NEC Electronics Inc.

μPD70335 (V35 Plus) 16-Bit Microcomputer: Advanced, High-Speed DMA, Single-Chip, CMOS

T-49-19-59

Description

The μPD70335 (V35 Plus) is a high-performance, 16-bit single-chip microcomputer with a 16-bit external data bus. The μPD70335 is fully software compatible with the μPD70108/116 (V20®/V30®) as well as the μPD70320/330 (V25™/V35™). The V35 Plus demonstrates numerous enhancements over the standard V35; however, it maintains strict pin compatibility with its predecessor, the V35.

The V35 Plus offers improved DMA transfer rates (over 5M bytes per second), additional serial channel status flags, improved memory access timing, and enhanced software control of register bank context switching.

The μPD70335 has the same complement of internal peripherals as the V35, and maintains compatibility with existing drivers; however, some modification of the DMA drivers may be necessary. The μPD70335 does not offer on-chip ROM or EPROM.

Features

- 16-bit CPU and internal data paths
- 16-bit non-multiplexed external data path
- Direct RAS/CAS DRAM interface
- Functional and pin compatibility with the V35
- Software compatible with μPD8086
- New and enhanced V-Series instructions
- Minimum instruction cycle 200 ns (at 10 MHz)
- 6-byte prefetch queue
- Two-channel high-speed DMA controller
- Internal 256 bytes RAM memory
- One 1M-byte memory address space
- Eight internal memory-mapped register banks
- Four multifunction I/O ports
  - 8-bit analog comparator port
  - 20 bidirectional port lines
  - 4 input-only port lines
- Two independent full-duplex serial channels
- Priority interrupt controller
  - Standard vectored service
  - Register bank switching
  - Macroservice
- Pseudo-SRAM and DRAM refresh controller
- Two 16-bit timers
- On-chip time base counter
- Programmable wait state generator
- Two standby modes: STOP and HALT

Ordering Information

4e

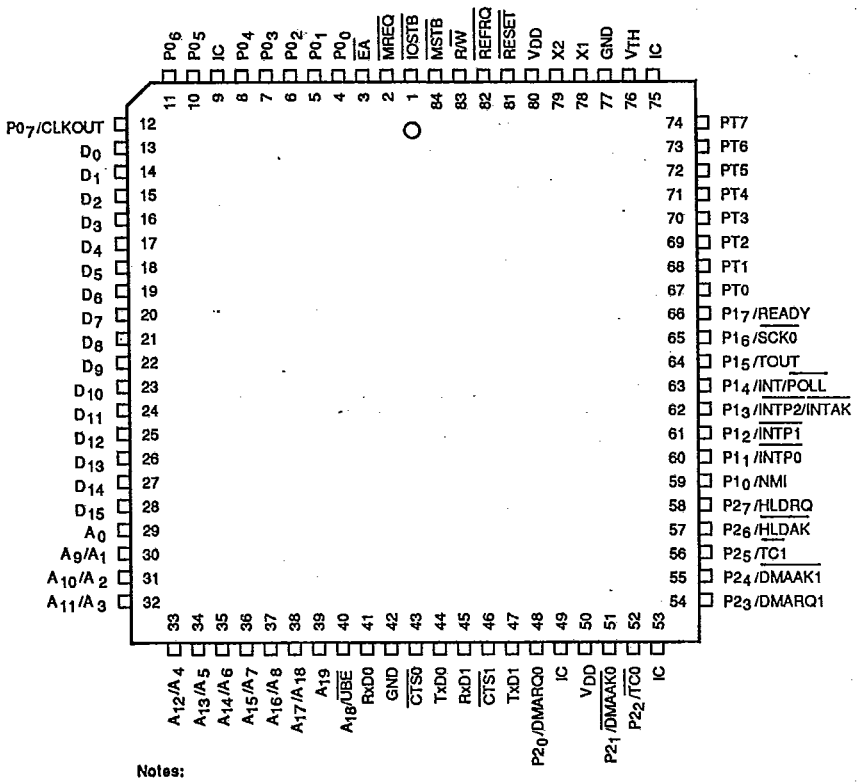
Part Number	Clock (MHz)	Package
μPD70335L-8	8	84-pin PLCC
L-10	10	
GJ-8	8	94-pin plastic QFP
GJ-10	10	

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

50155

Pin Configuration

84-Pin PLCC



Notes:

(1) Pin functions are identical to μPD70330.

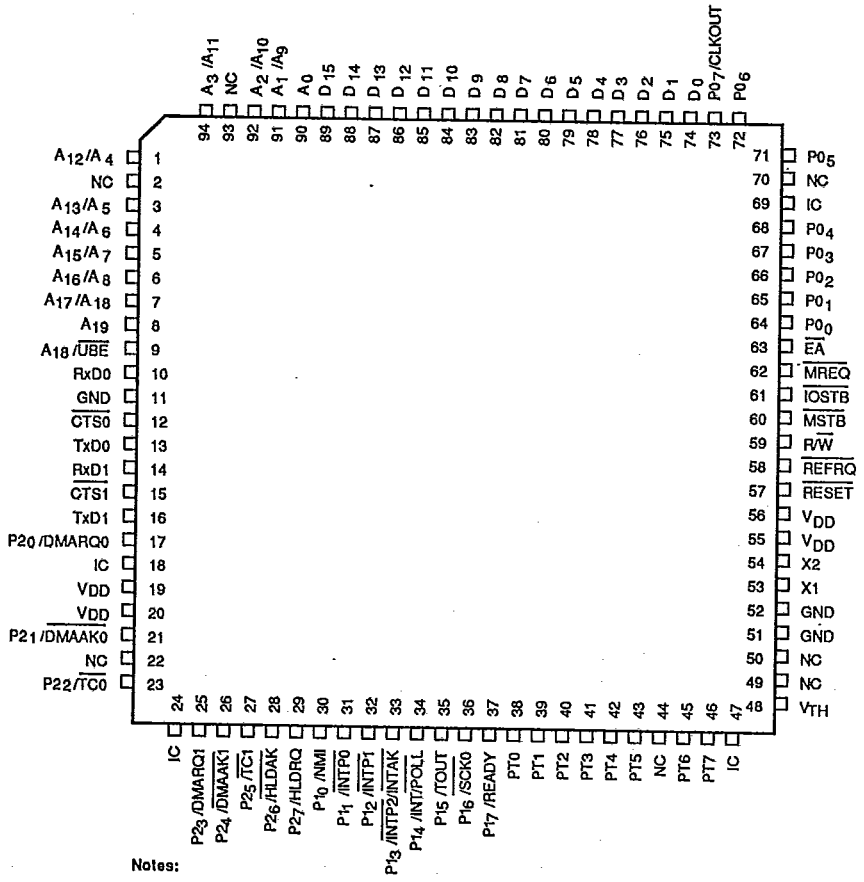
(2) IC pins should be tied together and pulled up to V_{DD} with a 10- to 20-kΩ resistor.

(3) $\overline{EA}$ must be tied low because μPD70335 does not support internal ROM or EPROM.

(4) Pin 9 should be tied to GND through a pull-down resistor.

83SL-6827B

94-Pin Plastic QFP



Notes:

(1) Pin functions are identical to μPD70330.

(2) IC pins should be tied together and pulled up to V_{DD} with a 10- to 20-kΩ resistor.

(3) $\overline{EA}$ must be tied low because μPD70335 does not support internal ROM or EPROM.

(4) Pin 69 should be tied to GND through a pull-down resistor.

83SL-6628B

Pin Identification

Symbol	Function
A_0-A_{19}	Address bus outputs
CLKOUT	System clock output
$\overline{CTS0}$	Clear-to-send input, serial channel 0
$\overline{CTS1}$	Clear-to-send input, serial channel 1
D_0-D_{15}	Bidirectional data bus
$\overline{DMAAK0}$	DMA acknowledge output, DMA controller channel 0
$\overline{DMAAK1}$	DMA acknowledge output, DMA controller channel 1
DMARQ0	DMA request input, DMA controller channel 0
DMARQ1	DMA request input, DMA controller channel 1
$\overline{EA}$	External memory access; fixed low for V35 Plus
$\overline{HLDAK}$	Hold acknowledge output
HLDRQ	Hold request input
INT	Interrupt request input
$\overline{INTAK}$	Interrupt acknowledge output
$\overline{INTP0}$	Interrupt request 0 input
$\overline{INTP1}$	Interrupt request 1 input
$\overline{INTP2}$	Interrupt request 2 input
$\overline{IOSTB}$	I/O read or write strobe output
$\overline{MREQ}$	Memory request output
$\overline{MSTB}$	Memory strobe output
NMI	Nonmaskable interrupt request
$P0_0$-$P0_7$	I/O port 0
$P1_0$-$P1_7$	I/O port 1
$P2_0$-$P2_7$	I/O port 2
$\overline{POLL}$	Input on $\overline{POLL}$ synchronizes the CPU and external devices
PT0-PT7	Comparator port input lines
READY	Ready signal input controls insertion of wait states
$\overline{REFRQ}$	DRAM refresh request output
$\overline{RESET}$	Reset signal input
$R/\overline{W}$	Read/write strobe output
RxD0	Receive data input, serial channel 0
RxD1	Receive data input, serial channel 1
$\overline{SCK0}$	Serial clock output
$\overline{TC0}$	Terminal count output; DMA completion, channel 0
$\overline{TC1}$	Terminal count output; DMA completion, channel 1
TOUT	Timer output
TxD0	Transmit data output, serial channel 0
TxD1	Transmit data output, serial channel 1
$\overline{UBE}$	Upper byte enable
X1, X2	Connections to external frequency control source (crystal, ceramic resonator, or clock)

Symbol	Function
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_0-A_{19} (Address Bus)

To support dynamic RAMs, the 20-bit address is multiplexed on 11 lines. When $\overline{MREQ}$ is asserted, A_9-A_{17} are valid. When $\overline{MSTB}$ or $\overline{IOSTB}$ is asserted, A_1-A_8 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_0-D_{15} (Data Bus)

D_0-D_{15} 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).

$\overline{DMARQ0}$ and $\overline{DMARQ1}$ (DMA Request)

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

$\overline{EA}$ (External Access)

This pin must be externally fixed low. Since the μPD70335 has no internal ROM, this will force execution of program code from external memory instead of 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-Ω pullup resistors: A_0-A_{19}, D_0-D_7, $\overline{IOSTB}$, $\overline{MREQ}$, $\overline{MSTB}$, $\overline{REFRQ}$, and R/$\overline{W}$.

HLDRQ (Hold Request)

The HLDRQ input from an external device requests that the μPD70335 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_0-D_7 in synchronization with $\overline{INTAK}$.

$\overline{INTP0}$, $\overline{INTP1}$, $\overline{INTP2}$ (Interrupt from Peripheral 0, 1, 2)

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

The $\overline{INTPn}$ inputs can be used to release the HALT mode.

$\overline{IOSTB}$ (I/O Strobe)

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

$\overline{MREQ}$ (Memory Request)

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

$\overline{MSTB}$ (Memory Strobe)

Together with $\overline{MREQ}$ and R/$\overline{W}$, $\overline{MSTB}$ controls memory-accessing operations. $\overline{MSTB}$ should be used either to enable data buffers or as a data strobe. During memory write, a low-level output on $\overline{MSTB}$ indicates that data on the data bus is valid and that multiplexed address bits A_1-A_8, A_{18} and $\overline{UBE}$ 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 several clock cycles. When the NMI is accepted, a number 2 vector interrupt is generated after completion of the instruction currently being executed.

4e

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

$P0_0$-$P0_7$ (Port 0)

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

$P1_0$-$P1_7$ (Port 1)

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

$P2_0$-$P2_7$ (Port 2)

$P2_0$-$P2_7$ 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.

$\overline{POLL}$ (Poll)

The $\overline{POLL}$ 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 $\overline{POLL}$ input is checked every five clock cycles until the level becomes low. The $\overline{POLL}$ functions are used to synchronize the CPU program and the operation of external devices.

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

PT0-PT7 (Port with Comparator)

The threshold port (PT) comprises 8 independent input bits, each of which is compared with a threshold voltage programmable to one of 16 voltage steps.

READY (Ready)

After READY is de-asserted low, the CPU will synchronize and insert 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. Use of the READY pin is controlled by the WTC register.

$\overline{REFRQ}$ (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. $\overline{REFRQ}$ also signals that A_0-A_8 contain a valid row address.

$\overline{RESET}$ (Reset)

This input signal is asynchronous. A low on $\overline{RESET}$ for the specified 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 an external 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 to 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.

$\overline{SCK0}$ (Serial Clock)

The $\overline{SCK0}$ 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 unit zero.

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 transmitter is idle.

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

$\overline{UBE}$ (Upper Byte Enable)

$\overline{UBE}$ is a high-order memory bank selection signal output. $\overline{UBE}$ and A_0 determine which bytes of the data bus will be used. UBE is used 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

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} (Power Supply)

+5-volt power source (two pins).

V_{TH} (Threshold Voltage)

Comparator port PT0-PT7 uses threshold voltage V_{TH} to determine the analog reference points. The actual threshold each comparator input is tested against is programmable to V_{TH} x n/16 where n = 1 to 16.

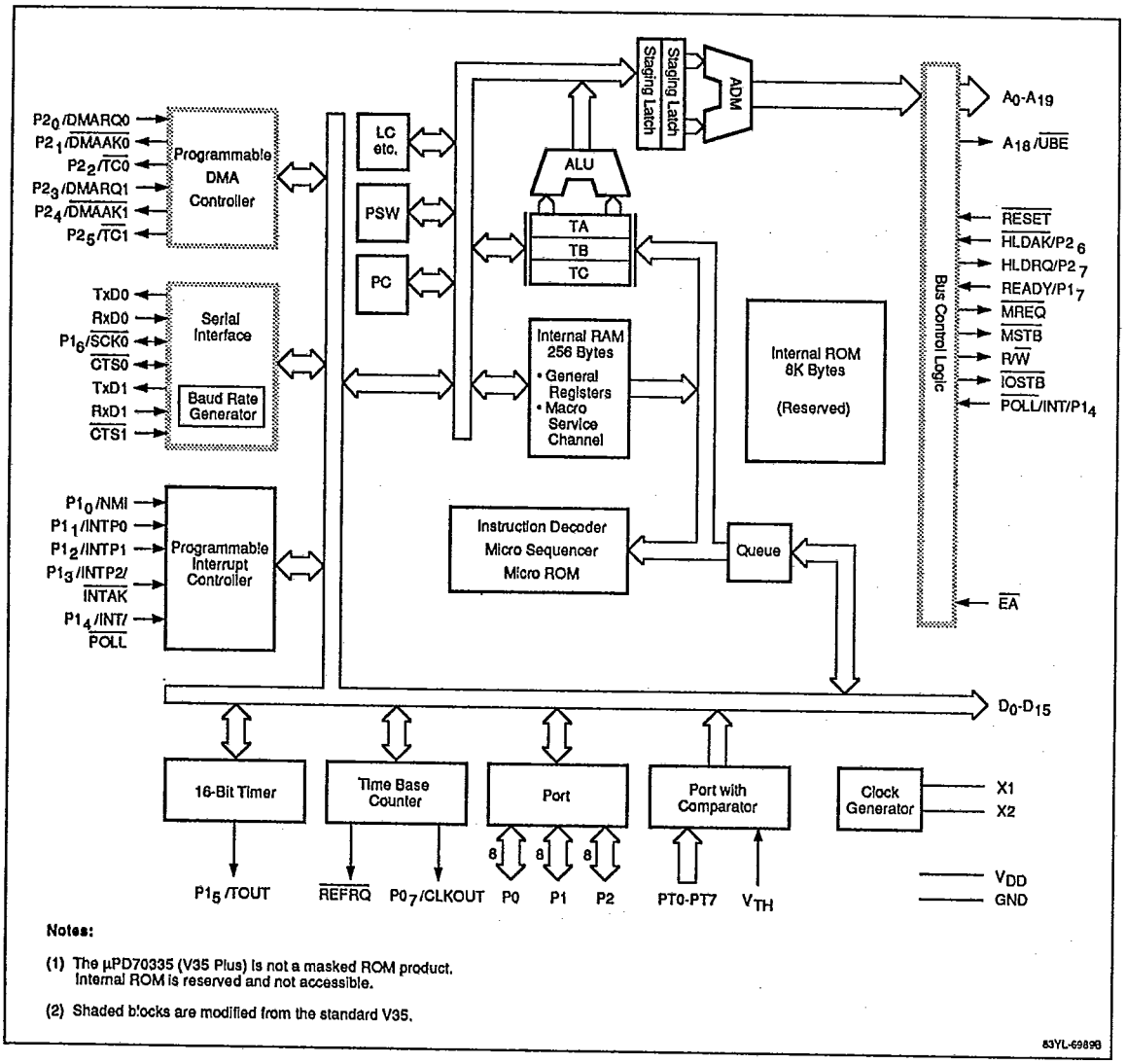
GND (Ground)

Ground reference (multiple pins).

IC (Internal Connection)

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

μPD70335 Block Diagram



Notes:

(1) The μPD70335 (V35 Plus) is not a masked ROM product. Internal ROM is reserved and not accessible.

(2) Shaded blocks are modified from the standard V35.

83YL-6989B

FUNCTIONAL DESCRIPTION

Architectural Enhancements

The following features enable the μPD70335 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

Dual Data Bus. The μPD70335 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 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 μPD70335 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.

Register Set

The μPD70335 CPU has a general-purpose register set compatible with the μPD70108/70116, the μPD70320/70322, and μPD70330/70332 microprocessors. Like the μPD70320/70322 and μPD70330/70332, it also has a set of special function registers for controlling the on-board peripherals. All registers reside in the CPU's memory space. They are grouped in a 512-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.

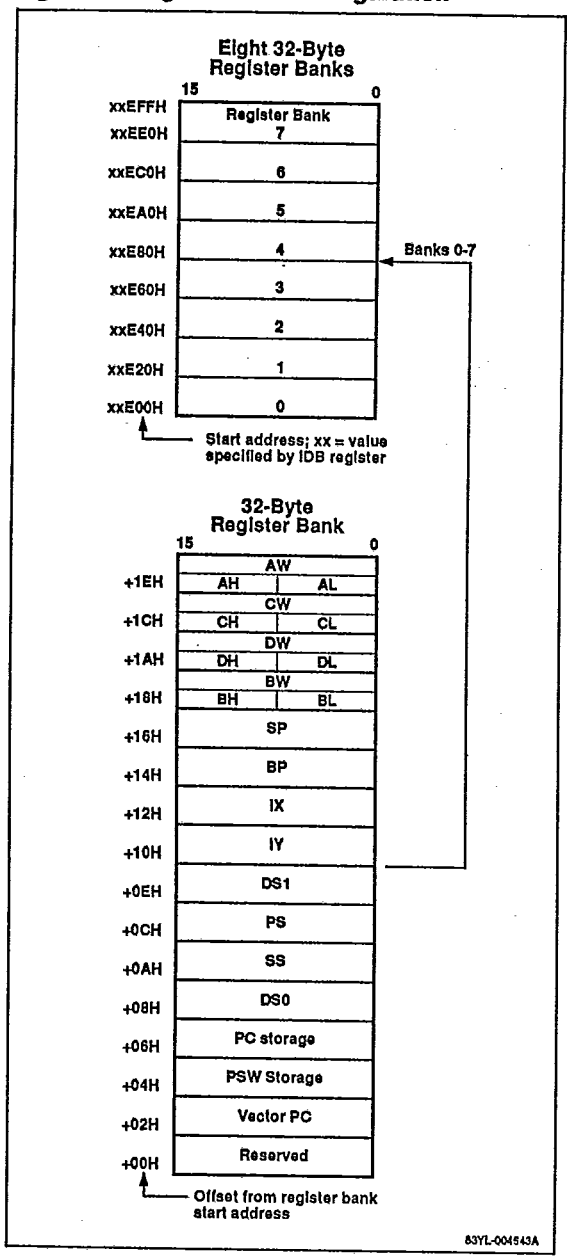
Register Banks. Because the general-purpose register set is in internal RAM, it is possible to have multiple banks of registers. The μPD70335 CPU supports up to 8 register banks. A bit field in the PSW selects which bank is currently being used. Each bank contains the entire CPU register set plus additional 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 upon termination of the context switch.

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.

PSW

15							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	Auxillary 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
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. 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.

Functional Comparison

The μPD70335 (V35 Plus) is built around the same core and contains the same peripherals as the μPD70325 (V25 Plus) as well as the μPD70330 (V35). The primary difference between the V35 and V25 is confined to the external bus interface and bus control logic. While V25 and V25 Plus are designed with an 8-bit external interface, V35 and V35 Plus provide the full 16-bit external data path.

The μPD70335 provides a direct DRAM style bus interface. This interface is obtained by multiplexing the 20 address lines in row/column fashion and also providing a non-multiplexed 16-bit external data bus. The resulting nominal bus cycle is three CLOCKOUT states. During the first bus state, the address lines output the high 9 bits of the physical address: A_9 to A_{17}.

During the second bus state, the address lines output the low address bits: A_1 to A_8. Address lines A_0 and A_{19} are not multiplexed and are valid during the entire bus cycle. The final address line (A18) is multiplexed with the Upper Byte Enable signal (UBE) and is valid as an address during bus state one. During 16-bit transfers to odd addresses (UBE = 0 and A_0 = 1), two bus cycles are performed; each cycle transfers eight bits.

Typically, the MREQ signal is used to generate the DRAM RAS control signal, and the MSTB signal is used to generate the CAS signal. Like the V35, the V35 Plus provides a refresh output from the internal refresh control unit, which is typically gated into the DRAM RAS signal.

As a result of this memory access scheme, the clock cycle counts for instruction execution on the V35 Plus are different from the V25 Plus.

Another V35 Plus difference is the operation of the READY input pin. This pin is sampled in the middle of the second bus cycle (BAW1) on the V25 Plus, whereas the V35 samples one clock period later in the middle of BAW2.

Other than these bus controller differences, the V35 Plus is identical to the V25 Plus in its operation. All internal peripherals are programmed and operate in the same manner as those of the V25 Plus. The instruction sets of the two processors are identical, and internally both processors operate on 16-bit data paths.

INSTRUCTIONS

The μPD70335 instruction set is fully upward 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 μPD70335 does not support the V20 8080 emulation mode. All of the instructions pertaining to this have been deleted from the μPD70335 instruction set.

Enhanced Instructions

In addition to the μPD8086/88 instructions, the μPD70335 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 μPD70335 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
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 μPD70335 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 2 and 3 show bit field insertion and bit field extraction.

4e

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 2. Bit Field Insertion

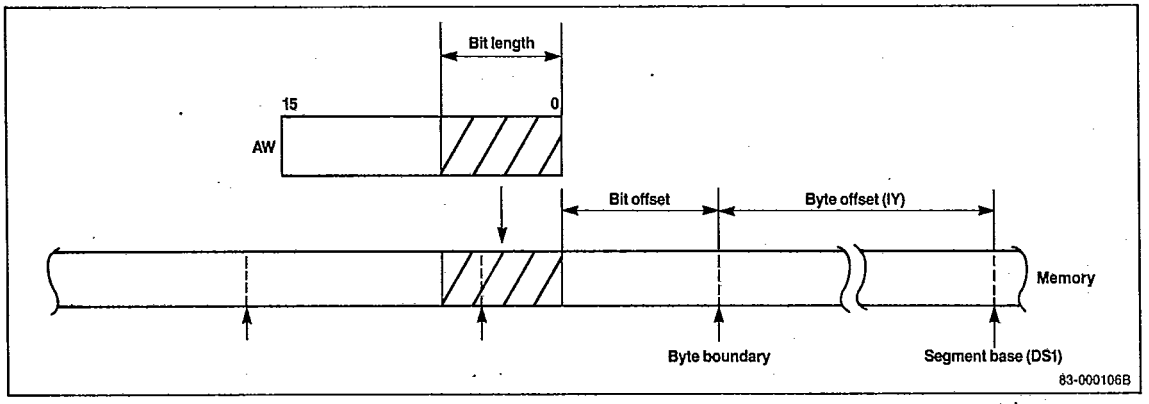
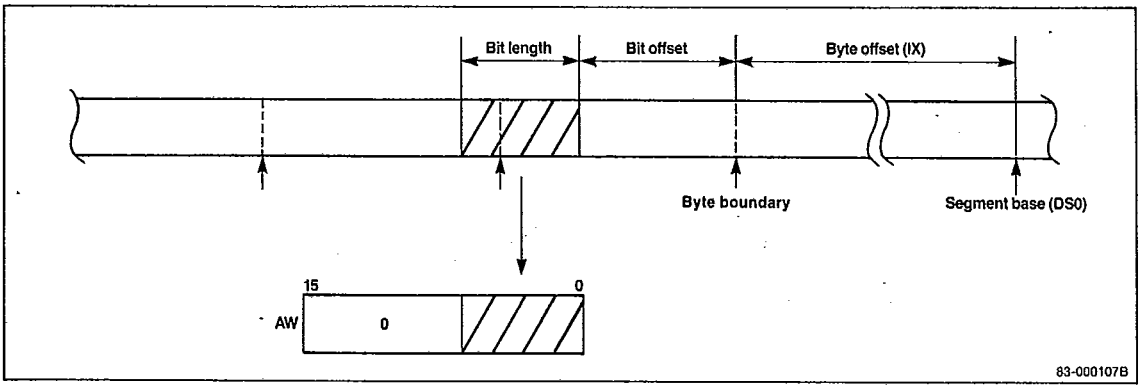


Figure 3. Bit Field Extraction



Bit Manipulation Instructions

The μPD70335 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 μPD70335 has the following four additional instructions.

Instruction	Function
BTCLR sfr. imm3 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 Plus has the following four instructions that allow the effective use of the register banks for software interrupts and multitasking. Also, see figures 7 and 9.

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 μPD70335 can service interrupts generated both by hardware and by software. Software interrupts are serviced through vectored interrupt processing. See table 1 for the various types of software interrupts.

Table 1. 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 μPD70335 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 2. 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 4.

Figure 4. Interrupt Vector

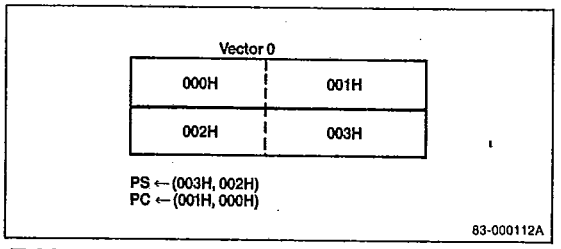


Table 2. Interrupt Vectors

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

4e

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 Plus 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 interrupt sources on the V35 Plus are similar to those on the V35. The 17 interrupt sources (table 3) 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.

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_2	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 maskable 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. ES0-ES2 correspond to INTP0-INTP2, respectively. See figure 5.

Figure 5. External Interrupt Mode Register (INTM)

0	ES2	0	ES1	0	ES0	0	ESNMI
7							0

ES2	INTP2 Input Effective Edge
0	Falling edge
1	Rising edge
ES1	**INTP1 Input Effective Edge**
0	Falling edge
1	Rising edge
ES0	**INTP0 Input Effective Edge**
0	Falling edge
1	Rising edge
ESNMI	**NMI Input Effective Edge**
0	Falling edge
1	Rising edge

Interrupt Factor Register

The primary enhancement of the V35 Plus interrupt control unit is the addition of a special function register that stores the vector type that last caused an interrupt. This IRQS register (figure 5A) stores the vector until the next interrupt request is accepted, but is not changed by response to NMI, INT, or macroservice interrupts.

The main purpose of the IRQS register is to allow several interrupts within a given priority level to be serviced with context switching. Once the interrupt service routine is executing, the cause of the interrupt can be determined only by reading this register, rather than by long and time-consuming software determination. It is recommended that the contents of the IRQS register be read before interrupts are re-enabled to avoid confusion within multiprocessing environments.

Figure 5A. Interrupt Factor Register (IRQS)

0	0	0	Interrupt Vector
7		5	4 0

Interrupt Factor	Interrupt Vector
INTTU0	1CH
INTTU1	IDH
INTTU2	IEH
INTD0	14H
INTD1	15H
INTP0	18H

Interrupt Factor	Interrupt Vector
INTP1	19H
INTP2	1AH
INTSER0	0CH
INTSR0	0DH
INTST0	0EH
INTSER1	10H
INTSR1	11H
INTST1	12H
INTTB	1FH

Table 3. 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	

Table 3. Interrupt Sources (cont)

Interrupt Source	External/ Internal	Vector	Macro Service	Bank Switching	Priority Order: Setting Possible	Priority Order: Between Groups	Priority Order: Within Groups	Multiple Processing Control
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 3, 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 6.

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 reset the highest priority bit set in the ISPR register.

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 7 and 8 show register bank context switching and register bank return. Figure 9 shows software-initiated task switching.

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 6. Interrupt Request Control Registers (IRC)

IF	IMK	MS/INT	ENCS	0	PR_2	PR_1	PR_0
7							0

IF	**Interrupt Flag**
0	No interrupt request generated
1	Interrupt request generated
IMK	**Interrupt Mask**
0	Open (interrupts enabled)
1	Closed (interrupts disabled)
MS/INT	**Interrupt Response Method**
0	Vector interrupt or register bank switching
1	Macroservice function
ENCS	**Register Bank Switching Function**
0	Not used
1	Used
PR_2-PR_0	**Interrupt Group Priority (0-7)**
0 0 0	Highest (0) ↓
1 1 1	Lowest (7)

Figure 7. Register Bank Context Switching

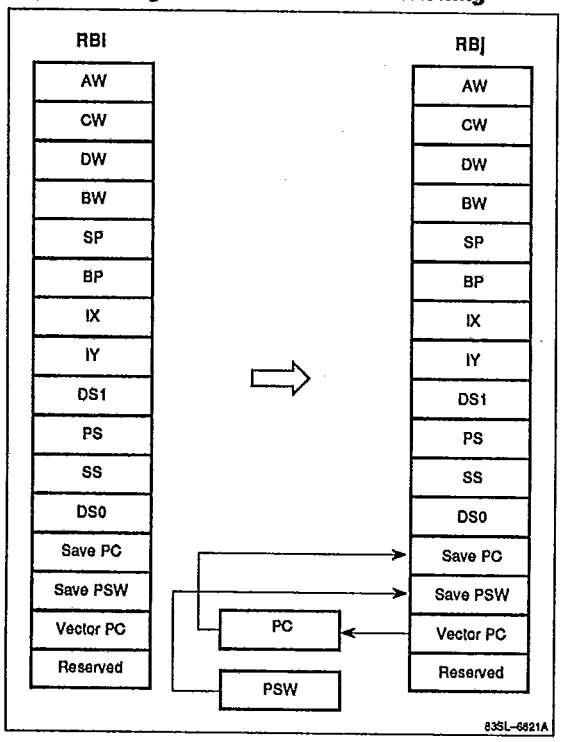


Figure 8. Register Bank Return

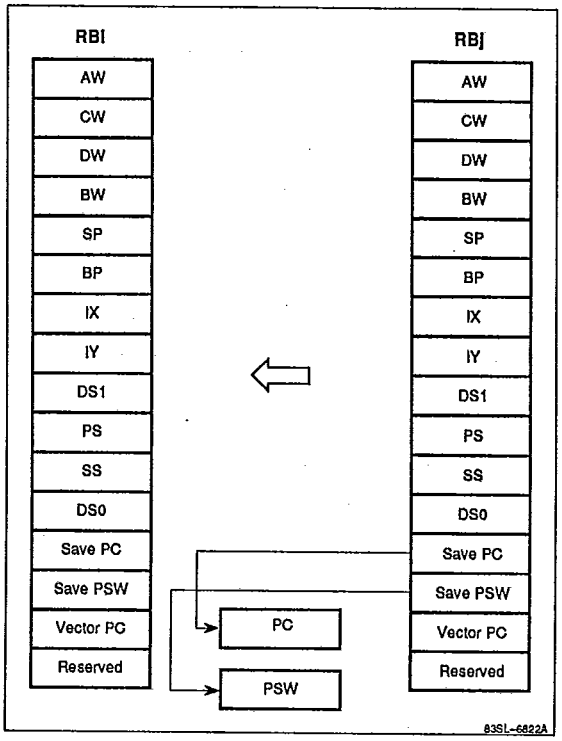
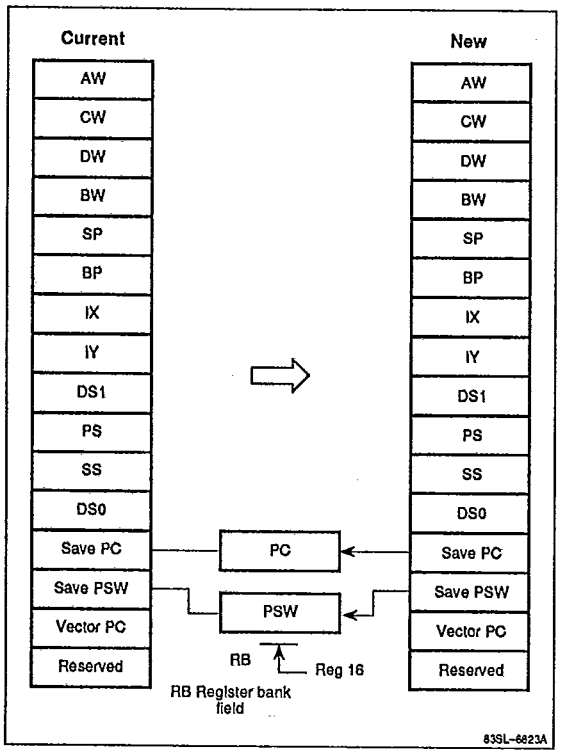


Figure 9. Task Switching



MACROSERVICE FUNCTION

The macroservice function (MSF) is a special microprogram 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 macroservice 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 macroservice counter counts out.

Like the NMI, INT, and INTTB, the two DMA controller interrupts (INTD0, INTD1) and the serial error interrupts (INTSER0, INTSER1) do not have MSF capability.

There are eight, 8-byte macroservice channels mapped into internal RAM from XXE00H to XXE3FH. Figure 10 shows the components of each channel.

Setting the macroservice mode for a given interrupt requires programming the corresponding macroservice control register. Each individual interrupt serviceable with the MSF has its own associated MSC special-function register. The general format for all MSC registers is shown in figure 11.

Figure 10. Macroservice Channels

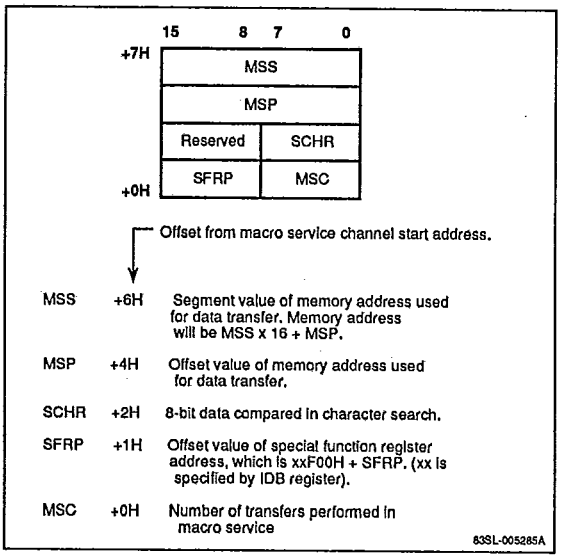


Figure 11. Macroservice Control Registers (MSC)

MSM_2	MSM_1	MSM_0	DIR	0	CH_2	CH_1	CH_0
7							0

MSM_2-MSM_0	Macroservice Mode
0 0 0	Normal (8-bit transfer)
0 0 1	Normal (16-bit transfer)
1 0 0	Character search (8-bit transfer
	Other combinations are not allowed.
DIR	**Data Transfer Direction**
0	Memory to SFR
1	SFR to memory
CH_2-CH_0	**Macroservice Channel**
0 0 0	Channel 0
	↓
1 1 1	Channel 7

TIMER UNIT

The μPD70335 (figure 12) 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 either 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

Timer Control Registers

Setting the desired timer mode requires programming the timer control register. See figures 13 and 14 for format.

Figure 12. Timer Unit Block Diagram

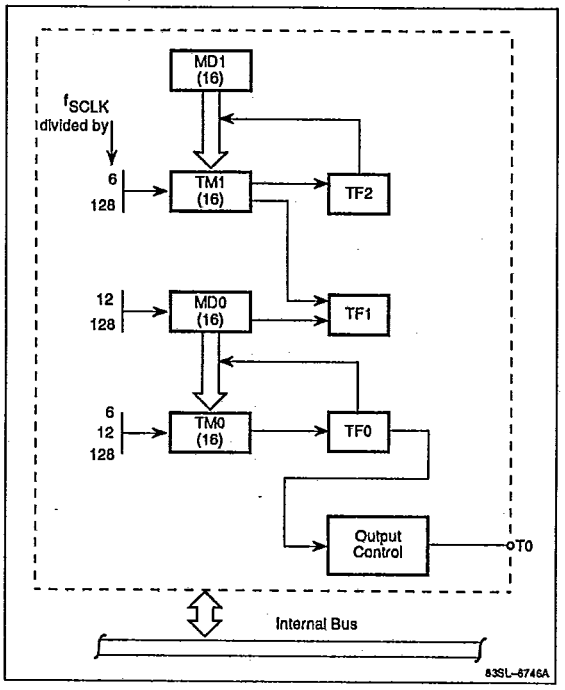


Figure 13. Timer Control Register 0 (TMC0)

TS0	TCLK0	MS0	MCLK0	ENT0	ALV	MOD_1	MOD_0
7							0

TS0	Timer 0 In Either Mode
0	Stop countdown
1	Start countdown

MOD_1	MOD_0	TCLK0	TM0 Register Clock Frequency
0	0	0	$f_{SCLK}/6$ (Interval)
0	0	1	$f_{SCLK}/128$ (Interval)
0	1	0	$f_{SCLK}/12$ (One-shot)
0	1	1	$f_{SCLK}/128$ (One-shot)

MS0	MD0 Register Countdown (One-Shot Mode)
0	Stop
1	Start

MCLK0	MD0 Register Clock Frequency
0	$f_{SCLK}/12$
1	$f_{SCLK}/128$

ENT0	TOUT Square-Wave Output
0	Disable
1	Enable

ALV	TOUT Initial Level (Counter Stopped)
0	Low
1	High

MOD_1	MOD_0	Timer Unit Mode
0	0	Interval timer
0	1	One-shot
1	X	Reserved

Figure 14. Timer Control Register 1 (TMC1)

TS1	TCLK1	0	0	0	0	0	0
7							0

TS1	Timer 1 Countdown
0	Stop
1	Start

TCLK1	Timer 1 Clock Frequency
0	$f_{SCLK}/6$
1	$f_{SCLK}/128$

TIME BASE COUNTER

The 20-bit free-running time base counter (TBC) controls internal timing sequences and is available as the source of periodic interrupts at lengthy intervals. One of four interrupt periods can be selected by programming the TB_0 and TB_1 bits in the processor control register (PRC). The TBC interrupt is unlike the others because it is fixed as a level 7 vectored interrupt. Macroservice and register bank switching cannot be used to service this interrupt. See figures 14A and 14B.

Figure 14A. Time Base Interrupt Request Control Register (TBIC)

TBF	TBMK	0	0	0	1	1	1
7			Address xxFECH				0

TBF	Time Base Interrupt Flag
0	No interrupt generated
1	Interrupt generated

TBMK	Time Base Interrupt Mask
0	Unmasked
1	Masked

Figure 14B. Processor Control Register (PRC)

0	RAMEN	0	0	TB_1	TB_0	PCK_1	PCK_0
7			Address xxFEBH				0

RAMEN	Built-In RAM
0	Disable
1	Enable

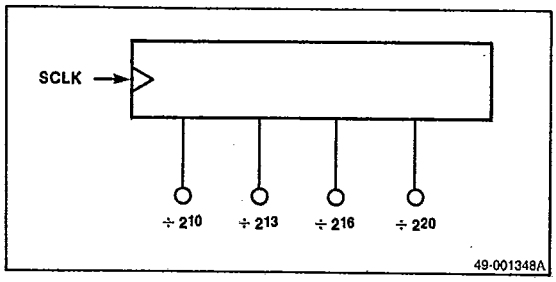
TB_1	TB_0	Time Base Interrupt Period
0	0	$2^{10}/f_{SCLK}$
0	1	$2^{13}/f_{SCLK}$
1	0	$2^{16}/f_{SCLK}$
1	1	$2^{20}/f_{SCLK}$

PCK_1	PCK_0	System Clock Frequency (f_{SCLK})
0	0	$f_{OSC}/2$
0	1	$f_{OSC}/4$
1	0	$f_{OSC}/8$
1	1	Reserved

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 14C) uses the system clock as the input frequency. The system clock can be changed by programming the PCK_0 and PCK_1 bits in the processor control register (PRC). Reset initializes the system clock to $f_{OSC}/8$ (f_{OSC} = external oscillator frequency).

Figure 14C. Time Base Counter (TBC) Block Diagram



REFRESH CONTROLLER

The μPD70335 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 zeros. The 9-bit refresh address is automatically incremented at every refresh timing for 512 row addresses. The 8-bit refresh mode (RFM) register (figure 15) specifies the refresh operation and allows refresh during both CPU HALT and HOLD modes. Refresh cycles are automatically timed to minimize the effect on system throughput.

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

It should be noted that since the V35 Plus directly supports dynamic RAM memory, the refresh controller output should be gated into the RAS input of the memory chips. When combined with the chip select logic and the MREQ signals, a direct DRAM interface is supported.

SERIAL CONTROL UNIT

The serial unit of the μPD70335 is functionally identical to that of the standard V35, with the exception of several enhanced features.

All serial status information is moved to the Serial Status Register (SSTn) on the V35 Plus. Included in this register is an additional flag which signals that the transmit shift register is clear of data. This flag allows software to poll for the completion of a message (the last bit of the last byte is shifted out when the ALL SENT bit is set). All error flags are available in this register (refer to figure 16).

Please refer to the μPD70330 (V35) data sheet for additional information on the serial channels.

Figure 15. Refresh Mode Register (RFM)

RFLV	HLDRF	HLTRF	RFEN	RFW_1	RFW_0	RFT_1	RFT_0
7			Address xxFE1H				0

RFLV	RFEN	$\overline{\text{REFRQ}}$ Output Signal Level
0	0	0
1	0	1
0	1	0
1	1	Refresh pulse
HLDRF		**Automatic Refresh Cycle in HOLD Mode**
0		Disabled
1		Enabled
HLTRF		**Automatic Refresh Cycle in HALT Mode**
0		Disabled
1		Enabled
RFEN		**Automatic Refresh Cycle**
0		Refresh pin = RFLV
1		Refresh enabled
RFW_1	RFW_0	**No. of Wait States Inserted in Refresh Cycle**
0	0	0
0	1	1
1	0	2
1	1	2
RFT_1	RFT_0	**Refresh Period**
0	0	16/SCLK
0	1	32/SCLK
1	0	64/SCLK
1	1	128/SCLK

4E

Figure 16. Serial Status Register (SSTn)

RxDN	ASn	TxBEn	RxBFn	0	ERPn	ERFn	EROn
7							0

Receive Terminal Pin State	RxDN
Input state of RxD_n pin is checked by RxDN bit	

All Sent Flag	ASn
Reset when transmit data has been written to transmit shifter	0
Set when all the data in the transmit buffer and transmit shift register has been sent (Note 1)	1

Transmit Buffer Empty Flag	TxBEn
Reset when transmit data has been written to transmit buffer (Note 1)	0
Set when transmit data in transmit buffer has been sent to shift register	1

Receive Buffer Full Flag	RxBFn
Reset when receive data has been read from receive buffer (Note 2)	0
Set when receive data has been sent from shift register to receive buffer (Note 3)	1

Parity Error Flag	ERPn
Indicates that transmit parity was not consistent with receive parity (Note 5)	

Framing Error Flag	ERFn
Indicates that stop bit was not detected (Note 5)	

Overrun Error Flag	EROn
Indicates that succeeding receive has completed before the previous receive data is taken over from the receive buffer (Note 5)	

Notes:

(1) Transmitter flags are reset to 1 when the value of either the band rate generator or serial control register is written.

(2) Receive buffer full flag is also reset when either the band rate generator or serial control register is written.

(3) Receive buffer full flag is not related to the receive error state.

(4) Error flags are cleared when the next data byte is received.

(5) In the table, n = 0 or 1.

Table 4. DMA Controller Operation

	Single-Step Mode	Burst Mode	Single-Transfer Mode	Demand Release Mode
Transmission coverage	Memory - memory	Memory - memory	Memory - I/O	Memory - I/O
Function	Under one time of DMA request instruction, one bus cycle and one DMA transmission are alternately executed the specified number of times.	Under one DMA request, specified number of DMA transmissions are executed.	One DMA transfer is executed every time DMA request occurs.	DMA transmission is executed while DMARQ terminal is kept high-level.
DMA start	Rise of DMARQ Setting TDMA bit of DMA control register	Rise of DMARQ Setting TDMA bit of DMA control register	Rise of DMARQ	High level of DMARQ
Halt method	Depends on software Terminal count decremented from zero	None Terminal count decremented from zero	Depends on software Terminal count decremented from zero	Halted at low level of DMARQ during DMA transmission Terminal count decremented from zero
Interrupt	All accepted	Not accepted during DMA transmission	All accepted	All accepted except during DMA transmission
During halt	Specified times of DMA transmission are executed consecutively	Specified number of DMA transfers are executed consecutively	Active	Active
DMA request during DMA transmission	DMA at channel 1 is retained while DMA at channel 0 is executed	Other DMA is retained until DMA transmission is terminated.	DMA transmission under request is executed after one DMA transmission is over	DMA at channel 1 is retained while DMA at channel 0 is executed.

DMA CONTROLLER

Two memory-to-memory transfer modes (single-step and burst) are supported as well as two I/O-to-memory modes (single-transfer and demand release). Refer to table 4.

The most significant V35 Plus enhancement boosts the transfer rates of the dual internal DMA channels to full bus bandwidth. All operational modes remain the same as the V35, but since the V35 Plus DMA controller is implemented in hard-wired logic, the control delays of a microprogrammed method are not present. As a result, the demand release mode transfer rate boasts a theoretical transfer rate of over 6M bytes per second.

The μPD70335 DMA control registers are moved from the internal RAM to the SFR area; thus the V35 Plus may effectively have a larger internal RAM memory area than comparable designs on the standard V35.

Additionally, the μPD70335 DMA controller uses linear registers for both source and destination address pointers. Thus, three 8-bit registers completely specify the DMA address pointers as shown in figure 17. These pointers may be updated by byte (±1) or word (±2) quantities as programmed in the DMA channel mode register shown in figure 18. This register also specifies the operational mode of the channel. The EDMA bit is automatically cleared when terminal count is reached, and DMA requests are ignored when this bit is cleared.

Figure 17. DMA Address Registers

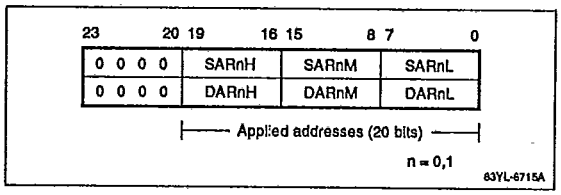


Figure 18. DMA Channel Mode Registers (DMAMn)

MD_2	MD_1	MD_0	W	EDMA	TDMA	0	0
7							0

MD_2-MD_0	Transfer Mode
000	Single-step (memory to memory)
001	Demand release (I/O to memory)
010	Demand release (memory to I/O)
011	Disabled
100	Burst (memory to memory)
101	Single-transfer (I/O to memory)
110	Single-transfer (memory to I/O)
111	Disabled

W	Transfer Method
0	Byte transfer
1	Word transfer

EDMA	TDMA	Transfer Condition
0	0	Disabled
1	0	DMA channel enabled
1	1	Software initiate DMA (memory to memory modes)

The TDMA bit is only valid for single-step and burst modes. This bit allows software initiation of the DMA transfer (provided the EDMA bit is set); the bit always reads as zero and has no meaning in the demand-release or single-transfer modes.

The DMA address pointers may be incremented or decremented per transfer as specified in the DMA address update register shown in figure 19. The address pointer can also be programmed to remain the same, allowing repeated transfers to or from a location.

Figure 19. DMA Address Control Registers (DMAC)

0	0	PD_1	PD_0	0	0	PS_1	PS_0
7							0

PD_1-PD_0	Destination Address Offset
00	No modification
01	Increment
10	Decrement
11	No modification

PS_1-PS_0	Source Address Offset
00	No modification
01	Increment
10	Decrement
11	No modification

The DMAAKn signals are not output for memory-to-memory transfer modes, but are driven low for each transfer I/O to/from memory. Nominal DMA bus cycles are three clock states; however, programmable wait

states may be added. Wait states for memory-to-memory transfers are added to both source and destination addresses as programmed for each specific address.

During memory-to-I/O transfers, the number of wait states inserted is determined by the slower of the source and destination. I/O-to-memory transfers add the number of wait states required by the memory write address.

PARALLEL I/O PORTS

The μPD70335 has three 8-bit parallel I/O ports: P0, P1, and P2. Refer to figures 20 through 24. 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 pin) x n/16, where n = 1 to 16. See figure 25.

Figure 20. Port Mode Registers 0 and 2 (PM0, PM2)

PM_7	PM_6	PM_5	PM_4	PM_3	PM_2	PM_1	PM_0
7							0

PM_n	Input or Output Bit Selection
0	Output port mode
1	Input port mode

n = 7 through 0

Figure 21. Port Mode Register 1 (PM1)

$PM1_7$	$PM1_6$	$PM1_5$	$PM1_4$	1	1	1	1
7							0

$PMC1_n$	$PM1_n$	Port Mode Input/Output (Port P1n)
0	0	Output port mode
0	1	Input port mode

n = 7, 6, 5, or 4.

Figure 22. Port Mode Control Register 0 (PMC0)

$PMC0_7$	–	–	–	–	–	–	–
7							0

$PMC0_7$	Port or Control Bit Selection
0	Port mode
1	CLKOUT

Figure 23. Port Mode Control Register 1 (PMC1)

$PMC1_7$	$PMC1_6$	$PMC1_5$	$PMC1_4$	$PMC1_3$	$PMC1_2$	$PMC1_1$	$PMC1_0$
7							0

$PMC1_7$	Port/Control Bit Selection
0	$P1_7$ I/O
1	READY Input
$PMC1_6$	**Port/Control Bit Selection**
0	$P1_6$ I/O
1	$\overline{SCKO}$ output
$PMC1_5$	**Port/Control Bit Selection**
0	$P1_5$ I/O
1	TOUT output
$PMC1_4$	**Port/Control Bit Selection**
0	$P1_4$ I/O or $\overline{POLL}$ input
1	INT Input
$PMC1_3$	**Port/Control Bit Selection**
0	INTP2/$P1_3$ Input
1	$\overline{INTAK}$ output
$PMC1_2$	**Port/Control Bit Selection**
x	INTP1/$P1_2$ Input
0	INTP2/$P1_3$ Input
$PMC1_1$	**Port/Control Bit Selection**
x	INTP0/$P1_1$ Input
$PMC1_0$	**Port/Control Bit Selection**
x	NMI/$P1_0$ Input

Figure 24. Port Mode Control Register 2 (PMC2)

$PMC2_7$	$PMC2_6$	$PMC2_5$	$PMC2_4$	$PMC2_3$	$PMC2_2$	$PMC2_1$	$PMC2_0$
7							0

$PMC2_7$	Port/Control Bit Selection
0	I/O port
1	HLDRQ output
$PMC2_6$	**Port/Control Bit Selection**
0	I/O port
1	$\overline{HLDAK}$ input
$PMC2_5$	**Port/Control Bit Selection**
0	I/O port
1	$\overline{TC1}$ output
$PMC2_4$	**Port/Control Bit Selection**
0	I/O port
1	$\overline{DMAAK1}$ output
$PMC2_3$	**Port/Control Bit Selection**
0	I/O port
1	$\overline{DMARQ1}$ input
$PMC2_2$	**Port/Control Bit Selection**
x	I/O port
0	TC0 output
$PMC2_1$	**Port/Control Bit Selection**
x	I/O port
1	$\overline{DMAAK0}$ output
$PMC2_0$	**Port/Control Bit Selection**
0	I/O port
1	DMARQ0 input

Figure 25. Port T Mode Register (PMT)

0	0	0	0	PMT_3	PMT_2	PMT_1	PMT_0
7							0

PMT_3-PMT_0	V_{REF}
0000	V_{TH} x 16/16
0001	V_{TH} x 1/16
0010	V_{TH} x 2/16
0011	V_{TH} x 3/16
0100	V_{TH} x 4/16
0101	V_{TH} x 5/16
0110	V_{TH} x 6/16
0111	V_{TH} x 7/16
1000	V_{TH} x 8/16
1001	V_{TH} x 9/16
1010	V_{TH} x 10/16
1011	V_{TH} x 11/16
1100	V_{TH} x 12/16
1101	V_{TH} x 13/16
1110	V_{TH} x 14/16
1111	V_{TH} x 15/16

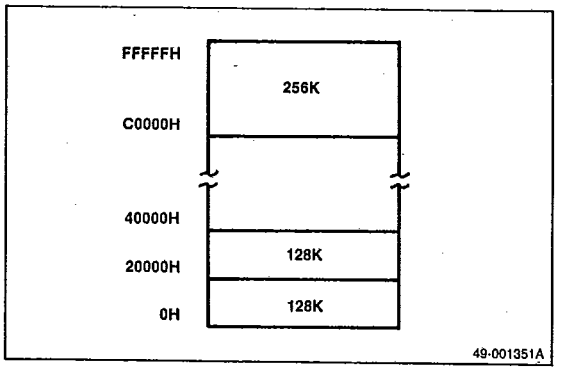
PROGRAMMABLE WAIT STATE GENERATION

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 external 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 26 shows the memory map for programmable wait state generation; see figure 27 for a graphic representation of the wait control word.

Figure 26. Programmable Wait State Generation



STANDBY MODES

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

HALT Mode

In the HALT mode, the CPU 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 serviced and the HALT state released. In the DI state, program execution is restarted with the instruction following the HALT instruction and the interrupt causing the release from HALT will be latched.

STOP Mode

The STOP mode allows the largest power reduction while maintaining RAM. The oscillator is stopped, halting the CPU and all internal peripherals. Internal status and port pin outputs are maintained. Only a RESET or NMI can release this mode.

A standby flag in the STBC register is reset by rises in the supply voltage. Its status is maintained during normal operation and standby. The STBC register (figure 28) 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 5 shows the special-function register mnemonic, type, address, reset value, and function. The eight high-order bits of each address (xx) are specified by the IDB register.

SFR area addresses not listed in table 5 are reserved. If read, the contents of these addresses are undefined, and any write operation will be meaningless.

Figure 27. Wait Control Word (WTC)

IO1	IO0	Block 61	Block 60	Block 51	Block 50	Block 41	Block 40
7			Wait Control, High				0

Block 31	Block 30	Block 21	Block 20	Block 11	Block 10	Block 01	Block 00
7			Wait Control, Low				0

Wait States	Block n1	Block n0
0	0	0
1	0	1
2	1	0
2 or more (control from READY pin)	1	1

n = 0 thru 6

Figure 28. Standby Register (STBC)

0	0	0	0	0	0	0	SBF
7							0

SBF	Standby Flag
0	No changes in V_{DD} (standby)
1	Rising edge on V_{DD} (cold start)

Table 5. Special-Function Registers

Address	Register Function	Symbol	R/W	Manipulation (Note 6)	When Reset
xxF00H	Port 0	P0	R/W	8/1	Undefined
xxF01H	Port mode 0	PM0	W	8	0FFH
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	0FFH
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	0FFH
xxF12H	Port mode control 2	PMC2	W	8	00H
xxF38H	Threshold port	PT	R	8	Undefined
xxF3BH	Threshold port mode	PMT	R/W	8/1	00H
xxF40H	External interrupt mode	INTM	R/W	8/1	00H
xxF44H	External interrupt macro service control 0 (Note 1)	EMS0	R/W	8/1	Undefined
xxF45H	External interrupt macro service control 1 (Note 1)	EMS1	R/W	8/1	
xxF46H	External interrupt macro service control 2 (Note 1)	EMS2	R/W	8/1	
xxF4CH	External interrupt request control 0 (Note 1)	EXIC0	R/W	8/1	47H
xxF4DH	External interrupt request control 1 (Note 1)	EXIC1	R/W	8/1	
xxF4EH	External interrupt request control 2 (Note 1)	EXIC2	R/W	8/1	

Table 5. Special-Function Registers (cont)

Address	Register Function	Symbol	R/W	Manipulation (Note 6)	When Reset
xxF60H	Receive buffer 0	RxB0	R	8	Undefined
xxF62H	Transmit buffer 0	TxB0	W	8	
xxF65H	Serial receive macro service control 0 (Note 1)	SRMS0	R/W	8/1	
xxF66H	Serial transmit macro service control 0 (Note 1)	STMS0	R/W	8/1	
xxF68H	Serial mode register 0	SCM0	R/W	8/1	00H
xxF69H	Serial control register 0	SCC0	R/W	8/1	
xxF6AH	Baud rate generator 0	BRG0	R/W	8/1	
xxF6BH	Serial status register 0	SST0	R	8	60H
xxF6CH	Serial error interrupt request register 0 (Note 1)	SEIC0	R/W	8/1	47H
xxF6DH	Serial receive interrupt request register 0 (Note 1)	SRIC0	R/W	8/1	
xxF6EH	Serial transmit interrupt request register 0 (Note 1)	STIC0	R/W	8/1	
xxF70H	Serial receive buffer 1	RxB1	R	8	Undefined
xxF72H	Serial transmit buffer 1	TxB1	W	8	
xxF75H	Serial receive macro service register 1 (Note 1)	SRMS1	R/W	8/1.	
xxF76H	Serial transmit macro service register 1 (Note 1)	STMS1	R/W	8/1	
xxF78H	Serial communication register 1	SCM1	R/W	8/1	00H
xxF79H	Serial control register 1	SCC1	R/W	8/1	
xxF7AH	Baud rate generator 1	BRG1	R/W	8/1	
xxF7BH	Serial status register 1	SCS1	R	8	60H
xxF7CH	Serial error interrupt request register I (Note 1)	SEIC1	R/W	8/1	47H
xxF7DH	Serial receive interrupt request register 1 (Note 1)	SRIC1	R/W	8/1	
xxF7EH	Serial transmit interrupt request register 1 (Note 1)	STIC1	R/W	8/1	
xxF80H	Timer register 0 (Note 2)	TM0	R/W	16	Undefined
xxF82H	Timer 0 modulo register (Note 2)	MD0	R/W	16	
xxF88H	Timer register 1 (Note 2)	TM1	R/W	16	
xxF8AH	Timer 1 modulo register (Note 2)	MD1	R/W	16	
xxF90H	Timer 0 control register (Note 2)	TMC0	R/W	8/1	00H
xxF91H	Timer 1 control register (Note 2)	TMC1	R/W	8/1	
xxF94H	Timer unit 0 macro service register (Note 1)	TMMS0	R/W	8/1	Undefined
xxF95H	Timer unit 1 macro service register (Note 1)	TMMS1	R/W	8/1	
xxF96H	Timer unit 2 macro service register (Note 1)	TMMS2	R/W	8/1	
xxF9CH	Timer unit 0 interrupt request register (Note 1)	TMIC0	R/W	8/1	47H
xxF9DH	Timer unit 1 interrupt request register (Note 1)	TMIC1	R/W	8/1	
xxF9EH	Timer unit 2 interrupt request register (Note 1)	TMIC2	R/W	8/1	
xxFA0H	DMA address update control register 0	DMAC0	R/W	8/1	Undefined
xxFA1H	DMA mode register 0	DMAM0	R/W	8/1	47H
xxFA2H	DMA address update control register 1	DMAC1	R/W	8/1	Undefined
xxFA3H	DMA mode register 1	DMAM1	R/W	8/1	00H
xxFACH	DMA interrupt request control register 0 (Note 1)	DIC0	R/W	8/1	47H
xxFADH	DMA interrupt request control register 1 (Note 1)	DIC1	R/W	8/1	

Table 5. Special-Function Registers (cont)

Address	Register Function	Symbol	R/W	Manipulation (Note 6)	When Reset
xxFC0H	DMA channel 0 source address pointer low	SAE0L	R/W	16/8	Undefined
xxFC1H	DMA channel 0 source address pointer mid	SAE0M	R/W	16/8	
xxFC0H	DMA Channel 0 source address pointer low	SAE0L	R/W	16/8	
xxFC1H	DMA channel 0 source address pointer mid	SAE0M	R/W	16/8	
xxFC2H	DMA channel 0 source address pointer high	SAR0H	R/W	8	
xxFC4H	DMA channel 0 destination address pointer low	DAR0L	R/W	16/8	
xxFC5H	DMA channel 0 destination address pointer mid	DAR0M	R/W	16/8	
xxFC6H	DMA channel 0 destination address pointer high	DAR0H	R/W	8	
xxFC8H	DMA channel 0 count register	DMATC0	R/W	16/8	
xxFD0H	DMA channel 1 source address pointer low	SAR1L	R/W	16/8	
xxFD1H	DMA channel 1 source address pointer mid	SAR1M	R/W	16/8	
xxFD2H	DMA channel 1 source address pointer high	SAR1H	R/W	8	
xxFD4H	DMA channel 1 destination address pointer low	DAR1L	R/W	16/8	
xxFD5H	DMA channel 1 destination address pointer mid	DAR1M	R/W	16/8	
xxFD6H	DMA channel 1 destination address pointer high	DAR1H	R/W	8	
xxFD8H	DMA channel 1 terminal count register	DMATC1	R/W	16/8	
xxFE0H	Standby control register	STBC	R/W (Note 3)	8/1	Undefined (Note 4)
xxFE1H	Refresh mode register	RFM	R/W	8/1	0FCH
xxFE8H	Wait state control	WTC	R/W	16/8	0FFFFH
xxFEAH	User flag (Note 5)	FLAG	R/W	8/1	00H
xxFEBH	Processor control register	PRC	R/W	8/1	4EH
xxFECH	Time base interrupt request control register (Note 1)	TBIC	R/W	8/1	47H
xxFEFH	Interrupt factor register (Note 1)	IRQS	R	8	Undefined
xxFFCH	Interrupt priority control register (Note 1)	ISPR	R	8	00H
xxFFFH	Internal data area base	IDB	R/W	8/1	0FFH

Notes:

(1) One wait state is inserted into accesses to these registers.

(2) A maximum of 6 wait states are added into accesses to these registers.

(3) 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).

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

(5) 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.

(6) The manipulation column indicates which memory operations can read or modify the register according to the following key.

16 Work operations
8 Byte operations
1 Bit operations

ELECTRICAL SPECIFICATIONS

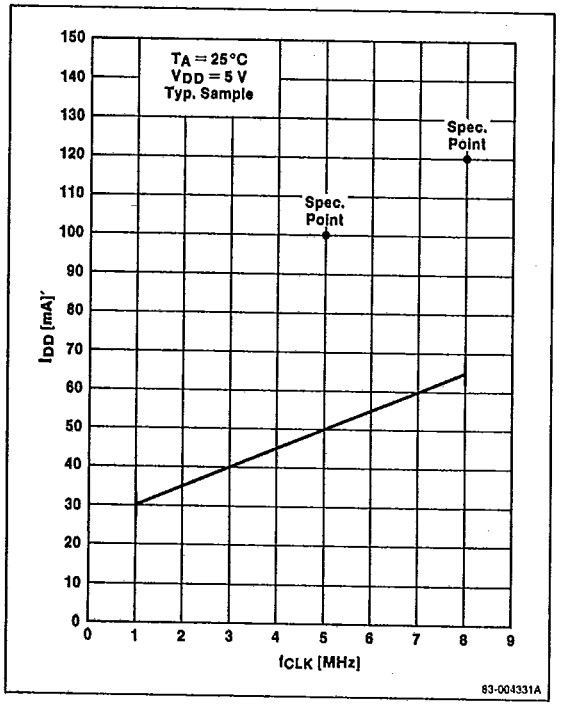
Absolute Maximum Ratings

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}	−10 to +70 °C
Storage temperature range, T_{STG}	−65 to +150 °C

Exposure to Absolute Maximum Ratings for extended periods may affect device reliability; exceeding the ratings could cause permanent damage.

Supply Current vs Clock Frequency



Comparator Characteristics

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

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

V_{DD} = 0 V; T_A = 25 °C

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)

Parameter	Symbol	Min	Typ	Max	Unit	Conditions
Supply current, operating	I_{DD1}		65	120	mA	
Supply current, HALT mode	I_{DD2}		25	50	mA	
Supply current, STOP mode	I_{DD3}		10	30	μA	
V_{TH} supply 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 $\overline{RESET}$, $P1_0$/NMI, X1, X2
	V_{IH2}	0.8 x V_{DD}		V_{DD}	V	$\overline{RESET}$, $P1_0$/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}

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)

Parameter	Symbol	Min	Max	Unit	Conditions
Input rise, fall time	t_{IR}, t_{IF}		20	ns	Except X1, X2, RESET, NMI
Input rise, fall time	t_{IRS}, t_{IFS}		30	ns	RESET, NMI (Schmitt)
Output rise, fall time	t_{OR}, t_{OF}		20	ns	Except CLKOUT
X1 cycle time	t_{CYX}	62	250	ns	
X1 width, low	t_{WXL}	20		ns	
X1 width, high	t_{WXH}	20		ns	
X1 rise, fall time	t_{XR}, t_{XF}		20	ns	
CLKOUT cycle time	t_{CYK}	125	2000	ns	
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}	15	90	ns	
Address valid to input data valid	t_{DADR}		T(n + 1.5) − 70	ns	Note 2
$\overline{MREQ}$ to address hold time	t_{HMRA}	0.5T − 30		ns	
$\overline{MREQ}$ to data delay	t_{DMRD}		T(n + 2) − 60	ns	
$\overline{MSTB}$ to data delay	t_{DMSD}		T(n + 1) − 60	ns	
$\overline{MREQ}$ to $\overline{MSTB}$ delay	t_{DMRMSR}	T − 35	T + 35	ns	
$\overline{MREQ}$ width, low	t_{WMRL}	T(n + 2) − 30		ns	
$\overline{MREQ}$, $\overline{MSTB}$ to address hold time	t_{HMA}	0.5T − 30		ns	
Input data hold time	t_{HMD}	0		ns	
Next control setup time	t_{SCC}	T − 25		ns	
$\overline{MREQ}$ to $\overline{TC}$ delay time	t_{DMRTC}		0.5T + 50	ns	
$\overline{MREQ}$ delay time	t_{DAMR}	0.5T − 30		ns	
$\overline{MSTB}$ read delay time	t_{DAMSR}	0.5 T − 30		ns	
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	
$\overline{MSTB}$ write delay time	t_{DAMSW}	T(n + 0.5) − 30		ns	
$\overline{MREQ}$ to $\overline{MSTB}$ write delay time	t_{DMRMSW}	T(n + 1) − 35	T(n + 1) + 35	ns	
$\overline{MSTB}$ write width low	t_{WMSLW}	T − 30		ns	
Data output hold time	t_{HMDW}	0.5T − 30		ns	
$\overline{IOSTB}$ delay time	t_{DAIS}	0.5T − 30		ns	
$\overline{IOSTB}$ to data input	t_{DISD}		T(n + 1) − 60	ns	
$\overline{IOSTB}$ width, low	t_{WISL}	T(n + 1) − 30		ns	
$\overline{MREQ}$ to $\overline{IOSTB}$ delay time	t_{DMRIS}	T − 35		ns	
Next DMARQ setup time	t_{SDADQ}		T − 50	ns	Demand mode
DMARQ hold time	t_{HDARQ}	0		ns	
$\overline{DMAAK}$ read width, low	t_{WDMRL}	T(n + 2.5) −30		ns	
$\overline{DMAAK}$ write width, low	t_{WDMWL}	T(n + 2) − 30		ns	
$\overline{DMAAK}$ to $\overline{TC}$ delay time	t_{DDATC}		0.5T + 50	ns	

AC Characteristics (cont)

Parameter	Symbol	Min	Max	Unit	Conditions
$\overline{TC}$ width, low	t_{WTCL}	(n + 2)T – 30		ns	
$\overline{REFRQ}$ delay time	t_{DARF}	0.5T – 30		ns	
$\overline{REFRQ}$ width, low	t_{WRFL}	T(n + 2) – 30		ns	
Address hold time	t_{HRFA}	0.5T – 30		ns	
$\overline{RESET}$ width low	t_{WRSL1}	30		ms	Crystal oscillator; STOP/ Power on reset
	t_{WRSL2}	5		μs	System warm reset
$\overline{MREQ}$, $\overline{IOSTB}$ to $\overline{READY}$ setup time	t_{SCRY}		T(n) – 100	ns	n ≥ 2
$\overline{MREQ}$, $\overline{IOSTB}$ to $\overline{READY}$ hold time	t_{HCRY}	T(n)		ns	n ≥ 2
HLDRQ setup time	t_{SHQK}	30		ns	
$\overline{HLDAK}$ output delay time	t_{DKHA}	15	80	ns	
Bus control float to $\overline{HLDAK}$ ↓	t_{CFHA}	T – 50		ns	
$\overline{HLDAK}$ ↑ to control output time	t_{DHAC}	T – 50		ns	
HLDRQ to $\overline{HLDAK}$ delay	t_{DHQHA}		3T + 160	ns	
HLDRQ ↓ to control float time	t_{DHQC}	3T + 30		ns	
HLDRQ width, low	t_{WHQL}	1.5T		ns	
$\overline{HLDAK}$ width, low	t_{WHAL}	T		ns	
$\overline{INTP}$, DMARQ setup	t_{SIQK}	30		ns	
$\overline{INTP}$, DMARQ width, high	t_{WIQH}	8T		ns	
$\overline{INTP}$, DMARQ width, low	t_{WIQL}	8T		ns	
$\overline{POLL}$ setup time	t_{SPLK}	30		ns	
NMI width, high	t_{WNIH}	5		μs	
NMI width, low	t_{WNIL}	5		μs	
$\overline{CTS}$ width, low	t_{WCTL}	2T		ns	
INT setup time	t_{SIRK}	30		ns	
$\overline{INTAK}$ delay time	t_{DKIA}	15	80	ns	
$\overline{INT}$ hold time	t_{HIAIQ}	0		ns	
$\overline{INTAK}$ width, low	t_{WIAL}	2T – 30		ns	
$\overline{INTAK}$ width, high	t_{WIAH}	T – 30		ns	
$\overline{INTAK}$ to data delay time	t_{DIAD}		2T – 130	ns	
$\overline{INTAK}$ to data hold time	t_{HIAD}	0	0.5T	ns	
$\overline{SCKO}$ (TSCK) cycle time	t_{CYTK}	1000		ns	
$\overline{SCKO}$ (TSCK) width, high	t_{WSTH}	450		ns	
$\overline{SCKO}$ (TSCK) width, low	t_{WSTL}	450		ns	
TxD delay time	t_{DTKD}		210	ns	
TxD hold time	t_{HTKD}	20		ns	
$\overline{CTS0}$ (RSCK) cycle time	t_{CYRK}	1000		ns	
$\overline{CTS0}$ (RSCK) width, high	t_{WSRH}	420		ns	

4e

AC Characteristics (cont)

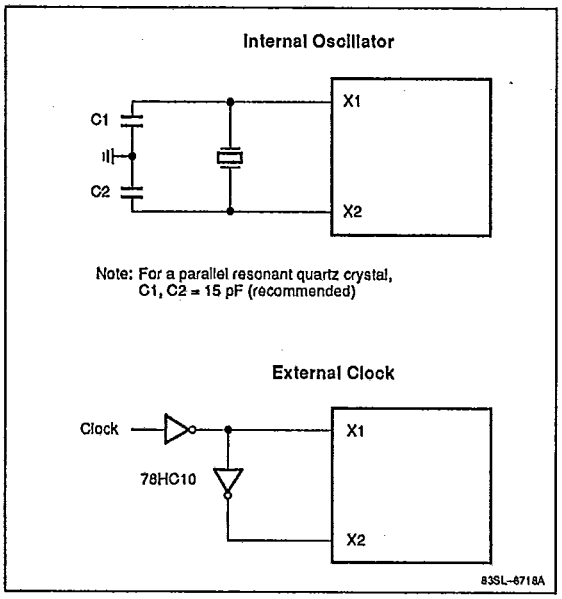
Parameter	Symbol	Min	Max	Unit	Conditions
$\overline{CTS0}$ (RSCK) width, low	t_{WSRL}	420		ns	
RxD setup time	t_{SRDK}	80		ns	
RxD hold time	t_{HKRD}	80		ns	

Notes:

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

(2) n = number of wait states inserted.

External System Clock Control Source



Recommended Oscillator Components

Ceramic Resonator		Capacitors	
Manufacturer	Product No.	C1 (pF)	C2 (pF)
Kyocera	KBR-10.0M	33	33
Murata Mfg.	CSA.10.0MT	47	47
	CSA16.0MX040	30	30
TDK	FCR10.M2S	30	30
	FCR16.0M2S	15	6

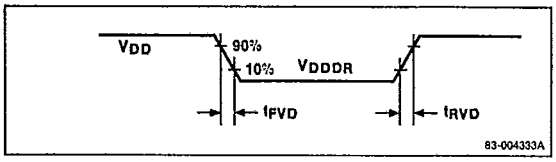
STOP Mode Data Retention Characteristics

T_A = −10 to +70°C

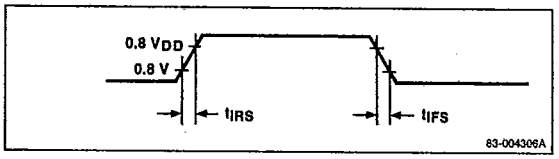
Parameter	Symbol	Min	Max	Unit
Data retention voltage	V_{DDDR}	2.4	5.5	V
V_{DD} rise time	t_{RVD}	200		μs
V_{DD} fall time	t_{FVD}	200		μs

Timing Waveforms

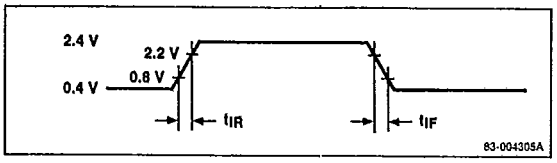
Stop Mode Data Retention Timing



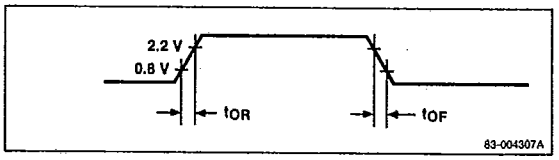
AC Input 2 (RESET, NMI)



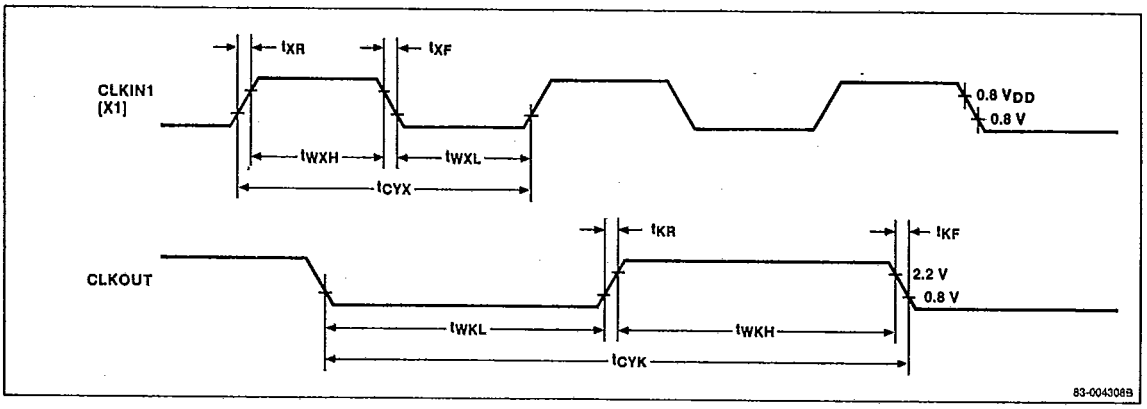
AC Input 1 (Except X1, X2, RESET, NMI)



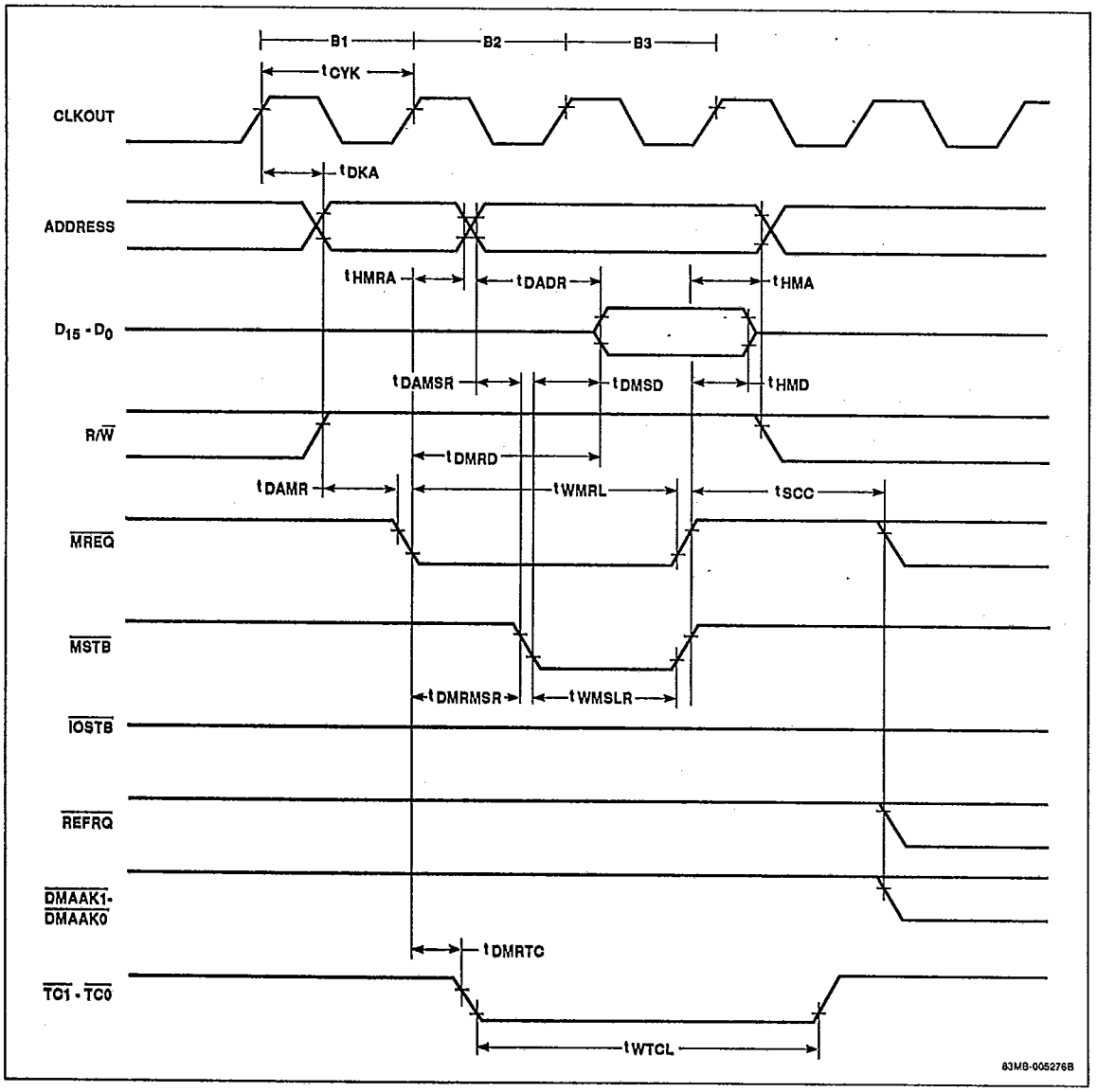
AC Output (Except CLKOUT)



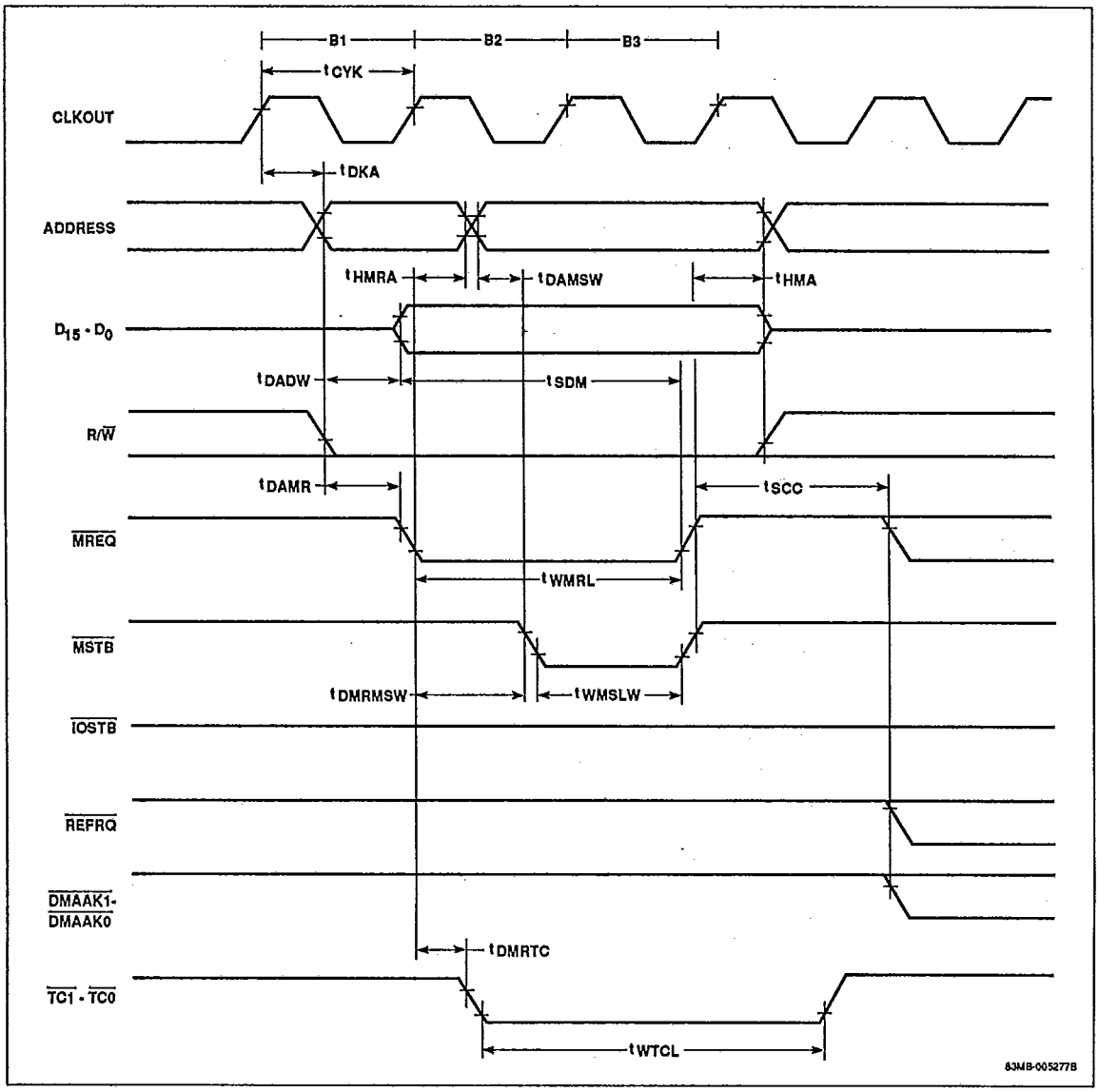
Clock In and Clock Out



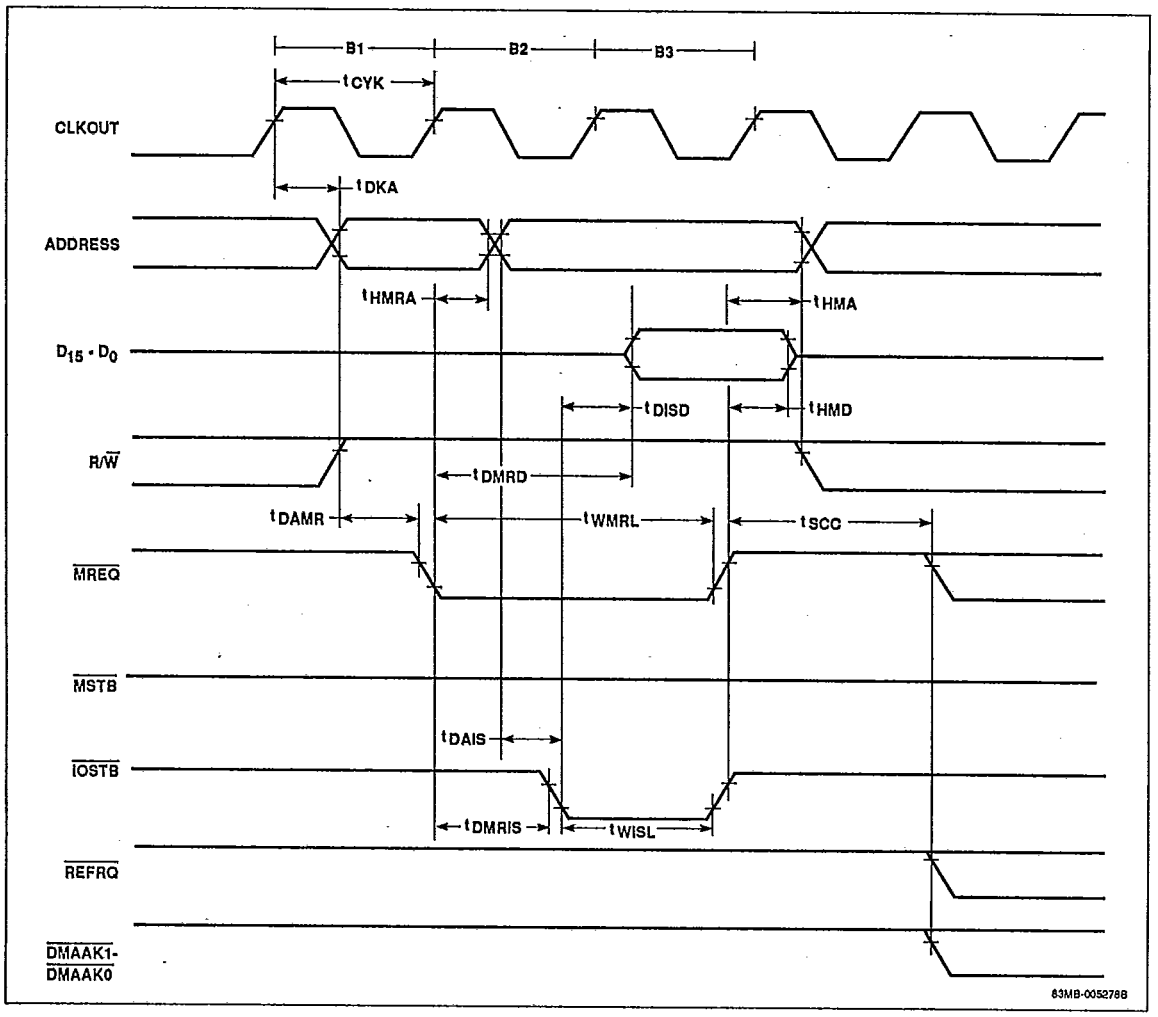
Memory Read



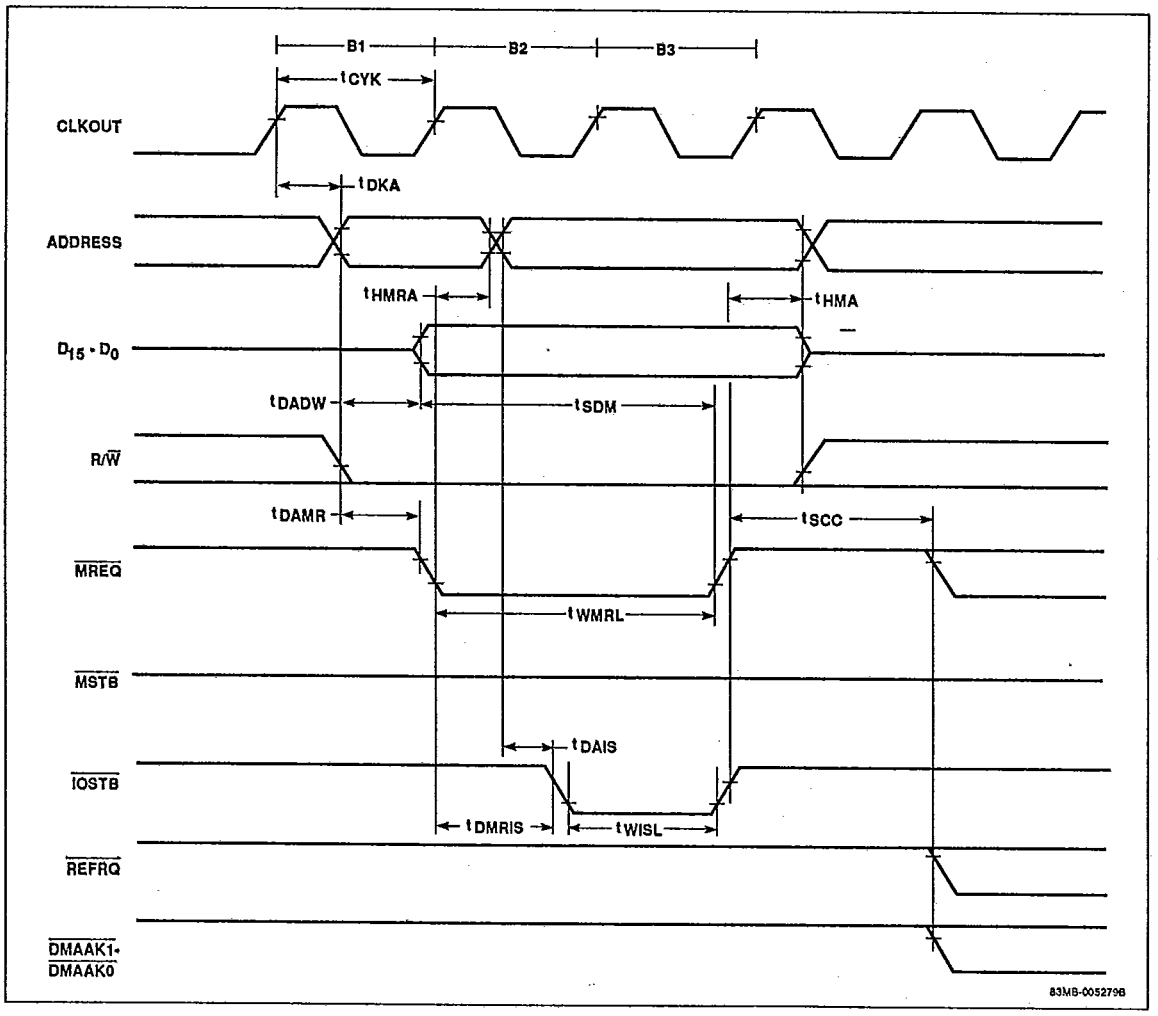
Memory Write



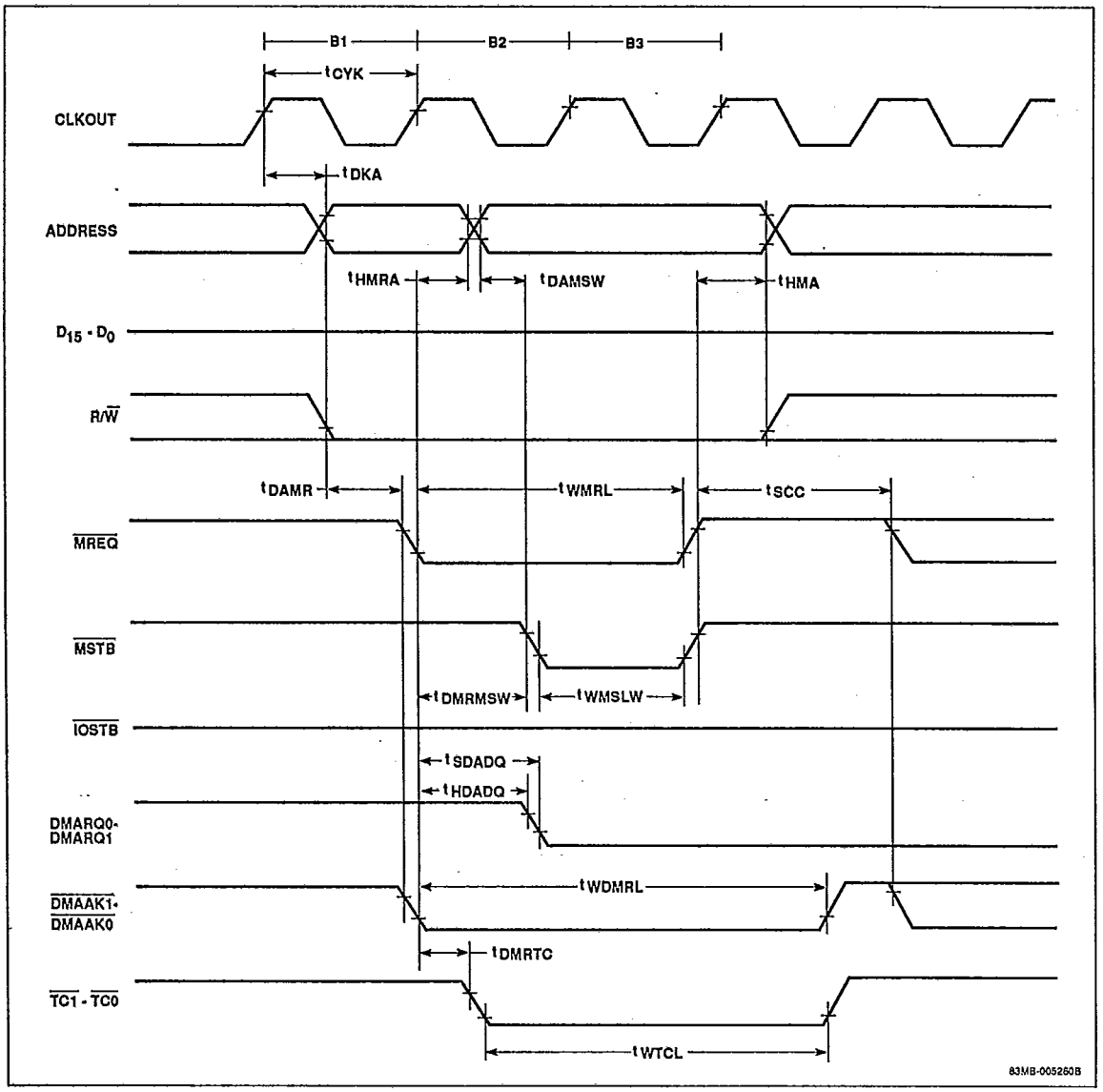
I/O Read



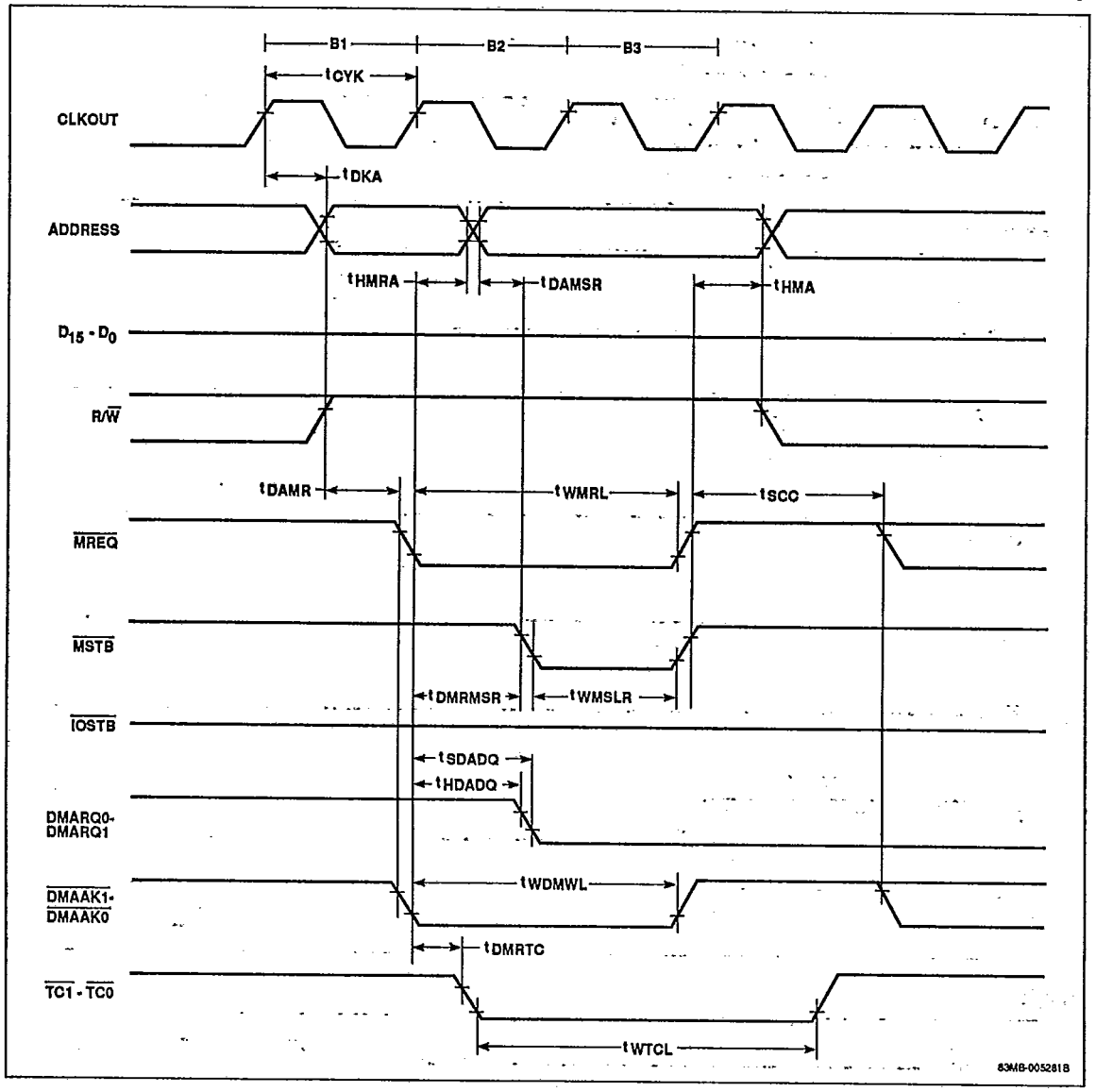
I/O Write



DMA, I/O to Memory

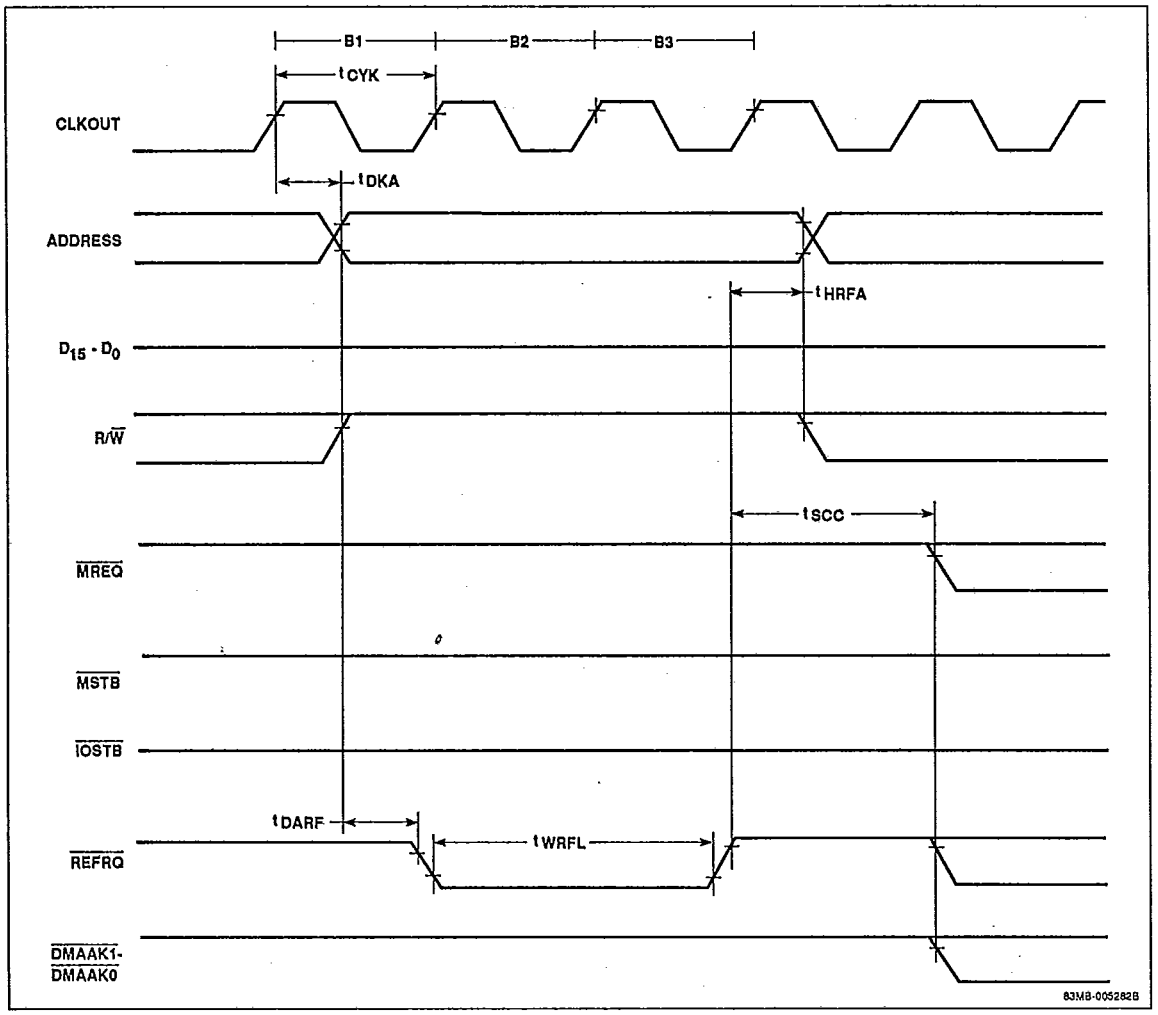


DMA, Memory to I/O

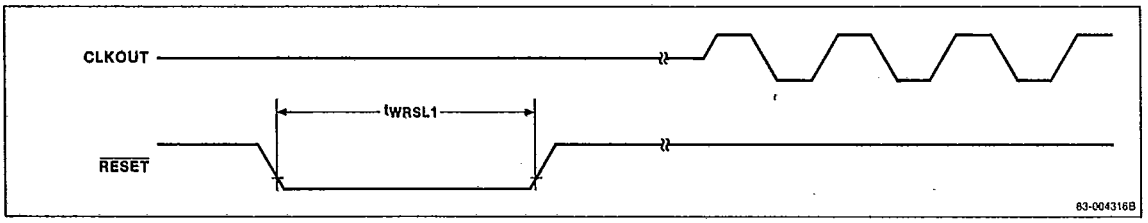


4e

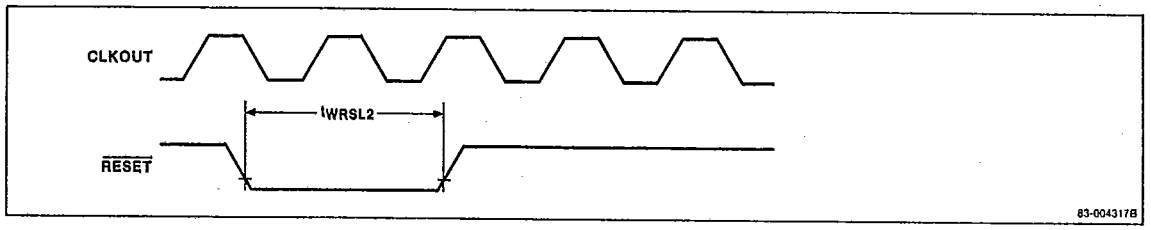
Refresh



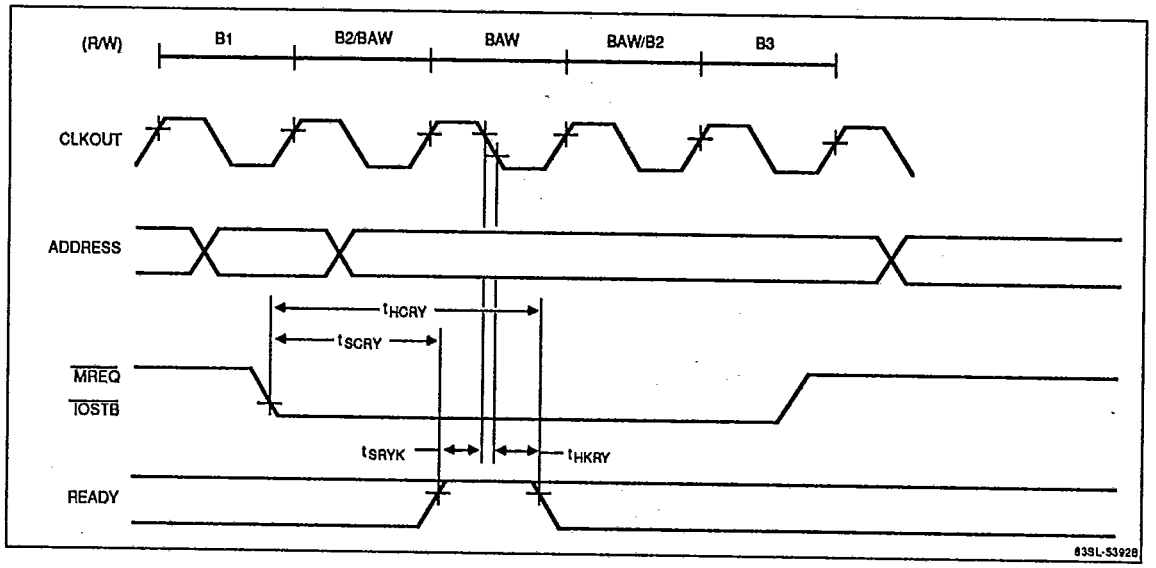
RESET 1



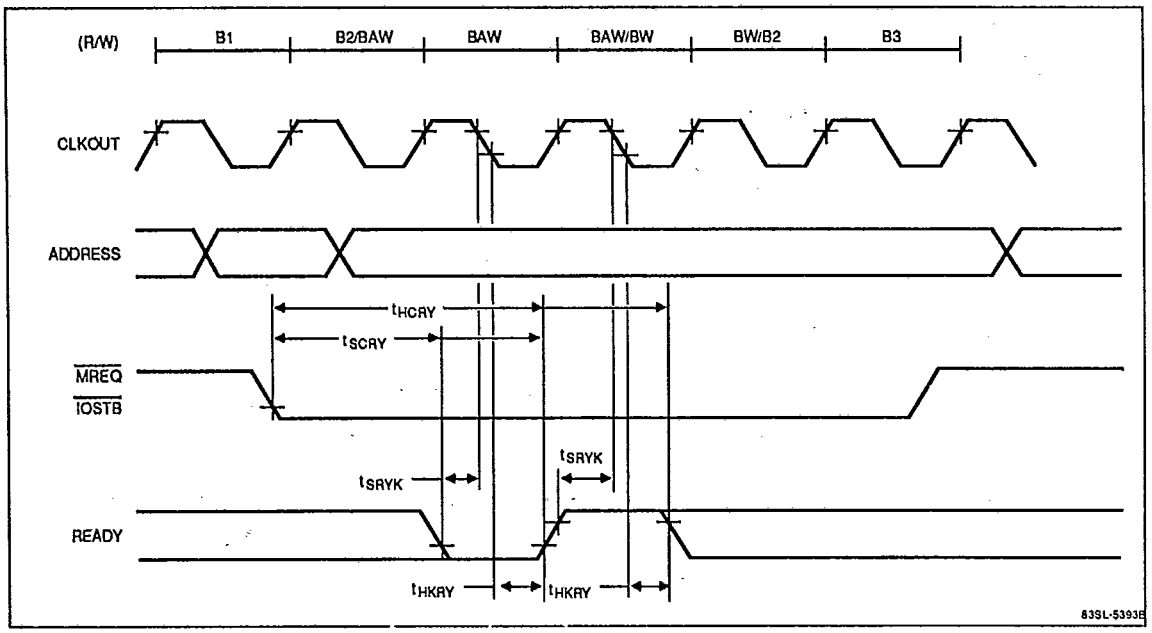
RESET 2



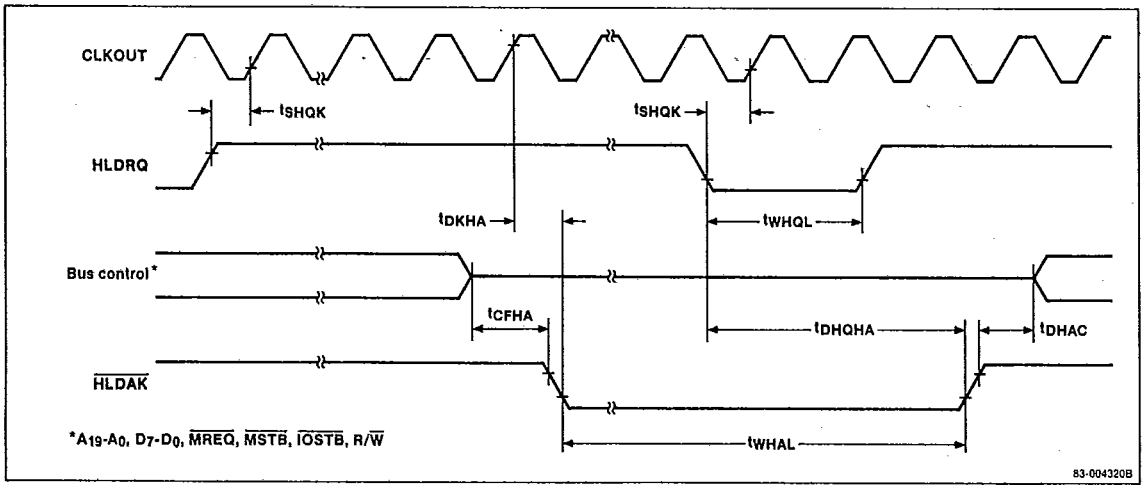
READY 1



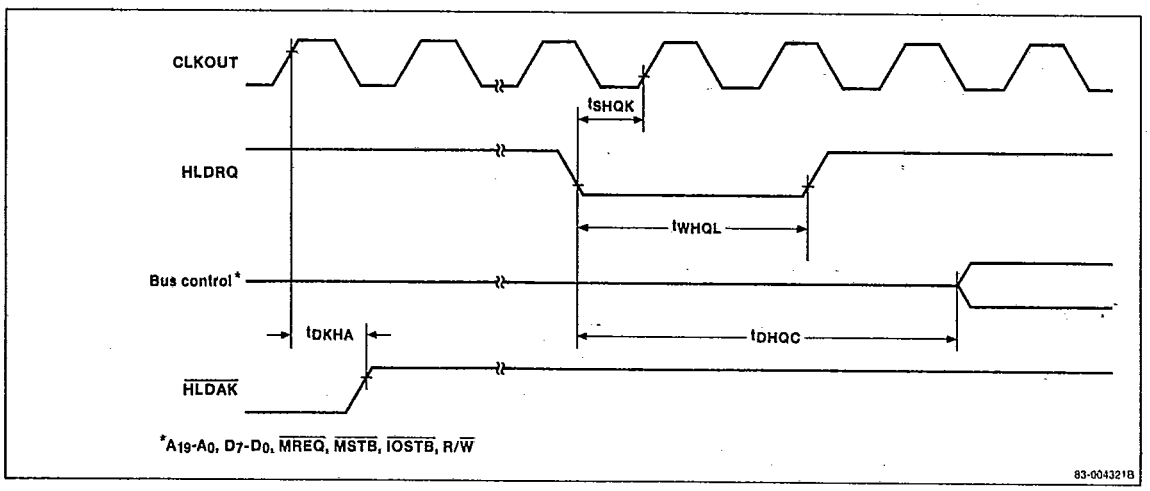
READY 2



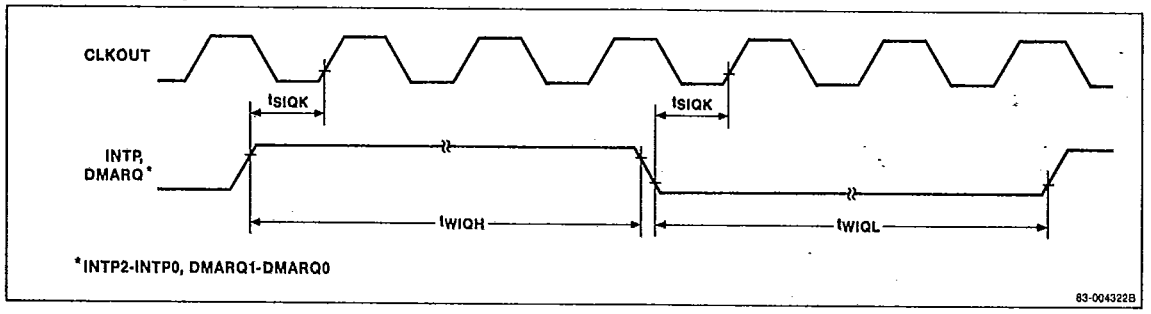
HLDRQ/ HLDAK 1



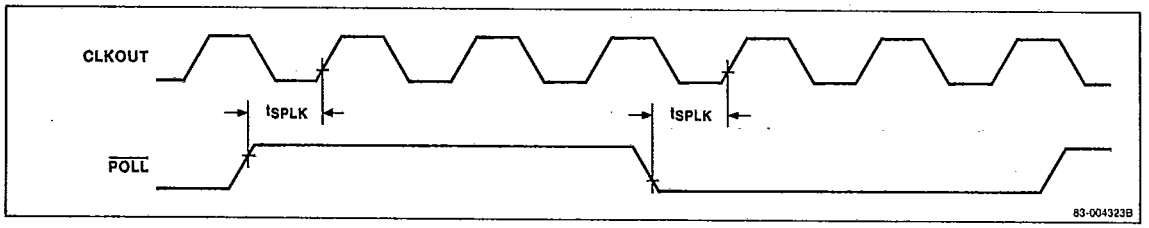
HLDRQ/$\overline{\text{HLDAK}}$ 2



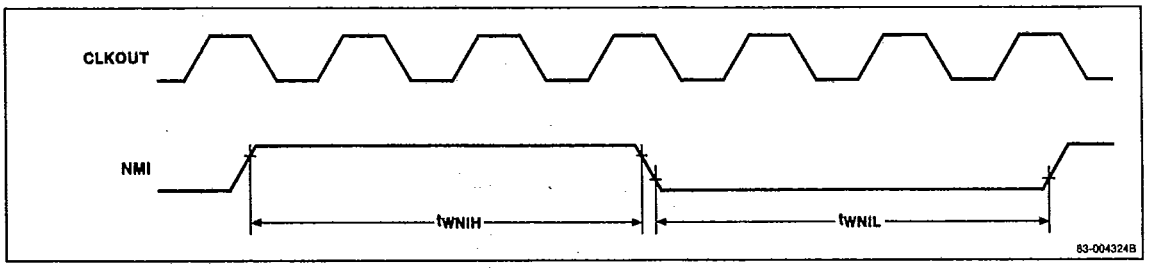
INTP, DMARQ Input



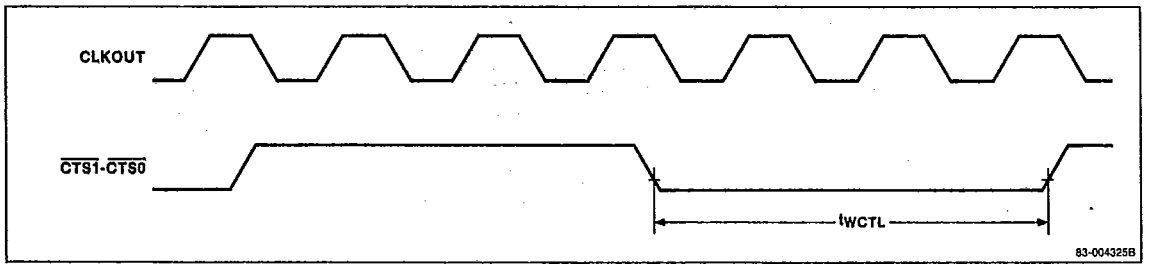
$\overline{\text{POLL}}$ Input



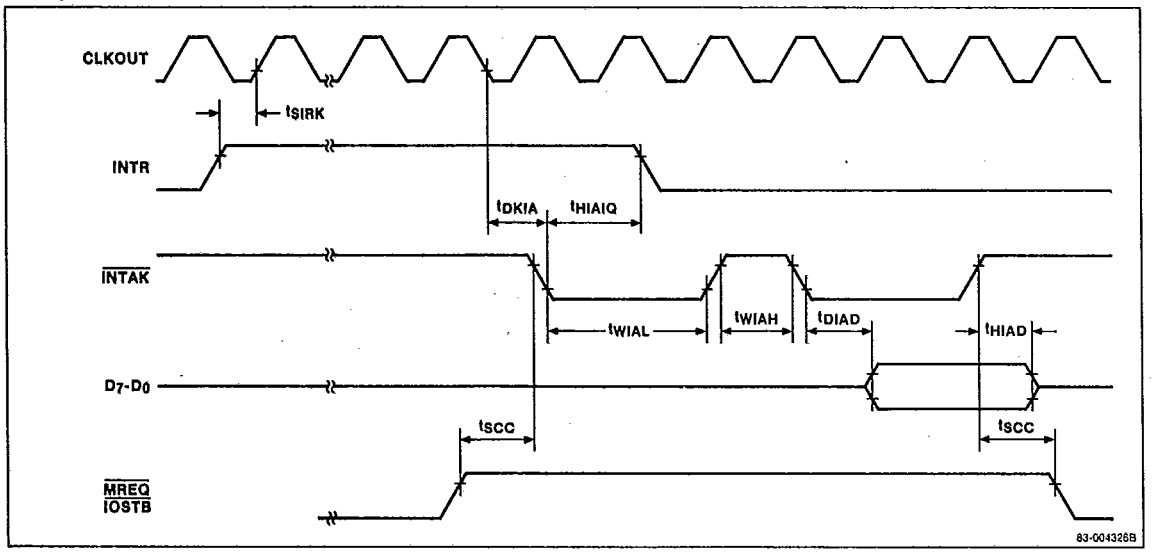
NMI Input



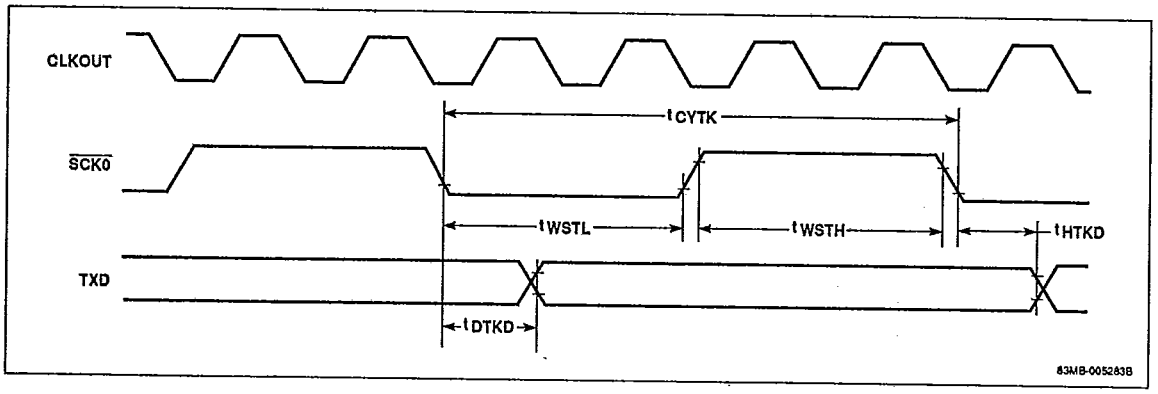
$\overline{CTS}$ Input



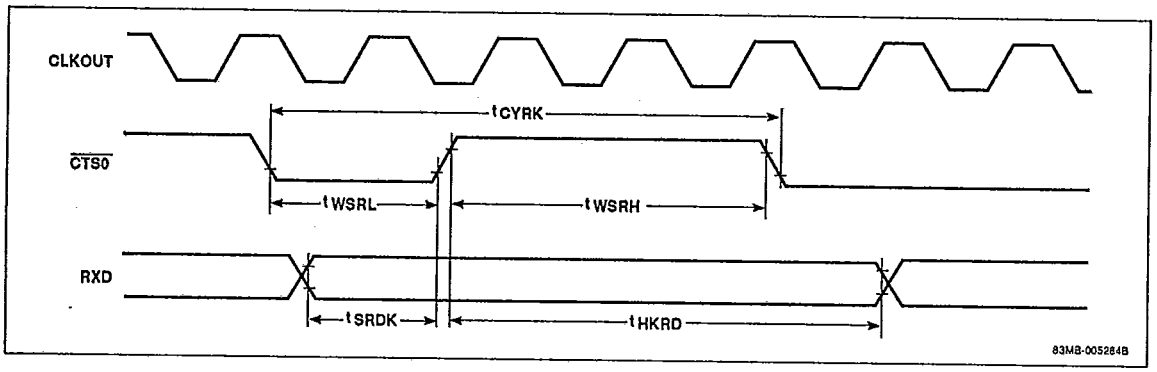
INTR/$\overline{INTAK}$



Serial Transmit



Serial Receive



4e

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

Symbols and Abbreviations (cont)

Identifier	Description
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)
IX	Index register (source) (16 bits)
IY	Index register (destination) (16 bits)

Symbols and Abbreviations (cont)

Identifier	Description
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]
	reg16, mem16	EA+10+2W [EA+7+W]
	reg8, imm8	5
	mem16, imm16	6
	mem8,imm8	EA+11+2W [EA+9+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]
	AL, imm8	5
	AW, imm16	6
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
BRK	3	50+5W [38+5W]
	imm8	51+5W [39+5W]
BRKCS		15
BRKV		50+5W [38+5W]
BTCLR		29
BUSLOCK		2

Mnemonic	Operand	Clocks
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	reg16, mem32	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]
CMPBKB		16+(23+2W)n
CMPBKW		16+(23+2W)n
CMPM	mem8	18+W
	mem16	19+2W
CMPMB	n > 1	16+(16+W)n
CMPMW	n > 1	16+(16+2W)n
CVTBD		19
CVTBW		3
CVTDB		20
CVTWL		8
DBNZ		8 or 17
DBNZE		8 or 17
DBNZNE		8 or 17

Instruction Clock Count (cont)

Mnemonic	Operand	Clocks
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
DSO:		2
DS1:		2
EI		12
EXT	reg8, reg8	41-121
	reg8, Imm4	42-122
FINT		2
FPO1		55+5W [43+5W]
FPO2		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]
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	n > 1	16+(11+W)n
LDMW	n > 1	16+(11+W)n

Mnemonic	Operand	Clocks
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 [19+W]
MOVBKB	n > 1	16+(18+2W)n [16+(13+W)n]
MOVBKW	n > 1	16+(18+2W)n [16+(10+W)n]
MOVSPA		16
MOVSPB	reg16	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
MULU	reg8	24
	mem8	EA+27+W
	reg16	32
	mem16	EA+33+W

Instruction Clock Count (cont)

Mnemonic	Operand	Clocks
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+W [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]
	DS0,1	12+W
	SS	12+W
	DS0	12+W
	PSW	13+W
	R	74+8W [58]
PREPARE	Imm16, Imm8	Imm8 = 0:26+W Imm8 = 1:37+2W Imm8 = n, n > 1:44+19 (n−1)+2Wn
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+2W
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

Instruction Clock Count (cont)

Mnemonic	Operand	Clocks
ROR4	reg8	21
	mem8	EA+26+2W [EA+24+2W]
RORC		Same as ROL
SET1	CY	2
	DIR	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]
SHL		Same as ROL
SHR		Same as ROL
SHRA		Same as ROL
SS:		2
STM	mem8	13+W [10]
	mem16	13+W (10]
STMB	n > 1	16+(9+W)n [16+(7+W)n]
STMW	n > 1	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
XCH	reg8, reg8	3
	reg16, reg16	3
	mem8, reg8/ reg8, mem8	EA+12+2W [EA+9+W]
	mem16, reg16/ reg16, mem16	EA+12+2W [EA+9+2W]
	AW, reg16	4
	reg16, AW	4
XOR		Same as AND

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

Instruction Clock Count for Operations

	Byte		Word	
	RAM Enable	RAM Disable	RAM Enable	RAM Disable
Context switch interrupt	—	—	33	33
DMA (Single-step mode)	6+2W	6+2W	6+2W	6+2W
DMA (Demand release mode)	3+W	3+W	3+W	3+W
DMA (Burst mode)	(6+2W)n	(6+2W)n	(6+2W)n	(6+2W)n
DMA (Single-transfer mode)	3+W	3+W	3+W	3+W
Interrupt (INT pin)	—	—	57+3W	57+3W
Macro service, sfr ← mem	31+W	26+W	31+W	26+W
Macro service, mem ← sfr	28+W	27+W	28+W	27+W
Macro service (Search char mode), sfr ← mem	34+W	34+W	—	—
Macro service (Search char mode), mem ← sfr	44+W	41+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

Bus Controller Latency

		Clocks	
Latnecy	Mode	Typ	Max
Hold request	Refresh active		9+3W
	Intack active		10+2W
	No refresh or intack		7+2W
DMA request (Notes 1, 2)	Burst	3	14+2W
	Single-step	3	14+2W
	Demand releas	3	14+2W
	Single-transfer	4	14+2W

Notes:

(1) The listed DMA latency times are the maximum number of clocks when a DMA request is asserted until $\overline{\text{DMAAK}}$ or $\overline{\text{MREQ}}$ goes low in the corresponding DMA cycle.

(2) The test conditions are: no wait states, no interrupts, no macroservice requests, and no hold requests.

Interrupt Latency

	Clocks	
Source	Typ	Max
NMI pin	12+N	18+N
INT pin	8+N	8+N
All others	27+N	15+N

Instruction Set

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Data Transfer																	
MOV	reg, reg	reg ← reg	1	0	0	0	1	0	1	W	2						
			1	1		reg			reg								
	mem, reg	(mem) ← reg	1	0	0	0	1	0	0	W	2-4						
			mod			reg			mem								
	reg, mem	reg ← (mem)	1	0	0	0	1	0	1	W	2-4						
			mod			reg			mem								
	mem, imm	(mem) ← imm	1	1	0	0	0	1	1	W	3-6						
			mod	0	0	0		mem									
	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	2						
			1	1	0	sreg			reg								
	sreg, mem16	sreg ← (mem16) sreg: SS, DS0, DS1	1	0	0	0	1	1	1	0	2-4						
			mod	0		sreg			mem								
	reg16, sreg	reg16 ← sreg	1	0	0	0	1	1	0	0	2						
			1	1	0	sreg			reg								
	mem16, sreg	(mem16) ← sreg	1	0	0	0	1	1	0	0	2-4						
			mod	0		sreg			mem								
	DS0, reg16, mem32	reg16 ← (mem32), DS0 ← (mem32 + 2)	1	1	0	0	0	1	0	1	2-4						
			mod			reg			mem								
	DS1, reg16, mem32	reg16 ← (mem32), DS1 ← (mem32 + 2)	1	1	0	0	0	1	0	0	2-4						
			mod			reg			mem								
	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	2-4						
			mod			reg			mem								
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	2						
			1	1		reg			reg								
	mem, reg or reg, mem	(mem) ↔ reg	1	0	0	0	0	1	1	W	2-4						
			mod			reg			mem								
	AW, reg16 or reg16, AW	AW ↔ reg16	1	0	0	1	0		reg		1						

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
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						
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						

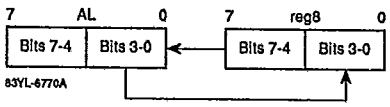
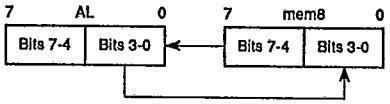
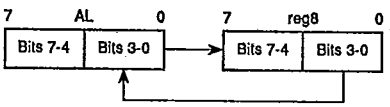
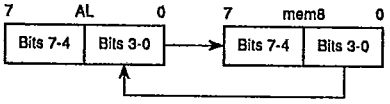
Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Bit Field Transfer																	
INS	reg8, reg8	16-bit field ← AW	0	0	0	0	1	1	1	1	3						
			0	0	1	1	0	0	0	1							
			1	1		reg			reg								
	reg8, imm4	16-bit field ← AW	0	0	0	0	1	1	1	1	4						
			0	0	1	1	1	0	0	1							
			1	1	0	0	0		reg								
EXT	reg8, reg8	AW ← 16-bit field	0	0	0	0	1	1	1	1	3						
			0	0	1	1	0	0	1	1							
			1	1		reg			reg								
	reg8, imm4	AW ← 16-bit field	0	0	0	0	1	1	1	1	4						
			0	0	1	1	1	0	1	1							
			1	1	0	0	0		reg								
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	1						

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Addition/Subtraction																	
ADD	reg, reg	reg ← reg + reg	0	0	0	0	0	0	1	W	2	x	x	x	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) ← (mem) + reg	0	0	0	0	0	0	0	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, mem	reg ← reg + (mem)	0	0	0	0	0	0	1	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, imm	reg ← reg + imm	1	0	0	0	0	0	S	W	3-4	x	x	x	x	x	x
			1	1	0	0	0		reg								
	mem, imm	(mem) ← (mem) + imm	1	0	0	0	0	0	S	W	3-6	x	x	x	x	x	x
			mod		0	0	0		mem								
	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
ADDC	reg, reg	reg ← reg + reg + CY	0	0	0	1	0	0	1	W	2	x	x	x	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) ← (mem) + reg + CY	0	0	0	1	0	0	0	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, mem	reg ← reg + (mem) + CY	0	0	0	1	0	0	1	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, imm	reg ← reg + imm + CY	1	0	0	0	0	0	S	W	3-4	x	x	x	x	x	x
			1	1	0	1	0		reg								
	mem, imm	(mem) ← (mem) + imm + CY	1	0	0	0	0	0	S	W	3-6	x	x	x	x	x	x
			mod		0	1	0		mem								
	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	2	x	x	x	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) ← (mem) − reg	0	0	1	0	1	0	0	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, mem	reg ← reg − (mem)	0	0	1	0	1	0	1	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, imm	reg ← reg − imm	1	0	0	0	0	0	S	W	3-4	x	x	x	x	x	x
			1	1	1	0	1		reg								
	mem, imm	(mem) ← (mem) − imm	1	0	0	0	0	0	S	W	3-6	x	x	x	x	x	x
			mod		1	0	1		mem								
	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	2	x	x	x	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) ← (mem)−reg−CY	0	0	0	1	1	0	0	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, mem	reg ← reg−(mem)−CY	0	0	0	1	1	0	1	W	2-4	x	x	x	x	x	
			mod			reg			mem								

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Addition/Subtraction (cont)																	
SUBC	reg, imm	reg ← reg − imm − CY	1	0	0	0	0	0	S	W	3-4	x	x	x	x	x	x
			1	1	0	1	1		reg								
	mem, imm	(mem) ← (mem) − imm − CY	1	0	0	0	0	0	S	W	3-6	x	x	x	x	x	x
			mod		0	1	1		mem								
	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
BCD Operation																	
ADD4S		dst BCD string ← dst BCD string + src BCD string	0	0	0	0	1	1	1	1	2	u	x	u	u	u	x
			0	0	1	0	0	0	0	0							
SUB4S		dst BCD string ← dst BCD string − src BCD string	0	0	0	0	1	1	1	1	2	u	x	u	u	u	x
			0	0	1	0	0	0	1	0							
CMP4S		dst BCD string − src BCD string	0	0	0	0	1	1	1	1	2	u	x	u	u	u	x
			0	0	1	0	0	1	1	0							
ROL4	reg 8		0	0	0	0	1	1	1	1	3						
			0	0	1	0	1	0	0	0							
			1	1	0	0	0		reg								
	mem 8		0	0	0	0	1	1	1	1	3-5						
			0	0	1	0	1	0	0	0							
			mod		0	0	0		mem								
ROR4	reg 8		0	0	0	0	1	1	1	1	3						
			0	0	1	0	1	0	1	0							
			1	1	0	0	0		reg								
	mem 8		0	0	0	0	1	1	1	1	3-5						
			0	0	1	0	1	0	1	0							
			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 = 1: 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 = 1: 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	Bytes	Flags AC	CY	V	P	S	Z
Increment/Decrement																	
INC	reg8	reg8 ← reg8 + 1	1	1	1	1	1	1	1	0	2	x		x	x	x	x
			1	1	0	0	0	reg									
	mem	(mem) ← (mem) + 1	1	1	1	1	1	1	1	W	2-4	x		x	x	x	x
			mod		0	0	0	mem									
	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	2	x		x	x	x	x
			1	1	0	0	1	reg									
	mem	(mem) ← (mem) − 1	1	1	1	1	1	1	1	W	2-4	x		x	x	x	x
			mod		0	0	1	mem									
	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	2	u	x	x	u	u	u
			1	1	1	0	0	reg									
	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	2-4	u	x	x	u	u	u
			mod		1	0	0	mem									
	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	2	u	x	x	u	u	u
			1	1	1	0	0	reg									
	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	2-4	u	x	x	u	u	u
			mod		1	0	0	mem									
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	2	u	x	x	u	u	u
			1	1	1	0	1	reg									
	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	2-4	u	x	x	u	u	u
			mod		1	0	1	mem									
	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	2	u	x	x	u	u	u
			1	1	1	0	1	reg									
	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	2-4	u	x	x	u	u	u
			mod		1	0	1	mem									
	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	3	u	x	x	u	u	u
			1	1		reg			reg								
	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	3-5	u	x	x	u	u	u
			mod			reg			mem								
	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	4	u	x	x	u	u	u
			1	1		reg			reg								
	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	4-6	u	x	x	u	u	u
			mod			reg			mem								

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
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	2	u	u	u	u	u	u
			1	1	1	1	0		reg								
	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	2-4	u	u	u	u	u	u
			mod		1	1	0		mem								
	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	2	u	u	u	u	u	u
			1	1	1	1	0		reg								
	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	2-4	u	u	u	u	u	u
			mod		1	1	0		mem								
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	2	u	u	u	u	u	u
			1	1	1	1	1		reg								
	mem8	temp ← AW 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	2-4	u	u	u	u	u	u
			mod		1	1	1		mem								

4e

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code	Bytes	Flags					
			7 6 5 4 3 2 1 0		AC	CY	V	P	S	Z
Signed Division (cont)										
DIV	reg16	temp ← DW, AW When temp ÷ reg16 > 0 and reg16 > 7FFFH or temp ÷ reg16 < 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 % reg16, AL ← temp ÷ reg16	1 1 1 1 0 1 1 1 1 1 1 1 1 reg	2	u	u	u	u	u	u
	mem16	temp ← DW, AW When temp ÷ (mem16) > 0 and (mem16) > 7FFFH or temp ÷ (mem16) < 0 and temp ÷ (mem16) < 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 % (mem16), AL ← temp ÷ (mem16)	1 1 1 1 0 1 1 1 mod 1 1 1 mem	2-4	u	u	u	u	u	u
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

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code								Bytes	Flags					
			7	6	5	4	3	2	1	0		AC	CY	V	P	S	Z
Complement																	
NOT	reg	reg ← $\overline{reg}$	1	1	1	1	0	1	1	W	2						
			1	1	0	1	0		reg								
	mem	(mem) ← $\overline{(mem)}$	1	1	1	1	0	1	1	W	2-4						
			mod			0	1	0	mem								
NEG	reg	reg ← $\overline{reg}$ + 1	1	1	1	1	0	1	1	W	2	x	x	x	x	x	x
			1	1	0	1	1		reg								
	mem	(mem) ← $\overline{(mem)}$ + 1	1	1	1	1	0	1	1	W	2-4	x	x	x	x	x	x
			mod			0	1	1	mem								
Logical Operation																	
TEST	reg, reg	reg AND reg	1	0	0	0	0	1	0	W	2	u	0	0	x	x	x
			1	1		reg			reg								
	mem, reg or reg, mem	(mem) AND reg	1	0	0	0	0	1	0	W	2-4	u	0	0	x	x	x
			mod			reg			mem								
	reg, imm	reg AND imm	1	1	1	1	0	1	1	W	3-4	u	0	0	x	x	x
			1	1	0	0	0		reg								
	mem, imm	(mem) AND imm	1	1	1	1	0	1	1	W	3-6	u	0	0	x	x	x
			mod			0	0	0	mem								
	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	2	u	0	0	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) ← (mem) AND reg	0	0	1	0	0	0	0	W	2-4	u	0	0	x	x	x
			mod			reg			mem								
	reg, mem	reg ← reg AND (mem)	0	0	1	0	0	0	1	W	2-4	u	0	0	x	x	x
			mod			reg			mem								
	reg, imm	reg ← reg AND imm	1	0	0	0	0	0	0	W	3-4	u	0	0	x	x	x
			1	1	1	0	0		reg								
	mem, imm	(mem) ← (mem) AND imm	1	0	0	0	0	0	0	W	3-6	u	0	0	x	x	x
			mod			1	0	0	mem								
	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 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Logical Operation (cont)																	
OR	reg, reg	reg ← reg OR reg	0	0	0	0	1	0	1	W	2	u	0	0	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) ← (mem) OR reg	0	0	0	0	1	0	0	W	2-4	u	0	0	x	x	x
			mod			reg			mem								
	reg, mem	reg ← reg OR (mem)	0	0	0	0	1	0	1	W	2-4	u	0	0	x	x	x
			mod			reg			mem								
	reg, imm	reg ← reg OR imm	1	0	0	0	0	0	0	W	3-4	u	0	0	x	x	x
			1	1	0	0	1		reg								
	mem, imm	(mem) ← (mem) OR imm	1	0	0	0	0	0	0	W	3-6	u	0	0	x	x	x
			mod		0	0	1		mem								
	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	2	u	0	0	x	x	x
			1	1		reg			reg								
	mem, reg	(mem) ← (mem) XOR reg	0	0	1	1	0	0	0	W	2-4	u	0	0	x	x	x
			mod			reg			mem								
	reg, mem	reg ← reg XOR (mem)	0	0	1	1	0	0	1	W	2-4	u	0	0	x	x	x
			mod			reg			mem								
	reg, imm	reg ← reg XOR imm	1	0	0	0	0	0	0	W	3-4	u	0	0	x	x	x
			1	1	1	1	0		reg								
	mem, imm	(mem) ← (mem) XOR imm	1	0	0	0	0	0	0	W	3-6	u	0	0	x	x	x
			mod		1	1	0		mem								
	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																	
TEST1	reg8, CL	reg8 bit no. CL = 0: Z ← 1 reg8 bit no. CL = 1: Z ← 0	0	0	0	0	1	1	1	1	3	u	0	0	u	u	x
			0	0	0	1	0	0	0	0							
			1	1	0	0	0		reg								
	mem8, CL	(mem8) bit no. CL = 0: Z ← 1 (mem8) bit no. CL = 1: Z ← 0	0	0	0	0	1	1	1	1	3-5	u	0	0	u	u	x
			0	0	0	1	0	0	0	0							
			mod		0	0	0		mem								
	reg16, CL	reg16 bit no. CL = 0: Z ← 1 reg16 bit no. CL = 1: Z ← 0	0	0	0	0	1	1	1	1	3	u	0	0	u	u	x
			0	0	0	1	0	0	0	1							
			1	1	0	0	0		reg								
	mem16, CL	(mem16) bit no. CL = 0: Z ← 1 (mem16) bit no. CL = 1: Z ← 0	0	0	0	0	1	1	1	1	3-5	u	0	0	u	u	x
			0	0	0	1	0	0	0	1							
			mod		0	0	0		mem								
	reg8, imm3	reg8 bit no. imm3 = 0: Z ← 1 reg8 bit no. imm3 = 1: Z ← 0	0	0	0	0	1	1	1	1	4	u	0	0	u	u	x
			0	0	0	1	1	0	0	0							
			1	1	0	0	0		reg								

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Bit Operation (cont)																	
TEST1	mem8, imm3	(mem8) bit no. imm3 = 0: Z ← 1 (mem8) bit no. imm3 = 1: Z ← 0	0	0	0	0	1	1	1	1	4-6	u	0	0	u	u	x
			0	0	0	1	1	0	0	0							
			mod		0	0	0		mem								
	reg16, imm4	reg16 bit no. imm4 = 0: Z ← 1 reg16 bit no. imm4 = 1: Z ← 0	0	0	0	0	1	1	1	1	4	u	0	0	u	u	x
			0	0	0	1	1	0	0	1							
			1	1	0	0	0		reg								
	mem16, imm4	(mem16) bit no. imm4 = 0: Z ← 1 (mem16) bit no. imm4 = 1: Z ← 0	0	0	0	0	1	1	1	1	4-6	u	0	0	u	u	x
			0	0	0	1	1	0	0	1							
			mod		0	0	0		mem								
NOT1	reg8, CL	reg8 bit no. CL ← $\overline{\text{reg8 bit no. CL}}$	0	0	0	0	1	1	1	1	3						
			0	0	0	1	0	1	1	0							
			1	1	0	0	0		reg								
	mem8, CL	(mem8) bit no. CL ← $\overline{\text{(mem8) bit no. CL}}$	0	0	0	0	1	1	1	1	3-5						
			0	0	0	1	0	1	1	0							
			mod		0	0	0		mem								
	reg16, CL	reg16 bit no. CL ← $\overline{\text{reg16 bit no. CL}}$	0	0	0	0	1	1	1	1	3						
			0	0	0	1	0	1	1	1							
			1	1	0	0	0		reg								
	mem16, CL	(mem16) bit no. CL ← $\overline{\text{(mem16) bit no. CL}}$	0	0	0	0	1	1	1	1	3-5						
			0	0	0	1	0	1	1	1							
			mod		0	0	0		mem								
	reg8, imm3	reg8 bit no. imm3 ← $\overline{\text{reg8 bit no. imm3}}$	0	0	0	0	1	1	1	1	4						
			0	0	0	1	1	1	1	0							
			1	1	0	0	0		reg								
	mem8, imm3	(mem8) bit no. imm3 ← $\overline{\text{(mem8) bit no. imm3}}$	0	0	0	0	1	1	1	1	4-6						
			0	0	0	1	1	1	1	0							
			mod		0	0	0		mem								
	reg16, imm4	reg16 bit no. imm4 ← $\overline{\text{reg16 bit no. imm4}}$	0	0	0	0	1	1	1	1	4						
			0	0	0	1	1	1	1	1							
			1	1	0	0	0		reg								
	mem16, imm4	(mem16) bit no. imm4 ← $\overline{\text{(mem16) bit no. imm4}}$	0	0	0	0	1	1	1	1	4-6						
			0	0	0	1	1	1	1	1							
			mod		0	0	0		mem								
	CY	CY ← CY	1	1	1	1	0	1	0	1	1		x				

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Bit Operation (cont)																	
CLR1	reg8, CL	reg8 bit no. CL ← 0	0	0	0	0	1	1	1	1	3						
			0	0	0	1	0	0	1	0							
			1	1	0	0	0	reg									
	mem8, CL	(mem8) bit no. CL ← 0	0	0	0	0	1	1	1	1	3-5						
			0	0	0	1	0	0	1	0							
			mod		0	0	0	mem									
	reg16, CL	reg16 bit no. CL ← 0	0	0	0	0	1	1	1	1	3						
			0	0	0	1	0	0	1	1							
			1	1	0	0	0	reg									
	mem16, CL	(mem16) bit no. CL ← 0	0	0	0	0	1	1	1	1	3-5						
			0	0	0	1	0	0	1	1							
			mod		0	0	0	mem									
	reg8, imm3	reg8 bit no. imm3 ← 0	0	0	0	0	1	1	1	1	4						
			0	0	0	1	1	0	1	0							
			1	1	0	0	0	reg									
	mem8, imm3	(mem8) bit no. imm3 ← 0	0	0	0	0	1	1	1	1	4-6						
			0	0	0	1	1	0	1	0							
			mod		0	0	0	mem									
	reg16, imm4	reg16 bit no. imm4 ← 0	0	0	0	0	1	1	1	1	4						
			0	0	0	1	1	0	1	1							
			1	1	0	0	0	reg									
	mem16, imm4	(mem16) bit no. imm4 ← 0	0	0	0	0	1	1	1	1	4-6						
			0	0	0	1	1	0	1	1							
			mod		0	0	0	mem									
	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						
SET1	reg8, CL	reg8 bit no. CL ← 1	0	0	0	0	1	1	1	1	3						
			0	0	0	1	0	1	0	0							
			1	1	0	0	0	reg									
	mem8, CL	(mem8) bit no. CL ← 1	0	0	0	0	1	1	1	1	3-5						
			0	0	0	1	0	1	0	0							
			mod		0	0	0	mem									
	reg16, CL	reg16 bit no. CL ← 1	0	0	0	0	1	1	1	1	3						
			0	0	0	1	0	1	0	1							
			1	1	0	0	0	reg									
	mem16, CL	(mem16) bit no. CL ← 1	0	0	0	0	1	1	1	1	3-5						
			0	0	0	1	0	1	0	1							
			mod		0	0	0	mem									

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Bit Operation (cont)																	
SET1	reg8, imm3	reg8 bit no. imm3 ← 1	0	0	0	0	1	1	1	1	4						
			0	0	0	1	1	1	0	0							
			1	1	0	0	0	reg									
	mem8, imm3	(mem8) bit no. imm3 ← 1	0	0	0	0	1	1	1	1	4-6						
			0	0	0	1	1	1	0	0							
			mod		0	0	0	mem									
	reg16, imm4	reg16 bit no. imm4 ← 1	0	0	0	0	1	1	1	1	4						
			0	0	0	1	1	1	0	1							
			1	1	0	0	0	reg									
	mem16, imm4	(mem16) bit no. imm4 ← 1	0	0	0	0	1	1	1	1	4-6						
			0	0	0	1	1	1	0	1							
			mod		0	0	0	mem									
	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	2	u	x	x	x	x	x
			1	1	1	0	0	reg									
	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	2-4	u	x	x	x	x	x
			mod		1	0	0	mem									
	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	2	u	x	u	x	x	x
			1	1	1	0	0	reg									
	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	2-4	u	x	u	x	x	x
			mod		1	0	0	mem									
	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	3	u	x	u	x	x	x
			1	1	1	0	0	reg									
	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	3-5	u	x	u	x	x	x
			mod		1	0	0	mem									

4e

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Shift (cont)																	
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	2	u	x	x	x	x	x
			1	1	1	0	1		reg								
	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	2-4	u	x	x	x	x	x
			mod			1	0	1	mem								
	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	2	u	x	u	x	x	x
			1	1	1	0	1		reg								
	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	2-4	u	x	u	x	x	x
			mod			1	0	1	mem								
	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	3	u	x	u	x	x	x
			1	1	1	0	1		reg								
	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	3-5	u	x	u	x	x	x
			mod			1	0	1	mem								
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	2	u	x	0	x	x	x
			1	1	1	1	1		reg								
	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	2-4	u	x	0	x	x	x
			mod			1	1	1	mem								
	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	2	u	x	u	x	x	x
			1	1	1	1	1		reg								
	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	2-4	u	x	u	x	x	x
			mod			1	1	1	mem								
	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	3	u	x	u	x	x	x
			1	1	1	1	1		reg								
	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	3-5	u	x	u	x	x	x
			mod			1	1	1	mem								
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	2		x	x			
			1	1	0	0	0		reg								
	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	2-4		x	x			
			mod			0	0	0	mem								

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
Rotation (cont)																	
ROL	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	2		x	u			
			1	1	0	0	0		reg								
	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	2-4		x	u			
			mod		0	0	0		mem								
	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	3		x	u			
			1	1	0	0	0		reg								
	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	3-5		x	u			
			mod		0	0	0		mem								
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	2		x	x			
			1	1	0	0	1		reg								
	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	2-4		x	x			
			mod		0	0	1		mem								
	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	2		x	u			
			1	1	0	0	1		reg								
	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	2-4		x	u			
			mod		0	0	1		mem								
	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	3		x	u			
			1	1	0	0	1		reg								
	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	3-5		x	u			
			mod		0	0	1		mem								

4e

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
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	2		x	x			
			1	1	0	1	0		reg								
	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	2-4		x	x			
			mod		0	1	0		mem								
	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	2		x	u			
			1	1	0	1	0		reg								
	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	2-4		x	u			
			mod		0	1	0		mem								
	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	3		x	u			
			1	1	0	1	0		reg								
	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	3-5		x	u			
			mod		0	1	0		mem								
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	2		x	x			
			1	1	0	1	1		reg								
	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	2-4		x	x			
			mod		0	1	1		mem								
	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	2		x	u			
			1	1	0	1	1		reg								
	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	2-4		x	u			
			mod		0	1	1		mem								
	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	3		x	u			
			1	1	0	1	1		reg								
	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	3-5		x	u			
			mod		0	1	1		mem								

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
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	2						
			1	1	0	1	0	reg									
	memptr16	(SP−1, SP−2) ← PC, SP ← SP−2, PC ← (memptr16)	1	1	1	1	1	1	1	1	2-4						
			mod		0	1	0	mem									
	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	2-4						
			mod		0	1	1	mem									
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	2-4						
			mod		1	1	0	mem									
	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	2-4						
			mod		0	0	0	mem									
	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	4						
DISPOSE		Dispose of stack frame	1	1	0	0	1	0	0	1	1						

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
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	2						
			1	1	1	0	0		reg								
	memptr16	PC ← (memptr16)	1	1	1	1	1	1	1	1	2-4						
			mod		1	0	0		mem								
	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	2-4						
			mod		1	0	1		mem								
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 or (sfr) ← 0	0	0	0	0	1	1	1	1	5						
			1	0	0	1	1	1	0	0							

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
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						
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	2	R	R	R	R	R	R
			1	0	0	1	0	0	0	1							
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	2						
			1	0	0	1	0	0	1	0							
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	2-4						
			mod				reg				mem						
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					
			1	0	1	1	1	1	1	0							
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	2						
			1	1	Y	Y	Y	Z	Z	Z							
	fp-op, mem	data bus ← (mem)	1	1	0	1	1	X	X	X	2-4						
			mod		Y	Y	Y		mem								
FP02 (Note 1)	fp-op	No Operation	0	1	1	0	0	1	1	X	2						
			1	1	Y	Y	Y	Z	Z	Z							
	fp-op, mem	data bus ← (mem)	0	1	1	0	0	1	1	X	2-4						
			mod		Y	Y	Y		mem								

Notes:
(1) Does not execute but does generate an interrupt.

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags AC	CY	V	P	S	Z
CPU Control (cont)																	
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						
Register Bank Switching																	
MOVSPA			0	0	0	0	1	1	1	1	2						
			0	0	1	0	0	1	0	1							
BRKCS	reg16		0	0	0	0	1	1	1	1	3						
			0	0	1	0	1	1	0	1							
MOVSPB	reg16		0	0	0	0	1	1	1	1	3						
			1	0	0	1	0	1	0	1							
			1	1	1	1	1		reg								
TSKSW	reg16		0	0	0	0	1	1	1	1	3	x	x	x	x	x	x
			1	0	0	1	0	1	0	0							
			1	1	1	1	1		reg								