NEC

NEC Electronics Inc.

# μPD70208 (V40) 8/16-Bit Microprocessor: High-Integration, CMOS

## Description

The μPD70208 (V40™) is a high-performance, low-power 16-bit microprocessor integrating a number of commonly used peripherals to dramatically reduce the size of microprocessor systems. The CMOS construction makes the μPD70208 ideal for the design of portable computers, instrumentation, and process control equipment.

The μPD70208 contains a powerful instruction set that is compatible with the μPD70108/μPD70116 (V20®/V30®) and μPD8086/μPD8088 instruction sets. Instruction set support includes a wide range of arithmetic, logical, and control operations as well as bit manipulation, BCD arithmetic, and high-speed block transfer instructions. The μPD70208 can also execute the entire μPD8080AF instruction set using the 8080 emulation mode. Also available is the μPD70216 (V50™), identical to the μPD70208 but with a 16-bit external data bus.

## Features

- ☐ Low-power CMOS technology
- ☐ V20/V30 instruction set compatible
- ☐ Minimum instruction execution time: 250 ns (at 8 MHz)
- ☐ Direct addressing of 1M bytes of memory
- ☐ Powerful set of addressing modes
- ☐ Fourteen 16-bit CPU registers
- ☐ On-chip peripherals including
  - — Clock generator
  - — Bus interface
  - — Bus arbitration
  - — Programmable wait state generator
  - — DRAM refresh controller
  - — Three 16-bit timer/counters
  - — Asynchronous serial I/O controller
  - — Eight-input interrupt controller
  - — Four-channel DMA controller
- ☐ Hardware effective address calculation logic
- ☐ Maskable and nonmaskable interrupts
- ☐ IEEE 796 compatible bus interface
- ☐ Low-power standby mode

V20 and V30 are registered trademarks of NEC Corporation.
V40 and V50 are trademarks of NEC Corporation.

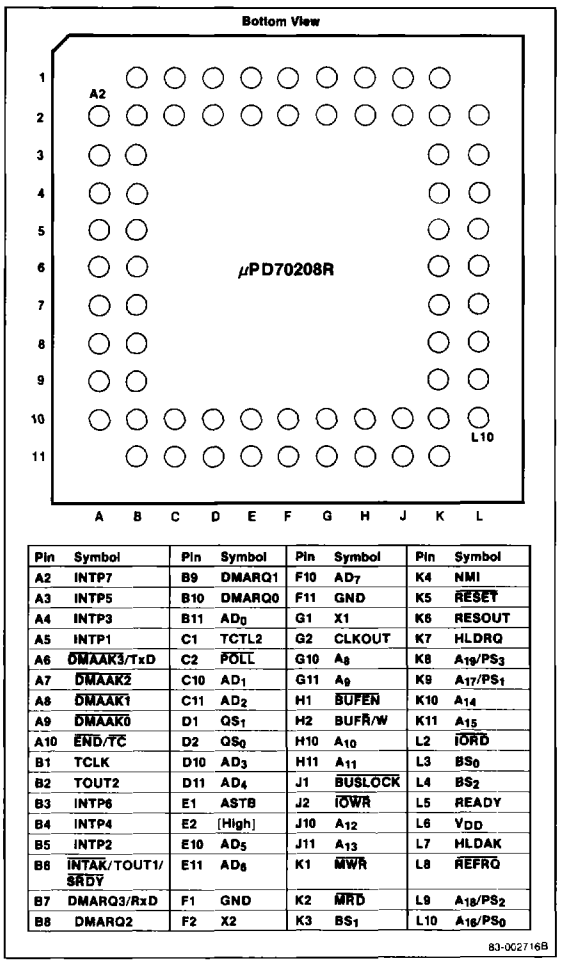
## Ordering Information

| Part Number | Max Frequency (MHz) | Package |
|---|---|---|
| μPD70208R-8 | 8 | 68-pin ceramic PGA |
| R-10 | 10 | |
| L-8 | 8 | 68-pin PLCC |
| L-10 | 10 | |
| GF-8 | 8 | 80-pin plastic QFP |
| GF-10 | 10 | |

## Pin Configurations

3c

### *68-Pin Ceramic PGA*

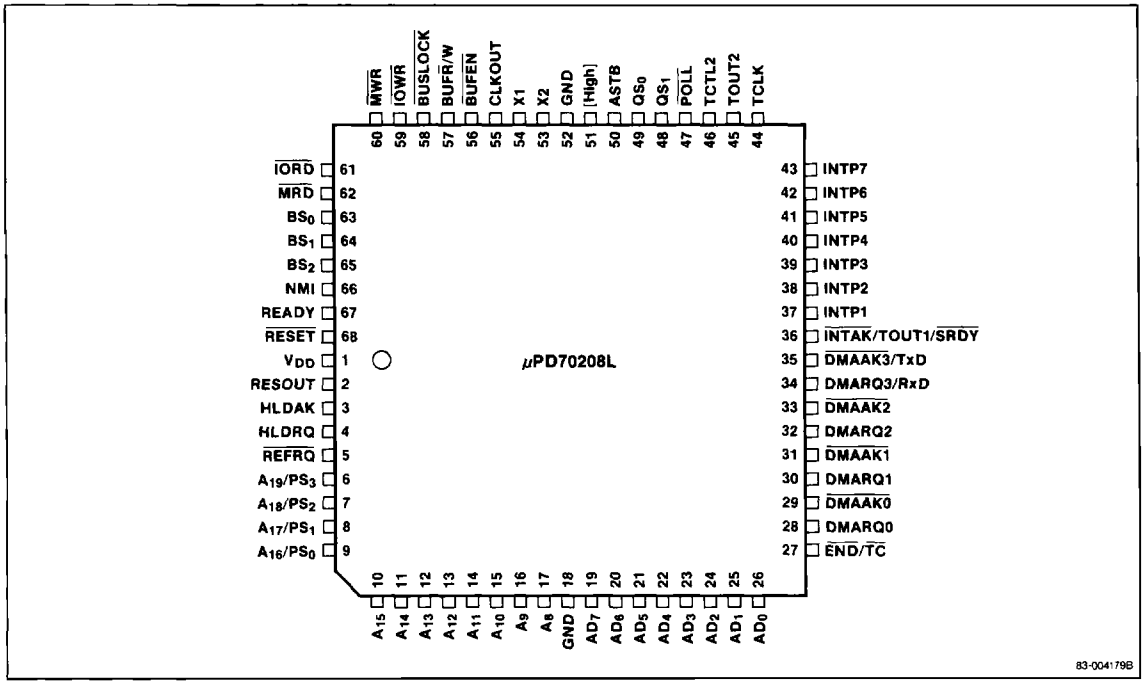



| Pin | Symbol | Pin | Symbol | Pin | Symbol | Pin | Symbol |
|---|---|---|---|---|---|---|---|
| A2 | INTP7 | B9 | DMARQ1 | F10 | $AD_7$ | K4 | NMI |
| A3 | INTP5 | B10 | DMARQ0 | F11 | GND | K5 | $\overline{RESET}$ |
| A4 | INTP3 | B11 | $AD_0$ | G1 | X1 | K6 | RESOUT |
| A5 | INTP1 | C1 | TCTL2 | G2 | CLKOUT | K7 | HLDRQ |
| A6 | $\overline{DMAAK3}$/TxD | C2 | $\overline{POLL}$ | G10 | $A_8$ | K8 | $A_{19}/PS_3$ |
| A7 | $\overline{DMAAK2}$ | C10 | $AD_1$ | G11 | $A_9$ | K9 | $A_{17}/PS_1$ |
| A8 | $\overline{DMAAK1}$ | C11 | $AD_2$ | H1 | $\overline{BUFEN}$ | K10 | $A_{14}$ |
| A9 | $\overline{DMAAK0}$ | D1 | $QS_1$ | H2 | BUFR/W | K11 | $A_{15}$ |
| A10 | $\overline{END}$/TC | D2 | $QS_0$ | H10 | $A_{10}$ | L2 | $\overline{IORD}$ |
| B1 | TCLK | D10 | $AD_3$ | H11 | $A_{11}$ | L3 | $BS_0$ |
| B2 | TOUT2 | D11 | $AD_4$ | J1 | $\overline{BUSLOCK}$ | L4 | $BS_2$ |
| B3 | INTP6 | E1 | ASTB | J2 | $\overline{IOWR}$ | L5 | READY |
| B4 | INTP4 | E2 | [High] | J10 | $A_{12}$ | L6 | $V_{DD}$ |
| B5 | INTP2 | E10 | $AD_5$ | J11 | $A_{13}$ | L7 | HLDAK |
| B6 | $\overline{INTAK}$/TOUT1/ $\overline{SRDY}$ | E11 | $AD_6$ | K1 | $\overline{MWR}$ | L8 | $\overline{REFRQ}$ |
| B7 | DMARQ3/RxD | F1 | GND | K2 | $\overline{MRD}$ | L9 | $A_{18}/PS_2$ |
| B8 | DMARQ2 | F2 | X2 | K3 | $BS_1$ | L10 | $A_{16}/PS_0$ |

83-002716B

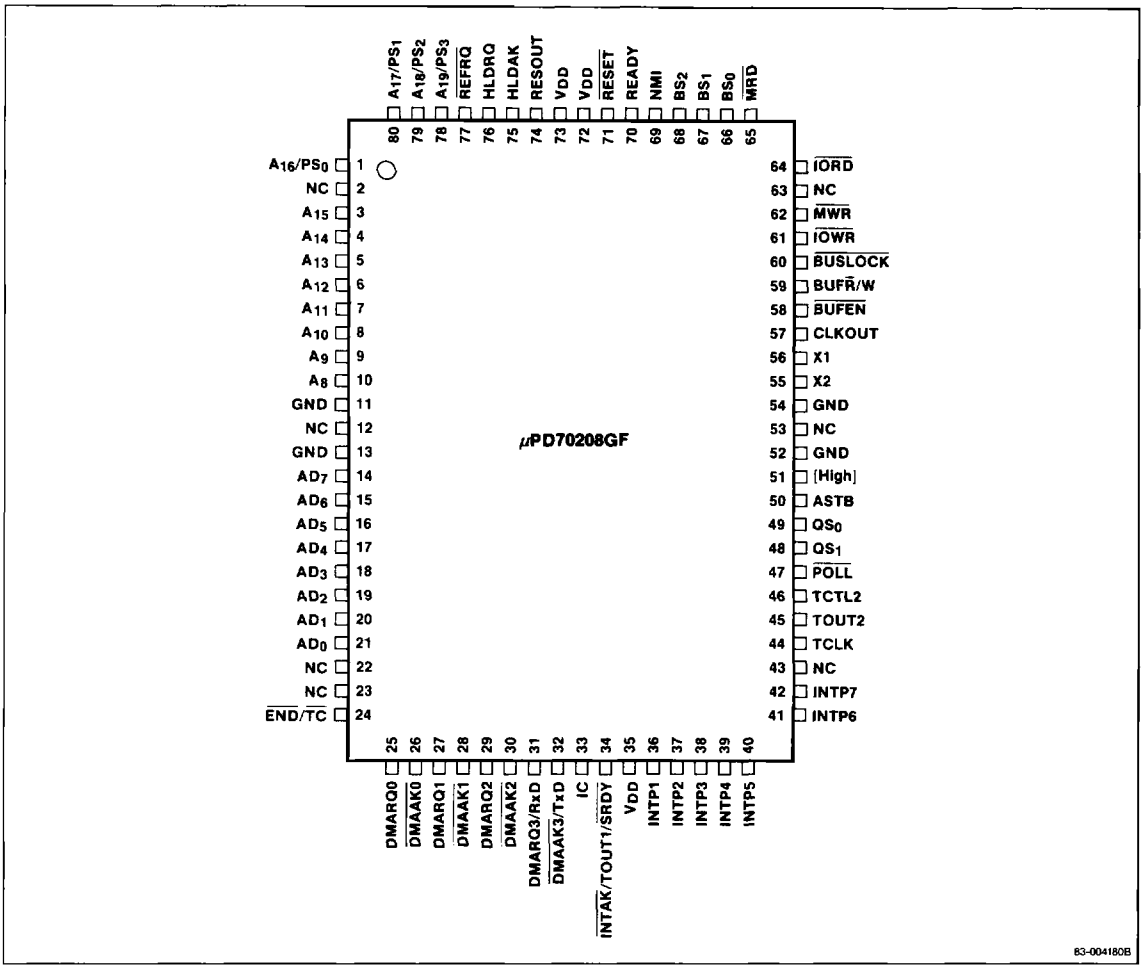
## Pin Configurations (cont)

*68-Pin Plastic Leaded Chip Carrier (PLCC)*



## Pin Configurations (cont)

*80-Pin Plastic QFP*




3c

## Pin Identification

| Symbol | Function |
|---|---|
| $A_{19}$-$A_{16}$/$PS_3$-$PS_0$ | Multiplexed address/processor status outputs |
| $A_{15}$-$A_8$ | Address bus outputs |
| $AD_7$-$AD_0$ | Multiplexed address/data bus |
| ASTB | Address strobe output |
| $BS_2$-$BS_0$ | Bus status outputs |
| $\overline{BUFEN}$ | Data bus transceiver enable output |
| BUFR/$\overline{W}$ | Data bus transceiver direction output |
| $\overline{BUSLOCK}$ | Buslock output |
| CLKOUT | System clock output |
| $\overline{DMAAK0}$ | DMA channel 0 acknowledge output |
| $\overline{DMAAK1}$ | DMA channel 1 acknowledge output |
| $\overline{DMAAK2}$ | DMA channel 2 acknowledge output |
| $\overline{DMAAK3}$/TxD | DMA channel 3 acknowledge output/Serial transmit data output |
| DMARQ0 | DMA channel 0 request input |
| DMARQ1 | DMA channel 1 request input |
| DMARQ2 | DMA channel 2 request input |
| DMARQ3/RxD | DMA channel 3 request input/Serial receive data input |
| $\overline{END}$/$\overline{TC}$ | End input/Terminal count output |
| GND | Ground |
| High | High-level output except during hold acknowledge when it is placed in the high-impedance state |
| HLDAK | Hold acknowledge output |
| HLDRQ | Hold request input |
| IC | Internal connection; leave unconnected |
| $\overline{INTAK}$/TOUT1/$\overline{SRDY}$ | Interrupt acknowledge output/Timer/counter 1 output/Serial ready output |
| INTP1-INTP7 | Interrupt request inputs |
| $\overline{IORD}$ | I/O read strobe output |
| $\overline{IOWR}$ | I/O write strobe output |
| $\overline{MRD}$ | Memory read strobe output |
| $\overline{MWR}$ | Memory write strobe output |
| NC | No connection |
| NMI | Nonmaskable interrupt input |
| $\overline{POLL}$ | Poll input |
| $QS_1$-$QS_0$ | CPU queue status outputs |
| READY | Ready input |
| $\overline{REFRQ}$ | Refresh request output |
| $\overline{RESET}$ | Reset input |
| RESOUT | Synchronized reset output |
| TCLK | Timer/counter external clock input |
| TCTL2 | Timer/counter 2 control input |
| TOUT2 | Timer/counter 2 output |
| $V_{DD}$ | +5 V power supply input |
| X1, X2 | Crystal/external clock inputs |

## Pin Functions

### $A_{19}$-$A_{16}$/$PS_3$-$PS_0$ [Address/Status Bus]

These three-state output pins contain the upper 4 bits of the 20-bit address during T1 and processor status information during the T2, T3, TW, and T4 states of a bus cycle. During T1 of a memory read or write cycle, these pins contain the upper 4 bits of the 20-bit address. These pins are forced low during T1 of an I/O bus cycle.

Processor status is output during T2, T3, TW, and T4 of both memory and I/O bus cycles. $PS_3$ is zero during any CPU native mode bus cycle. During any DMA, refresh, or 8080 emulation mode bus cycle, $PS_3$ outputs a high level. $PS_2$ outputs the contents of the interrupt enable (IE) flag in the CPU PSW register. $PS_1$ and $PS_0$ indicate the segment register used to form the physical address of a CPU bus cycle as follows:

| $PS_1$ | $PS_0$ | Segment |
|---|---|---|
| 0 | 0 | Data segment 1 (DS1) |
| 0 | 1 | Stack segment (SS) |
| 1 | 0 | Program segment (PS) |
| 1 | 1 | Data segment 0 (DS0) |

These pins are in the high-impedance state during hold acknowledge.

### $A_{15}$-$A_8$ [Address Bus]

These three-state pins form the middle byte of the active-high address bus. During any CPU, DMA, or refresh bus cycle, $A_{15}$-$A_8$ output the middle 8 bits of the 20-bit memory or I/O address. The $A_{15}$-$A_8$ pins enter the high-impedance state during hold acknowledge or an internal interrupt acknowledge bus cycle. During a slave interrupt acknowledge bus cycle, $A_{10}$-$A_8$ contain the address of the selected slave interrupt controller.

### $AD_7$-$AD_0$ [Address/Data Bus]

These three-state pins form the active-high, time-multiplexed address/data bus. During T1 of a bus cycle, $AD_7$-$AD_0$ output the lower 8 bits of the 20-bit memory or I/O address. During the T2, T3, TW, and T4 states, $AD_7$-$AD_0$ form the 8-bit bidirectional data bus.

The $AD_7$-$AD_0$ pins enter the high-impedance state during hold acknowledge or internal interrupt acknowledge bus cycles or while $\overline{\text{RESET}}$ is asserted.

### ASTB [Address Strobe]

This active-high output is used to latch the address from the multiplexed address bus in an external address latch during T1 of a bus cycle. ASTB is held at a low level during hold acknowledge.

3c

### $BS_2$-$BS_0$ [Bus Status]

Outputs $BS_2$-$BS_0$ indicate the type of bus cycle being performed as follows. $BS_2$-$BS_0$ become active during the state preceding T1 and return to the passive state during the bus state preceding T4.

| $BS_2$ | $BS_1$ | $BS_0$ | Bus Cycle |
|---|---|---|---|
| 0 | 0 | 0 | Interrupt acknowledge |
| 0 | 0 | 1 | I/O read |
| 0 | 1 | 0 | I/O write |
| 0 | 1 | 1 | Halt (Note 1) |
| 1 | 0 | 0 | Instruction fetch |
| 1 | 0 | 1 | Memory read (Note 2) |
| 1 | 1 | 0 | Memory write (Note 3) |
| 1 | 1 | 1 | Passive state |

**Note:**

(1) $BS_2$-$BS_0$ in a halt bus cycle returns to the passive state one clock earlier than normal CPU bus cycles.

(2) Memory read bus cycles include CPU, DMA read, DMA verify, and refresh bus cycles.

(3) Memory write bus cycles include CPU and DMA write bus cycles.

$BS_2$-$BS_0$ are three-state outputs and are high impedance during hold acknowledge.

### $\overline{\text{BUFEN}}$ [Buffer Enable]

$\overline{\text{BUFEN}}$ is an active-low output for enabling an external data bus transceiver during a bus cycle. $\overline{\text{BUFEN}}$ is asserted during T2 through T3 of a read cycle, T2 through T3 of a slave interrupt acknowledge cycle, and T1 through T4 of a write cycle. $\overline{\text{BUFEN}}$ is not asserted when the bus cycle corresponds to an internal peripheral, DMA, refresh, or internal interrupt acknowledge cycle. $\overline{\text{BUFEN}}$ enters the high-impedance state during hold acknowledge.

### BUF$\overline{\text{R}}$/W [Buffer Read/Write]

BUF$\overline{\text{R}}$/W is a three-state, active-low output used to control the direction of an external data bus transceiver during CPU bus cycles. A high level indicates the μPD70208 will perform a write cycle and a low level indicates a read cycle. BUF$\overline{\text{R}}$/W enters the high-impedance state during hold acknowledge.

### $\overline{\text{BUSLOCK}}$

This active-low output provides a means for the CPU to indicate to an external bus arbiter that the bus cycles of the next instruction are to be kept contiguous. $\overline{\text{BUSLOCK}}$ is asserted for the duration of the instruction following the $\overline{\text{BUSLOCK}}$ prefix. $\overline{\text{BUSLOCK}}$ is also asserted during interrupt acknowledge cycles and enters the high-impedance state during hold acknowledge. While $\overline{\text{BUSLOCK}}$ is asserted, DMAU, RCU, and external bus requests are ignored.

### CLKOUT

CLKOUT is a buffered clock output used as a reference for all timing. CLKOUT has a 50-percent duty cycle at half the frequency of the input clock source.

### $\overline{\text{DMAAK0}}$-$\overline{\text{DMAAK2}}$ [DMA Acknowledge]

This set of outputs contains the DMA acknowledge signals for channels 0-2 from the internal DMA controller and indicate that the peripheral can perform the requested transfer.

### $\overline{\text{DMAAK3}}$/TxD [DMA Acknowledge 3]/[Serial Transmit Data]

Two output signals multiplexed on this pin are selected by the PF field of the on-chip peripheral connection register.

- $\overline{\text{DMAAK3}}$ is an active-low output and enables an external DMA peripheral to perform the requested DMA transfer for channel 3.
- TxD is the serial data output from the serial control unit.

### DMARQ0-DMARQ2 [DMA Request]

These synchronized inputs are used by external peripherals to request DMA service for channels 0-2 from the internal DMA controller.

### DMARQ3/RxD [DMA Request 3]/[Serial Receive Data]

Two input signals multiplexed on this pin are selected by the PF field of the on-chip peripheral connection register.

- DMARQ3 is used by an external peripheral to request a DMA transfer cycle for channel 3.
- RxD is the serial data input to the serial control unit.

### $\overline{\text{END}}$/$\overline{\text{TC}}$ [End/Terminal Count]

This active-low bidirectional pin controls the termination of a DMA service. Assertion of $\overline{\text{END}}$ by external hardware during DMA service causes the service to terminate. When a DMA channel reaches its terminal count, the DMAU asserts $\overline{\text{TC}}$, indicating the programmed operation has completed.

$\overline{\text{END}}$/$\overline{\text{TC}}$ is an open-drain I/O pin, and requires an external 2.2-kΩ pull-up resistor.

### HLDAK [Hold Acknowledge]

When an external bus requester has become the highest priority requester, the internal bus arbiter will assert the HLDAK output indicating the address, data, and control buses have entered a high-impedance state and are available for use by the external bus master.

Should the internal DMAU or RCU (demand mode) request the bus, the bus arbiter will drive HLDAK low. When this occurs, the external bus master should complete the current bus cycle and negate the HLDRQ signal. This allows the bus arbiter to reassign the bus to the higher priority requester.

If a higher priority internal bus master subsequently requests the bus, the high-level width of HLDAK is guaranteed to be a minimum of one CLKOUT period.

### HLDRQ [Hold Request]

This active-high signal is asserted by an external bus master requesting to use the local address, data, and control buses. The HLDRQ input is used by the internal bus arbiter, which gives control of the buses to the highest priority bus requester in the following order.

| Bus Master | Priority |
|---|---|
| RCU | Highest (demand mode) |
| DMAU | • |
| HLDRQ | • |
| CPU | • |
| RCU | Lowest (normal operation) |

### $\overline{\text{INTAK}}$/TOUT1/$\overline{\text{SRDY}}$ [Interrupt Acknowledge]/[Timer 1 Output]/[Serial Ready]

Three output signals multiplexed on this pin are selected by the PF field of the on-chip peripheral connection register.

- $\overline{\text{INTAK}}$ is an interrupt acknowledge signal used to cascade external slave μPD71059 Interrupt Controllers. $\overline{\text{INTAK}}$ is asserted during T2, T3, and TW states of an interrupt acknowledge cycle.
- TOUT1 is the output of timer/counter 1.
- $\overline{\text{SRDY}}$ is an active-low output and indicates that the serial control unit is ready to receive the next character.

### INTP1-INTP7 [Peripheral Interrupts]

INTP1-INTP7 accept either rising-edge or high-level triggered asynchronous interrupt requests from external peripherals. These INTP1-INTP7 inputs are internally synchronized and prioritized by the interrupt control unit, which requests the CPU to perform an interrupt acknowledge bus cycle. External interrupt controllers such as the μPD71059 can be cascaded to increase the number of vectored interrupts.

These interrupt inputs cause the CPU to exit both the standby and 8080 emulation modes.

The INTP1-INTP7 inputs contain internal pull-up resistors and may be left unconnected.

3c

### $\overline{\text{IORD}}$ [I/O Read]

This three-state pin outputs an active-low I/O read strobe during T2, T3, and TW of an I/O read bus cycle. Both CPU I/O read and DMA write bus cycles assert $\overline{\text{IORD}}$. $\overline{\text{IORD}}$ is not asserted when the bus cycle corresponds to an internal peripheral or register. It enters the high-impedance state during hold acknowledge.

### $\overline{\text{IOWR}}$ [I/O Write]

This three-state pin outputs an active-low I/O write strobe during T2, T3, and TW of a CPU I/O write or an extended DMA read cycle and during T3 and TW of a DMA read bus cycle. $\overline{\text{IOWR}}$ is not asserted when the bus cycle corresponds to an internal peripheral or register. It enters the high-impedance state during hold acknowledge.

### $\overline{MRD}$ [Memory Read Strobe]

This three-state pin outputs an active-low memory read strobe during T2, T3, and TW of a memory read bus cycle. CPU memory read, DMA read, and refresh bus cycles all assert $\overline{MRD}$. $\overline{MRD}$ enters the high-impedance state during hold acknowledge.

### $\overline{MWR}$ [Memory Write Strobe]

This three-state pin outputs an active-low memory write strobe during T2, T3, and TW of a CPU memory write or DMA extended write bus cycle and during T3 and TW of a DMA normal write bus cycle. $\overline{MWR}$ enters the high-impedance state during hold acknowledge.

### NMI [Nonmaskable Interrupt]

The NMI pin is a rising-edge-triggered interrupt input that cannot be masked by software. NMI is sampled by CPU logic each clock cycle and when found valid for one or more CLKOUT cycles, the NMI interrupt is accepted. The CPU will process the NMI interrupt immediately after the current instruction finishes execution by fetching the segment and offset of the NMI handler from interrupt vector 2. The NMI interrupt causes the CPU to exit both the standby and 8080 emulation modes. The NMI input takes precedence over the maskable interrupt inputs.

### $\overline{POLL}$ [Poll]

The active-low $\overline{POLL}$ input is used to synchronize the operation of external devices with the CPU. During execution of the POLL instruction, the CPU checks the $\overline{POLL}$ input state every five clocks until $\overline{POLL}$ is once again asserted.

### $QS_1$-$QS_0$ [Queue Status]

The $QS_1$ and $QS_0$ outputs maintain instruction synchronization between the μPD70208 CPU and external devices. These outputs are interpreted as follows.

| $QS_1$ | $QS_0$ | Instruction Queue Status |
|---|---|---|
| 0 | 0 | No operation |
| 0 | 1 | First byte of instruction fetched |
| 1 | 0 | Flush queue contents |
| 1 | 1 | Subsequent byte of instruction fetched |

Queue status is valid for one clock cycle after the CPU has accessed the instruction queue.

### READY [Ready]

This active-high input synchronizes external memory and peripheral devices with the μPD70208. Slow memory and I/O devices can lengthen a bus cycle by negating the READY input and forcing the BIU to insert TW states. READY must be negated prior to the rising edge of CLKOUT during the T2 state or by the last internally generated TW state to guarantee recognition. When READY is once again asserted and recognized by the BIU, the BIU will proceed to the T4 state.

The READY input operates in parallel with the internal μPD70208 wait control unit and can be used to insert more than three wait states into a bus cycle.

### $\overline{REFRQ}$ [Refresh Request]

$\overline{REFRQ}$ is an active-low output indicating the current bus cycle is a memory refresh operation. $\overline{REFRQ}$ is used to disable memory address decode logic and refresh dynamic memories. The 9-bit refresh row address is placed on $A_8$-$A_0$ during a refresh bus cycle.

### $\overline{RESET}$ [Reset]

$\overline{RESET}$ is a Schmitt trigger input used to force the μPD70208 to a known state by resetting the CPU and on-chip peripherals. $\overline{RESET}$ must be asserted for a minimum of four clocks to guarantee recognition. After $\overline{RESET}$ has been released, the CPU will start program execution from address FFFF0H in the native mode.

$\overline{RESET}$ will release the CPU from the low-power standby mode and force it to the native mode.

### RESOUT [Reset Output]

This active-high output is available to perform a system-wide reset function. $\overline{RESET}$ is internally synchronized with CLKOUT and output on the RESOUT pin.

### TCLK

TCLK is an external clock source for the timer control unit. The three timer/counters can be programmed to operate with either the TCLK input or a prescaled CLKOUT input.

### TCTL2

TCTL2 is the control input for timer/counter 2.

### TOUT2

TOUT2 is the output of timer/counter 2.

### X1, X2 [Clock Inputs]

These pins accept either a parallel resonant, fundamental mode crystal or an external oscillator input with a frequency twice the desired operating frequency.

In the case of an external clock generator, the X2 pin can be either left unconnected or be driven by the complement of the X1 pin clock source.

## Pin States

Table 1 lists the output pin states during the Hold, Halt, Reset, and DMA Cascade conditions.

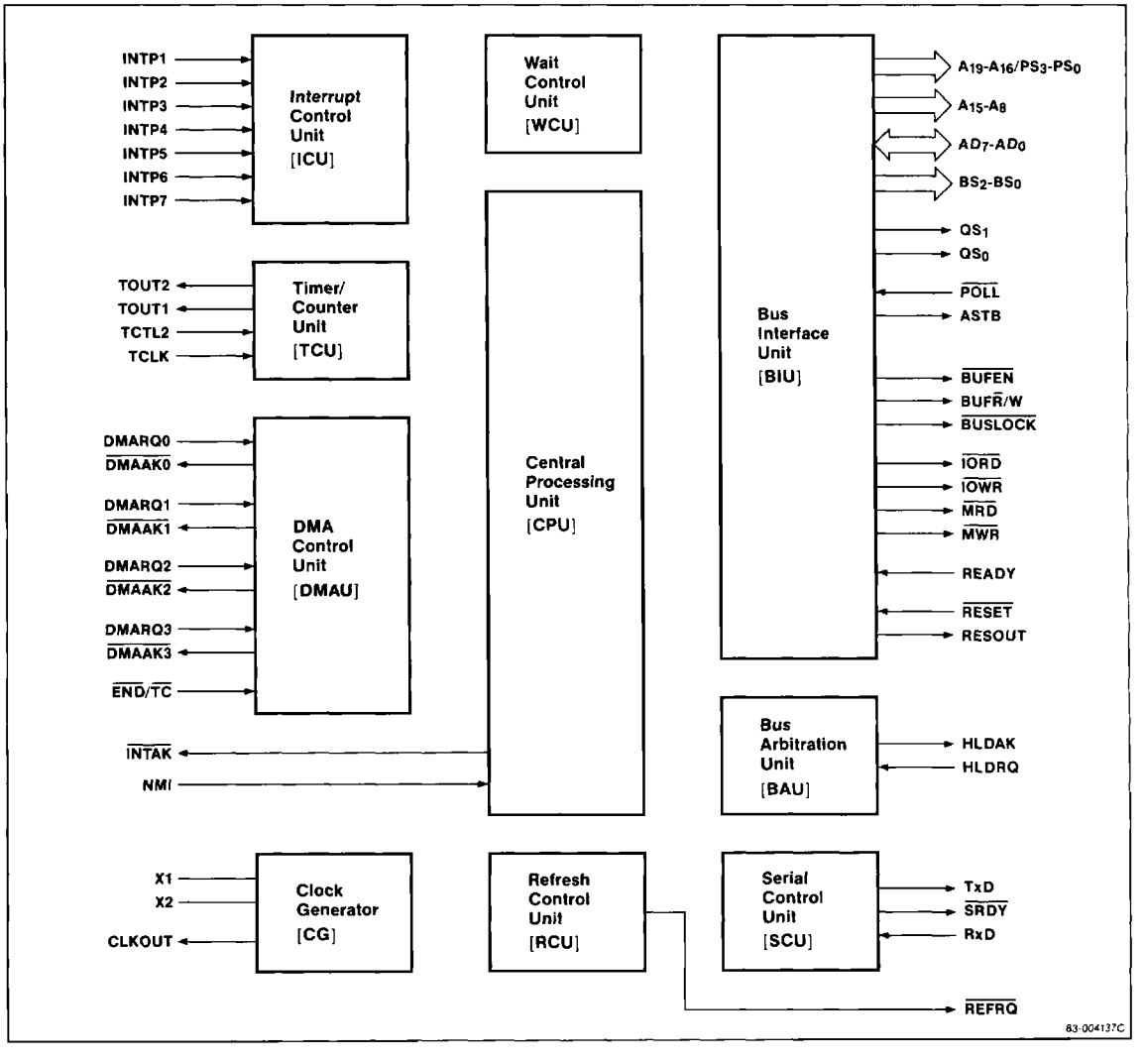
*Table 1. Input/Output Pin States*

| Symbol | Pin Type | Hold | Halt | Reset | DMA Cascade |
|---|---|---|---|---|---|
| $A_{19}$-$A_{16}$/$PS_3$-$PS_0$, $A_{15}$-$A_8$ | 3-state Out | Hi-Z | H/L | H/L | Hi-Z |
| $AD_7$-$AD_0$ | 3-state I/O | Hi-Z | H/L | Hi-Z | Hi-Z |
| ASTB | Out | L | L | L | L |
| $\overline{BUFEN}$ | 3-state Out | Hi-Z | H | H | Hi-Z |
| BUFR/$\overline{W}$ | 3-state Out | Hi-Z | H/L | H | Hi-Z |
| $\overline{BUSLOCK}$ | 3-state Out | Hi-Z | H/L | H | Hi-Z |
| $BS_2$-$BS_0$ | 3-state Out | Hi-Z | H | H | H |
| CLKOUT | Out | H/L | H/L | H/L | H/L |
| $\overline{DMAAK0}$-$\overline{DMAAK2}$ | Out | H | H/L | H | H/L |
| $\overline{DMAAK3}$ | Out | H | H/L | H | H/L |
| TxD | | H/L | | H/L | H/L |
| $\overline{END/TC}$ | I/O | H | H/L | H | H |
| HLDAK | Out | H | H/L | L | L |
| $\overline{INTAK}$ | Out | H | H | H | H |
| TOUT1 | | H/L | H/L | | H/L |
| $\overline{SRDY}$ | | H/L | H/L | | H/L |
| $\overline{IORD}$ | 3-state Out | Hi-Z | H | H | Hi-Z |
| $\overline{IOWR}$ | 3-state Out | Hi-Z | H | H | Hi-Z |
| $\overline{MRD}$ | 3-state Out | Hi-Z | H | H | Hi-Z |
| $\overline{MWR}$ | 3-state Out | Hi-Z | H | H | Hi-Z |
| $QS_1$-$QS_0$ | Out | H/L | L | L | H/L |
| $\overline{REFRQ}$ | Out | H | H/L | H | H |
| RESOUT | Out | L | L | H | L |
| TOUT2 | Out | H/L | H/L | H/L | H/L |

H: high level; L: low level; H/L: high or low level; Hi-Z: high impedance.

3c

## Block Diagram



## Absolute Maximum Ratings

$T_A = +25°C$

| | |
|---|---|
| Power supply voltage, $V_{DD}$ | −0.5 to +7.0 V |
| Input voltage, $V_I$ | −0.5 to $V_{DD}$ + 0.3 V |
| CLK input voltage, $V_K$ | −0.5 to $V_{DD}$ + 1.0 V |
| Output voltage, $V_O$ | −0.5 to $V_{DD}$ + 0.3 V |
| Operating temperature, $T_{OPT}$ | −10 to +70°C |
| Storage temperature, $T_{STG}$ | −65 to +150°C |

**Comment:** Exposure to Absolute Maximum Ratings for extended periods may affect device reliability; exceeding the ratings could cause permanent damage. The device should be operated within the limits specified under DC and AC Characteristics.

## Capacitance

$T_A = +25°C$, $V_{DD} = 0$ V

| Parameter | Symbol | Limits Min | Limits Max | Unit | Test Conditions |
|---|---|---|---|---|---|
| Input capacitance | $C_I$ | | 15 | pF | $f_C = 1$ MHz; unmeasured pins are returned to 0 V. |
| Output capacitance | $C_O$ | | 15 | pF | |

## DC Characteristics

$T_A = -10$ to +70°C, $V_{DD} = +5$ V ±10% (8 MHz), $V_{DD} = 5$ V ±5% (10 MHz)

| Parameter | Symbol | Limits Min | Limits Max | Unit | Test Conditions |
|---|---|---|---|---|---|
| Input voltage, high | $V_{IH}$ | 2.2 | $V_{DD}$ + 0.3 | V | |
| Input voltage, low | $V_{IL}$ | −0.5 | 0.8 | V | |
| X1, X2 input voltage, high | $V_{KH}$ | 3.9 | $V_{DD}$ + 1.0 | V | |
| X1, X2 input voltage, low | $V_{KL}$ | −0.5 | 0.6 | V | |
| Output voltage, high | $V_{OH}$ | 0.7 $V_{DD}$ | | V | $I_{OH} = -400$ μA |
| Output voltage, low | $V_{OL}$ | | 0.4 | V | $I_{OL} = 2.5$ mA |
| Input leakage current, high | $I_{LIH}$ | | 10 | μA | $V_I = V_{DD}$ |
| Input leakage current, low | $I_{LIPL}$ | | −300 | μA | $V_I = 0$ V, INTP input pins |
| | $I_{LIL}$ | | −10 | μA | $V_I = 0$ V, other input pins |
| Output leakage current, high | $I_{LOH}$ | | 10 | μA | $V_O = V_{DD}$ |
| Output leakage current, low | $I_{LOL}$ | | −10 | μA | $V_O = 0$ V |
| Supply current 8 MHz | $I_{DD}$ | | 90 | mA | Normal mode |
| | | | 20 | mA | Standby mode |
| 10 MHz | $I_{DD}$ | | 120 | mA | Normal mode |
| | | | 25 | mA | Standby mode |

3c

## AC Characteristics

$T_A = -10$ to $+70°C$; $V_{DD} = 5\ V \pm 10\%$ (8 MHz), $V_{DD} = 5\ V \pm 5\%$ (10 MHz), $C_L = 100\ pF$

| | | 8 MHz Limits | | 10 MHz Limits | | | |
|---|---|---|---|---|---|---|---|
| **Parameter** | **Symbol** | **Min** | **Max** | **Min** | **Max** | **Unit** | **Test Conditions** |
| External clock input cycle time | $t_{CYX}$ | 62 | 250 | 50 | 250 | ns | |
| External clock pulse width, high | $t_{XXH}$ | 20 | | 19 | | ns | $V_{KH} = 3.0\ V$ |
| External clock pulse width, low | $t_{XXL}$ | 20 | | 19 | | ns | $V_{KL} = 1.5\ V$ |
| External clock rise time | $t_{XR}$ | | 10 | | 5 | ns | $1.5 \rightarrow 3.0\ V$ |
| External clock fall time | $t_{XF}$ | | 10 | | 5 | ns | $3.0 \rightarrow 1.5\ V$ |
| CLKOUT cycle time | $t_{CYK}$ | 124 | 500 | 100 | 500 | ns | |
| CLKOUT pulse width, high | $t_{KKH}$ | $0.5\ t_{CYK} - 7$ | | $0.5\ t_{CYK} - 5$ | | ns | $V_{KH} = 3.0\ V$ |
| CLKOUT pulse width, low | $t_{KKL}$ | $0.5\ t_{CYK} - 7$ | | $0.5\ t_{CYK} - 5$ | | ns | $V_{KL} = 1.5\ V$ |
| CLKOUT rise time | $t_{KR}$ | | 7 | | 5 | ns | $1.5 \rightarrow 3.0\ V$ |
| CLKOUT fall time | $t_{KF}$ | | 7 | | 5 | ns | $3.0 \rightarrow 1.5\ V$ |
| CLKOUT delay time from external clock | $t_{DXK}$ | | 55 | | 40 | ns | |
| Input rise time (except external clock) | $t_{IR}$ | | 20 | | 15 | ns | $0.8 \rightarrow 2.2\ V$ |
| Input fall time (except external clock) | $t_{IF}$ | | 12 | | 10 | ns | $2.2 \rightarrow 0.8\ V$ |
| Output rise time (except CLKOUT) | $t_{OR}$ | | 20 | | 15 | ns | $0.8 \rightarrow 2.2\ V$ |
| Output fall time (except CLKOUT) | $t_{OF}$ | | 12 | | 10 | ns | $2.2 \rightarrow 0.8\ V$ |
| $\overline{RESET}$ setup time to CLKOUT↓ | $t_{SRESK}$ | 25 | | 20 | | ns | |
| $\overline{RESET}$ hold time after CLKOUT↓ | $t_{HKRES}$ | 35 | | 25 | | ns | |
| RESOUT delay time from CLKOUT↓ | $t_{DKRES}$ | 5 | 60 | 5 | 50 | ns | |
| READY inactive setup time to CLKOUT↑ | $t_{SRYLK}$ | 15 | | 15 | | ns | |
| READY inactive hold time after CLKOUT↑ | $t_{HKRYL}$ | 25 | | 20 | | ns | |
| READY active setup time to CLKOUT↑ | $t_{SRYHK}$ | 15 | | 15 | | ns | |
| READY active hold time after CLKOUT↑ | $t_{HKRYH}$ | 25 | | 20 | | ns | |
| NMI, $\overline{POLL}$ setup time to CLKOUT↑ | $t_{SIK}$ | 15 | | 15 | | ns | |
| Data setup time to CLKOUT↓ | $t_{SDK}$ | 15 | | 15 | | ns | |
| Data hold time after CLKOUT↓ | $t_{HKD}$ | 10 | | 10 | | ns | |
| Address delay time from CLKOUT↓ | $t_{DKA}$ | 10 | 55 | 10 | 50 | ns | $A_{19}$-$A_0$ $\overline{UBE}$ |
| Address hold time after CLKOUT↓ | $t_{HKA}$ | 10 | | 10 | | ns | |
| I/O recovery time | $t_{AI}$ | $2t_{CYK} - 50$ | | $2t_{CYK} - 40$ | | ns | (Note 1) |
| PS delay time from CLKOUT↓ | $t_{DKP}$ | 10 | 60 | 10 | 50 | ns | |
| PS float delay time from CLKOUT↑ | $t_{FKP}$ | 10 | 60 | 10 | 50 | ns | |
| Address setup time to ASTB↓ | $t_{SAST}$ | $t_{KKL} - 20$ | | $t_{KKL} - 30$ | | ns | |
| Address float delay time from CLKOUT↓ | $t_{FKA}$ | $t_{HKA}$ | 60 | $t_{HKA}$ | 50 | ns | |
| ASTB↑ delay time from CLKOUT↓ | $t_{DKSTH}$ | | 45 | | 40 | ns | |
| ASTB↓ delay time from CLKOUT↑ | $t_{DKSTL}$ | | 50 | | 45 | ns | |
| ASTB pulse width, high | $t_{STST}$ | $t_{KKL} - 10$ | | $t_{KKL} - 10$ | | ns | |
| Address hold time after ASTB↓ | $t_{HSTA}$ | $t_{KKH} - 20$ | | $t_{KKH} - 20$ | | ns | |
| Control delay time from CLKOUT | $t_{DKCT1}$ | 10 | 70 | 10 | 60 | ns | (Note 2) |
| | $t_{DKCT2}$ | 10 | 60 | 10 | 55 | ns | (Note 3) |

## AC Characteristics (cont)

| Parameter | Symbol | 8 MHz Limits Min | 8 MHz Limits Max | 10 MHz Limits Min | 10 MHz Limits Max | Unit | Test Conditions |
|---|---|---|---|---|---|---|---|
| $\overline{RD}\downarrow$ delay time from address float | $t_{DAFRL}$ | 0 | | 0 | | ns | (Note 4) |
| $\overline{RD}\downarrow$ delay time from CLKOUT$\downarrow$ | $t_{DKRL}$ | 10 | 75 | 10 | 65 | ns | |
| $\overline{RD}\uparrow$ delay time from CLKOUT$\downarrow$ | $t_{DKRH}$ | 10 | 70 | 10 | 60 | ns | |
| $\overline{REFRQ}\uparrow$ delay from $\overline{MRD}\uparrow$ | $t_{DRQHRH}$ | $t_{KKL} - 30$ | | $t_{KKL} - 30$ | | ns | (Note 5) |
| Address delay time from $\overline{RD}\uparrow$ | $t_{DRHA}$ | $t_{CYK} - 40$ | | $t_{CYK} - 40$ | | ns | |
| $\overline{RD}$ pulse width, low | $t_{RR}$ | $2t_{CYK} - 50$ | | $2t_{CYK} - 40$ | | ns | |
| BUF$\overline{R}$/W delay from $\overline{BUFEN}\uparrow$ | $t_{DBECT}$ | $t_{KKL} - 20$ | | $t_{KKL} - 20$ | | ns | Read cycle |
| | $t_{DWCT}$ | $t_{KKL} - 20$ | | $t_{KKL} - 20$ | | ns | Write cycle |
| Data output delay time from CLKOUT$\downarrow$ | $t_{DKD}$ | 10 | 60 | 10 | 55 | ns | |
| Data float delay time from CLKOUT$\downarrow$ | $t_{FKD}$ | 10 | 60 | 10 | 55 | ns | |
| $\overline{WR}$ pulse width, low | $t_{WW}$ | $2t_{CYK} - 40$ | | $2t_{CYK} - 40$ | | ns | (Note 4) |
| $\overline{BS}\downarrow$ delay time from CLKOUT$\uparrow$ | $t_{DKBL}$ | 10 | 60 | 10 | 55 | ns | |
| $\overline{BS}\uparrow$ delay time from CLKOUT$\downarrow$ | $t_{DKBH}$ | 10 | 60 | 10 | 55 | ns | |
| HLDRQ setup time to CLKOUT$\uparrow$ | $t_{SHQK}$ | 20 | | 15 | | ns | |
| HLDAK delay time from CLKOUT$\downarrow$ | $t_{DKHA}$ | 10 | 70 | 10 | 60 | ns | |
| $\overline{DMAAK}\downarrow$ delay time from CLKOUT$\uparrow$ | $t_{DKHDA}$ | 10 | 60 | 10 | 55 | ns | |
| $\overline{DMAAK}\downarrow$ delay time from CLKOUT$\downarrow$ | $t_{DKLDA}$ | 10 | 90 | 10 | 80 | ns | Cascade mode |
| $\overline{WR}$ pulse width, low (DMA cycle) | $t_{WW1}$ | $2t_{CYK} - 40$ | | $2t_{CYK} - 40$ | | ns | DMA extended write cycle |
| $\overline{WR}$ pulse width, low (DMA cycle) | $t_{WW2}$ | $t_{CYK} - 40$ | | $t_{CYK} - 40$ | | ns | DMA normal write cycle |
| $\overline{RD}\downarrow$, $\overline{WR}\downarrow$ delay from DMAAK$\downarrow$ | $t_{DDARW}$ | $t_{KKH} - 30$ | | $t_{KKH} - 30$ | | ns | |
| $\overline{DMAAK}\uparrow$ delay from $\overline{RD}\uparrow$ | $t_{DRHDAH}$ | $t_{KKL} - 30$ | | $t_{KKL} - 30$ | | ns | |
| $\overline{RD}\uparrow$ delay from $\overline{WR}\uparrow$ | $t_{DWHRH}$ | 5 | | 5 | | ns | |
| $\overline{TC}$ output delay time from CLKOUT$\uparrow$ | $t_{DKTCL}$ | | 60 | | 55 | ns | |
| $\overline{TC}$ off delay time from CLKOUT$\uparrow$ | $t_{DKTCF}$ | | 60 | | 55 | ns | |
| $\overline{TC}$ pulse width, low | $t_{TCTCL}$ | $t_{CYK} - 15$ | | $t_{CYK} - 15$ | | ns | |
| $\overline{TC}$ pullup delay time from CLKOUT$\uparrow$ | $t_{DKTCH}$ | | $t_{KKH} + t_{CYK} - 10$ | | $t_{KKH} + t_{CYK} - 10$ | ns | |
| $\overline{END}$ setup time to CLKOUT$\uparrow$ | $t_{SEDK}$ | 35 | | 30 | | ns | |
| $\overline{END}$ pulse width, low | $t_{EDEDL}$ | 100 | | 80 | | ns | |
| DMARQ setup time to CLKOUT$\uparrow$ | $t_{SDQK}$ | 35 | | 30 | | ns | |
| INTPn pulse width, low | $t_{IPIPL}$ | 100 | | 80 | | ns | |
| RxD setup time to SCU internal clock$\downarrow$ | $t_{SRX}$ | 1 | | 0.5 | | μs | |
| RxD hold time after SCU internal clock$\downarrow$ | $t_{HRX}$ | 1 | | 0.5 | | μs | |
| $\overline{SRDY}$ delay time from CLKOUT$\downarrow$ | $t_{DKSR}$ | | 150 | | 100 | ns | |

**Notes:**

(1) This is specified to guarantee a read/write recovery time for I/O devices.

(2) Delay from CLKOUT to DMA cycle $\overline{MWR}$/$\overline{IOWR}$ outputs.

(3) Delay from CLKOUT to BUF$\overline{R}$/W, $\overline{BUFEN}$, $\overline{INTAK}$, $\overline{REFRQ}$ outputs and CPU cycle $\overline{MWR}$/$\overline{IOWR}$ outputs.

(4) $\overline{RD}$ represents $\overline{IORD}$ and $\overline{MRD}$. $\overline{WR}$ represents $\overline{IOWR}$ and $\overline{MWR}$.

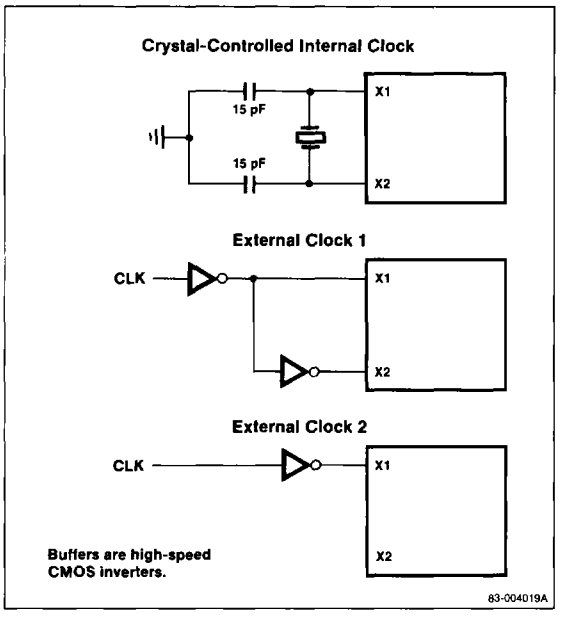
(5) This is specified to guarantee that $\overline{REFRQ}\uparrow$ is delayed from $\overline{MRD}\uparrow$ at all times.

3c

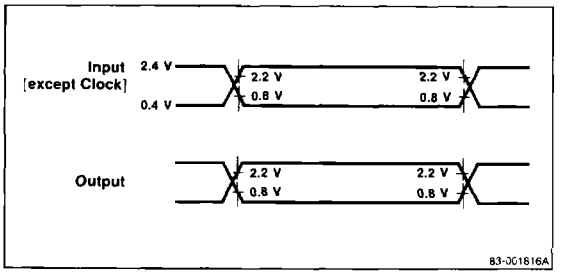
## AC Characteristics (cont)

| Parameter | Symbol | 8 MHz Limits | | 10 MHz Limits | | Unit | Test Conditions |
|---|---|---|---|---|---|---|---|
| | | Min | Max | Min | Max | | |
| TxD delay time from TOUT1↓ | $t_{DTX}$ | | 500 | | 200 | ns | |
| TCTL2 setup time from CLKOUT↓ | $t_{SGK}$ | 50 | | 40 | | ns | |
| TCTL2 setup time to TCLK↑ | $t_{SGTK}$ | 50 | | 40 | | ns | |
| TCTL2 hold time after CLKOUT↓ | $t_{HKG}$ | 100 | | 80 | | ns | |
| TCTL2 hold time after TCLK↑ | $t_{HTKG}$ | 50 | | 40 | | ns | |
| TCTL2 pulse width, high | $t_{GGH}$ | 50 | | 40 | | ns | |
| TCTL2 pulse width, low | $t_{GGL}$ | 50 | | 40 | | ns | |
| TOUT output delay time from CLKOUT↓ | $t_{DKTO}$ | | 200 | | 150 | ns | |
| TOUT output delay time from TOUT↓ | $t_{DTKTO}$ | | 150 | | 100 | ns | |
| TOUT output delay time from TCTL2↓ | $t_{DGTO}$ | | 120 | | 90 | ns | |
| TCLK rise time | $t_{TKR}$ | | 25 | | 25 | ns | |
| TCLK fall time | $t_{TKF}$ | | 25 | | 25 | ns | |
| TCLK pulse width, high | $t_{TKTKH}$ | 50 | | 45 | | ns | |
| TCLK pulse width, low | $t_{TKTKL}$ | 50 | | 45 | | ns | |
| TCLK cycle time | $t_{CYTK}$ | 124 | DC | 100 | DC | ns | |
| RESET pulse width low | $t_{RESET1}$ | 50 | | 50 | | μs | After power on |
| | $t_{RESET2}$ | 4 $t_{CYK}$ | | 4 $t_{CYK}$ | | | During operation |

### Clock Input Configurations



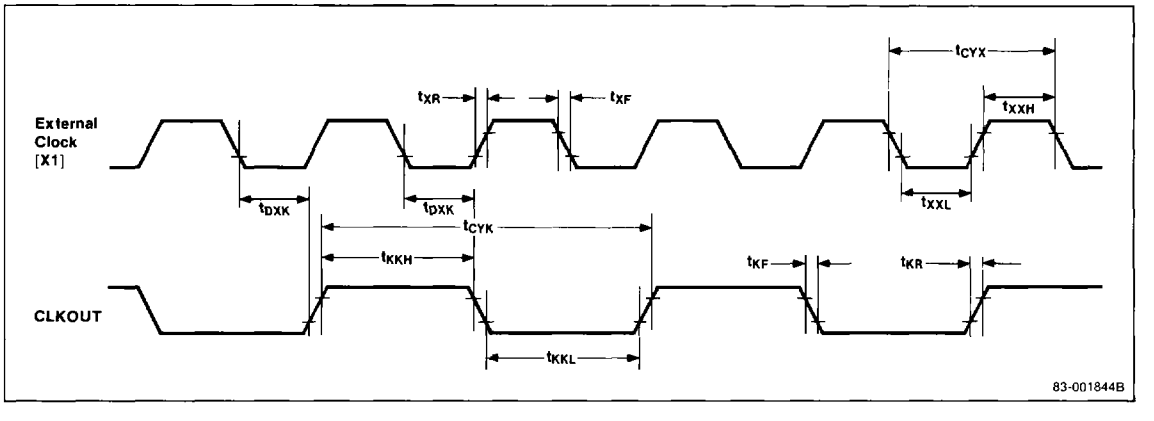

### Timing Measurement Points




3c

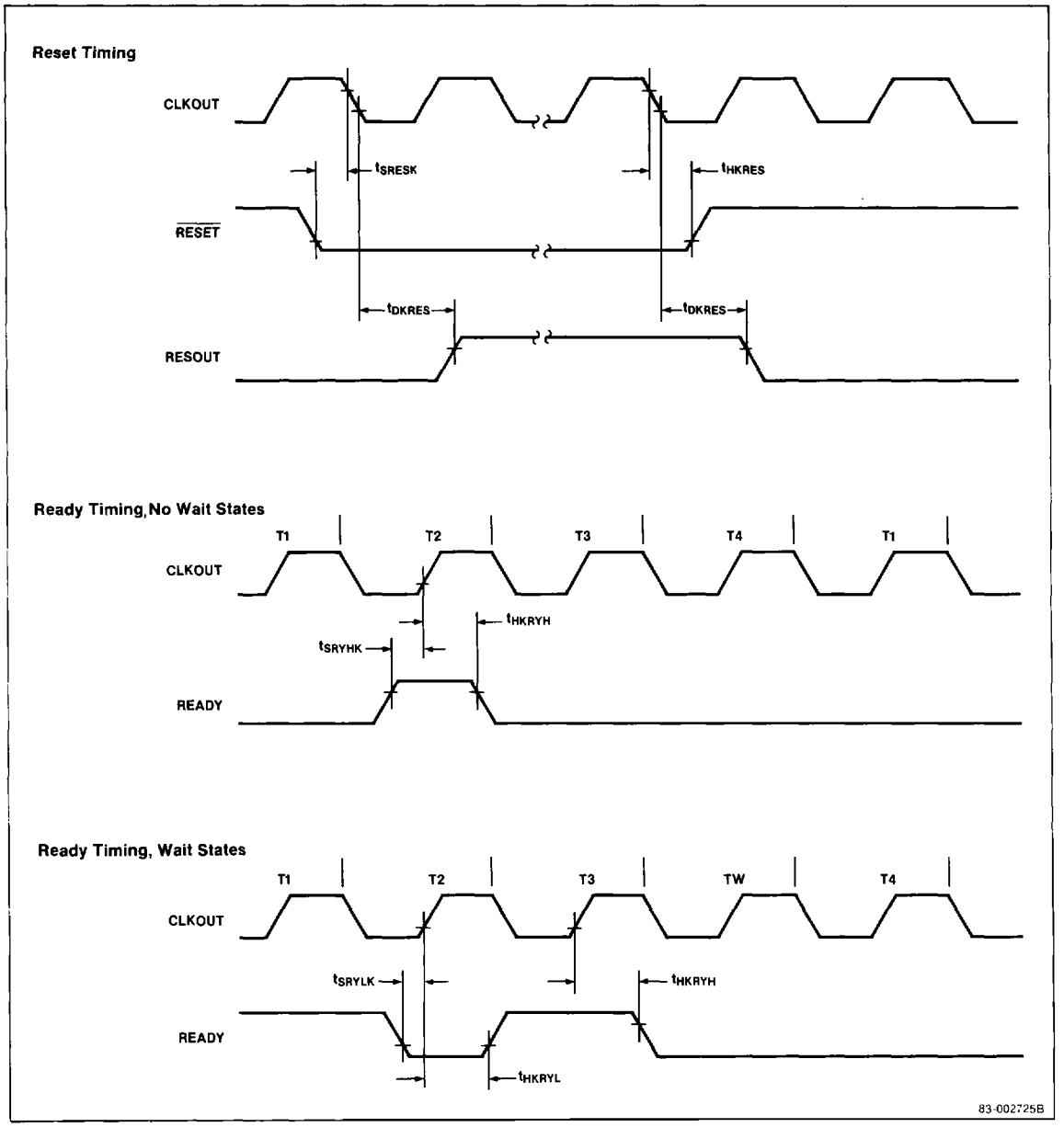
## Timing Waveforms

### Clock Timing



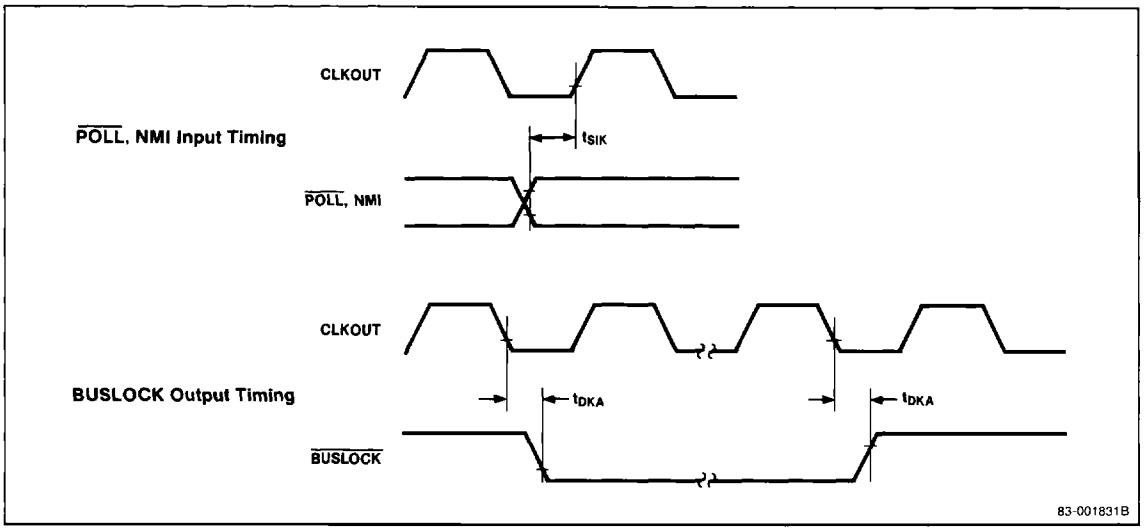
## Timing Waveforms (cont)

### *Reset and Ready Timing*

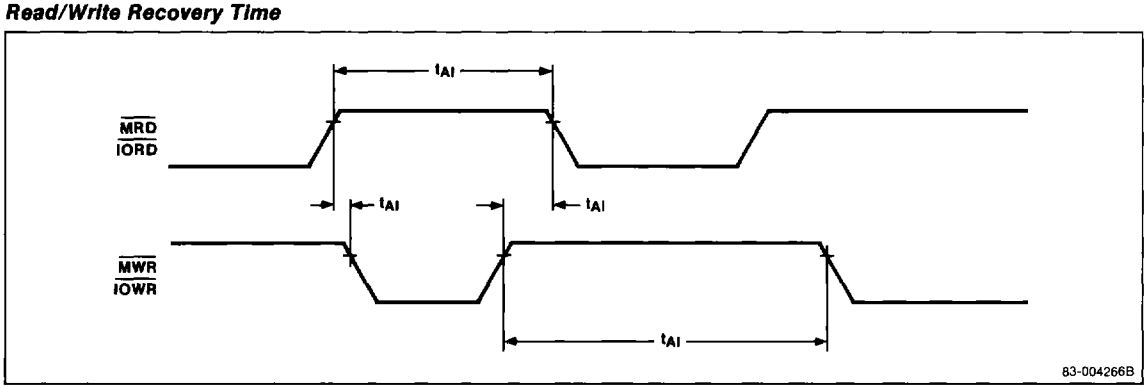


## Timing Waveforms (cont)

*Poll, NMI, and Buslock Timing*



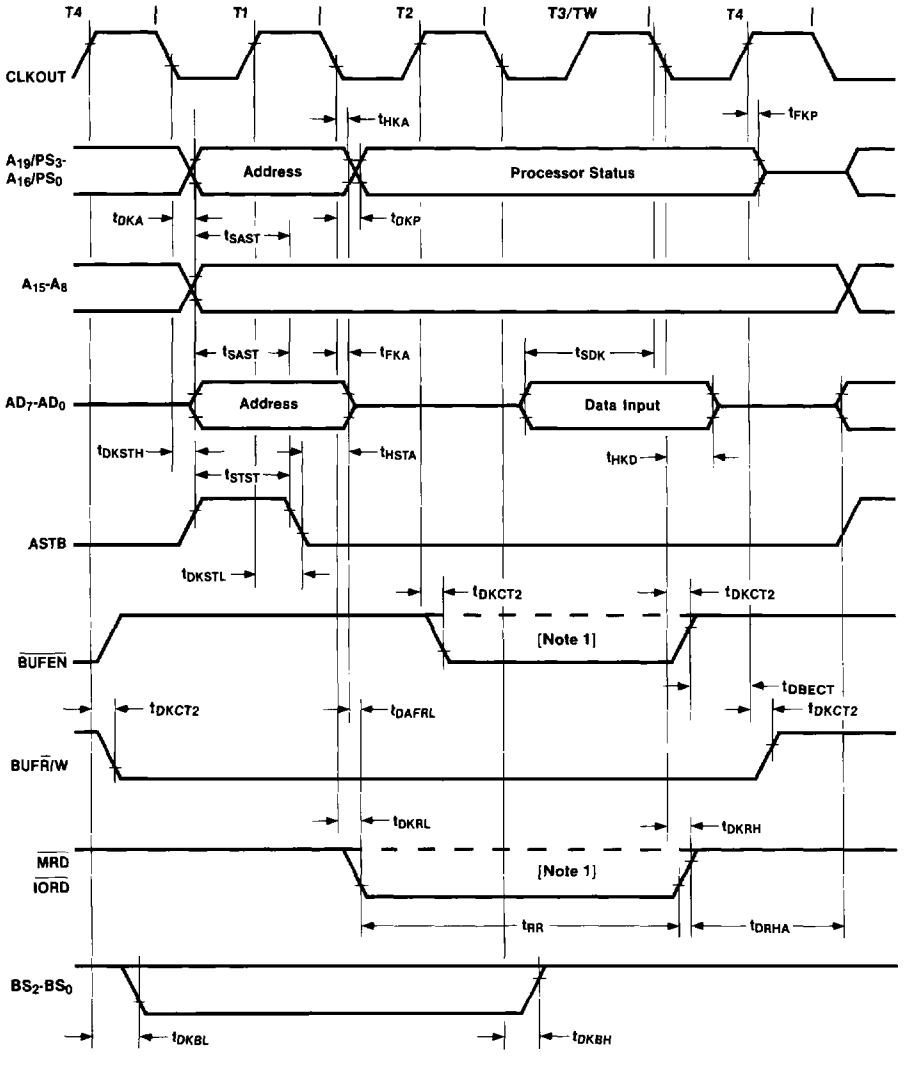

*Read/Write Recovery Time*




3c

## Timing Waveforms (cont)

### *Read Timing*

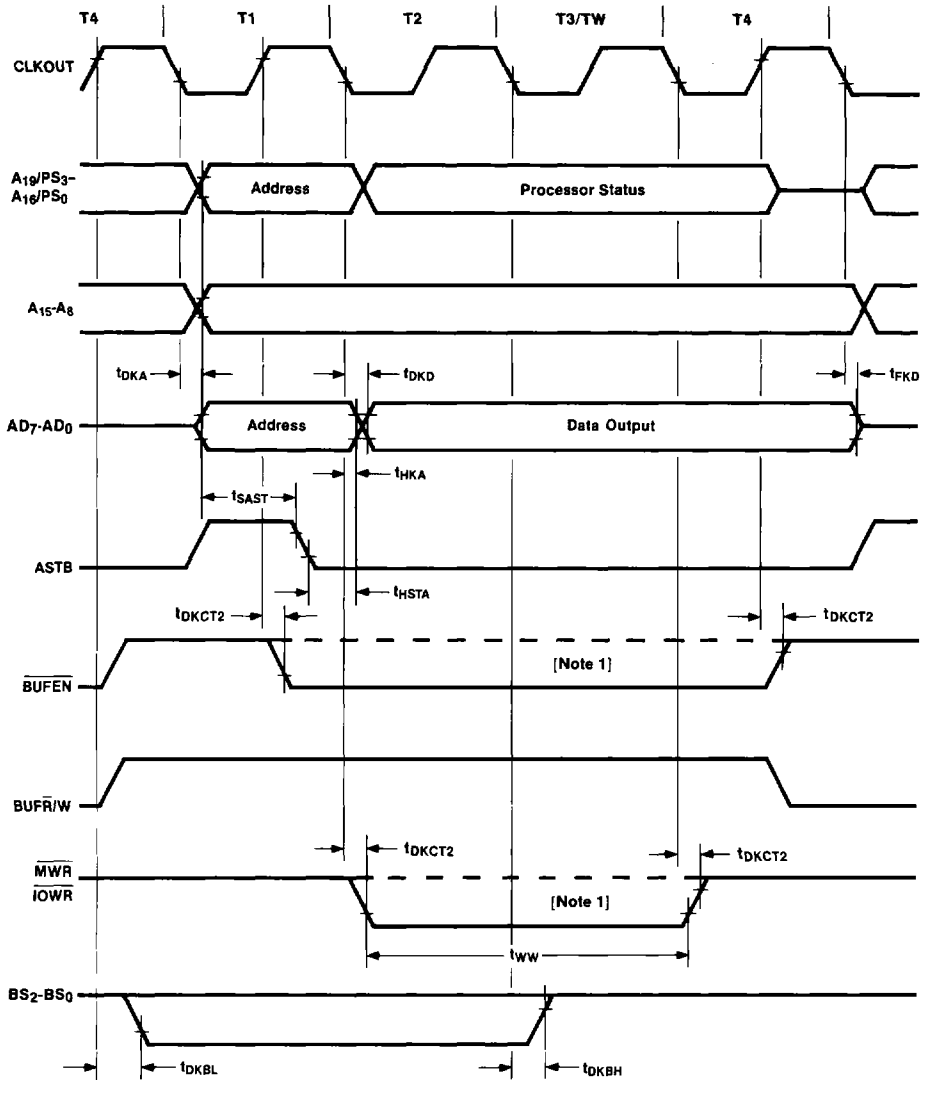



Note:
[1] Except internal I/O accesses.

83-001848B

## Timing Waveforms (cont)

*Write Timing*

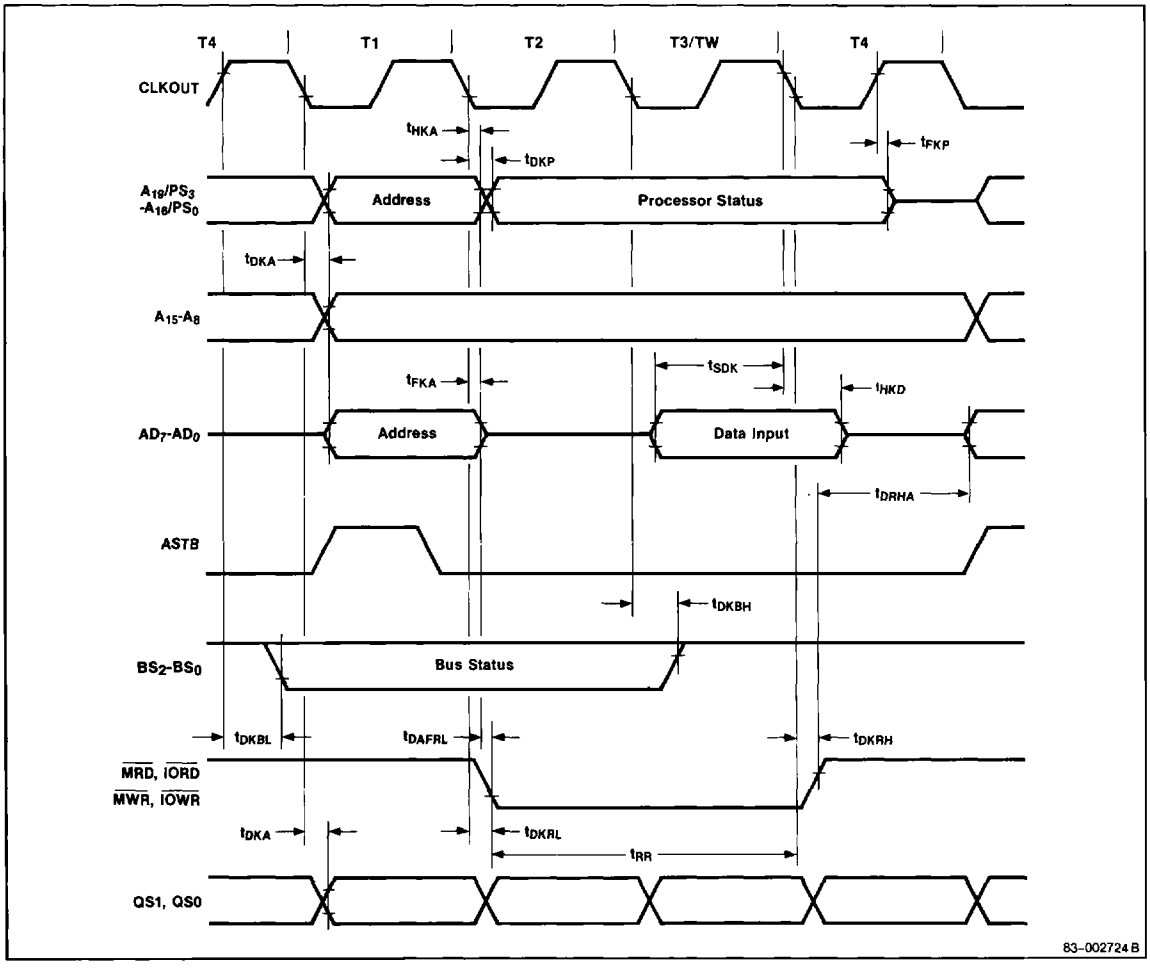



Note:
[1] Except internal I/O accesses.

83-001846 B

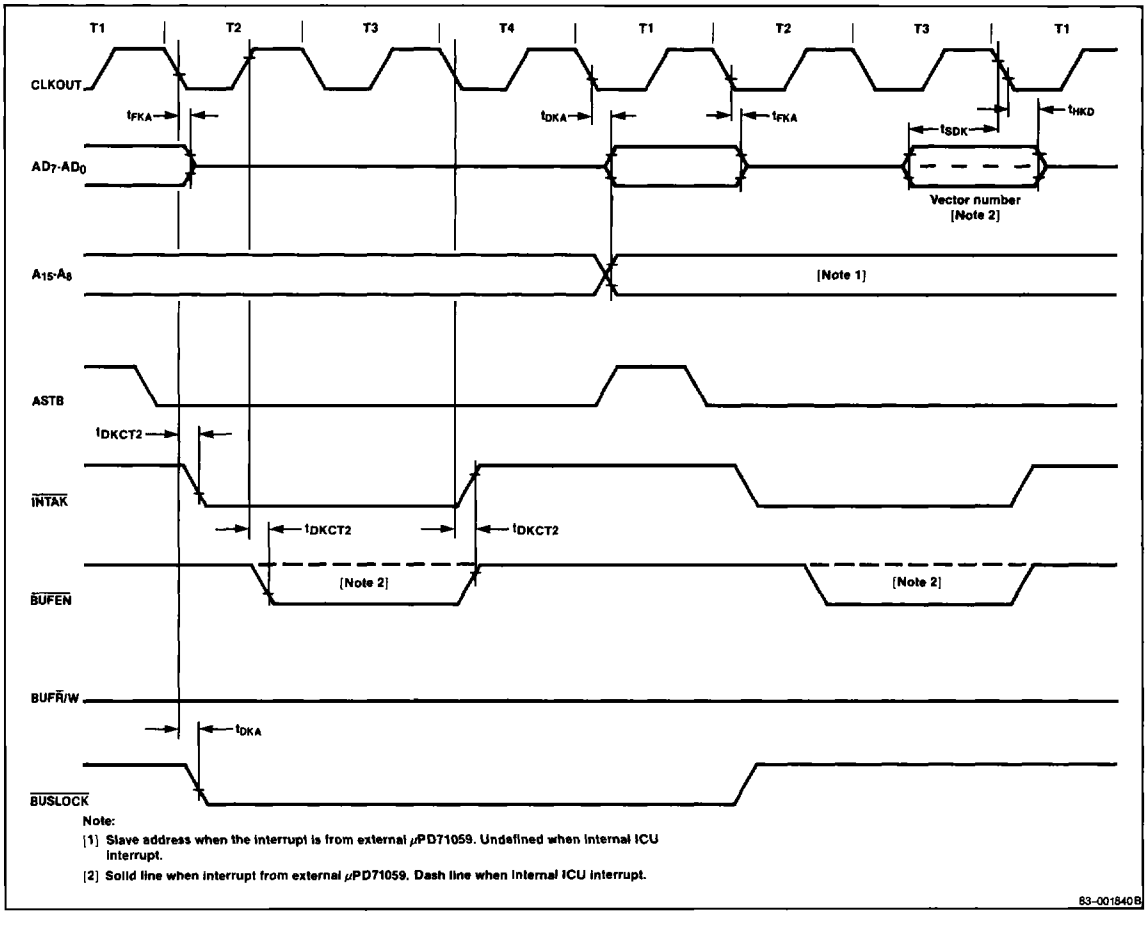
## Timing Waveforms (cont)

### *Status Timing*



## Timing Waveforms (cont)

### *Interrupt Acknowledge Timing*




Note:

[1] Slave address when the interrupt is from external μPD71059. Undefined when internal ICU interrupt.

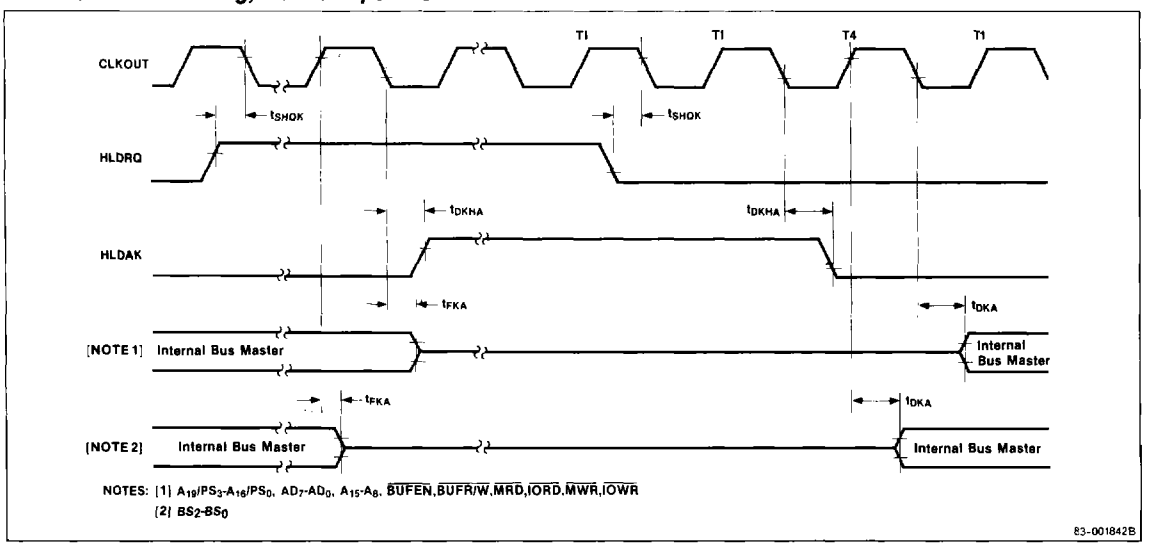
[2] Solid line when interrupt from external μPD71059. Dash line when internal ICU interrupt.

83-001840B

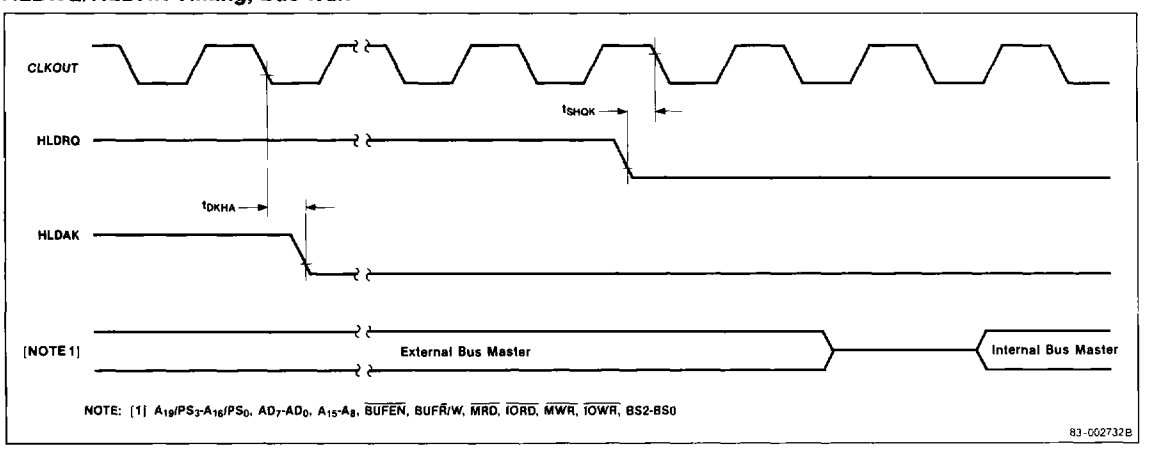
3c

## Timing Waveforms (cont)

### HLDRQ/HLDAK Timing, Normal Operation

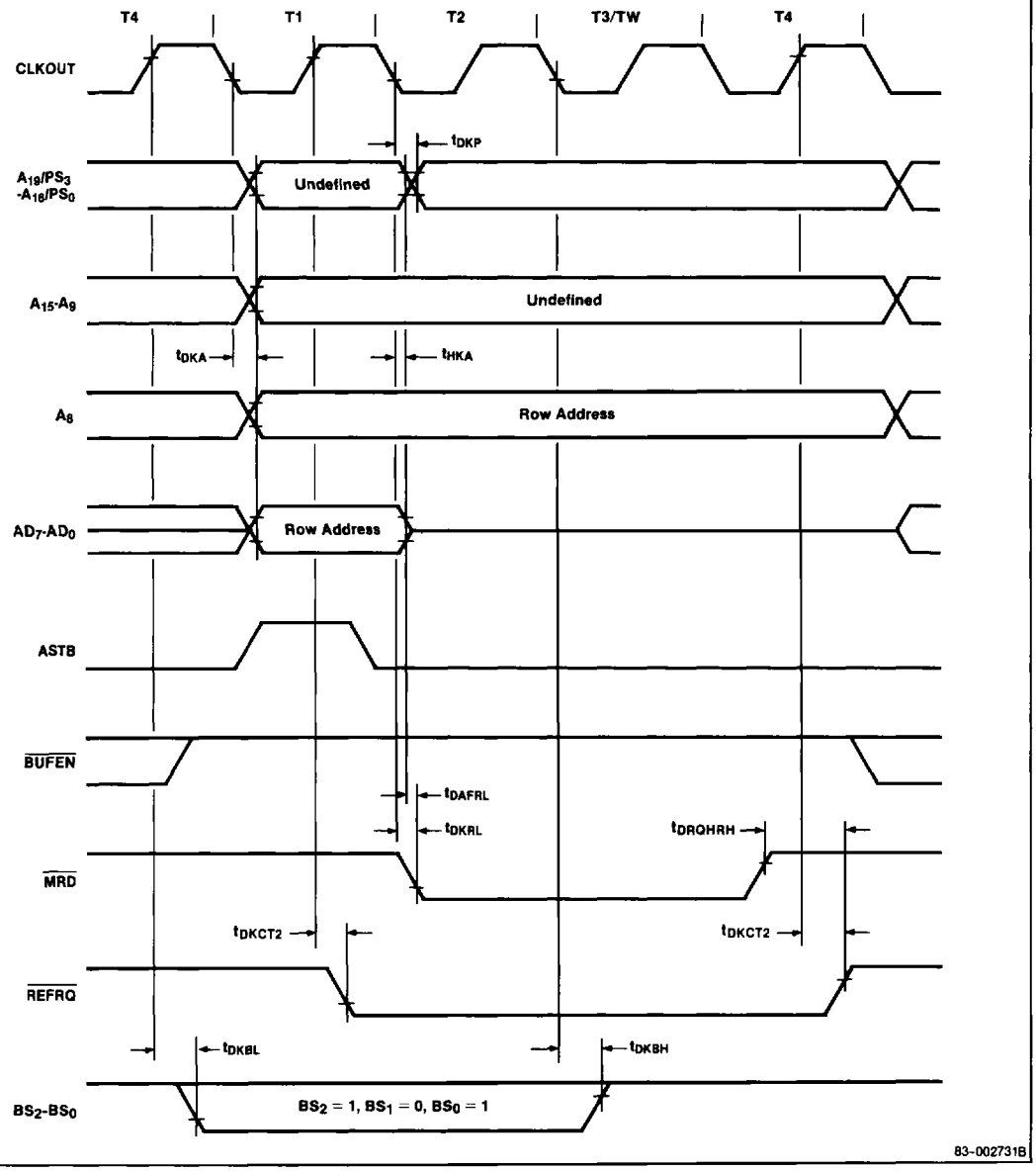



### HLDRQ/HLDAK Timing, Bus Wait



## Timing Waveforms (cont)

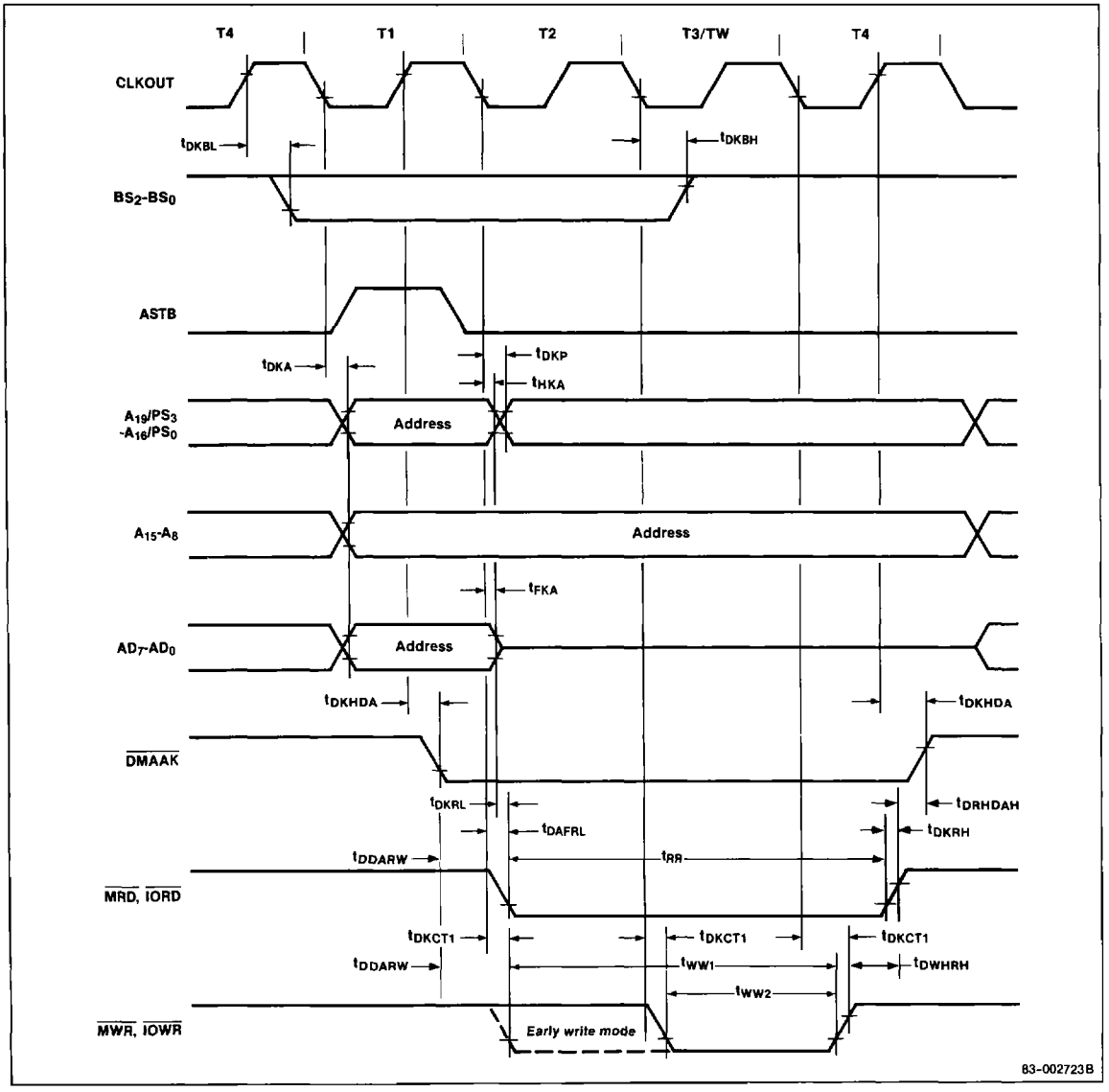
*Refresh Timing*




3c

## Timing Waveforms (cont)

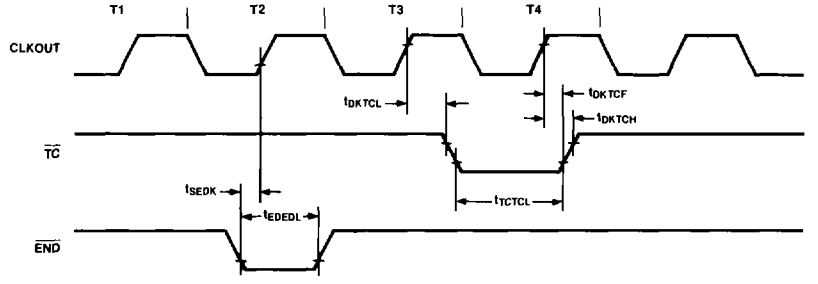
### *DMAU, DMA Transfer Timing*



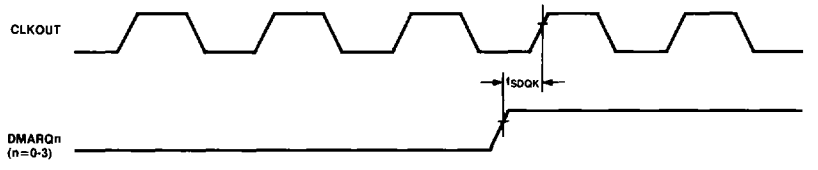
## Timing Waveforms (cont)

### *DMA Timing*

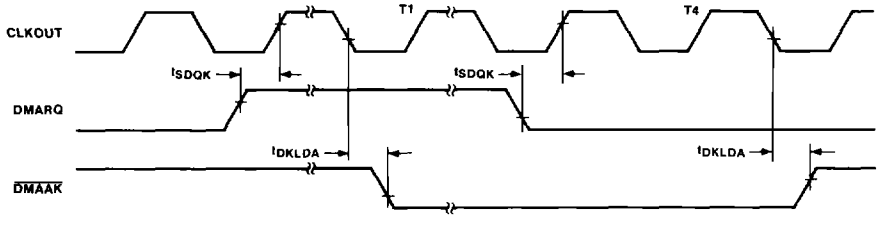
**$\overline{END}$/$\overline{TC}$ Timing**



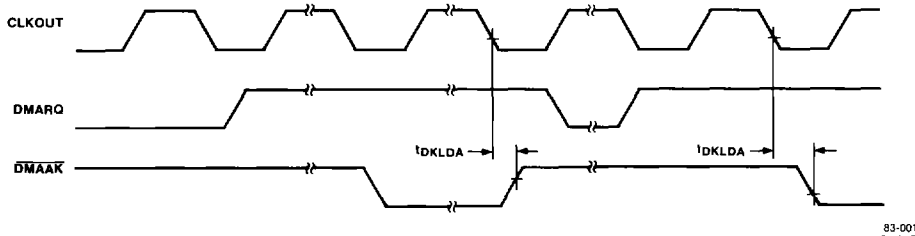

**DMA Request Timing**




**Cascade Mode, Normal Operation**




**Cascade Mode, Refresh Cycle Insertion**

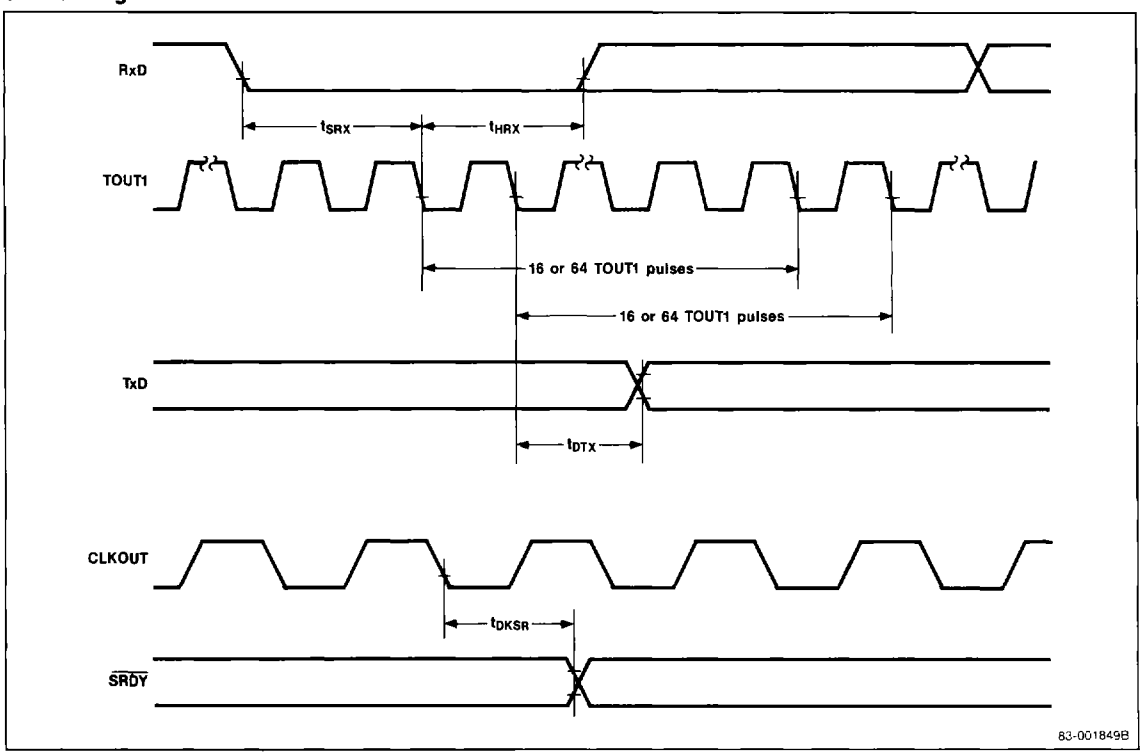



83-001826C

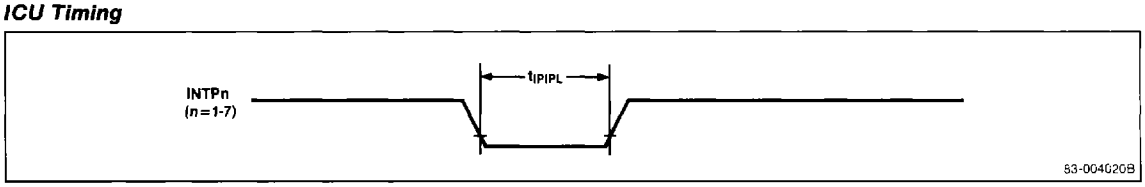
3c

## Timing Waveforms (cont)

### *SCU Timing*

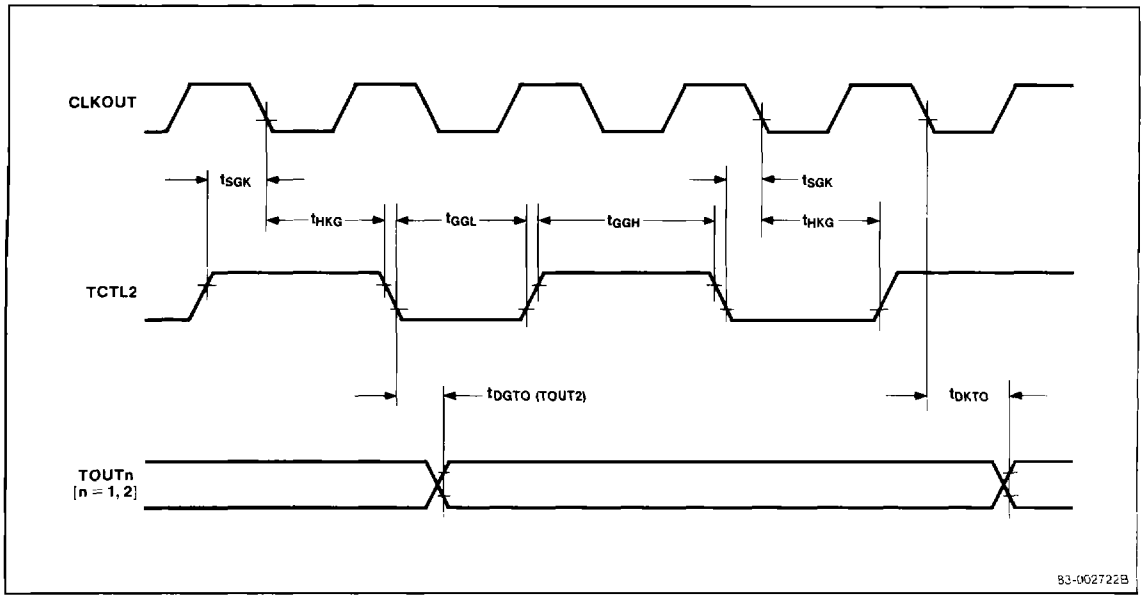



### *ICU Timing*

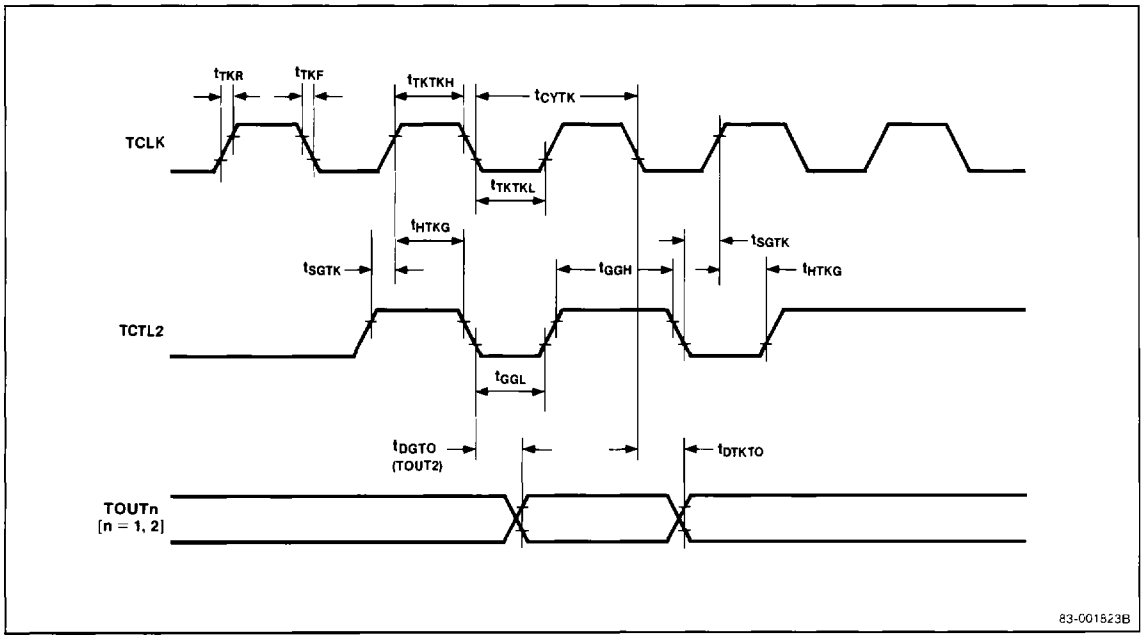


## Timing Waveforms (cont)

### *TCU Timing, Internal Clock Source*




### *TCU Timing, TCLK Source*



## Functional Description

Refer to the μPD70208 block diagram for an overview of the ten major functional blocks listed below.

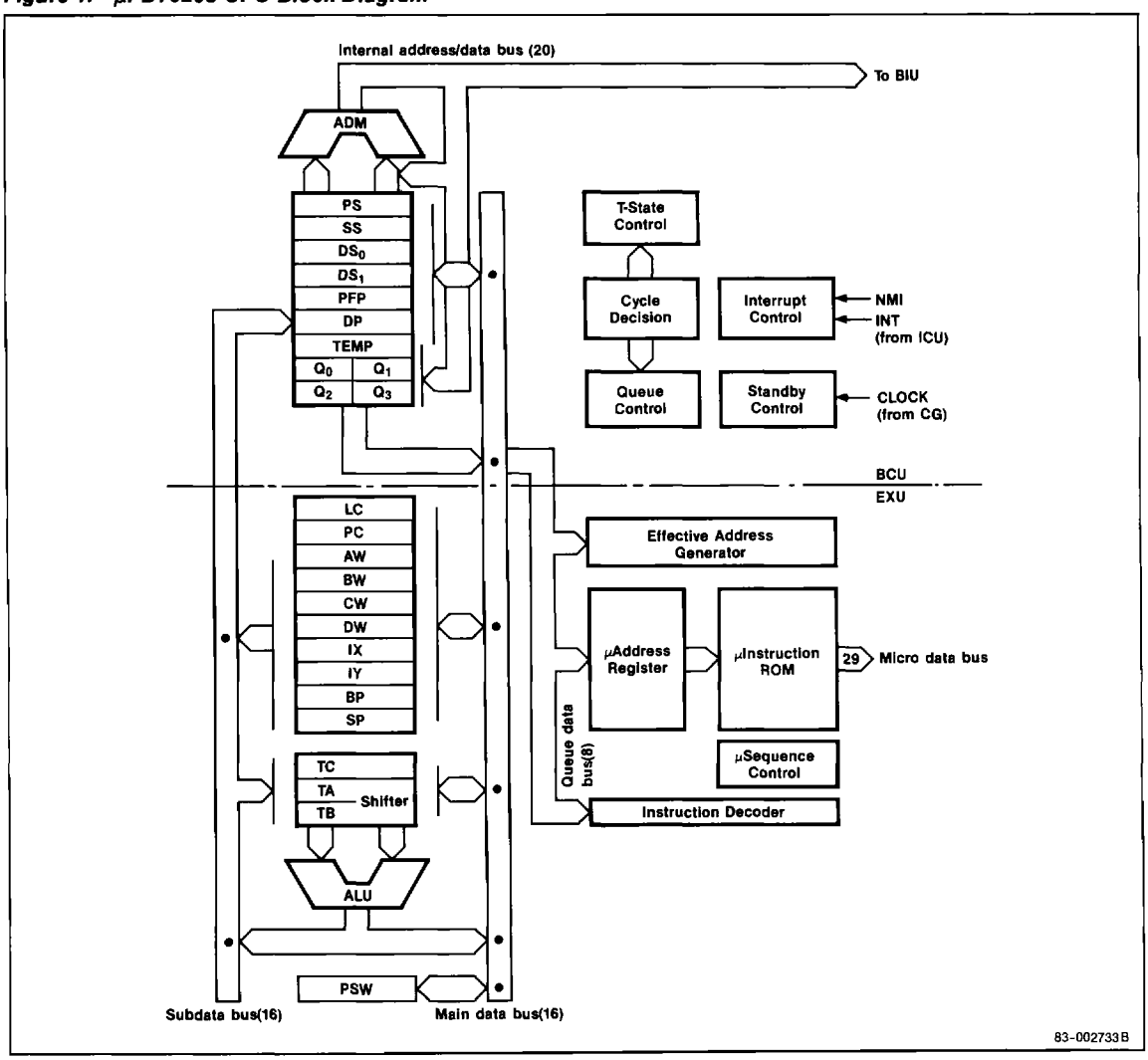
- Central processing unit (CPU)
- Clock generator (CG)
- Bus interface unit (BIU)
- Bus arbitration unit (BAU)
- Refresh control unit (RCU)
- Wait control unit (WCU)
- Timer/counter unit (TCU)
- Serial control unit (SCU)
- Interrupt control unit (ICU)
- DMA control unit (DMAU)

## Central Processing Unit

The μPD70208 CPU functions similarly to the CPU of the μPD70108 CMOS microprocessor. However, because the μPD70208 has internal peripheral devices, its bus architecture has been modified to permit sharing the bus with internal peripherals. The μPD70208 CPU is object code compatible with both the μPD70108/μPD70116 and the μPD8086/μPD8088 microprocessors.

Figure 1 is the μPD70208 CPU block diagram. A listing of the μPD70208 instruction set is in the final sections of this data sheet.

***Figure 1. μPD70208 CPU Block Diagram***



## Register Configuration

**Program Counter [PC].** The program counter is a 16-bit binary counter that contains the program segment offset of the next instruction to be executed. The PC is incremented each time the microprogram fetches an instruction from the instruction queue. The contents of the PC are replaced whenever a branch, call, return, or break instruction is executed and during interrupt processing. At this time, the contents of the PC are the same as the prefetch pointer (PFP).

**Prefetch Pointer [PFP].** The prefetch pointer is a 16-bit binary counter that contains the program segment offset of the next instruction to be fetched for the instruction queue. Because instruction queue prefetch is independent of instruction execution, the contents of the PFP and PC are not always identical. The PFP is updated each time the bus interface unit (BIU) fetches an instruction for the instruction queue. The contents of the PFP are replaced whenever a branch, call, return or break instruction is executed and during interrupt processing. At this time, the contents of the PFP and PC are the same.

**Segment Registers [PS, SS, $DS_0$, $DS_1$].** The μPD70216 memory address space is divided into 64K-byte logical segments. A memory address is determined by the sum of a 20-bit base address (obtained from a segment register) and a 16-bit offset known as the effective address (EA). I/O address space is not segmented and no segment register is used. The four segment registers are program segment (PS), stack segment (SS), data segment 0 ($DS_0$), and data segment 1 ($DS_1$). The following table lists their offsets and overrides.

| Default Segment Register | Offset | Override |
|---|---|---|
| PS | PFP register | None |
| SS | SP register | None |
| SS | Effective address (BP-based) | PS, $DS_0$, $DS_1$ |
| $DS_0$ | Effective address (non BP-based) | PS, SS, $DS_1$ |
| $DS_0$ | IX register (1) | PS, SS, $DS_1$ |
| $DS_1$ | IY register (2) | None |

**Note:**

(1) Includes source block transfer, output, BCD string, and bit field extraction.

(2) Includes destination block transfer, input, BCD string, and bit field insertion.

**General-Purpose Registers.** The μPD70208 CPU contains four 16-bit, general-purpose registers (AW, BW, CW, DW), each of which can be used as a pair of 8-bit registers by dividing into upper and lower bytes (AH, AL, BH, BL, CH, CL, DH, DL). General-purpose registers may also be specified implicitly in an instruction. The implicit assignments are:

- AW — Word multiplication/division, word I/O, data conversion
- AL — Byte multiplication/division, byte I/O, BCD rotation, data conversion, translation
- AH — Byte multiplication/division
- BW — Translation
- CW — Loop control, repeat prefix
- CL — Shift/rotate bit counts, BCD operations
- DW — Word multiplication/division, indirect I/O addressing

**Pointer [SP, BP] and Index Registers [IX, IY].** These registers serve as base pointers or index registers when accessing memory using one of the base, indexed, or base indexed addressing modes. Pointer and index registers can also be used as operands for word data transfer, arithmetic, and logical instructions. These registers are implicitly selected by certain instructions as follows.

- SP — Stack operations, interrupts
- IX — Source block transfer, BCD string operations, bit field extraction
- IY — Destination block transfer, BCD string operations, bit field insertion

## Program Status Word [PSW]

The program status word consists of six status flags and four control flags.

**Status Flags**

- V (Overflow)
- S (Sign)
- Z (Zero)
- AC (Auxiliary Carry)
- P (Parity)
- CY (Carry)

**Control Flags**

- MD (Mode)
- DIR (Direction)
- IE (Interrupt Enable)
- BRK (Break)

3c

When pushed onto the stack, the word image of the PSW is as follows:

| 15 | | | | | | | 8 |
|---|---|---|---|---|---|---|---|
| MD | 1 | 1 | 1 | V | DIR | IE | BRK |

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| S | Z | 0 | AC | 0 | P | 1 | CY |

The status flags are set and cleared automatically depending upon the result of the previous instruction execution. Instructions are provided to set, clear, and complement certain status and control flags. Other flags can be manipulated by using the POP PSW instruction.

Between execution of the BRKEM and RETEM instructions, the native mode RETI and POP PSW instructions can modify the MD bit. Care must be exercised by emulation mode programs to prevent inadvertent alteration of this bit.

## CPU Architectural Features

The major architectural features of the μPD70208 CPU are:

- Dual data buses
- Effective address generator
- Loop counter
- PC and PFP

**Dual Data Buses.** To increase performance, dual data buses (figure 2) have been employed in the CPU to fetch operands in parallel and avoid the bottleneck of a single bus. For two-operand instructions and effective address calculations, the dual data bus approach is 30 percent faster than single-bus systems.

**Effective Address Generator.** Effective address (EA) calculation requires only two clocks regardless of the addressing mode complexity due to the hardware effective address generator (figure 3). When compared with microprogrammed methods, the hardware approach saves between 3 and 10 clock cycles during effective address calculation.

**Loop Counter and Shifters.** A dedicated loop counter is used to count the iterations of block transfer and multiple shift instructions. This logic offers a significant performance advantage over architectures that control block transfers and multiple shifts using microprogramming. Dedicated shift registers also speed up the execution of the multiply and divide instructions. Compared with microprogrammed methods, multiply and divide instructions execute approximately four times faster.

***Figure 2. Dual Data Buses***

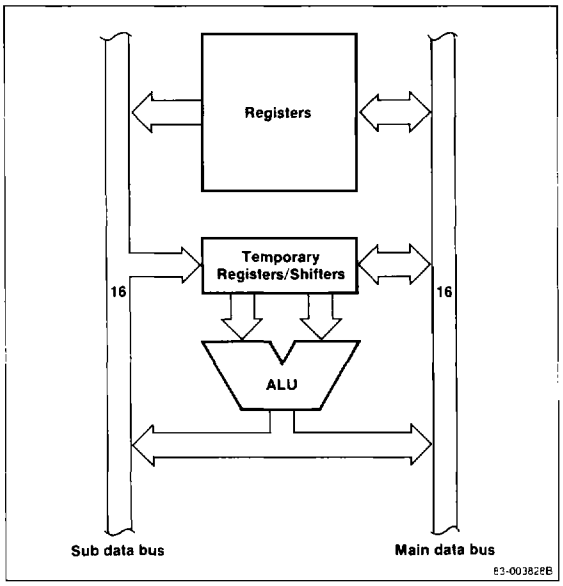
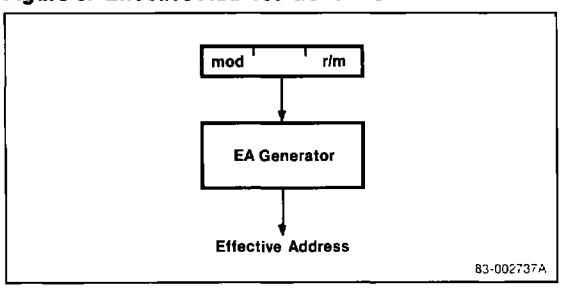



***Figure 3. Effective Address Generator***




**Program Counter and Prefetch Pointer.** The functions of instruction execution and queue prefetch are decoupled in the μPD70208. By avoiding a single instruction pointer and providing separate PC and PFP registers, the execution time of control transfers and the interrupt response latency can be minimized. Several clocks are saved by avoiding the need to readjust an instruction pointer to account for prefetching before computing the new destination address.

## Enhanced Instruction Set

In addition to the μPD8086/88 instruction set, the μPD70208 has added the following enhanced instructions.

| Instruction | Function |
|---|---|
| PUSH imm | Push immediate data onto stack |
| PUSH R | Push all general registers onto stack |
| POP R | Pop all general registers from stack |
| MUL imm | Multiply register/memory by immediate data |
| SHL imm8 | Shift/rotate by immediate count |
| SHR imm8 | |
| SHRA imm8 | |
| ROL imm8 | |
| ROR imm8 | |
| ROLC imm8 | |
| RORC imm8 | |
| CHKIND | Check array index |
| INM | Input multiple |
| OUTM | Output multiple |
| PREPARE | Prepare new stack frame |
| DISPOSE | Dispose current stack frame |

## Unique Instruction Set

In addition to the μPD70208 enhanced instruction set, the following unique instructions are supported.

| Instruction | Function |
|---|---|
| INS | Insert bit field |
| EXT | Extract bit field |
| ADD4S | BCD string addition |
| SUB4S | BCD string subtraction |
| CMP4S | BCD string comparison |
| ROL4 | Rotate BCD digit left |
| ROR4 | Rotate BCD digit right |
| TEST1 | Test bit |
| SET1 | Set bit |
| CLR1 | Clear bit |
| NOT1 | Complement bit |
| REPC | Repeat while carry set |
| REPNC | Repeat while carry cleared |
| FPO2 | Floating point operation 2 |

**Bit Fields.** Bit fields are data structures that range in length from 1 to 16 bits. Two separate operations on bit fields, insertion and extraction, with no restrictions on the position of the bit field in memory are supported. Separate segment, byte offset, and bit offset registers are used for bit field insertion and extraction. Because of their power and flexibility, these instructions are highly effective for graphics, high-level languages, and data packing/unpacking applications.

Insert bit field (INS) copies the bit field of specified length (0 = 1 bit, 15 = 16 bits) from the AW register to the bit field addressed by DS1:IY:reg8 (figure 4). The bit field length can be located in any byte register or supplied as an immediate value. The value in reg8 is a bit field offset. A content of 0 selects bit 0 and 15 selects bit 15 of the word that DS0:IX points to. Following execution, the IY and bit offset register are updated to point to the start of the next bit field.

Bit field extraction (EXT) copies the bit field of specified length (0 = 1 bit, 15 = 16 bits) from the bit field addressed by DS0:IX:reg8 to the AW register (figure 5). If the bit field is less than 16 bits, it is right justified with a zero fill. The bit field length can be located in any byte register or supplied as immediate data. The value in reg8 is a bit field offset. A content of 0 selects bit 0 and 15 selects bit 15 of the word that DS0:IX points to. Following execution, the IX and bit offset register are updated to point to the start of the next bit field.

**Packed BCD Strings.** These instructions are provided to efficiently manipulate packed BCD data as strings (length from 1 to 254 digits) or as a byte data type with a single instruction.



BCD string arithmetic is supported by the ADD4S, SUB4S, and CMP4S instructions. These instructions allow the source string (addressed by DS0:IX) and the destination string (addressed by DS1:IY) to be manipulated with a single instruction. When the number of BCD digits is even, the Z and CY flags are set according to the result of the operation. If the number of digits is odd, the Z flag will not be correctly set unless the upper 4 bits of the result are zero. The CY flag will not be correctly set unless there is a carry out of the upper 4 bits of the result.

The two BCD rotate instructions (ROR4, ROL4) perform rotation of a single BCD digit in the lower half of the AL register through the register or memory operand.

**Bit Manipulation.** Four bit manipulation instructions have been added to the μPD70208 instruction set. The ability to test, set, clear, or complement a single bit in a register or memory operand increases code readability as well as performance over the logical operations traditionally used to manipulate bit data.

**Repeat Prefixes.** Two repeat prefixes (REPC, REPNC) allow conditional block transfer instructions to use the state of the CY flag as a terminating condition. The use of these prefixes allows inequalities to be used when working on ordered data, increasing the performance of searching and sorting algorithms.

**Floating Point Operation Instructions.** Two floating point operation (FPO) instruction types are recognized by the μPD70208 CPU. These instructions are detected by the CPU, which performs any auxiliary processing such as effective address calculation and the initial bus cycle if specified by the instruction. It is the responsibility of the external coprocessor to latch the address information and data (if a read cycle) from the bus and complete the execution of the instruction.

*Figure 4. Bit Field Insertion*

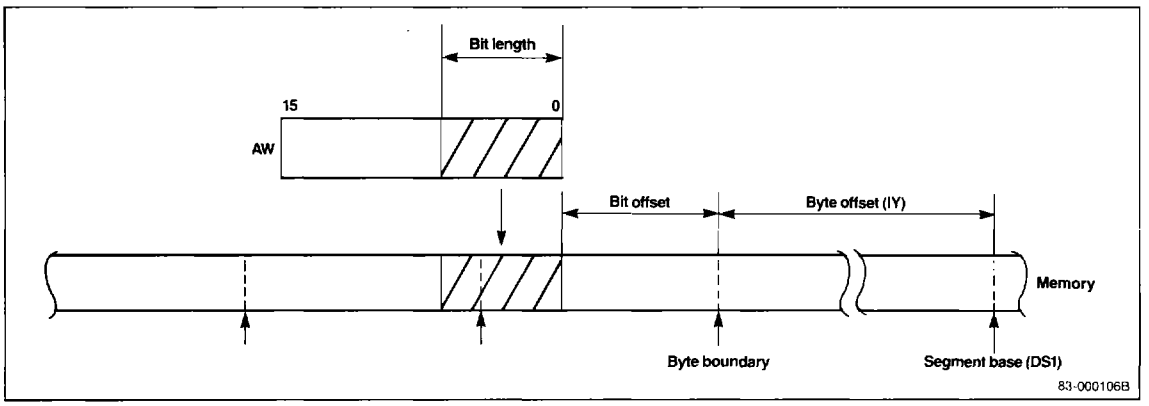
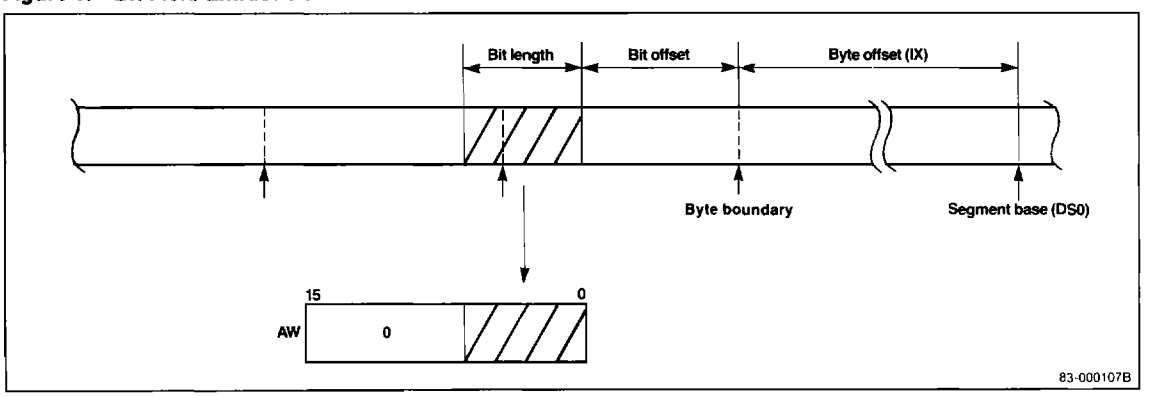



*Figure 5. Bit Field Extraction*




**8080 Emulation Mode.** The μPD70208 CPU can operate in either of two modes; see figure 6. Native mode allows the execution of the μPD8086/88, enhanced and unique instructions. The other operating mode is 8080 emulation mode, which allows the entire μPD8080AF instruction set to be executed. A mode (MD) flag is provided to distinguish between the two operating modes. Native mode is active when MD is 1 and 8080 emulation mode is active when MD is 0.

Two instructions are provided to switch from native to 8080 emulation mode and return back. Break for emulation (BRKEM) operates similarly to a BRK instruction, except that after the PSW has been pushed on the native mode stack, the MD flag becomes write-enabled and is cleared.

During 8080 emulation mode, the registers and flags of the 8080 are mapped onto the native mode registers and flags as shown below. Note that PS, SS, $DS_0$, $DS_1$, IX, IY, AH, and the upper half of the PSW registers are inaccessible to 8080 programs.

| | μPD8080AF | μPD70208 |
|---|---|---|
| Registers | A/PSW | AL/PSW (lower) |
| | B | CH |
| | C | CL |
| | D | DH |
| | E | DL |
| | H | BH |
| | L | BL |
| | SP | BP |
| | PC | PC |
| Flags | C | CY |
| | Z | Z |
| | S | S |
| | P | P |
| | AC | AC |

During 8080 emulation mode, the BP register functions as the 8080 stack pointer. The use of separate stack pointers prevents inadvertent damage to the native mode stack pointer by emulation mode programs.

The 8080 emulation mode PC is combined with the PS register to form the 20-bit physical address. All emulation mode data references use DS0 as the segment register. For compatibility with older 8080 software these registers must be equal. By using different segment register contents, separate 64K-byte code and data spaces are possible.

Either an NMI or maskable interrupt will cause the 8080 emulation mode to be suspended. The CPU pushes the PS, PC, and PSW registers on the native mode stack, sets the MD bit (indicating native mode), and enters the specified interrupt handler. When the return from interrupt (RETI) instruction is executed, the PS, PC, and PSW (containing MD=0) are popped from the native stack and execution in 8080 emulation mode continues. Reset will also force a return to native mode.

The 8080 emulation mode programs also have the capability to invoke native mode interrupt handlers by means of the call native (CALLN) instruction. This instruction operates like the BRK instruction except that the saved PSW indicates 8080 emulation mode.

To exit 8080 emulation mode, the return from emulation (RETEM) instruction pops the PS, PC, and PSW from the native mode stack, disables modification of the MD bit, and execution continues with the instruction following the BRKEM instruction. Nesting of 8080 emulation modes is prohibited.

### Interrupt Operation

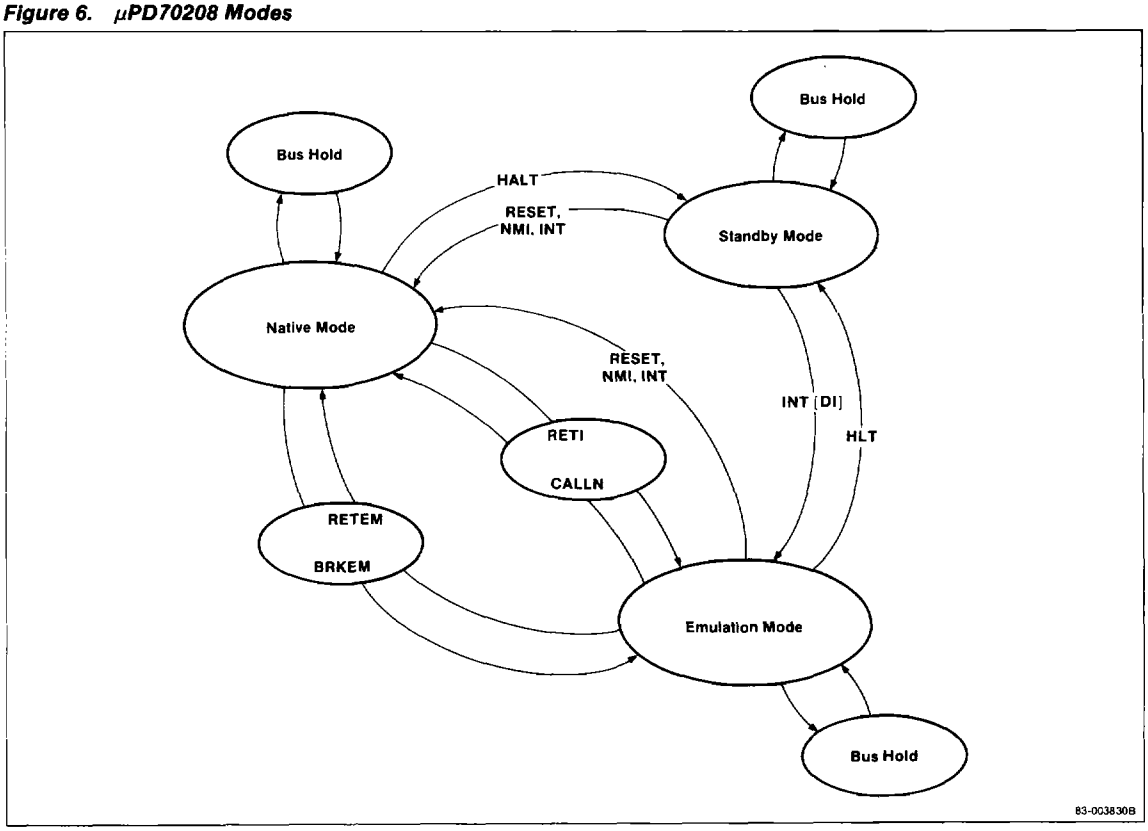
The μPD70208 supports a number of external interrupts and software exceptions. External interrupts are events asynchronous to program execution. On the other hand, exceptions always occur as a result of program execution.

The two types of external interrupts are:

- Nonmaskable interrupt (NMI)
- Maskable interrupt (INT)

3c

***Figure 6. μPD70208 Modes***



The six software exceptions are:

- Divide error (DIV, DIVU instructions)
- Array bound error (CHKIND instruction)
- Break on overflow (BRKV instruction)
- Break (BRK, BRK3 instructions)
- Single step (BRK bit in PSW set)
- Mode switch (BRKEM, CALLN instructions)

Interrupt vectors are determined automatically for exceptions and the NMI interrupt or supplied by hardware for maskable interrupts. The 256 interrupt vectors are stored in a table (figure 7) located at address 00000H. Vectors 0 to 5 are predetermined and vectors 6 to 31 are reserved. Interrupt vectors 32 to 255 are available for use by application software.

Each vector is made up of two words. The word located at the lower address contains the new PC for the interrupt handler. The word at the next-higher address is the new PS value for the interrupt handler. These must be initialized by software at the start of a program.

Nonmaskable interrupts and maskable interrupts (when enabled) are normally serviced following the execution of the current instruction. However, the following cases are exceptions to this rule and the occurrence of the interrupt will be delayed until after the execution of the next instruction.

- Moves to/from segment registers
- POLL instruction
- Instruction prefixes
- EI instruction (maskable interrupts only)

Another special case is the block transfer instructions. These instructions are interruptable and resumable, but because of the asynchronous operation of the BIU, the actual occurrence of the interrupt may be delayed up to three bus cycles.

## Standby Mode

The μPD70208 CPU has a low-power standby mode, which can dramatically reduce power consumption during idle periods. Standby mode is entered by simply executing a native or 8080 emulation HALT instruction; no external hardware is required. All other peripherals such as the timer/counter unit, refresh control unit, and DMA control unit continue to operate as programmed.

During standby mode, the clock is distributed only to the circuits required to release the standby mode. When a RESET, NMI, or INT event is detected, the standby mode is released. Both NMI and unmaskable interrupts are processed before control returns to the instruction following the HALT. In the case of the INT input being masked, execution will begin with the instruction immediately following the HALT instruction without an intervening interrupt acknowledge bus cycle. When maskable interrupts are again enabled, the interrupt will be serviced.

Output signal states in the standby mode are listed below.

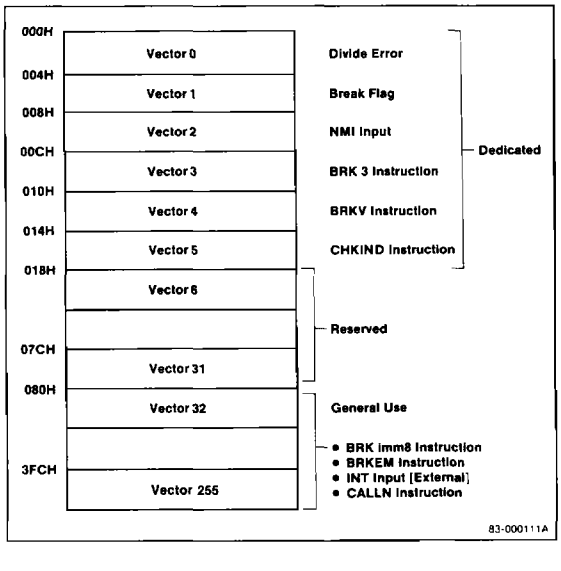
| Output Signal | Status in Standby Mode |
|---|---|
| $\overline{INTAK}$, $\overline{BUFEN}$, $\overline{MRD}$, $\overline{MWR}$, $\overline{IOWR}$, $\overline{IORD}$ | High level |
| $BS_2$-$BS_0$ (Note 2) | High level |
| $QS_1$-$QS_0$, ASTB | Low level |
| $\overline{BUSLOCK}$ | High level (low level if the HALT instruction follows the BUSLOCK prefix) |
| $\overline{BUFR}$/W, $A_{19}$-$A_{16}$/$PS_3$-$PS_0$, $A_{15}$-$A_8$, $AD_7$-$AD_0$ | High or low level |

**Note:**

(1) Output pin states during refresh and DMA bus cycles will be as defined for those operations.

(2) Halt status is presented prior to entering the passive state.

***Figure 7. Interrupt Vector Table***



## Clock Generator

The clock generator (CG) generates a clock signal half the frequency of a parallel-resonant, fundamental mode crystal connected to pins X1 and X2. Figure 8 shows the recommended circuit configuration. Capacitors C1 and C2, required for frequency stability, are selected to match the crystal load capacitance.

External clock sources are also accommodated as shown in figure 9. The CG distributes the clock to the CLKOUT pin and to each functional block of the μPD70208. The generated clock signal has a 50-percent duty cycle.

## Bus Interface Unit

The bus interface unit (BIU) controls the external address, data, and control buses for the three internal bus masters: CPU, DMA control unit (DMAU), and refresh control unit (RCU). The BIU is also responsible for synchronization of the $\overline{\text{RESET}}$ and READY inputs with the clock. The synchronized reset signal is used internally by the μPD70208 and provided externally at the RESOUT pin as a system-wide reset. The synchronized READY signal is combined with the output of the wait control unit (WCU) and is distributed internally to the CPU, DMAU, and RCU. Figure 10 shows the synchronization of $\overline{\text{RESET}}$ and READY.

The BIU also has the capability of overlapping the execution of the next instruction with memory write bus cycles. There is no overlap of instruction execution with read or I/O write bus cycles.

*Figure 9. External Oscillator Configuration*

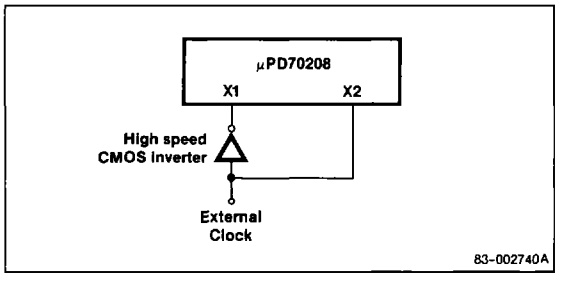
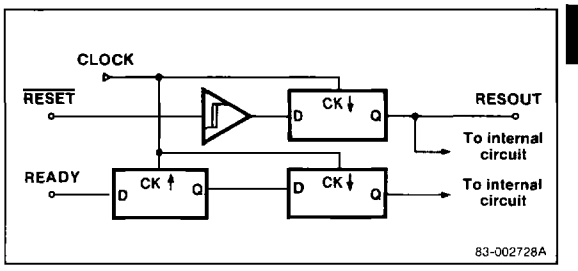
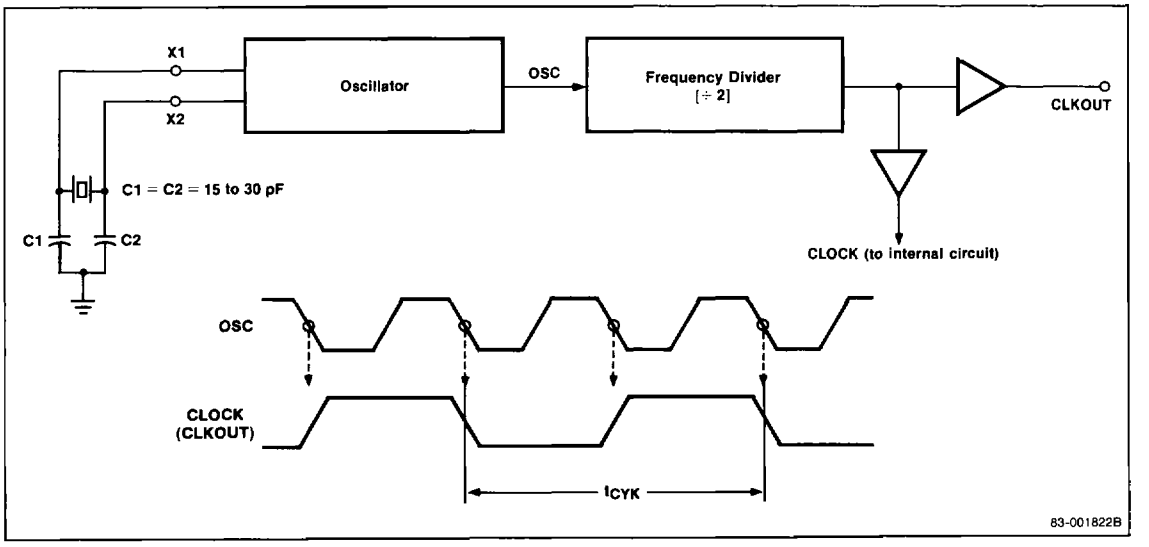



*Figure 10. RESET/READY Synchronization*




3c

*Figure 8. Crystal Configuration*



## Bus Arbitration Unit

The bus arbitration unit (BAU) arbitrates the external address, data and control buses between the internal CPU, DMAU, and RCU bus requesters and an external bus master. The BAU bus priorities from the highest priority requester to the lowest are:

RCU (Demand mode)
DMAU
HLDRQ
CPU
RCU (Normal mode)

Note that RCU requests the bus at either the highest or lowest priority depending on the status of the refresh request queue. Bus masters other than the CPU are prohibited from using the bus when the CPU is executing an instruction containing a BUSLOCK prefix. Therefore, caution should be exercised when using the BUSLOCK prefix with instructions having a long execution time.

If a bus master with higher priority than the current bus master requests the bus, the BAU inactivates the current bus master's acknowledge signal. When the BAU sees the bus request from the current master go inactive, the BAU gives control of the bus to the higher priority bus master. Whenever possible, the BAU performs bus switching between internal bus masters without the introduction of idle bus cycles, enhancing system throughput.

## System I/O Area

The I/O address space from addresses FF00H to FFFFH is reserved for use as the system I/O area. Located in this area are the 12 μPD70208 registers that determine the I/O addressing, enable/disable peripherals, and control pin multiplexing. Byte I/O instructions must be used to access the system I/O area.

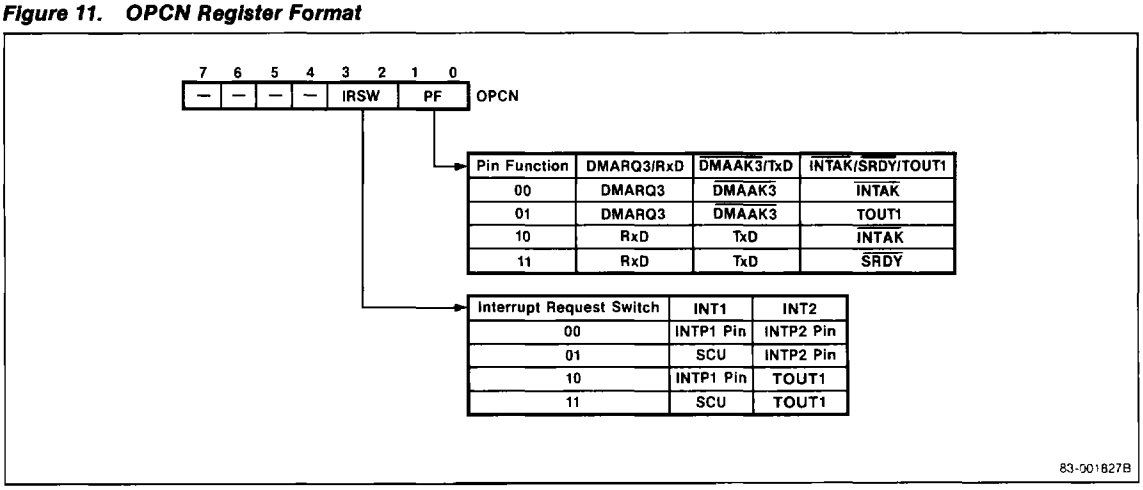
| I/O Address | Register | Operation |
|---|---|---|
| FFFFH | Reserved | — |
| FFFEH | OPCN | Read/Write |
| FFFDH | OPSEL | Read/Write |
| FFFCH | OPHA | Read/Write |
| FFFBH | DULA | Read/Write |
| FFFAH | IULA | Read/Write |
| FFF9H | TULA | Read/Write |
| FFF8H | SULA | Read/Write |
| FFF7H | Reserved | — |
| FFF6H | WCY2 | Read/Write |
| FFF5H | WCY1 | Read/Write |
| FFF4H | WMB | Read/Write |
| FFF3H | Reserved | — |
| FFF2H | RFC | Read/Write |
| FFF1H | Reserved | — |
| FFF0H | TCKS | Read/Write |

## On-Chip Peripheral Connection Register

The on-chip peripheral connection (OPCN) register controls multiplexing of the μPD70208 multiplexed pins. Figure 11 shows the format of the OPCN register. The interrupt request switch (IRSW) field controls multiplexing of ICU interrupt inputs INT1 and INT2. The output of an internal peripheral or an external interrupt source can be selected as the INT1 and INT2 inputs to the ICU.

The pin function (PF) field in the OPCN selects one of four possible states for the DMARQ3/RxD, DMAAK3/TxD, and INTAK/TOUT1/SRDY pins. Bit 0 of the

*Figure 11. OPCN Register Format*

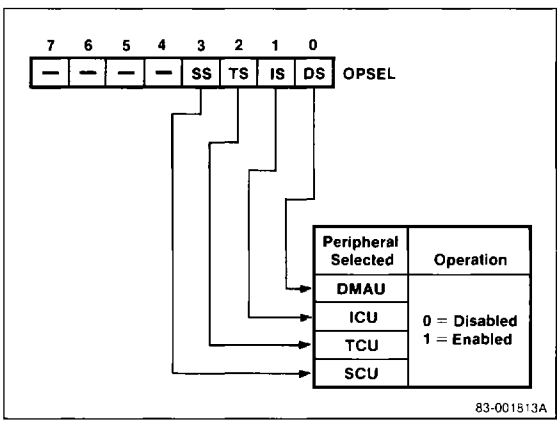


OPCN controls the function of the INTAK/TOUT1/SRDY pin. If cleared, INTAK will appear on this output pin. If bit 0 is set, either TOUT1 or SRDY will appear at the output depending on the state of bit 1. If bit 1 is cleared, DMA channel 3 I/O signals will appear on the DMARQ3/RxD and DMAAK3/TxD pins. If the SCU is to be used, bit 1 of the PF field must be set.

### On-Chip Peripheral Selection Register

The on-chip peripheral selection (OPSEL) register is used to enable or disable the μPD70208 internal peripherals. Figure 12 shows the format of the OPSEL register. Any of the four (DMAU, TCU, ICU, SCU) peripherals can be independently enabled or disabled by setting or clearing the appropriate OPSEL bit.

***Figure 12. OPSEL Register Format***




### Internal Peripheral Relocation Registers

The five internal peripheral relocation registers (figure 13) are used to fix the I/O addresses of the DMAU, ICU, TCU, and SCU. The on-chip peripheral high-address (OPHA) register is common to all four internal peripherals and fixes the high-order byte of the 16-bit I/O address. The individual DMAU low-address (DULA) register, ICU low-address (IULA) register, TCU low-address (TULA) register, and the SCU low-address (SULA) register select the low-order byte of the I/O addresses for the DMAU, ICU, TCU, and SCU peripherals.

The contents of the OPHA register are:

OPHA

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| $A_{15}$ | $A_{14}$ | $A_{13}$ | $A_{12}$ | $A_{11}$ | $A_{10}$ | $A_9$ | $A_8$ |

The formats for the individual internal peripheral registers appear below. Since address checking is not performed, do not overlap two peripheral I/O address spaces.

DULA

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| $A_7$ | $A_6$ | $A_5$ | $A_4$ | — | — | — | — |

IULA

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| $A_7$ | $A_6$ | $A_5$ | $A_4$ | $A_3$ | $A_2$ | — | — |

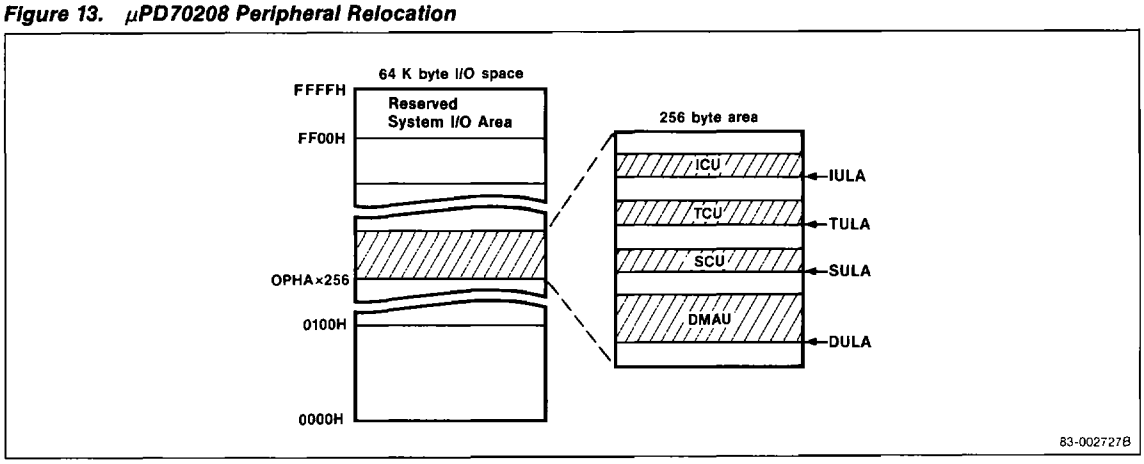
TULA

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| $A_7$ | $A_6$ | $A_5$ | $A_4$ | $A_3$ | $A_2$ | — | — |

SULA

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| $A_7$ | $A_6$ | $A_5$ | $A_4$ | $A_3$ | $A_2$ | — | — |

***Figure 13. μPD70208 Peripheral Relocation***



### Timer Clock Selection Register

The timer clock selection (TCKS) register selects the clock source for each of the timer/counters as well as the divisor for the internal clock prescaler. Figure 14 shows the format of the TCKS register. The clock source for each timer/counter is independently selected from either the prescaled internal CPU clock or from an external clock source (TCLK). The internal clock is derived from the CLKOUT signal and can be divided by 2, 4, 8, or 16 before being presented to the clock select logic.

## Refresh Control Unit

The refresh control unit (RCU) refreshes external dynamic RAM devices by outputting a 9-bit row address on address lines $A_8$-$A_0$ and performing a memory read bus cycle. External logic can distinguish a refresh bus cycle by monitoring the refresh request ($\overline{REFRQ}$) pin. Following each refresh bus cycle, the refresh row counter is incremented.

The refresh control (RFC) register in the system I/O area contains two fields. The refresh enable field enables or disables the refreshing function. The refresh timer (RTM) field selects a refresh interval to match the dynamic memory refresh requirements. Figure 15 shows the format for the RFC register.

To minimize the impact of refresh on the system bus bandwidth, the μPD70208 utilizes a refresh request queue to store refresh requests and perform refresh bus cycles in otherwise idle bus cycles.

The RCU normally requests the bus as the lowest-priority bus requester (normal mode). However, if seven refresh requests are allowed to accumulate in the RCU refresh request queue, the RCU will change to the highest-priority bus requester (demand mode). The RCU will then perform back-to-back refresh cycles until three requests remain in the queue. This guarantees the integrity of the DRAM system while maximizing performance.

The refresh count interval can be calculated as follows:

Refresh interval = 8 x N x $t_{CYK}$

where N is the timer factor selected by the RTM field.

When the μPD70208 is reset, the RE field in the RFC register is unaffected and the RTM field is set to 01000 (N = 9). No refresh bus cycles occur while $\overline{RESET}$ is asserted.

***Figure 15. Refresh Control Register***

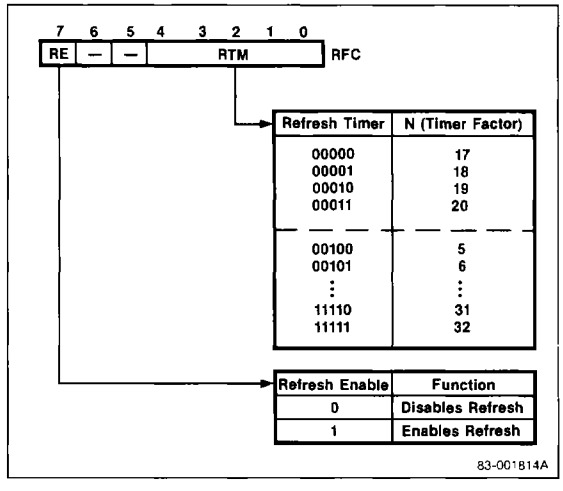
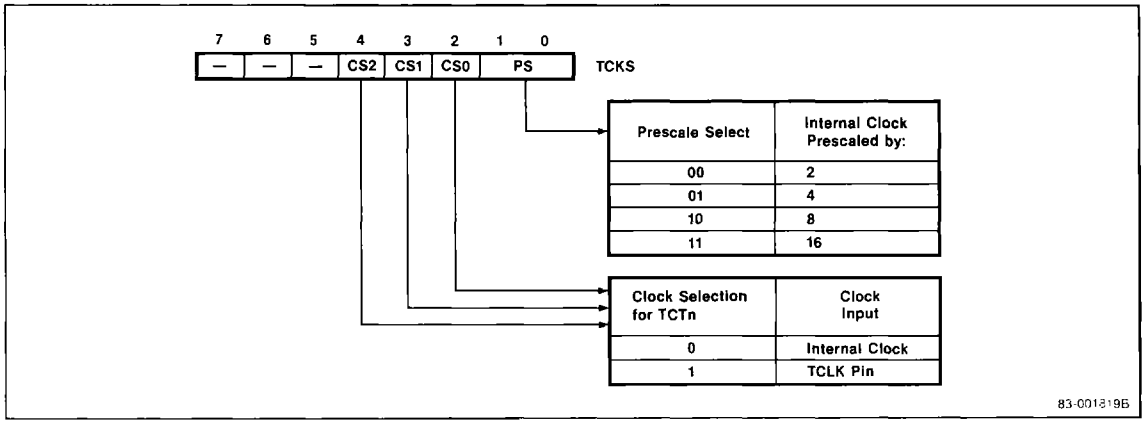



***Figure 14. Timer Clock Selection Register***



## Wait Control Unit

The wait control unit (WCU) inserts from zero to three wait states into a bus cycle in order to compensate for the varying access times of memory and I/O devices. The number of wait states for CPU, DMAU, and RCU bus cycles is separately programmable. In addition, the memory address space is divided into three independent partitions to accommodate a wide range of system designs. $\overline{\text{RESET}}$ initializes the WCU to insert three wait states in all bus cycles. This allows operation with slow memory and peripheral devices before the initialization of the WCU registers.

The three system I/O area registers that control the WCU are wait cycle 1 (WCY1), wait cycle 2 (WCY2), and wait state memory boundary (WMB). The WCU always inserts wait states corresponding to the wait count programmed in WCY1 or WCY2 registers into a bus cycle, regardless of the state of the external READY input. After the programmed number of wait states occurs, the WCU will insert Tw states as long as the READY pin remains inactive. When READY is again asserted, the bus cycle continues with T4 as the next cycle. The μPD70208 internal peripherals never require wait states; four clock cycles will terminate an internal peripheral bus cycle.

### CPU Wait States

The WMB register divides the 1M-byte memory address space into three independent partitions: lower, middle, and upper. Figure 16 shows the WMB register format.

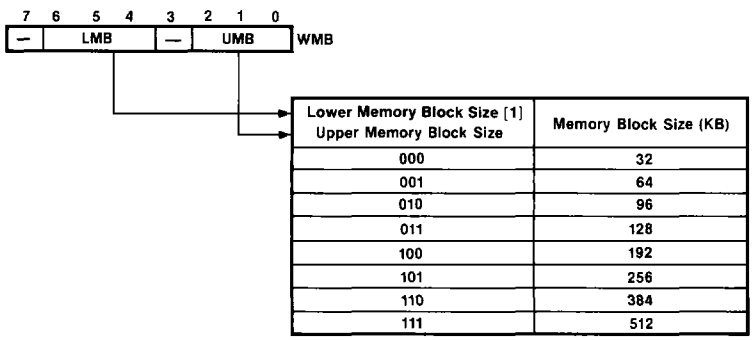
Initialization software can then set the number of wait states for each memory partition and the I/O partition via the WCY1 register (figure 17).

### DMA and Refresh Wait States

The WCY2 register (figure 18) specifies the number of wait states to be automatically inserted in DMA and refresh bus cycles. DMA wait states must be set to the maximum of the DMA memory and I/O partitions. Refresh wait states should be set to the maximum value of all DRAM memory partitions.

3c

***Figure 16. Wait State Memory Boundary Register***




Note:

[1] By default, the address space remaining between the UBM and LBM is the middle memory block.

*Figure 17. Wait Cycle 1 Register*

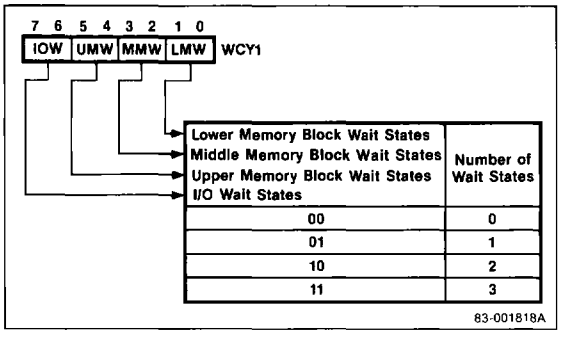
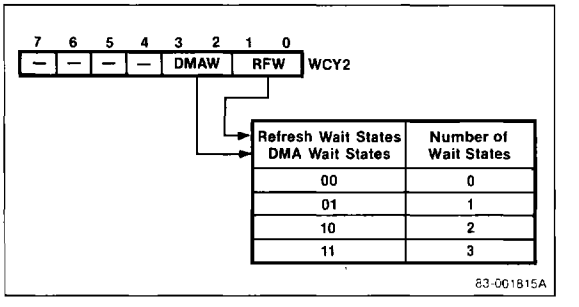



*Figure 18. Wait Cycle 2 Register*



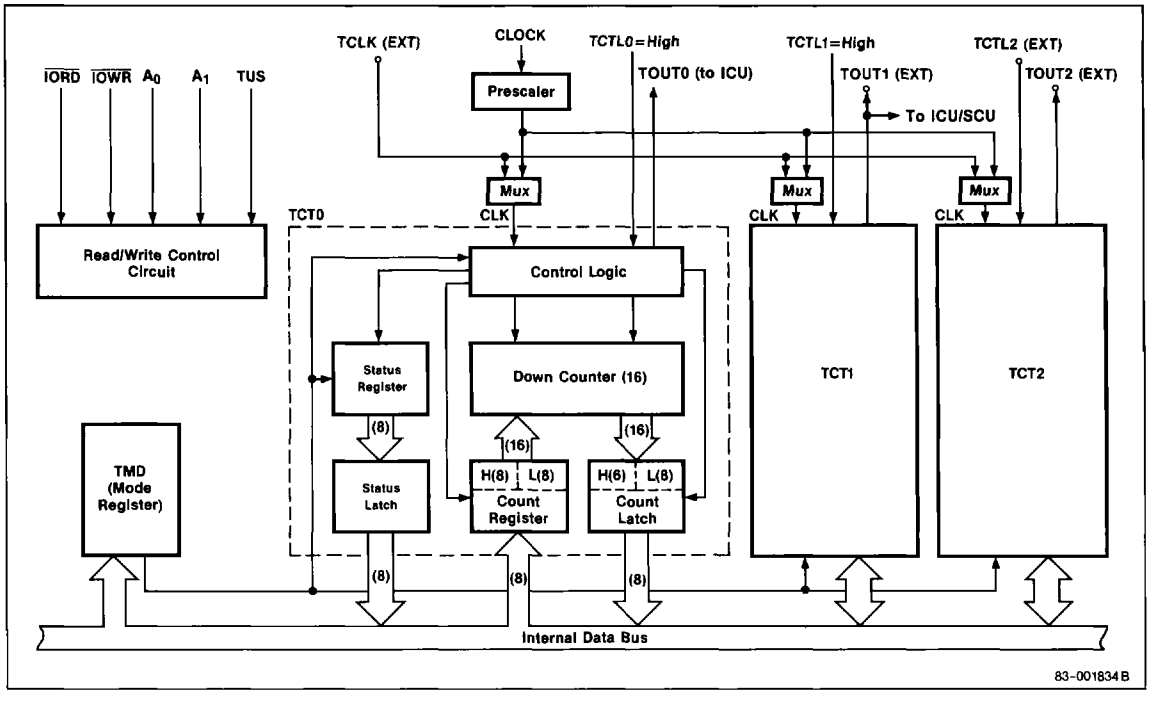

## Timer/Counter Unit

The timer/counter unit (TCU) provides a set of three independent 16-bit timer/counters. The output signal of timer/counter 0 is hardwired internally as an interrupt source. The output of timer/counter 1 is available internally as an interrupt source, used as a baud rate generator, or used as an external output. The timer/counter 2 output is available as an external output. Due to mode restrictions, the TCU is a subset of the μPD71054 Programmable Timer/Counter. Figure 19 shows the internal block diagram of the TCU.

The TCU has the following features:

- Three 16-bit timer/counters
- Six programmable count modes
- Binary/BCD counting
- Multiple latch command
- Choice of two clock sources

*Figure 19. TCU Block Diagram*



Because $\overline{\text{RESET}}$ leaves the TCU in an uninitialized state, each timer/counter must be initialized by specifying an operating mode and a count. Once programmed, a timer/counter will continue to operate in that mode until another mode is selected. When the count has been written to the counter and transferred to the down counter, a new count operation starts. Both the current count and the counter status can be read while count operations are in progress.

### TCU Commands

The TCU is programmed by issuing I/O instructions to the I/O port addresses programmed in the OPHA and TULA registers. The individual TCU registers are selected by address bits $A_1$ and $A_0$ as follows.

| $A_1$ | $A_0$ | Register | Operation |
|---|---|---|---|
| 0 | 0 | TCT0<br>TST0 | Read/Write<br>Read |
| 0 | 1 | TCT1<br>TST1 | Read/Write<br>Read |
| 1 | 0 | TCT2<br>TST2 | Read/Write<br>Read |
| 1 | 1 | TMD | Write |

The timer mode (TMD) register selects the operating mode for each timer/counter and issues the latch command for one or more timer/counters. Figure 20 shows the format for the TMD register.

Writes to the timer/counter 2-0 (TCT2-TCT0) registers stores the new count in the appropriate timer/counter. The count latch command is used before reading count data in order to latch the current count and prevent inaccuracies.

The timer status 2-0 (TST2-TST0) registers contain status information for the specified counter (figure 21). The latch command is used to latch the appropriate counter status before reading status information. If both status and counter data are latched for a counter, the first read operation returns the status data and subsequent read operations obtain the count data.

### Count Modes

There are six programmable timer/counter modes. The timing waveforms for these modes are in figure 22.

**Mode 0 [Interrupt on End of Count].** In this mode, TOUT changes from the low to high level when the specified count is reached. This mode is available on all timer/counters.

**Mode 1 [Retriggerable One-Shot].** In mode 1, a low-level one-shot pulse, triggered by TCTL2 is output from the TOUT2 pin. This mode is available only on timer/counter 2.

**Mode 2 [Rate Generator].** In mode 2, TOUT cyclically goes low for one clock period when the counter reaches the 0001H count. A counter in this mode operates as a frequency divider. All timer/counters can operate using mode 2.

**Mode 3 [Square-Wave Generator].** Mode 3 is a frequency divider similar to mode 2, but the output has a symmetrical duty cycle. This mode is available on all three timer/counters.

**Mode 4 [Software-Triggered Strobe].** In mode 4, when the specified count is reached, TOUT goes low for the duration of one clock pulse. Mode 4 is available on all timer/counters.

**Mode 5 [Hardware-Triggered Strobe].** Mode 5 is similar to mode 4 except that operation is triggered by the TCTL2 input and can be retriggered. This mode is available only on timer/counter 2.

## Serial Control Unit

The serial control unit (SCU) is a single asynchronous serial channel that performs serial communication between the μPD70208 and an external serial device. The SCU is similar to the μPD71051 Serial Control Unit except for the lack of synchronous communication protocols. Figure 23 is the block diagram of the SCU.

The SCU has the following features.

- Full-duplex asynchronous serial controller
- Clock rate divisor (x16, x64)
- Baud rates to 250 kb/s supported
- 7-, 8-bit character lengths
- 1-, 2-bit stop bit lengths
- Break transmission and detection
- Full-duplex, double-buffered transmitter/receiver
- Even, odd, or no parity
- Parity, overrun, and framing error detection
- Receiver-full/transmitter-empty interrupt

The SCU contains four separately addressable registers for reading/writing data, reading status, and controlling operation of the SCU. The serial receive buffer (SRB) and the serial transmit buffer (STB) store the incoming and outgoing character data. The serial status (SST) register allows software to determine the current state of both the transmitter and receiver. The serial command (SCM) and serial mode (SMD) registers determine the operating mode of the SCU while the serial interrupt mask (SIMK) register allows software control of the SCU receive and transmit interrupts.

*Figure 20. Timer Mode Register*

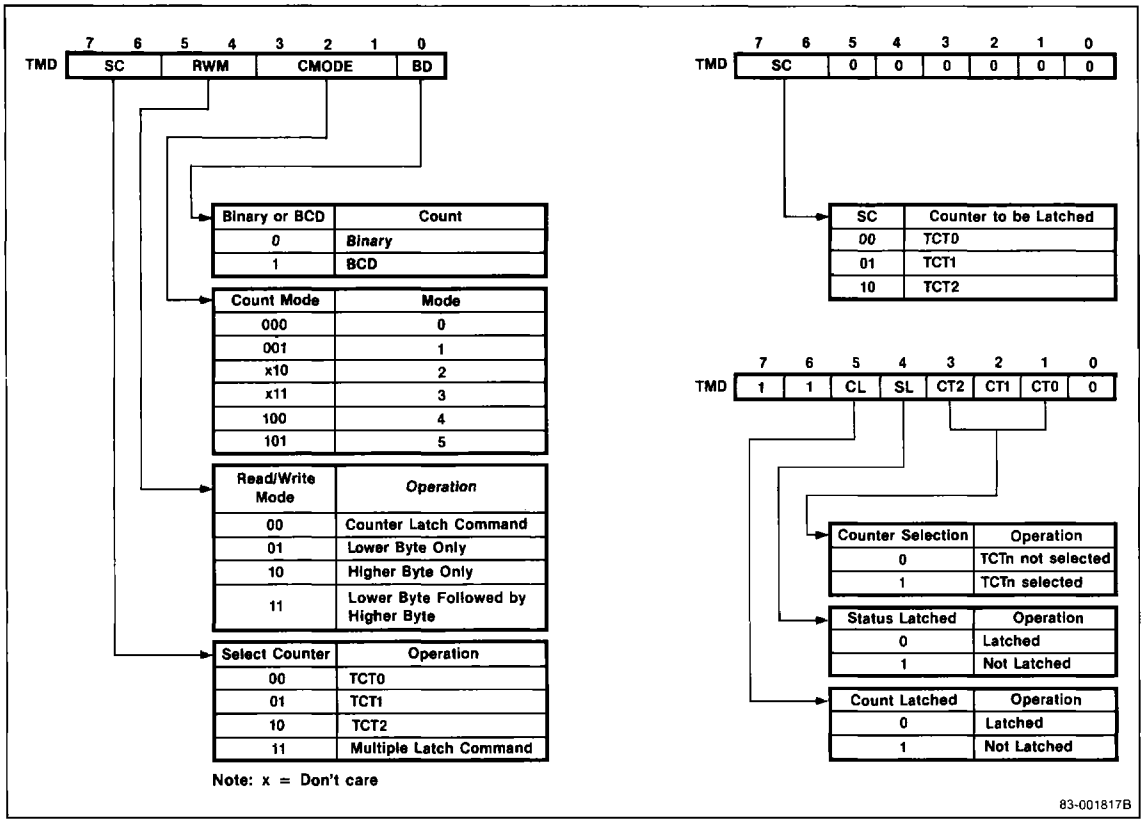



*Figure 21. TCU Status Register*

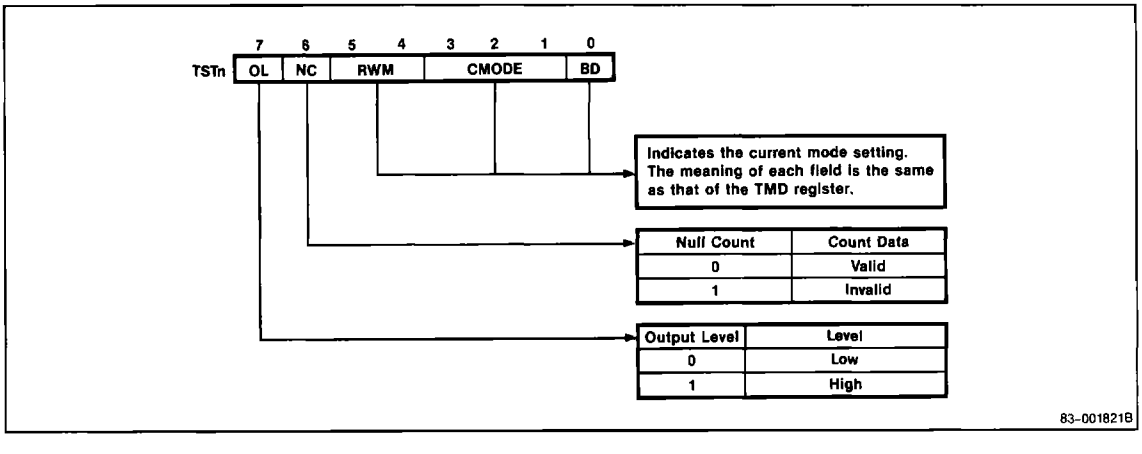
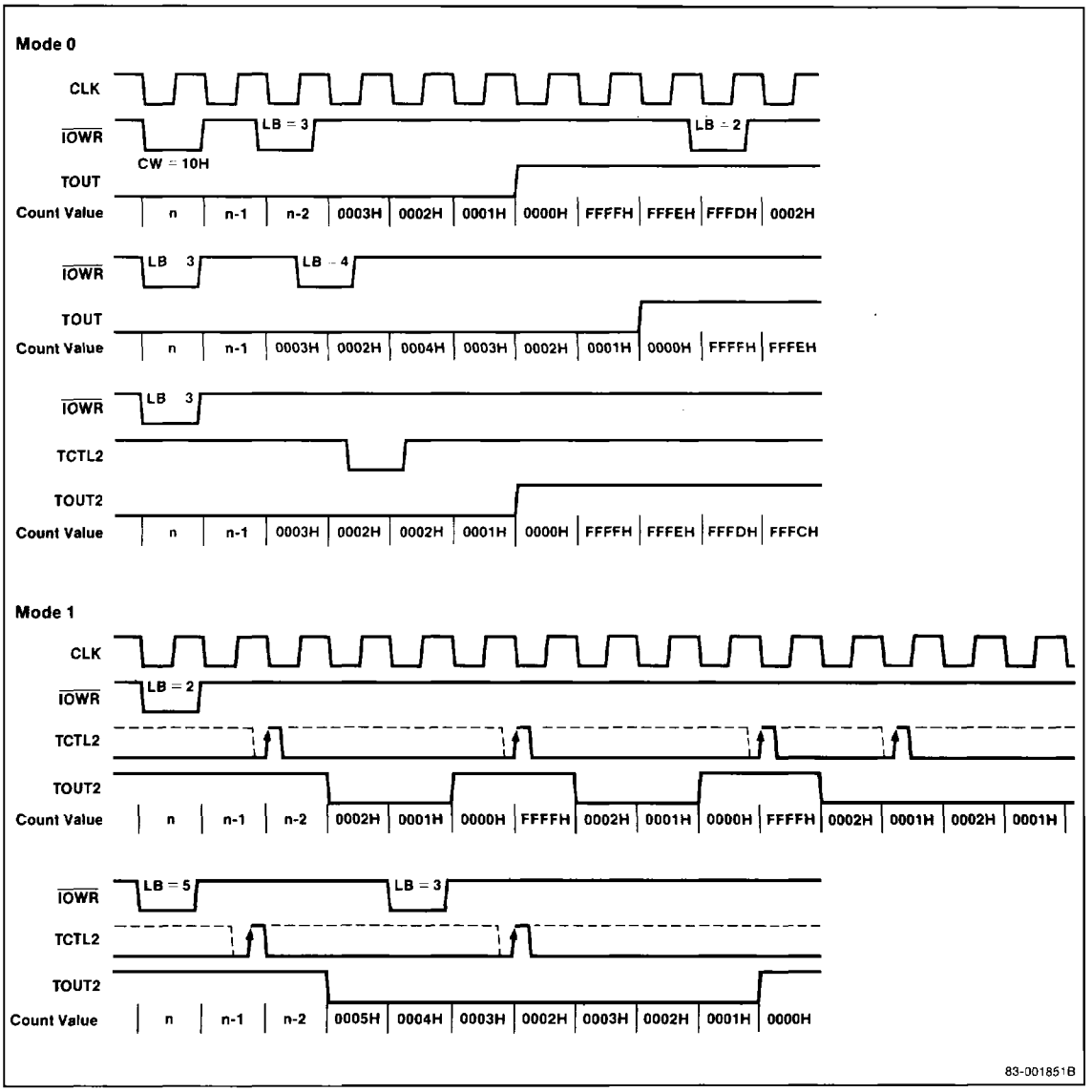


*Figure 22. TCU Waveforms (Sheet 1 of 3)*




3c

*Figure 22. TCU Waveforms (Sheet 2 of 3)*

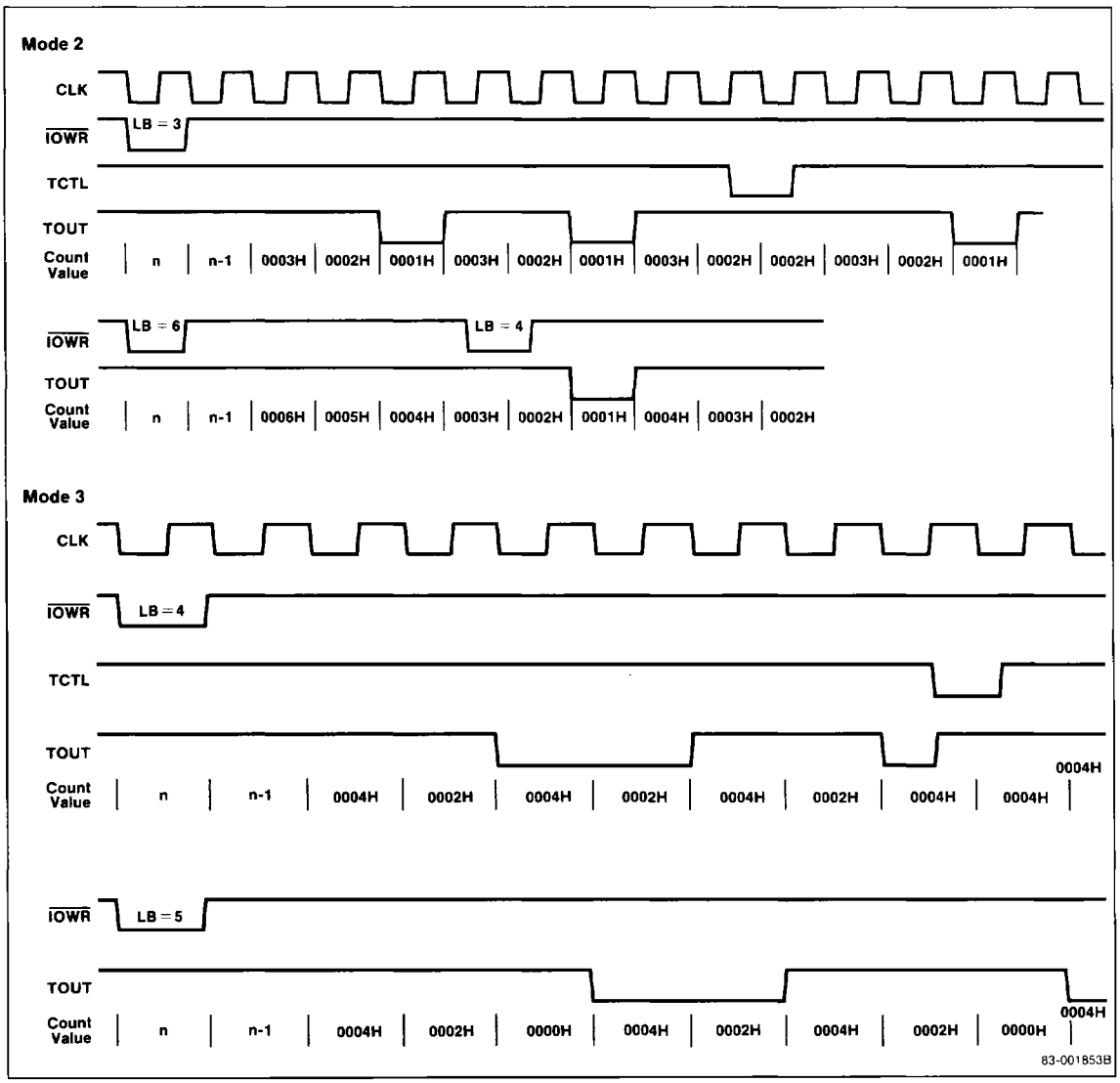
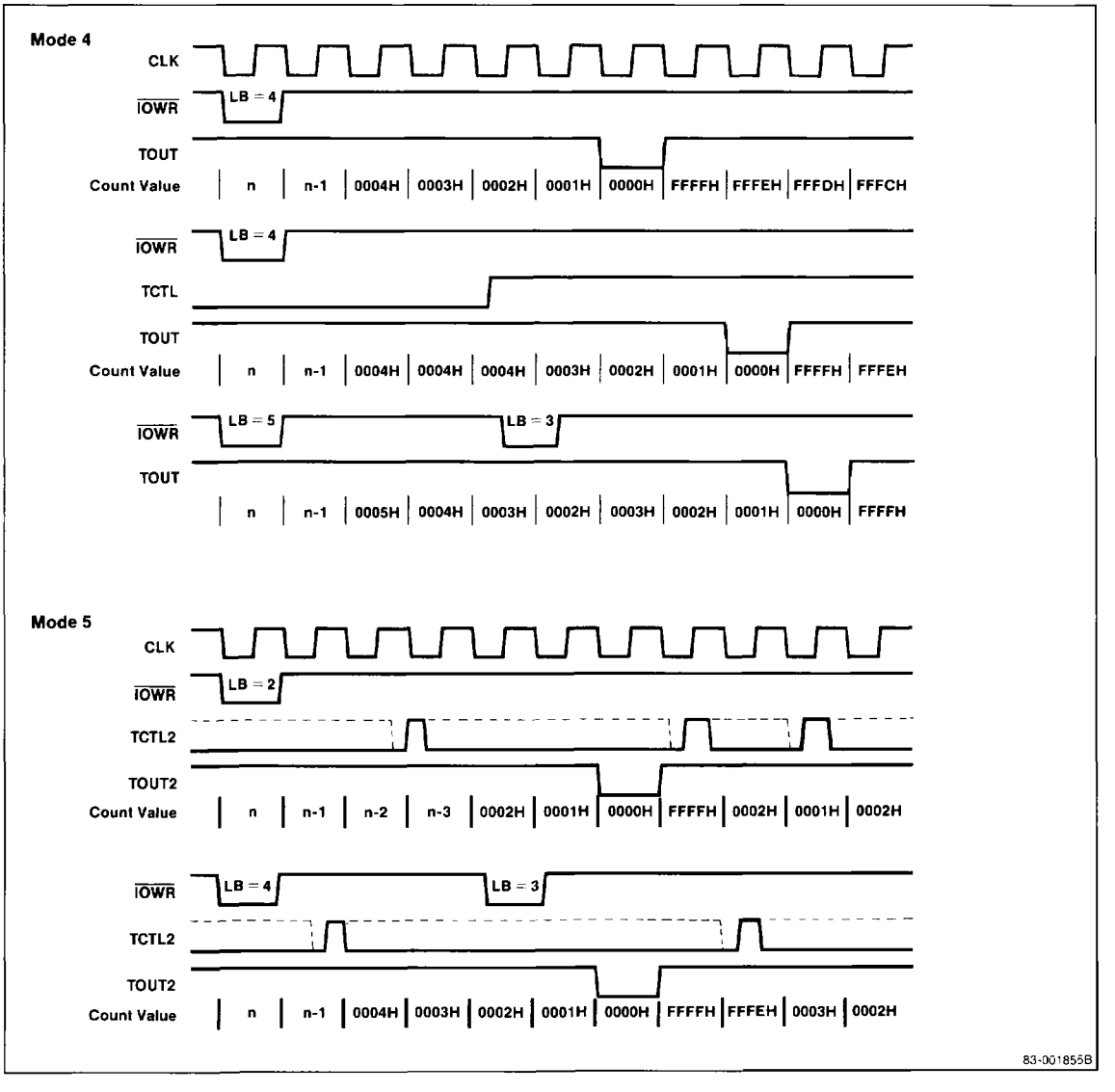
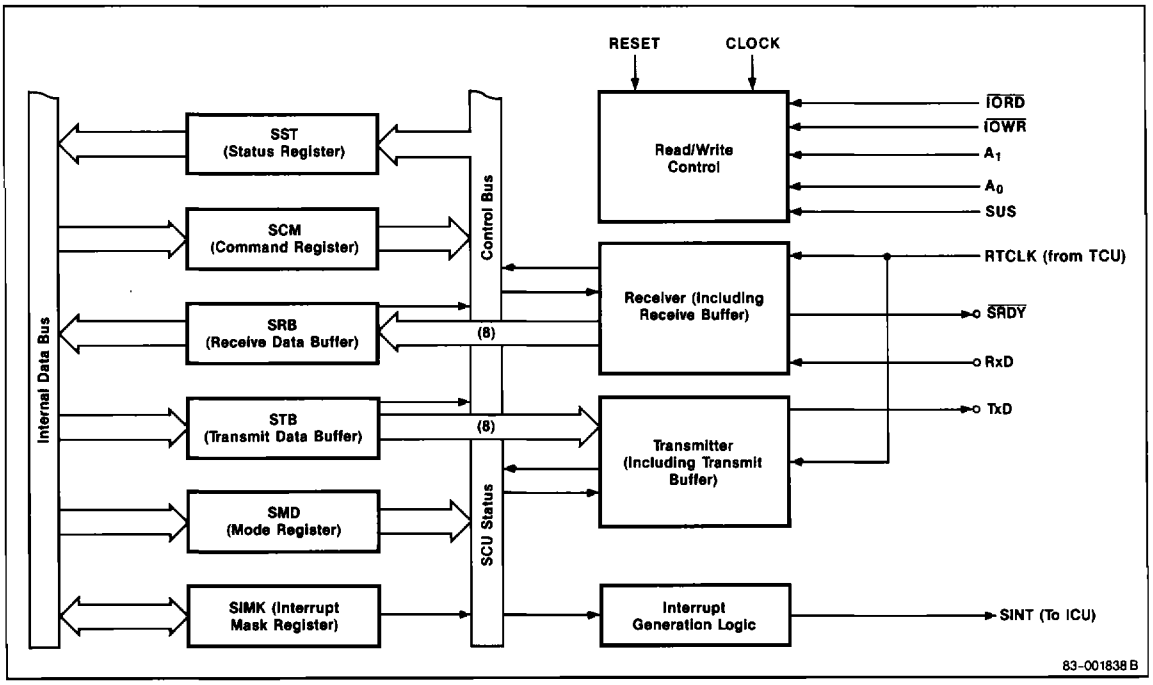


*Figure 22. TCU Waveforms (Sheet 3 of 3)*




3c

*Figure 23. SCU Block Diagram*




## Receiver Operation

While the RxD pin is high, the receiver is in an idle state. A transition on RxD from high to low indicates the start of new serial data. When a complete character has been received, it is transferred to the SRB; the receive buffer ready (RBRDY) bit in the SST register is set and (if unmasked) an interrupt is generated. The SST also latches any parity, overrun, or framing errors at this time.

The receiver detects a break condition when a null character with zero parity is received. The BRK bit is set for as long as the subsequent receive data is low and resets when RxD returns to a high level. The MRDY bit (SCM) and RBRDY (SST) are gated to form the output $\overline{\text{SRDY}}$. $\overline{\text{SRDY}}$ prevents overruns from occurring when the program is unable to process the input data. Software can control MRDY to prevent data from being sent from the remote transmitter while RBRDY can prevent the immediate overrun of a received character.

## Transmitter Operation

TxD is kept high while the STB register is empty. When the transmitter is enabled and a character is written to the STB register, the data is converted to serial format and output on the TxD pin. The start bit indicates the start of the transmission and is followed by the character stream (LSB to MSB) and an optional parity bit. One or two stop bits are then appended, depending on the programmed mode. When the character has been transferred from the STB, the TRBDY bit in the SST is set and if unmasked, a transmit buffer empty interrupt is generated.

Serial data can be transmitted and received by polling the SST register and checking the TBRDY or RBRDY flags. Data can also be transmitted and received by SCU-generated interrupts to the interrupt control unit. The SCU generates an interrupt in either of these conditions:

(1) The receiver is enabled, the SRB is full, and receive interrupts are unmasked.

(2) The transmitter is enabled, the STB is empty, and transmit interrupts are unmasked.

### SCU Registers and Commands

I/O instructions to the I/O addresses selected by the OPHA and SULA registers are used to read/write the SCU registers. Address bits $A_1$ and $A_0$ and the read/write lines select one of the six internal registers as follows:

| $A_1$ | $A_0$ | Register | Operation |
|---|---|---|---|
| 0 | 0 | SRB<br>STB | Read<br>Write |
| 0 | 1 | SST<br>SCM | Read<br>Write |
| 1 | 0 | SMD | Write |
| 1 | 1 | SIMK | Read/Write |

The SRB and STB are 8-bit registers. When the character length is 7 bits, the lower 7 bits of the SRB register are valid and bit 7 is cleared to 0. If programmed for 7-bit characters, bit 7 of the STB is ignored.

The SST register (figure 24) contains the status of the transmit and receive data buffers and the error flags. Error flags are persistent. Once an error flag is set, it remains set until a clear error flags command is issued.

Figure 25 shows the SCM and SMD registers. The SCM register stores the command word that controls transmission, reception, error flag reset, break transmission, and the state of the $\overline{SRDY}$ pin. The SMD register stores the mode word that determines serial characteristics such as baud rate divisor, parity, character length, and stop bit length.

Initialization software should first program the SMD register followed by the SCM register. Unlike the μPD71051, the SMD register can be modified at any time without resetting the SCU.

The SIMK register (figure 26) controls the occurrence of RBRDY and TBRDY interrupts. When an interrupt is masked, it is prevented from propagating to the interrupt control unit.

3c

### Baud Rate Generator

Timer/counter 1 is used as the baud rate generator when the SCU is enabled. The input baud rate clock is scaled by 16 or 64, as selected in the SMD register, to determine the receive/transmit data clock. There are no restrictions on the SCU input baud rate clock other than operating the TCU in mode 3 with a square-wave output.

***Figure 24. SST Register***

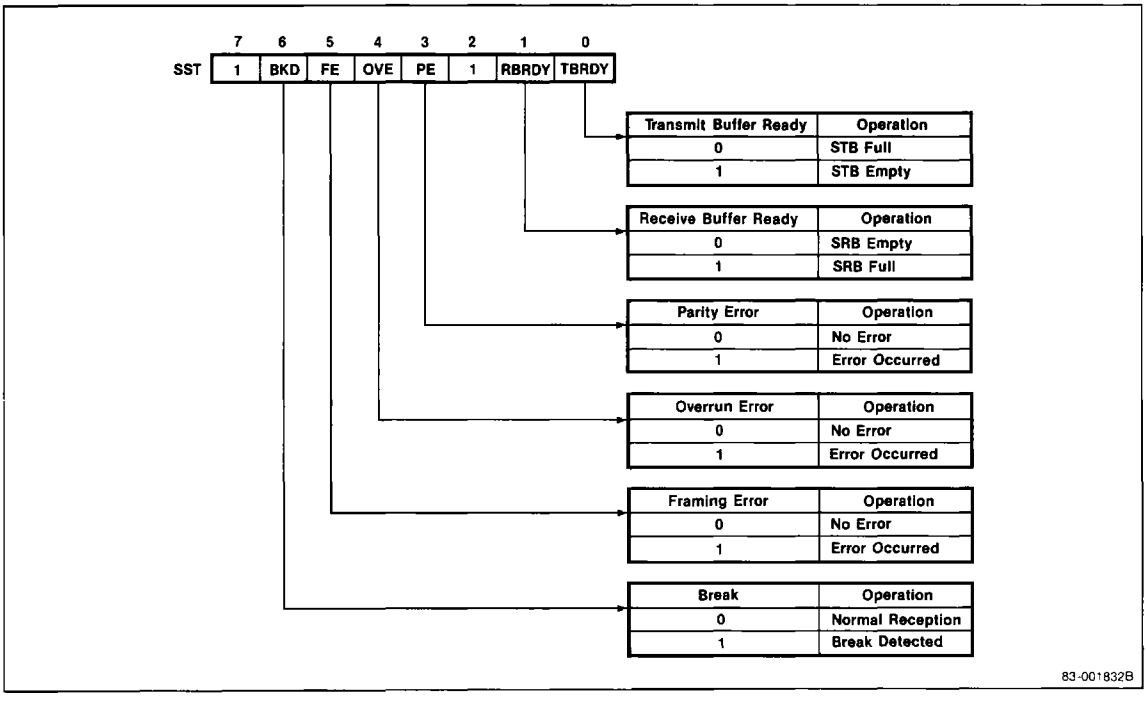


*Figure 25. SCM and SMD Registers*

SCM Register

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| SCM | — | — | $\overline{\text{MRDY}}$ | ECL | SBRK | RE | — | TE |

| Transmitter Enabled | Operation |
|---|---|
| 0 | Transmitter Disabled |
| 1 | Transmitter Enabled |

| Receiver Enabled | Operation |
|---|---|
| 0 | Receiver Disabled |
| 1 | Receiver Enabled |

| Send Break | Operation |
|---|---|
| 0 | Normal Operation |
| 1 | TxD = 0 (Break) |

| Error Clear | Operation |
|---|---|
| 0 | No Operation |
| 1 | Error Flag Clear |

| $\overline{\text{MRDY}}$ | Mask Ready |
|---|---|
| 0 | $\overline{\text{SRDY}}$ = 1 (Mask) |
| 1 | Normal Operation of $\overline{\text{SRDY}}$ Output |

SMD Register

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| SMD | STL | | PS | | CL | | BF | |

| Baud Rate Factor | Operation |
|---|---|
| 0- | Illegal |
| 10 | RTCLK Frequency ÷ 16 |
| 11 | RTCLK Frequency ÷ 64 |

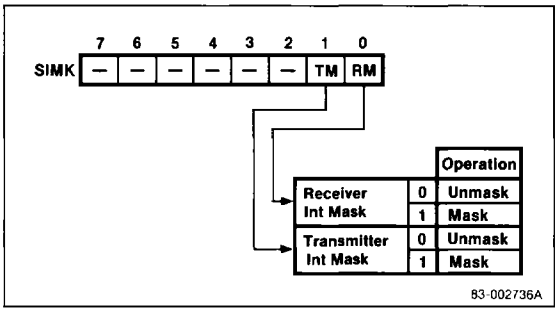
| Character Length | Operation |
|---|---|
| 0- | Illegal |
| 10 | 7 Bit Characters |
| 11 | 8 Bit Characters |

| Parity Select | Operation |
|---|---|
| -0 | No Parity |
| 01 | Odd Parity |
| 11 | Even Parity |

| Stop Bit Length | Operation |
|---|---|
| -0 | Illegal |
| 01 | 1 Stop Bit |
| 11 | 2 Stop Bits |

83-001836B

*Figure 26. SIMK Register*




## Interrupt Control Unit

The interrupt control unit (ICU) is a programmable interrupt controller equivalent to the μPD71059. The ICU arbitrates up to eight interrupt inputs, generates a CPU interrupt request, and outputs the interrupt vector number on the internal data bus during an interrupt acknowledge cycle. Cascading up to seven external slave μPD71059s permits the μPD70208 to support up to 56 interrupt sources. Figure 27 is the block diagram for the ICU.

The ICU has the following features.

- Eight interrupt request inputs
- Cascadable with μPD71059 Interrupt Controllers
- Programmable edge- or level-triggered interrupts (TCU, edge-triggered interrupts only)
- Individually maskable interrupt requests
- Programmable interrupt request priority
- Polling mode

### ICU Registers

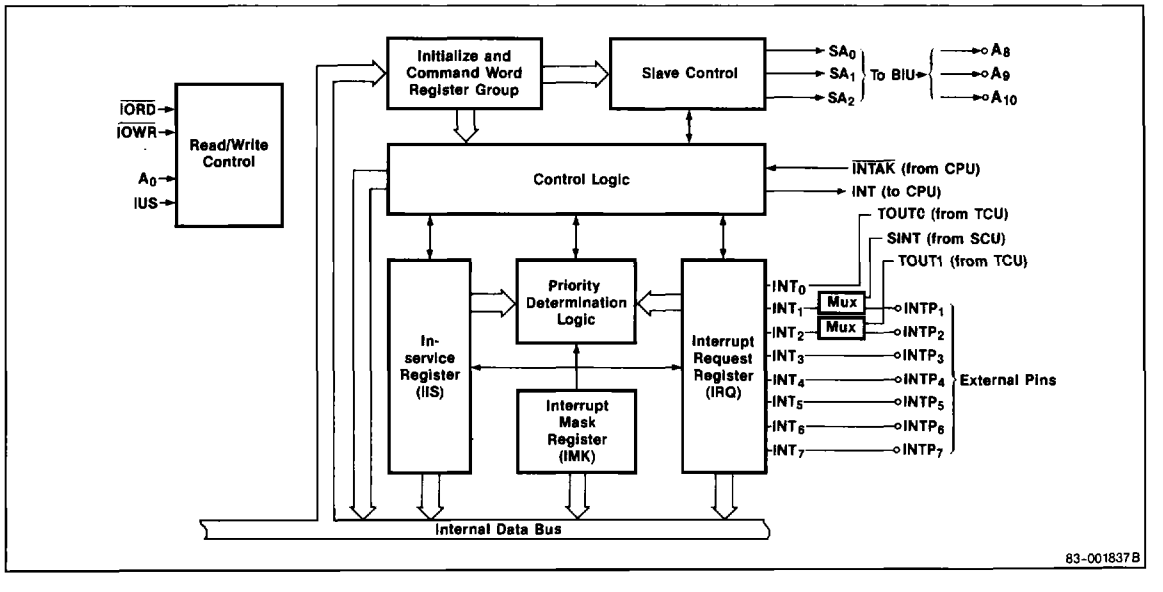
Use I/O instructions to the I/O addresses selected by the OPHA and IULA registers to read from and write to the ICU registers. Address bit $A_0$ and the command word selects an ICU internal register.

| | $A_0$ | Other Condition | Operation |
|---|---|---|---|
| Read | 0 | IMD selects IRQ | CPU ← IRQ data |
| | 0 | IMD selects IIS | CPU ← IIS data |
| | 0 | Polling phase | CPU ← Polling data |
| | 1 | — | CPU ← IMKW |
| Write | 0 | D4 = 1 | CPU → IIW1 |
| | 0 | D4 = 0 and D3 = 0 | CPU → IPFW |
| | 0 | D4 = 0 and D3 = 1 | CPU → IMDW |
| | 1 | During initialization | CPU → IIW2 |
| | 1 | | CPU → IIW3 |
| | 1 | | CPU → IIW4 |
| | 1 | After initialization | CPU → IMKW |

**Note:**

(1) In polling phase, polling data has priority over the contents of the IRQ or IIS register when read.

*Figure 27. ICU Block Diagram*



### Initializing the ICU

The ICU is always used to service maskable interrupts in a μPD70208 system. Prior to accepting maskable interrupts, the ICU must first be initialized (figure 28). Following initialization, command words from the CPU can change the interrupt request priorities, mask/unmask interrupt requests, and select the polling mode. Figures 29 and 30 list the ICU initialization and command words.

Interrupt initialization words 1-4 (IIW1-IIW4) initialize the ICU, indicate whether external μPD71059s are connected as slaves, select the base interrupt vector, and select edge- or level-triggered inputs for INT1-INT7. Interrupt sources from the TCU are fixed as edge-triggering. INT0 is internally connected to TOUT0, and INT2 may be connected to TOUT1 by the IRSW field in the OPCN.

The interrupt mask word (IMKW) contains programmable mask bits for each of the eight interrupt inputs. The interrupt priority and finish word (IPFW) is used by the interrupt handler to terminate processing of an interrupt or change interrupt priorities. The interrupt mode word (IMDW) selects the polling register, interrupt request (IRQ) or interrupt in service (IIS) register, and the nesting mode.

The initialization words are written in consecutive order starting with IIW1. IIW2 sets the interrupt vector. IIW3 specifies which interrupts are connected to slaves. IIW3 is only required in extended systems. The ICU will only expect to receive IIW3 if SNGL = 0 (bit $D_1$ of IIW1). IIW4 is only written if II4 = 1 (bit $D_0$ of IIW1).

### μPD71059 Cascade Connection

To increase the number of maskable interrupts, up to seven slave μPD71059 Interrupt Controllers can be cascaded. During cascade operation (figure 31), each slave μPD71059 INT output is routed to one of the μPD70208 INTP inputs. During the second interrupt acknowledge bus cycle, the ICU places the slave address on address lines $A_{10}$-$A_8$. Each slave compares this address with the slave address programmed using interrupt initialization word 3 (IIW3). If the same, the slave will place the interrupt vector on pins $AD_7$-$AD_0$ during the second interrupt acknowledge bus cycle.

***Figure 28. Initialization Sequence***

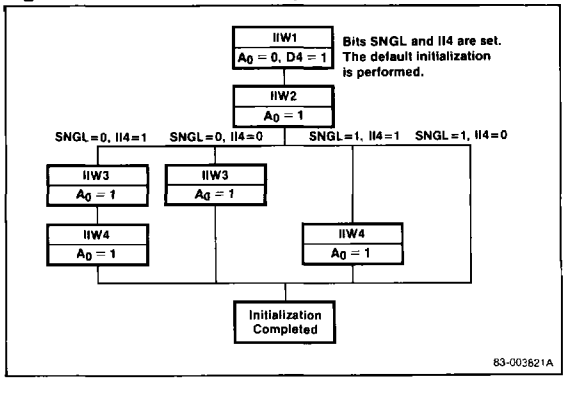



***Figure 29. Interrupt Initialization Words 1-4***

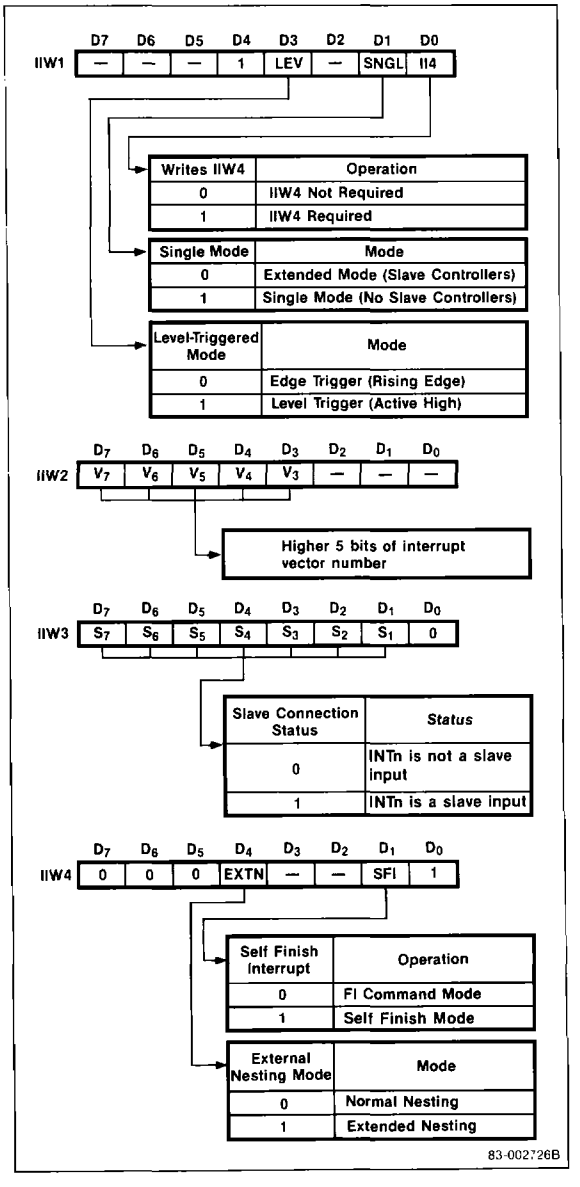
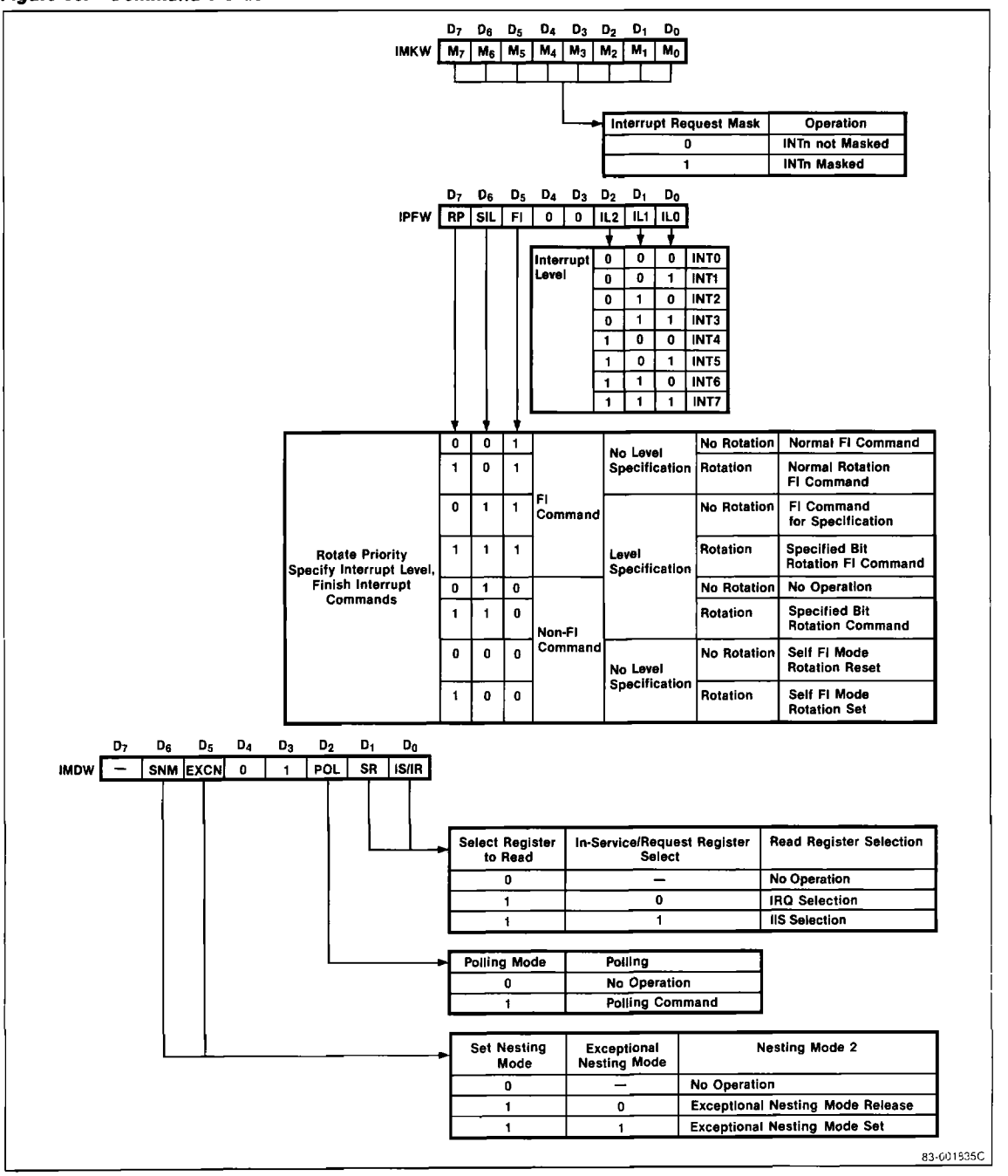
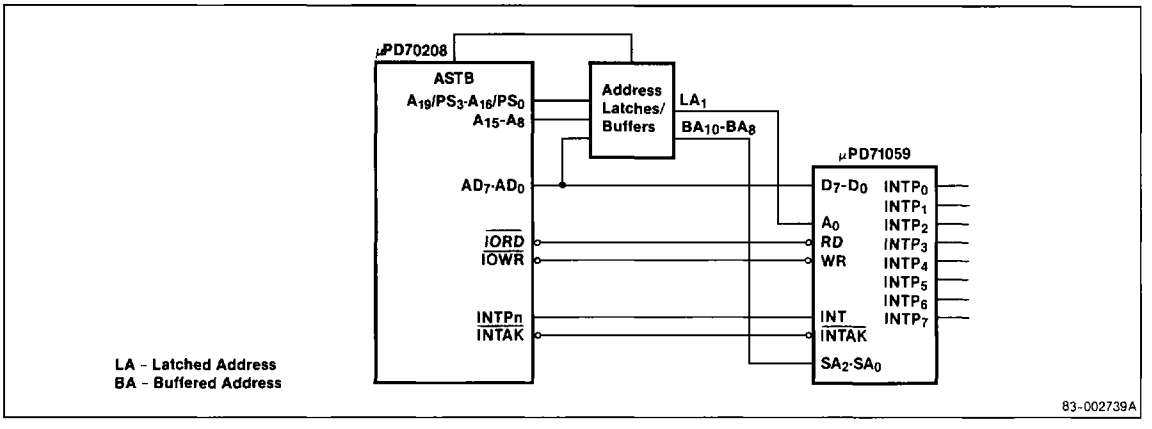


*Figure 30. Command Words*




3c

*Figure 31. μPD71059 Cascade Connection*




## DMA Control Unit

The DMA Control Unit (DMAU) is a high-speed DMA controller compatible with the μPD71071 DMA Controller. The DMAU has four independent DMA channels and performs high-speed data transfers between memory and external peripheral devices at speeds as high as 2 megabytes/second in an 8-MHz system. Figure 32 is the block diagram for the DMAU.

The DMAU has the following features.

- Four independent DMA channels
- Cascade mode for slave μPD71071 DMA controllers
- 20-bit address registers
- 16-bit transfer count registers
- Single, demand, and block transfer modes
- Bus release and bus hold modes
- Autoinitialization
- Address increment/decrement
- Fixed/rotating channel priorities
- TC output at transfer end
- Forced termination of service by $\overline{END}$ input

### DMAU Basic Operation

The DMAU operates in either a slave or master mode. In the slave mode, the DMAU samples the four DMARQ input pins every clock. If one or more inputs are active, the corresponding DMA request bits are set and the DMAU sends a bus request to the BAU while continuing to sample the DMA request inputs. After the BAU returns the DMA bus acknowledge signal, the DMAU stops DMA request sampling, selects the DMA channel with the highest priority, and enters the bus master mode to perform the DMA transfer. While in the bus master mode, the DMAU controls the external bus and performs DMA transfers based on the preprogrammed channel information.

### Terminal Count

The DMAU ends DMA service when the terminal count condition is generated or when the $\overline{END}$ input is asserted. A terminal count (TC) is produced when the contents of the current count register becomes zero. If autoinitialization is not enabled when DMA service terminates, the mask bit of the channel is set and the DMARQ input of that channel is masked. Otherwise, the current count and address registers are reloaded from the base registers and new DMA transfers are again enabled.

### DMA Transfer Type

The type of transfer the DMAU performs depends on the following conditions.

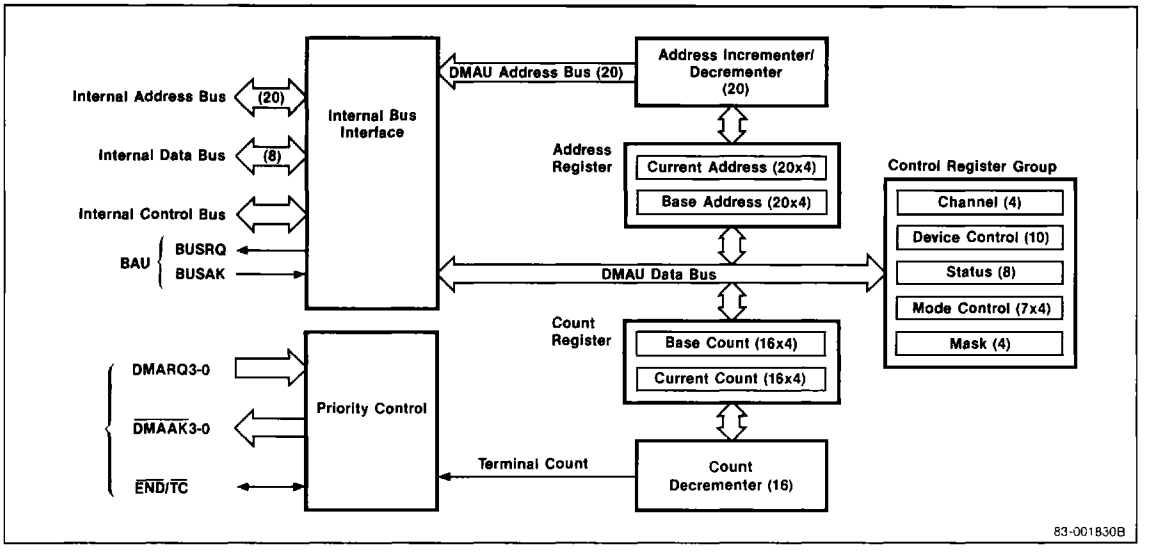
- Direction of the transfer (each channel)
- Transfer mode (each channel)
- Bus mode

### Transfer Direction

All DMA transfers use memory as a reference point. Therefore, a DMA read operation transfers data from memory to an I/O port. A DMA write operation reads an I/O port and writes the data into memory. During memory-to-I/O transfer, the DMA mode (DMD) register is used to select the transfer directions for each channel and activate the appropriate control signals.

| Operation | Transfer Direction | Activated Signals |
|---|---|---|
| DMA read | Memory → I/O | $\overline{IOWR}$, $\overline{MRD}$ |
| DMA write | I/O → Memory | $\overline{IORD}$, $\overline{MWR}$ |
| DMA verify | | Addresses only; no transfer performed |

*Figure 32. DMAU Block Diagram*

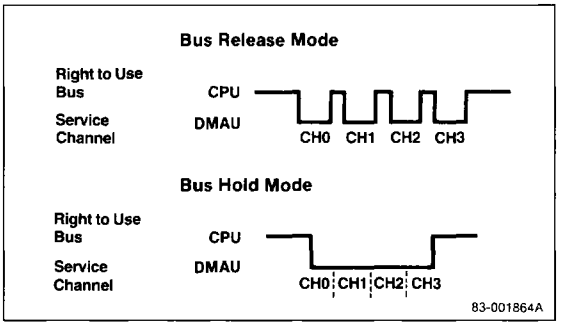



## Bus Mode

The DMA device control (DDC) register selects operation in either the bus release or bus hold mode. The selected bus mode determines the DMAU conditions for return of the bus to the BAU. Figure 33 shows that in bus release mode, only a single channel is serviced after the DMAU obtains the bus. When DMA service ends (termination conditions depend on the transfer mode), the DMAU returns the bus to the BAU regardless of the state of other DMA requests, and the DMAU reenters the slave mode. When the DMAU regains use of the bus, a new DMA operation can begin.

In bus hold mode, several channels can receive contiguous service without releasing the bus. If there is another valid DMA request when a channel's DMA service is finished, the new DMA service can begin immediately after the previous service without returning the bus to the BAU.

*Figure 33. Bus Modes*




## Transfer Modes

The DMD register also selects either single, demand, or block transfer mode for each channel. The conditions for the termination of each transfer characterize each transfer mode. The following table shows the various transfer modes and termination conditions.

| Transfer Mode | Termination Conditions |
|---|---|
| Single | After each byte/word transfer |
| Demand | $\overline{\text{END}}$ input<br>Terminal count<br>Inactive DMARQ<br>DMARQ of a higher priority channel becomes active (bus hold mode) |
| Block | $\overline{\text{END}}$ input<br>Terminal count |

The operation of single, demand, and block mode transfers depends on whether the DMAU is in bus release or bus hold mode. Figure 34 shows the operation flow for the six possible transfer and bus mode operations in DMA transfer.

**Single-Mode Transfer.** In bus release mode, when a channel completes transfer of a single byte, the DMAU enters the slave mode regardless of the state of DMA request inputs. In this manner, other lower-priority bus masters will be able to access the bus.

In bus hold mode, when a channel completes transfer of a single byte, the DMAU terminates the channel's service even if the DMARQ request signal is asserted. The DMAU will then service any other requesting channel. If there are no requests from any other DMA channels, the DMAU releases the bus and enters the slave state.

**Demand-Mode Transfer.** In bus release mode, the currently active channel continues to transfer data as long as the DMA request of that channel is active, even though other DMA channels are issuing higher-priority requests. When the DMA request of the serviced channel becomes inactive, the DMAU releases the bus and enters the slave state.

In bus hold mode, when the active channel completes a single transfer, the DMAU checks the other DMA request lines without ending the current service. If there is a higher-priority DMA request, the DMAU stops the service of the current channel and starts servicing the highest-priority channel requesting service. If there is no higher request than the current one, the DMAU continues to service the currently active channel. Lower-priority DMA requests are honored without releasing the bus after the current channel service is complete.

**Block-Mode Transfer.** In bus release mode, the current channel continues DMA transfers until a terminal count or the external $\overline{END}$ input becomes active. During this time, the DMAU ignores all other DMA requests. After completion of the block transfer, the DMAU releases the bus and enters the slave state, even if DMA requests from other channels are active.

In bus hold mode, the current channel transfers data until an internal or external $\overline{END}$ signal becomes active. When the service is complete, the DMAU checks all DMA requests without releasing the bus. If there is an active request, the DMAU immediately begins servicing the request. The DMAU releases the bus after it honors all DMA requests or a higher-priority bus master requests the bus.

### Byte Transfer

The DMD register can specify only byte DMA transfers for each channel. Depending on the mode selected, the address register can either increment or decrement, whereas the count register is always decremented.

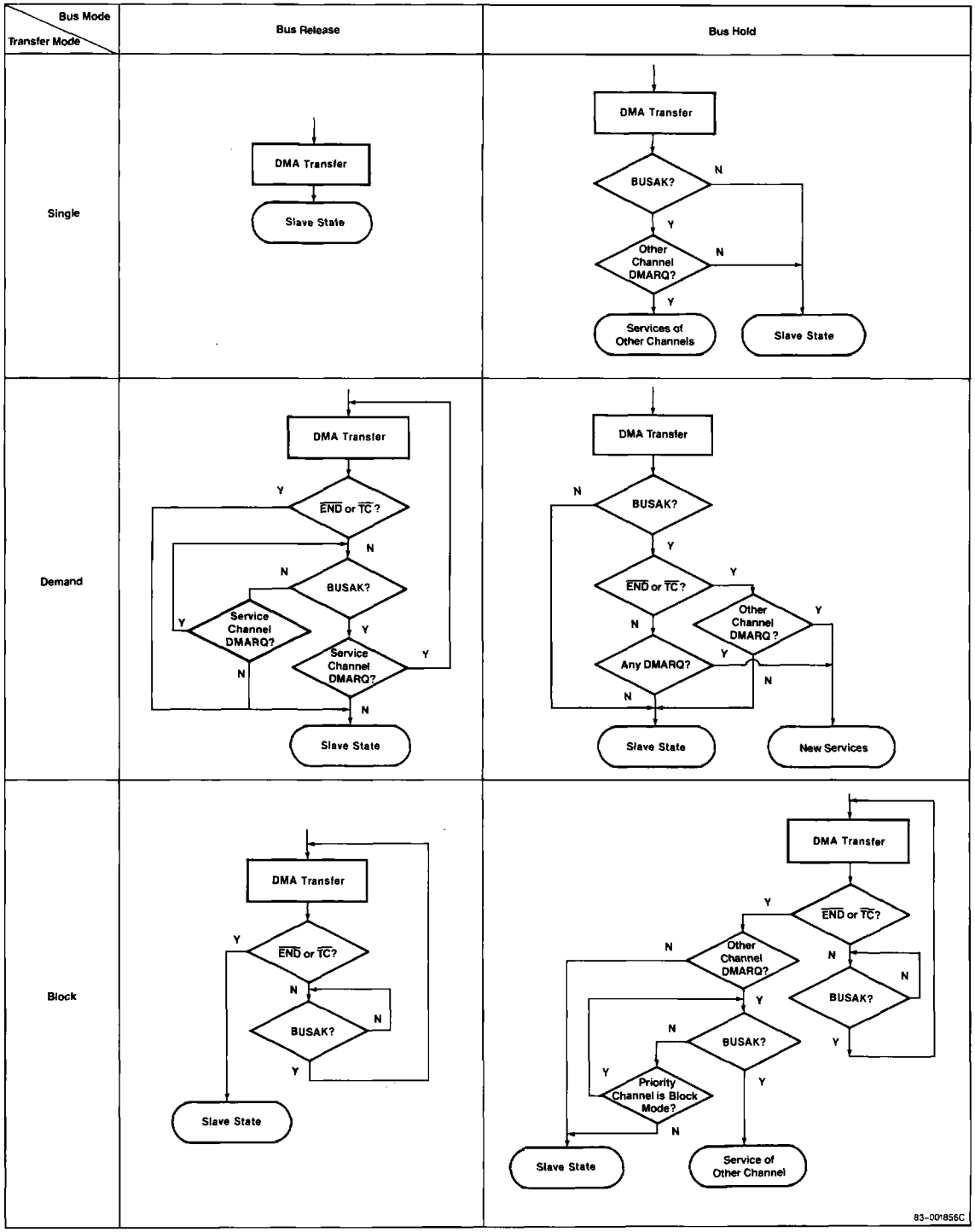
### Autoinitialize

When the DMD register selects autoinitialize for a channel, the DMAU automatically reinitializes the address and count registers when $\overline{END}$ is asserted or the terminal count condition is reached. The contents of the base address and base count registers are transferred to the current address and current count registers, and the applicable bit of the mask register remains cleared.

### Channel Priority

Each of the four DMAU channels is assigned a priority. When multiple DMA requests from several channels occur simultaneously, the channel with the highest priority will be serviced first. The DDC register selects one of two priority schemes: fixed or rotating (figure 35). In fixed priority, channel 0 is assigned the highest priority and channel 3, the lowest. In rotating priority, priority order is rotated after each service so that the channel last serviced receives the lowest priority. This method prevents the exclusive servicing of higher-priority channels and the lockout of lower-priority DMA channels.

***Figure 34. Transfer Modes***




3c

## Cascade Connection

Slave μPD71071 DMA Controllers can be cascaded to easily expand the system DMA channel capacity to 16 DMA channels. Figure 36 shows an example of cascade connection. During cascade operation, the DMAU acts as a mediator between the BAU and the slave μPD71071s. During DMA cascade mode operation, it is the responsibility of external logic to isolate the cascade bus master from the μPD70208 control outputs. These outputs are listed in a table at the front of this data sheet.

The DMAU always operates in the bus hold mode while a cascade channel is in service, even when the bus release mode is programmed. Other DMA requests are held pending while a slave μPD71071 channel is in service. When the cascaded μPD71071 ends service and moves into the slave state, the DMAU also moves to the slave state and releases the bus. At this time, all bits of the DMAU request register are cleared. The DMAU continues to operate normally with the other noncascaded channels.

***Figure 35. Priority Order***

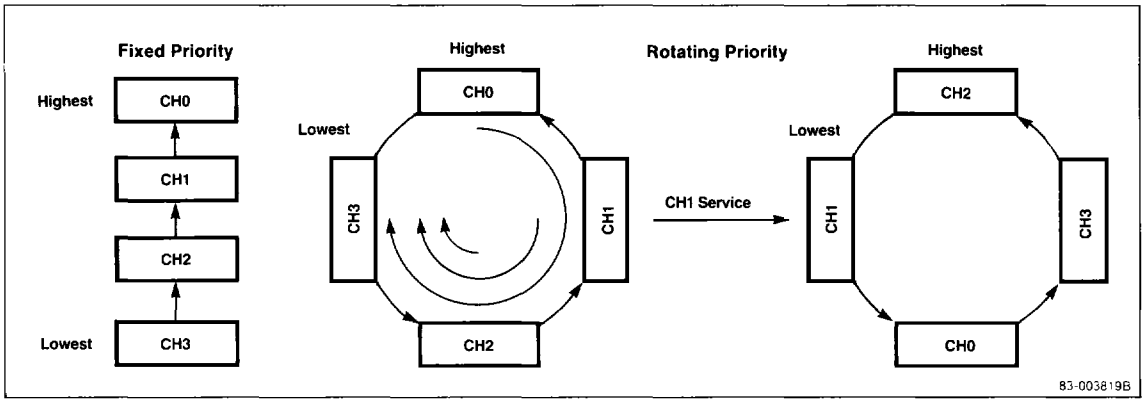
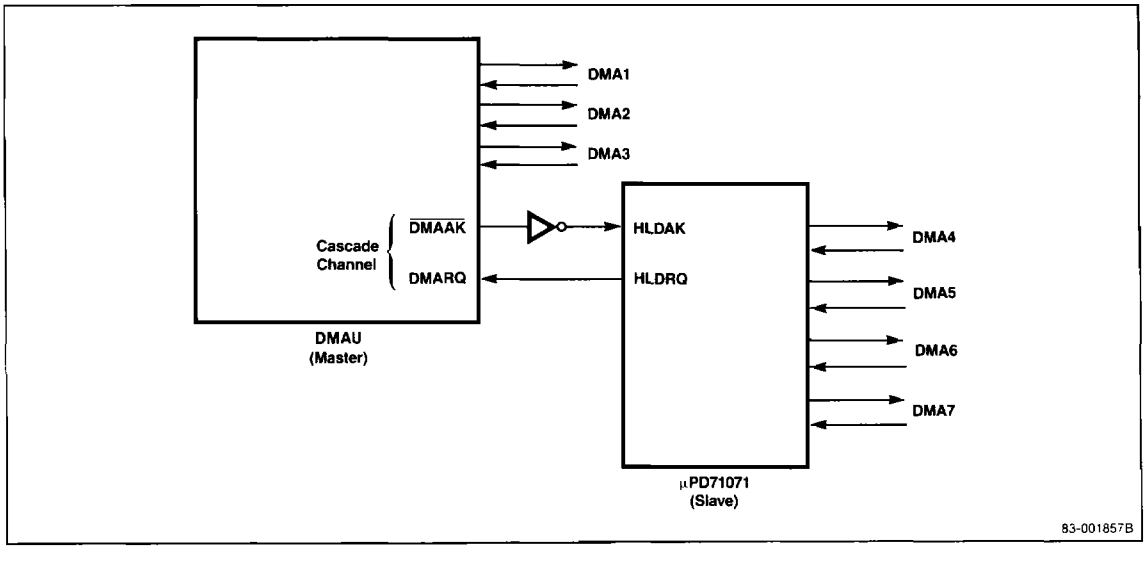



***Figure 36. μPD71071 Cascade Example***



### Bus Waiting Operation

The DMAU will automatically perform a bus waiting operation (figure 37) whenever the RCU refresh request queue fills. When the DMA bus acknowledge goes inactive, the DMAU enters the bus waiting mode and inactivates the DMA bus request signal. Control of the bus is then transferred to the higher-priority RCU by the BAU.

Two clocks later, the DMAU reasserts its internal DMA bus request. The bus waiting mode is continued until the DMA bus acknowledge signal again becomes active and the interrupted DMA service is immediately restarted.

### Programming the DMAU

To prepare a channel for DMA transfer, the following characteristics must be programmed.

- Starting address for the transfer
- Transfer count
- DMA operating mode
- Transfer size (byte/word)

The contents of the OPHA and DULA registers determine the base I/O port address of the DMAU. Addresses $A_3$-$A_0$ are used to select a particular register as follow:

| $A_3$ | $A_2$ | $A_1$ | $A_0$ | Register | Operation |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | DICM | Write |
| 0 | 0 | 0 | 1 | DCH | Read/Write |
| 0 | 0 | 1 | 0 | DBC/DCC (low) | Read/Write |
| 0 | 0 | 1 | 1 | DBC/DCC (high) | Read/Write |
| 0 | 1 | 0 | 0 | DBA/DCA (low) | Read/Write |
| 0 | 1 | 0 | 1 | DBA/DCA (high) | Read/Write |
| 0 | 1 | 1 | 0 | DBA/DCA (upper) | Read/Write |
| 0 | 1 | 1 | 1 | Reserved | — |
| 1 | 0 | 0 | 0 | DDC (low) | Read/Write |
| 1 | 0 | 0 | 1 | DDC (high) | Read/Write |
| 1 | 0 | 1 | 0 | DMD | Read/Write |
| 1 | 0 | 1 | 1 | DST | Read |
| 1 | 1 | 0 | 0 | Reserved | — |
| 1 | 1 | 0 | 1 | Reserved | — |
| 1 | 1 | 1 | 0 | Reserved | — |
| 1 | 1 | 1 | 1 | DMK | Read/Write |

Word I/O instructions can be used to read/write the register pairs listed below. All other registers are accessed via byte I/O instructions.

DBC/DCC
DBA/DCA (higher/lower only)
DDC

### DMAU Registers

**Initialize.** The DMA initialize command (DICM) register (figure 38) is used to perform a software reset of the DMAU. The DICM is accessed using the byte OUT instruction.

**Channel Register.** Writes to the DMA channel (DCH) register (figure 39) select one of the four DMA channels for programming and also the base/current registers. Reads of the DCH register return the currently-selected channel and the register access mode.

**Count Registers.** When bit 2 of the DCH register is cleared, a write to the DMA count register updates both the DMA base count (DBC) and the DMA current count (DCC) registers with a new count. If bit 2 of the DCH register is set, a write to the DMA count register affects only the DBC register. The DBC register holds the initial count value until a new count is specified. If autoinitialization is enabled, this value is transferred to the DCC register when a terminal count or $\overline{END}$ condition occurs. For each DMA transfer, the current count register is decremented by one. The format of the DMA count register is shown below. The count value loaded into the DBC/DCC registers is one less than the desired transfer count.

2H, IN/OUT

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| $C_7$ | $C_6$ | $C_5$ | $C_4$ | $C_3$ | $C_2$ | $C_1$ | $C_0$ |

3H, IN/OUT

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| $C_{15}$ | $C_{14}$ | $C_{13}$ | $C_{12}$ | $C_{11}$ | $C_{10}$ | $C_9$ | $C_8$ |

**Address Register.** Use either byte or word I/O instructions with the lower two bytes (4H and 5H) of the DMA address register. However, byte I/O instructions must be used to access the high-order byte (6H) of this register. When bit 2 of the channel register is cleared, a write to the DMA address register updates both the DMA base address (DBA) and the DMA current address (DCA) registers with the new address. If bit 2 of the DCH register is set, a write to the DMA address register affects only the DBA register.

4H, IN/OUT

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| $A_7$ | $A_6$ | $A_5$ | $A_4$ | $A_3$ | $A_2$ | $A_1$ | $A_0$ |

5H, IN/OUT

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| $A_{15}$ | $A_{14}$ | $A_{13}$ | $A_{12}$ | $A_{11}$ | $A_{10}$ | $A_9$ | $A_8$ |

6H, IN/OUT (Byte only)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | $A_{19}$ | $A_{18}$ | $A_{17}$ | $A_{16}$ |

3c

The DBA register holds the starting address value until a new address is specified. This value is transferred to the DCA register automatically if autoinitialization is selected. For each DMA transfer, the current address register is incremented/decremented by one.

**Device Control Register.** The DMA device control (DDC) register (figure 40) is used to to program the DMA transfer characteristics common to all DMA channels. It controls the bus mode, write timing, priority logic, and enable/disable of the DMAU.

**Status Register.** The DMA status (DST) register (figure 41) contains information about the current state of each DMA channel. Software can determine if a termination condition has been reached ($TC_3$-$TC_0$) or if a DMA service request is present ($RQ_3$-$RQ_0$). The byte IN instruction must be used to read this register.

***Figure 37. Bus Waiting Operation***

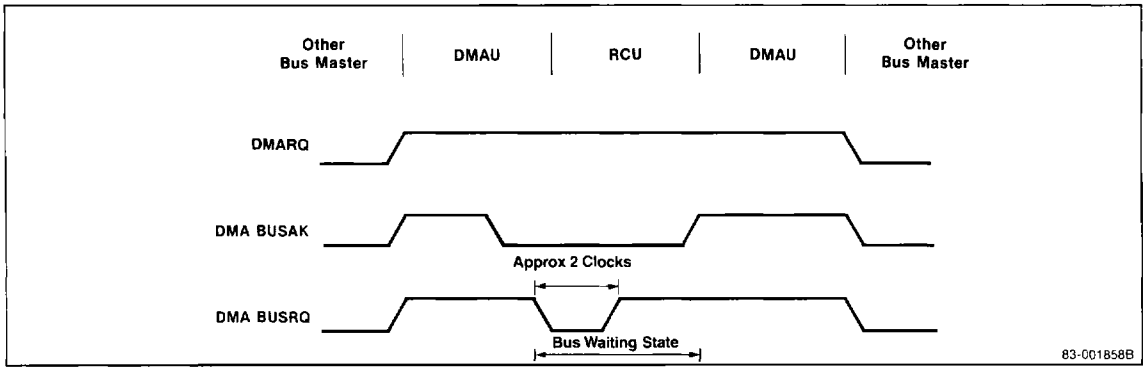
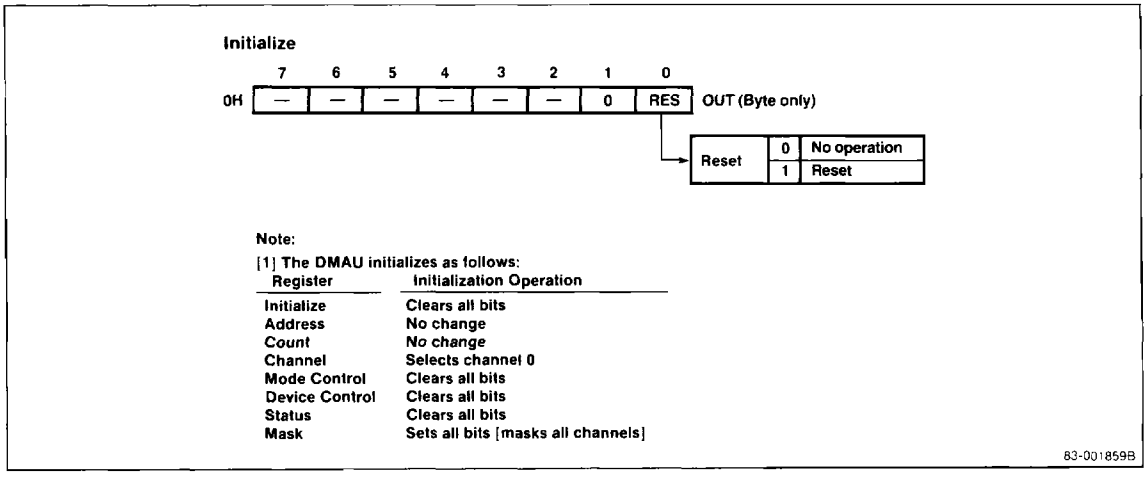



***Figure 38. DMA Initialize Command Register***




Note:

[1] The DMAU initializes as follows:

| Register | Initialization Operation |
|---|---|
| Initialize | Clears all bits |
| Address | No change |
| Count | No change |
| Channel | Selects channel 0 |
| Mode Control | Clears all bits |
| Device Control | Clears all bits |
| Status | Clears all bits |
| Mask | Sets all bits [masks all channels] |

*Figure 39. DMA Channel Register*

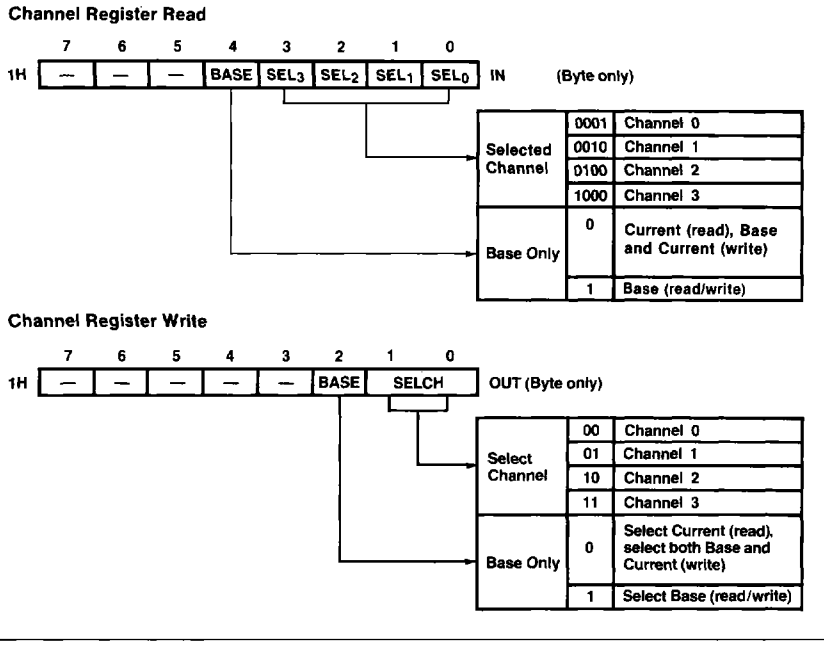
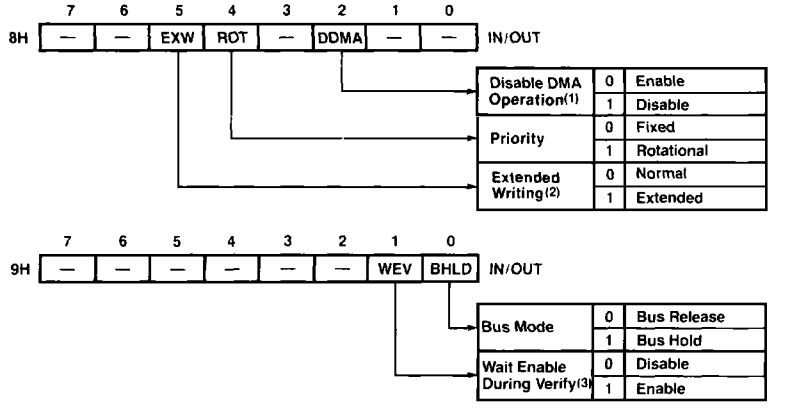



*Figure 40. DMA Device Control Register*




Note:

[1] Disables BUSRQ to the BAU to prevent incorrect DMA operation while the DMAU registers are being initialized or modified.

[2] When EXW is 0, the write signal becomes active [normal write] during T3 and TW [see timing waveforms]. When 1, the write signal becomes active during T2, T3, and TW [like the read signal].

[3] Wait states are generated by the READY signal during a verify transfer.

83-001863B

3c

**Mode Control Register.** The DMA mode (DMD) register (figure 42) selects the operating mode for each DMA channel. The DCH register selects which DMD register will be accessed. A byte IN/OUT instruction must be used to access this register.

**Mask Register Read/Write.** The DMA mask (DMK) register (figure 43) allows software to individually enable and disable DMA channels. The DMK register can only be accessed via byte I/O instructions.

***Figure 41. DMA Status Register***

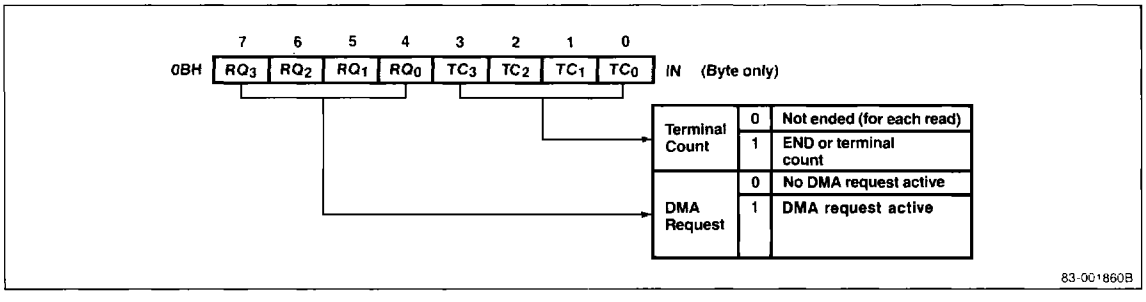



***Figure 42. DMA Mode Register***

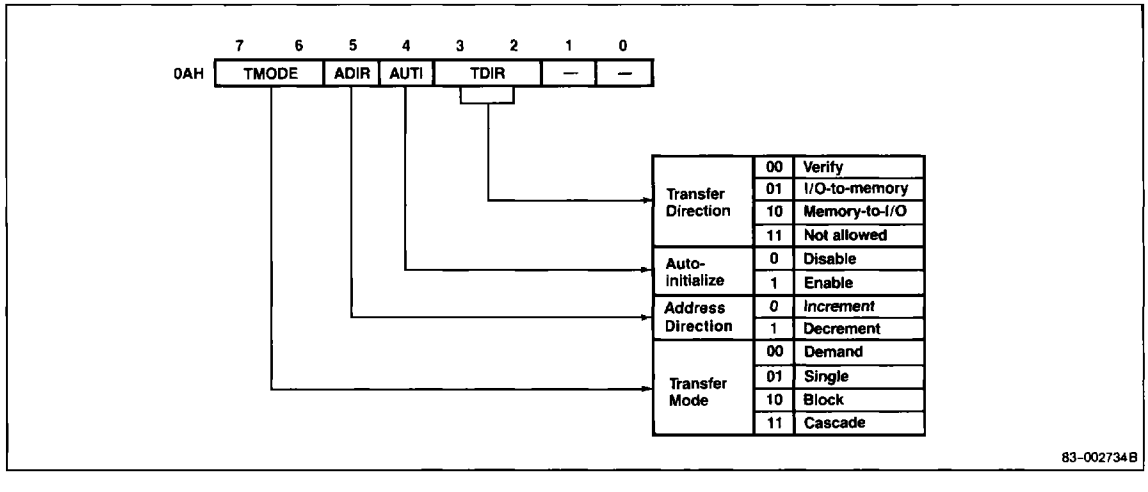
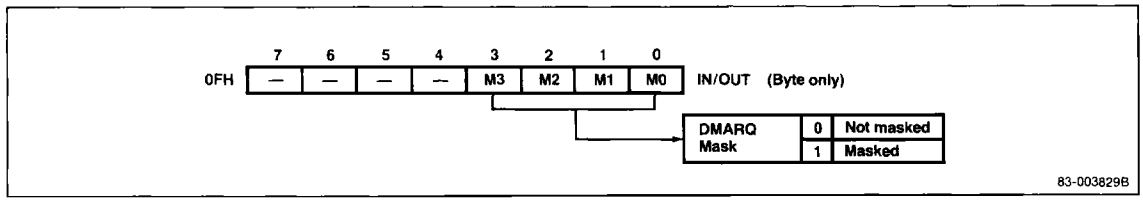



***Figure 43. DMA Mask Register***



## Reset

The falling edge of the $\overline{RESET}$ signal resets the μPD70208. The signal must be held low for at least four clock cycles to be recognized as valid.

| CPU Reset State<br>Register | Reset Value |
|---|---|
| PFP | 0000H |
| PC | 0000H |
| PS | FFFFH |
| SS | 0000H |
| DS0 | 0000H |
| DS1 | 0000H |
| PSW | F002H |
| AW, BW, CW, DW,<br>IX, IY, BP, SP | Undefined |
| Instruction queue | Cleared |

When RESET returns to the high level, the CPU will start fetching instructions from physical address FFFF0H.

### Internal Peripheral Registers

Internal peripheral devices initialized on reset are listed in the following table. I/O devices not listed are not initialized on reset and must be initialized by software.

| | Register | Reset Value |
|---|---|---|
| System<br>I/O area | OPCN | ----0000 |
| | OPSEL | ----0000 |
| | WCY1 | 11111111 |
| | WCY2 | ----1111 |
| | WMB | -111-111 |
| | TCKS | ---00000 |
| | RFC | x--01000 |
| SCU | SMD | 01001011 |
| | SCM | --0000-0 |
| | SIMK | ------11 |
| | SST | 10000100 |
| | DCH | ---00001 |
| | DMD | 000000-0 |
| DMAU | DDC (low) | --00-0-- |
| | DDC (high) | ------00 |
| | DST | xxxx0000 |
| | DMK | ----1111 |

**Symbols:** x = unaffected; 0 = cleared; 1 = set; (-) = unused.

### Output Pin Status

The following table lists output pin status during reset.

| Signal | Status |
|---|---|
| $\overline{INTAK}$, $\overline{BUFEN}$, $\overline{BUFR/W}$, $\overline{MRD}$, $\overline{MWR}$, $\overline{END/TC}$, $\overline{IOWR}$, $\overline{IORD}$, $\overline{REFRQ}$, $BS_2$-$BS_0$, $\overline{BUSLOCK}$, RESOUT, $\overline{DMAAK3}$-$\overline{DMAAK0}$ | High level |
| $QS_1$-$QS_0$, ASTB, HLDAK | Low level |
| $A_{19}$-$A_{16}$/$PS_3$-$PS_0$, TOUT2 | High or low level |
| $A_{15}$-$A_8$ | High or low level |
| $AD_7$-$AD_0$ | High impedance |
| CLKOUT | Continues to supply clock |

3c

## Instruction Set

### Symbols

Preceding the instruction set, several tables explain symbols, abbreviations, and codes.

### Clocks

In the Clocks column of the instruction set, the numbers cover these operations: instruction decoding, effective address calculation, operand fetch, and instruction execution.

Clock timings assume the instruction has been prefetched and is present in the four-byte instruction queue. Otherwise, add four clocks for each byte not present.

For instructions that reference memory operands, the number on the left side of the slash (/) is for byte operands and the number on the right side is for word operands.

For conditional control transfer or branch instructions, the number on the left side of the slash is applicable if the transfer or branch takes place. The number on the right side is applicable if it does not take place.

If a range of numbers is given, the execution time depends on the operands involved.

***Symbols***

| Symbol | Meaning |
|---|---|
| acc | Accumulator (AW or AL) |
| disp | Displacement (8 or 16 bits) |
| dmem | Direct memory address |
| dst | Destination operand or address |
| ext-disp8 | 16-bit displacement (sign-extension byte + 8-bit displacement) |
| far_label | Label within a different program segment |
| far_proc | Procedure within a different program segment |
| fp_op | Floating point instruction operation |
| imm | 8- or 16-bit immediate operand |
| imm3/4 | 3- or 4-bit immediate bit offset |
| imm8 | 8-bit immediate operand |
| imm16 | 16-bit immediate operand |
| mem | Memory field (000 to 111); 8- or 16-bit memory location |

***Symbols***

| Symbol | Meaning |
|---|---|
| mem8 | 8-bit memory location |
| mem16 | 16-bit memory location |
| mem32 | 32-bit memory location |
| memptr16 | Word containing the destination address within the current segment |
| memptr32 | Double word containing a destination address in another segment |
| mod | Mode field (00 to 10) |
| near_label | Label within the current segment |
| near_proc | Procedure within the current segment |
| offset | Immediate offset data (16 bits) |
| pop_value | Number of bytes to discard from the stack |
| reg | Register field (000 to 111); 8- or 16-bit general-purpose register |
| reg8 | 8-bit general-purpose register |
| reg16 | 16-bit general-purpose register |
| regptr | 16-bit register containing a destination address within the current segment |
| regptr16 | Register containing a destination address within the current segment |
| seg | Immediate segment data (16 bits) |
| short_label | Label between −128 and +127 bytes from the end of the current instruction |
| sr | Segment register |
| src | Source operand or address |
| temp | Temporary register (8/16/32 bits) |
| AC | Auxiliary carry flag |
| AH | Accumulator (high byte) |
| AL | Accumulator (low byte) |
| AW | Accumulator (16 bits) |
| BH | BW register (high byte) |
| BL | BW register (low byte) |
| BP | BP register |
| BRK | Break flag |
| BW | BW register (16 bits) |
| CH | CW register (high byte) |
| CL | CW register (low byte) |
| CW | CW register (16 bits) |
| CY | Carry flag |
| DH | DW register (high byte) |
| DIR | Direction flag |
| DL | DW register (low byte) |

### Symbols (cont)

| Symbol | Meaning |
|---|---|
| DS0 | Data segment 0 register (16 bits) |
| DS1 | Data segment 1 register (16 bits) |
| DW | DW register (16 bits) |
| IE | Interrupt enable flag |
| IX | Index register (source) (16 bits) |
| IY | Index register (destination) (16 bits) |
| MD | Mode flag |
| P | Parity flag |
| PC | Program counter (16 bits) |
| PS | Program segment register (16 bits) |
| PSW | Program status word (16 bits) |
| R | Register set |
| S | Sign extend operand field<br>S = 0 No sign extension<br>S = 1 Sign extend immediate byte operand |
| S | Sign flag |
| SP | Stack pointer (16 bits) |
| SS | Stack segment register (16 bits) |
| V | Overflow flag |
| W | Word/byte field (0 to 1) |
| X, XXX, YYY, ZZZ | Data to identify the instruction code of the external floating point arithmetic chip |
| XOR | Exclusive logical sum |
| XXH | Two-digit hexadecimal value |
| XXXXH | Four-digit hexadecimal value |
| Z | Zero flag |

### Flag Operations

| Symbol | Meaning |
|---|---|
| (blank) | No change |
| 0 | Cleared to 0 |
| 1 | Set to 1 |
| x | Set or cleared according to result |
| u | Undefined |
| R | Restored to previous state |

### Memory Addressing Modes

| mem | mod = 00 | mod = 01 | mod = 10 |
|---|---|---|---|
| 000 | BW + IX | BW + IX + disp8 | BW + IX + disp16 |
| 001 | BW + IY | BW + IY + disp8 | BW + IY + disp16 |
| 010 | BP + IX | BP + IX + disp8 | BP + IX + disp16 |
| 011 | BP + IY | BP + IY + disp8 | BP + IY + disp16 |
| 100 | IX | IX + disp8 | IX + disp16 |
| 101 | IY | IY + disp8 | IY + disp16 |
| 110 | Direct | BP + disp8 | BP + disp16 |
| 111 | BW | BW + disp8 | BW + disp16 |

### Register Selection (mod = 11)

| reg | W = 0 | W = 1 |
|---|---|---|
| 000 | AL | AW |
| 001 | CL | CW |
| 010 | DL | DW |
| 011 | BL | BW |
| 100 | AH | SP |
| 101 | CH | BP |
| 110 | DH | IX |
| 111 | BH | IY |

### Segment Register Selection

| sr | Segment Register |
|---|---|
| 00 | DS1 |
| 01 | PS |
| 10 | SS |
| 11 | DS0 |

## Instruction Set

| Mnemonic | Operand | Opcode | | | | | | | | | | | | | | | | Clocks | Bytes | Flags | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | AC | CY | V | P | S | Z |
| ***Data Transfer Instructions*** | | | | | | | | | | | | | | | | | | | | | | | | | |
| MOV | reg, reg | 1 | 0 | 0 | 0 | 1 | 0 | 1 | W | 1 | 1 | | reg | | | reg | | 2 | 2 | | | | | | |
| | mem, reg | 1 | 0 | 0 | 0 | 1 | 0 | 0 | W | mod | | | reg | | | mem | | 7/11 | 2-4 | | | | | | |
| | reg, mem | 1 | 0 | 0 | 0 | 1 | 0 | 1 | W | mod | | | reg | | | mem | | 10/14 | 2-4 | | | | | | |
| | mem, imm | 1 | 1 | 0 | 0 | 0 | 1 | 1 | W | mod | | | reg | | | mem | | 9/13 | 3-6 | | | | | | |
| | reg, imm | 1 | 0 | 1 | 1 | W | | reg | | | | | | | | | | 4 | 2-3 | | | | | | |
| | acc, dmem | 1 | 0 | 1 | 0 | 0 | 0 | 0 | W | | | | | | | | | 10/14 | 3 | | | | | | |
| | dmem, acc | 1 | 0 | 1 | 0 | 0 | 0 | 1 | W | | | | | | | | | 9/13 | 3 | | | | | | |
| | sr, reg16 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | sr | | | reg | | 2 | 2 | | | | | | |
| | sr, mem16 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | mod | | 0 | sr | | | mem | | 14 | 2-4 | | | | | | |
| | reg16, sr | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | sr | | | reg | | 2 | 2 | | | | | | |
| | mem16, sr | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | mod | | 0 | sr | | | mem | | 12 | 2-4 | | | | | | |
| | DS0, reg16, mem32 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | mod | | | reg | | | mem | | 25 | 2-4 | | | | | | |
| | DS1, reg16, mem32 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | mod | | | reg | | | mem | | 25 | 2-4 | | | | | | |
| | AH, PSW | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | 2 | 1 | | | | | | |
| | PSW, AH | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | | | | | | | | | 3 | 1 | x | x | | x | x | x |
| LDEA | reg16, mem16 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | mod | | | reg | | | mem | | 4 | 2-4 | | | | | | |
| TRANS | src_table | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | | | | | | | | | 9 | 1 | | | | | | |
| XCH | reg, reg | 1 | 0 | 0 | 0 | 0 | 1 | 1 | W | 1 | 1 | | reg | | | reg | | 3 | 2 | | | | | | |
| | mem, reg | 1 | 0 | 0 | 0 | 0 | 1 | 1 | W | mod | | | reg | | | mem | | 13/21 | 2-4 | | | | | | |
| | AW, reg16 | 1 | 0 | 0 | 1 | 0 | | reg | | | | | | | | | | 3 | 1 | | | | | | |
| ***Repeat Prefixes*** | | | | | | | | | | | | | | | | | | | | | | | | | |
| REPC | | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | | | | | | | | | 2 | 1 | | | | | | |
| REPNC | | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | | | | | | | | | 2 | 1 | | | | | | |
| REP<br>REPE<br>REPZ | | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | | | | | | | | | 2 | 1 | | | | | | |
| REPNE<br>REPNZ | | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | | | | | | | | | 2 | 1 | | | | | | |
| ***Block Transfer Instructions*** | | | | | | | | | | | | | | | | | | | | | | | | | |
| MOVBK | dst, src | 1 | 0 | 1 | 0 | 0 | 1 | 0 | W | | | | | | | | | 1<br>9 (9) + 8n (W = 0)<br>9 (17) + 16n (W = 1) | | | | | | | |
| CMPBK | dst, src | 1 | 0 | 1 | 0 | 0 | 1 | 1 | W | | | | | | | | | 1<br>7 (13) + 14n (W = 0)<br>7 (21) + 22n (W = 1) | | x | x | x | x | x | x |
| CMPM | dst | 1 | 0 | 1 | 0 | 1 | 1 | 1 | W | | | | | | | | | 1<br>7 (7) + 10n (W = 0)<br>7 (11) + 14n (W = 1) | | x | x | x | x | x | x |
| LDM | src | 1 | 0 | 1 | 0 | 1 | 1 | 0 | W | | | | | | | | | 1<br>7 (7) + 9n (W = 0)<br>7 (11) + 13n (W = 1) | | | | | | | |
| STM | dst | 1 | 0 | 1 | 0 | 1 | 0 | 1 | W | | | | | | | | | 1<br>5 (5) + 4n (W = 0)<br>5 (9) + 8n (W = 1) | | | | | | | |

n = number of transfers

String instruction execution clocks for a single instruction execution are in parentheses.

### Instruction Set (cont)

| Mnemonic | Operand | Opcode 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Clocks | Bytes | Flags AC | CY | V | P | S | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***I/O Instructions*** | | | | | | | | | | | | | | | | | | | | | | | | | |
| IN | acc, imm8 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | W | | | | | | | | | 9/13 | 2 | | | | | | |
| | acc, DW | 1 | 1 | 1 | 0 | 1 | 1 | 0 | W | | | | | | | | | 8/12 | 1 | | | | | | |
| OUT | imm8, acc | 1 | 1 | 1 | 0 | 0 | 1 | 1 | W | | | | | | | | | 8/12 | 2 | | | | | | |
| | DW, acc | 1 | 1 | 1 | 0 | 1 | 1 | 1 | W | | | | | | | | | 8/12 | 1 | | | | | | |
| INM | dst, DW | 0 | 1 | 1 | 0 | 1 | 1 | 0 | W | | | | | | | | | 9 (10) + 8n (W = 0)<br>9 (18) + 16n (W = 1) | 1 | | | | | | |
| OUTM | DW, src | 0 | 1 | 1 | 0 | 1 | 1 | 1 | W | | | | | | | | | 9 (10) + 8n (W = 0)<br>9 (18) + 16n (W = 1) | 1 | | | | | | |
| | | n = number of transfers<br>String instruction execution clocks for a single instruction execution are in parentheses. | | | | | | | | | | | | | | | | | | | | | | | |
| ***BCD Instructions*** | | | | | | | | | | | | | | | | | | | | | | | | | |
| ADJBA | | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | | | | | | | | | 7 | 1 | x | x | u | u | u | u |
| ADJ4A | | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | | | | | | | | | 3 | 1 | x | x | u | x | x | x |
| ADJBS | | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | 7 | 1 | x | x | u | u | u | u |
| ADJ4S | | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | | | | | | | | | 3 | 1 | x | x | u | x | x | x |
| ADD4S | dst, src | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 7 + 19n | 2 | u | x | u | u | u | x |
| SUB4S | dst, src | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 7 + 19n | 2 | u | x | u | u | u | x |
| CMP4S | dst, src | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 7 + 19n | 2 | u | x | u | u | u | x |
| ROL4 | reg8 | 0<br>1 | 0<br>1 | 0<br>0 | 0<br>0 | 1<br>0 | 1 | 1<br>reg | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 13 | 3 | | | | | | |
| | mem8 | 0<br>mod | 0 | 0 | 0 | 1<br>0 | 1 | 1<br>mem | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 25 | 3-5 | | | | | | |
| ROR4 | reg8 | 0<br>1 | 0<br>1 | 0<br>0 | 0<br>0 | 1<br>0 | 1 | 1<br>reg | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 17 | 3 | | | | | | |
| | mem8 | 0<br>mod | 0 | 0 | 0 | 1<br>0 | 1 | 1<br>mem | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 29 | 3-5 | | | | | | |
| | | n = number of BCD digits divided by 2 | | | | | | | | | | | | | | | | | | | | | | | |
| ***Data Type Conversion Instructions*** | | | | | | | | | | | | | | | | | | | | | | | | | |
| CVTBD | | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 15 | 2 | u | u | u | x | x | x |
| CVTDB | | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 7 | 2 | u | u | u | x | x | x |
| CVTBW | | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | | | | | | | | | 2 | 1 | | | | | | |
| CVTWL | | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | | | | | | | | | 4/5 | 1 | | | | | | |
| ***Arithmetic Instructions*** | | | | | | | | | | | | | | | | | | | | | | | | | |
| ADD | reg, reg | 0 | 0 | 0 | 0 | 0 | 0 | 1 | W | 1 | 1 | reg | | | reg | | | 2 | 2 | x | x | x | x | x | x |
| | mem, reg | 0 | 0 | 0 | 0 | 0 | 0 | 0 | W | mod | | reg | | | mem | | | 13/21 | 2-4 | x | x | x | x | x | x |
| | reg, mem | 0 | 0 | 0 | 0 | 0 | 0 | 1 | W | mod | | reg | | | mem | | | 10/14 | 2-4 | x | x | x | x | x | x |
| | reg, imm | 1 | 0 | 0 | 0 | 0 | 0 | S | W | 1 | 1 | 0 | 0 | 0 | reg | | | 4 | 3-4 | x | x | x | x | x | x |
| | mem, imm | 1 | 0 | 0 | 0 | 0 | 0 | S | W | mod | | 0 | 0 | 0 | mem | | | 15/23 | 3-6 | x | x | x | x | x | x |
| | acc, imm | 0 | 0 | 0 | 0 | 0 | 1 | 0 | W | | | | | | | | | 4 | 2-3 | x | x | x | x | x | x |

3c

### Instruction Set (cont)

| Mnemonic | Operand | Opcode 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Clocks | Bytes | Flags AC | CY | V | P | S | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***Arithmetic Instructions (cont)*** | | | | | | | | | | | | | | | | | | | | | | | | | |
| ADDC | reg, reg | 0 | 0 | 0 | 1 | 0 | 0 | 1 | W | 1 | 1 | | reg | | | reg | | 2 | 2 | x | x | x | x | x | x |
| | mem, reg | 0 | 0 | 0 | 1 | 0 | 0 | 0 | W | mod | | | reg | | | mem | | 13/21 | 2-4 | x | x | x | x | x | x |
| | reg, mem | 0 | 0 | 0 | 1 | 0 | 0 | 1 | W | mod | | | reg | | | mem | | 10/14 | 2-4 | x | x | x | x | x | x |
| | reg, imm | 1 | 0 | 0 | 0 | 0 | 0 | S | W | 1 | 1 | 0 | 1 | 0 | | reg | | 4 | 3-4 | x | x | x | x | x | x |
| | mem, imm | 1 | 0 | 0 | 0 | 0 | 0 | S | W | mod | | 0 | 1 | 0 | | mem | | 15/23 | 3-6 | x | x | x | x | x | x |
| | acc, imm | 0 | 0 | 0 | 1 | 0 | 1 | 0 | W | | | | | | | | | 4 | 2-3 | x | x | x | x | x | x |
| SUB | reg, reg | 0 | 0 | 1 | 0 | 1 | 0 | 1 | W | 1 | 1 | | reg | | | reg | | 2 | 2 | x | x | x | x | x | x |
| | mem, reg | 0 | 0 | 1 | 0 | 1 | 0 | 0 | W | mod | | | reg | | | mem | | 13/21 | 2-4 | x | x | x | x | x | x |
| | reg, mem | 0 | 0 | 1 | 0 | 1 | 0 | 1 | W | mod | | | reg | | | mem | | 10/14 | 2-4 | x | x | x | x | x | x |
| | reg, imm | 1 | 0 | 0 | 0 | 0 | 0 | S | W | 1 | 1 | 1 | 0 | 1 | | reg | | 4 | 3-4 | x | x | x | x | x | x |
| | mem, imm | 1 | 0 | 0 | 0 | 0 | 0 | S | W | mod | | 1 | 0 | 1 | | mem | | 15/23 | 3-6 | x | x | x | x | x | x |
| | acc, imm | 0 | 0 | 1 | 0 | 1 | 1 | 0 | W | | | | | | | | | 4 | 2-3 | x | x | x | x | x | x |
| SUBC | reg, reg | 0 | 0 | 0 | 1 | 1 | 0 | 1 | W | 1 | 1 | | reg | | | reg | | 2 | 2 | x | x | x | x | x | x |
| | mem, reg | 0 | 0 | 0 | 1 | 1 | 0 | 0 | W | mod | | | reg | | | mem | | 13/21 | 2-4 | x | x | x | x | x | x |
| | reg, mem | 0 | 0 | 0 | 1 | 1 | 0 | 1 | W | mod | | | reg | | | mem | | 10/14 | 2-4 | x | x | x | x | x | x |
| | reg, imm | 1 | 0 | 0 | 0 | 0 | 0 | S | W | 1 | 1 | 0 | 1 | 1 | | reg | | 4 | 3-4 | x | x | x | x | x | x |
| | mem, imm | 1 | 0 | 0 | 0 | 0 | 0 | S | W | mod | | 0 | 1 | 1 | | mem | | 15/23 | 3-6 | x | x | x | x | x | x |
| | acc, imm | 0 | 0 | 0 | 1 | 1 | 1 | 0 | W | | | | | | | | | 4 | 2-3 | x | x | x | x | x | x |
| INC | reg8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | | reg | | 2 | 2 | x | | x | x | x | x |
| | mem | 1 | 1 | 1 | 1 | 1 | 1 | 1 | W | mod | | 0 | 0 | 0 | | mem | | 13/21 | 2-4 | x | | x | x | x | x |
| | reg16 | 0 | 1 | 0 | 0 | 0 | | reg | | | | | | | | | | 2 | 1 | x | | x | x | x | x |
| DEC | reg8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | | reg | | 2 | 2 | x | | x | x | x | x |
| | mem | 1 | 1 | 1 | 1 | 1 | 1 | 1 | W | mod | | 0 | 0 | 1 | | mem | | 13/21 | 2-4 | x | | x | x | x | x |
| | reg16 | 0 | 1 | 0 | 0 | 1 | | reg | | | | | | | | | | 2 | 1 | x | | x | x | x | x |
| MULU | reg | 1 | 1 | 1 | 1 | 0 | 1 | 1 | W | 1 | 1 | 1 | 0 | 0 | | reg | | 21-30 | 2 | u | x | x | u | u | u |
| | mem | 1 | 1 | 1 | 1 | 0 | 1 | 1 | W | mod | | 1 | 0 | 0 | | mem | | 26-39 | 2-4 | u | x | x | u | u | u |
| MUL | reg | 1 | 1 | 1 | 1 | 0 | 1 | 1 | W | 1 | 1 | 1 | 0 | 1 | | reg | | 33-47 | 2 | u | x | x | u | u | u |
| | mem | 1 | 1 | 1 | 1 | 0 | 1 | 1 | W | mod | | 1 | 0 | 1 | | mem | | 38-56 | 2-4 | u | x | x | u | u | u |
| | reg16,reg16,imm8 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | | reg | | | reg | | 28-34 | 3 | u | x | x | u | u | u |
| | reg16,mem16,imm8 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | mod | | | reg | | | mem | | 37-43 | 3-5 | u | x | x | u | u | u |
| | reg16,reg16,imm16 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | | reg | | | reg | | 36-42 | 4 | u | x | x | u | u | u |
| | reg16,mem16,imm16 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | mod | | | reg | | | mem | | 45-51 | 4-6 | u | x | x | u | u | u |
| DIVU | reg | 1 | 1 | 1 | 1 | 0 | 1 | 1 | W | 1 | 1 | 1 | 1 | 0 | | reg | | 19-25 | 2 | u | u | u | u | u | u |
| | mem | 1 | 1 | 1 | 1 | 0 | 1 | 1 | W | mod | | 1 | 1 | 0 | | mem | | 24-34 | 2-4 | u | u | u | u | u | u |
| DIV | reg | 1 | 1 | 1 | 1 | 0 | 1 | 1 | W | 1 | 1 | 1 | 1 | 1 | | reg | | 29-43 | 2 | u | u | u | u | u | u |
| | mem | 1 | 1 | 1 | 1 | 0 | 1 | 1 | W | mod | | 1 | 1 | 1 | | mem | | 34-52 | 2-4 | u | u | u | u | u | u |

*Instruction Set (cont)*

| Mnemonic | Operand | Opcode 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Clocks | Bytes | Flags AC | CY | V | P | S | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***Comparison Instructions*** | | | | | | | | | | | | | | | | | | | | | | | | | |
| CMP | reg, reg | 0 | 0 | 1 | 1 | 1 | 0 | 1 | W | 1 | 1 | | reg | | | reg | | 2 | 2 | x | x | x | x | x | x |
| | mem, reg | 0 | 0 | 1 | 1 | 1 | 0 | 0 | W | mod | | | reg | | | mem | | 10/14 | 2-4 | x | x | x | x | x | x |
| | reg, mem | 0 | 0 | 1 | 1 | 1 | 0 | 1 | W | mod | | | reg | | | mem | | 10/14 | 2-4 | x | x | x | x | x | x |
| | reg, imm | 1 | 0 | 0 | 0 | 0 | 0 | S | W | 1 | 1 | 1 | 1 | 1 | | reg | | 4 | 3-4 | x | x | x | x | x | x |
| | mem, imm | 1 | 0 | 0 | 0 | 0 | 0 | S | W | mod | | 1 | 1 | 1 | | mem | | 12/16 | 3-6 | x | x | x | x | x | x |
| | acc, imm | 0 | 0 | 1 | 1 | 1 | 1 | 0 | W | | | | | | | | | 4 | 2-3 | x | x | x | x | x | x |
| ***Logical Instructions*** | | | | | | | | | | | | | | | | | | | | | | | | | |
| NOT | reg | 1 | 1 | 1 | 1 | 0 | 1 | 1 | W | 1 | 1 | 0 | 1 | 0 | | reg | | 2 | 2 | | | | | | |
| | mem | 1 | 1 | 1 | 1 | 0 | 1 | 1 | W | mod | | 0 | 1 | 0 | | mem | | 13/21 | 2-4 | | | | | | |
| NEG | reg | 1 | 1 | 1 | 1 | 0 | 1 | 1 | W | 1 | 1 | 0 | 1 | 1 | | reg | | 2 | 2 | x | x | x | x | x | x |
| | mem | 1 | 1 | 1 | 1 | 0 | 1 | 1 | W | mod | | 0 | 1 | 1 | | mem | | 13/21 | 2-4 | x | x | x | x | x | x |
| TEST | reg, reg | 1 | 0 | 0 | 0 | 0 | 1 | 0 | W | 1 | 1 | | reg | | | reg | | 2 | 2 | u | 0 | 0 | x | x | x |
| | mem, reg | 1 | 0 | 0 | 0 | 0 | 1 | 0 | W | mod | | | reg | | | mem | | 9/13 | 2-4 | u | 0 | 0 | x | x | x |
| | reg, imm | 1 | 1 | 1 | 1 | 0 | 1 | 1 | W | 1 | 1 | 0 | 0 | 0 | | reg | | 4 | 3-4 | u | 0 | 0 | x | x | x |
| | mem, imm | 1 | 1 | 1 | 1 | 0 | 1 | 1 | W | mod | | 0 | 0 | 0 | | mem | | 10/14 | 3-6 | u | 0 | 0 | x | x | x |
| | acc, imm | 1 | 0 | 1 | 0 | 1 | 0 | 0 | W | | | | | | | | | 4 | 2-3 | u | 0 | 0 | x | x | x |
| AND | reg, reg | 0 | 0 | 1 | 0 | 0 | 0 | 1 | W | 1 | 1 | | reg | | | reg | | 2 | 2 | u | 0 | 0 | x | x | x |
| | mem, reg | 0 | 0 | 1 | 0 | 0 | 0 | 0 | W | mod | | | reg | | | mem | | 13/21 | 2-4 | u | 0 | 0 | x | x | x |
| | reg, mem | 0 | 0 | 1 | 0 | 0 | 0 | 1 | W | mod | | | reg | | | mem | | 10/14 | 2-4 | u | 0 | 0 | x | x | x |
| | reg, imm | 1 | 0 | 0 | 0 | 0 | 0 | 0 | W | 1 | 1 | 1 | 0 | 0 | | reg | | 4 | 3-4 | u | 0 | 0 | x | x | x |
| | mem, imm | 1 | 0 | 0 | 0 | 0 | 0 | 0 | W | mod | | 1 | 0 | 0 | | mem | | 15/23 | 3-6 | u | 0 | 0 | x | x | x |
| | acc, imm | 0 | 0 | 1 | 0 | 0 | 1 | 0 | W | | | | | | | | | 4 | 2-3 | u | 0 | 0 | x | x | x |
| OR | reg, reg | 0 | 0 | 0 | 0 | 1 | 0 | 1 | W | 1 | 1 | | reg | | | reg | | 2 | 2 | u | 0 | 0 | x | x | x |
| | mem, reg | 0 | 0 | 0 | 0 | 1 | 0 | 0 | W | mod | | | reg | | | mem | | 13/21 | 2-4 | u | 0 | 0 | x | x | x |
| | reg, mem | 0 | 0 | 0 | 0 | 1 | 0 | 1 | W | mod | | | reg | | | mem | | 10/14 | 2-4 | u | 0 | 0 | x | x | x |
| | reg, imm | 1 | 0 | 0 | 0 | 0 | 0 | 0 | W | 1 | 1 | 0 | 0 | 1 | | reg | | 4 | 3-4 | u | 0 | 0 | x | x | x |
| | mem, imm | 1 | 0 | 0 | 0 | 0 | 0 | 0 | W | mod | | 0 | 0 | 1 | | mem | | 15/23 | 3-6 | u | 0 | 0 | x | x | x |
| | acc, imm | 0 | 0 | 0 | 0 | 1 | 1 | 0 | W | | | | | | | | | 4 | 2-3 | u | 0 | 0 | x | x | x |
| XOR | reg, reg | 0 | 0 | 1 | 1 | 0 | 0 | 1 | W | 1 | 1 | | reg | | | reg | | 2 | 2 | u | 0 | 0 | x | x | x |
| | mem, reg | 0 | 0 | 1 | 1 | 0 | 0 | 0 | W | mod | | | reg | | | mem | | 13/21 | 2-4 | u | 0 | 0 | x | x | x |
| | reg, mem | 0 | 0 | 1 | 1 | 0 | 0 | 1 | W | mod | | | reg | | | mem | | 10/14 | 2-4 | u | 0 | 0 | x | x | x |
| | reg, imm | 1 | 0 | 0 | 0 | 0 | 0 | 0 | W | 1 | 1 | 1 | 1 | 0 | | reg | | 4 | 3-4 | u | 0 | 0 | x | x | x |
| | mem, imm | 1 | 0 | 0 | 0 | 0 | 0 | 0 | W | mod | | 1 | 1 | 0 | | mem | | 15/23 | 3-6 | u | 0 | 0 | x | x | x |
| | acc, imm | 0 | 0 | 1 | 1 | 0 | 1 | 0 | W | | | | | | | | | 4 | 2-3 | u | 0 | 0 | x | x | x |

3c

*Instruction Set (cont)*

| Mnemonic | Operand | Opcode 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Clocks | Bytes | Flags AC | CY | V | P | S | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***Bit Manipulation Instructions*** | | | | | | | | | | | | | | | | | | | | | | | | | |
| INS | reg8, reg8 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 35-133 | 3 | | | | | | |
| | | 1 | 1 | | reg | | | | reg | | | | | | | | | | | | | | | | |
| | reg8, imm8 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 35-133 | 4 | | | | | | |
| | | 1 | 1 | 0 | 0 | 0 | | reg | | | | | | | | | | | | | | | | | |
| EXT | reg8, reg8 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 34-59 | 3 | | | | | | |
| | | 1 | 1 | | reg | | | | reg | | | | | | | | | | | | | | | | |
| | reg8, imm8 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 34-59 | 4 | | | | | | |
| | | 1 | 1 | 0 | 0 | 0 | | reg | | | | | | | | | | | | | | | | | |
| TEST1 | reg, CL | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | W | 3 | 3 | u | 0 | 0 | u | u | x |
| | | 1 | 1 | 0 | 0 | 0 | | reg | | | | | | | | | | | | | | | | | |
| | mem, CL | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | W | 7/11 | 3-5 | u | 0 | 0 | u | u | x |
| | | mod | | 0 | 0 | 0 | | mem | | | | | | | | | | | | | | | | | |
| | reg, imm3/4 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | W | 4 | 4 | u | 0 | 0 | u | u | x |
| | | 1 | 1 | 0 | 0 | 0 | | reg | | | | | | | | | | | | | | | | | |
| | mem, imm3/4 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | W | 8/12 | 4-6 | u | 0 | 0 | u | u | x |
| | | mod | | 0 | 0 | 0 | | mem | | | | | | | | | | | | | | | | | |
| SET1 | reg, CL | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | W | 4 | 3 | | | | | | |
| | | 1 | 1 | 0 | 0 | 0 | | reg | | | | | | | | | | | | | | | | | |
| | mem, CL | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | W | 10/18 | 3-5 | | | | | | |
| | | mod | | 0 | 0 | 0 | | mem | | | | | | | | | | | | | | | | | |
| | reg, imm3/4 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | W | 5 | 4 | | | | | | |
| | | 1 | 1 | 0 | 0 | 0 | | reg | | | | | | | | | | | | | | | | | |
| | mem, imm3/4 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | W | 11/19 | 4-6 | | | | | | |
| | | mod | | 0 | 0 | 0 | | mem | | | | | | | | | | | | | | | | | |
| | CY | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | | | | | | | | | 2 | 1 | | 1 | | | | |
| | DIR | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | | | | | | | | | 2 | 1 | | | | | | |
| CLR1 | reg, CL | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | W | 5 | 3 | | | | | | |
| | | 1 | 1 | 0 | 0 | 0 | | reg | | | | | | | | | | | | | | | | | |
| | mem, CL | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | W | 11/19 | 3-5 | | | | | | |
| | | mod | | 0 | 0 | 0 | | mem | | | | | | | | | | | | | | | | | |
| | reg, imm3/4 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | W | 6 | 4 | | | | | | |
| | | 1 | 1 | 0 | 0 | 0 | | reg | | | | | | | | | | | | | | | | | |
| | mem, imm3/4 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | W | 12/20 | 4-6 | | | | | | |
| | | mod | | 0 | 0 | 0 | | mem | | | | | | | | | | | | | | | | | |
| | CY | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | | | | | | | | | 2 | 1 | | 0 | | | | |
| | DIR | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | | | | | | | | | 2 | 1 | | | | | | |
| NOT1 | reg, CL | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | W | 4 | 3 | | | | | | |
| | | 1 | 1 | 0 | 0 | 0 | | reg | | | | | | | | | | | | | | | | | |
| | mem, CL | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | W | 10/18 | 3-5 | | | | | | |
| | | mod | | 0 | 0 | 0 | | mem | | | | | | | | | | | | | | | | | |
| | reg, imm3/4 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | W | 5 | 4 | | | | | | |
| | | 1 | 1 | 0 | 0 | 0 | | reg | | | | | | | | | | | | | | | | | |
| | mem, imm3/4 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | W | 11/19 | 4-6 | | | | | | |
| | | mod | | 0 | 0 | 0 | | mem | | | | | | | | | | | | | | | | | |
| | CY | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | | | | | | | | | 2 | 1 | | x | | | | |

*Instruction Set (cont)*

| Mnemonic | Operands | Opcode 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Clocks | Bytes | Flags AC | CY | V | P | S | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***Shift/Rotate Instructions*** | | | | | | | | | | | | | | | | | | | | | | | | | |
| SHL | reg, 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | W | 1 | 1 | 1 | 0 | 0 | | reg | | 2 | 2 | u | x | x | x | x | x |
| | mem, 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | W | mod | | | | | 1 | 0 | 0 | 13/21 | 2-4 | u | x | x | x | x | x |
| | reg, CL | 1 | 1 | 0 | 1 | 0 | 0 | 1 | W | 1 | 1 | 1 | 0 | 0 | | reg | | 7 + n | 2 | u | x | u | x | x | x |
| | mem, CL | 1 | 1 | 0 | 1 | 0 | 0 | 1 | W | mod | | | | | 1 | 0 | 0 | 16/24 + n | 2-4 | u | x | u | x | x | x |
| | reg, imm8 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | W | 1 | 1 | 1 | 0 | 0 | | reg | | 7 + n | 3 | u | x | u | x | x | x |
| | mem, imm8 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | W | mod | | | | | 1 | 0 | 0 | 16/24 + n | 3-5 | u | x | u | x | x | x |
| SHR | reg, 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | W | 1 | 1 | 1 | 0 | 1 | | reg | | 2 | 2 | u | x | x | x | x | x |
| | mem, 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | W | mod | | | | | 1 | 0 | 1 | 13/21 | 2-4 | u | x | x | x | x | x |
| | reg, CL | 1 | 1 | 0 | 1 | 0 | 0 | 1 | W | 1 | 1 | 1 | 0 | 1 | | reg | | 7 + n | 2 | u | x | u | x | x | x |
| | mem, CL | 1 | 1 | 0 | 1 | 0 | 0 | 1 | W | mod | | | | | 1 | 0 | 1 | 16/24 + n | 2-4 | u | x | u | x | x | x |
| | reg, imm8 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | W | 1 | 1 | 1 | 0 | 1 | | reg | | 7 + n | 3 | u | x | u | x | x | x |
| | mem, imm8 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | W | mod | | | | | 1 | 0 | 1 | 16/24 + n | 3-5 | u | x | u | x | x | x |
| SHRA | reg, 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | W | 1 | 1 | 1 | 1 | 1 | | reg | | 2 | 2 | u | x | 0 | x | x | x |
| | mem, 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | W | mod | | | | | 1 | 1 | 1 | 13/21 | 2-4 | u | x | 0 | x | x | x |
| | reg, CL | 1 | 1 | 0 | 1 | 0 | 0 | 1 | W | 1 | 1 | 1 | 1 | 1 | | reg | | 7 + n | 2 | u | x | u | x | x | x |
| | mem, CL | 1 | 1 | 0 | 1 | 0 | 0 | 1 | W | mod | | | | | 1 | 1 | 1 | 16/24 + n | 2-4 | u | x | u | x | x | x |
| | reg, imm8 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | W | 1 | 1 | 1 | 1 | 1 | | reg | | 7 + n | 3 | u | x | u | x | x | x |
| | mem, imm8 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | W | mod | | | | | 1 | 1 | 1 | 16/24 + n | 3-5 | u | x | u | x | x | x |
| ROL | reg, 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | W | 1 | 1 | 0 | 0 | 0 | | reg | | 2 | 2 | | x | x | | | |
| | mem, 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | W | mod | | | | | 0 | 0 | 0 | 13/21 | 2-4 | | x | x | | | |
| | reg, CL | 1 | 1 | 0 | 1 | 0 | 0 | 1 | W | 1 | 1 | 0 | 0 | 0 | | reg | | 7 + n | 2 | | x | u | | | |
| | mem, CL | 1 | 1 | 0 | 1 | 0 | 0 | 1 | W | mod | | | | | 0 | 0 | 0 | 16/24 + n | 2-4 | | x | u | | | |
| | reg, imm | 1 | 1 | 0 | 0 | 0 | 0 | 0 | W | 1 | 1 | 0 | 0 | 0 | | reg | | 7 + n | 3 | | x | u | | | |
| | mem, imm | 1 | 1 | 0 | 0 | 0 | 0 | 0 | W | mod | | | | | 0 | 0 | 0 | 16/24 + n | 3-5 | | x | u | | | |
| ROR | reg, 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | W | 1 | 1 | 0 | 0 | 1 | | reg | | 2 | 2 | | x | u | | | |
| | mem, 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | W | mod | | | | | 0 | 0 | 1 | 13/21 | 2-4 | | x | x | | | |
| | reg, CL | 1 | 1 | 0 | 1 | 0 | 0 | 1 | W | 1 | 1 | 0 | 0 | 1 | | reg | | 7 + n | 2 | | x | u | | | |
| | mem, CL | 1 | 1 | 0 | 1 | 0 | 0 | 1 | W | mod | | | | | 0 | 0 | 1 | 16/24 + n | 2-4 | | x | u | | | |
| | reg, imm8 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | W | 1 | 1 | 0 | 0 | 1 | | reg | | 7 + n | 3 | | x | u | | | |
| | mem, imm8 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | W | mod | | | | | 0 | 0 | 1 | 16/24 + n | 3-5 | | x | u | | | |
| ROLC | reg, 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | W | 1 | 1 | 0 | 1 | 0 | | reg | | 2 | 2 | | x | x | | | |
| | mem, 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | W | mod | | | | | 0 | 1 | 0 | 13/21 | 2-4 | | x | x | | | |
| | reg, CL | 1 | 1 | 0 | 1 | 0 | 0 | 1 | W | 1 | 1 | 0 | 1 | 0 | | reg | | 7 + n | 2 | | x | u | | | |
| | mem, CL | 1 | 1 | 0 | 1 | 0 | 0 | 1 | W | mod | | | | | 0 | 1 | 0 | 16/24 + n | 2-4 | | x | u | | | |
| | reg, imm8 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | W | 1 | 1 | 0 | 1 | 0 | | reg | | 7 + n | 3 | | x | u | | | |
| | mem, imm8 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | W | mod | | | | | 0 | 1 | 0 | 16/24 + n | 3-5 | | x | u | | | |

n = number of shifts

3c

### Instruction Set (cont)

| Mnemonic | Operands | Opcode 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Clocks | Bytes | Flags AC | CY | V | P | S | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Shift Rotate Instructions (cont)** | | | | | | | | | | | | | | | | | | | | | | | | | |
| RORC | reg, 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | W | 1 | 1 | 0 | 1 | 1 | reg | | | 2 | 2 | | x | x | | | |
| | mem, 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | W | mod | | 0 | 1 | 1 | mem | | | 13/21 | 2-4 | | x | x | | | |
| | reg, CL | 1 | 1 | 0 | 1 | 0 | 0 | 1 | W | 1 | 1 | 0 | 1 | 1 | reg | | | 7 + n | 2 | | x | u | | | |
| | mem, CL | 1 | 1 | 0 | 1 | 0 | 0 | 1 | W | mod | | 0 | 1 | 1 | mem | | | 16/24 + n | 2-4 | | x | u | | | |
| | reg, imm8 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | W | 1 | 1 | 0 | 1 | 1 | reg | | | 7 + n | 3 | | x | u | | | |
| | mem, imm8 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | W | mod | | 0 | 1 | 1 | mem | | | 16/24 + n | 3-5 | | x | u | | | |
| | | | | | | | | | | n = number of shifts | | | | | | | | | | | | | | | |
| **Stack Manipulation Instructions** | | | | | | | | | | | | | | | | | | | | | | | | | |
| PUSH | mem16 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | mod | | 1 | 1 | 0 | mem | | | 23 | 2-4 | | | | | | |
| | reg16 | 0 | 1 | 0 | 1 | 0 | reg | | | | | | | | | | | 10 | 1 | | | | | | |
| | sr | 0 | 0 | 0 | sr | | 1 | 1 | 0 | | | | | | | | | 10 | 1 | | | | | | |
| | PSW | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | | | | | | | | | 10 | 1 | | | | | | |
| | R | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | 65 | 1 | | | | | | |
| | imm | 0 | 1 | 1 | 0 | 1 | 0 | S | 0 | | | | | | | | | 9-10 | 2-3 | | | | | | |
| POP | mem16 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | mod | | 0 | 0 | 0 | mem | | | 25 | 2-4 | | | | | | |
| | reg16 | 0 | 1 | 0 | 1 | 1 | reg | | | | | | | | | | | 12 | 1 | | | | | | |
| | sr | 0 | 0 | 0 | sr | | 1 | 1 | 1 | | | | | | | | | 12 | 1 | | | | | | |
| | PSW | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | | | | | | | | | 12 | 1 | R | R | R | R | R | R |
| | R | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | | | | | | | | | 75 | 1 | | | | | | |
| PREPARE | imm16, imm8 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | *imm8 = 0 : 16<br>imm8 > 1 : 21 + 16 (imm8 − 1) | | | | | | | | * | 4 | | | | | | |
| DISPOSE | | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | | | | | | | | | 10 | 1 | | | | | | |
| **Control Transfer Instructions** | | | | | | | | | | | | | | | | | | | | | | | | | |
| CALL | near_proc | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | | | | | | | | | 20 | 3 | | | | | | |
| | regptr | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | reg | | | 18 | 1 | | | | | | |
| | memptr16 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | mod | | 0 | 1 | 0 | mem | | | 31 | 2-4 | | | | | | |
| | far_proc | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | | | | | | | | | 29 | 5 | | | | | | |
| | memptr32 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | mod | | 0 | 1 | 1 | mem | | | 47 | 2-4 | | | | | | |
| RET | | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | | | | 19 | 1 | | | | | | |
| | pop_value | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | | | | | | | | | 24 | 3 | | | | | | |
| | | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | | | | | | | | | 29 | 1 | | | | | | |
| | pop_value | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | | | | | | | | | 32 | 3 | | | | | | |
| BR | near_label | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | | | | | | | | | 13 | 3 | | | | | | |
| | short_label | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | | | | | | | | | 12 | 2 | | | | | | |
| | reg | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | reg | | | 11 | 2 | | | | | | |
| | memptr16 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | mod | | 1 | 0 | 0 | mem | | | 23 | 2-4 | | | | | | |
| | far_label | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | | | | | | | | | 15 | 5 | | | | | | |
| | memptr32 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | mod | | 1 | 0 | 1 | mem | | | 34 | 2-4 | | | | | | |
| BV | near_label | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | | | | | | | | | 14/4 | 2 | | | | | | |
| BNV | near_label | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | | | | | | | | | 14/4 | 2 | | | | | | |

### Instruction Set (cont)

| Mnemonic | Operands | Opcode 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Clocks | Bytes | Flags AC | CY | V | P | S | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***Control Transfer Instructions (cont)*** | | | | | | | | | | | | | | | | | | | | | | | | | |
| BC, BL | near_label | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | | | | | | | | | 14/4 | 2 | | | | | | |
| BNC, BNL | near_label | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | | | | | | | | | 14/4 | 2 | | | | | | |
| BE, BZ | near_label | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | | | | | | | | | 14/4 | 2 | | | | | | |
| BNE, BNZ | near_label | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | | | | | | | | | 14/4 | 2 | | | | | | |
| BNH | near_label | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | | | | | | | | | 14/4 | 2 | | | | | | |
| BH | near_label | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | | | | | | | | | 14/4 | 2 | | | | | | |
| BN | near_label | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | | | | | | | | | 14/4 | 2 | | | | | | |
| BP | near_label | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | | | | | | | | | 14/4 | 2 | | | | | | |
| BPE | near_label | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | | | | | | | | | 14/4 | 2 | | | | | | |
| BPO | near_label | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | | | | | | | | | 14/4 | 2 | | | | | | |
| BLT | near_label | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | | | | | | | | | 14/4 | 2 | | | | | | |
| BGE | near_label | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | | | | | | | | | 14/4 | 2 | | | | | | |
| BLE | near_label | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | | | | | | | | | 14/4 | 2 | | | | | | |
| BGT | near_label | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | 14/4 | 2 | | | | | | |
| DBNZNE | near_label | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | 14/5 | 2 | | | | | | |
| DBNZE | near_label | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | | | | | | | | | 14/5 | 2 | | | | | | |
| DBNZ | near_label | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | | | | | | | | | 13/5 | 2 | | | | | | |
| BCWZ | near_label | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | | | | | | | | | 13/5 | 2 | | | | | | |
| ***Interrupt Instructions*** | | | | | | | | | | | | | | | | | | | | | | | | | |
| BRK | 3 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | | | | | | | | | 50 | 1 | | | | | | |
| | imm8 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | | | | | | | | | 50 | 2 | | | | | | |
| BRKV | imm8 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | | | | | | | | | 52/3 | 1 | | | | | | |
| RETI | | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | | | | | | | | | 39 | 1 | R | R | R | R | R | R |
| CHKIND | reg16, mem32 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | mod | | | reg | | | | mem | 25/72-75 | 2-4 | | | | | | |
| BRKEM | imm8 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 50 | 3 | | | | | | |
| ***CPU Control Instructions*** | | | | | | | | | | | | | | | | | | | | | | | | | |
| HALT | | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | | | | | | | | | 2 | 1 | | | | | | |
| BUSLOCK | | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | | | | | | | | | 2 | 1 | | | | | | |
| FPO1 | fp_op | 1 | 1 | 0 | 1 | 1 | X | X | X | 1 | 1 | Y | Y | Y | Z | Z | Z | 2 | 2 | | | | | | |
| | fp_op, mem | 1 | 1 | 0 | 1 | 1 | X | X | X | mod | | Y | Y | Y | | mem | | 14 | 2-4 | | | | | | |
| FPO2 | fp_op | 0 | 1 | 1 | 0 | 0 | 1 | 1 | X | 1 | 1 | Y | Y | Y | Z | Z | Z | 2 | 2 | | | | | | |
| | fp_op, mem | 0 | 1 | 1 | 0 | 0 | 1 | 1 | X | mod | | Y | Y | Y | | mem | | 14 | 2-4 | | | | | | |
| POLL | | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | n = number of times POLL pin is sampled. | | | | | | | | 2 + 5n | 1 | | | | | | |
| NOP | | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | | | | | | | | | 3 | 1 | | | | | | |
| DI | | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | | | | | | | | | 2 | 1 | | | | | | |
| EI | | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | | | | | | | | | 2 | 1 | | | | | | |
| DS0:, DS1:, PS:, and SS: (segment override prefixes) | | 0 | 0 | 1 | seg | | 1 | 1 | 0 | | | | | | | | | 2 | 1 | | | | | | |
| ***8080 Instruction Set Enhancements*** | | | | | | | | | | | | | | | | | | | | | | | | | |
| RETEM | | 1 | 1 | 1 | 0 | | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 39 | 2 | R | R | R | R | R | R |
| CALLN | imm8 | 1 | 1 | 1 | 0 | | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 58 | 3 | | | | | | |

3c