NEC

NEC Electronics Inc.

µPD70330/70332 (V35) 16-Bit Microcomputers: Advanced, Single-Chip, CMOS

Description

The µPD70330/70332 (V35™) is a high-performance, 16-bit single-chip microcomputer with a 16-bit external data bus. The µPD70330/70332 is fully software compatible with µPD8086/8088 and µPD70108/70116 (V20®/30®) instruction set.

The µPD70330 is a ROMless part. The µPD70332 has 16K ROM, while the µPD70P322 has 16K EPROM and can be used as a µPD70330 (V35) or a µPD70320 (V25™).

Features

- ☐ Functionally compatible with µPD70320/322 (V25)
- ☐ Internal 16-bit architecture and external 16-bit data bus
- ☐ Software compatible with µPD8086/8088, µPD70108/70116 (V20/30) in the native mode
- ☐ New and enhanced instructions
- ☐ Six-byte prefetch queue
- ☐ Minimum instruction cycle: 500 ns at 8 MHz
- ☐ Internal memory
  - — ROM: 16K bytes (µPD70332 only)
  - — RAM: 256 bytes
- ☐ Memory space: 1M bytes
- ☐ Input port with comparator (port T): eight bits
- ☐ Bus interface optimized for use with dynamic RAMs
  - — Multiplexed address
  - — On-board refresh controller

V20 and V30 are registered trademarks of NEC Corporation.
V25 and V35 are trademarks of NEC Corporation.

- ☐ 24 parallel I/O lines
- ☐ Serial interface: two channels
  - — Dedicated baud rate generator
  - — Asynchronous mode, I/O interface mode
- ☐ Interrupt controller
  - — Programmable priority (eight levels)
  - — Three interrupt service functions
  - — Vectored interrupt, register bank switching, macro service
- ☐ DRAM, pseudo SRAM refresh function
- ☐ Two DMA channels
- ☐ Two 16-bit timers
- ☐ One 20-bit time base counter
- ☐ Clock generator
- ☐ Programmable wait function
- ☐ Low power modes
  - — HALT
  - — STOP
- ☐ 1.2-micron CMOS

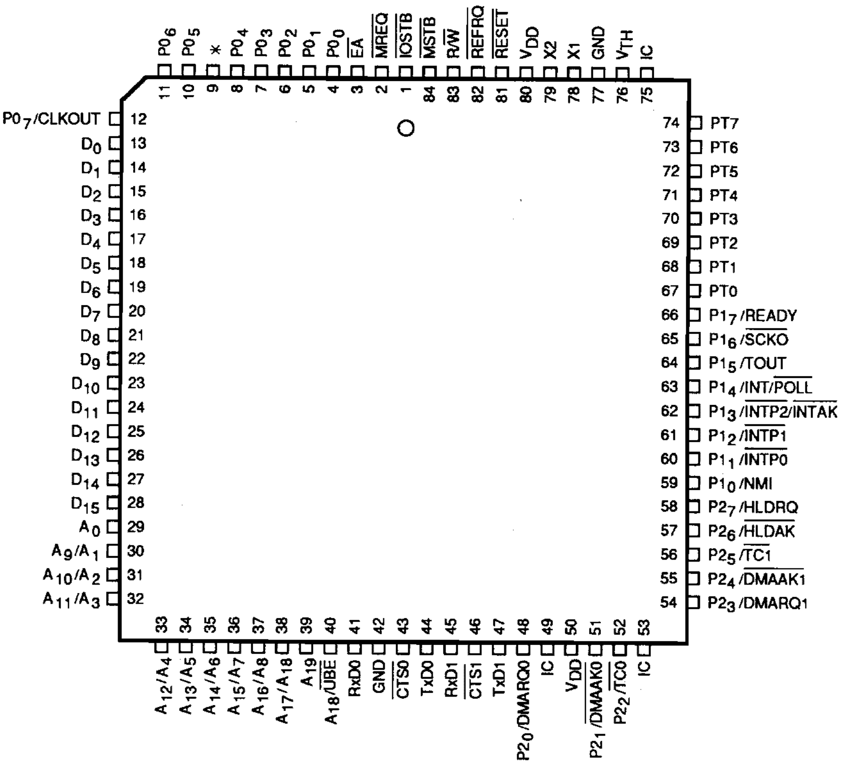
Ordering Information

Part Number	Clock (MHz)	Package	Internal ROM
µPD70330L-8	8	84-pin PLCC	ROMless
GJ-8	8	94-pin plastic QFP	
µPD70332L-8-xxx	8	84-pin PLCC	16K mask ROM
GJ-8-xxx	8	94-pin plastic QFP	
µPD70P322KE-8	8	84-pin LCC	16K EPROM (UV erasable)

NEC

Pin Configuration

84-Pin PLCC and 84-Pin LCC

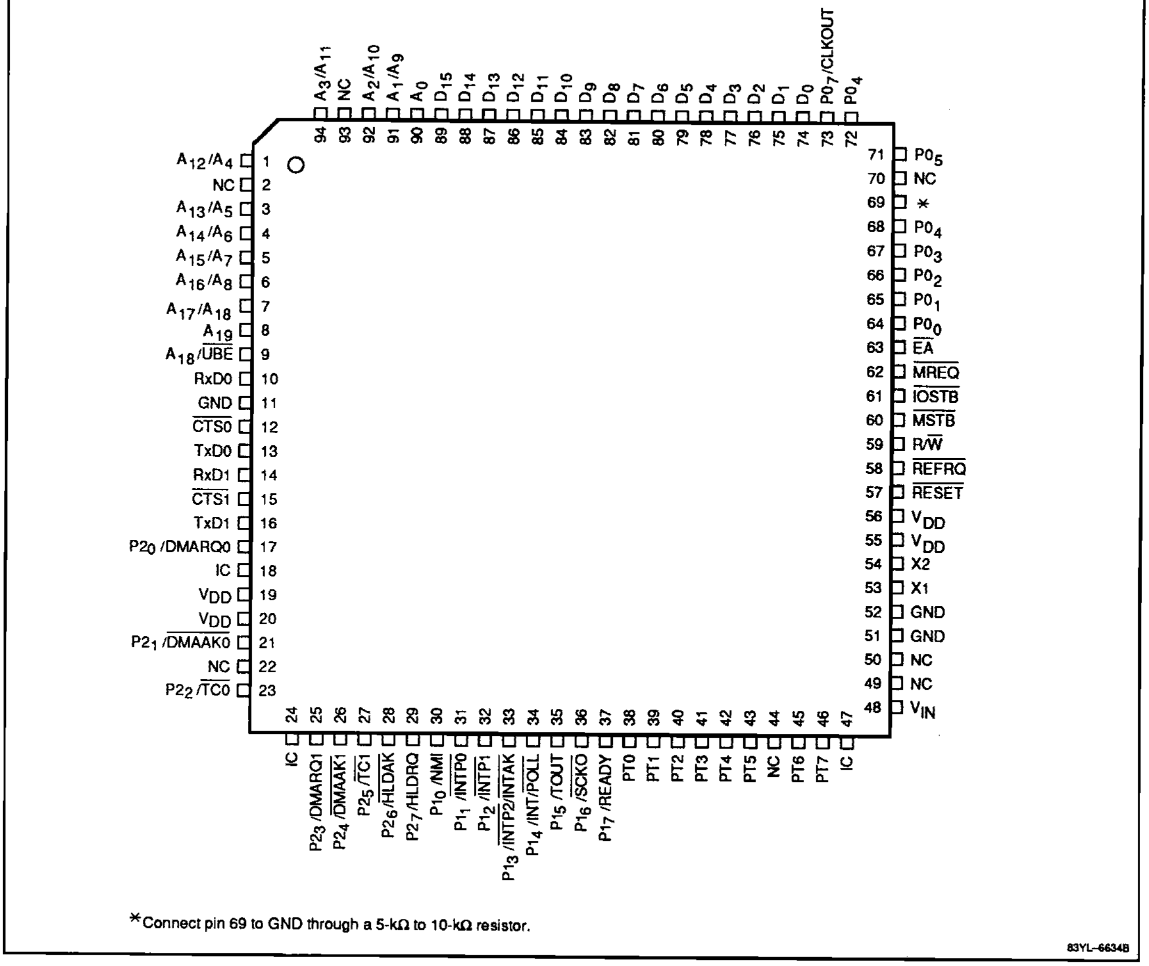


*Connect pin 9 to GND through a 5-kΩ to 10-kΩ resistor.

83YL-6633B

Pin Configuration (cont)

94-Pin Plastic QFP



*Connect pin 69 to GND through a 5-kΩ to 10-kΩ resistor.

83YL-6634B

Pin Identification

Symbol	Function
A_{19}-A_0	Address bus outputs
CLKOUT	System clock output
$\overline{\text{CTS0}}$	Clear-to-send input, serial channel 0
$\overline{\text{CTS1}}$	Clear-to-send input, serial channel 1
D_{15}-D_0	Bidirectional data bus
$\overline{\text{DMAAK0}}$	DMA acknowledge output, DMA controller channel 0
$\overline{\text{DMAAK1}}$	DMA acknowledge output, DMA controller channel 1
DMARQ0	DMA request input, DMA controller channel 0
DMARQ1	DMA request input, DMA controller channel 1
$\overline{\text{EA}}$	External access; clamped low or high according to program access requirements
$\overline{\text{HLDAK}}$	Hold acknowledge output
HLDRQ	Hold request input
INT	Interrupt request input
$\overline{\text{INTAK}}$	Interrupt acknowledge output
$\overline{\text{INTP0}}$	Interrupt request 0 input
$\overline{\text{INTP1}}$	Interrupt request 1 input
$\overline{\text{INTP2}}$	Interrupt request 2 input
$\overline{\text{IOSTB}}$	I/O read or write strobe output
$\overline{\text{MREQ}}$	Memory request output
$\overline{\text{MSTB}}$	Memory strobe output
NMI	Nonmaskable interrupt request
$\overline{\text{POLL}}$	Input on POLL synchronizes the CPU and external devices
$P0_7$-$P0_0$	I/O port 0
$P1_7$-$P1_0$	I/O port 1
$P2_7$-$P2_0$	I/O port 2
PT0-PT7	Comparator port input lines
READY	Ready signal input controls insertion of wait states
$\overline{\text{REFRQ}}$	DRAM refresh request output
$\overline{\text{RESET}}$	Reset signal input
R/$\overline{\text{W}}$	Read/write strobe output
RxD0	Receive data input, serial channel 0
RxD1	Receive data input, serial channel 1
$\overline{\text{SCKO}}$	Serial clock output
$\overline{\text{TC0}}$	Terminal count output; DMA completion, channel 0
$\overline{\text{TC1}}$	Terminal count output; DMA completion, channel 1
TOUT	Timer output
TxD0	Transmit data output, serial channel 0
TxD1	Transmit data output, serial channel 1
$\overline{\text{UBE}}$	Upper byte enable
X1, X2	Connections to external frequency control source (crystal, ceramic resonator, or clock)
V_{DD}	+5-volt power source input (two pins)
V_{TH}	Threshold voltage input to comparator circuits
GND	Ground reference (two pins)
IC	Internal connection; must be tied to V_{DD} externally through a pullup resistor

Pin Functions

A_{19}-A_0; Address Bus

To support dynamic RAMs, the 20-bit address is multiplexed on 11 lines. When $\overline{MREQ}$ is asserted, A_{17}-A_9 are valid. When $\overline{MSTB}$ or $\overline{IOSTB}$ are asserted, A_8-A_1 and A_{18} are valid. A_{18} is also multiplexed with $\overline{UBE}$ and is valid when $\overline{MREQ}$ is asserted. Therefore A_{18} is active throughout the bus cycle. A_{19} and A_0 are not multiplexed but have dedicated pins and are valid throughout the bus cycle.

CLKOUT; Clock Out

The system clock (CLK) is distributed from the internal clock generator to the CPU and output to peripheral hardware at the CLKOUT pin.

$\overline{CTS0}$; Clear-to-Send 0

This is the CTS pin of the channel 0 serial interface. In asynchronous mode, a low-level input on $\overline{CTS0}$ enables transmit operation. In I/O interface mode, $\overline{CTS0}$ is the receive clock pin.

$\overline{CTS1}$; Clear-to-Send 1

This is the CTS pin of the channel 1 serial interface. In asynchronous mode, a low-level input on $\overline{CTS1}$ enables transmit operation.

D_{15}-D_0; Data Bus

D_{15}-D_0 is the 16-bit data bus.

$\overline{DMAAK0}$ and $\overline{DMAAK1}$; DMA Acknowledge

These are the DMA acknowledge outputs of the DMA controller, channels 0 and 1. Signals are not output during DMA memory-to-memory transfer operations (burst mode, single-step mode).

DMARQ0 and DMARQ1; DMA Request

These are the DMA request inputs of the DMA controller, channels 0 and 1.

$\overline{EA}$; External Access

For the ROM-less μPD70330, connect this pin to ground. For the μPD70332, connect $\overline{EA}$ to ground if program code is in external memory; connect $\overline{EA}$ to +5 volts if program code is in the internal ROM.

$\overline{HLDAK}$; Hold Acknowledge

The $\overline{HLDAK}$ output signal indicates that the hold request (HLDRQ) has been accepted. When $\overline{HLDAK}$ is active (low), the following lines go to the high-impedance state with internal 4700-ohm pullup resistors: A_{19}-A_0, D_7-D_0, $\overline{IOSTB}$, $\overline{MREQ}$, $\overline{MSTB}$, $\overline{REFRQ}$, and R/W.

HLDRQ; Hold Request

The HLDRQ input from an external device requests that the μPD70330/332 relinquish the address, data, and control buses to an external bus master.

INT; Interrupt

The INT input is a vectored interrupt request from an external device that can be masked by software. The active high level is detected in the last clock cycle of an instruction. The external device confirms that the INT interrupt request has been accepted by the $\overline{INTAK}$ signal output from the CPU.

The INT signal must be held high until the first $\overline{INTAK}$ signal is output. Together with $\overline{INTAK}$, INT is used for operation with an interrupt controller such as μPD71059.

$\overline{INTAK}$; Interrupt Acknowledge

The $\overline{INTAK}$ output is the acknowledge signal for the software-maskable interrupt request INT. The $\overline{INTAK}$ signal goes low when the CPU accepts INT. The external device inputs the interrupt vector to the CPU via data bus D_7-D_0 in synchronization with $\overline{INTAK}$.

$\overline{\text{INTP0}}$, $\overline{\text{INTP1}}$, $\overline{\text{INTP2}}$; Interrupt from Peripheral 0, 1, 2

The $\overline{\text{INTPn}}$ inputs (n = 0, 1, 2) are external interrupt requests that can be masked by software. The $\overline{\text{INTPn}}$ input is detected at the effective edge specified by external interrupt mode register INTM.

The $\overline{\text{INTPn}}$ input is also used to release the HALT mode.

$\overline{\text{IOSTB}}$; I/O Strobe

A low-level output on $\overline{\text{IOSTB}}$ indicates that the I/O bus cycle has been initiated and that the I/O address output on A_{15}-A_0 is valid.

$\overline{\text{MREQ}}$; Memory Request

A low-level output on $\overline{\text{MREQ}}$ indicates that the memory or I/O bus cycle has started and that address bits A_0, A_{17}-A_9, A_{19} and A_{18} are valid.

$\overline{\text{MSTB}}$; Memory Strobe

Together with $\overline{\text{MREQ}}$ and $R/\overline{W}$, $\overline{\text{MSTB}}$ controls memory accessing operations. $\overline{\text{MSTB}}$ should be used either to enable data buffers or as a data strobe. During memory write, a low-level output on $\overline{\text{MSTB}}$ indicates that data on the data bus is valid. A low-level output on $\overline{\text{MSTB}}$ indicates that multiplexed address bits A_8-A_1, A_{18}, and $\overline{\text{UBE}}$ are valid.

NMI; Nonmaskable Interrupt

The NMI input is an interrupt request that cannot be masked by software. The NMI is always accepted by the CPU; therefore, it has priority over any other interrupt.

The NMI input is detected at the effective edge specified by external interrupt mode register INTM. Sampled in each clock cycle, NMI is accepted when the active level lasts for some clock cycles. When the NMI is accepted, a number 2 vector interrupt is generated after completion of the instruction currently being executed.

The NMI input is also used to release the CPU standby mode.

$P0_7$-$P0_0$; Port 0

Port 0 is an 8-bit bidirectional I/O port.

$P1_7$-$P1_0$; Port 1

Lines $P1_7$-$P1_4$ are individually programmable as an input, output, or control function. The status of $P1_3$-$P1_0$ can be read but these lines are always control functions.

$P2_7$-$P2_0$; Port 2

$P2_7$-$P2_0$ are the lines of port 2, an 8-bit bidirectional I/O port. These lines can also be used as control signals for the on-chip DMA controllers. See table 2-3.

$\overline{\text{POLL}}$; Poll

The $\overline{\text{POLL}}$ input is checked by the POLL instruction. If the level is low, execution of the next instruction is initiated. If the level is high, the $\overline{\text{POLL}}$ input is checked every five clock cycles until the level becomes low.

The POLL functions are used to synchronize the CPU program and the operation of external devices.

Note: $\overline{\text{POLL}}$ is effective when $P1_4$ is specified for the input port mode; otherwise, $\overline{\text{POLL}}$ is assumed to be at low level when the POLL instruction is executed.

PT0-PT7; Port with Comparator

The PT input is compared with a threshold voltage that is programmable to one of 16 voltage steps individually for each of the eight lines.

READY

After READY is de-asserted low, the CPU will synchronize and insert at least two wait states into a read or write cycle to memory or I/O. This allows the processor to accommodate devices whose access times are longer than normal execution allows.

$\overline{\text{REFRQ}}$; Refresh Request

This output pulse can refresh nonstatic RAM. It can be programmed to meet system specifications and is internally synchronized so that refresh cycles do not interfere with normal CPU operation.

$\overline{\text{RESET}}$

This input signal is asynchronous. A low on RESET for a certain duration resets the CPU and all on-chip peripherals regardless of clock operation. The reset operation has priority over all other operations.

The reset signal is used for normal initialization/startup and also for releasing the STOP or HALT mode. After the reset signal returns high, program execution begins from address FFFF0H.

$R/\overline{W}$; Read/Write Strobe

When the memory bus cycle is initiated, the $R/\overline{W}$ signal output to external hardware indicates a read (high level) or write (low level) cycle. It can also control the direction of bidirectional buffers.

RxD0, RxD1; Receive Data 0, 1

These pins input data from serial channels 0 and 1.

In the asynchronous mode, when receive operation is enabled, a low level on the RxD0 or RxD1 input pin is recognized as the start bit and receive operation is initiated.

In the I/O interface mode (channel 0 only), receive data is input to the serial register at the rising edge of the receive clock.

$\overline{SCKO}$; Serial Clock

The $\overline{SCKO}$ output is the transmit clock of serial channel 0.

$\overline{TC0}$, $\overline{TC1}$; Terminal Count 0, 1

The $\overline{TC0}$ and $\overline{TC1}$ outputs go low when the terminal count of DMA service channels 0 and 1, respectively, reach zero, indicating DMA completion.

TOUT; Timer Output

The TOUT signal is a square-wave output from the internal timer.

TxD0, TxD1; Transmit Data 0, 1

These pins output data from serial channels 0 and 1.

In the asynchronous mode, the transmit signal is in a frame format that consists of a start bit, 7 or 8 data bits (least significant bit first), parity bit, and stop bit. The TxD0 and TxD1 pins become mark state (high level) when transmit operation is disabled or when the serial register has no transmit data.

In the I/O interface mode (channel 0 only), the frame has 8 data bits and the most significant bit is transmitted first.

X1, X2; Clock Control

The frequency of the internal clock generator is controlled by an external crystal or ceramic resonator connected across pins X1 and X2. The crystal frequency is the same as the clock generator frequency f_X. By programming the PRC register, the system clock frequency f_{CLK} is selected as f_X divided by 2, 4, or 8.

As an alternative to the crystal or ceramic resonator, the positive and negative phases of an external clock (with frequency f_X) can be connected to pins X1 and X2.

V_{DD}

+5-volt power source (two pins).

V_{TH}

Comparator port PT0-PT7 uses threshold voltage V_{TH} to determine the analog reference points. The actual threshold to each comparator line is programmable to $V_{TH} \times n/16$ where $n = 1$ to 16.

GND

Ground reference (two pins).

IC

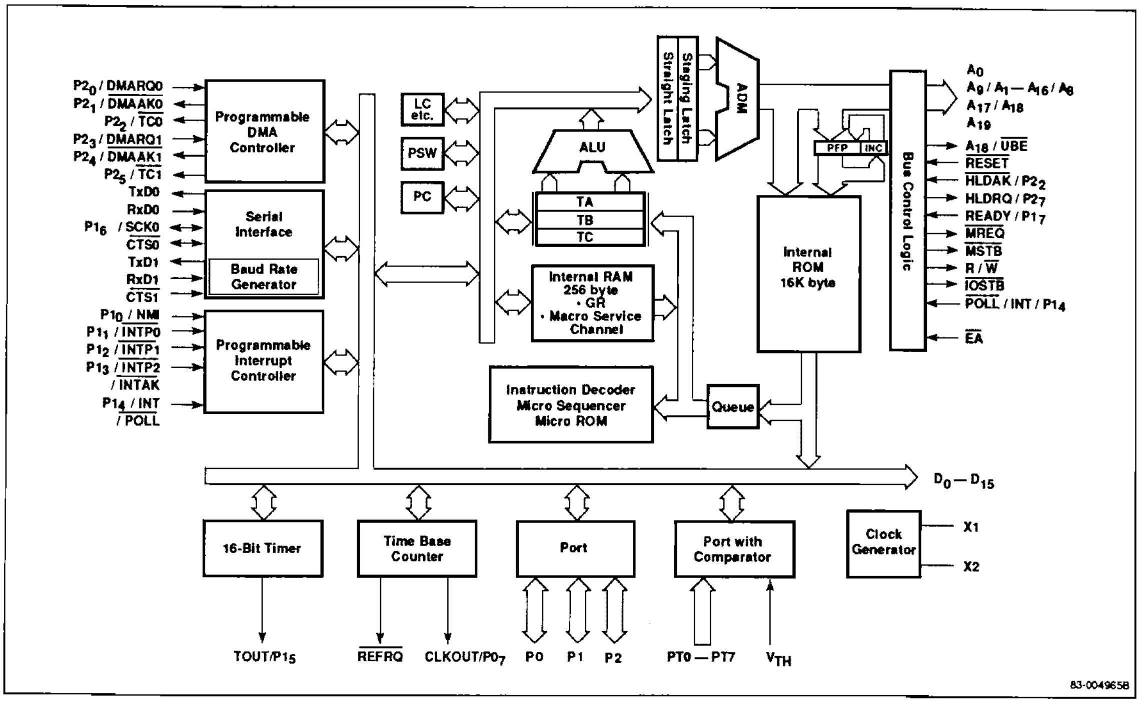
Internal connection; must be tied to V_{DD} externally through a 10-kΩ to 20-kΩ resistor.

$\overline{UBE}$, Upper Byte Enable

$\overline{UBE}$ is a high-order memory bank selection signal output. $\overline{UBE}$ and A_0 are used to decide which bytes of the data bus will be used. $\overline{UBE}$ is used along with A_0 to select the even/odd banks as follows.

Operand	$\overline{UBE}$	A_0	Number of bus cycles
Even address word	0	0	1
Odd address word	0	1	2
	1	0	
Even address byte	1	0	1
Odd address byte	0	1	1

Block Diagram



Functional Description

Architectural Enhancements

The following features enable the μPD70330/332 to perform high-speed execution of instructions:

- Dual data bus
- 16-/32-bit temporary registers/shifters (TA, TB, TA + TB)
- 16-bit loop counter (LC)
- Program counter (PC) and prefetch pointer (PFP)
- Internal ROM pass bus (μPD70332 only)

Dual Data Bus. The μPD70330/332 has two internal 16-bit data buses: the main data bus and a subdata bus. This reduces the processing time required for addition/subtraction and logical comparison instructions by one-third over single-bus systems. The dual data bus method allows two operands to be fetched simultaneously from the general-purpose registers and transferred to the ALU.

16-/32-Bit Temporary Registers/Shifters. The 16-bit temporary registers/shifters (TA, TB) allow high-speed execution of multiplication/division and shift/rotation instructions. By using the temporary registers/shifters, the μPD70330/332 can execute multiplication/division instructions about four times faster than with the microprogramming method.

Loop Counter [LC]. The dedicated hardware loop counter counts the number of loops for string operations and the number of shifts performed for multiple bit shift/rotation instructions. The loop counter works with internal dedicated shifters to speed the processing of multiplication/division instructions.

Program Counter and Prefetch Pointer [PC and PFP]. The hardware PC addresses the memory location of the instruction to be executed next. The hardware PFP addresses the program memory location to be accessed next. Several clocks are saved for branch, call, return, and break instructions compared with processors having only one instruction pointer.

Internal ROM Pass Bus. The μPD70332 features a dedicated data bus between the internal ROM and the instruction pre-fetch queue. This allows internal ROM opcode fetches to be performed in a single clock cycle (200 ns at 5 MHz); it also makes it possible for opcode fetches to be performed while the external data bus is busy. This feature gives the V35 a 10-20% performance increase when executing from the internal ROM.

Register Set

The μPD70330/70332 CPUs have general purpose register sets compatible with the μPD70108/70116 and the μPD70320/70322 microprocessors. Like the μPD70320/70322, they also have a set of special function registers for controlling the onboard peripherals. All registers reside in the CPU's memory space. They are grouped in a 4K byte block called the internal data area (IDA). The 256 byte internal RAM is also in the IDA. The addresses of the register are given as offsets into the IDA. The start address of the IDA is set by the Internal Data Area Base register (IDB), and may be programmed to any 4K boundary in the memory address space.

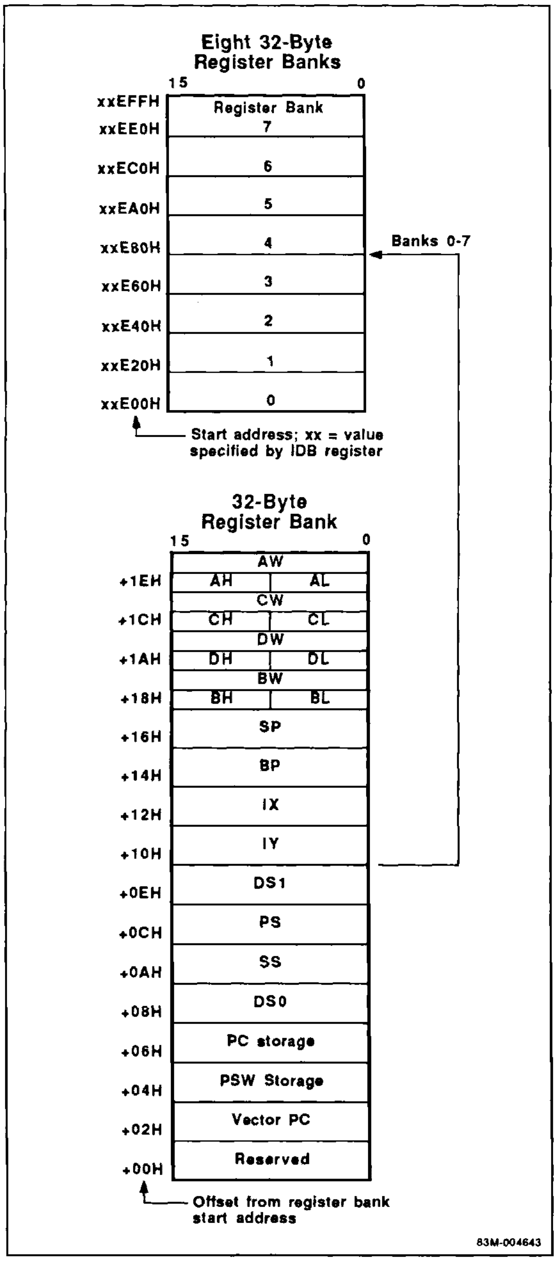
Register Banks. Because the general purpose register set is in internal RAM, it is possible to have multiple banks of registers. The μPD70330/70332 CPU supports up to 8 register banks. A bit field in the PSW selects which bank is currently being used. Each bank contains the entire CPU register set plus additional information needed for context switching. Register banks may be switched using special instructions (TSKSW, BRKCS, MOVSPA, MOVSPB), or may switch in response to an interrupt. This provides fast context switching and fast interrupt handling. During and after RESET, register bank 7 is selected.

Figure 1 shows the configuration of a register bank and how the banks are mapped to internal RAM. The Vector PC field contains the value that will be loaded into the PC when a register bank switch occurs. The PC Save and PSW Save fields contain the values of the PC and the PSW just before the banks are switched. The PSW is left unmodified after a bank switch; the PSW Save field is used to restore the PSW to its previous state is required.

General-Purpose Registers [AW, BW, CW, DW]. These four 16-bit general-purpose registers can also serve as independent 8-bit registers (AH, AL, BH, BL, CH, CL, DH, DL). The instructions below use general-purpose registers for default:

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 branch, repeat prefix
CL	Shift instructions, rotation instructions, BCD operations
DW	Word multiplication/division, indirect addressing I/O

Figure 1. Register Bank Configuration



Pointers [SP, BP] and Index Registers [IX, IY]. These registers are used as 16-bit base pointers or index registers in based addressing, indexed addressing, and based indexed addressing. The registers are used as default registers under the following conditions:

SP	Stack operations
IX	Block transfer (source), BCD string operations
IY	Block transfer (destination), BCD string operations

Segment Registers. The segment registers divide the 1M-byte address space into 64K-byte blocks. Each segment register functions as a base address to a block; the effective address is an offset from that base. Physical addresses are generated by shifting the associated segment register left four binary digits and then adding the effective address. The segment registers are:

Segment Register	Default Offset
PS (Program segment)	PC
SS (Stack segment)	SP, Effective address
DS0 (Data segment-0)	IX, Effective address
DS1 (Data segment-1)	IY, Effective address

During RESET, PS is set to FFFFH; DS0, DS1 and SS are set to 0000H.

Program Counter [PC]. The PC is a 16-bit binary counter that contains the offset address from the program segment of the next instruction to be executed. It is incremented every time an instruction is received from the queue. It is loaded with a new location whenever a branch, call, return, break, or interrupt is executed. During RESET, PC is set to 0000H.

Program Status Word [PSW]. The PSW contains the following status and control flags.

15 PSW 8

1	RB2	RB1	RB0	V	DIR	IE	BRK

7 0

S	Z	F1	AC	F0	P	BRKI	CY

Status Flags	
V	Overflow bit
S	Sign
Z	Zero
AC	Auxiliary carry
P	Parity
CY	Carry

Control Flags	
DIR	Direction of string processing
IE	Interrupt enable
BRK	Break (after every instruction)
RBn	Current register bank flags
BRKI	I/O trap enable (see software interrupts)
F0, F1	General-purpose user flags

The eight low-order bits of the PSW can be stored in the AH register and restored by a MOV instruction execution. The only way to alter the RBn bits via software is to execute an RETRBI or RETI instruction. During RESET, PSW is set to F002H. The F0 and F1 flags may be accessed as bits in the FLAG special functioning register.

Memory Map

The μPD70330/332 has a 20-bit address bus that can directly access 1M bytes of memory. Figure 2 shows that the 16K bytes of internal ROM (μPD70332 only) are located at the top of the address space from FC000H to FFFFFH.

Figure 2. Memory Map

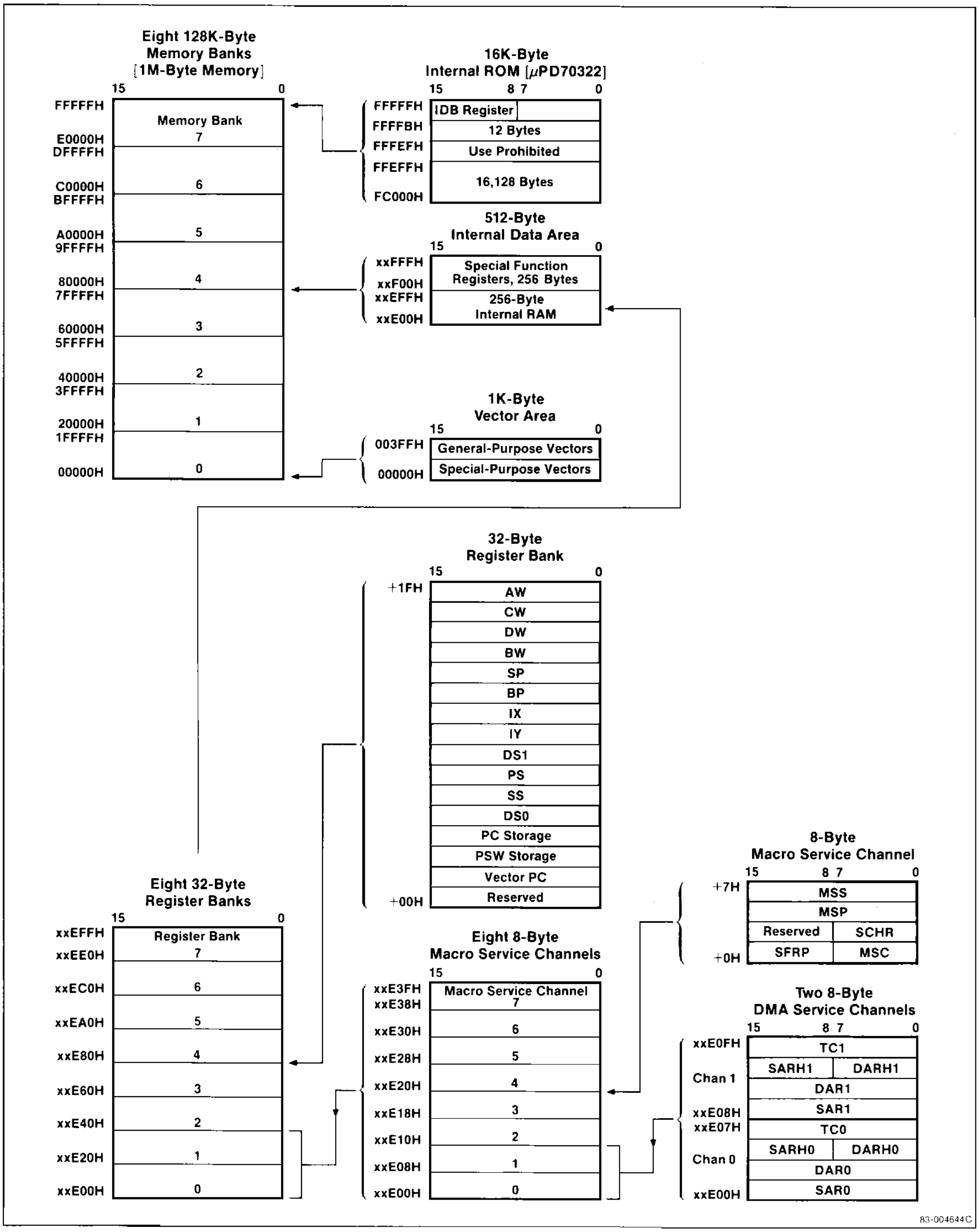


Figure 2 shows the internal data area (IDA) is a 256-byte internal RAM area followed consecutively by a 256-byte special function register (SFR) area. All the data and control registers for on-chip peripherals and I/O are mapped into the SFR area and accessed as RAM. For a description of these functions, see table 6. The IDA is dynamically relocatable in 4K-byte increments by changing the value in the internal data base (IDB) register. Whatever value is in this register will be assigned as the uppermost eight bits of the IDA address. The IDB register can be accessed from two different memory locations, FFFFFH and XXFFFH, where XX is the value in the IDB register.

On reset, the internal data base register is set to FFH which maps the IDA into the internal ROM space. However, since the μPD70332 has a separate bus to internal ROM, this does not present a problem. When these address spaces overlap, program code cannot be executed from the IDA and internal ROM locations cannot be accessed as data.

Figure 2 shows that the internal data area is divided into 2 parts: the 256 byte internal RAM and the special function register area.

The internal RAM area serves various purposes. When the RAMEN bit in the Processor Control Register is set, this area may be accessed as RAM and code may be executed from it. Note that the processor may run slower when the RAMEN bit is set. See the Instruction Clock Count table. In addition, whether the RAMEN bit is on or off, each of the 8 macroservice channels has an 8 byte control block that is assigned to a fixed location in the low 64 bytes of the internal RAM. Similarly, the two 8 byte DMA control blocks are assigned to the low 16 bytes of the RAM. The 8 CPU register banks use 32 bytes each. Since the RAM can't be used for more than one purpose, there are restrictions on how V35 features can be combined. For example, if register bank 0 is used, then macroservice channels 0-3 and both DMA channels cannot be used. If DMA channel 1 is used, then macroservice channel 1 cannot be used.

The special function register area contains the registers used to control the onboard peripheral functions. Table 6 shows the SFRs. The address shown in the table is an offset from the IDB register. Most SFRs can be both read and written, but some are read-only; others are write-only. Some SFRs may be accessed one bit at a time; others only 8 bits at a time, and some SFRs are 16 bits wide.

Instructions

The μPD70330/332 instruction set is fully compatible with the V20 native mode instruction set. The V20 instruction set is a superset of the μPD8086/8088 instruction set with different execution times and mnemonics.

The μPD70330/332 does not support the V20 8080 emulation mode. All of the instructions pertaining to this have been deleted from the μPD70330/332 instruction set.

Enhanced Instructions

In addition to the μPD8086/88 instructions, the μPD70330/332 has the following enhanced instructions.

Instruction	Function
PUSH imm	Pushes immediate data onto stack
PUSH R	Pushes eight general registers onto stack
POP R	Pops eight general registers from stack
MUL imm	Executes 16-bit multiply of register or memory contents by immediate data
SHL imm8 SHR imm8 SHRA imm8 ROL imm8 ROR imm8 ROLC imm8 RORC imm8	Shifts/rotates register or memory by immediate value
CHKIND	Checks array index against designated boundaries
INM	Moves a string from an I/O port to memory
OUTM	Moves a string from memory to an I/O port
PREPARE	Allocates an area for a stack frame and copies previous frame pointers
DISPOSE	Frees the current stack frame on a procedure exit

Unique Instructions

The μPD70330/332 has the following unique instructions.

Instruction	Function
INS	Inserts bit field
EXT	Extracts bit field
ADD4S	Performs packed BCD string addition
SUB4S	Performs packed BCD string subtraction
CMP4S	Performs packed BCD string comparison
ROL4	Rotates BCD digit left
ROR4	Rotates BCD digit right
TEST1	Tests bit
SET1	Sets bit
CLR1	Clears bit
NOT1	Complements bit
BTCLR	Tests bit; if true, clear and branch
REPC	Repeat while carry set
REPNC	Repeat while carry cleared

Variable Length Bit Field Operation Instructions

Bit fields are a variable length data structure that can range in length from 1 to 16 bits. The μPD70330/332 supports two separate operations on bit fields: insertion (INS) and extraction (EXT). There are no restrictions on the position of the bit field in memory. Separate segment, byte offset, and bit offset registers are used for insertion and extraction. Following the execution of these instructions, both the byte offset and bit offset are left pointing to the start of the next bit field, ready for the next operation. Bit field operation instructions are powerful and flexible and are therefore highly effective for graphics, high-level languages, and packing/unpacking applications.

Bit field insertion copies the bit field of specified length from the AW register to the bit field addressed by DS1:IY:reg8 (8-bit general-purpose register). The bit field length can be located in any byte register or supplied as immediate data. Following execution, both the IY and reg8 are updated to point to the start of the next bit field.

Bit field extraction copies the bit field of specified length from the bit field addressed by DS0:IX:reg8 to the AW register. If the length of the bit field is less than 16 bits, the bit field is right justified with a zero fill. The bit field length can be located in any byte register or supplied as immediate data. Following execution, both IX and reg8 are updated to point to the start of the next bit field.

Figures 3 and 4 show bit field insertion and bit field extraction.

4b

Packed BCD Instructions

Packed BCD instructions process packed BCD data either as strings (ADD4S, SUB4S, CMP4S) or byte format operands (ROR4, ROL4). Packed BCD strings may be 1 to 254 digits in length. The two BCD rotation instructions perform rotation of a single BCD digit in the lower half of the AL register through the register or the memory operand.

Figure 3. Bit Field Insertion

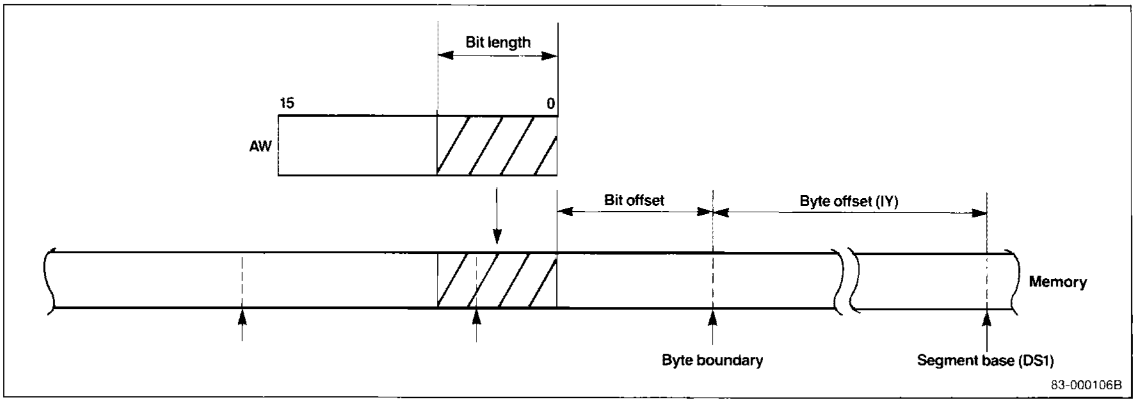
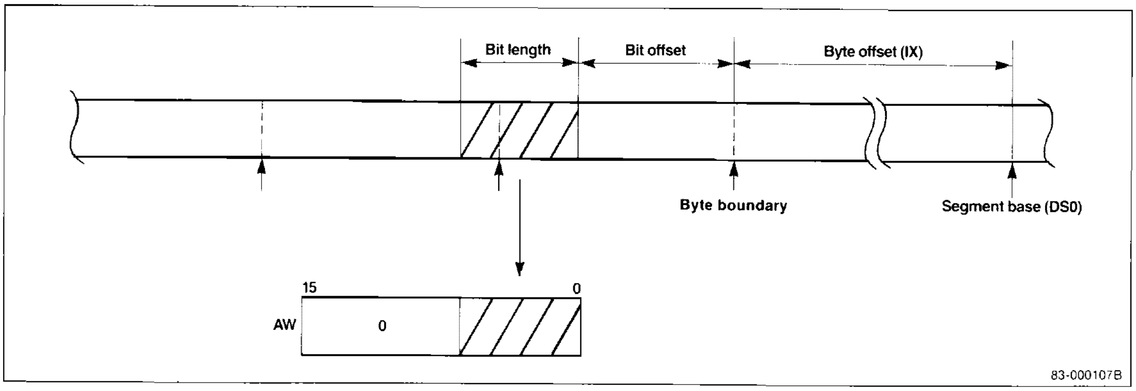


Figure 4. Bit Field Extraction



Bit Manipulation Instructions

The μPD70330/332 has five unique bit manipulation instructions. 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. This feature further enhances control over on-chip peripherals.

Additional Instructions

Besides the V20 instruction set, the μPD70330/0332 has the eight additional instructions described in table 1.

Table 1. Additional Instructions

Instruction	Function
BTCLR var,imm8, short label	Bit test and if true, clear and branch; otherwise, no operation
STOP (no operand)	Power down instruction, stops oscillator
RETRBI (no operand)	Return from register bank context switch interrupt
FINT (no operand)	Finished interrupt. After completion of a hardware interrupt request, this instruction must be used to reset the current priority bit in the in-service priority register (ISPR).*

*Do not use with NMI or INTR interrupt service routines.

Repeat Prefixes

Two new repeat prefixes (REPC, REPNC) allow conditional block transfer instructions to use the state of the CY flag as the termination condition. This allows inequalities to be used when working on ordered data, thus increasing performance when searching and sorting algorithms.

Bank Switch Instructions

The V35 has four new instructions that allow the effective use of the register banks for software interrupts and multitasking. These instructions are shown in table 2. Also, see figures 8 and 10.

Table 2. Bank Switch Instructions

Instruction	Function
BRKCS reg 16	Performs a high-speed software interrupt with context switch to the register bank indicated by the lower 3-bits of reg 16. This operation is identical to the interrupt operation shown in figure 9.
TSKSW reg 16	Performs a high-speed task switch to the register bank indicated by the lower 3-bits of reg 16. The PC and PSW are saved in the old banks. PC and PSW save registers and the new PC and PSW values are retrieved from the new register bank's save areas. See figure 10.
MOVSPA	Transfers both the SS and SP of the old register bank to the new register bank after the bank has been switched by an interrupt or BRKCS instruction.
MOVSPB	Transfers the SS and the SP of the current register bank before the switch to the SS and SP of the new register bank indicated by the lower 3-bits of reg 16.

Interrupt Structure

The μPD70330/332 can service interrupts generated both by hardware and by software. Software interrupts are serviced through vectored interrupt processing. See table 3 for the various types of software interrupts.

Table 3. Software Interrupts

Interrupt	Description
Divide error	The CPU will trap if a divide error occurs as the result of a DIV or DIVU instruction.
Single step	The interrupt is generated after every instruction if the BRK bit in the PSW is set.
Overflow	By using the BRKV instruction, an interrupt can be generated as the result of an overflow.
Interrupt instructions	The BRK 3 and BRK imm8 instructions can generate interrupts.
Array bounds	The CHKIND instruction will generate an interrupt if specified array bounds have been exceeded.
Escape trap	The CPU will trap on an FP01,2 instruction to allow software to emulate the floating point processor.
I/O trap	If the I/O trap bit in the PSW is cleared, a trap will be generated on every IN or OUT instruction. Software can then provide an updated peripheral address. This feature allows software interchangeability between different systems.

When executing software written for another system, it is better to implement I/O with on-chip peripherals to reduce external hardware requirements. However, since μPD70330/332 internal peripherals are memory mapped, software conversion could be difficult. The I/O trap feature allows easy conversion from external peripherals to on-chip peripherals.

Interrupt Vectors

The starting address of the interrupt processing routines may be obtained from table 4. The table begins at physical address 00H, which is outside the internal ROM space. Therefore, external memory is required to service these routines. By servicing interrupts via the macro service function or context switching, this requirement can be eliminated.

Each interrupt vector is four bytes wide. To service a vectored interrupt, the lower addressed word is transferred to the PC and the upper word to the PS. See figure 5.

Figure 5. Interrupt Vector 0

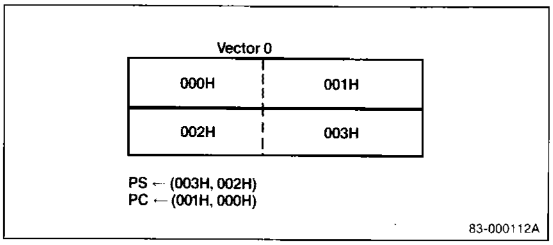


Table 4. Interrupt Vectors

Address	Vector No.	Assigned Use
00	0	Divide error
04	1	Break flag
08	2	NMI
0C	3	BRK3 instruction
10	4	BRKV instruction
14	5	CHKIND instruction
18	6	General purpose
1C	7	FPO instructions
20-2C	8-11	General purpose
30	12	INTSER0 (Interrupt serial error, channel 0)
34	13	INTSR0 (Interrupt serial receive, channel 0)
38	14	INTST0 (Interrupt serial transmit, channel 0)
3C	15	General purpose
40	16	INTSER1 (Interrupt serial error, channel 1)
44	17	INTSR1 (Interrupt serial receive, channel 1)
48	18	INTST1 (Interrupt serial transmit, channel 1)
4C	19	I/O trap
50	20	INTD0 (Interrupt from DMA, channel 0)
54	21	INTD1 (Interrupt from DMA, channel 1)
58	22	General purpose
5C	23	General purpose
60	24	INTP0 (Interrupt from peripheral 0)
64	25	INTP1 (Interrupt from peripheral 1)
68	26	INTP2 (Interrupt from peripheral 2)
6C	27	General purpose
70	28	INTTU0 (Interrupt from timer unit 0)
74	29	INTTU1 (Interrupt from timer unit 1)
78	30	INTTU2 (Interrupt from timer unit 2)
7C	31	INTTB (Interrupt from time base counter)
080-3FF	32-255	General purpose

Execution of a vectored interrupt occurs as follows:

(SP-1, SP-2) ← PSW
(SP-3, SP-4) ← PS
(SP-5, SP-6) ← PC
SP ← SP-6
IE ← 0, BRK ← 0
PS ← vector high bytes
PC ← vector low bytes

Hardware Interrupt Configuration

The V35 features a high-performance on-chip controller capable of controlling multiple processing for interrupts from up to 17 different sources (5 external, 12 internal). The interrupt configuration includes system interrupts that are functionally compatible with those of the V20/V30 and unique high-performance microcontroller interrupts.

Interrupt Sources

The 17 interrupt sources (table 5) are divided into groups for management by the interrupt controller. Using software, each of the groups can be assigned a priority from 0 (highest) to 7 (lowest). The priority of individual interrupts within a group is fixed in hardware.

If interrupts from different groups occur simultaneously and the groups have the same assigned priority level, the priority followed will be as shown in the Default Priority column of table 5.

The ISPR is an 8-bit SFR; bits PR_0-PR_7 correspond to the eight possible interrupt request priorities. The ISPR keeps track of the priority of the interrupt currently being serviced by setting the appropriate bit. The address of the ISPR is XXFFCH. The ISPR format is shown below.

PR_7	PR_6	PR_5	PR_4	PR_3	PR_2	PR_1	PR_0

NMI and INT are system-type external vectored interrupts. NMI is not maskable via software. INTR is maskable (IE bit in PSW) and requires that an external device provide the interrupt vector number. It allows expansion by the addition of an external interrupt controller (μPD71059).

NMI, INTP0, and INTP1 are edge-sensitive interrupt inputs. By selecting the appropriate bits in the interrupt mode register, these inputs can be programmed to be either rising or falling edge triggered. ES_0-ES_2 correspond to INTP0-INTP2, respectively. See figure 6.

Figure 6. Interrupt Mode Register (INTM)

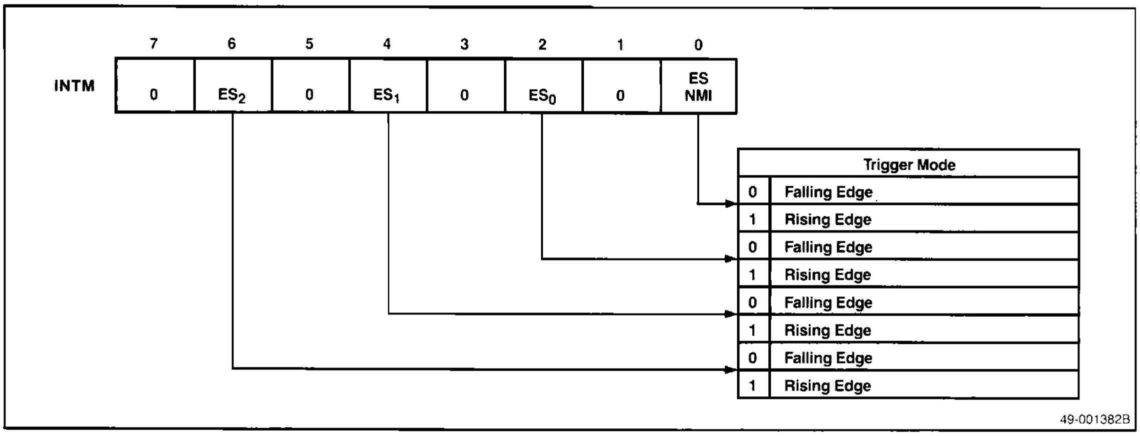


Table 5. Interrupt Sources

Interrupt Source	External/ Internal	Vector	Macro Service	Bank Switching	Priority Order: Setting Possible	Priority Order: Between Groups	Priority Order: Within Groups	Multiple Processing Control
NMI Nonmaskable interrupt	External	2	No	No	No	0	—	Not accepted
INTTU0 Interrupt from timer unit 0	Internal	28	Yes	Yes	Yes	1	1	Accepted
INTTU1 Interrupt from timer unit 1	Internal	29	Yes	Yes	Yes	1	2	
INTTU2 Interrupt from timer unit 2	Internal	30	Yes	Yes	Yes	1	3	
INTD0 Interrupt from DMA channel 0	Internal	20	No	Yes	Yes	2	1	Accepted
INTD1 Interrupt from DMA channel 1	Internal	21	No	Yes	Yes	2	2	
INTP0 Interrupt from peripheral 0	External	24	Yes	Yes	Yes	3	1	Accepted
INTP1 Interrupt from peripheral 1	External	25	Yes	Yes	Yes	3	2	
INTP2 Interrupt from peripheral 2	External	26	Yes	Yes	Yes	3	3	
INTSER0 Interrupt from serial error on channel 0	Internal	12	No	Yes	Yes	4	1	Accepted
INTSR0 Interrupt from serial receiver of channel 0	Internal	13	Yes	Yes	Yes	4	2	
INTST0 Interrupt from serial transmitter of channel 0	Internal	14	Yes	Yes	Yes	4	3	
INTSER1 Interrupt from serial error on channel 1	Internal	16	No	Yes	Yes	5	1	Accepted
INTSR1 Interrupt from serial receiver of channel 1	Internal	17	Yes	Yes	Yes	5	2	
INTST1 Interrupt from serial transmitter of channel 1	Internal	18	Yes	Yes	Yes	5	3	
INTTB Interrupt from time base counter	Internal	31	No	No	No (Preset to 7)	6	—	Accepted
INT Interrupt	External	Ext. input	No	No	No	7	—	Not accepted

Interrupt Processing Modes

Interrupts, with the exception of NMI, INT, and INTTB, have high-performance capability and can be processed in any of three modes: standard vectored interrupt, register bank context switching, or macro service function. The processing mode for a given interrupt can be chosen by enabling the appropriate bits in the corresponding interrupt request control register. As shown in table 6, each individual interrupt, with the exception of INTR and NMI, has its own associated IRC register. The format for all IRC registers is shown in figure 7. There is an IRC for every interrupt source except NHI and INT.

All interrupt processing routines other than those for NMI and INT must end with the execution of an FINT instruction. Otherwise, subsequently, only interrupts of a higher priority will be accepted. FINT allows the internal interrupt controller to begin looking for new interrupts.

In the vectored interrupt mode, the CPU traps to the vector location in the interrupt vector table.

Register Bank Switching

Register bank context switching allows interrupts to be processed rapidly by switching register banks. After an interrupt, the new register bank selected is that which has the same register bank number (0-7) as the priority of the interrupt to be serviced. The PC and PSW are automatically stored in the save areas of the new register bank and the address of the interrupt routine is loaded from the vector PC storage location in the new register bank. As in the vectored mode, the IE and BRK bits in the PSW are cleared to zero. After interrupt processing, execution of the RETRBI (return from register bank interrupt) returns control to the former register bank and restores the former PC and PSW. Figures 8 and 9 show register bank context switching and register bank return.

Specific IRC registers include the following.

Symbol	IRC Register
DIC0, DIC1	DMA
EXIC0-EXIC2	External
SEIC0, SEIC1	Serial error
SRIC0, SRIC1	Serial receive
STIC0, STIC1	Serial transmit
TMIC0-TMIC2	Timer

Figure 7. Interrupt Request Control Registers (IRC)

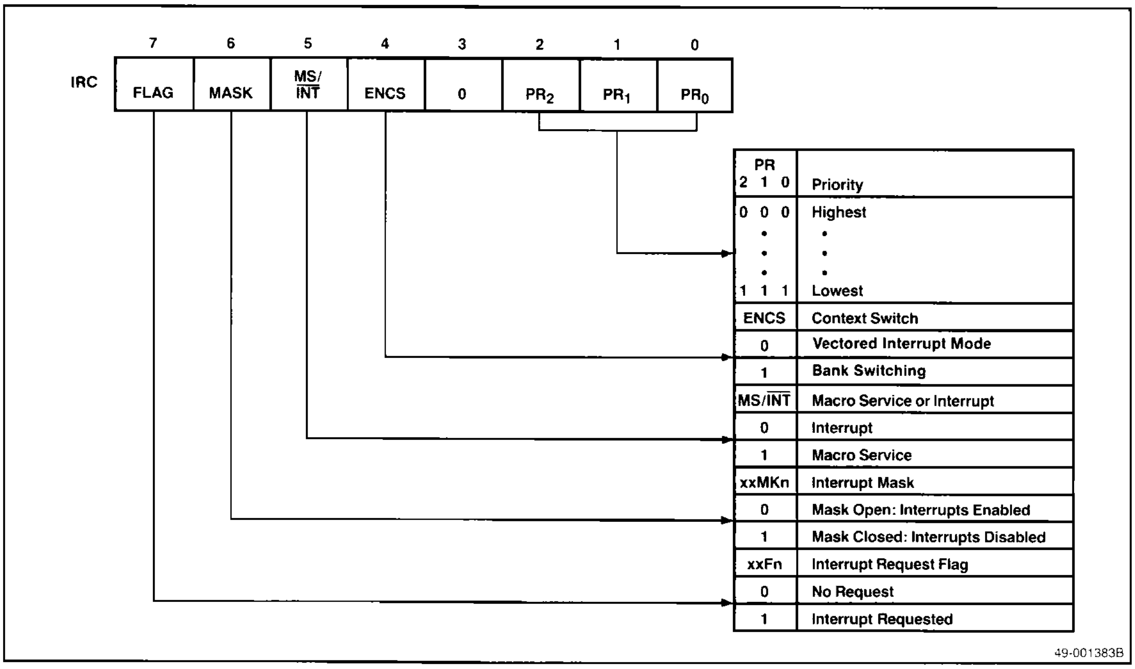


Figure 8. Register Bank Context Switching

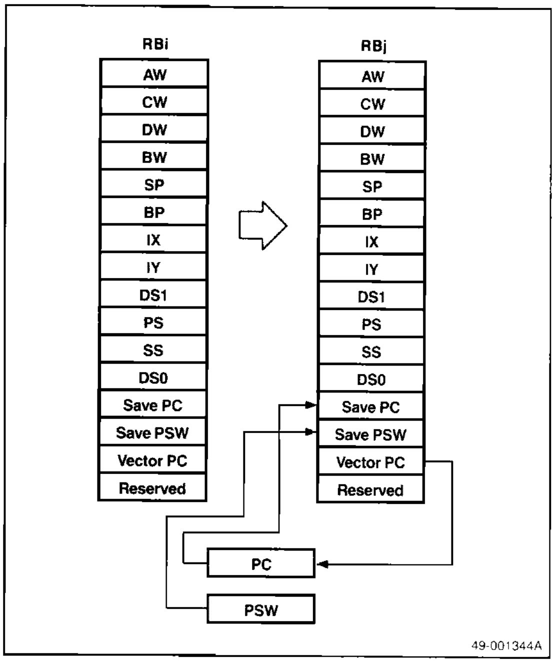
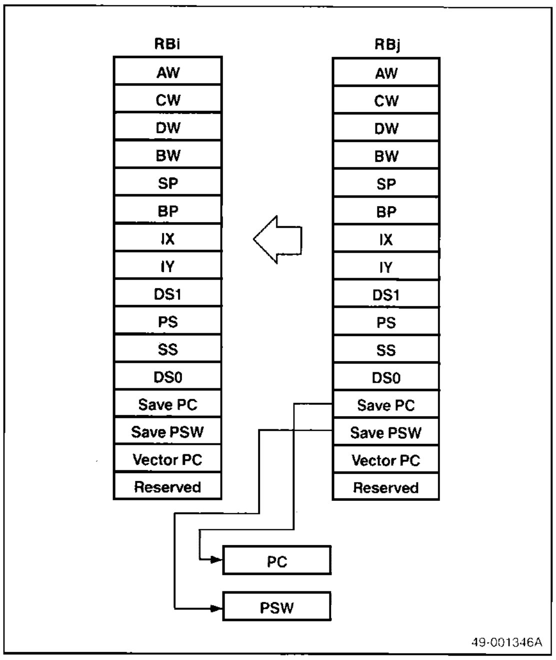


Figure 9. Register Bank Return



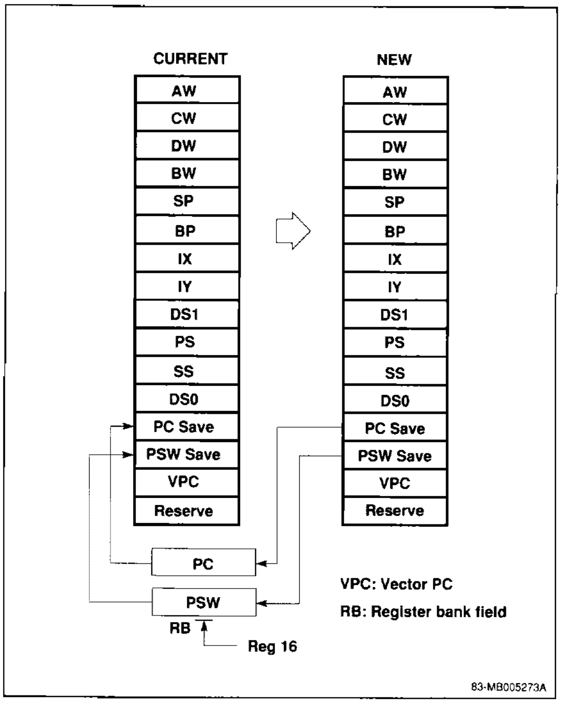
Macro Service Function

The macro service function (MSF) is a special micro-program that acts as an internal DMA controller between on-chip peripherals (special function registers, SFR) and memory. The MSF greatly reduces the software overhead and CPU time that other processors would require for register save processing, register returns, and other handling associated with interrupt processing.

If the MSF is selected for a particular interrupt, each time the request is received, a byte or word of data will be transferred between the SFR and memory without interrupting the CPU. Each time a request occurs, the macro service counter is decremented. When the counter reaches zero, an interrupt to the CPU is generated. The MSF also has a character search option. When selected, every byte transferred will be compared to an 8-bit search character and an interrupt will be generated if a match occurs or if the macro service counter counts out.

Like the NMI, INT and INTTB, the two DMA controller interrupts (INTD0, INTD1) do not have MSF capability.

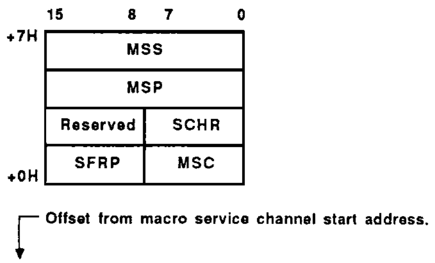
Figure 10. Task Switching



There are eight 8-byte macro service channels mapped into internal RAM from XXE00H to XXE3FH. Figure 11 shows the components of each channel.

Setting the macro service mode for a given interrupt requires programming the corresponding macro service control register. Each individual interrupt, excluding INTR, NMI and TBC, has its own associated MSC register. See table 6. Format for all MSC registers is shown in figure 12.

Figure 11. Macro Service Channels



MSS	+6H	Segment value of memory address used for data transfer. Memory address will be MSS x 16 + MSP.
MSP	+4H	Offset value of memory address used for data transfer.
SCHR	+2H	8-bit data compared in character search.
SFRP	+1H	Offset value of special function register address, which is xxF00H + SFRP. (xx is specified by IDB register).
MSC	+0H	Number of transfers performed in macro service

83M-005285A

On-Chip Peripherals

Timer Unit

The μPD70330/332 (figure 13) has two programmable 16-bit interval timers (TM0, TM1) on-chip, each with variable input clock frequencies. Each of the two 16-bit timer registers has an associated 16-bit modulus register (MD0, MD1). Timer 0 operates in the interval timer mode or one-shot mode; timer 1 has only the interval timer mode.

Interval Timer Mode. In this mode, TM0/TM1 are decremented by the selected input clock and, after counting out, the registers are automatically reloaded from the modulus registers and counting continues. Each time TM1 counts out, interrupts are generated through TF1 and TF2 (Timer Flags 1, 2). When TM0 counts out, an interrupt is generated through TF0. The timer-out signal can be used as a square-wave output whose half-cycle is equal to the count time. There are two selectable input clocks (SCLK: system clock = $f_{OSC}/2$; f_{OSC} = 10 MHz).

Clock	Timer Resolution	Full Count
SCLK/6	1.2 μs	78.643 ms
SCLK/128	25.6 μs	1.678 s

One-Shot Mode. In the one-shot mode, TM0 and MD0 operate as independent one-shot timers. Starting with a preset value, each is decremented to zero. At zero, counting ceases and an interrupt is generated by TF0 (from TM0) or TF1 (from MD0). One-shot mode allows two selectable input clocks (f_{OSC} = 10 MHz).

Clock	Timer Resolution	Full Count
SCLK/12	2.4 μs	157.283 ms
SCLK/128	25.6 μs	1.678 s

Setting the desired timer mode requires programming the timer control register. See figures 14 and 15 for format.

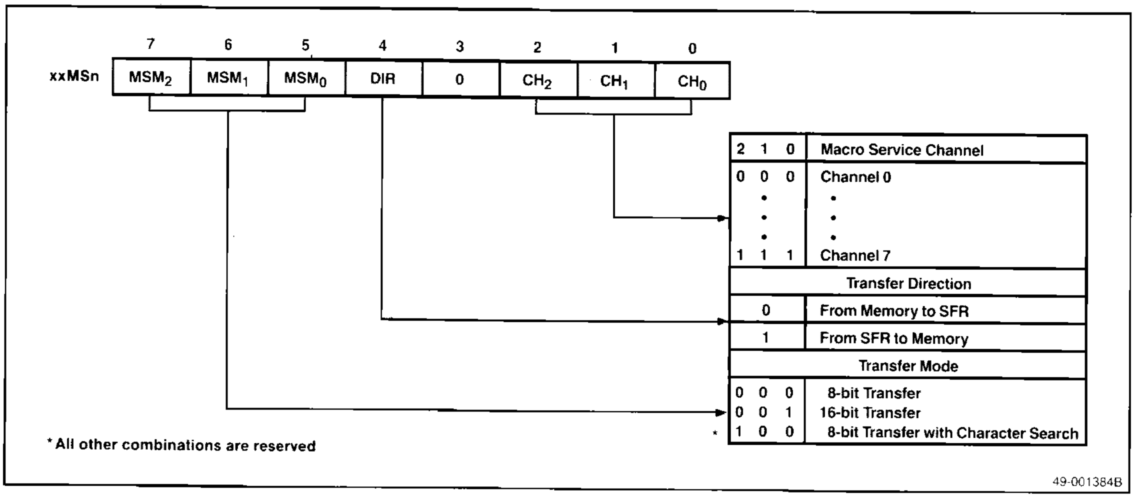
Time Base Counter/Processor Control Register

The 20-bit free-running time base counter controls internal timing sequences and is available to the user as the source of periodic interrupts at lengthy intervals. One of four interrupt periods can be selected by programming the TB0 and TB1 bits in the processor control register (PRC). The TBC interrupt is unlike the others in that it is fixed as a level 7 vectored interrupt. Macro service and register bank switching cannot be used to service this interrupt. See figures 16 and 17.

The RAMEN bit in the PRC register allows the internal RAM to be removed from the memory address space to implement faster instruction execution.

The TBC (figure 18) uses the system clock as the input frequency. The system clock can be changed by programming the PCK0 and PCK1 bits in the processor control register (PRC). Reset initializes the system clock to $f_{OSC}/8$ (f_{OSC} = external oscillator frequency).

Figure 12. Macro Service Control Registers (MSC)



xxMSn
7 6 5 4 3 2 1 0
MSM2 MSM1 MSM0 DIR 0 CH2 CH1 CH0
*All other combinations are reserved
49-001384B

2	1	0	Macro Service Channel
0	0	0	Channel 0
	•		•
	•		•
	•		•
1	1	1	Channel 7

Transfer Direction	
0	From Memory to SFR
1	From SFR to Memory

			Transfer Mode
0	0	0	8-bit Transfer
0	0	1	16-bit Transfer
1	0	0	8-bit Transfer with Character Search

Figure 13. Timer Unit Block Diagram

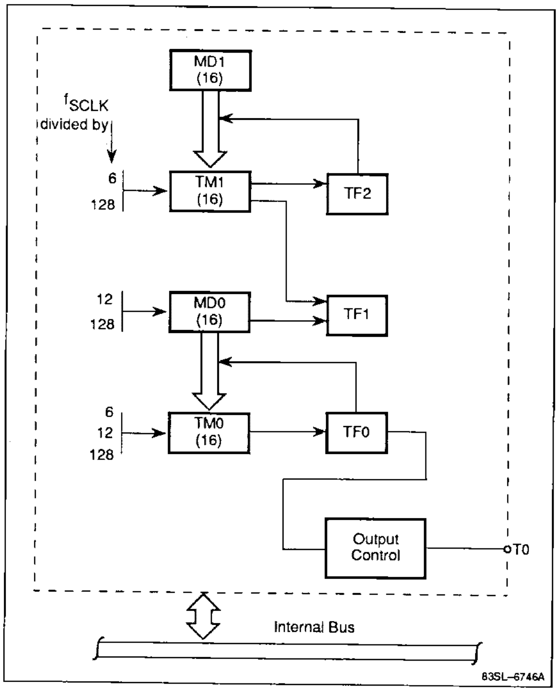


Figure 14. Timer Control Register 0

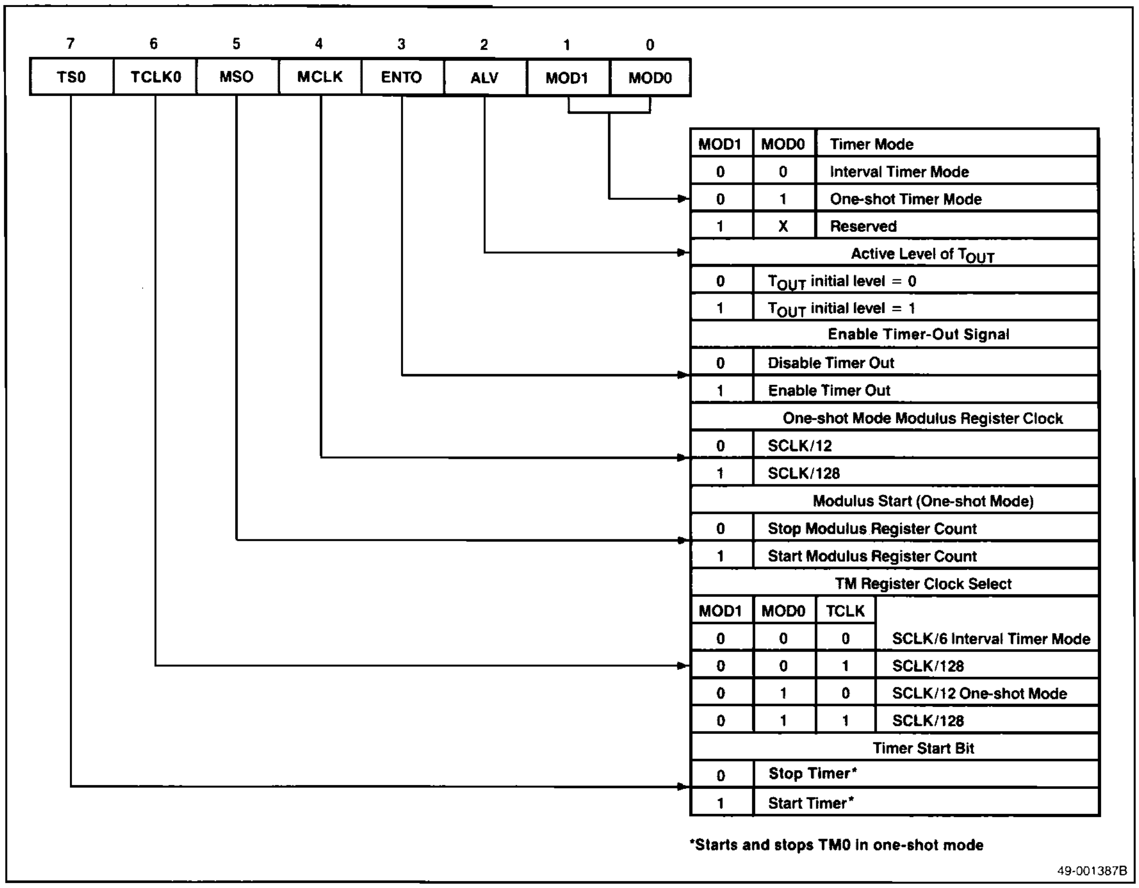

7 6 5 4 3 2 1 0
TS0 TCLK0 MSO MCLK ENTO ALV MOD1 MOD0
MOD1 MOD0 Timer Mode
0 0 Interval Timer Mode
0 1 One-shot Timer Mode
1 X Reserved
Active Level of T_{OUT}
0 T_{OUT} initial level = 0
1 T_{OUT} initial level = 1
Enable Timer-Out Signal
0 Disable Timer Out
1 Enable Timer Out
One-shot Mode Modulus Register Clock
0 SCLK/12
1 SCLK/128
Modulus Start (One-shot Mode)
0 Stop Modulus Register Count
1 Start Modulus Register Count
TM Register Clock Select
MOD1 MOD0 TCLK
0 0 0 SCLK/6 Interval Timer Mode
0 0 1 SCLK/128
0 1 0 SCLK/12 One-shot Mode
0 1 1 SCLK/128
Timer Start Bit
0 Stop Timer*
1 Start Timer*
*Starts and stops TM0 in one-shot mode
49-001387B

Figure 15. Timer Control Register 1

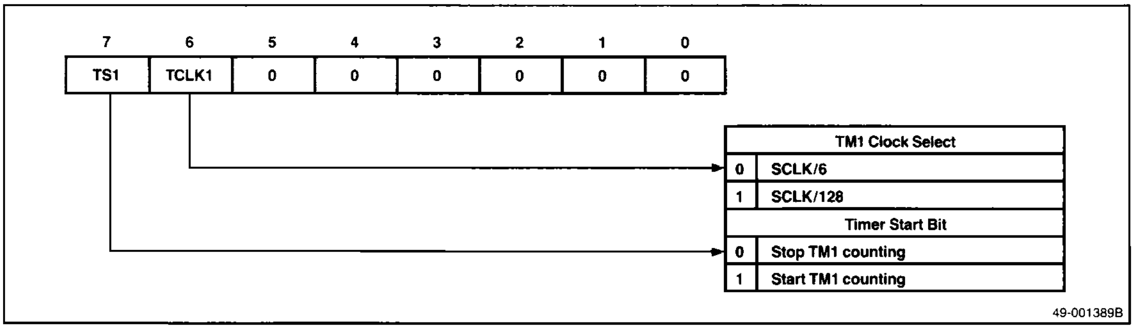
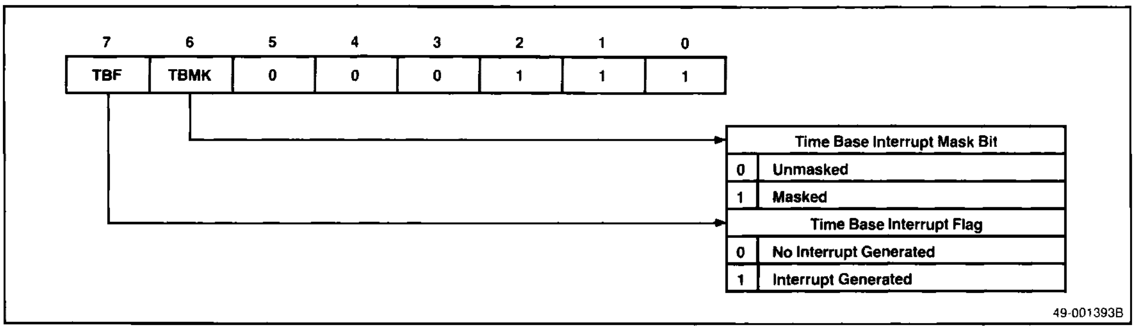


Figure 16. Time Base Interrupt Request Control Register

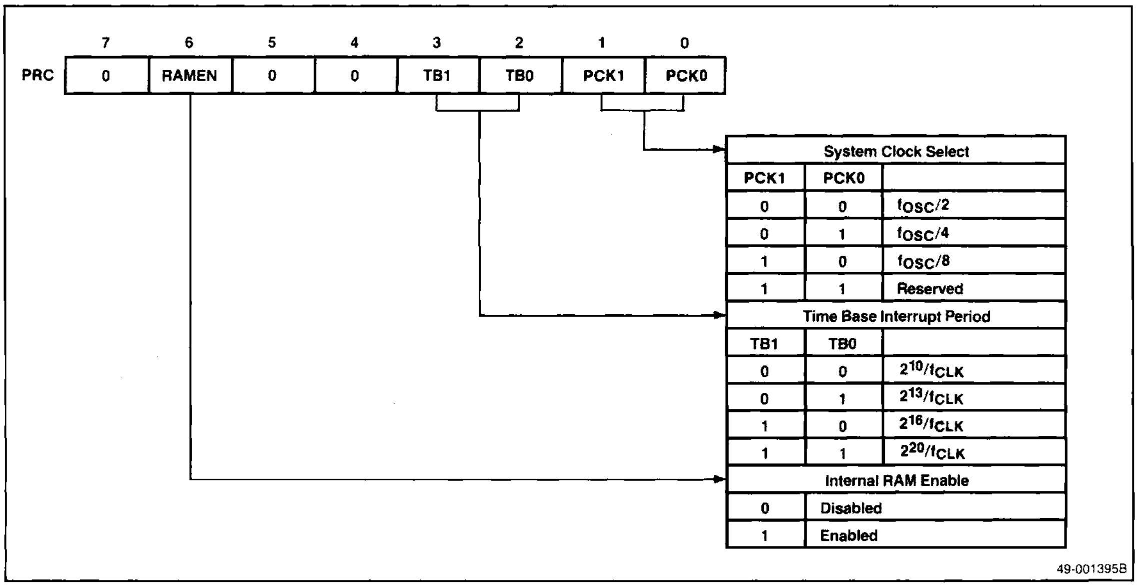


7	6	5	4	3	2	1	0
TBF	TBMK	0	0	0	1	1	1

Time Base Interrupt Mask Bit	
0	Unmasked
1	Masked

Time Base Interrupt Flag	
0	No Interrupt Generated
1	Interrupt Generated

Figure 17. Processor Control Register (PRC)



	7	6	5	4	3	2	1	0
PRC	0	RAMEN	0	0	TB1	TB0	PCK1	PCK0

System Clock Select

PCK1	PCK0	
0	0	fosc/2
0	1	fosc/4
1	0	fosc/8
1	1	Reserved

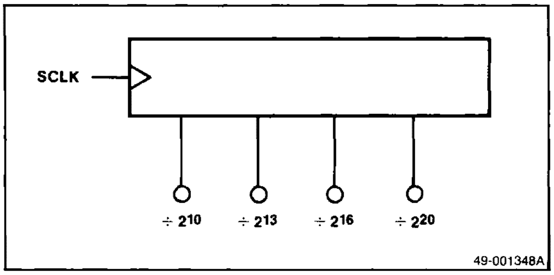
Time Base Interrupt Period

TB1	TB0	
0	0	$2^{10}/f_{CLK}$
0	1	$2^{13}/f_{CLK}$
1	0	$2^{16}/f_{CLK}$
1	1	$2^{20}/f_{CLK}$

Internal RAM Enable

0	Disabled
1	Enabled

Figure 18. Time Base Counter (TBC) Block Diagram



Refresh Controller

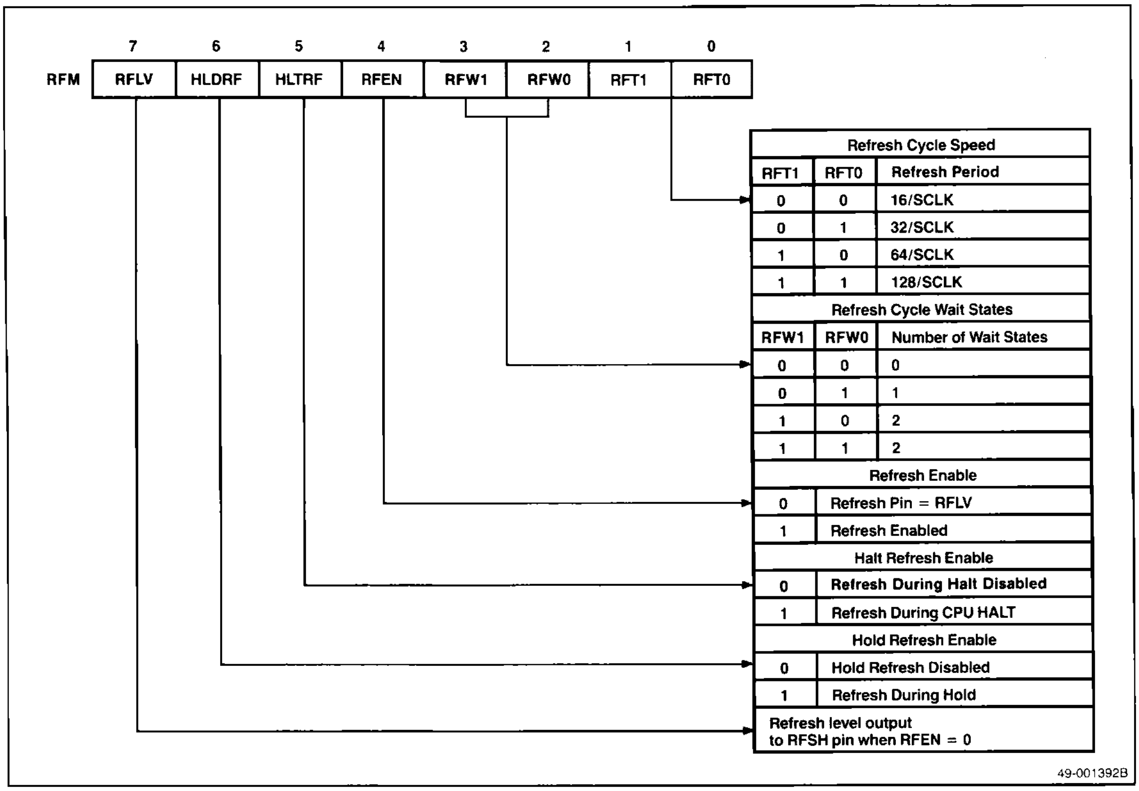
The μPD70330/332 has an on-chip refresh controller for dynamic and pseudostatic RAM mass storage memories. The refresh controller generates refresh addresses and refresh pulses. It inserts refresh cycles between the normal CPU bus cycles according to refresh specifications.

The refresh controller outputs a 9-bit refresh address on address bits A_0-A_8 during the refresh bus cycle. Address bits A_9-A_{19} are all 1's. The 9-bit refresh address is automatically incremented at every refresh timing for 512 row addresses. The 8-bit refresh mode (RFM) register (figure 19) specifies the refresh operation and allows refresh during both CPU HALT and HOLD modes. Refresh cycles are automatically timed to $\overline{\text{REFRQ}}$ following read/write cycles to minimize the effect on system thoughput.

The following shows the $\overline{\text{REFRQ}}$ pin level in relation to bits 4 (RFEN) and 7 (RFLV) of the refresh mode register.

RFEN	RFLV	$\overline{\text{REFRQ}}$ Level
0	0	0
0	1	1
1	0	0
1	1	Refresh pulse output

Figure 19. Refresh Mode Register (RFM)



Refresh Cycle Speed		
RFT1	RFT0	Refresh Period
0	0	16/SCLK
0	1	32/SCLK
1	0	64/SCLK
1	1	128/SCLK

Refresh Cycle Wait States		
RFW1	RFW0	Number of Wait States
0	0	0
0	1	1
1	0	2
1	1	2

Refresh Enable	
0	Refresh Pin = RFLV
1	Refresh Enabled

Halt Refresh Enable	
0	Refresh During Halt Disabled
1	Refresh During CPU HALT

Hold Refresh Enable	
0	Hold Refresh Disabled
1	Refresh During Hold

Refresh level output to RFSH pin when RFEN = 0

Serial Interface

The μPD70330/332 has two full-duplex UARTs, channel 0 and channel 1. Each serial port channel has a transmit line (TxDn), a receive line (RxDn), and a clear to send (CTSn) input line for handshaking. Communication is synchronized by a start bit, and you can program the ports for even, odd, or no parity, character lengths of 7 or 8 bits, and 1 or 2 stop bits.

The μPD70330/332 has dedicated baud rate generators for each serial channel. This eliminates the need to obligate the on-chip timers. The baud rate generator allows a wide range of data transfer rates (up to 1.25 Mb/s). This includes all of the standard baud rates without being restricted by the value of the particular external crystal.

Each baud rate generator has an 8-bit baud rate generator (BRGn) data register, which functions as a prescaler to a programmable input clock selected by the serial communication control (SCCn) register. Together these must be set to generate a frequency equivalent to the desired baud rate.

The baud rate generator can be set to obtain the desired transmission rate according to the following formula:

$$B \times G = \frac{SCLK \times 10^6}{2^{n+1}}$$

where B = baud rate
G = baud rate generator register (BRGn) value
n = input clock specifications (n between 0 and 8). This is the value that is loaded into the SCCn register. See figure 23.
SCLK = system clock frequency (MHz)

Based on the above expression, the following table shows the baud rate generator values used to obtain standard transmission rates when SCLK = 5 MHz.

Baud Rate	n	BRGn Value	Error (%)
110	7	178	0.25
150	7	130	0.16
300	6	130	0.16
600	5	130	0.16
1200	4	130	0.16
2400	3	130	0.16
4800	2	130	0.16
9600	1	130	0.16
19,200	0	130	0.16
38,400	0	65	0.16
1.25M	0	2	0

In addition to the asynchronous mode, channel 0 has a synchronous I/O interface mode. In this mode, each bit of data tranferred is synchronized to a serial clock ($\overline{SCKO}$). This is the same as the NEC μCOM75 and μCOM87 series, and allows easy interfacing to these devices. Figure 20 is the serial interface block diagram; figures 21, 22, and 23 show the three serial communication registers.

DMA Controller

The μPD70330/332 has a two-channel, on-chip DMA controller. This allows rapid data transfer between memory and auxiliary storage devices. The DMA controller supports four modes of operation, two for memory-to-memory transfers and two for transfers between I/O and memory. See figures 24, 25, and 26 for a graphic representation of the DMA registers.

Figure 20. Serial Interface Block Diagram

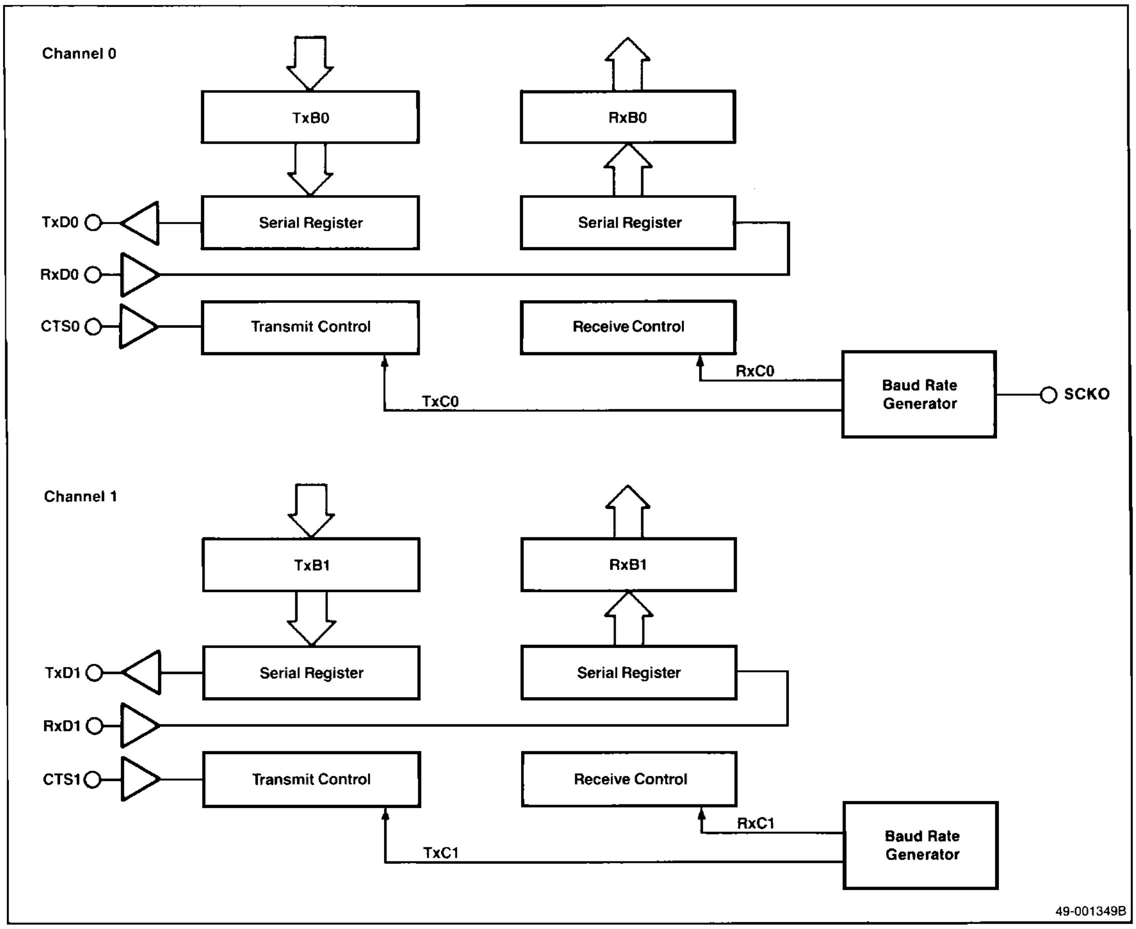
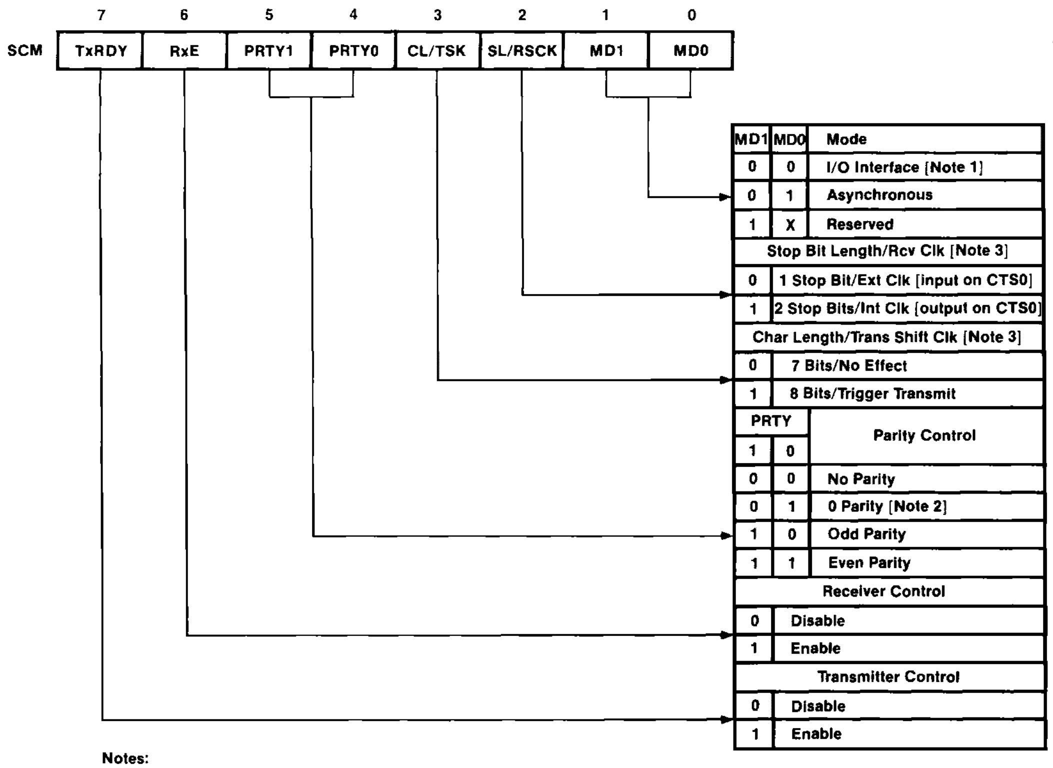


Figure 21. Serial Communication Mode Register (SCM)



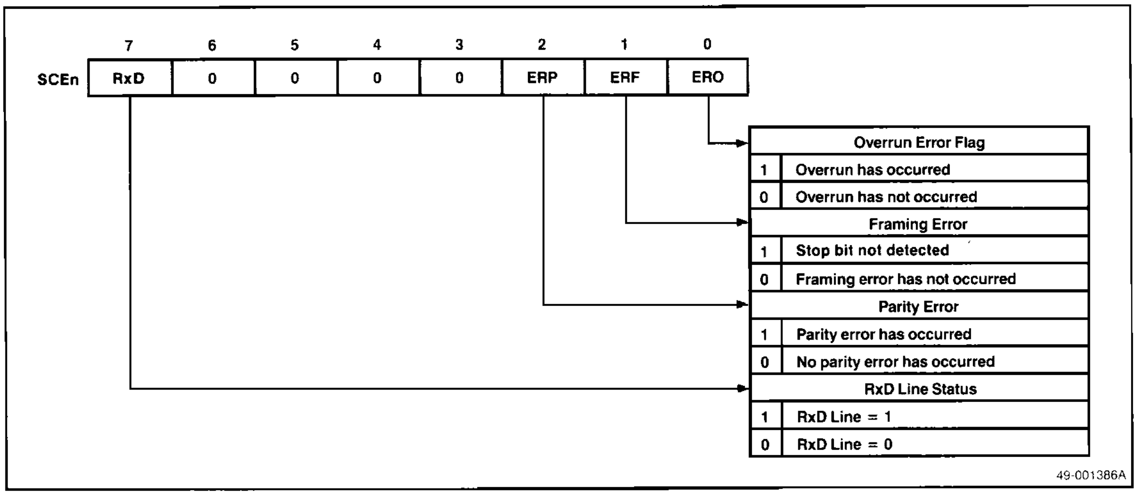
MD1	MD0	Mode
0	0	I/O Interface [Note 1]
0	1	Asynchronous
1	X	Reserved
		Stop Bit Length/Rcv Clk [Note 3]
0		1 Stop Bit/Ext Clk [input on CTS0]
1		2 Stop Bits/Int Clk [output on CTS0]
		Char Length/Trans Shift Clk [Note 3]
0		7 Bits/No Effect
1		8 Bits/Trigger Transmit
PRTY		Parity Control
1	0	
0	0	No Parity
0	1	0 Parity [Note 2]
1	0	Odd Parity
1	1	Even Parity
		Receiver Control
0		Disable
1		Enable
		Transmitter Control
0		Disable
1		Enable

Notes:
[1] Only Channel 0 has I/O interface mode.
[2] When 0 parity is selected, the parity is 0 during transmit and is ignored during receive.
[3] Applies only to I/O interface mode.

49-001385B

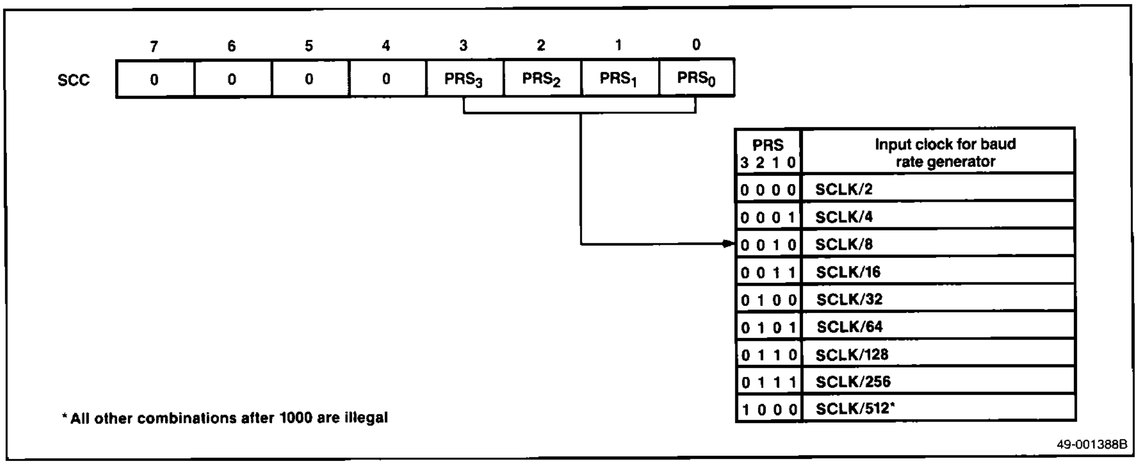
4b

Figure 22. Serial Communication Error Registers (SCE)



	Overrun Error Flag
1	Overrun has occurred
0	Overrun has not occurred
	Framing Error
1	Stop bit not detected
0	Framing error has not occurred
	Parity Error
1	Parity error has occurred
0	No parity error has occurred
	RxD Line Status
1	RxD Line = 1
0	RxD Line = 0

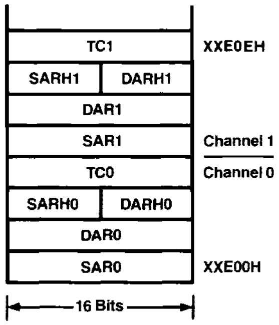
Figure 23. Serial Communication Control Register (SCC)



PRS 3 2 1 0	Input clock for baud rate generator
0 0 0 0	SCLK/2
0 0 0 1	SCLK/4
0 0 1 0	SCLK/8
0 0 1 1	SCLK/16
0 1 0 0	SCLK/32
0 1 0 1	SCLK/64
0 1 1 0	SCLK/128
0 1 1 1	SCLK/256
1 0 0 0	SCLK/512*

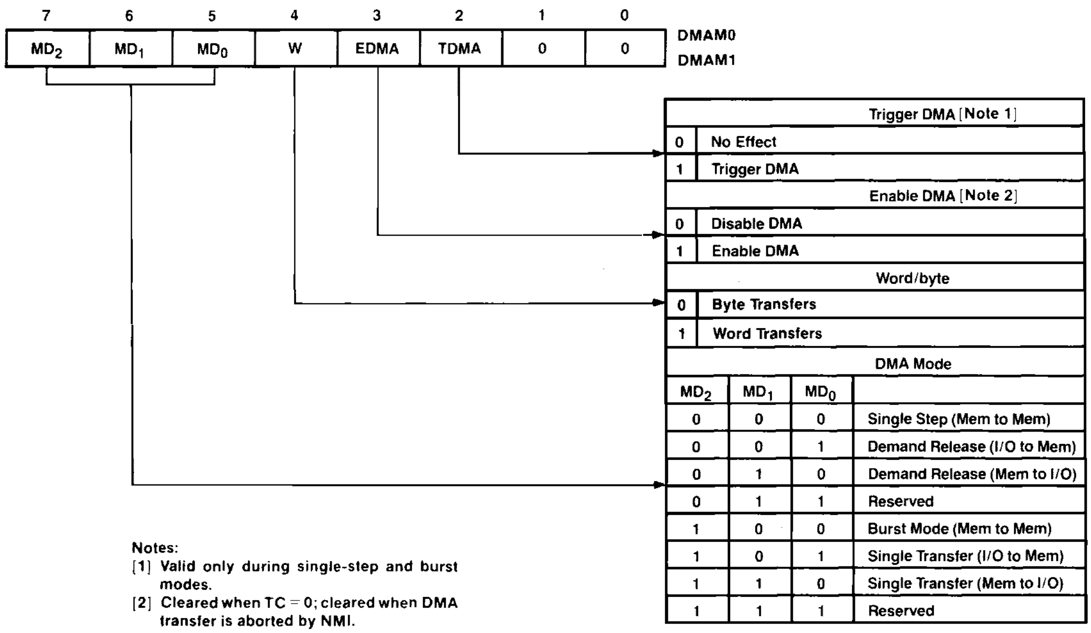
*All other combinations after 1000 are illegal

Figure 24. DMA Channels



49-001350A

Figure 25. DMA Mode Registers (DMAM)



7	6	5	4	3	2	1	0
MD_2	MD_1	MD_0	W	EDMA	TDMA	0	0

Trigger DMA [Note 1]	
0	No Effect
1	Trigger DMA

Enable DMA [Note 2]	
0	Disable DMA
1	Enable DMA

Word/byte	
0	Byte Transfers
1	Word Transfers

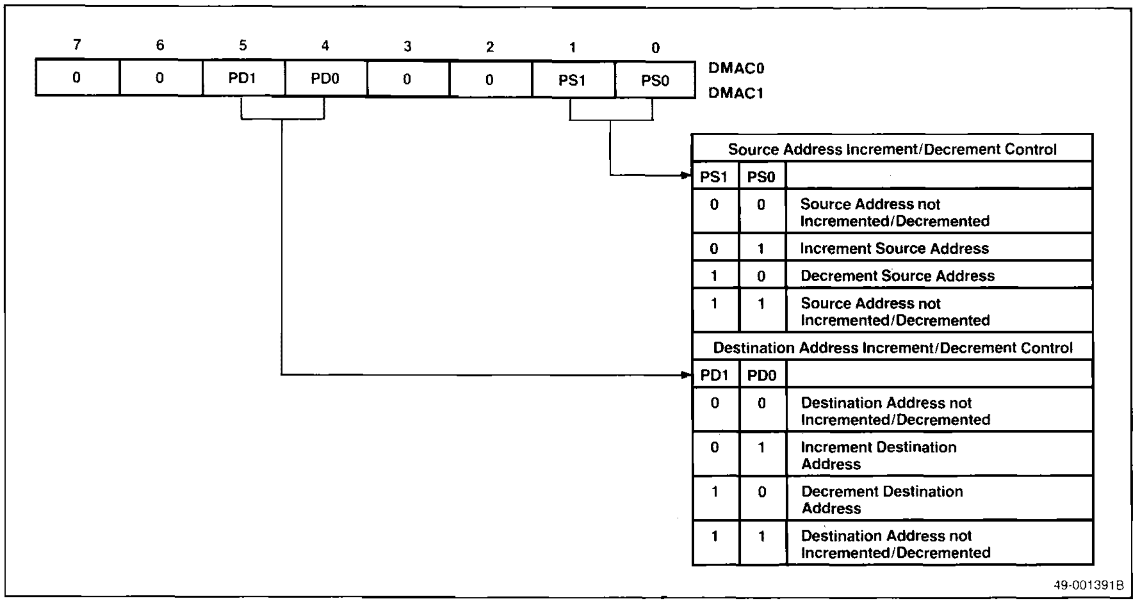
DMA Mode			
MD_2	MD_1	MD_0	
0	0	0	Single Step (Mem to Mem)
0	0	1	Demand Release (I/O to Mem)
0	1	0	Demand Release (Mem to I/O)
0	1	1	Reserved
1	0	0	Burst Mode (Mem to Mem)
1	0	1	Single Transfer (I/O to Mem)
1	1	0	Single Transfer (Mem to I/O)
1	1	1	Reserved

Notes:
[1] Valid only during single-step and burst modes.
[2] Cleared when TC = 0; cleared when DMA transfer is aborted by NMI.

49-001390B

4b

Figure 26. DMA Control Registers (DMAC)



Source Address Increment/Decrement Control		
PS1	PS0	
0	0	Source Address not Incremented/Decremented
0	1	Increment Source Address
1	0	Decrement Source Address
1	1	Source Address not Incremented/Decremented

Destination Address Increment/Decrement Control		
PD1	PD0	
0	0	Destination Address not Incremented/Decremented
0	1	Increment Destination Address
1	0	Decrement Destination Address
1	1	Destination Address not Incremented/Decremented

Memory-to-Memory Transfers. In the single-step mode, when one DMA request is made, execution of one instruction and one DMA transfer are repeated alternately until the prescribed number of DMA transfers has occurred. Interrupts can be accepted while in this mode. In burst mode, a DMA request causes DMA transfer cycles to continue until the DMA terminal counter decrements to zero. Software can also initiate memory-to-memory transfers.

Transfers Between I/O and Memory. In single-transfer mode, one DMA transfer occurs after each rising edge of DMARQ. After the transfer, the bus is returned to the CPU. In demand release mode, the rising edge of DMARQ enables DMA cycles, which continue as long as DMARQ is high.

In all modes, the $\overline{\text{TC}}$ (terminal count) output pin will pulse low and a DMA completion I/O request will be generated after the predetermined number of DMA cycles has been completed.

The bottom of internal RAM contains all the necessary address information for the designated DMA channels. The DMA channel mnemonics are as follows:

TC	Terminal counter
SAR	Source address register
SARH	Source address register high
DAR	Destination address register
DARH	Destination address register high

The DMA controller generates physical source addresses by offsetting SARH 12 bits to the left and then adding the SAR. The same procedure is also used to generate physical destination addresses. You can program the controller to increment or decrement source and/or destination addresses independently during DMA transfers.

When the EDMA bit is set, the internal DMARQ flag is cleared. Therefore, DMARQs are only recognized after the EDMA bit has been set.

Parallel Ports

The μPD70330/332 has three 8-bit parallel I/O ports: P0, P1, and P2. Refer to figures 27 through 31. Special function register (SFR) locations can access these ports. The port lines are individually programmable as inputs or outputs. Many of the port lines have dual functions as port or control lines.

Use the associated port mode and port mode control registers to select the mode for a given I/O line.

The analog comparator port (PT) compares each input line to a reference voltage. The reference voltage is programmable to be the V_{TH} input x n/16, where n = 1 to 16. See figure 32.

Figure 27. Port Mode Registers 0 and 2 (PM0, PM2)

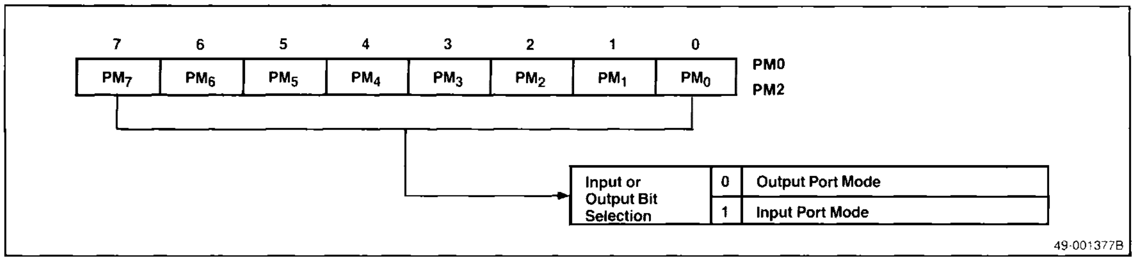
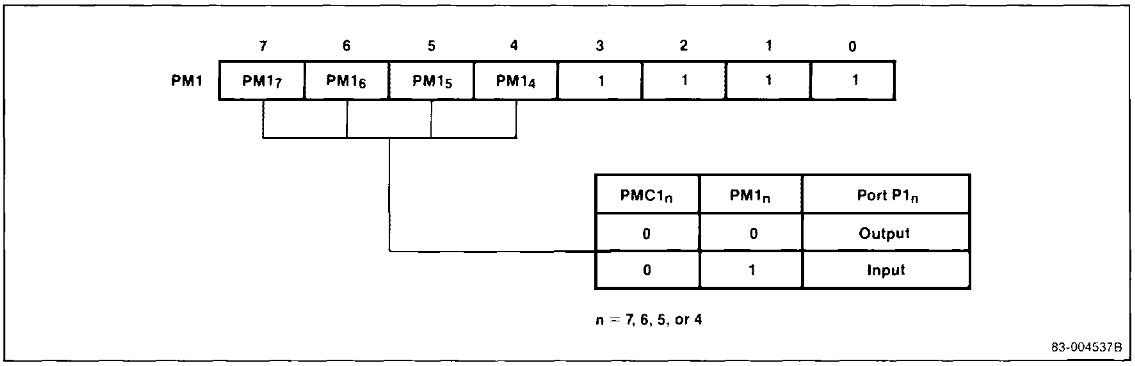


Figure 28. Port Mode Register 1 (PM1)



$PMC1_n$	$PM1_n$	Port $P1_n$
0	0	Output
0	1	Input

Figure 29. Port Mode Control Register 0 (PMC0)

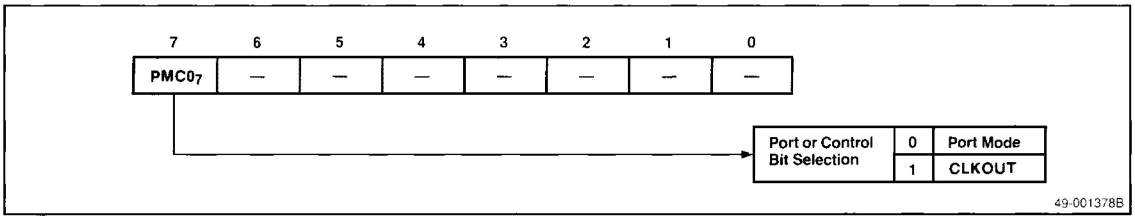
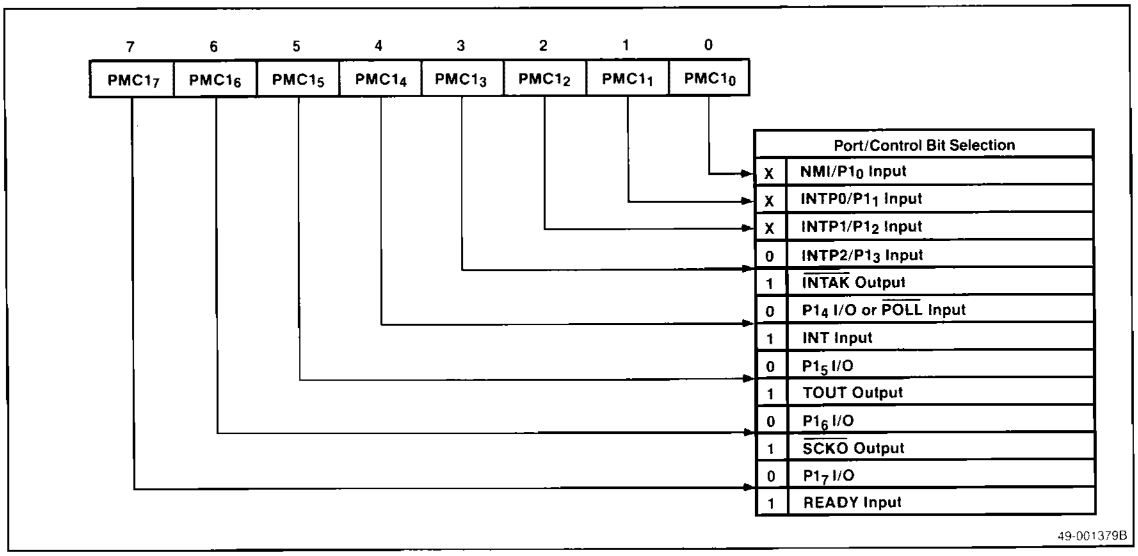
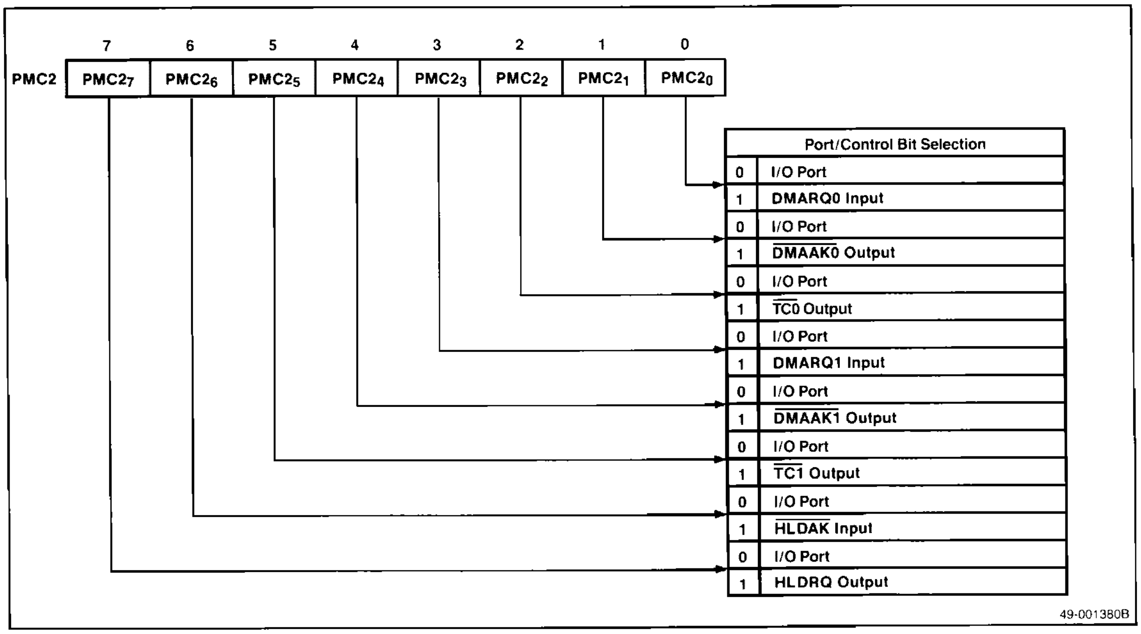


Figure 30. Port Mode Control Register 1 (PMC1)



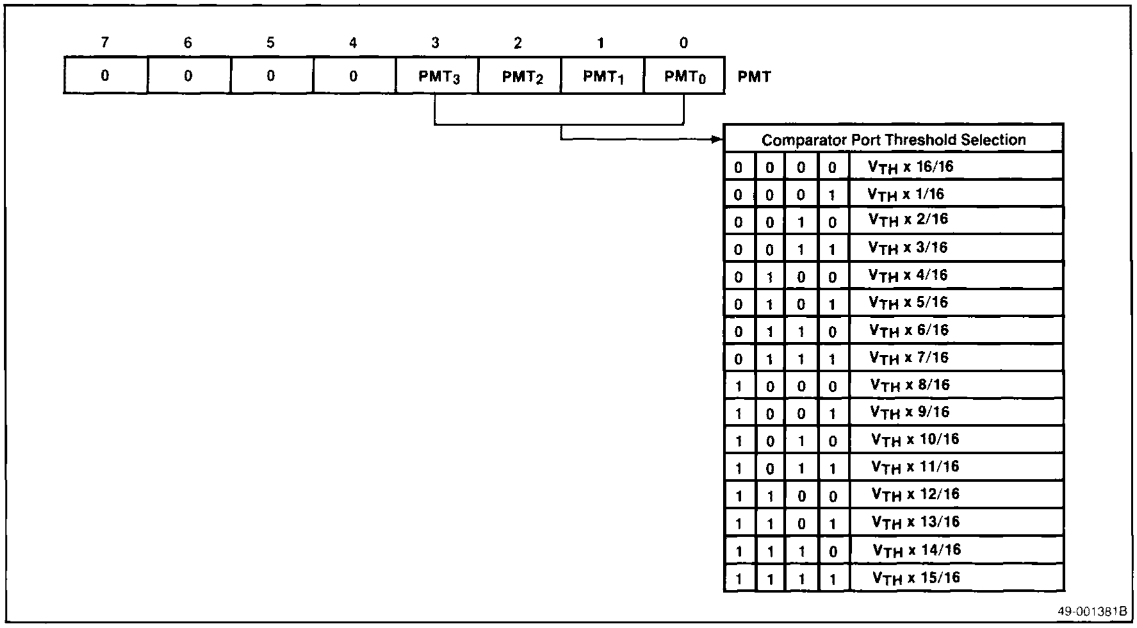
7 6 5 4 3 2 1 0
PMC1$_7$ PMC1$_6$ PMC1$_5$ PMC1$_4$ PMC1$_3$ PMC1$_2$ PMC1$_1$ PMC1$_0$
Port/Control Bit Selection
X NMI/P1$_0$ Input
X INTP0/P1$_1$ Input
X INTP1/P1$_2$ Input
0 INTP2/P1$_3$ Input
1 $\overline{\text{INTAK}}$ Output
0 P1$_4$ I/O or $\overline{\text{POLL}}$ Input
1 INT Input
0 P1$_5$ I/O
1 TOUT Output
0 P1$_6$ I/O
1 $\overline{\text{SCKO}}$ Output
0 P1$_7$ I/O
1 READY Input
49-001379B

Figure 31. Port Mode Control Register 2 (PMC2)



7 6 5 4 3 2 1 0
PMC2
PMC2$_7$ PMC2$_6$ PMC2$_5$ PMC2$_4$ PMC2$_3$ PMC2$_2$ PMC2$_1$ PMC2$_0$
Port/Control Bit Selection
0 I/O Port
1 DMARQ0 Input
0 I/O Port
1 $\overline{\text{DMAAK0}}$ Output
0 I/O Port
1 $\overline{\text{TC0}}$ Output
0 I/O Port
1 DMARQ1 Input
0 I/O Port
1 $\overline{\text{DMAAK1}}$ Output
0 I/O Port
1 $\overline{\text{TC1}}$ Output
0 I/O Port
1 HLDAK Input
0 I/O Port
1 HLDRQ Output
49-001380B

Figure 32. Port Mode Register T (PMT)



7	6	5	4	3	2	1	0	
0	0	0	0	PMT_3	PMT_2	PMT_1	PMT_0	PMT

Comparator Port Threshold Selection				
0	0	0	0	V_{TH} x 16/16
0	0	0	1	V_{TH} x 1/16
0	0	1	0	V_{TH} x 2/16
0	0	1	1	V_{TH} x 3/16
0	1	0	0	V_{TH} x 4/16
0	1	0	1	V_{TH} x 5/16
0	1	1	0	V_{TH} x 6/16
0	1	1	1	V_{TH} x 7/16
1	0	0	0	V_{TH} x 8/16
1	0	0	1	V_{TH} x 9/16
1	0	1	0	V_{TH} x 10/16
1	0	1	1	V_{TH} x 11/16
1	1	0	0	V_{TH} x 12/16
1	1	0	1	V_{TH} x 13/16
1	1	1	0	V_{TH} x 14/16
1	1	1	1	V_{TH} x 15/16

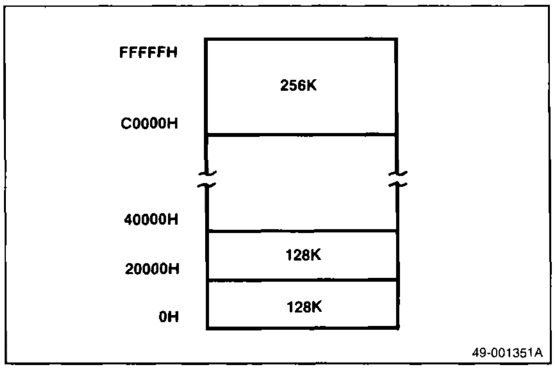
Programmable Wait State Generation

You can generate wait states internally to further reduce the necessity for external hardware. Insertion of these wait states allows direct interface to devices whose access times cannot meet the CPU read/write timing requirements.

When using this function, the entire 1M-byte memory address space is divided into 128K-blocks. Each block can be programmed for zero, one, or two wait states, or two plus those added by the extenal READY signal. The top two blocks are programmed together as one unit.

The appropriate bits in the wait control word (WTC) control wait state generation. Programming the upper two bits in the wait control word will set the wait state conditions for the entire I/O address space. Figure 33 shows the memory map for programmable wait state generation; see figure 34 for a graphic representation of the wait control word.

Figure 33. Programmable Wait State Generation



Standby Modes

The two low-power standby modes are HALT and STOP. Software causes the processor to enter either mode.

HALT Mode.

In the HALT mode, the processor is inactive and the chip consumes much less power than when operational. The external oscillator remains functional and all peripherals are active. Internal status and output port line conditions are maintained. Any unmasked interrupt can release this mode. In the EI state, interrupts subsequently will be processed in vector mode. In the DI state, program execution is restarted with the instruction following the HALT instruction.

STOP Mode.

The STOP mode allows the largest power reduction while maintaining RAM. The oscillator is stopped, halting all internal peripherals. Internal status is maintained. Only a reset or NMI can release this mode.

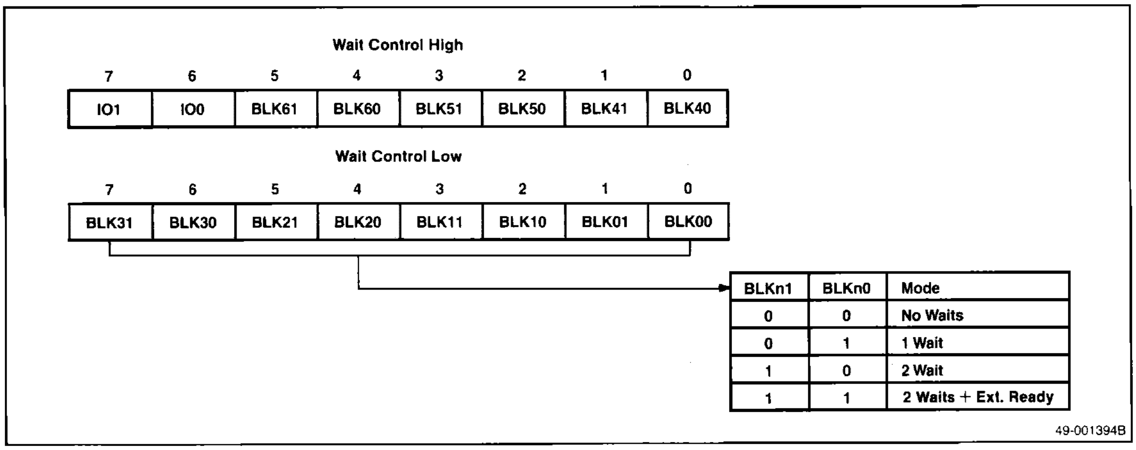
A standby flag in the SFR area is reset by rises in the supply voltage. Its status is maintained during normal operation and standby. The STBC register (figure 35) is not initialized by RESET. Use the standby flag to determine whether program execution is returning from standby or from a cold start by setting this flag before entering the STOP mode.

Special Function Registers

Table 6 shows the special function register mnemonic, type, address, reset value, and function. The 8 high-order bits of each address (xx) are specified by the IDB register.

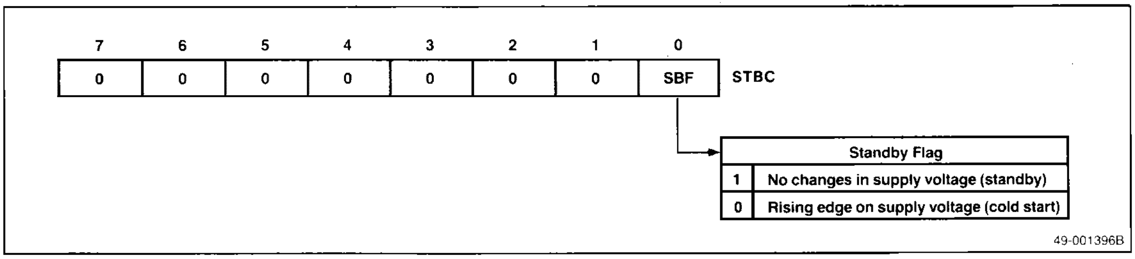
SFR area addresses not listed in table 6 are reserved. If read, the contents of these addresses are undefined, and any write operation will be meaningless.

Figure 34. Wait Control Word



BLKn1	BLKn0	Mode
0	0	No Waits
0	1	1 Wait
1	0	2 Wait
1	1	2 Waits + Ext. Ready

Figure 35. Standby Register



Standby Flag	
1	No changes in supply voltage (standby)
0	Rising edge on supply voltage (cold start)

Table 6. ***Special-Function Registers***

Address	Register Function	Symbol	R/W	Manipulation (Bit)	When RESET
xxF00H	Port 0	P0	R/W	8/1	Undefined
xxF01H	Port mode 0	PM0	W	8	FFH
xxF02H	Port mode control 0	PMC0	W	8	00H
xxF08H	Port 1	P1	R/W	8/1	Undefined
xxF09H	Port mode 1	PM1	W	8	FFH
xxF0AH	Port mode control 1	PMC1	W	8	00H
xxF10H	Port 2	P2	R/W	8/1	Undefined
xxF11H	Port mode 2	PM2	W	8	FFH
xxF12H	Port mode control 2	PMC2	W	8	00H
xxF38H	Port T	PT	R	8	Undefined
xxF3BH	Port mode T	PMT	R/W	8/1	00H
xxF40H	External interrupt mode	INTM	R/W	8/1	00H
xxF44H	External interrupt macro service control 0	EMS0	R/W	8/1	Undefined
xxF45H	External interrupt macro service control 1	EMS1	R/W	8/1	
xxF46	External interrupt macro service control 2	EMS2	R/W	8/1	
xxF4CH	External interrupt request control 0	EXIC0	R/W	8/1	47H
xxF4DH	External interrupt request control 1	EXIC1	R/W	8/1	
xxF4EH	External interrupt request control 2	EXIC2	R/W	8/1	
xxF60H	Receive buffer 0	RxB0	R	8	Undefined
xxF62H	Transmit buffer 0	TxB0	W	8	
xxF65H	Serial receive macro service control 0	SRMS0	R/W	8/1	
xxF66H	Serial transmit macro service control 0	STMS0	R/W	8/1	
xxF68H	Serial communication mode 0	SCM0	R/W	8/1	
xxF69H	Serial communication control 0	SCC0	R/W	8/1	00H
xxF6AH	Baud rate generator 0	BRG0	R/W	8/1	
xxF6BH	Serial communication error 0	SCE0	R	8	
xxF6CH	Serial error interrupt request control 0	SEIC0	R/W	8/1	47H
xxF6DH	Serial receive interrupt request control 0	SRIC0	R/W	8/1	
xxF6EH	Serial transmit interrupt request control 0	STIC0	R/W	8/1	
xxF70H	Receive buffer 1	RxB1	R	8	Undefined
xxF72H	Transmit buffer 1	TxB1	W	8	
xxF75H	Serial receive macro service control 1	SRMS1	R/W	8/1	
xxF76H	Serial transmit macro service control 1	STMS1	R/W	8/1	
xxF78H	Serial communication mode 1	SCM1	R/W	8/1	00H
xxF79H	Serial communication control 1	SCC1	R/W	8/1	
xxF7AH	Baud rate generator 1	BRG1	R/W	8/1	
xxF7BH	Serial communication error 1	SCE1	R	8	

Table 6. Special-Function Registers (cont)

Address	Register Function	Symbol	R/W	Manipulation (Bit)	When RESET
xxF7CH	Serial error interrupt request control 1	SEIC1	R/W	8/1	47H
xxF7DH	Serial receive interrupt request control 1	SRIC1	R/W	8/1	
xxF7EH	Serial transmit interrupt request control 1	STIC1	R/W	8/1	
xxF80H	Timer 0	TM0	R/W	16	Undefined
xxF82H	Modulo 0	MD0	R/W	16	
xxF88H	Timer 1	TM1	R/W	16	
xxF8AH	Modulo 1	MD1	R/W	16	
xxF90H	Timer control 0	TMC0	R/W	8/1	00H
xxF91H	Timer control 1	TMC1	R/W	8/1	
xxF94H	Timer macro service control 0	TMMS0	R/W	8/1	Undefined
xxF95H	Timer macro service control 1	TMMS1	R/W	8/1	
xxF96H	Timer macro service control 2	TMMS2	R/W	8/1	
xxF9CH	Timer interrupt request control 0	TMIC0	R/W	8/1	47H
xxF9DH	Timer interrupt request control 1	TMIC1	R/W	8/1	
xxF9EH	Timer interrupt request control 2	TMIC2	R/W	8/1	
xxFA0H	DMA control 0	DMAC0	R/W	8/1	Undefined
xxFA1H	DMA mode 0	DMAM0	R/W	8/1	00H
xxFA2H	DMA control 1	DMAC1	R/W	8/1	Undefined
xxFA3H	DMA mode 1	DMAM1	R/W	8/1	
xxFACH	DMA interrupt request control 0	DIC0	R/W	8/1	47H
xxFADH	DMA interrupt request control 1	DIC1	R/W	8/1	
xxFE0H	Standby control	STBC	R/W (Note 1)	8/1	Undefined (Note 2)
xxFE1H	Refresh mode	RFM	R/W	8/1	FCH
xxFE8H	Wait control	WTC	R/W	16/8	FFFFH
xxFEAH	User flag (Note 3)	FLAG	R/W	8/1	00H
xxFEBH	Processor control	PRC	R/W	8/1	4EH
xxFECH	Time base interrupt request control	TBIC	R/W	8/1	47H
xxFFCH	Inservice priority register	ISPR	R	8	Undefined
xxFFFH FFFFFH	Internal data area base	IDB	R/W	8/1	FFH

Notes:

(1) Each bit of the standby control register can be set to 1 by an instruction; however, once set, bits cannot be reset to 0 by an instruction (only 1 can be written to this register).

(2) Upon power-on reset = 00H; other = no change.

(3) For the user flag register (FLAG), manipulating bits other than bits 3 and 5 is meaningless. The contents of user flags 0 and 1 (F0 and F1) of the FLAG register are affected by manipulating F0 and F1 of the PSW.

Absolute Maximum Ratings

$T_A = 25°C$

Supply voltage, V_{DD}	−0.5 to +7.0 V
Input voltage, V_I	−0.5 to V_{DD} + 0.5 V (≤ +7.0 V)
Output voltage, V_O	−0.5 to V_{DD} + 0.5 V (≤ +7.0 V)
Threshold voltage, V_{TH}	−0.5 to V_{DD} + 0.5 V (≤ +7.0 V)
Output current, low; I_{OL}	
Each output pin	4.0 mA
Total	50 mA
Output current, high; I_{OH}	
Each output pin	−2.0 mA
Total	−20 mA
Operating temperature range, T_{OPT}	−40 to +85°C
Storage temperature range, 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.

Comparator Characteristics

V_{DD} = +5 V ±10%; T_A = −10 to +70°C

		Limits			Test
Parameter	Symbol	Min	Max	Unit	Conditions
Accuracy	VA_{COMP}		±100	mV	
Threshold voltage	V_{TH}	0	V_{DD} + 0.1	V	
Comparison time	t_{COMP}	64	65	t_{CYK}	
PT input voltage	V_{IPT}	0	V_{DD}	V	

Capacitance Characteristics

V_{DD} = 0 V; T_A = 25°C

		Limits			Test
Parameter	Symbol	Min	Max	Unit	Conditions
Input capacitance	C_I		10	pF	f_c = 1 MHz; Unmeasured pins returned to 0 V
Output capacitance	C_O		20	pF	
I/O capacitance	C_{IO}		20	pF	

DC Characteristics

V_{DD} = +5 V ±10%; T_A = −10 to +70°C (Note 1)

		Limits				Test
Parameter	Symbol	Min	Typ	Max	Unit	Conditions
Supply current, operating	I_{DD1}		50	100	mA	f_{CLK} = 5 MHz
			65	120	mA	f_{CLK} = 8 MHz
Supply current, HALT mode	I_{DD2}		20	40	mA	f_{CLK} = 5 MHz
			25	50	mA	f_{CLK} = 8 MHz
Supply current, STOP mode	I_{DD3}		10	30	μA	
Threshold current	I_{TH}		0.5	1.0	mA	V_{TH} = 0 to V_{DD}
Input voltage, low	V_{IL}	0		0.8	V	
Input voltage, high	V_{IH1}	2.2		V_{DD}	V	All inputs except RESET, $P1_0$/NMI, X1, X2
	V_{IH2}	0.8 x V_{DD}		V_{DD}	V	RESET, $P1_0$/NMI, X1, X2
Output voltage, low	V_{OL}			0.45	V	I_{OL} = 1.6 mA
Output voltage, high	V_{OH}	V_{DD} − 1.0			V	I_{OH} = −0.4 mA
Input current	I_{IN}			±20	μA	$\overline{EA}$, $P1_0$/NMI; V_I = 0 to V_{DD}
Input leakage current	I_{LI}			±10	μA	All except $\overline{EA}$, $P1_0$/NMI; V_I = 0 to V_{DD}
Output leakage current	I_{LO}			±10	μA	V_O = 0 to V_{DD}
Data retention voltage	V_{DDDR}	2.5		5.5	V	

Notes:

(1) The standard operating temperature range is −10 to +70°C. However, extended temperature range parts (−40 to +85°C) are available.

AC Characteristics

V_{DD} = +5 V ±10%; T_A = −10 to +70 °C; C_L = 100 pF (max)

		Limits			
Parameter	**Symbol**	**Min**	**Max**	**Unit**	**Test Conditions**
V_{DD} rise, fall time	t_{RVD}, t_{FVD}	200		μs	STOP mode
Input rise, fall time	t_{IR}, t_{IF}		20	ns	Except X1, X2, $\overline{\text{RESET}}$, $\overline{\text{NMI}}$
Input rise, fall time	t_{IRS}, t_{IFS}		30	ns	$\overline{\text{RESET}}$, $\overline{\text{NMI}}$ (Schmitt)
Output rise, fall time	t_{OR}, t_{OF}		20	ns	Except CLKOUT
X1 cycle time	t_{CYX}	98	250	ns	Note 3
		62	250	ns	Note 4
X1 width, low	t_{WXL}	35		ns	Note 3
		20		ns	Note 4
X1 width, high	t_{WXH}	35		ns	Note 3
		20		ns	Note 4
X1 rise, fall time	t_{XR}, t_{XF}		20	ns	
CLKOUT cycle time	t_{CYK}	200	2000	ns	Note 3
		125	2000	ns	Note 4
CLKOUT width, low	t_{WKL}	0.5T − 15		ns	Note 1
CLKOUT width, high	t_{WKH}	0.5T − 15		ns	
CLKOUT rise, fall time	t_{KR}, t_{KF}		15	ns	
Address delay time	t_{DKA}		90	ns	
Address valid to input data valid	t_{DADR}		T(n + 1.5) − 90	ns	Note 2
$\overline{\text{MREQ}}$ to address hold time	t_{HMRA}	0.5T − 30		ns	
$\overline{\text{MREQ}}$ to data delay	t_{DMRD}		T(n + 2) − 75	ns	
$\overline{\text{MSTB}}$ to data delay	t_{DMSD}		T(n + 1) − 75	ns	
$\overline{\text{MREQ}}$ to $\overline{\text{MSTB}}$ delay	t_{DMRMSR}	T − 35	T + 35	ns	
$\overline{\text{MREQ}}$ width, low	t_{WMRL}	T(n + 2) − 30		ns	
$\overline{\text{MREQ}}$, $\overline{\text{MSTB}}$ to address hold time	t_{HMA}	0.5T − 50		ns	
Input data hold time	t_{HMD}	0		ns	
Next control setup time	t_{SCC}	T − 25		ns	
$\overline{\text{MREQ}}$ to TC delay time	t_{DMRTC}		0.5T + 50	ns	
$\overline{\text{MREQ}}$ delay time	t_{DAMR}	0.5T − 30		ns	
$\overline{\text{MSTB}}$ delay time	t_{DAMSR}	T − 30		ns	
$\overline{\text{MSTB}}$ width, low	t_{WMSLR}	T(n + 1) − 30		ns	
Address data output	t_{DADW}		0.5T + 50	ns	
Data output setup time	t_{SDM}	T(n + 2) − 50		ns	
$\overline{\text{MSTB}}$ write delay time	t_{DAMSW}	T(n + 0.5) − 30		ns	
$\overline{\text{MREQ}}$ to $\overline{\text{MSTB}}$ write delay time	t_{DMRMSW}	T(n + 1) − 35		ns	
$\overline{\text{MSTB}}$ write width low	t_{WMSLW}	T − 30		ns	
Data output hold time	t_{HMDW}	0.5T − 50		ns	
$\overline{\text{IOSTB}}$ delay time	t_{DAIS}	0.5T − 30		ns	
$\overline{\text{IOSTB}}$ to data input	t_{DISD}		T(n + 1) − 90	ns	
$\overline{\text{IOSTB}}$ width, low	t_{WISL}	T(n + 1) − 30		ns	
Address hold time	t_{HISA}	0.5T − 30		ns	
Input data hold time	t_{HISDR}	0		ns	
Output data setup time	t_{SDIS}	T(n + 1) − 50		ns	
Output data hold time	t_{HISDW}	0.5T − 30		ns	
Next DMARQ setup time	t_{SDADQ}		T	ns	Demand mode
DMARQ hold time	t_{HDARQ}	0		ns	Demand mode
$\overline{\text{DMAAK}}$ read width, low	t_{WDMRL}	T(n + 2.5) − 30		ns	
$\overline{\text{DMAAK}}$ write width, low	t_{WDMWL}	T(n + 2) − 30		ns	
$\overline{\text{DMAAK}}$ to TC delay time	t_{DDATC}		0.5T + 50	ns	
$\overline{\text{TC}}$ width, low	t_{WTCL}	2T − 30		ns	
$\overline{\text{REFRQ}}$ delay time	t_{DARF}	0.5T − 30		ns	
$\overline{\text{REFRQ}}$ width, low	t_{WRFL}	T(n + 2) − 30		ns	
Address hold time	t_{HRFA}	0.5T − 30		ns	

AC Characteristics (cont)

Parameter	Symbol	Limits Min	Limits Max	Unit	Test Conditions
$\overline{\text{RESET}}$ width low	t_{WRSL1}	30		ms	STOP/POR (Power-on reset)
	t_{WRSL2}	5		μs	System reset
$\overline{\text{MREQ}}$, $\overline{\text{IOSTB}}$ to READY setup time	t_{SCRY}		T(n − 1) − 100	ns	n ≥ 2
$\overline{\text{MREQ}}$, $\overline{\text{IOSTB}}$ to READY hold time	t_{HCRY}	T(n)		ns	n ≥ 2
Ready setup time	t_{SRYK}	20		ns	
Ready hold time	t_{HKRY}	40		ns	
HLDRQ setup time	t_{SHQK}	30		ns	
$\overline{\text{HLDAK}}$ output delay	t_{DKHA}		80	ns	
Bus control float to $\overline{\text{HLDAK}}$↓	t_{CFHA}	T − 50		ns	
$\overline{\text{HLDAK}}$↑ to control output time	t_{DHAC}	T − 50		ns	
HLDRQ to $\overline{\text{HLDAK}}$ delay	t_{DHQHA}		3T + 160	ns	
HLDRQ↓ to control float	t_{DHQC}	3T + 30		ns	
HLDRQ width, low	t_{WHQL}	1.5T		ns	
$\overline{\text{HLDAK}}$ width, low	t_{WHAL}		T	ns	
$\overline{\text{INTP}}$, DMARQ setup	t_{SIQK}	30		ns	
$\overline{\text{INTP}}$, DMARQ width, high	t_{WIQH}	8T		ns	
$\overline{\text{INTP}}$, DMARQ width, low	t_{WIQL}	8T		ns	
$\overline{\text{POLL}}$ setup time	t_{SPLK}	30		ns	
NMI width, high	t_{WNIH}	5		μs	
NMI width, low	t_{WNIL}	5		μs	
$\overline{\text{CTS}}$ width, low	t_{WCTL}	2T		ns	
INTR setup time	t_{SIRK}	30		ns	
$\overline{\text{INTAK}}$ delay time	t_{DKIA}		80	ns	
INTR hold time	t_{HIAIQ}	0		ns	
$\overline{\text{INTAK}}$ width, low	t_{WIAL}	2T − 30		ns	
$\overline{\text{INTAK}}$ width, high	t_{WIAH}	T − 30		ns	
$\overline{\text{INTAK}}$ to data delay	t_{DIAD}		2T − 130	ns	
$\overline{\text{INTAK}}$ to data hold	t_{HIAD}	0	0.5T	ns	
$\overline{\text{SCK0}}$ (TSCK) cycle time	t_{CYTK}	1000		ns	
$\overline{\text{SCK0}}$ (TSCK) width, high	t_{WSTH}	450		ns	
$\overline{\text{SCK0}}$ (TSCK) width, low	t_{WSTL}	450		ns	
TxD delay time	t_{DTKD}		210	ns	

Parameter	Symbol	Limits Min	Limits Max	Unit	Test Conditions
$\overline{\text{CTS0}}$ (RSCK) cycle time	t_{CYRK}	1000		ns	
$\overline{\text{CTS0}}$ (RSCK) width, high	t_{WSRH}	420		ns	
$\overline{\text{CTS0}}$ (RSCK) width, low	t_{WSRL}	420		ns	
RxD setup time	t_{SRDK}	80		ns	
RxD hold time	t_{HKRD}	80		ns	

Notes:

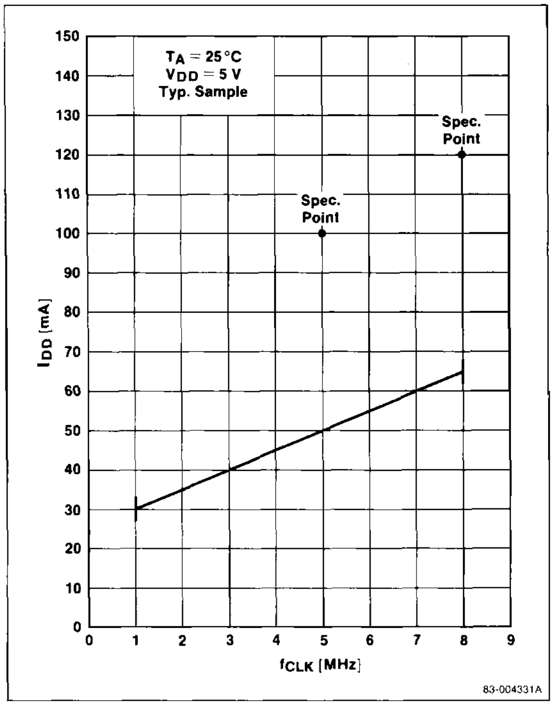
(1) T = CPU clock period (t_{CYK}).

(2) n = number of wait states inserted.

(3) For 5 MHz parts (μPD70320/322).

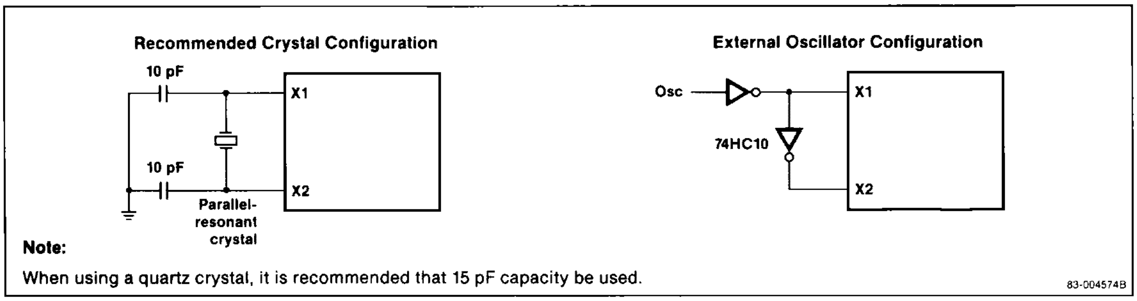
(4) For 10 MHz parts (μPD70320/322-8).

Supply Current vs Clock Frequency



4b

Figure 36. External System Clock Control Source


Recommended Crystal Configuration
10 pF
10 pF
X1
X2
Parallel-resonant crystal
External Oscillator Configuration
Osc
74HC10
X1
X2
83-004574B

Note:

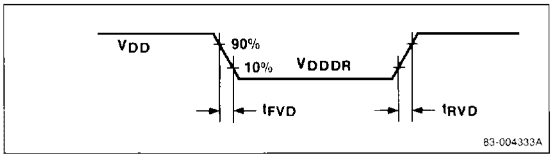
When using a quartz crystal, it is recommended that 15 pF capacity be used.

Resonator and Capacitance Requirements

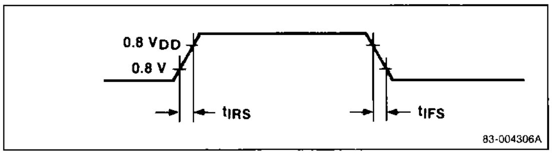
Manufacturer	Product Number	Recommended C1 (pF)	Constants C2 (pF)	Product Number	Recommended C1 (pF)	Constants C2 (pF)
Kyocera	KBR-10.0M	33	33			
Murata Manufacturing	CSA.10.0MT	47	47	CSA16.0MX040	30	30
TDK	FCR10.0M2S	30	30	FCR16.0M2S	15	6

Timing Waveforms

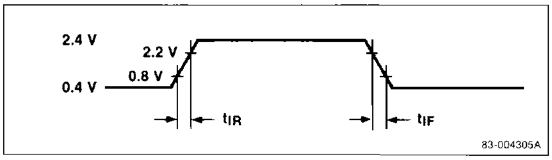
Stop Mode Data Retention Timing



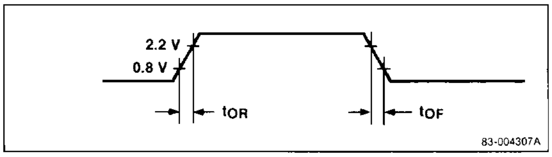
AC Input Waveform 2 (RESET, NMI)



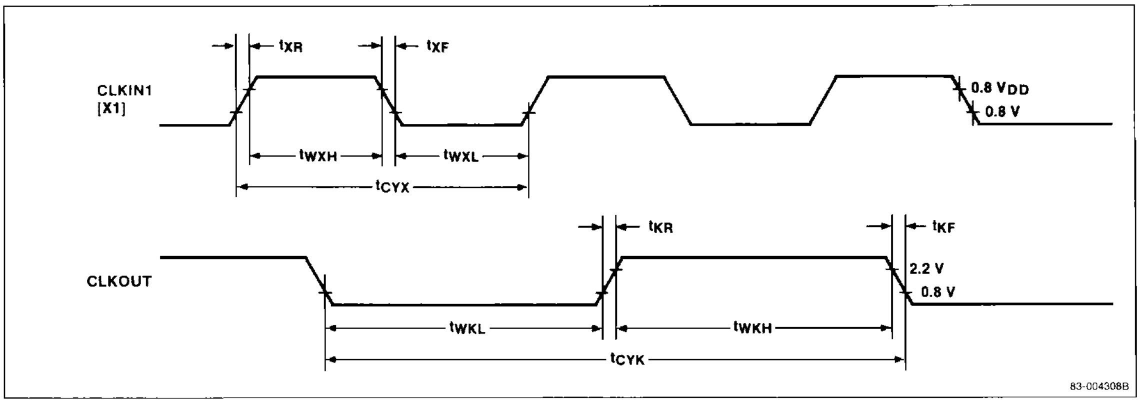
AC Input Waveform 1 (Except X1, X2, RESET, NMI)



AC Output Test Point (Except CLKOUT)

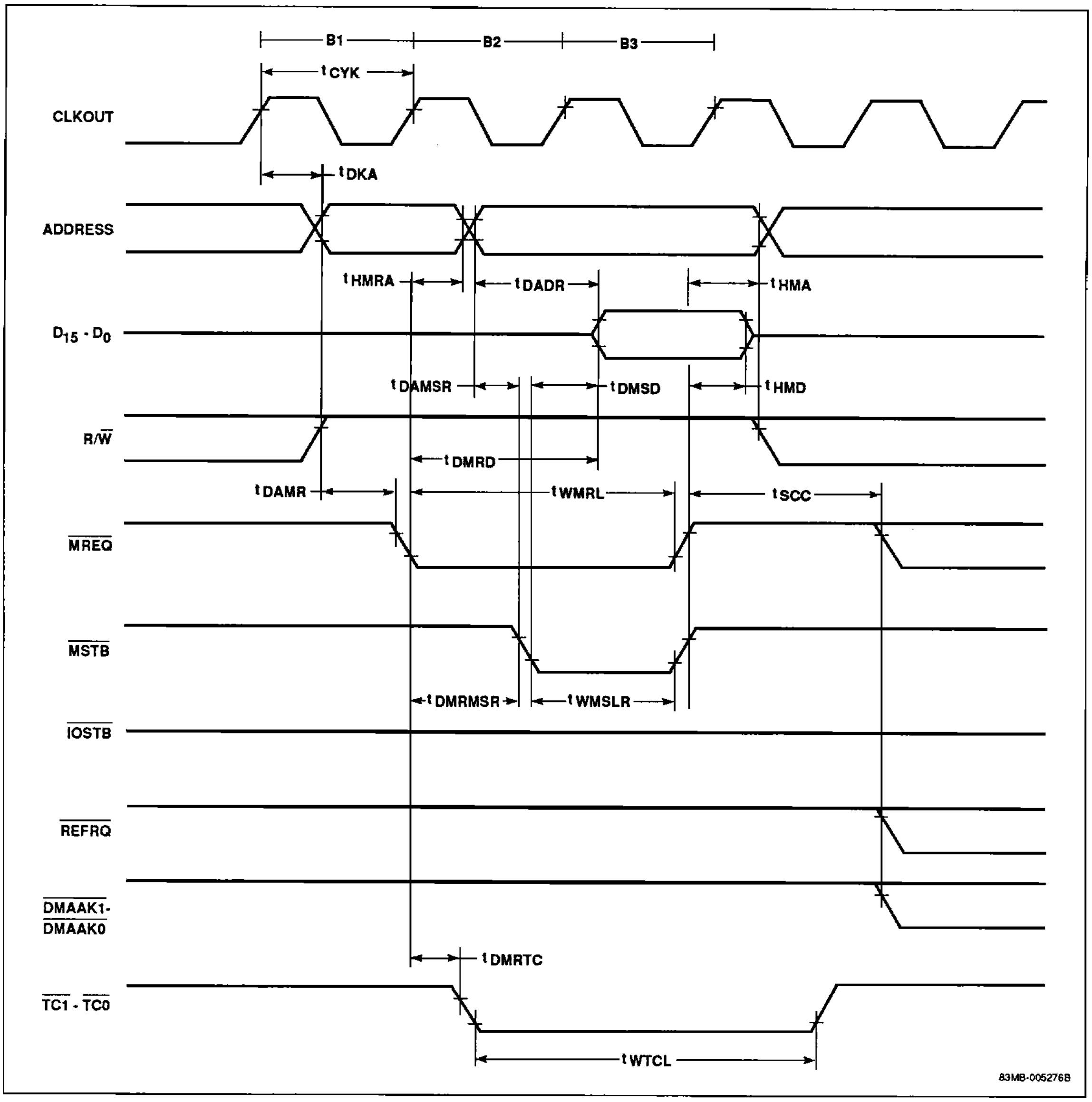


Clock In and Clock Out



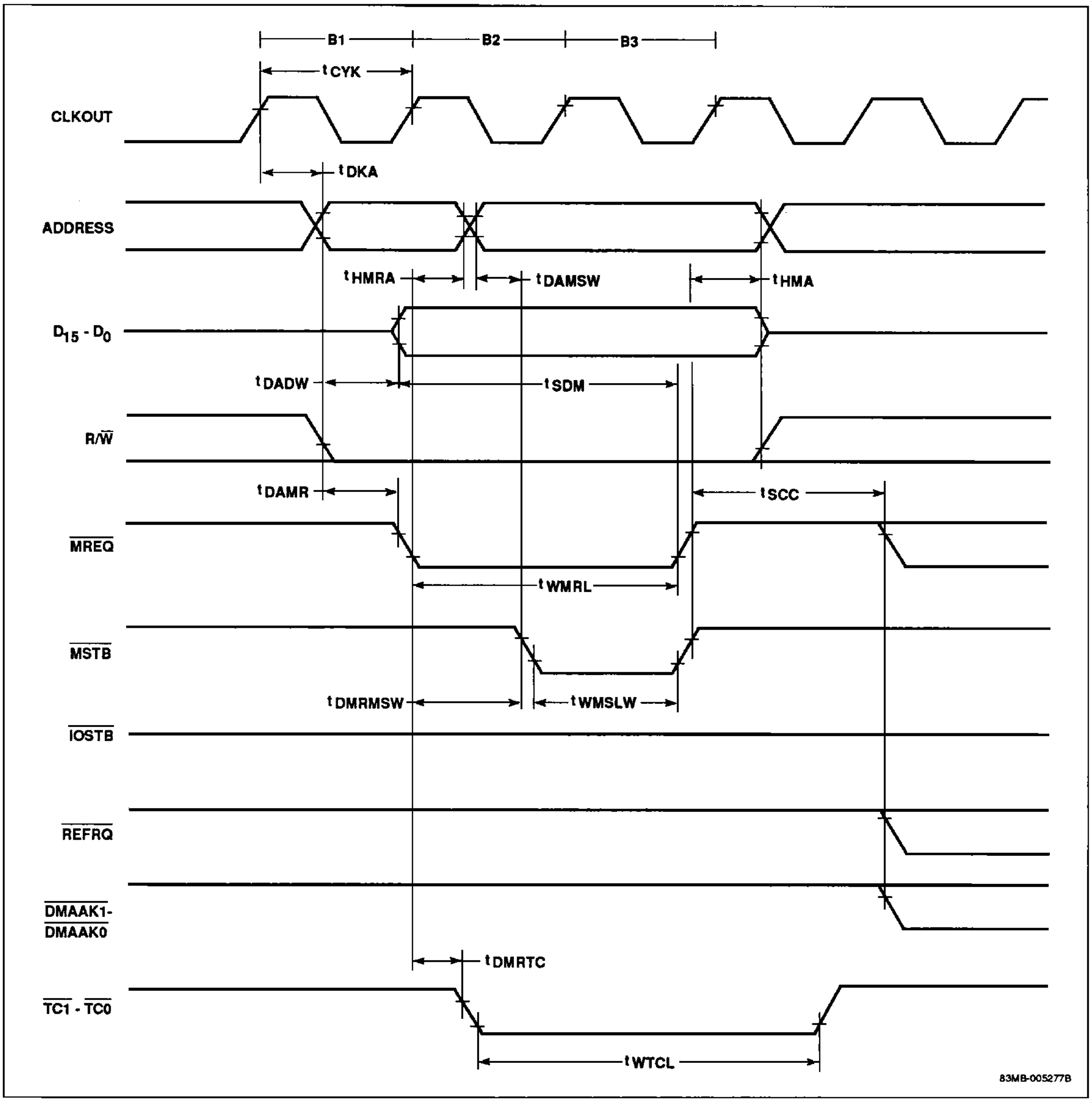
Timing Waveforms (cont)

Memory Read



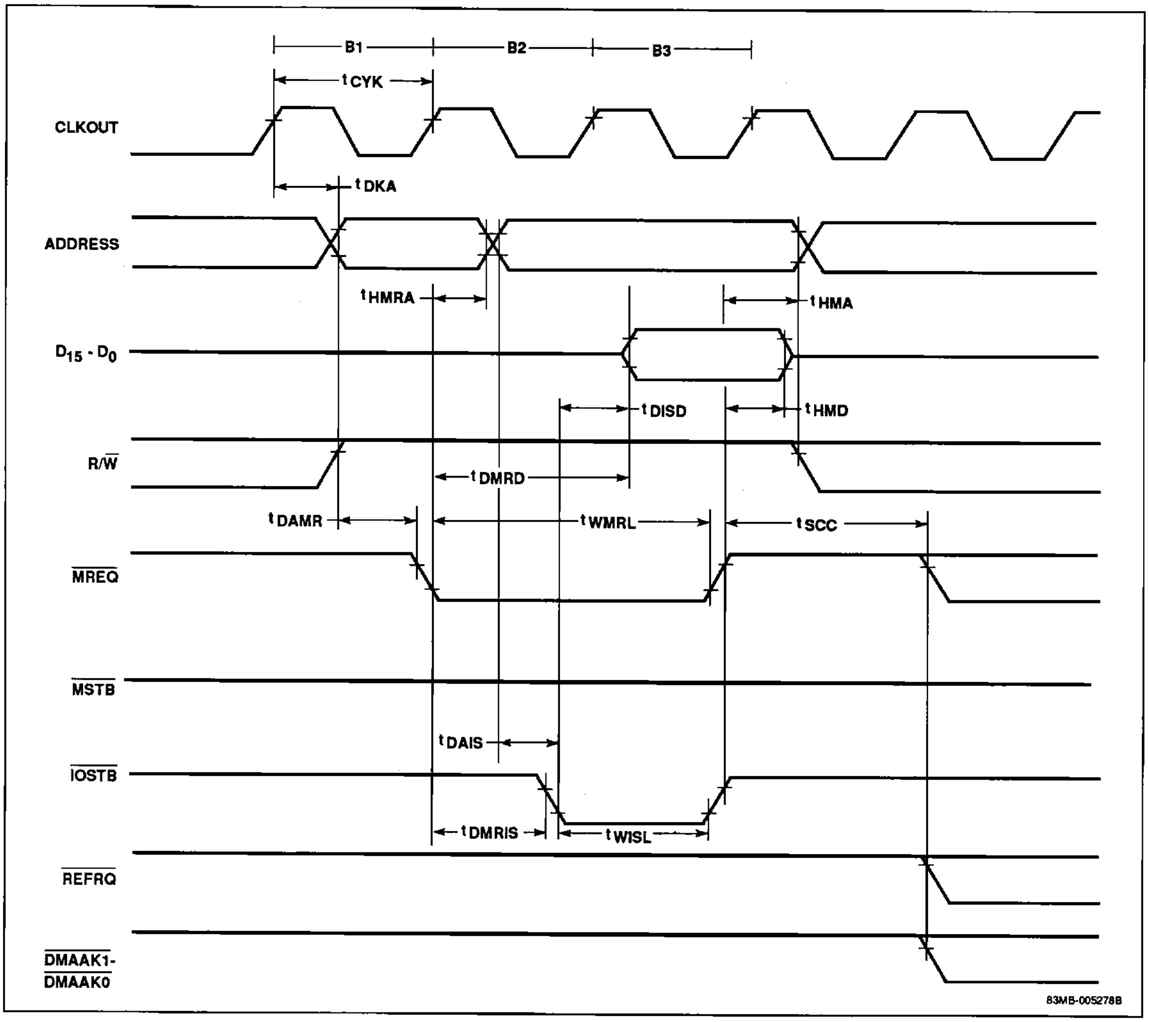
Timing Waveforms (cont)

Memory Write



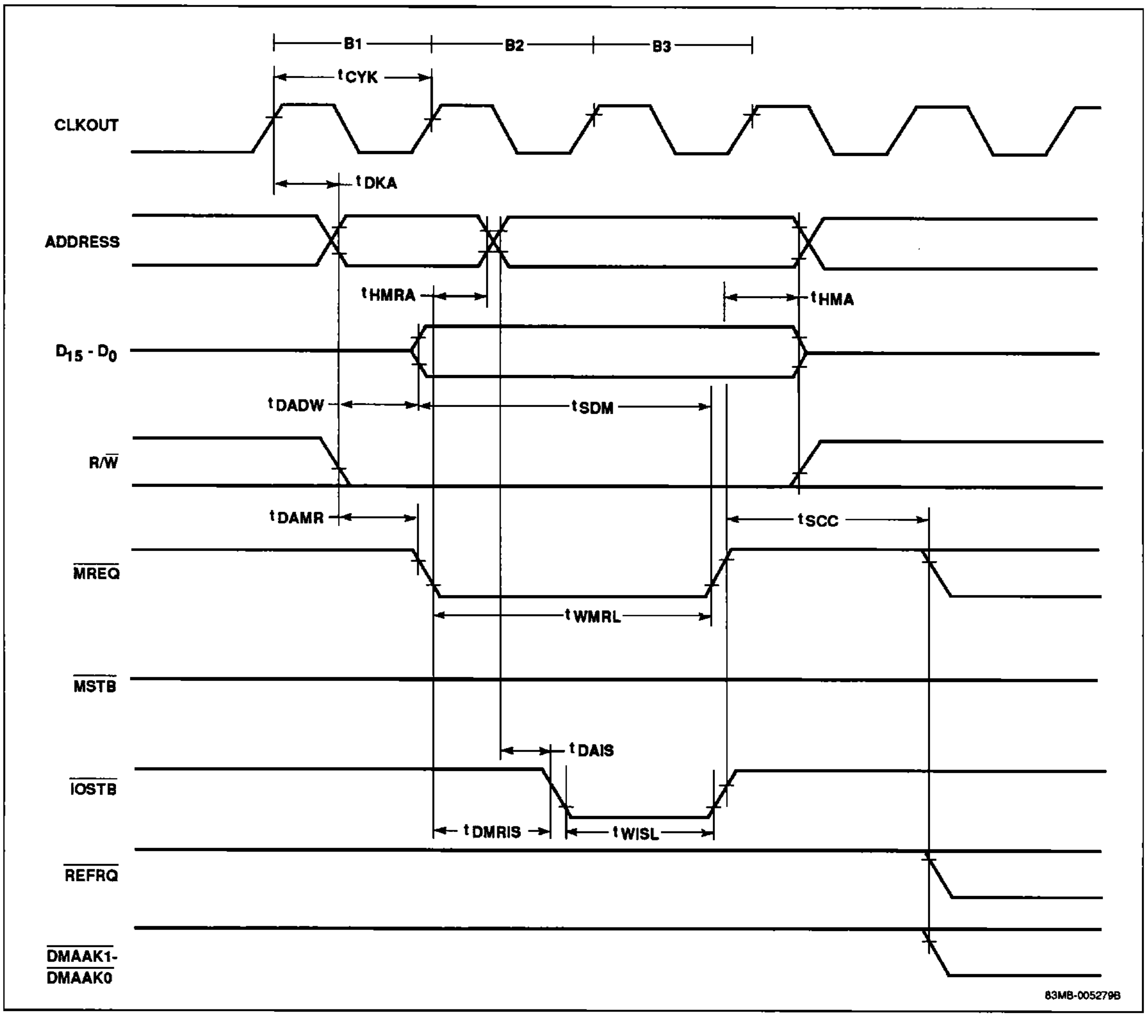
Timing Waveforms (cont)

I/O Read



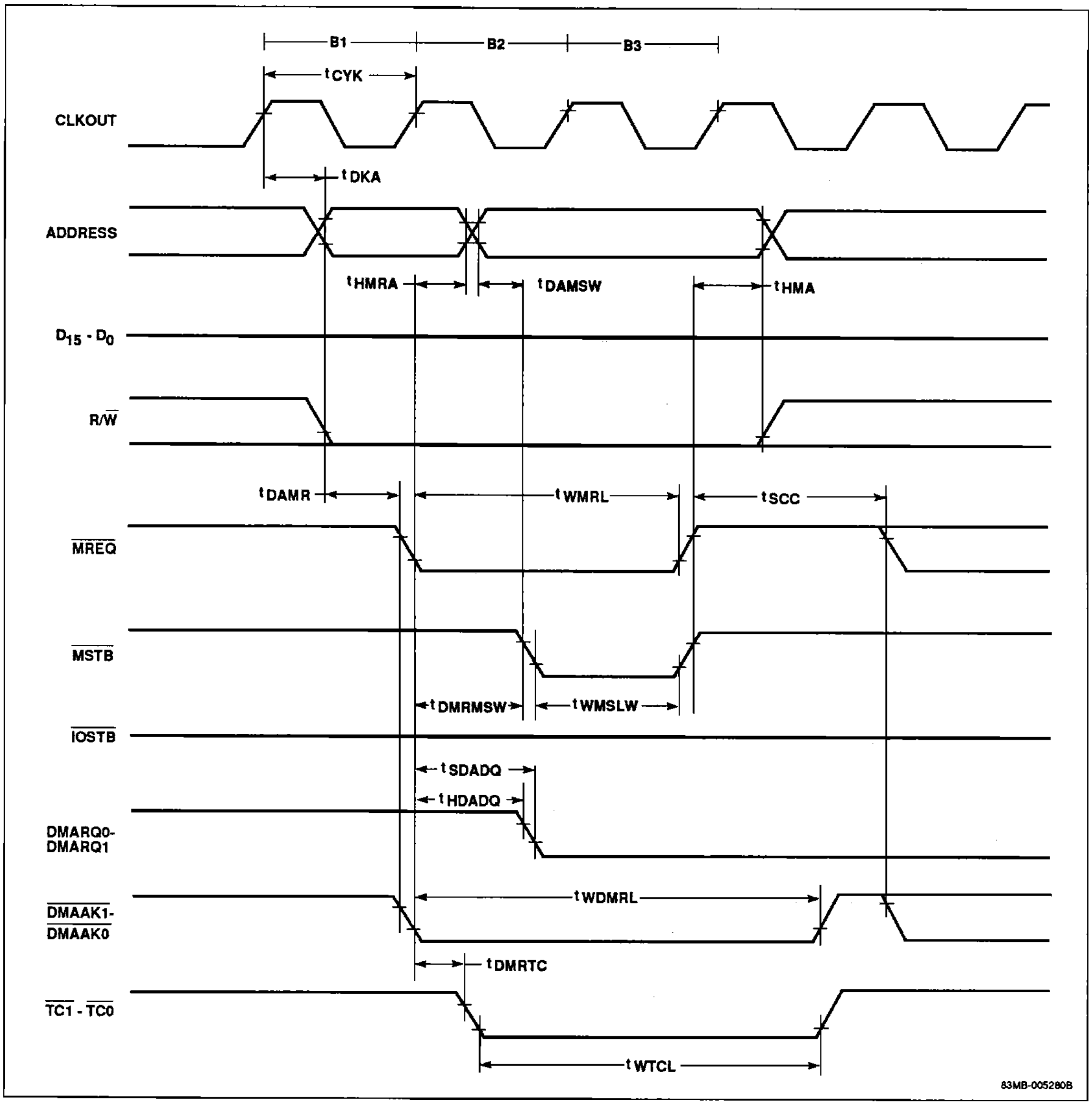
Timing Waveforms (cont)

I/O Write



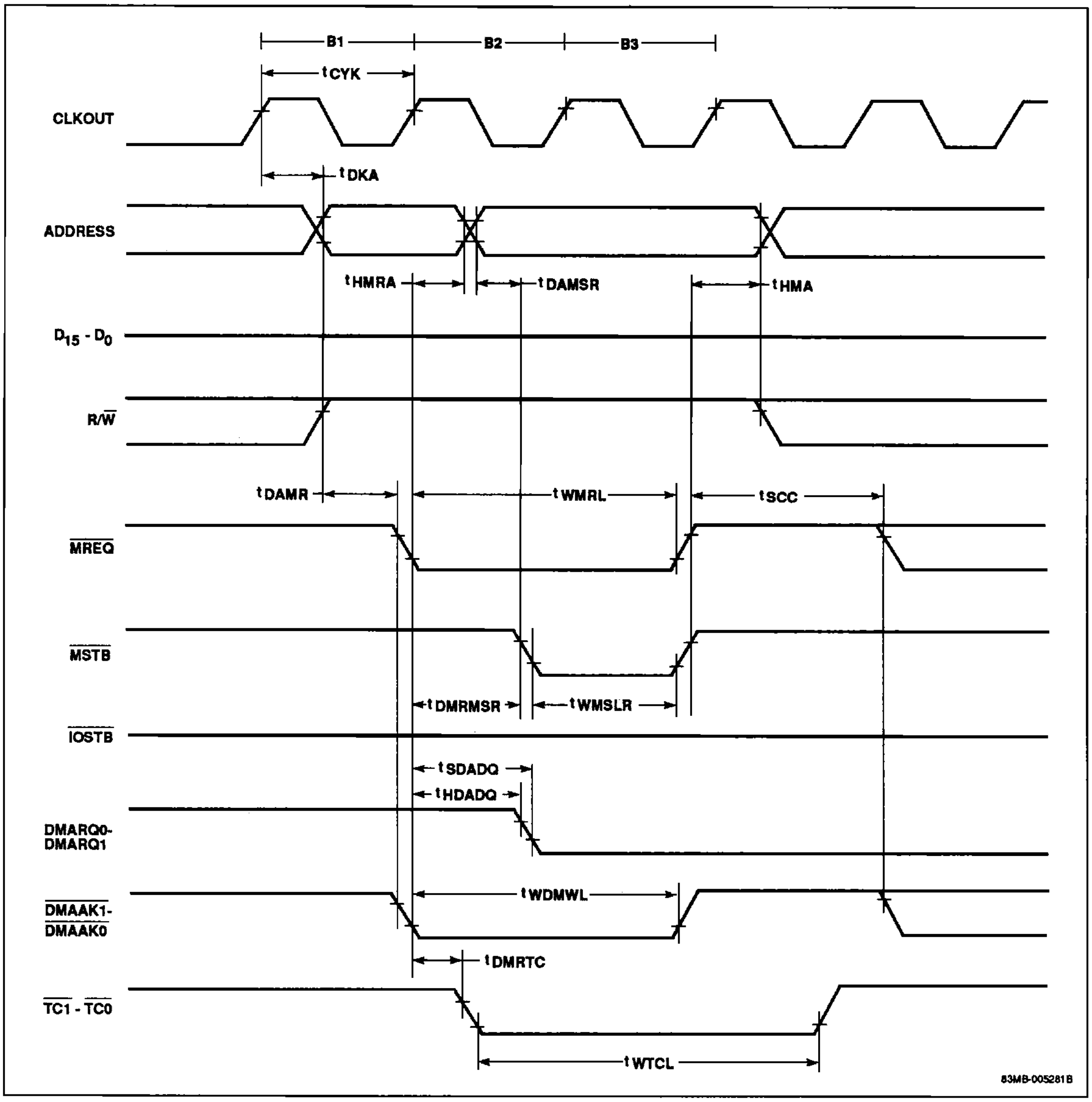
Timing Waveforms (cont)

DMA, I/O to Memory



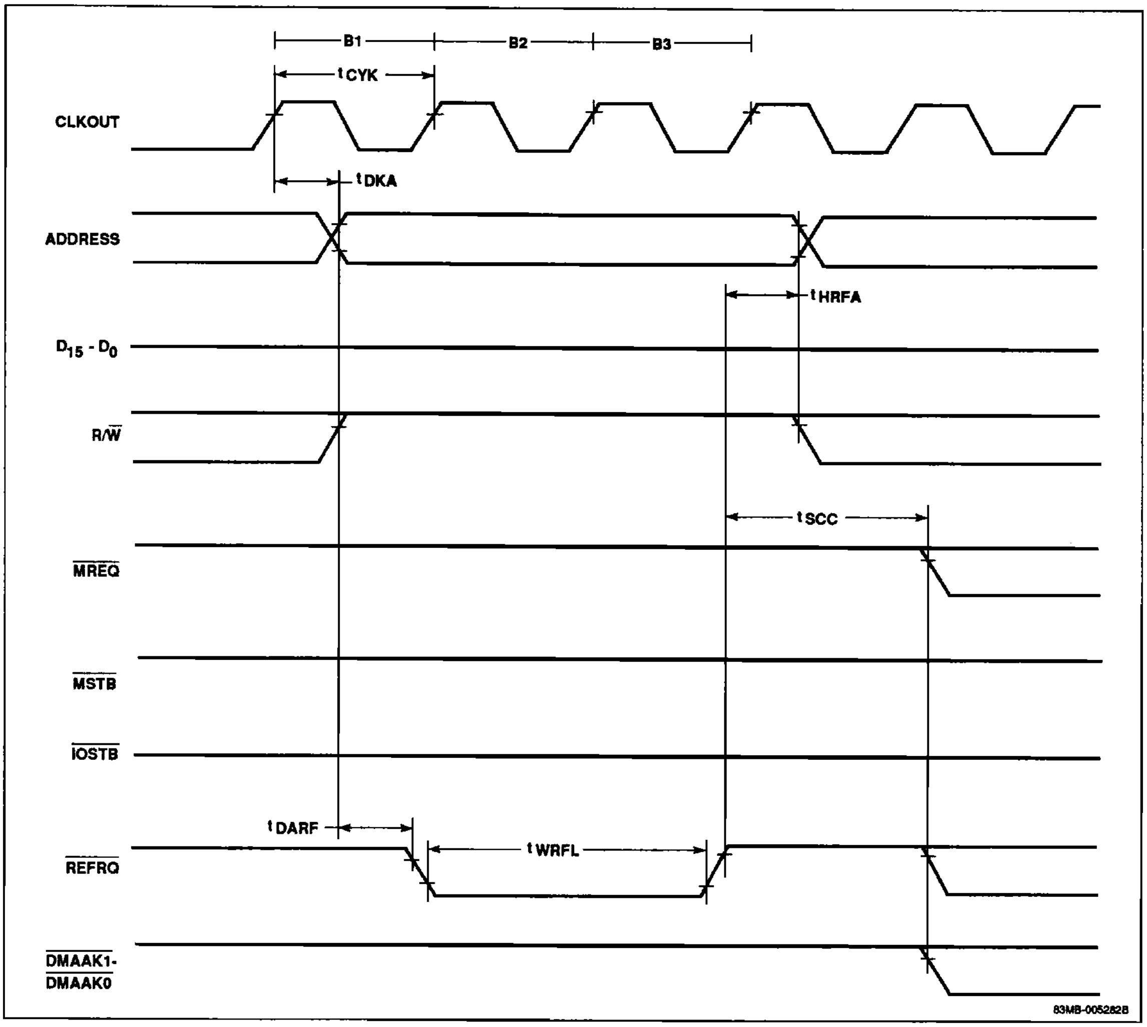
Timing Waveforms (cont)

DMA, Memory to I/O



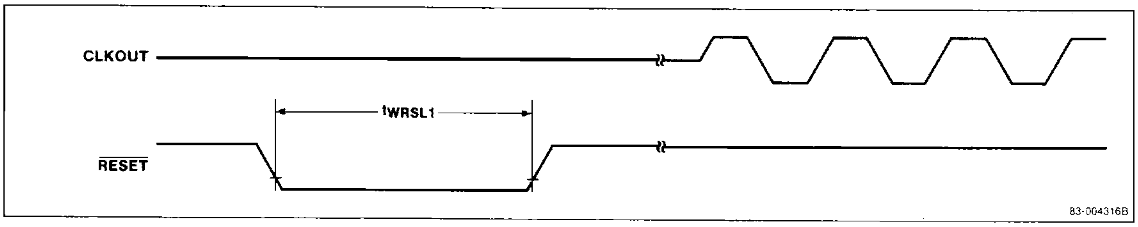
Timing Waveforms (cont)

Refresh

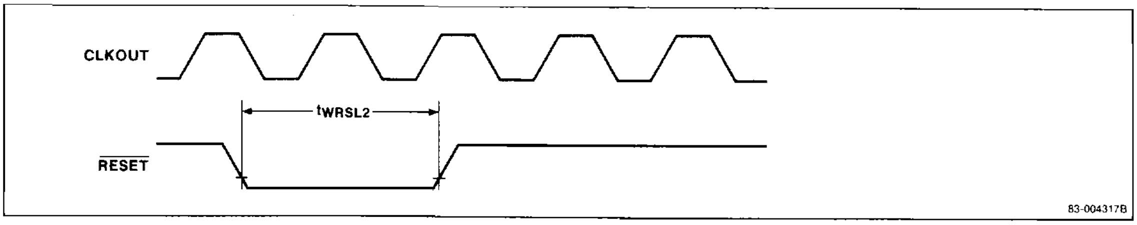


Timing Waveforms (cont)

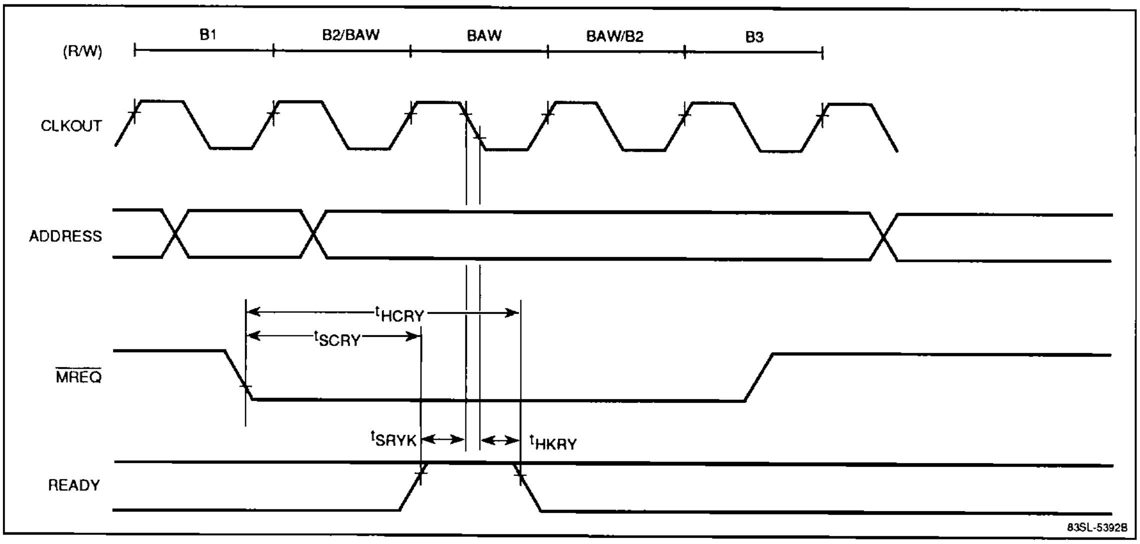
$\overline{RESET}$ 1



$\overline{RESET}$ 2

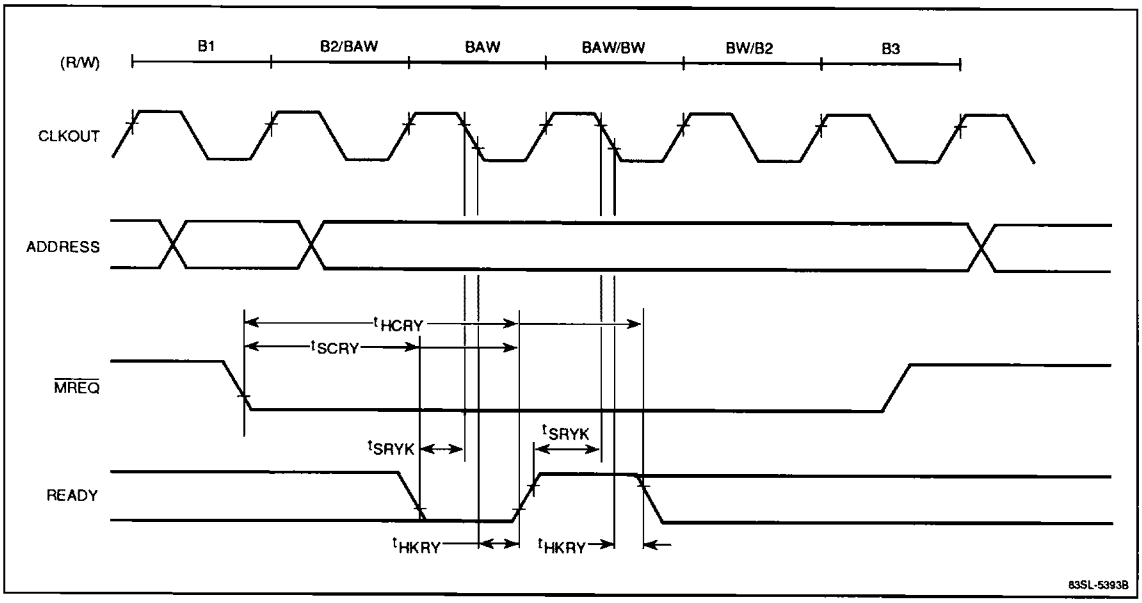


READY Timing 1

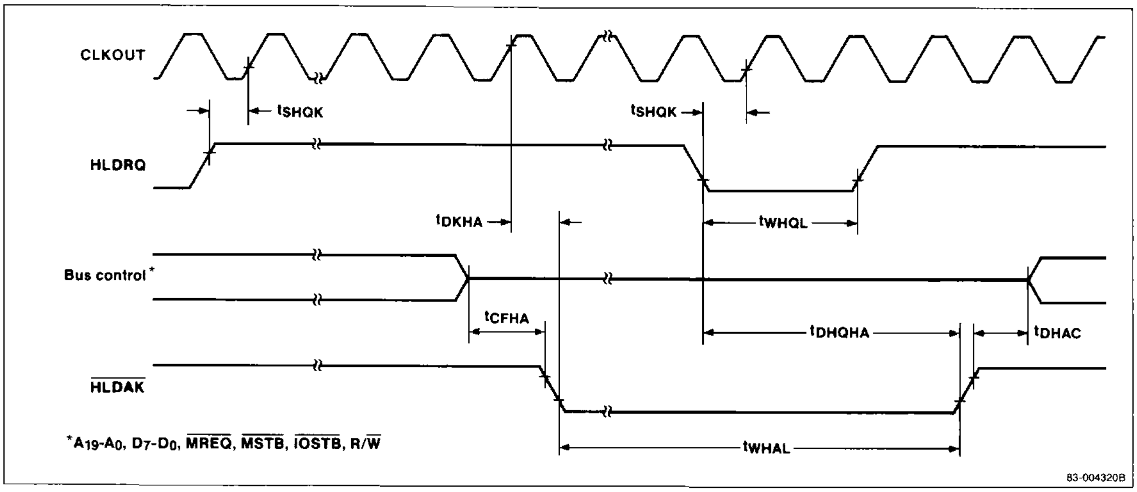


Timing Waveforms (cont)

READY Timing 2

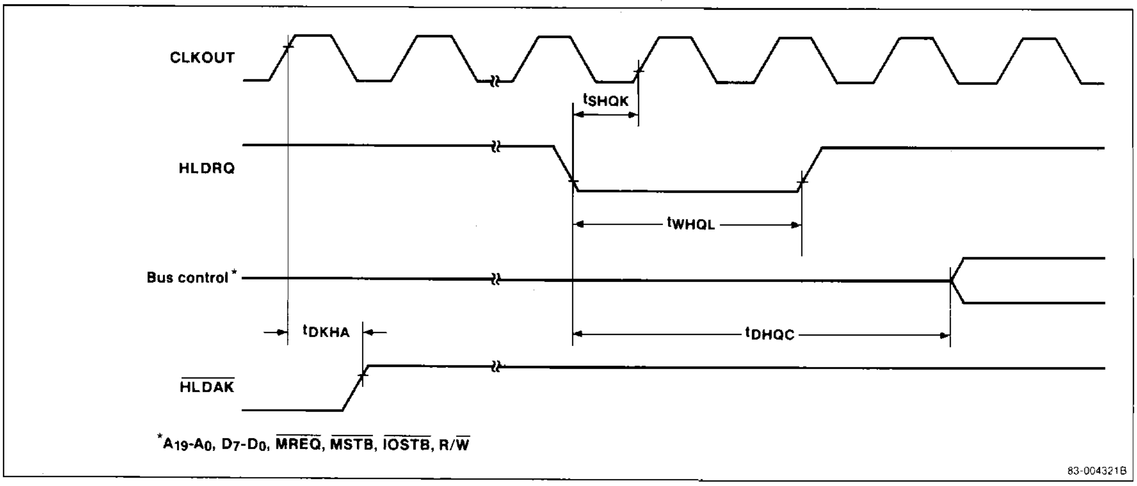


HLDRQ/$\overline{\text{HLDAK}}$ 1

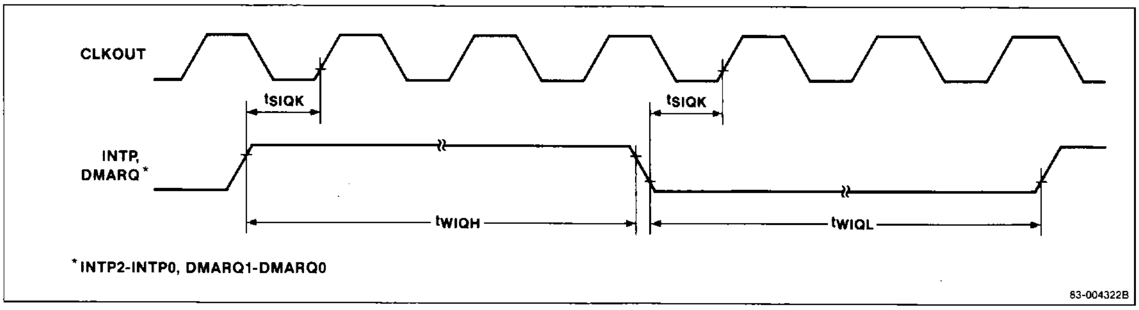


Timing Waveforms (cont)

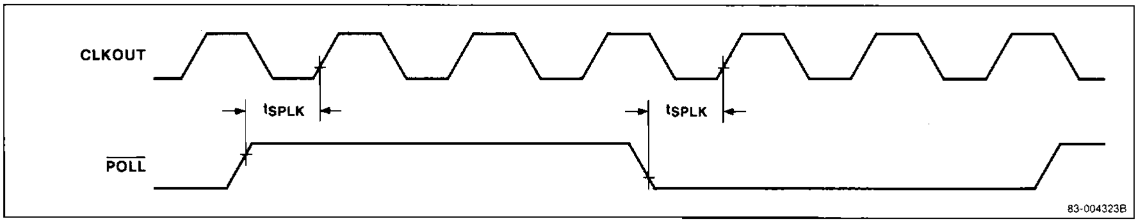
HLDRQ/$\overline{HLDAK}$ 2



INTP, DMARQ Input

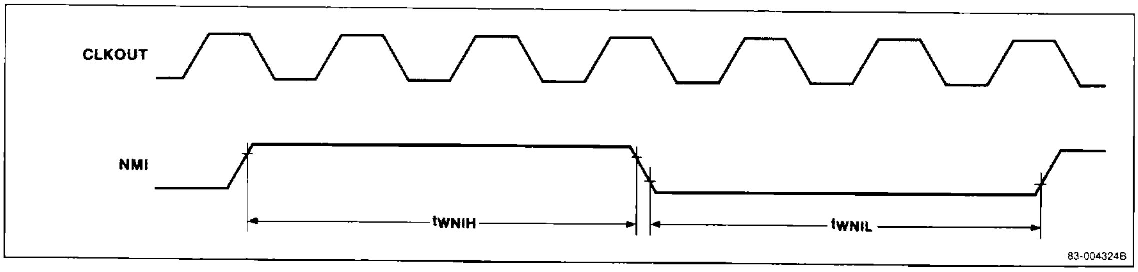


$\overline{POLL}$ Input

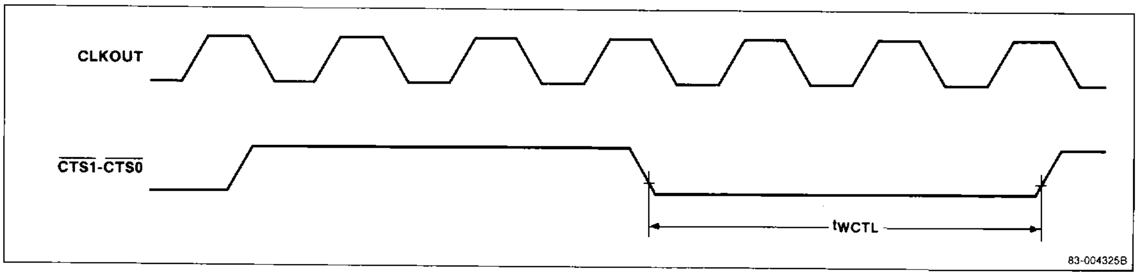


Timing Waveforms (cont)

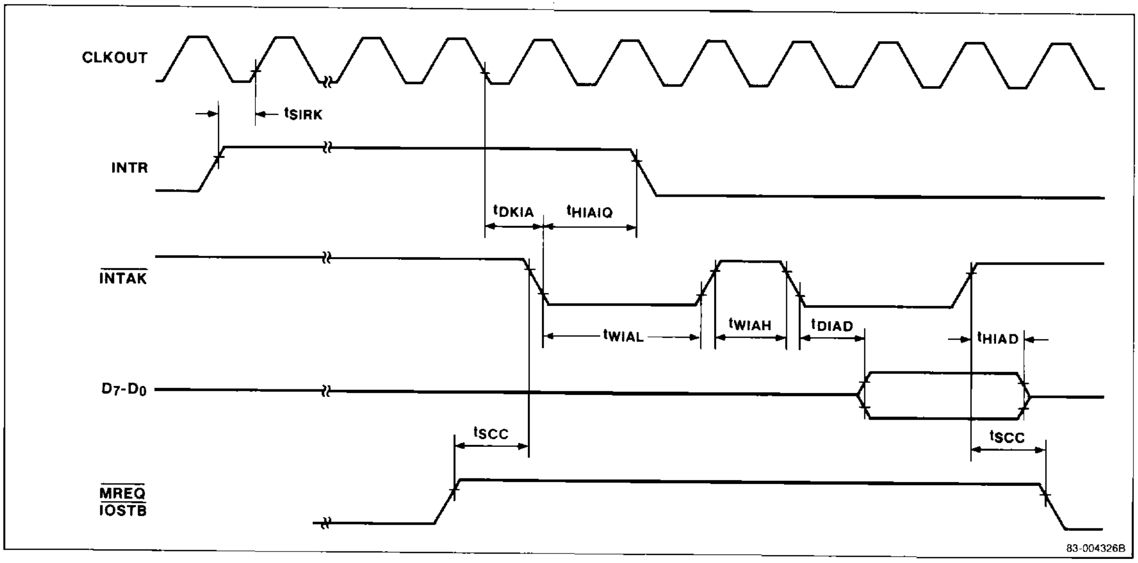
NMI Input



$\overline{\text{CTS}}$ Input

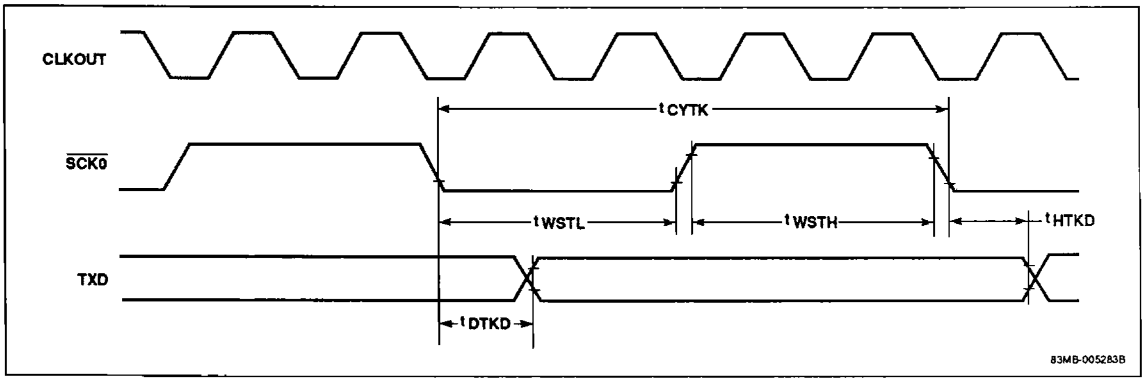


INTR/$\overline{\text{INTAK}}$

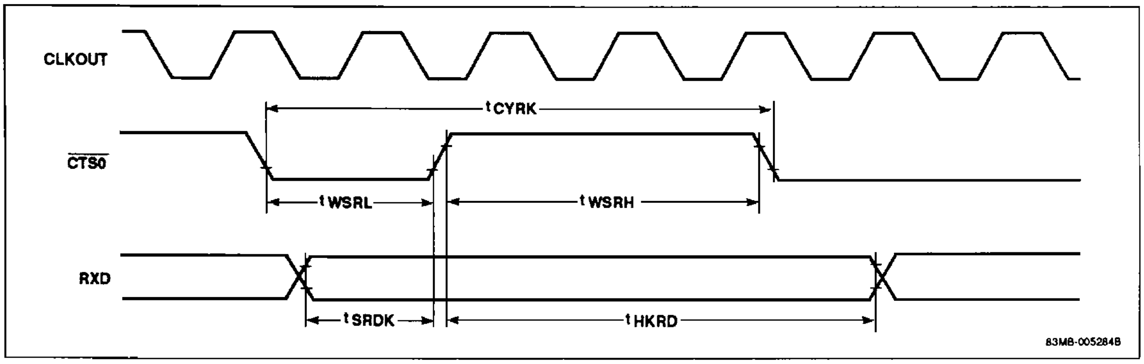


Timing Waveforms (cont)

Serial Transmit



Serial Receive



Instruction Set

Instructions, grouped according to function, are described in a table near the end of this data sheet. Descriptions include source code, operation, opcode, number of bytes, and flag status. Supplementary information applicable to the instruction set is contained in the following tables.

- Symbols and Abbreviations
- Flag Symbols
- 8- and 16-Bit Registers. When mod = 11, the register is specified in the operation code by the byte/word operand (W = 0/1) and reg (000 to 111).
- Segment Registers. The segment register is specified in the operation code by sreg (00, 01, 10, or 11).
- Memory Addressing. The memory addressing mode is specified in the operation code by mod (00, 01, or 10) and mem (000 through 111).
- Instruction Clock Count. This table gives formulas for calculating the number of clock cycles occupied by each type of instruction. The formulas, which depend on byte/word operand and RAM enable/disable, have variables such as EA (effective address), W (wait states), and n (iterations or string instructions).

Symbols and Abbreviations

Identifier	Description
reg	8- or 16-bit general-purpose register
reg8	8-bit general-purpose register
reg16	16-bit general-purpose register
dmem	8- or 16-bit direct memory location
mem	8- or 16-bit memory location
mem8	8-bit memory location
mem16	16-bit memory location
mem32	32-bit memory location
sfr	8-bit special function register location
imm	Constant (0 to FFFFH)
imm16	Constant (0 to FFFFH)
imm8	Constant (0 to FFH)
imm4	Constant (0 to FH)
imm3	Constant (0 to 7)
acc	AW or AL register
sreg	Segment register
src-table	Name of 256-byte translation table
src-block	Name of block addressed by the IX register
dst-block	Name of block addressed by the IY register
near-proc	Procedure within the current program segment
far-proc	Procedure located in another program segment
near-label	Label in the current program segment
short-label	Label between −128 and +127 bytes from the end of instruction
far-label	Label in another program segment
memptr16	Word containing the offset of the memory location within the current program segment to which control is to be transferred
memptr32	Double word containing the offset and segment base address of the memory location to which control is to be transferred
regptr16	16-bit register containing the offset of the memory location within the program segment to which control is to be transferred
pop-value	Number of bytes of the stack to be discarded (0 to 64K bytes, usually even addresses)
fp-op	Immediate data to identify the instruction code of the external floating point operation
R	Register set
W	Word/byte field (0 to 1)
reg	Register field (000 to 111)
mem	Memory field (000 to 111)
mod	Mode field (00 to 10)
S:W	When S:W = 01 or 11, data = 16 bits. At all other times, data = 8 bits.
X, XXX, YYY, ZZZ	Data to identify the instruction code of the external floating point arithmetic chip
AW	Accumulator (16 bits)
AH	Accumulator (high byte)
AL	Accumulator (low byte)
BP	Base pointer register (16 bits)
BW	BW register (16 bits)
BH	BW register (high byte)
BL	BW register (low byte)
CW	CW register (16 bits)
CH	CW register (high byte)
CL	CW register (low byte)
DW	DW register (16 bits)
DH	DW register (high byte)
DL	DW register (low byte)
SP	Stack pointer (16 bits)
PC	Program counter (16 bits)
PSW	Program status word (16 bits)

Symbols and Abbreviations (cont)

Identifier	Description
IX	Index register (source) (16 bits)
IY	Index register (destination) (16 bits)
PS	Program segment register (16 bits)
SS	Stack segment register (16 bits)
DS_0	Data segment 0 register (16 bits)
DS_1	Data segment 1 register (16 bits)
AC	Auxiliary carry flag
CY	Carry flag
P	Parity flag
S	Sign flag
Z	Zero flag
DIR	Direction flag
IE	Interrupt enable flag
V	Overflow flag
BRK	Break flag
MD	Mode flag
(...)	Values in parentheses are memory contents
disp	Displacement (8 or 16 bits)
ext-disp8	16-bit displacement (sign-extension byte + 8-bit displacement)
temp	Temporary register (8/16/32 bits)
tmpcy	Temporary carry flag (1-bit)
seg	Immediate segment data (16 bits)
offset	Immediate offset data (16 bits)
←	Transfer direction
+	Addition
−	Subtraction
x	Multiplication
÷	Division
%	Modulo
AND	Logical product
OR	Logical sum
XOR	Exclusive logical sum
XXH	Two-digit hexadecimal value
XXXXH	Four-digit hexadecimal value

Flag Symbols

Identifier	Description
(blank)	No change
0	Cleared to 0
1	Set to 1
X	Set or cleared according to the result
U	Undefined
R	Value saved earlier is restored

8- and 16-Bit Registers (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 Registers

sreg	Register
00	DS_1
01	PS
10	SS
11	DS_0

Memory Addressing

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

Instruction Clock Count

Mnemonic	Operand	Clocks
ADD	reg8, reg8	2
	reg16, reg16	2
	reg8, mem8	EA+7+W
	reg16, mem16	EA+7+W
	mem8, reg8	EA+10+2W [EA+7+W]
	mem16, reg16	EA+10+2W [EA+7+W]
	reg8, imm8	5
	reg16, imm8	5
	reg16, imm16	6
	mem8, imm8	EA+11+2W [EA+9+2W]
	mem16, imm8	EA+9+2W [EA+7+2W]
	mem16, imm16	EA+12+2W [EA+8+2W]
	AL, imm8	5
	AW, imm16	6
ADD4S		22+(30+3W)n [22+(28+3W)n]
ADDC	Same as ADD	
ADJ4A		9
ADJ4S		9
ADJBA		17
ADJBS		17
AND	reg8, reg8	2
	reg16, reg16	2
	reg8, mem8	EA+7+W
	reg16, mem16	EA+7+W
	mem8, reg8	EA+10+2W [EA+7+W]
	mem16, reg16	EA+10+2W [EA+7+W]
	reg8, imm8	5
	reg16, imm16	6
	mem8, imm8	EA+11+2W [EA+9+2W]
	mem16, imm16	EA+12+2W [EA+8+2W]
Bcond (conditional branch)		8 or 15
BCWZ		8 or 15
BR	near-label	12
	short-label	12
	regptr16	13
	memptr16	EA+16+W
	far-label	15
	memptr32	EA+23+2W

Notes:

(1) If the number of clocks is not the same for RAM enabled and RAM disabled conditions, the RAM enabled value is listed first, followed by the RAM disabled value in brackets; for example, EA+8+2W [EA+6+W].

(2) Symbols in the Clocks column are defined as follows.

EA = additional clock cycles required for calculation of the effective address

= 3 (mod 00 or 01) or 4 (mod 10)

W = number of wait states selected by the WTC register

n = number of iterations or string instructions

Mnemonic	Operand	Clocks
BRK	3	50+5W [38+5W]
	imm8	51+5W [39+5W]
BRKCS		15
BRKV		50+5W [38+5W]
BTCLR		29
BUSLOCK		2
CALL	near-proc	21+W [17+W]
	regptr16	21+W [17+W]
	memptr16	EA+24+2W [EA+22+2W]
	far-proc	36+2W [32+2W]
	memptr32	EA+32+4W [EA+20+4W]
CHKIND		EA+24+2W
CLR1	CY	2
	DIR	2
	reg8, CL	8
	reg16, CL	8
	mem8, CL	EA+16+2W [EA+13+W]
	mem16, CL	EA+16+2W [EA+13+W]
	reg8, imm3	7
	reg16, imm4	7
	mem8, imm3	EA+13+2W [EA+10+W]
	mem16, imm4	EA+13+2W [EA+9+W]
CMP	reg8, reg8	2
	reg16, reg16	2
	reg8, mem8	EA+7+W
	reg16, mem16	EA+7+W
	mem8, reg8	EA+7+W
	mem16, reg16	EA+7+W
	reg8, imm8	5
	reg16, imm8	5
	reg16, imm16	6
	mem8, imm8	EA+8+W
	mem16, imm8	EA+9+W
	mem16, imm16	EA+9+W
	AL, imm8	5
	AW, imm16	6
CMP4S		22+(25+2W)n
CMPBK	mem8, mem8	25+2W [21+2W]
	mem16, mem16	25+2W [19+2W]

4b

Instruction Clock Count (cont)

Mnemonic	Operand	Clocks
CMPBKB		16+(23+2W)n
CMPBKW		16+(23+2W)n
CMPM	mem8	18+W
	mem16	19+2W
CMPMB		16+(16+W)n
CMPMW		16+(16+2W)n
CVTBD		19
CVTBW		3
CVTDB		20
CVTWL		8
DBNZ		8 or 17
DBNZE		8 or 17
DBNZNE		8 or 17
DEC	reg8	5
	reg16	2
	mem8	EA+13+2W [EA+11+2W]
	mem16	EA+13+2W [EA+9+2W]
DI		4
DISPOSE		11+W
DIV	AW, reg8	46-56
	AW, mem8	EA+49+W to EA+59+W
	DW:AW, reg16	54-64
	DW:AW, mem16	EA+57+W to EA+67+W
DIVU	AW, reg8	31
	AW, mem8	EA+34+W
	DW:AW, reg16	39
	DW:AW, mem16	EA+43+2W
DS0:		2
DS1:		2
EI		12
EXT	reg8, reg8	41-121
	reg8, imm4	42-122
FINT		2
FP01		55+5W [43+5W]
FP02		55+5W [43+5W]
HALT		N/A
IN	AL, imm8	15+W
	AW, imm8	15+W
	AL, DW	14+W
	AW, DW	14+W
INC	reg8	5
	reg16	2
	mem8	EA+13+2W [EA+11+2W]
	mem16	EA+13+2W [EA+9+2W]

Mnemonic	Operand	Clocks
INM	mem8, DW	21+2W [19+2W]
	mem16, DW	19+2W [15+2W]
	mem8, DW	18+(15+2W)n [18+(13+2W)n]
	mem16, DW	18+(13+2W)n [18+(9+2W)n]
INS	reg8, reg8	63-155
	reg8, imm4	64-156
LDEA		EA+2
LDM	mem8	13+W
	mem16	13+W
LDMB	mem16	16+ (11+W)n
LDMW	mem8	16+(10+W)n
MOV	reg8, reg8	2
	reg16, reg16	2
	reg8, mem8	EA+7+W
	reg16, mem16	EA+7+W
	mem8, reg8	EA+5+W [EA+2]
	mem16, reg16	EA+5+W [EA+2]
	reg8, imm8	5
	reg16, imm16	6
	mem8, imm8	EA+6+W
	mem16, imm16	EA+6+W
	AL, dmem8	10+W
	AW, dmem16	10+W
	dmem8, AL	8+W [5]
	dmem16, AW	8+W [5]
	sreg, reg16	4
	sreg, mem16	EA+9+W
	reg16, sreg	3
	mem16, sreg	EA+6+W [EA+3]
	AH, PSW	2
	PSW, AH	3
	DS0, reg16, memptr32	EA+17+2W
	DS1, reg16, memptr32	EA+17+2W
MOVBK	mem8, mem8	22+2W [17+W]
	mem16, mem16	22+2W [17+W]
MOVBKB	mem8, mem8	16+(18+2W)n [16+(13+W)n]
MOVBKW	mem16, mem16	16+(18+2W)n [16+(10+W)n]
MOVSPA		16
MOVSPB		11
MUL	AW, AL, reg8	31-40
	AW, AL, mem8	EA+34+W to EA+43+W
	DW:AW, AW, reg16	39-48
	DW:AW, AW, mem16	EA+42+W to EA+51+W
	reg16, reg16, imm8	39-49
	reg16, mem16, imm8	EA+42+W to EA+52+W
	reg16, reg16, imm16	40-50
	reg16, mem16, imm16	EA+43+W to EA+53+W

Instruction Clock Count (cont)

Mnemonic	Operand	Clocks
MULU	reg8	24
	mem8	EA+27+W
	reg16	32
	mem16	EA+33+W
NEG	reg8	5
	reg16	5
	mem8	EA+13+2W [EA+10+W]
	mem16	EA+13+2W [EA+10+W]
NOP		4
NOT	reg8	5
	reg16	5
	mem8	EA+13+2W [EA+10+W]
	mem16	EA+13+2W [EA+10+W]
NOT1	CY	2
	reg8, CL	7
	reg16, CL	7
	mem8, CL	EA+15+2W [EA+12+W]
	mem16, CL	EA+15+2W [EA+12+W]
	reg8, imm3	6
	reg16, imm4	6
	mem8, imm3	EA+12+2W [EA+9+W]
	mem16, imm4	EA+12+2W [EA+9+W]
OR	reg8, reg8	2
	reg16, reg16	2
	reg8, mem8	EA+7+W
	reg16, mem16	EA+7+W
	mem8, reg8	EA+10+2W [EA+7+W]
	mem16, reg16	EA+10+2W [EA+7+W]
	reg8, imm8	5
	reg16, imm16	6
	mem8, imm8	EA+11+2W [EA+9+2W]
	mem16, imm16	EA+12+2W [EA+8+2W]
	AL, imm8	5
	AW, imm16	6
OUT	imm8, AL	11+W
	imm8, AW	9+W
	DW, AL	10+W
	DW, AW	8+W
OUTM	DW, mem8	21+2W [19+2W]
	DW, mem16	19+2W [15+2W]
	DW, mem8	18+(15+2W)n [18+(13+2W)n]
	DW, mem16	18+(13+2W)n [18+(9+2W)n]
POLL		N/A
POP	reg16	11+W
	mem16	EA+14+2W [EA+11+W]
	DS1	12+W
	SS	12+W
	DS0	12+W
	PSW	13+W
	R	74+8W [58]

Mnemonic	Operand	Clocks
PREPARE	imm16, imm8	imm8 = 0: 26+W
		imm8 = 1: 37+2W
		imm8 = n > 1: 44+19 (n−1)+2nW
PS:		2
PUSH	reg16	13+W [9+W]
	mem16	EA+16+2W [EA+12+2W]
	DS1	10+W [7]
	PS	10+W [7]
	SS	10+W [7]
	DS0	10+W [7]
	PSW	9+W [6]
	R	74+8W [50]
	imm8	12+W [9]
	imm16	13+W [10]
REP		2
REPE		2
REPZ		2
REPC		2
REPNC		2
REPNE		2
REPNZ		2
RET	null	19+W
	pop-value	19+W
	null	27+2W
	pop-value	28+W
RETI		40+3W [34+W]
RETRBI		12
ROL	reg8, 1	8
	reg16, 1	8
	mem8, 1	EA+16+2W [EA+13+W]
	mem16, 1	EA+16+2W [EA+13+W]
	reg8, CL	11+2n
	reg16, CL	11+2n
	mem8, CL	EA+19+2W+2n [EA+16+W+2n]
	mem16, CL	EA+19+2W+2n [EA+16+W+2n]
	reg8, imm8	9+2n
	reg16, imm8	9+2n
	mem8, imm8	EA+15+2W+2n [EA+12+W+2n]
	mem16, imm8	EA+15+2W+2n [EA+12+W+2n]
ROL4	reg8	17
	mem8	EA+20+2W [EA+18+2W]
ROLC		Same as ROL
ROR		Same as ROL
ROR4	reg8	21
	mem8	EA+26+2W [EA+24+2W]
RORC		Same as ROL
SET1	CY	2
	DIR	2

4b

Instruction Clock Count (cont)

Mnemonic	Operand	Clocks
SET1 (cont)	reg8, CL	7
	reg16, CL	7
	mem8, CL	EA+15+2W [EA+12+W]
	mem16, CL	EA+15+2W [EA+12+W]
	reg8, imm3	6
	reg16, imm4	6
	mem8, imm3	EA+12+2W [EA+9+W]
	mem16, imm4	EA+12+2W [EA+9+W]
SHL		Same as ROL
SHR		Same as ROL
SHRA		Same as ROL
SS:		2
STM	mem8	13+W [10]
	mem16	13+W [10]
STMB	mem8	16+(9+W)n [16+(7+W)n]
STMW	mem16	16+(9+W)n [16+(5+W)n]
STOP		N/A
SUB		Same as ADD
SUB4S		22+(30+3W)n [22+(28+3W)n]
SUBC		Same as ADD
TEST	reg8, reg8	4
	reg16, reg16	4
	reg8, mem8	EA+12+W
	reg16, mem16	EA+11+2W
	mem8, reg8	EA+12+W
	mem16, reg16	EA+11+2W
	reg8, imm8	7
	reg16, imm16	8
	mem8, imm8	EA+9+W
	mem16, imm16	EA+10+W
	AL, imm8	5
	AW, imm16	6
TEST1	reg8, CL	7
	reg16, CL	7
	mem8, CL	EA+12+W
	mem16, CL	EA+12+W
	reg8, imm3	6
	reg16, imm4	6
	mem8, imm3	EA+9+W
	mem16, imm4	EA+9+W
TRANS		11+W
TRANSB		11+W
TSKSW		20

Mnemonic	Operand	Clocks
XCH	reg8, reg8	3
	reg16, reg16	3
	reg8, mem8	EA+12+2W [EA+9+W]
	reg16, mem16	EA+12+2W [EA+9+W]
	mem8, reg8	EA+12+2W [EA+9+W]
	mem16, reg16	EA+12+2W [EA+9+W]
	AW, reg16	4
	reg16, AW	4
XOR		Same as AND

Instruction Clock Count for Operations

	Byte		Word	
	RAM Enable	RAM Disable	RAM Enable	RAM Disable
Context switch interrupt	—	—	27	27
DMA (Single-step mode)	20 + 2W	20 + 2W	20 + 2W	20 + 2W
DMA (Demand release mode)	17.5 + W+ (13+W)•(n−1)	17.5 + W+ (13+W)•(n−1)	17.5 + W+ (13+W)•(n−1)	17.5 + W+ (13+W)•(n−1)
DMA (Burst mode)	20.5 + 2W+ (16+2W)•(n−1)	20.5 + 2W+ (16+2W)•(n−1)	20.5 + 2W+ (16+2W)•(n−1)	20.5 + 2W+ (16+2W)•(n−)
DMA (Single-transfer mode)	19.5 + W	19.5 +W	17 + W	17 + W
Interrupt (INT pin)	—	—	57 + 3W	57 + 3W
Macro service, sfr ← mem	25 + W	20 + W	25 +W	20 + W
Macro service, mem ← sfr	22 + W	21 + W	22 + W	21 + W
Macro service (Search char mode), sfr ← mem	28 + W	28 + W	—	—
Macro service (Search char mode), mem ← sfr	38 + W	35 + W	—	—
Priority interrupt (Vectored mode)	—	—	55 + 5W	55 + 5W
NMI (Vectored mode)	—	—	53 + 5W	53 + 5W

W = number of wait states inserted into external bus cycle
n = number of iterations
N = number of clocks to complete the instruction currently executing

Notes:

(1) Every interrupt (except NMI) has an additional associated latency time of 27 + N clocks. During the 27 clocks, the interrupt controller performs some overhead tasks such as arbitrating priority. This time should be added to the above listed interrupt and macro service execution times. NMI latency time is 18 + N clocks.

(2) The DMA and macro service clock counts listed are the required number of CPU clocks for each transfer.

(3) When an external interrupt is asserted, a maximum of 6 clocks is required for internal synchronization before the interrupt request flag is set. For an internal interrupt, a maximum of 2 clocks is required.

Pin Request Latency

	Clocks	
Source	Typ	Max
NMI pin	12 + N	18 + N
INT pin	8 + N	8 + N
All other interrupts	15 + N	27 + N
DMARQ pin		14 + N
HLDRQ pin		7 + 2W

Instruction Set

			Operation Code																No. of			Flags			
Mnemonic	**Operand**	**Operation**	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Bytes	AC	CY	V	P	S	Z
Data Transfer																									
MOV	reg, reg	reg ← reg	1	0	0	0	1	0	1	W	1	1	reg			reg			2						
	mem, reg	(mem) ← reg	1	0	0	0	1	0	0	W	mod		reg			mem			2-4						
	reg, mem	reg ← (mem)	1	0	0	0	1	0	1	W	mod		reg			mem			2-4						
	mem, imm	(mem) ← imm	1	1	0	0	0	1	1	W	mod		0	0	0	mem			3-6						
	reg, imm	reg ← imm	1	0	1	1	W	reg											2-3						
	acc, dmem	When W = 0 AL ← (dmem) When W = 1 AH ← (dmem + 1), AL ← (dmem)	1	0	1	0	0	0	0	W									3						
	dmem, acc	When W = 0 (dmem) ← AL When W = 1 (dmem + 1) ← AH, (dmem) ← AL	1	0	1	0	0	0	1	W									3						
	sreg, reg16	sreg ← reg16 sreg : SS, DS0, DS1	1	0	0	0	1	1	1	0	1	1	0	sreg		reg			2						
	sreg, mem16	sreg ← (mem16) sreg : SS, DS0, DS1	1	0	0	0	1	1	1	0	mod		0	sreg		mem			2-4						
	reg16, sreg	reg16 ← sreg	1	0	0	0	1	1	0	0	1	1	0	sreg		reg			2						
	mem16, sreg	(mem16) ← sreg	1	0	0	0	1	1	0	0	mod		0	sreg		mem			2-4						
	DS0, reg16, mem32	reg16 ← (mem32) DS0 ← (mem32 + 2)	1	1	0	0	0	1	0	1	mod		reg			mem			2-4						
	DS1, reg16, mem32	reg16 ← (mem32) DS1 ← (mem32 + 2)	1	1	0	0	0	1	0	0	mod		reg			mem			2-4						
	AH, PSW	AH ← S, Z, x, AC, x, P, x, CY	1	0	0	1	1	1	1	1									1						
	PSW, AH	S, Z, x, AC, x, P, x, CY ← AH	1	0	0	1	1	1	1	0									1	x	x		x	x	x
LDEA	reg16, mem16	reg16 ← mem16	1	0	0	0	1	1	0	1	mod		reg			mem			2-4						
TRANS	src-table	AL ← (BW + AL)	1	1	0	1	0	1	1	1									1						
XCH	reg, reg	reg ⟷ reg	1	0	0	0	0	1	1	W	1	1	reg			reg			2						
	mem, reg or reg, mem	(mem) ⟷ reg	1	0	0	0	0	1	1	W	mod		reg			mem			2-4						
	AW, reg16 or reg16, AW	AW ⟷ reg16	1	0	0	1	0	reg											1						
Repeat Prefixes																									
REPC		While CW ≠ 0, the next byte of the primitive block transfer instruction is executed and CW is decremented (− 1). If there is a waiting interrupt, it is processed. When CY ≠ 1, exit the loop.	0	1	1	0	0	1	0	1									1						
REPNC		While CW ≠ 0, the next byte of the primitive block transfer instruction is executed and CW is decremented (− 1). If there is a waiting interrupt, it is processed. When CY ≠ 0, exit the loop.	0	1	1	0	0	1	0	0									1						

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Repeat Prefixes (cont)																									
REP REPE REPZ		While CW ≠ 0, the next byte of the primitive block transfer instruction is executed and CW is decremented (– 1). If there is a waiting interrupt, it is processed. If the primitive block transfer instruction is CMPBK or CMPM and Z ≠ 1, exit the loop.	1	1	1	1	0	0	1	1									1						
REPNE REPNZ		While CW ≠ 0, the next byte of the primitive block transfer instruction is executed and CW is decremented (– 1). If there is a waiting interrupt, it is processed. If the primitive block transfer instruction is CMPBK or CMPM and Z ≠ 0, exit the loop.	1	1	1	1	0	0	1	0									1						
Primitive Block Transfer																									
MOVBK	dst-block, src-block	When W = 0 (IY) ⟵ (IX) DIR = 0: IX ⟵ IX + 1, IY ⟵ IY + 1 DIR = 1: IX ⟵ IX – 1, IY ⟵ IY – 1 When W = 1 (IY + 1, IY) ⟵ (IX + 1, IX) DIR = 0: IX ⟵ IX + 2, IY ⟵ IY + 2 DIR = 1: IX ⟵ IX – 2, IY ⟵ IY – 2	1	0	1	0	0	1	0	W									1						
CMPBK	src-block, dst-block	When W = 0 (IX) – (IY) DIR = 0: IX ⟵ IX + 1, IY ⟵ IY + 1 DIR = 1: IX ⟵ IX – 1, IY ⟵ IY – 1 When W = 1 (IX + 1, IX) – (IY + 1, IY) DIR = 0: IX ⟵ IX + 2, IY ⟵ IY + 2 DIR = 1: IX ⟵ IX – 2, IY ⟵ IY – 2	1	0	1	0	0	1	1	W									1	x	x	x	x	x	x
CMPM	dst-block	When W = 0 AL – (IY) DIR = 0: IY ⟵ IY + 1; DIR = 1: IY ⟵ IY – 1 When W = 1 AW – (IY + 1, IY) DIR = 0: IY ⟵ IY + 2; DIR = 1: IY ⟵ IY – 2	1	0	1	0	1	1	1	W									1	x	x	x	x	x	x
LDM	src-block	When W = 0 AL ⟵ (IX) DIR = 0: IX ⟵ IX + 1; DIR = 1: IX ⟵ IX – 1 When W = 1 AW ⟵ (IX + 1, IX) DIR = 0: IX ⟵ IX + 2; DIR = 1: IX ⟵ IX – 2	1	0	1	0	1	1	0	W									1						
STM	dst-block	When W = 0 (IY) ⟵ AL DIR = 0: IY ⟵ IY + 1; DIR = 1: IY ⟵ IY – 1 When W = 1 (IY + 1, IY) ⟵ AW DIR = 0: IY ⟵ IY + 2; DIR = 1: IY ⟵ IY – 2	1	0	1	0	1	0	1	W									1						
Bit Field Transfer																									
INS	reg8, reg8	16-Bit field ⟵ AW	0	0	0	0	1	1	1	1	0	0	1	1	0	0	0	1	3						
			1	1		reg			reg																
	reg8, imm4	16-Bit field ⟵ AW	0	0	0	0	1	1	1	1	0	0	1	1	1	0	0	1	4						
			1	1	0	0	0		reg																

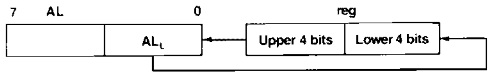
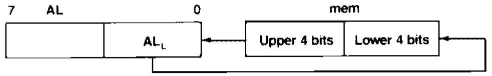
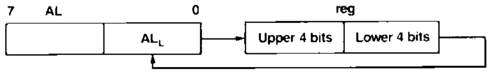
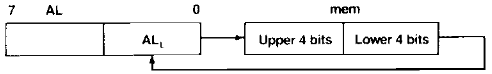
Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	No. of Bytes	AC	CY	V	P	S	Z
Bit Field Transfer (cont)											
EXT	reg8, reg8	AW ← 16-Bit field	0 0 0 0 1 1 1 1 1 1 reg reg	0 0 1 1 0 0 1 1	3						
	reg8, imm4	AW ← 16-Bit field	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 1 1 1 0 1 1	4						
I/O											
IN	acc, imm8	When W = 0 AL ← (imm8) When W = 1 AH ← (imm8 + 1), AL ← (imm8)	1 1 1 0 0 1 0 W		2						
	acc, DW	When W = 0 AL ← (DW) When W = 1 AH ← (DW + 1), AL ← (DW)	1 1 1 0 1 1 0 W		1						
OUT	imm8, acc	When W = 0 (imm8) ← AL When W = 1 (imm8 + 1) ← AH, (imm8) ← AL	1 1 1 0 0 1 1 W		2						
	DW, acc	When W = 0 (DW) ← AL When W = 1 (DW + 1) ← AH, (DW) ← AL	1 1 1 0 1 1 1 W		1						
Primitive Block I/O Transfer											
INM	dst-block, DW	When W = 0 (IY) ← (DW) DIR = 0: IY ← IY + 1; DIR = 1: IY ← IY − 1 When W = 1 (IY + 1, IY) ← (DW + 1, DW) DIR = 0: IY ← IY + 2; DIR = 1: IY ← IY − 2	0 1 1 0 1 1 0 W		1						
OUTM	DW, src-block	When W = 0 (DW) ← (IX) DIR = 0: IX ← IX + 1; DIR = 1: IX ← IX − 1 When W = 1 (DW + 1, DW) ← (IX + 1, IX) DIR = 0: IX ← IX + 2; DIR = 1: IX ← IX − 2	0 1 1 0 1 1 1 W	n: number of transfers	1						
Addition/Subtraction											
ADD	reg, reg	reg ← reg + reg	0 0 0 0 0 0 1 W	1 1 reg reg	2	x	x	x	x	x	x
	mem, reg	(mem) ← (mem) + reg	0 0 0 0 0 0 0 W	mod reg mem	2-4	x	x	x	x	x	x
	reg, mem	reg ← reg + (mem)	0 0 0 0 0 0 1 W	mod reg mem	2-4	x	x	x	x	x	x
	reg, imm	reg ← reg + imm	1 0 0 0 0 0 S W	1 1 0 0 0 reg	3-4	x	x	x	x	x	x
	mem, imm	(mem) ← (mem) + imm	1 0 0 0 0 0 S W	mod 0 0 0 mem	3-6	x	x	x	x	x	x
	acc, imm	When W = 0 AL ← AL + imm When W = 1 AW ← AW + imm	0 0 0 0 0 1 0 W		2-3	x	x	x	x	x	x

Instruction Set (cont)

			Operation Code																No. of	Flags					
Mnemonic	Operand	Operation	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Bytes	AC	CY	V	P	S	Z
Addition/Subtraction (cont)																									
ADDC	reg, reg	reg ⟵ reg + reg + CY	0	0	0	1	0	0	1	W	1	1		reg			reg		2	x	x	x	x	x	x
	mem, reg	(mem) ⟵ (mem) + reg + CY	0	0	0	1	0	0	0	W	mod			reg			mem		2-4	x	x	x	x	x	x
	reg, mem	reg ⟵ reg + (mem) + CY	0	0	0	1	0	0	1	W	mod			reg			mem		2-4	x	x	x	x	x	x
	reg, imm	reg ⟵ reg + imm + CY	1	0	0	0	0	0	S	W	1	1	0	1	0		reg		3-4	x	x	x	x	x	x
	mem, imm	(mem) ⟵ (mem) + imm + CY	1	0	0	0	0	0	S	W	mod		0	1	0		mem		3-6	x	x	x	x	x	x
	acc, imm	When W = 0 AL ⟵ AL + imm + CY When W = 1 AW ⟵ AW + imm + CY	0	0	0	1	0	1	0	W									2-3	x	x	x	x	x	x
SUB	reg, reg	reg ⟵ reg − reg	0	0	1	0	1	0	1	W	1	1		reg			reg		2	x	x	x	x	x	x
	mem, reg	(mem) ⟵ (mem) − reg	0	0	1	0	1	0	0	W	mod			reg			mem		2-4	x	x	x	x	x	x
	reg, mem	reg ⟵ reg − (mem)	0	0	1	0	1	0	1	W	mod			reg			mem		2-4	x	x	x	x	x	x
	reg, imm	reg ⟵ reg − imm	1	0	0	0	0	0	S	W	1	1	1	0	1		reg		3-4	x	x	x	x	x	x
	mem, imm	(mem) ⟵ (mem) − imm	1	0	0	0	0	0	S	W	mod		1	0	1		mem		3-6	x	x	x	x	x	x
	acc, imm	When W = 0 AL ⟵ AL − imm When W = 1 AW ⟵ AW − imm	0	0	1	0	1	1	0	W									2-3	x	x	x	x	x	x
SUBC	reg, reg	reg ⟵ reg − reg − CY	0	0	0	1	1	0	1	W	1	1		reg			reg		2	x	x	x	x	x	x
	mem, reg	(mem) ⟵ (mem) − reg − CY	0	0	0	1	1	0	0	W	mod			reg			mem		2-4	x	x	x	x	x	x
	reg, mem	reg ⟵ reg − (mem) − CY	0	0	0	1	1	0	1	W	mod			reg			mem		2-4	x	x	x	x	x	
	reg, imm	reg ⟵ reg − imm − CY	1	0	0	0	0	0	S	W	1	1	0	1	1		reg		3-4	x	x	x	x	x	x
	mem, imm	(mem) ⟵ (mem) − imm − CY	1	0	0	0	0	0	S	W	mod		0	1	1		mem		3-6	x	x	x	x	x	x
	acc, imm	When W = 0 AL ⟵ AL − imm − CY When W = 1 AW ⟵ AW − imm − CY	0	0	0	1	1	1	0	W									2-3	x	x	x	x	x	x

Instruction Set (cont)

			Operation Code																No. of		Flags				
Mnemonic	Operand	Operation	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Bytes	AC	CY	V	P	S	Z
BCD Operation																									
ADD4S		dst BCD string ← dst BCD string + src BCD string	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	2	u	x	u	u	u	x
SUB4S		dst BCD string ← dst BCD string − src BCD string	0	0	0	0	1	1	1	1	0	0	1	0	0	0	1	0	2	u	x	u	u	u	x
CMP4S		dst BCD string − src BCD string	0	0	0	0	1	1	1	1	0	0	1	0	0	1	1	0	2	u	x	u	u	u	x
ROL4	reg8		0	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0	3						
			1	1	0	0	0	reg																	
	mem8		0	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0	3-5						
			mod		0	0	0	mem																	
ROR4	reg8		0	0	0	0	1	1	1	1	0	0	1	0	1	0	1	0	3						
			1	1	0	0	0	reg																	
	mem8		0	0	0	0	1	1	1	1	0	0	1	0	1	0	1	0	3-5						
			mod		0	0	0	mem																	
BCD Adjust																									
ADJBA		When (AL AND 0FH) >9 or AC = 1, AL ← AL + 6, AH ← AH + 1, AC ← 1, CY ← AC, AL ← AL AND 0FH	0	0	1	1	0	1	1	1									1	x	x	u	u	u	u
ADJ4A		When (AL AND 0FH) >9 or AC = 1, AL ← AL + 6, CY ← CY OR AC, AC ← 1, When AL > 9FH, or CY — = AL ← AL + 60H, CY ← 1	0	0	1	0	0	1	1	1									1	x	x	u	x	x	x
ADJBS		When (AL AND 0FH) >9 or AC = 1, CY ← AC, AL ← AL AND 0FH	0	0	1	1	1	1	1	1									1	x	x	u	u	u	u
ADJ4S		When (AL AND 0FH) >9 or AC = 1, AL ← AL − 6, CY ← CY OR AC, AC ← 1, When AL > 9FH, or CY — = AL ← AL + 60H, CY ← 1	0	0	1	0	1	1	1	1									1	x	x	u	x	x	x

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	AC	Flags CY	V	P	S	Z
Increment/Decrement																									
INC	reg8	reg8 ← reg8 + 1	1	1	1	1	1	1	1	0	1	1	0	0	0	reg			2	x		x	x	x	x
	mem	(mem) ← (mem) + 1	1	1	1	1	1	1	1	W	mod		0	0	0	mem			2-4	x		x	x	x	x
	reg16	reg16 ← reg16 + 1	0	1	0	0	0	reg											1	x		x	x	x	x
DEC	reg8	reg8 ← reg8 − 1	1	1	1	1	1	1	1	0	1	1	0	0	1	reg			2	x		x	x	x	x
	mem	(mem) ← (mem) − 1	1	1	1	1	1	1	1	W	mod		0	0	1	mem			2-4	x		x	x	x	x
	reg16	reg16 ← reg16 − 1	0	1	0	0	1	reg											1	x		x	x	x	x
Multiplication																									
MULU	reg8	AW ← AL x reg8 AH = 0: CY ← 0, V ← 0 AH ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	1	1	1	0	0	reg			2	u	x	x	u	u	u
	mem8	AW ← AL x (mem8) AH = 0: CY ← 0, V ← 0 AH ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	mod		1	0	0	mem			2-4	u	x	x	u	u	u
	reg16	DW, AW ← AW x reg16 DW = 0: CY ← 0, V ← 0 DW ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	1	1	1	0	0	reg			2	u	x	x	u	u	u
	mem16	DW, AW ← AW x (mem16) DW = 0: CY ← 0, V ← 0 DW ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	mod		1	0	0	mem			2-4	u	x	x	u	u	u
MUL	reg8	AW ← AL x reg8 AH = AL sign expansion: CY ← 0, V ← 0 AH ≠ AL sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	1	1	1	0	1	reg			2	u	x	x	u	u	u
	mem8	AW ← AL x (mem8) AH = AL sign expansion: CY ← 0, V ← 0 AH ≠ AL sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	mod		1	0	1	mem			2-4	u	x	x	u	u	u
	reg16	DW, AW ← AW x reg16 DW = AW sign expansion: CY ← 0, V ← 0 DW ≠ AW sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	1	1	1	0	1	reg			2	u	x	x	u	u	u
	mem16	DW, AW ← AW x (mem16) DW = AW sign expansion: CY ← 0, V ← 0 DW ≠ AW sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	mod		1	0	1	mem			2-4	u	x	x	u	u	u
	reg16, reg16, imm8	reg16 ← reg16 x imm8 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	1	1	1	1	reg			reg			3	u	x	x	u	u	u
	reg16, mem16, imm8	reg16 ← (mem16) x imm8 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	1	1	mod		reg			mem			3-5	u	x	x	u	u	u

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Multiplication (cont)																									
MUL (cont)	reg16, reg16, imm16	reg16 ← reg16 x imm16 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	0	1	1	1	reg			reg			4	u	x	x	u	u	u
	reg16, mem16, imm16	reg16 ← (mem16) x imm16 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	0	1	mod		reg			mem			4-6	u	x	x	u	u	u
Unsigned Division																									
DIVU	reg8	temp ← AW When temp ÷ reg8 > FFH (SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % reg8, AL ← temp ÷ reg8	1	1	1	1	0	1	1	0	1	1	1	1	0	reg			2	u	u	u	u	u	u
	mem8	temp ← AW When temp ÷ (mem8) > FFH (SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % (mem8), AL ← temp ÷ (mem8)	1	1	1	1	0	1	1	0	mod		1	1	0	mem			2-4	u	u	u	u	u	u
	reg16	temp ← AW When temp ÷ reg16 > FFFFH (SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % reg16, AL ← temp ÷ reg16	1	1	1	1	0	1	1	1	1	1	1	1	0	reg			2	u	u	u	u	u	u
	mem16	temp ← AW When temp ÷ (mem16) > FFFFH (SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % (mem16), AL ← temp ÷ (mem16)	1	1	1	1	0	1	1	1	mod		1	1	0	mem			2-4	u	u	u	u	u	u

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	Flags AC	CY	V	P	S	Z
Signed Division																									
DIV	reg8	temp ← AW When temp ÷ reg8 > 0 and temp ÷ reg8 > 7FH or temp ÷ reg8 < 0 and temp ÷ reg8 < 0 - 7FH − 1 (SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % reg8, AL ← temp ÷ reg8	1	1	1	1	0	1	1	0	1	1	1	1	1		reg		2	u	u	u	u	u	u
	mem8	temp W ← When temp ÷ (mem8) > 0 and (mem8) > 7FH or temp ÷ (mem8) < 0 and temp ÷ (mem8) < 0 - 7FH — 1 (SP — 1, SP — 2) ← PSW, (SP — 3, SP — 4) ← PS (SP — 5, SP — 6) ← PC, SP ← SP — 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % (mem8), AL ← temp ÷ (mem8)	1	1	1	1	0	1	1	0	mod		1	1	1		mem		2-4	u	u	u	u	u	u
	reg 16	temp ← AW When temp ÷ reg 16 > 0 and reg 16 > 7FFFH or temp ÷ reg 16 < 0 - 7FFFH — 1 (SP — 1, SP — 2) ← PSW, (SP — 3, SP — 4) ← PS (SP — 5, SP — 6) ← PC, SP ← SP — 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % reg. 16, AL ← temp ÷ reg 16	1	1	1	1	0	1	1	1	1	1	1	1	1		reg		2	u	u	u	u	u	u
	mem 16	temp ← AW When temp ÷ (mem 16) > 0 and (mem 16) > 7FFFH or temp ÷ (mem 16) < 0 and temp ÷ **[mem 16]** < 0 - 7FFFH — 1 (SP — 1, SP — 2)) ← PSW, (SP —3, SP — 4) ← PS (SP — 5, SP — 6) ← PC, SP ← SP — 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % (mem 16), AL ← temp ÷ (mem 16)	1	1	1	1	0	1	1	1	mod		1	1	1		mem		2-4	u	u	u	u	u	u

4b

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code																No. of	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Bytes	AC	CY	V	P	S	Z
Data Conversion																									
CVTBD		AH ← AL ÷ 0AH, AL ← AL % 0AH	1	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	2	u	u	u	x	x	x
CVTDB		AH ← 0, AL ← AH x 0AH + AL	1	1	0	1	0	1	0	1	0	0	0	0	1	0	1	0	2	u	u	u	x	x	x
CVTBW		When AL < 80H, AH ← 0, all other times AH ← FFH	1	0	0	1	1	0	0	0									1						
CVTWL		When AL < 8000H, DW ← 0, all other times DW ← FFFFH	1	0	0	1	1	0	0	1									1						
Comparison																									
CMP	reg, reg	reg − reg	0	0	1	1	1	0	1	W	1	1	reg			reg			2	x	x	x	x	x	x
	mem, reg	(mem) − reg	0	0	1	1	1	0	0	W	mod		reg			mem			2-4	x	x	x	x	x	x
	reg, mem	reg − (mem)	0	0	1	1	1	0	1	W	mod		reg			mem			2-4	x	x	x	x	x	x
	reg, imm	reg − imm	1	0	0	0	0	0	S	W	1	1	1	1	1	reg			3-4	x	x	x	x	x	x
	mem, imm	(mem) − imm	1	0	0	0	0	0	S	W	mod		1	1	1	mem			3-6	x	x	x	x	x	x
	acc, imm	When W = 0, AL − imm When W = 1, AW − imm	0	0	1	1	1	1	0	W									2-3	x	x	x	x	x	x
Complement																									
NOT	reg	reg ← reg	1	1	1	1	0	1	1	W	1	1	0	1	0	reg			2						
	mem	(mem) ← (mem)	1	1	1	1	0	1	1	W	mod		0	1	0	mem			2-4						
NEG	reg	reg ← reg + 1	1	1	1	1	0	1	1	W	1	1	0	1	1	reg			2	x	x	x	x	x	x
	mem	(mem) ← (mem) + 1	1	1	1	1	0	1	1	W	mod		0	1	1	mem			2-4	x	x	x	x	x	x
Logical Operation																									
TEST	reg, reg	reg AND reg	1	0	0	0	0	1	0	W	1	1	reg			reg			2	u	0	0	x	x	x
	mem, reg or reg, mem	(mem) AND reg	1	0	0	0	0	1	0	W	mod		reg			mem			2-4	u	0	0	x	x	x
	reg, imm	reg AND imm	1	1	1	1	0	1	1	W	1	1	0	0	0	reg			3-4	u	0	0	x	x	x
	mem, imm	(mem) AND imm	1	1	1	1	0	1	1	W	mod		0	0	0	mem			3-6	u	0	0	x	x	x
	acc, imm	When W = 0, AL AND imm8 When W = 1, AW AND imm8	1	0	1	0	1	0	0	W									2-3	u	0	0	x	x	x
AND	reg, reg	reg ← reg AND reg	0	0	1	0	0	0	1	W	1	1	reg			reg			2	u	0	0	x	x	x
	mem, reg	(mem) ← (mem) AND reg	0	0	1	0	0	0	0	W	mod		reg			mem			2-4	u	0	0	x	x	x
	reg, mem	reg ← reg AND (mem)	0	0	1	0	0	0	1	W	mod		reg			mem			2-4	u	0	0	x	x	x
	reg, imm	reg ← reg AND imm	1	0	0	0	0	0	0	W	1	1	1	0	0	reg			3-4	u	0	0	x	x	x
	mem, imm	(mem) ← (mem) AND imm	1	0	0	0	0	0	0	W	mod		1	0	0	mem			3-6	u	0	0	x	x	x
	acc, imm	When W = 0, AL ← AL AND imm8 When W = 1, AW ← AW AND imm16	0	0	1	0	0	1	0	W									2-3	u	0	0	x	x	x

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code																No. of	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Bytes	AC	CY	V	P	S	Z
Logical Operation (cont)																									
OR	reg, reg	reg ← reg OR reg	0	0	0	0	1	0	1	W	1	1	reg			reg			2	u	0	0	x	x	x
	mem, reg	(mem) ← (mem) OR reg	0	0	0	0	1	0	0	W	mod		reg			mem			2-4	u	0	0	x	x	x
	reg, mem	reg ← reg OR (mem)	0	0	0	0	1	0	1	W	mod		reg			mem			2-4	u	0	0	x	x	x
	reg, imm	reg ← reg OR imm	1	0	0	0	0	0	0	W	1	1	0	0	1	reg			3-4	u	0	0	x	x	x
	mem, imm	(mem) ← (mem) OR imm	1	0	0	0	0	0	0	W	mod		0	0	1	mem			3-6	u	0	0	x	x	x
	acc, imm	When W = 0, AL ← AL OR imm8 When W = 1, AW ← AW OR imm16	0	0	0	0	1	1	0	W									2-3	u	0	0	x	x	x
XOR	reg, reg	reg ← reg XOR reg	0	0	1	1	0	0	1	W	1	1	reg			reg			2	u	0	0	x	x	x
	mem, reg	(mem) ← (mem) XOR reg	0	0	1	1	0	0	0	W	mod		reg			mem			2-4	u	0	0	x	x	x
	reg, mem	reg ← reg XOR (mem)	0	0	1	1	0	0	1	W	mod		reg			mem			2-4	u	0	0	x	x	x
	reg, imm	reg ← reg XOR imm	1	0	0	0	0	0	0	W	1	1	1	1	0	reg			3-4	u	0	0	x	x	x
	mem, imm	(mem) ← (mem) XOR imm	1	0	0	0	0	0	0	W	mod		1	1	0	mem			3-6	u	0	0	x	x	x
	acc, imm	When W = 0, AL ← AL XOR imm8 When W = 1, AW ← AW XOR imm16	0	0	1	1	0	1	0	W									2-3	u	0	0	x	x	x
Bit Operation																									
			2nd byte*								3rd byte*														
TEST1	reg8, CL	reg8 bit no. CL = 0: Z ← 1 reg8 bit no. CL = 1: Z ← 0	0	0	0	1	0	0	0	0	1	1	0	0	0	reg			3	u	0	0	u	u	x
	mem8, CL	(mem8) bit no. CL = 0: Z ← 1 (mem8) bit no. CL = 1: Z ← 0	0	0	0	1	0	0	0	0	mod		0	0	0	mem			3-5	u	0	0	u	u	x
	reg16, CL	reg16 bit no. CL = 0: Z ← 1 reg16 bit no. CL = 1: Z ← 0	0	0	0	1	0	0	0	1	1	1	0	0	0	reg			3	u	0	0	u	u	x
	mem16, CL	(mem16) bit no. CL = 0: Z ← 1 (mem16) bit no. CL = 1: Z ← 0	0	0	0	1	0	0	0	1	mod		0	0	0	mem			3-5	u	0	0	u	u	x
	reg8, imm3	reg8 bit no. imm3 = 0: Z ← 1 reg8 bit no. imm3 = 1: Z ← 0	0	0	0	1	1	0	0	0	1	1	0	0	0	reg			4	u	0	0	u	u	x
	mem8, imm3	(mem8) bit no. imm3 = 0: Z ← 1 (mem8) bit no. imm3 = 1: Z ← 0	0	0	0	1	1	0	0	0	mod		0	0	0	mem			4-6	u	0	0	u	u	x
	reg16, imm4	reg16 bit no. imm4 = 0: Z ← 1 reg16 bit no. imm4 = 1: Z ← 0	0	0	0	1	1	0	0	1	1	1	0	0	0	reg			4	u	0	0	u	u	x
	mem16, imm4	(mem16) bit no. imm4 = 0: Z ← 1 (mem16) bit no. imm4 = 1: Z ← 0	0	0	0	1	1	0	0	1	mod		0	0	0	mem			4-6	u	0	0	u	u	x
			2nd byte*								3rd byte*														

*Note: First byte = 0FH

Instruction Set (cont)

Mnemonic	Operand	Operation	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	AC	CY	V	P	S	Z
Bit Operation (cont)																									
			2nd byte*								3rd byte*														
NOT1	reg8, CL	reg8 bit no. CL ← reg8 bit no. CL	0	0	0	1	0	1	1	0	1	1	0	0	0	reg			3						
	mem8, CL	(mem8) bit no. CL ← (mem8) bit no. CL	0	0	0	1	0	1	1	0	mod		0	0	0	mem			3-5						
	reg16, CL	reg16 bit no. CL ← reg16 bit no. CL	0	0	0	1	0	1	1	1	1	1	0	0	0	reg			3						
	mem16, CL	(mem16) bit no. CL ← (mem16) bit no. CL	0	0	0	1	0	1	1	1	mod		0	0	0	mem			3-5						
	reg8, imm3	reg8 bit no. imm3 ← reg8 bit no. imm3	0	0	0	1	1	1	1	0	1	1	0	0	0	reg			4						
	mem8, imm3	(mem8) bit no. imm3 ← (mem8) bit no. imm3	0	0	0	1	1	1	1	0	mod		0	0	0	mem			4-6						
	reg16, imm4	reg16 bit no. imm4 ← (reg16) bit no. imm4	0	0	0	1	1	1	1	1	1	1	0	0	0	reg			4						
	mem16, imm4	(mem16) bit no. imm4 ← (mem16) bit no. imm4	0	0	0	1	1	1	1	1	mod		0	0	0	mem			4-6						
			2nd byte* *Note: First byte = 0FH								3rd byte*														
	CY	CY ← CY	1	1	1	1	0	1	0	1									1		x				
			2nd byte*								3rd byte*														
CLR1	reg8, CL	reg8 bit no. CL ← 0	0	0	0	1	0	0	1	0	1	1	0	0	0	reg			3						
	mem8, CL	(mem8) bit no. CL ← 0	0	0	0	1	0	0	1	0	mod		0	0	0	mem			3-5						
	reg16, CL	reg16 bit no. CL ← 0	0	0	0	1	0	0	1	1	1	1	0	0	0	reg			3						
	mem16, CL	(mem16) bit no. CL ← 0	0	0	0	1	0	0	1	1	mod		0	0	0	mem			3-5						
	reg8, imm3	reg8 bit no. imm3 ← 0	0	0	0	1	1	0	1	0	1	1	0	0	0	reg			4						
	mem8, imm3	(mem8) bit no. imm3 ← 0	0	0	0	1	1	0	1	0	mod		0	0	0	mem			4-6						
	reg16, imm4	reg16 bit no. imm4 ← 0	0	0	0	1	1	0	1	1	1	1	0	0	0	reg			4						
	mem16, imm4	(mem16) bit no. imm4 ← 0	0	0	0	1	1	0	1	1	mod		0	0	0	mem			4-6						
			2nd byte* *Note: First byte = 0FH								3rd byte*														
	CY	CY ← 0	1	1	1	1	1	0	0	0									1		0				
	DIR	DIR ← 0	1	1	1	1	1	1	0	0									1						

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code																No. of	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Bytes	AC	CY	V	P	S	Z
Bit Operation (cont)																									
SET1	reg8, CL	reg8 bit no. CL ← 1	0	0	0	1	0	1	0	0	1	1	0	0	0	reg			3						
	mem8, CL	(mem8) bit no. CL ← 1	0	0	0	1	0	1	0	0	mod		0	0	0	mem			3-5						
	reg16, CL	reg16 bit no. CL ← 1	0	0	0	1	0	1	0	1	1	1	0	0	0	reg			3						
	mem16, CL	(mem16) bit no. CL ← 1	0	0	0	1	0	1	0	1	mod		0	0	0	mem			3-5						
	reg8, imm3	reg8 bit no. imm3 ← 1	0	0	0	1	1	1	0	0	1	1	0	0	0	reg			4						
	mem8, imm3	(mem8) bit no. imm3 ← 1	0	0	0	1	1	1	0	0	mod		0	0	0	mem			4-6						
	reg16, imm4	reg16 bit no. imm4 ← 1	0	0	0	1	1	1	0	1	1	1	0	0	0	reg			4						
	mem16, imm4	(mem16) bit no. imm4 ← 1	0	0	0	1	1	1	0	1	mod		0	0	0	mem			4-6						
			2nd byte*								3rd byte*														
			*Note: First byte = 0FH																						
	CY	CY ← 1	1	1	1	1	1	0	0	1									1		1				
	DIR	DIR ← 1	1	1	1	1	1	1	0	1									1						
Shift																									
SHL	reg, 1	CY ← MSB of reg, reg ← reg x 2 When MSB of reg ≠ CY, V ← 1 When MSB of reg = CY, V ← 0	1	1	0	1	0	0	0	W	1	1	1	0	0	reg			2	u	x	x	x	x	x
	mem, 1	CY ← MSB of (mem), (mem) ← (mem) x 2 When MSB of (mem) ≠ CY, V ← 1 When MSB of (mem) = CY, V ← 0	1	1	0	1	0	0	0	W	mod		1	0	0	mem			2-4	u	x	x	x	x	x
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← MSB of reg, reg ← reg x 2, temp ← temp − 1	1	1	0	1	0	0	1	W	1	1	1	0	0	reg			2	u	x	u	x	x	x
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← MSB of (mem), (mem) ← (mem) x 2, temp ← temp − 1	1	1	0	1	0	0	1	W	mod		1	0	0	mem			2-4	u	x	u	x	x	x
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← MSB of reg, reg ← reg x 2, temp ← temp − 1	1	1	0	0	0	0	0	W	1	1	1	0	0	reg			3	u	x	u	x	x	x
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← MSB of (mem), (mem) ← (mem) x 2, temp ← temp − 1	1	1	0	0	0	0	0	W	mod		1	0	0	mem			3-5	u	x	u	x	x	x
								n: number of shifts																	
SHR	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2 When MSB of reg ≠ bit following MSB of reg: V ← 1 When MSB of reg = bit following MSB of reg: V ← 0	1	1	0	1	0	0	0	W	1	1	1	0	1	reg			2	u	x	x	x	x	x

4b

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0	No. of Bytes	Flags AC	CY	V	P	S	Z
Shift (cont)										
SHR (cont)	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2 When MSB of (mem) ≠ bit following MSB of (mem): V ← 1 When MSB of (mem) = bit following MSB of (mem): V ← 0	1 1 0 1 0 0 0 W mod 1 0 1 mem	2-4	u	x	x	x	x	x
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1	1 1 0 1 0 0 1 W 1 1 1 0 1 reg	2	u	x	u	x	x	x
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1	1 1 0 1 0 0 1 W mod 1 0 1 mem	2-4	u	x	u	x	x	x
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1	1 1 0 0 0 0 0 W 1 1 1 0 1 reg	3	u	x	u	x	x	x
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1	1 1 0 0 0 0 0 W mod 1 0 1 mem n: number of shifts	3-5	u	x	u	x	x	x
SHRA	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2, V ← 0 MSB of operand does not change	1 1 0 1 0 0 0 W 1 1 1 1 1 reg	2	u	x	0	x	x	x
	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2, V ← 0, MSB of operand does not change	1 1 0 1 0 0 0 W mod 1 1 1 mem	2-4	u	x	0	x	x	x
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1 MSB of operand does not change	1 1 0 1 0 0 1 W 1 1 1 1 1 reg	2	u	x	u	x	x	x
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1 MSB of operand does not change	1 1 0 1 0 0 1 W mod 1 1 1 mem	2-4	u	x	u	x	x	x
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1 MSB of operand does not change	1 1 0 0 0 0 0 W 1 1 1 1 1 reg	3	u	x	u	x	x	x
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1 MSB of operand does not change	1 1 0 0 0 0 0 W mod 1 1 1 mem n: number of shifts	3-5	u	x	u	x	x	x

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	No. of Bytes	AC	CY	V	P	S	Z
Rotation											
ROL	reg, 1	CY ← MSB of reg, reg ← reg x 2 + CY MSB of reg ≠ CY: V ← 1 MSB of reg = CY: V ← 0	1 1 0 1 0 0 0 W	1 1 0 0 0 reg	2		x	x			
	mem, 1	CY ← MSB of (mem), (mem) ← (mem) x 2 + CY MSB of (mem) ≠ CY: V ← 1 MSB of (mem) = CY: V ← 0	1 1 0 1 0 0 0 W	mod 0 0 0 mem	2-4		x	x			
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← MSB of reg, reg ← reg x 2 + CY temp ← temp − 1	1 1 0 1 0 0 1 W	1 1 0 0 0 reg	2		x	u			
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← MSB of (mem), (mem) ← (mem) x 2 + CY temp ← temp − 1	1 1 0 1 0 0 1 W	mod 0 0 0 reg	2-4		x	u			
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← MSB of reg, reg ← reg x 2 + CY temp ← temp − 1	1 1 0 0 0 0 0 W	1 1 0 0 0 reg	3		x	u			
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← MSB of (mem), (mem) ← (mem) x 2 + CY temp ← temp − 1	1 1 0 0 0 0 0 W	mod 0 0 0 mem n: number of shifts	3-5		x	u			
ROR	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2 MSB of reg ← CY MSB of reg ≠ bit following MSB of reg: V ← 1 MSB of reg = bit following MSB of reg: V ← 0	1 1 0 1 0 0 0 W	1 1 0 0 1 reg	2		x	x			
	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2 MSB of (mem) ← CY MSB of (mem) ≠ bit following MSB of (mem): V ← 1 MSB of (mem) = bit following MSB of (mem): V ← 0	1 1 0 1 0 0 0 W	mod 0 0 1 mem	2-4		x	x			
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← CY temp ← temp − 1	1 1 0 1 0 0 1 W	1 1 0 0 1 reg	2		x	u			
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, MSB of (mem) ← CY temp ← temp − 1	1 1 0 1 0 0 1 W	mod 0 0 1 mem n:number of shifts	2-4		x	u			

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code																No. of Bytes	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		AC	CY	V	P	S	Z
Rotation (cont)																									
ROR (cont)	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← CY temp ← temp − 1	1	1	0	0	0	0	0	W	1	1	0	0	1	reg			3		x	u			
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2 temp ← temp − 1	1	1	0	0	0	0	0	W	mod		0	0	1	mem			3-5		x	u			
			n: number of shifts																						
Rotate																									
ROLC	reg, 1	tmpcy ← CY, CY ← MSB of reg reg ← reg x 2 + tmpcy MSB of reg = CY: V ← 0 MSB of reg ≠ CY: V ← 1	1	1	0	1	0	0	0	W	1	1	0	1	0	reg			2		x	x			
	mem, 1	tmpcy ← CY, CY ← MSB of (mem) (mem) ← (mem) x 2 + tmpcy MSB of (mem) = CY: V ← 0 MSB of (mem) ≠ CY: V ← 1	1	1	0	1	0	0	0	W	mod		0	1	0	mem			2-4		x	x			
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of reg, reg ← reg x 2 + tmpcy temp ← temp − 1	1	1	0	1	0	0	1	W	1	1	0	1	0	reg			2		x	u			
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of (mem), (mem) ← (mem) x 2 + tmpcy temp ← temp − 1	1	1	0	1	0	0	1	W	mod		0	1	0	mem			2-4		x	u			
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of reg, reg ← reg x 2 + tmpcy temp ← temp − 1	1	1	0	0	0	0	0	W	1	1	0	1	0	reg			3		x	u			
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of (mem) (mem) ← (mem) x 2 + tmpcy temp ← temp − 1	1	1	0	0	0	0	0	W	mod		0	1	0	mem			3-5		x	u			
			n: number of shifts																						

Instruction Set (cont)

Mnemonic	Operand	Operation	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	No. of Bytes	AC	CY	V	P	S	Z
Rotate (cont)																									
RORC	reg, 1	tmpcy ← CY, CY ← LSB of reg reg ← reg ÷ 2, MSB of reg ← tmpcy MSB of reg ≠ bit following MSB of reg: V ← 1 MSB of reg = bit following MSB of reg: V ← 0	1	1	0	1	0	0	0	W	1	1	0	1	1	reg			2		x	x			
	mem, 1	tmpcy ← CY, CY ← LSB of (mem) (mem) ← (mem) ÷ 2, MSB of (mem) ← tmpcy MSB of (mem) ≠ bit following MSB of (mem): V ← 1 MSB of (mem) = bit following MSB of (mem): V ← 0	1	1	0	1	0	0	0	W	mod		0	1	1	mem			2-4		x	x			
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← tmpcy, temp ← temp − 1	1	1	0	1	0	0	1	W	1	1	0	1	1	reg			2		x	u			
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← LSB of (mem), (mem) ← (mem) ÷ 2 MSB of (mem) ← tmpcy, temp ← temp − 1	1	1	0	1	0	0	1	W	mod		0	1	1	mem			2-4		x	u			
	reg, imm8	temp ← imm8, while temp ≠ 0 repeat this operation, tmpcy ← CY, CY ← LSB of reg, reg ← reg ÷ 2 MSB of reg ← tmpcy, temp ← temp − 1	1	1	0	0	0	0	0	W	1	1	0	1	1	reg			3		x	u			
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← LSB of (mem), (mem) ← (mem) ÷ 2 MSB of (mem) ← tmpcy, temp ← temp − 1	1	1	0	0	0	0	0	W	mod		0	1	1	mem			3-5		x	u			
Subroutine Control Transfer																									
CALL	near-proc	(SP − 1, SP − 2) ← PC, SP ← SP − 2 PC ← PC + disp	1	1	1	0	1	0	0	0									3						
	regptr16	(SP − 1, SP − 2) ← PC, SP ← SP − 2 PC ← regptr16	1	1	1	1	1	1	1	1	1	1	0	1	0	reg			2						
	memptr16	(SP − 1, SP − 2) ← PC, SP ← SP − 2 PC ← (memptr16)	1	1	1	1	1	1	1	1	mod		0	1	0	mem			2-4						
	far-proc	(SP − 1, SP − 2) ← PS, (SP − 3, SP − 4) ← PC SP ← SP − 4, PS ← seg, PC ← offset	1	0	0	1	1	0	1	0									5						
	memptr32	(SP − 1, SP − 2) ← PS, (SP − 3, SP − 4) ← PC SP ← SP − 4, PS ← (memptr32 + 2), PC ← (memptr32)	1	1	1	1	1	1	1	1	mod		0	1	1	mem			2-4						

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	No. of Bytes	AC	CY	V	P	S	Z
Subroutine Control Transfer (cont)											
RET		PC ← (SP + 1, SP), SP ← SP + 2	1 1 0 0 0 0 1 1		1						
	pop-value	PC ← (SP + 1, SP) SP ← SP + 2, SP ← SP + pop-value	1 1 0 0 0 0 1 0		3						
		PC ← (SP + 1, SP), PS ← (SP + 3, SP + 2) SP ← SP + 4	1 1 0 0 1 0 1 1		1						
	pop-value	PC ← (SP + 1, SP), PS ← (SP + 3, SP + 2) SP ← SP + 4, SP ← SP + pop-value	1 1 0 0 1 0 1 0		3						
Stack Manipulation											
PUSH	mem16	(SP − 1, SP − 2) ← (mem16), SP ← SP − 2	1 1 1 1 1 1 1 1	mod 1 1 0 mem	2-4						
	reg16	(SP − 1, SP − 2) ← reg16, SP ← SP − 2	0 1 0 1 0 reg		1						
	sreg	(SP− 1, SP− 2) ← sreg, SP ← SP − 2	0 0 0 sreg 1 1 0		1						
	PSW	(SP − 1, SP − 2) ← PSW, SP ← SP − 2	1 0 0 1 1 1 0 0		1						
	R	Push registers on the stack	0 1 1 0 0 0 0 0		1						
	imm	(SP − 1, SP − 2) ← imm SP ← SP − 2, When S = 1, sign extension	0 1 1 0 1 0 S 0		2-3						
POP	mem16	(mem16) ← (SP + 1, SP), SP ← SP + 2	1 0 0 0 1 1 1 1	mod 0 0 0 mem	2-4						
	reg16	reg16 ← (SP + 1, SP), SP ← SP + 2	0 1 0 1 1 reg		1						
	sreg	sreg ← (SP + 1, SP) sreg : SS, DS0, DS1 SP ← SP + 2	0 0 0 sreg 1 1 1		1						
	PSW	PSW ← (SP + 1, SP), SP ← SP + 2	1 0 0 1 1 1 0 1		1	R	R	R	R	R	R
	R	Pop registers from the stack	0 1 1 0 0 0 0 1		1						
PREPARE	imm16, imm8	Prepare new stack frame	1 1 0 0 1 0 0 0 *: imm8 = 0: 16 imm8 > 1: 25 + 16 (imm8 − 1)		4						
DISPOSE		Dispose of stack frame	1 1 0 0 1 0 0 1		1						
Branch											
BR	near-label	PC ← PC + disp	1 1 1 0 1 0 0 1		3						
	short-label	PC ← PC + ext-disp8	1 1 1 0 1 0 1 1		2						
	regptr16	PC ← regptr16	1 1 1 1 1 1 1 1	1 1 1 0 0 reg	2						
	memptr16	PC ← (memptr16)	1 1 1 1 1 1 1 1	mod 1 0 0 mem	2-4						
	far-label	PS ← seg, PC ← offset	1 1 1 0 1 0 1 0		5						
	memptr32	PS ← (memptr32 + 2), PC ← (memptr32)	1 1 1 1 1 1 1 1	mod 1 0 1 mem	2-4						

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	No. of Bytes	Flags AC	CY	V	P	S	Z
Conditional Branch											
BV	short-label	if V = 1, PC ← PC + ext-disp8	0 1 1 1 0 0 0 0		2						
BNV	short-label	if V = 0, PC ← PC + ext-disp8	0 1 1 1 0 0 0 1		2						
BC, BL	short-label	if CY = 1, PC ← PC + ext-disp8	0 1 1 1 0 0 1 0		2						
BNC, BNL	short-label	if CY = 0, PC ← PC + ext-disp8	0 1 1 1 0 0 1 1		2						
BE, BZ	short-label	if Z = 1, PC ← PC + ext-disp8	0 1 1 1 0 1 0 0		2						
BNE, BNZ	short-label	if Z = 0, PC ← PC + ext-disp8	0 1 1 1 0 1 0 1		2						
BNH	short-label	if CY OR Z = 1, PC ← PC + ext-disp8	0 1 1 1 0 1 1 0		2						
BH	short-label	if CY OR Z = 0, PC ← PC + ext-disp8	0 1 1 1 0 1 1 1		2						
BN	short-label	if S = 1, PC ← PC + ext-disp8	0 1 1 1 1 0 0 0		2						
BP	short-label	if S = 0, PC ← PC + ext-disp8	0 1 1 1 1 0 0 1		2						
BPE	short-label	if P = 1, PC ← PC + ext-disp8	0 1 1 1 1 0 1 0		2						
BPO	short-label	if P = 0, PC ← PC + ext-disp8	0 1 1 1 1 0 1 1		2						
BLT	short-label	if S XOR V = 1, PC ← PC + ext-disp8	0 1 1 1 1 1 0 0		2						
BGE	short-label	if S XOR V = 0, PC ← PC + ext-disp8	0 1 1 1 1 1 0 1		2						
BLE	short-label	if (S XOR V) OR Z = 1, PC ← PC + ext-disp8	0 1 1 1 1 1 1 0		2						
BGT	short-label	if (S XOR V) OR Z = 0, PC ← PC + ext-disp8	0 1 1 1 1 1 1 1		2						
DBNZNE	short-label	CW ← CW − 1 if Z = 0 and CW ≠ 0, PC ← PC + ext-disp8	1 1 1 0 0 0 0 0		2						
DBNZE	short-label	CW ← CW − 1 if Z = 1 and CW ≠ 0, PC ← PC + ext-disp8	1 1 1 0 0 0 0 1		2						
DBNZ	short-label	CW ← CW − 1 if CW ≠ 0, PC ← PC + ext-disp8	1 1 1 0 0 0 1 0		2						
BCWZ	short-label	if CW = 0, PC ← PC + ext-disp8	1 1 1 0 0 0 1 1		2						
BTCLR	sfr. imm3, short-label	if bit no. imm3 of (sfr) = 1, PC ← PC + ext — disp8, bit no. imm3 of (sfr) ← 0	0 0 0 0 1 1 1 1	1 0 1 1 1 1 0 0	5						
Interrupt											
BRK	3	(SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS, (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0 PS ← (15, 14), PC ← (13, 12)	1 1 0 0 1 1 0 0		1						
	imm8 (≠ 3)	(SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS, (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0 PC ← (n x 4, + 1, n x 4) PS ← (n x 4 + 3, n x 4 + 2) n = imm8	1 1 0 0 1 1 0 1		2						

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0	No. of Bytes	AC	CY	V	P	S	Z
Interrupt (cont)										
BRKV		When V = 1 (SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS, (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0 PS ← (19, 18), PC ← (17, 16)	1 1 0 0 1 1 1 0	1						
RETI		PC ← (SP + 1, SP), PS ← (SP + 3, SP + 2), PSW ← (SP + 5, SP + 4), SP ← SP + 6	1 1 0 0 1 1 1 1	1	R	R	R	R	R	R
RETRBI		PC ← Save PC, PSW ← Save PSW	0 0 0 0 1 1 1 1 1 0 0 1 0 0 0 1	2	R	R	R	R	R	R
FINT		Indicates that interrupt service routine to the interrupt controller built in the CPU has been completed	0 0 0 0 1 1 1 1 1 0 0 1 0 0 1 0	2						
CHKIND	reg16, mem32	When (mem32) > reg16 or (mem32 + 2) < reg16 (SP − 1, SP − 2) ← PSW, (SP − 3, SP − 4) ← PS, (SP − 5, SP − 6) ← PC, SP ← SP − 6 IE ← 0, BRK ← 0, PS ← (23, 22), PC ← (21, 20)	0 1 1 0 0 0 1 0 mod reg mem	2-4						
CPU Control										
HALT		CPU Halt	1 1 1 1 0 1 0 0	1						
STOP		CPU Halt	0 0 0 0 1 1 1 1 1 0 1 1 1 1 1 0	1						
BUSLOCK		Bus Lock Prefix	1 1 1 1 0 0 0 0	1						
FP01 (Note 1)	fp-op	No Operation	1 1 0 1 1 X X X 1 1 Y Y Y Z Z Z	2						
	fp-op, mem	data bus ← (mem)	1 1 0 1 1 X X X mod Y Y Y mem	2-4						
FP02 (Note 1)	fp-op	No Operation	0 1 1 0 0 1 1 X 1 1 Y Y Y Z Z Z	2						
	fp-op, mem	data bus ← (mem)	0 1 1 0 0 1 1 X mod Y Y Y mem	2-4						
POLL		Poll and wait	1 0 0 1 1 0 1 1	1						
NOP		No Operation	1 0 0 1 0 0 0 0	1						
DI		IE ← 0	1 1 1 1 1 0 1 0	1						
EI		IE ← 1	1 1 1 1 1 0 1 1	1						
DS0; DS1; PS; SS		Segment override prefix	0 0 1 sreg 1 1 0	1						

Notes:

(1) Does not execute on the V25, but does generate an interrupt.

Register Banks

MOVSPA		0 0 0 0 1 1 1 1 0 0 1 0 0 1 0 1	2						
BRKCS	reg16	0 0 0 0 1 1 1 1 0 0 1 0 1 1 0 1	3						
MOVSPB	reg16	0 0 0 0 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 1 reg	3						
TSKSW	reg16	0 0 0 0 1 1 1 1 1 0 0 1 0 1 0 0 1 1 1 1 1 reg	3	x	x	x	x	x	x