SDK-86
MCS-86 SYSTEM DESIGN KIT
USER'S GUIDE

Manual Order No. 9800698A

SDK-86
MCS-86 SYSTEM DESIGN KIT
USER'S GUIDE

Manual Order No. 9800698A

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051

Additional copies of this manual or other Intel literature may be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

Printed in U.S.A./A41/479/7.5K/CP

SERVICE ASSISTANCE

If, following assembly, you cannot get your kit to operate satisfactorily, the Intel Technical Support Center "Service Hotline" is available for assistance. This service is provided during the hours of 8AM to 5PM (Pacific Time), Monday through Friday. The toll-free Hotline telephone numbers are:

(800) 538-8014 or (800) 538-8015 when dialing from outside the state of California

(800) 672-3507 when dialing from within the state of California

The Hotline is intended expressly to help you get your kit running and is *not* intended to assist you in circuit designs or applications. Telephone assistance is limited to *one* call per problem. If a problem cannot be remedied over the telephone, you may, at your discretion, return your assembled kit to Intel for repair. To return your kit, a Return Authorization Number *must be obtained* from the Technical Support Center prior to sending in your kit. Also, either a purchase order number for the repairs must be furnished to the center or a money order (no personal checks please) must be included with the kit being returned. Repairs resulting from defective components supplied with your kit will be done at no charge, and all prepayments will be refunded. Repairs necessitated as a result of customer error, damage or misuse will be billed at a fixed, flat-rate charge which will be quoted by the Technical Support Center.

NOTE

The Technical Support Center will not repair an SDK-86 Kit that has been modified and, when circuitry has been added to the user design area, may request that the circuitry be disconnected prior to submitting the kit to the center for repair.

PREFACE

This is the user's guide for the MCS-86 System Design Kit (SDK-86). Included within this guide are descriptions of the two ROM-resident monitor programs that allow you to communicate with your kit, a functional description of the SDK-86 itself, interfacing information and design examples for expanding your kit's capabilities. Additional information is available in the following Intel publications:

- **SDK-86 MCS-86 System Design Kit Assembly Manual,* Order No. 9800697
- **SDK-86 MCS-86 System Design Kit Monitor Listings,* Order No. 9800699
- **MCS-86 User's Manual,* Order No. 9800722
- *ASM86 Language Reference Manual,* Order No. 9800640
- *MCS-86 Software Development Utilities Operating Instructions for ISIS-II Users,* Order No. 9800639
- *MCS 80/85 Absolute Object File Formats Technical Specification,* Order No. 9800183
- *Application of Intel's 5V EPROM and ROM Family for Microprocessor Systems,* Application Note AP-30

*Supplied with SDK-86 Kit

CONTENTS

TABLES

ILLUSTRATIONS

CHAPTER 1
GENERAL INFORMATION

1-1. Introduction

This is the user's guide for the *MCS-86 System Design Kit*. Now that you have completed the assembly and initial checkout of your SDK-86, this guide will provide insight into the operation and function of the kit and also will explore circuit designs that can be implemented either off-board or within the user design area to increase or expand your kit's capabilities. Included within this guide are:

- A description of the SDK-86 circuitry.
- Functional descriptions of the two ROM-resident monitor programs with command examples.
- A description of the "loader" program that is used to perform upload/download program transfers between SDK-86 memory and an Intel microcomputer development system.
- Interfacing and board configuration information for connecting peripheral devices to the SDK-86.
- A complete schematic set for the kit.
- Appendices describing circuit designs that can be implemented in the user design area including program memory expansion using Intel 2708, 2716 and 2732 EPROM devices.

1-2. Specifications

Central Processor

CPU: 8086
Clock Frequency: 2.5 MHz or 5 MHz (jumper selectable)
Instruction Cycle Time: 800 ns (5 MHz)

Memory Type

ROM: 8K bytes 2316/2716
RAM: 2K bytes (expandable to 4K bytes) 2142

Memory Addressing

ROM: FE000 through FFFFF
RAM: 0 through 7FF (0-FFF with 4K bytes)

Input/Output

Parallel: 48 lines (two 8255A's)
Serial: RS232 or current loop (8251A)
Baud rate selectable from 110 to 4800 baud

Interface Signals

CPU Bus: All signals TTL compatible
Parallel I/O: All signals TTL compatible
Serial I/O: 20 mA current loop or RS232

Interrupts

External: Maskable and Non-Maskable, Interrupt vector 2 reserved for non-maskable interrupt (NMI).
Internal: Interrupt vectors 1 (single-step) and 3 (breakpoint) reserved by monitor.

DMA

Hold Request: Jumper selectable, TTL compatible input.

Software

System Monitors: Preprogrammed 2316 or 2716 ROMs
Addresses: FE000 through FFFFF
Monitor I/O: Keypad and Serial (teletypewriter or CRT)

Literature Supplied

- SDK-86 Assembly Manual
- SDK-86 User's Guide
- SDK-86 Monitor Listings
- MCS-86 User's Manual

Physical Characteristics

Length: 34.3 cm (13.5 in.)
Width: 30.5 cm (12.0 in.)
Height: 4.4 cm (1.75 in.)
Weight: Approximately 0.7 kg (1.5 lbs)

Power Requirements

V_{CC}: +5 volts (±5%), 3.5 Amperes
V_{TTY}: −12 volts (±10%), 0.3 Amperes (required only if teletypewriter or CRT terminal connected to serial interface port)

Environmental

Operating Temperature: 0 to 40°C

Accessories Kit (SDK-C86)

- SDK to microcomputer development system interface cable.
- Single-density diskette containing "Loader" program and I/O routine library
- Double-density diskette containing "Loader" program and I/O routine library

CHAPTER 2
FUNCTIONAL DESCRIPTION

2-1. Introduction

This chapter examines the circuits which make up the SDK-86. Also presented in this chapter are the I/O port address assignments for the various devices on the board and a definition of ROM and RAM memory allocated to the SDK-86. The individual circuits can be divided into logical blocks as shown in the detailed block diagram (Figure 2-2) at the conclusion of this chapter. Note that the block diagram can be folded out for viewing while reading the circuit descriptions. The individual blocks that will be described in this chapter are as follows:

- Clock Generator
- Wait State Generator
- CPU
- Parallel I/O Ports
- RAM
- PROM
- I/O Decode
- Off Board Decode
- Keypad/Display
- Serial Interface
- Bus Expansion

2-2. Clock Generator

The clock generator circuit is an Intel 8284 clock generator/driver. The circuit accepts a crystal input which operates at a fundamental frequency of 14.7456 MHz. (14.7456 MHz was selected since this frequency is a multiple of the baud rate clock and also provides a suitable frequency for the CPU.) The clock generator/driver divides the crystal frequency by three to produce the 4.9 MHz CLK signal required by the CPU. Additionally, the clock generator performs a further divide-by-two output called PCLK (peripheral clock) which is the primary clock signal used by the remainder of the circuits.

The clock generator/driver provides two control signal outputs which are synchronized (internally) to the 4.9 MHz CLK signal; RDY (ready) and RST (reset). RST is used to reset the SDK-86 to an initialized state and occurs when the $\overline{\text{RES}}$ input goes low (when power first is applied or when the **SYSTM RESET** key is pressed). The RDY output is active (logically high) when the RDY1 input from the wait state generator is active. As will be explained in the next section, the RDY1 input is active whenever on board memory is addressed or when a selected number of "wait states" occurs.

A jumper link (W40/W41) is provided for low-speed (2.45 MHz) operation and allows the PCLK signal to be routed to the CLK input of the CPU in place of the CLK output from the clock generator/driver. Note that this is the clock configuration that was selected when the SDK-86 kit was assembled and that this configura-

tion allows on-board memory and I/O operations to be performed without the necessity of wait states. When 8086 operation at the 5 (4.9) MHz rate is desired, simply remove the shorting plug from W40 and install it at W41. For additional information regarding the operation of the 8284 clock generator/driver, refer to the *MCS-86 User's Manual*.

NOTE

The SDK-86 kit may be supplied with an 8086 CPU that has a maximum clock input frequency of 4 MHz and therefore only can be operated with the 2.45 MHz clock (shorting plug installed at W40). These 4 MHz versions of the 8086 are noted by a "−4" suffix following the circuit number stamped on the top of the component (i.e., 8086-4).

2-3. Wait State Generator

The wait state generator circuit allows wait states to be inserted into the CPU's bus cycle to compensate for a "slow" peripheral I/O or memory circuit that has been interfaced to the expansion bus or to allow on-board memory and I/O operations to function correctly when the CPU is operated at a 5 MHz rate. (The SDK-86 memory and I/O devices do not require wait states when the 2.45 MHz clock is used.) When one or more wait states are required for proper CPU operation, the shorting plug originally installed at W27 (0 wait states) is removed and is positioned over the header pins corresponding to the number of wait states to be inserted into the bus cycle according to the following table.

Table 2-1. Wait State Selection

Plug Position	Wait States	Plug Position	Wait States
W27*	0	W31	4
W28	1	W32	5
W29	2	W33	6
W30	3	W34	7

*Omitting the wait state shorting plug is equivalent to 0 wait states.

Referring to the block diagram (Figure 2-2), the wait state generator is cleared following every read, write or interrupt cycle and subsequently is enabled at the beginning of the next read, write or interrupt cycle when the CLR input to the wait state generator goes inactive. When enabled, the wait state generator (a 74LS164 shift register) begins to shift a "one" through the register. When the selected number (0-7) of shifts have occurred, the jumpered output goes active which, coupled to the RDY1 input of the clock generator/driver, causes the clock generator's RDY output to the CPU to go active.

When a wait state is selected, on-board I/O operations are subject to the number of wait states selected, while on-board memory operations are performed without the wait state restriction. As shown on the block diagram, the RDY1 input to the clock generator/driver originates either from the wait state generator or from a three-input AND gate. The output from the gate is active during on-board memory read and write operations (M/$\overline{IO}$ and $\overline{OFF\ BOARD}$ high). The jumper link (W39) at one input to the AND gate, when installed, permanently disables the gate and causes on-board memory operations to be subject to the number of wait states selected. Note that it may be necessary to install this link if "slow" ROM or RAM on-board memory is added.

2-4. CPU

Information on the operation, function and instruction set of the 8086 CPU is contained in the *MCS-86 User's Manual* and is not repeated in this publication. Several points regarding the implementation of the 8086 CPU in the SDK-86 kit should be noted.

- The 8086 is used in the minimum mode (MN/$\overline{MX}$ input held logically high).
- The INTR, TEST and HOLD inputs to the 8086 were disabled by the shorting plugs installed in header W36-W38 when the kit was assembled. When a peripheral circuit is interfaced to the bus expansion logic and requires the use of any of these signals, the corresponding shorting plugs must be removed.
- When the SDK-86 is reset, the 8086 executes the instruction at location FFFF0H. The instruction at this location is an inter-segment direct jump to the beginning of the monitor program that resides in ROM locations FF000H to FFFFFH.
- The 8086's NMI (non-maskable interrupt) input originates from the **INTR** key and, when pressed, causes the CPU to save the current system status (pushing the IP, CS and FL register contents onto the stack after clearing the IF and TF flags) and to perform an indirect long jump through RAM location 08H (interrupt vector 2). Note that locations 08H through 0BH are initialized by the monitor on reset to contain the address of the monitor's interrupt routine.
- The maskable interrupt (INTR) is not used by the SDK-86, but is available to peripheral circuits through the expansion bus. To use the maskable interrupt, an interrupt vector pointer must be provided on the data bus when $\overline{INTA}$ is active. When there is more than one source for an interrupt, an interrupt priority resolution circuit must be provided (e.g., an Intel 8259A).

2-5. Parallel I/O Ports

The parallel I/O ports consist of two Intel 8255A programmable peripheral interface circuits. These two circuits each contain three 8-bit input/output data ports (designated ports A, B and C) and one write-only control port. The 8255A circuit that interfaces with the low-order data byte (D0-D7) is designated port 2 (P2), and the 8255A circuit that interfaces with the high-order data byte (D8-D15) is designated port 1 (P1). All ports can be addressed individually (e.g., port P1A, P2C, P1 Control, etc.) or a corresponding pair of ports (P1A and P2A, P1B and P2B or P1C and P2C) can be addressed simultaneously to form a single 16-bit wide data port.

The I/O port address assignments for the two parallel I/O port circuits are defined in the following table.

Table 2-2. Parallel I/O Port Addresses

Port	Address	Port	Address
P2A	FFF8	P1A	FFF9
P2B	FFFA	P1B	FFFB
P2C	FFFC	P1C	FFFD
P2 Control	FFFE	P1 Control	FFFF

During byte operations, the I/O decode logic generates the appropriate $\overline{CS}$ (chip select) input ($\overline{HIGH\ PORT\ SELECT}$ or $\overline{LOW\ PORT\ SELECT}$), while during word operations, the P2 port address is used to address the desired pair of ports, and the

I/O decode logic generates both $\overline{\text{HIGH PORT SELECT}}$ and $\overline{\text{LOW PORT SELECT}}$ coincidently. For detailed information on the operation and programming of the 8255A, refer to Appendix B.

2-6. RAM

The standard SDK-86 includes 2K bytes of random access memory (RAM) and on-board provision for the installation of an additional 2K bytes. RAM is located at the low end of memory (locations 0H through 07FFH or 0H through 0FFFH). The memory addressing arrangement of the 8086 CPU allows simultaneous reading or writing of a full 16 bits (two adjacent byte locations) or the reading or writing of either the high (D8-D15) or low (D0-D7) byte.

RAM is enabled by the appropriate output(s) from RAM decode PROM A29 and by a high level on the M/$\overline{\text{IO}}$ control line (memory operation). The PROM (A29) is enabled by the inactive (low) state of the A19 address bit (indicating an address below 80000H) at its $\overline{\text{CS1}}$ input. The PROM decodes $\overline{\text{BHE}}$ and the A0 and A11 through A18 address bits. When the A12 through A18 address bits are inactive, an address below 1000H is indicated. The A11 address bit determines if the address is between 0H and 07FFH (A11 inactive) or between 0800H and 0FFFH (A11 active). The A0 address bit and $\overline{\text{BHE}}$ determine the byte or bytes enabled. The following table defines the PROM decoding.

Table 2-3. PROM A29 Decoding

PROM Inputs				PROM Outputs*				Byte(s) Selected (Address Block)
A12-A18	A11	$\overline{\text{BHE}}$	A0	O4	O3	O2	O1	
0	0	0	0	1	1	0	0	Both Bytes (0H-07FFH)
0	0	0	1	1	1	0	1	High Byte (0H-07FFH)
0	0	1	0	1	1	1	0	Low Byte (0H-07FFH)
0	1	0	0	0	0	1	1	Both Bytes (0800H-0FFFH)
0	1	0	1	0	1	1	1	High Byte (0800H-0FFFH)
0	1	1	0	1	0	1	1	Low Byte (0800H-0FFFH)
All other states				1	1	1	1	None

* A "0" on the output enables the corresponding RAM byte.

RAM itself is addressed by the A1 through A10 address bits and is controlled (read or write) by the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ signals.

The first 256 (decimal) byte locations of RAM (locations 0H through 0FFH) are reserved by the monitor program. The following illustration shows the actual monitor allocations within this block.

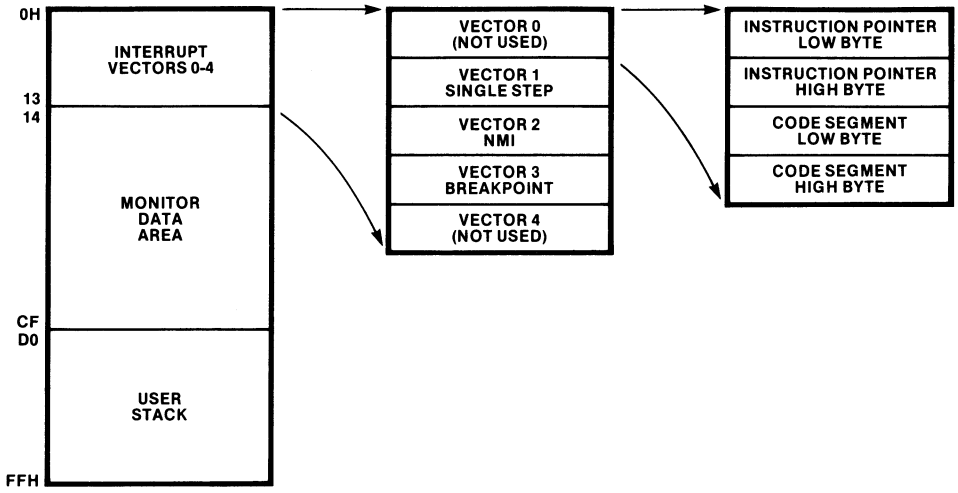


Figure 2-1. Monitor Memory Allocation

2-7. PROM

The SDK-86 kit includes 8K bytes of PROM; 4K bytes (in two 2K × 8, 2316 ROM or 2716 EPROM devices) containing the keypad monitor program and 4K bytes (in another set of 2K × 8 ROMs or EPROMs) containing the serial monitor program. The 8K bytes of PROM are located at the high end of memory (locations FE000H through FFFFFH). The principle system monitor program (keypad or serial) resides in locations FF000H through FFFFFH (sockets A27 and A30). The remaining 4K locations at sockets A36 and A37 (locations FE000H through FEFFFH) can be used for the other monitor program or for user PROM or ROM program storage.

PROM is enabled by the corresponding output from PROM decode PROM A26 and by the $\overline{BHE}$ and address bit A0 inputs. PROM A26 decodes the M/$\overline{IO}$ signal and the A12 through A19 address bits to determine which 4K bytes (FE000H-FEFFFH or FF000H-FFFFFH) are to be addressed, while the $\overline{BHE}$ and A0 inputs enable either high- or low-order byte or word (both bytes) output.

Referring to sheet 1 of the SDK-86 schematic diagram in Chapter 7, two additional PROM enable outputs (not shown on the block diagram) are provided. These outputs ($\overline{CSX}$ and $\overline{CSY}$) are routed to bus expansion connector J2 to permit two additional 4K bytes of PROM or ROM (at locations FC000H-FCFFFH and FD000H-FDFFFH) to be incorporated without requiring external address decoding logic.

Table 2-4 defines the PROM's address decoding.

Table 2-4. PROM A26 Decoding

PROM Inputs				PROM Outputs*				PROM Address Block Selected
M/$\overline{IO}$	A14-A19	A13	A12	O4	O3	O2	O1	
1	1	1	1	1	1	1	0	FF000H-FFFFFH
1	1	1	0	1	1	0	1	FE000H-FEFFFH
1	1	0	1	1	0	1	1	FD000H-FDFFFH ($\overline{CSX}$)
1	1	0	0	0	1	1	1	FC000H-FCFFFH ($\overline{CSY}$)
All other states				1	1	1	1	None

* A "0" on the output enables the corresponding PROM address block.

2-8. I/O Decode

I/O decode PROM A22 is enabled by a low level M/$\overline{IO}$ input (indicating an I/O operation) and an address between FE00H and FFFFH (address bits A11 through A15 active). When enabled, the PROM decodes $\overline{BHE}$ and the A0 and A3 through A10 address bits to generate the corresponding I/O port select signal(s) according to Table 2-5.

Table 2-5. PROM A22 Decoding

PROM Inputs					PROM Outputs*			
A5-A10	A4	A3	$\overline{BHE}$	A0	$\overline{O4}$ HIGH PORT SELECT	$\overline{O3}$ LOW PORT SELECT	$\overline{O2}$ USART SELECT	$\overline{O1}$ KDSEL
1	0	1	0	0	1	1	1	0
1	0	1	1	0	1	1	1	0
1	1	0	0	0	1	1	0	1
1	1	0	1	0	1	1	0	1
1	1	1	0	0	0	0	1	1
1	1	1	0	1	0	1	1	1
1	1	1	1	0	1	0	1	1
All other states					1	1	1	1

* A "0" on the output selects the corresponding I/O device.

Table 2-6 defines the individual I/O port address assignments.

Table 2-6. I/O Port Address Assignments

Port Address	Port Function
0000 to FFDF	Open
FFE8	Read/Write 8279 Display RAM or Read 8279 FIFO
E9	
EA	Read 8279 Status or Write 8279 Command
EB	
EC	Reserved
ED	
EE	Reserved
FFEF	
FFF0	Read/Write 8251A Data
F1	
F2	Read 8251A Status or Write 8251A Control
F3	
F4	Reserved
F5	
F6	Reserved
FFF7	
FFF8	Read/Write 8255A Port P2A
F9	Read/Write 8255A Port P1A
FA	Read/Write 8255A Port P2B
FB	Read/Write 8255A Port P1B
FC	Read/Write 8255A Port P2C
FD	Read/Write 8255A Port P1C
FE	Write 8255A P2 Control
FFFF	Write 8255A P1 Control

2-9. Off Board Decode

The off board decode logic is responsible for the generation of the $\overline{\text{OFF BOARD}}$ signal. This signal is used by the wait state generator to force off-board memory operations to be subjected to the number of wait states selected and also is responsible for the generation of the $\overline{\text{BUFFER ON}}$ signal that is used to enable the data transceivers on the expansion bus during both I/O operations and off-board memory accesses.

Off board decode PROM A12 accepts the A12 through A19 address bits and the M/$\overline{\text{IO}}$ signal to generate the $\overline{\text{OFF BOARD}}$ signal when off-board memory locations are addressed. Table 2-7 defines the PROM decoding. Note that the table shows the inactive states in which the PROM does *not* generate the $\overline{\text{OFF BOARD}}$ signal.

Table 2-7. PROM A12 Decoding

PROM Inputs									PROM Output	Corresponding
M/$\overline{\text{IO}}$	A19	A18	A17	A16	A15	A14	A13	A12	(O1)	Address Block
1	0	0	0	0	0	0	0	0	1 (inactive)	0H-0FFFH (On-Board RAM)
1	1	1	1	1	1	1	1	0	1 (inactive)	FE000H-FEFFFH (On-Board PROM)
1	1	1	1	1	1	1	1	1	1 (inactive)	FF000H-FFFFFH (On-Board PROM)
All other states									0 (active)	01000H-FDFFFH (Off-Board)

The $\overline{\text{OFF BOARD}}$ signal also is generated when an off-board I/O port is addressed (port addresses 0H through 0FFDFH). During an I/O operation, the gating shown above the PROM on the block diagram is enabled by the low state M/$\overline{\text{IO}}$ signal and generates the $\overline{\text{OFF BOARD}}$ signal if any of the A5 through A15 address bits is at a low state (an address below 0FFE0H).

2-10. Keypad/Display

The keypad/display logic is centered around an Intel 8279 Keyboard/Display Controller. The 8279 is responsible for the debouncing of the keys, the coding of keypad matrix and the refreshing of the display elements.

As stated in Section 2-8, the 8279 occupies two I/O ports within the on-board I/O address space and, since the 8279 is interfaced to the lower byte of the data bus (D0-D7), both ports have even-numbered port addresses (FFE8H and FFEAH). The individual 8279 port functions are determined by the A1 address bit and the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ signals as noted in Table 2-8.

Table 2-8. Keypad/Display I/O Ports

8279 Input			Port Address	Port Function
A1	$\overline{\text{RD}}$	$\overline{\text{WR}}$		
0	0	1	FFE8H	Read Display RAM or Keyboard FIFO
0	1	0	FFE8H	Write Display RAM
1	0	1	FFEAH	Read Status
1	1	0	FFEAH	Write Command

Since the A2 address bit is not decoded by the I/O decode PROM, port addresses FFECH and FFEEH are decoded as port addresses FFE8H and FFEAH, respectively. As noted in Table 2-6, these two additional addresses are reserved.

The monitor program, through the write command port, arranges the 8279 as follows:

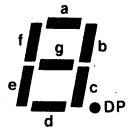
- Eight digit, 8-bit, left entry display and encoded scan keyboard with 2-key lockout (keyboard/display mode set command; byte value 00H).
- A prescale factor of 25 to provide 5 ms keyboard and display scan times and a 10 ms debounce time (program clock command; byte value 039H).

For information regarding the operation and programming of the 8279, refer to Appendix B.

Referring again to the block diagram, since the encoded scan keyboard mode is specified, the SL0 through SL3 outputs represent a binary count. The 7445 (a BCD to 10-line decoder) converts the binary count to eight, single-line outputs (two outputs are unused) which are used to enable the individual display elements. The SL0 through SL2 outputs additionally are routed to a 74LS156 (configured as a 3- to 8-line decoder) to provide the three row scan signal inputs to the keypad switch matrix. The outputs from the switch matrix, at the RL0 through RL7 inputs to the 8279, represent the eight switch columns (see sheet 7 of the SDK-86 schematic). When a switch is pressed while its row is being scanned, the corresponding column is enabled. The 8279 uses the enabled column bit value and its row scan value (SL0-SL2) to generate a 6-bit code representing the key pressed. This is the code that is stored in the FIFO and accessible to the CPU through the read keyboard FIFO I/O port.

The A0 through A3 and B0 through B3 outputs from the 8279 form a single 8-bit parallel output containing the individual segment enable bits. Table 2-9 defines the correlation between the individual bits and the display element segments (a high level output turns on the corresponding display segment).

Table 2-9. Segment Definition

Display Segments	8279 Output Bit	Segment Enabled	8279 Output Bit	Segment Enabled
	A0	e	B0	a
	A1	f	B1	b
	A2	g	B2	c
	A3	DP	B3	d

2-11. Serial Interface

The serial interface is based on an Intel 8251A USART (Universal Synchronous/Asynchronous Receiver/Transmitter) which is operated in the asynchronous mode. An associated baud rate generator provides jumper-selectable baud rates ranging from 75 to 4800 baud for use by the USART. The serial port jumper matrix is used to configure the USART's input/output signals at serial interface connector J7 for "stand alone" or "MDS slave" operation with either teletypewriter (20 mA current loop) or CRT terminal (RS232) interface capability.

The monitor program, by writing a mode command of 0CFH to the USART's control port, configures the USART for:

- 8-bit character length
- Parity disabled
- Two stop bits
- Baud rate factor of 64x

The baud rate generator (a 74LS393 dual 4-bit binary counter) uses the 614.4 kHz signal (PCLK/4) to provide a series of baud rate frequencies. Since the USART is operated in the 64x mode, these frequencies are 64 times the corresponding baud rate. Table 2-10 defines the baud rate selection and corresponding baud rate generator output frequency.

Table 2-10. Baud Rate Selection

Baud Rate	Shorting Plug Position	Output Frequency
4800	W25	307.2 kHz
2400	W24	153.6 kHz
1200	W23	76.8 kHz
600	W22	38.4 kHz
300	W21	19.2 kHz
150	W20	9.6 kHz
75	W19	4.8 kHz
110	W20, W26	6.98 kHz

Referring to sheet 9 of the SDK-86 schematic, the 110 baud rate is derived from the 76.8 kHz signal at the 2A input to the second binary counter. When the shorting plug at W26 is installed, the NAND gate on the 2QA, 2QB and 2QD outputs clears the second binary counter on the count of eleven to provide the required 6.98 kHz signal at the 2QC output.

The USART occupies two I/O ports within the on-board I/O address space and, since the USART is interfaced to the lower byte of the data bus (D0-D7), both ports have even-numbered port addresses (FFF0H and FFF2H). The individual USART port functions are determined by the A1 address bit and the $\overline{RD}$ and $\overline{WR}$ signals as noted in Table 2-11.

Table 2-11. USART I/O Ports

USART Input			Port Addresses	Port Function
A1	$\overline{RD}$	$\overline{WR}$		
0	0	1	FFF0H	Read USART Data
0	1	0	FFF0H	Write USART Data
1	0	1	FFF2H	Read USART Status
1	1	0	FFF2H	Write USART Control

Since the A2 address bit is not decoded by the I/O decode PROM, port addresses FFF4H and FFF6H are decoded as port addresses FFF0H and FFF2H, respectively. As noted in Table 2-6, these two additional addresses are reserved.

The serial port jumper matrix determines the individual pin assignments of the USART's input/output signals at connector J7 and the signal definition (current loop or RS232) and polarity. Table 2-12 defines the shorting plugs to be installed for the various interface configurations.

Table 2-12. Serial Port Jumper Matrix

Interface Configuration	Shorting Plugs Installed	PC Board Silkscreen
STAND ALONE CRT TERMINAL	W1-W5	⌊ CRT ⌋
STAND ALONE TELETYPEWRITER	W8-W16	⌊ TTY ⌋
INTELLEC SLAVE CRT TERMINAL	W3-W7	⌊ MDS-CRT ⌋
INTELLEC SLAVE TELETYPEWRITER	W14-W18	⌊ MDS-TTY ⌋

For additional information regarding the operation and programming of the 8251A USART, refer to Appendix B.

2-12. Bus Expansion

The bus expansion logic allows peripheral circuits to be interfaced to the SDK-86. The bus expansion logic, which consists of the control transceivers and drivers, the data transceivers and the address latches, provides all required address, data and control signals for either on-board (user design area) or off-board interfacing of peripheral circuits connected to bus expansion connectors J1/J2 and J3/J4.

The control transceivers (an Intel 8286 bidirection bus driver) determine the source of the five bidirection control signals. When a peripheral "master" device is granted bus control by the 8086 CPU, the $\overline{\text{HLDA}}$ signal at the transceiver's transmit (T) input is active and causes the transceivers to function as receivers (origin of the control signals is from the peripheral circuit). Conversely, while the 8086 has control of the bus, $\overline{\text{HLDA}}$ is inactive, and the control signals originate from the 8086 CPU.

The (control) drivers (a 74LS244 three-state latch) enable the associated 8086 CPU control and status signals onto the expansion bus while the 8086 has control of the bus (HLDA inactive) and, when bus control is granted to the peripheral circuit (HLDA active), the drivers are placed in their high-impedance state.

The data transceivers (two Intel 8286 bidirectional bus drivers) determine the source of the high- and low-order data bytes during off-board memory and I/O operations and interrupt acknowledge cycles. Referring to the block diagram, the transceivers are enabled by the $\overline{\text{BUFFER ON}}$ signal. This signal is active during the T_2 to T_4 instruction cycles ($\overline{\text{DEN}}$ active) for an off-board memory and I/O operation ($\overline{\text{OFF BOARD}}$ active) or an interrupt acknowledge cycle ($\overline{\text{INTA}}$ active). When the transceivers are enabled, the state of the DT/$\overline{\text{R}}$ signal at the T input determines the direction of the transceivers.

The expansion bus address lines (and the $\overline{\text{BHE}}$ signal) originate directly from the address latches (three 74S373 three-state latches) when the 8086 has control of the bus (HLDA inactive). When bus control is granted to the peripheral device (HLDA active), the latches are placed in their high-impedance state.

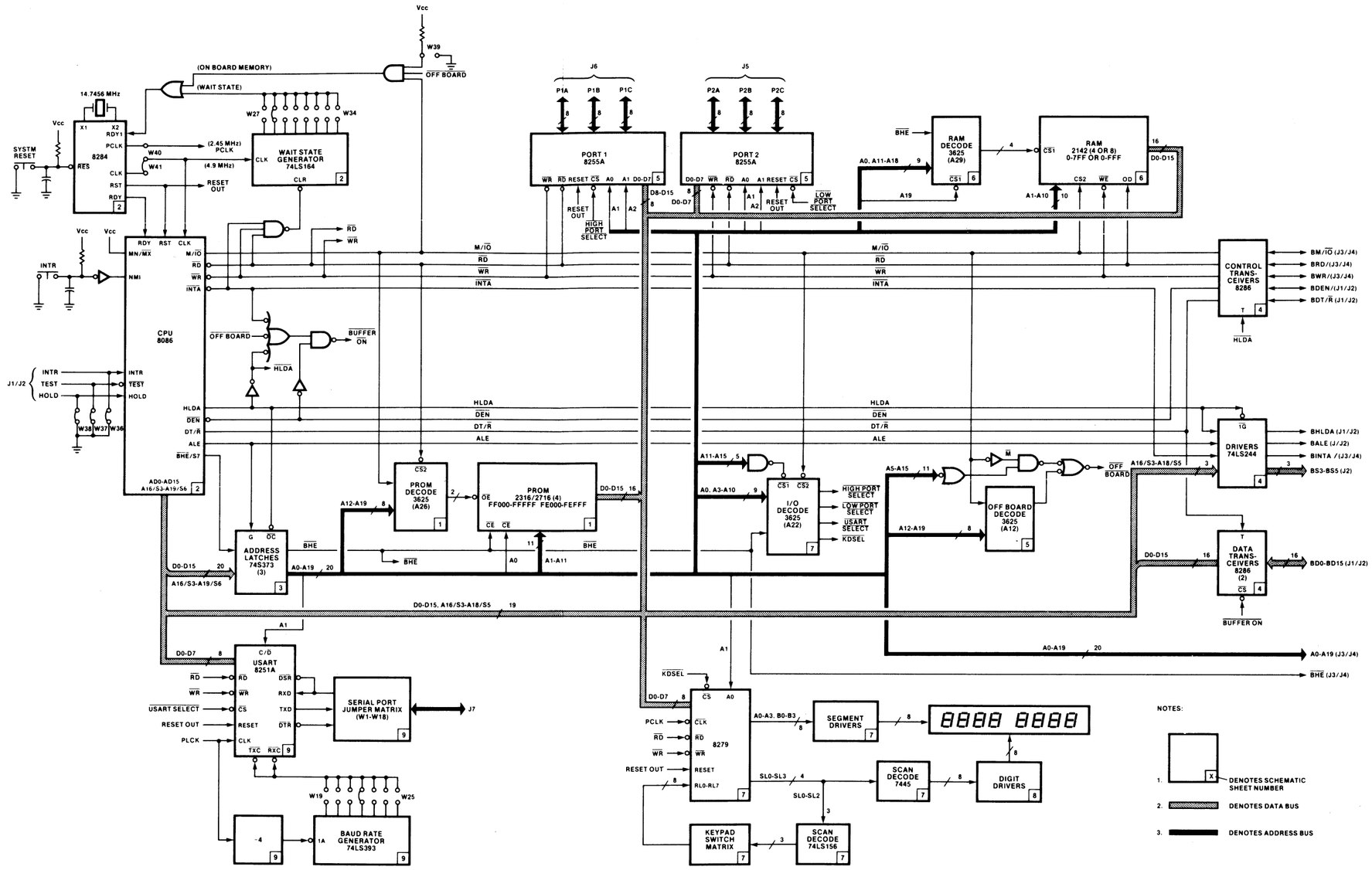


Figure 2-2. Detailed Block Diagram

CHAPTER 3
KEYPAD MONITOR

3-1. Introduction

This chapter describes your "interaction" or "how you communicate" with your MCS-86 System Design Kit through the keypad monitor program. The monitor program resides in 4K bytes of ROM (Read Only Memory) at the upper end of memory. The program is "initialized" or ready whenever power is turned on or any time the **SYSTM RESET** key is pressed and allows you to perform the following operations using the keypad and display.

- Examine and modify registers within the 8086 microprocessor.
- Examine and modify memory locations.
- Enter and initiate execution of your own programs or subroutines.
- Evaluate execution (debug) of your program through the monitor's single-step and breakpoint facilities.
- Move selected blocks of memory from one location to another.
- Write or read data to or from an I/O port.

3-2. Keypad

With the keypad monitor program, you enter both commands and data by pressing individual keys of the keypad. (The monitor communicates with you through the display.) As shown in Figure 3-1, the keypad is divided into two logical groups; the 16 hexadecimal keys on the right-hand side and the eight function keys on the left-hand side.

Most of the hexadecimal keys have combined functions as noted by their individual legends. The small letters appearing under the hexadecimal key values are acronyms for individual monitor commands and 8086 register names. Acronyms to the left of the slash sign are monitor commands, and acronyms to the right of the slash sign are 8086 register names. The function of a hexadecimal key at any one time is dependent on the current state of the monitor and what the monitor is expecting as input. Table 3-1 defines both the commands and registers associated with the hexadecimal keys.

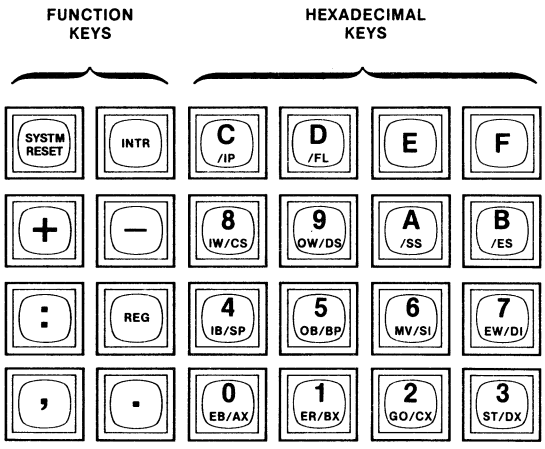


Figure 3-1. Keypad Arrangement

Table 3-1. Hexadecimal Keypad Legend Interpretation

Hexadecimal Key	Command		Register	
	Acronym	Name	Acronym	Name
0 EB/AX	EB	Examine Byte	AX	Accumulator
1 ER/BX	ER	Examine Register	BX	Base
2 GO/CX	GO	Go	CX	Count
3 ST/DX	ST	(Single) Step	DX	Data
4 IB/SP	IB	Input Byte	SP	Stack Pointer
5 OB/BP	OB	Output Byte	BP	Base Pointer
6 MV/SI	MV	Move	SI	Source Index
7 EW/DI	EW	Examine Word	DI	Destination Index
8 IW/CS	IW	Input Word	CS	Code Segment
9 OW/DS	OW	Output Word	DS	Data Segment
A /SS	none	N/A	SS	Stack Segment
B /ES	none	N/A	ES	Extra Segment
C /IP	none	N/A	IP	Instruction Pointer
D /FL	none	N/A	FL	Flag
E	none	N/A	none	N/A
F	none	N/A	none	N/A

The individual operation of the eight function keys is defined in Table 3-2.

Table 3-2. Function Key Operation

Function Key	Operation
SYSTM RESET	The **SYSTM RESET** key allows you to terminate any present activity and to return your SDK-86 to an initialized state. When pressed, the 8086 sign-on message appears in the display and the monitor is ready for command entry.
INTR	The **INTR** (interrupt) key is used to generate an immediate, non-maskable type 2 interrupt (NMI). The NMI interrupt vector is initialized on power up or system reset to point to a routine within the monitor which causes all of the 8086's registers to be saved. Control is returned to the monitor for subsequent command entry.
+	The + (plus) key allows you to add two hexadecimal values. This function simplifies relative addressing by allowing you to readily calculate an address location relative to a base address.
−	The − (minus) key allows you to subtract one hexadecimal value from another.
:	The : (colon) key is used to separate an address to be entered into two parts; a segment value and an offset value.
REG	The **REG** (register) key allows you to use the contents of any of the 8086's registers as an address or data entry.
,	The , (comma) key is used to separate keypad entries and to increment the address field to the next consecutive memory location.
.	The . (period) key is the command terminator. When pressed, the current command is executed. Note that when using the Go command, the 8086 begins program execution at the address specified when the key is pressed.

3-3. Display

Your SDK-86 kit uses the eight-digit display to communicate with you. Depending on the current state of the monitor, the information displayed will be the:

- Current contents of a register or memory location
- An "echo" of a hexadecimal key entry
- A monitor prompt sign
- An information or status message

The display itself is divided into two groups of four characters. The group on the left is referred to as the "address field," and the group on the right is referred to as the "data field." All values displayed are in hexadecimal and follow the format shown in Table 3-3.

Table 3-3. Hexadecimal Display Characters

Hexadecimal Value	Display Format
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
A	
B	
C	
D	
E	
F	

3-4. General Operation

When using the keypad monitor, you will be prompted through the display as to the input required. Whenever the monitor is expecting a command entry, a minus sign or "dash" appears in the most significant display digit of the address field. Pressing one of the hexadecimal "command" keys (keys 0-9) when the dash is displayed is interpreted as a command entry. When the key is pressed, the dash disappears and a decimal point (or decimal points) appears in the least significant display digit (or least significant digits) of the address field to indicate that subsequent keypad entry will be directed to the address field. Note that depending on the command, characters also may appear within the address and data fields. Monitor operation from this point is determined by the actual command entered. Refer to Sections 3-6 through 3-12 for individual command format and operation.

Following power-on or whenever the **SYSTM RESET** key is pressed, the monitor initializes the SDK-86 and displays the monitor sign-on message ("86" in the two least-significant digits of the address field and the program's version number in the two least-significant digits of the data field) and the command prompt character (dash) in the most significant digit of the address field. When initialized, the following 8086 registers are set to the values shown in Table 3-4 by the monitor.

Table 3-4. Register Initialization

Register	Value
CS (Code Segment)	0H
DS (Data Segment)	0H
ES (Extra Segment)	0H
SS (Stack Segment)	0H
IP (Instruction Pointer)	0H
FL (Flag)	0H
SP (Stack Pointer)	0100H

Note that in the above table and in the remainder of this manual, the letter "H" is used to indicate that the associated value is in hexadecimal.

The first 256 memory locations (locations 0H-0FFH) are reserved for the monitor and user stack area as shown in Figure 3-2. (First user-available memory location in RAM is 0100H.)

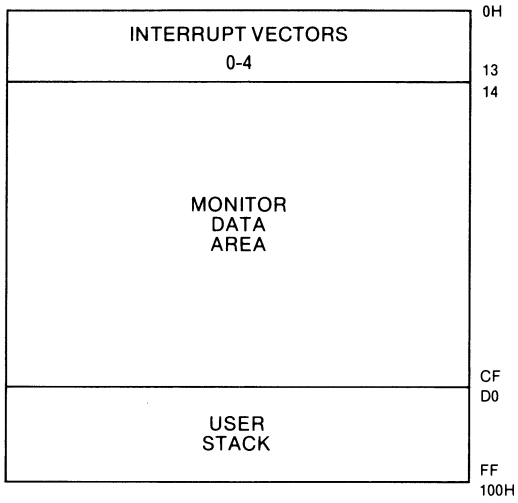


Figure 3-2. Reserved Memory

Whenever the SDK is powered up or whenever the **SYSTM RESET** key is pressed, the monitor immediately terminates its present activity and jumps to its initialization routine. This routine initializes interrupt vectors 1 through 3 as follows:

Interrupt 1: Single Step: Used with the Single Step command
Interrupt 2: NMI (Non-Maskable Interrupt): Monitors INTR key
Interrupt 3: Breakpoint: Used with the Go command

Whenever the monitor is re-entered as a result of a Single Step, NMI or Breakpoint interrupt, the monitor temporarily stores (pushes) the contents of the 8086 registers onto the user stack and subsequently removes (pops) the register contents from the stack before it prompts for command entry. Since the SP register is initialized to 0100H (base of the stack), the initial stack reserved for the user is 48 bytes (locations D0-0FFH) of which 26 bytes must be reserved for the register contents should one of the above interrupts occur.

Note that in the following command descriptions, the monitor always calculates a 20-bit physical memory address from a 16-bit segment address value and a 16-bit offset address value. The segment address value is entered first, the ":" key is pressed (to separate the two entries), and then the offset address value is entered. The two values entered, which are displaced from one another by four bits, are added together as shown in Figure 3-3 (Segment-Offset Entry) to form the 20-bit physical memory address. If only one address value is entered (the colon must be omitted), it is interpreted by the monitor as an offset address value entry, and the current contents of the code segment (CS) register are used as the segment address value. The CS register contents and the offset address value entered are combined as shown in the Single Offset Entry example of Figure 3-3 to form the 20-bit physical memory address.

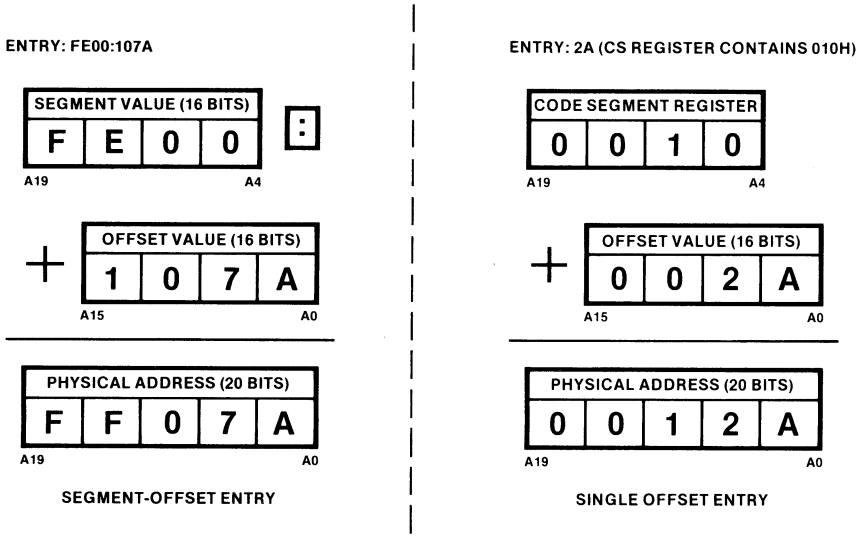


Figure 3-3. Physical Address Calculation

3-5. Monitor Commands

The keypad monitor is capable of executing ten individual commands. Each command is summarized in Table 3-5 and is described, in detail, within the sections which follow. In both the table and the individual command descriptions, the following command syntax is used:

[X] Indicates a keyboard key

[A] Indicates that "A" is optional

[A]* Indicates one or more optional occurrences of "A"

<B> Indicates that "B" is a variable

Table 3-5. Monitor Command Summary

Command	Function/Syntax
Examine Byte	Displays/modifies memory byte locations EB <addr> [[<data>]]*
Examine Word	Displays/modifies memory word locations EW <addr> [[<data>]]*
Examine Register	Displays/modifies 8086 register contents ER <reg key>[[<data>]]*[ ]
Input Byte	Displays data byte at input port IB <port addr> [ ]*
Input Word	Displays data word at input port IW <port addr> [ ]*
Output Byte	Outputs data byte to output port OB <port addr> <data>[<data>]*
Output Word	Outputs data word to output port OW <port addr> <data>[<data>]*
Go	Transfers control from monitor to user program GO [<addr>][<breakpoint addr>]
Move	Moves block of data within memory MV <start addr> <end addr> <destination addr>
Step	Executes single user program instruction ST [<start addr>] [[<start addr>]]*

3-6. Examine Byte and Examine Word Commands

FUNCTION

The Examine Byte (EB) and Examine Word (EW) commands are used to examine the contents of selected memory locations. If the memory location can be modified (e.g., a location in RAM), the contents additionally can be updated.

SYNTAX

EB <addr> [[<data>]]*

EW <addr> [[<data>]]*

OPERATION

To use either command, press the **EB** key (examine byte) or **EW** key (examine word) when the command prompt character (−) is displayed. When either key is pressed, the decimal point at the right of the address field will light (the rest of the display will be blanked) to indicate that entry from the keyboard will be directed to the address field. From the keypad, enter the memory address of the byte or word to be examined, most significant character first. Note that all memory addresses consist of both a segment value and an offset value. When no segment value is specified, the default segment value is the current contents of the code segment (CS) register. When a segment value is specified, the first address entry is the segment value, a colon (:) is entered as a separator, and the second address entry is the offset value. The capacity of an address field entry is limited to four characters and, if more than four characters are entered for either a segment or offset value, only the last four

characters entered (the four characters currently displayed) are valid. After the address is entered, press the "," key. The data byte or word contents of the addressed memory location will be displayed in the data field, and a decimal point will appear at the right of the data field to indicate that any subsequent hexadecimal keypad entry will be directed to the data field. Note that when using the Examine Word command, the byte contents of the memory location displayed appear in the two least-significant digits of the data field, and the byte contents of the next consecutive memory location (memory address + 1) appear in the two most-significant digits of the data field.

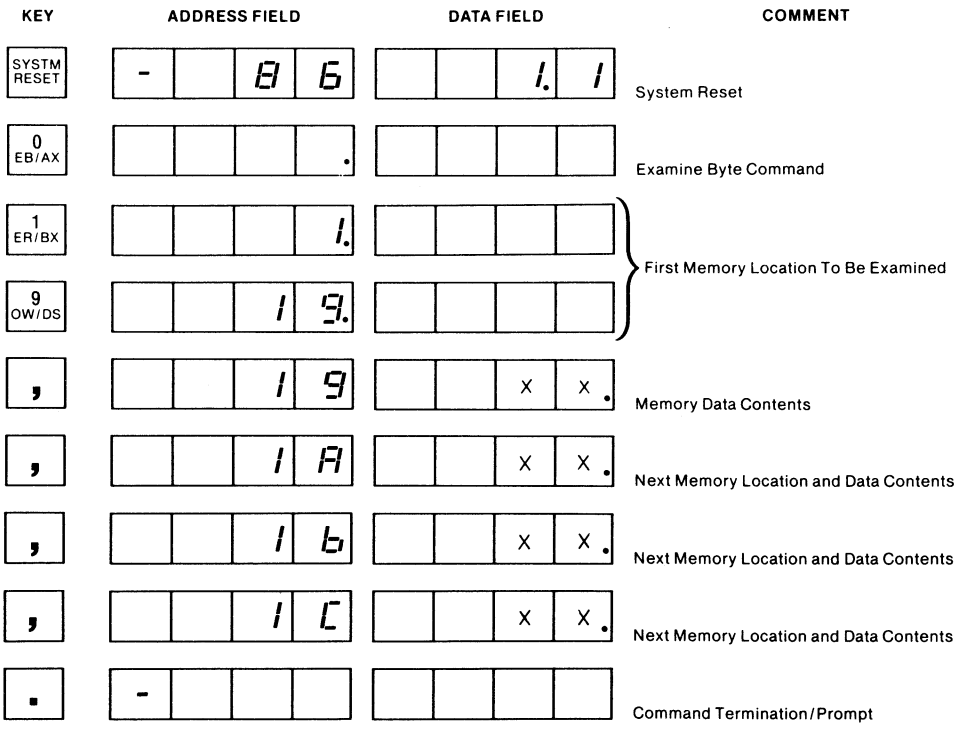
If the contents of the memory location addressed only are to be examined, press the "." key to terminate the command, or press the "," key to examine the next consecutive memory location (Examine Byte command) or the next two consecutive memory locations (Examine Word command). To modify the contents of an addressed memory location, key-in the new data from the hexadecimal keypad. Note that the data field is limited to either two (examine byte) or four (examine word) characters and that if more characters are entered, only the characters currently displayed in the field are valid. The data displayed is not updated in memory until either the "." or "," key is pressed. If the "." key is pressed, the command is terminated, and the command prompt character is displayed in the address field. If the "," key is pressed, the *offset* address and data contents of the next consecutive memory location (Examine Byte command) or memory locations (Examine Word command) are displayed.

ERROR CONDITIONS

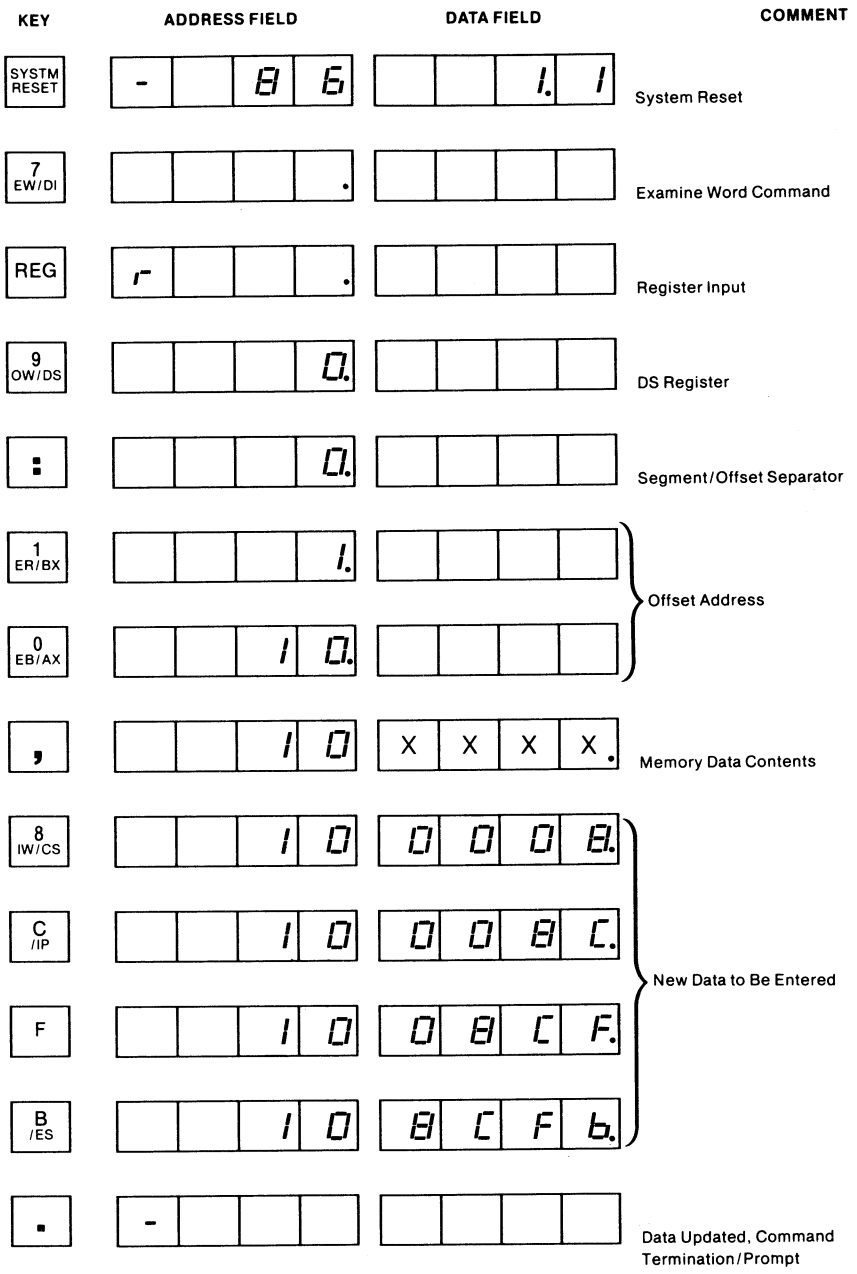
Attempting to modify a non-existent or read-only (e.g., ROM or PROM) memory location. Note that the error is not detected until the "," or "." key is pressed. When an error is detected, the characters "Err" are displayed with the command prompt character in the address field.

EXAMPLES

Example 1: Examining a Series of Memory Byte Locations Relative to the CS Register

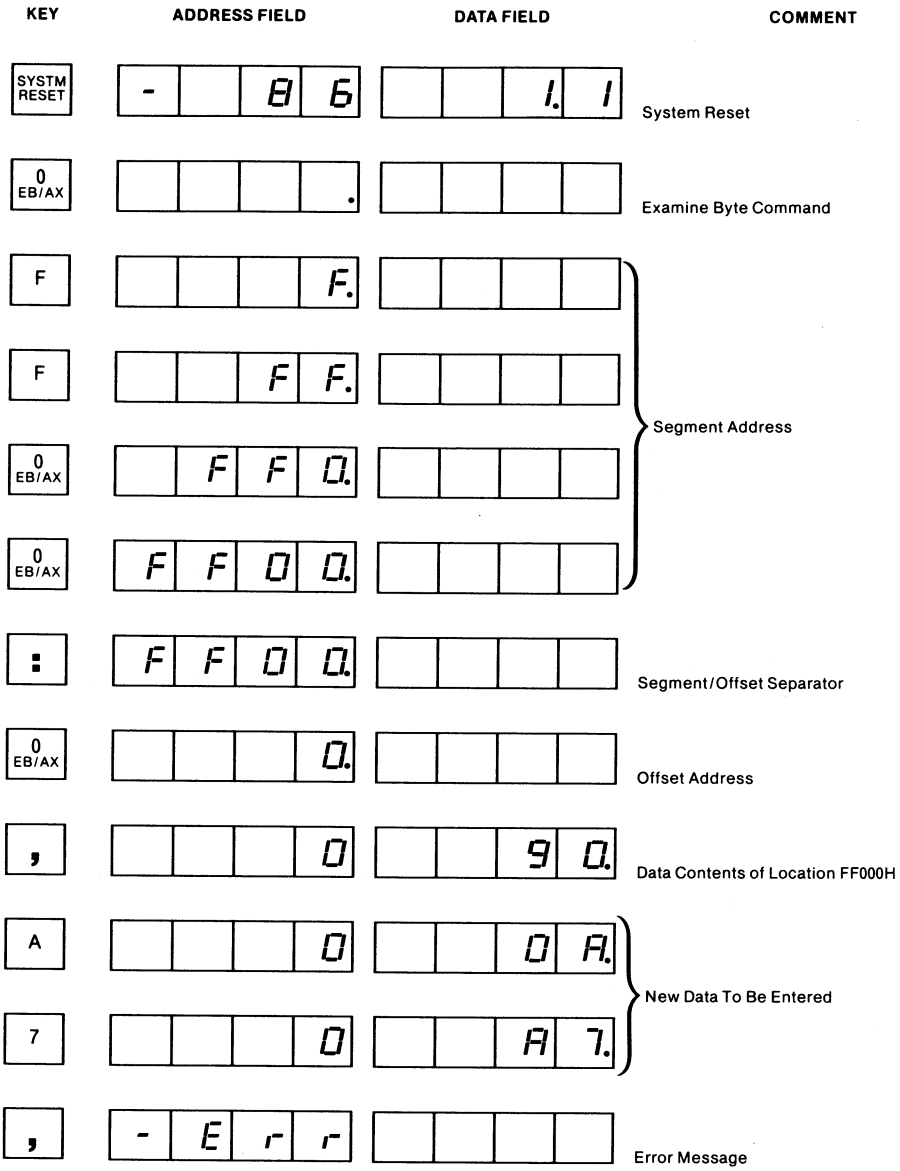


Example 2: Examining and Modifying Memory Word Location 10H Relative to the DS Register



To check that the data was updated successfully, press the **EW** key and enter the memory address (DS:10H). Press the "," key and note that "8CFb" is displayed in the data field.

Example 3: Attempting to Modify PROM



The error message (Err) resulted from trying to modify the data contents of a read only memory location. Repeat the keying sequence to see that the memory contents (90H) of location FF000H were unaltered.

3-7. Examine Register Command

FUNCTION

The Examine Register (ER) command is used to examine and, if desired, to modify the contents of any of the 8086's registers.

SYNTAX

ER <reg key>[[<data>]⊡]*[⊡]

OPERATION

To examine the contents of a register, press the **ER** key when prompted for a command entry. When the key is pressed, the decimal point at the right of the address field will light. Unlike the Examine Byte command, subsequent hexadecimal keypad entry will be interpreted as the register name (the acronym to the right of the slash sign on the key face) rather than its hexadecimal value. When the hexadecimal key is pressed, the corresponding register abbreviation will be displayed in the address field, the 16-bit contents of the register will be displayed in the data field and the decimal point on the right of the data field will light. Table 3-6 defines the 8086 register name, the hexadecimal keypad acronym, the display abbreviation and the sequence in which the registers are examined.

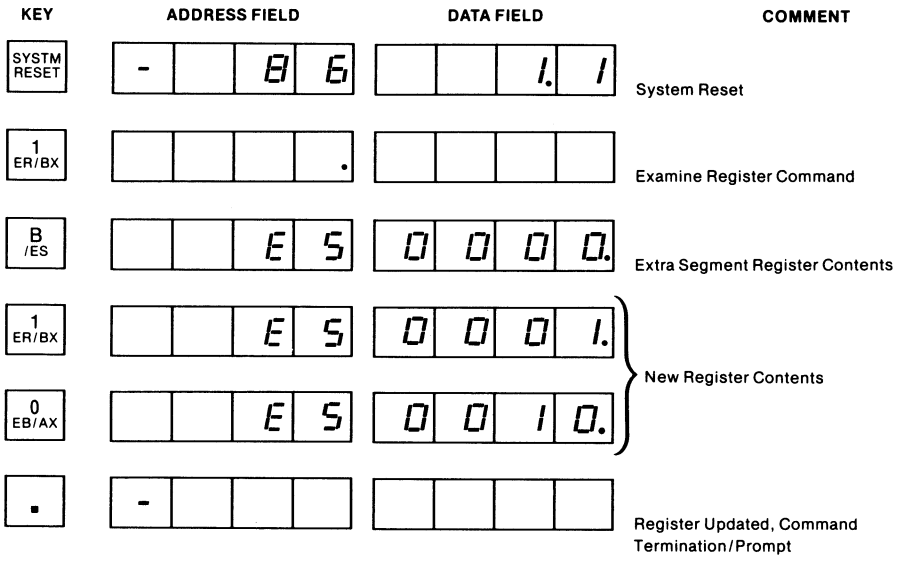
Table 3-6. 8086 Registers

Register Name	Keypad Acronym	Display Abbreviation
Accumulator	AX	A
Base	BX	b
Count	CX	C
Data	DX	d
Stack Pointer	SP	SP
Base Pointer	BP	bP
Source Index	SI	SI
Destination Index	DI	dI
Code Segment	CS	CS
Data Segment	DS	dS
Stack Segment	SS	SS
Extra Segment	ES	ES
Instruction Pointer	IP	IP
Flag	FL	FL

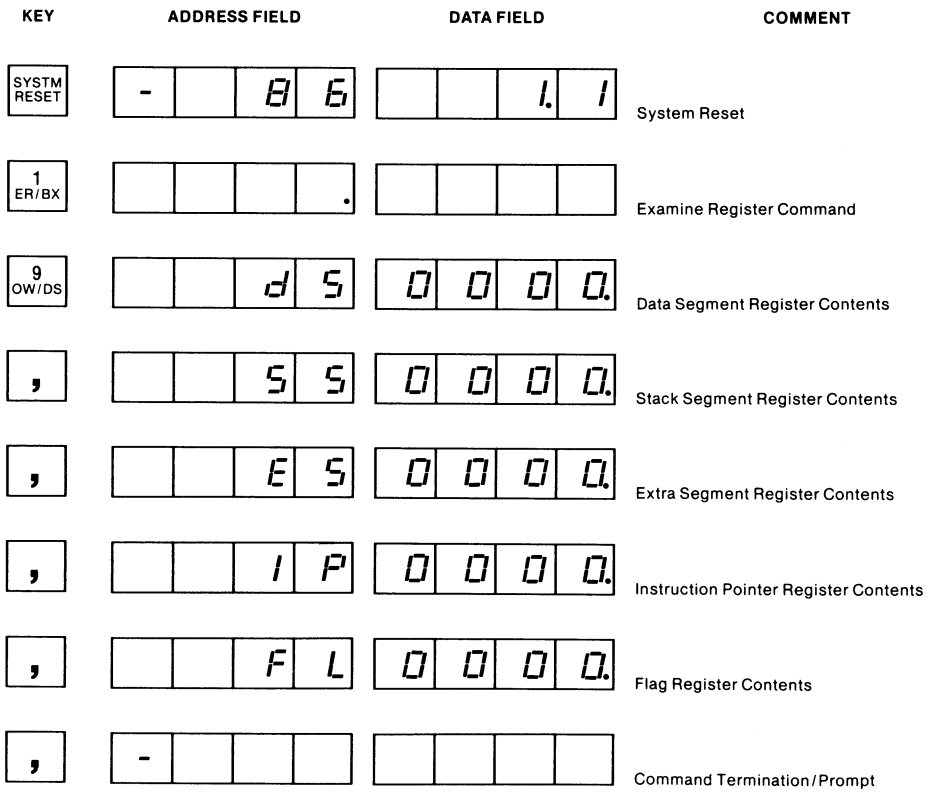
When the register contents are displayed (when the decimal point on the right of the data field lights), the register's contents can be modified or the register is said to be "open" for input. Keying in a value from the hexadecimal keypad will be echoed in the display's data field, and the register contents will be updated with the data value displayed when either the "," or "." key is pressed. If the "." key is pressed, the command is terminated, and the command prompt character is displayed. If the "," key is pressed, the abbreviation and contents of the "next" register are displayed and opened for modification according to the order in Table 3-6. Note that the sequence is *not* circular and that pressing the "," key when the flag (FL) register is displayed will terminate the examine register command and return to the command prompt character.

EXAMPLES

Example 1: Examining and Modifying a Register



Example 2: Examining a Series of Registers



3-8. Input Byte and Input Word Commands

FUNCTION

The Input Byte (IB) and Input Word (IW) commands are used to input (accept) an 8-bit byte or 16-bit word from an input port.

SYNTAX

[IB] <port addr> [,] [,]* [.]

[IW] <port addr> [,] [,]* [.]

OPERATION

To use either the Input Byte or Input Word command, press the corresponding hexadecimal key when prompted for command entry. When either the **IB** or **IW** key is pressed, the decimal point on the right of the address field will light to indicate that a port address entry is required. Using the hexadecimal keypad, enter the port address of the port to be read. Note that since I/O addressing is limited to 64K (maximum address FFFFH), no segment value is permitted with the port address.

After the port address has been entered, press the "," key. The input byte or word at the addressed port will be displayed in the data field. Again pressing the "," key updates the data field display with the current data byte or word at the addressed input port. Pressing the "." key terminates the command and prompts for command entry.

The SDK-86 includes two 8255A parallel I/O port circuits which can be used with the Input Byte and Input Word commands to input data from peripheral devices. These two circuits are designated P1 and P2. Each circuit, in turn, consists of three individual 8-bit ports which are designated port A, port B, and port C. As shown in Figure 3-4, each port operates independently during byte operations. During word operations, a pair of ports (e.g., P1A and P2A) operate together to form the 16-bit wide data word with P2 corresponding to the low-order byte.

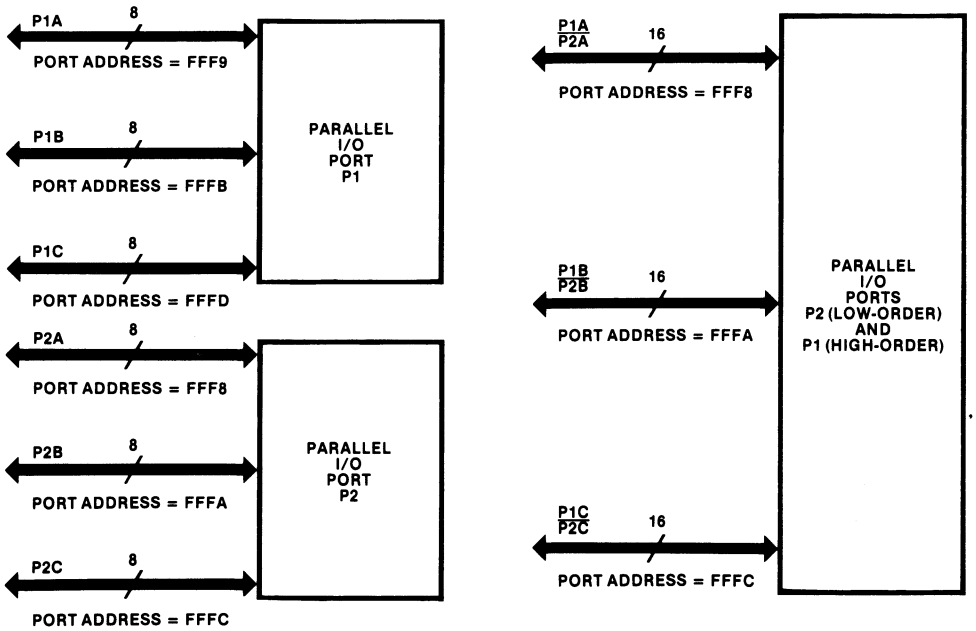


Figure 3-4. Parallel I/O Port Configuration

Table 3-7 defines the individual port addresses. Note that during word operations, the low-order (P2) port address is entered (the corresponding high-order port is addressed automatically).

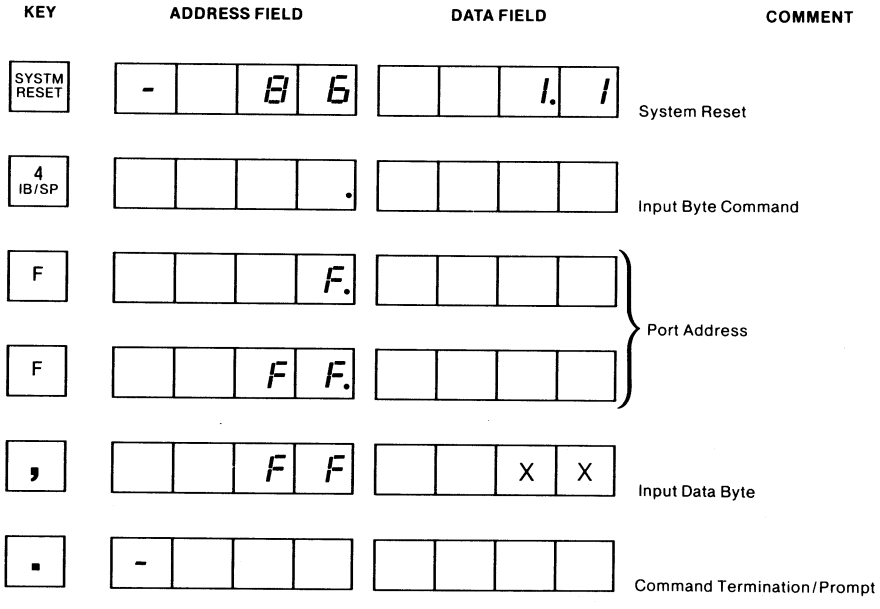
Table 3-7. Parallel I/O Port Addressing

Port	Address
P2A	FFF8
P1A	FFF9
P2B	FFFA
P1B	FFFB
P2C	FFFC
P1C	FFFD

The parallel I/O port circuits are programmed for input on power-up or whenever the **SYSTM RESET** key is pressed. If the circuit(s) previously has been programmed for output, press the **SYSTM RESET** key (before pressing the command key) or, referring to the next section, output the appropriate byte or word value to the circuit's control port to program the port(s) for input. For additional information regarding the operation of the 8255A parallel I/O port circuit, refer to the circuit description in Appendix B.

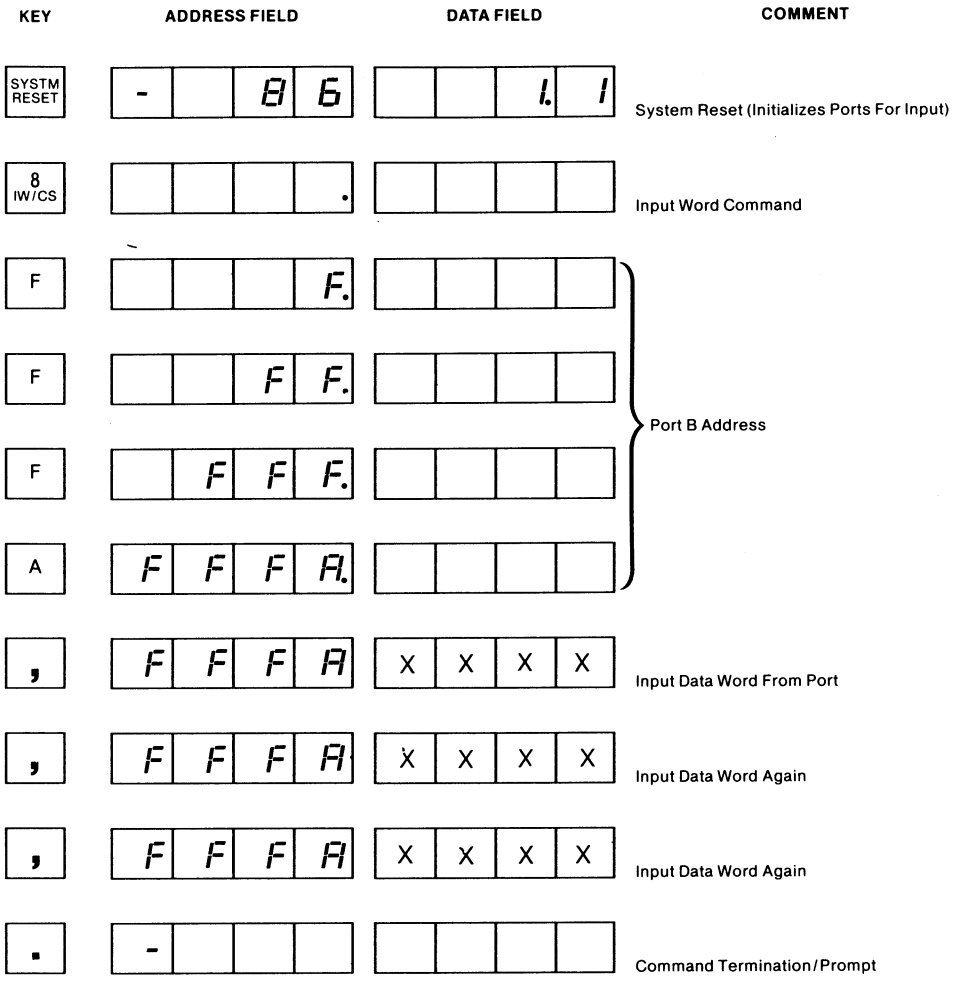
EXAMPLES

Example 1: Single Byte Input from Port 0FFH*



* Port 0FFH is not provided on the SDK-86

Example 2: Multiple Word Input from Parallel I/O Ports 1B and 2B



3-9. Output Byte and Output Word Commands

FUNCTION

The Output Byte (OB) and Output Word (OW) commands are used to output a byte or word to an output port.

SYNTAX

OB <port addr> ⊡ <data> [⊡ <data>]* ⊡

OW <port addr> ⊡ <data> [⊡ <data>]* ⊡

OPERATION

To use either command, press the corresponding hexadecimal key when prompted for command entry. When either the **OB** or **OW** key is pressed, the decimal point on the right of the address field will light to indicate that a port address entry is required. Like the Input Byte and Input Word commands, I/O addresses are limited to 64K, and no segment value is permitted. After the port address is entered, press the "," key. The decimal point on the right of the data field will light to indicate that the data byte or word to be output now can be entered. Using the keypad, enter

the byte or word to be output. After the data is entered, press the "." key to output the byte or word to the port and to terminate the command or press the "," key if additional data is to be output to the addressed port.

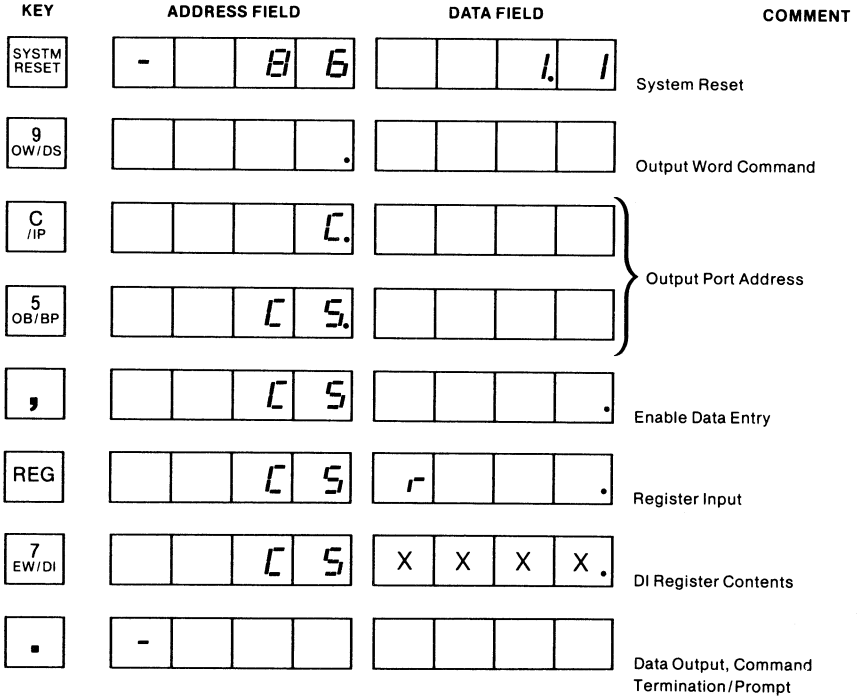
As mentioned in the previous section, the Output Byte and Output Word commands can be used to program the 8255A parallel I/O port circuits for input or output as well as to output data to the individual ports. The I/O port circuits are programmed for input on power-up or system reset and consequently first must be programmed for output (by outputting the appropriate data byte or word to the circuit's control port) before data can be output to the associated ports. Table 3-8 defines the control port addressing and associated data byte or word to be output to the control port. For more information regarding the operation of the 8255A parallel I/O port circuit, refer to the circuit description in Appendix B.

Table 3-8. Control Port Addressing

Port Number	Port Address	Data Byte or Word	
		Input	Output
P2	FFFEH	9BH	80H
P1	FFFFH	9BH	80H
P2/P1	FFFEH	9B9BH	8080H

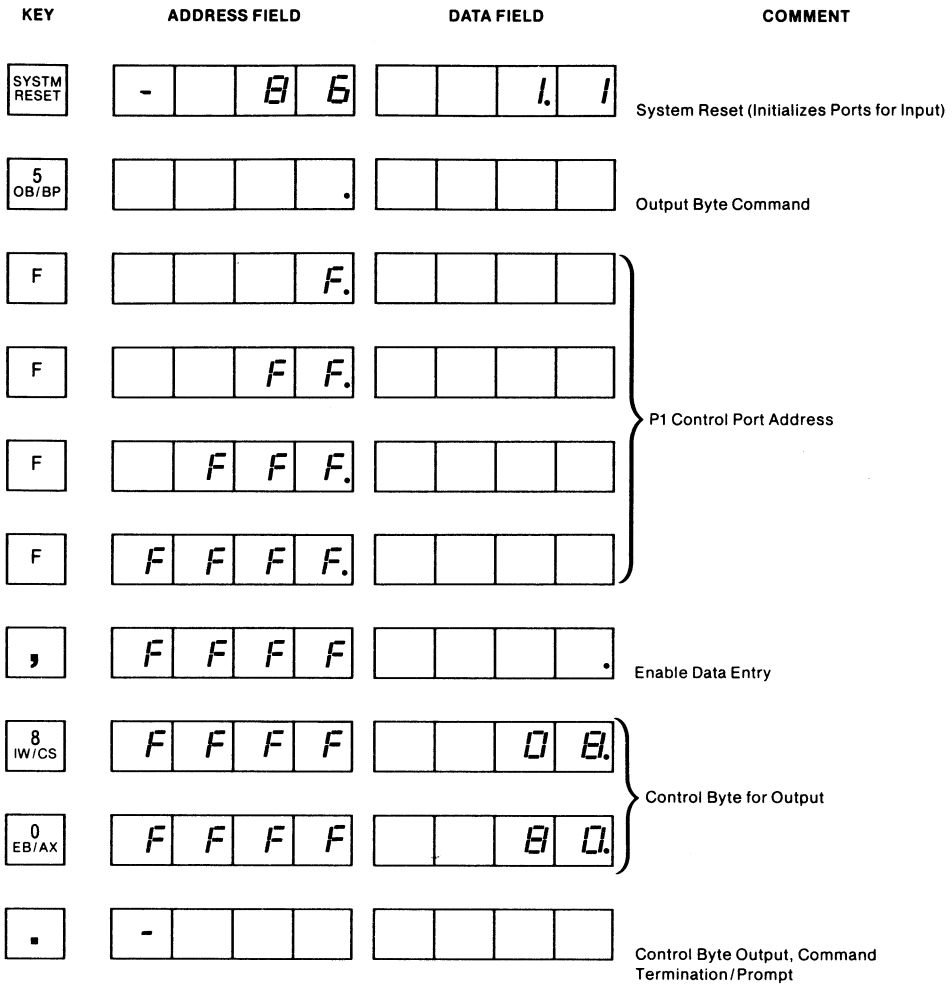
EXAMPLES

Example 1: Output DI Register Contents to Output Port 0C5H*



* Port 0C5H is not provided on the SDK-86

Example 2: Programming Port P1 for Output



3-10. Go Command

FUNCTION

The Go (GO) command is used to transfer control of the 8086 from the keypad monitor program to a user's program in memory.

SYNTAX

GO [<addr>][,<breakpoint addr>]

OPERATION

To use the Go command, press the **GO** key when prompted for command entry. When the key is pressed, the current IP (instruction pointer) register contents are displayed in the address field, the byte contents of the memory location addressed by the IP register are displayed in the data field, and the decimal point at the right of the address field lights to indicate that an alternate start address entry can be entered. If an alternate starting address is required, enter the address from the keypad. (When an address is entered, the data field is blanked.) To begin program execution (at the current instruction or alternate program address), press the "." key. When the key is pressed, the monitor displays an "E" in the most-significant digit of the address field before transferring control to the program.

To illustrate the operation of the Go command, the following sample program can be entered into memory using the Examine Byte command. The program simulates a dice game in which you "throw" a single die. After the program has been entered and after control is passed to the program by the Go command, pressing any keyboard key (other than **SYSTM RESET** or **INTR**) starts the die rolling. Again pressing any key stops the die and displays its value (a number between 1 and 6 appears in the leftmost digit of the address field).

Sample Program

Memory Location	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0100	8C	C9	8E	D9	BA	EA	FF	B0	D3	EE	BA	EA	FF	EC	24	0F
0110	74	FB	E8	28	00	BB	00	00	43	80	FB	07	74	F7	8B	FB
0120	8A	4D	47	90	BA	EA	FF	B0	87	EE	BA	E8	FF	8A	C1	EE
0130	BA	EA	FF	EC	24	0F	74	E0	E8	02	00	EB	CD	BA	EA	FF
0140	B0	40	EE	BA	E8	FF	EC	C3	06	5B	4F	66	6D	7D		

The sample program consists of 4E hexadecimal locations and is entered into memory beginning at location 100H. After you have entered all of the program data, press the "." key to terminate the Examine Byte command and to cause the monitor to request a command entry. When prompted, press the **GO** key and enter the program's starting memory address using a segment value of 10 and an offset value of 0 (10:0). Press the "." key and note that the display is blanked. Now pressing any keyboard key will start the die rolling, and again pressing any key will stop the die. Note that since the sample program uses the most-significant digit of the address field, the "E" is overwritten.

NOTE

After the program is entered into memory, it will remain in memory until power is removed and will not be affected by pressing the **SYSTM RESET** key.

To exit from the executing program and return control to the monitor, press either the **SYSTM RESET** or **INTR** key. If the **SYSTM RESET** key is pressed, the monitor is re-entered and the appropriate 8086 registers are initialized. If the **INTR** key is pressed, the monitor is re-entered, *all* of the 8086 registers are saved, and the monitor prompts for a command entry. When the monitor is re-entered by pressing the **INTR** key, pressing the **GO** key causes the current IP register value (the offset address of the next program instruction to be executed when the **INTR** key was pressed) and the byte contents of that location (addressed by both the IP and CS registers) to be displayed. Pressing the "." key transfers control from the monitor back to the program at the instruction addressed and program execution continues.

The Go command optionally permits a "breakpoint address" to be entered. A breakpoint address has the same affect as pressing the **INTR** key while a program is being executed. To enter a breakpoint address, press the "," key after entering the starting address and enter the breakpoint address. Note that when specifying a breakpoint address, the default segment value is either the starting address segment value (if specified) or the current CS register contents (if a segment value is not specified with the starting address). When the "." key is pressed, the monitor replaces the instruction at the breakpoint address with an interrupt instruction and saves the "breakpointed" instruction before transferring control to the user's program. When the program reaches the breakpoint address, control is returned to the

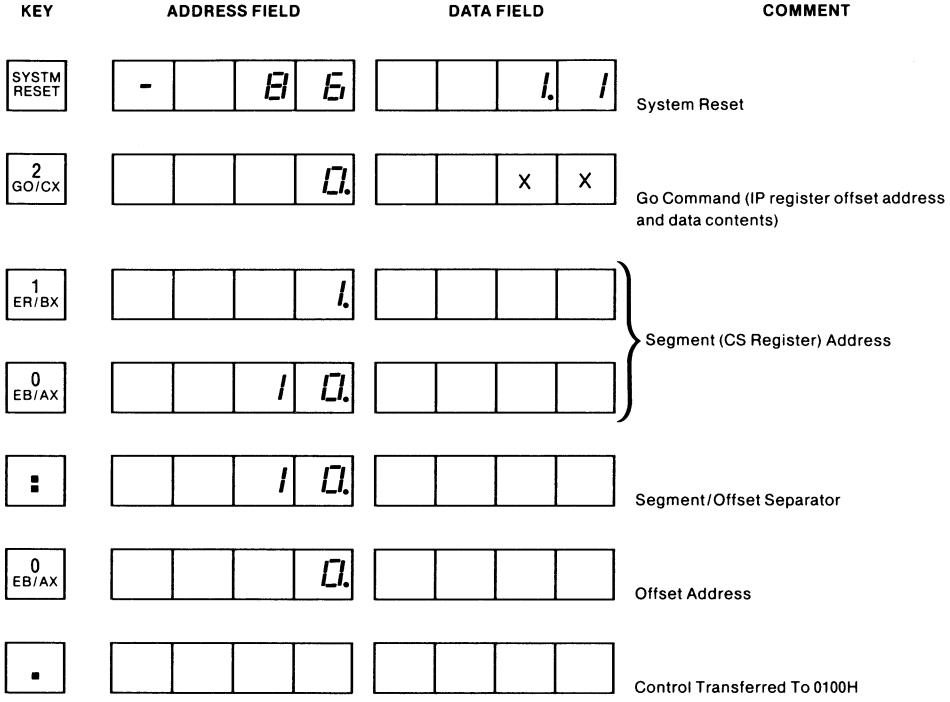
monitor, the breakpointed instruction is restored in the program, all registers are saved, and the monitor prompts for command entry to allow any of the registers to be examined. Note that since the breakpointed instruction is restored when control is returned to the monitor, the breakpoint address must be specified each time the program is to be executed with a breakpoint.

ERROR CONDITIONS

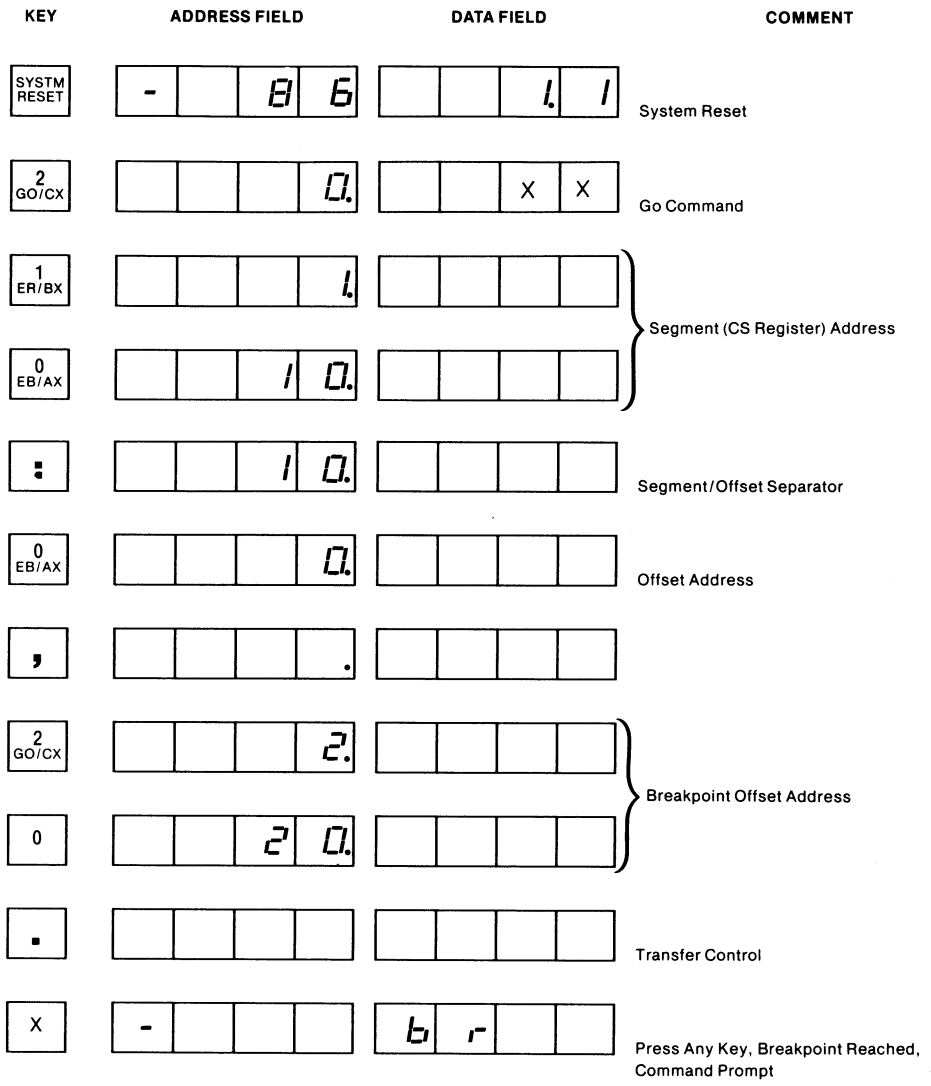
Attempting to breakpoint an instruction in read-only memory.

EXAMPLES

Example 1: Transferring Control to the Sample Program



Example 2: Entering and Executing a Breakpoint in the Sample Program



3-11. Move Command

FUNCTION

The Move (MV) command permits a block of data to be moved within memory.

SYNTAX

MV <start addr> , <end addr> , <destination addr> ,

OPERATION

The format of the Move command is unique in that three successive entries are made in the address field. To use the Move command, press the **MV** key when prompted for command entry. When the key is pressed, three decimal points appear in the address field to indicate that three entries are required. Each time an entry is made, the leftmost decimal point goes out so that the number of decimal points lit at any one time indicates the number of entries still required. The entries are, in order:

1. The starting memory address of the block of data to be moved.
2. The ending memory address of the block of data to be moved.

3. The starting memory address (destination address) into which the block of data is to be moved.

Note that no segment value is permitted with an ending address and that block moves consequently are limited to 64K bytes.

When the "." key is pressed, the data is moved and the command prompt sign is displayed. Note that when the block of data is moved, the data contained in the original (source) memory locations is *not* altered (unless the destination address falls within the original block of data in which case the overlapping memory locations will be overwritten by the data moved).

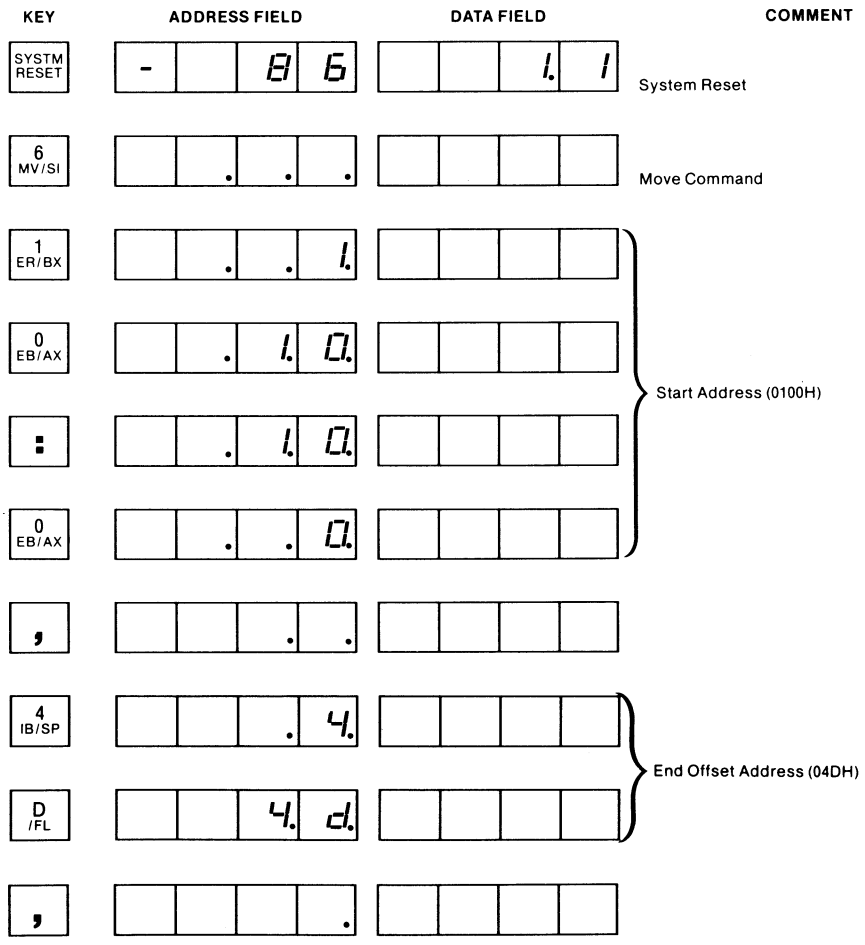
Since a move is performed one byte at a time, the Move command can be used to fill a block of memory with a constant. This is accomplished by specifying a destination address that is one greater than the start address. The block of memory locations from start address to end address + 1 will be filled with the value contained in the start address location. (The Examine Byte command can be used to specify the contents of start address.)

ERROR CONDITIONS

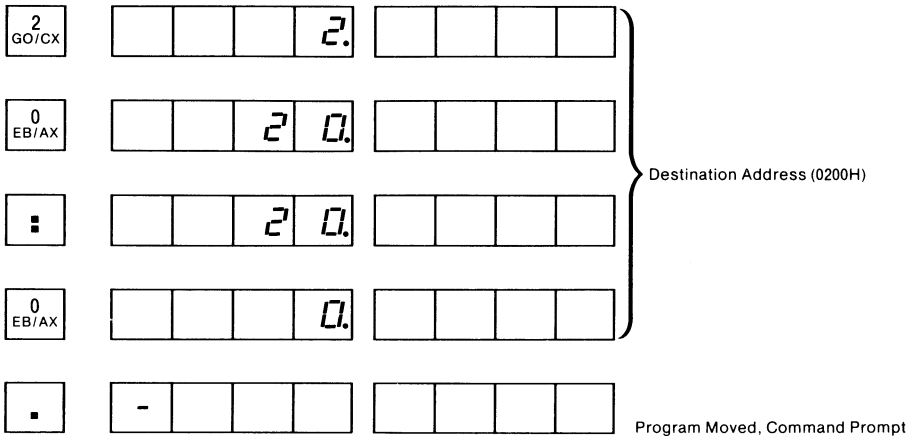
Attempting to move data into read-only or non-existent memory.

EXAMPLES

Example 1: Moving the Dice Program to Location 0200H



Example 1: Moving the Dice Program to Location 0200H (Cont'd.)



3-12. Step Command

FUNCTION

The Step (ST) command permits program instructions in memory to be executed individually. With each instruction executed, control is returned to the monitor from the program.

SYNTAX

ST [<start addr>]↵ [[<start addr>]↵]* ↵

OPERATION

To use the Step command, press the **ST** key when prompted for command entry. If a starting address other than the address displayed is required, enter the desired address. When the "," key is pressed, the instruction addressed is executed and the offset address of the next instruction to be executed is displayed in the address field and its associated instruction byte is displayed in the data field. Again pressing the "," key executes the current instruction and steps the program to the next instruction to be executed.

In the example which follows, the first few instructions of the "rolling dice" program are executed using the step command. The following table represents a listing of the beginning of that program. (A complete listing of the "rolling dice" program is included at the end of this chapter.)

```
LOCATION  CONTENTS  SYMBOLIC              COMMENTS

00        8CC9      MOV  CX,CS           ;DEFINE DATA SEGMENT REGISTER EQUAL
02        8ED9      MOV  DX,CX           ;TO CODE SEGMENT REGISTER
04        BAEAFF    MOV  DX,0FFEAH       ;CLEAR DISPLAY CONTROL WORD
07        B0D3      MOV  AL,0D3H         ;TO 8279
09        EE        OUT  DX

                    READKEY:
0A        BAEAFF    MOV  DX,0FFEAH       ;SET UP 8279 COMMAND PORT
0D        EC        IN   DX              ;READ 8279 FIF0 STATUS
0E        240F      AND  AL,0FH          ;MASK ALL BUT FIFO COUNT
10        74FB      JZ   READKEY + 3     ;KEEP READING IF ZERO
12        E82800    CALL READATA         ;DUMMY READ TO UNLOAD
                                         ;PRESSED KEY FROM FIFO
```

Note that when the program is stepped from the instruction at 10H, the instruction at 0DH is executed again and the instruction at 12H is not executed. This is caused by the JZ (jump zero) instruction at 10H which, since no key can be pressed to "roll

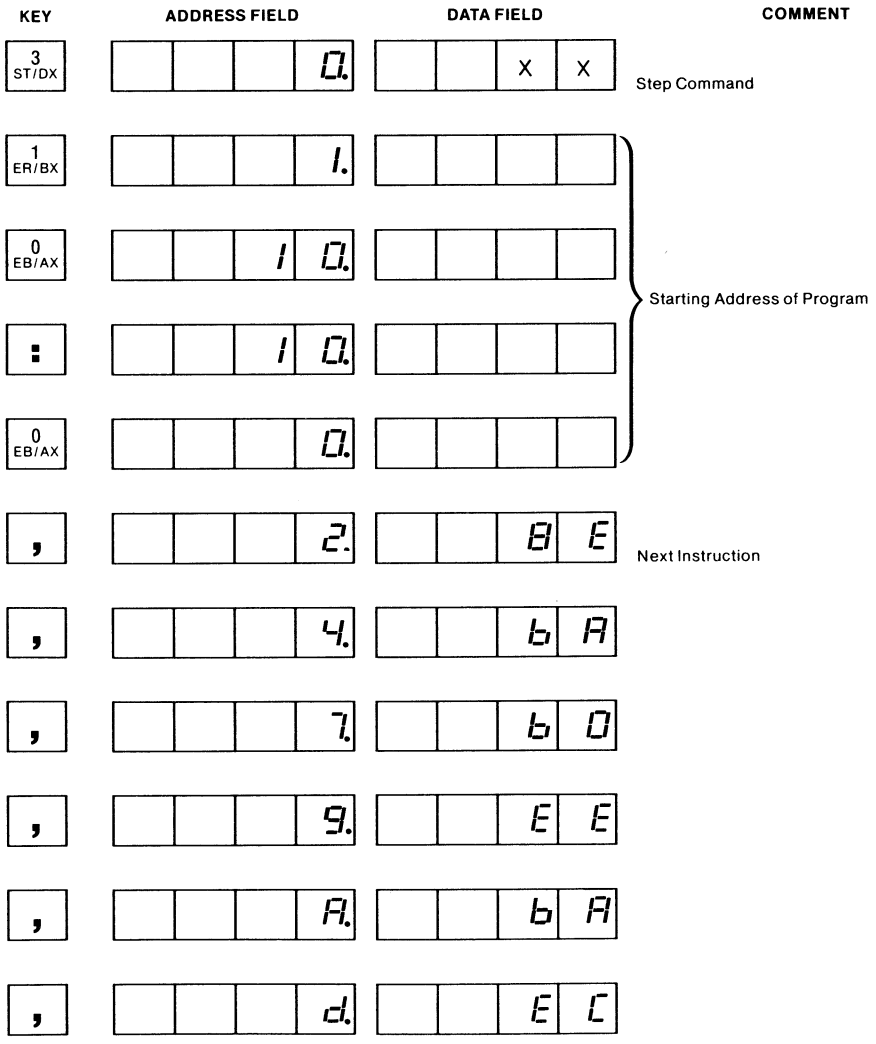
the dice'' (the monitor is in control of the keyboard), jumps back to the instruction at 0DH. Continuing to press the "," key will repeat the three instruction sequence (0DH, 0EH, 10H).

RESTRICTIONS

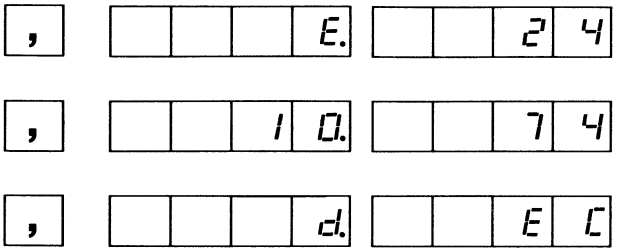
1. If an interrupt occurs prior to the completion of a single-stepped instruction or if a single-stepped instruction generates an interrupt, when the monitor is re-entered, the CS and IP registers will contain the address of the interrupt service routine. Consequently, a type 3 (breakpoint) interrupt instruction (0CCH or 0CDH) should not be single-stepped since its execution would step into the monitor.
2. An instruction that is part of a sequence of instructions that switches between stack segments (i.e., changes the SS and SP register contents) cannot be single-stepped.

EXAMPLES

Example 1: Program Stepping



Example 1: Program Stepping (Cont'd.)



3-13. Program Listing

The following is the complete program listing for the "rolling dice" program used in the previous command examples. For more information regarding program operation and interpretation, refer to the *MCS-86 User's Manual* or the *ASM86 Language Reference Manual,* Order Number 9800640.

```
LOCATION  CONTENTS  SYMBOLIC                          COMMENTS

                    ASSUME DS:DICE, CS:DICE
          DICE      SEGMENT AT 100
00        8CC9      MOV     CX,CS          ;DEFINE DATA SEGMENT
02        8ED9      MOV     DS,CX          ;REGISTER EQUAL TO
                                           ;CODE SEGMENT REGISTER
04        BAEAFF    MOV     DX,0FFEAH      ;CLEAR DISPLAY CONTROL
07        B0D3      MOV     AL,0D3H        ;WORD TO 8279
09        EE        OUT     DX

          READKEY:
0A        BAEAFF    MOV     DX,0FFEAH      ;SET UP 8279 COMMAND PORT
0D        EC        IN      DX             ;READ 8279 FIFO STATUS
0E        240F      AND     AL,0FH         ;MASK ALL BUT FIFO COUNT
10        74FB      JZ      READKEY + 3    ;KEEP READING IF ZERO
12        E82800    CALL    READATA        ;DUMMY READ TO UNLOAD THE
                                           ;PRESSED KEY FROM FIFO

          ZERO:
15        BB0000    MOV     BX,0000H       ;INITIALIZE DICE COUNT TO ZERO

          START:
18        43        INC     BX             ;INCREMENT DICE COUNT
19        80FB07    CMP     BL,07H         ;IF COUNT EQUALS 7, RESET
1C        74F7      JZ      ZERO           ;COUNT TO 0
1E        8BFB      MOV     DI,BX          ;PUT COUNT IN DI REGISTER
20        8A4D4790  MOV     CL,CDTBL[DI-1] ;PUT 7-SEGMENT DISPLAY
                                           ;CODE IN CL REGISTER,
                                           ;USE DI AS POINTER INTO
                                           ;CODE TABLE
24        BAEAFF    MOV     DX,0FFEAH      ;OUTPUT "WRITE DISPLAY"
27        B087      MOV     AL,087H        ;CONTROL WORD TO 8279
29        EE        OUT     DX
2A        BAE8FF    MOV     DX,0FFE8H      ;OUTPUT THE DICE COUNT
2D        8ACI      MOV     AL,CL          ;TO 8279 DATA PORT
2F        EE        OUT     DX
30        BAEAFF    MOV     DX,0FFEAH      ;READ 8279 FIFO STATUS
33        EC        IN      DX
34        240F      AND     AL,0FH         ;MASK ALL BUT FIFO COUNT
36        74E0      JZ      START          ;KEEP COUNTING IF NO
                                           ;KEY PRESSED
38        E80200    CALL    READATA        ;DUMMY READ TO UNLOAD
                                           ;THE PRESSED KEY FROM FIFO
3B        EBCD      JMP     READKEY        ;GO READ NEXT KEY PRESSED,
                                           ;THEN START COUNT AGAIN
```

```
                        READATA:
3D      BAEAFF          MOV     DX,0FFEAH       ;OUTPUT "READ DATA"
40      B040            MOV     AL,040H         ;CONTROL WORD TO 8279
42      EE              OUT     DX
43      BAE8FF          MOV     DX,0FFE8H       ;SET UP AND READ 8279
46      EC              IN      DX              ;DATA PORT, RESULT IN
                                                ;AL REGISTER
47      C3              RET                     ;RETURN

                        CDTBL
48      06              DB      06H             ;7-SEGMENT CODE FOR "1"
49      5B              DB      05BH            ;7-SEGMENT CODE FOR "2"
4A      4F              DB      04FH            ;7-SEGMENT CODE FOR "3"
4B      66              DB      066H            ;7-SEGMENT CODE FOR "4"
4C      6D              DB      06DH            ;7-SEGMENT CODE FOR "5"
4D      7D              DB      07DH            ;7-SEGMENT CODE FOR "6"
                DICE    ENDS
                        END
```

CHAPTER 4
SERIAL MONITOR

4-1. Introduction

This chapter defines the command set and command formats supported by the serial monitor program. The serial monitor program itself is contained in 4K bytes of ROM and is designed to operate with either a teletypewriter or CRT terminal that has been connected to serial interface EIA connector J7. (Refer to Chapter 6, Peripheral Interfacing, for connector pin assignments, cabling and board configuration.) The serial monitor performs all of the commands available with the keyboard monitor program and additionally provides the capability of reading and punching paper tape.

When power first is applied or whenever the **SYSTM RESET** key is pressed, the 8086 begins execution of the program which starts at location FF000H. If the keyboard monitor ROM's were installed in the sockets corresponding to starting location FF000H (sockets A27 and A30), the keypad monitor will be in control following power-on or reset. To permit operations using the serial monitor, you either must switch the positions of the monitor ROM sets (so that the serial monitor is in control following power-on or reset), or you must execute a Go command (from the keyboard) to location FE000H (so that control is transferred to the serial monitor's starting location).

The first 256 memory locations (locations 0H-0FFH) are reserved for the monitor and user stack area as shown in Figure 4-1. (First user-available memory location in RAM is 0100H.)

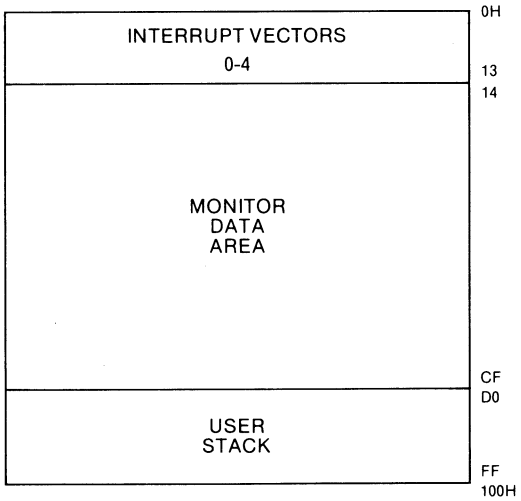


Figure 4-1. Reserved Memory

Whenever the SDK is powered up or whenever the **SYSTM RESET** key is pressed, the monitor immediately terminates its present activity and jumps to its initialization routine. This routine initializes interrupt vectors 1 through 3 as follows:

Interrupt 1: Single Step: Used with the Single Step command
Interrupt 2: NMI (Non-Maskable Interrupt): Monitors INTR key
Interrupt 3: Breakpoint: Used with the Go command

The routine also initializes the 8086's CS, DS, SS, IP and FL registers to 0H and the SP register to 0100H (base of the stack).

Whenever the monitor is re-entered as a result of a Single Step, NMI or Breakpoint interrupt, the monitor temporarily stores (pushes) the contents of the 8086 registers onto the user stack and subsequently removes (pops) the register contents from the stack before it prompts for command entry. Since the SP register is initialized to 0100H, the initial stack reserved for the user is 48 bytes (locations D0-0FFH) of which 26 bytes must be reserved for the register contents should one of the above interrupts occur.

Following power-on/reset (serial monitor at starting location FF000H) or execution of the Go command (serial monitor at starting location FE000H), the serial monitor displays its sign-on message (SDK-86 MONITOR, Vx.x) on one line and a period (".") on the next line to indicate that it is ready to accept command entry. The monitor's current version number appears in the two least-significant digits of the data field display, the numerals "86" appear in the two least-significant digits of the address field display and, with the exception of the **SYSTM RESET** and **INTR** keys, the keyboard is disabled.

4-2. Monitor Command Structure

When the monitor is ready for a command entry, it outputs a period (".") at the beginning of a new line. This line is referred to as the "command line" and consists of either a one- or two-character command mnemonic followed by from one to three command parameters or "arguments." (If desired for visual separation, a space can be entered between the command mnemonic and the first argument.) When more than one argument is required, a single comma (",") is used between arguments as a delimiter. A command line is terminated either by a carriage return or a comma, depending on the command itself. Commands are executed one at a time and only one command is permitted within a command line.

With the exception of the register abbreviations associated with the X (Examine/Modify Register) command, all arguments are entered as hexadecimal numbers. The valid range of hexadecimal values is from 00 to FF for byte entries and from 0000 to FFFF for word entries (leading zeroes may be omitted). If more than two (byte entries) or four (word entries) digits are entered, only the last two or four digits entered are valid. Address arguments consist of a segment value and an offset value. If a segment value is not entered, the default segment value is the current contents of the code segment (CS) register unless specified otherwise in the command description. When both a segment value and an offset value are entered as an address argument, the first entry is the segment value, and the second entry is the offset value. A colon (":") is entered between the entries as a separator.

Since command execution occurs only after a command terminator is entered, a command entry can be cancelled any time before the terminator is entered by pressing any character that is not legal for the entry expected. When a command is cancelled, the number symbol ("#") is output on the command line, a carriage return and a line feed are issued, and the command prompt character (".") is output on the new line.

4-3. Monitor Commands

The serial monitor is capable of executing ten individual commands. Each command is summarized in Table 4-1 and is described, in detail, within the sections which follow. In both the table and the individual command descriptions, the following command syntax is used:

[A] Indicates that "A" is optional
[A]* Indicates one or more optional occurrences of "A"
<B> Indicates that "B" is a variable
<cr> Indicates a carriage return is entered

Note that the preceding symbols are used only to clarify the command formats and that they are neither entered nor output on the console device. Also note that in the sections describing the individual commands, output from the monitor is underlined in the examples of command usage, while operator input from the console device is *not* underlined.

Table 4-1. Monitor Command Summary

COMMAND	FUNCTION/SYNTAX
S (Substitute Memory)	Displays/modifies memory locations S[W]<addr>,[[<new contents>],]*<cr>
X (Examine/Modify Register)	Displays/modifies 8086 registers X[<reg>][[<new contents>],]*<cr>
D (Display Memory)	Displays block of memory data D[W]<start addr>[,<end addr>]<cr>
M (Move)	Moves block of memory data M<start addr>,<end addr>,<destination addr ><cr>
I (Port Input)	Accepts and displays data at input port I[W]<port addr>,[,]*<cr>
O (Port Output)	Outputs data to output port O[W]<port addr>,<data>[,<data>]*<cr>
G (Go)	Transfers 8086 control from monitor to user program G[<start addr>][,<breakpoint addr>]<cr>
N (Single Step)	Executes single user program instruction N[<start addr>],[[<start addr>],]*<cr>
R (Read Hex File)	Reads hexadecimal object file from paper tape into memory R[<bias number>]<cr>
W (Write Hex File)	Outputs block of memory data to paper tape punch W[X]<start addr>,<end addr>[,<exec addr>]<cr>

4-4. Substitute Memory Command

FUNCTION

The Substitute Memory (S) command is used to examine the byte (S) or word (SW) contents of selected memory locations. If the contents of the memory location can be modified (e.g., a RAM location), the contents additionally can be updated with a new data value entered from the console device.

SYNTAX

S[W]<addr>,[[<new contents>],]*<cr>

OPERATION

To use the Substitute Memory command, enter S or SW when prompted for command entry and enter the address of the memory location to be examined. Note that if a segment address value is not specified, the current contents of the code segment (CS) register are used by default. After the address is entered, enter a comma. The monitor will then output the current data contents of the addressed memory location followed by a dash (the monitor's data entry prompt character) and a space to indicate that the addressed location is open for update. Note that when using the SW command, the byte contents of the next consecutive memory location (memory address + 1) are output first, followed by the byte contents of the actual location addressed. Similarly, when updating memory contents using the SW command, the first byte entry (first two digits) will be written into the next consecutive memory location, and the second byte entry (next two digits) will be written into the addressed memory location.

If only the one memory location is to be examined, enter a carriage return to terminate the command. (If new data was entered, it is updated when the carriage return is entered.) If a series of contiguous memory locations are to be examined and/or updated, enter a comma to advance to the next consecutive memory location (S command) or next two consecutive memory locations (SW command). Again, if the data contents are not to be updated, enter a comma to examine the next memory location, or enter the new data followed by a comma to update the current location and to examine the next location. Entering a carriage return terminates the command.

ERROR CONDITIONS

Attempting to modify a non-existent or read-only (e.g., ROM or PROM) memory location.

EXAMPLES

Example 1: Examine ROM location 0FF000H

```
_.S FF00:0, 90- <cr>
_.
```

Example 2: Examine RAM location 050H, relative to the DS register, and update the contents of location 051H to 0F7H

```
_.S DS:50, E4- ,
 0051 A4- F7<cr>
_.
```

Example 3: Examine and modify top element on stack

```
_.SW SS:SP, C98C- C98A<cr>
_.
```

Example 4: Examine RAM locations 0100H to 0107H, relative to the CS register, and modify location 0105H from 0FBH to 0BBH

```
_.SW 100, 3FF3- ,
 0102 FD7B- ,
 0104 FB98- BB98,
 0106 BFF1- <cr>
_.
```

4-5. Examine/Modify Register Command

FUNCTION

The Examine/Modify Register (X) command is used to examine and, if desired, to modify any of the 8086's individual registers or to examine the contents of all of the 8086's registers.

SYNTAX

X[<reg>][[new contents>],]*<cr>

OPERATION

To use the Examine/Modify Register command, enter X when prompted for command entry. If you only wish to examine the current contents of the registers, enter a carriage return. (The contents of all fourteen registers will be output.) If you wish to examine and optionally to modify the contents of an individual register, enter the register's abbreviation according to Table 4-2.

Table 4-2. Register Abbreviations

Register Name	Abbreviation
Accumulator	AX
Base	BX
Count	CX
Data	DX
Stack Pointer	SP
Base Pointer	BP
Source Index	SI
Destination Index	DI
Code Segment	CS
Data Segment	DS
Stack Segment	SS
Extra Segment	ES
Instruction Pointer	IP
Flag	FL

When a register abbreviation is entered, the monitor outputs an equal sign ("="), the current register contents, the data prompt character ("-") and a space. If you wish to change the register's contents, enter the new contents followed by a comma (to advance to the next "sequential" register) or a carriage return (to terminate the command). The register sequence is the order shown in Table 4-2. Note that the sequence is not circular and that if a comma is entered after the contents of the flag (FL) register are examined or modified, the monitor returns to the command mode. When a carriage return is entered, the register is updated (if new contents were entered) and the monitor returns to the command mode.

EXAMPLES

Example 1: Examine the 8086's registers

```
_.X<cr>
 AX=89D3 BX=0002 CX= 0010 DX=FFEA SP=0100 BP =D3EB SI=9295
 DI=0002  CS=0010 DS=0010 SS=0000  ES=0000 IP=000D   FL=F046
_.
```

Example 2: Modify the CS register and examine the next two registers

```
_.X CS=0000- 20,
 DS=0010- ,
 SS=0000- <cr>
_.
```

4-6. Display Memory Command

FUNCTION

The Display Memory (D) command is used to output the contents of a block of memory to the console device.

SYNTAX

D[W]<start addr>[,<end addr>]<cr>

OPERATION

The command provides a line-formatted output of the memory block bounded by *start address* and *end address* (inclusive). Since *end address* is relative to the segment address value specified or implied with *start address* (CS register contents if a segment value is not specified with *start address*), no segment value is permitted with *end address* and block transfers consequently are limited to 64K bytes or 32K words with each command execution.

To use the Display Memory command, enter D (for byte output) or DW (for word output) when prompted for command entry and then enter *start address* of the memory data block. If only one byte or word is to be displayed, enter a carriage return, while if a block of memory is to be displayed, enter *end address* and a carriage return. The monitor will output, beginning on the next line, the starting offset address, the data contents of that location and, when *end address* is specified, the data contents of a number of consecutive memory locations, each separated by a space. The line format output is arranged so that any subsequent lines (if required) will begin with the offset address of the first byte or word in the line and will consist of a maximum of either sixteen byte entries or eight word entries per line.

The Display Memory command can be cancelled or the output can be stopped at any time by entering control characters from the console device. Control-C immediately terminates the command and returns the monitor to the command entry mode. Control-S stops the output, but does not terminate the command. Control-Q resumes output that has been stopped. The only allowed console input following a Control-S is either a Control-Q or Control-C.

ERROR CONDITIONS

End address less than the offset value of *start address.*

EXAMPLES

Example 1: Display contents of locations 09H through 02AH relative to the DS register

```
_.D DS:9,2A<cr>
 0009 EE BA EA FF EC 24 0F
 0010 74 FB E8 27 00 BB 00 00 43 80 FB 07 74 F7 8B FB
 0020 8A 4D 46 BA EA FF B0 87 EE BA E8
_.
```

Example 2: Display contents of location 05FFH relative to the CS register

```
_.D 5FF<cr>
 05FF 08
_.
```

Example 3: Display contents of locations 0FF000H through 0FF02AH in word mode

```
_.DW FF00:0,2A<cr>
```

0000	E990	0098	0072	0088	4328	2029	3931	3837
0010	4920	544E	4C45	4320	524F	4050	0000	4000
0020	7F00	007D	8600	5006	0000	4000		

```
_.
```

4-7. Move Command

FUNCTION

The Move command is used to move a block of data within memory.

SYNTAX

M<start addr>,<end addr>,<destination addr><cr>

OPERATION

When using the Move command, the contents of the memory block bounded by *start address* and *end address* (inclusive) are moved to consecutive memory locations beginning at *destination address.* As with the D (Display Memory) command, *end address* is relative to the segment address value specified or implied with *start address* (if no segment value is specified, the CS register contents are used). Consequently, no segment value is permitted with *end address,* and block moves are limited to 64K bytes.

Since a move is performed one byte at a time, the Move command can be used to fill a block of memory with a predefined constant. This is accomplished by specifying a *destination address* which is one greater than *start address.* The block of memory locations from *start address* to *end address* + 1 are filled with the value contained in *start address*. (The S command is used prior to the Move command to define the constant at *start address.*)

ERROR CONDITIONS

Attempting to move data to a read-only (e.g., ROM or PROM) or non-existent memory location.

Specifying an *end address* value which is less than the offset value of *start address.*

EXAMPLES

Example 1: Move the contents of locations 0100H through 014CH to the memory block beginning at 0500H relative to the CS register

```
_.M 100,14C,500<cr>
_.
```

Example 2: Move the contents of locations 0200H through 0250H, relative to the DS register, to the memory block starting at the destination address defined by a segment value equal to the ES register plus 010H, and an offset value of 02CH

```
_.M DS:200,250,ES+10:2C<cr>
_.
```

Example 3: Fill memory locations 0300H through 0500H, relative to the CS register, with a constant of 04FH

```
_.S 300,_6C-_4F<cr>
_.M 300,4FF,301<cr>
_.
```

4-8. Port Input Command

FUNCTION

The Port Input (I) command is used to display a byte or word at an input port.

SYNTAX

I[W]<port addr>,[,]*<cr>

OPERATION

The Port Input command inputs a byte (I command) or word (IW command) from the port specified by *port address* and displays the byte or word value on the console device. Since I/O addressing is limited to 64K I/O byte addresses, no segment value is permitted with *port address.* After *port address* is entered, a comma is required to cause the byte or word at the input port to be displayed at the console. Each subsequent comma entered causes the current data at the addressed input port to be displayed on a new line. A carriage return terminates the command and causes the monitor to prompt for command entry.

When using the Port Input command to input data from the 8255A parallel I/O port circuits, refer to Table 4-3 for the assigned addresses of the individual ports. Note that these circuits are programmed for input on power-on/reset and that if they were last programmed for output, they must be reprogrammed for input (see Section 4-9, *Port Output Command*).

Table 4-3. Parallel I/O Port Addressing

Port	Address
P2A	FFF8
P1A	FFF9
P2B	FFFA
P1B	FFFB
P2C	FFFC
P1C	FFFD

When using word input (IW command), the port P2 (low-order byte) address is entered as *port address.*

EXAMPLES

Example 1: Input single word from parallel I/O ports P1A and P2A

```
_.IW FFF8,
_C7C5<cr>
_.
```

Example 2: Input multiple bytes from port 02FAH

```
_.I 2FA,
_FF,
_FC,
_00,
_B7,
_3F<cr>
_.
```

4-9. Port Output Command

FUNCTION

The Port Output (O) command is used to output a byte or word to an output port.

SYNTAX

O[W]<port addr>,<data>[,<data>]*<cr>

OPERATION

The Port Output command outputs the byte (O command) or word (OW command) entered as data to the output port specified by *port address.* Like the Port Input command, I/O addressing is limited to 64K I/O byte addresses, and no segment value is permitted with *port address.* After entering *port address,* a comma and the data to be output, a carriage return is entered to cause the data to be output to the port and to terminate the command, or a comma is entered to permit subsequent data output to the addressed port. Data can be output repetitively to the port by entering new data followed by a comma. A carriage return following a data entry outputs the data and terminates the command.

As mentioned in the previous section, the two 8255A parallel I/O port circuits are programmed for input on power-on or system reset. Consequently, to use the Port Output command to output data from the parallel I/O ports, the circuits first must be programmed for output. This is accomplished by using the Port Output command to output a control byte (or word) to the 8255A circuit's control port. Table 4-4 defines the control port addressing and the associated data byte or word to be output to the control port to program the circuits for input or output.

Table 4-4. Control Port Addressing

Port Number	Port Address	Control Byte or Word	
		Input Mode	Output Mode
P2	FFFEH	9BH	80H
P1	FFFFH	9BH	80H
P2/P1	FFFEH	9B9BH	8080H

EXAMPLES

Example 1: Program parallel I/O port P2 for output

```
_.O FFFE,80<cr>
_.
```

Example 2: Output multiple words to I/O port 020F0H

```
_.OW 20F0,BAEA,
_- 4CFF,
_- B0AE,
_- EE47,
_- F9D3<cr>
_.
```

Example 3: Output single byte to parallel I/O port P1C

```
_.O FFFD,3C<cr>
_.
```

4-10. Go Command

FUNCTION

The Go (G) command is used to transfer control of the 8086 from the serial monitor program to a user's program in memory.

SYNTAX

G[<start addr>][,<breakpoint addr>]<cr>

OPERATION

To use the Go command, enter G when prompted for command entry. The current IP (instruction pointer) register contents, the data entry prompt character and the byte contents of the location addressed by the IP register are output. If an alternate starting address is required, enter *start address.* To transfer control from the monitor to the program and to begin program execution, enter a carriage return.

To exit from the executing program and to return control to the monitor, press either the **SYSTM RESET** or the **INTR** key on the keypad. If the **SYSTM RESET** key is pressed, control is transferred to the monitor program that starts at location FF000H, and the appropriate 8086 registers are initialized. If the **INTR** key is pressed, the program is *interrupted,* the serial monitor is re-entered, *all* of the 8086 registers are saved, and the following message is output followed by a command prompt.

@aaaa:bbbb

In the above message, "aaaa" is the current CS register value, and "bbbb" is the current contents of the IP register. (The combined CS and IP register value is the address of the next program instruction to be executed when the **INTR** key was pressed.) If a subsequent Go command is entered, the current IP register contents ("bbbb") and the data byte addressed by the CS and IP registers are output. When the carriage return is entered, control is transferred back to the program, and execution resumes at the current instruction addressed.

The Go command optionally permits a "breakpoint address" to be entered. A breakpoint address has the same affect as pressing the **INTR** key while a program is being executed. Note that when specifying *breakpoint address,* the default segment value is either the *start address* segment value (if specified) or the current CS register contents (if a segment value is not specified with *start address*). When *breakpoint address* is specified, the monitor replaces the instruction at the addressed location with an interrupt instruction and saves the "breakpointed" instruction. When the program reaches *breakpoint address,* control is returned to the monitor, the breakpointed instruction is replaced in the program, all registers are saved, and the monitor outputs the following message followed by a command prompt to allow any of the registers to be examined:

BR @aaaa:bbbb

In the above message, "aaaa" is the current CS register value, and "bbbb" is the current IP register value. (The combined register value is the address of the breakpointed instruction.) If a subsequent Go command is entered, execution resumes at the replaced breakpointed instruction. Note that since the breakpointed instruction is replaced when control is returned to the monitor, *breakpoint address* must be specified each time a program to be breakpointed is executed.

ERROR CONDITIONS

Attempting to breakpoint an instruction in read-only memory.

EXAMPLES

Example 1: Transfer control to the program at 04C0H, relative to the CS register

```
-G 0000D- EC 4C0<cr>
-
```

Example 2: Transfer control to the program at 10:0H and break at the instruction in location 10:37H

```
-G 010E- 24 10:0,37<cr>
 BR @0010:0037
-
```

4-11. Single Step Command

FUNCTION

The Single Step (N) command is used to execute a single user-program instruction. With each instruction executed, control is returned to the monitor to allow evaluation of the instruction executed.

SYNTAX

N[<start addr>],[[<start addr>],]*<cr>

OPERATION

To use the Single Step command, enter N when prompted for command entry. The monitor will output the current instruction pointer (IP) register contents (the offset address of the next instruction to be executed) and the instruction byte pointed to by the IP (and CS) register. If execution of an instruction at another address is desired, enter *start address.* If *start address* includes a segment value, both the CS and IP registers are modified. When the comma is entered, the instruction addressed is executed and control is returned to the monitor. The monitor saves all of the register contents and outputs the address (IP register contents) and instruction byte contents of the next instruction to be executed on the following line. Each time a comma is entered, the addressed instruction is executed and the address and instruction byte contents of the next instruction to be executed are output on a new line.

While using the Single Step command to step through a program, a new *start address* can be entered without repeating the command entry. When the comma is entered, the instruction addressed is executed and the next instruction's address and byte contents are output. A carriage return terminates the command.

RESTRICTIONS

1. If an interrupt occurs prior to the completion of a single-stepped instruction or if a single-stepped instruction generates an interrupt, when the monitor is re-entered, the CS and IP registers will contain the address of the interrupt service routine. Consequently, a type 3 (breakpoint) interrupt instruction (0CCH or 0CDH) should not be single-stepped since its execution would step into the monitor.
2. An instruction that is part of a sequence of instructions that switches between stack segments (i.e., changes the SS and SP register contents) cannot be single-stepped.

EXAMPLES

Example 1: Single step a series of instructions beginning at 0100H, relative to the CS register

```
_.N 0000- 00 100,
  0102- 8E ,
  0104- BA ,
  0107- B0 ,
  0109- EE ,
  010A- BA <cr>
_.
```

Example 2: Single step two instructions and repeat single stepping the second instruction

```
_.N 0018- 03 ,
  001A- 40 ,
  001B- 50 1A,
  001B- 50 <cr>
_.
```

4-12. Read Hex File Command

FUNCTION

The Read Hex File (R) command allows the monitor to read an 8086 or 8080 hexadecimal object file from a paper tape and to load the data read from the file into memory.

SYNTAX

R[<bias number>]<cr>

OPERATION

To use the Read Hex File command, enter R when prompted for command entry. When the tape is loaded in the reader and ready, enter a carriage return. The data read from the file will be written into memory beginning at each record's load address. If the file is in the 8086 format and includes an execution start address record, the CS and IP registers will be updated with the execution address specified in that record. If the file is in the 8080 format and includes an EOF (end-of-file) record, the IP register is updated with the execution address specified in the EOF record. Note that a segment address value is not used with the 8080 file format; the data read is written into memory locations relative to a segment value of zero and, when an EOF record execution address is specified, the CS register is not changed.

When an optional *bias number* is specified, it is added to each record's load address to offset the file in memory.

ERROR CONDITIONS

Tape checksum error.

Attempting to load data into non-existent or read-only memory.

EXAMPLES

Example 1: Read a file and load the data into memory 256 (decimal) bytes above the load addresses specified in the file.

```
_.R 100<cr>
_.
```

4-13. Write Hex File Command

FUNCTION

The Write Hex File (W) command allows a block of memory to be output, in either 8086 or 8080 hexadecimal object file format, to a paper tape punch.

SYNTAX

W[X]<start addr>,<end addr>[,<exec addr>]<cr>

OPERATION

To use the Write Hex File command, enter W for 8086 file format or WX for 8080 file format and enter *start address* and *end address* of the memory block to be output. Note that no segment address value is permitted with *end address* (the *start address* segment value is implied) and that if no segment address value is specified with *start address,* the current CS register value is used. When the carriage return is entered, the following information is punched on the paper tape:

- Six inches of leader (60 null characters)
- An extended address record (8086 format only)
- The data contents of the memory block bounded by *start address* and *end address* (inclusive)
- An end-of-file (EOF) record
- Six inches of trailer (60 null characters)

Optionally, an *execution address* can be specified prior to entering the carriage return. This is the memory address that is loaded into the CS and IP registers (IP register only with 8080 format) when the tape is read with the R command. Depending on the format selected, when *execution address* is specified, either an execution start address record containing *execution address* is punched immediately following the tape leader (8086 format) or the offset address value of *execution address* is punched in the EOF record (8080 format).

When using the 8086 format (W command), the *start address* segment value (CS register value if a segment value is not specified) is entered (punched) in the extended address record, and the *start address* offset value is entered in the load address field of the first data record. The segment and offset address values of *execution address* are entered in the execution start address record (CS register contents if a segment address value is not specified with *execution address*).

When using the 8080 format (WX command), the *start address* offset value is punched in the load address field of the first data record. *Execution address,* if specified, is punched in the EOF record. Note that a segment address value is not permitted with *execution address* or *end address* and that the *start address* segment value is used only to define the starting address of the memory block and that it is not punched on the tape.

The Write Hex File command can be cancelled or stopped at any time by entering control characters from the console device. Control-C cancels the command and prompts for new command entry. Control-S stops the output, but does not cancel the command. Control-Q resumes output that has been stopped. The only console input allowed following a Control-S is either a Control-Q or a Control-C.

Additional information regarding Intel object file formats is available in the *MCS 80/85 Absolute Object File Formats Technical Specification,* Order Number 9800183 and the *MCS-86 Software Development Utilities Operating Instructions for ISIS-II Users,* Order Number 9800639.

ERROR CONDITIONS

Specifying a value for *end address* that is less than the offset value of *start address.*

EXAMPLES

Example 1: Output the memory block bounded by 04H and 06DDH, relative to the current CS register, to an 8086 file with an execution address of CS:040H

```
_.W 4,6DD,40<cr>
_.
```

Example 2: Output the memory block bounded by FF200H and FF2FFH to an 8080 file with a starting load address of 0100H and an execution address of 011AH

```
_.WX FF10:100,1FF,11A<cr>
_.
```

CHAPTER 5
SERIAL LOADER

5-1. Introduction

When an SDK-86 is interfaced to an Intel Intellec microcomputer development system, the serial loader program that is supplied with the SDK-C86 kit adds two commands to the serial monitor's command set. These commands allow a user to load an ISIS-II file into SDK memory and to transfer the contents of SDK memory to an ISIS-II file.

With the exception of the R and W commands, all of the serial monitor's commands described in Chapter 4 are available with the serial loader program. Except for the operation of the serial loader's load and transfer commands, serial monitor operation with the development system is identical to its operation with a teletypewriter or CRT terminal; characters entered at the system console are input to the SDK and characters output from the SDK are displayed on the system console. The serial loader has the capability of detecting when the serial monitor is prompting for command entry in order to "intercept" a load or transfer command entered from the system console.

When the serial loader intercepts a load command, it opens the specified file for reading and then issues an R (read hex file) command to the,serial monitor followed by the data read from the file. When the serial loader intercepts a transfer command, it opens the specified file for writing and issues a W (write hex file) command and the associated command arguments to the serial monitor. The serial monitor responds by outputting the requested data to the serial port, and the serial loader writes the data into the opened file.

5-2. System Operation

Unlike the two monitor programs that reside in ROM, the serial loader program resides on a diskette. Two diskettes are supplied with the SDK-C86 kit. (The contents of both diskettes are identical.) The diskette labeled S.D.D.A. ISIS-II SDK-86 LOADER (part number 9500044) is for single-density drives, and the diskette labeled D.D.D.A. ISIS-II SDK-86 LOADER (part number 9700040) is for double-density drives. Insert the appropriate diskette into the secondary drive and, using the ISIS-II load-and-go command, enter:

```
:Fn:SDK86
```

where n is the drive unit number of the secondary drive. The serial loader will sign-on with the message:

```
ISIS-II SDK-86 LOADER, Vx.x
```

After the serial loader has been loaded (after the sign-on message is displayed) either press the SDK's **SYSTM RESET** key if the serial monitor is located at FF000H or, if the serial monitor is located at FE000H, execute a Go command from the keypad monitor to transfer control to the serial monitor. The serial monitor's sign-on message (SDK-86 MONITOR, Vx.x) will be displayed on the system console followed by the command prompt character (".") to indicate that the system is ready to accept command entry.

5-3. Loader Commands

As previously mentioned, the serial loader adds two commands to the serial monitor. These commands (Load Hex File and Transfer Hex File) follow the same syntax conventions described in Chapter 4. Additionally, an Exit command is included with the serial loader to return control to ISIS. Table 5-1 defines the serial loader commands and syntax.

Table 5-1. Serial Loader Command Summary

Command	Function/Syntax
T (Transfer Hex File)	Tranfers memory block from SDK to ISIS-II file T[X]<start addr>,<end addr>,<filename>[,<exec addr>]<cr>
L (Load Hex File)	Loads ISIS-II file into SDK memory L<filename>[,<bias-number>]<cr>
E (Exit)	Returns control to ISIS E<cr>

5-4. Transfer Hex File Command

FUNCTION

The Transfer Hex File (T) command is used to transfer the contents of a block of SDK memory to an ISIS-II hexadecimal object file in the microcomputer development system.

SYNTAX

T[X]<start addr>,<end addr>,<filename>[,< exec addr>]<cr>

OPERATION

To use the Transfer Hex File command, enter T for 8086 file format or TX for 8080 file format and *start address* and *end address* of the memory block to be transferred and the *filename* of the ISIS-II file to receive the memory data. Note that no segment address value is permitted with *end address* (the *start address* segment value is implied) and that if no segment address value is specified with *start address,* the current CS register value is used. When the carriage return is entered, the data contents of the memory block bounded by *start address* and *end address* (inclusive) are transferred to the file specified.

Optionally, an *execution address* can be specified prior to entering the carriage return. This is the memory address that is loaded into the CS and IP registers (IP register only with 8080 format) when the file is loaded with the L command. Depending on the format selected, when an *execution address* is specified, either an execution start address record containing *execution address* is entered in the file (8086 format) or the offset address value of *execution address* is entered in the EOF record (8080 format).

When using the 8086 format (T command), the *start address* segment value (CS register value if a segment value is not specified) is entered in the file's extended address record, and the *start address* offset value is entered in the load address field of the first data record. The segment and offset address values of *execution address* are entered in the execution start address record (CS register contents if a segment address value is not specified with *execution address*).

When using the 8080 format (TX command), the *start address* offset value is entered in the load address field of the first data record. The *execution address,* if specified, is entered in the EOF record. Note that a segment address value is not permitted with *execution address* or *end address* and that the *start address* segment value is used only to define the starting address of the memory block and that it is not entered in the file.

ERROR CONDITIONS

End address specified is less than the offset value of *start address*

ISIS-II type errors (1-99)

Timeout

Checksum error

EXAMPLES

Example 1: Transfer, in the 8086 file format, the SDK memory contents between locations 0200H and 0600H, relative to the current CS register, into the ISIS-II file named :F2:GARYL

```
-T 200,600,:F2:GARYL<cr>
-
```

5-5. Load Hex File Command

FUNCTION

The Load Hex File command loads an ISIS-II hexadecimal object file from the microcomputer development system into SDK memory.

SYNTAX

L<filename>[,<bias number>]<cr>

OPERATION

To use the Load Hex File command enter L and the ISIS-II *filename* of the file to be loaded. The data read from the specified file will be written into memory beginning at each record's load address. If the file is in the 8086 format and includes an execution start address record, the CS and IP registers will be updated with the execution address specified in that record. If the file is in the 8080 format and includes an EOF (end-of-file) record, the IP register is updated with the execution address specified in the EOF record. Note that a segment address value is not used with the 8080 file format; the data read is written into memory locations relative to a segment value of zero and, when an EOF record execution address is specified, the CS register is not changed.

When an optional *bias number* is specified, it is added to each record's load address to offset the file in memory.

ERROR CONDITIONS

Attempting to load data into non-existent or read-only memory

ISIS-II type errors (1-99)

Timeout

Checksum error

EXAMPLES

Example 1: Load the ISIS-II file named F1:GEL.HEX into SDK memory, 0100H locations above the load addresses in the file.

```
-L F1:GEL.HEX, 100<cr>
-
```

5-6. Exit Command

FUNCTION

The exit command transfers control from the monitor to ISIS.

SYNTAX

```
E<cr>
```

5-7. System I/O Routines

The SDK-86 loader diskette also includes two libraries containing I/O routines for the console and teletypewriter paper tape reader and punch. Although the routines in both libraries have the same names and functions, two libraries are necessary to support the two types of subroutine linkages provided by the 8086 architecture. The routines in SDKIOS.LIB are called with *intra*segment subroutine calls (a PL/M-86 module compiled with the "small" control generates this type of call). The routines in SDKIOL.LIB are called with *inter*segment subroutine calls (a PL/M-86 module compiled with either the "medium" or "large" control generates this type of call). The routines in both libraries are written in PL/M-86. The modules in SDKIOS.LIB are compiled with the "small" control, and the modules in SDKIOL.LIB are compiled with the "large" control. The names assigned to the segments, classes and groups are the standard names generated by the PL/M-86 compiler. (See *MCS-86 Software Development Utilities Operating Instructions for ISIS-II Users,* Order Number 9800639.)

When using the serial loader program, the console input and output routines provided in the library should be used for console I/O. (The loader has special requirements which are met by the library routines.) The paper tape routines (RI and PO) only perform I/O when a teletypewriter is connected directly to the SDK-86's serial interface port.

5-8. Console Input Routine

The console input routine (CI) returns an 8-bit character received from the system console to the caller in the AL register. The AX, CX and DX registers and CPU condition codes are affected by this operation.

When a Control-S (13H) or Control-Q (11H) character is read, it is discarded and another character is read. This function is necessary to support the control character feature of the CO routine.

EXAMPLES

Example 1: PL/M-86 CI Call

```
CI: PROCEDURE BYTE EXTERNAL;
  END CI;

DECLARE CHAR BYTE;
    .
    .
    .
CHAR = CI AND 7FH;      /* INPUT CHARACTER AND STRIP PARITY BIT */
    .
    .
    .
```

Example 2: ASM86 CI Intrasegment Call

```
            ASSUME   DS:DATA,SS:STACK,CS:CODE

DATA        SEGMENT  PUBLIC 'DATA'
CHAR        DB       ?
DATA        ENDS

STACK       SEGMENT  STACK 'STACK'
            DW       15 DUP ?
BASESTACK   LABEL    WORD
STACK       ENDS

CODE        SEGMENT  PUBLIC 'CODE'
            EXTRN    CI:NEAR
INIT:
            MOV      AX,STACK              ; INITIALIZE
            MOV      SS,AX                 ;   SS
            MOV      SP,OFFSET BASESTACK   ; INITIALIZE SP
            MOV      AX,DATA               ; INITIALIZE
            MOV      DS,AX                 ;   DS
            .
            .
            .
            CALL     CI                    ; INPUT CHARACTER FROM CONSOLE
            AND      AL,7FH                ; STRIP OFF PARITY BIT
            MOV      CHAR,AL               ; SAVE CHARACTER
            .
            .
            .
CODE        ENDS
            END      INIT
```

Example 3: ASM86 CI Intersegment Call

```
            EXTRN     CI:FAR

            ASSUME    DS:MYDATA,SS:STACK,CS:MYCODE

MYDATA      SEGMENT   'DATA'
CHAR        DB        ?
MYDATA      ENDS

STACK       SEGMENT   STACK 'STACK'
            DW        16 DUP ?
BASESTACK   LABEL     WORD
STACK       ENDS

MYCODE      SEGMENT   'CODE'
INIT:
            MOV       AX,STACK              ; INITIALIZE
            MOV       SS,AX                 ;   SS
            MOV       SP,OFFSET BASESTACK   ; INITIALIZE SP
            MOV       AX,MYDATA             ; INITIALIZE
            MOV       DS,AX                 ;   DS
            .
            .
            .
            CALL      CI                    ; INPUT CHARACTER FROM CONSOLE
            AND       AL,7FH                ; STRIP OFF PARITY BIT
            MOV       CHAR,AL               ; SAVE CHARACTER
            .
            .
            .
MYCODE      ENDS
            END       INIT
```

5-9. Console Output Routine

The console output routine (CO) transmits an 8-bit character, passed from the caller on the stack in the low-order byte, to the system console. The AX, CX and DX registers and the CPU conditions codes are affected by this operation.

Before the character is output, the console output routine checks to see if a Control-S has been input. If a Control-S has been input, the monitor waits until a Control-Q is input before outputting the character. Consequently, if a Control-S is entered at the console when console output is pending, the output is stopped temporarily. Entering Control-Q resumes output.

EXAMPLES

Example 1: PL/M-86 CO Call

```
CO:PROCEDURE(X) EXTERNAL;
   DECLARE X BYTE;
   END CO;

DECLARE CHAR BYTE;
   .
   .
   .
CALL CO(CHAR);      /* OUTPUT CHARACTER */
   .
   .
   .
```

Example 2: ASM86 CO Intrasegment Call

```
                ASSUME  DS:DATA,SS:STACK,CS:CODE

DATA            SEGMENT PUBLIC 'DATA'
CHAR            DB      ?
DATA            ENDS

STACK           SEGMENT STACK 'STACK'
                DW      15 DUP ?
BASESTACK       LABEL   WORD
STACK           ENDS

CODE            SEGMENT PUBLIC 'CODE'
                EXTRN   CO:NEAR
INIT:
                MOV     AX,STACK                ; INITIALIZE
                MOV     SS,AX                   ;   SS
                MOV     SP,OFFSET BASESTACK     ; INITIALIZE SP
                MOV     AX,DATA                 ; INITIALIZE
                MOV     DS,AX                   ;   DS
                .
                .
                .
                MOV     AL,CHAR                 ; LOAD CHARACTER INTO AL
                PUSH    AX                      ; PUSH CHARACTER ONTO STACK
                CALL    CO                      ; OUTPUT CHARACTER TO CONSOLE
                .
                .
                .
CODE            ENDS
                END     INIT
```

Example 3: ASM86 CO Intersegment Call

```
                EXTRN   CO:FAR
                ASSUME  DS:MYDATA,SS:STACK,CS:MYCODE

MYDATA          SEGMENT 'DATA'
CHAR            DB      ?
MYDATA          ENDS

STACK           SEGMENT STACK 'STACK'
                DW      16 DUP ?
BASESTACK       LABEL   WORD
STACK           ENDS

MYCODE          SEGMENT 'CODE'
INIT:
                MOV     AX,STACK                ; INITIALIZE
                MOV     SS,AX                   ;   SS
                MOV     SP,OFFSET BASESTACK     ; INITIALIZE SP
                MOV     AX,MYDATA               ; INITIALIZE
                MOV     DS,AX                   ;   DS
                .
                .
                .
                MOV     AL,CHAR                 ; LOAD CHARACTER INTO AL
                PUSH    AX                      ; PUSH CHARACTER ONTO STACK
                CALL    CO                      ; OUTPUT CHARACTER TO CONSOLE
                .
                .
                .
MYCODE          ENDS
                END     INIT
```

5-10. Reader Input Routine

The reader input routine, (RI) returns an 8-bit character received from the paper tape reader to the caller in the AL register. The AX, CX and DX registers and the CPU condition codes are affected by this operation. If a character is not received within two seconds, the carry flag of the FL register is set on return.

EXAMPLES

Example 1: PL/M-86 RI Call

```
RI: PROCEDURE BYTE EXTERNAL;
  END RI;

DECLARE CHAR BYTE;
    .
    .
    .
CHAR = RI;      /* READ CHARACTER */
IF CARRY THEN GOTO END$OF$FILE;
CHAR = CHAR AND 7FH;      /* STRIP PARITY BIT */
    .
    .
    .
```

Example 2: ASM86 RI Intrasegment Call

```
            ASSUME   DS:DATA,SS:STACK,CS:CODE

DATA        SEGMENT  PUBLIC 'DATA'
CHAR        DB       ?
DATA        ENDS

STACK       SEGMENT  STACK 'STACK'
            DW       15 DUP ?
BASESTACK   LABEL    WORD
STACK       ENDS

CODE        SEGMENT  PUBLIC 'CODE'
            EXTRN    RI:NEAR
INIT:
            MOV      AX,STACK              ; INITIALIZE
            MOV      SS,AX                 ;   SS
            MOV      SP,OFFSET BASESTACK   ; INITIALIZE SP
            MOV      AX,DATA               ; INITIALIZE
            MOV      DS,AX                 ;   DS
            .
            .
            .
            CALL     RI                    ; INPUT CHARACTER FROM READER
            JB       DONE                  ; EXIT IF NO MORE
            AND      AL,7FH                ; STRIP OFF PARITY BIT
            MOV      CHAR,AL               ; SAVE CHARACTER
            .
            .
            .
CODE        ENDS
            END      INIT
```

Example 3: ASM86 RI Intersegment Call

```
            EXTRN    RI:FAR
            ASSUME   DS:MYDATA,SS:STACK,CS:MYCODE

MYDATA      SEGMENT  'DATA'
CHAR        DB       ?
MYDATA      ENDS

STACK       SEGMENT  STACK 'STACK'
            DW       16 DUP ?
BASESTACK   LABEL    WORD
STACK       ENDS

MYCODE      SEGMENT  'CODE'
INIT:
            MOV      AX,STACK              ; INITIALIZE
            MOV      SS,AX                 ;   SS
            MOV      SP,OFFSET BASESTACK   ; INITIALIZE SP
            MOV      AX,MYDATA             ; INITIALIZE
            MOV      DS,AX                 ;   DS
            .
            .
            .
            CALL     RI                    ; INPUT CHARACTER FROM READER
            JB       DONE                  ; EXIT IF NO MORE
            AND      AL,7FH                ; STRIP OFF PARITY BIT
            MOV      CHAR,AL               ; SAVE CHARACTER
            .
            .
            .
MYCODE      ENDS
            END      INIT
```

5-11. Punch Output Routine

The punch output routine (PO) outputs an 8-bit character, passed from the caller on the stack in the low-order byte, to the paper tape punch. The AX and DX registers and the CPU condition codes are affected by this operation.

EXAMPLES

Example 1: PL/M-86 PO Call

```
PO: PROCEDURE(X) EXTERNAL;
   DECLARE X BYTE;
   END PO;

DECLARE CHAR BYTE;
      .
      .
      .
CALL PO(CHAR);      /* PUNCH CHARACTER */
      .
      .
      .
```

Example 2: ASM86 PO Intrasegment Call

```
            ASSUME   DS:DATA,SS:STACK,CS:CODE

DATA        SEGMENT  PUBLIC 'DATA'
CHAR        DB       ?
DATA        ENDS
```

```
STACK       SEGMENT  STACK 'STACK'
            DW       15 DUP ?
BASESTACK   LABEL    WORD
STACK       ENDS

CODE        SEGMENT  PUBLIC 'CODE'
            EXTRN    PO:NEAR
INIT:
            MOV      AX,STACK              ; INITIALIZE
            MOV      SS,AX                 ;   SS
            MOV      SP,OFFSET BASESTACK   ; INITIALIZE SP
            MOV      AX,DATA               ; INITIALIZE
            MOV      DS,AX                 ;   DS
            .
            .
            .
            MOV      AL,CHAR               ; LOAD CHARACTER INTO AL
            PUSH     AX                    ; PUSH CHARACTER ONTO STACK
            CALL     PO                    ; OUTPUT CHARACTER TO PUNCH
            .
            .
            .
CODE        ENDS
            END      INIT
```

Example 3: ASM86 PO Intersegment Call

```
            EXTRN    PO:FAR

            ASSUME   DS:MYDATA,SS:STACK,CS:MYCODE

MYDATA      SEGMENT  'DATA'
CHAR        DB       ?
MYDATA      ENDS

STACK       SEGMENT  STACK 'STACK'
            DW       16 DUP ?
BASESTACK   LABEL    WORD
STACK       ENDS

MYCODE      SEGMENT  'CODE'
INIT:
            MOV      AX,STACK              ; INITIALIZE
            MOV      SS,AX                 ;   SS
            MOV      SP,OFFSET BASESTACK   ; INITIALIZE SP
            MOV      AX,MYDATA             ; INITIALIZE
            MOV      DS,AX                 ;   DS
            .
            .
            .
            MOV      AL,CHAR               ; LOAD CHARACTER INTO AL
            PUSH     AX                    ; PUSH CHARACTER ONTO STACK
            CALL     PO                    ; OUTPUT CHARACTER TO PUNCH
            .
            .
            .
MYCODE      ENDS
            END      INIT
```

6-1. Introduction

This chapter provides the necessary information for connecting a teletypewriter/CRT terminal or an Intel Intellec microcomputer development system to the SDK-86's serial interface port. Additionally, the connector pin assignments for the two parallel I/O ports (P1 and P2) and the bus expansion interface are provided.

6-2. Serial Interface Port

The serial interface port uses a DB-25S female type 25 pin EIA connector (J7). The required mating connector (not supplied) is a DB-25P. The pin assignments for the connector are determined by installing shorting plugs at 2- by 18-pin header W1-W18 (located just below serial interface connector J7). The two rows of header pins are grouped into four sets as noted by the silkscreening on the board (TTY, CRT, MDS-TTY and MDS-CRT). Depending on the device to be interfaced, shorting plugs are inserted over all of the pins in the appropriate set. Table 6-1 defines the connector pin assignments and signal names for each set.

Table 6-1. Serial Interface Connector J7 Pin Assignments

PIN	HEADER W1-W18 SHORTING PLUG CONFIGURATION			
	TTY	CRT	MDS-TTY	MDS-CRT
1				
2		RECEIVE DATA		TRANSMIT DATA
3		TRANSMIT DATA		RECEIVE DATA
4				
5				
6				
7	GROUND	GROUND	GROUND	GROUND
8				
9				
10				
11				
12	RECEIVE DATA		TRANSMIT DATA	
13	TRANSMIT DATA		RECEIVE DATA	
14				
15				
16	READER CONTROL			
17				
18				
19				
20				
21	READER CONTROL RETURN			
22				
23				
24	RECEIVE DATA RETURN			
25	TRANSMIT DATA RETURN			

The SDK-86's baud rate must be set to match the baud rate of the device connected to the serial interface port. The SDK-86's baud rate is determined by installing a shorting plug at BAUD RATE SELECT header W19-W26 (located towards the center of the Serial Interface area). The available baud rates are individually silkscreened adjacent to their associated header pins. To select the baud rate, simply install a shorting plug over the corresponding pair of header pins. Note that for 110 baud, two shorting plugs are installed.

6-3. Microcomputer Development System Interface

The SDK-C86 System Design Kit includes a cable for connecting the SDK-86 to any of the Intel Intellec microcomputer development systems. This cable (Intel part number 4001927) is installed between serial interface connector J7 on the SDK-86 and the appropriate connector on the rear of the development system enclosure. Figure 6-1 shows the interface configurations for the various systems and console devices. Referring to the figure, the shorting plug set to be installed on header W1-W18 is shown within the SDK-86 outline as is the required baud rate.

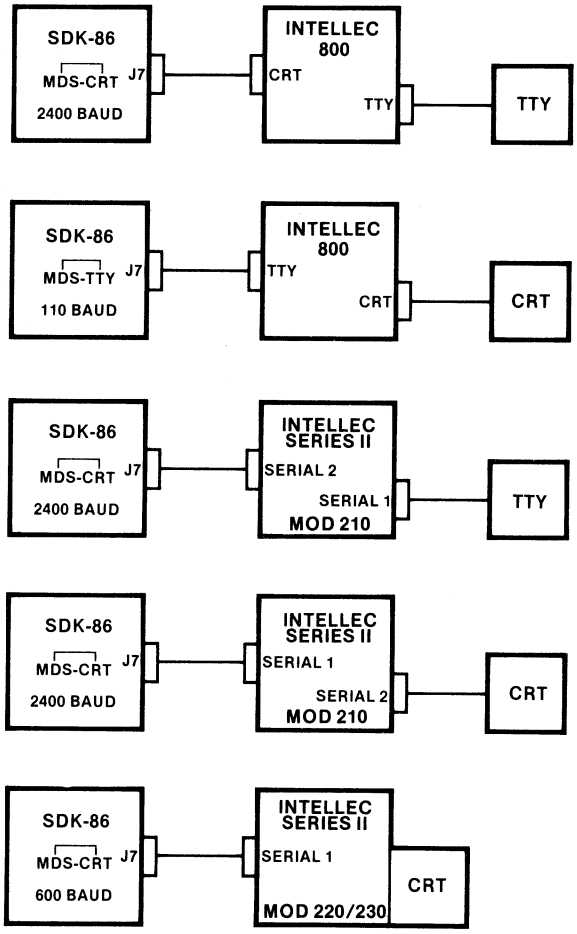


Figure 6-1. Microcomputer Development System Interfacing

6-4. Parallel I/O Ports

The input/output signals for the two 8255A parallel I/O port circuits are available at the two sets of double row pads to the left of the circuits. As indicated by the silkscreen, the set of pads at J5 corresponds to port circuit P2, and the set of pads at J6 corresponds to port circuit P1. The individual 8-bit ports (A, B and C) also are silkscreened. The pin assignments for the individual port bits are defined in Tables 6-2 (Port 2-J5) and 6-3 (Port 1-J6).

If an off-board peripheral device is to be interfaced to the parallel I/O ports, the 2-by 25-pin headers supplied with the kit can be installed to permit ribbon cable and mass-termination type connectors to be used. Conversely, if the parallel I/O ports are to be interfaced to circuitry installed in the user design area, wire-wrap pins (not supplied) can be installed in the individual pads (from the bottom of the board) to allow wire-wrap connections into the user design area.

Table 6-2. Parallel I/O Port Connector J5 Pin Assignments

Pin	Function	Pin	Function
1	Ground	34	Port P2A bit 0
3		38	Port P2A bit 1
5		42	Port P2A bit 2
7		46	Port P2A bit 3
9		48	Port P2A bit 4
11		44	Port P2A bit 5
13		40	Port P2A bit 6
15	Ground	36	Port P2A bit 7
17	Ground	24	Port P2C bit 0
19		2	Port P2C bit 1
21		4	Port P2C bit 2
23		6	Port P2C bit 3
25		26	Port P2C bit 4
27		28	Port P2C bit 5
29		30	Port P2C bit 6
31	Ground	32	Port P2C bit 7
33	Ground	10	Port P2B bit 0
35		22	Port P2B bit 1
37		18	Port P2B bit 2
39		14	Port P2B bit 3
41		16	Port P2B bit 4
43		20	Port P2B bit 5
45		12	Port P2B bit 6
47	Ground	8	Port P2B bit 7
49	Ground	50	Not Used

Table 6-3. Parallel I/O Port Connector J6 Pin Assignments

Pin	Function	Pin	Function
1	Ground	36	Port P1A bit 0
3		40	Port P1A bit 1
5		44	Port P1A bit 2
7		48	Port P1A bit 3
9		50	Port P1A bit 4
11		46	Port P1A bit 5
13		42	Port P1A bit 6
15	Ground	38	Port P1A bit 7
17	Ground	26	Port P1C bit 0
19		24	Port P1C bit 1
21		22	Port P1C bit 2

Table 6-3. Parallel I/O Port Connector J6 Pin Assignments

Pin	Function	Pin	Function
23	↑	20	Port P1C bit 3
25	↑	28	Port P1C bit 4
27	↑	30	Port P1C bit 5
29	↑	32	Port P1C bit 6
31	Ground	34	Port P1C bit 7
33	Ground	16	Port P1B bit 0
35	↑↓	12	Port P1B bit 1
37	↑↓	8	Port P1B bit 2
39	↑↓	4	Port P1B bit 3
41	↑↓	6	Port P1B bit 4
43	↑↓	10	Port P1B bit 5
45	↑↓	14	Port P1B bit 6
47	Ground	18	Port P1B bit 7
49	Ground	2	Not Used

6-5. Bus Expansion Interface

The bus expansion logic, which allows the CPU bus to be extended into the user design area as well as off the board, uses the two sets of double row pads adjacent to the user design area for interfacing. The set of pads designated J1 and J2 provides access to the CPU's data bus and certain control signals while the set of pads designated J3 and J4 provides access to the CPU's 20-bit address bus and the remaining control signals. The signals appearing at J3 and J4 are pin-for-pin identical, while five additional signals are available at J2 that are not available at J1 (the remaining signals are pin-for-pin identical). Table 6-4 lists the pin assignments and corresponding silkscreen signal nomenclature. Note that the signal names in parentheses are the actual signal names appearing on the schematics.

Table 6-4. Bus Expansion Pin Assignments

Pin*	Connector J1/J2 Signal Name	Connector J3/J4 Signal Name
2	D0 (BD0)	$\overline{BHE}$
4	D1 (BD1)	A0
6	D2 (BD2)	A1
8	D3 (BD3)	A2
10	D4 (BD4)	A3
12	D5 (BD5)	A4
14	D6 (BD6)	A5
16	D7 (BD7)	A6
18	D8 (BD8)	A7
20	D9 (BD9)	A8
22	D10 (BD10)	A9
24	D11 (BD11)	A10
26	D12 (BD12)	A11
28	D13 (BD13)	A12
30	D14 (BD14)	A13
32	D15 (BD15)	A14
34	$\overline{RST}$ (RESET OUT)	A15
36	$\overline{PCLK}$	A16
38	INTR	A17
40	TEST	A18
42	HOLD	A19

*All undesignated pins are logic ground.

Table 6-4. Bus Expansion Pin Assignments (Cont'd.)

Pin*	Connector J1/J2 Signal Name	Connector J3/J4 Signal Name
44	HLDA (BHLDA)	M/$\overline{IO}$ (BM/$\overline{IO}$)
46	$\overline{DEN}$ (BDEN/)	$\overline{RD}$ (BRD/)
48	DT/$\overline{R}$ (BDT/$\overline{R}$)	$\overline{WR}$ (BWR/)
50	ALE (BALE)	$\overline{INTA}$ (BINTA/)
J2-41	$\overline{CSX}$	
J2-43	$\overline{CSY}$	
J2-45	S3 (BS3)	
J2-47	S4 (BS4)	
J2-49	S5 (BS5)	

*All undesignated pins are logic ground.

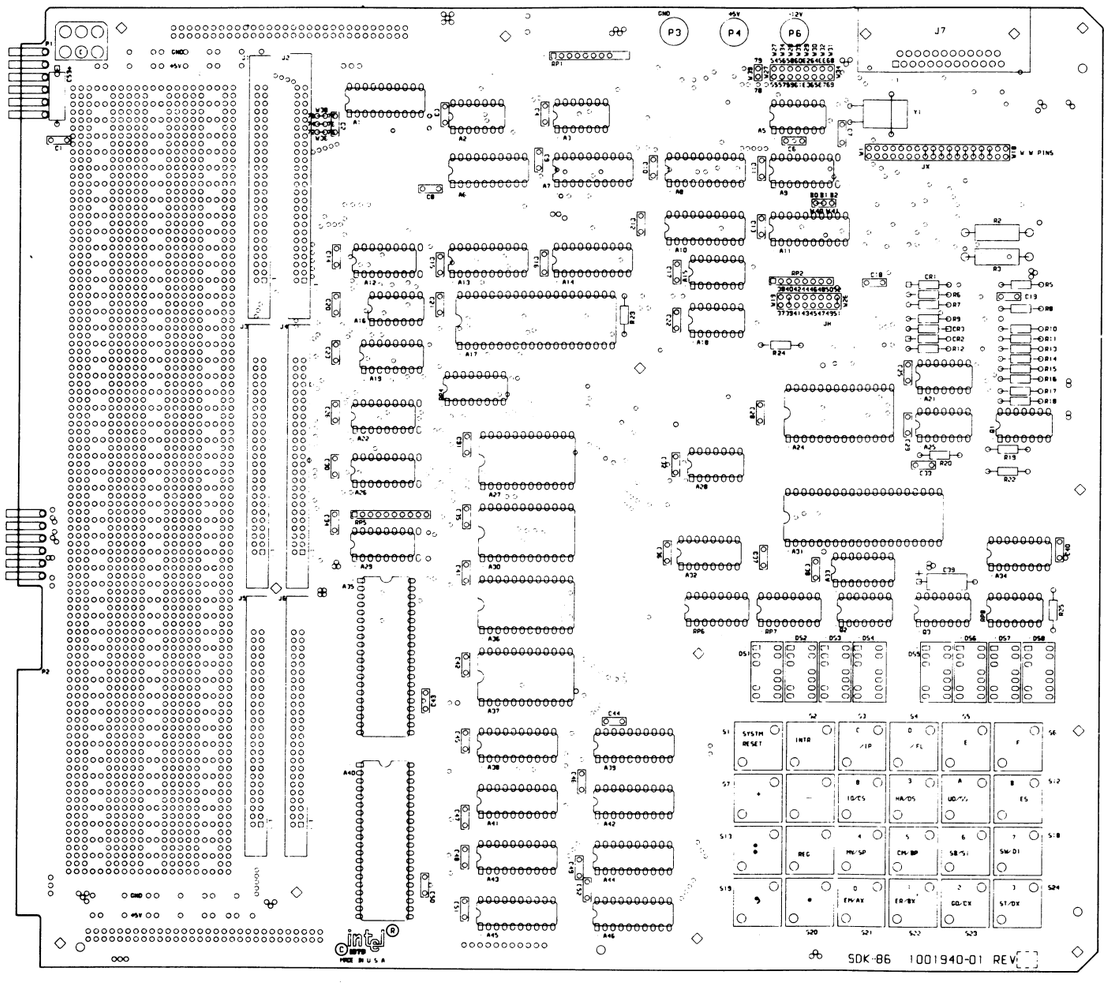
Since the two sets of signals at J1/J2 and J3/J4 are identical (except for the additional signals at J2), the CPU bus can be routed off-board by installing one of the supplied 2- by 25-pin headers into one of the double row pads in each set (e.g., J2 and J4) and also can be wired into the user design area through the other set (e.g., J1 and J3).

Note that the INTR, TEST and HOLD signals were disabled when the kit was assembled (shorting plugs were installed in W36, W37 and W38) and that if the circuit to be interfaced to the CPU bus requires any of these signals, the corresponding shorting plugs must be removed.

CHAPTER 7
SCHEMATICS AND RELATED DRAWINGS

The schematics and related drawings on the following pages are for reference only and reflect the SDK-86 design level at the time this manual was printed. The schematics and related drawings included in the kit's literature package reflect the current design level and, when not identical to the corresponding drawings in the manual, supercede the manual drawing.

DWG NO. E1001940 | SH 1 | REV 2



QUANTITY PER DASH NO.	ITEM NO.	PART NUMBER	DESCRIPTION
		PARTS LIST	

UNLESS OTHERWISE SPECIFIED:
1. DIMENSIONS ARE IN INCHES.
2. BREAK ALL SHARP EDGES.
3. DO NOT SCALE DRAWING.
4. TOLERANCES:
ANGLES ± 2°
XX ± .030
XXX ± .010
SURFACE FINISH √

SIGNATURE	DATE
DRN BY	
CHK BY	
ENGR APVD	
APVD	
APVD	

intel® 3065 BOWERS AVE. SANTA CLARA CALIF. 95051

TITLE: PWA SYSTEM DESIGN KIT

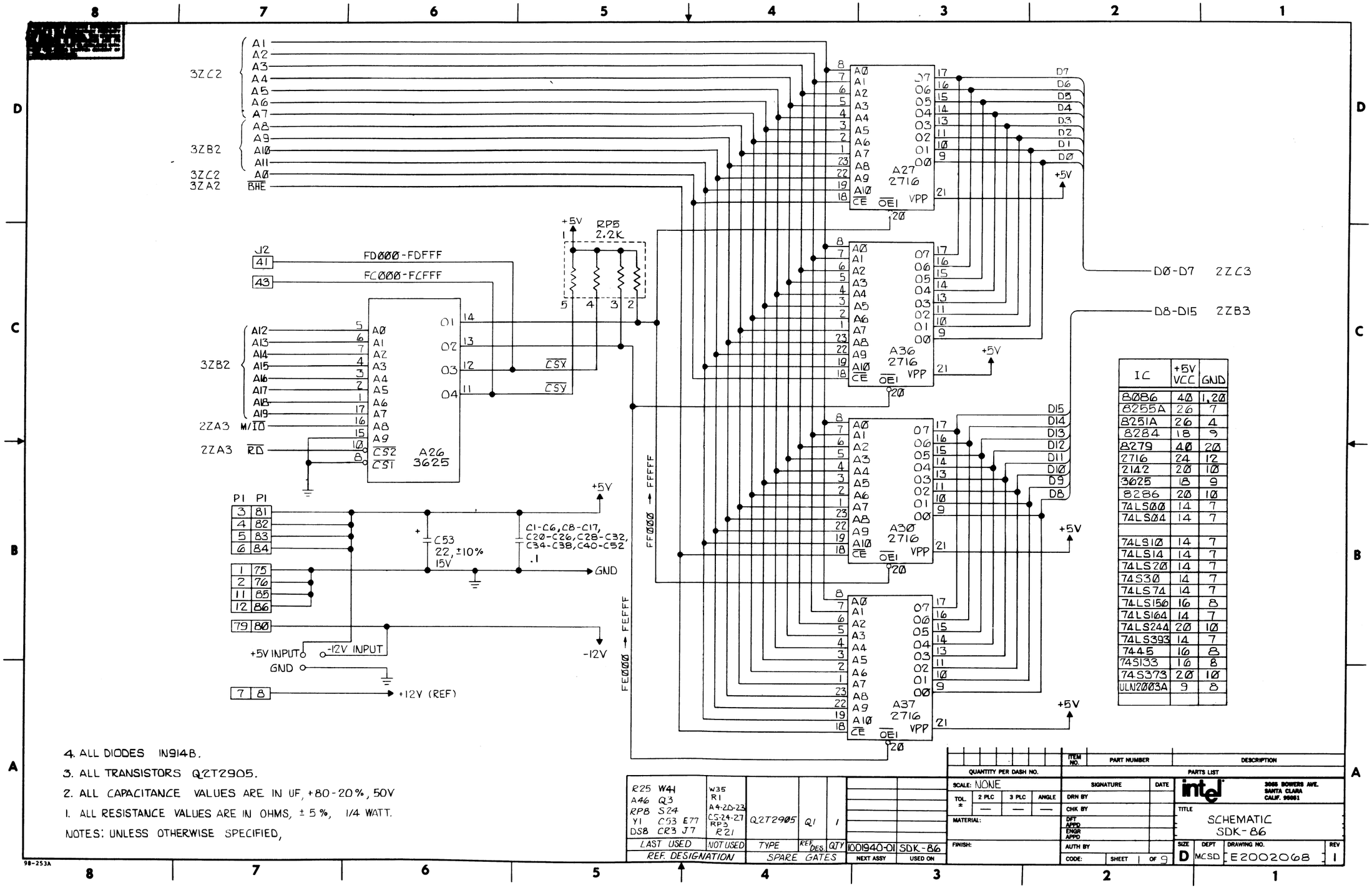
SIZE D | CODE MCSD | DWG. NO. E1001940 | REV 2

SCALE NONE | SHEET 1 OF 1

400193S | SDK-86
NEXT ASSY | USED ON

9366621

Figure 7-1. Parts Location Diagram



IC	+5V VCC	GND
8086	40	1,20
8255A	26	7
8251A	26	4
8284	18	9
8279	40	20
2716	24	12
2142	20	10
3625	18	9
8286	20	10
74LS00	14	7
74LS04	14	7
74LS10	14	7
74LS14	14	7
74LS20	14	7
74S30	14	7
74LS74	14	7
74LS156	16	8
74LS164	14	7
74LS244	20	10
74LS393	14	7
7445	16	8
74S133	16	8
74S373	20	10
ULN2003A	9	8

4. ALL DIODES IN914B.
3. ALL TRANSISTORS Q2T2905.
2. ALL CAPACITANCE VALUES ARE IN UF, +80-20%, 50V
1. ALL RESISTANCE VALUES ARE IN OHMS, ± 5%, 1/4 WATT.
NOTES: UNLESS OTHERWISE SPECIFIED,

REF. DESIGNATION	SPARE GATES	NEXT ASSY	USED ON
R25 W41 A46 Q3 RP8 S24 Y1 C53 E77 DS8 CR3 J7	W35 R1 A4-20-23 C5-24-27 RP3 R21	1001940-01	SDK-86
LAST USED: NOT USED	TYPE: Q2T2905 REF DES: Q1 QTY: 1		

QUANTITY PER DASH NO.

ITEM NO. | PART NUMBER | DESCRIPTION — PARTS LIST

SCALE: NONE

TOL. ± | 2 PLC | 3 PLC | ANGLE

MATERIAL:

FINISH:

SIGNATURE | DATE

DRN BY
CHK BY
DFT APPD
ENGR APPD
AUTH BY
CODE: | SHEET | OF 9

intel 3065 BOWERS AVE. SANTA CLARA CALIF. 95051

TITLE: SCHEMATIC SDK-86

SIZE: D | DEPT: MCSD | DRAWING NO.: E2002068 | REV: 1

98-253A

Figure 7-2. SDK-86 Schematic Diagram (Sheet 1 of 9)

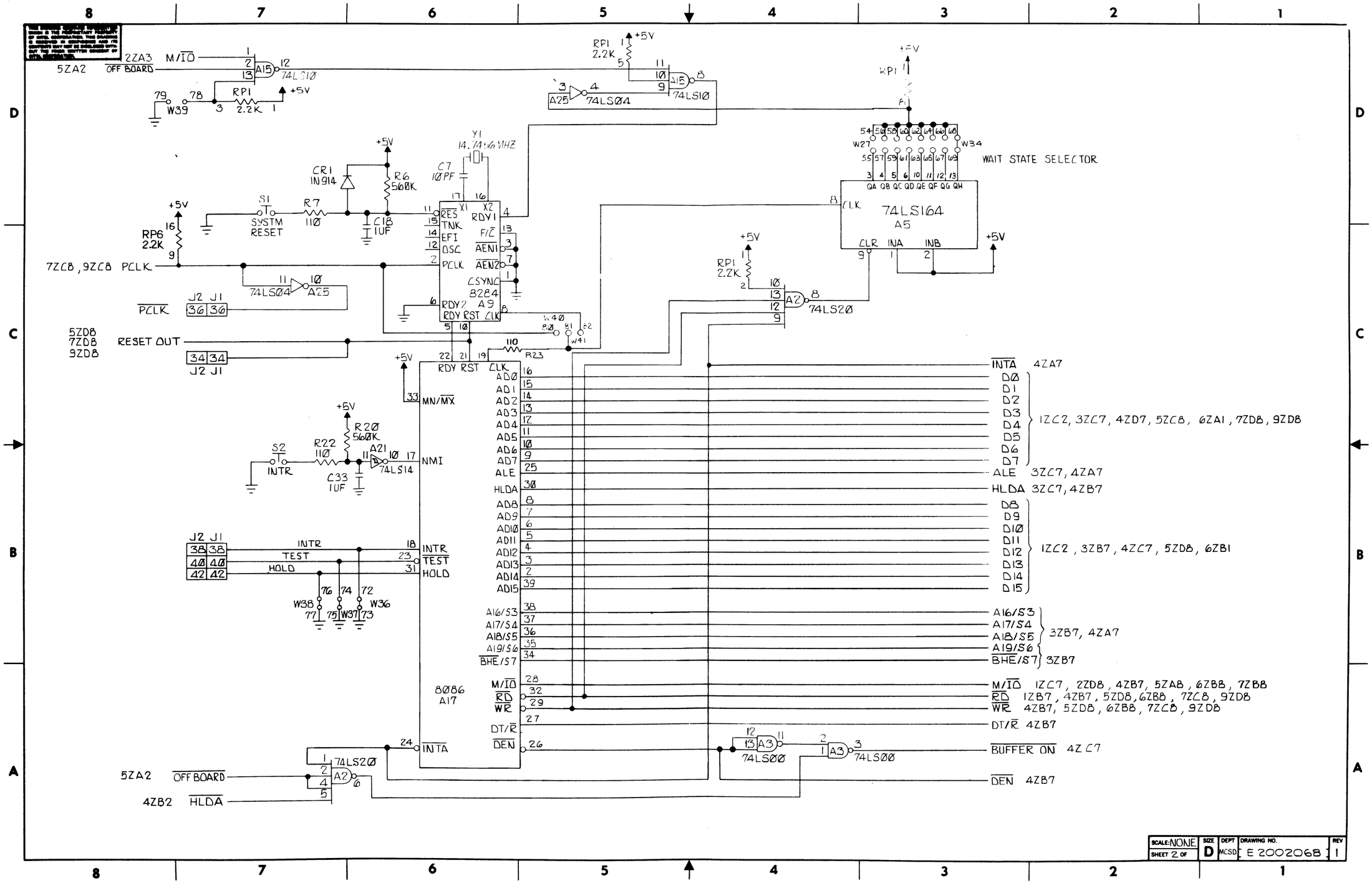


Figure 7-2. SDK-86 Schematic Diagram (Sheet 2 of 9)

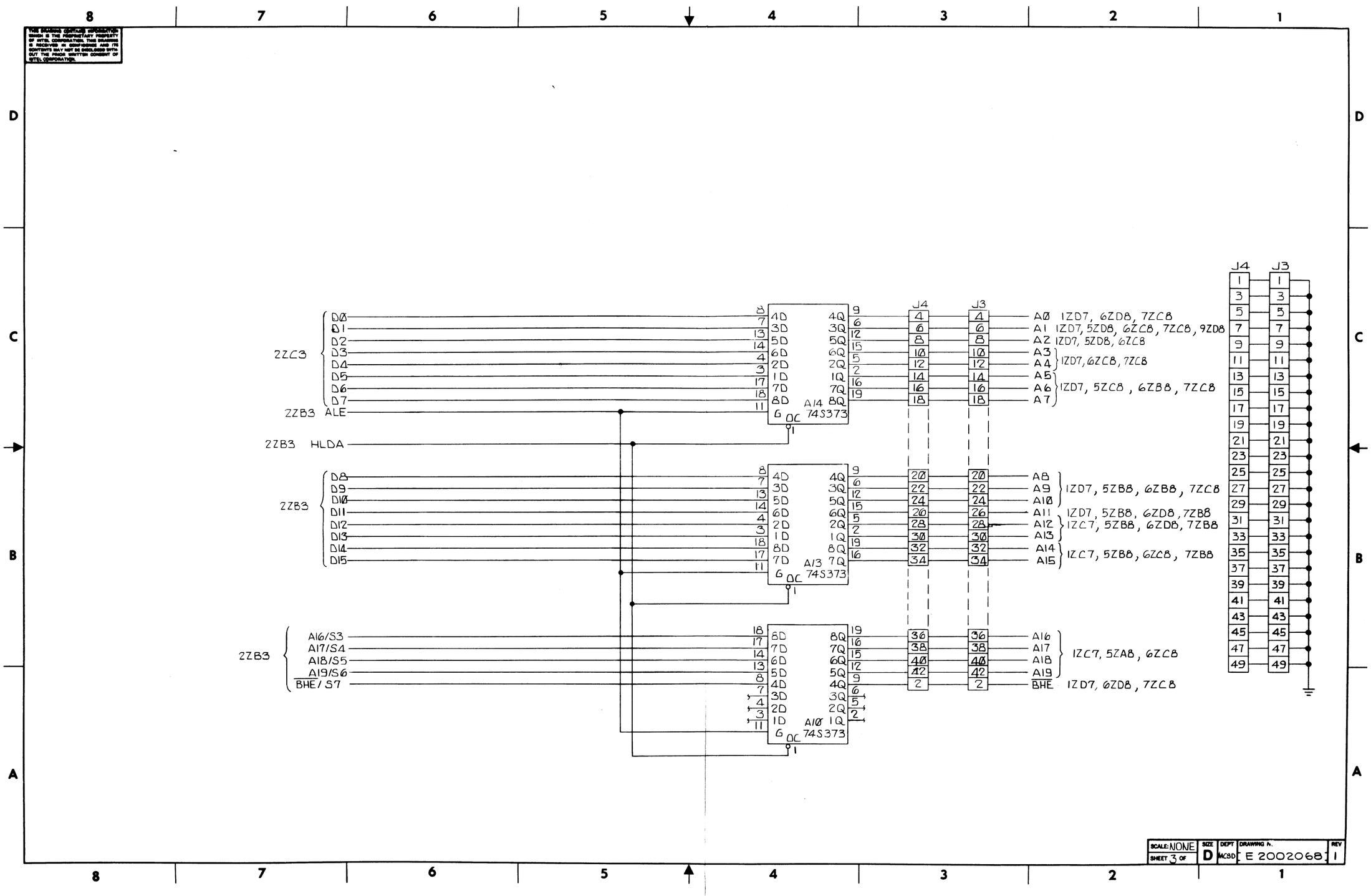


Figure 7-2. SDK-86 Schematic Diagram (Sheet 3 of 9)

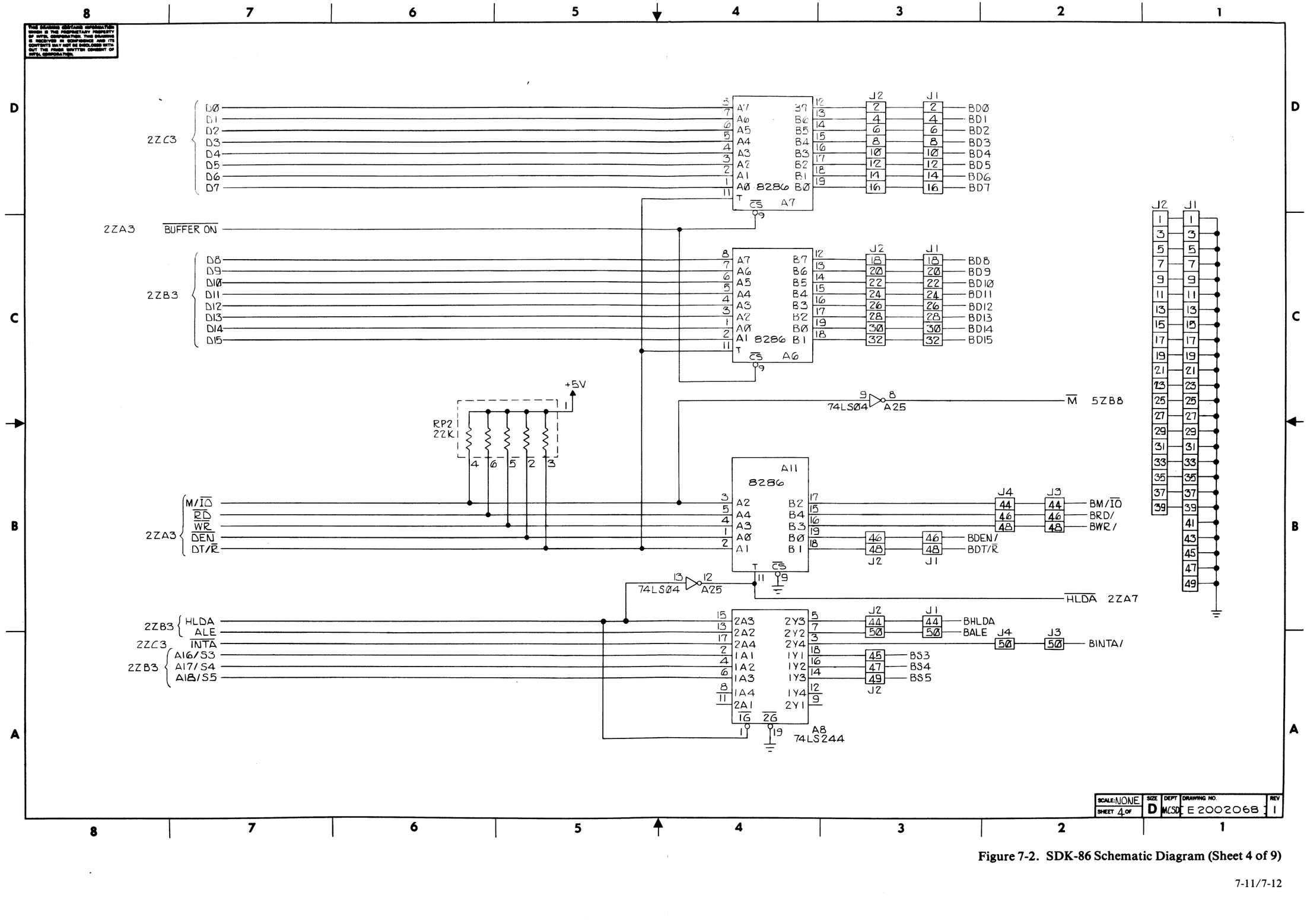


Figure 7-2. SDK-86 Schematic Diagram (Sheet 4 of 9)

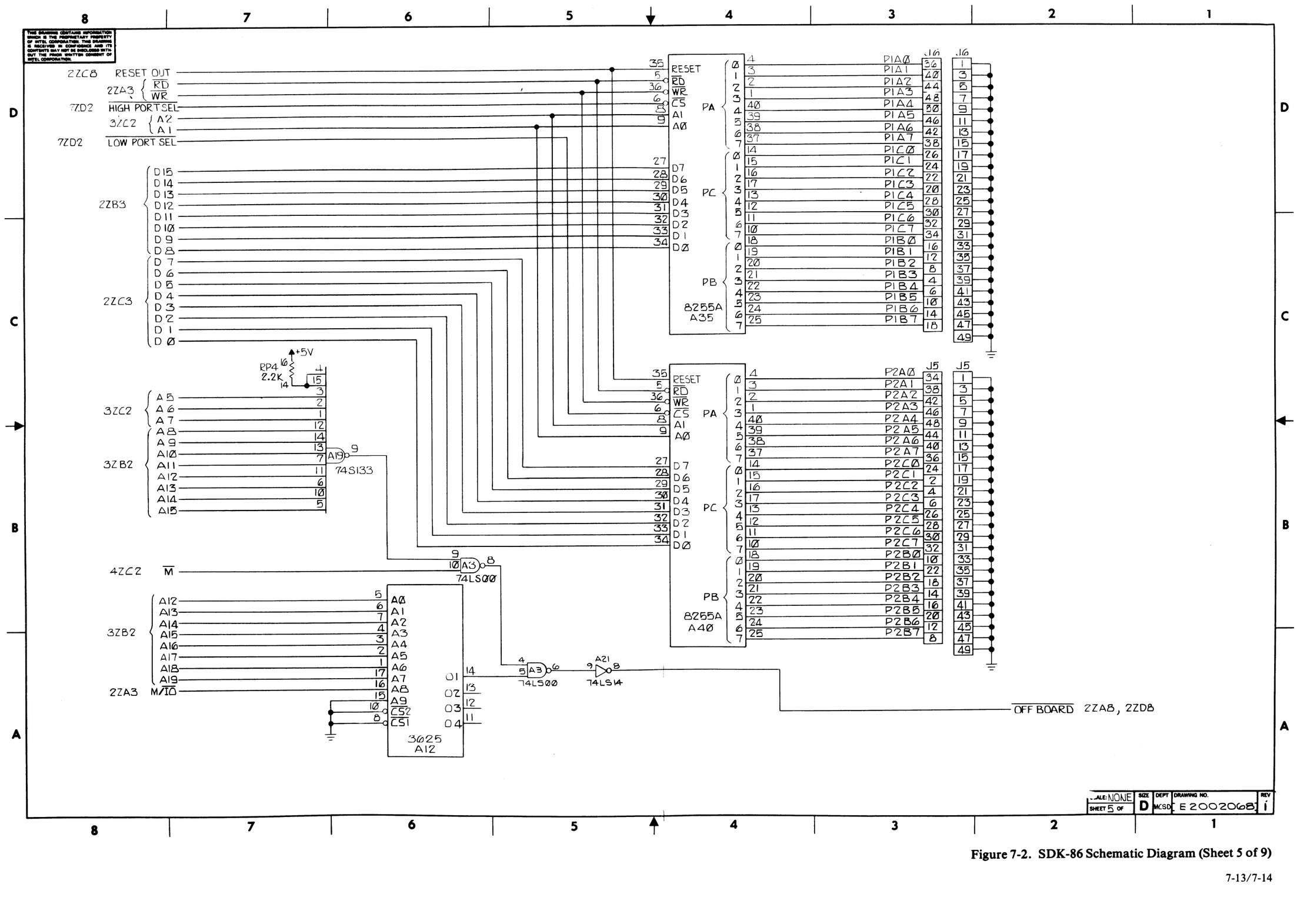


Figure 7-2. SDK-86 Schematic Diagram (Sheet 5 of 9)

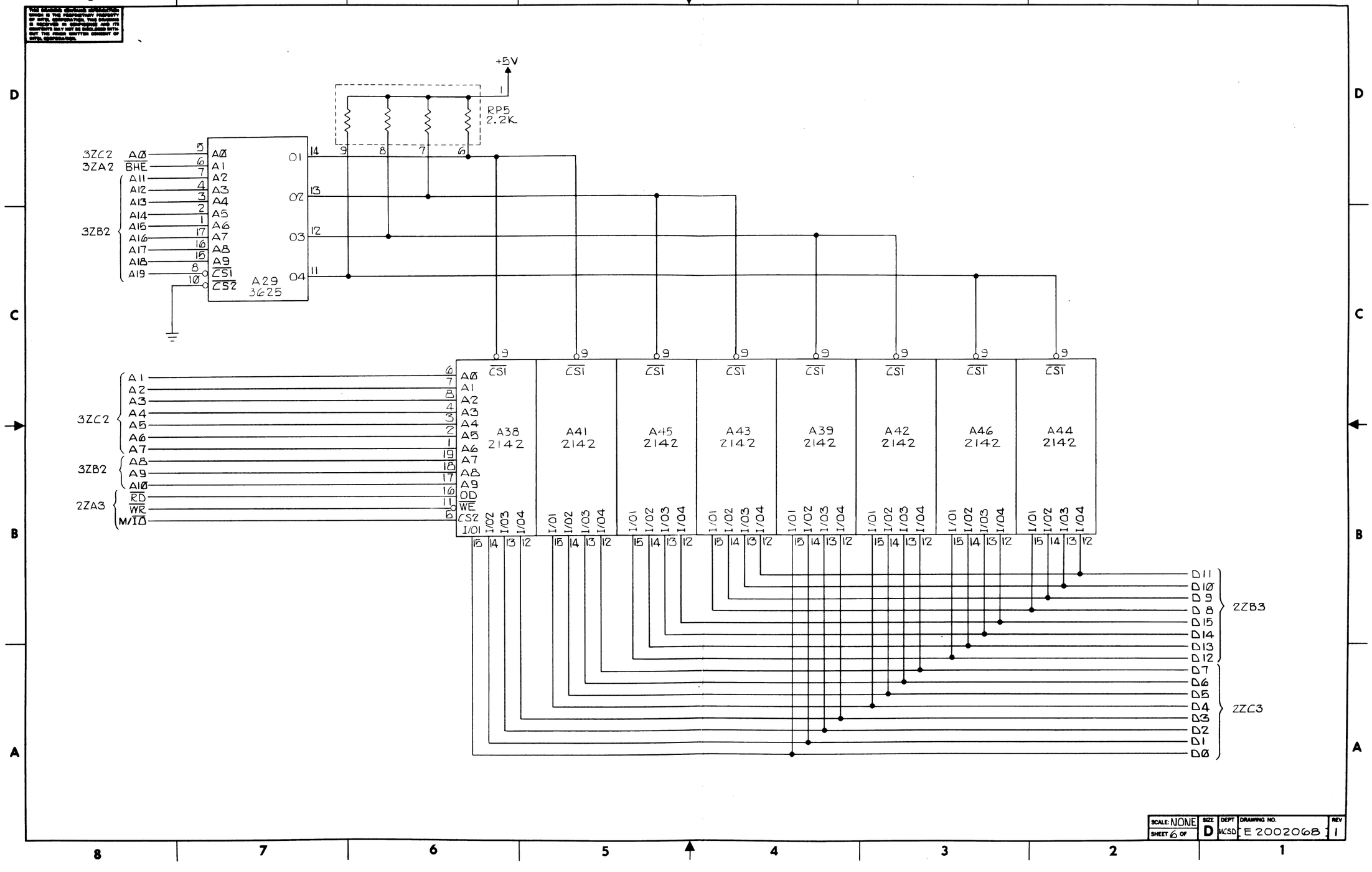


Figure 7-2. SDK-86 Schematic Diagram (Sheet 6 of 9)

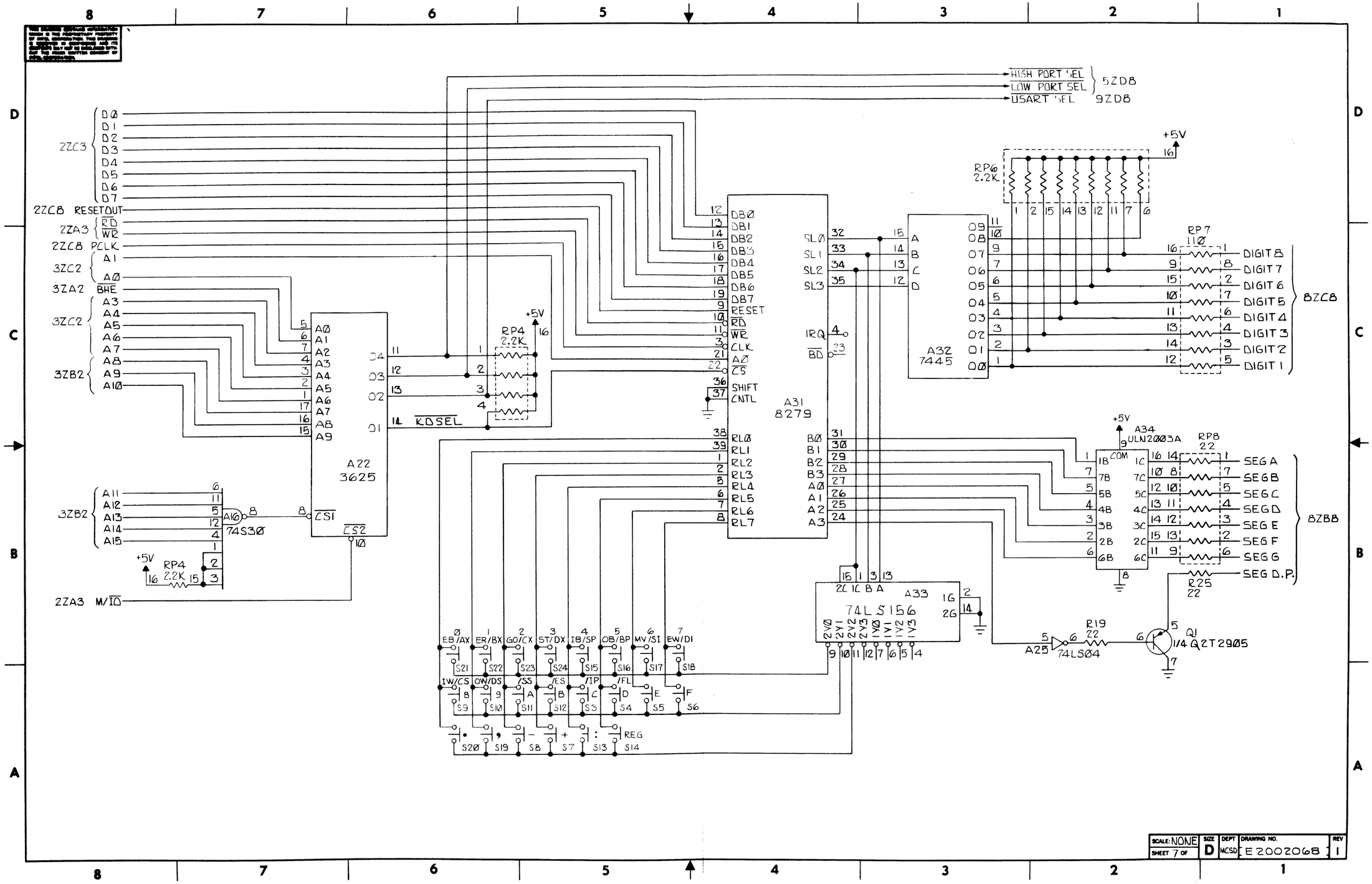


Figure 7-2. SDK-86 Schematic Diagram (Sheet 7 of 9)

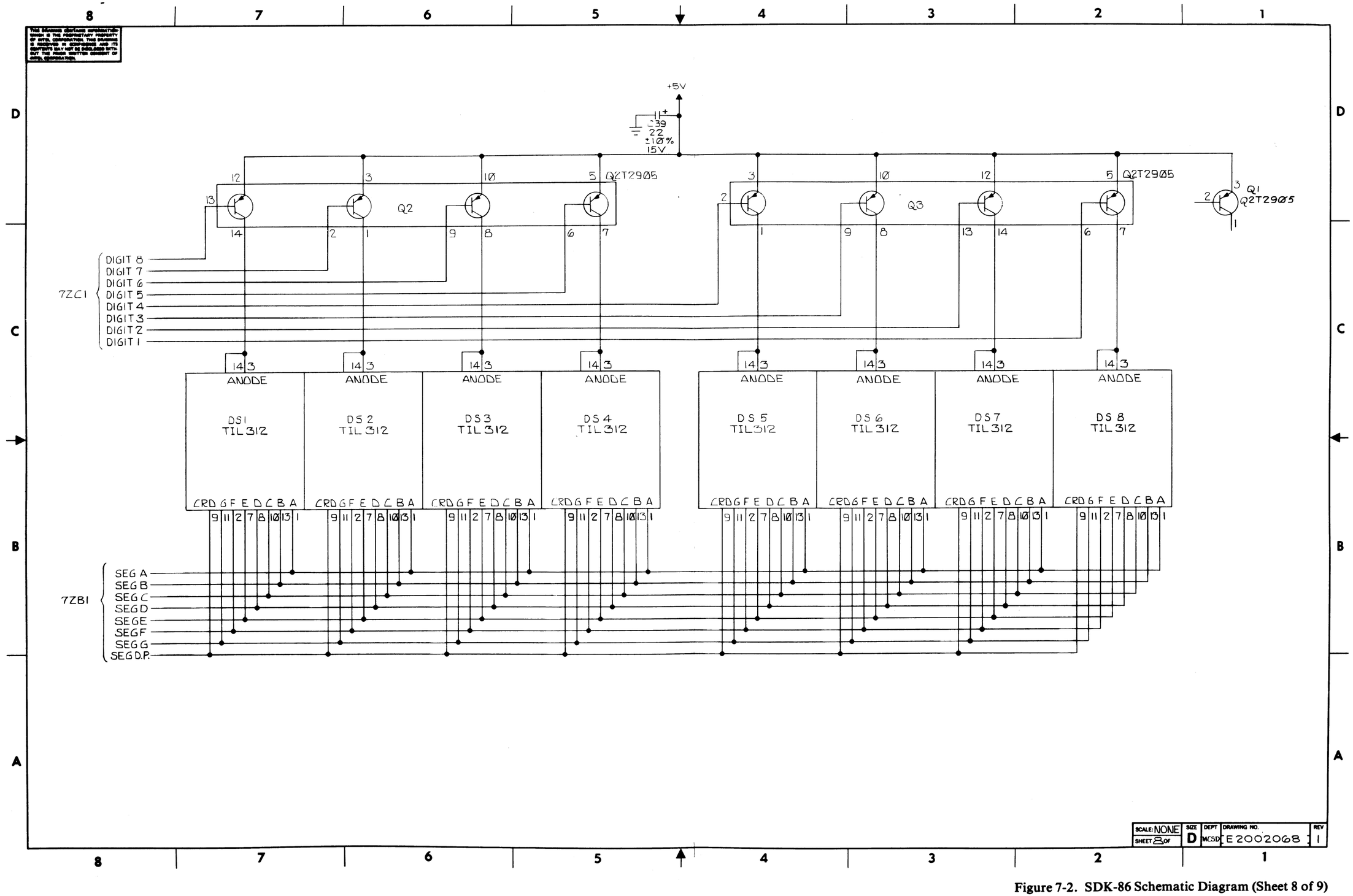
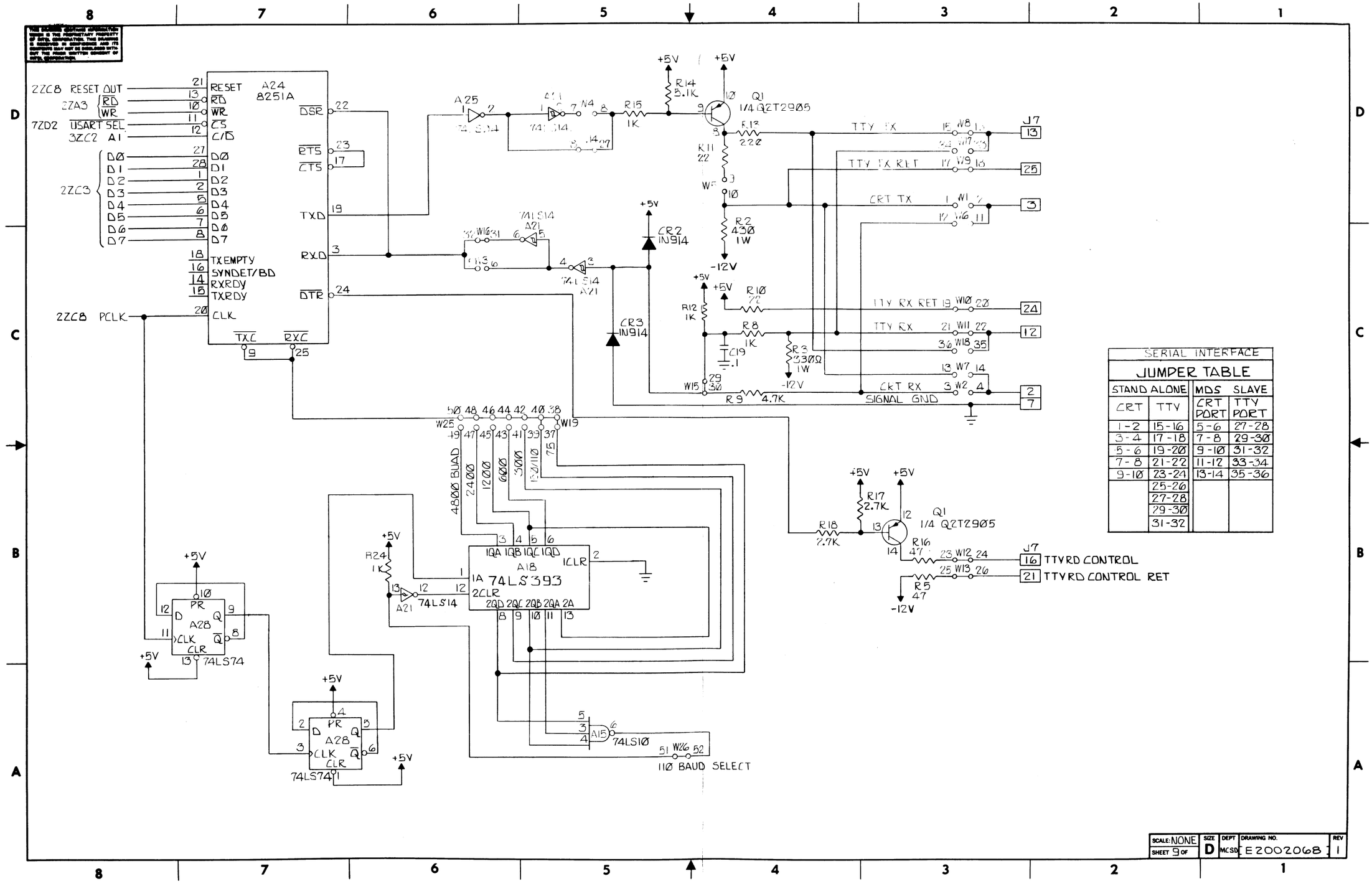


Figure 7-2. SDK-86 Schematic Diagram (Sheet 8 of 9)



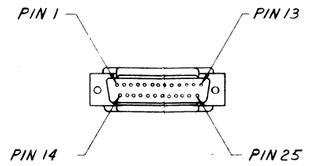
SERIAL INTERFACE

JUMPER TABLE

STAND ALONE		MDS SLAVE	
CRT	TTY	CRT PORT	TTY PORT
1-2	15-16	5-6	27-28
3-4	17-18	7-8	29-30
5-6	19-20	9-10	31-32
7-8	21-22	11-12	33-34
9-10	23-24	13-14	35-36
	25-26		
	27-28		
	29-30		
	31-32		

SCALE: NONE | SIZE D | DEPT MCSD | DRAWING NO. E2002068 | REV 1 | SHEET 9 OF

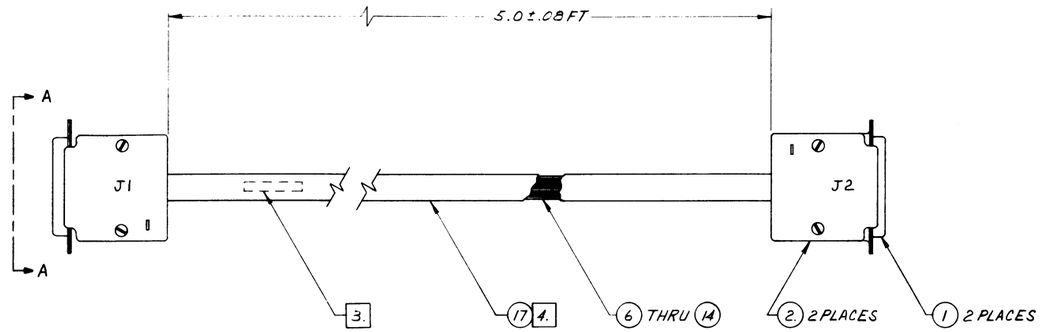
Figure 7-2. SDK-86 Schematic Diagram (Sheet 9 of 9)

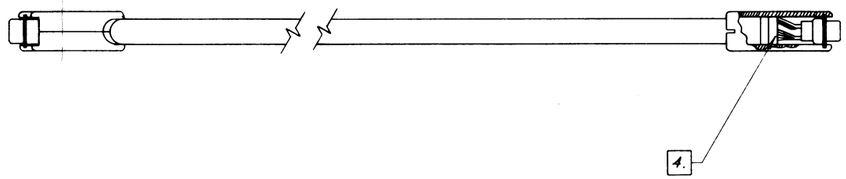


VIEW A-A
ROTATED 90° CCW
(SCALE 1:1)

J1	J2
2	2
3	3
7	7
12	12
13	13
16	16
21	21
24	24
25	25

SCHEMATIC (REF.)





SEE SEPARATE PARTS LIST

4. BRING ITEM 17 INSIDE HOOD AND BEYOND TIE WRAP APPROX .12 AS SHOWN.
3. MARK ASSY NO. AND REV. LETTER WITH CONTRASTING PERM COLOR APPROX SHOWN.
2. WORKMANSHIP PER MCSD, QAW 007.
1. ASSY PART NO. IS 4001927-01.

NOTES: UNLESS OTHERWISE SPECIFIED,

ITEM NO.	PART NUMBER	DESCRIPTION
	PARTS LIST	

QUANTITY PER DASH NO.

SCALE:

TOL. ± | 2 PLC .10 | 3 PLC — | ANGLE —

MATERIAL:

FINISH:

SIGNATURE	DATE
DRN BY	
CHK BY	
DFT APPD	
ENGR APPD	
AUTH BY	
CODE:	SHEET 1 OF

intel 3065 BOWERS AVE. SANTA CLARA CALIF. 95051

TITLE: CABLE ASSEMBLY SDK-86 PERIPHERAL

SIZE	DEPT	DRAWING NO.	REV
D	MCSD	E4001927	2

NEXT ASSY	USED ON
4002129	SDK-CBL

98-253A

Figure 7-3. SDK-C86 Cable Assembly

APPENDIX A
TELETYPEWRITER MODIFICATIONS

A-1. Introduction

This appendix provides information required to modify a Model ASR-33 Teletypewriter for use with the SDK-86.

A-2. Internal Modifications

WARNING

Hazardous voltages are exposed when the top cover of the teletypewriter is removed. To prevent accidental shock, disconnect the teleprinter power cord before proceeding beyond this point.

Remove the top cover and modify the teletypewriter as follows:

a. Remove blue lead from 750-ohm tap on current source register; reconnect this lead to 1450-ohm tap. (Refer to figures A-1 and A-2.)

b. On terminal block, change two wires as follows to create an internal full-duplex loop (refer to figures A-1 and A-3):

   1. Remove brown/yellow lead from terminal 3; reconnect this lead to terminal 5.
   2. Remove white/blue lead from terminal 4; reconnect this lead to terminal 5.

c. On terminal block, remove violet lead from terminal 8; reconnect this lead to terminal 9. This changes the receiver current level from 60 mA to 20 mA.

A relay circuit card must be fabricated and connected to the paper tape reader driver circuit. The relay circuit card to be fabricated requires a relay, a diode, a thyractor, a small 'vector' board for mounting the components, and suitable hardware for mounting the assembled relay card.

A circuit diagram of the relay circuit card is included in figure A-4; this diagram also includes the part numbers of the relay, diode, and thyractor. (Note that a 470-ohm resistor and a 0.1 μF capacitor may be substituted for the thyractor.) After the relay circuit card has been assembled, mount it in position as shown in figure A-5. Secure the card to the base plate using two self-tapping screws. Connect the relay circuit to the distributor trip magnet and mode switch as follows:

a. Refer to figure A-4 and connect a wire (Wire 'A') from relay circuit card to terminal L2 on mode switch. (See figure A-6.)

b. Disconnect brown wire shown in figure A-7 from plastic connector. Connect this brown wire to terminal l2 on mode switch. (Brown wire will have to be extended.)

c. Refer to figure A-4 and connect a wire (Wire 'B') from relay circuit board to terminal L1 on mode switch.

A-3. External Connections

Connect a two-wire receive loop, a two-wire send loop, and a two-wire tape reader control loop to the external device as shown in figure A-4. The external connector pin numbers shown in figure A-4 are for interface with an RS232C type, 25-pin connector.

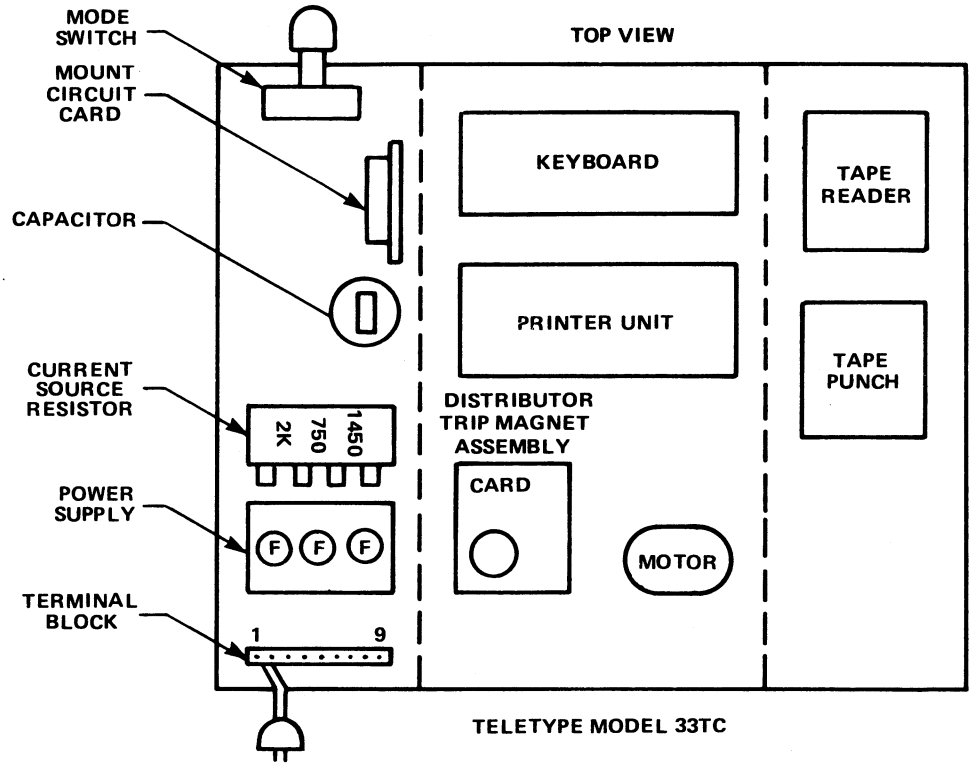


Figure A-1. Teletype Component Layout

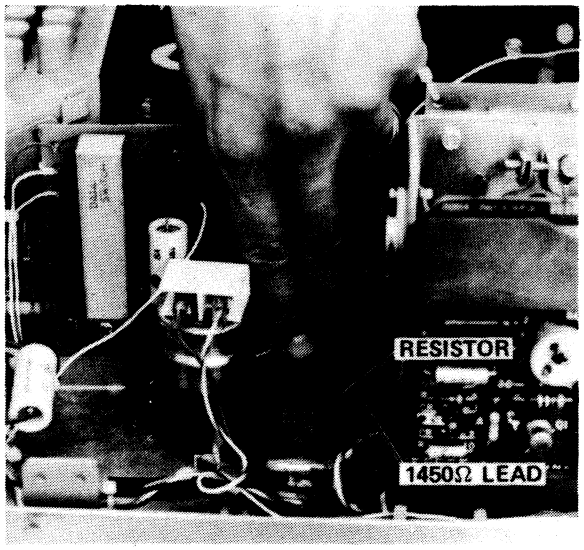


Figure A-2. Current Source Resistor

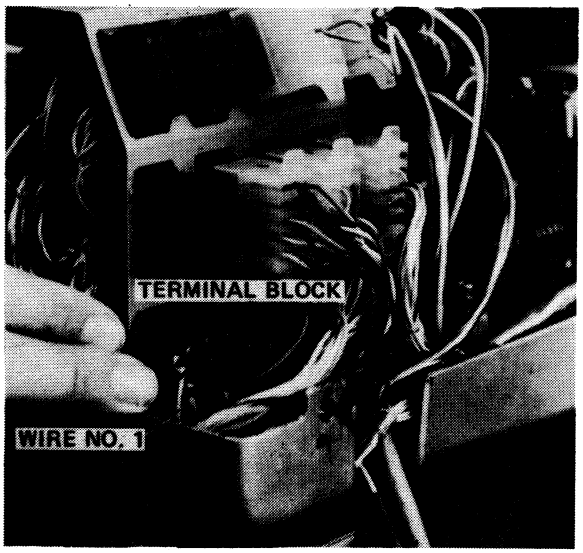


Figure A-3. Terminal Block

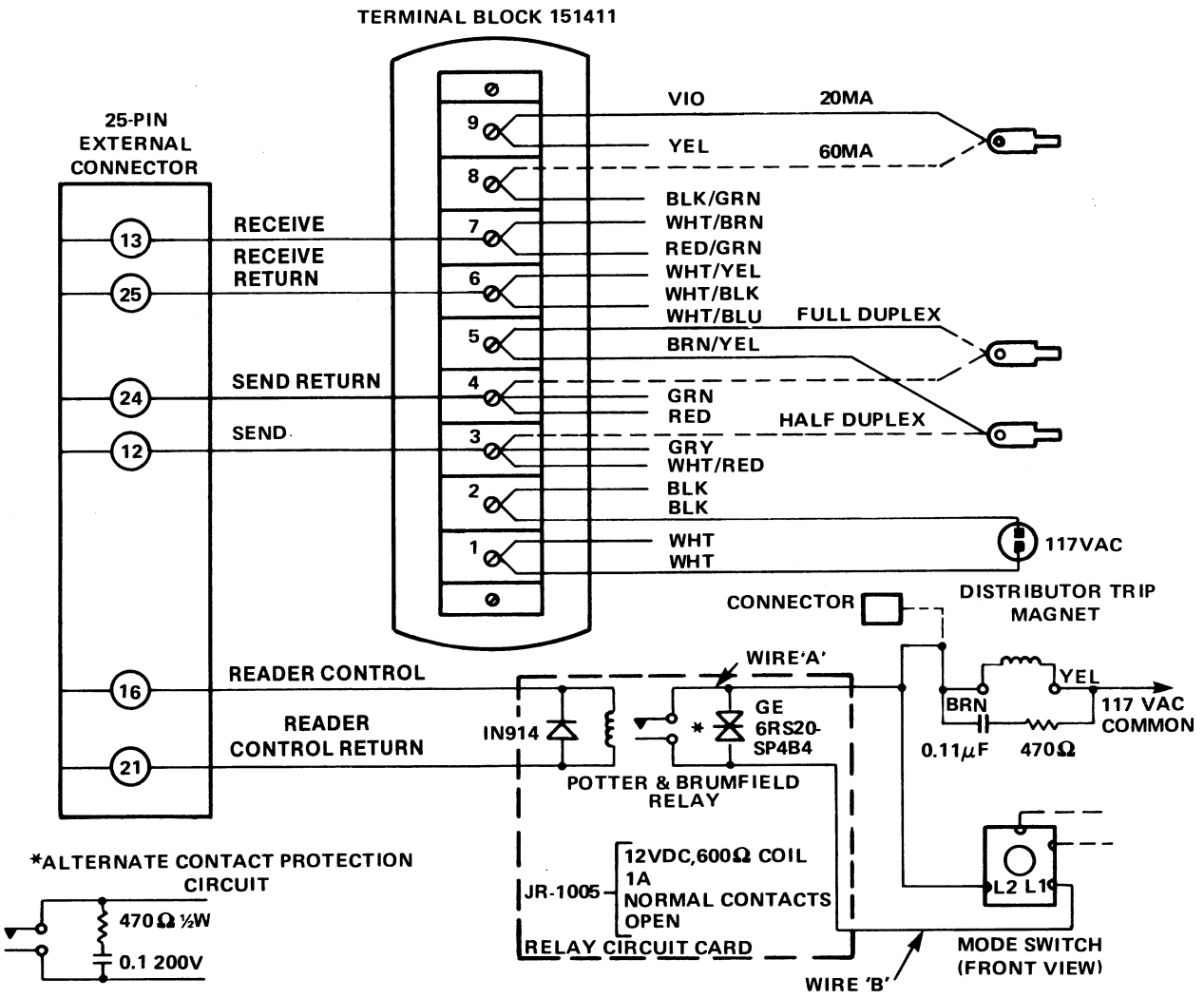


Figure A-4. Teletypewriter Modifications

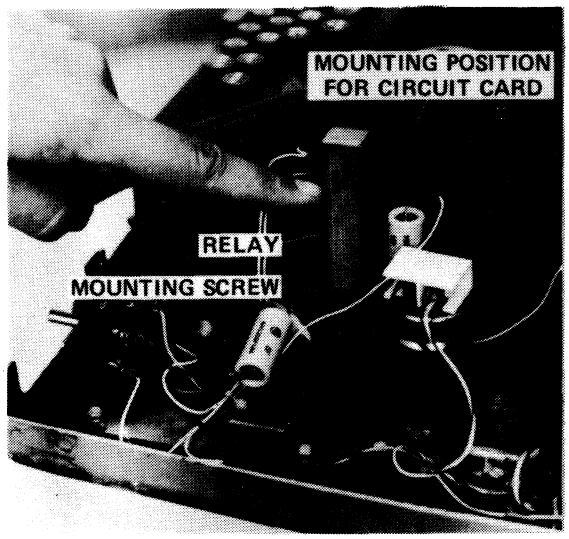


Figure A-5. Relay Circuit

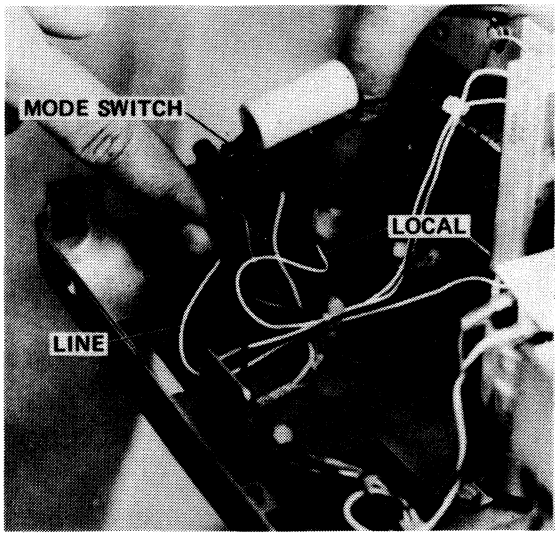


Figure A-6. Mode Switch

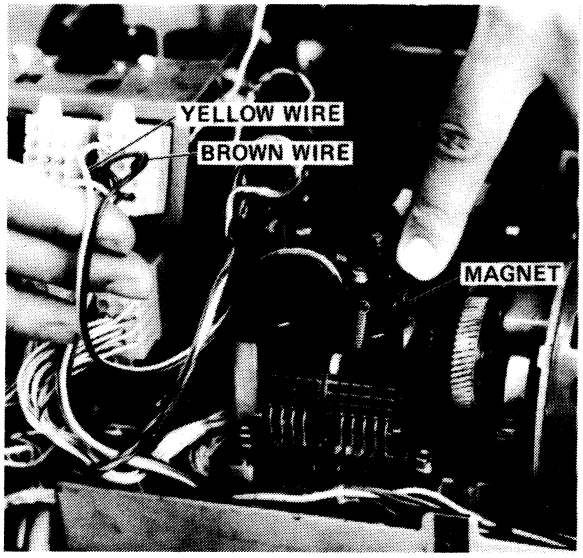


Figure A-7. Distributor Trip Magnet

APPENDIX B
INTEL LSI INTEGRATED CIRCUITS

This appendix provides supplemental information on the operation and function of Intel LSI integrated circuits incorporated into the design of the SDK-86. Included within this appendix are descriptions of the following circuits:

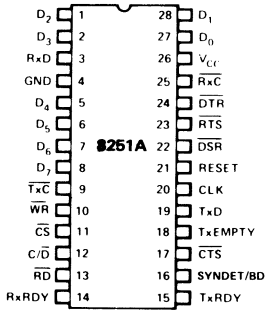
- 8251A Programmable Communication Interface (USART)
- 8255A Programmable Peripheral Interface (Parallel I/O Port)
- 8279 Programmable Keyboard/Display Interface

8251A
PROGRAMMABLE COMMUNICATION INTERFACE

- **Synchronous and Asynchronous Operation**
- **Synchronous 5-8 Bit Characters; Internal or External Character Synchronization; Automatic Sync Insertion**
- **Asynchronous 5-8 Bit Characters; Clock Rate—1, 16 or 64 Times Baud Rate; Break Character Generation; 1, 1½, or 2 Stop Bits; False Start Bit Detection; Automatic Break Detect and Handling; 19.2K Baud.**
- **Baud Rate — DC to 64K Baud**
- **Full Duplex, Double Buffered, Transmitter and Receiver**
- **Error Detection — Parity, Overrun and Framing**
- **Fully Compatible with 8080/8085 CPU**
- **28-Pin DIP Package**
- **All Inputs and Outputs are TTL Compatible**
- **Single +5V Supply**
- **Single TTL Clock**

The Intel® 8251A is the enhanced version of the industry standard, Intel® 8251 Universal Synchronous/Asynchronous Receiver/Transmitter (USART), designed for data communications with Intel's new high performance family of microprocessors such as the 8085. The 8251A is used as a peripheral device and is programmed by the CPU to operate using virtually any serial data transmission technique presently in use (including IBM "bi-sync"). The USART accepts data characters from the CPU in parallel format and then converts them into a continuous serial data stream for transmission. Simultaneously, it can receive serial data streams and convert them into parallel data characters for the CPU. The USART will signal the CPU whenever it can accept a new character for transmission or whenever it has received a character for the CPU. The CPU can read the complete status of the USART at any time. These include data transmission errors and control signals such as SYNDET, TxEMPTY. The chip is constructed using N-channel silicon gate technology.

PIN CONFIGURATION

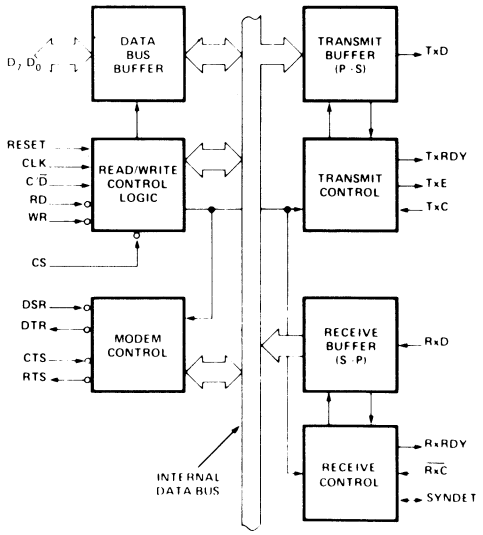


PIN NAMES

D_7 D_0	Data Bus (8 bits)
C/D	Control or Data is to be Written or Read
RD	Read Data Command
WR	Write Data or Control Command
CS	Chip Enable
CLK	Clock Pulse (TTL)
RESET	Reset
TxC	Transmitter Clock
TxD	Transmitter Data
RxC	Receiver Clock
RxD	Receiver Data
RxRDY	Receiver Ready (has character for 8080)
TxRDY	Transmitter Ready (ready for char from 8080)

DSR	Data Set Ready
DTR	Data Terminal Ready
SYNDET/BD	Sync Detect/ Break Detect
RTS	Request to Send Data
CTS	Clear to Send Data
TxE	Transmitter Empty
Vcc	+5 Volt Supply
GND	Ground

BLOCK DIAGRAM



February, 1978

FEATURES AND ENHANCEMENTS

8251A is an advanced design of the industry standard USART, the Intel® 8251. The 8251A operates with an extended range of Intel microprocessors that includes the new 8085 CPU and maintains compatibility with the 8251. Familiarization time is minimal because of compatibility and involves only knowing the additional features and enhancements, and reviewing the AC and DC specifications of the 8251A.

The 8251A incorporates all the key features of the 8251 and has the following additional features and enhancements:

- 8251A has double-buffered data paths with separate I/O registers for control, status, Data In, and Data Out, which considerably simplifies control programming and minimizes CPU overhead.
- In asynchronous operations, the Receiver detects and handles "break" automatically, relieving the CPU of this task.
- A refined Rx initialization prevents the Receiver from starting when in "break" state, preventing unwanted interrupts from a disconnected USART.
- At the conclusion of a transmission, TxD line will always return to the marking state unless SBRK is programmed.
- Tx Enable logic enhancement prevents a Tx Disable command from halting transmission until all data previously written has been transmitted. The logic also prevents the transmitter from turning off in the middle of a word.
- When External Sync Detect is programmed, Internal Sync Detect is disabled, and an External Sync Detect status is provided via a flip-flop which clears itself upon a status read.
- Possibility of false sync detect is minimized by ensuring that if double character sync is programmed, the characters be contiguously detected and also by clearing the Rx register to all ones whenever Enter Hunt command is issued in Sync mode.
- As long as the 8251A is not selected, the $\overline{RD}$ and $\overline{WR}$ do not affect the internal operation of the device.
- The 8251A Status can be read at any time but the status update will be inhibited during status read.
- The 8251A is free from extraneous glitches and has enhanced AC and DC characteristics, providing higher speed and better operating margins.
- Baud rate from DC to 64K.
- Fully compatible with Intel's new industry standard, the MCS-85.

8251A BASIC FUNCTIONAL DESCRIPTION

General

The 8251A is a Universal Synchronous/Asynchronous Receiver/Transmitter designed specifically for the 80/85 Microcomputer Systems. Like other I/O devices in a Microcomputer System, its functional configuration is programmed by the system's software for maximum flexibility. The 8251A can support virtually any serial data technique currently in use (including IBM "bi-sync").

In a communication environment an interface device must convert parallel format system data into serial format for transmission and convert incoming serial format data into parallel system data for reception. The interface device must also delete or insert bits or characters that are functionally unique to the communication technique. In essence, the interface should appear "transparent" to the CPU, a simple input or output of byte-oriented system data.

Data Bus Buffer

This 3-state, bidirectional, 8-bit buffer is used to interface the 8251A to the system Data Bus. Data is transmitted or received by the buffer upon execution of INput or OUTput instructions of the CPU. Control words, Command words and Status information are also transferred through the Data Bus Buffer. The command status and data in, and data out are separate 8-bit registers to provide double buffering.

This functional block accepts inputs from the system Control bus and generates control signals for overall device operation. It contains the Control Word Register and Command Word Register that store the various control formats for the device functional definition.

RESET (Reset)

A "high" on this input forces the 8251A into an "Idle" mode. The device will remain at "Idle" until a new set of control words is written into the 8251A to program its functional definition. Minimum RESET pulse width is 6 t_{CY} (clock must be running).

CLK (Clock)

The CLK input is used to generate internal device timing and is normally connected to the Phase 2 (TTL) output of the 8224 Clock Generator. No external inputs or outputs are referenced to CLK but the frequency of CLK must be greater than 30 times the Receiver or Transmitter data bit rates.

$\overline{WR}$ (Write)

A "low" on this input informs the 8251A that the CPU is writing data or control words to the 8251A.

$\overline{RD}$ (Read)

A "low" on this input informs the 8251A that the CPU is reading data or status information from the 8251A.

C/$\overline{D}$ (Control/Data)

This input, in conjunction with the $\overline{WR}$ and $\overline{RD}$ inputs, informs the 8251A that the word on the Data Bus is either a data character, control word or status information.
1 = CONTROL/STATUS 0 = DATA

$\overline{CS}$ (Chip Select)

A "low" on this input selects the 8251A. No reading or writing will occur unless the device is selected. When $\overline{CS}$ is high, the Data Bus in the float state and $\overline{RD}$ and $\overline{WR}$ will have no effect on the chip.

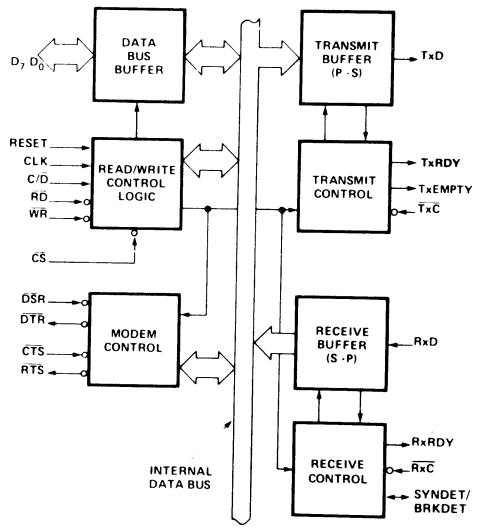


Figure 1. 8251A Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions

C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	
0	0	1	0	8251A DATA ⇒ DATA BUS
0	1	0	0	DATA BUS ⇒ 8251A DATA
1	0	1	0	STATUS ⇒ DATA BUS
1	1	0	0	DATA BUS ⇒ CONTROL
X	1	1	0	DATA BUS ⇒ 3-STATE
X	X	X	1	DATA BUS ⇒ 3-STATE

Modem Control

The 8251A has a set of control inputs and outputs that can be used to simplify the interface to almost any Modem. The Modem control signals are general purpose in nature and can be used for functions other than Modem control, if necessary.

$\overline{\text{DSR}}$ (Data Set Ready)

The $\overline{\text{DSR}}$ input signal is a general purpose, 1-bit inverting input port. Its condition can be tested by the CPU using a Status Read operation. The $\overline{\text{DSR}}$ input is normally used to test Modem conditions such as Data Set Ready.

$\overline{\text{DTR}}$ (Data Terminal Ready)

The $\overline{\text{DTR}}$ output signal is a general purpose, 1-bit inverting output port. It can be set "low" by programming the appropriate bit in the Command Instruction word. The $\overline{\text{DTR}}$ output signal is normally used for Modem control such as Data Terminal Ready or Rate Select.

$\overline{\text{RTS}}$ (Request to Send)

The $\overline{\text{RTS}}$ output signal is a general purpose, 1-bit inverting output port. It can be set "low" by programming the appropriate bit in the Command Instruction word. The $\overline{\text{RTS}}$ output signal is normally used for Modem control such as Request to Send.

$\overline{\text{CTS}}$ (Clear to Send)

A "low" on this input enables the 8251A to transmit serial data if the Tx Enable bit in the Command byte is set to a "one." If either a Tx Enable off or CTS off condition occurs while the Tx is in operation, the Tx will transmit all the data in the USART, written prior to Tx Disable command before shutting down.

Transmitter Buffer

The Transmitter Buffer accepts parallel data from the Data Bus Buffer, converts it to a serial bit stream, inserts the appropriate characters or bits (based on the communication technique) and outputs a composite serial stream of data on the TxD output pin on the falling edge of $\overline{\text{TxC}}$. The transmitter will begin transmission upon being enabled if $\overline{\text{CTS}}$ = 0. The TxD line will be held in the marking state immediately upon a master Reset or when Tx Enable/$\overline{\text{CTS}}$ off or TxEMPTY.

Transmitter Control

The transmitter Control manages all activities associated with the transmission of serial data. It accepts and issues signals both externally and internally to accomplish this function.

TxRDY (Transmitter Ready)

This output signals the CPU that the transmitter is ready to accept a data character. The TxRDY output pin can be used as an interrupt to the system, since it is masked by Tx Disabled, or, for Polled operation, the CPU can check TxRDY using a Status Read operation. TxRDY is automatically reset by the leading edge of $\overline{\text{WR}}$ when a data character is loaded from the CPU.

Note that when using the Polled operation, the TxRDY status bit is *not* masked by Tx Enabled, but will only indicate the Empty/Full Status of the Tx Data Input Register.

TxE (Transmitter Empty)

When the 8251A has no characters to transmit, the TxEMPTY output will go "high". It resets automatically upon receiving a character from the CPU. TxEMPTY can be used to indicate the end of a transmission mode, so that the CPU "knows" when to "turn the line around" in the half-duplexed operational mode. TxEMPTY is independent of the Tx Enable bit in the Command instruction.

In SYNChronous mode, a "high" on this output indicates that a character has not been loaded and the SYNC character or characters are about to be or are being transmitted automatically as "fillers". TxEMPTY does not go low when the SYNC characters are being shifted out.

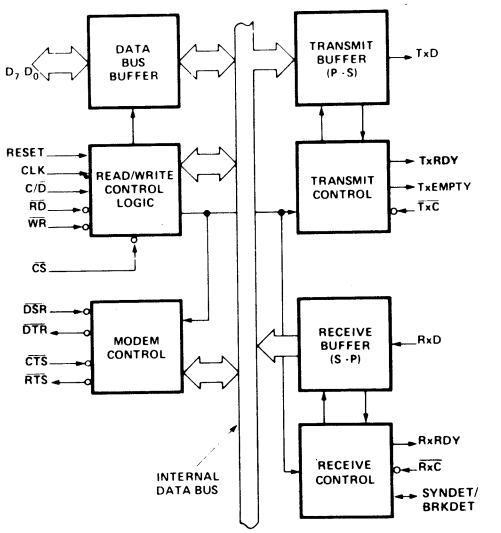


Figure 2. 8251A Block Diagram Showing Modem and Transmitter Buffer and Control Functions

$\overline{\text{TxC}}$ (Transmitter Clock)

The Transmitter Clock controls the rate at which the character is to be transmitted. In the Synchronous transmission mode, the Baud Rate (1x) is equal to the $\overline{\text{TxC}}$ frequency. In Asynchronous transmission mode the baud rate is a fraction of the actual $\overline{\text{TxC}}$ frequency. A portion of the mode instruction selects this factor; it can be 1, 1/16 or 1/64 the $\overline{\text{TxC}}$.

For Example:

If Baud Rate equals 110 Baud,
$\overline{\text{TxC}}$ equals 110 Hz (1x)
$\overline{\text{TxC}}$ equals 1.76 kHz (16x)
$\overline{\text{TxC}}$ equals 7.04 kHz (64x).

The falling edge of $\overline{\text{TxC}}$ shifts the serial data out of the 8251A.

Receiver Buffer

The Receiver accepts serial data, converts this serial input to parallel format, checks for bits or characters that are unique to the communication technique and sends an "assembled" character to the CPU. Serial data is input to RxD pin, and is clocked in on the rising edge of $\overline{RxC}$.

Receiver Control

This functional block manages all receiver-related activities which consist of the following features:

The RxD initialization circuit prevents the 8251A from mistaking an unused input line for an active low data line in the "break condition". Before starting to receive serial characters on the RxD line, a valid "1" must first be detected after a chip master Reset. Once this has been determined, a search for a valid low (Start bit) is enabled. This feature is only active in the asynchronous mode, and is only done once for each master Reset.

The False Start bit detection circuit prevents false starts due to a transient noise spike by first detecting the falling edge and then strobing the nominal center of the Start bit (RxD = low).

The Parity Toggle F/F and Parity Error F/F circuits are used for parity error detection and set the corresponding status bit.

The Framing Error Flag F/F is set if the Stop bit is absent at the end of the data byte (asynchronous mode), and also sets the corresponding status bit.

RxRDY (Receiver Ready)

This output indicates that the 8251A contains a character that is ready to be input to the CPU. Rx RDY can be connected to the interrupt structure of the CPU or, for Polled operation, the CPU can check the condition of RxRDY using a Status Read operation.

Rx Enable off both masks and holds RxRDY in the Reset Condition. For Asynchronous mode, to set RxRDY, the Receiver must be Enabled to sense a Start Bit and a complete character must be assembled and transferred to the Data Output Register. For Synchronous mode, to set RxRDY, the Receiver must be enabled and a character must finish assembly and be transferred to the Data Output Register.

Failure to read the received character from the Rx Data Output Register prior to the assembly of the next Rx Data character will set overrun condition error and the previous character will be written over and lost. If the Rx Data is being read by the CPU when the internal transfer is occurring, overrun error will be set and the old character will be lost.

$\overline{RxC}$ (Receiver Clock)

The Receiver Clock controls the rate at which the character is to be received. In Synchronous Mode, the Baud Rate (1x) is equal to the actual frequency of $\overline{RxC}$. In Asynchronous Mode, the Baud Rate is a fraction of the actual $\overline{RxC}$ frequency. A portion of the mode instruction selects this factor; 1, 1/16 or 1/64 the $\overline{RxC}$.

For Example:

Baud Rate equals 300 Baud, if
$\overline{RxC}$ equals 300 Hz (1x)
$\overline{RxC}$ equals 4800 Hz (16x)
$\overline{RxC}$ equals 19.2 kHz (64x).

Baud Rate equals 2400 Baud, if
$\overline{RxC}$ equals 2400 Hz (1x)
$\overline{RxC}$ equals 38.4 kHz (16x)
$\overline{RxC}$ equals 153.6 kHz (64x).

Data is sampled into the 8251A on the rising edge of $\overline{RxC}$.

NOTE: In most communications systems, the 8251A will be handling both the transmission and reception operations of a single link. Consequently, the Receive and Transmit Baud Rates will be the same. Both $\overline{TxC}$ and $\overline{RxC}$ will require identical frequencies for this operation and can be tied together and connected to a single frequency source (Baud Rate Generator) to simplify the interface.

SYNDET (SYNC Detect)/BRKDET (Break Detect)

This pin is used in SYNChronous Mode for SYNDET and may be used as either input or output, programmable through the Control Word. It is reset to output mode low upon RESET. When used as an output (internal Sync mode), the SYNDET pin will go "high" to indicate that the 8251A has located the SYNC character in the Receive mode. If the 8251A is programmed to use double Sync characters (bisync), then SYNDET will go "high" in the middle of the last bit of the second Sync character. SYNDET is automatically reset upon a Status Read operation.

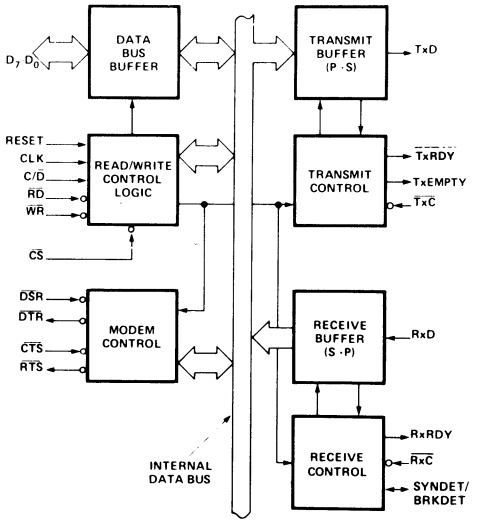


Figure 3. 8251A Block Diagram Showing Receiver Buffer and Control Functions

When used as an input (external SYNC detect mode), a positive going signal will cause the 8251A to start assembling data characters on the rising edge of the next $\overline{RxC}$. Once in SYNC, the "high" input signal can be removed. the period of $\overline{RxC}$. When External SYNC Detect is programmed, the Internal SYNC Detect is disabled.

BREAK DETECT (Async Mode Only)

This output will go high whenever an all zero word of the programmed length (including start bit, data bit, parity bit, and *one* stop bit) is received. Break Detect may also be read as a Status bit. It is reset only upon a master chip Reset or Rx Data returning to a "one" state.

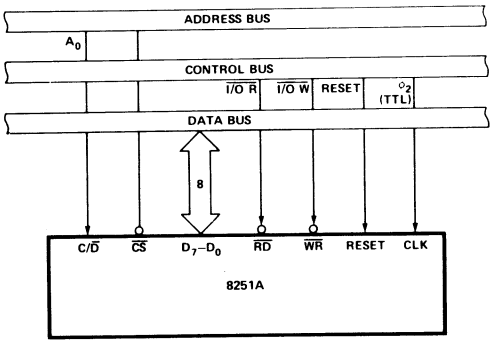


Figure 4. 8251A Interface to 8080 Standard System Bus

DETAILED OPERATION DESCRIPTION

General

The complete functional definition of the 8251A is programmed by the system's software. A set of control words must be sent out by the CPU to initialize the 8251A to support the desired communications format. These control words will program the: BAUD RATE, CHARACTER LENGTH, NUMBER OF STOP BITS, SYNCHRONOUS or ASYNCHRONOUS OPERATION, EVEN/ODD/OFF PARITY, etc. In the Synchronous Mode, options are also provided to select either internal or external character synchronization.

Once programmed, the 8251A is ready to perform its communication functions. The TxRDY output is raised "high" to signal the CPU that the 8251A is ready to receive a data character from the CPU. This output (TxRDY) is reset automatically when the CPU writes a character into the 8251A. On the other hand, the 8251A receives serial data from the MODEM or I/O device. Upon receiving an entire character, the RxRDY output is raised "high" to signal the CPU that the 8251A has a complete character ready for the CPU to fetch. RxRDY is reset automatically upon the CPU data read operation.

The 8251A cannot begin transmission until the Tx Enable (Transmitter Enable) bit is set in the Command Instruction and it has received a Clear To Send ($\overline{CTS}$) input. The TxD output will be held in the marking state upon Reset.

Programming the 8251A

Prior to starting data transmission or reception, the 8251A must be loaded with a set of control words generated by the CPU. These control signals define the complete functional definition of the 8251A and must immediately follow a Reset operation (internal or external).

The control words are split into two formats:

1. Mode Instruction
2. Command Instruction

Mode Instruction

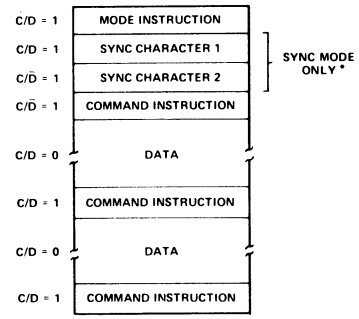
This format defines the general operational characteristics of the 8251A. It must follow a Reset operation (internal or external). Once the Mode Instruction has been written into the 8251A by the CPU, SYNC characters or Command Instructions may be inserted.

Command Instruction

This format defines a status word that is used to control the actual operation of the 8251A.

Both the Mode and Command Instructions must conform to a specified sequence for proper device operation. The Mode Instruction must be inserted immediately following a Reset operation, prior to using the 8251A for data communication.

All control words written into the 8251A after the Mode Instruction will load the Command Instruction. Command Instructions can be written into the 8251A at any time in the data block during the operation of the 8251A. To return to the Mode Instruction format, the master Reset bit in the Command Instruction word can be set to initiate an internal Reset operation which automatically places the 8251A back into the Mode Instruction format. Command Instructions must follow the Mode Instructions or Sync characters.



* The second SYNC character is skipped if MODE instruction has programmed the 8251A to single character Internal SYNC Mode. Both SYNC characters are skipped if MODE instruction has programmed the 8251A to ASYNC mode.

Figure 5. Typical Data Block

Mode Instruction Definition

The 8251A can be used for either Asynchronous or Synchronous data communication. To understand how the Mode Instruction defines the functional operation of the 8251A, the designer can best view the device as two separate components sharing the same package, one Asynchronous the other Synchronous. The format definition can be changed only after a master chip Reset. For explanation purposes the two formats will be isolated.

NOTE: When parity is enabled it is not considered as one of the data bits for the purpose of programming the word length. The actual parity bit received on the Rx Data line cannot be read on the Data Bus. In the case of a programmed character length of less than 8 bits, the least significant Data Bus bits will hold the data; unused bits are "don't care" when writing data to the 8251A, and will be "zeros" when reading the data from the 8251A.

Asynchronous Mode (Transmission)

Whenever a data character is sent by the CPU the 8251A automatically adds a Start bit (low level) followed by the data bits (least significant bit first), and the programmed number of Stop bits to each character. Also, an even or odd Parity bit is inserted prior to the Stop bit(s), as defined by the Mode Instruction. The character is then transmitted as a serial data stream on the TxD output. The serial data is shifted out on the falling edge of $\overline{TxC}$ at a rate equal to 1, 1/16, or 1/64 that of the $\overline{TxC}$, as defined by the Mode Instruction. BREAK characters can be continuously sent to the TxD if commanded to do so.

When no data characters have been loaded into the 8251A the TxD output remains "high" (marking) unless a Break (continuously low) has been programmed.

Asynchronous Mode (Receive)

The RxD line is normally high. A falling edge on this line triggers the beginning of a START bit. The validity of this START bit is checked by again strobing this bit at its nominal center (16X or 64X mode only). If a low is detected again, it is a valid START bit, and the bit counter will start counting. The bit counter thus locates the center of the data bits, the parity bit (if it exists) and the stop bits. If parity error occurs, the parity error flag is set. Data and parity bits are sampled on the RxD pin with the rising edge of $\overline{RxC}$. If a low level is detected as the STOP bit, the Framing Error flag will be set. The STOP bit signals the end of a character. Note that the *receiver* requires only *one* stop bit, regardless of the number of stop bits programmed. This character is then loaded into the parallel I/O buffer of the 8251A. The RxRDY pin is raised to signal the CPU that a character is ready to be fetched. If a previous character has not been fetched by the CPU, the present character replaces it in the I/O buffer, and the OVERRUN Error flag is raised (thus the previous character is lost). All of the error flags can be reset by an Error Reset Instruction. The occurrence of any of these errors will not affect the operation of the 8251A.

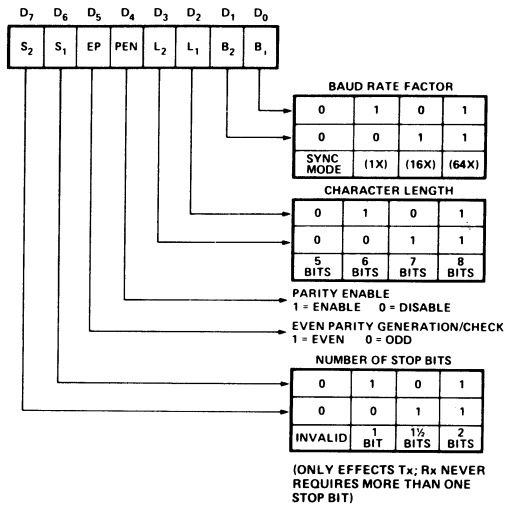


Figure 6. Mode Instruction Format, Asynchronous Mode

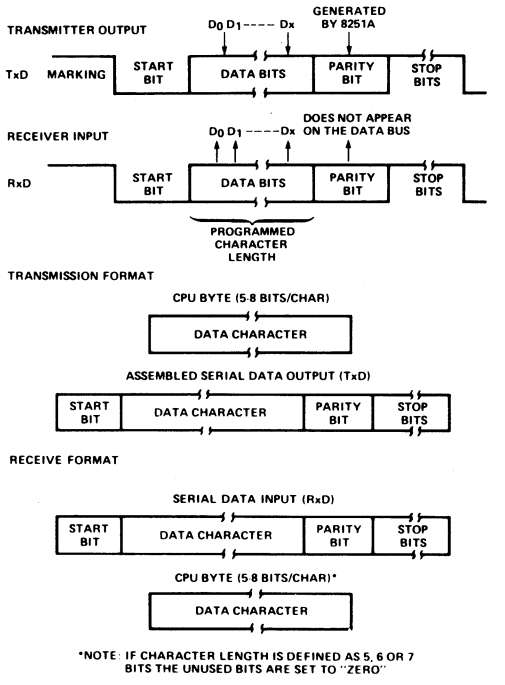
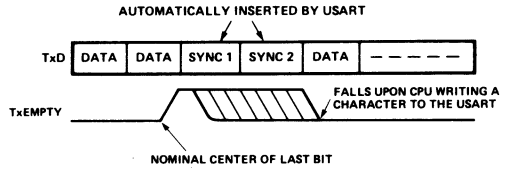


Figure 7. Asynchronous Mode

Synchronous Mode (Transmission)

The TxD output is continuously high until the CPU sends its first character to the 8251A which usually is a SYNC character. When the $\overline{CTS}$ line goes low, the first character is serially transmitted out. All characters are shifted out on the falling edge of $\overline{TxC}$. Data is shifted out at the same rate as the $\overline{TxC}$.

Once transmission has started, the data stream at the TxD output must continue at the $\overline{TxC}$ rate. If the CPU does not provide the 8251A with a data character before the 8251A Transmitter Buffers become empty, the SYNC characters (or character if in single SYNC character mode) will be automatically inserted in the TxD data stream. In this case, the TxEMPTY pin is raised high to signal that the 8251A is empty and SYNC characters are being sent out. TxEMPTY does not go low when the SYNC is being shifted out (see figure below). The TxEMPTY pin is internally reset by a data character being written into the 8251A.



Synchronous Mode (Receive)

In this mode, character synchronization can be internally or externally achieved. If the SYNC mode has been programmed, ENTER HUNT command should be included in the first command instruction word written. Data on the RxD pin is then sampled in on the rising edge of $\overline{RxC}$. The content of the Rx buffer is compared at every bit boundary with the first SYNC character until a match occurs. If the 8251A has been programmed for two SYNC characters, the subsequent received character is also compared; when both SYNC characters have been detected, the USART ends the HUNT mode and is in character synchronization. The SYNDET pin is then set high, and is reset automatically by a STATUS READ. If parity is programmed, SYNDET will not be set until the middle of the parity bit instead of the middle of the last data bit.

In the external SYNC mode, synchronization is achieved by applying a high level on the SYNDET pin, thus forcing the 8251A out of the HUNT mode. The high level can be removed after one $\overline{RxC}$ cycle. An ENTER HUNT command has no effect in the asynchronous mode of operation.

Parity error and overrun error are both checked in the same way as in the Asynchronous Rx mode. Parity is checked when not in Hunt, regardless of whether the Receiver is enabled or not.

The CPU can command the receiver to enter the HUNT mode if synchronization is lost. This will also set all the used character bits in the buffer to a "one", thus preventing a possible false SYNDET caused by data that happens to be in the Rx Buffer at ENTER HUNT time. Note that the SYNDET F/F is reset at each Status Read, regardless of whether internal or external SYNC has been programmed. This does not cause the 8251A to return to the HUNT mode. When in SYNC mode, but not in HUNT, Sync Detection is still functional, but only occurs at the "known" word boundaries. Thus, if one Status Read indicates SYNDET and a second Status Read also indicates SYNDET, then the programmed SYNDET characters have been received since the previous Status Read. (If double character sync has been programmed, then both sync characters have been contiguously received to gate a SYNDET indication.) When external SYNDET mode is selected, internal Sync Detect is disabled, and the SYNDET F/F may be set at any bit boundary.

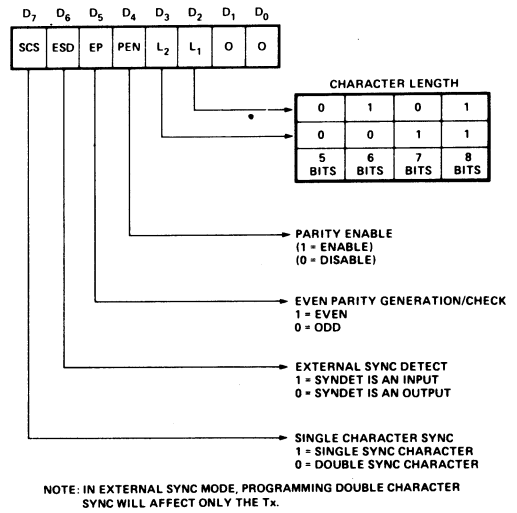


Figure 8. Mode Instruction Format

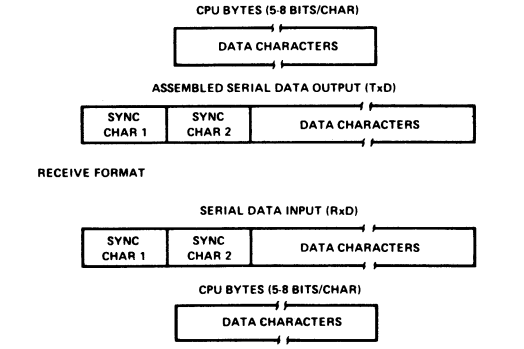
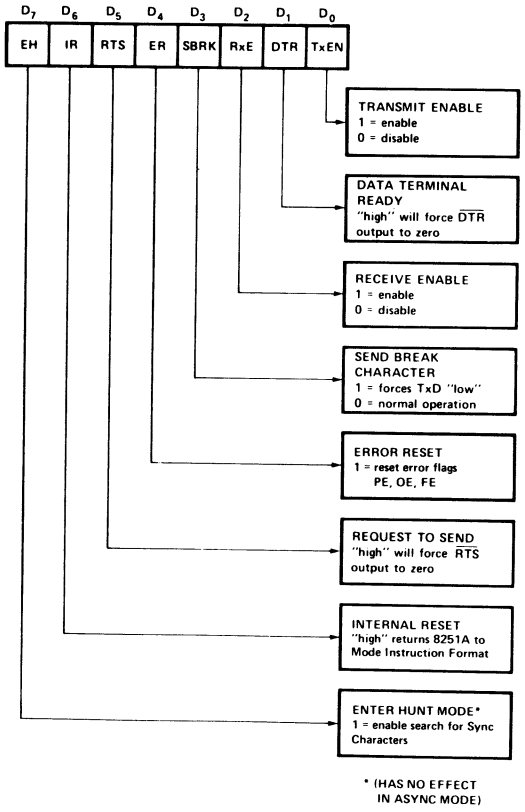


Figure 9. Data Format, Synchronous Mode

COMMAND INSTRUCTION DEFINITION

Once the functional definition of the 8251A has been programmed by the Mode Instruction and the Sync Characters are loaded (if in Sync Mode) then the device is ready to be used for data communication. The Command Instruction controls the actual operation of the selected format. Functions such as: Enable Transmit/Receive, Error Reset and Modem Controls are provided by the Command Instruction.

Once the Mode Instruction has been written into the 8251A and Sync characters inserted, if necessary, then all further "control writes" ($C/\overline{D}$ = 1) will load a Command Instruction. A Reset Operation (internal or external) will return the 8251A to the Mode Instruction format.



Note: Error Reset must be performed whenever RxEnable and Enter Hunt are programmed.

Figure 10. Command Instruction Format

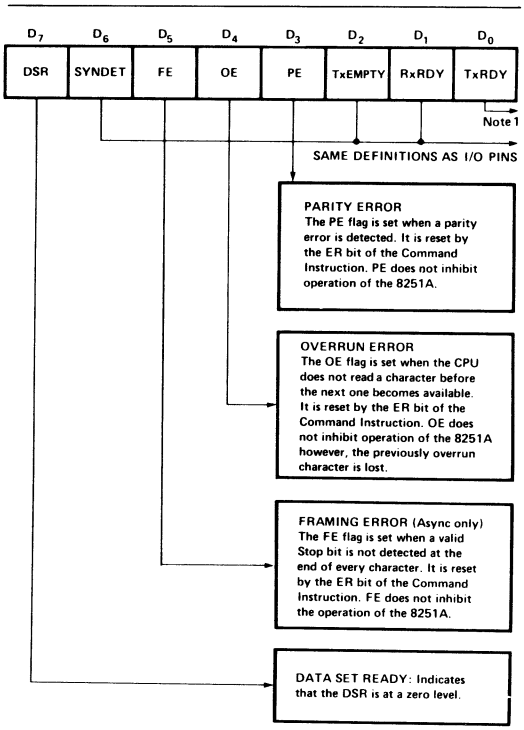
STATUS READ DEFINITION

In data communication systems it is often necessary to examine the "status" of the active device to ascertain if errors have occurred or other conditions that require the processor's attention. The 8251A has facilities that allow the programmer to "read" the status of the device at any time during the functional operation. (The status update is inhibited during status read).

A normal "read" command is issued by the CPU with $C/\overline{D}$ = 1 to accomplish this function.

Some of the bits in the Status Read Format have identical meanings to external output pins so that the 8251A can be used in a completely Polled environment or in an interrupt driven environment. TxRDY is an exception.

Note that status update can have a maximum delay of 28 clock periods from the actual event affecting the status.



Note 1: TxRDY status bit has different meanings from the TxRDY output pin: The former is not conditioned by $\overline{CTS}$ and TxEN; the latter is conditioned by both $\overline{CTS}$ and TxEN.

i.e. TxRDY status bit = DB Buffer Empty

TxRDY pin out = DB Buffer Empty · (CTS=0) · (TxEN=1)

Figure 11. Status Read Format

APPLICATIONS OF THE 8251A

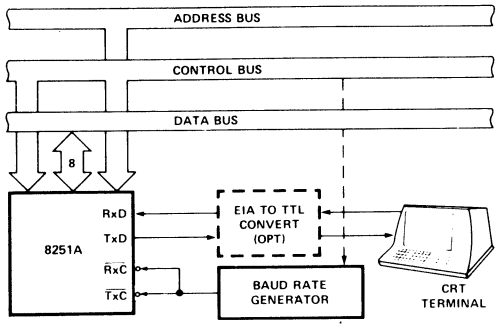


Figure 12. Asynchronous Serial Interface to CRT Terminal, DC—9600 Baud

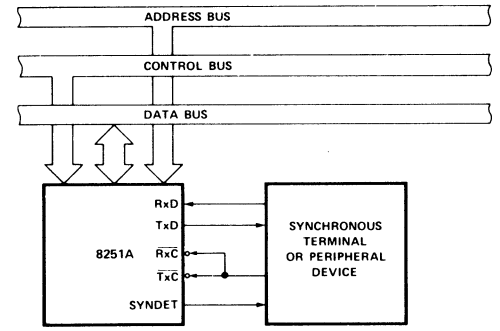


Figure 13. Synchronous Interface to Terminal or Peripheral Device

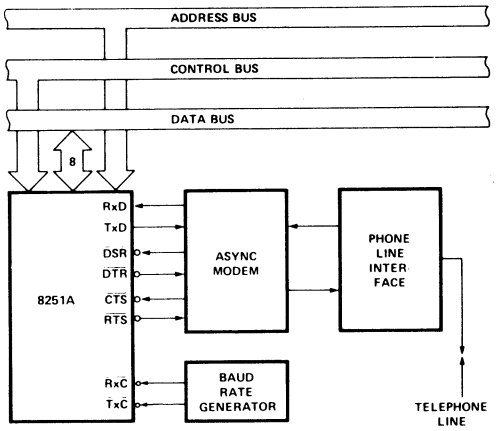


Figure 14. Asynchronous Interface to Telephone Lines

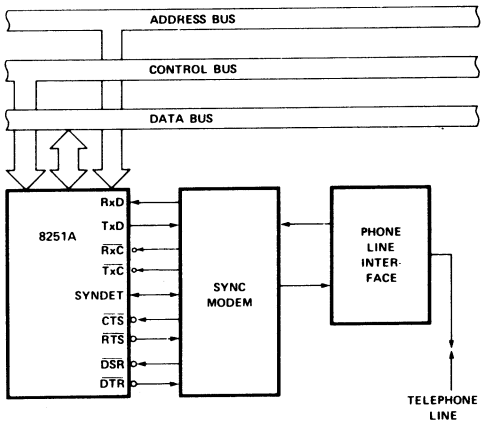


Figure 15. Synchronous Interface to Telephone Lines

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias. . . . . 0°C to 70°C
Storage Temperature . . . . . –65°C to +150°C
Voltage On Any Pin
With Respect to Ground . . . . . –0.5V to +7V
Power Dissipation . . . . . 1 Watt

COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS

T_A = 0°C to 70°C; V_{CC} = 5.0V ±5%; GND = 0V

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2.2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -400 μA
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} TO 0.45V
I_{IL}	Input Leakage		±10	μA	V_{IN} = V_{CC} TO 0.45V
I_{CC}	Power Supply Current		100	mA	All Outputs = High

CAPACITANCE

T_A = 25°C; V_{CC} = GND = 0V

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
C_{IN}	Input Capacitance		10	pF	fc = 1MHz
$C_{I/O}$	I/O Capacitance		20	pF	Unmeasured pins returned to GND

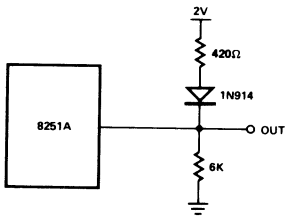


Figure 16. Test Load Circuit

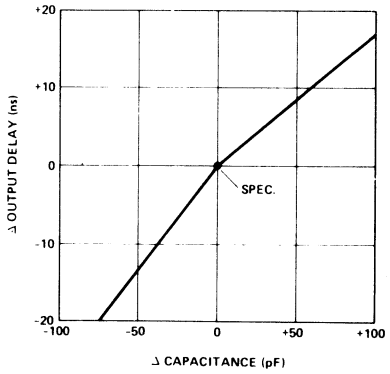


Figure 17. Typical Δ Output Delay vs. Δ Capacitance (pF)

A.C. CHARACTERISTICS

T_A = 0°C to 70°C; V_{CC} = 5.0V ±5%; GND = 0V

Bus Parameters (Note 1)

Read Cycle:

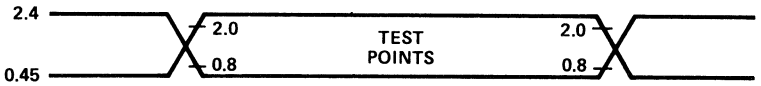
SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
t_{AR}	Address Stable Before $\overline{READ}$ ($\overline{CS}$, C/$\overline{D}$)	0		ns	Note 2
t_{RA}	Address Hold Time for $\overline{READ}$ ($\overline{CS}$, C/$\overline{D}$)	0		ns	Note 2
t_{RR}	$\overline{READ}$ Pulse Width	250		ns	
t_{RD}	Data Delay from $\overline{READ}$		200	ns	3, C_L = 150 pF
t_{DF}	$\overline{READ}$ to Data Floating	10	100	ns	

Write Cycle:

SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
t_{AW}	Address Stable Before $\overline{WRITE}$	0		ns	
t_{WA}	Address Hold Time for $\overline{WRITE}$	0		ns	
t_{WW}	$\overline{WRITE}$ Pulse Width	250		ns	
t_{DW}	Data Set Up Time for $\overline{WRITE}$	150		ns	
t_{WD}	Data Hold Time for $\overline{WRITE}$	0		ns	
t_{RV}	Recovery Time Between WRITES	6		t_{CY}	Note 4

NOTES: 1. AC timings measured V_{OH} = 2.0, V_{OL} = 0.8, and with load circuit of Figure 1.
2. Chip Select ($\overline{CS}$) and Command/Data (C/$\overline{D}$) are considered as Addresses.
3. Assumes that Address is valid before $\overline{RD}\downarrow$.
4. This recovery time is for Mode Initialization only. Write Data is allowed only when TxRDY = 1. Recovery Time between Writes for Asynchronous Mode is 8 t_{CY} and for Synchronous Mode is 16 t_{CY}.

Input Waveforms for AC Tests



Other Timings:

SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
t_{CY}	Clock Period	320	1.35	μs	Notes 5, 6
t_{ϕ}	Clock High Pulse Width	120	t_{CY-90}	ns	
$t_{\overline{\phi}}$	Clock Low Pulse Width	90		ns	
t_R, t_F	Clock Rise and Fall Time	5	20	ns	
t_{DTx}	TxD Delay from Falling Edge of $\overline{TxC}$		1	μs	
t_{SRx}	Rx Data Set-Up Time to Sampling Pulse	2		μs	
t_{HRx}	Rx Data Hold Time to Sampling Pulse	2		μs	
f_{Tx}	Transmitter Input Clock Frequency				
	1x Baud Rate	DC	64	kHz	
	16x Baud Rate	DC	310	kHz	
	64x Baud Rate	DC	615	kHz	
t_{TPW}	Transmitter Input Clock Pulse Width				
	1x Baud Rate	12		t_{CY}	
	16x and 64x Baud Rate	1		t_{CY}	
t_{TPD}	Transmitter Input Clock Pulse Delay				
	1x Baud Rate	15		t_{CY}	
	16x and 64x Baud Rate	3		t_{CY}	
f_{Rx}	Receiver Input Clock Frequency				
	1x Baud Rate	DC	64	kHz	
	16x Baud Rate	DC	310	kHz	
	64x Baud Rate	DC	615	kHz	
t_{RPW}	Receiver Input Clock Pulse Width				
	1x Baud Rate	12		t_{CY}	
	16x and 64x Baud Rate	1		t_{CY}	
t_{RPD}	Receiver Input Clock Pulse Delay				
	1x Baud Rate	15		t_{CY}	
	16x and 64x Baud Rate	3		t_{CY}	
t_{TxRDY}	TxRDY Pin Delay from Center of last Bit		8	t_{CY}	Note 7
$t_{TxRDY\ CLEAR}$	TxRDY ↓ from Leading Edge of $\overline{WR}$		150	ns	Note 7
t_{RxRDY}	RxRDY Pin Delay from Center of last Bit		24	t_{CY}	Note 7
$t_{RxRDY\ CLEAR}$	RxRDY ↓ from Leading Edge of $\overline{RD}$		150	ns	Note 7
t_{IS}	Internal SYNDET Delay from Rising Edge of $\overline{RxC}$		24	t_{CY}	Note 7
t_{ES}	External SYNDET Set-Up Time Before Falling Edge of $\overline{RxC}$		16	t_{CY}	Note 7
$t_{TxEMPTY}$	TxEMPTY Delay from Center of Data Bit		20	t_{CY}	Note 7
t_{WC}	Control Delay from Rising Edge of WRITE (TxEn, $\overline{DTR}$, $\overline{RTS}$)		8	t_{CY}	Note 7
t_{CR}	Control to READ Set-Up Time ($\overline{DSR}$, $\overline{CTS}$)		20	t_{CY}	Note 7

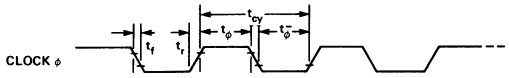
5. The TxC and RxC frequencies have the following limitations with respect to CLK.
   For 1x Baud Rate, f_{Tx} or $f_{Rx} \leq 1/(30\ t_{CY})$
   For 16x and 64x Baud Rate, f_{Tx} or $f_{Rx} \leq 1/(4.5\ t_{CY})$

6. Reset Pulse Width = 6 t_{CY} minimum; System Clock must be running during Reset.

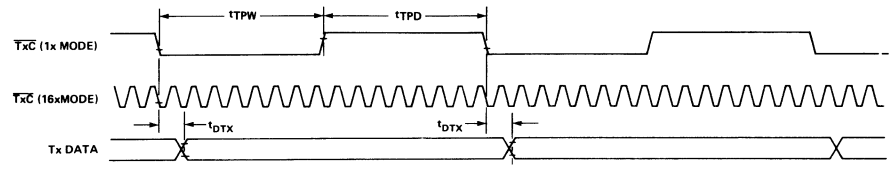
7. Status update can have a maximum delay of 28 clock periods from the event affecting the status.

WAVEFORMS

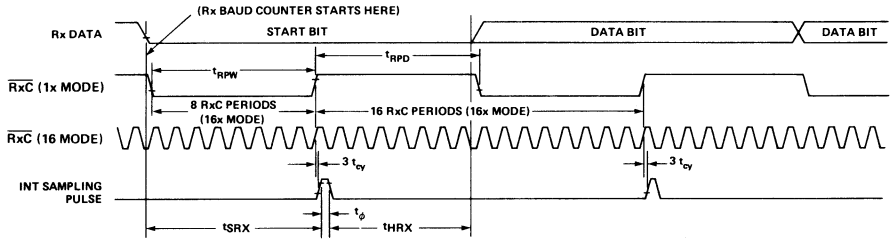
System Clock Input



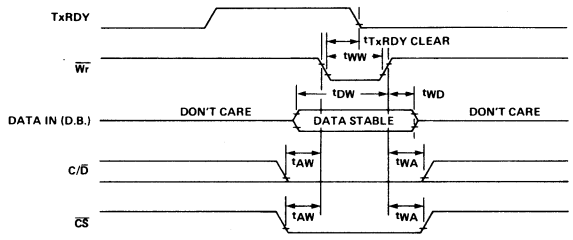
Transmitter Clock & Data



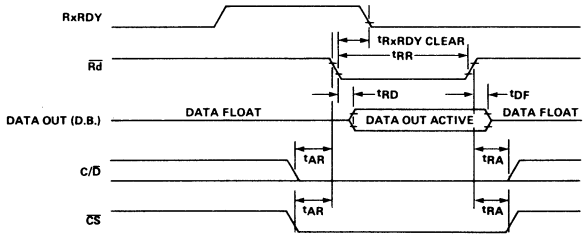
Receiver Clock & Data



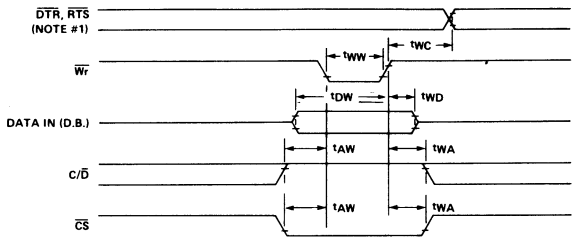
Write Data Cycle (CPU → USART)



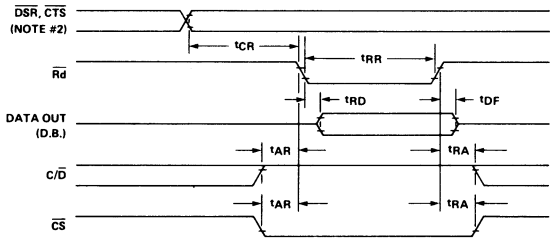
Read Data Cycle (CPU ← USART)



Write Control or Output Port Cycle (CPU → USART)

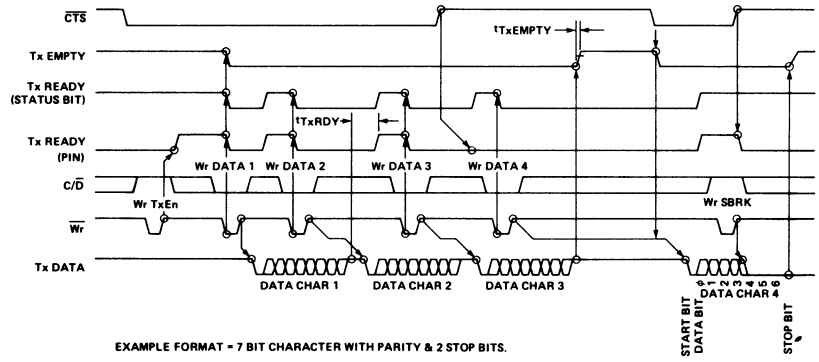


Read Control or Input Port (CPU ← USART)



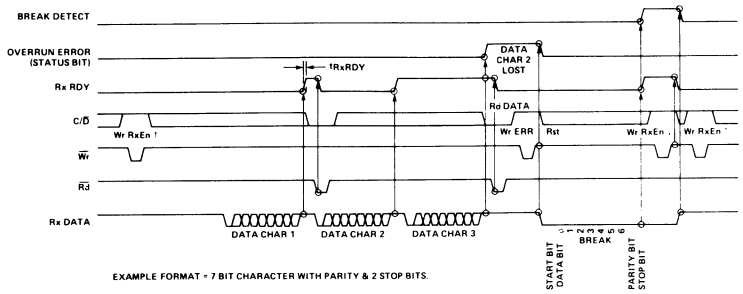
NOTE #1: T_{WC} INCLUDES THE RESPONSE TIMING OF A CONTROL BYTE.
NOTE #2: T_{CR} INCLUDES THE EFFECT OF CTS ON THE TxENBL CIRCUITRY.

Transmitter Control & Flag Timing (ASYNC Mode)



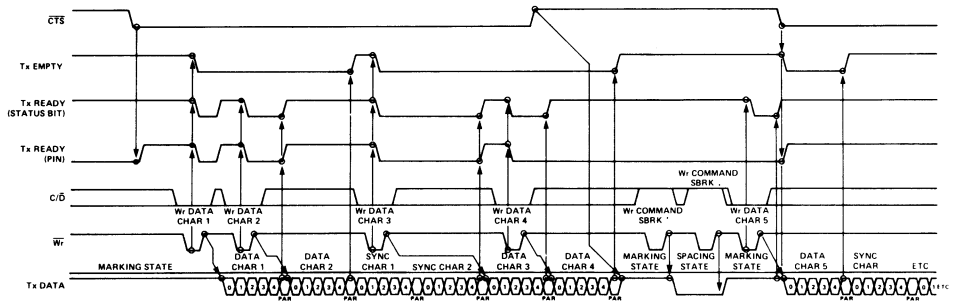
EXAMPLE FORMAT = 7 BIT CHARACTER WITH PARITY & 2 STOP BITS.

Receiver Control & Flag Timing (ASYNC Mode)



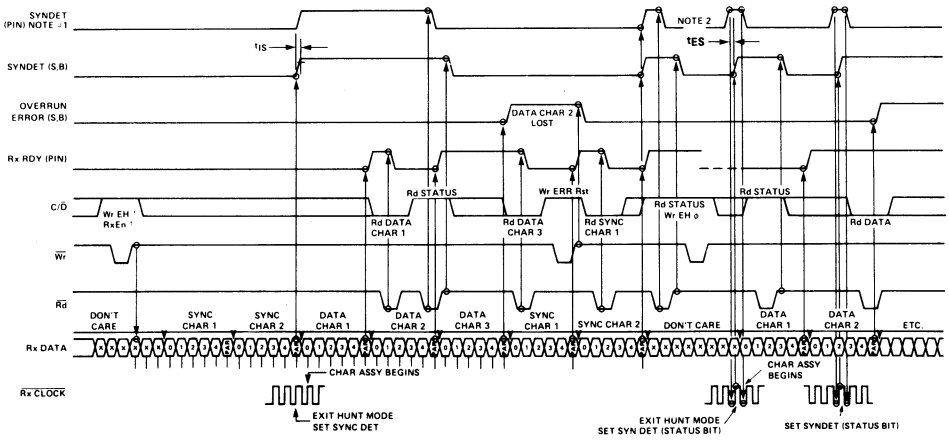
EXAMPLE FORMAT = 7 BIT CHARACTER WITH PARITY & 2 STOP BITS.

Transmitter Control & Flag Timing (SYNC Mode)



EXAMPLE FORMAT = 5 BIT CHARACTER WITH PARITY, 2 SYNC CHARACTERS.

Receiver Control & Flag Timing (SYNC Mode)



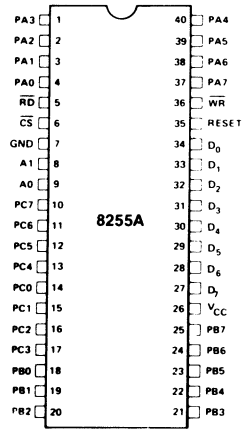
NOTE 1: INTERNAL SYNC, 2 SYNC CHARACTERS, 5 BITS, WITH PARITY
NOTE 2: EXTERNAL SYNC, 5 BITS, WITH PARITY

8255A/8255A-5
PROGRAMMABLE PERIPHERAL INTERFACE

- **MCS-85™ Compatible 8255A-5**
- **24 Programmable I/O Pins**
- **Completely TTL Compatible**
- **Fully Compatible with Intel® Microprocessor Families**
- **Improved Timing Characteristics**
- **Direct Bit Set/Reset Capability Easing Control Application Interface**
- **40-Pin Dual In-Line Package**
- **Reduces System Package Count**
- **Improved DC Driving Capability**

The Intel® 8255A is a general purpose programmable I/O device designed for use with Intel® microprocessors. It has 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. In the first mode (MODE 0), each group of 12 I/O pins may be programmed in sets of 4 to be input or output. In MODE 1, the second mode, each group may be programmed to have 8 lines of input or output. Of the remaining 4 pins, 3 are used for handshaking and interrupt control signals. The third mode of operation (MODE 2) is a bidirectional bus mode which uses 8 lines for a bidirectional bus, and 5 lines, borrowing one from the other group, for handshaking.

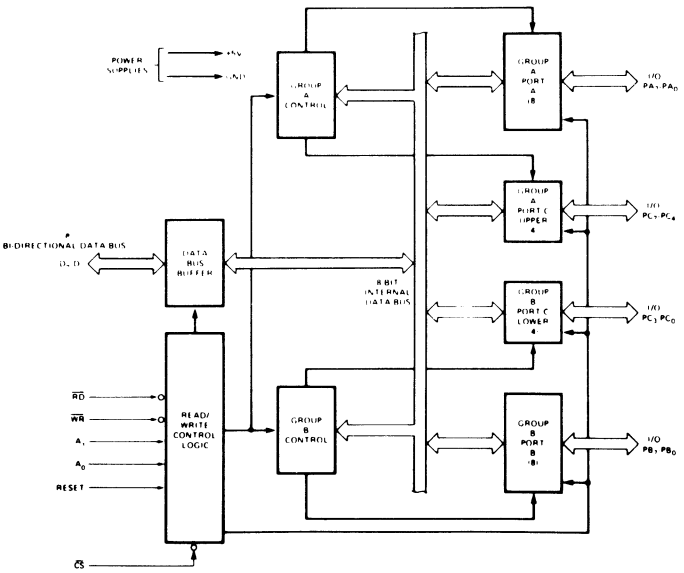
PIN CONFIGURATION



PIN NAMES

D_7-D_0	DATA BUS (BI-DIRECTIONAL)
RESET	RESET INPUT
$\overline{CS}$	CHIP SELECT
$\overline{RD}$	READ INPUT
$\overline{WR}$	WRITE INPUT
A0, A1	PORT ADDRESS
PA7-PA0	PORT A (BIT)
PB7-PB0	PORT B (BIT)
PC7-PC0	PORT C (BIT)
V_{CC}	+5 VOLTS
GND	0 VOLTS

8255A BLOCK DIAGRAM



February, 1978

8255A FUNCTIONAL DESCRIPTION

General

The 8255A is a programmable peripheral interface (PPI) device designed for use in Intel® microcomputer systems. Its function is that of a general purpose I/O component to interface peripheral equipment to the microcomputer system bus. The functional configuration of the 8255A is programmed by the system software so that normally no external logic is necessary to interface peripheral devices or structures.

Data Bus Buffer

This 3-state bidirectional 8-bit buffer is used to interface the 8255A to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

($\overline{CS}$)

Chip Select. A "low" on this input pin enables the communiction between the 8255A and the CPU.

($\overline{RD}$)

Read. A "low" on this input pin enables the 8255A to send the data or status information to the CPU on the data bus. In essence, it allows the CPU to "read from" the 8255A.

($\overline{WR}$)

Write. A "low" on this input pin enables the CPU to write data or control words into the 8255A.

(A_0 and A_1)

Port Select 0 and Port Select 1. These input signals, in conjunction with the RD and WR inputs, control the selection of one of the three ports or the control word registers. They are normally connected to the least significant bits of the address bus (A_0 and A_1).

8255A BASIC OPERATION

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	INPUT OPERATION (READ)
0	0	0	1	0	PORT A ⇒ DATA BUS
0	1	0	1	0	PORT B ⇒ DATA BUS
1	0	0	1	0	PORT C ⇒ DATA BUS
					OUTPUT OPERATION (WRITE)
0	0	1	0	0	DATA BUS ⇒ PORT A
0	1	1	0	0	DATA BUS ⇒ PORT B
1	0	1	0	0	DATA BUS ⇒ PORT C
1	1	1	0	0	DATA BUS ⇒ CONTROL
					DISABLE FUNCTION
X	X	X	X	1	DATA BUS ⇒ 3-STATE
1	1	0	1	0	ILLEGAL CONDITION
X	X	1	1	0	DATA BUS ⇒ 3-STATE

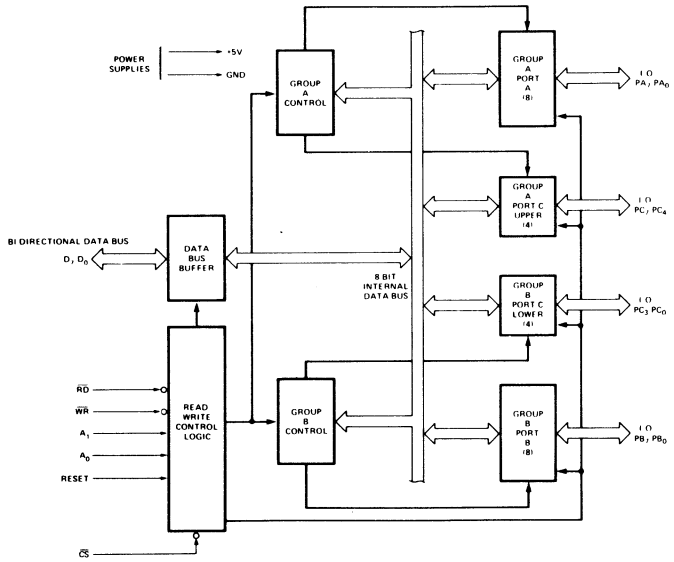


Figure 1. 8255A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions

(RESET)

Reset. A "high on this input clears the control register and all ports (A, C, C) are set to the input mode.

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 8255A. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 8255.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A – Port A and Port C upper (C7-C4)
Control Group B – Port B and Port C lower (C3-C0)

The Control Word Register can **Only** be written into. No Read operation of the Control Word Register is allowed.

Ports A, B, and C

The 8255A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 8255A.

Port A. One 8-bit data output latch/buffer and one 8-bit data input latch.

Port B. One 8-bit data input/output latch/buffer and one 8-bit data input buffer.

Port C. One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B.

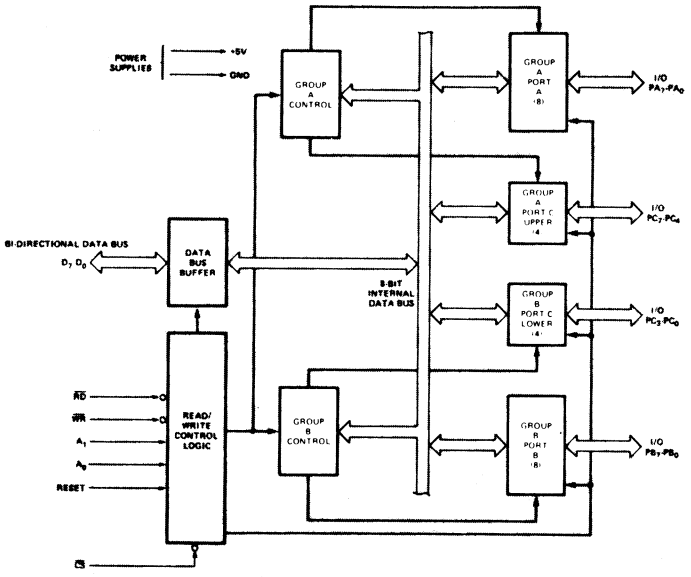
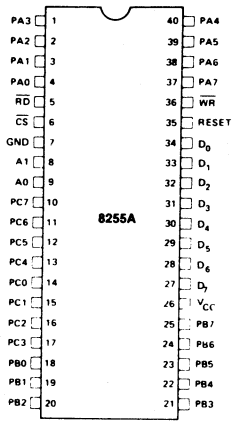


Figure 2. 8225A Block Diagram Showing Group A and Group B Control Functions

PIN CONFIGURATION



PIN NAMES

D_7-D_0	DATA BUS (BI DIRECTIONAL)
RESET	RESET INPUT
$\overline{CS}$	CHIP SELECT
$\overline{RD}$	READ INPUT
$\overline{WR}$	WRITE INPUT
A0, A1	PORT ADDRESS
PA7-PA0	PORT A (BIT)
PB7-PB0	PORT B (BIT)
PC7-PC0	PORT C (BIT)
V_{CC}	+5 VOLTS
GND	0 VOLTS

8255A OPERATIONAL DESCRIPTION

Mode Selection

There are three basic modes of operation that can be selected by the system software:

Mode 0 — Basic Input/Output
Mode 1 — Strobed Input/Output
Mode 2 — Bi-Directional Bus

When the reset input goes "high" all ports will be set to the input mode (i.e., all 24 lines will be in the high impedance state). After the reset is removed the 8255A can remain in the input mode with no additional initialization required. During the execution of the system program any of the other modes may be selected using a single output instruction. This allows a single 8255A to service a variety of peripheral devices with a simple software maintenance routine.

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance; Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.

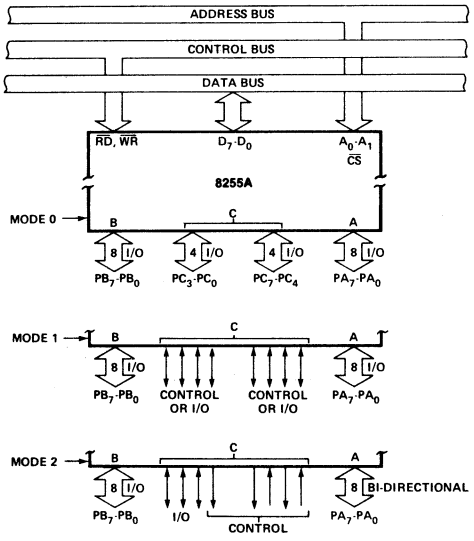


Figure 3. Basic Mode Definitions and Bus Interface

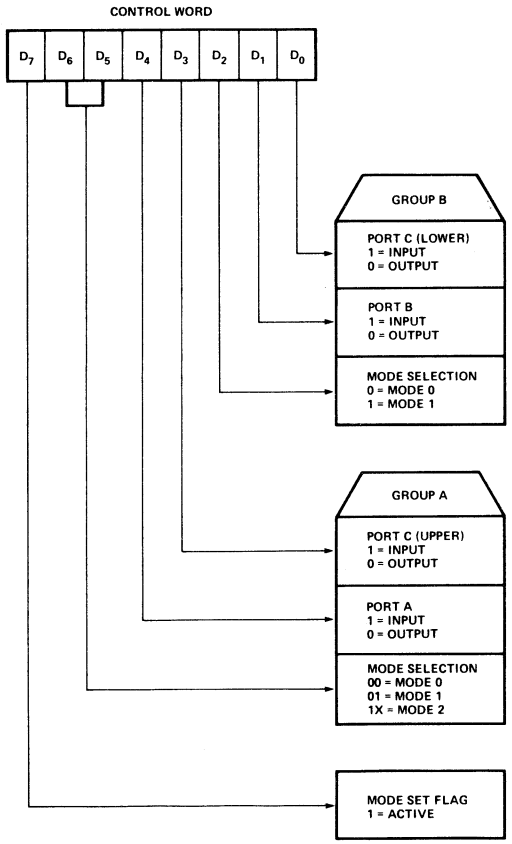


Figure 4. Mode Definition Format

The mode definitions and possible mode combinations may seem confusing at first but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 8255A has taken into account things such as efficient PC board layout, control signal definition vs PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

Single Bit Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTput instruction. This feature reduces software requirements in Control-based applications.

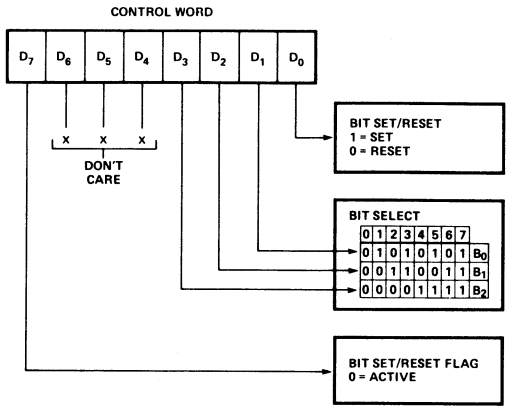


Figure 5. Bit Set/Reset Format

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were data output ports.

Interrupt Control Functions

When the 8255A is programmed to operate in mode 1 or mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the bit set/reset function of port C.

This function allows the Programmer to disallow or allow a specific I/O device to interrupt the CPU without affecting any other device in the interrupt structure.

INTE flip-flop definition:

(BIT-SET) — INTE is SET — Interrupt enable
(BIT-RESET) — INTE is RESET — Interrupt disable

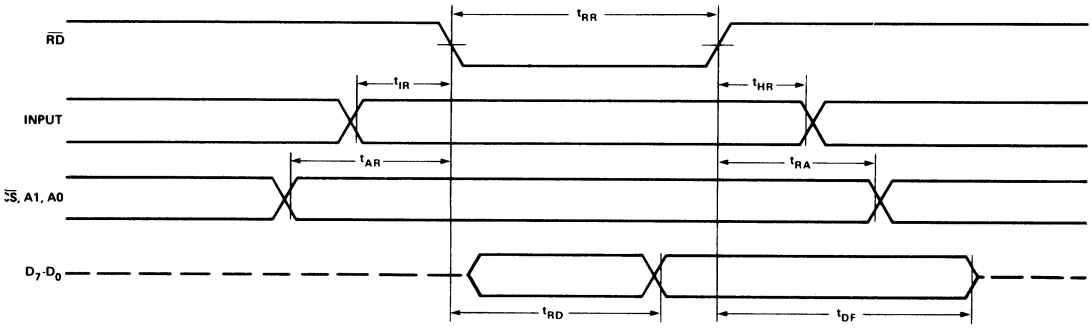
Note: All Mask flip-flops are automatically reset during mode selection and device Reset.

Operating Modes

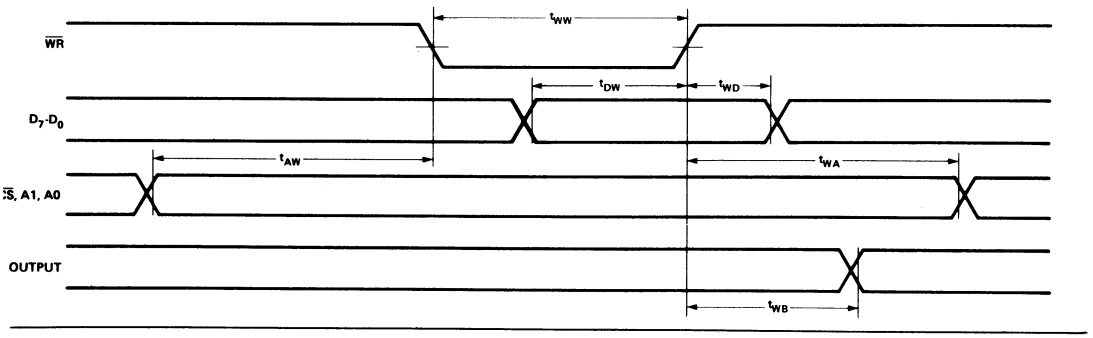
MODE 0 (Basic Input/Output). This functional configuration provides simple input and output operations for each of the three ports. No "handshaking" is required, data is simply written to or read from a specified port.

Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.



MODE 0 (Basic Input)

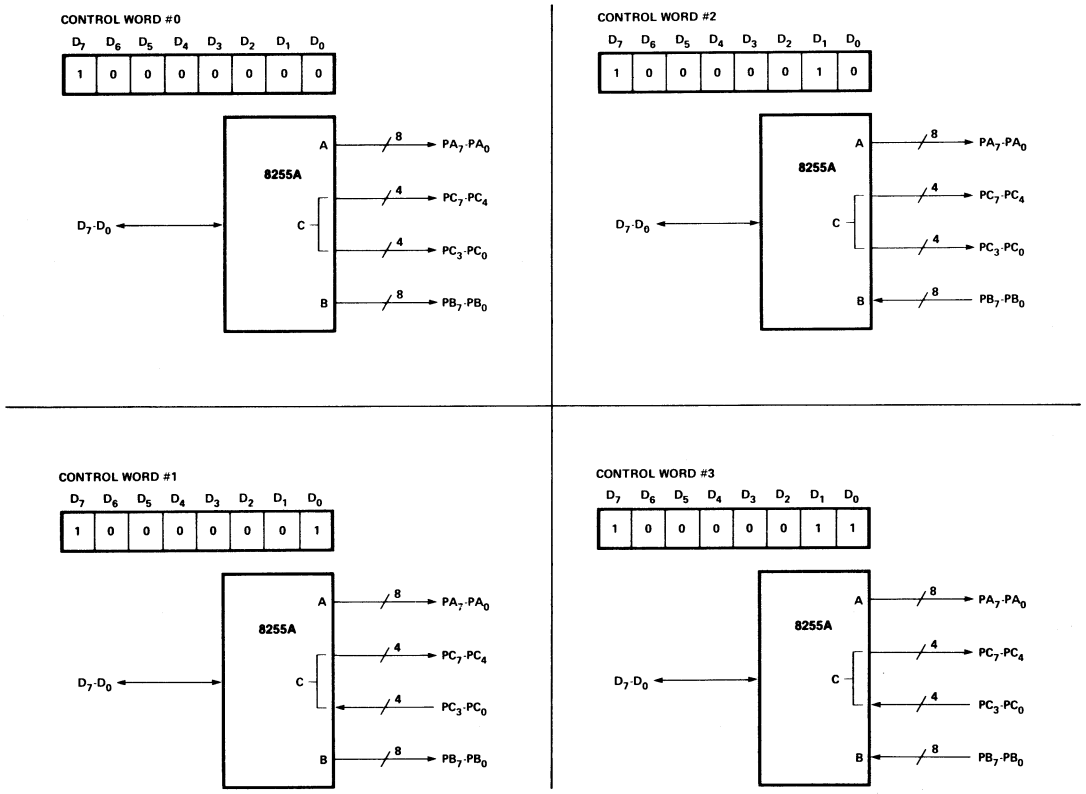


MODE 0 (Basic Output)

MODE 0 Port Definition

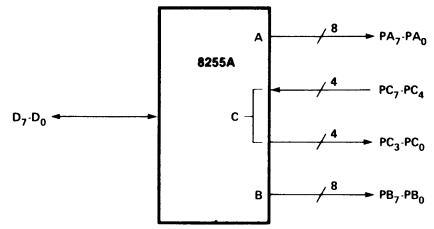
A		B		GROUP A			GROUP B	
D_4	D_3	D_1	D_0	PORT A	PORT C (UPPER)	#	PORT B	PORT C (LOWER)
0	0	0	0	OUTPUT	OUTPUT	0	OUTPUT	OUTPUT
0	0	0	1	OUTPUT	OUTPUT	1	OUTPUT	INPUT
0	0	1	0	OUTPUT	OUTPUT	2	INPUT	OUTPUT
0	0	1	1	OUTPUT	OUTPUT	3	INPUT	INPUT
0	1	0	0	OUTPUT	INPUT	4	OUTPUT	OUTPUT
0	1	0	1	OUTPUT	INPUT	5	OUTPUT	INPUT
0	1	1	0	OUTPUT	INPUT	6	INPUT	OUTPUT
0	1	1	1	OUTPUT	INPUT	7	INPUT	INPUT
1	0	0	0	INPUT	OUTPUT	8	OUTPUT	OUTPUT
1	0	0	1	INPUT	OUTPUT	9	OUTPUT	INPUT
1	0	1	0	INPUT	OUTPUT	10	INPUT	OUTPUT
1	0	1	1	INPUT	OUTPUT	11	INPUT	INPUT
1	1	0	0	INPUT	INPUT	12	OUTPUT	OUTPUT
1	1	0	1	INPUT	INPUT	13	OUTPUT	INPUT
1	1	1	0	INPUT	INPUT	14	INPUT	OUTPUT
1	1	1	1	INPUT	INPUT	15	INPUT	INPUT

MODE 0 Configurations



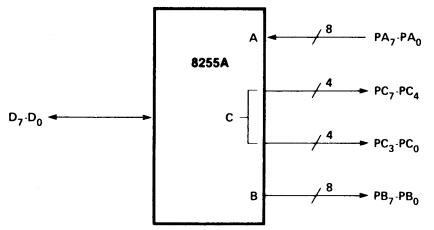
CONTROL WORD #4

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	0	0



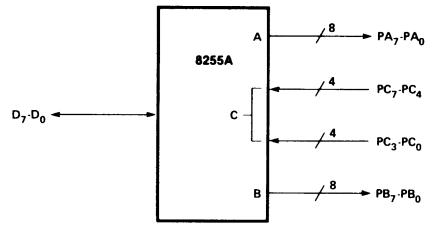
CONTROL WORD #8

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	0	0



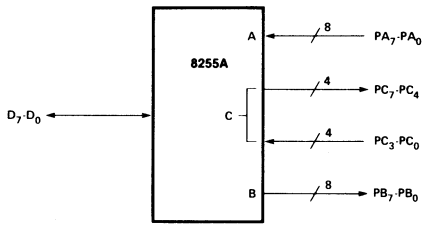
CONTROL WORD #5

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	0	1



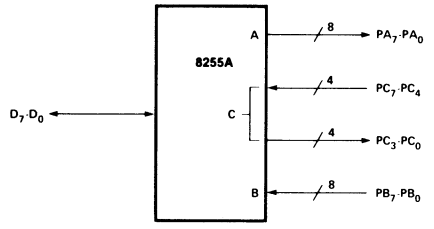
CONTROL WORD #9

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	0	1



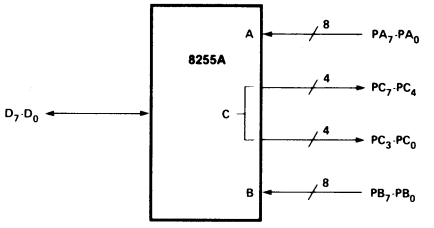
CONTROL WORD #6

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	1	0



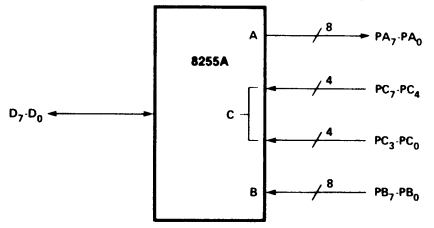
CONTROL WORD #10

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	1	0



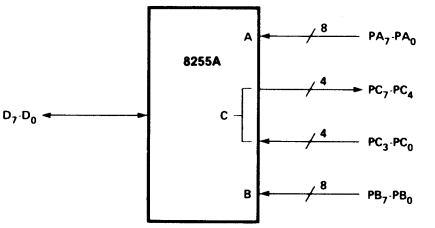
CONTROL WORD #7

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	1	1



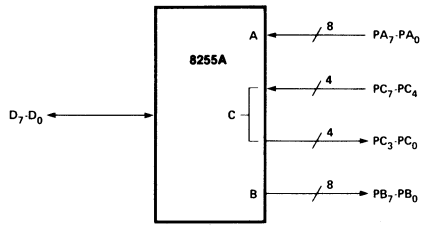
CONTROL WORD #11

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	1	1



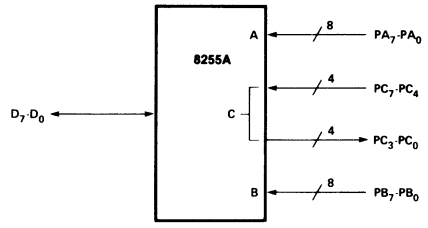
CONTROL WORD #12

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	1	0	0	0



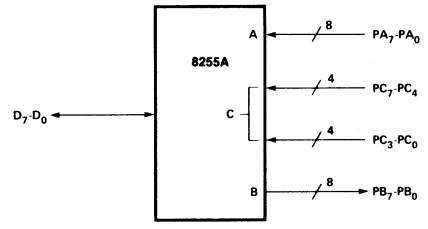
CONTROL WORD #14

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	1	0	1	0



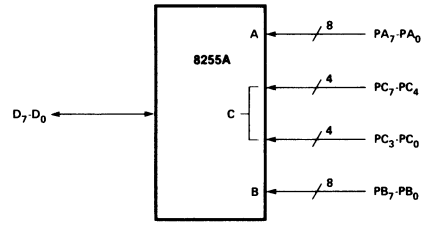
CONTROL WORD #13

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	1	0	0	1



CONTROL WORD #15

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	1	0	1	1



Operating Modes

MODE 1 (Strobed Input/Output). This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "handshaking" signals. In mode 1, port A and Port B use the lines on port C to generate or accept these "handshaking" signals.

Mode 1 Basic Functional Definitions:

- Two Groups (Group A and Group B)
- Each group contains one 8-bit data port and one 4-bit control/data port.
- The 8-bit data port can be either input or output. Both inputs and outputs are latched.
- The 4-bit port is used for control and status of the 8-bit data port.

Input Control Signal Definition

STB (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F)

A "high" on this output indicates that the data has been loaded into the input latch; in essence, an acknowledgement. IBF is set by STB input being low and is reset by the rising edge of the RD input.

INTR (Interrupt Request)

A "high" on this output can be used to interrupt the CPU when an input device is requesting service. INTR is set by the $\overline{STB}$ is a "one", IBF is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{RD}$. This procedure allows an input device to request service from the CPU by simply strobing its data into the port.

INTE A

Controlled by bit set/reset of PC_4.

INTE B

Controlled by bit set/reset of PC_2.

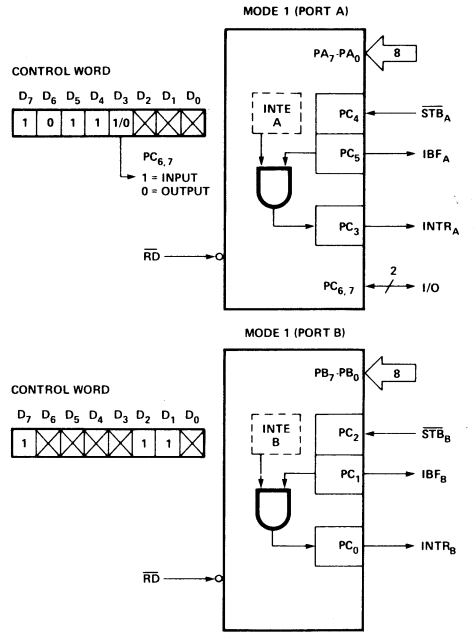


Figure 6. MODE 1 Input

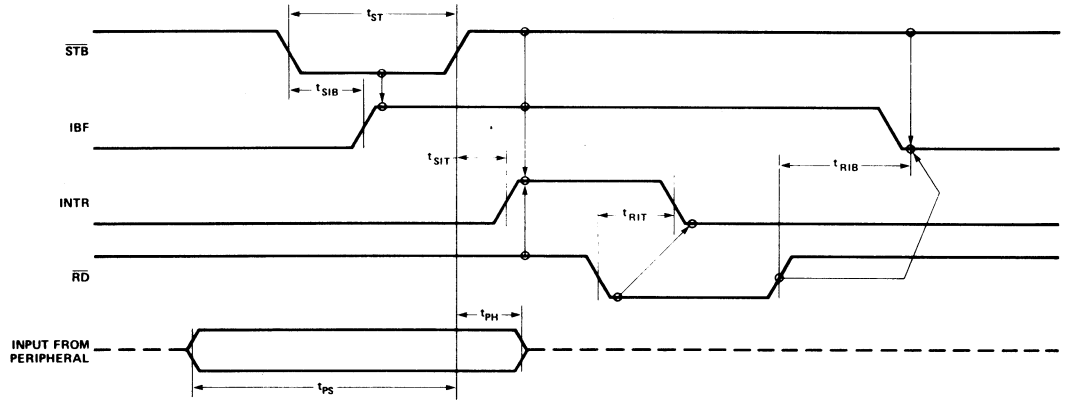


Figure 7. MODE 1 (Strobed Input)

Output Control Signal Definition

$\overline{OBF}$ (Output Buffer Full F/F). The OBF output will go "low" to indicate that the CPU has written data out to the specified port. The OBF F/F will be set by the rising edge of the WR input and reset by ACK Input being low.

$\overline{ACK}$ (Acknowledge Input). A "low" on this input informs the 8255A that the data from port A or port B has been accepted. In essence, a response from the peripheral device indicating that it has received the data output by the CPU.

INTR (Interrupt Request). A "high" on this output can be used to interrupt the CPU when an output device has accepted data transmitted by the CPU. INTR is set when ACK is a "one", OBF is a "one" and INTE is a "one". It is reset by the falling edge of WR.

INTE A

Controlled by bit set/reset of PC_6.

INTE B

Controlled by bit set/reset of PC_2.

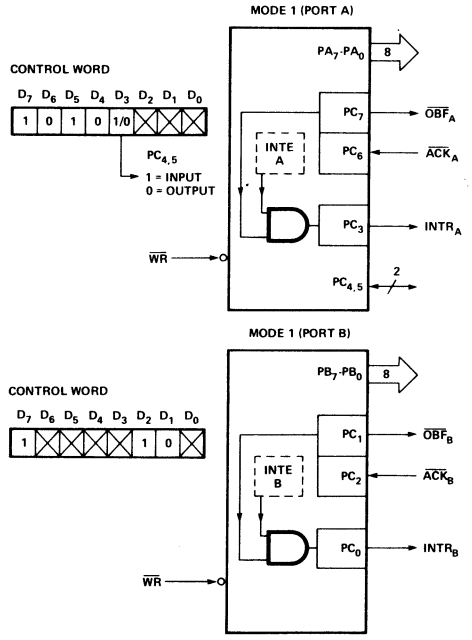


Figure 8. MODE 1 Output

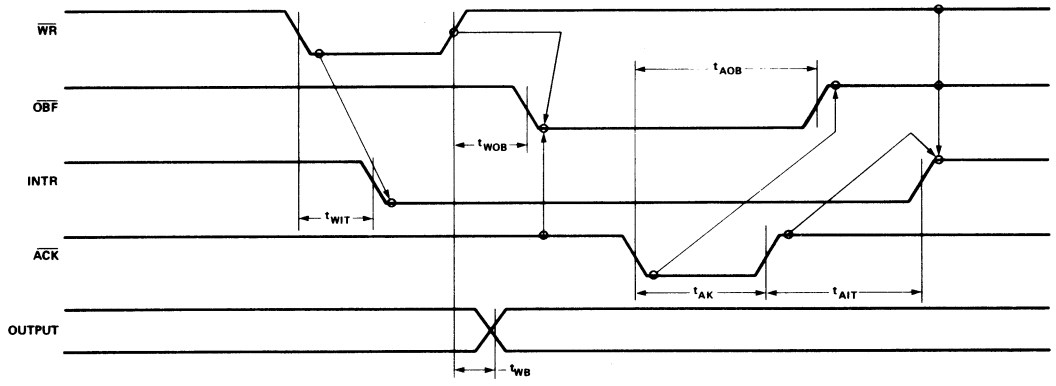


Figure 9. Mode 1 (Strobed Output)

Combinations of MODE 1

Port A and Port B can be individually defined as input or output in Mode 1 to support a wide variety of strobed I/O applications.

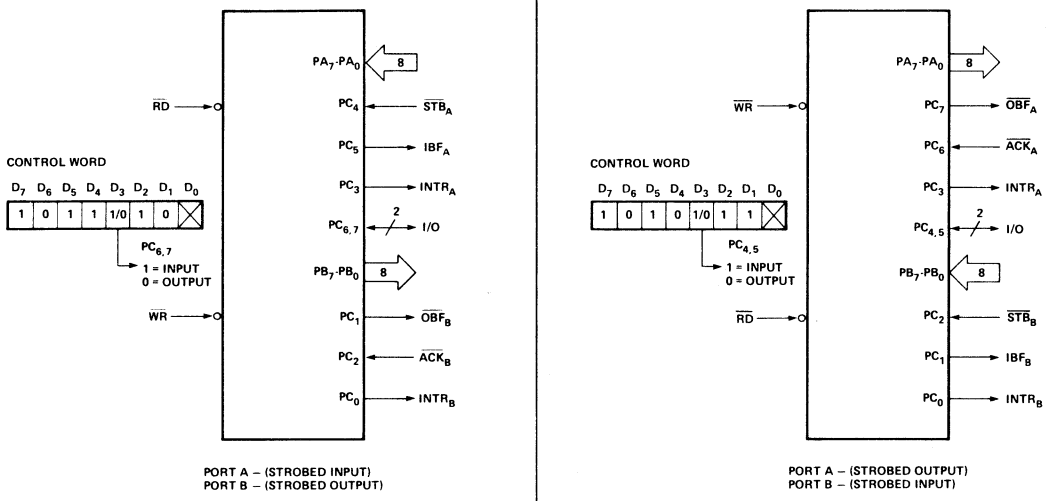


Figure 10. Combinations of MODE 1

Operating Modes

MODE 2 (Strobed Bidirectional Bus I/O). This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bidirectional bus I/O). "Handshaking" signals are provided to maintain proper bus flow discipline in a similar manner to MODE 1. Interrupt generation and enable/disable functions are also available.

MODE 2 Basic Functional Definitions:

- Used in Group A only.
- One 8-bit, bi-directional bus Port (Port A) and a 5-bit control Port (Port C).
- Both inputs and outputs are latched.
- The 5-bit control port (Port C) is used for control and status for the 8-bit, bi-directional bus port (Port A).

Bidirectional Bus I/O Control Signal Definition

INTR (Interrupt Request). A high on this output can be used to interrupt the CPU for both input or output operations.

Output Operations

$\overline{OBF}$ (Output Buffer Ful). The OBF output will go "low" to indicate that the CPU has written data out to port A.

$\overline{ACK}$ (Acknowledge). A "low" on this input enables the tri-state output buffer of port A to send out the data. Otherwise, the output buffer will be in the high impedance state.

INTE 1 (The INTE Flip-Flop Associated with OBF). Controlled by bit set/reset of PC_6.

Input Operations

$\overline{STB}$ (Strobe Input)

STB (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F). A "high" on this output indicates that data has been loaded into the input latch.

INTE 2 (The INTE Flip-Flop Associated with IBF). Controlled by bit set/reset of PC_4.

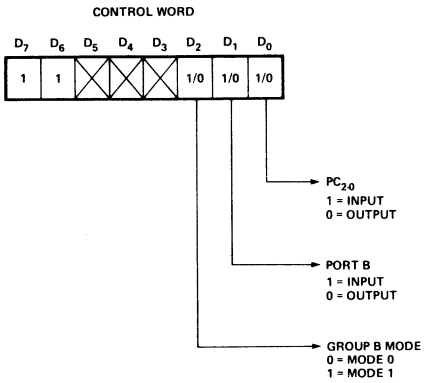


Figure 11. MODE Control Word

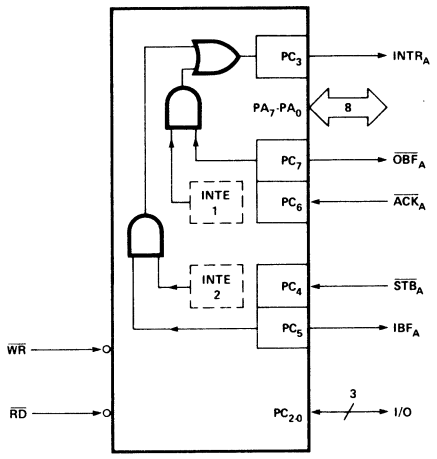


Figure 12. MODE 2

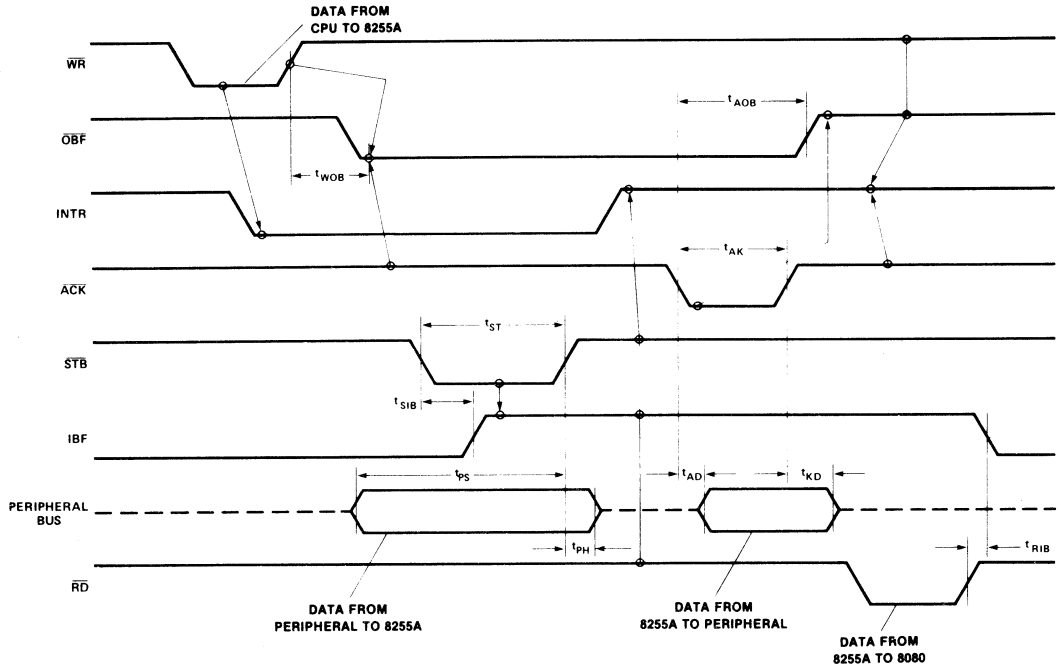


Figure 13. MODE 2 (Bidirectional)

NOTE: Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$ and $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
($INTR = IBF \cdot \overline{MASK} \cdot \overline{STB} \cdot \overline{RD} + \overline{OBF} \cdot \overline{MASK} \cdot \overline{ACK} \cdot \overline{WR}$)

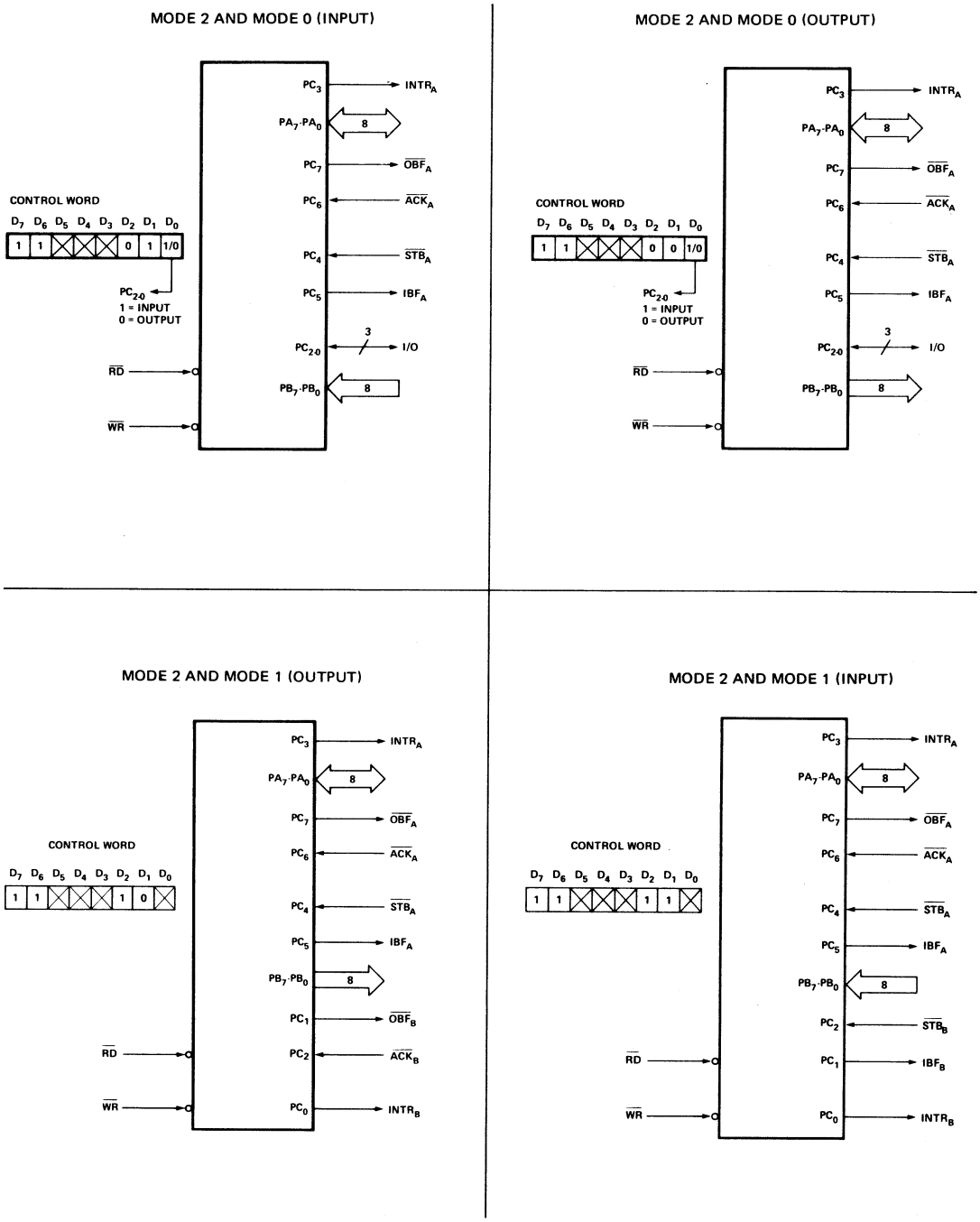


Figure 14. MODE 2 Combinations

Mode Definition Summary

	MODE 0		MODE 1		MODE 2	
	IN	OUT	IN	OUT	GROUP A ONLY	
PA_0	IN	OUT	IN	OUT	◄—►	
PA_1	IN	OUT	IN	OUT	◄—►	
PA_2	IN	OUT	IN	OUT	◄—►	
PA_3	IN	OUT	IN	OUT	◄—►	
PA_4	IN	OUT	IN	OUT	◄—►	
PA_5	IN	OUT	IN	OUT	◄—►	
PA_6	IN	OUT	IN	OUT	◄—►	
PA_7	IN	OUT	IN	OUT	◄—►	
PB_0	IN	OUT	IN	OUT	—	MODE 0 OR MODE 1 ONLY
PB_1	IN	OUT	IN	OUT	—	
PB_2	IN	OUT	IN	OUT	—	
PB_3	IN	OUT	IN	OUT	—	
PB_4	IN	OUT	IN	OUT	—	
PB_5	IN	OUT	IN	OUT	—	
PB_6	IN	OUT	IN	OUT	—	
PB_7	IN	OUT	IN	OUT	—	
PC_0	IN	OUT	$INTR_B$	$INTR_B$	I/O	
PC_1	IN	OUT	IBF_B	$\overline{OBF}_B$	I/O	
PC_2	IN	OUT	$\overline{STB}_B$	$\overline{ACK}_B$	I/O	
PC_3	IN	OUT	$INTR_A$	$INTR_A$	$INTR_A$	
PC_4	IN	OUT	$\overline{STB}_A$	I/O	$\overline{STB}_A$	
PC_5	IN	OUT	IBF_A	I/O	IBF_A	
PC_6	IN	OUT	I/O	$\overline{ACK}_A$	$\overline{ACK}_A$	
PC_7	IN	OUT	I/O	$\overline{OBF}_A$	$\overline{OBF}_A$	

Special Mode Combination Considerations

There are several combinations of modes when not all of the bits in Port C are used for control or status. The remaining bits can be used as follows:

If Programmed as Inputs –
All input lines can be accessed during a normal Port C read.

If Programmed as Outputs –
Bits in C upper (PC_7-PC_4) must be individually accessed using the bit set/reset function.

Bits in C lower (PC_3-PC_0) can be accessed using the bit set/reset function or accessed as a threesome by writing into Port C.

Source Current Capability on Port B and Port C

Any set of eight output buffers, selected randomly from Ports B and C can source 1mA at 1.5 volts. This feature allows the 8255 to directly drive Darlington type drivers and high-voltage displays that require such source current.

Reading Port C Status

In Mode 0, Port C transfers data to or from the peripheral device. When the 8255 is programmed to function in Modes 1 or 2, Port C generates or accepts "hand-shaking" signals with the peripheral device. Reading the contents of Port C allows the programmer to test or verify the "status" of each peripheral device and change the program flow accordingly.

There is no special instruction to read the status information from Port C. A normal read operation of Port C is executed to perform this function.

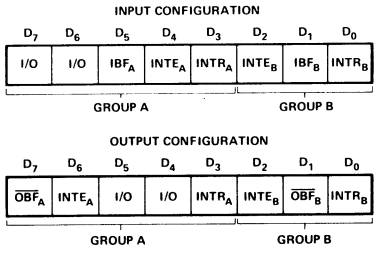


Figure 15. MODE 1 Status Word Format

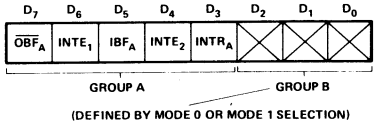


Figure 16. MODE 2 Status Word Format

APPLICATIONS OF THE 8255A

The 8255A is a very powerful tool for interfacing peripheral equipment to the microcomputer system. It represents the optimum use of available pins and is flexible enough to interface almost any I/O device without the need for additional external logic.

Each peripheral device in a microcomputer system usually has a "service routine" associated with it. The routine manages the software interface between the device and the CPU. The functional definition of the 8255A is programmed by the I/O service routine and becomes an extension of the system software. By examining the I/O devices interface characteristics for both data transfer and timing, and matching this information to the examples and tables in the detailed operational description, a control word can easily be developed to initialize the 8255A to exactly "fit" the application. Figures 17 through 23 present a few examples of typical applications of the 8255A.

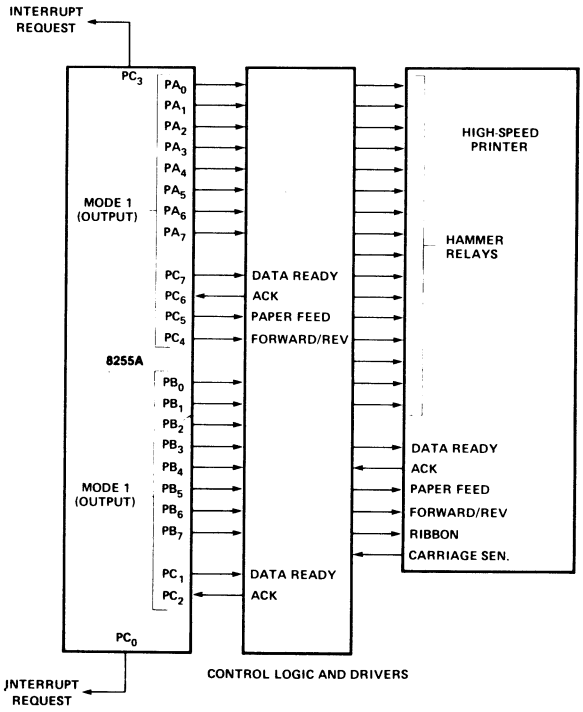


Figure 17. Printer Interface

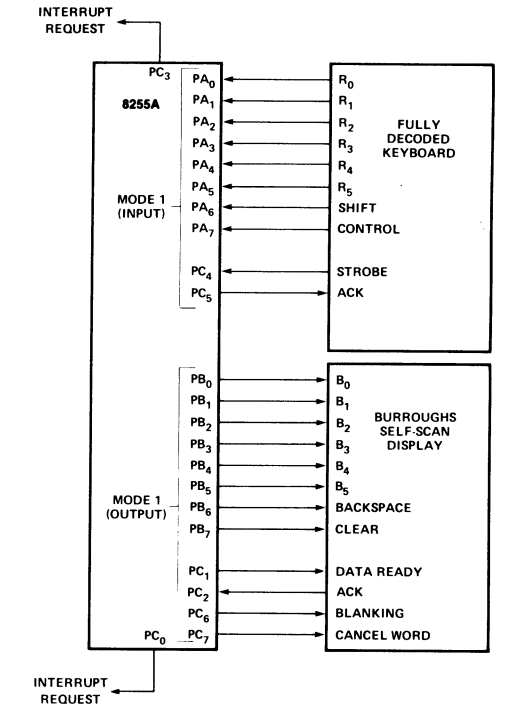


Figure 18. Keyboard and Display Interface

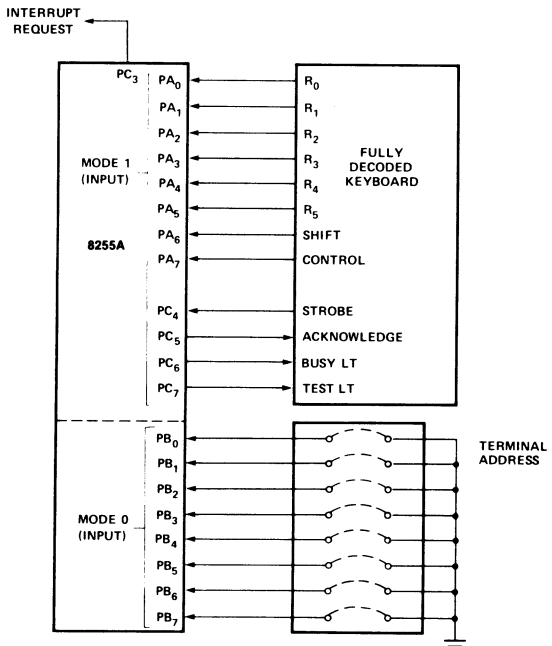


Figure 19. Keyboard and Terminal Address Interface

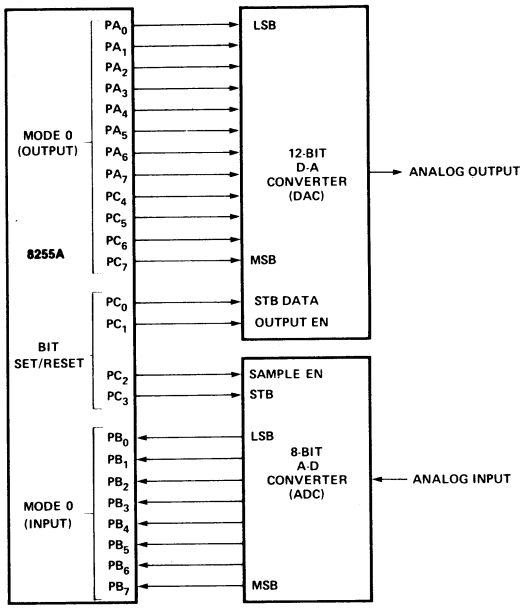


Figure 20. Digital to Analog, Analog to Digital

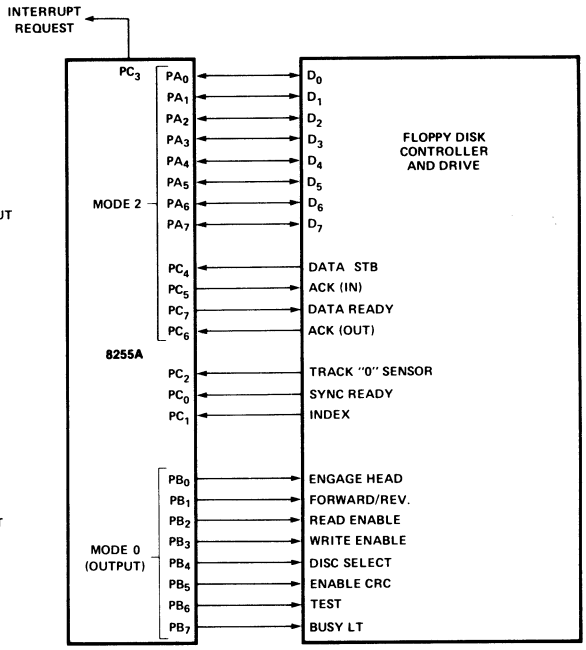


Figure 22. Basic Floppy Disc Interface

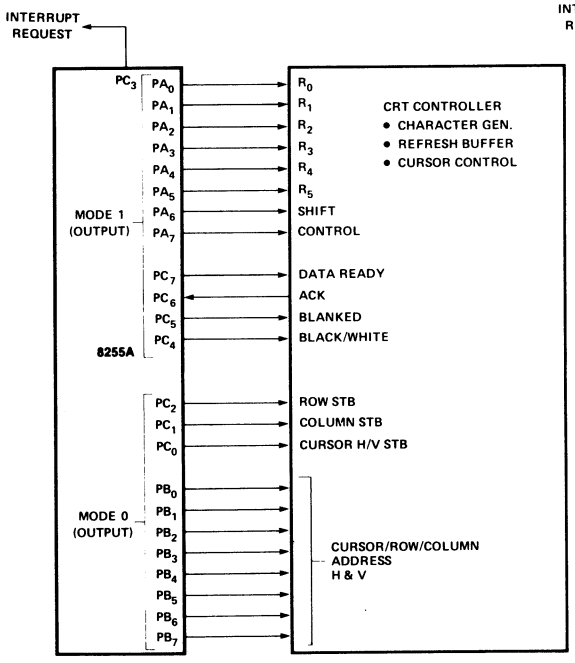


Figure 21. Basic CRT Controller Interface

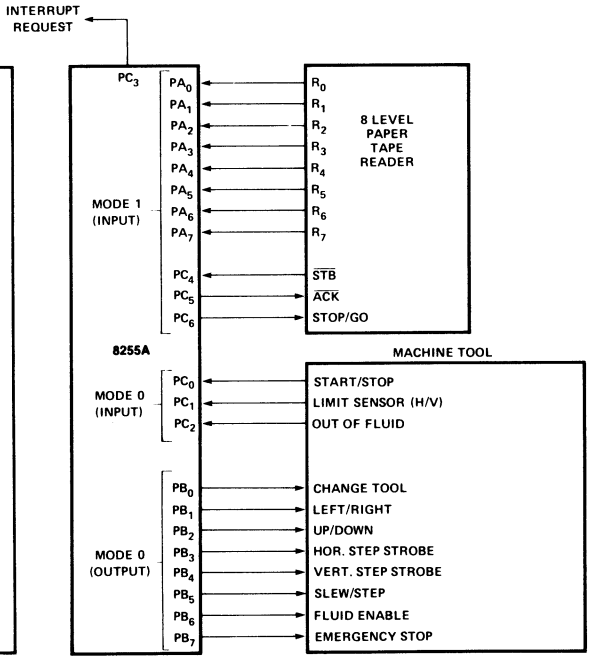


Figure 23. Machine Tool Controller Interface

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias. . . . . 0°C to 70°C
Storage Temperature . . . . . –65°C to +150°C
Voltage on Any Pin
With Respect to Ground . . . . . –0.5V to +7V
Power Dissipation . . . . . 1 Watt

COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ±5%; GND = 0V

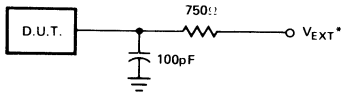
SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL} (DB)	Output Low Voltage (Data Bus)		0.45	V	I_{OL} = 2.5mA
V_{OL}(PER)	Output Low Voltage (Peripheral Port)		0.45	V	I_{OL} = 1.7mA
V_{OH}(DB)	Output High Voltage (Data Bus)	2.4		V	I_{OH} = -400μA
V_{OH}(PER)	Output High Voltage (Peripheral Port)	2.4		V	I_{OH} = -200μA
I_{DAR}[1]	Darlington Drive Current	-1.0	-4.0	mA	R_{EXT} = 750Ω; V_{EXT} = 1.5V
I_{CC}	Power Supply Current		120	mA	
I_{IL}	Input Load Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} to 0V

Note 1: Available on any 8 pins from Port B and C.

CAPACITANCE

T_A = 25°C; V_{CC} = GND = 0V

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT	TEST CONDITIONS
C_{IN}	Input Capacitance			10	pF	fc = 1MHz
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to GND



*V_{EXT} is set at various voltages during testing to guarantee the specification.

Figure 24. Test Load Circuit (for dB)

A.C. CHARACTERISTICS

T_A = 0°C to 70°C; V_{CC} = +5V ±5%; GND = 0V

NOTE:
The 8255A-5 specifications are not final. Some parametric limits are subject to change.

Bus Parameters

Read:

SYMBOL	PARAMETER	8255A		8255A-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{AR}	Address Stable Before READ	0		0		ns
t_{RA}	Address Stable After READ	0		0		ns
t_{RR}	READ Pulse Width	300		300		ns
t_{RD}	Data Valid From READ[1]		250		200	ns
t_{DF}	Data Float After READ	10	150	10	100	ns
t_{RV}	Time Between READs and/or WRITEs	850		850		ns

Write:

SYMBOL	PARAMETER	8255A		8255A-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{AW}	Address Stable Before WRITE	0		0		ns
t_{WA}	Address Stable After WRITE	20		20		ns
t_{WW}	WRITE Pulse Width	400		300		ns
t_{DW}	Data Valid to WRITE (T.E.)	100		100		ns
t_{WD}	Data Valid After WRITE	30		30		ns

Other Timings:

SYMBOL	PARAMETER	8255A		8255A-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{WB}	WR = 1 to Output[1]		350		350	ns
t_{IR}	Peripheral Data Before RD	0		0		ns
t_{HR}	Peripheral Data After RD	0		0		ns
t_{AK}	ACK Pulse Width	300		300		ns
t_{ST}	STB Pulse Width	500		500		ns
t_{PS}	Per. Data Before T.E. of STB	0		0		ns
t_{PH}	Per. Data After T.E. of STB	180		180		ns
t_{AD}	ACK = 0 to Output[1]		300		300	ns
t_{KD}	ACK = 1 to Output Float	20	250	20	250	ns
t_{WOB}	WR = 1 to OBF = 0[1]		650		650	ns
t_{AOB}	ACK = 0 to OBF = 1[1]		350		350	ns
t_{SIB}	STB = 0 to IBF = 1[1]		300		300	ns
t_{RIB}	RD = 1 to IBF = 0[1]		300		300	ns
t_{RIT}	RD = 0 to INTR = 0[1]		400		400	ns
t_{SIT}	STB = 1 to INTR = 1[1]		300		300	ns
t_{AIT}	ACK = 1 to INTR = 1[1]		350		350	ns
t_{WIT}	WR = 0 to INTR = 0[1]		850		850	ns

Notes: 1. Test Conditions: 8255A: C_L = 100pF; 8255A-5: C_L = 150pF.
2. Period of Reset pulse must be at least 50μs during or after power on. Subsequent Reset pulse can be 500 ns min.

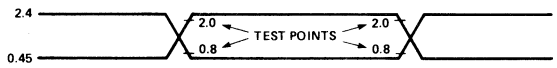


Figure 25. Input Waveforms for A.C. Tests

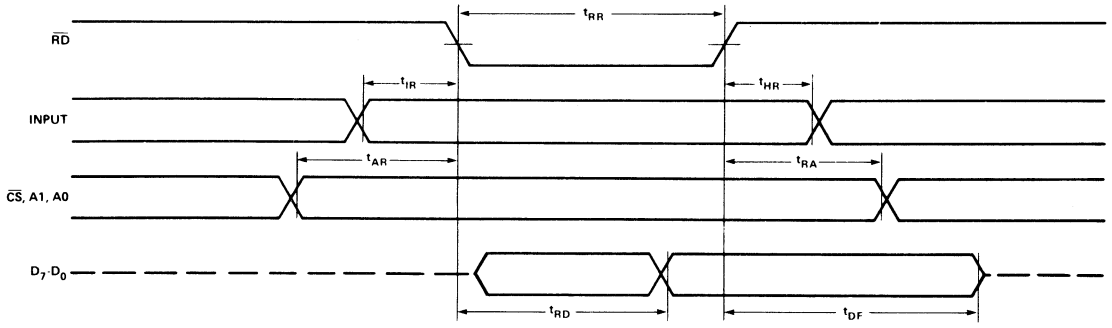


Figure 26. MODE 0 (Basic Input)

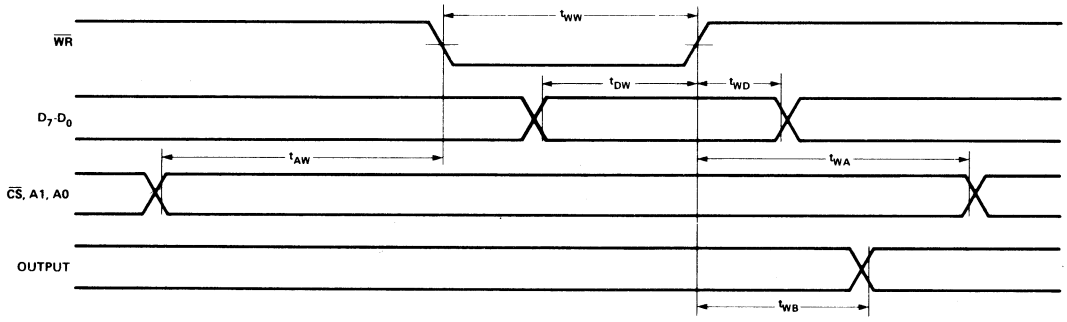


Figure 27. MODE 0 (Basic Output)

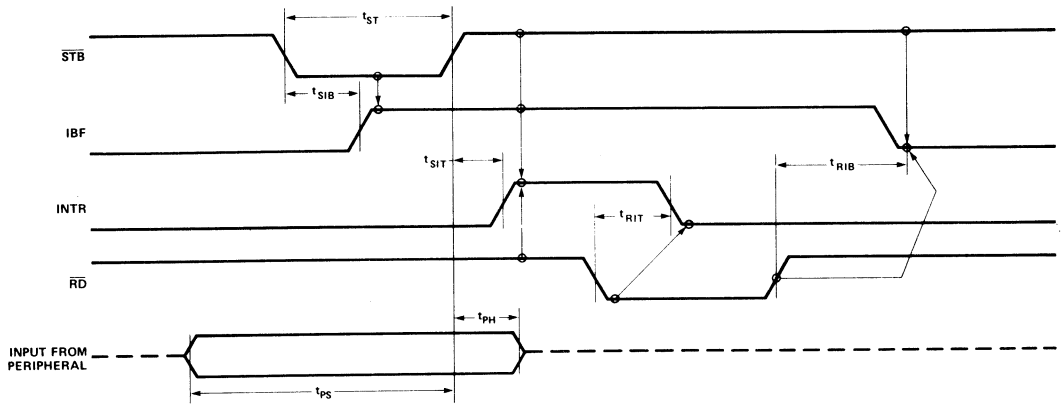


Figure 28. MODE 1 (Strobed Inut)

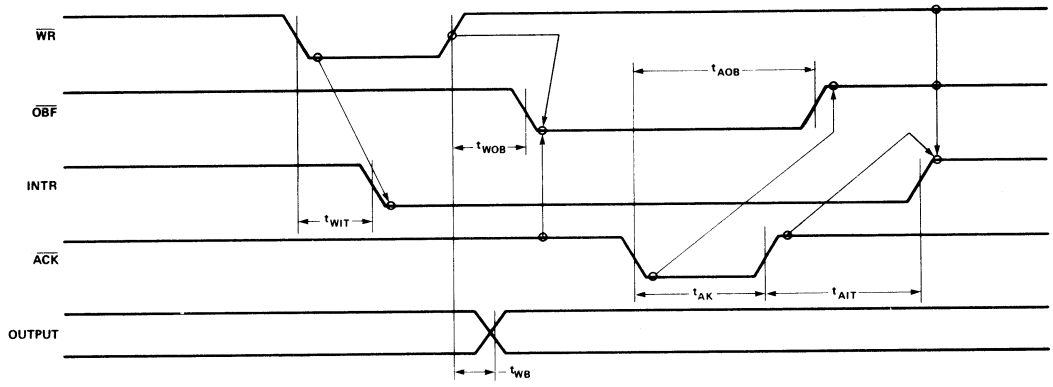


Figure 29. MODE 1 (Strobed Output)

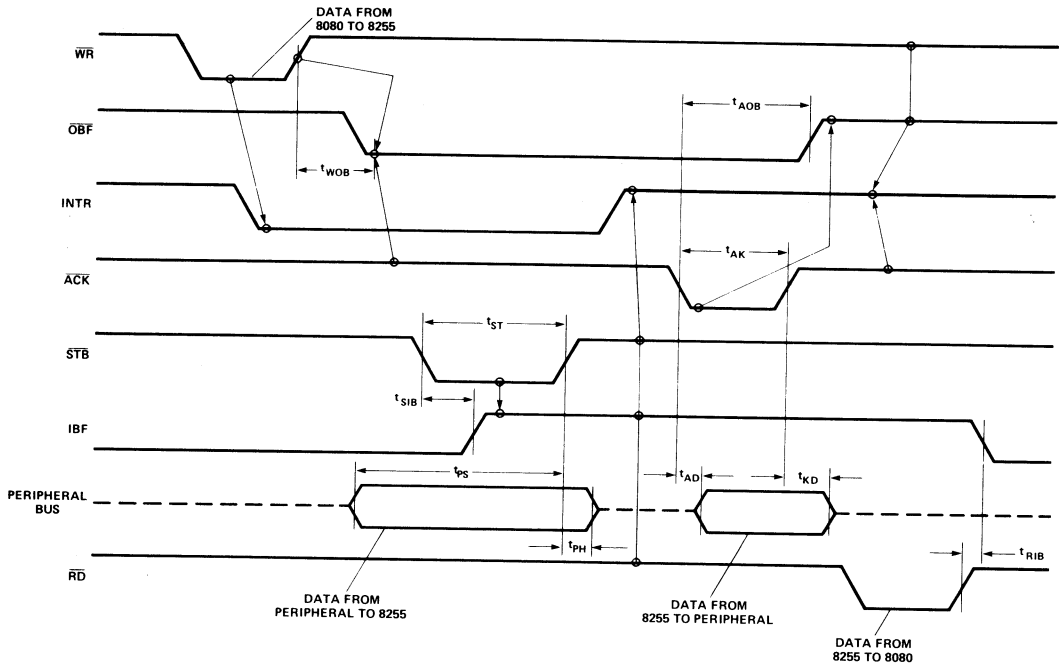


Figure 30. MODE 2 (Bidirectional)

NOTE: Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$ and $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
(INTR = IBF • $\overline{MASK}$ • $\overline{STB}$ • $\overline{RD}$ + $\overline{OBF}$ • $\overline{MASK}$ • $\overline{ACK}$ • $\overline{WR}$)

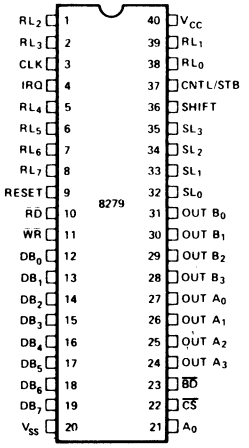
8279/8279-5
PROGRAMMABLE KEYBOARD/DISPLAY INTERFACE

- **MCS-85™ Compatible 8279-5**
- **Simultaneous Keyboard Display Operations**
- **Scanned Keyboard Mode**
- **Scanned Sensor Mode**
- **Strobed Input Entry Mode**
- **8-Character Keyboard FIFO**
- **2-Key Lockout or N-Key Rollover with Contact Debounce**
- **Dual 8- or 16-Numerical Display**
- **Single 16-Character Display**
- **Right or Left Entry 16-Byte Display RAM**
- **Mode Programmable from CPU**
- **Programmable Scan Timing**
- **Interrupt Output on Key Entry**

The Intel® 8279 is a general purpose programmable keyboard and display I/O interface device designed for use with Intel® microprocessors. The keyboard portion can provide a scanned interface to a 64-contact key matrix. The keyboard portion will also interface to an array of sensors or a strobed interface keyboard, such as the hall effect and ferrite variety. Key depressions can be 2-key lockout or N-key rollover. Keyboard entries are debounced and strobed in an 8-character FIFO. If more than 8 characters are entered, overrun status is set. Key entries set the interrupt output line to the CPU.

The display portion provides a scanned display interface for LED, incandescent, and other popular display technologies. Both numeric and alphanumeric segment displays may be used as well as simple indicators. The 8279 has 16X8 display RAM which can be organized into dual 16X4. The RAM can be loaded or interrogated by the CPU. Both right entry, calculator and left entry typewriter display formats are possible. Both read and write of the display RAM can be done with auto-increment of the display RAM address.

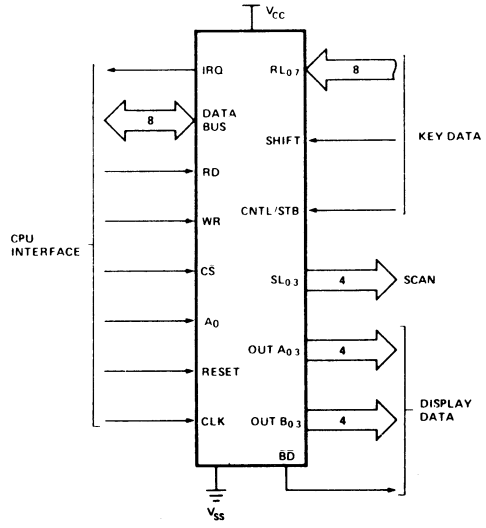
PIN CONFIGURATION



PIN NAMES

DB_{0-7}	I/O	DATA BUS (BI DIRECTIONAL)
CLK	I	CLOCK INPUT
RESET	I	RESET INPUT
$\overline{CS}$	I	CHIP SELECT
$\overline{RD}$	I	READ INPUT
$\overline{WR}$	I	WRITE INPUT
A_0	I	BUFFER ADDRESS
IRQ	O	INTERRUPT REQUEST OUTPUT
SL_{0-3}	O	SCAN LINES
RL_{0-7}	I	RETURN LINES
SHIFT	I	SHIFT INPUT
CNTL/STB	I	CONTROL/STROBE INPUT
$OUT A_{0-3}$	O	DISPLAY (A) OUTPUTS
$OUT B_{0-3}$	O	DISPLAY (B) OUTPUTS
$\overline{BD}$	O	BLANK DISPLAY OUTPUT

LOGIC SYMBOL



FUNCTIONAL DESCRIPTION

Since data input and display are an integral part of many microprocessor designs, the system designer needs an interface that can control these functions without placing a large load on the CPU. The 8279 provides this function for 8-bit microprocessors.

The 8279 has two sections: keyboard and display. The keyboard section can interface to regular typewriter style keyboards or random toggle or thumb switches. The display section drives alphanumeric displays or a bank of indicator lights. Thus the CPU is relieved from scanning the keyboard or refreshing the display.

The 8279 is designed to directly connect to the microprocessor bus. The CPU can program all operating modes for the 8279. These modes include:

Input Modes

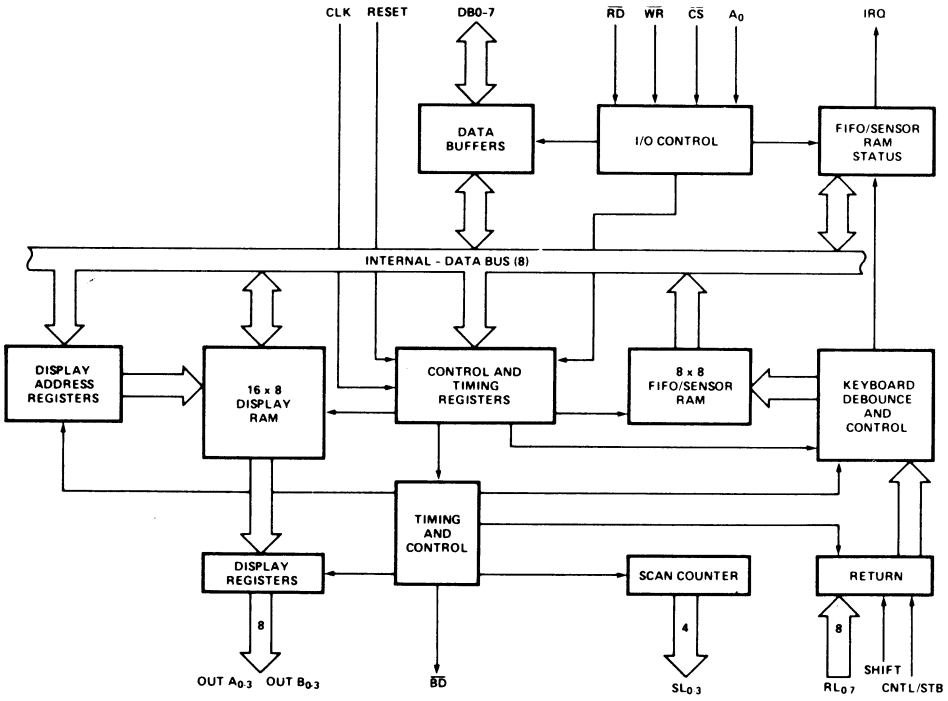
- Scanned Keyboard — with encoded (8 x 8 x 4 key keyboard) or decoded (4 x 8 x 4 key keyboard) scan lines. A key depression generates a 6-bit encoding of key position. Position and shift and control status are stored in the FIFO. Keys are automatically debounced with 2-key lockout or N-key rollover.
- Scanned Sensor Matrix — with encoded (8 x 8 matrix switches) or decoded (4 x 8 matrix switches) scan lines. Key status (open or closed) stored in RAM addressable by CPU.
- Strobed Input -- Data on return lines during control line strobe is transferred to FIFO.

Output Modes

- 8 or 16 character multiplexed displays that can be organized as dual 4-bit or single 8-bit.
- Right entry or left entry display formats.

Other features of the 8279 include:

- Mode programming from the CPU.
- Programmable clock to match the 8279 scan times to the CPU cycle time.
- Interrupt output to signal CPU when there is keyboard or sensor data available.
- An 8 byte FIFO to store keyboard information.
- 16 byte internal Display RAM for display refresh. This RAM can also be read by the CPU.



HARDWARE DESCRIPTION

The 8279 is packaged in a 40 pin DIP. The following is a functional description of each pin.

No. Of Pins	Designation	Function
8	DB_0-DB_7	Bi-directional data bus. All data and commands between the CPU and the 8279 are transmitted on these lines.
1	CLK	Clock from system used to generate internal timing.
1	RESET	A high signal on this pin resets the 8279.
1	$\overline{CS}$	Chip Select. A low on this pin enables the interface functions to receive or transmit.
1	A_0	Buffer Address. A high on this line indicates the signals in or out are interpreted as a command or status. A low indicates that they are data.
2	$\overline{RD}$, $\overline{WR}$	Input/Output read and write. These signals enable the data buffers to either send data to the external bus or receive it from the external bus.
1	IRQ	Interrupt Request. In a keyboard mode, the interrupt line is high when there is data in the FIFO/Sensor RAM. The interrupt line goes low with each FIFO/Sensor RAM read and returns high if there is still information in the RAM. In a sensor mode, the interrupt line goes high whenever a change in a sensor is detected.
2	V_{SS}, V_{CC}	Ground and power supply pins.
4	SL_0-SL_3	Scan Lines which are used to scan the key switch or sensor matrix and the display digits. These lines can be either encoded (1 of 16) or decoded (1 of 4).
8	RL_0-RL_7	Return line inputs which are connected to the scan lines through the keys or sensor switches. They have active internal pullups to keep them high until a switch closure pulls one low. They also serve as an 8-bit input in the Strobed Input mode.
1	SHIFT	The shift input status is stored along with the key position on key closure in the Scanned Keyboard modes. It has an active internal pullup to keep it high until a switch closure pulls it low.
1	CNTL/STB	For keyboard modes this line is used as a control input and stored like status on a key closure. The line is also the strobe line that enters the data into the FIFO in the Strobed Input mode. (Rising Edge). It has an active internal pullup to keep it high until a switch closure pulls it low.
4 4	OUT A_0-OUT A_3 OUT B_0-OUT B_3	These two ports are the outputs for the 16 x 4 display refresh registers. The data from these outputs is synchronized to the scan lines (SL_0-SL_3) for multiplexed digit displays. The two 4 bit ports may be blanked independently. These two ports may also be considered as one 8 bit port.
1	$\overline{BD}$	Blank Display. This output is used to blank the display during digit switching or by a display blanking command.

PRINCIPLES OF OPERATION

The following is a description of the major elements of the 8279 Programmable Keyboard/Display interface device. Refer to the block diagram in Figure 1.

I/O Control and Data Buffers

The I/O control section uses the $\overline{CS}$, A_0, $\overline{RD}$ and $\overline{WR}$ lines to control data flow to and from the various internal registers and buffers. All data flow to and from the 8279 is enabled by $\overline{CS}$. The character of the information, given or desired by the CPU, is identified by A_0. A logic one means the information is a command or status. A logic zero means the information is data. $\overline{RD}$ and $\overline{WR}$ determine the direction of data flow through the Data Buffers. The Data Buffers are bi-directional buffers that connect the internal bus to the external bus. When the chip is not selected ($\overline{CS}$ = 1), the devices are in a high impedance state. The drivers input during $\overline{WR} \bullet \overline{CS}$ and output during $\overline{RD} \bullet \overline{CS}$.

Control and Timing Registers and Timing Control

These registers store the keyboard and display modes and other operating conditions programmed by the CPU. The modes are programmed by presenting the proper command on the data lines with A_0 = 1 and then sending a $\overline{WR}$. The command is latched on the rising edge of $\overline{WR}$.

The command is then decoded and the appropriate function is set. The timing control contains the basic timing counter chain. The first counter is a ÷ N prescaler that can be programmed to match the CPU cycle time to the internal timing. The prescaler is software programmed to a value between 2 and 31. A value which yields an internal frequency of 100 kHz gives a 5.1 ms keyboard scan time and a 10.3 ms debounce time. The other counters divide down the basic internal frequency to provide the proper key scan, row scan, keyboard matrix scan, and display scan times.

Scan Counter

The scan counter has two modes. In the encoded mode, the counter provides a binary count that must be externally decoded to provide the scan lines for the keyboard and display. In the decoded mode, the scan counter decodes the least significant 2 bits and provides a decoded 1 of 4 scan. Note than when the keyboard is in decoded scan, so is the display. This means that only the first 4 characters in the Display RAM are displayed.

In the encoded mode, the scan lines are active high outputs. In the decoded mode, the scan lines are active low outputs.

Return Buffers and Keyboard Debounce and Control

The 8 return lines are buffered and latched by the Return Buffers. In the keyboard mode, these lines are scanned, looking for key closures in that row. If the debounce circuit detects a closed switch, it waits about 10 msec to check if the switch remains closed. If it does, the address of the switch in the matrix plus the status of SHIFT and CONTROL are transferred to the FIFO. In the scanned Sensor Matrix modes, the contents of the return lines is directly transferred to the corresponding row of the Sensor RAM (FIFO) each key scan time. In Strobed Input mode, the contents of the return lines are transferred to the FIFO on the rising edge of the CNTL/STB line pulse.

FIFO/Sensor RAM and Status

This block is a dual function 8 x 8 RAM. In Keyboard or Strobed Input modes, it is a FIFO. Each new entry is written into successive RAM positions and each is then read in order of entry. FIFO status keeps track of the number of characters in the FIFO and whether it is full or empty. Too many reads or writes will be recognized as an error. The status can be read by an $\overline{RD}$ with $\overline{CS}$ low and A_0 high. The status logic also provides an IRQ signal when the FIFO is not empty. In Scanned Sensor Matrix mode, the memory is a Sensor RAM. Each row of the Sensor RAM is loaded with the status of the corresponding row of sensor in the sensor matrix. In this mode, IRQ is high if a change in a sensor is detected.

Display Address Registers and Display RAM

The Display Address Registers hold the address of the word currently being written or read by the CPU and the two 4-bit nibbles being displayed. The read/write addresses are programmed by CPU command. They also can be set to auto increment after each read or write. The Display RAM can be directly read by the CPU after the correct mode and address is set. The addresses for the A and B nibbles are automatically updated by the 8279 to match data entry by the CPU. The A and B nibbles can be entered independently or as one word, according to the mode that is set by the CPU. Data entry to the display can be set to either left or right entry. See Interface Considerations for details.

SOFTWARE OPERATION

8279 commands

The following commands program the 8279 operating modes. The commands are sent on the Data Bus with $\overline{CS}$ low and A_0 high and are loaded to the 8279 on the rising edge of $\overline{WR}$.

Keyboard/Display Mode Set

	MSB							LSB
Code:	0	0	0	D	D	K	K	K

Where DD is the Display Mode and KKK is the Keyboard Mode.

DD

0	0	8 8-bit character display — Left entry
0	1	16 8-bit character display — Left entry*
1	0	8 8-bit character display — Right entry
1	1	16 8-bit character display — Right entry

For description of right and left entry, see Interface Considerations. Note that when decoded scan is set in keyboard mode, the display is reduced to 4 characters independent of display mode set.

KKK

0	0	0	Encoded Scan Keyboard — 2 Key Lockout
0	0	1	Decoded Scan Keyboard — 2-Key Lockout
0	1	0	Encoded Scan Keyboard — N-Key Rollover
0	1	1	Decoded Scan Keyboard — N-Key Rollover
1	0	0	Encoded Scan Sensor Matrix
1	0	1	Decoded Scan Sensor Matrix
1	1	0	Strobed Input, Encoded Display Scan
1	1	1	Strobed Input, Decoded Display Scan

Program Clock

Code:	0	0	1	P	P	P	P	P

Where PPPPP is the prescaler value 2 to 31. The programmable prescaler divides the external clock by PPPPP to get the basic internal frequency. Choosing a divisor that yields 100 KHz will give the specified scan and debounce times. Default after a reset pulse (but not a program clear) is 31.

Read FIFO/Sensor RAM

Code:	0	1	0	AI	X	A	A	A	X = Don't Care

Where AI is the Auto-Increment flag for the Sensor RAM and AAA is the row that is going to be read by the CPU. AI and AAA are used only if the mode is set to Sensor Matrix. This command is used to specify that the source of data reads ($\overline{CS}$ • RD • $\overline{A_0}$) by the CPU is the FIFO/Sensor RAM. No additional commands are necessary as long as

Default after reset.

data is desired from the FIFO/Sensor RAM. Another command is necessary if reading is desired from a different row than has been selected. If AI is a one, the row select counter will be incremented after each read so the next read will be from the next Sensor RAM row.

In the Auto Increment mode for reading data from the FIFO/Sensor RAM, each read advances the address by one so that the next read is from the next character. This Auto Incrementing has no effect on the display.

Read Display RAM

Code:

0	1	1	AI	A	A	A	A

Where AI is the Auto-Increment flag for the Display RAM and AAAA is the character that the CPU is going to read next. Since the CPU uses the same counter for reading and writing, this command also sets the next write location and Auto-Increment mode. This command is used to specify the display RAM as the data source for CPU data reads. If AI is set, the character address will be incremented after each read (or write) so that the next read (or write) will be from (to) the next character.

Write Display RAM

Code:

1	0	0	AI	A	A	A	A

Where AI is the Auto-Increment flag for the Display RAM and AAAA is the character that the CPU is going to write next. The addressing and Auto-Increment are identical to Read Display RAM. The difference is that Write Display RAM does not affect the source of CPU reads. The CPU will read from whichever RAM (Display or FIFO/Sensor) was last specified. This command will, however, change the location the next Display RAM read will be from if that source was specified.

Display Write Inhibit/Blanking

Code:

1	0	1	X	IW	IW	BL	BL
				A	B	A	B

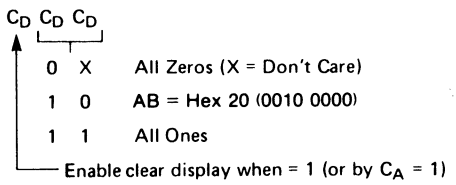
Where IW is Inhibit Writing (nibble A or B) and BL is Blanking (nibble A or B). If the display is being used as a dual 4-bit display, then it is necessary to mask one of the 4-bit halves so that entries to the Display from the CPU do not affect the other half. The IW flags allow the programmer to do this. It is also useful to be able to blank either half when that half is not to be displayed. The BL flags blank the display. The next command sets the output code to be used as a "blank". Default after reset is all zeros. Note that to blank a display formatted as a single 8-bit output, it is necessary to set both BL flags to entirely blank the display. A "1" sets the flag. Reissuing the command with a "0" resets the flag.

Clear

Code:

1	1	0	C_D	C_D	C_D	C_F	C_A

Where C_D is Clear Display, C_F is Clear FIFO Status (including interrupt), and C_A is Clear All. C_D is used to clear all positions of the Display RAM to a programmable code. All ones, all zeros and hexadecimal 20 are possible. The 2 least significant bits of C_D are also used to specify the blanking code (see below).



Clearing the display takes approximately 160 μs. During this time the CPU cannot write to the Display RAM. The MSB of the FIFO status word will be set during this time. C_F set the FIFO status to empty and resets the interrupt output line. After execution of a clear command with C_F set, the Sensor Matrix mode RAM pointer will be set to row 0.

C_A has the combined effect of C_D and C_F. C_A uses the C_D clearing code to determine how to clear the Display RAM. C_A also resets the internal timing chain to resynchronize it.

End Interrupt/Error Mode Set

Code:

1	1	1	E	X	X	X	X

X = Don't care.

For the sensor matrix modes this command lowers the IRQ line and enables further writing into RAM. (The IRQ line would have been raised upon the detection of a change in a sensor value. This would have also inhibited further writing into the RAM until reset).

For the N-key rollover mode — if the E bit is programmed to "1" the chip will operate in the special Error mode. (For further details, see Interface Considerations Section.)

Status Word

The status word contains the FIFO status, error, and display unavailable signals. This word is read by the CPU when A_0 is high and $\overline{CS}$ and $\overline{RD}$ are low. See Interface Considerations for more detail on status word.

Data Read

Data is read when A_0, $\overline{CS}$ and $\overline{RD}$ are all low. The source of the data is specified by the Read FIFO or Read Display commands. The trailing edge of $\overline{RD}$ will cause the address of the RAM being read to be incremented if the Auto-Increment flag is set. FIFO reads always increment (if no error occurs) independent of AI.

Data Write

Data that is written with A_0, $\overline{CS}$ and $\overline{WR}$ low is always written to the Display RAM. The address is specified by the latest Read Display or Write Display command. Auto-Incrementing on the rising edge of $\overline{WR}$ occurs if AI set by the latest display command.

INTERFACE CONSIDERATIONS

Scanned Keyboard Mode, 2-Key Lockout

There are three possible combinations of conditions that can occur during debounce scanning. When a key is depressed, the debounce logic is set. Other depressed keys are looked for during the next two scans. If none are encountered, it is a single key depression and the key position is entered into the FIFO along with the status of CNTL and SHIFT lines. If the FIFO was empty, IRQ will be set to signal the CPU that there is an entry in the FIFO. If the FIFO was full, the key will not be entered and the error flag will be set. If another closed switch is encountered, no entry to the FIFO can occur. If all other keys are released before this one, then it will be entered to the FIFO. If this key is released before any other, it will be entirely ignored. A key is entered to the FIFO only once per depression, no matter how many keys were pressed along with it or in what order they were released. If two keys are depressed within the debounce cycle, it is a simultaneous depression. Neither key will be recognized until one key remains depressed alone. The last key will be treated as a single key depression.

Scanned Keyboard Mode, N-Key Rollover

With N-key Rollover each key depression is treated independently from all others. When a key is depressed, the debounce circuit waits 2 keyboard scans and then checks to see if the key is still down. If it is, the key is entered into the FIFO. Any number of keys can be depressed and another can be recognized and entered into the FIFO. If a simultaneous depression occurs, the keys are recognized and entered according to the order the keyboard scan found them.

Scanned Keyboard — Special Error Modes

For N-key rollover mode the user can program a special error mode. This is done by the "End Interrupt/Error Mode Set" command. The debounce cycle and key-validity check are as in normal N-key mode. If during a single debounce cycle, two keys are found depressed, this is considered a simultaneous multiple depression, and sets an error flag. This flag will prevent any further writing into the FIFO and will set interrupt (if not yet set). The error flag could be read in this mode by reading the FIFO STATUS word. (See "FIFO STATUS" for further details.) The error flag is reset by sending the normal CLEAR command with $C_F = 1$.

Sensor Matrix Mode

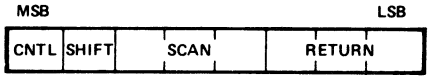
In Sensor Matrix mode, the debounce logic is inhibited. The status of the sensor switch is inputted directly to the Sensor RAM. In this way the Sensor RAM keeps an image of the state of the switches in the sensor matrix. Although debouncing is not provided, this mode has the advantage that the CPU knows how long the sensor was closed and when it was released. A keyboard mode can only indicate a validated closure. To make the software easier, the designer should functionally group the sensors by row since this is the format in which the CPU will read them.

The IRQ line goes high if any sensor value change is detected at the end of a sensor matrix scan. The IRQ line is cleared by the first data read operation if the Auto-Increment flag is set to zero, or by the End Interrupt command if the Auto-Increment flag is set to one.

Note: Multiple changes in the matrix Addressed by ($SL_{0-3} = 0$) may cause multiple interrupts. ($SL_0 = 0$ in the Decoded Mode). Reset may cause the 8279 to see multiple changes.

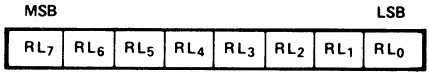
Data Format

In the Scanned Keyboard mode, the character entered into the FIFO corresponds to the position of the switch in the keyboard plus the status of the CNTL and SHIFT lines (non-inverted). CNTL is the MSB of the character and SHIFT is the next most significant bit. The next three bits are from the scan counter and indicate the row the key was found in. The last three bits are from the column counter and indicate to which return line the key was connected.

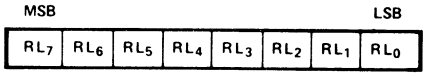


SCANNED KEYBOARD DATA FORMAT

In Sensor Matrix mode, the data on the return lines is entered directly in the row of the Sensor RAM that corresponds to the row in the matrix being scanned. Therefore, each switch postion maps directly to a Sensor RAM position. The SHIFT and CNTL inputs are ignored in this mode. Note that switches are not necessarily the only thing that can be connected to the return lines in this mode. Any logic that can be triggered by the scan lines can enter data to the return line inputs. Eight multiplexed input ports could be tied to the return lines and scanned by the 8279.



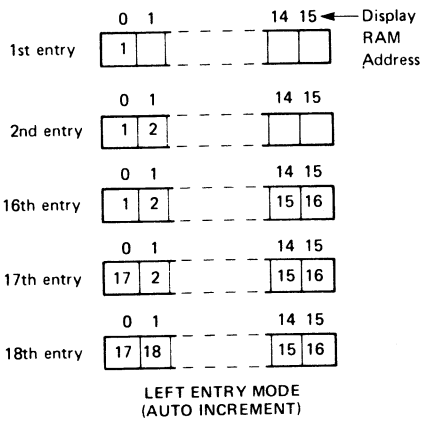
In Strobed Input mode, the data is also entered to the FIFO from the return lines. The data is entered by the rising edge of a CNTL/STB line pulse. Data can come from another encoded keyboard or simple switch matrix. The return lines can also be used as a general purpose strobed input.



Display

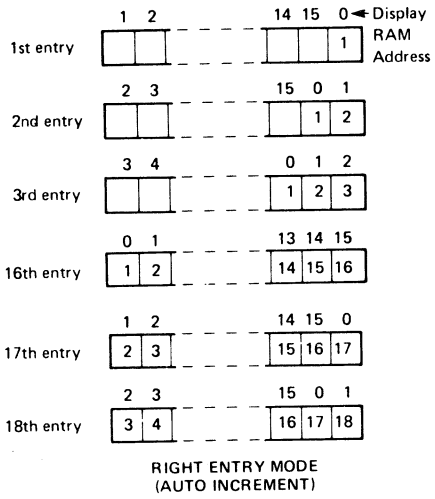
Left Entry

Left Entry mode is the simplest display format in that each display position directly corresponds to a byte (or nibble) in the Display RAM. Address 0 in the RAM is the left-most display character and address 15 (or address 7 in 8 character display) is the right most display character. Entering characters from position zero causes the display to fill from the left. The 17th (9th) character is entered back in the left most position and filling again proceeds from there.



Right Entry

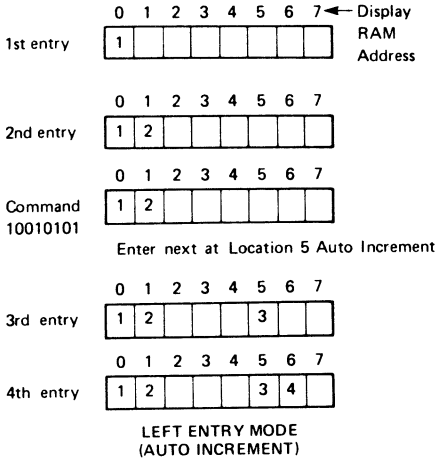
Right entry is the method used by most electronic calculators. The first entry is placed in the right most display character. The next entry is also placed in the right most character after the display is shifted left one character. The left most character is shifted off the end and is lost.



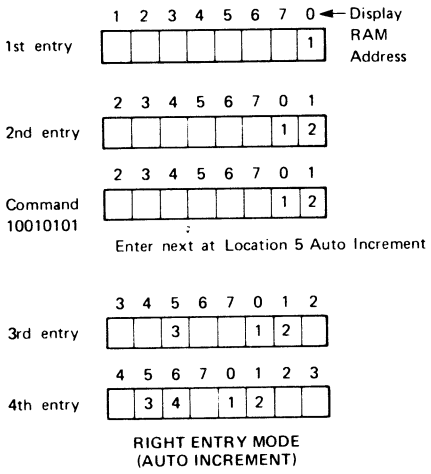
Note that now the display position and register address do not correspond. Consequently, entering a character to an arbitrary position in the Auto Increment mode may have unexpected results. Entry starting at Display RAM address 0 with sequential entry is recommended.

Auto Increment

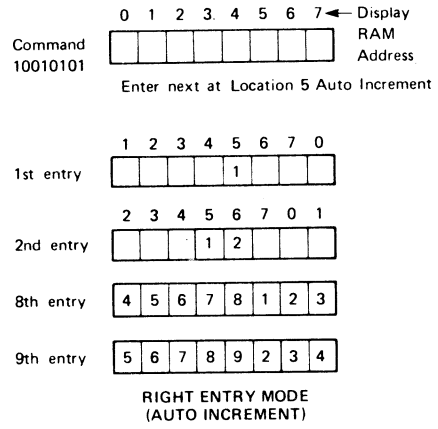
In the Left Entry mode, Auto Incrementing causes the address where the CPU will next write to be incremented by one and the character appears in the next location. With non-Auto Incrementing the entry is both to the same RAM address and display position. Entry to an arbitrary address in the Auto Increment mode has no undesirable side effects and the result is predictable:



In the Right Entry mode, Auto Incrementing and non Incrementing have the same effect as in the Left Entry except if the address sequence is interrupted:



Starting at an arbitrary location operates as shown below:



Entry appears to be from the initial entry point.

8/16 Character Display Formats

If the display mode is set to an 8 character display, the on duty-cycle is double what it would be for a 16 character display (e.g., 5.1 ms scan time for 8 characters vs. 10.3 ms for 16 characters with 100 kHz internal frequency).

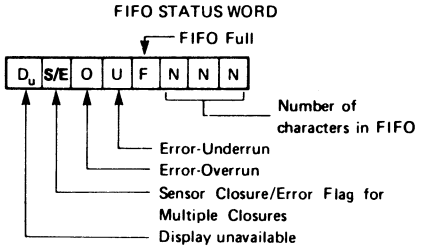
G. FIFO Status

FIFO status is used in the Keyboard and Strobed Input modes to indicate the number of characters in the FIFO and to indicate whether an error has occurred. There are two types of errors possible: overrun and underrun. Overrun occurs when the entry of another character into a full FIFO is attempted. Underrun occurs when the CPU tries to read an empty FIFO.

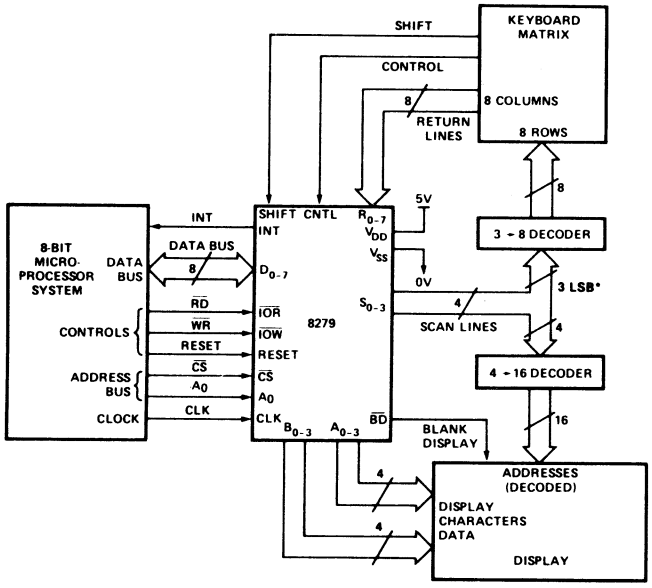
The FIFO status word also has a bit to indicate that the Display RAM was unavailable because a Clear Display or Clear All command had not completed its clearing operation.

In a Sensor Matrix mode, a bit is set in the FIFO status word to indicate that at least one sensor closure indication is contained in the Sensor RAM.

In Special Error Mode the S/E bit is showing the error flag and serves as an indication to whether a simultaneous multiple closure error has occurred.



APPLICATIONS



*Do not drive the keyboard decoder with the MSB of the scan lines.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature . . . . . 0°C to 70°C
Storage Temperature . . . . . -65°C to 125°C
Voltage on any Pin with
Respect to Ground . . . . . -0.5V to +7V
Power Dissipation . . . . . 1 Watt

**COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{SS} = 0V, Note 1

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL1}	Input Low Voltage for Return Lines	-0.5	1.4	V	
V_{IL2}	Input Low Voltage for All Others	-0.5	0.8	V	
V_{IH1}	Input High Voltage for Return Lines	2.2		V	
V_{IH2}	Input High Voltage for All Others	2.0		V	
V_{OL}	Output Low Voltage		0.45	V	Note 2
V_{OH}	Output High Voltage on Interrupt Line	3.5		V	Note 3
I_{IL1}	Input Current on Shift, Control and Return Lines		+10 -100	μA μA	$V_{IN} = V_{CC}$ $V_{IN} = 0V$
I_{IL2}	Input Leakage Current on All Others		±10	μA	$V_{IN} = V_{CC}$ to 0V
I_{OFL}	Output Float Leakage		±10	μA	$V_{OUT} = V_{CC}$ to 0V
I_{CC}	Power Supply Current		120	mA	

Notes:
1. 8279, V_{CC} = +5V ±5%; 8279-5, V_{CC} = +5V ±10%.
2. 8279, I_{OL} = 1.6mA; 8279-5, I_{OL} = 2.2mA.
3. 8279, I_{OH} = -100μA; 8279-5, I_{OH} = -400μA.

CAPACITANCE

SYMBOL	TEST	TYP.	MAX.	UNIT	TEST CONDITIONS
C_{in}	Input Capacitance	5	10	pF	$V_{in}=V_{CC}$
C_{out}	Output Capacitance	10	20	pF	$V_{out}=V_{CC}$

A.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{SS} = 0V, (Note 1)

Bus Parameters

Read Cycle:

Symbol	Parameter	8279 Min.	8279 Max.	8279-5 Min.	8279-5 Max.	Unit
t_{AR}	Address Stable Before $\overline{READ}$	50		0		ns
t_{RA}	Address Hold Time for $\overline{READ}$	5		0		ns
t_{RR}	$\overline{READ}$ Pulse Width	420		250		ns
t_{RD}[2]	Data Delay from $\overline{READ}$		300		150	ns
t_{AD}[2]	Address to Data Valid		450		250	ns
t_{DF}	$\overline{READ}$ to Data Floating	10	100	10	100	ns
t_{RCY}	Read Cycle Time	1		1		μs

Write Cycle:

Symbol	Parameter	8279 Min.	8279 Max.	8279-5 Min.	8279-5 Max.	Unit
t_{AW}	Address Stable Before $\overline{WRITE}$	50		0		ns
t_{WA}	Address Hold Time for $\overline{WRITE}$	20		0		ns
t_{WW}	$\overline{WRITE}$ Pulse Width	400		250		ns
t_{DW}	Data Set Up Time for $\overline{WRITE}$	300		150		ns
t_{WD}	Data Hold Time for $\overline{WRITE}$	40		0		ns

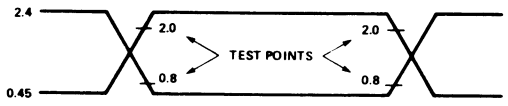
Notes:
1. 8279, V_{CC} = +5V ±5%; 8279-5, V_{CC} = +5V ±10%.
2. 8279, C_L = 100pF; 8279-5, C_L = 150pF.

Other Timings:

Symbol	Parameter	8279 Min.	8279 Max.	8279-5 Min.	8279-5 Max.	Unit
$t_{\phi W}$	Clock Pulse Width	230		120		nsec
t_{CY}	Clock Period	500		320		nsec

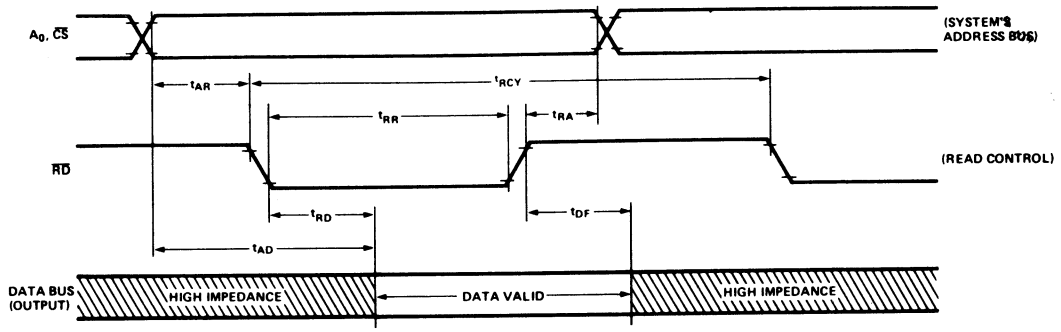
Keyboard Scan Time:	5.1 msec	Digit-on Time:	480 μsec
Keyboard Debounce Time:	10.3 msec	Blanking Time:	160 μsec
Key Scan Time:	80 μsec	Internal Clock Cycle:	10 μsec
Display Scan Time:	10.3 msec		

Input Waveforms For A.C. Tests

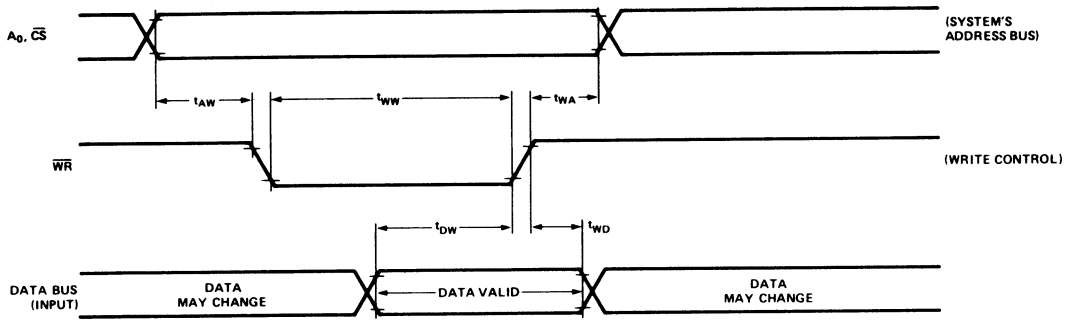


WAVEFORMS

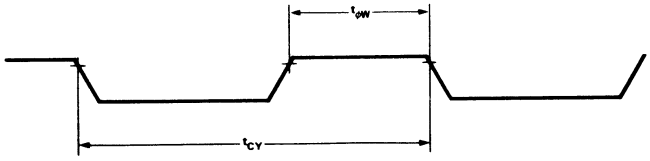
Read Operation



Write Operation



Clock Input



APPENDIX C
ROM/EPROM MEMORY EXPANSION

C-1. Introduction

This appendix examines the interfacing of additional read-only memory to either the user design area or off-board. The memory devices cited in this appendix are members of the Intel 5 volt ROM and EPROM family and are interfaced to the SDK-86 through the bus expansion logic. More information on this ROM/EPROM family is available in *Application of Intel's 5V EPROM and ROM Family for Microprocessor Systems,* Application Note AP-30.

C-2. ROM/EPROM Family Characteristics

The current and future generations of Intel 5 volt ROMs and EPROMs operate from a single 5 volt source and feature 2-line control (separate Chip Enable and Output Enable inputs) to eliminate bus contention problems in multiple memory systems. While a majority of the currently available devices are contained in 24-pin packages, the 2364 ROM and next generation devices, because of their increased density, require a 28-pin package. Even though the number of pins on the package is increased, compatibility within the entire family is maintained by keeping the functions on the lower 24 pins consistent between the two package sizes as shown in Figure C-1.

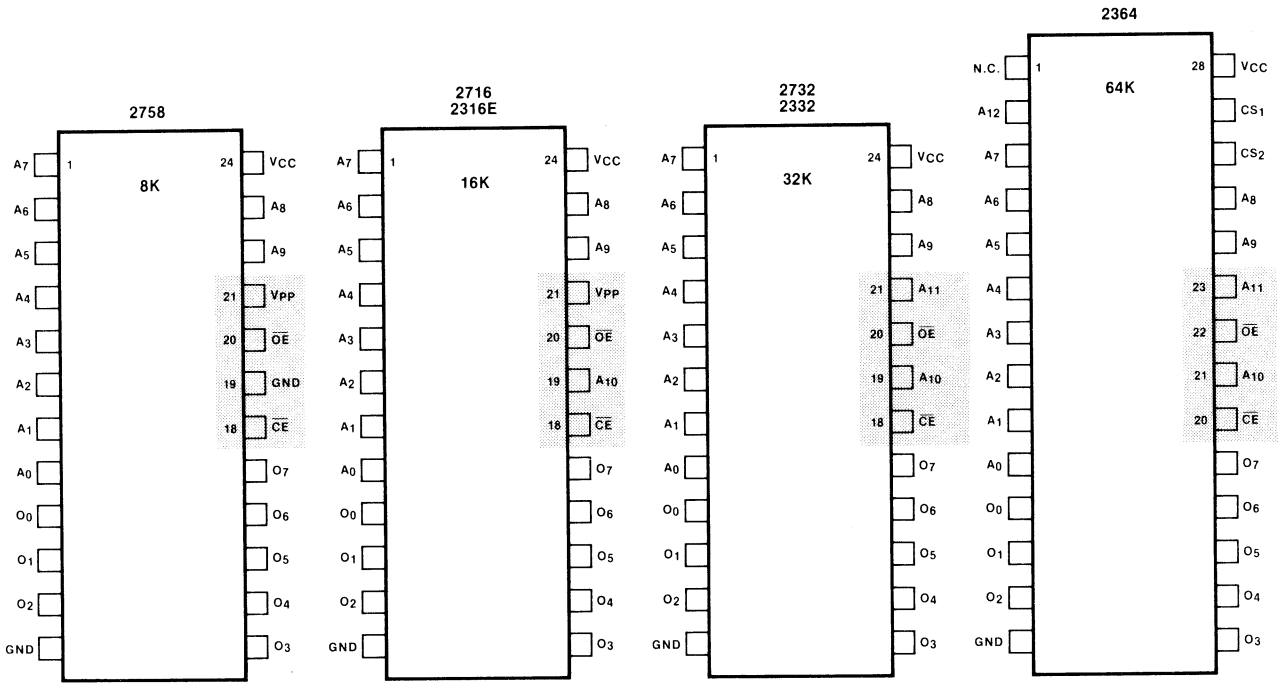


Figure C-1. 5 Volt ROM/EPROM Compatible Family

The following table defines the Intel 5 volt ROM/EPROM family.

Table C-1. Intel 5V ROM/EPROM Family

Device	Device Type	Memory Capacity	Memory Organization	Package Size
2758	EPROM	8K	1K x 8	24-pin
2316E	ROM	16K	2K x 8	24-pin
2616	PROM	16K	2K x 8	24-pin
2716	EPROM	16K	2K x 8	24-pin
2332	ROM	32K	4K x 8	24-pin
2732	EPROM	32K	4K x 8	24-pin
2364	ROM	64K	8K x 8	28-pin

C-3. 24-Pin and 28-Pin Sites

As mentioned in the previous section, Intel ROMs and EPROMs, depending on the bit density, are available in either a 24- or 28-pin package. To ensure future upgrading, all designs currently using 24-pin devices should be planned for 28-pin sites. In this way, when a higher-density device of 28 pins is used, the required circuit modifications are limited to only the proper connection of the next high-order address bit(s).

C-4. 8086 Compatability

Since the 8086 CPU is capable of 16-bit word memory access, ROM and EPROM devices are used in pairs. One device is connected to the low-order data byte (D0-D7), and the other device is connected to the high-order data byte (D8-D15). By offsetting the address bit inputs to the two devices (e.g., A1 address bit connected to the A0 address input of each device) and by using the A0 address bit as a decode input in conjunction with the $\overline{BHE}$ (Byte High Enable) signal, either the low- or high-order byte or both bytes can be accessed. Table C-2 defines the A0/$\overline{BHE}$ truth table used by the SDK-86 for byte and word memory access.

Table C-2. Byte Decoding

Decode Input		Byte Accessed
A0	$\overline{BHE}$	
0	0	Both Bytes (D0-D15)
1	0	High Byte (D8-D15)
0	1	Low Byte (D0-D7)
1	1	None

C-5. Design Example 1

The following design example (Figure C-2) shows the logic design for the addition of 16, 32 or 64K ROM/EPROM devices in the unassigned SDK-86 memory area immediately below FC000H. In the example, 28-pin sites are used to allow compatibility with a 64K device (2364).

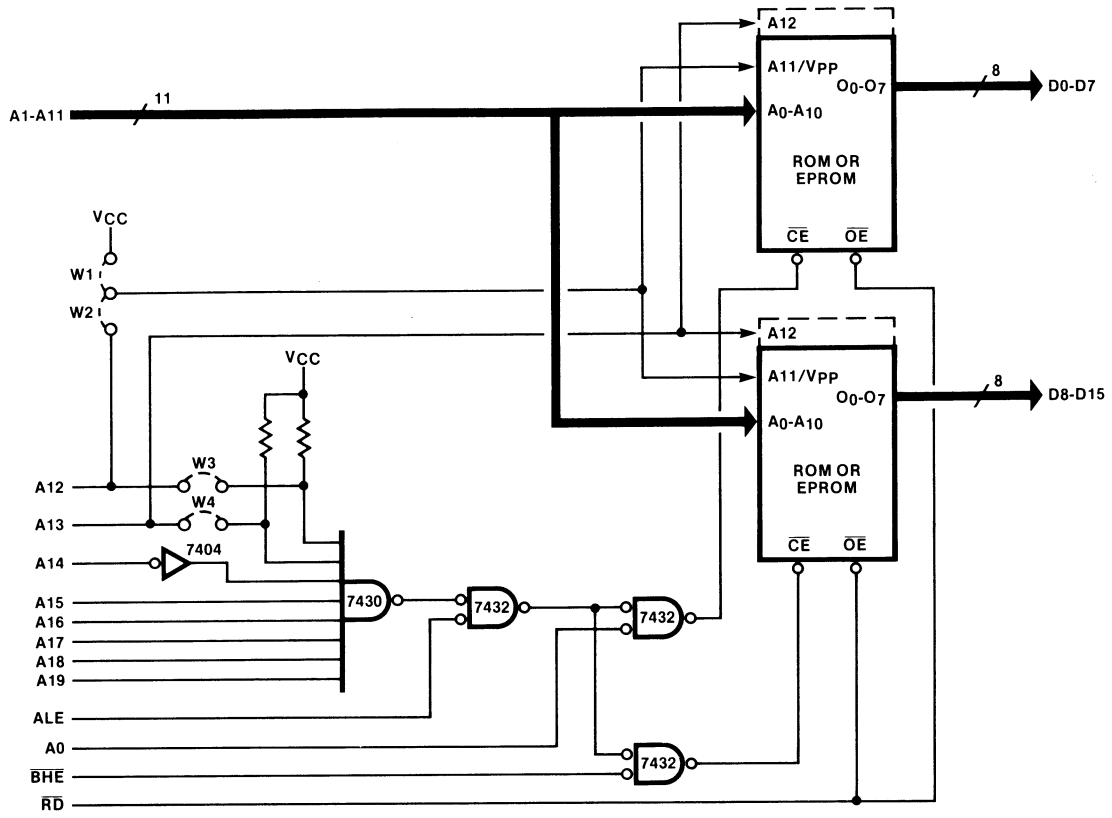


Figure C-2. Design Example 1

In the example, the ALE (Address Latch Enable) signal is gated with the enabled address output from the 7430 for compatibility with edge-enabled devices and is not required for the currently available devices in the family. Table C-3 defines the jumper configurations for 16, 32 and 64K devices. Refer to Table 6-4 for expansion bus interface pin assignments.

Table C-3. Jumper Configurations

Device	Jumpers Installed	Memory Block Selected
2316E/2716	W1, W3, W4	FB000H-FBFFFH
2332/2732	W2, W4	FA000H-FBFFFH
2364	W2	F8000H-FBFFFH

C-6. Design Example 2

The following design example (Figure C-3) shows a logic design which uses the two bus expansion address block decode signals ($\overline{CSX}$ and $\overline{CSY}$) generated by the SDK-86. Since the combined address capability of these two signals is 8K bytes (addresses FC000H-FDFFFH), the maximum memory device that can be used is a 2332/2732, 4K × 8 ROM/EPROM, and 28-pin sites consequently are not required.

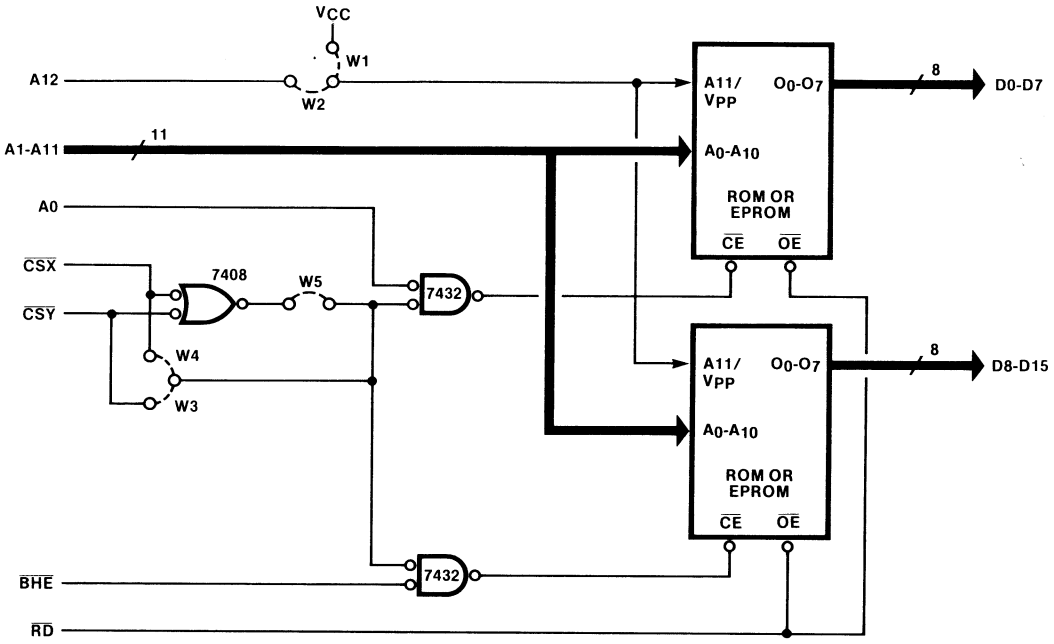


Figure C-3. Design Example 2

Since the existing 16K and 32K devices are static, the ALE signal used in design example 1 is not used. The following table defines the jumper configurations for a 16K or 32K device and the corresponding memory block accessed.

Table C-4. Jumper Configurations

Device	Jumpers Installed	Memory Block Selected
2316E/2716	W1, W4	FD000H-FDFFFH
	W1, W3	FC000H-FCFFFH
2332/2732	W2, W5	FC000H-FDFFFH

REQUEST FOR READER'S COMMENTS

The Microcomputer Division Technical Publications Department attempts to provide documents that meet the needs of all Intel product users. This form lets you participate directly in the documentation process.

Please restrict your comments to the usability, accuracy, readability, organization, and completeness of this document.

1. Please specify by page any errors you found in this manual.

2. Does the document cover the information you expected or required? Please make suggestions for improvement.

3. Is this the right type of document for your needs? Is it at the right level? What other types of documents are needed?

4. Did you have any difficulty understanding descriptions or wording? Where?

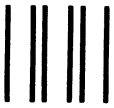
5. Please rate this document on a scale of 1 to 10 with 10 being the best rating. _____

NAME _____ DATE _____
TITLE _____
COMPANY NAME/DEPARTMENT _____
ADDRESS _____
CITY _____ STATE _____ ZIP CODE _____

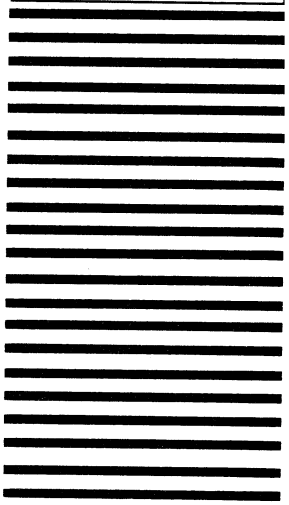
Please check here if you require a written reply. ☐

WE'D LIKE YOUR COMMENTS . . .

This document is one of a series describing Intel products. Your comments on the back of this form will help us produce better manuals. Each reply will be carefully reviewed by the responsible person. All comments and suggestions become the property of Intel Corporation.



NO POSTAGE
NECESSARY IF MAILED
IN THE U.S.



BUSINESS REPLY CARD
FIRST CLASS PERMIT NO. 621, SANTA CLARA, CA

Postage will be paid by Addressee:

Intel Corporation
Attn: Literature Department
3065 Bowers Avenue
Santa Clara, California 95051

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, California 95051 (408) **987-8080**

Printed in U.S.A.