HITACHI®

HD641016 USER'S MANUAL

#U15

HITACHI®
HD641016 USER'S MANUAL
#U15

HD641016 USER'S MANUAL

#U15

HITACHI®

ADE-602-003A

Section 1. Contents

Section 2. Overview

2.1 Description

The HD641016 is a high-density integrated 16-bit microprocessor (MPU) with a high-speed CPU (10-MHz clock), 1-kbyte RAM, DMA controller, timer asynchronous serial communications interface (ASCI), interrupt controller, peripheral controller, and memory access control devices on a single chip. It is based on advanced CMOS manufacturing and microcomputer technology.

The on-chip high-speed RAM can function as both high speed data RAM and global register banks for high-speed task switching. The HD641016 has three low power consumption modes.

The HD641016 is useful in general control system applications requiring high performance and low cost.

2.2 Features

2.2.1 CPU Features

- Register Configuration :
  - — General-purpose registers : 2, 4, 8, or 16 banks of 16 32-bit registers
  - — Control registers : 7 32- bit, 2 16-bit, and 3 8-bit control registers
- Bank Mode : Global and ring modes
- Instruction Set :
  - — Source and destination operands can be independently addressed
  - — Byte, word, and long word operand sizes provided
  - — Byte-basis instruction code for efficiency
- 16-Mbyte linear address space

2.2.2 On-Chip Hardware Features

- RAM:
  - — 1024 bytes
  - — Functions as high-speed data RAM or register banks
  - — Relocatable in memory space, controlled by the RAM base register
- DMA Controller:
  - — Four channels
  - — Memory to/from memory, I/O, internal ASCI, or internal timer DMA transfers
- 16-bit Timers:
  - — Two channels
  - — Function as interval timer and perform PWM, two-phase output, one-shot pulse output, event count, pulse width measurement, and frequency measurement
- ASCI:
  - — Two channels
  - — Asynchronous or clock synchronous mode
  - — Band rate generator for each channel incorporated
- Interrupt Controller:
  - — Three external interrupt sources ($\overline{NMI}$, $\overline{IRQ_0}$, $\overline{IRQ_1}$) (See Figure 2-1 and Table 3-1.)
  - — 22 internal interrupt sources
  - — External vector or autovector
  - — Vector Table relocatable by the exception vector base register
- Peripheral Controller:
  - — Switches multiplexed pin function
- Memory Access Support:
  - — Dynamic RAM refresh controller
  - — Wait state controller
  - — Chip select controller

2.2.3 Other Features

- E clock output for 6800-family bus interface
- Timing generator
- Low power consumption modes

2.2.4 HD641016 Family

Table 2-1. HD641016 Types

Type Name	Operating Frequency (MHz)	Package
HD641016 CP8	8.0	84-pin PLCC (CP-84)
HD641016 CP10	10.0	
HD641016 CP12*	12.5	
HD641016 Y8*	8.0	135-pin ceramic PGA (PC-135)
HD641016 Y10*	10.0	
HD641016 Y12*	12.5	
HD641016 YP8*	8.0	135-pin plastic PGA (PP-135)
HD641016YP10*	10.0	
HD641016 YP12*	12.5	

* Under development

2.3 Block Diagram

Figure 2-1 is a block diagram of the HD641016.

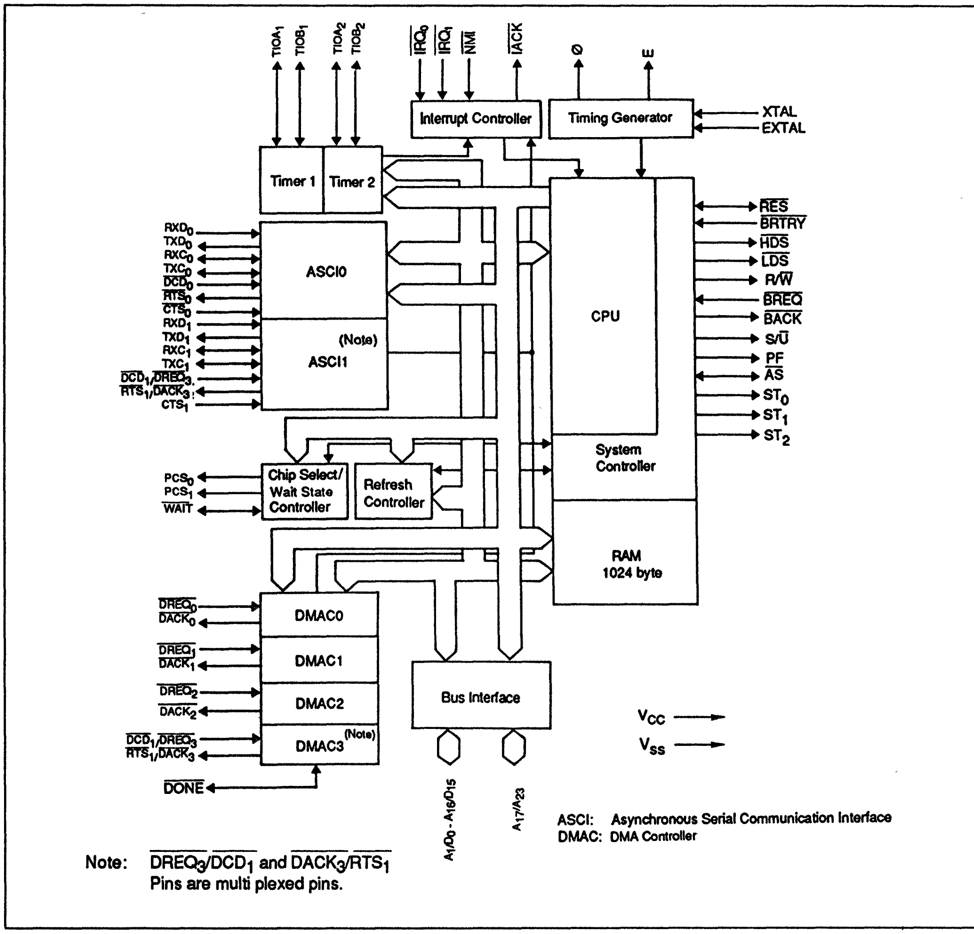


Figure 2-1. HD641016 Block Diagram

Section 3. Pin Description

Figures 3-1 and 3-2 show HD641016 CP-84 and PC/PP-135 pin configurations. Table 3-1 describes the pins of the HD641016.

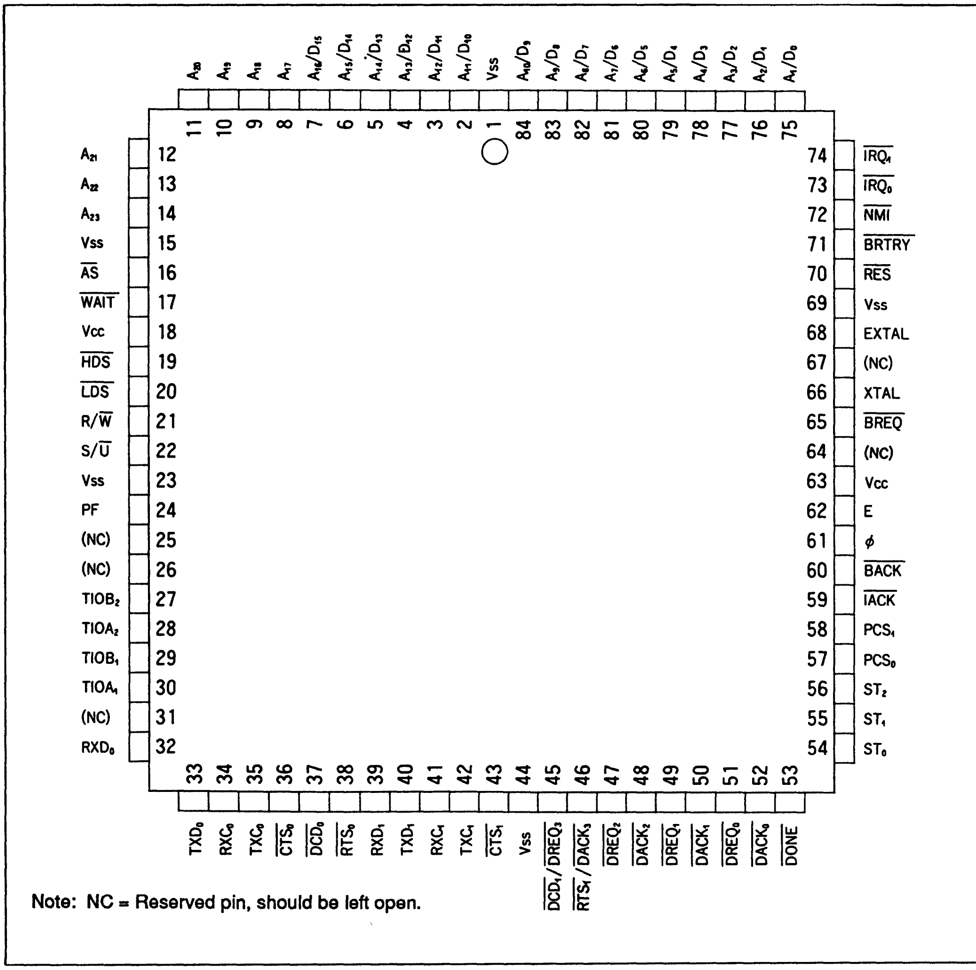


Figure 3-1. CP-84 Pin Arrangement (Top View)

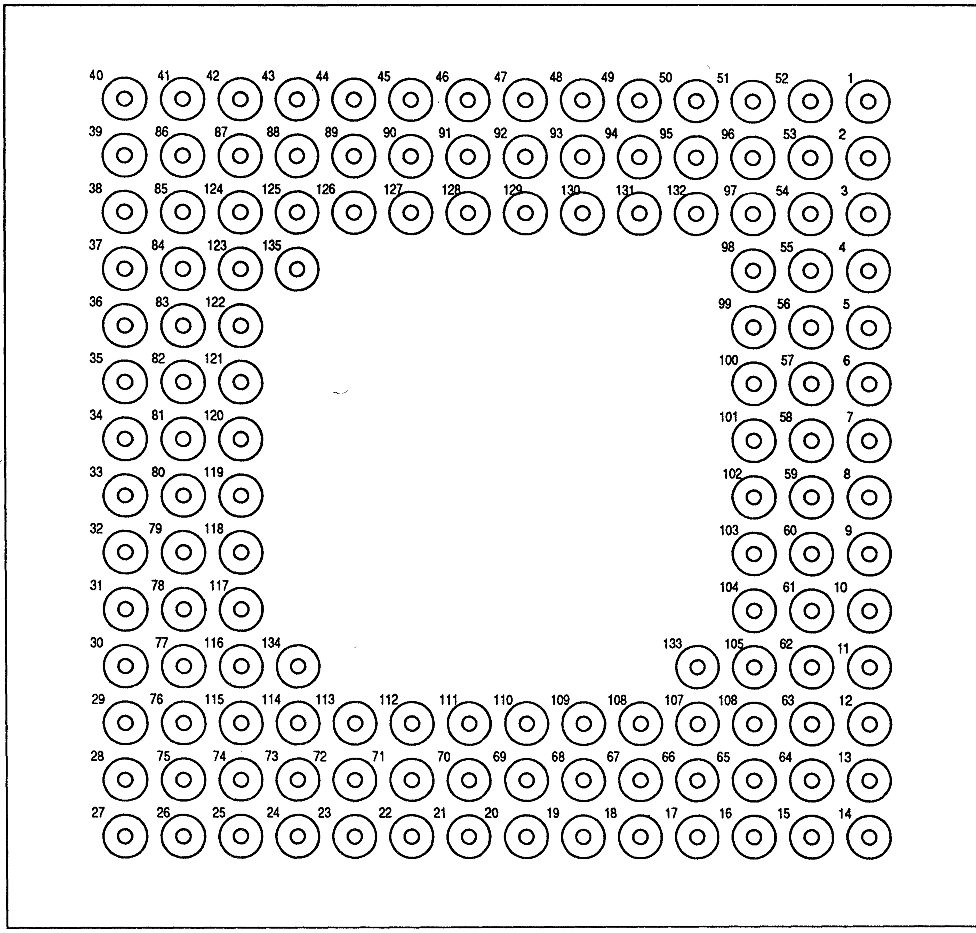


Figure 3-2. PC/PP-135 Pin Arrangement (Bottom View)

Table 3-1. Pin Description

Symbol	Pin Number		I/O	Function
	CP-84	PC/PP-135		
V_{CC}	18, 63	100, 119	I	Power supply
V_{SS}	1, 15, 23, 44, 69	4, 101, 102, 131, 69, 91, 82	I	Ground
XTAL	66	120	I	Crystal connection
EXTAL	68	34	I	Crystal or external clock connection
Ø	61	79	O	System clock
E	62	32	O	Enable clock
$\overline{RES}$	70	121	I/O (Open-drain)	Reset
$\overline{BRTRY}$	71	36	I	Bus cycle retry
$\overline{BREQ}$	65	33	I	Bus request
$\overline{BACK}$	60	118	O	Bus request acknowledge
$\overline{NMI}$	72	83	I	Nonmaskable interrupt request
$\overline{IRQ_0}$, $\overline{IRQ_1}$	73, 74	37, 84	I	Interrupt requests 0, 1
$\overline{IACK}$	59	31	O (Three-state)	Interrupt acknowledge
$\overline{HDS}$	19	57	O	High-order data strobe
$\overline{LDS}$	20	6	O	Low-order data strobe
R/$\overline{W}$	21	7	O	Read/write
$\overline{WAIT}$	17	56	I/O (Three-state)	Wait
$\overline{AS}$	16	99	I/O (Three-state)	Address strobe
ST_0-ST_2	54-56	29, 116, 77	O (Three-state)	Status

Table 3-1. Pin Description (cont.)

Symbol	Pin Number CP-84	Pin Number PC/PP-135	I/O	Function
$S/\overline{U}$	22	58	O (Three-state)	Supervisor/user
PF	24	59	O (Three-state)	Program fetch
A_1/D_0- A_{16}/D_{15}	75-84, 2-7	41, 42, 126, 88, 43 127, 89, 44, 90, 128, 46, 47, 92, 48, 49, 129	I/O (Three-state)	Multiplexed address/data bus
A_{17}-A_{23}	8-14	93,50,94,130,3, 98,55	I/O (Three-state)	Address bus, bits 17-23
$\overline{DREQ_0}$- $\overline{DREQ_3}$	51, 49, 47, 45	24, 23, 111, 21	I	DMA request for channels 0, 1, 2, and 3
$\overline{DACK_0}$- $\overline{DACK_3}$	52, 50, 48 46	73, 72, 71, 22	O	DMA acknowledge for channels 0, 1, 2, and 3
$\overline{DONE}$	53	113	I/O (Open-drain)	DMA done
$TIOA_1$, $TIOB_1$	30, 29	104, 61	I/O	Timer 1 input/outputs A, B
$TIOA_2$, $TIOB_2$	28, 27	10, 103	I/O	Timer 2 input/outputs A, B
RXD_0, RXD_1	32, 39	62, 17	I	Receive data channels 0, 1
TXD_0, TXD_1	33, 40	15, 68	O	Transmit data channels 0, 1
RXC_0, RXC_1	34, 41	108, 110	I/O	Receive clock channels 0, 1
TXC_0, TXC_1	35, 42	66, 18	I/O	Transmit clock channels 0, 1
$\overline{RTS_0}$, $\overline{RTS_1}$	38, 46	67, 22	O	Request to send channels 0, 1
$\overline{CTS_0}$, $\overline{CTS_1}$	36, 43	16, 19	I	Clear to send channels 0, 1
$\overline{DCD_0}$, $\overline{DCD_1}$	37, 45	109, 21	I	Data carrier detect channels 0, 1
PCS_0, PCS_1	57, 58	117, 78	O	Programmable chip selects 0, 1

3.1 Multiplexed Pins

The lower 16 bits of the address bus (A_1-A_{16}) are multiplexed with the data bus (D_0-D_{15}). $\overline{\mathrm{DCD}_1}/\overline{\mathrm{DREQ}_3}$ and $\overline{\mathrm{RTS}_1}/\overline{\mathrm{DACK}_3}$ are also multiplexed pins. The function of $\overline{\mathrm{DCD}_1}/\overline{\mathrm{DREQ}_3}$ is controlled by the PTF_1 bit of PCR_0. The function of $\overline{\mathrm{RTS}_1}/\overline{\mathrm{DACK}_3}$ is controlled by the PTF_0 bit of PCR_0. After reset, these pins function as $\overline{\mathrm{DCD}_1}$ and $\overline{\mathrm{RTS}_1}$, respectively.

3.2 Pin Function

3.2.1 Power Supply

V_{CC}: V_{CC} is the +5 V power supply pin.

V_{SS}: V_{SS} is the 0 V ground pin.

3.2.2 Clock Signals

Crystal, External Clock (XTAL, EXTAL): A crystal resonator can be connected between XTAL and EXTAL to supply the system clock. The crystal should have a resonant frequency of twice the system clock Ø. Ø max is 12.5 MHz.

Alternately, an external clock pulse at twice the system clock frequency can be input directly to EXTAL. When an external clock pulse is input at EXTAL, XTAL should be left floating (open). See Figure 3-3.

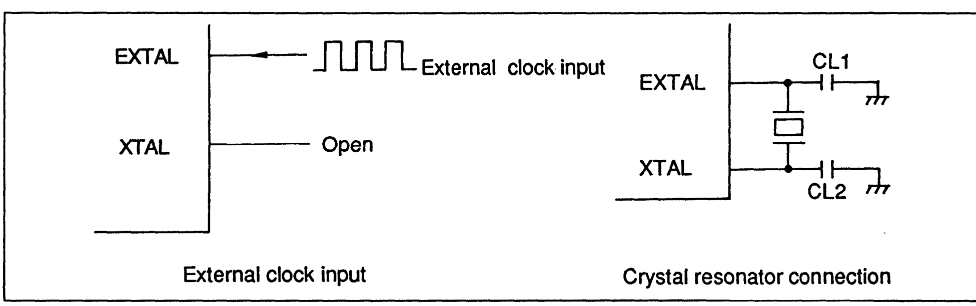


Figure 3-3. Resonator Circuit Example

System Clock (Ø): The Ø output supplies the system clock to peripheral devices.

Enable Clock (E): The E output provides a clock to 6800-family devices.

3.2.3 System Control

Reset ($\overline{RES}$): When $\overline{RES}$ and $\overline{BRTRY}$ are both pulled low, the MPU enters reset state processing. When either $\overline{RES}$ or $\overline{BRTRY}$ is pulled high after that, the CPU begins reset exception processing. All on-chip I/O functions are then initialized (power-on reset).

When $\overline{RES}$, $\overline{BRTRY}$, and $\overline{NMI}$ are asserted low, simultaneously the MPU enters reset state but the internal refresh controller and external bus master continue operating (manual reset).

When the MPU executes the RESET instruction, it pulls $\overline{RES}$ low to reset peripherals. $\overline{RES}$ is open drain.

Bus Cycle Retry ($\overline{BRTRY}$): $\overline{BRTRY}$ requests a restart of the bus cycle. If the same bus cycle is requested to be restarted 3 consecutive times while BRTE = 1, the MPU starts bus error exception processing. However, if BRTE = 0, the MPU immediately begins bus error exception processing if $\overline{BRTRY}$ is asserted low even once. If $\overline{BRTRY}$ is asserted in interrupt acknowledge cycle, the CPU performs acknowledge error exception processing without perform bus error exception processing. When $\overline{BRTRY}$ and $\overline{RES}$ are pulled low with $\overline{RES}$, it causes a reset exception. (Refer to "4.6.4 Exception Processing".)

Bus Request ($\overline{BREQ}$): The $\overline{BREQ}$ input requests that the HD641016 release the bus.

Bus Request Acknowledge ($\overline{BACK}$): The $\overline{BACK}$ output indicates that the HD641016 has released the bus.

3.2.4 Interrupt Control

Nonmaskable Interrupt ($\overline{NMI}$): The $\overline{NMI}$ is a non-maskable interrupt request input. It is also used with $\overline{RES}$ and $\overline{BRTRY}$ to perform a system reset with DRAM refresh. (Refer to "4.6.4 Exception Processing".)

Interrupt Requests 0, 1 ($\overline{IRQ_0}$, $\overline{IRQ_1}$): $\overline{IRQ_0}$ and $\overline{IRQ_1}$ are maskable external interrupt request inputs.

Interrupt Request Acknowledge ($\overline{IACK}$): The MPU outputs $\overline{IACK}$ to acknowledge receipt of an interrupt request from $\overline{IRQ_0}$ during the double-acknowledge cycle. This pin must not be pulled down.

3.2.5 Memory Interface

High-, Low-Order Data Strobe ($\overline{\text{HDS}}$, $\overline{\text{LDS}}$): The MPU outputs $\overline{\text{HDS}}$ to indicate that the upper byte of the data bus (D_{15}-D_8) is valid. The $\overline{\text{LDS}}$ output indicates that the lower byte of the data bus (D_7-D_0) is valid.

Read/Write Strobe (R/$\overline{\text{W}}$): The MPU outputs R/$\overline{\text{W}}$ high when using the bus for a read cycle, and low for a write cycle.

Wait ($\overline{\text{WAIT}}$): When the HD641016 is the bus master, an external device requests a Tw state insertion into the bus cycle by asserting the $\overline{\text{WAIT}}$ input. When another device is the bus master while WTOE = 1, the MPU outputs $\overline{\text{WAIT}}$ to request a Tw state insertion into the bus cycle of the external master.

Address Strobe ($\overline{\text{AS}}$): When the HD641016 is the bus master, it outputs $\overline{\text{AS}}$ to indicate that the address is valid. External circuits can latch multiplexed address information on the falling edge of $\overline{\text{AS}}$. When another device is the bus master, the $\overline{\text{AS}}$ input indicates that the address information from the external the bus master is valid.

3.2.6 Status

Status (ST_0-ST_2): The ST_0-ST_2 outputs indicate the MPU status as shown in Table 3-2.

Table 3-2. MPU Status Signals

ST_2	ST_1	ST_0	Status
0	0	0	Sleep or system stop
0	0	1	System halt
0	1	0	Bus locked
0	1	1	Normal
1	0	0	DMA single address
1	0	1	DMA dual address
1	1	0	Refresh
1	1	1	Interrupt acknowledge

Refer to "Appendix D Pin State" for details on priority and change timing at synchronization of these signals.

Supervisor/User ($S/\overline{U}$): $S/\overline{U}$ output high indicates supervisor address space is being accessed. $S/\overline{U}$ low indicates user address space is being accessed.

Program Fetch (PF): If the CPU is the bus master, PF output high indicates that the CPU is fetching a program instruction. If the internal DMAC is the bus master, PF output high indicates that program space is being accessed, while PF low indicates data space is being accessed.

3.2.7 Address and Data Bus

Address/Data Bus (A_1/D_0-A_{16}/D_{15}): When the HD641016 is the bus master, A_1/D_0-A_{16}/D_{15} output multiplexed address information and input/output data information. Otherwise they input address information.

Address Bus (A_{17}-A_{23}): When the HD641016 is the bus master, A_{17}-A_{23} output address information. Otherwise they input address information.

3.2.8 DMAC

DMA Requests, Channels 0-3 ($\overline{DREQ_0}$-$\overline{DREQ_3}$): $\overline{DREQ_0}$-$\overline{DREQ_3}$ inputs request DMA transfers for DMA channels 0-3 respectively.

DMA Acknowledge, Channels 0-3 ($\overline{DACK_0}$-$\overline{DACK_3}$): $\overline{DACK_0}$-$\overline{DACK_3}$ outputs indicate that a DMA request has been received by DMA channels 0-3 respectively.

DMA Done ($\overline{DONE}$): $\overline{DONE}$ output indicates that the current DMA operation is completed. As an input, it requests that the current DMA transfer be terminated.

3.2.9 Timer

Timers 1, 2 Input/Output A, B, Channels 1, 2 ($TIOA_1$, $TIOB_1$, $TIOA_2$, $TIOB_2$): $TIOA_1$, $TIOB_1$, $TIOA_2$, and $TIOB_2$ are external clock outputs, trigger inputs, or timer outputs for channels 1 and 2, depending on the timer mode.

3.2.10 Asynchronous Serial I/O (ASCI)

Receive, Transmit, Channels 0, 1 (RXD_0, RXD_1, TXD_0, TXD_1): RXD_0 and RXD_1 receive serial data for ASCI channels 0 and 1. TXD_0 and TXD_1 transmit serial data for ASCI channels 0 and 1.

Receive Clock, Transmit Clock, Channels 0, 1 (RXC_0, RXC_1, TXC_0, TXC_1): RXC_0 and RXC_1 input or output the receive clock for ASCI channels 0 and 1. TXC_0 and TXC_1 input or output the transmit clock for ASCI channels 0 and 1.

Request to Send, Channels 0, 1 ($\overline{RTS_0}$, $\overline{RTS_1}$): $\overline{RTS_0}$ and $\overline{RTS_1}$ are modem control outputs for ASCI channels 0 and 1. $\overline{RTS_0}$ and $\overline{RTS_1}$ are controlled by software.

Clear to Send, Channels 0, 1 ($\overline{CTS_0}$, $\overline{CTS_1}$): $\overline{CTS_0}$ and $\overline{CTS_1}$ are modem control inputs for ASCI channels 0 and 1. $\overline{CTS_0}$ and $\overline{CTS_1}$ control the ASCI transmission.

Data Carrier Detect, Channels 0, 1 ($\overline{DCD_0}$, $\overline{DCD_1}$): $\overline{DCD_0}$ and $\overline{DCD_1}$ are modem control inputs for ASCI channels 0 and 1. $\overline{DCD_0}$ and $\overline{DCD_1}$ can reset the ASCI receiver.

3.2.11 Chip Select

Programmable Chip Select 0, 1 (PCS_0, PCS_1): PCS_0 and PCS_1 decode address information and output a chip select signal.

Section 4. CPU

The CPU performs 32-bit data processing using a maximum of sixteen global banks of sixteen 32-bit general-purpose registers.

4.1 Programming Model

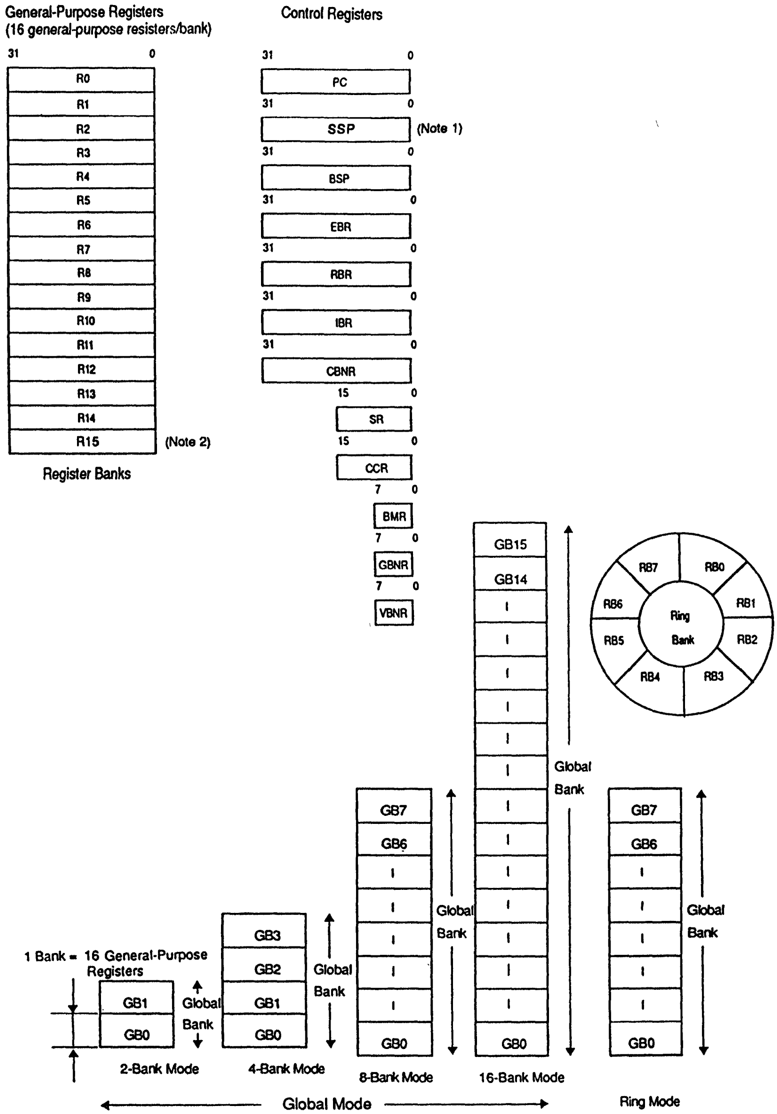
Figure 4-1 is the CPU register programming model. Each bank has sixteen 32-bit general-purpose registers (R0-R15) which can be used as byte (8-bit), word (16-bit), or long-word (32-bit) data registers, address pointers, or index registers. R15 can also be used as the user stack pointer (USP). The CPU can use 2, 4, 8, or 16 global banks, selected by the bank mode register (BMR). In addition, the CPU can use 8 global banks and a set of ring banks (8 banks) in ring mode. See "4.4 Register Banks" for details.

The CPU also has the following 12 control registers:

- Program counter, PC (32 bits)
- Supervisor stack pointer, SSP (32 bits)
- Bank stack pointer, BSP (32 bits)
- Exception vector base register, EBR (32 bits)
- RAM base register, RBR (32 bits)
- Internal I/O base register, IBR (32 bits)
- Current bank number register, CBNR (32 bits)
- Status register, SR (16 bits)
- Condition code register, CCR (16 bits)
- Bank mode register, BMR (8 bits)
- Global bank number register, GBNR (8 bits)
- Valid bank number register, VBNR (8 bits)

The HD641016 operates in either of two privilege modes, supervisor or user. The privilege mode is selected by the S bit in the status register. The privilege mode defines the available hardware resources.

In the programming model, the stack pointer differs according to the S bit. When S = 1, supervisor stack pointer (SSP) is selected as the stack pointer. When S = 0, the user stack pointer (USP), which is R15 of global bank, is selected as the stack pointer.



Notes: 1. If R15 or stack pointer is selected by an instruction when the S bit of the SR register is set to 1, the supervisor stack pointer (SSP) is used.

2. If the S bit of the SR register is cleared to 0, the user stack pointer (USP) which is R15 of general-purpose register is used.

Figure 4-1. Programming Model

4.2 Data Types

The CPU can support six data types: 1-bit data, binary-coded decimal (BCD) data, byte data (8 bits), word data (16 bits), long word data (32 bits), or bit field data. Bit manipulation instructions support 1-bit data. Bit field instructions support bit field data. 8-bit BCD arithmetic instructions usually operate on two BCD digits. Some instructions have implicit data types.

4.2.1 General-Purpose Register Data Organization

Figure 4-2 shows the six data types supported by the general-purpose registers. For 1-bit data, the LSB is addressed as bit 0 and the MSB is addressed as bit 31. Two BCD digits or one byte occupy the lower 8 bits of a general purpose register. A word occupies the lower 16 bits, and a long word occupies the entire register. In 1-bit, BCD, byte or word, and long word data types, the LSB is addressed as bit 0. For bit field data, the MSB is addressed as bit position 0 and the LSB is addressed as bit position 31. The upper bytes of the register are not affected by the BCD, byte, or word operation.

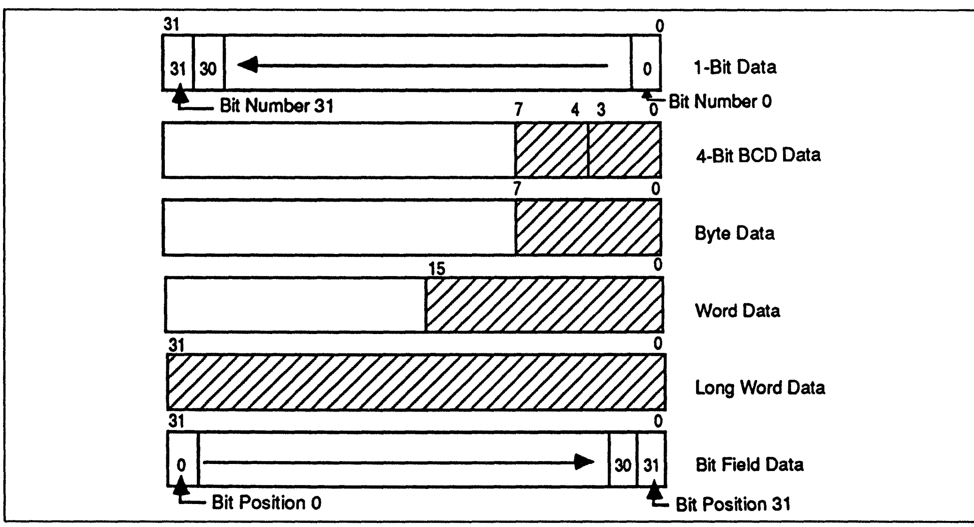


Figure 4-2. General-Purpose Register Data Organization

4.2.2 Memory Data Organization

The HD641016 has a 16-bit external data bus (D_{15}-D_0), and the memory is organized on a 16-bit basis. However, byte or long word data can be easily addressed. Figures 4-3 through 4-5 show byte, word, and long word data organization in memory.

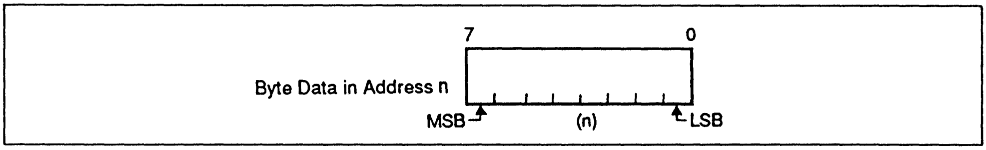


Figure 4-3. Byte Data Organization in Memory

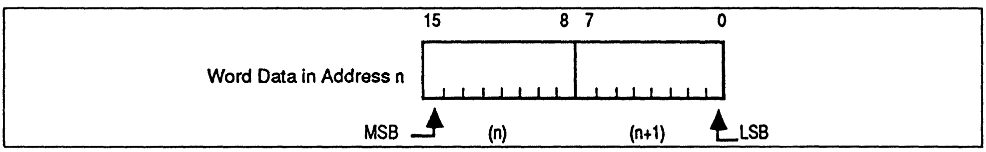


Figure 4-4. Word Data Organization in Memory

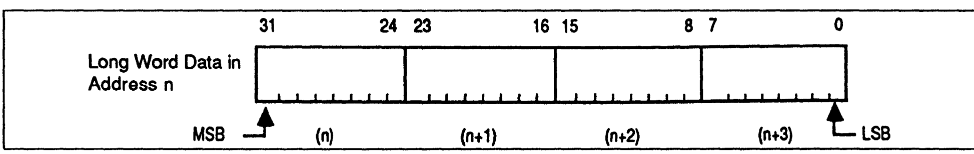


Figure 4-5. Long Word Data Organization in Memory

Two BCD digits can be addressed in the same way as byte data (Figure 4-6). The upper BCD digit is located at the upper four bits, and the lower digit is located at the lower 4 bits of address n.

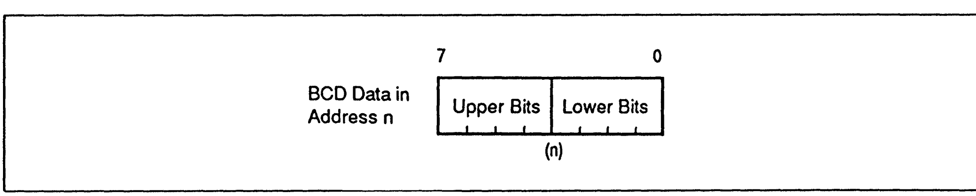


Figure 4-6. BCD Data Organization in Memory

Bit data is addressed by a byte, word, or long word address, and the number of the bit containing the data. The bit number of the LSB is 0, and the MSB is bit 7, 15, or 31, depending on the data type. Figure 4-7 shows bit numbers in a long word.

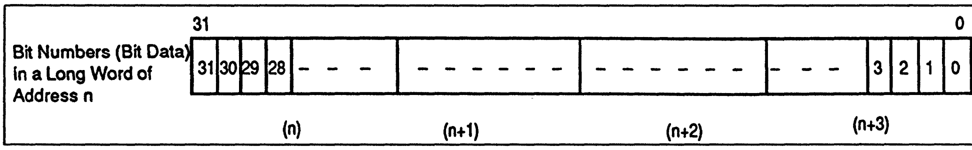


Figure 4-7. Bit Data in a Long Word in Memory

Bits in bit field data (Figure 4-8) are addressed in the opposite order. The MSB is bit 0.

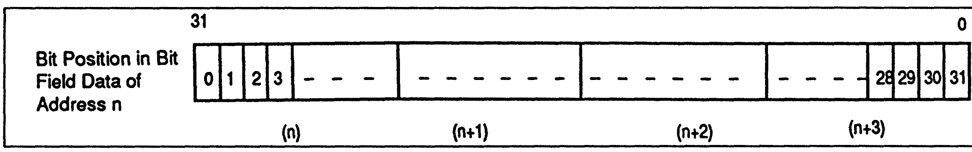


Figure 4-8. Bit Field Data in a Long Word in Memory

Word data is aligned on two-byte boundaries. For example, the upper byte of the word is located at byte address n, and the lower byte at byte address n + 1. If n is even, a word can be accessed in one bus cycle. If n is odd, word data access requires two bus cycles (two byte access cycles). A long word occupies 4 bytes. The upper and lower 16 bits of the long word at address n correspond to the words at word addresses n and n + 2, respectively. If n is even, a long word data can be accessed in two bus cycles. If n is odd, long word data access requires three bus cycles (one byte access, one word access, and another byte access cycle). Note that 8 bits of invalid data appears on the unused data bus lines during byte data writes or writes to odd addresses. See Figure 4-9.

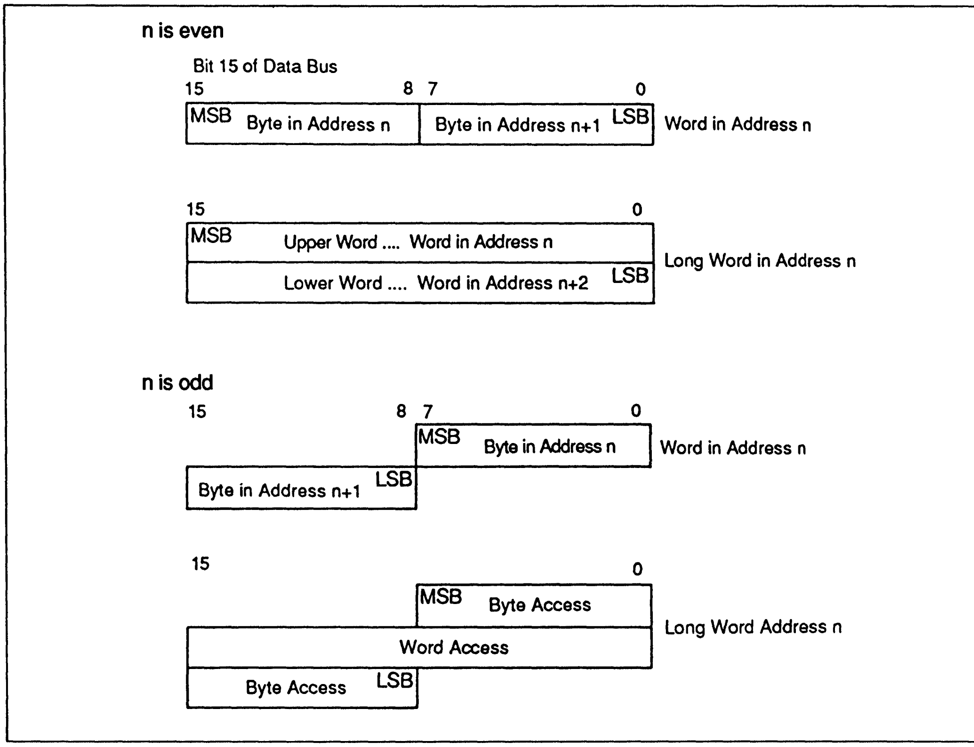


Figure 4-9. Word and Long Word Data Access at Address n

4.3 Registers

4.3.1 Program Counter (PC)

The 32-bit PC (Figure 4-10) indicates the address at which the instruction being executed is stored. It is incremented by one after every instruction byte. Note that the upper 8 bits of the PC are reserved and only bits 23-0 are valid.

A dedicated pointer, independent of the PC, prefetches instruction for the 8-bit prefetch FIFO. Refer to "4.10 Prefetch" for details.

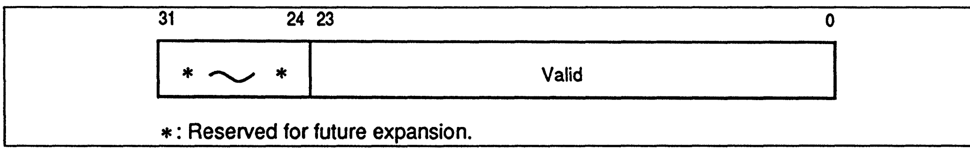


Figure 4-10. Program Counter (PC)

4.3.2 Status Register (SR)

Figure 4-11 shows the 16-bit status register.

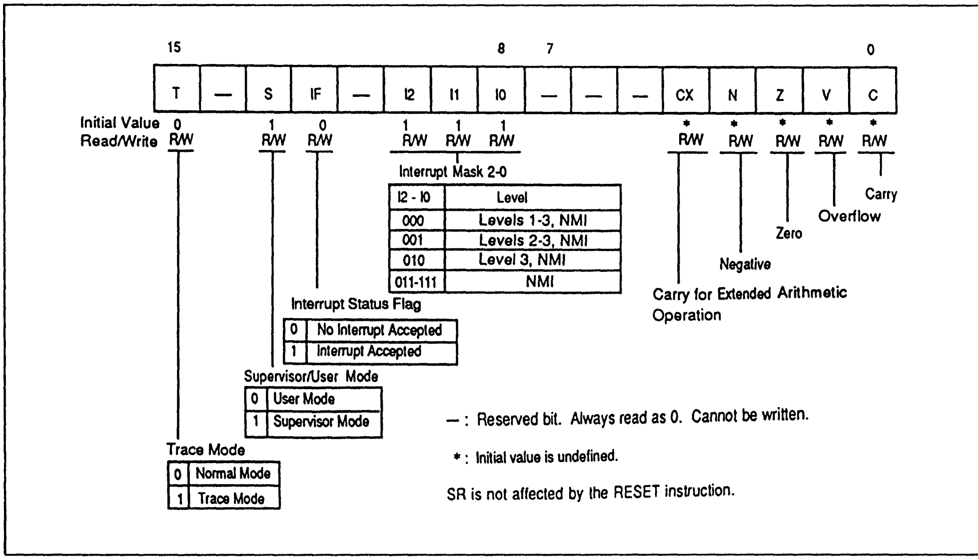


Figure 4-11. Status Register (SR)

The lower 8 bits of the SR register also functions as those of the CCR register. The SR register can only be accessed in the supervisor state. SR is not affected by the RESET instruction.

Note : "Reset" means "hardware reset by the $\overline{\text{RES}}$ pin" and "RESET instruction" means "software reset by the RESET instruction".

Trace Mode (T): When T = 0, instructions are executed normally. When T = 1, the CPU is in trace mode. In trace mode, trace exception processing is started after the execution of every instruction. It is cleared to 0 at reset.

Supervisor/User Mode (S): Bit S selects the privilege state. When S = 0, the MPU is in the user state. When S = 1, it is in the supervisor state. It is set to 1 at reset.

Interrupt Status Flag (IF): When IF = 0, no interrupts have been accepted. When IF = 1, an interrupt has been accepted. It is cleared to 0 at reset.

Interrupt Mask 2-0 (I2-I0): I2-I0 are used to inhibit interrupts. The HD641016 supports 4 interrupt levels: NMI, and maskable interrupt levels 1 through 3. All interrupts of level equal to or lower than the level set in I2-I0 cannot be accepted. Table 4-1 shows I2-I0 and the interrupts which can be accepted. I2-I0 are set to 1 at reset.

After the MPU receives an interrupt, I2-I0 are set to inhibit equal or lower level interrupts. Table 4-2 shows I2-I0 after an interrupt.

Table 4-1. I2-I0 and Acceptable Interrupt

I2	I1	I0	Interrupt Accepted
0	0	0	Levels 1-3, NMI
0	0	1	Levels 2-3, NMI
0	1	0	Levels 3, NMI
0	1	1	NMI
1	0	0	NMI
1	0	1	NMI
1	1	0	NMI
1	1	1	NMI

Table 4-2. Interrupt Masking After Interrupt

Interrupt Received	I2	I1	I0
Level 1	0	0	1
Level 2	0	1	0
Level 3	0	1	1
NMI	1	1	1

See the condition code register (CCR) for details on the CX, N, Z, V, and C bits.

4.3.3 Condition Code Register (CCR)

The condition code register CCR (Figure 4-12) stores the result of an operation. The CCR register is not affected by the RESET instruction.

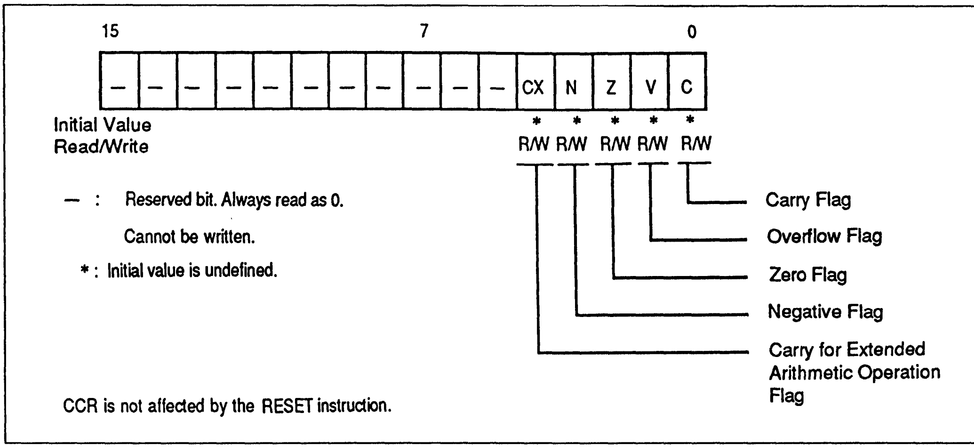


Figure 4-12. Condition Code Register (CCR)

Carry for Extended Arithmetic Operation Flag (CX): CX is the same as C after arithmetic or shift operations, but is unchanged after data transmission operations which affect C.

Negative Flag (N): N = 1 when the MSB of the result is 1. N = 0 when the MSB of the result is 0.

Zero Flag (Z): Z = 1 when the result equals 0. Otherwise it is 0.

Overflow Flag (V): V becomes 1 when an arithmetic operation causes an overflow. Otherwise it is 0.

Carry Flag (C): C becomes 1 when a carry or borrow from the MSB occurs. Otherwise it is 0.

See "Appendix B Condition Code Affected" for details.

4.3.4 Exception Vector Base Register (EBR)

The exception vector base register EBR (Figure 4-13) specifies the start address of the exception processing vector table in address space. It is discussed in "4.6 Processing States and Privilege Modes". The EBR register is not affected by the RESET instruction.

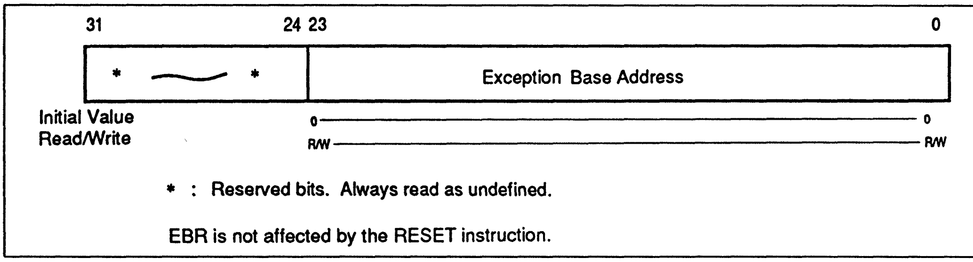


Figure 4-13. Exception Vector Base Register (EBR)

4.3.5 RAM Base Register (RBR)

The RAM base register RBR (Figure 4-14) specifies the start address of internal RAM . Bits 23-10 of RBR are valid. The RBR register can relocate the internal RAM in 1-kbyte units within a 16-Mbyte address space. RBR is not affected by the RESET instruction. See "Section 5. RAM" for details.

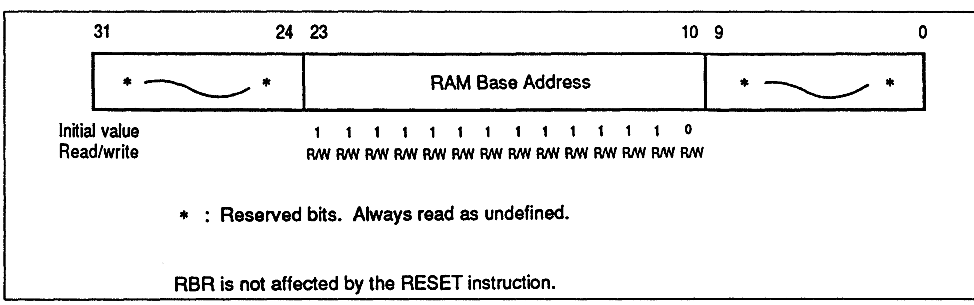


Figure 4-14. RAM Base Register (RBR)

4.3.6 I/O Base Register (IBR)

The I/O base register IBR specifies an internal I/O area. The physical addresses of the internal I/O registers are determined by bits 23-16 (IBR23-IBR16) of the IBR register and the address offset value of each internal I/O register (Figure 4-15). IBR is not affected by the RESET instruction. See "Section 6. Internal I/O" for details.

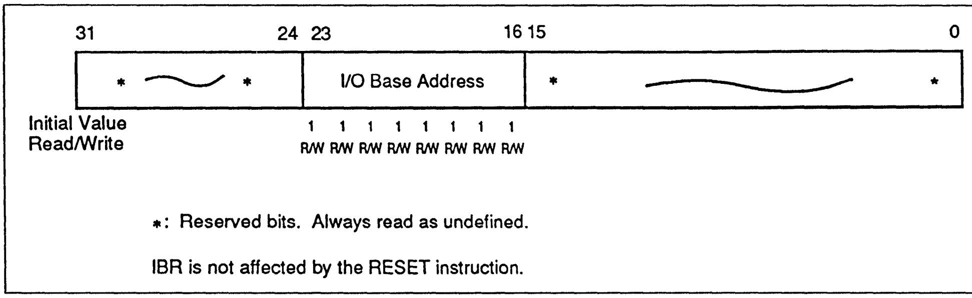
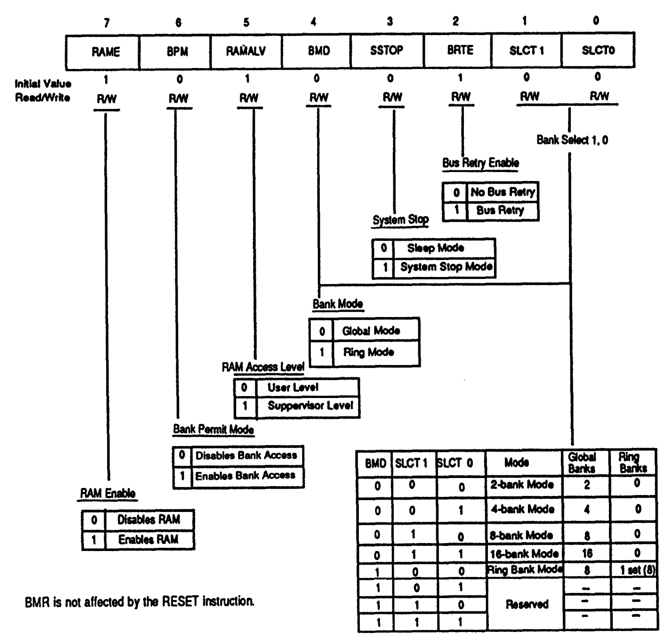


Figure 4-15. I/O Base register (IBR)

4.3.7 Bank Mode Register (BMR)

Figure 4-16 shows the bank mode register BMR, which determines the usage of internal RAM. The BMR register is not affected by the RESET instruction.



BMD	SLCT 1	SLCT 0	Mode	Global Banks	Ring Banks
0	0	0	2-bank Mode	2	0
0	0	1	4-bank Mode	4	0
0	1	0	8-bank Mode	8	0
0	1	1	16-bank Mode	16	0
1	0	0	Ring Bank Mode	8	1 set (8)
1	0	1	Reserved	–	–
1	1	0		–	–
1	1	1		–	–

Figure 4-16. Bank Mode Register (BMR)

RAM Enable (RAME): If RAME = 1, internal RAM is enabled. If RAME = 0, it is disabled. If an access is made to an internal RAM address when RAME = 0, external RAM is accessed. RAME is initialized to 1 at reset.

Bank Permit Mode (BPM): If BPM = 1, register banks cannot be accessed as data RAM. If BPM = 0, they can be accessed as data RAM. When BPM = 0, the register banks will always be read as undefined value. When BPM = 0, the contents of the register banks will not be affected by writes. When RAME = 0, external RAM is always accessed, regardless of BPM. BPM is initialized to 0 at reset.

RAM Access Level (RAMALV): If RAMALV = 0, an internal bus master (CPU or internal DMA controller) can access the internal RAM in user mode. If RAMALV = 1, an internal bus master cannot access the internal RAM in user mode; if accessed, an access level exception occurs. RAMALV is initialized to 1 at reset.

Bank Mode (BMD): If BMD = 1, register banks can be used in the ring mode. If BMD = 0, they can be used in the global mode. BMD is initialized to 0 at reset.

System Stop (SSTOP): When SSTOP is set to 1, the MPU enters system stop mode when it executes the SLEEP instruction. When SSTOP is cleared to 0, the MPU enters sleep mode when it executes the SLEEP instruction. See "Section 15. Low Power Consumption Modes" for details.

Bus Retry Enable (BRTE): If BRTE = 0, the CPU executes the bus cycle in bus error mode. If BRTE = 1, the CPU executes the bus cycle in bus retry mode. See "4. 6 Processing States and Privilege Modes" for details.

Bank Select 1, 0 (SLCT1, SLCT0): SLCT1 and SLCT0, combined with BMD, determine register bank function as shown in Table 4-3. They are initialized to 00 at reset.

Table 4-3. Global Register Bank Function

BMD	SLCT1	SLCT0	Mode	No. of Global Banks	No. of Ring Banks
0	0	0	2-bank mode	2	0
0	0	1	4-bank mode	4	0
0	1	0	8-bank mode	8	0
0	1	1	16-bank mode	16	0
1	0	0	Ring mode	8	1 set (8)
1	0	1	Reserved	-	-
1	1	0			
1	1	1			

4.3.8 Bank Stack Pointer (BSP)

Figure 4-17 shows the bank stack pointer BSP. BSP can be used to save/restore the ring bank registers when the ring bank overflows or underflows in ring mode. BSP is not initialized at reset and is not affected by the RESET instruction. See "4.4 Register Banks" for details.

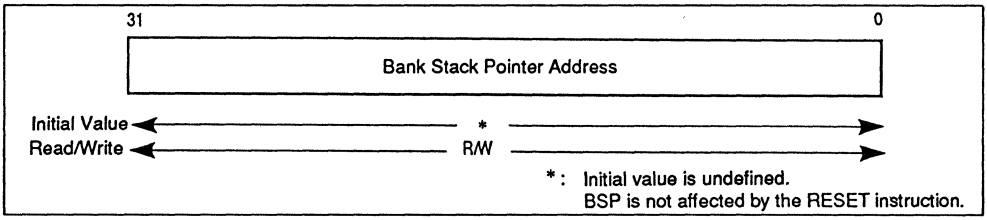


Figure 4-17. Bank Stack Pointer (BSP)

4.3.9 Global Bank Number Register (GBNR)

Figure 4-18 shows the global bank number register (GBNR) which specifies the global bank number. GBNR is not affected by the RESET instruction. See "4.4 Register Banks" for details.

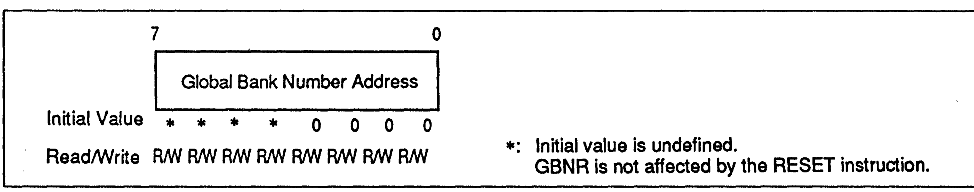


Figure 4-18. Global Bank Number Register (GBNR)

4.3.10 Current Bank Number Register (CBNR)

Figure 4-19 shows the current bank number register (CBNR) which specifies the current bank number. CBNR is not affected by the RESET instruction. See "4.4 Register Banks" for details.

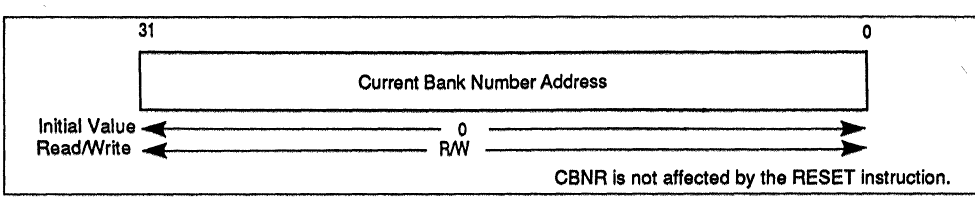


Figure 4-19. Current Bank Number Register (CBNR)

4.3.11 Valid Bank Number Register (VBNR)

Figure 4-20 shows the valid bank number register (VBNR) which indicates the valid bank number. VBNR is not affected by the RESET instruction. See "4.4 Register Banks" for details.

	7							0
	Valid Bank Number Address							
Initial value	*	*	*	*	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

*: Initial value is undefined.
VBNR is not affected by the RESET instruction.

Figure 4-20. Valid Bank Number Register (VBNR)

4.3.12 General-Purpose Registers

Each bank has sixteen (R0-R15) 32-bit general-purpose registers. They enable the CPU to handle data effectively and support versatile addressing modes. Normal instructions and addressing modes specify general-purpose registers by encoding the register's number in 4 bits. That is, R0 is 0000, R5 is 0101, R15 is 1111.

Each of the general-purpose registers can be used as a data register, address registers, or index registers. R15 can also be used as a stack pointer USP or SSP. See 4.3.13 for details. The general-purpose registers are not initialized at reset.

4.3.13 R15 and the Stack Pointer

The HD641016 has two types of stack pointers, user stack pointer (USP) and supervisor stack pointer (SSP). The S bit of SR determines which is used. If S is 1, SSP is used, if 0, USP is used. The designated stack pointer is used under the following circumstances:

- The stack pointer is implied by an instruction (for example, subroutine call instruction BSR).
- R15 of the global bank is specified in an addressing mode (for example, @R15). However, when R15 of a ring bank is specified, R15 of the currently selected ring bank is used, regardless of the S bit value.

SSP is always used during exception processing since the S bit is set to 1. When byte data is pushed by using USP or SSP, the USP or SSP is decremented by two; when byte data is pulled, the USP or SSP is incremented by two. For further details, refer to "4.6 Processing States and Privilege Modes".

4.3.14 Stack Configuration

Figure 4-21 shows stack access using USP or SSP when the stack pointer (USP or SSP) is set to an even address.

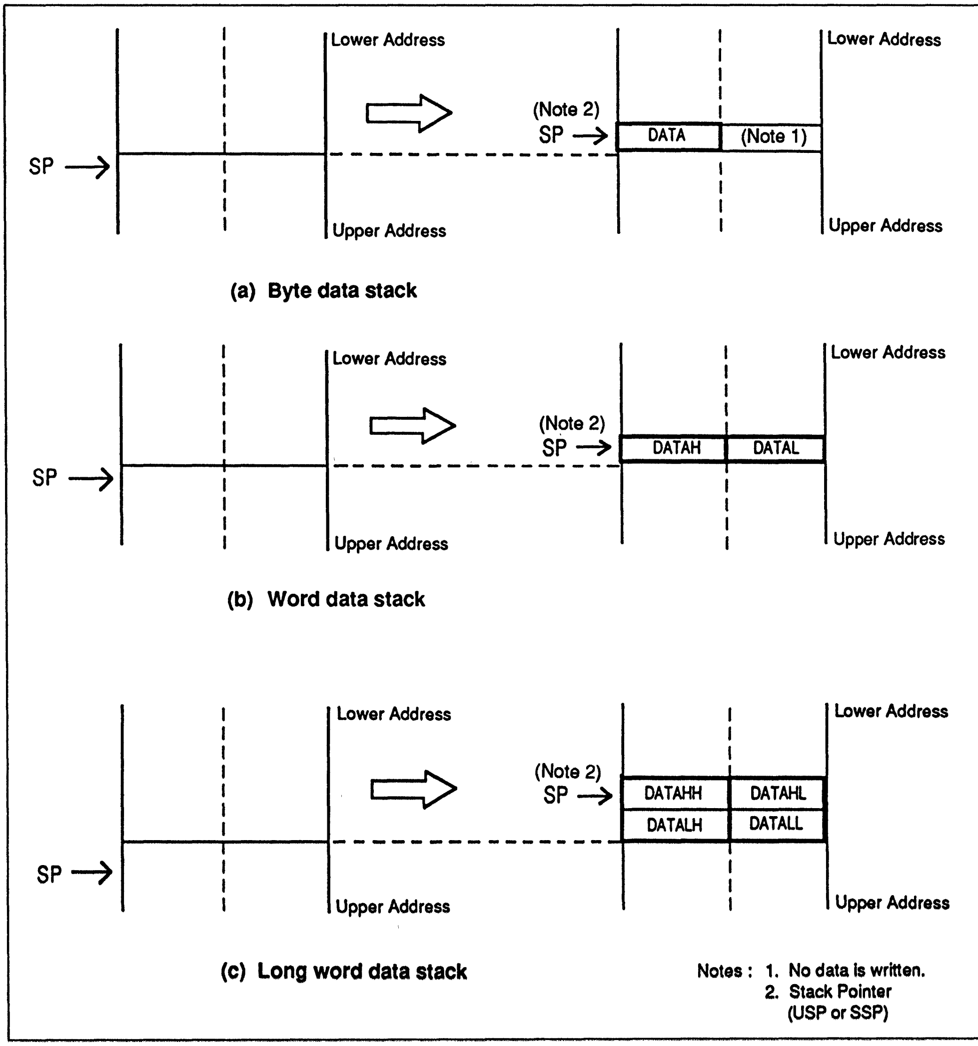


Figure 4-21. Stack Configuration (SP: Even Address)

Figure 4-22 shows stack access when the stack pointer (USP or SSP) is set to an odd address.

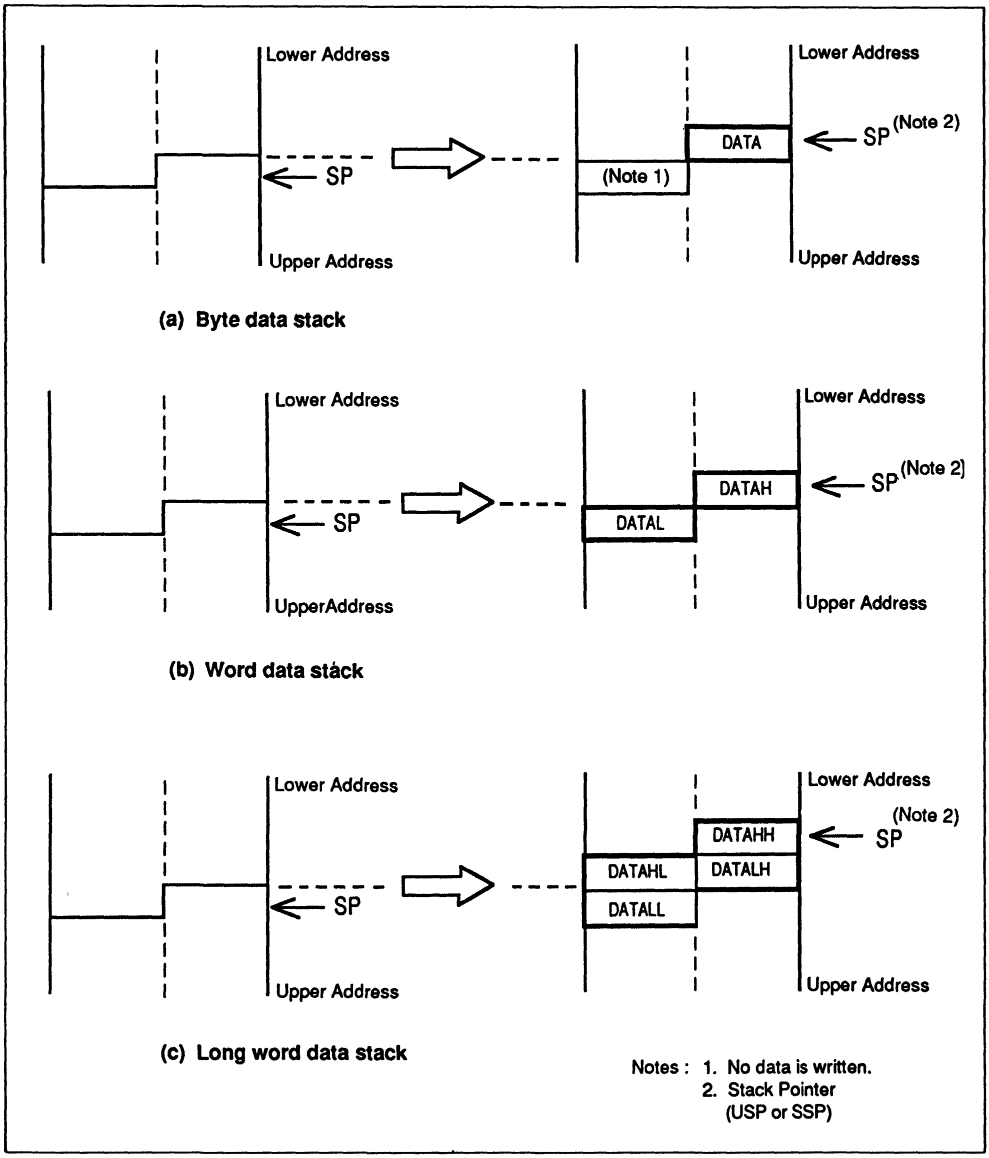


Figure 4-22. Stack Configuration (SP: Odd Address)

The HD641016 provides the register indirect auto-increment and the register indirect auto-decrement addressing modes. In these two modes, any general-purpose registers can be used as the stack pointer. In these modes, stack configuration for word and long word data is the same as in Figure 4-21 and Figure 4-22. However, stack configuration for byte data differs as shown in Figure 4-23.

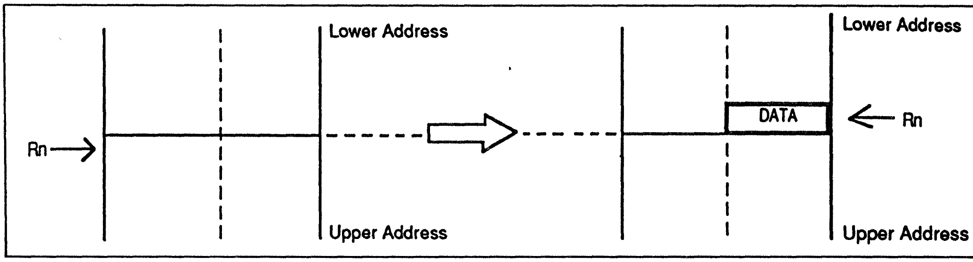


Figure 4-23. Stack Configuration (@-Rn, Data Size: Byte)

4.4 Register Banks

The HD641016 contains a maximum of sixteen register banks in 1024 bytes of internal RAM. Each bank consists of sixteen 32-bit registers R0-R15. Accordingly, if such a register is accessed, internal RAM must be accessed. In addition, the banks can be accessed in both global and ring modes as described below. See Figure 4-24.

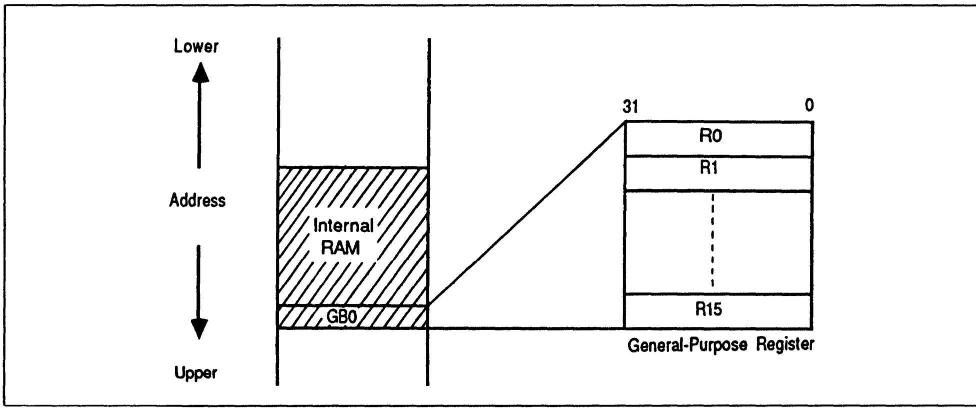


Figure 4-24. Internal RAM Space and Banks

4.4.1 Global Mode

Global Mode Specification: In global mode, either 2, 4, 8, or 16 banks can be used by programming the BMR register as shown in Table 4-4.

Table 4-4. BMR Values and Their Corresponding Bank Modes

BMR Bits

BMD	SLCT1	SLCT0	Mode
0	0	0	2-bank mode
0	0	1	4-bank mode
0	1	0	8-bank mode
0	1	1	16-bank mode

In 2-bank mode, up to 2 register banks can be used. The two banks can be switched by programming GBNR. If bit 0 of GBNR is programmed as 0, bank 0 is used, while if it is set to 1, bank 1 can be used. Note that only bit 0 of GBNR is valid in this mode. See Figure 4-25.

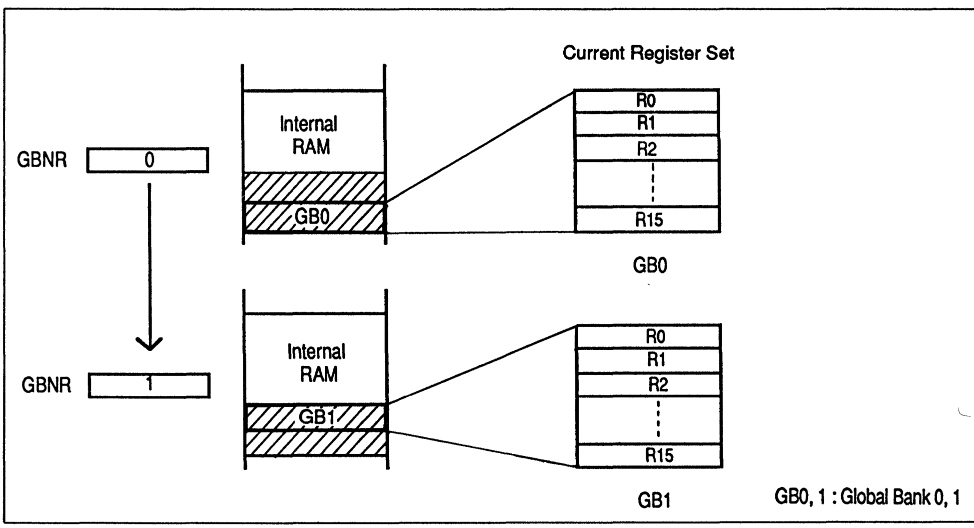
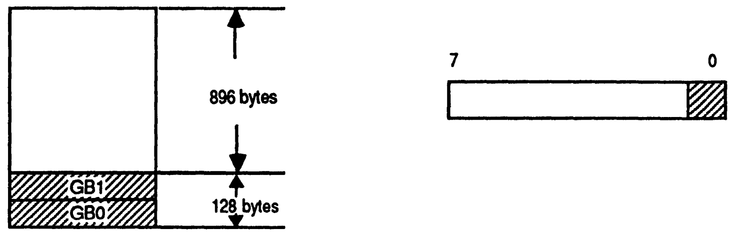
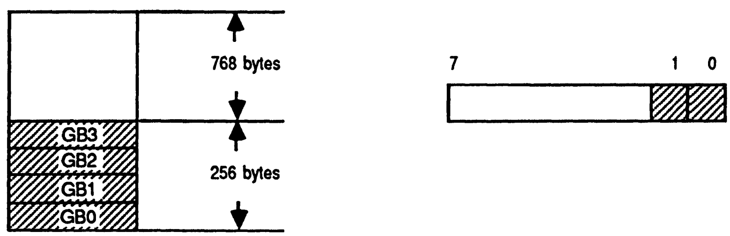
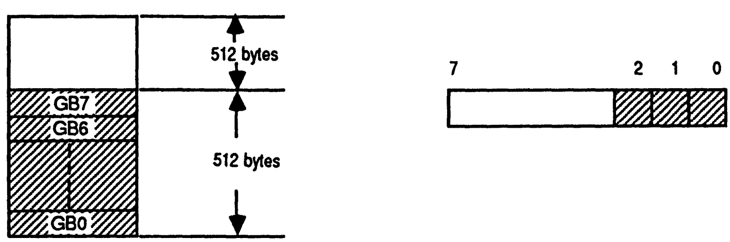
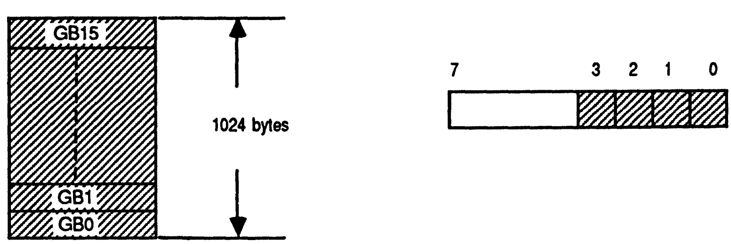


Figure 4-25. Global Bank Switch

In 4-, 8-, or 16-bank mode, bits 0-1, 0-2, or 0-3 of GBNR are used to control banks, respectively. Table 4-5 shows the relationship between bank mode and GBNR valid bits.

Table 4-5. Bank Mode and GBNR Valid Bits

Bank Mode	Available Banks	Valid Bits of GBNR
2-Bank Mode		
4-Bank Mode		
8-Bank Mode		
16-Bank Mode		

Global Bank Switching: As mentioned above, banks can be switched by programming the GBNR register with either a control register (CR) handling instruction (ANDC, ORC, XORC, LDC or STC), or a bank instruction (CGBN, or PGBN).

1. Global bank switching by a CR handling instruction (ANDC, ORC, XORC, LDC or STC)

   Figure 4-26 shows an example of global bank switching by the LDC instruction. In this example, bank 0 is switched to bank 3 in 4-bank mode by the LDC instruction. See "4.5 Instruction Set" for details on CR handling instructions and their addressing modes.

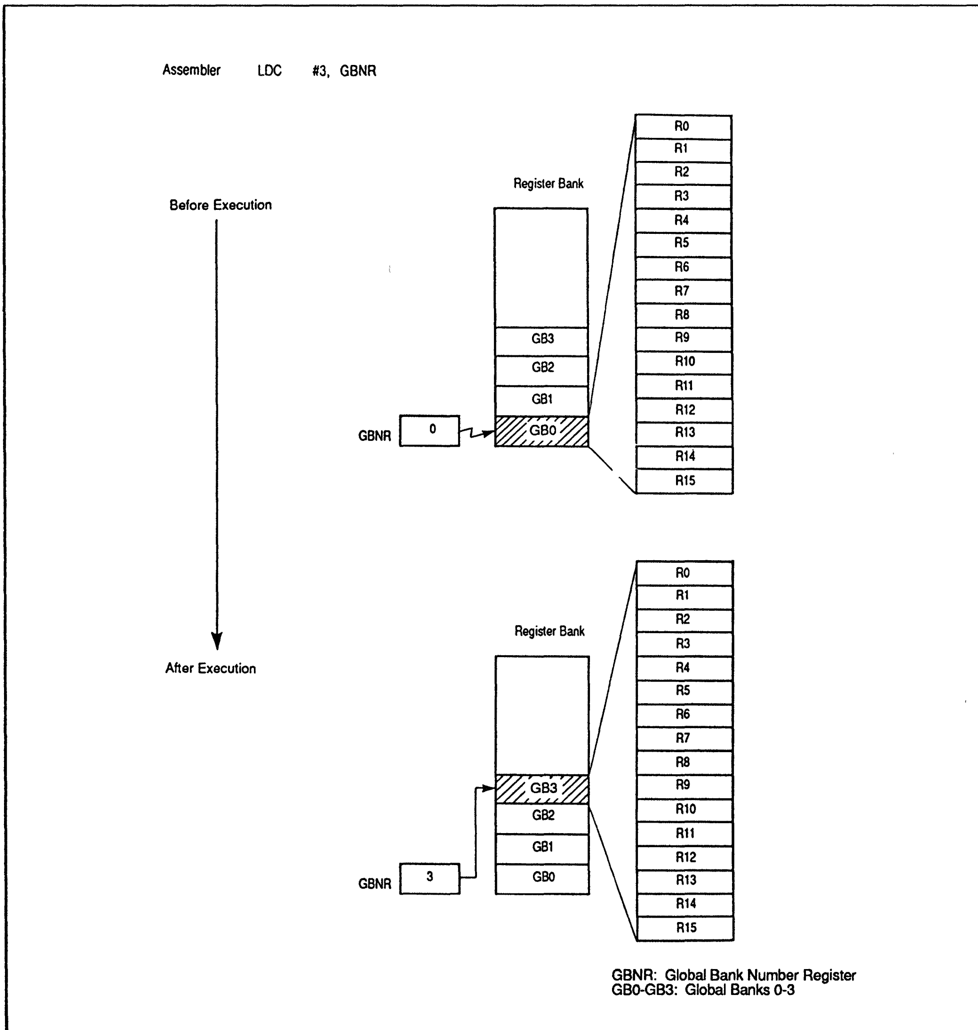


Figure 4-26. Global Bank Switching by LDC

2. Global bank switching by a bank instruction (CGBN or PGBN)

When global bank is switched by a bank instruction, register copy and the save and restore of the bank number are enabled.

Figure 4-27 shows an example of global bank switching by the privileged CGBN (push and change global bank number) instruction. CGBN first pushes the current GBNR value onto SSP and then loads-operand data into the new GBNR. At this time, the contents of selected global bank registers can be copied into new global bank registers if required. In addition, the operand of CGBN can be specified as static data or dynamic data: that is, immediate data or data specified by register. CGBN can be used only in supervisor state.

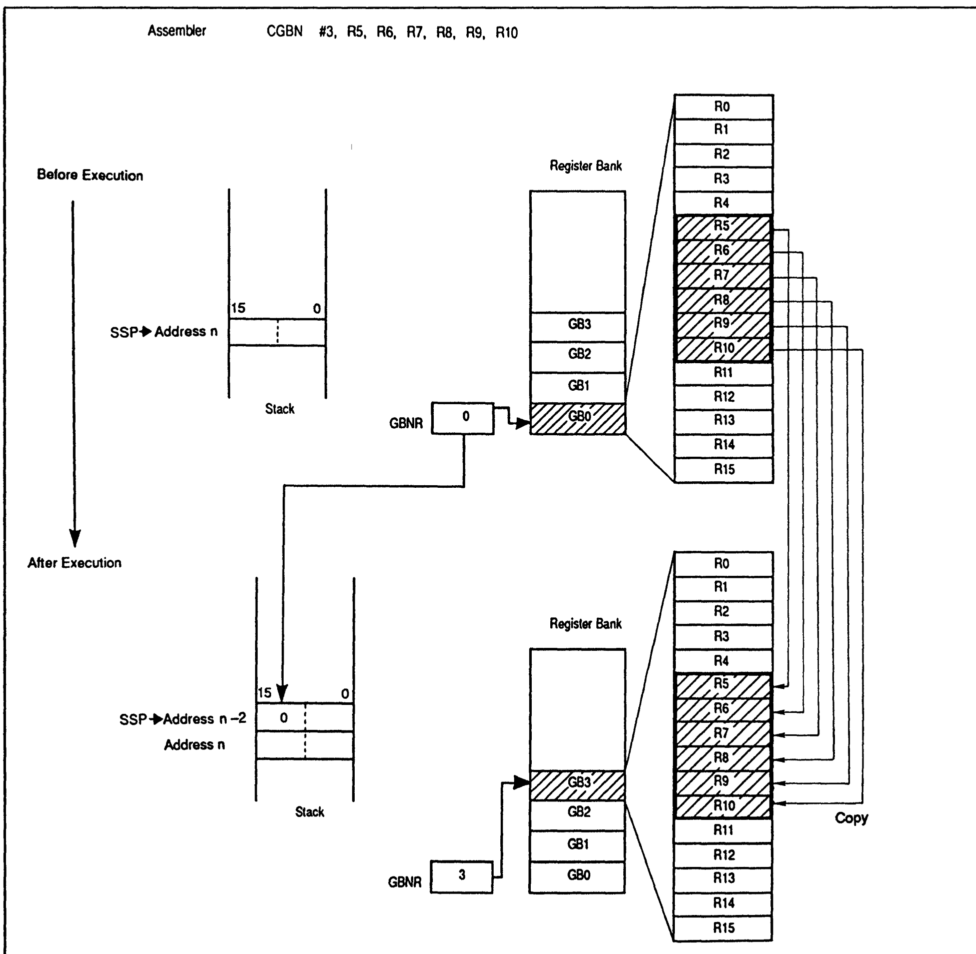


Figure 4-27. Global Bank Switching by CGBN

Figure 4-28 shows an example of bank switching by the privileged PGBN (Pull Global Bank Number) instruction. PGBN first pulls GBNR from the SSP and then loads the popped value into the new GBNR. At this time, it can copy the contents of selected global bank registers into new global bank registers if required. PGBN can be used only in supervisor state.

In Figure 4-28, bank 3 is switched to bank 0 and the contents of R0-R3 are copied into the new R0-R3.

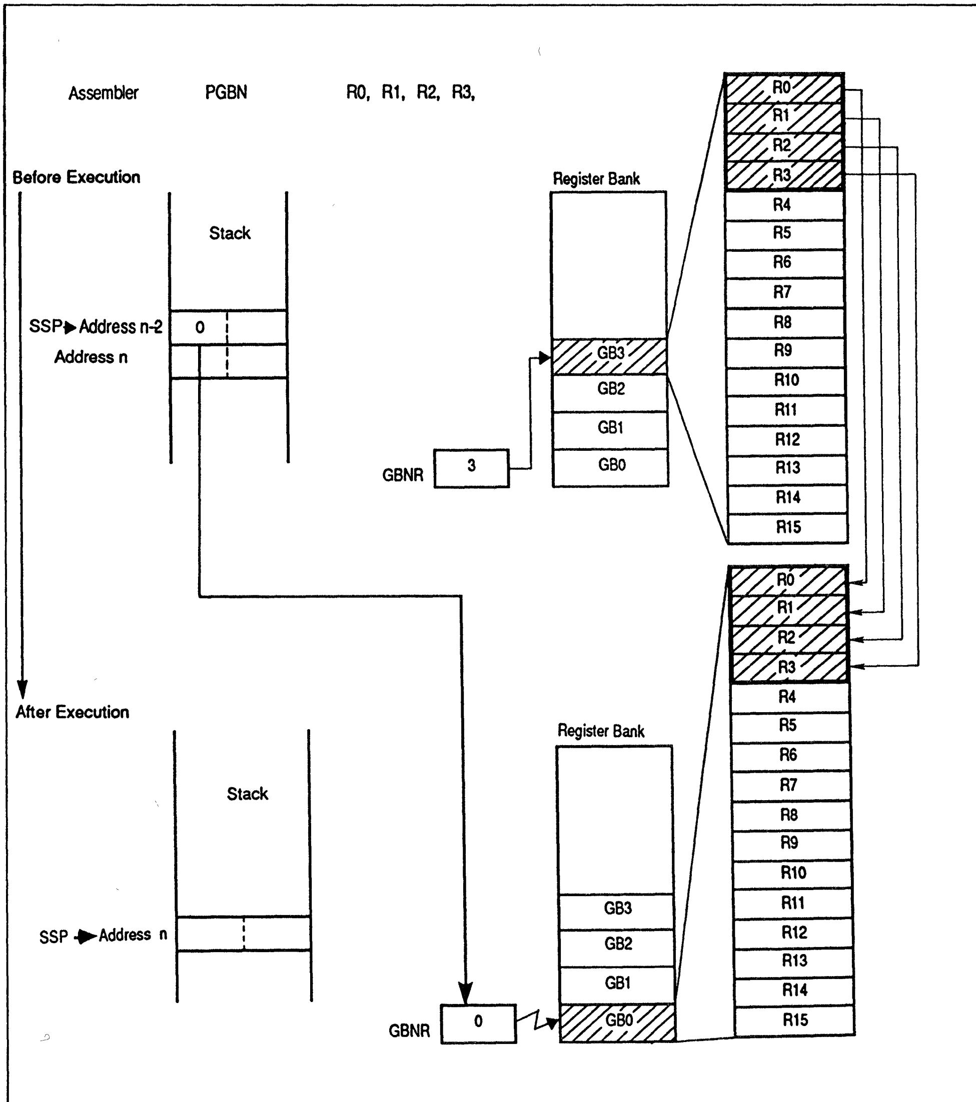


Figure 4-28. Global Bank Switching by PGBN

RAM Access in Global Mode: In 2-, 4-, or 8-bank mode, 896, 768, or 512 bytes of the remaining internal RAM area can be used as either data memory space or external memory space according to the setting to the RAME and BPM bits of BMR.

If RAME is set to 1, the remaining internal RAM area can be accessed as data memory area. If it is cleared to 0, the remaining RAM area will be accessed as external memory. In addition, the remaining internal RAM area, other than register banks, can be accessed as data memory area if the RAME is set to 1 and the BPM is cleared to 0. See " Section 5. RAM " for details. Figure 4-29 shows the data accessible areas.

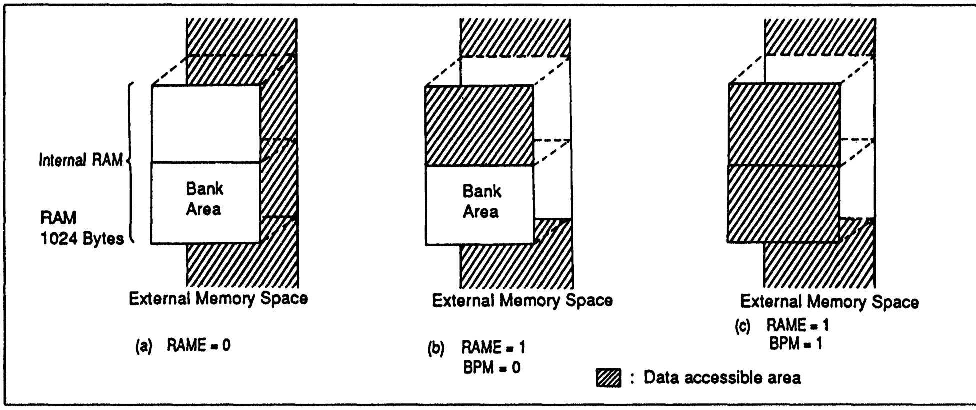


Figure 4-29. Data Accessible Area

4.4.2 Ring Mode

Ring Mode Specification: In ring mode, the CPU can use 8 global banks and a set of 8 ring banks. 8 global banks can be used in the same way as in 8-bank mode. In this mode, the BMD, SLCT1, and SLCT0 bits are programmed as shown in Table 4-6. Figure 4-30 shows bank configuration in ring mode.

Table 4-6. BMR Bit Settings and Mode

BMR Bits			
BMD	**SLCT1**	**SLCT0**	**Mode**
1	0	0	Ring mode

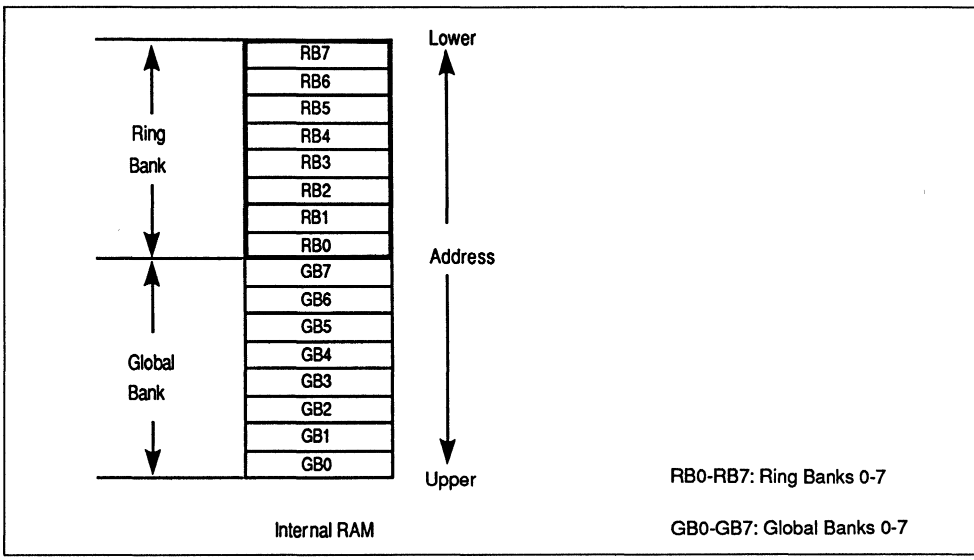


Figure 4-30. Ring Bank Configuration

Ring Bank Model: Figure 4-31 shows the ring bank model. Ring banks are controlled by BSP, CBNR, and VBNR.

The LCBNR bits (lower 3 bits) of CBNR indicate which bank is currently used. In general the bank indicated by the LCBNR bits is called "the current bank", while the bank indicated by the value of LCBNR bits – 1 is known as the "previous bank".

The LVBNR bits (lower 3 bits) of VBNR indicate up to which bank contains valid data. The upper 5 bits of VBNR are undefined.

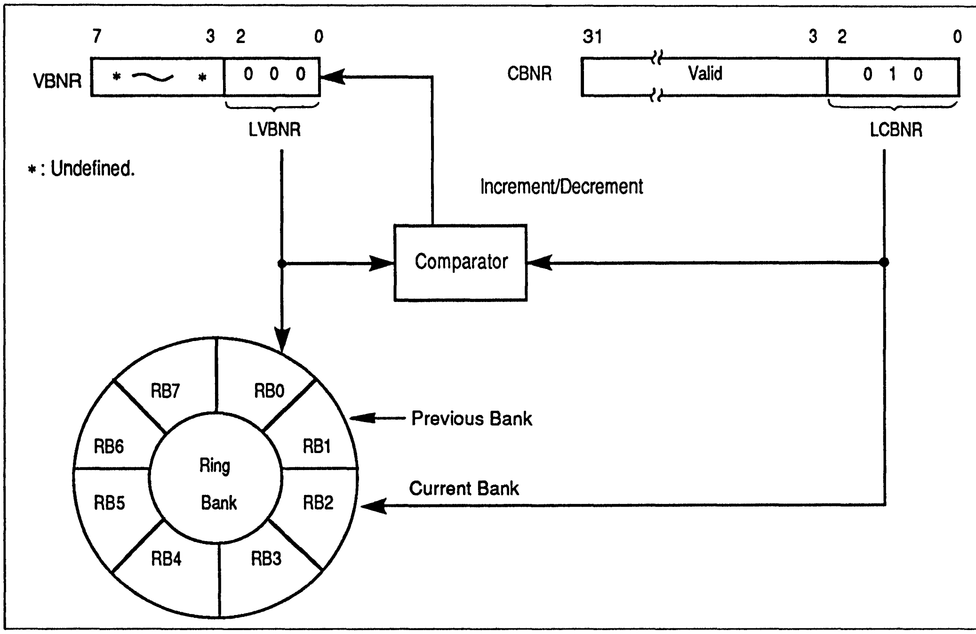


Figure 4-31. Ring Bank Model (When RB0-RB2 contain valid data)

Ring Bank Access: The current bank or previous bank can be accessed by selecting either if current bank or previous bank addressing mode. Figure 4-32 shows an example of global and ring bank accesses by GBNR and CBNR. In this example, CBNR is set to 2 and GBNR is set to 5. If global bank register 9 (R9) is specified, R9 of global bank 5 (GB5) specified by GBNR is selected. If current bank register 9 (CR9) is specified, R9 of ring bank 2 (RB2) specified by CBNR is selected. If previous bank register 9 (PR9) is specified, R9 of ring bank 1 (RB1) whose value is indicated by CBNR – 1 is selected.

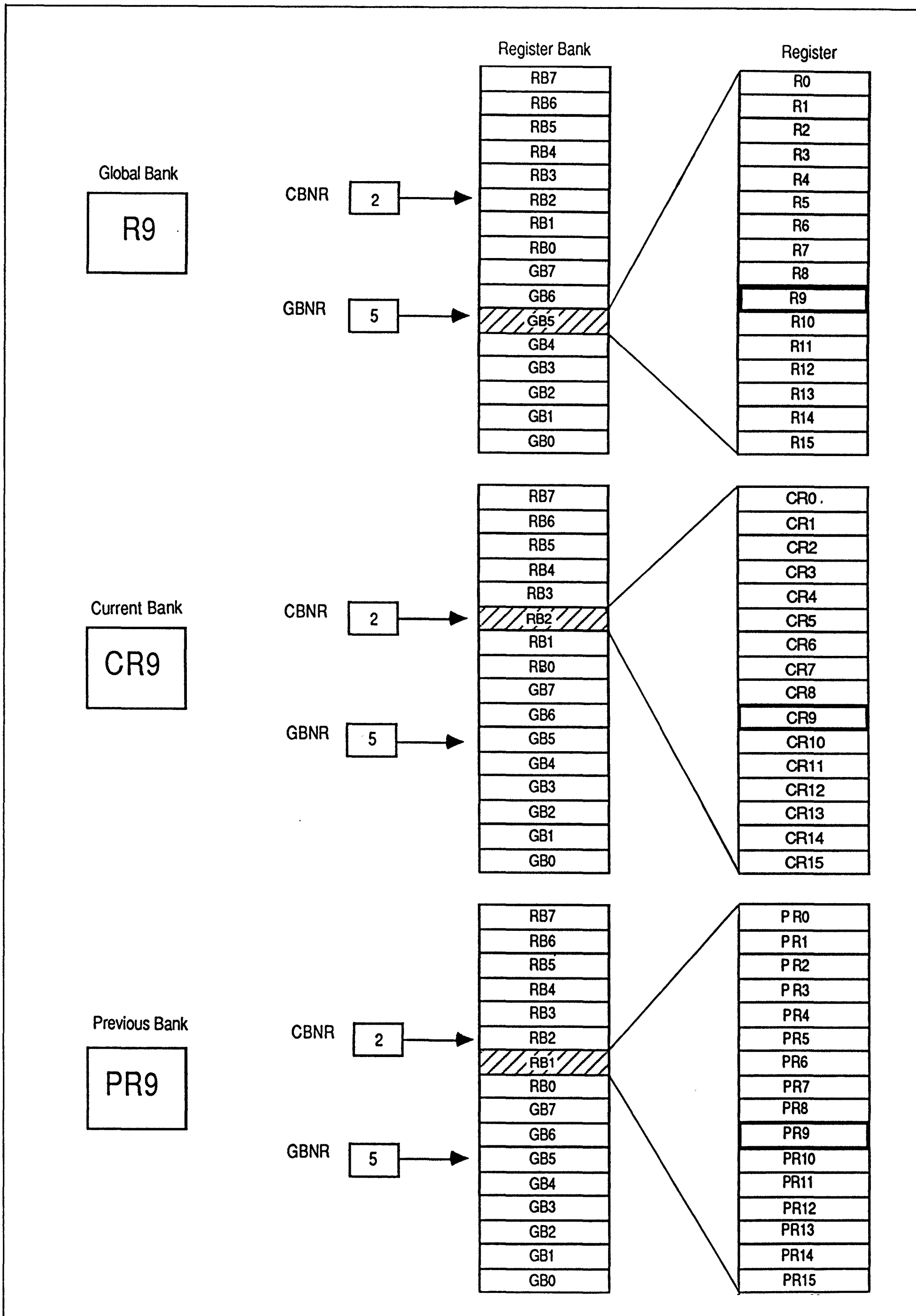


Figure 4-32. Global and Ring Bank Accessing Though GBNR and CBNR

Figure 4-33 shows an example in which the contents of R4 in a global bank are moved to CR12 in the current bank by the MOV instruction.

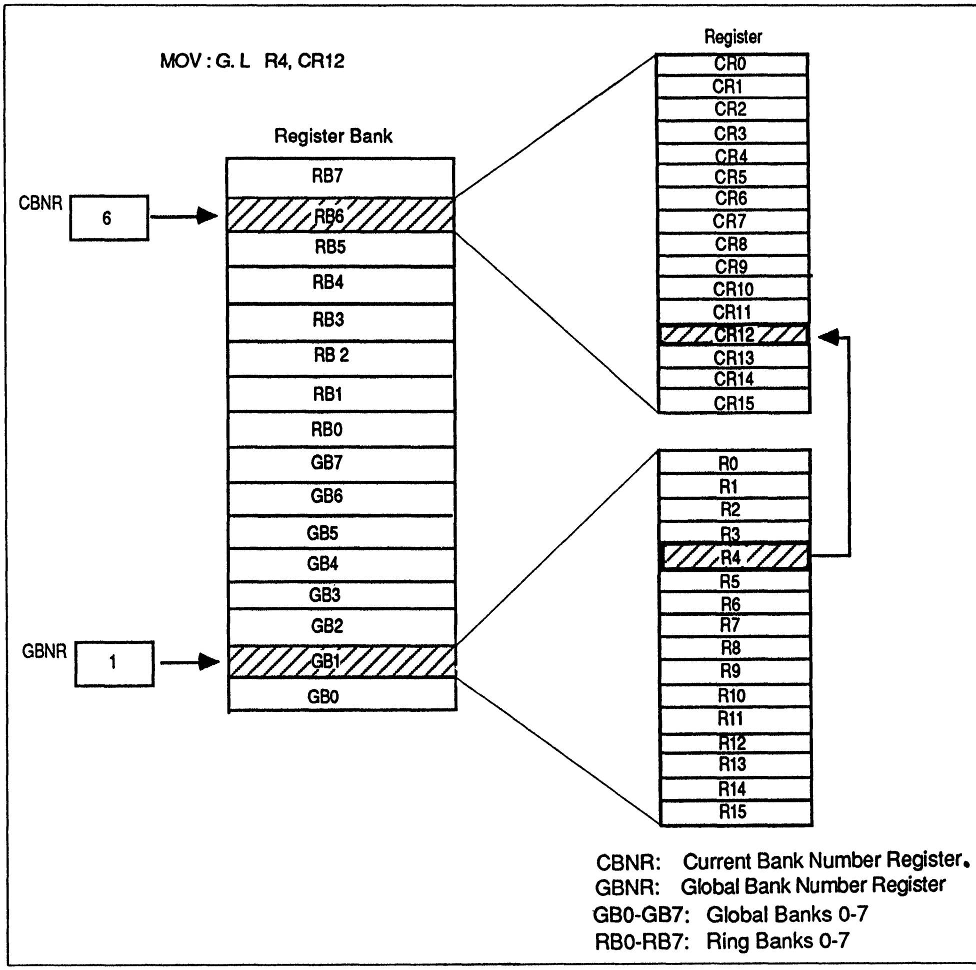


Figure 4-33. Current Bank Specification

Ring Bank Control Instructions: The ICBN (increment current bank number) and DCBN (decrement current bank number) instructions can be used to switch ring banks.

The ICBN instruction increments the CBNR register by 1. The DCBN instruction decrements CBNR by 1. Figure 4-34 shows an example of ring bank control by ICBN and DCBN. In the example, the current bank is RB3. If ICBN is executed, RB4 becomes the current bank. On the other hand, if DCBN is executed, RB2 becomes the current bank.

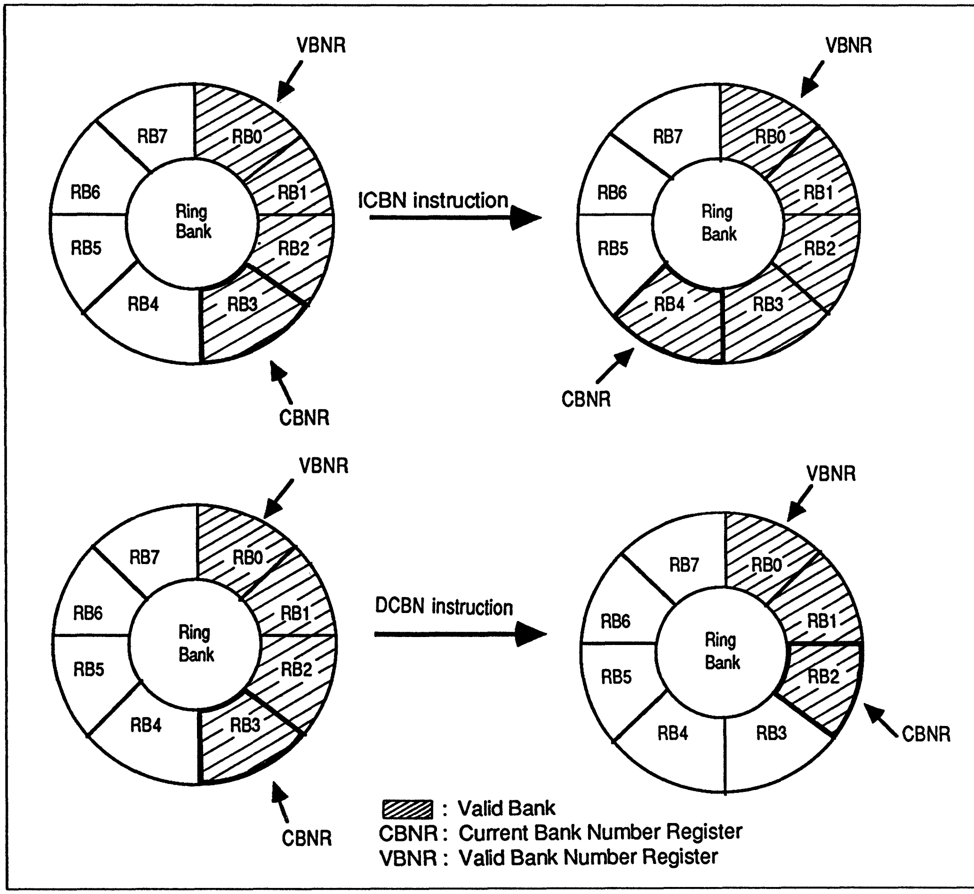


Figure 4-34. Ring Bank Control by ICBN and DCBN Instructions

In addition, ring banks can be used as variable areas for subroutines. This feature supports high-speed subroutine call and return. Figure 4-35 shows an example with the local variable areas of the procedures located in ring banks.

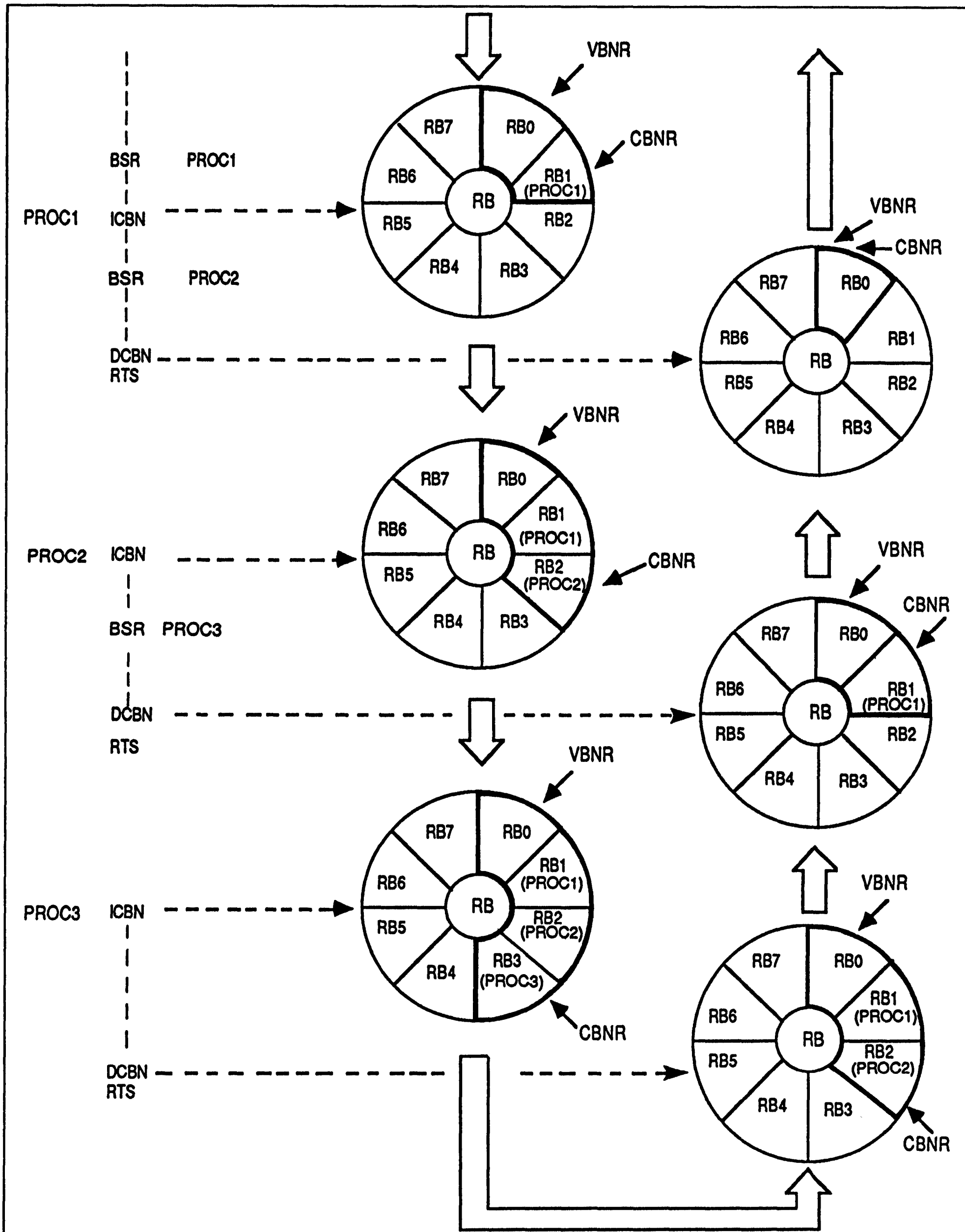


Figure 4-35. Procedure Call Ring Bank Application

Ring Bank Overflow: If the LCBNR bits of CBNR match the LVBNR bits of VBNR after executing the ICBN instruction, the ring bank overflows. At this time, the CPU operates as follows:

1. Pushes the registers of the bank indicated by LCBNR onto the stack area pointed to by 32-bit BSP. At this time, the lower 16 bits of R15 in the bank indicated by the LVBNR bits determine which register are pushed onto the stack. Bits 0-15 of 32-bit R15 correspond to 32-bit R0-R15 of the bank. If a bit in R15 is set, the corresponding 32-bit register is pushed onto the stack. The registers are pushed onto the stacks in the ascending order. However, note that R15 itself is always stacked regardless of the state of bit 15 of R15.

2. Increments VBNR by 1 and decrements BSP according to the number of stacked bytes.

3. Executes the next instruction.

Figure 4-36 shows an example of ring bank overflow. In this example, VBNR and CBNR indicate RB0 and RB7 respectively before executing the ICBN instruction. If ICBN is executed, the LCBNR bits and LVBNR bits match and ring bank overflows occurs. The CPU then stack registers. R0, R1, R6, R7, R10, R13, R14, and R15 as specified by the bit values of R15 value. Finally, the CPU increments VBNR by 1.

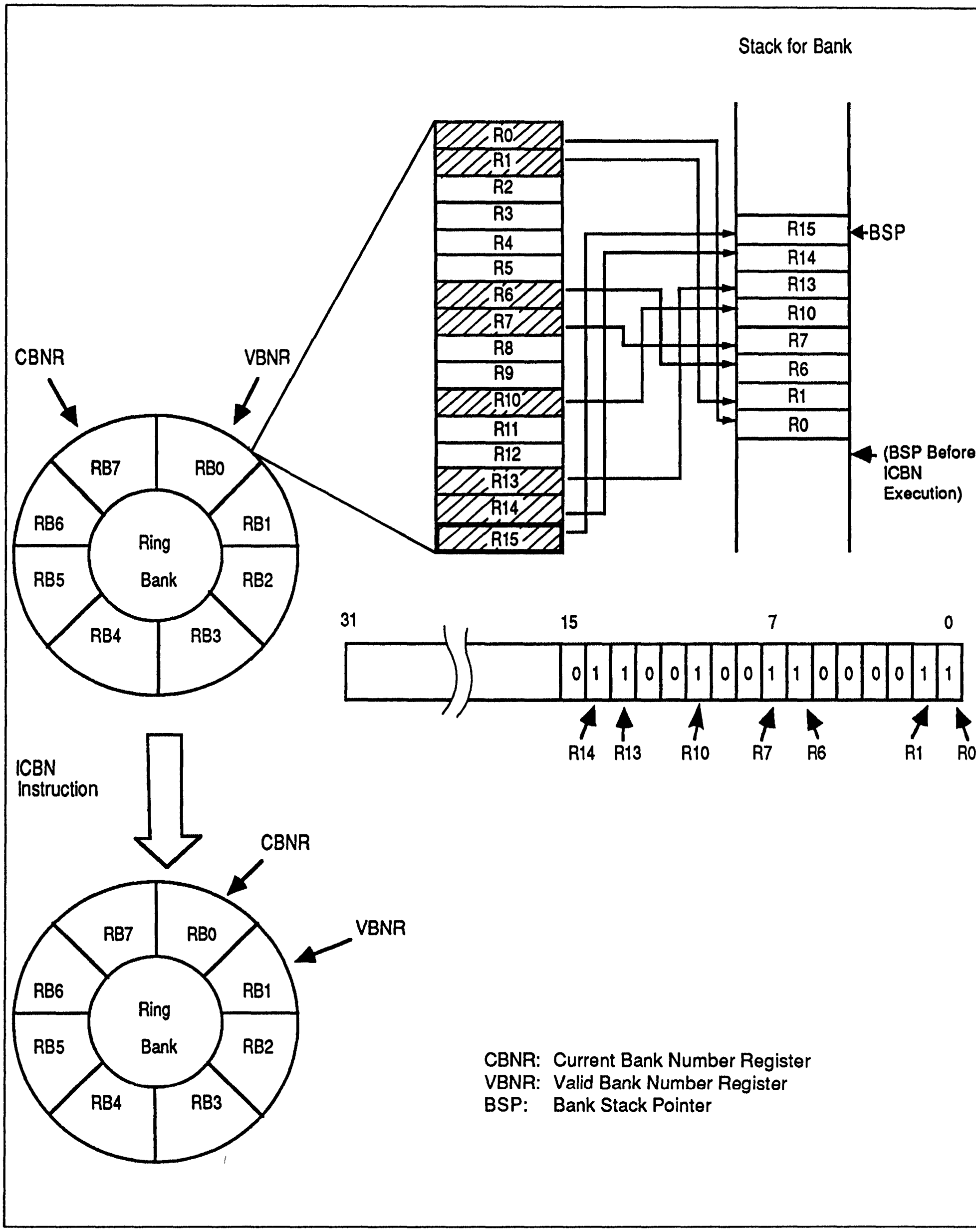


Figure 4-36. Ring Bank Overflow

Ring Bank Underflow: If the LCBNR bits of CBNR match the LVBNR bits of VBNR after executing the DCBN instruction, ring bank underflow occurs. If CBNR is 0, the CPU executes the next instruction. Otherwise, the CPU operates as follows:

1. Decrements VBNR by 1.
2. Pulls R15 from the stack location pointed to by BSP. The CPU then pulls the registers described by R15 into the bank designated by VBNR. R15 functions as a register list as mentioned in ring bank overflow.
3. Increments BSP according to the number of bytes to be pulled.
4. Executes the next instruction.

Figure 4-37 shows an example when the ring bank underflow. In this example, VBNR and CBNR indicate RB2 and RB3 respectively before executing the ICBN instruction. If DCBN is executed, the LCBNR bits and LVBNR bits match and ring bank underflow occurs. At the same time R15 is read from the stack location pointed to by BSP. Then R9, R6, R2, and R0 are pulled from the stack onto RB1, as is R15.

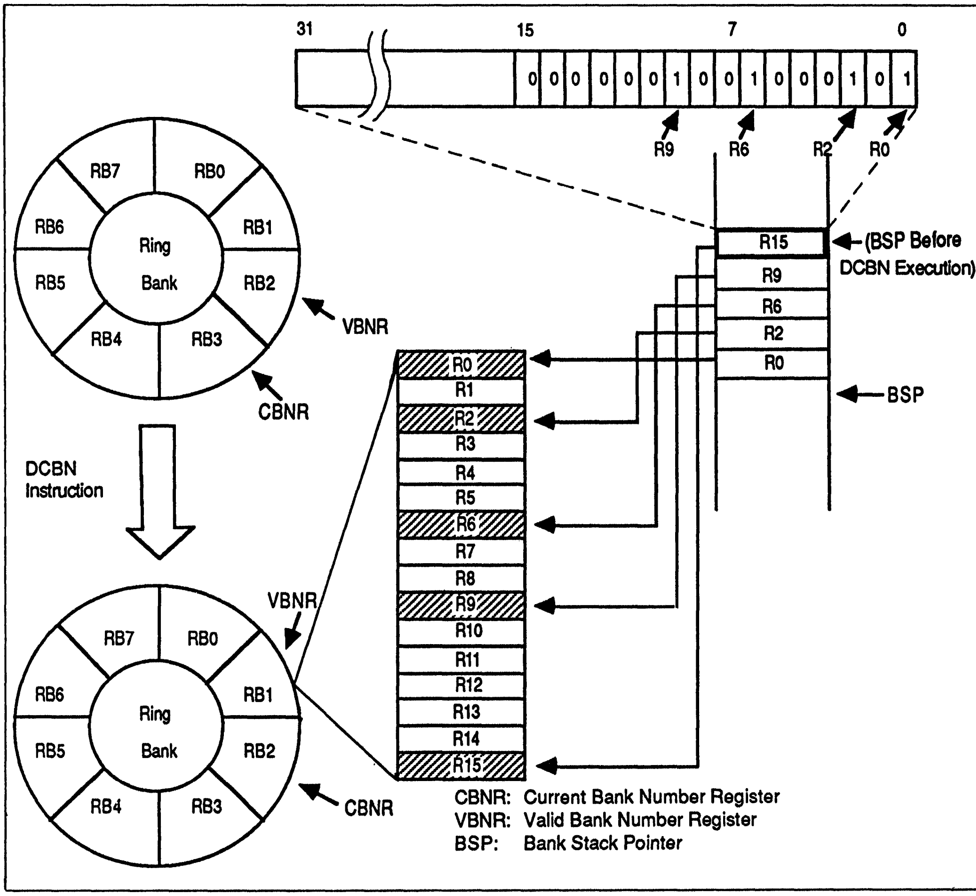


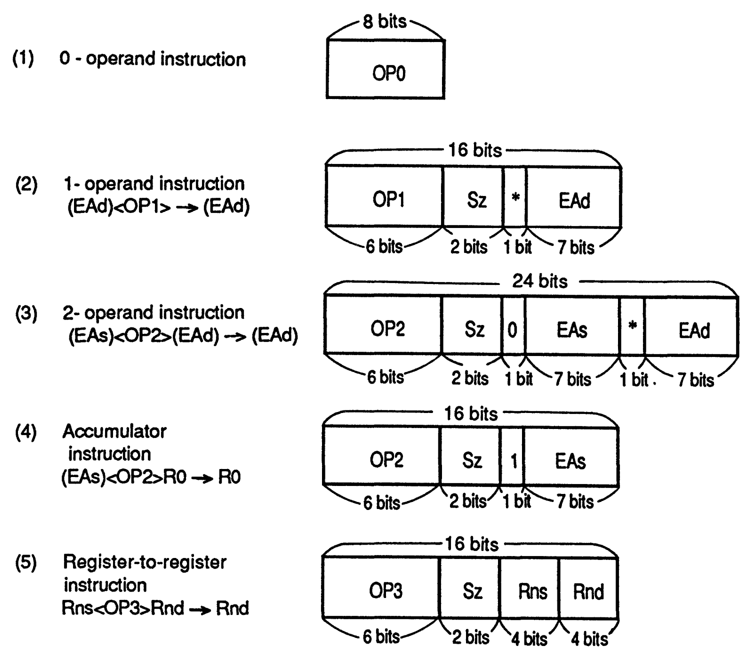
Figure 4-37. Ring Bank Underflow

4.5 Instruction Set

For most instructions, source and destination operand can be specified independently in the allowable addressing modes. Operand size (byte, word, or long word) can be specified independently of an instruction. Instructions consist of one or more bytes. In addition, short format instruction are provided to improve software efficiency. Moreover, this instruction set considering high level languages such as C and PASCAL allows the programmer to improve program development efficiency.

4.5.1 Basic Instruction Formats

The HD641016 has five kinds of basic instruction (Figure 4-38): no-operand, 1-operand, 2-operand, accumulator (R0), and register-to-register instructions (R0-R15).



OP3-OP0 :	Opcode fields which specify instruction function
EAs, EAd :	Effective address codes which specify source and destination addressing modes.
Sz :	Size specification field which specifies operand size
Rns, Rnd :	Global bank register number specification field
R0 :	Global bank register 0.
* :	Don't care

Figure 4-38. Basic Instruction Formats

An EA expansion field may be inserted after each effective address (EA) specifying field, according to the addressing mode. Figure 4-39 shows this for a 2-operand instruction.

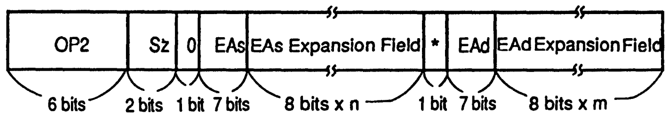


Figure 4-39. 2-operand Instruction with EA Expansion Field

4.5.2 Addressing Mode

The addressing mode is specified by the EA specifying field, and, if necessary, an expansion field. The EA specifying field and the EA expansion field together are called the EA field. The EA specifying field consists of an accumulator specifying bit and an EA code (Figure 4-40).

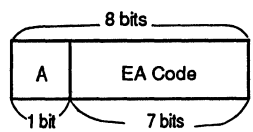


Figure 4-40. EA Specifying Field

Accumulator Specifying Bit (A): A indicates whether R0 (accumulator) is implicitly specified for the destination of a 2-operand instruction. If A = 1, R0 is always selected as the destination and the destination EA specifying field (EAd) is not required. If A = 0, the EAd field specifies the destination operand. See Figure 4-41.

Note that A is ignored in one operand instructions. In addition, an illegal instruction exception occurs if A is set to 1 in a two-operand instruction whose destination addressing mode must not be register direct.

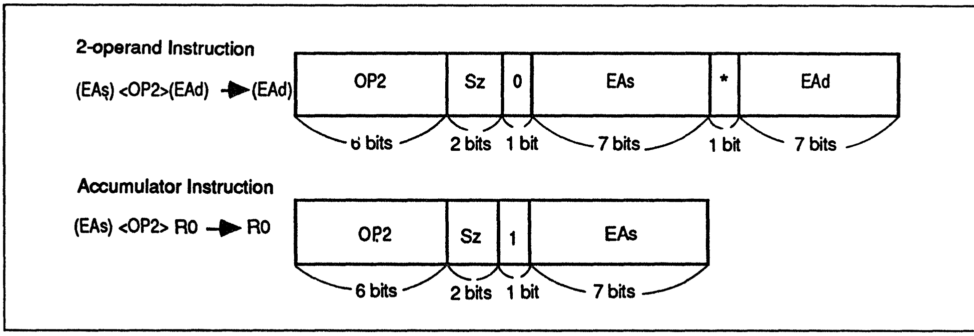


Figure 4-41. 2-Operand Instruction and Accumulator Instruction

EA Code: A 7-bit EA code specifies an addressing mode or general-purpose register. In addition, an EA expansion byte may follow the EA code depending on the addressing mode to be specified (Table 4-7).

Table 4-7. EA Codes and Addressing Modes

No.	Addressing Mode	Mnemonic	EA Code	Expansion Byte, Others		
1	Register direct	Rn	1 0 0 [Rn]	No expansion byte Rn : Register number		
2	Register indirect	@Rn or @(disp[:lng], Rn)	0 [Sd] [Rn]	**Sd**	**Expansion Byte**	**No. of Bytes**
				00	None	0 byte
				01	d8	1 byte
				10	d16	2 bytes
				11	d32	4 bytes
3	Register indirect auto-increment	@Rn+	1 0 1 [Rn]	No expansion byte		
4	Register indirect auto-decrement	@-Rn	1 1 0 [Rn]	No expansion byte		

Table 4-7. EA Codes and Addressing Modes (cont.)

<table>
<tr><th>No.</th><th>Addressing Mode</th><th>Mnemonic</th><th>EA Code</th><th>Expansion Byte, Others</th></tr>
<tr><td>5</td><td>Immediate</td><td># xxxx[.Sz]</td><td>1 1 1 0 0 [Si]</td><td>
<table>
<tr><th>Si</th><th>Expansion Byte</th><th>No. of Bytes</th></tr>
<tr><td>01</td><td>[Imm8]</td><td>1 byte</td></tr>
<tr><td>10</td><td>[Imm16]</td><td>2 bytes</td></tr>
<tr><td>11</td><td>[Imm32]</td><td>4 bytes</td></tr>
</table>
Note: Si = 00 indicates the EA code for current bank addressing mode.
</td></tr>
<tr><td>6</td><td>Absolute Address</td><td>@ aaaa[.Sz]</td><td>1 1 1 0 1 [Sa]</td><td>
<table>
<tr><th>Sa</th><th>Expansion Byte</th><th>No.of Bytes</th></tr>
<tr><td>01</td><td>[Abs8]</td><td>1 byte</td></tr>
<tr><td>10</td><td>[Abs16]</td><td>2 bytes</td></tr>
<tr><td>11</td><td>[Abs32]</td><td>4 bytes</td></tr>
</table>
Note: Sa = 00 indicates the EA code for previous bank addressing mode.
</td></tr>
<tr><td>7</td><td>Register indirect with scale</td><td>@Rn * Sf,
@(disp, Rn * Sf)
or
@(disp[:lng], Rn * Sf)</td><td>1 1 1 1 0 [Sd]</td><td>
[* | * | Sf | Rn] [disp]
<table>
<tr><th>Sf</th><th>Scaling Factor</th></tr>
<tr><td>00</td><td>x 1</td></tr>
<tr><td>01</td><td>x 2</td></tr>
<tr><td>10</td><td>x 4</td></tr>
<tr><td>11</td><td>x 8</td></tr>
</table>
* : Don't care
Refer to No. 2 for details on Sd and disp.
</td></tr>
<tr><td>8</td><td>Register indirect with index</td><td>@([disp[:lng],] Xm [.Sz][*Sf], Rn)</td><td>1 1 1 1 1 0 0</td><td>
[0 | L | Sd | Rn] [** | Sf | Xm] [disp]
<table>
<tr><th>L</th><th>Index Register Size</th></tr>
<tr><td>0</td><td>Xm: Word</td></tr>
<tr><td>1</td><td>Xm: Long Word</td></tr>
</table>
Xm: Index Register number Refer No. 7 for details on other symbols.
</td></tr>
</table>

Table 4-7. EA Codes and Addressing Modes (cont.)

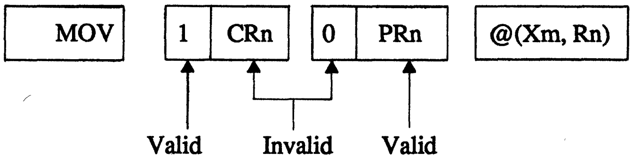
No.	Addressing Mode	Mnemonic	EA Code	Expansion Byte, Others
9	Program counter relative with index	@([disp[:lng],] Xm [.Sz][*Sf], PC]	1 1 1 1 1 0 1	0 | L | Sd | **** | | ** | Sf | Xm | | disp Refer to No. 8 for details on each symbol.
10	Program counter relative	@PC or @(disp[:lng], PC)	1 1 1 1 1 0 1	1 | 0 | Sd | **** | | disp Refer to No. 2 for details on each symbol.
11	Register double indirect	@@Rn or @([disp1[:lng],] @([disp2[:lng],] Rn))	1 1 1 1 1 1 0 (note) S2, ds2 is the same as S1, ds1.	S2 | S1 | Rn | | ds1 | | ds2 **Si** — **Expansion Byte** — **No.of Bytes** 01 — d8 — 1 byte 11 — d32 — 4 bytes
12	Current Bank	<CRn>	1 1 1 0 0 0 0	Specify any EA code
13	Previous Bank	<PRn>	1 1 1 0 1 0 0	Specify any EA code
14	Not used (Note: If unused addressing mode is specified, the CPU begins illegal instruction exception processing.)		1 1 1 1 1 0 0	1 0 0 0 0 0 0 0 ~ 1 1 1 1 1 1 1 1
			1 1 1 1 1 0 1	1 1 0 0 0 0 0 0 ~ 1 1 1 1 1 1 1 1
			1 1 1 1 1 1 0	All other S1 and S2 combinations than the following in the register indirect mode **S2** — **S1** 01 — 01 01 — 11 11 — 01 11 — 11
			1 1 1 1 1 1 1	None

Symbols:

Rn	:	Register number
Sd	:	Displacement size
disp	:	Displacement
d8	:	Displacement (8 bits)
d16	:	Displacement (16 bits)
d32	:	Displacement (32 bits)
Si	:	Immediate data size
Imm8	:	Immediate value (8 bits)
Imm16	:	Immediate value (16 bits)
Imm32	:	Immediate value (32 bits)
Sa	:	Absolute size
Abs8	:	Absolute value (8 bits)
Abs16	:	Absolute value (16 bits)
Abs32	:	Absolute value (32 bits)
Sf	:	Scale factor
Xm	:	Index register number
L	:	Index register size
PC	:	Program counter
ds1	:	Displacement 1
ds2	:	Displacement 2
S1	:	Displacement size 1
S2	:	Displacement size 2
CRn	:	Current bank register number
PRn	:	Previous bank register number
*	:	Don't care

Notes:
1. Rn and Xm are global bank registers specified by GBNR.
2. The <CRn> and <PRn> addressing modes are allowed only for ring mode. If they are specified in global mode, a bank mode exception occurs.
3. <CRn> mode in No. 12 can specify any of addressing modes No. 1 through No. 13. In <CRn> mode in No. 12, Rn is always a register in the current bank specified by CBNR. Xm in No. 8 is a register in the global bank specified by GBNR. Xm in No. 9 is a register in the current bank specified by CBNR.
   <PRn> mode in No. 13 is almost the same as <CRn> mode in No. 12 except that <PRn> mode uses a register in the previous bank specified by CBNR –1.
4. If more than one <CRn> or <PRn> mode is specified in <CRn> or <PRn> mode, the HD641016 uses only the last <CRn> or <PRn> and ignores all others as shown below. In these modes, bit 7 of the EA field is valid as the A bit. In the following

example, A is regarded as 1, and global bank and previous bank are specified as Xm and Rn, respectively.



Effective Address Calculation: The effective address (EA) for each addressing mode is calculated as shown in Table 4-8.

Table 4-8. Effective Address Calculation

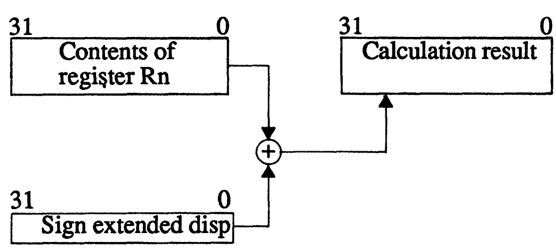
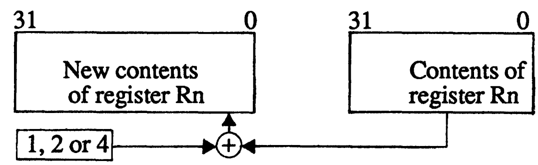
No.	EA Type		EA Calculation Data	Effective Address
1	EA Code	1 0 0 Rn		3 0 Rn
	Addressing Mode	Register direct	None	Note: Rn contains operand data.
	Mnemonic	Rn		
2	EA Code	0 Sd Rn		
	Addressing Mode	Register indirect		
	Mnemonic	@ Rn or @(disp[:lng], Rn)		
	Note: Sd = 00 No displacement = 01 8-bit displacement = 10 16-bit displacement = 11 32-bit displacement			

Table 4-8. Effective Address Calculation (cont.)

No.	EA Type		EA Calculation Data	Effective Address
3	EA Code	1 0 1 Rn		
	Addressing Mode	Register indirect auto-increment		
	Mnemonic	@Rn+	Rn is incremented by 1 for byte operand. Rn is incremented by 2 for word operand. Rn is incremented by 4 for long word operand. However, if Rn = R15, Rn is incremented by 2 for byte or word operand. It is incremented by 4 for long word operand.	



No.	EA Type		EA Calculation Data	Effective Address
4	EA code	1 1 0 Rn		
	Addressing Mode	Register indirect auto-decrement		
	Mnemonic	@-Rn	Rn is decremented in the same way as in auto-increment addressing mode.	

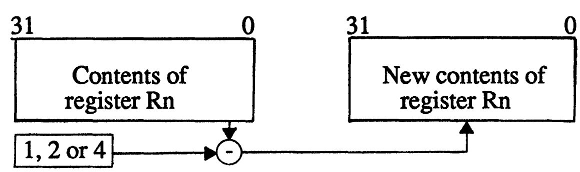


Table 4-8. Effective Address Calculation (cont.)

No.	EA Type	EA Calculation data	Effective Address
5	EA Code 1 1 1 0 0 $\boxed{Si}$ Addressing Mode: Immediate Mnemonic: # XXXX [.Sz] Note: Si = 01 Byte size = 10 Word size = 11 Long word size	None Operand data is contained in program.	None Byte or word immediate data is sign extended to long word.
6	EA Code 1 1 1 0 1 $\boxed{Sa}$ Addressing Mode: Absolute address Mnemonic: @ aaaa [.Sz] Note: Sa = 01 1-byte address = 10 2-byte address = 11 4-byte address	None Note: Address must be written to the EA expansion field in advance.	31 … 0 Sign extended absolute address

Table 4-8. Effective Address Calculation (cont.)

No. **EA Type** **EA Calculation Data** **Effective Address**

---

7 EA Code 1 1 1 1 0 Sd

Addressing Mode	Register indirect with scale
Mnemonic	@ Rn * Sf or @(disp[:lng], Rn * Sf)

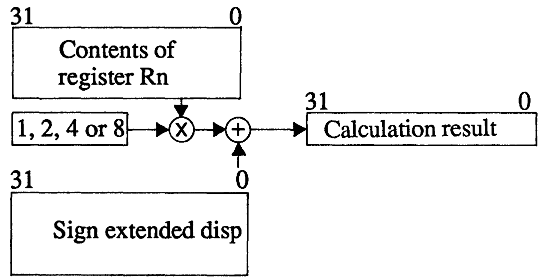
Note:
First expansion byte

**	Sf	Rn

*: Don't care

Sf = 00 Scale factor x 1
= 01 Scale factor x 2
= 10 Scale factor x 4
= 11 Scale factor x 8

Sd is the same as in @ Rn.



---

8 EA Code 1 1 1 1 1 0 0

Addressing Mode	Register indirect with index
Mnemonic	@([disp[:lng],] Xm [.Sz][*Sf], Rn)

Note:
First expansion byte

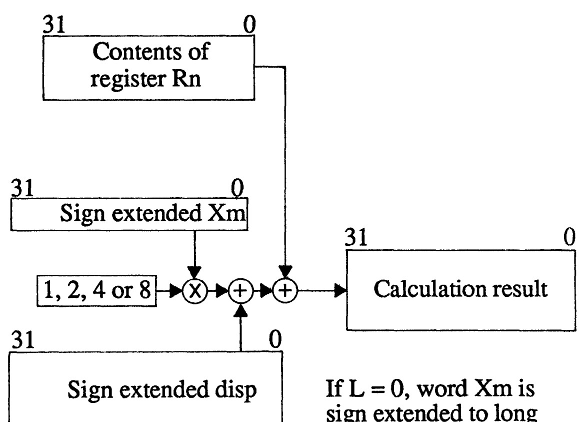
0	L	Sd	Rn

Another byte

**	Sf	Xm

*: Don't care.

All symbols are the same as these in @Rn * Sf mode.



If L = 0, word Xm is sign extended to long word.
If L = 1, long word Xm is used directly.

Table 4-8. Effective Address Calculation (cont.)

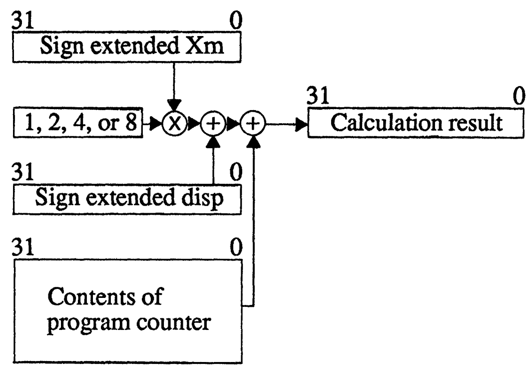
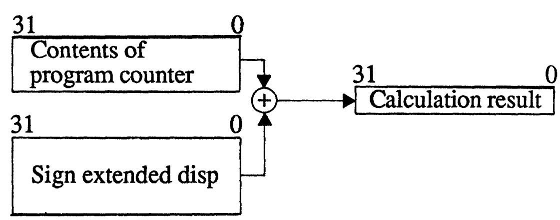
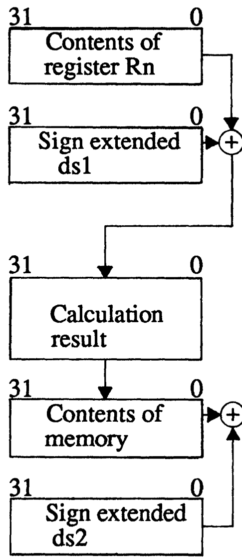
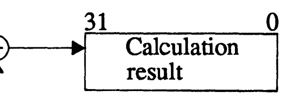
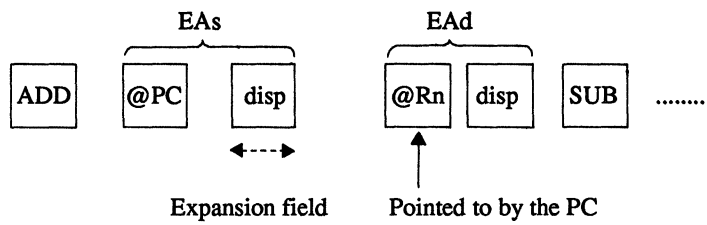
No.	EA Type	EA Calculation Data	Effective Address
9	EA Code 1 1 1 1 1 0 1 Addressing Mode: Program counter relative with index Mnemonic: @([disp[:lng],] Xm [.Sz][*Sf], PC) Note: First expansion byte | 0 | L | Sd | * * * * | Another byte | * * | Sf | Xm | *: Don't care PC points to the address following the expansion byte. All symbols are the same as those is @ (Xm,Rn) mode.		
10	EA Code 1 1 1 1 1 0 1 Addressing Mode: Program counter relative Mnemonic: @ PC or @(disp[:lng], PC) Note: First expansion byte | 1 | 0 | Sd | * * * * | *: Don't care PC points to the address following the expansion byte. Sd is the same as in @ Rn mode.		

Table 4-8. Effective Address Calculation (cont.)

No.	EA Type		EA Calculation Data	Effective Address
11	EA Code	1 1 1 1 1 1 0		(Note) S2 S1 / ds2 Expansion / ds1 Expansion 01 01 / 1 byte / 1 byte 01 11 / 1 byte / 4 bytes 11 01 / 4 bytes / 1 byte 11 11 / 4 bytes / 4 bytes All other combinations disabled. If specified,an illegal instruction exception processing occurs. 
	Addressing Mode	Register double indirect		
	Mnemonic	@@Rn or @([disp1[:lng],]) @([disp2[:lng],] Rn)		
	Note: First expansion byte S2 | S1 | Rn 2nd and 3rd expansion bytes are described in order of disp1 and disp2.			
12	EA Code	1 1 1 0 0 0 0		• All EAs available. • The CRn register is used instead of Rn and Xm. However, Xm in @ (Xm, Rn) always uses a global bank register.
	Addressing Mode	Current bank		
	Mnemonic	<CRn>		
13	EA Code	1 1 1 0 1 0 0		• All EAs available. • The PRn register is used instead of Rn and Xm. However, Xm in @ (Xm, Rn) move always uses a global bank register.
	Addressing Mode	Previous bank		
	Mnemonic	<PRn>		

Note: In PC relative with index or PC relative addressing modes, the PC points to the address following the expansion field. For example, in 1-operand instructions, the PC indicates the next opcode address. In 2-operand instruction, it points to the EAd (2nd EA) field address as shown below.

Example



Instruction Execution Examples in Each Addressing Mode: Figures 4-42 through 4-48 show instruction execution in all addressing modes in machine code.

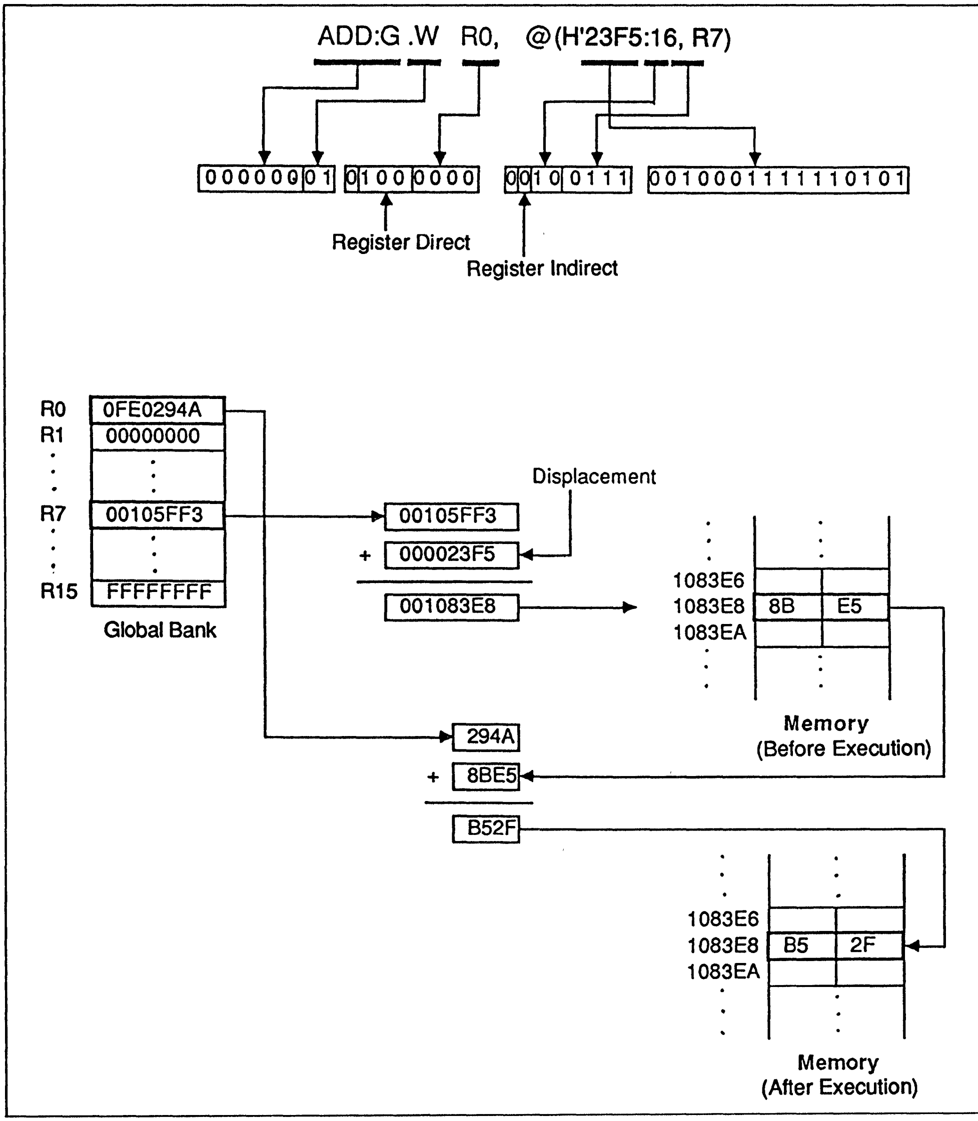


Figure 4-42. Instruction Execution Example in Register Direct/ Register Indirect Addressing Mode

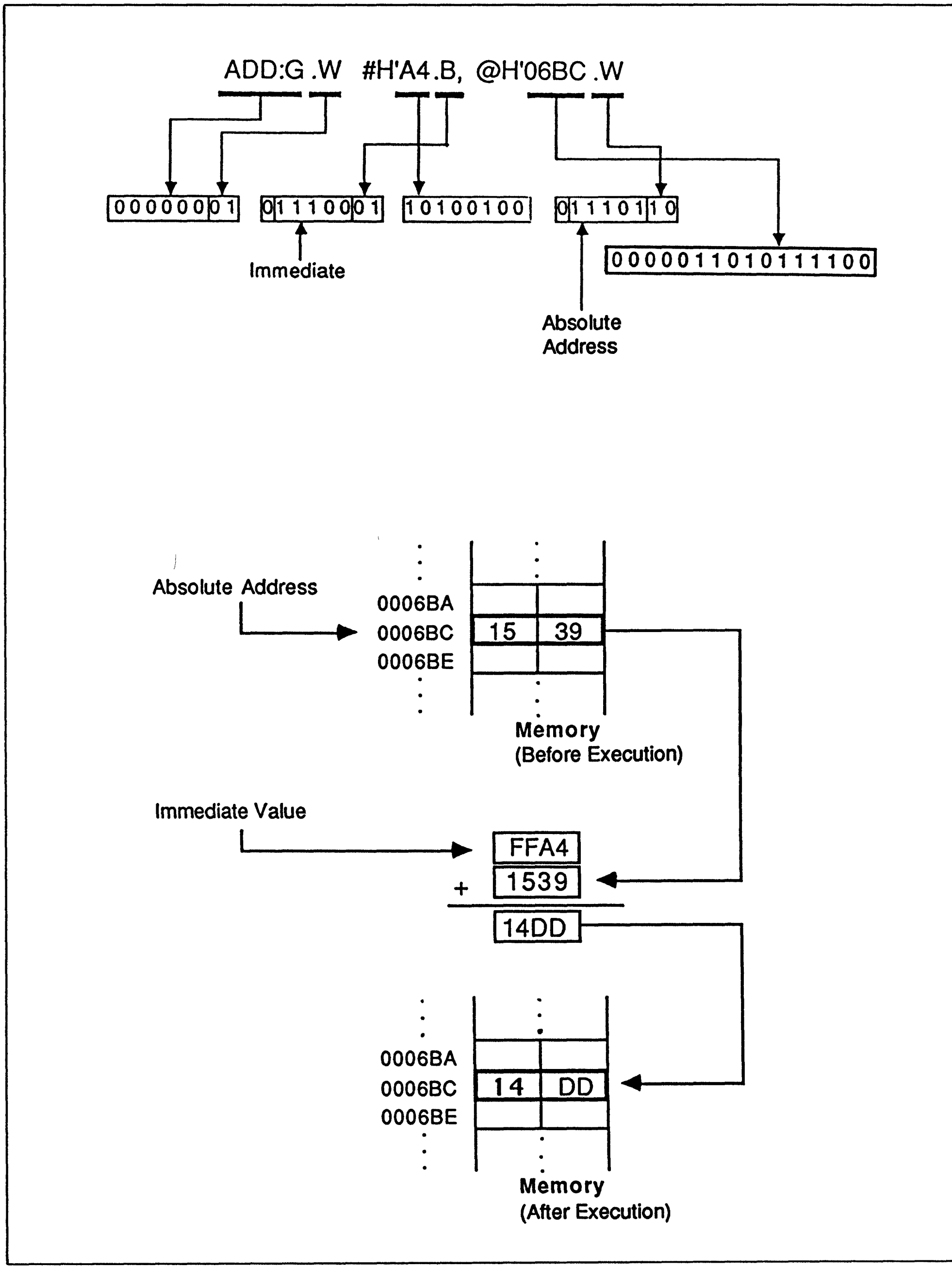


Figure 4-43. Instruction Execution Example in Immediate/ Absolute Addressing Mode

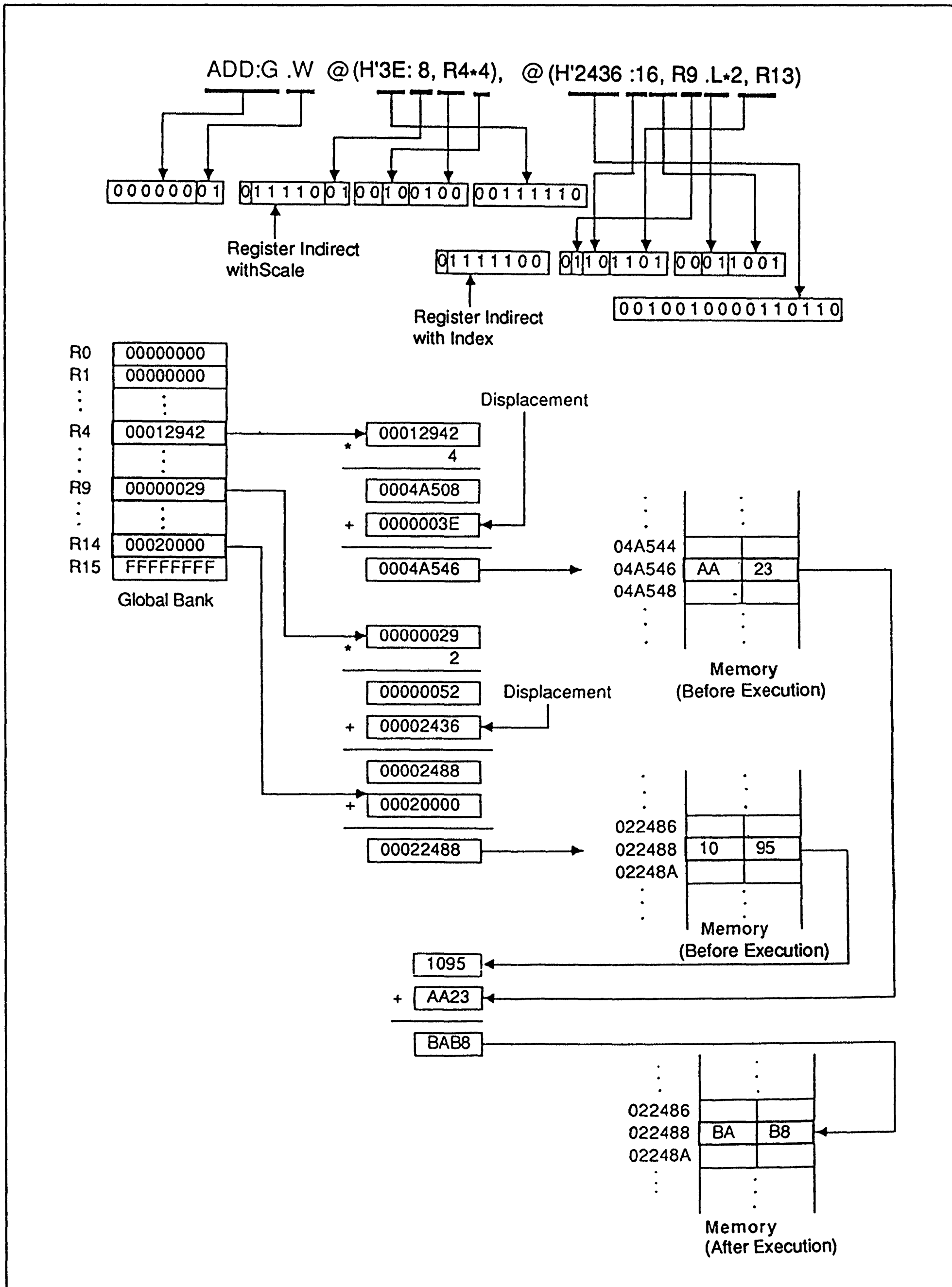


Figure 4-44. Instruction Execution Example in Register Indirect with Scale/ Register Indirect with Index

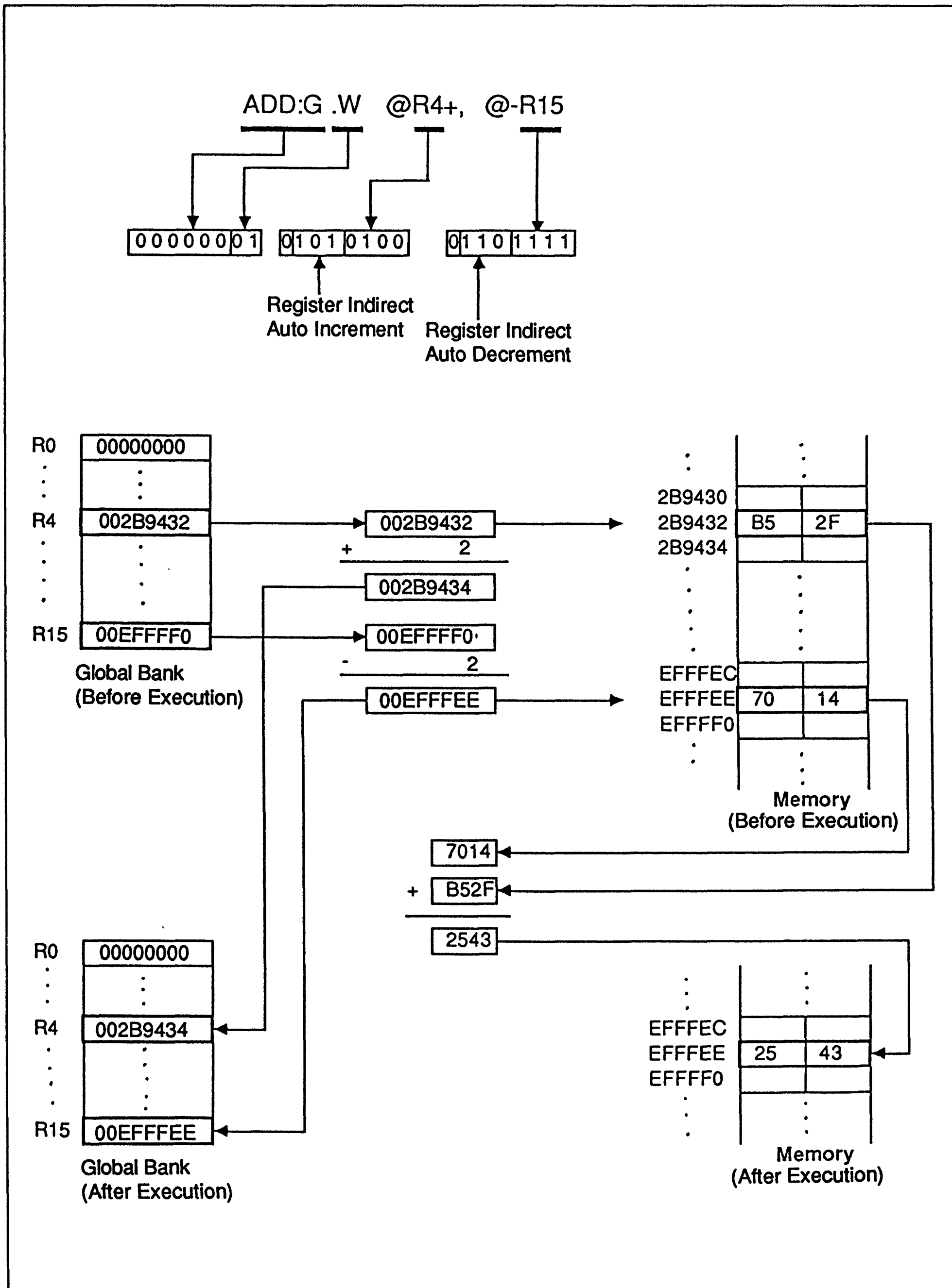


Figure 4-45. Instruction Execution Example in Register Indirect Auto Increment/ Register Indirect Auto Decrement Addressing Mode

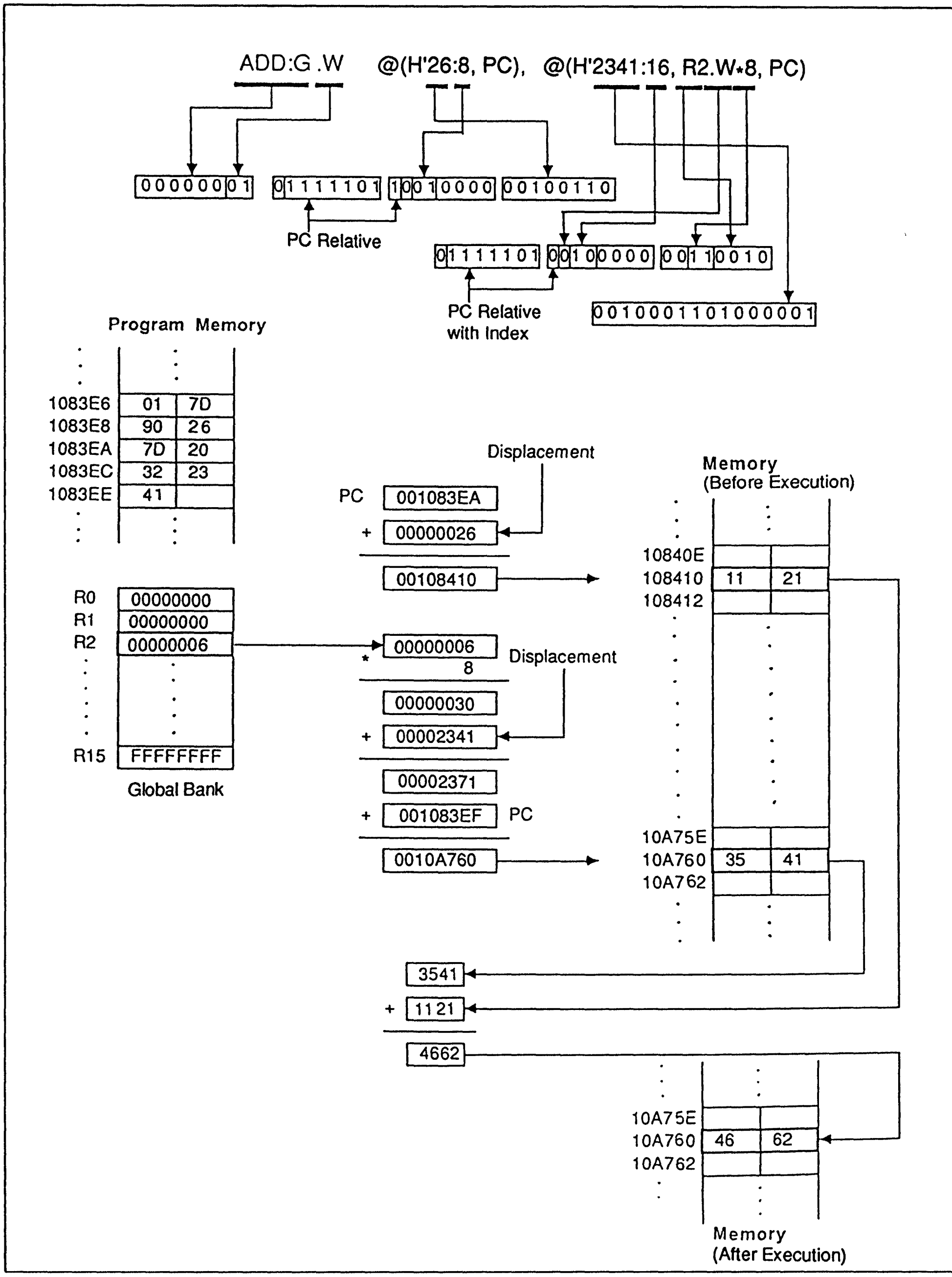


Figure 4-46. Instruction Execution Example in PC Relative/ PC Relative with Index Addressing Mode

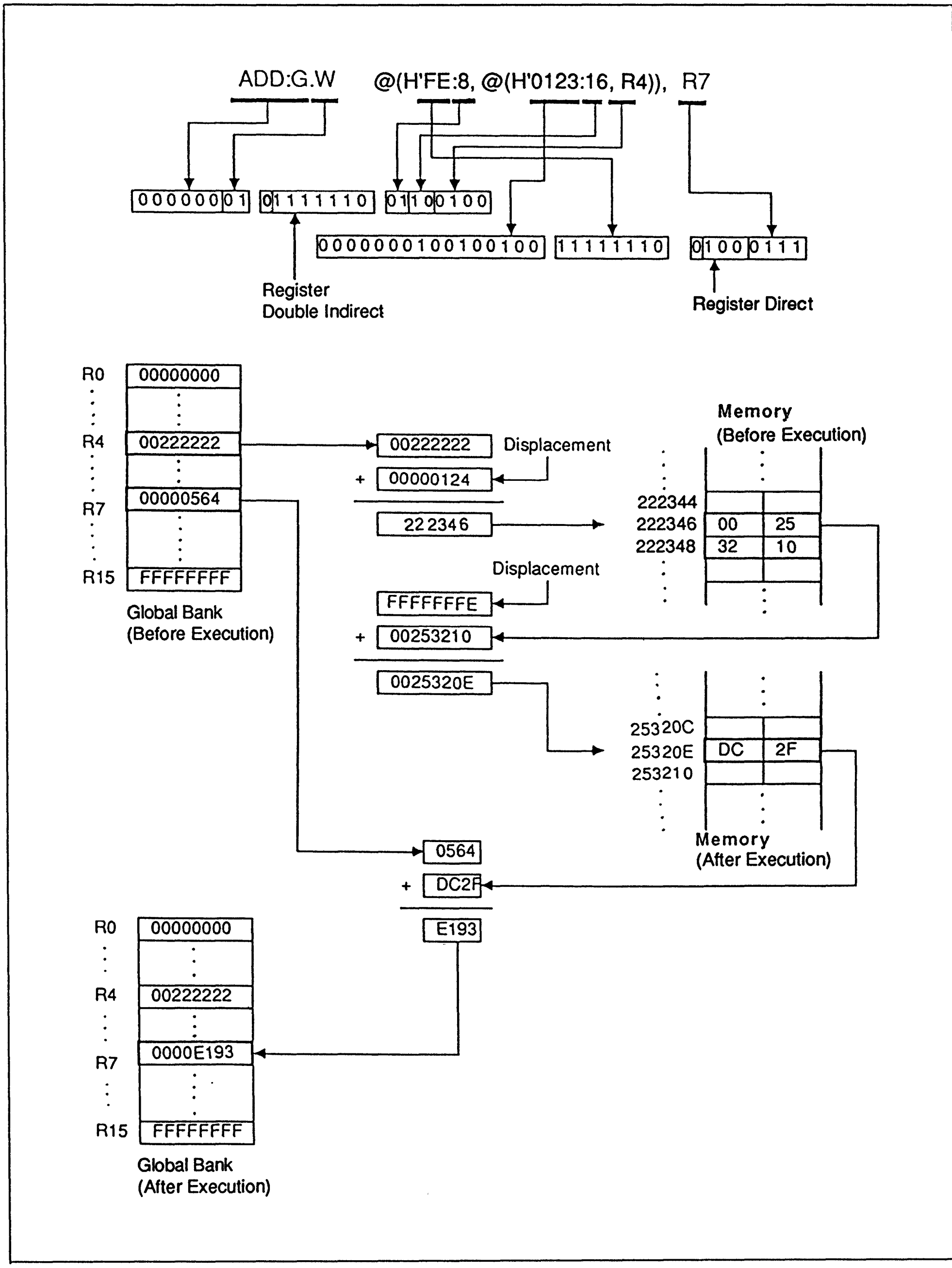


Figure 4-47. Instruction Execution Example in Register Double Indirect/ Register Direct Addressing Mode

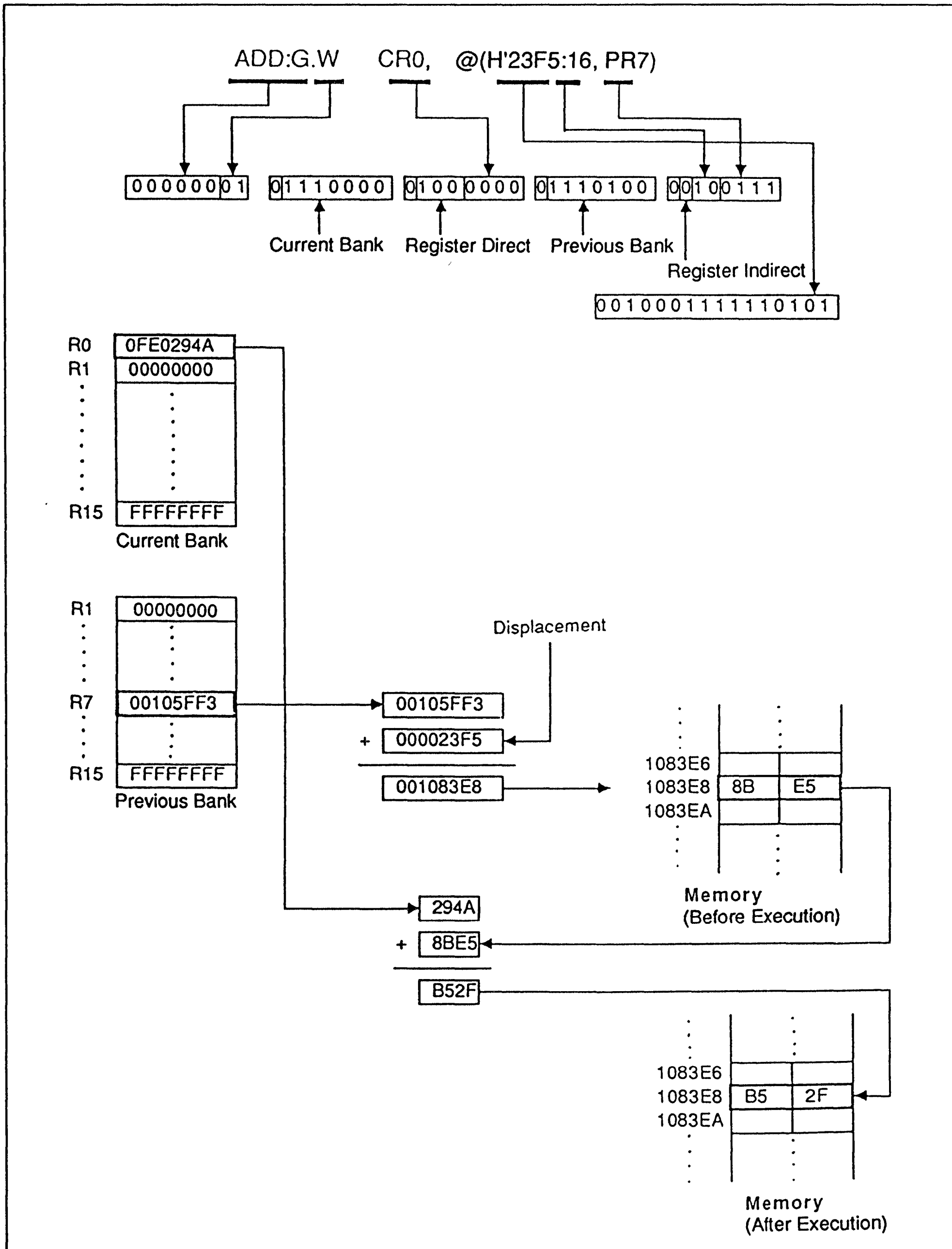


Figure 4-48. Instruction Execution Example in Current Bank Register Direct/ Previous Bank Register Indirect Addressing Mode

4.5.3 Instruction Set Summary

Each instruction's format and the opcode map (Figure 4-48) are described here. See "Section 16. Instruction Set" for details.

Upper 4 bits ↓ / Lower 4 bits →	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ADD : G				SUB : G				CMP : G				MOV : G			
1	ADD : Q				CLR				CMP : Q				MOV : Q			
2	ADD : R				SUB : R				CMP : R				MOV : R			
3	ADD : RQ				SUB : RQ				CMP : RQ				MOV : RQ			
4	ADDS				SUBS				CMPS				MOVS			
5	ADDX				SUBX				TST				MOVF			
6	SFT				SFT				BIT				BIT			
7	STM				LDM				MOVTPE				MOVFPE			
8	AND				XOR				OR				NEG			
9	NOT				STRING				BRA			JMP	NEGX			
A	BEQ			DADD	Bcc : G			DSUB	BSR			JSR	EXTU			DNEG
B	BNE			XCH	SCB			SET	RTD			RTS	EXTS			MOVA
C																
D	LINK			UNLK	BFEXT	BFINS	BFSCH	BFMOV								
E	MOVTP		MOVFP		CGBN				PGBN		SWAP		TAS		MUL	DIV
F	RESET	RTE	TRAPA	TRAP	RTR	SLEEP			ANDC	ORC	XORC	LDC	STC	ICBN	DCBN	NOP

Figure 4-49. Opcode Map

2-Byte Opcode Instructions: The leading opcode bytes specify the instructions in Table 4-9. See "Section 16. Instruction Set" for details.

Table 4-9. 2-Byte Opcode Instructions

Symbol in Opcode Map	Specified Instruction
SFT	SHAL, SHAR, SHLL, SHLR, ROTL, ROTR, ROTXL, ROTXR
BIT	BCLR, BNOT, BSET, BTST
STRING	SMOV, SSTR, SCMP, SSCH
MUL	MULXS, MULXU
DIV	DIVXS, DIVXU

CR Register Access Instructions: CR instructions such as ANDC, ORC, XORC, LDC and STC access CR registers other than PC and SSP by CR codes (Figure 4-50). Table 4-10 shows CR codes and their corresponding registers.

CR code

S/U	Sz	code

S/U	Function
0	Register accessible in user mode or supervisor mode
1	Register accessible in only supervisor mode

Sz	Function
00	Byte access
01	Word access
10	Long word access
11	Reserved

Figure 4-50. CR Code Format

Table 4-10. CR Codes and CR Registers

CR Code	CR Register
00100000	CCR
00000001	VBNR
01000000	CBNR
01000001	BSP
10000000	BMR
10000001	GBNR
10100000	SR
11000000	EBR
11000001	RBR
11000010	USP
11000011	IBR

4.5.4 Special Function Instructions

Bank Switching Instructions: The CGBN, PGBN, ICBN, and DCBN instructions support bank switching. They allow high speed subroutine calls and interrupt response for banks. These instructions are covered in "4.4 Register Banks".

String Transfer/Compare Instructions: The string transfer/compare instructions perform high-speed string data transfer and comparison with many conditions for variable comparisons by using a counter.

Bit Field Instruction: Bit field instructions perform bit field data extraction, insertion, "1" bit search, and block transfer at high speed, with the on-chip barrel shifter.

Figure 4-51 shows a block transfer performed by the BFMOV (move bit field data) instruction. BFMOV transfers variable length data to any bit field. Data length n and number of data m are specified by registers.

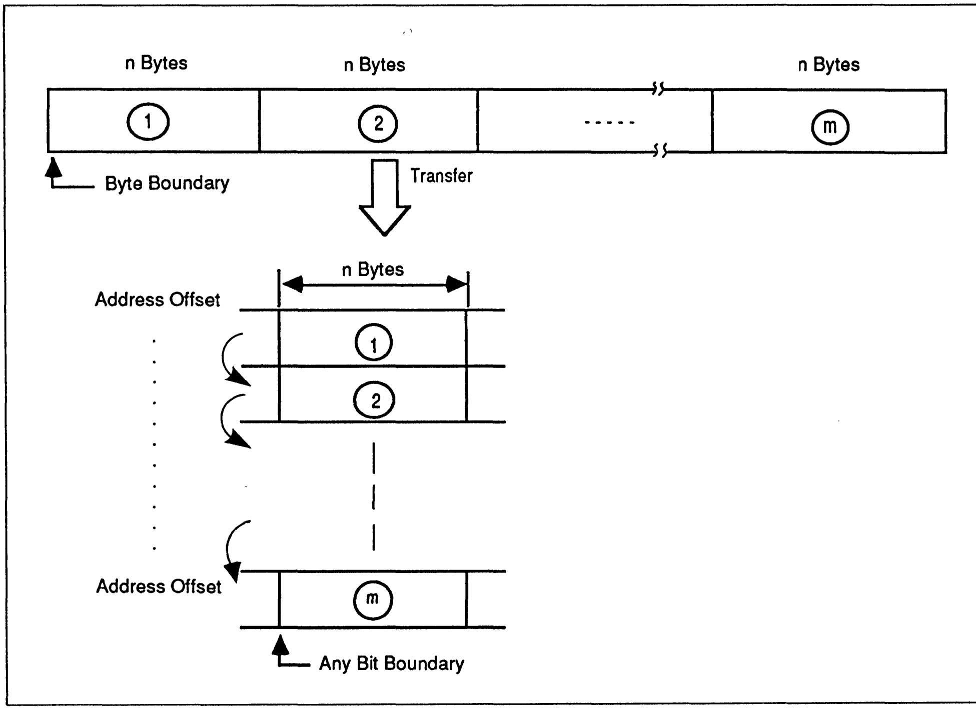


Figure 4-51. BFMOV Execution

SLEEP Instruction: Executing the SLEEP instruction puts the HD641016 in sleep or system stop mode. Sleep or system stop mode is selected by the SSTOP bit of BMR. In sleep mode, the CPU stops operation. In system stop mode, the CPU and the internal peripheral functions other than the DRAM refresh controller stop operation.

4.6 Processing States and Privilege Modes

4.6.1 Processing States

HD641016 has four processing states: normal processing, exception processing, halt processing, and low power consumption processing.

In the normal processing state, the CPU operates normally and executes instructions sequentially.

The exception processing state is associated with interrupts, traps, and some types of errors. It allows the CPU to handle unusual conditions quickly. See "4.6.3 Execution Processing State" for details.

The HD641016 is put into a halt state by a fatal memory access error such as a double bus error (a bus error which occurs during reset, bus error or access level violation exception processing). In the halt state, the HD641016 stops operating and ST2-ST0 indicate a halt state. Only an external reset can restart a halt processor.

The HD641016 has two low power consumption modes: sleep mode and system stop mode. In these modes, power consumption is reduced by stopping the CPU and on-chip I/O devices. See "Section 15. Low Power Consumption Modes" for details.

4.6.2 Privilege Modes

The HD641016 operates in one of two privilege modes: supervisor or user mode. The operating mode is determined by the S bit in SR. If S = 1, the CPU is in supervisor mode. If S = 0, it is in user mode.

Supervisor mode is a higher privilege mode than user mode. In supervisor mode, all instructions can be executed, and all control registers other than the PC can be accessed. However, in global mode, ring mode instructions and addressing modes cannot be used. In supervisor mode, the supervisor stack pointer SSP is the effective stack pointer.

User mode is the lower privilege mode. In user mode, instructions that have important system effects cannot be executed, and privileged control registers, BMR, GBNR, SR, EBR, RBR, SSP and IBR cannot be accessed. The user stack pointer USP is the effective stack pointer. See "4.3.13 R15 and the Stack Pointer" for details.

The privilege mode can be changed by changing the S bit in SR. Transition from supervisor to user mode can be made by bit manipulation of the S bit, or by the RTE instruction; however, transition from user to supervisor mode can only be made by TRAP instructions or an interrupt exception that stores the current state of the S bit, and sets the S bit by forcing the CPU into supervisor mode.

4.6.3 Exception Processing State

Exception Processing Steps: Exception processing takes place in four steps, with variations for different exception causes.

1. SR change: The CPU temporarily copies the SR to a temporary internal register. Then the S bit is set to 1 to set supervisor mode, and the T bit is cleared. For reset exception processing, the I2-I0 bits are set to 111. For interrupt exception processing, the interrupt priority level is copied into the I2-I0 bits and the IF bit is set.

2. Vector address generation: For external interrupt processing in external vector mode, the HD641016 fetches the interrupt vector number from an external device during the interrupt acknowledge cycle. For other exception processing, the HD641016 generates the vector number internally. The CPU then caluculates an exception vector address from the vector number.

3. Processor context stack: Current processor context is maintained, except during reset exception processing.

4. New PC generation: The HD641016 obtains the start address of the exception routine from the exception vector address and starts exception processing.

Exception Types: Exceptions can be generated either internally or externally. External exceptions are reset, bus error, DMA bus error, interrupts, and acknowledge error. Internal exceptions are access level violation, DMA access level violation, chip select controller bus error, trace, interrupts by on-chip I/O devices, illegal instructions, unimplemented instructions, privilege violation, bank mode violation, and the instruction TRAP. Table 4-11 shows these levels, exception sampling timing, and when exception processing starts.

Table 4-11. Exception Types

Priority	Exception	Source	Sampling Timing	Start of Processing
1 (Highest)	Reset	External	$\overline{RES}$ and $\overline{BRTRY}$ are asserted low	Exception processing begins immediately.
2	Access level violation	Internal	When address is output	Exception processing begins after the current bus cycle has completed.
3	Bus error	Internal/ external	Falling edge of T_2 or T_w state prior to T_3 state	
4	DMA access level violation	Internal	When DMA address is output	DMA operation immediately stops after the current DMA cycle has completed. The CPU then executes the suspended instruction and begins exception processing.
5	DMA bus error	Internal/ external	Falling edge of T_2 or T_w state prior to T_3 state	
6	Trace	Internal	End of instruction cycle	Exception processing begins after completing current instruction.
7	Interrupt	Internal/ external	End of instruction cycle or exception processing	Exception processing begins after completing current instruction or exception processing.
8	Illegal instruction	Internal	During instruction cycle	During instruction cycle
9	Unimplemented instruction	Internal		

Table 4-11. Exception Types (cont.)

Priority	Exception	Source	Sampling Timing	Start of Processing
10	Privilege violation	Internal	During instruction cycle	During instruction cycle
11	Bank mode violation	Internal		
12 (Lowest)	Instruction trap TRAPA, TRAP, Division by 0 with DIVXS or DIVXU	Internal	During instruction cycle	Exception processing begins in the same way as for normal instruction execution.

4.6.4 Exception Processing

The following paragraphs describe in detail each type of exception processing.

Reset: Reset is the highest priority exception. If $\overline{\text{RES}}$ and $\overline{\text{BRTRY}}$ are asserted low together for at least 12 clock cycles and then $\overline{\text{RES}}$ or $\overline{\text{BRTRY}}$ is negated high, the HD641016 will always start reset exception processing. The reset exception vector numbers are H'00 and H'01.

On-chip I/O devices and some CPU registers (Table 4-12) are initialized at reset. Then the CPU gets 8 bytes of data from the reset vector location determined by the vector number internally generated. The CPU loads the upper 4 bytes into the SSP and the lower 4 bytes into the PC. The CPU then restarts instruction execution at the address pointed to by the PC. See Figure 4-52.

When $\overline{\text{RES}}$, $\overline{\text{BRTRY}}$, and $\overline{\text{NMI}}$ are asserted low for at least 12 clock cycles, only the DRAM refresh controller continues operating; reset exception processing begins when these three pins are pulled high. See "4.11 Reset" and "Section 13. DRAM Refresh Controller" for details.

The reset exception should not be confused with the RESET instruction, which resets external peripherals by asserting $\overline{\text{RES}}$ low for 256 clock cycles. Since it does not affect any internal conditions, the CPU executes the next instruction following the RESET instruction.

Note that the HD641016 accesses memory for instruction prefetch after it gets the 8-byte data from the reset vector address. Therefore, if a bus error occurs during reset exception processing, the HD641016 is halted. Accordingly to complete the reset exception processing normally, $\overline{\text{BRTRY}}$ must be pulled high during the 8 bytes of memory access following the 8-byte data access from the reset vector access.

Table 4-12. Control Registers at Reset

CPU Reg.	**Initial Value (Binary)** **Bit** 31 23 15 7 0
SR	0-10-111---*****
CCR	-----------*****
EBR	********000000000000000000000000
RBR	********11111111111110**********
IBR	********11111111****************
CBNR	00000000000000000000000000000000
BMR	10100100
GBNR	****0000
VBNR	*****000

Note: 0: Cleared *: Undefined
1: Set -: Reserved

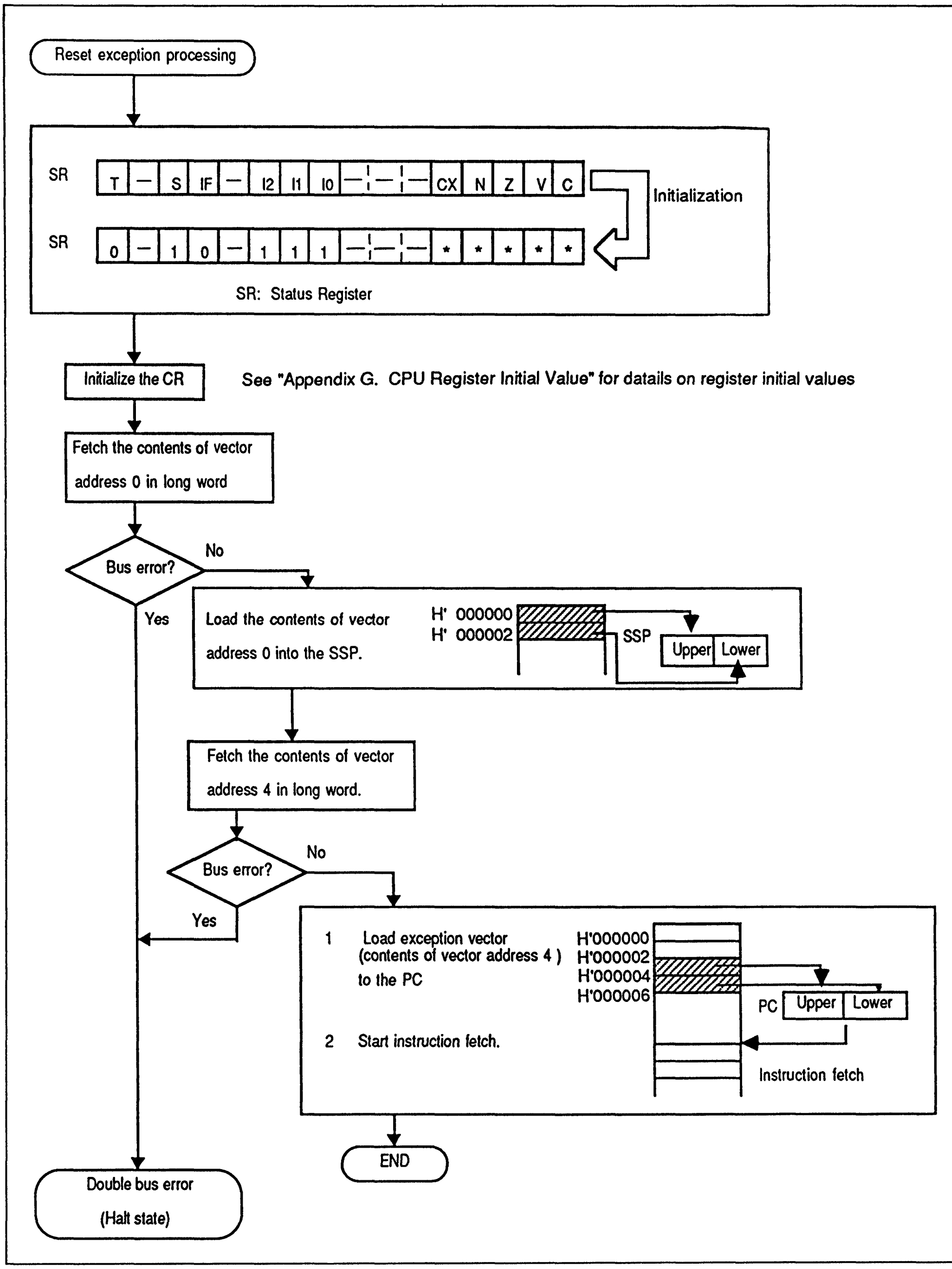


Figure 4-52. Reset Exception Processing Sequence

Bus Error: Bus error exception processing is caused by either $\overline{\text{BRTRY}}$ or illegal chip select area access. The bus error exception vector number is H'02.

For a bus error exception caused by $\overline{\text{BRTRY}}$, the CPU has two processing modes: bus error mode and bus retry mode. The bus error exception mode is selected by the BRTE bit of BMR. In bus error mode (BRTE = 0), the CPU begins bus error exception processing immediately after $\overline{\text{BRTRY}}$ is asserted low. In bus retry mode (BRTE = 1), the CPU retries the bus cycle up to three times. However, if $\overline{\text{BRTRY}}$ is still asserted, bus error exception processing begins. Figure 4-53 shows the bus error retry processing flow. Figures 4-54 and 4-55 show bus retry and bus error timings. Figure 4-56 shows bus retry, bus error and access level violation exception processing flow.

For a bus error exception caused by a chip select area access error, the CPU begins bus error exception processing after the bus cycle has completed. See "Section 11. Chip Select Controller" for details.

Note that the HD641016 generates a double bus error and its operation is halted if a bus error occurs during reset, bus error, or access level violation exception processing. A double bus error is cancelled only by reset. In addition, if a bus error occurs during the acknowledge cycle of interrupt exception processing, an acknowledge error interrupt occurs.

Since a bus error exception begins processing after the bus cycle generating the bus error has completed, the read or write operation in this bus cycle is actually performed. However, if a bus error occurs during an instruction cycle requiring multiple bus cycles, bus error exception processing begins immediately without completing the instruction cycle.

The stack configuration for a bus error exception is the same as that for access level violation. See "4.6.7 Stack Configuration for Exception Processing" for details.

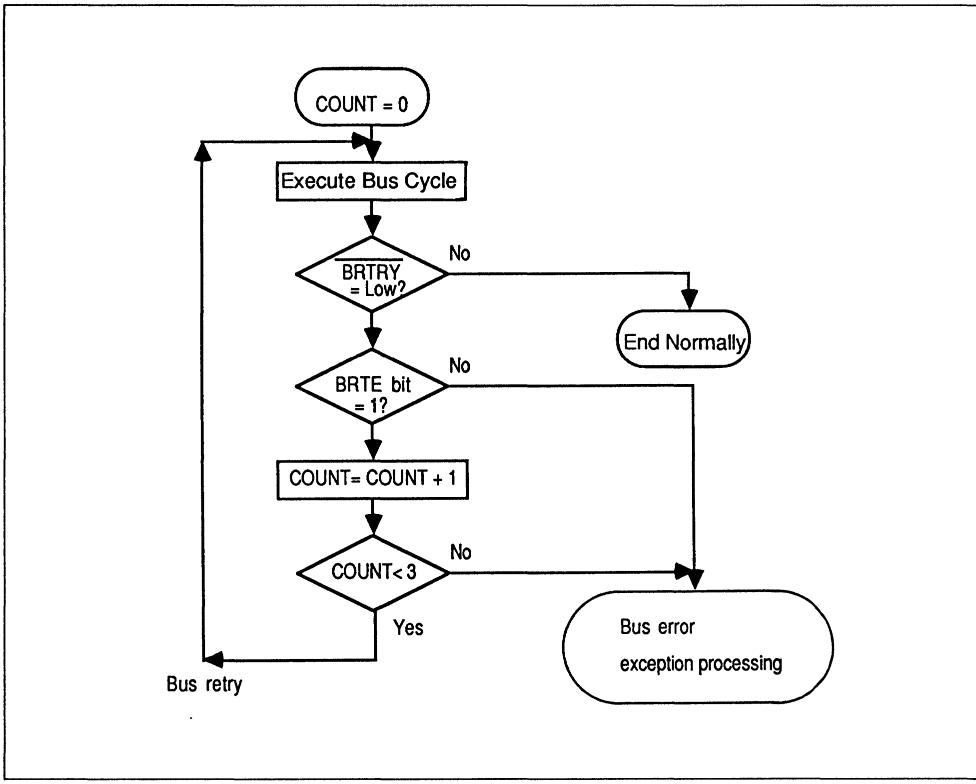


Figure 4-53. Bus Error Retry Flow

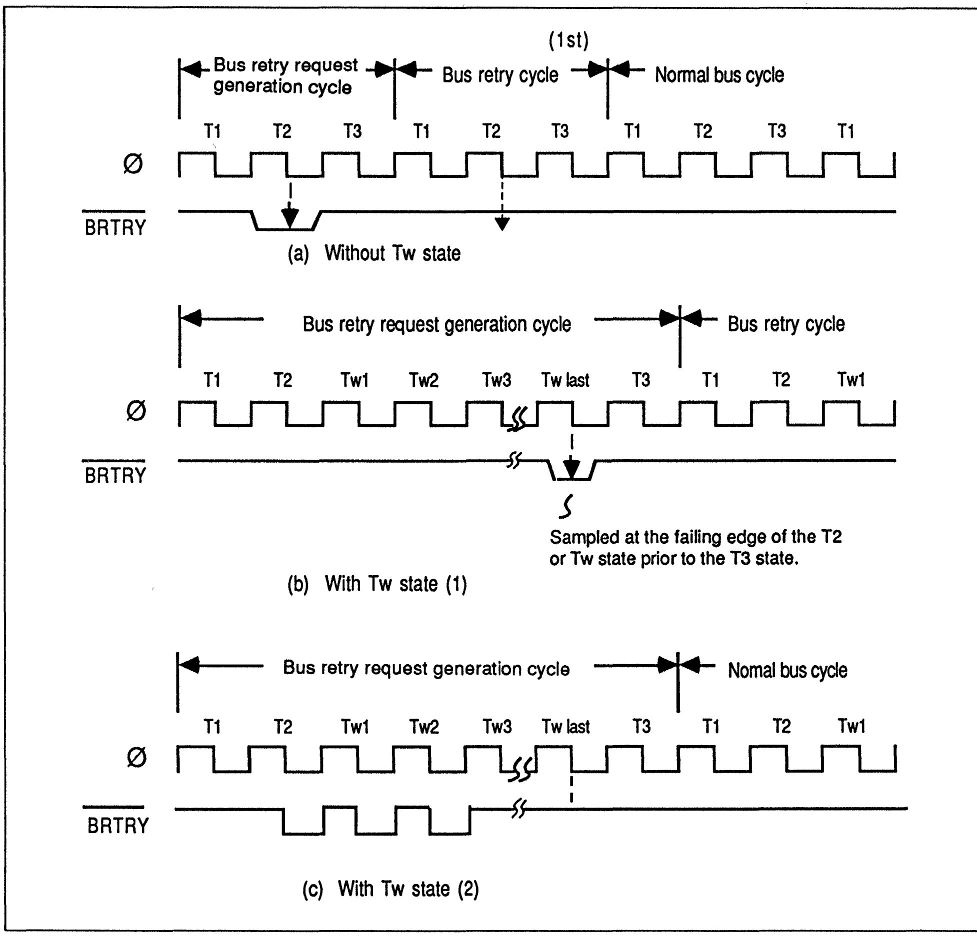


Figure 4-54. Bus Retry Timing

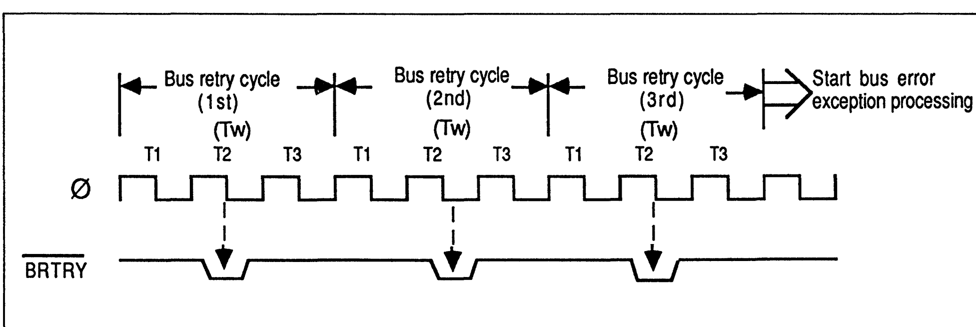


Figure 4-55. Bus Retry to Bus Error Timing

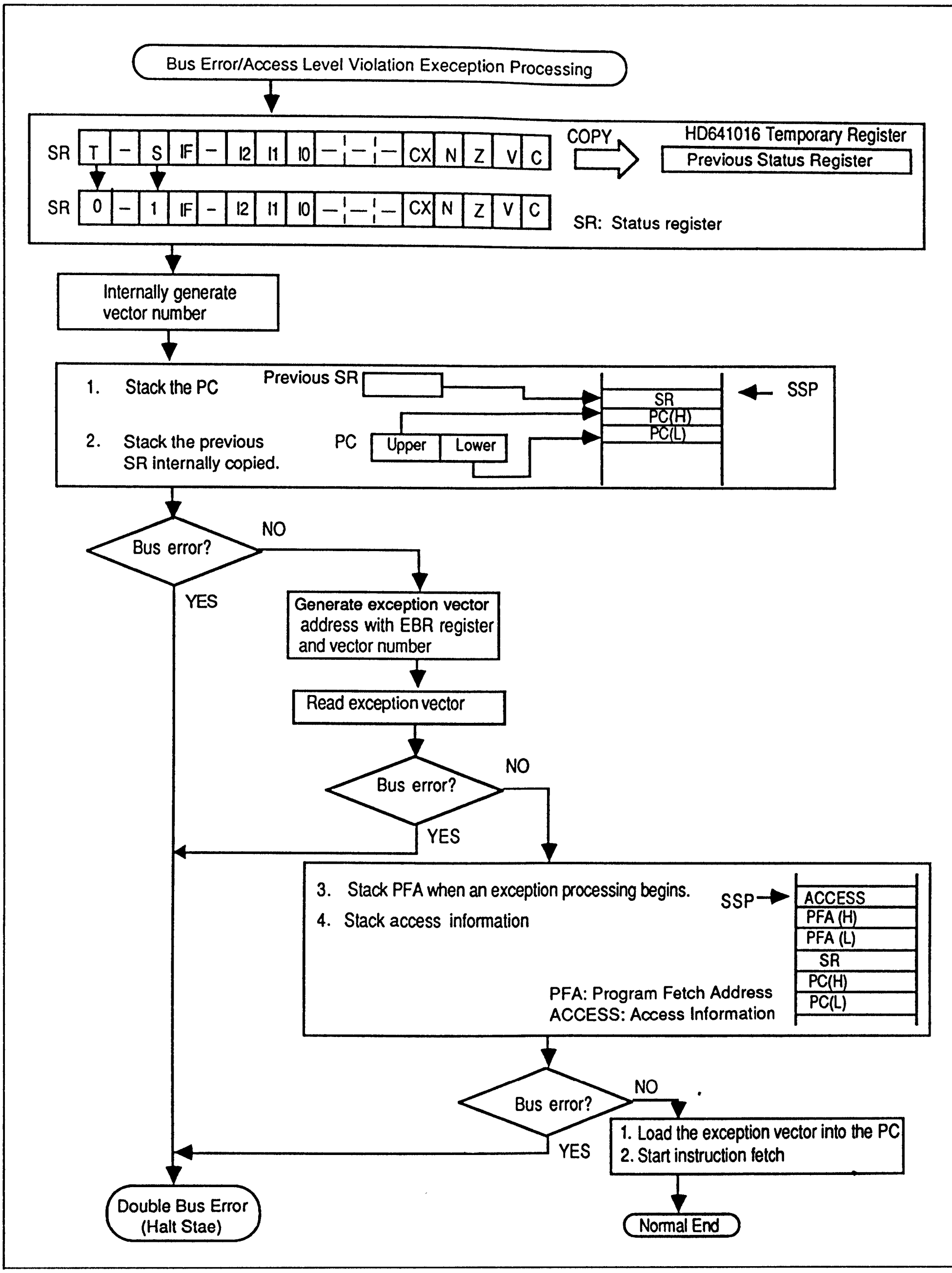


Figure 4-56. Bus Retry/Bus Error and Access Level Violation Exception Processing

Trace: If the T bit of SR is set, the HD641016 generates a trace exception whenever an instruction completes execution. This exception is valid from the next instruction after the T bit is set. The trace exception triggers the system's trace routine (Figure 4-58). The trace exception vector number is H'09. See "4.6.7 Stack Configuration for Exception Processing" for details on the stacked information.

Note that if the SLEEP instruction is executed while T = 1, the CPU enters sleep or system stop mode for one clock cycle and then performs trace exception processing.

Illegal Instruction Exception: An attempt to execute an illegal instruction causes the HD641016 to start illegal instruction exception processing instead of executing the illegal instruction (Figure 4-58). The illegal instruction exception vector number is H'04.

In illegal instruction exception processing, the PC stacked varies depending on the instruction field containing the illegal pattern. If an opcode contains an illegal pattern, the PC points to the opcode address. If the second opcode of the 2-byte instruction includes an illegal pattern, the PC points to the second opcode address. In addition, if the operand contains an illegal pattern, the PC points to the address following the illegal operand address. See "4.6.7 Stack Configuration for Exception Processing" for details.

Unimplemented Instruction Exception: When the HD641016 attempts to execute an instruction whose opcode is H'C0 to H'C7 or H'C8 to H'CF, it causes an unimplemented instruction exception (Figure 4-58). This allows the system to emulate these instructions in software. The H'C0-H'C7 instruction vector number is H'10, and the H'C8-H'CF instruction vector number is H'11.

See "4.6.7 Stack Configuration for Exception Processing" for details on the stacked information.

Privilege Violation: The HD641016 operates in one of two privilege modes: supervisor or user mode. Privileged instructions cannot be executed in user mode. When the HD641016 tries to execute a privileged instruction in user mode it causes a privilege violation exception (Figure 4-58). The privilege violation exception vector number is H'08.

Privileged instructions are:

- STC, LDC, ANDC, XORC, and ORC instructions specifying the BMR, GBNR, SR, EBR, RBR, USP, or IBR register
- CGBN, PGBN

- RTE
- RESET
- SLEEP

In privilege violation exception processing, the stacked PC points to the first byte of the instruction. See "4.6.7 Stack Configuration for Exception Processing" for details.

Trap Exception: When the MPU executes a TRAPA instruction, TRAP instruction, or DIVXS or DIVXU division by zero, it can cause a trap exception (Figure 4-58).

The TRAPA instruction always causes a trap exception. The vector number is H'32-H'47, depending on the lower 4 bits of the second opcode.

The TRAP instruction causes a trap exception if the condition (cc) is true. The vector number is H'07.

Division by zero with the DIVXS or DIVXU instruction always causes a trap exception, vector number H'05.

See "4.6.7 Stack Configuration for Exception Processing" for details on the stacked information.

Access Level Violation: If the MPU accesses an illegal area in user mode, it causes an access level violation exception (Figure 4-56). The vector number is H'12.

Illegal areas in user mode are defined as follows:

- Internal I/O area
- Internal RAM area which is protected from access in user mode under the control of the RAMALV bit in BMR
- Chip select area which is protected from access in user mode by the chip select controller

Since access level violation exception processing begins after the bus cycle generating the access level violation has completed, the read or write operation in this bus cycle is actually performed. However, if an access level violation occurs during an instruction cycle requiring multiple bus cycles, the access level violation exception processing begins immediately without completing the instruction cycle. See " Section 8. DMA Controller" for details.

See "4.6.7 Stack Configuration for Exception Processing" for details on the stacked information.

Bank Mode Violation: The HD641016 has two register handling modes, ring and global. In ring mode, ring registers can be used, but in global mode, they cannot. If an addressing mode or instruction used only for ring mode, such as the ICBN or DCBN instruction, is specified in global mode, it causes bank mode violation exception (Figure 4-58). The bank mode violation vector number is H'13.

See "4.6.7 Stack Configuration for Exception Processing" for details on the stacked information.

DMA Bus Error: When $\overline{\text{BRTRY}}$ is pulled low during on-chip DMAC operation, the DMAC stops after the DMA cycle has completed. DMA bus error exception processing (Figure 4-58) then begins at the end of the suspended instruction cycle. During DMA bus error exception processing, the data access address and access state are not stacked since the user can obtain the required information from the DMAC registers. The DMA bus error exception vector number is H'18.

See "4.6.7 Stack Configuration for Exception Processing" for details on the stacked information.
See "Section 8. DMA Controller" for details on the DMAC BRTRY sampling timing.

DMA Access Level Violation: If the MPU accesses an illegal area when the DMAC is in user area access mode, it causes a DMAC access level violation exception (Figure 4-58). The DMA access level violation vector number is H'19.

Illegal areas in this mode are defined as follows:

- Internal I/O
- Internal RAM space which is protected from access in user mode under the control of the RAMALV bit in BMR
- Chip select area which is protected from access in user mode by the chip select controller

If the DMAC access level violation exception occurs, the DMAC stops operating after completing the current DMA cycle and the CPU executes the suspended instruction. DMAC access level violation exception processing then begins.

The data access address and access state are not stacked during the exception processing since the user can obtain the required information from the DMAC registers. See "4.6.7 Stack Configuration for Exception Processing" for details on the stacked information.
See "Section 8. DMA Controller" for details.

Interrupt: The HD641016 has 3 external and 22 internal interrupt sources at three priority levels which will be discussed further in the following sections (see Figure 4-59). See "4.6.7 Stack Configuration for Exception Processing" for details on the stacked information. Refer to "4.7 Interrupt Controller" for details on interrupt priority.

The 3 external interrupt sources are $\overline{\text{NMI}}$, $\overline{\text{IRQ0}}$, and $\overline{\text{IRQ1}}$. The $\overline{\text{IACK}}$ pin is the interrupt acknowledge pin. The interrupt priority register (IPR) controls the interrupt priority. The interrupt request control register (ICR) and the interrupt priority registers 2-0 (IPR2-IPR0) control the interrupt trigger mode, mask, and acknowledge mode. In non-acknowledge mode, the NMI, IRQ0, and IRQ1 interrupt exception vector numbers are H'31, H'30, and H'29, respectively.

The 22 internal interrupts have fixed interrupt vector numbers. Therefore, an interrupt source can be identified by the vector number. This allows simple interrupt processing. Internal interrupt sources and vector numbers are listed in Table 4-14.

IF (Interrupt Status Flag): The IF bit in the SR register is set to 1 when an interrupt is accepted and its value before the interrupt is restored after the interrupt service routine has completed.

In addition, the IF bit can also be used to control interrupt processing in a real-time OS, as follows. Sub-IRQ processing is performed in supervisor state and main IRQ processing in user state (Figure 4-57 (a)). If multiple interrupts occur simultaneously, IRQ processing is quickly performed in supervisor state and the task dispatcher determines which main IRQ processing is performed first in user state (Figure 4-57 (b)).

In this sequence, the HD641016 uses the IF bit value in the stacked SR for the task switch. When the HD641016 detects IF = 0 (point A), the task dispatcher gains task control after the IRQ processing has completed. This function of the IF flag effectively supports high speed IRQ processing in a real-time OS.

However, if the HD641016 cannot use the IF bit for task switching, the HD641016 detects an interrupt through the S bit of SR at point B and IRQ processing is delayed by the period C shown in Figure 4-57 (b).

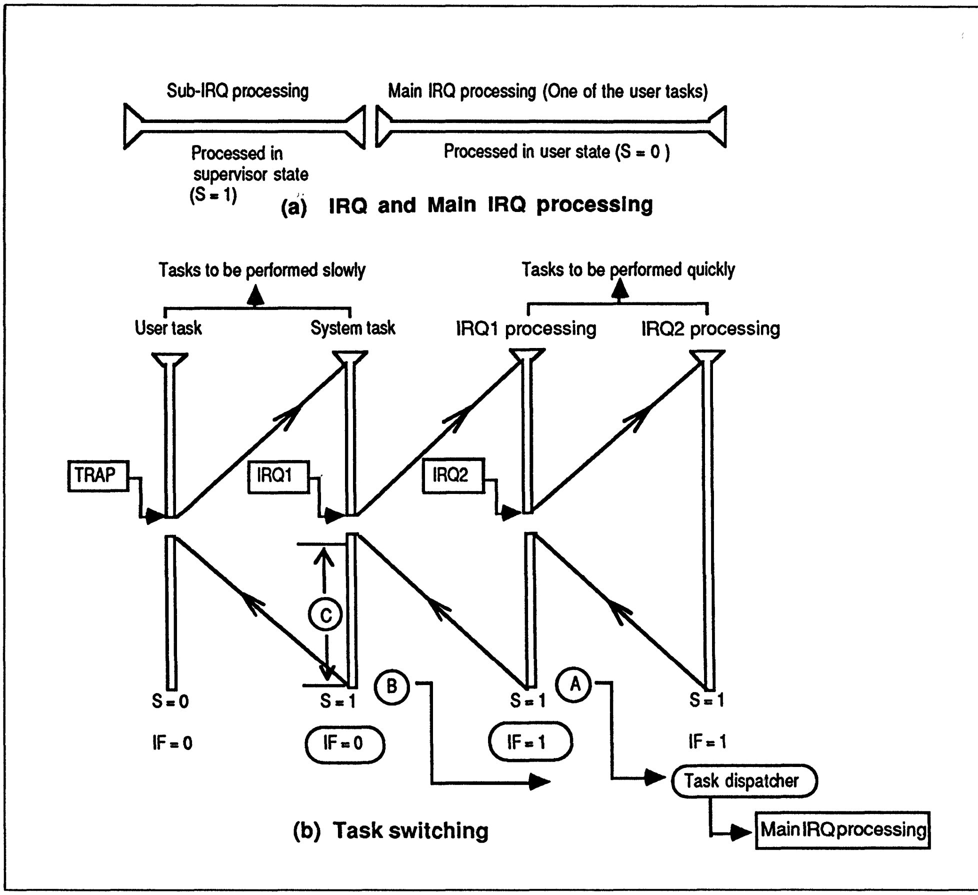


Figure 4-57. IF Flag Function

Acknowledge Error: An acknowledge error exception occurs, if a bus error caused by BRTRY occurs during single or double acknowledge cycle of an external interrupt. The CPU then stops the external interrupt vector access and begins the acknowledge error interrupt exception processing by internally generating the exception vector number H'24. See Figure 4-59.

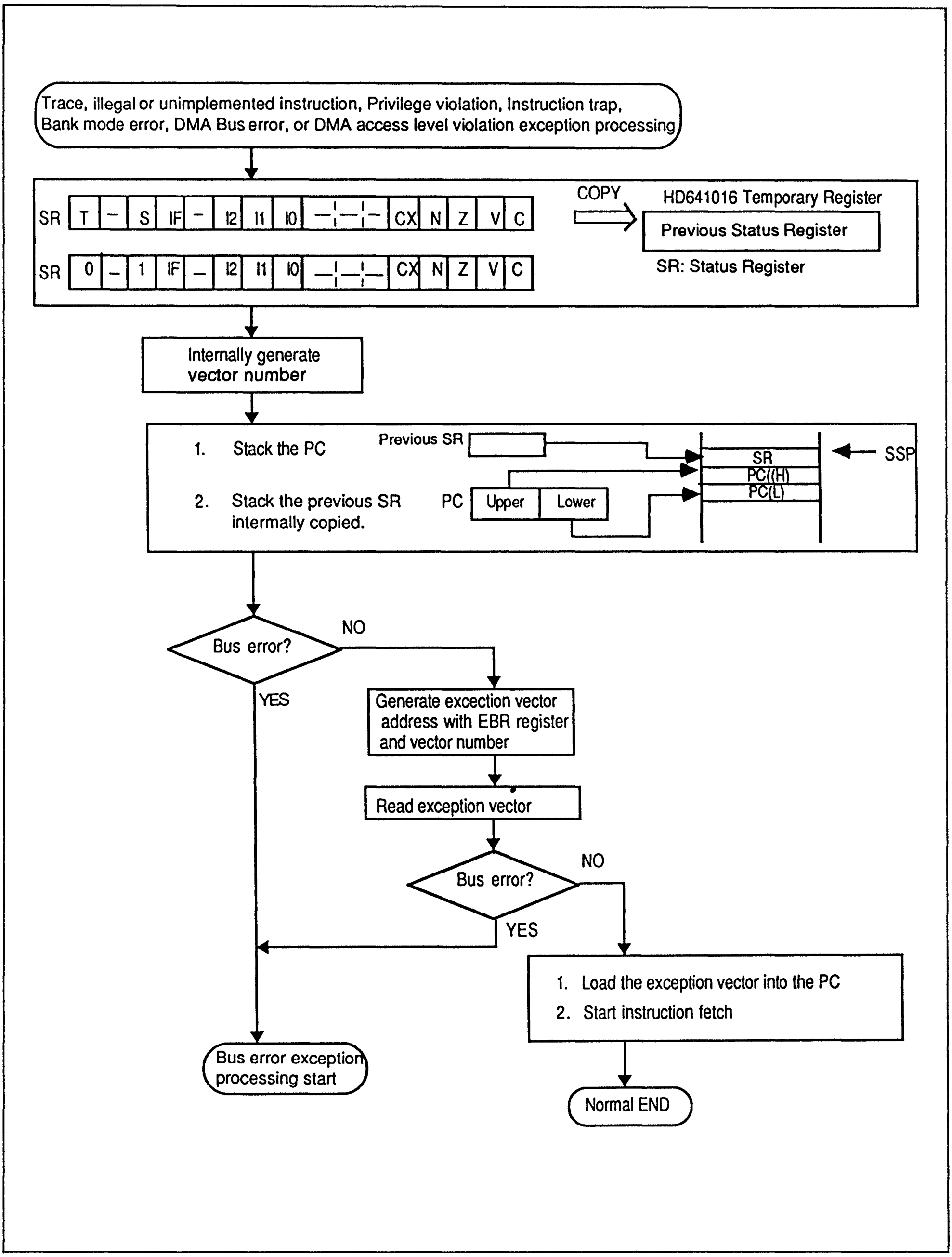


Figure 4-58. Trace, Illegal or Unimplemented Instruction, Trap, Bank Mode Error, DMA Bus Error, DMA Access Level Violation, or Privilege Violation Exception Processing Sequence

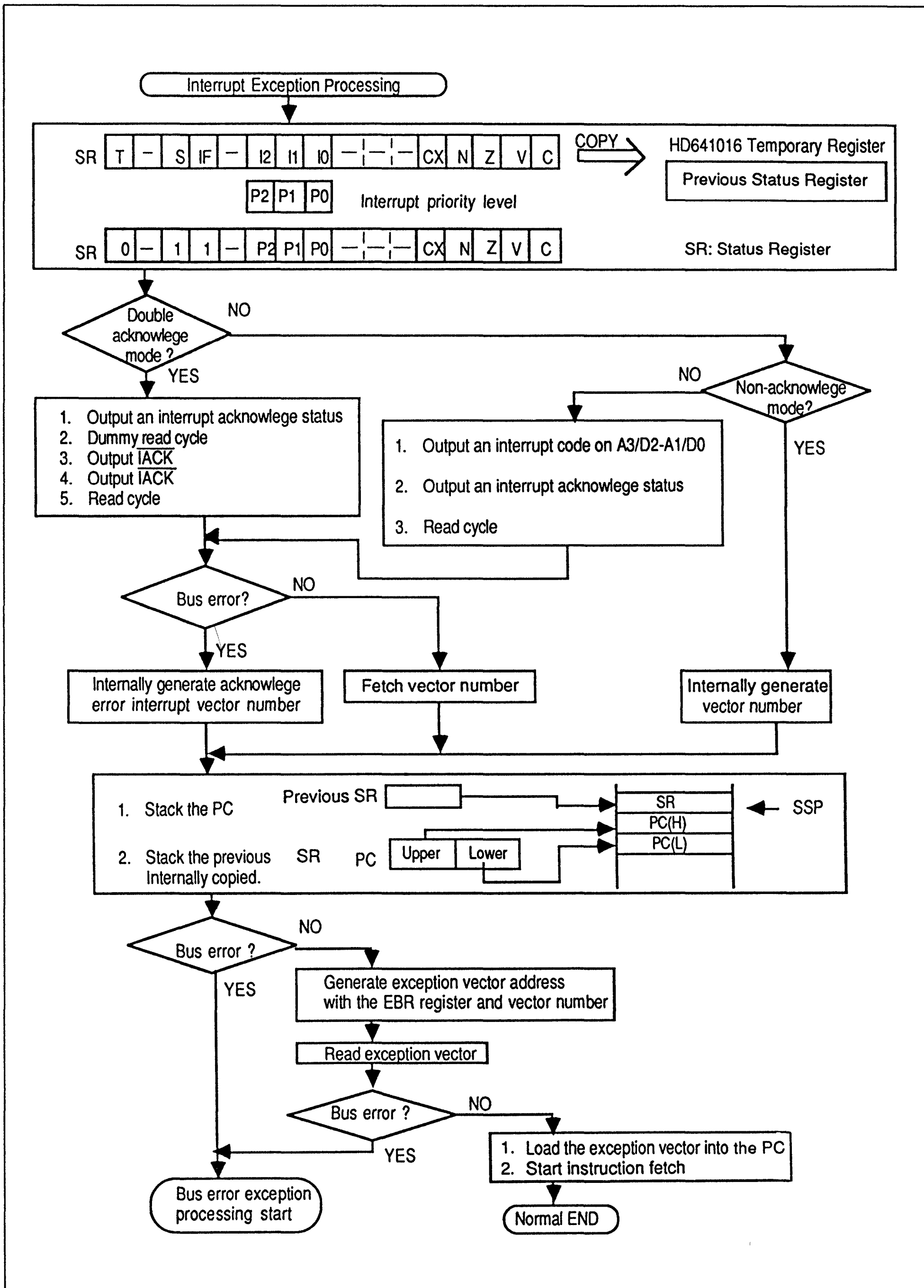


Figure 4-59. Interrupt and Acknowledge Error Exception Processing Sequence

4.6.5 Exception Processing Conflicts

When more than two exceptions occur simultaneously, the CPU processes the highest priority exception first (Table 4-11). For example, if an interrupt exception and a trace exception occur at the same time during an instruction cycle while T = 1, the MPU processes the trace exception first, after which it processes the pending interrupt exception. However, note that although trace exception processing is executed before interrupt exception processing, the interrupt routine will actually be executed before the trace routine. See Figure 4-60.

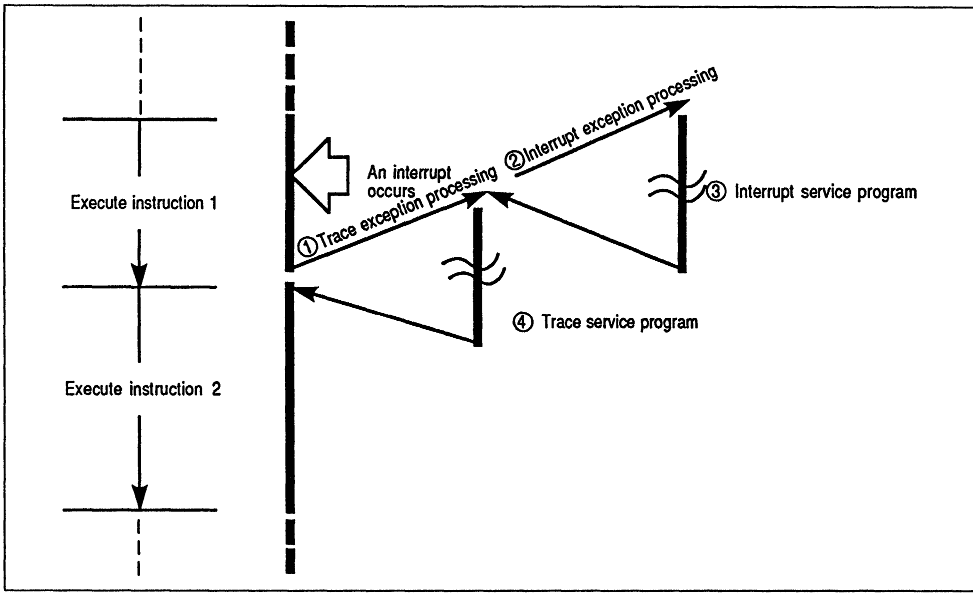


Figure 4-60. TRAP and Interrupt Exception Processing Flow

4.6.6 Exception Vectors

Each exception vector is obtained by adding the 32-bit EBR to the 8-bit vector number. The exception vector is internally generated by multiplying the vector number by 4. See Figure 4-61. The CPU then fetches the 4-byte start address of the exception processing routine from the vector location.

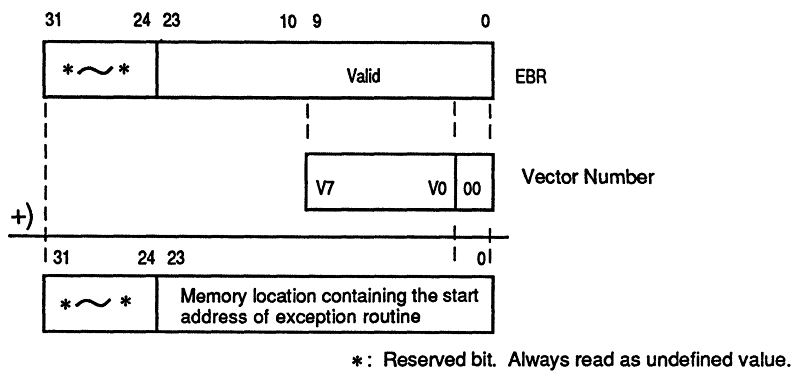


Figure 4-61. Exception Vector Calculation

8-bit vector numbers are determined as follows.

External Interrupts NMI, IRQ0, and IRQ1: In single acknowledge or double acknowledge mode, an external device transfers the 8-bit vector number to the HD641016 through data bus A8/D7 - A1/D0. In non-acknowledge mode, the HD641016 internally generates the vector number (Table 4-14) by the same method as for exceptions other than external interrupts. See "4.7 Interrupt Controller" for details.

Exception Other Than External Interrupts: The HD641016 internally generates the vector number. See Tables 4-13 and 4-14 for details.

Table 4-13. Exception Vector Table

Exception	Exception Vector Number (Decimal)	(Hexadecimal)	Offset from EBR value
Reset: SSP initial value	0	H'00	H'000
Reset: PC initial value	1	H'01	H'004
Bus error	2	H'02	H'008
Reserved	3	H'03	H'00C
Illegal instruction	4	H'04	H'010
Division by Zero	5	H'05	H'014
Reserved	6	H'06	H'018
TRAP instruction	7	H'07	H'01C
Privilege violation	8	H'08	H'020
Trace	9	H'09	H'024
H'C0-H'C7 unimplemented instruction	10	H'0A	H'028
H'C8-H'CF unimplemented instruction	11	H'0B	H'02C
Access level violation	12	H'0C	H'030
Bank mode violation	13	H'0D	H'034
Reserved	14	H'0E	H'038
User vector number	15	H'0F	H'03C
Reserved	16-17	H'10-H'11	H'040-H'044
DMA bus error	18	H'12	H'048
DMA access level violation	19	H'13	H'04C
Reserved	20-23	H'14-H'17	H'050-H'05C
Acknowledge error	24	H'18	H'060
Reserved	25-28	H'19-H'1C	H'064-H'070
External interrupt IRQ 1	29	H'1D	H'074
External interrupt IRQ 0	30	H'1E	H'078

Table 4-13. Exception Vector Table (cont.)

Exception	Exception Vector Number (Decimal)	(Hexadecimal)	Offset from EBR value
External interrupt NMI	31	H'1F	H'07C
TRAPA instruction (#0 - #15)	32-47	H'20-H'2F	H'080-H'0BC
Reserved	48-63	H'30-H'3F	H'0C0-H'0FC
Internal interrupt (See Table 4-13.)	64-90	H'40-H'5A	H'100-H'168
Reserved	91-127	H'5B-H'7F	H'16C-H'1FC
User vector number	128-255	H'80-H'FF	H'200-H'3FC

Note: Reset vectors are stored in supervisor program area.
Other vectors are stored in supervisor data area.

Table 4-14. Internal Interrupt Vector Numbers

On-chip I/O	Internal Interrupt	Interrupt Vector Number Decimal	Interrupt Vector Number Hexadecimal	Offset from EBR value
	Reserved	64	H'40	H'100
	Reserved	65	H'41	H'104
TMR 1	Count match	66	H'42	H'108
	Measurement end	67	H'43	H'10C
	Overflow	68	H'44	H'110
TMR 2	Count match	69	H'45	H'114
	Measurement end	70	H'46	H'118
	Overflow	71	H'47	H'11C
	Reserved	72	H'48	H'120
	Reserved	73	H'49	H'124
	Reserved	74	H'4A	H'128
DMAC 0	Block transfer end	75	H'4B	H'12C
	DMA transfer end	76	H'4C	H'130
DMAC 1	Block transfer end	77	H'4D	H'134
	DMA transfer end	78	H'4E	H'138
DMAC 2	Block transfer end	79	H'4F	H'13C
	DMA transfer end	80	H'50	H'140
DMAC 3	Block transfer end	81	H'51	H'144
	DMA transfer end	82	H'52	H'148
ASCI0	RX ready	83	H'53	H'14C
	TX ready	84	H'54	H'150
	RX interrupt	85	H'55	H'154
	TX interrupt	86	H'56	H'158
ASCI1	RX ready	87	H'57	H'15C
	TX ready	88	H'58	H'160
	RX interrupt	89	H'59	H'164
	TX interrupt	90	H'5A	H'168

4.6.7 Stack Configuration for Exception Processing

Exceptions Other than Bus Error or Access Level Violation: For exceptions other than bus error or access level violations, long word PC and word SR are stacked as shown in Figure 4-62.

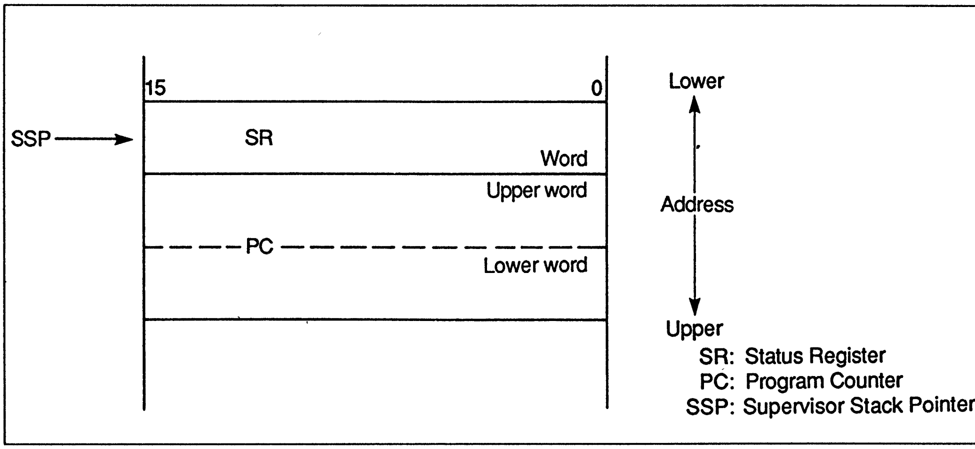


Figure 4-62. Stack Configuration for Exceptions Other Than Bus Error and Access Level Violation Exceptions

Bus Error and Access Level Violation: For bus error or access level violations, the PC, SR, program fetch address, and access information are stacked as shown in Figure 4-63.

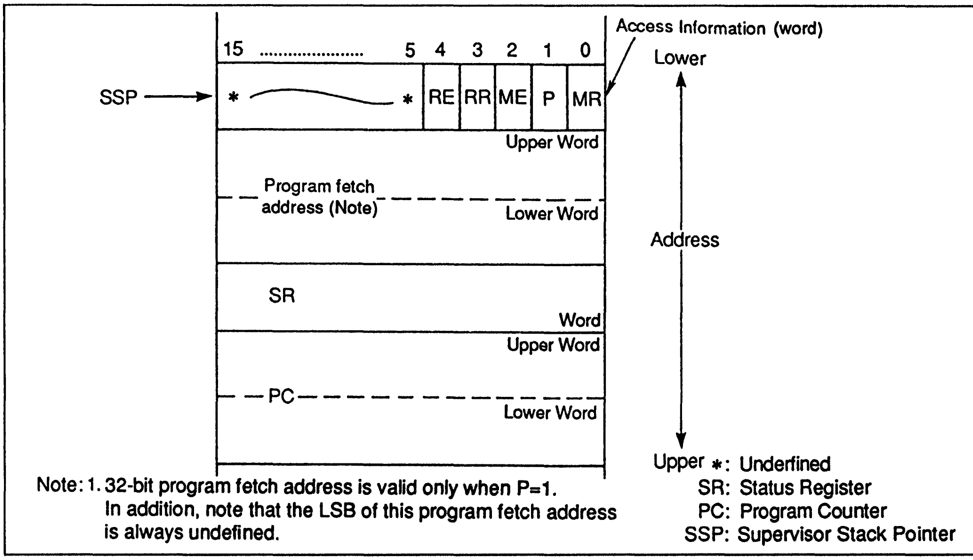


Figure 4-63. Stack Configuration for Bus Error and Access Level Violation Exceptions

The following explains each bit function in the above access information (Figure 4-64). Note that only the lower 5 bits of the access information are valid. The upper 11 bits are undefined.

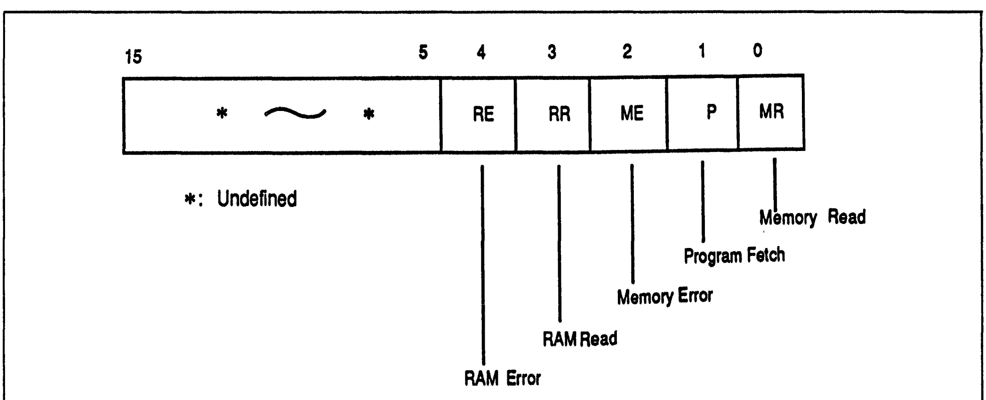


Figure 4-64. Access Information

RAM Error (RE): RE = 1 indicates that the access level violation occurred during the internal RAM access cycle. RE = 0 indicates that the exception occurred during a bus cycle other than the internal RAM access cycle.

RAM Read (RR): RR is valid only when RE = 1. RR = 1 indicates that the access level violation exception occurred during the internal RAM read cycle. RR = 0 indicates that the exception occurred during the internal RAM write cycle. Note that RR is undefined during an internal RAM boundary access (Figure 5-7).

Memory Error (ME): ME = 1 indicates that a bus error or access level violation exception occurred during an internal I/O or external memory access cycle. In addition, ME is set to 1 during the internal RAM boundary access (Figure 5-7). ME = 0 indicates that the exception occurred during a bus cycle other than an internal I/O or external memory access cycle.

Program Fetch (P): P is valid only when ME = 1. P = 1 indicates that a bus error or access level violation exception occurred during a program fetch cycle. In addition, the program fetch address on the stack is valid only when the P bit is set to 1. P = 0 indicates that the exception occurred during a data fetch cycle.

Memory Read (MR): MR is valid only when ME = 1. MR = 1 indicates that a bus error or access level violation exception processing occurred during an internal I/O or external memory access read cycle. MR = 0 indicates that the exception occurred during a write cycle.

Table 4-15 summarizes the access information bit values and their corresponding information.

Table 4-15. Access Information

RE	RR	ME	P	MR	Access Information
0	*	0	*	*	Reserved
0	*	1	0	0	Memory data write
0	*	1	0	1	Memory data read
0	*	1	1	0	Reserved
0	*	1	1	1	Memory program read (fetch)
1	0	0	*	*	Internal RAM write
1	1	0	*	*	Internal RAM read
1	*	1	0	0	Internal RAM write (Note 1)
1	*	1	0	1	Internal RAM read (Note 1)
1	*	1	1	0	Reserved
1	0	1	1	1	Internal RAM write/ Memory program read (fetch)
1	1	1	1	1	Internal RAM read/ Memory program read (fetch)

Notes: 1. Access from chip select area or I/O area to the internal RAM.
2. * = Don't care

For example, a value of 0*101 for the lower 5 bits of access information indicates that an exception occurred during memory data read cycle. At this time, the address of the instruction to be read can be obtained from the stacked PC. However, note that the PC must be modified to get the actual instruction start address. In additions a value of 10111 for the lower 5 bits of access information indicates that multiple exceptions occurred during the internal RAM write cycle and memory program read cycle simultaneously. Moreover, a value of 1*100 for the lower 5 bits of access information indicates that an exception occurred during an internal RAM boundary access as shown in Figure 5-7.

The reason is as follows: The HD641016 can fetch a program instruction without the intervention of the CPU. Accordingly, the CPU can execute instructions which do not use the bus during the program fetch cycle. In addition, the internal RAM can be accessed in parallel with a program fetch since the internal RAM uses its specific internal bus. Consequently, the internal RAM can be accessed in parallel with the memory program read and multiple exceptions can occur both in RAM access and memory read cycles simultaneously. At this time, the memory program read address is contained in the program fetch address on the stack. The internal RAM address can be obtained from the PC on the stack.

4.6.8 Exception Processing Pointers

This section outlines the stack pointer and pointer register status when a bus error or access level violation is generated.

Exception during an ICBN or DCBN Instruction Cycle: If an overflow or underflow occurs during an ICBN or DCBN instruction cycle, the CPU stacks registers according to the register list in R15. However, if a bus error or access level violation occurs during stacking, the CPU stops stacking. Therefore, BSP is not updated, while CBNR and VBNR are updated. See "4.4 Register Banks" and "Section 16. Instruction Set" for details.

Exception during an LDM or STM Instruction Cycle: If a bus error or access level violation occurs during an LDM instruction cycle whose source addressing mode is auto-increment or auto-decrement, register Rn concerned with addressing is incremented or decremented by operation size. If the LDM instruction is completed normally, register Rn concerned with the addressing mode is updated to (initial value ± operation size x the number of registers). This also applies to the STM instruction, whose destination addressing mode is auto-increment or auto-decrement.

Exception When an Operand Is Specified by Auto-Increment or Auto-Decrement Addressing Mode: If a bus error or access level violation occurs while an operand is specified by an auto-increment or auto-decrement addressing mode, Rn is updated.

Exception during Information Stacking for Another Exception: If a bus error or access level error occurs during information stacking of another exception, the CPU stops stacking and SSP is maintained. The CPU then begins the bus error exception processing.

4.6.9 Stacked PC Value When an Exception Occurs

Table 4-16 shows the PC stacked when an exception occurs.

Table 4-16. PC Stacked in Exception

Exception	PC
Bus error Access level violation	• When the bus error occurs during a bus cycle with an instruction prefetch: The PC to be stacked is undefined since the bus cycle is performed asynchronously with instruction execution. • When the bus error occurs during an instruction cycle: The PC indicates an address from the 2nd byte of the instruction to the 1st byte of the next instruction if a bus error occurs in the read cycle. Otherwise, the PC is undefined since the CPU continues executing instructions in the prefetch queue.
Interrupt Trace DMA bus error DMA access level violation Acknowledge error TRAP instruction	The PC indicates the address of the 1st byte of the next instruction.
Illegal instruction Bank mode violation	The PC indicates an address from the 2nd to the last byte of the instruction generating this exception.
Unimplemented instruction Privilege violation	The PC indicates the address of the 1st byte of the instruction generating this exception.

4.7 Interrupt Controller

The HD641016 has 3 external (NMI, IRQ1, and IRQ0) and 22 internal interrupt sources (Table 4-17). The interrupt priority levels of internal interrupts IRQ1 and IRQ0 are programmable. NMI always has the highest priority (Figure 4-69).

The HD641016 interrupts have the following features:

- Three levels of programmable interrupt priority (other than NMI)
- Three interrupt acknowledge modes for external interrupts: Non-acknowledge, single acknowledge, and double acknowledge modes
- $\overline{\text{IRQ1}}$ and $\overline{\text{IRQ0}}$ individually maskable
- $\overline{\text{IRQ1}}$ and $\overline{\text{IRQ0}}$ either edge or level triggered

In non-acknowledge mode, the CPU internally generates the vector numbers for NMI, IRQ0, and IRQ1 which are H'31, H'30, and H'29, respectively.

Table 4-17. Internal Interrupt Vector Numbers

On-Chip I/O	Internal Interrupt	Interrupt Vector Number Decimal	Interrupt Vector Number Hexadecimal	Offset from EBR value
	Reserved	64	H'40	H'100
	Reserved	65	H'41	H'104
Timer 1	Count match	66	H'42	H'108
	Measurement end	67	H'43	H'10C
	Overflow	68	H'44	H'110
Timer 2	Count match	69	H'45	H'114
	Measurement end	70	H'46	H'118
	Overflow	71	H'47	H'11C
	Reserved	72	H'48	H'120
	Reserved	73	H'49	H'124
	Reserved	74	H'4A	H'128
DMAC 0	Block transfer end	75	H'4B	H'12C
	DMA transfer end	76	H'4C	H'130
DMAC 1	Block transfer end	77	H'4D	H'134
	DMA transfer end	78	H'4E	H'138
DMAC 2	Block transfer end	79	H'4F	H'13C
	DMA transfer end	80	H'50	H'140
DMAC 3	Block transfer end	81	H'51	H'144
	DMA transfer end	82	H'52	H'148
ASCI0	RX ready	83	H'53	H'14C
	TX ready	84	H'54	H'150
	RX interrupt	85	H'55	H'154
	TX interrupt	86	H'56	H'158
ASCI1	RX ready	87	H'57	H'15C
	TX ready	88	H'58	H'160
	RX interrupt	89	H'59	H'164
	TX interrupt	90	H'5A	H'168

The HD641016 interrupt controller consists of interrupt priority registers and an interrupt control register (Table 4-18), all of which are read/write registers.

Table 4-18. Interrupt Controller Registers

Register Name	Symbol	Address Offset	R/W	Value at Reset	Size
Interrupt priority register 0	IPR0	H'FF40	R/W	H'0000	W
Interrupt priority register 1	IPR1	H'FF42	R/W	H'0000	W
Interrupt priority register 2	IPR2	H'FF44	R/W	H'0000	W
Interrupt control register	ICR	H'FF46	R/W	H'0000	W

4.7.1 Interrupt Controller Registers

Interrupt Priority Register 0 (IPR0): The 16-bit read/write IPR0 register (Figure 4-65) specifies the interrupt priorities for timer 1, ASCI0, and ASCI1. IPR0 is not affected by the reset instruction.

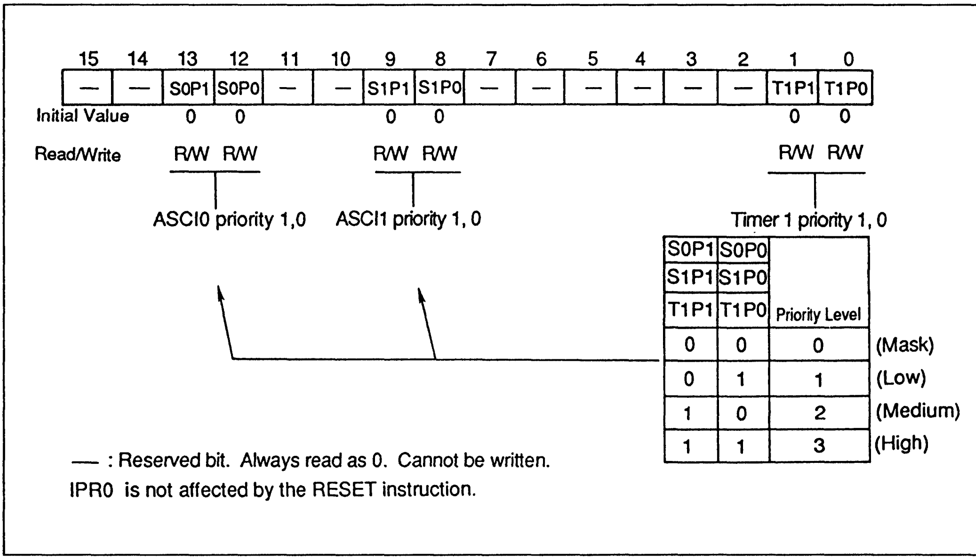


Figure 4-65. Interrupt Priority Register 0 (IPR0)

ASCI Channel 0 Priority 1, 0 (S0P1, S0P0): S0P1 and S0P0 specify the interrupt priority level for ASCI channel 0 as shown in Table 4-19.

Table 4-19. S0P1-S0P0 Setting and Priority Level

S0P1	S0P0	Priority Level
0	0	0 (Mask)
0	1	1 (Low)
1	0	2 (Medium)
1	1	3 (High)

ASCI Channel 1 Priority 1, 0 (S1P1, S1P0): S1P1 and S1P0 specify the interrupt priority level for ASCI channel 1 as shown in Table 4-20.

Table 4-20. S1P1-S1P0 Setting and Priority Level

S1P1	S1P0	Priority Level
0	0	0 (Mask)
0	1	1 (Low)
1	0	2 (Medium)
1	1	3 (High)

Timer Channel 1 Priority 1, 0 (T1P1, T1P0): T1P1 and T1P0 specify the interrupt priority level for timer channel 1 as shown in Table 4-21.

Table 4-21. T1P1-T1P0 Setting and Priority Level

T1P1	T1P0	Priority Level
0	0	0 (Mask)
0	1	1 (Low)
1	0	2 (Medium)
1	1	3 (High)

Interrupt Priority Register 1 (IPR1) : The 16-bit read/write IPR1 (Figure 4-66) specifies the interrupt priorities for timer 2, DMAC0, and DMAC1. IPR1 is not affected by the RESET instruction.

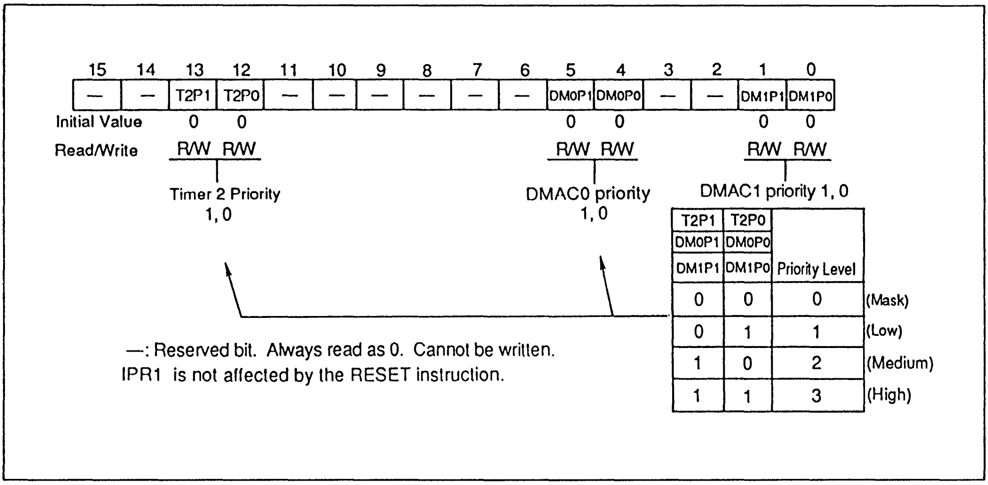


Figure 4-66. Interrupt Priority Register 1 (IPR1)

Timer 2 Priority 1, 0 (T2P1, T2P0): T2P1 and T2P0 specify the interrupt priority level for timer 2 as shown in Table 4-22.

Table 4-22. T2P1-T2P0 Setting and Priority Level

T2P1	T2P0	Priority Level
0	0	0 (Mask)
0	1	1 (Low)
1	0	2 (Medium)
1	1	3 (High)

DMAC Channel 0 Priority 1, 0 (DM0P1, DM0P0): DM0P1 and DM0P0 specify the interrupt priority level for DMAC channel 0 as shown in Table 4-23.

Table 4-23. DM0P1-DM0P0 Setting and Priority Level

DM0P1	DM0P0	Priority Level
0	0	0 (Mask)
0	1	1 (Low)
1	0	2 (Medium)
1	1	3 (High)

DMAC Channel 1 Priority 1, 0 (DM1P1, DM1P0): DM1P1 and DM1P0 specify the interrupt priority level for DMAC channel 1 as shown in Table 4-24.

Table 4-24. DM1P1-DM1P0 Setting and Priority Level

DM1P1	DM1P0	Priority Level
0	0	0 (Mask)
0	1	1 (Low)
1	0	2 (Medium)
1	1	3 (High)

Interrupt Priority Register 2 (IPR2): The 16-bit read/write IPR2 (Figure 4-67) specifies the interrupt priorities for DMAC2, DMAC3, IRQ0, and IRQ1. IPR2 is not affected by the RESET instruction.

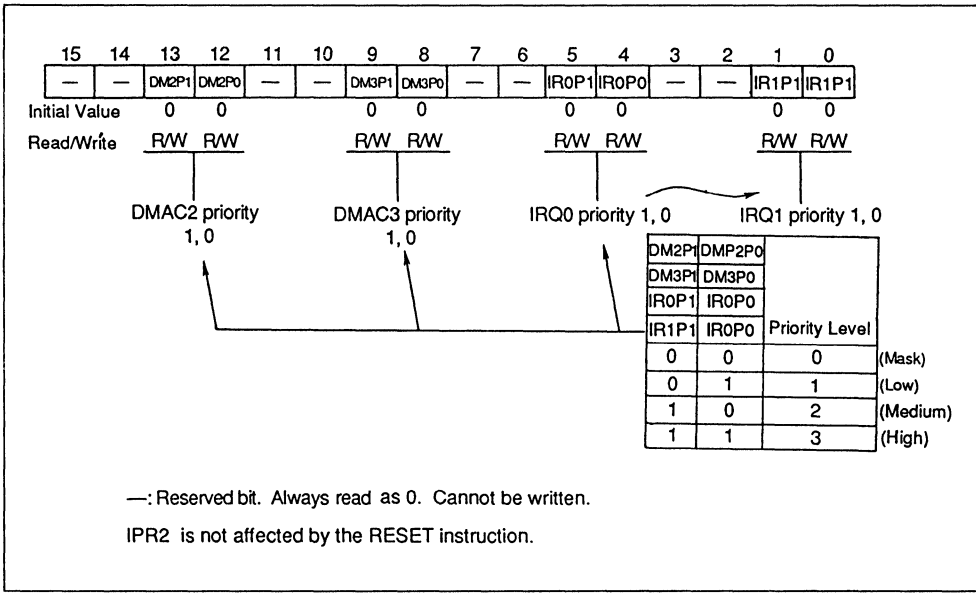


Figure 4-67. Interrupt Priority Register 2 (IPR2)

DMAC Channel 2 Priority 1, 0 (DM2P1, DM2P0): DM2P1 and DM2P0 specify the interrupt priority level for DMAC channel 2 as shown in Table 4-25.

Table 4-25. DM2P1-DM2P0 Setting and Priority Level

DM2P1	DM2P0	Priority Level
0	0	0 (Mask)
0	1	1 (Low)
1	0	2 (Medium)
1	1	3 (High)

DMAC Channel 3 Priority 1, 0 (DM3P1, DM3P0): DM3P1 and DM3P0 specify the interrupt priority level for DMAC channel 3 as shown in Table 4-26.

Table 4-26. DM3P1-DM3P0 Setting and Priority Level

DM3P1	DM3P0	Priority Level
0	0	0 (Mask)
0	1	1 (Low)
1	0	2 (Medium)
1	1	3 (High)

IRQ0 Priority 1, 0 (IR0P1, IR0P0): IR0P1 and IR0P0 specify the IRQ0 interrupt priority level as shown in Table 4-27.

Table 4-27. IR0P1-IR0P0 Setting and Priority Level

IR0P1	IR0P0	Priority Level
0	0	0 (Mask)
0	1	1 (Low)
1	0	2 (Medium)
1	1	3 (High)

IRQ1 Priority 1, 0 (IR1P1, IR1P0): IR1P1 and IR1P0 specify the IRQ1 interrupt priority level as shown in Table 4-28.

Table 4-28. IR1P1-IR1P0 Setting and Priority Level

IR1P1	IR1P0	Priority Level
0	0	0 (Mask)
0	1	1 (Low)
1	0	2 (Medium)
1	1	3 (High)

Interrupt Control Register (ICR): ICR controls external interrupt request vector modes, interrupt masking, edge or level triggering, and interrupt acknowledge modes (Figure 4-68). The upper byte of ICR is reserved, and always read as 0. ICR is not affected by the RESET instruction.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	—	—	IMOD	NMIV	IRQV0	IRQV1	IRQIM0	IRQIM1	—	—
Initial Value									0	0	0	0	0	0		
Read/Write									R/W	R/W	R/W	R/W	R/W	R/W		

IRQ Input Mode 0, 1

$\overline{\text{IRQ1}}$ Pin

IRQV1	IRQIM1	Acknowledge Mode	Trigger Mode
0	0	Non-acknowledge mode	Level
	1	Non-acknowledge mode	Edge
1	0	Single acknowledge mode	Level
	1	Single acknowledge mode	Edge

$\overline{\text{IRQ0}}$ Pin

IMOD	IRQV0	IRQIM0	Acknowledge Mode	Trigger Mode	IACK Pin
0	0	0	Non-acknowledge mode	Level	Unused
		1	Non-acknowledge mode	Edge	
	1	0	Single acknowledge mode	Level	
		1	Single acknowledge mode	Edge	
1	0	0	Double acknowledge mode	Level	Used
		1			
	1	0			
		1			

IRQ Vector Select 0,1

NMI Vector Select

0	Non-acknowledge mode
1	Single acknowledge mode

Interrupt Mode

—: Reserved bit. Always read as 0. Cannot be written.

ICR is not affected by the RESET instruction.

Figure 4-68. Interrupt Control Register

Interrupt Mode (IMOD): IMOD determines whether the IRQ0 interrupt acknowledge mode is double acknowledge. When IMOD = 0, the IRQ0 interrupt acknowledge mode is single acknowledge or non-acknowledge, depending on IRQV0. When IMOD = 1, the IRQ0 interrupt acknowledge mode is double acknowledge. At this time, the $\overline{\text{IACK}}$ signal is output, bits 6-2 of ICR are ignored, and $\overline{\text{IRQ0}}$ is specified as level triggered.

NMI Vector Select and IRQ Vector Select 0-1 (NMI, IRQV0-1): NMIV, IRQV0, and IRQV1 are the NMI, IRQ0, and IRQ1 vector mode bits. If the vector mode bit is 0, that interrupt will be auto-vectored (non-acknowledge mode). If the bit is 1, that interrupt will be in external vector mode (single acknowledge for IRQ1 and NMI; single or double acknowledge for IRQ0, depending on IMOD).

$\overline{\text{IRQ}}$ Input Mode 0, 1 (IRQIM0, IRQIM1): IRQIM1 and IRQIM0 are the $\overline{\text{IRQ1}}$ and $\overline{\text{IRQ0}}$ input trigger mode bits. If the bit is 0, the interrupt is low-level triggered. If the bit is 1, the interrupt is falling-edge triggered. $\overline{\text{NMI}}$ is always edge triggered.

4.7.2 Interrupt Controller Operation

Priority Specification: Interrupt priority is specified as shown in Figure 4-69.

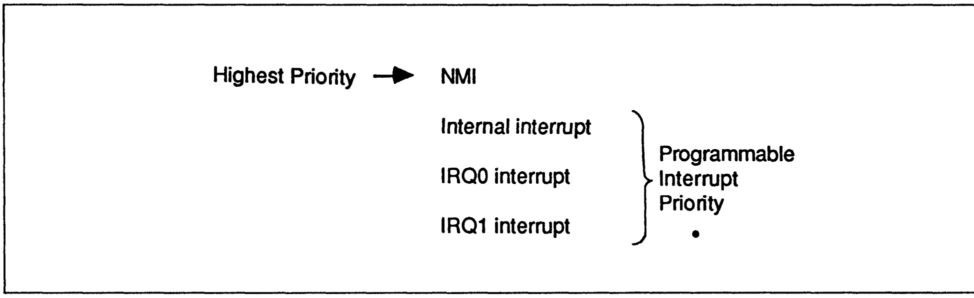


Figure 4-69. Interrupt Priority

Internal interrupts and IRQ0, and IRQ1 interrupts are programmable into three levels by IPR0-IPR2. If interrupts of different priorities are requested simultaneously, the highest priority interrupt processing precedes the lower priority interrupt processing. If interrupts of the same priority level occur simultaneously, the CPU services the interrupt of the highest priority offset as shown in Table 4-29.

Table 4-29. Internal Interrupt Offset Priority

No.	Offset Priority	On-Chip I/O Device
1	Highest	ASCI0
2	↑	ASCI1
3		Timer1
4		Timer2
5		DMAC0
6		DMAC1
7		DMAC2
8		DMAC3
9	↓	IRQ0
10	Lowest	IRQ1

If the CPU accepts an interrupt request, the CPU copies the interrupt priority level into the I0-I2 bits of SR to mask any other interrupts of the same or lower priority. At this time, the interrupt priority level for $\overline{\text{NMI}}$ is the highest level 7. See "4.3.2 Status Register" for details on the I0-I2 bits.

Figure 4-70 shows an example of interrupt priority programming by IPR0-IPR2. Level 3 is the highest priority. Interrupts of the same level are prioritized by the offset listed in Table 4-29.

In Figure 4-70,

Level 3: DMAC0 > DMAC1

Level 2: ASCI0 > Timer1 > DMAC2 > DMAC3

Level 1: ASCI1 > $\overline{\text{IRQ0}}$

Level 0: Timer 2 and $\overline{\text{IRQ1}}$ interrupts are masked.

Table 4-30 shows the interrupt priority for each on-chip I/O device.

Table 4-30. Interrupt Priority in Each On-Chip I/O Device

	Interrupt		
Priority	**DMA Controller**	**Timer**	**ASCI**
Highest	Block transfer end	Count match	RX ready (RXRDY)
↕	Transfer end	Measurement end	TX ready (TXRDY)
		Overflow	RX interrupt (RXINT)
Lowest			TX interrupt (TXINT)

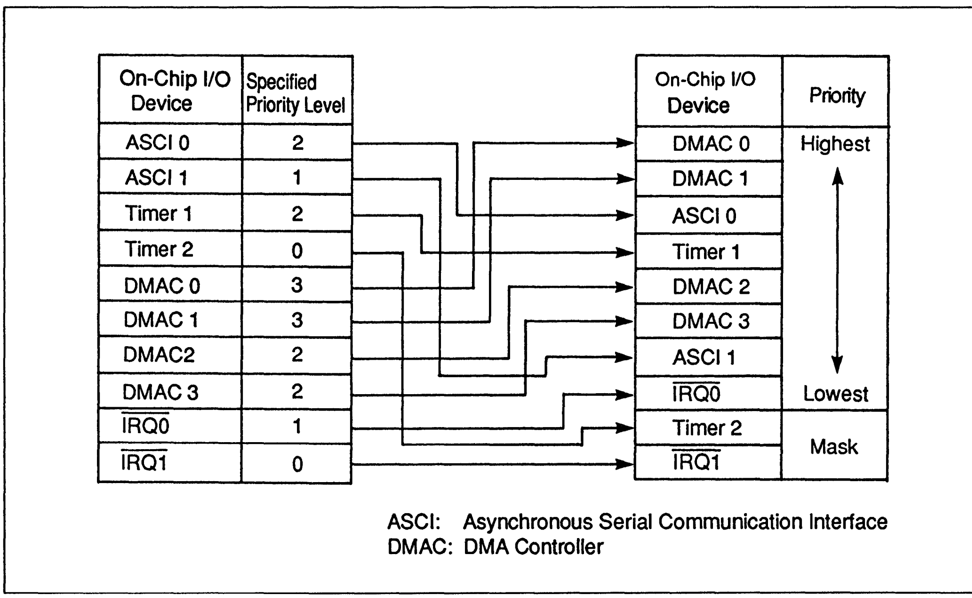


Figure 4-70. Interrupt Priority Programming Example

Interrupt Arbitrator: Figure 4-71 shows the implementation of the arbitrator.

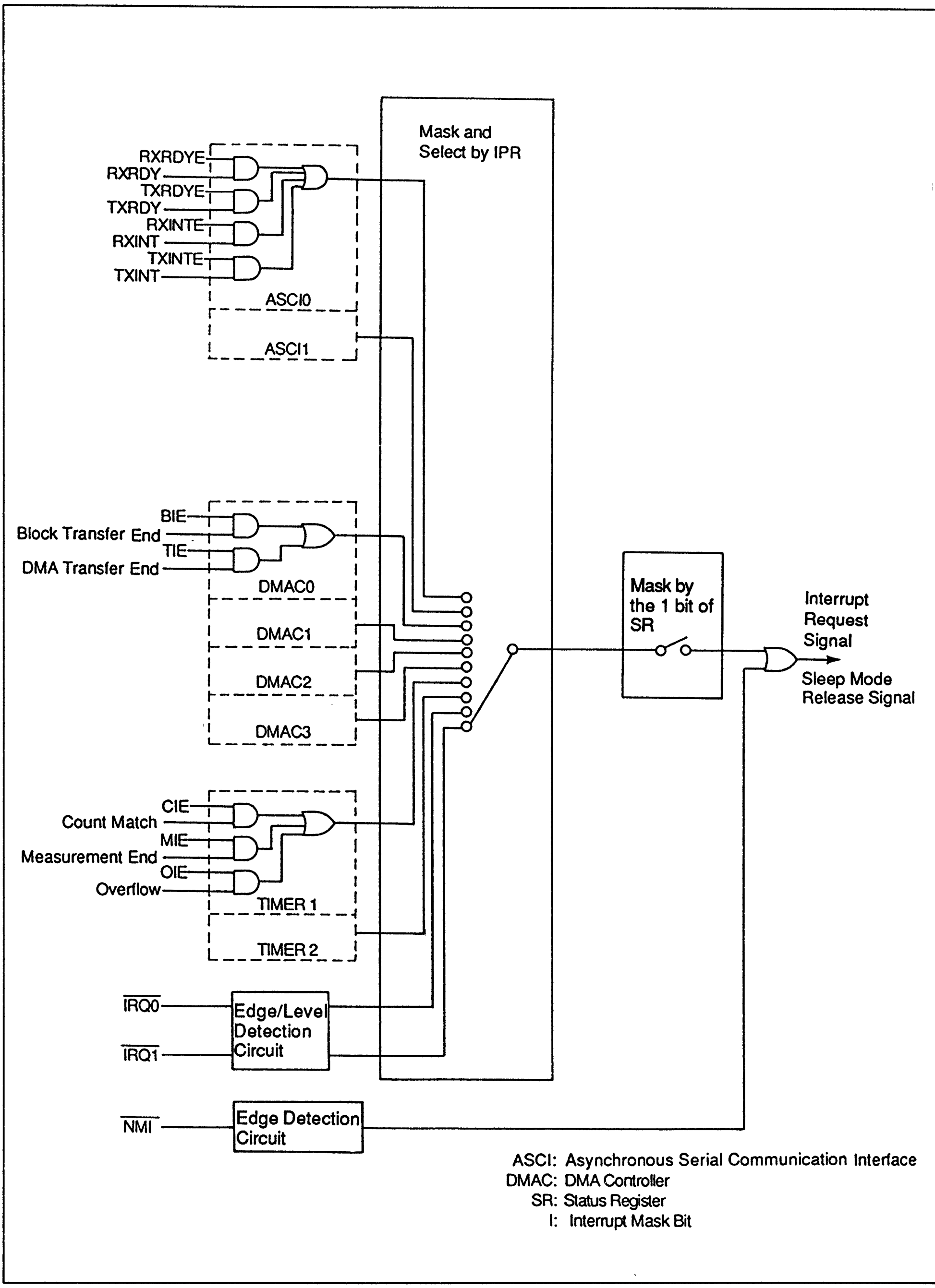


Figure 4-71. Interrupt Arbitrator

External Interrupt: The HD641016 has three external interrupt sources: $\overline{\text{NMI}}$, $\overline{\text{IRQ0}}$, and $\overline{\text{IRQ1}}$.

Trigger and acknowledge modes of these interrupts can be specified as shown in Table 4-31.

Table 4-31. Valid Trigger and Acknowledge Modes

	Trigger Mode		**Acknowledge Mode**		
	Edge	**Level**	**Non-Acknowledge Mode**	**Single-Acknowledge Mode**	**Double-Acknowledge Mode**
$\overline{\text{NMI}}$	O	X	O	O	X
$\overline{\text{IRQ0}}$	O	O	O	O	O
$\overline{\text{IRQ1}}$	O	O	O	O	X

Notes: O = Available

X = Not available

Tables 4-32 through 4-34 show acknowledge and trigger mode combinations for each interrupt. Note that ICR controls both acknowledge and trigger modes.

Table 4-32. $\overline{\text{NMI}}$ Modes (Always Edge Triggered)

NMIV	**Acknowledge Mode**
0	Non-acknowledge mode
1	Single acknowledge mode

Table 4-33. $\overline{IRQ1}$ Modes

IRQV1	IRQIM1	Acknowledge Mode	Trigger Mode
0	0	Non-acknowledge mode	Level
	1	Non-acknowledge mode	Edge
1	0	Single acknowledge mode	Level
	1	Single acknowledge mode	Edge

Table 4-34. $\overline{IRQ0}$ Modes

IMOD	IRQV0	IRQIM0	Acknowledge Mode	Trigger Mode	IACK Pin
0	0	0	Non-acknowledge mode	Level	Unused
		1	Non-acknowledge mode	Edge	Unused
	1	0	Single acknowledge mode	Level	Unused
		1	Single acknowledge mode	Edge	Unused
1	0	0	Double acknowledge mode	Level	Used
		1			
	1	0			
		1			

Trigger Mode: IRQ0 and IRQ1 interrupts can be programmed as either edge or level triggered. If an IRQ interrupt is programmed as level triggered, the CPU accepts the interrupt if the $\overline{IRQ}$ pin is asserted low. However, the IRQ interrupt is ignored if the $\overline{IRQ}$ pin is pulled high before it is accepted. (See Figures 4-72 to 4-73.)

If an IRQ interrupt is programmed as edge triggered, the CPU accepts the interrupt when the $\overline{IRQ}$ pin's falling edge is internally latched. The $\overline{IRQ}$ pin must be asserted low for at least 2 clock cycles to be latched correctly. Once the $\overline{IRQ}$ pin's falling edge is internally latched, the $\overline{IRQ}$ interrupt can be accepted even if the $\overline{IRQ}$ pin is negated. (See Figures 4-72 to 4-73.)

In addition, the flip-flop set by the $\overline{IRQ}$ pin's falling edge is cleared when an IRQ interrupt exception begins.

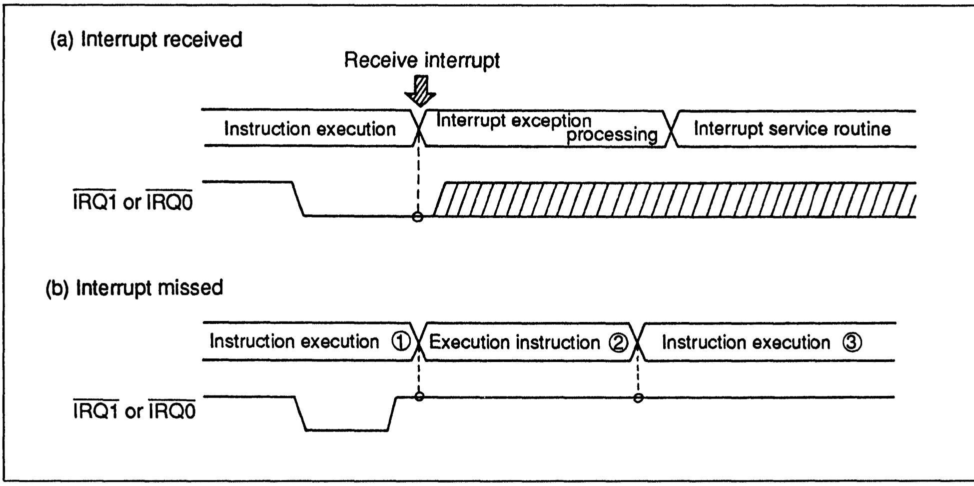


Figure 4-72. Level-Triggered Interrupt Accept Timing

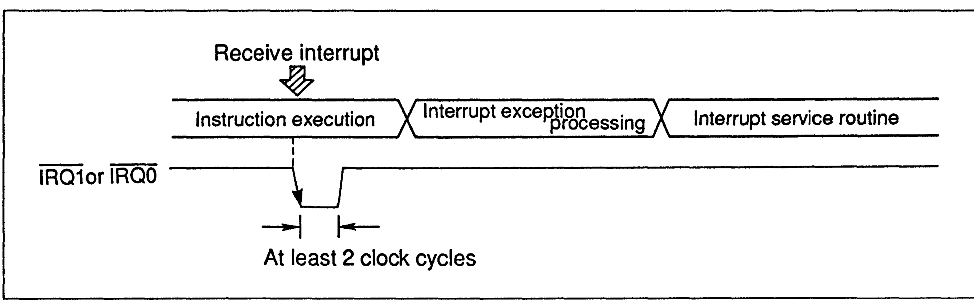


Figure 4-73. Edge-Triggered Interrupt Accept Timing

Acknowledge Modes: This section outlines each of the three acknowledge sequences which can be selected by ICR.

Non-Acknowledge: In non-acknowledge mode, the CPU internally generates fixed vector numbers for interrupt sources to fetch the start address of the interrupt service routine (ISR) from the exception processing vector table. Figure 4-74 shows the non-acknowledge mode sequence.

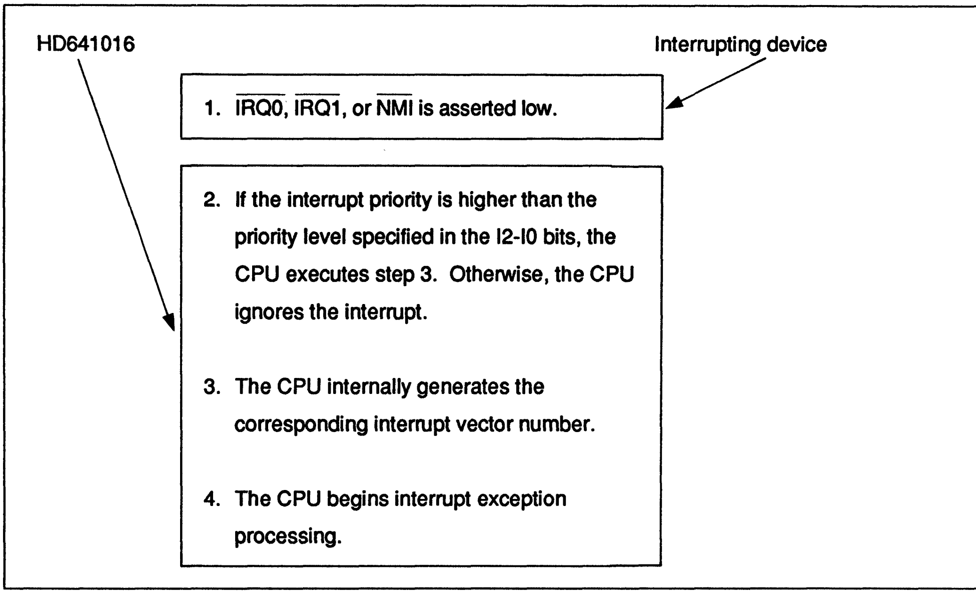


Figure 4-74. Non-Acknowledge Mode Interrupt Sequence

Single Acknowledge Mode: In single acknowledge mode, the CPU internally latches an interrupt vector number on the data bus during the first interrupt acknowledge cycle to fetch the start address of the ISR (interrupt service routine) from the exception processing vector table. Therefore, the interrupting device must place the interrupt vector on the data bus. In addition, note that the CPU executes bus retry or acknowledge error interrupt processing depending on the BRTE bit value, if $\overline{BRTRY}$ is asserted low during interrupt acknowledge cycle. Figure 4-75 shows the interrupt sequence in single acknowledge mode. Figure 4-76 shows single acknowledge mode bus timing.

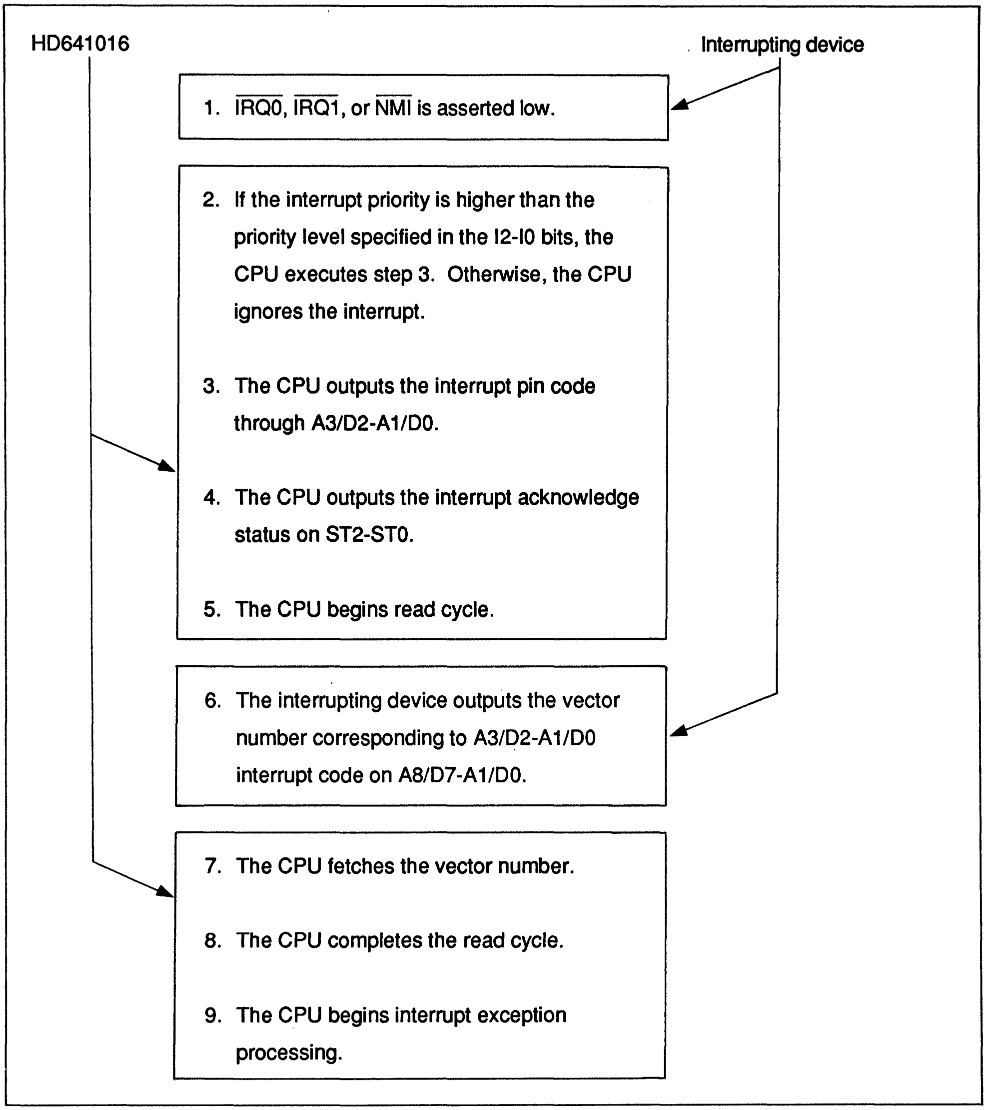


Figure 4-75. Single Acknowledge Mode Interrupt Sequence

Table 4-35 shows the interrupt pin codes for interrupt request pins.

Table 4-35. Interrupt Pin Code

Interrupt Request Pin	Code
NMI	7
IRQ0	6
IRQ1	5

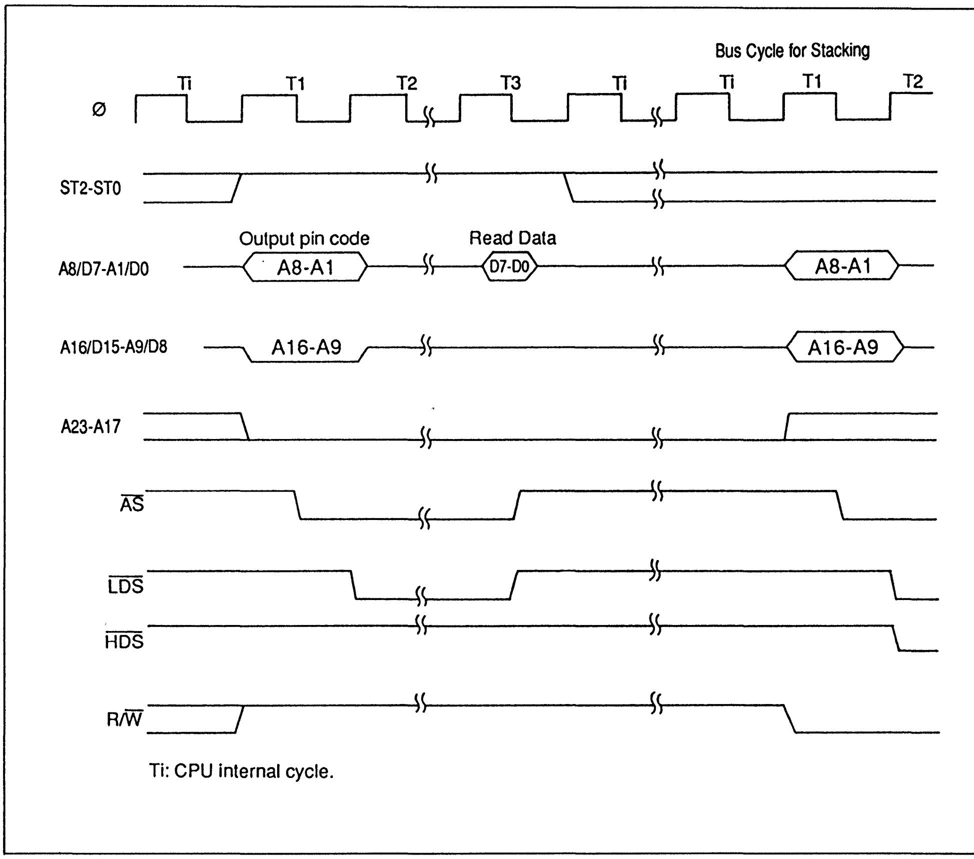


Figure 4-76. Single Acknowledge Mode Bus Cycle Timing

Double Acknowledge Mode: In double acknowledge mode, the CPU internally latches an interrupt vector number on the data bus during the second interrupt acknowledge cycle to fetch the start address of the ISR from the exception processing vector table. Figure 4-77 shows the interrupt sequence in double acknowledge mode. Figure 4-78 shows double acknowledge mode bus timing.

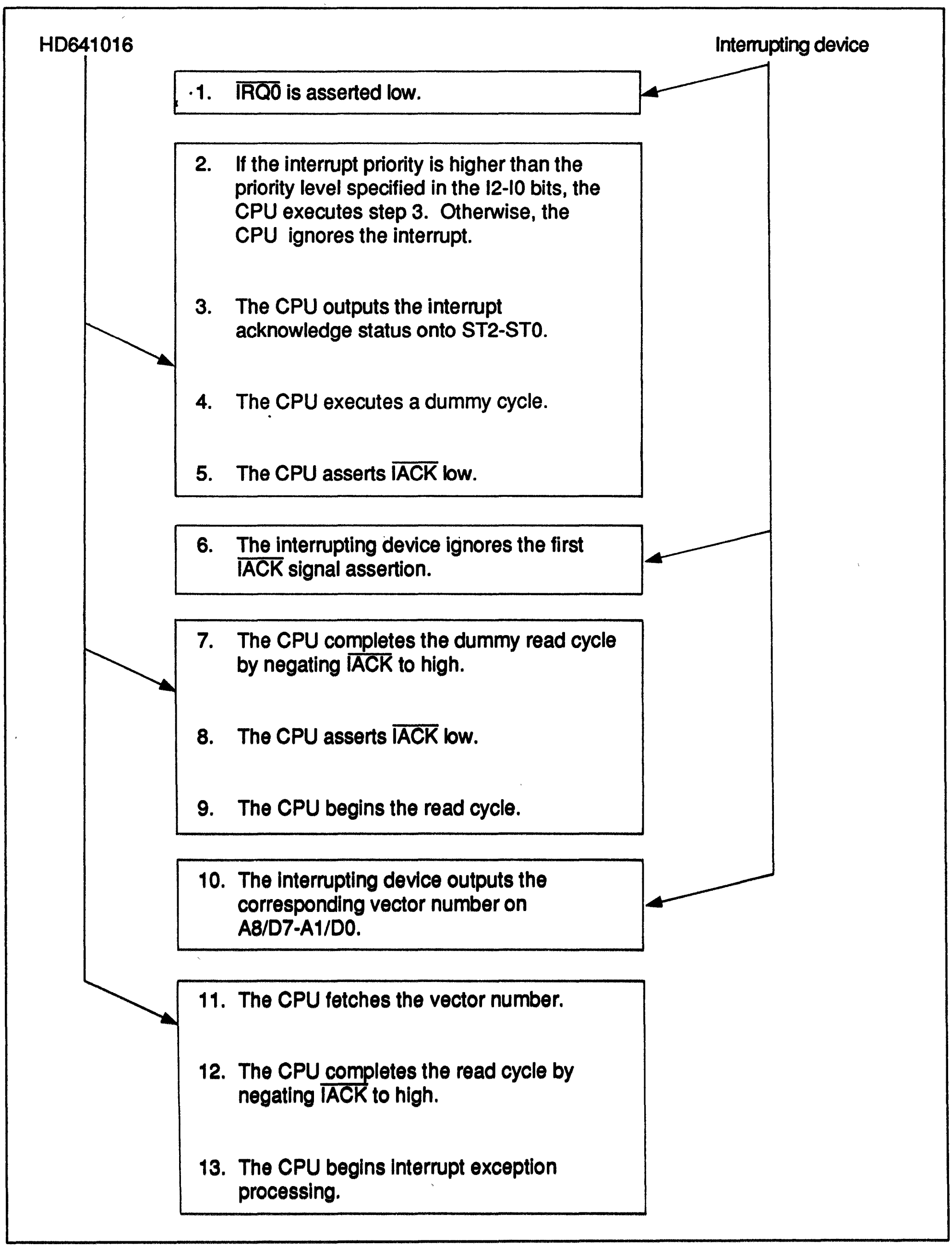


Figure 4-77. Double Acknowledge Mode Interrupt Sequence

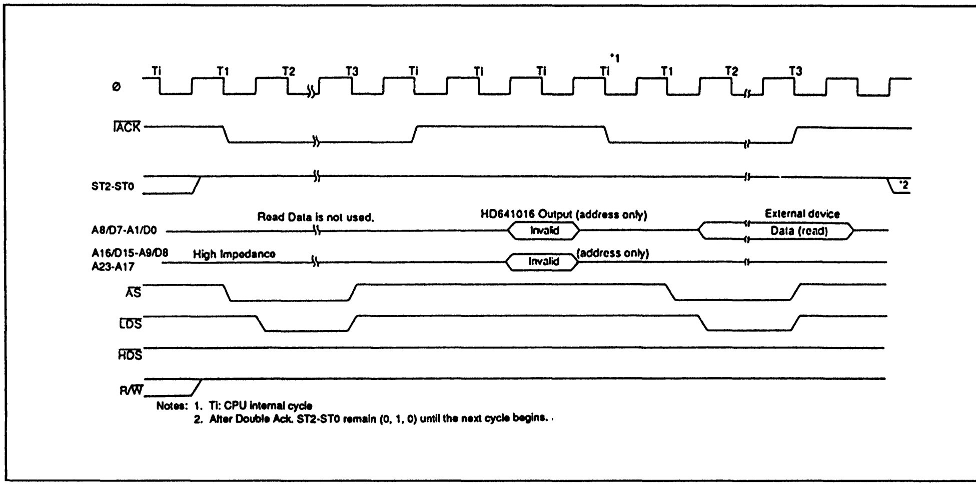


Figure 4-78. Double Acknowledge Mode Bus Cycle Timing

4.7.3 External Interrupt Applications

Non-Acknowledge Mode Interrupt Application: Non-acknowledge mode supports external interrupt devices which cannot generate interrupt vector numbers (Figure 4-79). The CPU generates separate fixed vector numbers for $\overline{\text{NMI}}$, $\overline{\text{IRQ0}}$, and $\overline{\text{IRQ1}}$ as shown in Table 4-36. In this mode, the ICR is set to H'0000.

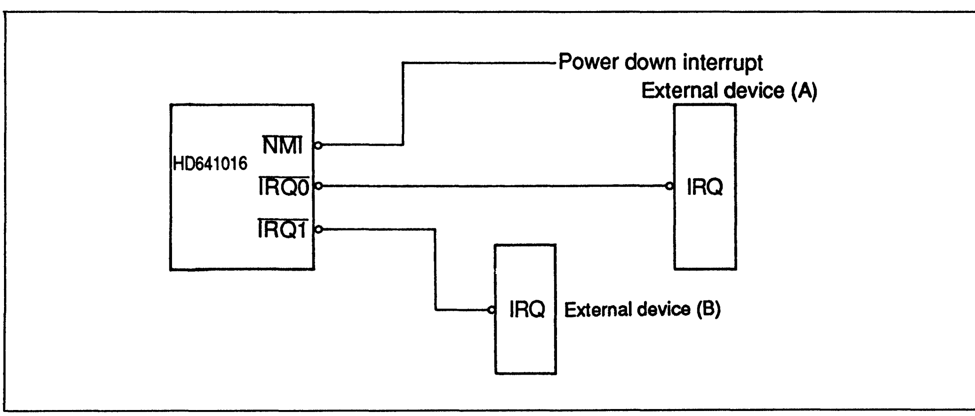


Figure 4-79. Non-Acknowledge Mode Application

Table 4-36. External Interrupt Fixed Vector Numbers

Interrupt Source	Vector Number
$\overline{\text{NMI}}$	31
$\overline{\text{IRQ0}}$	30
$\overline{\text{IRQ1}}$	29

Single Acknowledge Mode Interrupt Application: Single acknowledge mode allows an external circuit to generate multiple vector numbers for each external interrupt source. The system in Figure 4-80 generates only one vector number for each source. However, multiple vectors can be generated using a daisy-chain scheme and external circuits. During the interrupt acknowledge cycle, A3-A1 of the address bus indicate the interrupt source as shown in Table 4-37. External devices put the vector number on the lower 8 bits of the data bus.

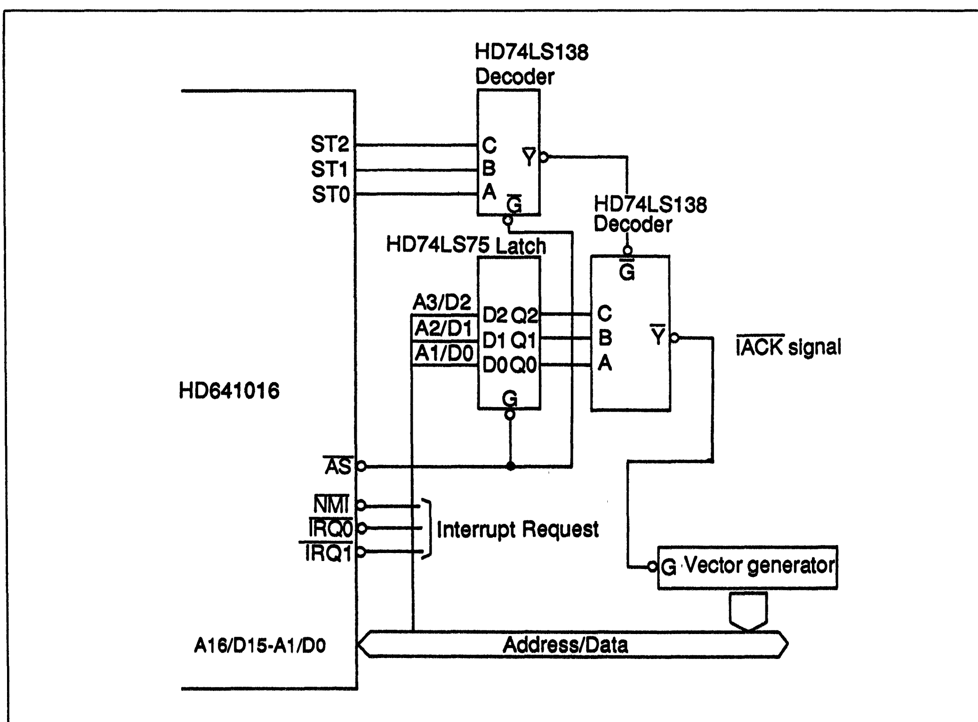


Figure 4-80. Single Acknowledge Mode Application

Table 4-37. A3-A1 Code in Interrupt Acknowledge Cycle

Interrupt Source	A3-A1
$\overline{NMI}$	7
$\overline{IRQ0}$	6
$\overline{IRQ1}$	5

Double Acknowledge Mode 8259A Interface: Double acknowledge mode allows the HD641016 to interface easily with an 8259A interrupt controller, as shown in Figure 4-81.

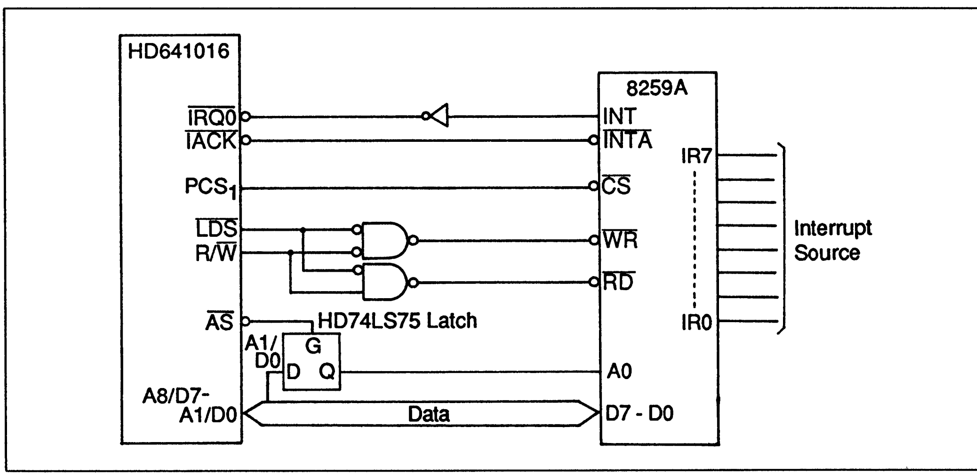


Figure 4-81. 8259A Interrupt Interface

4.8 Bus Release

4.8.1 Bus Release Timing

The HD641016 provides the $\overline{\text{BREQ}}$ and $\overline{\text{BACK}}$ pins for releasing the bus. An external bus master requests bus release by asserting the $\overline{\text{BREQ}}$ signal. Upon receiving $\overline{\text{BREQ}}$, the HD641016 completes the current bus cycle and then asserts $\overline{\text{BACK}}$ to notify the external bus master that the bus has been released to it as long as another higher priority bus master does not request bus release. Upon receiving $\overline{\text{BACK}}$, the external bus master begins its bus cycle. Figure 4-82 shows bus release timing.

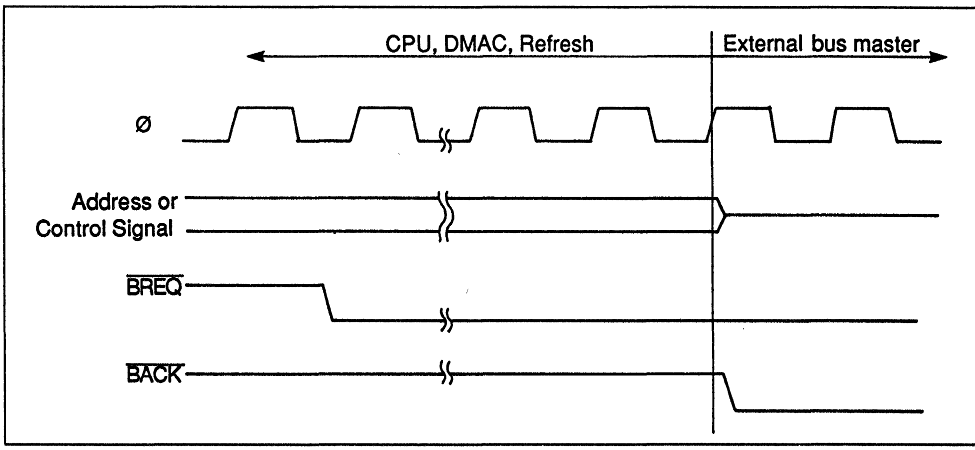


Figure 4-82. Bus Release Timing

The external bus master can release the bus mastership by one of the following methods. In the first method, the external bus master negates $\overline{\text{BREQ}}$ to release the bus and $\overline{\text{BREQ}}$ is sampled at the falling edge of the Ø clock. $\overline{\text{BACK}}$ is also negated at the falling edge of the Ø clock one clock cycle after the $\overline{\text{BREQ}}$ negation. Figure 4-83 shows this bus release timing.

In the second method, $\overline{\text{BACK}}$ is forcibly negated by a higher priority bus master. If an internal DRAM refresh is requested during an external bus master cycle of lower priority, $\overline{\text{BACK}}$ is forcibly negated. $\overline{\text{BREQ}}$ must then be negated as shown in Figure 4-84.

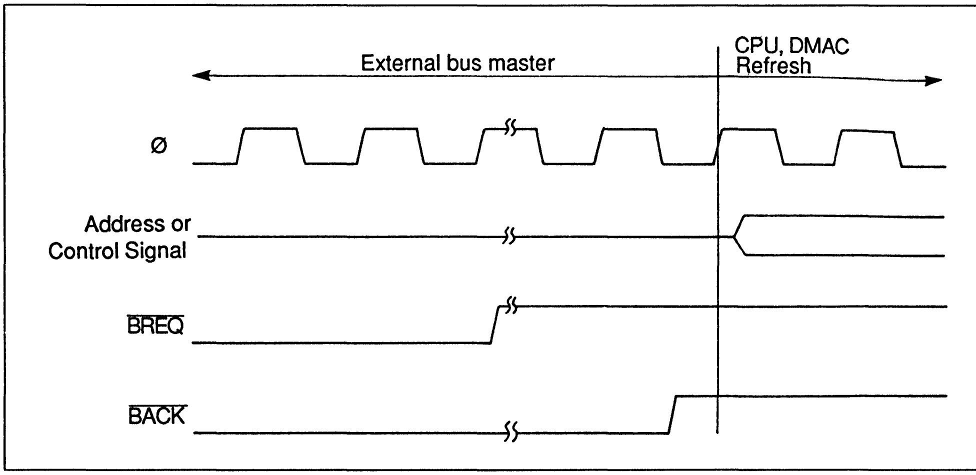


Figure 4-83. Bus Release Timing for External Bus Master 1

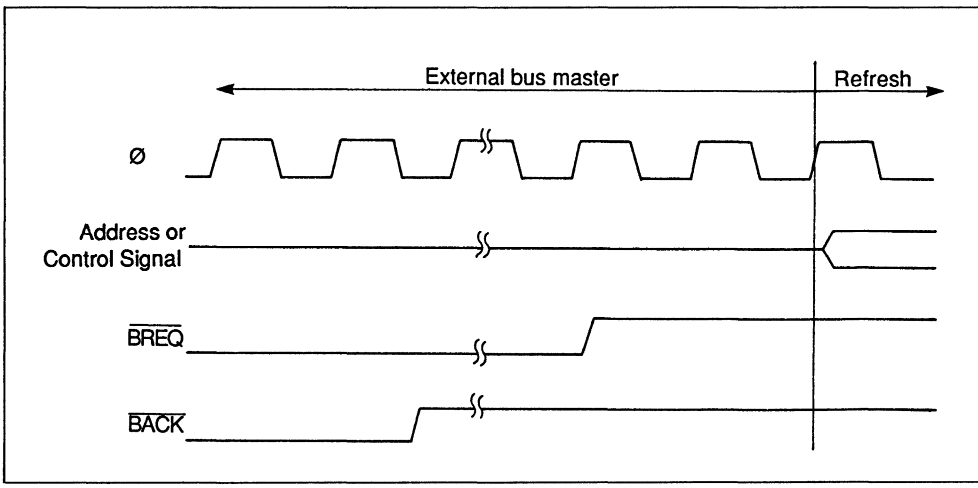


Figure 4-84. Bus Release Timing for External Bus Master 2

4.8.2 $\overline{\text{WAIT}}$ Output Function

When the bus is released for an external bus master, address lines and the $\overline{\text{AS}}$ pin function as inputs. At this time, if the $\overline{\text{WTOE}}$ bit of the area wait control register, (AWCR) is set to 1, the address output by the external bus master is compared with the chip select area. If the address output by the external bus master is included within the chip select area, the $\overline{\text{WAIT}}$ signal is asserted according to the number of wait states specified for the area (the number of wait states + 1 state). Note that at least one Tw state is always inserted for all bus cycles. (One Tw state is inserted even if 0 Tw state insertion is selected for an area.)

Figure 4-85 shows the Tw state insertion timing during an external bus master cycle. PCS1 and PCS0 output the chip select area code. However, note that the external bus masters cannot access the internal RAM and I/O areas. If they try, the external bus masters actually accesses an external memory areas.

See "Section 11. Chip Select Controller" and "Section 12. Wait State Controller" for details.

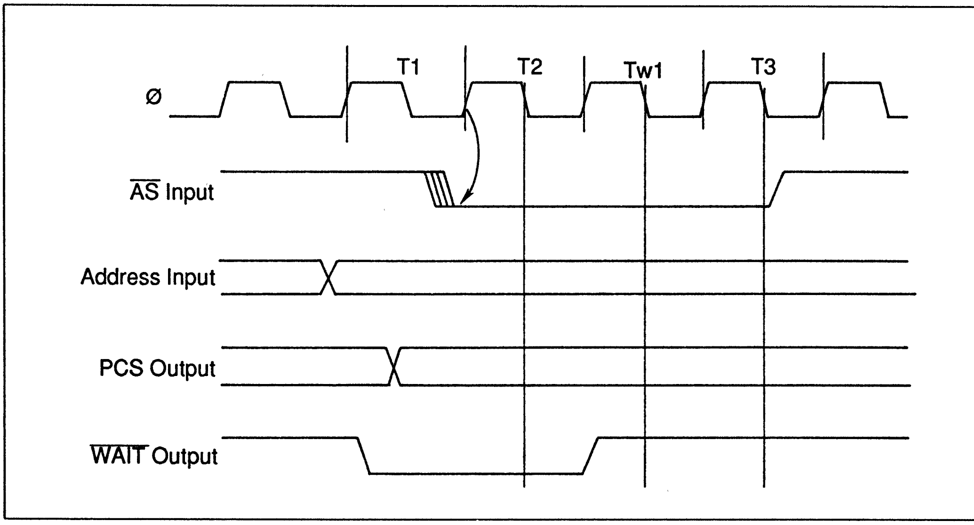


Figure 4-85. Tw State Insertion Timing

4.9 CPU Basic Timing

4.9.1 Timing Generator

The external clock is connected to EXTAL or a crystal resonator is connected to XTAL and EXTAL as shown in Figure 4-86. This external clock or crystal resonator signal is divided by two to generate system clock Ø. Furthermore, Ø is divided by eight to produce the E clock for 6800 family LSI interface.

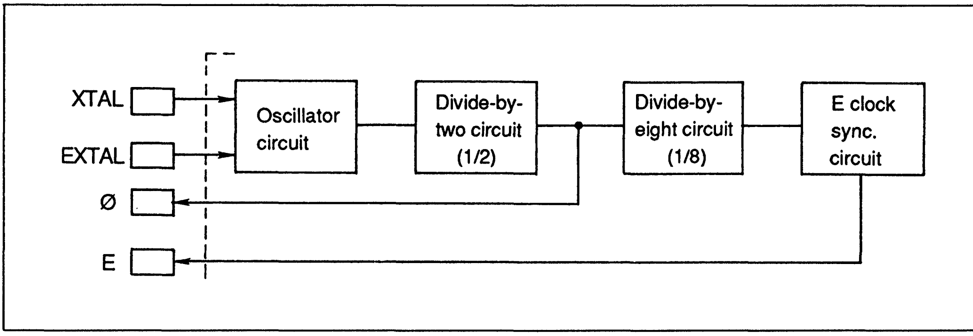


Figure 4-86. HD641016 Timing Generator Circuit

Figure 4-87 shows EXTAL and Ø relationship. One state is defined as the period from one rising edge to the next rising edge of Ø.

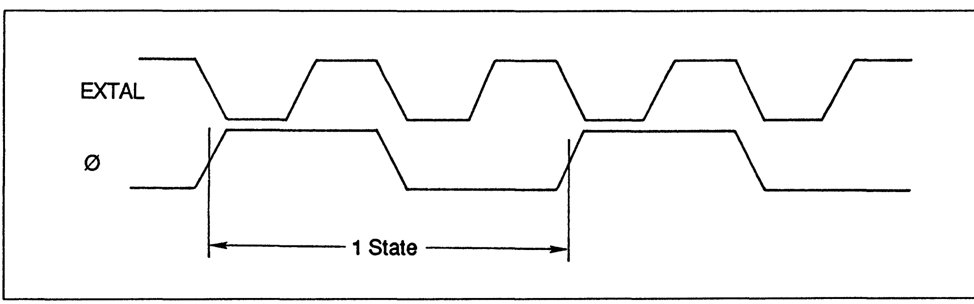


Figure 4-87. HD641016 Clock Timing

4.9.2 CPU Read/Write Cycle

The basic CPU operation consists of one or more machine cycles (MC). One machine cycle consists of three system states T1, T2, and T3. In addition, Tw states can be inserted between T2 and T3 states when accessing slow memory or I/O devices by $\overline{\text{WAIT}}$ input or by software. For precharging DRAM RAS, a Tp state can be inserted before T1 state by software. See "Section 12. Wait State Controller" for details on Tw and Tp states.

Figure 4-88 shows the CPU read/write cycle with no Tw or Tp states.

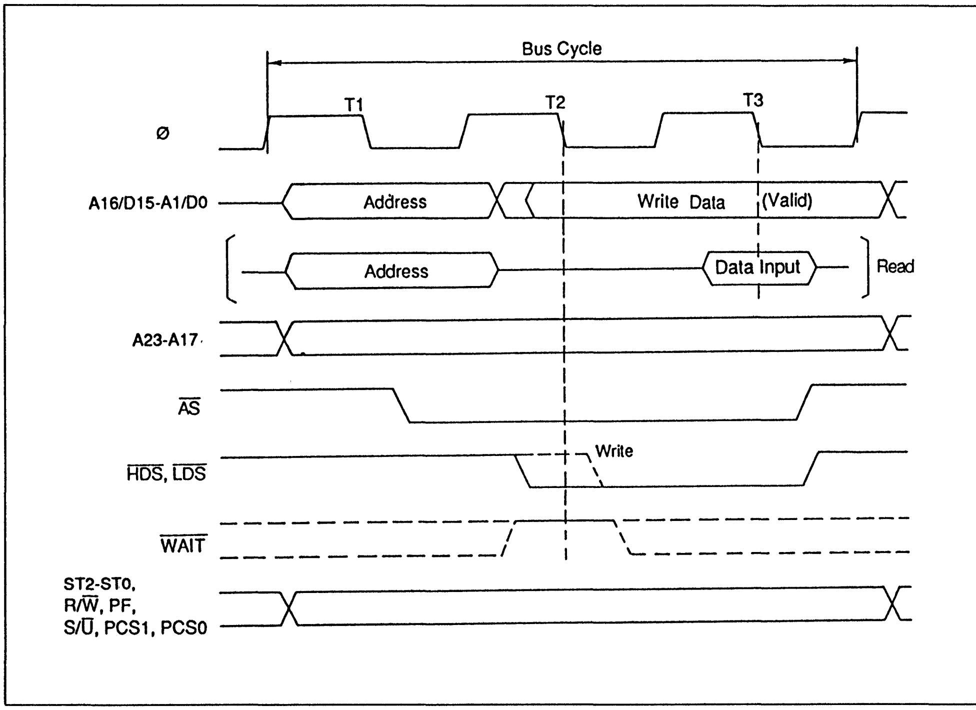


Figure 4-88. CPU Read/Write Cycle with No Tw State

Figures 4-89 and 4-90 show the CPU read/write cycle with Tw and Tp states.

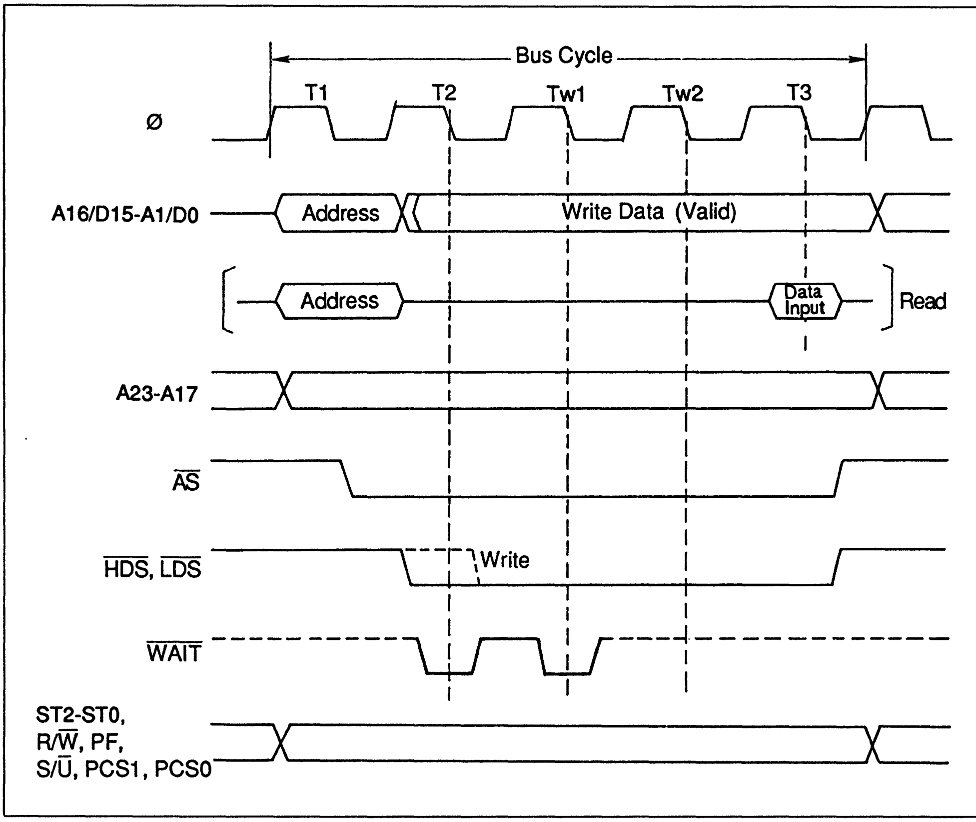


Figure 4-89. CPU Read/Write Cycle with Tw State

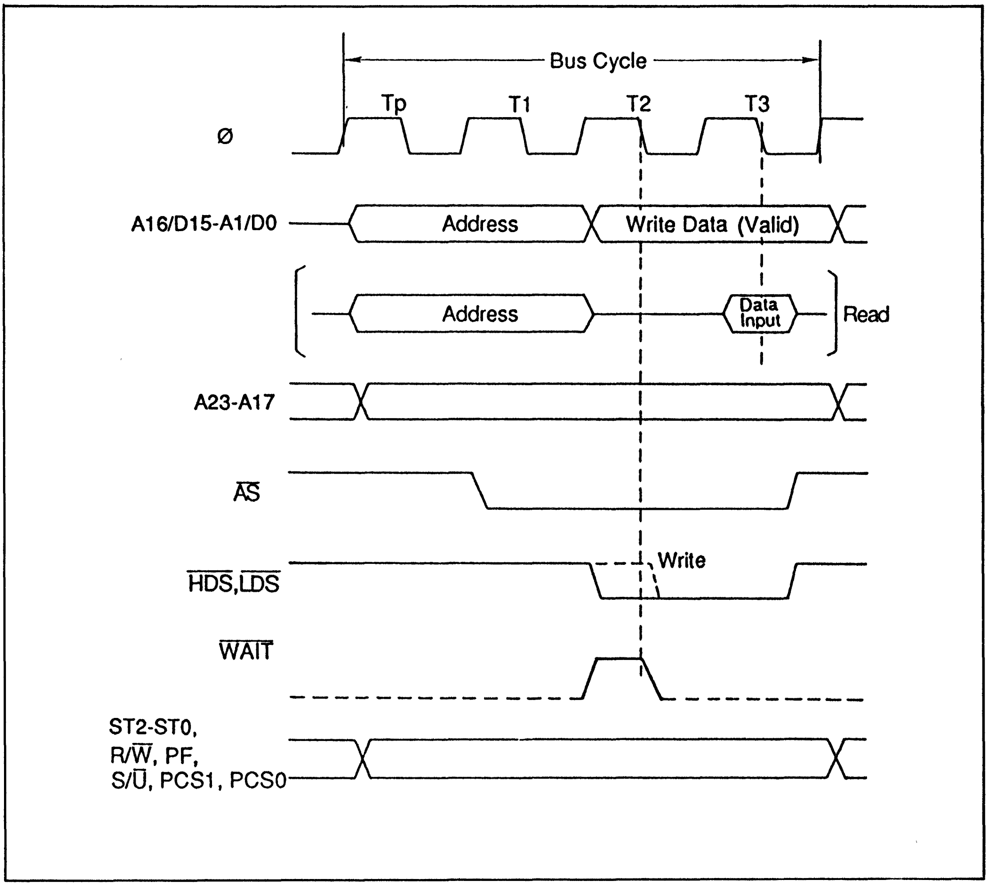


Figure 4-90. CPU Read/Write Cycle with Tp State

Figure 4-91 shows a read-modify-write cycle for the TAS instruction. In this cycle, ST2-ST0 indicate the bus lock state. Here, the CPU does not accept any interrupt or perform bus arbitration. See "Section 7. Bus Arbitrator" for details. If a bus error or access level violation occurs during the read cycle, the CPU does not execute the write cycle and begins the corresponding exception processing.

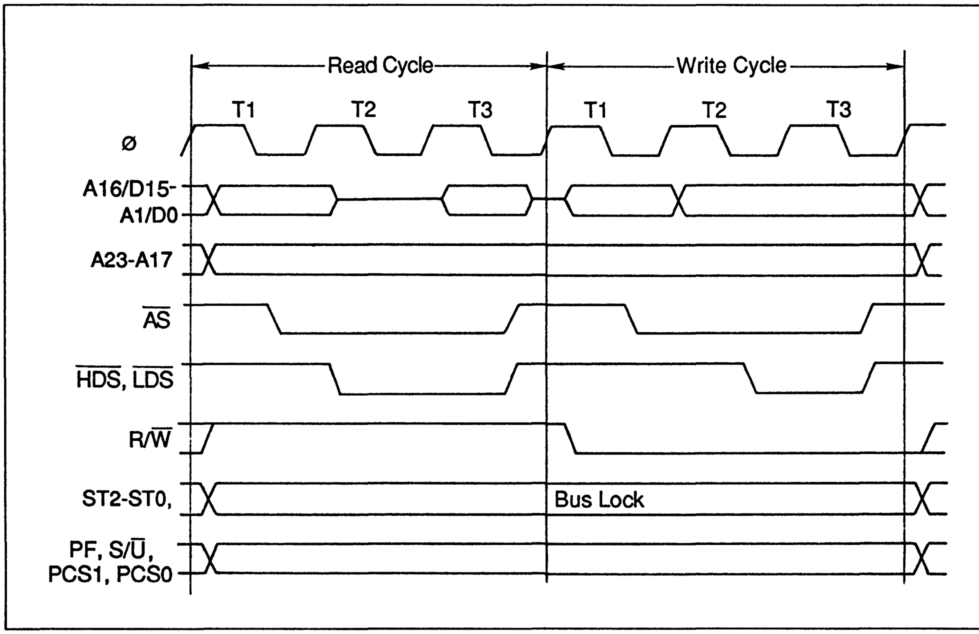


Figure 4-91. CPU Read Modify Write Cycle

Even addresses and odd addresses are located in the upper byte and lower byte of the 16-bit data bus. When the HD641016 accesses the upper byte of the data bus, HDS is asserted; when the HD641016 accesses the lower byte, LDS is asserted. Table 4-38 shows access size, start address, HDS and bus cycle relationship.

For byte-size memory write or memory write from an odd address, invalid data the same size as the unused byte in the write is output on the data bus.

Table 4-38. Size/Address and Bus Access

		Start Address					
	Number of Bus Cycles	**Even Address**			**Odd Address**		
		$\overline{HDS}$	**$\overline{LDS}$**		**$\overline{HDS}$**	**$\overline{LDS}$**	
Byte	1	0	1	Byte Access	1	0	Byte Access
Word	1	0	0	Word Access	1	0	Byte Access
	2	-	-	-	0	1	Byte Access
Long word	1	0	0	Word Access	1	0	Byte Access
	2	0	0	Word Access	0	0	Word Access
	3	-	-	-	0	1	Byte Access

Example: Long word access from the odd address
1st access: A byte is accessed with $\overline{HDS} = 1$ and $\overline{LDS} = 0$
2nd access: A word is accessed with $\overline{HDS} = 0$ and $\overline{LDS} = 0$
3rd access: A byte is accessed with $\overline{HDS} = 0$ and $\overline{LDS} = 1$

Notes: 0 = Low level
1 = High level

4.9.3 E Clock Timing

The HD641016 provides the E clock for easy interfacing with 6800 family peripherals. The E clock is produced by dividing the Ø clock by eight. Figure 4-92 shows E clock timing.

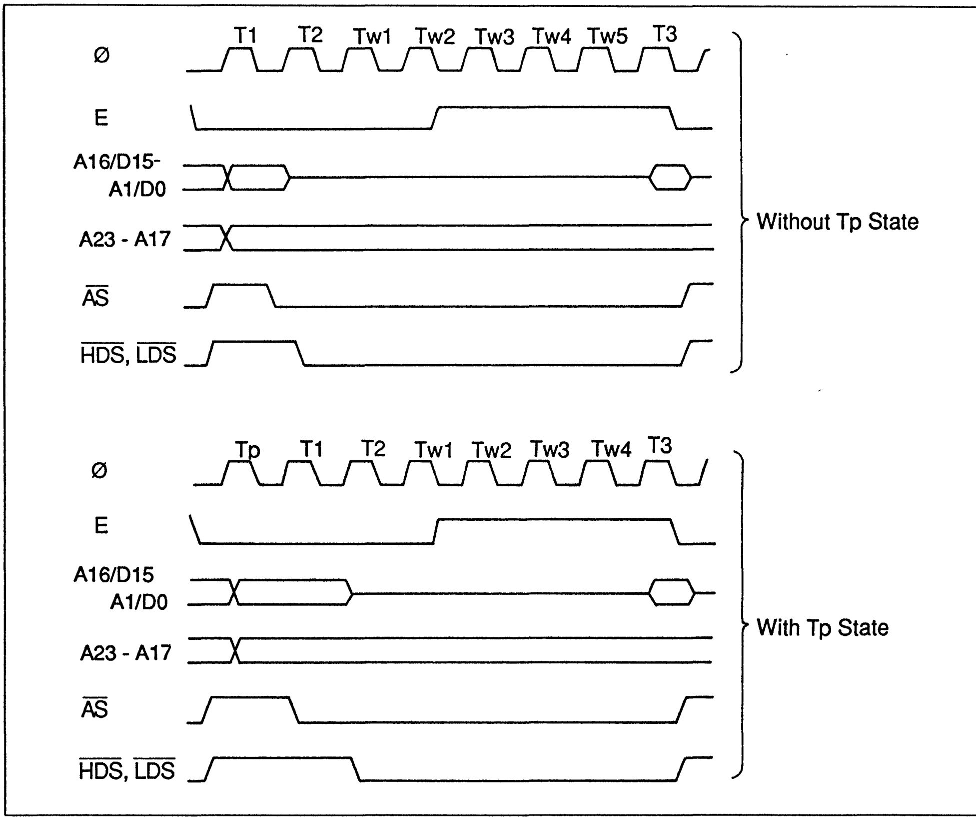


Figure 4-92. E Clock Timing

As shown in Figure 4-92, an instruction cycle synchronous with the E clock begins at the next Ø clock cycle after the E clock has pulled low. A Tw state is automatically inserted between T2 and T3 states. Note that hardware or software $\overline{\text{WAIT}}$ is ignored during this instruction cycle.

4.9.4 DRAM Refresh Cycle

Figure 4-93 shows a DRAM refresh cycle with no Tw states. See "Section 13. Dynamic RAM Refresh Controller" for details.

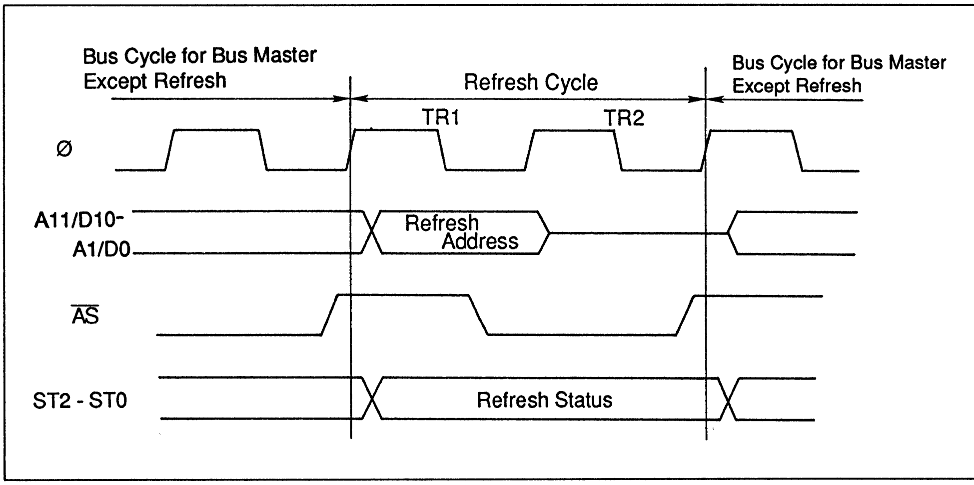


Figure 4-93. DRAM Refresh Cycle

4.10 Prefetch

The HD641016 provides an 8-byte prefetch queue (Figure 4-94) in which instructions from external memory are prefetched on a word basis. This allows the CPU to fetch instructions from the prefetch queue on a byte basis and execute them at high speed.

The HD641016 prefetches instructions in the following cases:

- When the prefetch queue is not full and the bus is not released to another master
- During a CPU instruction cycle (register-to-register transfer instruction cycles) which does not use buses, since the HD641016 prefetches instructions independently of the CPU operation
- During the internal RAM access cycle, since the CPU uses a specific internal bus for RAM access

The prefetch queue is reset in the following cases:

- Hardware reset (reset by $\overline{\text{RESET}}$ pin)
- Branch instructions such as in Bcc:G, BEQ, BNE, BRA, BQR or SCB instructions.
- Jump instructions such as in JMP or JSR instructions.
- Return instructions such as in RTD, RTE, RTR or RTS instructions.
- When the ANDC, LDC, ORC or XORC instruction is executed.

Figure 4-94 shows the prefetch queue. The fetch pointer FP indicates the next instruction address to be prefetched, while the PC of the HD641016 indicates the next instruction address to be executed, contained in the first byte of the prefetch queue.

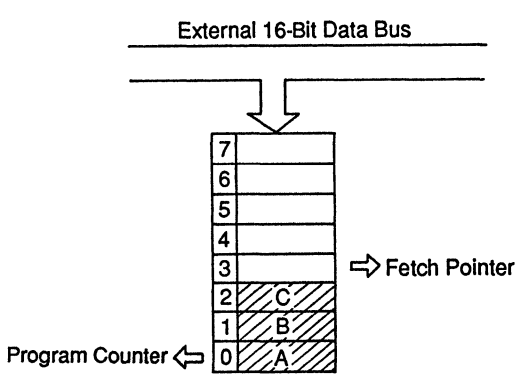


Figure 4-94. Prefetch Queue

4.10.1 Instruction Prefetch Sequence

Figure 4-95 demonstrates the instruction prefetch sequence.

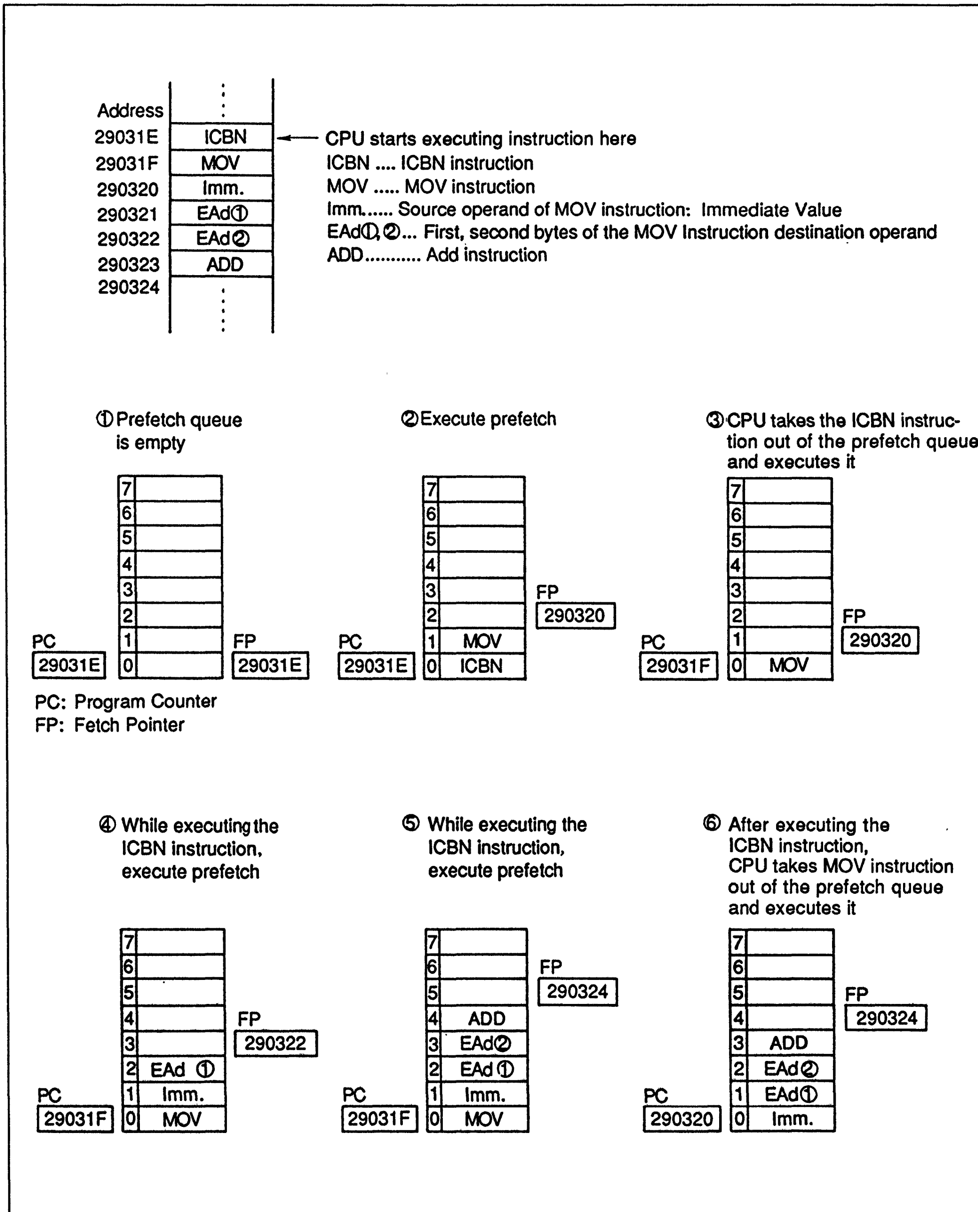


Figure 4-95. Instruction Prefetch Sequence

4.10.2 Prefetch Notes

The HD641016 can prefetch up to 8 bytes during a CPU instruction cycle requiring no bus operation. Accordingly, the user must note the following to prevent an exception from occurring.

Boundaries of Chip Select Areas and Internal RAM Area: Programs must not be located within 8 bytes from the upper limit of the chip select area, internal I/O, or internal RAM area. If they are, a bus error exception may occur when the HD641016 prefetches from outside the chip select area, internal I/O, or internal RAM area. See Figure 4-96.

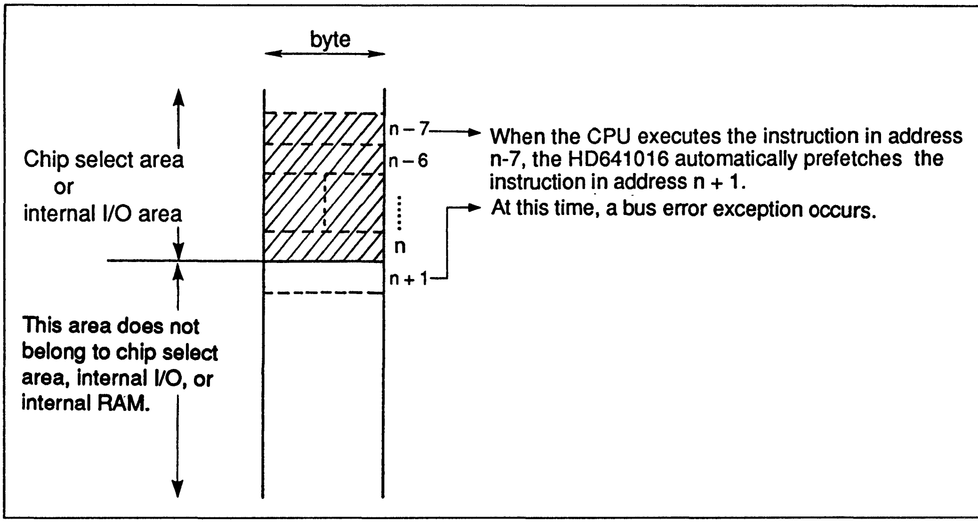


Figure 4-96. Instruction Prefetch Generating a Bus Error

Boundaries of Supervisor and User Areas: Programs must not be located within 8 bytes from the upper limit of the user and supervisor areas as shown in Figure 4-97. If they are, an access level violation exception may occur when the HD641016 prefetches from the supervisor area in user mode.

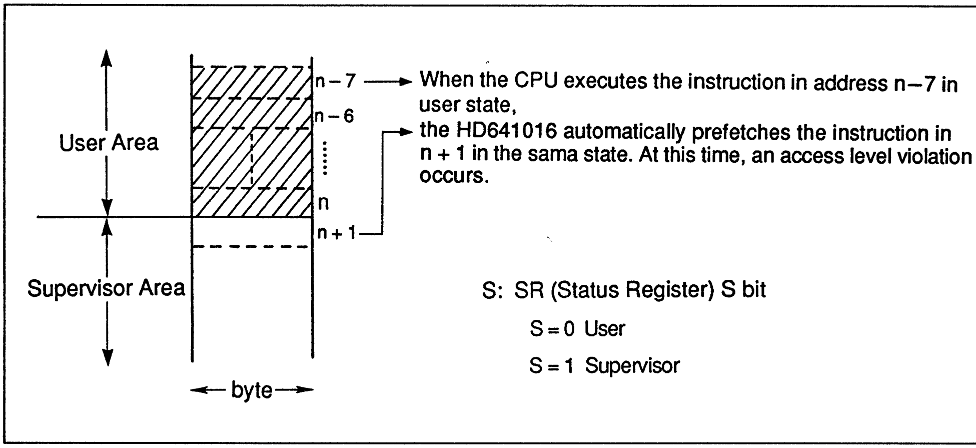


Figure 4-97. Instruction Prefetch Generating an Access Level Violation

Using an Enable Signal to Assert $\overline{\text{BRTRY}}$: In the system as shown in Figure 4-98, programs must not be located within 8 bytes from the memory area upper limit. If they are, a bus retry or bus error exception occurs when the HD641016 prefetches from an area other than the memory area.

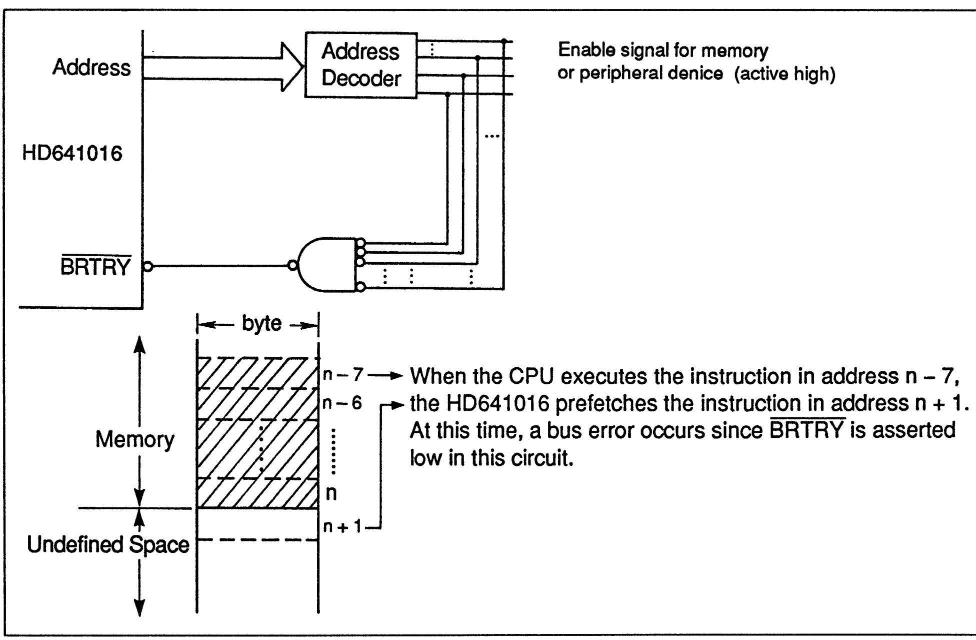


Figure 4-98. Instruction Prefetch Generating a Bus Retry or Bus Error

Reset Vector Fetch: If the HD641016 is reset, the EBR register is cleared to H'000000. The HD641016 then begins prefetch from H'000000. Accordingly, the HD641016 prefetches SSP from H'000000-H'000003 and PC from H'000004-H'000007. The CPU then fetches the SSP and PC from the prefetch queue and begins the instruction cycle. During this CPU cycle, the HD641016 prefetches from H'000008-H'00000F. Therefore, a bus error may occur, if the user system is designed to assert $\overline{\text{BRTRY}}$ when data is fetched from H'000008 or higher addresses. See Figure 4-99.

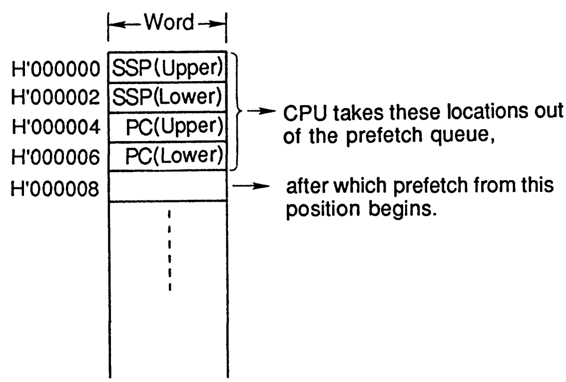


Figure 4-99. Instruction Prefetch Generating a Bus Error

4.11 Reset

The HD641016 supports two reset types (Table 4-39, Figures 4-100 to 4-101): power-on reset and manual reset under the control of the $\overline{RES}$, $\overline{BRTRY}$ and $\overline{NMI}$ pins.

Table 4-39. Reset

	Pin State			**Signal Assertion**	**Initialized**
Reset Type	$\overline{\textbf{RES}}$	$\overline{\textbf{BRTRY}}$	$\overline{\textbf{NMI}}$	**Period**	**Function**
Power-on reset	0	0	1	At least 12 clock cycles	CPU and all internal I/O devices
Manual reset	0	0	0	At least 12 clock cycles	CPU and internal I/O devices other than DRAM refresh controller

For power-on reset, $\overline{NMI}$ must be fixed to high before asserting $\overline{RES}$ and $\overline{BRTRY}$ for at least 12 clock cycles. However, for system power-on, power-on reset state must be maintained for at least 50 ms.

For manual reset, $\overline{NMI}$ must be fixed to low before asserting $\overline{RES}$ and $\overline{BRTRY}$ for at least 12 clock cycles, and $\overline{RES}$ and $\overline{BRTRY}$ are then negated to high.

Each pin's status changes three clock cycles after the HD641016 has been reset. See "Appendix D. Pin State" for details.

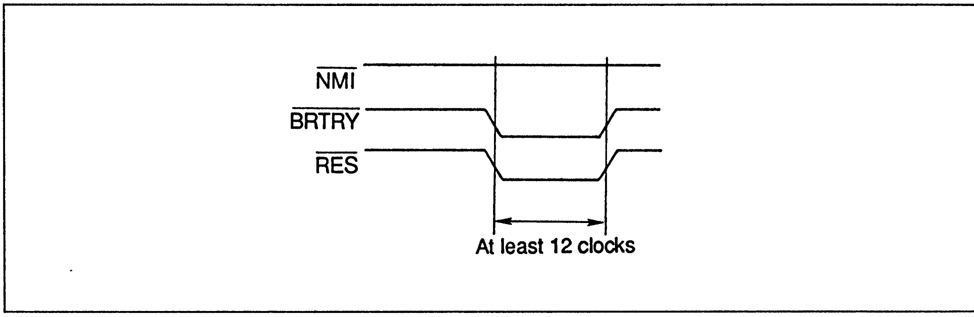


Figure 4-100. Power-On Reset Timing

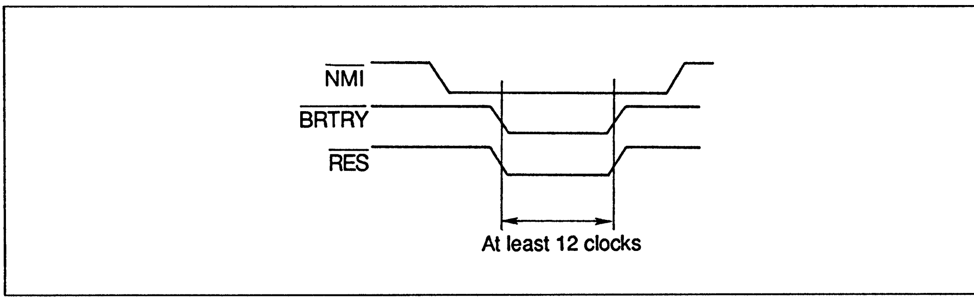


Figure 4-101. Manual Reset Timing

Section 5. RAM

5.1 Overview

The HD641016 contains 1024 bytes of high-speed internal RAM, which can be used as data memory space and as register banks. Since the internal RAM accesses long words aligned on long word boundaries, or words aligned on word boundaries, or byte in only two system clocks, it can be used as a high-speed stack area or working data area. However, misaligned data accesses requires four system clocks. In addition, three system clocks are required when the DMAC accesses a byte data or word data aligned on a word boundary. Moreover, six system clocks are required when the DMAC accesses a word data misaligned on a word boundary.

Internal RAM bytes are logically addressed by RA9-RA0. The logical address is translated into the physical address by adding bits 23-10 of the RBR register to RA9-RA0 (Figure 5-2). The CPU and DMAC access the internal RAM by the physical address. In addition, the start address of internal RAM is determined according to bits 23-10 of RBR.

See Figures 5-1 and 5-2 for details on RAM physical address calculation.

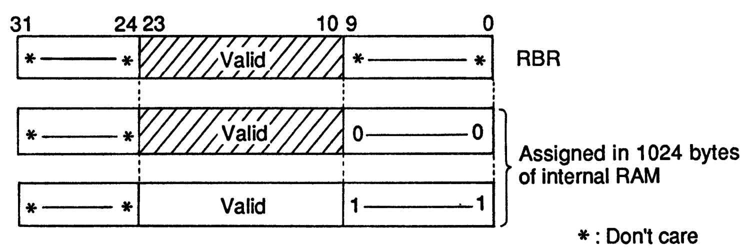


Figure 5-1. Address Assignment in Internal RAM

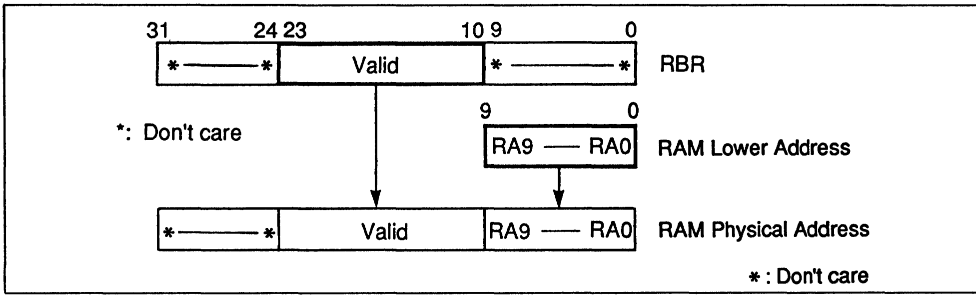


Figure 5-2. RAM Physical Address Calculation

Figure 5-3 shows the relationships between RBR, internal RAM, and HD641016 physical address space.

In Figure 5-3, internal RAM address offset and RBR are specified as H'17E and H'00000C00, respectively.

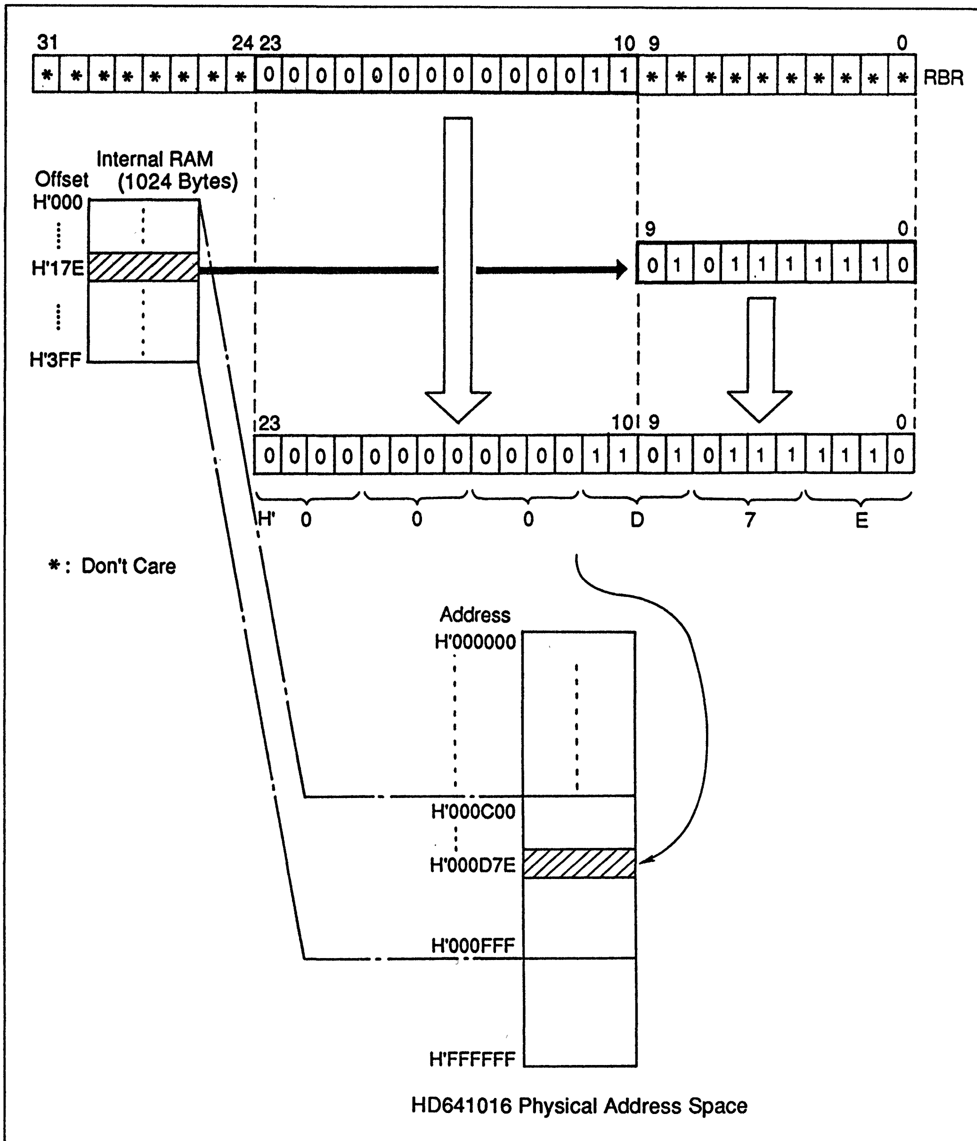


Figure 5-3. RBR, Internal RAM, and HD641016 Physical Address Space Relationships

5.2 RAM Configuration

Figure 5-4 shows the logical RAM configuration.

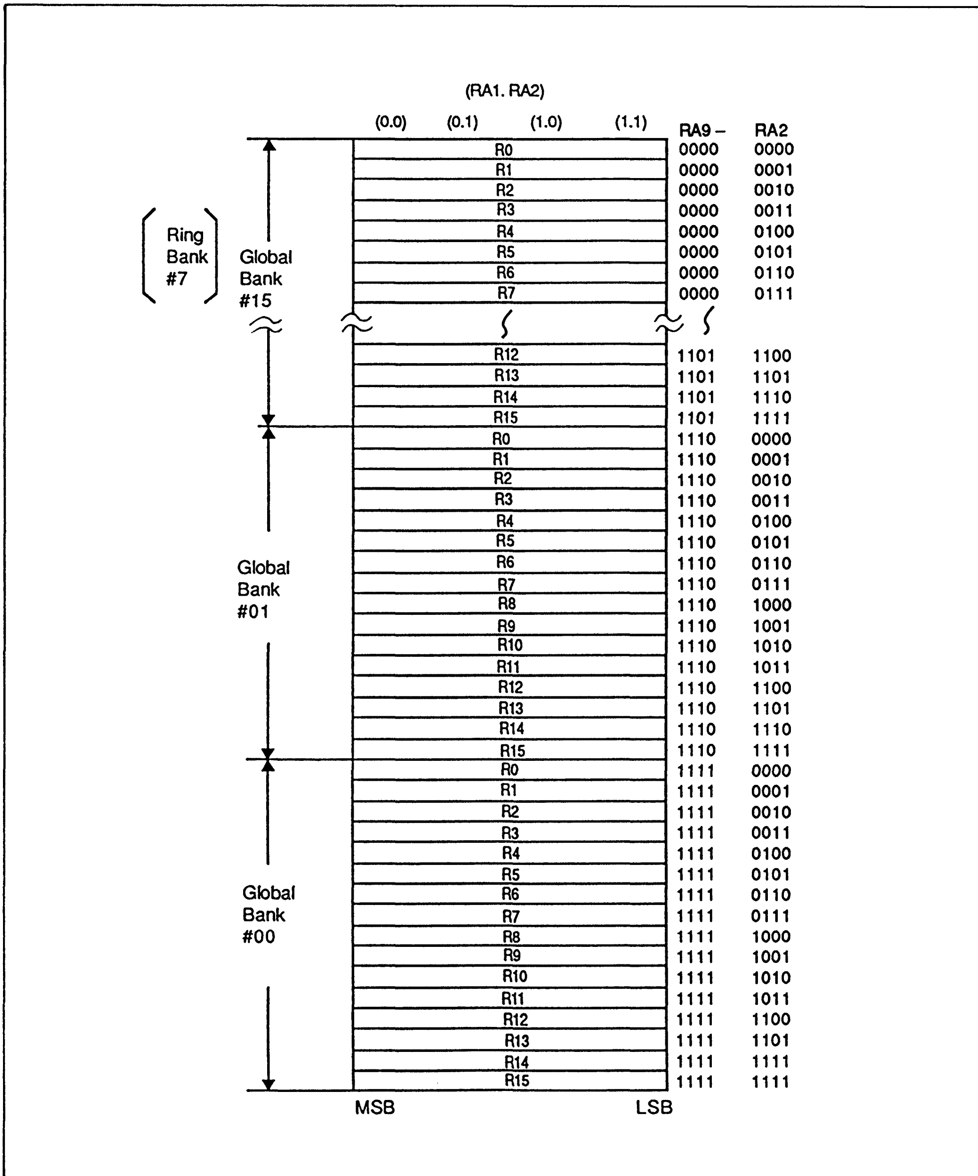


Figure 5-4. Logical RAM Configuration

5.3 RAM Access

As mentioned before, the internal RAM can be accessed as both data memory space and register banks. However, if a program is written to the internal RAM, the program will not be executed since the HD641016 does not fetch instructions from the internal RAM. If the CPU tries to access an instruction from an address assigned for the internal RAM, the instruction is fetched from external memory. Internal RAM access is controlled by the RAME, BPM, and RAMALV bits of the BMR register as shown below:

RAME: RAME = 0 disables the internal RAM access. At this time, peripheral devices of the same address can be accessed. RAME = 1 enables the internal RAM access.

BPM: If BPM = 0, register banks in the internal RAM cannot be used as data memory. At this time, register banks are read as undefined values and are not affected by write. If BPM = 1, the register banks can be used as both register banks and data memory.

RAMALV: If RAMALV = 0, the internal RAM can be accessed in user state. If RAMALV = 1, the internal RAM can be accessed only in supervisor state. If the internal RAM is accessed in user state while RAMALV=1, an access level violation occurs.

See "4.3.9 Bank Mode Register (BMR)" for details.

5.4 RAM Access Cycle

During the internal RAM access cycle, the HD641016 does not output the AS, HDS, and LDS signals to external devices. Accordingly, external devices are not affected when an internal RAM cycle is executed. External memory or internal I/O access, and internal RAM access are shown in Figures 5-5 and 5-6, respectively.

Figures 5-7 and 5-8 show internal RAM access examples in special cases. When accessing word or long word data which starts in the chip select area or I/O area and continues into internal RAM, the data access is performed as shown in Figure 5-7. In addition, when accessing word or long word data whose first bytes are the upper limit of the internal RAM, data is accessed within the internal RAM: the first bytes from the upper limit of the internral RAM, the following bytes from the lower limit of the internal RAM as shown in Figure 5-8. The internal RAM is undefined at reset.

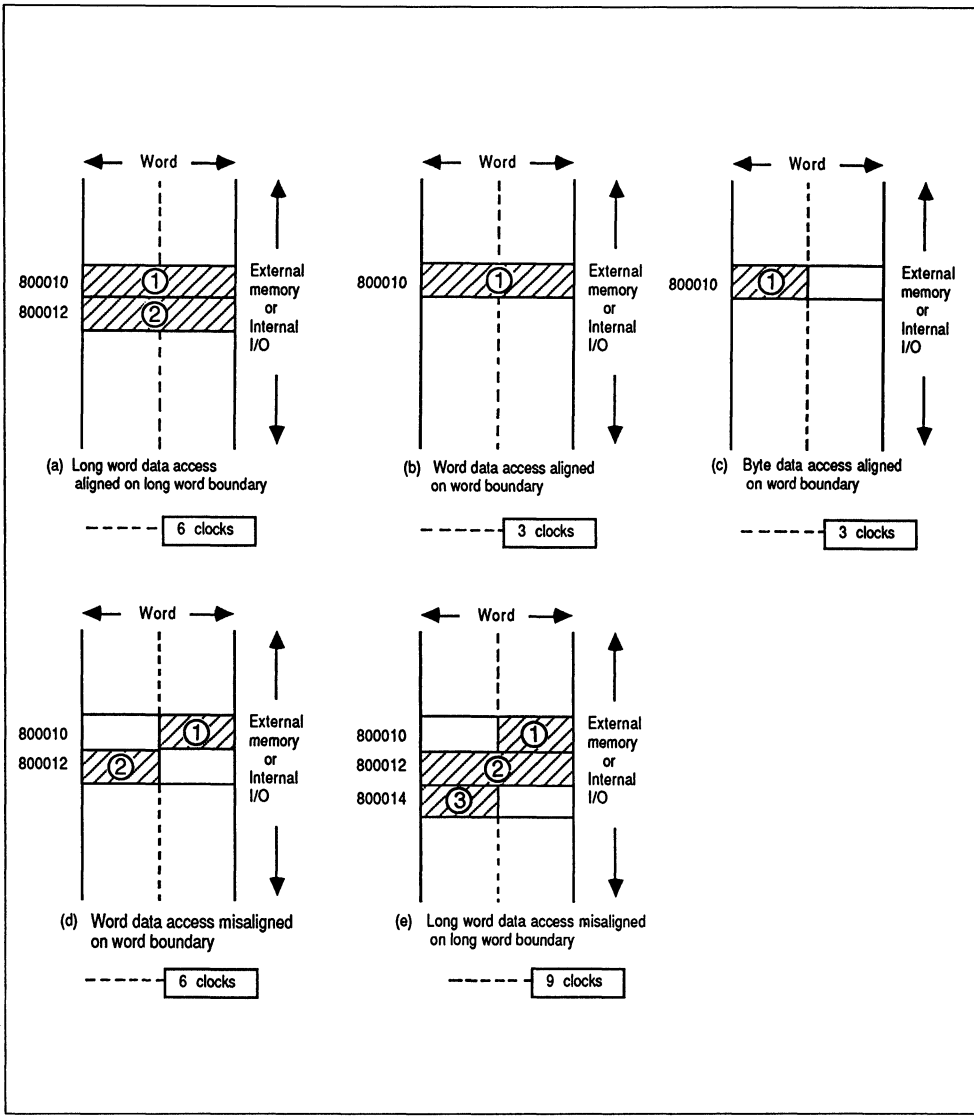


Figure 5-5. Number of Clocks When Accessing External Memory (No Wait States) or Internal I/O (Circled numbers indicate access sequence)

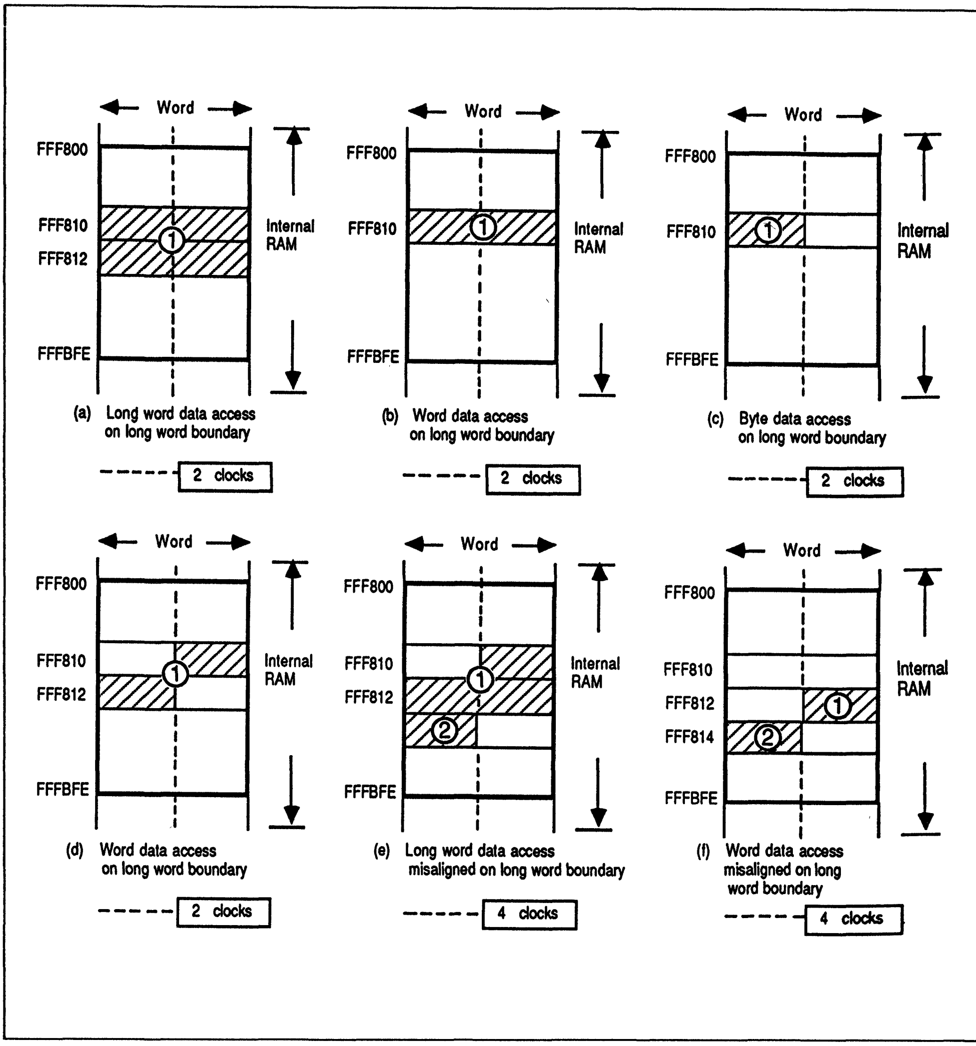


Figure 5-6. Number of Clocks When Accessing Internal RAM (Data on long word boundary can be accessed by 2 clocks regardless of size)

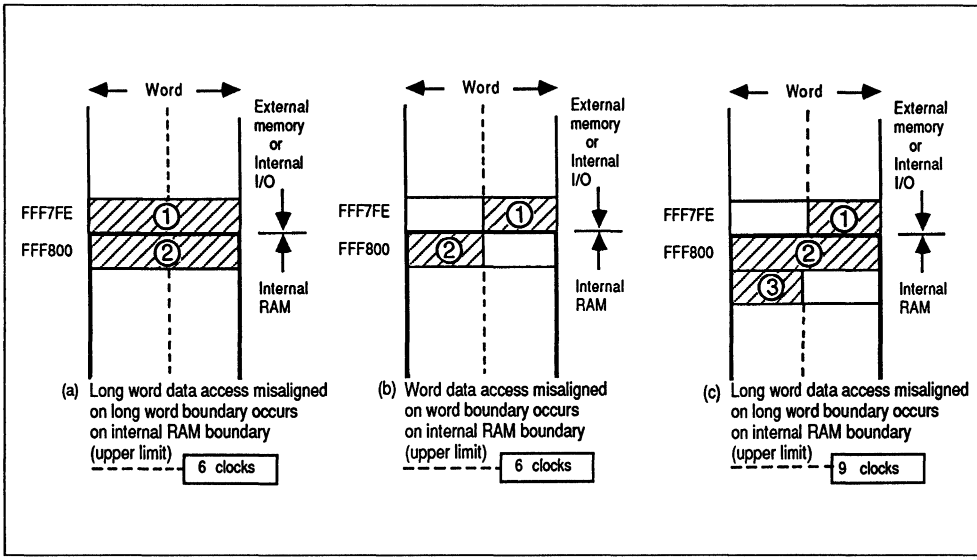


Figure 5-7. Number of Clocks When Accessing from External Memory or Internal I/O to Internal RAM (No external memory wait states)

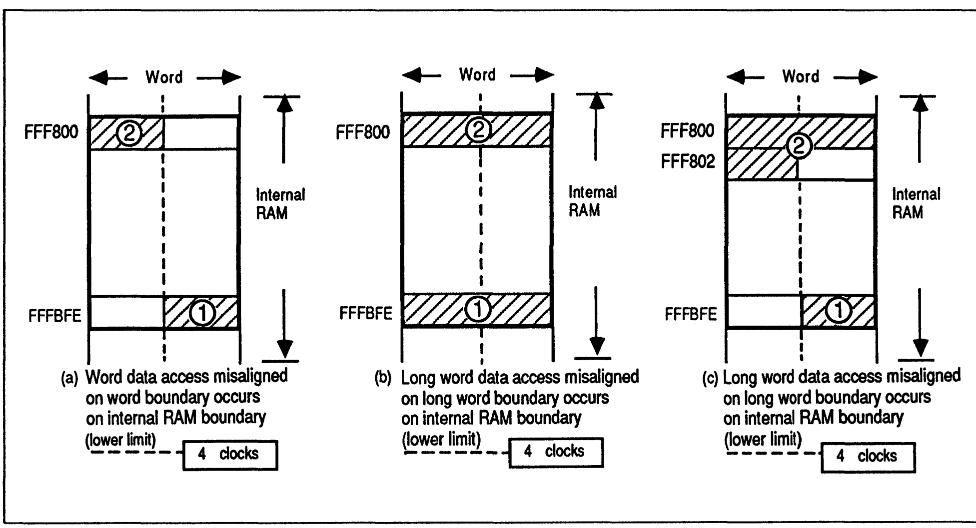
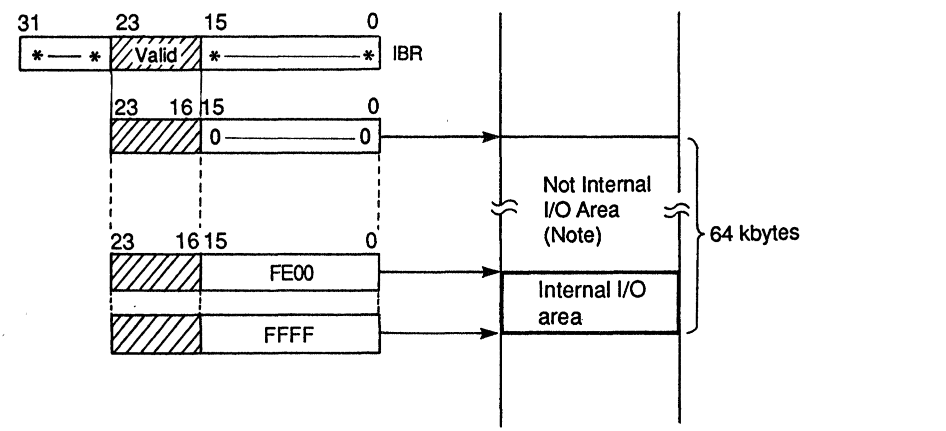


Figure 5-8. Number of Clocks When Accessing from Upper Limit of Internal RAM

Section 6. Internal I/O

6.1 Overview

The HD641016 incorporates a DMA controller, timers, asynchronous serial communication interface, interrupt controller, DRAM refresh controller, wait state controller, chip select controller, and peripheral controller in a single chip. The internal I/O is relocatable on 64-kbyte boundaries under the control of the internal I/O base register (IBR). However, internal I/O are actually located in addresses whose offsets are H'FE00 to H'FFFF. Addresses of offsets H'0000 to H'FDFF are then handled as external memory or device areas. Accordingly, if addresses with offsets H'0000 to H'FDFF are accessed, external memory or devices are actually accessed. See Figure 6-1.



Note: If this area is accessed, external memory or device is accessed.

Figure 6-1. Internal I/O Access

The internal I/O access cycle consists of three states: T1, T2, and T3. In the internal I/O cycle, software or hardware $\overline{WAIT}$ state insertion is ignored. However, the ASWC bit of the memory control register (MCR) is valid. Accordingly, ASWC = 1 enables insertion of a Tp state prior to the T1 state. External devices cannot detect the internal I/O cycle since the HD641016 does not output the $\overline{AS}$, $\overline{HDS}$, and $\overline{LDS}$ signals. Note that the internal I/O can be accessed only in supervisor state. If it is accessed in user state, an access level violation is generated. See "Section 11. Chip Select Controller" concerning the overlap of the internal I/O area with the internal RAM or chip select area.

6.2 I/O Register Listing

The internal I/O registers are listed in Table 6-1. "Reserved" indicates areas reserved for future expansion. The reserved areas are always read as 0.

Table 6-1. I/O Register Listing

Internal I/O Name	Register Name	Symbol	Offset	R/W	Initial Value	Size
——	(Reserved)	——	H'FE00 H'FF27	—	——	—
Chip Select Controller	Area base register 0	ABR0	H'FF28	R/W	H'0000	W
	Area range register 0	ARR0	H'FF2A	R/W	H'0000	W
	Area wait control register 0	AWCR0	H'FF2C	R/W	H'0047	W
	Area base register 1	ABR1	H'FF2E	R/W	H'0000	W
	Area range register 1	ARR1	H'FF30	R/W	H'0000	W
	Area wait control register 1	AWCR1	H'FF32	R/W	H'0047	W
	Area base register 2	ABR2	H'FF34	R/W	H'0000	W
	Area range register 2	ARR2	H'FF36	R/W	H'0000	W
	Area wait control register 2	AWCR2	H'FF38	R/W	H'0047	W
	Area base register 3	ABR3	H'FF3A	R/W	H'0000	W
	Area range register 3	ARR3	H'FF3C	R/W	H'0000	W
	Area wait control register 3	AWCR3	H'FF3E	R/W	H'0047	W

Table 6-1. I/O Register Listing (cont.)

Internal I/O Name	Register Name	Symbol	Offset	R/W	Initial Value	Size
Interrupt Controller	Interrupt priority register 0	IPR0	H'FF40	R/W	H'0000	W
	Interrupt priority register 1	IPR1	H'FF42	R/W	H'0000	W
	Interrupt priority register 2	IPR2	H'FF44	R/W	H'0000	W
	Interrupt control register	ICR	H'FF46	R/W	H'0000	W
—	(Reserved)	—	H'FF48 H'FF4D	—	—	—
Peripheral Controller	Peripheral control register 0	PCR0	H'FF4E	R/W	H'0000	W
—	(Reserved)	—	H'FF50 H'FF57	—	—	—
Asyn-chronous Serial Commu-nication Interface 0 (Note 1)	TX/RX buffer register	TRB	H'FF58	R/W	Undefined	
	Status register 0	ST0	H'FF59	R	H'00	B
	Status register 1	ST1	H'FF5A	R/W	H'00	B
	Status register 2	ST2	H'FF5B	R/W	H'00	B
	Status register 3	ST3	H'FF5C	R	H'00 (Note 2)	B
	(Reserved)	—	H'FF5D	—	—	—
	Interrupt enable register 0	IE0	H'FF5E	R/W	H'00	B
	Interrupt enable register 1	IE1	H'FF5F	R/W	H'00	B

Table 6-1. I/O Register Listing (cont.)

Internal I/O Name	Register Name	Symbol	Offset	R/W	Initial Value	Size
Asyn-chronous Serial Commu-nication Interface 0 (Note 1) (cont.)	Interrupt enable register 2	IE2	H'FF60	R/W	H'00	B
	(Reserved)	—	H'FF61	—	—	—
	Command register	CMD	H'FF62	W	(Note 3) —	B
	Mode register 0	MD0	H'FF63	R/W	H'00	B
	Mode register 1	MD1	H'FF64	R/W	H'00	B
	Mode register 2	MD2	H'FF65	R/W	H'00	B
	Control register	CTL	H'FF66	R/W	H'01	B
	(Reserved)	—	H'FF67	—	—	—
	(Reserved)	—	H'FF68	—	—	—
	(Reserved)	—	H'FF69	—	—	—
	Time constant register	TMC	H'FF6A	R/W	H'01	B
	RX clock source register	RXS	H'FF6B	R/W	H'00	B
	TX clock source register	TXS	H'FF6C	R/W	H'00	B
	(Reserved)	—	H'FF6D	—	—	—
	(Reserved)	—	H'FF6E	—	—	—
	(Reserved)(Note 4)	—	H'FF6F	—	—	—

Table 6-1. I/O Register Listing (cont.)

Internal I/O Name	Register Name	Symbol	Offset	R/W	Initial Value	Size
Asynchronous Serial Communications Interface 1 (Note 1) (cont.)	TX/RX buffer register	TRB	H'FF70	R/W	Undefined	B
	Status register 0	ST0	H'FF71	R	H'00	B
	Status register 1	ST1	H'FF72	R/W	H'00	B
	Status register 2	ST2	H'FF73	R/W	H'00	B
	Status register 3	ST3	H'FF74	R	(Note 2) H'00	B
	(Reserved)	—	H'FF75	—	—	—
	Interrupt enable register 0	IE0	H'FF76	R/W	H'00	B
	Interrupt enable register 1	IE1	H'FF77	R/W	H'00	B
	Interrupt enable register 2	IE2	H'FF78	R/W	H'00	B
	(Reserved)	—	H'FF79	—	—	—
	Command register	CMD	H'FF7A	W	(Note 3) —	B
	Mode register 0	MD0	H'FF7B	R/W	H'00	B
	Mode register 1	MD1	H'FF7C	R/W	H'00	B
	Mode register 2	MD2	H'FF7D	R/W	H'00	B
	Control register	CTL	H'FF7E	R/W	H'01	B
	(Reserved)	—	H'FF7F	—	—	—
	(Reserved)	—	H'FF80	—	—	—

Table 6-1. I/O Register Listing (cont.)

Internal I/O Name	Register Name	Symbol	Offset	R/W	Initial Value	Size
Asynchronous Serial Communications Interface 1 (Note 1) (cont.)	(Reserved)	—	H'FF81	—	—	—
	Time constant register	TMC	H'FF82	R/W	H'01	B
	RX clock source register	RXS	H'FF83	R/W	H'00	B
	TX clock source register	TXS	H'FF84	R/W	H'00	B
	(Reserved)	—	H'FF85	—	—	—
	(Reserved)	—	H'FF86	—	—	—
	(Reserved)(Note 4)	—	H'FF87	—	—	—
—	(Reserved)	—	H'FF88 H'FF8D	—	—	—
Timer 1	Upcount register	UCR	H'FF8E	R/W	H'0000	W
	Count compare register A	CCRA	H'FF90	R/W	H'FFFF	W
	Count compare register B	CCRB	H'FF92	R/W	H'FFFF	W
	Control register	CNTR	H'FF94	R/W	H'0000	W
	Status register	STR	H'FF96	R	H'0000	W
Timer 2	Upcount register	UCR	H'FF98	R/W	H'0000	W
	Count compare register A	CCRA	H'FF9A	R/W	H'FFFF	W
	Count compare register B	CCRB	H'FF9C	R/W	H'FFFF	W

Table 6-1. I/O Register Listing (cont.)

Internal I/O Name	Register Name	Symbol	Offset	R/W	Initial Value	Size
Timer 2 (cont.)	Control register	CNTR	H'FF9E	R/W	H'0000	W
	Status register	STR	H'FFA0	R	H'0000	W
———	(Reserved)	——	H'FFA2 H'FFAB	—	—	—
———	(Reserved)	——	H'FFAC H'FFAF	—	—	—
DMA Controller 0	Memory address register	MADR	H'FFB0	R/W	Undefined	L
	Device/Next block address register	DADR/NADR	H'FFB4	R/W	Undefined	L
	Execute transfer count register	ETCR	H'FFB8	R/W	Undefined	W
	Base transfer count register	BTCR	H'FFBA	R/W	Undefined	W
	Channel control register	CHCRA	H'FFBC	R/W	H'0000	W
	Channel control register B	CHCRB	H'FFBE	R/W	H'0000	W
DMA Controller 1	Memory address register	MADR	H'FFC0	R/W	Undefined	L
	Device/Next block address register	DADR/NADR	H'FFC4	R/W	Undefined	L
	Execute transfer count register	ETCR	H'FFC8	R/W	Undefined	W
	Base transfer count register	BTCR	H'FFCA	R/W	Undefined	W

Table 6-1. I/O Register Listing (cont.)

Internal I/O Name	Register Name	Symbol	Offset	R/W	Initial Value	Size
DMA Controller 1 (cont.)	Channel control register A	CHCRA	H'FFCC	R/W	H'0000	W
	Channel control register B	CHCRB	H'FFCE	R/W	H'0000	W
DMA Controller 2	Memory address register	MADR	H'FFD0	R/W	Undefined	L
	Device/Next block address register	DADR/NADR	H'FFD4	R/W	Undefined	L
	Execute transfer count register	ETCR	H'FFD8	R/W	Undefined	W
	Base transfer count register	BTCR	H'FFDA	R/W	Undefined	W
	Channel control register A	CHCRA	H'FFDC	R/W	H'0000	W
	Channel control register B	CHCRB	H'FFDE	R/W	H'0000	W
DMA Controller 3	Memory address register	MADR	H'FFE0	R/W	Undefined	L
	Device/Next block address register	DADR/NADR	H'FFE4	R/W	Undefined	L
	Execute transfer count register	ETCR	H'FFE8	R/W	Undefined	W
	Base transfer count register	BTCR	H'FFEA	R/W	Undefined	W
	Channel control register A	CHCRA	H'FFEC	R/W	H'0000	W
	Channel control register B	CHCRB	H'FFFE	R/W	H'0000	W
All DMA Controller	Operation control register	OPCR	H'FFF0	R/W	H'0000	W

Table 6-1. I/O Register Listing (cont.)

Internal I/O Name	Register Name	Symbol	Offset	R/W	Initial Value	Size
—	(Reserved)	—	H'FFF2 H'FFF7	—	—	—
Memory Control	Memory control register	MCR	H'FFF8	R/W	H'F0E0	W
—	(Reserved)	—	H'FFFA H'FFFF	—	—	—

Notes: 1. The ASCI registers are located in consecutive addresses on a byte basis. They cannot be accessed on a word or long word basis.

2. Bits 3 and 2 indicates the $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$ pin levels, respectively.

3. Always read as 0.

4. Always read as undefined.

Section 7. Bus Arbitrator

7.1 Overview

The HD641016 incorporates a bus arbitrator to allocate bus mastership between the CPU, internal DMAC, DRAM refresh controller, and external bus masters. Figure 7-1 shows the bus arbitrator to bus masters interface. The bus arbitrator is connected to the internal bus masters through the REQ and ACK signals. It is interfaced to an external bus master through the $\overline{\text{BREQ}}$ and $\overline{\text{BACK}}$ signals.

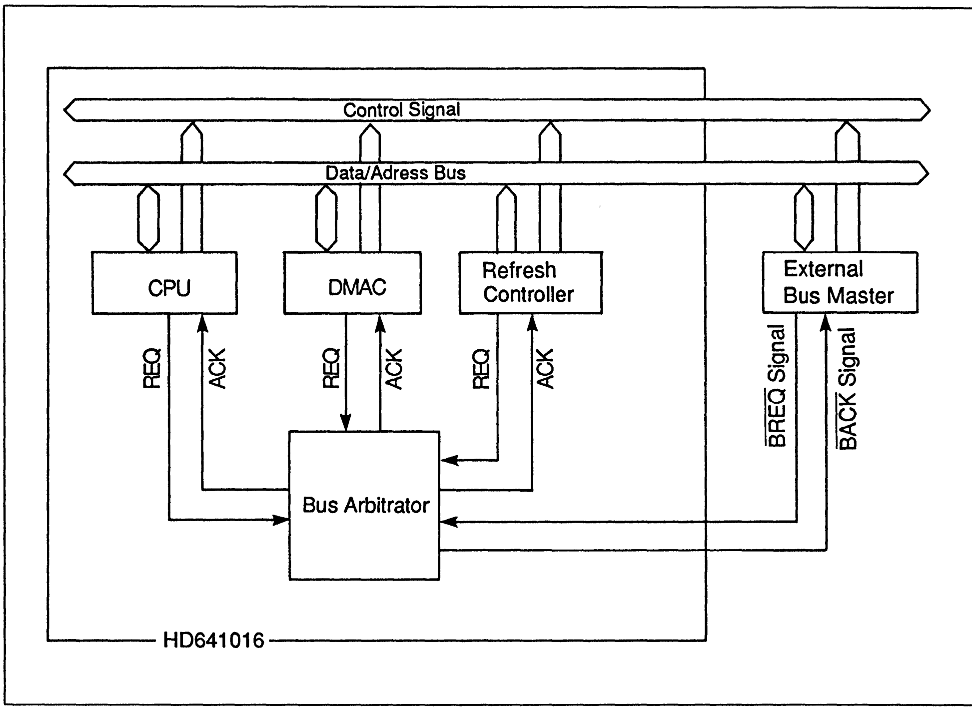


Figure 7-1. Bus Arbitrator to Bus Masters Interface

7.2 Bus Arbitrator Operation

If the REQ signal is low, the bus arbitrator outputs ACK to the bus master. If multiple bus masters assert REQ simultaneously, the bus arbitrator returns ACK to the highest priority bus master. Upon receiving ACK, the bus master can use the bus until ACK is negated. See "Section 13. DRAM Refresh Controller" for details on bus master priorities.

The bus arbitrator always samples REQ and checks the priority of the bus master requesting bus mastership. The bus arbitrator then releases the bus for each bus master at the specific timing described below.

7.3 Bus Master Arbitration

7.3.1 CPU Is Current Bus Master

If the CPU is the current bus master, the bus arbitrator can release the bus for another bus master while the CPU is not using the bus. However, the bus arbitrator will not release the bus in the following cases:

- Between the read and write cycles in read-modify-write cycles for the TAS instruction (Figure 7-2)
- When $\overline{\text{BRTRY}}$ is low level and the CPU begins the bus retry cycle (Figure 7-3)
- During the interrupt acknowledge cycle in double acknowledge mode (Figure 7-4)
- During operations requiring 2 or more bus cycles, such as long word accesses or word accesses from odd addresses

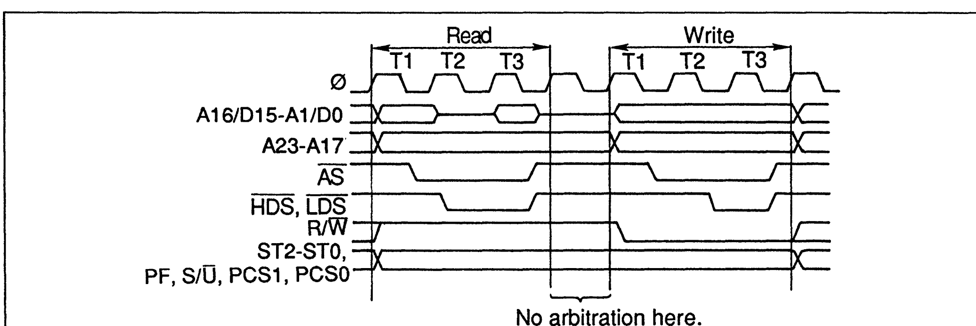


Figure 7-2. Timing Example 1

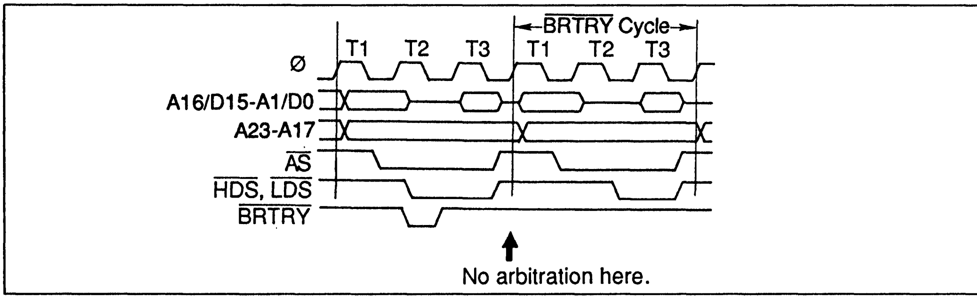


Figure 7-3. Timing Example 2

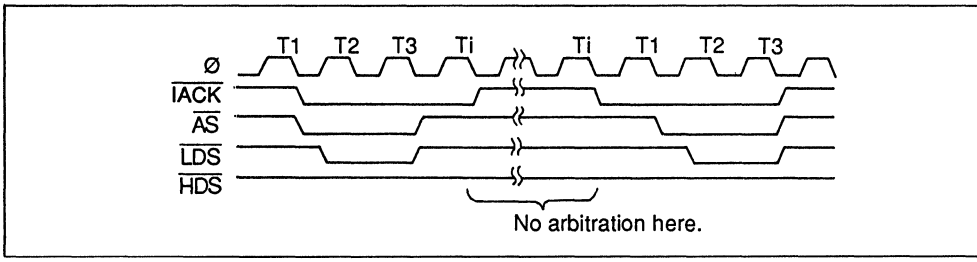


Figure 7-4. Timing Example 3

7.3.2 DMAC Is Current Bus Master

The DMA controller can release the bus for a higher priority bus master at the breakpoint of bus cycles.

In burst mode, if a higher priority bus master requests bus mastership, the DMA controller stops transfer and releases the bus. The DMA controller restarts the remaining DMA transfer after the higher priority bus master has completed its bus cycle.

In dual address mode, the DMA controller executes a DMA transfer in two consecutive cycles: read and write cycles. Bus arbitration is not performed during these two cycles. However, if a bus error is generated during the read cycle, the DMA controller stops DMA transfer immediately without executing the write cycle and releases the bus for the other bus master. When two or more express channels start transfer simultaneously, the DMA controller also releases the bus for the other bus master at every express channel transfer completion.

7.3.3 DRAM Refresh Controller Is Current Bus Master

The DRAM refresh controller releases the bus for the other bus master at the breakpoint of the refresh cycle. However, note that bus arbitration is not performed until the refresh counter decrements to 0 while refresh priority level (RPL) bit is set in burst refresh cycle.

7.3.4 External Bus Master Is Current Bus Master

See "4.9 Bus Release" for details.

Section 8. DMA Controller (DMAC)

8.1 Overview

The HD641016 DMA controller has four independent high-speed DMA channels. Each channel can perform high-speed data transfer without intervention of the CPU in single or dual address modes. In single address mode, a continue operation can be specified to transfer several blocks of data continuously.

In addition to memory-to/from-memory and memory-to/from-I/O device data transfers, the DMAC can transfer data between memory and the internal ASCIs, timers, and RAM.

The DMAC has the following functions:

- Address space: 16 Mbytes
- Transfer unit: byte or word
- Byte/word transfer capacity: 64 kbytes or 64 kwords
- Maximum transfer rate: 3.33 Mwords/second (10-MHz version, single address mode)
- Address modes: single or dual
- DMA transfer request: external or auto
  —Transfers to/from the internal ASCIs, timers
  —External requests edge or level triggered
- DMA transfers
  —External memory to/from I/O device with $\overline{\text{DACK}}$ signal in single address mode
  —External memory to/from memory mapped I/O, external memory, internal RAM, internal ASCI, internal timer, or internal I/O registers other than DMAC in dual address mode
- Bus modes: burst, obedient, or cycle steal
- Continue operation in single address mode
- Priority: normal or express
  —Express channels have priority over normal channels
  —Normal channel priority is rotated, but express channel holds channel until transfer is complete

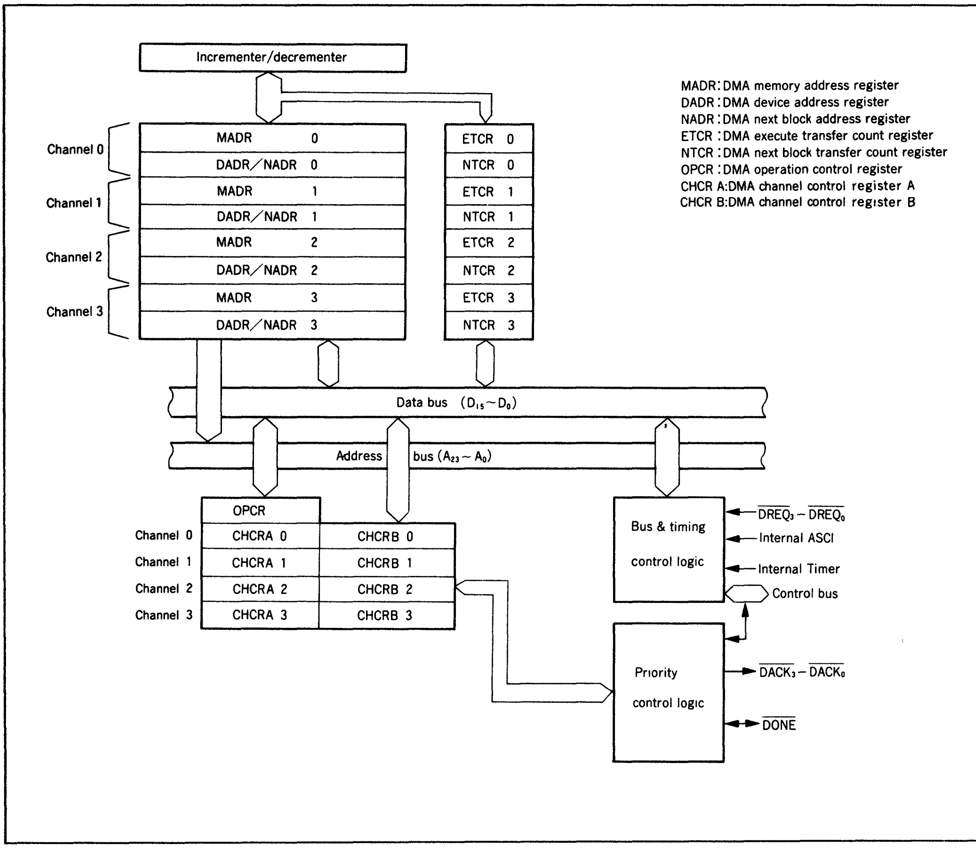


Figure 8-1. DMAC Block Diagram

8.2 DMAC Registers

Table 8-1 lists the DMAC registers.

Table 8-1. DMAC Register

Channel	Name	Symbol	Address Offset	R/W	Initial Value	Size
0	DMA memory address register ch 0	MADR0	H'FFB0	R/W	Undefined	L
	DMA device address register/ next block address register ch 0	DADR/ NADR0	H'FFB4	R/W	Undefined	L
	DMA execute transfer count register ch 0	ETCR0	H'FFB8	R/W	Undefined	W
	DMA next block transfer count register ch 0	NTCR0	H'FFBA	R/W	Undefined	W
	DMA channel control register A ch 0	CHCRA0	H'FFBC	R/W	H'0000	W
	DMA channel control register B ch 0	CHCRB0	H'FFBE	R/W	H'0000	W
1	DMA memory address register ch 1	MADR1	H'FFC0	R/W	Undefined	L
	DMA device address register/ next block address register ch 1	DADR/ NADR1	H'FFC4	R/W	Undefined	L
	DMA execute transfer count register ch 1	ETCR1	H'FFC8	R/W	Undefined	W
	DMA next block transfer count register ch 1	NTCR1	H'FFCA	R/W	Undefined	W
	DMA channel control register A ch 1	CHCRA1	H'FFCC	R/W	H'0000	W
	DMA channel control register B ch 1	CHCRB1	H'FFCE	R/W	H'0000	W

Table 8-1. DMAC Register (cont.)

Channel	Name	Symbol	Address Offset	R/W	Initial Value	Size
2	DMA memory address register ch 2	MADR2	H'FFD0	R/W	Undefined	L
	DMA device address register/ next block address register ch 2	DADR/ NADR2	H'FFD4	R/W	Undefined	L
	DMA execute transfer count register ch 2	ETCR2	H'FFD8	R/W	Undefined	W
	DMA next block transfer count register ch 2	NTCR2	H'FFDA	R/W	Undefined	W
	DMA channel control register A ch 2	CHCRA2	H'FFDC	R/W	H'0000	W
	DMA channel control register B ch 2	CHCRB2	H'FFDE	R/W	H'0000	W
3	DMA memory address register ch 3	MADR3	H'FFE0	R/W	Undefined	L
	DMA device address register/ next block address register ch 3	DADR/ NADR3	H'FFE4	R/W	Undefined	L
	DMA execute transfer count register ch 3	ETCR3	H'FFE8	R/W	Undefined	W
	DMA next block transfer count register ch 3	NTCR3	H'FFEA	R/W	Undefined	W
	DMA channel control register A ch 3	CHCRA3	H'FFEC	R/W	H'0000	W
	DMA channel control register B ch 3	CHCRB3	H'FFEE	R/W	H'0000	W
All	Operation control register	OPCR	H'FFF0	R/W	H'0000	W

8.2.1 Memory Address Register Channels 3-0 (MADR3-MADR0)

DMAC channels 3-0 have identical registers MADR3-MADR0 with identical functions. The 32-bit read/write MADR register points to the DMA operation memory address. The lower 24 bits are valid and can specify 16M memory addresses. The upper 8 bits are reserved for future expansion. MADR is not initialized by reset.

MADR is set to the start or last address at initialization and is updated every byte/word transfer to point to the next transfer data location. The MRC bit of the channel control register A (CHCRA) determines the update. In addition, the update step (±1 or ±2) is automatically determined by the transfer operand size or device port size. See Figure 8-2.

31 — 24	23 — 0
—	Vaild

—: Reserved bit. Always read as 0.
Cannot be written.

Figure 8-2. Memory Address Register (MADR)

8.2.2 Device Address Register/Next Block Address Register Channels 3-0 (DADR3/NADR3-DADR0/NADR0)

DMAC channels 3-0 have identical 32-bit read/write registers DADR/NADR3-DADR/NADR0 (Figure 8-3) with identical functions. The lower 24 bits of the DADR/NADR register are valid and the upper 8 bits are reserved for future expansions. The DADR/NADR register performs different functions depending on the transfer mode used. In the dual address mode (see "8.3 DMAC Operation and Procedures" for details), for memory to/from memory or memory to/from memory mapped I/O transfer, it specifies the device address (or memory space address). In this case, it is updated like the MADR register.

In single address continue operation mode, this register holds the first (or last) address of the next block transfer. (Blocks are defined as a group of data which can be transferred by programming MADR and ETCR together. Therefore, block data exists within a memory area of 64 kbytes or kwords.) DADR/NADR is set at initialization or during block transfer. The DMAC transfers the contents of DADR/NADR into MADR upon completion of the block transfer.

In single address mode without continue, DADR/NADR has no effect on DMAC operation.

31 — 24	23 — 0
—	Vaild

—: Reserved bit. Always read as 0.
Cannot be written.

Figure 8-3. Device Address/ Next Block Address Register (DADR/NADR)

8.2.3 Execute Transfer Count Register Channels 3-0 (ETCR3-ETCR0)

DMAC channels 3-0 have identical registers ETCR3-ETCR0 with identical functions. The ETCR register is a 16-bit read/write register that counts the number of bytes/words (up to 64 kbytes/kwords) to be transferred. To transfer N bytes/words (block length), the user should set the ETCR register to N. Then ETCR is decremented by one after every byte/word transfer. When N bytes/words have been transferred, ETCR becomes 0, and the DMAC terminates the transfer. If ETCR is specified as H'0000, 64 kbytes/kwords will be transferred.

Note that ETCR can be read or written by the CPU.

8.2.4 Next Block Transfer Count Register Channels 3-0 (NTCR3-NTCR0)

The DMAC channels 3-0 have identical registers NTCR3-NTCR0 with identical functions. In the single address continue operation mode, the NTCR register holds the number of bytes/words to be transferred for the next block transfer. The user should set NTCR by software during initialization or during block transfer. The contents of the ETCR register and those of the NTCR register are automatically exchanged when a block transfer is completed.

In other modes, NTCR has no effect on DMA operation.

Note that NTCR can be read or written by the CPU.

8.2.5 Channel Control Register A Channels 3-0 (CHCRA3-CHCRA0)

DMAC channels 3-0 have identical registers CHCRA3-CHCRA0.

Figure 8-4 shows the bits of the CHCRA register.

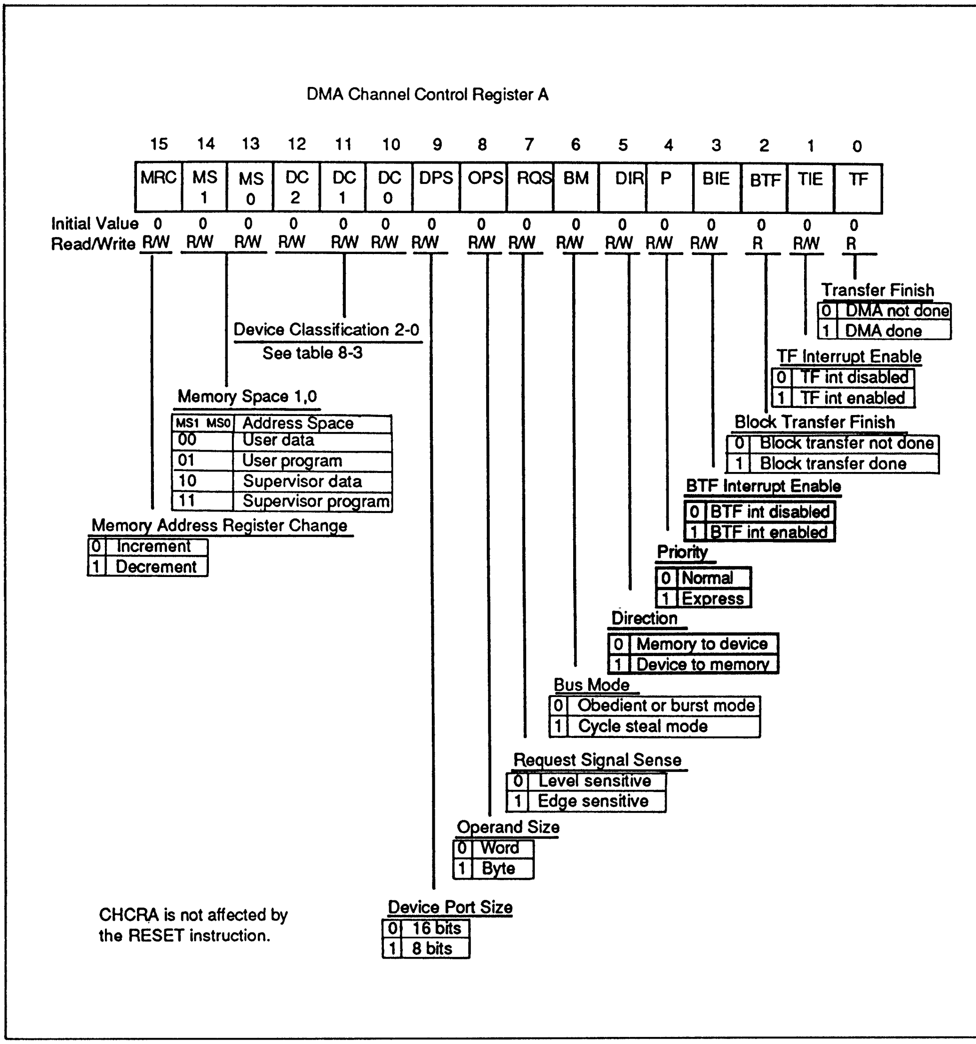


Figure 8-4. Channel Control Register A (CHCRA)

Memory Address Register Change (MRC): MRC determines whether the memory address register will be incremented or decremented. When MRC = 0, MADR is incremented. When MRC = 1, MADR is decremented. MRC is cleared by reset.

Memory Space 1, 0 (MS1, MS0): MS1 and MS0 specify the address space of the MADR register (Table 8-2). The contents of these bits are encoded onto S/$\overline{U}$ and PF pins as status during DMA cycles. MS1 and MS0 are cleared by reset.

Table 8-2. Memory Space Bits

CHCRA Bits		
MS1	**MS0**	**Address Space**
0	0	User data
0	1	User program
1	0	Supervisor data
1	1	Supervisor program

Device Classification 2-0 (DC2-DC0): DC2-DC0 define the type of device requesting DMA transfers as shown in Table 8-3. DC2-DC0 are cleared by reset.

Table 8-3. Device Classification Bits

CHCRA Bits			Address			
DC2	DC1	DC0	Mode	Operation Mode	Request	Device
0	0	0	Single	Normal	DREQ pin	ACK type
0	0	1	Single	Continue	DREQ pin	ACK type
0	1	0		(Reserved)(Note)		
0	1	1		(Reserved)(Note)		
1	0	0	Dual	Normal	DREQ pin	Memory-mapped I/O, memory
1	0	1	Dual	Normal	Internal ASCI	Memory-mapped I/O, memory
1	1	0	Dual	Normal	Internal timer	Memory-mapped I/O, memory
1	1	1	Dual	Normal	Auto	Memory-mapped I/O, memory

Note: Reserved for future expansion. If a reserved value is specified, the HD641016 may malfunction.

Device Port Size (DPS): DPS designates the data bus size of a device. DPS = 0 designates a 16-bit bus; DPS = 1 designates an 8-bit bus. DPS is cleared by reset.

Operand Size (OPS): OPS specifies DMA transfer word-size. It is valid only when the port size is 16 bits (DPS = 0) in dual address mode. OPS = 0 specifies word-size transfers; OPS = 1 specifies byte-size (8-bit) transfers. OPS is cleared by reset.

Request Signal Sense (RQS): RQS = 0 specifies level-sensitive DMA request; RQS = 1 specifies edge-sensitive DMA request. For the built-in ASCI DMA transfer, RQS must be cleared to 0. For the built-in timer DMA transfer, RQS must be set to 1. RQS is cleared by reset.

Bus Mode (BM): BM = 1 specifies cycle steal as the DMA bus mode. For external requests, BM = 0 specifies obedient mode. For auto requests, BM = 0 specifies burst mode. BM is cleared to 0 by reset.

Direction (DIR): DIR determines the direction of the DMA transfer. In dual address mode, a memory address has been specified by the MADR register and a device has been specified by the DADR register. In single address mode, a device has been specified by the acknowledge ($\overline{\text{DACK}}$) signal. DIR = 0 specifies memory to device, and DIR = 1 specifies device to memory. DIR is cleared by reset.

Priority (P): P = 1 sets the channel to express priority. P = 0 sets the channel to normal priority. P is cleared by reset. See "8.3.8 DMA Priority" for details.

BTF Interrupt Enable (BIE): BIE = 1 enables interrupt when a block transfer is finished (BTF bit set). BIE = 0 disables interrupt on BTF set. BIE is cleared by reset.

Block Transfer Finished (BTF): The BTF flag is set to 1 at the completion of a block transfer in the continue operation mode. However, it is not set if the TF flag is set. This bit is cleared to 0 by accessing the LSB (least significant byte) of the NTCR register after reading the CHCRA register or by reset.

TF Interrupt Enable (TIE): TIE = 1 enables interrupt when a DMA transfer is finished (TF bit set). TIE = 0 disables interrupt on TF set. TIE is cleared by reset.

Transfer Finished (TF): The TF flag is set to 1 when the DMAC completes a DMA transfer; that is, when ETCR reaches final H'0000 or $\overline{\text{DONE}}$ acknowledges that all transfers are complete. Reading the LSB of the ETCR register after reading the CHCRA register clears this bit automatically. TF is cleared by reset.

8.2.6 Channel Control Register B Channels 3-0 (CHCRB3-CHCRB0)

Figure 8-5 shows the bits of CHCRB.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DRC1	DRC0	DS1	DS0	DBER	—	—	—	—	—	—	—	—	—	—	—
Initial Value	0	0	0	0	0											
Read/Write	R/W	R/W	R/W	R/W	R/W											

DMA Bus Error

0	No access level violation or bus error
1	Access level violation or bus error

Device Space 1,0

DS1-DS0	Address Space
00	User data
01	User program
10	Supervisor data
11	Supervisor program

Device Address Register Change 1,0

DRC1-DRC0	Register Change
00	Increment
01	Decrement
10	Remain unchanged
11	Reserved

—: Reserved bit. Always read as 0. Cannot be written.
CHCRB is not affected by the RESET instruction

Figure 8-5. Channel Control Register B (CHCRB)

Device Address Register Change 1, 0 (DRC1, DRC0): DRC1 and DRC0 determine whether the device address register (DADR) will be incremented, decremented, or remain the same after each DMA transfer (Table 8-4) in dual address mode. In single address mode, these bits are not used. DRC1 and DRC0 are cleared by reset.

Table 8-4. Device Address Register Change Bits

CHCRB Bits		
DRC1	**DRC0**	**Register Change**
0	0	Increment
0	1	Decrement
1	0	Remain unchanged
1	1	Reserved (Note)

(Note) Reserved for future expansion. If the reserved value is specified, the HD641016 may malfunction.

Device Space 1, 0 (DS1, DS0): DS1 and DS0 specify the address space of the DADR (Table 8-5). The contents of these bits are encoded and output as status on the S/U and PF pins during the DMA device cycle.

In continue mode, the address space of the next block address must be specified in DS1 and DS0. They will be loaded automatically into the MS0 and MS1 bits of CHCRA when the current block transfer is completed. DS1 and DS0 are cleared by reset.

Table 8-5. Device Space Bits

CHCRB Bits		
DS1	**DS0**	**Address Space**
0	0	User data
0	1	User program
1	0	Supervisor data
1	1	Supervisor program

DMA Bus Error (DBER): DBER is set to 1 by an access level violation or bus error during DMA transfer cycles. It is cleared to 0 by writing 1 to it or by reset. Note that 1 is written to DBER only when it is to be cleared.

8.2.7 Operation Control Register (OPCR)

Figure 8-6 shows the bits of the OPCR register.

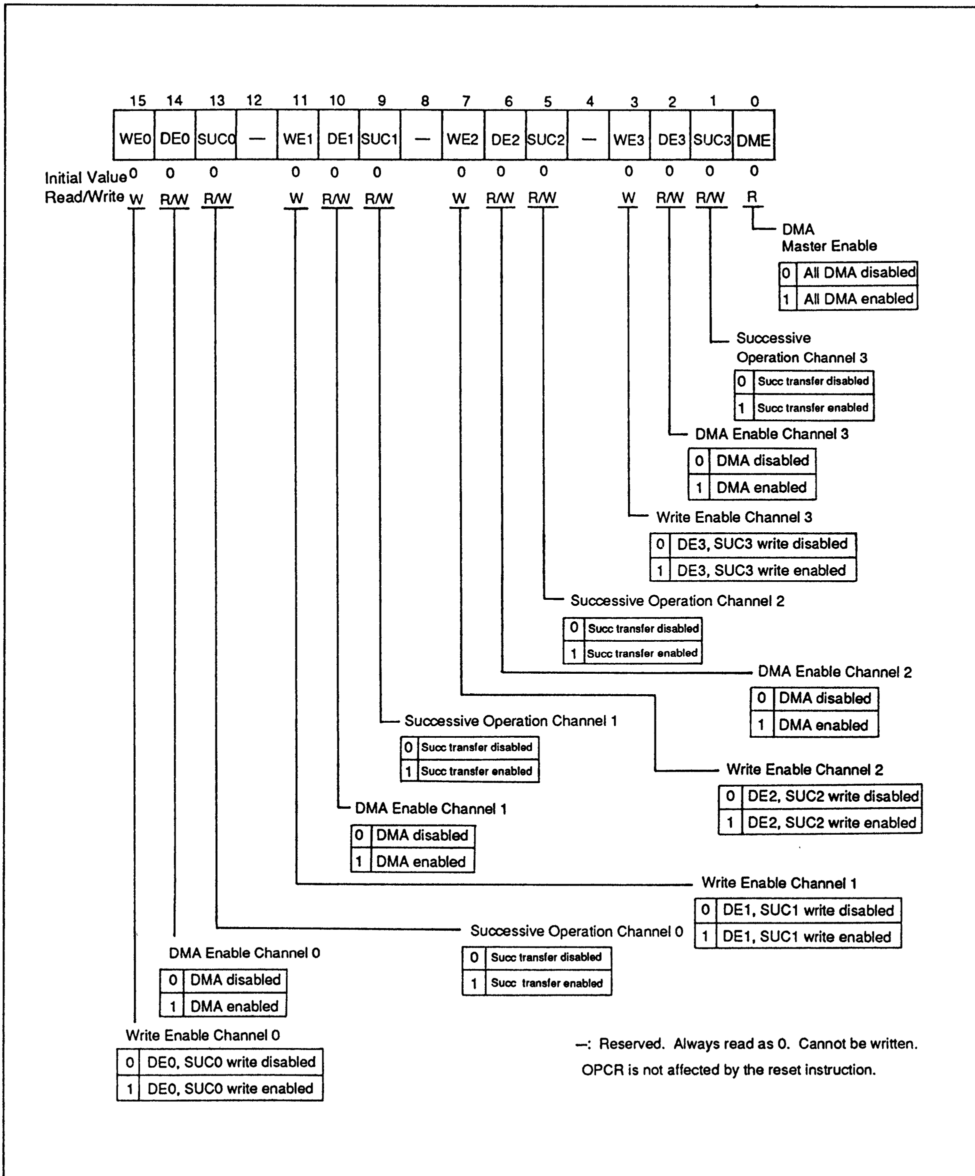


Figure 8-6. Operation Control Register (OPCR)

Write Enable Channels 3-0 (WE3-WE0): When the corresponding DE and SUC bits are updated, 1 must be written to the corresponding WE bit, simultaneously. When WE = 0, the corresponding DE and SUC bits are locked and cannot be changed. Note that this is a write-only bit and always read as 0.

DMA Enable Channels 3-0 (DE3-DE0): DE = 1 enables DMA operation on the corresponding channel. In auto-request mode, DMA operation starts when DE is set. When an external device, ASCI, or timer request occurs when DE is set, DMA transfer starts. Clearing this bit during transfer temporarily stops the transfer. DE3-DE0 are cleared by reset.

Successive Operation Channels 3-0 (SUC3-SUC0): SUC = 1 enables successive block transfers in continue mode. SUC3-SUC0 are cleared by reset. See "8.3.5 Continue Operation" for details.

DMA Master Enable (DME): DME is set with DE3-DE0 to enable DMA transfer. However, it cannot be set if the corresponding channels DBER flag is set. DME is cleared at NMI interrupt, access level violation, or bus error which stops DMA on all channels and passes control to the CPU. See "8.5 DMA Transfer Stop by NMI and Bus Error" for details. Note that read-only bit DME cannot be written. It is cleared by reset.

8.3 DMA Operation and Procedures

8.3.1 Transfer Requests

To perform DMA transfers between memory and a device, the device requests the transfer. In memory-to-memory transfers, the memory doesn't request the transfer, so it is initiated by the program. Selecting a device by setting the DC2-DC0 bits of CHCRA determines which way a transfer is requested.

Auto Request: When a transfer request is generated internally, as a memory-to-memory transfer, the CPU initializes the DMAC and starts the DMA transfer. The DMAC will perform transfers automatically until the ETCR register reaches H'0000.

In this auto request mode, transfer requests are generated automatically in the DMAC while the DE bit of OPCR is 1. Auto request mode is selected by setting the DC2-DC0 bits of CHCRA to 111.

External Request: To perform DMA transfers between memory and an external device including the built-in ASCI and timer, the CPU initializes the DMAC and the DMAC then transfers one word or byte per transfer according to a transfer request signal (external request) provided from $\overline{\text{DREQ}}$ pin or transfer request signals which are directly connected to the DMAC.

In this external request mode, transfers are allowed only when the DE bit of OPCR is 1. After ETCR reaches H'0000, the DE bit is automatically cleared, and no further transfers will be allowed (see "8.3.5 Continue Operation"). In addition, the external request is effective only when the corresponding mode is selected by the DC2-DC0 bits of CHCRA.

Moreover, the $\overline{\text{DREQ}}$ input can be specified as either edge or level sensitive by the RAS bit of CHCRA. However, $\overline{\text{DREQ}}$ must be level sensitive for the internal ASCI and edge sensitive for the internal timer DMA transfers.

In bus cycles other than DMA bus cycles, the DMAC samples the $\overline{\text{DREQ}}$ pin at the falling edge of the Ø clock, if it is selected as level sensitive. If $\overline{\text{DREQ}}$ is low, the CPU gets a bus request. When a transfer request is cancelled 2 or more clocks before the DMA operation begins, the DMA operation is cancelled. If it is cancelled 1 clock before, the DMA operation is executed. See Figure 8-7.

If there are transfer requests for more than one DMAC channel at sampling, priority will be determined depending on the priority specified in "8.3.8 DMA priority".

In the DMA bus cycle, if $\overline{\text{DREQ}}$ is low at the falling edge of T2 in the device access cycle (final wait state, if wait states are inserted), it causes a transfer request and the bus is held before executing the next transfer. In cycle steal mode, the DMAC releases the bus once after execution.

When $\overline{\text{DREQ}}$ is selected as edge sensitive, DMA transfers are initiated by the falling edge of the $\overline{\text{DREQ}}$ pin. Even if the falling edge occurs twice at the same $\overline{\text{DREQ}}$ pin before DMA execution, only one DMA transfer is performed.

The DMAC samples $\overline{\text{DREQ}}$ at the falling edge of the Ø clock in CPU bus cycle and the falling edge of T2 (final wait state, if wait states are inserted) in the device access cycle. In addition, DMA priority is determined in the same as in the level sensitive case.

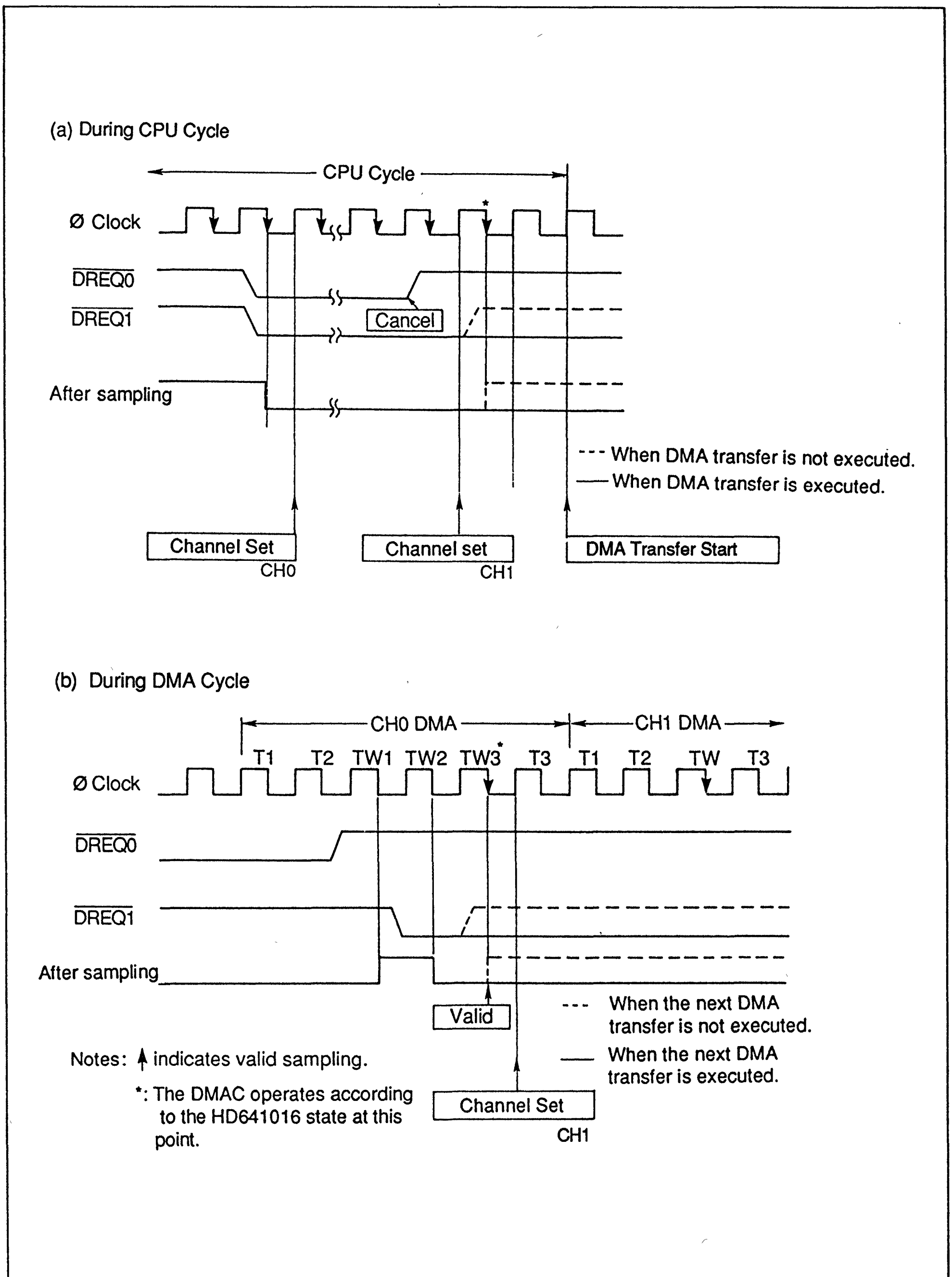


Figure 8-7. Level-Sensitive External Request Sample Timing

8.3.2 Applicable Devices

The DMAC can perform DMA transfers between external memory and external I/O with $\overline{\text{DACK}}$ signal in single address mode, and between external memory, external memory-mapped I/O, internal peripheral I/O registers (except internal DMAC registers), and internal RAM in dual address mode. However, DMA transfer between DMACs is disabled. Note that the HD641016 cannot support DMA transfers synchronous with E clock. Table 8-6 shows the types of external I/O devices the DMAC supports.

Table 8-6. External I/O Devices

Device Type	Address Mode	Port Size	Operand Size	Request Mode
I/O device with $\overline{\text{DACK}}$	Single address	8 bits	Byte	External
		16 bits	Word	External
Memory-mapped I/O	Dual address	8 bits	Byte	External, auto
		16 bits	Byte, word	External, auto

8.3.3 Data Transfer

The DMAC provides two transfer modes: single address and dual address. In single address mode, source and destination are accessed at the same time, and DMA transfers are done in a single bus cycle. Accordingly, high-speed transfers are performed in single address mode. In dual address mode, source and destination are accessed in separate bus cycles. Transfer data is temporarily saved in a DMAC.

Single Address Mode: In the single address mode, I/O devices can be selected by the $\overline{\text{DACK}}$ signal. Data transfers between memory and the external I/O device are controlled by the "handshake" of $\overline{\text{DREQ}}$ transfer request and $\overline{\text{DACK}}$ acknowledge signals.

To initialize the DMAC, set the transfer mode the CHCRA and CHCRB registers, the source or destination address in the MADR register, and number of bytes/words to be transferred in the ETCR register. Then, setting the DE bit of OPCR enables DMAC operation, and DMA transfer will start when the DMAC gains control of the bus by $\overline{\text{DREQ}}$. The ETCR register is decremented after every transfer and the MADR register is incremented or decremented to point to the next transfer address.

If the DE bit of the OPCR register is cleared before the transfer is completed, the transfer will be suspended.

Single address transfers are always performed in the external request mode. They can be performed in obedient or cycle steal bus mode. Figure 8-8 is a single address mode operation flowchart.

Figure 8-9 shows DMA transfer timing in single address mode.

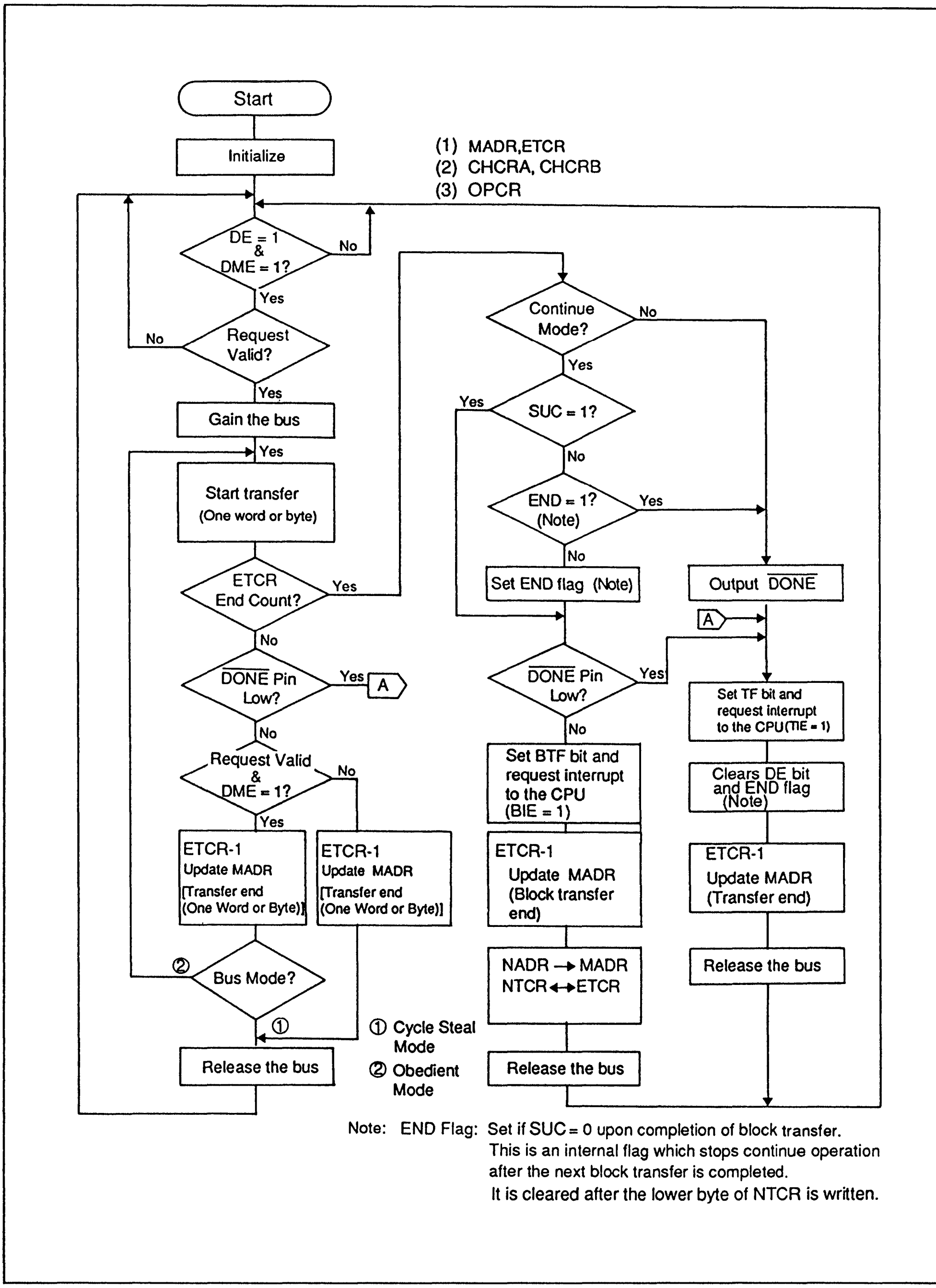
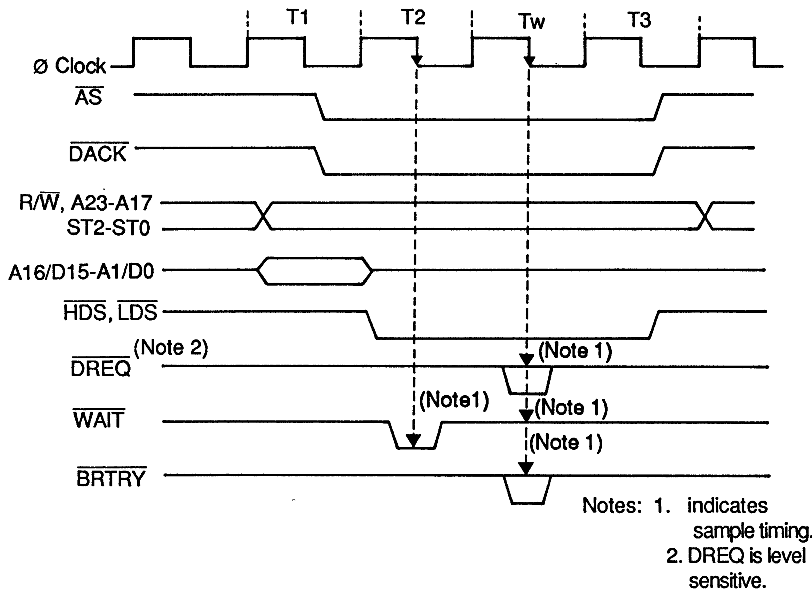
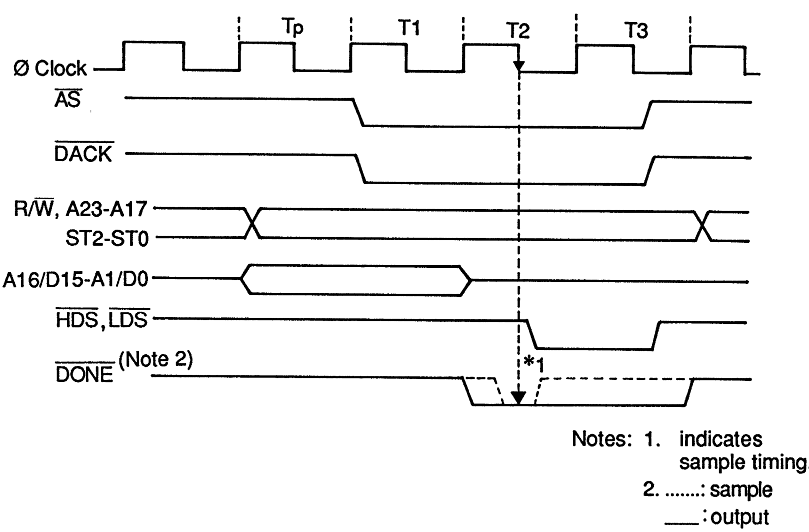


Figure 8-8. Single Address Mode DMA Operation



(i) Single Address Mode with Tw State (Memory Read)



(ii) Single Address Mode with Tp State (Memory Write)

Figure 8-9. DMA Transfer Timing in Single Address Mode

Dual Address Mode: When both the source and destination have an address, such as memory-to-memory, or memory-to/from-memory-mapped I/O, the DMAC divides the DMA cycle into two parts, a read cycle and a write cycle, to access both addresses. In dual address mode, the data is held temporarily in the DMAC. Since the DMAC regards the read and write cycle as successive cycles, it does not release the bus between the cycles. However, if an access level error or bus error occurs during a read cycle, the DMAC does not perform a write cycle.

External request and auto request modes are available in dual address mode. For memory-to-memory transfers, auto request mode is selected by the CHCRA register and the memory does not provide the $\overline{\text{DREQ}}$ signal. Instead, the number of bytes/words set by the program are transferred automatically.

To initialize the DMAC, set the transfer mode in CHCRA and CHCRB, the source and destination addresses in MADR (for memory address) and DADR (for device address or memory address), and number of bytes/words to be transferred in ETCR. Then setting the DE bit of OPCR enables DMA operation. DMA transfer then begins when the DMAC gains control of the bus by setting the DE bit of OPCR in auto request mode. In external request mode, DMA transfer begins by the DMAC gaining bus control by an external request. ETCR is decremented after every transfer, and DADR and MADR are incremented or decremented to point to the next transfer address.

If the DE bit is cleared before the transfer is completed, the transfer will be suspended.

In addition, burst, cycle steal, and obedient modes are available in dual address mode.

Figures 8-10 and 8-11 show the dual address DMA transfer operation for memory-to-I/O device transfer and for I/O device-to-memory transfer. As can be seen from the figures, operation depends on transfer direction.

Figure 8-12 shows DMA transfer timing in dual address mode.

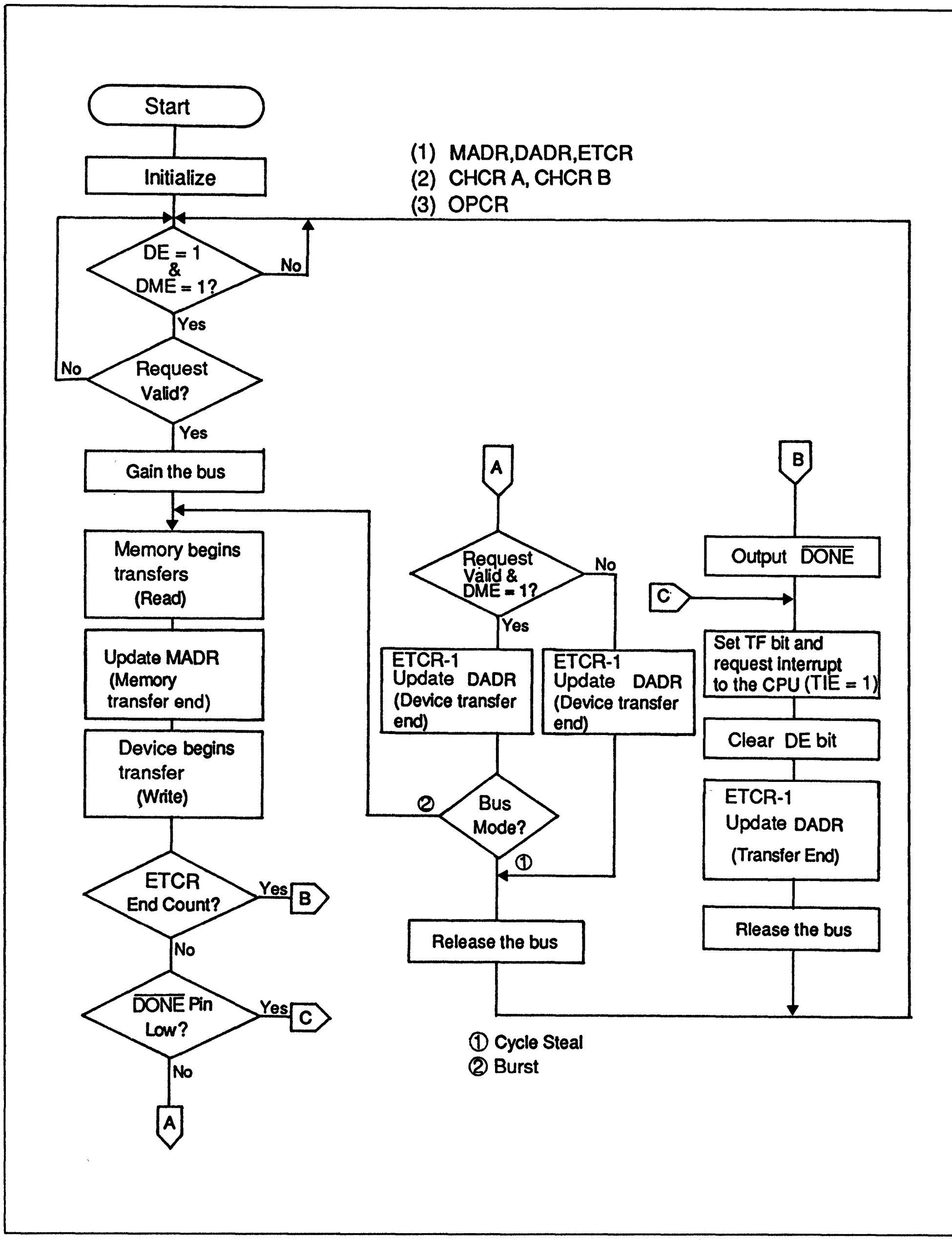


Figure 8-10. Dual Address Mode DMA Operation Flow (Memory ---> I/O Device Transfers)

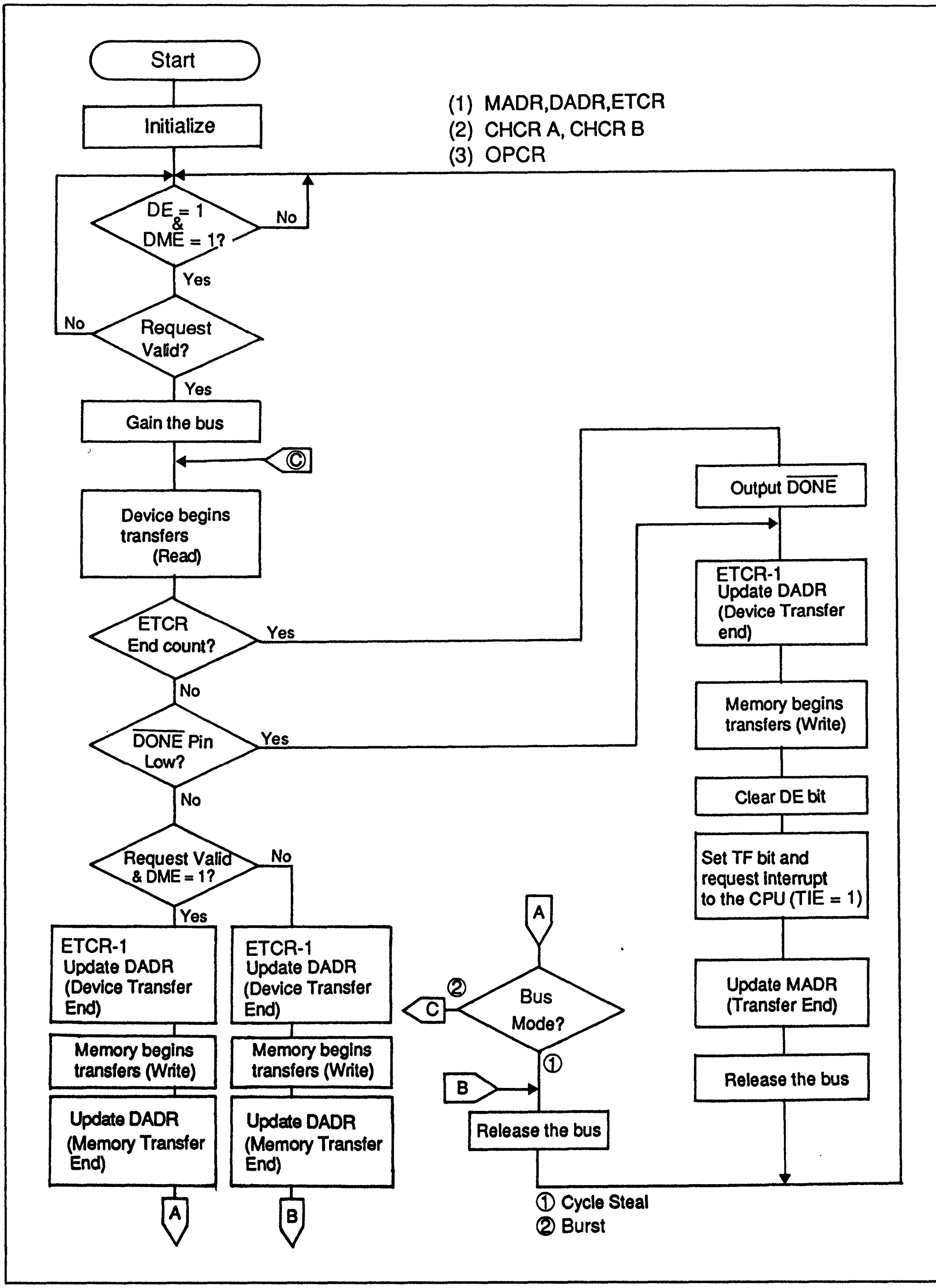
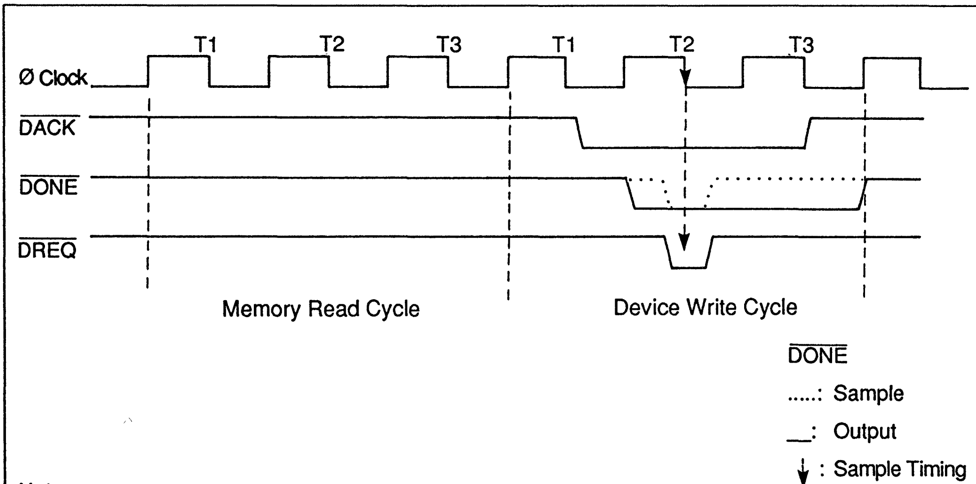


Figure 8-11. Dual Address Mode DMA Operation Flow (I/O Device ---> Memory Transfers)



Notes:

1. $\overline{\text{AS}}$, $\overline{\text{HDS}}$, $\overline{\text{LDS}}$, A16/D15-A1/D0, A23-A17, ST2-ST0 are output, and $\overline{\text{WAIT}}$ and $\overline{\text{BRTRY}}$ are sampled with the same timing as in the CPU bus cycle.
   However, in memory to internal I/O device DMA cycles, $\overline{\text{DACK}}$ is output and $\overline{\text{DONE}}$ is not masked; while $\overline{\text{AS}}$, $\overline{\text{HDS}}$ and $\overline{\text{LDS}}$ are not output, and $\overline{\text{WAIT}}$ and $\overline{\text{BRTRY}}$ are masked.
2. When $\overline{\text{DREQ}}$ is level sensitive

Figure 8-12. Dual Address Mode DMA Operation Timing (Memory to I/O Device)

8.3.4 Bus Mode

The DMAC supports obedient, burst, and cycle steal bus modes. The following explains the DMAC operation in a channel.

Obedient Mode: In obedient mode, DMA transfers are executed when the external device acquires the bus mastership by external $\overline{\text{DREQ}}$ request.

Accordingly, if an I/O device, which can output $\overline{\text{DREQ}}$ at the same rate as the machine cycle or which can fix $\overline{\text{DREQ}}$ at low level, performs DMA transfer in this obedient mode, the I/O device performs DMA transfer in the same way as in burst mode described below.

Burst Mode: In burst mode, bus control is acquired when DE of OPCR = 1, and DMA transfers will be executed until a transfer end condition is satisfied. The transfer end conditions are; ETCR reaches end count (H'0000), $\overline{\text{DONE}}$ input asserted externally, or the DME bit of OPCR is cleared. See "8.5 DMA Transfer Stop by NMI and Bus Error" for details.

Cycle Steal Mode: Cycle steal mode can be selected in external or auto request modes. The DMAC always returns the bus to a bus master other than DMAC after one word or byte DMA transfer.

Figure 8-13 shows obedient and cycle steal timing. $\overline{\text{DREQ}}$ is level sensitive, and sampled at the arrow. If $\overline{\text{DREQ}}$ is low when sampled, the bus is requested for a DMA transfer. In cycle steal mode, however, this request won't occur for one state after a transfer. A bus master other than DMAC can get the bus during this one state. If other bus master does not gain the bus control, DMA transfer begins again after one state.

If multiple DMAC channels execute DMA transfer with rotating priority in cycle steal mode, the BM bit of CHCRA of the currently executing DMA transfer is valid. For example, if DMAC channel 0 is specified in cycle steal mode and DMAC channel 1 is specified in obedient mode, DMAC channel 0 first executes DMA transfer and releases the bus for one state one byte/word DMA transfer. DMAC channel 1 then gains bus control and begins DMA transfer if $\overline{\text{DREQ}}$ is asserted.

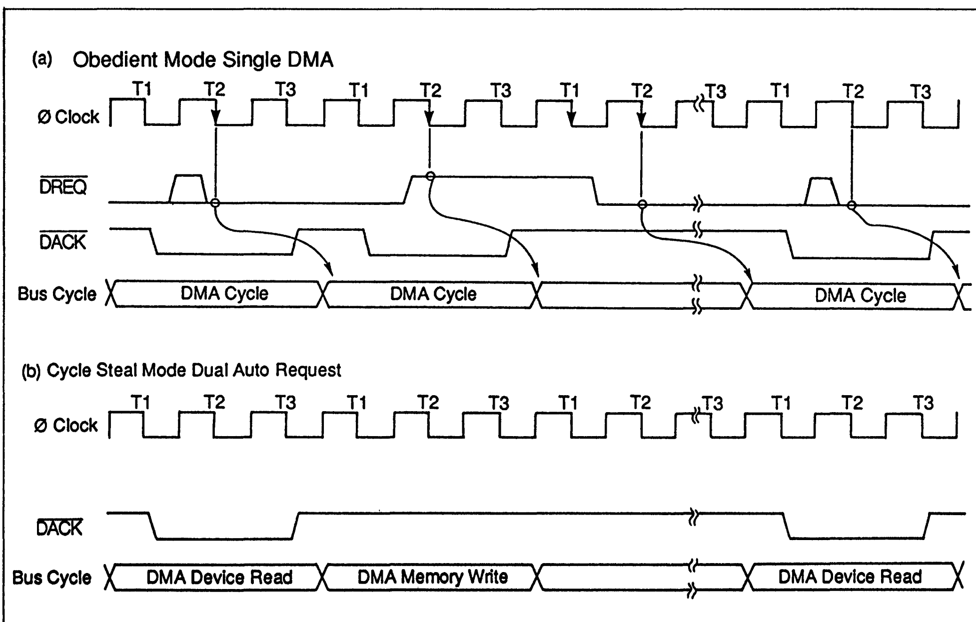


Figure 8-13. Obedient and Cycle Steal Mode Timing

8.3.5 Continue Operation

The DMAC can perform continue operation in single address mode by using the NADR, NTCR, and CHCRB registers.

The continue operation can be selected by specifying the DC2-DC0 bits of CHCRA as single address mode with continue. In continue operation, after the DMAC completes the block transfer specified by MADR and ETCR, the contents of the NADR are automatically transferred to the MADR. The contents of ETCR and of NTCR are then exchanged. The address space of the next block transfer specified by the DS1-DS0 bits of CHCRB is automatically transferred to the MS1 and MS0 bits of CHCRA. Then, the DMAC begins the next block transfer.

In normal continue operation, the SUC bit of OPCR is cleared to 0. Updating the lower byte of NTCR causes the block transfer to be continued. If the lower byte of NTCR is not updated, the block transfer stops after the current block has been transferred. Note that the number of bytes/words to be transferred must be specified in NTCR before beginning the block transfer since the NTCR is reloaded at least once. In addition, NADR and the DS1-DS0 bits must be updated before NTCR is updated.

If all blocks are 64 kbytes/words, the continue operation can be performed by clearing ETCR and NTCR before setting the SUC bit. At this time, block transfers can continue without the software updating NTCR. The SUC bit must be cleared one block before the block transfer is completed. Accordingly, clear the SUC bit only when stopping block transfer at the end of the next block transfer. Moreover, NTCR must not be written to in this case. See Figure 8-8.

Note that the number of bytes/words of the next block must be specified in NTCR before setting continue mode by the DC2-DC0 bits to perform continue operation correctly.

8.3.6 End Operation

The conditions that end each channel transfer are: ETCR reaches H'0000, the $\overline{\text{DONE}}$ signal is asserted, or program clears the DE bit of OPCR. In addition, all channels' DMA operations can be stopped by the DME bit. See "8.5 DMA Transfer Stop by NMI and Bus Error" for details.

ETCR Reaches H'0000: ETCR is decremented after one byte/word transfer. When ETCR becomes 1, the DMAC performs the last transfer.

The DMAC outputs $\overline{\text{DONE}}$, unless it is performing a continue operation. This sets the TF bit in CHCRA to release the bus for a bus master other than the DMAC and finishes the transfer (ETCR = 0 after transfer). At this time, if the TIE bit of CHCRA is set, the DMAC interrupts the CPU.

In a continue operation, when each block transfer has completed, the BTF bit of CHCRA is set, but $\overline{\text{DONE}}$ is not output. When all blocks are transferred, the TF bit is set, the BTF bit is not set, but $\overline{\text{DONE}}$ is output. Accordingly, if the BIE TIE bit is set while the BTF or TF bit is set, it will cause an BTF or TF interrupt, respectively. DREQ after the last block transfer is ignored.

After the last transfer cycle, the DE bit of OPCR is cleared. Each address register points to an address following the final transfer address (updated in the same way as during transfer). ETCR is H'0000.

Figure 8-14 shows the $\overline{\text{DONE}}$ output timing. $\overline{\text{DONE}}$ is an open-drain I/O pin and is output while a device is accessed.

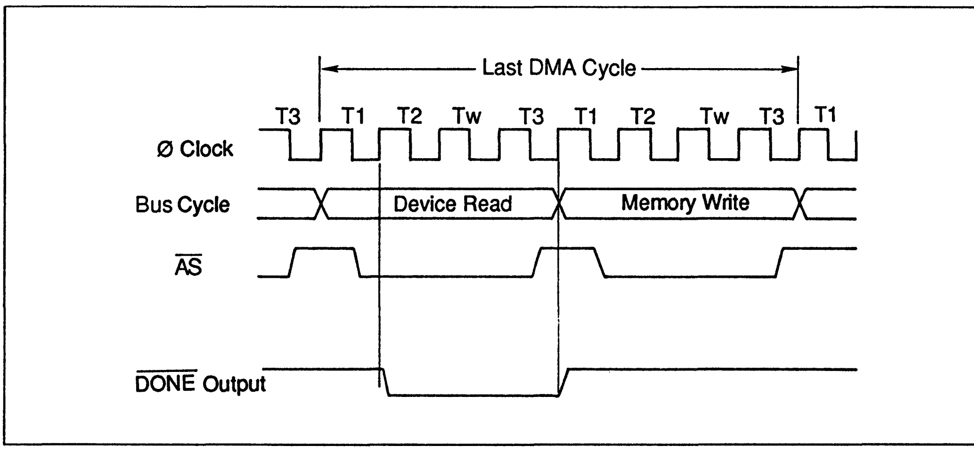


Figure 8-14. $\overline{\text{DONE}}$ Output Timing

DONE Assertion: If $\overline{\text{DONE}}$ is asserted low at the falling edge of T2 or Tw state of the last DMA cycle before ETCR reaches H'0000, the DMAC will complete the DMA transfer of the current channel. The TF bit of CHCRA and each register status other than ETCR are the same as when ETCR reaches H'0000. ETCR is decremented by the number of bytes/words to be transferred. Moreover, if $\overline{\text{DONE}}$ is asserted low at the above timing, the next block transfer address are not reloaded.

The DMAC samples $\overline{\text{DONE}}$ externally at the falling edge of T2 or wait state of device access cycle as shown in Figure 8-15.

DREQ is ignored after $\overline{\text{DONE}}$ has been asserted.

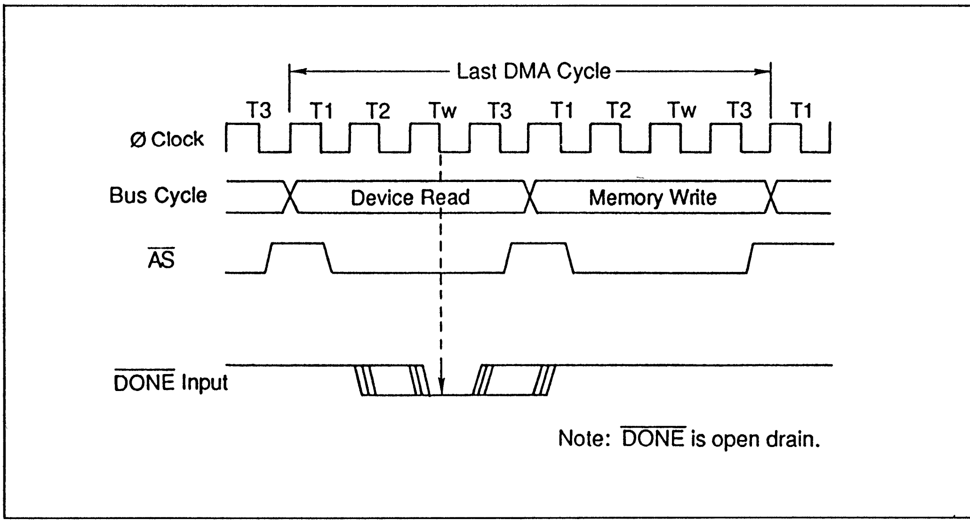


Figure 8-15. DONE Input Timing

DE Bit Cleared: If the program clears the DE bit of OPCR, DMA transfer is suspended. DMA transfer starts from the next address if DE is set again. However, the DMA transfer requested by the $\overline{\text{DREQ}}$ falling edge is cancelled.

8.3.7 Address Update and Transfer Map

The DMAC address update depends on the address modifier (increment, decrement, no change), device port size, operand size, and data size (byte or word) as shown in Table 8-7. The following paragraphs describe the data destination locations for these different cases. Note that the LSB of MADR and DADR are not used for word transfer.

Table 8-7. Address Update and Transfer Length

Address Mode	Device Port Transfer Size (DPS)	Operand Size (OPS)	Memory Address Modifier	Device Address Modifier	Data Size
Single address	8 Bits	Byte	±1	-	Byte
	16 Bits	Word	±2	-	Word
Dual address	8 Bits	Byte	±1	±2, fixed	Byte
	16 Bits	Byte	±1	±1, fixed	Byte
	16 Bits	Word	±2	±2, fixed	Word

Single Address Mode, DPS = 8 Bits: In single address mode, when DPS specifies 8 bits, operand size should match DPS, since the destination must receive the upper or lower data on the data bus provided from the source in a one bus cycle. Although DPS is 8 bits, the data bus is 16 bits wide. Data can be transferred over the upper or lower byte of the data bus as shown in Figure 8-16. Accordingly, the bus switch is required for reading/writing upper and lower byte data to/from an I/O device using the $\overline{HDS}$ and $\overline{LDS}$ signals.

Transfers can be started from even or odd addresses when DPS = 8 bits. See Figure 8-16.

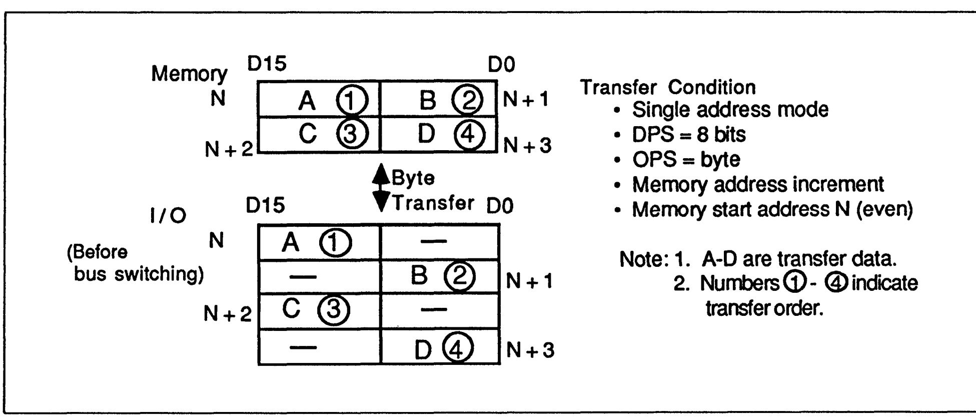


Figure 8-16. Single Address, DPS = 8 Bits Transfer Map

Single Address Mode, DPS = 16 Bits: When DPS = 16 bits, it is not necessary to switch the bus as when DPS = 8 bits, because the data bus for the memory and I/O devices are identical. However, transfer must start on an even address. If the start address is odd, the LSB will be assumed to be 0. Figure 8-17 is an example of transfer map.

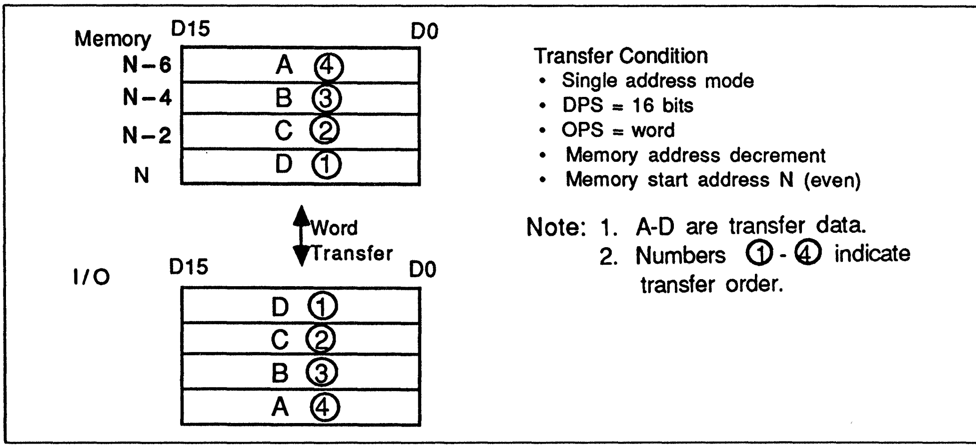


Figure 8-17. Single Address, DPS = 16 Bits Transfer Map

Dual Address Mode, DPS = 8 Bits, OPS = Byte: In dual address mode with DPS = 8 bits, the sizes of the memory and I/O device data bus are different. However, the DMAC uses the upper or lower byte of the data bus (D15-D8 or D7-D0) for reading and writing by switching bus.

The device address can be specified as fixed, or incremented or decremented by 2. Data transfers between DMAC and I/O are performed through the upper or lower data bus, depending on whether the I/O start address is specified as even or odd (Figure 8-18).

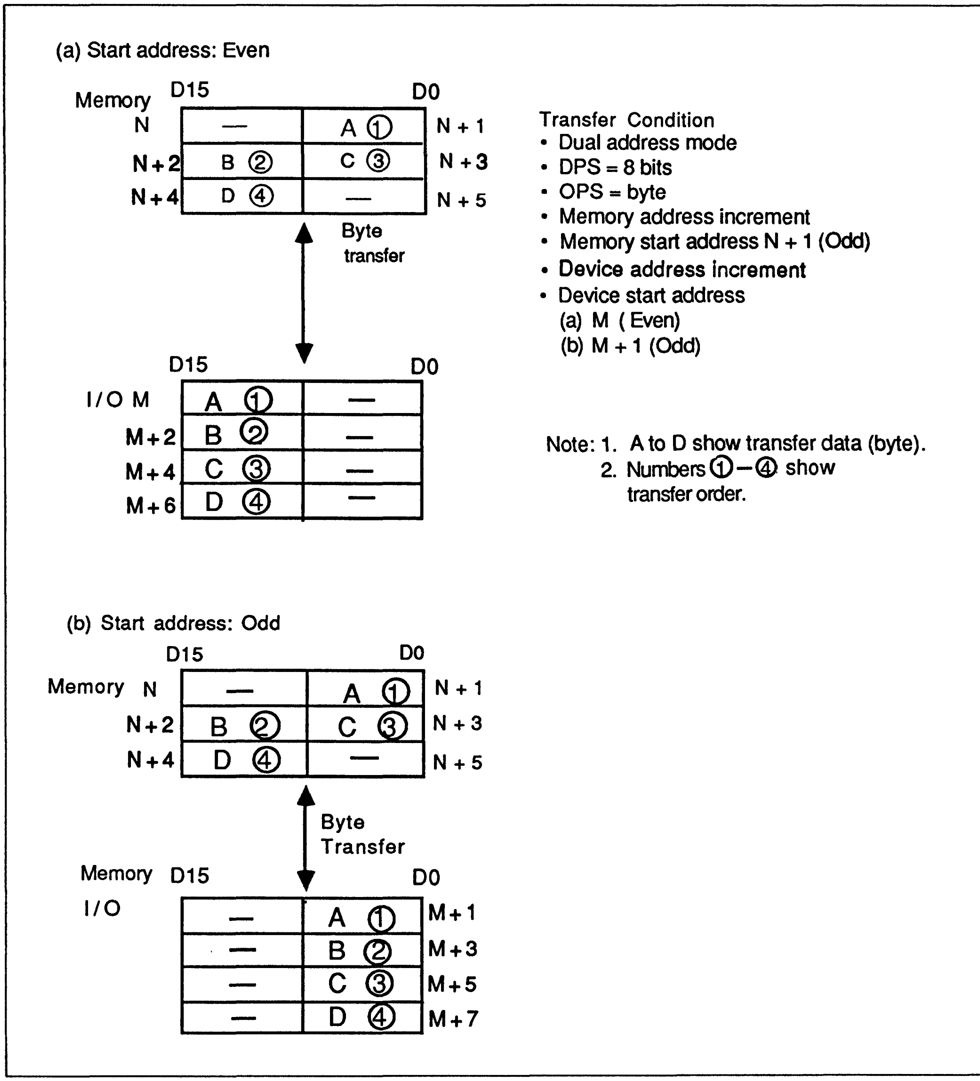


Figure 8-18. Dual Address, DPS = 8 Bits Transfer Map

Dual Address Mode, DPS = 16 Bits, OPS = Byte: In dual address mode with DPS = 16 bits and OPS = byte, transfer start address can be even or odd. The DMAC accesses upper and lower bytes alternately even if the device address is fixed. See Figure 8-19.

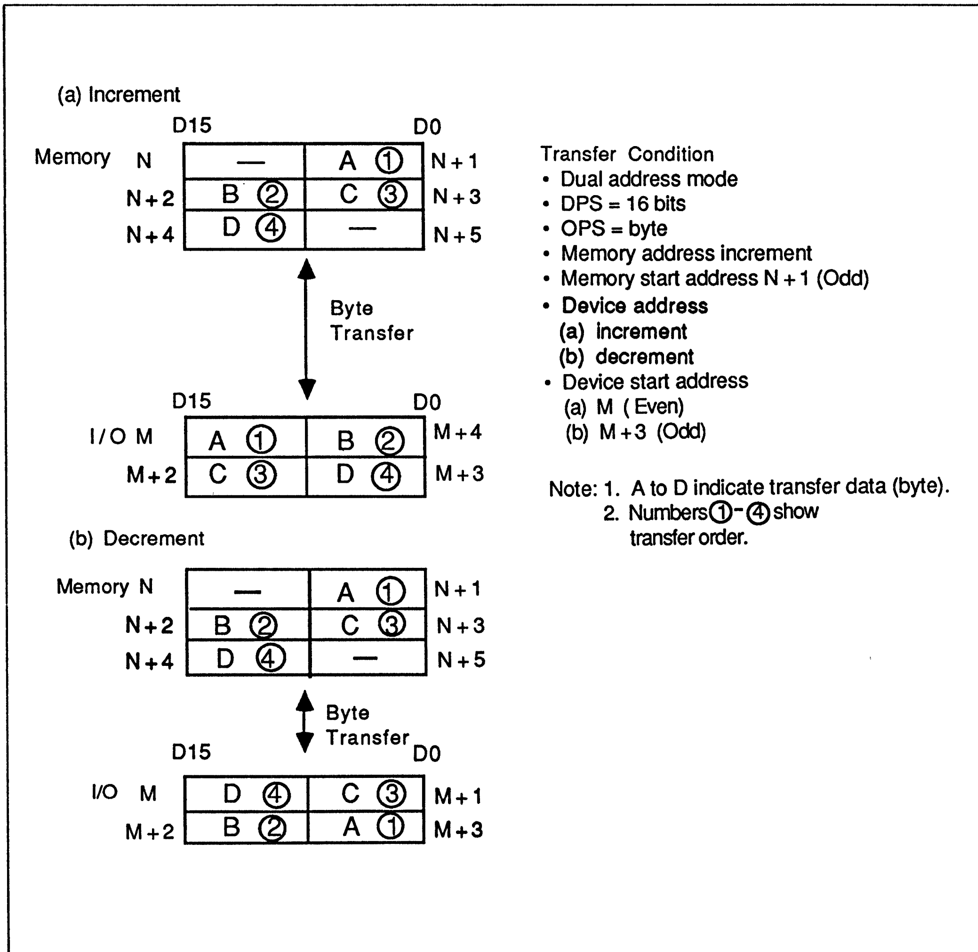


Figure 8-19. Dual Address, DPS = 16 Bits, OPS = Byte Transfer Map

Dual Address Mode, DPS = 16 Bits, OPS = Word: In dual address mode with DPS = 16 bits and OPS = word, memory and I/O device start address should be even. If an odd address is selected, the LSB will be assumed to be 0. See Figure 8-20.

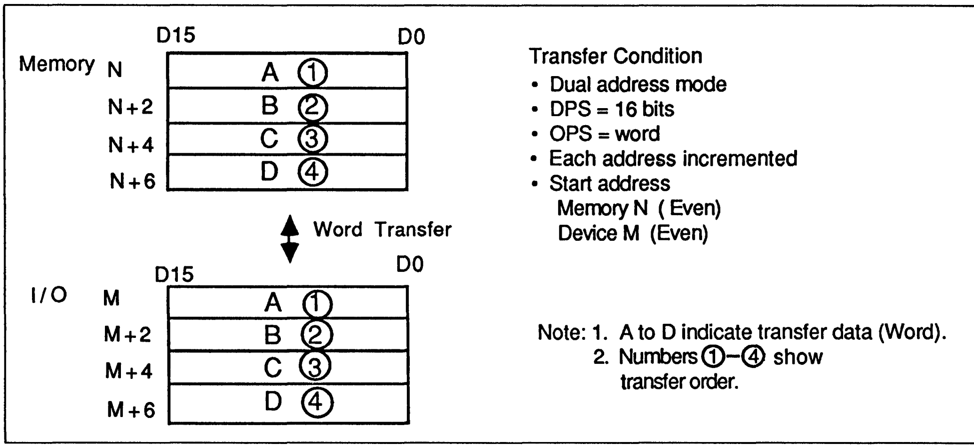


Figure 8-20. Dual Address, DPS = 16 Bits, OPS = Word Transfer Map

8.3.8 DMA Priority

Although, the internal DMA controller's channel can handle either a normal channel or express channel, the express channel has priority over the normal channel. If an express channel requests DMA transfer during a normal channel's DMA transfer, the normal channel is interrupted and the express channel, in turn, begins DMA transfer. Once an express channel gains bus control, it continues DMA transfer until it negates its requests.

Prioritization among channels in the same mode is described below.

Priority Among Normal Channels: When two or more normal channels request DMA transfer at one time, the DMA controller performs one word transfer at each channel by rotating bus control in low-to-high channel number order. If transfer requests are no longer made or if they are cancelled, channel 0 takes a priority. When a normal channel is interrupted by an express channel, servicing jumps to the next normal channel after the express channel's transfer has been completed. (Once transfer requests are cancelled, priority again returns to channel 0.)

Priority Among Express Channels: When more than one express channel request DMA transfer simultaneously, the lowest-numbered channel is given priority. However, an express channel whose request is made first continues to be serviced even before lower-number channels.

When a DMA request is made in a non-DMA bus cycle, the internal DMA controller determines the priority channel and then waits for bus availability. However, if the transfer request is cancelled, the priority channel is re-determined as follows:

When an additional request is added from among normal channels, the priority channel is not re-determined. However, if the transfer request serviced is cancelled, the normal channel with the next lower number gains priority after which priority is passed consecutively to lower numbered channels.

When an express channel which has priority cancels its request, priority is assigned in the following order:
1. Other express channels (in descending channel number order)
2. Normal channel having the next lower number from that of the cancelled express channel.
3. Descending normal channel number

When a normal channel is interrupted by an express channel, the priority channel is re-determined according to express channel prioritization.

8.4 Internal DMA

8.4.1 Internal ASCI and DMAC

The DMAC is connected internally to the ASCI. The ASCI is selected by setting DC2-DC0 of the CHCRA to 101. The DMAC channel 0 is connected to the ASCI channel 0 receiver, DMAC channel 1 is connected to the ASCI channel 0 transmitter, DMAC channel 2 is connected to the ASCI channel 1 receiver, and DMAC channel 3 is connected to the ASCI channel 1 transmitter.

In the internal ASCI DMA transfers, dual address mode and level-sensitive request must be selected. If the ASCI register addresses are set to the registers, the ASCI DMA transfer is executed according to the request.

The internal ASCI DMA operation is the same as that of an external devices. $\overline{\text{DACK}}$ and $\overline{\text{DONE}}$ are also valid.

8.4.2 Internal Timer and DMAC

The internal timer 1 is connected to DMAC channel 1, timer 2 is connected to DMAC channel 2. DMAC channels 0 and 3 are not connected to the timer.

The DMAC receives the timer DMA request when DC2-DC0 is set to 110. Dual address mode and edge-sensitive request must be used for this operation. If the timer register addresses are set at the DADR registers, the timer DMA transfer can be executed according to the request.

If an I/O address other than a timer is specified in the device address register, the data is transferred to or from I/O under control of the timer DMA request.

Internal timer DMA operation is the same as for external devices.

8.5 DMA Transfer Stop by NMI and Bus Error

The DME bit of OPCR is cleared to 0 to stop DMA transfer when an NMI, access level violation, or bus error occurs during DMA transfers.

If the DME bit is cleared, the DMAC transfer does not begin regardless of the DE bit. If the DME bit is cleared during DMA transfer, the DMAC stops DMA transfers after completing the remaining DMA bus cycles. To start DMA transfer after an interrupt processing has completed, the DE bit of OPCR must be set to 1 again. However, after an access level violation bus error, the DBER bit must be cleared before setting the DE bit.

Note that the HD641016 does not retry the DMA transfer cycle if an error occurs during the DMA transfer regardless of the BRTE bit (of BMR) state. The HD641016 stops the DMA transfer as a bus error and set the DBER bit of CHCRB.

In addition, if a bus error occurs during dual address mode DMA transfer, the write cycle is not executed. Moreover, DADR and ETCR are updated during device access cycle, and MADR is updated during memory access cycle.

Note that if an access level violation or bus error occurs in the last DMA cycle (ETCR = H'0000 or DONE is input externally), the BTF and TF bits are not set. However, a bus error occurs simultaneously with the completion of a block transfer, the next address is loaded.

8.6 DMAC and Reset

The DMAC operates as follows after power-on reset or manual reset:

1. The CHCRA, CHCRB, and OPCR registers are reset to H'0000.

2. The MADR, DADR/NADR, ETCR, and NTCR registers are undefined.

3. At power-on reset or manual reset, ongoing DMA transfers stop and the bus is released, even if the bus cycle is not complete. At this time, control signals are in the same status as in DMA transfer halt state. If reset and DMA end operation occur simultaneously, $\overline{\mathrm{DONE}}$ is not output and no interrupt is requested.

8.7 DMAC Notes

1. The DE bit of OPCR must be cleared before CHCRA is written.

2. The DMAC does not allow DMA transfer to its own registers. Writing to them is disabled. If read, they are invalid.

3. If the SUC bit is set to 1 in continue operation mode, the CPU may not be able to control bus.

4. During single address mode DMA transfers, the I/O device should interpret $\mathrm{R}/\overline{\mathrm{W}}$ polarity as reversed: $\mathrm{R}/\overline{\mathrm{W}}$ high indicates a write to the I/O device, $\mathrm{R}/\overline{\mathrm{W}}$ low indicates a read from the I/O device.

5. To enter system stop mode, write H'8888 to OPCR twice before executing the SLEEP instruction.

6. The DBER flag must be set only when it is to be cleared.

7. If an access level violation or bus error exception occurs in the next block address after a block transfer has completed, the bus error exception processing may precede the block transfer end exception processing.

8. If the DE bit is cleared, DMA transfer requested by the $\overline{\mathrm{DREQ}}$ falling edge is cancelled.

9. When DMA transfer is performed in cycle steal mode, a bus master other than DMAC can gain the bus under the control of the arbitrator. Accordingly, the normal bus cycle changes as follows:

   DMA bus cycle → CPU bus cycle → DMA bus cycle

   However, if a refresh is requested during the normal bus cycle, the bus cycle changes as follows:

   DMA bus cycle → Refresh bus cycle → DMA bus cycle

Section 9. Timers

The HD641016 has two multifunction timers.

9.1 Features

9.1.1 Multifunction Timers

- 16-Bit upcounter
- Two 16-bit count compare registers to generate rectangular waveforms of any duty cycle
- PWM output
- Two-phase stepper motor drive output
- One-shot pulse output
- Event count
- Pulse width or frequency measurement
- Hardware triggered
- Software triggered
- Internal DMAC interface
- Timer 1 cascade operation
- Count match, overflow, and measurement end interrupts

9.1.2 Block Diagram

Figure 9-1 is the block diagrams for timer 1 and timer 2. Timers 1 and 2 have the same configuration.

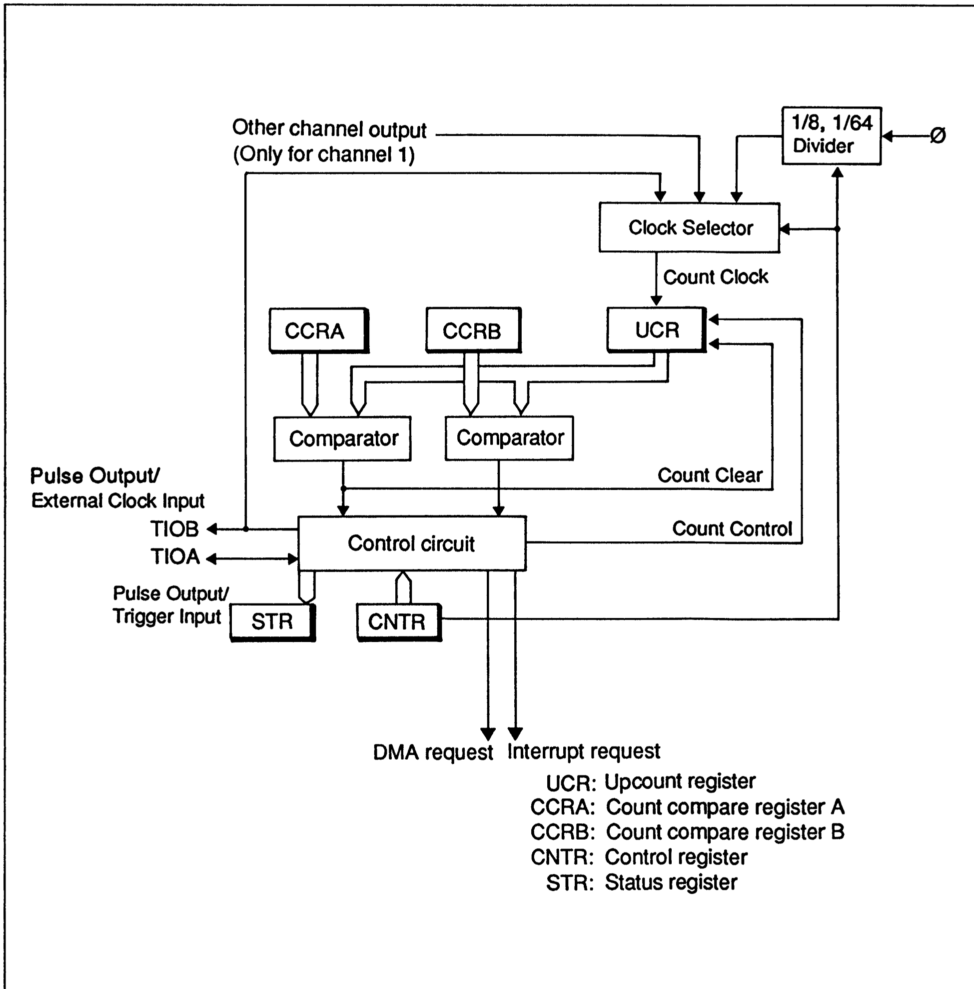


Figure 9-1. Timer Block Diagram

9.2 Timer Registers

Table 9-1 lists the timer registers.

Table 9-1. Timer Registers

Timer	Register	Symbol	Address Offset	R/W	Initial Value	Size
Timer 1	Upcount register ch 1	UCR	H'FF8E	R/W	H'0000	W
	Count compare register A ch 1	CCRA	H'FF90	R/W	H'FFFF	W
	Count compare register B ch 1	CCRB	H'FF92	R/W	H'FFFF	W
	Control register ch 1	CNTR	H'FF94	R/W	H'0000	W
	Status register ch 1	STR	H'FF96	R	H'0000	W
Timer 2	Upcount register ch 2	UCR	H'FF98	R/W	H'0000	W
	Count compare register A ch 2	CCRA	H'FF9A	R/W	H'FFFF	W
	Count compare register B ch 2	CCRB	H'FF9C	R/W	H'FFFF	W
	Control register ch 2	CNTR	H'FF9E	R/W	H'0000	W
	Status register ch 2	STR	H'FFA0	R	H'0000	W

9.2.1 Upcount Registers (UCR)

Timer 1 and timer 2 have identical 16-bit read/write UCR registers. The contents of UCR are incremented every count input clock. It can be read or written to by the program independently of counting. It is initialized to H'0000 at reset.

9.2.2 Count Compare Registers A, B (CCRA, CCRB)

Timer 1 and timer 2 have identical 16-bit read/write CCRA and CCRB registers. Their contents are compared with the corresponding UCR. CCRA and CCRB are initialized to H'FFFF at reset.

9.2.3 Control Registers (CNTR)

Figure 9-3 shows the 16-bit read/write control register (CNTR). Timer 1 and timer 2 have identical 16-bit read/write CNTR registers. It controls timer operation. It is initialized to H'0000 at reset.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MDS1	MDS0	CSS2	CSS1	CSS0	TSS1	TSS0	DMA	CIE	OIE	MIE	OLB	OLA	UCE	UCRC	STP
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Stop Point

0	TIOA precedes
1	TIOB precedes

Upcount Register Clear

Upcount Enable

UCE	UCRC	Timer Operation
0	0	Count Stop
1	0	Count Run
*	1	Count Initialization

*: Don't care

Output Level B.A

OLB	OLA	Timer output
0	0	Low
1	1	High

Measurement End Interrupt Enable

0	Disable
1	Enable

Overflow Interrupt Enable

0	Disable
1	Enable

Compare Interrupt Enable

0	Disable
1	Enable

Direct Memory Access Request

0	Disable
1	Enable

Trigger Source Select 1, 0

TSS1	TSS0	Trigger input
0	0	Reserved
0	1	Reserved
1	0	TTGI
1	1	TOUT

Clock Source Select 2-0

CSS2	CSS1	CSS0	Input clock
0	0	0	1/8ø
0	0	1	Reserved
0	1	0	Reserved
0	1	1	Reserved
1	0	0	1/64 ø
1	0	1	Reserved
1	1	0	TCKI
1	1	1	TOUT

Mode Select 1, 0

MDS1	MDS0	Mode
0	0	Measurement
0	1	One shot
1	0	Normal
1	1	Two-phase output

CNTR is not affected by the RESET instruction.

Figure 9-2. Control Register (CNTR)

Mode Select 1, 0 (MDS1,MDS0): MDS1 and MDS0 select the timer operation mode. The operation mode determines the functions of TIOA and TIOB, the clock source, trigger input, and count operation mode for the corresponding timer. Table 9-2 shows how MDS1 and MDS0 determine timer's operation mode.

Table 9-2. Mode Select 1-0 and Timer Operation

	MDS1, MDS0			
	00	**01**	**10**	**11**
Mode	Measurement	One-shot	Normal	Two-phase
TIOB Function	External clock input	Timer output	External clock input	Timer output
TIOA Function	External trigger input	External trigger input	Timer output	Timer output
Clock Input	1/8 Ø, 1/64 Ø, TCKI, TOUT	1/8 Ø, 1/64 Ø, TOUT	1/8 Ø, 1/64 Ø, TCKI, TOUT	1/8 Ø, 1/64 Ø, TOUT
Count Operation Mode	Free-run	Auto-clear	Auto-clear	Auto-clear
DMA Request	Enable	Enable	Enable	Enable (Note 2)
Output Level	Enable	Enable	Enable	Disable
Trigger Input	TTGI	TTGI, TOUT	-	-

Notes: 1. In measurement mode, timer output to an external device is disabled.

2. If DMA is requested in two-phase output mode, only CCRA can determine DMA request, interrupt request, and timer output level. See "9.3 Timer Operation".

3. TCKI: External clock input
TTGI: External trigger input
TOUT: Another channel output (Timer 1 can use timer 2 output as an external clock input or a trigger input. However, timer 2 cannot use timer 1 output as an external clock input or a trigger input).

Clock Source Select 2, 1, 0 (CSS2-CSS0): CSS2-CSS0 determine the timer clock source (Table 9-3).

Table 9-3. Clock Select 2-0 and Clock Source

CSS2	CSS1	CSS0	Clock Source
0	0	0	1/8 Ø
0	0	1	Reserved (Note)
0	1	0	Reserved (Note)
0	1	1	Reserved (Note)
1	0	0	1/64 Ø
1	0	1	Reserved (Note)
1	1	0	TCKI
1	1	1	TOUT

Note: Reserved for future expansion. If a reserved value is specified, the timer operation cannot be guaranteed.

Trigger Source Select 1, 0 (TSS1-TSS0): TSS1 and TSS0 determine the corresponding timer's trigger input (Table 9-4). "Hardware trigger" starts timer counting by a trigger input selected by TSS1 and TSS0.

Table 9-4. Trigger Source Select 1, 0 and Trigger Source

TSS1	TSS0	Trigger Source
0	0	Reserved (Note)
0	1	Reserved (Note)
1	0	TTGI
1	1	TOUT

Note: Reserved for future expansion. If a reserved value is specified, the timer operation cannot be guaranteed.

DMA Request (DMA): DMA = 1 enables DMA requests from the corresponding timer to be sent to the internal DMAC. See "9.5 Timer and On-chip DMAC" for details.

Compare Interrupt Enable (CIE): CIE = 1 enables an interrupt from the corresponding timer when CCRA or CCRB matches UCR.

Overflow Interrupt Enable (OIE): OIE = 1 enables an interrupt from the corresponding timer when UCR overflows.

Measurement End Interrupt Enable (MIE): MIE = 1 enables an interrupt from the corresponding timer at the falling edge of the trigger input in measurement mode.

Output Level A and B (OLA, OLB): OLA determines the output level (level A) for the corresponding timer after UCR and CCRB match until UCR and CCRA match. OLB determines the output level (level B) from the start of counting from H'0000 until UCR and CCRB match. The output level is the same as the value set in OLA or OLB. CCRA must be greater than CCRB. If CCRA is equal to or less than CCRB, OLA and OLB determine the output level. When the timer is in two-phase output mode, OLA and OLB must be set to 0.

OLA = 1 and OLB = 1 correspond to timer output high level, OLA = 0 and OLB = 0 correspond to timer output low level.

Upcount Enable, Upcount Register Clear (UCE, UCRC)): UCE and UCRC control timer counting as shown in Table 9-5.

Table 9-5. UCE/UCRC and Counter Operation

UCE	UCRC	Counter Operation
0	0	Counter stop: Timer stops counting (UCR and output maintained).
1	0	Counter run: Timer continues counting up ("software trigger"). In one-shot mode, timer stops counting after one or two count matches.
*	1	Counter initialization: UCR is set and fixed to H'0000, timer stops counting. In two-phase mode, output is low; in other modes, output is level B.

Note: * Don't care.

Even if the timer counting is stopped by software control, a trigger input can start it counting again, unless the timer is in counter initialization mode.

Stop Point (STP): STP is used when the timer is in one-shot or two-phase output mode. In one-shot mode, STP determines whether CCRA or CCRB stops the timer. When STP = 1 while CCRA > CCRB, the timer stops when UCR matches CCRA. When STP = 0 while CCRA > CCRB, the timer stops when UCR matches CCRB.

When the timer is in two-phase mode, STP determines the phase of the outputs. When STP = 0, TIOA precedes TIOB. When STP = 1, TIOB precedes TIOA. See "9.3 Timer Operation" for details.

9.2.4 Status Register (STR)

Timer 1 and timer 2 have identical 16-bit read-only status registers, STR (Figure 9-3). It is initialized to H'0000 at reset. Table 9-6 summarizes CF, OF, and MF flags.

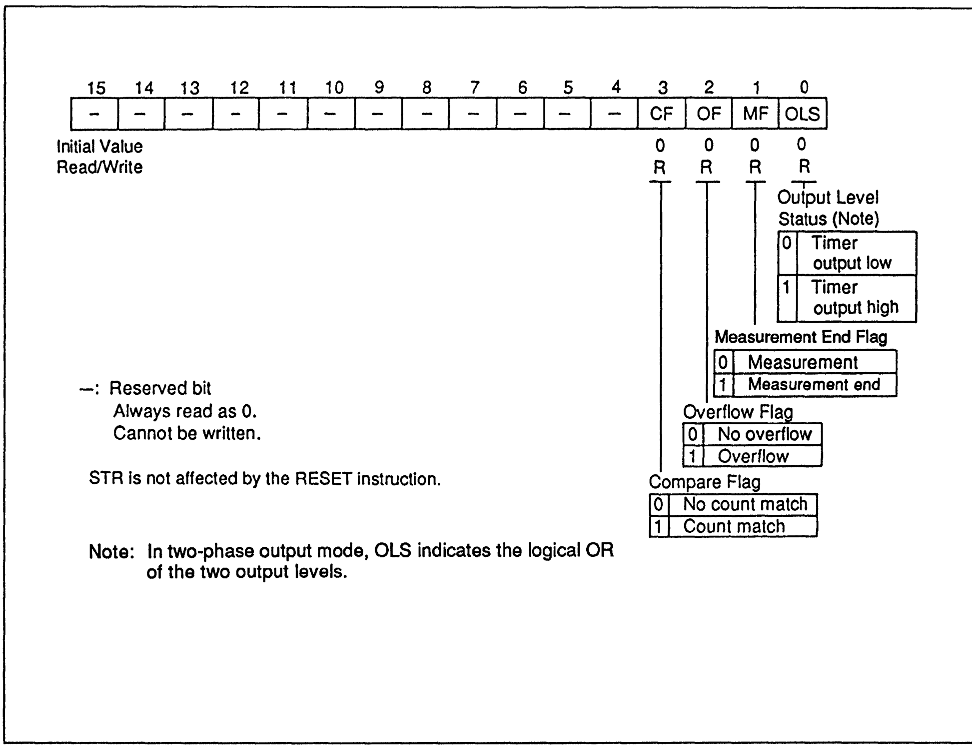


Figure 9-3. Timer 1 Status Register (STR)

Compare Flag (CF): CF is set to 1 if CCRA or CCRB matches UCR. CF is cleared to 0 if the CNTR is read after reading STR.

Overflow Flag (OF): OF is set to 1 if the UCR overflows. OF is cleared to 0 if the CCRA is written after reading STR.

Measurement End Flag (MF): MF is set to 1 at the falling edge of a trigger input in measurement mode. MF is cleared to 0 if the UCR is read after reading STR.

Output Level Status (OLS): OLS indicates the current state of the timer output. When the output is low, OLS is 0. When the output is high, OLS is 1. In two-phase output mode, OLS is the logical OR of the two output levels as shown in Figure 9-4.

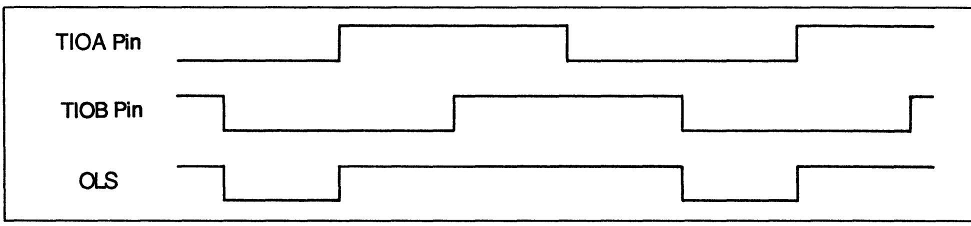


Figure 9-4. OLS Flag in Two-Phase Output Mode

Table 9-6. Status Flags' Set and Clear Conditions

Flag	Set Condition	Clear Condition
CF	UCR matches CCRA or CCRB	CNTR is read after reading STR while CF = 1.
OF	UCR overflows	CCRA is written after reading STR while OF = 1.
MF	Falling edge of trigger input is detected in measurement mode.	UCR is read after reading STR while MF = 1.

9.3 Timer Operation

Timer 1 and timer 2 operate in normal, measurement, one-shot, or two-phase output mode depending on the MDS1 and MDS0 bits' values.

9.3.1 Normal Mode

The timer can generate a rectangular waveform of any duty cycle by using CCRA and CCRB. In addition, the timer can function as interval timer or event counter. If timer output is required, the contents of CCRA must be greater than CCRB.

The output level is determined by setting the OLA and OLB bits. The OLB bit determines the output level from the start of counting until UCR matches CCRB. The OLA bit determines the output level from when UCR matches CCRB until UCR matches CCRA. Then UCR is cleared to H'0000. See Figure 9-5. The contents of CCRA must be greater than CCRB. Timer output continues while UCE = 1.

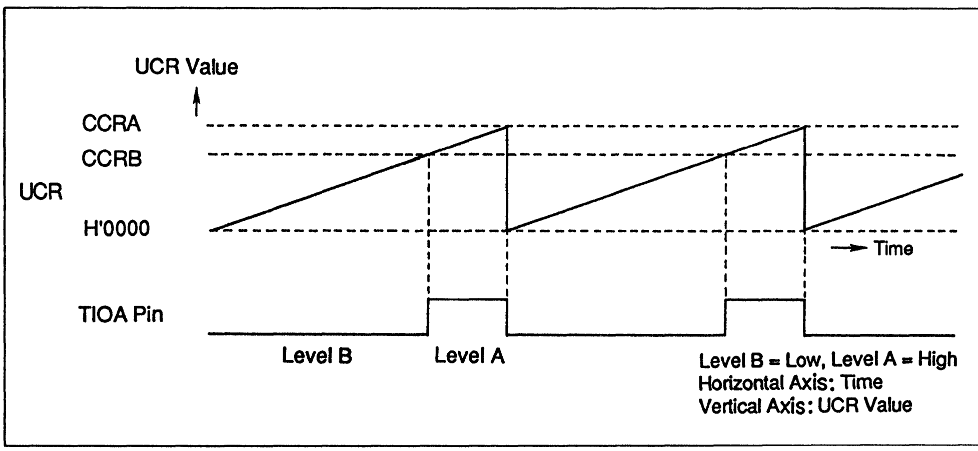


Figure 9-5. Timer 1 Normal Mode

Note that CCRA can be less than or equal to CCRB if timer output is not required. If so, TIOA always outputs level B.

9.3.2 Measurement Mode

The MDS1 and MDS0 bits in the CNTR can select the measurement mode. In measurement mode, the timer can measure high level period of an external trigger input (TIOA) or an external clock frequency (TIOB). If the timer encounters the rising edge of TIOA, it clears UCR and begins counting up while TIOA is pulled high. If the timer detects the falling edge of TIOA, it sets MF to 1 and maintains UCR. In measurement mode, since timer operation is controlled by external trigger input, the UCE bit of CNTR must be cleared. In addition, UCR functions as free-running counter and it will not be cleared even if it matches CCRA. See Figure 9-6.

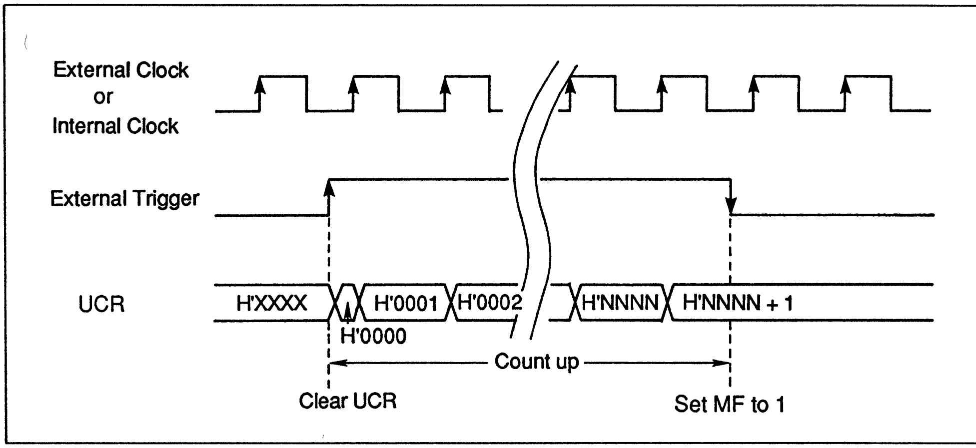


Figure 9-6. Measurement Mode Timing

9.3.3 One-Shot Mode

The MDS1 and MDS0 bits in the CNTR select the one-shot mode (Figure 9-7). In one-shot mode, timer 1 stops counting according to the STP bit.

In one-shot mode, the timer can be triggered either by hardware (trigger input) or software (UCE programming).

Note that CCRA must be greater than CCRB. If CCRA ≤ CCRB, the output level of TIOB is always level B.

One Count Match: STP = 0 selects one count match. The timer can generate a TIOB falling or rising edge at any time. At the beginning of counting, TIOB outputs level B. It changes to level A after the first count match. UCR, then, stops counting and is maintained.

Note that UCRC of CNTR must be set to 1 to initialize UCR and TIOB before restarting timer counting by software trigger. However, this operation is not required if timer counting is restarted by hardware trigger, since TIOB returns to level B on detecting the rising edge of trigger input.

Two Count Match: STP = 1 selects two count matches. Timer can generate one-shot pulse of any duty rate. At the beginning of counting, TIOB outputs level B and changes to level A after the first count match. It again changes to level B after the second count match. UCR is cleared to H'0000 and then stops counting.

Trigger: Timer counting is triggered by software or hardware.

- Software trigger: If the UCE bit of CNTR is set to 1, the timer begins counting. Timer stops counting after one or two count match and UCE is automatically cleared to 0.

- Hardware trigger: If the timer detects the rising edge of a specified trigger input, it begins counting after clearing UCR and setting TIOB to B level. Timer counting is not affected by the falling edge of the trigger input.

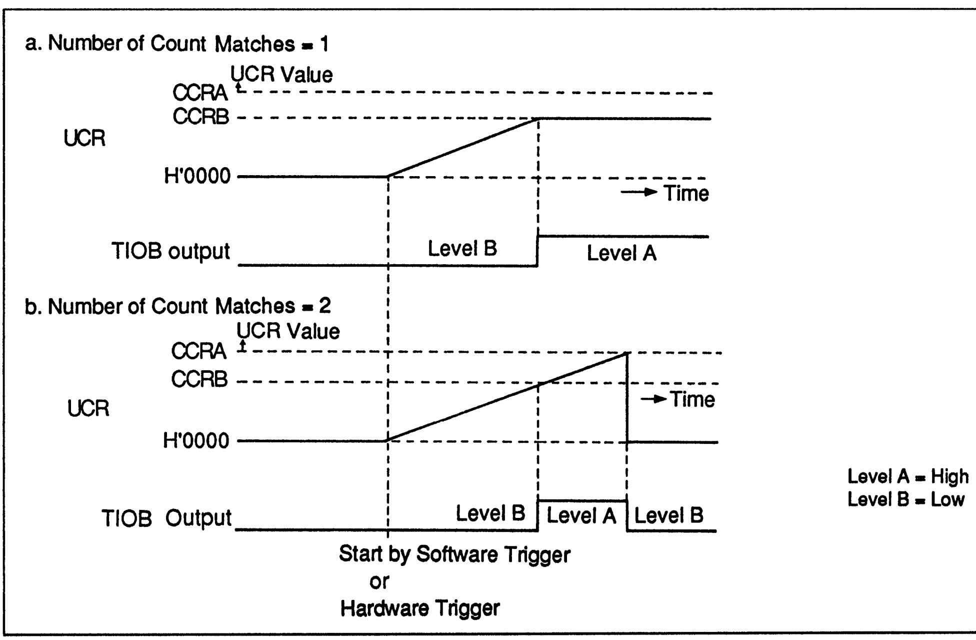


Figure 9-7. One-Shot Mode Timing

9.3.4 Two-Phase Output Mode

The MDS1 and MDS0 bits in the CNTR select the two-phase output mode. In this mode, the timer outputs two-phase pulses with 50% duty rate. The output levels are not determined by the OLA and OLB bits. Output levels of both phases begin at 0. Therefore, the OLB and OLA bits must be cleared to 0. Instead, the STP bit of the CNTR determines the phase of the outputs. TIOA precedes when STP = 0 and TIOB precedes when STP = 1. See Figure 9-8.

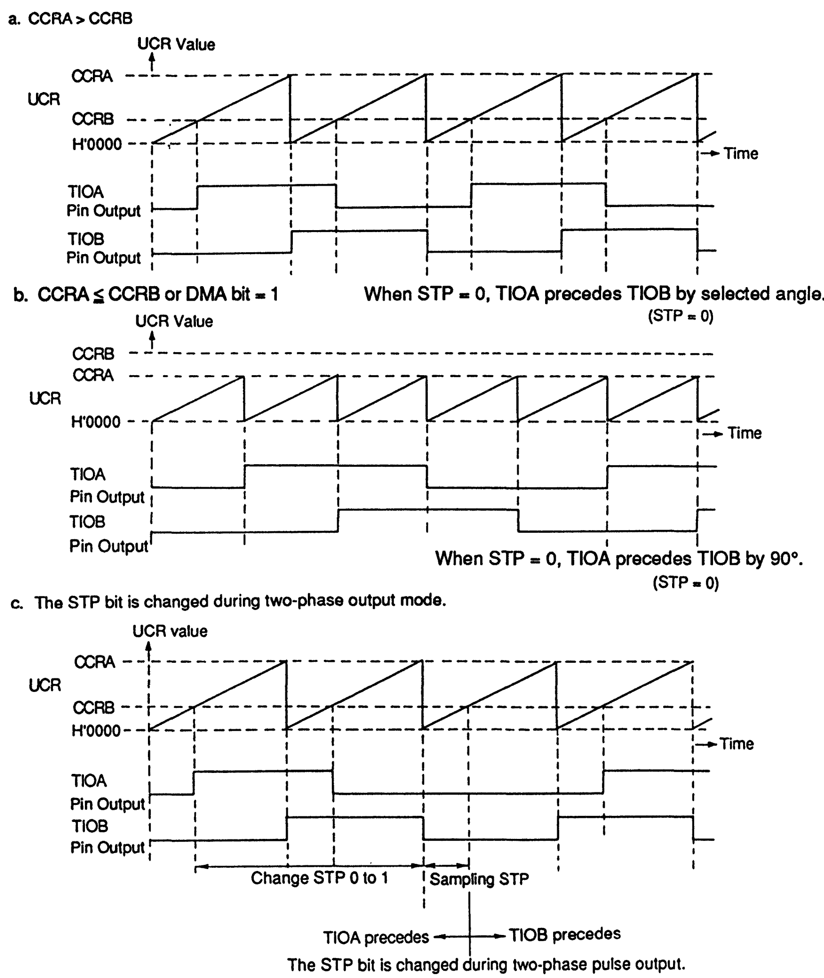


Figure 9-8. Two-Phase Output Mode

TIOA Precedes TIOB by Arbitrary Angle: Variable two-phase output can be obtained if CCRA > CCRB as shown in Figure 9-8. Output phase changes depending on the CCRA and CCRB values.

TIOA Precedes TIOB by 90°: In addition, two-phase output of ± 90° can be obtained if CCRA ≦ CCRB. The timer output is determined only by CCRA.

If the DMA bit of CNTR is set to 1, the timer outputs are determined only by CCRA. At this time, the timer does not request a DMA transfer even if UCR matches CCRB and the CF bit of STR remains cleared.

If STP is modified during two-phase output, the bit is sampled when both outputs go low. The phase change takes place at the next count match and begins from "0, 0".

9.4 Timer Application

The following paragraphs show how to use the timer for various functions. Note that t_{cyc} (sec) is defined as one Ø clock cycle time in the following descriptions.

9.4.1 Interval Timer

Function: Timer requests a count match interrupt every 1024 t_{cyc}.

Operating Procedure: It is assumed that the timer uses 1/8 Ø clock.

1. Write H'0080 (=1024 ÷ 8) to CCRA.

2. Write H'8084 (normal mode, 1/8 Ø clock, count match interrupt enable, count run) to CNTR.

3. Clear CF to enable a next interrupt when a count match interrupt is accepted.

4. Repeat operation 3.

9.4.2 Event Counter

Function: Timer counts external events (rising edge of TIOB) input.

Operating Procedure:

1. Write H'B004 (normal mode, external clock, count run) to CNTR.

2. Read UCR if necessary.

9.4.3 One-Phase Pulse Generator (PWM)

Function: Timer outputs one-phase pulse as shown in Figure 9-9.

Operating Procedure: It is assumed that the timer uses 1/8 Ø clock.

1. Write H'0080 (= 1024 ÷ 8) to CCRB.

2. Write H'00C0 (= 1536 ÷ 8) to CCRA.

3. Write H'800C (normal mode, 1/8 Ø clock, level A = 1, level B = 0, count run) to CNTR.

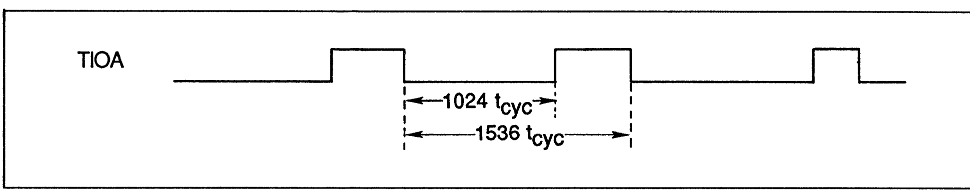


Figure 9-9. One-Phase Pulse Generator Timing

9.4.4 Two-Phase Pulse Generator

Function: Timer generates two-phase pulse (Figure 9-10).

Operating Procedure: It is assumed that the timer uses 1/8 Ø clock.

1. Write H'0080 (= 1024 ÷ 8) to CCRB.

2. Write H'00C0 (= 1536 ÷ 8) to CCRA.

3. Write H'C005 (two-phase output mode, 1/8 Ø clock, count run, TIOB precedes) to CNTR.

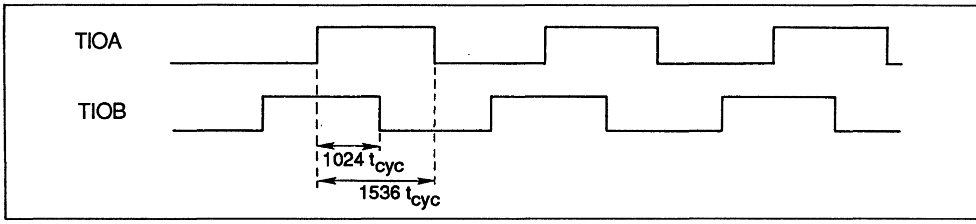


Figure 9-10. Two-Phase Pulse Generator Timing

9.4.5 ±90° Two-Phase Pulse Generator

Function: Timer generates two-phase pulse of ± 90° as shown in Figure 9-11.

Operating Procedure: It is assumed that the timer uses 1/8 Ø clock.

1. Write H'0040 (= 512 ÷ 8) to CCRA.

2. Write H'C004 (two-phase output mode, 1/8 Ø clock, count run, TIOA precedes) to CNTR.

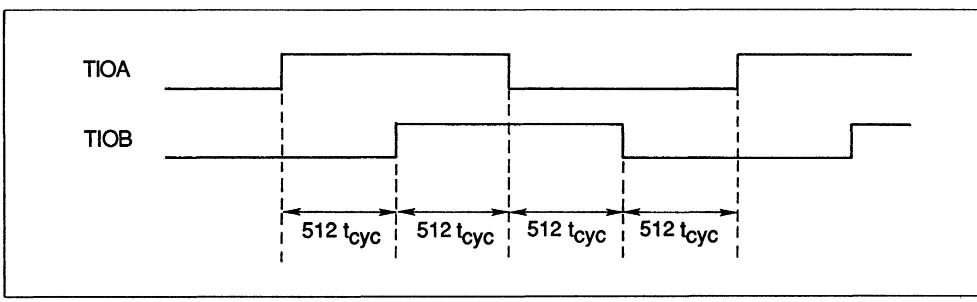


Figure 9-11. ±90° Two-Phase Pulse Generator Timing

9.4.6 Pulse Width Measurement

Function: Timer measures high level period of an external trigger (TIOA) (Figure 9-12).

Operating Procedure: It is assumed that the timer uses 1/8 Ø clock.

1. Write H'0420 (measurement mode, 1/8 Ø clock, external trigger, measurement end interrupt enable, count stop) to CNTR.

2. Input a pulse to be measured through TIOA.

3. Read UCR to obtain high-level period when the timer accepts a measurement end interrupt.

It is calculated as H'80 (128) x 8 = 1024. Note that there is a maximum error of ± a clock cycle.

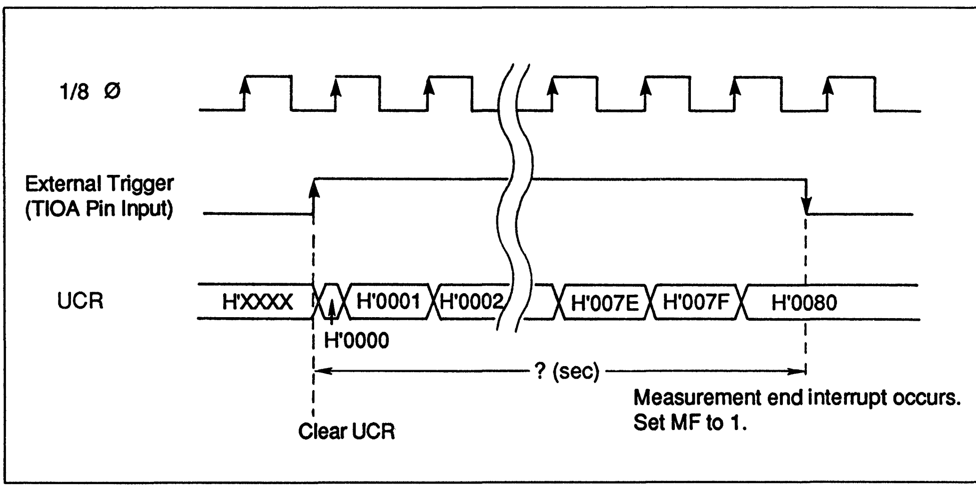


Figure 9-12. Pulse Width Measurement Timing

9.4.7 Frequency Measurement

Function: Timer measures an external clock (TIOB input) frequency (Figure 9-13).

Operating Procedure:

1. Write H'3420 (measurement mode, external clock, external trigger, measurement end interrupt enable, count stop) to CNTR.

2. Input an external trigger through TIOA.

3. Read UCR to obtain an external clock frequency if the timer accepts a measurement end interrupt.

The external clock frequency is calculated as 1024 ÷ H'80 (= 128) = 8. Accordingly, it is 1/8 Ø (Hz).

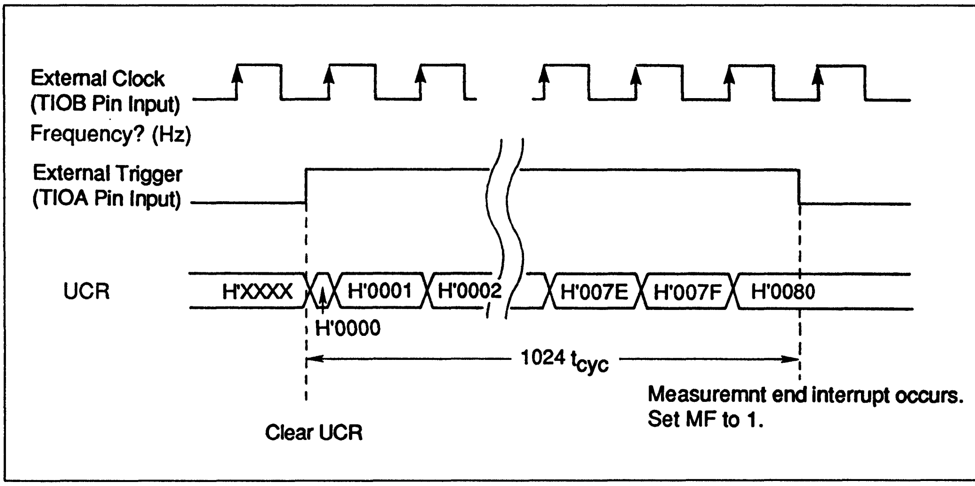


Figure 9-13. Frequency Measurement Timing

9.4.8 Software-Triggered One-Shot (1)

Function: Timer is activated by software trigger and TIOB output falls after 1024 t_{cyc} (Figure 9-14).

Operating Procedure: It is assumed that the timer uses 1/8 Ø clock.

1. Write H'0080 (= 1024 ÷ 8) to count compare register (CCRB).

2. Write H'4014 (one-shot mode, 1/8 Ø clock, level B = 1, level A = 0, count run, one count match) to count control register (CNTR). TIOB, then, falls after 1024 t_{cyc}.

3. Write H'4012 (counter initialize) to CNTR for bringing TIOB to level B to restart.

Perform operations 2 and 3 to restart.

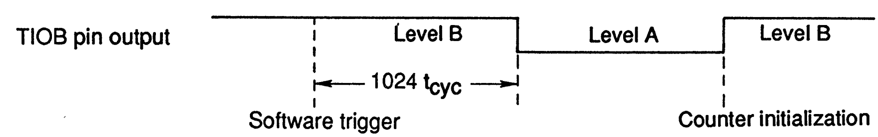


Figure 9-14. Software-Triggered One-Shot (1) Timing

9.4.9 Hardware-Triggered One-Shot (1)

Function: Timer is activated by hardware trigger and TIOB output rises after 1024 t_{cyc} (Figure 9-15).

Operating Procedure: It is assumed that the timer uses 1/8 Ø clock.

1. Write H'0080 (= 1024 ÷ 8) to CCRB.

2. Write H'4408 (one-shot mode, 1/8 Ø clock, external trigger, level B = 0, level A = 1, count stop, one count match) to CNTR.

3. TIOB rises 1024 t_{cyc} after an external trigger input has risen.

Perform operation 3 to restart. TIOB automatically becomes level B at the rising edge of trigger input.

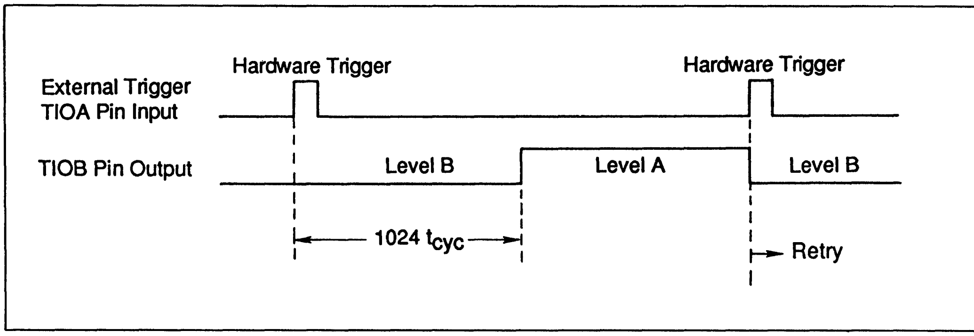


Figure 9-15. Hardware-Triggered One-Shot (1) Timing

9.4.10 Software-Triggered One-Shot (2)

Function: Timer is activated by software trigger and generates one-shot pulse (Figure 9-16).

Operating Procedure: It is assumed that the timer uses 1/8 Ø clock.

1. Write H'0080 (= 1024 ÷ 8) to CCRB.

2. Write H'0088 (= (1024 + 64) ÷ 8) to CCRA.

3. Write H'4015 (one-shot mode, 1/8 Ø clock, level A = 0, level B = 1, count run, two count matches) to CNTR. The timer then generates one-shot pulse.

Perform operation 3 to restart.

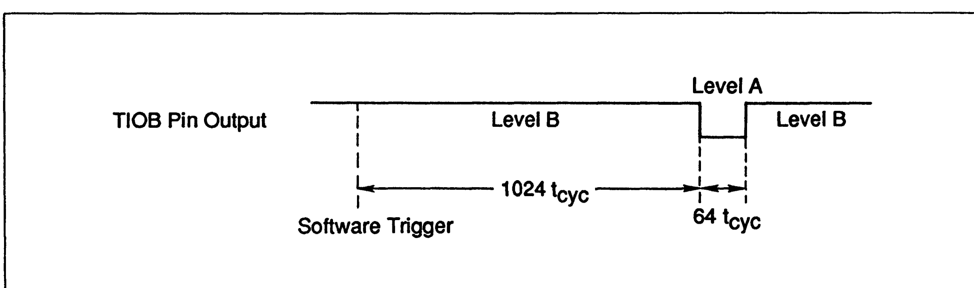


Figure 9-16. Software-Triggered One-Shot (2) Timing

9.4.11 Hardware-Triggered One-Shot (2)

Function: Timer is activated by hardware trigger and generates a one-shot pulse as shown in Figure 9-17.

Operating Procedure: It is assumed that the timer uses 1/8 Ø clock.

1. Write H'0080 (= 1024 ÷ 8) to CCRB.

2. Write H'0088 (= (1024 + 64) ÷ 8) to CCRA.

3. Write H'4409 (one-shot mode, 1/8 Ø clock, external trigger, level B = 0, level A = 1, count stop, two count matches) to CNTR.

4. Input an external trigger rising edge to obtain a one-shot pulse.

Perform operation 4 to restart.

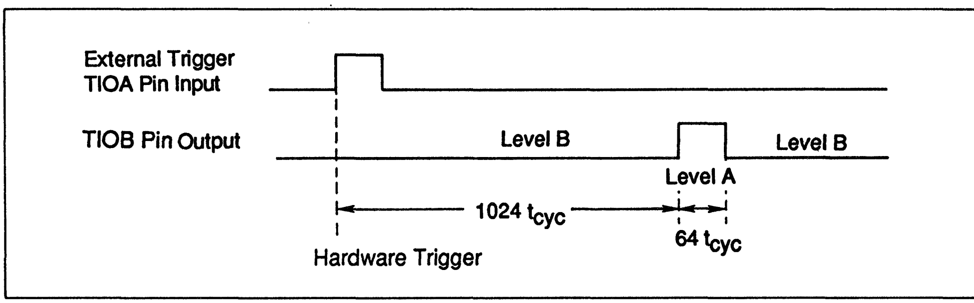


Figure 9-17. Hardware-Triggered One-Shot (2) Timing

9.5 Timer and On-Chip DMAC

The HD641016 supports DMA transfers to and from timers 1 and 2.

If the DMA bit of CNTR is set to 1, the timer generates a DMA transfer request whenever UCR matches CCRA or CCRB. The DMAC must be programmed as edge sensing to detect above.

Note that if the timer requests DMA transfer in two-phase output mode, the timer requests a DMA transfer only when UCR matches CCRA.

9.6 Timer Interrupt Priority

Table 9-7 shows the priority when multiple interrupts occurs simultaneously.

Table 9-7. Timer Interrupt Priority

Priority	Interrupt Elements
High	Count match
	Measurement end
Low	Overflow

9.7 Timer and Reset

Table 9-8 summarizes timer status just after reset. If a timer is reset during its operation, all timer registers are initialized. See "9.2 Timer Registers" for details.

Table 9-8. Timer After Reset

Item	Timer State
Operation mode	Measurement mode
Clock source	1/8 Ø
Trigger input	None
DMA transfer request	Disabled
Interrupt request	Disabled
Timer output	Low
Timer counting	Stopped
Count match	1
Precede	TIOB precedes

9.8 Timer Notes

1. Timer counting must be stopped before mode, count clock input, or trigger input is modified.

2. When UCR is set to the same value as that of CCRA or CCRB, a count match does not occur. When UCR equals to CCRA, the timer operates normally after UCR overflows.

3. Either the upper or lower half of CNTR or STR can be accessed by bytes. However, both upper and lower halves of UCR, CCRA or CCRB must be accessed sequentially in byte access. If the upper half is accessed in the timer after accessing the upper bit, for example, the lower half data will be destroyed since 16-bit data becomes valid at the upper bit access.

4. The timer output changes when a compare match occurs 4.5 to 8.5 states after external clock input, as shown in Figure 9-18. The external clock is sampled at its rising edge.

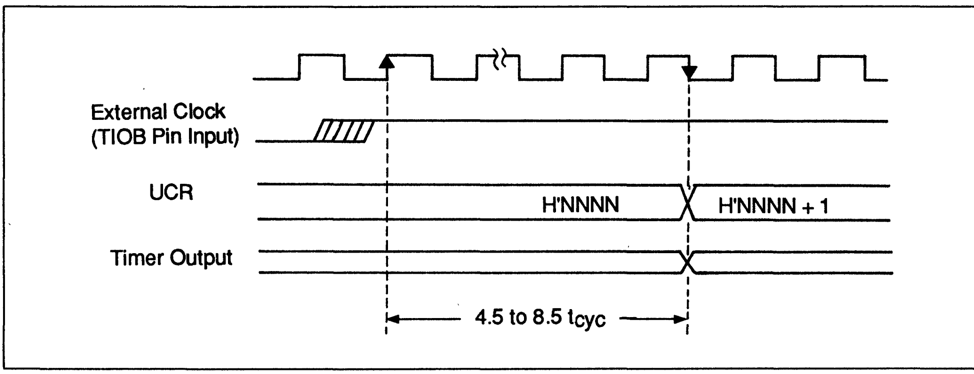


Figure 9-18. Timer Output Timing

5. Figure 9-19 shows the timing when the TIOB output level changes and UCR is cleared to H'0000 after TIOA has risen as an external trigger.

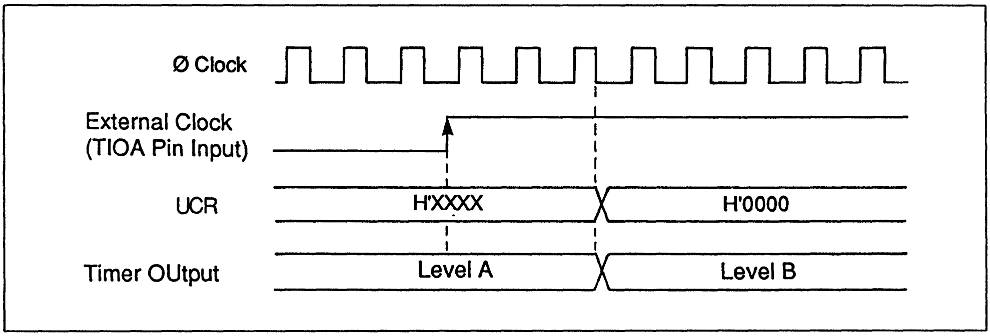


Figure 9-19. TIOB Output Timing

6. The maximum count frequency of an external clock is 1/8 Ø.

7. When setting the CNTR, do not simultaneously set the UCRC bit to 1 (count initialization) and switch the OLA bit (level A output). Set level B during count initialization and then set level A if necessary.

Section 10. Asynchronous Serial Communication Interface

10.1 Overview

The asynchronous serial communication interface (ASCI, Figure 10-1) has two software-selectable modes: asynchronous and clock synchronous. The ASCI has the following features:

- Full-duplex communication
- Data format: 7 or 8 bits/character
- Stop bit: 1 or 2 bits
- Parity: Even, odd, or no parity
- Bit rate: 1/1, 1/16, 1/32, or 1/64 clock rate for asynchronous mode
- Programmable baud rate generator
- Clock source: External clock or BRG output
- Modem control signals: CTS, DCD, and RTS
- Multiprocessor communication capability
- Built-in DMAC interface
- Local loopback and auto-echo functions

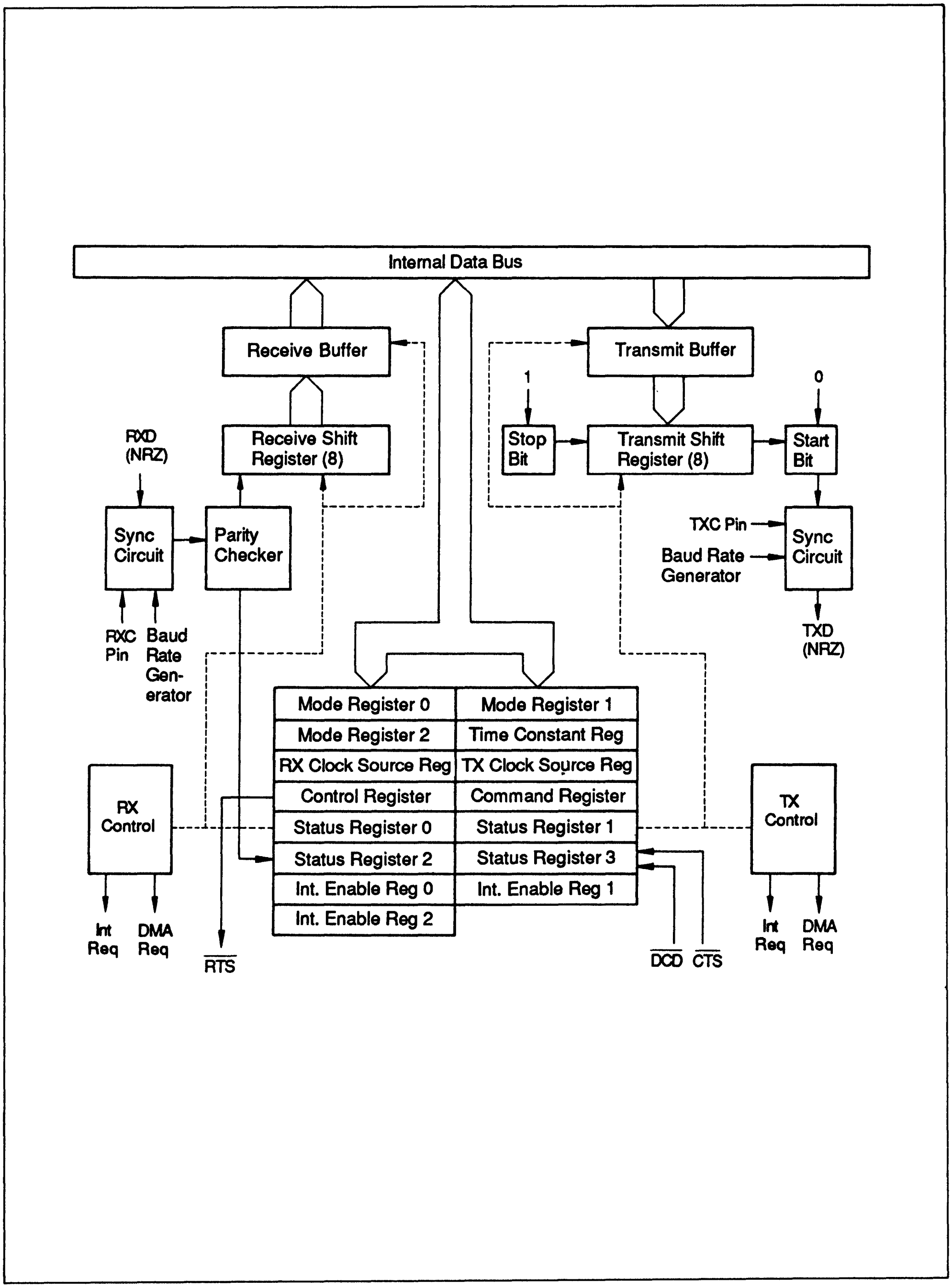


Figure 10-1. Block Diagram of Asynchronous Serial Communication Interface

10.2 ASCI Registers

ASCI registers are all 8-bit registers assigned to consecutive memory addresses on a byte basis. Accordingly, these registers cannot be accessed on a word or long word basis. Table 10-1 shows the ASCI registers.

Table 10-1. ASCI Registers

	Register	Symbol	Address Offset	R/W	Initial Value	Size
ASCI0	TX/RX buffer register	TRB	H'FF58	R/W	Undefined	B
	Status register 0	ST0	H'FF59	R	H'00	B
	Status register 1	ST1	H'FF5A	R/W (Note 1)	H'00	B
	Status register 2	ST2	H'FF5B	R/W (Note 1)	H'00	B
	Status register 3	ST3	H'FF5C	R	H'00 (Note 2)	B
	Interrupt enable register 0	IE0	H'FF5E	R/W	H'00	B
	Interrupt enable register 1	IE1	H'FF5F	R/W	H'00	B
	Interrupt enable register 2	IE2	H'FF60	R/W	H'00	B
	Command register	CMD	H'FF62	W	— (Note 3)	B
	Mode register 0	MD0	H'FF63	R/W	H'00	B
	Mode register 1	MD1	H'FF64	R/W	H'00	B
	Mode register 2	MD2	H'FF65	R/W	H'00	B

Table 10-1. ASCI Registers (cont.)

	Register	Symbol	Address Offset	R/W	Initial Value	Size
ASCI0 (cont.)	Control register	CTL	H'FF66	R/W	H'01	B
	Time constant register	TMC	H'FF6A	R/W	H'01	B
	RX clock source register	RXS	H'FF6B	R/W	H'00	B
	TX clock source register	TXS	H'FF6C	R/W	H'00	B
ASCI1	TX/RX buffer register	TRB	H'FF70	R/W	Undefined	B
	Status register 0	ST0	H'FF71	R	H'00	B
	Status register 1	ST1	H'FF72	R/W(Note 1)	H'00	B
	Status register 2	ST2	H'FF73	R/W(Note 1)	H'00	B
	Status register 3	ST3	H'FF74	R	H'00(Note 2)	B
	Interrupt enable register 0	IE0	H'FF76	R/W	H'00	B
	Interrupt enable register 1	IE1	H'FF77	R/W	H'00	B
	Interrupt enable register 2	IE2	H'FF78	R/W	H'00	B
	Command register	CMD	H'FF7A	W	(Note 3)	B
	Mode register 0	MD0	H'FF7B	R/W	H'00	B
	Mode register 1	MD1	H'FF7C	R/W	H'00	B
	Mode register 2	MD2	H'FF7D	R/W	H'00	B
	Control register	CTL	H'FF7E	R/W	H'01	B

Table 10-1. ASCI Registers (cont.)

	Register	Symbol	Address Offset	R/W	Initial Value	Size
ASCI1 (cont.)	Time constant register	TMC	H'FF82	R/W	H'01	B
	RX clock source register	RXS	H'FF83	R/W	H'00	B
	TX clock source register	TXS	H'FF84	R/W	H'00	B

Notes: 1. Cleared by writing 1 to a bit, not affected by writing 0.

2. Bits 3-2: $\overline{CTS}$ and $\overline{DCD}$ input levels.

3. Always read as 0.

10.2.1 Mode Register 0 (MD0)

Figure 10-2 shows mode register 0 (MD0). This register is not affected by the RESET instruction.

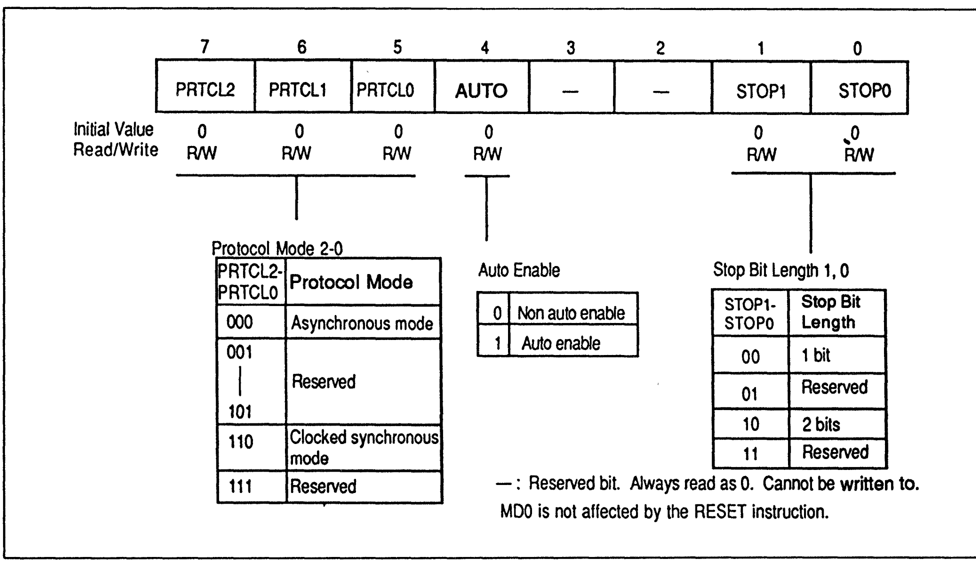


Figure 10-2. Mode Register 0 (MD0)

Protocol Mode (PRTCL2-PRTCL0): The PRTCL2-PRTCL0 bits specify the ASCI protocol mode: 000 = asynchronous mode, 110 = clock synchronous mode. If PRTCL2-PRTCL0 are not set to 000 or 110 during ASCI operation, a malfunction may occur. In addition, PRTCL2-PRTCL0 should be modified just after reset or a write to the command register by the channel reset command. See Table 10-2.

Table 10-2. PRTCL2-PRTCL0 and ASCI Protocol Mode

PRTCL2	PRTCL1	PRTCL0	Protocol Mode
0	0	0	Asynchronous mode
0	0	1	Reserved (Note)
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	Clock synchronous mode
1	1	1	Reserved (Note)

Note: Reserved for future expansion. If a reserved value is specified, the ASCI may malfunction.

Auto Enable (AUTO): AUTO determines the function of modem control signals $\overline{CTS}$, $\overline{DCD}$, and $\overline{RTS}$.

If AUTO = 0, $\overline{CTS}$ and $\overline{DCD}$ are general-purpose input pins, and $\overline{RTS}$ is a general-purpose output pin independent of receive/transmit operation.

If AUTO = 1, $\overline{CTS}$, $\overline{DCD}$, and $\overline{RTS}$ function as modem control signals. $\overline{CTS}$ functions as the ASCI transmission control pin. If $\overline{CTS}$ is high, data transmission from the TX buffer to the TX shift register is disabled. The transmitter idles after it transmits data from the TX shift register. $\overline{DCD}$ functions as the ASCI receive control pin. If $\overline{DCD}$ is high, reception is disabled. If the $\overline{DCD}$ goes high during character assembly, the data currently being assembled is lost. However, data in the receive buffer is retained. $\overline{RTS}$ indicates the ASCI transmit operation state. During the ASCI transmission, $\overline{RTS}$ is asserted low independently of the $\overline{RTS}$ bit of the CTL register. During TX disable or idle state, $\overline{RTS}$ reflects the $\overline{RTS}$ bit value.

Stop Bit Length (STOP1, STOP0): In asynchronous mode, STOP1 and STOP0 determine the stop bit length of the transmit data: 1 or 2 stop bits (Table 10-3). STOP1 and STOP0 can be modified during ASCI transmission. The new value becomes valid for the current transmit data.

In clock synchronous mode, STOP1 and STOP0 are not used for ASCI operation.

Table 10-3. STOP1-STOP0 and Stop Bits Length

STOP1	STOP0	Stop Bit Length
0	0	1 stop bit
0	1	Reserved(Note)
1	0	2 stop bits
1	1	Reserved(Note)

Note: Reserved for future expansion. If a reserved value is specified, the ASCI may malfunction.

10.2.2 Mode Register 1 (MD1)

Figure 10-3 shows mode register 1 (MD1). This register is not affected by the RESET instruction.

	7	6	5	4	3	2	1	0
	BRATE1	BRATE0	TXCHR1	TXCHR0	RXCHR1	RXCHR0	PMPM1	PMPM0
Initial Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit Rate 1, 0

BRATE1-BRATE0	Bit Rate
00	1/1 clock rate
01	1/16 clock rate
10	1/32 clock rate
11	1/64 clock rate

TX Character Length 1, 0

TXCHR1-TXCHR0	Tx Character Length
00	8 bits
01	7 bits
10	Reserved
11	Reserved

RX Character Length 1, 0

RXCHR1-RXCHR0	RX Character Length
00	8 bits
01	7 bits
10	Reserved
11	Reserved

Parity/Multiprocessor Mode 1, 0

PMP1-PMP0	Parity/ Multiprocessor Mode
00	None
01	MP bit
10	Even parity
11	Odd parity

MD1 is not affected by the RESET instruction.

Figure 10-3. Mode Register 1 (MD1)

Bit Rate (BRATE1, BRATE0): In asynchronous mode, BRATE1 and BRATE0 specify the relation between the transmitter/receiver clock and the bit rate. In clock synchronous mode, they must be always set to 00 (1/1 clock rate). See Table 10-4.

BRATE1 and BRATE0 must not be changed during transmission or reception. If they are, the ASCI may malfunction.

Table 10-4. BRATE1-BRATE0 and Bit Rate

BRATE1	BRATE0	Bit Rate
0	0	1/1 clock rate
0	1	1/16 clock rate
1	0	1/32 clock rate
1	1	1/64 clock rate

TX Character Length (TXCHR1, TXCHR0): TXCHR1 and TXCHR0 specify the transmit character length. They can be changed during transmission. The new value becomes valid at the next transmit character. See Table 10-5.

Table 10-5. TXCHR1-TXCHR0 and Transmit Character Length

TXCHR1	TXCHR0	Transmit Character Length
0	0	8 bits/character
0	1	7 bits/character
1	0	Reserved(Note)
1	1	Reserved(Note)

Note: Reserved for future expansion. If a reserved bit is specified, the ASCI may malfunction.

RX Character Length (RXCHR1, RXCHR0): RXCHR1 and RXCHR0 specify the receive character length. If they are changed during reception, the new value becomes valid at the next receive character. See Table 10-6.

Table 10-6. RXCHR1-RXCHR0 and Receive Character Length

RXCHR1	RXCHR0	Receive Character Length
0	0	8 bits/character
0	1	7 bits/character
1	0	Reserved(Note)
1	1	Reserved(Note)

Note: Reserved for future expansion. If a reserved value is specified, the ASCI may malfunction.

Parity/Multiprocessor Mode (PMPM1, PMPM0): PMPM1 and PMPM0 specify whether even, odd, or no parity will be used in asynchronous mode, or if a multiprocessor bit (MP) will be used instead of the parity bit. They can be changed during operation. The new value becomes valid at the next transmit/receive character. See Table 10-7.

Table 10-7. PMPM1-PMPM0 and Bit Rate

PMPM1	PMPM0	Parity/MP
0	0	No parity/No MP bit
0	1	MP bit
1	0	Even parity
1	1	Odd parity

10.2.3 Mode Register 2 (MD2)

Figure 10-4 shows the bits of mode register 2 (MD2). This register is not affected by the RESET instruction.

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	CNCT1	CNCT0
Initial Value							0	0
Read/Write							R/W	R/W

Channel Connection 1, 0

CNCT1-CNCT0	Transmit/ Receive Mode
00	Full-duplex
01	Auto-echo
10	Not used
11	Local loopback

– : Reserved bit. Always read as 0. Cannot be written to.

MD2 is not affected by the RESET instruction.

Figure 10-4. Mode Register 2 (MD2)

Channel Connection (CNCT1, CNCT0): CNCT1 and CNCT0 specify the operation of the transmitter and receiver. Full-duplex is the normal operation mode. See "10.3 ASCI Asynchronous Mode Operation" and "10.4 ASCI Clock Synchronous Mode Operation" for details.

In auto-echo mode, the RXD input is output directly to the TXD pin. Reception is enabled, but transmission is disabled.

In local loopback mode, the data from the TX shift register goes to the RX shift register. In addition, data input to the RXD pin is directly output to the TXD pin. See Table 10-8.

Table 10-8. CNCT1-CNCT0 and Transmit/Receive Mode

CNCT1	CNCT0	Transmit/Receive Mode
0	0	Full-duplex mode
0	1	Auto-echo mode
1	0	Reserved(Note)
1	1	Local loopback mode

Note: Reserved for future expansion. If a reserved value is specified, the ASCI may malfunction.

10.2.4 Control Register (CTL)

Figure 10-5 shows the bits of the control register (CTL). This register is not affected by the RESET instruction.

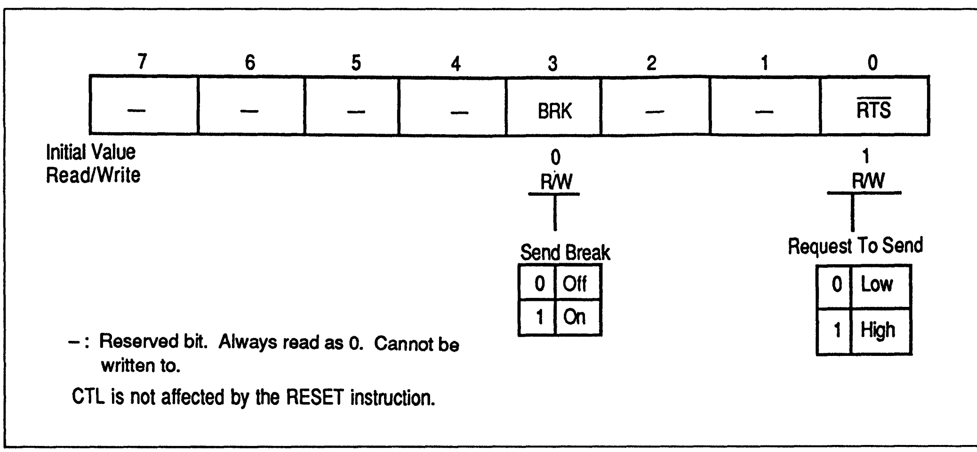


Figure 10-5. Control Register (CTL)

Send Break (BRK): In asynchronous mode, writing a 1 to the BRK bit sends a break pattern from the next falling edge of the TXC clock. TXD sends spaces (0) and the TX shift register is cleared. TXD must be held low for at least two character periods to ensure that the break is correctly received. See "10.3.9 Break Send/Detector" for details.

In clock synchronous mode, BRK must always be 0. If set to 1, the ASCI may malfunction. See Table 10-9.

Table 10-9. BRK Bit and Transmit Operation Mode

BRK	Transmit Mode
0	Normal transmission
1	Break send

Request to Send ($\overline{RTS}$): Writing a 0 to $\overline{RTS}$ sets the $\overline{RTS}$ output low and writing a 1 to $\overline{RTS}$ sets the $\overline{RTS}$ output high.

In auto-enable mode (when the AUTO bit of the MD0 register is set), the $\overline{RTS}$ output goes low while one character is transmitted in asynchronous mode regardless of the $\overline{RTS}$ bit value.

10.2.5 RX Clock Source Register (RXS)

Figure 10-6 shows the bits of the RX clock source register. This register is not affected by the RESET instruction.

	7	6	5	4	3	2	1	0
	—	RXCS2	RXCS1	RXCS0	(Note)	(Note)	(Note)	(Note)
Initial Value		0	0	0				
Read/Write		R/W	R/W	R/W				

RX Clock Source Select 2- 0

Asynchronous Mode	000	RXC input
	100	BRG output
Clock synchronous Mode	000	Slave mode
	100	Master mode

–: Reserved bit. Always read as 0. Cannot be written to.
RXS is not affected by the RESET instruction.
Note: Reserved bit. Set to the same value as the TXBR3-TXBR6 bits of TXS.

Figure 10-6. RX Clock Source Register (RXS)

RX Clock Source Select (RXCS2-RXCS0): RXCS2-RXCS0 select the receiver clock source in asynchronous mode. If RXCS2-RXCS0 = 000, the source is the RXC input. If RXCS2-RXCS0 = 100, the source is the on-chip baud rate generator (BRG) output. The BRG output is transmitted from the RXC pin. See Table 10-10.

Table 10-10. RXCS2-RXCS0 and Receive Clock Source in Asynchronous Mode

RXCS2	RXCS1	RXCS0	Receive Clock Source
0	0	0	RXC input
0	0	1	Reserved(Note)
0	1	0	
0	1	1	
1	0	0	BRG output
1	0	1	Reserved(Note)
1	1	0	
1	1	1	

Note: Reserved for future expansion. If a reserved value is specified, the ASCI may malfunction.

In clock synchronous mode, RXCS2-RXCS0 select slave mode or master mode. If RXCS2-RXCS0 = 000, the receiver is configured in slave mode. If RXCS2-RXCS0 = 100, the receiver (and transmitter) are configured in master mode. At this time, the transmitter must be also configured in master mode for receiving data correctly. Refer to "10.4 ASCI Clock Synchronous Mode Operation" for details. See Table 10-11.

Table 10-11. RXCS2-RXCS0 and Receiver Mode in Clock Synchronous Mode

RXCS2	RXCS1	RXCS0	Receiver Mode
0	0	0	Slave mode
0	0	1	Reserved(Note)
0	1	0	
0	1	1	
1	0	0	Master mode
1	0	1	Reserved(Note)
1	1	0	
1	1	1	

Note: Reserved for future expansion. If a reserved value is specified, the ASCI may malfunction.

RXCS2-RXCS0 must be set while both the receiver and transmitter are disabled or idled; otherwise, the ASCI may malfunction.

Reserved Bits 3-0: Reserved bits 3-0 of the RXS register must be set to the same value as the TXBR3-TXBR0 bits of the TXS register for future expansion.

10.2.6 TX Clock Source Register (TXS)

Figure 10-7 shows the bits of the TX clock source register. This register is not affected by the RESET instruction.

Bit	7	6	5	4	3	2	1	0
	—	TXCS2	TXCS1	TXCS0	TXBR3	TXBR2	TXBR1	TXBR0
Initial Value		0	0	0	0	0	0	0
Read/Write		R/W	R/W	R/W	R/W	R/W	R/W	R/W

TX Clock Source Select 2 - 0

Asynchronous Mode	000	TXC input
	100	BRG output
Clock Synchronous Mode	000	Slave mode
	100	Master mode

TX Baud Rate Select 3-0

0000	+1
0001	+2
0010	+4
0011	+8
0100	+16
0101	+32
0110	+64
0111	+128
1000	+256
1001	+512
1010 | 1111	Reserved

–: Reserved bit. Always read as 0. Cannot be written to.

TXS is not affected by the RESET instruction.

Figure 10-7. TX Clock Source Register (TXS)

TX Clock Source Select (TXCS2-TXCS0): TXCS2-TXCS0 select the transmitter clock source in asynchronous mode. If TXCS2-TXCS0 = 000, the source is the TXC input. If TXCS2-TXCS0 = 100, the source is the BRG output and the TXC pin outputs the BRG output. See Table 10-12.

Table 10-12. TXCS2-TXCS0 and Transmitter Clock Source

TXCS2	TXCS1	TXCS0	Transmitter Clock Source
0	0	0	TXC input
0	0	1	Reserved(Note)
0	1	0	
0	1	1	
1	0	0	BRG output
1	0	1	Reserved(Note)
1	1	0	
1	1	1	

Note: Reserved for future expansion. If a reserved value is specified, the ASCI may malfunction.

In clock synchronous mode, TXCS2-TXCS0 select slave or master mode. TXCS2-TXCS0 must be changed only during ASCI transmit and receive disable or idle state, and must only be set to 000 or 100; otherwise the ASCI may malfunction. Refer to "10.4 ASCI Clock Synchronous Mode Operation" for details. See Table 10-13.

Table 10-13. TXCS2-TXCS0 and Transmitter Mode

TXCS2	TXCS1	TXCS0	Transmitter Mode
0	0	0	Slave mode
0	0	1	Reserved(Note)
0	1	0	
0	1	1	
1	0	0	Master mode
1	0	1	Reserved(Note)
1	1	0	
1	1	1	

Note: Reserved for future expansion. If a reserved value is specified, the ASCI may malfunction.

TXCS2-TXCS0 must be changed during transmitter and receiver disable state or idle state. If they are changed during ASCI operation, a malfunction may occur.

TX Baud Rate Select (TXBR3-TXBR0): TXBR3-TXBR0 determine the divide ratio of the reload timer output to generate the baud rate generator (BRG) output as shown in Table 10-14. The BRG output is used in the transmitter and receiver. See "10.7 Baud Rate Generator" for details.

Table 10-14. TXBR3-TXBR0 and Divide Ratio of BRG Reload Timer Output

TXBR3-TXBR0	Divide Ratio of BRG Reload Timer Output
0 0 0 0	÷1
0 0 0 1	÷2
0 0 1 0	÷4
0 0 1 1	÷8
0 1 0 0	÷16
0 1 0 1	÷32
0 1 1 0	÷64
0 1 1 1	÷128
1 0 0 0	÷256
1 0 0 1	÷512
1010-1111	Reserved for future expansion. If selected, the baud rate divide ratio is undefined.

10.2.7 Time Constant Register (TMC)

Figure 10-8 shows the time constant register which contains the value to be loaded into the reload timer in the baud rate generator. This register is not affected by the RESET instruction.

	7	6	5	4	3	2	1	0
	TMC7	TMC6	TMC5	TMC4	TMC3	TMC2	TMC1	TMC0
Initial Value	0	0	0	0	0	0	0	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Time Constant 7—0
Value to be loaded to the Reload Timer (fØ/TMC)

TMC is not affected by the RESET instruction.

Figure 10-8. Time Constant Register (TMC)

Time Constant 7-0 (TMC7-TMC0): TMC7-TMC0 specify the data to be loaded into the reload timer in the baud rate generator. The timer output frequency equals fØ/TMC. See "10.7 Baud Rate Generator" for details.

10.2.8 Command Register (CMD)

Writing to bits 5 through 0 of the command register (Figure 10-9) specifies a command. The command register is always read as 0. This register is not affected by the RESET instruction.

	7	6	5	4	3	2	1	0
	—	—	CMD5	CMD4	CMD3	CMD2	CMD1	CMD0
Initial Value			0	0	0	0	0	0
Read/write			W	W	W	W	W	W

Command 5 - 0

Transfer Commands

000001	TX reset
000010	TX enable
000011	TX disable
001000	MP bit on
001001	TX buffer clear

Receive Commands

010001	RX reset
010010	RX enable
010011	RX disable
010110	Search for MP bit

Miscellaneous Commands

100001	Channel reset
000000	No operation

—: Reserved bit. Always read as 0. Cannot be written to.
CMD is not affected by the RESET instruction.

Figure 10-9. Command Register (CMD)

TX Reset: The TX reset command disables the transmitter by setting the TXD pin output to mark "1" and clearing the TXRDY bit of ST0. It clears the TX buffer and BRK bit of CTL and resets the TX status in the ST0-ST3 registers. No other registers are affected.

TX Enable: The TX enable command puts the transmitter into idle state (TXD = 1).

TX Disable: The TX disable command forcibly disables the TXRDY bit of ST0. It then disables the transmitter after it transmits the contents of the TX buffer and shift register.

MP Bit On: In asynchronous mode, the "MP bit on" command sets the MP bit for the next character to be sent to the TX buffer. This command is valid for only one character. If parity is set for the character, the parity is inverted.

TX Buffer Clear: The TX buffer clear command clears the contents of the TX buffer and sets the TXRDY bit of ST0. No other registers are affected.

RX Reset: RX reset stops reception and disables the receiver. It clears the RX buffer and resets the RX status in the ST0-ST3 registers. No other registers are affected.

RX Enable: RX enable puts the transmitter into the start bit detection state. RX enable is ignored if the receiver is enabled.

RX Disable: RX disable stops reception and disables the receiver. The contents of the RX shift register are lost, but the RX buffer is not affected.

Search for MP Bit: Received data will not be loaded into the receive buffer unless its MP bit is set. This command is effective only until the ASCI receives a character with MP = 1.

Channel Reset: Channel reset performs the same function as hardware reset. Channel reset disables the receiver and transmitter, clears the RX and TX buffers, and then initializes all registers.

No Operation: No operation is performed. Transmitter and receiver continue current operations.

10.2.9 Status Register 0 (ST0)

Status register 0 (ST0) indicates whether the TXINT or RXINT interrupts are enabled, and indicates the state of the TX/RX buffers (Figure 10-10). This register is not affected by the RESET instruction.

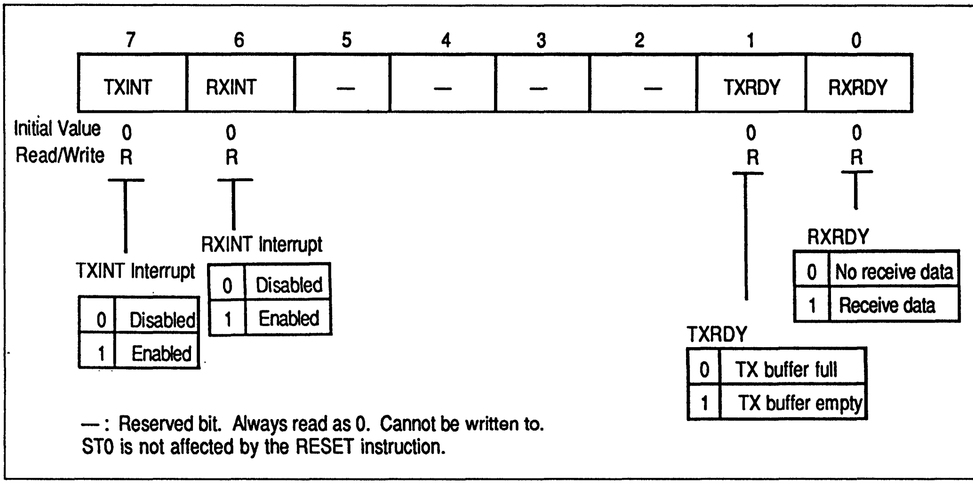


Figure 10-10. Status Register 0 (ST0)

TXINT Interrupt (TXINT): TXINT indicates a transmit interrupt request. TXINT is set by the following conditions:

TXINT = IDL • IDLE + CCTS • CCTSE

Where: IDL = Bit 6 of the ST1 register
CCTS = Bit 3 of the ST1 register
IDLE = Bit 6 of the IE1 register
CCTSE = Bit 3 of the IE1 register

Accordingly, in either the following cases, the TXINT bit is set to 1.

1. The ASCI enters idle state while the IDLE bit is set.
2. The $\overline{CTS}$ pin changes while the CCTSE bit is set.

If TXINTE of IE0 is set when TXINT is set, a TXINT interrupt is requested.

RXINT Interrupt (RXINT): RXINT indicates a receive interrupt request. RXINT is set by the following conditions:

RXINT = CDCD • CDCDE + BRKD • BRKDE + BRKE • BRKEE + PMP • PMPE + PE •PEE + FRME • FRMEE + OVRN • OVRNE

Where: CDCD = Bit 2 of the ST1 register
BRKD = Bit 1 of the ST1 register
BRKE = Bit 0 of the ST1 register
PMP = Bit 6 of the ST2 register
PE = Bit 5 of the ST2 register
FRME = Bit 4 of the ST2 register
OVRN = Bit 3 of the ST2 register
CDCDE = Bit 2 of the IE1 register
BRKDE = Bit 1 of the IE1 register
BRKEE = Bit 0 of the IE1 register
PMPE = Bit 6 of the IE2 register
PEE = Bit 5 of the IE2 register
FRMEE = Bit 4 of the IE2 register
OVRNE = Bit 3 of the IE2 register

Accordingly, the RXINT bit is set in any of the following cases:

1. The $\overline{\text{DCD}}$ pin changes while the CDCDE bit is set to 1.
2. The break start is detected while the BRKDE bit is set to 1.
3. The break end is detected while the BRKEE bit is set to 1.
4. The parity bit, MP bit or MSB bit is set to 1 while the PMPE bit is set to 1.
5. A parity error occurs while the PEE bit is set to 1.
6. A framing error occurs while the FRMEE bit is set to 1.
7. An overrun error occurs while the OVRNE bit is set to 1.

If RXINTE of IE0 is set when RXINT is set, an RXINT interrupt is requested.

TX Ready (TXRDY): TXRDY is set to indicate the TX buffer is ready to be written if the TX buffer is empty during TX enable state. It is cleared when data is sent to the TX buffer, negating the TX ready state.

If TXRDYE of IE0 is set while TXRDY is set, the DMAC requests a TXRDY interrupt to the CPU. See "10.5 Serial Data Transfer by CPU and DMAC" for details on the DMA request.

RX Ready (RXRDY): RXRDY is set if the RX buffer holds data. It is cleared when the RX buffer is empty.

If RXRDYE of IE0 is set while RXRDY is set, the ASCI requests an RXRDY interrupt to the CPU. See "10.5 Serial Data Transfer by CPU and DMAC" for details on the DMA request.

10.2.10 Status Register 1 (ST1)

Status register 1 (ST1) (Figure 10-11) indicates the transmitter state, $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$ pin changes, and break start and completion.

The status register 1 is not affected by the RESET instruction.

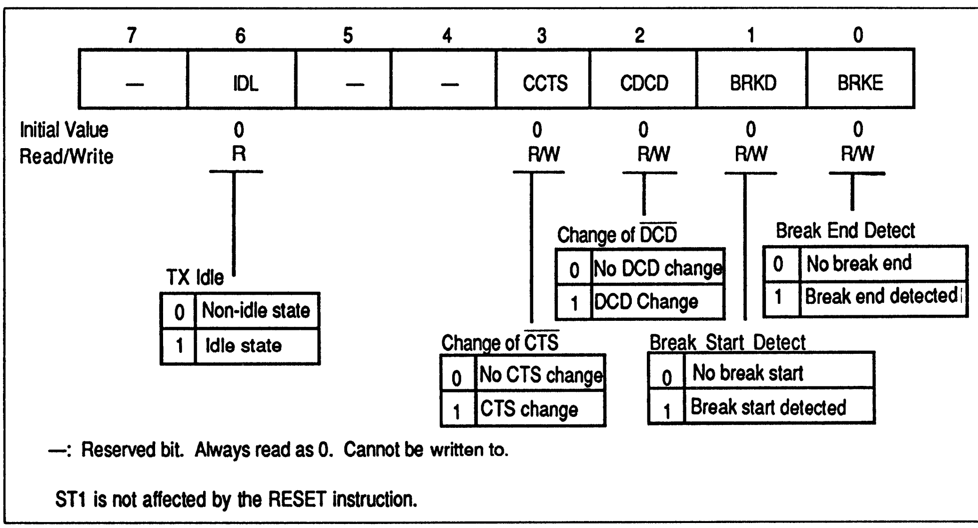


Figure 10-11. Status Register 1 (ST1)

TX Idle (IDL): IDL is set when the transmitter enters to idle state. In asynchronous mode, the TXD pin is set to mark "1", while, in clock synchronous mode, it is set to the MSB of the last character.

IDL is cleared if the transmitter changes from idle state to another state. For example, when transmit data is written to the TRB (TX buffer) in asynchronous mode, the transmitter enters start bit send state and IDL is cleared to 0.

IDL together with the IDLE bit of IE1 can be a TXINT interrupt source.

Change of $\overline{\text{CTS}}$ (CCTS): CCTS is set by a change of state (0 to 1 or 1 to 0) of the $\overline{\text{CTS}}$ input. CCTS can be cleared by writing a 1 to it.

CCTS together with the CCTSE bit of 1E1 can be an TXINT interrupt source.

Change of $\overline{\text{DCD}}$ (CDCD): CDCD is set by a change of state (0 to 1 or 1 to 0) of the $\overline{\text{DCD}}$ input. CDCD can be cleared by writing a 1 to it.

CDCD together with the CDCDE bit of IE1 can be an RXINT interrupt source.

Break Start Detect (BRKD): In asynchronous mode, BRKD is set by break sequence (space state) start. It is cleared by writing a 1 to it. In clock synchronous mode, it is fixed to 0 and always read as 0. See "10.3.9 Break Send/Detection" for details.

BRKD together with the BRKDE bit of IE1 can be an RXINT interrupt source.

Break End Detect (BRKE): In asynchronous mode, BRKE is set by break sequence (space state) end. It is cleared by writing a 1 to it. In clock synchronous mode, it is fixed to 0 and always read as 0. See "10.3.9 Break Send/Detection" for details.

BRKE together with the BRKEE bit of IE1 can be an RXINT interrupt source.

10.2.11 Status Register 2 (ST2)

Status register 2 (ST2) (Figure 10-12) indicates the parity/MP bit status, and the occurrence of parity error, framing error, and overrun. This register is not affected by the RESET instruction.

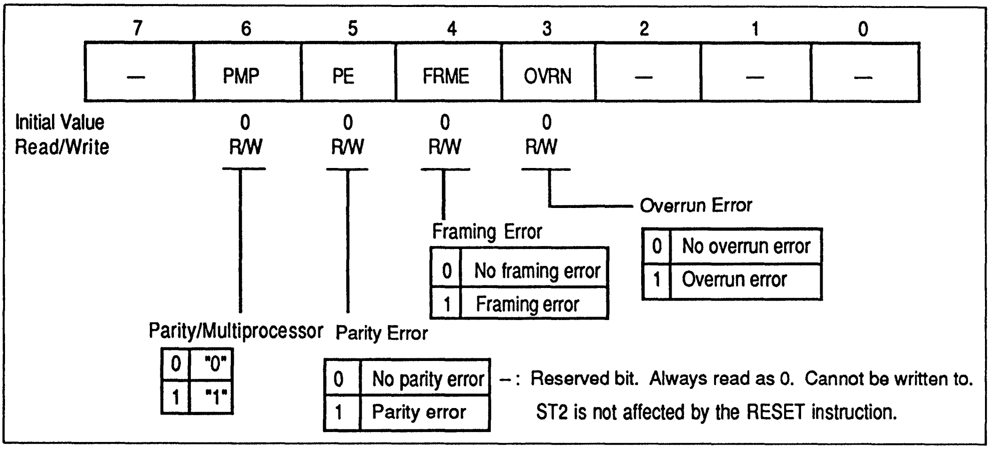


Figure 10-12. Status Register 2 (ST2)

Parity/Multiprocessor Bit (PMP): PMP has the same value as the parity or MP bit of the corresponding received character. Whether the bit indicates parity or MP depends on the PMPM0 and PMPM1 bits of the MD1 register. See Table 10-15.

The PMP bit is modified when the next receive character can be read. It is cleared by writing a 1 to it or by reset.

PMP together with PMPE of the IE2 register can be an RXINT interrupt source.

Table 10-15. PMP1-PMP0, PMP Bits and Parity/MP Relationships

PMPM1	PMPM0	Parity/MP	PMP
0	0	No parity/No MP bit	MSB
0	1	MP bit	MP bit
1	0	Even parity	Parity bit
1	1	Odd parity	Parity bit

Parity Error (PE): PE is set if the receive character has a parity error. It can be cleared by writing a 1 to it or by reset.

PE together with PEE of the IE2 register can be an RXINT interrupt source.

See "10.3.8 Error Check" for details on parity error.

Framing Error (FRME): In asynchronous mode, FRME is set if the receiver detects a framing error, as described in the asynchronous mode section. It can be cleared by writing a 1 to it or by reset. See "10.3.8 Error Check" for details on framing error.

FRME together with FRMEE of the IE2 register can be an RXINT interrupt source.

In clock synchronous mode, it is fixed to 0.

Overrun (OVRN): OVRN is set by a receive overrun. OVRN can be cleared by writing a 1 to it or by reset. See "10.3.8 Error Check" for details on overrun error.

OVRN together with OVRNE of the IE2 register can be an RXINT interrupt source.

10.2.12 Status Register 3 (ST3)

The bits of status register 3 (Figure 10-13) indicate the status of $\overline{\text{DCD}}$, $\overline{\text{CTS}}$, transmitter, and receiver. This register is not affected by the RESET instruction.

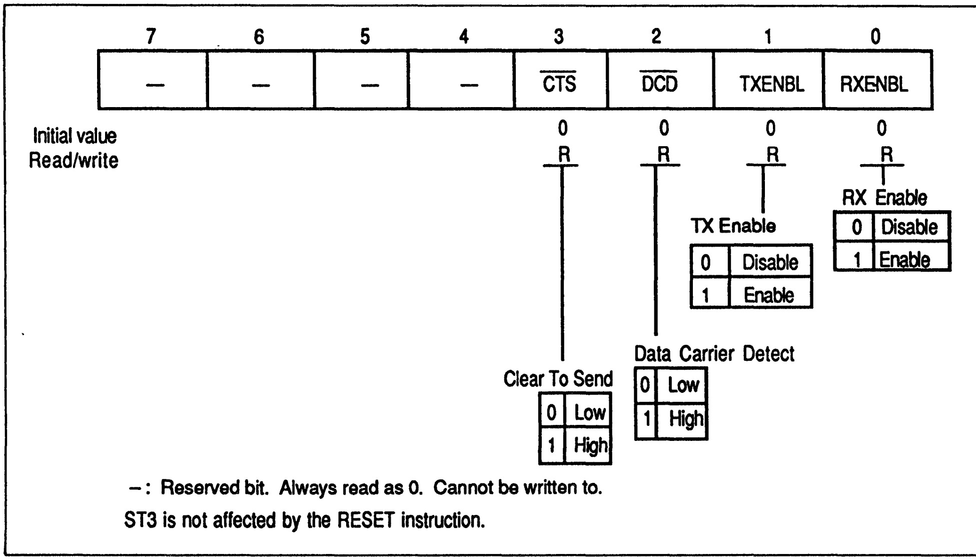


Figure 10-13. Status Register 3 (ST3)

Clear to Send ($\overline{\text{CTS}}$): $\overline{\text{CTS}}$ copies the state of the $\overline{\text{CTS}}$ input. It is a read-only bit and is not affected by a write.

Data Carrier Detect ($\overline{\text{DCD}}$): $\overline{\text{DCD}}$ copies the state of the $\overline{\text{DCD}}$ input. It is a read-only bit and is not affected by a write.

TX Enable (TXENBL): TXENBL is set when the transmitter is enabled. It is cleared when the transmitter is disabled. It is a read-only bit and is not affected by a write.

Note that the transmitter is disabled or enabled by commands. See "10.2.8 Command Register" for details on these commands.

RX Enable (RXENBL): RXENBL is set when the receiver is enabled. It is cleared when the receiver is disabled. It is a read-only bit and is not affected by a write.

Note that the receiver is disabled or enabled by commands. See "10.2.8 Command Register" for details on these commands.

10.2.13 Interrupt Enable Register 0 (IE0)

Interrupt enable register 0 (Figure 10-14) indicates enable/disable of TXINT, RXINT, TXRDY, and RXRDY interrupts. This register is not affected by the RESET instruction.

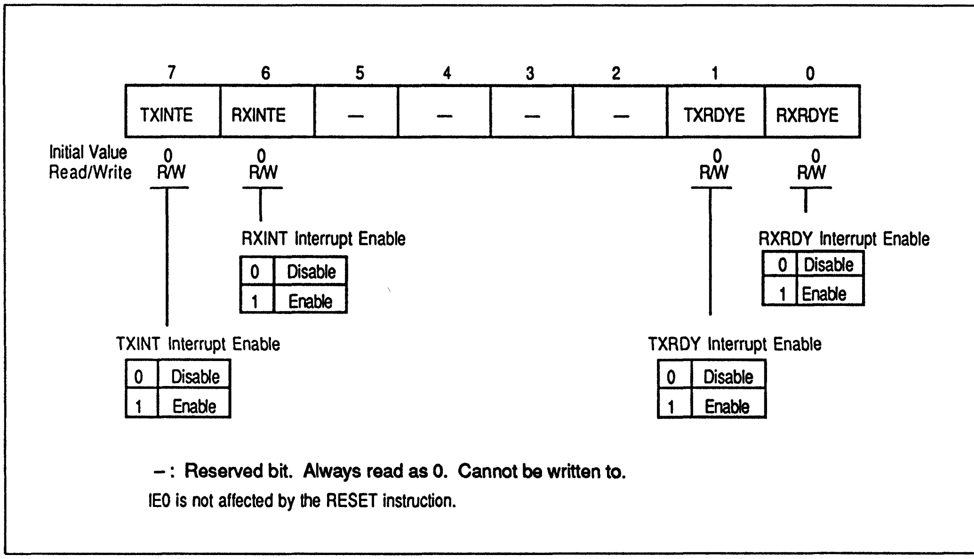


Figure 10-14. Interrupt Enable Register 0 (IE0)

TXINT Interrupt Enable (TXINTE): TXINTE = 1 enables a TXINT interrupt. If a TXINT interrupt occurs while TXINTE = 1, the ASCI requests a TXINT interrupt to the CPU. TXINTE = 0 disables a TXINT interrupt. See "10.6 ASCI Interrupts" for details.

RXINT Interrupt Enable (RXINTE): RXINTE = 1 enables an RXINT interrupt. If an RXINT interrupt occurs while RXINTE = 1, the ASCI requests an RXINT interrupt to the CPU. RXINTE = 0 disables an RXINT interrupt. See "10.6 ASCI Interrupts" for details.

TXRDY Interrupt Enable (TXRDYE): TXRDYE = 1 enables a TXRDY interrupt. If a TXRDY interrupt occurs while TXRDYE = 1, the ASCI requests a TXRDY interrupt to the CPU. TXRDYE = 0 disables a TXINT interrupt. See "10.6 ASCI Interrupts" for details.

RXRDY Interrupt Enable (RXRDYE): RXRDYE = 1 enables an RXRDY interrupt. If an RXRDY interrupt occurs while RXRDYE = 1, the ASCI requests an RXRDY interrupt to the CPU. RXRDYE = 0 disables an RXINT interrupt. See "10.6 ASCI Interrupts" for details.

10.2.14 Interrupt Enable Register 1 (IE1)

Interrupt enable register 1 (Figure 10-15) indicates whether TXINT and RXINT interrupts will be generated. This register is not affected by the RESET instruction.

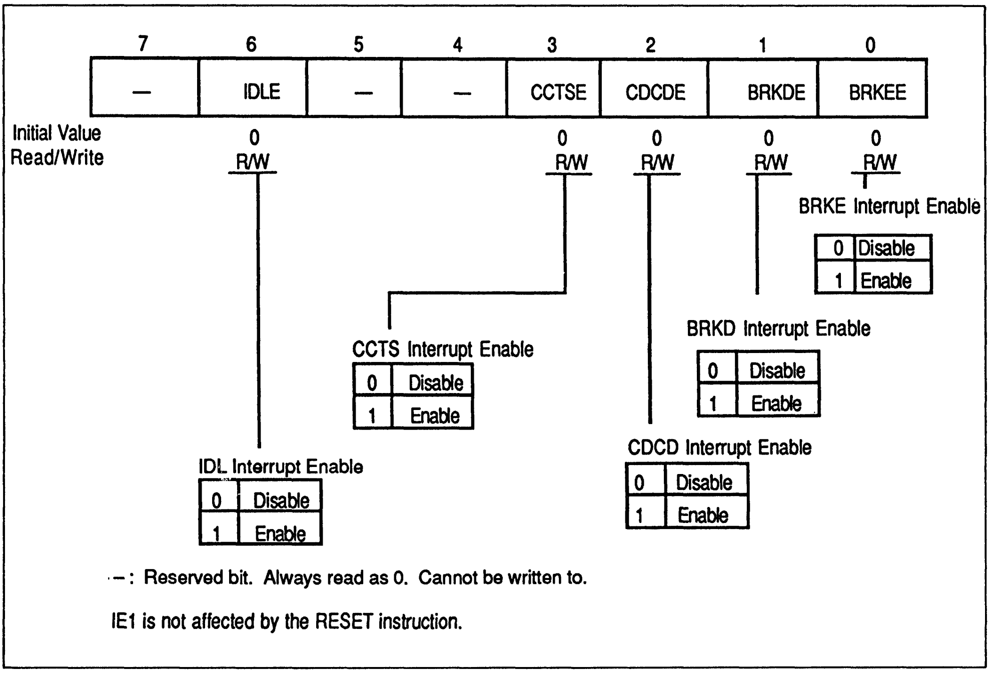


Figure 10-15. Interrupt Enable Register 1 (IE1)

IDL Interrupt Enable (IDLE): IDLE = 1 enables the IDL bit of the ST1 register. If IDL is set while IDLE = 1, the TXINT bit of the ST0 register is set. IDLE = 0 disables IDL.

CCTS Interrupt Enable (CCTSE): CCTSE = 1 enables the CCTS bit of the ST1 register. If CCTS is set while CCTSE = 1, the TXINT bit of the ST0 register is set. CCTSE = 0 disables CCTS.

CDCD Interrupt Enable (CDCDE): CDCDE = 1 enables the CDCD bit of the ST1 register. If CDCD is set while CDCDE = 1, the RXINT bit of the ST0 register is set. CDCDE = 0 disables CDCD.

BRKD Interrupt Enable (BRKDE): BRKDE = 1 enables the BRKD bit of the ST1 register. If BRKD is set while BRKDE = 1, the RXINT bit of the ST0 register is set. BRKDE = 0 disables BRKD.

BRKE Interrupt Enable (BRKEE): BRKEE = 1 enables the BRKE bit of the ST1 register. If BRKE is set while BRKEE = 1, the RXINT bit of the ST0 register is set. BRKEE = 0 disables BRKE.

10.2.15 Interrupt Enable Register 2 (IE2)

Interrupt enable register 2 (Figure 10-16) indicates whether or not RXINT interrupts are enabled. This register is not affected by the RESET instruction.

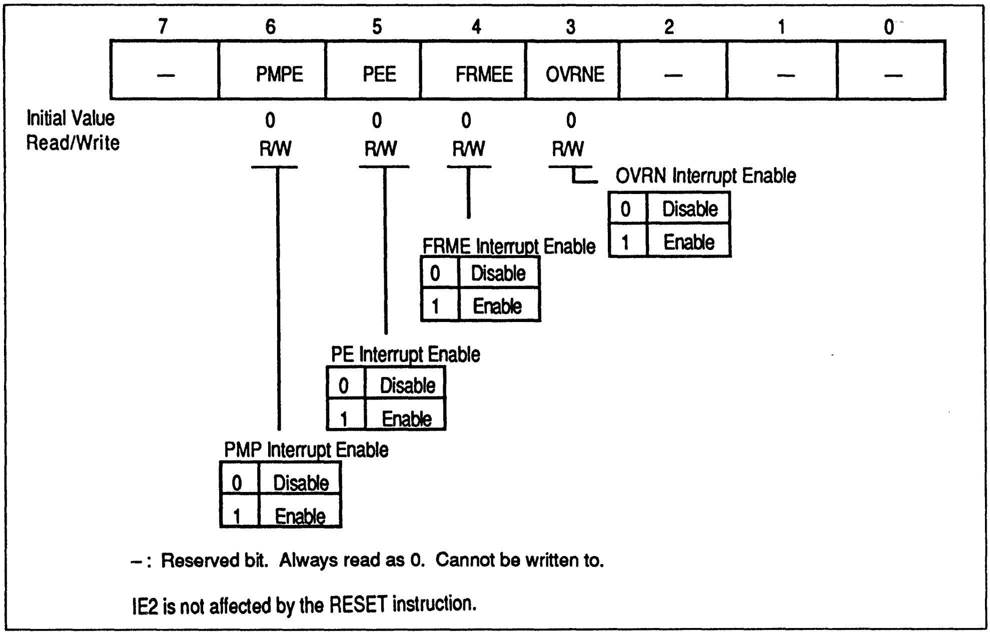


Figure 10-16. Interrupt Enable Register 2 (IE2)

PMP Interrupt Enable (PMPE): PMPE = 1 enables the PMP bit of the ST2 register. If PMP is set while PMPE = 1, the RXINT bit of the ST0 register is set. PMPE = 0 disables PMP.

PE Interrupt Enable (PEE): PEE = 1 enables the PE bit of the ST2 register. If PE is set while PEE = 1, the RXINT bit of the ST0 register is set. PEE = 0 disables PE.

FRME Interrupt Enable (FRMEE): FRMEE = 1 enables the FRME bit of the ST2 register. If FRME is set while FRMEE = 1, the RXINT bit of the ST0 register is set. FRMEE = 0 disables FRME.

OVRN Interrupt Enable (OVRNE): OVRNE = 1 enables the OVRN bit of the ST2 register. If OVRN is set while OVRNE = 1, the RXINT bit of the ST0 register is set. OVRNE = 0 disables OVRN.

10.2.16 TX/RX Buffer Register (TRB)

Reading the TX/RX buffer register (TRB) (Figure 10-17) reads a byte from the RX buffer. Writing to the TRB register puts the data into the TX buffer.

If the TRB is read while RXRDY = 0, the TRB contents are undefined. While TXRDY = 0, TRB cannot be written to.

This register is not affected by the RESET instruction.

	7	6	5	4	3	2	1	0
	TRB 7	TRB 6	TRB 5	TRB 4	TRB 3	TRB 2	TRB 1	TRB 0
Initial Value	*	*	*	*	*	*	*	*
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TX / RX Buffer 7—0

Write	Transmit character is written to the TX buffer.
Read	Receive character is read from the RX buffer.

*: undefined

TRB is not affected by the RESET instruction.

Figure 10-17. TX/RX Buffer Register (TRB)

10.3 ASCI Asynchronous Mode Operation

In asynchronous mode, the ASCI transfers characters synchronized with the start and stop bits which are included with each character. In this mode, the TXD and RXD pins are pulled high (mark) when characters are not being transferred. Accordingly, TXD and RXD pins of low (space) indicate the beginning of character transfer, i.e., a start bit encountered.

Asynchronous mode can be set by clearing the PRTCL2-PRTCL bits of the MD0 register.

10.3.1 Character Format

Figure 10-18 shows an example of asynchronous mode character format. A character transfer begins with the 1-bit start bit. Next, 7- or 8-bit data is transferred in order of the LSB first and the MSB last. A parity or MP bit may then be transferred depending on the PMPM1-PMPM0 bits of MD1. Finally, 1 or 2 stop bits are transferred.

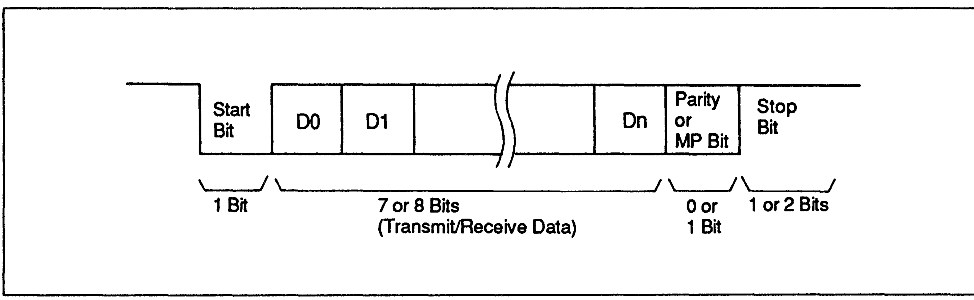


Figure 10-18. Asynchronous Mode Character Format

10.3.2 Baud Rate

The ASCI can operate at a baud rate of 1/1, 1/16, 1/32, 1/64 of an input clock provided from an external pin or the baud rate generator (Figure 10-19). Bit rates for both receiver and transmitter can be specified as 1/1, 1/16, 1/32, or 1/64 clock rate by the BRATE1-BRATE0 bits of the MD1 register. See "10.7 Baud Rate Generator" for details.

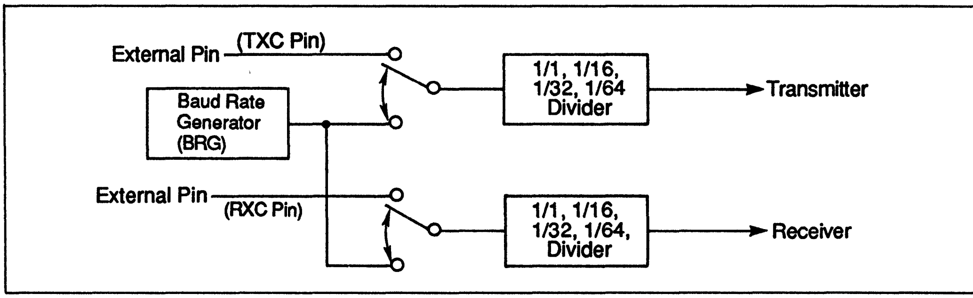


Figure 10-19. Baud Rate Selection Circuit

10.3.3 Transmission State

Figure 10-20 is a transition state diagram for asynchronous transmission.

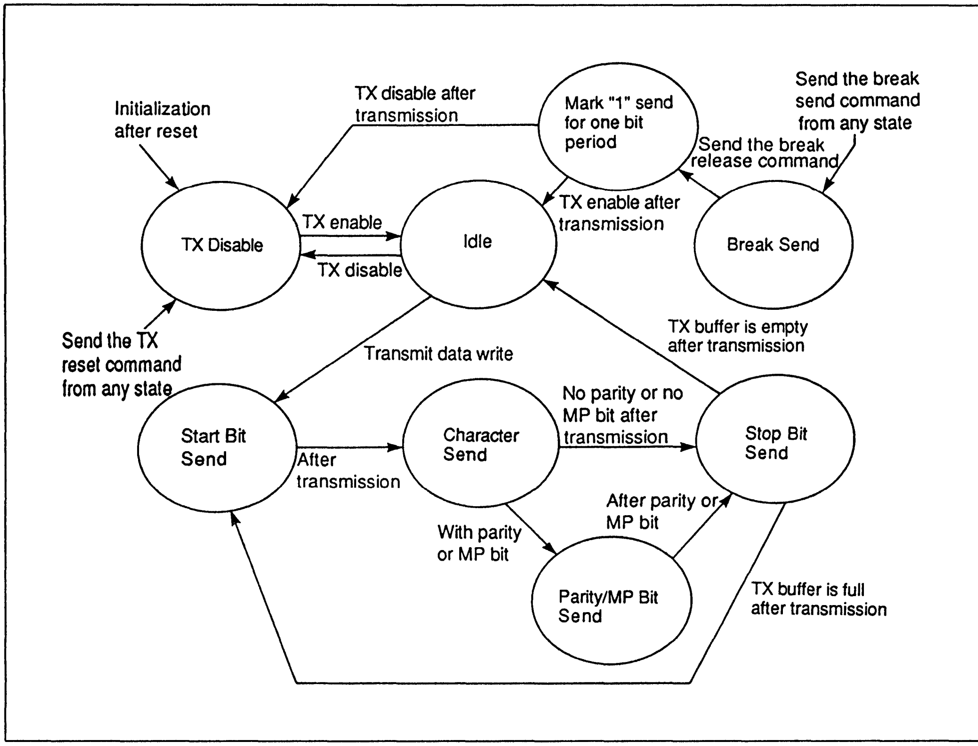


Figure 10-20. Transmission State Transition Diagram

TX Disable: The transmitter enters TX disable state by reset, TX reset command, or TX disable command. In this state, the TXD pin outputs mark "1" and the TXRDY bit of ST0 is cleared.

Idle State: TXD pin outputs mark "1" when the TX buffer is empty. The ASCI goes from idle state to start bit send state if a transmit character is written.

Start Bit Send: The ASCI sends space "0" for 1 bit period through the TXD pin and enters the character send state.

Character Send: The ASCI transmits a character LSB first, MSB last.

Parity/MP Bit Send: The ASCI sends parity or MP bit depending on the PMPM1-PMPM0 bits of MD1. See "10.3.7 Parity/MP Bit" for details.

Stop Bit Send: The TXD pin outputs mark "1" for the number of bit periods determined by the STOP1-STOP0 bits of the MD0 register. The ASCI then enters idle state.

Break Send: The TXD pin outputs space "0". Setting the BRK bit of CTL to 1 sends a break. This can be stopped by clearing the BRK bit to 0. See "10.3.9 Break Send/Detection" for details.

Mark Send for 1 Bit Period: The TXD pin sends mark "1" for 1 bit period after break send state is cancelled.

10.3.4 Transmit Operation

TXD output changes at the falling edge of TXC clock (Figure 10-21). The ASCI transmit operation is initiated by writing a character to the TX buffer in idle state. Figure 10-21 shows an example of transmit operation when a character consists of 8-bit data, a parity bit and 1 stop bit.

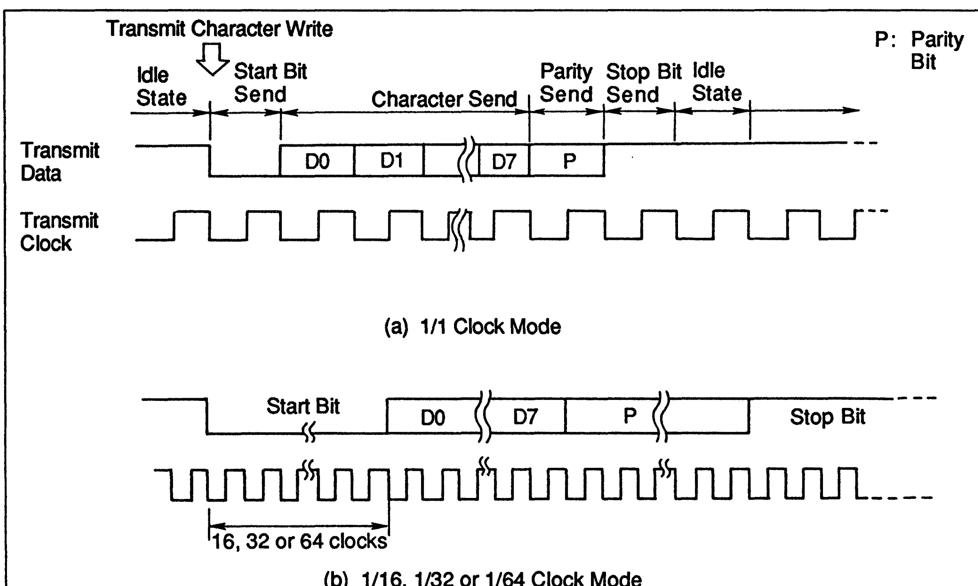


Figure 10-21. Asynchronous Transmission

10.3.5 Receive State

Figure 10-22 shows the receive state transition diagram.

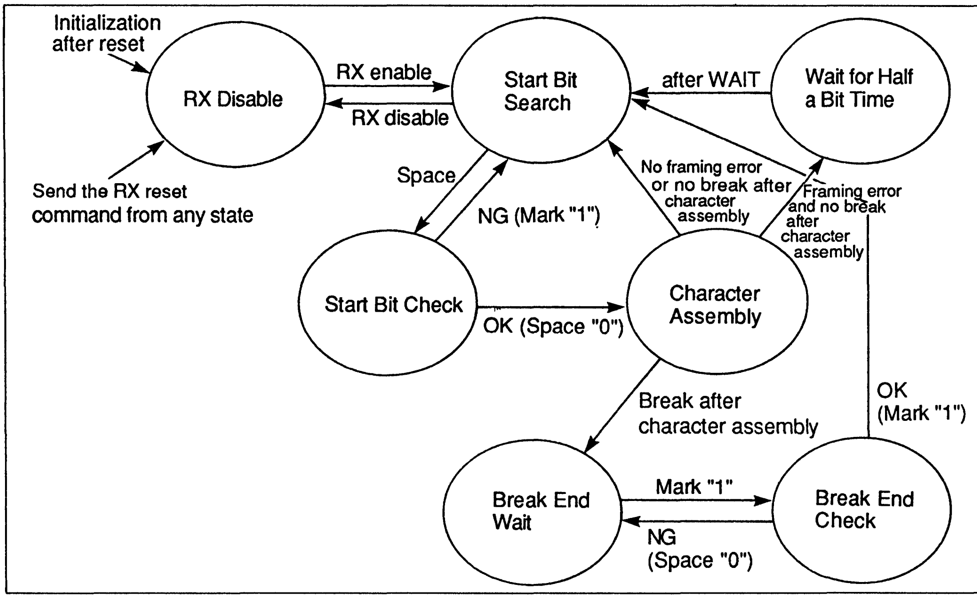


Figure 10-22. Receive State Transition Diagram

RX Disable: The ASCI receiver is disabled by reset, the RX reset command, or RX disable command. In this state, the RXD input is ignored. The contents of the receive shift register is lost, while the contents of the RX buffer is maintained.

Start Bit Search: The ASCI goes from RX disable state to start bit search state by the RX enable command. The ASCI checks the RXD pin at each rising edge of the RXC clock to search for the start bit (space: "0").

Start Bit Check: The ASCI enters start bit check state if it detects space "0" in start bit search state. The ASCI then checks the RXD pin again one-half bit period later. At this time, if RXD is not 0, the ASCI returns to start bit search state. If RXD is 0, the ASCI enters character assembly state. However, in 1/1 clock mode, note that the ASCI does not check the start bit, but assembles the character immediately.

Character Assembly: The ASCI assembles characters by sampling data received through the RXD pin at every bit period. The character assembly ends when a stop bit is sampled.

Wait for Half a Bit Period: The ASCI waits for one half a bit period after character assembly to skip the stop bit of a character with a framing error. It then returns to start bit search state. See "10.3.8 Error Check" for further details.

Break End Wait: The ASCI enters break end wait state if it detects break after character assembly. The ASCI checks the RXD pin at each rising edge of the RXC clock to search for mark "1". See "10.3.9 Break Send/Detection" for details.

Break End Check: If the ASCI detects mark "1" on the RXD pin in break end wait state, it enters break end check state. It checks RXD again one half a bit period later. If RXD does output mark "1", the ASCI searches for the start bit. Otherwise, it returns to break end wait state.

10.3.6 Receive Operation

Figure 10-23 illustrates the data sampling timing.

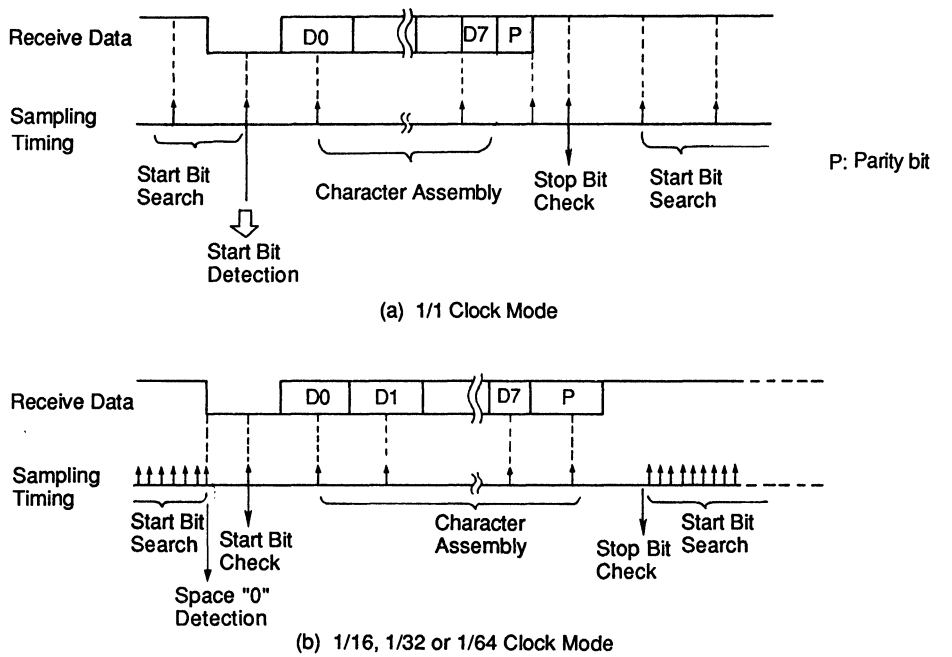


Figure 10-23. Receive Data Sampling Timing

Figure 10-24 shows the ASCI receive character format in asynchronous mode. 8-bit or 7-bit characters can be received depending on the RXCHR1-RXCHR0 bits of the MD1 register. In 7-bit character format, the MSB contains 0.

8 bits/character	D7	D6	D5	D4	D3	D2	D1	D0
7 bits/character	0	D6	D5	D4	D3	D2	D1	D0

Figure 10-24. Receive Character Format

The ASCI receive operation is initiated by sending RX enable command.

In 1/1 clock mode, the receiver samples data at the rising edge of RXC clock. If it detects a space "0" on the RXD pin during start bit search, the ASCI begins character assembly at the rising edge of the next RXC clock. During character assembly, the ASCI assembles a character by sampling 1-bit data every RXC clock and loading them into the RX shift register (Figure 10-25). After the RX shift register receives 7 or 8 bits (depending on the RXCH1 and RXCH0 bits of the MD1 register), the ASCI samples the parity or MP bit depending on the PMPM bits of MD1. The ASCI then samples the stop bit at the rising edge of the next RXC clock and completes character assembly. At this time, the contents of the RX shift register are loaded into the RX buffer.

The ASCI begins to search for a start bit at the rising edge of the next RXC clock after it detects the stop bit.

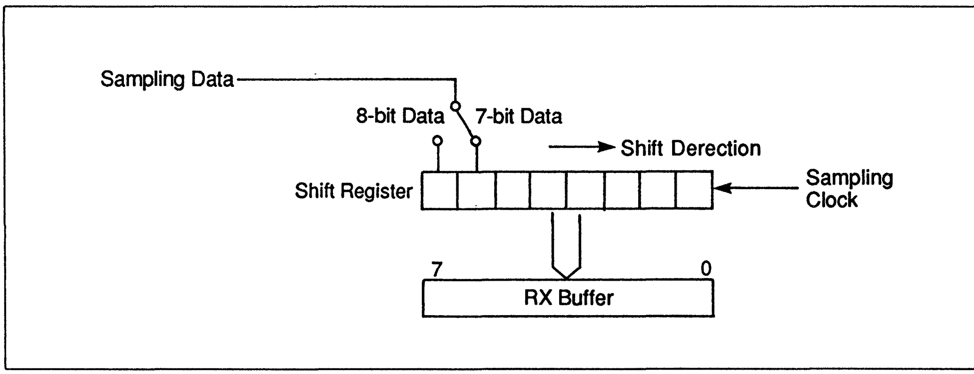


Figure 10-25. Character Assembly by Receive Shift Register

In 1/16, 1/32, or 1/64 clock mode, when the bit rate is 1/16, 1/32, or 1/64 of clock, the receiver samples data at every rising edge of the RXC clock during start bit search. When the ASCI detects space "0" on the RXD pin, it checks RXD again one-half a bit period later. If the ASCI detects space "0" again, it begins character assembly with a one-bit-period delay. If the ASCI detects mark "1", it interprets the transition of the RXD pin as noise and continues searching for a start bit. See Figure 10-26.

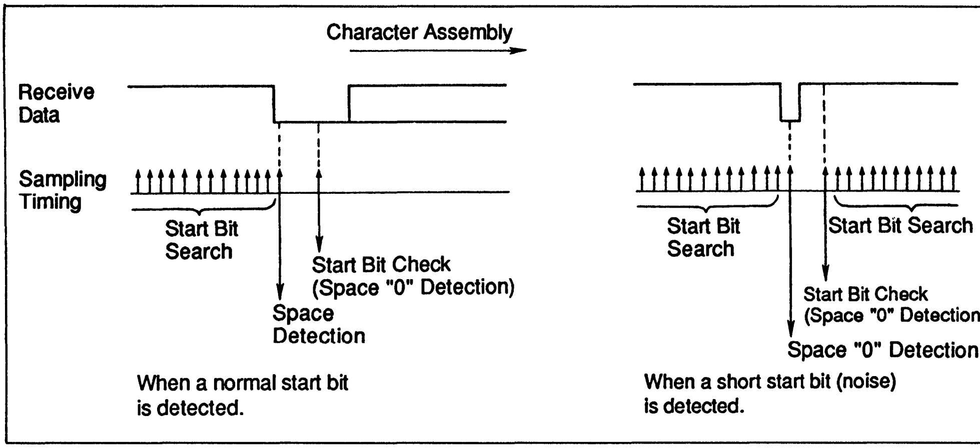


Figure 10-26. Start Bit Sampling Timing

The ASCI assembles characters by sampling data each bit period.

The ASCI checks for a stop bit 1-bit period after it receives the MSB or parity. At this time, if the ASCI receives mark "1", it immediately begins to search for a start bit. If it receives space "0", it begins searching for a start bit after a one-half bit period delay.

Note that in 1/16, 1/32, or 1/64 clock mode, the ASCI samples data received through the RXD pin three times starting from the two RXC clock cycles prior to sampling timing, and determines the data value by majority consideration. This function protects the received data from noise. See Figure 10-27.

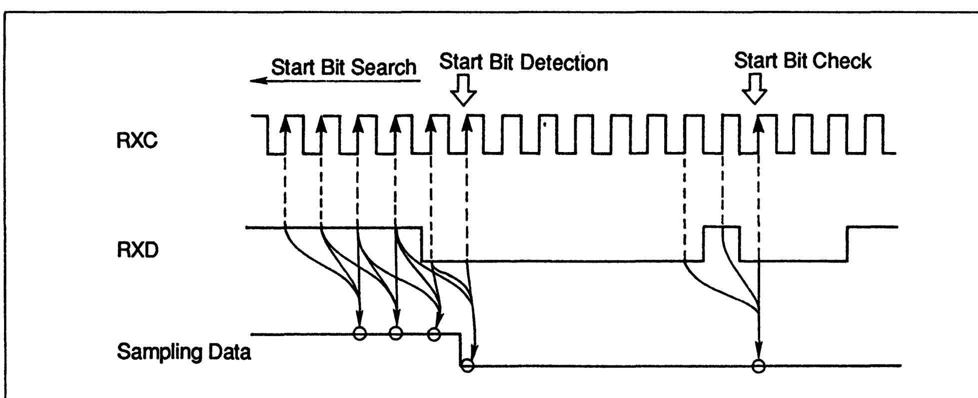


Figure 10-27. Noise Elimination in 1/16, 1/32, or 1/64 Clock Mode

10.3.7 Parity/MP Bit

Even parity, odd parity, MP bit, or no parity/MP bit can be selected by the PMPM1 and PMPM0 bits of the MD1 register.

When even parity is selected, the transmitter checks the number of 1s in the transmitted character. If the number of 1s is even, the transmitter sends 0 following the character. Otherwise, the transmitter sends 1 following the character. Accordingly, the parity bit value is determined in order to force the number of 1s in the character to be even. In addition, the receiver also checks whether or not the number of 1s in the received character is even, simultaneously. When odd parity is selected, the ASCI transmitter and receiver function in the same way as in even parity, but forces the number of 1s to be odd.

When the MP bit is selected, the transmitter or receiver sends or receives the MP bit following the character to support multiprocessor communications. See "10.3.10 Multiprocessor Support" for details.

10.3.8 Error Check

Parity Check: Odd, even, or no parity is checked under software control. If a parity error occurs in a received character, the ASCI sets the PE bit of the ST2 register.

When a parity error occurs, sub equent characters can be received normally. However, once the PE bit is set it will not be cleared even if a parity error does not occur in the next character. The PE bit can be cleared by writing a 1 to the bit or by reset.

Framing Error: The ASCI detects a framing error when it samples space "0" during stop bit check. Note that only the first stop bit of 2 stop bits is checked. If a framing error occurs, the FRME bit of the ST2 register is set.

The ASCI can continue the receive operation after detecting a framing error. Note that the ASCI begins to search for a start bit after a one-half bit period delay in 1/16, 1/32, or 1/64 clock mode and from the next rising edge of the clock in 1/1 clock mode. This avoids the interpretation of a framing error as a new start bit.

Once the FRME bit is set, it cannot be cleared even if a framing error does not occur in the next character. The FRME bit can be cleared by writing a 1 to the bit or by reset.

Overrun Error: The ASCI detects an overrun error if the next receive character is transferred from the RX shift register to the RX buffer register before the previous receive character in the RX buffer is read. At this time, the OVRN bit of the ST2 register is set to 1.

Once the OVRN bit is set, it is not cleared until it is cleared by writing a 1 to it or by reset.

10.3.9 Break Send/Detection

The BRK bit of the CTL register controls the ASCI break sending and detection as described below.

Break Send Sequence: When the CPU requests break output, the TXD pin immediately outputs space "0" at the falling edge of TXC immediately after the BRK bit is set to 1.

The TXD pin outputs mark "1" at the falling edge of TXC immediately after the BRK bit is changed from 0 to 1, and it is held in the mark condition for at least one bit period.

During break output, the data in the TX shift register is lost.

The procedure for sending a normal break is as follows:

1. Wait for end of transmission (idle state).
2. Write a 1 to the BRK bit.
3. Wait for at least 2 character periods.
4. Write a 0 to the BRK bit.

Break Detection: The ASCI detects the start of break when it receives a character with data and parity bit all 0s, and with a framing error as well.

The ASCI detects the end of break when it detects mark "1" on the RXD pin for at least one-half a bit period. Note that the ASCI detects end of break immediately after it detects mark "1" for 1/1 clock mode.

RXINT interrupt can be generated on both start and end of break output since the BRKD bit and BRKE bit are set on break output start detection and break output end detection, respectively.

When the start of break is detected, the NULL character (all bits are 0s) with framing error is aborted and not stored in the RX buffer. At this time, the FRME bit of the ST2 register is not set to 1.

Figure 10-28 shows break start/end detection timing when the break starts in the middle of the character transmission. Note that the transmitter must maintain the break level for at least two character periods to be received correctly.

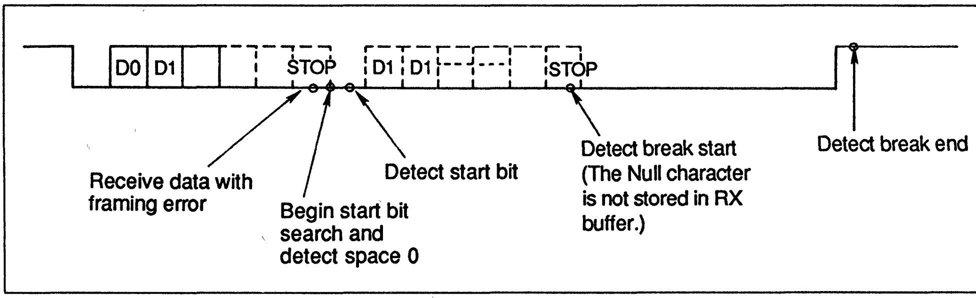


Figure 10-28. Break During Character Transmission

10.3.10 Multiprocessor Support

The ASCI asynchronous mode can support multiprocessor mode where the characters have an MP bit instead of a parity bit.

Transmission: The MP bit of transmit character is normally 0. It can be set to 1 by the "MP bit on" command. Note that the "MP bit on" command is effective only for one character transmitted following the command.

Receive: The MP bit status of the corresponding character is stored in the PMP bit of the ST2 register.

The ASCI can abort characters with MP bit = 0 by sending the search for MP bit command to the CMD register. This command is effective until the ASCI receives a character with MP bit = 1.

Figure 10-29 shows a multiprocessor communication example.

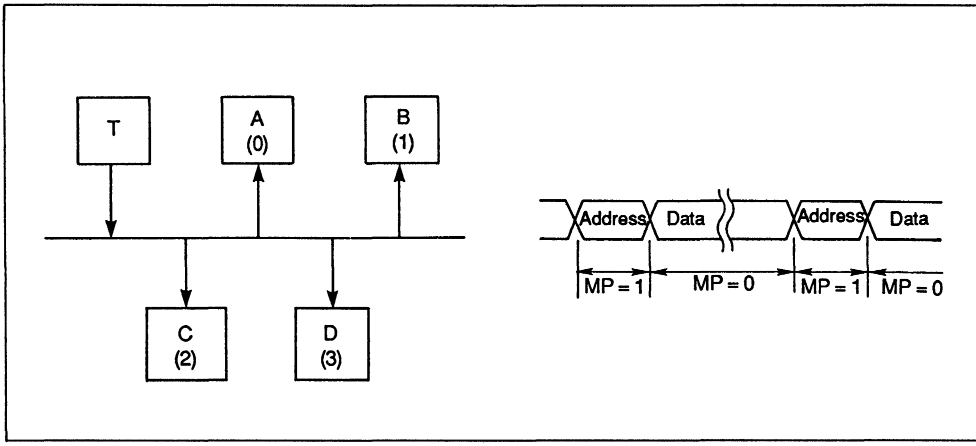


Figure 10-29. Multiprocessor Communication by MP Bit

T is a transmitter and A, B, C, and D are receivers addressed as 0, 1, 2, and 3, respectively.

When transmitter T transmits data to receiver B, T first transfers address 1 of B on the communication line after the MP bit is set to 1. At this time, B checks the communication line. If B receives a character with MP bit = 1, it regards the character as an address and compares the address with its own address. If it matches, B receives the next characters with MP bit = 0. During the communication between T and B, other receivers A, C, and D execute the search for MP bit command and ignore characters with MP bit = 0. Consequently, the transmitter T can communicate with a specific receiver by sending an address with MP = 1 followed by characters with MP bit = 0.

The transmitter T can change receivers by sending a new address with MP bit = 1. If it does, the search for MP bit command is cancelled and a new communication sequence begins.

10.4 ASCI Clock Synchronous Mode Operation

In clock synchronous mode, the ASCI transfers data synchronous with the clock. Clock synchronous mode can be selected by setting the PRTCL2-PRTCL0 bits of the MD0 register to 110.

10.4.1 Character Format

Figure 10-30 shows character format in clock synchronous mode. In clock synchronous mode, data always changes at the falling edge of the synchronous clock. 7- or 8-bit character length and parity/MP bit can be selected and checked in the same way as in asynchronous mode. However, there is no start bit or stop bit in this mode.

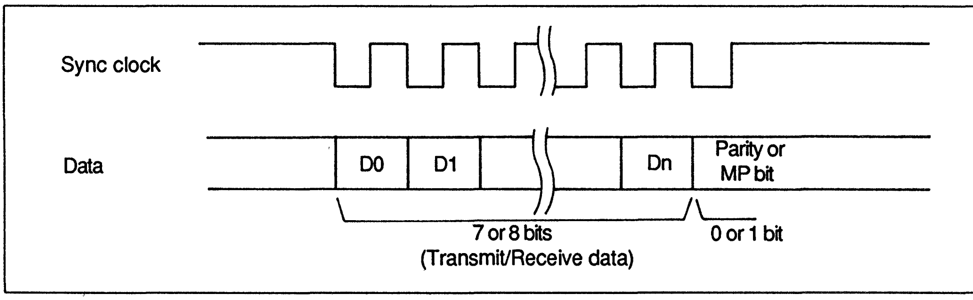


Figure 10-30. Clock Synchronous Mode Character Format

Table 10-16 summarizes ASCI error checking in each protocol mode.

Table 10-16. ASCI Error Check

	ASCI Error			
Mode	**Parity**	**MP**	**Framing Error**	**Overrun Error**
Asynchronous	O	O	O	O
Clock synchronous	O	O	X	O

Note: O = Available
X = Not available

10.4.2 ASCI Clock Synchronous Mode Operation

In clock synchronous mode, the ASCI provides two modes: master and slave. The transmitter's master or slave mode can be selected by the TXCS2-TXCS0 bits of the TXS register. The receiver's master or slave mode can be selected by the RXCS2-RXCS0 bits of the RXS register.

Clock mode in clock synchronous mode is always 1/1. The master outputs the clock through the TXC and RXC pins. The clock level is usually high. However, the low clock pulses generated by the master correspond to bits being transferred. The slave operates synchronously with the clocks input through the TXC and RXC pins.

Data changes at the falling edge of the TXC clock and is received at the rising edge of the TXC clock.

The following describes ASCI clock synchronous mode operation in slave and master modes.

Slave Mode: ASCI slave mode is specified by selecting an external clock as clock source. The TXC and RXC pins are inputs. The ASCI transmits data through the TXD pin synchronously with the external clock input through the TXC pin. After data transmission, the TXD pin retains the last bit and the TXC clock is ignored.

The ASCI receives data through the RXD pin synchronously with external clock input through the RXC pin.

Master Mode: ASCI master mode is specified by selecting built-in BRG output as the clock source. The TXC and RXC pins are outputs. The ASCI transmits data through the TXD pin and outputs the clock generated by the built-in BRG.

The ASCI receives data through the RXD pin synchronously with the TXC clock. Therefore, even when the ASCI performs only receive operations, the transmitter must be enabled to perform dummy transmission.

Recovery from Character Distortion: Receiver and transmitter must be reset by RX reset and TX reset commands respectively after character distortion.

Operation: Figure 10-31 shows the ASCI clock synchronous mode operation. Figure 10-31 (a) shows master to slave data transmission. Data transmission is initiated when the master writes data to the TX buffer. The transmit data changes at the falling edge of the TXC clock. The slave samples data at the rising edge of the TXC clock.

Figure 10-31 (b) shows slave to master data transmission. Data to be transmitted is written to the TX buffer in the slave. The slave then transmits data synchronously with the TXC clock provided by the master. Accordingly, the master must transmit dummy data to provide a TXC clock. The master then samples data at the falling edge of the TXC clock. At this time, the master outputs the same clock from both TXC and RXC pins.

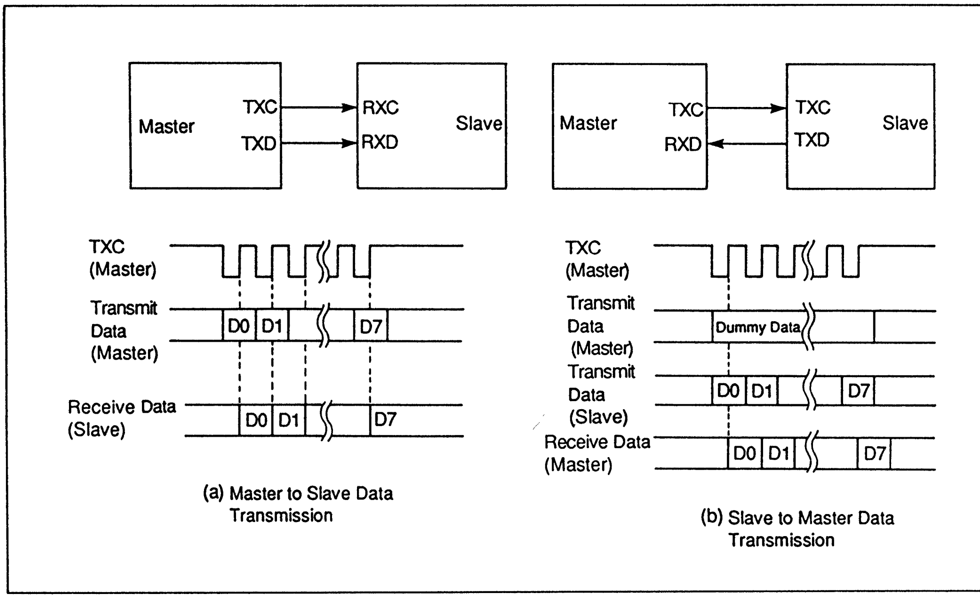


Figure 10-31. ASCI Clock Synchronous Mode Operation

Table 10-17 shows master and slave combinations in the ASCI transmitter and receiver. If the ASCI transmitter and receiver are specified as slave and master respectively, the ASCI may malfunction.

Table 10-17. Slave/Master Combinations

Transmitter	Receiver	Allowed/Not allowed
Slave	Slave	Allowed
Slave	Master	Not allowed
Master	Slave	Allowed
Master	Master	Allowed

10.5 Serial Data Transfer by CPU and DMAC

The ASCI can transmit and receive data under CPU polling or interrupt control, or DMA transfers.

10.5.1 Polling

The CPU obtains data transmission or reception timing by checking the TXRDY or RXRDY bits of the ST0 register. TXRDY and RXRDY interrupt should be disabled, and the DMAC should be programmed so as not to respond to DMA requests by the ASCI.

10.5.2 Interrupt

If the TXINTE or RXINTE bit is 1, a TXRDY or RXRDY interrupt occurs when the TXRDY or RXRDY bit is set, signaling the CPU to transfer data to the TX buffer or from the RX buffer. The DMAC should be programmed so as not to respond to DMA request by the ASCI.

10.5.3 DMA Transfer

The internal DMA signal becomes active when the TXRDY or RXRDY bit is set, causing the DMAC to transfer data to the TX buffer or from the RX buffer. At this time, the TXRDY or RXRDY interrupt must be disabled. Table 10-18 shows DMA transfer conditions.

Table 10-18. DMA Transfer Conditions

Item	Program
Address mode	Dual address
Device address register	TRB of the ASCI
Request signal sense	Level
Bus mode	Cycle steal mode

10.6 ASCI Interrupts

Table 10-19 shows the ASCI interrupt sources. Figure 10-32 shows a block diagram of the ASCI interrupts.

Table 10-19. ASCI Interrupt Sources

Interrupt	Status Bit	Enable Bit	Interrupt Source	Source Status Bit	Source Enable Bit	(Note 1) Clear Condition
RXRDY (RX Ready)	RXRDY	RXRDYE	RX Ready	-	-	RX buffer empty
TXRDY (TX Ready)	TXRDY	TXRDYE	TX Ready	-	-	TX buffer full or TX disabled
RXINT (RX Interrupt)	RXINT	RXINTE	$\overline{DCD}$ change	CDCD	CDCDE	1 is written to the corresponding source status bit
			Break start detect	BRKD	BRKDE	
			Break end detect	BRKE	BRKEE	
			Parity/MP bit set	PMP (Note 2)	PMPE	
			Parity error	PE	PEE	
			Framing error	FRME	FRMEE	
			Overrun error	OVRN	OVRNE	
TXINT (TX Interrupt)	TXINT	TXINTE	TX idle	IDL	IDLE	Transmitter state goes to a state other than TX idle state
			$\overline{CTS}$ change	CCTS	CCTSE	1 is written to interrupt status bit

Notes: 1. RXRDY and RXINT interrupts are cleared by either the channel reset command or RX reset command. TXRDY and TXINT interrupts are cleared by either the channel reset or TX reset command.

2. The PMP bit can be cleared when the next receive character can be read. (It is cleared when RXRDY is set to 1 after the receive character has read.)

Each interrupt shown in Table 10-20 occurs under the conditions shown. Each interrupt can be enabled or disabled by the corresponding interrupt source status or enable bit since interrupt source bit values are always reflected in the RXINT or TXINT bit of the ST0 register regardless of the RXINTE or TXINTE bit of the IE0 register (Figure 10-32). See "4.7 Interrupt Controller" for details on interrupt priority.

Table 10-20. Interrupt Enable Conditions

	Condition	
Interrupt	**Interrupt Status/Enable Bits**	**Interrupt Source Status/Enable Bits**
RXRDY interrupt	RXRDY • RXRDYE	-
TXRDY interrupt	TXRDY • TXRDYE	-
RXINT interrupt	RXINT • RXINTE	RXINT = CDCD • CDCDE + BRKD • BRKDE + BRKE • BRKEE + PMP • PMPE + PE • PEE + FRME • FRMEE + OVRN • OVRNE
TXINT interrupt	TXINT • TXINTE	TXINT = IDL • IDLE + CCTS • CCTSE

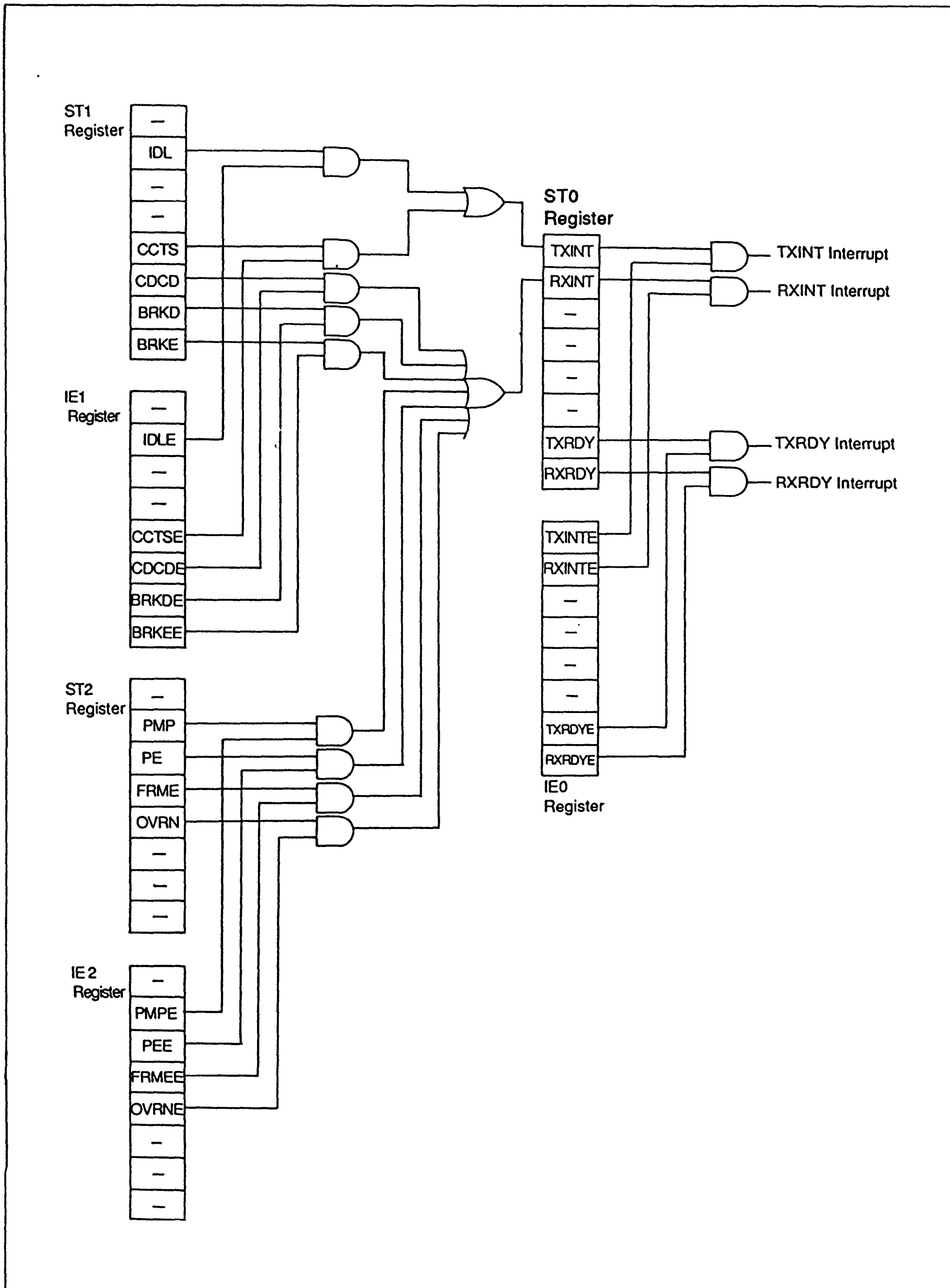


Figure 10-32. ASCI Interrupt Block Diagram

10.7 Baud Rate Generator (BRG)

The ASCI contains a baud rate generator (BRG) to generate the ASCI clock.

10.7.1 BRG Features

- Output frequency: $f\varnothing/2$ to $f\varnothing/2^{17}$
- Frequency accuracy: $\pm 0.5\%$ for $f\varnothing/100 \geq f \geq f\varnothing/2^{17}$

  Frequency accuracy for $f\varnothing/2 \geq f \geq f\varnothing/2^{17}$:

  $| f\text{-}fBRG| \leq 50/\text{Time constant register value } (\%)$, where f = target frequency, fBRG = actual BRG output frequency, and $f\varnothing$ = frequency of the system clock Ø.

Figure 10-33 shows the BRG block diagram.

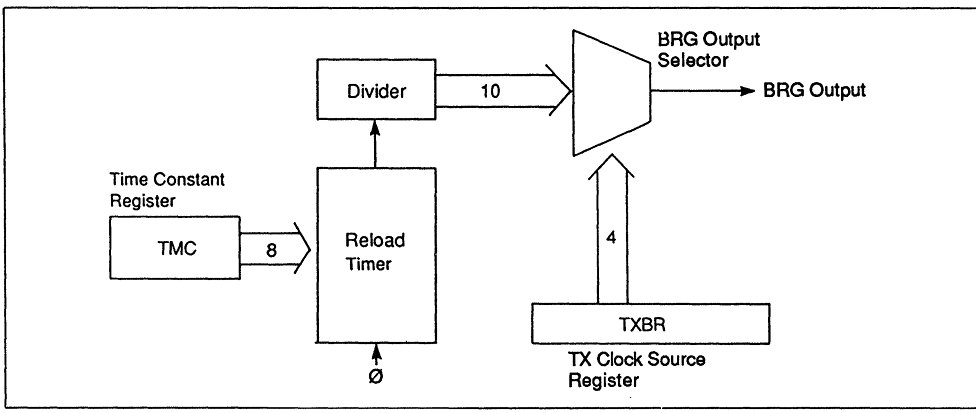


Figure 10-33. BRG Block Diagram

10.7.2 BRG Operation

BRG output frequency (fBRG) is determined by the TMC register and the TXBR3-TXBR0 bits of the TXS register.

The 8-bit TMC register specifies a value to be reloaded into the BRG reload timer. The BRG reload timer counts down every Ø system clock and outputs high level for one clock cycle when it reaches 1 as shown in Figure 10-34. However, note that TMC = 0 is interpreted as 256 and TMC = 1 fixes clock to high level.

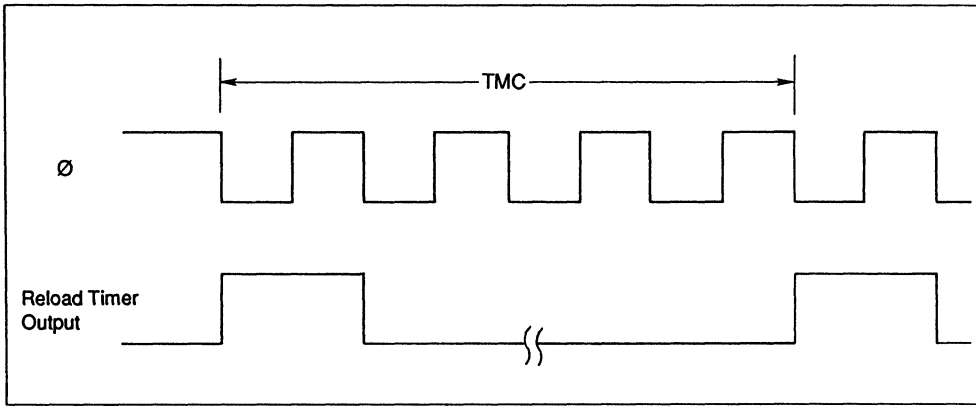


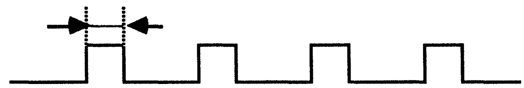
Figure 10-34. Reload Timer Output

Moreover, the reload timer output is supplied to the divider whose dividing ratio is specified by the TXBR3-TXBR0 bits. When TXBR3-TXBR0 = 0, The reload timer output is directly output as the BRG output. When TXBR3-TXBR0 is between 1 and 9, the BRG output has a 50% duty rate and its frequency is determined as follows:

Reload timer output frequency $/2^{TXBR}$ ..... TMC $\neq 1$ (Reload timer output $\neq$ high)
$f\text{Ø}/2^{TXBR}$ .... TMC = 1 (Reload timer output = high)

Table 10-21 summarizes the TXBR bits and TMC values and their corresponding BRG output waveforms.

Table 10-21. TXBR, TMC Values, and BRG Output Waveform Relationships

TXBR3-TXBR0	TMC	BRG Output Waveform
TXBR = 0	TMC ≠1	High-level period 1 System clock TMC = 2, duty rate = 50% TMC ≠ 2, duty rate ≠ 50% 
	TMC = 1	Fixed high (cannot be used as clock)
TXBR = 1-9	—	Duty rate = 50% 

10.7.3 BRG Output Frequency

The BRG output is calculated as follows:

fBRG = fØ/TMC/2^{TXBR}

Where: fBRG = BRG output frequency

fØ = System clock frequency

TMC = Timer constant register (1-256)

TXBR= bits TXBR3-TXBR0 of the TXS register

Note: fBRG must not be equal to fØ.

10.7.4 Register Values and Bit Rates

Table 10-22 lists the BRG register values for given clock frequencies to produce a range of bit rates in asynchronous and clock synchronous modes. TMC is the value of the TMC register value. BR is the value of the bits from TXBR0 to TXBR3. CM is clock mode in asynchronous mode (bit rate/clock rate).

Table 10-22. Clock Frequency/Bit Rate Settings

Asynchronous Mode

fØ	1.7898 MHz				2.4576 MHz			
Bit Rate	**TMC**	**BR**	**CM**	**Deviation (%)**	**TMC**	**BR**	**CM**	**Deviation (%)**
38400	-	-	-	-	1	1	1/32	0.00
1920	-	-	-	-	1	1	1/64	0.00
9600	-	-	-	-	1	2	1/64	0.00
4800	-	-	-	-	1	3	1/64	0.00
2400	47	0	1/16	-0.83	1	4	1/64	0.00
1200	93	0	1/16	-0.25	1	5	1/64	0.00
600	93	0	1/32	-0.25	1	6	1/64	0.00
300	93	0	1/64	-0.25	1	7	1/64	0.00
150	93	1	1/64	-0.25	1	8	1/64	0.00
110	127	1	1/64	-0.10	175	1	1/64	-0.25

fØ	3.072 MHz				4 MHz			
Bit Rate	**TMC**	**BR**	**CM**	**Deviation (%)**	**TMC**	**BR**	**CM**	**Deviation (%)**
38400	5	0	1/16	0.00	-	-	-	-
19200	5	0	1/32	0.00	13	0	1/16	0.16
9600	5	0	1/64	0.00	13	0	1/32	0.16
4800	5	1	1/64	0.00	13	0	1/64	0.16
2400	5	2	1/64	0.00	13	1	1/64	0.16
1200	5	3	1/64	0.00	13	2	1/64	0.16
600	5	4	1/64	0.00	13	3	1/64	0.16
300	5	5	1/64	0.00	13	4	1/64	0.16
150	5	6	1/64	0.00	13	5	1/64	0.16
110	109	2	1/64	0.08	71	3	1/64	0.03

Table 10-22. Clock Frequency/Bit Rate Settings (cont.)

fØ	4.608 MHz				4.9152 MHz			
Bit Rate	TMC	BR	CM	Deviation (%)	TMC	BR	CM	Deviation (%)
38400	-	-	-	-	1	1	1/64	0.00
19200	15	0	1/16	0.00	1	2	1/64	0.00
9600	15	0	1/32	0.00	1	3	1/64	0.00
4800	15	0	1/64	0.00	1	4	1/64	0.00
2400	15	1	1/64	0.00	1	5	1/64	0.00
1200	15	2	1/64	0.00	1	6	1/64	0.00
600	15	3	1/64	0.00	1	7	1/64	0.00
300	15	4	1/64	0.00	1	8	1/64	0.00
150	15	5	1/64	0.00	1	9	1/64	0.00
110	41	4	1/64	-0.22	175	2	1/64	-0.25

fØ	6 MHz				6.144 MHz			
Bit Rate	TMC	BR	CM	Deviation (%)	TMC	BR	CM	Deviation (%)
38400	-	-	-	-	5	0	1/32	0.00
19200	-	-	-	-	5	0	1/64	0.00
9600	39	0	1/16	0.16	5	1	1/64	0.00
4800	39	0	1/32	0.16	5	2	1/64	0.00
2400	39	0	1/64	0.16	5	3	1/64	0.00
1200	39	1	1/64	0.16	5	4	1/64	0.00
600	39	2	1/64	0.16	5	5	1/64	0.00
300	39	3	1/64	0.16	5	6	1/64	0.00
150	39	4	1/64	0.16	5	7	1/64	0.00
110	213	2	1/64	0.03	109	3	1/64	0.08

Table 10-22. Clock Frequency/Bit Rate Settings (cont.)

fØ	8 MHz				9.216 MHz			
Bit Rate	TMC	BR	CM	Deviation (%)	TMC	BR	CM	Deviation (%)
38400	13	0	1/16	0.16	15	0	1/16	0.00
19200	13	0	1/32	0.16	15	0	1/32	0.00
9600	13	0	1/64	0.16	15	0	1/64	0.00
4800	13	1	1/64	0.16	15	1	1/64	0.00
2400	13	2	1/64	0.16	15	2	1/64	0.00
1200	13	3	1/64	0.16	15	3	1/64	0.00
600	13	4	1/64	0.16	15	4	1/64	0.00
300	13	5	1/64	0.16	15	5	1/64	0.00
150	13	6	1/64	0.16	15	6	1/64	0.00
110	71	4	1/64	0.03	41	5	1/64	-0.22

fØ	9.830 MHz				10 MHz			
Bit Rate	TMC	BR	CM	Deviation (%)	TMC	BR	CM	Deviation (%)
38400	2	1	1/64	0.00	-	-	-	-
19200	2	2	1/64	0.00	-	-	-	-
9600	2	3	1/64	0.00	65	0	1/16	0.16
4800	2	4	1/64	0.00	65	0	1/32	0.16
2400	2	5	1/64	0.00	65	0	1/64	0.16
1200	2	6	1/64	0.00	65	1	1/64	0.16
600	2	7	1/64	0.00	65	2	1/64	0.16
300	2	8	1/64	0.00	65	3	1/64	0.16
150	2	9	1/64	0.00	65	4	1/64	0.16
110	175	3	1/64	-0.25	89	4	1/64	-0.25

Table 10-22. Clock Frequency/Bit Rate Settings (cont.)

Clock Synchronous Mode

fØ	2.4576 MHz			3.072 MHz		
Bit Rate	TMC	BR	Deviation (%)	TMC	BR	Deviation (%)
38400	32	1	0.00	40	1	0.00
19200	32	2	0.00	40	2	0.00
9600	32	3	0.00	40	3	0.00
4800	32	4	0.00	40	4	0.00
2400	32	5	0.00	40	5	0.00
1200	32	6	0.00	40	6	0.00
600	32	7	0.00	40	7	0.00
300	32	8	0.00	40	8	0.00

fØ	4 MHz			4.608 MHz		
Bit Rate	TMC	BR	Deviation (%)	TMC	BR	Deviation (%)
38400	52	1	0.16	60	1	0.00
19200	52	2	0.16	60	2	0.00
9600	52	3	0.16	60	3	0.00
4800	52	4	0.16	60	4	0.00
2400	52	5	0.16	60	5	0.00
1200	52	6	0.16	60	6	0.00
600	52	7	0.16	60	7	0.00
300	52	8	0.16	60	8	0.00

Table 10-22. Clock Frequency/Bit Rate Settings (cont.)

fØ	4.9152 MHz			6 MHz		
Bit Rate	TMC	BR	Deviation (%)	TMC	BR	Deviation (%)
38400	64	1	0.00	78	1	0.16
19200	64	2	0.00	78	2	0.16
9600	64	3	0.00	78	3	0.16
4800	64	4	0.00	78	4	0.16
2400	64	5	0.00	78	5	0.16
1200	64	6	0.00	78	6	0.16
600	64	7	0.00	78	7	0.16
300	64	8	0.00	78	8	0.16

fØ	6.144 MHz			8 MHz		
Bit Rate	TMC	BR	Deviation (%)	TMC	BR	Deviation (%)
38400	80	1	0.00	104	1	0.16
19200	80	2	0.00	104	2	0.16
9600	80	3	0.00	104	3	0.16
4800	80	4	0.00	104	4	0.16
2400	80	5	0.00	104	5	0.16
1200	80	6	0.00	104	6	0.16
600	80	7	0.00	104	7	0.16
300	80	8	0.00	104	8	0.16

Table 10-22. Clock Frequency/Bit Rate Settings (cont.)

fØ	9.216 MHz			9.8304 MHz		
Bit Rate	**TMC**	**BR**	**Deviation (%)**	**TMC**	**BR**	**Deviation (%)**
38400	120	1	0.00	128	1	0.00
19200	120	2	0.00	128	2	0.00
9600	120	3	0.00	128	3	0.00
4800	120	4	0.00	128	4	0.00
2400	120	5	0.00	128	5	0.00
1200	120	6	0.00	128	6	0.00
600	120	7	0.00	128	7	0.00
300	120	8	0.00	128	8	0.00

fØ	10 MHz		
Bit Rate	**TMC**	**BR**	**Deviation(%)**
38400	130	1	0.16
19200	130	2	0.16
9600	130	3	0.16
4800	130	4	0.16
2400	130	5	0.16
1200	130	6	0.16
600	130	7	0.16
300	130	8	0.16

TMC: TMC7-TMC0 bits in TMC

BR: TXBR3-TXBR0 bits in TXS

CM: Clock mode in asynchronous mode (bit rate / clock rate)

10.8 Future ASCI Compatibility

To maintain compatibility with future expanded versions, the following must be observed:

1. The lower 4 bits of the RXS register and of the TXS register must be the same. Reserved bits must always be cleared to 0.

2. TRB must be programmed while TXRDY = 1. TRB must be read while RXRDY = 1.

The maximum bit rates for the ASCI can be obtained by the calculations listed in Table 10-23. Transmission at bit rates exceeding these limits may cause malfunction.

Table 10-23. ASCI Maximum Bit Rates

		Maximum Bit Rate Calculations	
Protocol Mode	**Clock Mode**	**External Clock**	**Internal BRG**
Asynchronous	1/64	fØ ÷ 160	fØ ÷ 128
	1/32	fØ ÷ 80	fØ ÷ 64
	1/16	fØ ÷ 40	fØ ÷ 32
	1/1	fØ ÷ 2.5	fØ ÷ 2
Clock synchronous	1/1	fØ ÷ 2.5	fØ ÷ 2

fØ: System clock frequency

For example, for 1/32 clock mode in asynchronous mode in which an external clock signal application and system clock frequency (H16 operation frequency) of 10 MHz are chosen, the maximum bit rate is obtained from calculating fØ ÷ 80 as follows:

10 MHz ÷ 80 = 125 kbps

This calculation shows the maximum rate at which transmission and reception can operate. In actuality, however, the maximum transmitting rates are lowered, while the maximum receiving rates match the calculations. For 1/1 clock mode in asynchronous mode or clock synchronous mode, transmitted data is defined after a t_{TDLY} delay following the falling edge of the clock signal, as shown in the Figure 10-35. The receiving device samples data at the rising edge of the clock signal after receive setup time t_{RSUT}. Accordingly, the minimum low period of the clock signal t_L is obtained by:

$$t_L = t_{TDLY} + t_{RSUT}$$

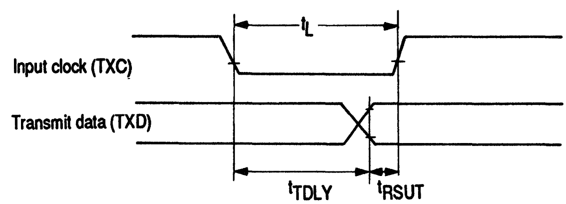


Figure 10-35. Input Clock and Transmit Data

The maximum frequency that satisfies this low period is the maximum bit rate. For example, with t_{TDLY} of 310 ns and t_{RSUT} of 90 ns, the clock low period is:

$$t_L = 310 \text{ ns} + 90 \text{ ns} = 400 \text{ ns}$$

Assuming that the duty ratio of the clock signal with this low period is 50%, the clock signal frequency is as follows:

$$400 \text{ ns} + 400 \text{ ns} = 800 \text{ ns}$$

Thus, the maximum bit rate is obtained by:

$$\frac{1}{800 \text{ ns}} = 1.25 \text{ Mbps}$$

Section 11. Chip Select Controller

11.1 Overview

The built-in chip select controller can define four areas of 64 kbytes to 16 Mbytes within a 16 Mbytes memory space. When an internal bus master (CPU or DMAC) or external bus master accesses one of the four areas, PCS_0 and PCS_1 on the chip select controller provide signals indicating to the accessed area.

The chip select controller provides the following features:

- Four independently definable areas
- Area size from 64 kbytes to 16 Mbytes
- Each area can be protected from user access
- Operation with external bus master
- Bus error exception processing for illegal address accesses

Figure 11-1 shows a block diagram of the chip select controller.

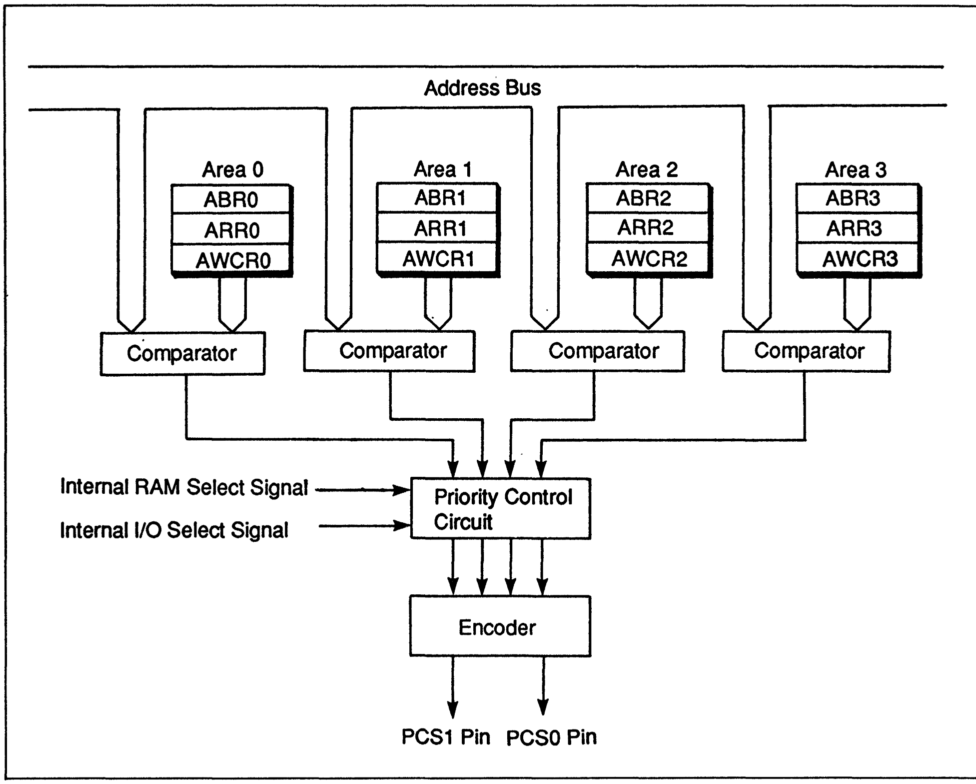


Figure 11-1. Block Diagram of Chip Select Controller

11.2 Chip Select Controller Registers

Table 11-1 shows the chip select controller's read/write registers.

Table 11-1. Chip Select Controller Registers

Area	Register Name	Symbol	Address Offset	R/W	Initial Value	Size
0	Area base register 0	ABR0	H'FF28	R/W	H'0000	W
	Area range register 0	ARR0	H'FF2A	R/W	H'0000	W
	Area wait control register 0	AWCR0	H'FF2C	R/W	H'0047	W
1	Area base register 1	ABR1	H'FF2E	R/W	H'0000	W
	Area range register 1	ARR1	H'FF30	R/W	H'0000	W
	Area wait control register 1	AWCR1	H'FF32	R/W	H'0047	W
2	Area base register 2	ABR2	H'FF34	R/W	H'0000	W
	Area range register 2	ARR2	H'FF36	R/W	H'0000	W
	Area wait control register 2	AWCR2	H'FF38	R/W	H'0047	W
3	Area base register 3	ABR3	H'FF3A	R/W	H'0000	W
	Area range register 3	ARR3	H'FF3C	R/W	H'0000	W
	Area wait control register 3	AWCR3	H'FF3E	R/W	H'0047	W

The above four areas contain functionally identical registers.

11.2.1 Area Base Registers 3-0 (ABR3-ABR0)

Areas 3-0 contain area base registers ABR3-ABR0 having the same function. The ABR register (Figure 11-2) determines the start address of the area on 64 kbyte boundaries within a 16 Mbyte memory space. The upper byte of ABR is reserved and is always read as H'00. At reset, ABR is initialized to H'0000. ABR is not affected by the RESET instruction.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	—	—	AB23	AB22	AB21	AB20	AB19	AB18	AB17	AB16
Initial Value									0	0	0	0	0	0	0	0
Read/Write									R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Area Base 23-16

Specifies bits 23-16 of the area start address

—: Reserved bit. Always read as 0. Cannot be written to.
ABR is not affected by the RESET instruction.

Figure 11-2. Area Base Register (ABR)

11.2.2 Area Range Registers 3-0 (ARR3-ARR0)

The area range register ARR (Figure 11-3) determines the area size from 64 kbytes to 16 Mbytes. The upper byte of the ARR register is reserved and is always read as H'00. At reset, ARR is cleared to H'0000. ARR is not affected by the RESET instruction.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	—	—	AR23	AR22	AR21	AR20	AR19	AR18	AR17	AR16
Initial Value									0	0	0	0	0	0	0	0
Read/Write									R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Area Range 23-16

Specifies area range

—: Reserved bit. Always read as 0. Cannot be written to.
ARR is not affected by the RESET instruction.

Figure 11-3. Area Range Register (ARR)

Table 11-2 shows typical ARR values and area sizes.

Table 11-2. Typical ARR Values and Area Size

ARR Value	00FF	00FE	00FC	00F8	00F0	00E0	00C0	0080	0000
Area Size (bytes)	64K	128K	256K	512K	1M	2M	4M	8M	16M

11.2.3 Area Wait Control Registers 3-0 (AWCR3-AWCR0)

The area wait control register AWCR (Figure 11-4) determines the area $\overline{\text{WAIT}}$ output, access level, and the number of Tw states. The AWCR upper byte is reserved and is always read as H'00. At reset, AWCR is initialized to H'0047. AWCR is not affected by the RESET instruction.

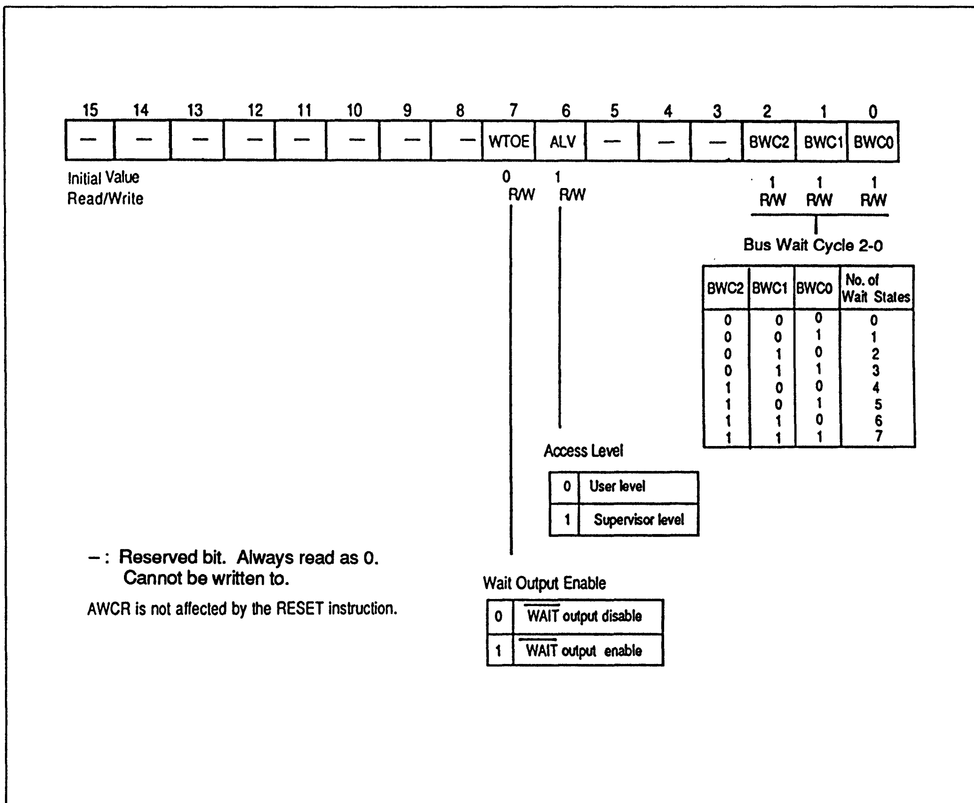


Figure 11-4. Area Wait Control Register (AWCR)

$\overline{\text{WAIT}}$ Output Enable (WTOE): WTOE = 1 enables $\overline{\text{WAIT}}$ output for external device access. The $\overline{\text{WAIT}}$ output depends on BWC2-BWC0 settings. WTOE = 0 disables $\overline{\text{WAIT}}$ output. See "Section 12. Wait State Controller" for details.

Access Level (ALV): ALV determines the area access level. External bus master accesses are performed in supervisor level. If ALV = 0, the area can be accessed in user level. If ALV = 1, the area cannot be accessed in user level. An access level violation exception occurs if an area whose access level is supervisor level is accessed in user level. In addition, an external bus master can access all areas regardless of ALV since external device access is always performed in supervisor level.

Bus Wait Cycle 2-0 (BWC2-BWC0): BWC2-BWC0 determine the number of Tw states to be inserted during internal or external bus master bus cycles. See "Section 12. Wait State Controller" for details.

11.3 Chip Select Controller Operation

11.3.1 Area Determination

The chip select controller determines memory area by the ABR and ARR registers. An area start address can be relocated on 64 kbyte boundaries within the 16 Mbyte memory space by writing bits 23-16 of the desired address into ABR. For example, if the user intends to specify the start address of area 0 as H'C50000, the user writes H'C5 into ABR0 (Figure 11-5).

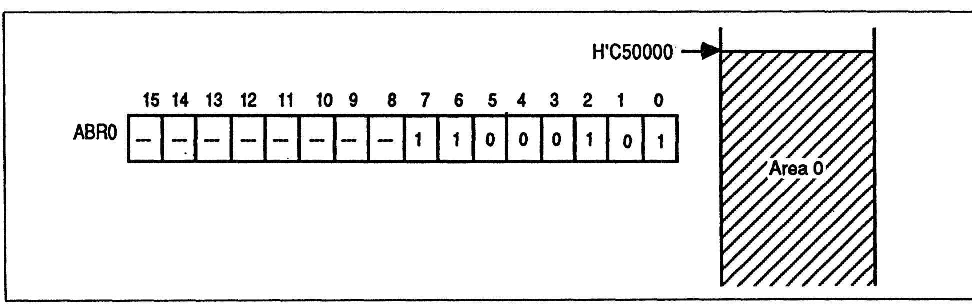


Figure 11-5. Area Start Address Specification

ARR specifies area size. The chip select controller first generates an area match code by ANDing ARR and ABR. At this time, a bit in the area match code is "don't care" if the corresponding ARR bit is 0. Thus, if bits 23-16 of address output from a bus master completely match an area match code, the address is within the area range.

Figure 11-6 shows the area comparator circuit.

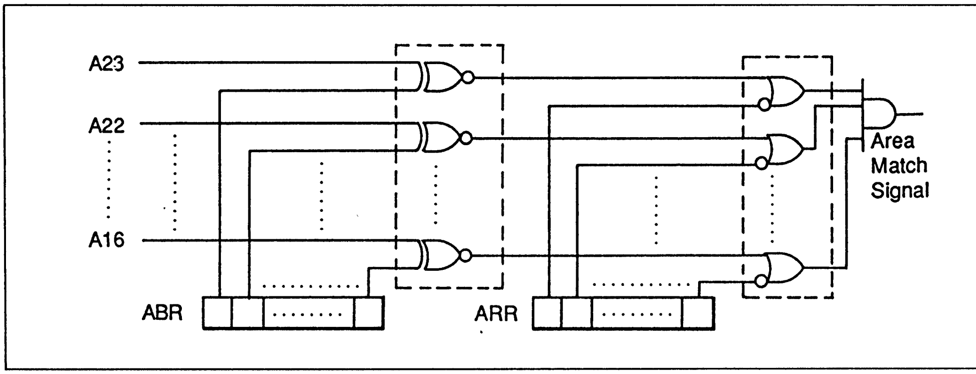


Figure 11-6. Area Comparator

Figure 11-7 gives an example. ABR and ARR are specified as H'00E0 and H'00FC respectively. Accordingly, the area range is H'E00000 to H'E3FFFF. If bits 23-18 of an address output from a bus master match the area match code, the address ranges within the area.

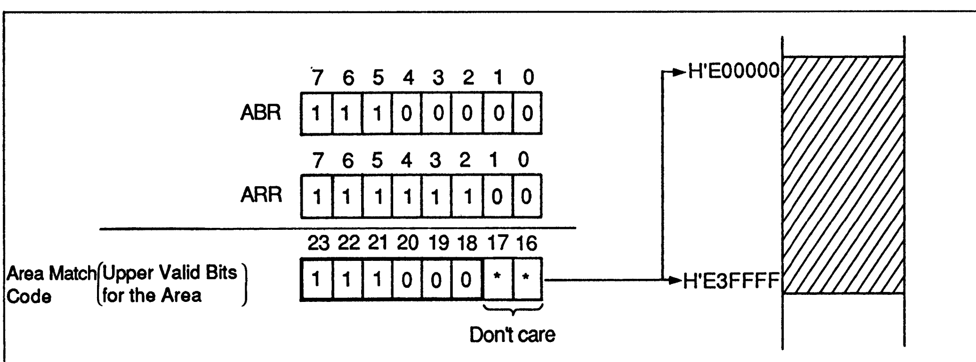


Figure 11-7. Chip Select Area Specification 1

Figure 11-8 shows a general area specification when ABR and ARR are specified as H'15 and H'FC respectively.

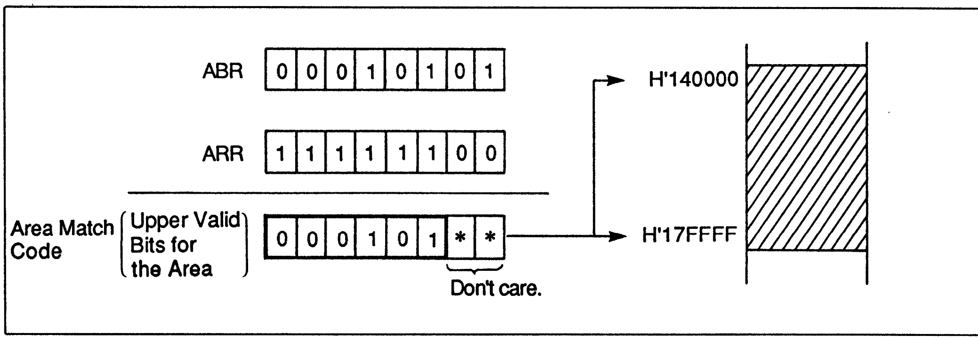


Figure 11-8. Chip Select Area Specification 2

Another special case is shown in Figure 11-9.

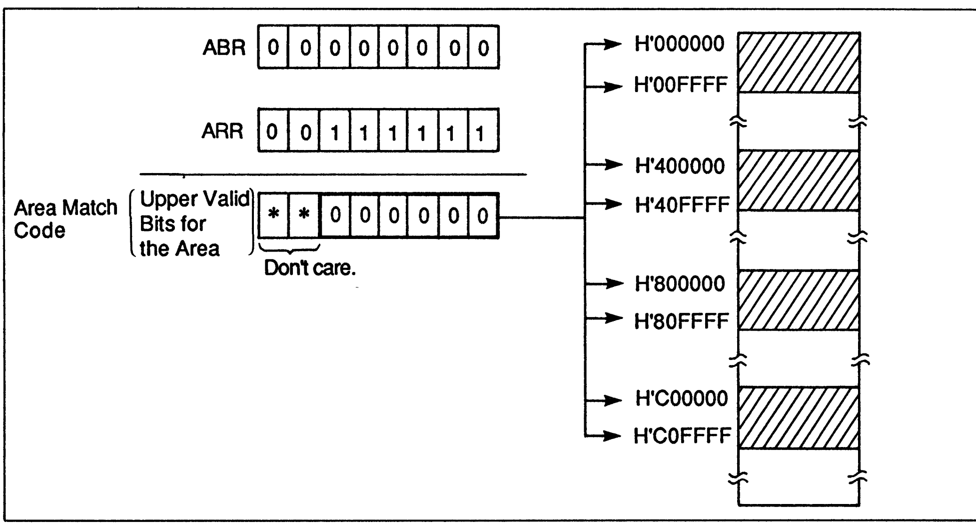


Figure 11-9. Chip Select Area Specification 3

11.3.2 Chip Select Area, Internal RAM, and Internal I/O Overlap

Areas 3-0 can be specified as chip select areas. If an area overlaps another area, the valid area is specified according to the following priority:

Area 0 > Area 1 > Area 2 > Area 3

Figure 11-10 shows an example when chip select areas overlap.

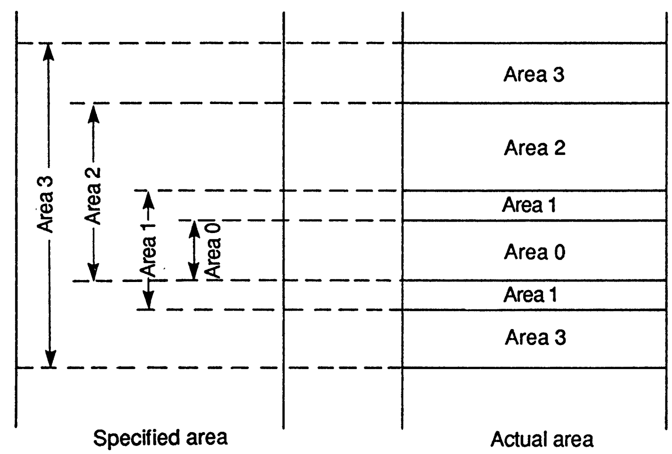


Figure 11-10. Memory Areas When Areas Overlap

If the internal RAM, internal I/O, and chip select areas overlap, each area is determined according to the following priorities:

- For program fetch:

Internal I/O > Area 0 > Area 1 > Area 2 > Area 3

- For data access:

Internal RAM > Internal I/O > Area 0 > Area 1 > Area 2 > Area 3

Note: The internal RAM is not accessed during program fetch since it is in data space. See "Section 5. RAM" for details.

Figure 11-11 shows an example of area determination when the internal RAM, internal I/O, and chip select areas overlap.

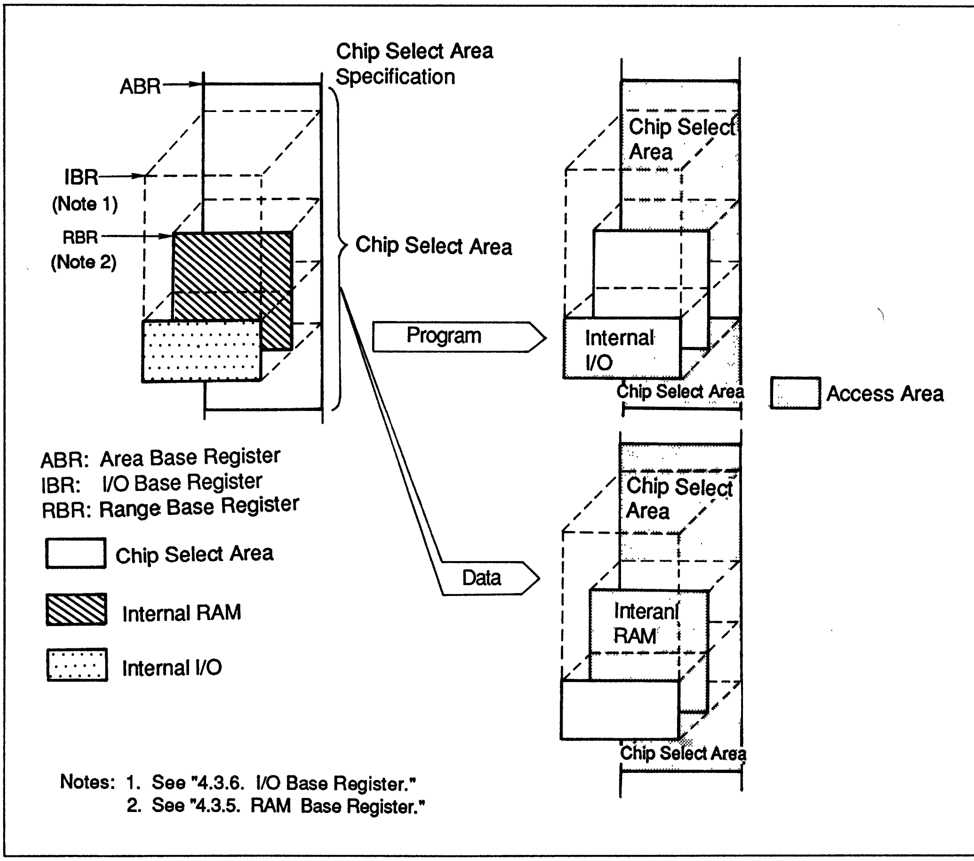


Figure 11-11. Memory Areas When Internal RAM, Internal I/O, and Chip Select Areas Overlap

11.3.3 PCS Output

If an address output from the CPU, internal DMAC, or an external bus master is located in a chip select area, the chip select controller encodes the area on $\overline{PCS_0}$ and $\overline{PCS_1}$ as shown in Table 11-3.

Table 11-3. Area Code

Area	PCS1	PCS0
Area 0	0	0
Area 1	0	1
Area 2	1	0
Area 3, or an area other than areas 0-2, internal I/O, and internal RAM	1	1
Internal I/O or internal RAM	Undefined	Undefined

11.3.4 Access Level

Each area can specify an access level by the ALV bit of AWCR. ALV = 1 specifies the area as accessible only in supervisor level and disables any access in user level. If a supervisor level area is accessed in user level, an access level violation exception occurs. See "4.6 Processing States and Privilege Modes" for details. ALV = 0 specifies the area as user level and allows accesses in both user and supervisor levels.

11.3.5 Access Disable Area

Areas other than chip select area, internal RAM, and internal I/O are called access disable areas. If the CPU or internal DMAC accesses an access disable area, a bus error exception occurs. See "4.6 Processing States and Privilege Modes" for details.

11.4 Chip Select Controller Applications

11.4.1 Small Application System with ROM and I/O

Figure 11-12 shows a small application system in which ROM and I/O are selected by PCS_0 and PCS_1, respectively. Figure 11-13 shows the resulting memory map when the registers are programmed as shown.

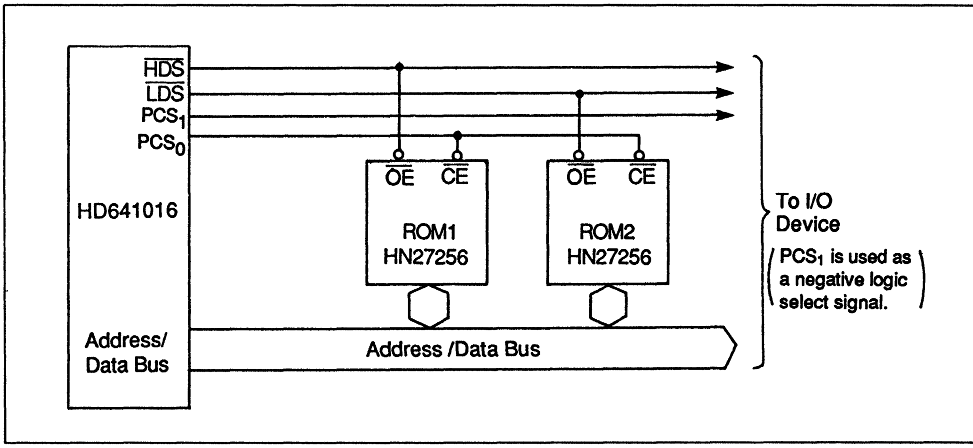


Figure 11-12. Small Application System

Register Name	Value
ABR0	H'00FE
ARR0	H'00FF
ABR1	H'0080
ARR1	H'00FF
ABR 2	H'0000
ARR2	H'00FF
ABR3	H'0000
ARR3	H'00FF

RBR	H'FFFF0000
IBR	H'FFFFFE00

Notes: 1. This area cannot be specified as memory or external I/O area.
2. Area 3 overlaps area 2.

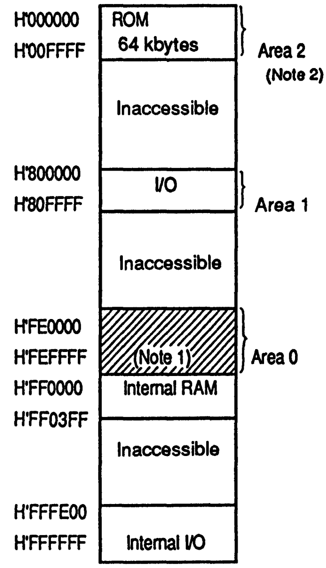


Figure 11-13. Memory Map of Small System

11.4.2 Expanded Application

Figure 11-14 shows an expanded application system with ROM, DRAM, SRAM, and I/O. Figure 11-15 shows the resulting memory map when the registers are programmed as shown.

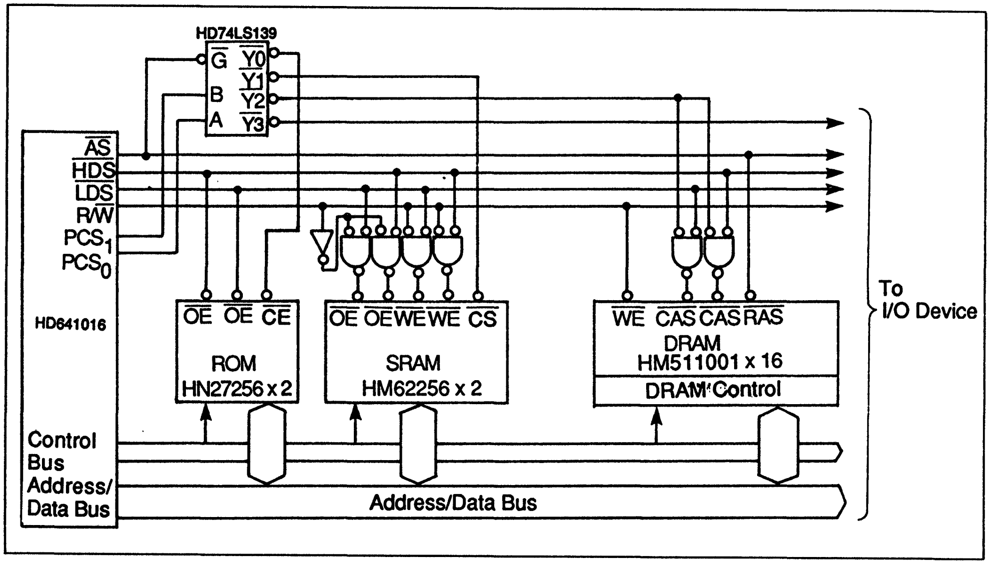


Figure 11-14. Expanded Application System

Register Name	Value
ABR0	H'0000
ARR0	H'00FF
ABR1	H'0001
ARR1	H'00FF
ABR2	H'0080
ARR2	H'00E0
ABR3	H'00E0
ARR3	H'00FF

RBR	H'FFFF0000
IBR	H'FFFFFE00

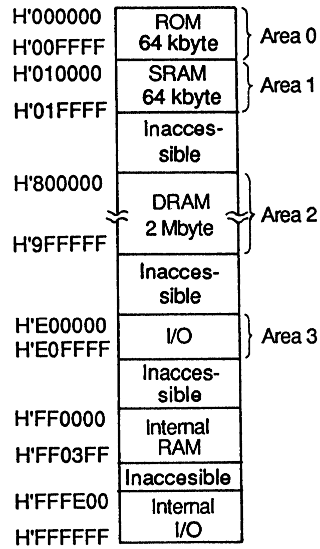


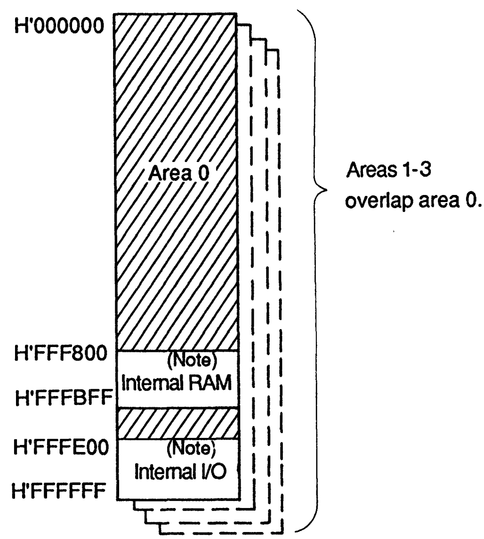
Figure 11-15. Memory Map of Expanded Application System

11.5 Chip Select Controller and Reset

At reset, the chip select controller is initialized as shown in Table 11-4 and all areas overlap as shown in Figure 11-16. At this time, the highest priority area, area 0, occupies the whole address space. Therefore, reset vectors are accessed from area 0 during reset.

Note that the ALV bits are all set to 1 during reset to disable user level accesses during reset. If a user level access is attempted, it will generate an access level violation exception.

Register Name	Initial Value
ABR0	H'0000
ARR0	H'0000
AWCR0	H'0047
ABR1	H'0000
ARR1	H'0000
AWCR1	H'0047
ABR2	H'0000
ARR2	H'0000
AWCR2	H'0047
ABR3	H'0000
ARR3	H'0000
AWCR3	H'0047



Note: See "Section 5. RAM" and "Section 6. Internal I/O" for details.

Figure 11-16. Memory Areas at Reset

Table 11-4. Areas Status at Reset

Area	Area Start Address	Area Size	Access Level	$\overline{\text{WAIT}}$ Output	No. of Wait States
Area 0	H'000000	16 Mbytes	Supervisor	High Impedance	7
Area 1	H'000000	16 Mbytes	Supervisor	High Impedance	7
Area 2	H'000000	16 Mbytes	Supervisor	High Impedance	7
Area 3	H'000000	16 Mbytes	Supervisor	High Impedance	7

11.6 Chip Select Controller Operation Notes

1. Any attempt to change the area which is pointed to by the PC will result in the HD641016 malfunction.

2. The chip select controller does not perform any area checks during interrupt acknowledge cycles or dynamic RAM refresh cycles (PCS0 and PCS1 must be set to 1s).

3. To insure correct program prefetching, the topmost 8 bytes of an area must not be used for storing program instructions (see "5.3 RAM Access" for details). If they are, the CPU may eventually prefetch from an area other than a chip select areas and cause a bus error exception. As shown in Figure 11-17, if the CPU prefetches ② after fetching ① from the prefetch queue, a bus error exception occurs since ② is not located within a chip select area.

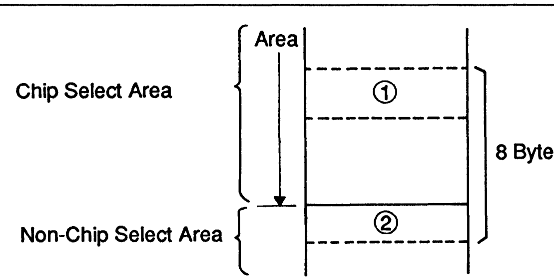


Figure 11-17. Bus Error Caused by an Illegal Area Access

Section 12. Wait State Controller

12.1 Overview

To facilitate interface with slow memories and I/O devices, the HD641016 can employ Tw states to extend bus cycles. External $\overline{\text{WAIT}}$ signals or software controls the insertion of Tw states. The wait state controller has the following features:

- Individually programmable Tw states for each area
- Hardware Tw state insertion by external $\overline{\text{WAIT}}$ input
- Flexible bus cycle extension

Figure 12-1 shows a block diagram of the wait state controller.

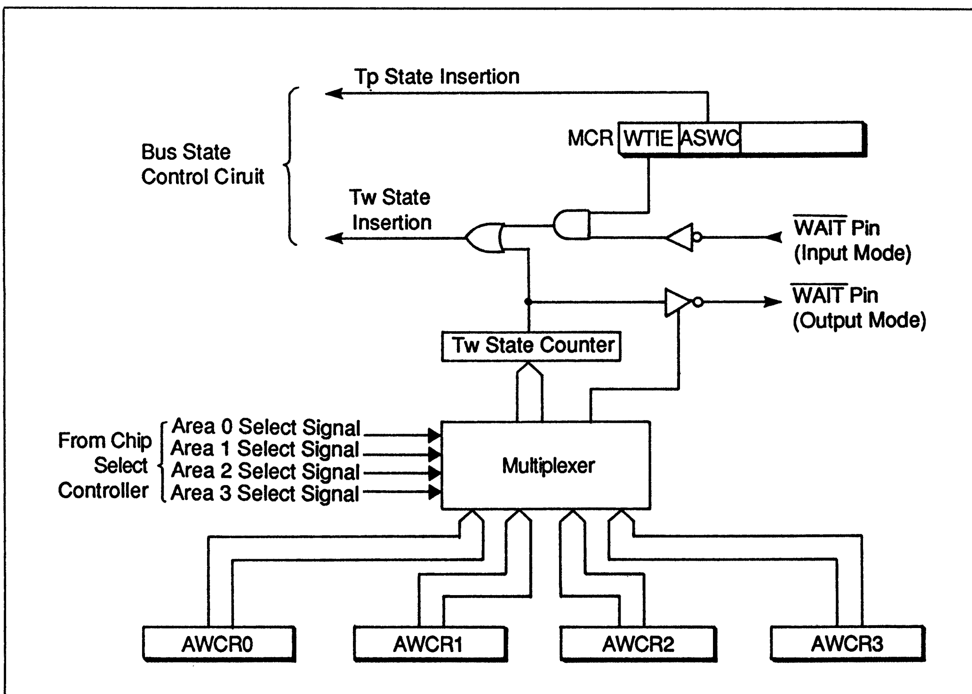


Figure 12-1. Wait State Controller Block Diagram

12.2 Wait State Controller Registers

Table 12-1 shows the wait state controller registers.

Table 12-1. Wait State Controller Registers

Area	Register Name	Symbol	Address Offset	R/W	Initial Value	Size
0	Area wait control register 0	AWCR0	H'FF2C	R/W	H'0047	W
1	Area wait control register 1	AWCR1	H'FF32	R/W	H'0047	W
2	Area wait control register 2	AWCR2	H'FF38	R/W	H'0047	W
3	Area wait control register 3	AWCR3	H'FF3E	R/W	H'0047	W
0-3	Memory control register	MCR	H'FFF8	R/W	H'F0E0	W

12.2.1 Area Wait Control Registers 3-0 (AWCR3-AWCR0)

The wait state controller shares the 16-bit area wait control registers AWCR (Figure 12-2) with the chip select controller (Section 11). They determines $\overline{\text{WAIT}}$ output, access level, and the number of Tw states for each area. The upper byte of AWCR is reserved and is always read as 00H. At reset, AWCR is initialized to H'0047. AWCR is not affected by the RESET instruction.

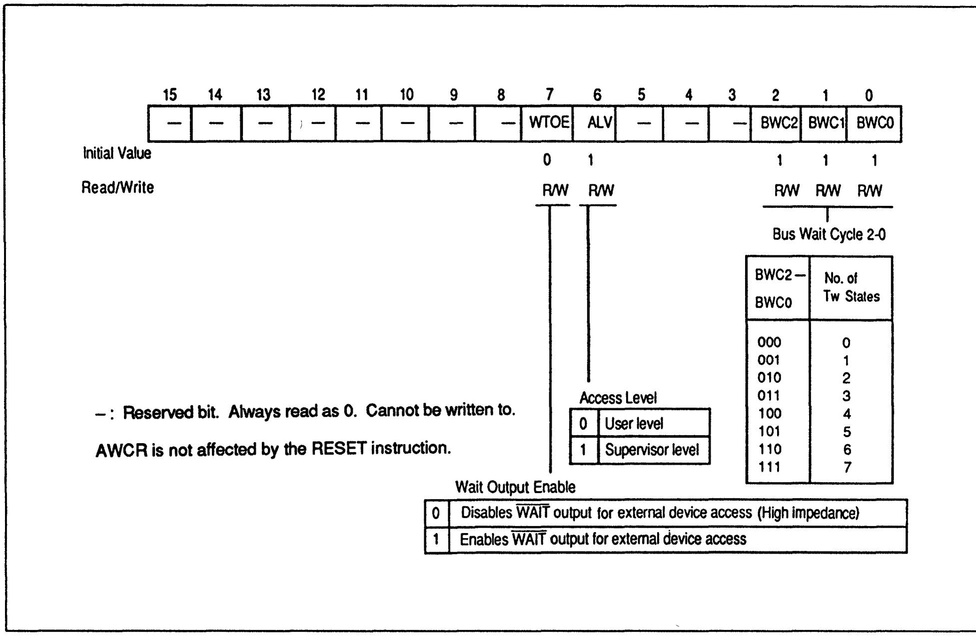


Figure 12-2. Area Wait Control Register (AWCR)

$\overline{\text{WAIT}}$ Output Enable (WTOE): WTOE = 1 enables the $\overline{\text{WAIT}}$ output for external device access. The $\overline{\text{WAIT}}$ output depends on BWC2-BWC0 settings. WTOE = 0 disables $\overline{\text{WAIT}}$ output.

Access Level (ALV): See "Section 11. Chip Select Controller" for details.

Bus Wait Cycle 2-0 (BWC2-BWC0): BWC2-BWC0 determine the number of Tw states to be inserted during internal or external bus master bus cycles as shown in Table 12-2. If WTOE = 1, the $\overline{\text{WAIT}}$ signal is output for a number of states specified by the number of Tw states (BWC2-BWC0) +1 during external bus master bus cycles.

Table 12-2. BWC2-BWC0 and Tw States

BWC2	BWC1	BWC0	Number of Tw States
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

12.2.2 Memory Control Register (MCR)

The memory control register MCR (Figure 12-3) controls the $\overline{WAIT}$ input, the number of Tp states to be inserted prior to the T_{R1} state, and DRAM refresh. During reset, the MCR register is initialized to H'F0E0. MCR is not affected by the RESET instruction.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	WTIE	ASWC	IWC1	IWC0	—	—	—	RPL	REFE	RWC1	RWC0	RRN1	RRN0	CYC2	CYC1	CYC0
Initial Value	1	1	1	1				0	1	1	1	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W				R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Interrupt Wait Cycle 1, 0

IWC1-IWC0	No. of Tw States,
00	0
01	1
10	2
11	3

AS Wait Control

0	Disable Tp state insertion
1	Enable Tp state insertion

$\overline{WAIT}$ Input Enable

0	Disable $\overline{WAIT}$ input
1	Enable $\overline{WAIT}$ input

Refresh Request Number

RRN1-RRN0	Max No. of Deferred Refresh Cycles
00	31
01	63
10	127
11	255

Refresh Wait Cycle

RWC1-RWC0	No. of TRW States
00	0
01	1
10	2
11	3

Refresh Cycle 2-0

CYC2-CYC0	Refresh Cycle Interval
000	32
001	64
010	96
011	128
100	160
101	192
110	224
111	256

Refresh Enable

0	Disable refresh cycle insertion
1	Enable refrresh cycle insertion

Refresh Priority Level

0	Normal refresh
1	Burst refresh

–: Reserved bit. Always read as 0.
Cannot be written to.

MCR is not affected by the RESET instruction.

Figure 12-3. Memory Control Register (MCR)

$\overline{\text{WAIT}}$ Input Enable (WTIE): WTIE controls how the wait state controller responds to $\overline{\text{WAIT}}$ input. If WTIE = 0, it ignores the external $\overline{\text{WAIT}}$ input. If WTIE = 1, the wait state controller inserts a Tw state depending on the Tw state control signal input to the $\overline{\text{WAIT}}$ pin during CPU, DRAM refresh, or internal DMAC cycles. See "12.3 Wait State Controller Operation" for details.

AS Wait Control (ASWC): ASWC = 1 enables Tp state insertion prior to T1 state of CPU, DMA, or refresh cycles. ASWC = 0 disables Tp insertion.

Interrupt Wait Cycle 1,0 (IWC1, IWC0): IWC1 and IWC0 determine the number of Tw states to be inserted during an interrupt acknowledge cycle as shown in Table 12-3.

Table 12-3 IWC1-IWC0 and Tw State

IWC1	IWC0	Number of Tw States
0	0	0
0	1	1
1	0	2
1	1	3

Refresh Priority Level (RPL): RPL determines the refresh priority level. See "Section 13. DRAM Refresh Controller" for details.

Refresh Enable (REFE): REFE enables or disables refresh cycle insertion. See "Section 13. DRAM Refresh Controller" for details.

Refresh Wait Cycle 1, 0 (RWC1, RWC0): RWC1 and RWC0 determine the number of TRW states to be inserted during a refresh cycle. See "Section 13. DRAM Refresh Controller" for details.

Refresh Request Number 1, 0 (RRN1, RRN0): RRN1 and RRN0 determine the maximum number of deferred refresh requests. See "Section 13. DRAM Refresh Controller" for details.

Refresh Cycle 2-0 (CYC2-CYC0): CYC2-CYC0 determine the interval between refresh cycles. See "Section 13. DRAM Refresh Controller" for details.

12.3 Wait State Controller Operation

The wait state controller internally generates a $\overline{\text{WAIT}}$ signal when the wait counter receives the number of Tw states programmed in the corresponding AWCR as shown in Figure 12-4.

The internal WAIT signal ORed with the external $\overline{\text{WAIT}}$ input controls the bus state. Therefore either internal WAIT or external $\overline{\text{WAIT}}$ which is longer, can cause Tw state insertion. Clearing WTIE to 0 masks the external $\overline{\text{WAIT}}$ input.

When WTOE = 1 and external bus masters have control of the bus, the $\overline{\text{WAIT}}$ output is provided to control the external bus masters' bus cycle. Clearing the WTOE bit of an AWCR to 0 masks that area's $\overline{\text{WAIT}}$ output.

The ASWC bit of MCR determines whether or not a Tp state is inserted prior to T1 state.

Table 12-4 shows $\overline{\text{WAIT}}$ pin functions based on bus mode.

Table 12-4. $\overline{\text{WAIT}}$ Pin Functions

Bus Mode	$\overline{\text{WAIT}}$ Pin	
Internal bus mode	Input	WTIE = 1
External bus master mode	High impedance	WTOE = 0
External bus master/Bus control modes	Output low	WTOE = 1

Figure 12-4 shows the wait state controller block diagram.

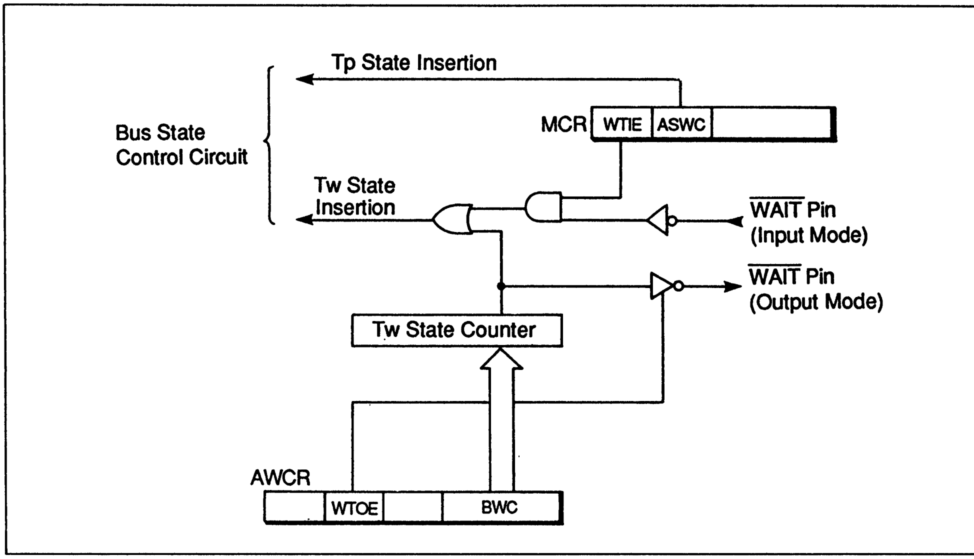


Figure 12-4. Wait State Controller Block Diagram

12.3.1 Programmable Tw State Insertion

Programmable Tw states (Figure 12-5) can be inserted by setting the number of Tw states in the corresponding AWCR. Table 12-5 shows the AWCR0 and MCR programming values when two Tw states are inserted into the area 0 bus cycle.

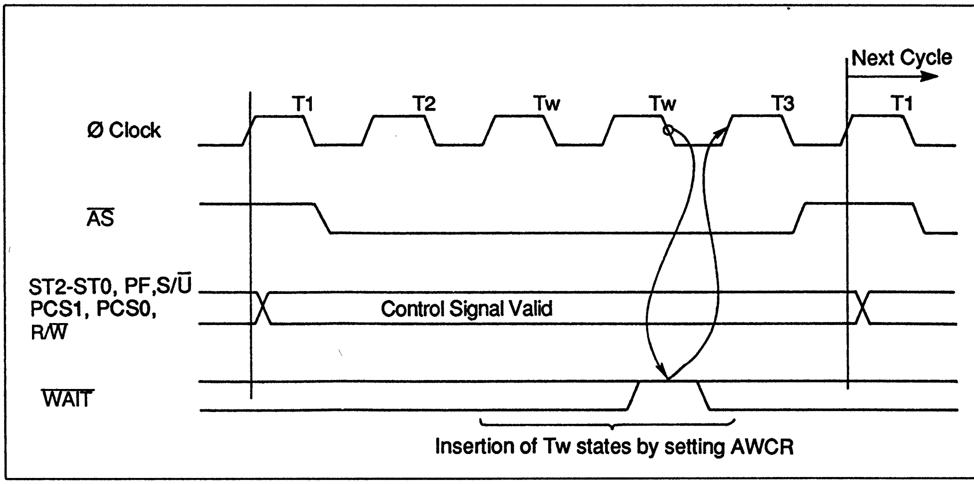


Figure 12-5. Programmable Wait Timing

Table 12-5. Register Values

Register Name	Value
AWCR0	H'0042
MCR	H'8000

12.3.2 Tp State Insertion

A Tp state can be inserted prior to the T1 state if the ASWC bit of MCR is set. Note that the ASWC bit of MCR is common to all chip select areas: if the ASWC bit is set, a Tp state is inserted into all bus cycles other than internal RAM access cycles. Figure 12-6 shows a bus cycle with Tp states inserted when MCR is programmed as shown in Table 12-6.

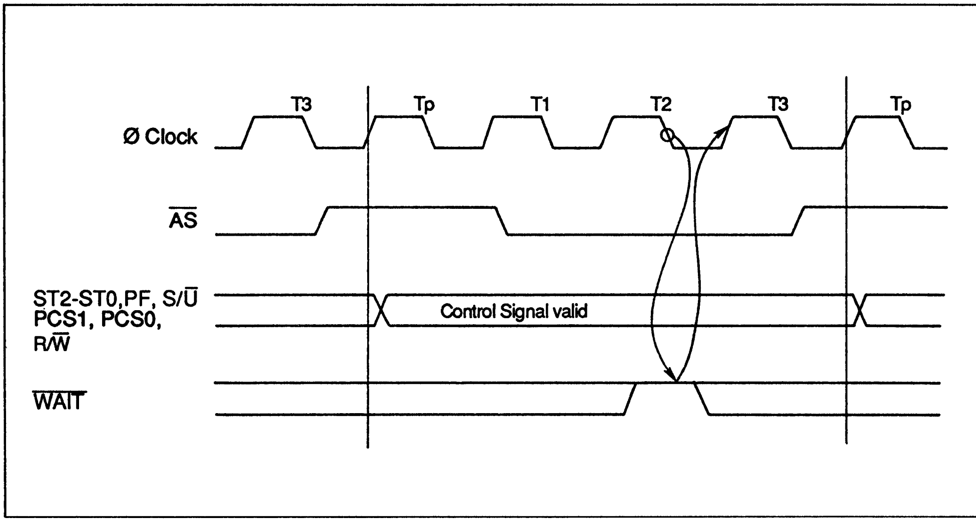


Figure 12-6. Bus Cycle with Tp State

Table 12-6. Register Value

Register Name	Value
MCR	H'C000

12.3.3 Hardware Tw State Insertion (Tw State Insertion with $\overline{\text{WAIT}}$ Input)

If the WTIE bit of MCR is set, the wait state controller samples the $\overline{\text{WAIT}}$ signal at the falling edge of the T2 state. In this situation, a Tw state is inserted if $\overline{\text{WAIT}}$ is low, and the wait state controller samples $\overline{\text{WAIT}}$ at the falling edge of Tw state. Figure 12-7 shows the $\overline{\text{WAIT}}$ timing. The $\overline{\text{WAIT}}$ input is ignored if the WTIE bit is cleared.

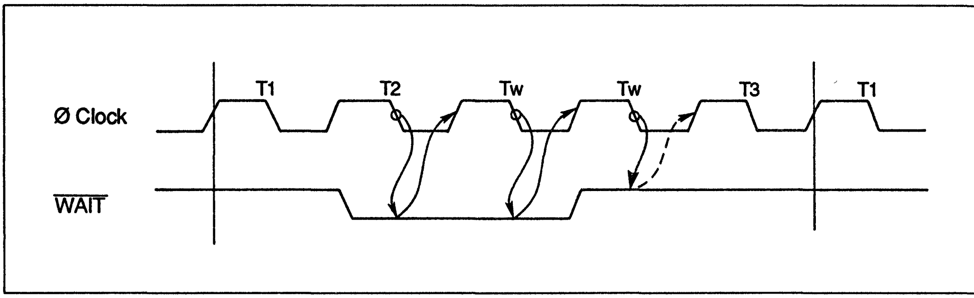


Figure 12-7. $\overline{\text{WAIT}}$ Timing

12.3.4 $\overline{\text{WAIT}}$ Output during Bus Release

If the WTOE bit of AWCR is set during bus release, Tw states are inserted depending on the BWC2-BWC0 bit settings. At this time, the number of Tw states to be inserted are: the number of Tw states specified by BWC2-BWC0 bits + 1. However, note that one Tw state is inserted even if the BWC2-BWC0 bits are all cleared. Figure 12-8 shows $\overline{\text{WAIT}}$ output timing when the BWC2-BWC0 bits are set to "001". In this case, two Tw states are inserted.

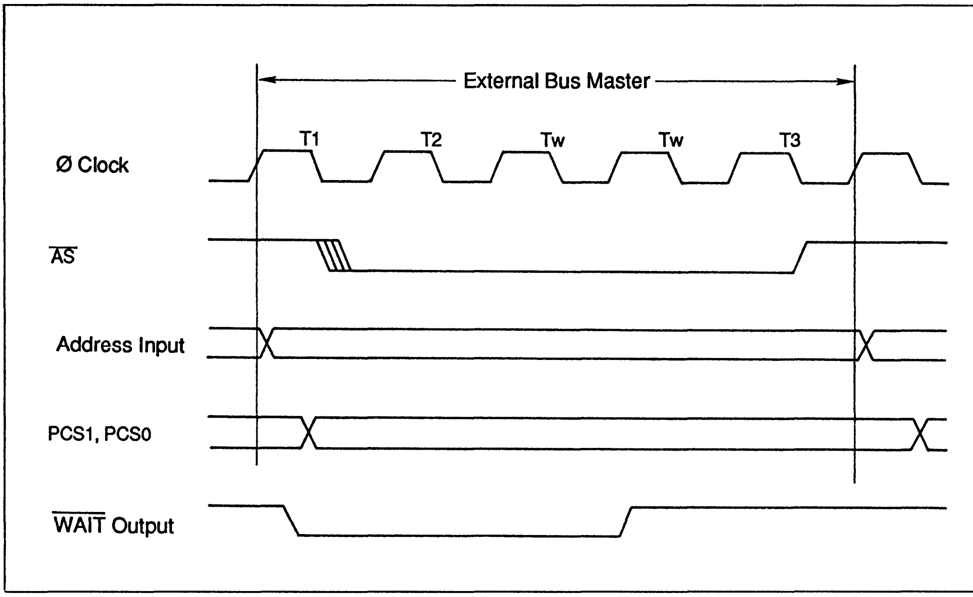


Figure 12-8 . $\overline{\text{WAIT}}$ Output Timing

12.3.5 Tw State Insertion during an Interrupt Acknowledge Cycle

0-3 Tw states can be inserted during an interrupt acknowledge cycle if the number of Tw states is programmed in the IWC1-IWC0 bits of MCR. In addition, if the WTIE bit of MCR is set, an external $\overline{\text{WAIT}}$ input can be also inserted.

12.4 Wait State Controller Notes

If hardware Tw state insertion is requested by external $\overline{\text{WAIT}}$ at the same time that programmable Tw state insertion is requested when WTIE = 1, the number of Tw states will be determined by either the external $\overline{\text{WAIT}}$ or the BWC2-BWC0 bits or, whichever is greater.

Section 13. Dynamic RAM Refresh Controller

13.1 Overview

The HD641016 DRAM refresh controller facilitates interface with dynamic RAM. The DRAM refresh controller has the following features:

- Programmable refresh intervals: 32-256 states
- Refresh address: 11 bits
- Programmable refresh cycle length: 2-6 states
- Programmable refresh bus priority
- Burst refresh function

Figure 13-1 shows a block diagram of the DRAM refresh controller.

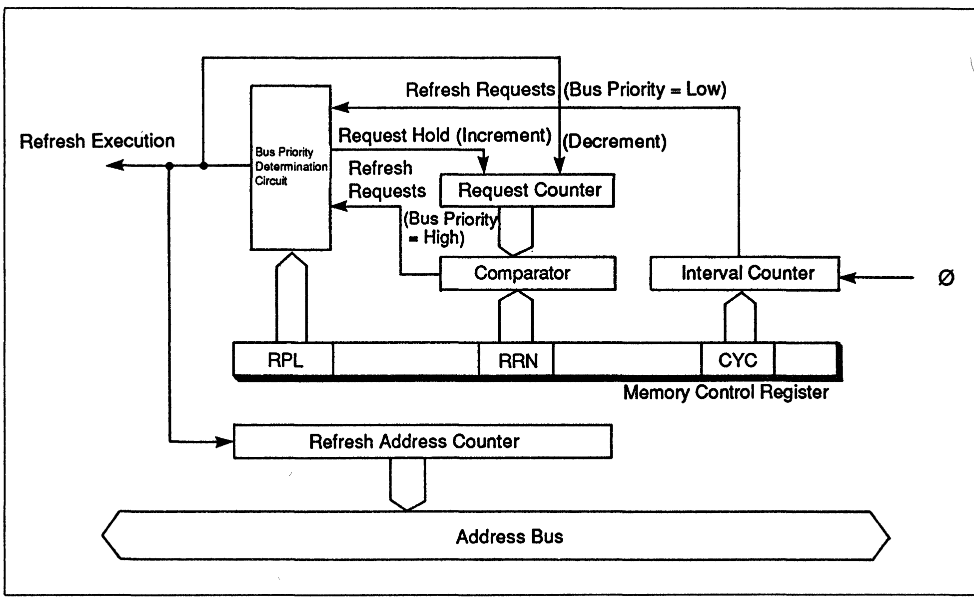


Figure 13-1. DRAM Refresh Controller Block Diagram

13.2 DRAM Refresh Controller Register

The DRAM refresh controller has one register, the memory control register, which it shares with the wait state controller. See Table 13-1.

Table 13-1 DRAM Refresh Controller Register

Register Name	Symbol	Address Offset	R/W	Initial Value	Size
Memory control register	MCR	H'FFF8	R/W	H'F0E0	W

13.2.1 Memory Control Register (MCR)

The memory control register MCR (Figure 13-2) controls the $\overline{\text{WAIT}}$ input, the number of Tp states to be inserted prior to the TR1 state, and DRAM refresh. During power-on reset, the MCR register is initialized to H'F0E0. MCR is not affected by the RESET instruction and manual reset.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	WTIE	ASWC	IWC1	IWC0	—	—	—	RPL	REFE	RWC1	RWC0	RRN1	RRN0	CYC2	CYC1	CYC0
Initial Value	1	1	1	1				0	1	1	1	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W				R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Interrupt Wait Cycle 1, 0

IWC1-IWC0	No. of Tw States
00	0
01	1
10	2
11	3

Refresh Request Number

RRN1-RRN0	Max No. of Deferred Refresh Cycles
00	31
01	63
10	127
11	255

AS Wait Control

0	Disable Tp state insertion
1	Enable Tp state insertion

$\overline{\text{WAIT}}$ Input Enable

0	Disable $\overline{\text{WAIT}}$ input
1	Enable $\overline{\text{WAIT}}$ input

Refresh Wait Cycle

RWC1-RWC0	No. of T_{RW} States
00	0
01	1
10	2
11	3

Refresh Cycle 2-0

CYC2-CYC0	Refresh Cycle Interval
000	32
001	64
010	96
011	128
100	160
101	192
110	224
111	256

Refresh Enable

0	Disable refresh cycle insertion
1	Enable refresh cycle insertion

Refresh Priority Level

0	Normal refresh
1	Burst refresh

—: Reserved bit. Always read as 0.
Cannot be written to.

MCR is not affected by the RESET instruction.

Figure 13-2. Memory Control Register (MCR)

$\overline{\text{WAIT}}$ Input Enable (WTIE): WTIE controls how the wait state controller responds to $\overline{\text{WAIT}}$ input. See "Section 12. Wait State Controller" for details.

AS Wait Control (ASWC): ASWC enables or disables Tp state insertion during CPU, DMA, or refresh cycles. See "Section 12. Wait State Controller," for details.

Interrupt Wait Control 1, 0 (IWC1, 0): IWC1-IWC0 determine the number of Tw states to be inserted during an interrupt acknowledge cycle. See "Section 12. Wait State Controller" for details.

Refresh Priority Level (RPL): In a normal bus cycle, bus priority is as follows:

External bus master > Internal DMAC > Refresh controller > CPU

The RPL bit determines the refresh priority level when the refresh request count exceeds the limit set in the RRN bits.

If RPL = 0, the DRAM refresh controller performs normal refresh and does not perform burst refresh. Bus priority is determined as follows:

External bus master > Refresh controller > Internal DMAC > CPU

If RPL = 1, the DRAM refresh controller performs burst refresh. Bus priority is determined as follows:

Refresh controller > External bus master > Internal DMAC > CPU

Refresh Enable (REFE): REFE = 1 enables refresh cycle insertion. REFE = 0 disables refresh cycle insertion.

Refresh Wait Cycle 1, 0 (RWC1, RWC0): RWC1 and RWC0 determine the number of TRW states to be inserted during a refresh cycle (Table 13-2).

Table 13-2. Number of TRW States

RWC1	RWC0	Number of TRW States
0	0	0
0	1	1
1	0	2
1	1	3

Refresh Request Number 1, 0 (RRN1, RRN0): RRN1 and RRN0 determine the maximum number of deferred refresh requests (Table 13-3).

Table 13-3. Refresh Request Number

RRN1	RRN0	Maximum Number of Deferred Refresh Requests
0	0	31
0	1	63
1	0	127
1	1	255

Refresh Cycle 2-0 (CYC2-CYC0): CYC2-CYC0 determine the interval between refresh cycles (Table 13-4).

Table 13-4. Refresh Request Interval

CYC2	CYC1	CYC0	Refresh Request Intervals (States)
0	0	0	32
0	0	1	64
0	1	0	96
0	1	1	128
1	0	0	160
1	0	1	192
1	1	0	224
1	1	1	256

13.3 DRAM Refresh Controller Operation and Procedure

13.3.1 Basic Operation

The DRAM refresh controller requests refresh cycles according to the interval specified in the CYC2-CYC0 bits. In addition, a Tp state can be inserted prior to a TR1 state by programming the ASWC bit, and a TRW state can be inserted prior to a TR2 state by programming the RWC1-RWC0 bits. Figures 13-3 and 13-4 show refresh bus cycle timings.

If a refresh is requested while a bus master other than the CPU is using the bus, the refresh is not executed and the refresh counter increments by 1. This refresh counter accumulates the number of unexecuted refresh requests; it increments by 1 up to 255 if a refresh is not executed, and it decrements by 1 if a refresh is executed. The refresh counter is cleared to 0 by power-on-reset.

The DRAM refresh controller operates as follows according to the refresh counter and the RPL bit values.

1. RPL = 0
   When the refresh counter exceeds the limit set in the RRN bits, bus priority changes as follows:

   External bus master > Refresh controller > Internal DMAC > CPU

   Moreover, when the refresh counter then exceeds 255, it is cleared to 0 and bus priority returns to the normal priority:

   External bus master > Internal DMAC > Refresh controller > CPU

   Normal refresh is then performed by an external bus master. This function is effective for a bus master, other than the DRAM refresh controller, which uses a bus for a long time performing DRAM refreshes.

2. RPL = 1
   When the refresh counter exceeds the limit set in the RRN bits, bus priority changes as follows:

   Refresh controller > External bus master > Internal DMAC > CPU

If the refresh controller gains the bus, it performs consecutive refresh operation (burst refresh) until the refresh request counter. At this time, the refresh counter stops incrementing when it reaches 255.

The refresh address counter is initialized to 0 and increments by 1 after a refresh is executed. Note that bits A11-A1 of the refresh counter are valid and A23-A12 are fixed to 0s.

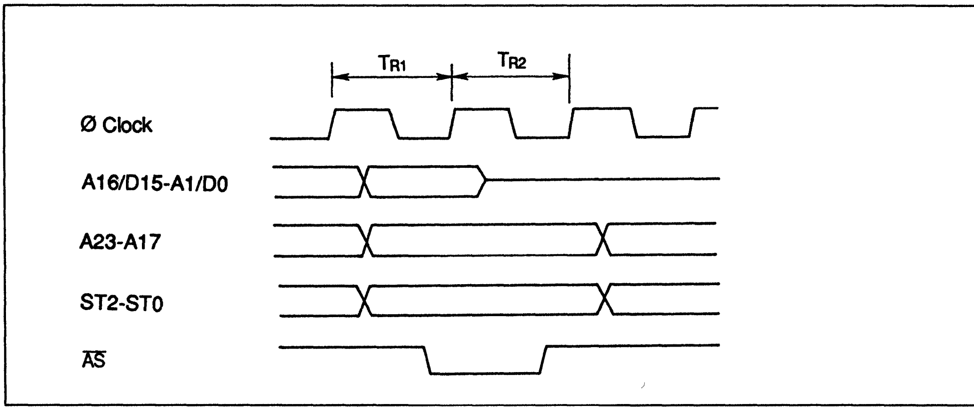


Figure 13-3. Refresh Bus Cycle Timing

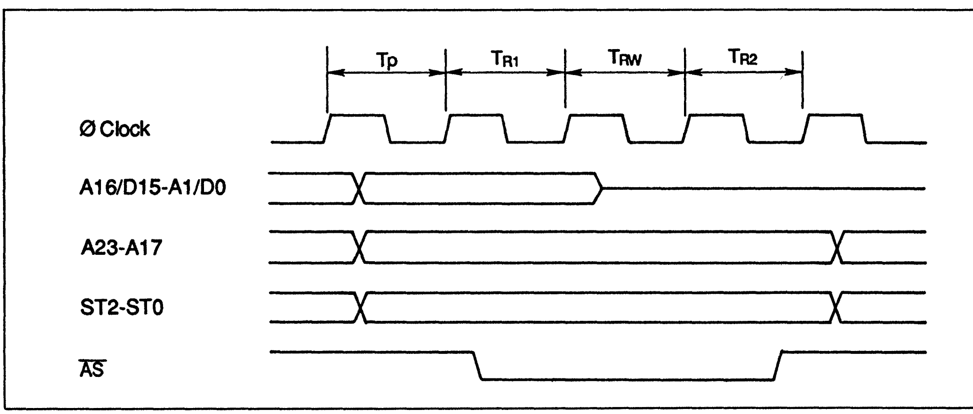


Figure 13-4. Refresh Bus Cycle Timing with Tp and TRW States

13.3.2 TRW State Insertion by $\overline{\text{WAIT}}$

If the WTIE bit of the MCR register is set and the RWC1-RWC0 bits of MCR are programmed to insert one or more TRW states, the refresh controller samples $\overline{\text{WAIT}}$ at the falling edge of a state prior to the TR2 state. In this situation, a TRW state is inserted if $\overline{\text{WAIT}}$ is asserted low. The DRAM refresh controller then samples $\overline{\text{WAIT}}$ at the falling edge of TRW state to determine whether or not another TRW state is inserted. Figure 13-5 shows the refresh timing with a TRW state.

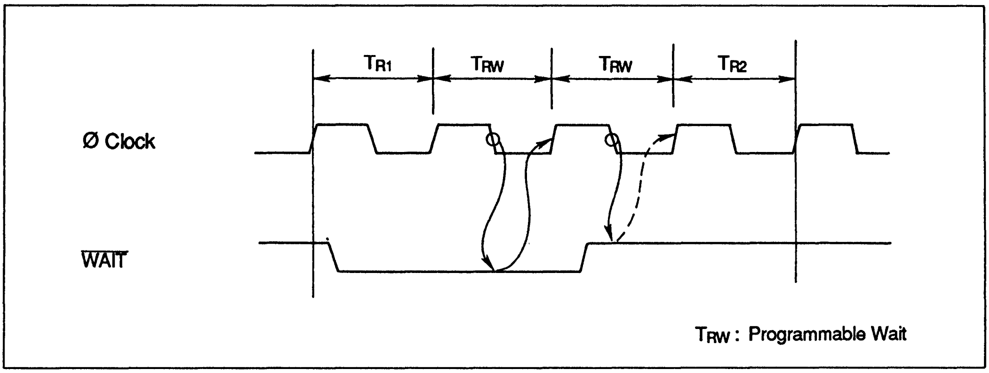


Figure 13-5. Wait State Insertion by $\overline{\text{WAIT}}$

13.4 Application Example

13.4.1 Refresh Insertion Interval

For 256 refresh cycles every 4 ms, the calculated intervals are:

4 ms/256 = 15.625 µs

The values of CYC2-CYC0 for various clock rates are shown in Table 13-5.

Table 13-5. CYC2 - CYC0 for Various Clock Rates

				Insertion Rate (µs) at Given Clock Rate				
CYC2	CYC1	CYC0	Insertion Interval (States)	Ø 12.5 MHz	10 MHz	8 MHz	6 MHz	4 MHz
0	0	0	32	2.4	3.2	4.0	5.3	8.0
0	0	1	64	5.1	6.4	8.0	10.6	16.0
0	1	0	96	7.7	9.6	12.0	16.0	24.0
0	1	1	128	10.2	12.8	16.0	21.3	32.0
1	0	0	160	12.8	16.0	20.0	26.6	40.0
1	0	1	192	15.4	19.2	24.0	32.0	48.0
1	1	0	224	17.9	22.4	28.0	37.3	56.0
1	1	1	256	20.5	25.6	32.0	42.6	64.0

The required insertion interval for Ø = 8 MHz is 96 states. At this interval, the time limit count of the number of requests (Refresh Request Number RRN0-RRN1) is 255. Therefore, the external bus master can control the bus for 96 states x 255 x 0.125 µs = 3.06 ms. If there is a possibility that an external bus master may occupy the bus for more than 3.06 ms, give priority to refresh by setting the RPL bit of the MCR register to 1.

13.4.2 DRAM Interface Circuit

Figure 13-6 shows an interface circuit between the HD641016 and DRAM in area 1. Figure 13-7 shows a timing chart for the circuit. In this example, if Ø = 8 MHz, one bus cycle (T1, T2, T3) is 375 ns.

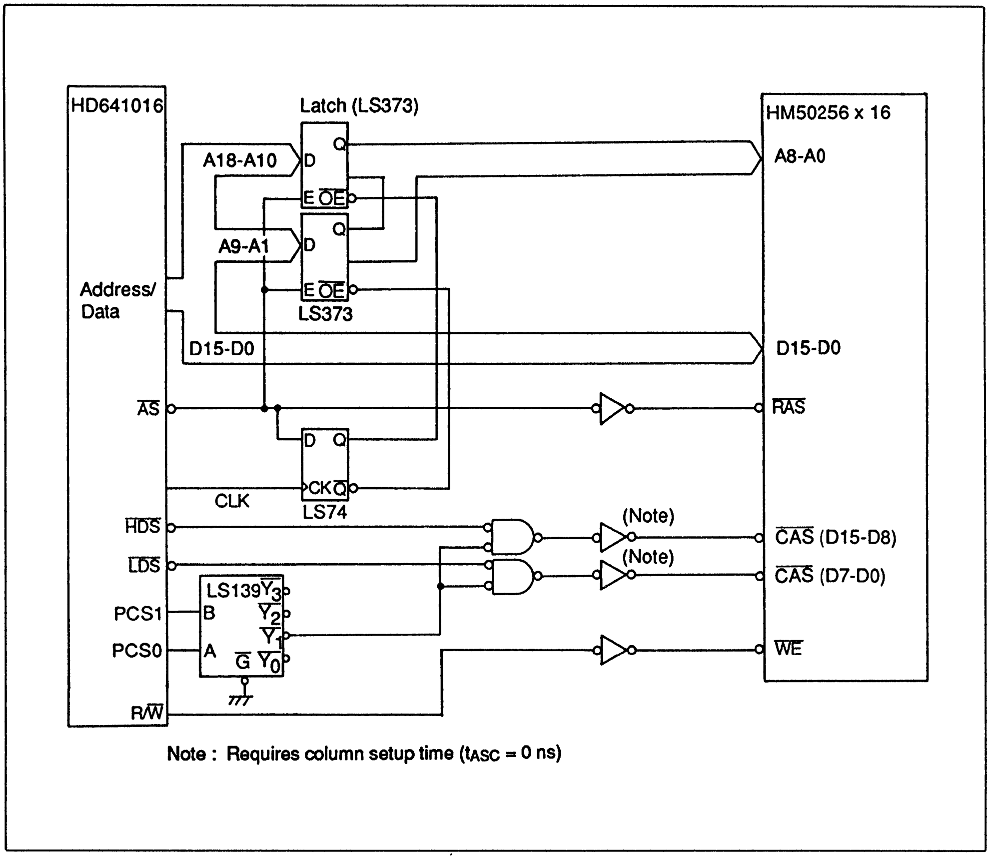


Figure 13-6. DRAM Interface Circuit

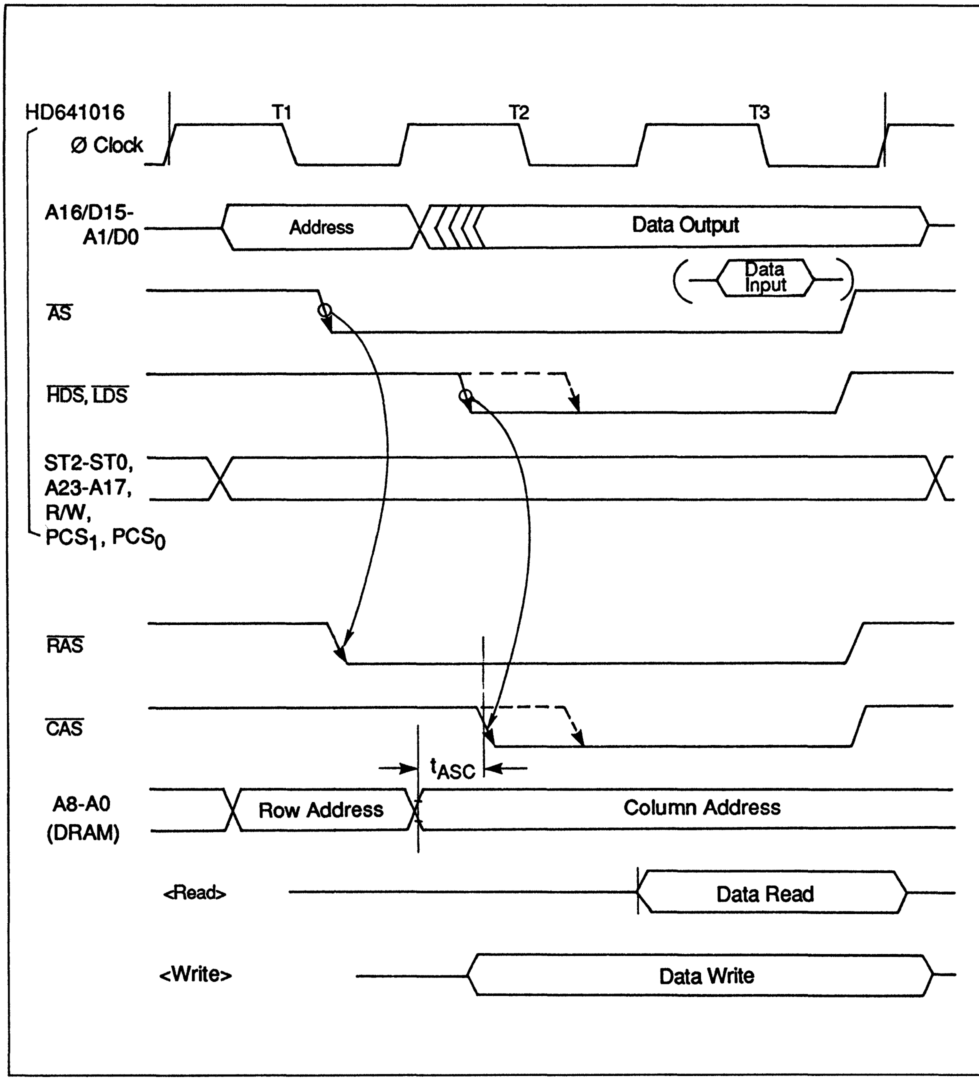


Figure 13-7. DRAM Interface Circuit Timing

13.5 DRAM Refresh Controller and Reset

Note that the HD641016 provides two types of reset: power-on reset and manual reset. See "4.11 Reset" for details.

13.5.1 DRAM Refresh Controller Operation at Power-On Reset

- DRAM refresh: executed
- Refresh interval: 32 states
- Tp state insertion: enabled
- TRW state: 3
- Tw state insertion by $\overline{\text{WAIT}}$ pin: enabled
- Limit of refresh request counts: 31
- Refresh address counter : H'000000
- Bus priority: External bus master > Internal DMAC > Refresh controller > CPU
- MCR: H'F0E0

13.5.2 DRAM Refresh Controller Operation at Manual Reset

The DRAM refresh controller is not affected by manual reset and continues operation normally.

13.5.3 DRAM Refresh Controller Operation After the RESET Instruction Execution

The DRAM refresh controller is not affected by the RESET instruction and continues operation normally.

13.6 DRAM Refresh Controller Notes

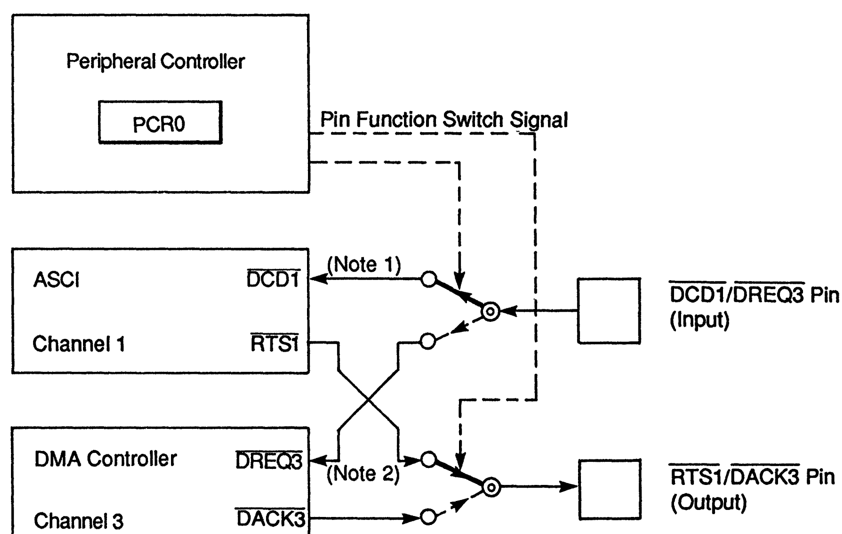
If the refresh counter exceeds the limit when RPL = 1, $\overline{\text{BACK}}$ is forcibly negated. After this, $\overline{\text{BREQ}}$ must be negated, after which the bus master releases bus mastership. See "4.8 Bus Release" for details.

Section 14. Peripheral Controller

14.1 Overview

The peripheral controller selects the function of multiplexed pins: $\overline{\text{DCD1}}$/ $\overline{\text{DREQ3}}$ and $\overline{\text{RTS1}}$/$\overline{\text{DACK3}}$.

Figure 14-1 shows a block diagram of the peripheral controller.



Notes: 1. If the $\overline{\text{DCD1}}$/$\overline{\text{DREQ3}}$ pin functions as the $\overline{\text{DREQ3}}$ pin of DMA Controller 3, $\overline{\text{DCD1}}$ of ASCI1 is pulled low.

2. If the $\overline{\text{DCD1}}$/$\overline{\text{DREQ3}}$ pin functions as the $\overline{\text{DCD1}}$ pin of ASCI1, $\overline{\text{DREQ3}}$ of DMA Controller 3 is pulled high.

Figure 14-1. Wait State Controller Block Diagram

14.2 Peripheral Controller Register

Table 14-1 shows the peripheral control register 0.

Table 14-1. Peripheral Control Register 0

Register Name	Symbol	Address Offset	R/W	Initial Value	Size
Peripheral control register 0	PCR0	H'FF4E	R/W	H'0000	W

14.2.1 Peripheral Control Register 0 (PCR0)

The PCR0 register (Figure 14-2) selects the function of multiplexed pins: $\overline{\mathrm{RTS1}}$/$\overline{\mathrm{DACK3}}$ and $\overline{\mathrm{DCD1}}$/$\overline{\mathrm{DREQ3}}$. Bits 15-13, 11 and 9-0 are reserved and always read as 0s. At reset, PCR0 is initialized to H'0000. PCR0 is not affected by the RESET instruction.

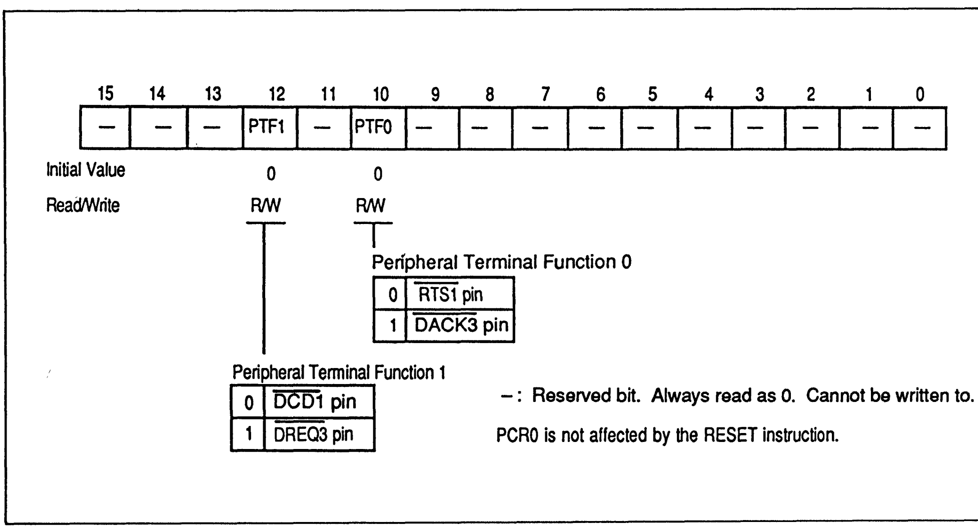


Figure 14-2. Peripheral Control Register 0 (PCR0)

Peripheral Terminal Function 1 (PTF1): PTF1 selects the function of $\overline{DCD1}$/ $\overline{DREQ3}$ multiplexed pin. If PTF1 = 1, the $\overline{DCD1}$/ $\overline{DREQ3}$ pin functions as $\overline{DREQ3}$ of DMA controller channel 3 and $\overline{DCD1}$ of the ASCI channel 1 is pulled low. If PTF1 = 0, the multiplexed pin functions as $\overline{DCD1}$ and $\overline{DREQ3}$ is pulled high.

Peripheral Terminal Function 0 (PTF0): PTF0 selects the function of $\overline{RTS1}$/ $\overline{DACK3}$ multiplexed pin. If PTF0 = 1, the multiplexed pin functions as $\overline{DACK3}$ of DMA controller channel 3. If PTF0 = 0, the multiplexed pin functions as $\overline{RTS1}$ of ASCI channel 1.

14.3 Peripheral Controller Operation

The $\overline{DCD1}$/ $\overline{DREQ3}$ multiplexed pin functions as $\overline{DCD1}$ pin of ASCI channel 1 at reset. If the PTF1 bit of PCR0 is set, the $\overline{DCD1}$/ $\overline{DREQ3}$ multiplexed pin becomes the $\overline{DREQ3}$ pin of DMA controller channel 3 and $\overline{DCD1}$ is pulled low. If the PTF1 bit is cleared, the multiplexed pin functions as the $\overline{DCD1}$ pin and $\overline{DREQ3}$ is pulled high.

The $\overline{RTS1}$/ $\overline{DACK3}$ multiplexed pin functions as $\overline{RTS1}$ of ASCI channel 1 at reset. If the PTF0 bit of PCR0 is set, the multiplexed pin functions as $\overline{DACK3}$ of DMA controller channel 3. If the PTF0 bit is cleared, the multiplexed pin functions as $\overline{RTS1}$ instead.

14.4 Peripheral Controller Operation Notes

The peripheral controller is reset if both $\overline{RES}$ and $\overline{BRTRY}$ pins are asserted low.

Section 15. Low Power Consumption Modes

15.1 Overview

The HD641016 provides two low power consumption modes: sleep and system stop modes.

Table 15-1 shows the bank mode register related to low power consumption modes.

Table 15-1. Bank Mode Register

Register Name	Symbol	R/W	Initial Value	Size
Bank mode register	BMR	R/W	H'A4	B

15.2 Sleep Mode

Executing the SLEEP instruction while the SSTOP bit of BMR is cleared puts the HD641016 in sleep mode. Since the CPU clock stops, this mode reduces power consumption effectively. Sleep mode has the following characteristics:

- Internal clock generator continues operating, Ø and E clocks are outputs.
- The CPU stops operating.
- Internal I/O devices (DMA controller, timer, ASCI, peripheral controller) continue operating.
- DRAM refresh performed if it is programmed.
- $\overline{\text{BREQ}}$ accepted.
- Internal and external ($\overline{\text{NMI}}$, $\overline{\text{IRQ0}}$, $\overline{\text{IRQ1}}$) interrupts accepted.
- Contents of internal RAM and CPU registers maintained.
- ST2-ST0 pins negated low.

Sleep mode can be cancelled by the following methods:

- Reset: Sleep mode is cancelled by a reset exception caused by a power-on reset or manual reset.
- Interrupt: Sleep mode is cancelled and the corresponding interrupt routine begins if an internal or external interrupt is accepted. However, note that sleep mode cannot be cancelled if the requested interrupt is masked by the IPR and SR registers. The NMI interrupt is always accepted and cancels sleep mode. At this time, the SSTOP bit of BMR remains cleared. See "4.7 Interrupt Controller" for details.

15.3 System Stop Mode

Executing the SLEEP instruction while the SSTOP bit of BMR is set puts the HD641016 in system stop mode. Since the CPU and internal I/O devices stop operating, this mode saves more power than sleep mode. System stop mode has the following characteristics:

- Internal clock generator continues operating, Ø and E clocks are outputs.
- The CPU stops operating.
- Internal I/O devices (DMA controller, timer, ASCI, peripheral controller) stop operating. Note that the built-in DMA controller must be stopped before the HD641016 enters system stop mode. See "Section 8. DMA Controller" for details.
- DRAM refresh performed if it is programmed.
- $\overline{\text{BREQ}}$ accepted.
- External ($\overline{\text{NMI}}$, $\overline{\text{IRQ0}}$, $\overline{\text{IRQ1}}$) interrupts accepted.
- Contents of internal RAM and CPU registers maintained.
- ST2-ST0 pins negated low.

System stop mode can be cancelled by the following methods:

- Reset: System stop mode is cancelled by a reset exception caused by a power on reset or manual reset. The CPU begins reset exception processing.
- External interrupt: System stop mode is cancelled and the corresponding interrupt routine begins if an external interrupt is accepted. However, note that system stop mode cannot be cancelled if the requested external interrupt is masked by the IPR and SR registers. The NMI interrupt is always accepted and cancels system stop mode. Note that after system stop mode has been cancelled by an external interrupt, the SSTOP bit of BMR is set.

15.4 Low Power Consumption Mode Transition

Figure 15-1 shows low power consumption mode transition.

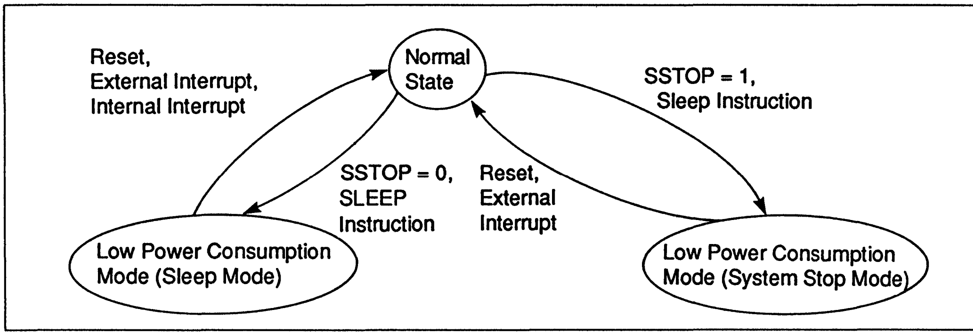


Figure 15-1. Low Power Consumption Mode Transition

15.5 Bank Mode Register (BMR)

The BMR register (Figure 15-2) contains the SSTOP bit related to low power consumption modes.

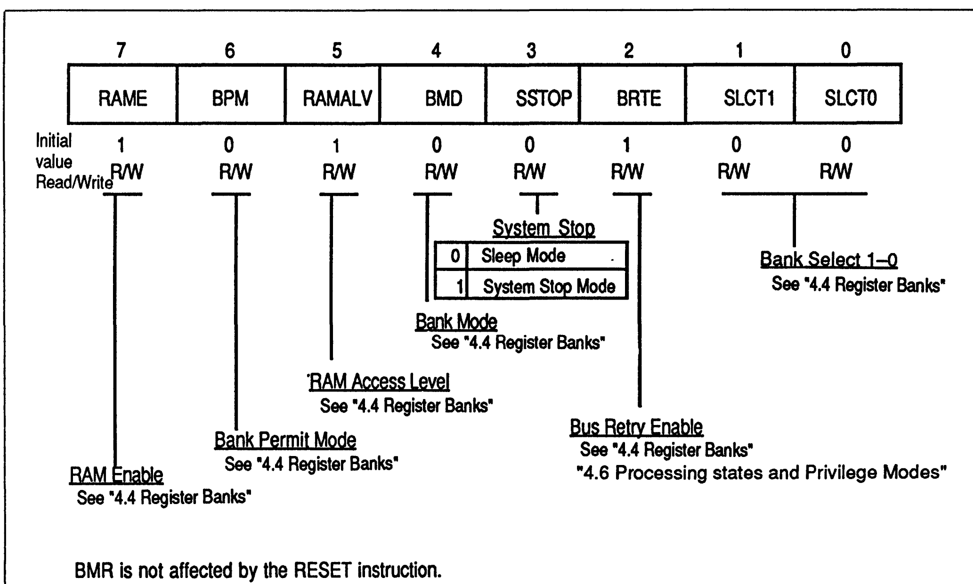


Figure 15-2. Bank Mode Register (BMR)

RAM Enable (RAME): RAME determines whether or not the internal RAM is valid. See "4.4 Register Banks" for details.

Bank Permit Mode (BPM): BPM determines whether or not the bank area can be accessed as memory. See "4.4 Register Banks" for details.

RAM Access Level (RAMALV): RAMALV specifies the access level of internal RAM. See "4.4 Register Banks" for details.

Bank Mode (BMD): BMD determines register bank mode. See "4.4 Register Banks" for details.

System Stop (SSTOP): If the SLEEP instruction is executed while SSTOP = 0, the HD641016 enters the sleep mode. If the SLEEP instruction is executed while SSTOP = 1, the HD641016 enters the system stop mode. This bit is cleared at reset.

Bus Retry Enable (BRTE): BRTE selects the bus retry or the bus error mode. See "4.6 Processing States and Privilege Modes" for details.

Select 1, 0 (SLCT1,0): SLCT1 and SLCT0 determines bank use. See "4.4 Register Banks" for details.

15.6 Comparison between Sleep and System Stop Modes

Table 15-2 summarizes differences between sleep and system stop modes.

Table 15-2. Difference between Sleep and System Stop Modes

Item	Sleep Mode	System Stop Mode
Internal clock oscillator	Operates	Operates
CPU	Stops	Stops
Refresh controller	Operates	Operates
BREQ	Accepted	Accepted
External interrupt	Accepted	Accepted
Internal interrupt	Accepted	Not Accepted (Not generated)
Internal RAM and CPU registers	Maintained	Maintained
Internal I/O (Note)	Operates	Stops
ST2 - ST0	Low	Low

Note: The chip select controller, wait state controller, and DRAM refresh controller continue operating in both sleep and system stop modes. The internal DMA controller, ASCI, timer, and peripheral controller stop operating in system stop mode.

Section 16. Instruction Set

16.1 Addressing Modes

This section discusses the various addressing modes: how they are specified and how the effective addresses are calculated in each.

16.1.1 Register Direct

Mnemonic: Rn

EA Code:

3	4
1 0 0	Rn

Expansion Bytes: None

EA Calculation: Rn contains operand data

16.1.2 Register Indirect

Mnemonic: @Rn or @(disp [: lng], Rn)

EA Code:

1	2	4
0	Sd	Rn

Expansion Bytes:

Sd	Expansion Byte	
00	None	
01	d 8	(1 byte)
10	d 16	(2 bytes)
11	d 32	(4 bytes)

EA Calculation: 0, 1, 2, or 4 extension bytes (depending on Sd, Table 16-1) contain a displacement. The displacement is sign-extended to 32 bits and added to the contents of register Rn. The sum is the effective address. See Figure 16-1.

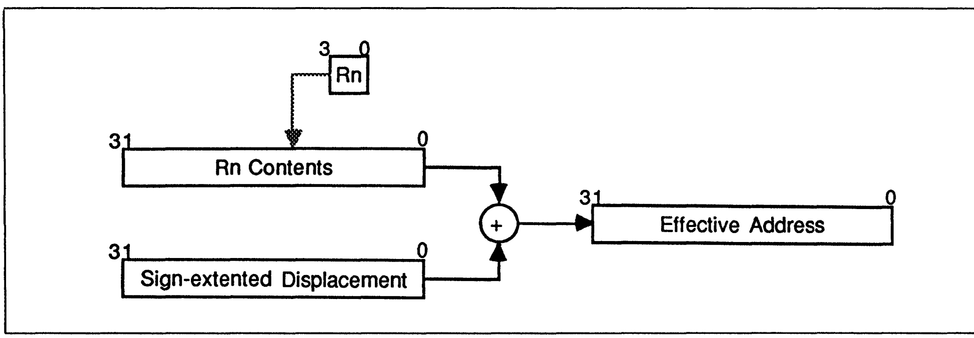


Figure 16-1. Register Indirect Effective Address

Table 16-1. Displacement Size

Sd	Displacement Size
00	No displacement
01	8 bits (1 byte)
10	16 bits (2 bytes)
11	32 bits (4 bytes)

16.1.3 Register Indirect, Auto Increment

Mnemonic: @Rn+

EA Code:

3	4
101	Rn

Expansion Bytes: None

EA Calculation: Rn contains the effective address. After the operation, the contents of Rn are incremented by 1, 2, or 4, depending on the operand size (Table 16-2). See Figure 16-2.

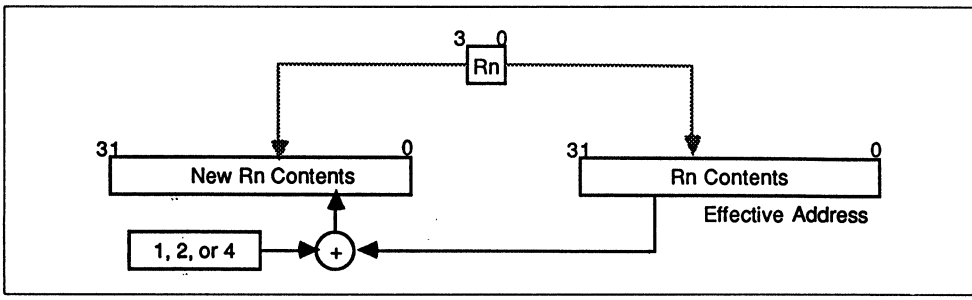


Figure 16-2. Register Indirect, Auto Increment Effective Address

Table 16-2. Auto Increment and Operand Size

Increment	Operand Size
1	Byte(Note)
2	Word
4	Long word

Note: If Rn = R15, increment by 2 for byte or word data, and by 4 for long word data.

16.1.4 Register Indirect, Auto Decrement

Mnemonic: @–Rn

EA Code:

3	4
1 1 0	Rn

Expansion Bytes: None

EA Calculation: Rn contains the effective address. After the operation, the contents of Rn are decremented by 1, 2, or 4, depending on the operand size (Table 16-3). See Figure 16-3.

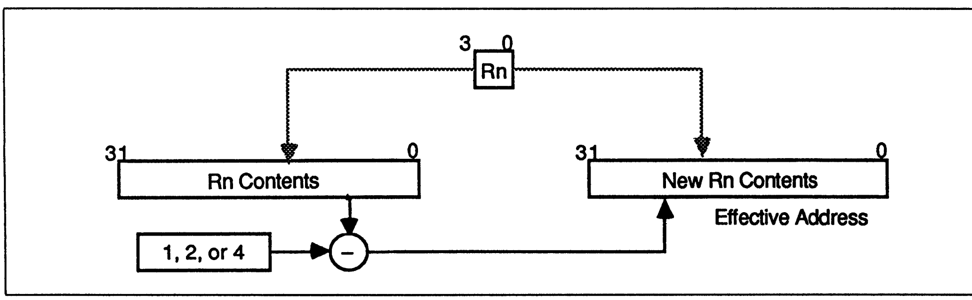


Figure 16-3. Register Indirect, Auto Decrement Effective Address

Table 16-3. Auto Decrement and Operand Size

Decrement	Operand Size
1	Byte(Note)
2	Word
4	Long word

Note: If Rn = R15, decrement by 2 for byte or word data, and by 4 for long word data.

16.1.5 Immediate

Mnemonic: #xxxx [.Sz]

EA Code:

5	2
1 1 1 0 0	Si

Expansion Bytes:

Si	Expansion Byte	
01	Imm 8	(1 byte)
10	Imm 16	(2 bytes)
11	Imm 32	(4 bytes)

Note: Si = 00 is the EA code for current bank addressing mode.

EA Calculation: Data follows the instruction. Si determines the size of the immediate data (Table 16-4). Byte or word immediate data is sign extended to long word.

Table 16-4. Si and Immediate Data Size

Si	Immediate Data Size
01	Byte (8 bits)
10	Word (16 bits)
11	Long word (32 bits)

16.1.6 Absolute Address

Mnemonic: @aaaa [.Sz]

EA Code:

5	2
1 1 1 0 1	Sa

Expansion Bytes:

Sa	Expansion Byte	
01	Abs 8	(1 byte)
10	Abs 16	(2 bytes)
11	Abs 32	(4 bytes)

Note: Sa = 00 is the EA code for previous bank addressing mode.

EA Calculation: The effective address is written to the EA expansion field in advance. Its size depends on Sa (Table 16-5). It is sign extended to 32 bits if required.

Table 16-5. Sa and Absolute Address Size

Sa	Absolute Address Size
01	8 bits (1 byte)
10	16 bits (2 bytes)
11	32 bits (4 bytes)

16.1.7 Register Indirect with Scale

Mnemonic: @Rn * Sf or @(disp [: lng], Rn * Sf)

EA Code:

5	2
1 1 1 1 0	Sd

See Table 16-1 for Sd.

Expansion Bytes:

First Expansion Byte

2	2	4
*	Sf	Rn

*: Don't care

Second Expansion Byte

disp

EA Calculation: The contents of Rn are multiplied by 1, 2, 4, or 8, depending on Sf (Table 16-6). This value is added to the displacement to get the effective address. Sd specifies the displacement size as in the register indirect mode. See Figure 16-4.

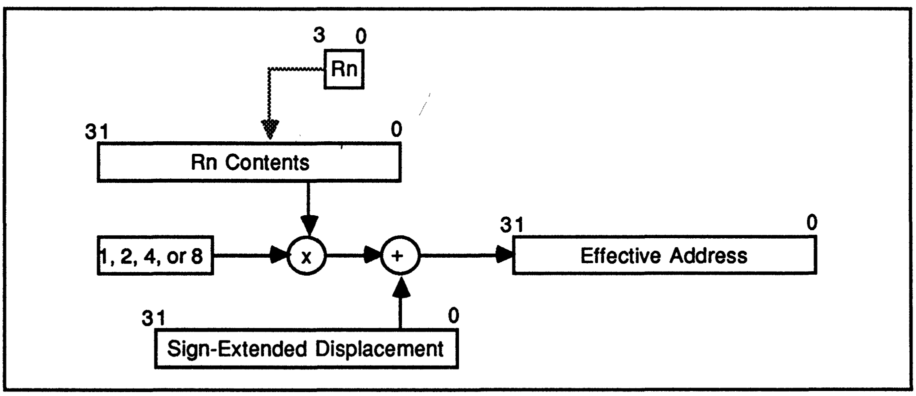


Figure 16-4. Register Indirect with Scale Effective Address

Table 16-6. Sf and Scale Factor

Sf	Scale Factor
00	x 1
01	x 2
10	x 4
11	x 8

16.1.8 Register Indirect with Index

Mnemonic: @ ([disp [:lng],] Xm [.Sz] [*Sf], Rn)

EA Code:

7
1111100

Expansion Bytes:

First Expansion Byte

1	1	2	4
0	L	Sd	Rn

Second Expansion Byte

2	2	4
*	Sf	Xm

*: Don't care

Third Expansion Byte

disp

EA Calculation: If L = 0, contents of Xm are sign extended from word to long word. If L = 1, the contents of Xm are treated as a long word. Then, the contents of Xm are multiplied by 1, 2, 4, or 8, depending on Sf. The result is added to the contents of Rn and a displacement. Sd specifies the length of the displacement as in the register indirect mode. See Figure 16-5.

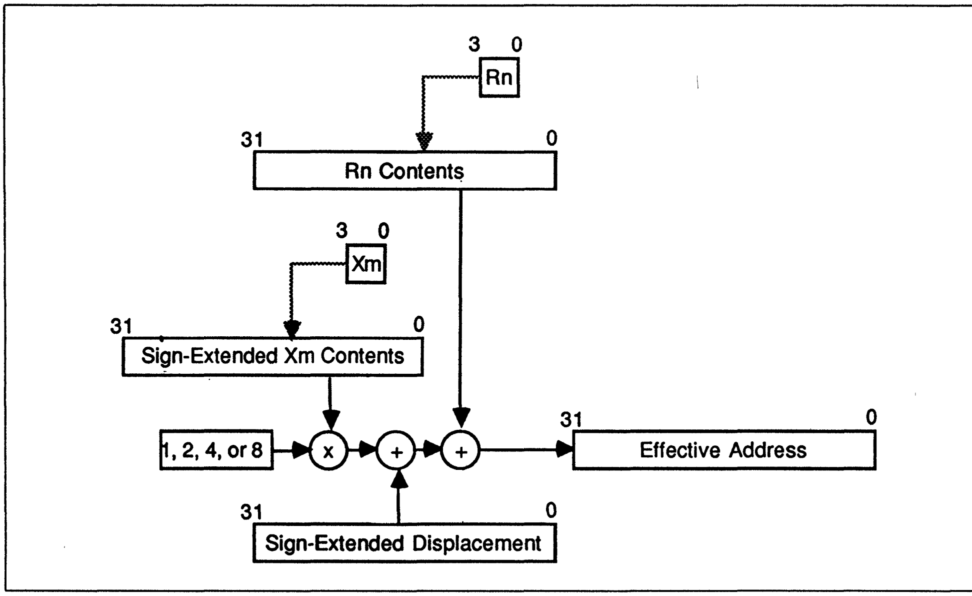


Figure 16-5. Register Indirect with Index Effective Address

16.1.9 PC Relative with Index

Mnemonic: @ ([disp [:lng],] Xm [.Sz] [*Sf], PC])

EA Code:

7
1111101

Expansion Bytes:

First Expansion Byte

1	1	2	4
0	L	Sd	*

*: Don't care

Second Expansion Byte

2	2	4
*	Sf	Xm

*: Don't care

Third Expansion Byte

disp

EA Calculation: If L = 0, contents of Xm are sign extended from word to long word. If L = 1, the contents of Xm are treated as a long word. Then, the contents of Xm are multiplied by 1, 2, 4, or 8, depending on Sf. The result is added to the contents of the PC and a displacement. Sd specifies the length of the displacement as in the register indirect mode. See Figure 16-6.

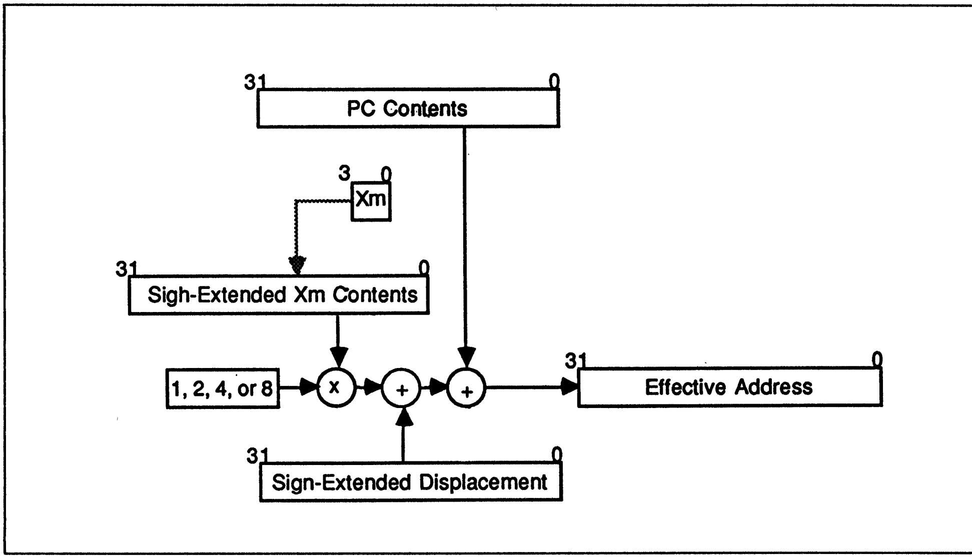


Figure 16-6. PC Relative with Index Effective Address

16.1.10 PC Relative

Mnemonic: @PC or @(disp [:lng], PC)

EA Code:

7

1111101

Expansion Bytes:

First Expansion Byte

2	2	4
10	Sd	*

*: Don't care

Second Expansion Byte

disp

EA Calculation: The contents of the sign-extended displacement are added to the PC. Sd specifies the displacement size as in the register indirect mode. See Figure 16-7.

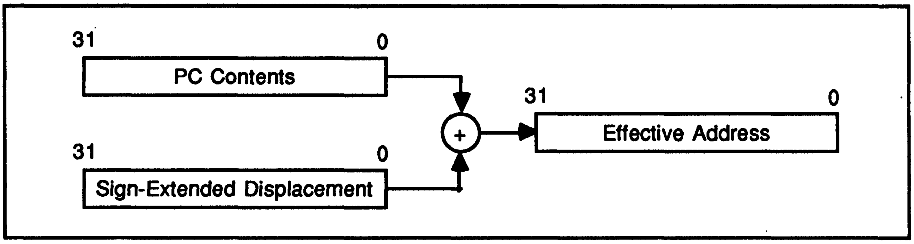


Figure 16-7. PC Relative Effective Address

16.1.11 Register Double Indirect

Mnemonic: @@Rn or @(ds2, @(ds1, Rn))

EA Code:

7
1 1 1 1 1 1 0

Expansion Bytes:

First Expansion Byte

2	2	4
S2	S1	Rn

Second Expansion Byte

ds1

Third Expansion Byte

ds2

EA Calculation: The contents of Rn and displacement ds1 are added. The contents of the memory location addressed by the sum is added to ds2. This second sum is the effective address of the operand. S1 and S2 specify the size of ds1 and ds2 as Ssd does in register indirect mode. However, only certain combinations are legally allowed (Table 16-7). If another combination is specified, an illegal instruction exception processing occurs. See Figure 16-8.

Table 16-7. Allowable S1 and S2 Combinations

S1	S2	ds1 Size	ds2 Size
01	01	1 byte	1 byte
01	11	1 byte	4 bytes
11	01	4 bytes	1 bytes
11	11	4 bytes	4 bytes

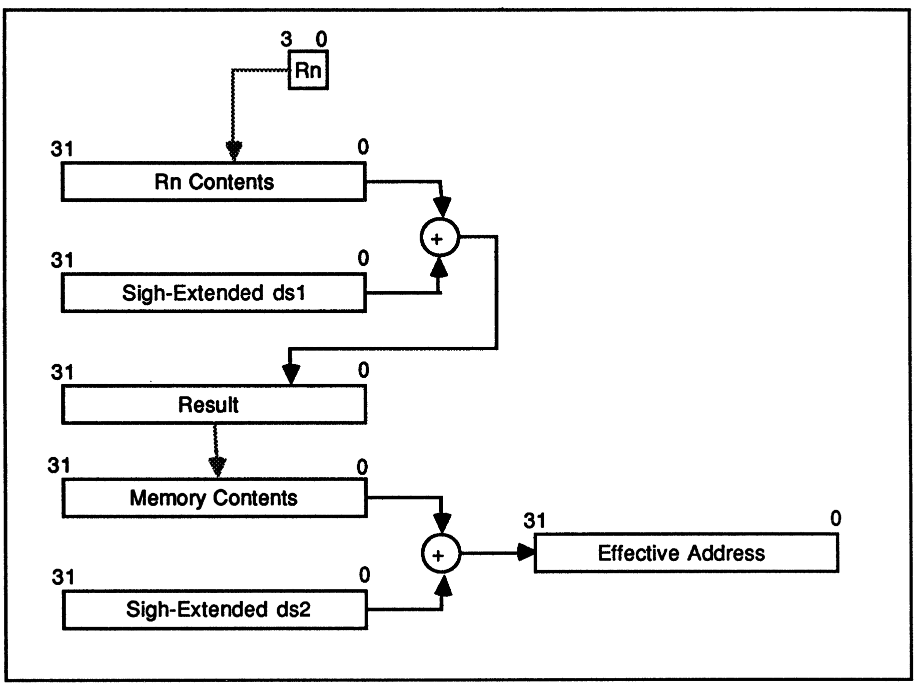


Figure 16-8. Register Double Indirect Effective Address

16.1.12 Current Bank

Mnemonic: <CRn>

EA Code:

7
1 1 1 0 0 0 0

Expansion Bytes: None

EA Calculation: Any EA mode can be used following the <CRn> prefix. Current bank registers will be used instead of the global bank registers. However, in @(Xm, Rn) mode, the global bank register is used for Xm.

16.1.13 Previous Bank

Mnemonic: <PRn>

EA Code:

7
1 1 1 0 1 0 0

Expansion Bytes: None

EA Calculation: Any EA mode can be used following the <PRn> prefix. The previous bank registers (CBNR – 1) will be used instead of the global bank registers. However, in @(Xm, Rn) mode, the global bank register is used for Xm.

If more than one <PRn> is used, only the last is valid. Thus the combination <PRn> <PRn> uses CBNR – 1, not CBNR – 2.

16.2 Instruction Set Description

This section describes the symbols and abbreviations used in the instruction set summary.

16.2.1 Assembler Format

Format codes: Five format codes are provided. These format codes are omittable.

: G : General format of instruction

: Q : Quick format of instruction

: R : Register format of instruction

: RQ : Register quick format of instruction

: d : Displacement size (8, 16, or 32 bits)

<EAs> : Source operand

<EAd> : Destination operand

#: Number : "#" indicates "immediate data" and ": number" indicates the number of bits.
"# : Number" does not indicate assembler syntax.

Rn : Global bank register. Smaller case alphabetical character may be appended for further identification.

/: "/" is followed by option. Options are:

i) Options for direction
/F: positive direction
/B: negative direction

ii) Options for condition
/cc: condition

(<register list>): specifies general purpose registers to be used for an instruction as shown below.
e.g. (<R0, R3>) specifies R0 and R3.
(<R0, R3-R10, R14>) specifies R0, R3-R10, and R14.

16.2.2 Mnemonics

PC: Program counter

SP: Stack pointer

USP: User stack pointer

SSP: Supervisor stack pointer

CR: CR register

SR: Status register

CCR: Condition code register

CX: The CX bit of condition code register

N: The N bit of condition code register

Z: The Z bit of condition code register

V: The V bit of condition code register

C: The C bit of condition code register

disp: displacement

Rn: Global bank register(Note)

Temp.: Internal temporary register

Note: Smaller case alphabetical character may be appended for further identification.

E.g. Rnc, Rnf

16.2.3 Operation

->: Transfer

<->: Exchange

+: Add

–: Subtract

*: Multiply

/: Divide

∧: Logical AND

∨: Logical OR

⊕: Logical exclusive OR

~: Take logical complement

( ): The contents of the EA

@Rn+: Register indirect with auto-increment

@-Rn: Register indirect with auto-decrement

16.2.4 Instruction Fields in Instruction Format

Op: Opcode field

S (1 bit): Operand size field

Sz (2 bits): Operand size field

Sd: Displacement size field

Si: Immediate data size field

EA: Effective address field

EAs: Source effective address field

EAd: Destination effective address field

Rns: Source global bank register field

Rnd: Destination global bank register field

Rn: Global bank register field. Smaller case alphabetical character may be appended for further identification.

CR: CR code field

cc: Condition field

Imm: Immediate data field

FNC: Function field

TYP: Type field

16.2.5 Condition Codes

The flags of the condition register (CCR), CX, N, Z, V, C, are either set to 1 or 0, left unchanged, or set according to the results. For most instructions, they are set normally; that is, N indicates a negative result, Z indicates a zero, V indicates an overflow, C indicates a carry, and X is the same as C. If a flag is affected differently, an explanation will be given with the instruction.

1: Set to 1

0: Cleared to 0

U: Unaffected

S: Set normally, that is:

- X: Set same as C bit.
- N: Set if result is negative (MSB = 1), cleared otherwise.
- Z: Set if result is zero, cleared otherwise.
- V: Set if result overflows, cleared otherwise.
- C: Set if carry is generated, cleared otherwise.

S*1 : See explanation.

16.2.6 Operand Size

Sz in the opcode indicates the operand size as follows:

Sz	Operand Size
00	Byte
01	Word
10	Long word
11	Reserved for future expansion. Cannot be used.

16.2.7 Conditions

The conditions in Table 16-8 can be used for the Bcc:G, SCB, SCMP, SET, SSCH and TRAP instructions.

16.2.8 Available EAs

: Effective addressing mode available
: Effective addressing mode not available

Table 16-8. Conditions

Mnemonic	Condition	Code	Conditional Expression	Signed Condition Code
CC, HS	Carry clear	0100	$\overline{C}$	x ≥ y Unsigned
CS, LO	Carry set	0101	C	x < y Unsigned
NE	Not equal	0110	$\overline{Z}$	x ≠ y Unsigned and signed
EQ	Equal	0111	Z	x = y Unsigned and signed
GE	Greater or equal	1100	$N \cdot V + \overline{N} \cdot \overline{V}$	x ≥ y Signed
LT	Less than	1101	$N \cdot \overline{V} + \overline{N} \cdot V$	x < y Signed
GT	Greater than	1110	$N \cdot V \cdot \overline{Z} + \overline{N} \cdot \overline{V} \cdot \overline{Z}$	x > y Signed
LE	Less or equal	1111	$Z + N \cdot \overline{V} + \overline{N} \cdot V$	x ≤ y Signed
HI	High	0010	$\overline{C} \cdot \overline{Z}$	x > y Unsigned
LS	Lower or same	0011	C + Z	x ≤ y Unsigned
PL	Plus	1010	$\overline{N}$	
MI	Minus	1011	N	
VC	Overflow clear	1000	$\overline{V}$	
VS	Overflow set	1001	V	
T	Always true	0000	1	
F	Always false	0001	0	

Note: N: N (negative) bit in condition code

Z: Z (zero)bit in condition code

V: V (overflow) bit in condition code

C: C (carry) bit in condition code

x: Destination

y: Source

16.2.9 CR Registers

CR registers used for instructions are specified as in Table 16-9. The operation size depends on the specified CR register.

Table 16-9. CR Registers and CR Codes

CR Register	b_7 .... b_0	Operation Size	Notes
CCR	00100000	Word	
VBNR	00000001	Byte	
CBNR	01000000	Long word	
BSP	01000001	Long word	
BMR	10000000	Byte	These registers are used for privileged instructions.
GBNR	10000001	Byte	
SR	10100000	Word	
EBR	11000000	Long word	
RBR	11000001	Long word	
USP	11000010	Long word	
IBR	11000011	Long word	

16.2.10 2-Byte Opcode Instructions

Shift, bit manipulation, string, multiple, and division instructions are 2-byte opcode instructions. These 2-byte opcode instructions are categorized as SFT, BIT, STRING, MUL, and DIV on the opcode map by the first opcode.

The second opcode, in turn, defines the instruction mnemonics above as in Figure 16-9. (EA and register specifications after the third byte are omitted.)

First opcode	Second opcode	Mnemonic on opcode map
0 1 1 0 0 0 | Sz	MODE | * | Rn	SFT
0 1 1 0 0 1 | Sz	MODE | Imm	

MODE	000	001	010	011	100	101	110	111
Mnemonic	SHAR	SHLR	ROTR	ROTXR	SHAL	SHLL	ROTL	ROTXL

First opcode	Second opcode	Mnemonic on opcode map
0 1 1 0 1 0 | Sz	* | MODE | * | Rn	BIT
0 1 1 0 1 1 | Sz	* | MODE | Imm	

MODE	00	01	10	11
Mnemonic	BSET	BNOT	BCLR	BTST

First opcode	Second opcode	Mnemonic on opcode map
1 0 0 1 0 1 | Sz	MODE | Rnc	STRING

MODE	0000	0100	1000	1100	0010	0101	1010	1101
Mnemonic	SSTR		SSCH		SMOV		SCMP	

Figure 16-9. Two-Byte Instruction Codes

First opcode	Second opcode	Mnemonic on opcode map
11101110	MODE | *	MUL

MODE	0000	0001	0100	0101
Mnemonic	MULXS		MULXU	

First opcode	Second opcode	Mnemonic on opcode map
11101111	MODE | *	DIV

MODE	0000	0001	0100	0101
Mnemonic	DIVXS		DIVXU	

Notes: * = don't care

An illegal exception occurs if codes other than above are specified for the second opcode.

Figure 16-9. Two-Byte Instruction Codes (cont.)

16.2.11 Instruction Index

Index in Alphabetical Order

Index by Instruction Type

Page

Arithmetic Operation

Logic Operation

16.3 Instruction Set Details

16.3.1 ADD:G (Add Binary)

Operation: (Destination) + (Source) —> (Destination)

Assembler Format: ADD:G <EAs>, <EAd>

Description: Add source operand to destination operand, and load the result to the destination location.

Operand Size: Byte, word, long word.

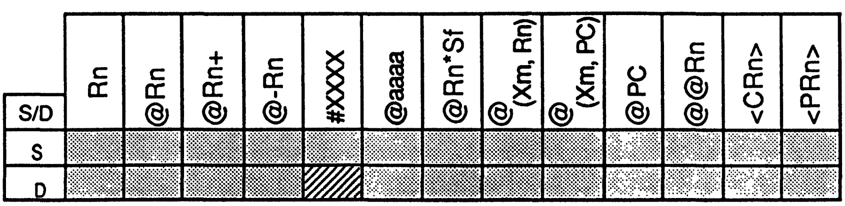
Instruction Format:

	2		
000000	Sz	EAs	EAd

Condition Codes:

CX	N	Z	V	C
S	S	S	S	S

Available EA:



16.3.2 ADD:Q (Add Quick)

Operation: (Destination) + Immediate Data —> (Destination)

Assembler Format: ADD:Q #:8, <EAd>

Description: Add the sign extended immediate data to destination operand, and load the result to the destination location. Immediate data must be between –128 to +127.

Operand Size: Byte, word, long word

Instruction Format:

	2	8	
000100	Sz	Imm	EAd

Condition Codes:

CX	N	Z	V	C
S	S	S	S	S

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.3 ADD:R (Add Register)

Operation: Rnd + Rns $\longrightarrow$ Rnd

Assembler Format: ADD:R Rns, Rnd

Description: Add source operand to destination operand, and load the result to the destination location. A global bank register can be specified as source and destination.

Operand Size: Byte, word, long word

Instruction Format:

	2	4	4
0 0 1 0 0 0	Sz	Rns	Rnd

Condition Codes:

CX	N	Z	V	C
S	S	S	S	S

Available EA: None

16.3.4 ADD:RQ (Add Register Quick)

Operation: Rnd + Immediate Data —> Rnd

Assembler Format: ADD:RQ #:4, Rnd

Description: Add the zero-extended immediate data to destination operand, and load the result to the destination location. A global bank register can be specified as destination. Immediate data must be between 0 and 15.

Operand Size: Byte, word, long word

Instruction Format:

	2	4	4
001100	Sz	Rnd	Imm

Condition Codes:

CX	N	Z	V	C
S	S	S	S	S

Available EA: None

16.3.5 ADDS (Add with Sign Extension)

Operation: (Destination) + (Source) —> (Destination)

Assembler Format: ADDS <EAs>, <EAd>

Description: Add source operand to destination operand, and load the result to the destination location. Byte or word operands are sign extended to long word (32 bits) prior to addition. The destination operand is always accessed as a long word. The result is also stored as a long word.

Operand Size: Byte, word, long word

Instruction Format:

	2		
0 1 0 0 0 0	Sz	EAs	EAd

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.6 ADDX (Add with CX Flag)

Operation: (Destination) + (Source) + CX —> (Destination)

Assembler Format: ADDX <EAs>, <EAd>

Description: Add source operand to destination operand with CX bit, and load the result to the destination location.

Operand Size: Byte, word, long word.

Instruction Format:

	2		
0 1 0 1 0 0	Sz	EAs	EAd

Condition Codes:

CX	N	Z	V	C
S	S	S*1	S	S

*1: Z set if the result is zero and Z = 1 prior to the operation, unaffected if the result is 0 and Z = 0 prior to the operation, cleared otherwise.

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.7 AND (AND Logical)

Operation: (Destination) ∧ (Source) —> (Destination)

Assembler Format: AND <EAs>, <EAd>

Description: AND source operand with destination operand, and load the result to the destination location.

Operand Size: Byte, word, long word

Instruction Format:

	2		
100000	Sz	EAs	EAd

Condition Codes:

CX	N	Z	V	C
U	S	S	0	0

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.8 ANDC (AND CR Register)

Operation: (CR) Λ (Source) $\longrightarrow$ CR

Assembler Format: ANDC <EAs>, CR

Description: AND source operand with CR register (CR), and store the result in the CR register. Table 16-10 shows CR registers and CR codes.

Notes: 1. If BMR, GBNR, EBR, SR, RBR, USP, or IBR is specified as a CR register, ANDC is executed as a privileged instruction.

2. If the CR field specifies a CR code not shown in Table 16-10, the source is ANDed with 0. This result is not stored in CR. The condition codes change according to the source operand size specified by bits 6-5 in the CR field. If bits 6-5 =11, the source operand is accessed as a long word.

3. If this instruction is executed, the prefetch queue is reset and instructions following this instruction are fetched again.

4. No interrupt can be accepted immediately after the ANDC instruction cycle.

Operand Size: Same as CR

Instruction Format:

	8	
11111000	CR	EAs
	b_7, b_6, $b_5 \ldots b_0$	

Table 16-10. CR Registers CR Codes

CR	b_7..........b_0	Operand Size
CCR	0010 0000	Byte
VBNR	0000 0001	Byte
CBNR	0100 0000	Long word
BSP	0100 0001	Long word
BMR	1000 0000	Byte
GBNR	1000 0001	Byte
SR	1010 0000	Word
EBR	1100 0000	Long word
RBR	1100 0001	Long word
USP	1100 0010	Long word
IBR	1100 0011	Long word

Condition Codes:

CR ≠ CCR and CR ≠ SR:

CX	N	Z	V	C
S	S	S	S	S

CR = CCR or CR = SR:

CX	N	Z	V	C
S*1	S*1	S*1	S*1	S*1

*1: CX same as bit 4 of result
N same as bit 3 of result
Z same as bit 2 of result
V same as bit 1 of result
C same as bit 0 of result

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.9 Bcc:G (Branch According to Condition Code)

Operation: If cc then PC + disp —> PC

Assembler Format: Bcc:G disp

Description: If the specified condition is true, branch to the target label by adding its displacement to the PC. The displacement disp is the distance in bytes from the end of the instruction to the target label. Note that byte or word size displacement is sign-extended to long word. If the condition is false, the next instruction is executed.

Table 16-11 shows the condition codes and their mnemonics. Write the corresponding mnemonic in place of "cc" in the instruction. For example, if the branch condition is "branch if equal", write "BEQ:G".

Table 16-11. Conditions

Mnemonic	Condition	Code	Conditional expression	Signed condition code
CC, HS	Carry clear	0100	$\overline{C}$	x ≧ y Unsigned
CS, LO	Carry set	0101	C	x < y Unsigned
NE	Not equal	0110	$\overline{Z}$	x ≠ y Unsigned and signed
EQ	Equal	0111	Z	x = y Unsigned and signed
GE	Greater or equal	1100	$N \bullet V + \overline{N} \bullet \overline{V}$	x ≧ y Signed
LT	Less than	1101	$N \bullet \overline{V} + \overline{N} \bullet V$	x < y Signed
GT	Greater than	1110	$N \bullet V \bullet \overline{Z} + \overline{N} \bullet \overline{V} \bullet \overline{Z}$	x > y Signed
LE	Less or equal	1111	$Z + N \bullet \overline{V} + \overline{N} \bullet V$	x ≦ y Signed
HI	High	0010	$\overline{C} \bullet \overline{Z}$	x > y Unsigned
LS	Lower or same	0011	$C + Z$	x ≦ y Unsigned
PL	Plus	1010	$\overline{N}$	
MI	Minus	1011	N	
VC	Overflow clear	1000	$\overline{V}$	
VS	Overflow set	1001	V	
T	Always true	0000	1	
F	Always false	0001	0	

Note: N, Z, V, C = Condition codes

x = Destination

y = Source

Operand Size: None

Instruction Format:

	2	4	4	
101001	Sd	*	cc	disp

*: Don't care

Sd specifies the displacement size as follows.

Sd	disp Size
00	Byte
01	Word
10	Long word

Note that byte or word displacements are sign extended to long word.

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.10 BCLR (Bit Test and Clear)

Operation: ~(<bit number> of <destination>) —> Z
0 —> (<bit number> of <destination>)

Assembler Format: BCLR Rn <EAd> or BCLR #:5, <EAd>

Description: Test a bit of destination operand, and set Z if it is 0. Clear Z if it is 1. After the test, clear the bit to 0.

The bit number is determined in the following ways:

1. Dynamic: The bit number is specified by the lower i bits of register Rn (global bank register). The remaining upper bits are ignored.

2. Static: The bit number is specified by the lower i bits of the immediate data. The remaining upper bits are ignored.

The following table shows Sz and i relationships.

Sz	i
00 (Byte)	3
01 (Word)	4
10 (Long word)	5

Operand Size: Byte, word, long word

Instruction Format:

Dynamic:

	2	1		1	4		
0 1 1 0 1 0	Sz	*	1 0	*	Rn	EAd	*: Don't care

Static:

	2	1		5		
0 1 1 0 1 1	Sz	*	1 0	Imm	EAd	*: Don't care

Condition Codes:

CX	N	Z	V	C
U	U	S*1	U	U

*1: Z set if the bit is 0, cleared otherwise

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.11 BEQ (Branch If Equal)

Operation: If EQ then PC + disp —> PC

Assembler Format: BEQ disp

Description: If Z is 1, branch to the target label by adding its displacement to the PC. If Z is 0, execute the next instruction. The PC contains the start address of the next instruction. Note that byte or word size displacements are sign-extended to long word.

Operand Size: None

Instruction Format:

	2	
1 0 1 0 0 0	Sz	disp

Sd	**disp Size**
00	Byte
01	Word
10	Long word

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.12 BFEXT (Bit Field Data Extract)

Operation: <Bit field> of (Source) —> (Destination)

Assembler Format: BFEXT Rnb, Rnf, <EAs>, <EAd>

Description: Fetch source operand as a long word operand and then extract bit field data from the source operand depending on the bit position and bit field length specified by the Rnb and Rnf fields. Then, zero-extend the bit field to word (16 bits) and load the result to the lower word of the destination. See Figures 16-10 and 16-11.

Note that the Rnb and Rnf fields specify global bank registers.

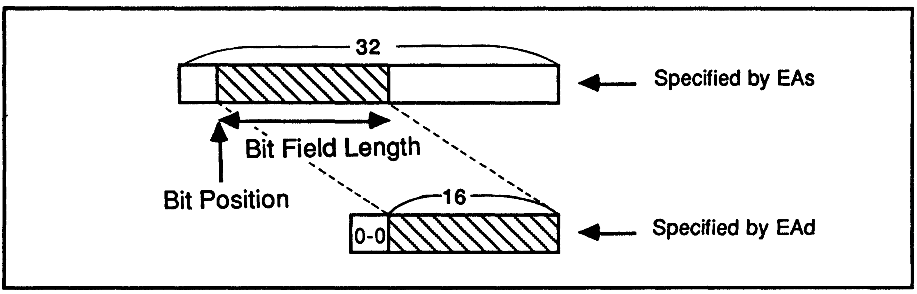


Figure 16-10. BFEXT Operation

When Bit position + Bit field length > 32, BFEXT operates as follows:

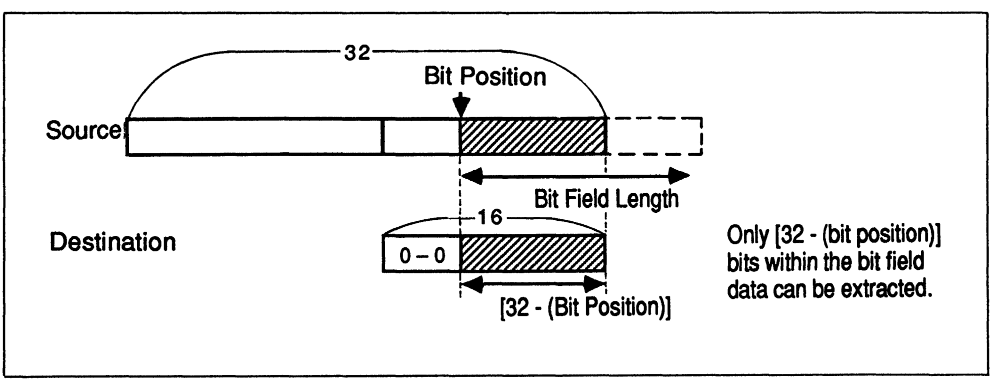


Figure 16-11. BFEXT Operation When Rnb + Rnf > 32

Operand size: Variable bit length up to 16 bits

Instruction Format:

	4	4		
11010100	Rnb	Rnf	EAs	EAd

Rnb: Specifies a global bank register which specifies bit position. Only the lower 5 bits of register Rnb are valid.

Rnf: Specifies a global bank register which specifies bit field length. Only the lower 4 bits of register Rnf are valid. If Rnf = 0, bit field length is specified as 16 bits.

Condition Codes:

CX	N	Z	V	C
U	S*1	S*1	0	0

*1: N set if the MSB of the bit field is 1, cleared otherwise.
Z set if the bit field is zero, cleared otherwise.

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.13 BFINS (Bit Field Data Insert)

Operation: <Bit field> of (Source) —> <Bit field> of (Destination)

Assembler Format: BFINS Rnb, Rnf, <EAs>, <EAd>

Description: Insert bit field data from the source operand to the destination location using bit position. At this time, the bit field data is lower bits (bit field length) of the source operand.

The source and destination operands are accessed as a word and a long word, respectively. Bit field length and bit position are specified by global bank registers specified in the Rnf and Rnd fields, respectively. See Figures 16-12 and 16-13.

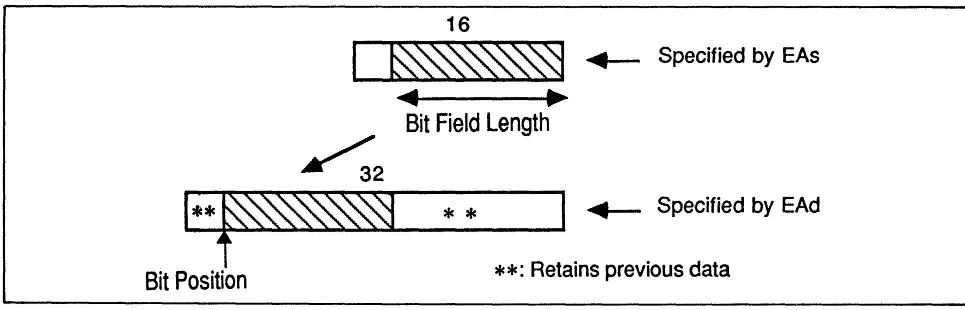


Figure 16-12. BFINS Operation

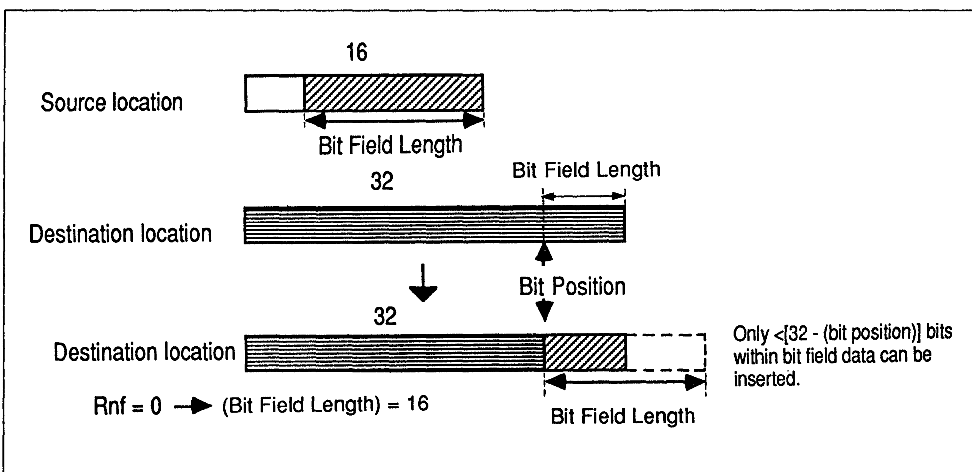


Figure 16-13. BFINS Operation When Rnb + Rnf > 32

Operand Size: Variable bit length up to 16 bits

Instruction Format:

	4	4		
11010101	Rnb	Rnf	EAs	EAd

Rnb: Specifies a global bank register which specifies bit position. Only the lower 5 bits of register Rnb are valid.

Rnf: Specifies a global bank register which specifies bit field length. Only the lower 4 bits of register Rnf are valid. If Rnf = 0, bit field length is specified as 16 bits.

Condition Codes:

CX	N	Z	V	C
U	S*1	S*1	0	0

*1: N set if the MSB of the bit field is 1, cleared otherwise.
Z set if the bit field is zero, cleared otherwise.

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.14 BFMOV (Bit Field Data Move)

Operation: Repeat Y times
X bytes of (Source) —> <Bit field> of (Destination)

Assembler Format: BFMOV Rnx, Rnb, Rny, Rno, Rns, Rnd

Description: Move X bytes of data in the location specified by register Rns into the bit field specified by the destination and bit position.

Then add Rnd to Rno and move X bytes of data into the next sequential field. Repeat this operation Y times. The data length is specified by Rnx. The number of moves is specified by Rny. See Figure 16-14. Rnx, Rnb, Rny, Rno, Rns, and Rnd specify global bank registers.

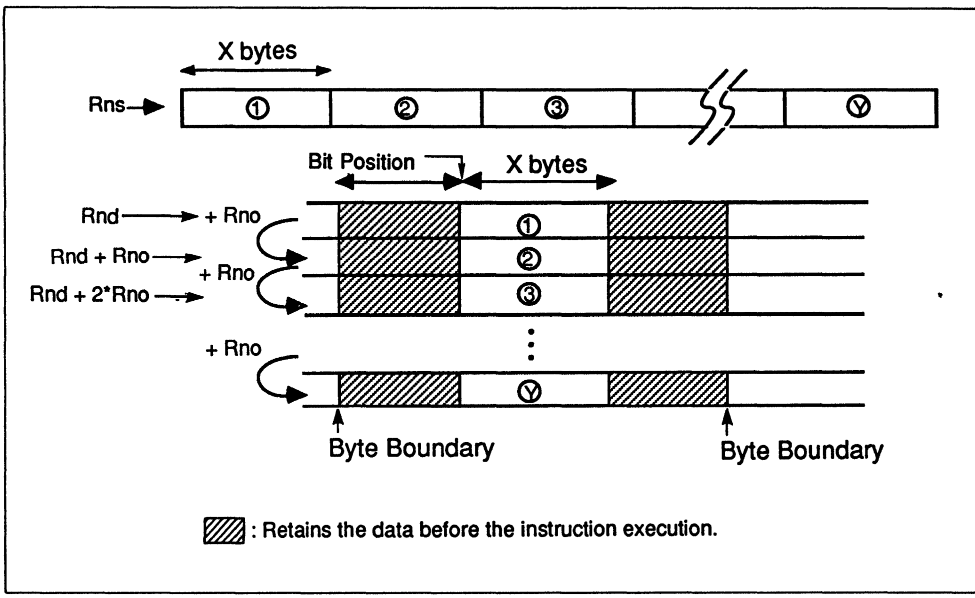


Figure 16-14. BFMOV Operation

The CPU checks for interrupt requests whenever X bytes data transfer is performed. If there is an interrupt request, the CPU stops BFMOV execution and puts the current register status on the stack. At this time, the PC to be stacked contains the start address of BFMOV. It then processes the interrupt exception. After interrupt servicing, the CPU continues the BFMOV instruction. Tables 16-12 and 16-13 show the register values when an interrupt is accepted and when BFMOV execution completes.

Table 16-12. Register Values When an Interrupt is Accepted

Register Name	Register Value
Rnx	Unchanged
Rnb	Unchanged
Rny	(Initial Rny) – (X bytes transfer count)
Rno	Unchanged
Rnx	(Initial Rns) + X * (X-bytes transfer count)
Rnd	(Initial Rnd) + Rno * (X-bytes transfer count)

Table 16-13. Register Values When BFMOV Execution Terminates

Register Name	Register Value
Rnx	Unchanged
Rnb	Unchanged
Rny	Undefined
Rno	Unchanged
Rns	(Initial Rns) + X * Y
Rnd	(Initial Rnd) + Rno * Y

Note that Rns, Rnb, Rny, Rnx, and Rnd must not overlap.

Operand Size: X-byte unit

Instruction Format:

	4	4	4	4	4	4
11010111	Rnx	Rnb	Rny	Rno	Rns	Rnd

Rnx: Specifies a global bank register which specifies the number of bytes X. (Lower 16 bits are valid.)

Rnb: Specifies a global bank register which specifies bit position in the destination. (Lower 4 bits are valid.)

Rny: Specifies a global bank register which specifies the number of movements Y. (Lower 16 bits are valid.)

Rno: Specifies a global bank register which specifies data to be added to Rnd after an X-byte transfer.

Rns: Specifies a global bank register which specifies the source.

Rnd: Specifies a global bank register which specifies the destination.

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.15 BFSCH (Bit Field Data Search)

Operation: <Bit number of first 1> of {<Bit field> of (Source)} —> (Destination)

Assembler Format: BFSCH Rnb, Rnf, <EAs>, <EAd>

Description: Extract bit field data from the source operand using bit position and bit field length and set condition codes. Then search for the first "1" bit from the lower to upper bit position in the bit field. Load the bit position of the first (least significant) "1" bit into the destination location. Bits in a bit field are numbered from MSB (0) to LSB. If there is no "1" bit, load the next bit position (bit position + bit field length + 1). See Figures 16-15 and 16-16.

Rnb and Rnf specify global bank registers.

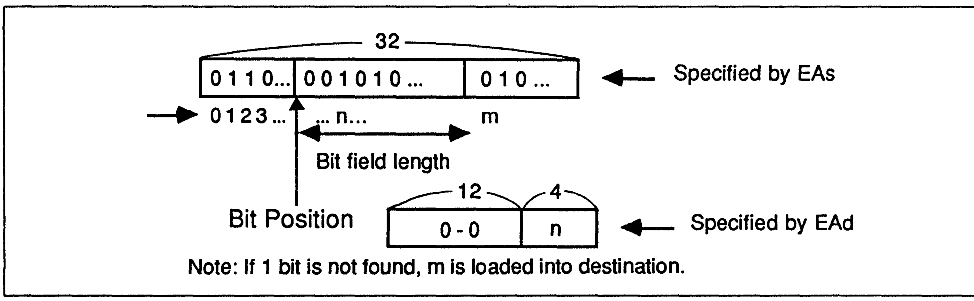


Figure 16-15. BFSCH Operation

When bit position + bit field length >32, BFSCH operates as shown in Figure 16-16.

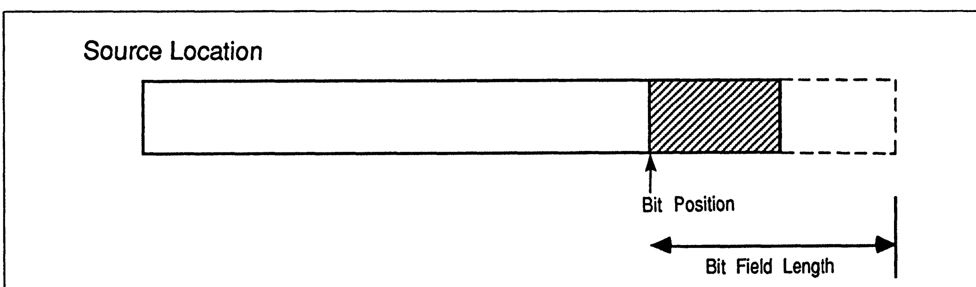


Figure 16-16. BFSCH Operation When Rnb + Rnf > 32

When Rnb + Rnf > 32, only 32 – Rnb bits in the specified bit field can be searched. If a "1" bit is found, load the bit position. Otherwise, return the value (Rnb + Rnf) (> 32).

Operand Size: Variable bit length up to 16 bits

Instruction Format:

	4	4		
11010110	Rnb	Rnf	EAs	EAd

Rnb: Specifies a global bank register which specifies bit position. Only the lower 5 bits of register Rnb are valid.

Rnf: Specifies a global bank register which specifies bit field length. Only the lower 4 bits of register Rnf are valid. If Rnf = 0, bit field length is specified as 16 bits.

Condition Codes:

CX	N	Z	V	C
U	S*1	S*1	0	0

*1: N set if the MSB of the bit field is 1, cleared otherwise.
Z set if the bit field is zero, cleared otherwise.

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.16 BNE (Branch If Not Equal)

Operation: If not EQ then PC + disp —> PC

Assembler Format: BNE disp

Description: If Z is cleared to 0, branch to the target label by adding its displacement to the PC. The PC contains the start address of the next instruction. If Z is set to 1, the next instruction is executed.

Operand Size: None

Instruction Format:

	2	
1 0 1 1 0 0	Sd	disp

Sd specifies the disp size as follows.

Sd	disp Size
00	Byte
01	Word
10	Long Word

Note that byte and word displacements are sign-extended to long word.

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.17 BNOT (Bit Test and Not)

Operation: ~(<Bit number> of <Destination>) —> Z
~(<Bit number> of <Destination>) —> (<Bit number> of <Destination>)

Assembler Format: BNOT Rn, <EAd> or BNOT #:5, <EAd>

Description: Test a bit of destination operand, and set Z if it is 0. Clear Z if it is 1. After the test, invert the specified bit.

The bit number is determined in the following ways:

1. Dynamic: The bit number is specified by the lower i bits of register Rn (global bank register). The remaining upper bits are ignored.

2. Static: The bit number is specified by the lower i bits of the immediate data. The remaining upper bits are ignored.

The following table shows the relationship between the Sz field and i.

Sz	i
00 (Byte)	3
01 (Word)	4
10 (Long word)	5

Operand Size: Byte, word, long word

Instruction Format:

Dynamic:

	2	1		1	4	
0 1 1 0 1 0	Sz	*	0 1	*	Rn	EAd

* : Don't care

Static:

	2	1		5	
0 1 1 0 1 1	Sz	*	0 1	Imm	EAd

* : Don't care

Condition Codes:

CX	N	Z	V	C
U	U	S*1	U	U

*1: Z set if the bit is 0, cleared otherwise.

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.18 BRA (Branch Always)

Operation: PC + disp —> PC

Assembler Format: BRA disp

Description: Always branch to the target label by adding its displacement to the PC. The PC contains the start address of the next instruction.

Operand Size: None

Instruction Format:

	2	
100110	Sd	disp

The Sd field specifies the disp size as follows.

Sd	disp Size
00	Byte
01	Word
10	Long word

Note that byte or word displacements are always sign-extended to long word.

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.19 BSET (Bit Test and Set)

Operation: ~(<Bit number> of <Destination>) —> Z
1 —> (<Bit number> of <Destination>)

Assembler Format: BSET Rn, <EAd> or BSET #:5, <EAd>

Description: Test a bit of destination operand, and set Z if it is 0. Clear Z if it is 1. After the test, set the bit to 1.

The bit number is determined in the following ways:

1. Dynamic: The bit number is specified by the lower i bits of register Rn (global bank register). The remaining upper bits are ignored.

2. Static: The bit number is specified by the lower i bits of the immediate data. The remaining upper bits are ignored.

The following table shows the relationship between the Sz field and i.

Sz	i
00 (Byte)	3
01 (Word)	4
10 (Long word)	5

Operand Size: Byte, word, long word

Instruction Format:

Dynamic:

	2		1		1	4		
0 1 1 0 1 0	Sz	*	0 0	*		Rn	EAd	* : Don't care

Static:

	2		1		5		
0 1 1 0 1 1	Sz	*	0 0		Imm	EAd	* : Don't care

Condition Codes:

CX	N	Z	V	C
U	U	S*1	U	U

*1: Z set if the bit is 0, cleared otherwise.

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.20 BSR (Branch to Subroutine)

Operation: PC —> @-SP

PC + disp —> PC

Assembler Format: BSR disp

Description: Push the PC onto the stack, and branch to the target label by adding its displacement disp to the PC. At this time the PC contains the start address of the next instruction. The RTS instruction together with BSR allows return from a subroutine call.
Note that byte or word size displacements are sign-extended to long word.

Operand Size: None

Instruction Format:

	2	
1 0 1 0 1 0	Sd	disp

The Sd field specifies the disp size as follows.

Sd	disp Size
00	Byte
01	Word
10	Long word

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.21 BTST (Bit Test)

Operation: ~(<Bit number> of <Destination>) —> Z

Assembler Format: BTST Rn, <EAd> or BTST #:5, <EAd>

Description: Test a bit of destination operand, and set Z if it is 0. Clear Z if it is 1.

The bit number is determined in the following ways:

1. Dynamic: The bit number is specified by the lower i bits of register Rn (global bank register). The remaining upper bits are ignored.

2. Static: The bit number is specified by the lower i bits of the immediate data. The remaining upper bits are ignored.

The following table shows the relationship between the Sz field and i.

Sz	i
00 (Byte)	3
01 (Word)	4
10 (Long word)	5

Operand Size: Byte, word, long word

Instruction Format:

Dynamic:

	2	1		1	4		
0 1 1 0 1 0	Sz	*	1 1	*	Rn	EAd	*: Don't care

Static:

	2	1		5		
0 1 1 0 1 1	Sz	*	1 1	Imm	EAd	*: Don't care

Condition Codes:

CX	N	Z	V	C
U	U	S*1	U	U

*1: Z set if the bit is 0, cleared otherwise.

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.22 CGBN (Push and Change Global Bank Number)

Operation: GBNR -> @-SSP
specified value -> GBNR
[Register copy]

Assembler Format: CGBN Rn [, register list] or CGBN #:8 [, register list]

Description: Push the current contents of the global bank number register (GBNR) onto the supervisor stack, and change the bank by loading the contents of register Rn (dynamic) or the immediate data (static) into the GBNR. If the R field is cleared, registers are not copied. If R is set, registers are copied according to the register list (16 bits following Rn or immediate data) into the new bank. The lower 2-4 bits of Rn or immediate data are used, depending on the bank mode (Table 16-14).

Rn specifies a global bank register.

Note that this instruction is a priviledge instruction. If this instruction is executed in user mode, a priviledge violation occurs. In addition, if R15 is copied, the USP instead of the SSP is copied.

Table 16-14. Bank Mode

Mode		Rn or Imm
Global	2	Lower 1 bit
	4	Lower 2 bits
	8	Lower 3 bits
	16	Lower 4 bits
Ring		Lower 3 bits

Operand Size: Byte

Instruction Format:

Dynamic:

	1	4	4	16				
1110010	R	*	Rn	R15	R14	• • •	R1	R0

Register List

*: Don't care.

Static:

	1	8	16				
1110011	R	Imm	R15	R14	• • •	R1	R0

Register List

R=0: No register list exists and registers are not copied. (Instruction length = 2 bytes)

R=1: Registers are copied according to the 2-byte register list. (Instruction length = 4 bytes)

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.23 CLR (Clear)

Operation: 0 —> (Destination)

Assembler Format: CLR <EAd>

Description: Clear all bits of the destination operand.

Operand Size: Byte, word, long word

Instruction Format:

	2	
0 0 0 1 0 1	Sz	EAd

Condition Codes:

CX	N	Z	V	C
U	0	1	0	0

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.24 CMP:G (Compare)

Operation: (Destination) – (Source)

Assembler Format: CMP:G <EAs>, <EAd>

Description: Subtract the source operand from the destination operand, and set the condition codes according to the result. Do not change destination contents.

Operand Size: Byte, word, long word

Instruction Format:

	2		
000010	Sz	EAs	EAd

Condition Codes:

CX	N	Z	V	C
U	S	S	S	S

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.25 CMP:Q (Compare Quick)

Operation: (Destination) – Immediate data

Assembler Format: CMP:Q #:8, <EAd>

Description: Subtract the immediate data from the destination operand, and set the condition codes according to the result. Do not change destination contents. The immediate data must be in the range –128 to +127, because it is always 8 bits and is sign-extended to match the destination operation size.

Operand Size: Byte, word, long word

Instruction Format:

	2	8	
000110	Sz	Imm	EAd

Condition Codes:

CX	N	Z	V	C
U	S	S	S	S

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.26 CMP:R (Compare Register)

Operation: Rnd – Rns

Assembler Format: CMP:R Rns, Rnd

Description: Subtract the source operand from the destination operand, and set the condition codes according to the result. Do not change destination contents. The source and destination operands are global bank registers specified by Rns and Rnd, respectively.

Operand Size: Byte, word, long word

Instruction Format:

	2	4	4
0 0 1 0 1 0	Sz	Rns	Rnd

Condition Codes:

CX	N	Z	V	C
U	S	S	S	S

Available EA: None

16.3.27 CMP:RQ (Compare Register Quick)

Operation: Rnd – Immediate data

Assembler Format: CMP:RQ #:4, Rnd

Description: Subtract the immediate data from the destination operand, and set the condition codes according to the result. Do not change destination contents. The immediate data must be in the range –8 to +7, because it is always 4 bits and is sign extended to match the destination operation size. The destination is a global bank register specified by Rnd.

Operand Size: Byte, word, long word

Instruction Format:

	2	4	4
0 0 1 1 1 0	Sz	Rnd	Imm

Condition Codes:

CX	N	Z	V	C
U	S	S	S	S

Available EA: None

16.3.28 CMPS (Compare with Sign Extension)

Operation: (Destination) – (Source)

Assembler Format: CMPS <EAs>, <EAd>

Description: Subtract the source operand from the destination operand, and set the condition codes according to the result. Do not change destination contents. Byte or word operands are always sign-extended to long words before comparison.

Operand Size: Byte, word, long word

Instruction Format:

	2		
010010	Sz	EAs	EAd

Condition Codes:

CX	N	Z	V	C
U	S	S	S	S

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.29 DADD (Decimal ADD with CX Flag)

Operation: $(\text{Destination})_{BCD} + (\text{Source})_{BCD} + (\text{CX}) \longrightarrow (\text{Destination})_{BCD}$

Assembler Format: DADD <EAs>, <EAd>

Description: Add source operand to destination operand with the CX bit, and load the result to the destination location. The addition is performed with BCD arithmetic.

Operand Size: Byte

Instruction Format:

10100011	EAs	EAd

Condition Codes:

CX	N	Z	V	C
S	S*1	S*1	S*1	S

*1 : N undefined.
Z set if the result is zero and Z = 1 prior to the operation, unaffected if the result is zero and Z = 0 prior to the operation, cleared otherwise.
V undefined.

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.30 DCBN (Decrement Current Bank Number)

Operation: CBNR – 1 —> CBNR

if (the lower 3 bits of CBNR = the lower 3 bits of VBNR) and (CBNR $\neq$ 0)

then VBNR – 1 —> VBNR

{Pop registers from stack}

Assembler Format: DCBN

Description: First decrement the CBNR by 1 (switching ring bank). If the lower 3 bits of CBNR = the lower 3 bits of VBNR and CBNR $\neq$ 0, decrement the VBNR by 1. Then pop the long word data from the stack pointed to by BSP, pull registers according to the register list to the ring register bank pointed by VBNR, and execute the next instruction. Otherwise, just execute the next instruction.

This instruction can only be used in ring mode. If it is executed in global mode, bank mode violation exception processing is executed.

Operand Size: Long word

Instruction Format:

11111110

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.31 DIVXS (Divide Extended as Signed)

Operation: (Destination) / (Source) —> (Destination)

Assembler Format: DIVXS <EAs>, <EAd>

Description: Divide destination operand by source operand using signed arithmetic, and load the result in the destination. The lower byte of the destination contains the quotient, and the upper byte contains the remainder. The sign of the remainder, if any, is the same as the sign of the destination.

Note that division by zero will cause a trap exception. Note also that if an overflow is detected before the end of the instruction, the V bit is set, and the division is not executed. Therefore, the destination is not changed.

If bit 7 of the second instruction code is set to 1, or if the FNC is not specified as "000" or "001", an illegal instruction exception occurs. At this time, the PC indicates the second byte of DIVXS.

Operand Size: See the FNC field.

Instruction Format:

		3	4			
1 1 1 0 1 1 1 1	0	FNC	*	EAs	EAd	*: Don't care

FNC = 000: 16 bits/8 bits —> 8 bit quotient, 8 bit remainder
FNC = 001: 32 bits/16 bits —> 16 bit quotient, 16 bit remainder

Condition Codes:

CX	N	Z	V	C
U	S	S	S	0

Notes: 1. If the quotient overflows, N and Z are undefined.
2. If divisor is specified as 0, the condition codes changes as follows:

CX	N	Z	V	C
U	0	1	0	0

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.32 DIVXU (Divide Extended as Unsigned)

Operation: (Destination) / (Source) —> (Destination)

Assembler Format: DIVXU <EAs>, <EAd>

Description: Divide destination operand by source operand using unsigned arithmetic, and load the result to the destination. The lower byte of the destination contains the quotient, and the upper byte contains the remainder.

Note that division by zero will cause a trap exception. Note also that if an overflow is detected before the end of the instruction, the V bit is set, and the division is not executed. Therefore, the destination is not changed.

If bit 7 of the second instruction code is set to 1, or if the FNC is not specified as "000" or "001", an illegal instruction exception occurs. At this time, the PC indicates the second byte of DIVXU.

Operand Size: See the FNC field.

Instruction Format:

		3	4			
11101111	0	FNC	*	EAs	EAd	*: Don't care

FNC = 100: 16 bits/8 bits —> 8 bit quotient, 8 bit remainder
FNC = 101: 32 bits/16 bits —> 16 bit quotient, 16 bit remainder

Condition Codes:

CX	N	Z	V	C
U	S	S	S	0

Notes: 1. If the quotient overflows, N and Z are undefined.
2. If divisor is specified as 0, the condition codes changes as follows:

CX	N	Z	V	C
U	0	1	0	0

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.33 DNEG (Decimal Negate with CX Flag)

Operation: 0 – $(\text{Destination})_{BCD}$ – CX ⟶ $(\text{Destination})_{BCD}$

Assembler Format: DNEG <EAd>

Description: Subtract the destination operand and the CX bit from zero using BCD arithmetic, and load the result to the destination location. 10's complement is obtained if the CX bit is 1; 9's complement is obtained otherwise.

Operand Size: Byte

Instruction Format:

1 0 1 0 1 1 1 1	EAd

Condition Codes:

CX	N	Z	V	C
S	S*1	S*1	S*1	S

*1: N undefined.
Z set if the result is zero and Z = 1 prior to the operation, unaffected if the result is zero and Z = 0 prior to the operation, cleared otherwise.
V undefined.

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.34 DSUB (Decimal Subtract with CX Flag)

Operation: $(\text{Destination})_{BCD} - (\text{Source})_{BCD} - CX \longrightarrow (\text{Destination})_{BCD}$

Assembler Format: DSUB <EAs>, <EAd>

Description: Subtract the source operand and CX from the destination operand, and load the result to the destination location. The subtraction is a BCD operation.

Operand Size: Byte

Instruction Format:

10100111	EAs	EAd

Condition Codes:

CX	N	Z	V	C
S	S*1	S*1	S*1	S

*1: N undefined.
Z set if the result is zero and Z = 1 prior to the operation, unaffected if the result is zero and Z = 0 prior to the operation, cleared otherwise.
V undefined.

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.35 EXTS (Extend as Signed)

Operation: Rnd sign extension —> Rnd

Assembler Format: EXTS Rnd

Description: Sign extend the register contents from byte to word, word to long word or byte to long word.

For byte to word sign-extension, bit 7 of the register is copied into bits 15-8. For word to long word sign-extension, bit 15 is copied into bits 31-16. For byte to long word sign-extension, bit 7 is copied into bits 31-8.

A global bank register is specified by Rnd.

Operand Size: See Table in "Instruction Format".

Instruction Format:

	2	4	4
1 0 1 1 1 1	Sz	*	Rnd

*: Don't care

The Sz field specifies sign extension type:

Sz	Type	Assembler
00	Byte -> Word	W
01	Word -> Long Word	L
10	Byte -> Long Word	B

Condition Codes:

CX	N	Z	V	C
U	S	S	0	0

Available EA: None

16.3.36 EXTU (Extend as Unsigned)

Operation: 0 —> Upper bits of Rnd

Assembler Format: EXTU Rnd

Description: Extend the register contents from byte to word, or word to long word, or byte to long word.

For byte to word zero extension, zero is copied into bits 15-8. For word to long word zero extension, zero is copied into bits 31-16. For byte to long word zero extension, zero is copied into bits 31-8.

A global bank register is specified by Rnd.

Operand Size: See Table in "Instruction Format".

Instruction Format:

	2	4	4
101011	Sz	*	Rnd

*: Don't care

The Sz field specifies sign extension type:

Sz	Type	Assembler
00	Byte -> Word	W
01	Word -> Long Word	L
10	Byte -> Long Word	B

Condition Codes:

CX	N	Z	V	C
U	0	S	0	0

Available EA: None

16.3.37 ICBN (Increment Current Bank Number)

Operation: CBNR + 1 —> CBNR

if (the lower 3 bits of CBNR = the lower 3 bits of VBNR) then

{Stack the bank register}

VBNR + 1 —> VBNR

Assembler Format: ICBN

Description: First increment the CBNR by 1. If the lower 3 bits of CBNR = the lower 3 bits of the VBNR, push the registers specified by the lower 16 bits of R15 (register list) onto the stack specified by BSP with predecrement. Then, stack R15 as register list. Then increment the VBNR by 1 and execute the next instruction. Note that registers are stacked as long words. If the lower 3 bits of CBNR ≠ the lower 3 bits of VBNR, just execute the next instruction.

Note that this instruction can be executed only in ring mode. If this instruction is executed in global mode, a bank mode violation exception occurs.

Figure 16-17 shows the stack configuration.

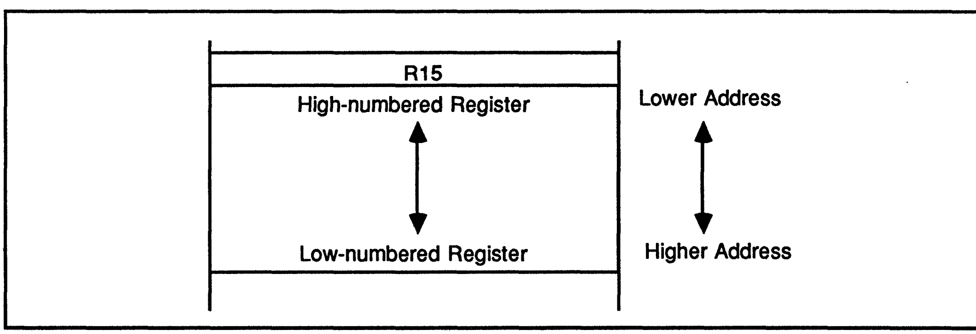


Figure 16-17. Stack Configuration after Register Stacking

Operand Size: Long word

Instruction Format:

1 1 1 1 1 1 0 1

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.38 JMP (Jump)

Operation: Effective address of destination —> PC

Assembler Format: JMP <EAd>

Description: Jump to the address specified by EAd.

Operand Size: None

Instruction Format:

10011011	EAd

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.39 JSR (Jump to Subroutine)

Operation: PC —> @–SP

Effective address of destination —> PC

Assembler Format: JSR <EAd>

Description: Push a start address of the next sequential instruction onto the stack, and jump to the address specified by EAd.

Operand Size: None

Instruction Format:

	8
10101011	EAd

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.40 LDC (Load to CR Register)

Operation: (Source) ⟶ CR

Assembler Format: LDC <EAs>, CR

Description: Load the source operand to the specified CR register (CR). Table 16-10 shows CR registers and CR codes.

Notes: 1. If BMR, GBNR, SR, EBR, RBR, USP, or IBR is specified as a CR register, this instruction is handled as a privileged instruction.

2. If the CR field specifies a CR code not shown in Table 16-10, the result is not stored in CR. The condition codes change according to the source operand size specified by bits 6-5 in the CR field. If bits 6-5 are "11", the source operand is accessed as a long word.

3. If this instruction is executed, the prefetch queue is reset and instructions following LDC are fetched again.

4. No interrupt can be accepted immediately after this instruction cycle.

Operation Size: Same as CR

Instruction Format:

	8	
11111011	CR	EAs
	$b_7, b_6, b_5, \ldots b_0$	

Condition Codes:

CR ≠ CCR and CR ≠ SR:

CX	N	Z	V	C
U	U	U	U	U

CR = CCR or CR = SR:

CX	N	Z	V	C
S*1	S*1	S*1	S*1	S*1

*1: CX same as bit 4 of result
N same as bit 3 of result
Z same as bit 2 of result
V same as bit 1 of result
C same as bit 0 of result

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													

16.3.41 LDM (Load to Multiple Registers)

Operation: (Source) —> Register

Assembler Format: LDM <EAs>, <register list>

Description: Load the data in memory area starting at the address specified by the source operand into the registers specified by the register list. In register indirect auto-decrement addressing mode, the data are moved from upper to lower addresses. In other addressing modes, they are moved from lower to upper addresses.

Note that the LDM operation in register indirect auto-increment or auto-decrement mode differs as follows: a register used for addressing is incremented or decremented by (the number of registers to be transferred) x operand size.

Moreover, if LDM utilizes current bank or previous bank mode, data is transferred to the current bank or previous bank registers, respectively.

Figure 16-18 shows memory configuration before LDM execution.

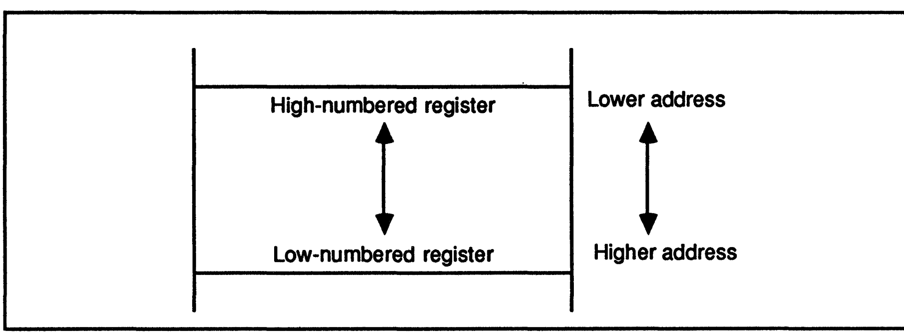


Figure 16-18. Memory Configuration Before LDM Execution

Operand Size: Byte, word, long word

Instruction Format:

	2		16				
0 1 1 1 0 1	Sz	EAs	R15	R14	• • •	R1	R0
			Register List				

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.42 LINK (Link)

Operation: Rn —> @–SP
SP —> Rn
SP + Imm —> SP

Assembler Format: LINK Rn, # Imm

Description: Push the current contents of the specified register onto the stack. After the push, load the current SP contents to the specified register and update SP by adding an immediate data Imm (Figure 16-19). At this time, Imm is sign extended. LINK can define a variable area in the stack area.

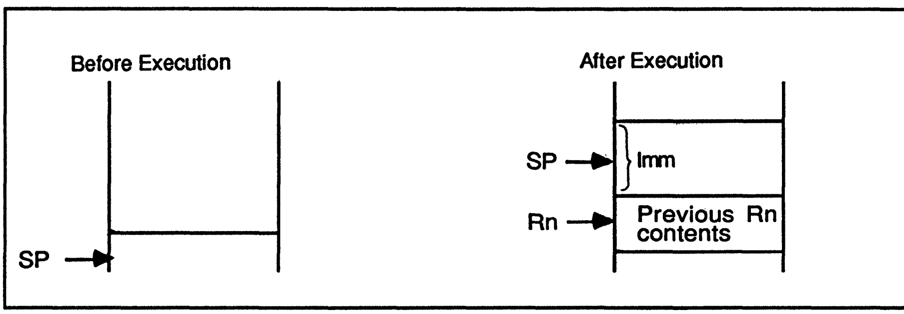
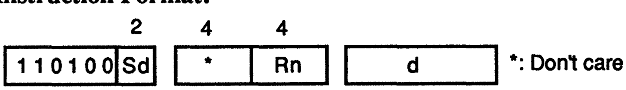


Figure 16-19. LINK Operation

Operand Size: Byte, word, long word

Instruction Format:



Note: Sd specifies Imm size as follows:

Sd	Imm Size
00	Byte
01	Word
10	Long word

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.43 MOVA (Move Effective Address)

Operation: Source —> (Destination)

Assembler Format: MOVA <EAs>, <EAd>

Description: Calculate the effective address of the source operand and move the address to the destination.

Operand Size: Long word

Instruction Format:

10111111	EAs	EAd

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.44 MOVF (Move Register 0 Fast)

Operation: R0 —> (Destination)

Assembler Format: MOVF <EAd>

Description: Move the contents of R0 in global bank to the destination.

Operand Size: Byte, word, long word

Instruction Format:

	2	
010111	Sz	EAd

Condition Codes:

CX	N	Z	V	C
U	S	S	0	0

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.45 MOVFP (Move from Peripheral)

Operation: (Source) —> (Destination)

Assembler Format: MOVFP <EAs>, <EAd>

Description: Move two or four bytes of data from the source address to the destination operand. Transfer the first byte from the source address, then increment the source address by 2, move the next byte, and repeat until the last byte is transferred. If the source address is even, then data will be moved from even addresses (Figure 16-20). If it is odd, then data will be moved from odd addresses (Figure 16-21).

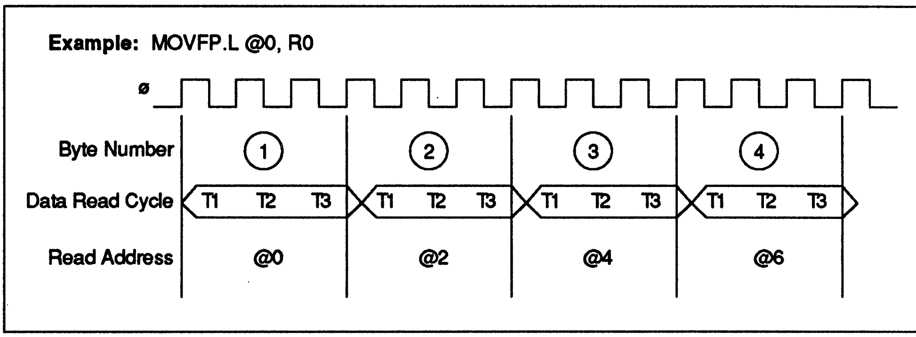


Figure 16-20. MOVFP From Even Addresses (Destination = Register)

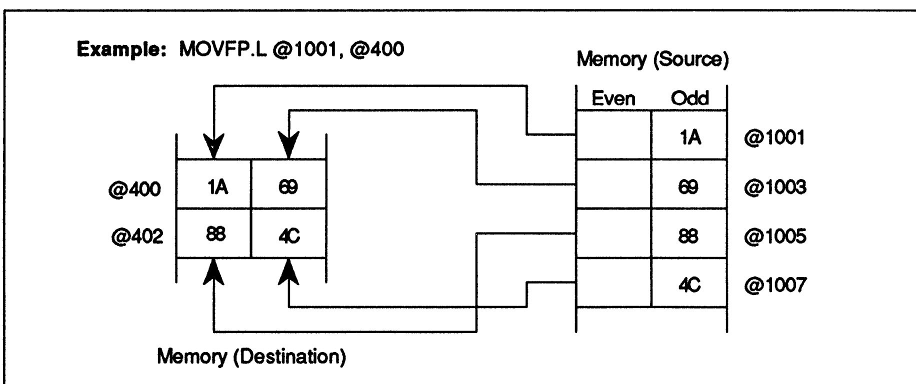


Figure 16-21. MOVFP From Odd Addresses (Destination = Memory)

Operand Size: Word, long word

Instruction Format:

1110001S	EAs	EAd

(bit above S: 1)

Note: S specifies operand size as follows:

S	Operand Size
0	Word
1	Long word

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.46 MOVFPE (Move from Peripheral with E Clock)

Operation: (Source) —> (Destination)

Assembler Format: MOVFPE <EAs>, <EAd>

Description: Move two or four bytes of data from the source address to the destination operand, synchronously with the E clock. Transfer the first byte from the source address, then increment the source address by 2, move the next byte, and repeat until the last byte is trnnsferred. If the source address is even, then data will be moved from even addresses (Figure 16-22). If it is odd, then data will be moved from odd addresses (Figure 16-23).

Note that if the internal RAM is specified as the source address, the read cycle begins at the falling edge of E clock and completes in two cycles.

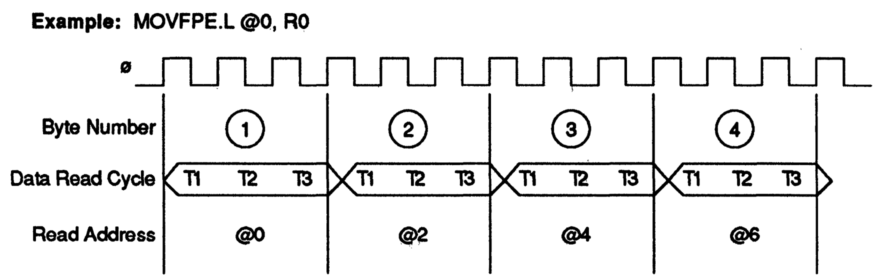


Figure 16-22. MOVFPE From Even Addresses (Destination = Register)

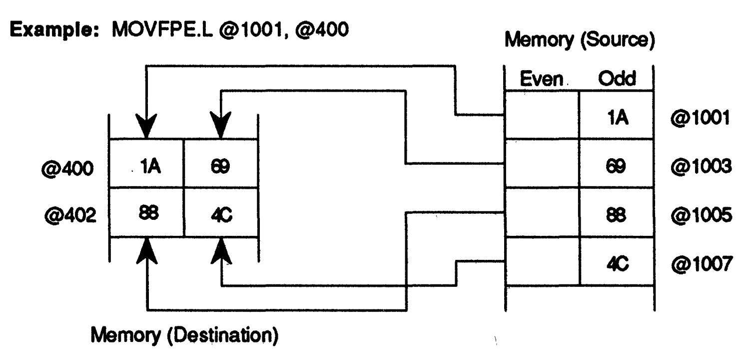


Figure 16-23. MOVFPE From Odd Addresses (Destination = Memory)

Operand Size: Byte, word, long word

Instruction Format:

0 1 1 1 1 1	Sz (2)	EAs	EAd

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.47 MOV:G (Move Data from Source to Destination)

Operation: (Source) —> (Destination)

Assembler Format: MOV:G <EAs>, <EAd>

Description: Move the source operand to the destination.

Operand Size: Byte, word, long word

Instruction Format:

	2		
0 0 0 0 1 1	Sz	EAs	EAd

Condition Codes:

CX	N	Z	V	C
U	S	S	0	0

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.48 MOV:Q (Move Quick)

Operation: Immediate data —> (Destination)

Assembler Format: MOV:Q #: 8, <EAd>

Description: Move the immediate data, which is sign extended to the operand size, to the destination. The immediate data must be between –128 and +127.

Operand Size: Byte, word, long word

Instruction Format:

	2	8	
0 0 0 1 1 1	Sz	Imm	EAd

Condition Codes:

CX	N	Z	V	C
U	S	S	0	0

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.49 MOV:R (Move Register)

Operation: Rns —> Rnd

Assembler Format: MOV:R Rns, Rnd

Description: Move the contents of source register to the destination register. Global bank registers can be specified as the source and destination registers.

Operand Size: Byte, word, long word

Instruction Format:

	2	4	4
0 0 1 0 1 1	Sz	Rns	Rnd

Condition Codes:

CX	N	Z	V	C
U	S	S	0	0

Available EA: None

16.3.50 MOV:RQ (Move Register Quick)

Operation: Immediate data —> Rnd

Assembler Format: MOV:RQ #:4, Rnd

Description: Move the sign extended immediate data to the destination register. The immediate data must be in the range from –8 to +7. A global bank register can be specified as the destination register (Rnd).

Operand Size: Byte, word, long word

Instruction Format:

	2		4	4
0 0 1 1 1 1	Sz		Rnd	Imm

Condition Codes:

CX	N	Z	V	C
U	S	S	0	0

Available EA: None

16.3.51 MOVS (Move with Sign Extension)

Operation: (Source) —> (Destination)

Assembler Format: MOVS <EAs>, <EAd>

Description: Move the source operand to the destination location. Operation size may be byte, word, or long word. Byte or word operands are sign extended to long words (32 bits) before the move, and all 32 bits are moved.

Operand Size: Byte, word, long word

Instruction Format:

	2		
0 1 0 0 1 1	Sz	EAs	EAd

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.52 MOVTP (Move to Peripheral)

Operation: (Source) —> (Destination)

Assembler Format: MOVTP <EAs>, <EAd>

Description: Move two or four bytes of data from the source operand to the Destination Address (DA). Transfer the first byte to DA plus 2 if word data, or DA plus 6 if long-word data. Then decrement the destination address by 2, transfer the next byte, and repeat until the last byte is transferred. If destination address is even, then data will be moved to even addresses (Figure 16-24). If it is odd, then data will be moved to odd addresses (Figure 16-25).

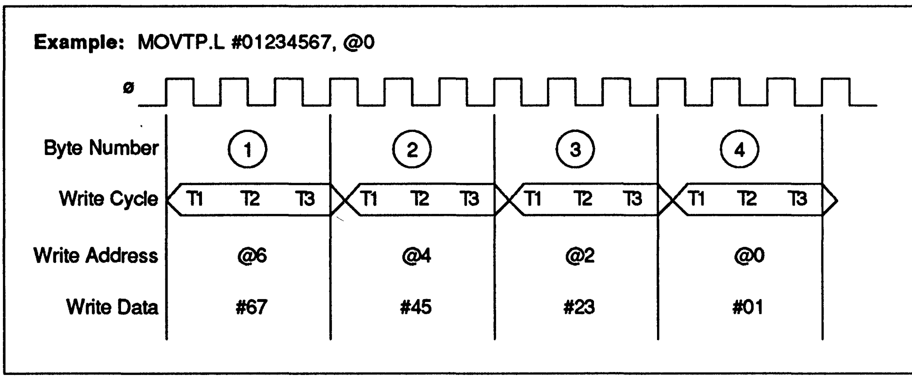


Figure 16-24. MOVTP To Even Addresses (Source = Immediate)

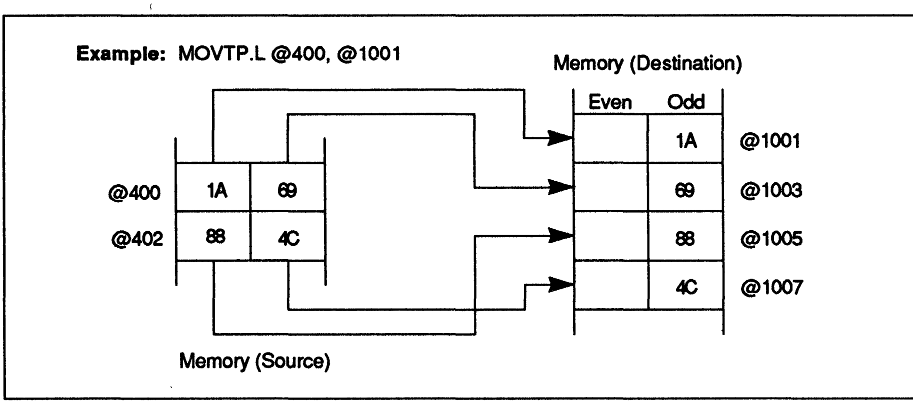


Figure 16-25. MOVTP To Odd Addresses (Source = Memory)

Operand Size: Word, long word

Instruction Format:

1110000S (bit above S: 1)	EAs	EAd

Note: S specifies operand size as follows:

S	Operand Size
0	Word
1	Long word

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.53 MOVTPE (Move to Peripheral with E Clock)

Operation: (Source) ⟶ (Destination)

Assembler Format: MOVTPE <EAs>, <EAd>

Description: Move two or four bytes of data from the source operand to the Destination Address (DA) synchronously with the E clock. Transfer the first byte to DA plus 2 if word data, or DA plus 6 if long-word data. Then decrement the destination address by 2, transfer the next byte, and repeat until the last byte is transferred. If destination address is even, then data will be moved to even addresses (Figure 16-26). If it is odd, then data will be moved to odd addresses (Figure 16-27).

Note that if the internal RAM is specified as the destination, the write cycle begins at the falling edge of the E clock and completes in 2 cycles.

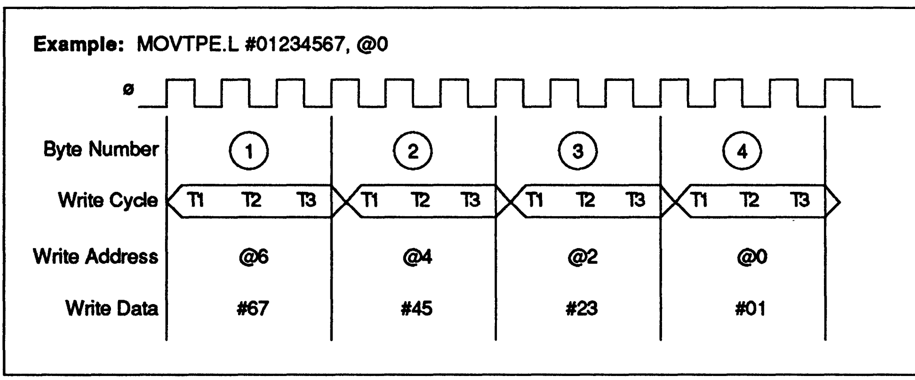


Figure 16-26. MOVTPE To Even Addresses (Source = Immediate)

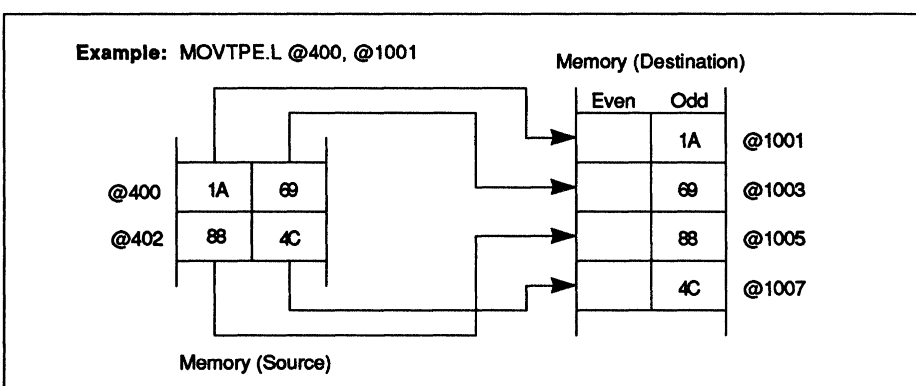


Figure 16-27. MOVTPE To Odd Addresses (Source = Memory)

Operand Size: Byte, word, long word

Instruction Format:

0 1 1 1 1 0	Sz	EAs	EAd
	2		

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.54 MULXS (Multiply Extended as Signed)

Operation: (Destination) * (Source) —> (Destination)

Assembler Format: MULXS <EAs>, <EAd>

Description: Multiply the two byte or word operands and load the signed word or long word result to the destination. Multiply is a signed operation.

Operand Size: See FNC

Instruction Format:

		3	4			
11101110	0	FNC	*	EAs	EAd	*: Don't care

FNC = 000: 8 bits x 8 bits —> 16 bits (The destination is read as word data.)
FNC = 001: 16 bits x 16 bits —> 32 bits (The destination is read as long word data.)

Condition Codes:

CX	N	Z	V	C
U	S	S	0	0

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.55 MULXU (Multiply Extended as Unsigned)

Operation: (Destination) * (Source) —> (Destination)

Assembler Format: MULXU <EAs>, <EAd>

Description: Multiply the two unsigned byte or word operands and load the unsigned word or long word result to the destination. Multiply is an unsigned operation.

Operand Size: See FNC.

Instruction Format:

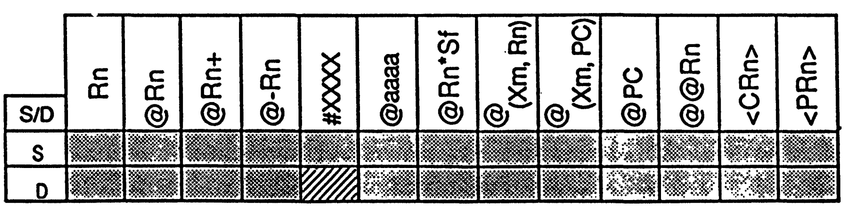
		3	4			
1 1 1 0 1 1 1 0	0	FNC	*	EAs	EAd	*: Don't care

FNC = 100: 8 bits x 8 bits —> 16 bits (The destination is read as word data.)
FNC = 101: 16 bits x 16 bits —> 32 bits (The destination is read as long word data.)

Condition Codes:

CX	N	Z	V	C
U	S	S	0	0

Available EA:



16.3.56 NEG (Negate)

Operation: 0 – (Destination) —> (Destination)

Assembler Format: NEG <EAd>

Description: Subtract the destination operand from zero using unsigned arithmetic, and load the result to the destination location.

Operand Size: Byte, word, long word

Instruction Format:

	2		
1 0 0 0 1 1	Sz		EAd

Condition Codes:

CX	N	Z	V	C
S	S	S	S	S

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.57 NEGX (Negate with CX Flag)

Operation: 0 – (Destination) – CX ⟶ (Destination)

Assembler Format: NEGX <EAd>

Description: Subtract the destination operand and CX from zero, and load the result to the destination location.

Operand Size: Byte, word, long word

Instruction Format:

	2	
100111	Sz	EAd

Condition Codes:

CX	N	Z	V	C
S	S	S*1	S	S

*1: Z set if the result is zero and Z = 1 prior to the operation, unaffected if the result is zero and Z = 0 prior to the operation, cleared otherwise.

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.58 NOP (No Operation)

Operation: No operation

Assembler Format: NOP

Description: No operation is performed. The PC is incremented and the next instruction is executed. Otherwise, the processor state is unaffected.

Operand Size: None

Instruction Format:

11111111

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.59 NOT (Not-Logical Complement)

Operation: ~ (Destination) —> (Destination)

Assembler Format: NOT <EAd>

Description: Take the one's complement of the destination operand, and store the result in the destination location.

Operand Size: Byte, word, long word

Instruction Format:

	2	
1 0 0 1 0 0	Sz	EAd

Condition Codes:

CX	N	Z	V	C
U	S	S	0	0

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.60 OR (OR Logical)

Operation: (Destination) ∨ (Source) ⟶ (Destination)

Assembler Format: OR <EAs>, <EAd>

Description: Inclusive OR the source operand with the destination operand and load the result to the destination location.

Operand Size: Byte, word, long word

Instruction Format:

	2		
100010	Sz	EAs	EAd

Condition Codes:

CX	N	Z	V	C
U	S	S	0	0

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.61 ORC (Logical OR CR Register)

Operation: CR $\vee$ (Source) —> CR

Assembler Format: ORC <EAs>, CR

Description: Inclusive OR the CR register (CR) with the source operand and load the result to the CR register. Table 16-10 shows CR registers and CR codes.

Notes: 1. If BMR, GBNR, SR, EBR, RBR, USP, or IBR is specified as a CR register, this instruction is handled as a privileged instruction.

2. If the CR field specifies a CR code not shown in not in Table 16-10, the source operand is ORed with 0. This result is not stored in CR. The condition codes also change according to the source operand size specified by bits 6-5 in the CR field. If bits 6-5 are "11", the source operand is accessed as a long word.

3. If this instruction is executed, the prefetch queue is reset and instructions following this instruction are fetched again.

4. No interrupt can be accepted immediately after this instruction cycle.

Operand Size: Same as CR

Instruction Format:

	8	
11111001	CR	EAs
	$b_7, b_6, b_5, ... b_0$	

Condition Codes:

CR ≠ CCR and CR ≠ SR:

CX	N	Z	V	C
S	S	S	0	0

CR = CCR or CR = SR:

CX	N	Z	V	C
S*1	S*1	S*1	S*1	S*1

*1: CX same as bit 4 of result
N same as bit 3 of result
Z same as bit 2 of result
V same as bit 1 of result
C same as bit 0 of result

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													

16.3.62 PGBN (Pull Global Bank Number)

Operation: @SSP+ —> GBNR
[Register copy]

Assembler Format: PGBN [<register list>]

Description: Pull the byte data from the supervisor stack and transfer it to the global bank number register (GBNR) to switch the bank. If R = 1, copy the registers from the previous bank into the corresponding registers in the current bank according to the next 16 bits (register list). If R = 0, execute the next instruction.

Note that this is a privileged instruction. If it is executed in user mode, privilege violation is generated.

Operand Size: Byte

Instruction Format:

	1			16		
1110100	R	R15	R14	• • •	R1	R0

Register List

R = 0: No register list exists and registers are not copied. (Instruction length = 1 byte)
R = 1: Registers are copied according to the 2-byte register list. (Instruction length = 3 bytes)

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.63 RESET (Reset)

Operation: Assert $\overline{RES}$ output, no processor operation

Assembler Format: RESET

Description: Assert $\overline{RES}$ output low for 256 clock cycles, resetting all external devices connected to the $\overline{RES}$ pin. The PC increments, and the CPU executes the next instruction. Otherwise the processor state is unaffected.

Note that this is a privileged instruction, and does not initialize the internal I/O devices.

Operand Size: None

Instruction Format:

11110000

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.64 ROTL (Rotate Left)

Operation: (Destination) Rotate left —> (Destination)

Assembler Format: ROTL Rn, <EAd> or ROTL #:5, <EAd>

Description: Rotate the bits of the destination left. The bit shifted out of the MSB is copied into both C and the LSB.

The rotate count can be specified in the following ways:

1. Dynamic: The rotate count is specified by register Rn (global bank register). The rotate count ranges from 0 to 255.

2. Static: The rotate count is specified by the immediate data. The rotate count ranges from 0 to 31.

See Figure 16-28.

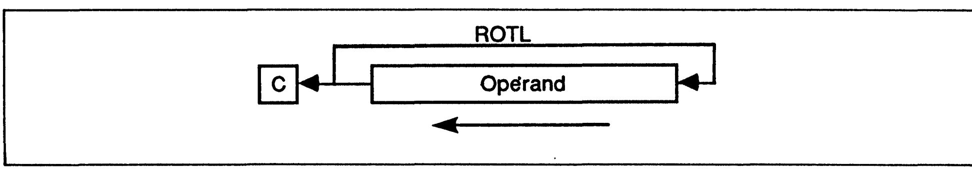


Figure 16-28. ROTL

Operand Size: Byte, word, long word

Instruction Format:

Dynamic:

	2		1	4	
0 1 1 0 0 0	Sz	1 1 0	*	Rn	EAd

*: Don't care

Static:

	2		5	
0 1 1 0 0 1	Sz	1 1 0	Imm	EAd

Condition Codes:

CX	N	Z	V	C
U	S	S	0	S*1

*1: C same as the last bit shifted out of the destination operand, cleared if shift count is 0.

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.65 ROTR (Rotate Right)

Operation: (Destination) Rotate right —> (Destination)

Assembler Format: ROTR Rn, <EAd> or ROTR #:5, <EAd>

Description: Rotate the bits of the destination right. The bit shifted out of the LSB is copied into C and the MSB.

The rotate count can be specified in the following ways:

1. Dynamic: The rotate count is specified by register Rn (global bank register). The rotate count ranges from 0 to 225.

2. Static: The rotate count is specified by the immediate data. The rotate count ranges from 0 to 31.

See Figure 16-29.

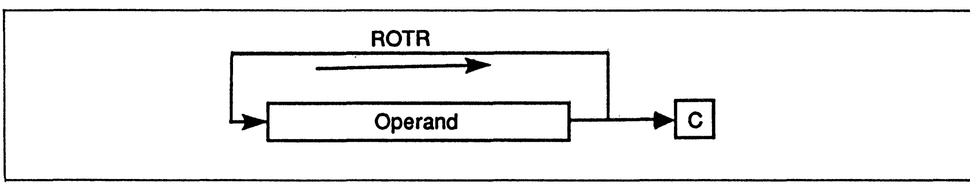


Figure 16-29. ROTR

Operand Size: Byte, word, long word

Instruction Format:

Dynamic:

	2		1	4	
0 1 1 0 0 0	Sz	0 1 0	*	Rn	EAd

*: Don't care

Static:

	2		5	
0 1 1 0 0 1	Sz	0 1 0	Imm	EAd

Condition Codes:

CX	N	Z	V	C
U	S	S	0	S*1

*1: C same as the last bit shifted out of the destination operand, cleared if shift count is 0.

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.66 ROTXL (Rotate Left with CX Flag)

Operation: (Destination) Rotate left ⟶ (Destination)

Assembler Format: ROTXL Rn, <EAd> or ROTXL #:5, <EAd>

Description: Rotate the bits of the destination left. The bit shifted out of the MSB is copied into C and CX, and the old CX is shifted into the LSB.

The rotate count can be specified in the following ways:

1. Dynamic: The rotate count is specified by register Rn (global bank register). The rotate count ranges from 0 to 255.

2. Static: The rotate count is specified by the immediate data. The rotate count ranges from 0 to 31.

See Figure 16-30.

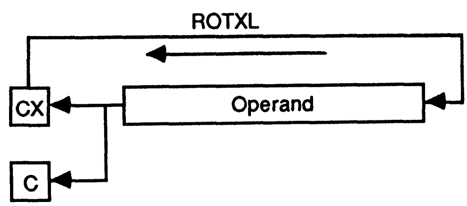


Figure 16-30. ROTXL

Operand Size: Byte, word, long word

Instruction Format:

Dynamic:

	2		1	4	
0 1 1 0 0 0	Sz	1 1 1	*	Rn	EAd

*: Don't care

Static:

	2		5	
0 1 1 0 0 1	Sz	1 1 1	Imm	EAd

Condition Codes:

CX	N	Z	V	C
U	S	S	0	S*1

*1: C same as the last bit shifted out of the destination operand, cleared if shift count is 0.

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.67 ROTXR (Rotate Right with CX Flag)

Operation: (Destination) Rotate right —> (Destination)

Assembler Format: ROTXR Rn, <EAd> or ROTXR #:5, <EAd>

Description: Rotate the bits of the destination right. The bit shifted out of the LSB is copied into C and CX, and the old CX is shifted into the MSB.

The rotate count can be specified in the following ways:

1. Dynamic: The rotate count is specified by register Rn (global bank register). The rotate count ranges from 0 to 255.

2. Static: The rotate count is specified by the immediate data. The rotate count ranges form 0 to 31.

See Figure 16-31.

Note that a global bank register can be specified as register Rn.

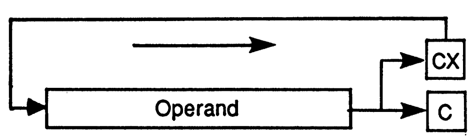
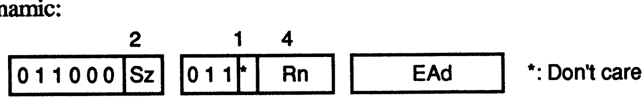


Figure 16-31. ROTXR

Operand Size: Byte, word, long word

Instruction Format:

Dynamic:



Static:

	2		5	
0 1 1 0 0 1	Sz	0 1 1	Imm	EAd

Condition Codes:

CX	N	Z	V	C
S	S	S	0	S*1

*1 : C same as the last bit shifted out of the destination operand, cleared if shift count is 0.

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.68 RTD (Return and Deallocate)

Operation: @SP+ —> PC
SP + disp —> SP

Assembler Format: RTD disp

Description: Pull the program counter from the stack. Then sign-extended displacement disp is added to the stack pointer. Program execution continues at the address specified by the PC.

Operand Size: None

Instruction Format:

	2	
1 0 1 1 1 0	Sd	disp

Note: The Sd field specifies the disp size as shown below:

Sd	disp Size
00	Byte
01	Word
10	Long word

Note that byte or word displacement is sign-extended to long word.

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.69 RTE (Return from Exception)

Operation: @SSP + —> SR
@SSP + —> PC

Assembler Format: RTE

Description: Pull the status register and program counter from the supervisor stack. Program execution continues at the address specified by the PC.

Load the processor state information saved on the supervisor stack at the beginning of exception processing back into the processor.

Note that RTE is a privileged instruction. Accordingly, if this instruction is executed in user mode, a privilege violation occurs.

Operand Size: None

Instruction Format:

11110001

Condition Codes:

CX	N	Z	V	C
S*1	S*1	S*1	S*1	S*1

*1: CX same as bit 4 of the word on the stack
N same as bit 3 of the word on the stack
Z same as bit 2 of the word on the stack
V same as bit 1 of the word on the stack
C same as bit 0 of the word on the stack

Available EA: None

16.3.70 RTR (Return and Restore Condition Codes)

Operation: @SP+ —> CCR
@SP+ —> PC

Assembler Format: RTR

Description: Pull the condition and program counter from the stack. Program execution continues at the address specified by the PC.

Figure 16-32 shows the stack configuration in the RTR instruction.

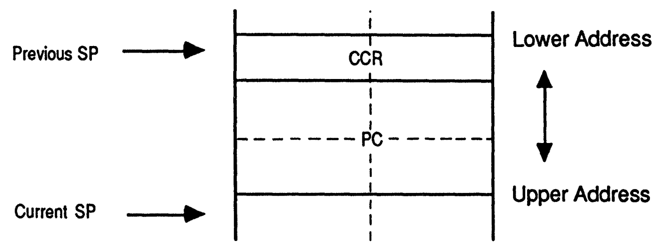


Figure 16-32. Stack Configuration in RTR Instruction

Operand Size: None

Instruction Format:

1 1 1 1 0 1 0 0

Condition Codes:

CX	N	Z	V	C
S*1	S*1	S*1	S*1	S*1

*1: CX same as bit 4 of the word on the stack

N same as bit 3 of the word on the stack

Z same as bit 2 of the word on the stack

V same as bit 1 of the word on the stack

C same as bit 0 of the word on the stack

Available EA: None

16.3.71 RTS (Return from Subroutine)

Operation: @SP+ —> PC

Assembler Format: RTS

Description: Pull the program counter from the stack. Program execution continues at the address specified by the pulled PC. This instruction is used to return to the address stacked by the BSR and JSR instructions.

Operand Size: None

Instruction Format:

1 0 1 1 1 0 1 1

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.72 SCB (Subtract, Compare, and Branch Conditionally)

Operation: If not cc then
begin
Rn–1 —> Rn
If Rn ≠ –1 then PC + disp —> PC
end

Assembler Format: SCB/cc Rn, disp

Description: If the specified condition is true, only execute the next instruction. If the condition is false, subtract 1 from the specified register. At this time, if the subtract result is –1, execute the next instruction. Otherwise branch to the target label by adding the displacement to the PC.

Note that the PC used for the address calculation indicates the next instruction start address. In addition, a global bank register can be specified as Rn (only lower 16 bits are valid). See Table 16-10 for conditions and their mnemonics. Write the corresponding mnemonic in place of "cc" in the instruction.

Operand Size: None

Instruction Format:

	2	4	4	
1 0 1 1 0 1	Sd	Rn	cc	disp

Note: Sd specifies the disp size as follows:

Sd	disp Size
00	Byte
01	Word
10	Long word

Note that byte or word displacements are sign-extended to long word.

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.73 SCMP (String Compare According to Condition Codes)

Operation: SCMP/F:

Rnc —> Temp.
If Rnc = 0, then {execute next instruction}
Repeat Temp. – 1 —> Temp.
    Increment Rns and Rnd
    (Destination) – (Source)
Until cc or Temp. = 0
Temp. —> Rnf

or

SCMP/B:
Rnc -> Temp.
If Rnc = 0, then {execute next instruction}
Repeat Temp. – 1 —> Temp.
    Decrement Rns and Rnd
    (Destination) – (Source)
Until cc or Temp. = 0
Temp. -> Rnf

Assembler Format: SCMP/cc/F Rns, Rnd, Rnc, Rnf or SCMP/cc/B Rns, Rnd, Rnc, Rnf

Description: Tests the contents of the destination operand pointer Rnc. If the contents of Rnc is zero, the SCMP instruction is terminated without affecting Rnc and condition codes. On the other hand, if the contents of Rnc is zero, subtracts the contents of Rnc from the destination subtract the source operand from the destination operand and set condition codes accordingly while the condition is false and Rnc $\neq$ 0. Each time, the source and destination registers are incremented (/F: forward option) or decremented (/B: backward option) by 1. Global bank registers can be specified as Rnc, Rns, Rnd, and Rnf. Rnf equals (Initial Rnc—actual comparison count).

The CPU checks for interrupt requests whenever a comparison is performed. If there is an interrupt request, the CPU stops SCMP execution and puts the current register status on the stack. At this time, the PC to be stacked indicates the start address of the SCMP instruction. It then processes the interrupt exception. After interrupt servicing, the CPU continues the SCMP execution. Tables 16-15 and 16-16 show register values after interrupt acceptance and register values after SCMP execution completion.

Table 16-15. Register Values after Interrupt Acceptance

Register name	Value
Rnc	(Initial Rnc) - (Actual comparison count)
Rns	Source location +1 when SCMP execution stops (@Rns+) Source location when SCMP execution stops (@-Rns)
Rnd	Destination location + 1 when SCMP execution stops (@Rnd+) Destination location when SCMP execution stops (@-Rnd)
Rnf	Rnf is not stacked

Table 16-16. Register Values after SCMP Execution Completion

Register name	Value
Rnc	Undefined
Rns	The last source location + 1 (@Rns+) The last source location (@-Rns)
Rnd	The last destination location + 1 (@Rnd+) The last destination location (@-Rnd)
Rnf	(Initial Rnc) – (Actual comparison count) Rnf does not change if initial Rnc = 0.

Note that only Rnf and Rnc registers can overlap.

See Table 16-11 for conditions and their mnemonics. Write the corresponding mnemonic in place of "cc" in the instruction.

Operand Size: Byte, word

Instruction Format:

	2		3	4	4	4	4	4
100101	Sz	1	TYP	Rnc	Rns	Rnd	Rnf	cc

*: Don't care

(1) Sz specifies the operand size as follows:

Sz	Size
00	Byte
01	Word

(2) TYP specifies the source and destination addressing modes.
SCMP/F: TYP = 010....Combination of @Rns+ and @Rnd+
SCMP/B: TYP = 101....Combination of @–Rns and @–Rnd

(3) Rnc: Specifies a global bank register which specifies the limit value of comparison count. (Lower 16 bits are valid.)
Rns: Specifies a global bank register which specifies the source location (32 bits).
Rnd: Specifies a global bank register which specifies the destination location (32 bits).
Rnf: Specifies a global bank register which stores (Initial Rnc) – (Actual comparison count). (Lower 16 bits are valid.)

Condition Codes:

CX	N	Z	V	C
U	U	S*1	U	U

*1: If the contents of Rnc is zero before SCMP execution, Z is not affected.
If the contents of Rnc is not zero before SCMP execution, Z is set if SCMP is completed with cc false; cleared otherwise.

Available EA: None

16.3.74 SET (Set According to Condition Codes)

Operation: If cc then H'FF —> (Destination)
else H'00 —> (Destination)

Assembler Format: SET/cc <EAd>

Description: If the specified condition is true, all bits of the byte specified by the effective address are set to 1. If it is false, all bits of the operand are cleared to 0. Table 16-11 shows the conditions and their mnemonics. Write the corresponding mnemonic in place of "cc" in the instruction.

Operand Size: Byte

Instruction Format:

	4	4	
10110111	*	cc	EAd

*: Don't care

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.75 SHAL (Shift Arithmetically Left)

Operation: (Destination) Shift left —> (Destination)

Assembler Format: SHAL Rn, <EAd> or SHAL #:5, <EAd>

Description: Arithmetically shift the bits of the destination operand to the left. The bit shifted out of the MSB is copied into both C and CX. The LSB becomes 0. V indicates whether or not the MSB of data changes.

The shift count can be specified in the following ways:

1. Dynamic: The shift count is specified by register Rn (global bank register). The shift count ranges from 0 to 255.

2. Static: The shift count is specified by the immediate data. The shift count ranges from 0 to 31.

See Figure 16-33.

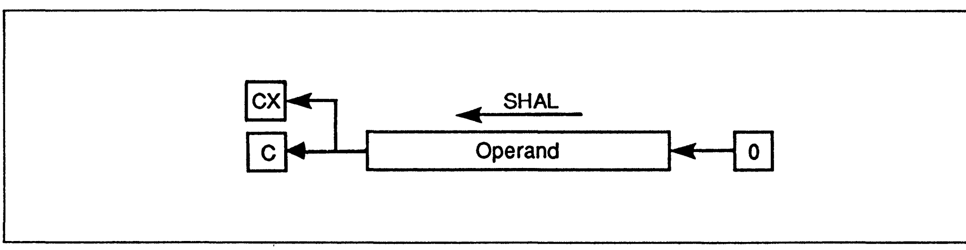
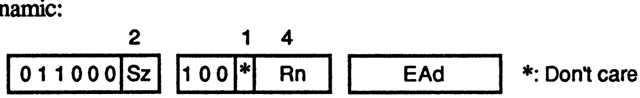


Figure 16-33. SHAL

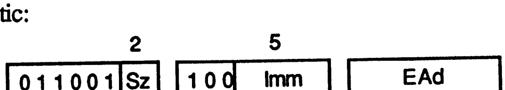
Operand Size: Byte, word, long word

Instruction Format:

Dynamic:



Static:



Condition Codes:

CX	N	Z	V	C
S*1	S	S	S*1	S*1

*1: CX same as C, unaffected if the shift count is 0.
V set if the MSB changes, cleared otherwise.
C same as the last bit shifted out of the destination operand, cleared if the shift count is zero.

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.76 SHAR (Shift Arithmetically Right)

Operation: (Destination) Shift right —> (Destination)

Assembler Format: SHAR Rn, <EAd> or SHAR #:5, <EAd>

Description: Arithmetically shift the bits of the destination operand to the right. The bit shifted out of the LSB is copied into both C and CX. The old MSB is copied into the MSB.

The shift count can be specified in the following ways:

1. Dynamic: The shift count is specified by register Rn (global bank register). The shift count ranges from 0 to 255.

2. Static: The shift count is specified by the immediate data. The shift count ranges from 0 to 31.

See Figure 16-34.

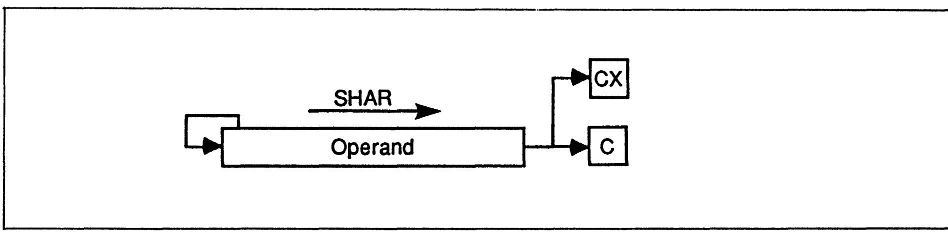
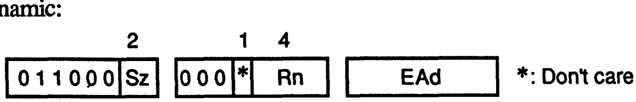


Figure 16-34. SHAR

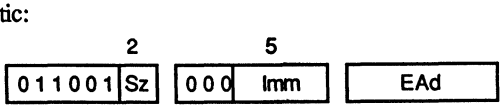
Operand Size: Byte, word, long word

Instruction Format:

Dynamic:



Static:



Condition Codes:

CX	N	Z	V	C
S*1	S	S	0	S*1

*1: CX same as C, unaffected if the shift count is zero.
C same as the last bit shifted out of the destination operand, cleared if the shift count is zero.

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.77 SHLL (Shift Logically Left)

Operation: (Destination) Shift left —> (Destination)

Assembler Format: SHLL Rn, <EAd> or SHLL #:5, <EAd>

Description: Logically shift the bits of the destination operand to the left. The bit shifted out of the MSB is copied into both C and CX. The LSB becomes 0.

The shift count can be specified in the following ways:

1. Dynamic: The shift count is specified by register Rn (global bank register). The shift count ranges from 0 to 255.

2. Static: The shift count is specified by the immediate data. The shift count ranges from 0 to 31.

See Figure 16-35.

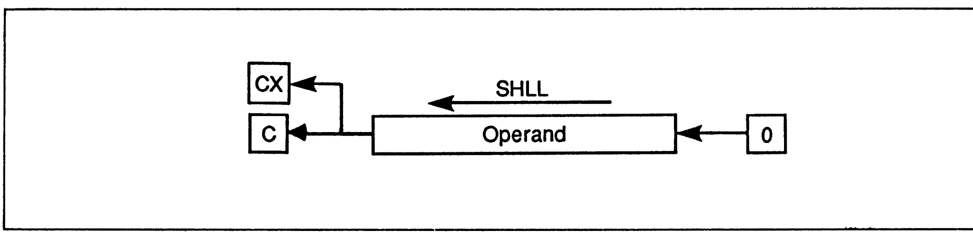


Figure 16-35. SHLL

Operand Size: Byte, word, long word

Instruction Format:

Dynamic:

	2			1	4	
011000	Sz		101	*	Rn	EAd

*: Don't care

Static:

	2			5	
011001	Sz		101	Imm	EAd

Condition Codes:

CX	N	Z	V	C
S*1	S	S	0	S*1

*1: CX same as C, unaffected if the shift count is zero.
C same as the last bit shifted out of the destination operand, cleared if the shift count is zero.

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.78 SHLR (Shift Logically Right)

Operation: (Destination) shift right —> (Destination)

Assembler Format: SHLR Rn, <EAd> or SHLR #:5, <EAd>

Description: Logically shift the bits of the destination operand to the right. The bit shifted out of the LSB is copied into C and CX. The MSB becomes 0.

The shift count can be specified in the following ways:

1. Dynamic: The shift count is specified by register Rn (global bank register). The shift count ranges from 0 to 255.

2. Static: The shift count is specified by the immediate data. The shift count ranges from 0 to 31.

See Figure 16-36.

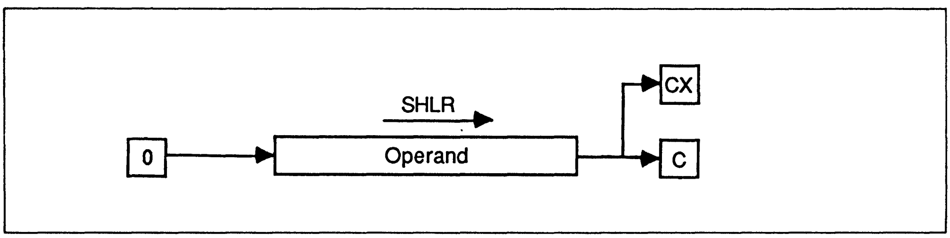


Figure 16-36. SHLR

Operand Size: Byte, word, long word

Instruction Format:

Dynamic:

	2		1	4		
011000	Sz	001	*	Rn	EAd	*: Don't care

Static:

	2		5	
011001	Sz	001	Imm	EAd

Condition Codes:

CX	N	Z	V	C
S*1	S	S	0	S*1

*1: CX same as C, unaffected if the shift count is zero.
C same as the last bit shifted out of the destination operand, cleared if the shift count is zero.

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.79 SLEEP (Sleep)

Operation: The HD641016 enters low power consumption mode.

Assembler Format: SLEEP

Description: Put the MPU into sleep mode. In sleep mode, the MPU stops instruction execution and waits for exception processing requests. See "Section 15. Low Power Consumption Modes" for details.

This instruction is privileged. If this instruction is executed in user mode, a privilege violation occurs.

Operand Size: None

Instruction Format:

1 1 1 1 0 1 0 1

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.80 SMOV (String Move)

Operation: SMOV/F:

Rnc —> Temp.
if Rnc = 0 then {execute next instruction}
Repeat Temp. –1 —> Temp.
(Source) —> (Destination)
Increment Rns and Rnd
Until Temp. = 0

or

SMOV/B:

Rnc —> Temp.
if Rnc = 0 then {execute next instruction}
Repeat Temp. –1 —> Temp.
Decrement Rns and Rnd
(Source) —> (Destination)
Until Temp. = 0

Assembler Format: SMOV/F Rns, Rnd, Rnc or SMOV/B Rns, Rnd, Rnc

Description: Move the data string addressed by register Rns to the destination specified by Rnd, until Rnc becomes 0. Only the lower 16 bits of Rnc is used for the counter. Rns and Rnd are incremented or decremented, depending on whether the /F or /B option is selected.

The CPU checks for interrupt requests whenever a comparison is performed. If there is an interrupt request, the CPU stops SMOV execution and puts the current register status on the stack. At this time, the PC to be stacked indicates the start address of the SMOV instruction. It then processes the interrupt exception. After interrupt servicing, the CPU continues the SMOV execution. Tables 16-17 and 16-18 show register values after interrupt acceptance and register values after SMOV execution completion.

Table 16-17. Register Values after Interrupt Acceptance

Register name	Value
Rnc	(Initial Rnc) – (Actual comparison count)
Rns	Source location +1 when SMOV execution stops (@Rns+) Source location when SCMP execution stops (@-Rns)
Rnd	Destination location +1 when SMOV execution stops (@Rnd+) Destination location when SCMP execution stops (@-Rnd)

Table 16-18. Register Values after SCMP Execution Completion

Register name	Value
Rnc	Undefined
Rns	The last source location + 1 (@Rns+) The last source location (@-Rns)
Rnd	The last destination location + 1 (@Rnd+) The last destination location (@-Rnd)

Note that Rnc, Rns and Rnd registers must not overlap.

Operand Size: Byte, word

Instruction Format:

	2		3	4	4	4
1 0 0 1 0 1	Sz	0	TYP	Rnc	Rns	Rnd

(1) The Sz field specifies the operand size as follows:

Sz	Size
00	Byte
01	Word

(2) TYP specifies the source and destination addressing modes.
SMOV/F: TYP = 010 ... Combination of @Rns+ and @Rnd+
SMOV/B: TYP = 101 ...Combination of @–Rns and @–Rnd

(3) Rnc: Specifies a global bank register which specifies the limit value of comparison count (Lower 16 bits are valid).
Rns: Specifies a global bank register which specifies the source location (32 bits).
Rnd: Specifies a global bank register which specifies the destination location (32 bits).

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.81 SSCH (String Search According to Condition Codes)

Operation: SSCH/F:

Rnc —> Temp.

if Rnc = 0 then {execute next instruction}

Repeat Temp. – 1 —> Temp.

(Destination) – (Rns)

Increment Rnd

Until cc or Temp. = 0

Temp. —> Rnf

or

SSCH/B:

Rnc —> Temp.

if Rnc = 0 then {execute next instruction}

Repeat Temp. – 1 —> Temp.

Decrement Rnd

(Destination) – (Rns)

Until cc or Temp. = 0

Temp. —> Rnf

Assembler Format: SSCH/cc/F Rns, Rnd, Rnc, Rnf or SSCH/cc/B Rns, Rnd, Rnc, Rnf

Description: Tests the contents of the destination operand pointer Rnc. If the contents of Rnc is zero, the SSCH instruction execution is terminated without affecting Rnc and condition codes. On the other hand, if the contents of Rnc is zero, subtracts the contents of Rnc from the destination subtract the source operand from the destination operand and set the condition codes accordingly. Decrement Rnc and repeat while cc is not true and Rnc $\neq$ 0. The destination operand pointer is either incremented or decremented, depending on the specification. Only the lower 16 bits of Rnc is used for the counter. Rnf equals Rnc minus actual comparison count.

The CPU checks for interrupt requests whenever a comparison is performed. If there is an interrupt request, the CPU stops SSCH execution and puts the current register status on the stack. It then processes the interrupt exception. After interrupt servicing, the CPU restarts the SSCH execution. Tables 16-19 and 16-20 show register values after interrupt acceptance and register values after SSCH execution completion.

Table 16-19. Register Values after Interrupt Acceptance

Register name	Value
Rnc	(Initial Rnc) – (Actual comparison count)
Rns	Unchanged
Rnd	Destination location + 1 when SSCH execution stops (@Rnd+) Destination location when SSCH execution stops (@-Rnd)
Rnf	Unchanged

Table 16-20. Register Values after SSCH Execution Completion

Register name	Value
Rnc	Undefined
Rns	Unchanged
Rnd	The last destination location + 1 (@Rnd+) The last destination location (@-Rnd)
Rnf	(Initial Rnc) – (Actual comparison count) Rnf does not change if initial Rnc = 0.

Note that only Rnf and Rnc registers can overlap.

Table 16-11 shows the conditions and their mnemonics. Write the corresponding mnemonic in place of "cc" in the instruction.

Operand Size: Byte, word

Instruction Format:

	2		3	4	4	4	4	4
100101	Sz	1	TYP	Rnc	Rns	Rnd	Rnf	cc

(1) The Sz field specifies the operand size.

Sz	Size
00	Byte
01	Word

(2) TYP specifies the source and destination addressing modes.
SSCH/F: TYP=000. . .Combination of @Rns and @Rnd+
SSCH/B: TYP=000. . .Combination of @Rns and @–Rnd

(3) Rnc: Specifies a global bank register which specifies the limit value of comparison count (Lower 16 bits are valid.).
Rns: Specifies a global bank register which specifies the source location (Lower 8 bits or 16 bits are valid).
Rnd: Specifies a global bank register which specifies the destination location (32 bits).
Rnf: Specifies a global bank register which contains (Rnc) – (Actual comparison count). (Lower 16 bits are valid.)

Condition Codes:

CX	N	Z	V	C
U	U	S*1	U	U

*1: If the contents of Rnc is zero before SSCH execution, Z is not affected.
If the contents of Rnc is not zero before SSCH execution, Z is set if SSCH is completed with cc false; cleared otherwise.

Available EA: None

16.3.82 SSTR (String Store)

Operation: SSTR/F:

Rnc —> Temp.

if Rnc = 0 then {execute next instruction}

Repeat Temp. – 1 —> Temp.

    Rns —> (Destination)

    Increment Rnd

Until Temp. = 0

or

SSTR/B:

Rnc —> Temp.

if Rnc = 0 then {execute next instruction}

Repeat Temp. – 1 —> Temp.

    Decrement Rnd

    Rns —> (Destination)

Until Temp. = 0

Assembler Format: SSTR/F Rns, Rnd, Rnc or SSTR/B Rns, Rnd, Rnc

Description: Move the data string addressed by register Rns to the destination specified by Rnd, and increment or decrement Rnd until Rnc becomes 0.

The CPU checks for interrupt requests whenever a comparison is performed. If there is an interrupt request, the CPU stops SSTR execution and puts the current register status on the stack. It then processes the interrupt exception. After interrupt servicing, the CPU restarts the SSTR execution. Tables 16-21 and 16-22 show register values after interrupt acceptance and register values after SSTR execution completion. In addition, global bank registers can be specified as Rnc, Rns and Rnd.

Table 16-21. Register Values after Interrupt Acceptance

Register name	Value
Rnc	(Initial Rnc) – (Actual comparison count)
Rns	Unchanged
Rnd	Destination location + 1 when SSTR execution stops (@Rnd+) Destination location when SSTR execution stops (@-Rnd)

Table 16-22. Register Values after SSTR Execution Completion

Register name	Value
Rnc	Undefined
Rns	Unchanged
Rnd	The last destination location + 1 (@Rnd+) The last destination location (@-Rnd)

Note that Rnc, Rns, and Rnd registers must not overlap.

Operand Size: Byte, word

Instruction Format:

	2		3	4		4	4
1 0 0 1 0 1	Sz	0	TYP	Rnc		Rns	Rnd

(1) Sz specifies the operand size as follows:

Sz	Size
00	Byte
01	Word

If any other combination is specified, an illegal instruction exception occurs.

(2) TYP specifies the move direction of field destination operand pointer:
SSTR/F: TYP = 000 ... Combination of Rns and @Rnd+
SSTR/B: TYP = 100 ... Combination of Rns and @–Rnd

(3) Rnc: Specifies a global bank register which specifies the limit value of comparison count (Lower 16 bits are valid).
Rns: Specifies a global bank register which specifies the source location (Lower 8 bits or 16 bits are valid).
Rnd: Specifies a global bank register which specifies the destination location(32 bits).

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.83 STC (Store CR Register)

Operation: CR —> (Destination)

Assembler Format: STC CR, <EAd>

Description: Store the contents of the CR register (CR) in the destination location. Table 16-10 shows CR registers and CR codes.

Notes: 1. If BMR, GBNR, SR, EBR, RBR, USP, or IBR is specified as a CR register, STC is handled as a privileged instruction.

2. If the CR field specifies a CR code not shown in Table 16-10, 0 is stored in the destination. If bits 5 and 6 of the CR field is "11", the CR size is long word.

3. If this instruction is executed, the prefetch queue is reset and the instructions following STC are fetched again.

4. No interrupt can be accepted just after STC execution.

Operand Size: Same as CR

Instruction Format:

11111100	CR (8)	EAs
	$b_7, b_6, b_5 \dots b_0$	

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.84 STM (Store Multiple Registers)

Operation: Register —> (Destination)

Assembler Format: STM <Register List>, <EAd>

Description: Transfer the registers specified by the register list to memory, starting at the location specified by the destination operand. In register direct auto-decrement addressing mode, the registers are stored starting at the upper address and proceeding to lower addresses. In other modes, they are stored from lower to upper addresses.

Note that the STM operation in register indirect auto-increment or auto-decrement mode differs from that in other mode as follows:

1. A register used for the addressing is incremented or decremented by the operand size after the STM execution.

2. Registers used for the STM execution are incremented or decremented by (the number of registers to be transferred) x operand size after the STM execution.

Furthermore, when the STM utilizes current bank or previous bank mode, current bank or previous bank registers are transferred, respectively.

Figure 16-37 shows memory configuration after STM execution.

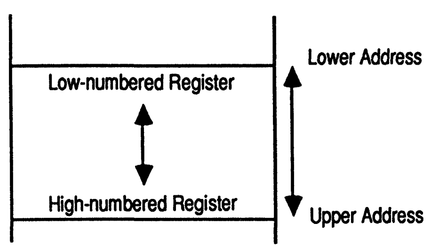


Figure 16-37. Memory Configuration after STM Execution

Operand Size: Byte, word, long word

Instruction Format:

	2		16				
0 1 1 1 0 0	Sz	EAd	R15	R14	• • •	R1	R0
			Register List				

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.85 SUB:G (Subtract Binary)

Operation: (Destination) – (Source) —> (Destination)

Assembler Format: SUB:G <EAs>, <EAd>

Description: Subtract the source operand from the destination operand and load the result to the destination location.

Operand Size: Byte, word, long word.

Instruction Format:

	2		
000001	Sz	EAs	EAd

Condition Codes:

CX	N	Z	V	C
S	S	S	S	S

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.86 SUB:R (Subtract Register)

Operation: Rnd – Rns —> Rnd

Assembler Format: SUB:R Rns, Rnd

Description: Subtract the source operand from the destination operand and load the result to the destination location. Global bank registers can be specified as Rns and Rnd.

Operand Size: Byte, word, long word.

Instruction Format:

	2	4	4
001001	Sz	Rns	Rnd

Condition Codes:

CX	N	Z	V	C
S	S	S	S	S

Available EA: None

16.3.87 SUB:RQ (Subtract Register Quick)

Operation: Rns – Immediate data —> Rnd

Assembler Format: SUB:RQ #:4, Rnd

Description: Subtract the zero extended immediate data from the destination operand and load the result to the destination location. A global bank register can be specified as Rnd. Immediate data must range from 0-15.

Operand Size: Byte, word, long word.

Instruction Format:

	2	4	4
0 0 1 1 0 1	Sz	Rns	Imm

Condition Codes:

CX	N	Z	V	C
S	S	S	S	S

Available EA: None

16.3.88 SUBS (Subtract with Sign Extension)

Operation: (Destination) – (Source) ⟶ (Destination)

Assembler Format: SUBS <EAs>, <EAd>

Description: Subtract the source operand from the destination operand and load the long word result to the destination location. Byte and word operands are sign extended to 32 bits (long word) before the operation, since this subtraction is performed in long words. The destination is always accessed as long word.

Operand Size: Byte, word, long word.

Instruction Format:

	2		
010001	Sz	EAs	EAd

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.89 SUBX (Subtract with CX Flag)

Operation: (Destination) – (Source) – CX ⟶ (Destination)

Assembler Format: SUBX <EAs>, <EAd>

Description: Subtract the source operand and CX from the destination operand and load the result to the destination location.

Operand Size: Byte, word, long word.

Instruction Format:

	2		
0 1 0 1 0 1	Sz	EAs	EAd

Condition Codes:

CX	N	Z	V	C
S	S	S*1	S	S

*1: Z set if its previous value was 1 and the result is zero, cleared otherwise.

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.90 SWAP (Swap Register Halves)

Operation: <EAd> [Upper half] <—> <EAd> [Lower half]

Assembler Format: SWAP <EAd>

Description: Exchange the contents of the upper half of the operand and the lower half.

Figure 16-38 shows the SWAP operation.

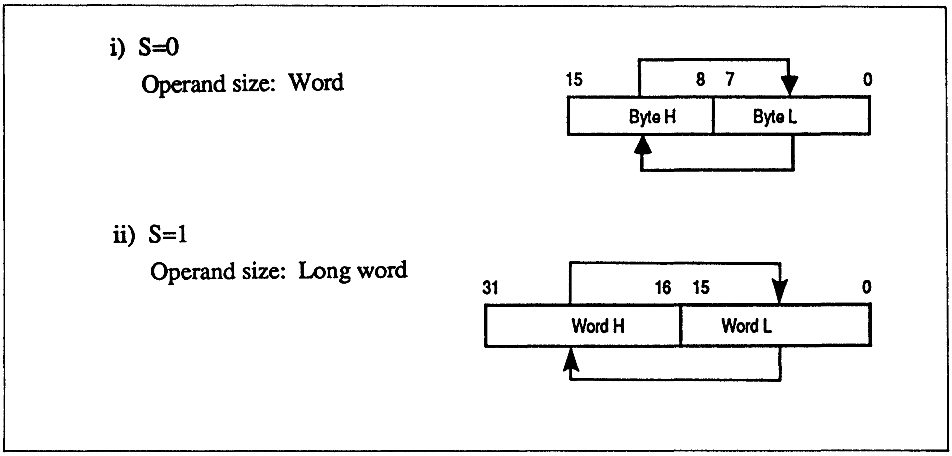


Figure 16-38. SWAP Operation

Operand Size: Word, long word

Instruction Format:

	1	
1 1 1 0 1 0 1	S	EAd

Note: S specifies operand size as follows:

S	Operand Size
0	Byte
1	Word

Condition Codes:

CX	N	Z	V	C
U	S	S	0	0

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.91 TAS (Test and Set)

Operation: (Destination) – 0
1 —> (<bit 7> of <Destination>)

Assembler Format: TAS <EAd>

Description: Compare the byte operand specified by the effective address to 0. Set N and Z according to the result. Then, set the operand's MSB to 1.

Note that this instruction performs read-modify-write cycle. See "4.9.2 CPU Read/Write Cycle" for details.

Operand Size: Byte

Instruction Format:

	8
11101110	EA

Condition Codes:

CX	N	Z	V	C
U	S	S	0	0

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.92 TRAP (Trap According to Condition Codes)

Operation: If cc then

PC —> @-SSP; SR —> @-SSP; H'7 —> PC

Assembler Format: TRAP/cc

Description: Initiate trap exception processing if selected condition is true. Use the corresponding conditions (Table 16-11) in place of "cc" in the instruction. For example, to trap on equal condition, write "TRAP/EQ". The vector number is H'7.

Operand Size: None

Instruction Format:

	4	4
11110011	*	cc

*: Don't care

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.93 TRAPA (Trap Always)

Operation: PC —> @–SSP
SR —> @–SSP
(Vector) —> PC

Assembler Format: TRAPA #:4

Description: Initiates trap instruction exception processing. The vector number is calculated by adding 32 to the immediate value. See "4.6.4 Exception Processing" for details.

Operand Size: None

Instruction Format:

	4	4	
11110010	*	Imm.	*: Don't care

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.94 TST (Test)

Operation: (Destination) – 0

Assembler Format: TST <EAd>

Description: Subtract 0 from the destination operand specified by the effective address. Set N and Z according to the result. The destination operand is not affected.

Operand Size: Byte, word, long word

Instruction Format:

	2	
0 1 0 1 1 0	Sz	EAd

Condition Codes:

CX	N	Z	V	C
U	S	S	0	0

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

16.3.95 UNLK (Unlink)

Operation: Rn —> SP
@SP+ —> Rn

Assembler Format: UNLK Rn

Description: Load the contents of the specified register into the stack pointer. Load the long word data from the stack to the register. ULNK is used to release the stack area defined by the LINK instruction.

Operand Size: None

Instruction Format:

	4	4
11010011	*	Rn

*: Don't care

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.96 XCH (Exchange Registers)

Operand: Rnx <—> Rny

Assembler Format: XCH Rnx, Rny

Description: Exchange 32-bit data in register Rnx with 32-bit data in register Rny.

Operand Size: Long word

Instruction Format:

8 bits	4	4
1 0 1 1 0 0 1 1	Rnx	Rny

Condition Codes:

CX	N	Z	V	C
U	U	U	U	U

Available EA: None

16.3.97 XOR (Exclusive OR Logical)

Operation: (Destination) ⊕ (Source) —> (Destination)

Assembler Format: XOR <EAs>, <EAd>

Description: Exclusive OR the source and destination operands, and load the result to the destination location.

Operand Size: Byte, word, long word

Instruction Format:

	2		
100001	Sz	EAs	EAd

Condition Codes:

CX	N	Z	V	C
U	S	S	0	0

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
S													
D													

16.3.98 XORC (Exclusive OR Control Register)

Operation: CR ⊕ (Source) —> CR

Assembler Format: XORC <EAs>, CR

Description: Exclusive OR the CR register (CR) and source operand, and load the result to the destination location. Table 16-10 shows CR registers and CR codes.

Notes:
1. If BMR, GBNR, SR, EBR, RBR, USP, or IBR is specified as a CR register, XORC is handled as a privileged instruction.
2. If the CR field specifies a CR code not shown in Table 16-10, XORC performs exclusive OR operation between 0 and the source operand. At this time, CCR changes according to the source operand size specified by the bits 6-5 in the CR field. If bits 6-5 = 11, the source operand is assumed to be long word.
3. If this instruction is executed, the prefetch queue is reset and the instructions following XORC are fetched again.
4. No interrupt can be accepted just after XORC execution.

Operand Size: Same as CR.

Instruction Format:

	8	
11111010	CR	EAs
	$b_7, b_6, b_5 \ldots b_0$	

Condition Codes:

CR ≠ CCR and CR ≠ SR:

CX	N	Z	V	C
S	S	S	0	0

CR = CCR or CR = SR:

CX	N	Z	V	C
S*1	S*1	S*1	S*1	S*1

*1: CX same as bit 4 of result
N same as bit 3 of result
Z same as bit 2 of result
V same as bit 1 of result
C same as bit 0 of result

Available EA:

S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
D													

Section 17. Electrical Characteristics

17.1 Absolute Maximum Rating

Item	Symbol	Value	Unit
Supply voltage	Vcc	–0.3 to +7.0	V
Input voltage	Vin	–0.3 to Vcc + 0.3	V
Operating temperature	Topr	–20 to +75	°C
Storage temperature	Tstg	–55 to +150	°C

Note: DC and AC characteristics for 10 MHz version are preliminary values.

17.2 DC Characteristics (Vcc = 5 V ± 5%, Vss = 0 V, Ta = –20°C to +75°C)

Item	Symbol	Min	Type	Max	Unit	Condition
Input high voltage (EXTAL, $\overline{RES}$, $\overline{NMI}$, $\overline{IRQ0}$, $\overline{IRQ1}$)	V_{IH1}	Vcc – 0.6	-	Vcc + 0.3	V	-
Input high voltage (All inputs other than EXTAL, $\overline{RES}$, $\overline{NMI}$, $\overline{IRQ0}$, $\overline{IRQ1}$)	V_{IH2}	2.1	-	Vcc + 0.3	V	-
Input low voltage (EXTAL)	V_{IL0}	–0.3	-	0.6	V	
Input low voltage ($\overline{RES}$, $\overline{NMI}$, $\overline{IRQ0}$, $\overline{IRQ1}$)	V_{IL1}	–0.3	-	0.6	V	-
Input low voltage (All inputs other than EXTAL, $\overline{RES}$, $\overline{NMI}$, $\overline{IRQ0}$, $\overline{IRQ1}$)	V_{IL2}	–0.3	-	0.8	V	-
Output high voltage (All outputs)	V_{OH}	2.4	-	-	V	I_{OH} = –200 μA
		Vcc – 1.2	-	-		I_{OH} = –20 μA
Output low voltage ($\overline{DONE}$)	V_{OL1}	-	-	0.45	V	I_{OL} = 6.4 mA
Output low voltage (All outputs other than $\overline{DONE}$)	V_{OL2}	-	-	2.0	V	I_{OL} = 3.2 mA
Input leakage current (All inputs other than XTAL and EXTAL)	I_{IL}	-	-	2.0	μA	Vin = 0.7 to Vcc – 0.7V
Three-state leakage current	I_{TL}	-	-	2.0	μA	Vin = 0.7 to Vcc – 0.7V
Open-drain input current (off)	I_{OD}	-	-	1.0	μA	-
Current dissipation (Normal operation)	I_{CC}	-	60	90	mA	f = 8 MHz
		-	70	110		f = 10 MHz

DC Characteristics (Vcc = 5 V ± 5%, Vss = 0 V, Ta = –20°C to +75°C) (cont.)

Item	Symbol	Min	Type	Max	Unit	Condition
Current dissipation (Sleep mode)	I_{CC}	—	40	65	mA	f = 8 MHz
		—	45	80		f = 10 MHz
Current dissipation (System stop mode)		—	20	45	mA	f = 8 MHz
		—	25	55		f = 10 MHz
Pin capacity	C_P	—	—	15	pF	Vin = 0 V f = 1 MHz Ta = 25°C
Oscillation limit resistance	R_S	—	—	500	Ω	f = 1 MHz
		—	—	60		f = 4 MHz
		—	—	20		f = 8 MHz
		—	—	15		f = 10 MHz

17.3 AC Characteristics (Vcc = 5 V ±5%, Vss = 0 V, Ta = –20°C to +75°C)

Item		Symbol	8 MHz Min	8 MHz Max	10 MHz Min	10 MHz Max	Unit	Reference Timing Chart
Clock	Clock cycle time	t_{cyc}	125	1000	100	1000	ns	Figure 17-1
	Clock width low	t_{CL}	50	-	40	-	ns	
	Clock width high	t_{CH}	50	-	40	-	ns	
	Clock rise time	t_{Cr}	-	15	-	12	ns	
	Clock fall time	t_{Cf}	-	15	-	12	ns	
	E clock delay time	t_{ED}	-	15	-	13	ns	Figure 17-2
	Input clock rise time	t_{EXr}	-	25	-	10	ns	Figure 17-3
	Input clock fall time	t_{EXf}	-	25	-	10	ns	
CPU Bus Cycle	Address delay time 1	t_{AD1}	-	60	-	60	ns	Figure 17-4
	Address delay time 2	t_{AD2}	30	-	20	-	ns	
	Address delay time to high impedance 1	t_{ADZ1}	-	50	-	50	ns	
	Address delay time to high impedance 2	t_{ADZ2}	–5	-	–5	-	ns	
	Read data setup time	t_{RDS}	40	-	35	-	ns	
	Read data hold time	t_{RDH}	5	-	5	-	ns	
	Write data setup time	t_{WDS}	5	-	5	-	ns	Figure 17-5
	Write data hold time	t_{WDH}	10	-	10	-	ns	
	Write data delay time	t_{WDD}	-	60	-	55	ns	
	Setup time from $\overline{AS}$	t_{ASS}	10	-	7	-	ns	Figure 17-4
	Hold time from $\overline{AS}$ 1	t_{ASH1}	30	-	25	-	ns	
	Hold time from $\overline{AS}$ 2	t_{ASH2}	10	-	10	-	ns	
	$\overline{AS}$ delay time	t_{ASD}	-	45	-	45	ns	
	R/$\overline{W}$ delay time	t_{AD1}	-	60	-	60	ns	
	PF delay time	t_{AD1}	-	60	-	60	ns	

AC Characteristics (Vcc = 5 V ±5%, Vss = 0 V, Ta = –20°C to +75°C) (cont.)

Item		Symbol	8 MHz Min	8 MHz Max	10 MHz Min	10 MHz Max	Unit	Reference Timing Chart
CPU Bus Cycle (cont.)	S/$\overline{U}$ delay time	t_{AD1}	-	60	-	60	ns	Figure 17-4
	ST2, ST1, ST0 delay time	t_{AD1}	-	60	-	60	ns	
	PCS1, PCS0 delay time 1	t_{PCSD1}	-	60	-	60	ns	
	$\overline{HDS}$, $\overline{LDS}$ delay time 1	t_{DSD1}	-	45	-	45	ns	
	$\overline{HDS}$, $\overline{LDS}$ delay time 2	t_{DSD2}	-	45	-	45	ns	Figure 17-5
	$\overline{HDS}$, $\overline{LDS}$ delay time 3	t_{DSD3}	-	45	-	45	ns	Figure 17-4
	$\overline{AS}$ width high 1	t_{ASW1}	95	-	60	-	ns	Figure 17-5
	$\overline{AS}$ width high 2	t_{ASW2}	120	-	105	-	ns	Figure 17-6
	$\overline{HDS}$, $\overline{LDS}$ width low	t_{DSW}	90	-	65	-	ns	Figure 17-10
	$\overline{WAIT}$ setup time	t_{WTS}	35	-	35	-	ns	Figure 17-16
	$\overline{WAIT}$ hold time	t_{WTH}	10	-	7	-	ns	
	$\overline{BRTRY}$ setup time	t_{BRS}	40	-	40	-	ns	Figure 17-17
	$\overline{BRTRY}$ hold time	t_{BRH}	20	-	0	-	ns	
Refresh	Refresh address delay time	t_{RAD}	–	60	–	60	ns	Figure 17-18
	Refresh address delay time to high impedance	t_{RADZ}	–	50	–	50	ns	
	Refresh address setup time from $\overline{AS}$	t_{RASS}	10	–	7	–	ns	
	Refresh address hold time from $\overline{AS}$	t_{RASH}	30	–	25	–	ns	
Interrupt	$\overline{NMI}$ pulse width	t_{NMIW}	2.0	-	2.0	-	tcyc	Figure 17-19
	$\overline{IRQ0}$, $\overline{IRQ1}$ pulse width	t_{IRQW}	2.0	-	2.0	-	tcyc	Figure 17-20
	$\overline{IRQ0}$, $\overline{IRQ1}$ setup time	t_{IRQS}	50	-	50	-	ns	
	$\overline{IACK}$ delay time	t_{IACKD}	-	40	-	40	ns	Figure 17-21

AC Characteristics (Vcc = 5 V ±5%, Vss = 0 V, Ta = –20°C to +75°C) (cont.)

Item		Symbol	8 MHz Min	8 MHz Max	10 MHz Min	10 MHz Max	Unit	Reference Timing Chart
Bus Release	$\overline{\text{BREQ}}$ setup time	t_{BRQS}	40	-	35	-	ns	Figure 17-22
	$\overline{\text{BREQ}}$ hold time	t_{BRQH}	10	-	10	-	ns	
	PCS1, PCS0 delay time 2	t_{PCSD2}	-	85	-	85	ns	
	Address delay time to high impedence 3	t_{ADZ3}	-	50	-	45	ns	
	$\overline{\text{BACK}}$ delay time	t_{BACKD}	-	45	-	40	ns	
	$\overline{\text{AS}}$ input setup time	t_{ASIS}	40	-	40	-	ns	Figure 17-23
	Address setup time	t_{ADS}	40	-	35	-	ns	
	Address hold time	t_{ADH}	40	-	35	-	ns	
	$\overline{\text{WAIT}}$ output delay time	t_{WTOD}	-	40	-	40	ns	Figure 17-24
DMAC	$\overline{\text{DREQ}}$ setup time	t_{DRQS}	40	-	35	-	ns	Figure 17-25
	$\overline{\text{DREQ}}$ hold time	t_{DRQH}	10	-	10	-	ns	
	$\overline{\text{DREQ}}$ width low	t_{DRQW}	2.0	-	2.0	-	tcyc	
	$\overline{\text{DACK}}$ delay time 1	t_{DACD1}	-	50	-	45	ns	
	$\overline{\text{DACK}}$ delay time 2	t_{DACD2}	-	50	-	45	ns	
	$\overline{\text{DACK}}$ delay time 3	t_{DACD3}	-	50	-	45	ns	
	$\overline{\text{DONE}}$ delay time 1	t_{DOND1}	-	50	-	45	ns	
	$\overline{\text{DONE}}$ delay time 2	t_{DOND2}	-	50	-	45	ns	
	$\overline{\text{DONE}}$ setup time	t_{DONS}	40	-	35	-	ns	
	$\overline{\text{DONE}}$ hold time	t_{DONH}	10	-	5	-	ns	

AC Characteristics (Vcc = 5 V ±5%, Vss = 0 V, Ta = –20°C to +75°C) (cont.)

Item		Symbol	8 MHz Min	8 MHz Max	10 MHz Min	10 MHz Max	Unit	Reference Timing Chart
Timer	Timer clock width	t_{TMC1}	2.0	–	2.0	–	t_{cyc}	Figure 17-26
		t_{TMC2}	2.0	–	2.0	–	t_{cyc}	
		t_{TMCKW}	8.0	–	8.0	–	t_{cyc}	
	Timer clock setup time	t_{TMCKS}	30	–	25	–	ns	
	Timer trigger pulse width	t_{TMIWL}	2.0	–	2.0	–	t_{cyc}	
		t_{TMIWH}	2.0	–	2.0	–	t_{cyc}	
	Timer trigger setup time	t_{TMIS}	35	–	30	–	ns	
	Timer output delay time	t_{TMOD}	–	60	–	60	ns	
ASCI	Transmit clock cycle time	t_{Tcyc}	2.5	-	2.5	-	t_{cyc}	Figure 17-27
	Transmit clock width low	t_{TCLW}	1.0	-	1.0	-	t_{cyc}	
	Transmit clock width high	t_{TCHW}	1.0	-	1.0	-	t_{cyc}	
	Transmit clock fall time	t_{TCf}	-	50	-	50	ns	
	Transmit clock rise time	t_{TCr}	-	50	-	50	ns	
	Transmit clock delay time	t_{TCD}	-	60	-	50	ns	
	Transmit data delay time 1	t_{TDD1}	-	60	-	50	ns	
	Transmit data delay time 2	t_{TDD2}	1.5	2.5	1.5	2.5	t_{cyc}	
	Receive clock cycle time	t_{Rcyc}	2.5	-	2.5	-	t_{cyc}	
	Receive clock width low	t_{RCLW}	1.0	-	1.0	-	t_{cyc}	
	Receive clock width high	t_{RCHW}	1.0	-	1.0	-	t_{cyc}	
	Receive clock fall time	t_{RCf}	-	50	-	50	ns	
	Receive clock rise time	t_{RCr}	-	50	-	50	ns	
	Receive clock delay time	t_{RCD}	-	60	-	50	ns	
	Receive data setup time 1	t_{RDS1}	40	-	35	-	ns	
	Receive data hold time 1	t_{RDH1}	10	-	10	-	ns	

AC Characteristics (Vcc = 5 V ±5%, Vss = 0 V, Ta = –20°C to +75°C) (cont.)

Item		Symbol	8 MHz Min	8 MHz Max	10 MHz Min	10 MHz Max	Unit	Reference Timing Chart
ASCI (cont.)	Receive data setup time 2	t_{RDS2}	40	-	40	-	ns	Figure 17-27
	Receive data hold time 2	t_{RDH2}	10	-	10	-	ns	
	$\overline{RTS}$ delay time	t_{RTSD}	-	60	-	55	ns	
	$\overline{CTS}$ width low	t_{CTSLW}	2.0	-	2.0	-	t_{cyc}	
	$\overline{CTS}$ width high	t_{CTSHW}	2.0	-	2.0	-	t_{cyc}	
	$\overline{DCD}$ width low	t_{DCDLW}	2.0	-	2.0	-	t_{cyc}	
	$\overline{DCD}$ width high	t_{DCDHW}	2.0	-	2.0	-	tcyc	
Reset	$\overline{RES}$ setup time	t_{RESS}	30	-	30	-	ns	Figure 17-28
	$\overline{RES}$ hold time	t_{RESH}	0	-	0	-	ns	
	$\overline{RES}$ rise time	t_{Rr}	-	50 (Note)	-	50 (Note)	ms	
	$\overline{RES}$ fall time	t_{Rf}	-	50 (Note)	-	50 (Note)	ms	
	$\overline{RES}$ width low	t_{RESW}	12	-	12	-	t_{cyc}	

Note: $\overline{RES}$ rise and fall times are specified as 50 ns (max). However, if reset does not satisfy all other AC characteristics, its fall and rise time must be changed to satisfy these.

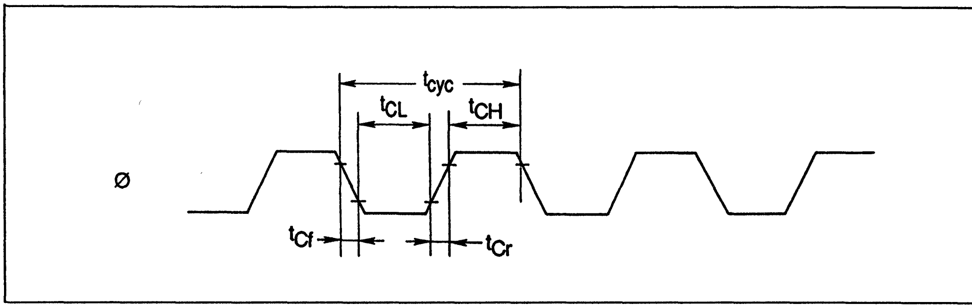


Figure 17-1. Ø Clock Timing

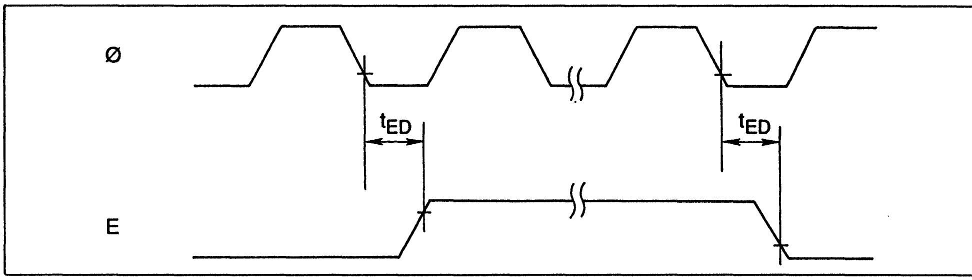


Figure 17-2. E Clock Timing

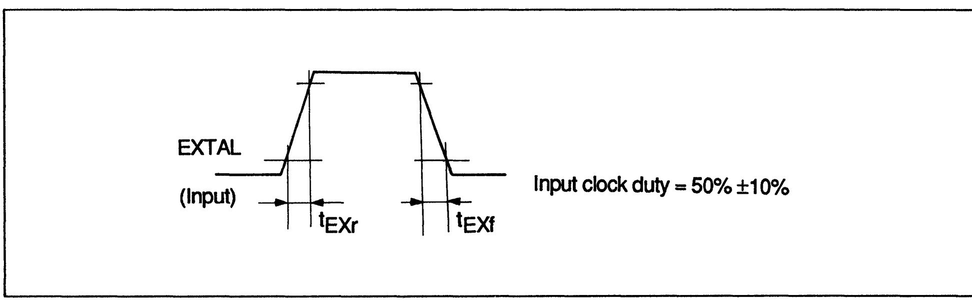


Figure 17-3. Input Clock Timing

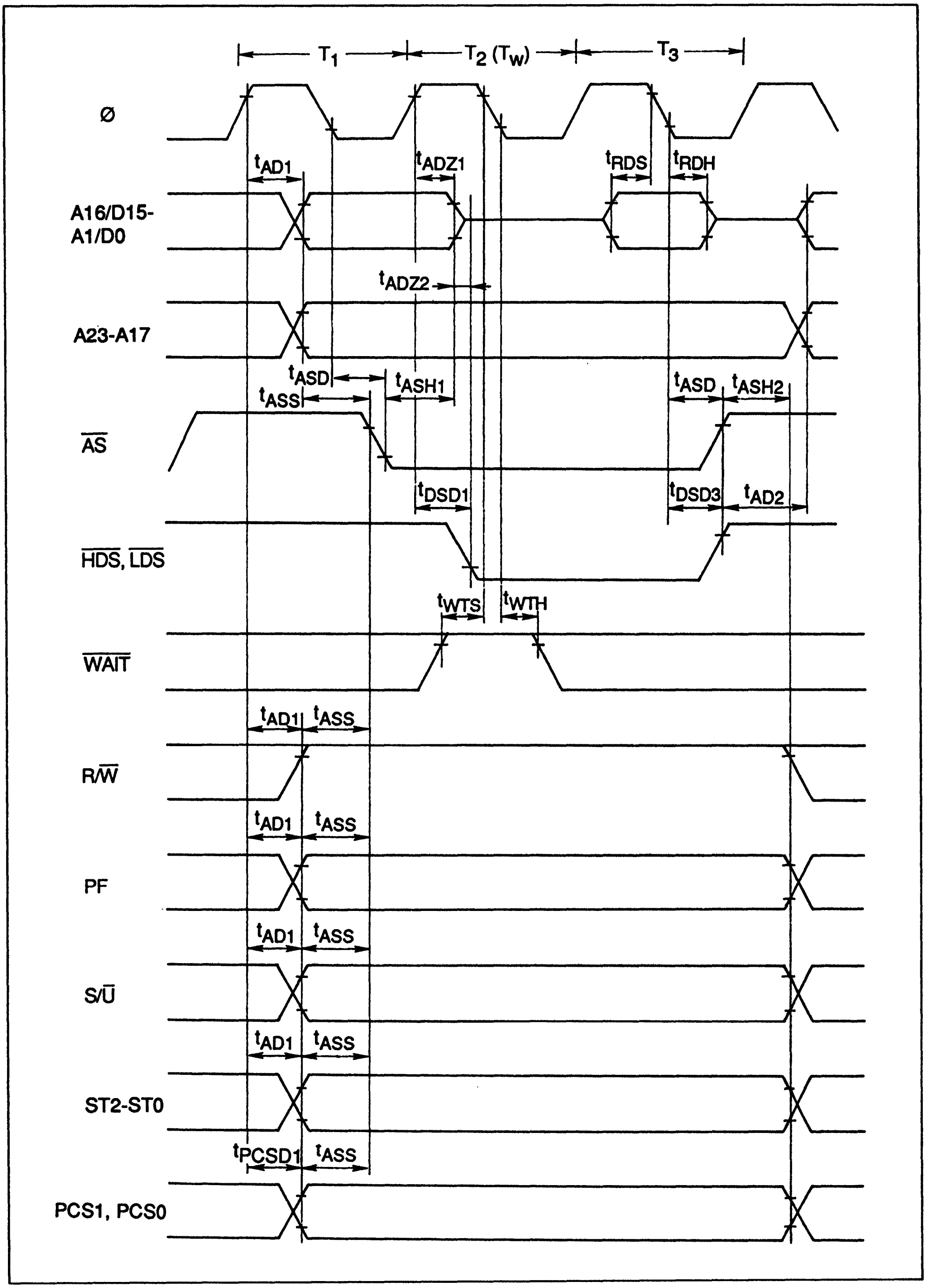


Figure 17-4. Read Cycle Timing (without Tp)

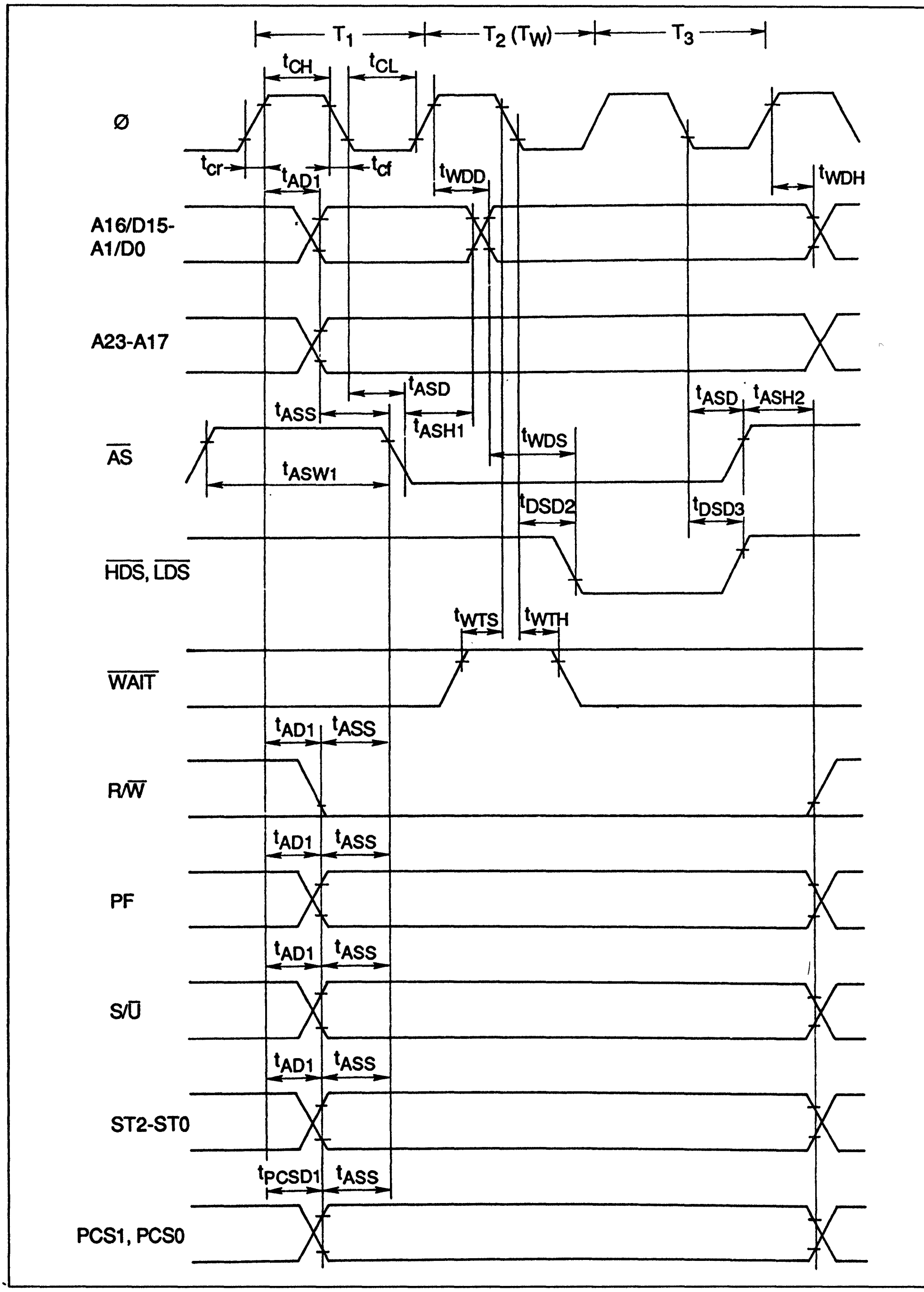


Figure 17-5. Write Cycle Timing (without Tp)

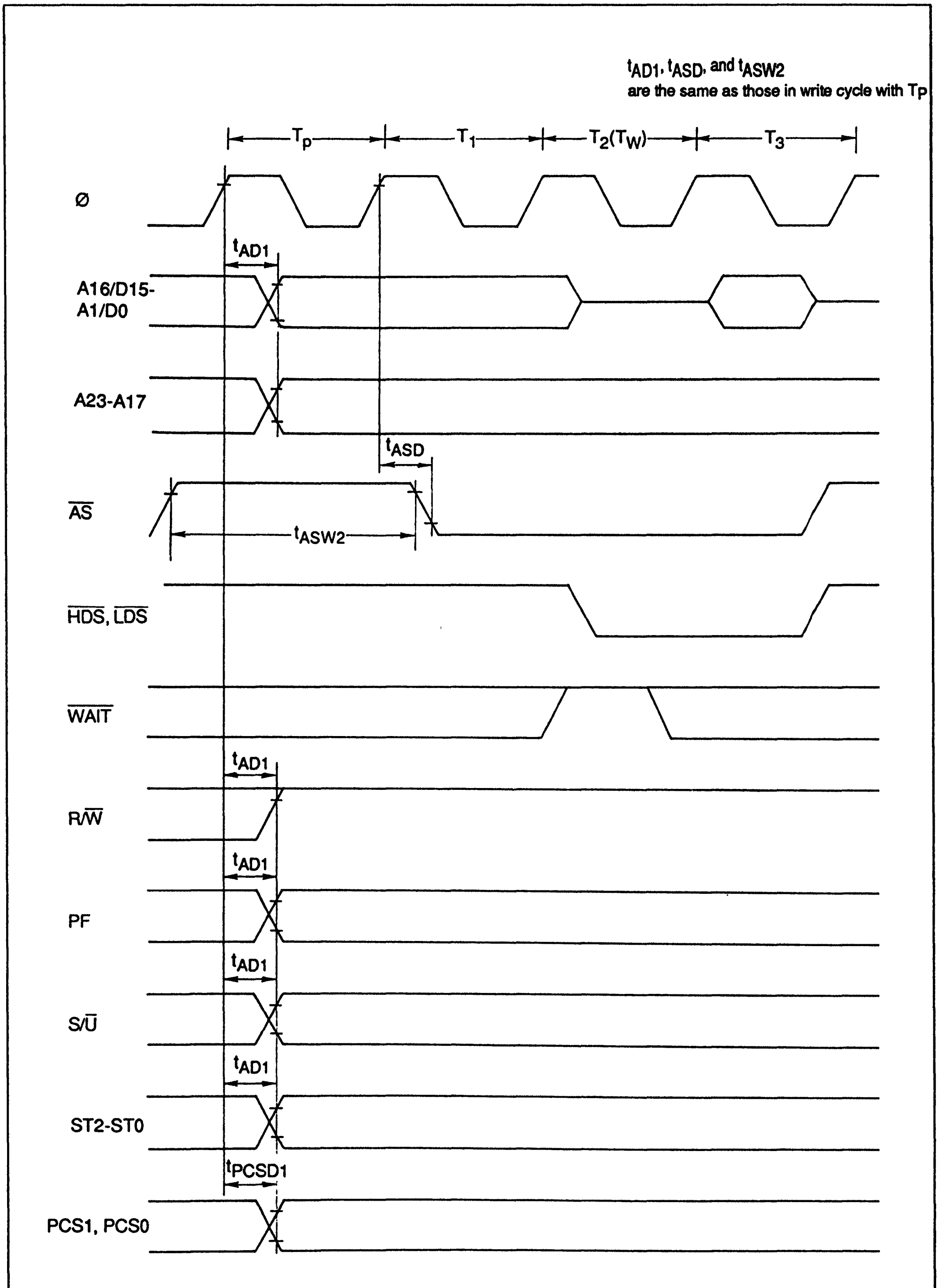


Figure 17-6. Read Cycle Timing (with Tp)

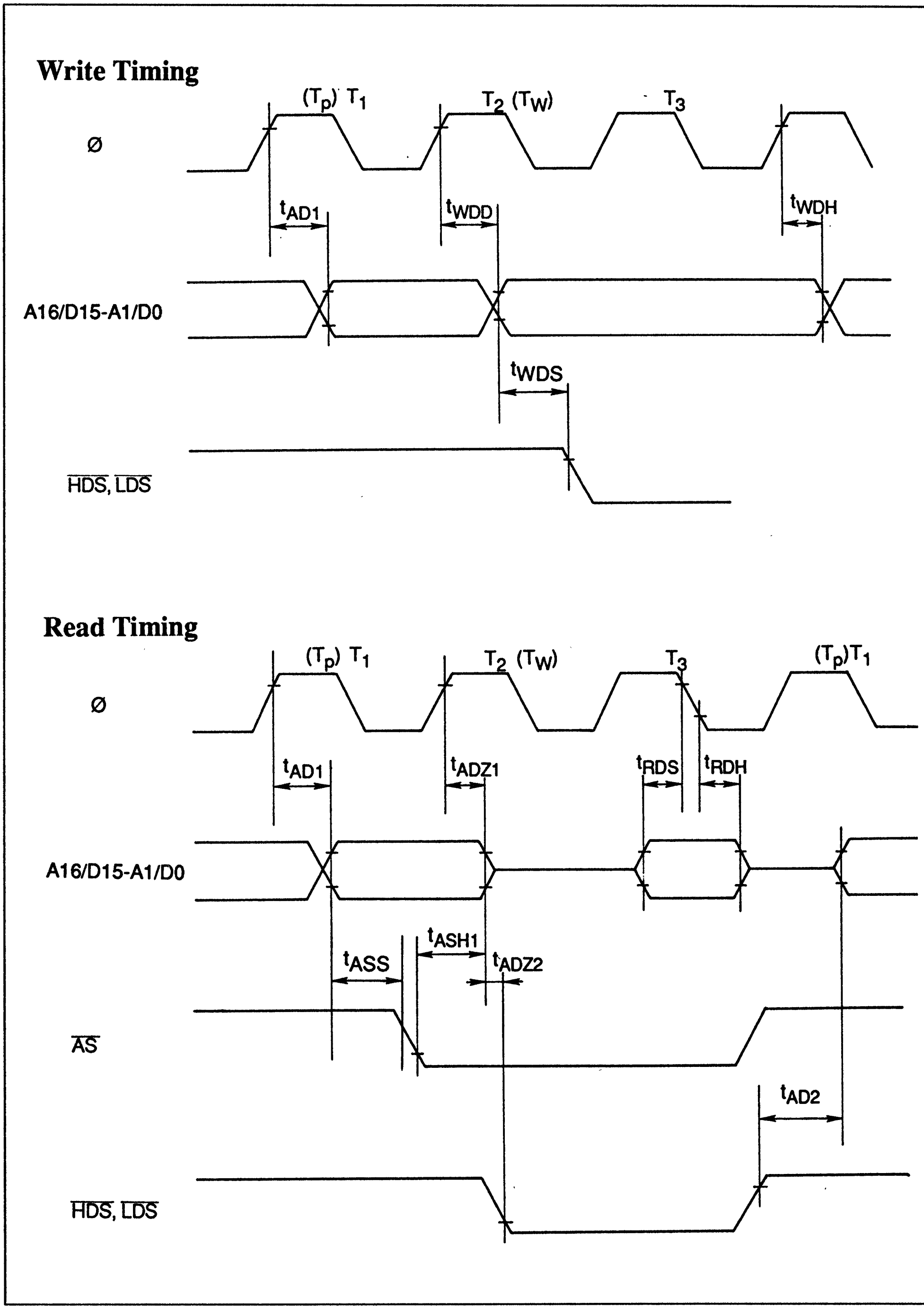


Figure 17-7. Address/Data Timing

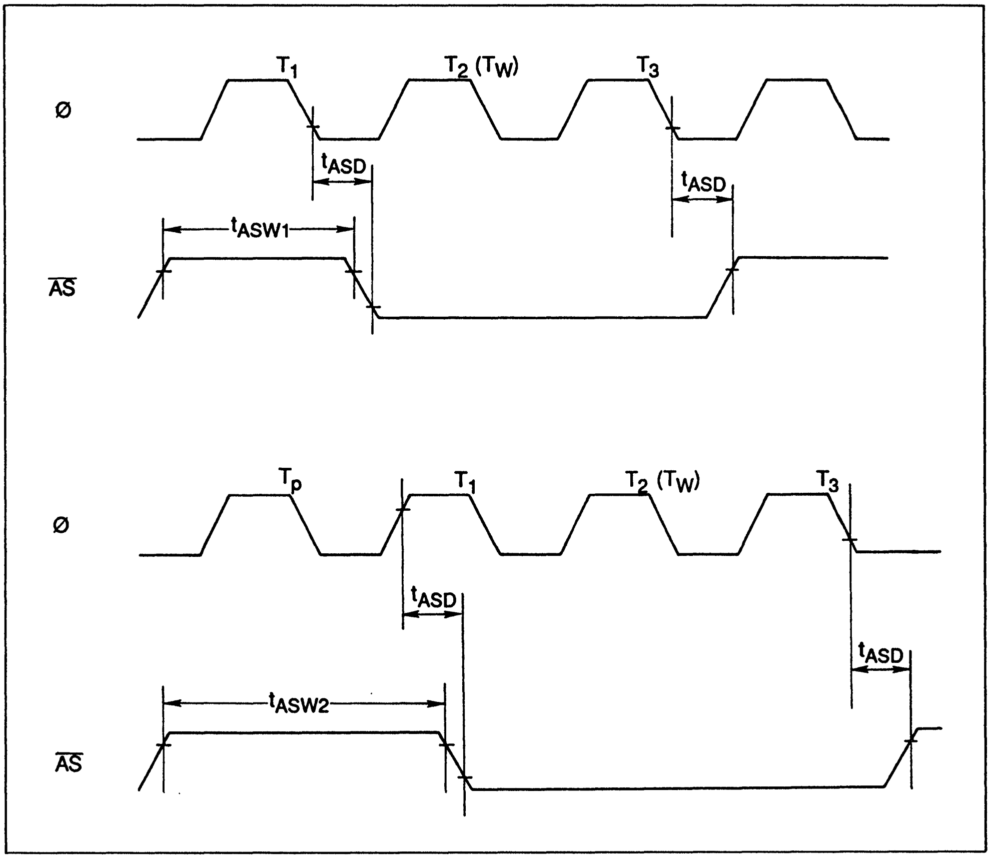


Figure 17-8. $\overline{\text{AS}}$ Timing

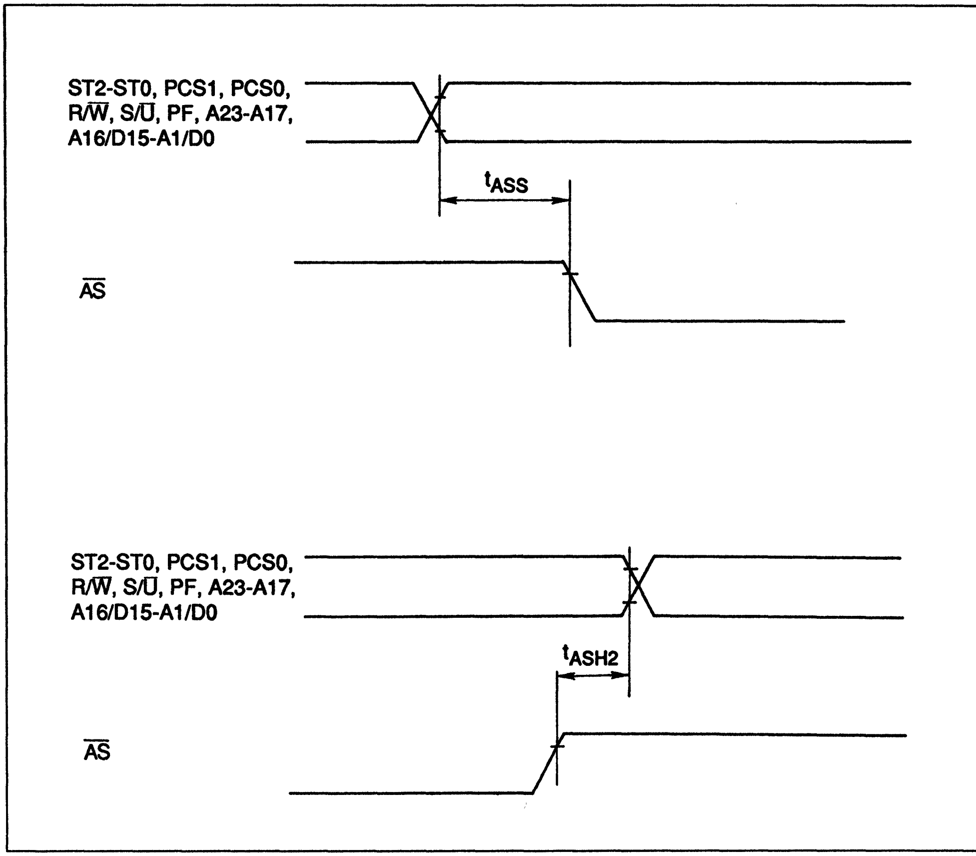


Figure 17-9. Control Signal Timing

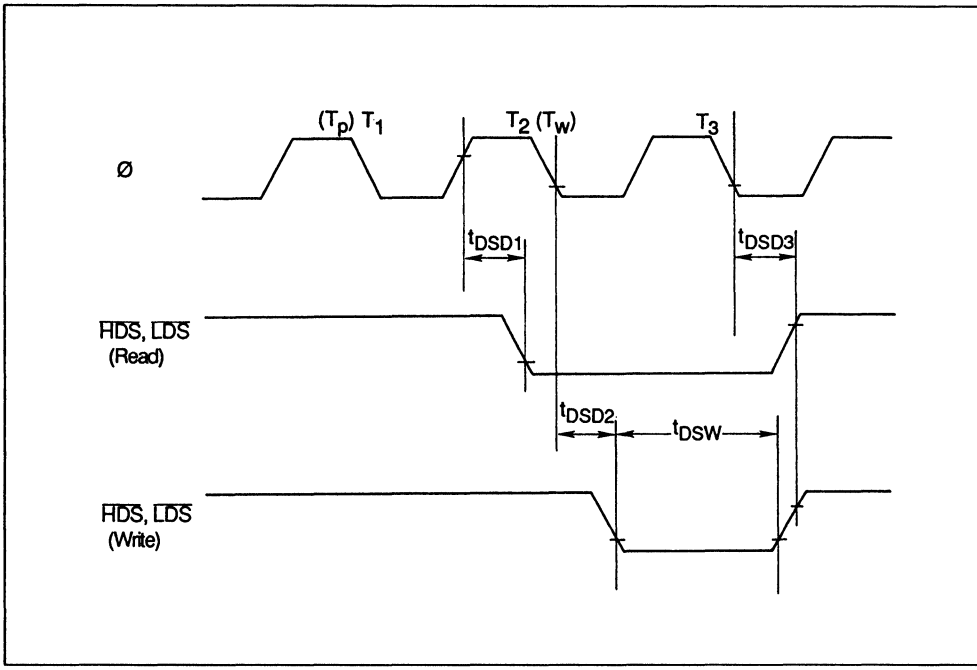


Figure 17-10. $\overline{HDS}$, $\overline{LDS}$ Timing

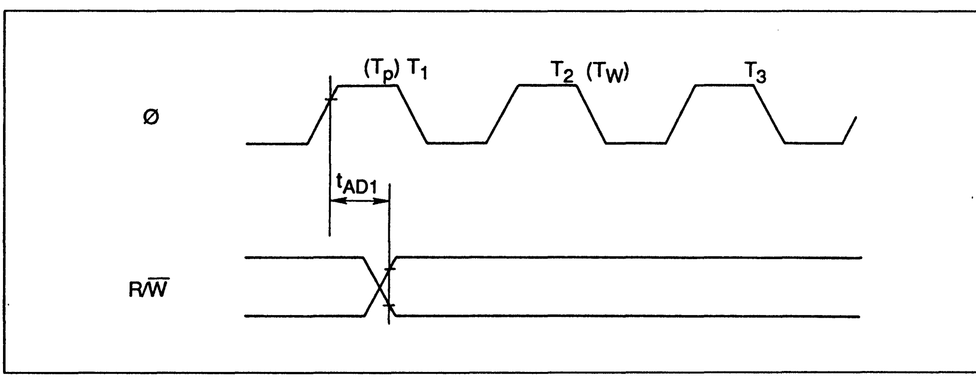


Figure 17-11. R/$\overline{W}$ Timing

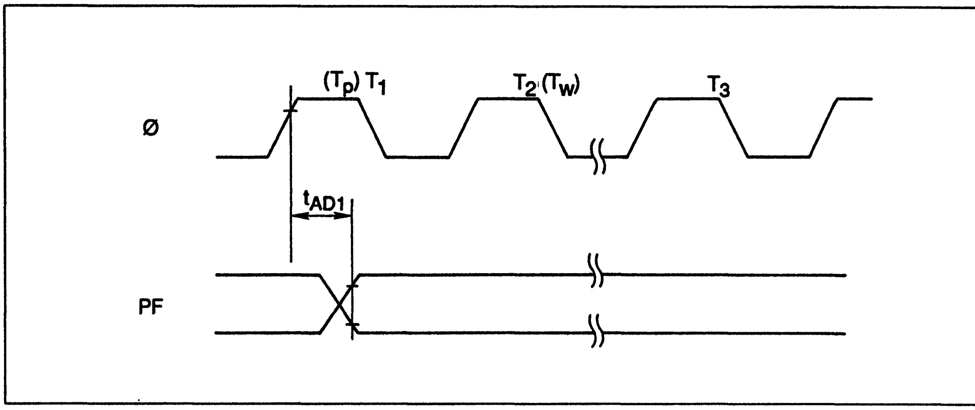


Figure 17-12. PF Timing

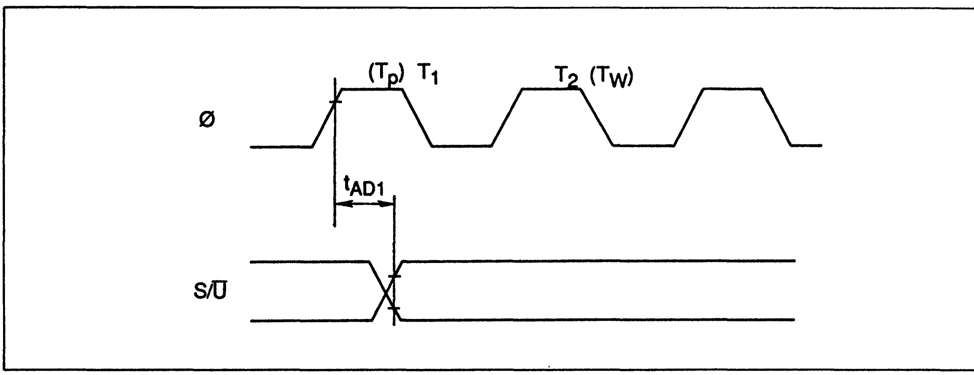


Figure 17-13. S/Ū Timing

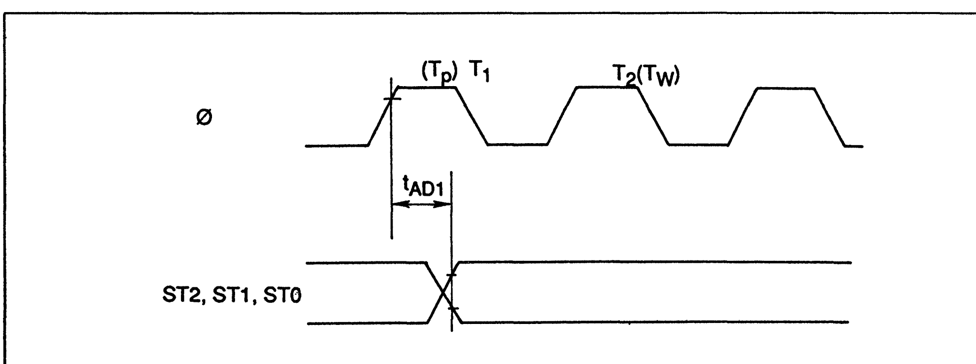


Figure 17-14. ST2, ST1, ST0 Timing

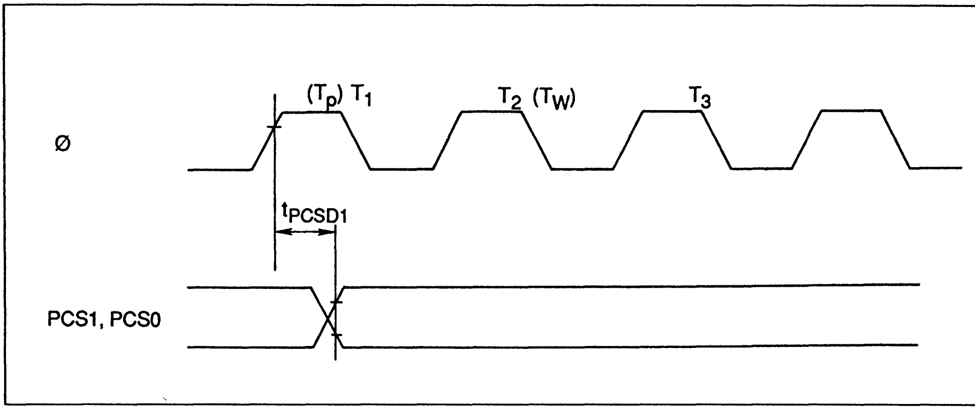


Figure 17-15. PCS1-PCS0 Timing

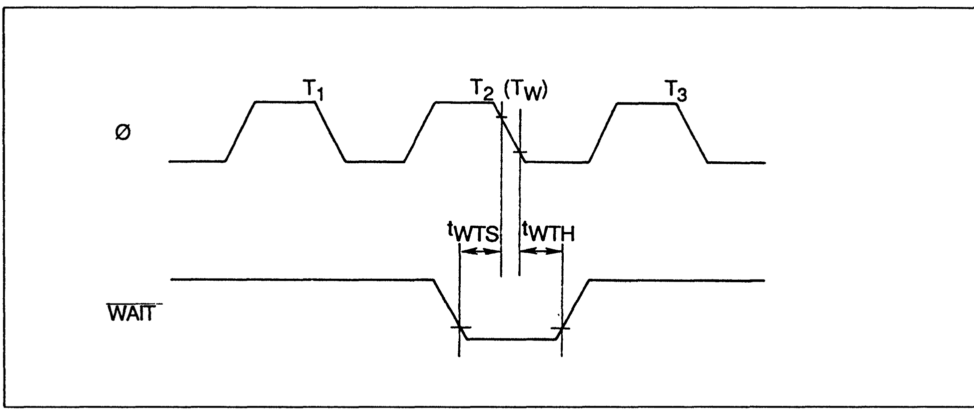


Figure 17-16. $\overline{\text{WAIT}}$ Input Timing

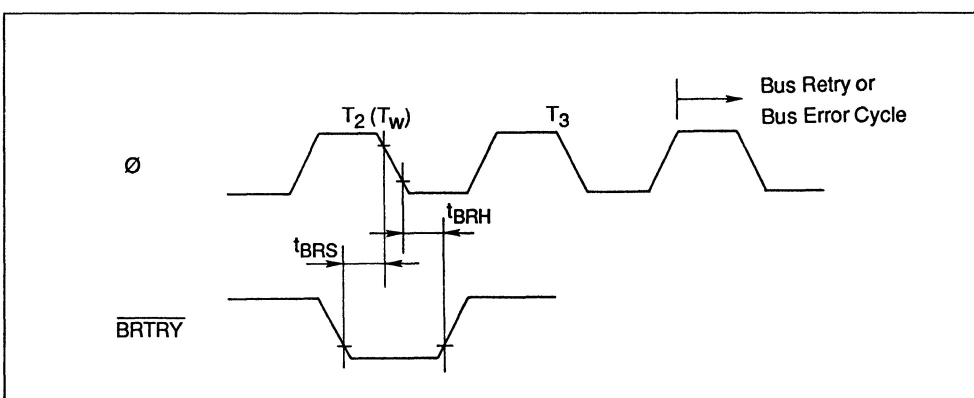


Figure 17-17. $\overline{\text{BRTRY}}$ Timing

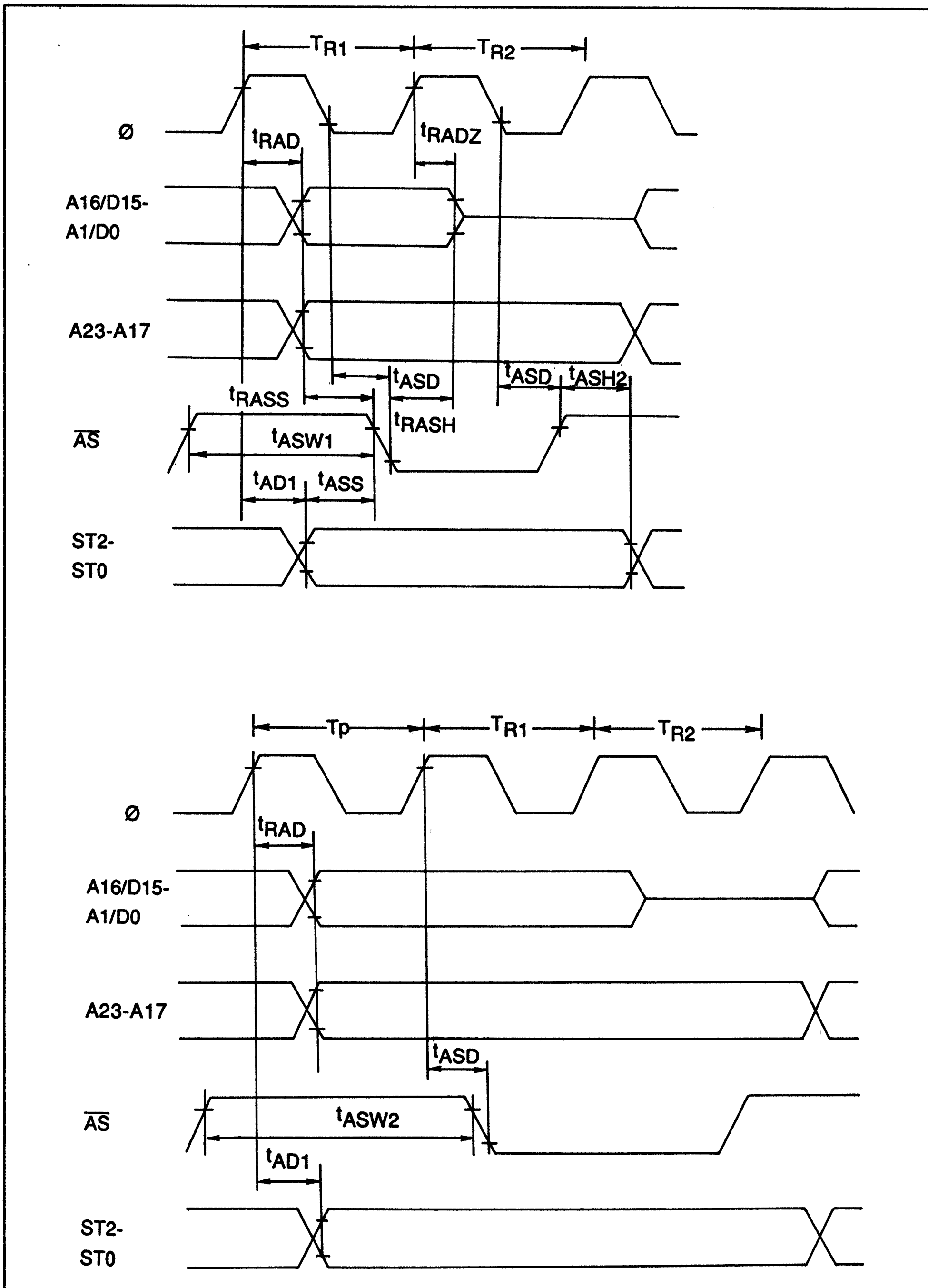


Figure 17-18. Refresh Timing (with or without Tw state)

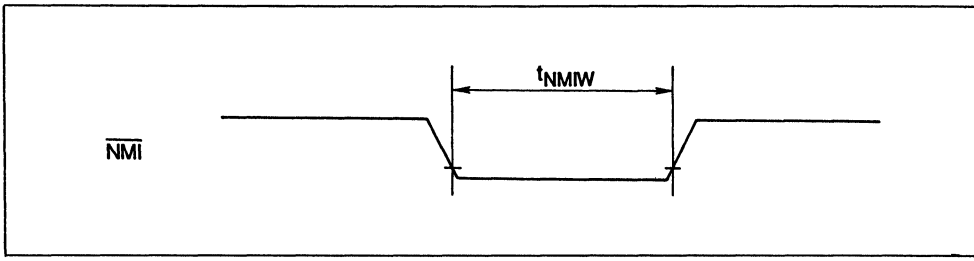


Figure 17-19. $\overline{NMI}$ Timing

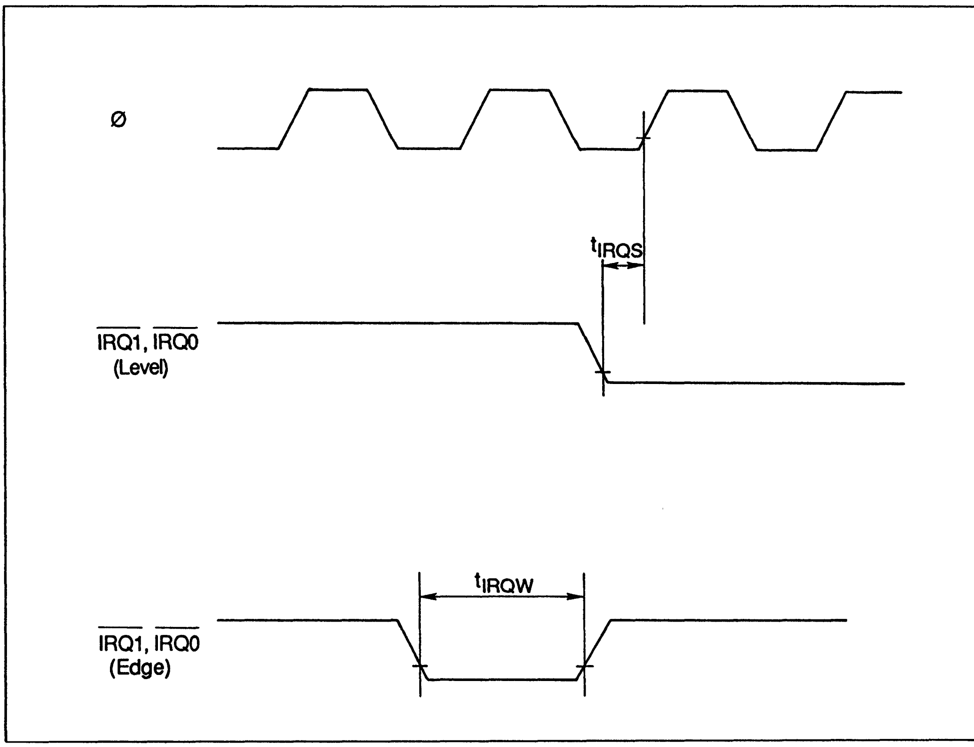


Figure 17-20. $\overline{IRQ0}$, $\overline{IRQ1}$ Timing

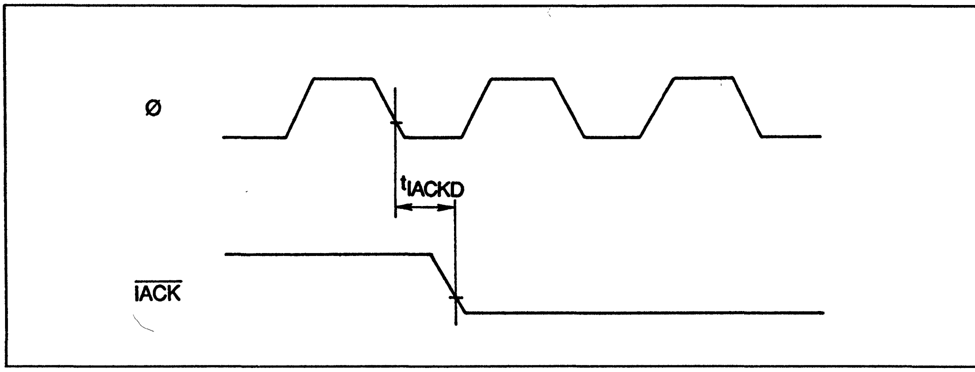


Figure 17-21. $\overline{\text{IACK}}$ Timing

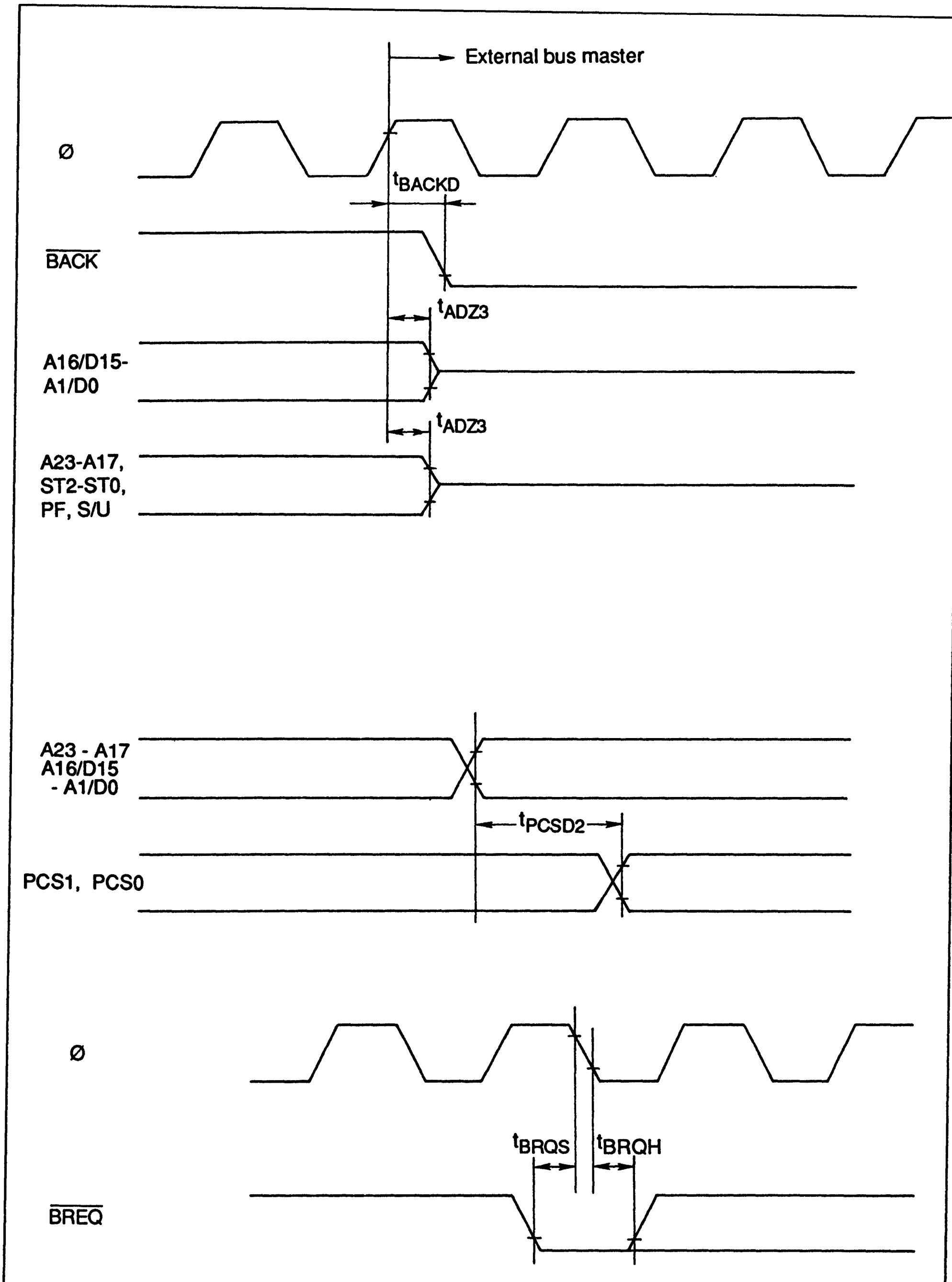


Figure 17-22. External Bus Master Bus Timing

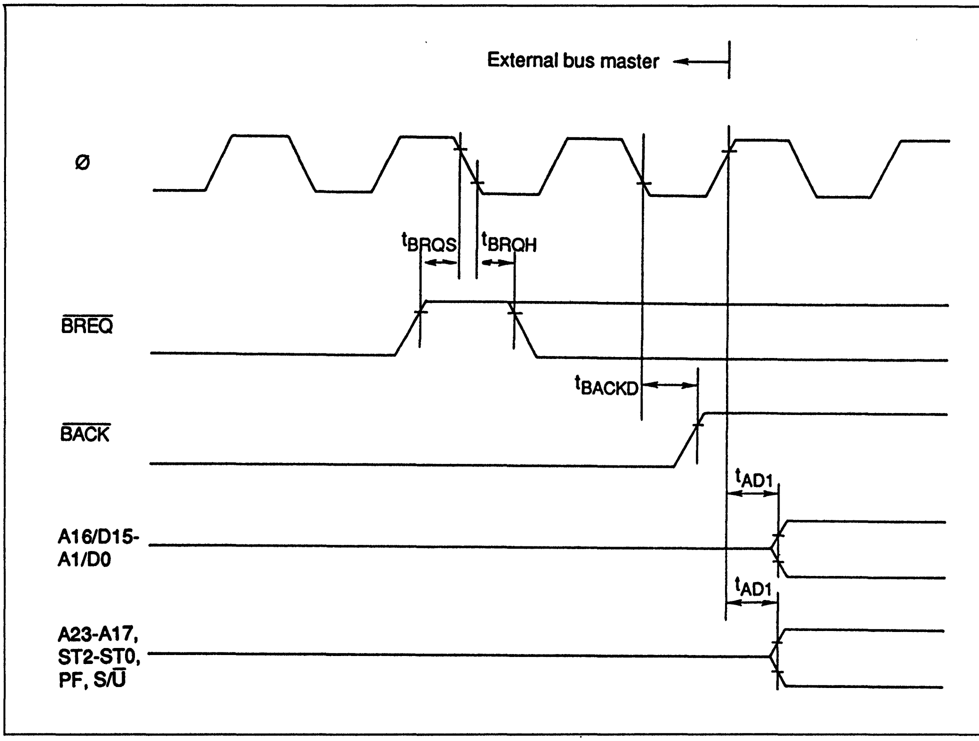


Figure 17-22. External Bus Master Bus Timing (cont.)

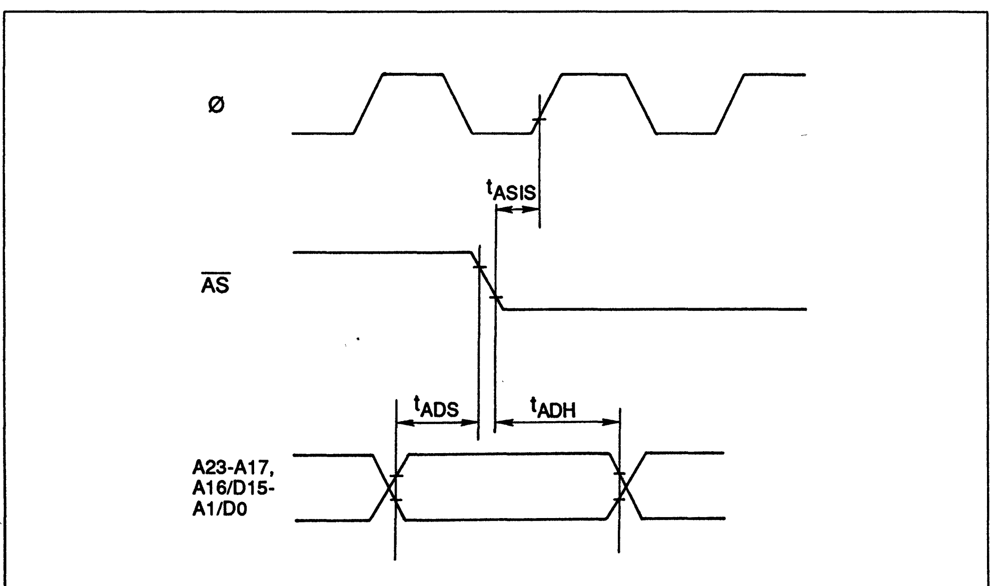


Figure 17-23. $\overline{AS}$, Address Input Timing (bus release)

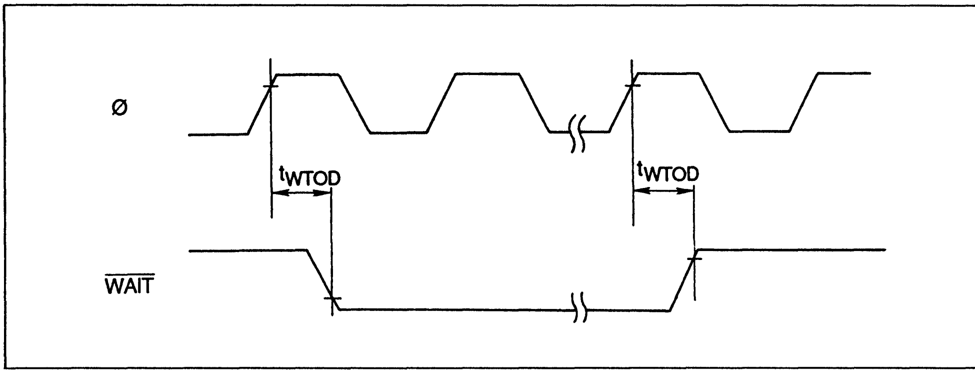


Figure 17-24. $\overline{WAIT}$ Output Timing

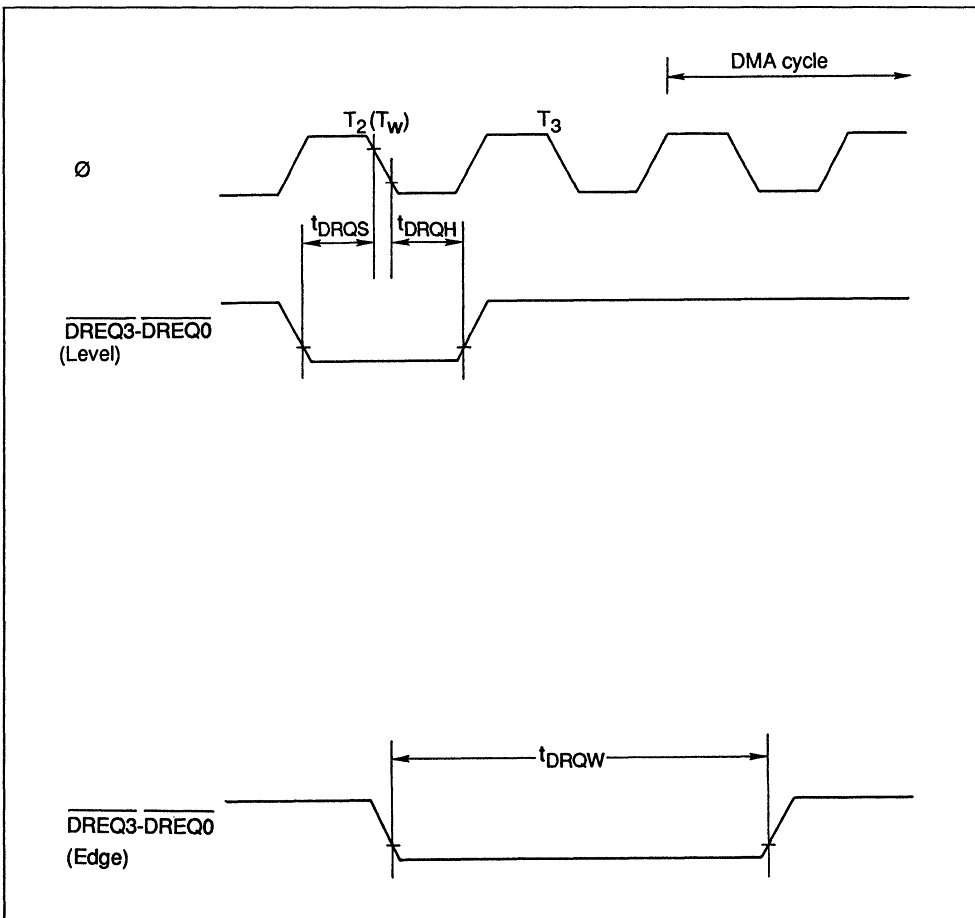


Figure 17-25. DMAC Timing

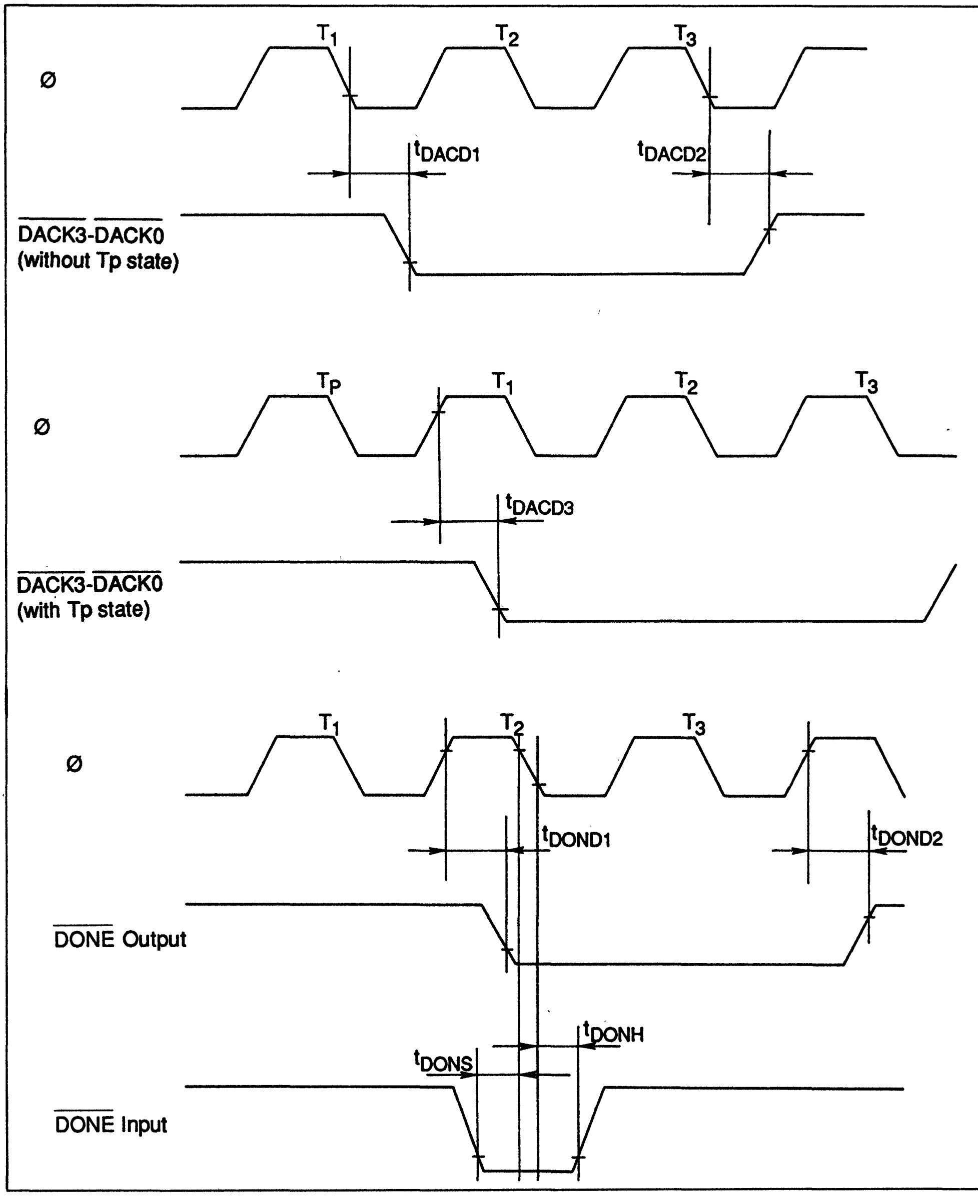


Figure 17-25. DMAC Timing (cont.)

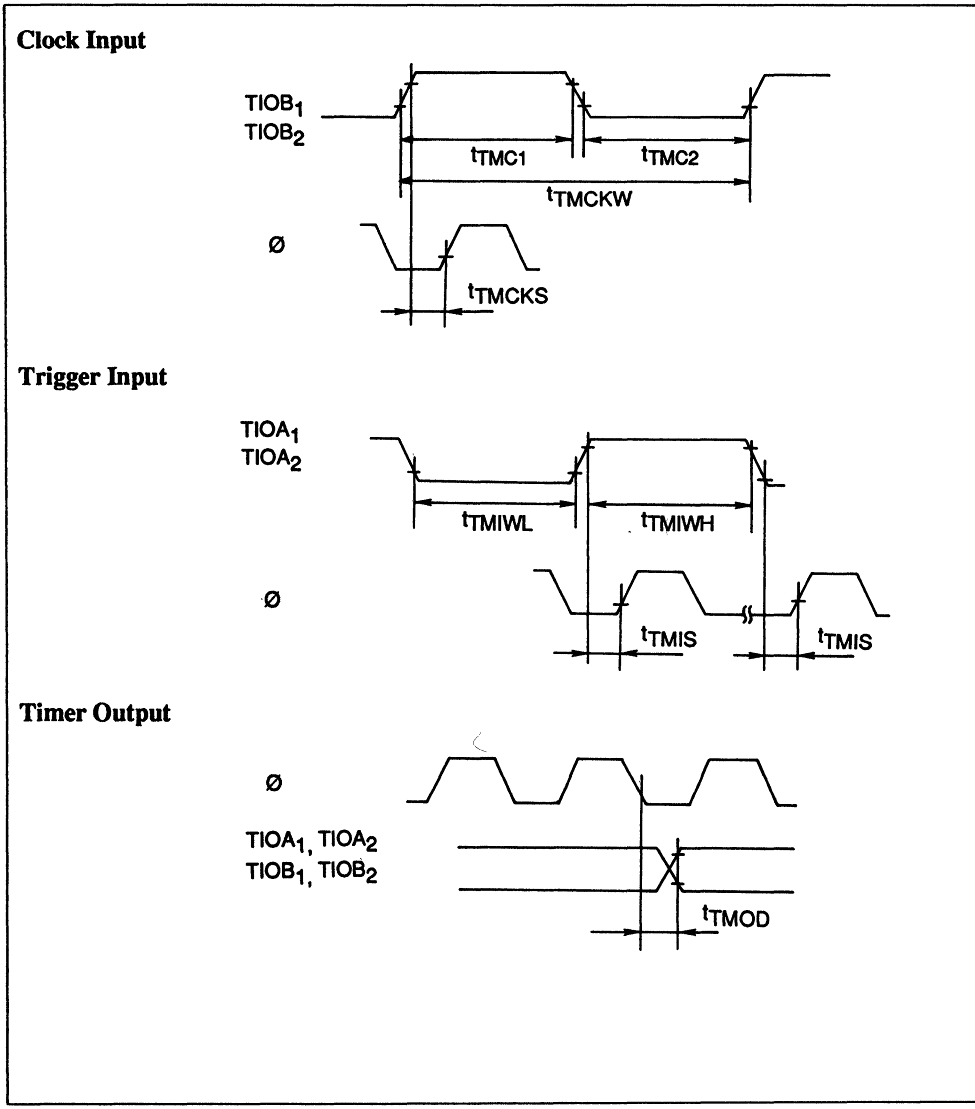
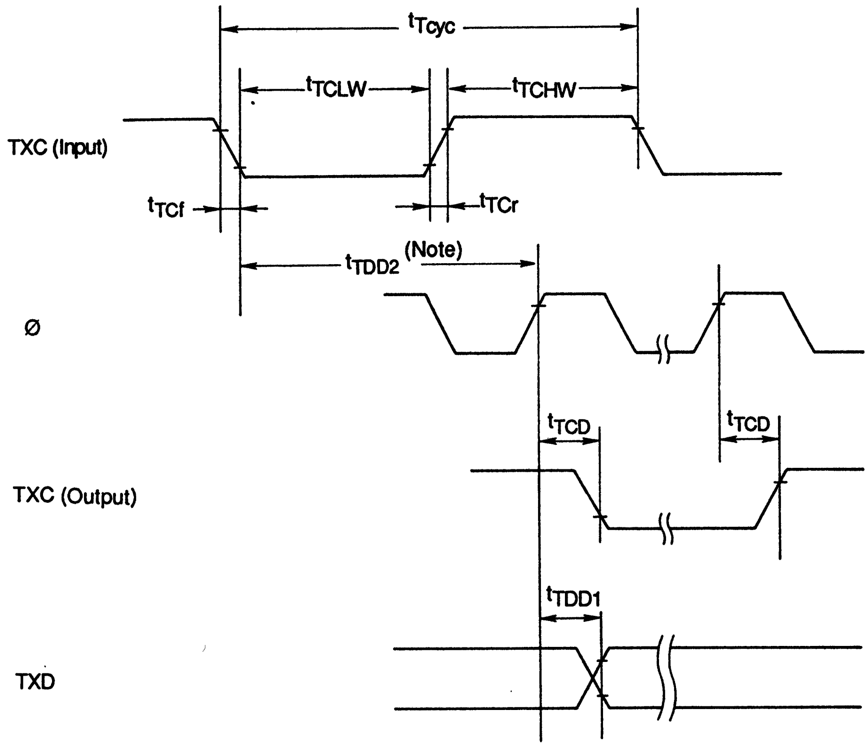


Figure 17-26. Timer Timing

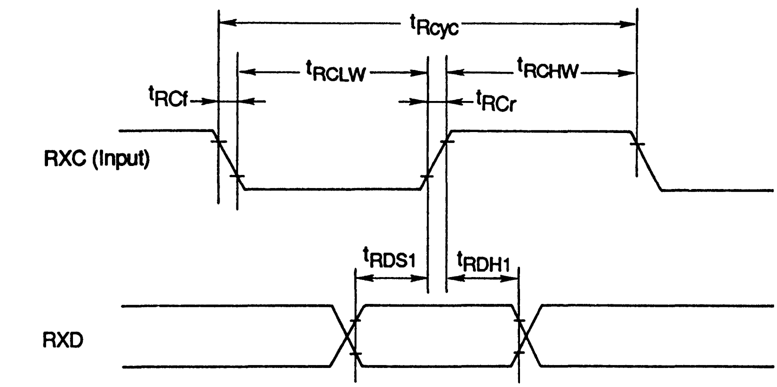
Transmission (TXC, TXD)



Note: TXD delay time in TXC input mode

Figure 17-27. ASCI Timing

Reception (RXC Input, RXD)



Reception (RXC Output, RXD)

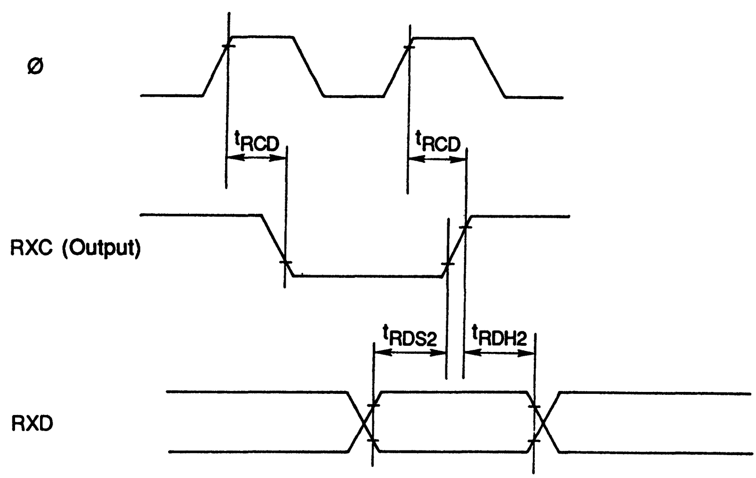


Figure 17-27. ASCI Timing (cont.)

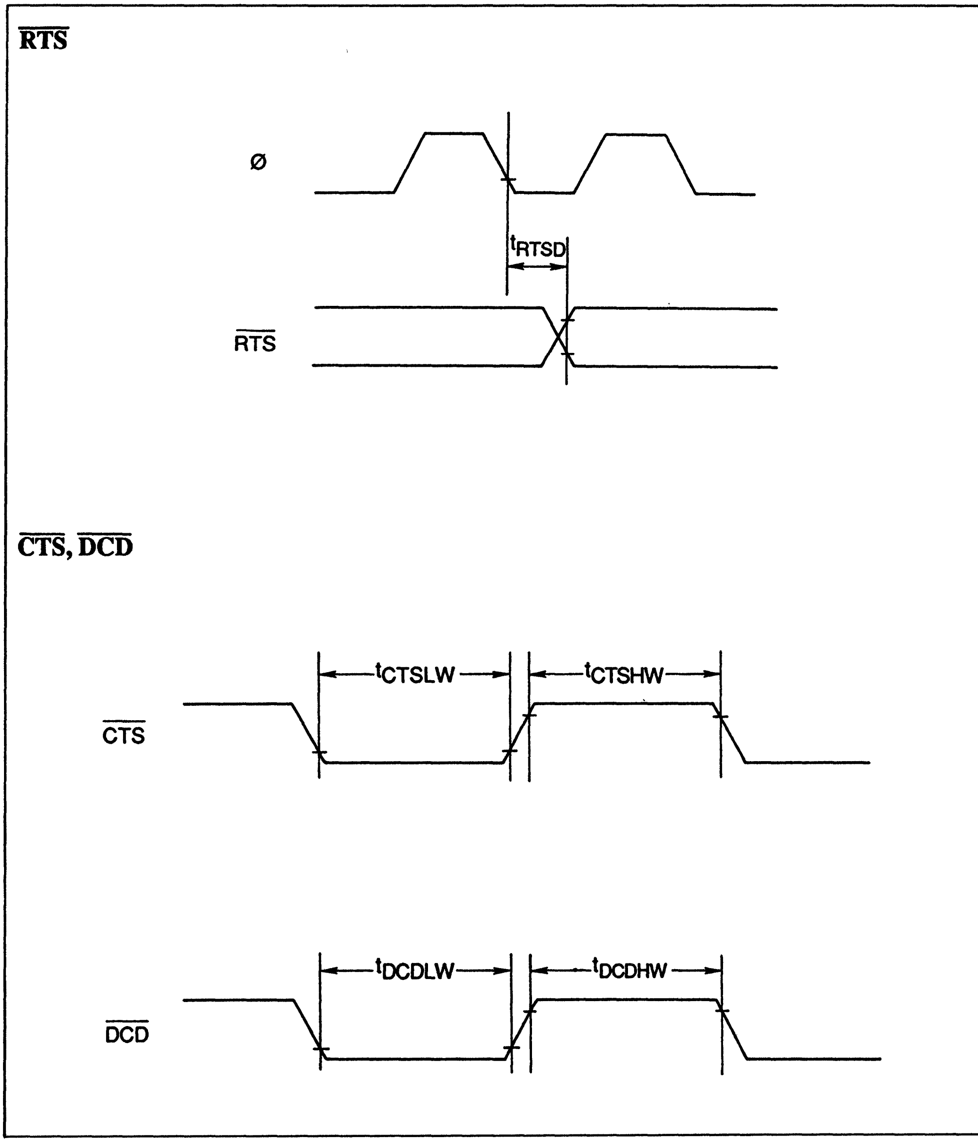


Figure 17-27. ASCI Timing (cont.)

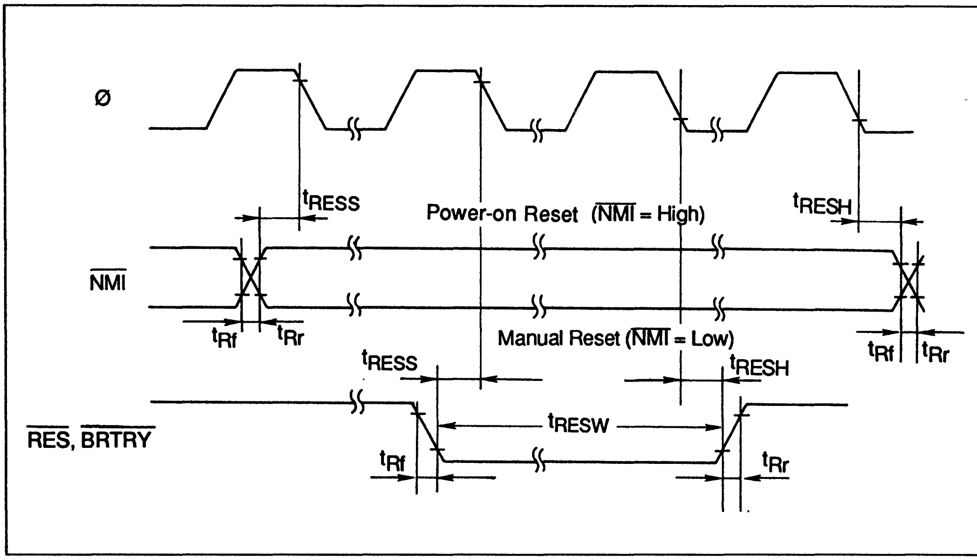


Figure 17-28. RES Timing

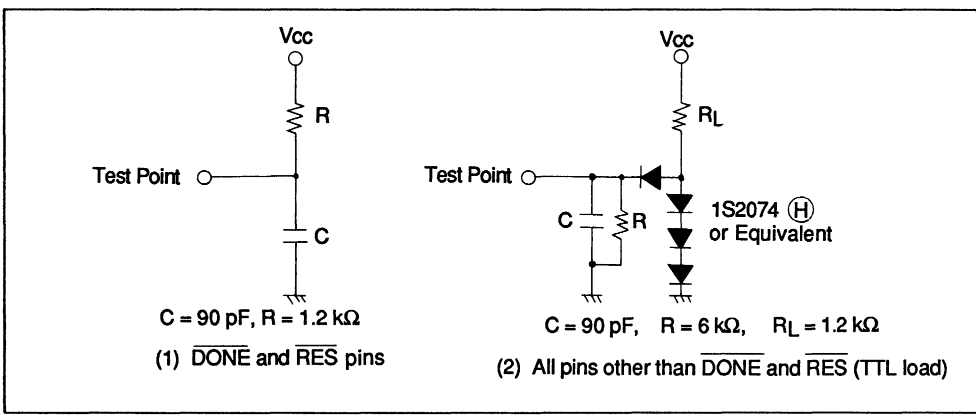


Figure 17-29. Bus Timing Test Loads

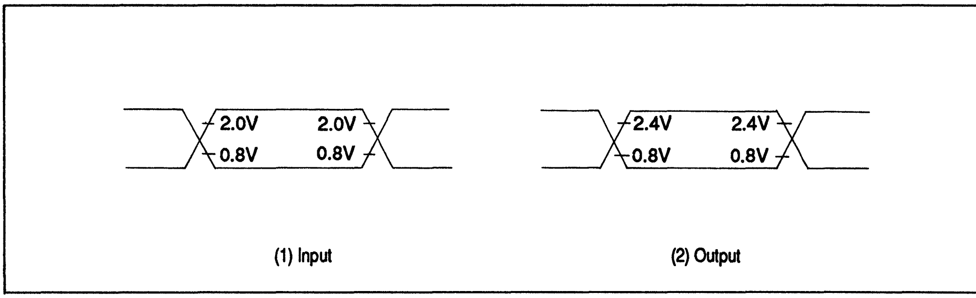


Figure 17-30. Reference Levels

Appendix A. Available EAs

The HD641016 available EAs are described in Table A-1. Note that only instructions containing an EA field are listed in tables.

Symbols: S : Source EA
: Destination EA
: EA available
: EA not available (If specified, illegal instruction exception processing begins.)

Table A-1. Available EAs

Instruction	S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
ADD:G	S													
ADD:G	D													
ADD:Q	D													
ADDS	S													
ADDS	D													
ADDX	S													
ADDX	D													
AND	S													
AND	D													
ANDC	S													
BCLR	D													
BFEXT	S													
BFEXT	D													
BFINS	S													
BFINS	D													
BFSCH	S													
BFSCH	D													
BNOT	D													
BSET	D													
BTST	D													
CLR	D													
CMP:G	S													
CMP:G	D													
CMP:Q	D													
CMPS	S													
CMPS	D													

Table A-1. Available EAs (cont.)

Instruction	S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
DADD	S													
DADD	D													
DIVXS	S													
DIVXS	D													
DIVXU	S													
DIVXU	D													
DNEG	D													
DSUB	S													
DSUB	D													
JMP	D													
JSR	D													
LDC	S													
LDM	S													
MOVA	S													
MOVA	D													
MOVF	D													
MOVFP	S													
MOVFP	D													
MOVFPE	S													
MOVFPE	D													

Table A-1. Available EAs (cont.)

Instruction	S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
MOV:G	S													
MOV:G	D													
MOV:Q	D													
MOVS	S													
MOVS	D													
MOVTP	S													
MOVTP	D													
MOVTPE	S													
MOVTPE	D													
MULXS	S													
MULXS	D													
MULXU	S													
MULXU	D													
NEG	D													
NEGX	D													
NOT	D													
OR	S													
OR	D													
ORC	S													
ROTL	D													
ROTR	D													
ROTXL	D													
ROTXR	D													

Table A-1. Available EAs (cont.)

Instruction	S/D	Rn	@Rn	@Rn+	@-Rn	#XXXX	@aaaa	@Rn*Sf	@(Xm, Rn)	@(Xm, PC)	@PC	@@Rn	<CRn>	<PRn>
SET	D													
SHAL	D													
SHAR	D													
SHLL	D													
SHLR	D													
STC	D													
STM	D													
SUB:G	S													
SUB:G	D													
SUBS	S													
SUBS	D													
SUBX	S													
SUBX	D													
SWAP	D													
TAS	D													
TST	D													
XOR	S													
XOR	D													
XORC	S													

Appendix B. Condition Code Affected

Symbols:

U: Unaffected

*: Undefined

S: Set normally, that is:

$CX = C$

$N = Rm$

$Z = \overline{Rm} \cdot ... \cdot \overline{R0}$

S*1: Set special (See instruction description)

Sm: MSB of the source operand

Dm: MSB of the destination operand

Rm: MSB of the result

n: Bit number

r: Shift count

See Table B-2 for symbols used in table.

Table B-1. Condition Codes Affected

Category	Type	Mne-monic	CX	N	Z	V	C	Special Definition
Arithmetic Operation	Add	ADD:G	S	S	S	S	S	A: See Table B-2.
		ADDS	U	U	U	U	U	
		ADDX	S	S	S*1	S	S	$V=Sm \cdot Dm \cdot \overline{Rm}+\overline{Sm} \cdot \overline{Dm} \cdot Rm$ $C=Sm \cdot Dm+\overline{Rm} \cdot Dm+Sm \cdot \overline{Rm}$, $Z=Z \cdot \overline{Rm} \cdot \overline{Rm-1} \ldots \cdot \overline{Ro}$
		DADD	S	*	S	*	S	C=Decimal Carry $Z=Z \cdot \overline{Rm} \cdot \overline{Rm-1} \ldots \cdot \overline{Ro}$
		ADD:Q	S	S	S	S	S	A: See Table B-2.
		ADD:R	S	S	S	S	S	A: See Table B-2
		ADD:RQ	S	S	S	S	S	A: See Table B-2.
	Sub-tract	SUB:G	S	S	S	S	S	B: See Table B-2.
		SUBS	U	U	U	U	U	
		SUBX	S	S	S*1	S	S	$V=\overline{Sm} \cdot Dm \cdot \overline{Rm}+Sm \cdot \overline{Dm} \cdot Rm$ $C=Sm \cdot \overline{Dm}+Rm \cdot \overline{Dm}+Sm \cdot Rm$, $Z=Z \cdot \overline{Rm} \cdot \overline{Rm-1} \ldots \cdot \overline{Ro}$
		DSUB	S	*	S	*	S	C=Decimal borrow $Z=Z \cdot \overline{Rm} \cdot \overline{Rm-1} \cdot \ldots \cdot \overline{Ro}$
		SUB:R	S	S	S	S	S	B: See Table B-2.
		SUB:RQ	S	S	S	S	S	B: See Table B-2.
	Multiply	MULXS	U	S	S	0	0	
		MULXU	U	S	S	0	0	
	Divide	DIVXS	U	S	S	S	0	V=Division overflow
		DIVXU	U	S	S	S	0	V=Division overflow
	Negate	NEG	S	S	S	S	S	$V=Dm \cdot Rm$, $C=Dm+Rm$
		NEGX	S	S	S*1	S	S	$V=Dm \cdot Rm$, $C=Dm+Rm$ $Z=Z \cdot \overline{Rm} \cdot \overline{Rm-1} \ldots \cdot \overline{Ro}$
		DNEG	S	*	S	*	S	C=Decimal Borrow $Z=Z \cdot \overline{Rm} \cdot \overline{Rm-1} \ldots \cdot \overline{Ro}$

Table B-1. Condition Codes Affected (cont.)

Category	Type	Mnemonic	CX	N	Z	V	C	Special Definition
Logical Operation	Logical AND	AND	U	S	S	0	0	
		ANDC	S*1	S*1	S*1	S*1	S*1	C: See Table B-2.
	Logical OR	OR	U	S	S	0	0	
		ORC	S*1	S*1	S*1	S*1	S*1	C: See Table B-2.
	Exclusive OR	XOR	U	S	S	0	0	
		XORC	S*1	S*1	S*1	S*1	S*1	C: See Table B-2.
	Invert	NOT	U	S	S	0	0	
Compare	Compare	CMP:G	U	S	S	S	S	B: See Table B-2.
		CMPS	U	S	S	S	S	B: See Table B-2.
		CMP:Q	U	S	S	S	S	B: See Table B-2.
		CMP:R	U	S	S	S	S	B: See Table B-2.
		CMP:RQ	U	S	S	S	S	B: See Table B-2.
Test	Test	TAS	U	S	S	0	0	
		TST	U	S	S	0	0	
Shift & Rotate	Arithmetic Shift	SHAL	S*1	S	S	S*1	S*1	$V=Dm\cdot(\overline{Dm-1}+\ldots+\overline{Dm-r})+\overline{Dm}\cdot(Dm-1+\ldots+Dm-r)$, C=Dm-r+1, V=C=0 (r=0)
		SHAR	S*1	S	S	0	S*1	C=Dr-1, C=0 (r=0)
	Logical Shift	SHLL	S*1	S	S	0	S*1	C=Dm-r+1, C=0 (r=0)
		SHLR	S*1	S	S	0	S*1	C=Dr-1, C=0 (r=0)
	Rotate	ROTL	U	S	S	0	S*1	C=Dm-r+1, C=0 (r=0)
		ROTR	U	S	S	0	S*1	C=Dr-1, C=0 (r=0)
	Rotate with Extend	ROTXL	S	S	S	0	S*1	C=Dm-r+1, C=CX (r=0)
		ROTXR	S	S	S	0	S*1	C=Dr-1, C=CX (r=0)

Table B-1. Condition Codes Affected (cont.)

Category	Type	Mnemonic	CX	N	Z	V	C	Special Definition
Branch, Jump, and Return	Branch	Bcc:G	U	U	U	U	U	
		BEQ	U	U	U	U	U	
		BNE	U	U	U	U	U	
		BRA	U	U	U	U	U	
		BSR	U	U	U	U	U	
		SCB	U	U	U	U	U	
	Jump	JMP	U	U	U	U	U	
		JSR	U	U	U	U	U	
	Return	RTD	U	U	U	U	U	
		RTE	S*1	S*1	S*1	S*1	S*1	*1: Corresponding bit of the stack word
		RTR	S*1	S*1	S*1	S*1	S*1	*1: Corresponding bit of the stack word
		RTS	U	U	U	U	U	
Single Operand	Clear	CLR	U	0	1	0	0	
	Extend	EXTS	U	S	S	0	0	
		EXTU	U	0	S	0	0	
	Set	SET	U	U	U	U	U	
	Swap	SWAP	U	S	S	0	0	
Transfer	Move	MOV:G	U	S	S	0	0	
		LDC	S*1	S*1	S*1	S*1	S*1	D: See Table B-2.
		STC	U	U	U	U	U	
		MOVTP	U	U	U	U	U	
		MOVTPE	U	U	U	U	U	
		MOVFP	U	U	U	U	U	
		MOVFPE	U	U	U	U	U	
		MOVA	U	U	U	U	U	

Table B-1. Condition Codes Affected (cont.)

Category	Type	Mnemonic	CX	N	Z	V	C	Special Definition
Transfer (cont.)	Move	MOVF	U	S	S	0	0	
		MOV:Q	U	S	S	0	0	
		MOV:R	U	S	S	0	0	
		MOV:RQ	U	S	S	0	0	
		MOVS	U	U	U	U	U	
		STM	U	U	U	U	U	
		LDM	U	U	U	U	U	
	Exchange	XCH	U	U	U	U	U	
String	String Data Transfer	SMOV	U	U	U	U	U	
		SSTR	U	U	U	U	U	
	String Data Compare	SCMP	U	U	S*1	U	U	*1: Set if this instruction completes when cc is not true, cleared otherwise.
		SSCH	U	U	S*1	U	U	*1: Set if this instruction completes when cc is not true, cleared otherwise.
Bit		BCLR	U	U	S*1	U	U	$Z = \overline{Dn}$
		BNOT	U	U	S*1	U	U	$Z = \overline{Dn}$
		BSET	U	U	S*1	U	U	$Z = \overline{Dn}$
		BTST	U	U	S*1	U	U	$Z = \overline{Dn}$
Bit Field		BFEXT	U	S*1	S*1	0	0	
		BFINS	U	S*1	S*1	0	0	
		BFSCH	U	S	S	0	0	
		BFMOV	U	U	U	U	U	

Table B-1. Condition Codes Affected (cont.)

Category	Type	Mne-monic	CX	N	Z	V	C	Special Definition
Miscel-laneous		TRAPA	U	U	U	U	U	
		TRAP	U	U	U	U	U	
		LINK	U	U	U	U	U	
		UNLK	U	U	U	U	U	
		NOP	U	U	U	U	U	
		ICBN	U	U	U	U	U	
		DCBN	U	U	U	U	U	
		CGBN	U	U	U	U	U	
		PGBN	U	U	U	U	U	
		SLEEP	U	U	U	U	U	
		RESET	U	U	U	U	U	

Table B-2. Special Definition

Type	Condition Codes Affected
A	$V = Sm \bullet Dm \bullet \overline{Rm} + \overline{Sm} \bullet \overline{Dm} \bullet Rm$ $C = Sm \bullet Dm + Rm \bullet Dm + Sm \bullet \overline{Rm}$
B	$V = \overline{Sm} \bullet Dm \bullet \overline{Rm} + Sm \bullet \overline{Dm} \bullet Rm$ $C = Sm \bullet \overline{Dm} + Rm \bullet \overline{Dm} + Sm \bullet Rm$
C	When CR ≠ CCR and CR ≠ SR, CX N Z V C U S S 0 0 When CR = CCR or CR = SR, same as the corresponding bit of the result
D	When CR ≠ CCR and CR ≠ SR, unaffected When CR = CCR or CR = SR, same as the corresponding bit of the result

Appendix C. Instruction Cycles

1. The number of instruction cycles listed here assumes the best case. Accordingly, they may increase depending on the instruction combination.

2. The number of instruction cycles are calculated as follows:

   The number of fixed instruction cycles + the number of EA calculation cycles

3. See Tables C-2 through C-15 for details on the fixed instruction cycles.

4. See Tables C-2 through C-14 for details on the source and destination EA types.

5. See Table C-1 for details on the EA calculation cycles.

6. Instruction execution cycles are calculated as follows.

<u>ADD:G.B</u> <u>R1,</u> <u>@(WORK.B, R2)</u>
1 2 3

First, find "ADD:G" in the instruction cycle table. "B" in the Sz field indicates the number of instruction execution cycle. Fixed instruction cycles indicated in the B.CYCLE field are 4. Both EA (S) and EA (D) types are type b.

No.	OP	Sz	Base Cycles	Type EA(S)	Type EA(D)
1	ADD:G ADDX SUB:G SUBX	B	4	b	b
		W	4	b	b
		L	4	c	c
2	ADD:Q	B	5		b
		W	5		b
		L	5		c
3	ADD:R SUB:R	B	5		
		W	5		
		L	5		

Next, determine the instruction execution cycles for EA (S) and EA (D) from the EA mode cycle table. Then, there is 1 instruction execution cycle for EA (S) and 9 instruction cycles for EA (D).

EA Mode		Type a	Type b
Rn		1	1
@Rn	disp = 0	5	6
	disp = 8	8	9
	disp = 16	11	12
	disp = 32	16	17
@Rn+		5	6
@-Rn		5	6

EA Mode		Type a	Type b
Rn		1	1
@Rn	disp = 0	5	6
	disp = 8	8	9
	disp = 16	11	12
	disp = 32	16	17
@Rn+		5	6
@-Rn		5	6

Finally, add these instruction execution cycles to get total number of instruction execution cycles:

4 + 1+ 9 = 14 (cycles)

7. Instruction execution cycles in Tables C-2 to C-14 are calculated under the following conditions:

- One bus cycle consists of three system clocks.
- Word or long word data is aligned on word boundaries.
- Opcodes and the EA fields are prefetched before the CPU fetches them.

Accordingly, instruction execution cycles will change in the following cases:

- Tw states are inserted.
- Word or long word data is not aligned on word boundaries.
- Instruction prefetch cycle is required since instruction prefetch is not performed synchronously with the CPU operations.
- Internal RAM is accessed.

Table C-1. EA Mode Cycle

EA Mode			Type a	Type b	Type c
Rn			1	1	1
@Rn		disp = 0	5	6	9
		8	8	9	12
		16	11	12	15
		32	16	17	20
@Rn+			5	6	9
@-Rn			5	6	9
#xxxx		8	3	3	3
		16	7	7	7
		32	12	12	12
@aaaa		8	6	7	10
		16	9	10	13
		32	14	15	18
@Rn * Sf		disp = 0	6	7	10
		8	9	10	13
		16	12	13	16
		32	17	18	21
@(Xm, Rn)	Xm: W	disp = 0	10	11	14
		8	13	14	17
		16	16	17	20
		32	21	22	25
	Xm: LW	disp = 0	9	10	13
		8	12	13	16
		16	15	16	19
		32	20	21	24
@(Xm, PC)	Xm: W	disp = 0	10	11	14
		8	13	14	17
		16	16	17	20
		32	21	22	25
@PC	Xm: L	disp = 0	9	10	13
		8	12	13	16
		16	15	16	19
		32	20	21	24
@PC		disp = 0	6	7	10
		8	8	9	12
		16	11	12	15
		32	16	17	20

Table C-1. EA Mode Cycle (cont.)

EA Mode			Type a	Type b	Type c
@@Rn	disp1 = 8	disp 2 = 8	20	21	24
		32	28	29	32
	disp 1 = 32	disp 2 = 8	28	29	32
		32	36	37	40
<CRn>			2	2	2
<PRn>			2	2	2

Table C-2. Arithmetic Operation

No.	OP	Sz	Base Cycles	Type EA (S)	Type EA (D)	Note
1	ADD: G, ADDX, SUB: G, SUBX	B	4	b	b	
		W	4	b	b	
		L	4	c	c	
2	ADD: Q	B	5	-	b	
		W	5	-	b	
		L	5	-	c	
3	ADD: R, SUB: R	B	5	-	-	
		W	5	-	-	
		L	5	-	-	
4	ADD: RQ, SUB: RQ	B	4	-	-	
		W	4	-	-	
		L	4	-	-	
5	ADDS, SUBS	B	4	b	c	
		W	4	b	c	
		L	4	c	c	
6	MULXS	B	36	b	b	
		W	61	b	b	
7	MULXU	B	35	b	b	
		W	58	b	b	
8	DIVXS	B	< 61	b	b	
			38	b	b	Divide by 0
		W	< 80	b	c	
			38	b	c	Divide by 0
9	DIVXU	B	< 51	b	b	
			37	b	b	Divide by 0
		W	< 70	b	c	
			37	b	c	Divide by 0
10	DADD, DSUB	B	5	b	b	
11	NEG, NEGX	B	3	-	b	
		W	3	-	b	
		L	3	-	c	
12	DNEG	B	4	-	b	

Table C-3. Logical Operation

No.	OP	Sz	Base Cycles	Type EA (S)	Type EA (D)	Note
1	AND, OR, XOR	B	4	b	b	
		W	4	b	b	
		L	4	c	c	
2	ANDC, ORC, XORC	B	10	-	b	
		W	10	-	b	
		L	10	-	c	
3	CMP: G, CMPS	B	4	a	a	
		W	4	a	a	
		L	4	b	b	
4	CMP: Q	B	5	-	b	
		W	5	-	b	
		L	5	-	c	
5	CMP: R	B	5	-	-	
		W	5	-	-	
		L	5	-	-	
6	CMP: RQ	B	4	-	-	
		W	4	-	-	
		L	4	-	-	
7	NOT	B	3	-	b	
		W	3	-	b	
		L	3	-	c	
8	TST	B	3	-	b	
		W	3	-	b	
		L	3	-	c	
9	TAS	B	7	-	a	

Table C-4. Shift & Rotate

No.	OP	Sz	Base Cycles	Type		Note
				EA (S)	EA (D)	
1	SHAL	B	< 19	-	b	
		W	< 19	-	b	
		L	< 19	-	c	
2	SHAR	B	< 16	-	b	
		W	< 16	-	b	
		L	< 16	-	c	
3	SHLL	B	< 12	-	b	
		W	< 12	-	b	
		L	< 12	-	c	
4	SHLR	B	< 13	-	b	
		W	< 13	-	b	
		L	< 13	-	c	
5	ROTL, ROTR	B	< 16	-	b	
		W	< 16	-	b	
		L	< 16	-	c	
6	ROTXL, ROTXR	B	< 17	-	b	
		W	< 17	-	b	
		L	< 17	-	c	

Table C-5. Branch

No.	OP	Sz	Base Cycles	Type EA (S)	Type EA (D)	Note
1	Bcc: G	B	4	-	-	c.c. = False
		W	5	-	-	Branch not taken
		L	7	-	-	
		B	12	-	-	c.c. = True
		W	15	-	-	Branch taken
		L	20	-	-	
2	BEQ, BNE	B	4	-	-	Branch not taken
		W	5	-	-	
		L	7	-	-	
		B	12	-	-	Branch taken
		W	15	-	-	
		L	20	-	-	
3	BRA	B	10	-	-	
		W	14	-	-	
		L	19	-	-	
4	BSR	B	10	-	-	
		W	14	-	-	
		L	18	-	-	
5	SCB	B	5	-	-	c.c. = True
		W	6	-	-	Branch not taken
		L	8	-	-	
		B	7	-	-	Rn = -1
		W	8	-	-	Branch not taken
		L	10	-	-	
		B	11	-	-	Rn $\neq -1$
		W	13	-	-	Branch taken
		L	17	-	-	
6	JMP		10	-	c	
7	JSR		10	-	c	

Table C-6. Return

No.	OP	Sz	Base Cycles	Type EA (S)	Type EA (D)	Note
1	RTD	B	15	-	-	
		W	19	-	-	
		L	22	-	-	
2	RTE	-	19	-	-	
3	RTR	-	17	-	-	
4	RTS	-	14	-	-	

Table C-7. Single Operand

No.	OP	Sz	Base Cycles	Type EA (S)	Type EA (D)	Note
1	CLR	B	3	-	a	
		W	3	-	a	
		L	3	-	a	
2	EXTS, EXTU	-	4	-	-	B -> W
		-	4	-	-	W -> L
		-	4	-	-	B -> L
3	SET	B	5	-	a	
4	SWAP	W	6	-	b	
		L	8	-	c	

Table C-8. Transfer

No.	OP	Sz	Base Cycles	Type EA (S)	Type EA (D)	Note
1	MOV: G, MOVS	B	4	b	a	
		W	4	b	a	
		L	4	c	a	
2	MOV: Q, MOVF	B	5	-	a	
		W	5	-	a	
		L	5	-	a	
3	MOV: R, MOV: RQ	B	4	-	-	
		W	4	-	-	
		L	4	-	-	
4	MOVA	-	4	a	a	
5	MOVFP	W	10	a	a	
		L	18	a	a	
6	MOVTP	W	10	b	a	
		L	17	c	a	
7	MOVFPE	B	14	a	a	Best case
		W	23	a	a	
		L	46	a	a	
		B	21	a	a	Worst case
		W	30	a	a	
		L	53	a	a	
8	MOVTPE	B	12	b	a	Best case
		W	25	b	a	
		L	43	c	a	
		B	19	b	a	Worst case
		W	32	b	a	
		L	50	c	a	
9	XCH	-	6	-	-	
10	LDC	B	13	-	b	
		W	13	-	b	
		L	13	-	c	
11	STC	-	7	-	a	

Table C-8. Transfer (cont.)

No.	OP	Sz	Base Cycles	Type: EA (S)	Type: EA (D)	Note
12	LDM	B	N = 0	N > 1		
			8	6N + 9	a	@–Rn
			8	6N + 9		@ Rn+
			8	6N + 8		Others
		W	N = 0	N > 1		
			8	6N + 9	a	@–Rn
			8	6N + 9		@Rn+
			8	6N + 8		Others
		L	N = 0	N > 1		
			8	8N + 9	a	@–Rn
			8	8N + 9		@ Rn+
			8	8N + 8		Others
13	STM	B	N = 0	N > 1		
			9	4N + 13	a	@–Rn
			8	4N + 12		@ Rn+
			9	4N + 9		Others
		W	N = 0	N > 1		
			9	4N + 13	a	@–Rn
			8	4N + 12		@ Rn+
			9	4N + 9		Others
		L	N = 0	N > 1		
			9	6N + 13	a	@–Rn
			8	6N + 12		@ Rn+
			9	6N + 9		Others

Table C-9. Register Bank

No.	OP	Sz	Base Cycles	Type EA (S)	EA (D)	Note	
1	DCBN	-	5N + 16	-	-	Load registers	
		-	5	-	-	Others	
2	ICBN	-	4N + 12	-	-	Store registers	
		-	3	-	-	Others	
3	CGBN	-	6N + 14	-	-	Copy registers	Dynamic
		-	8	-	-	Others	
		-	6N + 13	-	-	Copy registers	Static
		-	7	-	-	Others	
4	PGBN	-	6N + 12	-	-	Copy registers	
		-	9	-	-	Others	

Table C-10. Bit

No.	OP	Sz	Base Cycles	Type EA (S)	EA (D)	Note
1	BNOT, BCLR, BSET	B	8	-	b	
		W	7	-	b	
		L	8	-	b	
2	BTST	B	7	-	b	
		W	6	-	b	
		L	7	-	c	

Table C-11. String

No.	OP	Sz	Base Cycles	Type EA (S)	EA (D)	Note
1	SSTR	B	4N + 12	-	-	
		W	4N + 12	-	-	
2	SMOV	B	8N + 12	-	-	
		W	8N + 12	-	-	
3	SSCH	B	9N + 17	-	-	
		W	9N + 17	-	-	
4	SCMP	B	12N + 17	-	-	
		W	12N + 17	-	-	

Table C-12. Bit Field

No.	OP	Sz	Base Cycles	Type EA (S)	EA (D)	Note
1	BFEXT		14	c	a	
2	BFINS		23	b	c	
3	BFSCH		20 22	c c	a a	
4	BFMOV		12 + 14Y 12 + 32Y 12 + (13A + 22) Y 13 + (13B + 36) Y	- - - -	- - - -	X = 1, Rnb < 8 X = 1, Rnb ≥ 8 X = EVEN, A = X/2 X = ODD, B = (X – 1)/2, X≠ 1

Table C-13. Link

No.	OP	Sz	Base Cycles	Type EA (S)	EA (D)	Note
1	LINK	B W L	10 12 18	- - -	- - -	
2	UNLK	-	11	-	-	

Table C-14. Miscellaneous

No.	OP	Sz	Base Cycles	Type EA (S)	EA (D)	Note
1	TRAP	-	26 3	- -	- -	Trap taken Trap not taken
2	TRAPA	-	26	-	-	
3	NOP	-	1	-	-	
4	SLEEP	-	4	-	-	
5	RESET	-	265	-	-	

Table C-15. Execution Cycles Required for Exception Processing

Exception Processing		No. of Execution Cycles
Reset	Power-on	105 (Note)
	Manual	43
CPU bus error		51
CPU access level violation		49
DMA bus error		33
DMA access level violation		33
Interrupt	Non-acknowledge	34
	Single acknowledge	41
	Double acknowledge	48
Trace		33
Illegal instruction		34

Note: No. of execution cycles is calculated under the following conditions:
Power-on reset: 1 bus cycle = 11 system clocks
Other exceptions: 1 bus cycle = 3 system clocks

Appendix D. Pin State

Symbols: 1 = Output High

0 = Output Low

* = Output undefined value

Z = High impedance

IN = Input

OUT = Output effective value

Notes: 1. A23-A12: 0

A11-A1: Refresh address

2. A23-A17: Output the address in the CPU last bus cycle

A16/D15 - A1/Do: Z

3. Asserted only in the double acknowledge mode
4. Switched by the WTOE bit in the area wait control register (AWCR)
5. Pull up all input and I/O pins.
6. Do not pull down $\overline{\text{IACK}}$ pin.

Table D-1 shows the external pins' state in each operation mode of the HD641016.

Table D-1. Pin State

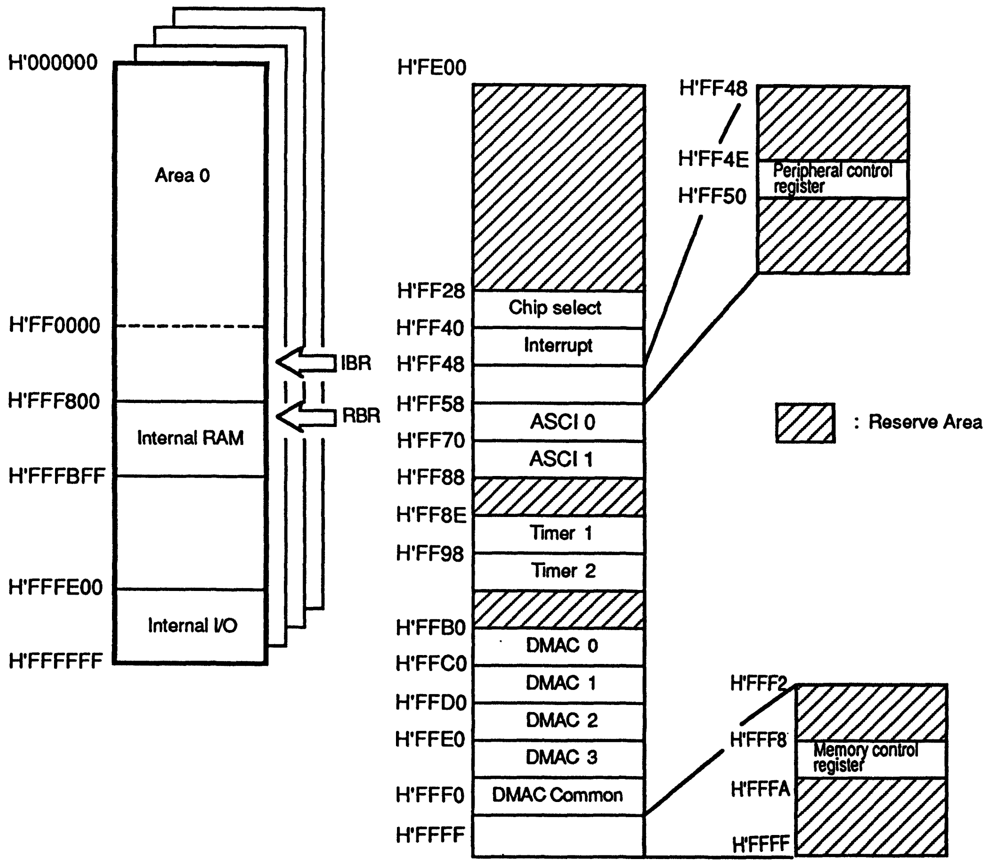
NO.	Operation Mode	Address Bus	Data Bus	$\overline{AS}$	$\overline{HDS}$ $\overline{LDS}$	$R/\overline{W}$	$S/\overline{U}$	PF	ST2	ST1	ST0	PCS1	PCS0	$\overline{WAIT}$	$\overline{BREQ}$	$\overline{BACK}$	$\overline{IACK}$	$\overline{BRTRY}$	Ø	E
1	Power-on reset	Z	Z	Z	Z	Z	Z	Z	1	1	1	1	1	IN	IN	1	1	IN	OUT	OUT
2	Manual reset	Z	Z	1	1	1	1			1	1	1	1	IN	IN	1	1	IN	OUT	OUT
3	Sleep mode	*2 Retained	Z	1	1	1	1	Retained	0	0	0	1	1	IN	IN	1	1	IN	OUT	OUT
4	System stop mode	*2 Retained	Z	1	1	1	1	Retained	0	0	0	1	1	IN	IN	1	1	IN	OUT	OUT
5	Halt	Z	Z	1	1	1	1	Retained	0	0	1	1	1	IN	IN	1	1	IN	OUT	OUT
6	Refresh	*1 OUT	Z	OUT	1	1	1	0	1	1	0	1	1	IN	IN	1	1	IN	OUT	OUT
7	Bus release	IN	Z	IN	Z	Z	Z	Z	Z	Z	Z	OUT	OUT	*4 OUT/Z	IN	0	Z	IN	OUT	OUT
8	Interrupt acknowledge	OUT	IN	OUT	OUT	1	1	0	1	1	1	1	1	IN	IN	1	*3 0	IN	OUT	OUT
9	Normal (External fetch)	OUT	IN	OUT	OUT	OUT	OUT	OUT	0	1	1	OUT	OUT	IN	IN	1	1	IN	OUT	OUT
10	Normal (Internal fetch)	OUT	Z	1	1	OUT	OUT	OUT	0	1	1	*	*	IN	IN	1	1	IN	OUT	OUT
11	Normal (External data write)	OUT	IN	OUT	OUT	OUT	OUT	OUT	0	1	1	OUT	OUT	IN	IN	1	1	IN	OUT	OUT
12	Normal (Internal I/O data)	OUT	OUT	OUT	OUT	OUT	OUT	OUT	0	1	1	OUT	OUT	IN	IN	1	1	IN	OUT	OUT
13	Normal (Internal I-O data read)	OUT	Z	1	1	OUT	OUT	OUT	0	1	1	*	*	IN	IN	1	1	IN	OUT	OUT
14	Normal (Internal I/O data write)	OUT	OUT	1	1	OUT	OUT	OUT	0	1	1	*	*	IN	IN	1	1	IN	OUT	OUT
15	Internal DMAC (Dual external read)	OUT	IN	OUT	OUT	OUT	OUT	OUT	1	0	1	OUT	OUT	IN	IN	1	1	IN	OUT	OUT
16	Internal DMAC (Dual external write)	OUT	OUT	OUT	OUT	OUT	OUT	OUT	1	0	1	OUT	OUT	IN	IN	1	1	IN	OUT	OUT
17	Internal DMAC (Dual internal I/O read)	OUT	Z	1	1	OUT	OUT	OUT	1	0	1	*	*	IN	IN	1	1	IN	OUT	OUT
18	Internal DMAC (Dual internal I/O write)	OUT	OUT	1	1	OUT	OUT	OUT	1	0	1	*	*	IN	IN	1	1	IN	OUT	OUT
19	Internal DMAC (Single memory read)	OUT	Z	OUT	OUT	OUT	OUT	OUT	1	0	0	OUT	OUT	IN	IN	1	1	IN	OUT	OUT
20	Internal DMAC (Single memory write)	OUT	Z	OUT	OUT	OUT	OUT	OUT	1	0	0	OUT	OUT	IN	IN	1	1	IN	OUT	OUT

Table D-2 shows the state priority when multiple states, which can be decoded by ST2-ST0, occur simultaneously.

Table D-2. State Priority

No.	States Name	Priority	State Occurring Simultaneously (No.)
1	DMA Single address mode	2	5
2	DMA Dual address mode	2	5
3	Bus lock	3	
4	Normal	4	
5	Sleep	3	1, 2, 7
6	System stop	3	7
7	Refresh	1	5, 6
8	Interrupt	3	

Appendix E. Memory Map



Appendix F. CPU Register Initial Values

CPU Register are initialized as shown in Figure F-1. Note that general purpose registers and BSP are not initialized.

SR

15								7							0
0	–	1	0	–	1	1	1	–	–	–	·	·	·	·	·

CCR

15								7							0
–	–	–	–	–	–	–	–	–	–	–	·	·	·	·	·

EBR

31								23								15								7							0
·	·	·	·	·	·	·	·	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RBR

31								23								15								7							0
·	·	·	·	·	·	·	·	1	1	1	1	1	1	1	1	1	1	1	1	1	0	·	·	·	·	·	·	·	·	·	·

IBR

31								23								15								7							0
·	·	·	·	·	·	·	·	1	1	1	1	1	1	1	1	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·

CBNR

31								23								15								7							0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- 0 .... Cleared (0)
- 1 .... Set (1)
- · .... Undefined
- — .... Reserve bit. Always read as 0. Cannot be written to.

General perpose register and BSP are not initialized.

BMR

7							0
1	0	1	0	0	1	0	0

GBNR

7							0
·	·	·	·	0	0	0	0

VBNR

7							0
·	·	·	·	·	0	0	0

Figure F-1. Register Initialization

Appendix G. I/O Register

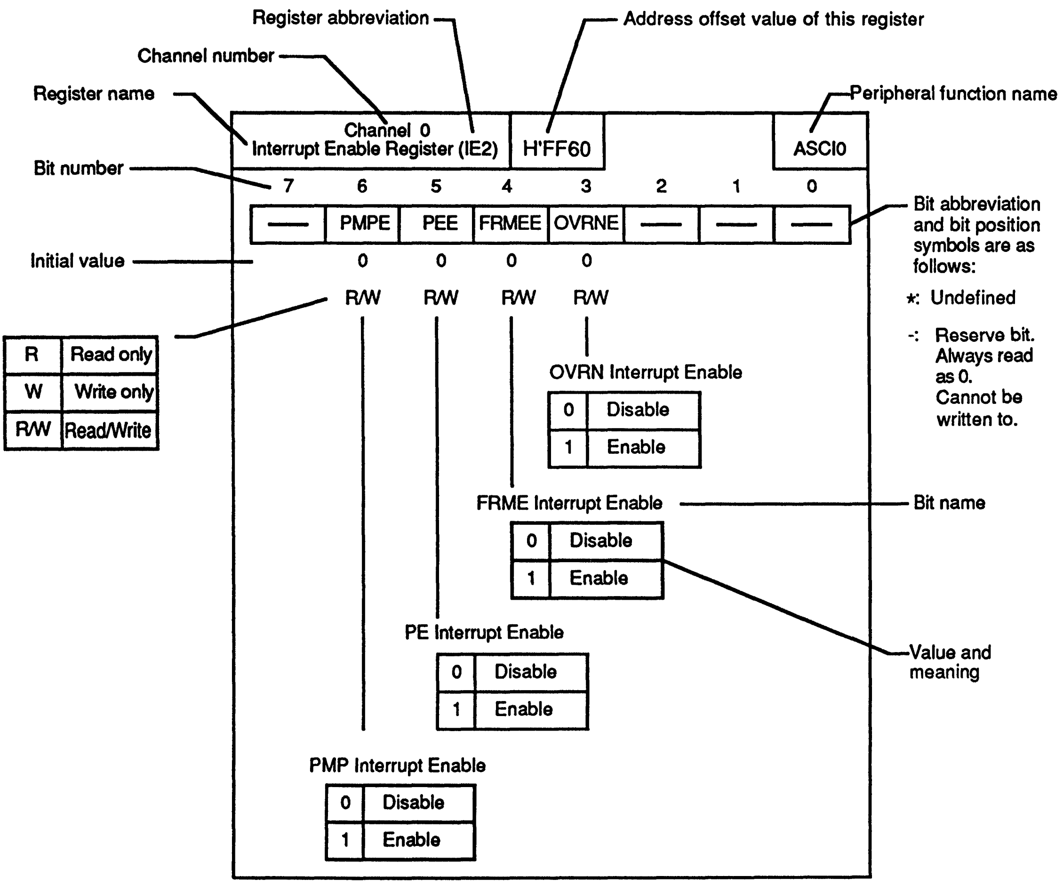


Figure G-1. Register Description Key

Area Base Register 0 (ABR0) — H'FF28 — CS

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	—	—	AB23	AB22	AB21	AB20	AB19	AB18	AB17	AB16
Initial value									0	0	0	0	0	0	0	0
Read/Write									R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Area Base 23—16

Specifies bits 23—16 of the area start address

Area Range Register 0 (ARR0) — H'FF2A — CS

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	—	—	AR23	AR22	AR21	AR20	AR19	AR18	AR17	AR16
Initial value									0	0	0	0	0	0	0	0
Read/Write									R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Area Range 23 – 16

Specifies bits 23—16 of the area length

Area Wait Control Register 0 (AWCR0) — H'FF2C — CS

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	—	—	WTOE	ALV	—	—	—	BWC2	BWC1	BWC0
Initial value									0	1				1	1	1
Read/Write									R/W	R/W				R/W	R/W	R/W

Wait Output Enable

0	Disables $\overline{\text{WAIT}}$ output
1	Enables $\overline{\text{WAIT}}$ output

Access Level

0	User level
1	Supervisor level

Bus Wait Cycle 2-0

BWC2	BWC1	BNC0	No. of wait states
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Area Base Register 1 (ABR1) | H'FF2E | CS

Same as area base register 0.

Area Range Register 1 (ARR1) | H'FF30 | CS

Same as area base register 0.

Area Wait Control Register 1 (AWCR1) | H'FF32 | CS

Same as area wait control register 0.

Area Base Register 2 (ABR 2) | H'FF34 | CS

Same as area base register 0.

Area Range Register 2 (ARR 2) | H'FF36 | CS

Same as area range register 0.

Area Wait Control Register 2 (AWCR2) | H'FF38 | CS

Same as area wait control register 0.

Area Base Register 3 (ABR 3) | H'FF3A | CS

Same as area base register 0.

Area Range Register 3 (ARR3) | H'FF3C | CS

Same as area range register 0.

Area Wait Control Register 3 (AWCR3) | H'FF3E | CS

Same as area wait control register 0.

Interrupt Priority Register 0 (IPR0) H'FF40 IC

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—	—	S0P1	S0P0	—	—	S1P1	S1P0	—	—	—	—	—	—	T1P1	T1P0
Initial value			0	0			0	0							0	0
Read/Write			R/W	R/W			R/W	R/W							R/W	R/W

ASCA0 Priority 1, 0 ASCI1 Priority 1, 0 Timer 1 Priority 1, 0

S0P1 / S1P1 / T1P1	S0P0 / S1P0 / T1P0	Priority Level	
0	0	0	(Mask)
0	1	1	(Low)
1	0	2	(Medium)
1	1	3	(High)

Interrupt Priority Register 1 (IPR1) H' FF42 IC

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—	—	T2P1	T2P0	—	—	—	—	—	—	DM0P1	DM0P0	—	—	DM1P1	DM1P0
Initial value			0	0							0	0			0	0
Read/Write			R/W	R/W							R/W	R/W			R/W	R/W

Timer 2 Priority 1, 0 DMAC0 Priority 1, 0 DMAC1 Priority 1, 0

T2P1 / DM0P1 / DM1P1	T2P0 / DM0P0 / DM1P0	Priority Level	
0	0	0	(Mask)
0	1	1	(Low)
1	0	2	(Medium)
1	1	3	(High)

Interrupt Priority Register 2 (IPR 2) H' FF44 IC

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—	—	DM2P1	DM2P0	—	—	DM3P1	DM3P0	—	—	IR0P1	IR0P0	—	—	IR1P1	IR1P1
Initial value			0	0			0	0			0	0			0	0
Read/Write			R/W	R/W			R/W	R/W			R/W	R/W			R/W	R/W

DMAC2 Priority 1, 0 DMAC3 Priority 1, 0 IRQ0 Priority 1, 0 IRQ1 Priority 1, 0

DM2P1 / DM3P1 / IR0P1 / IR1P1	DMP2P0 / DM3P0 / IR0P0 / IR0P0	Priority Level	
0	0	0	(Mask)
0	1	1	(Low)
1	0	2	(Medium)
1	1	3	(High)

Interrupt Control Register 1 (ICR)	H'FF46	IC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	IMOD	NMIV	IRQV0	IRQV1	IRQIM0	IRQIM1	—	—
								0	0	0	0	0	0		
								R/W	R/W	R/W	R/W	R/W	R/W		

IRQ Input Mode 0, 1

IRQ Vector Select 0, 1

NMI Vector Select

0	Non-acknowledge mode
1	Single acknowledge mode

Interrupt Mode

$\overline{\text{IRQ0}}$ Pin

IRQV1	IRQIM1	Acknowledge Mode	Trigger Mode
0	0	Non-acknowledge mode	Level
	1	Non-acknowledge mode	Edge
1	0	Single acknowledge mode	Level
	1	Single acknowledge mode	Edge

$\overline{\text{IRQ0}}$ Pin

IMOD	IRQIM0	IRQIM1	Acknowledge Mode	Trigger Mode	IACK Pin
0	0	0	Non-acknowledge mode	Level	Unused
		1	Non-acknowledge mode	Edge	
	1	0	Single acknowledge mode	Level	
		1	Single acknowledge mode	Edge	
1	0	0	Double acknowledge mode	Level	Used
		1			
	1	0			
		1			

Peripheral Control Register 0 (PCR0)	H'FF4E	PC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	PTF1	—	PTF0	—	—	—	—	—	—	—	—	—	—
			0		0										
			R/W		R/W										

Peripheral Terminal Function 0

0	$\overline{\text{RTS}}$ 1 pin
1	$\overline{\text{DACK}}$ 3 pin

Peripheral Terminal Function 1

0	$\overline{\text{DCD1}}$ pin
1	$\overline{\text{DREQ}}$ 3 pin

TX/RX Buffer Register (TRB) Channel 0	H'FF58	ASCI0

7	6	5	4	3	2	1	0
TRB 7	TRB 6	TRB 5	TRB 4	TRB 3	TRB 2	TRB 1	TRB 0
*	*	*	*	*	*	*	*
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TX/RX Buffer 7—0

Write	Transmit character is written to the TX buffer.
Read	Receive character is read from the RX buffer.

Status Register 0 (ST0) Channel 0 — H'FF59 — ASCI0

Bit	7	6	5	4	3	2	1	0
Name	TXINT	RXINT	—	—	—	—	TXRDY	RXRDY
Initial value	0	0					0	0
Read/Write	R	R					R	R

TXINT Interrupt

0	Disabled
1	Enabled

RXINT Interrupt

0	Disabled
1	Enabled

TXRDY Interrupt

0	TX buffer full
1	TX buffer empty

RXRDY Interrupt

0	No receive data
1	Receive data

Status Register 1 (ST1) Channel 0 — H' FF5A — ASCI0

Bit	7	6	5	4	3	2	1	0
Name	—	IDL	—	—	CCTS	CDCD	BRKD	BRKE
Initial value		0			0	0	0	0
Read/Write		R/W			R/W	R/W	R/W	R/W

TX Idle

0	Non-idle state
1	Idle state

Change of CTS

0	No CTS change
1	CTS change

Change of DCD

0	No DCD change
1	DCD Change

Break Start Detect

0	No break start
1	Break start detected

Break End Detect

0	Nobreak end
1	Break end detected

Status Register 2 (ST2) Channel 0 — H'FF5B — ASCI0

Bit	7	6	5	4	3	2	1	0
Name	—	PMP	PE	FRME	OVRN	—	—	—
Initial value		0	0	0	0			
Read/Write		R/W	R/W	R/W	R/W			

Parity/Multiprocessor

0	"0"
1	"1"

Parity Error

0	No parity error
1	Parity error

Framing Error

0	No framing error
1	Framing error

Overrun Error

0	No overrun error
1	Overrun error

Status Register 3 (ST3) Channel 0 — H'FF5C — ASCI0

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	$\overline{CTS}$	$\overline{DCD}$	TXENBL	RXENBL
Initial value					0	0	0	0
Read/Write					R	R	R	R

RX Enable

0	Disable
1	Enable

TX Enable

0	Disable
1	Enable

Data Carrier Detect

0	Low
1	High

Clear To Send

0	Low
1	High

Interrupt Enable Register 0 (IE0) Channel 0 — H'FF5E — ASCI0

Bit	7	6	5	4	3	2	1	0
	TXINTE	RXINTE	—	—	—	—	TXRDYE	RXRDYE
Initial value	0	0					0	0
Read/Write	R/W	R/W					R/W	R/W

RXINT Interrupt Enable

0	Disable
1	Enable

RXRDY Interrupt Enable

0	Disable
1	Enable

TXINT Interrupt Enable

0	Disable
1	Enable

TXRDY Interrupt Enable

0	Disable
1	Enable

Interrupt Enable Register 1 (IE1) Channel 0 — H'FF5F — ASCI0

Bit	7	6	5	4	3	2	1	0
	—	IDLE	—	—	CCTSE	CDCDE	BRKDE	BRKEE
Initial value		0			0	0	0	0
Read/Write		R/W			R/W	R/W	R/W	R/W

BRKE Interrupt Enable

0	Disable
1	Enable

BRKD Interrupt Enable

0	Disable
1	Enable

CCTS Interrupt Enable

0	Disable
1	Enable

CDCD Interrupt Enable

0	Disable
1	Enable

IDL Interrupt Enable

0	Disable
1	Enable

Interrupt Enable Register 2 (IE2) Channel 0 — H'FF60 — ASCI0

Bit	7	6	5	4	3	2	1	0
	—	PMPE	PEE	FRMEE	OVRNE	—	—	—
Initial value		0	0	0	0			
Read/Write		R/W	R/W	R/W	R/W			

OVRN Interrupt Enable

0	Disable
1	Enable

FRME Interrupt Enable

0	Disable
1	Enable

PE Interrupt Enable

0	Disable
1	Enable

PMP Interrupt Enable

0	Disable
1	Enable

Command Register (CMD) Channel 0 — H'FF62 — ASCI0

Bit	7	6	5	4	3	2	1	0
	—	—	CMD 5	CMD 4	CMD 3	CMD 2	CMD 1	CMD 0
Initial value			0	0	0	0	0	0
Read/Write			W	W	W	W	W	W

Command 5—0

Transfer Commands

000001	TX reset
000010	TX enable
000011	TX disable
001000	MP bit on
001001	TX buffer clear

Receive Commands

010001	RX reset
010010	RX enable
010011	RX disable
010110	Search MP bit

Miscellaneous Commands

100001	Channel reset
000000	No operation

Mode Register 0 (MD0) Channel 0 — H'FF63 — ASCI0

Bit	7	6	5	4	3	2	1	0
	PRTCL2	PRTCL1	PRTCL0	AUTO	—	—	STOP 1	STOP 0
Initial value	0	0	0	0			0	0
Read/Write	R/W	R/W	R/W	R/W			R/W	R/W

Protocol Mode 2-0

PRTCL2 - PRTCL0	Protocol mode
000	Asynchronus mode
001 - 101	Reserved
110	Clock synchronus mode
111	Reserved

Auto Enable

0	Non auto enable
1	Autoenable

Stop Bit Length 1, 0

STOP1 STOP0	Stop bit length
00	1 bit
01	Reserved
10	2 bits
11	Reserved

Mode Register 1 (MD1) Channel 0 | H'FF64 | ASCI0

Bit	7	6	5	4	3	2	1	0
	BRATE 1	BRATE 0	TXCHR 1	TXCHR0	RXCHR 1	RXCHR 0	PMPM 1	PMPM 0
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit Rate 1, 0

BRATE1 BRATE0	Bit Rate
00	1/1 clock rate
01	1/16 clock rate
10	1/32 clock rate
11	1/64 clock rate

TX Character Length 1, 0

TXCHR1 TXCHR0	TX character length
00	8 bits
01	7 bits
10	Reserved
11	Reserved

RX Character Length 1, 0

RXCHR1 RXCHR0	RX character length
00	8 bits
01	7 bits
10	Reserved
11	Reserved

Parity/Multiprocessor Mode 1,0

PMPM1 PMPM0	Parity/ Multiprocessor mode
00	None
01	MP bit
10	Even parity
11	Odd parity

Mode Register 2 (MD2) Channel 0 | H'FF65 | ASCI0

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	CNCT 1	CNCT 0
Initial value							0	0
R/W							R/W	R/W

Channel Connection 1, 0

CNCT1 CNCT0	Transmit/Receive Mode
00	Full-duplex
01	Auto-echo
10	Reserved
11	Local loopback

Control Register (CTL) Channel 0 | H'FF66 | ASCI0

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	BRK	—	—	$\overline{RTS}$
Initial value					0			1
R/W					R/W			R/W

Send Break

0	off
1	on

Request to Send

0	Low
1	High

Time Constant Register (TMC) Channel 0 | **H'FF6A** | **ASCI0**

Bit	7	6	5	4	3	2	1	0
	TMC 7	TMC 6	TMC 5	TMC 4	TMC 3	TMC 2	TMC 1	TMC 0
Initial value	0	0	0	0	0	0	0	1
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Time Constant 7 – 0
Value to be loaded Reload Timer ($f_{\varnothing}$/TMC)

RX Clock Source Register (RXS) Channel 0 | **H'FF6B** | **ASCI0**

Bit	7	6	5	4	3	2	1	0
	—	RXCS 2	RXCS 1	RXCS 0	—	—	—	—
Initial value		0	0	0				
		R/W	R/W	R/W				

RX Clock Source Select 2- 0

Asynchronous	000	RXC input
	100	BRG output
Clock synchronous	000	Slave mode
	100	Master mode

TX Clock Source Register (TXS) Channel 0 | **H'FF6C** | **ASCI0**

Bit	7	6	5	4	3	2	1	0
	—	TXCS 2	TXCS 1	TXCS 0	TXBR 3	TXBR 2	TXBR 1	TXBR 0
Initial value		0	0	0	0	0	0	0
		R/W	R/W	R/W	R/W	R/W	R/W	R/W

TX Clock Source Select 2 – 0

Asynchronous	000	TXC input
	100	BRG output
Clock synchronous	000	Slave mode
	100	Master mode

TX Baud Rate Select 3-0

TXBR3 - TXBR0	Baud Rate
0000	1/1
0001	1/2
0010	1/4
0011	1/8
0100	1/16
0101	1/32
0110	1/64
0111	1/128
1000	1/256
1001	1/512
1010 – 1111	Reserved

TX/RX Buffer Register (TRB) Channel 1	H' FF70	ASCI 1

Same as TX/RX buffer register channel 0.

Status Register 0 (ST0) Channel 1	H'FF71	ASCI1

Same as status register 0 channel 0.

Status Register 1 (ST1) Channel 1	HFF72	ASCI1

Same as status register 1 channel 0.

Status Register 2 (ST2) Channel 1	H'FF73	ASCI1

Same as status register 2 channel 0.

Status Register 3 (ST3) Channel 1	H'FF74	ASCI1

Same as status register 3 channel 0.

Interrupt Enable Register 0 (IE0) Channel 1	H'FF76	ASCI1

Same as interrupt enable register 0 channel 0.

Interrupt Enable Register 1 (IE1) Channel 1	H'FF77	ASCI1

Same as interrupt enable register 1 channel 0.

Interrupt Enable Register 2 (IE2) Channel 1	H'FF78	ASCI1

Same as interrupt enable register 2 channel 0.

Command Register (CMD) Channel 1	H'FF7A	ASCI1

Same as command register chaanel 0.

Mode Register 0 (MD0) Channel 1	H'FF7B	ASCI1

Same as mode register 0 channel 0.

Mode Register 1 (MD1) Channel 1	H'FF7C	ASCI1

Same as mode register 1 channel 0.

Mode Register 2 (MD2) Channel 1	H'FF7D	ASCI1

Same as mode register 2 channel 0.

Control Register (CTL) Channel 1	H'FF7E	ASCI1

Same as control register channel 0.

Time Constant Register (TMC) Channel 1	H'FF82	ASCI1

Same as time constant register channel 0.

RX Clock Source Register (RXS) Channel 1	H'FF83	ASCI1

Same as RX clock source register channel 0.

TX Clock Source Register (TMS) Channel 1	H'FF84	ASCI1

Same as TX clock source register channel 0.

Up Count Register (UCR) Channel 1 | H'FF8E | Timer 1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Count value

Count Compare Register A (CCRA) Channel 1 | H'FF90 | Timer 1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Compare value A

Count Compare Register B (CCRB) Channel 1 | H' FF92 | Timer 1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Compare value B

Control Register (CNTR) Channel 1	H'FF94	Timer 1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MDS1	MDS0	CSS2	CSS1	CSS0	TSS1	TSS0	DMA	CIE	OIE	MIE	OLB	OLA	UCE	UCRC	STP
Initial value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Stop Point

0	TIOA precedes
1	TIOB precedes

Upcount Register Clear

Upcount Enable

UCE	UCRC	Timer operation
0	0	Count stop
1	0	Count run
*	1	Count initialization

* don't care

Output Level B.A

OLB	OLA	Timer output
0	0	Low
1	1	High

Measurement End Interrupt Enable

0	Disable
1	Enable

Overflow Interrupt Enable

0	Disable
1	Enable

Compare Interrupt Enable

0	Disable
1	Enable

Direct Memory Access Request

0	Disable
1	Enable

Trigger Source Select 1-0

TSS1	TSS0	Trigger input
0	0	Reserved
0	1	Reserved
1	0	TTGI
1	1	TOUT

Clock Sourse Select 2-0

CSS2	CSS1	CSS0	Input clock
0	0	0	1/8Ø
0	0	1	Reserved
0	1	0	Reserved
0	1	1	Reserved
1	0	0	1/64Ø
1	0	1	Reserved
1	1	0	TCKI
1	1	1	TOUT

Mode Select I.O

MDS1	MDS0	Mode
0	0	Measurement
0	1	One shot
1	0	Normal
1	1	Two-phase output

Status Register (STR) Channel 1	H'FF96	Timer 2

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	—	—	—	—	—	—	CF	OF	MF	OLS
Initial value													0	0	0	0
Read/Write													R	R	R	R

Output Level Status (Note)

0	Timer output Low
1	Timer output High

Measurement End Flag

0	Measurement
1	Measurement end

Overflow Flag

0	No overflow
1	Overflow

Compare Flag

0	No count match
1	Count match

Note: In two-phase output mode, OLS indicates the logical OR of the two output levels.

Up Count Register (UCR) Channel 2	H'FF98	Timer 2

Same as up count register channel 1.

Count Compare Register A (CCRA) Channel 2	H'FF9A	Timer 2

Same as count compare register A channel 1.

Count Compare Rgister B (CCRB) Channel 2	H'FF9C	Timer 2

Same as count compare register B channel 1.

Control Register (CNTR) Channel 2	H'FF9E	Timer 2

Same as control register channel 1.

Status Register (STR) Channel 2	H'FFA0	Timer 2

Same as status register channel 1.

Memory Address Register (MADR) Channel 0	H'FFB0	DMAC 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
—	—	—	—	—	—	—	—								
								*	*	*	*	*	*	*	*
								R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Device Address/Next Block Address Register (DADR/ NADR) Channel 0	H'FFB4	DMAC 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
—	—	—	—	—	—	—	—								
								*	*	*	*	*	*	*	*
								R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Execute Transfer Count Register (ETCR) Channel 0	H'FFB8	DMAC 0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Specifies transfer count.

Next Block Transfer Count Register (NTCR) Channel 0	H'FFBA	DMAC 0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Contains transfer count.

Channel Control Register A (CHCRA) Channel 0	H'FFBC	DMAC0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MRC	MS 1	MS 0	DC 2	DC 1	DC 0	DPS	OPS	RQS	BM	DIR	P	BIE	BTF	TIE	TF
Initial value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Transfer Finish

0	DMA not done
1	DMA done

TF Interrupt Enable

0	TF int disabled
1	TF int enabled

Block Transfer Finish

0	Block transfer not done
1	Block transfer done

BTF Interrupt Enable

0	BTF int disabled
1	BTF int enabled

Priority

0	Normal
1	Express

Direction

0	Memory to device
1	Device to memory

Bus Mode

0	Obedient or burst mode
1	Cycle steal mode

Request Signal Sense

0	Level sensitive
1	Edge sensitive

Operand Size

0	Word
1	Byte

Device Port Size

0	16 bits
1	8 bits

Device Classification 2-0
See Table 8-3

Memory Space 1,0

00	User data
01	User program
10	Supervisor data
11	Supervisor program

Memory Address Register Change

0	Increment
1	Decrement

Channel Control Register B (CHCRB) Channel 1	H'FFBE	DMAC0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DRC1	DRC0	DS1	DS0	DBER	—	—	—	—	—	—	—	—	—	—	—
	0	0	0	0	0											
	R/W	R/W	R/W	R/W	R/W											

DMA BUS Error

0	No bus error
1	Bus error

Device Space 1,0

00	User data
01	User program
10	Supervisor data
11	Supervisor program

Device Address Register Change 1,0

00	Increment
01	Decrement
1X	Reserve
10	Fixed
11	Reserved

Memory Address Register (MADR) Channel 1	H'FFC0	DMAC1

Same as memory address register channel 0.

Device Address/Next Block Address Register (DADR/NADR) Channel 1	H'FFC4	DMAC1

Same as device address/next block address register channel 0.

Execute Transfer Count Register (ETCR) Channel 1	H'FFC8	DMAC1

Same as execute transfer count register channel 0.

Next Block Transfer Count Register (NTCR) Channel 1	H'FFCA	DMAC1

Same as next block transfer count register channel 0.

Channel Control Register A (CHCRA) Channel 1	H'FFCC	DMAC1

Same as channel control register A channel 0.

Channel Control Register B (CHCRB) Channel 1	H'FFCE	DMAC1

Same as channel control register B channel 0.

Memory Address Register (MADR) Channel 2	H'FFD0	DMAC 2

Same as memory address register channel 0.

Device Address/Next Block Address Register (DADR /NADR) Channel 2	H'FFD4	DMAC 2

Same as device address/next block address register channel 0.

Execute Transfer Count Register (ETCR) Channel 2	H'FFD8	DMAC 2

Same as execute transfer count register channel 0.

Next Block Transfer Count Register (NTCR) Channel 2	H'FFDA	DMAC 2

Same as next block transfer count register channel 0.

Channel Control Register A (CHCRA) Channel 2	H'FFDC	DMAC 2

Same as channel control register A channel 0.

Channel Control Register B (CHCRB) Channel 2	H'FFDE	DMAC 2

Same as channel control register B channel 0.

Memory Address Register (MADR) Channel 3	H'FFE0	DMAC 3

Same as memory address register channel 0.

Device Address/Next Block Address Register (DADR/NADR) Channel 3	H'FFE4	DMAC 3

Same as device address /next block address register channel 0.

Execute Transfer Count Register (ETCR) Channel 3	H'FFE8	DMAC 3

Same as execute transfer count register channel 0.

Next Block Transfer Count Register (NTCR) Channel 3	H'FFEA	DMAC3

Same as next block transfer count register channel 0.

Channel Control Register A (CHCRA) Channel 3	H'FFEC	DMAC 3

Same as channel control register A channel 0.

Channel Control Register B (CHCRB) Channel 3	H'FFEE	DMAC 3

Same as channel control register B channel 0.

Operation Control Register (OPCR)	H'FFF0	DMAC 3

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	WE0	DE0	SUC0	—	WE1	DE1	SUC1	—	WE2	DE2	SUC2	—	WE3	DE3	SUC3	DME
Initial Value	0	0	0		0	0	0		0	0	0		0	0	0	0
Read/Write	W	R/W	R/W		W	R/W	R/W		W	R/W	R/W		W	R/W	R/W	R

DMA Master Enable

0	All DMA disabled
1	All DMA enabled

Successive Operation Channel 3

0	Succ transfer disabled
1	Succ transfer enabled

DMA Enable Channel 3

0	DMA disabled
1	DMA enabled

Write Enable Channel 3

0	DE3, SUC3 write disabled
1	DE3, SUC3 write enabled

Successive Operation Channel 2

0	Succ transfer disabled
1	Succ transfer enabled

DMA Enable Channel 2

0	DMA disabled
1	DMA enabled

Successive Operation Channel 1

0	Succ transfer disabled
1	Succ transfer enabled

Write Enable Channel 2

0	DE2, SUC2 write disabled
1	DE2, SUC2 write enabled

DMA Enable Channel 1

0	DMA disabled
1	DMA enabled

Write Enable Channel 1

0	DE1, SUC1 write disabled
1	DE1, SUC1 write enabled

Successive Operation Channel 0

0	Succ transfer disabled
1	Succ transfer enabled

DMA Enable Channel 0

0	DMA disabled
1	DMA enabled

Write Enable Channel 0

0	DE0, SUC0 write disabled
1	DE0, SUC0 write enabled

—: Reserved. Always read as 0. Cannot be written.
OPCR is not affected by the RESET instruction.

Memory Control Register (MCR)	H'FFF8	MC

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	WITE	ASWC	IWC1	IWC0	—	—	—	RPL	REFE	RWC1	RWC0	RRN1	RRN0	CYC2	CYC1	CYC0
Initial value	1	1	1	1				0	1	1	1	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W				R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Interrupt Wait Cycle 1, 0

IWC1 IWC0	No. of Tw states
00	0
01	1
10	2
11	3

AS Wait Control

0	Disable Tp state insertion
1	Enable Tp state insertion

$\overline{\text{WAIT}}$ Input Enable

0	Disable $\overline{\text{WAIT}}$ input
1	Enable $\overline{\text{WAIT}}$ input

Refresh Request Number

RRN1 RRN0	Max No. of Deferred Refresh Cycles
00	31
01	63
10	127
11	255

Refresh Wait Cycle 1, 0

RWC1 RWC0	No. of T_{RW} States
00	0
01	1
10	2
11	3

Refresh Cycle 2-0

CYC-2 CYC0	Refresh Cycle Interval
000	32
001	64
010	96
011	128
100	160
101	192
110	224
111	256

Refresh Enable

0	Disable refresh cycle insertion
1	Enable refresh cycle insertion

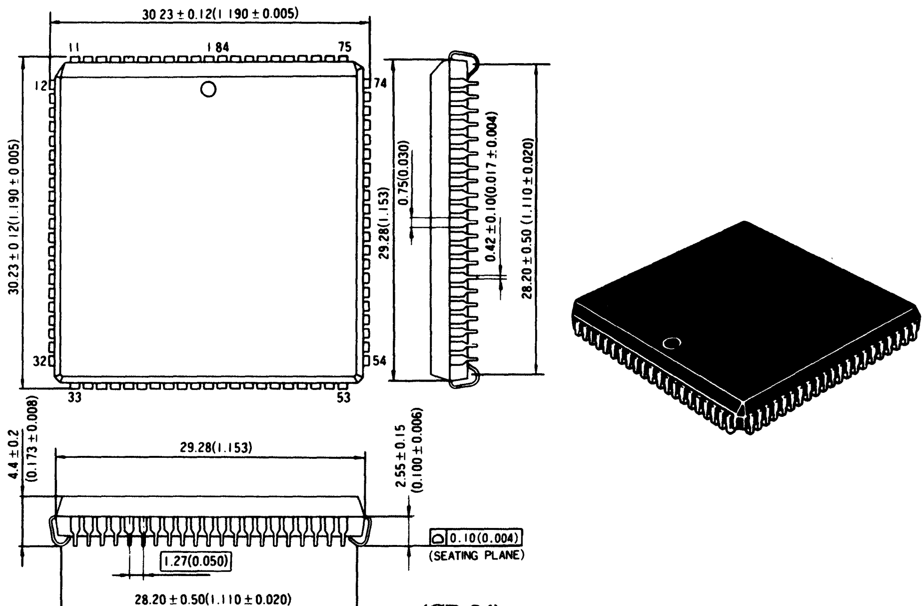
Refresh Priority Level

0	Normal refresh
1	Burst refresh

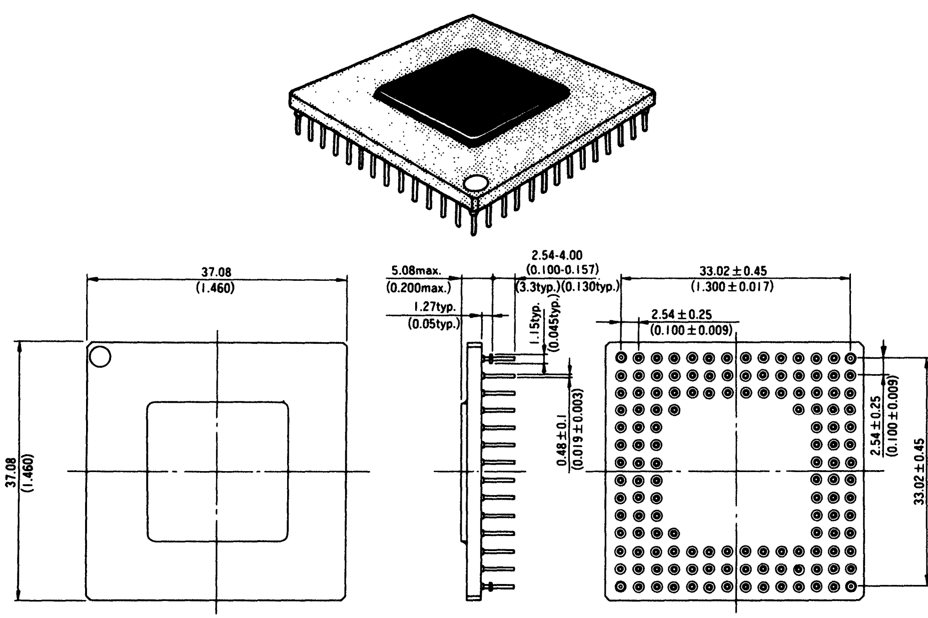
–: Reserved bit. Always read as 0. Cannot be written to.
MCR is not affected by the RESET instruction.

Appendix H. Package Dimensions

Unit: mm (inch)

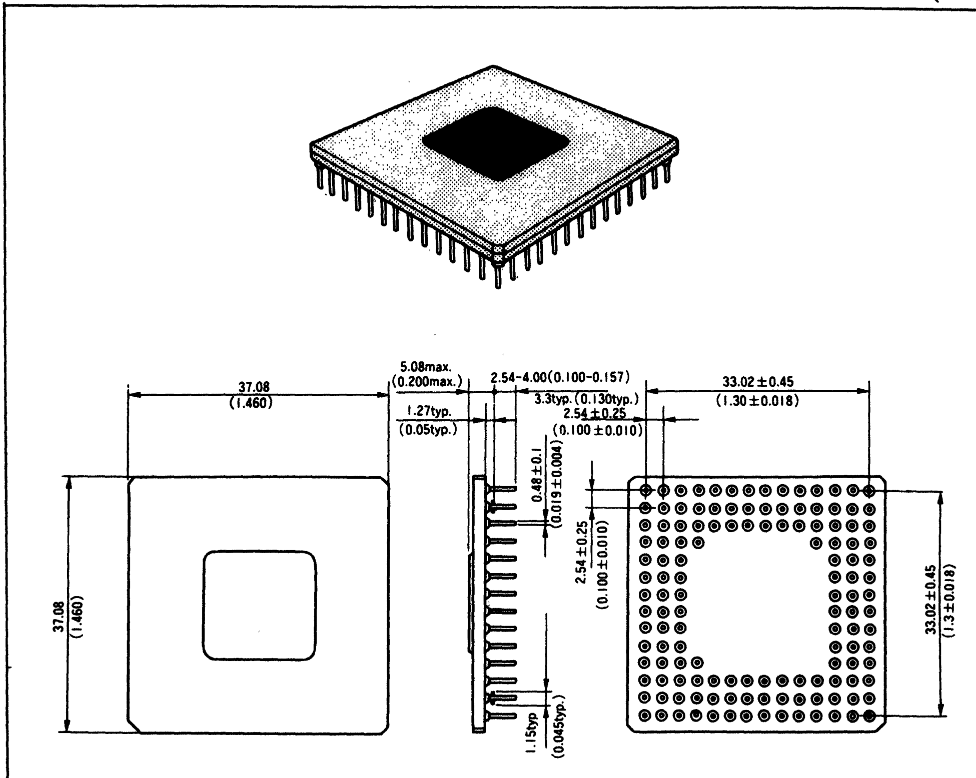


(CP-84)



(PC-135)

Unit: mm (inch)



(PP-135)

Appendix I. PLCC Installation Note

Description: When a method such as infrared reflow method, which requires heating of the entire HD641016 device, is used for installing the HD641016 package on a printed circuit board (hereinafter called PCB), the HD641016 package must be baked to prevent package cracking or characteristic failures before mounting on the PCB, because the chip size of the HD641016 is comparatively large. In addition, baking is also required before the HD641016 is mounted on the PCB after being taken from a dry-pack package.

Recommended reflow conditions are shown in Figure I-1.

Baking Condition: 125 C   10 C, 16 to 24 hours, non-continuity state.

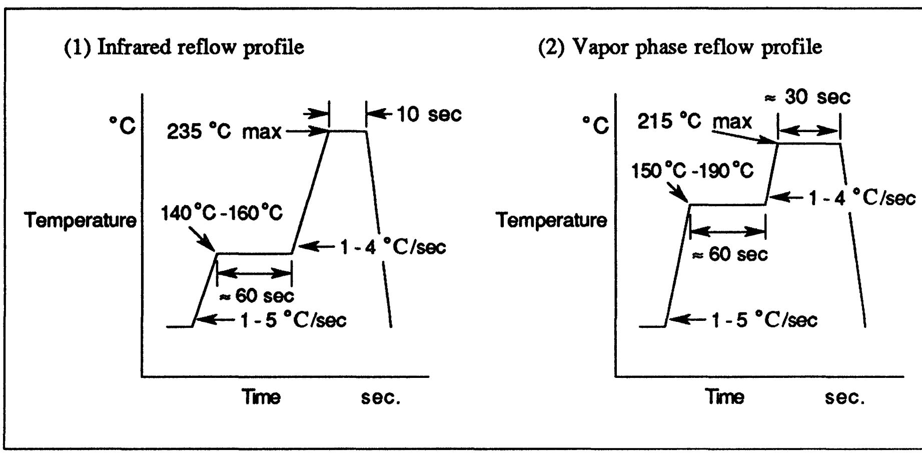


Figure I-1. Recommended Reflow Conditions

NOTES

NOTES

Hitachi America, Ltd.

SEMICONDUCTOR and IC DIVISION

Hitachi America, Ltd.
Semiconductor & IC Division
Hitachi Plaza
2000 Sierra Point Parkway
Brisbane, CA 94005-1819
Telephone: 415-589-8300
Telex: 17-1581
Twx: 910-338-2103
FAX: 415-583-4207

REGIONAL OFFICES

MID-ATLANTIC REGION

Hitachi America, Ltd.
1700 Galloping Hill Rd.
Kenilworth, NJ 07033
201/245-6400

NORTHEAST REGION

Hitachi America, Ltd.
5 Burlington Woods Drive
Burlington, MA 01803
617/229-2150

NORTH CENTRAL REGION

Hitachi America, Ltd.
500 Park Blvd., Suite 415
Itasca, IL 60143
312/773-4864

NORTHWEST REGION

Hitachi America, Ltd.
2000 Sierra Point Parkway
Brisbane, CA 94005-1819
415/589-8300

SOUTH CENTRAL REGION

Hitachi America, Ltd.
Two Lincoln Centre, Suite 865
5420 LBJ Freeway
Dallas, TX 75240
214/991-4510

SOUTHWEST REGION

Hitachi America, Ltd.
18300 Von Karman Avenue,
Suite 730
Irvine, CA 92715
714/553-8500

SOUTHEAST REGION

Hitachi America, Ltd.
4901 N.W. 17th Way, Suite 302
Fort Lauderdale, FL 33309
305/491-6154

AUTOMOTIVE

Hitachi America, Ltd.
6 Parklane Blvd., #558
Dearborn, MI 48126
313/271-4410

DISTRICT OFFICES

Hitachi America, Ltd.
3800 W. 80th Street
Suite 1050
Bloomington, MN 55431
612/896-3444

Hitachi America, Ltd.
21 Old Main Street, Suite 104
Fishkill, NY 12524
914/897-3000

Hitachi America, Ltd.
6161 Savoy Dr., Suite 850
Houston, TX 77036
713/974-0534

Hitachi (Canadian) Ltd.
2625 Queensview Dr.
Ottawa, Ontario, Canada K2A 3Y4
613/596-2777

Hitachi America, Ltd.
401 Harrison Oaks Blvd.
Suite #317
Cary, NC 27513
919/481-3908

HITACHI®

HITACHI®

Our Standards Set Standards

Hitachi America, Ltd.
Semiconductor and IC Division
Hitachi Plaza
2000 Sierra Point Parkway, Brisbane, CA 94005-1819
1-415-589-8300

7.5M September, 1989

Printed in U.S.A.