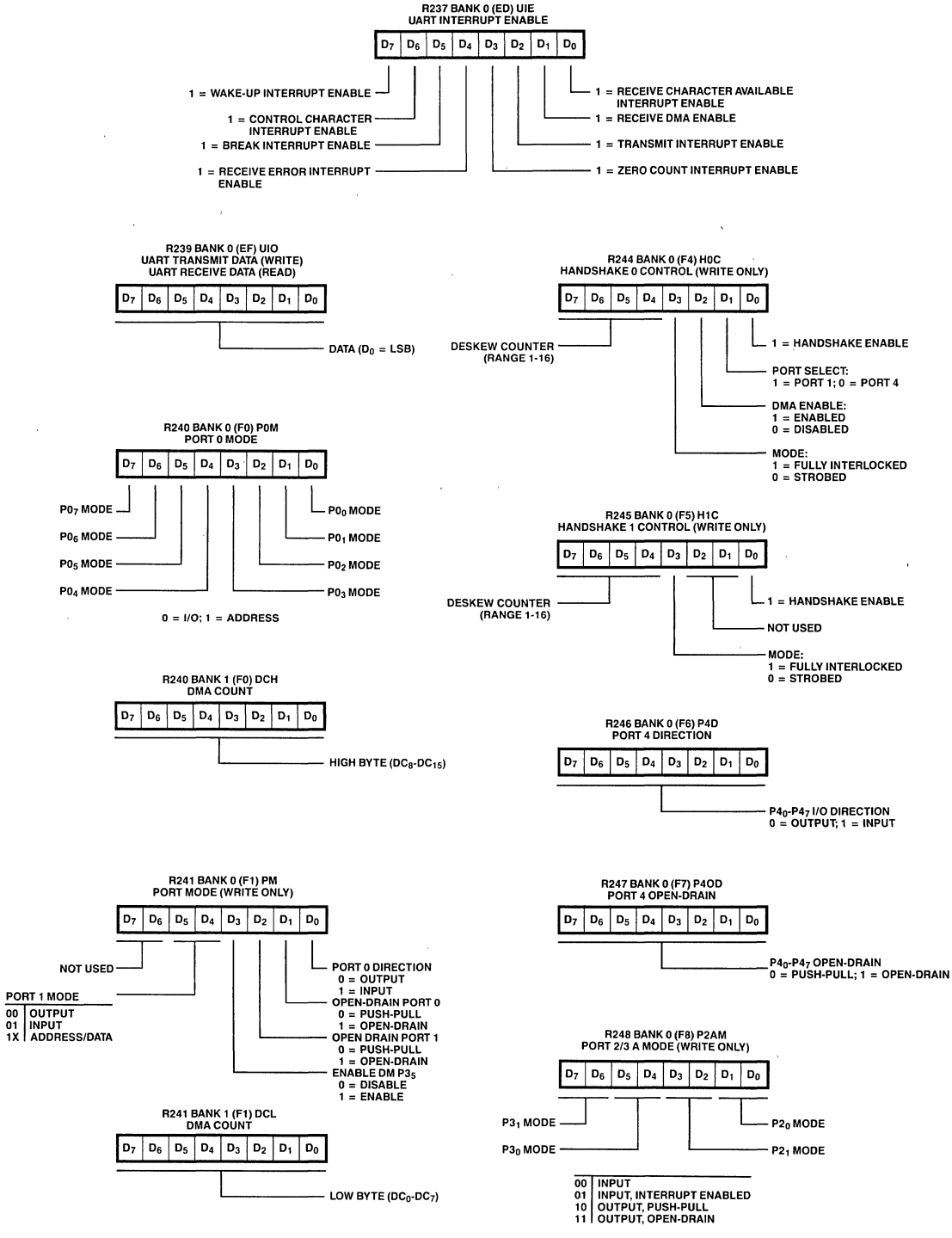


Figure 8. Mode and Control Registers (Continued)

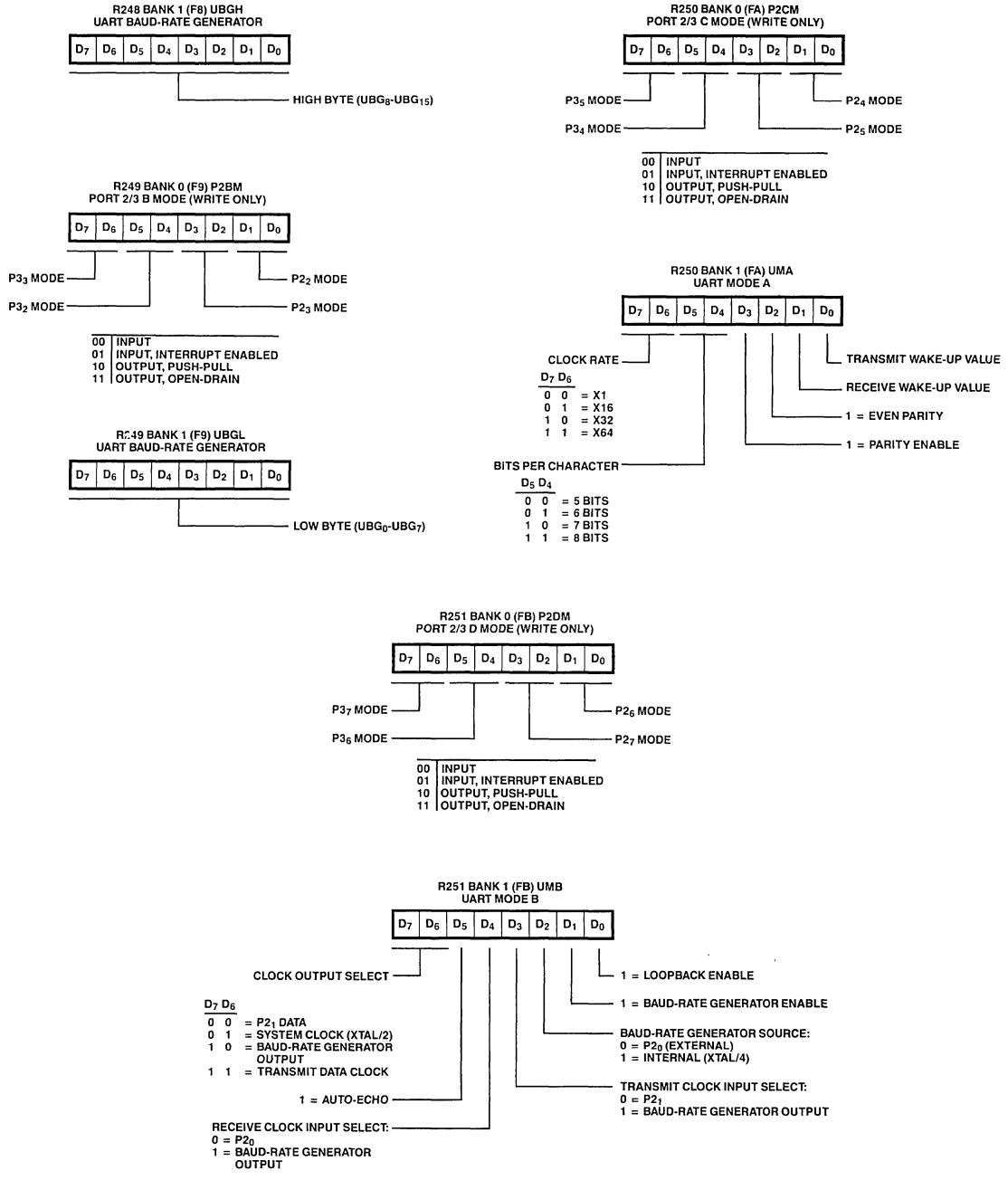


Figure 8. Mode and Control Registers (Continued)

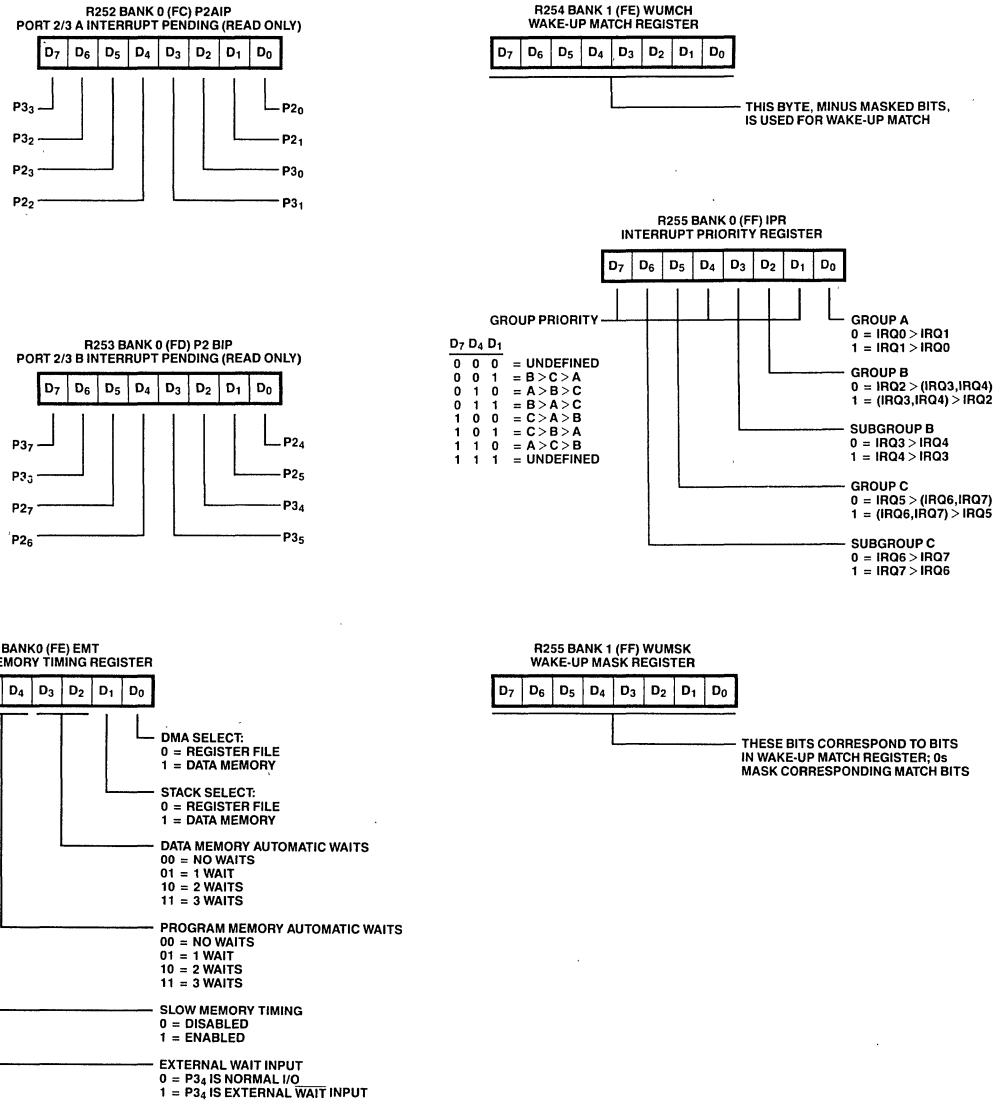


Figure 8. Mode and Control Registers (Continued)

I/O PORTS

The Super8 has 40 I/O lines arranged into five 8-bit ports. These lines are all TTL-compatible, and can be configured as inputs or outputs. Some can also be configured as address/data lines.

Each port has an input register, an output register, and a register address. Data coming into the port is stored in the input register, and data to be written to a port is stored in the output register. Reading a port's register address returns the value in the input register; writing a port's register address loads the value in the output register. If the port is configured for an output, this value will appear on the external pins.

When the CPU reads the bits configured as outputs, the data on the external pins is returned. Under normal output loading, this has the same effect as reading the output register, unless the bits are configured as open-drain outputs.

The ports can be configured as shown in Table 2.

Table 2. Port Configuration

Port	Configuration Choices
0	Address outputs and/or general I/O
1	Multiplexed address/data(or I/O, only for ROM and Protopack)
2 and 3	Control I/0 for UART, handshake channels, and counter/timers; also general I/O and external interrupts
4	General I/O

Port 0

Port 0 can be configured as an I/O port or an output for addressing external memory, or it can be divided and used as both. The bits configured as I/O can be either all outputs or all inputs; they cannot be mixed. If configured for outputs, they can be push-pull or open-drain type.

Any bits configured for I/O can be accessed via R208. To write to the port, specify R208 as the destination (dst) of an instruction; to read the port, specify R208 as the source (src).

Port 0 bits configured as I/O can be placed under handshake control of handshake channel 1.

Port 0 bits configured as address outputs cannot be accessed via the register.

In ROMless devices, initially the four lower bits are configured as address eight through twelve.

Port 1

In the ROMless device, Port 1 is configured as a byte-wide address/data port. It provides a byte-wide multiplexed address/data path. Additional address lines can be added by configuring Port 0.

The ROM and Protopack Port 1 can be configured as above or as an I/O port; it can be a byte-wide input, open-drain output, or push-pull output. It can be placed under handshake control or handshake channel 0.

Ports 2 and 3

Ports 2 and 3 provide external control inputs and outputs for the UART, handshake channels, and counter/timers. The pin assignments appear in Table 3.

Bits not used for control I/O can be configured as general-purpose I/O lines and/or external interrupt inputs.

Those bits configured for general I/O can be configured individually for input or output. Those configured for output can be individually configured for open-drain or push-pull output.

All Port 2 and 3 input pins are Schmitt-triggered.

The port address for Port 2 is R210, and for Port 3 is R211.

Table 3. Pin Assignments for Ports 2 and 3

Port 2		Port 3	
Bit	**Function**	**Bit**	**Function**
0	UART receive clock	0	UART receive data
1	UART transmit clock	1	UART transmit data
2	Reserved	2	Reserved
3	Reserved	3	Reserved
4	Handshake 0 input	4	Handshake 1 input/$\overline{\text{WAIT}}$
5	Handshake 0 output	5	Handshake 1 output/$\overline{\text{DM}}$
6	Counter 0 input	6	Counter 1 input
7	Counter 0 I/O	7	Counter 1 I/O

Port 4

Port 4 can be configured as I/O only. Each bit can be configured individually as input or output, with either push-pull or open-drain outputs. All Port 4 inputs are Schmitt-triggered.

Port 4 can be placed under handshake control of handshake channel 0. Its register address is R212.

UART

The UART is a full-duplex asynchronous channel. It transmits and receives independently with 5 to 8 bits per character, has options for even or odd bit parity, and a wake-up feature.

Data can be read into or out of the UART via R239, Bank 0. This single address is able to serve a full-duplex channel because it contains two complete 8-bit registers—one for the transmitter and the other for the receiver.

Pins

The UART uses the following Port 2 and 3 pins:

Port/Pin	UART Function
2/0	Receive Clock
3/0	Receive Data
2/1	Transmit Clock
3/1	Transmit Data

Transmitter

When the UART's register address is specified as the destination (dst) of an operation, the data is output on the UART, which automatically adds the start bit, the programmed parity bit, and the programmed number of stop bits. It can also add a wake-up bit if that option is selected.

If the UART is programmed for a 5-, 6-, or 7-bit character, the extra bits in R239 are ignored.

Serial data is transmitted at a rate equal to 1, 1/16, 1/32 or 1/64 of the transmitter clock rate, depending on the programmed data rate. All data is sent out on the falling edge of the clock input.

When the UART has no data to send, it holds the output marking (High). It may be programmed with the Send Break command to hold the output Low (Spacing), which it continues until the command is cleared.

Receiver

The UART begins receive operation when Receive Enable (URC, bit 0) is set High. After this, a Low on the receive input pin for longer than half a bit time is interpreted as a start bit. The UART samples the data on the input pin in the middle of each clock cycle until a complete byte is assembled. This is placed in the Receive Data register.

If the 1X clock mode is selected, external bit synchronization must be provided, and the input data is sampled on the rising edge of the clock.

For character lengths of less than eight bits, the UART inserts ones into the unused bits, and, if parity is enabled, the parity bit is not stripped. The data bits, extra ones, and the parity bit are placed in the UART Data register (UIO).

While the UART is assembling a byte in its input shift register, the CPU has time to service an interrupt and manipulate the data character in UIO.

Once a complete character is assembled, the UART checks it and performs the following:

- If it is an ASCII control character, the UART sets the Control Character status bit.
- It checks the wake-up settings and completes any indicated action.
- If parity is enabled, the UART checks to see if the calculated parity matches the programmed parity bit. If they do not match, it sets the Parity Error bit in URC (R236 Bank 0), which remains set until reset by software.
- It sets the Framing Error bit (URC, bit 4) if the character is assembled without any stop bits. This bit remains set until cleared by software.

Overrun errors occur when characters are received faster than they are read. That is, when the UART has assembled a complete character before the CPU has read the current character, the UART sets the Overrun Error bit (URC, bit 3), and the character currently in the receive buffer is lost.

The overrun bit remains set until cleared by software.

ADDRESS SPACE

The Super8 can access 64K bytes of program memory and 64K bytes of data memory. These spaces can be either combined or separate. If separate, they are controlled by the $\overline{DM}$ line (Port $P3_5$), which selects data memory when Low and program memory when High.

Figure 9 shows the system memory space.

CPU Program Memory

Program memory occupies addresses 0 to 64K. External program memory, if present, is accessed by configuring Ports 0 and 1 as a memory interface.

The address/data lines are controlled by $\overline{AS}$, $\overline{DS}$ and R/$\overline{W}$.

The first 32 program memory bytes are reserved for interrupt vectors; the lowest address available for user programs is 32 (decimal). This value is automatically loaded into the program counter after a hardware reset.

ROMless

Port 0 can be configured to provide from 0 to 8 additional address lines. Port 1 is always used as an 8-bit multiplexed address/data port.

ROM and Protopack

Port 1 is configured as multiplexed address/data or as I/O. When Port 1 is configured as address/data, Port 0 lines can be used as additional address lines, up to address 15. External program memory is mapped above internal program memory; that is, external program memory can occupy any space beginning at the top of the internal ROM space up to the 64K (16-bit address) limit.

CPU Data Memory

The external CPU data memory space, if separated from program memory by the $\overline{DM}$ optional output, can be mapped anywhere from 0 to 64K (full 16-bit address space). Data memory uses the same address/data bus (Port 1) and additional addresses (chosen from Port 0) as program memory. Data memory is distinguished from program memory by the DM pin ($P3_5$), and by the fact that data memory can begin at address 0000_H. This feature differs from the Z8.

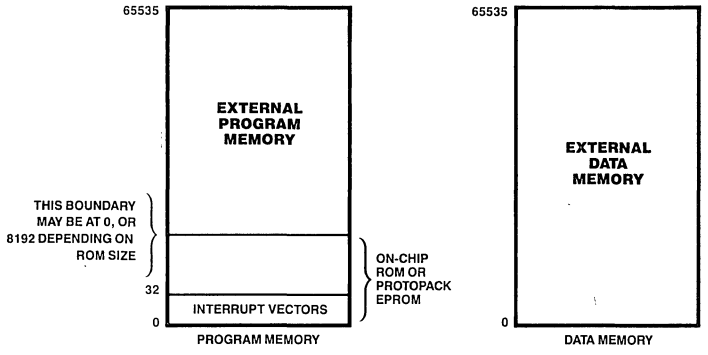


Figure 9. Program and Data Memory Address Spaces

INSTRUCTION SET

The Super8 instruction set is designed to handle its large register set. The instruction set provides a full complement of 8-bit arithmetic and logical operations, including multiply and divide. It supports BCD operations using a decimal adjustment of binary values, and it supports incrementing and decrementing 16-bit quantities for addresses and counters.

It provides extensive bit manipulation, and rotate and shift operations, and it requires no special I/O instructions—the I/O ports are mapped into the register file.

Instruction Pointer

A special register called the Instruction Pointer (IP) provides hardware support for threaded-code languages. It consists of register-pair R218 and R219, and it contains memory addresses. The MSB is R218.

Threaded-code languages deal with an imaginary higher-level machine within the existing hardware machine. The IP acts like the PC for that machine. The command NEXT passes control to or from the hardware machine to the imaginary machine, and the commands ENTER and EXIT are imaginary machine equivalents of (real machine) CALLS and RETURNS.

If the commands NEXT, ENTER, and EXIT are not used, the IP can be used by the fast interrupt processing, as described in the Interrupts section.

Flag Register

The Flag register (FLAGS) contains eight bits that describe the current status of the Super8. Four of these can be tested and used with conditional jump instructions; two others are used for BCD-arithmetic. FLAGS also contains the Bank Address bit and the Fast Interrupt Status bit.

The flag bits can be set and reset by instructions.

CAUTION

Do not specify FLAGS as the destination of an instruction that normally affects the flag bits or the result will be unspecified.

The following paragraphs describe each flag bit:

Bank Address. This bit is used to select one of the register banks (0 or 1) between (decimal) addresses 224 and 255. It is cleared by the SB0 instruction and set by the SB1 instruction.

Fast Interrupt Status. This bit is set during a fast interrupt cycle and reset during the IRET following interrupt servicing. When set, this bit inhibits all interrupts and causes the fast interrupt return to be executed when the IRET instruction is fetched.

Half-Carry. This bit is set to 1 whenever an addition generates a carry out of bit 3, or when a subtraction borrows out of bit 4. This bit is used by the Decimal Adjust (DA) instruction to convert the binary result of a previous addition or subtraction into the correct decimal (BCD) result. This flag, and the Decimal Adjust flag, are not usually accessed by users.

Decimal Adjust. This bit is used to specify what type of instruction was executed last during BCD operations, so a subsequent Decimal Adjust operation can function correctly. This bit is not usually accessible to programmers, and cannot be used as a test condition.

Overflow Flag. This flag is set to 1 when the result of a twos-complement operation was greater than 127 or less than -128. It is also cleared to 0 during logical operations.

Sign Flag. Following arithmetic, logical, rotate, or shift operations, this bit identifies the state of the MSB of the result. A 0 indicates a positive number and a 1 indicates a negative number.

Zero Flag. For arithmetic and logical operations, this flag is set to 1 if the result of the operation is zero.

For operations that test bits in a register, the zero bit is set to 1 if the result is zero.

For rotate and shift operations, this bit is set to 1 if the result is zero.

Carry Flag. This flag is set to 1 if the result from an arithmetic operation generates a carry out of, or a borrow into, bit 7.

After rotate and shift operations, it contains the last value shifted out of the specified register.

It can be set, cleared, or complemented by instructions.

Condition Codes

The flags C, Z, S, and V are used to control the operation of conditional jump instructions.

The opcode of a conditional jump contains a 4-bit field called the condition code (cc). This specifies under which conditions it is to execute the jump. For example, a conditional jump with the condition code for "equal" after a compare operation only jumps if the two operands are equal.

The condition codes and their meanings are given in Table 4.

Addressing Modes

All operands except for immediate data and condition codes are expressed as register addresses, program memory addresses, or data memory addresses. The addressing modes and their designations are:

Register (R)
Indirect Register (IR)
Indexed (X)
Direct (DA)
Relative (RA)
Immediate (IM)
Indirect (IA)

Table 4. Condition Codes and Meanings

Binary	Mnemonic	Flags	Meaning
0000	F	—	Always false
1000	—	—	Always true
0111*	C	C = 1	Carry
1111*	NC	C = 0	No carry
0110*	Z	Z = 1	Zero
1110*	NZ	Z = 0	Not zero
1101	PL	S = 0	Plus
0101	MI	S = 1	Minus
0100	OV	V = 1	Overflow
1100	NOV	V = 0	No overflow
0110*	EQ	Z = 1	Equal
1110*	NE	Z = 0	Not equal
1001	GE	(S XOR V) = 0	Greater than or equal
0001	LT	(S XOR V) = 1	Less than
1010	GT	(Z OR (S XOR V)) = 0	Greater than
00†0	LE	(Z OR (S XOR V)) = 1	Less than or equal
1111*	UGE	C = 0	Unsigned greater than or equal
0111*	ULT	C = 1	Unsigned less than
1011	UGT	(C = 0 AND Z = 0) = 1	Unsigned greater than
0011	ULE	(C OR Z) = 1	Unsigned less than or equal

NOTE: Asterisks (*) indicate condition codes that relate to two different mnemonics but test the same flags. For example, Z and EQ are both True if the Zero flag is set, but after an ADD instruction, Z would probably be used, while after a CP instruction, EQ would probably be used.

Registers can be addressed by an 8-bit address in the range of 0 to 255. Working registers can also be addressed using 4-bit addresses, where five bits contained in a register pointer (R218 or R219) are concatenated with three bits from the 4-bit address to form an 8-bit address.

Registers can be used in pairs to generate 16-bit program or data memory addresses.

Notation and Encoding

The instruction set notations are described in Table 5.

Functional Summary of Commands

Figure 10 shows the formats followed by a quick reference guide to the commands.

Table 5. Instruction Set Notations

Notation	Meaning	Notation	Meaning
cc	Condition code (see Table 4)	DA	Direct address (between 0 and 65535)
r	Working register (between 0 and 15)	RA	Relative address
rb	Bit of working register	IM	Immediate
r0	Bit 0 of working register	IML	Immediate long
R	Register or working register	dst	Destination operand
RR	Register pair or working register pair (Register pairs always start on an even-number boundary)	src	Source operand
		@	Indirect address prefix
IA	Indirect address	SP	Stack pointer
Ir	Indirect working register	PC	Program counter
IR	Indirect register or indirect working register	IP	Instruction pointer
Irr	Indirect working register pair	FLAGS	Flags register
IRR	Indirect register pair or indirect working register pair	RP	Register pointer
X	Indexed	#	Immediate operand prefix
XS	Indexed, short offset	%	Hexadecimal number prefix
XL	Indexed, long offset	OPC	Opcode

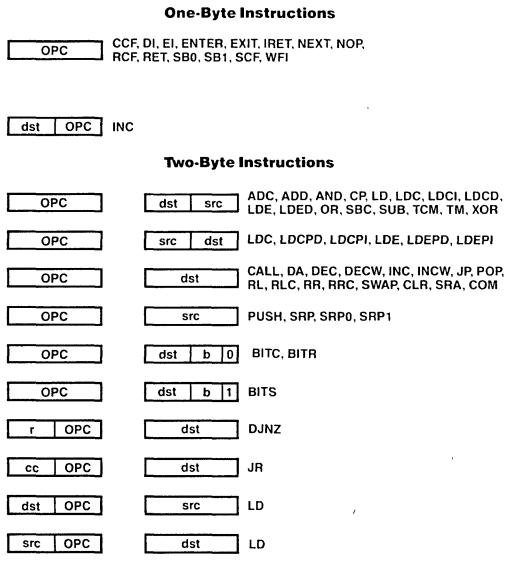


Figure 10. Instruction Formats

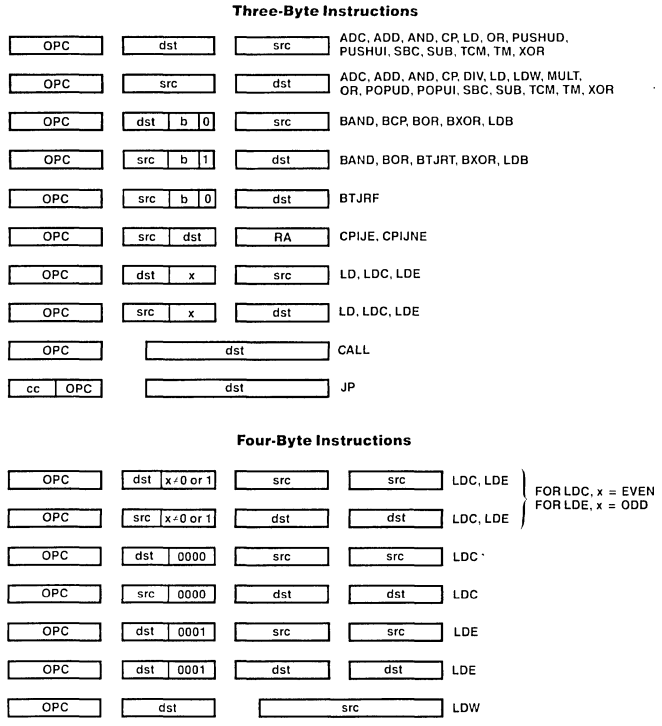


Figure 10. Instruction Formats (Continued)

INSTRUCTION SUMMARY

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
ADC dst,src dst ← dst + src + C	(Note 1)		1☐	*	*	*	—	0	*
ADD dst,src dst ← dst + src	(Note 1)		0☐	*	*	*	*	0	*
AND dst,src dst ← dst AND src	(Note 1)		5☐	—	*	*	0	—	—
BAND dst,src dst ← dst AND src	r0 Rb	Rb r0	67 67	—	*	0	U	—	—
BCP dst, src dst – src	r0	Rb	17	—	*	0	U	—	—
BITC dst dst ← NOT dst	rb		57	—	*	0	U	—	—
BITR dst dst ← 0	rb		77	—	—	—	—	—	—
BITS dst dst ← 1	rb		77	—	—	—	—	—	—

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
BOR dst, src dst ← dst OR src	r0 Rb	rB r0	07	—	*	0	U	—	—
BTJRF if src = 0, PC = PC + dst	RA	rb	37	—	—	—	—	—	—
BTJRT if src = 1, PC = PC + dst	RA	rb	37	—	—	—	—	—	—
BXOR dst, src dst ← dst XOR src	r0 Rb	Rb r0	27 27	—	*	0	U	—	—
CALL dst SP ← SP – 2 @SP ← PC PC ← dst	DA IRR IA		F6 F4 D4	—	—	—	—	—	—
CCF C = NOT C			EF	*	—	—	—	—	—
CLR dst dst ← 0	R IR		B0 B1	—	—	—	—	—	—

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
COM dst dst ← NOT dst	R IR		60 61	—	*	*	0	—	—
CP dst,src dst − src	(Note 1)		A□	*	*	*	*	—	—
CPIJE if dst − src = 0,then PC ← PC + RA Ir ← Ir + 1	r	Ir	C2	—	—	—	—	—	—
CPIJNE if dst − src = 0,then PC ← PC + RA Ir ← Ir + 1	r	Ir	D2	—	—	—	—	—	—
DA dst dst ← DA dst	R IR		40 41	*	*	*	U	—	—
DEC dst dst ← dst − 1	R IR		00 01	—	*	*	*	—	—
DECW dst dst ← dst − 1	RR IR		80 81	—	*	*	*	—	—
DI SMR (0) ← 0			8F	—	—	—	—	—	—
DIV dst, src dst ÷ src dst (Upper) ← Quotient dst (Lower) ← Remainder	RR RR RR	R IR IM	94 95 96	*	*	*	*	—	—
DJNZ r,dst r ← r − 1 if r = 0 PC ← PC + dst	RA	r	rA (r = 0 to F)	—	—	—	—	—	—
EI SMR (0) ← 1			9F	—	—	—	—	—	—
ENTER SP ← SP − 2 @ SP ← IP IP ← PC PC ← @ IP IP ← IP + 2			1F	—	—	—	—	—	—
EXIT IP ← @SP SP ← SP + 2 PC ← @IP IP ← IP + 2			2F	—	—	—	—	—	—
INC dst dst ← dst + 1	r R IR		rE (r = 0 to F) 20 21	—	*	*	*	—	—

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
INCW dst dst ← 1 + dst	RR IR		A0 A1	—	*	*	*	—	—
IRET (Fast) PC ↔ IP FLAG ← FLAG' FIS ← 0			BF	Restored to before interrupt					
IRET (Normal) FLAGS ← @SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; SMR (0) ← 1			BF	Restored to before interrupt					
JP cc,dst if cc is true, PC ← dst	DA IRR		ccD (cc = 0 to F) 30	—	—	—	—	—	—
JR cc,dst if cc is true, PC ← PC + d	RA		ccB (cc = 0 to F)	—	—	—	—	—	—
LD dst,src dst ← src	r r R r IR R R R IR IR r x	IM R r IR r R IR IM IM R x r	rC r8 r9 (r = 0 to F) C7 D7 E4 E5 E6 D6 F5 87 97	—	—	—	—	—	—
LDB dst, src dst ← src	r0 Rb	Rb r0	47 47	—	—	—	—	—	—
LDC/LDE dst ← src	r Irr r xs r r x1 r DA	Irr r xs r x1 r r DA r	C3 D3 E7 F7 A7 B7 A7 B7	—	—	—	—	—	—
LDCD/LDED dst, src dst ← src rr ← rr − 1	r	Irr	E2	—	—	—	—	—	—
LDEI/LDCI dst, src dst ← src rr ← rr + 1	r	Irr	E3	—	—	—	—	—	—
LDCPD/LDEPD dst,src rr ← rr − 1 dst ← src	Irr	r	F2	—	—	—	—	—	—

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
LDCPI/LDEPI dst, src rr ← rr + 1 dst ← src	Irr	r	F3	—	—	—	—	—	—
LDW dst, src dst ← src	RR RR RR	RR IR IMM	C4 C5 C6	—	—	—	—	—	—
MULT dst, src	RR RR RR	R IR IM	84 85 86	*	0	*	*	—	—
NEXT PC ← @IP IP ← IP + 2			0F	—	—	—	—	—	—
NOP			FF	—	—	—	—	—	—
OR dst,src dst ← dst OR src	(Note 1)		4□	—	*	*	0	—	—
POP dst dst ← @SP; SP ← SP + 1		R IR	50 51	—	—	—	—	—	—
POPUD dst, src dst ← src IR ← IR − 1	R	IR	92	—	—	—	—	—	—
POPUI dst, src dst ← src IR ← IR + 1	R	IR	93	—	—	—	—	—	—
PUSH src SP ← SP − 1; @SP ← src		R IR	70 71	—	—	—	—	—	—
PUSHUD dst, src IR ← IR − 1 dst ← src	IR	R	82	—	—	—	—	—	—
PUSHUI dst, src IR ← IR + 1 dst ← src	IR	R	83	—	—	—	—	—	—
RCF C ← 0			CF	0	—	—	—	—	—
RET PC ← @SP; SP ← SP + 2			AF	—	—	—	—	—	—
RL dst C ← dst (7) dst (0) ← dst (7) dst (N + 1) ← dst (N) N = 0 to 6	R IR		90 91	*	*	*	*	—	—
RLC dst dst (0) ← C C ← dst (7) dst (N + 1) ← dst (N) N = 0 to 6	R IR		10 11	*	*	*	*	—	—
RR dst C ← dst (0) dst (7) ← dst (0) dst (N) ← dst (N + 1) N = 0 to 6	R IR		E0 E1	*	*	*	*	—	—
RRC dst C ← dst (0) dst (7) ← C dst (N) ← dst (N + 1) N = 0 to 6	R IR		C0 C1	*	*	*	*	—	—
SB0 BANK ← 0			4F	—	—	—	—	—	—
SB1 BANK ← 1			5F	—	—	—	—	—	—
SBC dst,src dst ← dst − src − C	(Note 1)		3□	*	*	*	*	1	*
SCF C ← 1			DF	1	—	—	—	—	—
SRA dst dst (7) ← dst (7) C ← dst (0) dst (N) ← dst (N + 1) N = 0 to 6	R IR		D0 D1	*	*	*	0	—	—
SRP src RP0 ← IM RP1 ← IM + 8		IM	31	—	—	—	—	—	—
SRP0 RP0 ← IM		IM	3I	—	—	—	—	—	—
SRP1 RP1 ← IM		IM	3I	—	—	—	—	—	—
SUB dst,src dst ← dst − src	(Note 1)		2□	*	*	*	*	1	*

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
SWAP dst dst (0-3) ↔ dst (4-7)	R IR		F0 F1	—	*	*	U	—	—
TCM dst,src (NOT dst) AND src	(Note 1)		6□	—	*	*	0	—	—
TM dst,src dst AND src	(Note 1)		7□	—	*	*	0	—	—
WFI			3F	—	—	—	—	—	—
XOR dst,src dst ← dst XOR src	(Note 1)		B□	—	*	*	0	—	—

NOTE 1: These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble identifies the command, and is found in the table above. The second nibble, represented by a □, defines the addressing mode as shown in Table 6.:

Table 6. Second Nibble

Addr Mode dst	Addr Mode src	Lower Opcode Nibble
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6

For example, to use an opcode represented as x□ with an "RR" addressing mode, use the opcode "x4."

0 = Cleared to Zero
1 = Set to One
— = Unaffected
* = Set or reset, depending on result of operation.
U = Undefined

SUPER-8 OPCODE MAP

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6 **DEC** R_1	6 **DEC** IR_1	6 **ADD** r_1,r_2	6 **ADD** r_1,Ir_2	10 **ADD** R_2,R_1	10 **ADD** IR_2,R_1	10 **ADD** R_1,IM	10 **BOR*** r_0-R_b	6 **LD** r_1,R_2	6 **LD** r_2,R_1	12/10 **DJNZ** r_1,RA	12/10 **JR** cc,RA	6 **LD** r_1,IM	12/10 **JP** cc,DA	6 **INC** r1	14 **NEXT**
1	6 **RLC** R_1	6 **RLC** IR_1	6 **ADC** r_1,r_2	6 **ADC** r_1,Ir_2	10 **ADC** R_2,R_1	10 **ADC** IR_2,R_1	10 **ADC** R_1,IM	10 **BCP** r_1,b,R_2								20 **ENTER**
2	6 **INC** R_1	6 **INC** IR_1	6 **SUB** r_1,r_2	6 **SUB** r_1,Ir_2	10 **SUB** R_2,R_1	10 **SUB** IR_2,R_1	10 **SUB** R_1,IM	10 **BXOR*** r_0-R_b								22 **EXIT**
3	10 **JP** IRR_1	**NOTE C**	6 **SBC** r_1,r_2	6 **SBC** r_1,Ir_2	10 **SBC** R_2,R_1	10 **SBC** IR_2,R_1	10 **SBC** R_1,IM	**NOTE A**								6 **WFI**
4	6 **DA** R_1	6 **DA** IR_1	6 **OR** r_1,r_2	6 **OR** r_1,Ir_2	10 **OR** R_2,R_1	10 **OR** IR_2,R_1	10 **OR** R_1,IM	10 **LDB*** r_0-R_b								6 **SBO**
5	10 **POP** R_1	10 **POP** IR_1	6 **AND** r_1,r_2	6 **AND** r_1,Ir_2	10 **AND** R_2,R_1	10 **AND** IR_2,R_1	10 **AND** R_1,IM	8 **BITC** r_1,b								6 **SBI**
6	6 **COM** R_1	6 **COM** IR_1	6 **TCM** r_1,r_2	6 **TCM** r_1,Ir_2	10 **TCM** R_2,R_1	10 **TCM** IR_2,R_1	10 **TCM** R_1,IM	10 **BAND*** r_0-R_b								
7	10/12 **PUSH** R_2	12/14 **PUSH** IR_2	6 **TM** r_1,r_2	6 **TM** r_1,Ir_2	10 **TM** R_2,R_1	10 **TM** IR_2,R_1	10 **TM** R_1,IM	**NOTE B**								
8	10 **DECW** RR_1	10 **DECW** IR_1	10 **PUSHUD** IR_1,R_2	10 **PUSHUI** IR_1,R_2	24 **MULT** R_2,RR_1	24 **MULT** IR_2,RR_1	24 **MULT** IM,RR_1	10 **LD** r_1,x,r_2								6 **DI**
9	6 **RL** R_1	6 **RL** IR_1	10 **POPUD** IR_2,R_1	10 **POPUI** IR_2,R_1	28/12 **DIV** R_2,RR_1	28/12 **DIV** IR_2,RR_1	28/12 **DIV** IM,RR_1	10 **LD** r_2,x,r_1								6 **EI**
A	10 **INCW** RR_1	10 **INCW** IR_1	6 **CP** r_1,r_2	6 **CP** r_1,Ir_2	10 **CP** R_2,R_1	10 **CP** IR_2,R_1	10 **CP** R_1,IM	**NOTE D**								14 **RET**
B	6 **CLR** R_1	6 **CLR** IR_1	6 **XOR** r_1,r_2	6 **XOR** r_1,Ir_2	10 **XOR** R_2,R_1	10 **XOR** IR_2,R_1	10 **XOR** R_1,IM	**NOTE E**								16/6 **IRET**
C	6 **RRC** R_1	6 **RRC** IR_1	16/18 **CPIJE** Ir,r_2,RA	12 **LDC*** r_1,Irr_2	10 **LDW** RR_2,RR_1	10 **LDW** IR_2,RR_1	12 **LDW** RR_1,IML	6 **LD** r_1,Ir_2								6 **RCF**
D	6 **SRA** R_1	6 **SRA** IR_1	16/18 **CPIJNE** Ir_1,r_2,RA	12 **LDC*** r_2,Irr_1	20 **CALL** IA_1		10 **LD** IR_1,IM	16 **LD** Ir_1,r_2								6 **SCF**
E	6 **RR** R_1	6 **RR** IR_1	16 **LDCD*** r_1,Irr_2	16 **LDCI*** r_1,Irr_2	10 **LD** R_2,R_1	10 **LD** IR_2,R_1	10 **LD** R_1,IM	18 **LDC*** r_1,Irr_2,xs								6 **CCF**
F	8 **SWAP** R_1	8 **SWAP** IR_1	16 **LDCPD*** r_2,Irr_1	16 **LDCPI*** r_2,Irr_1	18 **CALL** IRR_1	10 **LD** R_2,IR_1	18 **CALL** DA_1	18 **LDC*** r_2,Irr_1,xs								6 **NOP**

NOTE A

16/18	16/18
BTJRF	**BTJRT**
r_2,b,RA	r_2,b,RA

NOTE B

8	8
BITR	**BITS**
r_1,b	r_1,b

NOTE C

6	6	6
SRP	**SRP0**	**SRP1**
IM	IM	IM

NOTE D

20	20
LDC*	**LDC***
r_1,Irr_2,xL	r_1,DA_2

NOTE E

20	20
LDC*	**LDC***
r_2,Irr_2,xL	r_2,DA_1

Legend:
r = 4-bit address
R = 8-bit address
b = bit number
R_1 or r_1 = dst address
R_2 or r_2 = src address

***Examples:**
BOR r_0-R_2
is BOR r_1,b,R_2
or BOR r_2,b,R_1
LDC r_1,Irr_2
is LDC r_1,Irr_2 = program
or LDE r_1,Irr_2 = data

Sequence:
Opcode, first, second, third operands

NOTE The blank areas are not defined

Figure 11. Opcode Map

Table 7. Super8 Instructions

Mnemonic	Operands	Instruction
Load Instructions		
CLR	dst	Clear
LD	dst, src	Load
LDB	dst, src	Load bit
LDC	dst, src	Load program memory
LDE	dst, src	Load data memory
LDCD	dst, src	Load program memory and decrement
LDED	dst, src	Load data memory and decrement
LDCI	dst, src	Load program memory and increment
LDEI	dst, src	Load data memory and increment
LDCPD	dst, src	Load program memory with pre-decrement
LDEPD	dst, src	Load data memory with pre-decrement
LDCPI	dst, src	Load program memory with pre-increment
LDEPI	dst, src	Load data memory with pre-increment
LDW	dst, src	Load word
POP	dst	Pop stack
POPUD	dst, src	Pop user stack (decrement)
POPUI	dst, src	Pop user stack (increment)
PUSH	src	Push stack
PUSHUD	dst, src	Push user stack (decrement)
PUSHUI	dst, src	Push user stack (increment)
Arithmetic Instructions		
ADC	dst, src	Add with carry
ADD	dst, src	Add
CP	dst, src	Compare
DA	dst	Decimal adjust
DEC	dst	Decrement
DECW	dst	Decrement word
DIV	dst, src	Divide
INC	dst	Increment
INCW	dst	Increment word
MULT	dst, src	Multiply
SBC	dst, src	Subtract with carry
SUB	dst, src	Subtract
Logical Instructions		
AND	dst, src	Logical AND
COM	dst	Complement
OR	dst, src	Logical OR
XOR	dst, src	Logical exclusive

Mnemonic	Operands	Instruction
Program Control Instructions		
BTJRT	dst, src	Bit test jump relative on True
BTJRF	dst, src	Bit test jump relative on False
CALL	dst	Call procedure
CPIJE	dst, src	Compare, increment and jump on equal
CPIJNE	dst, src	Compare, increment and jump on non-equal
DJNZ	r, dst	Decrement and jump on non-zero
ENTER		Enter
EXIT		Exit
IRET		Return from interrupt
JP	cc, dst	Jump on condition code
JP	dst	Jump unconditional
JR	cc, dst	Jump relative on condition code
JR	dst	Jump relative unconditional
NEXT		Next
RET		Return
WFI		Wait for interrupt
Bit Manipulation Instructions		
BAND	dst, src	Bit AND
BCP	dst, src	Bit compare
BITC	dst	Bit complement
BITR	dst	Bit reset
BITS	dst	Bit set
BOR	dst, src	Bit OR
BXOR	dst, src	Bit exclusive OR
TCM	dst, src	Test complement under mask
TM	dst, src	Test under mask
Rotate and Shift Instructions		
RL	dst	Rotate left
RLC	dst	Rotate left through carry
RR	dst	Rotate right
RRC	dst	Rotate right through carry
SRA	dst	Shift right arithmetic
SWAP	dst	Swap nibbles
CPU Control Instructions		
CCF		Complement carry flag
DI		Disable interrupts
EI		Enable interrupts
NOP		Do nothing
RCF		Reset carry flag
SB0		Set bank 0
SB1		Set bank 1
SCF		Set carry flag
SRP	src	Set register pointers
SRP0	src	Set register pointer zero
SRP1	src	Set register pointer one

INTERRUPTS

The Super8 interrupt structure contains 8 levels of interrupt, 16 vectors, and 27 sources.

Interrupt priority is assigned by level, controlled by the Interrupt Priority register (IPR). Each level is masked (or enabled) according to the bits in the Interrupt Mask register (IMR), and the entire interrupt structure can be disabled by clearing a bit in the System Mode register (R222).

The three major components of the interrupt structure are sources, vectors, and levels. These are shown in Figure 10 and discussed in the following paragraphs.

Sources

A source is anything that generates an interrupt. This can be internal or external to the Super8 MCU. Internal sources are hardwired to a particular vector and level, while external sources can be assigned to various external events.
External interrupts are falling-edge triggered.

Vectors

The 16 vectors are divided unequally among the eight levels. For example, vector 12 belongs to level 2, while level 3 contains vectors 0, 2, 4, and 6.

The vector number is used to generate the address of a particular interrupt servicing routine; therefore all interrupts using the same vector must use the same interrupt handling routine.

Levels

Levels provide the top level of priority assignment. While the sources and vectors are hardwired within each level, the priorities of the levels can be changed by using the Interrupt Priority register (see Figure 8 for bit details).

If more than one interrupt source is active, the source from the highest priority level will be serviced first. If both sources are from the same level, the source with the lowest vector will have priority. For example, if the UART Receive Data bit and UART Parity Error bit are both active, the UART Parity Error bit will be serviced first because it is vector 16, and UART receive data is vector 20.

The levels are shown in Figure 12.

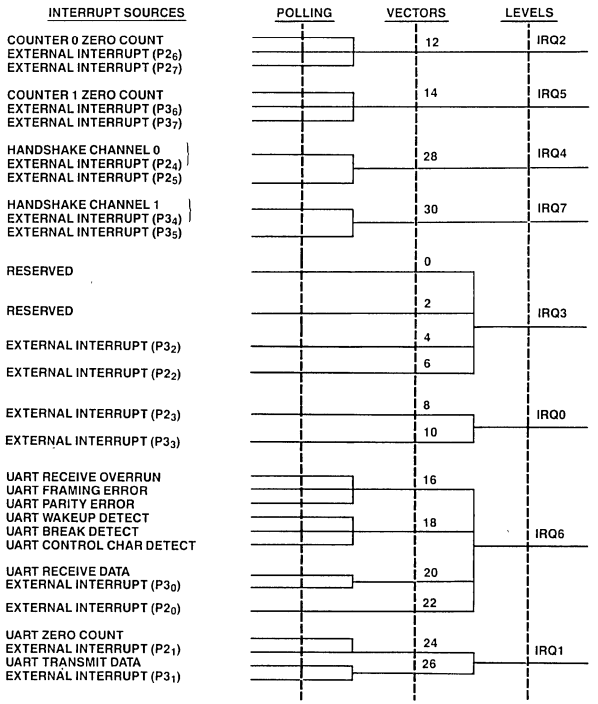


Figure 12. Interrupt Levels and Vectors

Enables

Interrupts can be enabled or disabled as follows:

- Interrupt enable/disable. The entire interrupt structure can be enabled or disabled by setting bit 0 in the System Mode register (R222).
- Level enable. Each level can be enabled or disabled by setting the appropriate bit in the Interrupt Mask register (R221).
- Level priority. The priority of each level can be controlled by the values in the Interrupt Priority register (R255, Bank 0).
- Source enable/disable. Each interrupt source can be enabled or disabled in the sources' Mode and Control register.

Service Routines

Before an interrupt request can be granted, a) interrupts must be enabled, b) the level must be enabled, c) it must be the highest priority interrupting level, d) it must be enabled at the interrupting source, and e) it must have the highest priority within the level.

If all this occurs, an interrupt request is granted.

The Super8 then enters an interrupt machine cycle that completes the following sequence:

- It resets the Interrupt Enable bit to disable all subsequent interrupts.
- It saves the Program Counter and status flags on the stack.
- It branches to the address contained within the vector location for the interrupt.
- It passes control to the interrupt servicing routine.

When the interrupt servicing routine has serviced the interrupt, it should issue an interrupt return (IRET) instruction. This restores the Program Counter and status flags and sets the Interrupt Enable bit in the System Mode register.

Fast Interrupt Processing

The Super8 provides a feature called fast interrupt processing, which completes the interrupt servicing in 6 clock periods instead of the usual 22.

Two hardware registers support fast interrupts. The Instruction Pointer (IP) holds the starting address of the service routine, and saves the PC value when a fast interrupt occurs. A dedicated register, FLAG', saves the contents of the FLAGS register when a fast interrupt occurs.

To use this feature, load the address of the service routine in the Instruction Pointer, load the level number into the Fast Interrupt Select field, and turn on the Fast Interrupt Enable bit in the System Mode register.

When an interrupt occurs in the level selected for fast interrupt processing, the following occurs:

- The contents of the Instruction Pointer and Program Counter are swapped.
- The contents of the Flag register are copied into FLAG'.
- The Fast Interrupt Status Bit in FLAGS is set.
- The interrupt is serviced.
- When IRET is issued after the interrupt service outline is completed, the Instruction Pointer and Program Counter are swapped again.
- The contents of FLAG' are copied back into the Flag register.
- The Fast Interrupt Status bit in FLAGS is cleared.

The interrupt servicing routine selected for fast processing should be written so that the location after the IRET instruction is the entry point the next time the (same) routine is used.

Level or Edge Triggered

Because internal interrupt requests are levels and interrupt requests from the outside are (usually) edges, the hardware for external interrupts uses edge-triggered flip-flops to convert the edges to levels.

The level-activated system requires that interrupt-serving software perform some action to remove the interrupting source. The action involved in serving the interrupt may remove the source, or the software may have to actually reset the flip-flops by writing to the corresponding Interrupt Pending register.

STACK OPERATION

The Super8 architecture supports stack operations in the register file or in data memory. Bit 1 in the external Memory Timing register (R254 bank 0) selects between the two.

Register pair 216-217 forms the Stack Pointer used for all stack operations. R216 is the MSB and R217 is the LSB.

The Stack Pointer always points to data stored on the top of the stack. The address is decremented prior to a PUSH and incremented after a POP.

The stack is also used as a return stack for CALLs and interrupts. During a CALL, the contents of the PC are saved on the stack, to be restored later. Interrupts cause the contents of the PC and FLAGS to be saved on the stack, for recovery by IRET when the interrupt is finished.

When the Super8 is configured for an internal stack (using the register file), R217 contains the Stack Pointer. R216 may be used as a general-purpose register, but its contents will be changed if an overflow or underflow occurs as the result of incrementing or decrementing the stack address during normal stack operations.

User-Defined Stacks

The Super8 provides for user-defined stacks in both the register file and program or data memory. These can be made to increment or decrement on a push by the choice of opcodes. For example, to implement a stack that grows from low addresses to high addresses in the register file, use PUSHUI and POPUD. For a stack that grows from high addresses to low addresses in data memory, use LDEI for pop and LDEPD for push.

COUNTER/TIMERS

The Super8 has two identical independently programmable 16-bit counter/timers that can be cascaded to produce a single 32-bit counter. They can be used to count external events, or they can obtain their input internally. The internal input is obtained by dividing the crystal frequency by four.

The counter/timers can be set to count up or down, by software or external events. They can be set for single or continuous cycle counting, and they can be set with a bi-value option, where two preset time constants alternate in loading the counter each time it reaches zero. This can be used to produce an output pulse train with a variable duty cycle.

The counter/timers can also be programmed to capture the count value at an external event or generate an interrupt whenever the count reaches zero. They can be turned on and off in response to external events by using a gate and/or a trigger option. The gate option enables counts only when the gate line is Low; the trigger option turns on the counter after a transient High. The gate and trigger options used together cause the counter/timer to work in gate mode after initially being triggered.

The control and status register bits for the counter/timers are shown in Figure 5.

DMA

The Super8 features an on-chip Direct Memory Access (DMA) channel to provide high bandwidth data transmission capabilities. The DMA channel can be used by the UART receiver, UART transmitter, or handshake channel 0. Data can be transferred between the peripheral and contiguous locations in either the register file or external data memory. A 16-bit count register determines the number of transactions to be performed; an interrupt can be generated when the count is exhausted. DMA transfers to or from the register file require six CPU clock cycles; DMA transfers to or from external memory take ten CPU clock cycles, excluding wait states.

ABSOLUTE MAXIMUM RATINGS

Voltage on all pins with respect to ground . . . . . −0.3V to +7.0V
Ambient Operating Temperature . . . . . .See Ordering Information
Storage Temperature . . . . . −65°C to +150°C

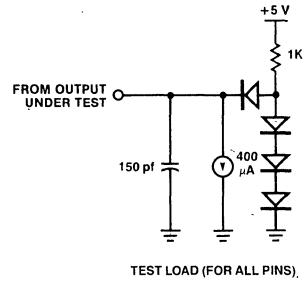
Stresses greater than these may cause permanent damage to the device. This is a stress rating only; operation of the device under conditions more severe than those listed for operating conditions may cause permanent damage to the device. Exposure to absolute maximum ratings for extended periods may also cause permanent damage.

STANDARD TEST CONDITIONS

Figure 14 shows the setup for standard test conditions. All voltages are referenced to ground, and positive current flows into the reference pin.

Standard conditions are:

- $+4.75V \leq V_{CC} \leq +5.25V$
- GND = 0V
- $0°C \leq T_A \leq +70°C$

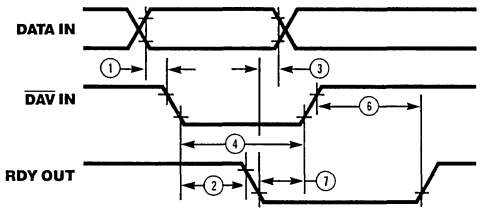


Standard Test Load

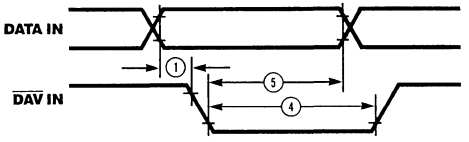
DC CHARACTERISTICS

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	3.8	V_{CC}	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	−0.3	0.8	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.2	V_{CC}	V	
V_{IL}	Input Low Voltage	−0.3	0.8	V	
V_{RH}	Reset Input High Voltage	3.8	V_{CC}	V	
V_{RL}	Reset Input Low Voltage	−0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −400 µA
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +4.0 mA
I_{IL}	Input Leakage	−10	10	µA	
I_{OL}	Output Leakage	−10	10	µA	
I_{IR}	Reset Input Current		−50	µA	
I_{CC}	V_{CC} Supply Current		320	mA	

INPUT HANDSHAKE TIMING



Fully Interlocked Mode



Strobed Mode

AC CHARACTERISTICS (20 MHz)

Input Handshake

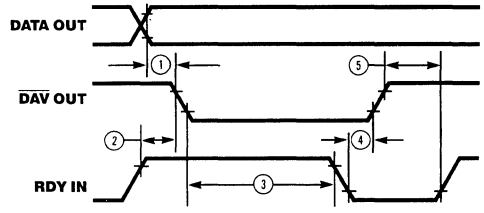
Number	Symbol	Parameter	Min	Max	Notes*‡
1	TsDI(DAV)	Data In to Setup Time	0		
2	TdDAVlf(RDY)	$\overline{DAV}$ ↓ Input to RDY ↓ Delay		200	1
3	ThDI(RDY)	Data In Hold Time from RDY ↓	0		
4	TwDAV	$\overline{DAV}$ In Width	45		
5	ThDI(DAV)	Data In Hold Time from $\overline{DAV}$ ↓	130		
6	TdDAV(RDY)	$\overline{DAV}$ ↑ Input to RDY ↑ Delay		100	2
7	TdRDYf(DAV)	RDY ↓ Output to $\overline{DAV}$ ↑ Delay	0		

NOTES:
1. Standard Test Load
2. This time assumes user program reads data before $\overline{DAV}$ Input goes high. RDY will not go high before data is read.

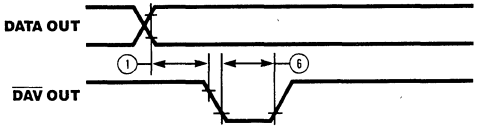
‡Times given are in ns.

*Times are preliminary and subject to change.

OUTPUT HANDSHAKE TIMING



Fully Interlocked Mode



Strobed Mode

AC CHARACTERISTICS (12 MHz, 20 MHz)

Output Handshake

Number	Symbol	Parameter	Min	Max	Notes*‡
1	TdDO(DAV)	Data Out to $\overline{DAV}$ ↓ Delay	90		1,2
2	TdRDYr(DAV)	RDY ↑ Input to $\overline{DAV}$ ↓ Delay	0	110	1
3	TdDAVOf(RDY)	$\overline{DAV}$ ↓ Output to RDY ↓ Delay	0		
4	TdRDYf(DAV)	RDY ↓ Input to $\overline{DAV}$ ↑ Delay	0	110	1
5	TdDAVOr(RDY)	$\overline{DAV}$ ↑ Output to RDY ↑ Delay	0		
6	TwDAVO	$\overline{DAV}$ Output Width	150		2

NOTES:
1. Standard Test Load
2. Time given is for zero value in Deskew Counter. For nonzero value of n where n = 1, 2,. . . 15 add 2 × n × TpC to the given time.
‡Times given are in ns.
*Times are preliminary and subject to change.

AC CHARACTERISTICS (12 MHz)

Read/Write

			Normal Timing		Extended Timing		
Number	Symbol	Parameter	Min	Max	Min	Max	Notes‡*
1	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	35		115		
2	TdAS(A)	$\overline{AS}$ ↑ to Address Float Delay	65		150		
3	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Required Valid		270		600	1
4	TwAS	$\overline{AS}$ Low Width	65		150		
5	TdA(DS)	Address Float to $\overline{DS}$ ↓	20		20		
6a	TwDS(Read)	$\overline{DS}$ (Read) Low Width	225		470		1
6b	TwDS(Write)	$\overline{DS}$ (Write) Low Width	130		295		1
7	TdDS(DR)	$\overline{DS}$ ↓ to Read Data Required Valid		180		420	1
8	ThDS(DR)	Read Data to $\overline{DS}$ ↑ Hold Time	0		0		
9	TdDS(A)	$\overline{DS}$ ↑ to Address Active Delay	50		135		
10	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	60		145		
11	TdDO(DS)	Write Data Valid to $\overline{DS}$ (Write) ↓ Delay	35		115		
12	TdAS(W)	$\overline{AS}$ ↑ to Wait Delay		220		600	2
13	ThDS(W)	$\overline{DS}$ ↑ to Wait Hold Time	0		0		
14	TdRW(AS)	R/$\overline{W}$ Valid to $\overline{AS}$ ↑ Delay	50		135		

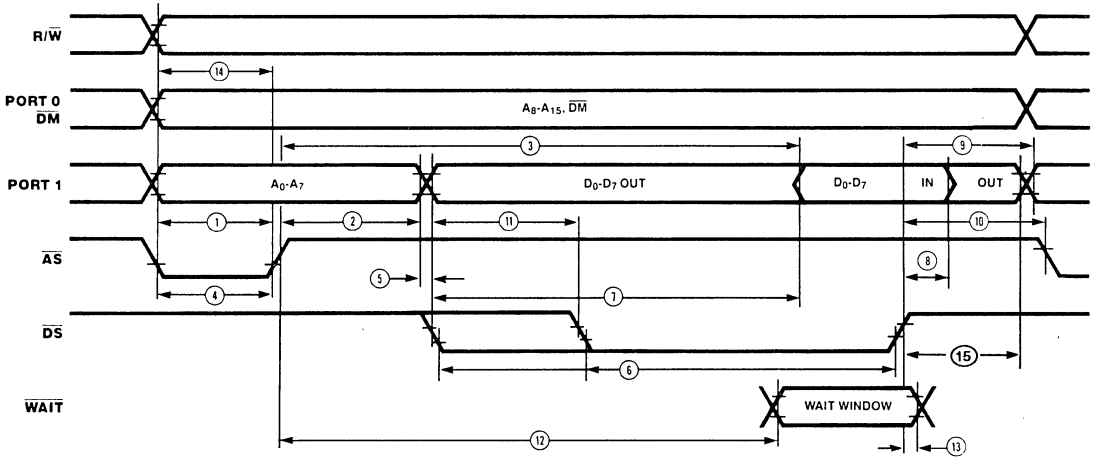
NOTES:
1. WAIT states add 167 ns to these times.
2. Auto-wait states add 167 ns to this time.
‡ All times are in ns and are for 12 MHz input frequency.
* Timings are preliminary and subject to change.

AC CHARACTERISTICS (20 MHz)

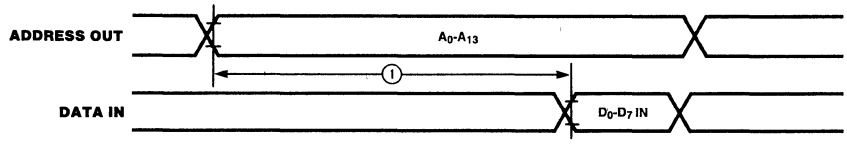
Read/Write

Number	Symbol	Parameter	Normal Timing Min	Normal Timing Max	Extended Timing Min	Extended Timing Max	Notes‡*
1	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	10		50		
2	TdAS(A)	$\overline{AS}$ ↑ to Address Float Delay	35		85		
3	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Required Valid		140		335	1
4	TwAS	$\overline{AS}$ Low Width	35		85		
5	TdA(DS)	Address Float to $\overline{DS}$ ↓	0		0		
6a	TwDS(Read)	$\overline{DS}$ (Read) Low Width	125		275		1
6b	TwDS(Write)	$\overline{DS}$ (Write) Low Width	65		165		1
7	TdDS(DR)	$\overline{DS}$ ↓ to Read Data Required Valid		80		225	1
8	ThDS(DR)	Read Data to $\overline{DS}$ ↑ Hold Time	0		0		
9	TdDS(A)	$\overline{DS}$ ↑ to Address Active Delay	20		70		
10	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	30		80		
11	TdDO(DS)	Write Data Valid to $\overline{DS}$ (Write) ↓ Delay	10		50		
12	TdAS(W)	$\overline{AS}$ ↑ to Wait Delay		90		335	2
13	ThDS(W)	$\overline{DS}$ ↑ to Wait Hold Time	0		0		
14	TdRW(AS)	R/$\overline{W}$ Valid to $\overline{AS}$ ↑ Delay	20		70		
15	TdDS(DW)	$\overline{DS}$ ↑ to Write Data Not Valid Delay	20		70		

NOTES:
1. WAIT states add 100 ns to these times.
2. Auto-wait states add 100 ns to this time.
‡ All times are in ns and are for 20 MHz input frequency.
* Timings are preliminary and subject to change.



External Memory Read and Write Timing



EPROM Read Timing

AC CHARACTERISTICS (20 MHz)

EPROM Read Cycle

Number	Symbol	Parameter	Min	Max	Notes‡*
1	TdA(DR)	Address Valid to Read Data Required Valid		170	1

NOTES:
1. WAIT states add 167 ns to these times.
‡All times are in ns and are for 12 MHz input frequency.
*Timings are preliminary and subject to change.

PERIPHERAL PRODUCTS

Z86128
CLOSED-CAPTIONED CONTROLLER

FEATURES

- Complete stand-alone Line 21 Closed-Caption Controller

  -Conforms to FCC Line 21 Closed-Caption Specifications

- Simple system interface
- Requires only two inputs to operate

  - Composite video
  - H flyback

- On-board Analog Sync and Data Slicer
- CMOS VLSI design for low power and low cost
- On-board Display RAM
- On-board character font ROM - 6x9 character in 8x13 cell

  - Character rounding - 12x18 actual character cell

  - Visual Attributes
    Color
    Underline
    Italic
    Blink

- Smooth scrolling
- Automatic screen blanking after 1.5 sec with no input.
- Automatic erase after 16 sec with no input
- 18-pin package

GENERAL DESCRIPTION

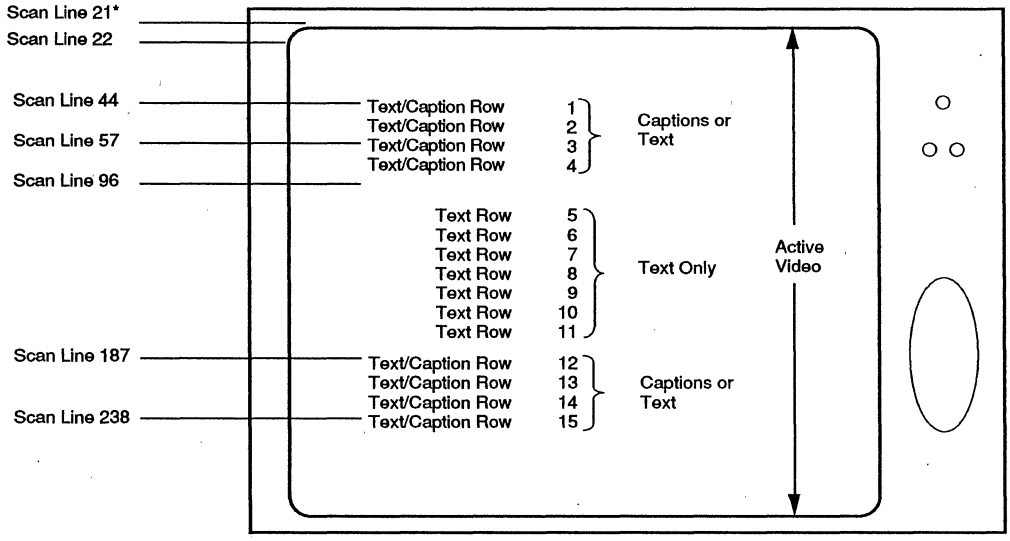
The Z86128 Line 21 Closed-Caption Controller is a single IC, designed to provide the functional performance of a Line 21 Closed-Caption depicted in the Figure 1 Decoder module. This Super Integration™ VLSI device is completely self contained, requiring only composite video, H Flyback inputs and an external keyer* to produce captioned video. The Z86128 uses a wired logic approach to perform the functions selected through its input control signals. It is fabricated using standard CMOS technology and designed to achieve the lowest possible cost.

The Z86128 is intended for use in a set-top decoder or in any television receiver conforming to the NTSC standard. It is capable of processing all standard Line 21 closed-caption format transmissions. If and when PAL and SECAM TV standards define a protocol using the Line 21 format, this design will be readily convertible to that standard.

*External keyer = video switch between TV video and Closed-Caption video.

The Line 21 Closed Captioning System

The Line 21 Closed-Caption system provides for the transmission of CAPTION information and other TEXT material as an encoded composite data signal this is during the unblanked portion of Line 21, field 1 of the standard NTSC video signal. In addition, a framing code is transmitted during the first half of Line 21, field 2. The encoded composite video signal for Line 21, field 1 and 2 is shown in Figure 2. The video signal conforms to the Standard Synchronizing Waveform for Color Transmission given in Sub-part E, Part 73 of the FCC Rules and Regulations.



*Scan Line 21 is the last scan Line of the vertical retrace blank interval

Figure 1. Closed-Caption TV Display Format

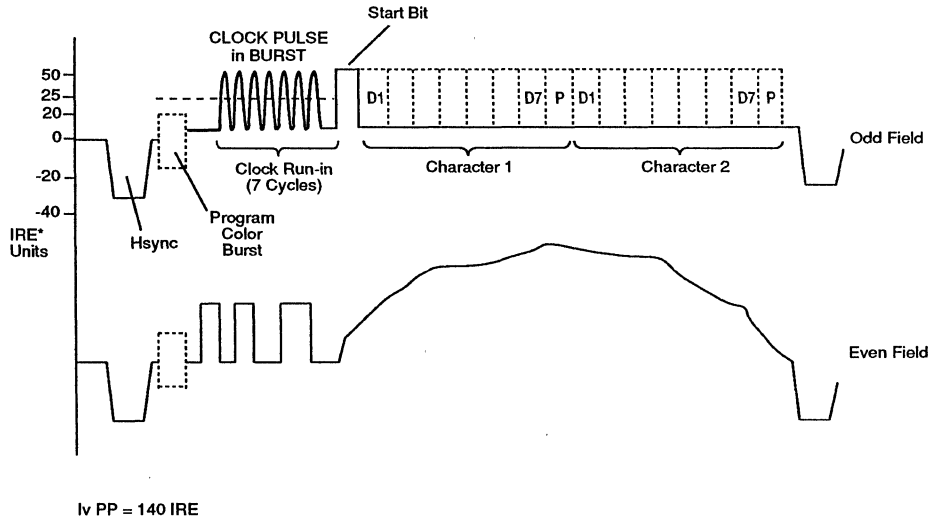


Figure 2. Encoded Composite Video Signal

Fifteen additional displayable characters are sent by transmitting a two-byte code. The byte pair has a non-printing

PIN DESCRIPTION (Actual pinout to be defined)

Pin number assignments subject to change.
For an 18-pin package the additional two pins will be another set of power pins.

Pin 1 Composite Video Input
Composite NTSC video, nominally 1.0V p-p,. band limited to 600 KHz.
Circuit operates with signal variations between 0.5-2.0V p-p. It is recommended that this signal pin be driven by an emitter follower through a 0.1 µF capacitor.

Pin 2 Sync Slice Level
Capacitor (0.1µF) to store sync slice level voltage.

Pin 3 H Flyback Input
Horizontal sync input at CMOS levels, polarity independent.
Typically derived from the H Flyback pulse.

Pin 4 H Loop Filter
Value to be specified

Pin 5 V_{SS}
IC ground. Connect to system ground

Pin 6 Decoder On/Off Input (control)
Decoder display control. CMOS input with HIGH = ON, LOW = OFF

Pin 7 CAPTIONS/TEXT Input (control)
Selects Data Channel to be processed (along with Captions/Text).
CMOS input with HIGH = CAPTIONS, LOW = TEXT.

Pin 8 LANGUAGE I/II Input (control)
Selects Data Channel to be processed (along with Captions/Text).
CMOS input HIGH = LANGUAGE I, LOW = LANGUAGE II.

Pin 9 Box Output
CMOS level "black box" keying signal for Caption/Text display area.

Pin 10 Luminance Output
CMOS level signal.
Character video luminance signal.

Pins 11,12,13 Color signals, RGB Outputs
CMOS level color character video for color receiver use.

Pins 14,15 Test Pins For use in IC test.

Pin 16 V_{DD} ID Power pin. Connect to +5V source.

Pin 17 Analog V_{DD}

Pin 18 Analog V_{SS}

FUNCTIONAL DESCRIPTION

Data Transmission Format

The composite data signal contained within the active portion of Line 21 consists of a seven cycle sine-wave clock run-in burst, a start bit and 16 bits of data. These 16 bits consist of two 8-bit alphanumeric characters formulated according to the USA Standard Code for Information Interchange (USASCII; x 3.4-1967) with odd parity. The clock rate is 0.5035 MHz which is 32fH. The clock burst and data packet are 50 IRE units peak-to-peak and are filtered to a "2T" response. Data is sent with the least significant bit (bit D1) being sent first and the most significant bit (bit D8, the parity bit) sent last.

Multiplexed Data Channels

The Line 21 closed-captioned system defines four different data channels which can be time multiplexed within the Line 21 data stream. They are Captions - LANGUAGE I (C1), Captions - LANGUAGE II (C2), Text - LANGUAGE I (T1) and Text - LANGUAGE II (T2). Both languages can be English in either case.

TEXT is defined as non-video related information therefore its display can fill the screen. TEXT mode displays a black box 15 rows high by 34 columns wide. Text appears starting at the top with a maximum of 32 characters per row (the first and last character locations are always blank and used for asserting visual attributes). When all 15 rows have been used, the display scrolls up as additional information is received.

Captions are video related information so they are not permitted to overwrite the entire screen. Captions may only be displayed in the top four rows and/or the bottom four rows, in any combination. Eight rows may be displayed at one time. All the rows in each caption appear at once in Pop-on Captions mode.

A secondary Caption display mode, called Rollup Captions, is also provided. In this mode, caption information is displayed on the bottom two, three or four rows. Data appears in the bottom row and scrolls up as new information is received. The data scrolls off the top row selected as in the TEXT mode. This mode is usually used for captioning unscripted and fast turn around programming such as talk shows and news.

Data Format

The four data channels are transmitted in Line 21 as a time multiplexed data stream. The start of a particular channel's data stream is identified by the occurrence of one of its unique command codes. Once a unique command code is received, all subsequent data are considered to belong to that data channel until a unique command code is received for another data channel. The Alarm On and Alarm Off codes are an exception to this rule. Alarm codes are ignored by everything except the Alarm output control circuits.

The 7-bit ASCII table defines two types of information, printing and non-printing. Printable data are data bytes having values between x0100000 (20H) and x111111 (7FH), where x represents the parity bit. Data bytes having values between x0000000 (00H) and x0011111 (1FH) are called non-printing characters since they have no displayable font character in the standard ASCII table.

Displayable Character Set

The specifications* define a modified ASCII table character set where eight of the alphanumeric characters have been changed to provide some foreign characters.

In addition, 15 additional characters are defined by special character commands. The changes from the standard ASCII table characters are shown in Table 1.

* - The PBS 1979 specifications are slightly different than the NCI 1985 specifications. The information presented here essentially conforms to the NCI specifications with the exception of the response to "space" characters.

Table 1. Different ASCII Characters

Hex Code	ASCII Value	Line 21 Value
2A	*	ä
5C	\	ë
5E	^	ï
5F	-	ö
60	`	ü
7B	{	ç
7D	}	Ñ
7E	~	ñ

Fifteen additional displayable characters are sent by transmitting a two-byte code. The byte pair has a non-printing character followed by a printing character, where the non-printing character is 11H for LANGUAGE I and 19H for LANGUAGE II. The printing character determines the special font character that will be displayed according to Table 2.

Table 2. Print Character Font Determination

Print	Character
30	1/4
31	#
32	1/2
33	¿
34	3/4
35	¢
36	£
37	k
38	Ä
3A	Ë
3B	â
3C	ê
3D	î
3E	ô
3F	û

The byte pair 11H,39H (or 19H,39H) is defined as a transparent space in the 1985 NCI specification. The Z86128 uses the spacing rule defined in the 1979 PBS specification. In Text mode all spaces are treated as non-transparent characters and always appear within the full box. In Caption mode all spaces are treated as transparent i.e., box is dropped. However, every character must have another character or a black box before and after it.

Commands and Special Information

Data channel commands and special information are transmitted as two byte pairs consisting of a non-printing character followed by a printing character. The two bytes of the pair must be transmitted in the same field and the pair is transmitted twice in successive frames. This redundancy provides some immunity for errors due to noise.

Throughout the Line 21, system bit 4 of the non-printing character identifies the Language. Bit D4 = 0 signifies LANGUAGE I commands and D4 = 1 signifies LANGUAGE II. Only eight of the available 32 non-printing characters are used in the Line 21 system, 11H -14H for LANGUAGE I and 19H - 1CH for LANGUAGE II.

Data Channel Commands. All the data channel command codes use the non-printing character 14H for LANGUAGE I and 1CH for LANGUAGE II. The printing character determines the particular command function. The commands are shown in Table 3. The printing character's value is given in Hex.

Table 3. Data Channel Commands

(Command = 14H + Hex code below)

Data Channel = Captions (C1 or C2) Hex

Print	Function
20	Resume Caption Loading (off screen)
25	Resume 2 Line Roll-up
26	Resume 3 Line Roll-up
27	Resume 4 Line Roll-up
29	Resume Direct Loading (on screen)
2C	Erase Displayed Memory
2E	Erase Non-displayed Memory
2F	Show Caption (flip memories)

Data Channel = Text (T1 or T2)

Print	Function
2A	Start Text
2B	Resume Text

The following commands are shared by all of the data channels:

Print	Function
21	Backspace
28	Flash On/Off
2D	New Line (carriage return)

The Alarm circuit command codes are:

Print	Function
22	Alarm Off
23	Alarm On

FUNCTIONAL DESCRIPTION (Continued)

Data Location and Attribute Codes. Additional codes are used for positioning the data on the screen and for controlling the character attributes. There are two location attributes, row and column (indent) position and three character attributes, color, italics and underline. All attribute information is contained in the Preamble Codes (Precodes) and Midrow Codes (Midcodes).

The Precodes identify the display row and character attributes for the caption data that follows it. These attributes hold for the entire line unless changed by a Midcode or Indent code. All the non-printing characters, 11H-14H for LANGUAGE I and 19H - 1CH for LANGUAGE II are used. The code pair assignments for the location and character attributes are given in Table 4. The grid represents LANGUAGE I only. To use the grid for LANGUAGE II simply replace the non-printcodes (IIH-14H) with 19H-ICH.

Table 4. Code Pair Assignments for Location and Character Attributes

Non-print	11H		12H		13H		14H	
Caption Row	1	2	3	4	12	13	14	15
ATTRIBUTE								
Monochrome	40	60	40	60	40	60	40	60
Mono Underline	41	61	41	61	41	61	41	61
Green	42	62	42	62	42	62	42	62
Green Underline	43	63	43	63	43	63	43	63
Blue	44	64	44	64	44	64	44	64
Blue Underline	45	65	45	65	45	65	45	65
Cyan	46	66	46	66	46	66	46	66
Cyan Underline	47	67	47	67	47	67	47	67
Red	48	68	48	68	48	68	48	68
Red Underline	49	69	49	69	49	69	49	69
Yellow	4A	6A	4A	6A	4A	6A	4A	6A
Yellow Underline	4B	6B	4B	6B	4B	6B	4B	6B
Magenta	4C	6C	4C	6C	4C	6C	4C	6C
Magenta Underline	4D	6D	4D	6D	4D	6D	4D	6D
Italics (mono)	4E	6E	4E	6E	4E	6E	4E	6E
Italics Underline	4F	6F	4F	6F	4F	6F	4F	6F
Indent 0 (mono)	50	70	50	70	50	70	50	70
Indent 0 Underline	51	71	51	71	51	71	51	71
Indent 4	52	72	52	72	52	72	52	72
Indent 4 Underline	53	73	53	73	53	73	53	73
Indent 8	54	74	54	74	54	74	54	74
Indent 8 Underline	55	75	55	75	55	75	55	75
Indent 12	56	76	56	76	56	76	56	76
Indent 12 Underline	57	77	57	77	57	77	57	77
Indent 16	58	78	58	78	58	78	58	78
Indent 16 Underline	59	79	59	79	59	79	59	79
Indent 20	5A	7A	5A	7A	5A	7A	5A	7A
Indent 20 Underline	5B	7B	5B	7B	5B	7B	5B	7B
Indent 24	5C	7C	5C	7C	5C	7C	5C	7C
Indent 24 Underline	5D	7D	5D	7D	5D	7D	5D	7D
Indent 28	5E	7E	5E	7E	5E	7E	5E	7E
Indent 28 Underline	5F	7F	5F	7F	5F	7F	5F	7F

The Midcodes are used to change the character attributes in the middle of a caption row. The Midcode occupies a space in the display and causes a black box to appear just as if the space character 20H had been sent. The characters following the Midcode are displayed with the attributes assigned by the Midcode. These hold until the end of the row unless changed by another Midcode. The Indent codes listed above in the Precode table actually perform in the same manner as a Midcode. Indent commands move the pointer in increments of 4 across the screen.

The Midcodes use the non-printing characters 11H and 19H, respectively, for the two Languages. The printing character of the two byte pair contains the character attributes as shown in Table 5.

Table 5. Print Character Attributes

Print	Character Attribute
20	Monochrome
21	Monochrome Underline
22	Green
23	Green Underlined
24	Blue
25	Blue Underlined
26	Cyan
27	Cyan Underlined
28	Red
29	Red Underlined
2A	Yellow
2B	Yellow Underlined
2C	Magenta
2D	Magenta Underlined
2E	Italics
2F	Italics Underlined

DECODER OPERATION

The Z86128 provides full function Line 21 performance. Switch selection inputs are included to enable the decoder to process and display any of the four data channels (C1, C2, T1 or T2) transmitted in Line 21 of the incoming video. An additional input, Decoder ON/OFF, controls the display. When switched to the decoder off (TV) state, incoming data in the selected channel is still processed but not displayed. An internal pad is also provided to select field 1 or field 2 operation which is present for field 1 operation at this time.

Display Format

Characters are displayed as white or colored; dot matrix character on a black background. The characters are described by 6 X 9 dot patterns within a character cell which is 8 dots wide by 13 dots high. This leaves a one dot border of black around each character and provision for one scanline for underline, offset by one scanline of black, between the character and the bottom edge of the cell. Character luminance is 90 IRE units and the black box surround, 10 IRE units.

The character ROM consists of 12 X 18 dot matrix pattern per character. Alternate rows and columns are read out of each field to produce an interleaved and rounded charater.

A display row contains a maximum of 32 characters plus a leading and trailing blank box, each a character cell in width, making the overall width of a display row 34 x 8 = 272 dots. Successive display rows are butted together so that the total display occupies 195 dots high. The black box is 34 character cells wide by 195 dots high resulting in a box size of 45.018 usec in width by 195 tv scan lines in height. When centered in the video display, this box will start 13 usec after the leading edge of H in-scan line 43 and extend-to-scan line 237. This places the display within the safe title area for NTSC receivers. Character width is 42.37 usec also centered on the screen, resulting in a leading and trailing 1.32 usec black border.

Text Mode Display

When TEXT mode, in either language, has been selected (and valid Line 21 code has been detected in the incoming video) the 15 row by 34 character black box appears. Received TEXT characters are displayed as they are received starting in the top row. Successive carriage returns (new line command) moves the display row/column pointer down successive rows until all 15 display rows have been used. Thereafter, the text scrolls up as new characters are added to the bottom row.

If the data for the selected channel is interrupted by a command for another channel, data processing stops but the display remains. When a Resume Text command is received, data processing will resume and the new characters are added starting at the position that the display row/column pointer was prior to the interruption of data processing. If a Start Text command is received, the display is cleared and new characters are displayed starting in row 1 (top) column 1 (left side)

When scrolling, the display will shift one scan line per frame until a complete row has been scrolled. If a carriage return is received before scrolling is complete, the display immediately completes the "scroll" by jumping up the remaining scan lines and starts displaying the new text.

There will never be any transparent boxes in the TEXT display.

Caption Mode Display

When Caption mode, in either language, has been selected, the screen is transparent (display box disappears). Caption data only appears in the top four rows, rows 1-4, and/or in the bottom four rows, rows 12-15, in any combination. The form of the caption display depends on the caption mode indicated by the transmitted caption command, *Pop-on*, *Paint-on* or *Roll-up*.

Pop-on captions work with two caption memories. One of them is always being displayed while the other is being used to accumulate new caption data. A new caption is popped-on by swapping the two memories (the show caption command). When the on-screen memory is erased, the screen is blank (transparent) and the memory defaults to the row/column pointer at row 1, column 1 and monochrome non-underlined.

When caption mode is selected, the decoder processes any data following the Resume Caption Loading (RCL) command (or the Show Caption command). Normally, this command is followed by a Precode to indicate the row, column and character attributes to be used with the following data. If no Precode is received the data is added to the location last indicated by the row/column pointer prior to the receipt of the RCL command and with the character attributes previously assigned.

Paint-on caption mode is essentially equivalent to the Pop-on mode except that the data received after the Resume Direct Loading (RDL) command is written to the on-screen memory rather than the off-screen memory. All the rules for Precodes, Midcodes, etc., are otherwise the same.

Roll-up caption mode presents a "text" like display that is limited to the bottom 2, 3 or 4 rows depending on the Resume Roll-up (RRn) command used. In this case, a black box does not appear until characters are being displayed and the box is only wide enough to provide a leading and trailing box in each line. The new data appears in the bottom row and as each carriage return is received, the row scrolls up and the new data added to the bottom. When the number of rows indicated by the Resume command has been reached, the data in the top row scrolls off as new data is added to the bottom.

In all the caption display modes "black box" is dropped whenever there are three or more consecutive spaces. The number of boxes dropped are two less than the total number of consecutive spaces so that the two characters that are separated by the transparent space will both have adjoining boxes.

Display Erase and Autoblanking

The display is erased in the TEXT mode by the Start Text command (but box is maintained) and in the Caption mode by the Erase Displayed Memory command. The non-displayed memory is erased by the Erase Non-displayed Memory command.

Three other events can also cause the display to be erased. First, changing the data channel for processing by switching between CAPTIONs and TEXT, or between LANGUAGEs I and II, clears the memory and hence the display. Secondly, if the autoblanking circuit is activated

by the loss of valid code, then the display is turned off and the memory cleared. Lastly, in the Caption mode, if no valid caption command in the selected language is received for a 16 second period, the on-screen memory is erased.

The autoblanking circuit maintains the status of the presence of valid data. The decoder is held in the Decoder off (TV) state until data is continuously detected for a period of 0.5 seconds. Once the valid data decision has been made, and assuming that the user has selected the Decoder on state, the normal display for the data channel selected is presented.

The autoblanking circuit will not be activated again until valid data has been lost for 1.5 seconds. Any valid data received during the 1.5 second period resets the counter so that the autoblanking will only be activated on continuous loss of data for 1.5 seconds.

BLOCK DIAGRAM DESCRIPTION

The Z86128 is designed to provide the functional performance of a Line 21 Closed-Caption decoder with only two input signals being required, Composite Video and H Flyback. The Decoder performs two basic functions, namely extracting the Line 21 code from the incoming video and converting the recovered code, for the channel selected, into displayable information. Figure 3 shows the Z86128's block diagram.

The Z86128 generates its own H and V sync signals so that all internal processing is performed with noise immune signals. The decoder design has been formulated to achieve the best performance at the lowest possible cost.

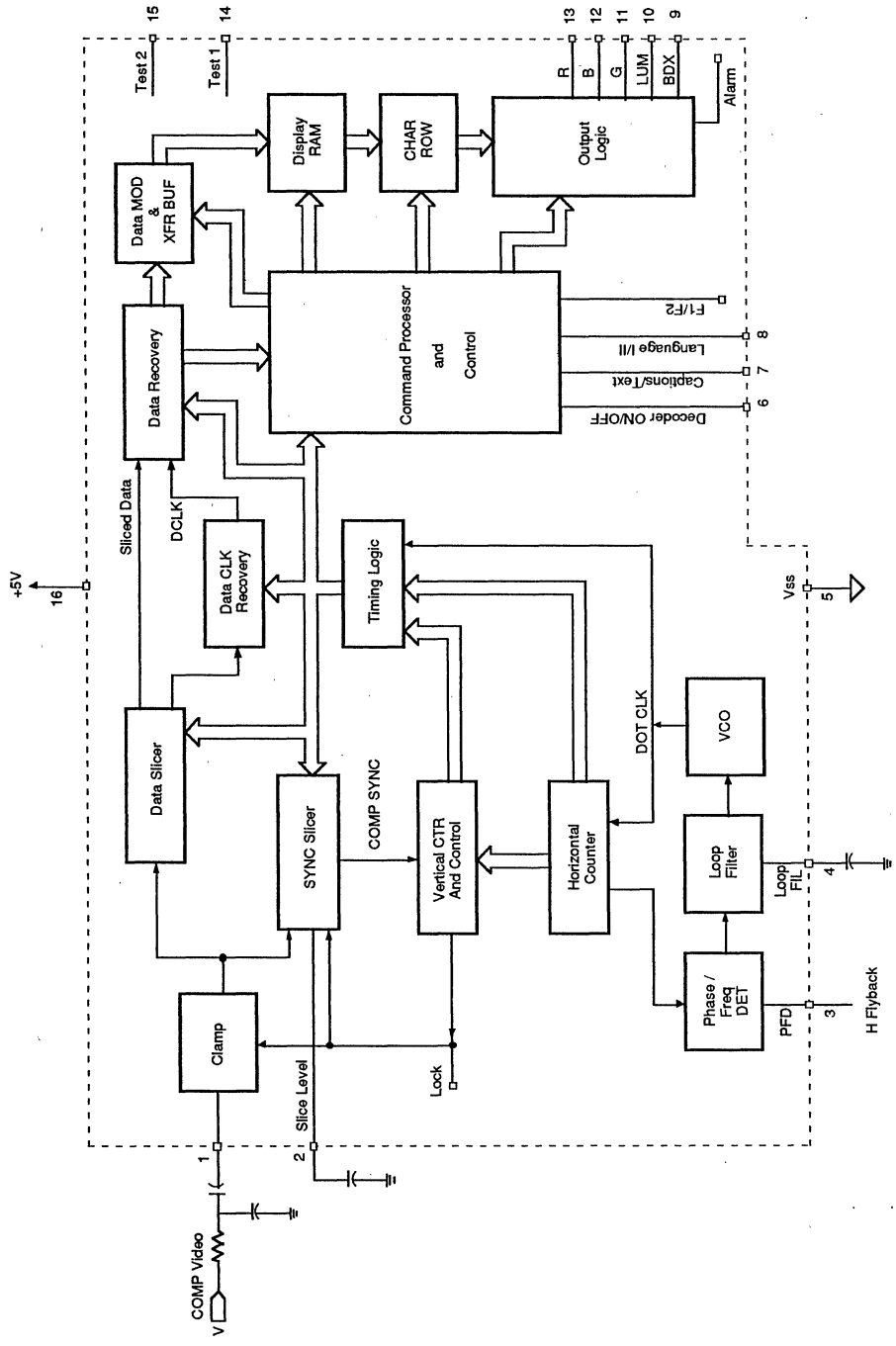


Figure 3. Z86128 Block Diagram

Input Signals

The Composite Video input should be a signal which is nominally 1 V p-p with sync tips negative and band limited to 600 KHz. The Z86128 operates with in an input level variation of 0.5 to 2.0 v p-p.

H Flyback is a TTL level input signal which provides horizontal sync information for the Phase/Frequency detector. It maintains a coarse lock of the VCO whether composite video is present of not. This signal can also be used in future applications to provide horizontal timing information in the absence of video. It can be positive or negative polarity.

Video Input Signal Processing

The Comp Video input is AC coupled to the IC and the sync tip is internally clamped to a fixed reference voltage. Initially, the signal is clamped using a simple clamp, but improved impulse noise performance is achieved once the internal sync circuits lock to the incoming signal. Noise rejection is obtained by making the clamp operative only during the sync tip. The clamped composite video signal is fed to both the Data Slicer and Sync Slicer blocks.

The Data Slicer generates a clean TTL level data signal by slicing the signal at its midpoint. The slice level is established on an adaptive basis during Line 21 of the odd field. The resultant value is stored until the next odd field Line 21 begins. A high level of noise immunity is achieved by using this process.

The Data Clock Recovery circuit produces a 32H clock signal (DCLK) that is locked in phase to the sliced clock run-in burst obtained from the Data Slicer. The Dot Clock is locked in phase with H sync but the DCLK phase is not determined until occurrence of Line 21 data. When Line 21 code appears, DCLK phase lock is achieved during the clock run-in burst and used to reclock the sliced data. Once phase lock is established it is maintained until a change in video signal occurs.

The Sync Slicer processes the clamped Comp Video signal to extract Comp Sync, which is then used to lock the internally generated sync to the incoming video. Sync slicing is performed in two steps. Initially, the sync is sliced at a fixed offset level from the sync tip. When the internal vertical counter locks, the slice level reference switches from a fixed to an adaptive basis. An external capacitor stores the slice level.

Timing and Synchronizing Circuits

All internal timing and synchronizing signals are derived from the on-board 12 MHz VCO. Its output is the Dot Clk signal used to drive the Horizontal and Vertical counter chains and for display timing. No external components are required to bring it within the pull-in range of the Phase/Frequency detector.

The Horizontal Counter is a divide by 768 circuit with intermediate outputs needed to generate the timing logic signals used in data recovery and data output (display). It produces pulse signals at 1H, 2H, 32H and 48H rates as well as the horizontal square wave, Q768, that is used to phase lock the VCO.

The Vertical Counter and Control circuits produce a noise free vertical pulse by dividing the horizontal signal in a 525 counter. The internal synchronizing signals are phased up with the incoming video by comparing the internally generated vertical pulse to an input vertical pulse derived from the Comp Sync signal provided by the Sync Slicer. When proper phasing has been established, this circuit outputs the LOCK signal which is used to provide additional noise immunity to the slicing circuits.

The LOCKed state is established only after several successive fields have occurred in which these two vertical pulses remain in sync. Once LOCKed, the internal timing will flywheel until such time as the two vertical pulses lose coincidence for a number of consecutive fields. Until LOCK is established, the decoder operates on a pulse for pulse basis.

Data Recovery

The Data Recovery circuits perform the initial processing of the data in Line 21. The sliced data is relocked using DCLK and the relocked data stream is checked for the presence of valid data. When data is present the two bytes are clocked into the Serial /Parallel register and output in parallel form.

This block checks the bytes for valid (odd) parity. It also determines whether the recovered byte pair is a repeat of the previously received byte pair. That information is used with the redundancy flag in the Command Processor to determine whether the command should be executed or not.

Command Processor

The Command Processor circuits control the manipulation of the data for storage and display. It decodes the control inputs (Decoder ON/OFF, Captions/Text, LANGUAGE I/II) to determine the display status desired and the data channel selected. This information is then used to perform its most important function, the control of the loading, addressing and clearing of the Display RAM.

During data recovery time (TV lines 21-42), the Command Processor transfers only the data received for the data channel selected, to the RAM for storage and display. In those cases such as special characters, midcodes, parity errors etc., where the data stored or action to be taken is different than the specific bytes received, the Command Processor converts the input data to the appropriate form.

During the display time (line 43-237), the Command Processor controls the operations of the Display RAM, Character ROM and output Logic circuits.

Memory and Display Circuits

These circuits operate together to generate the output color signals R, G, B and the monochrome signals Luminance and Box. The character ROM contains the dot pattern for all the characters but not the underline characteristic. The output logic provides the hardware underline control circuits and the Italics slant generator. The smooth scroll display control is also performed in the output logic block.

Z765A FDC
Floppy Disk Controller

October 1988

FEATURES

Address Mark detection circuitry internal to the FDC simplifies the phase locked loop and read electronics. The track stepping rate, head load time, and head unload time are user-programmable.

Z765A features are:

- IBM-compatible format, Single and Double Density
- Multisector and multitrack transfer capability
- Data scan capability—scans a single sector or an entire cylinder comparing byte-for-byte host memory and disk data
- Drives up to 4 floppy-disk drives (FDD)
- Data transfers in DMA or non-DMA mode
- Parallel seek operations on up to four drives
- Compatible with most general-purpose microprocessors
- Single phase 8 MHz clock
- +5V Only
- **40-Pin Dual-In-Line (DIP) package, 44-Pin plastic chip carrier (PLCC) package.**

GENERAL DESCRIPTION

The Z765A is an LSI Floppy Disk Controller (FDC) chip which contains the circuitry and control functions for interfacing a processor to four floppy-disk drives. It supports IBM System 3740 Single Density format (FM) and IBM System 34 Double Density format (MFM) including double-sided recording. The Z765A provides control signals which simplify the design of an external phase locked loop and write precompensation circuitry. The FDC simplifies and handles most of the burdens associated with implementing a floppy-disk interface. (Figure 1).

Handshaking signals make DMA operation easily incorporated with the aid of an external DMA Controller chip, such as the Z80 DMA. The FDC operates in either the DMA or non-DMA mode. In the non-DMA mode the FDC generates interrupts to the processor every time a data byte is to be transferred. In the DMA mode, the processor need only load the command into the FDC and all data transfers occur under control of the FDC and DMA controllers.

The Z765A executes 15 commands; each command requires multiple 8-bit bytes to fully specify the operation which the processor wishes the FDC to perform. The commands are:

- READ DATA
- WRITE DATA
- WRITE DELETED DATA
- READ DELETED DATA
- READ TRACK
- READ ID
- FORMAT TRACK
- SCAN EQUAL
- SCAN HIGH OR EQUAL
- SCAN LOW OR EQUAL
- SEEK
- RECALIBRATE
- SENSE INTERRUPT STATUS
- SPECIFY
- SENSE DRIVE STATUS

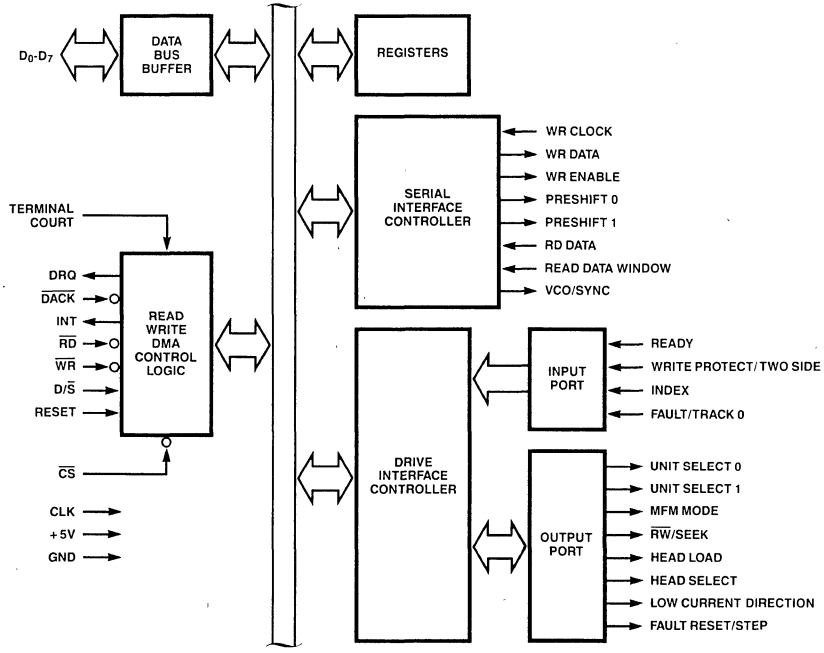


Figure 1. Z765A FDC Block Diagram

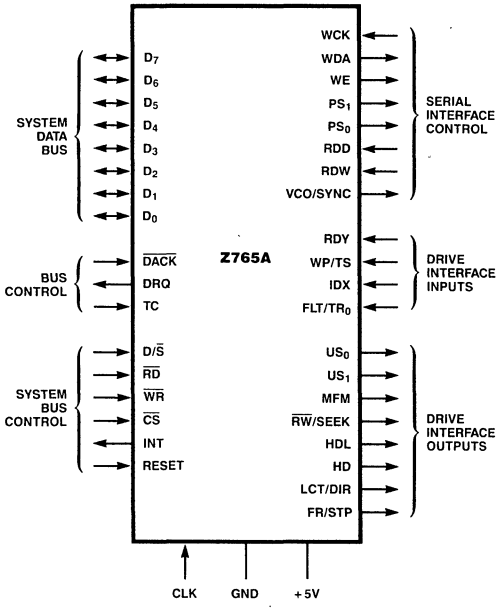


Figure 2. Pin Functions

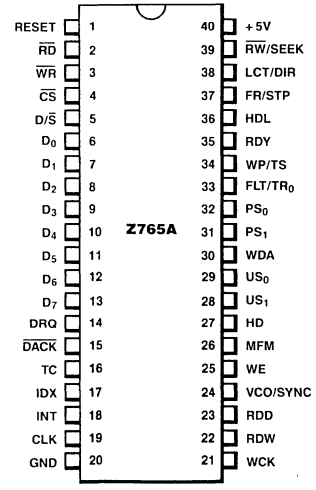


Figure 3a. Pin Assignments

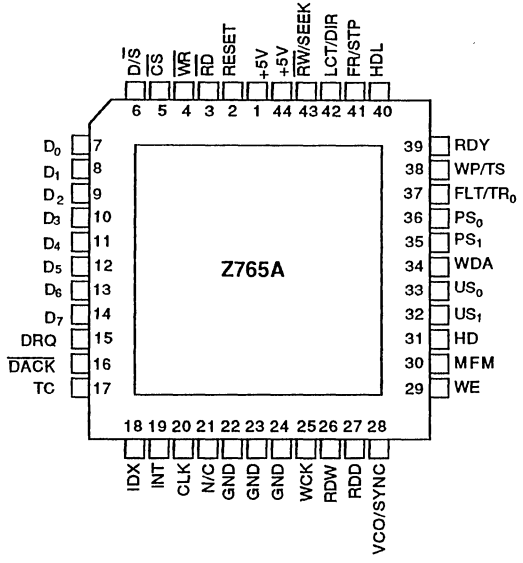


Figure 3b. Pin Assignments

PIN DESCRIPTIONS (Figures 2 and 3)

CLK. *Clock* (input). Single phase 8MHz square wave clock.

$\overline{CS}$. *Chip Select* (input). IC selected when 0 (Low), allowing $\overline{RD}$ and $\overline{WR}$ to be enabled.

D_0-D_7. *Data Bus.* Bidirectional 8-bit Data Bus. Disabled when $\overline{CS}$ = 1.

$\overline{DACK}$. *DMA Acknowledge* (input). DMA cycle is active when 0, and controller is performing DMA transfer.

DRQ. *Data DMA Request* (output). DMA Request is being made by FDC when DRQ = 1.

D/$\overline{S}$. *Data/Status Register Select* (input). Selects Data Register (D/$\overline{S}$ = 1) or Status Register (D/$\overline{S}$ = 0) contents of the FDC to be sent to Data Bus. Disabled when $\overline{CS}$ = 1.

FR/STP. *Fault Reset/Step* (output). Resets fault FF in FDD in Read/Write mode, contains step pulses to move head to another cylinder in Seek mode.

FLT/TR_0. *Fault/Track 0* (input). Senses FDD fault condition in Read/Write mode and Track 0 condition in Seek mode.

HD. *Head Select* (output). Head 1 selected when 1 (High); Head 0 selected when 0 (Low).

HDL. *Head Load* (output). Command which causes read/write head in FDD to contact diskette.

IDX. *Index* (input). Indicates the beginning of a disk track.

INT. *Interrupt* (output). Interrupt Request generated by FDC.

LCT/DIR. *Low Current/Direction* (output). Lowers Write current on inner tracks in Read/Write mode; determines direction head will step in Seek mode. A fault reset pulse is issued at the beginning of each Read or Write command prior to the occurrence of the Head Load signal.

MFM. *MFM Mode* (output). MFM mode when 1; FM mode when 0.

PS_1, PS_0. *Precompensation (preshift)* (output). Write precompensation status during MFM mode. Determines early, late, and normal times.

$\overline{RD}$. *Read* (input). When 0, control signal for transfer of data from FDC to Data Bus. Disabled when $\overline{CS}$ = 1.

RDD. *Read Data* (input). Read data from FDD, containing clock and data bits.

RDW. *Read Data Window* (input). Generated by PLL, and used to sample data from FDD.

RDY. *Ready* (input). Indicates FDD is ready to send or receive data.

RESET. *Reset* (input). Places FDC in idle state. Resets output lines to FDD to 0. Does not affect SRT, HUT or HLT in Specify command. If RDY pin is held High during Reset, FDC generates an interrupt within 1.024 msec. To clear this interrupt use Sense Interrupt Status command.

$\overline{RW}$/SEEK. *Read Write/Seek* (output). When 1 (High) Seek mode selected; when 0 (Low) Read/Write mode selected.

TC. *Terminal Count* (input). Indicates the termination of a DMA transfer when 1 (High). It terminates data transfer during Read/Write/Scan command in DMA or Interrupt mode.

US_1, US_0. *Unit Select* (output). FDD Unit selected.

VCO/SYNC. (output). Inhibits VCO in PLL when 0 (Low); enables VCO when 1.

WCK. *Write Clock* (input). Write data rate to FDD. FM = 500 KHz, MFM = 1 MHz with a pulse width of 250 ns for both FM and MFM.

WDA. *Write Data* (output). Serial clock and data bits to FDD.

WE. *Write Enable* (output). Enables write data into FDD.

WP/TS. *Write Protect/Two Side* (input). Senses Write Protect status in Read/Write mode and Two-Side Media in Seek mode.

$\overline{WR}$. *Write* (input). When 0, control signal for transfer of data to FDC via Data Bus. Disabled when $\overline{CS}$ = 1.

Table 1. Internal Registers

The bits in the Main Status Register are defined as follows:

Bit No.	Name	Symbol	Description
D_0	FDD 0 Busy	D_0B	FDD number 0 is in the Seek mode. If any bit is set, FDC will not accept read or write command.
D_1	FDD 1 Busy	D_1B	FDD number 1 is in the Seek mode. If any bit is set, FDC will not accept read or write command.
D_2	FDD 2 Busy	D_2B	FDD number 2 is in the Seek mode. If any bit is set, FDC will not accept read or write command.
D_3	FDD 3 Busy	D_3B	FDD number 3 is in the Seek mode. If any bit is set, FDC will not accept read or write command.
D_4	FDC Busy	CB	A read or write command is in process. FDC will not accept any other command.
D_5	Execution Mode	EXM	This bit is set only during execution phase in non-DMA mode. When D_5 goes low, execution phase has ended and result phase has started. It operates only during non-DMA mode of operation.
D_6	Data Input/Output	DIO	Indicates direction of data transfer between FDC and Data Register. If DIO = 1, then transfer is from Data Register to the processor. If DIO = 0, transfer is from the processor to Data Register.
D_7	Request for Master	RQM	Indicates Data Register is ready to send or receive data to or from the processor. Both bits DIO and RQM should be used to perform the handshaking functions of "ready" and "direction" to the processor.

INTERNAL REGISTERS

The Z765A contains two registers which may be accessed by the main system processor: a Status register and a Data register. The 8-bit Main Status register (Table 1) contains the FDC status information and may be accessed at any time. The 8-bit Data register is several registers in a stack; one register at a time is presented to the data bus. The Data register stores data, commands, parameters, and FDD status information. Data bytes are read out of, or written into, the Data register in order to program or obtain the results after a particular command. Only the Status register may be read and used to facilitate the transfer of data between the processor and Z765A.

The relationship between the Status/Data registers and the signals $\overline{RD}$, $\overline{WR}$, and D/$\overline{S}$ is shown in Table 2.

The Data Input/Output (DIO) and Request for Master (RQM) bits in the Status register indicate when data is ready and the direction transfer on the data bus (Figure 4). The maximum time between the last $\overline{RD}$ or $\overline{WR}$ during a command or result phase and the set or reset DIO and RQM is 12μs; every time the Main Status register is read the CPU should wait 12μs. The maximum time from the trailing edge of the last $\overline{RD}$ in the result phase to when D_4 (FDC busy) goes Low is 12μs.

Table 2. Relationships Between Status/Data Registers and $\overline{RD}$, $\overline{WR}$, and D/$\overline{S}$

D/$\overline{S}$	$\overline{RD}$	$\overline{WR}$	Function
0	0	1	Read Main Status Register
0	1	0	Illegal
0	0	0	Illegal
1	0	0	Illegal
1	0	1	Read from Data Register
1	1	0	Write into Data Register

STATUS REGISTER IDENTIFICATION

Bit No.	Name	Symbol	Description
			Status Register 0
D_7 D_6	Interrupt Code	IC	$D_7 = 0$ and $D_6 = 0$ Normal Termination of command, (NT). Command was completed and properly executed.
			$D_7 = 0$ and $D_6 = 1$ Abnormal Termination of command, (AT). Execution of command was started but was not successfully completed.
			$D_7 = 1$ and $D_6 = 0$ Invalid Command issue, (IC). Command which was issued was never started.
			$D_7 = 1$ and $D_6 = 1$ Abnormal Termination because during command execution the ready signal from FDD changed state.
D_5	Seek End	SE	When the FDC completes the SEEK command, this flag is set to 1 (High).
D_4	Equipment Check	EC	If a fault signal is received from the FDD, or if the Track 0 signal fails to occur after 77 step pulses (Recalibrate Command) then this flag is set.
D_3	Not Ready	NR	When the FDD is in the not-ready state and a read or write command is issued, this flag is set. If a read or write command is issued to Side 1 of a single-sided drive, then this flag is set.
D_2	Head Address	HD	This flag is used to indicate the state of the head at Interrupt.
D_1	Unit Select 1	US_1	This flag is used to indicate a Drive Unit Number at Interrupt.
D_0	Unit Select 0	US_0	This flag is used to indicate a Drive Unit Number at Interrupt.
			Status Register 1
D_7	End of Cylinder	EN	When the FDC tries to access a sector beyond the final sector of a cylinder, this flag is set.
D_6			Not used. This bit is always 0 (Low).
D_5	Data Error	DE	When the FDC detects a Cyclic Redundancy Check (CRC) error in either the ID field or the data field, this flag is set.
D_4	Overrun	OR	If the FDC is not serviced by the host system during data transfers within a certain time interval, this flag is set.
D_3			Not used. This bit always 0 (Low).
D_2	No Data	ND	During execution of READ DATA, WRITE DELETED DATA or SCAN command, if the FDC cannot find the sector specified in the Internal Data Register (IDR), this flag is set.
			During execution of the READ ID command, if the FDC cannot read the ID field without an error, then this flag is set.
			During execution of the READ A cylinder command, if the starting sector cannot be found, then this flag is set.

STATUS REGISTER IDENTIFICATION (Continued)

Bit No.	Name	Symbol	Description
			Status Register 1 (Continued)
D_1	Not Writeable	NW	During execution of WRITE DATA, WRITE DELETED DATA or Format A cylinder command, if the FDC detects a write protect signal from the FDD, then this flag is set.
D_0	Missing Address Mark	MA	If the FDC cannot detect the ID Address Mark after encountering the index hole twice, then this flag is set. If the FDC cannot detect the Data Address Mark or Deleted Data Address Mark, this flag is set. Also at the same time, the MD (Missing Address Mark in data field) of Status register 2 is set.
			Status Register 2
D_7			Not used. This bit is always 0 (Low).
D_6	Control Mark	CM	During execution of the READ DATA or SCAN command, if the FDC encounters a sector which contains a Deleted Data Address Mark, this flag is set.
D_5	Data Error in Data Field	DD	If the FDC detects a CRC error in the data field then this flag is set.
D_4	Wrong Cylinder	WC	This bit is related to the ND bit, and when the contents of Cylinder (C) on the medium is different from that stored in IDR, this flag is set.
D_3	Scan Equal Hit	SH	During execution of the SCAN command, if the condition of "equal" is satisfied, this flag is set.
D_2	Scan Not Satisfied	SN	During execution of the SCAN command, if the FDC cannot find a sector on the cylinder which meets the condition, then this flag is set.
D_1	Bad Cylinder	BC	This bit is related to the ND bit, and when the contents of C on the medium is different from that stored in the IDR and the contents of C is FF_H, then this flag is set.
D_0	Missing Address Mark in Data Field	MD	When data is read from the medium, if the FDC cannot find a Data Address Mark or Deleted Data Address Mark, then this flag is set.
			Status Register 3
D_7	Fault	FT	This bit is used to indicate the status of the Fault signal from the FDD.
D_6	Write Protected	WP	This bit is used to indicate the status of the Write Protected signal from the FDD.
D_5	Ready	RY	This bit is used to indicate the status of the Ready signal from the FDD.
D_4	Track 0	T0	This bit is used to indicate the status of the Track 0 signal from the FDD.
D_3	Two Side	TS	This bit is used to indicate the status of the Two Side signal from the FDD.
D_2	Head Address	HD	This bit is used to indicate the status of the Side Select signal to the FDD.
D_1	Unit Select 1	US_1	This bit is used to indicate the status of the Unt Select 1 signal to the FDD.
D_0	Unit Select 0	US_0	This bit is used to indicate the status of the Unit Select 0 signal to the FDD.

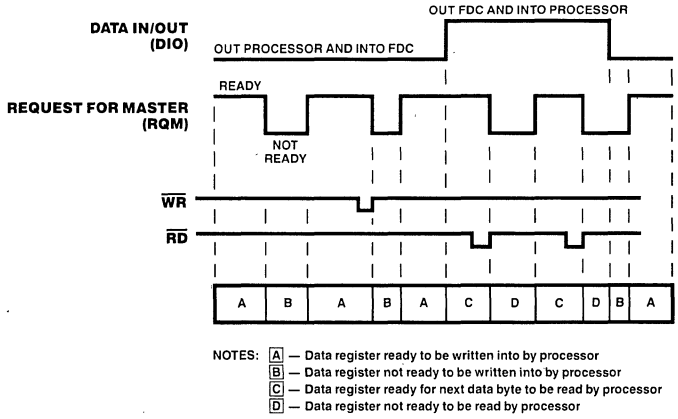


Figure 4. Data Transfer

COMMAND SEQUENCE

The Z765A is capable of performing 15 different commands. Each command is initiated by a multibyte transfer from the processor; the result after execution of the command may also be a multibyte transfer back to the processor. Because of this multibyte interchange of information between the Z765A and the processor, each command consists of three phases:

Command Phase. The FDC receives all information required to perform a particular operation form the processor.

Execution Phase. The FDC performs the operation it was instructed to do.

Result Phase. After completion of the operation, status and other housekeeping information are made available to the processor.

The Instruction set shows the required preset parameters and results for each command. Most commands require 9 command bytes and return 7 bytes during the result phase. The W to the left of each byte indicates a command phase byte to be written; an R indicates a result byte.

PROCESSOR INTERFACE

During Command or Result phases the Main Status register must be read by the processor before each byte of information is written into, or read from, the Data register. Then the CPU should wait for 12μs before reading the Main Status register. Bits D_6 and D_7 in the Main Status register must be in a 0 and 1 state, respectively, before each byte of the command word may be written into the Z765A. Many of the commands require multiple bytes and, as a result, the Main Status register must be read prior to each byte transfer to the Z765A. During the Result phase, D_6 and D_7 in the Main Status register must both be 1's before reading each byte from the Data Register. Reading the Main Status register before each byte transfer to the Z765A is required only in the Command and Result phases, not during the Execution phase.

If the Z765A is in the non-DMA mode and reading data from FDD, then the receipt of each data byte is indicated by an interrupt signal on pin 18(INT = 1). The generation of a Read signal ($\overline{RD}$ = 0) or Write signal ($\overline{WR}$ = 0) will clear the interrupt and output the data onto the data bus. If the processor cannot handle interrupts fast enough (every 13μs for the MFM mode and 27μs for the FM mode), then it may poll the Main Status register and bit D_7 (RQM) functions as the interrupt signal. If a Write command is in process, the $\overline{WR}$ signal negates the reset to the interrupt signal.

In the non-DMA mode it is necessary to examine the Main Status register to determine the cause of the interrupt, since it could be a data interrupt or a command termination interrupt, either normal or abnormal. If the Z765A is in the

COMMAND SYMBOL DESCRIPTION

Symbol	Name	Description
D/$\overline{S}$	Data/Status Select	D/$\overline{S}$ controls selection of Main Status register (D/$\overline{S}$ = 0) or Data register (D/$\overline{S}$ = 1)
C	Cylinder Number	C stands for the current/selected cylinder (track) numbers 0 through 76 of the medium.
D	Data	D stands for the data pattern which is going to be written into a sector.
D_7-D_0	Data Bus	8-bit Data Bus, where D_7 stands for a most significant bit, and D_0 stands for a least significant bit.
DTL	Data Length	When N is defined as 00, DTL stands for the data length which users are going to read out or write into the sector.
EOT	End of Track	EOT stands for the final sector number on a cylinder. During Read or Write operations, FDC will stop data transfer after a sector number equal to EOT.
GPL	Gap Length	GPL stands for the length of Gap 3. During Read/Write commands this value determines the number of bytes that VCO/SYNC will stay low after two CRC bytes. During Format command it determines the size of Gap 3.
H	Head Address	H stands for head number 0 or 1, as specified in ID field.
HD	Head	HD stands for a selected head number 0 or 1 and controls the polarity of pin 27. (H = HD in all command words.)
HLT	Head Load Time	HLT stands for the head load time in the FDD (2 to 254 ms in 2 ms increments).
HUT	Head Unload Time	HUT stands for the head unload time after a Read or Write operation has occurred (16 to 240 ms in 16 ms increments).
MF	FM or MFM Mode	If MF is Low, FM mode is selected, and if it is High. MFM mode is selected.
MT	Multitrack	If MT is high, a Multitrack operation is performed. If MT = 1 after finishing Read/Write operation on side 0, FDC automatically starts searching for sector 1 on side 1.
N	Number	N stands for the Number of data bytes written in a sector.
NCN	New Cylinder Number	NCN stands for a New Cylinder Number or desired position of head which is going to be reached as a result of the Seek operation.
ND	Non-DMA Mode	ND stands for operation in the Non-DMA mode.
PCN	Present Cylinder Number	PCN stands for the cylinder number or present position of Head at the completion of Sense Interrupt Status command.
R	Record	R stands for the sector number which will be read or written.
R/W	Read/Write	R/W stands for either Read (R) or Write (W) signal.
SC	Sector	SC indicates the number of Sectors per Cylinder.
SK	Skip	SK stands for Skip Deleted Data Address mark.
SRT	Step Rate Time	SRT stands for the Stepping Rate for the FDD (1 to 16 ms in 1 ms increments). Stepping Rate applies to all drives ($F_{(16)}$ = 1 ms, $E_{(16)}$ = 2 ms, $D_{(16)}$ = 3 ms, . . .).
ST0 ST1 ST2 ST3	Status 0 Status 1 Status 2 Status 3	ST0-3 stands for one of four registers which store the status information after a command has been executed. This information is available during the result phase after command execution. These registers should not be confused with the main status register (selected by D/$\overline{S}$ = 0). ST0-3 may be read only after a command has been executed and contains information relevant to that particular command.
STP	Step	During a Scan operation, if STP = 1, the data in contiguous sectors is compared byte by byte with data sent from the processor (or DMA); if STP = 2, then alternate sectors are read and compared.
US_0, US_1	Unit Select	Used to select between drives 0-3.

INSTRUCTION SET[1, 2]

Phase	R/W	Data Bus D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Remarks
					Read Data					
Command	W	MT	MF	SK	0	0	1	1	0	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	See Note 3
	W	C								Sector ID information prior to
	W	H								command execution. The 4 bytes
	W	R								are commanded against header
	W	N								on Floppy disk.
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data transfer between the FDD and main system
Result	R	ST0								Status information after command
	R	ST1								execution
	R	ST2								
	R	C								Sector ID information after
	R	H								command execution
	R	R								
	R	N								
					Read Deleted Data					
Command	W	MT	MF	SK	0	1	1	0	0	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
	W	C								Sector ID information prior to
	W	H								command execution. The 4 bytes
	W	R								are commanded against header
	W	N								on Floppy Disk.
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data transfer between the FDD and main system
Result	R	ST0								Status information after command
	R	ST1								execution
	R	ST2								
	R	C								Sector ID information after
	R	H								command execution
	R	R								
	R	N								

NOTES: 1. Symbols used in this table are described at the end of this section.
2. D/$\overline{S}$ should equal binary 1 for all operations.
3. X = Don't care, usually made to equal binary 0.

INSTRUCTION SET[1, 2] (Continued)

Phase	R/W	Data Bus D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Remarks
Write Data										
Command	W	MT	MF	0	0	0	1	0	1	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
	W	C								Sector IC information prior to command execution. The 4 bytes are commanded against header on Floppy Disk.
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data transfer between the main system and FDD
Result	R	ST0								Status information after command execution
	R	ST1								
	R	ST2								
	R	C								Sector ID information after command execution.
	R	H								
	R	R								
	R	N								
Write Deleted Data										
Command	W	MT	MF	0	0	1	0	0	1	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
	W	C								Sector ID information prior to command execution. The 4 bytes are commanded against header on Floppy disk.
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data transfer between the FDD and main system
Result	R	ST0								Status information after command execution
	R	ST1								
	R	ST2								
	R	C								Sector ID information after command execution
	R	H								
	R	R								
	R	N								

NOTES: 1. Symbols used in this table are described at the end of this section.
2. D/$\overline{S}$ should equal binary 1 for all operations.
3. X = Don't care, usually made to equal binary 0.

INSTRUCTION SET[1, 2] (Continued)

Phase	R/W	Data Bus D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Remarks
Read A Track										
Command	W	0	MF	SK	0	0	0	1	0	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
	W	C								Sector ID information prior to command execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data transfer between the FDD and main system. FDC reads all data fields from index hole to EOT.
Result	R	ST0								Status information after command execution
	R	ST1								
	R	ST2								
	R	C								Sector ID information after command execution
	R	H								
	R	R								
	R	N								
Read ID										
Command	W	0	MF	0	0	1	0	1	0	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
Execution										The first correct ID information on the cylinder is stored in Data Register.
Result	R	ST0								Status information after command execution
	R	ST1								
	R	ST2								
	R	C								Sector ID information read during Execution phase from Floppy Disk.
	R	H								
	R	R								
	R	N								

NOTES: 1. Symbols used in this table are described at the end of this section.
2. D/$\overline{S}$ should equal binary 1 for all operations.
3. X = Don't care, usually made to equal binary 0.

INSTRUCTION SET[1, 2] (Continued)

Phase	R/W	Data Bus D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Remarks
Format A Track										
Command	W	0	MF	0	0	1	1	0	1	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
	W	N								Bytes Sector
	W	SC								Sectors/Track
	W	GPL								Gap 3
	W	D								Filler byte
Execution										FDC formats an entire track.
Result	R	ST0								Status information after command
	R	ST1								execution
	R	ST2								
	R	C								In this case, the ID information
	R	H								has no meaning.
	R	R								
	R	N								
Scan Equal										
Command	W	MT	MF	SK	1	0	0	0	1	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
	W	C								Sector ID information prior to
	W	H								command execution
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data compared between the FDD and the main system.
Result	R	ST0								Status information after command
	R	ST1								execution
	R	ST2								
	R	C								Sector ID information after
	R	H								command execution
	R	R								
	R	N								

NOTES: 1. Symbols used in this table are described at the end of this section.
2. D/$\overline{S}$ should equal binary 1 for all operations.
3. X = Don't care, usually made to equal binary 0.

INSTRUCTION SET[1, 2] (Continued)

Phase	R/W	Data Bus D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Remarks
Scan Low or Equal										
Command	W	MT	MF	SK	1	1	0	0	1	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
	W	C								Sector ID information prior to command execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	STP								
Execution										Data compared between the FDD and main system
Result	R	ST0								Status information after command execution
	R	ST1								
	R	ST2								
	R	C								Sector ID information after command execution
	R	H								
	R	R								
	R	N								
Scan High or Equal										
Command	W	MT	MF	SK	1	1	1	0	1	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
	W	C								Sector ID information prior to command execution.
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	STP								
Execution										Data compared between the FDD and main system.
Result	R	ST0								Status information after command execution
	R	ST1								
	R	ST2								
	R	C								Sector ID information after command execution.
	R	H								
	R	R								
	R	N								
Recalibrate										
Command	W	0	0	0	0	0	1	1	1	Command Codes
	W	X	X	X	X	X	0	US_1	US_0	
Execution										Head retracted to Track 0

NOTES: 1. Symbols used in this table are described at the end of this section.
2. D/$\overline{S}$ should equal binary 1 for all operations.
3. X = Don't care, usually made to equal binary 0.

INSTRUCTION SET[1, 2] (Continued)

Phase	R/W	Data Bus D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Remarks
Sense Interrupt Status										
Command	W	0	0	0	0	1	0	0	0	Command Codes
Result	R	ST0								Status information about the FDC
	R	PCN								at the end of seek operation
Specify										
Command	W	0	0	0	0	0	0	1	1	Command Codes
	W	SRT							HUT	
	W	HLT							ND	
Sense Drive Status										
Command	W	0	0	0	0	0	1	0	0	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
Result	R	ST3								Status information about FDD
Seek										
Command	W	0	0	0	0	1	1	1	1	Command Codes
	W	X	X	X	X	X	HD	US_1	US_0	
	W	NCN								
Execution										Head is positioned over proper cylinder on diskette.
Invalid										
Command	W	Invalid Codes								Invalid Command Codes (NoOp—FDC goes into Standby state.)
Result	R	ST0								ST0 = $80_{(H)}$

NOTES: 1. Symbols used in this table are described at the end of this section.
2. D/$\overline{S}$ should equal binary 1 for all operations.
3. X = Don't care, usually made to equal binary 0.

DMA mode, no interrupts are generated during the Execution phase. The Z765A generates DRQs (DMA Requests) when each byte of data is available. The DMA Controller responds to this request with both a $\overline{DACK}$ (DMA Acknowledge) = 0 and an $\overline{RD}$ (Read signal) = 0. When the DMA Acknowledge signal goes Low ($\overline{DACK}$ = 0), then the DMA request is cleared (DRQ = 0). If a Write command has been issued, a $\overline{WR}$ signal appears instead of $\overline{RD}$. After the Execution phase has been completed [Terminal Count (TC) has occurred] or the last sector on the cylinder (EOT) read/written, then an interrupt occurs (INT = 1) which signifies the beginning of the Result phase. When the first byte of data is read during the Result phase, the interrupt is automatically cleared (INT = 0).

The $\overline{RD}$ or $\overline{WR}$ signals should be asserted while $\overline{DACK}$ is true. The $\overline{CS}$ signal is used in conjunction with $\overline{RD}$ and $\overline{WR}$ as a gating function during programmed I/O operations. $\overline{CS}$ has no effect during DMA operations. If the non-DMA mode is chosen, the $\overline{DACK}$ signal should be pulled up to V_{CC}.

During the Result phase all bytes shown in the Command Table must be read. For example, the Read Data command has seven bytes of data in the Result phase; all seven bytes must be read to successfully complete the Read Data command and allow the Z765A to accept a new command.

The Z765A contains five Status registers. The Main Status register can be read at any time by the processor. The other four Status registers (ST0, ST1, ST2, and ST3) are available only during the Result phase and can be read only after completing a command. The particular command that has been executed determines how many of the Status registers are read.

The bytes of data which are sent to the Z765A to form the Command phase and are read out of the Z765A in the Result phase must occur in the order shown in the Command Table. That is, the Command Code must be sent first and the other bytes sent in the prescribed sequence. No foreshortening of the Command or Result phases is allowed. After the last byte of data in the Command phase is sent to the Z765A, the Execution phase automatically starts. In a similar fashion, when the last byte of data is read out in the Result phase, the command is automatically ended and the Z765A is ready for a new command.

POLLING FEATURE OF THE Z765A

After Reset is sent to the Z765A, the Unit Select lines US_0 and US_1 automatically go into a polling mode (Figure 5). Between commands (and between step pulses in the Seek command) the Z765A polls all four FDDs looking for a change in the Ready line from any of the drives. If the Ready line changes state (usually due to a door opening or closing), then the Z765A generates an interrupt. When Status register 0 (ST0) is read (after Sense Interrupt Status is issued), Not Ready (NR) is indicated. The polling of the Ready line by the Z765A occurs continuously between commands, thus notifying the processor which drives are on or off line. Each drive is polled every 1.024 ms except during the Read/Write commands. When used with a 4 MHz clock for interfacing to minifloppies, the polling rate is 2.048 ms.

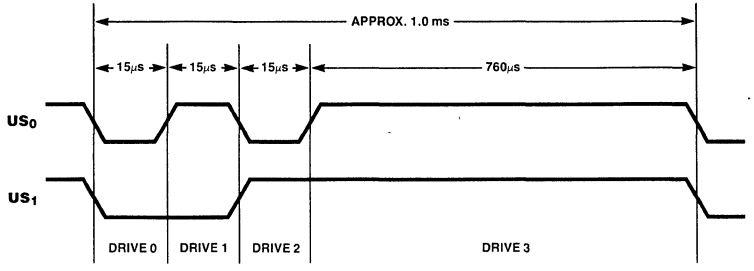


Figure 5. Polling Features

COMMANDS

Read Data

A set of nine (9) byte words are required to place the FDC into the Read Data Mode. After the Read Data command is issued, the FDC loads the head (if it is in the unloaded state), waits the specified head settling time (defined in the Specify command), and begins reading ID Address Marks and ID fields. When the current sector number (R) stored in the ID Register (IDR) compares with the sector number read off the diskette, then the FDC, via the data bus, outputs data byte-to-byte from the data field to the main system.

After completion of the read operation from the current sector, the Sector Number is incremented by one, and the data from the next sector is read and output on the data bus. This continuous read function is called a Multi-Sector Read Operation. The Read Data command can be terminated by the receipt of a TC signal which should be issued when the DACK for the last byte of data is sent. Upon receipt of this signal, the FDC stops outputting data to the processor, but continues to read data from the current sector, checks Cyclic Redundancy Count (CRC), and at the end of the sector, terminates the Read Data command. The amount of data which can be handled with a single command to the FDC depends upon multitrack (MT), MFM/FM (MF), and Number of Bytes/Sector (N). Table 3 shows the Transfer Capacity.

Table 3. Transfer Capacity

Multi-Track MT	MFM/FM MF	Bytes/Sector N	Maximum Transfer Capacity (Bytes/Sector) (Number of Sectors)	Final Sector Read from Diskettes
0	0	00	(128) (26) = 3,328	26 at Side 0
0	1	01	(256) (26) = 6,656	or 26 at Side 1
1	0	00	(128) (52) = 6,656	26 at Side 1
1	1	01	(256) (52) = 13,312	
0	0	01	(256) (15) = 3,840	15 at Side 0
0	1	02	(512) (15) = 7,680	or 15 at Side 1
1	0	01	(256) (30) = 7,680	15 at Side 1
1	1	02	(512) (30) = 15,360	
0	0	02	(512) (8) = 4,096	8 at Side 0
0	1	03	(1024) (8) = 8,192	or 8 at Side 1
1	0	02	(512) (16) = 8,192	8 at Side 1
1	1	03	(1024) (16) = 16,384	

MT allows the FDC to read data from both sides of the diskette. For a particular cylinder, data is transferred starting at Sector 1, Side 0 and completing at the last sector, Sector L, Side 1. This function pertains to only one cylinder (the same track) on each side of the diskette.

When N = 0, then DTL defines the data length which the FDC must treat as a sector. If DTL is smaller than the actual data length in a Sector, the data beyond DTL in the Sector is not sent to the Data Bus. The FDC internally reads the complete sector performing the CRC check and, depending upon the manner of command termination, may perform a Multi-Sector Read Operation. When N is non-zero, then DTL has no meaning and should be set to FF_H.

At the completion of the Read Data Command the head is unloaded, after the Head Unload Time Interval specified in the Specify Command has elapsed. If the processor issues another command before the head unloads, there is no head settling time between subsequent reads. This time saved is particularly valuable when a diskette is copied.

If the FDC twice detects the index hole without finding the right sector (R), then the FDC sets Status register 1's No Data (ND) flag to 1, and terminates the Read Data command. (Status register 0 also has bits 7 and 6 set to 0 and 1 respectively.)

After reading the ID and Data fields in each sector, the FDC checks the CRC bytes. If a read error is detected indicating incorrect CRC in the ID field, the FDC sets Status register 1's Data Error (DE) flag to 1, and if a CRC error occurs in the Data Field, the FDC also sets Status register 2's Data Error in Data Field (DD) flag to 1, and terminates the Read Data command. (Status register 0, bit 7 = 0, bit 6 = 1.)

If the FDC reads a Deleted Data Address Mark off the diskette, and the SK bit D in the first Command Word = 0, then the FDC sets Status register 2's Control Mark (CM) flag to 1, and after reading all the data in the sector, terminates the Read Data command. If SK = 1, the FDC skips the sector with the Deleted Data Address Mark and reads the next sector. When SK = 1, the CRC bits in the deleted data field are not checked.

During disk data transfers between the FDC and the processor, via the data bus, the FDC must be serviced by the processor every 27μs in the FM Mode, and every 13μs in the MFM Mode, or the FDC sets Status register 1's Overrun (OR) flag to 1, and terminates the Read Data command.

If the processor terminates a read or write operation in the FDC, then the ID information in the Result Phase is dependent upon the state of the MT bit and EOT byte. Table 4 shows the values for C, H, R, and N when the processor terminates the command.

Table 4. C, H, R, and N Values When Processor Terminates Commands

MT	HD	Final Sector Transferred to Processor	ID Information at Result Phase			
			C	H	R	N
0	0	Less than EOT	NC	NC	R+1	NC
	0	Equal to EOT	C + 1	NC	R = 01	NC
	1	Less than EOT	NC	NC	R + 1	NC
	1	Equal to EOT	C + 1	NC	R = 01	NC
1	0	Less than EOT	NC	NC	R + 1	NC
	0	Equal to EOT	NC	LSB	R = 01	NC
	1	Less than EOT	NC	NC	R + 1	NC
	1	Equal to EOT	C + 1	LSB	R = 01	NC

NOTES: NC (No Change): The same value as the one at the beginning of command execution.
LSB (Least Significant Bit): The least significant bit of H is complemented.

Write Data

A set of nine (9) bytes is required to set the FDC in the Write Data mode. After the Write Data command is issued, the FDC loads the head, waits the specified head setting time, and begins reading ID fields. When all four bytes (C, H, R, and N) loaded during the command match the four bytes of the ID field from the diskette, the FDC takes data from the processor byte-by-byte via the data bus and outputs it to the FDD.

After writing data into the current sector, the sector number stored in the R register is incremented by one, and new data is written into the next data field. The FDC continues this Multisector Write Operation until a Terminal Count signal is issued. If a Terminal Count signal is sent to the FDC, it continues writing into the current sector to complete the data field. If the Terminal Count signal is received while a data field is being written, the remainder of the data field is filled with zeros.

The FDC reads the ID field of each sector and checks the CRC bytes. If the FDC detects a read error (CRC error) in one of the ID fields, it sets Status register 1's DE flag to 1, and terminates the Write Data command. (Status register 0, bit 7 = 0, bit 6 = 1.)

The Write command operates in the same manner as the Read command for the following items:

- Transfer capacity
- End of cylinder (EN) flag
- No data (ND) flag
- Head unload time interval
- ID information when the processor terminates command
- Definition of DTL when N = 0 and when N ≠ 0

Refer to the Read Data command for details.

In the Write Data mode, data transfers between the processor and FDC via the data bus, must occur every 27μs in the FM mode and every 13μs in the MFM mode. If the time interval between data transfers is longer, then the FDC sets Status register 1's Overrun (OR) flag to 1, and terminates the Write Data command. (Status register 0, bit 7 = 0, bit 6 = 1.)

Write Deleted Data

This command is the same as the Write Data command except a Deleted Data Address mark, instead of the normal Data Address mark, is written at the beginning of the data field.

Read Deleted Data

This command is the same as the Read Data command except that when the FDC detects a Data Address mark at the beginning of a data field and SK = 0, the FDC reads all the data in the sector and sets Status register 2's CM flag to 1, and terminates the command. If SK = 1, then the FDC skips the sector with the Data Address mark and reads the next sector.

Read Track

This command is similar to the Read Data command except that this is a continuous Read operation where the entire data field from each of the sectors is read. Immediately after

sensing the index hole, the FDC starts reading all data fields on the track as continuous blocks of data. If the FDC finds an error in the ID or Data CRC check bytes, it continues to read data from the track. The FDC compares the ID information read from each sector with the value stored in the IDR and, if there is no comparison, sets Status register 1's ND flag to 1. Multitrack or skip operations are not allowed with this command.

This command terminates when the number of sectors read is equal to EOT. If the FDC does not find an ID Address mark on the diskette after it senses the index hole for the second time, it sets Status register 1's Missing Address mark (MA) flag to 1 and terminates the command. (Status Register 0, bit 7 = 0, bit 6 = 1.)

Read ID

The Read ID command gives the present position of the recording head. The FDC stores the values from the first ID field it can read. If no proper ID Address mark is found on the diskette before the index hole is encountered for the second time, Status register 1's MA flag is set to 1; if no data is found, Status register 1's No Data (ND) flag is set to 1. The command is then terminated with STO bit 7 = 0 and bit 6 = 1. During this command, data transfer between FDC and the CPU occurs only during the result phase.

Format Track

The Format command allows an entire track to be formatted. After the index hole is detected, data is written on the diskette; Gaps, Address marks, ID fields and data fields, all per the IBM 3740 Single Density format or IBM System 34 Double Density format, are recorded. The processor, during the command phase, supplies values i.e., Number of bytes/sector (N), Sectors Cylinder (SC), Gap Length (GPL), and Data Pattern (D) which determine the particular format to be written.

The data field is filled with the byte of data stored in D. The ID field for each sector is supplied by the processor; that is, four data requests per sector are made by the FDC for Cylinder number (C), Head number (H), Sector number (R), and Number of bytes/sector (N). This allows diskette formatting with nonsequential sector numbers.

The processor must send new values for C, H, R, and N to the Z765A for each sector on the track. If FDC is set for the DMA mode, it issues four DMA requests per sector. If it is set for the Interrupt mode, it issues four interrupts per sector and the processor must supply C, H, R, and N loads for each sector. The contents of the R register are incremented by 1 after each sector is formatted; thus, the R register contains a value of R when it is read during the Result phase. This incrementing and formatting continues for the whole track until the FDC detects the index hole for the second time, whereupon it terminates the command.

If the Fault signal is received from the FDD at the end of a Write operation, the FDC sets Status register 0's EC flag to 1 and terminates the command after setting Status register 0, bit 7 to 0 and bit 6 to 1. Also the loss of a Ready signal at the beginning of a command execution phase causes Status register 0, bit 7 and 6 to be set to 0 and 1 respectively.

Table 5 shows the sector size relationship between N, SC, and GPL.

Table 5. Functional Description of Commands

Format	Sector Size	N	SC	GPL[1]	GPL[2,3]
	8" Standard Floppy				
FM Mode	128 bytes sector	00	1A	07	1B
	256	01	0F	0E	2A
	512	02	08	1B	3A
	1024	03	04	47	8A
	2048	04	02	C8	FF
	4096	05	01	C8	FF
MFM Mode[4]	256	01	1A	0E	36
	512	02	0F	1B	54
	1024	03	08	35	74
	2048	04	04	99	FF
	4096	05	02	C8	FF
	8192	06	01	C8	FF
	5¼" Minifloppy				
FM Mode	128 bytes/sector	00	12	07	09
	128	00	10	10	19
	256	01	08	18	30
	512	02	04	46	87
	1024	03	02	C8	FF
	2048	04	01	C8	FF
MFM Mode[4]	256	01	12	0A	0C
	256	01	10	20	32
	512	02	08	2A	50
	1024	03	04	80	F0
	2048	04	02	C8	FF
	4096	05	01	C8	FF

NOTES:
1. Suggested values of GPL in Read or Write commands to avoid splice point between data field and ID field of contiguous sections.
2. Suggested values of GPL in format command.
3. All values except sector size are hexidecimal.
4. In MFM mode FDC cannot perform a Read/Write format operation with 128 bytes sector. (N = 00)

Scan Commands

The Scan commands allow comparison of data read from the diskette and data supplied from the main system. The FDC compares the data on a byte-by-byte basis and looks for a sector of data which meets the conditions of $D_{FDD} = D_{Processor}$, $D_{FDD} \leq D_{Processor}$, or $D_{FDD} \geq D_{Processor}$. The hexadecimal byte of FF from memory or from FDD can be used as a mask byte because it always meets the condition of the comparison. One's complement arithmetic is used for comparison (FF = largest number, 00 = smallest number). After a whole sector of data is compared, if the conditions are not met, the sector number is incremented (R + STP → R) and the scan operation continues until one of the following conditions occur: the conditions for scan are met (equal, low, or high), the last sector on the track is reached (EOT), or the terminal count (TC) signal is received.

If the conditions for scan are met, the FDC sets the Status register 2's Scan Hit (SH) flag to 1 and terminates the Scan command. If the conditions for scan are not met between the starting sector number (R) and the last sector on the cylinder (EOT), then the FDC sets Status register 2's Scan Not Satisfied (SN) flag to 1, and terminates the Scan command. During the scan operation, the receipt of a signal from the processor or DMA controller causes the FDC to complete the comparison of the particular byte in process and then to terminate the command. Table 6 shows the status of bits SH and SN under various conditions of Scan.

Table 6.

	Status Register 2		
Command	**Bit 2 = SN**	**Bit 3 = SH**	**Comments**
Scan Equal	0	1	$D_{FDD} = D_{Processor}$
	1	0	$D_{FDD} \neq D_{Processor}$
Scan Low or Equal	0	1	$D_{FDD} = D_{Processor}$
	0	0	$D_{FDD} < D_{Processor}$
	1	0	$D_{FDD} > D_{Processor}$
Scan High or Equal	0	1	$D_{FDD} = D_{Processor}$
	0	0	$D_{FDD} > D_{Processor}$
	1	0	$D_{FDD} < D_{Processor}$

If the FDC encounters a Deleted Data Address mark on one of the sectors and SK = 0, then it regards the sector as the last sector on the cylinder, sets Status register 2's Control Mark (CM) flag to 1 and terminates the command. If SK = 1, the FDC skips the sector with the Deleted Address mark, reads the next sector, and sets Status register 2's Control Mark (CM) flag to 1 to show that a Deleted sector has been encountered.

When either the Step (STP) (contiguous sectors = 01 or alternate sectors = 02) sectors are read or the Multitrack (MT) is programmed, the last sector on the track must be read. For example, if STP = 02, MT = 0, the sectors are numbered sequentially 1 through 26 and the Scan command is started at sector 21, the following happens. Sectors 21, 23, and 25 are read, then the next sector, 26, is skipped and the index hole is encountered before the EOT value of 26 can be read resulting in an abnormal termination of the command. If the EOT had been set at 25 or the scanning started at sector 20, then the Scan command would be completed in a normal manner.

During the Scan command, data is supplied by either the processor or DMA Controller for comparison against the data read from the diskette. In order to avoid having Status register 1's Overrun (OR) flag set, it is necessary to have the data available in less than 27μs (FM mode) or 13μs (MFM mode). If an Overrun occurs, the FDC ends the command with Status register 0, bit 7 cleared to 0 and bit 6 set to 1.

Seek

The Read/Write head within the FDD is moved from cylinder to cylinder under control of the Seek command. The FDC has four independent Present Cylinder registers for each drive which are cleared only after the Recalibrate command. The FDC compares the Present Cylinder Number (PCN) which is the current head position with the New Cylinder Number (NCN), and if there is a difference, performs the following operations:

PCN < NCN: Direction signal to FDD set to 1, and Step Pulses are issued. (Step In)

PCN > NCN: Direction signal to FDD cleared to 0, and Step Pulses are issued. (Step Out)

The rate at which Step pulses are issued is controlled by Stepping Rate Time (SRT) in the Specify command. After each Step pulse is issued NCN is compared against PCN, and when NCN = PCN, Status register 0's Seek End (SE) flag is set to 1, and the command is terminated. At this point FDC interrupt goes High. Bits D_0-D_3 in the Main Status register are set during the Seek operation and are cleared by the Sense Interrupt Status command.

During the command phase of the Seek operation the FDC is in the FDC Busy state, but during the execution phase it is in the Nonbusy state. While the FDC is in the Nonbusy state, another Seek command may be issued, and in this manner parallel Seek operations may be done on up to four drives at once. No other command can be issued for as long as the FDC is in the process of sending step pulses to any drive.

If an FDD is in a Not Ready state at the beginning of the command execution phase or during the Seek operation, then Status register 0's Not Ready (NR) flag is set to 1, and the command is terminated after bit 7 is set to 1 and bit 6 to 0.

If writing three bytes of Seek command exceeds 150μs, the timing between the first two step pulses may be 1ms shorter than that set in the Specify command.

Recalibrate

The function of this command is to retract the Read/Write head within the FDD to the Track 0 position. The FDC clears the contents of the PCN counter and checks the status of the Track 0 signal from the FDD. As long as the Track 0 signal is Low, the Direction signal remains 0 and step pulses are issued. When the Track 0 signal goes High, the Status register 0's SE flag is set to 1 and the command is terminated. If the Track 0 signal is still Low after 77 step pulses have been issued, the FDC sets Status register 0's SE and Equipment Check (EC) flags to 1s and terminates the command after Status register 0, bit 7 is cleared to 0 and bit 6 is set to 1.

The ability to do overlap Recalibrate commands to multiple FDDs and the loss of the Ready signal, as described in the Seek command, also applies to the Recalibrate command. If the Diskette has more than 77 tracks, the Recalibrate command should be issued twice, in order to position the Read/Write head to Track 0.

Sense Interrupt Status

An interrupt signal is generated by the FDC for one of the following reasons:

1. Upon entering the Result phase of command:
   - ☐ Read Data
   - ☐ Write Data
   - ☐ Write Deleted Data
   - ☐ Read Deleted Data
   - ☐ Read Track
   - ☐ Read ID
   - ☐ Format Track
   - ☐ Scan
2. Ready Line of FDD changes state
3. End of Seek or Recalibrate command
4. During Execution phase in the non-DMA mode

Interrupts caused by reasons 1 and 4 occur during normal command operations and are easily discernible by the processor. During an execution phase in non-DMA mode, D_5 in the Main Status Register is High. Upon entering the Result phase this bit is cleared. Reasons 1 and 4 do not require Sense Interrupt Status commands. The interrupt is cleared by Reading/Writing data to the FDC. Interrupts caused by reasons 2 and 3 may be uniquely identified with the aid of the Sense Interrupt Status command which resets the Interrupt signal and, via bits 5, 6, and 7 of Status register 0, identifies the cause of the interrupt (Table 7).

Table 7. Interrupt Identification

Seek End	Interrupt Code		
Bit 5	Bit 6	Bit 7	Cause
0	1	1	Ready Line changed state, either polarity
1	0	0	Normal Termination of Seek or Recalibrate command
1	1	0	Abnormal Termination of Seek or Recalibrate command

The Sense Interrupt Status command is used in conjunction with the Seek and Recalibrate commands which have no result phase. When the disk has reached the desired head position, the Z765A sets the interrupt line true. The host CPU must then issue a Sense Interrupt Status command to determine the actual cause of the interrupt, which could be Seek End or a change in ready status from one of the drives. Figure 6 is a graphic example.

Specify

The Specify command sets the initial values for each of the three internal timers. The Head Unload Time (HUT) defines the time from the end of the execution phase of one of the Read/Write commands to the head unload state. This timer is programmable from 16 to 240ms in increments of 16ms (01 = 16ms, 02 = 32ms...OF_{16} = 240ms). The Step Rate Time (SRT) defines the time interval between adjacent step pulses. This timer is programmable from 1 to 16ms in increments of 1ms (F = 1ms, E = 2ms, and D = 3ms). The Head Load Time (HLT) defines the time between the Head Load signal's going High and the start of the Read/Write operation. This timer is programmable from 2 to 254ms in increments of 2ms (01 = 2ms, 02 = 4ms, 03 = 6ms...7F = 254ms).

The time intervals mentioned are a direct function of the 8MHz clock; if the clock were reduced to 4MHz (minifloppy application), all time intervals would be increased by a factor of 2.

The choice of a DMA or non-DMA operation is made by the Non-DMA (ND) bit. When this bit is High (ND = 1), the Non-DMA mode is selected; when ND = 0, the DMA mode is selected.

Sense Drive Status

The processor uses this command to obtain the status of the FDDs. Status register 3 contains the Drive Status information stored internally in FDC registers.

Invalid

If an Invalid command (not defined above) is sent to the FDC, then the FDC terminates the command after Status Register 0 bit 7 is set to 1 and bit 6 to 0. No interrupt is generated by the Z765A during this condition. Bits 6 and 7 (DIO and RQM) in the Main Status register are both High, indicating to the processor that the Z765A is in the Result phase and the contents of Status register 0 (STO) must be read. When the processor reads Status register 0, it finds an 80_H indicating the receipt of an Invalid command.

A Sense Interrupt Status command must be sent after a Seek or Recalibrate Interrupt, otherwise the FDC considers the next command as an Invalid command.

This command may be used as a No-Op command to place the FDC in a standby or No Operation state.

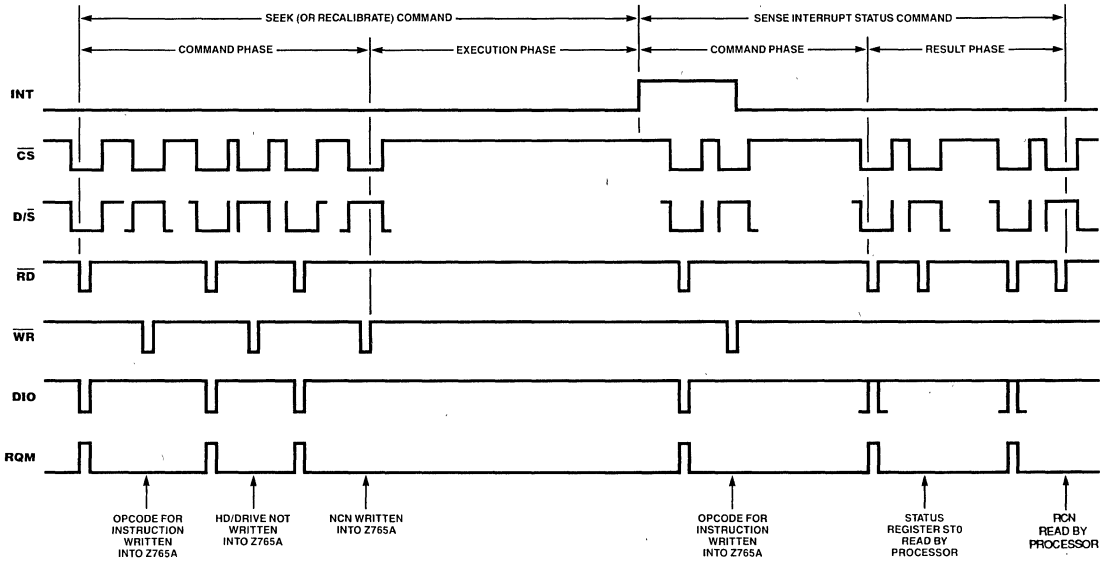


Figure 6. Seek, Recalibrate, and Sense Interrupt Status

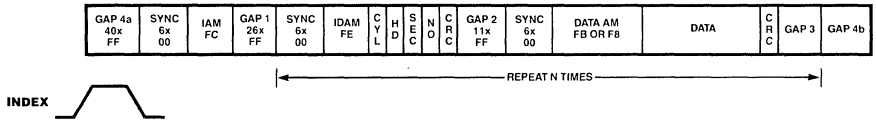


Figure 7. Data Format, FM Mode

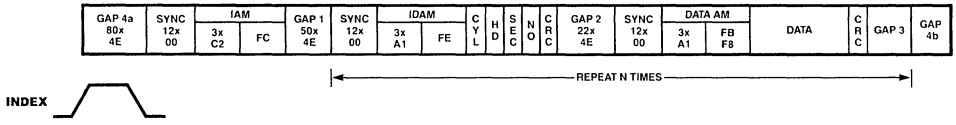


Figure 8. Data Format, MFM Mode

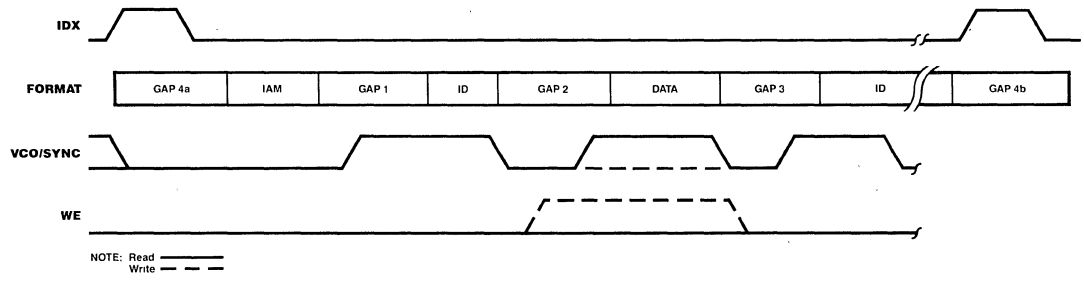


Figure 9. Data Timing Relationships

AC CHARACTERISTICS

T_A = −10°C to + 70°C; V_{CC} = +5V ± 5% unless otherwise specified.

Number	Symbol	Parameter	Min	Typ[1]	Max	Unit	Test Condition
			120	125	500	ns	
1	TcC	Clock Cycle Time		125		ns	8″ FDD
				250		ns	5¼″ FDD
2	TwCh	Clock Width (High)	40			ns	
2a	TwCl	Clock Width (Low)	40			ns	
3	TrC	Clock Rise Time			20	ns	
4	TfC	Clock Fall Time			20	ns	
5	TsAR	D/$\overline{S}$, $\overline{CS}$, $\overline{DACK}$ to $\overline{RD}$ ↓ Setup Time	0			ns	
6	ThRA	D/$\overline{S}$, $\overline{CS}$, $\overline{DACK}$ from $\overline{RD}$ ↑ Hold Time	0			ns	
7	TwRD	$\overline{RD}$ Width	250			ns	
8	TdRDf (Do)	$\overline{RD}$ ↓ to Data Output Delay			200	ns	C_L = 100 pf
9	TdRDr (Dz)	$\overline{RD}$ ↑ to Data Float Delay	20		100	ns	C_L = 100 pf
10	TsCS(WRf)	Control Signal (D/$\overline{S}$, $\overline{CS}$, $\overline{DACK}$) to $\overline{WR}$ ↓ Setup Time	0			ns	
11	ThCS(WRr)	Control Signal (D/$\overline{S}$, $\overline{CS}$, $\overline{DACK}$) from $\overline{WR}$ ↑ Hold Time	0			ns	
12	TwWR	$\overline{WR}$ Width	250			ns	
13	TsD(WRr)	Data to $\overline{WR}$ ↑ Setup Time	150			ns	
14	ThD(WRr)	Data from $\overline{WR}$ ↑ Hold Time	5			ns	
15	TdRDr(INT)	$\overline{RD}$ ↑ to INT Delay Time			500	ns	
16	TdWRr(INT)	$\overline{WR}$ ↑ to INT Delay Time			500	ns	
17	TcDRQ	DRQ Cycle Time	13			µs	
18	TdDRQ(DACK)	$\overline{DACK}$ ↓ to DRQ ↓ Delay			200	ns	
19	TdDACK(DRQ)	DRQ ↑ to $\overline{DACK}$ ↓ Delay	200			ns	TcC = 125 ns
20	TwDACK	$\overline{DACK}$ Width	2			TcC	
21	TwTC	TC Width	1			TcC	
22	TwRST	Reset Width	14			TcC	
				4		µs	MFM = 0 5¼″
				2		µs	MFM = 1 5¼″
23	TcWCK	WCK Cycle Time		2		µs	MFM = 0 8″
				1		µs	MFM = 1 8″
24	TwWCKh	WCK Width (High)	80	250	350	ns	
25	TrWCK	WCK Rise Time			20	ns	
26	TfWCK	WCK Fall Time			20	ns	
27	TdWCKr(PS)	WCK ↑ to Preshift Delay Time	20		100	ns	
28	TdWCKr(WEr)	WCK ↑ to WE ↑ Delay Time	20		100	ns	
29	TdWCKr(WDA)	WCK ↑ to WDA Delay Time	20		100	ns	
30	TwRDDh	RDD Width (High)	40			ns	
				4		µs	MFM = 0 5¼″
				2		µs	MFM = 1 5¼″
31	TWCY	Window Cycle Time		2		µs	MFM = 0 8″
				1		µs	MFM = 1 8″
32	TsW(RDDh) ThW(RDDl)	Window to RDD ↑ Setup Time Window from RDD ↓ Hold Time	15			ns	
33	TsUS(RWh)	Unit Select to $\overline{RW}$/SEEK ↑ Setup Time	12			µs	
34	TsRWr(DIR)	$\overline{RW}$/SEEK ↑ to LCT/DIR Setup Time	7			µs	
35	TsDIR(STEPr)	LCT/DIR to STEP ↑ Setup Time	1			µs	
36	ThUS(STEPl)	Unit Select from STEP ↓ Hold Time	5			µs	

NOTES: 1. Typical values for T_A = 25°C and nominal supply voltage
2. Under software control, the range is from 1 ms to 16 ms at 8 MHz clock period.

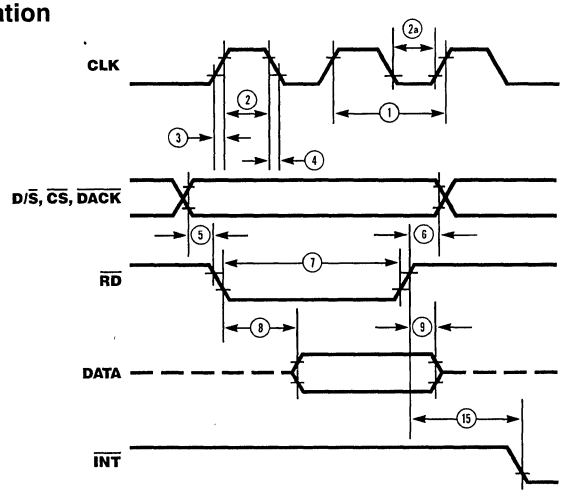
AC CHARACTERISTICS (Continued)

T_A = −10°C to +70°C; V_{CC} = +5V ± 5% unless otherwise specified.

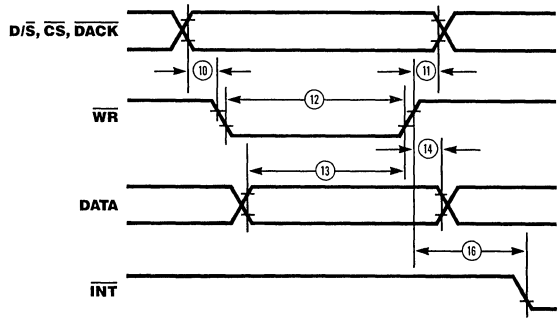
Number	Symbol	Parameter	Min	Typ[1]	Max	Unit	Test Condition
37	TwSTEPh	STEP Width (High)	6	7	8	µs	
38	TcSTEP	STEP Cycle Time	16	Note 2	Note 2	µs	
39	TwFRh	FAULT RESET Width (High)	8		10	µs	
40	TwWDAh	Write Data (WDA) Width (High)	T_0-50			ns	
41	ThUS(SEEKf)	Unit Select from $\overline{RW}$/SEEK ↓ Hold Time	15			µs	
42	ThSEEK(DIR)	$\overline{RW}$/SEEK from LCT/DIR Hold Time	30			µs	
43	ThDIR(STEPf)	LCT/DIR from STEP ↓ Hold Time	24			µs	
44	TwIDX	INDEX Width (High and Low)	4			TcC	
45	TdDRQh(RDl)	DRQ ↑ to $\overline{RD}$ ↓ Delay Time	800			ns	
46	TdDRQh(WRl)	DRQ ↑ to $\overline{WR}$ ↓ Delay Time	250			ns	
47	TdDRQh(RWh)	DRQ ↑ to $\overline{RD}$ ↑ or $\overline{WR}$ ↑ Delay Time			12	µs	

NOTES: 1. Typical values for T_A = 25°C and nominal supply voltage.
2. Under software control, the range is from 1 ms to 16 ms at 8 MHz clock period.

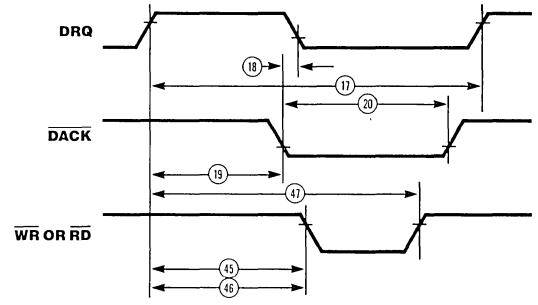
Processor Read Operation



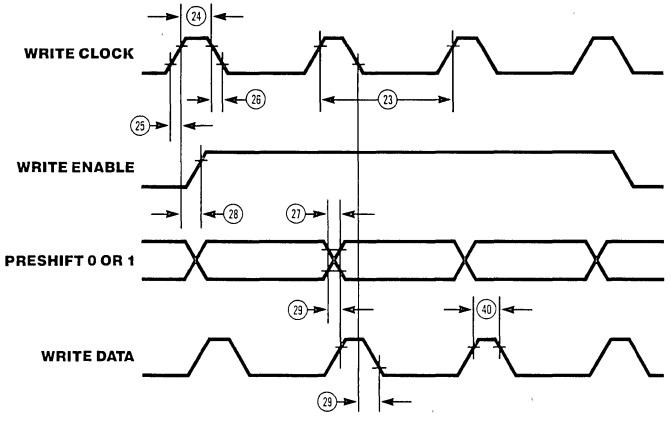
Processor Write Operation



DMA Operation

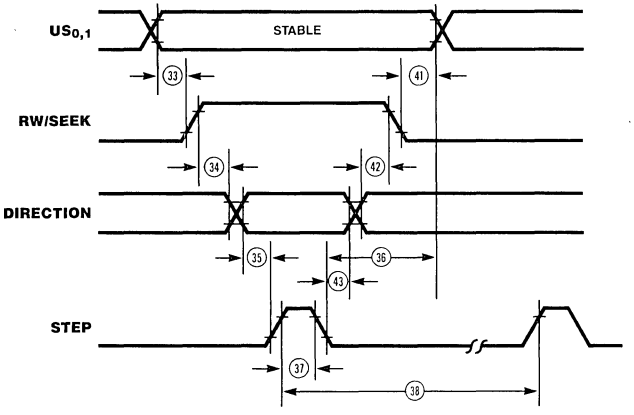


FDD Write Operation



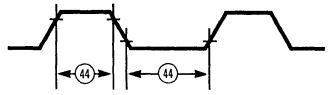
	Preshift 0	Preshift 1
Normal	0	0
Late	0	1
Early	1	0

Seek Operation

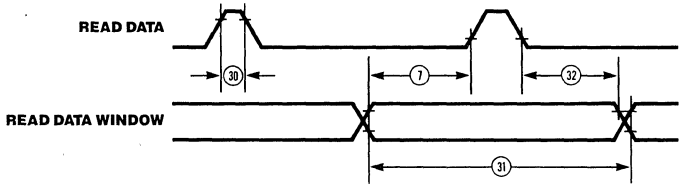


FLT Reset

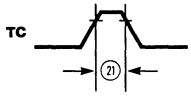
INDEX



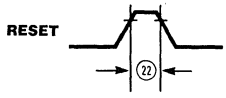
FDD Read Operation



Terminal Count



RESET



ABSOLUTE MAXIMUM RATINGS

T_A = 25°C

Operating Temperature . . . . . 0°C to + 70°C
Storage Temperature . . . . . −65°C to +150°C
All Output Voltages . . . . . −.5V to +7V
All Input Voltages . . . . . −.3V to +7V
Supply Voltage V_{CC} . . . . . −.5V to +7V
Power Dissipation 1W

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above these indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC CHARACTERISTICS

T_A = 0°C to +70°C; V_{CC} = +5V ± 5% unless otherwise specified.

Symbol	Parameter	Min	Typ*	Max	Unit	Test Condition
V_{IL}	Input Low Voltage	−0.5		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage			0.40	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	2.4		V_{CC}	V	I_{OH} = −200 μA
V_{ILC}	Input Low Voltage (CLK + WR Clock)	−0.5		0.65	V	
V_{IHC}	Input High Voltage (CLK + WR Clock)	2.4		V_{CC} + 0.5	V	
I_{CC}	V_{CC} Supply Current			150	mA	
I_{LI}	Input Load Current			10	μA	V_{IN} = V_{CC}
	(All Input Pins)			−10	μA	V_{IN} = 0V
I_{LOH}	High Level Output Leakage Current			10	μA	V_{OUT} = V_{CC}
I_{LOL}	Low Level Output Leakage Current			−10	μA	V_{OUT} = +0.40V

*Typical values for T_A = 25°C and nominal supply voltage.

CAPACITANCE

T_A = 25°C; f_c = 1 MHz; V_{CC} = 0V

Symbol	Parameter	Min	Max	Unit	Test Condition
C_{CLOCK}	Clock Input Capacitance		20	pF	All pins except pin under
C_{IN}	Input Capacitance		10	pF	test tied to AC Ground
C_{OUT}	Output Capacitance		20	pF	

PRODUCT SPECIFICATION

Z5380

SCSI SMALL COMPUTER SYSTEM INTERFACE

FEATURES

- Compatible 5380 pinout
- Low power CMOS
- Asynchronous interface, supports 1.5 MB/s
- Direct SCSI Bus interface with on-board 48 mA drivers
- Supports Target and Initiator roles
- Arbitration support
- DMA or programmed I/O data transfers
- Supports Normal or Block Mode DMA
- Memory or I/O Mapped CPU interface

GENERAL DESCRIPTION

The Z5380 SCSI (Small Computer System Interface) controller is a 40-pin DIP or 44-pin PLCC CMOS device (Figure 1). It is designed to implement the SCSI protocol as defined by the ANSI X3.131-1986 standard, and is fully compatible with the industry standard 5380. It is capable of operating both as a Target and as an Initiator. Special high-current open-drain outputs enable it to directly interface to, and drive, the SCSI bus. The Z5380 has the necessary interface hook-ups so the system CPU can communicate with it like with any other peripheral device. The CPU can read from, or write to, the SCSI registers which are addressed as standard or memory-mapped I/Os.

The Z5380 increases the system performance by minimizing the CPU intervention in DMA operations which the SCSI controls. The CPU is interrupted by the SCSI when it detects a bus condition that requires attention. It also supports arbitration and reselection. The Z5380 has the proper hand-shake signals to support normal and block mode DMA operations with most DMA controllers available (Figure 2).

Note: All Signals with a preceding front slash, "/", are active Low, e.g.: B//W (WORD is active Low); /B/W (BYTE is active Low, only); /N//S (NORMAL and SYSTEM are both active Low).

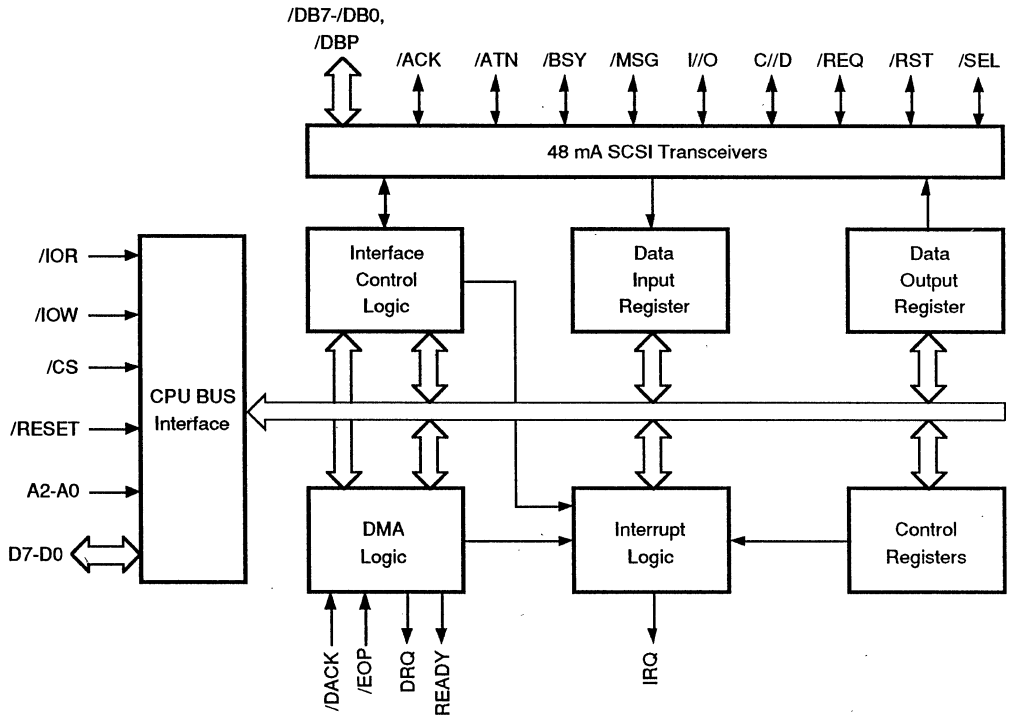


Figure 1. Block Diagram

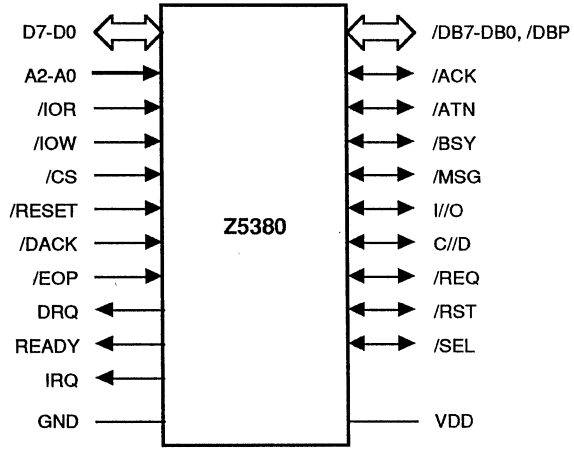


Figure 2. Logic Symbol

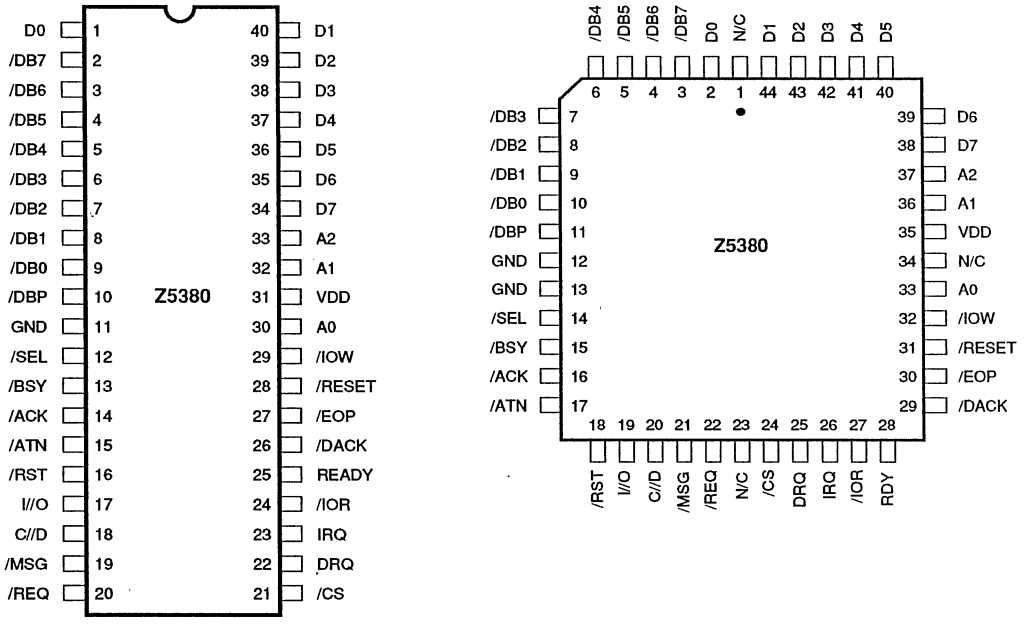


Figure 3. Pin Diagrams

PIN DESCRIPTION

Microprocessor Bus

Figure 3 shows the pins and their respective functions for both the DIP and PLCC.

A2-A0. *Address Lines* (Input). Address lines are used with /CS, /IOR, or /IOW to address all internal registers.

/CS. *Chip Select* (Input, Active Low). This signal, in conjunction with /IOR or /IOW, enables the internal register selected by A2-A0, to be read from or written to.

/DACK. *DMA Acknowledge* (Input, Active Low). /DACK resets DRQ and selects the data register for input or output data transfers. /DACK is used by DMA controller instead of /CS.

DRQ. *DMA Request* (Output, Active High). DRQ indicates that the data register is ready to be read or written. DRQ is asserted only if DMA mode is set in the Command Register. DRQ is cleared by /DACK.

D7-D0. *Data Lines* (Bidirectional, three-state, Active High). Bidirectional microprocessor data bus lines. D0 is the Least Significant Bit of the bus. Data bus lines carry data and commands to and from the SCSI.

/EOP. *End of Process* (Input, Active Low). /EOP is used to terminate a DMA transfer. If asserted during a DMA cycle, the current byte will be transferred, but no additional bytes will be requested.

/IOR. *I/O Read* (Input, Active Low). /IOR is used in conjunction with /CS and A2-A0 to read an internal register. It also selects the Input Data Register when used with /DACK.

/IOW. *I/O Write* (Input, Active Low). /IOW is used in conjunction with /CS and A2-A0 to write an internal register. It also selects the Output Data Register when used with /DACK.

PIN DESCRIPTION (Continued)

IRQ. *Interrupt Request* (Output, Active High). IRQ alerts a microprocessor of an error condition or an event completion.

READY. *Ready* (Output, Active High). Ready is used to control the speed of Block Mode DMA transfers. This signal goes active to indicate the chip is ready to send/receive data and remains Low after a transfer until the last byte is sent or until the DMA Mode bit is reset.

/RESET. *Reset* (Input, Active Low). /RESET clears all registers. It has no effect upon the SCSI /RST signal.

SCSI Bus

The following signals are all bidirectional, active Low, open-drain, with 48 mA sink capability. All pins interface directly with the SCSI bus.

/ACK. *Acknowledge* (Bidirectional, Open-drain, Active Low). Driven by an Initiator, /ACK indicates an acknowledgement for a /REQ//ACK data-transfer handshake. In the Target role, /ACK is received as a response to the /REQ signal.

/ATN. *Attention* (Bidirectional, Open-drain, Active Low). Driven by an Initiator, received by the Target, /ATN indicates an Attention condition.

/BSY. *Busy* (Bidirectional, Open-drain, Active Low). This signal indicates that the SCSI bus is being used and can be driven by both the Initiator and the Target device.

C//D. *Control/Data* (Bidirectional, Open-drain). Driven by the Target and received by the Initiator, C//D indicates whether Control or Data information is on the Data Bus. True indicates Control.

/DB7-/DB0, /DBP. *Data Bus Bits, Data Bus Parity Bit* (Bidirectional, Open-drain). These eight data bits (/DB7-/DB0), plus a parity bit (/DBP) form the data bus. /DB7 is the most significant bit (MSB) and has the highest priority during the Arbitration phase. Data parity is odd. Parity is always generated and optionally checked. Parity is not valid during Arbitration.

I//O. *Input/Output* (Bidirectional, Open-drain). I/O is a signal driven by a Target which controls the direction of data movement on the SCSI bus. True indicates input to the Initiator. This signal is also used to distinguish between Selection and Reselection phases.

/MSG. *Message* (Bidirectional, Open-drain, Active Low). This signal is driven by the Target during the Message phase. This signal is received by the Initiator.

/REQ. *Request* (Bidirectional, Open-drain, Active Low). Driven by the Target and received by the Initiator, this signal indicates a request for a /REQ//ACK data-transfer handshake.

/RST. *SCSI Bus Reset* (Bidirectional, Open-drain, Active Low). This signal indicates a SCSI bus Reset condition.

/SEL. *Select* (Bidirectional, Open-drain, Active Low). This signal is used by an Initiator to select a Target, or by a Target to reselect an Initiator.

Power Signals:

GND. Ground (0V)

VDD. VDD Supply (+5V)

FUNCTIONAL DESCRIPTION

The Z5380 Small Computer System Interface (SCSI) has a set of eight registers that are controlled by the CPU. By reading and writing the appropriate registers, the CPU may initiate any SCSI Bus activity or may sample and assert any signal on the SCSI Bus. This allows the user to implement all or any of the SCSI protocol in software. These registers are read (written) by activating /CS with an address on A2-A0 and then issuing an /IOR (/IOW) pulse. This section describes the operation of the internal registers (Table 1).

Table 1. Register Summary

Address				
A2	A1	A0	R/W	Register Name
0	0	0	R	Current SCSI Data
0	0	0	W	Output Data
0	0	1	R/W	Initiator Command
0	1	0	R/W	Mode
0	1	1	R/W	Target Command
1	0	0	R	Current SCSI Bus Status
1	0	0	W	Select Enable
1	0	1	R	Bus and Status
1	0	1	W	Start DMA Send
1	1	0	R	Input Data
1	1	0	W	Start DMA Target Receive
1	1	1	R	Reset Parity/Interrupt
1	1	1	W	Start DMA Initiator Receive

Data Registers

The data registers are used to transfer SCSI commands, data, status, and message bytes between the microprocessor Data Bus and the SCSI Bus. The Z5380 does not interpret any information that passes through the data registers. The data registers consist of the transparent Current SCSI Data Register, the Output Data Register, and the Input Data Register.

Current SCSI Data Register. *Address 0* (Read Only). The Current SCSI Data Register (Figure 4) is a read-only register which allows the microprocessor to read the active SCSI Data Bus. This is accomplished by activating /CS with an address on A2-A0 of 000 and issuing an /IOR pulse. If parity checking is enabled, the SCSI Bus parity is checked at the beginning of the read cycle. This register is used during a programmed I/O data read or during Arbitration to check for higher priority arbitrating devices. Parity is not guaranteed valid during Arbitration.

Output Data Register. *Address 0* (Write Only). The Output Data Register (Figure 5) is a write-only register that is used to send data to the SCSI Bus. This is accomplished by either using a normal CPU write, or under DMA control, by using /IOW and /DACK. This register also asserts the proper ID bits on the SCSI Bus during the Arbitration and Selection phases.

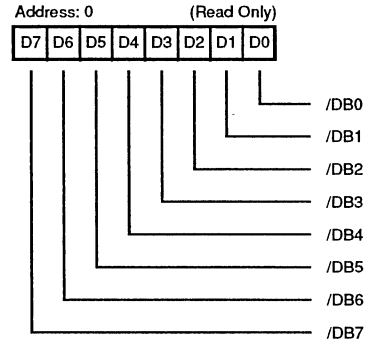


Figure 4. Current SCSI Data Register

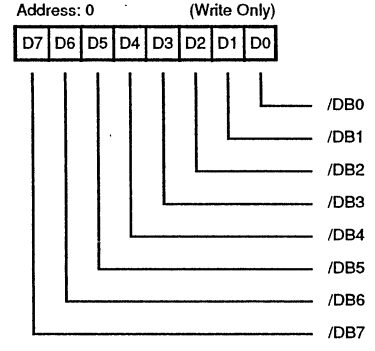


Figure 5. Output Data Register

Initiator Command Register. *Address 1* (Read/Write). The Initiator Command Register (Figures 6 and 7) are read and write registers which assert certain SCSI Bus signals, monitors those signals, and monitors the progress of bus arbitration. Many of these bits are significant only when being used as an Initiator; however, most can be used during Target role operation.

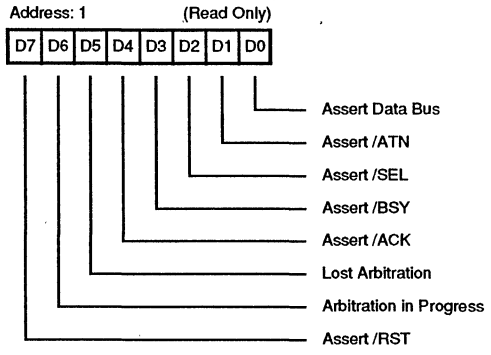


Figure 6. Initiator Command Register

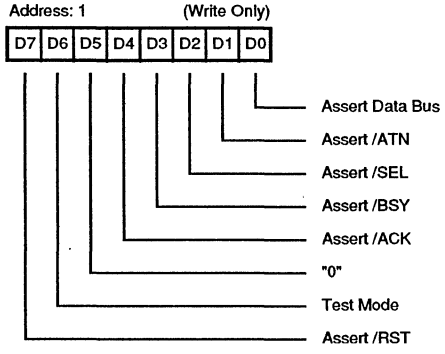


Figure 7. Initiator Command Register

The following describes the operation of all bits in the Initiator Command Register:

Bit 0. *Assert Data Bus.* This bit, when set, allows the contents of the Output Data Register to be enabled as chip outputs on the signals /DB7-DB0. Parity is also generated and asserted on /DBP.

When connected as an Initiator, the outputs are only enabled if the Target Mode bit (Mode Register, bit 6) is 0, the received signal I/O is False, and the phase signals (C/D, I/O, and /MSG) match the contents of the Assert C/D, Assert I/O, and Assert /MSG in the Target Command Register.

This bit should also be set during DMA send operations.

Bit 1. *Assert /ATN.* /ATN may be asserted on the SCSI Bus by setting this bit to a one (1) if the Target Mode bit (Mode Register, bit 6) is False. /ATN is normally asserted by the initiator to request a Message Out bus phase. Note that since Assert /SEL and Assert /ATN are in the same register, a select with /ATN may be implemented with one CPU write. /ATN may be deasserted by resetting this bit to zero. A read of this register simply reflects the status of this bit.

Bit 2. *Assert /SEL.* Writing a one (1) into this bit position asserts /SEL onto the SCSI Bus. /SEL is normally asserted after Arbitration has been successfully completed. /SEL may be disabled by resetting bit 2 to a zero. A read of this register reflects the status of this bit.

Bit 3. *Assert /BSY.* Writing a one (1) into this bit position asserts /BSY onto the SCSI Bus. Conversely, a zero resets the /BSY signal. Asserting /BSY indicates a successful selection or reselection. Resetting this bit creates a Bus-Disconnect condition. Reading this register reflects bit status.

Bit 4. *Assert /ACK.* Bit 4 is used by the bus initiator to assert /ACK on the SCSI Bus. In order to assert /ACK, the Target Mode bit (Mode Register, bit 6) must be False. Writing a zero to this bit deasserts /ACK. Reading this register reflects bit status.

Bit 5. *"0"* (Write Bit). Bit 5 should be written with a zero for proper operation.

Bit 5. *LA* (Lost Arbitration - Read Bit). Bit 5, when active, indicates that the SCSI detected a Bus-Free condition, arbitrated for use of the bus by asserting /BSY and its ID on the Data Bus, and lost Arbitration due to /SEL being asserted by another bus device. This bit is active only when the Arbitrate bit (Mode Register, bit 0) is active.

Bit 6. *Test Mode* (Write Bit). Bit 6 is written during a test environment to disable all output drivers, effectively removing the Z5380 from the circuit. Resetting this bit returns the part to normal operation.

Bit 6. *AIP* (Arbitration in Process - Read Bit). Bit 6 is used to determine if Arbitration is in progress. For this bit to be active, the Arbitrate bit (Mode Register, bit 0) must have been set previously. It indicates that a Bus-Free condition has been detected and that the chip has asserted /BSY and put the contents of the Output Data Register onto the SCSI Bus. AIP will remain active until the Arbitrate bit is reset.

Bit 7. *Assert /RST.* Whenever a one is written to bit 7 of the Initiator Command Register, the /RST signal is asserted on the SCSI Bus. The /RST signal will remain asserted until this bit is reset or until an external /RESET occurs. After this bit

is set (1), IRQ goes active and all internal logic and control registers are reset (except for the interrupt latch and the Assert /RST bit). Writing a zero to bit 7 of the Initiator Command Register deasserts the /RST signal. The status of this bit is monitored by reading the Initiator Command Register.

Mode Register. *Address 2*(Read/Write). The Mode Register controls the operation of the chip. This register determines whether the Z5380 operates as an Initiator or a Target, whether DMA transfers are being used, whether parity is checked, and whether interrupts are generated on various external conditions. This register is read to check the value of these internal control bits (Figure 8).

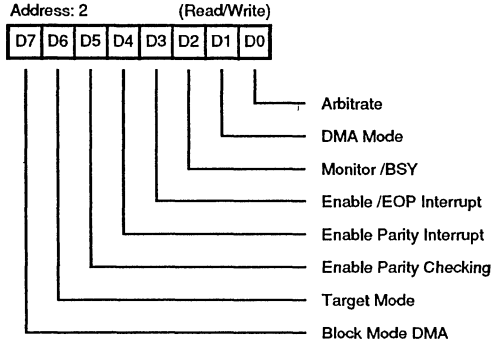


Figure 8. Mode Register

The following describes the operation of all bits in the Initiator Command Register:

Bit 0. *Arbitrate.* The Arbitrate bit is set (1) to start the Arbitration process. Prior to setting this bit, the Output Data Register should contain the proper SCSI device ID value. Only one data bit should be active for SCSI Bus Arbitration. The Z5380 waits for a Bus-Free condition before entering the Arbitration phase. The results of the Arbitration phase is determined by reading the status bits LA and AIP (Initiator Command Register, bits 5 and 6, respectively).

Bit 1. *DMA Mode.* The DMA Mode bit is normally used to enable a DMA transfer and must be set (1) prior to writing Start DMA Send Register, Start DMA Target Register, and Start DMA Initiator Receiver Register. These three registers are used to start DMA transfers. The Target Mode bit (Mode Register, bit 6) must be consistent with writes to Start DMA Target Receive and Start DMA Initiator Receive Registers; i.e., set (1) for a write to Start DMA Target Receive Register and set (0) for Start DMA Initiator Receive Register. The control bit Assert Data Bus (Initiator Command Register, bit 0) must be True (1) for all DMA send operations. In the DMA mode, /REQ and /ACK are automatically controlled.

The DMA Mode bit is not reset upon the receipt of an /EOP signal. Any DMA transfer is stopped by writing a zero into this bit location; however, care must be taken not to cause /CS and /DACK to be active simultaneously.

Bit 2. *Monitor Busy.* The Monitor Busy bit, when True (1), causes an interrupt to be generated for an unexpected loss of /BSY. When the interrupt is generated due to loss of /BSY, the lower six bits of the Initiator Command Register are reset (0) and all signals are removed from the SCSI Bus.

Bit 3. *Enable /EOP interrupt.* The enable /EOP interrupt bit, when set (1), causes an interrupt to occur when the /EOP (End Of Process) signal is received from the DMA controller logic.

Bit 4. *Enable Parity Interrupt.* The Enable Parity Interrupt bit, when set (1), will cause an interrupt (IRQ) to occur if a parity error is detected. A parity interrupt will only be generated if the Enable Parity Checking bit (bit 5) is also enabled (1).

Bit 5. *Enable Parity Checking.* The Enable Parity Checking bit determines whether parity errors are ignored or saved in the parity error latch. If this bit is reset (0), parity is ignored. Conversely, if this bit is set (1), parity errors are saved.

Bit 6. *Target Mode.* The Target Mode bit allows the Z5380 to operate as a SCSI Bus Initiator or Target. With this bit reset (0), the Z5380 operates as a SCSI Bus Initiator. Setting Target Mode bit to 1 programs the Z5380 to ope ate as a SCSI Bus Target device. If the signals /ATN and /ACK are to be asserted on the SCSI Bus, the Target Mode bit must be reset (0). If the signals C//D, I//O, /MSG, and /REQ are to be asserted on the SCSI Bus, the Target Mode bit must be set (1).

Bit 7. *Block Mode DMA.* The Block Mode DMA bit controls the characteristics of the DMA DRQ-/DACK handshake. When this bit is reset (0) and the DMA Mode bit is active (1), the DMA handshake uses the normal interlocked handshake, and the rising edge of /DACK indicates the end of each byte being transferred. In Block Mode operation, when the Block Mode DMA bit is set (1) and DMA Mode bit is active (1), the end of /IOR or /IOW signifies the end of each byte transferred and /DACK is allowed to remain active throughout the DMA operation. Ready can then be used to request the next transfer.

Target Command Register. *Address 3* (Read/Write). When connected as a target device, the Target Command Register (Figure 9) allows the CPU to control the SCSI Bus Information Transfer phase and/or to assert /REQ by writing this register. The Target Mode bit (Mode Register, bit 6) must be True (1) for bus assertion to occur. The SCSI Bus phases are described in Table 2.

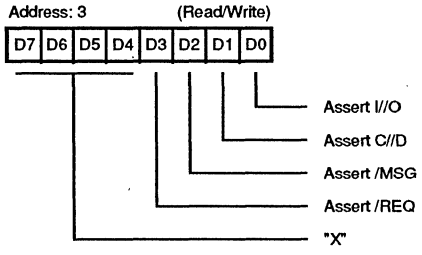


Figure 9. Target Command Register

Table 2. SCSI Information Transfer Phases

Bus Phase	Assert I//O	Assert C//D	Assert /MSG
Data Out	0	0	0
Unspecified	0	0	1
Command	0	1	0
Message Out	0	1	1
Data In	1	0	0
Unspecified	1	0	1
Status	1	1	0
Message In	1	1	1

When connected as an Initiator with DMA Mode bit True, if the phase lines (I//O, C//D, and /MSG) do not match the phase bits in the Target Command Register, a phase mismatch interrupt is generated when /REQ goes active. To send data as an Initiator, the Assert I//O, Assert C//D, and Assert /MSG bits must match the corresponding bits in the Current SCSI Bus Status Register. The Assert /REQ bit (bit 3) has no meaning when operating as an Initiator.

Bits 4, 5, 6, and 7 are not used.

Current SCSI Bus Status Register. *Address 4* (Read Only). The Current SCSI Bus Register is a read-only register which is used to monitor seven SCSI Bus control signals, plus the Data Bus parity bit. For example, an Initiator device can use this register to determine the current bus phase and to poll /REQ for pending data transfers. This register may also be used to determine why a particular interrupt occurred. Figure 10 describes the Current SCSI Bus Status Register.

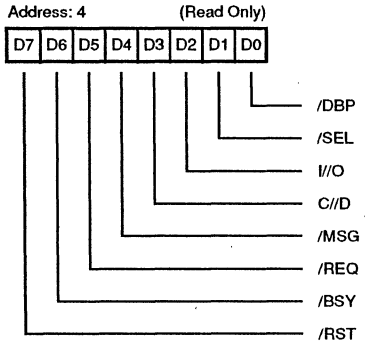


Figure 10. Current SCSI Bus Status Register

Select Enable Register. *Address 4* (Write Only). The Select Enable Register (Figure 11) is a write-only register which is used as a mask to monitor a signal ID during a selection attempt. The simultaneous occurrence of the correct ID bit, /BSY False, and /SEL True causes an interrupt. This interrupt can be disabled by resetting all bits in this register. If the Enable Parity Checking bit (Mode Register, bit 5) is active (1), parity is checked during selection.

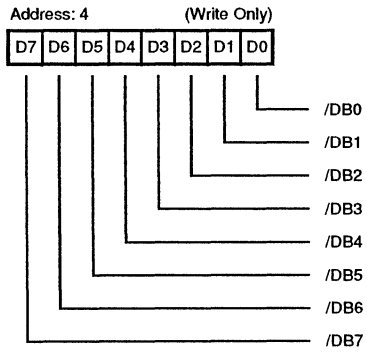


Figure 11. Select Enable Register

Bus and Status Register. *Address 5* (Read Only). The Bus and Status Register (Figure 12) is a read-only register which can be used to monitor the remaining SCSI control signals not found in the Current SCSI Bus Status Registers (/ATN and /ACK), as well as six other status bits. The following describes each bit of the Bus and Status Register individually.

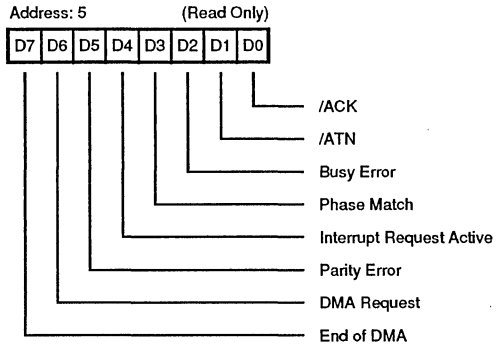


Figure 12. Bus and Status Register

Bit 0. */ACK.* Bit 0 reflects the condition of the SCSI Bus control signal /ACK. This signal is normally monitored by the Target device.

Bit 1. */ATN.* Bit 1 reflects the condition of the SCSI Bus control signal /ATN. This signal is normally monitored by the Target device.

Bit 2. *Busy Error.* The Busy Error bit is active if an unexpected loss of the /BSY signal has occurred. This latch is set whenever the Monitor Busy bit (Mode Register, bit 2) is True and /BSY is False. An unexpected loss of /BSY disables any SCSI outputs and resets the DMA Mode bit (Mode Register, bit 1).

Bit 3. *Phase Match.* The SCSI signals /MSG, C//D, and I//O, represent the current information Transfer phase. The Phase Match bit indicates whether the current SCSI Bus phase matches the lower 3 bits of the Target Command Register. Phase Match is continuously updated and is only significant when operating as a Bus Initiator. A phase match is required for data transfers to occur on the SCSI Bus.

Bit 4.*Interrupt Request ACTIVE.* Bit 4 is set if an enabled interrupt condition occurs. It reflects the current state of the IRQ output and can be cleared by reading the Reset Parity/ Interrupt Register.

Bit 5. *Parity Error.* Bit 5 is set if a parity error occurs during a data receive or a device selection. The Parity Error bit can only be set (1) if the Enable Parity Check bit (Mode Register, bit 5) is active (1). This bit may be cleared by reading the Reset Parity/Interrupt Register.

Bit 6. *DMA Request.* The DMA Request bit allows the CPU to sample the output pin DRQ. DRQ can be cleared by asserting /DACK or by resetting the DMA Mode bit (bit 1) in the Mode Register. The DRQ signal does not reset when a phase-mismatch interrupt occurs.

Bit 7. *End of DMA Transfer.* The End of DMA Transfer bit is set if /EOP, /DACK, and either /IOR or /IOW are simultaneously active for at least 100ns. Since the /EOP signal can occur during the last byte sent to the Output Data Register, the /REQ and /ACK signals should be monitored to ensure that the last byte has been transferred. This bit is reset when the DMA Mode bit is reset (0) in the Mode Register.

Input Data Register. *Address 6* (Read Only). The input Data Register (Figure 13) is a read-only register that is used to read latched data from the SCSI Bus. Data is latched either during a DMA Target receive operation when /ACK goes active or during a DMA Initiator receive when /REQ goes active. The DMA Mode bit (bit 1) must be set before data can be latched in the Input Data Register. This register is read under DMA control using /IOR and /DACK. Parity is optionally checked when the Input Data Register is loaded.

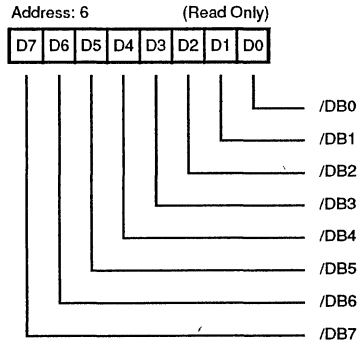


Figure 13. Input Data Register

DMA Registers

Three write-only registers are used to initiate all DMA activity. They are: Start DMA Send, Start DMA Target Receive, and Start DMA Initiator Receive. Performing a write operation into one of these registers starts the desired type of DMA transfer. Data presented to the Z5380 on signals D7-D0 during the register write is meaningless and has no effect on the operation. Prior to writing these registers, the Block Mode DMA bit (bit 7), the DMA Mode bit (bit 1), and the Target Mode bit (bit 6) in the Mode Register must be appropriately set. The individual registers are briefly described as follows:

Start DMA Send. *Address 5* (Write Only). This register is written to initiate a DMA send, from the DMA to the SCSI Bus, for either Initiator or Target role operations. The DMA Mode bit (Mode Register, bit 1) is set prior to writing this register.

Start DMA Target Receive. *Address 6* (Write Only). This register is written to initiate a DMA receive - from the SCSI Bus to the DMA, for Target operation only. The DMA Mode bit (bit 1) and the Target Mode bit (bit 6) in the Mode Register must both be set (1) prior to writing this register.

Start DMA Initiator Receive. *Address 7* (Write Only). This register is written to initiate a DMA receive - from the SCSI Bus to the DMA, for Initiator operation only. The DMA Mode bit (bit 6) must be False (0) in the Mode Register prior to writing this register.

Reset Parity/Interrupt. *Address 7* (Read Only). Reading this register resets the Parity Error bit (bit 5), the Interrupt Request bit (bit 4), and the Busy Error bit (bit 2) in the Bus and Status Register.

On-Chip SCSI Hardware Support

The Z5380 is easy to use because of its simple architecture. The chip allows direct control and monitoring of the SCSI Bus by providing a latch for each signal. However, portions of the protocol define timings which are much too quick for traditional microprocessors to control. Therefore, hardware support has been provided for DMA transfers, bus arbitration, phase change monitoring, bus disconnection, bus reset, parity generation, parity checking, and device selection/reselection.

Arbitration is accomplished using a bus-free filter to continuously monitor /BSY. If /BSY remains inactive for at least 1.2us, the SCSI Bus is considered free and Arbitration may begin. Arbitration will begin if the bus is free, /SEL is inactive, and the Arbitrate bit (Mode Register, bit 0) is active. Once arbitration has begun (/BSY asserted), an arbitration delay of 2.2us must elapse before the Data Bus can be examined to determine if Arbitration is enabled. This delay is implemented in the controlling software driver.

The Z5380 is a clockwise device. Delays such as bus-free delay, bus-set delay, and bus-settle delay are implemented using gate delays. These delays may differ between devices because of inherent process variations, but are well within the proposed ANSI X3.131 - 1986 specification.

Interrupts

The Z5380 provides an interrupt output (IRQ) to indicate a task completion or an abnormal bus occurrence. The use of interrupts is optional and may be disabled by resetting the appropriate bits in the Mode Register or the Select Enable Register.

When an interrupt occurs, the Bus and Status Register and the Current SCSI Bus Status Register (Figures 12 and 10) must be read to determine which condition created the interrupt. IRQ can be reset simply by reading the Reset Parity/Interrupt Register or by an external chip reset /RESET active for 200ns.

Assuming the Z5380 has been properly initialized, an interrupt is generated if the chip is selected or reselected; if an /EOP signal occurs during a DMA transfer; if a SCSI Bus reset occurs; if a parity error occurs during a data transfer; if a bus phase mismatch occurs; or if a SCSI Bus disconnection occurs.

Selection/Reselection Interrupt

The Z5380 generates a select interrupt if /SEL is active (0), its device ID is True and /BSY is False for at least a bus-settle delay. If I//O is active, this is considered a reselect interrupt. The correct ID bit is determined by a match in the Select Enable Register. Only a single bit match is required to generate an interrupt. This interrupt may be disabled by writing zeros into all bits of the Select Enable Register.

If parity is supported, parity should be good during the selection phase. Therefore, if the Enable Parity bit (Mode Register, bit 5) is active, the Parity Error bit is checked to ensure that a proper selection has occurred. The Enable Parity Interrupt bit need not be set for this interrupt to be generated.

The proposed SCSI specification also requires that no more than two device ID's be active during the selection process. To ensure this, the Current SCSI Data Register is read.

The proper values for the Bus and Status Register and the Current SCSI Bus Status Register are displayed in Figures 14 and 15, respectively.

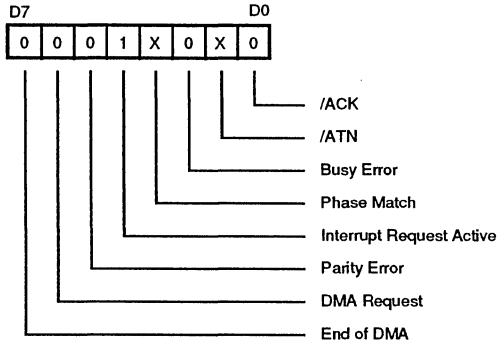


Figure 14. Bus and Status Register

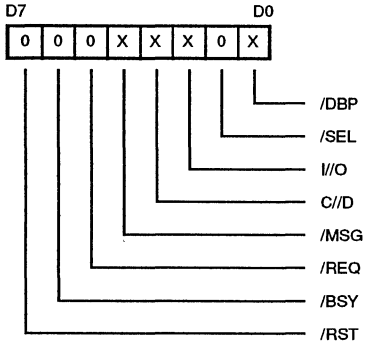


Figure 15. Current SCSI Bus Status Register

End Of Process (EOP) Interrupt

An End Of Process signal (EOP) which occurs during a DMA transfer (DMA Mode True) will set the End of DMA Status bit (bit 7) and will optionally generate an interrupt if Enable EOP Interrupt bit (Mode Register, bit 3) is True. The /EOP pulse will not be recognized (End of DMA bit set) unless /EOP, /DACK, and either /IOR or /IOW are concurrently active for at least 100 ns. DMA transfers can still occur if /EOP was not asserted at the correct time. This interrupt is disabled by resetting the Enable EOP Interrupt bit.

The proper values for the Bus and Status Register and the Current SCSI Bus Status Register for this interrupt are shown in Figures 16 and 17.

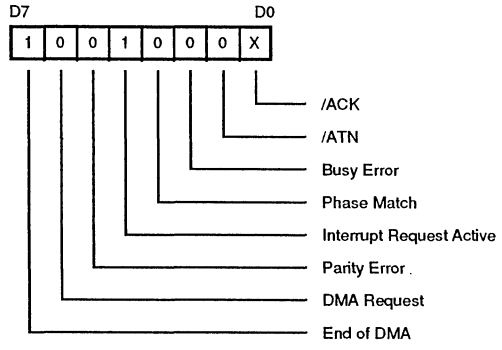


Figure 16. Bus and Status Register

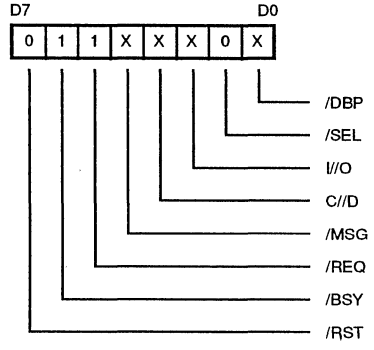


Figure 17. Current SCSI Bus Status Register

The End of DMA bit is used to determine when a block transfer is complete. Receive operations are complete when there is no data left in the chip and no additional handshakes occurring. The only exception to this is receiving data as an Initiator and the Target opts to send additional data for the same phase. In this /REQ goes active and the new data is present in the Input Data Register. Since a phase-mismatch interrupt will not occur, /REQ and /ACK need to be sampled to determine that the Target is attempting to send more data.

For send operations, the End of DMA bit is set when the DMA finishes its transfers, but the SCSI transfer may still be in progress. If connected as a Target, /REQ and /ACK should be sampled until both are False. If connected as an Initiator, a phase change interrupt is used to signal the completion of the previous phase. It is possible for the Target to request additional data for the same phase. In this case, a phase change will not occur and both /REQ and /ACK are sampled to determine when the last byte was transferred.

SCSI Bus Reset Interrupt

The Z5380 generates an interrupt when the /RST signal transitions to True. The device releases all bus signals within a bus-clear delay of this transition. This interrupt also occurs after setting the Assert /RST bit (Initiator Command Register, bit 7). This interrupt cannot be disabled. (Note: /RST is not latched in bit 7 of the Current SCSI Bus Status Register and is not active when this port is read. For this case, the Bus Reset interrupt is determined by default.)

The proper values for the Bus and Status Register and the Current SCSI Bus Status Register are displayed in Figures 18 and 19, respectively.

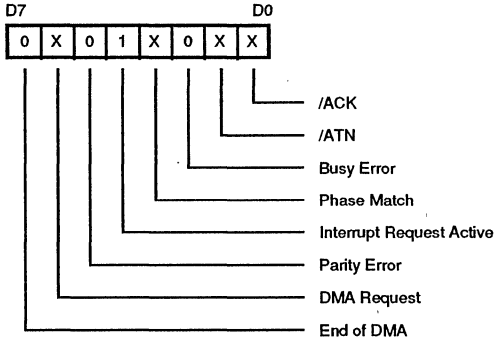


Figure 18. Bus and Status Register

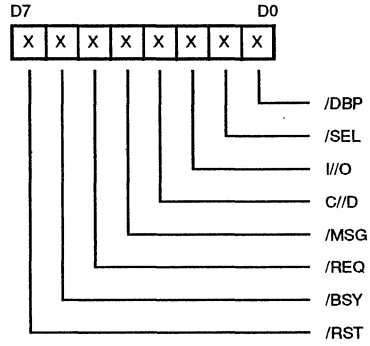


Figure 19. Current SCSI Bus Status Register

Parity Error Interrupt

An Interrupt is generated for a received parity error if the Enable Parity Check (bit 5) and the Enable Parity Interrupt (bit 4) bits are set (1) in the Mode Register. Parity is checked during a read of the Current SCSI Data Register and during a DMA receive operation. A parity error can be detected without generating an interrupt by disabling the Enable Parity Interrupt bit and checking the Parity Error flag (Bus and Status Register, bit 5).

The proper values for the Bus and Status Register and the Current SCSI Bus Status Register are displayed in Figures 20 and 21, respectively.

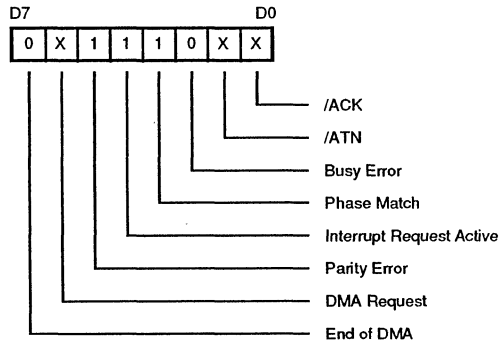


Figure 20. Bus and Status Register

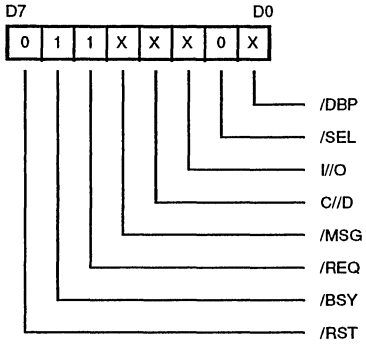


Figure 21. Current SCSI Bus Status Register

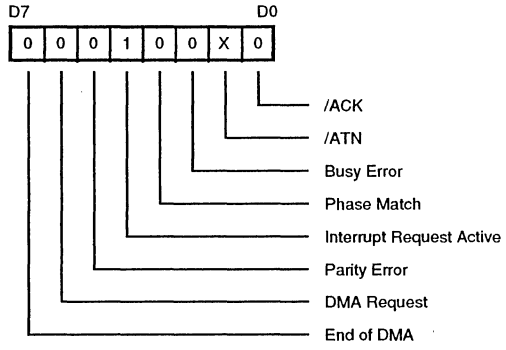


Figure 22. Bus and Status Register

Bus Phase Mismatch Interrupt

The SCSI phase lines are comprised of the signals I//O, C//D, and /MSG. These signals are compared with the corresponding bits in the Target Command Register: Assert I//O (bit 0), Assert C//D (bit 1), and Assert /MSG (bit 2). The comparison occurs continually and is reflected in the Phase Match bit (bit 3) of the Bus and Status Register. If the DMA Mode bit (Mode Register, bit 1) is active and a phase mismatch occurs when /REQ transitions from False to True, an interrupt (IRQ) is generated.

A phase mismatch prevents the recognition of /REQ and removes the chip from the bus during an Initiator send operation (/DB7-/DB0 and /DBP will not be driven even through the Assert Data Bus bit (Initiator Command Register, bit 0) is active). This may be disabled by resetting the DMA Mode bit (Note: It is possible for this interrupt to occur when connected as a Target if another device is driving the phase lines to a different state).

The proper values for the Bus and Status Register and the Current SCSI Bus Status Register are displayed in Figures 22 and 23, respectively.

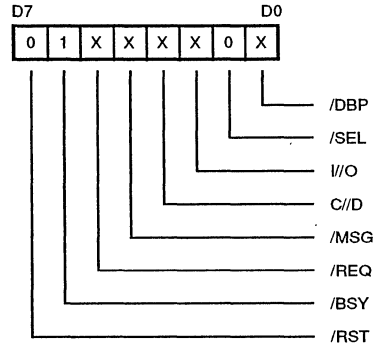


Figure 23. Current SCSI Bus Status Register

Loss of BSY Interrupt

If the Monitor Busy bit (bit 2) in the Mode Register is active, an interrupt is generated if the BSY signal goes False for at least a bus-settle delay. This interrupt is disabled by resetting the Monitor Busy bit. Register values are displayed in Figures 24 and 25.

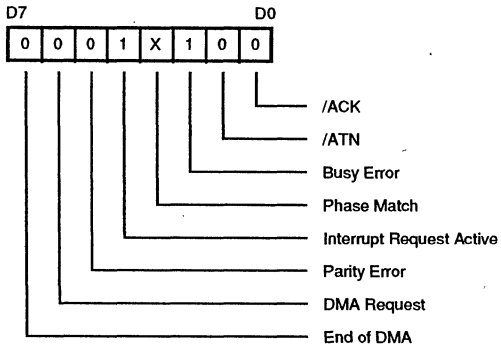


Figure 24. Bus and Status Register

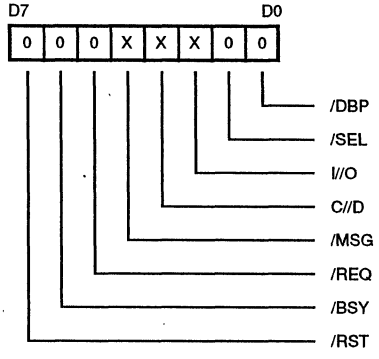


Figure 25. Current SCSI Bus Status Register

Reset Conditions

Three possible reset situations exist with the Z5380, as follows:

Hardware Chip Reset

When the signal /RST is active for at least 200 ns, the Z5380 device is re-initialized and all internal logic and control registers are cleared. This is a chip reset only and does not create a SCSI Bus-Reset condition.

SCSI Bus Reset (/RST) Received

When a SCSI /RST signal is received, an IRQ interrupt is generated and a chip reset is performed. All internal logic and registers are cleared, except for the IRQ interrupt latch and the Assert /RST bit (bit 7) in the Initiator Command Register. (Note: The /RST signal may be sampled by reading the Current SCSI Bus Status Register; however, this signal is not latched and may not be present when this port is read).

SCSI Bus Reset (/RST) Issued

If the CPU sets the Assert /RST bit (bit 7) in the Initiator Command Register, the /RST signal goes active on the SCSI Bus and an internal reset is performed. Again, all internal logic and registers are cleared except for the IRQ interrupt latch and the Assert /RST bit (bit 7) in the Initiator Command Register. The /RST signal will continue to be active until the Assert /RST bit is reset or until a hardware reset occurs.

Data Transfers

Data is transferred between SCSI Bus devices in one of four modes (Reference Figures 26-41):

1. Programmed I/O
2. Normal DMA
3. Block Mode DMA
4. Pseudo DMA

The following sections describe these modes in detail (**Note:** For all data transfer operations, /DACK and /CS should never be active simultaneously).

Programmed I/O Transfers

Programmed I/O is the most primitive form of data transfer. The /REQ and /ACK handshake signals are individually monitored and asserted by reading and writing the appropriate register bits. This type of transfer is normally used when transferring small blocks of data such as command blocks or message and status bytes. An Initiator send operation would begin by setting the C//D, I//O, and /MSG bits in the Target Command Register to the correct state so that a phase match exists. In addition to the phase match condition, it is necessary for the Assert Data Bus bit (Initiator Command Register, bit 0) to be True and the received I/O signal to be False for the Z5380 to send data. For each transfer, the data is loaded into the Output Data Register. The CPU then waits for the /REQ bit (Current SCSI Bus Status Register, bit 5) to become active. Once /REQ goes active, the Phase Match bit (Bus and Status Register, bit 3) is checked and the Assert /ACK bit (Initiator Command Register, bit 4) is set. The /REQ bit is sampled until it becomes False and the CPU resets the Assert /ACK bit to complete the transfer.

Normal DMA Mode

DMA transfers are normally used for large block transfers. The SCSI chip outputs a DMA request (DRQ) whenever it is ready for a byte transfer. External DMA logic uses this

DRQ signal to generate /DACK and an /IOR or an /IOW pulse to the Z5380. DRQ goes inactive when /DACK is asserted and /DACK goes inactive some time after the minimum read or write pulse width. This process is repeated for every byte. For this mode, /DACK should not be allowed to cycle unless a transfer is taking place.

Block Mode DMA

Some popular DMA Controllers, such as the 9517A, provide a Block Mode DMA transfer. This type of transfer allows the DMA controller to transfer blocks of data without relinquishing the use of the Data Bus to the CPU after each byte is transferred; thus, faster transfer rates are achieved by eliminating the repetitive access and release of the CPU Bus. If the Block Mode DMA bit (Mode Register, bit 7) is active, the Z5380 begins the transfer by asserting DRQ. The DMA controller then asserts /DACK for the remainder of the block transfer. DRQ goes inactive for the duration of the transfer. The Ready output is used to control the transfer rate. Non-Block Mode DMA transfers end when /DACK goes False, whereas Block Mode DMA transfers end when /IOR or /IOW becomes inactive. Since this is the case, DMA transfers may be started sooner in a Block Mode transfer. To obtain optimum performance in Block Mode operation, the DMA logic optionally uses the normal DMA mode interlocking handshake. Ready is still available to throttle the DMA transfer, but DRQ is 30 to 40 ns faster than Ready and is used to start the cycle sooner. The methods described under "Halting a DMA Operation" apply for all DMA operations.

Pseudo DMA Mode

To avoid the tedium of monitoring and asserting the request/acknowledgement handshake signals for programmed I/O transfers, the system can be designed to implement a pseudo DMA mode. This mode is implemented by programming the Z5380 to operate in the DMA mode, but using the CPU to emulate the DMA handshake. DRQ may be detected by polling the DMA Request bit (bit 6) in the Bus and Status Register, by sampling the signal through an external port, or by using it to generate a CPU interrupt. Once DRQ is detected, the CPU can perform a read or write data transfer. This CPU read/write is externally decoded to generate the appropriate /DACK and /IOR or /IOW signals.

Often, external decoding logic is necessary to generate the Z5380 /CS signal. This same logic may be used to generate /DACK at no extra cost and provide an increased performance in programmed I/O transfers.

Halting a DMA Operation

The /EOP signal is not the only way to halt a DMA transfer. A bus phase mismatch or a reset of the DMA Mode bit (Mode Register, bit 1) can also terminate a DMA cycle for the current bus phase.

Using the /EOP Signal

If /EOP is used, it should be asserted for at least 100ns while /DACK and /IOR or /IOW are simultaneously active. Note, however, that if /IOR or /IOW is not active, an interrupt is generated, but the DMA activity continues. The /EOP signal does not reset the DMA Mode bit. Since the /EOP signal can occur during the last byte sent to the Output Data Register, the /REQ and /ACK signals are monitored to ensure that the last byte has transferred.

Bus Phase Mismatch Interrupt

A bus phase mismatch interrupt is used to halt the transfer if operating as an Initiator. Using this method frees the host from maintaining a data length counter and frees the DMA logic from providing the /EOP signal. If performing an Initiator send operation, the Z5380 requires /DACK to cycle before /ACK goes inactive. Since phase changes cannot occur if /ACK is active, either /DACK must be cycled after the last byte is sent or the DMA Mode bit must be reset in order to receive the phase mismatch interrupt.

Resetting the DMA Mode Bit

A DMA operation may be halted at any time simply by resetting the DMA Mode bit. It is recommended that the DMA Mode bit be reset after receiving an /EOP or bus phase-mismatch interrupt. The DMA Mode bit must then be set before writing any of the start DMA registers for subsequent bus phases.

If resetting the DMA Mode bit is used instead of /EOP for Target role operation, then care must be taken to reset this bit at the proper time. If receiving data as a Target device, the DMA Mode bit must be reset once the last DRQ is received and before /DACK is asserted to prevent an additional /REQ from occurring. Resetting this bit causes DRQ to go inactive. However, the last byte received remains in the Input Data Register and may be obtained either by performing a normal CPU read or by cycling /DACK and /IOR. In most cases, /EOP is easier to use when operating as a Target device.

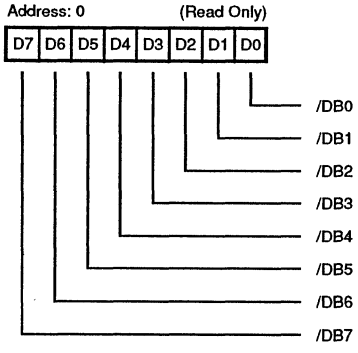


Figure 26. Current SCSI Data Register

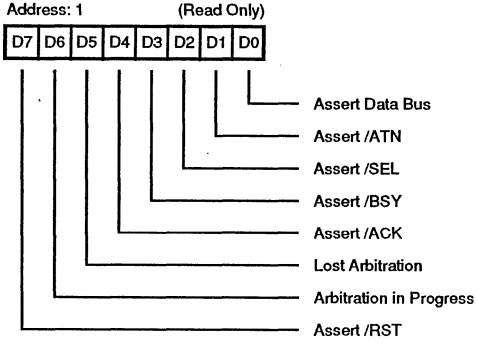


Figure 27. Initiator Command Register

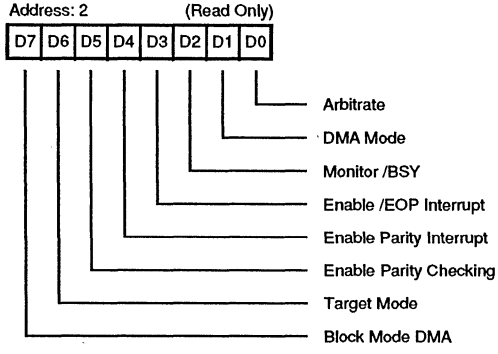


Figure 28. Mode Register

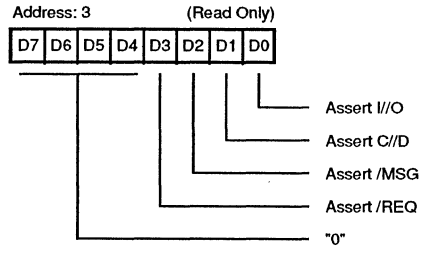


Figure 29. Target Command Register

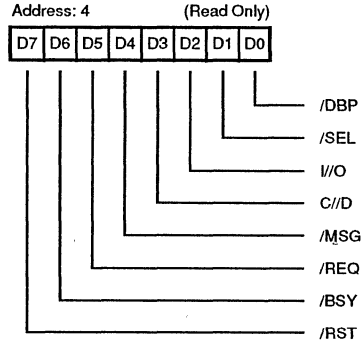


Figure 30. Current SCSI Bus Status Register

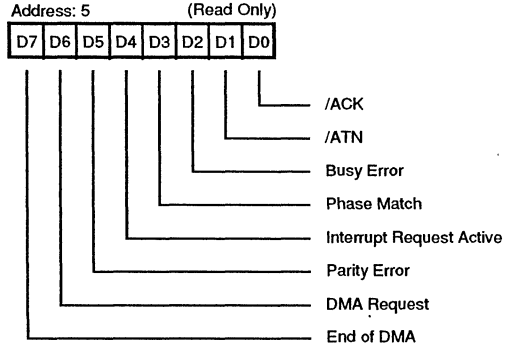


Figure 31. Bus and Status Register

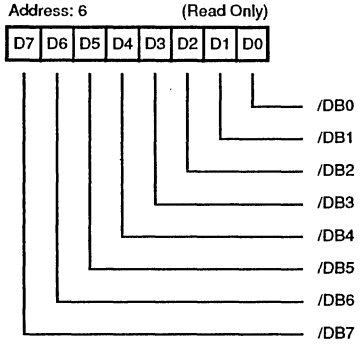


Figure 32. Input Data Register

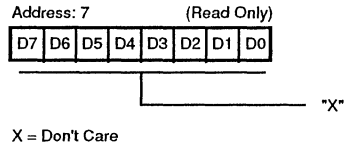


Figure 33. Reset Parity/Interrupt

WRITE REGISTERS

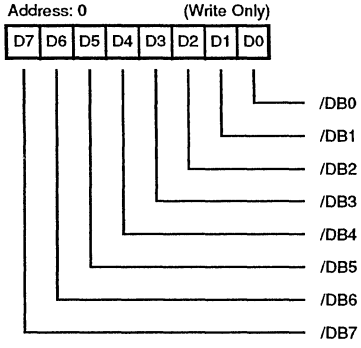


Figure 34. Output Data Register

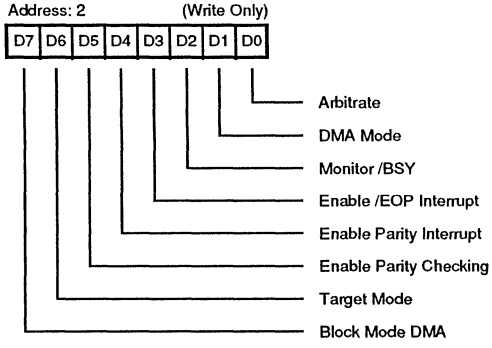


Figure 36. Mode Register

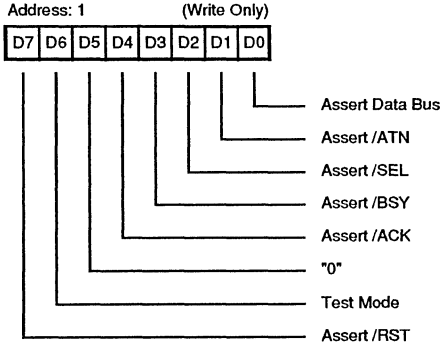


Figure 35. Initiator Command Register

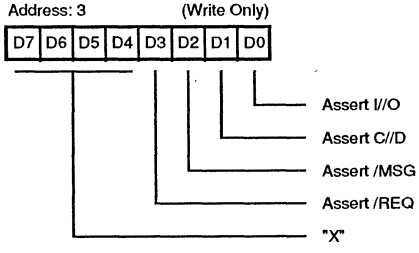


Figure 37. Target Command Register

WRITE REGISTERS (Continued)

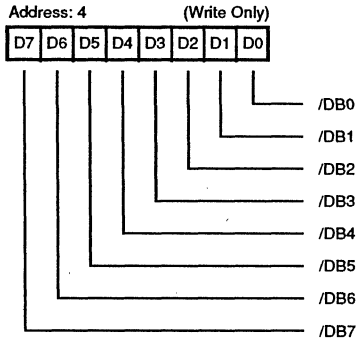


Figure 38. Select Enable Register

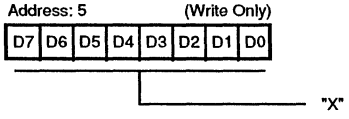


Figure 39. Start DMA Send

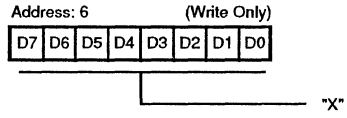


Figure 40. Start DMA Target Receive

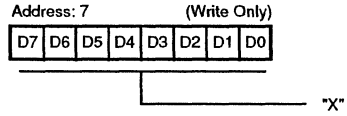


Figure 41. Start DMA Initiator Receive

ABSOLUTE MAXIMUM RATINGS

Voltages on all pins
with respect to GND ..... -0.3V to +7.0V
Operating Ambient Temperature ..... †
Storage Temperature ..... -65°C to +150°C

Note:
† See Ordering Information

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

STANDARD TEST CONDITIONS

The DC Characteristics section below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows (Figures 42 and 43):

- $+4.5V < V_{CC} < +5.5V$
- GND = 0V
- T_A as specified in Ordering Information

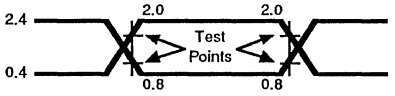


Figure 42. Switching Test Circuit

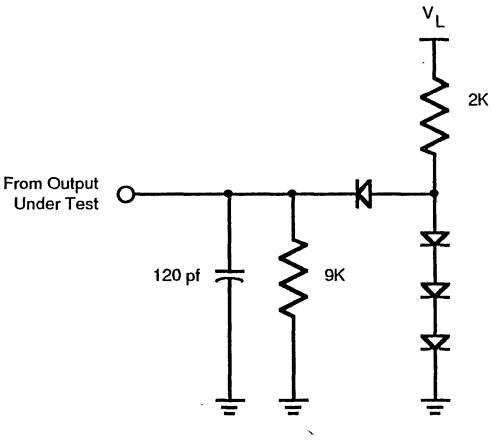


Figure 43. Standard Test Load

DC CHARACTERISTICS
Z5380

Symbol	Parameter	Conditions	Min	Max	Units
V_{DD}	Supply Voltage		4.75	5.25	V
V_{IH}	High-Level Input Voltage		2.0	5.25	V
V_{IL}	Low-Level Input Voltage		-0.3	0.8	V
$I_{IH}1$	High-Level Input Current	V_{IH} = 5.25V			
	SCSI Bus Pins	V_{IL} = 0V		50	µA
$I_{IH}2$	High-Level Input Current	V_{IH} = 5.25V			
	All Other Pins	V_{IL} = 0V		10	µA
$I_{IL}1$	Low-Level Input Current	V_{IH} = 5.25V			
	SCSI Bus Pins	V_{IL} = 0V	-50		µA
$I_{IL}2$	Low-Level Input Current	V_{IH} = 5.25V			
	All Other Pins	V_{IL} = 0V	-10 µA		
V_{OH}	High-Level Output Voltage	I_{OH} = -3mA	2.4		V
		V_{DD} = 4.75V			
$V_{OL}1$	Low-Level Output Voltage	I_{OL} = 48mA	0.5		V
	SCSI Bus Pins	V_{DD} = 4.75V			
$V_{OL}2$	Low-Level Output Voltage	I_{OL} = 7mA	0.5		V
	All Other Pins	V_{DD} = 4.75V			
I_{DD}	Supply Current	15 mA			
T_A	Operating Free-Air		0	70	C

AC CHARACTERISTICS
CPU Write Cycle Timing Diagram

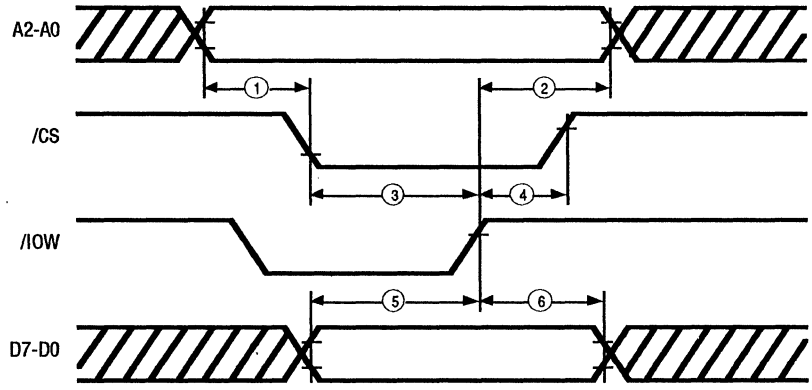


Figure 44. CPU Write Cycle

AC CHARACTERISTICS
CPU Write Cycle Timing Table

No	Description	Min	Max	Units
1	Address Setup to Write Enable[1]	20		ns
2	Address Hold from End Write Enable[1]	20		ns
3	Write Enable Width[1]	70		ns
4	Chip Select Hold from End of /IOW	0		ns
5	Data Setup to end of Write Enable[1]	50		ns
6	Data Hold Time form End of /IOW	30		ns

Note:
[1] Write Enable is the occurrence of /IOW and /CS

AC CHARACTERISTICS
CPU Read Cycle Timing Diagram

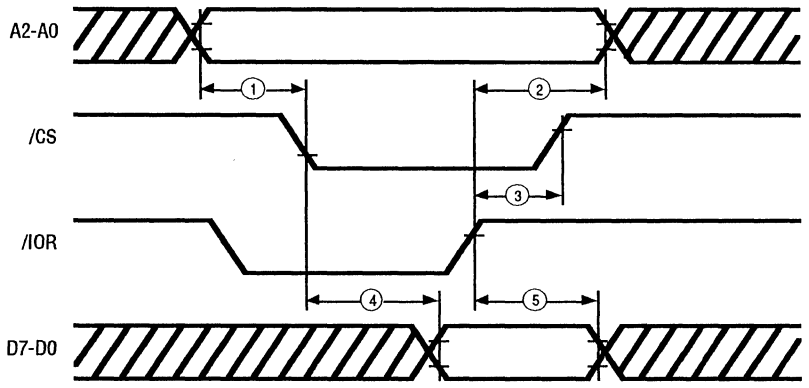


Figure 45. CPU Read Cycle

AC CHARACTERISTICS
CPU Read Cycle Timing Table

No	Description	Min	Max	Units
1	Address Setup to Read Enable[1]	20		ns
2	Address Hold from End Read Enable[1]	20		ns
3	Chip Select Hold from End of /IOR	0		ns
4	Data Access Time from Read Enable[1]	130		ns
5	Data Hold Time from End of Read Enable[1]	20		ns

Note:
[1] Read Enable is the occurrence of /IOR and /CS.

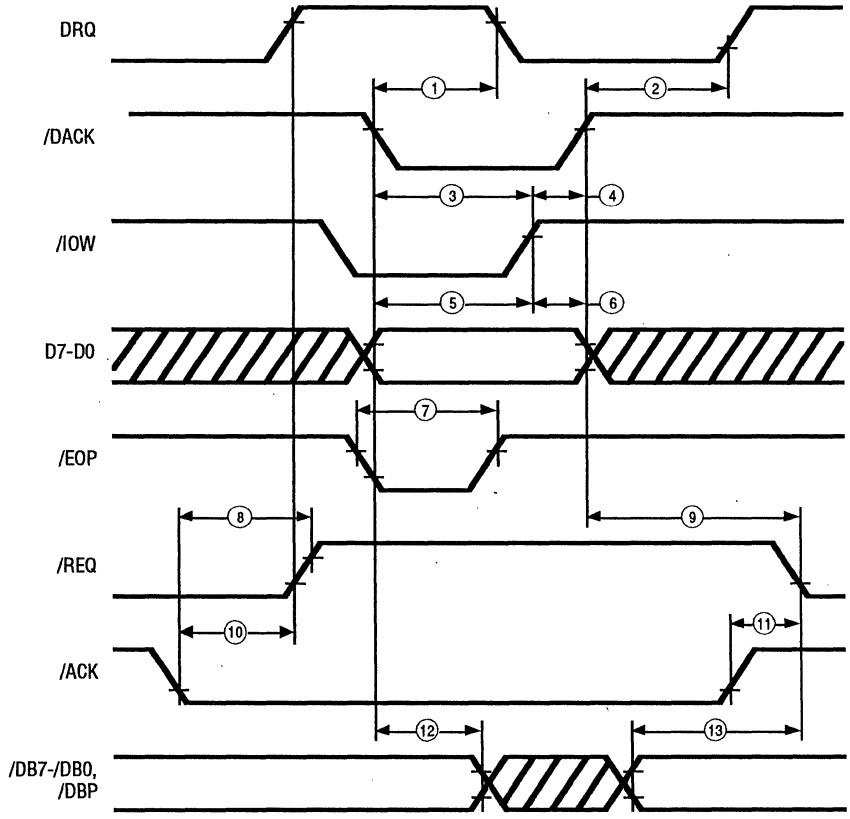


Figure 46. DMA Write (Non-Block Mode) Target Send Cycle

AC CHARACTERISTICS

DMA Write (Non-Block Mode) Target Send Cycle Table

No	Description	Min	Max	Units
1	DRQ Low from /DACK Low	130		ns
2	/DACK High to DRQ High	30		ns
3	Write Enable Width[1]	100		ns
4	/DACK Hold from /IOW High	0		ns
5	Data Setup to End of Write Enable[1]	50		ns
6	Data Hold Time from End of /IOW	40		ns
7	Width of /EOP Pulse[2]	100		ns
8	/ACK Low to /REQ High	25	125	ns
9	/REQ from End of /DACK (/ACK High)	30	150	ns
10	/ACK Low to DRQ High (Target)	15	110	ns
11	/ACK High to /REQ Low (/DACK High)	20	150	ns
12	Data Hold from Write Enable	15		ns
13	Data Setup to /REQ Low (Target)	60		ns

Notes:

[1] Write Enable is the occurrence of /IOW and /DACK.

[2] /EOP, /IOW, and /DACK must be concurrently Low for at least T7 for proper recognition of the /EOP pulse.

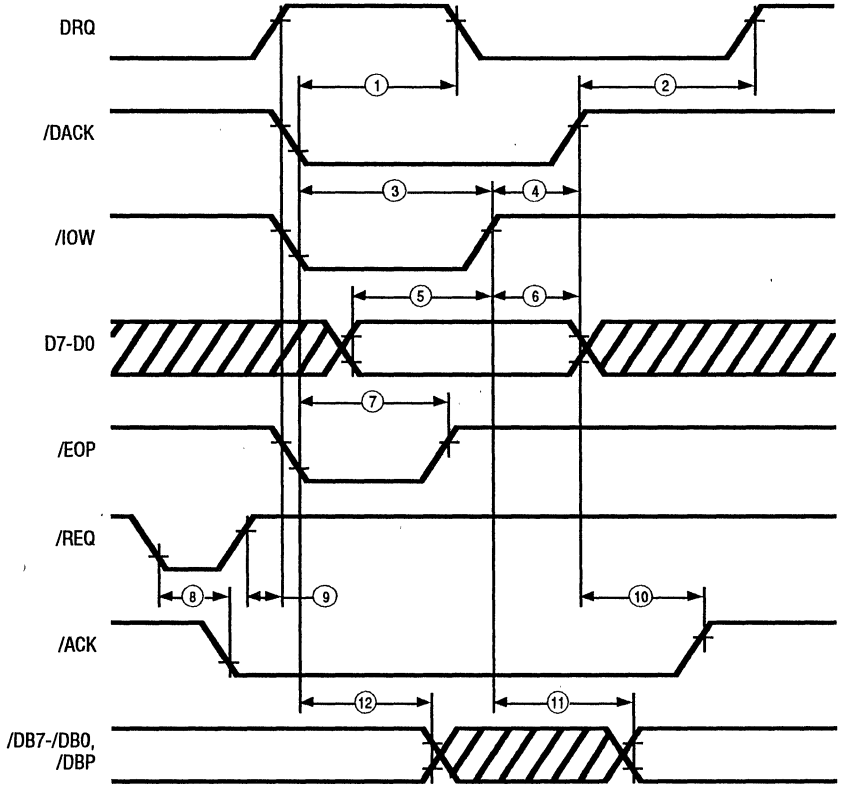


Figure 47. DMA Write (Non-Block Mode) Initiator Send Cycle

AC CHARACTERISTICS

DMA Write (Non-Block Mode) Initiator Send Cycle Table

No	Description	Min	Max	Units
1	DRQ Low from /DACK Low	130		ns
2	/DACK High to DRQ High	30		ns
3	Write Enable Width[1]	100		ns
4	/DACK Hold from End of /IOW	0		ns
5	Data Setup to End of Write Enable[1]	50		ns
6	Data Hold Time from End of /IOW	40		ns
7	Width of /EOP Pulse[2]	100		ns
8	/REQ Low to /ACK Low	20	160	ns
9	/REQ High to DRQ High	20	110	ns
10	/DACK High to /ACK High	25	150	ns
11	/IOW High to Valid SCSI Data	100		ns
12	Data Hold from Write Enable[1]	15		ns

Notes:

[1] Write Enable is the occurrence of /IOW and /DACK.

[2] /EOP, /IOW, and /DACK must be concurrently Low for at least T7 for proper recognition of the /EOP pulse.

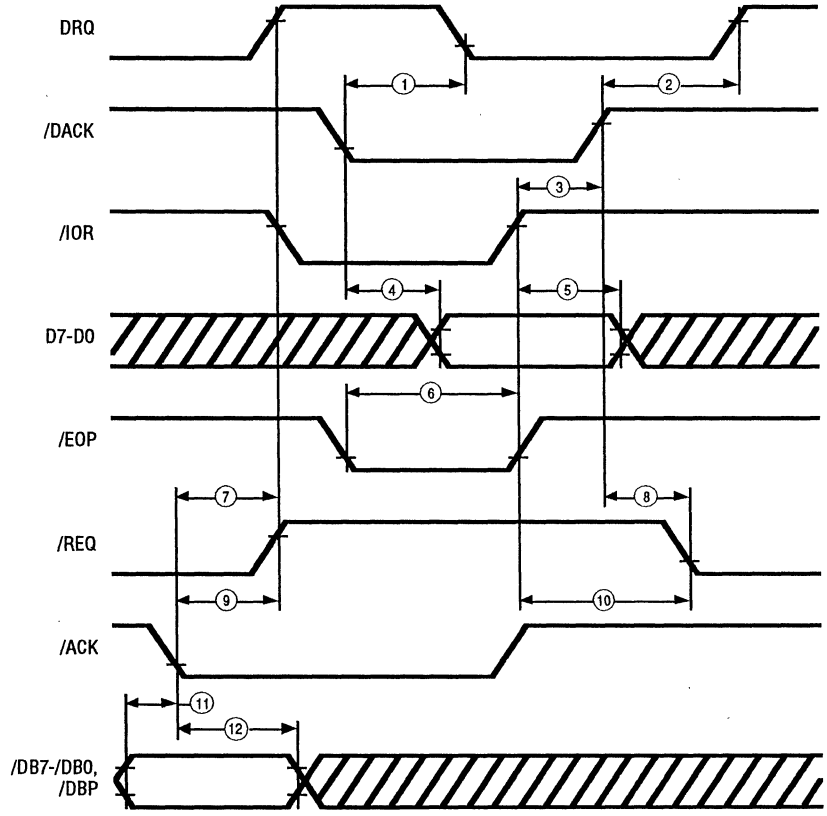


Figure 48. DMA Read (Non-Block Mode) Target Receive Cycle

AC CHARACTERISTICS

DMA Read (Non-Block Mode) Target Receive Cycle Table

No	Description	Min	Max	Units
1	DRQ Low from /DACK Low	130		ns
2	/DACK High to DRQ High	30		ns
3	/DACK Hold Time from End of /IOR	0		ns
4	Data Access Time from Read Enable[1]	115		ns
5	Data Hold Time from End of /IOR	20		ns
6	Width of /EOP Pulse[2]	100		ns
7	/ACK Low to DRQ High	15	110	ns
8	/DACK High to /REQ Low (/ACK High)	30	150	ns
9	/ACK Low to /REQ High	25	125	ns
10	/ACK High to /REQ Low (/DACK High)	20	150	ns
11	Data Setup Time to /ACK	20		ns
12	Data Hold Time from /ACK	50		ns

Notes:

[1] Read Enable is the occurrence of /IOR and /DACK.

[2] /EOP, /IOR, and /DACK must be concurrently Low for at least T6 for proper recognition of the /EOP pulse.

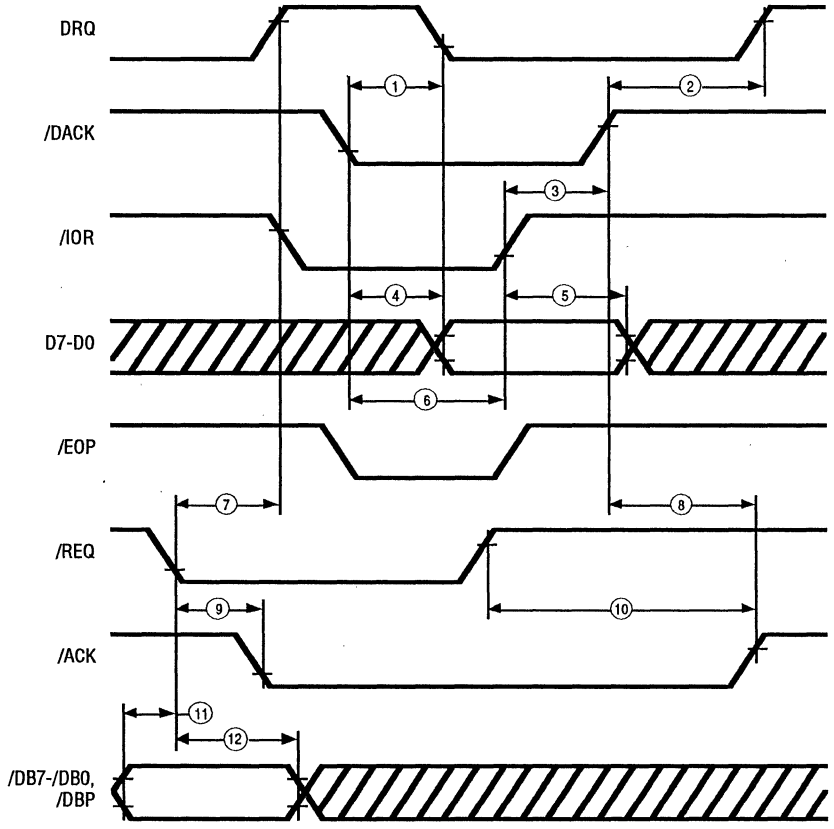


Figure 49. DMA Read (Non-Block Mode) Initiator Receive Cycle

AC CHARACTERISTICS

DMA Read (Non-Block Mode) Initiator Receive Cycle Table

No	Description	Min	Max	Units
1	DRQ Low from /DACK Low	130		ns
2	/DACK High to DRQ High	30		ns
3	/DACK Hold Time from End of /IOR	0		ns
4	Data Access Time from Read Enable[1]	115		ns
5	Data Hold Time from End of /IOR	20		ns
6	Width of /EOP Pulse[2]	100		ns
7	/REQ Low to DRQ High	20		ns
8	/DACK High to /ACK High (/REQ High)	25	160	ns
9	/REQ Low to /ACK Low	20	160	ns
10	/REQ High to /ACK High (/DACK High)	15	140	ns
11	Data Setup Time to /REQ	20		ns
12	Data Hold Time from /REQ	50		ns

Notes:

[1] Read Enable is the occurrence of /IOR and /DACK.

[2] /EOP, /IOR, and /DACK must be concurrently Low for at least T6 for proper recognition of the /EOP pulse.

AC CHARACTERISTICS

DMA Write (Block Mode) Target Send Cycle Timing Diagram

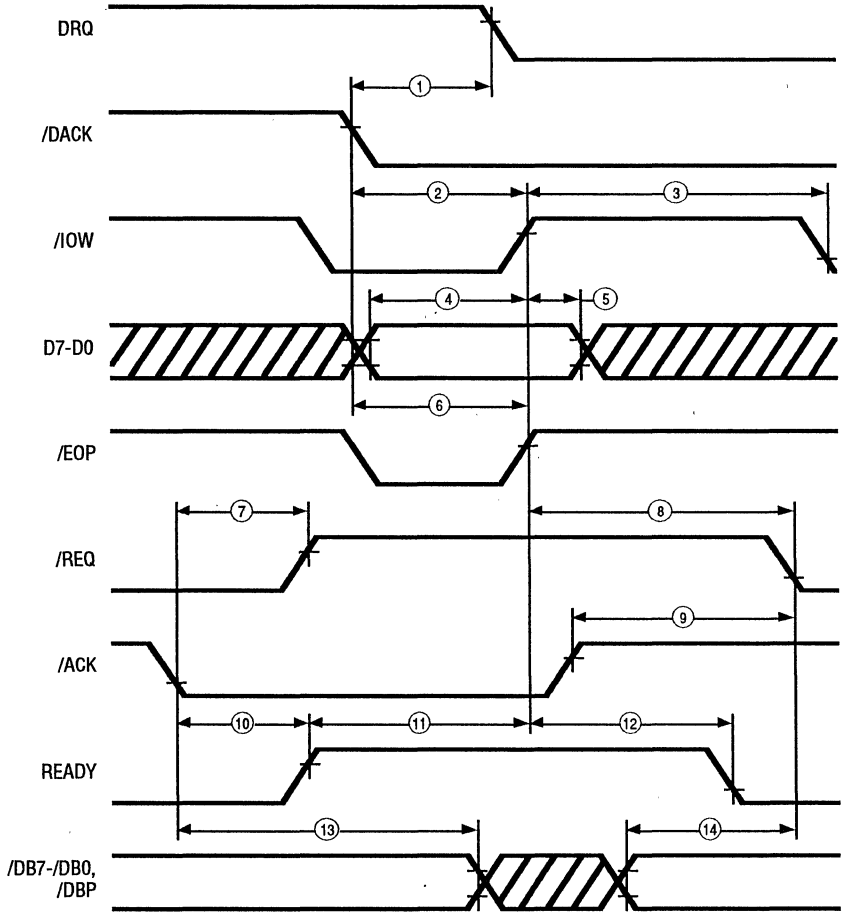


Figure 50. DMA Write (Block Mode) Target Send Cycle

AC CHARACTERISTICS

DMA Write (Block Mode) Target Send Cycle Table

No	Description	Min	Max	Units
1	DRQ Low from /DACK Low	130		ns
2	Write Enable Width[1]	100		ns
3	Write Recovery Time	120		ns
4	Data Setup to End of Write Enable[1]	50		ns
5	Data Hold Time from End of /IOW	40		ns
6	Width of /EOP Pulse[2]	100		ns
7	/ACK Low to /REQ High	25	125	ns
8	/REQ from End of /IOW (/ACK High)	40	180	ns
9	/REQ from End of /ACK (/IOW High)	20	170	ns
10	/ACK Low to READY High	20	140	ns
11	READY High to /IOW High	70		ns
12	/IOW High to READY Low	20	140	ns
13	Data Hold from /ACK Low	40		ns
14	Data Setup to /REQ Low	60		ns

Notes:

[1] Write Enable is the occurrence of /IOW and /DACK.

[2] /EOP, /IOW, and /DACK must be concurrently Low for at least T6 for proper recognition of the /EOP pulse.

AC CHARACTERISTICS

DMA Read (Block Mode) Target Receive Cycle Timing Diagram

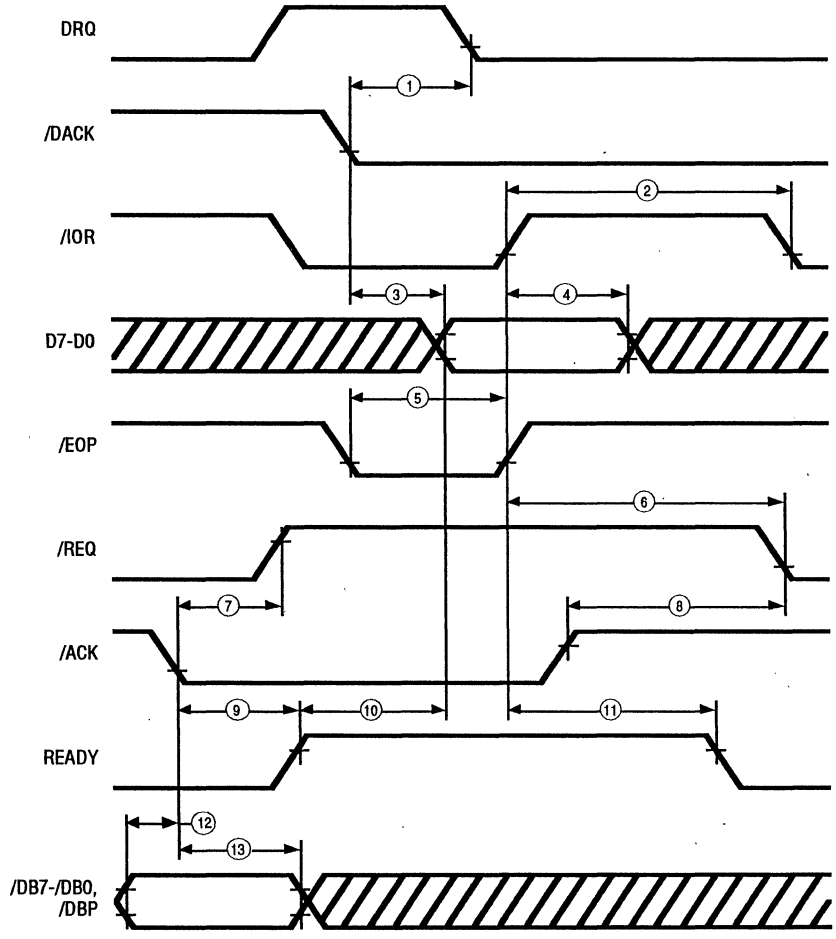


Figure 51. DMA Read (Block Mode) Target Receive Cycle

AC CHARACTERISTICS

DMA Read (Block Mode) Target Receive Cycle Table

No	Description	Min	Max	Units
1	DRQ Low from /DACK Low	130		ns
2	/IOR Recovery Time	120		ns
3	Data Access Time from Read Enable[1]	110		ns
4	Data Hold Time from End of /IOR	20		ns
5	Width of /EOP Pulse[2]	100		ns
6	/IOR High to /REQ Low	30	190	ns
7	/ACK Low to /REQ High	25	125	ns
8	/ACK High to /REQ Low (/IOR High)	20	170	ns
9	/ACK Low to READY High	20	140	ns
10	READY High to Valid Data	50		ns
11	/IOR High to READY Low	20	140	ns
12	Data Setup Time to /ACK	20		ns
13	Data Hold Time from /ACK	50		ns

Notes:

[1] Read Enable is the occurrence of /IOR and /DACK.

[2] /EOP, /IOR, and /DACK must be concurrently Low for at least T5 for proper recognition of the /EOP pulse.

AC CHARACTERISTICS
Arbitration

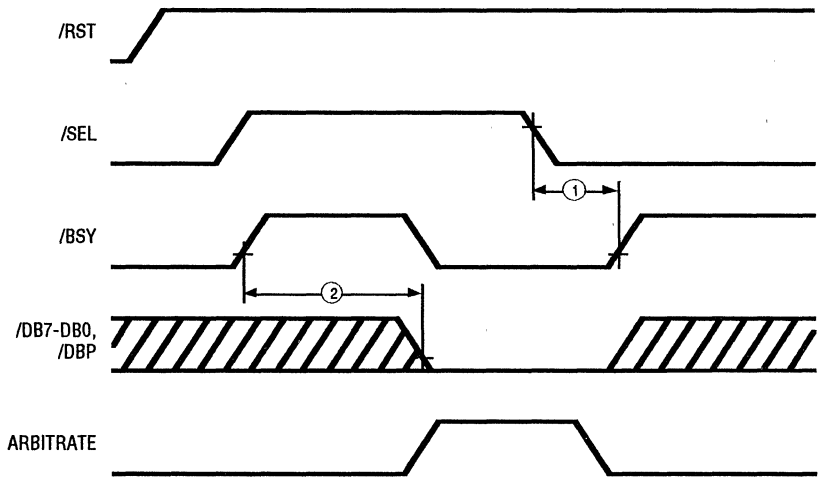


Figure 52. Arbitration

No	Description	Min	Max	Units
1	Bus Clear from /SEL Low		600	ns
2	Arbitrate Start from /BSY High	1200	2200	ns

AC CHARACTERISTICS
Reset

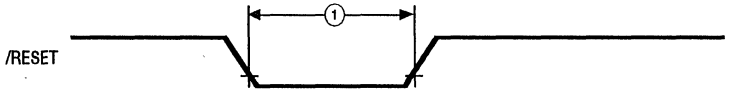


Figure 53. Reset

No	Description	Min	Max	Units
1	Minimum Width of /RESET	200		ns

Z5380 NOTES

1. Edge-triggered /RST Interrupt - If the SCSI Bus is not terminated, the /RST interrupt is continually generated.

2. True End of DMA Interrupt - The Z5380 generates an interrupt when it receives the last byte from the DMA, not when the last byte is transferred to the SCSI Bus.

3. Return to Ready after /EOP Interrupt - When operating in Block Mode DMA, the Z5380 does not return the Ready signal to a Ready condition. This locks up the bus and prevents the CPU from executing.

4. SCSI handshake after /EOP occurs - If an EOP occurs when receiving data, a subsequent request will cause /ACK to be asserted even though no DRQ is issued.

5. Reselection Interrupt - During reselection, if the Target Command Register does not reflect the current bus phase (most likely Data Out), the reselection interrupt may get reset.

6. Phase Mismatch Interrupt - A phase mismatch interrupt is not guaranteed after a reselection for the following reasons:

DMA Mode bit must be set in order to receive a phase mismatch interrupt.

DMA Mode bit can not be set unless /BSY is active.

/BSY can not be asserted until after the reselection has occurred.

Once /BSY is asserted, the Target may assert /REQ in less than 500ns.

The phase mismatch interrupt is generated on the active edge of /REQ. If the DMA Mode bit is not set before the /REQ goes active, the phase mismatch interrupt will not occur.

ADVANCE INFORMATION SPECIFICATION

Z53C80

SMALL COMPUTER SYSTEM INTERFACE (SCSI)

FEATURES

- Supports 53C80 pinout
- Low power CMOS
- Asynchronous Interface, supports data transfers up to 3 Mbytes/sec
- Direct SCSI Bus Interface with On-Board 48 mA drivers
- Supports Target and Initiator roles
- Arbitration Support
- DMA or Programmed I/O Data Transfers
- Supports Normal or Block Mode DMA
- Memory or I/O Mapped CPU Interface

GENERAL DESCRIPTION

The Z53C80 SCSI (Small Computer System Interface) controller is a 44-pin PLCC or 48-pin DIP CMOS device. It is designed to implement the SCSI protocol as defined by the ANSI X3.131-1986 standard, and is fully compatible with the industry standard 53C80. It is capable of operating both as a Target and as an Initiator. Special high-current open-drain outputs enable it to directly interface to the SCSI bus. The Z53C80 has the necessary interface hook-ups so the system CPU can communicate with it like with any other peripheral device. The CPU can read from, or write to, the SCSI registers which are addressed as standard or memory-mapped I/Os.

The Z53C80 increases the system performance by minimizing the CPU intervention in DMA operations which the SCSI controls. The CPU is interrupted when it detects a bus condition that requires attention. It also supports arbitration and reselection. The Z53C80 has the proper handshake signals to support normal and block mode DMA operations with most DMA controllers available.

Figure 1 is the functional block diagram of the Z53C80. The pin functions of the Z53C80 are shown in Figure 2. The device is housed in a 48-pin DIP (Figure 3) and a 44-pin PLCC package (Figure 4).*

Note: All Signals with a preceding front slash, "/", are active Low, e.g.: B//W (WORD is active Low); /B/W (BYTE is active Low, only).

*Note: Power connections follow Conventional descriptions below

Connection	Circuit	Device
Power	V_{CC}	V_{DD}
Ground	GND	V_{SS}

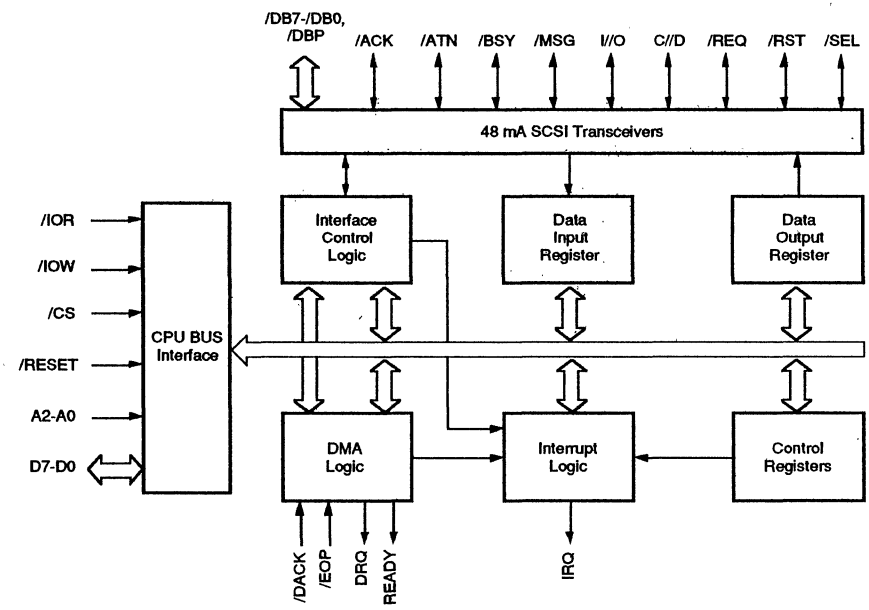


Figure 1. Block Diagram

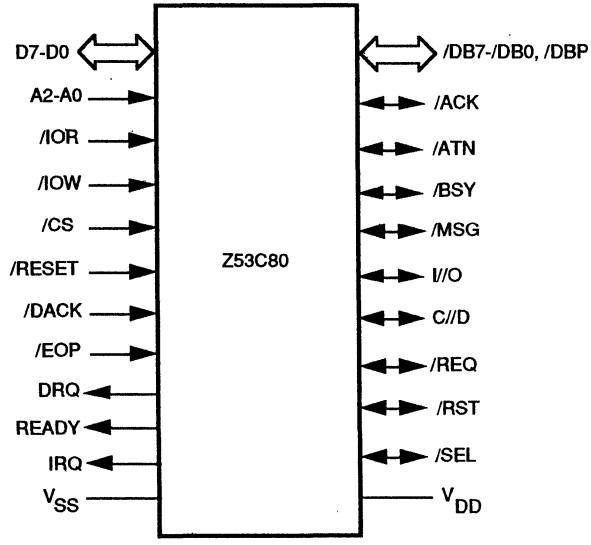


Figure 2. Pin Functions

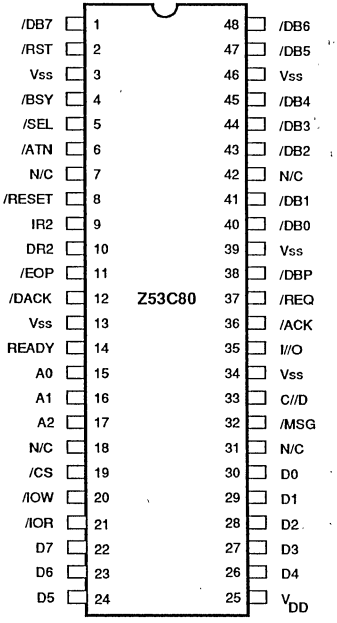


Figure 3. 40-Pin DIP Assignments

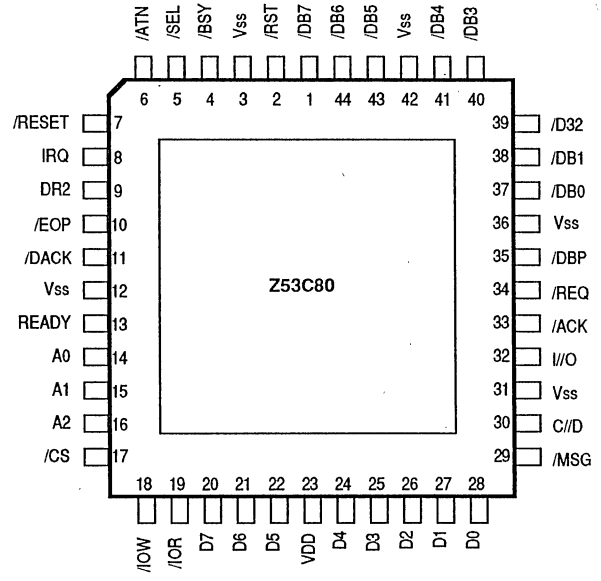


Figure 4. 44-Pin PLCC Pin Assignments

PIN DESCRIPTIONS

Microprocessor Bus

A2-A0. *Address Lines* (Input). Address lines are used with /CS, /IOR, and /IOW to address all internal registers.

/CS. *Chip Select* (Input, Active Low). This signal, in conjuction with /IOR or /IOW, enables the internal registers selected by A2-A0, to be read from or written to.

/DACK. *DMA Acknowledge* (Input, Active Low). /DACK resets DRQ and selects the data register for input or output data transfers. /DACK is used by DMA controller instead of /CS.

DRQ. *DMA Request* (Output, Active High). DRQ indicates that the data request is ready to be read or written. DRQ is asserted only if DMA mode is set in the Command Register. DRQ is cleared by /DACK.

D7-D0. *Data Lines* (Bidirectional, three-state, Active High). Bidirectional microprocessor data bus lines.

/EOP. *End of Process* (Input, Active Low). /EOP is used to terminate a DMA transfer. If asserted during a DMA cycle, the current byte will be transferred, but no additional bytes will be requested.

/IOR. *I/O Read* (Input, Active Low). /IOR is used in conjunction with /CS and A2-A0 to read an internal register. It also selects the Input Data Register when used with /DACK.

/IOW. *I/O Write* (Input, Active Low). /IOW is used in conjunction with /CS and A2-A0 to write to an internal register. It also selects the Output Data Register when used with /DACK.

IRQ. *Interrupt Request* (Output, Active High). IRQ alerts a microprocessor of an error condition or an event completion.

READY. *Ready* (Output, Active High). READY is used to control the speed of Block Mode DMA transfers. This signal goes active to indicate that the chip is ready to send/receive data and remains Low after a transfer until the last byte is sent or until the DMA Mode bit is reset.

/RESET. *Reset* (Input, Active Low). /RESET clears all registers. It has no effect upon the SCSI /RST signal.

SCSI Interface Signals

The following signals are all bi-directional, active low, open-drain signals with 48mA sink capability. All pins interface directly with the SCSI.Bus.

/ACK. *Acknowledge* (Bidirectional, Open-drain, Active Low). Driven by an Initiator, /ACK indicates an acknowledgment for a /REQ//ACK data-transfer handshake. In the Target role, /ACK is received as a response to the /REQ Signal.

/ATN. *Attention* (Bidirectional, Open-drain, Active Low). Driven by an Initiator, received by the Target, /ATN indicates an Attention condition.

/BSY. *Busy* (Bidirectional, Open-drain, Active Low). This signal indicates that the SCSI Bus is being used and can be driven by both the Initiator and the Target device.

C//D. *Control//Data.* (Bidirectional, Open-drain). Driven by the Target and received by the Initiator, C//D indicates whether Control or Data information is on the Data Bus. True indicates Control.

/DB7-/DB0, /DBP. *Data Bus Bits, Data Bus Parity Bit* (Bidirectional, Open-drain). These eight data bits (/DB7-/DB0), plus a parity bit (/DBP) form the data bus. /DB7 is the most significant bit (MSB) and has the highest priority during the Arbitration phase. Data parity is odd. Parity is always generated and optionally checked. Parity is not valid during Arbitration.

I//O. *Input//Output* (Bidirectional, Open-drain). I//O is a signal driven by a Target which controls the direction of data movement on the SCSI bus. True indicates input to the Initiator. This signal is also used to distinguish between Selection and Reselection phases.

/MSG. *Message* (Bidirectional, Open-drain, Active Low). This signal is driven by the Target during the Message phase. The signal is received by the Initiator.

/REQ. *Request* (Bidirectional, Open-drain, Active Low). Driven by a Target and received by the Initiator, this signal indicates a request for a /REQ//ACK data-transfer handshake.

/RST. *SCSI Bus Reset* (Bidirectional, Open-drain, Active Low). This signal indicates a SCSI Bus Reset condition.

/SEL. *Select* (Bidirectional, Open-drain, Active Low). This signal is used by an Initiator to select a Target, or by a Target to reselect an Initiator.

APPLICATION NOTES AND TECHNICAL ARTICLES

ZiLOG

ON-CHIP OSCILLATOR DESIGN

DESIGN AND BUILD RELIABLE, COST-EFFECTIVE, ON-CHIP OSCILLATOR CIRCUITS THAT ARE TROUBLE FREE. PUTTING OSCILLATOR THEORY INTO A PRACTICAL DESIGN MAKES FOR A MORE DEPENDABLE CHIP.

INTRODUCTION

This Application Note (App Note) is written for designers using Zilog Integrated Circuits with on-chip oscillators; circuits in which the amplifier portion of a feedback oscillator is contained on the IC. This App Note covers common theory of oscillators, and requirements of the circuitry (both internal and external to the IC) which comes from the theory for crystal and ceramic resonator based circuits.

Purpose and Benefits

The purposes and benefits of this App Note include:

1. Providing designers with greater understanding of how oscillators work and how to design them to avoid problems.

2. To eliminate field failures and other complications resulting from an unawareness of critical on-chip oscillator design constraints and requirements.

Problem Background

Inadequate understanding of the theory and practice of oscillator circuit design, especially concerning oscillator startup, has resulted in an unreliable design and subsequent field problems (See on page 10 for reference materials and acknowledgements).

OSCILLATOR THEORY OF OPERATION

The circuit under discussion is called the Pierce Oscillator (Figures 1, 2). The configuration used is in all Zilog on-chip oscillators. Advantages of this circuit are low power consumption, low cost, large output signal, low power level in the crystal, stability with respect to V_{cc} and temperature, and low impedances (not disturbed by stray effects). One drawback is the need for high gain in the amplifier to compensate for feedback path losses.

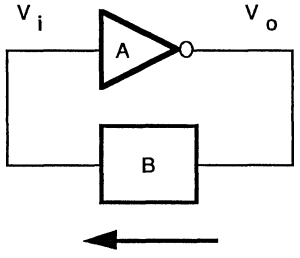


Figure 1. Basic Circuit and Loop Gain

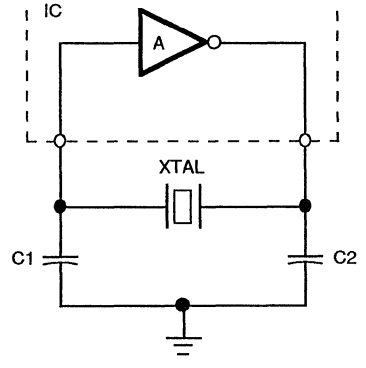


Figure 2. Zilog Pierce Oscillator

Pierce Oscillator (Feedback Type)

The basic circuit and loop gain is shown in Figure 1. The concept is straightforward; gain of the amplifier is A = Vo/Vi. The gain of the passive feedback element is B = Vi/Vo. Combining these equations gives the equality AB = 1. Therefore, the total gain around the loop is unity. Also, since the gain factors A and B are complex numbers, they have phase characteristics. It is clear that the total phase shift around the loop is forced to zero (i.e., 360 degrees), since V_{IN} must be in phase with itself. In this circuit, the amplifier ideally provides 180 degrees of phase shift (since it is an inverter). Hence, the feedback element is forced to provide the other 180 degrees of phase shift.

Additionally, these gain and phase characteristics of both the amplifier and the feedback element vary with frequency. Thus, the above relationships must apply at the frequency of interest. Also, in this circuit the amplifier is an active element and the feedback element is passive. Thus, by definition, the gain of the amplifier at frequency must be greater than unity, if the loop gain is to be unity.

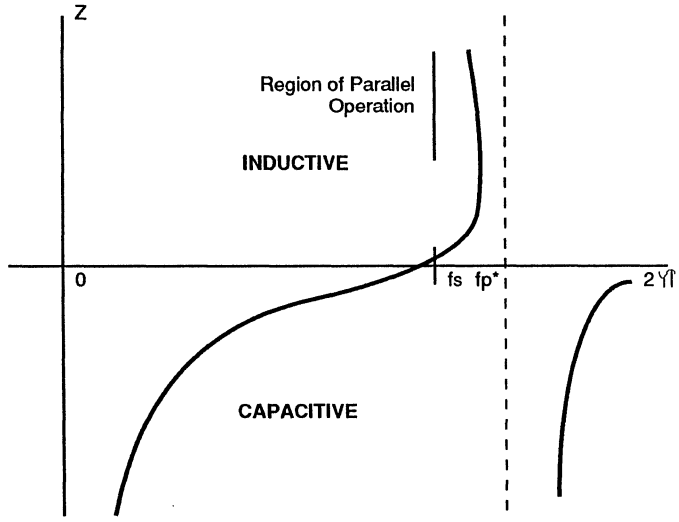
The described oscillator amplifies its own noise at startup until it settles at the frequency which satisfies the gain/phase requirement AB = 1. This means loop gain equals one, and loop phase equals zero (360 degrees). To do this, the loop gain at points around the frequency of oscillation must be greater than one. This achieves an average loop gain of one at the operating frequency.

The amplifier portion of the oscillator provides gain > 1 plus 180 degrees of phase shift. The feedback element provides the additional 180 degrees of phase shift without attenuating the loop gain to < 1. To do this the feedback element is inductive, i.e., it must have a positive reactance at the frequency of operation. The feedback elements discussed are quartz crystals and ceramic resonators.

Quartz Crystals

A quartz crystal is a piezoelectric device; one which transforms electrical energy to mechanical energy and vice versa. The transformation occurs at the resonant frequency of the crystal. This happens when the applied AC electric field is sympathetic in frequency with the mechanical resonance of the slice of quartz. Since this characteristic can be made very accurate, quartz crystals are normally used where frequency stability is critical. Typical frequency tolerance is .005 to 0.3%.

The advantage of a quartz crystal in this application is its wide range of positive reactance values (i.e., it looks inductive) over a narrow range of frequencies (Figure 3).



* fs - fp is very small (approximately 300 parts per million)

Figure 3. Series vs. Parallel Resonance

However, there are several ranges of frequencies where the reactance is positive; these are the fundamental (desired frequency of operation), and the third and fifth mechanical overtones (approximately 3 and 5 times the fundamental frequency). Since the desired frequency range in this application is always the fundamental, the overtones must be suppressed. This is done by reducing the loop gain at these frequencies. Usually, the amplifier's gain roll off, in combination with the crystal parasitics and load capacitors, is sufficient to reduce gain and prevent oscillation at the overtone frequencies.

The following parameters are for an equivalent circuit of a quartz crystal (Figure 4):

L - motional inductance (typ 120 mH @ 4 MHz)

C - motional capacitance (typ .01 pf @ 4 MHz)

R - motional resistance (typ 36 ohm @ 4 MHz)

Cs - shunt capacitance resulting from the sum of the capacitor formed by the electrodes (with the quartz as a dielectric) and the parasitics of the contact wires and holder (typ 3 pf @ 4 MHz).

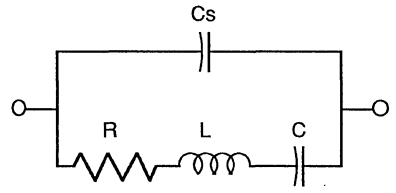
The series resonant frequency is given by:

Fs = $1/(2\pi \times \text{sqrt of LC})$,
where Xc and Xl are equal.

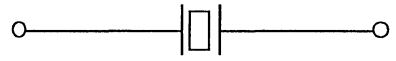
Thus, they cancel each other and the crystal is then R shunted by Cs with zero phase shift.

The parallel resonant frequency is given by:

Fp = $1/[2\pi \times \text{sqrt of L (C Ct/C+Ct)}]$,
where: $Ct = C_L + C_S$



Quartz Equivalent Circuit



Symbolic Representation

Figure 4. Quartz Oscillator

Series vs. Parallel Resonance. There is very little difference between series and parallel resonance frequencies (Figure 3). A series resonant crystal (operating at zero phase shift) is desired for non-inverting amplifiers. A parallel resonant crystal (operating at or near 180 degrees of phase shift) is desired for inverting amps. Figure 3 shows that the difference between these two operating modes is small. Actually, all crystals have operating points in both serial and parallel modes. A series resonant circuit will NOT have load caps C1 and C2. A data sheet for a crystal designed for series operation does not have a load cap spec. A parallel resonant crystal data sheet specifies a load cap value which is the series combination of C1 and C2. For this App Note discussion, since all the circuits of interest are inverting amplifier based, only the parallel mode of operation is considered.

OSCILLATOR THEORY OF OPERATION

Ceramic Resonators

Ceramic resonators are similar to quartz crystals, but are used where frequency stability is less critical and low cost is desired. They operate on the same basic principle as quartz crystals as they are piezoelectric devices and have a similar equivalent circuit. The frequency tolerance is wider (0.3 to 3%), but the ceramic costs less than quartz.

Figure 5 shows reactance vs. frequency and Figure 6 shows the equivalent circuit.

Typical values of parameters are L = .092 mH, C = 4.6 pf, R = 7 ohms and Cs = 40 pf, all at 8 MHz. Generally, ceramic resonators tend to start up faster but have looser frequency tolerance than quartz. This means that external circuit parameters are more critical with resonators.

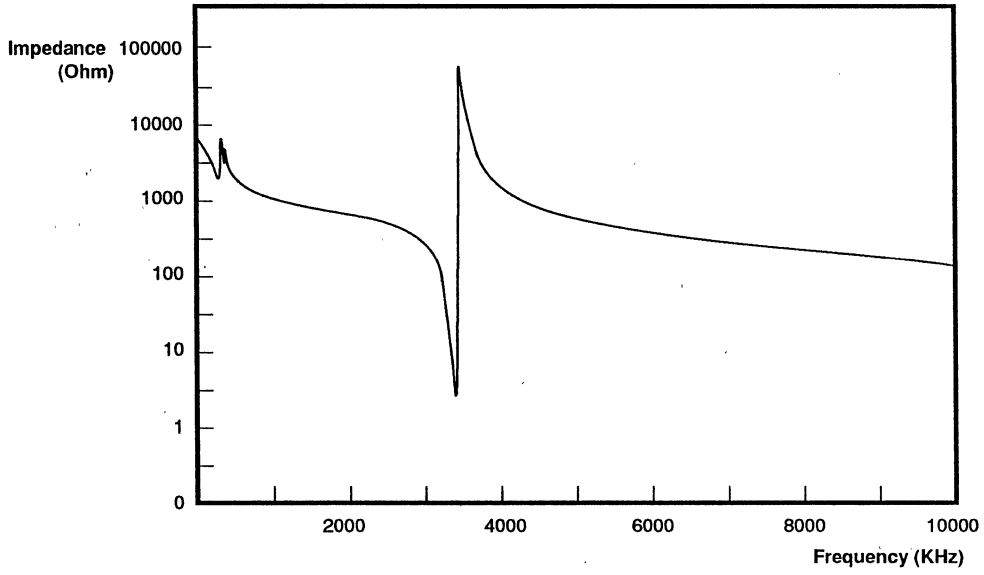


Figure 5. Ceramic Resonator Reactance

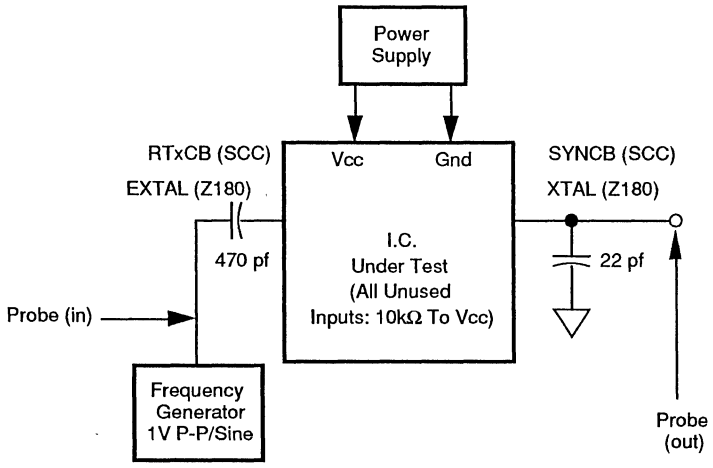


Figure 6. Gain Measurement

Load Capacitors

The effects/purposes of the load caps are:

Cap C2 combined with the amp output resistance provides a small phase shift. It also provides some attenuation of overtones.

Cap C1 combined with the crystal resistance provides additional phase shift.

These two phase shifts place the crystal in the parallel resonant region of Figure 3.

Crystal manufacturers specify a load capacitance number. This number is the load seen by the crystal which is the series combination of C1 and C2, including all parasitics (PCB and holder). This load is specified for crystals meant to be used in a parallel resonant configuration. The effect on startup time; if C1 and C2 increase, startup time increases to the point at which the oscillator will not start. Hence, for fast and reliable startup, over manufacture of large quantities, the load caps should be sized as low as possible without resulting in overtone operation.

Amplifier Characteristics

The following text discusses open loop gain vs. frequency, open loop phase vs. frequency, and internal bias.

Open Loop Gain vs. Frequency over lot, VCC, Processs Split, and Temp. Closed loop gain must be adequate to start the oscillator and keep it running at the desired frequency. This means that the amplifier open loop gain must be equal to one plus the gain required to overcome the losses in the feedback path, across the frequency band and up to the frequency of operation. This is over full process, lot, V_{CC}, and temperature ranges. Therefore, measuring the open loop gain is not sufficient; the losses in the feedback path (crystal and load caps) must be factored in.

Open Loop Phase vs. Frequency. Amplifier phase shift at and near the frequency of interest must be 180 degrees plus some, minus zero. The parallel configuration allows for some phase delay in the amplifier. The crystal adjusts to this by moving slightly down the reactance curve (Figure 3).

Internal Bias. Internal to the IC, there is a resistor placed from output to input of the amplifier. The purpose of this feedback is to bias the amplifier in its linear region and to provide the startup transition. Typical values are 1M to 20M ohms.

PRACTICE: CIRCUIT ELEMENT AND LAY OUT CONSIDERATIONS

The discussion now applies prior theory to the practical application.

Amplifier and Feedback Resistor

The elements of the circuit, internal to the IC, include the amplifier, feedback resistor, and output resistance. The amplifier is modeled as a transconductance amplifier with a gain specified as I_{OUT}/V_{IN} (amps per volt).

Transconductance/Gain. The loop gain AB = gm x Z1, where gm is amplifier transconductance (gain) in amps/volt and Z1 is the load seen by the output. AB must be greater than unity at and about the frequency of operation to sustain oscillation.

Gain Measurement Circuit. The gain of the amplifier can be measured using the circuits of Figures 6 & 7. This may be necessary to verify adequate gain at the frequency of interest and in determining design margin.

Gain Requirement vs. Temperature, Frequency and Supply Voltage. The gain to start and sustain oscillation (Figure 8) must comply with:

$gm > 4\pi^2 f^2 Rq\ C_{IN}\ C_{OUT} t \times M$
where: M is a quartz form factor $= (1 + C_{OUT}/C_{IN} + C_{OUT}/C_{OUT})^2$

Output Impedance. The output impedance limits power to the XTAL and provides small phase shift with load cap C2.

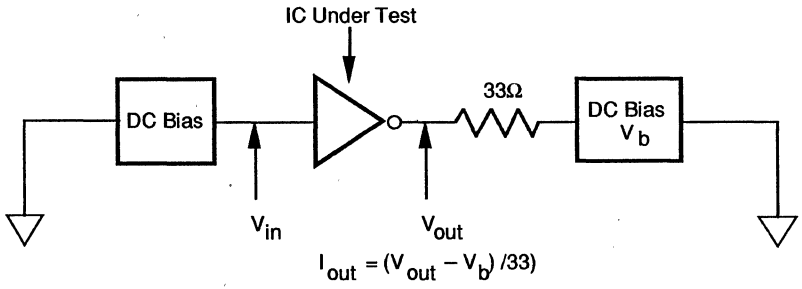
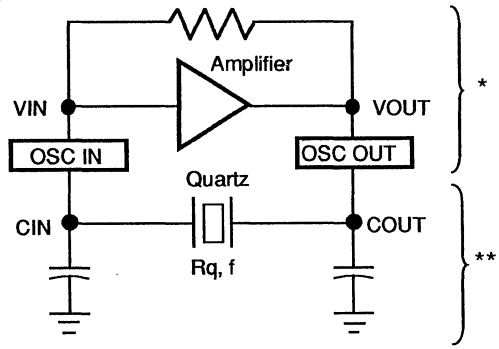


Figure 7. Transconductance (gm) Measurement



* Inside chip, feedback resistor biases the amplifier in the high gm region.

** External components typically: CIN = COUT = 30 to 50 pf (add 10 pf pin cap).

Figure 8. Quartz Oscillator Configuration

Load Capacitors

In the selection of load caps it is understood that parasitics are always included.

Upper Limits. If the load caps are too large, the oscillator will not start because the loop gain is too low at the operating frequency. This is due to the impedance of the load capacitors. Larger load caps produce a longer startup.

Lower Limits. If the load caps are too small, either the oscillator will not start (due to inadequate phase shift around the loop), or it will run at a 3rd, 5th, or 7th overtone frequency (due to inadequate suppression of higher overtones).

Capacitor Type and Tolerance. Ceramic caps of ±10% tolerance should be adequate for most applications.

Ceramic vs. Quartz. Manufacturers of ceramic resonators generally specify larger load cap values than quartz crystals. Quartz C is typically 15 to 30 pf and ceramic typically 100pf.

Summary. For reliable and fast startup, capacitors should be as small as possible without resulting in overtone operation. The selection of these capacitors is critical and all of the factors covered in this note should be considered.

Feedback Element

The following text describes the specific parameters of a typical crystal:

Drive Level. There is no problem at frequencies greater than 1 MHz and V_{cc} = 5V since high frequency AT cut crystals are designed for relatively high drive levels (5-10 mw max).

A typical calculation for the approximate power dissipated in a crystal is:

$$P = 2R (\pi \times f \times C \times V_{cc})^2$$

Where. R = crystal resistance of 40 ohms, C = C1 + Co = 20 pf. The calculation gives a power dissipation of 2 mW at 16 MHz.

Series Resistance. Lower series resistance gives better performance but costs more. Higher R results in more power dissipation and longer startup, but can be compensated by reduced C1 and C2. This value ranges from 200 ohms at 1 MHz down to 15 ohms at 20 MHz.

Frequency. The frequency of oscillation in parallel resonant circuits is mostly determined by the crystal (99.5%). The external components have a negligible effect (0.5%) on frequency. The external components (C1,C2) and layout are chosen primarily for good startup and reliability reasons.

Frequency Tolerance (initial temperature and aging). Initial tolerance is typically ±.01%. Temperature tolerance is typically ±.005% over the temp range (-30 to +100 degrees C). Aging tolerance is also given, typically ±.005%.

Holder. Typical holder part numbers are HC6, 18, 25, 33, 44.

Shunt Capacitance. (Cs) typically <7 pf.

Mode. Typically the mode (fundamental, 3rd or 5th overtone) is specified as well as the loading configuration (series vs. parallel).

The ceramic resonator equivalent circuit is the same as shown in Figure 4. The values differ from those specified in the theory section. Note that the ratio of L/C is much lower than with quartz crystals. This gives a lower Q which allows a faster startup and looser frequency tolerance (typically ±0.9% over time and temperature) than quartz.

Layout

The following text explains trace layout as it affects the various stray capacitance parameters (Figure 9).

Traces and Placement. Traces connecting crystal,caps, and the IC oscillator pins should be as short and wide as possible (this helps reduce parasitic inductance and resistance). Therefore, the components (caps and crystal) should be placed as close to the oscillator pins of the IC as possible.

Grounding/Guarding. The traces from the oscillator pins of the IC should be guarded from all other traces (clock, V_{cc}, address/data lines) to reduce crosstalk. This is usually accomplished by keeping other traces away from the oscillator circuit and by placing a ground ring around the traces/components (Figure 9).

Measurement and Observation

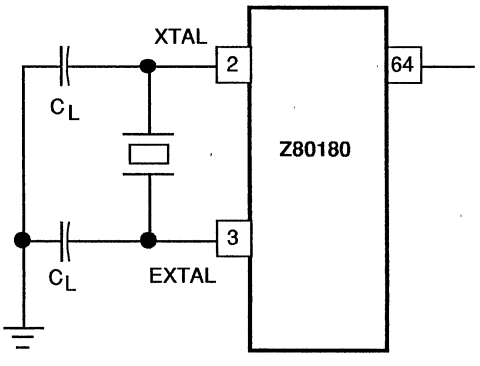
Connection of a scope to either of the circuit nodes is likely to affect operation because the scope adds 3-30 pf of capacitance and 1M-10M ohms of resistance to the circuit.

Indications of an Unreliable Design

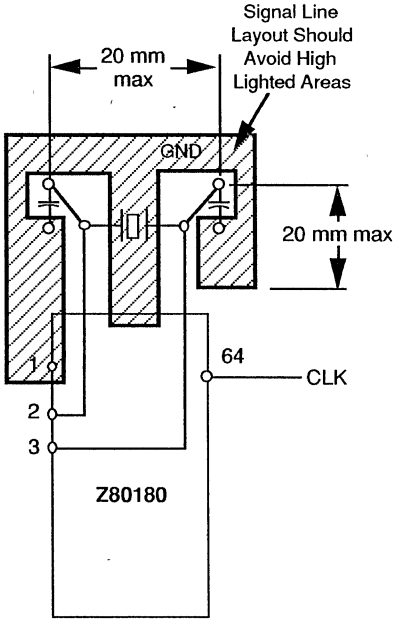
There are two major indicators which are used in working designs to determine their reliability over full lot and temperature variations. They are:

Start Up Time. If start up time is excessive, or varies widely from unit to unit, there is probably a gain problem. C1/C2 needs to be reduced; the amplifier gain is not adequate at frequency, or crystal Rs is too large.

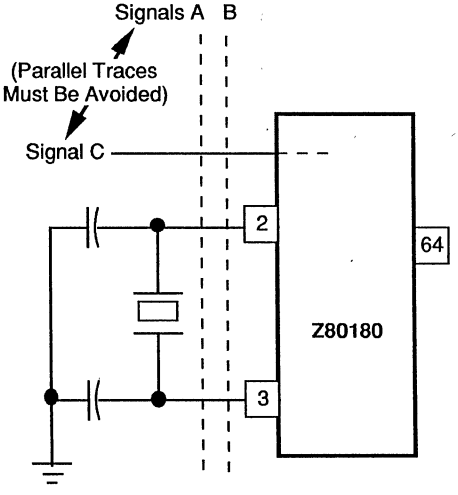
Output Level. The signal at the amplifier output should swing from ground to V_{cc}. This indicates there is adequate gain in the amplifier. As the oscillator starts up, the signal amplitude grows until clipping occurs, at which point, the loop gain is effectively reduced to unity and constant oscillation is achieved. A signal of less than 2.5 Vp-p is an indication that low gain may be a problem. Either C1/C2 should be made smaller or a low R crystal should be used.



Clock Generator Circuit



Board Design Example
(Top View)



- To prevent induced noice, the crystal and load capacitors should be physically located as close to the LSI as possible.
- Signal lines should not run parallel to the clock oscillator inputs. In particullar, the clock input circuitry and the system clock output (pin 64) should be separated as much as possible.
- V_{CC} power lines should be separated from the clock oscillator input circuitry.
- Resistivity between XTAL or EXTAL and the other pin should be greater than 10 MΩ

Figure 9. Circuit Board Design Rules

SUMMARY

Understanding the Theory of Operation of oscillators, combined with practical applications, should give designers enough information to design reliable oscillator circuits. Proper selection of crystals and load capacitors, along with good layout practices, results in a cost effective, trouble free design.Reference the following text for Zilog products with on-chip oscillators and their general/specific requirements.

ZILOG PRODUCT USING ON-CHIP OSCILLATORS

Zilog products that have on-chip oscillators:

Z8® Family: All
Z80®: C01, C11, C13, C15, C50, C90, 180, 181, 280
Z8000®: 8581
Communications Products: SCC™, ISCC™, ESCC™

ZILOG CHIP PARAMETERS

The following are some recommendations on values/parameters of components for use with Zilog on-chip oscillators. These are only recommendations; no guarantees are made by performance of components outside of Zilog ICs. Finally, the values/parameters chosen depend on the application. This App Note is meant as a guideline to making these decisions. Selection of optimal components is always a function of desired cost/performance tradeoffs.

Note: All load capacitance specs include stray capacitance.

Z8 Family

General Requirements:

Crystal Cut: AT cut, parallel resonant, fundamental mode.
Crystal Co: < 7 pf for all frequencies.
Crystal Rs: < 100 ohms for all frequencies.
Load Capacitance: 10 to 22 pf, 15 pf typical.

Specific Requirements:

8604: xtal or ceramic, f = 1 - 8 MHz.
8600/10: f = 8 MHz.
8601/03/11/13: f = 12.5 MHz.
8602: xtal or ceramic, f = 4 MHz.
8680/81/82/84/91: f = 8, 12, 16, MHz.
8671: f = 8 MHz.
8612: f = 12, 16 MHz.
86C08/E08: f = 8, 12 MHz.
86C09/19: xtal/resonator, f = 8 MHz, C = 47 pf max.
86C00/10/20/30: f = 8, 12, 16 MHz.
86C11/21/91/40/90: f = 12, 16, 20 MHz.
86C27/97: f = 4, 8 MHz.
86C12: f = 12, 16 MHz.
Super8 (all): f = 1 - 20 MHz.

Z8000 Family (8581 only)

General Requirements:

Crystal cut: AT cut, parallel resonant, fundamental mode.
Crystal Co: < 7 pf for all frequencies.
Crystal Rs: < 150 ohms for all frequencies.
Load capacitance: 10 to 33 pf.

Z80 Family

General Requirements:

Crystal cut: AT cut, parallel resonant, fundamental mode.
Crystal Co: < 7 pf for all frequencies.
Crystal Rs: < 60 ohms for all frequencies.
Load capacitance: 10 to 22 pf.

Specific Requirements:

84C01: C1 = 22 pf, C2 = 33 pf (typ); f = DC to 10 MHz.
84C90: DC to 8 MHz.
84C50: same as 84C01.
84C11/13/15: C1 = C2 = 20 -33 pf; f = 6 -10 MHz
80180: f = 12, 16, 20 MHz (Fxtal = 2 x sys. clock).
80280: f = 20 MHz (Fxtal = 2 x Fsysclk).
80181: TBD.

ZILOG CHIP PARAMETERS (Continued)

Communications Family

General Requirements:

Crystal cut: AT cut, parallel resonant, fundamental mode.
Crystal Co: < 7 pf for all frequencies.
Crystal Rs: < 150 ohms for all frequencies.
Load capacitance: 20 to 33 pf.
Frequency: cannot exceed PCLK.

Specific Requirements:

8530/85C30/SCC: f = 1 - 6 MHz (10 MHz SCC), 1 - 8.5 MHz (8 MHz SCC).
85130/ESCC (16/20 MHz), f = 1 - 16.384 MHz.
16C35/ISCC: f = 1 -10 MHz.

REFERENCES MATERIALS AND ACKNOWLEDGEMENTS

Intel Corp., Application Note AP-155, "Oscillators for Micro Controllers", order #230659-001, by Tom Williamson, Dec. 1986.

Motorola 68HC11 Reference Manual.

National Semiconductor Corp., App Notes 326 and 400.

Zilog, Inc., Steve German; Figures 4 and 8.

Zilog, Inc., Application Note, "Design Considerations Using Quartz Crystals with Zilog Components" - Oct. 1988.

Data Sheets; CTS Corp. Knights Div., Crystal Oscillators.

Z8 Low Cost Thermal Printer

Using the Z8 microcontroller to control thermal printers offers flexibility and performance at low cost

Introduction

Compact and low cost thermal printers are popular for many applications: point of sale equipment; medical and industrial instrumentation; micro-fax and micro-printers for the consumer markets, and other applications. Each application is unique with different operating environments, operator skill levels and performance requirements. The Z8 microcontroller (Z86E21) is used as a common controller in this application example. It provides wide flexibility and performance potential in both hardware and software. This Application Note provides design engineers with a window to view how the Z8 interfaces with thermal printers, providing a platform from which they can customize their designs.

Mechanical Considerations

Thermal printers have several common, yet important, attributes. Most important, is a thermal print head which has a number of resistive heating elements. When heated, these elements produce an image on heat sensitive paper. Secondly, thermal printers must advance the print head across the paper media. It must make provisions to advance the paper after the head has traversed the width of the paper. Depending upon the cost and complexity of the thermal printer used, the mechanism for paper advance can be an independent motor and gear drive assembly, or simply a ratchet driven line feed for every print head carriage return. One disadvantage of the latter is that it will not print bidirectionally.

Additionally, there are mechanical features which enable interconnect to the controller circuit. Also, provisions for securing the assembly to a case and paper cutter. The printer must supply sufficient pressure on the paper, sandwiched between the platen and roller. Then, the paper must be torn across a fixed, serrated cutter bar. Figure 1 shows a typical thermal printer. Note the threaded, head feed screw attached via a gear system to stepper motor #1. Motor #2 advances the paper as required by rotating the platen through a gear drive.

Electrical Considerations

The key elements to controlling thermal printers are in properly synchronizing the movement of the print head and paper to the required application. Also, as in any closed loop control system, the printer must deliver a status signal back to the controller. Since paper movement is continuous, it has no electronic analog of "begin and end", except for "paper out" or "paper jammed" conditions. To close the loop, thermal printers define a position for the print head as "home", and give an active low feedback status when "home." Normally, this is at the far left of the carriage span.

Four-phase stepper motors are commonly used to transport the print head. If the printer is to print bidirectionally, a second stepper motor is used for paper advance. The equivalent circuit for the four-phase stepper motor, and the required support components are shown in Figure 2. The four diodes are used to discharge the inductive kick which is generated when the transistors are switched off. The Zener diode provides threshold for this action. Table 1 shows typical phase drive sequences for clockwise and counter clockwise rotational motion of a four-phase stepper motor (note equivalent Hex code values).

Table 1. Typical Phase Drive Sequences

Clockwise			Counter Clockwise		
Step	Phases	Hex Code	Step	Phases	Hex Code
1	1101	C	1	0011	3
2	0110	6	2	0110	6
3	0011	3	3	1100	C
4	1001	9	4	1001	9

Thermal print heads are arranged in a typical column of 7, 8, or 9 elements. Each element is called a dot. Figure 3 shows the arrangement of an eight dot head.

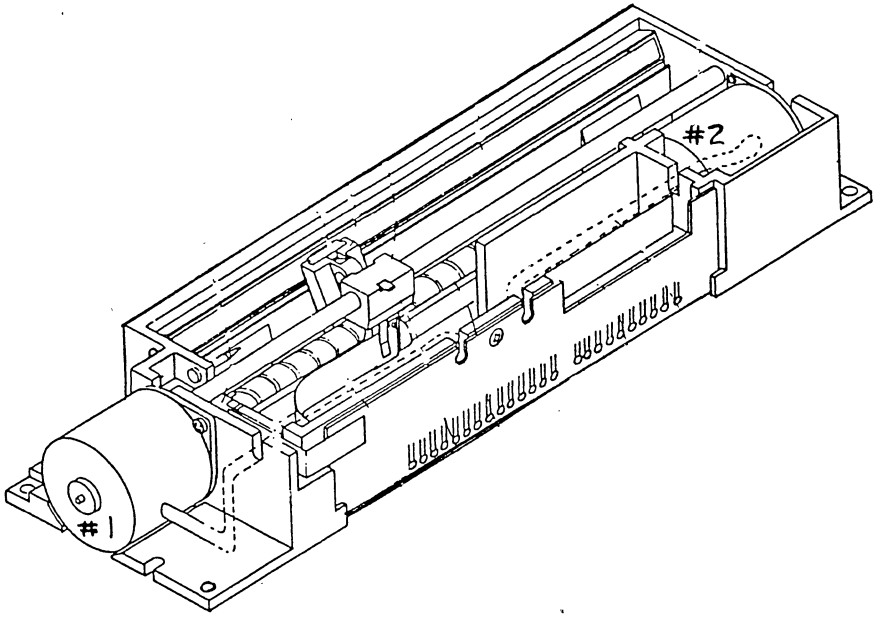


Figure 1. Thermal Printer

Since the print head traverses the width of the paper on the head feed screw, the electronic signals required to operate the head are delivered via a flexible printed circuit (typically copper traces sandwiched between mylar). Each dot has a closely controlled value of resistance. This resistance ranges between 14 and 18 ohms in compact thermal printers. Heat is generated by a given dot through I^2R dissipation.

A dot has a maximum applied energy specification and a recommended or rated specification. Printer manufacturers describe this energy in Joules, or more closely, milli-joules (mj). One joule per second is one watt. Energy is usually expressed with a time base. Understanding this concept is essential in thermal printer design. The print circuit must not exceed the energy rating of a dot. Simultaneously, it must control the duration (dissipation) to ensure proper print quality and protection of the print head elements (dots).

For example, a dot has a resistance of 16 ohms, a rated energy spec of 2.10 mj, and a maximum energy spec of 2.52 mj, at 5 Volts. The pulse width (time) required to apply the rated energy in milliseconds is given by the equation in Figure 4.

$$t = R * E / V2$$

Where: R = Head Resistance (Ohms)
E = Applied Energy (mj)
V = Voltage (Volts)
t = Pulse Width (ms)

Figure 4. Thermal Energy Equation

The Thermal Energy Equation yields a required pulse width of 1.34 ms to apply the rated energy of the head. The value of E, the applied energy in the equation, needs to reflect the effects that ambient temperature fluctuations induce. The equation to compensate the applied energy used in deriving pulse width is given by Figure 5.

$$E = E_0 (1 + T_0 - T/100)$$

Where: E = Applied Energy (mj)
E_0 = Rated Energy (mj)
T_0 = 24 °C
T = Ambient Temperature (°C)

Figure 5. Ambient Temp Compensation Equation

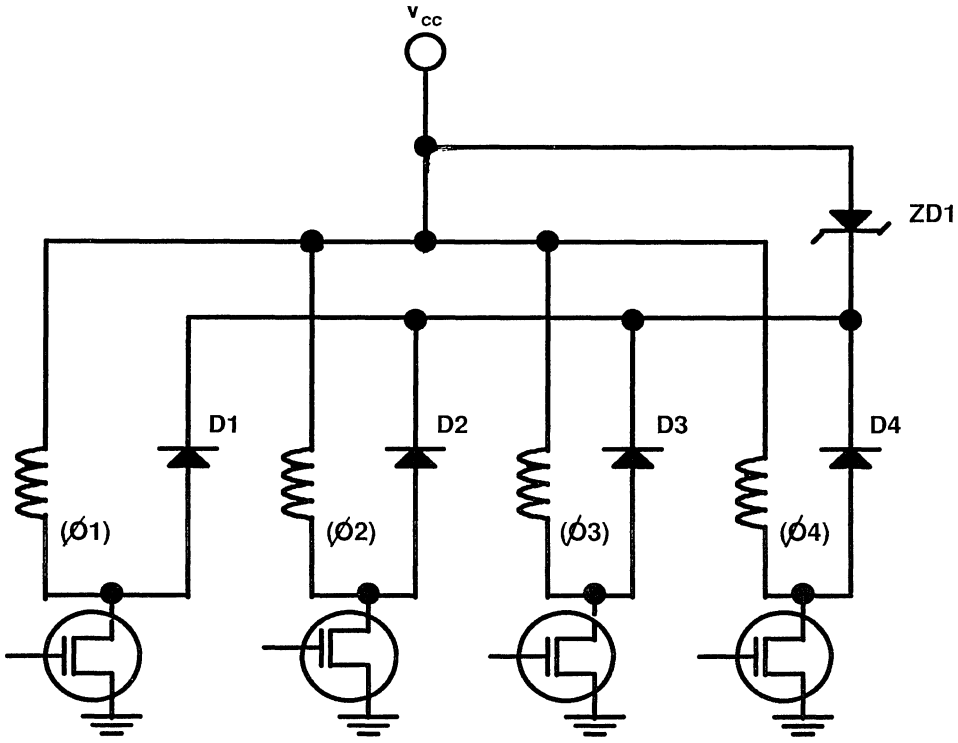


Figure 2. Equivalent Circuit for Four Phase Stepper Motor

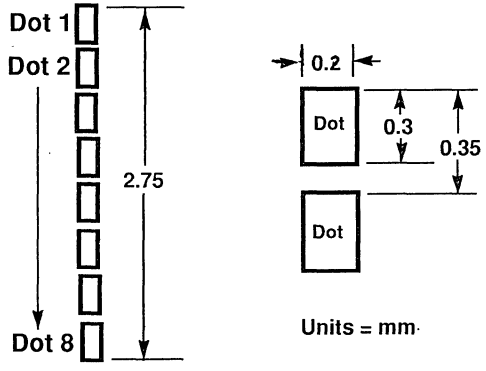


Figure 3. Head Arrangement - Eight Dot

Clearly from Figure 5, as ambient temperature decreases, supplied energy must be increased. Thus, the pulse width is increased to maintain print quality

Printing a Character

Again, the key element to controlling a thermal printer is synchronizing the head movement to the print application. Figure 6 is a timing chart that shows the combination of the head feed motor steps with the the applied energy pulses in printing the character two (2).

Using Figure 6 and starting from the left (Step 1), no print elements are activated. In Step 2, the print head advances from its previous position and dots two and eight are activated. This "step then print...step then print...etc." process is repeated until the character is

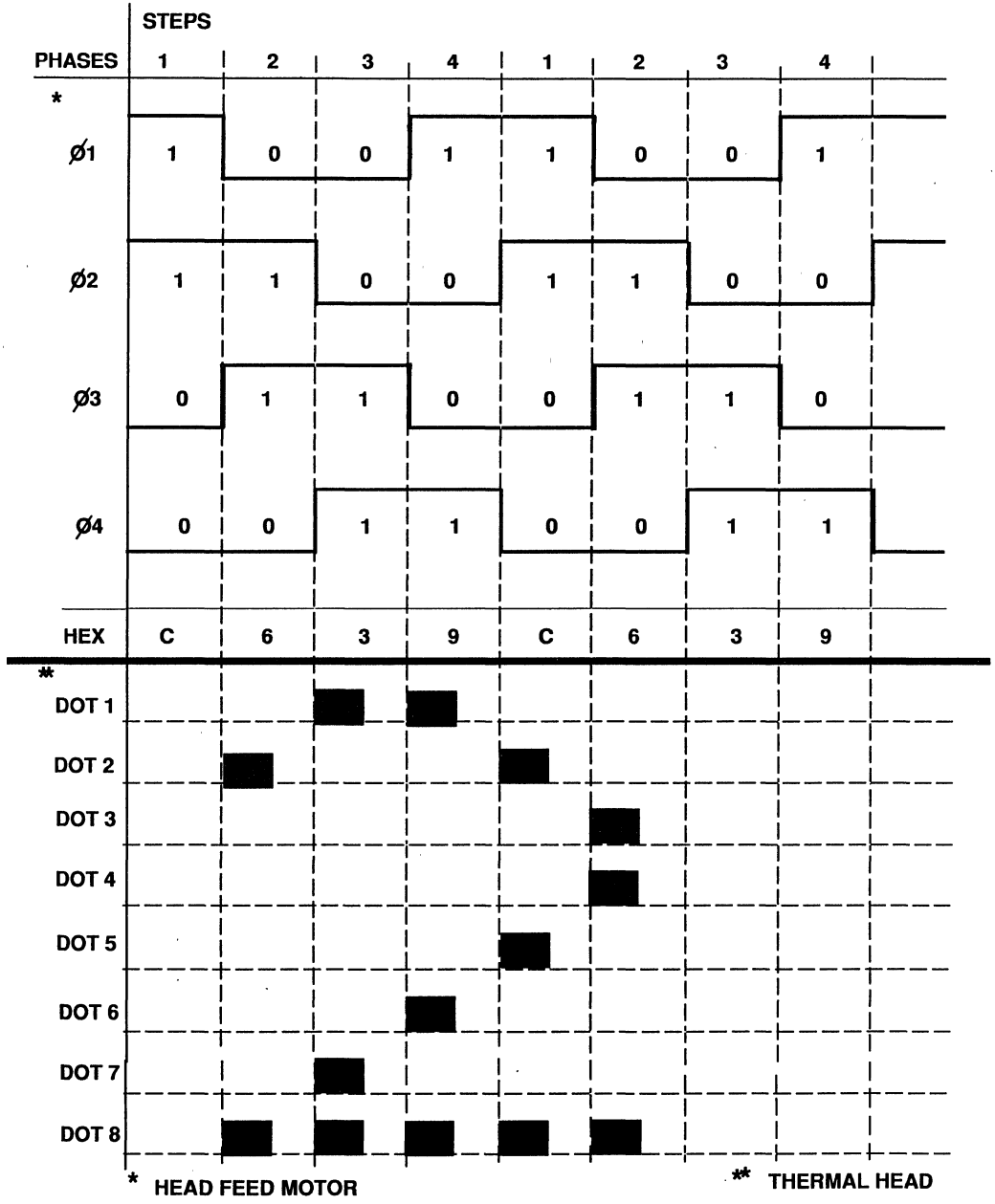


Figure 6. Printing the Character "2"

complete. If each step took 4 ms, this character would be complete in five steps, or 20 ms.

Z8 Controller Application

Using the Z8 microcontroller (Z86E21) for control of low cost thermal printers offers the user good flexibility and performance potential. The Schematic for an 8-dot, dual stepper motor control interface to an Z86E21 is shown in Figure 7.

Ports 0 and 1 are used on the Z86E21. Low cost 18-pin, bipolar, octal peripheral array drivers (ULN2801A) are used to interface TTL logic signals from the Z86E21 to the printer. The outputs from the ULN2801As are capable of sinking the 500 mA currents that the stepper motor windings source. Additionally, an active dot can typically source 300 to 400 mA of current. The equivalent circuit of the ULN2801A is shown in Figure 8. Note the inclusion of discharge diodes and the use of Zener diodes discussed in the earlier section (Figure 7).

When the print head reaches the far left position of the carriage travel, it is in the home position. The home pin is actively low when this status is present (Figure 7). The 0.1 uf capacitor (C1) debounces the event. The 2.2K resistor provides an RC time constant and current limit. The 1K resistor is a pull-up for the Z86E21 input. In this example, a home status produces a falling edge interrupt. The 5K port pull-ups provide sufficient bias current to the bipolar inverters of the ULN2801As.

Print Head Protection

Depending upon the application environment and operating conditions, the designer must consider to what degree he provides print head protection. Bad resets, power fails, ESD jolts, and printing errors can produce potential conditions whereby the head is left on too long. Depending on the maximum energy rating, the heads can be permanently damaged after only seconds of carrying current continuously. The designer should consider these different potentially dangerous scenarios and make arrangements to avoid/manage them. One helpful design is to install a MOSFET in the ULN2801A to VCC path. Then, a Watchdog chip can be used to switch off the MOSFET in the event that the Z86E21 has lost its way due to upset. This then removes power from the print circuit.

Initializing and Print Coding Flowcharts

The coding necessary to make this interface operate is shown in the Flowchart of Figure 9A. First, the Z86E21 is initialized and ports 0 and 1 are set to zero. Counter/Timer one of the Z8 is used to provide a main task interrupt clock of 100 us. This task clock yields the millisecond order signals for incrementing stepper motors and activating thermal printer heads with the proper accuracy. Continuing the flow, the print head carriage is "homed." To do this, the home status is first polled for a low. Then, actions are taken to move the head in order to generate a falling edge interrupt for IRQ2, Pin 32.

Once home status is secure, Figure 9B shows typical print coding flow. A character is available once it has reached an internal FIFO stack in the Z8 register file. This FIFO stack is maintained by the interrupts which handle the received characters on the serial or parallel interface. The ASCII character is then converted to an index address in the ROM where the bit map for that particular character is located. This bit map is then printed one column at a time, sequentially incrementing the head position one step.

Z86E21 Flexibility

The Z86E21 further complements this application when considering the host interface possibilities available to the designer. The Z8 has UART on board to easily implement an RS232 style serial interface to a host. Note that the Z8 is only required to receive data, thus RxD is the only active UART pin. Flow control comes from P31, where an active high condition flags printer not ready (Figure 10).

Another possible parallel interface is the Centronics. With 32 potential I/O signals on the Z8, 17 of which are used for the printer interface, only a subset of the Centronics interface can be adapted directly to the Z8. The designer could adapt full interfaces if required by the application, through addition al logic (Table 2).

The Z8 has two-wire handshake logic built-in, greatly simplifying data flow. The stars in Table 2 show the boundary between a simple parallel interface, and a complex one.

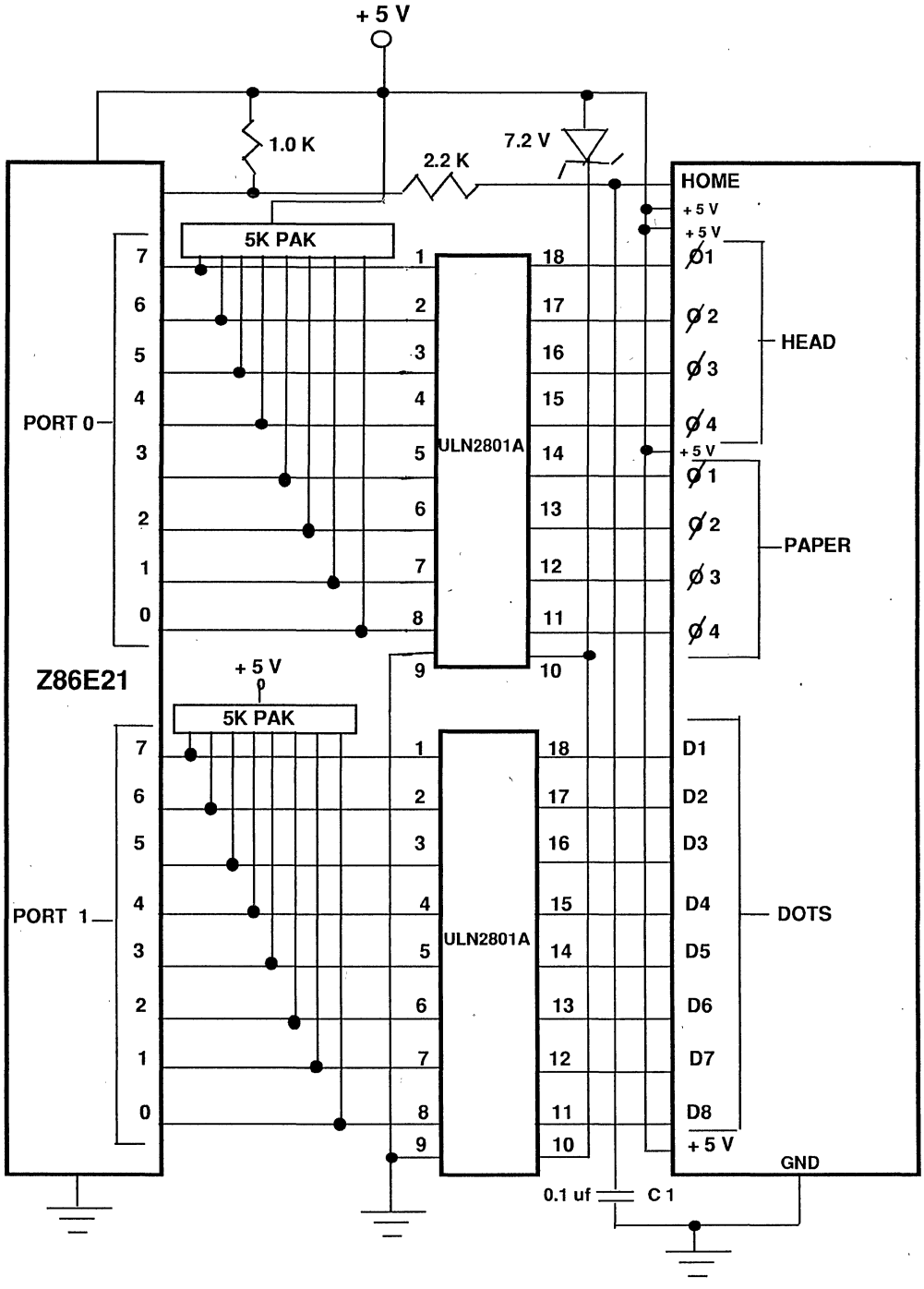


Figure 7. Z86E21 to 8-Dot Dual Stepper Motor Control Interface

PIN CONNECTIONS

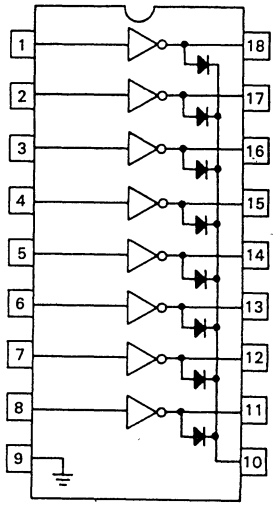


Figure 8. ULN2801A Equivalent Circuit

Table 2. The Parallel Interface

#	Name	Transmitter	Function
1	/STB	Host	Active Low (/) indicates valid data
2	D0	Host	Parallel data byte
3	D1	Host	
4	D2	Host	
5	D3	Host	
6	D4	Host	
7	D5	Host	
8	D6	Host	
9	D7	Host	
10	/ACK	Printer	Active Low (/) indicates data received
11	BSY	Printer	off line, error, or handling data
12	PE	Printer	Printer out of paper
13	SLCT	Printer	Active high, printer is on, and selected
14	/ALF	Host	Auto Line Feed
15	/INT	Host	Reset
16	/ERR	Printer	General Machine Error
17	/SLN	Host	Select-in, from from host

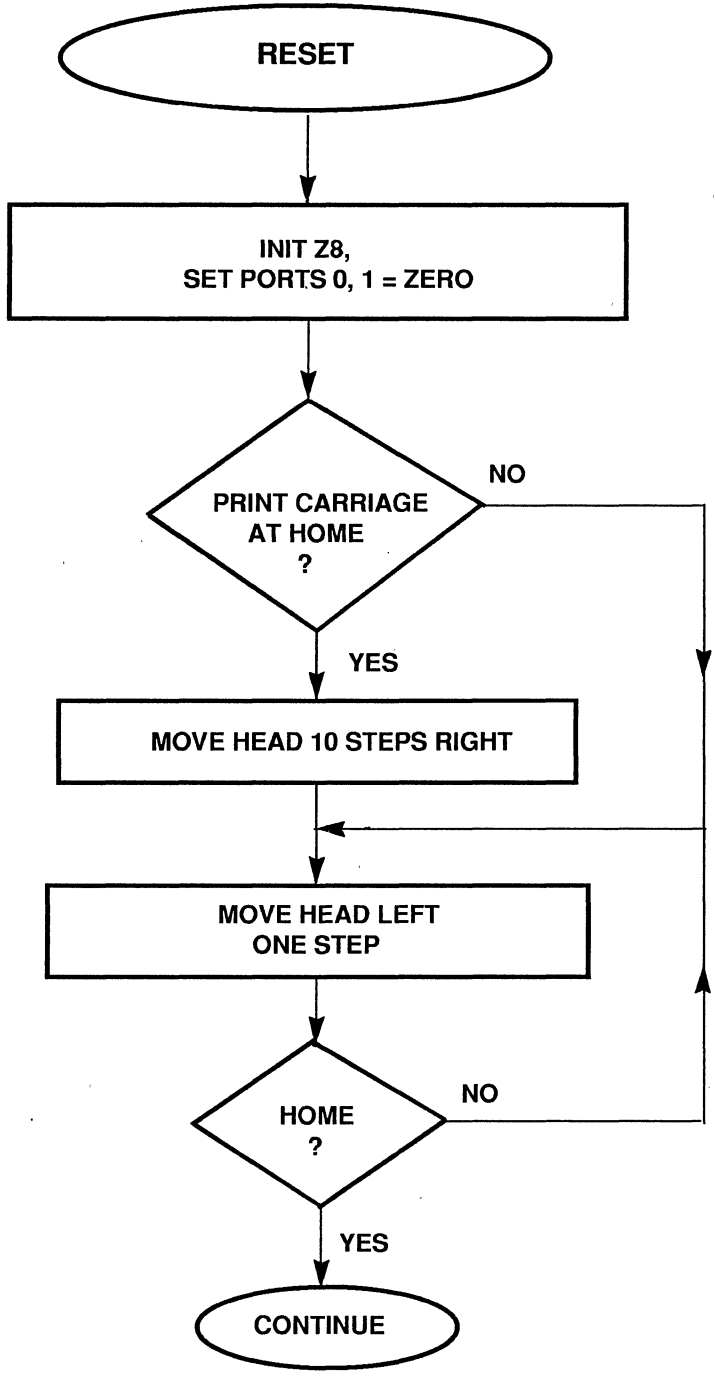


Figure 9A. Initialized Coding Flowchart

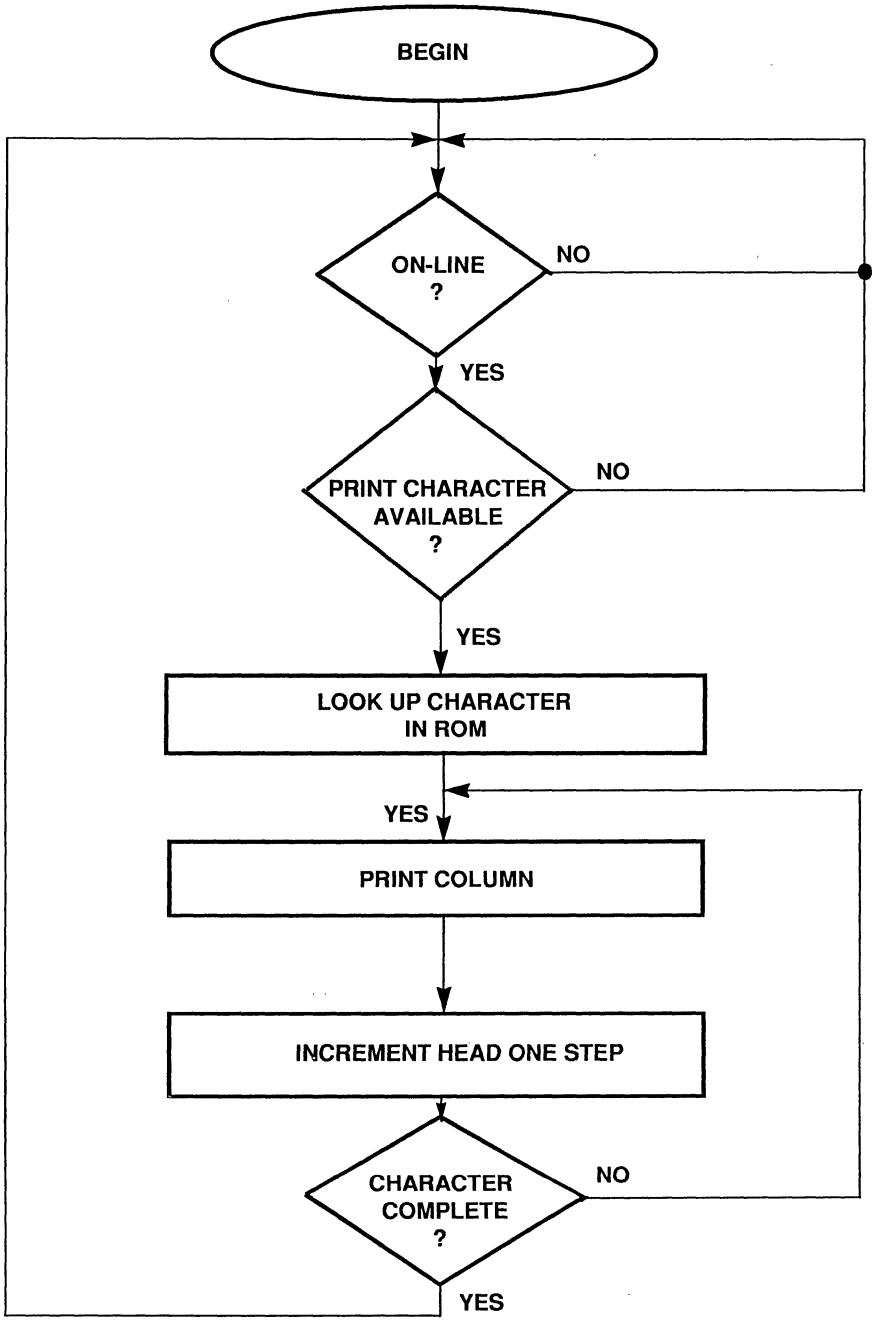


Figure 9B. Print Coding Flowchart

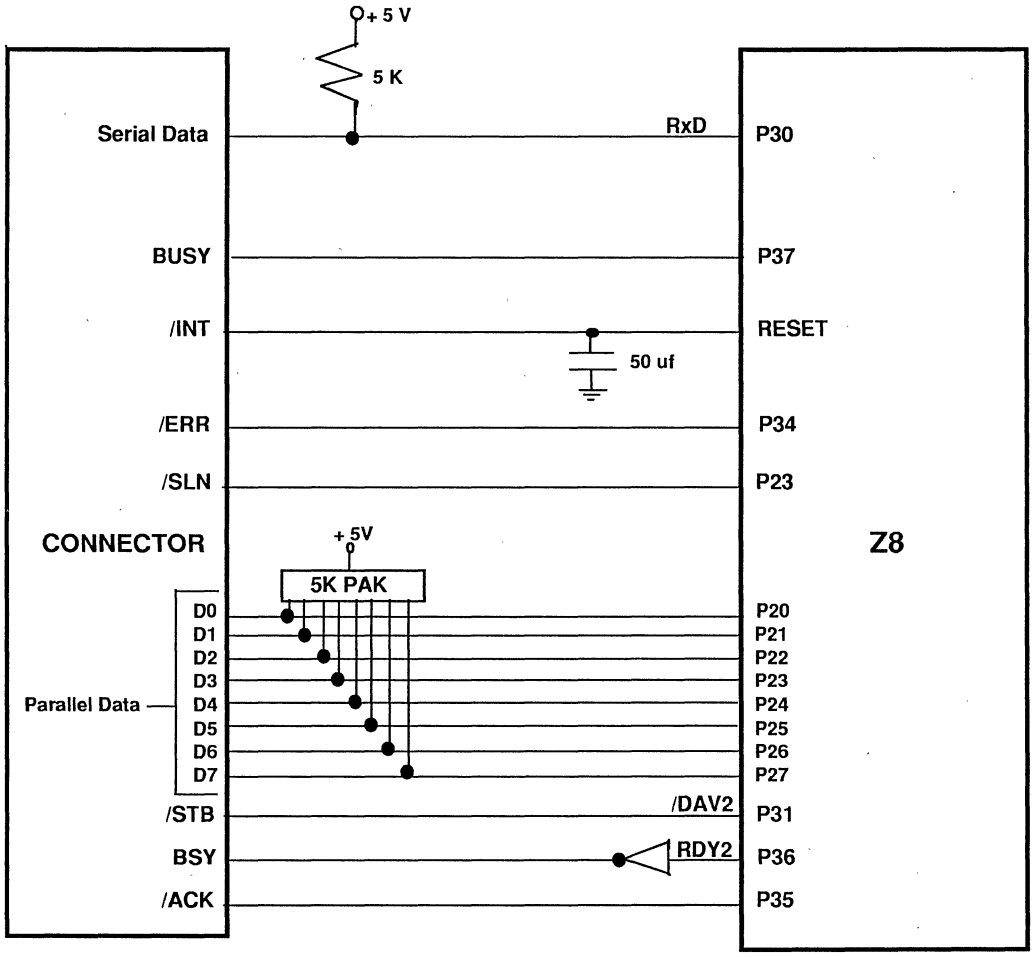
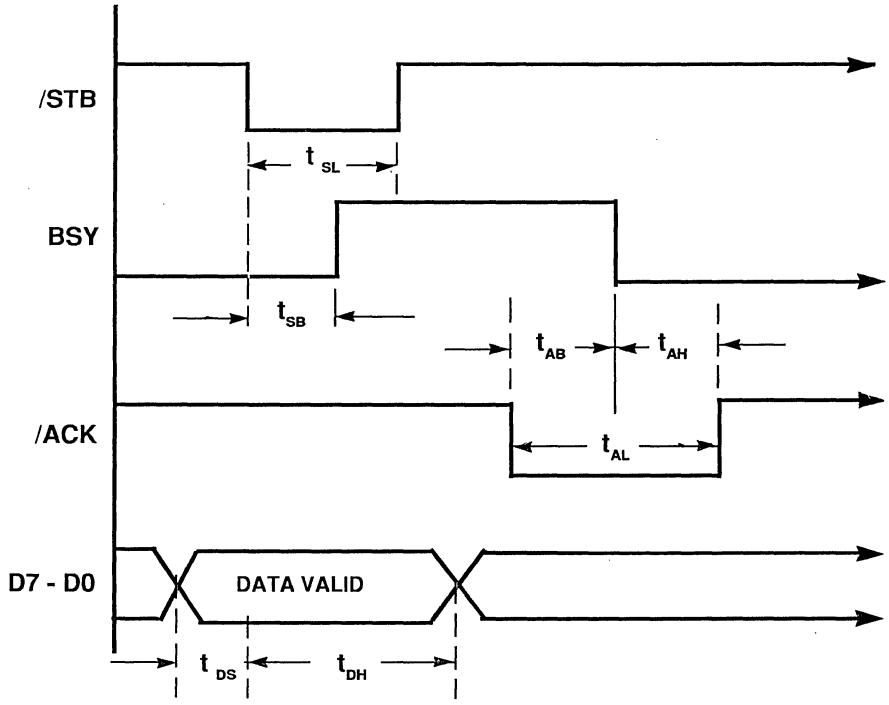


Figure 10. Z8 Implementation of a RS232 Interface.

Figure 10 shows the parallel interface. Data lines are pulled up to improve drive characteristics on the bus. The /STB signal easily maps into the Z8 /DAV2 function, where a falling edge generates a service interrupt for the current valid data byte. The BSY strobe output from the Z8 closely matches the RDY2 function. Refer to the Z8 input handshake functions for further details (Z8 Family Design Handbook - 03- 8275). The /ACK strobe can be implemented via software in the character interrupt subroutine. Figure 11 shows the ideal handshake timing that the software implements.

Conclusion

Hopefully, this Application Note has been helpful in the area of thermal printer design. It condenses a lot of information into a fast format package. It is recommended that if you pursue a Z8 design with a thermal printer, use this document to complement the OEM printer spec. If there are questions, please refer to the back of this document for your local Zilog Sales office. A resident Field Application Engineer is available to further discuss your application.



t_{SL}	= 0.5 us	min	t_{DH}	= 0.5 us	min	t_{AH}	= 6 us	typ
t_{SB}	= 2.8 us	typ	t_{AL}	= 10 us	typ			
t_{DS}	= 0 us	min	t_{AB}	= 4 us	typ			

Figure 11. Parallel Handshake Timing

EXPAND THE I/O PORTS IN YOUR Z8 APPLICATIONS

Add Extra IO Ports to Your Z8 Applications – Cost Effect - ively

INTRODUCTION

Certain Z8 applications require extra I/O ports. Some engineers use the 74LS374 for expanded output ports and the 74LS244 for expanded input ports. The advantage of using these TTLs is low cost. The disadvantage is inflexibility (for example: you have to configure the port direction by hardware). This Application Note explains the use of the Z80 PIO to expand the I/O ports in your Z8 applications.

HARDWARE

The Z8691 circuit diagram interface with the Z80 PIO is shown in Figure 2. In this application, the Z80 PIO interrupts and handshakes are not used. Also, because the Z8 does not provide M1 signals to the PIO, PIO pin 37 is pulled high. The /RD (Read Cycle Status) and / IORQ (Input/Output Request) signals are composed of /CE and R//W by using U5 (74LS04) and U6 (74LS32).

The address of the PIO in this application is %1000 - %1FFF (HEX). Ports A and B of the PIO are selected by A0 (Address line) = "0" and A0 = 1, respectively. The Data Mode or Control Mode of the PIO can be selected by A1 = "0" or A1 = "1", respectively. Table 1 shows the relationship between A0, A1, A//B and C//D.

Table 1. Address/Port Relationships

A1	A0	STATUS
0	0	Port A Data Mode
1	0	Port A Control Mode
0	1	Port B Data Mode
1	0	Port B Control Mode

The clock source is taken from the Z8's 8MHz crystal out signal. Pin 6 of U6 provides the /IORQ signal which is about 50ns behind /CE. Pin 11 of U6 provides /RD which is about 61ns behind /CE (Figure 1).

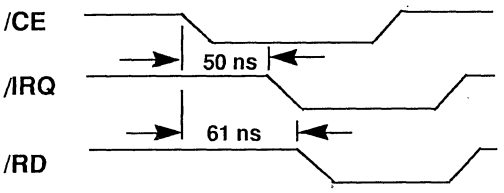


Figure 1. Interrupt/Read Timing

SOFTWARE

The Z80 PIO ports can be programmed to operate in four modes: Output (Mode 0), Input (Mode 1), Bidirectional (Mode 2) and Bit Control (Mode 3). These four modes have different program sequencing (Page 85 of the Z80 Family Data Book January, 1989 describe the details). This document uses Mode 3 as an example. Program 1 (Figure 3) shows the initialization of Mode 3.

DEVELOPMENT TOOL APPLICATIONS

The following text and illustrations show hardware/ software parameter applications as a development tool for expandable ports.

Hardware Applications

Since the ICE (In Circuit Emulation) chip of the CCP family is not available, it is very difficult to emulate the Z86C30/40 by using the Z86C90. Because the Z86C90 is configured with Port 1 as an Address/Data Bus and Port 0 as an Address Bus, the chip still needs two ports (Port 0 and Port 1) for Z86C40 emulation. Figure 6 illustrates the PIO application for the Z86C90 to emulate the missing ports (Port A = Port 0 and Port B = Port 1).

Four methods are proposed to emulate the Z86C30/ C40 by using this PIO application:

1. EPROM debugging method

   Build a PIO board like Figure 6. Burn the application program into an EPROM. Plug in the U3 socket of the PIO board and test the application program.

2. Z86C1900ZEM method

   Build a PIO board like Figure 7. Disconnect the

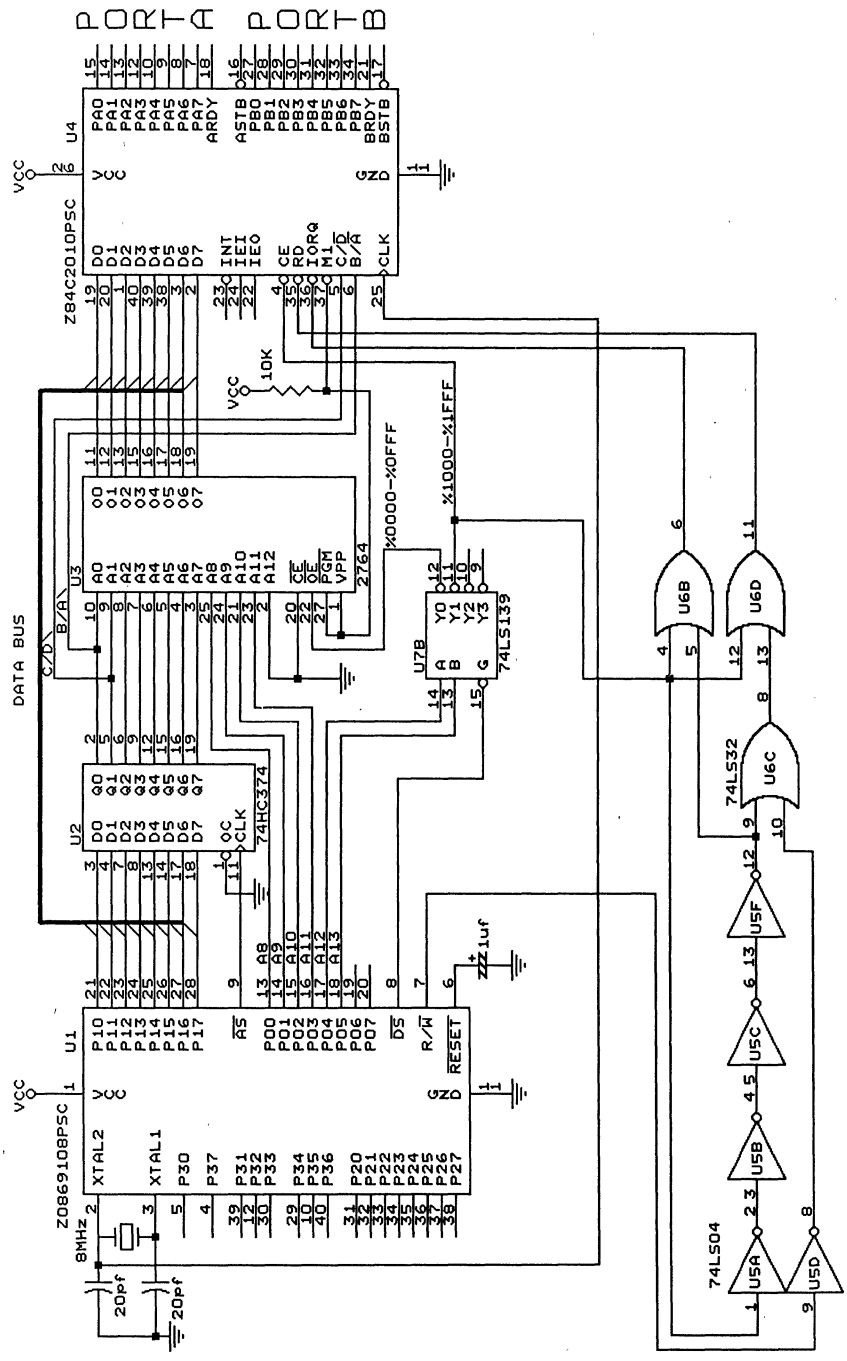


Figure 2. Z8 Interface to the CMOS Z80 PIO

```
MODE3:          EQU   11001111B     ;define Mode 3
PA_CTRL:        EQU   %1002         ;address of Port A control
PA_DATA:        EQU   %1000         ;address of Port A Data
PB_CTRL:        EQU   %1003         ;address of Port B Control
PB_DATA:        EQU   %1001         ;address of Port B Data
REGA_CTRL:  EQU   ????????B         ;?=0/1, 0=OUTPUT; 1=INPUT
REGB_CTRL:  EQU   ????????B         ;?=0/1, 0=OUTPUT; 1=INPUT
INT_VECT:       EQU   XXXXXXXXB     ;int vector; X=don't care
INT_CTRL:       EQU   01110111B     ;int control;
M_CTRL:         EQU   11111111B     ;no interrupt
INT_DIS:        EQU   XXXX0011B     ;int disable; X=don't care
;
pio_init: ld    r4,#>PA_CTRL ;config Port A
          ld    r5,#<PA_CTRL
          ld    r6,#MODE3           ;step 1: load Mode 3
          ldc   @rr4,r6
          ld    r6,#REGA_CTRL       ;step 2: load Reg Ctrl Word
          ldc   @rr4,r6
          ld    r6,#INT_VECT        ;step 3: load Interrupt Vector
          ldc   @rr4,r6
          ld    r6,#INT_CTRL        ;step 4:load Interrupt Ctrl
          ld    @rr4,r6
          ld    r6,#M_CTRL          ;step 5: load Mask Contrl
          ld    @rr4,r6
          ld    r6,#INT_DIS         ;step 6: load Int Disable
          ld    @rr4,r6

          ld    r4,#>PB_CTRL ;config Port B
          ld    r5,#<PB_CTRL
          ld    r6,#MODE3           ;step 1: load Mode 3
          ldc   @rr4,r6
          ld    r6,#REGA_CTRL       ;step 2: load Reg Ctrl Word
          ldc   @rr4,r6
          ld    r6,#INT_VECT        ;step 3: load Interrupt Vector
          ldc   @rr4,r6
          ld    r6,#INT_CTRL        ;step 4: load Interrupt Ctrl
          ld    @rr4,r6
          ld    r6,#M_CTRL          ;step 5: load Mask Contrl
          ld    @rr4,r6
          ld    r6,#INT_DIS         ;step 6: load Int Disable
          ld    @rr4,r6
```

Figure 3. Initialization of PIO

2. Z86C1900ZEM method

Build a PIO board like Figure 7. Disconnect the /CSRAM signal (Pin3 of U6 to Pin 20 of U3) of the Z86C1900ZEM board (contact Zilog Sales Offices for details). Make the following signal connections:

PIO Board	to	Z86C1900ZEM Board
AD0 - AD7		AD0 - AD7
A0		A0
A1		A1
A14		A14
A15		A15
/DS		/DS
XTAL2		XTAL2
R//W		R//W

Pin 10 of U3B (PIO board) connects to Pin 20 (/CE) of U3 (Z86C1900ZEM board). After this modification, the address map is changed as follows:

%0000 - %7FFF	32K Byte Z86C1900ZEM Monitor EPROM
%8000 - %BFFF	16K Byte User RAM
%C000	PIO Port A Data Mode
%C002	PIO Port A Control Mode
%C001	PIO Port B Data Mode
%C003	PIO Port B Control Mode

The user program can now be downloaded into the user RAM for emulation.

3. ROM Emulator Method

Build a PIO board like Figure 6. Connect Port A, Port B, Port 2, Port 3 and all control signals of the PIO board to the target board (or group all these I/O lines and control signals into a 40 line emulation cable). Plug the emulator cable from the ROM emulator to U3 socket of the PIO board. Debug your application program by using the ROM emulator.

4. Z86C90 In Circuit Emulator Method

Build a PIO board like Figure 8. Connect Port A, Port B, Port 2, Port 3 and all control signals of the PIO board to the target board. Plug the 40-pin emulator cable from the emulator to the Z86C90 socket (U1) of the PIO board. Now, the application program can be downloaded into the RAM (6264) of the PIO board for debug.

SOFTWARE APPLICATION

Actually, the user has to change his debug program when using this application to emulate the Z86C40/30. However, the final program is similar to the debug program. First, the user has to put the PIO initialization routine after the Z8 initialization routine. The PIO initialization routine has to execute on each I/O Port direction change. Figure 4 shows the Flow-Chart of the initialization routine.

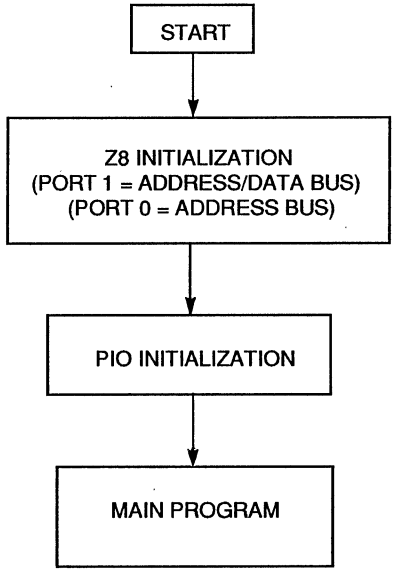


Figure 4. Initialization Routine

Secondly, the I/O port changes to memory accessable instead of register accessable. For example: with the real Z8 I/O port, use the instruction "ld 1,#data" to load the data to Port 1. But, for this application, use the instructions of "ld rn',#data" and "ldc @rrn,rn'" (where rrn will store the address of PIO Port 1).

In the software application, the simple way to access the PIO is using the "MACRO" in the program. Having finished the debugging program, change the definition of the "MACRO". Figure 5 illustrates the "MACRO" application program.

```
                                ;THIS APPLICATION USES REGISTER GROUP %00
;Define the registers
r4:     reg     h_pioa          ;r4 for PIO A high-byte address
r5:     reg     l_pioa          ;r5 for PIO A low-byte address
rr4:    reg     pioa            ;register pair
r6:     reg     pio_data        ;r6 for PIO R/W data
r8:     reg     h_piob          ;r8 for PIO B high-byte address
r9:     reg     l_piob          ;r9 for PIO B low-byte address
rr8:    reg     piob            ;register pair

;Define the PIO address
pa_data:equ%1000                ;address of port A data mode
pa_ctrl: equ    %1002           ;address of port A control mode
pb_data:equ     %1001           ;address of port B data mode
pb_ctrl: equ    %1003           ;address of port B control mode

;Define the Variables for the PIO init.
rega_ctrl: equ  ????????b       ;configure Port A. ? = 0 or 1
                                ;0 = output, 1 = input
regb_ctrl: equ  ????????b       ;configure Port B.
int_vect:  equ  %00             ;Interrupt Vector Word
int_ctrl:  equ  %77             ;Interrupt Control Word
m_ctrl:    equ  %ff             ;Mask Control Word
int_dis:   equ  %03             ;Interrupt Disable Word
mode3:  equ     %cf             ;Mode Control Word - Mode 3

;Define the MACRO for PIO accessing
init_add: MACRO                 ;This macro sets the PIO PA/PB address
        ld      h_pioa,#>pa_data    ;put PIO PA address to r4
        ld      l_pioa,#<pa_data
        ld      h_piob,#>pb_data    ;put PIO PB address to r5
        ldl_piob,#<pb_data
        MACEND

r0_rd:  MACRO                   ;working register mode = ld pio_data,r0
        ldc     pio_data,@pioa  ;load data from PA to r6
        MACEND
p0_rd:  MACRO                   ;register mode = ld pio_data,0
        push    rp              ;save the register pointer
        srp     #%0             ;point to group %00
        ldc     pio_data,@pioa  ;load data from PA to r6
        pop     rp              ;install the register pointer
        MACEND

r0_wt:  MACRO                   ;working register mode = ld r0,pio_data
        ldc     @pioa,pio_data  ;load data from r6 to PA
        MACEND

p0_wt:  MACRO                   ;register mode = ld 0,pio_data
        push    rp              ;save the register pointer
        srp     #%0             ;point to group %00
        ldc     @pioa,pio_data  ;load data from r6 to PA
        pop     rp              ;install the register pointer
        MACEND
```

Figure 5. Application of MACRO

```
r1_rd:  MACRO                          ;working register mode = ld pio_data,r1
        ldc     pio_data,@piob         ;load data from PB to r6
        MACEND

p1_rd:  MACRO                          ;register mode = ld pio_data,1
        push    rp                     ;save the register pointer
        srp     #%0                    ;point to group %00
        ldc     pio_data,@piob         ;load data from PB to r6
        pop     rp                     ;install the register pointer
        MACEND

r1_wt:  MACRO                          ;working register mode = ld r1,pio_data
        ldc     @piob,pio_data         ;load data from r6 to PB
        MACEND

p1_wt:  MACRO                          ;register mode = ld 1,pio_data
        push    rp                             ;save the register pointer
        srp     #%0                              ;point to group %00
        ldc     @piob,pio_data                ;load data from r6 to PB
        pop     rp                               ;install the register pointer
        MACEND

;Z8 Initialization Routine

        ORG     %0
        .word   irq0                   ;define the interrupt vectors
        .word   irq1
        .word   irq2
        .word   irq3
        .word   irq4
        .word   irq5
        ORG     %c
        di                             ;disable interrupts
        ld      p01m,#%96              ;init. P0 and P1
                :
                :
                :
;PIO Initialization Routine
        srp     #%70                   ;Put the PIO init Value into group %70 ;registers
        ld      r0,#>pa_ctrl           ;Set PIO high byte address to R0
        ld      r2,#mode3              ;Load Mode 3 value to R2
        ld      r3,#rega_ctrl          ;load PA I/O Register Control Word
        ld      r4,#regb_ctrl          ;load PB I/O Register Control Word
        ld      r5,#int_vect           ;load PIO Interrupt Vector Word
        ld      r6,#int_ctrl           ;load PIO Interrupt Control Word
        ld      r7,#m_ctrl             ;load PIO Mask Control Word
        ld      r8,#int_dis            ;load PIO Interrupt Disable Word
        call    pio_init               ;init the PIO
        jr      next_job               ;jump to main program

pio_init:
        ld      r1,#<pa_ctrl           ;init PA
        ldc     @rr0,r2                ;load Mode 3
        ldc     @rr0,r3                ;load PA Register Control Word
        ldc     @rr0,r5                ;load Interrupt Vector Word
```

Figure 5. Application of MACRO (continued)

```
        ldc     @rr0.r6         ;load Interrupt Control Word
        ldc     @rr0,r7         ;load Mask Control Word
        ldc     @rr0,r8         ;load Interrupt Disable Word

        ld      r1,#<pb_ctrl    ;init PB
        ldc     @rr0,r2         ;load Mode 3
        ldc     @rr0,r4         ;load PB Register Control Word
        ldc     @rr0,r5         ;load Interrupt Vector Word
        ldc     @rr0.r6         ;load Interrupt Control Word
        ldc     @rr0,r7         ;load Mask Control Word
        ldc     @rr0,r8         ;load Interrupt Disable Word
        ret

;Main Program
next_job:
        init_add                ;use the MACRO to init PIO's address
;
;an example of load the data to Port A
;
        ld      pio_data,#data  ;load the data to the buffer
        r0_wt                   ;use the MACRO to load the data to PIO
;
;an example of reading the data from Port A
;
        r0_rd                   ;The data will read from PA to r6
;
;an example of AND the data at Port A
;
        r0_rd                   ;read data from PA to r6
        and     pio_data,#data  ;AND    data with r6
        r0_wt                   ;writ data back to PIO PA
```

Figure 5. Application of MACRO (continued)

Now, change the MACRO when you finish the debugging (Table 2)

Table 2. Comparison of System Clocks (CLK).

PIO MACRO	CLK *	Z8 instructions	CLK *
[WRITE DATA TO PIO PA] ld pio_data,#data r0_wt	 6 12	[WRITE DATA TO Z8 PORT 0] ld r0,#data	 6
[WRITE DATA TO PIO PA] ld 6,#data p0_wt	 10 38	[WRIT DATA TO Z8 PORT 0] ld 0,#data	 10
[READ DATA FROM PIO PA] r0_rd	 12	[READ DATA FROM Z8 PORT 0) ld pio_data,r	 10
[READ DATA FROM PIO PA] p0_rd	 38	[READ DATA FROM Z8 PORT 0] ld pio_data,0	 10
[AND DATA AT PIO PA] r0_rd and pio_data,#data r0_wt	 12 10 12	[AND DATA AT Z8 PORT 0] and r0,#data	 10
[AND DATA AT PIO PA] p0_rd and 6,#data p0_wt	 38 10 38	[AND DATA AT Z8 PORT 0] and 0,#data	 10

* Number of System Clock cycles

APPLICATION LIMITATIONS

1. The P0 and P1 emulations are not real time.

2. Driving capability.

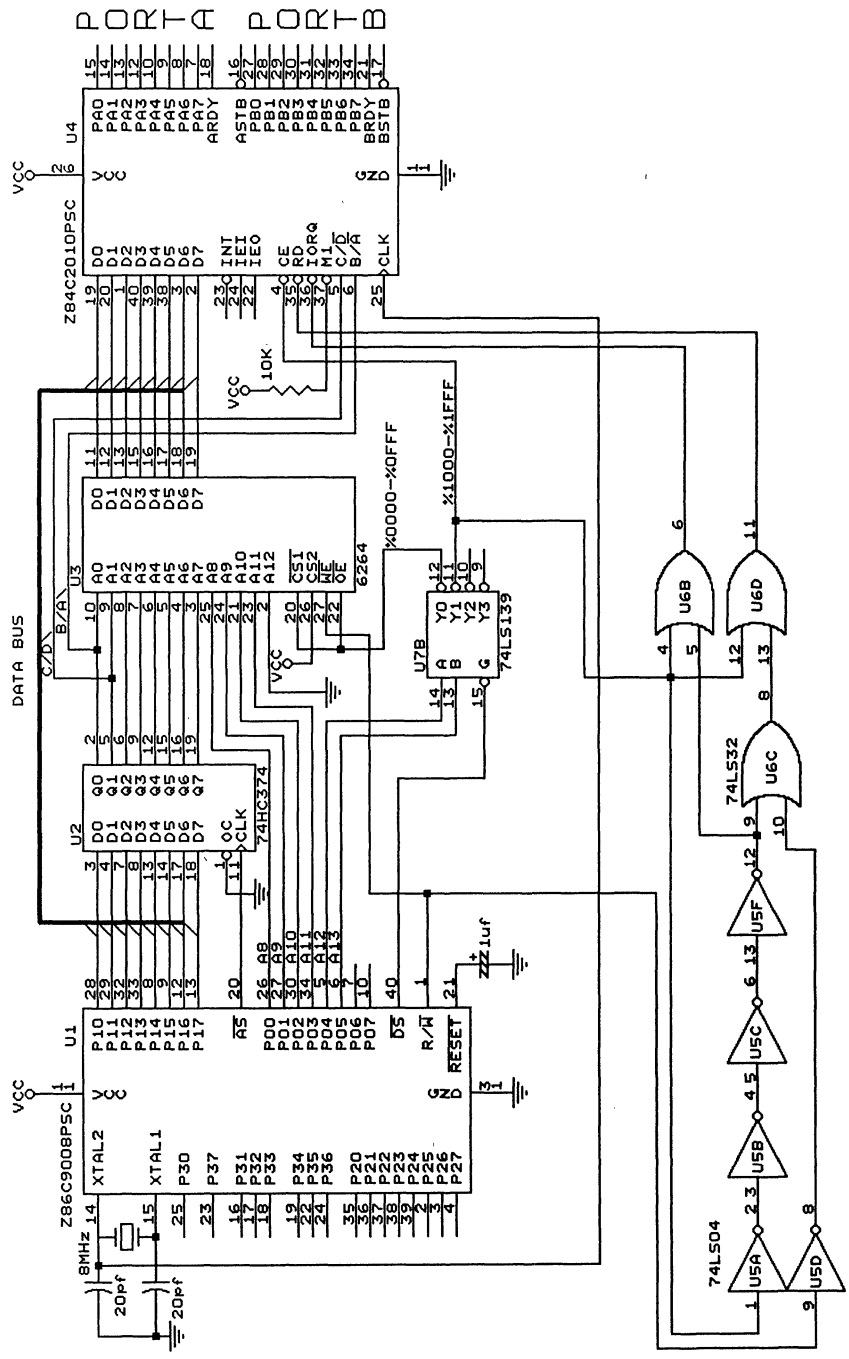


Figure 6. Z86C90 Interface to The CMOS Z80 PIO

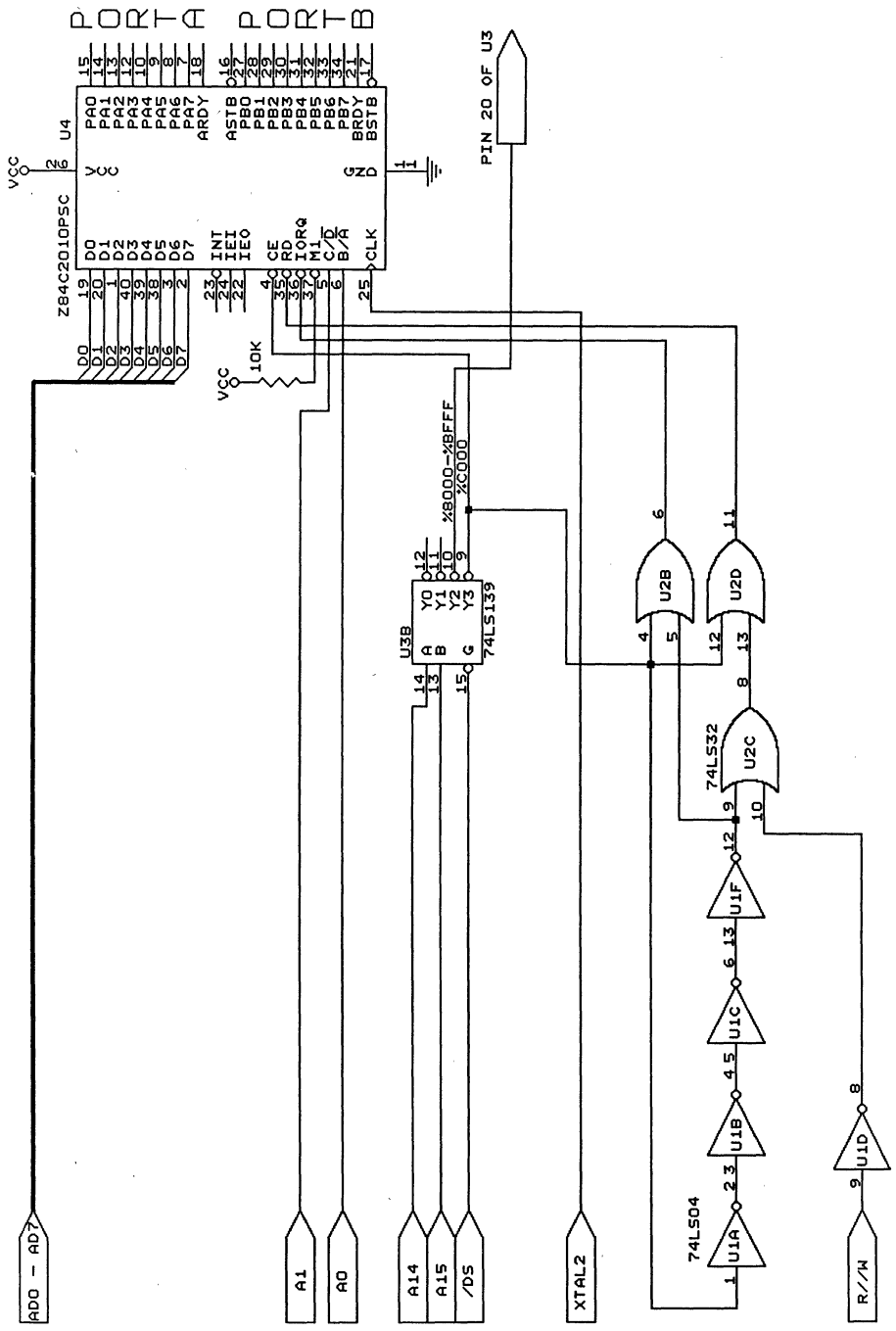


Figure 7. Z86C9000ZEM Interface with the CMOS Z80 PIO

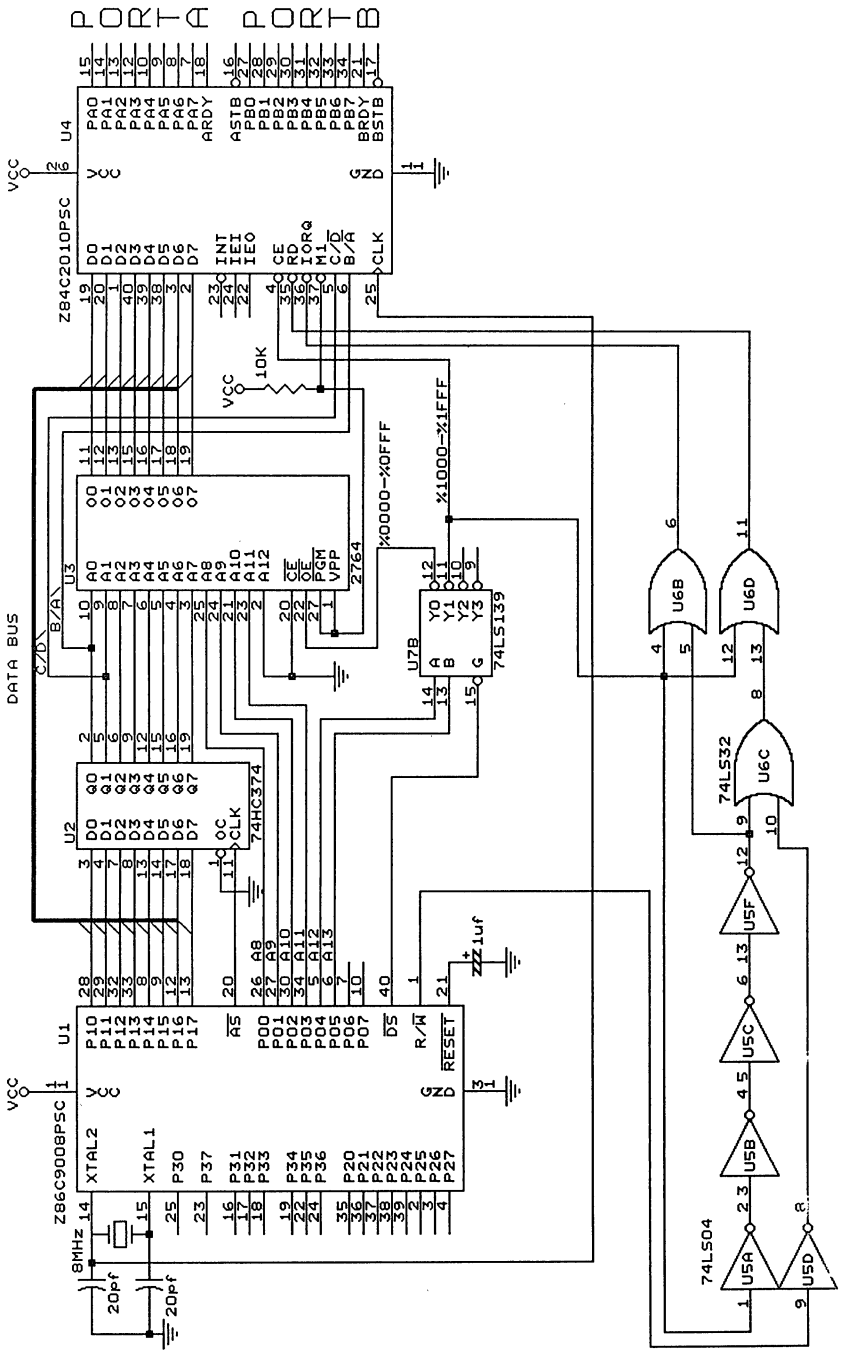


Figure 8. Z86C90 ICE to the CMOS Z80 PIO

LOW COST Z8 MCU EMULATOR

A powerful emulation tool in a small, low-cost, 18-pin package ... and this is only one aspect of this versatile CCP

INTRODUCTION

The Z86C09 and Z86C19 are low cost powerful microcontrollers that embody the full core of the Consumer Control Processor (CCP) in a small 18-pin package. In addition to its small size, dual analog comparators, and low power modes of operation, the watchdog timers make these products useful in many applications.

An emulator is an excellent development tool for economical code development, for reducing the expense of using One Time Programmable (OTP) parts during early development, and primarily for reducing risk before going to a maskable ROM part. Commercial emulators have their applications, but in many instances are not required. One case where the investment in a commercial emulator is unwanted, is during part evaluation and parameter testing before initial application development actually begins. The simple emulator outlined in this Application Note is also a sufficient development tool for applications requiring only small amounts of code.

CCP EMULATOR DETAILED DESCRIPTION

The basic 18-pin CCP Emulator is very simple to build and use. Its major parts are the Z86C90 ROMless CCP microcontroller, an EEPROM, and some Address/Data demultiplexing logic. An EEPROM serves as an excellent development device with the ROMless CCP microcontroller as it allows endless versions of code checking and modification with little effort.

The complete schematic for the basic Low Cost Z8 MCU Emulator is shown in Figure 1. The pin-outs of the components shown are for Dual In-Line Package (DIP) parts. This circuit configuration provides the user with the most basic real-time hardware emulation capability. To maximize the ease of use, a Zero Insertion Force (ZIF) socket should be used for the EEPROM. The core building block can be enhanced in a number of different ways to provide the user with an emulator that can be tailored to the specific needs of the user.

Oscillator and Power Considerations

The schematic diagram shown in Figure 1 assumes that a ceramic resonator or crystal is being used as a clocking source for the microcontroller. Note, the source should be directly applied to the Z86C90. This provides a more accurate representation of the oscillator performance than the alternative method of pumping the clocking signals through a cable, via the socketing connector, to the Z86C90. There are two significant reasons for this. First, the cable adds a significant capacitive load to these signals. Secondly, because the clock is the highest frequency signal, it is more susceptible to distortion and noise.

The emulator consumes more power than the target microcontroller. As a result, basic power distribution and filtering rules apply to the emulator power source. It is recommended that the same power source use both the targeted microcontroller and the emulator rather than using a separate isolated supply. The schematic (Figure 1) accounts for 0.1uf capacitors placed near the VCC pins of each of the active devices. A 10uf capacitor is placed near the emulator input power source.

Multiplexed Address/Data Logic

For this emulator application, Port 1 of the Z86C90 is configured as a multiplexed Address/Data bus, and Port 0 is configured as the upper portion of the Address Bus. This gives the emulator the capability of addressing more than the 4K of ROM memory limit imposed by the Z86C19. The EEPROM (Xicor X2864B-18) is an 8Kx8 device that allows twice the program storage memory of the Z86C19 and four times the storage of the Z86C09. This extra memory is useful for patching the code under development. Extra memory allows the programmer to concentrate on code development as a primary concern, and then code optimization and "squeezing" can be a secondary concern.

Because the Address and Data buses of the Z86C90 are multiplexed, they are separated for accessing program memory. The Z86C90 makes this task easy by supplying the /AS (Address Strobe - active low) and the /DS (Data Strobe - active low) signals. An inverted /AS signal is used by the transparent latches to hold the Address/Data for the EEPROM. The /DS signal is used as on Output Enable (/OE - active low) for the EEPROM to place the program data on the multiplexed Address/Data bus. A timing diagram of the program memory

access is shown in Figure 2. Note that the usage of program memory is always a read operation by the Z86C90 when emulating the Z86C09/19.

Table 1. Emulator EEPROM Timing Parameters

No.	Parameter	Min	Max
1	Address valid to /AS high delay	35	
2	Address float to /DS active delay	0	
3	/AS inactive to /DS active	65	
4	/DS active to EEPROM data valid delay		100
5	MCU address valid to EEPROM address valid		44
6	EEPROM address valid to EEPROM valid data		180
7	/AS inactive to Data input required		250
8	/DS active to Data input required		130
9	Data input setup time to /DS inactive	75	
10	Data input hold time to /DS inactive	0	
11	/DS inactive to EEPROM data float	0	50
12	/DS inactive to MCU address valid	65	

SPECIAL PROGRAMMING

There are a few programming considerations when using the Z86C90 to emulate the Z86C09/19. First, the Z86C90 has 236 General Purpose registers (R4-R239) while the Z86C09/19 is limited to 124 General Purpose registers (R4-R127).

In addition to the number of general purpose registers available to the user, there are also some differences in the control and status registers (R240-R255) between the devices. The first of these differences occurs in the Watchdog Timer Mode Register [WTMR - Extended Register %(F)0F]. The differences occur if you program bit D4 of this register to select the XTAL option as the watchdog timer driving source (D4=1). By using the on-chip RC circuit (the default condition) there are no differences in the watchdog timer activation periods. The differences in the watchdog timer activation periods between the devices when the crystal option is selected is outlined in Table 2.

Table 2. WDTMR XTAL Differences

D1	D0	XTAL1 Timeout Z86C09/19	XTAL1 Timeout Z86C90
0	0	XTAL1/512	XTAL1/256
0	1	XTAL1/1024	XTAL1/512
1	0	XTAL1/2048	XTAL1/1024
1	1	XTAL1/8192	XTAL1/4096

Take care when programming the Port 3 Mode Register (R247 P3M). To properly emulate the Z86C09/19, bits D7:D2 must be programmed as all zeros. This sets P31, P32 and P33 as inputs; P34, P35 and P36 as outputs. Bits D7:D2 are reserved in the Z86C09/19, so setting these bits to 0 will have no effect.

The Port 0 and Port 1 Mode Register (R248 P01M) is one register that must be programmed differently between the devices. To properly emulate the Z86C09/19, the P01M register of the Z86C90 is set to 96h. However, bit D4 of the Z86C09/19 register must be set to 0. The remaining bits of the P01M register in the Z86C09/19 are reserved, and should be programmed as all zeros. It is important that these differences are remembered when converting the code from one processor to the other.

The RAM protect option of the Z86C90 (R251 - IMR, Bit D6) should not be enabled. This bit should be programmed as a 0 for both types of devices.

ADDITIONAL ENHANCEMENTS

Additional EEPROM (or other) memory is supported by using the unused latched address lines, a 3:8 demultiplexer, and the /CE inputs of the EEPROMS. A full 64K of memory is accessed using the schematic shown in Figure 3. Because additional circuitry is being added, there is an additional time delay in data availability. The maximum specified value, from /DS going active to data required by the Z86C9012, is 130 ns. A faster EEPROM (Xicor X2864B-12) may be required. This depends on the clock speeds being used in the application.

By adding some RAM, and setting bit D2 of register 248 to a one, an external stack can be added. This is useful for debugging applications that are interrupt intensive. Registers 254 and 255 are programmed to map into the appropriate RAM space.

Additional memory is not the only useful enhancement. Other useful features include single stepping, breakpoints, real time traces, and adding a direct computer link. Unfortunately the Z86C90 is not the perfect In Circuit Emulator (ICE) chip, and adding single stepping and breakpoints is not done easily. This is where the commercial emulators come in. However, adding real time trace capability and an external computer link are possible.

A real time trace is achieved by storing the latched address information and EEPROM data. These values are stored at the rising edges of /AS and /DS. The user

then designs circuitry control of the trace (turn on, turn off, stop when full, overwrite, etc.).

Adding a computer link is probably the most challenging task, and for those who do not like removing the EEPROM and placing it in a programmer device, it is a useful enhancement. Also, it controls other enhancements like the trace function and can report the contents of an external stack. A logical candidate for this application is the Z86C91. The UART on this microcontroller can be used to execute an RS-232 interface. The design rules for the Z86C91 are the same as those outlined for the Z86C90, so half of the design is already complete!

CONCLUSION

The Z8 microcontroller family is very powerful. The CCP series offers very cost effective solutions for consumer and automotive applications. Emulating these devices is simple and cost effective plus it provides keen insight into their specific uses.

REFERENCES

The following Zilog publications contain specific information on the use and programming of the Z8 CCP microcontroller:

- o Z8 Family Design Handbook 03-8725-03
- o Z86C09/19 Product Specification 00-2506-01
- o Z86C40/90 Product Specification 00-2510-01

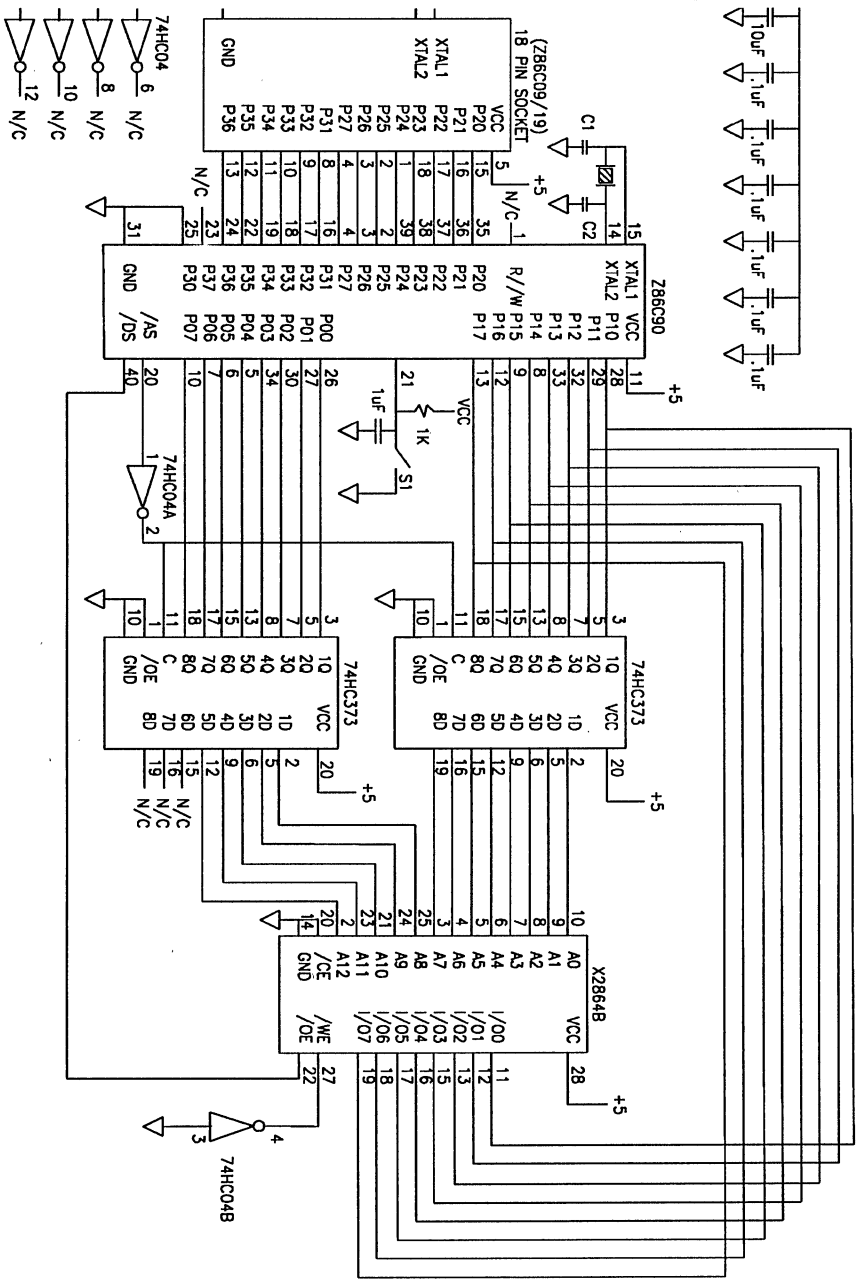


Figure 1. Low Cost Z8 MCU Emulator

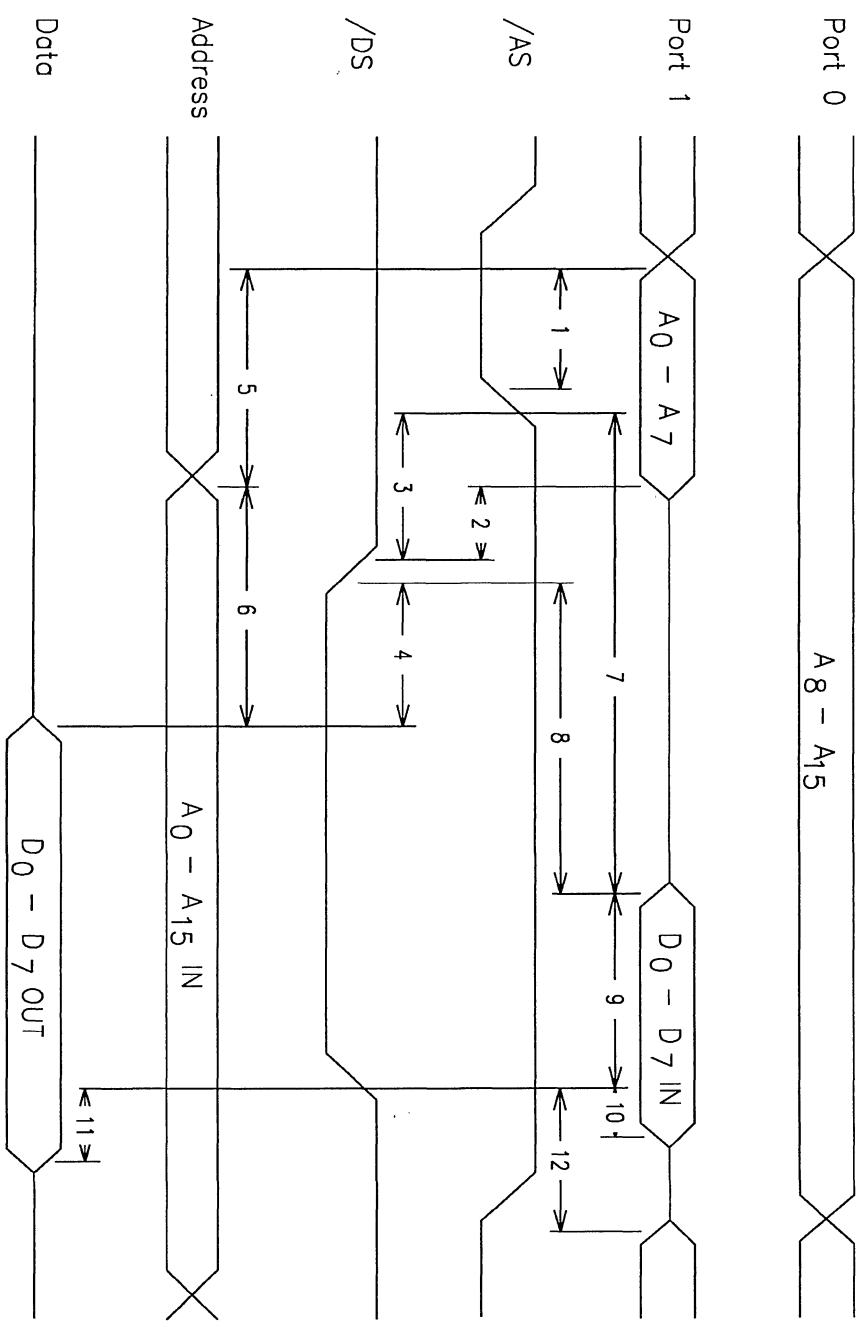


Figure 2. Emulator EEPROM Memory Timing

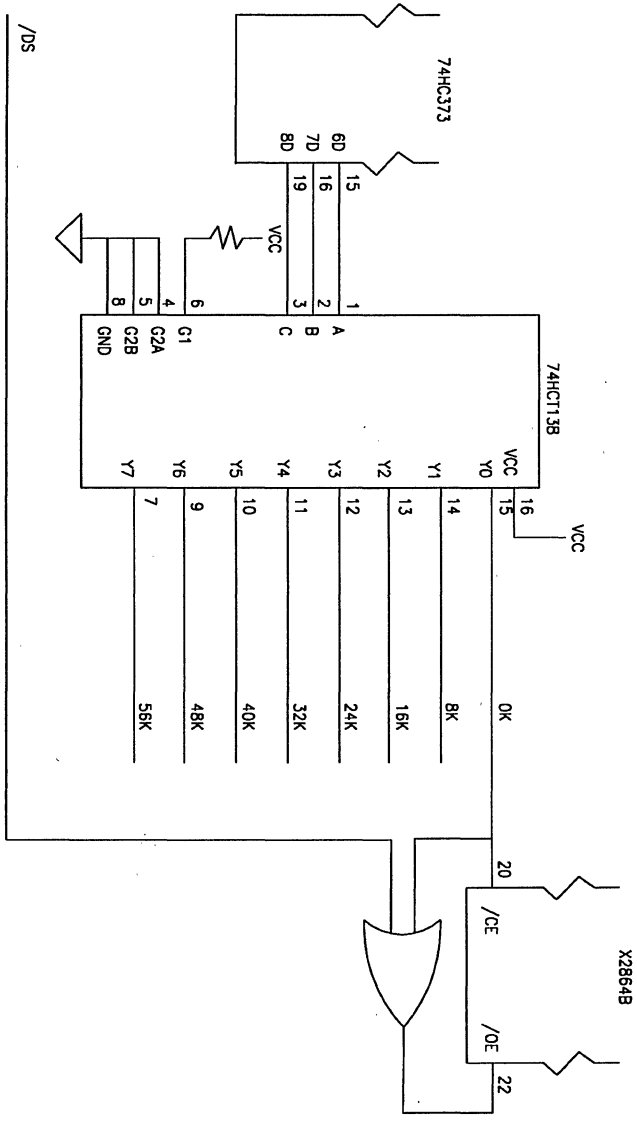


Figure 3. Additional Emulator Memory

Z8602 Controls a 101/102 PC/Keyboard

Scan codes, line status modes, key bounce, make/break status, scan timing... the Z8602 Microcontroller does all this and much more.

Keyboard Configuration and Functions

The Z8602 Microcontroller is designed into a PC keyboard to control all scan codes, line status modes, scan timing and communications between the keyboard and PC. This Application Note depicts a typical method of interfacing a standard keyboard to an XT/AT PC.

The Keyboard includes: 101/102 keypads, three LED indicators for lock status, a cable wire between PC and Keyboard, a selection switch for PC/XT and PC/AT Scan Codes, and a Z8602 microcontroller for the control (Figure 1).

The three indicators are Num Lock, Caps Lock and Scroll Lock. The lock status transfers back from the PC soon after the Key Scan Code transmits.

The keyboard has three Key Scan Code Sets. Scan Code Set 1 uses PC/XTs and Scan Code Set 2 uses PC /ATs. Scan Code Set 3 is similar to Scan Code Set 2 except for the different typematic, make, and make/break defaults (Table 1). It is enabled by software. The initial status of the Scan Code Set is specified by the selection switch. Scan Code Set 1 activates when the switch closes. Scan Code Set 2 is selected if it is open (switch shown at bottom of Figure 4).

5 Volt power supply and common ground are supplied by the cable from the PC with bidirectional Data and Clock lines for serial data communication (Figure 2).

Keyboard Scanning and the Keyboard Buffer

The keyboard contains 101/102 keypads. All keypads are scanned every 4.17 milliseconds by the Z8602. The microcontroller handles a maximum of six keys concurrently by means of the key bouncethe key bounce process and case conditions. Quick multiple key pressing for the first six keys generate the Scan Codes. If more than six keys are pressed concurrently they are ignored.

Each keypad sends multiple data bytes to the PC under the control of the Z86C02 . This is a scan code. There are two kinds of Scan Codes called Make Code and Break Code. The Make Scan Code is sent to the PC when the key is pressed. When the key releases, its Break Scan Code is sent. Additionally, these keys are Typematic which means when a key is pressed and held down, the keyboard sends the Make Code with a particular delay and rate. The typematic delay and rate are specified by the F3 (Hex) command sent by the PC.

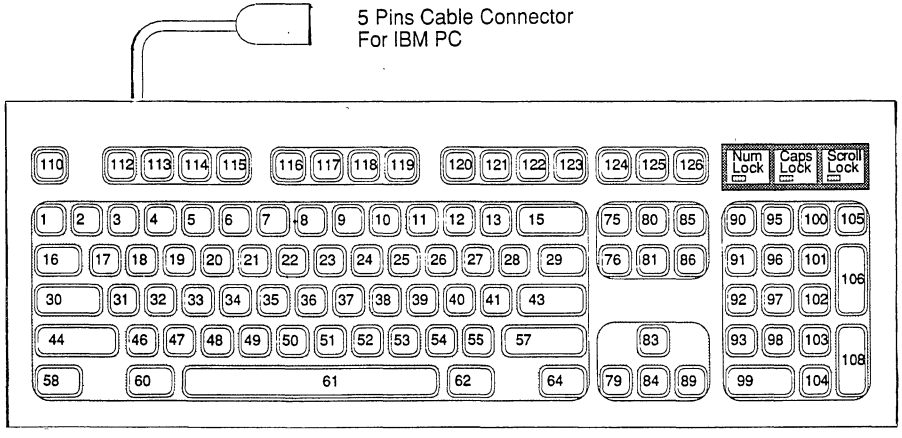
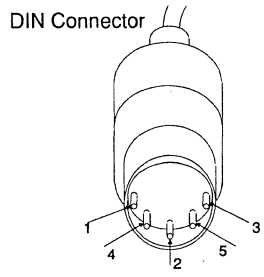


Figure 1. PC Keyboard 101/102 Overview



DIN Pins	Signal Name	Signal Type
1	+KBD CLK	Input/Output
2	+KBD DATA	Input/Output
3		
4	Ground	Power
5	+5.0 Vdc	Power
SHIELD	Frame GND	

Figure 2. PC Cable Connector

Keyboard Code Generation

There are three program modules to implement keyboard code generation; keyboard scanning, Make/Break/Typematic timing control and Scan Code generation. The modules are serviced by the Timer 1 interrupt.

The keyboard scanning module cuts key bounce for both press and release. It also configures the FIFO buffer for a maximum of six keys and allows time to generate both Make code and Break code.

The Make/Break/Typematic timing control module checks the current key status for up to six keys. It also sets up the timing for Make Code, Break Code and typematic delay and rate.

The Make/Break Scan Code generation module transfers Make code and Break code into the FIFO buffer via several ROM tables.

Keyboard Scanning

The keyboard scanning repeats every 4.17 milliseconds for all keypads. The key scanning is done from column output 15 toward column output 0 by keeping one of the column outputs to a low level and the remaining outputs at a high level. Whenever a low level output sets on one of the column ports, eight row inputs test 20 microseconds later. The timing chart of keyboard scanning is illustrated in Figure 3.

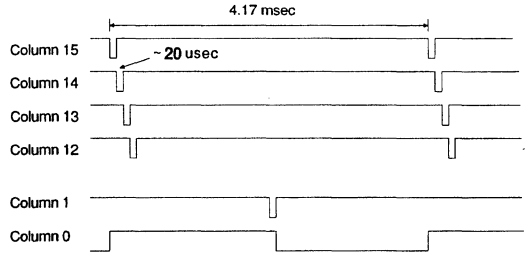


Figure 3. Keyboard Scan Timing

The key bounce ends by detecting the key pressing two times using the scanning method. Once a key is detected, it converts to a key matrix number (the key matrix number is the index address of the Scan Code in Table 1) through the row and column data appearing in Figures 4 and 5. This process repeats until all six keys fill the buffer. When the key is detected twice, Make code status sets. When the key releases twice, Break code status sets. The state diagram of the Make code and Break code process for one particular key is illustrated in Figure 6.

Table 1. Scan Code Set Table

The following table shows three scan code sets used in the keyboard.

Key Number	Scan Code Set 1		Scan Code Set 2	
	Make Code	Break Code	Make Code	Break Code
1	29	A9	0E	F0 0E
2	02	82	16	F0 16
3	03	83	1E	F0 1E
4	04	84	26	F0 26
5	05	85	25	F0 25
6	06	86	2E	F0 2E
7	07	87	36	F0 36
8	08	88	3D	F0 3D
9	09	89	3E	F0 3E
10	0A	8A	46	F0 4E
11	0B	8B	45	F0 45
12	0C	8C	4E	F0 4E
13	0D	8D	55	F0 55
15	0E	8E	66	F0 66
16	0F	8F	0D	F0 0D
17	10	90	15	F0 15
18	11	91	1D	F0 1D
19	12	92	24	F0 24
20	13	93	2D	F0 2D
21	14	94	2C	F0 2C
22	15	95	35	F0 35
23	16	96	3C	F0 3C
24	17	97	43	F0 43
25	18	98	44	F0 44
26	19	99	4D	F0 4D
27	1A	9A	54	F0 54
28	1B	9B	5B	F0 5B
29	2B	AB	5D	F0 5D
30	3A	BA	58	F0 58
31	1E	9E	1C	F0 1C
32	1F	9F	1B	F0 1B
33	20	A0	23	F0 23
34	21	A1	2B	F0 2B
35	22	A2	34	F0 34
36	23	A3	33	F0 33
37	24	A4	3B	F0 3B
38	25	A5	42	F0 42
39	26	A6	4B	F0 4B
40	27	A7	4C	F0 4C
41	28	A8	52	F0 52
42	2B	AB	5D	F0 5D
43	1C	9C	5A	F0 5A
44	2A	AA	12	F0 12
45	56	D6	61	F0 61
46	2C	AC	1A	F0 1A
47	2D	AD	22	F0 22
48	2E	AE	21	F0 21
49	2F	AF	2A	F0 2A
50	30	B0	32	F0 32
51	31	B1	31	F0 31
52	32	B2	3A	F0 3A
53	33	B3	41	F0 41
54	34	B4	49	F0 49
55	35	B5	4A	F0 4A
57	36	B6	59	F0 59
58	1D	9D	14	F0 14
60	38	B8	11	F0 11
61	39	B9	29	F0 29
62	E0 38	E0 B8	E0 11	E0 F0 11
64	E0 1D	E0 9D	E0 14	E0 F0 14
90	45	C5	77	F0 77
91	47	C7	6C	F0 6C
92	4B	CB	6B	F0 6B
93	4F	CF	69	F0 69
96	48	C8	75	F0 75
97	4C	CC	73	F0 73
98	50	D0	72	F0 72
99	52	D2	70	F0 70
100	37	B7	7C	F0 7C
101	49	C9	7D	F0 7D
102	4D	CD	74	F0 74
103	51	D1	7A	F0 7A
104	53	D3	71	F0 71
105	4A	CA	7B	F0 7B
106	4E	CE	79	F0 79
108	E0 1C	E0 9C	E0 5A	E0 F0 5A
110	01	81	76	F0 76
112	3B	BB	05	F0 05
113	3C	BC	06	F0 06
114	3D	BD	04	F0 04
115	3E	BE	0C	F0 0C
116	3F	BF	03	F0 03
117	40	C0	0B	F0 0B
118	41	C1	83	F0 83
119	42	C2	0A	F0 0A
120	43	C3	01	F0 01
121	44	C4	09	F0 09
122	57	D7	78	F0 78
123	58	D8	07	F0 07
125	46	C6	7E	F0 7E

Key Number	Scan Code Set 1		
	Base Case, Shift + Num Lock Make/Break	Shift Case * Make/Break	Num Lock On Make/Break
75	E0 52 / E0 D2	E0 AA E0 52 /E0 D2 E0 2A	E0 2A E0 52 /E0 D2 E0 AA
76	E0 53 / E0 D3	E0 AA E0 53 /E0 D3 E0 2A	E0 2A E0 53 /E0 D3 E0 AA
79	E0 4B / E0 CB	E0 AA E0 4B /E0 CB E0 2A	E0 2A E0 4B /E0 CB E0 AA
80	E0 47 / E0 C7	E0 AA E0 47 /E0 C7 E0 2A	E0 2A E0 47 /E0 C7 E0 AA
81	E0 4F / E0 CF	E0 AA E0 4F /E0 CF E0 2A	E0 2A E0 4F /E0 CF E0 AA
83	E0 48 / E0 C8	E0 AA E0 48 /E0 C8 E0 2A	E0 2A E0 48 /E0 C8 E0 AA
84	E0 50 / E0 D0	E0 AA E0 50 /E0 D0 E0 2A	E0 2A E0 50 /E0 D0 E0 AA
85	E0 49 / E0 C9	E0 AA E0 49 /E0 C9 E0 2A	E0 2A E0 49 /E0 C9 E0 AA
86	E0 51 / E0 D1	E0 AA E0 51 /E0 D1 E0 2A	E0 2A E0 51 /E0 D1 E0 AA
89	E0 4D / E0 CD	E0 AA E0 4D /E0 CD E0 2A	E0 2A E0 4D /E0 CD E0 AA

* If the left shift Key is held down, the AA/2A shift make and break is sent with the other scan codes. If the right Shift key is held down, B6/36 is sent. If both Shift keys are down, both sets of codes are sent with the other scan code.

Table 1. Scan Code Set Table (Continued)

Scan Code Set 1		
Key No.	Make/Break Code	Shift Case Make/Break *
95	E0 35 / E0 B5	E0 AA E0 35 / E0 B5 E0 2A
*If the left Shift key is held down, the AA/2A shift make and break is sent with the other scan codes. If the right Shift key is held down, B6/36 is sent. If both Shift keys are down, both sets of codes are sent with the other scan code.		

Scan Code Set 1			
Key No.	Make/Break Code	Ctrl, Shift Case Make/Break	Alt Case Make/Break
124	E0 2A E0 37 /E0 B7 E0 AA	E0 37 /E0 B7	54/D4

Scan Code Set 1		
Key No.	Make Code	Ctrl Key Pressed
126	E1 1D 45 E1 9D C5	E0 46 E0 C6
*This key is not typematic. All associated scan codes occur on the make of the key.		

Key Number	Scan Code Set 2 Base Case, Shift + Num Lock Make/Break	Shift Case * Make/Break	Num Lock On Make/Break
75	E0 70 / E0 F0 70	E0 F0 12 E0 70 /E0 F0 70 E0 12	E0 12 E0 70 /E0 F0 70 E0 12
76	E0 71 / E0 F0 71	E0 F0 12 E0 71 /E0 F0 71 E0 12	E0 12 E0 71 /E0 F0 71 E0 12
79	E0 6B/E0 F0 6B	E0 F0 12 E0 6B /E0 F0 6B E0 12	E0 12 E0 6B /E0 F0 6B E0 12
80	E0 6C/E0F0 6C	E0 F0 12 E0 6C /E0 F0 6C E0 12	E012 E0 6C /E0 F0 6C E0 12
81	E0 69 / E0 F0 69	E0 F0 12 E0 69 /E0 F0 69 E0 12	E0 12 E0 69 /E0 F0 69 E0 12
83	E0 75 / E0 F0 75	E0 F0 12 E0 75 /E0 F0 75 E0 12	E0 12 E0 75 /E0 F0 75 E0 12
84	E0 72 / E0 F0 72	E0 F0 12 E0 72 /E0 F0 72 E0 12	E0 12 E0 72 /E0 F0 72 E0 12
85	E0 7D/E0 F0 7D	E0 F0 12 E0 7D /E0 F0 7D E0 12	E012 E0 7D /E0 F0 7D E0 12
86	E0 7A/E0 F0 7A	E0 F0 12 E0 7A /E0 F0 7A E0 12	E0 12 E0 7A /E0 F0 7A E0 12
89	E0 74 / E0 F0 74	E0 F0 12 E0 74 /E0 F0 74 E0 12	E0 12 E0 74 /E0 F0 74 E0 12
* If the left shift Key is held down, the F0 12/12 shift make and break is sent with the other scan codes. If the right Shift key is held down, F0 59/59 is sent. If both Shift keys are down, both sets of codes are sent with the other scan code.			

Scan Code Set 2		
Key No.	Make/Break Code	Shift Case Make/Break *
95	E0 4A / E0 F0 4A	E0 F0 12 4A / E0 12 F0 4A
*If the left Shift key is held down, the AA/2A shift make and break is sent with the other scan codes. If the right Shift key is held down, B6/36 is sent. If both Shift keys are down, both sets of codes are sent with the other scan code.		

Scan Code Set 2			
Key No.	Make/Break Code	Ctrl, Shift Case Make/Break	Alt Case Make/Break
124	E0 12 E0 7C /E0 F0 7C E0 F0 12	E0 7C /E0 F0 7C	84/F0 84

Scan Code Set 2		
Key No.	Make Code	Ctrl Key Pressed
126	E1 14 77 E1 F0 14 F0 77	E0 7E E0 F0 7E
*This key is not typematic. All associated scan codes occur on the make of the key.		

Scan Code Set 3 T=Typematic, M/B=Make/Break, M=Make only							
Key No.	Make Code	Break Code	Default Stratus	Key No.	Make Code	Break Code	Default Status
1	0E	F0 0E	T	51	31	F0 31	T
2	16	F0 16	T	52	3A	F0 3A	T
3	1E	F0 1E	T	53	41	F0 41	T
4	26	F0 26	T	54	49	F0 49	T
5	25	F0 25	T	55	4A	F0 4A	T
6	2E	F0 2E	T	57	59	F0 59	M/B
7	36	F0 36	T	58	11	F0 11	M/B
8	3D	F0 3D	T	60	19	F0 19	M/B
9	3E	F0 3E	T	61	29	F0 29	T
10	46	F0 46	T	62	39	F0 39	M
11	45	F0 45	T	64	58	F0 58	M
12	4E	F0 4E	T	75	67	F0 67	M
13	55	F0 55	T	76	64	F064	T
15	66	F0 66	T	79	61	F0 61	T
16	0D	F0 0D	T	80	6E	F0 6E	M
17	15	F0 15	T	81	65	F0 65	M
18	1D	F0 1D	T	83	63	F0 63	T
19	24	F0 24	T	84	60	F0 60	T
20	2D	F0 2D	T	85	6F	F0 6F	M
21	2C	F0 2C	T	86	6D	F0 6D	M
22	35	F0 35	T	89	6A	F0 6A	T
23	3C	F0 3C	T	90	76	F0 76	M
24	43	F0 43	T	91	6C	F0 6C	M
25	44	F0 44	T	92	6B	F0 6B	M
26	4D	F0 4D	T	93	69	F0 69	M
27	54	F0 54	T	95	77	F0 77	M
28	5B	F0 5B	T	96	75	F0 75	M
29	5C	F0 5C	T	97	73	F0 73	M
30	14	F0 14	M/B	98	72	F0 72	M
31	1C	F0 1C	T	99	70	F0 70	M
32	1B	F0 1B	T	100	7E	F0 7E	M
33	23	F0 23	T	101	7D	F0 7D	M
34	2B	F0 2B	T	102	74	F0 74	M
35	34	F0 34	T	103	7A	F0 7A	M
36	33	F0 33	T	104	71	F0 71	M
37	3B	F0 3B	T	105	84	F0 84	M
38	42	F0 42	T	106	7C	F0 7C	T
39	4B	F0 4B	T	108	79	F0 79	M
40	4C	F0 4C	T	110	08	F0 08	M
41	52	F0 52	T	112	07	F0 07	M
42	53	F0 53	T	113	0F	F0 0F	M
43	5A	F0 5A	T	114	17	F0 17	M
44	12	F0 12	M/B	115	1F	F0 1F	M
45	13	F0 13	T	116	27	F0 27	M
46	1A	F0 1A	T	117	2F	F0 2F	M
47	22	F0 22	T	118	37	F0 37	M
48	21	F0 21	T	119	3F	F0 3F	M
49	2A	F0 2A	T	120	47	F0 47	M
50	32	F0 32	T	121	4F	F0 4F	M
				122	56	F0 56	M
				123	5E	F0 5E	M
				124	57	F0 57	M
				125	5F	F0 5F	M
				126	62	F0 62	M

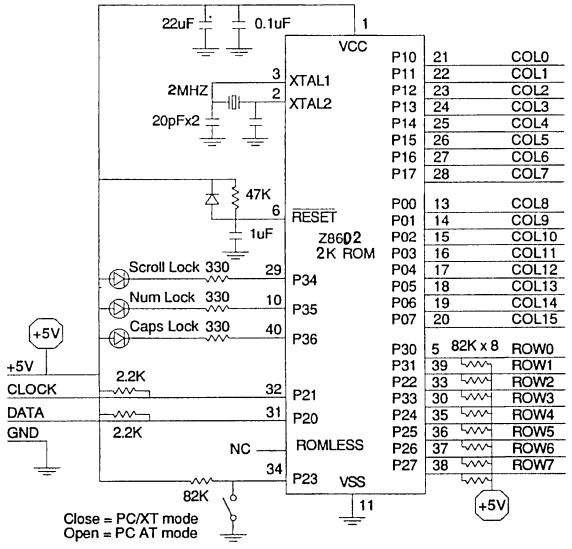


Figure 4. Z8602 Schematic Diagram for the Keyboard

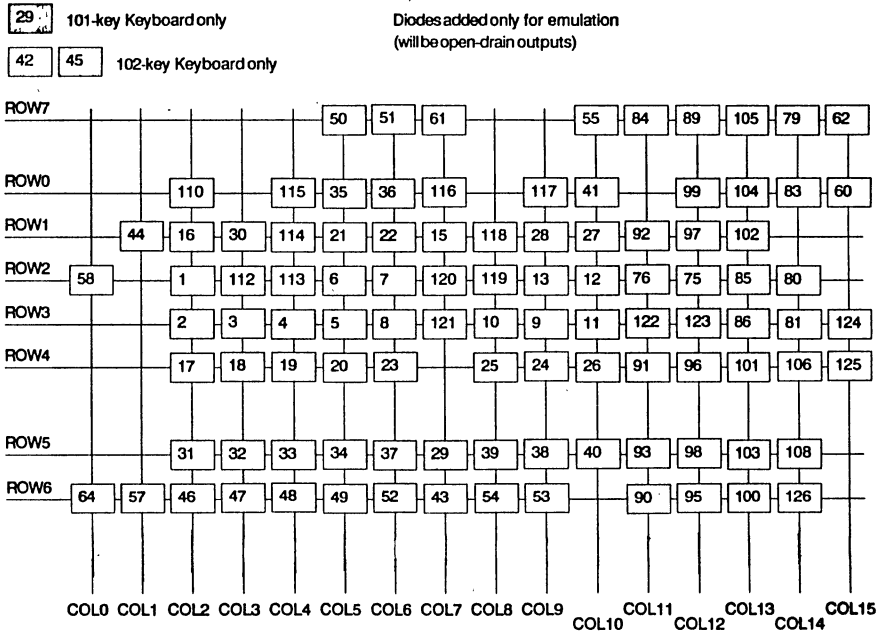


Figure 5. 101/102 Keyboard Matrix

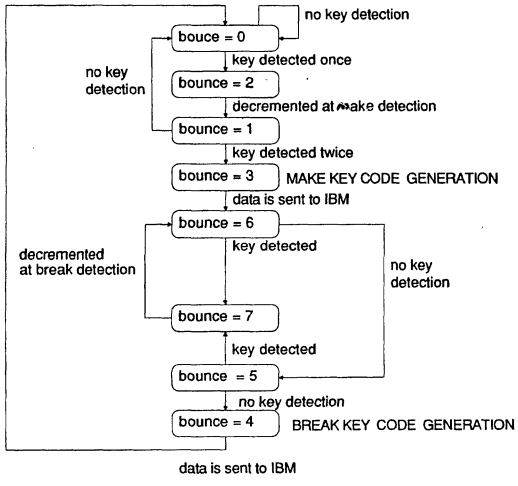


Figure 6. Make/Break Code Generation State Diagram

Six pairs of working registers are manipulated in the keyboard scanning program. They are KEY_STATUS and KEY_DATA. The key bounce is handled in the bounce counter of KEY_STATUS (Figure 7).

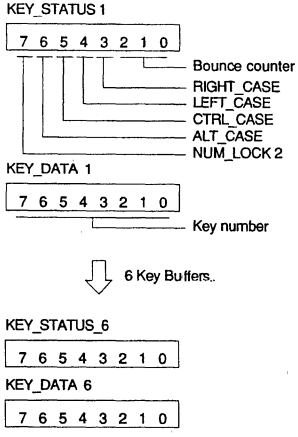


Figure 7. Six Pairs of Key Buffers

State Diagram Explanation (Figure 6):
The converted key matrix number and the initial bounce data store in one of the empty KEY_DATA and KEY_STATUS registers, respectively, just after detecting the key. When the key is detected once (bounce =2), it is decremented at Make Detection to establish one bounce detection (bounce=1). The loop continues until the key is detected twice. When detected twice, the bounce bit is three (bounce = 3) until generation of the Make code. Then the bounce bits change to six (bounce = 6) in the Make/Break code generation module.

The key detection loop continues (loop from bounce=5, bounce=7, bounce=6) until two key release detections (from bounce=6 to bounce=5). The bounce bit is set to four (bounce=4) when the key release is detected two times. The number is stored until generation of the Break code. Then, it is reset to zero in the Make/Break module. The first six keys generate the Make Code and Break Code when multiple keys are pressed. Concurrently, the rest of the keys are ignored.

Make/Break/Typematic Timing Control

The make code is generated when the bounce bits of KEY_STATUS is three. Then, the delay timer is calculated by using the TYPEMATIC_RATE register (Figure 8•) and sets to one of the periods which are 250 milliseconds, 500 msec, 750 msec or 1 second. The keys for the left/right shift case, control case and alternate case test to hold the current key configuration case. The Num lock, Caps lock, and Scroll lock status cases are stored in three of the six KEY_STATUS registers to keep the same pair of Make and Break codes for each key. The Make code is stored in the FIFO keyboard buffer.

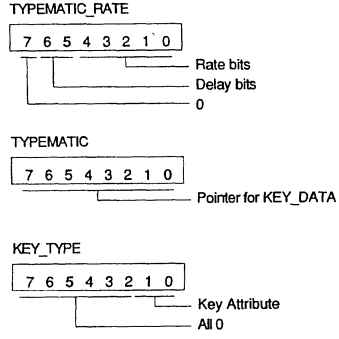


Figure 8. Key Control Registers

The current pointer of KEY_DATA stores in the TYPEMATIC pointer register to generate Make Scan Code when typematic timeout occurs. When the Make Code is generated, two flag bits in the KEY_TYPE register are set to determine the key attributes: Make-typematic-break (00), Make-typematic (01), Make-break (10) and Make only (11).

Break Code Timing Control:
Code is generated when the bounce bits of KEY_STATUS are set to four. This means the key is released. The TYPEMATIC pointer register is reset when the current typematic key releases. The case key tests and the case flag is reset when released. Then, Break Code is stored in the FIFO keyboard buffer and KEY_DATA plus KEY_STATUS reset to show an empty key. This procedure repeats for the six KEY_STATUS registers (Figure 7).

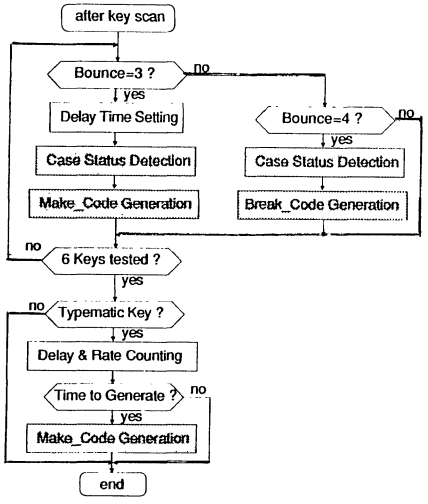


Figure 9. Flow Chart for Make/Break/Typematic Timing

Typematic Code Timing Control:
To generate typematic code, the key attribute tests to be sure the key is typematic. If the current TYPEMATIC pointer is not zero, the typematic process is carried out. The delay timer decrements every 4.17 msec and when it reaches 0, there is Make Code generation. Once the delay timer sets to 0, the rate timer takes over the typematic code generation which continues whenever the rate timer decrements to 0. The rate timer then reloads from the TYPEMATIC_RATE register.

Make/Break Scan Code Generation

The system has one macro command to expand one byte of Make Code to multiple Scan Codes. The macro is data_gen. The macro structure appears in Figure 10.

Macro Description:
After getting one byte of Make code, this is expanded to multiple scan codes by the data_gen macro command. The data_gen macro contains a total number of bytes generated (lower four bits) minus the offset value. This offset value addresses a byte that is XORed with a byte-make code and other generated data bytes.

The previously kept "index address" indicates the entry point of the data_gen macro command when the data is generated for Break Code. If it is generated for Make Code, then the index address is decremented by one and the ROM data at the address is read to specify the entry pointer for the Make Code generation.

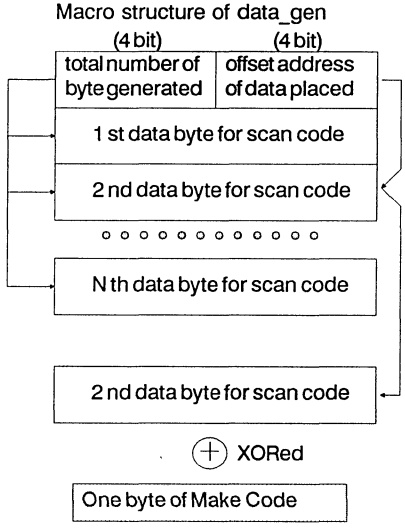


Figure 10. Macro Structure of Data_Attr and Data_Gen

Four steps for generating multiple scan codes are the following:

1. Get data for the total number of bytes generated (lower four bits of first byte).

2. Temporarily store the offset address which shows the position to be XORed with a byte-make code.

3. Store all the bytes from the ROM table to the FIFO buffer while incrementing the FIFO_SIZE and the ROM address pointer.

4. Subtract the offset address from the temporary register containing the same value of FIFO_SIZE. Change the FIFO data by taking the XOR with the byte-make code.

Scan Code Organization:
The multiple scan code generation starts from conversion of a key matrix number which points at the Key Matrix Table. The Key Matrix Table has the index address offset of the Scan Code Table. The Scan Code Set Table contains all Make Code keys. The Scan Code Table Map organization is shown in Table 2.

Typematic Attribute:
The Scan Code Set 3 is similar to Scan Code Set 2. It is the scan_code set flag (1 = Scan Code Set 1, 2 = Scan Code Set 2, and 3 = Scan Code Set 3) which specifies the Scan Code Table to use. If the scan_code_set flag is 3 (Scan Code Set 3), its data checks for typematic *attributes. The typematic attribute for scan codes (addresses 7 -83h) store in the scratchpad RAM. The four typematic attributes include; Typematic, Make/Break, Make, and Typematic/ Make/Break. To select one of the attributes, two bits decode to determine which one of the four attributes to use.

Phantom Keys:
If the key matrix number is none of the valid keys, it is the Phantom key. The Phantom key happens when multiple keypads depress concurrently. The Phantom key is zero in the Keyboard Matrix (Figure 5). If the microcontroller sees a zero from the Key Matrix Table it sends any code to the PC (data conversion for Phantom key is ignored).

Z8602 Microcontroller Control and Interface

The following subsections define the Z8602 parameters and explain the control and interfacing of the Z8602 microcontroller (located in the keyboard) to the PC.

Z8602 Pin Descriptions and Assignments

Figure 11 and Table 3 show the Z8602 pin assignments and pin descriptions , respectively. Figure 12 shows the communications format between the PC and the keyboard.

Communication Between the PC and Keyboard

Before the keyboard microcontroller (hereinafter referred to as Z8602) drives the keyboard clock and data lines, it sets both lines to a high level to check the current line status. The three line status modes between the PC and the keyboard are the following:

- Communication Inhibit by PC (PC forces clock line low).
- Request to send a data packet from PC to keyboard (PC forces data line low).
- Scan Code transmission by Keyboard (both data and clock lines are high).

The keyboard clock line is always driven by the keyboard's Z8602, except when the PC inhibits the communication. This happens by forcing the clock line to a low level (inactive level). This keeps the keyboard from sending any data packet and from generating clock pulses during this stage.

Once the clock line releases high (active level), the keyboard has to check the data line. When the data line is inactive, the PC requests to send the serial data to the keyboard. The keyboard has to send serial clock streams to receive the data packet from the PC. The data output of the Z8602 is high at this stage. When both data and clock lines are active, the keyboard sends the Scan Codes to the PC at any time.

The serial data bit stream consists of 11 bits which include a start bit, 8 data bits, an odd parity bit and a stop bit. When the bit stream sends data to the PC, all the data bits are guaranteed while the clock line is low. The data changes during a high level of the clock line. When the bit stream arrives from the PC, the data

fetches at the leading edge of the clock. After the stop bit detects high, the Z8602 forces the data line to a low level for one bit period. The start bit is always low and the stop bit is high. The 8-bit data transmits from the LSB (the least significant bit). The odd parity bit means that the number of 1's for the data bit and the parity bit must be odd all the time.

Command Communication Between the Keyboard and PC

This subsection shows how the following three kinds of data are handled between the keyboard and the PC:

- Command and Acknowledge
- Optional Data
- Key Scan Code from the Keyboard FIFO Buffer

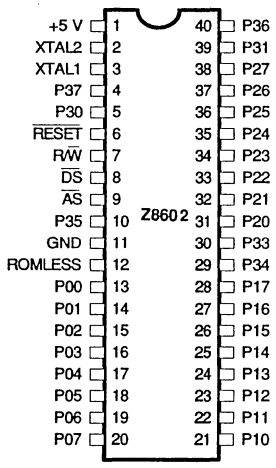


Figure 11. Z8602 Pin Assignments

Table 2. Scan Code Table Map

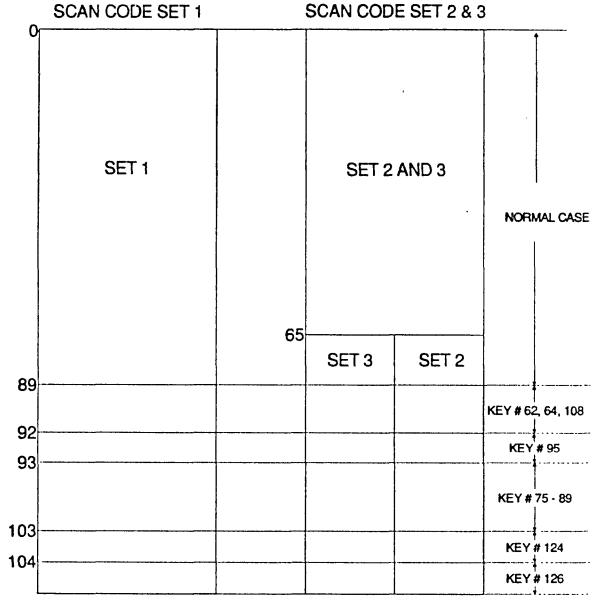


Table 3. Z8602 Pin Assignments

IN=INput port, PUR=Pull-Up Resistor, OUT=OUTput port, OD=Open-Drain output, NC=Non Connection

Pin	No.	I/O	Pin Description
+5 V	1	IN	+5 V Power Supply
GND	11	IN	Common Ground
XTAL2	2	OUT	8 MHZ Ceramic Resonator
XTAL1	3	IN	8 MHZ Ceramic Resonator
RESET	6	IN	Reset Input (active low)
R/W	7	OUT,NC	Read/Write Strobe
DS	8	OUT,NC	Data Strobe
AS	9	OUT,NC	Address Strobe
ROMless	12	IN, PUR	ROMless Selection (=GND)
P00	13	OUT,OD	Column 8 Low Output Scan Line
P01	14	OUT,OD	Column 9 Low Output Scan Line
P02	15	OUT,OD	Column 10 Low Output Scan Line
P03	16	OUT,OD	Column 11 Low Output Scan Line
P04	17	OUT,OD	Column 12 Low Output Scan Line
P05	18	OUT,OD	Column 13 Low Output Scan Line
P06	19	OUT,OD	Column 14 Low Output Scan Line
P07	20	OUT,OD	Column 15 Low Output Scan Line
P10	21	OUT,OD	Column 0 Low Output Scan Line
P11	22	OUT,OD	Column 1 Low Output Scan Line
P12	23	OUT,OD	Column 2 Low Output Scan Line
P13	24	OUT,OD	Column 3 Low Output Scan Line
P14	25	OUT,OD	Column 4 Low Output Scan Line
P15	26	OUT,OD	Column 5 Low Output Scan Line
P16	27	OUT,OD	Column 6 Low Output Scan Line
P17	28	OUT,OD	Column 7 Low Output Scan Line
P20	31	IN/OUT, OD	DATA line for IBM Communication
P21	32	IN/OUT, OD	CLOCK line for IBM Communication
P22	33	IN	Row 2 Input Scan Line
P23	34	IN	PC/XT or AT(=high) Selection
P24	35	IN	Row 4 Input Scan Line
P25	36	IN	Row 5 Input Scan Line
P26	37	IN	Row 6 Input Scan Line
P27	38	IN	Row 7 Input Scan Line
P30	5	IN	Row 0 Input Scan Line
P31	39	IN	Row 1 Input Scan Line
P33	30	IN	Row 3 Input Scan Line
P34	29	OUT	Scroll Lock Indicator
P35	10	OUT	Num Lock Indicator
P36	40	OUT	Caps Lock Indicator
P37	4	NC	No Connection

The keyboard clock line is always driven by the keyboard's Z8602, except when the PC inhibits the communication. This happens by forcing the clock line to a low level (inactive level). This keeps the keyboard from sending any data packet and from generating clock pulses during this stage.

Once the clock line releases high (active level), the keyboard has to check the data line. When the data line is inactive, the PC requests to send the serial data to the keyboard. The keyboard has to send serial clock streams to receive the data packet from the PC. The data output of the Z8602 is high at this stage. When both data and clock lines are active, the keyboard sends the Scan Codes to the PC at any time.

The serial data bit stream consists of 11 bits which include a start bit, 8 data bits, an odd parity bit and a stop bit. When the bit stream sends data to the PC, all the data bits are guaranteed while the clock line is low. The data changes during a high level of the clock line. When the bit stream arrives from the PC, the data fetches at the leading edge of the clock. After the stop bit detects high, the Z8602 forces the data line to a low level for one bit period. The start bit is always low and the stop bit is high. The 8-bit data transmits from the LSB (the least significant bit). The odd parity bit means that the number of 1's for the data bit and the parity bit must be odd all the time.

Transmission Mode	Clock	Data
Transmission Inhibit	Low	High
System to send	High	Low
Keyboard to send	High	High

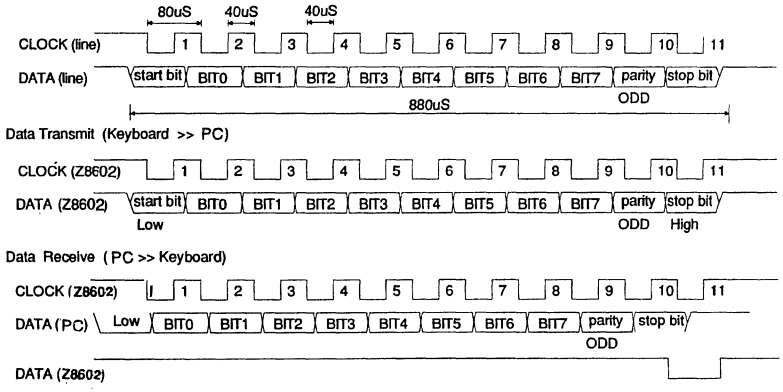


Figure 12. Data Communication Format Timing Between Z8602/PC

Command Communication Between the Keyboard and PC

This subsection shows how the following three kinds of data are handled between the keyboard and the PC:

- Command and Acknowledge
- Optional Data
- Key Scan Code from the Keyboard FIFO Buffer

The command reception has the highest priority of the data communication. A new command always overrides to the old command even during communication. The PC starts the command transfer by lowering the DATA line. Then, the Z8602 sends eleven clock pulses to receive the serial data packet from the PC. When the data arrives, the Z8602 sends an acknowledge (0FAh) and accepts an echo command (0EEh). The optional data arrives or departs after sending the acknowledge. The key scan code only sends from the keyboard FIFO buffer when no data is coming from the PC.

Table 4 shows the commands used in the PC*.

* Table 4 term definitions

ACK = Acknowledge Data to PC (0FAh)

XX = Received Data from PC.

YY = Result of Basic Assurance Test

AB 83 = ID Number

Table 4. Commands Between the Keyboard and the PC

Name	Hex values	Function
STATUS_IND	ED,ACK,XX,ACK	Set / Reset Lock Status Indicators
ECHO	EE,EE(=ACK)	Echo Command
ALT_SCAN	F0,ACK,XX,ACK	Select Alternate Scan Codes
TYPE_RATE_DELAY	F3,ACK,XX,ACK	Set Typematic Rate/Delay
ENABLE	F4,ACK	Enable Key Scanning
DISABLE	F5,ACK	Default Disable
SET_DEFAULT	F6,ACK	Set Default Value
ALL_MAKE_TYPE	F7,ACK	Set All Keys - Typematic
ALL_MAKE_BREAK	F8,ACK	Sert All Keys - Make/Break
ALL_MAKE	F9,ACK	Set All Keys - Make
ALL_M_T_B	FA,ACK	Set All Keys - Typematic/MAke/Break
KEY_MAKE_TYPE	FB,ACK,XX,ACK	Set Key Type - Typematic
KEY_MAKE_BREAK	FC,ACK,XX,ACK	Set Key Type - Make/Break
KEY_MAKE	FD,ACK,XX,ACK	Set Key Type - Make
RESEND	FE	Resend Command
RESET	FF,ACK,YY	Reset Command
READ_ID	F3,ACK,AB,83	Read ID Command

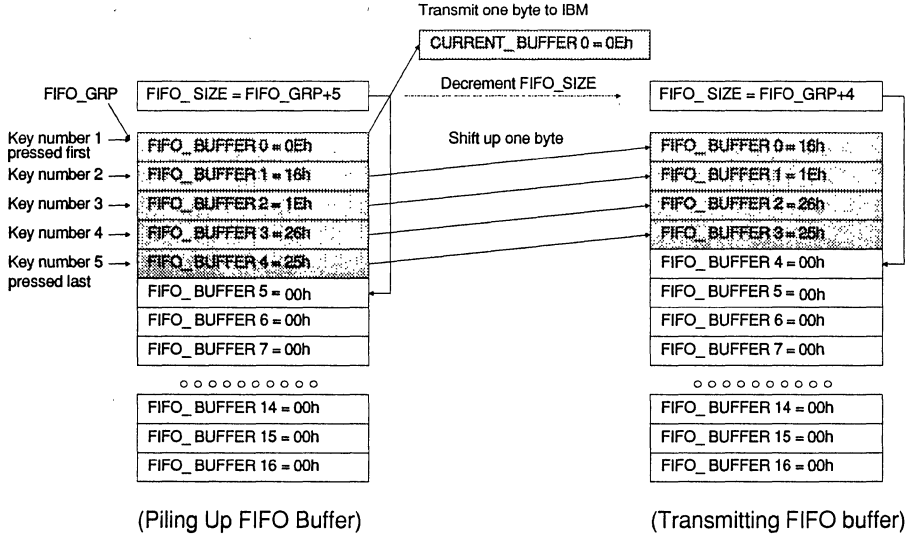


Figure 13. Stacking and Sending Status of Keyboard FIFO Buffer

To do the command communication, use the macro command com_gen . The com_gen macro contains three parameters which ensures the number of bytes handled after command reception, command data, and Jump address at the end of the communication (Figure 14).

Whenever DATA line goes low (from PC), the Z8602 receives the data by driving the CLOCK line eleven times. The received data is always checked whether or not it is a new command. The command compares to the Command data defined in com_gen macro.

If it is a new command, then it is stored in a Z8602 COMMAND register. The first parameter of com_gen saves to the COM_STATUS register (Figure 14). The COM_STATUS shows how many bytes are handled in the current command. The acknowledge data (0FAh) sets to COMMAND_BUFFER and departs in the next Timer 0 interrupt.

After that transmission, COM_STATUS decrements by one and tests for zero. If it is zero, the communication is over and the program executes by getting a jump address from the com_gen macro table.

If it is not zero, the communication remains active. The new command buffer sets to 0 unless the current command is READ_ID. The command buffer at 0 means data receive; non-zero means a data transmission to the PC. After the proper data transfer, the acknowledge data is sent. Now, each command is executable.

Basic Serial Data Input and Output Drivers

This module includes a parity generation for data transmission; 11 bits of data transmission with detection of line contention and 11 bits of data reception with an 11th bit acknowledge pulse.

Table 5 shows working registers specifying the initial values used to handle serial data transfers.

Table 5. Serial Data Transfer Working Registers

Register	Function
Serial Data Output	
CF(carry flag)	0 to set low start bit
serial_data_hi	bit7-1 = 1 and bit 0 = odd parity bit
serial_data_lo	8 bits data to be transmitted
bit_count	11 = number of bits transmitted
serial_bit	temporary register for P2 I/O port
transmit	0ffh to specify transmit mode
com_delay	to set up 80 usec/bit timing
P2	P2.1=CLOCK to make pulse, P2.0=DATA to transmit data
Serial Data Input	
CF (carry flag)	1 to set low high output for input mode
serial_data_hi	0ffh to receive 8 bits data
serial_data_lo	0ffh to receive 3 bits data
bit_count	11 = number of bits received
serial_bit	temporary register for P2 I/O port
transmit	0 to specify receive mode
com_delay	to set up 80 usec/bit timing
P2	P2.1=CLOCK to make pulse, P2.0=DATA to receive

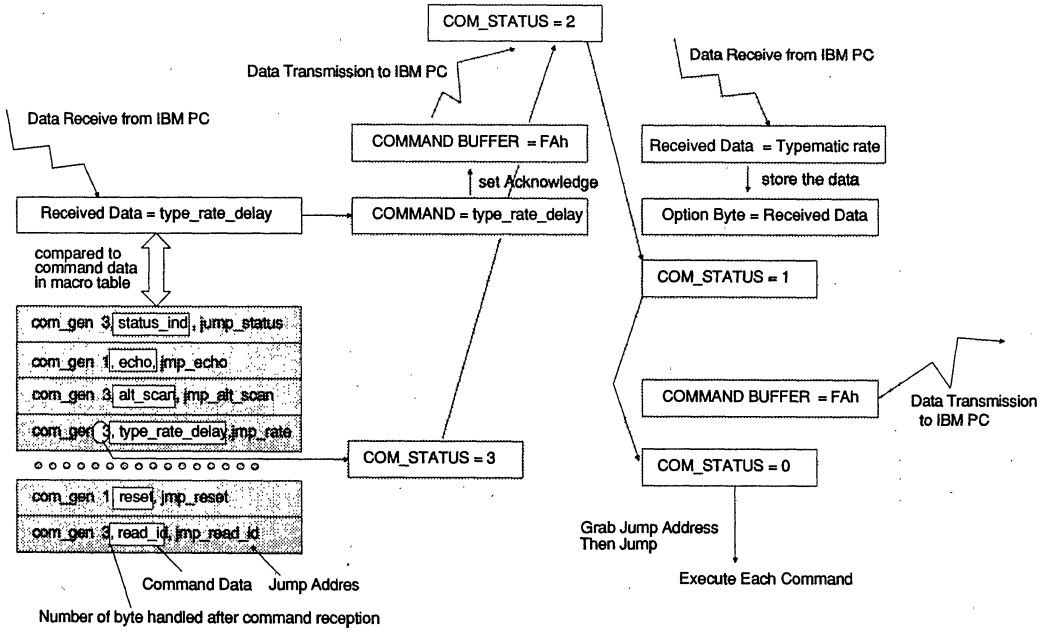


Figure 14. Register Manipulation in Command Communication

After receiving all 11 bits, the serial_data_hi and serial_data_lo shift right five times to set up 8 bits of data into serial_data_lo. Now, parity is in bit 0 of serial_data_hi and is checked by the subroutine of parity_gen (Figure 15).

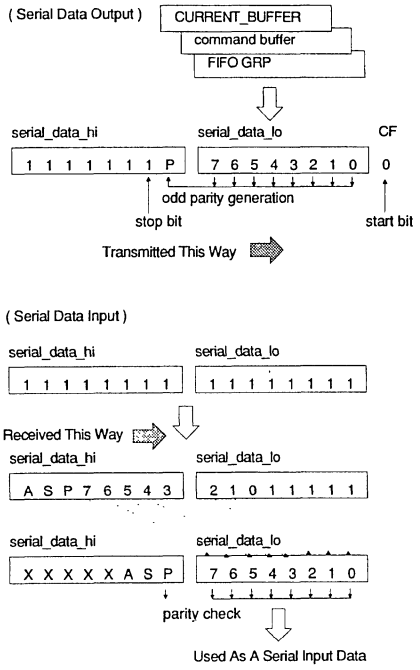


Figure 15. Register Manipulation of the Serial Data Buffer

The period of one data bit is 80 microseconds; the data must change when the CLOCK pulse is high. In fact, the switching is done 20 microseconds after the leading edge of the CLOCK pulse.

Transmit Mode:
In the send mode, line contention is always detected by the CLOCK line during a high level. If the CLOCK line goes low, the PC drives the line. If line contention appears before the 10th bit transmission, the system immediately quits the process and sets both DATA and CLOCK lines to high. If detection occurs after the 10th bit, sending continues until complete.

Receive Mode:
In the receive mode, the 10th bit tests for high. If high, the PC sends a low level for the 11th bit period to show an acknowledge. This means successful data reception from the PC. Otherwise, the PC sends multiple CLOCK pulses until it receives the correct stop bit.

KEYBOARD/Z8602 HARDWARE/SOFTWARE DETAILS

The following subsections explain and illustrate the Keyboard/Z8602 hardware details and program parameters. These involve multiple scan code working registers, FIFO dynamics, a scratchpad RAM map, and overall flow chart.

Multiple Scan Codes:
The multiple scan codes are stored into a 16-byte First-In-First-Out (FIFO) buffer until the PC is ready to receive them. A buffer-overrun condition occurs when more than 16 bytes remain in the FIFO buffer. This FIFO uses 16 general purpose registers in the Z86C02.

FIFO Dynamics:
The overrun code appears at the last position of the buffer. When the full 16 bytes of scan code are in the FIFO buffer and a further scan code appears, the overrun code goes into the 17th byte of the last occupied buffer register. This produces an audible "beep" warning. The FIFO buffer pointer points to the working register plus one. Therefore, If there are no scan codes available in the buffer, the pointer is the same address as the top of the FIFO buffer. The keyboard buffer only contains the scan codes and does not include any commands from the PC or acknowledges any data.

Scratchpad RAM Map

Table 6 describes the Z8602 RAM map. The table shows the function name, RAM address, bit position, bit name, and descriptions for all map functions.

PC/Keyboard Overall Flow Chart

The overall flow chart in Figure 16 shows three different diagrams involving the main program loop, keyboard scan and make/break code generation, and the PC/Keyboard communication. The following text explains the basic program flow

Table 6. Z8602 Scratchpad RAM Map

Name	R A M Address	Bit position	Bit Name	Function
P0	0			Port 0 Keyboard Column 8 - 15 Scan Output Ports
P1	1			Port 1 Keyboard Column 0 - 7 Scan Output Ports
P2	2	Bit0		Data line for serial data communication with IBM PC
		Bit1		Clock line for serial data communication with IBM PC
		Bit3		PC/XT or PC AT mode selection switch (high=PC AT mode)
		Bit2,4-7		Keybord Row 2, 4-7 Scan Input Ports
P3	3	Bit0,1,3		Keybord Row 0, 1,3 Scan Input Ports
		Bit4		Scroll Lock Indicator output (low=turn on LED)
		Bit5		Num Lock Indicator output (low=turn on LED)
		Bit6		Caps Lock Indicator output (low=turn on LED)
KEY_STATUS1	4	Bit0-2	bounce	3 Bit bounce counter (MAKE_CODE=3, BREAK_CODE=4)
		Bit3	RIGHT_SHIFT	Right Shift key is pressed when the key is detected
		Bit4	LEFT_SHIFT	Left Shift key is pressed when the key is detected
		Bit5	CTRL_CASE	Control key is pressed when the key is detected
		Bit6	ALT_CASE	Alternate key is pressed when the key is detected
		Bit7	NUM_LOCK2	Num Lock status is kept in this bit when the key is detected
KEY_DATA1	5	Bit0-7		1st Key Number in the schematic is kept when a key is detected
KEY_STATUS2	6			Same kind of data as KEY_STATUS1
KEY_DATA2	7			2nd Key Number in the schematic is kept when a key is detected
KEY_STATUS3	8			Same kind of data as KEY_STATUS1
KEY_DATA3	9			3rd Key Number in the schematic is kept when a key is detected
KEY_STATUS4	Ah			Same kind of data as KEY_STATUS1
KEY_DATA4	Bh			4th Key Number in the schematic is kept iwhen a key is detected
KEY_STATUS5	Ch			Same kind of data as KEY_STATUS1
KEY_DATA5	Dh			5th Key Number in the schematic is kept when a key is detected
KEY_STATUS6	Eh			Same kind of data as KEY_STATUS1
KEY_DATA6	Fh			6th Key Number in the schematic is kept when a key is detected
WORK_GRP	10h-17h			WORK_GRP working registers
TYPEMATIC_RATE	19h	Bit0-4	RATE_BITS	Typematic rate bits received from IBM PC
		Bit5-6	DELAY_BITS	Typematic delay bits received from IBM PC
SCAN_CODE_SET	1Ah	Bit0-1		Current Scan Code Set (1=scan code set 1, 2=scs 2, 3=scs 3)
LOCK_STATUS	1Bh	Bit0	SCROLL_LOCK	Current Scroll Lock status (1=scroll lock)
		Bit1	NUM_LOCK	Current Num Lock status (1=Num lock)
		Bit2	CAPS_LOCK	Current Caps Lock status (1=Caps lock)
		Bit3-4	SHIFT_CASE	Current right and left shift status (1=pressed)
		Bit5	CTRL_CASE	Current Control key status (1=pressed)
		Bit6	ALT_CASE	Current Alternate key status (1=pressed)
		Bit7	MAKE_CASE	Temporary flag to inform either Make(=1) or Break code generation
DELAY	1Ch			Delay timer (60=250 ms, 120=500ms, 180=750ms, 240=1000ms)
RATE	1Dh			Typematic rate timer (30.0/sec to 2.0/sec)
TYPEMATIC	1Eh			Current typematic key pointer to address one of KEY DATA (0=no typematic key , KEY_DATA1, 2,3,4,5,6)
FIFO_SIZE	1Fh			FIFO Buffer size (if no buffer valid, FIFO_SIZE=FIFO_GRP)
COM_GRP	20h-27h			COM_GRP working registers
COM_STATUS	28h	Bit0-6		Communication status (0=no communication, 1 to X)
		Bit7	resend flag	Resend flag (=1)
COMMAND	29h			Current command received from IBM PC
OPTION_BYTE	2Ah			Current option byte received fro IBM PC
COMMAND_BUFFER	2Bh			Communication data includes Acknowledge byte, transmitting data. (if it is 0, then receive mode)
KEY_TYPE	2Ch	Bit0-1		Current key type (make_type_break, make_type, make_break, make)
CURRENT_BUFFER	2Dh			Curren Communication output buffer
MAIN_CONTROL	2Eh	Bit7	enable_scan	Keyboard enable scan flag
FIFO_GRP	2Fh-3Fh		17 bytes buffer	16 byte Keyboard FIFO buffer + 1 byte overrun data buffer
CODE3_GRP	40h-5Fh		32 bytes attribute	Scan Code Set 3 Attribute Data (2bits attribute x 128 keys, scan code=minimum 07, maximum 86h)
Stack Area	60h-7Fh		32 byte	32 byte Stack Area

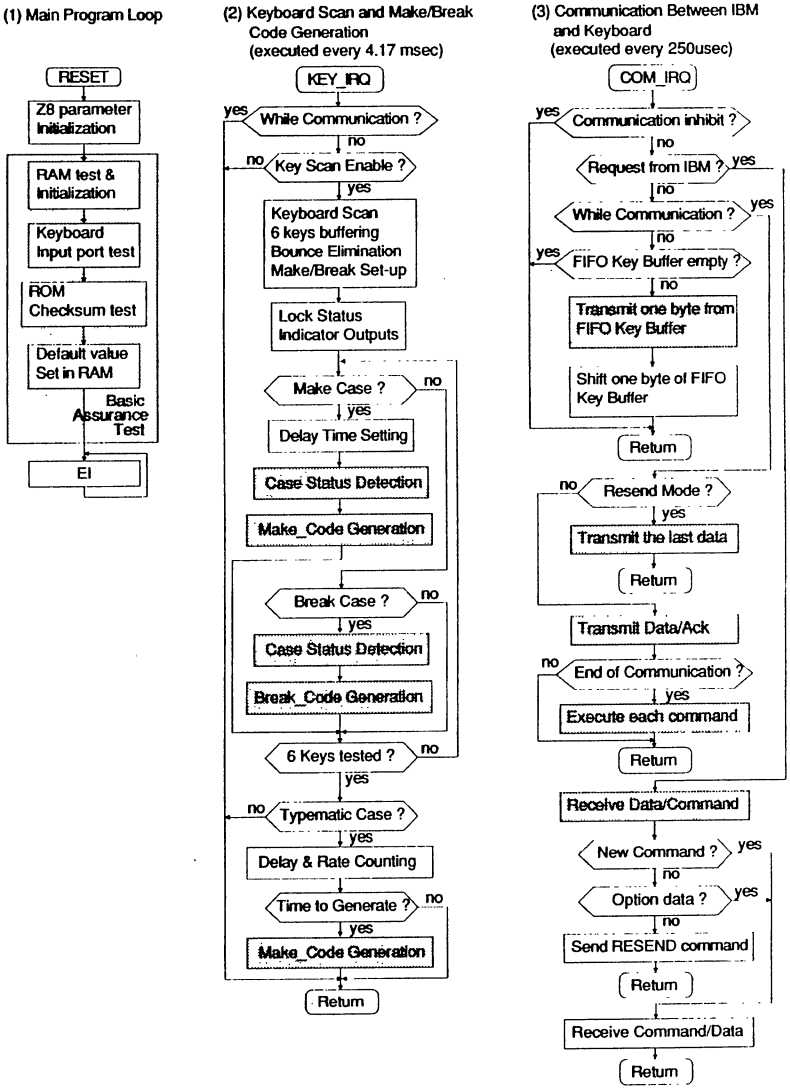


Figure 16. PC/Keyboard Program Flow Chart

Main Program Loop:
Upon reset, a Basic Assurance Test on the RAM, Keyboard, and ROM is enabled. The Z8602 parameters start, and test data transfers between the Z8602's CPU and the RAM, Keyboard and ROM. If no error detection, RAM receives the default values and the initialization testing is complete.

Keyboard Scan and Make/Break Code Generation:
This program executes every 4.17 milliseconds. If there is a current communication, the flow bypasses and returns to the main program. If there is no current communication and the Key Scan (KEY_IRQ) is enabled, the main process for six key buffering bounce elimination in make or break setup, goes active.

Now, Lock Status indicators are monitored and then make/break cases examined. A decision determines which case is active. The active case tests for six keys. If it is the make case and is Typematic, the Make Code Generation parameters execute. If it is a break case, the flow returns to the main program.

PC/Keyboard Communication:
Communication between the PC and the keyboard executes every 250 microseconds. When there is an active COM_IRQ but a positive Communication Inhibit, the flow returns to the main program. If there is an active COM_IRQ and no Communication Inhibit and there is a request from the PC, the Receive Data/Command activates. Depending upon whether it is a new command, optional data, or send Resend, the pertinent logic goes active and the flow returns to the main program.

When there is no request from the PC, but there is current communication, the flow jumps to the Resend Mode. If the Resend mode is active, the last data is sent and then flow returns to the main program. If the Resend mode is inactive, the flow jumps to Transmit Data/Ack. If there is no more communication, the flow returns to the main program. If there is more communication, the remaining commands execute and the flow returns to the main program.

When there is a COM_IRQ but no communication inhibit or request from the PC, and not during communication, the FIFO Key Buffer is checked. If it is empty, the flow returns to the main program. If not empty, one byte is sent from the FIFO Key Buffer and one byte shifts. The flow then returns to the main program.

Three Facets of a Many Faceted Microcontroller

If you need D/A conversion, or a zero crossing detector, or a current sensing device... Use the Z86C08's dual comparator.

Dual Analog Comparator

Using the dual analog comparators on the Z86C08 in conjunction with several on-chip features, provides a cost effective way to monitor power failures and frequency excursions (comparator used as a zero crossing detector), as a blood pressure tester and digital readout (comparator used as a A/D converter), or as a current sensing device in automotive design to detect and subsequently shutoff any short circuiting of relays, lights, monitors, etc.

In many microcontroller applications, the digital designer is often concerned with sampling and controlling non-digital elements within his system. However, when the designer is forced to deviate from the precise world of TTL logic and regulated 5 volt supplies, frequently, microcontroller architectures and specifications fall short in the areas of cost sensitivity and consumer orientation. Therefore, using the analog comparators in these specific areas are a few of the reliable, inexpensive design applications for the Z86C08.

Comparator Basics

The dual comparators share a common inverting terminal with non-inverting terminals bonded directly to external I/O ports (Figure 1). The comparators are enabled by a bit in the I/O port mode/control register. If bit D1 of R247 is zero, then the comparators are in digital mode. If D1 is one, then they are in analog mode. With the comparators disabled, the I/O ports are available for normal activities. These particular I/O ports can be used to generate external interrupt requests to the Z8. With the comparators enabled, interrupts can also be generated.

The ideal comparator is a three terminal device (Figure 2). V1 is a non-inverting terminal. Signals entering at V2, the inverting terminal, exit Vout 180o out of phase. Since a comparator is essentially an operational amplifier, it has an associated gain. The open loop gain (no feedback) of a comparator is defined as the Voltage Out (Vout) over the Differential Input Voltage. The Differential Input Voltage is the voltage at the non-inverting input with respect to the inverting input. Thus, gain is:

GAIN = Vout/V1 - (V2) = Voltage Out/Differential Input Volt

The Input Offset Voltage, the difference between V1 and V2, forces Vout to a specified level. The Input Offset Voltage is typically below 50 mV.

Zero Crossing Detect Applications

The dual comparator can be used as a zero crossing detector to monitor 110 VAC (or other power line parameters) and its frequency (Figure 3). Each time the voltage passes through zero an interrupt is generated. The outputs of the comparators on the Z86C08 connect directly to the on-chip CPU. When using the comparators to detect zero crossing of a signal, interrupts are generated at every crossing. Interrupt subroutines can then calculate period and phase angle relationships between any two analog signals. The phase angle being critical when calculating power factor in power line circuits.

In the case of 110 VAC, 60 Hz power line, an interrupt is generated every 1/120 of a second. This means that whenever the monitor stops (no interrupts), there is a power fail or other problem which can be translated by a control device for quick recovery action (Figure 4).

Frequency checks can also be made by zero crossing detection. Whenever frequency drifts from the normal monitoring zero points, interrupts are either increased (higher frequency) or decreased (lower frequency) from the norm. If necessary, appropriate action is then taken.

Another application is threshold detection for low voltage battery operated devices . Whenever the VBB drops below the Zener reference voltage level, an interrupt is generated to alert a control device or alarm.

The addition of two on-chip counter/timers further complement the above mentioned applications. Crystal precision timing is done on the period of zero crossings. The sum or difference of two separate analog signals then can be calculated. For example, negative or positive feedback is returned from the Z86C08 in closed loop calculations. In power circuits, a time-of-day clock could be implemented with a timer. Then, date and time of power failures and frequency excursions can be recorded. CMOS technology allows for battery backup.

Analog to Digital (A/D) Conversion

Accurate low speed A/D conversion is implemented with the Z86C08 using the dual slope or ratiometric method. With this method, a dv/dt is applied to the inverting terminal of a comparator. The analog input (Vinput) signal is applied to the non-inverting terminal. The charge rate of the RC circuit is a dv/dt (Figure 5). As Vref ramps upward from zero volts during time T1, Vref will exceed Vinput. This causes the comparator to

change state and produce an interrupt. By using the on-chip timer, time T1 can be quickly determined.
The RC circuit is immediately discharged over fixed time T2 (Figure 6), where T2 is determined by the time constant Tc = RCn. Since the product of RC is only an approximate indicator of discharge time, a value of n should be multiplied to improve accuracy. A general guideline should equate n to 1.4. Then, T2 = 1.4 RC.
The dual slope A/D converter measures voltage by converting voltage into time intervals. Or,

T2/T1 = Vinput/Vref, then, Vinput = Vref T2/T1

By using an I/O port on the Z86C08 as the Vref input, interrupts generated by the comparators can alternately switch Vref ON or OFF to perform the conversions.

Example: Blood Pressure Tester
A pressure transducer in a blood pressure tester is a good example of the dual slope A/D conversion method. A minimum system consists of display logic, Z86C08 circuitry and a transducer signal input (Figure 7). P00 outputs the appropriate signal to the RC ramp circuit of the Vref input. The output from the pressure transducer (Figure 8) is a linear voltage response to the applied pressure. This signal is input to An2, the non-inverting terminal of the comparator.
In this configuration, the sampling cycle for the A/D conversion begins when a logical 1 is output on P00 and a timer is enabled. When the comparator transitions, an interrupt is generated, the timer is stopped and P00 is toggled to discharge the RC circuit. By storing the count T1 and resetting the timer, the converter is now ready to take another sample. The value of Vin is mathematically determined later and software algorithms are used to determine corresponding pressure.
The display is driven from a simple multiplexer circuit. The Z86C08 can sink large Ioc currents which reduces or eliminates buffering.

Current Sensing

The dual comparator is used as a current sensing device in many application areas, e.g., in automotive relays, lights, monitors, etc. In the automotive arena, current sensing is used in a typical case as shown in Figure 9. If the functional block shorts, then current (I) surges causing voltage (V) to fall. When V reaches 2.5V, the comparator triggers an interrupt which allows software to enable an emergency shutoff.

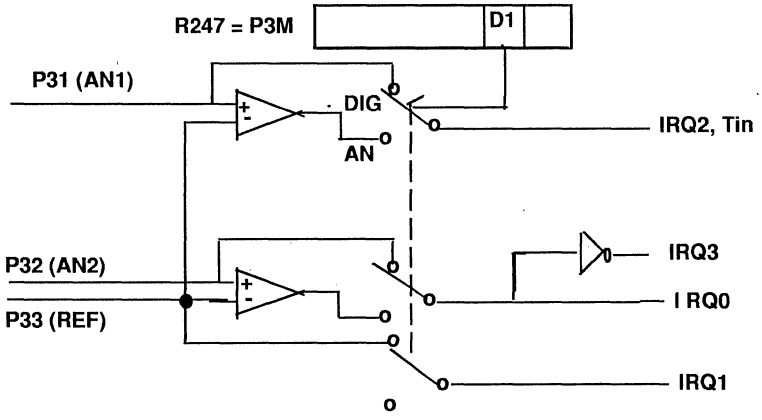


Figure 1. Dual Analog Comparator

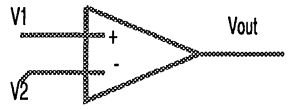


Figure 2. Ideal Comparator

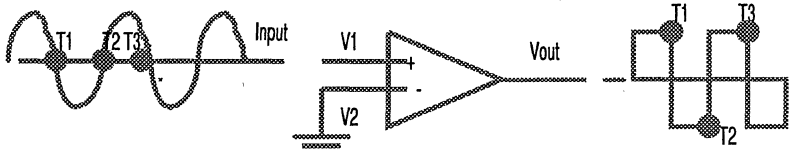


Figure 3. Zero Crossing Detector

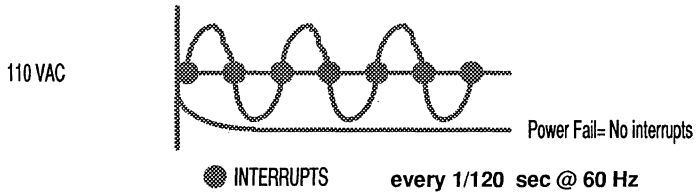


Figure 4. Interrupt After Power Failure

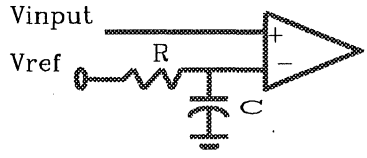


Figure 5. A/D Converter

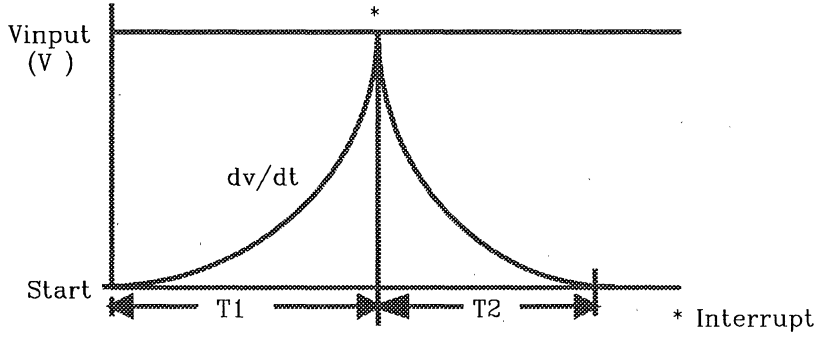


Figure 6. Voltage vs. Time

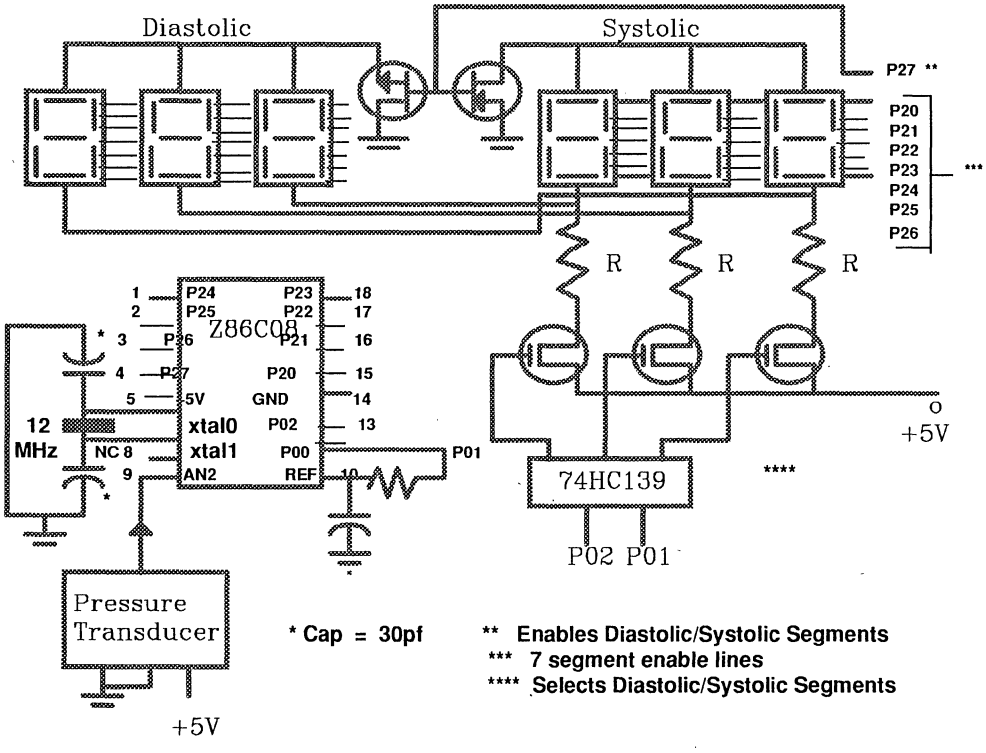


Figure 7. A/DBlood Pressure Test and Readout

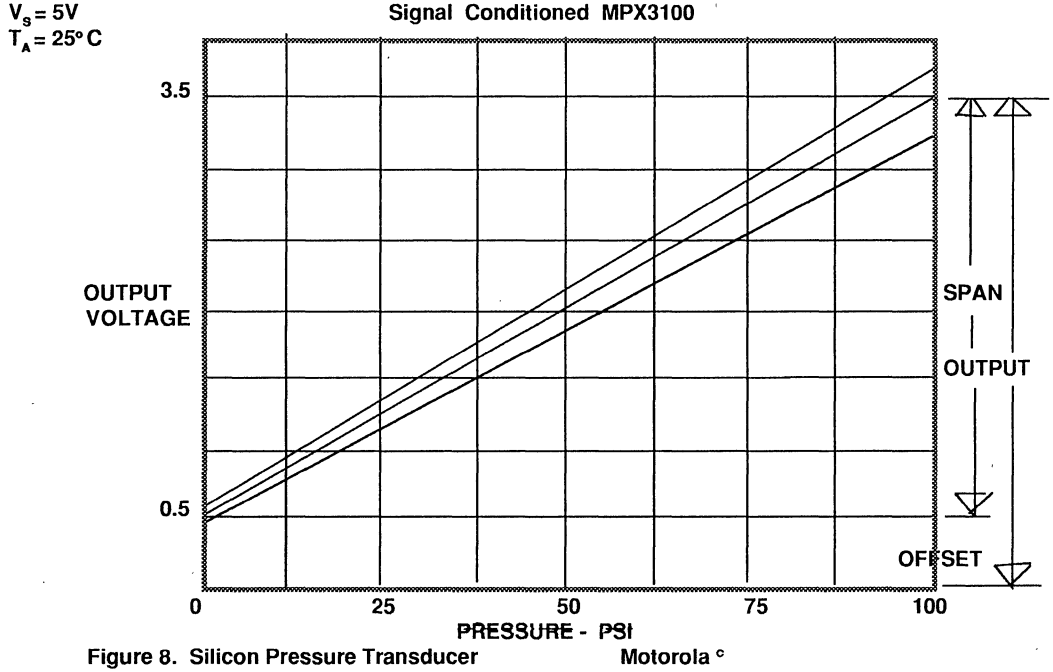


Figure 8. Silicon Pressure Transducer

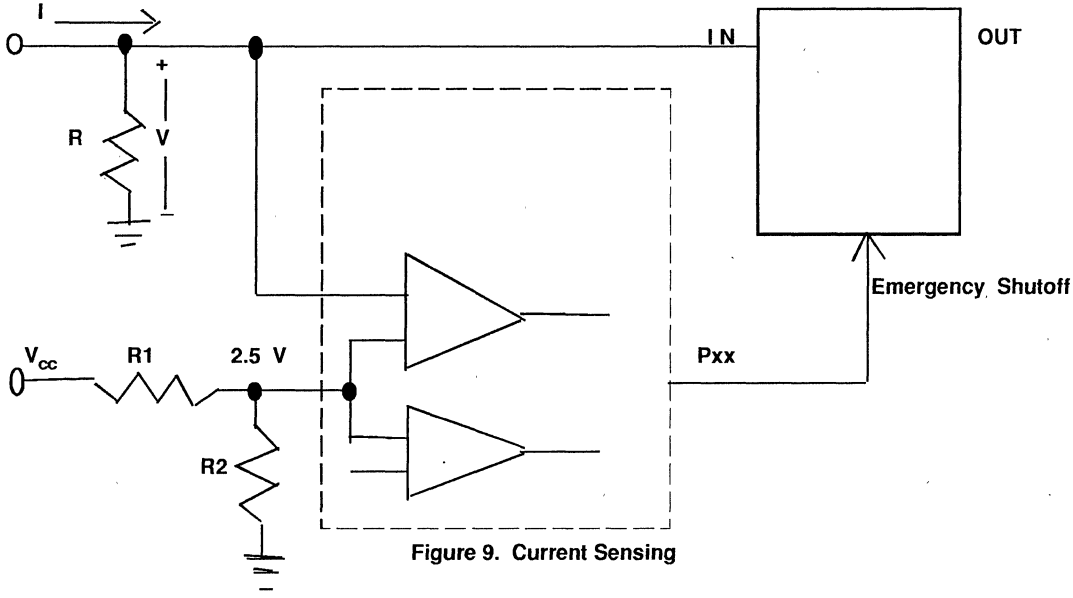
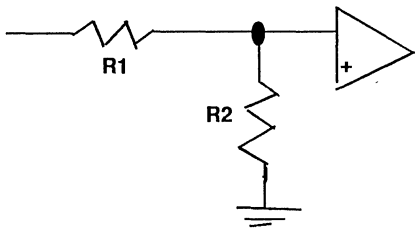


Figure 9. Current Sensing

R is large compared to the equivalent impedance of the Functional Block input. R1 and R2 are user selectable and are generally in a 10K to 100K range for power dissipation considerations. R1 and R2 are determined from the following formula:

$$V_{ref} \cdot \frac{R2}{R1 + R2}$$



The Z8 MCU in Telephone Answering Systems

There are telephone and answering machine combinations and answering machines only. This App Note deals with answering machines.

Basic Functions

The following subsections describe the way messages are handled, playback features, ring control, remote operation and other features.

Out-Going Message

Most high-end machines do not use a cassette tape for the out-going message (called OGM). Out-going messages (up to 16 seconds duration) are digitized and stored in RAM, and played back on demand.

Incoming Message

Incoming messages are recorded on a cassette. The length of the message can be either unlimited (recording continues as long as the caller is speaking), or limited (exactly 30 seconds, for example).

Playback Features

Typical playback features are:

- One-touch playback.
- A display showing the number of calls received.
- Dial tone elimination (after the caller hangs-up, the tape rewinds to eliminate the dial tone).
- Playback automatically stops after the last message.

Ring Control

Answering systems can be configured to respond after 2 or 4 rings. Some machines can be programmed to respond after between 1 and 7 rings.

Remote Operation

The owner of an answering machine can access the machine from another telephone to playback messages. This is done by dialing the telephone number, followed by a secret code. Rewind, fast-forward, system on, changing the OGM, and other functions can also be carried out thru remote control.

Other Features

Some machines allow phone conversations to be recorded and also personal messages to be recorded. Several machines can be configured in the Toll Saver mode. In this mode, the machine responds after 2 rings only if messages have been left. If no messages have been left, the machine responds after 4 rings.
Voice Menu is a feature which uses speech synthesis circuitry to generate speech. This replaces displays, beeps and tones found in lower-end systems.

Operation of a Telephone Answering System

Figure 1 shows the entire block diagram of an answering system. This is divided into four major blocks. The four blocks are:

- Ring Detection and Attenuation block.
- Mux/De-Mux block.
- User Interface block (display, keypad, and mode selection switches).
- Remote Control Mechanism

The Ring Detection and Attenuation block.

The telephone line (out of the wall socket) plugs into the PHONE LINE IN jack (Figure 2). The telephone plugs into the TO PHONE jack, and is essentially in parallel with the answering machine. The phone ring is caused by a large AC voltage across the PHONE LINE IN jack. This is passed thru the ring detector circuitry which is comprised of a diode rectifier and transformer followed by an analog comparator. The output of the ring detector is a square wave as shown in Figure 2. Each burst is one ring of the telephone. These are monitored by the microcontroller.
After a preset number of rings (usually 2 or 4) the microcontroller closes the relay. This simulates the telephone off-hook condition, and the circuit is closed.

The Mux/De-Mux block

This is the main section of an answering system (Figure 3). After the phone rings and the relay is closed, the

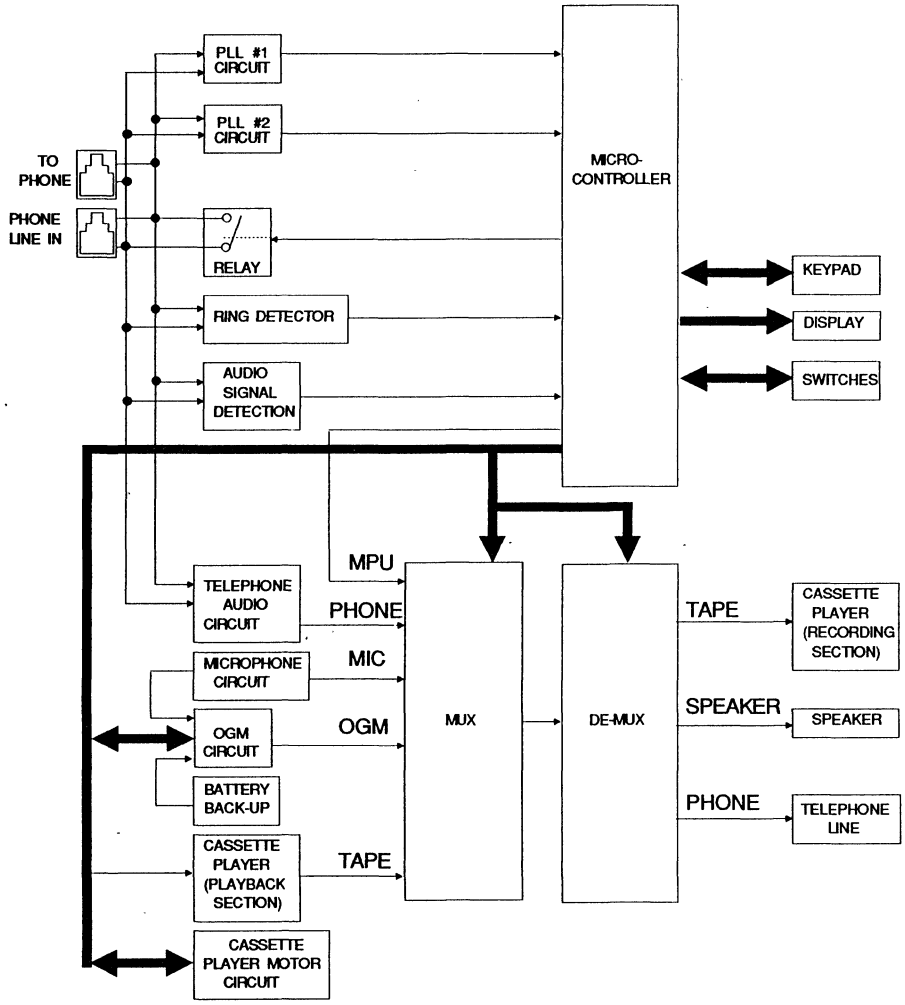


Figure 1. Answering System Block Diagram

caller hears the Out Going Message. In other words, the microcontroller selects the OGM line as the MUX input. The MUX output is fed to the DE-MUX block. The microcontroller selects the SPEAKER and PHONE lines as the outputs of the DE-MUX block. In effect, the caller hears the OGM and it is also heard from the speaker of the answering machine.

When the OGM is over, the caller hears a tone. This tone (usually 1 KHz) is generated by the microcontroller. At this point, the MPU line is selected as the MUX input, while the DE-MUX outputs are the same as before. Following the tone, the PHONE is selected as the MUX input, and the caller can leave a message. The TAPE (which records the message) and SPEAKER lines are now selected by the microcontroller as the outputs of the DE-MUX block. When the caller hangs up, the relay is opened and the outputs of the DE-MUX block are not connected to the input.

For message playback, the TAPE line is selected as the MUX input and the SPEAKER line as the DE-MUX output. A common feature of several answering machines is the Personal Memo function. The speaker speaks into the microphone and the machine acts as a cassette recorder. The message is recorded on the cassette. In this case, the input to the MUX is the MIC line, and the output of the DE-MUX is the TAPE line.

Figure 4 shows the Audio Signal Detection block. This block is comprised of analog circuitry which detects the presence of an audio signal. When a caller is leaving a message on the answering machine, and pauses for a long period of time, the microcontroller automatically stops recording and goes on-hook.

A typical analog signal is shown in Figure 4. This is the input to the audio signal detection circuitry. The output of this circuit is a square wave. If it is continuously low for a preset length of time (4 seconds, for example), the answering machine goes on-hook.

The User Interface block

The user interface block diagram is comprised of the display, keypad, and mode selection switches (Figure 5). The keypad usually has 5 or 6 keys for functions such as Playback, Rewind and Stop. The keypad is scanned periodically by the microcontroller. The display (usually of the 7 segment LED type) shows the number of calls received.

In most machines, the microcontroller updates a 7 segment display driver/decoder/latch IC which drives the LED display.

The selection switches are used to determine the number of rings before the answering machine responds, the maximum length of the message and other features. These switches are scanned when the phone rings, and the microcontroller uses this data to execute the appropriate sections of code.

The Remote Control Mechanism.

The remote control mechanism allows control of the answering machine from another telephone. The caller dials the telephone number and when the answering machine responds, the caller keys in a secret code and from that point on, the answering machine can be controlled by the keys on the caller's phone. The caller must use a DTMF (TouchTone) phone. If a rotary dial (pulse) phone is used, a tone generator is required for remote control.

Most answering systems use tone decoder (ie. phase locked loop) ICs for the remote control function. The phone line acts as the inputs to the PLLs (Figure 6). As soon as the answering machine responds by going off-hook, the PLL outputs are scanned periodically by the microcontroller. Each PLL has a center frequency corresponding to 1 of the 7 tones required for DTMF signals. The 7 frequencies are 697, 770, 852, 941,

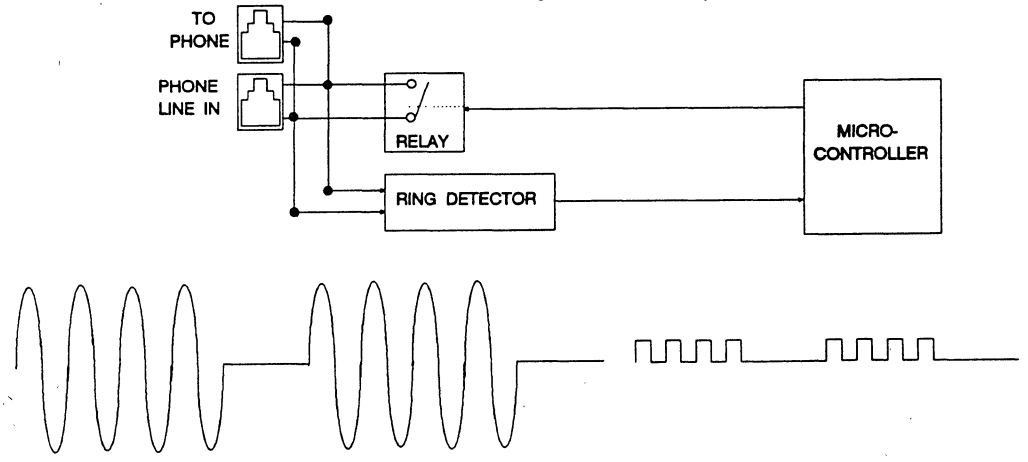


Figure 2. Ring Detector Block

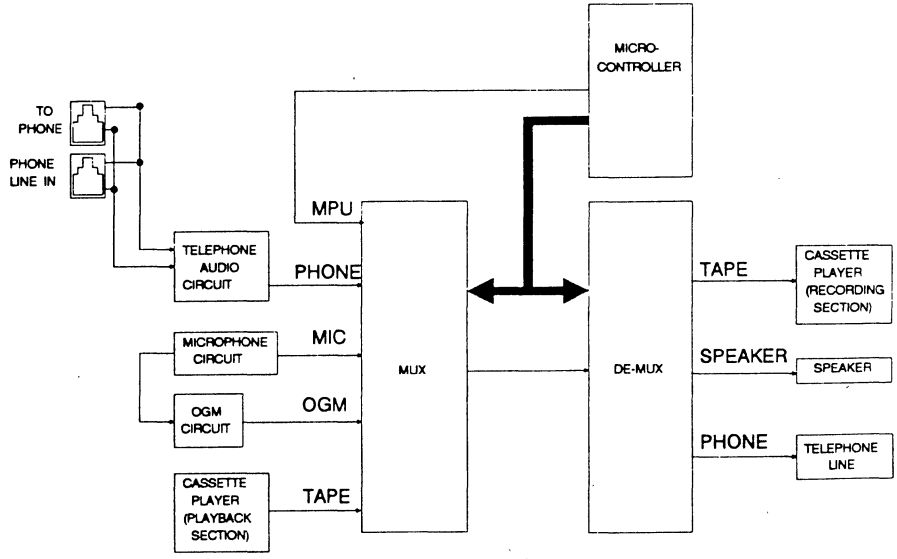


Figure 3. Mux/De-Mux Block

1209, 1336, and 1477 Hz.
The PLLs have very narrow detection bands since the 7 tones are within a few hertz of one another. If a key is pressed at the remote telephone, and either of the two tones of the DTMF signal match the center frequency of a PLL, the output of that PLL will drive a load (i.e., go low). This is detected by an input line of the microcontroller. If the outputs of several PLLs go low in the correct sequence (corresponding to the correct secret code), the microcontroller allows subsequent keys to control functions such as Playback, Rewind and Stop.

Miscellaneous Circuitry

The microcontroller drives a motor controller which drives the cassette player motor. The basic signals to the motor consist of speed and direction information. The microcontroller receives a signal from the motor position encoder circuitry. This signal is used to keep track of the length of the tape that has elapsed.

Microcontroller Suitability for Answering Systems

The Intel 8049 (40-pin HMOS, 2K ROM) is a microcontroller often used in answering systems. A Zilog 40-pin NMOS Z8 MCU would be an excellent replacement for

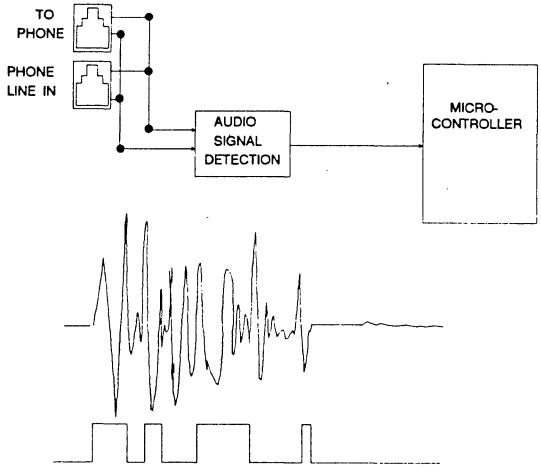


Figure 4. Audio Signal Detection Block

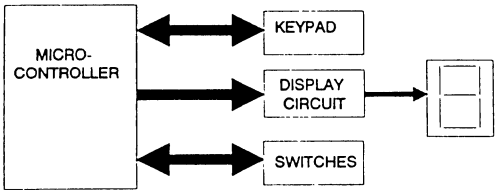


Figure 5. User Interface Block

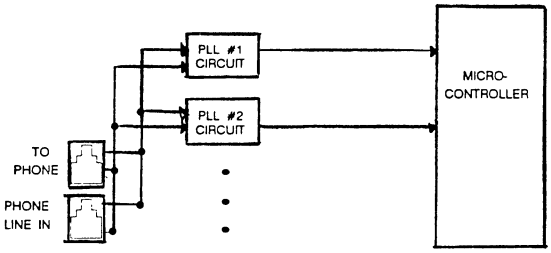


Figure 6. Remote Control Block

the 8049. The block diagrams for functionally identical systems using an 8049 and a 40-pin Z8 microcontroller are shown in Figure 7.

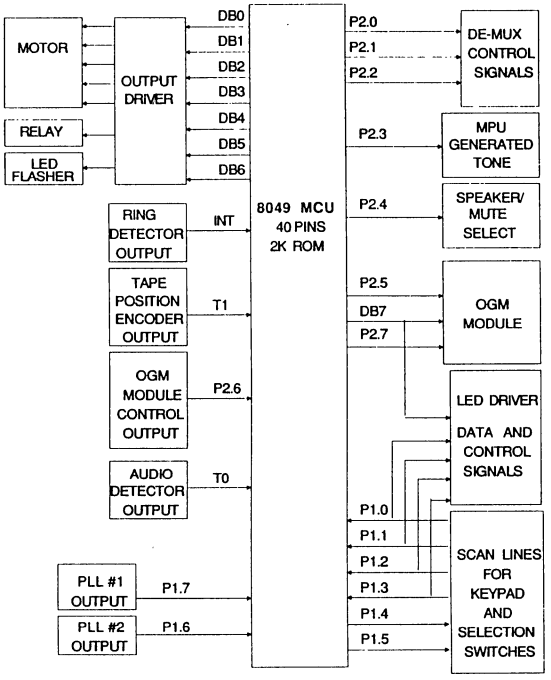
This Z8 part comes in 2K and 4K ROM versions, whereas the 8049 is available with 2K ROM only. CMOS versions of the 40- pin Z8 are available with 8K ROM.

Ease of programming in Z8 assembly language is a major advantage, especially since the JUMP instruction is not limited to only locations within the current page. The 40- pin Z8 offers 2 counter/timers and 144 bytes of on-chip RAM, whereas the 8049 offers only 1 counter/ timer and 128 bytes of on-chip RAM. Further, the 40-pin Z8 has 5 more I/O lines (32 in all) than the 8049 (27 I/ O lines). Availability of a 28-pin Z8 and a 64-pin Z8 allow future simplification/upgrade options for systems based on the 40- pin Z8.

Orion Emulatoranalyzer

The Z8 In-Circuit Emulator target board and an XT/AT compatible computer provide an excellent development system at a reasonable price. A full bus analyzer and indepth debug facility allows efficient software development.

The Z8 pin functionality for an answering machine application is shown in Figure 8.



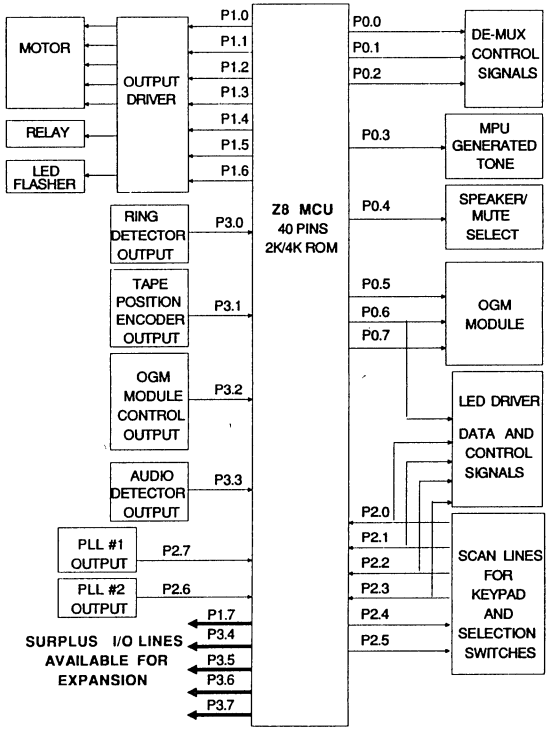


Figure 7. Intel 8049 MCU vs. Zilog Z8 MCU

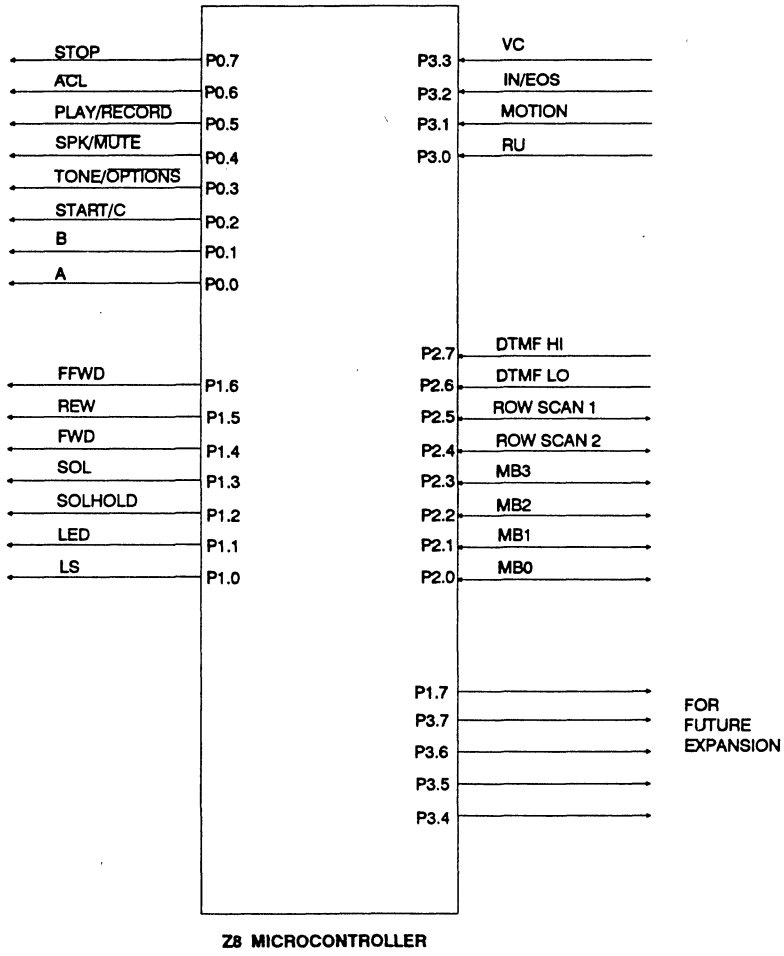


Figure 8. Z8 Pin Functions for Answering Machines

Software Implementation

Refer to the listing of the Z8 applications software for a medium-end answering machine. The Z8 port functions are shown in Figure 9 and the registers referred to as FLAG, TAPE_FLAG, and REMOTE_FLAG are shown in Figure 10.

The remaining illustrations are block flow diagrams showing the various functional modes.

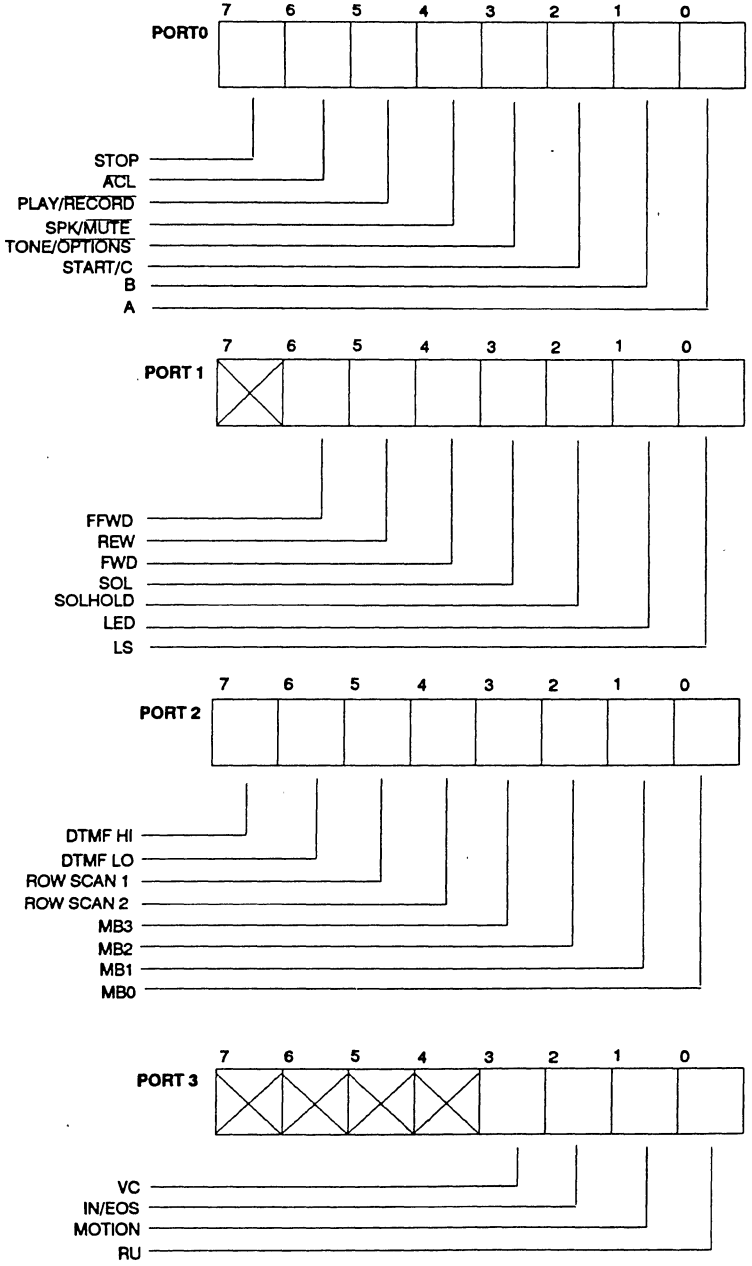


Figure 9. Z8 Port Functions

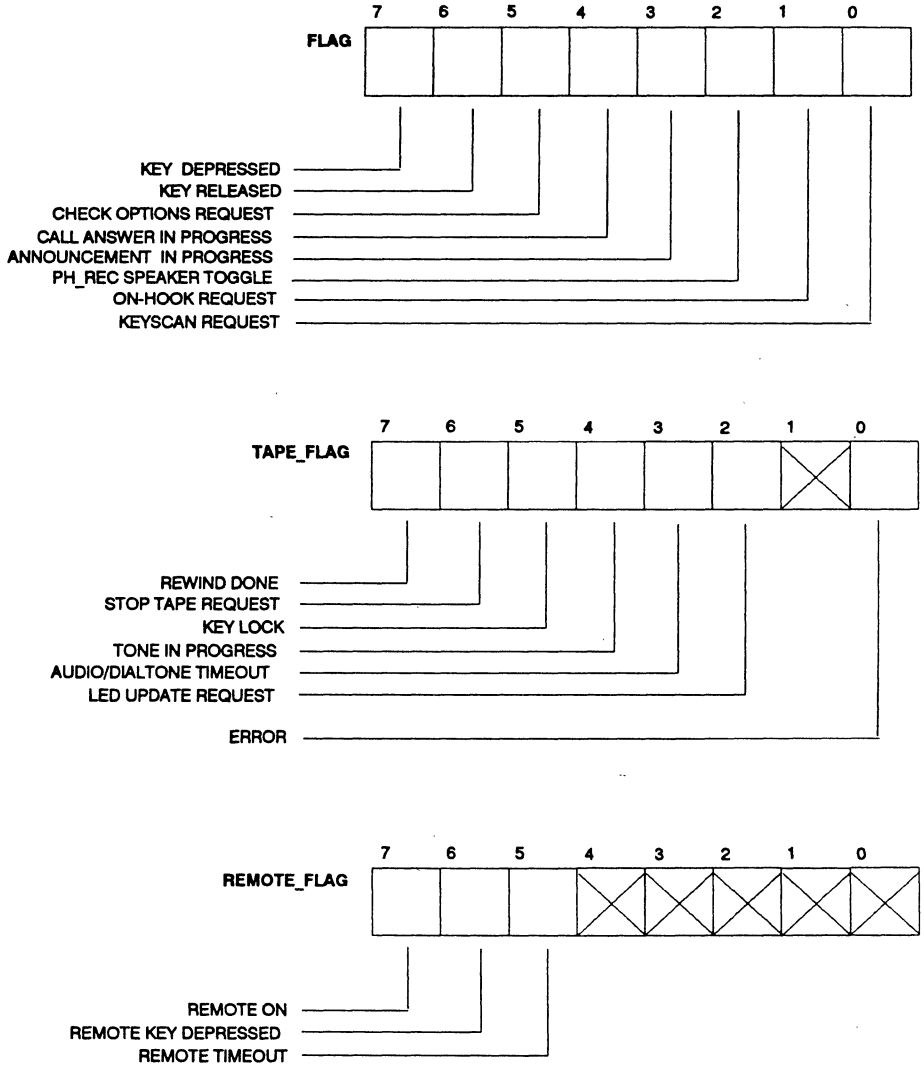


Figure 10. Flag, Tape-Flag and Remote Flag Registers

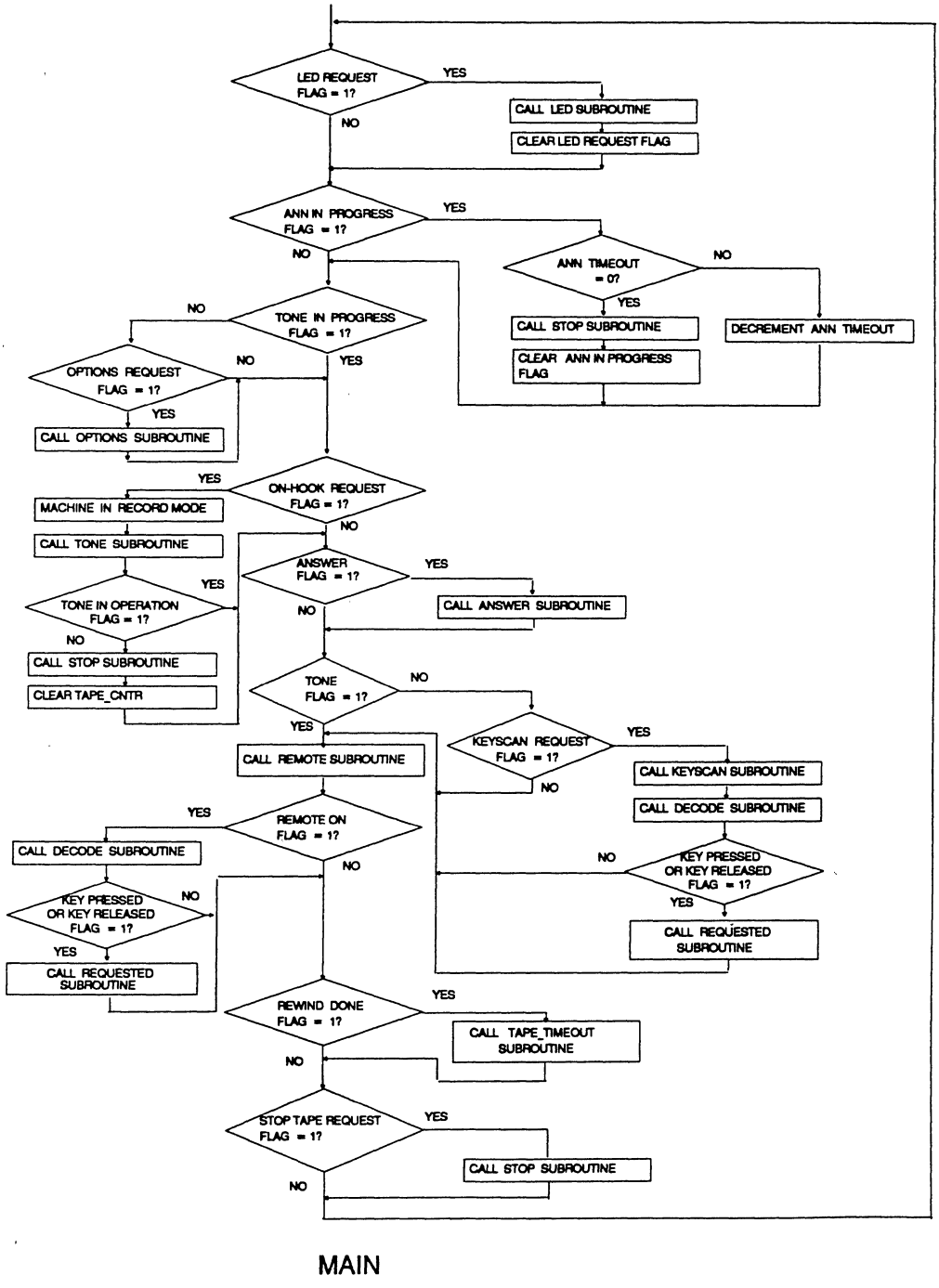


Figure 11. Answering Machine Main Block Flow

Z8® Subroutine Library

ZiLOG

Application Note

INTRODUCTION

This application note describes a preprogrammed Z8601 MCU that contains a bootstrap to external program memory and a collection of general-purpose subroutines. Routines in this application note can be implemented with a Z8 Protopack and a 2716 EPROM programmed with the bootstrap and subroutine library.

In a system, the user's software resides in external memory beginning at hexidecimal address 0800. This software can use any of the subroutines in the library wherever appropriate for a given application. This application example makes certain assumptions about the environment; the reader should exercise caution when copying these programs for other cases.

Following RESET, software within the subroutine library is executed to initialize the control registers (Table 1). The control register selections can be subsequently modified by the user's program (for example, to use only 12 bits of Ports 0 and 1 for addressing external memory). Following control register initialization, an EI

Table 1. Control Register Initialization

Control Register Name	Address	Initial Value	Meaning
TMR	F1H	00H	T0 and T1 disabled
P2M	F6H	FFH	$P2_0$-$P2_7$: inputs
P3M	F7H	10H	P2 pull-ups open drain; $P3_0$-$P3_3$: inputs; $P3_5$-$P3_7$: outputs; $P3_4$: DM
P01M	F8H	D7H	$P1_0$-$P1_7$: AD_0-AD_7; $P0_0$-$P0_7$: A_8-A_{15}; normal memory timing; internal stack
IRQ	FAH	00H	no interrupt requests
IMR	FBH	00H	no interrupts enabled
RP	FDH	00H	working register file 00H-0FH
SPL	FFH	65H	1st byte of stack is register 64H

instruction is executed to enable interrupt processing, and a jump instruction is executed to transfer control to the user's program at location 0812_H. The interrupt vectors for IRQ_0 through IRQ_5 are rerouted to locations 0800_H through $080F_H$, respectively, in three-byte increments, allowing enough room for a jump instruction to the appropriate interrupt service routine. That is, IRQ_0 is routed to location 0800_H, IRQ_1 to 0803_H, IRQ_2 to 0806_H, IRQ_3 to 0809_H, IRQ_4 to $080C_H$, and IRQ_5 to $080F_H$. Figure 1 illustrates the allocation of Z8 memory as defined by this application note.

The subroutines available to the user are referenced by a jump table beginning at location 001BH. Entry to a subroutine is made via the jump table. The 32 subroutines provided in the library are grouped into six functional classifications. These classifications are described below, each with a brief overview of the functions provided by each category. Table 2 defines one set of entry addresses for each subroutine in the library.

- Binary Arithmetic: Multiplication and division of unsigned 8- and 16-bit quantities.

- BCD Arithmetic: Addition and subtraction of variable-precision floating-point BCD values.

- Conversion Algorithms: BCD to and from decimal ASCII, binary to and from decimal ASCII, binary to and from hex ASCII.

- Bit Manipulations: Packs selected bits into the low-order bits of a byte, and optionally uses the result as an index into a jump table.

- Serial I/O: Inputs bytes under vectored interrupt control, outputs bytes under polled interrupt control. Options provided include:
  - odd or even parity
  - BREAK detection
  - echo
  - input editing (backspace, delete)
  - auto line feed

- Timer/Counter: Maintains a time-of-day clock with a variable number of ticks per second, generates an interrupt after a specified delay, generates variable width, variable frequency pulse output.

The listings in the "Canned Subroutine Library" provide a specification block prior to each subroutine, explain the subroutine's purpose, lists the input and output parameters, and gives pertinent notes concerning the subroutines. The following notes provide additional information on data formats and algorithms used by the subroutines.

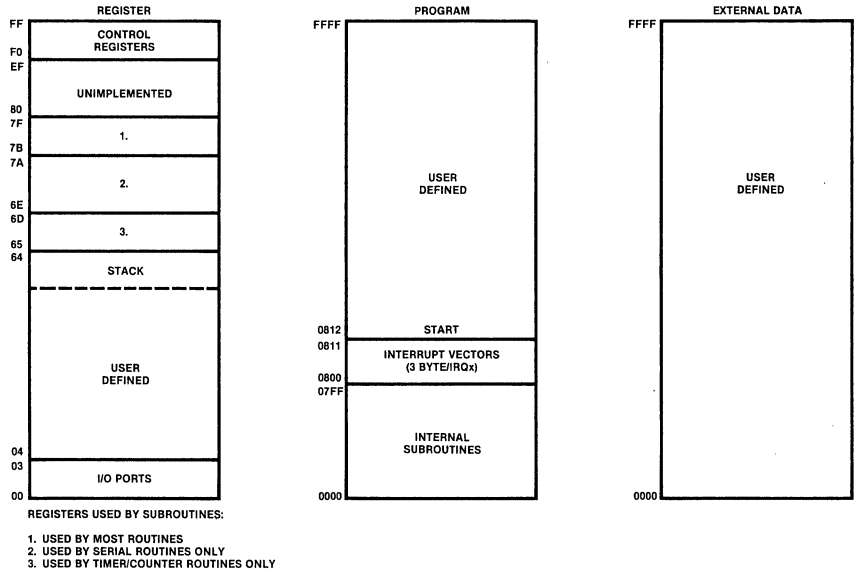


Figure 1. "ROMless Z8" Subroutine Library Memory Usage Map

1. Although the user is free to modify the conditions selected in the Port 3 Mode register (P3M, $F7_H$), P3M is a write-only register. This subroutine library maintains an image of P3M in its register P3M__save ($7F_H$). If software outside of the subroutine package is to modify P3M, it should reference and modify P3M__save prior to modification of P3M. For example, to select P32/P35 for handshake, the following instruction sequence could be used:

```
OR      P3M__save, #04H
LD      P3M, P3M__save
```

2. For many of the subroutines in this library, the location of the operands (source/destination) is flexible between register memory, external memory (code/data), and the serial channel (if enabled). The description of each parameter in the specification blocks tells what the location options are.

   - The location designation "in reg/ext memory" implies that the subroutine allows the operand to exist in register or in external data memory. The address of such an operand is contained in the designated register pair. If the high byte of that pair is 0, the operand is in register memory at the address held in the low byte of the register pair. Otherwise, the operand is in external data memory (accessed via LDE).

   - The location designation "in reg/ext/ser memory" implies the same considerations as above with one enhancement: if both bytes of the register pair are 0, the operand exists in the serial channel. In this case, the register pair is not modified (updated). For example, rather than storing a destination ASCII string in memory, it might be desirable to output the string to the serial line.

3. The BCD format supported by the following arithmetic and conversion routines allows representation of signed variable-precision BCD numbers. A BCD number of 2n digits is represented in n+1 consecutive bytes, where the byte at the lowest memory address (byte 0) represents the sign and post-decimal digit count, and the bytes in the n higher memory locations (bytes 1 through n) represent the magnitude of the BCD number. The address of byte 0 and the value n are passed to the subroutines in specified working registers.

   Digits are packed two per byte with the most-significant digit in the high-order nibble of byte 1 and the least-significant digit in the low-order nibble of byte n. Byte 0 is organized as two fields:

   Bit 7 represents sign:
   1 = negative;
   0 = positive.

   Bits 0-6 represent post-decimal digit count.

   For example:

   byte 0 = 05_H = positive, with five post-decimal digits
   = 80_H = negative, with no post-decimal digits
   = 90_H = negative, with 16 post-decimal digits

4. The format of the decimal ASCII character string expected as input to the conversion routines "dascbcd" and "dascwrd" is defined as:

   (+ 1 -) (<digit>) [(<digit>)]

   in which
   ( ) Parentheses mean that the enclosed times or can be omitted.
   [ ] Brackets denote that the enclosed element is optional.

   Table 3 illustrates how various input strings are interpreted by the conversion routines.

5. The format of the decimal ASCII character string output from the conversion routine "bcddasc" operating on an input BCD string of 2n digits is

   1 sign of character (+ 1 -)
   2n-x pre-decimal digits
   1 decimal point if x does not equal 0
   x post-decimal digits

6. The format of the decimal ASCII character string output from the conversion routine "wrddassc" is

   1 sign character (determined by bit 15 of input word)
   6 pre-decimal digits
   no decimal point
   no post-decimal digits

Table 2. Subroutine Entry Points

Address	Name	Description
Binary Arithmetic Routines		
001B	divide	16/8 unsigned binary division
001E	div_16	16/16 unsigned binary division
0021	multiply	8x8 unsigned binary multiplication
0024	mult_16	16x16 unsigned binary multiplication
BCD Arithmetic Routines		
0027	bcdadd	BCD addition
002A	bcdsub	BCD subtraction
Conversion Routines		
002D	bcddasc	BCD to decimal ASCII
0030	dascbcd	Decimal ASCII to BCD
0033	bcdwrd	BCD to binary word
0036	wrdbcd	Binary word to BCD
0039	bythasc	Binary byte to hexadecimal ASCII
003C	wrdhasc	Binary word to hexadecimal ASCII
003F	hascwrd	Hexadecimal ASCII to binary word
0042	wrddasc	Binary word to decimal ASCII
0045	dascwrd	Decimal ASCII to binary word
Bit Manipulation Routines		
0048	clb	Collect bits in a byte
004B	tmj	Table jump under mask
Serial Routines		
004E	ser_init	Initialize serial I/O
0051	ser_input	IRQ_3 (receive) service
0054	ser_rlin	Read line
0057	ser_rabs	Read absolute
005A	ser_break	Transmit BREAK
005D	ser_flush	Flush (clear) input buffer
0060	ser_wlin	Write line
0063	ser_wabs	Write absolute
0066	ser_wbyt	Write byte
0069	ser_disable	Disable serial I/O
Timer/Counter Routines		
006C	tod_i	Initialize for time-of-day clock
006F	tod	Time-of-day IRQ service
0072	delay	Initialize for delay interval
0075	pulse_i	Initialize for pulse output
0078	pulse	Pulse IRQ service

7. Procedure name: ser_input

The conclusion of the algorithm for BREAK detection requires the Serial Receive Shift register to be cleared of the character currently being collected (if any). This requires a software wait loop of a one-character duration. The following explains the algorithm used (code lines 464 through 472, Part II):

$$1 \text{ character time} = \frac{(128 \times PREO \times TO)}{XTAL} \frac{sec}{bit} \times 10 \frac{bit}{char}$$

$$= \frac{1280 \times PREO \times TO}{XTAL} \frac{sec}{char}$$

A software loop equal to one character time is needed:

$$1 \text{ character time} = \frac{2}{XTAL} \frac{sec}{cycle} \times n \frac{cycle}{loop}$$

$$= \frac{2n}{XTAL} \frac{sec}{loop}$$

Solve for n:

$$\frac{(1280 \times PREO \times TO)}{XTAL} = \frac{2n}{XTAL}$$

$$n = 640 \times PREO \times TO$$

The register pair SERhtime, SER1time was initialized during ser_init to equal the product of the prescaler and the counter selected for the baud rate clock. That is,

SERhtime, SER1time = PREO x TO

The instruction sequence

```
inlop: ld    rSERtmpl, #53  (6 cycles)

lpl:   djnz  rSERtmpl, lpl  (12/10 cycles
                             taken/not taken)
```

executes in

$$6 + (52 \times 12) + 10 \text{ cycles} = 640 \text{ cycles}$$

8. BREAK detection on the serial input line requires that the receive interrupt service routine be entered within a half-a-bit time, since the routine reads the input line to detect a true (=1) or false (=0) stop bit. Since the interrupt request is generated halfway through reception of the stop bit, half-a-bit time remains in which to read the stop bit level. Interrupt priorities and interrupt nesting should be established appropriately to ensure this requirement.

$$1/2 \text{ bit time} = \frac{(128 \times PREO \times TO)}{XTAL \times 2} \text{ sec}$$

Table 3. Decimal ASCII Character String Interpretation

Input String	Result			Terminator
	Sign	Pre-Decimal Digits	Post-Decimal Digits	
+1234.567,	+	1234	567	,
+---+.789+	-		789	+
1234..	+	1234		.
4976-	+		4976	-

NOTE: The terminator can be any ASCII character that is not a valid ASCII string character.

```
Z8ASM      3.02
LOC     OBJ CODE     STMT SOURCE STATEMENT

                       1
                       2
                       3 PART_I  MODULE
                       4
                       5
                       6 !'ROMLESS Z8'   SUBROUTINE LIBRARY  PART I
                       7
                       8  Initialize:     a) Port 0 & Port 1 set up to address
                       9                     64K external memory;
                      10                  b) internal stack below allocated
                      11                     RAM for subroutines;
                      12                  c) normal memory timing;
                      13                  d) IMR, IRQ, TMR, RP cleared;
                      14                  e) Port 2 inputs open-drain pull-ups;
                      15                  f) Data Memory select enabled;
                      16                  g) EI executed to 'unfreeze' IRQ;
                      17                  h) Jump to %0812.
                      18
                      19
                      20 Note:    The user is free to modify the initial
                      21          conditions selected for a, b, and c above,
                      22          via direct modification of the Port 0 & 1
                      23          Mode register (P01M, %F8).
                      24
                      25          The user is free to modify the conditions
                      26          selected in the Port 3 Mode register (P3M, %F7).
                      27          However, please note that P3M is a write-only
                      28          register.  This subroutine library maintains
                      29          an image of P3M in its register P3M_save (%7F).
                      30          If software outside of the subroutine package
                      31          is to modify P3M, it should reference and modify
                      32          P3M_save, prior to modification of P3M.  For
                      33          example, to select P32/P35 for handshake, use
                      34          an instruction sequence such as:
                      35
                      36                  OR      P3M_save,#%04
                      37                  LD      P3M,P3M_save
                      38
                      39          This is important if the serial and/or timer/
                      40          counter subroutines are to be used, since these
                      41          routines may modify P3M.
                      42 !
```

```
!Access to GLOBAL subroutines in this library should
 be made via a CALL to the corresponding entry in the
 jump table which begins at address %000F.  The jump
 table should be referenced rather than a CALL to the
 actual entry point of the subroutine to avoid future
 conflict in the event such entry points change in
 potential future revisions.

 Each GLOBAL subroutine in this listing is headed by a
 comment block specifying its PURPOSE and calling
 sequence (INPUT and OUTPUT parameters).  For many of
 the subroutines in this library, the location of the
 operands (sources/destinations) is quite flexible
 between register memory, external memory (code/data),
 and the serial channel (if enabled).  The description
 of each parameter specifies what the location choices
 are:

        - The location designation 'in reg/ext memory'
 implies that the subroutine allows that the operand
 exist in either register or external data memory
 The address of such an operand is contained
 in the designated register pair.  If the high byte of
 that pair is zero, the operand is in register memory
 at the address given by the low byte of the register
 pair.  Otherwise, the operand is in external data
 memory (accessed via LDE).

        - The location designation
 'in reg/ext/ser memory' implies the same
 considerations as above with one enhancement: if both
 bytes of the reg. pair are zero, the operand exists
 in the serial channel.  In this case, the register
 pair is not modified (updated).  For example, rather
 than storing a destination ASCII string in memory, it
 might be desirable to output such to the serial line.
!
```

```
 82 CONSTANT
 83 !Register Usage!
 84
 85 RAM_START        :=      %7F
 86
 87 P3M_save         :=      RAM_START
 88 TEMP_3           :=      P3M_save-1
 89 TEMP_2           :=      TEMP_3-1
 90 TEMP_1           :=      TEMP_2-1
 91 TEMP_4           :=      TEMP_1-1
 92
 93 !The following registers are modified/referenced
 94  by the Serial Routines ONLY.  They are
 95  available as general registers to the user
 96  who does not intend to make use of the
 97  Serial Routines!
 98
 99 SER_char         :=      TEMP_4-1
100 SER_tmp2         :=      SER_char-1
101 SER_tmp1         :=      SER_tmp2-1
102 SER_put          :=      SER_tmp1-1
103 SER_len          :=      SER_put-1
104 SER_buf          :=      SER_len-2
105 SER_imr          :=      SER_buf-1
106 SER_cfg          :=      SER_imr-1
107 !Serial Configuration Data
108 bit 7 : =1 => odd parity on
109 bit 6 : =1 => even parity on
110  (bit 6,7 = 11 => undefined)
111 bit 5 : undefined
112 bit 4 : undefined
113 bit 3 : =1 => input editting on
114 bit 2 : =1 => auto line feed enabled
115 bit 1 : =1 => BREAK detection enabled
116 bit 0 : =1 => input echo on
117 !
118 op       :=      %80
119 ep       :=      %40
120 ie       :=      %08
121 al       :=      %04
122 be       :=      %02
123 ec       :=      %01
124 SER_get          :=      SER_cfg-1
125 SER_flg          :=      SER_get-1
126 !Serial Status Flags
127 bit 7 : =1 => serial I/O disabled
128 bit 6 : undefined
129 bit 5 : undefined
130 bit 4 : =1 => parity error
131 bit 3 : =1 => BREAK detected
132 bit 2 : =1 => input buffer overflow
133 bit 1 : =1 => input buffer not empty
134 bit 0 : =1 => input buffer full
135 !
136 sd       :=      %80
137 pe       :=      %10
138 bd       :=      %08
139 bo       :=      %04
140 bne      :=      %02
141 bf       :=      %01
142
143 RAM_TMR          :=      RAM_START-%10
144
145 SERltime         :=      SER_flg-1
```

```
146 SERhtime        :=      SERltime-1
147
148 !The following registers are modified/referenced
149  by the Timer/Counter Routines ONLY.  They are
150  available as general registers to the user
151  who does not intend to make use of the
152  Timer/Counter Routines!
153
154 TOD_tic         :=      RAM_TMR-2
155 TOD_imr         :=      TOD_tic-1
156 TOD_hr          :=      TOD_imr-1
157 TOD_min         :=      TOD_hr-1
158 TOD_sec         :=      TOD_min-1
159 TOD_tt          :=      TOD_sec-1
160 PLS_1           :=      TOD_tt-1
161 PLS_tmr         :=      PLS_1-1
162 PLS_2           :=      PLS_tmr-1
163
164 RAM_END         :=      PLS_2
165 STACK           :=      RAM_END
166
167 !Equivalent working register equates
168  for above register layout!
169
170 !register file %70 - %7F!
171 RAM_STARTr      :=      %70     !for SRP!
172
173 rP3Msave        :=      R15
174 rTEMP_3         :=      R14
175 rTEMP_2         :=      R13
176 rTEMP_1         :=      R12
177 rrTEMP_1        :=      RR12
178 rTEMP_1h        :=      R12
179 rTEMP_1l        :=      R13
180 rTEMP_4         :=      R11
181 rSERchar       :=      R10
182 rSERtmp2        :=      R9
183 rSERtmp1        :=      R8
184 rrSERtmp       :=      RR8
185 rSERtmpl        :=      R9
186 rSERtmph        :=      R8
187 rSERput         :=      R7
188 rSERlen         :=      R6
189 rrSERbuf        :=      RR4
190 rSERbufh        :=      R4
191 rSERbufl        :=      R5
192 rSERimr         :=      R3
193 rSERcfg         :=      R2
194 rSERget         :=      R1
195 rSERflg         :=      R0
196
197
198 !register file %60 - %6F!
199 RAM_TMRr        :=      %60     !for SRP!
200 rTODtic         :=      R13
201 rTODimr         :=      R12
202 rTODhr          :=      R11
203 rTODmin         :=      R10
204 rTODsec         :=      R9
205 rTODtt          :=      R8
206 rPLS_1          :=      R7
207 rPLStmr         :=      R6
208 rPLS_2          :=      R5
```

```
                    210 EXTERNAL
                    211  ser_init        PROCEDURE
                    212  ser_input       PROCEDURE
                    213  ser_rlin        PROCEDURE
                    214  ser_rabs        PROCEDURE
                    215  ser_break       PROCEDURE
                    216  ser_flush       PROCEDURE
                    217  ser_wlin        PROCEDURE
                    218  ser_wabs        PROCEDURE
                    219  ser_wbyt        PROCEDURE
                    220  ser_disable     PROCEDURE
                    221  ser_get         PROCEDURE
                    222  ser_output      PROCEDURE
                    223  tod_i           PROCEDURE
                    224  tod             PROCEDURE
                    225  delay           PROCEDURE
                    226  pulse_i         PROCEDURE
                    227  pulse           PROCEDURE
                    228
                    229
                    230         $SECTION PROGRAM
                    231 GLOBAL
                    232
                    233
                    234 !Interrupt vectors!
P 0000 0800         235 IRQ_0    ARRAY   [1 word]  :=    [%0800]
P 0002 0803         236 IRQ_1    ARRAY   [1 word]  :=    [%0803]
P 0004 0806         237 IRQ_2    ARRAY   [1 word]  :=    [%0806]
P 0006 0809         238 IRQ_3    ARRAY   [1 word]  :=    [%0809]
P 0008 080C         239 IRQ_4    ARRAY   [1 word]  :=    [%080C]
P 000A 080F         240 IRQ_5    ARRAY   [1 word]  :=    [%080F]
                    241
                    242
```

```
                        244 GLOBAL
                        245
                        246 !Jump Table!
P 000C                  247 ENTER   PROCEDURE
                        248 ENTRY
P 000C 8D  007B'        249         JP      INIT
P 000F                  250 END     ENTER
                        251
                        252
P 000F 28  43  29       253 copyright ARRAY [* BYTE] := '(C)1980ZILOG'
P 0012 31  39  38
P 0015 30  5A  49
P 0018 4C  4F  47
                        254
                        255 !Subroutine Entry Points!
P 001B                  256 JUMP    PROCEDURE
                        257 ENTRY
                        258
                        259 !Binary Arithmetic Routines!
                        260
P 001B 8D  0099'        261         JP      divide          !16/8 unsigned binary
                        262                                  division!
P 001E 8D  00B7'        263         JP      div_16          !16/16 unsigned binary
                        264                                  division!
P 0021 8D  00E2'        265         JP      multiply        !8x8 unsigned binary
                        266                                  multiplication!
P 0024 8D  00F6'        267         JP      mult_16         !16x16 unsigned binary
                        268                                  multiplication!
                        269
                        270 !BCD Arithmetic Routines!
                        271
P 0027 8D  011A'        272         JP      bcdadd          !BCD addition!
                        273
P 002A 8D  0117'        274         JP      bcdsub          !BCD subtraction!
                        275
                        276 !Conversion Routines!
                        277
P 002D 8D  0205'        278         JP      bcddasc         !BCD to decimal ASCII!
                        279
P 0030 8D  0363'        280         JP      dascbcd         !Decimal ASCII to BCD!
                        281
P 0033 8D  0284'        282         JP      bcdwrd          !BCD to binary word!
                        283
P 0036 8D  02CD'        284         JP      wrdbcd          !binary word to BCD!
                        285
P 0039 8D  025C'        286         JP      bythasc         !Bin. byte to Hex ASCII!
                        287
P 003C 8D  0257'        288         JP      wrdhasc         !Bin. word to hex ASCII!
                        289
P 003F 8D  0319'        290         JP      hascwrd         !Hex ASCII to bin word!
                        291
P 0042 8D  03BE'        292         JP      wrddasc         !Bin. word to dec ASCII!
                        293
P 0045 8D  034D'        294         JP      dascwrd         !dec ASCII to bin word!
                        295
                        296 !Bit Manipulation Routines!
                        297
P 0048 8D  04A1'        298         JP      clb             !collect bits in a byte!
                        299
P 004B 8D  04B9'        300         JP      tjm             !Table Jump Under Mask!
                        301
                        302 !Serial Routines!
                        303
P 004E 8D  0000*        304         JP      ser_init        !initialize serial I/O!
```

```
                        305
P 0051 8D  0000*        306          JP       ser_input       !IRQ3 (receive) service!
                        307
P 0054 8D  0000*        308          JP       ser_rlin        !read line!
                        309
P 0057 8D  0000*        310          JP       ser_rabs        !read absolute!
                        311
P 005A 8D  0000*        312          JP       ser_break       !transmit BREAK!
                        313
P 005D 8D  0000*        314          JP       ser_flush       !flush (clear)
                        315                                    input buffer!
P 0060 8D  0000*        316          JP       ser_wlin        !write line!
                        317
P 0063 8D  0000*        318          JP       ser_wabs        !write absolute!
                        319
P 0066 8D  0000*        320          JP       ser_wbyt        !write byte!
                        321
P 0069 8D  0000*        322          JP       ser_disable     !disable serial I/O!
                        323
                        324 !Timer/Counter Routines!
                        325
P 006C 8D  0000*        326          JP       tod_i           !init for time of day!
                        327
P 006F 8D  0000*        328          JP       tod             !tod IRQ service!
                        329
P 0072 8D  0000*        330          JP       delay           !init for delay interval
                        331
P 0075 8D  0000*        332          JP       pulse_i         !init for pulse output!
                        333
P 0078 8D  0000*        334          JP       pulse           !pulse IRQ service!
                        335
P 007B                  336 END      JUMP

                        338 !Initialization!
P 007B                  339 INIT     PROCEDURE
                        340 ENTRY
                        341
P 007B E6  F8  D7       342          LD       P01M,#%(2)11010111
                        343                                    !internal stack;
                        344                                     AD0-A15;
                        345                                     normal memory
                        346                                     timing !
P 007E E6  7F  10       347          LD       P3M_save,#%(2)00010000
                        348                                    !P3M is write-only,
                        349                                     so keep a copy in
                        350                                     RAM for later
                        351                                     reference !
P 0081 E4  7F  F7       352          LD       P3M,P3M_save    !set up Port 3 !
P 0084 E6  FF  65       353          LD       SPL,#STACK      !stack pointer !
P 0087 B0  F1           354          CLR      TMR             !reset timers!
P 0089 E6  F6  FF       355          LD       P2M,#%FF        !all inputs!
P 008C B0  FA           356          CLR      IRQ             !reset int. requests!
P 008E B0  FB           357          CLR      IMR             !disable interrupts !
P 0090 B0  FD           358          CLR      RP              !register pointer!
P 0092 E6  70  80       359          LD       SER_flg,#%80    !serial disabled!
P 0095 9F               360          EI                       !globally enable
                        361                                     interrupts !
P 0096 8D  0812         362          JP       %0812
                        363
P 0099                  364 END      INIT
```

```
                    397 CONSTANT
                    398  div_LEN          :=      R10
                    399  DIVISOR          :=      R11
                    400  dividend_HI      :=      R12
                    401  dividend_LO      :=      R13
                    402 GLOBAL
P 0099              403 divide  PROCEDURE
                    404 !***************************************************
                    405  Purpose =        To perform a 16-bit by 8-bit unsigned
                    406                   binary division.
                    407
                    408  Input =          R11 = 8-bit divisor
                    409                   RR12 = 16-bit dividend
                    410
                    411  Output =         R13  = 8-bit quotient
                    412                   R12  = 8-bit remainder
                    413                   Carry flag = 1 if overflow
                    414                              = 0 if no overflow
                    415                   R11 unmodified
                    416 ***************************************************!
                    417 ENTRY
P 0099 A9  7C       418          ld       TEMP_1,div_LEN  !save caller's R10!
P 009B AC  08       419          ld       div_LEN,#8      !LOOP COUNTER!
                    420
                    421 !CHECK IF RESULT WILL FIT IN 8 BITS!
P 009D A2  BC       422          cp       DIVISOR,dividend_HI
P 009F BB  02       423          jr       UGT,LOOP        !CARRY = 0 (FOR RLC)!
                    424 !overflow!
P 00A1 DF           425          SCF                      !CARRY = 1!
P 00A2 AF           426          ret
                    427
P 00A3 10  ED       428 LOOP:    RLC      dividend_LO     !DIVIDEND * 2!
P 00A5 10  EC       429          RLC      dividend_HI
P 00A7 7B  04       430          jr       c,subt
P 00A9 A2  BC       431          cp       DIVISOR,dividend_HI
P 00AB BB  03       432          jr       UGT,next        !CARRY = 0!
P 00AD 22  CB       433 subt:    SUB      dividend_HI,DIVISOR
P 00AF DF           434          SCF                 !TO BE SHIFTED INTO RESULT!
P 00B0 AA  F1       435 next:    djnz     div_LEN,LOOP    !no flags affected!
                    436
                    437 !ALL     DONE!
P 00B2 10  ED       438          RLC      dividend_LO
                    439                                   !CARRY = 0: no overflow!
P 00B4 A8  7C       440          ld       div_LEN,TEMP_1  !restore caller's R10!
P 00B6 AF           441          ret
P 00B7              442 END divide
```

```
                        444 CONSTANT
                        445  d16_LEN          :=      R7
                        446  dvsr_hi          :=      R8
                        447  dvsr_lo          :=      R9
                        448  rem_hi           :=      R10
                        449  rem_lo           :=      R11
                        450  quot_hi          :=      R12
                        451  quot_lo          :=      R13
                        452 GLOBAL
P 00B7                  453 div_16  PROCEDURE
                        454 !*****************************************************
                        455  Purpose =        To perform a 16-bit by 16-bit unsigned
                        456                   binary division.
                        457
                        458  Input =          RR8 = 16-bit divisor
                        459                   RR12 = 16-bit dividend
                        460
                        461  Output =         RR12  = 16-bit quotient
                        462                   RR10  = 16-bit remainder
                        463                   RR8 unmodified
                        464 *****************************************************!
                        465 ENTRY
P 00B7 79  7C           466          ld       TEMP_1,d16_LEN  !save caller's R10!
P 00B9 7C  10           467          ld       d16_LEN,#16     !LOOP COUNTER!
P 00BB CF               468          rcf                      !carry = 0!
P 00BC B0  EA           469          clr      rem_hi
P 00BE B0  EB           470          clr      rem_lo
P 00C0 10  ED           471 dlp_16:  rlc      quot_lo
P 00C2 10  EC           472          rlc      quot_hi
P 00C4 10  EB           473          rlc      rem_lo
P 00C6 10  EA           474          rlc      rem_hi
P 00C8 7B  0A           475          jr       c,subt_16
P 00CA A2  8A           476          cp       dvsr_hi,rem_hi
P 00CC BB  0B           477          jr       ugt,skp_16
P 00CE 7B  04           478          jr       ult,subt_16
P 00D0 A2  9B           479          cp       dvsr_lo,rem_lo
P 00D2 BB  05           480          jr       ugt,skp_16
P 00D4 22  B9           481 subt_16: sub      rem_lo,dvsr_lo
P 00D6 32  A8           482          sbc      rem_hi,dvsr_hi
P 00D8 DF               483          scf
P 00D9 7A  E5           484 skp_16:  djnz     d16_LEN,dlp_16  !no flags affected!
P 00DB 10  ED           485          rlc      quot_lo
P 00DD 10  EC           486          rlc      quot_hi
P 00DF 78  7C           487          ld       d16_LEN,TEMP_1
P 00E1 AF               488          ret
P 00E2                  489 END div_16

                        491 CONSTANT
                        492  MULTIPLIER       :=      R11
                        493  PRODUCT_LO       :=      R13
                        494  PRODUCT_HI       :=      R12
                        495  mul_LEN          :=      R10
                        496 GLOBAL
P 00E2                  497 multiply         PROCEDURE
                        498 !*****************************************************
                        499  Purpose =        To perform an 8-bit by 8-bit unsigned
                        500                   binary multiplication.
                        501
                        502  Input =          R11 = multiplier
                        503                   R13 = multiplicand
                        504
                        505  Output =         RR12 = product
                        506                   R11 unmodified
                        507 *****************************************************!
                        508 ENTRY
P 00E2 A9  7C           509          ld       TEMP_1,mul_LEN  !save caller's R10!
P 00E4 AC  09           510          ld       mul_LEN,#9      !8 BITS!
P 00E6 B0  EC           511          clr      PRODUCT_HI      !INIT HIGH RESULT BYTE!
P 00E8 CF               512          RCF                      !CARRY = 0!
P 00E9 C0  EC           513 LOOP1:   RRC      PRODUCT_HI
P 00EB C0  ED           514          RRC      PRODUCT_LO
P 00ED FB  02           515          jr       NC,NEXT
P 00EF 02  CB           516          ADD      PRODUCT_HI,MULTIPLIER
P 00F1 AA  F6           517 NEXT:    djnz     mul_LEN,LOOP1
P 00F3 A8  7C           518          ld       mul_LEN,TEMP_1  !restore caller's R10!
P 00F5 AF               519          ret
P 00F6                  520 END      multiply
```

```
                        522 CONSTANT
                        523  m16_LEN          :=      R7
                        524  plier_hi         :=      R8
                        525  plier_lo         :=      R9
                        526  prod_hi          :=      R10
                        527  prod_lo          :=      R11
                        528  mult_hi          :=      R12
                        529  mult_lo          :=      R13
                        530 GLOBAL
P 00F6                  531 mult_16 PROCEDURE
                        532 !***************************************************
                        533  Purpose =        To perform an 16-bit by 16-bit unsigned
                        534                   binary multiplication.
                        535
                        536  Input =          RR8 = multiplier
                        537                   RR12 = multiplicand
                        538
                        539  Output =         RQ10 = product (R10, R11, R12, R13)
                        540                   RR8 unmodified
                        541                   Zero FLAG = 0 if result > 16 bits
                        542                             = 1 if result fits in 16
                        543                     (unsigned) bits (RR12 = result)
                        544 ***************************************************!
                        545 ENTRY
P 00F6 79  7C           546          ld       TEMP_1,m16_LEN  !save caller's R7!
P 00F8 7C  11           547          ld       m16_LEN,#17     !16 BITS!
P 00FA B0  EA           548          clr      prod_hi
P 00FC B0  EB           549          clr      prod_lo         !init product!
P 00FE CF               550          rcf                      !CARRY = 0!
P 00FF C0  EA           551 loop16:  rrc      prod_hi
P 0101 C0  EB           552          rrc      prod_lo         !bit 0 to carry!
P 0103 C0  EC           553          rrc      mult_hi         !multiplicand / 2!
P 0105 C0  ED           554          rrc      mult_lo
P 0107 FB  04           555          jr       nc,next16
P 0109 02  B9           556          add      prod_lo,plier_lo
P 010B 12  A8           557          adc      prod_hi,plier_hi
P 010D 7A  F0           558 next16:  djnz     m16_LEN,loop16  !next bit!
P 010F 78  7C           559          ld       m16_LEN,TEMP_1  !restore caller's R7!
P 0111 A9  7C           560          ld       TEMP_1,prod_hi  !test product...!
P 0113 44  EB  7C       561          or       TEMP_1,prod_lo  !...bits 31 - 16!
P 0116 AF               562          ret
P 0117                  563 END      mult_16
```

```
                    593 !The BCD format supported by the following arithmetic
                    594  and conversion routines allows representation
                    595  of signed magnitude variable precision BCD
                    596  numbers.  A BCD number of 2n digits is
                    597  represented in n+1 consecutive bytes where
                    598  the byte at the lowest memory address
                    599  ('byte 0') represents the sign and post-
                    600  decimal digit count, and the bytes in the
                    601  next n higher memory locations ('byte 1'
                    602  through 'byte n') represent the magnitude
                    603  of the BCD number.  The address of 'byte 0'
                    604  and the value n are passed to the subroutines
                    605  in specified working registers.  Digits are
                    606  packed two per byte with the most
                    607  significant digit in the high order nibble
                    608  of 'byte 1' and the least significant digit
                    609  in the low order nibble of 'byte n'.  'Byte 0'
                    610  is organized as two fields:
                    611         bit 7 represents sign:
                    612              = 1 => negative
                    613              = 0 => positive
                    614         bit 6-0 represent post-decimal digit
                    615                 count
                    616 For example:
                    617 'byte 0'= %05 => positive, with 5 post-decimal digits
                    618         = %80 => negative, with no post-decimal digits
                    619         = %90 => negative, with 16 post-decimal digits
                    620 !

                    622 CONSTANT
                    623  bcd_LEN := R12
                    624  bcd_SRC := R14
                    625  bcd_DST := R15
                    626 GLOBAL
P 0117              627 bcdsub  PROCEDURE
                    628 !*****************************************************
                    629  Purpose =      To subtract two packed BCD strings of
                    630                 equal length.
                    631                 dst  <-- dst - src
                    632
                    633  Input =        R15 = address of destination BCD
                    634                      string (in register memory).
                    635                 R14 = address of source BCD
                    636                      string (in register memory).
                    637                 R12 = BCD digit count / 2
                    638
                    639  Output =       Destination BCD string contains the
                    640                 difference.
                    641                 Source BCD string may be modified.
                    642                 R12, R14, R15 unmodified if no error
                    643                 R13 modified.
                    644                 Carry FLAG = 1 if underflow or format
                    645                                  error.
                    646 *****************************************************!
                    647 ENTRY
P 0117 B7  EE  80   648         xor     @bcd_SRC,#%80   !complement sign of
                    649                                  subtrahend!
                    650 !fall into bcdadd!
P 011A              651 END     bcdsub
```

```
                        653 GLOBAL
P 011A                  654 bcdadd  PROCEDURE
                        655 !*****************************************************
                        656  Purpose =       To add two packed BCD strings of
                        657                  equal length.
                        658                  dst  <-- dst + src
                        659
                        660  Input =         R15 = address of destination BCD
                        661                        string (in register memory).
                        662                  R14 = address of source BCD
                        663                        string (in register memory).
                        664                  R12 = BCD digit count / 2
                        665
                        666  Output =        Destination BCD string contains the sum.
                        667                  Source BCD string may be modified.
                        668                  R12, R14, R15 unmodified if no error
                        669                  R13 modified.
                        670                  Carry FLAG = 1 if overflow or format
                        671                                error.
                        672 *****************************************************!
                        673 ENTRY
                        674 !delete all leading pre-decimal zeroes!
P 011A E6  7E  02       675          ld      TEMP_3,#2
P 011D D8  EE           676          ld      R13,bcd_SRC
P 011F C9  7B           677 ba_3:    ld      TEMP_4,bcd_LEN
P 0121 04  7B  7B       678          add     TEMP_4,TEMP_4    !total digit count!
P 0124 E5  ED  7D       679          ld      TEMP_2,@R13      !get sign/post dec #!
P 0127 56  7D  7F       680          and     TEMP_2,#%7F      !isolate post dec #!
P 012A 24  7D  7B       681          sub     TEMP_4,TEMP_2    !pre-dec digit cnt!
P 012D 7D  0203'        682          jp      ult,ba_err       !format error!
P 0130 6B  1A           683          jr      z,ba_1           !no pre-dec. digits!
P 0132 70  EC           684 ba_2:    push    R12              !save!
P 0134 C7  CD  01       685          ld      R12,1(R13)       !leading byte!
P 0137 76  EC  F0       686          tm      R12,#%F0         !test leading digit!
P 013A 50  EC           687          pop     R12              !restore!
P 013C EB  0E           688          jr      nz,ba_1          !no more leading 0's!
P 013E B0  7C           689          clr     TEMP_T
P 0140 D6  0463'        690          call    rdl              !rotate left!
P 0143 21  ED           691          inc     @R13             !update post dec #!
P 0145 4D  0203'        692          jp      ov,ba_err        !oops!
P 0148 00  7B           693          dec     TEMP_4           !dec pre-dec #!
P 014A EB  E6           694          jr      nz,ba_2          !loop!
P 014C D8  EF           695 ba_1:    ld      R13,bcd_DST
P 014E 00  7E           696          dec     TEMP_3           !SRC and DST done?!
P 0150 EB  CD           697          jr      nz,ba_3          !do DST!
                        698 !leading zero deletion complete!
                        699 !insure DST is > or = SRC; exchange if necessary!
P 0152 E3  DF           700          ld      R13,@bcd_DST
P 0154 56  ED  7F       701          and     R13,#%7F         !isolate post dec #!
P 0157 E5  EE  7D       702          ld      TEMP_2,@bcd_SRC
P 015A 56  7D  7F       703          and     TEMP_2,#%7F      !isolate post dec #!
P 015D A4  7D  ED       704          cp      R13,TEMP_2
P 0160 70  ED           705          push    R13              !save!
P 0162 7B  39           706          jr      ult,ba_4         !DST > SRC!
P 0164 BB  18           707          jr      ugt,ba_5         !DST < SRC!
                        708 !decimal points in same position.
                        709  must compare magnitude!
P 0166 D8  EC           710          ld      R13,bcd_LEN
P 0168 E9  7C           711          ld      TEMP_1,bcd_SRC
P 016A F9  7B           712          ld      TEMP_4,bcd_DST
P 016C 20  7C           713 ba_6:    inc     TEMP_1
P 016E 20  7B           714          inc     TEMP_4
P 0170 E5  7C  7E       715          ld      TEMP_3,@TEMP_1   !get SRC byte!
P 0173 A5  7B  7E       716          cp      TEMP_3,@TEMP_4   !compare DST byte!
```

```
P 0176 BB  06        717          jr      ugt,ba_5        !SRC > DST!
P 0178 7B  23        718          jr      ult,ba_4        !SRC < DST!
P 017A DA  F0        719          djnz    R13,ba_6        !loop!
P 017C 8B  1F        720          jr      ba_4            !DST > or = SRC!
                     721 !swap source and destination operands!
P 017E D8  EC        722 ba_5:    ld      R13,bcd_LEN
P 0180 DE            723          inc     R13             !include flag/size byte!
P 0181 02  ED        724          add     bcd_SRC,R13
P 0183 02  FD        725          add     bcd_DST,R13
P 0185 00  EE        726 ba_7:    dec     bcd_SRC
P 0187 00  EF        727          dec     bcd_DST
P 0189 E5  EE  7C    728          ld      TEMP_1,@bcd_SRC
P 018C E5  EF  7B    729          ld      TEMP_4,@bcd_DST
P 018F F5  7B  EE    730          ld      @bcd_SRC,TEMP_4
P 0192 F5  7C  EF    731          ld      @bcd_DST,TEMP_1 !one byte swapped!
P 0195 DA  EE        732          djnz    R13,ba_7
P 0197 D8  7D        733          ld      R13,TEMP_2
P 0199 50  7D        734          pop     TEMP_2
P 019B 70  ED        735          push    R13
                     736 !exchange complete!
P 019D 50  ED        737 ba_4:    pop     R13             !restore!
                     738 !R13 = DST post decimal digit count
                     739  TEMP_2 = SRC post decimal digit count
                     740  R13 =< TEMP_2                    !
P 019F 24  ED  7D    741          sub     TEMP_2,R13
P 01A2 C0  7D        742          rrc     TEMP_2          !alignment offset!
P 01A4 FB  09        743          jr      nc,ba_8         !digits word aligned!
                     744 !rotate out least significant SRC post decimal digit!
P 01A6 D8  EE        745          ld      R13,bcd_SRC
P 01A8 01  ED        746          dec     @R13            !dec post dec digit #!
P 01AA B0  7C        747          clr     TEMP_1
P 01AC D6  0485'     748          call    rdr
                     749 !determine if addition or subtraction!
P 01AF E5  EE  7B    750 ba_8:    ld      TEMP_4,@bcd_SRC !sign of SRC!
P 01B2 B5  EF  7B    751          xor     TEMP_4,@bcd_DST !sign of DST!
                     752 !get starting addresses!
P 01B5 D8  EC        753          ld      R13,bcd_LEN
P 01B7 24  7D  ED    754          sub     R13,TEMP_2
P 01BA 6B  45        755          jr      z,ba_14         !done already!
P 01BC 02  ED        756          add     bcd_SRC,R13
P 01BE 02  FC        757          add     bcd_DST,bcd_LEN
                     758 !ready!!!
P 01C0 CF            759          rcf                     !carry = 0!
P 01C1 E5  EF  7C    760 ba_11:   ld      TEMP_1,@bcd_DST
P 01C4 76  7B  80    761          tm      TEMP_4,#%80     !add or sub?!
P 01C7 6B  05        762          jr      z,ba_9          !add!
P 01C9 35  EE  7C    763          sbc     TEMP_1,@bcd_SRC
P 01CC 8B  03        764          jr      ba_10
P 01CE 15  EE  7C    765 ba_9:    adc     TEMP_1,@bcd_SRC
P 01D1 40  7C        766 ba_10:   da      TEMP_1
P 01D3 F5  7C  EF    767          ld      @bcd_DST,TEMP_1
P 01D6 00  EF        768          dec     bcd_DST
P 01D8 00  EE        769          dec     bcd_SRC
P 01DA DA  E5        770          djnz    R13,ba_11
                     771 !propagate carry thru TEMP_2 bytes of DST!
P 01DC D8  7D        772          ld      R13,TEMP_2
P 01DE DE            773          inc     R13             !may be zero!
P 01DF DA  02        774          djnz    R13,ba_12
P 01E1 8B  09        775          jr      ba_13
P 01E3 17  EF  00    776 ba_12:   adc     @bcd_DST,#0
P 01E6 41  EF        777          da      @bcd_DST
P 01E8 00  EF        778          dec     bcd_DST
P 01EA DA  F7        779          djnz    R13,ba_12
```

```
                        780 !carry propagate complete!
P 01EC FB   13          781 ba_13:  jr      nc,ba_14        !done!
                        782 !Rotate out least significant post decimal DST
                        783  digit to make room for carry at high end!
P 01EE E5   EF  7C      784         ld      TEMP_1,@bcd_DST
P 01F1 56   7C  7F      785         and     TEMP_1,#%7F
P 01F4 6D   0203'       786         jp      z,ba_err        !no post dec digits!
P 01F7 E6   7C  10      787         ld      TEMP_1,#%10
P 01FA D8   EF          788         ld      R13,bcd_DST
P 01FC D6   0485'       789         call    rdr
P 01FF 01   EF          790         dec     @bcd_DST        !dec digit cnt!
P 0201 CF               791 ba_14:  rcf
P 0202 AF               792         ret
                        793
P 0203 DF               794 ba_err: scf
P 0204 AF               795         ret
P 0205                  796 END     bcdadd
```

```
                                821 CONSTANT
                                822  bca_LEN          :=      R12
                                823  bca_SRC          :=      R13
                                824 GLOBAL
P 0205                          825 bcddasc PROCEDURE
                                826 !*************************************************
                                827  Purpose =        To convert a variable length BCD
                                828                   string to decimal ASCII.
                                829
                                830  Input =          RR14 = address of destination ASCII
                                831                        string (in reg/ext/ser memory).
                                832                   R13 = address of source BCD
                                833                        string (in register memory).
                                834                   R12 = BCD digit count / 2
                                835
                                836  Output =         ASCII string in designated
                                837                   destination buffer.
                                838                   Carry FLAG = 1 if input format error
                                839                                or serial disabled,
                                840                              = 0 if no error.
                                841                   R12, R13, R14, R15 modified.
                                842                   Input BCD string ummodified.
                                843 *************************************************!
                                844 ENTRY
P 0205 E6  7C  2D               845          ld       TEMP_1,#'-'      !minus sign!
P 0208 77  ED  80               846          tm       @bca_SRC,#%80    !src negative?!
P 020B EB  03                   847          jr       nz,bcd_d1        !yes!
P 020D E6  7C  2B               848          ld       TEMP_1,#'+'      !positive sign!
P 0210 E5  ED  7E               849 bcd_d1:  ld       TEMP_3,@bca_SRC
P 0213 56  7E  7F               850          and      TEMP_3,#%7F      !isolate post dec cnt!
P 0216 02  CC                   851          add      bca_LEN,bca_LEN  !total digit count!
P 0218 70  EC                   852          push     bca_LEN
P 021A 24  7E  EC               853          sub      bca_LEN,TEMP_3   !pre-dec digit cnt!
P 021D 50  7E                   854          pop      TEMP_3           !total digit count!
P 021F 7B  35                   855          jr       ult,bcd_d2       !format error!
P 0221 D6  03F4'                856          call     put_dest         !sign to dest.!
P 0224 7B  30                   857          jr       c,bcd_d2         !serial error!
P 0226 A6  EC  00               858          cp       bca_LEN,#0       !any pre-dec digits?!
P 0229 6B  22                   859          jr       z,bcd_d6         !no. start with '.'!
P 022B 76  7E  01               860 bcd_d4:  tm       TEMP_3,#1        !need next byte?!
P 022E EB  04                   861          jr       nz,bcd_d3        !not yet.!
P 0230 DE                       862          inc      bca_SRC          !update pointer!
P 0231 E5  ED  7D               863          ld       TEMP_2,@bca_SRC  !get next byte!
P 0234 F0  7D                   864 bcd_d3:  swap     TEMP_2
P 0236 E4  7D  7C               865          ld       TEMP_1,TEMP_2
P 0239 56  7C  0F               866          and      TEMP_1,#%0F      !isolate digit!
P 023C A6  7C  09               867          cp       TEMP_1,#9        !verify bcd!
P 023F BB  14                   868          jr       ugt,bcd_d5       !no good!
P 0241 06  7C  30               869          add      TEMP_1,#%30      !convert to ASCII!
P 0244 D6  03F4'                870          call     put_dest         !to destination!
P 0247 00  7E                   871          dec      TEMP_3           !digit count!
P 0249 6B  0B                   872          jr       z,bcd_d2         !all done!
P 024B CA  DE                   873          djnz     bca_LEN,bcd_d4   !next digit!
P 024D E6  7C  2E               874 bcd_d6:  ld       TEMP_1,#'.'      !time for dec. pt.!
P 0250 D6  03F4'                875          call     put_dest         !to destination!
P 0253 8B  D6                   876          jr       bcd_d4           !continue!
P 0255 DF                       877 bcd_d5:  scf                       !set error return!
P 0256 AF                       878 bcd_d2:  ret
P 0257                          879 END      bcddasc

                                881 GLOBAL
P 0257                          882 wrdhasc PROCEDURE
                                883 !*************************************************
                                884  Purpose =        To convert a binary word to Hex ASCII.
                                885
                                886  Input =          RR12 = source binary word.
                                887                   RR14 = address of destination ASCII
                                888                        string (in reg/ext/ser memory).
                                889
                                890  Note =           All other details same as for bythasc.
                                891 *************************************************!
                                892 ENTRY
P 0257 D6  025C'                893          call     bythasc          !convert R12!
P 025A C8  ED                   894          ld       R12,R13
                                895 !fall into bythasc!
P 025C                          896 END      wrdhasc
```

```
                        898 CONSTANT
                        899  bna_SRC        :=      R12
                        900 GLOBAL
P 025C                  901 bythasc PROCEDURE
                        902 !***************************************************
                        903  Purpose =      To convert a binary byte to Hex ASCII.
                        904
                        905  Input =        RR14 = address of destination ASCII
                        906                       string (in reg/ext/ser memory).
                        907                 R12 = Source binary byte.
                        908
                        909  Output =       ASCII string in designated
                        910                 destination buffer.
                        911                 Carry = 1 if error (serial only).
                        912                 R14, R15 modified.
                        913 ***************************************************!
                        914 ENTRY
P 025C B0  7E           915         clr     MODE    !flag => binary to ASCII!
P 025E E6  7D  02       916 bca_go: ld      TEMP_2,#2
P 0261 F0  EC           917 bca_go1: SWAP   bna_SRC         !look at next nibble!
P 0263 C9  7C           918         ld      TEMP_1,bna_SRC
P 0265 56  7C  0F       919         and     TEMP_1,#%0F     !isolate low nibble!
P 0268 06  7C  30       920         ADD     TEMP_1,#%30     !convert to ASCII!
P 026B A6  7C  3A       921         cp      TEMP_1,#%3A     !>9?!
P 026E 7B  09           922         jr      ult,skip        !no!
P 0270 DF               923         SCF                     !in case error!
P 0271 76  7E  01       924         TM      MODE,#1         !input is BCD?!
P 0274 EB  0D           925         JR      NZ,bca_ex       !yes. error.!
P 0276 06  7C  07       926         ADD     TEMP_1,#%07     !input hex. adjust!
P 0279 D6  03F4'        927 skip:   call    put_dest        !put byte in dest!
P 027C 7B  05           928         jr      c,bca_ex        !error!
P 027E 00  7D           929         dec     TEMP_2
P 0280 EB  DF           930         jr      nz,bca_go1      !loop till done!
P 0282 CF               931         RCF                     !carry = 0: no error!
P 0283 AF               932 bca_ex: ret                     !done!
P 0284                  933 END     bythasc
```

```
                        935 CONSTANT
                        936  bcd_adr          :=      R14
                        937  bcd_cnt          :=      R15
                        938 GLOBAL
P 0284                  939 bcdwrd  PROCEDURE
                        940 !*****************************************************
                        941  Purpose =        To convert a variable length BCD
                        942                   string to a signed binary word.  Only
                        943                   pre-decimal digits are converted.
                        944
                        945  Input =          R14 = address of source BCD
                        946                        string (in register memory).
                        947                   R15 = BCD digit count / 2
                        948
                        949  Output =         RR12 = binary word
                        950                   Carry FLAG = 1 if input format error
                        951                                or dest overflow,
                        952                              = 0 if no error.
                        953                   R14,R15 modified.
                        954 *****************************************************!
                        955 ENTRY
P 0284 B0  EC           956          clr      R12              !init destination!
P 0286 B0  ED           957          clr      R13
P 0288 E5  EE  7B       958          ld       TEMP_4,@bcd_adr  !get sign/post_length!
P 028B 56  7B  7F       959          and      TEMP_4,#%7F      !isolate post_length!
P 028E 02  FF           960          add      bcd_cnt,bcd_cnt  !# bcd digits!
P 0290 24  7B  EF       961          sub      bcd_cnt,TEMP_4   !# pre-dec digits!
P 0293 7B  37           962          jr       ult,bcd_w2       !format error!
P 0295 E5  EE  7B       963          ld       TEMP_4,@bcd_adr  !remember sign!
P 0298 E6  7E  02       964 bcd_w3:  ld       TEMP_3,#2        !digits per byte!
P 029B EE               965          inc      bcd_adr          !src address!
P 029C E5  EE  7D       966          ld       TEMP_2,@bcd_adr  !get next src byte!
P 029F A6  EF  00       967 bcd_w1:  cp       bcd_cnt,#0       !digit count = 0?!
P 02A2 6B  12           968          jr       z,bcd_w4         !conversion complete!
P 02A4 F0  7D           969          swap     TEMP_2           !next digit!
P 02A6 E4  7D  7C       970          ld       TEMP_1,TEMP_2
P 02A9 D6  042C'        971          call     bcd_bin          !accumulate in binary!
P 02AC 7B  1E           972          jr       c,bcd_w2         !overflow or format err!
P 02AE 00  EF           973          dec      bcd_cnt          !update digit count!
P 02B0 00  7E           974          dec      TEMP_3           !next byte?!
P 02B2 EB  EB           975          jr       nz,bcd_w1        !no. same.!
P 02B4 8B  E2           976          jr       bcd_w3           !next byte!
P 02B6 DF               977 bcd_w4:  scf                       !in case!
P 02B7 76  EC  80       978          tm       R12,#%80         !result > 15 bits?!
P 02BA EB  10           979          jr       nz,bcd_w2        !overflow!
P 02BC 76  7B  80       980 bcd_w5:  tm       TEMP_4,#%80      !source negative?!
P 02BF 6B  0A           981          jr       z,bcd_w6         !no. done.!
P 02C1 60  EC           982          com      R12
P 02C3 60  ED           983          com      R13
P 02C5 06  ED  01       984          add      R13,#1
P 02C8 16  EC  00       985          adc      R12,#0           !RR12 two's complement!
P 02CB CF               986 bcd_w6:  rcf                       !carry = 0!
P 02CC AF               987 bcd_w2:  ret
P 02CD                  988 END      bcdwrd
```

```
                        990 GLOBAL
P 02CD                  991 wrdbcd  PROCEDURE
                        992 !*****************************************************
                        993  Purpose =      To convert a signed binary word
                        994                 to a variable length BCD string.
                        995
                        996  Input =        R14 = address of destination BCD
                        997                       string (in register memory)
                        998                 RR12 = source binary word
                        999                 R15 = BCD digit count / 2
                       1000
                       1001  Output =       BCD string in destination buffer
                       1002                 Carry FLAG = 1 if dest overflow
                       1003                            = 0 if no error.
                       1004                 R12,R13,R14,R15 modified.
                       1005 *****************************************************!
                       1006 ENTRY
P 02CD B1  EE          1007          clr     @bcd_adr          !init sign/post_dec_cnt!
P 02CF 76  EC  80      1008          tm      R12,#%80          !is input word negative?
P 02D2 6B  0D          1009          jr      z,wrd_b0
P 02D4 47  EE  80      1010          or      @bcd_adr,#%80     !set result negative!
P 02D7 60  ED          1011          com     R13
P 02D9 60  EC          1012          com     R12
P 02DB 06  ED  01      1013          add     R13,#1
P 02DE 16  EC  00      1014          adc     R12,#0            !RR12 two's complement!
P 02E1 10  ED          1015 wrd_b0:  rlc     R13
P 02E3 10  EC          1016          rlc     R12               !bit 15 not magnitude!
P 02E5 EE              1017          inc     bcd_adr           !update dest pointer!
P 02E6 E9  7C          1018          ld      TEMP_1,bcd_adr
P 02E8 F9  7D          1019          ld      TEMP_2,bcd_cnt    !dest byte count!
P 02EA 04  EF  7C      1020          add     TEMP_1,bcd_cnt
P 02ED 00  7C          1021          dec     TEMP_1            != bcd end addr!
P 02EF B1  EE          1022 wrd_b1:  clr     @bcd_adr          !initialize dest!
P 02F1 EE              1023          inc     bcd_adr
P 02F2 FA  FB          1024          djnz    bcd_cnt,wrd_b1
P 02F4 E6  7E  0F      1025          ld      TEMP_3,#15        !source bit count!
P 02F7 70  7E          1026 wrd_b3:  push    TEMP_3
P 02F9 10  ED          1027          rlc     R13
P 02FB 10  EC          1028          rlc     R12               !bit 15 to carry!
P 02FD E8  7C          1029          ld      bcd_adr,TEMP_1    !start at end!
P 02FF F8  7D          1030          ld      bcd_cnt,TEMP_2    !dest byte count!
                       1031 !(dest bcd string) <-- (dest bcd string * 2) + carry!
P 0301 E5  EE  7E      1032 wrd_b2:  ld      TEMP_3,@bcd_adr
P 0304 15  EE  7E      1033          adc     TEMP_3,@bcd_adr   !* 2 + carry!
P 0307 40  7E          1034          da      TEMP_3
P 0309 F5  7E  EE      1035          ld      @bcd_adr,TEMP_3
P 030C 00  EE          1036          dec     bcd_adr           !next two digits!
P 030E FA  F1          1037          djnz    bcd_cnt,wrd_b2    !loop for all digits!
P 0310 50  7E          1038          pop     TEMP_3            !restore src bit cnt!
P 0312 7B  04          1039          jr      c,wrd_ex          !dest. overflow!
P 0314 00  7E          1040          dec     TEMP_3
P 0316 EB  DF          1041          jr      nz,wrd_b3         !next bit!
P 0318 AF              1042 wrd_ex:  ret
P 0319                 1043 END      wrdbcd
```

```
                    1045 GLOBAL
P 0319              1046 hascwrd PROCEDURE
                    1047 !*****************************************************
                    1048  Purpose =       To convert a variable length Hex
                    1049                  ASCII string to binary.
                    1050
                    1051  Input =         RR14 = address of source ASCII
                    1052                         string (in reg/ext/ser memory).
                    1053
                    1054  Output =        RR12 = binary word (any overflow
                    1055                  high order digits are truncated
                    1056                  without error).
                    1057                  Carry FLAG = 1 if input error
                    1058                                       (serial only)
                    1059                         (SER_flg indicates cause)
                    1060                           = 0 if no error
                    1061                  R14, R15 modified
                    1062
                    1063  Note =          The ASCII input string processing is
                    1064                  terminated with the occurrence of a
                    1065                  non-hex ASCII character.
                    1066 *****************************************************!
                    1067 ENTRY
P 0319 B0  7E       1068          clr     TEMP_3
P 031B B0  EC       1069          clr     R12
P 031D B0  ED       1070          clr     R13             !init output!
P 031F D6  03DA'    1071 has_c1:  call    get_src         !get input!
P 0322 7B  28       1072          jr      c,has_ex1       !error!
P 0324 D6  040D'    1073          call    ver_asc         !verify hex ASCII!
P 0327 7B  22       1074          jr      c,has_ex        !end conversion!
P 0329 A6  7C  39   1075          cp      TEMP_1,#%39
P 032C 3B  03       1076          jr      ule,has_c2
P 032E 26  7C  37   1077          sub     TEMP_1,#%37
                    1078 !Shift left one nibble!
                    1079 !Insert new nibble in least significant nibble!
P 0331 F0  ED       1080 has_c2:  swap    R13
P 0333 D9  7D       1081          ld      TEMP_2,R13
P 0335 56  ED  F0   1082          and     R13,#%F0
P 0338 56  7C  0F   1083          and     TEMP_1,#%0F
P 033B 44  7C  ED   1084          or      R13,TEMP_1
P 033E F0  EC       1085          swap    R12
P 0340 56  EC  F0   1086          and     R12,#%F0
P 0343 56  7D  0F   1087          and     TEMP_2,#%0F
P 0346 44  7D  EC   1088          or      R12,TEMP_2
P 0349 8B  D4       1089          jr      has_c1          !loop!
P 034B CF           1090 has_ex:  rcf                     !no error!
P 034C AF           1091 has_ex1:ret
P 034D              1092 END      hascwrd
```

```
                    1094 GLOBAL
P 034D              1095 dascwrd PROCEDURE
                    1096 !*****************************************************
                    1097  Purpose =       To convert a variable length decimal
                    1098                  ASCII string to signed binary.
                    1099
                    1100  Input =         RR14 = address of source ASCII
                    1101                         string (in reg/ext/ser memory).
                    1102
                    1103  Output =        RR12 = binary word
                    1104                  R8,R9,R10,R11 holds the packed BCD
                    1105                  version of the result.
                    1106                  Carry FLAG = 1 if input error
                    1107                                       (serial only)
                    1108                         (SER_flg indicates cause)
                    1109                                  or dest overflow
                    1110                            = 0 if no error
                    1111                  R14, R15 modified
                    1112
                    1113  Note =          The ASCII input string processing is
                    1114                  terminated with the occurrence of a
                    1115                  non-decimal ASCII character.
                    1116                  Decimal ASCII string may be no more
                    1117                  than 6 digits in length, else Carry
                    1118                  will be returned.
                    1119                  Post decimal digits are not included
                    1120                  in the binary result.
                    1121 *****************************************************!
                    1122 ENTRY
P 034D CC   03      1123         ld       R12,#3          !6 digits!
P 034F DC   08      1124         ld       R13,#8          !temp addr =!
P 0351 04   FD  ED  1125         add      R13,RP          !R8 thru R11!
P 0354 D6   0363'   1126         call     dascbcd         !convert to bcd!
P 0357 7B   F3      1127         jr       c,has_ex1       !error!
P 0359 EC   08      1128         ld       R14,#8
P 035B 04   FD  EE  1129         add      R14,RP
P 035E FC   03      1130         ld       R15,#3
P 0360 8D   0284'   1131         jp       bcdwrd          !convert to binary!
P 0363              1132 END     dascwrd
```

```
                        1134 CONSTANT
                        1135  dab_LEN           :=      R12
                        1136  dab_DST           :=      R13
                        1137 GLOBAL
P 0363                  1138 dascbcd PROCEDURE
                        1139 !*****************************************************
                        1140  Purpose =         To convert a variable length decimal
                        1141                    ASCII string to BCD.
                        1142
                        1143  Input =           R13 = address of destination BCD
                        1144                        string (in register memory).
                        1145                    RR14 = address of source ASCII
                        1146                           string (in reg/ext/ser memory).
                        1147                    R12 = BCD digit count / 2
                        1148
                        1149  Output =          BCD string in designated destination
                        1150                    buffer (any overflow high order
                        1151                    digits are truncated without error).
                        1152                    Carry FLAG = 1 if input error
                        1153                                          (serial only)
                        1154                           (SER_flg indicates cause)
                        1155                                 or overflow
                        1156                    R14, R15 modified.
                        1157
                        1158  Note =            The ASCII input string processing is
                        1159                    terminated with the occurrence of a
                        1160                    non-decimal ASCII character.
                        1161 *****************************************************!
                        1162 ENTRY
P 0363 70  EC           1163          push    dab_LEN           !save!
P 0365 70  ED           1164          push    dab_DST
P 0367 B1  ED           1165 das_g1:  clr     @dab_DST          !init. destination!
P 0369 DE               1166          inc     dab_DST
P 036A CA  FB           1167          djnz    dab_LEN,das_g1
P 036C B1  ED           1168          clr     @dab_DST          !init.!
P 036E 50  ED           1169          pop     dab_DST           !restore!
P 0370 50  EC           1170          pop     dab_LEN
P 0372 E6  7E  01       1171          ld      TEMP_3,#1         !for ver_asc!
P 0375 B0  7B           1172          clr     TEMP_4            !bit 0 => digit seen;
                        1173                                     bit 1 => dec pt seen;
                        1174                                     bit 7 => overflow!
P 0377 D6  03DA'        1175 das_g2:  call    get_src           !get input byte!
P 037A 7B  41           1176          jr      c,dab_ex1         !serial error!
P 037C 56  7C  7F       1177          and     TEMP_1,#%7F       !7-bit ASCII!
P 037F 76  7B  03       1178          tm      TEMP_4,#%03       !check status!
P 0382 EB  0F           1179          jr      nz,das_g5         !sign char not valid!
P 0384 A6  7C  2B       1180          cp      TEMP_1,#'+'       !positive?!
P 0387 6B  EE           1181          jr      z,das_g2          !yes. no affect!
P 0389 A6  7C  2D       1182          cp      TEMP_1,#'-'       !negative?!
P 038C EB  07           1183          jr      nz,das_g4         !not sign char!
P 038E B7  ED  80       1184          xor     @dab_DST,#%80     !complement sign!
P 0391 8B  E4           1185          jr      das_g2            !get next input!
P 0393 5B  0A           1186 das_g5:  jr      mi,das_g6         !dec pt has been seen!
P 0395 A6  7C  2E       1187 das_g4:  cp      TEMP_1,#'.'       !is char dec pt?!
P 0398 EB  05           1188          jr      nz,das_g6         !nope.!
P 039A 46  7B  03       1189          or      TEMP_4,#%03       !dec pt and digit seen!
P 039D 8B  D8           1190          jr      das_g2            !get next input!
P 039F D6  040D'        1191 das_g6:  call    ver_asc           !is bcd digit?!
P 03A2 7B  16           1192          jr      c,dab_ex          !end conversion.!
P 03A4 46  7B  01       1193          or      TEMP_4,#%01       !digit seen!
P 03A7 D6  0463'        1194          call    rdl               !new digit to dest!
P 03AA EB  09           1195          jr      nz,das_g7         !overflow!
P 03AC 76  7B  02       1196          tm      TEMP_4,#%02       !post dec digit?!
P 03AF 6B  C6           1197          jr      z,das_g2          !no. get next input!
```

```
P 03B1 21   ED       1198          inc     @dab_DST          !inc post dec cnt!
P 03B3 8B   C2       1199          jr      das_g2            !get next input!
P 03B5 46   7B  80   1200 das_g7: or      TEMP_4,#%80       !set overflow!
P 03B8 8B   BD       1201          jr      das_g2            !get next input!
                     1202
P 03BA E4   7B  FC   1203 dab_ex: ld      FLAGS,TEMP_4      !carry = 0 or 1!
P 03BD AF            1204 dab_ex1: ret
P 03BE               1205 END      dascbcd

                     1207 GLOBAL
P 03BE               1208 wrddasc PROCEDURE
                     1209 !*************************************************
                     1210  Purpose =       To convert a signed binary word to
                     1211                  decimal ASCII
                     1212
                     1213  Input =         RR12 = source binary word.
                     1214                  RR14 = address of dest (in reg/ext/ser
                     1215                         memory).
                     1216
                     1217  Output =        Decimal ASCII in dest buffer.
                     1218                  R8,R9,R10,R11 holds the packed BCD
                     1219                  version of the result.
                     1220                  R12, R13, R14, R15 modified.
                     1221 *************************************************!
                     1222 ENTRY
P 03BE 70   EE       1223          push    R14
P 03C0 70   EF       1224          push    R15               !save dest addr!
P 03C2 EC   08       1225          ld      R14,#8
P 03C4 04   FD  EE   1226          add     R14,RP            !R8,9,10 & 11 temp!
P 03C7 FC   03       1227          ld      R15,#3            !temp byte length!
P 03C9 D6   02CD'    1228          call    wrdbcd            !convert input word!
P 03CC 50   EF       1229          pop     R15
P 03CE 50   EE       1230          pop     R14               !restore dest addr!
P 03D0 CC   03       1231          ld      R12,#3            !length of temp!
P 03D2 DC   08       1232          ld      R13,#8
P 03D4 04   FD  ED   1233          add     R13,RP            !addr of temp!
P 03D7 8D   0205'    1234          jp      bcddasc           !convert to ASCII!
P 03DA               1235 END      wrddasc
```

```
                        1237 GLOBAL          !for PART II only!
P 03DA                  1238 get_src PROCEDURE
                        1239 !*****************************************************
                        1240  Purpose =      To get source byte from
                        1241                 reg/ext/ser memory into TEMP_1.
                        1242
                        1243  Output =       Carry FLAG = 1 if error (serial)
                        1244                            = 0 if all ok
                        1245                 TEMP_1 = source byte.
                        1246                 RR14 updated.
                        1247 *****************************************************!
                        1248 ENTRY
P 03DA CF               1249          rcf                      !set good return code!
P 03DB EE               1250          inc     R14              !test R14 = 0!
P 03DC EA  06           1251          djnz    R14,get_s1       !src in ext memory!
P 03DE FE               1252          inc     R15              !test R15 = 0!
P 03DF FA  0E           1253          djnz    R15,get_s2       !src in reg memory!
P 03E1 8D  0000*        1254          jp      ser_get          !src in ser memory!
P 03E4 70  EB           1255 get_s1:  push    R11              !save user's!
P 03E6 82  BE           1256          lde     R11,@RR14        !get byte!
P 03E8 B9  7C           1257          ld      TEMP_1,R11       !move to common!
P 03EA 50  EB           1258          pop     R11              !restore user's!
P 03EC A0  EE           1259          incw    RR14             !update src ptr!
P 03EE AF               1260          ret
P 03EF E5  EF  7C       1261 get_s2:  ld      TEMP_1,@R15      !get byte!
P 03F2 FE               1262          inc     R15              !update src ptr!
P 03F3 AF               1263          ret
P 03F4                  1264 END      get_src
                        1265
                        1266 GLOBAL          !for PART II only!
P 03F4                  1267 put_dest        PROCEDURE
                        1268 !*****************************************************
                        1269  Purpose =      To store destination byte from TEMP_1
                        1270                 into reg/ext/ser memory
                        1271
                        1272  Output =       RR14 updated.
                        1273 *****************************************************!
                        1274 ENTRY
P 03F4 EE               1275          inc     R14              !test R14 = 0!
P 03F5 EA  06           1276          djnz    R14,put_s1       !dest in ext memory!
P 03F7 FE               1277          inc     R15              !test R15 = 0!
P 03F8 FA  0E           1278          djnz    R15,put_s2       !dest in reg memory!
P 03FA 8D  0000*        1279          jp      ser_output       !dest in ser memory!
P 03FD 70  EB           1280 put_s1:  push    R11              !save user's!
P 03FF B8  7C           1281          ld      R11,TEMP_1
P 0401 92  BE           1282          lde     @RR14,R11
P 0403 50  EB           1283          pop     R11              !restore user's!
P 0405 A0  EE           1284          incw    RR14
P 0407 AF               1285          ret
P 0408 F5  7C  EF       1286 put_s2:  ld      @R15,TEMP_1
P 040B FE               1287          inc     R15
P 040C AF               1288          ret
P 040D                  1289 END      put_dest
```

```
                        1291 CONSTANT
                        1292  MODE              :=      TEMP_3
                        1293  char              :=      TEMP_1
                        1294 INTERNAL
P 040D                  1295 ver_asc PROCEDURE
                        1296 !*****************************************************
                        1297  Purpose =         To verify input character as valid
                        1298                    hex or decimal ASCII.
                        1299
                        1300  Input =           TEMP_1 = 8-bit input
                        1301                    TEMP_3 = 0 => test for hex,
                        1302                             1 => test for decimal
                        1303
                        1304  Output =          Carry FLAG = 0 if no error
                        1305                                 1 if error.
                        1306 *****************************************************!
                        1307 ENTRY
P 040D 56  7C  7F       1308          and       char,#%7F         !7-bit ASCII!
P 0410 A6  7C  30       1309          cp        char,#'0'         !range start: '0'!
P 0413 7B  16           1310          jr        ult,ver_err       !no good!
P 0415 A6  7C  3A       1311          cp        char,#'9'+1       !dec range end: '9'!
P 0418 7B  10           1312          jr        ult,ver_ok        !all's well!
P 041A 76  7E  01       1313          tm        MODE,#1           !dec or hex?!
P 041D EB  0B           1314          jr        nz,ver_erc        !no good!
P 041F 56  7C  DF       1315          and       char,#LNOT('a'-'A') !insure upper case!
P 0422 A6  7C  41       1316          cp        char,#'A'         !check A-F range!
P 0425 7B  04           1317          jr        ult,ver_err       !no good!
P 0427 A6  7C  47       1318          cp        char,#'F'+1       !end hex range!
                        1319 ver_ok:
P 042A EF               1320 ver_erc: ccf                       !complement carry!
P 042B AF               1321 ver_err: ret
P 042C                  1322 END      ver_asc

                        1324 INTERNAL
P 042C                  1325 bcd_bin PROCEDURE
                        1326 !*****************************************************
                        1327  Purpose =         To convert next bcd digit to binary.
                        1328
                        1329  Input =           TEMP_1 = digit
                        1330
                        1331  Output =          RR12 = RR12 * 10 + digit
                        1332 *****************************************************!
                        1333 ENTRY
P 042C 56  7C  0F       1334          and       TEMP_1,#%0F       !isolate digit!
P 042F A6  7C  09       1335          cp        TEMP_1,#9         !verify valid!
P 0432 BB  2D           1336          jr        ugt,bcd_b1        !error!
P 0434 02  DD           1337          add       R13,R13
P 0436 12  CC           1338          adc       R12,R12           !2x!
P 0438 7B  27           1339          jr        c,bcd_b1          !overflow!
P 043A 70  EC           1340          push      R12
P 043C 70  ED           1341          push      R13
P 043E 02  DD           1342          add       R13,R13
P 0440 12  CC           1343          adc       R12,R12           !4x!
P 0442 7B  19           1344          jr        c,bcd_b2          !overflow!
P 0444 02  DD           1345          add       R13,R13
P 0446 12  CC           1346          adc       R12,R12           !8x!
P 0448 7B  13           1347          jr        c,bcd_b2          !overflow!
P 044A 04  7C  ED       1348          add       R13,TEMP_1
P 044D 16  EC  00       1349          adc       R12,#0            !8x + d!
P 0450 7B  0B           1350          jr        c,bcd_b2          !overflow!
P 0452 50  7C           1351          pop       TEMP_1
P 0454 04  7C  ED       1352          add       R13,TEMP_1
P 0457 50  7C           1353          pop       TEMP_1
P 0459 14  7C  EC       1354          adc       R12,TEMP_1        !10x + d!
P 045C AF               1355          ret
                        1356
P 045D 50  7C           1357 bcd_b2:  pop       TEMP_1
P 045F 50  7C           1358          pop       TEMP_1            !restore stack!
P 0461 DF               1359 bcd_b1:  scf                       !error!
P 0462 AF               1360          ret
P 0463                  1361 END      bcd_bin
```

```
                    1363 CONSTANT
                    1364  s_len           :=      R12
                    1365  s_adr           :=      R13
                    1366 INTERNAL
P 0463              1367 rdl      PROCEDURE
                    1368 !*****************************************************
                    1369  Rotate Digit Left
                    1370
                    1371  Input =         R12 = BCD string length
                    1372                  R13 = BCD string address
                    1373                  TEMP_1 bit 3-0 = new digit
                    1374
                    1375  Output =        BCD string rotated left one digit;
                    1376                  new digit inserted in units position.
                    1377                  TEMP_1 bit 3-0 = digit rotated out
                    1378                         of high order digit position
                    1379                       bit 7-4 = 0
                    1380                  Zero FLAG = 1 if TEMP_1 <> 0
                    1381                  R12, R13 unmodified
                    1382 *****************************************************!
                    1383 ENTRY
P 0463 70  EC       1384          push    s_len
P 0465 02  DC       1385          add     s_adr,s_len     !address of units place!
P 0467 F1  ED       1386 rdl_01:  swap    @s_adr
P 0469 E5  ED  7D   1387          ld      TEMP_2,@s_adr
P 046C 57  ED  F0   1388          and     @s_adr,#%F0     !isolate digit!
P 046F 56  7C  0F   1389          and     TEMP_1,#%0F     !isolate new digit!
P 0472 45  ED  7C   1390          or      TEMP_1,@s_adr
P 0475 F5  7C  ED   1391          ld      @s_adr,TEMP_1   !save new byte!
P 0478 E4  7D  7C   1392          ld      TEMP_1,TEMP_2
P 047B 00  ED       1393          dec     s_adr           !back-up pointer!
P 047D CA  E8       1394          djnz    s_len,rdl_01    !loop till done!
P 047F 56  7C  0F   1395          and     TEMP_1,#%0F     !old high order digit!
P 0482 50  EC       1396          pop     s_len           !restore R12!
P 0484 AF           1397          ret
P 0485              1398 END      rdl

                    1400 INTERNAL
P 0485              1401 rdr      PROCEDURE
                    1402 !*****************************************************
                    1403  Rotate Digit Right
                    1404
                    1405  Input =         R12 = BCD string length
                    1406                  R13 = BCD string address
                    1407                  TEMP_1 bit 7-4 = new digit
                    1408
                    1409  Output =        BCD string rotated right one digit;
                    1410                  new digit inserted in high order
                    1411                  position.
                    1412                  R12 unmodified
                    1413                  R13 modified
                    1414 *****************************************************!
                    1415 ENTRY
P 0485 70  EC       1416          push    s_len
P 0487 DE           1417 rdr_01:  inc     s_adr
P 0488 F1  ED       1418          swap    @s_adr
P 048A E5  ED  7E   1419          ld      TEMP_3,@s_adr
P 048D 57  ED  0F   1420          and     @s_adr,#%0F     !isolate digit!
P 0490 56  7C  F0   1421          and     TEMP_1,#%F0     !isolate new digit!
P 0493 45  ED  7C   1422          or      TEMP_1,@s_adr
P 0496 F5  7C  ED   1423          ld      @s_adr,TEMP_1   !save new byte!
P 0499 E4  7E  7C   1424          ld      TEMP_1,TEMP_3
P 049C CA  E9       1425          djnz    s_len,rdr_01    !loop till done!
P 049E 50  EC       1426          pop     s_len           !restore R12!
P 04A0 AF           1427          ret
P 04A1              1428 END      rdr
```

```
                    1460 CONSTANT
                    1461  tjm_bits        :=      R12
                    1462  tjm_mask        :=      R13
                    1463 GLOBAL
P 04A1              1464 clb      PROCEDURE
                    1465 !*****************************************************
                    1466  Purpose =       To collect selected bits in a byte
                    1467                  into adjacent bits in the low order
                    1468                  end of the byte.  Upper bits in byte
                    1469                  are set to zero.
                    1470
                    1471  Input =         R12 = input byte
                    1472                  R13 = mask. Bit = 1 => corresponding
                    1473                        input bit is selected.
                    1474
                    1475  Output =        R12 = collected bits
                    1476
                    1477  Note =          For example:
                    1478                  Input :  R12 = %(2)01110110
                    1479                           R13 = %(2)10000101
                    1480
                    1481                  Output : R12 = %(2)00000010
                    1482 *****************************************************!
                    1483 ENTRY
P 04A1 E6  7C  08   1484          ld      TEMP_1,#8       !bit count!
P 04A4 B0  7D       1485          clr     TEMP_2          !bits collected here!
P 04A6 90  EC       1486 next1:   rl      tjm_bits        !bit 7 to bit 0!
P 04A8 90  ED       1487          rl      tjm_mask        !bit 7 to carry!
P 04AA FB  06       1488          jr      nc,no_select    !don't use this bit!
P 04AC E0  EC       1489          rr      tjm_bits
P 04AE 90  EC       1490          rl      tjm_bits        !bit 7 to 0 and carry!
P 04B0 10  7D       1491          rlc     TEMP_2          !collect source bit!
                    1492 no_select:
P 04B2 00  7C       1493          dec     TEMP_1
P 04B4 EB  F0       1494          jr      nz,next1        !repeat!
P 04B6 C8  7D       1495          ld      R12,TEMP_2
P 04B8 AF           1496          ret
P 04B9              1497 END      clb
```

```
                    1499 CONSTANT
                    1500  tjm_tabh        :=      R14
                    1501  tjm_tabl        :=      R15
                    1502  tjm_tab         :=      RR14
                    1503 GLOBAL
P 04B9              1504 tjm      PROCEDURE
                    1505 !*****************************************************
                    1506  Purpose =       To take a jump to a routine address
                    1507                  determined by the state of selected
                    1508                  bits in a source byte.  A bit
                    1509                  is 'selected' by a one in the
                    1510                  corresponding position of a mask.
                    1511                  The 'selected' bits are packed into
                    1512                  adjacent bits in the low order end of
                    1513                  the byte.  This value is then doubled,
                    1514                  and used as an index into the jump
                    1515                  table.
                    1516
                    1517  Input =         RR14 = address of jump table in
                    1518                         program memory.
                    1519                  R12 = input data
                    1520                  R13 = mask
                    1521 *****************************************************!
                    1522 ENTRY
P 04B9 D6  04A1'    1523          call    clb             !collect selected bits!
P 04BC 02  CC       1524          add     tjm_bits,tjm_bits !collected bits * 2!
P 04BE 16  EE  00   1525          adc     tjm_tabh,#0     !in case carry!
P 04C1 02  FC       1526          add     tjm_tabl,tjm_bits
P 04C3 16  EE  00   1527          adc     tjm_tabh,#0     !tjm_tab points to...!
P 04C6 C2  DE       1528          ldc     tjm_mask,@tjm_tab !...table entry!
P 04C8 A0  EE       1529          incw    tjm_tab
P 04CA C2  FE       1530          ldc     tjm_tabl,@tjm_tab !get table entry...!
P 04CC E8  ED       1531          ld      tjm_tabh,tjm_mask !...into tjm_tab!
                    1532
P 04CE 30  EE       1533          jp      @tjm_tab        !bye!
                    1534
P 04D0              1535 END      tjm
                    1536 END PART_I

   0 errors
Assembly complete
```

```
Z8ASM     3.02
LOC     OBJ CODE     STMT SOURCE STATEMENT

                       1
                       2
                       3 PART_II MODULE
                       4
                       5
                       6 !'ROMLESS Z8'   SUBROUTINE LIBRARY  PART II
                       7 !
                       9 CONSTANT
                      10 !Register Usage!
                      11
                      12 RAM_START        :=      %7F
                      13
                      14 P3M_save         :=      RAM_START
                      15 TEMP_3           :=      P3M_save-1
                      16 TEMP_2           :=      TEMP_3-1
                      17 TEMP_1           :=      TEMP_2-1
                      18 TEMP_4           :=      TEMP_1-1
                      19
                      20 !The following registers are modified/referenced
                      21  by the Serial Routines ONLY.  They are
                      22  available as general registers to the user
                      23  who does not intend to make use of the
                      24  Serial Routines!
                      25
                      26 SER_char         :=      TEMP_4-1
                      27 SER_tmp2         :=      SER_char-1
                      28 SER_tmp1         :=      SER_tmp2-1
                      29 SER_put          :=      SER_tmp1-1
                      30 SER_len          :=      SER_put-1
                      31 SER_buf          :=      SER_len-2
                      32 SER_imr          :=      SER_buf-1
                      33 SER_cfg          :=      SER_imr-1
                      34 !Serial Configuration Data
                      35 bit 7 : =1 => odd parity on
                      36 bit 6 : =1 => even parity on
                      37  (bit 6,7 = 11 => undefined)
                      38 bit 5 : undefined
                      39 bit 4 : undefined
                      40 bit 3 : =1 => input editting on
                      41 bit 2 : =1 => auto line feed enabled
                      42 bit 1 : =1 => BREAK detection enabled
                      43 bit 0 : =1 => input echo on
                      44 !
                      45 op       :=      %80
                      46 ep       :=      %40
                      47 ie       :=      %08
                      48 al       :=      %04
                      49 be       :=      %02
                      50 ec       :=      %01
                      51 SER_get          :=      SER_cfg-1
                      52 SER_flg          :=      SER_get-1
                      53 !Serial Status Flags
                      54 bit 7 : =1 => serial I/O disabled
                      55 bit 6 : undefined
                      56 bit 5 : undefined
                      57 bit 4 : =1 => parity error
                      58 bit 3 : =1 => BREAK detected
                      59 bit 2 : =1 => input buffer overflow
                      60 bit 1 : =1 => input buffer not empty
                      61 bit 0 : =1 => input buffer full
                      62 !
                      63 sd       :=      %80
                      64 pe       :=      %10
                      65 bd       :=      %08
                      66 bo       :=      %04
                      67 bne      :=      %02
                      68 bf       :=      %01
                      69
```

```
 70 RAM_TMR          :=      RAM_START-%10
 71
 72 SER1time         :=      SER flg-1
 73 SERhtime         :=      SER1time-1
 74
 75 !The following registers are modified/referenced
 76  by the Timer/Counter Routines ONLY.  They are
 77  available as general registers to the user
 78  who does not intend to make use of the
 79  Timer/Counter Routines!
 80
 81 TOD_tic          :=      RAM_TMR-2
 82 TOD_imr          :=      TOD_tic-1
 83 TOD_hr           :=      TOD_imr-1
 84 TOD_min          :=      TOD_hr-1
 85 TOD_sec          :=      TOD_min-1
 86 TOD_tt           :=      TOD_sec-1
 87 PLS_1            :=      TOD_tt-1
 88 PLS_tmr          :=      PLS_1-1
 89 PLS_2            :=      PLS_tmr-1
 90
 91 RAM_END          :=      PLS_2
 92 STACK            :=      RAM_END
 93
 94 !Equivalent working register equates
 95  for above register layout!
 96
 97 !register file %70 - %7F!
 98 RAM_STARTr       :=      %70     !for SRP!
 99
100 rP3Msave         :=      R15
101 rTEMP_3          :=      R14
102 rTEMP_2          :=      R13
103 rTEMP_1          :=      R12
104 rrTEMP_1         :=      RR12
105 rTEMP_1h         :=      R12
106 rTEMP_1l         :=      R13
107 rTEMP_4          :=      R11
108 rSERchar         :=      R10
109 rSERtmp2         :=      R9
110 rSERtmp1         :=      R8
111 rrSERtmp         :=      RR8
112 rSERtmpl         :=      R9
113 rSERtmph         :=      R8
114 rSERput          :=      R7
115 rSERlen          :=      R6
116 rrSERbuf         :=      RR4
117 rSERbufh         :=      R4
118 rSERbufl         :=      R5
119 rSERimr          :=      R3
120 rSERcfg          :=      R2
121 rSERget          :=      R1
122 rSERflg          :=      R0
123
124
125 !register file %60 - %6F!
126 RAM_TMRr         :=      %60     !for SRP!
127 rTODtic          :=      R13
128 rTODimr          :=      R12
129 rTODhr           :=      R11
130 rTODmin          :=      R10
131 rTODsec          :=      R9
132 rTODtt           :=      R8
133 rPLS_1           :=      R7
134 rPLStmr          :=      R6
135 rPLS_2           :=      R5
```

```
                        164 CONSTANT
                        165  si_PTR          :=      RR14
                        166  si_TMP1         :=      R11
                        167  si_TMP2         :=      R13
                        168 GLOBAL
P 0000                  169 ser_init         PROCEDURE
                        170 !*****************************************************
                        171 serial initialize
                        172
                        173  Purpose =       To initialize the serial channel and
                        174                  RAM flags for serial I/O.  Serial
                        175                  input occurs under interrupt control.
                        176                  Serial output occurs in a polled mode.
                        177
                        178  Input =         RR14 = address of parameter list in
                        179                         program memory (if R14 = 0,
                        180                         use defaults):
                        181                   1 byte = Serial Configuration Data
                        182                         (see definition of SER_cfg)
                        183                   1 byte = IMR mask for nestable
                        184                         interrupts
                        185                   1 word = address of circular input
                        186                         buffer (in reg/ext memory)
                        187                   1 byte = Length of input buffer
                        188                   1 byte = Baud rate counter value
                        189                   1 byte = Baud rate prescaler value
                        190                            (unshifted)
                        191
                        192  Output =        Serial I/O operations initialized.
                        193                  R11, R12, R13, R14, R15 modified.
                        194
                        195  Note =          Defaults:
                        196                   Input echo on
                        197                   Input editting on
                        198                   BREAK detection enabled
                        199                   No parity
                        200                   Auto line feed on
                        201                   Input Buffer Address = SER_char
                        202                   Input buffer length = 1 byte
                        203                   Baud Rate = 9600 (assuming
                        204                         XTAL = 7.3728 MHz)
                        205
                        206                  The instruction at %0809 must result
                        207                  in a jump to the jump table entry for
                        208                  ser_input.
                        209
                        210                  If BREAK detection is disabled, and a
                        211                  BREAK occurs, it will be received as a
                        212                  continuous string of null characters.
                        213
                        214                  The parameter list is not referenced
                        215                  following initialization.
                        216 *****************************************************!
                        217 ENTRY
P 0000 EE               218          inc     R14              !use defaults?!
P 0001 EA  04           219          djnz    R14,si_1         !no. given by caller.!
P 0003 EC  00*          220          ld      R14,#HI ser_def  !address of default...!
P 0005 FC  51*          221          ld      R15,#LO ser_def  !... parameter list.  !
P 0007 BC  72           222 si_1:    ld      si_TMP1,#SER_cfg
P 0009 DC  05           223          ld      si_TMP2,#5
P 000B C3  BE           224 si_2:    ldci    @si_TMP1,@si_PTR !get initialization...!
P 000D DA  FC           225          djnz    si_TMP2,si_2     !...parameters!
P 000F 56  73  F7       226          and     SER_imr,#%F7     !insure no self-nesting!
                        227
```

```
                        228 !initialize Port 3 Mode Register for serial I/O!
P 0012 56  F1  FC       229         AND     TMR,#%FC          !disable T0!
P 0015 B8  72           230         ld      si_TMP1,SER_cfg   !configuration data!
P 0017 56  EB  80       231         AND     si_TMP1,#%80      !odd parity select!
P 001A 46  EB  40       232         OR      si_TMP1,#%40      !P30/7 = Sin/Sout!
P 001D 56  7F  3F       233         AND     P3M_save,#%3F     !mask off old settings!
P 0020 44  EB  7F       234         OR      P3M_save,si_TMP1  !new selection!
P 0023 E4  7F  F7       235         LD      P3M,P3M_save      !to write-only register!
                        236
                        237 !initialize T0!
P 0026 BC  F4           238         ld      si_TMP1,#T0
P 0028 C2  DE           239         ldc     si_TMP2,@si_PTR !save counter!
P 002A C3  BE           240         ldci    @si_TMP1,@si_PTR !init counter!
P 002C C2  BE           241         ldc     si_TMP1,@si_PTR !get prescaler!
P 002E D6  0000*        242         call    multiply          !T0 x PRE0!
P 0031 C9  6E           243         ld      SERhtime,R12      !save for BREAK...!
P 0033 D9  6F           244         ld      SERltime,R13      !...detection     !
P 0035 90  EB           245         rl      si_TMP1           !SHL 1!
P 0037 DF               246         scf                       !continuous mode!
P 0038 10  EB           247         rlc     si_TMP1           !SHL 2!
P 003A B9  F5           248         ld      PRE0,si_TMP1
                        249 !initialize RAM flags and pointers!
P 003C 8F               250         DI                        !disable interrupts!
P 003D B0  71           251         clr     SER_get           !input buffer...!
P 003F B0  77           252         clr     SER_put           !...empty!
P 0041 B0  70           253         clr     SER_flg           !no errors!
                        254
                        255 !initialize interrupts!
P 0043 56  FA  E7       256         AND     IRQ,#%E7          !clear IRQ3 & 4!
P 0046 56  FB  EF       257         and     IMR,#%EF          !disable IRQ4 (xmt)!
P 0049 46  FB  08       258         or      IMR,#%08          !enable IRQ3 (rcv)!
P 004C 9F               259         EI
                        260 !go!
P 004D 46  F1  03       261         or      TMR,#%03          !load/enable T0!
P 0050 AF               262         ret
P 0051                  263 END     ser_init
                        264
                        265
                        266
                        267 !Defaults for serial initialization!
                        268
P 0051 0F  00           269 ser_def RECORD  [cfg_, imr_              BYTE
P 0053 007A  01
P 0056 02  03
                        270                  buf_                    WORD
                        271                  len_, ctr_, pre_        BYTE]
                        272         :=
                        273         [ec+al+ie+be, %00, SER_char, 1, %02, %03]
```

```
                    275 CONSTANT
                    276  rli_len        :=      R13
                    277 GLOBAL
P 0058              278 ser_rlin        PROCEDURE
                    279 !***************************************************
                    280 read line
                    281
                    282  Purpose =      To return input from serial channel
                    283                 up to 'carriage return' character or
                    284                 maximum length requested or BREAK.
                    285
                    286  Input =        RR14 = address of destination buffer
                    287                        (in reg/ext memory)
                    288                 R13 = maximum length
                    289
                    290  Output =       Input characters is destination buffer.
                    291                 RR14 = unmodified
                    292                 R13 = length returned
                    293                 Carry Flag = 1 if any error,
                    294                            = 0 if no error.
                    295                 R12 indicates read status
                    296
                    297  Note =         1. Return will be made to the calling
                    298                 program only after the requisite
                    299                 characters have been received from
                    300                 the serial line.
                    301
                    302                 2. If input editting is enabled, a
                    303                 'backspace' character will cause
                    304                 the previous character (if any) in the
                    305                 the destination buffer to be deleted;
                    306                 a 'delete' character will cause all
                    307                 previous characters (if any) in the
                    308                 destination buffer to be deleted.
                    309
                    310                 3. If parity (odd or even) is enabled,
                    311                 the parity error flag (R14) will be set
                    312                 if any character returned had a parity
                    313                 error. (Bit 7 of each character may
                    314                 then be examined if it is desirable to
                    315                 know which character(s) had the error).
                    316
                    317                 4. The status flags 'BREAK detected',
                    318                 'parity error', and 'input buffer
                    319                 overflow' will be returned
                    320                 as part of R12, but will be cleared in
                    321                 SER_stat.
                    322
                    323                 5. The staus flags: 'input buffer full'
                    324                 and 'input buffer not empty' will be
                    325                 updated in SER_stat.
                    326 ***************************************************!
                    327 ENTRY
P 0058 B0  7E       328         clr     TEMP_3          !flag => read line!
                    329 ser_read:
P 005A 70  EE       330         push    R14             !save original...!
P 005C 70  EF       331         push    R15             !...dest. pointer!
P 005E 70  ED       332         push    rli_len         !...and length!
P 0060 D6  0170'    333 rli_4:  call    ser_get         !get input character!
P 0063 7B  48       334         jr      c,rli_3         !error!
P 0065 76  72  C0   335         tm      SER_cfg,#op LOR ep !parity enabled?!
P 0068 6B  08       336         jr      z,rli_1         !no!
P 006A 76  7C  80   337         tm      TEMP_1,#%80     !parity error?!
P 006D 6B  03       338         jr      z,rli_1         !no!
```

```
P 006F 46  70  10    339          or      SER_flg,#pe      !yes. set error flag!
P 0072 D6  0000*     340 rli_1:   call    put_dest         !store in buffer!
P 0075 A6  7E  00    341          cp      TEMP_3,#0        !read line?!
P 0078 EB  31        342          jr      nz,rli_2         !no!
P 007A 56  7C  7F    343          and     TEMP_1,#%7F      !ignore parity bit!
P 007D 76  72  08    344          tm      SER_cfg,#ie      !input editting on?!
P 0080 6B  21        345          jr      z,rli_9          !no.!
                     346 !input editting!
P 0082 A6  7C  7F    347          cp      TEMP_1,#%7F      !char = delete?!
P 0085 6B  3E        348          jr      z,rli_6          !yes!
P 0087 A6  7C  08    349          cp      TEMP_1,#%08      !char = backspace?!
P 008A EB  17        350          jr      nz,rli_9         !no. continue!
P 008C 50  7C        351          pop     TEMP_1           !get original length!
P 008E 70  7C        352          push    TEMP_1
P 0090 A4  ED  7C    353          cp      TEMP_1,rli_len   !any characters?!
P 0093 6B  30        354          jr      eq,rli_6         !none!
P 0095 DE            355          inc     rli_len          !undo last decrement!
P 0096 26  EF  02    356          sub     R15,#2           !backspace & previous!
P 0099 EE            357          inc     R14              !reg or ext mem?!
P 009A EA  02        358          djnz    R14,rli_7        !ext!
P 009C 8B  C2        359          jr      rli_4            !reg!
P 009E 36  EE  00    360 rli_7:   sbc     R14,#0
P 00A1 8B  BD        361          jr      rli_4
                     362
P 00A3 00  ED        363 rli_9:   dec     rli_len          !in case cr!
P 00A5 A6  7C  0D    364          cp      TEMP_1,#%0D      !carriage return?!
P 00A8 6B  03        365          jr      z,rli_3          !end input!
P 00AA DE            366          inc     rli_len          !restore!
P 00AB DA  B3        367 rli_2:   djnz    rli_len,rli_4    !loop for max length!
P 00AD 50  7C        368 rli_3:   pop     TEMP_1           !original length!
P 00AF 24  ED  7C    369          sub     TEMP_1,rli_len   !# chars returned!
P 00B2 D8  7C        370          ld      rli_len,TEMP_1   !tell caller!
P 00B4 C8  70        371          ld      R12,SER_flg      !return read status!
P 00B6 56  70  E3    372          and     SER_flg,#LNOT (pe LOR bd LOR bo)
                     373                                   !reset for next time!
P 00B9 CF            374          rcf                      !good return code!
P 00BA 76  EC  9C    375          tm      R12,#pe LOR bd LOR bo LOR sd
P 00BD 6B  01        376          jr      z,rli_5          !no error!
P 00BF DF            377          scf                      !set error return!
P 00C0 50  EF        378 rli_5:   pop     R15
P 00C2 50  EE        379          pop     R14              !original buffer addr!
P 00C4 AF            380          ret
                     381
P 00C5 50  ED        382 rli_6:   pop     rli_len
P 00C7 50  EF        383          pop     R15
P 00C9 50  EE        384          pop     R14
P 00CB 8B  8D        385          jr      ser_read         !start over!
P 00CD               386 END      ser_rlin
                     388 GLOBAL
P 00CD               389 ser_rabs         PROCEDURE
                     390 !*****************************************************
                     391 read absolute
                     392
                     393  Purpose =       To return input from serial channel
                     394                  of maximum length requested.  (Input
                     395                  is not terminated with the receipt of
                     396                  a 'carriage return'.  BREAK will
                     397                  terminate read.)
                     398
                     399  Note =          All other details are as for 'ser_rlin'.
                     400 *****************************************************!
                     401 ENTRY
P 00CD E6  7E  01    402          ld      TEMP_3,#1        !flag => read absolute!
P 00D0 8B  88        403          jr      ser_read
P 00D2               404 END      ser_rabs
```

```
                        406 GLOBAL
P 00D2                  407 ser_input       PROCEDURE
                        408 !***************************************************
                        409 Interrupt service - Serial Input
                        410
                        411  Purpose =      To service IRQ3 by inputting current
                        412                 character into next available position
                        413                 in circular buffer.
                        414
                        415  Input =        None.
                        416
                        417  Output =       New character inserted in buffer.
                        418                 SER_stat , SER_put updated.
                        419
                        420  Note =         1. If even parity enabled, the software
                        421                 replaces the eigth data bit with a
                        422                 parity error flag.
                        423
                        424                 2. If BREAK detection is enabled, and
                        425                 the received character is null,
                        426                 the serial input line is monitored to
                        427                 detect a potential BREAK condition.
                        428                 BREAK is defined as a zero start bit
                        429                 followed by 8 zero data bits and a
                        430                 zero stop bit.
                        431
                        432                 3. If 'buffer full' on entry, 'input
                        433                 buffer overflow' is flagged.
                        434
                        435                 4. If input echo is on, the character is
                        436                 immediately sent to the output serial
                        437                 channel.
                        438
                        439                 5. IMR is modified to allow selected
                        440                 nested interrupts (see ser_init).
                        441 ***************************************************!
                        442 ENTRY
P 00D2 E4  03  78       443         ld      SER_tmp1,%03    !read stop bit level!
P 00D5 70  FB           444         push    imr             !save entry imr!
P 00D7 54  73  FB       445         and     imr,SER_imr     !allow nesting!
P 00DA 9F               446         ei
P 00DB 70  FD           447         push    rp              !save user's!
P 00DD 31  70           448         srp     #RAM_STARTr
P 00DF A8  F0           449         ld      rSERchar,SIO    !capture input!
P 00E1 76  E2  02       450         tm      rSERcfg,#be     !break detect enabled?!
P 00E4 6B  2F           451         jr      z,ser_30        !nope.!
P 00E6 B0  E9           452         clr     rSERtmp2
P 00E8 76  E2  80       453         tm      rSERcfg,#op     !odd parity enabled?!
P 00EB 6B  02           454         jr      z,ser_23        !no.!
P 00ED 9C  80           455         ld      rSERtmp2,#%80
P 00EF A2  A9           456 ser_23: cp      rSERchar,rSERtmp2 !8 received bits = 0?!
P 00F1 EB  22           457         jr      ne,ser_30       !no!
P 00F3 76  E8  01       458         tm      rSERtmp1,#1     !test stop bit!
P 00F6 EB  1D           459         jr      nz,ser_30       !not BREAK!
                        460 !is BREAK. Wait for marking!
P 00F8 46  E0  08       461         or      rSERflg,#bd     !set BREAK flag!
P 00FB 76  03  01       462 ser_24: tm      %03,#1          !marking yet?!
P 00FE 6B  FB           463         jr      z,ser_24        !not yet!
                        464 !wait 1 char time to flush receive shift register!
P 0100 70  6E           465         push    SERhtime
P 0102 70  6F           466         push    SERltime        !save PRE0 x T0!
P 0104 8C  35           467 in_loop: ld     rSERtmp1,#53
P 0106 8A  FE           468 lp1:    djnz    rSERtmp1,lp1    !delay 640 cycles!
P 0108 80  6E           469         decw    SERhtime
```

```
P 010A EB  F8       470          jr       nz,in_loop       !delay (128x10xPREOxTO)!
                    471                                    !      ----------------!
                    472                                    !              2       !
P 010C 50  6F       473          pop      SERltime
P 010E 50  6E       474          pop      SERhtime         !restore PREO x TO!
P 0110 56  FA  F7   475          and      IRQ,#LNOT %08    !clear int req!
P 0113 8B  49       476          jr       ser_i5           !bye!
                    477
P 0115 76  E0  01   478 ser_30:  tm       rSERflg,#bf      !buffer full?!
P 0118 EB  4A       479          jr       nz,ser_i1        !yes.overflow!
P 011A 76  E2  01   480          tm       rSERcfg,#ec      !echo on?!
P 011D 6B  0A       481          jr       z,ser_i0         !no!
P 011F A9  F0       482          ld       SIO,rSERchar     !echo!
P 0121 66  FA  10   483 ser_i6:  tcm      IRQ,#%10         !poll!
P 0124 EB  FB       484          jr       nz,ser_i6        !loop!
P 0126 56  FA  EF   485          and      IRQ,#LNOT %10    !clear irq bit!
P 0129 76  E2  40   486 ser_i0:  tm       rSERcfg,#ep      !even parity?!
P 012C 6B  14       487          jr       z,ser_22         !no parity!
                    488 !calculate parity error flag!
P 012E 8C  07       489          ld       rSERtmp1,#7
P 0130 B0  E9       490          clr      rSERtmp2         !count 1's here!
P 0132 C0  EA       491 ser_20:  rrc      rSERchar         !bit to carry!
P 0134 16  E9  00   492          adc      rSERtmp2,#0      !update 1's count!
P 0137 8A  F9       493          djnz     rSERtmp1,ser_20  !loop till done!
P 0139 56  E9  01   494          and      rSERtmp2,#1      !1's count even or odd?!
P 013C B2  A9       495          xor      rSERchar,rSERtmp2
P 013E C0  EA       496          rrc      rSERchar         !parity error flag...!
P 0140 C0  EA       497          rrc      rSERchar         !...to bit 7!
P 0142 88  E4       498 ser_22:  ld       rSERtmph,rSERbufh
P 0144 98  E5       499          ld       rSERtmpl,rSERbufl
P 0146 02  97       500          add      rSERtmpl,rSERput !next char address!
P 0148 8E           501          inc      rSERtmph         !in external memory?!
P 0149 8A  1E       502          djnz     rSERtmph,ser_i2  !yes.!
P 014B F3  9A       503          ld       @rSERtmpl,rSERchar !store char in buf!
P 014D 46  E0  02   504 ser_i3:  or       rSERflg,#bne     !buffer not empty!
P 0150 7E           505          inc      rSERput          !update put ptr!
P 0151 A2  76       506          cp       rSERput,rSERlen  !wrap-around?!
P 0153 EB  02       507          jr       ne,ser_i4        !no!
P 0155 B0  E7       508          clr      rSERput          !set to start!
P 0157 A2  71       509 ser_i4:  cp       rSERput,rSERget  !if equal, then full!
P 0159 EB  03       510          jr       ne,ser_i5
P 015B 46  E0  01   511          or       rSERflg,#bf
P 015E 50  FD       512 ser_i5:  pop      rp               !restore user's!
P 0160 8F           513          di
P 0161 50  FB       514          pop      imr              !restore entry imr!
P 0163 BF           515          iret
                    516
P 0164 46  E0  04   517 ser_i1:  or       rSERflg,#bo      !buffer overflow!
P 0167 8B  F5       518          jr       ser_i5
                    519
P 0169 16  E8  00   520 ser_i2:  adc      rSERtmph,#0
P 016C 92  A8       521          lde      @rrSERtmp,rSERchar !store in buf!
P 016E 8B  DD       522          jr       ser_i3
P 0170              523 END      ser_input
```

```
                        525 GLOBAL          !for PART I!
P 0170                  526 ser_get PROCEDURE
                        527 !***************************************************
                        528  Purpose =      To return one serial input character.
                        529
                        530  Input =        None.
                        531
                        532  Output =       Carry FLAG = 1 if BREAK detected or
                        533                                 serial not enabled
                        534                                 or buffer overflow
                        535                         = 0 otherwise
                        536                 TEMP_1 = character
                        537
                        538  Note =         This routine will not return control
                        539                 until a character is available in the
                        540                 input buffer or an error is detected.
                        541 ***************************************************!
                        542 ENTRY
P 0170 70  FD           543         push    rp              !save caller's rp!
P 0172 31  70           544         srp     #RAM_STARTr     !point to subr. RAM!
P 0174 DF               545         scf                     !in case error!
P 0175 76  E0  8C       546 ser_g1: tm      rSERflg,#sd LOR bd LOR bo
                        547                                 !serial disabled or
                        548                                  BREAK detected or
                        549                                  buffer overflow?!
P 0178 EB  24           550         jr      nz,ser_g6       !yes.!
P 017A 76  E0  02       551         tm      rSERflg,#bne    !buffer not empty?!
P 017D 6B  F6           552         jr      z,ser_g1        !empty. wait!
P 017F D8  E5           553         ld      rTEMP_1l,rSERbufl
P 0181 C8  E4           554         ld      rTEMP_1h,rSERbufh
P 0183 8F               555         di                      !prevent IRQ3 conflict!
P 0184 02  D1           556         add     rTEMP_1l,rSERget !next char address!
P 0186 CE               557         inc     rTEMP_1h        !input buffer in...!
P 0187 CA  18           558         djnz    rTEMP_1h,ser_g3 !...external memory!
                        559                                 !...register memory!
P 0189 E3  CD           560         ld      rTEMP_1,@rTEMP_1l !get char!
P 018B 56  E0  FE       561 ser_g4: and     rSERflg,#LNOT bf !buffer not full!
P 018E 1E               562         inc     rSERget         !update get pointer!
P 018F A2  16           563         cp      rSERget,rSERlen !wrap-around?!
P 0191 EB  02           564         jr      ne,ser_g2       !no.!
P 0193 B0  E1           565         clr     rSERget         !yes. set to start!
P 0195 A2  17           566 ser_g2: cp      rSERget,rSERput !buffer empty if get...!
P 0197 EB  03           567         jr      ne,ser_g5       !...and put =!
P 0199 56  E0  FD       568         and     rSERflg,#LNOT bne !buffer empty now!
P 019C CF               569 ser_g5: rcf                     !set good return!
P 019D 9F               570         ei                      !re-enable interrupts!
P 019E 50  FD           571 ser_g6: pop     rp              !restore caller's rp!
P 01A0 AF               572         ret
                        573
P 01A1 16  EC  00       574 ser_g3: adc     rTEMP_1h,#0     !rrTEMP_1 has char addr!
P 01A4 82  CC           575         lde     rTEMP_1,@rrTEMP_1 !get char!
P 01A6 8B  E3           576         jr      ser_g4          !clean up!
P 01A8                  577 END     ser_get
```

```
                        579 GLOBAL
P 01A8                  580 ser_break       PROCEDURE
                        581 !***************************************************
                        582 break transmission
                        583
                        584  Purpose =      To transmit BREAK on the serial line.
                        585
                        586  Input =        RR14 = break length
                        587
                        588  Output =       None.
                        589
                        590  Note =         BREAK is defined as:
                        591                  serial out (P37) = 0 for
                        592                  2       x 28 cycles/loop x RR14 loops
                        593                  -----
                        594                  XTAL
                        595
                        596                  RR14 should yield at least 1 bit time
                        597                  so that the last 'clr SIO' will
                        598                  have been preceded by at least 1 bit
                        599                  time of spacing. Therefore, RR14 should
                        600                  be greater than or equal to
                        601
                        602                  4 x 16 x PREO x TO
                        603                  ------------------
                        604                          28
                        605 ***************************************************!
                        606 ENTRY
                        607 ser_b1:
P 01A8 B0  F0           608         clr     SIO
P 01AA 80  EE           609         decw    RR14
P 01AC EB  FA           610         jr      nz,ser_b1
                        611 !wait for last null to be fully transmitted!
P 01AE 8D  0238'        612         jp      ser_o1
P 01B1                  613 END     ser_break

                        615 GLOBAL
P 01B1                  616 ser_flush       PROCEDURE
                        617 !***************************************************
                        618 input flush
                        619
                        620  Purpose =      To flush (clear) the serial input
                        621                 buffer of characters.
                        622
                        623  Input =        None
                        624
                        625  Output =       Empty input buffer.
                        626
                        627  Note =         This routine might be useful to clear
                        628                 all past input after a BREAK has been
                        629                 detected on the line.
                        630 ***************************************************!
                        631 ENTRY
P 01B1 8F               632         di              !disable interrupts!
                        633                         !(to avoid collision with
                        634                          serial input)!
P 01B2 B0  71           635         clr     SER_get !buffer start!
P 01B4 B0  77           636         clr     SER_put != buffer end!
P 01B6 56  70  80       637         and     SER_flg,#%80    !clear status!
P 01B9 9F               638         ei              !re-enable interrupts!
P 01BA AF               639         ret
P 01BB                  640 END     ser_flush
```

```
                        642 CONSTANT
                        643  wli_len          :=      R13
                        644 GLOBAL
P 01BB                  645 ser_wlin          PROCEDURE
                        646 !*****************************************************
                        647 write line
                        648
                        649  Purpose =        To output a character string to serial
                        650                   line, ending with either a 'carriage
                        651                   return' character or the maximum length
                        652                   specified.
                        653
                        654  Input =          RR14 = address of source buffer
                        655                         (in reg/ext memory)
                        656                   R13 = length
                        657
                        658  Output =         RR14 = updated
                        659                   Carry Flag = 1 if serial not enabled,
                        660                              = 0 if no error.
                        661                   R13 = # bytes output (not including
                        662                                 auto line feed)
                        663
                        664  Note =           If auto line feed is enabled, a
                        665                   line feed character will be output
                        666                   following each carriage return
                        667                   (ser_wlin only).
                        668 *****************************************************!
                        669 ENTRY
P 01BB B0  7E           670          clr      TEMP_3              !flag => write line!
                        671
P 01BD DF               672 write:   scf                          !in case error!
P 01BE 76  70  80       673          tm       SER_flg,#sd         !serial disabled?!
P 01C1 EB  30           674          jr       nz,wli_1            !yes. error!
P 01C3 70  ED           675          push     wli_len
P 01C5 D6  0000*        676 wli_4:   call     get_src
P 01C8 D6  020B'        677          call     ser_output          !write the character!
P 01CB 7B  1E           678          jr       c,wli_2             !serial disabled!
P 01CD A6  7E  00       679          cp       TEMP_3,#0           !write line?!
P 01D0 EB  17           680          jr       nz,wli_5            !no, absolute.!
P 01D2 56  7C  7F       681          and      TEMP_1,#%7F         !mask off parity!
P 01D5 A6  7C  0D       682          cp       TEMP_1,#%0D         !line done?!
P 01D8 EB  0F           683          jr       nz,wli_5            !yes.!
P 01DA 00  ED           684          dec      wli_len
P 01DC 76  72  04       685          tm       SER_cfg,#al         !auto line feed?!
P 01DF 6B  0A           686          jr       z,wli_2             !disabled!
P 01E1 E6  7C  0A       687          ld       TEMP_1,#%0A         !output line feed!
P 01E4 D6  020B'        688          call     ser_output
P 01E7 8B  02           689          jr       wli_2
P 01E9 DA  DA           690 wli_5:   djnz     wli_len,wli_4       !loop!
P 01EB 50  7C           691 wli_2:   pop      TEMP_1              !original length!
P 01ED 24  ED  7C       692          sub      TEMP_1,wli_len
P 01F0 D8  7C           693          ld       wli_len,TEMP_1      !return output count!
P 01F2 CF               694          rcf                          !no error!
P 01F3 AF               695 wli_1:   ret
P 01F4                  696 END      ser_wlin
```

```
                        698 GLOBAL
P 01F4                  699 ser_wabs        PROCEDURE
                        700 !***************************************************
                        701 write absolute
                        702
                        703  Purpose =      To output a character string to serial
                        704                 line for the length specified.  (Output
                        705                 is not terminated with the output of
                        706                 a 'carriage return').
                        707
                        708  Note =         All other details are as for 'ser_wlin'.
                        709 ***************************************************!
                        710 ENTRY
P 01F4 E6  7E  01       711         ld      TEMP_3,#1
P 01F7 8B  C4           712         jr      write
P 01F9                  713 END     ser_wabs

P 01F9                  715 ser_wbyt        PROCEDURE
                        716 !***************************************************
                        717 write byte
                        718
                        719  Purpose =      To output a given character to the
                        720                 serial line. If the character is a
                        721                 carriage return and auto line feed
                        722                 is enabled, a line feed will be output
                        723                 as well.
                        724
                        725  Input =        R12 = character to output
                        726
                        727  Note =         Equivalent to ser_wlin with length = 1.
                        728 ***************************************************!
                        729 ENTRY
P 01F9 C9  7C           730         ld      TEMP_1,R12
P 01FB D6  020B'        731         call    ser_output      !output it!
P 01FE 76  72  04       732         tm      SER_cfg,#al     !auto line feed?!
P 0201 6B  3E           733         jr      z,ser_05        !not enabled!
P 0203 A6  EC  0D       734         cp      R12,#%0D        !char = car. ret?!
P 0206 EB  39           735         jr      nz,ser_05       !nope!
P 0208 E6  7C  0A       736         ld      TEMP_1,#%0A     !output line feed!
                        737 !fall into ser_output!
P 020B                  738 END     ser_wbyt
```

```
                        740 GLOBAL          !for PART I!
P 020B                  741 ser_output      PROCEDURE
                        742 !***************************************************
                        743  Purpose =      To output one character to the serial
                        744                 line.
                        745
                        746  Input =        TEMP_1 = character
                        747
                        748  Output =       Carry FLAG = 1 if serial disabled
                        749                           = 0 otherwise.
                        750
                        751  Note =         1. If even parity is enabled, the eigth
                        752                 data bit is modified prior to character
                        753                 output to SIO.
                        754
                        755                 2. IRQ4 is polled to wait for completion
                        756                 of character transmission before control
                        757                 returns to the calling program.
                        758 ***************************************************!
                        759 ENTRY
P 020B DF               760         scf                     !in case error!
P 020C 76  70  80       761         tm      SER_flg,#sd     !serial disabled?!
P 020F EB  30           762         jr      nz,ser_05       !yes. error!
P 0211 76  72  40       763         tm      SER_cfg,#ep     !even parity enabled?!
P 0214 6B  1F           764         jr      z,ser_o2        !no. just output!
                        765 !calculate parity!
P 0216 70  7E           766         push    TEMP_3
P 0218 E6  7E  07       767         ld      TEMP_3,#7
P 021B B0  7D           768         clr     TEMP_2
P 021D C0  7C           769 ser_04: rrc     TEMP_1          !character bit to carry!
P 021F 16  7D  00       770         adc     TEMP_2,#0       !count 1's!
P 0222 00  7E           771         dec     TEMP_3
P 0224 EB  F7           772         jr      nz,ser_04       !next bit!
P 0226 56  7D  01       773         and     TEMP_2,#01      !1's count odd/even!
P 0229 56  7C  FE       774         and     TEMP_1,#%FE
P 022C 44  7D  7C       775         or      TEMP_1,TEMP_2   !parity bit in D0!
P 022F C0  7C           776         rrc     TEMP_1
P 0231 C0  7C           777         rrc     TEMP_1          !parity bit in D7!
P 0233 50  7E           778         pop     TEMP_3
P 0235 E4  7C  F0       779 ser_o2: ld      SIO,TEMP_1      !output character!
P 0238 66  FA  10       780 ser_o1: tcm     IRQ,#%10        !check IRQ4!
P 023B EB  FB           781         jr      nz,ser_o1       !wait for complete!
P 023D 56  FA  EF       782         and     IRQ,#%EF        !clear IRQ4!
P 0240 CF               783         rcf                     !all ok!
P 0241 AF               784 ser_05: ret
P 0242                  785 END     ser_output

                        787 GLOBAL
P 0242                  788 ser_disable     PROCEDURE
                        789 !***************************************************
                        790 disable
                        791
                        792  Purpose =      To disable serial I/O operations.
                        793
                        794  Input =        None.
                        795
                        796  Output =       Serial I/O disabled.
                        797 ***************************************************!
                        798 ENTRY
P 0242 8F               799         di              !avoid IRQ3 conflict!
P 0243 46  70  80       800         or      SER_flg,#sd
                        801                         !set serial disabled!
P 0246 56  F1  FC       802         and     TMR,#%FC
                        803                         !disable T0!
P 0249 56  FB  E7       804         and     IMR,#%E7
                        805                         !disable IRQ3,4!
P 024C 56  7F  BF       806         and     P3M_save,#%BF
                        807                         !P30/7 normal i/o pins!
P 024F E4  7F  F7       808         ld      P3M,P3M_save
P 0252 9F               809         ei              !re-enable interrupts!
P 0253 AF               810         ret
P 0254                  811 END     ser_disable
```

```
                    840 CONSTANT
                    841  TMP    :=      R13
                    842  PTR    :=      RR14
                    843  PTRh   :=      R14
                    844 GLOBAL
P 0254              845 tod_i   PROCEDURE
                    846 !*************************************************
                    847 time of day : initialize
                    848
                    849  Purpose =      To initialize T0 or T1 to function as
                    850                 a time of day clock.
                    851
                    852  Input =        RR14 = address of parameter list in
                    853                        program memory:
                    854                  1 byte = IMR mask for nestable
                    855                        interrupts
                    856                  1 byte = # of clock ticks per second
                    857                  1 byte = counter # : = %F4 => T0
                    858                                     = %F2 => T1
                    859                  1 byte = Counter value
                    860                  1 byte = Prescaler value (unshifted)
                    861
                    862                 TOD_hr, TOD_min, TOD_sec, TOD_tt
                    863                  initialized to the starting time of
                    864                  hours, minutes, seconds, and ticks
                    865                  respectively.
                    866
                    867  Output =       Selected timer is loaded and
                    868                 enabled; corresponding interrupt
                    869                 is enabled.
                    870                 R13, R14, R15 modified.
                    871
                    872  Note =         The cntr and prescaler values provided
                    873                 are those values which will generate an
                    874                 interrupt (tick) the designated # of
                    875                 times per second.
                    876
                    877                 For example:
                    878                 for XTAL = 8 MHZ, cntr = 250 and
                    879                 prescaler = 40 yield a .01 sec interval;
                    880                 the 2nd byte of the parameter list
                    881                 should = 100 .
                    882
                    883                 For T0 the instruction at %080C or
                    884                 for T1 the instruction at %080F must
                    885                 result in a jump to the jump table entry
                    886                 for 'tod'.
                    887
                    888                 The parameter list is not referenced
                    889                 following initialization.
                    890 *************************************************!
                    891 ENTRY
P 0254 DC  6C       892         ld      TMP,#TOD_imr
P 0256 C3  DE       893         ldci    @TMP,@PTR       !imr mask!
P 0258 C3  DE       894         ldci    @TMP,@PTR       !ticks/second!
P 025A E6  7B  6C   895         ld      TEMP_4,#TOD_imr
P 025D 8D  02B2'    896         jp      pre_ctr         !ctr & prescaler!
P 0260              897 END     tod_i
```

```
                        899 GLOBAL
P 0260                  900 tod     PROCEDURE
                        901 !*****************************************************
                        902 Interrupt service - time of day
                        903
                        904  Purpose =      To update the time of day clock.
                        905 *****************************************************!
                        906 ENTRY
P 0260 70  FB           907         push    imr             !save entry imr!
P 0262 54  6C  FB       908         and     imr,TOD_imr     !allow nested interrupts
P 0265 9F               909         ei                      !enable interrupts!
P 0266 70  FD           910         push    rp              !save rp!
P 0268 31  60           911         srp     #RAM_TMRr       !point to our set!
P 026A 8E               912         inc     rTODtt          !ticks/second!
P 026B A2  8D           913         cp      rTODtt,rTODtic  !second complete?!
P 026D EB  13           914         jr      ne,tod_ex       !nope.!
P 026F B0  E8           915         clr     rTODtt
P 0271 9E               916         inc     rTODsec         !seconds!
P 0272 A6  E9  3C       917         cp      rTODsec,#60     !minute complete?!
P 0275 EB  0B           918         jr      ne,tod_ex       !nope.!
P 0277 B0  E9           919         clr     rTODsec
P 0279 AE               920         inc     rTODmin         !minutes!
P 027A A6  EA  3C       921         cp      rTODmin,#60     !hour complete?!
P 027D EB  03           922         jr      ne,tod_ex       !nope.!
P 027F B0  EA           923         clr     rTODmin
P 0281 BE               924         inc     rTODhr          !hours!
                        925
P 0282 50  FD           926 tod_ex: pop     rp              !restore rp!
P 0284 8F               927         di                      !disable interrupts!
P 0285 50  FB           928         pop     imr             !restore entry imr!
P 0287 BF               929         iret
P 0288                  930 END     tod
```

```
                        932 GLOBAL
P 0288                  933 pulse_i PROCEDURE
                        934 !*****************************************************
                        935  Purpose =      To initialize one of the timers
                        936                 to generate a variable frequency/
                        937                 variable pulse width output.
                        938
                        939  Input =        RR14 = address of parameter list in
                        940                       program memory:
                        941                  1 byte = cntr value for low interval
                        942                  1 byte = counter # : = %F4 => T0
                        943                                     = %F2 => T1
                        944                  1 byte = cntr value for high interval
                        945                  1 byte = prescaler (unshifted)
                        946
                        947  Output =       Selected timer is loaded and
                        948                 enabled; corresponding interrupt
                        949                 is enabled.  P36 is enabled as Tout.
                        950                 R13, R14, R15 modified.
                        951
                        952  Note =         The parameter list is not referenced
                        953                 following initialization.
                        954
                        955                 The value of  Prescaler  x  Counter
                        956                 must be > 26 (=%1A) for proper
                        957                 operation.
                        958 *****************************************************!
                        959 ENTRY
P 0288 DC  65           960         LD      TMP,#PLS_2
P 028A C3  DE           961         ldci    @TMP,@PTR       !low interval cntr!
P 028C C3  DE           962         ldci    @TMP,@PTR       !timer addr!
P 028E C3  DE           963         ldci    @TMP,@PTR       !high interval cntr!
P 0290 80  EE           964         decw    PTR
P 0292 80  EE           965         decw    PTR             !back to flag!
P 0294 56  F1  3F       966         and     TMR,#%3F        !will be modifying TMR!
P 0297 56  7F  DF       967         and     P3M_save,#%DF   !P36 = Tout!
P 029A E4  7F  F7       968         ld      P3M,P3M_save
P 029D E6  7B  01       969         ld      TEMP_4,#%1      !flag for pre_ctr!
P 02A0 8D  02B2'        970         jp      pre_ctr         !set up timer!
P 02A3                  971 END     pulse_i
                        972
                        973
                        974 GLOBAL
P 02A3                  975 pulse   PROCEDURE
                        976 !*****************************************************
                        977  Purpose =      To modify the counter load value
                        978                 to continue the pulse output generation.
                        979
                        980 *****************************************************!
                        981 ENTRY
                        982 !exchange values!
P 02A3 B4  65  67       983         xor     PLS_1,PLS_2
P 02A6 B4  67  65       984         xor     PLS_2,PLS_1
P 02A9 B4  65  67       985         xor     PLS_1,PLS_2
                        986 !exchange complete!
P 02AC F5  67  66       987         ld      @PLS_tmr,PLS_1  !load new value!
P 02AF BF               988         iret
P 02B0                  989 END     pulse
```

```
                    991 GLOBAL
P 02B0              992 delay   PROCEDURE
                    993 !*****************************************************
                    994  Purpose =      To generate an interrupt after a
                    995                 designated amount of time.
                    996
                    997  Input =        RR14 = address of parameter list in
                    998                        program memory:
                    999                  1 byte = counter # : = %F4 => T0
                   1000                                       = %F2 => T1
                   1001                  1 byte = Counter value
                   1002                  1 byte = Prescaler value and count mode
                   1003                         (to be loaded as is into
                   1004                          PRE0 or PRE1).
                   1005
                   1006  Output =       Selected timer is loaded and
                   1007                enabled; corresponding interrupt
                   1008                is enabled.
                   1009                R13, R14, R15 modified.
                   1010
                   1011  Note =         This routine will initialize the timer
                   1012                for single-pass or continuous mode
                   1013                as determined by bit 0 of byte 3 in
                   1014                the parameter list.
                   1015                The caller is responsible for provid-
                   1016                ing the interrupt service routine.
                   1017
                   1018                The parameter list is not referenced
                   1019                following initialization.
                   1020 *****************************************************!
                   1021 ENTRY
P 02B0 B0  7B      1022         clr     TEMP_4
                   1023 !fall into pre_ctr!
P 02B2             1024 END     delay
```

```
                        1026 INTERNAL
P 02B2                  1027 pre_ctr PROCEDURE
                        1028 !*****************************************************
                        1029  Purpose =       To get counter and prescaler values
                        1030                  from parameter list and modify control
                        1031                  registers appropriately.
                        1032
                        1033  Input   =       TEMP_4  = 0 => for 'delay'
                        1034                          = 1 => for 'pulse'
                        1035                          = TOD_imr => for 'tod'
                        1036 *****************************************************!
                        1037 ENTRY
P 02B2 C2  DE           1038          ldc     TMP,@PTR          !T0 or T1!
P 02B4 A0  EE           1039          incw    PTR
P 02B6 E6  7D  8C       1040          ld      TEMP_2,#%8C       !for TMR!
P 02B9 E6  7E  20       1041          ld      TEMP_3,#%20       !for IMR!
P 02BC A6  ED  F2       1042          cp      TMP,#T1
P 02BF 6B  06           1043          jr      eq,pre_1          !is for T1!
P 02C1 E6  7D  43       1044          ld      TEMP_2,#%43       !for TMR!
P 02C4 E6  7E  10       1045          ld      TEMP_3,#%10       !for IMR!
P 02C7 C3  DE           1046 pre_1:   ldci    @TMP,@PTR         !init counter!
P 02C9 C2  EE           1047          ldc     PTRh,@PTR         !prescaler!
P 02CB A6  7B  00       1048          cp      TEMP_4,#0         !shift prescaler?!
P 02CE 6B  12           1049          jr      eq,pre_2          !no!
P 02D0 DF               1050          scf                       !internal clock!
P 02D1 10  EE           1051          rlc     PTRh
P 02D3 DF               1052          scf                       !continuous mode!
P 02D4 10  EE           1053          rlc     PTRh
P 02D6 A6  7B  6C       1054          cp      TEMP_4,#TOD_imr
P 02D9 EB  0A           1055          jr      ne,pre_3          !for 'pulse'!
P 02DB 60  7E           1056          com     TEMP_3
P 02DD 54  7E  6C       1057          and     TOD_imr,TEMP_3    !insure no self-nesting!
P 02E0 60  7E           1058          com     TEMP_3
P 02E2 56  7D  0F       1059 pre_2:   and     TEMP_2,#%0F       !no Tout mode mod!
P 02E5 F3  DE           1060 pre_3:   ld      @TMP,PTRh         !init prescaler!
P 02E7 44  7D  F1       1061          or      TMR,TEMP_2        !init tmr mode!
P 02EA 8F               1062          di
P 02EB 44  7E  FB       1063          or      imr,TEMP_3        !enable interrupt!
P 02EE 9F               1064          ei
P 02EF AF               1065          ret
P 02F0                  1066 END      pre_ctr
                        1067 END PART_II

   0 errors
Assembly complete
```

A Comparison of Microcomputer Units

ZiLOG

Benchmark Report

INTRODUCTION

The microcomputer industry has recently developed single-chip microcomputers that incorporate on one chip functions previously performed by peripherals. These microcomputer units (MCUs) are aimed at markets requiring a dedicated computer. This report describes and compares the most powerful MCUs in today's market: the Zilog Z8611, the Intel 8051, and the Motorola MC6801. Table 1 lists facts that should be considered when comparing these MCUs.

Table 1. MCU Comparison

FEATURES	Zilog Z8611	Intel 8051	Motorola MC6801
On-Chip ROM	4Kx8	4Kx8	2Kx8
General-Purpose Registers	124	128	128
Special-Function Registers			
Status/Control	16	16	17
I/O ports	4	4	4
I/O			
Parallel lines	32	32	29
Ports	Four 8-bit	Four 8-bit	Three 8-bit,one 5-bit
Handshake	Hardware on three ports	None	Hardware on one port
Interrupts			
Source	8	5	7
External source	4	2	2
Vector	6	5	7
Priority	48 Programmable orders	2 Programmable orders	Nonprogrammable
Maskable	6	5	6
External Memory	120K bytes	124K bytes	64K bytes
Stack			
Stack pointer	16-Bit	8-Bit	16-Bit
Internal stack	Yes, uses 8-bits	Yes	Yes
External stack	Yes	No	Yes

Table 1. MCU Comparison
(Continued)

FEATURES	Zilog Z8611	Intel 8051	Motorola MC6801
Counter/ Timers			
Counters	Two 8-bit	Two 16-bit or two 8-bit	One 16-bit
Prescalers	Two 6-bit	No prescale with 16-bits; 5-bit prescale with 8-bits	None
Addressing Modes			
Register	Yes	Yes	No
Indirect Register	Yes	Yes	No
Indexed	Yes	Yes	Yes
Direct	Yes	Yes	Yes
Relative	Yes	Yes	Yes
Immediate	Yes	Yes	Yes
Implied	Yes	Yes	Yes
Index Registers	124, Any general-purpose register	1, Uses the accumulator for 8-bit offset	1, Uses 16-bit index register
Serial Communication Interface			
Full duplex UART	Yes	Yes	Yes
Interrupts for transmit and receive	One for each	One for both	One for both
Registers Double buffer	Receiver	Receiver	Transmitter/Receiver
Serial Data Rate	62.5K b/s @8 MHz 93.5K b/s @12 MHz	187.5K b/s @12 MHz	62.5K b/s @4 MHz
Speed			
Instruction execution average	2.2 Usec 1.5 Usec @12 MHz	1.5 Usec	3.9 Usec
Longest instruction	4.25 Usec 2.8 Usec @12 MHz	4 Usec	10 Usec
Clock Frequency	8 and 12 MHz	12 MHz	4 MHz
Power Down Mode	Saves first 124 registers	Saves first 128 registers	Saves first 64 registers
Context Switching	Saves PC and flags	Saves PC; programmer must save all registers	Saves PC, PSW, accumulators, and Index register

Table 1. MCU Comparison
(Continued)

FEATURES	Zilog Z8611	Intel 8051	Motorola MC6801
Development	40-Pin Protopack (8613) 64-Pin (8612) 40-Pin ROMless (Z8681)	40-Pin (8751)	40-Pin (68701)
Eprom	4K bytes (2732) 2K bytes (2716)	4K bytes	2K bytes
Availability	Now	TBA	Now

ARCHITECTURAL OVERVIEW

This section examines three chips: the on-chip functions and data areas manipulated by the Zilog, Intel and Motorola MCUs. The three chips have somewhat similar architectures. There are, however, fundamental differences in design criteria. The 8051 and the MC6801 were designed to maintain compatability with older products, whereas the Z8611 design was free from such restrictions and could experiment with new ideas. Because of this, the accumulator architectures of the MC6801 and the 8051 are not as flexible as that of the Z8611, which allows any register to be used as an accumulator.

Memory Spaces

The Z8611 CPU manipulates data in four memory spaces:

- 60K bytes of external data memory
- 60K bytes of external program memory
- 4K bytes of internal program memory (ROM)
- 144-byte register file

The 8051 CPU manipulates data in four memory spaces:

- 64K bytes of external data memory
- 60K bytes of external program memory
- 4K bytes of internal program memory
- 148-byte register file

The MC6801 manipulates data in three memory spaces:

- 62K bytes of external memory
- 2K bytes of internal program memory
- 149-byte register file

On-Chip ROM. All three chips have internal ROM for program memory. The Z8611 and the 8051 have 4K bytes of internal ROM, and the MC6801 has 2K bytes. In some cases, external memory may be required with the MC6801 that is not necessary with the Z8611 or the 8051.

On Chip RAM. All three chips use internal RAM as registers. These registers are divided into two catagories: general-purpose registers and special function registers (SFRs).

The 124 general-purpose registers in the Z8611 are divided into eight groups of 16 registers each. In the first group, the lowest four registers are the I/O port registers. The other registers are general purpose and can be accessed with an 8-bit address or a short 4-bit address. Using the 4-bit address saves bytes and execution time. Four-bit short addresses are discussed later. The general-purpose registers can be used as accumulators, address pointers, or Index registers.

The 128 general-purpose registers in the 8051 are grouped into two sets. The lower 32 bytes are allocated as four 8-register banks, and the upper registers are used for the stack or for general purpose. The registers cannot be used for indexing or as address pointers.

The MC6801 also has a 128-byte, general-purpose register bank, which can be used as a stack or as address pointers, but not as Index registers.

As pointed out in Table 1, any of the Z8611 general-purpose registers can be used for indexing; the MC6801 and the 8051 cannot use registers this way. The Z8611 can use any register as an accumulator; the MC6801 and the 8051 have fixed accumulators. The use of registers as memory pointers is very valuable, and only the Z8611 can use its registers in this way.

The number of general-purpose registers on each chip is comparable. However, because of its flexible design, the Z8611 clearly has a more powerful register architecture.

The Z8611 has 20 special function registers used for status, control, and I/O. These registers include:

- Two registers for a 16-bit Stack Pointer (SPH, SPL)
- One register used as Register Pointer for working registers (RP)
- One register for the status flags (FLAGS)
- One register for interrupt priority (IPR)
- One register for interrupt mask (IMR)
- One register for interrupt request (IRQ)
- Three mode registers for the four ports (P01M, P2M, P3M)
- Serial communications port used like a register (SIO)
- Two counter/timer registers (T0, T1)
- One Timer Mode Register (TMR)
- Two prescaler registers (PRE0, PRE1)
- Four I/O ports accessed as registers (PORT0, PORT1, PORT2, PORT3)

The 8051 also has 20 special function registers used for status, control, and I/O. They include:

- One register for the Stack Pointer (SP)
- Two accumulators (A,B)
- One register for the Program Status Word (PSW)
- Two registers for pointing to data memory (DPH, DPL)
- Four registers that serve as two 16-bit counter/timers (TH0, TH1, TL0, TL1)
- One mode register for the counter/timers (TMOD)
- One control register for the counter/timers (TCON)
- One register for interrupt enable (IEC)
- One register for interrupt priority (IPC)
- One register for serial communications buffer (SBUF)
- One register for serial communications control (SCON)
- Four registers used as the four I/O ports (P0, P1, P2, P3)

The MC6801 has 21 special function registers used for status, control, and I/O. These include:

- One register for RAM/EROM control
- One serial receive register
- One serial transmit register
- One register for serial control and status
- One serial rate and mode register
- One register for status and control of port 3
- One register for status and control of the timer
- Two registers for the 16-bit timer
- Two registers for 16-bit input capture used with timer
- Two registers for 16-bit output compare used with timer
- Four data direction registers associated with the four I/O ports
- Four I/O ports

The special function registers in the three chips seem comparable in number and function. However, upon closer examination, the SFRs of the MC6801 prove less efficient than those of the Z8611. The MC6801 has five registers associated with the I/O ports, whereas the Z8611 uses only three registers for the same functions. The MC6801 uses four registers to perform the serial communication function, whereas the Z8611 uses only one register and part of another.

The 8051 uses two registers for the accumulators; the Z8611 is not limited by this restriction. The 8051 also uses two registers for the serial communication interface, whereas the Z8611 accomplishes the same job with one register. Another two registers in the 8051 are used for data pointers; these are not necessary in the Z8611 since any register can be used as an address pointer.

The Z8611 uses registers more efficiently than either the MC6801 or the 8051. The registers saved by this optimal design are used to perform the functions needed for enhanced interrupt handling and for register pointing with short addresses. The Z8611 also supplies the extra register required for the external stack. These features are not available on the 8051 or the MC6801.

External Memory. All three chips can access external memory. The Z8611 and the 8051 can generate signals used for selecting either program or data memory. The Data Memory strobe (the signal used for selecting data or program memory) gives the Z8611 access to 120K bytes of external memory (60K bytes in both program and data memory). The 8051 can use 124K bytes of external memory (64K bytes of external data memory and 60K bytes of external program memory). The MC6801 can access only 62K bytes of external memory and does not distinguish between program and data memory. Thus, the Z8611 and the 8051 are clearly able to access more external memory than the MC6801.

On-Chip Peripheral Functions

In addition to the CPU and memory spaces, all chips provide an interrupt system and extensive I/O facilities including I/O pins, parallel I/O ports, a bidirectional address/ data bus, and a serial port for I/O expansion.

Interrupts. The Z8611 acknowledges interrupts from eight sources, four are external from pins IRQ_0-IRQ_3, and four are internal from serial-in, serial-out, and the two counter/timers. All interrupts are maskable, and a wide variety of priorities are realized with the Interrupt Mask Register and the Interrupt Priority Registers (see Table 1). All Z8611 interrupts are vectored, with six vectors located in the on-chip ROM. The vectors are fixed locations, two bytes long, that contain the memory address of the service routine.

The 8051 acknowledges interrupts from five sources: two external sources (from INT0 and INT1) and three internal sources (one from each of the internal counters and one from the serial I/O port). All interrupts can be disabled individually or globally. Each of the five sources can be assigned one of two priorities: high or low. All 8051 interrupts are vectored. There are five fixed locations in memory, each eight bytes long, allocated to servicing the interrupt.

The MC6801 has one external interrupt, one nonmaskable interrupt, an internal interrupt request, and a software interrupt. The internal interrupts are caused by the serial I/O port, timer overflow, timer output compare, and timer input capture. The priority of each interrupt is preset and cannot be changed. The external interrupt can be masked in the Condition Code register. The MC6801 vectors the interrupts to seven fixed addresses in ROM where the 16-bit address of the service routine is located.

When an interrupt occurs in the 8051, only the Program Counter is saved; the user must save the flags, accumulator, and any registers that the interrupt service routine might affect. The MC6801 saves the Program Counter, acumulators, Index register, and the PSW; the user must save all registers that the interrupt service routine might affect. The Z8611 saves the Program Counter and the Flags register. To save the 16 working registers, only the Register Pointer register need be pushed onto the stack and another set of working registers is used for the service routine. For more detail on working registers and interrupt context switching, see the Z8 Technical Manual (03-3047-02).

With regard to interrupts, the Z8611 is clearly superior. The Z8611 requires only one command to save all the working registers, which greatly increases the efficiency of context switching.

I/O Facilities. The Z8611 has 32 lines dedicated to I/O functions. These lines are grouped into four ports with eight lines per port. The ports can be configured individually under software control to provide input, output, multiplexed address/data lines, timing, and status. Input and output can be serial or parallel, with or without handshake. One port can be configured for serial transmission and four ports can be configured for parallel transmission. With parallel transmission, ports 0, 1, and 2 can transmit data with the handshake provided by port 3.

The 8051 also has 32 I/O lines grouped together into four ports of eight lines each. The ports can be configured under program control for parallel or serial I/O. The ports can also be configured for multiplexed address/data lines, timing, and status. Handshake is provided by user software.

The MC6801 has 29 lines for I/O (three 8-bit ports and one 5-bit port). One port has two lines for handshake. The ports provide all the signals needed to control input and output either serially or in parallel, with or without multiplexed address/data lines. They can be used to interface with external memory.

The main differences in I/O facilities are the number of 8-bit ports and the hardware handshake. The Z8611 and the 8051 have four 8-bit ports, whereas the MC6801 has three 8-bit ports and an additional 5-bit port. The Z8611 has hardware handshake on three ports, the MC6801 has hardware handshake on only one port, and the 8051 has no hardware handshake.

Counter/timers. The Z8611 has two 8-bit counters and two 6-bit programmable prescalers. One prescaler can be driven internally or externally; the other prescaler is driven internally only. Both timers can interrupt the CPU when counting is completed. The counters can operate in one of two modes: they can count down until interrupted, or they can count down, reload the initial value, and start counting down again (continuously). The counters for the Z8611 can be used for measuring time intervals and pulse widths, generating variable pulse widths, counting events, or generating periodic interrupts.

The 8051 has two 16-bit counter/timers for measuring time intervals and pulse widths, generating pulse widths, counting events, and generating periodic interrupts. The counter/timers have several modes of operation. They can be used as 8-bit counters or timers with two 5-bit programmable prescalers. They can also be used as 16-bit counter/timers. Finally, they can be set as 8-bit modulo-n counters with the reload value held in the high byte of the 16-bit register. An interrupt is generated when the counter/timer has completed counting.

The MC6801 has one 16-bit counter which can be used for pulse-width measurement and generation. The counter/timer actually consists of three 16-bit registers and an 8-bit control/status register. The timer has an input capture register, an output compare register, and a free-running counter. All three 16-bit registers can generate interrupts.

Serial Communications Interface. The Z8611 has a programmable serial communication interface. The chip contains a UART for full-duplex, asynchronous, serial receiver/ transmitter operation. The bit rate is controlled by counter/timer 0 and has a maximum bit rate of 93.500 b/s. An interrupt is generated when an assembled character is transferred to the receive buffer. The transmitted character generates a separate interrupt. The receive register is double-buffered. A hardware parity generator and detector are optional.

The 8051 handles serial I/O using one of its parallel ports. The 8051 bit rate is controlled

by counter/timer 1 and has a maximum bit rate of 187,500 b/s. The 8051 generates one interrupt for both transmission and receipt. The receive register is double-buffered.

The MC6801 contains a full-duplex, asynchronous, serial communication interface. The bit rate is controlled by a rate register and by the MCU's clock or an external clock. The maximum bit rate is 62,500 b/s. Both the transmit and the receive registers are double-buffered. The MC6801 generates only one interrupt for both transmit and receive operations. No hardware parity generation or detection is available, although it does have automatic detection of framing errors and overrun conditions.

The 8051 and the MC6801 generate only one interrupt for both transmit and receive, whereas the Z8611 has a separate interrupt for each. The ability to generate separate interrupts greatly enhances the use of serial communications, since separate service routines are often required for transmitting and receiving.

Other differences between the Z8611, MC6801, and the 8051 occur in the hardware parity detector, the double-buffering of registers, framing error detectors and overrun conditions. The 8051 has a faster data rate than either the Z8611 or the MC6801. The MC6801 has the advantage of a hardware framing error detector and automatic detection of overrun conditions. The MC6801 also has both its transmit and receive registers double-buffered. The Z8611 has a hardware parity detector. For detection of framing errors and overrun conditions, a simple, low-overhead software check is available that uses only two instructions. See Z8600 Software Framing Error Detection Application Brief (document #617-1881-0004).

INSTRUCTION ARCHITECTURE

The architecture of the Z8611 is designed specifically for microcomputer applications. This fact is manifest in the instruction composition. The arduous task of programming the MC6801 and the 8051 starkly contrasts that of programming the Z8611.

Addressing Modes

The Z8611 and the 8051 both have six addressing modes: Register, Indirect Register, Indexed, Direct, Relative, and Immediate. The MC6801 has five addressing modes: Accumulator, Indexed, Direct, Relative, and Immediate. A quick comparison of these addressing modes reveals the versatility of the Z8611 and the 8051. The addressing modes of the MC6801 have several restrictions, as shown in Table 1. While the 8051 has all the addressing modes of the Z8611, its use of them is restricted. The Z8611 allows many more combinations of addressing modes per instruction, because any of its registers can be used as an accumulator. For example, the instructions to clear, complement, rotate, and swap nibbles are all accumulator oriented in the 8051 and operate on the accumulator only. These same commands in the Z8611 can use any register and access it either directly, with register addressing, or with indirect register addressing.

Indexed Addressing. All three chips differ in their handling of indexing. The Z8611 can use any register for indexing. The 8051 can use only the accumulator as an Index register in conjunction with the data pointer or the Program Counter. The MC6801 has one 16-bit Index register. The address located in the second byte of an instruction is added to the lower byte of the Index register. The carry is added to the upper byte for the complete address. The MC6801 requires the index value to be an immediate value.

The MC6801 has only one 16-bit Index register and an immediate 8-bit value from the second byte of the instruction. Hence, the Indexed mode of the MC6801 is much more restrictive than that of the Z8611. The 8051 must use the accumulator as its only Index register, loading the accumulator with the register address each time a reference is made. Then, using indexing, the data is moved into the accumulator, eradicating the previous index. This forces a stream of data through the accumulator and requires a reload of the index before access can be made again. The Z8611 is clearly superior to both the MC6801 and the 8051 in the flexibility of its indexed addressing mode.

Short and Long Addressing. Short addressing helps to optimize memory space and execution speed. In sample applications of short register addressing, an eight percent decrease in the number of bytes used was recorded.

All three chips have short addressing modes, but the Z8611 has short addressing for both external memory and register memory. The 8051 has short addressing for the lowest 32 registers only.

The Z8611 has two different modes for register addressing. The full-byte address can be used to provide the address, or a 4-bit address can be used with the Register Pointer. To use the working registers, the Register Pointer is set for a particular bank of 16 registers, and then one of the 16 registers is addressed with four bits. Another feature for addressing external memory is the use of a 12-bit address in place of a full 16-bit address. To use the 12-bit address, one port supplies the eight multiplexed address/data lines and another port supplies four bits for the address. The remaining four bits of the second port can be used for I/O. This feature allows access to a maximum of 10K bytes of memory.

The 8051 uses short addresses by organizing its lowest 32 registers into four banks. The bank select is located in a 2-bit field in the PSW, with three bits addressing the register in the bank.

The MC6801 uses extended addressing for addressing external memory. With a special, nonmultiplexed expansion mode, 256 bytes of external memory can be accessed without the need for an external address latch. The MC6801 uses one 8-bit port for the address and another port for the data.

Stacks

The Z8611 and the MC6801 provide for external stacks, which require a 16-bit Stack Pointer. Internal stacks use only an 8-bit Stack Pointer. The 8051 uses only a limited internal stack requiring an 8-bit Stack Pointer. Using an external stack saves the internal RAM registers for general-purpose use.

Summary

The stack structure of the Z8611 and the MC6801 is better than that of the 8051. In most applications, the 8051 is more flexible and easier to program than the MC6801. The Z8611 is easier to use than either the 8051 or the MC6801 because of its register flexibility and its numerous combinations of addressing modes. The 8051 features a unique 4 μn multiply and divide command. The MC6801 has a multiply, but it takes 10μs to perform it.

In summary, the Z8611 has the most flexible addressing modes, the most advanced indexing capabilities, and superior space- and time-saving abilities with respect to short addressing.

DEVELOPMENT SUPPORT

All three vendors provide development support for their products. This section discusses the different support features, including development chips, software, and modules.

Chips

Zilog offers an entire family of microcomputer chips for product development and final product. The Z8611 is a single-chip microcomputer with 4K bytes of mask-programmed ROM. For development, two other chips are offered. The Z8612 is a 64-pin, development version with full interface to external memory. The Z8613 is a prototype version that uses a functional, piggy-back, EPROM protopak. The Z8613 can use either a 4K EPROM (2732) or a 2K EPROM (2716). Zilog also offers a ROMless version in a 40-pin package that has all the features of the Z8611 except on-board ROM (Z8681).

Intel offers a similar line of development chips with its 8051 family. The 8031 has no internal ROM and the 8751 has 4K of internal EPROM.

Motorola offers the MC6801, MC6803, MC6803NR, and MC68701. These are all similar except the MC68701 has 2K bytes of EPROM and the MC6801 has 2K bytes of ROM. The MC6803 has no internal ROM and the MC6803NR has neither ROM nor RAM on board.

The Z8613 and the MC68701 are both available now, but the 8751 is still unavailable (as of April 1981).

Software

Development software includes assemblers, and conversion programs. All manufacturers offer some or all of these features.

Since the MC6801 is compatible with the 6800, there is no need for a new assembler. The Z8611 and the 8051 both offer assemblers for their products. The Zilog PLZ/ASM assembler generates relocatable and absolute object code. PLZ/ASM also supports high-level control and data statements, such as IF... THEN...ELSE. Intel offers an absolute macroassembler, ASM51, with their product. They also offer a program for converting 8048 code to 8051 code.

Modules

The Z8611 development module has two 64-pin development versions of the 40-pin, ROM-masked Z8611. Intel offers the EM-51 emulation board, which contains a modified 8051 and PROM or EPROM in place of memory. Motorola has the MEX6801EVM evaluation board for program development. All three development boards are available now.

ADDITIONAL FEATURES

Additional features include Power Down mode, self-testing, and family-compatibility.

Power Down Mode

All three microcomputers offer a Power Down mode. The Z8611 and the 8051 save all of their registers with an auxilary power supply. The MC6801 uses an auxiliary power supply to save only the first 64 bytes of its register file.

The Z8611 uses one of the crystal input pins for the external power supply to power the registers in Power Down mode. Since the XTAL2 input must be used, an external clock generator is necessary and is input via XTAL1. The 8051 and the MC6801 both have an input reserved for this function. The MC6801 uses the V_{CC} standby pin, and the 8051 uses the V_{pd} pin.

Family Compatibility

Another strength of the Z8611 is its expansion bus, which is completely compatible with the Zilog Z-BUS™. This means that all Z-BUS peripherals can be used directly with the Z8611.

The MC6801 is fully compatible with all MC6800 family products. The 8051 is software compatible with the older 8048 series and all others in that family.

BENCHMARKS

The following benchmark tests were used in this report to compare the Z8611, 8051, and MC6801:

- Generate CRC check for 16-bit word.
- Search for a character in a block of memory.
- Execute a computed GOTO - jump to one of eight locations depending on which of the eight bits is set.
- Shift a 16-word five places to the right.
- Move a 64-byte block of data from external memory to the register file.
- Toggle a single bit on a port.
- Measure the subroutine overhead time.

These programs were selected because of their importance in microcomputer applications. Algorithms that reflect a unique function or feature were excluded for the sake of comparison. Although programs can be optimized for a particular chip and for a particular attribute (code density or speed) these programs were not.

The figures cited in this text are taken directly from the vendor's documentation. Therefore, the cycles given below for the MC6801 and the 8051 are in machine cycles and the Z8611 figures are given in clock cycles. The Z8611 clock cycles should be divided by six to give the instruction time in microseconds. The 8051 and MC6801 machine cycle is 1 μs, and the Z8611 clock cycle is .166 μs at 12 MHz.

Because of the lack of availability of the MC6801 and the 8051, the benchmark programs listed here have not yet been run. When these products are readily available, the programs will be run and later editions of this document will reflect any changes in the findings.

Program Listings

CRC Generation

8051

			Machine Cycles	Bytes
	MOV	INDEX, #8	1	2
LOOP:	MOV	A, DATA	1	2
	XRL	A, HCHECK	1	2
	RLC	A	1	1
	MOV	A, LCHECK	1	2
	XRL	A, LPOLY	1	2
	RLC	A	1	1
	MOV	LCHECK, A	1	2
	MOV	A, HCHECK	1	2
	XRL	A, HPOLY	1	2
	RLC	A	1	1
	MOV	HCHECK, A	1	2
	CLR	C	1	1
	MOV	A, DATA	1	2
	RLC	A	1	1
	MOV	DATA, A	1	2
	DJNZ	INDEX, LOOP	2	3
	RET		2	1

N = 3+17X8 = **139 cycles**
@12 MHz = 139 μs
Instructions = 18
Bytes = 31

MC6801

			Machine Cycles	Bytes
	LDAA	#$08	2	2
LOOP:	STAA	COUNT	3	2
	LDAA	HCHECK	3	2
	EORA	DATA	3	2
	ROLA		2	1
	LDAD	POLY	4	2
	EORA	HCHECK	3	2
	EORB	LCHECK	3	2
	ROLB		2	1
	ROLA		2	1
	STAD	LCHECK	4	2
	ASL	DATA	6	3
	DEC	COUNT	6	3
	BNE	LOOP	4	2
	RTS		5	1

N = 45X8+7 = **367 cycles**
@4 MHz = 367 μs
Instructions = 15
Bytes = 28

Z8611

			Clock Cycles	Bytes
	LD	INDEX, #8	6	2
LOOP:	LD	R6, DATA	6	2
	XOR	R6, HCHECK	6	2
	RLC	R6	6	2
	XOR	LCHECK, LPOLY	6	2
	RLC	LCHECK	6	2
	XOR	HCHECK, HPOLY	6	2
	RLC	HCHECK	6	2
	RCF		6	1
	RLC	DATA	6	2
	DJNZ	INDEX, LOOP	12 or 10	2
	RET		14	1

N = 20+66X7+64 = **546 cycles**
@12 MHz = 91 μs
Instructions = 12
Bytes = 22

Character Search Through Block of 40 Bytes

8051

			Machine Cycles	Bytes
	MOV	INDEX, #41	1	2
	MOV	DPTR, #TABLE	2	3
LOOP1:	DJNZ	INDEX, LOOP 2	2	2
	SJMP	OUT	2	2
LOOP2:	MOV	A, INDEX	1	2
	MOVC	A, @A+DPTR	2	1
	CJNE	A, CHARAC, LOOP1	2	3
OUT:				

N = 3+39X7+4 = **280 cycles**
@12 MHz = 280μs
Instructions = 7
Bytes = 15

MC6801

			Machine Cycles	Bytes
	LDAB	#$40	2	2
	LDAA	#CHARAC	2	2
	LDX	#TABLE	3	3
LOOP:	CMPA	$0, X	4	2
	BEQ	OUT	4	2
	INX		3	1
	DECB		2	1
	BNE	LOOP	4	2
OUT:	-			
	-			
	-			

N = 7+40X17 = **687 cycles**
@4 MHz = 687μs
Instructions = 8
Bytes = 15

Z8611

			Clock Cycles	Bytes
	LD	INDEX, #40	6	2
LOOP:	LD	DATA, TABLE (INDEX)	10	3
	CP	DATA, CHARAC	6	2
	JR	Z, OUT	12 or 10	2
	DJNZ	INDEX, LOOP	12 or 10	2
OUT:	-			
	-			

N = 6+38X40 = **1524 cycles**
@12 MHz = 254μs
Instructions = 5
Bytes = 11

Shift 16-Bit Word to Right 5-Bits

8051

			Machine Cycles	Bytes
	MOV	INDEX #5	1	2
LOOP:	CLR	C	1	1
	MOV	A, WORD + 1	1	2
	RRC	A	1	1
	MOV	WORD + 1, A	1	2
	MOV	A, WORK	1	2
	RRC	A	1	1
	MOV	WORD, A	1	2
	DJNZ	INDEX, LOOP	2	2

N = 1+9X5 = **46 Cycles**
@12 MHz = 46μs
Instructions = 9
Bytes = 15

MC6801

			Machine Cycles	Bytes
	LDX	#5	6	3
	LDAD	WORK	4	2
LOOP:	LSRD		3	1
	DEX		3	1
	BNE	LOOP	4	2
	STAD	WORD	4	2

N = 10X5+11 = **61 Cycles**
@4 MHz = 61μs
Instructions = 6
Bytes = 11

Z8611

			Clock Cycles	Bytes
	LD	INDEX, #5	6	2
LOOP:	CCF		6	1
	RRC	WORD + 1	6	2
	RRC	WORD	6	2
	DJNZ	INDEX, LOOP	12 or 10	2

N = 6+4X30+28 = **154 Cycles**
@12 MHz = 26μs
Instructions = 5
Bytes = 9

Computed GOTO

8051

			Machine Cycles	Bytes
	MOV	INDEX, #40	1	2
LOOP:	MOV	A, DATA	1	2
	RLC	A	1	1
	JC	OUT	2	2
	MOV	A, INDEX	1	1
	ADD	A, #3	1	2
	MOV	INDEX, A	1	1
	SJMP	LOOP	2	2
OUT:	MOV	DPTR, #TABLE	2	3
	MOV	A, INDEX	1	1
	JMP	@A+DPTR	2	1
TABLE:	LCALL	ADDR1		3
	-			
	-			
	LCALL	ADDRN	2	

N = 1+9X7+11 = **75 Cycles**
@12 MHz = 75μs
Instructions = 12
Bytes = 21

MC6801

			Machine Cycles	Bytes
	LDAB	#2	2	2
	LDX	TABLE	3	3
LOOP:	RORA		2	1
	BCS	OUT	4	2
	ABX		3	1
	JMP	LOOP	3	2
OUT:	LDX	0, X	5	3
	JMP	0, X	4	3

N = 8X12+14 = **110 Cycles**
@4 MHz = 110μs
Instructions = 8
Bytes = 17

Z8611

			Clock Cycles	Bytes
	CLR	INDEX	6	2
LOOP:	INC	INDEX	6	1
	RLC	DATA	6	2
	JR	NC, LOOP	12 or 10	2
	LD	ADDR,TABLE 1, (INDEX)	10	3
	LD	ADDR+1,TABLE 2, (INDEX)	10	3
	JP	@ADDR	12	2

N = 6+24X7+54 = **228 Cycles**
@12 MHz = 38μs
Instructions = 7
Bytes = 15

Move 64-Byte Block

8051

			Machine Cycles	Bytes
	MOV	INDEX, #COUNT	1	2
LOOP:	MOV	DPTR, #ADDR1	2	3
	MOVX	A, @DPTR	2	1
	INC	#ADDR1	1	1
	MOV	@ADDR2,A	1	1
	INC	ADDR2	1	1
	DJNZ	INDEX, LOOP	2	1

N = 1+9X64 = **577 Cycles**
@12 MHz = 577μs
Instructions = 7
Bytes = 10

MC6801

			Machine Cycles	Bytes
	LDAB	#COUNT	2	2
LOOP:	LDX	ADDR1	4	3
	LDAA	0, X	4	2
	INX		3	1
	STAA	ADDR1	4	2
	LDX	ADDR2	4	3
	STAA	0, X	4	2
	INX		3	1
	STX	ADDR2	4	2
	DECB		2	1
	BNE	LOOP	4	2

N = 64X36+2 = **2306 Cycles**
@4 MHz =2306μs
Instructions = 11
Bytes = 21

Z8611

			Clock Cycles	Bytes
	LD	INDEX, #COUNT	6	2
LOOP:	LDEI	@ADDR2, @ADDR1	18	2
	DJNZ	INDEX, LOOP	12 or 10	2

N = 6+63X30+28 = **1924 Cycles**
@12 MHz = 321μs
Instructions = 3
Bytes = 6

Toggle a Port Bit

8051	Machine Cycles	Bytes
XRL PO, #YY	2	3
N = **2 Cycles**		
@12 MHz = 2 μs		
Instructions = 1		
Bytes = 3		

MC6801	Machine Cycles	Bytes
LDAA PORTO	3	2
EORA #YY	2	2
STAA PORTO	3	2
N = **8 Cycles**		
@4 MHz = 8 μs		
Instructions = 3		
Bytes = 6		

Z8611	Clock Cycles	Bytes
XOR PORTO, #YY	10	2
N = **10 Cycles**		
@12 MHz = 1.7 μs		
Instructions = 1		
Byte = 2		

Subroutine Call/Return Overhead

8051	Machine Cycles	Bytes
LCALL SUBR	2	3
-		
-		
-		
SUBR: -		
-		
-		
RET	2	1
N = **4 Cycles**		
@12 MHz = 4 μs		
Instructions = 2		
Bytes = 4		

MC6801	Machine Cycles	Bytes
JSR SUBR	9	2
-		
-		
-		
SUBR: -		
-		
-		
RTS	5	1
N = **14 Cycles**		
@4 MHz = 14 μs		
Instructions = 2		
Bytes = 3		

Z8611	Clock Cycles	Bytes
CALL @SUBR	20	2
-		
-		
-		
SUBR: -		
-		
-		
RET	14	1
N = **34 Cycles**		
@12 MHz = 5.7 μs		
Instructions = 2		
Bytes = 3		

Results

Table 2 summarizes the results of this comparison. The relative performance column lists the speeds of the MC6801 and 8051 divided by the Z8611 speeds (12 MHz). The overall performance averages the separate relative performances. The higher the number, the faster the Z8611 as compared to the MC6801 and the 8051.

The relative performance figures show that the Z8611 runs 50 percent faster than the 8051 and 250 percent faster than the MC6801. Although speed is not necessarily the most important criterion for selecting a particular product, the Z8611 proves to be an undeniably superior product when speed is added to the advantages of programming ease, code density, and flexibility.

Table 2. Benchmark Program Results

Benchmark Test	MC6801 (4 MHz) cycles	time	8051 (12 MHz) cycles	time	Z8 (8 MHz) cycles	time	Z8 (12 MHz) cycles	time	Relative Performance MC6801	8051
CRC Generation	367	367	139	139	546	137	546	91	4.03	1.53
Character Search	687	687	280	280	1524	382	1524	254	2.70	1.10
Computed GOTO	110	110	75	75	228	57	228	38	2.89	1.97
Shift Right 5 Bits	61	61	46	46	154	38	154	26	2.35	1.78
Move 64-byte block	2306	2306	577	577	1924	481	1924	321	7.18	1.80
Subroutine Overhead	14	14	4	4	34	8.5	34	5.7	2.46	0.70
Toggle a Port Bit	8	8	2	2	10	2.5	10	1.7	4.71	1.18
					Overall Performance				3.76	1.44

Note: All times are given in microseconds.

Table 3. Byte/Instruction/Time Comparison

	Bytes				Instructions				Time (microseconds)		
	MC6801	8051	Z8611		MC6801	8051	Z8611		MC6801	8051	Z8611
CRC Generation	28	31	22		15	18	12		367	139	91
Character Search	15	15	11		8	7	5		687	280	254
Shift Right 5 Bits	11	15	9		6	9	5		61	46	26
Computed GOTO	17	21	15		8	12	7		110	75	38
Move Block	21	10	6		11	7	3		2306	577	321
Toggle Port Bit	6	3	2		3	1	1		8	2	1.7
Subroutine Call	3	4	3		2	2	2		14	4	5.7

SUMMARY

The hardware of the three chips compared is very similar. The Z8611, however, has several advantages, the most important of which is its interrupt structure. It is more advanced than the interrupt structures of both the 8051 and the MC6801. Other advantages of the Z8611 over either the MC6801 or the 8051 include I/O facilities with parity detection and hardware handshake and a larger amount of internal ROM (the MC6801 has only 2K bytes).

Substantial differences are apparent with regard to software architecture. The addressing modes of the Z8611 are more flexible than those of either the MC6801 or the 8051. The Z8611 can use byte-saving addressing with working registers, and it has short external addresses for saving I/O lines. It can also provide for an external stack. The register architecture (as opposed to the accumulator architecture) of the Z8611 saves execution time and enhances programming speed by reducing the byte count.

The Z8611 microcomputer stands out as the most powerful chip of the three, and concurrently, it is the easiest to program and configure.

Z86XX Interrupt Request Register

ZiLOG

Application Brief

The Interrupt Request Register (IRQ, R250) stores requests from the six possible interrupt sources (IRQ^0-IRQ^5) in the Z8600 series microcomputer. In addition to other functions, a hardware reset to the Z8600 disables the IRQ register and resets its request bits. Before the IRQ will register requests, it must first be enabled by executing an Enable Interrupts (EI) instruction. Setting the Enable Interrupt bit in the Interrupt Mask Register (IMR, R251) is not an equivalent operation for this purpose; to enable the IRQ, an EI instruction is required. The function of this EI instruction is distinct from its task of globally enabling the interrupt system. Even in a polled system where IRQ bits are tested in software, it is necessary to execute the EI.

The designer must ensure that unexpected and undesirable interrupt requests will not occur after the EI is executed. One method of doing this is to reset all interrupt enable bits in the IMR for levels that are possible interrupt sources; the EI instruction may then be safely executed. Once EI is executed, the program may immediately execute a Disable Interrupts (DI) instruction. The code necessary to perform these operations is as follows:

```
RESET:  LD   IMR, #%XX   !SET INTERRUPT MASK!
        EI               !ENABLE GLOBAL INTER-
                          RUPT, ENABLE IRQ!
```

where XX has a Ø in each bit position corresponding to the interrupt level to be disabled. If all IMR bits are to be reset, a CLR IMR instruction may be used.

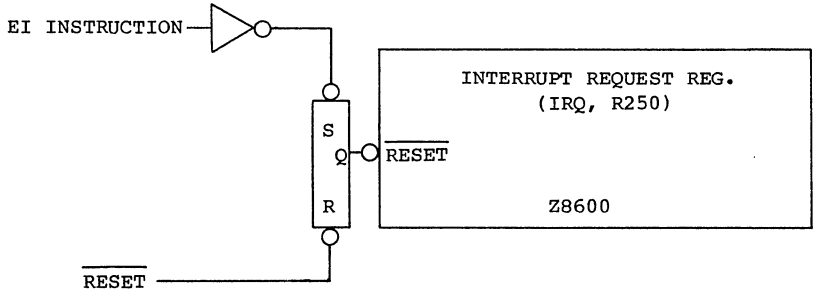


Figure 1 - IRQ Reset Functional Logic Diagram

Z8 Family Software Framing Error Detection

ZiLOG

Application Brief

The Zilog Z8600 UART microcomputer is a high-performance, single-chip device that incorporates on-chip ROM, RAM, parallel I/O, serial I/O, and a baud rate generator. The UART is capable of full-duplex, asynchronous serial communication at nine standard software-selectable baud rates from 110 to 19.2K baud; other nonstandard rates can also be obtained under software control. Odd parity generation and checking can also be selected.

Three possible error conditions can occur during reception of serial data: framing error, parity error, and overrun error. A framing error condition occurs when a stop bit is not received at the proper time (Figure 1). This can result from noise in the data channel, causing erroneous detection of the previous start bit or lack of detection of a properly transmitted stop bit. The Z8600 UART does not incorporate hardware framing error detection but does facilitate a simple, low-overhead software detection method.

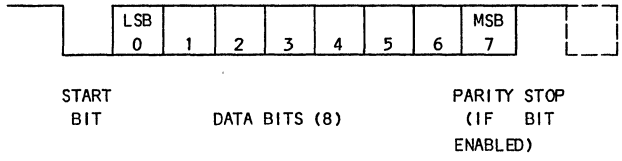


Fig. 1 - Asynchronous Data Format

In the middle of the stop bit time, the Z8600 UART automatically posts a serial input interrupt request on IRQ_3. The serial input can also be tested by reading Port 3 bit 0 ($P3_0$) as shown in Figure 2. Thus, within the interrupt service routine or polling loop, it is only necessary to test $P3_0$ in order to identify a framing error. If $P3_0$ is Low when IRQ_3 goes High, a framing error condition exists and the following code is used to test this:

```
TM P3, #%01    ! TEST FOR P30 = 1 !
JR Z, FERR     ! ELSE FRAMING ERROR !
```

The execution time of this framing error test is only 5.5 µs at 8 MHz. In the worst case (19.2K baud), this would result in 1% overhead. Only five program bytes are required.

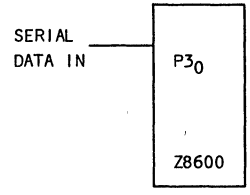


Fig. 2 - Z8600 Serial Input Connection

Z8 is a trademark of Zilog, Inc.

CONCLU

While the Z8600 UART does not incorporate hardware framing error detection, this feature can be implemented in software with a maximum penalty of 1% at 19.2K baud using no additional hardware and only five bytes of program memory.

A Programmer's Guide to the Z8™ Microcomputer

ZiLOG

Application Note

Doll Freund

SECTION 1

Introduction

The Z8 is the first microcomputer to offer both a highly integrated microcomputer on a single chip and a fully expandable microprocessor for I/O-and memory-intensive applications. The Z8 has two timer/counters, a UART, 2K bytes internal ROM, and a 144-byte internal register file including 124 bytes of RAM, 32 bits of I/O, and 16 control and status registers. In addition, the Z8 can address up to 124K bytes of external program and data memory, which can provide full, memory-mapped I/O capability.

This application note describes the important features of the Z8, with software examples that illustrate its power and ease of use. It is divided into sections by topic; the reader need not read each section sequentially, but may skip around to the sections of current interest.

It is assumed that the reader is familiar with the Z8 and its assembly language, as described in the following documents:

- *Z8 Technical Manual* (03-3047-02)
- *Z8 PLZ/ASM Assembly Language Programming Manual* (03-3023-02)

Accessing Register Memory

The Z8 register space consists of four I/O ports, 16 control and status registers, and 124 general-purpose registers. The general-purpose registers are RAM areas typically used for accumulators, pointers, and stack area. This section describes these registers and how they are used. Bit manipulation and stack operations affecting the register space are discussed in Sections 4 and 5, respectively.

2.1 Registers and Register Pairs. The Z8 supports 8-bit registers and 16-bit register pairs. A register pair consists of an even-numbered register concatenated with the next higher numbered register (%00 and %01, %02 and %03, ... %7E and %7F, %F0 and %F1, ... %FE and %FF). A register pair must be addressed by reference to the even-numbered register. For example,

%F1 and %F2 is not a valid register pair;
%F0 and %F1 is a valid register pair, addressed by reference to %F0.

Register pairs may be incremented (INCW) and decremented (DECW) and are useful as pointers for accessing program and external data memory. Section 3 discusses the use of register pairs for this purpose.

Any instruction which can reference or modify an 8-bit register can do so to any of the 144 registers in the Z8, regardless of the inherent nature of that register. Thus, I/O ports, control, status, and general-purpose registers may all be accessed and manipulated without the need for special-purpose instructions. Similarly, instructions which reference or modify a 16-bit register pair can do so to any of the valid 72 register pairs. The only exceptions to this rule are:

- The DJNZ (decrement and jump if non-zero) instruction may successfully operate on the general-purpose RAM registers (%04-%7F) only.
- Six control registers are write-only registers and therefore, may be modified only by such instructions as LOAD, POP, and CLEAR. Instructions such as OR and AND require that the current contents of the operand be readable and therefore will not function properly on the write-only registers. These registers are the following: *the timer/counter prescaler registers PRE0 and PRE1, the port mode registers P01M, P2M, and P3M, the interrupt priority register IPR.*

2. Accessing Register Memory (Continued)

2.2 Register Pointer. Within the register addressing modes provided by the Z8, a register may be specified by its full 8-bit address (0-%7F, %F0-%FF) or by a short 4-bit address. In the latter case, the register is viewed as one of 16 working registers within a working register group. Such a group must be aligned on a 16-byte boundary and is addressed by Register Pointer RP (%FD). As an example, assume the Register Pointer contains %70, thus pointing to the working register group from %70 to %7F. The LD instruction may be used to initialize register %76 to an immediate value in one of two ways:

```
LD  %76,#1  !8-bit register address is given
            by instruction (3 byte instruc-
            tion)!
   or
LD  R6,#1   !4-bit working register address
            is given by instruction; 4-bit
            working register group
            address is given by Register
            Pointer (2 byte instruction)!
```

The address calculation for the latter case is illustrated in Figure 1. Notice that 4-bit working-register addressing offers code compactness and fast execution compared to its 8-bit counterpart.

To modify the contents of the Register Pointer, the Z8 provides the instruction

```
SRP  #value
```

Execution of this instruction will load the upper four bits of the Register Pointer; the lower four bits are always set to zero. Although a load instruction such as

```
LD  RP,#value
```

could be used to perform the same function, SRP provides execution speed (six vs. ten cycles) and code space (two vs. three bytes) advantages over the LD instruction. The instruction

```
SRP  #%70
```

is used to set the Register Pointer for the above example.

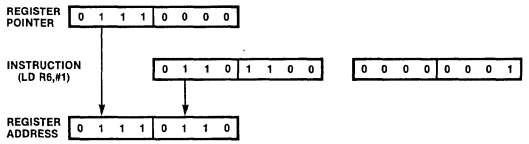


Figure 1. Address Calculation Using the Register Pointer

2.3 Context Switching. A typical function performed during an interrupt service routine is context switching. Context switching refers to the saving and subsequent restoring of the program counter, status, and registers of the interrupted task. During an interrupt machine cycle, the Z8 automatically saves the Program Counter and status flags on the stack. It is the responsibility of the interrupt service routine to preserve the register space. The recommended means to this end is to allocate a specific portion of the register file for use by the service routine. The service routine thus preserves the register space of the interrupted task by avoiding modification of registers not allocated as its own. The most efficient scheme with which to implement this function in the Z8 is to allocate a working register group (or portion thereof) to the interrupt service routine. In this way, the preservation of the interrupted task's registers is solely a matter of saving the Register Pointer on entry to the service routine, setting the Register Pointer to its own working register group, and restoring the Register Pointer prior to exiting the service routine. For example, assume such a register allocation scheme has been implemented in which the interrupt service routine for IRQ0 may access only working register Group 4 (registers %40-%4F). The service routine for IRQ0 should be headed by the code sequence:

```
PUSH RP     !preserve Register Pointer of
            interrupted task!
SRP  #%40   !address working register
            group 4!
```

Before exiting, the service routine should execute the instruction

```
POP  RP
```

to restore the Register Pointer to its entry value.

It should be noted that the technique described above need not be restricted to interrupt service routines. Such a technique might prove efficient for use by a subroutine requiring intermediate registers to produce its outputs. In this way, the calling task can assume that its environment is intact upon return from the subroutine.

2. Accessing Register Memory (Continued)

2.4 Addressing Mode. The Z8 provides three addressing modes for accessing the register space: Direct Register, Indirect Register, and Indexed.

2.4.1 Direct Register Addressing. This addressing mode is used when the target register address is known at assembly time. Both long (8-bit) register addressing and short (4-bit) working register addressing are supported in this mode. Most instructions supporting this mode provide access to single 8-bit registers. For example:

```
LD    %FE,#HI STACK
                !load register %FE (SPH) with
                 the upper 8-bits of the label
                 STACK!
AND 0,MASK_REG
                !AND register 0 with register
                 named MASK_REG!
OR  1,R5    !OR register 1 with working
                 register 5!
```

Increment word (INCW) and decrement word (DECW) are the only two Z8 instructions which access 16-bit operands. These instructions are illustrated below for the direct register addressing mode.

```
INCW  RR0  !increment working register
                 pair R0, R1:
                 R1 ← R1 + 1
                 R0 ← R0 + carry!
DECW  %7E
                !decrement working register
                 pair %7E, %7F:
                 %7F ← %7F - 1
                 %7E ← %7E - carry!
```

Note that the instruction

```
INCW  RR5
```

will be flagged as an error by the assembler (RR5 not even-numbered).

2.4.2 Indirect Register Addressing. In this addressing mode, the operand is pointed to by the register whose 8-bit register address or 4-bit working register address is given by the instruction. This mode is used when the target register address is not known at assembly time and must be calculated during program execution. For example, assume registers %60-%7F contain a buffer for output to the serial line via repetitive calls to procedure SERIAL_OUT. SERIAL_OUT expects working register 0 to hold the output character. The following instructions illustrate the use of the indirect addressing mode to accomplish this task:

```
    LD    R1,#%20
                !working register 1 is the byte
                 counter: output %20 bytes!
    LD    R2,#%60
                !working register 2 is the buf-
                 fer pointer register!
out_again:
    LD    R0,@R2
                !load into working register 0
                 the byte pointed to by working
                 register 2!
    INC   R2    !increment pointer!
    CALL  SERIAL_OUT
                !output the byte!
    DJNZ  R1,out_again
                !loop till done!
```

Indirect addressing may also be used for accessing a 16-bit register pair via the INCW and DECW instructions. For example,

```
INCW  @R0 !increment the register pair
                 whose address is contained in
                 working register 0!
DECW  @%7F
                !decrement the register pair
                 whose address is contained in
                 register %7F!
```

The contents of registers R0 and %7F should be even numbers for proper access; when referencing a register pair, the least significant address bit is forced to the appropriate value by the Z8. However, the register used to point to the register pair need not be an even-numbered register.

Since the indirect addressing mode permits calculation of a target address prior to the desired register access, this mode may be used to simulate other, more complex addressing modes. For example, the instruction

```
SUB  4,BASE(R5)
```

requires the indexed addressing mode which is not directly supported by the Z8 SUBtract instruction. This instruction can be simulated as follows:

```
LD    R6,#BASE
                !working register 6 has the
                 base address!
ADD R6,R5  !calculate the target address!
SUB 4,@R6  !now use indirect addressing to
                 perform the actual subtract!
```

Any available register or working register may be used in place of R6 in the above example.

2.4.3 Indexed Addressing. The indexed addressing mode is supported by the load instruction (LD) for the transference of bytes between a working register and another register. The effective address of the latter register is given by the instruction which is offset by the contents of a designated working (index)

2. Accessing Register Memory (Continued)

register. This addressing mode provides efficient memory usage when addressing consecutive bytes in a block of register memory, such as a table or a buffer. The working register used as the index in the effective address calculation can serve the additional role of counter for control of a loop's duration.

For example, assume an ASCII character buffer exists in register memory starting at address BUF for LENGTH bytes. In order to determine the logical length of the character string, the buffer should be scanned backward until the first nonoccurrence of a blank character. The following code sequence may be used to accomplish this task:

```
    LD    R0,#LENGTH
                  !length of buffer!
                  !starting at buffer end, look for
                   1st non-blank!
loop:
    LD    R1,BUF-1(R0)
    CP    R1,#' '
    JR    ne,found
                  !found non-blank!
    DJNZ  R0,loop
                  !look at next!
all_blanks:       !length = 0!
found:
```

5 instructions
12 bytes
1.5 μs overhead
10.5 μs (average) per character tested

At labels "all_blanks" and "found," R0 contains the length of the character string. These labels may refer to the same location, but they are shown separately for an application where special processing is required for a string of zero length. To perform this task without indexed addressing would require a code sequence such as:

```
    LD    R1,#BUF+LENGTH-1
    LD    R0,#LENGTH
                  !starting at buffer end, look for
                   1st non-blank!
loop1:
    CP    @R1,#' '
    JR    ne,found1
                  !found non-blank!
    DEC   R1      !dec pointer!
    DJNZ  R0,loop1
                  !are we done?!
all_blanks1:      !length = 0!
found1:
```

6 instructions
13 bytes
3 μs overhead
9.5 μs (average) per character tested

The latter method requires one more byte of program memory than the former, but is faster by four execution cycles (1 μs) per character tested.

As an alternate example, assume a buffer exists as described above, but it is desired to scan this buffer forward for the first occurrence of an ASCII carriage return. The following illustrates the code to do this:

```
    LD    R0,#-LENGTH
                  !starting at buffer start, look for
                   1st carriage return (=%0D)!
next:
    LD    r1,BUF+LENGTH(R0)
    CP    R1,#%0D
    JR    eq,cr   !found it!
    INC   R0      !update counter/index!
    JR    nz,next
                  !try again!
cr:
    ADD   R0,#LENGTH
                  !R0 has length to CR!
```

7 instructions
16 bytes
1.5 μs overhead
12 μs (average) per character tested

SECTION 3

Accessing Program and External Data Memory

In a single instruction, the Z8 can transfer a byte between register memory and either program or external data memory. Load Constant (LDC) and Load Constant and Increment (LDCI) reference program memory; Load External (LDE) and Load External and Increment (LDEI) reference external data memory. These instructions require that a working register pair contain the address of the byte in either program or external data memory to be accessed by the instruction (indirect working register pair addressing mode). The register byte operand is specified by using the direct working register addressing mode in LDC and LDE or the indirect working register addressing mode in LDCI and LDEI. In addition to performing the designated byte transfer, LDCI and LDEI automatically increment both the indirect registers specified by the instruction. These instructions are therefore efficient for performing block moves between register and either program or external data memory. Since the indirect addressing mode is used to specify the operand address within program or external data memory, more complex addressing modes may be simulated as discussed earlier in Section 2.4.2. For example, the instruction

```
LDC   R3,BASE(R2)
```

requires the indexed addressing mode, where

3. Accessing Program and External Data Memory (Continued)

BASE is the base address of a table in program memory and R2 contains the offset from table start to the desired table entry. The following code sequence simulates this instruction with the use of two additional registers (R0 and R1 in this example).

```
LD    R0,#HI BASE
LD    R1,#LO BASE
          !RR0 has table start address!
ADD   R1,R2
ADC   R0,#0
          !RR0 has table entry address!
LDC   R3,@RR0
          !R3 has the table entry!
```

3.1 Configuring the Z8 for I/O Applications vs. Memory Intensive Applications. The Z8 offers a high degree of flexibility in memory and I/O intensive applications. Thirty-two port bits are provided of which 16, 12, eight, or zero may be configured as address bits to external memory. This allows for addressing of 62K, 4K or 256 bytes of external memory, which can be expanded to 124K, 8K, or 512 bytes if the Data Memory Select output ($\overline{DM}$) is used to distinguish between program and data memory accesses. The following instructions illustrate the code sequence required to configure the Z8 with 12 external addressing lines and to enable the Data Memory Select output.

```
LD    P01M,#%(2)00010010
          !bit 3-4: enable AD0-AD7;
           bit 0-1: enable A8-A11!
LD    P3M,#%(2)00001000
          !bit 3-4: enable DM!
```

The two bytes following the mode selection of ports 0 and 1 should not reference external memory due to pipelining of instructions within the Z8. Note that the load instruction to P3M satisfies this requirement (providing that it resides within the internal 2K bytes of memory).

3.2 LDC and LDE. To illustrate the use of the Load Constant (LDC) and Load External (LDE) instructions, assume there exists a hardware configuration with external memory and Data Memory Select enabled. The following module illustrates a program for tokenizing an ASCII input buffer. The program assumes there is a list of delimiters (space, comma, tab, etc.) in program memory at address DELIM for COUNT bytes (accessed via LDC) and that an ASCII input buffer exists in external data memory (accessed via LDE). The program scans the input buffer from the current location and returns the start address of the next token (i.e. the address of the first nondelimiter found) and the length of that token (number of characters from token start to next delimiter).

```
Z8ASM     2.0
LOC     OBJ CODE    STMT SOURCE STATEMENT

                      1 SCAN     MODULE
                      2 CONSTANT
                      3  COUNT  :=      6
                      4 GLOBAL
                      5          $SECTION PROGRAM
P 0000 20  3B  2C     6 DELIM    ARRAY   [COUNT BYTE]    :=
P 0003 2E  0A  0D
                      7                   [' ' , ';' , ',' , '.' , %0A , %0D]
                      8
P 0006                9 scan     PROCEDURE
                     10 !****************************************************
                     11  Purpose =       To find the next token within an
                     12                   ASCII buffer.
                     13
                     14  Input =         RR0 = address of current location
                     15                       within input buffer in external
                     16                       memory.
                     17
                     18  Output =        RR4 = address of start of next token
                     19                   RR0 = address of new token's ending
                     20                       delimiter
                     21                   R2  = length of token
                     22                   R3  = ending delimiter
                     23                   R6,R7,R8,R9 destroyed
                     24
                     25 ****************************************************!
                     26 ENTRY
P 0006 B0  E2        27          clr      R2       !init. length counter!
                     28         DO
P 0008 82  30        29          LDE      R3,@RR0 !get byte from input buffer!
P 000A A0  E0        30          incw     RR0      !increment pointer!
P 000C D6  002E'     31          call     check    !look for non-delimiter'!
P 000F FD  0015'     32          IF   C THEN
P 0012 8D  0018'     33            EXIT           !found token start!
                     34          FI
P 0015 8D  0008'     35         OD
```

```
                        36
P 0018 48  E0           37          ld      R4,R0
P 001A 58  E1           38          ld      R5,R1   !RR4 = token starting addr!
                        39        DO
P 001C 2E               40          inc     R2      !inc. length counter!
P 001D 82  30           41          LDE     R3,@RR0 !get next input byte!
P 001F D6  002E'        42          call    check   !look for delimiter!
P 0022 7D  0028'        43          IF  NC  THEN
P 0025 8D  002D'        44            EXIT          !found token end!
                        45          FI
P 0028 A0  E0           46          incw    RR0     !point to next byte!
P 002A 8D  001C'        47        OD
                        48
P 002D AF               49          ret
P 002E                  50 END      scan
                        51
P 002E                  52 check    PROCEDURE
                        53 !****************************************************
                        54  Purpose =       compare current character with
                        55                  delimiter table until table
                        56                  end or match found
                        57
                        58  input =         DELIM = start address of table
                        59                  COUNT = length of that table
                        60                  R3 = byte to be scrutinized
                        61
                        62  output =        Carry flag = 1 => input byte
                        63                  is not a delimiter (no match found)
                        64
                        65                  Carry flag = 0 => input byte
                        66                  is a delimiter (match found)
                        67                  R6,R7,R8,R9  destroyed
                        68
                        69 ****************************************************!
                        70 ENTRY
P 002E 6C  00*          71          ld      R6,#HI DELIM
P 0030 7C  00*          72          ld      R7,#LO DELIM    !RR6 points to
                        73                                   delimiter list!
P 0032 8C  06           74          ld      R8,#COUNT       !R8 = length of list!
                        75 here:
P 0034 C2  96           76          LDC     R9,@RR6         !get table entry!
P 0036 A0  E6           77          incw    RR6             !point to next entry!
P 0038 A2  93           78          cp      R9,R3           !R3 = delimiter?!
P 003A 6B  03           79          jr      eq,bye          !yes. carry = 0!
P 003C 8A  F6           80          djnz    R8,here         !next entry!
P 003E DF               81          scf                     !table done. R3
                        82                                   not a delimiter!
                        83 bye:
P 003F AF               84          ret
P 0040                  85 END      check
                        86 END      SCAN

   0 ERRORS
ASSEMBLY COMPLETE
```

27 instructions
58 bytes
Execution time is a function of the number of leading delimiters before token start (x) and the number of characters in the token (y): 123 μs overhead + 59x μs + 102y μs (average) per token

3.3 LDCI. A common function performed in Z8 applications is the initialization of the register space. The most obvious approach to this function is the coding of a sequence of "load register with immediate value" instructions (each occupying three program bytes for a register or two program bytes for a working register). This approach is also the most efficient technique for initializing less than eight consecutive registers or 14 consecutive working registers. For a larger register block, the

3. Accessing Program and External Data Memory (Continued)

LDCI instruction provides an economical means of initializing consecutive registers from an initialization table in program memory. The following code excerpt illustrates this technique of initializing control registers %F2 through %FF from a 14-byte array (INIT_tab) in program memory:

```
SRP   #%00
            !RP not %F0!
LD    R6,#HI INIT_tab
LD    R7,#LO INIT_tab
LD    R8,#%F2
            !1st reg to be initialized!
LD    R9,#14
            !length of register block!
loop:
LDCI  @R8,@RR6
            !load a register from the
            init table!
DJNZ  R9,loop
            !continue till done!
```

7 instructions
14 bytes
7.5 μs overhead
7.5 μs per register initialized

3.4 LDEI. The LDEI instruction is useful for moving blocks of data between external and register memory since auto-increment is performed on both indirect registers designated by the instruction. The following code excerpt illustrates a register buffer being saved at address %40 through %60 into external memory at address SAVE:

```
LD    R10,#HI SAVE
            !external memory!
LD    R11,#LO SAVE
            !address!
LD    R8,#%40
            !starting register!
LD    R9,#%21
            !number of registers to save in
             external data memory!
loop:
LDEI  @RR10,@R8
            !init a register!
DJNZ  R9,loop
            !until done!
```

6 instructions
12 bytes
6 μs overhead
7.5 μs per register saved

SECTION 4

Bit Manipulations

Support of the test and modification of an individual bit or group of bits is required by most software applications suited to the Z8 microcomputer. Initializing and modifying the Z8 control registers, polling interrupt requests, manipulating port bits for control of or communication with attached devices, and manipulation of software flags for internal control purposes are all examples of the heavy use of bit manipulation functions. These examples illustrate the need for such functions in all areas of the Z8 register space. These functions are supported in the Z8 primarily by six instructions:

- Test under Mask (TM)
- Test Complement under Mask (TCM)
- AND
- OR
- XOR
- Complement (COM)

These instructions may access any Z8 register, regardless of its inherent type (control, I/O, or general purpose), with the exception of the six write-only control registers (PRE0, PRE1, P01M, P2M, P3M, IPR) mentioned earlier in Section 2.1. Table 1 summarizes the function performed on the destination byte by each of the above instructions. All of these instructions, with the exception of COM, require a mask operand. The "selected" bits referenced in Table 1 are those bits in the destination operand for which the corresponding mask bit is a logic 1.

Opcode	Use
TM	To test selected bits for logic 0
TCM	To test selected bits for logic 1
AND	To reset all but selected bits to logic 0
OR	To set selected bits to logic 1
XOR	To complement selected bits
COM	To complement all bits

Table 1. Bit Manipulation Instruction Usage

The instructions AND, OR, XOR, and COM have functions common to today's microprocessors and therefore are not described in depth here. However, examples of the use of these instructions are laced throughout the remainder of this document, thus giving an integrated view of their uses in common functions. Since they are unique to the Z8, the functions of Test under Mask and Test Complement under Mask, are discussed in more detail next.

4.1 Test under Mask (TM). The Test under Mask instruction is used to test selected bits for logic 0. The logical operation performed is

destination AND source

Neither source nor destination operand is modified; the FLAGS control register is the only register affected by this instruction. The zero flag (Z) is set if all selected bits are logic 0; it is reset otherwise. Thus, if the selected destination bits are either all logic 1 or a combination of 1s and 0s, the zero flag would be cleared by this instruction. The sign flag (S) is either set or reset to reflect the result of the

4. Bit Manipulations (Continued)

AND operation; the overflow flag (V) is always reset. All other flags are unaffected. Table 2 illustrates the flag settings which result from the TM instruction on a variety of source and destination operand combinations. Note that a given TM instruction will never result in both the Z and S flags being set.

4.2 Test Complement under Mask. The Test Complement under Mask instruction is used to test selected bits for logic 1. The logical operation performed is

(NOT destination) AND source.

As in Test under Mask, the FLAGS control register is the only register affected by this operation. The zero flag (Z) is set if all selected destination bits are 1; it is reset otherwise. The sign flag (S) is set or reset to reflect the result of the AND operation; the overflow flag (V) is always reset. Table 3 illustrates the flag settings which result from the TCM instruction on a variety of source and destination operand combinations. As with the TM instruction, a given TCM instruction will never result in both the Z and S flags being set.

Destination	Source	Flags		
(binary)	(binary)	Z	S	V
10001100	01110000	1	0	0
01111100	01110000	0	0	0
10001100	11110000	0	1	0
11111100	11110000	0	1	0
00011000	10100001	1	0	0
01000000	10100001	1	0	0

Table 2. Effects of the TM Instruction

Destination	Source	Flags		
(binary)	(binary)	Z	S	V
10001100	01110000	0	0	0
01111100	01110000	1	0	0
10001100	11110000	0	0	0
11111100	11110000	1	0	0
00011000	10100001	0	1	0
01000000	10100001	0	1	0

Table 3. Effects of the TCM Instruction

Stack Operations

The Z8 stack resides within an area of data memory (internal or external). The current address in the stack is contained in the stack pointer, which decrements as bytes are pushed onto the stack, and increments as bytes are popped from it. The stack pointer occupies two control register bytes (%FE and %FF) in the Z8 register space and may be manipulated like any other register. The stack is useful for subroutine calls, interrupt service routines, and parameter passing and saving. Figure 2 illustrates the downward growth of a stack as bytes are pushed onto it.

5.1 Internal vs. External Stack. The location of the stack in data memory may be selected to be either internal register memory or external data memory. Bit 2 of control register P01M (%F8) controls this selection. Register pair SPH (%FE), SPL (%FF) serves as the stack pointer for an external stack. Register SPL is the stack pointer for an internal stack. In the latter configuration, SPH is available for use as a data register. The following illustrates a code sequence that initializes external stack operations:

```
LD P01M,#%(2)00000000
                !bit 2: select external stack!
LD SPH,#HI STACK
LD SPL,#LO STACK
```

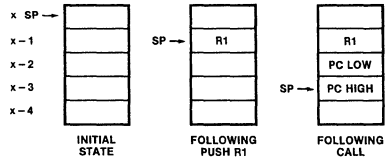


Figure 2. Growth of a Stack

5.2 CALL. A subroutine call causes the current Program Counter (the address of the byte following the CALL instruction) to be pushed onto the stack. The Program Counter is loaded with the address specified by the CALL instruction. This address may be a direct address or an indirect register pair reference. For example,

```
LABEL 1: CALL   %4F98
                !direct addressing: PC is
                 loaded with the hex value
                 4F98;
                 address LABEL 1 + 3 is pushed
                 onto the stack!

LABEL 2: CALL   @RR4
                !indirect addressing: PC is
                 loaded with the contents of
                 working register pair R4, R5;
                 address LABEL 2 + 2 is pushed
                 onto the stack!
```

5. Stack Operations (Continued)

```
LABEL 3: CALL   @%7E
                !indirect addressing: PC is
                 loaded with the contents of
                 register pair %7E, %7F;
                 address'LABEL 3+2 is pushed
                 onto the stack!
```

5.3 RET. The return (RET) instruction causes the top two bytes to be popped from the stack and loaded into the Program Counter. Typically, this is the last instruction of a subroutine and thus restores the PC to the address following the CALL to that subroutine.

5.4 Interrupt Machine Cycle. During an interrupt machine cycle, the PC followed by the status flags is pushed onto the stack. (A more detailed discussion of interrupt processing is provided in Section 6.)

5.5 IRET. The interrupt return (IRET) instruction causes the top byte to be popped from the stack and loaded into the status flag register, FLAGS (%FC); the next two bytes are then popped and loaded into the Program Counter. In this way, status is restored and program execution continues where it had left off when the interrupt was recognized.

5.6 PUSH and POP. The PUSH and POP instructions allow the transfer of bytes between the stack and register memory, thus providing program access to the stack for saving and restoring needed values and passing parameters to subroutines.

Execution of a PUSH instruction causes the stack pointer to be decremented by 1; the operand byte is then loaded into the location pointed to by the decremented stack pointer. Execution of a POP instruction causes the byte addressed by the stack pointer to be loaded into the operand byte; the stack pointer is then incremented by 1. In both cases, the operand byte is designated by either a direct register address or an indirect register reference. For example:

```
PUSH R1     !direct address: push working
             register 1 onto the stack!
POP  5      !direct address: pop the top
             stack byte into register 5!
PUSH @R4    !indirect address: pop the top
             stack byte into the byte
             pointed to by working reg-
             ister 4!
PUSH @17    !indirect address: push onto
             the stack the byte pointed to
             by register 17!
```

SECTION 6

Interrupts

The Z8 recognizes six different interrupts from four internal and four external sources, including internal timer/counters, serial I/O, and four Port 3 lines. Interrupts may be individually or globally enabled/disabled via Interrupt Mask Register IMR (%FB) and may be prioritized for simultaneous interrupt resolution via Interrupt Priority Register IPR (%F9). When enabled, interrupt request processing automatically vectors to the designated service routine. When disabled, an interrupt request may be polled to determine when processing is needed.

6.1 Interrupt Initialization. Before the Z8 can recognize interrupts following RESET, some initialization tasks must be performed. The initialization routine should configure the Z8 interrupt requests to be enabled/disabled, as required by the target application and assigned a priority (via IPR) for simultaneous enabled-interrupt resolution. An interrupt request is enabled if the corresponding bit in the IMR is set (= 1) and interrupts are globally enabled (bit 7 of IMR = 1). An interrupt request is disabled if the corresponding bit in the IMR is reset (= 0) or interrupts are globally disabled (bit 7 of IMR = 0).

A RESET of the Z8 causes the contents of the Interrupt Request Register IRQ (%FA) to be held to zero until the execution of an EI instruction. Interrupts that occur while the Z8 is in this initial state will not be recognized, since the corresponding IRQ bit cannot be set. The EI instruction is specially decoded by the Z8 to enable the IRQ; simply setting bit 7 of IMR is therefore *not* sufficient to enable interrupt processing following RESET. However, subsequent to this initial EI instruction, interrupts may be globally enabled either by the instruction

```
EI          !enable interrupts!
```

or by a register manipulation instruction such as

```
OR   IMR,#%80
```

To globally disable interrupts, execute the instruction

```
DI          !disable interrupts!
```

This will cause bit 7 of IMR to be reset.

Interrupts *must* be globally disabled prior to any modification of the IMR, IPR or enabled bits of the IRQ (those corresponding to enabled interrupt requests), unless it can be *guaranteed* that an enabled interrupt will not occur during the processing of such instructions. Since interrupts represent the occurrence of events asynchronous to program execution, it is highly unlikely that such a guarantee can be made reliably.

6.2 Vectored Interrupt Processing. Enabled interrupt requests are processed in an automatic vectored mode in which the interrupt service routine address is retrieved from within the first 12 bytes of program memory. When an enabled interrupt request is recognized by the Z8, the Program Counter is pushed onto the stack (low order 8 bits first, then high-order 8 bits) followed by the FLAGS register (#%FC). The corresponding interrupt request bit is reset in IRQ, interrupts are globally disabled (bit 7 of IMR is reset), and an indirect jump is taken on the word in location 2x, 2x + 1 (x = interrupt request number, $0 \leq x \leq 5$). For example, if the bytes at addresses %0004 and %0005 contain %05 and %78 respectively, the interrupt machine cycle for IRQ2 will cause program execution to continue at address %0578.

When interrupts are sampled, more than one interrupt may be pending. The Interrupt Priority Register (IPR) controls the selection of the pending interrupt with highest priority. While this interrupt is being serviced, a higher-priority interrupt may occur. Such interrupts may be allowed service within the current interrupt service routine (nested) or may be held until the current service routine is complete (non-nested).

To allow nested interrupt processing, interrupts must be selectively enabled upon entry to an interrupt service routine. Typically, only higher-priority interrupts would be allowed to nest within the current interrupt service. To do this, an interrupt routine must "know" which interrupts have a higher priority than the current interrupt request. Selection of such nesting priorities is usually a reflection of the priorities established in the Interrupt Priority Register (IPR). Given this data, the first instructions executed in the service routine should be to save the current Interrupt Mask Register, mask off all interrupts of lower and equal priority, and globally enable interrupts (EI). For example, assume that service of interrupt requests 4 and 5 are nested within the service of interrupt request 3. The following illustrates the code required to enable IRQ4 and IRQ5:

```
CONSTANT
        INT_MASK_3          :=      %(2) 00110000
GLOBAL
IRQ3_service        PROCEDURE       ENTRY
!service routine for IRQ3!
        PUSH IMR                    !save Interrupt Mask Register!
                !interrupts were globally disabled during the interrupt
                 machine cycle - no DI is needed prior to modification of IMR!
        AND  IMR,#INT_MASK_3        !disable all but IRQ4 & 5!
        EI
        !...!       !service interrupt!
                    !interrupts are globally enabled now — must disable them prior to
                     modification of IMR!
        DI
        POP  IMR                    !restore entry IMR!
        IRET
END IRQ3_service
```

Note that IRQ4 and IRQ5 are enabled by the above sequence only if their respective IMR bits = 1 on entry to IRQ3_service.

The service routine for an interrupt whose processing is to be completed without interruption should not allow interrupts to be nested within it. Therefore, it need not modify the IMR, since interrupts are disabled automatically during the interrupt machine cycle.

The service routine for an enabled interrupt is typically concluded with an IRET instruction, which restores the FLAGS register and Program Counter from the top of the stack and globally enables interrupts. To return from an interrupt service routine without re-enabling interrupts, the following code sequence could be used:

```
POP   FLAGS
              !FLAGS ← @SP!
RET           !PC ← @SP!
```

This accomplishes all the functions of IRET, except that IMR is not affected.

6.3 Polled Interrupt Processing Disabled interrupt requests may be processed in a polled mode, in which the corresponding bits of the Interrupt Request Register (IRQ) are examined by the software. When an interrupt request bit is found to be a logic 1, the interrupt should be processed by the appropriate

service routine. During such processing, the interrupt request bit in the IRQ must be cleared by the software in order for subsequent interrupts on that line to be distinguished from the current one. If more than one interrupt request is to be processed in a polled mode, polling should occur in the order of established priorities. For example, assume that IRQ0, IRQ1, and IRQ4 are to be polled and that established priorities are, from high to low, IRQ4, IRQ0, IRQ1. An instruction sequence like the following should be used to poll and service the interrupts:

```
!...!
 !poll interrupt inputs here!
              TCM     IRQ, #%(2)00010000          !IRQ4 need service?!
              JR      NZ, TEST0                   !no!
              CALL    IRQ4__service               !yes!
TEST0:        TCM     IRQ, #%(2)00000001          !IRQ0 need service?!
              JR      NZ, TEST1                   !no!
              CALL    IRQ0__service               !yes!
TEST1:        TCM     IRQ, #%(2)00000010          !IRQ1 need service?!
              JR      NZ, DONE                    !no!
              CALL    IRQ1__service               !yes!
DONE:         !...!

IRQ4__service         PROCEDURE          ENTRY
              !...!
              AND     IRQ, #%(2)11101111          !clear IRQ4!
              !...!
              RET
END IRQ4__service

IRQ0__service         PROCEDURE          ENTRY
              !...!
              AND     IRQ, #%(2)11111110          !clear IRQ0!
              !...!
              RET
END IRQ0__service

IRQ1__service         PROCEDURE          ENTRY
              !...!
              AND     IRQ, #%(2)11111101          !clear IRQ1!
              !...!
              RET
END IRQ1__service
!...!
```

SECTION 7

Timer/Counter Functions

The Z8 provides two 8-bit timer/counters, T_0 and T_1, which are adaptable to a variety of application needs and thus allow the software (and external hardware) to be relieved of the bulk of such tasks. Included in the set of such uses are:

- Interval delay timer
- Maintenance of a time-of-day clock
- Watch-dog timer
- External event counting
- Variable pulse train output
- Duration measurement of external event
- Automatic delay following external event detection

Each timer/counter is driven by its own 6-bit prescaler, which is in turn driven by the internal Z8 clock divided by four. For T_1, the internal clock may be gated or triggered by an external event or may be replaced by an external clock input. Each timer/counter may operate in either single-pass or continuous mode where, at end-of-count, either counting stops or the counter reloads and continues counting. The counter and prescaler registers may be altered individually while the timer/counter is running; the software controls whether the new values are loaded immediately or when end-of-count (EOC) is reached.

Although the timer/counter prescaler registers (PRE0 and PRE1) are write-only, there is a technique by which the timer/

counters may simulate a readable prescaler. This capability is a requirement for high resolution measurement of an event's duration. The basic approach requires that one timer/counter be initialized with the desired counter and prescaler values. The second timer/counter is initialized with a counter equal to the prescaler of the first timer/counter and a prescaler of 1. The second timer/counter must be programmed for continuous mode. With both timer/counters driven by the internal clock and started and stopped simultaneously, they will run synchronous to one another; thus, the value read from the second counter will always be equivalent to the prescaler of the first.

7.1 Time/Count Interval Calculation To determine the time interval (i) until EOC, the equation

$$i = t \times p \times v$$

characterizes the relation between the prescaler (p), counter (v), and clock input period (t); t is given by

$$1/(XTAL/8)$$

where XTAL is the Z8 input clock frequency; p is in the range $1-64$; v is in the range $1-256$. When programming the prescaler and counter registers, the maximum load value is truncated to six and eight bits, respectively, and is therefore programmed as zero. For an input clock frequency of 8 MHz, the prescaler and counter register values may be programmed to time an interval in the range

$$1\ \mu s \times 1 \times 1 \leq i \leq 1\ \mu s \times 64 \times 256$$

$$1\ \mu s \leq i \leq 16.384\ ms$$

To determine the count (c) until EOC for T_1 with external clock input, the equation

$$c = p \times v$$

characterizes the relation between the T_1 prescaler (p) and the T_1 counter (v). The divide-by-8 on the input frequency is bypassed in this mode. The count range is

$$1 \times 1 \leq c \leq 64 \times 256$$

$$1 \leq c \leq 16{,}384$$

7.2 T_{OUT} Modes. Port 3, bit 6 ($P3_6$) may be configured as an output (T_{OUT}) which is dynamically controlled by one of the following:

- T_0
- T_1
- Internal clock

When driven by T_0 or T_1, T_{OUT} is reset to a logic 1 when the corresonding load bit is set in timer control register TMR (%F1) and toggles on EOC from the corresponding counter.

When T_{OUT} is driven by the internal clock, that clock is directly output on $P3_6$.

While programmed as T_{OUT}, $P3_6$ is disabled from being modified by a write to port register %03; however, its current output may be examined by the Z8 software by a read to port register %03.

7.3 T_{IN} Modes. Port 3, bit 1 ($P3_1$) may be configured as an input (T_{IN}) which is used in conjunction with T_1 in one of four modes:

- External clock input
- Gate input for internal clock
- Nonretriggerrable input for internal clock
- Retriggerable input for internal clock

For the latter two modes, it should be noted that the existence of a synchronizing circuit within the Z8 causes a delay of two to three internal clock periods following an external trigger before clocking of the counter actually begins.

Each High-to-Low transition on T_{IN} will generate interrupt request IRQ2, regardless of the selected T_{IN} mode or the enabled/disabled state of T_1. IRQ2 must therefore be masked or enabled according to the needs of the application.

The "external clock input" T_{IN} mode supports the counting of external events, where an event is seen as a High-to-Low transition on T_{IN}. Interrupt request IRQ5 is generated on the n^{th} occurrence (single-pass mode) or on every n^{th} occurrence (continuous mode) of that event.

The "gate input for internal clock" T_{IN} mode provides for duration measurement of an external event. In this mode, the T_1 prescaler is driven by the Z8 internal clock, gated by a High level on T_{IN}. In other words, T_1 will count while T_{IN} is High and stop counting while T_{IN} is Low. Interrupt request IRQ2 is generated on the High-to-Low transition on T_{IN}. Interrupt request IRQ5 is generated on T_1 EOC. This mode may be used when the width of a High-going pulse needs to be measured. In this mode, IRQ2 is typically the interrupt request of most importance, since it signals the end of the pulse being measured. If IRQ5 is generated prior to IRQ2 in this mode, the pulse width on T_{IN} is too large for T_1 to measure in a single pass.

The "nonretriggerable input" T_{IN} mode provides for automatic delay timing following an external event. In this mode, T_1 is loaded and clocked by the Z8 internal clock following the first High-to-Low transition on T_{IN} after T_1 is enabled. T_{IN} transitions that occur after this point do not affect T_1. In single-pass mode, the

7. Timer/ Counter Functions (Continued)

enable bit is reset on EOC; further T_{IN} transitions will not cause T_1 to load and begin counting until the software sets the enable bit again. In continuous mode, EOC does not modify the enable bit, but the counter is reloaded and counting continues immediately; IRQ5 is generated every EOC until software resets the enable bit. This T_{IN} mode may be used, for example, to time the line feed delay following end of line detection on a printer or to delay data sampling for some length of time following a sample strobe.

The "retriggerable input" T_{IN} mode will load and clock T_1 with the Z8 internal clock on every occurrence of a High-to-Low transition on T_{IN}. T_1 will time-out and generate interrupt request IRQ5 when the programmed time interval (determined by T_1 prescaler and load register values) has elapsed since the last High-to-Low transition on T_{IN}. In single-pass mode, the enable bit is reset on EOC; further T_{IN} transitions will not cause T_1 to load and begin counting until the software sets the enable bit again. In continuous mode, EOC does not modify the enable bit, but the counter is reloaded and counting continues immediately; IRQ5 is generated at every EOC until the software resets the enable bit. This T_{IN} mode may provide such functions as watch-dog timer (e.g., interrupt if conveyor belt stopped or clock pulse missed), or keyboard time-out (e.g., interrupt if no input in x ms).

7.4 Examples. Several possible uses of the timer/counters are given in the following four examples.

7.4.1 Time of Day Clock. The following module illustrates the use of T_1 for maintenance of a time of day clock, which is kept in binary format in terms of hours, minutes, seconds, and hundredths of a second. It is desired that the clock be updated once every hundredth of a second; therefore, T_1 is programmed in continuous mode to interrupt 100 times a second. Although T_1 is used for this example, T_0 is equally suited for the task.

The procedure for initializing the timer (TOD_INIT), the interrupt service routine (TOD) which updates the clock, and the interrupt vector for T_1 end-of-count (IRQ_5) are illustrated below. XTAL = 7.3728 MHz is assumed.

```
Z8ASM      2.0
LOC     OBJ CODE    STMT SOURCE STATEMENT

                      1 TIMER1  MODULE
                      2 CONSTANT
                      3  HOUR   :=      R12
                      4  MINUTE :=      R13
                      5  SECOND :=      R14
                      6  HUND   :=      R15
                      7         $SECTION PROGRAM
                      8 GLOBAL
                      9 !IRQ5 interrupt vector!
                     10         $ABS     10
P 0000 000F'         11 IRQ_5    ARRAY    [1 WORD]  :=    [TOD]
                     12
                     13         $REL
P 000C               14 TOD_INIT         PROCEDURE
                     15 ENTRY
P 0000 E6  F3  93    16         LD       PRE1,#%(2)10010011
                     17                             !bit 2-7: prescaler = 36;
                     18                              bit 1: internal clock;
                     19                              bit 0: continuous mode!
P 0003 E6  F2  00    20         LD       T1,#0      !(256) time-out =
                     21                              1/100 second!
P 0006 46  F1  0C    22         OR       TMR,#%0C   !load, enable T1!
P 0009 8F            23         DI
P 000A 46  FB  20    24         OR       IMR,#%20   !enable T1 interrupt!
P 000D 9F            25         EI
P 000E AF            26         RET
P 000F               27 END      TOD_INIT
                     28
P 000F               29 TOD      PROCEDURE
                     30 ENTRY
P 000F 70  FD        31         PUSH     RP
                     32 !Working register file %10 to %1F contains
                     33  the time of day clock!
P 0011 31  10        34         SRP      #%10
P 0013 FE            35         INC      HUND              !1 more .01 sec!
P 0014 A6  EF  64    36         CP       HUND,#100         !full second yet?!
P 0017 EB  13        37         JR       NE,TOD_EXIT       !jump if no!
P 0019 B0  EF        38         CLR      HUND
P 001B EE            39         INC      SECOND            !1 more second!
P 001C A6  EE  3C    40         CP       SECOND,#60        !full minute yet?!
P 001F EB  0B        41         JR       NE,TOD_EXIT       !jump if no!
```

```
P 0021 B0  EE          42          CLR     SECOND
P 0023 DE              43          INC     MINUTE          !1 more minute!
P 0024 A6  ED  3C      44          CP      MINUTE,#60      !full hour yet?!
P 0027 EB  03          45          JR      NE,TOD_EXIT     !jump if no!
P 0029 B0  ED          46          CLR     MINUTE
P 002B CE              47          INC     HOUR
                       48 TOD_EXIT:
P 002C 50  FD          49          POP     RP              !restore entry RP!
P 002E BF              50          IRET
P 002F                 51 END      TOD
                       52 END      TIMER1

   0 ERRORS
ASSEMBLY COMPLETE
```

TOD_INIT:
7 instructions
15 bytes
16 μs

TOD:
17 instruction
32 bytes
19.5 μs (average) including interrupt response time

7.4.2 Variable Frequency, Variable Pulse Width Output. The following module illustrates one possible use of T_{OUT}. Assume it is necessary to generate a pulse train with a 10% duty cycle, where the output is repetitively high for 1.6 ms and then low for 14.4 ms. To do this, T_{OUT} is controlled by end-of-count from T_1, although T_0 could alternately be chosen. This example makes use of the Z8 feature that allows a timer's counter register to be modified without disturbing the count in progress. In continuous mode, the new value is loaded when T_1 reaches EOC. T_1 is first loaded and enabled with values to generate the short interval. The counter register is then immediately modified with the value to generate the long interval; this value is loaded into the counter automatically on T_1 EOC. The prescaler selected value must be the same for both long and short intervals. Note that the initial loading of the T_1 counter register is followed by setting the T_1 load bit of timer control register TMR (%F1); this action causes T_{OUT} to be reset to a logic 1 output. Each subsequent modification of the T_1 counter register does not affect the current T_{OUT} level, since the T_1 load bit is NOT altered by the software. The new value is loaded on EOC, and T_{OUT} will toggle at that time. The T_1 interrupt service routine should simply modify the T_1 counter register with the new value, alternating between the long and short interval values.

In the example which follows, bit 0 of register %04 is used as a software flag to indicate which value was loaded last. This module illustrates the procedure for T_1/T_{OUT} initialization (PULSE_INIT), the T_1 interrupt service routine (PULSE), and the interrupt vector for T_1 EOC (IRQ_5). XTAL = 8 MHz is assumed.

```
Z8ASM     2.0
LOC    OBJ CODE   STMT SOURCE STATEMENT

                    1 TIMER2  MODULE
                    2         $SECTION PROGRAM
                    3 GLOBAL
                    4 !IRQ5 interrupt vector!
                    5         $ABS    10
P 0000 0017'        6 IRQ_5   ARRAY   [1 WORD]  :=    [PULSE]
                    7
                    8         $REL
P 000C              9 PULSE_INIT      PROCEDURE
                   10 ENTRY
P 0000 E6  F3  03  11         LD      PRE1,#%(2)00000011
                   12                         !bit 2-7: prescaler = 64;
                   13                          bit 1: internal clock;
                   14                          bit 0: continuous mode!
P 0003 E6  F7  00  15         LD      P3M,#00   !bit 5: let P36 be Tout!
P 0006 E6  F2  19  16         LD      T1,#25          !for short interval!
P 0009 8F          17         DI
P 000A 46  FB  20  18         OR      IMR,#%(2)00100000  !enable T1 interrupt!
P 000D E6  F1  8C  19         LD      TMR,#%(2)10001100
                   20                         !bit 6-7: Tout controlled
                   21                                   by T1;
                   22                          bit 3: enable T1;
                   23                          bit 2: load T1 !
                   24 !Set long interval counter, to be loaded on T1 EOC!
P 0010 E6  F2  E1  25         LD      T1,#225
                   26 !Clear alternating flag for PULSE!
```

```
P 0013 B0  04          27          CLR     %04             != 0 : 25 next;
                       28                                   = 1 : 225 next !
P 0015 9F              29          EI
P 0016 AF              30          RET
P 0017                 31 END      PULSE_INIT
                       32
                       33
P 0017                 34 PULSE    PROCEDURE
                       35 ENTRY
P 0017 E6  F2  E1      36          LD      T1,#225         !new load value!
P 001A B6  04  01      37          XOR     %04,#1          !which value next?!
P 001D 6B  03          38          JR      Z,PULSE_EXIT    !should be 225!
P 001F E6  F2  19      39          LD      T1,#25          !should be 25!
                       40 PULSE_EXIT:
P 0022 BF              41          IRET
P 0023                 42 END      PULSE
                       43 END      TIMER2

   0 ERRORS
ASSEMBLY COMPLETE
```

PULSE_INIT:
10 instructions
23 bytes
23 µs

PULSE:
5 instructions
12 bytes
25 µs (average) including interrupt response time

7.4.3 Cascaded Timer/Counters. For some applications it may be necessary to measure a greater time interval than a single timer/ counter can measure (16.384 ms). In this case, T_{IN} and T_{OUT} may be used to cascade T_0 and T_1 to function as a single unit. T_{OUT}, programmed to toggle on T_0 end-of-count, should be wired back to T_{IN}, which is selected as the external clock input for T_1. With T_0 programmed for continuous mode, T_{OUT} (and therefore T_{IN}) goes through a High-to-Low transition (causing T_1 to count) on every other T_0 EOC. Interrupt request IRQ5 is generated when the programmed time interval has elapsed. Interrupt requests IRQ2 (generated on every T_{IN} High-to-Low transition) and IRQ4 (generated on T_0 EOC) are of no importance in this application and are therefore disabled.

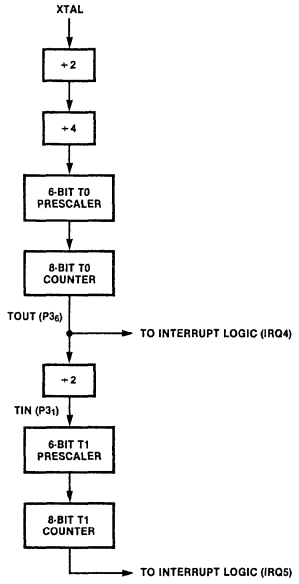


Figure 3. Cascaded Timer/Counters

To determine the time interval (i) until EOC, the equation

$$i = t \times p0 \times v0 \times (2 \times p1 \times v1 - 1)$$

characterizes the relation between the T_0 prescaler (p0) and counter (v0), the T_1 prescaler (p1) and counter (v1), and the clock input period (t); t is defined in Section 7.1. Assuming XTAL = 8 MHz, the measurable time interval range is

$$1\ \mu s \times 1 \times 1 \times (2 \times 1 - 1) \leq i \leq 1\ \mu s \times 64 \times 256 \times (2 \times 64 \times 256 - 1)$$

$$1\ \mu s \leq i \leq 536.854528\ s$$

Figure 3 illustrates the interconnection between T_0 and T_1. The following module illustrates the procedure required to initialize the timers for a 1.998 second delay interval:

```
Z8ASM      2.0
LOC    OBJ CODE    STMT SOURCE STATEMENT

                     1 TIMER3  MODULE
                     2 GLOBAL
P 0000               3 TIMER_16        PROCEDURE
                     4 ENTRY
P 0000 E6  F3  28    5         LD      PRE1,#%(2)00101000
                     6                         !bit 2-7: prescaler = 10;
                     7                          bit 1: external clock;
                     8                          bit 0: single-pass mode!
P 0003 E6  F7  00    9         LD      P3M,#00   !bit 5: let P36 be Tout!
P 0006 E6  F2  64   10         LD      T1,#100          !T1 counter register!
P 0009 E6  F5  29   11         LD      PRE0,#%(2)00101001
                    12                         !bit 2-7: prescaler = 10;
                    13                          bit 0: continuous mode!
P 000C E6  F4  64   14         LD      T0,#100          !T0 counter register!
P 000F 8F           15         DI
P 0010 56  FB  2B   16         AND     IMR,#%(2)00101011  !disable IRQ2 (Tin);
                    17                                          and IRQ4 (T0) !
P 0013 46  FB  20   18         OR      IMR,#%(2)00100000  !enable IRQ5 (T1)!
P 0016 9F           19         EI
P 0017 E6  F1  4F   20         LD      TMR,#%(2)01001111
                    21                         !bit 6-7: Tout controlled
                    22                                   by T0;
                    23                          bit 4-5: Tin mode is ext.
                    24                                   clock input;
                    25                          bit 3: enable T1;
                    26                          bit 2: load T1;
                    27                          bit 1: enable T0;
                    28                          bit 0: load T0 !
P 001A AF           29         RET
P 001B              30 END     TIMER_16
                    31 END     TIMER3

   0 ERRORS
ASSEMBLY COMPLETE

11 instructions
27 bytes
26.5 µs
```

7.4.4 Clock Monitor. T_1 and T_{IN} may be used to monitor a clock line (in a diskette drive, for example) and generate an interrupt request when a clock pulse is missed. To accomplish this, the clock line to be monitored is wired to $P3_1$ (T_{IN}). T_{IN} should be programmed as a retriggerable input to T_1, such that each falling edge on T_{IN} will cause T_1 to reload and continue counting. If T_1 is programmed to time-out after an interval of one-and-a-half times the clock period being monitored, T_1 will time-out and generate interrupt request IRQ5 only if a clock pulse is missed.

The following module illustrates the procedure for initializing T_1 and T_{IN} (MONITOR__INIT) to monitor a clock with a period of 2 µs. XTAL = 8 MHz is assumed. Note that this example selects single-pass rather than continuous mode for T_1. This is to prevent a continuous stream of IRQ5 interrupt requests in the event that the monitored clock fails completely. Rather, the interrupt service routine (CLK__ERR) is left with the choice of whether or not to re-enable the monitoring. Also shown is the T_1 interrupt vector (IRQ__5).

```
Z8ASM      2.0
LOC    OBJ CODE    STMT SOURCE STATEMENT

                     1 TIMER4  MODULE
                     2         $SECTION PROGRAM
                     3 GLOBAL
                     4 !IRQ5 interrupt vector!
                     5         $ABS    10
P 0000 0015'         6 IRQ_5   ARRAY   [1 WORD]  :=    [CLK_ERR]
                     7
                     8         $REL
P 000C               9 MONITOR_INIT    PROCEDURE
                    10 ENTRY
P 0000 E6  F3  04   11         LD      PRE1,#%(2)00000100
                    12                         !bit 2-7: prescaler = 1;
                    13                          bit 1: external clock;
                    14                          bit 0: single-pass mode!
P 0003 E6  F7  00   15         LD      P3M,#00   !bit 5: let P36 be Tout!
P 0006 E6  F2  03   16         LD      T1,#3            !T1 load register,
                    17                                  = 1.5 * 2 usec   !
```

7. Timer/ Counter Functions (Continued)

```
P 0009 8F              18          DI
P 000A 56  FB  3B      19          AND     IMR,#%(2)00111011  !disable IRQ2 (Tin)!
P 000D 46  FB  20      20          OR      IMR,#%(2)00100000  !enable IRQ5 (T1)!
P 0010 9F              21          EI
                       22
P 0011 E6  F1  38      23          LD      TMR,#%(2)00111000
                       24                              !bit 4-5: Tin mode is
                       25                                        retrig. input;
                       26                               bit 3: enable T1 !
P 0014 AF              27          RET
P 0015                 28 END      MONITOR_INIT
                       29
                       30
P 0015                 31 CLK_ERR  PROCEDURE
                       32 ENTRY
                       33          !...!           !handle the missed clock!
                       34
                       35 !if clock monitoring should continue...!
P 0015 46  F1  08      36          OR      TMR,#%(2)00001000
                       37                              !bit 3: enable T1 !
P 0018 BF              38          IRET
P 0019                 39 END      CLK_ERR
                       40 END      TIMER4

   0 ERRORS
ASSEMBLY COMPLETE
```

MONITOR_INIT:
9 instructions
21 bytes
21.5 µs

CLK_ERR:
2 + instructions
4 + bytes
18.5 + µs including interrupt response time

SECTION 8

I/O Functions

The Z8 provides 32 I/O lines mapped into registers 0-3 of the internal register file. Each nibble of port 0 is individually programmable as input, output, or address/data lines (A_{15}-A_{12}, A_{11}-A_8). Port 1 is programmable as a single entity to provide input, output, or address/data lines (AD_7-AD_0). The operating modes for the bits of Ports 0 and 1 are selected by control register P01M (%F8). Selection of I/O lines as address/data lines supports access to external program and data memory; this is discussed in Section 3. Each bit of Port 2 is individually programmable as an input or an output bit. Port 2 bits programmed as outputs may also be programmed (via bit 0 of P3M) to all have active pull-ups or all be open-drain (active pull-ups inhibited). In Port 3, four bits ($P3_0$-$P3_3$) are fixed as inputs, and four bits ($P3_4$-$P3_7$) are fixed as outputs, but their functions are programmable. Special functions provided by Port 3 bits are listed in Table 4. Use of the Data Memory select output is discussed in Section 3; uses of T_{IN} and T_{OUT} are discussed in Section 7.

Function	Bit	Signal
Handshake	$P3_1$	$\overline{DAV}2$/RDY2
	$P3_2$	$\overline{DAV}0$/RDY0
	$P3_3$	$\overline{DAV}1$/RDY1
	$P3_4$	RDY1/$\overline{DAV}1$
	$P3_5$	RDY0/$\overline{DAV}0$
	$P3_6$	RDY2/$\overline{DAV}2$
Interrupt Request	$P3_0$	IRQ3
	$P3_1$	IRQ2
	$P3_2$	IRQ0
	$P3_3$	IRQ1
Counter/ Timer	$P3_1$	T_{IN}
	$P3_6$	T_{OUT}
Data Memory Select	$P3_4$	$\overline{DM}$
Status Out		
Serial I/O	$P3_0$	Serial In
	$P3_7$	Serial Out

Table 4. Port 3 Special Functions

8.1 Asynchronous Receiver/Transmitter Operation. Full-duplex, serial asynchronous receiver/transmitter operation is provided by the Z8 via $P3_7$ (output) and $P3_0$ (input) in conjunction with control register SIO (%F0), which is actually two registers: receiver buffer and transmitter buffer. Counter/Timer T_0 provides the clock for control of the bit rate.

The Z8 always receives and transmits eight bits between start and stop bits. However, if parity is enabled, the eighth bit (D_7) is replaced by the odd-parity bit when transmitted and a parity-error flag (= 1 if error) when received. Table 5 illustrates the state of the parity bit/parity error flag during serial I/O with parity enabled.

Although the Z8 directly supports either odd parity or no parity for serial I/O operation, even parity may also be provided with additional software support. To receive and transmit with even parity, the Z8 should be configured for serial I/O with odd parity disabled. The Z8 software must calculate parity

Character Loaded Into SIO	Transmitted To Serial Line	Received From Serial Line	Character Transferred To SIO	Note*
11000011	01000011	01000011	01000011	no error
11000011	01000011	01000111	11000111	error
01111000	11111000	11111000	01111000	no error
01111000	11111000	01111000	11111000	error

Table 5. Serial I/O With Odd Parity

* Left-most bit is D7

and modify the eighth bit prior to the load of a character into SIO and then modify a parity error flag following the load of a character from SIO. All other processing required for serial I/O (e.g. buffer management, error handling, etc.) is the same as that for odd parity operations.

To configure the Z8 for Serial I/O, it is necessary to:

- Enable $P3_0$ and $P3_7$ for serial I/O and select parity,
- Set up T_0 for the desired bit rate,
- Configure IRQ3 and IRQ4 for polled or automatic interrupt mode,
- Load and enable T_0.

To enable $P3_0$ and $P3_7$ for serial I/O, bit 6 of P3M (R247) is set. To enable odd parity, bit 7 of P3M is set; to disable it, the bit is reset. For example, the instruction

```
LD      P3M,#%40
```

will enable serial I/O, but disable parity. The instruction

```
LD      P3M,#%C0
```

will enable serial I/O, and enable odd parity.

In the following discussions, bit rate refers to all transmitted bits, including start, stop, and parity (if enabled). The serial bit rate is given by the equation:

$$\text{bit rate} = \frac{\text{input clock frequency}}{(2 \times 4 \times T_0 \text{ prescaler} \times T_0 \text{ counter} \times 16)}$$

The final divide-by-16 is incurred for serial communications, since in this mode T_0 runs at 16 times the bit rate in order to synchronize the data stream. To configure the Z8 for a specific bit rate, appropriate values must first be selected for T_0 prescaler and T_0 counter by the above equation; these values are then programmed into registers T_0 (%F4) and PRE0 (%F5) respectively. Note that PRE0 also controls the continuous vs. single-pass mode for T_0; continuous mode should be selected for serial I/O. For example, given an input clock frequency of 7.3728 MHz and a selected bit rate of 9600 bits per second, the equation is satisfied by T_0 counter = 2 and prescaler = 3. The following code sequence will configure the T_0 counter and T_0 prescaler registers:

```
LD   T0,#2    !T0 counter = 2!
LD   PRE0,#%(2)00001101
                !bit 2-7: prescaler = 3; bit 0:
                 continuous mode!
```

Interrupt request 3 (IRQ3) is generated whenever a character is transferred into the receive buffer; interrupt request 4 (IRQ4) is generated whenever a character is transferred out of the transmit buffer. Before accepting such interrupt requests, the Interrupt Mask, Request, and Priority Registers (IMR, IRQ, and IPR) must be programmed to configure the mode of interrupt response. The section on Interrupt Processing provides a discussion of interrupt configurations.

To load and enable T_0, set bits 0 and 1 of the timer mode register (TMR) via an instruction such as

```
OR   TMR,#%03
```

This will cause the T_0 prescaler and counter registers (PRE0 and T_0) to be transferred to the T_0 prescaler and counter. In addition, T_0 is enabled to count, and serial I/O operations will commence.

Characters to be output to the serial line should be written to serial I/O register SIO (%F0). IRQ4 will be generated when all bits have been transferred out.

Characters input from the serial line may be read from SIO. IRQ3 will be generated when a full character has been transferred into SIO.

The following module illustrates the receipt of a character and its immediate echo back to the serial line. It is assumed that the Z8 has been configured for serial I/O as described above, with IRQ3 (receive) enabled to interrupt, and IRQ4 (transmit) configured to be polled. The received character is stored in a circular buffer in register memory from address %42 to %5F. Register %41 contains the address of the next available buffer position and should have been initialized by some earlier routine to #%42.

8. I/O Functions (Continued)

```
Z8ASM     2.0
LOC    OBJ CODE    STMT SOURCE STATEMENT

                     1 SERIAL_IO       MODULE
                     2 CONSTANT
                     3  next_addr      :=      %41
                     4  start          :=      %42
                     5  length         :=      %1E
                     6 $SECTION PROGRAM
                     7 GLOBAL
                     8 !IRQ3 vector!
                     9         $ABS    6
P 0006 0000'        10 IRQ_3    ARRAY [1 WORD] :=  [GET_CHARACTER]
                    11
                    12         $REL    0
P 0000              13 GET_CHARACTER    PROCEDURE       ENTRY
                    14
                    15 !Serial I/O receive interrupt service!
                    16 !Echo received character and wait for
                    17  echo completion!
P 0000 E4  F0  F0   18         ld      SIO,SIO         !echo!
                    19
                    20 !save it in circular buffer!
P 0003 F5  F0  41   21         ld      @next_addr,SIO  !save in buffer!
P 0006 20  41       22         inc     next_addr       !point to next position!
P 0008 A6  41  60   23         cp      next_addr,#start+length
                    24                                 !wrap-around yet?!
P 000B EB  03       25         jr      ne,echo_wait    !no.!
P 000D E6  41  42   26         ld      next_addr,#start !yes. point to start!
                    27 !now, wait for echo complete!
                    28 echo_wait:
P 0010 66  FA  10   29         tcm     IRQ,#%10        !transmitted yet?!
P 0013 EB  FB       30         jr      nz,echo_wait    !not yet!
                    31
P 0015 56  FA  EF   32         and     IRQ,#%EF        !clear IRQ4!
P 0018 BF           33         IRET                    !return from interrupt!
P 0019              34 END     GET_CHARACTER
                    35 END     SERIAL_IO

   0 ERRORS
ASSEMBLY COMPLETE
```

10 instructions
25 bytes
35.5 μs + 5.5 μs for each additional pass through the echo_wait loop, including interrupt response time

8.2 Automatic Bit Rate Detection. In a typical system, where serial communication is required (e.g. system with a terminal), the desired bit rate is either user-selectable via a switch bank or nonvariable and "hard-coded" in the software. As an alternate method of bit-rate detection, it is possible to automatically determine the bit rate of serial data received by measuring the length of a start bit. The advantage of this method is that it places no requirements on the hardware design for this function and provides a convenient (automatic) operator interface.

In the technique described here, the serial channel of the Z8 is initialized to expect a bit rate of 19,200 bits per second. The number of bits (n) received through Port pin P30 for each bit transmitted is expressed by

$$n = 19{,}200/b$$

where b = transmission bit rate. For example, if the transmission bit rate were 1200 bits per second, each incoming bit would appear to the receiving serial line as 19,200/1200 or 16 bits.

The following example is capable of distinguishing between the bit rates shown in Table 6 and assumes an input clock frequency of 7.3728 MHz, a T_0 prescaler of 3, and serial I/O enabled with parity disabled. This example requires that a character with its low order bit = 1 (such as a carriage return) be sent to the serial channel. The start bit of this character can be measured by counting the number of zero bits collected before the low order 1 bit. The number of zero bits actually collected into data bits by the serial channel is less than n (as given in the above equation), due to the detection of start and stop bits. Figure 4 illustrates the collection (at 19,200

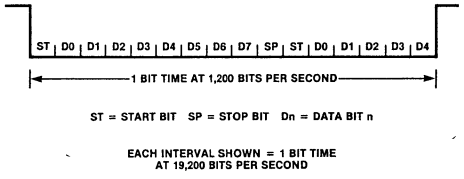


Figure 4. Collection of a Start Bit Transmitted at at 19,200 BPS

Bit Rate	Number of Bits Received Per Bit Transmitted	Number of 0 Bits Collected as Data Bits		T_0 Counter	
		dec	binary	dec	binary
19200	1	0	00000000	1	00000001
9600	2	1	00000001	2	00000010
4800	4	3	00000011	4	00000100
2400	8	7	00000111	8	00001000
1200	16	13	00001101	16	00010000
600	32	25	00011001	32	00100000
300	64	49	00110001	64	01000000
150	128	97	01100001	128	10000000

Table 6. Inputs to the Automatic Bit Rate Detection Algorithm

bits per second) of a zero bit transmitted to the Z8 at 1,200 bits per second. Notice that only 13 of the 16 zero bits received are collected as data bits.

Once the number of zero bits in the start bit has been collected and counted, it remains to translate this count into the appropriate T_0 counter value and program that value into T_0 (%F4). The patterns shown in the two binary columns of Table 6 are utilized in the algorithm for this translation.

As a final step, if incoming data is to commence immediately, it is advisable to wait until the remainder of the current "elongated" character has been received, thus "flushing" the serial line. This can be accomplished either via a software loop, or by programming T_1 to generate an interrupt request after the appropriate amount of time has elapsed. Since a character is composed of eight bits plus a minimum of one stop bit following the start bit, the length of time to delay may be expressed as

$$(9 \times n)/b$$

where n and b are as defined above. The following module illustrates a sample program for automatic bit rate detection.

```
Z8ASM     2.0
LOC     OBJ CODE    STMT SOURCE STATEMENT

                      1 bit_rate        MODULE
                      2 EXTERNAL
                      3 DELAY   PROCEDURE
                      4 GLOBAL
P 0000                5 main    PROCEDURE
                      6         ENTRY
P 0000 8F             7         di                  !disable interrupts!
P 0001 56  FB  77     8         and     IMR,#%77    !IRQ3 polled mode!
P 0004 56  FA  F7     9         and     IRQ,#%F7    !clear IRQ3!
P 0007 E6  F7  40    10         ld      P3M,#%40    !enable serial I/O!
P 000A E6  F4  01    11         ld      T0,#1
P 000D E6  F5  0D    12         ld      PRE0,#(3 SHL 2)+1  !bit rate = 19,200;
                     13                                    continuous count mode!
P 0010 B0  E0        14         clr     R0          !init. zero byte counter!
P 0012 E6  F1  03    15         ld      TMR,#3      !load and enable T0!
                     16
                     17 !collect input bytes by counting the number of null
                     18  characters received.  Stop when non-zero byte received!
                     19 collect:
P 0015 76  FA  08    20         TM      IRQ,#%08    !character received?!
P 0018 6B  FB        21         jr      z,collect   !not yet!
P 001A 18  F0        22         ld      R1,SIO      !get the character!
P 001C 56  FA  F7    23         and     IRQ,#%F7    !clear interrupt request!
P 001F 1E            24         inc     R1          !compare to 0 ...!
P 0020 1A  05        25         djnz    R1,bitloop  !...(in 3 bytes of code)!
P 0022 06  E0  08    26         add     R0,#8       !update count of 0 bits!
P 0025 8B  EE        27         jr      collect
                     28 bitloop:                    !add in zero bits from low
                     29                              end of 1st non-zero byte!
P 0027 E0  E1        30         RR      R1
P 0029 7B  03        31         jr      c,count_done
P 002B 0E            32         inc     R0
P 002C 8B  F9        33         jr      bitloop
                     34
                     35 !R0 has number of zero bits collected!
                     36 !translate R0 to the appropriate T0 counter value!
                     37 count_done:                 !R0 has count of zero bits!
P 002E 1C  07        38         ld      R1,#7
P 0030 2C  80        39         ld      R2,#%80     !R2 will have T0 counter value!
P 0032 90  E0        40         RL      R0
                     41
P 0034 90  E0        42 loop:   RL      R0
```

```
P 0036 7B  04          43          jr      c,done
P 0038 E0  E2          44          RR      R2
P 003A 1A  F8          45          djnz    r1,loop
                       46
P 003C 29  F4          47 done:    ld      T0,R2       !load value for detected
                       48                                  bit rate!
                       49 !Delay long enough to clear serial line of bit stream!
P 003E D6  0000*       50          call    DELAY
                       51 !clear receive interrupt request!
P 0041 56  FA  F7      52          and     IRQ,#%F7
                       53
P 0044                 54 END     main
                       55 END     bit_rate

   0 ERRORS
ASSEMBLY COMPLETE
```

30 instructions
68 bytes
Execution time is variable based on transmission bit rate.

8.3 Port Handshake. Each of Ports 0, 1 and 2 may be programmed to function under input or output handshake control. Table 7 defines the port bits used for the handshaking and the mode bit settings required to select handshaking. To input data under handshake control, the Z8 should read the input port when the $\overline{DAV}$ input goes Low (signifying that data is available from the attached device). To output data under handshake control, the Z8 should write the output port when the RDY input goes Low (signifying that the previously output data has been accepted by the attached device). Interrupt requests IRQ0, IRQ1, and IRQ2 are generated by the falling edge of the handshake signal input to the Z8 for Port 0, Port 1, and Port 2 respectively. Port handshake operations may therefore be processed under interrupt control.

Consider a system that requires communication of eight parallel bits of data under handshake control from the Z8 to a peripheral device and that Port 2 is selected as the output port. The following assembly code illustrates the proper sequence for initializing Port 2 for output handshake.

```
CLR   P2M    !Port 2 mode register: all Port
               2 bits are outputs!
OR    %03,#%40
               !set DAV2: data not available!
LD    P3M,#%20
               !Port 3 mode register: enable
               Port 2 handshake!
LD    %02,DATA
               !output first data byte; DAV2
               will be cleared by the Z8 to
               indicate data available to
               the peripheral device!
```

Note that following the initialization of the output sequence, the software outputs the first data byte without regard to the state of the RDY2 input; the Z8 will automatically hold $\overline{DAV}2$ High until the RDY2 input is High. The peripheral device should force the Z8 RDY2 input line Low after it has latched the data in response to a Low on $\overline{DAV}2$. The Low on RDY2 will cause the Z8 to automatically force $\overline{DAV}2$ High until the next byte is output. Subsequent bytes should be output in response to interrupt request IRQ2 (caused by the High-to-Low transition on RDY2) in either a polled or an enabled interrupt mode.

	Port 0	Port 1	Port 2
Input handshake lines	$P3_2 = \overline{DAV}$ $P3_5 = RDY$	$P3_3 = \overline{DAV}$ $P3_4 = RDY$	$P3_1 = \overline{DAV}$ $P3_6 = RDY$
Output handshake lines	$P3_2 = RDY$ $P3_5 = \overline{DAV}$	$P3_3 = RDY$ $P3_4 = \overline{DAV}$	$P3_1 = RDY$ $P3_6 = \overline{DAV}$
To select input handshake:	set bit 6 & reset bit 7 of P01M (program high nibble as input)	set bit 3 & reset bit 4 of P01M (program byte as input)	set bit 7 of P2M (program high bit as input)
To select output handshake:	reset bits 6, 7 of P01M (program high nibble as output)	reset bits 3, 4 of P01M (program byte as output)	reset bit 7 of P2M (program high bit as output)
To enable handshake:	set bit 5 of Port 3 ($P3_5$); set bit 2 of P3M	set bit 4 of Port 3 ($P3_4$); set bits 3, 4 of P3M	set bit 6 of Port 3 ($P3_6$); set bit 5 of P3M

Table 7. Port Handshake Selection

Arithmetic Routines

This section gives examples of the arithmetic and rotate instructions for use in multiplication, division, conversion, and BCD arithmetic algorithms.

9.1 Binary to Hex ASCII. The following module illustrates the use of the ADD and SWAP arithmetic instructions in the conversion of a 16-bit binary number to its hexadecimal ASCII representation. The 16-bit number is viewed as a string of four nibbles and is processed one nibble at a time from left to right, beginning with the high-order nibble of the lower memory address. %30 is added to each nibble if it is in the range 0 to 9; otherwise %37 is added. In this way, %0 is converted to %30, %1 to %31, . . . %A to %41, . . . %F to %46. Figure 5 illustrates the conversion of RR0 (contents = %F2BE) to its hex ASCII equivalent; the destination buffer is pointed to by RR4.

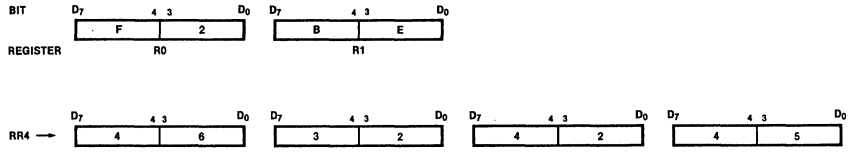


Figure 5. Conversion of (RR0) to Hex ASCII

```
Z8ASM      2.99     INTERNAL RELEASE
LOC     OBJ CODE     STMT SOURCE STATEMENT

                       1 ARITH   MODULE
                       2 GLOBAL
P 0000                 3 BINASC  PROCEDURE
                       4 !*************************************************
                       5  Purpose =       To convert a 16-bit binary
                       6                  number to Hex ASCII
                       7
                       8  Input =         RR0 = 16-bit binary number.
                       9                  RR4 = pointer to destination
                      10                        buffer in external memory.
                      11
                      12  Output =        Resulting ASCII string (4 bytes)
                      13                  in destination buffer.
                      14                  RR4 incremented by 4 .
                      15                  R0,R2,R6 destroyed.
                      16 *************************************************!
                      17 ENTRY
                      18
P 0000 6C  04         19          ld      R6,#%04 !nibble count!
P 0002 F0  E0         20 again:   SWAP    R0      !look at next nibble!
P 0004 28  E0         21          ld      R2,R0
P 0006 56  E2  0F     22          and     R2,#%0F !isolate 4 bits!
                      23 !convert to ASCII : R2 + #%30 if R0 in range 0 to 9
                      24                   else  R2 + #%37 (in range 0A to 0F)
                      25 !
P 0009 06  E2  30     26          ADD     R2,#%30
P 000C A6  E2  3A     27          cp      R2,#%3A
P 000F 7B  03         28          jr      ult,skip
P 0011 06  E2  07     29          ADD     R2,#%07
P 0014 92  24         30 skip:    lde     @RR4,R2          !save ASCII in buffer!
P 0016 A0  E4         31          incw    RR4              !point to next
                      32                                    buffer position!
P 0018 A6  E6  03     33          cp      R6,#%03 !time for second byte?!
P 001B EB  02         34          jr      ne,same_byte     !no.!
P 001D 08  E1         35          ld      R0,R1            !2nd byte!
                      36 same_byte:
P 001F 6A  E1         37          djnz    R6,again
P 0021 AF             38          ret
P 0022                39 END      BINASC
                      40 END      ARITH

   0 errors
Assembly complete


15 instructions
34 bytes
120.5 µs (average)
```

9.2 BCD Addition. The following module illustrates the use of the add with carry (ADC) and decimal adjust (DA) instructions for the addition of two unsigned BCD strings of equal length. Within a BCD string, each nibble represents a decimal digit (0-9). Two such digits are packed per byte with the most significant digit in bits 7-4. Bytes within a BCD string are arranged in memory with the most significant digits stored in the lowest memory location. Figure 6 illustrates the representation of 5970 in a 6-digit BCD string, starting in register %33.

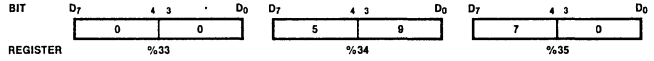


Figure 6. Unsigned BCD Representation

```
Z8ASM     2.0
LOC    OBJ CODE    STMT SOURCE STATEMENT

                     1 ARITH    MODULE
                     2 CONSTANT
                     3  BCD_SRC := R1
                     4  BCD_DST := R0
                     5  BCD_LEN := R2
                     6 GLOBAL
P 0000               7 BCDADD   PROCEDURE
                     8 !***************************************************
                     9  Purpose =       To add two packed BCD strings of
                    10                   equal length.
                    11                   dst  <-- dst + src
                    12
                    13  Input =         R0 = pointer to dst BCD string.
                    14                   R1 = pointer to src BCD string.
                    15                   R2 = byte count in BCD string
                    16                        (digit count = (R2)*2 ).
                    17
                    18  Output =        BCD string pointed to by R0 is
                    19                   the sum.
                    20                   Carry FLAG = 1 if overflow.
                    21                   R0 , R1 as on entry.
                    22                   R2 = 0
                    23 ***************************************************!
                    24 ENTRY
                    25
P 0000 02  12       26          add      BCD_SRC,BCD_LEN !start at least... !
P 0002 02  02       27          add      BCD_DST,BCD_LEN !significant digits!
P 0004 CF           28          rcf                      !carry = 0!
                    29 add_again:
P 0005 00  E1       30          dec      BCD_SRC         !point to next two
                    31                                    src digits!
P 0007 00  E0       32          dec      BCD_DST         !point to next two
                    33                                    dst digits!
P 0009 E3  31       34          ld       R3,@BCD_SRC     !get src digits!
P 000B 13  30       35          ADC      R3,@BCD_DST     !add dst digits!
P 000D 40  E3       36          DA       R3              !decimal adjust!
P 000F F3  03       37          ld       @BCD_DST,R3     !move to dst!
P 0011 2A  F2       38          djnz     BCD_LEN,add_again  !loop for next
                    39                                       digits!
P 0013 AF           40          ret                      !all done!
                    41
P 0014              42 END      BCDADD
                    43 END      ARITH

   0 ERRORS
ASSEMBLY COMPLETE
```

11 instructions
20 bytes
Execution time is a function of the number of bytes (n) in input BCD string:
20 μs + 12.5 (n - 1) μs

9.3 Multiply. The following module illustrates an efficient algorithm for the multiplication of two unsigned 8-bit values, resulting in a 16-bit product. The algorithm repetitively shifts the multiplicand right (using RRC), with the low-order bit being shifted out (into the carry flag). If a one is shifted out, the multiplier is added to the high-order byte of the partial product. As the high-order bits of the multiplicand are vacated by the shift, the resulting partial-product bits are rotated in. Thus, the multiplicand and the low byte of the product occupy the same byte, which saves register space, code, and execution time.

```
Z8ASM      2.99     INTERNAL RELEASE
LOC     OBJ CODE     STMT SOURCE STATEMENT

                       1 ARITH   MODULE
                       2 CONSTANT
                       3  MULTIPLIER     :=      R1
                       4  PRODUCT_LO     :=      R3
                       5  PRODUCT_HI     :=      R2
                       6  COUNT          :=      R0
                       7 GLOBAL
P 0000                 8 MULT    PROCEDURE
                       9 !***************************************************
                      10  Purpose =      To perform an 8-bit by 8-bit unsigned
                      11                 binary multiplication.
                      12
                      13  Input =        R1 = multiplier
                      14                 R3 = multiplicand
                      15
                      16  Output =       RR2 = product
                      17                 R0   destroyed
                      18 ***************************************************!
                      19 ENTRY
P 0000 0C  09         20         ld      COUNT,#9         !8 BITS + 1!
P 0002 B0  E2         21         clr     PRODUCT_HI       !INIT HIGH RESULT BYTE!
P 0004 CF             22         RCF                      !CARRY = 0!
P 0005 C0  E2         23 LOOP:   RRC     PRODUCT_HI
P 0007 C0  E3         24         RRC     PRODUCT_LO
P 0009 FB  02         25         jr      NC,NEXT
P 000B 02  21         26         ADD     PRODUCT_HI,MULTIPLIER
P 000D 0A  F6         27 NEXT:   djnz    COUNT,LOOP
P 000F AF             28         ret
P 0010                29 END     MULT
                      30 END     ARITH

   0 errors
Assembly complete

9 instructions
16 bytes
92.5 µs (average)
```

9.4 Divide. The following module illustrates an efficient algorithm for the division of a 16-bit unsigned value by an 8-bit unsigned value, resulting in an 8-bit unsigned quotient. The algorithm repetitively shifts the dividend left (via RLC). If the high-order bit shifted out is a one or if the resulting high-order dividend byte is greater than or equal to the divisor, the divisor is subtracted from the high byte of the dividend. As the low-order bits of the dividend are vacated by the shift left, the resulting partial-quotient bits are rotated in. Thus, the quotient and the low byte of the dividend occupy the same byte, which saves register space, code, and execution time.

9. Arithmetic Routines (Continued)

```
Z8ASM     2.0
LOC    OBJ CODE   STMT SOURCE STATEMENT

                    1 ARITH   MODULE
                    2 CONSTANT
                    3  COUNT          :=      R0
                    4  DIVISOR        :=      R1
                    5  DIVIDEND_HI    :=      R2
                    6  DIVIDEND_LO    :=      R3
                    7 GLOBAL
P 0000              8 DIVIDE  PROCEDURE
                    9 !***************************************************
                   10  Purpose =      To perform a 16-bit by 8-bit unsigned
                   11                 binary division.
                   12
                   13  Input =        R1 = 8-bit divisor
                   14                 RR2 = 16-bit dividend
                   15
                   16  Output =       R3  = 8-bit quotient
                   17                 R2  = 8-bit remainder
                   18                 Carry flag = 1 if overflow
                   19                            = 0 if no overflow
                   20 ***************************************************!
                   21 ENTRY
P 0000 0C  08      22         ld      COUNT,#8        !LOOP COUNTER!
                   23
                   24 !CHECK IF RESULT WILL FIT IN 8 BITS!
P 0002 A2  12      25         cp      DIVISOR,DIVIDEND_HI
P 0004 BB  02      26         jr      UGT,LOOP        !CARRY = 0 (FOR RLC)!
                   27 !WON'T FIT.  OVERFLOW!
P 0006 DF          28         SCF                     !CARRY = 1!
P 0007 AF          29         ret
                   30
                   31 LOOP:    !RESULT WILL FIT.  GO AHEAD WITH DIVISION!
P 0008 10  E3      32         RLC     DIVIDEND_LO     !DIVIDEND * 2!
P 000A 10  E2      33         RLC     DIVIDEND_HI
P 000C 7B  04      34         jr      c,subt
P 000E A2  12      35         cp      DIVISOR,DIVIDEND_HI
P 0010 BB  03      36         jr      UGT,next        !CARRY = 0!
P 0012 22  21      37 subt:   SUB     DIVIDEND_HI,DIVISOR
P 0014 DF          38         SCF                  !TO BE SHIFTED INTO RESULT!
P 0015 0A  F1      39 next:   djnz    COUNT,LOOP      !no flags affected!
                   40
                   41 !ALL    DONE!
P 0017 10  E3      42         RLC     DIVIDEND_LO
                   43                                 !CARRY = 0: no overflow!
P 0019 AF          44         ret
P 001A             45 END DIVIDE
                   46 END ARITH

   0 ERRORS
ASSEMBLY COMPLETE
```

15 instructions
26 bytes
124.5 μs (average)

SECTION 10 Conclusion

This Application Note has focused on ways in which the Z8 microcomputer can easily yet effectively solve various application problems. In particular, the many sample routines illustrated in this document should aid the reader in using the Z8 to greater advantage. The major features of the Z8 have been described so that the user can continue to expand and explore the Z8's repertoire of uses.

MEMORY SPACE AND REGISTER ORGANIZATION

Memory Space

The Z8 can address up to 126K bytes of program and data memory separately from the on chip registers. The 16-bit program counter provides for 64K bytes of program memory, the first 2K bytes of which are internal to the Z8. The remaining 62K bytes of program memory are located externally and can be implemented with ROM, EPROM, or RAM.

The 62K bytes of data memory are also located external to the Z8 and begin with location 2048. The two address spaces, program memory and data memory, are individually selected by the Data Memory Select output ($\overline{DM}$) which is available from Port 3.

The Program Memory Map and the Data Memory Map are shown in Figure 2.

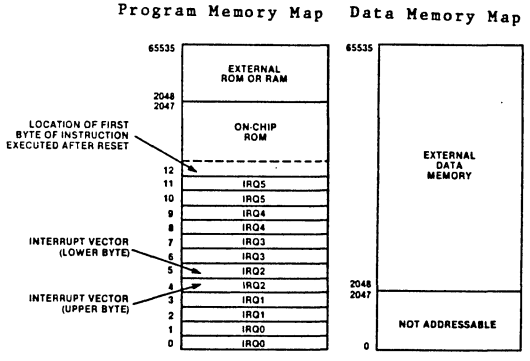


Figure 2 Program Memory Map And Data Memory Map

External memory access is accomplished by the Z8 through its I/O Ports. When less than 256 bytes of external memory are required, Port 1 is programmed for the multiplexed address/data mode (AD0-AD7). In this configuration 8-bits of address and 8-bits of data are time multiplexed on the 8 I/O lines for memory transfers. The memory "handshake" control lines are provided by the Address Strobe ($\overline{AS}$), Data Strobe ($\overline{DS}$), and the Read/Write (R/$\overline{W}$) pins on the Z8. If program and data are included in the external memory space, the Data Memory Select ($\overline{DM}$) function may be programmed into the Port 3 Mode register. When this is done, the $\overline{DM}$ signal is available on line 4 of the Port 3 (P34) to select between program and data memory for external memory operations.

Port 0 is used to provide the additional address bits for external memory beyond the first 256 locations up to a full 16-bits of external memory address. It becomes immediately obvious that the first 8-bits of external memory address from Port 1 must be latched externally to the Z8 so that program or data may be transferred over the same 8 lines during the external memory transaction machine cycle. The $\overline{AS}$, $\overline{DS}$, and R/$\overline{W}$ control lines simplify the required interface logic. The timing for external memory transactions is given in Figure 3.

Registers

The Z8 has 144 8-bit registers including four Port registers (R0-R3), 124 general purpose registers (R4-R127), and 16 control and status register (R240-R255). The 144 registers are all located in the same 8-bit address space to allow any Z8 instruction to operate on them. The 124 general purpose registers can function as accumulators, address pointers, or index registers. The registers are read when they are referenced as source registers, and written when they are referenced as destination registers. Registers may be addressed directly with an 8-bit address, or indirectly through another register with an 8-bit address, or with a 4-bit address and Register Pointer.

The entire Z8 register space may be divided into 16 contiguous Working Register Areas, each having 16 registers. A control register, called the Register Pointer, may be loaded with the most significant nibble of a Working Register Area address. The Register Pointer provides for the selection of the Working Register Area, and allows registers within that area to be selected with a 4-bit address.

The Z8 register organization is shown in Figure 4.

Stacks

The Z8 provides for stack operations through the use of a stack pointer, and the stack may be located in the internal register space or in the external data memory space. The "stack selection" bit (D2) in the Port 0-1 Mode control register selects an internal or external stack. When the stack is located internally, register 255 contains an 8-bit stack pointer and register 254 is available as a general purpose register. If an external stack is used, register 255 or registers 254 and 255 may be used as the stack pointer depending on the anticipated "depth" of the stack. When registers 254 and 255 are both used, the stack pointer is a full 16-bits wide. The CALL, IRET, RET, PUSH, and

POP instructions are Z8 instructions which include implicit stack operations.

I/O STRUCTURE

Parallel I/O

The Z8 microcomputer has 32 lines of I/O arranged as four 8-bit ports. All of the I/O ports are TTL compatible and are configurable as input, output, input/output, or address/data. The handshake control lines for Ports 0, 1, and 2 are bits from Port 3 that have been programmed through a Mode control register, except for $\overline{AS}$, $\overline{DS}$, and R/$\overline{W}$ which are available as separate Z8 pins. The I/O ports are accessed as separate internal registers by the Z8. Ports 0 and 1 share one Mode control register, and Ports 2 and 3 each have a Mode control register for configuring the port.

Port 0 can be programmed to be an I/O port or as an address output port. More specifically Port 0 can be configured to be an 8-bit I/O port, or a 4-bit address output port (A8-A11) for external memory and one 4-bit I/O port, or an 8-bit address output port (A8-A15) for external memory.

Port 1 can be programmed as an I/O port (with or without handshake), or an address/data port (AD0-AD7) for interfacing with external memory. If Port 1 is programmed to be an address/data port, it cannot be accessed as a register.

Port 2 can be configured as individual input or output bits, and Port 3 can be programmed to be parallel I/O bits, and/or serial I/O bits, and/or handshake control lines for the other ports. Figure 5 shows the port Mode registers.

The off chip expansion capability using Ports 0 and 1 offers the added feature of being Z-Bus compatible. All Z-Bus compatible peripheral chips that are available now, and will be available in the future, will interface directly with the Z8 multiplexed address/data bus.

Serial I/O

As mentioned in the last section, Port 3 can be programmed to be a serial I/O port with bits 0 and 7, the serial input and serial output lines respectively. The serial I/O capability provides for full duplex asynchronous serial data at rates up to 62.5K bits per second. The transmitted format is one start bit, eight data bits including odd parity (if parity is enabled), and two stop bits. The received data format is one start bit, eight data bits and at least one stop bit. If parity is enabled, the eighth data bit received (bit 7) is replaced by a parity error flag which indicates a parity error if it is set to a ONE.

Timer/Counter T_0 is the baud rate generator and runs at 16 times the serial data bit rate. The receiver is double duffered and an internal interrupt (IRQ3) is generated when a character is loaded into the receive buffer register. A different internal interrupt (IRQ4) is generated when a character is transmitted.

COUNTER/TIMERS

The Z8 has two 8-bit programmable counter/timers, each of which is driven by a programmable 6-bit prescaler. The T_1 prescaler can be driven by internal or external clock sources, and the T_0 prescaler is driven by the internal clock only. The two prescalers and the two counters are loaded through four control registers (see Figure 4) and when a counter/timer reaches the "end of count" a timer interrupt is generated (IRQ4 for T_0, and IRQ5 for T_1). The counter/timers can be programmed to stop upon reaching the end of count, or to reload and continue counting. Since either counter (one at a time) can have its output available external to the Z8, and Counter/Timer T_1 can have an external input, the two counters can be cascaded.

Port 3 can be programmed to provide timer outputs for external time base generation or trigger pulses.

INTERRUPT STRUCTURE

The Z8 provides for six interrupts from eight different sources including four Port 3 lines (P30-P33), serial in, serial out, and two counter/timers. These interrupts can be masked and prioritized using the Interrupt Mask Register (register 251) and the Interrupt Priority Register (register 249). All interrupts can be disabled with the master interrupt enable bit in the Interrupt Mask Register.

Each of the six interrupts has a 16-bit interrupt vector that points to its interrupt service routine. These six 2-byte vectors are placed in the first twelve locations in the program memory space (see Figure 2).

When simultaneous interrupts occur for enabled interrupt sources, the Interrupt Priority Register determines which interrupt is serviced first. The priority is programmable in a way that is described by Figure 6.

When an interrupt is recognized by the Z8, all other interrupts are disabled, the program counter and program control flags are saved, and the program counter is loaded with the corresponding interrupt vector. Interrupts must be re-enabled by the user upon entering the service

SUPER8
APPLICATION NOTES
AND TECHNICAL ARTICLES

GETTING STARTED WITH THE ZILOG SUPER8

by Charles M. Link, II

Any time an engineer switches to a new processor, he usually begins the time consuming process of learning the quirks of the new part. This article is the first of a series of articles written to speed that transition time from any other processor to the Zilog Super8.

Getting started is the most difficult part of switching to a strange new processor and development tools. Weeks can be spent just getting the first lines of initialization code written and successfully assembled. Testing the code becomes another problem. The software from this article series has been tested and it should be possible to copy most of the software directly to a user's application. All of the software is available in machine readable form as noted at the end of the article.

This first article demonstrates the proper initialization of the Zilog Super8 microcontroller. It sets up a Z8800 ROMLESS for 64K bytes of external program memory, although most typical applications probably do not require more than maybe 4K or 8K bytes. Ports 2 and 3, which are bit mappable as inputs or outputs, are set into the output mode. Port 4, also bit mappable, is set into the input mode. A hardware schematic has been included as an example.

The hardware schematic shown defines a simple Super8 implementation that was used to test the code in this series of articles. This example defines a simple evaluation board that contains 32K bytes of programable EPROM, and up to 32K bytes of RAM. The design contains a simple RS-232 interface that is used in future articles of the series. The entire board, including the RS-232 interface, is powered from 5 volts. The RAM battery option allows the software to be downloaded into the RAM and saved if power fails. Additional logic on the design allows a user to protect the lower half of RAM with a simple jumper change. This prevents the processor from destroying executable code if it goes off into space on a power failure.

Specifically, the ROMLESS Super8 is used as the core. The Super8 requires a latch to demultiplex the address from the data bus. A 74LS373 fits nicely here, requiring only an inverter to correct for the address strobe. The 'LS373 with inverter is preferred here rather than a single 'LS374 because the 'LS373 is a transparent latch and will present the address earlier than the 'LS374. JU1 selects the EPROM size, correcting for the /PGM pin on 2764 and 27128 EPROMs. It is necessary to use pull down resistors on the upper 4 bits of the address bus because on reset, the ROMLESS Super8 defines only 12 bits for address; the other 4 are set as inputs. Since LS-TTL devices require more current to pull down the inputs, this pull down trick will only work for MOS and CMOS inputs, hence the requirement for the logic chips in this design to be HCT type devices.

The remaining logic is required to select the EPROM or RAM. JU2 selects the half-RAM protect mode. JU3 is set to determine what size ram to protect. This circuit allows the lower half of CMOS battery backed RAM to be read only, and removes chip select on any writes to that address space. Of course, that exact circuitry and the battery is optional, and might be replaced by a power threshold detector. On the other front, a Maxim MAX 232 provides the RS-232 interface requiring only 5 volts.

To make the software initialization more interesting, a few other typical initialization tasks are demonstrated. The entire block of registers (user ram) is cleared to zero, and one of the counter timer units is initialized to provide a periodic interrupt to form the heart of a real time clock function.

The program shows the typical pseudo-op usage demonstrated. This article series uses a cross assembler available from Zilog for either an IBM PC or a VAX operating under VMS. The program begins by defining the registers used as general purpose storage. This is done so the user does not have to refer to register numbers, but may refer to a name equated to the register.

The first 32 bytes of every program (beginning at 0000H) always contain the interrupt vectors for the different sources. Using the Zilog assembler, the .WORD pseudo-op defines a pair of bytes for each of the 16 sources. Program execution begins at location 0020H. Since copyright requirements usually require the notice as close to the beginning as possible, it becomes necessary to jump around an ASCII string. The .ASCII pseudo-op generates the necessary string for this notice.

The source code describes almost completely, without further explaination, the entire initialization. Once initialized, the processor loops in a WAIT loop waiting on the periodic interrupt generated by the counter/timer. The counter timer interrupts 60 times per second, and the interrupt bumps ram storage locations representing seconds, minutes, and hours. Each time a location is bumped, an external port line is toggled so that those without emulators can see some activity with an oscilloscope.

One point of notice, is the interrupt service routine for the timer. One must reset the end of count interrupt bit (the source of interrupt) before exiting the interrupt service routine.

In the next article of this series, we will take the same basic initialization routine and modify it to support the serial UART. That article will demonstrate polled serial communications using the Zilog Super 8.

[Editors note: The sofware for this series is available on an IBM PC diskette and is included with the Super 8 Emulator package available from Creative Technology Corporation, 5144 Peachtree Road, Suite 301, Atlanta, GA 30341. (404) 455-8255. Any Zilog Field Application engineer should also be able to provide copies of the software on a user provided diskette.]

```
;
        .TITLE  Sample Zilog Super 8 Initialization
;
;=============================================================
;=      TITLE:          INIT.S8                              =
;=      DATE:           JUNE 17, 1986                        =
;=      PURPOSE:        TO DEMONSTRATE INITIALIZATION        =
;=                      OF THE ZILOG SUPER 8 USING THE       =
;=                      ZILOG ASMS8 ASSEMBLER                =
;=      PROGRAMMER:     CHARLES M. LINK, II                  =
;=============================================================
;
;
;
        .PAGE   55      ;set maximum page size to 55 lines
;
;*************************************************************
;*                                                           *
;*                  REGISTER EQUATE TABLE                    *
;*                                                           *
;*************************************************************
;
period: .equ    0       ;period timer
second: .equ    1       ;seconds timer
minute: .equ    2       ;minutes timer
hours:  .equ    3       ;hours timer
;
;*************************************************************
;*                                                           *
;*                  INTERRUPT VECTOR TABLE                   *
;*                                                           *
;*************************************************************
;
INTR0:  .WORD   INTRET          ;this area should always be defined
INTR1:  .WORD   INTRET          ;as it reserves the lower 32 bytes
INTR2:  .WORD   INTRET          ;for the interrupt table.  the name
INTR3:  .WORD   INTRET          ;of the subroutine for each particular
INTR4:  .WORD   INTRET          ;interrupt service would normally be
INTR5:  .WORD   INTRET          ;named here.
INTR6:  .WORD   TIMER0
INTR7:  .WORD   INTRET
INTR8:  .WORD   INTRET
INTR9:  .WORD   INTRET
INTR10: .WORD   INTRET
INTR11: .WORD   INTRET
INTR12: .WORD   INTRET
INTR13: .WORD   INTRET
INTR14: .WORD   INTRET
INTR15: .WORD   INTRET
;
;*************************************************************
;*                                                           *
;*              START OF PROGRAM EXECUTION                   *
;*                                                           *
;*************************************************************
;
```

```
START:  jr      START1          ;program execution unconditionally
                                ;begins at this location after reset
                                ;and power up.
        .ASCII  'REL 0 6/16/86' ;jump around optional ascii string
                                ;containing release info, copyright, etc.
START1: di                      ;begin
        sb0                     ;select register bank 0
        ld      EMT,#00000000B  ;external memory timing=no wait input, normal
                                ;memory timing, no wait states, stack internal,
                                ;and DMA internal
        ld      P0,#00H         ;address begins at 0000h, set upper byte
        ld      P0M,#11111111B  ;select all lines as address
        ld      PM,#00110000B   ;enable port 0 as upper 8 bits address
        ld      H1C,#00000000B  ;handshake not enabled port 0
;
;port 1 is defined in romless part as address/data.  it is not necessary
;here to initialize that port
;
        ld      P2,#00H         ;port 2 outputs low
        ld      P3,#00H         ;port 3 outputs low
        ld      P2AM,#10101010B ;p30,31,20,21 as output
        ld      P2BM,#10101010B ;p32,33,22,23 as output
        ld      P2CM,#10101010B ;p34,35,24,25 as output
        ld      P2DM,#10101010B ;p36,37,26,27 as output
;
        ld      P4,#00000000B   ;clear port 4 register
        ld      P4D,#11111111B  ;set all bits of P4 as inputs
        ld      P4OD,#00000000B ;active push/pull [not necessary since all
                                ; bits are inputs
;
;basic Super 8 I/O is initialized, now internal registers
;
        ld      RP0,#0C0H       ;set working register low to lower 8 bytes
        ld      RP1,#0C8H       ;set working register high to upper 8 bytes
        ld      SPL,#0FFH       ;set stack pointer to start at top of set two
                                ;note here that only lower 8 bits are used
                                ;for stack pointer.  location 0FFH is wasted
                                ;as stack operation.  SPH is general purpose
                                ;storage.
;
;now clear the internal memory and stack area
;
        ld      SPH,#0FFH       ;point to top of general purpose register
ZERO:   clr     @SPH            ;zero it
        dec     SPH
        jr      nz,ZERO         ;do it until register set is all cleared
        clr     @SPH            ;zero last register
;
;now everything except working registers is cleared
;
;cpu and memory now initialized, set up timer for real time clock
;
        ld      SYM,#00000000B  ;disable fast interrupt response
        ld      IPR,#00000010B  ;interrupt priority
                                ;IRQ2>IRQ3>IRQ4>IRQ5>IRQ6>IRQ7>IRQ0>IRQ1
        ld      IMR,#00000100B  ;enable only interrupt 2
        sb1                     ;select bank 1
        ld      C0TCH,#^HB(50000)       ;high byte of time constant
        ld      C0TCL,#^LB(50000)       ;low byte of time constant
                                ;12,000,000 hertz / 4 / 50,000 = 60 hertz
                                ;12 Mhz is xtal freq, 4 is internal divider
        ld      C0M,#00000100B  ;p27,37 is I/O, programmed up/down, no capture
                                ;timer mode is selected
        sb0                     ;select bank 0
        ld      C0CT,#10100101B ;continuous, count down, load counter,
                                ;zero count interrupt enable, enable counter
;
;timer is initialized, now lets enable interrupts and wait
;
        ei                      ;enable interrupts
WAIT:   nop
        nop
        nop
        nop
        jr      WAIT            ;loop back
;
```

```
;
        nop
        nop
        nop
TIMER0: inc     period          ;bump periodic counter (60 hertz)
        cp      period,#60      ;one second yet?
        jr      ne,NOROLL       ;no rollover
        xor     P2,#00000001B   ;complement the second bit
        clr     period          ;start it over again
        inc     second          ;bump the seconds timer
        cp      second,#60      ;reached maximum
        jr      ne,NOROLL       ;no rollover
        xor     P2,#00000010B   ;complement the minute bit
        clr     second          ;start it over again
        inc     minute          ;bump the minutes timer
        cp      minute,#60      ;reached maximum
        jr      ne,NOROLL       ;no rollover
        xor     P2,#00000100B   ;complement the hour bit
        clr     minute          ;start it over again
        inc     hours           ;bump the hours timer
        cp      hours,#24       ;reached maximum
        jr      ne,NOROLL       ;no rollover
        clr     hours           ;start it over again
NOROLL: or      C0CT,#00000010B ;reset end of count interrupt
        nop
        nop
INTRET: iret                    ;and return from interrupt
;
;
;
        .END
```

ZiLOG

Application Note

POLLED ASYNCHRONOUS SERIAL OPERATION WITH THE ZILOG SUPER8

by Charles M. Link, II

The transition from one processor to another often involves many hours of trial-and-error software development to determine the quirks (manufacturers call it features) of the part. Once the real features are discovered, programming the processor to perform as described can be hazardous to one's health. This article, the second in a series of eight, attempts to introduce the Zilog Super8 user to the serial communications port, and its initialization in a polled serial environment.

The universal asynchronous receiver/transmitter (UART) on the Super8 is a fairly unique implementation among single chip microcomputers in that it supports all of the functions generally available only on chip level UARTs. The UART is a close approximation of the Z80 DART device in one channel. It supports independent receiver/transmitter clocking, 5 to 8 bits per character, plus optional odd or even parity, and even an optional wake-up bit. The UART can serve full duplex communications via polled, interrupt, or DMA modes of operation. Auto-echo and internal loopback can be programmed as options. The most unique of the UART features is the character match and interrupt option.

The following article describes the initialization and use of the UART in a polled environment. This software has been tested and provides several routines that may be copied into a user's software. Although the demonstration software does not do much, it is fully functional as a stand-alone program, and may be "burned" into eprom as a test.

The basic software is almost the same general purpose initialization software from the first article in the series. Routines set-up counter/timer 0 for a real time clock option. Note, however, the change to configuration register P2AM. It is necessary to configure port 30 as input for receive data and p31 as output for transmit data.

The UART initialization sequence begins by setting the functions in the UART MODE A register. Since the UMA register is in the alternate bank, the instruction SB1 must be executed to gain access to the following registers. The loaded data selects a X16 clock, 8 bits per character, no parity, and no wake up values. Note that the clock options are X1, X16, X32, and X64. For true asynchronous operation, a clock multiplier option of at least X16 is required. The X1 mode could be used for externally syncing the received data to the UART. The transmitter is not affected.

Next, the baud rate generator must be loaded. The formula for determining the baud rate is shown below:

TIME CONSTANT = (XTAL FREQ / 8 / CLOCK MULT / DESIRED RATE) - 1

where TIME CONSTANT is a 16 bit value, XTAL FREQ is the crystal ifrequency in hertz, CLOCK MULT is the clock rate loaded into UART MODE A register (as above X1, X16, X32, and X64), and DESIRED rate is the desired bit rate in bits per second. Note that the baud rate generator may be used as an additional counter, and may be loaded with any value permitting just about any crystal frequency to operate the Super8.

The cross-assembler permitted a single 16-bit decimal number to be loaded into the UART BAUD RATE GENERATOR, high and low byte, without unnecessary figuring using the high/low byte pseudo-op.

The initialization sequence continues, with the UART MODE B register next. This example sends port 21 data to the port 21 pin. An option allows different clocks to be sent out from this pin. It could be used for clocking external logic, or for diagnostic purposes to make sure the baud rate generator is running. Auto-echo is not selected in this application, as that is primarily what the example software does. The receive and transmit clock input is the baud rate generator and the generator source is the internal clock; the crystal divided by four. Since the baud rate generator has been loaded, it is enabled, and the UART is set for normal operation (without loopback). Loopback operation permits transmitting and receiving data without any external logic in front of the Super8.

The UART TRANSMIT CONTROL register is initialized next in the sequence. Select transmit data out on port 31 and transmit enable. The stop bits are optional, and the DMA and WAKE-UP enables are for features discussed in future application articles. At this point, the transmitter is operational, and except for housekeeping, is usable. The housekeeping is in reference to selecting the bank 0 by executing the SB0 instruction.

Since polled mode communications are desired, all of the UART interrupts are disabled by loading the UART INTERRUPT ENABLE with all zeros. Lastly, the receiver must be enabled by setting bit 0 of the UART RECEIVE CONTROL register.

This program primarily sends a message to the console and then accepts input from the console and echos it upon receiving a carriage return. It is necessary to delay sending data to the console after initialization because the transmit data line is in the SPACE state when idle. Alternately, add a pull-up resistor to the output, and while idle and before initialized, it would exibit the MARK state.

The transmit character routine "SENDC" monitors the TRANSMIT BUFFER EMPTY bit of the UART TRANSMIT CONTROL register. When this bit is a "1", the transmit buffer is empty and may be loaded with a new character for transmission. To transmit a character, load the character into the UART data register (UIO).

The receive character routine "GETC" monitors the RECEIVE CHARACTER AVAILABLE bit of the UART RECEIVE CONTROL register. When this bit is a "1", a new character has been received by the UART.

The polled mode of UART operation is simple. Making the UART operate in an interrupt mode requires a few minor modifications, and DMA mode requires a few more modifications. Those modes are the subject of future application articles in this series.

```
;
        .TITLE  Sample Zilog Super 8 Serial Port Initialization
;
;
;=============================================================
;=      TITLE:          UART1.S                              =
;=      DATE:           JULY 17, 1986                        =
;=      PURPOSE:        TO DEMONSTRATE INITIALIZATION        =
;=                      AND USAGE OF SERIAL PORT IN          =
;=                      POLLED MODE.                         =
;=      ASSEMBLER:      ZILOG ASMS8 ASSEMBLER                =
;=      PROGRAMMER:     CHARLES M. LINK, II                  =
;=============================================================
;
;
;
        .PAGE   55      ;set maximum page size to 55 lines
;*************************************************************
;*                                                           *
;*                    GENERAL EQUATES                        *
;*                                                           *
;*************************************************************
;
CR:     .equ    0dH     ;carriage return
LF:     .equ    0aH     ;line feed
;
;
;*************************************************************
;*                                                           *
;*                  REGISTER EQUATE TABLE                    *
;*                                                           *
;*************************************************************
;
period: .equ    0       ;period timer
second: .equ    1       ;seconds timer
minute: .equ    2       ;minutes timer
hours:  .equ    3       ;hours timer
;working register equates
MPTR:   .equ    RR8     ;message pointer for external memory
;
;*************************************************************
;*                                                           *
;*                  INTERRUPT VECTOR TABLE                   *
;*                                                           *
;*************************************************************
;
INTR0:  .WORD   INTRET          ;this area should always be defined
INTR1:  .WORD   INTRET          ;as it reserves the lower 32 bytes
INTR2:  .WORD   INTRET          ;for the interrupt table.  the name
INTR3:  .WORD   INTRET          ;of the subroutine for each particular
INTR4:  .WORD   INTRET          ;interrupt service would normally be
INTR5:  .WORD   INTRET          ;named here.
INTR6:  .WORD   TIMER0
INTR7:  .WORD   INTRET
INTR8:  .WORD   INTRET
INTR9:  .WORD   INTRET
INTR10: .WORD   INTRET
INTR11: .WORD   INTRET
INTR12: .WORD   INTRET
```

```
INTR13: .WORD   INTRET
INTR14: .WORD   INTRET
INTR15: .WORD   INTRET
;
;*************************************************************
;*                                                           *
;*              START OF PROGRAM EXECUTION                   *
;*                                                           *
;*************************************************************
;
START:  jr      START1          ;program execution unconditionally
                                ;begins at this location after reset
                                ;and power up.
        .ASCII  'REL 0 7/17/86' ;jump around optional ascii string
                                ;containing release info, copyright, etc.
START1: di                      ;begin
        sb0                     ;select register bank 0
        ld      EMT,#00000000B  ;external memory timing=no wait input, normal
                                ;memory timing, no wait states, stack internal,
                                ;and DMA internal
        ld      P0,#00H         ;address begins at 0000h, set upper byte
        ld      P0M,#11111111B  ;select all lines as address
        ld      PM,#00110000B   ;enable port 0 as upper 8 bits address
        ld      H1C,#00000000B  ;handshake not enabled port 0
;
;port 1 is defined in romless part as address/data.  it is not necessary
;here to initialize that port
;
        ld      P2,#00H         ;port 2 outputs low
        ld      P3,#00H         ;port 3 outputs low
        ld      P2AM,#10001010B ;p31,20,21 as output,p30 input
                                ;it is necessary here to configure p30 as input
                                ;for the receive data, and p31 as output for
                                ;transmit data for UART
        ld      P2BM,#10101010B ;p32,33,22,23 as output
        ld      P2CM,#10101010B ;p34,35,24,25 as output
        ld      P2DM,#10101010B ;p36,37,26,27 as output
;
        ld      P4,#00000000B   ;clear port 4 register
        ld      P4D,#11111111B  ;set all bits of P4 as inputs
        ld      P4OD,#00000000B ;active push/pull [not necessary since all
                                ; bits are inputs
;
;basic Super 8 I/O is initialized, now internal registers
;
        ld      RP0,#0C0H       ;set working register low to lower 8 bytes
        ld      RP1,#0C8H       ;set working register high to upper 8 bytes
        ld      SPL,#0FFH       ;set stack pointer to start at top of set two
                                ;note here that only lower 8 bits are used
                                ;for stack pointer.  location 0FFH is wasted
                                ;as stack operation.  SPH is general purpose
                                ;storage.
;
;now clear the internal memory and stack area
;
        ld      SPH,#0FFH       ;point to top of general purpose register
ZERO:   clr     @SPH            ;zero it
        dec     SPH
        jr      nz,ZERO         ;do it until register set is all cleared
        clr     @SPH            ;zero last register
;
;now everything except working registers is cleared
;
;cpu and memory now initialized, set up timer for real time clock
;
        ld      SYM,#00000000B  ;disable fast interrupt response
        ld      IPR,#00000010B  ;interrupt priority
                                ;IRQ2>IRQ3>IRQ4>IRQ5>IRQ6>IRQ7>IRQ0>IRQ1
        ld      IMR,#00000100B  ;enable only interrupt 2
        sb1                     ;select bank 1
        ld      C0TCH,#^HB(50000)       ;high byte of time constant
        ld      C0TCL,#^LB(50000)       ;low byte of time constant
                                ;12,000,000 hertz / 4 / 50,000 = 60 hertz
                                ;12 Mhz is xtal freq, 4 is internal divider
        ld      C0M,#00000100B  ;p27,37 is I/O, programmed up/down, no capture
                                ;timer mode is selected
```

```
        sb0                     ;select bank 0
        ld      COCT,#10100101B ;continuous, count down, load counter,
                                ;zero count interrupt enable, enable counter
;
;timer is set, now lets initialize the UART for polled operation
;
        sb1                     ;bank 1
        ld      UMA,#01110000B
                                ;time constant  = (12,000,000/4/16/9600/2)-1=
                                ;8.76 rounded to 9.
                                ;note that a 12 Mhz does not make a very
                                ;accurate baud rate source.  error is large
        ld      UBGH,#^HB(00009)        ;high byte of time constant
        ld      UBGL,#^LB(00009)        ;low byte of time constant
        ld      UMB,#00011110B  ;p21=p21data,auto-echo is off, transmit and
                                ;receive clock is baud rate generator output,
                                ;baud rate generator input is system clock / 2,
                                ;baud rate generator is enabled, loopback
                                ;is disabled
        sb0                     ;select bank 0
        ld      UTC,#10001000B  ;select p31 as transmit data out, 1 stop bit
                                ;and transmit enable
        ld      UIE,#00000000B  ;disable all interrupts, no DMA
        ld      URC,#00000010B  ;enable receive

;UART is initialized, enable interrupts for real time clock
;
        ei                      ;enable interrupts
;
;wait 1 full second for serial line to mark before sending anything
;
WAIT:   cp      second,#1       ;wait 1 second
        jr      ne,WAIT
;
;display the logon message
;
LOGON:  ldw     MPTR,#MSG       ;load the address of MSG into word reg MPTR
        call    SENDM           ;send the message
;
;logon message displayed, get response from console
;and move to upper register memory
;
GET:    ld      r1,#80          ;maximum character count
        ld      r2,#80H         ;point to first location in upper register bank
GETN:   call    GETC            ;get input from console
        and     r0,#7fH         ;remove upper parity bit
        call    SENDC           ;echo to console
        ld      @r2,r0          ;move to upper internal ram in Super8
        cp      r0,#CR          ;was the received character a carriage return
        jr      eq,ECHO         ;if so, echo it to console
        inc     r2              ;bump pointer
        djnz    r1,GETN         ;get next character if not done
;
;if carriage return typed, or 80 characters exceeded, echo message
;
ECHO:   ldw     MPTR,#MSG1      ;load the address of MSG1 in word reg MPTR
        call    SENDM           ;send the message
        ld      r1,#80          ;maximum character count
        ld      r2,#80H         ;first location of character buffer
ECHO1:  ld      r0,@r2          ;get character from buffer
        call    SENDC           ;send the character to console
        cp      r0,#CR          ;carriage return?
        jr      eq,LOGON        ;if so, end message display
        inc     r2              ;bump pointer
        djnz    r1,ECHO1        ;display next character if not done
        jr      LOGON
;
;subroutines
;
;send message at MPTR until '$' character found
SENDM:  ldci    r0,@MPTR        ;get the character
        call    SENDC           ;otherwise send character
        cp      r0,#'$'         ;last character?
        jr      ne,SENDM        ;and loop back to send next one
        ret
```

```
;send character in r0
SENDC:  tm      UTC,#00000010B  ;transmit buffer empty yet
        jr      z,SENDC         ;if not, wait until it is
        ld      UIO,r0          ;load the character into the transmitter
        ret
;get a character from the uart, return in r0
GETC:   tm      URC,#00000001B  ;character available
        jr      z,GETC          ;if not, wait until it is
        ld      r0,UIO          ;get the character from the receiver
        ret
;
;real time interrupt running in background
;
TIMER0: inc     period          ;bump periodic counter (60 hertz)
        cp      period,#60      ;one second yet?
        jr      ne,NOROLL       ;no rollover
        xor     P2,#00000001B   ;complement the second bit
        clr     period          ;start it over again
        inc     second          ;bump the seconds timer
        cp      second,#60      ;reached maximum
        jr      ne,NOROLL       ;no rollover
        xor     P2,#00000010B   ;complement the minute bit
        clr     second          ;start it over again
        inc     minute          ;bump the minutes timer
        cp      minute,#60      ;reached maximum
        jr      ne,NOROLL       ;no rollover
        xor     P2,#00000100B   ;complement the hour bit
        clr     minute          ;start it over again
        inc     hours           ;bump the hours timer
        cp      hours,#24       ;reached maximum
        jr      ne,NOROLL       ;no rollover
        clr     hours           ;start it over again
NOROLL: or      COCT,#00000010B ;reset end of count
        nop
        nop
INTRET: iret                    ;and return from interrupt
;
;
;
MSG:    .ASCII  CR,LF,'Super8 Uart test program.',CR,LF
        .ASCII  'Enter up to one full line followed by return',CR,LF,'$'
MSG1:   .ASCII  CR,LF,'Echoed back, your line was...',CR,LF,'$'


        .END
```

USING THE ZILOG SUPER8 IN INTERRUPT DRIVEN COMMUNICATIONS

by Charles M. Link, II

The power of the Super8 microcomputer lies in its on board peripherals. One of those peripherals is the full duplex UART. The UART can operate under program control in polled mode, or under interrupt control, and in a DMA mode. This article, the third in a series, discusses using the UART in a fully interrupt driven system. Since it is assumed that the reader has access to the eariler article discussing the UART and the polled mode of operation, this article will only discuss the differences.

The Zilog Super8 contains an on board interrupt controller that is tightly linked to the other on-board peripherals. The UART, being on-board, can be operated in an interrupt mode permitting very little execution overhead time while monitoring the UART for incomming characters and waiting for the UART to send outgoing characters.

Operation of an interrupt driven system demands more software logic to control the interrupt. Although more software is present, less time is spent executing it, because most of the overhead is in the setup for interrupt transfers. Generally, interrupt driven serial I/O overlaps some other process or processes, and therefore enhances total system speed and operation. Interrupt driven I/O has no advantages in a system that must wait on the serial port. In the example program, no real advantage has been gained by interrupt operation. The program displays a simple message to the console, and accepts input responses and echos them. For program simplicity, the main program waits on the interrupt to complete before starting the next phase of the program.

In any interrupt driven system, the central processor must know what to do when an interrupt occurs. The Super8 is no exeception. An interrupt vector table directs the processor to begin execution at certain addresses for particular interrupt inputs. The UART can be the source for up to five different interrupts and therefore up to five of the sixteen vectors can be designated for it. This sample program ignores errors and special condition interrupts, and therefore only two vectors are used; one for transmit buffer empty and one for receive character available. These vectors are programmed into the vector table by setting interrupt vector 10 (zero reference) to the address for the receive data service routine, and setting interrupt vector 13 to the address for the transmit data service routine.

The setup of the Super8 is essentially the same as that of the serial port in a polled mode of operation. The

proper priority for the interrupts are assigned arbitrarily. The real time clock as highest priority, the receive character available as second priority, and transmit character buffer empty as the lowest priority. Generally, the transmit interrupt should be the lowest in an asynchronous system because if it does not get serviced immediately, no major problems occur. If the real time interrupt took more time in relationship to the time required to transmit a single character, then maybe the receive should be put higher. If the receiver is not serviced, that character would be lost.

Enabling the interrupts is a two stage process. First the mask in the INTERRUPT MASK REGISTER must be enabled for each level of the interrupts used. Next, it is necessary to enable the individual transmit and receive interrupts. In the example program, a character is loaded into the transmit buffer and then the interrupt is enabled by setting bit 2 in the UART INTERRUPT ENABLE (UIE) register. Each successive transmit interrupt indicates an empty buffer, and the next character is loaded into the buffer. When the last character is loaded into the buffer, the transmit interrupt is disabled to prevent further interruptions by clearing bit 2 of the UIE register.

The receiver interrupt is enabled to allow the processor to accept incoming characters by setting bit 0 of the UIE register. Once set, any received character will cause the processor to transfer control to the "RXDATI" routine. In this example, the receive service routine reads, echos, and stores each received character until a carriage routine is received. The input is then repeated.

The example program does not fully utilize the interrupt system, as it waits for each routine to complete before moving to the next. However, it does however work, and demonstrates interrupt service routines. Serial interrupt software is not complex, and could lead to very powerful user programs. With the addition of the on board DMA to automaticlly transfer characters, the Super8 can complete many tasks that previously would require complex hardware and software. The next article in the series demonstrates using the DMA controller with the serial port.

```
;
        .TITLE  Sample Zilog Super 8 Serial Interrupt Mode Operation
;
;
;=============================================================
;=      TITLE:          UART2.S                              =
;=      DATE:           JULY 17, 1986                        =
;=      PURPOSE:        TO DEMONSTRATE INTERRUPT             =
;=                      DRIVEN SERIAL PORT                   =
;=                      COMMUNICATIONS                       =
;=      ASSEMBLER:      ZILOG ASMS8 ASSEMBLER                =
;=      PROGRAMMER:     CHARLES M. LINK, II                  =
;=============================================================
;
;
;
        .PAGE   55      ;set maximum page size to 55 lines
;*************************************************************
;*                                                           *
;*                    GENERAL EQUATES                        *
;*                                                           *
;*************************************************************
;
CR:     .equ    0dH     ;carriage return
LF:     .equ    0aH     ;line feed
;
;
;*************************************************************
;*                                                           *
;*                  REGISTER EQUATE TABLE                    *
;*                                                           *
;*************************************************************
;
period: .equ    0       ;period timer
second: .equ    1       ;seconds timer
minute: .equ    2       ;minutes timer
hours:  .equ    3       ;hours timer
;working register equates
MPTR:   .equ    RR8     ;message pointer for external memory
;
;*************************************************************
;*                                                           *
;*                  INTERRUPT VECTOR TABLE                   *
;*                                                           *
;*************************************************************
;
INTR0:  .WORD   INTRET          ;this area should always be defined
INTR1:  .WORD   INTRET          ;as it reserves the lower 32 bytes
INTR2:  .WORD   INTRET          ;for the interrupt table.  the name
INTR3:  .WORD   INTRET          ;of the subroutine for each particular
INTR4:  .WORD   INTRET          ;interrupt service would normally be
INTR5:  .WORD   INTRET          ;named here.
INTR6:  .WORD   TIMER0
INTR7:  .WORD   INTRET
INTR8:  .WORD   INTRET
INTR9:  .WORD   INTRET
INTR10: .WORD   RXDATI
INTR11: .WORD   INTRET
INTR12: .WORD   INTRET
INTR13: .WORD   TXDATI
INTR14: .WORD   INTRET
INTR15: .WORD   INTRET
;
;*************************************************************
;*                                                           *
;*              START OF PROGRAM EXECUTION                   *
;*                                                           *
;*************************************************************
;
START:  jr      START1          ;program execution unconditionally
                                ;begins at this location after reset
                                ;and power up.
        .ASCII  'REL 0 7/17/86' ;jump around optional ascii string
                                ;containing release info, copyright, etc.
START1: di                      ;begin
        sb0                     ;select register bank 0
```

```
        ld      EMT,#00000000B  ;external memory timing=no wait input, normal
                                ;memory timing, no wait states, stack internal,
                                ;and DMA internal
        ld      P0,#00H         ;address begins at 0000h, set upper byte
        ld      P0M,#11111111B  ;select all lines as address
        ld      PM,#00110000B   ;enable port 0 as upper 8 bits address
        ld      H1C,#00000000B  ;handshake not enabled port 0
;
;port 1 is defined in romless part as address/data.  it is not necessary
;here to initialize that port
;
        ld      P2,#00H         ;port 2 outputs low
        ld      P3,#00H         ;port 3 outputs low
        ld      P2AM,#10001010B ;p31,20,21 as output,p30 input
                                ;it is necessary here to configure p30 as input
                                ;for the receive data, and p31 as output for
                                ;transmit data for UART
        ld      P2BM,#10101010B ;p32,33,22,23 as output
        ld      P2CM,#10101010B ;p34,35,24,25 as output
        ld      P2DM,#10101010B ;p36,37,26,27 as output
;
        ld      P4,#00000000B   ;clear port 4 register
        ld      P4D,#11111111B  ;set all bits of P4 as inputs
        ld      P4OD,#00000000B ;active push/pull [not necessary since all
                                ; bits are inputs
;
;basic Super 8 I/O is initialized, now internal registers
;
        ld      RP0,#0C0H       ;set working register low to lower 8 bytes
        ld      RP1,#0C8H       ;set working register high to upper 8 bytes
        ld      SPL,#0FFH       ;set stack pointer to start at top of set two
                                ;note here that only lower 8 bits are used
                                ;for stack pointer.  location 0FFH is wasted
                                ;as stack operation.  SPH is general purpose
                                ;storage.
;
;now clear the internal memory and stack area
;
        ld      SPH,#0FFH       ;point to top of general purpose register
ZERO:   clr     @SPH            ;zero it
        dec     SPH
        jr      nz,ZERO         ;do it until register set is all cleared
        clr     @SPH            ;zero last register
;
;now everything except working registers is cleared
;
;cpu and memory now initialized, set up timer for real time clock
;
        ld      SYM,#00000000B  ;disable fast interrupt response
        ld      IPR,#00000010B  ;interrupt priority
                                ;IRQ2>IRQ3>IRQ4>IRQ5>IRQ6>IRQ7>IRQ0>IRQ1
        ld      IMR,#01000110B  ;enable counter, rx and tx interrupts
        sb1                     ;select bank 1
        ld      C0TCH,#^HB(50000)       ;high byte of time constant
        ld      C0TCL,#^LB(50000)       ;low byte of time constant
                                ;12,000,000 hertz / 4 / 50,000 = 60 hertz
                                ;12 Mhz is xtal freq, 4 is internal divider
        ld      C0M,#00000100B  ;p27,37 is I/O, programmed up/down, no capture
                                ;timer mode is selected
        sb0                     ;select bank 0
        ld      C0CT,#10100101B ;continuous, count down, load counter,
                                ;zero count interrupt enable, enable counter
;
;timer is set, now lets initialize the UART for polled operation
;
        sb1                     ;bank 1
        ld      UMA,#01110000B
                                ;time constant  = (12,000,000/4/16/9600/2)-1=
                                ;8.76 rounded to 9.
                                ;note that a 12 Mhz does not make a very
                                ;accurate baud rate source.  error is large
        ld      UBGH,#^HB(00009)        ;high byte of time constant
        ld      UBGL,#^LB(00009)        ;low byte of time constant
        ld      UMB,#00011110B  ;p21=p21data,auto-echo is off, transmit and
                                ;receive clock is baud rate generator output,
                                ;baud rate generator input is system clock / 2,
                                ;baud rate generator is enabled, loopback
                                ;is disabled
```

```
        sb0                     ;select bank 0
        ld      UTC,#10001000B  ;select p31 as transmit data out, 1 stop bit
                                ;and transmit enable
        ld      UIE,#00000000B  ;no interrupts, no DMA
        ld      URC,#00000010B  ;enable receive

;UART is initialized, enable interrupts for real time clock
;
        ei                      ;enable interrupts
;
;wait 1 full second of serial line mark before sending anything
;
WAIT:   cp      second,#1       ;wait 1 second
        jr      ne,WAIT
;
;display the logon message
;
LOGON:  ldw     MPTR,#MSG       ;load the address of MSG into word reg MPTR
        call    SENDM           ;send the message
        call    TXWAT           ;wait for transmitter to complete
;
;logon message displayed, get response from console
;and move to upper register memory
;
GET:    ld      r1,#80          ;maximum character count
        ld      r2,#80H         ;point to first location in upper register bank
        di                      ;stop interrupts
        or      UIE,#00000001B  ;receive character enable
        ei
;now wait for input to be completed
GW:     tm      UIE,#00000001B  ;wait for interrupt to be disabled
        jr      nz,GW           ;if interrupt still enabled

;
;if carriage return typed, or 80 characters exceeded, echo message
;
ECHO:   ldw     MPTR,#MSG1      ;load the address of MSG1 in word reg MPTR
        call    SENDM           ;send the message
;
;since messages are interrupt driven, we must wait for message to
;complete before transmitting next message
;
        call    TXWAT           ;wait on transmitter
        ld      r1,#80          ;maximum character count
        ld      r2,#80H         ;first location of character buffer
ECHO1:  ld      r0,@r2          ;get character from buffer
        call    SENDC           ;send the character to console
        cp      r0,#CR          ;carriage return?
        jr      eq,LOGON        ;if so, end message display
        inc     r2              ;bump pointer
        djnz    r1,ECHO1        ;display next character if not done
        jr      LOGON
;
;subroutines
;
;send message at MPTR until '$' character found
SENDM:  ldci    r0,@MPTR        ;get the character
        call    SENDC           ;start UART transmitting
        di                      ;no interrupts
        or      UIE,#00000100B  ;enable transmit interrupts
        ei
        ret
;send character in r0
SENDC:  tm      UTC,#00000010B  ;transmit buffer empty yet
        jr      z,SENDC         ;if not, wait until it is
        ld      UIO,r0          ;load the character into the transmitter
        ret
;transmit buffer available interrupt
TXDATI: ldci    r0,@MPTR        ;get next character to transmit
        ld      UIO,r0          ;load the character in transmitter
        cp      r0,#'$'         ;last character
        jr      eq,LASTT        ;if last transmit character
        iret
LASTT:  and     UIE,#11111011B  ;disable transmit interrupts
        iret                    ;ignore it if no character to transmit
;transmitter wait routine
TXWAT:  tm      UIE,#00000100B  ;wait until interrupts disabled
        jr      nz,TXWAT        ;wait if bit set
        ret
```

```
;receive character available interrupt
RXDATI: ld      r0,UIO          ;get input from console
        and     r0,#7fH         ;remove upper parity bit
        call    SENDC           ;echo to console
        ld      @r2,r0          ;move to upper internal ram in Super8
        cp      r0,#CR          ;was the received character a carriage return
        jr      eq,LASTR        ;if so, disable interrupts
        inc     r2              ;bump pointer
        djnz    r1,RXR          ;exit if not last
LASTR:  and     UIE,#11111110B  ;disable the receive interrupts
RXR:    iret
;
;real time interrupt running in background
;
TIMER0: inc     period          ;bump periodic counter (60 hertz)
        cp      period,#60      ;one second yet?
        jr      ne,NOROLL       ;no rollover
        xor     P2,#00000001B   ;complement the second bit
        clr     period          ;start it over again
        inc     second          ;bump the seconds timer
        cp      second,#60      ;reached maximum
        jr      ne,NOROLL       ;no rollover
        xor     P2,#00000010B   ;complement the minute bit
        clr     second          ;start it over again
        inc     minute          ;bump the minutes timer
        cp      minute,#60      ;reached maximum
        jr      ne,NOROLL       ;no rollover
        xor     P2,#00000100B   ;complement the hour bit
        clr     minute          ;start it over again
        inc     hours           ;bump the hours timer
        cp      hours,#24       ;reached maximum
        jr      ne,NOROLL       ;no rollover
        clr     hours           ;start it over again
NOROLL: or      COCT,#00000010B ;reset end of count
        nop
        nop
INTRET: iret                    ;and return from interrupt
;
;
;
MSG:    .ASCII  CR,LF,'Super8 Uart test program.',CR,LF
        .ASCII  'Enter up to one full line followed by return',CR,LF,'$'
MSG1:   .ASCII  CR,LF,'Echoed back, your line was...',CR,LF,'$'
.END
```

USING THE SUPER8 SERIAL PORT WITH DMA

by Charles M. Link, II

With the increasing integration available today, microprocessor manufacturers are incorporating new peripherals that typically were off board in previous products, and sometimes required a large amount of external logic to utilize. The direct memory access function is a good example. Zilog has incorporated a very powerful DMA in the new Super8 microcontroller. It has the capability of linking to several on board peripherals, including the serial port, and can control data transfers to the different memory mediums.

The Super8, with its on-board DMA can reduce processor overhead in data transfer tasks. It allows direct transfer of serial input characters to either internal register memory (256 bytes) or external ram memory. For example, this transfer can be set to transfer a specific number of input characters, then interrupt the processor. Processor program service overhead is minimal. Serial output characters can be transfered from external EPROM or ram memory, or the internal register memory.

The required setup for the DMA transfers are much the same as that of interrupt or polled operation. This program example uses the DMA to interrupt upon termination of data transfers so that approopriate vectors and routines are required. Since the program links to the serial port, the DMA uses the serial port receive and transmit interrupt vectors 10 and 13, respectively. Upon completion of a receive DMA transfer, the service routine defined by the receive vector is executed. Upon completion of the transmit DMA transfer, service routine defined by the transmit vector is executed.

It is necessary to define the memory source/destination by setting the appropriate state of bit 0 in the EXTERNAL MEMORY TIMING (EMT) register. Initially, the example program selects external memory as the source/destination. A special note: read the fine print in the technical manual. Many hours were spent debugging the DMA mode of operation, with the final realization that internal rom does not qualify as external memory. Only that memory that would be selected if the /DM line was true would be a valid source/destination. Since this article uses the hardware defined from the first of the series, and uses a Z8800 with external EPROM, it will work perfectly. ROM and PIGGYBACK or prototype type parts will not work. Neither will emulators.

This sample uses the DMA mode to transmit a few lines of ASCII data to a console. The DMA requires a total byte count to properly transfer the data and terminate. Be careful to recognize that the ASCIL pseudo-op in the Zilog assembler, or many other assemblers, is not an easy way to generate the byte count. Warning! The Zilog assembler generates a length for each subgroup, e.g., "MSG" generates a separate length for each group separated by commas, not one total length.

Initially, the DMA transfers from EPROM. The address from which to transfer is C0 and C1 as defined by the working register pointers. It is necessary to set RP0 to C0 to access the register, and it is accessed as R0 and R1 or RR0. The count for the transfer is taken from DMA COUNT HIGH and DMA COUNT LOW. For each transfer, initialize the address and count values. Upon completion of the DMA transmit process, when the count goes to -1, a transmit interrupt is generated. The example program disables transmit interrupts and DMA, and returns. The main line program was polling the interrupt enable bit for completion.

Next, the DMA is set up to transfer 25 characters into the internal register memory. One must select internal memory in the EMT register by clearing bit 0. The address for transfer requires only one byte, so that working register 1 (R1), when RP0 equals C0, is the address pointer. The DMA count must also be loaded, in this case with 25. For demonstration purposes, the auto-echo bit of the UART MODE B register is selected. This causes any characters received to be automatically looped back to the transmit port. Finally, the receive interrupt and DMA enable bits (BITS 0 and 1) are set to enable and begin DMA operation. When 25 characters have been input to the Super8, a receive interrupt will be generated, and control will be transfered to the "RXDATI" routine, where interrupts and DMA are disabled.

The last routine in the example software sends another message from EPROM to the console and then sends the characters from the internal memory buffer that were previously entered. The prime consideration is to remember to select the source/destination memory in the EMT register.

In this DMA example, the code is simple for DMA operation. It is important to note that this example does not

fully utilize the functionality of the DMA transfer. The example purposely waits in a software loop while the DMA transfer occurs. This prevents the supporting code from becoming too complex to follow for an example. Normal operation might have the UART receiving characters under DMA controls and transmitting characters under interrupt control with processing occurring somewhere in the middle.

```
;
        .TITLE  Sample Zilog Super 8 Serial DMA Mode Operation
;
;
;=============================================================
;=      TITLE:          UART3.S                              =
;=      DATE:           JULY 17, 1986                        =
;=      PURPOSE:        TO DEMONSTRATE DMA                   =
;=                      DRIVEN SERIAL PORT                   =
;=                      COMMUNICATIONS                       =
;=      ASSEMBLER:      ZILOG ASMS8 ASSEMBLER                =
;=      PROGRAMMER:     CHARLES M. LINK, II                  =
;=============================================================
;
;
;
        .PAGE   55      ;set maximum page size to 55 lines
;*************************************************************
;*                                                           *
;*                    GENERAL EQUATES                        *
;*                                                           *
;*************************************************************
;
CR:     .equ    0dH     ;carriage return
LF:     .equ    0aH     ;line feed
;
;
;*************************************************************
;*                                                           *
;*                  REGISTER EQUATE TABLE                    *
;*                                                           *
;*************************************************************
;
period: .equ    0       ;period timer
second: .equ    1       ;seconds timer
minute: .equ    2       ;minutes timer
hours:  .equ    3       ;hours timer
;working register equates
MPTR:   .equ    RR0     ;message pointer for external memory
;
;*************************************************************
;*                                                           *
;*                  INTERRUPT VECTOR TABLE                   *
;*                                                           *
;*************************************************************
;
INTR0:  .WORD   INTRET          ;this area should always be defined
INTR1:  .WORD   INTRET          ;as it reserves the lower 32 bytes
INTR2:  .WORD   INTRET          ;for the interrupt table.  the name
INTR3:  .WORD   INTRET          ;of the subroutine for each particular
INTR4:  .WORD   INTRET          ;interrupt service would normally be
INTR5:  .WORD   INTRET          ;named here.
INTR6:  .WORD   TIMER0
INTR7:  .WORD   INTRET
INTR8:  .WORD   INTRET
INTR9:  .WORD   INTRET
INTR10: .WORD   RXDATI
INTR11: .WORD   INTRET
INTR12: .WORD   INTRET
INTR13: .WORD   TXDATI
INTR14: .WORD   INTRET
INTR15: .WORD   INTRET
;
;*************************************************************
;*                                                           *
;*              START OF PROGRAM EXECUTION                   *
;*                                                           *
;*************************************************************
;
START:  jr      START1          ;program execution unconditionally
```

```
                                ;begins at this location after reset
                                ;and power up.
        .ASCII  'REL 0 7/17/86' ;jump around optional ascii string
                                ;containing release info, copyright, etc.
START1: di                      ;begin
        sb0                     ;select register bank 0
        ld      EMT,#00000001B  ;external memory timing=no wait input, normal
                                ;memory timing, no wait states, stack internal,
                                ;and DMA external
        ld      P0,#00H         ;address begins at 0000h, set upper byte
        ld      P0M,#11111111B  ;select all lines as address
        ld      PM,#00110000B   ;enable port 0 as upper 8 bits address
        ld      H1C,#00000000B  ;handshake not enabled port 0
;
;port 1 is defined in romless part as address/data.  it is not necessary
;here to initialize that port
;
        ld      P2,#00H         ;port 2 outputs low
        ld      P3,#00H         ;port 3 outputs low
        ld      P2AM,#10001010B ;p31,20,21 as output,p30 input
                                ;it is necessary here to configure p30 as input
                                ;for the receive data, and p31 as output for
                                ;transmit data for UART
        ld      P2BM,#10101010B ;p32,33,22,23 as output
        ld      P2CM,#10101010B ;p34,35,24,25 as output
        ld      P2DM,#10101010B ;p36,37,26,27 as output
;
        ld      P4,#00000000B   ;clear port 4 register
        ld      P4D,#11111111B  ;set all bits of P4 as inputs
        ld      P4OD,#00000000B ;active push/pull [not necessary since all
                                ; bits are inputs
;
;basic Super 8 I/O is initialized, now internal registers
;
        ld      RP0,#0C0H       ;set working register low to lower 8 bytes
        ld      RP1,#0C8H       ;set working register high to upper 8 bytes
        ld      SPL,#0FFH       ;set stack pointer to start at top of set two
                                ;note here that only lower 8 bits are used
                                ;for stack pointer.  location 0FFH is wasted
                                ;as stack operation.  SPH is general purpose
                                ;storage.
;
;now clear the internal memory and stack area
;
        ld      SPH,#0FFH       ;point to top of general purpose register
ZERO:   clr     @SPH            ;zero it
        dec     SPH
        jr      nz,ZERO         ;do it until register set is all cleared
        clr     @SPH            ;zero last register
;
;now everything except working registers is cleared
;
;cpu and memory now initialized, set up timer for real time clock
;
        ld      SYM,#00000000B  ;disable fast interrupt response
        ld      IPR,#00000010B  ;interrupt priority
                                ;IRQ2>IRQ3>IRQ4>IRQ5>IRQ6>IRQ7>IRQ0>IRQ1
        ld      IMR,#01000110B  ;enable counter, rx and tx interrupts
        sb1                     ;select bank 1
        ld      C0TCH,#^HB(50000)       ;high byte of time constant
        ld      C0TCL,#^LB(50000)       ;low byte of time constant
                                ;12,000,000 hertz / 4 / 50,000 = 60 hertz
                                ;12 Mhz is xtal freq, 4 is internal divider
        ld      C0M,#00000100B  ;p27,37 is I/O, programmed up/down, no capture
                                ;timer mode is selected
        sb0                     ;select bank 0
        ld      C0CT,#10100101B ;continuous, count down, load counter,
                                ;zero count interrupt enable, enable counter
;
;timer is set, now lets initialize the UART for polled operation
;
        sb1                     ;bank 1
        ld      UMA,#01110000B
                                ;time constant  = (12,000,000/4/16/9600/2)-1=
                                ;8.76 rounded to 9.
                                ;note that a 12 Mhz does not make a very
                                ;accurate baud rate source.  error is large
```

```
        ld      UBGH,#^HB(00009)        ;high byte of time constant
        ld      UBGL,#^LB(00009)        ;low byte of time constant
        ld      UMB,#00011110B  ;p21=p21data,auto-echo is off, transmit and
                                ;receive clock is baud rate generator output,
                                ;baud rate generator input is system clock / 2,
                                ;baud rate generator is enabled, loopback
                                ;is disabled
        sb0                     ;select bank 0
        ld      UTC,#10001000B  ;select p31 as transmit data out, 1 stop bit
                                ;and transmit enable
        ld      UIE,#00000000B  ;no interrupts, no DMA
        ld      URC,#00000010B  ;enable receive

;UART is initialized, enable interrupts for real time clock
;
        ei                      ;enable interrupts
;
;because uart was just enabled, allow data line to mark for at least 1 second
;
WAIT:   cp      second,#1
        jr      ne,WAIT         ;wait 1 second
;
;display the logon message
;
LOGON:  ldw     MPTR,#MSG       ;load the address of MSG into word reg MPTR
        call    SENDM           ;send the message
        call    TXWAT           ;wait for transmitter to complete
;
;logon message displayed, get response from console
;and move to upper register memory
;
GET:    di                      ;no interrupts while setting up for DMA
        ldw     MPTR,#0080H     ;first character receive location
        and     EMT,#11111110B  ;select register file for receiving character
        sb1                     ;select bank one
        ld      DCH,#0          ;DMA count high byte
        ld      DCL,#25         ;DMA count low byte
        or      UMB,#00100000B  ;auto echo enable
        sb0                     ;restore to bank zero
        or      UIE,#00000011B  ;receive character DMA link, interrupt enable
        ei
        call    RXWAT           ;wait for receiver to complete receiving input
;
;receive characters in buffer, restore Super8 non DMA state
;
        di                      ;no interrupts while cleaning up
        sb1                     ;bank 1
        and     UMB,#11011111B  ;disable auto echo
        sb0                     ;restore bank 0
        or      EMT,#00000001B  ;select data memory for DMA transfers
        ei
;
;25 characters received via DMA, now display "ECHO" message
;
ECHO:   ldw     MPTR,#MSG1      ;load the address of MSG1 in word reg MPTR
        call    SENDM           ;send the message
        call    TXWAT           ;wait on transmitter
;
;message sent, now replay typed input
;
        di
        ldw     MPTR,#0080H     ;point to beginning of buffer
        and     EMT,#11111110B  ;select register bank for DMA transfer
        sb1                     ;select bank 1
        ld      DCH,#0          ;DMA count high byte
        ld      DCL,#25         ;DMA count low byte
        sb0                     ;select bank 0
        or      UIE,#00000100B  ;enable transmit interrupts
        or      UTC,#00000001B  ;transmit DMA enable
        ei                      ;enable interrupts
        call    TXWAT           ;wait on transmitter
        di
        or      EMT,#00000001B  ;select external data memory for DMA transfer
        ei
;
;replay complete, loop back and do it again
;
        jr      LOGON
```

```
;
;subroutines
;
;send message at MPTR for length in first byte
SENDM:  ldci    r7,@MPTR        ;get the character
        dec     r7              ;count actually should be n-1 for n bytes
        di                      ;no interrupts while setting up
        or      EMT,#00000001B  ;select external data memory for DMA transfer
        sb1                     ;select bank 1
        ld      DCH,#0          ;DMA count high byte is 0
        ld      DCL,r7          ;move the count DMA count low byte
        sb0                     ;select bank 0
        or      UIE,#00000100B  ;enable transmit interrupts
        or      UTC,#00000001B  ;transmit DMA enable
        ei
        ret
;transmit DMA complete
TXDATI: and     UIE,#11111011B  ;disable transmit interrupts
        and     UTC,#11111110B  ;disable transmit DMA
        iret                    ;ignore it if no character to transmit
;transmitter wait routine
TXWAT:  tm      UIE,#00000100B  ;wait until interrupts disabled
        jr      nz,TXWAT        ;wait if bit set
        ret
;receive character available interrupt
RXDATI: and     UIE,#11111100B  ;disable the receive interrupts
        iret
;receive wait routine
RXWAT:  tm      UIE,#00000001B  ;wait until interrupts disabled
        jr      nz,RXWAT        ;wait if bit still set
        ret
;
;real time interrupt running in background
;
TIMER0: inc     period          ;bump periodic counter (60 hertz)
        cp      period,#60      ;one second yet?
        jr      ne,NOROLL       ;no rollover
        xor     P2,#00000001B   ;complement the second bit
        clr     period          ;start it over again
        inc     second          ;bump the seconds timer
        cp      second,#60      ;reached maximum
        jr      ne,NOROLL       ;no rollover
        xor     P2,#00000010B   ;complement the minute bit
        clr     second          ;start it over again
        inc     minute          ;bump the minutes timer
        cp      minute,#60      ;reached maximum
        jr      ne,NOROLL       ;no rollover
        xor     P2,#00000100B   ;complement the hour bit
        clr     minute          ;start it over again
        inc     hours           ;bump the hours timer
        cp      hours,#24       ;reached maximum
        jr      ne,NOROLL       ;no rollover
        clr     hours           ;start it over again
NOROLL: or      COCT,#00000010B ;reset end of count
        nop
        nop
INTRET: iret                    ;and return from interrupt
;
;
;
MSG:    .BYTE   56
        .ASCII  CR,LF,'Super8 Uart DMA test program.',CR,LF
        .ASCII  'Enter 25 characters',CR,LF,'$'
MSG1:   .BYTE   34
        .ASCII  CR,LF,'Echoed back, your line was...',CR,LF,'$'


        .END
```

GENERATING SINE WAVES WITH THE ZILOG SUPER8

by Charles M. Link, II

Generally digital microprocessors are thought of as only being able to generate digital signals...that is either on or off. With the simple addition of a digital-to-analog converter (DAC), more complex waveforms may be generated. Since the advent of the microprocessor and the DAC, many methods have been used by hardware and software designers to generate sine waves, including some that involve precise instruction and clock cycle calculations. This example is different.

The Zilog Super8 microcomputer is a single chip device requiring only a latch and EPROM to operate in its ROMLESS state. Leaving 24 I/O lines for user configuration, it is extremely easy to interface with peripherals, including, in this case, the DAC- 08. The hardware in this application example is essentially the same base hardware as the previous application articles. Since it is assumed that the reader has access to those articles, detailed explaination of the base will not be made here. Only the additions to the base will be explained.

The base Super8 microprocessor has ports 2, 3 and 4 available for user connection. For this example, the DAC-08 is connected to port 4 (P4). The DAC-08 is tied, with the least significant bit tied to P40 and the most significant bit tied to P47. The other connections to the DAC-08 are mostly out of the test circuit description shown in the data manuals associated with it. The DAC requires -12 volts for proper operation. The output for this example is tied to a simple op- amp filter with a sharp roll off at about 3500 hertz. This type filter might be quite suitable for telecommunications applications, but may not be so good for many others. An oscilloscope displays the resultant waveform.

The software to operate the Super8 is in the original initialization software from eariler in this article series. Initialization is essentially the same. Port 4 must be set up as output, with active push-pull drivers. The main consideration for this program is the software "sample" rate. For this example, 8000 samples per second was chosen. Any other rate may be chosen, and the author has successfully used values up to 16000 samples per second without timing problems. Higher base clock rates are possible with the recently introducecd 20 megahertz Super8 chips available. With the sample method used, the sample rate does not vary with the different sine wave frequencies generated.

The sample method requires a sine wave table stored in ROM or EPROM. This example uses 256 values, although 64, 128 or more values are quite acceptable. The BASICA program that generated the sine table is included for user modification. Once the values were generated, they were manually typed into the program. Using the Zilog macro assembler would have signigicantly slowed assembling. Note that the comments in the BASICA program ìmust be removed before the PC can execute.

The values generated by the BASICA program are values ranging from 01H to 0FEH. Since the DAC represents 00H as zero volts and 0FFH as 5 volts, this table will product sine outputs from almost zero to almost five volts.

The principle of operation requires that a sixteen bit frequency increment be maintained. This increment is generated by the simple formula

FREQUENCY INCREMENT = (TABLESTEP X 256 X FREQUENCY) / SAMPLE

where FREQUENCY INCREMENT is a sixteen bit value saved in an increment register, TABLESTEP is the number of values in the sine wave table, FREQUENCY is the desired frequency of generation in hertz, and SAMPLE is the number of samples per second. In the example program, this increment is stored in "FINCR".

A current offset into the sine table is maintained in the register pair labeled "INCR". At each periodic interrupt, FINCR must be added to INCR and saved in INCR. This sixteen bit value remains the offset into the table. The upper byte of the offset is used to point to the value in the 256 byte sine table that is loaded into the DAC. In the sample program, the value loaded into the DAC is generated in the previous interrupt and saved until the first instruction of the next interrupt. This allows the interrupt to perform some other varying length transactions, without introducing bit jitter into the sine wave.

Changing the "FINCR" by program control causes different frequencies to be generated. In this case, the sine wave may be turned off by disabling the counter 0 interrupt. Depending upon the number of steps in the sine

table and the sample frequency, very accurate sine frequencies may be generated. Calculate the actual error by using the following formula:

[ABS (REAL FREQI - INTEGER FREQI) / REAL FREQI] X 100 = % ERROR

where REAL FREQI is the actual calculated frequency increment, INTEGER FREQI is the nearest rounded integer of the calculated frequency increment, and the result is the actual percent error form the desired value.

With the addition of a filter with sharp cutoff just above the highest desired frequency, the Super8 serves quite well as a programmable sine wave generator. In addition to sine waves, complex waveforms may be easily generated by the Super8 with the addition of the low-cost DAC. The next article in this series will describe how to generate some of these more complex waveforms.

```
;
        .TITLE  Super8 Example Sine Wave Generation
;
;=============================================================
;=      TITLE:          SINE.S                               =
;=      DATE:           JUNE 17, 1986                        =
;=      PURPOSE:        TO DEMONSTRATE USING SUPER8          =
;=                      TO GENERATE HIGH QUALITY SINE        =
;=                      WAVES.                               =
;=      HARDWARE:       DAC-08 ON PORT 4                     =
;=                      SEE DIAGRAM                          =
;=      ASSEMBLER:      ZILOG ASMS8 ASSEMBLER                =
;=      PROGRAMMER:     CHARLES M. LINK, II                  =
;=============================================================
;
;
;
        .PAGE   55      ;set maximum page size to 55 lines
;
;************************************************************
;*                                                          *
;*                  REGISTER EQUATE TABLE                   *
;*                                                          *
;************************************************************
;
INCR:   .equ    rr0             ;current increment in sine table
INCRH:  .equ    r0              ;high byte of current increment value
INCRL:  .equ    r1              ;low byte of current increment value
FINCR:  .equ    rr2             ;increment in sine table for frequency
FINCRH: .equ    r2              ;high byte of frequency increment value
FINCRL: .equ    r3              ;low byte of frequency increment value
POINT:  .equ    rr4             ;pointer into sine table
POINTH: .equ    r4              ;high byte of sine table pointer
POINTL: .equ    r5              ;low byte of sine table pointer
CVAL:   .equ    r6              ;current value to output to DAC-08
;
;************************************************************
;*                                                          *
;*                    GENERAL EQUATES                       *
;*                                                          *
;************************************************************
;
XTAL:   .equ    12000000                ;crystal freq in hertz
SAMPLE: .equ    8000                    ;sample frequency in hertz
CTVAL:  .equ    XTAL/4/SAMPLE           ;counter load value
TABSTP: .equ    256                     ;number of values in sine table
FREQ:   .equ    697                     ;desired sine wave frequency
FREQI:  .equ    (TABSTP*256*FREQ)/SAMPLE
;
;************************************************************
;*                                                          *
;*                  INTERRUPT VECTOR TABLE                  *
;*                                                          *
;************************************************************
;
INTR0:  .WORD   INTRET          ;this area should always be defined
INTR1:  .WORD   INTRET          ;as it reserves the lower 32 bytes
INTR2:  .WORD   INTRET          ;for the interrupt table.  the name
INTR3:  .WORD   INTRET          ;of the subroutine for each particular
INTR4:  .WORD   INTRET          ;interrupt service would normally be
INTR5:  .WORD   INTRET          ;named here.
INTR6:  .WORD   TIMER0
INTR7:  .WORD   INTRET
```

```
INTR8:  .WORD   INTRET
INTR9:  .WORD   INTRET
INTR10: .WORD   INTRET
INTR11: .WORD   INTRET
INTR12: .WORD   INTRET
INTR13: .WORD   INTRET
INTR14: .WORD   INTRET
INTR15: .WORD   INTRET
;
;*********************************************************
;*                                                       *
;*              START OF PROGRAM EXECUTION               *
;*                                                       *
;*********************************************************
;
START:  jr      START1          ;program execution unconditionally
                                ;begins at this location after reset
                                ;and power up.
        .ASCII  'REL 0 6/16/86' ;jump around optional ascii string
                                ;containing release info, copyright, etc.
START1: di                      ;begin
        sb0                     ;select register bank 0
        ld      EMT,#00000000B  ;external memory timing=no wait input, normal
                                ;memory timing, no wait states, stack internal,
                                ;and DMA internal
        ld      P0,#00H         ;address begins at 0000h, set upper byte
        ld      P0M,#11111111B  ;select all lines as address
        ld      PM,#00110000B   ;enable port 0 as upper 8 bits address
        ld      H1C,#00000000B  ;handshake not enabled port 0
;
;port 1 is defined in romless part as address/data.  it is not necessary
;here to initialize that port
;
        ld      P2,#00H         ;port 2 outputs low
        ld      P3,#00H         ;port 3 outputs low
        ld      P2AM,#10101010B ;p30,31,20,21 as output
        ld      P2BM,#10101010B ;p32,33,22,23 as output
        ld      P2CM,#10101010B ;p34,35,24,25 as output
        ld      P2DM,#10101010B ;p36,37,26,27 as output
;
        ld      P4,#10000000B   ;set midpoint for DAC inputs
        ld      P4D,#00000000B  ;set all bits of P4 as output
        ld      P4OD,#00000000B ;active push/pull
;
;basic Super 8 I/O is initialized, now internal registers
;
        ld      RP0,#0C0H       ;set working register low to lower 8 bytes
        ld      RP1,#0C8H       ;set working register high to upper 8 bytes
        ld      SPL,#0FFH       ;set stack pointer to start at top of set two
                                ;note here that only lower 8 bits are used
                                ;for stack pointer.  location 0FFH is wasted
                                ;as stack operation.  SPH is general purpose
                                ;storage.
;
;now clear the internal memory and stack area
;
        ld      SPH,#0FFH       ;point to top of general purpose register
ZERO:   clr     @SPH            ;zero it
        dec     SPH
        jr      nz,ZERO         ;do it until register set is all cleared
        clr     @SPH            ;zero last register
;
;now everything except working registers is cleared
;
;cpu and memory now initialized, set up timer for real time clock
;
        ld      SYM,#00000000B  ;disable fast interrupt response
        ld      IPR,#00000010B  ;interrupt priority
                                ;IRQ2>IRQ3>IRQ4>IRQ5>IRQ6>IRQ7>IRQ0>IRQ1
        ld      IMR,#00000100B  ;enable only interrupt 2
        sb1                     ;select bank 1
        ld      C0TCH,#^HB(CTVAL)       ;high byte of time constant
        ld      C0TCL,#^LB(CTVAL)       ;low byte of time constant
        ld      C0M,#00000100B  ;p27,37 is I/O, programmed up/down, no capture
                                ;timer mode is selected
        sb0                     ;select bank 0
        ld      C0CT,#10100101B ;continuous, count down, load counter,
```

```
                                ;zero count interrupt enable, enable counter
;
;timer is initialized, now lets enable interrupts and wait
        ldw     INCR,#1         ;start at the beginning of sine table
        ldw     FINCR,#FREQI    ;load frequency of increment
        ldw     POINT,#SINTAB   ;pointer points to sine table
        ld      CVAL,#080H      ;initial value to prevent glitch at start
        ei                      ;enable interrupts
WAIT:   nop
        nop
        nop
        nop
        jr      WAIT            ;loop back
;
;Timer interrupt.  Occurs SAMPLE times per second
;interrupt outputs value to DAC-08 and then determines value for next
;interrupt.  This assures no bit jitter.
;
TIMER0: ld      p4,CVAL         ;write new value to DAC-08
        rcf                     ;clear carry flag
        add     INCRL,FINCRL    ;find next position in sine table
        adc     INCRH,FINCRH    ;by adding frequency offset to last position
        ld      POINTL,INCRH    ;set new pointer into sine table
                                ;upper byte ok since on boundary
        ldc     CVAL,@POINT     ;get value from sine table
        or      COCT,#00000010B ;reset end of count interrupt
INTRET: iret                    ;and return from interrupt
;
;*************************************************************
;*                                                           *
;*                      SINE WAVE LOOKUP                     *
;*                                                           *
;*************************************************************
;
;sine table for sine wave generation using DAC-08.  Table based upon
;case of waveform with minumum amplititude = 0 volts and maximum
;amplititude = 5 volts.  DAC-08 input for 0 volts = 00H
;5 volts = 0FFH.  Table generated using following BASICA program,
;then typed into program.
;
;       10 CLS                  ;clear screen
;       20 PI=3.141593          ;define PI
;       30 FOR I=0 TO 255       ;256 total values
;       40 C=360/256            ;define basic interval value
;       50 D=C*I                ;value from zero on sine wave
;       60 E=D*PI/180
;       70 F=SIN(E)             ;figure sine for interval from 0
;       80 G=F*127              ;sine range should be from -127 to 127
;       90 H=128+G              ;make result from 0 to 255
;       100 J=CINT(H)           ;round to nearest integer
;       110 A$=HEX$(J)          ;convert to hex
;       120 PRINT A$            ;on screen
;       130 LPRINT A$           ;on printer
;       140 NEXT                ;do next inverval
;       150 END
;
;*note-remove comments, BASICA will not accept ; as comment delimiter
;
;
SINTAB: .ORG    0400H           ;begin sine table on even byte boundary
        .byte   080H,083H,086H,089H,08CH,090H,093H,096H,099H,09CH,09FH,0A2H
        .byte   0A5H,0A8H,0ABH,0AEH,0B1H,0B3H,0B6H,0B9H,0BCH,0BFH,0C1H,0C4H
        .byte   0C7H,0C9H,0CCH,0CEH,0D1H,0D3H,0D5H,0D8H,0DAH,0DCH,0DEH,0E0H
        .byte   0E2H,0E4H,0E6H,0E8H,0EAH,0EBH,0EDH,0EFH,0F0H,0F1H,0F3H,0F4H
        .byte   0F5H,0F6H,0F8H,0F9H,0FAH,0FAH,0FBH,0FCH,0FDH,0FDH,0FEH,0FEH
        .byte   0FEH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FEH,0FEH,0FEH,0FDH
        .byte   0FDH,0FCH,0FBH,0FAH,0FAH,0F9H,0F8H,0F6H,0F5H,0F4H,0F3H,0F1H
        .byte   0F0H,0EFH,0EDH,0EBH,0EAH,0E8H,0E6H,0E4H,0E2H,0E0H,0DEH,0DCH
        .byte   0DAH,0D8H,0D5H,0D3H,0D1H,0CEH,0CCH,0C9H,0C7H,0C4H,0C1H,0BFH
        .byte   0BCH,0B9H,0B6H,0B3H,0B1H,0AEH,0ABH,0A8H,0A5H,0A2H,09FH,09CH
```

```
.byte   099H,096H,093H,090H,08CH,089H,086H,083H,080H,07DH,07AH,077H
.byte   074H,070H,06DH,06AH,067H,064H,061H,05EH,05BH,058H,055H,052H
.byte   04FH,04DH,04AH,047H,044H,041H,03FH,03CH,039H,037H,034H,032H
.byte   02FH,02DH,02BH,028H,026H,024H,022H,020H,01EH,01CH,01AH,018H
.byte   016H,015H,013H,011H,010H,00FH,00DH,00CH,00BH,00AH,008H,007H
.byte   006H,006H,005H,004H,003H,003H,002H,002H,002H,001H,001H,001H
.byte   001H,001H,001H,001H,002H,002H,002H,003H,003H,004H,005H,006H
.byte   006H,007H,008H,00AH,00BH,00CH,00DH,00FH,010H,011H,013H,015H
.byte   016H,018H,01AH,01CH,01EH,020H,022H,024H,026H,028H,02BH,02DH
.byte   02FH,032H,034H,037H,039H,03CH,03FH,041H,044H,047H,04AH,04DH
.byte   04FH,052H,055H,058H,05BH,05EH,061H,064H,067H,06AH,06DH,070H
.byte   074H,077H,07AH,07DH

.END
```

GENERATING DTMF TONES WITH THE ZILOG SUPER8

by Charles M. Link, II

In the previous article, a sine wave generation example was demonstrated. Sine waves are great, but, sometimes, more complex waveforms must be generated. One of the most widely used complex waveforms is the DTMF tone. The DTMF tone is used on millions of telephones under the AT&T registered name "TOUCH TONE". Generally, telecommunications designers purchase one of the many DTMF encoder chips and hang it beside a microprocessor. This application article contains an example of a DTMF generation scheme that produces nearly as pure and probably as accurate a tone as the external chip method.

Generating sine waves requires some type of digital-to-analog converter to interface to the microprocessor. For this application, a DAC-08 is used. This DAC-08 is tied to port 4 of the Super8. Since it is assumed that the reader has access to the previous article, a detailed description of the hardware will be left to that article. Why not use the DTMF generator chip, when it might be just as inexpensive as the DAC- 08? The answer is that the DTMF generator chip requires an external crystal or clock, and it might not be convenient to pick a processor frequency that is a direct multiple of the one required by the generator. The second and more important reason is that the DAC-08 can be used to generate other call progress tones such as ringback and busy, or any other complex waveform.

Since the previous article discussed the method for generating sine wave tones, this article will only discuss how to turn that into the DTMF tone. The DTMF tone is actually a combination of two tones, hence, the name DUAL TONE MULTI-FREQUENCY. The tones are arranged such that each row and each column has a corresponding single frequency tone assigned. An additional, normally unseen column, contains an eighth tone frequency. A simple diagram below shows the arrangement.

DTMF TONE ASSIGNMENT

	1209	1336	1477	1633
697	1	2	3	A
770	4	5	6	B
852	7	8	9	C
941	*	0	#	D

The method used to combine the two tones into one single complex waveform is simple: add the two individual tones together. Adding the tones together is usually what happens when analog circuitry produces the DTMF tone. In fact, most of the DTMF encoder chips usually add the tones together either internally or externally to produce the single waveform.

Generating the two tones is no task for the Super8 microcomputer. Just set up two current table offset values and two different frequency increments. At each periodic interrupt the 16 bit frequency increment is added to the current table offset producing a new current table offset. The upper byte of each current table offset (one for the row frequency and one for the column) is used as a pointer into a 256 byte table. The sine values retrieved from the table are then added together and loaded into the DAC-08.

Since the DAC input of 00H corresponds to an output of 0 volts and the input of 0FFH corresponds to an output of 5 volts, adding two values that could possibly be 0FFH presents a problem. Since two sines must add to no more 5 volts, the maximum for one single sine value must be one half of 5 volts, or 80H. The sine table has been adjusted so that the 2.5 volt value is mid-range. The maximum or mimumum for the sine wave is plus or minus 1.25 volts.

The interrupt service routine is almost exactly the same as the interrupt routine for the sine wave, except that two sine waves are calculated. The final values are added together and stored for the first instruction of the next interrupt. In order to change tones, or disable the tone generation, additional software logic could enable or disable the interrupt, and modify the two values "CINCR", and "RINCR".

It is clear from the example, that ringback, busy, MF, and other signaling tones can be easily generated without additional hardware. Increased sampling rates could be used to generate tones of much higher frequencies and accuracies. The accuracy, using the above method and sampling frequencies, is much less than one percent, totally suitable for telecommunications needs.

```
;
        .TITLE  Super8 Example DTMF Generation
;
;=============================================================
;=      TITLE:          DTMF.S                               =
;=      DATE:           JUNE 17, 1986                        =
;=      PURPOSE:        TO DEMONSTRATE USING SUPER8          =
;=                      TO GENERATE HIGH QUALITY DTMF        =
;=                      WAVES.                               =
;=      HARDWARE:       DAC-08 ON PORT 4                     =
;=                      SEE DIAGRAM                          =
;=      ASSEMBLER:      ZILOG ASMS8 ASSEMBLER                =
;=      PROGRAMMER:     CHARLES M. LINK, II                  =
;=============================================================
;
;
;
        .PAGE   55      ;set maximum page size to 55 lines
;
;************************************************************
;*                                                          *
;*                  REGISTER EQUATE TABLE                   *
;*                                                          *
;************************************************************
;
;column tone equates
CINCR:  .equ    rr0             ;current increment in sine table
CINCRH: .equ    r0              ;high byte of current increment value
CINCRL: .equ    r1              ;low byte of current increment value
CFINCR: .equ    rr2             ;increment in sine table for frequency
CFINCH: .equ    r2              ;high byte of frequency increment value
CFINCL: .equ    r3              ;low byte of frequency increment value
POINT:  .equ    rr4             ;pointer into sine table
POINTH: .equ    r4              ;high byte of sine table pointer
POINTL: .equ    r5              ;low byte of sine table pointer
;row tone equates
RINCR:  .equ    rr6             ;current increment in sine table
RINCRH: .equ    r6              ;high byte of current increment value
RINCRL: .equ    r7              ;low byte of current increment value
RFINCR: .equ    rr8             ;increment in sine table for frequency
RFINCH: .equ    r8              ;high byte of frequency increment value
RFINCL: .equ    r9              ;low byte of frequency increment value
CVAL:   .equ    r10             ;current value to output to DAC-08
RVAL:   .equ    r11             ;current row value
;
;************************************************************
;*                                                          *
;*                    GENERAL EQUATES                       *
;*                                                          *
;************************************************************
;
XTAL:   .equ    12000000                ;crystal freq in hertz
SAMPLE: .equ    8000                    ;sample frequency in hertz
CTVAL:  .equ    XTAL/4/SAMPLE           ;counter load value
TABSTP: .equ    256                     ;number of values in sine table
CFREQ:  .equ    1209                    ;desired column frequency
RFREQ:  .equ    697                     ;desired row frequency
CFREQI: .equ    (TABSTP*256*CFREQ)/SAMPLE
RFREQI: .equ    (TABSTP*256*RFREQ)/SAMPLE
;note dtmf frequencies are 697,770,852,941,1209,1336,1477,1633
;
;************************************************************
;*                                                          *
;*                  INTERRUPT VECTOR TABLE                  *
;*                                                          *
;************************************************************
;
INTR0:  .WORD   INTRET          ;this area should always be defined
INTR1:  .WORD   INTRET          ;as it reserves the lower 32 bytes
INTR2:  .WORD   INTRET          ;for the interrupt table.  the name
INTR3:  .WORD   INTRET          ;of the subroutine for each particular
INTR4:  .WORD   INTRET          ;interrupt service would normally be
INTR5:  .WORD   INTRET          ;named here.
INTR6:  .WORD   TIMER0
INTR7:  .WORD   INTRET
INTR8:  .WORD   INTRET
INTR9:  .WORD   INTRET
INTR10: .WORD   INTRET
```

```
INTR11: .WORD   INTRET
INTR12: .WORD   INTRET
INTR13: .WORD   INTRET
INTR14: .WORD   INTRET
INTR15: .WORD   INTRET
;
;*********************************************************
;*                                                       *
;*              START OF PROGRAM EXECUTION               *
;*                                                       *
;*********************************************************
;
START:  jr      START1          ;program execution unconditionally
                                ;begins at this location after reset
                                ;and power up.
        .ASCII  'REL 0 6/16/86' ;jump around optional ascii string
                                ;containing release info, copyright, etc.
START1: di                      ;begin
        sb0                     ;select register bank 0
        ld      EMT,#00000000B  ;external memory timing=no wait input, normal
                                ;memory timing, no wait states, stack internal,
                                ;and DMA internal
        ld      P0,#00H         ;address begins at 0000h, set upper byte
        ld      P0M,#11111111B  ;select all lines as address
        ld      PM,#00110000B   ;enable port 0 as upper 8 bits address
        ld      H1C,#00000000B  ;handshake not enabled port 0
;
;port 1 is defined in romless part as address/data.  it is not necessary
;here to initialize that port
;
        ld      P2,#00H         ;port 2 outputs low
        ld      P3,#00H         ;port 3 outputs low
        ld      P2AM,#10101010B ;p30,31,20,21 as output
        ld      P2BM,#10101010B ;p32,33,22,23 as output
        ld      P2CM,#10101010B ;p34,35,24,25 as output
        ld      P2DM,#10101010B ;p36,37,26,27 as output
;
        ld      P4,#10000000B   ;set midpoint for DAC inputs
        ld      P4D,#00000000B  ;set all bits of P4 as output
        ld      P4OD,#00000000B ;active push/pull
;
;basic Super 8 I/O is initialized, now internal registers
;
        ld      RP0,#0C0H       ;set working register low to lower 8 bytes
        ld      RP1,#0C8H       ;set working register high to upper 8 bytes
        ld      SPL,#0FFH       ;set stack pointer to start at top of set two
                                ;note here that only lower 8 bits are used
                                ;for stack pointer.  location 0FFH is wasted
                                ;as stack operation.  SPH is general purpose
                                ;storage.
;
;now clear the internal memory and stack area
;
        ld      SPH,#0FFH       ;point to top of general purpose register
ZERO:   clr     @SPH            ;zero it
        dec     SPH
        jr      nz,ZERO         ;do it until register set is all cleared
        clr     @SPH            ;zero last register
;
;now everything except working registers is cleared
;
;cpu and memory now initialized, set up timer for real time clock
;
        ld      SYM,#00000000B  ;disable fast interrupt response
        ld      IPR,#00000010B  ;interrupt priority
                                ;IRQ2>IRQ3>IRQ4>IRQ5>IRQ6>IRQ7>IRQ0>IRQ1
        ld      IMR,#00000100B  ;enable only interrupt 2
        sb1                     ;select bank 1
        ld      C0TCH,#^HB(CTVAL)       ;high byte of time constant
        ld      C0TCL,#^LB(CTVAL)       ;low byte of time constant
        ld      C0M,#00000100B  ;p27,37 is I/O, programmed up/down, no capture
                                ;timer mode is selected
        sb0                     ;select bank 0
        ld      C0CT,#10100101B ;continuous, count down, load counter,
                                ;zero count interrupt enable, enable counter
;
;timer is initialized, now lets enable interrupts and wait
        ldw     CINCR,#1        ;start column at beginning of sine table
        ldw     RINCR,#1        ;start row at beginning of sine table
```

```
;
;this example loads the tones for digit '1'
;user software would, of course have to manipulate these registers for
;proper tone control
;
        ldw     CFINCR,#CFREQI  ;load column frequency increment
        ldw     RFINCR,#RFREQI  ;load row frequency increment
        ldw     POINT,#SINTAB   ;pointer points to sine table
        ld      CVAL,#080H      ;initial value to prevent glitch at start
        ei                      ;enable interrupts
WAIT:   nop
        nop
        nop
        nop
        jr      WAIT            ;loop back
;
;Timer interrupt.  Occurs SAMPLE times per second
;interrupt outputs value to DAC-08 and then determines value for next
;interrupt.  This assures no bit jitter.
;
TIMER0: ld      p4,CVAL         ;write new value to DAC-08
        rcf                     ;clear carry flag
        add     CINCRL,CFINCL   ;find next position in sine table
        adc     CINCRH,CFINCH   ;by adding frequency offset to last position
        ld      POINTL,CINCRH   ;set new pointer into sine table
        ldc     CVAL,@POINT     ;get value from sine table
        add     RINCRL,RFINCL   ;find next position in sine table
        adc     RINCRH,RFINCH   ;by adding frequencty offset to last position
        ld      POINTL,RINCRH   ;set new pointer into sine table
        ldc     RVAL,@POINT     ;get second value from sine table
        add     CVAL,RVAL       ;form a complex waveform from two sine values
        or      COCT,#00000010B ;reset end of count interrupt
INTRET: iret                    ;and return from interrupt
;
;*********************************************************
;*                                                       *
;*                      SINE WAVE LOOKUP                 *
;*                                                       *
;*********************************************************
;
;sine table for DTMF generation using DAC-08.  Table based upon
;case of waveform consisting of two sine waves summed to provide a single
;complex waveform with minumum amplititude = 0 volts and maximum
;amplitude = 5 volts.  DAC-08 input for 0 volts = 00H
;5 volts = 0FFH.  Both waves must total no more than 0FFH, therefore
;maximum for one wave must be 1/2 5 volts or 080H.
;Table generated using following BASICA program,
;then typed into program.
;
;       10 CLS                  ;clear screen
;       20 PI=3.141593          ;define PI
;       30 FOR I=0 TO 255       ;256 total values
;       40 C=360/256            ;define basic interval value
;       50 D=C*I                ;value from zero on sine wave
;       60 E=D*PI/180
;       70 F=SIN(E)             ;figure sine for interval from 0
;       80 G=F*63               ;sine range should be from -63 to 63
;       90 H=64+G               ;make result from 0 to 127
;       100 J=CINT(H)           ;round to nearest integer
;       110 A$=HEX$(J)          ;convert to hex
;       120 PRINT A$            ;on screen
;       130 LPRINT A$           ;on printer
;       140 NEXT                ;do next inverval
;       150 END
;
;*note-remove comments, BASICA will not accept ; as comment delimiter
;
;
SINTAB: .ORG    0400H           ;begin sine table on even byte boundary
        .byte   040H,042H,043H,045H,046H,048H,049H,04BH,04CH,04EH,04FH,051H
        .byte   052H,054H,055H,057H,058H,05AH,05BH,05CH,05EH,05FH,060H,062H
        .byte   063H,064H,066H,067H,068H,069H,06AH,06BH,06DH,06EH,06FH,070H
        .byte   071H,072H,073H,074H,074H,075H,076H,077H,078H,078H,079H,07AH
        .byte   07AH,07BH,07BH,07CH,07CH,07DH,07DH,07DH,07EH,07EH,07EH,07FH
        .byte   07FH,07FH,07FH,07FH,07FH,07FH,07FH,07FH,07FH,07FH,07EH,07EH
        .byte   07EH,07DH,07DH,07DH,07CH,07CH,07BH,07BH,07AH,07AH,079H,078H
        .byte   078H,077H,076H,075H,074H,074H,073H,072H,071H,070H,06FH,06EH
        .byte   06DH,06BH,06AH,069H,068H,067H,066H,064H,063H,062H,060H,05FH
        .byte   05EH,05CH,05BH,05AH,058H,057H,055H,054H,052H,051H,04FH,04EH
        .byte   04CH,04BH,049H,048H,046H,045H,043H,042H,040H,03EH,03DH,03BH
        .byte   03AH,038H,037H,035H,034H,032H,031H,02FH,02EH,02CH,02BH,029H
        .byte   028H,026H,025H,024H,022H,021H,020H,01EH,01DH,01CH,01AH,019H
        .byte   018H,017H,016H,015H,013H,012H,011H,010H,00FH,00EH,00DH,00CH
        .byte   00CH,00BH,00AH,009H,008H,008H,007H,006H,006H,005H,005H,004H
        .byte   004H,003H,003H,003H,002H,002H,002H,001H,001H,001H,001H,001H
        .byte   001H,001H,001H,001H,001H,001H,002H,002H,002H,003H,003H,003H
        .byte   004H,004H,005H,005H,006H,006H,007H,008H,008H,009H,00AH,00BH
        .byte   00CH,00CH,00DH,00EH,00FH,010H,011H,012H,013H,015H,016H,017H
        .byte   018H,019H,01AH,01CH,01DH,01EH,020H,021H,022H,024H,025H,026H
        .byte   028H,029H,02BH,02CH,02EH,02FH,031H,032H,034H,035H,037H,038H
        .byte   03AH,03BH,03DH,03EH
        .END
```

A SIMPLE SERIAL TO PARALLEL CONVERTER USING THE ZILOG SUPER8

by Charles M. Link, II

The Zilog Super8 has many on-board peripherals that provide multiple user applications. Earlier articles have demonstrated simple application "stubs" or short test programs. This article and the next article demonstrate a useful application for the Super8. Although it underutilizes the Super8's power, the simple serial to parallel converter in this application and the print buffer in the next application demonstrate the ease at which applications are developed with the Super8.

The Zilog Super8 has several features that enhance its use as a communication controller. The interrupt or DMA driven serial port are helpful, but the handshaking parallel ports finish the job. In the serial to parallel converter, the 256 byte internal register memory is used as a small circular queue.

Hardware for this application is fairly simple. Port 4 is buffered and hooked to the data lines, as shown, to interface to a centronics type printer connector. The strobe from P25 provides the strobe (pin 1) to the printer. The acknowledge line from the printer is inverted and tied to P24 of the Super8. The busy signal from the printer is buffered and tied to P23 of the Super8. The design was tested on an Okidata printer and is not guaranteed to work on all printers.

Software is fairly straightforward. The serial port is initialized just like it was in the application article on the interrupt driven serial port. Port 4 must be set-up as outputs with active push-pull drivers. Port 2, bits 3 and 4, are set up as input with P24 set to enable interrupts. P25 is set as output and handshake 0 is set in H0C to provide a strobe of 16 clock periods in length.

```
;
        .TITLE  Sample Zilog Super 8 Serial to Parallel Converter
;
;
;=============================================================
;=      TITLE:          SERPAR.S                             =
;=      DATE:           JULY 17, 1986                        =
;=      PURPOSE:        TO DEMONSTRATE INTERRUPT             =
;=                      DRIVEN SERIAL PORT IN A              =
;=                      REALISTIC APPLICATION.               =
;=                      THIS APPLICATION RECEIVES            =
;=                      SIMPLE SERIAL DATA A SENDS IT        =
;=                      OUT THE PARALLEL PORT TO A           =
;=                      PRINTER.                             =
;=      ASSEMBLER:      ZILOG ASMS8 ASSEMBLER                =
;=      PROGRAMMER:     CHARLES M. LINK, II                  =
;=============================================================
;
;
;
        .PAGE   55      ;set maximum page size to 55 lines
;*************************************************************
;*                                                           *
;*                    GENERAL EQUATES                        *
;*                                                           *
;*************************************************************
;
CR:     .equ    0dH     ;carriage return
LF:     .equ    0aH     ;line feed
;
;
;*************************************************************
;*                                                           *
;*                  REGISTER EQUATE TABLE                    *
;*                                                           *
;*************************************************************
;
;working register equates
INPNT:  .equ    R3      ;input character pointer
OUTPNT: .equ    R4      ;output character pointer
```

```
MPTR:   .equ    RR6     ;message pointer for external memory
ACKB:   .equ    R5      ;byte containing acknowledge bit
ACKBIT: .equ    0       ;bit set = no acknowledge yet
                        ;bit clear = not waiting on acknowledge
;
;*********************************************************
;*                                                       *
;*                  INTERRUPT VECTOR TABLE               *
;*                                                       *
;*********************************************************
;
INTR0:  .WORD   INTRET          ;this area should always be defined
INTR1:  .WORD   INTRET          ;as it reserves the lower 32 bytes
INTR2:  .WORD   INTRET          ;for the interrupt table.  the name
INTR3:  .WORD   INTRET          ;of the subroutine for each particular
INTR4:  .WORD   INTRET          ;interrupt service would normally be
INTR5:  .WORD   INTRET          ;named here.
INTR6:  .WORD   INTRET
INTR7:  .WORD   INTRET
INTR8:  .WORD   INTRET
INTR9:  .WORD   INTRET
INTR10: .WORD   RXDATI          ;receive data interrupt
INTR11: .WORD   INTRET
INTR12: .WORD   INTRET
INTR13: .WORD   INTRET
INTR14: .WORD   ACKSTB          ;acknowledge strobe interrupt
INTR15: .WORD   INTRET
;
;*********************************************************
;*                                                       *
;*              START OF PROGRAM EXECUTION               *
;*                                                       *
;*********************************************************
;
START:  jr      START1          ;program execution unconditionally
                                ;begins at this location after reset
                                ;and power up.
        .ASCII  'REL 0 7/17/86' ;jump around optional ascii string
                                ;containing release info, copyright, etc.
START1: di                      ;begin
        sb0                     ;select register bank 0
        ld      EMT,#00000000B  ;external memory timing=no wait input, normal
                                ;memory timing, no wait states, stack internal,
                                ;and DMA internal
        ld      P0,#00H         ;address begins at 0000h, set upper byte
        ld      P0M,#11111111B  ;select all lines as address
        ld      PM,#00110000B   ;enable port 0 as upper 8 bits address
        ld      H1C,#00000000B  ;handshake not enabled port 0
;
;port 1 is defined in romless part as address/data.  it is not necessary
;here to initialize that port
;
        ld      P2,#00100000B   ;port 2 outputs low, except strobe bit
        ld      P3,#00H         ;port 3 outputs low
        ld      P2AM,#10001010B ;p31,20,21 as output,p30 input
                                ;it is necessary here to configure p30 as input
                                ;for the receive data, and p31 as output for
                                ;transmit data for UART
        ld      P2BM,#10100010B ;p32,33,22 as output, 23 as input
        ld      P2CM,#10101001B ;p34,35,25 as output, 24 as input, interrupt en
        ld      P2DM,#10101010B ;p36,37,26,27 as output
;
        ld      P4,#00000000B   ;clear port 4 register
        ld      P4D,#00000000B  ;set all bits of P4 as outputs
        ld      P4OD,#00000000B ;active push/pull
        ld      H0C,#11110001B  ;handshake enable for port 4, 16 clock pulse
;
;basic Super 8 I/O is initialized, now internal registers
;
        ld      RP0,#0C0H       ;set working register low to lower 8 bytes
        ld      RP1,#0C8H       ;set working register high to upper 8 bytes
        ld      SPL,#0FFH       ;set stack pointer to start at top of set two
                                ;note here that only lower 8 bits are used
                                ;for stack pointer.  location 0FFH is wasted
                                ;as stack operation.  SPH is general purpose
                                ;storage.
;
;now clear the internal memory and stack area
```

```
;
        ld      SPH,#0FFH       ;point to top of general purpose register
ZERO:   clr     @SPH            ;zero it
        dec     SPH
        jr      nz,ZERO         ;do it until register set is all cleared
        clr     @SPH            ;zero last register
;
;now everything except working registers is cleared
;
;cpu and memory now initialized, set up timer for real time clock
;
        ld      SYM,#00000000B  ;disable fast interrupt response
        ld      IPR,#10111111B  ;interrupt priority
                                ;IRQ6>IRQ7>IRQ5>IRQ4>IRQ3>IRQ2>IRQ1>IRQ0
        ld      IMR,#01010000B  ;rx interrupts, acknowledge strobe
;
;timer is set, now lets initialize the UART for polled operation
;
        sb1                     ;bank 1
        ld      UMA,#01110000B
                                ;time constant  = (12,000,000/4/16/9600/2)-1=
                                ;8.76 rounded to 9.
                                ;note that a 12 Mhz does not make a very
                                ;accurate baud rate source.  error is large
        ld      UBGH,#^HB(00009)        ;high byte of time constant
        ld      UBGL,#^LB(00009)        ;low byte of time constant
        ld      UMB,#00011110B  ;p21=p21data,auto-echo is off, transmit and
                                ;receive clock is baud rate generator output,
                                ;baud rate generator input is system clock / 2,
                                ;baud rate generator is enabled, loopback
                                ;is disabled
        sb0                     ;select bank 0
        ld      UTC,#10001000B  ;select p31 as transmit data out, 1 stop bit
                                ;and transmit enable
        ld      UIE,#00000001B  ;receive interrupts, no DMA
        ld      URC,#00000010B  ;enable receiver
;
;UART is initialized, reset acknowledge bit and begin
;
        bitr    ACKB,#ACKBIT    ;reset acknowldege bit if set
        ld      P2BIP,#00000001B        ;reset interrupt input flip-flop
        ei                      ;enable interrupts
WAIT:   ldw     MPTR,#MSG       ;point to message
        call    SENDM           ;send the message
        ld      INPNT,#0        ;set input pointer to register 0
        ld      OUTPNT,#0       ;set output pointer to register 0
WAIT1:  call    SNDBUF          ;send any characters in buffer
        jr      WAIT1           ;loop back
;
;
;
SENDM:  tm      P2,#00001000B   ;printer busy
        jr      nz,SENDM        ;wait for printer unbusy
        btjrt   SENDM,ACKB,#ACKBIT      ;see if the acknowledge has occurred
                                        ;from possible last byte
        bits    ACKB,#ACKBIT    ;set acknowledge bit before writing to output
        ldci    r0,@MPTR        ;get the character
        ld      P4,r0           ;send to printer
        nop                     ;allow 18 clocks for strobe
        nop
        nop
        cp      r0,#'$'         ;last character?
        jr      ne,SENDM        ;loop back for next
        ret
;
;
SNDBUF: cp      INPNT,OUTPNT    ;compare inpointer to outpointer
        jr      ne,SC1          ;send character if any to send
        ret                     ;otherwise return
SC1:    tm      P2,#00001000B   ;printer busy?
        jr      nz,SC1          ;if so, wait until it is not busy
        btjrt   SC1,ACKB,#ACKBIT        ;see if acknowledge has occurred
                                ;from possible last byte

        di
        bits    ACKB,#ACKBIT    ;set acknowledge bit before writing to output
        ld      P4,@OUTPNT      ;send the character
        tm      P2,#00000001B
```

```
        jr      z,HON           ;if host is on
        ld      r0,OUTPNT       ;get the output pointer
        xor     r0,#10000000B   ;add 128 to it
        cp      INPNT,r0        ;turn host back on when 128 bytes left in buf
        jr      ne,HON          ;otherwise keep sending
        and     P2,#11111110B   ;host back on
HON:    nop
        inc     OUTPNT          ;bump pointer
        ei                      ;to make sure pointer not changed
        ret
;
;send character in r0
SENDC:  tm      UTC,#00000010B  ;transmit buffer empty yet
        jr      z,SENDC         ;if not, wait until it is
        ld      UIO,r0          ;load the character into the transmitter
        ret
;receive character available interrupt
RXDATI: ld      r0,UIO          ;get input from console
        and     r0,#7fH         ;remove upper parity bit
        call    SENDC           ;echo to console
        ld      @INPNT,r0       ;save the character
        inc     INPNT           ;bump input pointer
        cp      INPNT,OUTPNT    ;has the input made a complete loop?
        jr      ne,RXIT
;
;receive character buffer full, stop sending device
;
        or      P2,#00000001B   ;raise DTR to stop host sending
INTRET:
RXIT:   iret
;
ACKSTB: tm      P2,#00010000B   ;is line low or high now
        bitr    ACKB,#ACKBIT    ;reset acknowledge bit in register
;
ACKS1:  tm      P2,#00010000B           ;test ack bit
        jr      z,ACKS1         ;wait here till end of strobe
        ld      P2BIP,#00000001B        ;reset p24 interrupt pending register
        iret                    ;and return
;
MSG:    .ASCII  CR,LF,'Super8 serial/parallel test program.',CR,LF
        .ASCII  'Second line test data',CR,LF,'$'

        .END
```

```
;
        .TITLE  Sample Zilog Super 8 Serial to Parallel Converter with XON/XOFF
;
;
;============================================================
;=      TITLE:          SERPAR1.S                           =
;=      DATE:           JULY 17, 1986                       =
;=      PURPOSE:        TO DEMONSTRATE INTERRUPT            =
;=                      DRIVEN SERIAL PORT IN A             =
;=                      REALISTIC APPLICATION.              =
;=                      THIS APPLICATION RECEIVES           =
;=                      SIMPLE SERIAL DATA A SENDS IT       =
;=                      OUT THE PARALLEL PORT TO A          =
;=                      PRINTER. FLOW CONTROL IS BY         =
;=                      XON/XOFF COMMANDS ON THE BACK       =
;=                      CHANNEL TO THE HOST                 =
;=      ASSEMBLER:      ZILOG ASMS8 ASSEMBLER               =
;=      PROGRAMMER:     CHARLES M. LINK, II                 =
;============================================================
;
;
;
        .PAGE   55      ;set maximum page size to 55 lines
;************************************************************
;*                                                          *
;*                    GENERAL EQUATES                       *
;*                                                          *
;************************************************************
;
CR:     .equ    0dH     ;carriage return
LF:     .equ    0aH     ;line feed
```

```
XON:    .equ    11H     ;control-Q or DC1
XOFF:   .equ    13H     ;control-S or DC3
;
;
;*************************************************************
;*                                                           *
;*                  REGISTER EQUATE TABLE                    *
;*                                                           *
;*************************************************************
;
;working register equates
INPNT:  .equ    R3      ;input character pointer
OUTPNT: .equ    R4      ;output character pointer
MPTR:   .equ    RR6     ;message pointer for external memory
ACKB:   .equ    R5      ;byte containing acknowledge bit
ACKBIT: .equ    0       ;bit set = no acknowledge yet
                        ;bit clear = not waiting on acknowledge
XBIT:   .equ    1       ;XOFF send to host
;
;*************************************************************
;*                                                           *
;*                  INTERRUPT VECTOR TABLE                   *
;*                                                           *
;*************************************************************
;
INTR0:  .WORD   INTRET          ;this area should always be defined
INTR1:  .WORD   INTRET          ;as it reserves the lower 32 bytes
INTR2:  .WORD   INTRET          ;for the interrupt table.  the name
INTR3:  .WORD   INTRET          ;of the subroutine for each particular
INTR4:  .WORD   INTRET          ;interrupt service would normally be
INTR5:  .WORD   INTRET          ;named here.
INTR6:  .WORD   INTRET
INTR7:  .WORD   INTRET
INTR8:  .WORD   INTRET
INTR9:  .WORD   INTRET
INTR10: .WORD   RXDATI          ;receive data interrupt
INTR11: .WORD   INTRET
INTR12: .WORD   INTRET
INTR13: .WORD   INTRET
INTR14: .WORD   ACKSTB          ;acknowledge strobe interrupt
INTR15: .WORD   INTRET
;
;*************************************************************
;*                                                           *
;*              START OF PROGRAM EXECUTION                   *
;*                                                           *
;*************************************************************
;
START:  di                      ;for emulation if nothing else
        jr      START1          ;program execution unconditionally
                                ;begins at this location after reset
                                ;and power up.
        .ASCII  'REL 0 7/17/86' ;jump around optional ascii string
                                ;containing release info, copyright, etc.
START1: sb0                     ;select register bank 0
        ld      EMT,#00000000B  ;external memory timing=no wait input, normal
                                ;memory timing, no wait states, stack internal,
                                ;and DMA internal
        ld      P0,#00H         ;address begins at 0000h, set upper byte
        ld      P0M,#11111111B  ;select all lines as address
        ld      PM,#00110000B   ;enable port 0 as upper 8 bits address
        ld      H1C,#00000000B  ;handshake not enabled port 0
;
;port 1 is defined in romless part as address/data.  it is not necessary
;here to initialize that port
;
        ld      P2,#00100000B   ;port 2 outputs low, except strobe bit
        ld      P3,#00H         ;port 3 outputs low
        ld      P2AM,#10001010B ;p31,20,21 as output,p30 input
                                ;it is necessary here to configure p30 as input
                                ;for the receive data, and p31 as output for
                                ;transmit data for UART
        ld      P2BM,#10100010B ;p32,33,22 as output, 23 as input
        ld      P2CM,#10101001B ;p34,35,25 as output, 24 as input, interrupt en
        ld      P2DM,#10101010B ;p36,37,26,27 as output
;
        ld      P4,#00000000B   ;clear port 4 register
        ld      P4D,#00000000B  ;set all bits of P4 as outputs
```

```
        ld      P4OD,#00000000B ;active push/pull
        ld      H0C,#11110001B  ;handshake enable for port 4, 16 clock pulse
;
;basic Super 8 I/O is initialized, now internal registers
;
        ld      RP0,#0C0H       ;set working register low to lower 8 bytes
        ld      RP1,#0C8H       ;set working register high to upper 8 bytes
        ld      SPL,#0FFH       ;set stack pointer to start at top of set two
                                ;note here that only lower 8 bits are used
                                ;for stack pointer.  location 0FFH is wasted
                                ;as stack operation.  SPH is general purpose
                                ;storage.
;
;now clear the internal memory and stack area
;
        ld      SPH,#0FFH       ;point to top of general purpose register
ZERO:   clr     @SPH            ;zero it
        dec     SPH
        jr      nz,ZERO         ;do it until register set is all cleared
        clr     @SPH            ;zero last register
;
;now everything except working registers is cleared
;
;cpu and memory now initialized, set up timer for real time clock
;
        ld      SYM,#00000000B  ;disable fast interrupt response
        ld      IPR,#10111111B  ;interrupt priority
                                ;IRQ6>IRQ7>IRQ5>IRQ4>IRQ3>IRQ2>IRQ1>IRQ0
        ld      IMR,#01010000B  ;rx interrupts, acknowledge strobe
;
;timer is set, now lets initialize the UART for polled operation
;
        sb1                     ;bank 1
        ld      UMA,#01110000B
                                ;time constant  = (12,000,000/4/16/9600/2)-1=
                                ;8.76 rounded to 9.
                                ;note that a 12 Mhz does not make a very
                                ;accurate baud rate source.  error is large
        ld      UBGH,#^HB(00009)        ;high byte of time constant
        ld      UBGL,#^LB(00009)        ;low byte of time constant
        ld      UMB,#00011110B  ;p21=p21data,auto-echo is off, transmit and
                                ;receive clock is baud rate generator output,
                                ;baud rate generator input is system clock / 2,
                                ;baud rate generator is enabled, loopback
                                ;is disabled
        sb0                     ;select bank 0
        ld      UTC,#10001000B  ;select p31 as transmit data out, 1 stop bit
                                ;and transmit enable
        ld      UIE,#00000001B  ;receive interrupts, no DMA
        ld      URC,#00000010B  ;enable receiver
;
;UART is initialized, reset acknowledge bit and begin
;
        bitr    ACKB,#ACKBIT    ;reset acknowldege bit if set
        bitr    ACKB,#XBIT      ;reset XON/XOFF bit
        ld      P2BIP,#00000001B        ;reset interrupt input flip-flop
        ei                      ;enable interrupts
WAIT:   ldw     MPTR,#MSG       ;point to message
        call    SENDM           ;send the message
        ld      INPNT,#0        ;set input pointer to register 0
        ld      OUTPNT,#0       ;set output pointer to register 0
WAIT1:  call    SNDBUF          ;send any characters in buffer
        jr      WAIT1           ;loop back
;
;
;
SENDM:  tm      P2,#00001000B   ;printer busy
        jr      nz,SENDM        ;wait for printer unbusy
        btjrt   SENDM,ACKB,#ACKBIT      ;see if the acknowledge has occurred
                                        ;from possible last byte
        bits    ACKB,#ACKBIT    ;set acknowledge bit before writing to output
        ldci    r0,@MPTR        ;get the character
        ld      P4,r0           ;send to printer
        nop                     ;allow 18 clocks for strobe
        nop
        nop
        cp      r0,#'$'         ;last character?
        jr      ne,SENDM        ;loop back for next
        ret
```

```
;
;timer is initialized, now lets enable interrupts and wait
        ldw     CINCR,#1        ;start column at beginning of sine table
        ldw     RINCR,#1        ;start row at beginning of sine table
;
;this example loads the tones for digit '1'
;user software would, of course have to manipulate these registers for
;proper tone control
;
        ldw     CFINCR,#CFREQI  ;load column frequency increment
        ldw     RFINCR,#RFREQI  ;load row frequency increment
        ldw     POINT,#SINTAB   ;pointer points to sine table
        ld      CVAL,#080H      ;initial value to prevent glitch at start
        ei                      ;enable interrupts
WAIT:   nop
        nop
        nop
        nop
        jr      WAIT            ;loop back
;
;Timer interrupt.  Occurs SAMPLE times per second
;interrupt outputs value to DAC-08 and then determines value for next
;interrupt.  This assures no bit jitter.
;
TIMER0: ld      p4,CVAL         ;write new value to DAC-08
        rcf                     ;clear carry flag
        add     CINCRL,CFINCL   ;find next position in sine table
        adc     CINCRH,CFINCH   ;by adding frequency offset to last position
        ld      POINTL,CINCRH   ;set new pointer into sine table
        ldc     CVAL,@POINT     ;get value from sine table
        add     RINCRL,RFINCL   ;find next position in sine table
        adc     RINCRH,RFINCH   ;by adding frequencty offset to last position
        ld      POINTL,RINCRH   ;set new pointer into sine table
        ldc     RVAL,@POINT     ;get second value from sine table
        add     CVAL,RVAL       ;form a complex waveform from two sine values
        or      COCT,#00000010B ;reset end of count interrupt
INTRET: iret                    ;and return from interrupt
;
;*************************************************************
;*                                                           *
;*                      SINE WAVE LOOKUP                     *
;*                                                           *
;*************************************************************
;
;sine table for DTMF generation using DAC-08.  Table based upon
;case of waveform consisting of two sine waves summed to provide a single
;complex waveform with minumum amplititude = 0 volts and maximum
;amplititude = 5 volts.  DAC-08 input for 0 volts = 00H
;5 volts = 0FFH.  Both waves must total no more than 0FFH, therefore
;maximum for one wave must be 1/2 5 volts or 080H.
;Table generated using following BASICA program,
;then typed into program.
;
;       10 CLS                  ;clear screen
;       20 PI=3.141593          ;define PI
;       30 FOR I=0 TO 255       ;256 total values
;       40 C=360/256            ;define basic interval value
;       50 D=C*I                ;value from zero on sine wave
;       60 E=D*PI/180
;       70 F=SIN(E)             ;figure sine for interval from 0
;       80 G=F*63               ;sine range should be from -63 to 63
;       90 H=64+G               ;make result from 0 to 127
;       100 J=CINT(H)           ;round to nearest integer
;       110 A$=HEX$(J)          ;convert to hex
;       120 PRINT A$            ;on screen
;       130 LPRINT A$           ;on printer
;       140 NEXT                ;do next inverval
;       150 END
;
;*note-remove comments, BASICA will not accept ; as comment delimiter
;
;
SINTAB: .ORG    0400H           ;begin sine table on even byte boundary
        .byte   040H,042H,043H,045H,046H,048H,049H,04BH,04CH,04EH,04FH,051H
        .byte   052H,054H,055H,057H,058H,05AH,05BH,05CH,05EH,05FH,060H,062H
        .byte   063H,064H,066H,067H,068H,069H,06AH,06BH,06DH,06EH,06FH,070H
        .byte   071H,072H,073H,074H,074H,075H,076H,077H,078H,078H,079H,07AH
        .byte   07AH,07BH,07BH,07CH,07CH,07DH,07DH,07DH,07EH,07EH,07EH,07FH
```

```
;
;
SNDBUF: cp      INPNT,OUTPNT    ;compare inpointer to outpointer
        jr      ne,SC1          ;send character if any to send
        ret                     ;otherwise return
SC1:    tm      P2,#00001000B   ;printer busy?
        jr      nz,SC1          ;if so, wait until it is not busy
        btjrt   SC1,ACKB,#ACKBIT        ;see if acknowledge has occurred
                                ;from possible last byte

        di
        bits    ACKB,#ACKBIT    ;set acknowledge bit before writing to output
        ld      P4,@OUTPNT      ;send the character
        btjrf   HON,ACKB,#XBIT  ;host is still sending
        ld      r0,OUTPNT       ;get the output pointer
        xor     r0,#10000000B   ;add 128 to it
        cp      INPNT,r0        ;turn host back on when 128 bytes left in buf
        jr      ne,HON          ;otherwise keep sending
        ld      r0,XON          ;send XON to host to start it sending again
        call    SENDC
        bitr    ACKB,#XBIT      ;reset XOFF bit
HON:    nop
        inc     OUTPNT          ;bump pointer
        ei                      ;to make sure pointer not changed
        ret
;
;send character in r0
SENDC:  tm      UTC,#00000010B  ;transmit buffer empty yet
        jr      z,SENDC         ;if not, wait until it is
        ld      UIO,r0          ;load the character into the transmitter
        ret
;receive character available interrupt
RXDATI: ld      r0,UIO          ;get input from console
        and     r0,#7fH         ;remove upper parity bit
        call    SENDC           ;echo to console
        ld      @INPNT,r0       ;save the character
        inc     INPNT           ;bump input pointer
        ld      r0,INPNT        ;get the input pointer
        add     r0,#5           ;allow 5 characters after XOFF
        cp      r0,OUTPNT       ;has the input made a complete loop?
        jr      ne,RXIT
;
;receive character buffer full, stop sending device
;
        ld      r0,#XOFF        ;send XOFF to host
        call    SENDC           ;send it
        bits    ACKB,#XBIT      ;set the XOFF bit
INTRET:
RXIT:   iret
;
ACKSTB: tm      P2,#00010000B   ;is line low or high now
        bitr    ACKB,#ACKBIT    ;reset acknowledge bit in register
;
ACKS1:  tm      P2,#00010000B           ;test ack bit
        jr      z,ACKS1         ;wait here till end of strobe
        ld      P2BIP,#00000001B        ;reset p24 interrupt pending register
        iret                    ;and return
;
MSG:    .ASCII  CR,LF,'Super8 serial/parallel test program.',CR,LF
        .ASCII  'Second line test data',CR,LF,'$'

        .END
```

ADDITIONAL INFORMATION

ZiLOG

Support Products Summary

SUPPORT PRODUCTS SUMMARY

KIT-TO-PART CROSS REFERENCE MATRIX

APPLICATIONS BOARDS

Tool Box Family

EMULATORS

ICE BOX™ Family

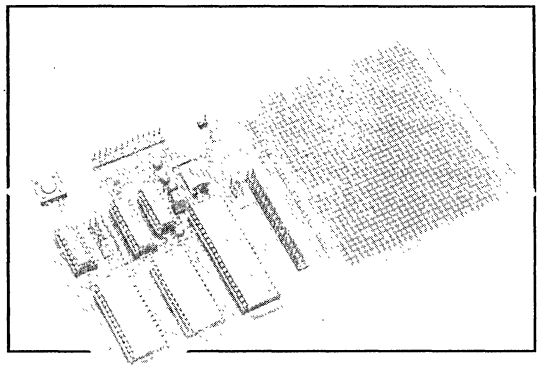
C Series | S Series

Part Number	Z08600ZCO	Z08600ZDP	Z08602ZCO	Z08602ZDP	Z86C0800ZCO	Z86C0800ZDP	Z86C0800ZEM	Z86E0800ZPR	Z86C1200ZDP	Z86C1900ZCO	Z86C1900ZEM	Z86E2100ZDP	Z86C2700ZCO	Z86C2700ZEM	Z86E3000ZDP	Z86E4000ZDP	Z08800ZCO	Z86C1200ZEM	Z86C5000ZEM	Z86C9300ZEM	Z86C1200ZPD	Z86C5000ZPD	Z86C9300ZPD
Z08600	●	●																					
Z08601	●																						
Z08602			●	●																			
Z08604									●														
Z08611	●																						
Z86C06																		●			●		
Z86C08					●	●	●											●			●		
Z86E08					●	●	●	●										●			●		
Z86C09									●	●								●			●		
Z86C12									●														
Z86C19									●	●								●			●		
Z86C21	●																	●			●		
Z86E21										●								●			●		
Z86C27													●	●									
Z86E30															●			●			●		
Z86E40																●		●			●		
Z86C90										●								●			●		
Z86C91	●																	●			●		
Z86C93	●																		●				●
Z86C97													●	●									
Z08800																	●						

*Excluding SPI

Zilog Z86C0800ZDP Required

Z0860000ZCO
PRODUCT SPECIFICATION



SUPPORTED DEVICES
Z86C91, Z86C21, Z8600, Z8601, Z8611

DESCRIPTION
The Z8 Development Kit can be used for several purposes. As an evaluation tool, one can learn the Z8 instruction set plus the manipulation of the Z8's interrupt vectors and register set. Secondly, the Z8 Development Kit is designed to aid the user in constructing specific applications using the Z8 microcontroller.

SPECIFICATIONS
Power Requirements
+5 Vdc @ 50 mA

Dimensions
Width: 4.0 in. (10.2 cm)
Length: 8.0 in. (20.3 cm)

Serial Interface
RS-232C @ 9600 baud

KIT CONTENTS
Z8 Development Board
CMOS Z86C91 MPU
12 MHz Crystal
(32K)/8K x 8 EPROM
(32K)/8K x 8 STATIC RAM
RS-232C PC Interface
Z86C91 Expansion Header

Cables
25-Pin RS-232 Cable

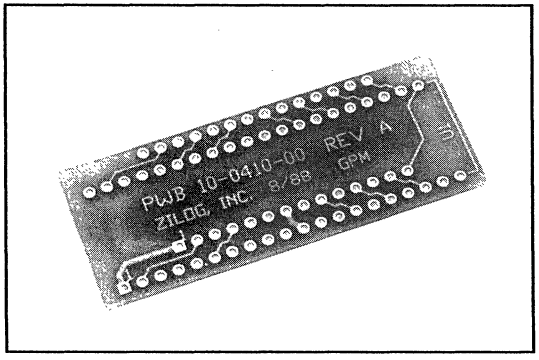
Software (IBM®*-PC Platform)
Z8/Super8 Assembler and Utilities
Host Communication Package
Monitor Instructions
Tutorial
Sample Z86C91 Application Software

Documentation
Z8 Family Data Book
Z8 Cross Assembler User Guide
MOBJ Link/Loader User Guide

ORDERING INFORMATION
Part No: Z0860000ZCO

* IBM is a registered trademark of International Business Machines Corp.

Z0860000ZDP
PRODUCT SPECIFICATION



SUPPORTED DEVICES
Z8600

DESCRIPTION
The Z8600 Adapter Kit allows a standard Z8 emulator to emulate a Z8600 microcontroller by converting a 40-pin Z8 pinout to a 28-pin Z8 pinout.

SPECIFICATIONS
Power Requirements
Not Applicable

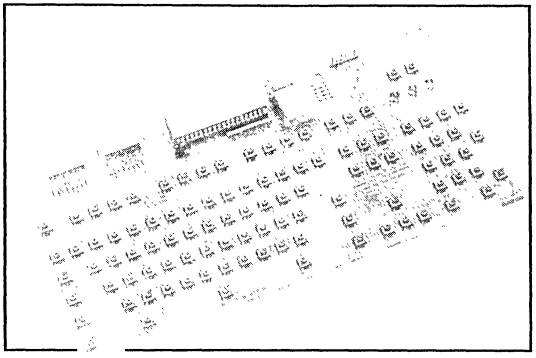
Dimensions
Width: 0.9 in. (2.3 cm)
Length: 2.2 in. (5.4 cm)

KIT CONTENTS
Z8600 Adapter Board
PC Board

ORDERING INFORMATION
Part No: Z0860000ZDP

Z0860200ZCO

PRODUCT SPECIFICATION



SUPPORTED DEVICES

Z08602

DESCRIPTION

The kit contains an assembled circuit board, Z08602 with keyboard, ROM-code, and documentation to help the user become familiar with the features of the Z08602 keyboard controller.

The Z08602 microcontroller is designed into a 101/102 PC keyboard circuit to control all scan codes, line status modes, scan timing and communication between the keyboard and PC.

SPECIFICATIONS

Power Requirements

+5 Vdc @ .2 A (Supplied By PC)

Dimensions

Width: 4.6 in. (11.7 cm)
Length: 9.3 in. (23.6 cm)

KIT CONTENTS

Z08602 101/102 Keyboard

NMOS Z08602 MPU
2 MHz Crystal
101/102 Keyboard Option
3 LEDs
Two 8-Position Dip Switches
6-Pin Communication Header

Software (IBM-PC Platform)

Z8/Z80/Z8000 Cross Assembler
MOBJ Link/Loader
Application Source Code

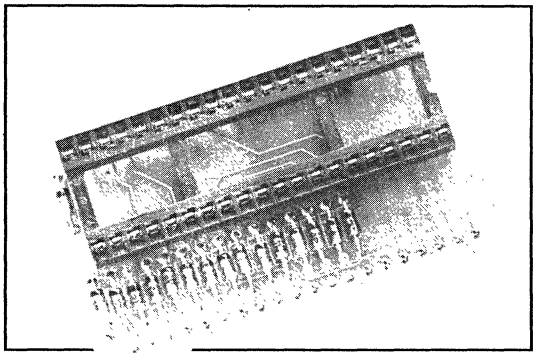
Documentation

Z08602 Application Kit User Guide
Z8 Family Data Book
Z8 Cross Assembler User Guide
MOBJ Link/Loader User Guide

ORDERING INFORMATION

Part No: Z0860200ZCO

Z0860200ZDP
PRODUCT SPECIFICATION



SUPPORTED DEVICES
Z08602

DESCRIPTION
The Z08602 adapter board is a tool used to adapt a standard Z8601 type device or emulation system to a Z8602 target socket.

SPECIFICATIONS
Power Requirements
Not Applicable

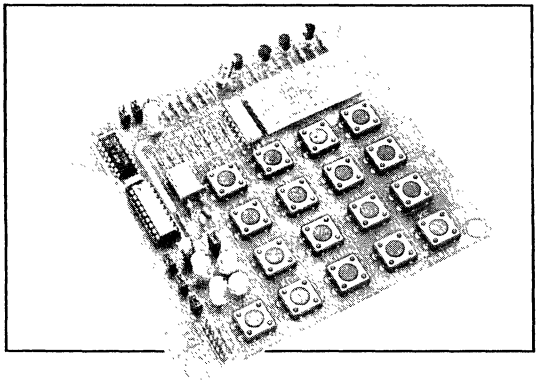
Dimensions
Width: 1.3 in. (3.3 cm)
Length: 2.3 in. (5.8 cm)

KIT CONTENTS
Z08602 Adapter Board
40-pin Z08601/Z08611 MPU Socket
40-pin Z08602 Connecter

Documentation
Z08602 Adapter Kit User Guide

ORDERING INFORMATION
Part No: Z0860200ZDP

Z86C0800ZCO
PRODUCT SPECIFICATION



SUPPORTED DEVICES
Z86C08

DESCRIPTION
The kit contains an assembled circuit board, software and documentation to help the user become familiar with the features of the Z86C08 microcontroller.

The Applications Board is used to demonstrate the advantages and versatility of the 18-pin Z8 device. Included is simple hardware and software that demonstrates the implementation of WDT, HALT, and STOP MODE, low cost D to A and A to D conversion techniques.

SPECIFICATIONS
Power Requirements
+5 Vdc @ 50 mA

Dimensions
Width: 4.4 in. (11.2 cm)
Length: 4.8 in. (12.2 cm)

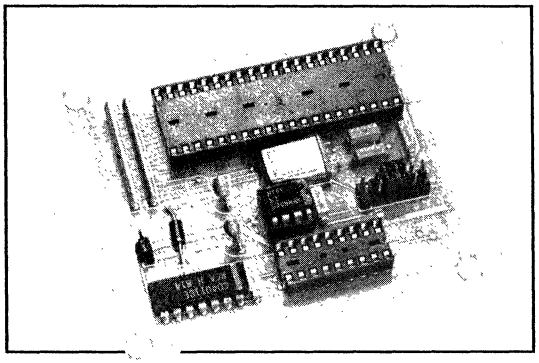
KIT CONTENTS
Z86C08 Application Board
CMOS Z86C08 MPU
4 MHz Crystal
Four 7-segment LED Displays
17 Key Keypad

Software (IBM-PC Platform)
Application Source Code
Z8/Z80/Z8000 Cross Assembler
MOBJ Link/Loader

Documentation
Z8 Family Data Book
Z8 Cross Assembler User Guide
MOBJ Link/Loader User Guide
Z86C08 Application Kit User Guide

ORDERING INFORMATION
Part No: Z86C0800ZCO

Z86C0800ZDP
PRODUCT SPECIFICATION



SUPPORTED DEVICES
Z86C08

DESCRIPTION
The Z86C08 adapter board converts the Z8 40-pin pinout to a Z8 18-pin part. This adapter board allows a standard Z8 emulation device to emulate the Z86C08. The Z86C08 Adapter Board is placed between the Z8 emulator and the user's target socket. The board does not emulate the watchdog timer function.

SPECIFICATIONS
Power Requirements
Not Applicable

Dimensions
Width: 2.5 in. (6.4 cm)
Length: 2.9 in. (7.4 cm)

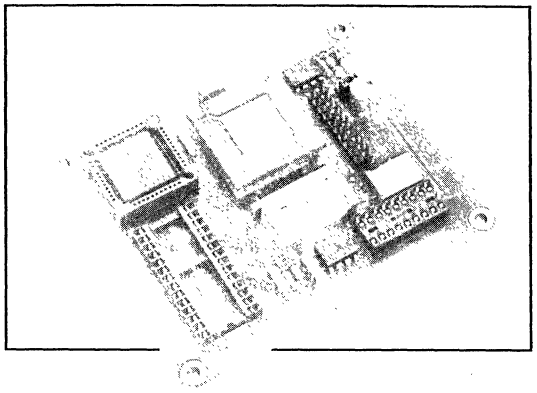
KIT CONTENTS
Z86C08 Adapter Board
40-pin Z8 MPU Socket
18-pin Z86C08 Socket
12 MHz Crystal

Cables
18-Pin Z86C08 Emulation Cable

Documentation
Z86C08 Adapter Kit User Guide

ORDERING INFORMATION
Part No: Z86C0800ZDP

Z86C0800ZEM
PRODUCT SPECIFICATION



SUPPORTED DEVICES
Z86C08

DESCRIPTION
The Z86C08 Emulation Board allows the user to plug a programmed EPROM into the board to verify operation of code before submitting for Mask ROM. The board will emulate the watchdog timer function, but does not allow the development of Z8 code. To fully emulate the Z86C08, the code must be developed using a standard Z8 emulator along with the Z86C08 Adapter Board. The code can then be verified in the user's application with the Z86C08 Emulator Board.

SPECIFICATIONS
Power Requirements
+5 Vdc @ 100 mA from target board

Dimensions
Width: 3.0 in. (7.6 cm)
Length: 3.6 in. (9.1 cm)

KIT CONTENTS
Z86C08 Emulation Board
CMOS Z86C12 ICE
4 MHz Crystal
EP900LC-3 EPLD
(32K)/8K x 8 EPROM Socket
Z86E08 Socket
8-Position Dip Switch

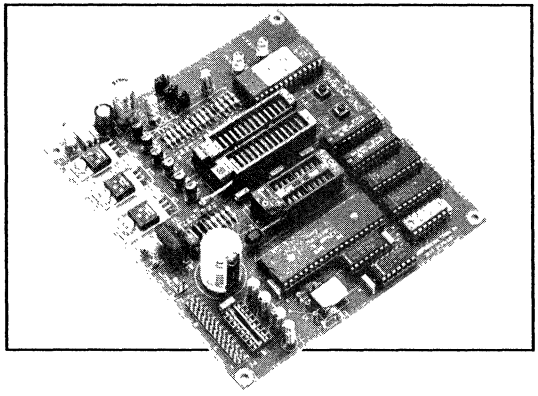
Cables
18-Pin Z86C19 Emulation Cable

Software (IBM-PC Platform)
Z8/Z80/Z8000 Cross Assembler
MOBJ Link/Loader

Documentation
Z8 Family Data Book
Z86C08ZEM Kit User Guide
Z8 Cross Assembler User Guide
MOBJ Link/Loader User Guide

ORDERING INFORMATION
Part No: Z86C0800ZEM

Z86E0800ZPR
PRODUCT SPECIFICATION



SUPPORTED DEVICES
Z86E08

DESCRIPTION
The Kit contains an assembled circuit board, software, and documentation to program the Z86E08 OTP.
The Z86E08 is an OTP version of the Z86C08 single chip microcomputer housed in an 18-pin DIP. It offers the same architecture and all the features of the Z86C08. The Z86E08 also offers "LOW NOISE" and "ROM PROTECT" options, which can be programmed by the programmer.

SPECIFICATIONS
Power Requirements
+15 Vdc @ 1 A
Or 12-15 Vac @ 1 A

Dimensions
Width: 4.9 in. (12.4 cm)
Length: 5.4 in. (13.7 cm)

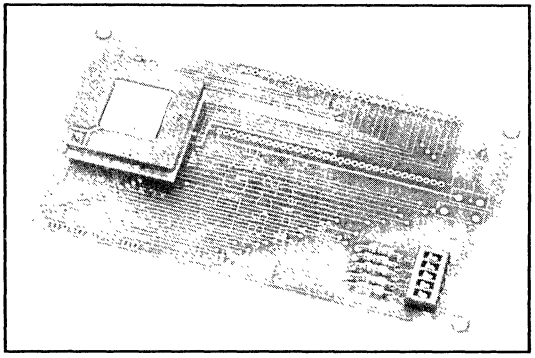
KIT CONTENTS
Z86E08 Programmer Board
CMOS Z86C91 MPU
7.3728 MHz Crystal
8K x 8 EPROM
(32K)/8K x 8 ZIF Socket (for user EPROM)
Z86E08 ZIF Socket
Two 7805 Voltage Regulators
7812 Voltage Regulator
Bridge Rectifier
2 LEDs
2 Key Switches

Software (IBM-PC Platform)
Programming source code

Documentation
Z86E08 Product Specification
Z86E08 Kit User Guide

ORDERING INFORMATION
Part No: Z86E0800ZPR

Z86C1200ZDP
PRODUCT SPECIFICATION



SUPPORTED DEVICES
Z86C12

DESCRIPTION
The Z86C12 adapter board is a simple adapter which converts the 64-pin footprint of the Zilog Z8612 ICE chip to the 84-pin PGA configuration of the CMOS Z86C12 ICE chip.

The Z86C12 Adapter Kit allows a standard Z8 emulator to emulate Z8 CMOS microcontrollers.

SPECIFICATIONS
Power Requirements
Not Applicable

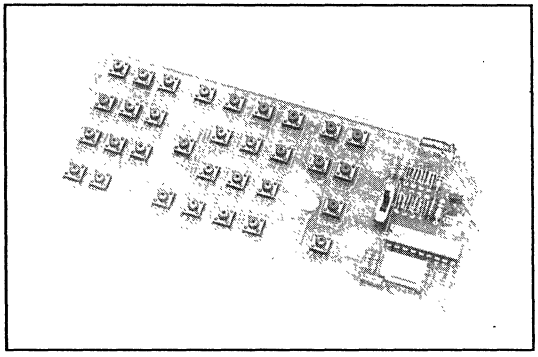
Dimensions
Width: 2.4 in. (6.1 cm)
Length: 4.3 in. (10.9 cm)

KIT CONTENTS
Z86C12 Adapter Board
CMOS Z86C12 ICE
64-Pin Z8612 Connector
5-Position Dip Switch

Documentation
Z86C12 Adapter Kit User Guide

ORDERING INFORMATION
Part No: Z86C1200ZDP
Price:

Z86C1900ZCO
PRODUCT SPECIFICATION



SUPPORTED DEVICES
Z86C09, Z86C19

DESCRIPTION
The kit contains an assembled circuit board, software and documentation for the Universal I.R. Transmitter Application. The transmitter can be set up to operate most models of remote-controlled TVs, VCRs and Cable TV Decoders even if they are in different brands.

With the set-up feature and the easy operation capability, the Universal I.R. Transmitter can be used to replace several remote controllers. The documentation contains the look up codes of each corresponding brand.

SPECIFICATIONS
Power Requirements
+3 < Vcc < +5 Vdc

Dimensions
Width: 2.3 in. (5.8 cm)
Length: 5.5 in. (14.0 cm)

KIT CONTENTS
Z86C19 Universal I.R. Remote Control Board
CMOS Z86C19 MPU (With Mask ROM)
8 MHz Crystal
29 Key Switches

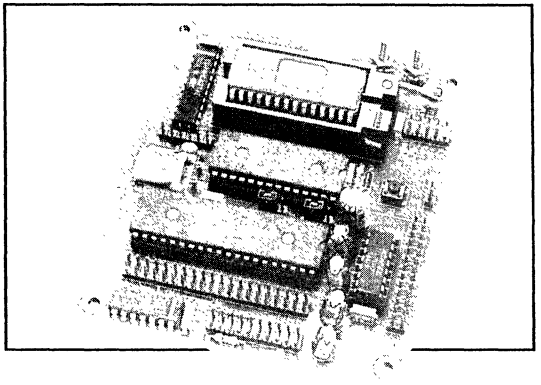
Software (IBM-PC Platform)
I.R. Application Source Code
Z8/Z80/Z8000 Cross Assembler
MOBJ Link/Loader

Documentation
Z8 Family Data Book
Z8 Cross Assembler User Guide
MOBJ Link/Loader User Guide
Z86C09/19 Product Specification
Z86C19 I.R. Application Kit User Guide

ORDERING INFORMATION
Part No: Z86C1900ZCO

Z86C1900ZEM

PRODUCT SPECIFICATION



SUPPORTED DEVICES

Z86C09, Z86C19, Z86C90

DESCRIPTION

The kit contains an assembled circuit board, software and documentation to support software and hardware development for the maskROM Z86C09/19 and ROMless Z86C90 devices.

The supplied cross assembler and link/loader package allows full assembly language programming support. A board resident debug monitor program allows object code to be down-loaded and subsequently debugged.

Code targeted for the Z86C09/19 device may be verified in the target application before submitting to Zilog for production masking.

SPECIFICATIONS

Power Requirements

+5 Vdc @ .5 A

Dimensions

Width: 3.5 in. (8.9 cm)
Length: 4.0 in. (10.2 cm)

Serial Interface

RS-232C @ 9600 baud

KIT CONTENTS

Z86C19 Evaluation Board

CMOS Z86C90 MPU
8 MHz Crystal
(32K)/8K x 8 ZIF Socket (supplied with Debug Monitor EPROM)
(32K)/8K x 8 STATIC RAM
RS-232C PC Interface
Z86C90 Expansion Header
Z86C09/19 Emulation Header

Cables

25-Pin RS-232 Cable
18-Pin Z86C19 Emulation Cable

Software (IBM-PC Platform)

Z8/Z80/Z8000 Cross Assembler
MOBJ Link/Loader
Resident Debug Monitor Source Code
Z86C09 Example Software

Documentation

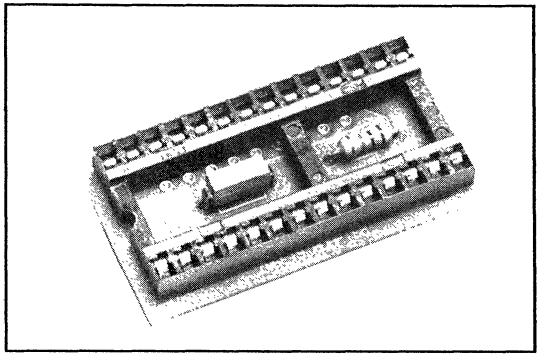
Z8 Family Data Book
Z86C09/19 Product Specification
Z86C30/40/90 Product Specification
Z86C19ZEM Kit User Guide
Z8 Cross Assembler User Guide
MOBJ Link/Loader User Guide

ORDERING INFORMATION

Part No: Z86C19ZEM

Z86E0600ZDP

PRODUCT SPECIFICATION



SUPPORTED DEVICES

Z86E06/09/19

DESCRIPTION

The Z86E06 converter board is a simple adapter which converts the 28-pin footprint of the Zilog Z86E30 OTP chip to the 18-pin DIP configuration of the Z86E06/09/19 OTP chip. The converter supports all the functions of the Z86E06/09/19 except for SPI function.

SPECIFICATIONS

Power Requirements

Not applicable

Dimensions

Width: 0.8 in. (2.0 cm)
Length: 1.5 in. (3.8 cm)

KIT CONTENTS

Z86E06 Converter Board

28-Pin Z86E30 MCU Socket
18-Pin Z86E06/09/19 Connector

Cables

25-Pin RS-232 Cable
18-Pin Z86C19 Emulation Cable

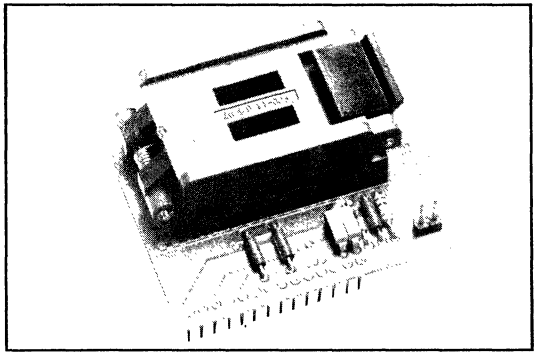
Documentation

Z86E06 OTP Converter Kit User Guide

ORDERING INFORMATION

Part No: Z86E0600ZDP

Z86E2100ZDF
PRODUCT SPECIFICATION



SUPPORTED DEVICES
Z86E21

DESCRIPTION
The Z86E21 QFP OTP Adapter Kit allows the 2764A standard EPROM programmer to program the Z86E21 One Time Programmable microcontroller.

SPECIFICATIONS
Power Requirements
+12.5 Vdc @ .5 A

Dimensions
Width: 1.75 in. (4.4 cm)
Length: 2.20 in. (5.6 cm)

KIT CONTENTS
Z86E21 QFP OTP Program Adapter Board
44-Pin QFP ZIF Socket
28-Pin Connector

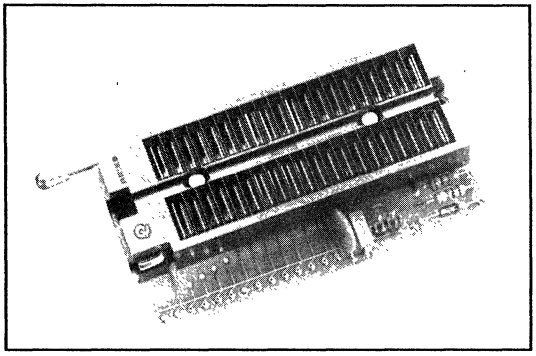
Documentation
Z86E2100ZDF Adapter User Guide

ORDERING INFORMATION

Part No: Z86E2100ZDF

Z86E2100ZDP

PRODUCT SPECIFICATION



SUPPORTED DEVICES

Z86E21

DESCRIPTION

The Z86E21 DIP OTP Adapter Kit allows the 2764A standard EPROM programmer to program the Z86E21 One Time Programmable microcontroller.

SPECIFICATIONS

Power Requirements

+12.5 Vdc @ .5 A

Dimensions

Width: 1.4 in. (3.6 cm)
Length: 2.6 in. (6.6 cm)

KIT CONTENTS

Z86E21 OTP Program Adapter Board

40-Pin DIP ZIF Socket
28-Pin Connector

Documentation

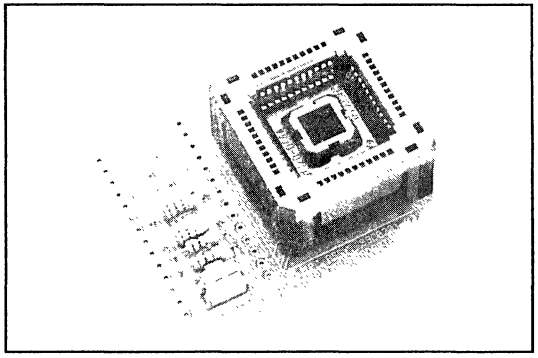
Z86E21ZDP Adapter User Guide

ORDERING INFORMATION

Part No: Z86E2100ZDP

Z86E2100ZDV

PRODUCT SPECIFICATION



SUPPORTED DEVICES

Z86E21

DESCRIPTION

The Z86E21 PLCC OTP Adapter Kit allows the 2764A standard EPROM programmer to program the Z86E21 One Time Programmable microcontroller.

SPECIFICATIONS

Power Requirements

+12.5 Vdc @ .5 A

Dimensions

Width: 1.75 in. (4.4 cm)
Length: 2.20 in. (5.6 cm)

KIT CONTENTS

Z86E21 PLCC OTP Program Adapter Board

44-Pin PLCC ZIF Socket
28-Pin Connector

Documentation

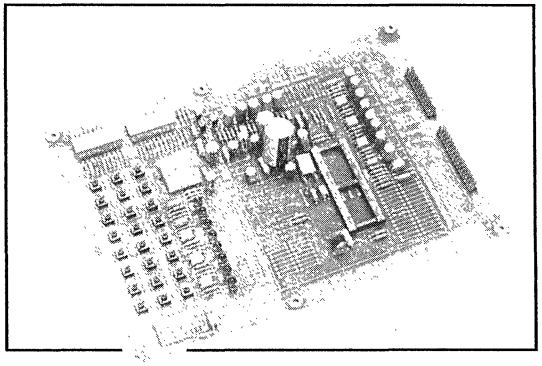
Z86E2100ZDV Adapter User Guide

ORDERING INFORMATION

Part No: Z86E2100ZDV

Z86C2700ZCO

PRODUCT SPECIFICATION



SUPPORTED DEVICES

Z86C27, Z86C97

DESCRIPTION

The Z86C2700ZCO Application Kit is specifically designed for users to evaluate the Hardware and Software of Zilog's Z86C27 Digital Television Controller (DTC). The Z86C2700ZCO Application Kit can be used with an Z86C2700ZEM Emulation Adapter Board to develop application code.

SPECIFICATIONS

Power Requirements

Supplied By TV Set

Dimensions

Width: 6.2 in. (15.7 cm)
Length: 8.6 in. (21.8 cm)

KIT CONTENTS

Z86C27 Application Board

CMOS Z86C27 MPU Socket
4 MHz Crystal
24 Key Multiplexed Keypad
Two 7-segment LED Displays
8 LEDs
13 PWMs
Low Pass Filter Interface
PLL Interface

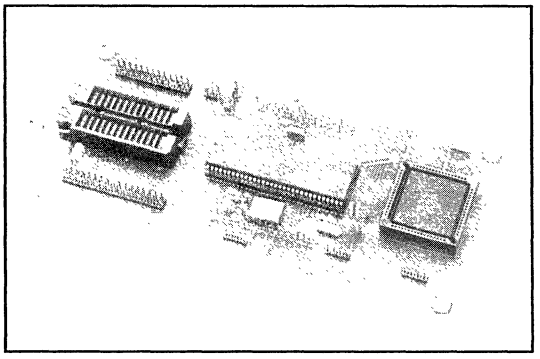
Documentation

Z8 Family Data Book
Z86C27 Application Kit User Guide

ORDERING INFORMATION

Part No: Z86C2700ZCO

Z86C2700ZEM
PRODUCT SPECIFICATION



SUPPORTED DEVICES
Z86C27, Z86C97

DESCRIPTION
The Z86C2700ZEM Emulation Board allows the user to plug a programmed EPROM into the board to test out code or to connect with the Orion ROM emulator to emulate code. The board comes with Z86C97 (ROMLESS of Z86C27) and Altera EP1810JEPLD to emulate I/O Ports.

SPECIFICATIONS
Power Requirements
+5 Vdc @ .1 A

Dimensions
Width: 3.0 in. (7.6 cm)
Length: 6.8 in. (17.3 cm)

KIT CONTENTS
Z86C27 Emulation Board
CMOS Z86C97 MPU
EP1800LC-2 EPLD
(32K)/8K x 8 ZIF Socket
4 MHz Crystal
Z86C97 Adapter Board
Z86C97 Expansion Header
Unilab 8620 or 8420 Analyzer-Emulator Headers

Cables
Unilab 8620 or 8420 Analyzer Cable
Unilab 8620 or 8420 Emulator Cable

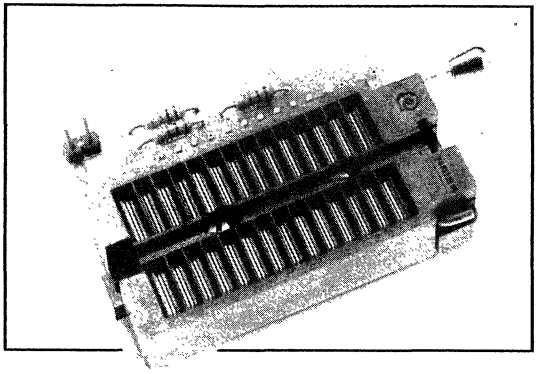
Software (IBM-PC Platform)
Z8/Z80/Z8000 Cross Assembler
MOBJ Link/Loader

Documentation
Z8 Family Data Book
Z8 Cross Assembler User Guide
MOBJ Link/Loader User Guide

ORDERING INFORMATION
Part No: Z86C2700ZEM

Z86E3000ZDP

PRODUCT SPECIFICATION



SUPPORTED DEVICES

Z86E30

DESCRIPTION

The Z86E30 OTP DIP Adapter Kit allows a standard EPROM programmer to program the Z86E30 One-Time-Programmable microcontroller.

SPECIFICATIONS

Power Requirements

+12.5 Vdc @ .5A

Dimensions

Width: 1.45 in. (3.68 cm)
Length: 2.0 in. (5.08 cm)

KIT CONTENTS

Z86E30 OTP Program Adapter Board

28-Pin DIP ZIF Socket
28-Pin Connector

Documentation

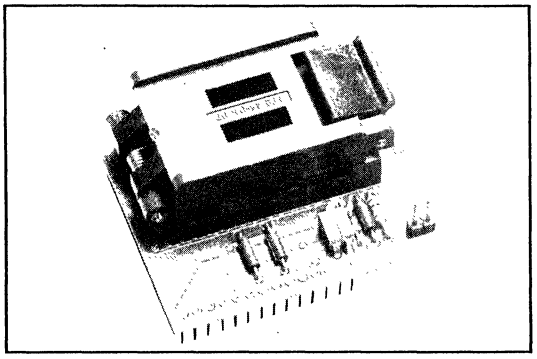
Z86E30ZDP Adapter User Guide

ORDERING INFORMATION

Part No: Z86E3000ZDP

Z86E4000ZDF

PRODUCT SPECIFICATION



SUPPORTED DEVICES

Z86E40

DESCRIPTION

The Z86E40 QFP OTP Adapter Kit allows a standard EPROM programmer to program the Z86E40 One-Time-Programmable microcontroller.

SPECIFICATIONS

Power Requirements

+12.5 Vdc @ .5 A

Dimensions

Width: 1.75 in. (4.4 cm)
Length: 2.20 in. (5.6 cm)

KIT CONTENTS

Z86E40 QFP OTP Program Adapter Board

44-Pin QFP ZIF Socket
28-Pin Connector

Documentation

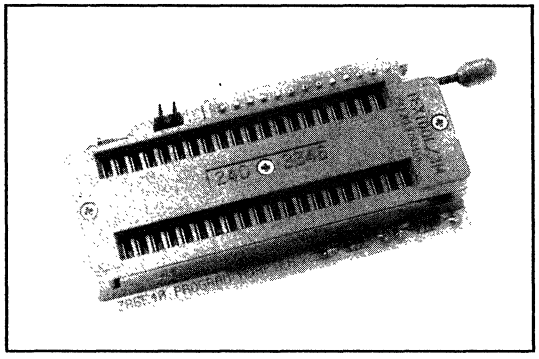
Z86E4000ZDF Adapter User Guide

ORDERING INFORMATION

Part No: Z86E4000ZDF

Z86E4000ZDP

PRODUCT SPECIFICATION



SUPPORTED DEVICES

Z86E40

DESCRIPTION

The Z86E40 OTP DIP Adapter Kit allows a standard EPROM programmer to program the Z86E40 One-Time-Programmable microcontroller.

SPECIFICATIONS

Power Requirements

+12.5 Vdc @ .5 A

Dimensions

Width: 1.4 in. (3.6 cm)
Length: 2.6 in. (6.6 cm)

KIT CONTENTS

Z86E40 OTP Program Adapter Board

40-Pin DIP ZIF Socket
28-Pin Connector

Documentation

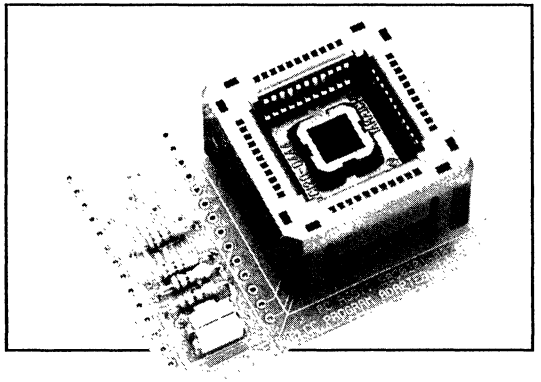
Z86E40ZDP Adapter User Guide

ORDERING INFORMATION

Part No: Z86E4000ZDP

Z86E4000ZDV

PRODUCT SPECIFICATION



SUPPORTED DEVICES

Z86E40

DESCRIPTION

The Z86E40 PLCC OTP DIP Adapter Kit allows a standard EPROM programmer to program the Z86E40 One-Time-Programmable microcontroller.

SPECIFICATIONS

Power Requirements

+12.5 Vdc @ .5 A

Dimensions

Width: 1.6 in. (4.1 cm)
Length: 2.0 in. (5.1 cm)

KIT CONTENTS

Z86E40 PLCC OTP Program Adapter Board

40-Pin ZIF Socket
28-Pin Connector

Documentation

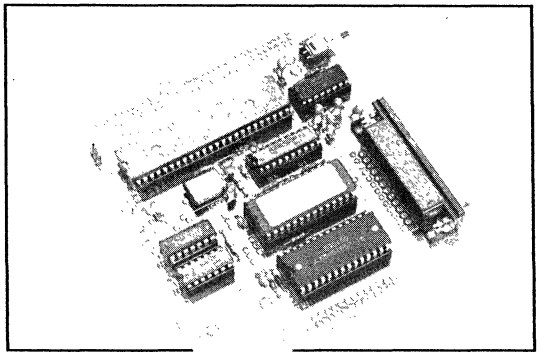
Z86E4000ZDV Adapter User Guide

ORDERING INFORMATION

Part No: Z86E4000ZDV

Z0880000ZCO

PRODUCT SPECIFICATION



SUPPORTED DEVICES

Z08800

DESCRIPTION

The Super8 Development Kit can be used for several purposes. As an evaluation tool, the user can learn the Super8's instruction set plus the manipulation of the Super8's interrupt vectors and register set. Secondly, The Super8 Development Kit is designed to aid the user in constructing specific applications using the Super8 microcontroller. Lastly, application prototypes may be run using the Super8 Development Kit.

SPECIFICATIONS

Power Requirements

+5 Vdc @ .4 A

Dimensions

Width: 4.0 in. (10.2 cm)
Length: 4.5 in. (11.4 cm)

Serial Interface

RS-232C @ 9600 baud

KIT CONTENTS

Super8 Development Board

NMOS Z08800 Super8 MPU
20 MHz Crystal
(32K)/8K x 8 EPROM
(32K)/8K x 8 STATIC RAM
RS-232C PC Interface

Software (IBM-PC Platform)

Z8/Super8 Assembler/Utilities
Host Communication Package
Monitor Instructions
Tutorial
Sample Z08802 Application Software

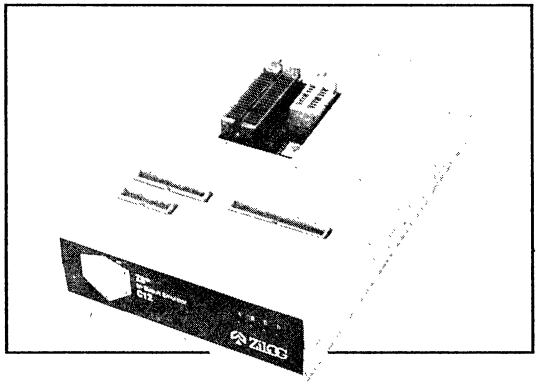
Documentation

Z8 Family Data Book
Z8 Cross Assembler User Guide
MOBJ Link/Loader User Guide

ORDERING INFORMATION

Part No: Z0880000ZCO

Z86C1200ZEM
PRODUCT SPECIFICATION



SUPPORTED DEVICES
Z86C08, Z86E08, Z86C00, Z86C10, Z86C11 Z86C20, Z86C21, Z86E21, Z86C91

DESCRIPTION
The Z86C1200ZEM is a member of Zilog's ICE BOX product family of in-circuit emulators. The ICE BOX -C12 provides emulation and OTP programming support for Zilog's Z8 microcontroller. The Emulator provides all the essential MCU timing and I/O circuitry which simplifies user emulation of the prototype hardware/software product. The Emulator can be connected to a serial port COM 1 or COM 2 of the host computer (IBM* XT, AT Compatible).

SPECIFICATIONS
Emulation Specification
Maximum Emulation Speed 16 MHz

Power Requirements
+5 Vdc @ 1.0 A

Dimensions
Width: 6.0 in. (15.2 cm)
Length: 8.8 in. (22.4 cm)

Serial Interface
RS-232C @ 19200 baud

KIT CONTENTS
Z86C12 Emulator
Z8 Emulation Base Board
- CMOS Z86C9120PSC
- 8K x 8 EPROM (Programmed with Debug Monitor)
- EPM5128 EPLD
- 32K x 8 STATIC RAM
- 3 64K x 4 STATIC RAM
- RS-232C Interface
- Reset Switch

Z86C12 Emulation Daughter Board
- EPM5032 EPLD
- 16 MHz CMOS Z86C1216GSE ICE Chip
- 40/18 Pin ZIF OTP Sockets
- 80/60/40 Pin Target Connectors

Components
Z86E2112PSC
Z86E0812PSC

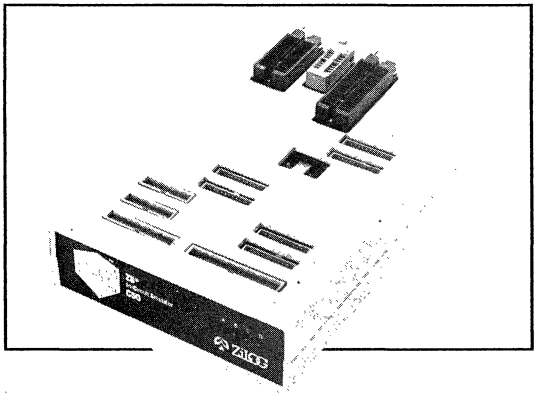
Cables
12", 40-Pin DIP Emulation Cable
12", 28-Pin DIP Emulation Cable
12", 18-Pin DIP Emulation Cable
48" Power Cable
15" Power Cable with Banana Plugs
60" DB 25 RS-232C Cable

Software (IBM-PC Platform)
Z8/Z80/Z8000 Cross Assembler
MOBJ Link/Loader
Host Package

Documentation
Emulator User Guide
Support Products Catalog
Z8 Cross Assembler User Guide
MOBJ Link/Loader User Guide
Registration Card

ORDERING INFORMATION
PART NO: Z86C1200ZEM

Z86C5000ZEM
PRODUCT SPECIFICATION



SUPPORTED DEVICES
Z86C06, Z86C09/19, Z86E19, Z86C30, Z86E30, Z86C40, Z86E40, Z86C89, Z86C90

DESCRIPTION
The Z86C5000ZEM is a member of Zilog's ICE BOX product family of in-circuit emulators. The ICE BOX -C50 provides emulation and OTP programming support for Zilog's Consumer Controller Processor (CCP) microcontroller. The Emulator provides all the essential MCU timing and I/O circuitry which simplifies user emulation of the prototype hardware/software product. The Emulator can be connected to a serial port COM 1 or COM 2 of the host computer (IBM* XT, AT Compatible).

SPECIFICATIONS
Emulation Specification
Maximum Emulation Speed 16 MHz

Power Requirements
+5 Vdc @ 1.0 A

Dimensions
Width: 6.0 in. (15.2 cm)
Length: 8.8 in. (22.4 cm)

Serial Interface
RS-232C @ 19200 baud

KIT CONTENTS
Z86C50 Emulator
Z8 Emulation Base Board
- CMOS Z86C9120PSC
- 8K x 8 EPROM (Programmed with Debug Monitor)
- EPM5128 EPLD
- 32K x 8 STATIC RAM
- 3 64K x 4 STATIC RAMs
- RS-232C Interface
- Reset Switch

Z86C50 Emulation Daughter Board
- 20 MHz CMOS Z86C5020GSE ICE Chip
- EPM5128 EPLD
- 2K x 8 STATIC RAM
- 40/28/18 Pin ZIF OTP Sockets
- 6 HP-16500A Logic Analysis System interface Connectors
- 80/60/40 Pin Target Connectors

Components
Z86E4012PSC
Z86E3012PSC

Cables
12", 40-Pin DIP Emulation Cable
12", 28-Pin DIP Emulation Cable
12", 18-Pin DIP Emulation Cable
48" Power Cable
15" Power Cable with Banana Plugs
60" DB 25 RS-232C Cable

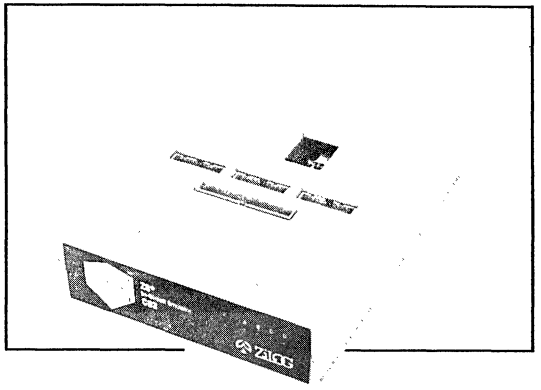
Software (IBM-PC Platform)
Z8/Z80/Z8000 Cross Assembler
MOBJ Link/Loader
Host Package

Documentation
Emulator User Guide
Support Products Catalog
Z8 Cross Assembler User Guide
MOBJ Link/Loader User Guide
Registration Card

ORDERING INFORMATION
Part No: Z86C5000ZEM

Z86C9300ZEM

PRODUCT SPECIFICATION



SUPPORTED DEVICES

Z86C93

DESCRIPTION

The Z86C9300ZEM is a member of Zilog's ICE BOX product family of in-circuit emulators. The ICE BOX -C93 provides emulation for Zilog's Z86C93 microcontroller. This includes all the essential MCU timing and I/O circuitry which simplifies user emulation of the prototype hardware/software product. The Emulator can be connected to a serial port COM 1 or COM 2 of the host computer (IBM* XT, AT, 386, 486 Compatible).

SPECIFICATIONS

Emulation Specification

Maximum Emulation Speed 16 MHz

Power Requirements

+5 Vdc @ .5A

Dimensions

Width: 6.0 in. (15.2 cm)
Length: 8.8 in. (22.4 cm)

Serial Interface

RS-232C @ 19200 baud

KIT CONTENTS

Z86C93 Emulator

Z8 Emulation Base Board
- CMOS Z86C9120PSC
- 8K x 8 EPROM
  (Programmed with Debug Monitor)
- EPM5128 EPLD
- 32K x 8 STATIC RAM
- 3 64K x 4 STATIC RAMs
- RS-232C Interface
- Reset Switch

Z86C93 Emulation Daughter Board
- 20 MHz CMOS Z86C5020GSE ICE Chip
- EPM5128 EPLD
- 3 HP-16500A Logic Analysis System Interface Connectors
- 80-Pin Target Connector

Cables

12", 44-Pin DIP Emulation Cable
12", 40-Pin DIP Emulation Cable
12", 18-Pin DIP Emulation Cable
48" Power Cable
15" Power Cable with Banana Plugs
60" DB 25 RS-232C Cable

Software (IBM-PC Platform)

Z8/Z80/Z8000 Cross Assembler
MOBJ Link/Loader
Host Package

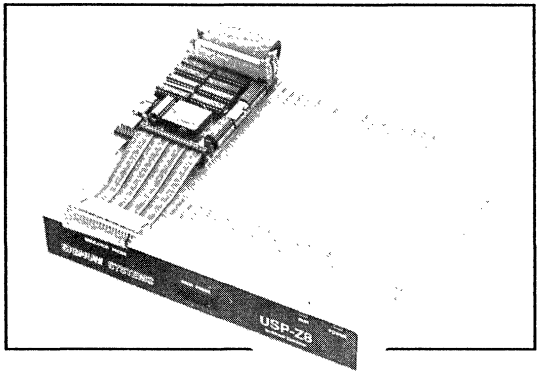
Documentation

Emulator User Guide
Support Products Catalog
Z8 Cross Assembler User Guide
MOBJ Link/Loader User Guide
Registration Card

ORDERING INFORMATION

Part No: Z86C9300ZEM

Z8
REAL TIME EMULATOR



DESCRIPTION

The system comprises three base unit options, (64K, 128K, or 256K of emulation program ROM), and three pod options which allow the emulation of various Z8 microcontrollers. Features include real-time transparent emulation up to 20MHz, in-line symbolic assembler and disassembler, real-time hardware breakpoints, eight channel user logic analyser, external trigger input and outputs, trace display and memory display/edit during execution, and window or command driven user interface.

SPECIFICATIONS

Microcontrollers Emulated:

Z86C1200ZPD	Z86C00, Z86C10, Z86C20, Z86C11, Z86C21, Z86E21, Z86C91
Z86C5000ZPD	Z86C09, Z86C19, Z86C30, Z86C40, Z86C90
Z86C9300ZPD	Z86C93

Maximum Emulation Speed:
Up to 20 MHz (microcontroller dependent)

Size:
260 mm wide, 260 mm deep, 64 mm high

Operating Temperature:
0°C to +40°C

Storage Temperature:
-10°C to +65°C

Operating Humidity:
0 to 90%

Maximum Emulation Program Memory:
64 Kbytes with Z86C0000ZUSP064
128 Kbytes with Z86C0000ZUSSP128
256 Kbytes with Z86C0000ZUSP256

Maximum Emulation Data Memory:
64 Kbytes

Program Memory Mapping:
1K blocks

Pass Counters:
Two, 16-bit each

Trace Buffer:
32K - 80 bits

Sequencer:
Hardware, 8 levels

User Probe:
Eight channel logic input
One trigger input
Sever trigger outputs (Events, Pass Counters, Sequencer)

Host Interface:
Asynchronous RS-232C
9600/115 KBaud
XON/XOFF support

File Upward/Downward Format:
Zilog MUFOM (EEE 695-1985)
Intel HEX
Intel AOMF
2500AD Software

MINIMUM HOST REQUIREMENTS

- IBM compatible PC/XT/AT/386 or PS-2
- 640 Kbyte memory
- 20 Mbyte hard disk
- RS-232 serial port (COM1 or COM2)
- Mouse (serial or bus)
- MDA, CGA, EGA, or VGA video adaptor

MINIMUM EMULATION SUPPORT

One base unit and one emulation pod is required.

Part Numbers

Base Systems	Pod Systems
Z86C0000ZUSP064	Z86C9300ZPD
Z86C0000ZUSP128	Z86C1200ZPD
Z86C0000ZUSP256	Z86C5000ZPD

ZiLOG QUALITY AND RELIABILITY

Zilog's Quality and Reliability Program

Introduction

Zilog has an excellent reputation for the quality and reliability of its products.

Zilog's Quality and Reliability Program is based on careful study of the principles laid down by such pioneers as W.E. Deming and J.M. Juran and, perhaps even more important, observation of the practical implementation of those principles in Japanese, European and American manufacturing facilities.

The Zilog program begins with employee involvement. Whether the judgement of our performance is based on perfection in incoming inspection, trouble free service in the field or timely and accurate customer service, we recognize that our employees ultimately control these factors. Hence, our Quality Program is broadly shared throughout the organization.

1. Harmony Between Design and Process

High product quality and reliability in VLSI products is possible only if there is structural harmony between product design and the manufacturing process. Great care is taken to assure that the statistical process control limits observed within the manufacturing plants properly guardband the design technology used to configure the circuit and layout in Zilog's automated design methodology.

Through use of a technique which we call Process Templating, the technology file in the automated design system is periodically updated to assure that product design parameters fall within the statistical control limits with which the process is actually operated. In simple terms, the Process Template is the profile displayed by the process evaluation parameters which are automatically recorded from the test patterns on wafers as they proceed through the production line. These parameters are translated into the design technology file attributes such that every product design bears a key and lock relationship to the process.

2. Training

Product Design and Processing are people dependent. Zilog training emphasizes the fundamentals involved in design for quality and reliability.

Customer Service, an important aspect of Zilog's quality performance as a vendor, also depends upon our people clearly understanding their jobs, and our obligations to our customers. This too is part of the training curriculum administered by Zilog.

3. Order Acknowledgement Policy

One definition of vendor quality performance is that the vendor "does what he promises or acknowledges." Reliability and quality warranties can be met only if Zilog and the customer are in agreement on product and delivery specifications. Zilog makes an extra effort to assure that the customer is fully informed by providing documents with its purchase order acknowledgements that clearly state what Zilog understands the specifications to be.

4. Test Guardbanding

No physical attribute is absolute. Customers' test methods may differ from Zilog's due to variations in test equipment, temperature or specification interpretation. To assure that every Zilog product performs to full customer expectations, Zilog uses a "waterfall" methodology in its testing. The first electrical tests made on the circuit, at the wafer probe operations, are guardbanded to the final test specifications. The final test specifications, in turn, are guardbanded to the quality control outgoing sample. The quality control outgoing sample is guardbanded to the customer procurement or data sheet specifications. This technique of "waterfall" guardbanding assures that circuits which may be marginal to the customer's expectations are eliminated in the manufacturing process long before they get to the shipping container.

5. Probe at Temperature

Semiconductor devices tend to exhibit their most limited performance at the highest operating temperature. Therefore, it is Zilog's policy that all chips are tested at high temperature the very first time they are electrically screened, at the wafer probe station. The circuits are tested again at their upper operating temperature limit in the 100% final test operation.

6. Process Characterization

Before release to production, every process is thoroughly characterized by an exhaustive series of pilot production runs and tests which identify the statistical, electrical, and mechanical limits of which that particular process regime is capable. This documentation, which fills a large loose leaf binder for each process, is maintained as the historical record or "footprint" for that particular regime.

Process recharacterization is done any time there is a major process or manufacturing site change, and the resulting documentation is then added to the characterization history. Once the process is fully characterized, the frequent test site evaluation and process template data demonstrates that the process remains in specification.

7. Product Characterization

Every Zilog product design is evaluated over extremes of operating temperature, supply voltage and clock frequencies, prior to release to production. This information permits the proper guardbanding of the test program waterfall and identification of many marginal "corners" in design tolerances.

A product characterization report, which summarizes the more important tolerances identified in the process of this exhaustive product design evaluation, is available to Zilog's customers.

8. Process Qualification

Zilog also qualifies every process prior to production by an exhaustive stress sequence performed on test chips and on representative products. Once a process regime is qualified, a process requalification is performed any time there is a major process change, or whenever the process template statistical quality limits are significantly exceeded or adjusted.

9. Product Qualification

In addition to characterization, every new Zilog product design is fully qualified by a comprehensive series of life, electrical, and environmental tests before release to production. Again, a qualification report is available to our customers which summarizes certain key life and environmental data taken in the course of these evaluations. Whenever possible, industry standard environmental and life tests are employed.

10. PPM Measurement, Direct and Indirect

It is frequently said that if you want to improve something, you need to put a measure on it. Therefore, Zilog measures its outgoing quality "parts per million" by the maintenance of careful records on the statistical sampling of production lots prepared for shipment. This information is then translated by our statisticians to a statement of our parts per million (or parts per billion) outgoing quality performance.

Of course, it is one thing for Zilog to think it is doing a good job in outgoing product quality and it is another for a customer to agree. Therefore, we ask certain key customers to provide us with their incoming inspection data which helps us calibrate our outgoing performance in terms of the actual results in the field. The fact that Zilog has been awarded "ship to stock" status by many customers testifies to our success in this area.

11. FIT Measurement Direct and Indirect

Just as Zilog records its outgoing quality in terms of parts per million, it also measures its outgoing product reliability in terms of "FITS" or failures per billion device hours, using the results of weekly operating life test measurements on the circuits, performed in accordance with the standard specifications.

12. Field Quality Engineers

It is frequently said that, "the customer is always right." If the customer has an application quality or reliability problem while using a Zilog product, whether it is Zilog's responsibility or not, we believe that we have a responsibility to resolve it. Therefore, Zilog maintains a force of skilled Applications Engineers who are also trained as field quality engineers and are available on immediate call to consult at the customer's locations on any problems they may be experiencing with Zilog product performance.

13. Product Analysis

As noted earlier, we feel that a customer problem is a Zilog problem. Accordingly, Product Analysis facilities, staffed by experienced professionals, exist at each Zilog site to provide rapid evaluation of in-process and in-field rejects to determine the cause and provide corrective action through a feedback loop into the production, design, and applications process. Zilog is pleased to share product analysis reports on specific products with the customer upon request.

14. Test Site Step-Stress

The process evaluation test sites on the wafer are packaged and subjected to step-stress testing. Any drift in parameters under severe conditions of stress outside the norm is taken as an indication of possible process contamination or variation.

15. Statistical Process Control

Zilog employs Statistical Process Control at all critical process steps. Deviations from norms must be evaluated by a Q/R review board.

16. Perfection Plus Program

Zilog employees actively participate in meetings in which methods which will enable a department to do its job more perfectly are proposed, reviewed, and adopted. Employees who have made suggestions proudly wear the Zilog Perfection Plus pin.

17. Zilog Vendor of the Year Award

Zilog is proud of the many quality and performance awards it has received from its customers. In turn, Zilog makes an annual award to the vendor who has done the best overall job for Zilog.

Zilog's Quality and Reliability Summary

Zilog is proud of its Quality and Reliability programs and is pleased to share this data with its customers. For further information, contact Zilog's Director of R/QA.

ZiLOG LITERATURE GUIDE

Z8®/SUPER8™ MICROCONTROLLER FAMILY

Handbook	Part No	Unit Cost
Microcontrollers Data Book (includes the following documents)	DC-8275-04	5.00

Z8 CMOS Microcontrollers

Z86C00/C10/C20 MCU OTP Product Specification
Z86C06 Z8 CCP™ Preliminary Product Specification
Z86C08 8-Bit MCU Product Specification
Z86E08 Z8 OTP MCU Product Specification
Z86C09/19 Z8 CCP Product Specification
Z86E19 Z8 OTP MCU Advance Information Specification
Z86C11 Z8 MCU Product Specification
Z86C12 Z8 ICE Product Specification
Z86C21 Z8 MCU Product Specification
Z86E21/Z86E22 OTP Product Specification
Z86C30 Z8 CCP Product Specification
Z86E30 Z8 OTP CCP Product Specification
Z86C40 Z8 CCP Product Specification
Z86E40 Z8 OTP CCP Product Specification
Z86C27/97 Z8 DTC™ Product Specification
Z86127 Low-Cost Digital Television Controller Adv. Info. Spec.
Z86C50 Z8 CCP ICE Advance Information Specification
Z86C61 Z8 MCU Advance Information Specification
Z86C62 Z8 MCU Advance Information Specification
Z86C89/C90 CMOS Z8 CCP Product Specification
Z86C91 Z8 ROMless MCU Product Specification
Z86C93 Z8 ROMless MCU Preliminary Product Specification
Z86C94 Z8 ROMless MCU Product Specification
Z86C96 Z8 ROMless MCU Advance Information Specification
Z88C00 CMOS Super8 MCU Advance Information Specification

Z8 NMOS Microcontrollers

Z8600 Z8 MCU Product Specification
Z8601/03/11/13 Z8 MCU Product Specification
Z8602 8-Bit Keyboard Controller Preliminary Product Spec.
Z8604 8-Bit MCU Product Specification
Z8612 Z8 ICE Product Specification
Z8671 Z8 MCU With BASIC/Debug Intrepreter Product Spec.
Z8681/82 Z8 MCU ROMless Product Specification
Z8691 Z8 MCU ROMless Product Specification
Z8800/01/20/22 Super8 ROMless/ROM Product Specification

Peripheral Products

Z86128 Closed-Captioned Controller Adv. Info. Specification
Z765A Floppy Disk Controller Product Specification
Z5380 SCSI Product Specification
Z53C80 SCSI Advance Information Specification

Z8 Application Notes and Technical Articles

Zilog Family On-Chip Oscillator Design
Z86E21 Z8 Low Cost Thermal Printer
Z8 Applications for I/O Port Expansions
Z86C09/19 Low Cost Z8 MCU Emulator
Z8602 Controls A 101/102 PC/Keyboard
The Z8 MCU Dual Analog Comparator
The Z8 MCU In Telephone Answering Systems
Z8 Subroutine Library
A Comparison of MCU Units
Z86xx Interrupt Request Registers
Z8 Family Framing
A Programmer's Guide to the Z8 MCU
Memory Space and Register Organization

Super8 Application Notes and Technical Articles

Getting Started with the Zilog Super8
Polled Async Serial Operations with the Super8
Using the Super8 Interrupt Driven Communications
Using the Super8 Serial Port with DMA
Generating Sine Waves with Super8
Generating DTMF Tones with Super8
A Simple Serial Parallel Converter Using the Super8

Additional Information

Z8 Support Products
Zilog Quality and Reliability Report
Literature List
Package Information
Ordering Information

Z8®/SUPER8™ MICROCONTROLLER FAMILY (Continued)

Z8 Product Specifications, Technical Manuals and Users Guides	Part No	Unit Cost
asm S8 Super8/Z8 Cross Assembler User's Guide	DC-8267-05	3.00
Z86C06 CMOS Z8 CCP Preliminary Product Specification	DC-2563-00	N/C
Z86C08 CMOS Z8 8-Bit Microcontroller Product Specification	DC-2527-02	N/C
Z86E08 CMOS Z8 8-Bit Microcontroller Product Specification	DC-2542-02	N/C
Z86C09/C19 CMOS Z8 8-Bit Microcontroller Product Specification	DC-2506-02	N/C
Z86C11 CMOS Z8 Microcontroller Product Specification	DC-2572-01	N/C
Z86C12 Z8 ICE Product Specification	DC-2553-01	N/C
Z86C21 CMOS Z8 Microcontroller	DC-2568-01	N/C
Z86E21 CMOS OTP Microcontroller Product Specification	DC-2514-01	N/C
Z86C27/97 Z8 DTC™ Product Specification	DC-2561-01	N/C
Z86127 Low-Cost Digital Television Controller Advance Information Specification	DC-2574-0A	N/C
Z86C30 CMOS Z8 8-Bit Microcontroller Product Specification	DC-2509-03	N/C
Z86E30 CMOS Z8 OTP CCP Product Specification	DC-2573-01	N/C
Z86C40 CMOS CCP Product Specification	DC-2550-01	N/C
Z86E40 CMOS OTP CCP Product Specification	DC-2571-01	N/C
Z86C50 CMOS Z8 CCP ICE Advance Information Specification	DC-2559-0A	N/C
Z86C89/C90 ROMless CMOS Z8 8-Bit Microcontroller Product Specification	DC-2506-01	N/C
Z86C91 Z8 CMOS ROMless Microcontroller Product Specification	DC-2566-01	N/C
Z86C93 CMOS Z8 ROMless Microcontroller Preliminary Product Specification	DC-2508-01	N/C
Z88C00 CMOS Super8 ROMless Microcontroller Advance Information Specification	DC-2551-0A	N/C
Z8602 NMOS Z8 8-Bit Microcomputer Keyboard Controller Preliminary Product Spec.	DC-2525-01	N/C
Z8604 NMOS Z8 8-Bit Microcontroller Preliminary Product Specification	DC-2524-02	N/C
Z86128 Closed-Captioned Controller Advance Information Specification	DC-2570-0A	N/C
Z8671 Single Chip Basic Interpreter Basic Debug Software Reference Manual	DC-3149-03	3.00

Z8 Application Notes	Part No	Unit Cost
The Z8 MCU In Telephone Answering Systems	DC-2514-01	N/C
Z8602 Controls A 101/102 PC/Keyboard	DC-2521-01	N/C
The Z8 MCU Dual Analog Comparator	DC-2516-01	N/C
Z86C09/19 Low Cost Z8 MCU Emulator	DC-2537-01	N/C
Z8 Applications for I/O Port Expansions	DC-2539-01	N/C
Z86E21 Z8 Low Cost Thermal Printer	DC-2541-01	N/C
Zilog Family On-Chip Oscillator Design	DC-2496-01	N/C

(Additional Application Notes are contained in the above Design Handbook)

ZiLOG LITERATURE GUIDE

Z80®/Z180™/Z280™ MICROPROCESSOR FAMILY

Data Book	Part No	Unit Cost
Intelligent Peripherals Data Book (includes the following documents)	DC-2480-02	5.00

Z80 NMOS/CMOS

Z84C00 NMOS/CMOS Z80 CPU Prelim. Product Specification
Z84C01 Z80 CPU with CGC Product Specification
Z8410/C10 NMOS/CMOS Z80 DMA Product Specification
Z8420/C20 NMOS/CMOS Z80 PIO Product Specification
Z8430/C30 NMOS/CMOS Z80 CTC Product Specification
Z8440/1/2/4/C40/1/2/3/4 NMOS/CMOS Z80 SIO Product Spec.
Z84C50 RAM80™ Preliminary Product Specification
Z8470 Z80 DART Product Specification
Z84C90 CMOS Z80 KIO™ Product Specification
Z84011/C11 PIO Parallel I/O Controller Product Specification
Z84013/15, Z84C13/C15 CMOS IPC™ Product Specification
Z80180 Z180™ MPU Product Specification
Z80181 SAC™ Product Specification
Z280™ MPU Preliminary Product Specification

Z80 Application Notes and Technical Articles

Z80 Family Interrupt Structure
Using the Z80 SIO in Async Communications
Using the Z80 SIO with SDLC
Binary Synchronous Comm Using the Z80 SIO
Serial Communication with the Z80A DART
Interfacing Z8500 Peripherals to the Z80
Serial Clock Generation using the Z8536 CIO
Timing in Interrupt-Based System with Z80 CTC
A Z80-Based System Using the DMA with the SIO
Interfacing Z80 CPUs to the Z8500 Peripheral Family
Z180/SCC™ Serial Communications Controller Interface at 10 MHz
Z80 Using the 84C11/C13/C15 in place of the 84011/013/015
Z80 Questions and Answers
Z180 Questions and Answers

Other Information

Z80 Product Support
Zilog Quality and Reliability Report
Zilog Literature List
Package Information
Ordering Information

Z80/Z180/Z280 Product Specifications, Technical Manuals and Users Guides	Part No	Unit Cost
Z80 CPU Central Processing Unit Technical Manual	DC-0029-03	3.00
Z80 Family Programmer's Reference Guide	DC-0012-04	3.00
Z80 DMA Direct Memory Access Technical Manual	DC-2013-A0	3.00
Z80 PIO Parallel Input/Output Technical Manual	DC-0008-03	3.00
Z80 CTC Counter/Timer Circuit Technical Manual	DC-0036-02	3.00
Z80 SIO Serial I/O Technical Manual	DC-3033-01	3.00
Z80181 Z180 MPU Microprocessor Unit Technical Manual	DC-8276-03	3.00
Z280 MPU Microprocessor Unit Technical Manual	DC-8224-03	3.00
Z80181 Z181 SAC™ Smart Access Controller Preliminary Product Specification	DC-2519-02	N/C
Z84013/15, Z84C13/C15 CMOS IPC™ Intelligent Peripheral Controller Preliminary Product Specification	DC-2507-02	N/C
Z84011/C11 PIO Parallel I/O Controller Product Specification	DC-2526-04	N/C
Z84C00 20 MHz Z80 CPU Central Processing Unit Preliminary Product Specification	DC-2523-02	N/C
Z84C50 Z80 RAM80 Z80 CPU/2K SRAM Preliminary Product Specification	DC-2498-01	N/C

Z80/Z180/Z280 Application Notes	Part No	Unit Cost
Z180/SCC™ Serial Communications Controller Interface at 10 MHz	DC-2521-02	N/C
Z80 Using the 84C11/C13/C15 in place of the 84011/013/015	DC-2499-02	N/C

(Additional Application Notes are contained in the above Databook)

ZiLOG LITERATURE GUIDE

Z8000® MICROPROCESSOR FAMILY

Data Book	Part No	Unit Cost
Datacom Family Data Book (includes the following documents)	DC-2503-02	5.00

Z8000/80,000 NMOS/CMOS Microprocessors
Z16C30 CMOS USC™ Product Specification
Z16C31 IUSC™ Product Specification
Z16C33 CMOS MUSC™ Product Specification
Z16C35 CMOS ISCC™ Product Specification
Z16C50 DDPLL™ Product Specification
Z5380 CMOS SCSI Product Specification
Z85230 CMOS ESCC™ Product Specification
Z80C30/Z85C30 CMOS Z-BUS® SCC™ Product Specification
Z8030/Z8530 Z-BUS SCC Product Specification
Z80181 CMOS SAC Product Specification
Z84013/015 Z84C13/C15 IPC™ Product Specification
Z8440/Z84C40 SIO Product Specification
Z85C80 SCSCI™ Product Specification

Application Notes and Technical Articles
Design a Serial Board to Handle Multiple Protocols
Using the Z16C30 USC Universal Serial Controller with MIL-STD-1553B
Datacommunications IUSC/MUSC Time Slot Assigner
ISCC Interface to the 68000® and 8086®
The Z180 SCC Interfaced with the SCC at 10 MHz
Using SCC with Z8000® in SDLC Protocol
SCC In Binary Synchronous Communications
On-Chip Oscillator Design
Interfacing the Z8500 Peripherals to the 68000
Interfacing Z80 CPUs to the Z8500 Peripheral Family
Zilog Quality and Reliability Report
Literature List
Package Information
Ordering Information

Z8000 Product Specifications, Technical Manuals and Users Guides	Part No	Unit Cost
Z8000 CPU Central Processing Unit Technical Manual	DC-2010-06	3.00
Z8010 MMU Memory Management Unit Technical Manual	DC-2015-A0	3.00
Z8030/Z8530 SCC Serial Communications Controller Technical Manual	DC-2057-06	3.00
Z8036 Z-CIO/Z8536 CIO Counter/Timer and Parallel Input/Output Technical Manual	DC-2091-02	3.00
Z8038 Z8000 Z-FIO FIFO Input/Output Interface Technical Manual	DC-2051-01	3.00
Z8000 CPU Central Processing Unit Programmer's Pocket Guide	DC-0122-03	3.00
Z5380 SCSI Small Computer System Interface Preliminary Product Specification	DC-2477-01	N/C
Z80C30/Z85C30 CMOS SCC Serial Communications Controller Product Specification	DC-2442-05	N/C
Z85C80 SCSCI™ Serial Communications and Small Computer Interface Preliminary Product Specification	DC-2534-02	N/C
Z16C01/2/3 CPU Central Processing Unit Preliminary Product Specification	DC-2504-02	N/C
Z16C30 CMOS USC™ Universal Serial Controller Preliminary Product Specification	DC-2492-03	N/C
Z16C30/Z16C33 CMOS USC/MUSC™ Universal Serial Controller Technical Manual	DC-8285-01	3.00
Z16C30/Z16C33 CMOS USC/MUSC Universal Serial Controller Addendum	DC-8285-01A	N/C
Z16C31 IUSC Integrated Universal Serial Controller Advanced Information Specification	DC-2544-01	N/C
Z16C33 CMOS MUSC Mono-Universal Serial Controller Preliminary Product Specification	DC-2517-03	N/C
Z16C35 CMOS ISCC™ Integrated Serial Communications Controller Product Specification	DC-2515-03	N/C
Z16C35 ISCC Integrated Serial Communications Controller Technical Manual	DC-8286-01	3.00
Z16C35 ISCC Integrated Serial Communications Controller Addendum	DC-8286-01A	N/C
Z16C50 DDPLL Dual Digital Phase-Locked Loop Preliminary Product Specification	DC-2540-01	N/C
Z85130 ESCC Enhanced Serial Communications Controller Preliminary Product Specification	DC-2543-01	N/C
Z85230/Z80230 ESCC Enhanced Serial Communications Controller Technical Manual	DC-8288-01	3.00

Z8000 Application Notes	Part No	Unit Cost
Z16C30 Using the USC in Military Applications	DC-2536-01	N/C
Z16C35 ISCC Interface to Intel and Motorola Microprocessors	DC-2522-01	N/C
Datacom IUSC/MUSC Time Slot Assigner	DC-2497-02	N/C
Datacom Evaluation Board Using The Zilog Family With The 80186 CPU	DC-2560-01	N/C
ESCC Enhancements Over The SCC	DC-2555-01	N/C
Z16C30 USC - Design a Serial Board for Multiple Protocols	DC-2554-01	N/C

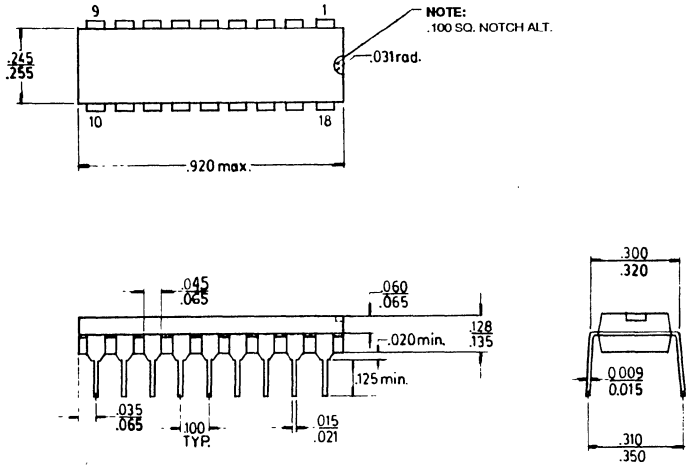
MILITARY COMPONENTS FAMILY

Military Specifications	Part No	Unit Cost
Z8681 ROMless Microcomputer Military Product Specification	DC-2392-02	N/C
Z8001/8002 Military Z8000 CPU Central Processing Unit Military Product Specification	DC-2342-03	N/C
Z8581 Military CGC Clock Generator and Controller Military Product Specification	DC-2346-01	N/C
Z8030 Military Z8000 Z-SCC Serial Communications Controller Military Product Specification	DC-2388-02	N/C
Z8530 Military SCC Serial Communications Controller Military Product Specification	DC-2397-02	N/C
Z8036 Military Z8000 Z-CIO Counter/Timer Controller and Parallel I/O Military Electrical Specification	DC-2389-01	N/C
Z8038/8538 Military FIO FIFO Input/Output Interface Unit Military Product Specification	DC-2463-02	N/C
Z8536 Military CIO Counter/Timer Controller and Parallel I/O Military Electrical Specification	DC-2396-01	N/C
Z8400 Military Z80 CPU Central Processing Unit Military Electrical Specification	DC-2351-02	N/C
Z8420 Military PIO Parallel Input/Output Controller Military Product Specification	DC-2384-02	N/C
Z8430 Military CTC Counter/Timer Circuit Military Electrical Specification	DC-2385-01	N/C
Z8440/1/2/4 Z80 SIO Serial Input/Output Controller Military Product Specification	DC-2386-02	N/C
Z80C30/85C30 Military CMOS SCC Serial Communications Controller Military Product Specification	DC-2478-02	N/C
Z84C00 CMOS Z80 CPU Central Processing Unit Military Product Specification	DC-2441-02	N/C
Z84C20 CMOS Z80 PIO Parallel Input/Output Military Product Specification	DC-2384-02	N/C
Z84C30 CMOS Z80 CTC Counter/Timer Circuit Military Product Specification	DC-2481-01	N/C
Z84C40/1/2/4 CMOS Z80 SIO Serial Input/Output Military Product Specification	DC-2482-01	N/C
Z16C30 CMOS USC Universal Serial Controller Military Preliminary Product Specification	DC-2531-01	N/C
Z16C01/2 CPU Central Processing Unit Military Product Specification	DC-2532-01	N/C
Z80180 Z180 MPU Microprocessor Unit Military Product Specification	DC-2538-01	N/C
Z84C90 CMOS KIO Serial/Parallel/Counter Timer	DC-2502-00	N/C

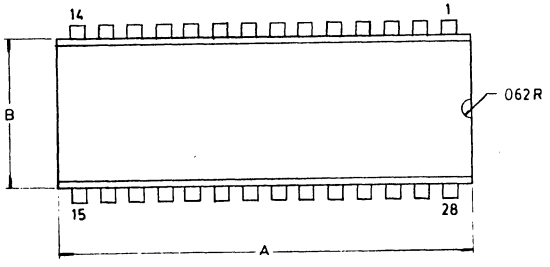
GENERAL LITERATURE

Catalogs, Handbooks and Users Guides	Part No	Unit Cost
Superintegration Short Form Catalog 1991	DC-5472-07	N/C
Quality and Reliability Report	DC-2475-07	N/C
Superintegration Products Guide	DC-5499-03	N/C
The Handling and Storage of Surface Mount Device's User Guide	DC-5500-02	N/C
Support Products Summary	DC-2545-03	N/C
Universal Object File Utilities User's Guide	DC-8236-04	3.00
Zilog 1990 Annual Report	DC-1990-AR	N/C

PACKAGE INFORMATION

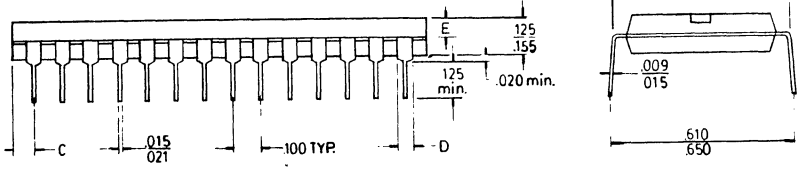


18-Lead Plastic Dual-In-Line Package (DIP)



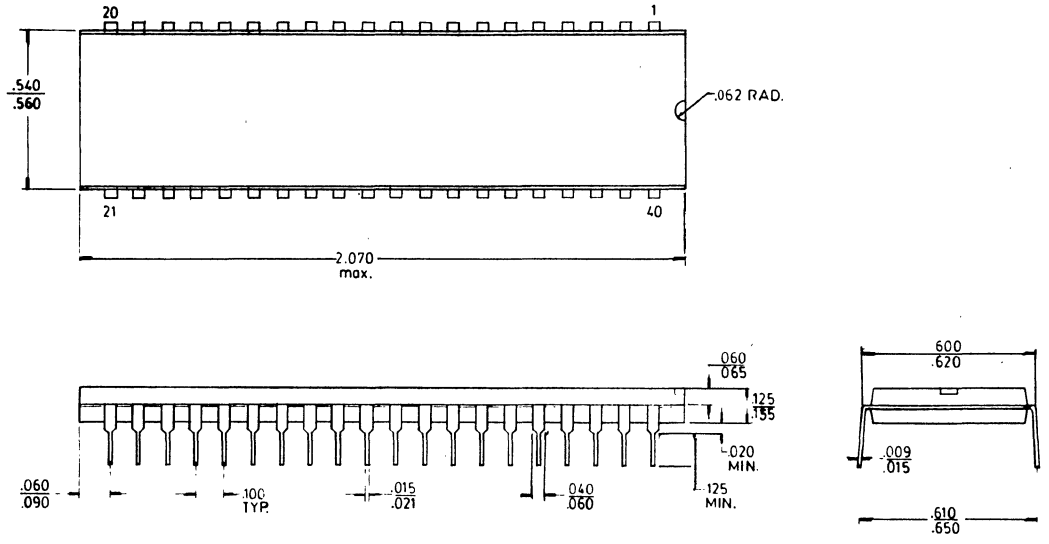
LTR	OPTION 01 STD.PACKAGE	OPTION 02 IDF PACKAGE
A	1.470 max.	1.415 max.
B	.545 / .555	.505 / .515
C	.060 / .090	.040 / .060
D	.060 / .070	.050 / .060
E	.060 / .065	.060 / .070

NOTE:
IDF Package use interdigitated leadframe (IDF) design outline.



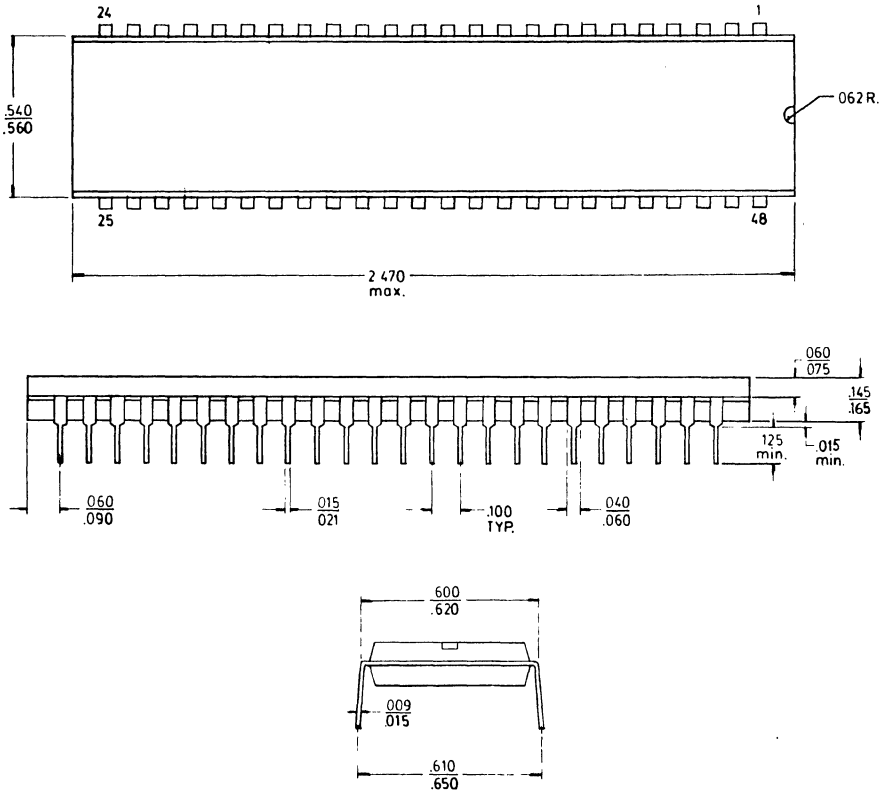
28-Lead Plastic Dual-In-Line Package (DIP)

PACKAGE INFORMATION (Continued)

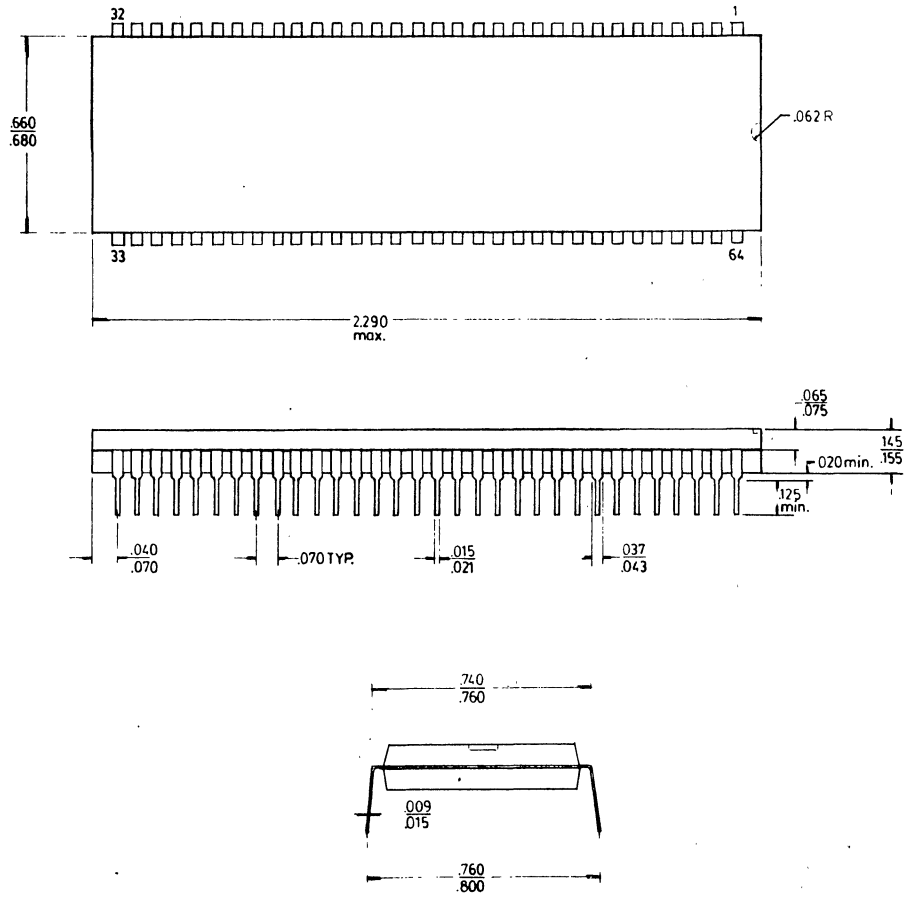


40-Lead Plastic Dual-In-Line Package (DIP)

PACKAGE INFORMATION (Continued)

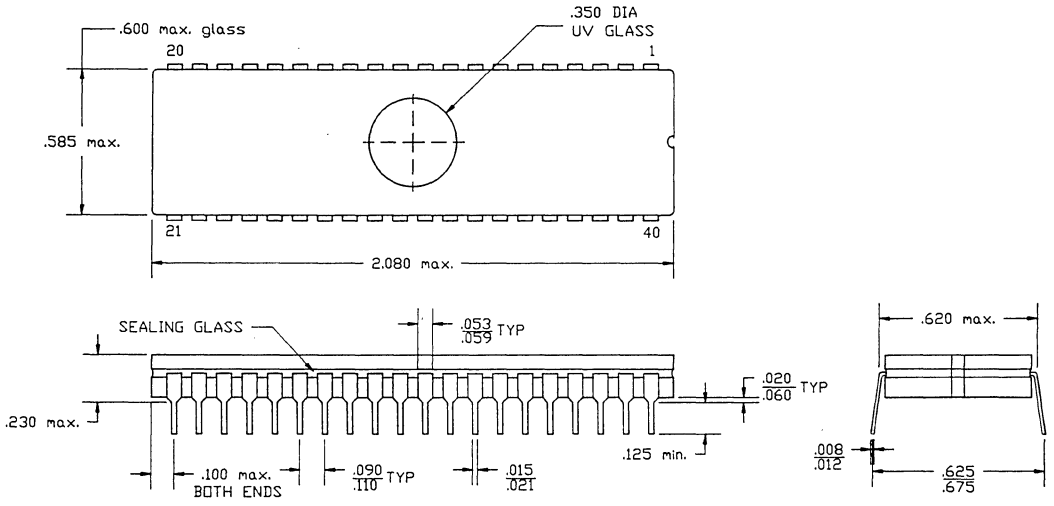


48-Lead Plastic Dual-In-Line Package (DIP)

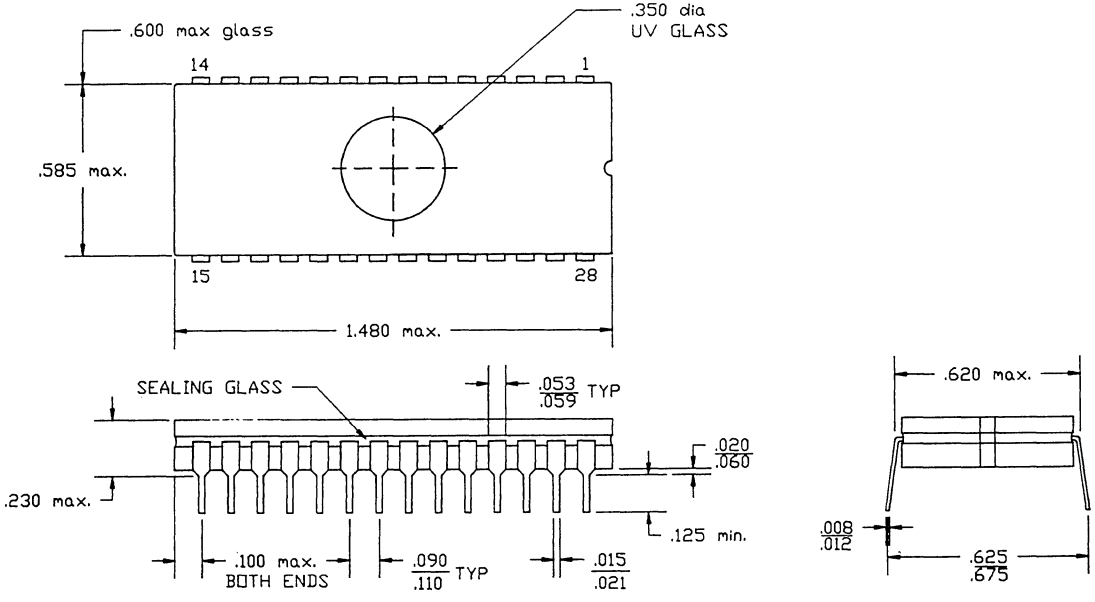


64-Lead Plastic Dual-In-Line Package (DIP)
with 0.070" Lead Centers

PACKAGE INFORMATION (Continued)

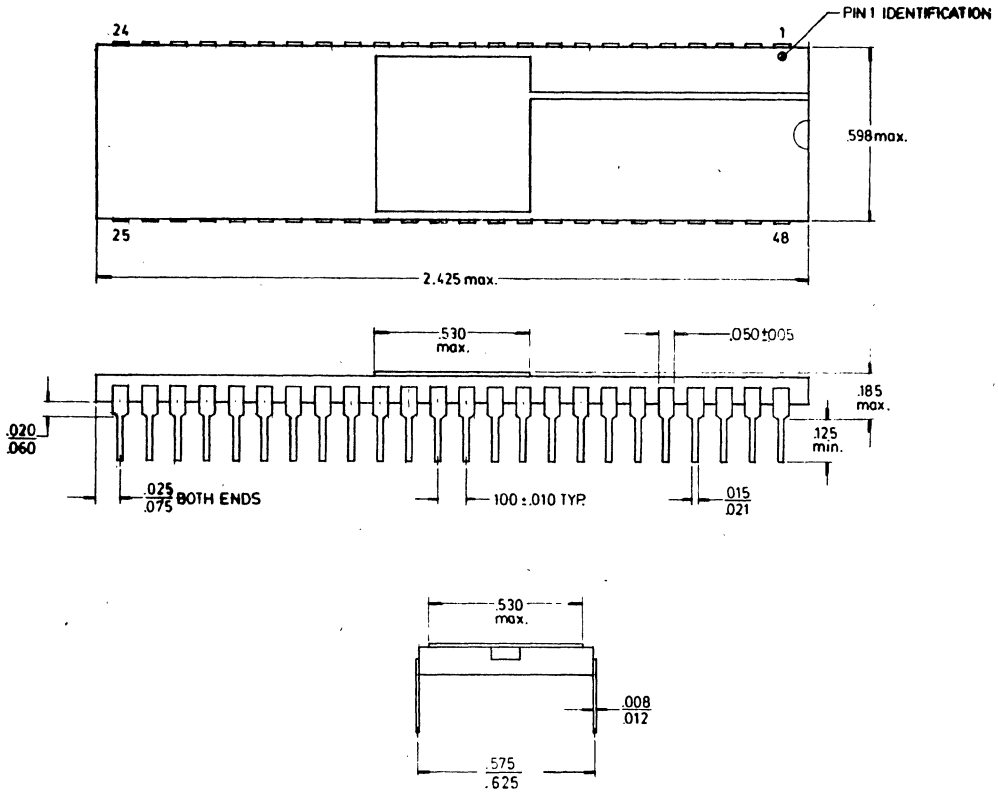


40-Pin Cerdip, Window



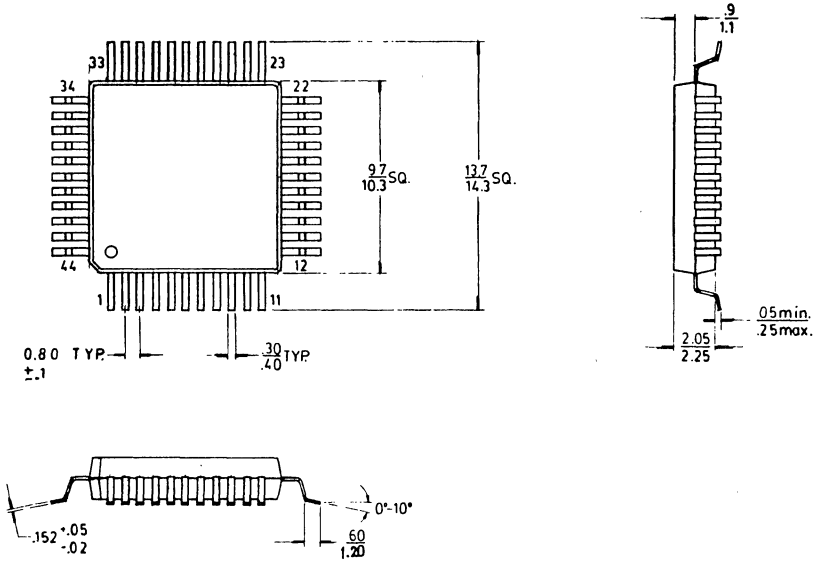
28-Pin Cerdip, Window

PACKAGE INFORMATION (Continued)

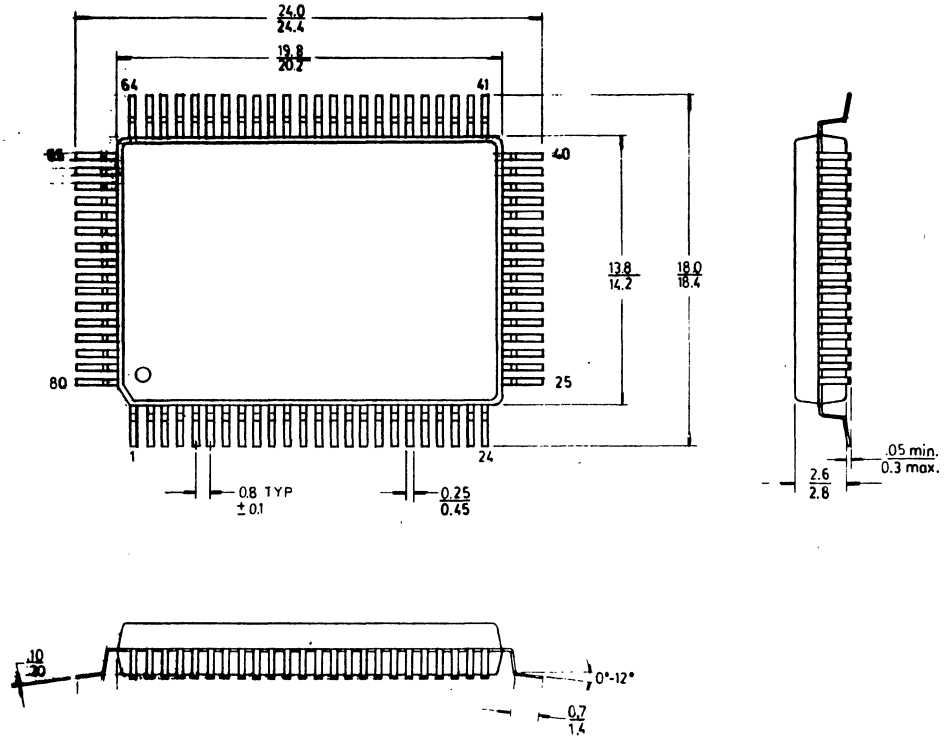


48-Lead Ceramic Sidebrazed Dual-In-Line Package

PACKAGE INFORMATION (Continued)

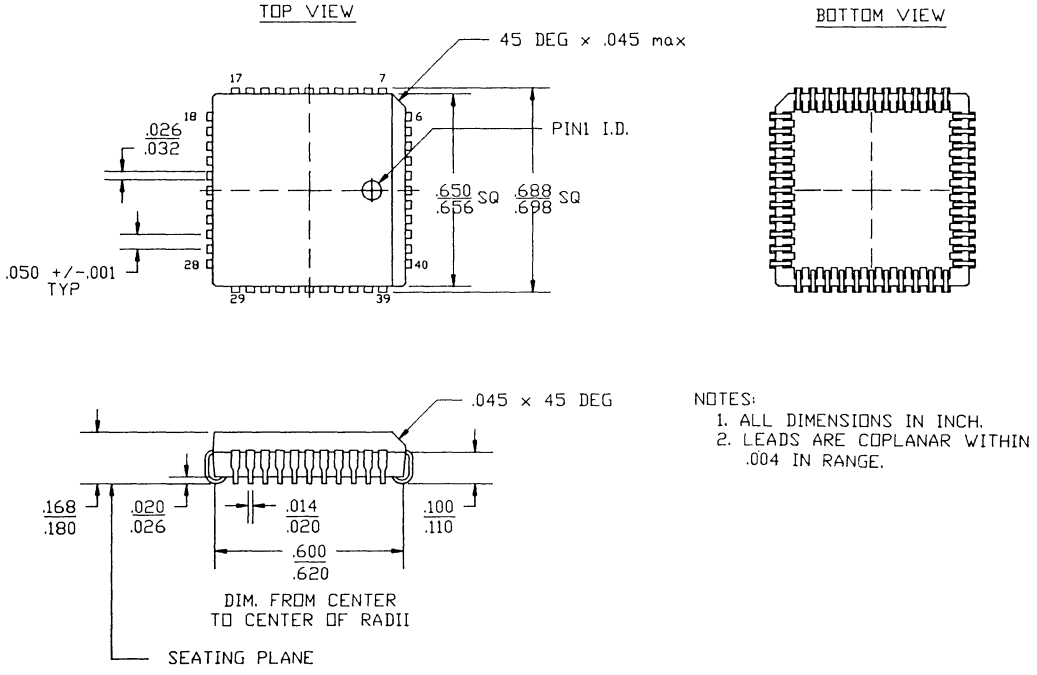


44-Lead Plastic Quad Flatpack (QFP)

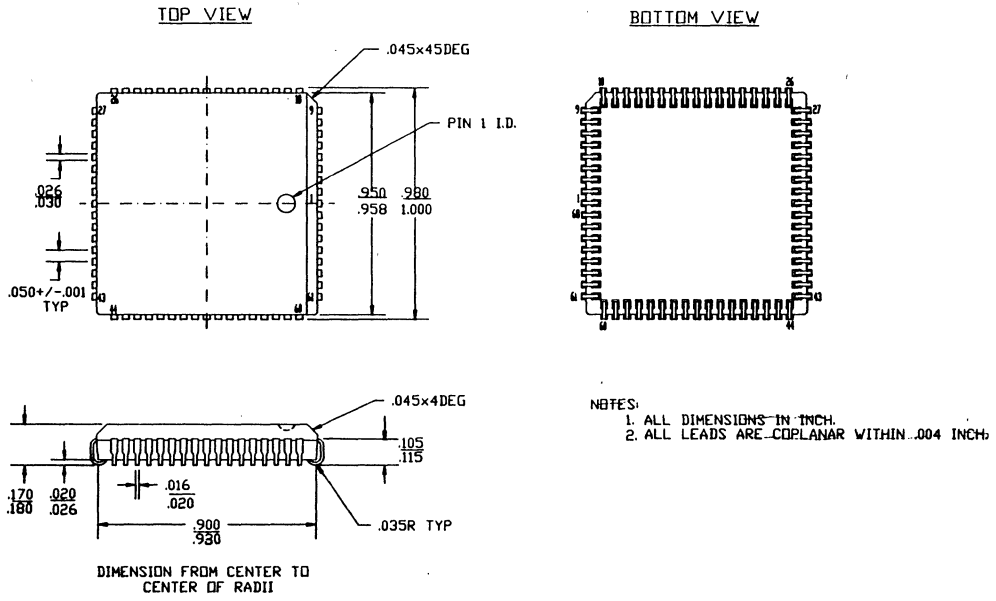


80-Lead Plastic Quad Flatpack (QFP)

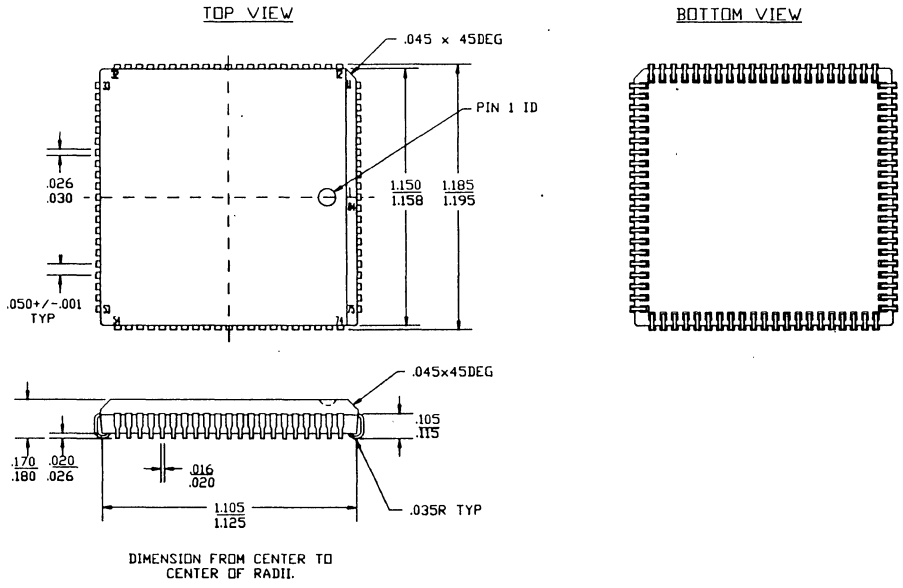
PACKAGE INFORMATION (Continued)



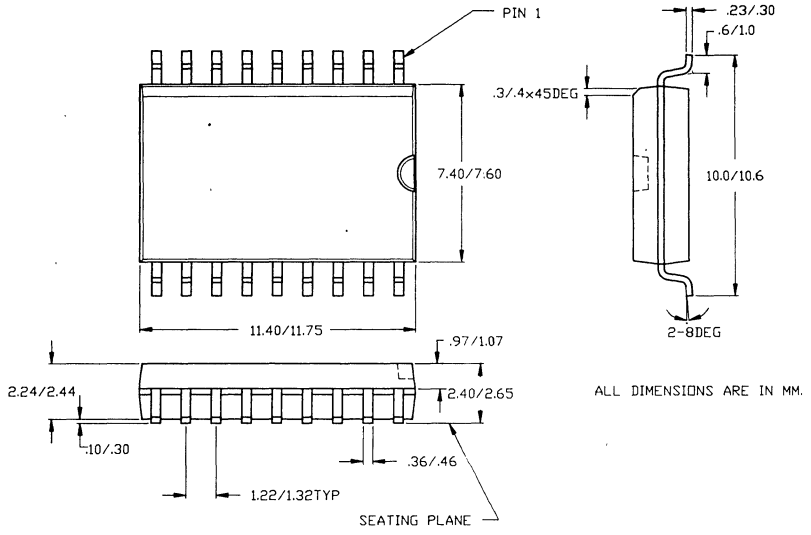
44-Lead Plastic Leaded Chip Carrier (PLCC)



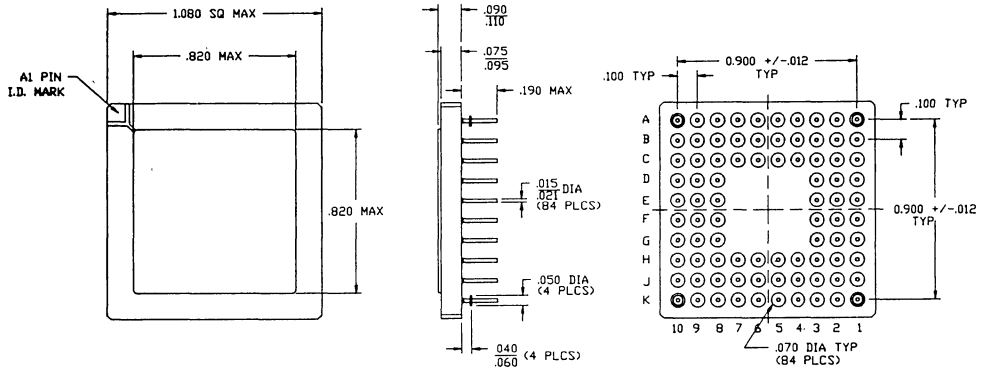
68-Lead Plastic Leaded Chip Carrier (PLCC)



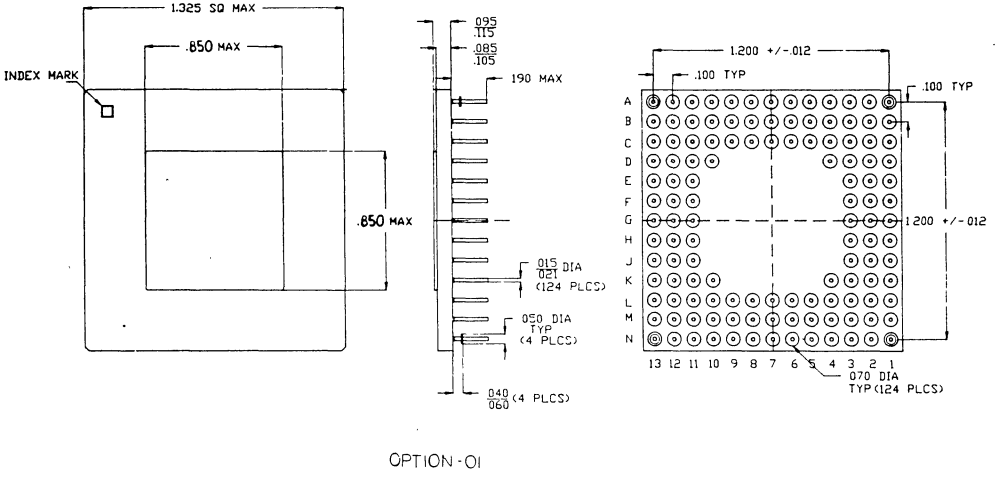
84-Lead Plastic Leaded Chip Carrier (PLCC)



18-Lead Small Outline Integrated Circuit (SOIC)

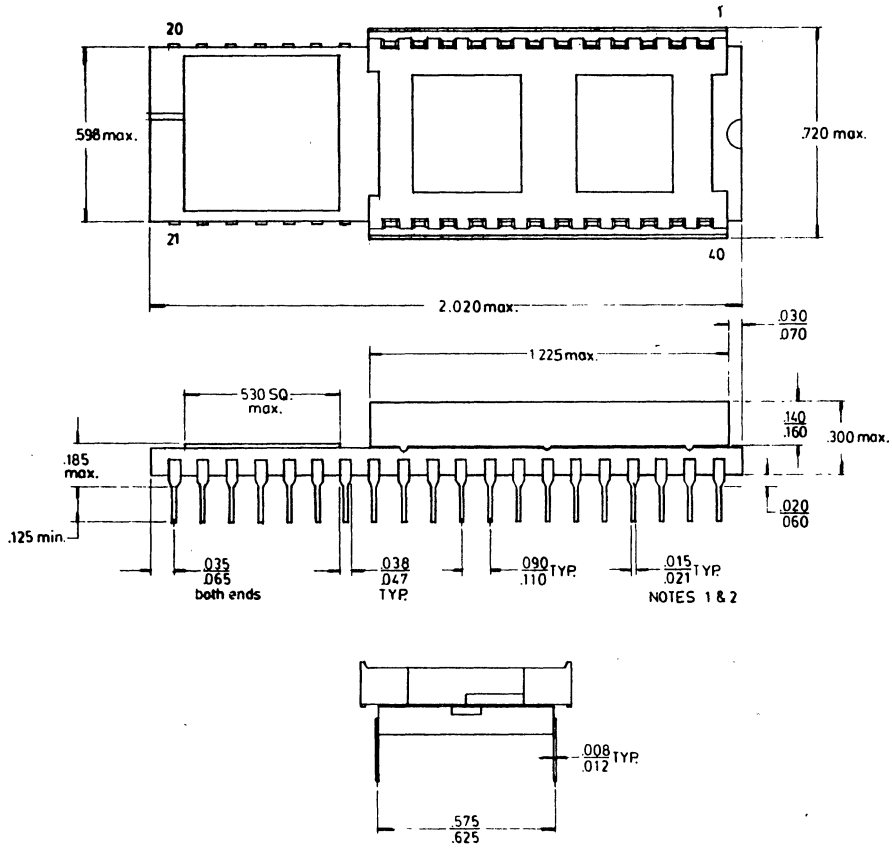


84-Lead Ceramic Pin Grid Array (10x10) (PGA)
Device Z86C12



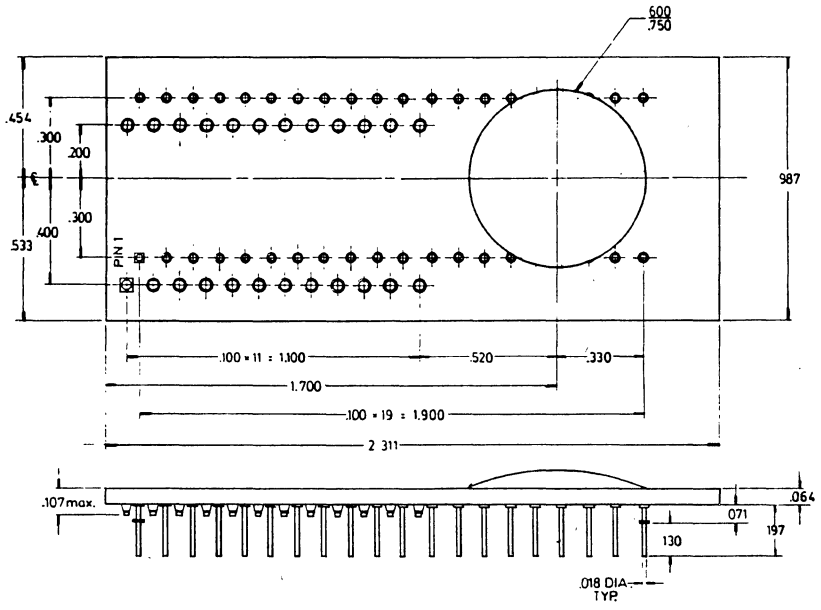
OPTION-01

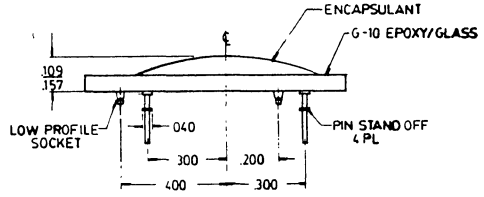
124-Lead Ceramic Pin Grid Array (PGA)



40-Lead Protopak

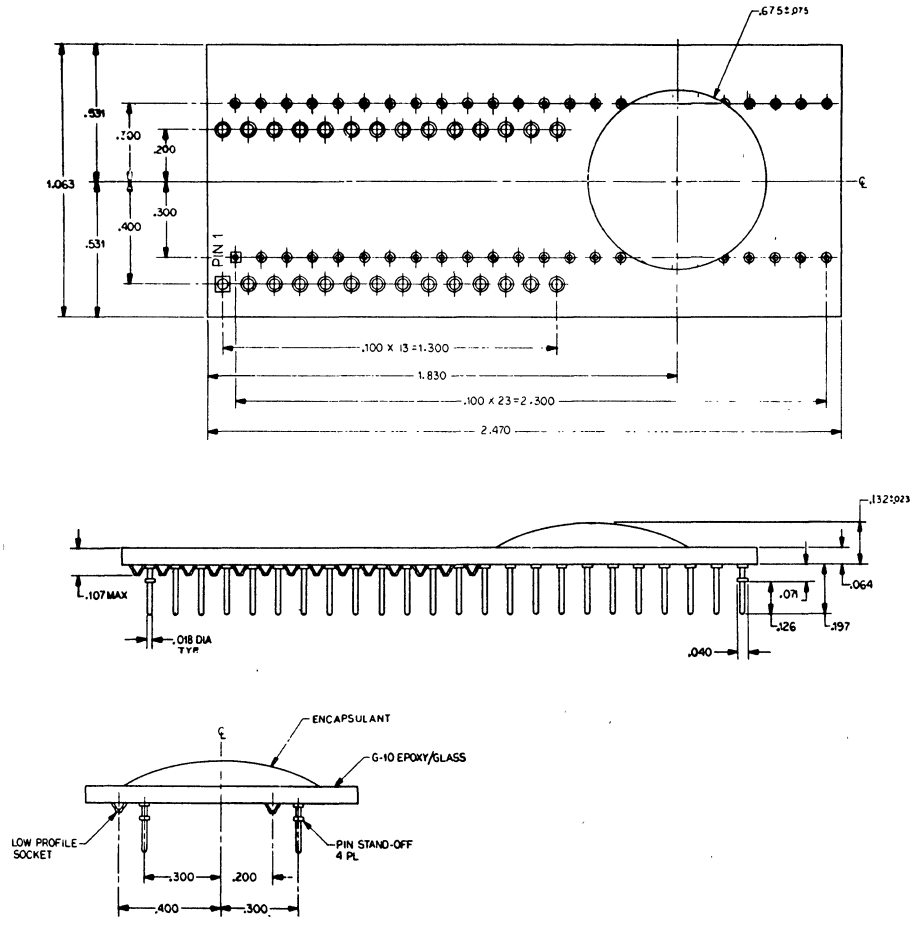
PACKAGE INFORMATION (Continued)





40/24-Lead Low Profile Protopak

PACKAGE INFORMATION (Continued)



48/28-Lead Low Profile Protopak

ORDERING INFORMATION

Support Products

Tool Box

Z0860000ZCO
Z0860000ZDP
Z0860200ZCO
Z0860200ZDP
Z86C0800ZCO
Z86C0800ZDP
Z86C0800ZEM
Z86E0800ZPR
Z86C1200ZDP
Z86C1900ZCO
Z86C1900ZEM
Z86E2100ZDP
Z86E2100ZDV
Z86E2100ZDF
Z86E3000ZDP
Z86E4000ZDP
Z86E4000ZDV
Z86E4000ZDF
Z86C2700ZCO
Z86C2700ZEM
Z0880000ZCO

Ice Box

Z86C1200ZEM
Z86C5000ZEM
Z86C9300ZEM
Z86C0000ZUSP064
Z86C0000ZUSP128
Z86C0000ZUSP256
Z86C1200ZPD
Z86C5000ZPD
Z86C9300ZPD

Z8 MCU, ICE Chip

Z0861212VSC
Z0861212CSE
Z86C1216GSE
Z86C5020GSE
Z0888420GSE

Z8 MCU, Protopak

Z0860312RSE
Z0861312RSE
Z0861312TSC
Z0882220TSF

Z8 MCU, NMOS, ROM

Z08600
28-pin DIP
Z0860008PSCRXXX
Z0860008PECRXXX

Z08601
40-pin DIP
Z0860108PSCRXXX
Z0860108PECRXXX

Z08601
44-pin PLCC
Z0860108VSCRXXX
Z0860108VECRXXX

Z08602
40-pin DIP
Z0860204PSCLXXX

Z08604
18-pin DIP
Z0860408PSCRXXX

Z08610
40-pin DIP
Z0861008PSCRXXX
Z0861008PECRXXX
Z0861012PSCRXXX

Z08611
40-pin DIP
Z0861108PSCRXXX
Z0861108PECRXXX
Z0861112PSCRXXX
Z0861112PECRXXX

Z08611
44-pin PLCC
Z0861108VSCRXXX
Z0861108VECRXXX
Z0861112VSCRXXX
Z0861112VECRXXX

Z8 MCU, BASIC/Debug Interpreter

Z08671
40-pin DIP
Z0867108PECR002

Z8 FDC, NMOS, ROMLESS

Z0765A
40-pin DIP
Z0765A08PSC

44-pin PLCC
Z0765A08VSC

Z8 MCU, NMOS, ROMLESS

Z08681
40-pin DIP
Z0868108PSC
Z0868108PEC
Z0868112PSC
Z0868112PEC

Z08681
44-pin PLCC
Z0868108VSC
Z0868108VEC
Z0868112VSC
Z0868112VEC

Z08691
40-pin DIP
Z0869108PSC
Z0869112PSC

Z08691
44-pin PLCC
Z0869108VSC
Z0869112VSC

ORDERING INFORMATION (Continued)

Z8 MCU, SUPER8

Z08800
48-pin DIP
Z0880020PSC
Z0882020PSCRXXX

Z08800
68-pin PLCC
Z0880020VSC
Z0882020VSCRXXX

Z08801
44-pin PLCC
Z0880120VSC
Z0882120VSCRXXX

Z88C00
48-pin DIP
Z88C0020PSC
Z88C0020PEC

Z88C00
68-pin PLCC
Z88C0020VSC
Z88C0020VEC

Z8 MCU, SUPER8 FORTH

48-pin DIP
Z0887520PSC

68-pin PLCC
Z0887520VSC

Z8 MCU, CMOS, ROM

Z86C00
28-pin DIP
Z86C0008PSCRXXX
Z86C0008PECRXXX
Z86C0012PSCRXXX
Z86C0012PECRXXX

Z86C06
18-pin DIP
Z86C0608PSCRXXX
Z86C0608PECRXXX
Z86C0612PSCRXXX
Z86C0612PECRXXX

Z86C06
18-pin SOIC
Z86C0612SSCRXXX
Z86C0612SECRXXX

Z86C08
18-pin DIP
Z86C0808PSCRXXX
Z86C0808PECRXXX
Z86C0812PSCRXXX
Z86C0812PECRXXX

Z86C08
18-pin SOIC
Z86C0812SSCRXXX
Z86C0812SECRXXX

Z86C09
18-pin DIP
Z86C0908PSCRXXX
Z86C0908PECRXXX
Z86C0912PSCRXXX
Z86C0912PECRXXX

Z86C09
18-pin SOIC
Z86C0912SSCRXXX
Z86C0912SECRXXX

Z86C10
28-pin DIP
Z86C1008PSCRXXX
Z86C1008PECRXXX
Z86C1012PSCRXXX
Z86C1012PECRXXX

Z86C11
40-pin DIP
Z86C1112PSCRXXX
Z86C1112PECRXXX
Z86C1116PSCRXXX

Z86C11
44-pin PLCC
Z86C1112VSCRXXX
Z86C1112VECRXXX

Z86C11
44-pin QFP
Z86C1112FSCRXXX
Z86C1112FECRXXX
Z86C1116FSCRXXX

Z86C19
18-pin DIP
Z86C1908PSCRXXX
Z86C1908PECRXXX
Z86C1912PSCRXXX
Z86C1912PECRXXX

Z86C19
18-pin SOIC
Z86C1912SSCRXXX
Z86C1912SECRXXX

Z86C20
40-pin DIP
Z86C2012PSCRXXX

Z86C21
40-pin DIP
Z86C2112PSCRXXX
Z86C2112PECRXXX
Z86C2116PSCRXXX

Z86C21
44-pin PLCC
Z86C2112VSCRXXX
Z86C2112VECRXXX

Z86C21
44-pin QFP
Z86C2112FSCRXXX
Z86C2112FECRXXX
Z86C2116FSCRXXX

Z86C30
28-pin DIP
Z86C3008PSCRXXX
Z86C3008PECRXXX
Z86C3012PSCRXXX
Z86C3012PECRXXX

Z86C40
40-pin DIP
Z86C4008PSCRXXX
Z86C4008PECRXXX
Z86C4012PSCRXXX
Z86C4012PECRXXX

Z86C40
44-pin PLCC
Z86C4008VSCRXXX
Z86C4008VECRXXX
Z86C4012VSCRXXX
Z86C4012VECRXXX

Z86C40
44-pin QFP
Z86C4008FSCRXXX
Z86C4012FSCRXXX
Z86C4012FECRXXX

Z86C61
40-pin DIP
Z86C6116PSCRXXX
Z86C6116PECRXXX

Z86C61
44-pin PLCC
Z86C6116VSCRXXX
Z86C6116VECRXXX

Z86C62
64-pin DIP
Z86C6216PSCRXXX
Z86C6216PECRXXX

Z86C62
68-pin PLCC
Z86C6216VSCRXXX
Z86C6216VECRXXX

Z8 SCSI, CMOS, ROMless

Z05380
40-pin DIP
Z0538010PSC

44-pin PLCC
Z0538010VSC

Z53C80
48-pin DIP
Z53C8003PSC

44-pin PLCC
Z53C8003VSC

Z8 MCU, CMOS, ROMless

Z86C89
40-pin DIP
Z86C8908PSC
Z86C8908PEC

Z86C89
44-pin PLCC
Z86C8908VSC
Z86C8908VEC

Z86C89
44-pin QFP
Z86C8908FSC
Z86C8908FEC

Z86C90
40-pin DIP
Z86C9008PSC
Z86C9008PEC
Z86C9012PSC
Z86C9012PEC

Z86C90
44-pin PLCC
Z86C9008VSC
Z86C9008VEC
Z86C9012VSC
Z86C9012VEC

Z86C90
44-pin QFP
Z86C9008FSC
Z86C9008FEC
Z86C9012FSC
Z86C9012FEC

Z86C91
40-pin DIP
Z86C9112PSC
Z86C9112PEC
Z86C9116PSC

Z86C91
44-pin PLCC
Z86C9112VSC
Z86C9112VEC
Z86C9116VSC

Z86C91
44-pin QFP
Z86C9112FSC
Z86C9112FEC
Z86C9116FSC

Z86C93
40-pin DIP
Z86C9320PSC

Z86C93
44-pin PLCC
Z86C9320VSC

Z86C93
44-pin QFP
Z86C9320FSC

Z86C94
80-pin QFP
Z86C9425FSC

Z86C94
84-pin PLCC
Z86C9425VSC

Z86C96
64-pin DIP
Z86C9620PSC

Z86C96
68-pin PLCC
Z86C9620VSC

Z8 MCU, CMOS, DTC

Z86C27, ROM
64-pin DIP
Z86C2704PSCRXXX

Z86C97, ROMLESS
64-pin DIP
Z86C9704PSCXXXX

Z86127
64-pin DIP
Z8612704PSCRXXX

Z86128
18-pin DIP
Z8612812PSC

Z8 MCU, OTP

Z86E08
18-pin DIP
Z86E0812PSC

Z86E30
28-pin DIP
Z86E3012PSC
Z86E3012KSE

Z86E21
40-pin DIP
Z86E2112PSC
Z86E2116PSC
Z86E2116KSE

Z86E21
44-pin PLCC
Z86E2112VSC
Z86E2116VSC

Z86E21
44-pin QFP
Z86E2112FSC
Z86E2116FSC

Z86E22
40-pin DIP
Z86E2204PSC

Z86E40
40-pin DIP
Z86E4012PSC
Z86E4012KSE

Z86E40
44-pin PLCC
Z86E4012VSC

Z86E40
44-pin QFP
Z86E4012FSC

PACKAGE

PREFERRED
D = Cerdip
P = Plastic DIP
V = Plastic Leaded Chip Carrier

LONGER LEAD TIME
C = Ceramic Sidebrazed
E = Ceramic Window Lid
F = Plastic Quad Flatpack
G = Ceramic PGA (Pin Grid Array)
K = Cerdip Window Lid
L = Ceramic LCC (Leadless Chip Carrier)
R = Ceramic Protopak
S = SOIC (Small Outline Integrated Circuit)
T = Low Profile Protopack

ENVIRONMENTAL

PREFERRED
C = Plastic Standard
E = Hermetic Standard
F = Protopack Standard

LONGER LEAD TIME
A = Hermetic Stressed
B = 883 Class B Military
D = Plastic Stressed
J = JAN 38510 Military

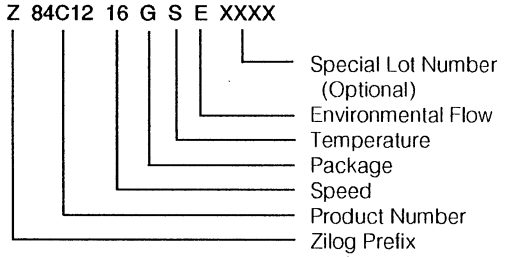
TEMPERATURE

PREFERRED
S = 0°C to +70°C

LONGER LEAD TIME
E = -40°C to +100°C
M = -55°C to +125°C

EXAMPLE

Z86C1216GSE is a CMOS 86C12, 16 MHz, PGA, 0°C to +70°C, Hermetic Standard Flow.



ZILOG DOMESTIC SALES OFFICES AND TECHNICAL CENTERS

CALIFORNIA
Agoura 818-707-2160
Campbell 408-370-8120
Tustin 714-838-7800

COLORADO
Boulder 303-494-2905

FLORIDA
Largo 813-585-2533

GEORGIA
Norcross 404-448-9370

ILLINOIS
Schaumburg 708-517-8080

NEW HAMPSHIRE
Nashua 603-888-8590

NEW JERSEY
Clark 201-382-5700

NORTH CAROLINA
Raleigh 919-790-7706

OHIO
Independence 216-447-1480

PENNSYLVANIA
Ambler 215-653-0230

TEXAS
Dallas 214-987-9987

WASHINGTON
Seattle 206-523-3591

INTERNATIONAL SALES OFFICES

CANADA
Toronto 416-673-0634

GERMANY
Munich 49-89-672-045
Sömmerda 37-626-23906

JAPAN
Tokyo 81-3-3587-0528

HONG KONG
Kowloon 852-7238979

KOREA
Seoul 82-2-552-5401

SINGAPORE
Singapore 65-2357155

TAIWAN
Taipei 886-2-741-3125

UNITED KINGDOM
Maidenhead 44-628-392-00

Zilog, Inc. 210 E. Hacienda Ave., Campbell, CA 95008-6600, Tel: (408) 370-8000, FAX: 408-370-8056/8027

DC-8275-04