NEC
NEC Electronics Inc.

μPD70325 (V25 Plus) 16-Bit Microcomputer: High-Speed DMA, Single-Chip, CMOS

T-49-19-59

Description

The μPD70325 (V25 Plus) is a high-performance, 16-bit, single-chip microcomputer with an 8-bit external data bus. The μPD70325 is fully software compatible with the μPD70108/116 (V20®/30®) as well as the μPD70320/330 (V25™/35™). The V25 Plus microcomputer demonstrates numerous enhancements over the standard V25; however, it maintains strict pin compatibility with its predecessor, the V25.

The V25 Plus offers improved DMA transfer rates to 5 megabytes/second, additional serial channel status flags, improved memory access timing, and enhanced software control of register bank context switching.

The μPD70325 has the same complement of internal peripherals as the V25 and maintains compatibility with existing drivers; however, some modification of DMA device drivers may be necessary. The μPD70325 does not offer on-chip ROM or EPROM.

Features

- 16-bit CPU and internal data paths
- Functional and pin compatibility with V25
- Software compatible with μPD8086
- New and enhanced V-Series instructions
- 6-byte prefetch queue
- Two-channel high-speed DMA controller
- Minimum instruction cycle
  - —250 ns at 8 MHz
  - —200 ns at 10 MHz
- Internal 256-byte RAM memory
- 1-megabyte memory address space
- Eight internal memory-mapped register banks
- Four multifunction I/O ports
  - —8-bit analog comparator port
  - —20 bidirectional port lines
  - —Four 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

40

Ordering Information

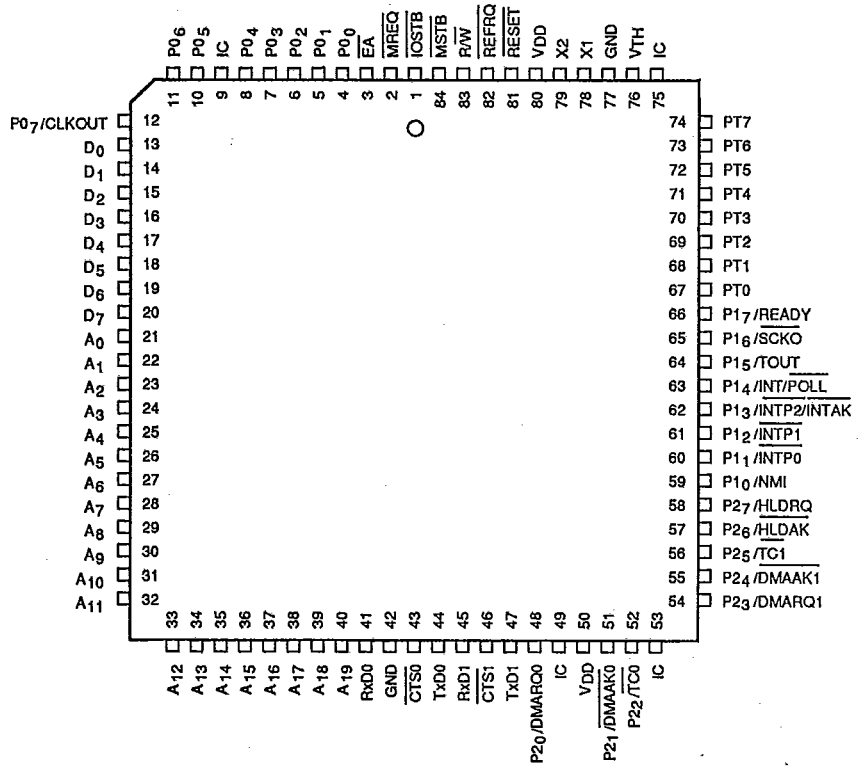
Part Number	Clock (MHz)	Package
μPD70325L-8	8	84-pin PLCC
L-10	10	
GJ-8	8	94-pin plastic QFP
GJ-10	10	

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

50134-1

Pin Configurations

84-Pin PLCC



Notes:

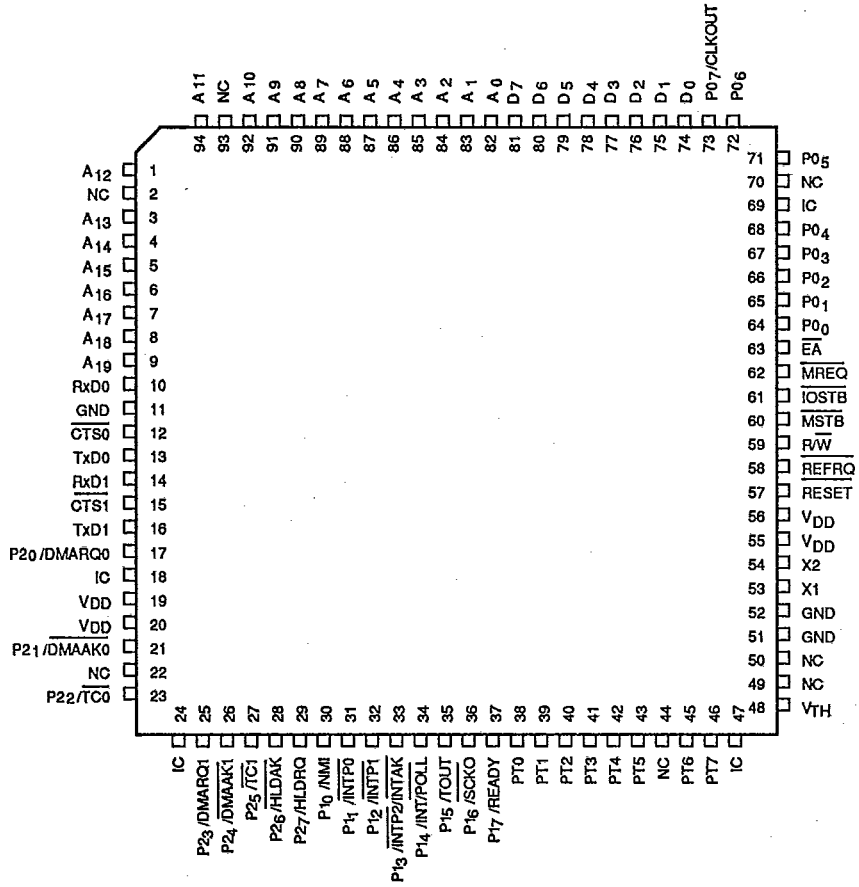
(1) Pin functions are identical to μPD70320.

(2) All 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 μPD70325 does not support internal ROM or EPROM.

83SL-6899B

94-Pin Plastic QFP



Notes:

(1) Pin functions are identical to μPD70320.

(2) All 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 μPD70325 does not support internal ROM or EPROM.

83SL-6690B

Pin Identification

Symbol	Function
A_0-A_{19}	Address bus outputs
CLKOUT	System clock output
$\overline{CTS0}$	Clear to send channel 0 input
$\overline{CTS1}$	Clear to send channel 1 input
D_0-D_7	Bidirectional data bus
$\overline{EA}$	External access
$\overline{IOSTB}$	I/O strobe output
$\overline{MREQ}$	Memory request output
$\overline{MSTB}$	Memory strobe output
$P0_0$-$P0_7$	I/O port 0
$P1_0$/NMI	Port 1 input line; nonmaskable interrupt
$P1_1$-$P1_2$/INTP0-INTP1	Port 1 input lines; external interrupt input lines
$P1_3$/INTP2/$\overline{INTAK}$	Port 1 input line; external interrupt input line; interrupt acknowledge output
$P1_4$/INT/$\overline{POLL}$	I/O port 1; interrupt request input; I/O poll input
$P1_5$/TOUT	I/O port 1; timer out
$P1_6$/$\overline{SCKO}$	I/O port 1; serial clock output
$P1_7$/READY	I/O port 1; ready input
$P2_0$/DMARQ0	I/O port 2; DMA request 0
$P2_1$/$\overline{DMAAK0}$	I/O port 2; DMA acknowledge 0
$P2_2$/$\overline{TC0}$	I/O port 2; DMA terminal count 0
$P2_3$/DMARQ1	I/O port 2; DMA request 1
$P2_4$/$\overline{DMAAK1}$	I/O port 2; DMA acknowledge 1
$P2_5$/$\overline{TC1}$	I/O port 2; DMA terminal count 1
$P2_6$/$\overline{HLDAK}$	I/O port 2; hold acknowledge output
$P2_7$/HLDRQ	I/O port 2; hold request input
PT0-PT7	Comparator port input lines
$\overline{REFRQ}$	Refresh pulse output
$\overline{RESET}$	Reset input
RxD0	Serial receive data channel 0 input
RxD1	Serial receive data channel 1 input
R/$\overline{W}$	Read/Write output
TxD0	Serial transmit data, channel 0 input
TxD1	Serial transmit data, channel 1 input
X1, X2	Crystal connection terminals
V_{DD}	Positive power supply voltage
V_{TH}	Threshold voltage input
GND	Ground reference
IC	Internal connection

PIN FUNCTIONS

A_0-A_{19} (Address Bus)

A_0-A_{19} is the nonmultiplexed 20-bit address bus used to access all external devices.

CLKOUT (System Clock)

This is the internal system clock. It can be used to synchronize external devices to the CPU.

$\overline{CTS}$n, RxDn, TxDn, $\overline{SCKO}$ (Clear to Send, Receive Data, Transmit Data, Serial Clock Out)

The two serial ports (channels 0 and 1) use these lines for transmitting and receiving data, handshaking, and serial clock output.

D_0-D_7 (Data Bus)

D_0-D_7 is the 8-bit external data bus.

DMARQn, $\overline{DMAAK}$n, TCn (DMA Request, DMA Acknowledge, Terminal Count)

These are the control signals to and from the on-chip DMA controller.

$\overline{EA}$(External Access)

If this pin is low on reset, the μPD70322 (V25) will execute program code from external memory instead of internal ROM.

Because the V25 Plus does not support internal ROM, the $\overline{EA}$ pin must be fixed low in hardware.

$\overline{HLDAK}$ (Hold Acknowledge)

The $\overline{HLDAK}$ output (active low) informs external devices that the CPU has released the system bus.

HLDRQ (Hold Request)

The HLDRQ input (active high) is used by external devices to request the CPU to release the system bus to an external bus master. The following lines go into a high-impedance status with internal 4.7-kΩ pullup resistors: A_0-A_{19}, D_0-D_7, $\overline{MREQ}$, R/$\overline{W}$, $\overline{MSTB}$, $\overline{REFRQ}$, and $\overline{IOSTB}$.

INT (Interrupt Request)

INT is a maskable, active-high, vectored interrupt request . After assertion, external hardware must provide the interrupt vector number.

The INT pin allows direct connection of slave μPD71059 interrupt controllers.

$\overline{\text{INTAK}}$ (Interrupt Acknowledge)

After INT is asserted, the CPU will respond with $\overline{\text{INTAK}}$ (active low) to inform external devices that the interrupt request has been granted.

INTP0-INTP2 (External Interrupt)

INTP0-INTP2 allow external devices to generate interrupts. Each can be programmed to be rising or falling edge triggered.

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

$\overline{\text{IOSTB}}$ is asserted during read and write operations to external I/O.

$\overline{\text{MREQ}}$ (Memory Request)

$\overline{\text{MREQ}}$ (active low) informs external memory that the current bus cycle is a memory access bus cycle.

$\overline{\text{MSTB}}$ (Memory Strobe)

$\overline{\text{MSTB}}$ (active low) is asserted during read and write operations to external memory.

NMI (Nonmaskable Interrupt)

NMI cannot be masked through software and is typically used for emergency processing. Upon execution, the interrupt starting address is obtained from interrupt vector number 2. NMI can release the standby modes and can be programmed to be either rising or falling edge triggered.

$P0_0$-$P0_7$ (Port 0)

$P0_0$-$P0_7$ are the lines of port 0, an 8-bit bidirectional parallel I/O port.

$P1_0$-$P1_7$ (Port 1)

The status of $P1_0$-$P1_3$ can be read but these lines are always control functions. $P1_4$-$P1_7$ are the remaining lines of parallel port 1; each line is individually programmable as either an input, an output, or a control function.

$P2_0$-$P2_7$ (Port 2)

$P2_0$-$P2_7$ are the lines of port 2, an 8-bit bidirectional parallel I/O port. The lines can also be used as control signals for the on-chip DMA controller.

$\overline{\text{POLL}}$ (Poll)

Upon execution of the $\overline{\text{POLL}}$ instruction, the CPU checks the status of this pin and, if low, program execution continues. If high, the CPU checks the level of the line every five clock cycles until it is low. $\overline{\text{POLL}}$ can be used to synchronize program execution to external conditions.

PT0-PT7 (Comparator Port)

PT0-PT7 are inputs to the analog comparator port.

READY (Ready)

After READY is de-asserted low, the CPU synchronizes and inserts at least two wait states into a read or write cycle to memory or I/O. This allows the processor to accommodate devices whose access times are longer than nominal μPD70325 bus cycles.

$\overline{\text{REFRQ}}$ (Refresh)

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

$\overline{\text{RESET}}$ (Reset)

A low on $\overline{\text{RESET}}$ resets the CPU and all on-chip peripherals. $\overline{\text{RESET}}$ can also release the standby modes. After $\overline{\text{RESET}}$ returns high, program execution begins from address FFFF0H.

R/$\overline{\text{W}}$ (Read/Write)

R/$\overline{\text{W}}$ output allows external hardware to determine if the current operation is a read or a write cycle. It can also control the direction of bidirectional buffers.

TOUT (Timer Out)

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

X1, X2 (Crystal Connections)

The internal clock generator requires an external crystal across these terminals. By programming the PRC register, the system clock frequency can be selected as the oscillator frequency (f_{OSC}) divided by 2, 4, or 8.

V_{DD} (Power Supply)

Two positive power supply pins (V_{DD}) reduce internal noise.

μPD70325 (V25 Plus)

V_{TH} (Threshold Voltage)

The comparator port uses this pin to determine the analog reference point. The actual threshold to each comparator line is programmable to V_{TH} x n/16 where n =1 to 16.

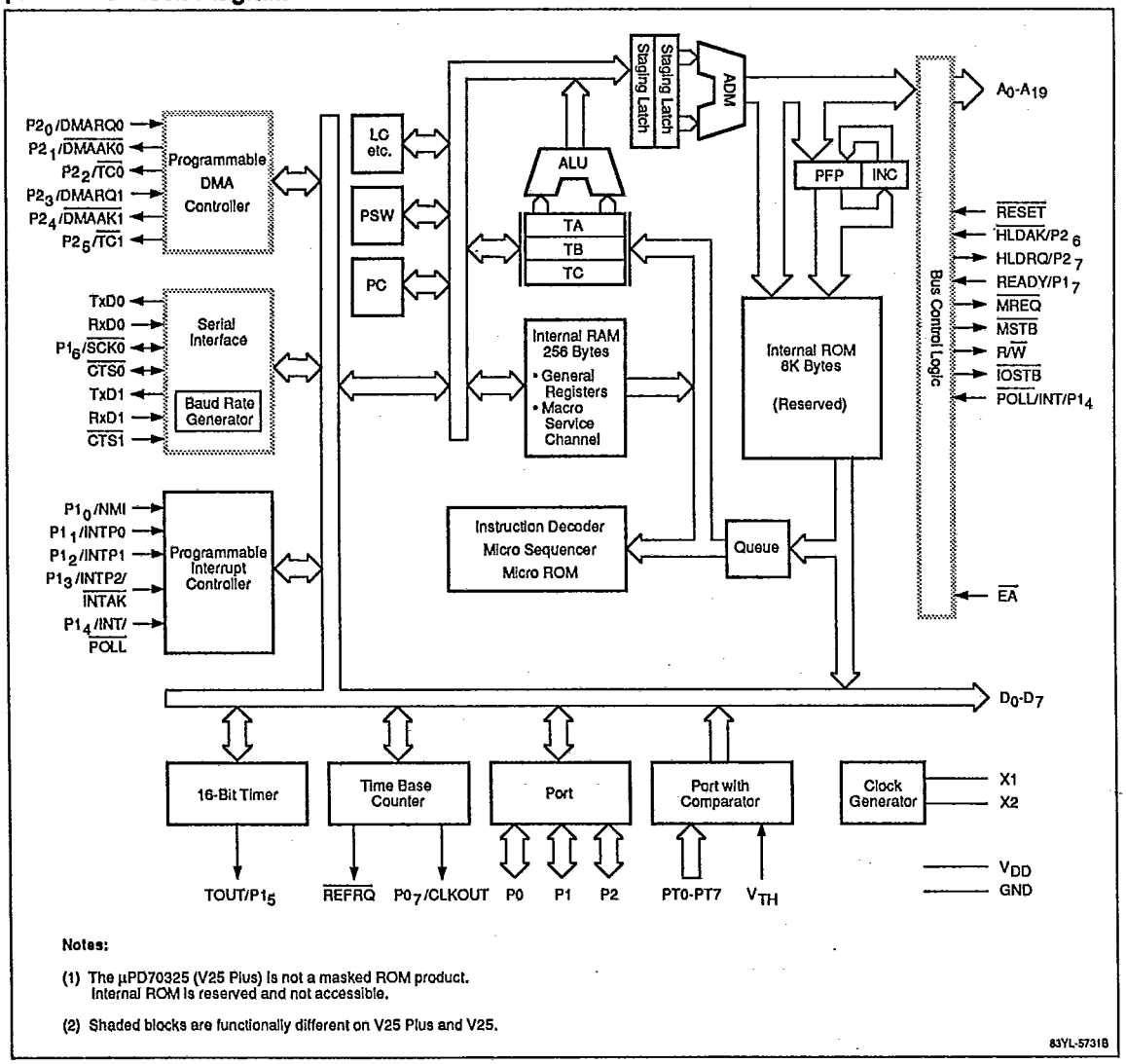
GND (Ground)

Two ground connections reduce internal noise.

IC (Internal Connection)

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

μPD70325 Block Diagram



Notes:

(1) The μPD70325 (V25 Plus) is not a masked ROM product. Internal ROM is reserved and not accessible.

(2) Shaded blocks are functionally different on V25 Plus and V25.

83YL-5731B

FUNCTIONAL DESCRIPTION

The following features enable the μPD70325 to perform high-speed execution of instructions.

- Dual internal data bus
- 16- and 32-bit temporary registers/shifters
- 16-bit loop counter
- Program counter and prefetch pointer

Dual Data Bus

The μPD70325 has two internal 16-bit data buses: the main data bus and the secondary data bus. This reduces the processing time required for addition/subtraction and logical comparison instructions by one third over single-bus systems. The dual data bus method allows two operands to be fetched simultaneously from the general-purpose registers and transferred to the ALU.

16- and 32-Bit Temporary Registers/Shifters

The 16-bit temporary registers/shifters (TA and TB) allow high-speed execution of multiplication/division and shift/rotate instructions. Using the temporary registers, the μPD70325 can execute multiplication/division instructions about four times faster than with the microprogrammed method.

Loop Counter (LC)

The dedicated hardware loop counter (LC) counts the number of iterations for string operations and the number of shifts performed for multiple-bit shift/rotate 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 by the instruction queued next. Several clock cycles are saved for branch, call, return, and break instructions.

Register Set

Figure 1 shows the eight banks of internal registers, which the μPD70325 has functionally mapped into internal RAM. Each bank contains general-purpose registers, pointer and index registers, segment registers, and save areas for context switching.

Although these memory locations may be accessed as normal RAM with the full set of memory addressing modes provided by the V25 family, the capability of context switching provides superior speed in register access. When used in the internal memory disabled state, many instructions execute considerably faster.

Eight macroservice channel control blocks are also mapped into register banks 0 and 1. The V25 Plus does not map the DMA channel control blocks into the internal RAM like the V25; instead, these control blocks are mapped into the special function register area.

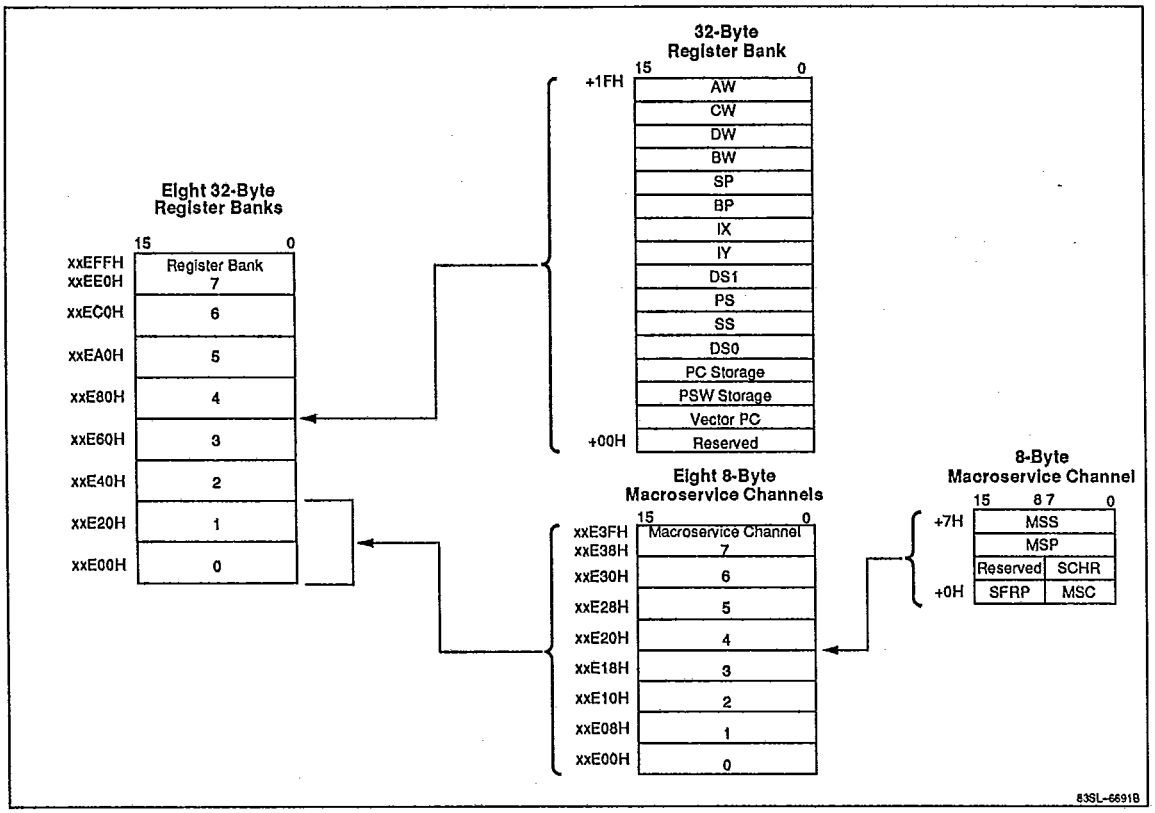
General-Purpose Registers (AW, BW, CW, DW). Four 16-bit general-purpose registers (AW, BW, CW, and DW) can serve as 16-bit registers or as four sets of dual 8-bit registers (AH, AL, BH, BL, CH, CL, DH, and DL). The instruction classes default to the following general-purpose registers.

- 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, and repeat prefixes.
- CL Shift instructions, rotate instructions, BCD operations.
- DW Word multiplication/division, indirect I/O addressing.

Pointers (SP, BP) and Index Registers (IX, IY). These registers are 16-bit base pointers (SP, BP) or index registers (IX, IY) in based addressing, indexed addressing, and based indexed addressing. They 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

Figure 1. Internal RAM Mapping



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 by four binary digits and then adding the offset address. The segment registers and default offsets are listed below.

Segment Register	Default Offset
PS (Program Segment)	PC (Program Counter)
SS (Stack Segment)	SP and Effective Address
DS0 (Data Segment 0)	IX and Effective Address
DS1 (Data Segment 1)	IY and Effective Address

Save Registers (Save PC and Save PSW). Save PC and save PSW are used as the storage areas during register bank context-switching operations. The Vector PC save location contains the effective address of the interrupt service routine when register bank switching is used to service interrupts.

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 the branch, call, return, break, or interrupt is executed.

Program Status Word (PSW). The PSW contains status and control flags used by the CPU and two general-purpose user flags. The configuration of this 16-bit register is shown below.

15							8
1	RB2	RB1	RB0	V	DIR	IE	BRK

7							0
S	Z	F1	AC	F0	P	BRKI	CY

Status Flags		Control Flags	
V	Overflow bit	DIR	Direction of string processing
S	Sign		
Z	Zero	IE	Interrupt enable
AC	Auxiliary carry	BRK	Break (after every instruction)
P	Parity		
CY	Carry	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 register AH and restored using a MOV instruction. The only way to alter the RBn bits with software is to execute an RETRBI or RETI instruction.

Memory Map

The μPD70325 has a 20-bit address bus that can directly access 1 megabyte of memory. Figure 2 shows the memory map. The internal data area (IDA) is a 256-byte internal RAM area followed consecutively by a 256-byte special function register (SFR) area.

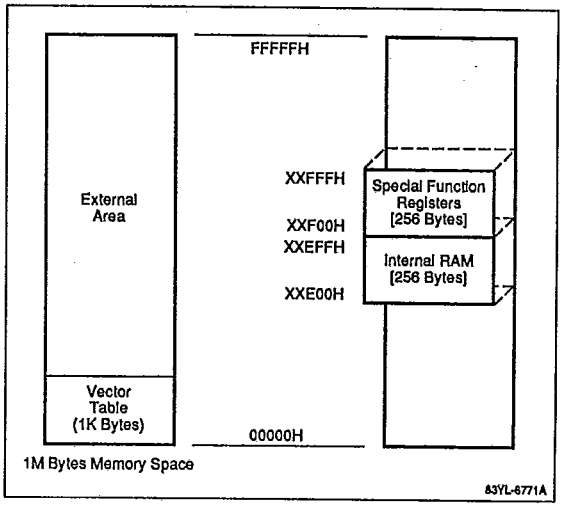
All the data and control registers for on-chip peripherals and I/O are mapped into the SFR area and accessed as RAM.

The IDA is dynamically relocatable in 4K-byte increments by changing the value in the internal data base (IDB) register. The value in this register is assigned as the uppermost eight bits of the IDA address. The IDB register is accessed from two memory locations, FFFFFH and XXFFFH, where XX is the value in the IDB register.

On reset, the internal data base register is set to FFH, which maps the IDA into the internal ROM space. However, since internal ROM is not present on the μPD70325, this does not present a problem. You can select any of the eight possible register banks, which occupy the entire internal RAM space. Multiple register bank selection allows faster interrupt processing and facilitates multitasking.

In large-scale systems where internal RAM is not required for data memory, internal RAM can be removed completely from the address space and dedicated entirely to register banks and control functions such as macroservice. You do this by clearing the RAMEN bit in the processor control register. When the RAMEN bit is cleared, internal RAM can only be accessed by register addressing or internal control processes. Many instructions execute faster when internal RAM is disabled.

Figure 2. Memory Map



INSTRUCTIONS

The μPD70325 instruction set is fully compatible with the V20 native mode instruction set. The V25 Plus is a superset of the μPD8086/8088 instruction set with different execution times and mnemonics.

The μPD70325 does not support the V20 8080 emulation mode.

Enhanced Instructions

In addition to the μPD8086/8088 instructions, the μPD70325 provides the following enhanced instructions.

Instruction	Description
PUSH imm	Pushes immediate data onto stack
PUSH R	Pushes 8 general registers onto stack
POP R	Pops 8 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 data
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 μPD70325 provides the following unique instructions.

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

Variable-Length Bit Field Operation Instructions

Bit fields are a variable-length data structure that can range from 1 to 16 bits. The μPD70325 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 IY and reg8 are updated to point to the start of the next bit field.

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

Figures 3 and 4 further illustrate bit field insertion and bit field extraction, respectively.

Packed BCD Instructions

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

Bit Manipulation Instructions

The μPD70325 provides five unique bit manipulation instructions that allow you to test, set, clear, or complement a single bit in a register or memory operand. This increases code readability as well as performance over the logical operations traditionally used to manipulate bit data. These instructions also give you additional control over on-chip peripherals.

Figure 3. Bit Field Insertion

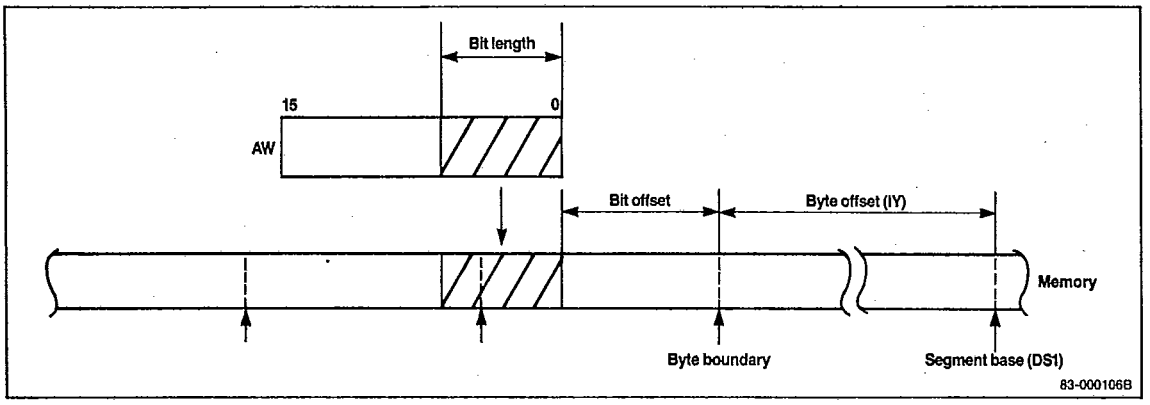
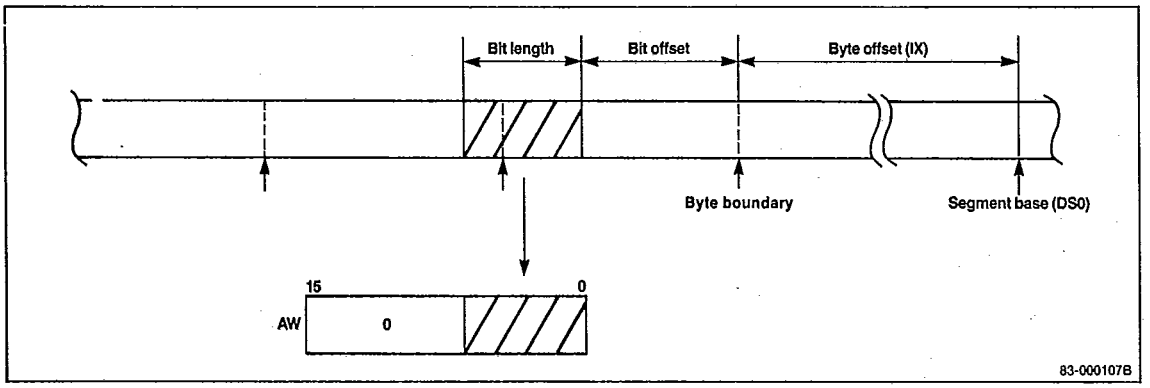


Figure 4. Bit Field Extraction



4d

Additional Instructions

Besides the V20 instruction set, the μPD70325 provides the following additional instructions.

Instruction	Description
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. Not for use with NMI or INT interrupt service routines.

Repeat Prefixes

Two repeat prefixes (REPC and 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 following instructions allow the effective use of the register banks for software interrupts and multitasking.

Instruction	Description
BRKCS reg 16	Performs a high-speed software interrupt with context switch to register bank indicated by lower 3 bits of register 16.
TSKSW reg 16	Performs a high-speed task switch to register bank indicated by lower 3 bits of register 16. The PC and PSW are saved in the old banks. PS and PSW save registers and the new PC and PSW values are retrieved from the new register bank's save areas.
MOVSPA	Transfers both SS and SP of old register bank to new register bank after bank has been switched by an interrupt or BRKCS instruction.
MOVSPB reg16	Transfers SS and SP of current register bank before switching to SS and SP of new register bank indicated by lower 3 bits of register 16.

INTERRUPT STRUCTURE

The μPD70325 can service interrupts generated by both hardware and software. Software interrupts are serviced through vectored interrupt processing. The following interrupts are provided.

Interrupt	Description
Divide error	The CPU traps 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	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 generates an interrupt if the specified array bounds are exceeded.
Escape trap	The CPU traps in an FP01,2 instruction to allow software to emulate the floating-point processor since the μPD70325 does not support an external hardware coprocessor.
I/O trap	If the I/O trap bit in the PSW is cleared, a trap is generated on every IN or OUT instruction. Software can then provide an updated peripheral address. This feature provides software portability between different systems.

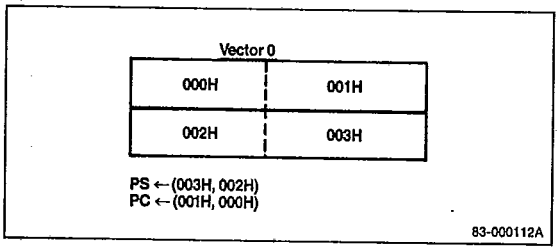
When executing software written for another system, it is better to implement I/O with on-chip peripherals to reduce external hardware requirements. However, since μPD70325 internal peripherals are memory mapped, software conversion may be difficult. The I/O trap feature allows for easy conversion from external peripherals to on-chip peripherals.

Interrupt Vectors

Table 1 lists the interrupt vectors beginning at physical address 00H. External memory is required to service these routines. By servicing interrupts via the macroservice function or context switching, this requirement can be eliminated.

Each interrupt vector is 4 bytes wide. To service a vectored interrupt, the lower addressed word is transferred to the PC and the upper word to the PS. See figure 5.

Figure 5. Interrupt Vector 0



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

Table 1. Interrupt Vectors

Address	Vector	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

Hardware Interrupt Configuration

The V25 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 17 interrupt sources 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.

Be careful when assigning the priority of a given interrupt group; the assignment is done by the three priority bits in only one interrupt control register in each group. If interrupts from different groups occur simultaneously and the groups have the same priority level, the priority is as shown in table 2.

Table 2. Interrupt Sources

Group	Interrupt Source (Priority Within Group) 1	2	3	Default Priority
Nonmaskable interrupt	NMI	–	–	0
Timer unit	INTTU0	INTTU1	INTTU2	1
DMA controller	INTD0	INTD1	–	2
External peripheral interrupt	INTP0	INTP1	INTP2	3
Serial channel 0	INTSER0	INTSR0	INTST0	4
Serial channel 1	INTSER1	INTSR1	INTST1	5
Time base counter	INTTB	–	–	6
Interrupt request	INT	–	–	7

The priority of the currently active interrupt is stored in the ISPR special function register. Bits PR_7-PR_0 correspond to the eight possible interrupt request priorities. The ISPR keeps track of the priority of active interrupts by setting the appropriate bit of this register. The address of this 8-bit register is xxFFCH, and the format is shown below.

PR_7	PR_6	PR_5	PR_4	PR_3	PR_2	PR_1	PR_0

NMI and INT are system type external vectored interrupts. NMI is not maskable via software, and is also recognized by the μPD70325 during DMA demand-release transfer. INT is maskable by the IE bit in the PSW and requires that an external device provide the interrupt vector number. It is designed to allow the interrupt controller to be expanded by the addition of an external interrupt controller such as the μPD71059.

NMI, INTP0-INTP1 are edge-sensitive inputs. By selecting the appropriate bits in the interrupt mode register, these inputs can be programmed to be either rising- or falling-edge triggered. Bits ES0-ES2 correspond to INTP0-INTP2, respectively, as shown in figure 6.

Figure 6. External Interrupt Mode Register (INTM)

0	ES2	0	ES1	0	ES0	0	ESNMI
7			Address xxF40H				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

The five external interrupts are:

NMI	Nonmaskable interrupt
INT	Cascaded PIC interrupt
INTP0	Interrupt from peripheral 0
INTP1	Interrupt from peripheral 1
INTP2	Interrupt from peripheral 2

The twelve internal interrupts are:

INTTU0	Timer unit 0 interrupt
INTTU1	Timer unit 1 interrupt
INTTU2	Timer unit 2 interrupt
INTD0	DMA channel 0 interrupt
INTD1	DMA channel 1 interrupt
INTSER0	Serial channel 0 error interrupt
INTSR0	Serial channel 0 receive interrupt
INTST0	Serial channel 0 transmit interrupt
INTSER1	Serial channel 1 error interrupt
INTSR1	Serial channel 1 receive interrupt
INTST1	Serial channel 1 transmit interrupt
INTTB	Time base counter interrupt

Table 3 shows the various interrupt request control registers, the options for service, their relative priorities, and the options for multiple control.

Table 3. Interrupt Processing

Interrupt Source	Interrupt Vector	Macro Service	Bank Switching	Priority Setting	Priority Between Groups	Priority Within Group	Multiple Process Control
NMI	2	No	No	Not available	0	—	Not accepted
INT	External	No	No	Not available	7	—	Not accepted
INTTU0	28	Yes	Yes	Available	1	1	Accepted
INTTU1	29					2	
INTTU2	30					3	
INTD0	20	No	Yes	Available	2	1	
INTD1	21					2	
INTP0	24	Yes	Yes	Available	3	1	
INTP1	25					2	
INTP2	26					3	
INTSER0	12	No	Yes	Available	4	1	
INTSR0	13	Yes				2	
INTST0	14	Yes				3	
INTSER1	16	No	Yes	Available	5	1	
INTSR1	17	Yes				2	
INTST1	18	Yes				3	
INTTB	31	No	No	Not available	6	—	

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 vector method (compatible with V20/V30), register bank context switching (supported in hardware), and macroservice (SFR transfers). The processing mode for a given interrupt can be chosen by enabling the appropriate bits in the corresponding interrupt request control register. Each interrupt, except INT and NMI, has its own associated IRC register. The general format for each of these registers is shown in figure 7.

All interrupt processing routines other than those for NMI and INT must end with the execution of the FINT instruction. This instruction informs the interrupt controller that the current interrupt service routine is complete; if FINT is not executed within the service routine, subsequently only interrupts of higher priority will be accepted.

In the vectored service mode, the CPU traps to a vector location.

Figure 7. 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)

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 has the same bank number (0-7) as the priority programmed in the associated IRC register. The PC and PSW are automatically sorted in the save areas of the new register bank, and the address of the interrupt routine is loaded from the vector PC storage register in the new register bank.

As in the vectored mode, the IE and BRK bits in the PSW are cleared to zero. After processing, execution of the RETRBI instruction must be executed to return control to the original register bank and restore the former PC and PSW. Figures 8 and 9 show register bank context switching and register bank return.

This method of interrupt service offers a dramatic performance advantage over normal vectored service because there is no need to store and retrieve data/registers on the stack. This also allows hardware-based real-time task switching in high-speed environments.

In addition to context switching, the μPD70325 has a task switch opcode (TSKSW) that allows multiple independent processes to be internally resident. Figure 10 shows the task switching function.

Figure 8. Register Bank Context Switching

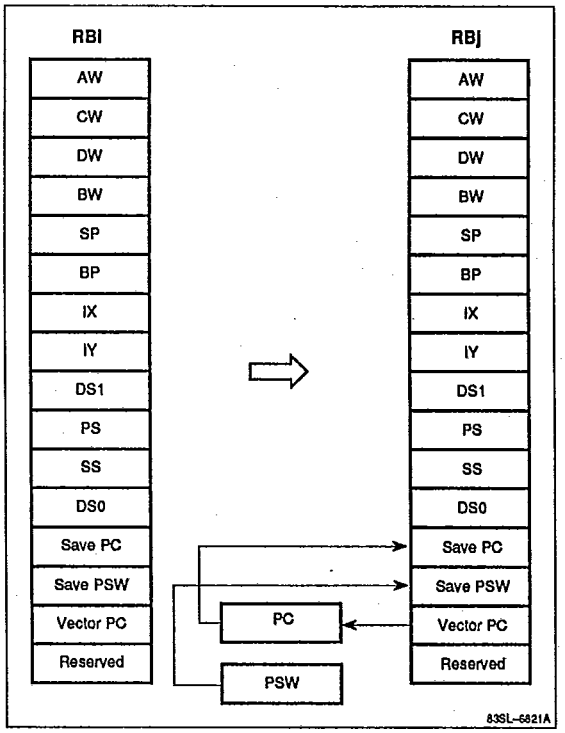


Figure 9. Register Bank Return

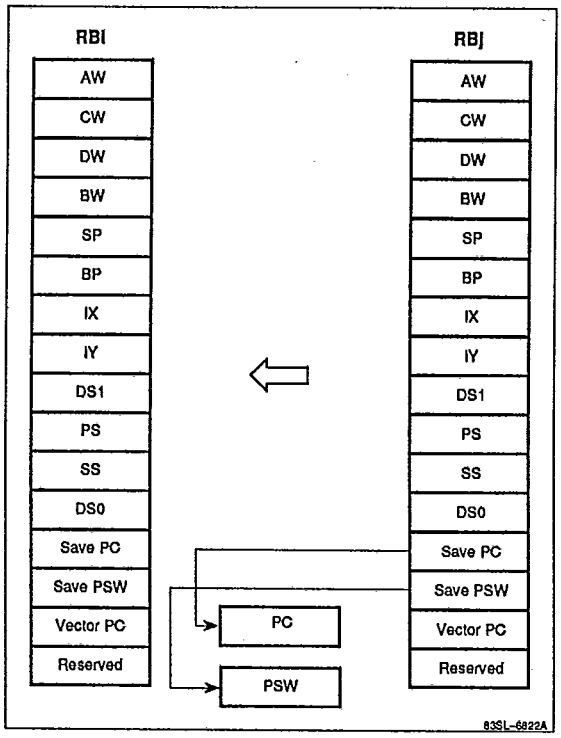
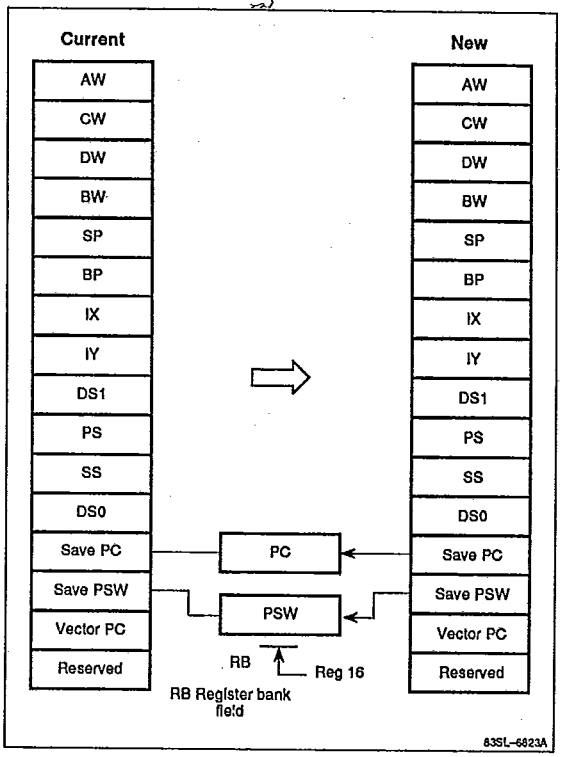


Figure 10. Task Switching



Interrupt Factor Register

The μPD70325 provides an additional register that stores the interrupt vector number of the last-serviced interrupt request. The register is located in special-function memory and is read only in 8-bit operations. This register facilitates the use of one register bank to service multiple interrupt sources, particularly those within the same group (interrupts within the same group will all context switch to the same register bank).

The interrupt vector is stored in the IRQS register as shown in figure 11, and is retained until the next interrupt request is accepted. The value of the IRQS register is not altered by NMI, INT, or macroservice transfers. It is generally recommended that the IRQS register be read before the EI bit is set within the interrupt service routine to assure that its contents will not be altered by multiple processing routines.

Figure 11. Interrupt Factor Register (IRQS)

0	0	0	Interrupt Vector	
7		5	4 Address xxFEFH	0

Interrupt Factor	Interrupt Vector
INTTU0	1CH
INTTU1	IDH
INTTU2	IEH
INTD0	14H
INTD1	15H
INTP0	18H
INTP1	19H
INTP2	1AH
INTSER0	0CH
INTSR0	0DH
INTST0	0EH
INTSER1	10H
INTSR1	11H
INTST1	12H
INTTB	1FH

Macroservice Function

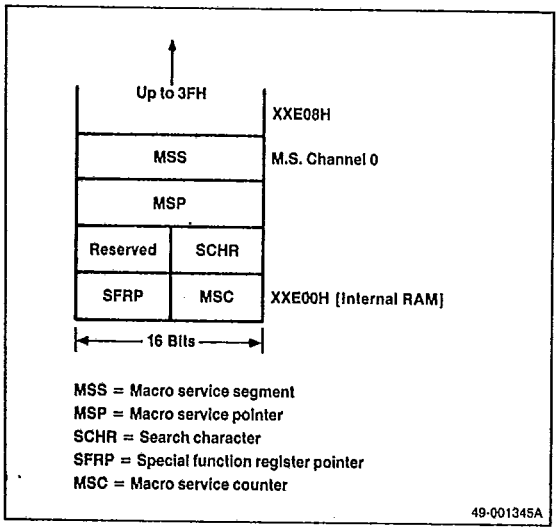
The macroservice function (MSF) is a special microprogram that acts as an internal DMA controller between on-chip peripheral special-function registers and memory. The MSF greatly reduces the software overhead and CPU time that other processors would require for register save processing, register returns, 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 is transferred between an 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 is compared to an 8-bit search character and an interrupt is generated if a match occurs or if the macroservice counter reaches zero.

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

Eight 8-byte macroservice channels are mapped into internal RAM from XXE00H to XXE3FH. Each macroservice channel contains all necessary information to execute the macroservice process. Figure 12 shows the components of each channel.

Figure 12. Macroservice Channels



Setting the macroservice mode requires programming the corresponding macroservice control register. Each individual interrupt, excluding INT, NMI, serial error, DMA, and TBC, has its own associated register. Figure 13 shows the generic format for all MSC registers.

Figure 13. 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 μPD70325 (figure 14) has two programmable 16-bit interval timers (TM0 and 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 and MD1). Timer 0 operates in the interval timer mode or one-shot mode; timer 1 has only the interval timer mode.

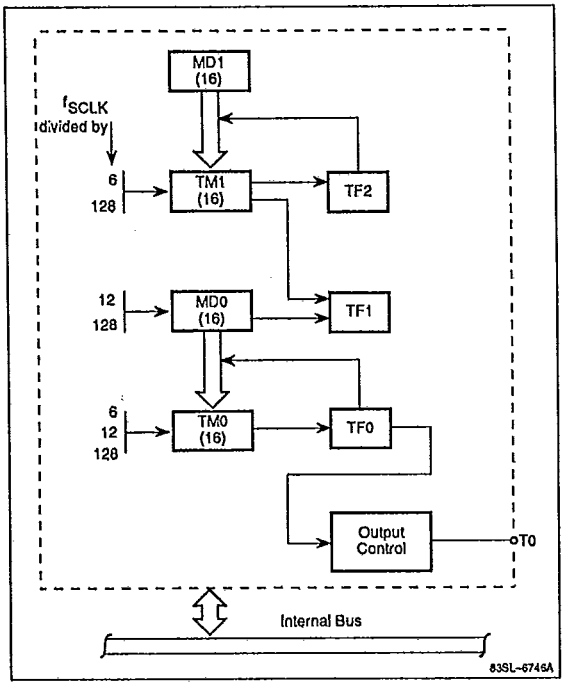
Interval Timer Mode

In this mode, TM0/TM1 are decremented by the selected input clock, and, after counting out, the registers are automatically reloaded from the modulus registers and counting continues. Each time TM1 counts out, interrupts are generated through TF1 and TF2 (timer flags 1 and 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.

Two input clocks derived from the system clock are SCLK/6 and SCLK/128. Typical timer values shown below are based on f_{OSC} = 10 MHz and $f_{SCLK} = f_{OSC}/2$.

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

Figure 14. Timer Unit Block Diagram



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).

When TM0 is programmed to one-shot mode, TM1 may still operate in interval mode.

Two input clocks derived from the system clock are SCLK/12 and SCLK/128. Typical timer values shown below are based on f_{OSC} = 10 MHz and $f_{SCLK} = f_{OSC}/2$.

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 15 and 16 for the TMC register format.

Figure 15. Timer Control Register 0 (TMC0)

TS0	TCLK0	MS0	MCLK0	ENT0	ALV	MOD_1	MOD_0
7			Address xxF90H				0

TS0	TM0 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 When TOUT Disabled by ENT0 = 0
0	Low
1	High

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

Figure 16. Timer Control Register 1 (TMC1)

TS1	TCLK1	0	0	0	0	0	0
7			Address xxF91H				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 the four interrupt periods can be selected by programming the TB0 and TB1 bits in the processor control register (PRC). The TBC interrupt is unlike the others because it is fixed as a level 7 vectored interrupt. Macroservice and register bank switching cannot be used to service this interrupt. See figures 17 and 18.

Figure 17. 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 18. 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

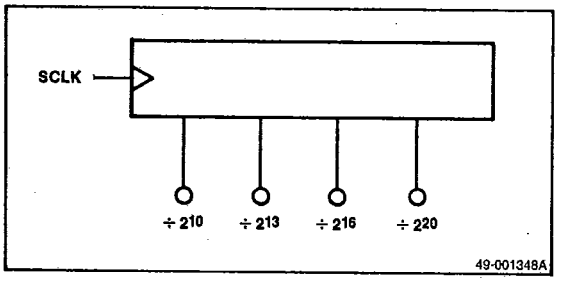
TB_1	TB_0	Time Base Interrupt Period
0	0	$2^{10}/f_{CLK}$
0	1	$2^{13}/f_{CLK}$
1	0	$2^{16}/f_{CLK}$
1	1	$2^{20}/f_{CLK}$

PCK_1	PCK_0	System Clock Frequency (f_{CLK})
0	0	$f_X/2$
0	1	$f_X/4$
1	0	$f_X/8$
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 19) uses the system clock as the input frequency. The system clock can be changed by programming the PCK0 and PCK1 bits in the processor control register (PRC). Reset initializes the system clock to $f_{OSC}/8$ (f_{OSC} = external oscillator frequency).

Figure 19. Time Base Counter (TBC) Block Diagram



REFRESH CONTROLLER

The μPD70325 has an on-chip refresh controller for dynamic and pseudostatic RAM memory. The refresh controller generates refresh cycles between the normal CPU bus cycles according to the refresh specifications programmed.

The refresh controller outputs a 9-bit refresh address on address bits A_8-A_0 during the refresh bus cycle. Address bits A_{19}-A_9 are fixed to 0s during this cycle. The 9-bit refresh address is automatically incremented at every refresh cycle for 512 row addresses. The 8-bit refresh mode (RFM) register (figure 20) specifies the refresh operation and allows refresh during both CPU HALT and HOLD modes. Refresh cycles are automatically timed to $\overline{\text{REFRQ}}$ following read/write cycles to minimize the effect on system throughput.

As shown in figure 20, the $\overline{\text{REFRQ}}$ output level is determined by the by the RFLV and RFEN bits of the RFM register.

Figure 20. Refresh Mode Register (RFM)

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

RFLV	RFEN	$\overline{REFRQ}$ 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

SERIAL INTERFACE

The μPD70325 has two full-duplex UARTs, channels 0 and 1. Each channel has a transmit line (TxDn), a receive line (RxDn), and a clear-to-send (CTSn) handshaking line. Communication is synchronized by a start bit, and either even, odd, or no parity may be programmed. Character length may be programmed to either 7 or 8 bits, and either 1 or 2 stop bits may be selected.

Each serial channel of the μPD70325 has a dedicated baud rate generator, so there is no need to obligate any of the on-chip timers to handle this function. The baud rate generators allow individual transfer rates for each channel and support rates up to 1.25 Mb/s. All standard baud rates are available and are not restricted by the value of the particular external crystal.

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

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

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

where:

B = baud rate
G = baud rate generator register (BRGn) value
n = input clock specification; the value loaded into the SCCn register ($0 < n < 8$)
SCLK = system clock frequency (MHz)

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

Baud Rate	n	BRGn	Error (%)
110	7	178	0.25
150	7	130	0.16
300	6	130	0.16
600	5	130	0.16
1200	4	130	0.16
2400	3	130	0.16
4800	2	130	0.16
9600	1	130	0.16
19.2k	0	130	0.16
38.4k	0	65	0.16
1.25M	0	2	0.00

In addition to the asynchronous mode, channel 0 has a synchronous I/O interface mode. In this mode, each bit of data transferred is synchronized to a serial clock (SCLK0). The receive clock may be specified as either the internal baud rate generator output or an external signal provided on the $\overline{CTS0}$ input pin (the RSCK bit of the SCM0 register must be programmed to 0).

This mode is functionally equivalent to using the serial channel as a shift register because data is synchronous with the SCLK signal. Data bits from consecutive bytes may directly follow one another since no extra bits (parity, start, or stop) are added. This mode is compatible with the μCOM75 and μCOM87 series, and allows direct interfacing to these devices.

Figure 21 details the serial communication mode register, which controls the operational mode and data format of the serial channel. The serial communication control register shown in figure 22 specifies the baud rate generator input clock frequency.

Figure 21. Serial Communication Mode Registers (SCM)

TxRDY	RxB	PRTY1	PRTY0	CLTSK	SLRSCK	MD1	MD0
7							0

TxRDY	Transmitter Control
0	Disabled
1	Enabled
RxB	**Receiver Control**
0	Disabled
1	Enabled
PRTY1-PRTY0	**Parity Control**
0 0	No parity
0 1	0 parity (Note 1)
1 0	Odd parity
1 1	Even parity
CLTSK	**Character Length (Async Mode) Tx Shift Clock (I/O Interface Mode)**
0	7 bits (Async) No effect (I/O intfc)
1	8 bits (Async) Trigger transmit (I/O intfc)
SLRSCK	**Stop Bits (Async Mode) Receiver Clock (I/O Interface Mode)**
0	1 stop bit (Async) Ext clock input on CTS0 (I/O intfc)
1	2 stop bits (Async) Int clock output on CTS1 (I/O intfc)
MD1-MD0	**Mode**
0 0	I/O Interface (Note 2)
0 1	Asynchronous
1 x	Reserved

Notes:

(1) Parity is 0 during transmit and ignored during receive.

(2) Channel only.

The serial communication error registers of the V25 are replaced in the μPD70325 with the serial status registers shown in figure 23. These registers provide error flags and buffer status information. The error bits are automatically cleared when the next data byte is received; otherwise, these flags are persistent.

Figure 22. Serial Communication Control Register (SCC)

0	0	0	0	PRS_3	PRS_2	PRS_1	PRS_0
7							0

PRS_3-PRS_0	Baud Rate Generator Input Clock Frequency
0000	$f_{SCLK}/2$ (n = 0)
0001	$f_{SCLK}/4$
0010	$f_{SCLK}/8$
0011	$f_{SCLK}/16$
0100	$f_{SCLK}/32$
0101	$f_{SCLK}/64$
0110	$f_{SCLK}/128$
0111	$f_{SCLK}/256$
1000	$f_{SCLK}/512$ (n = 8)

The TxBE and RxBF bits signal the status of the respective transmit and receive buffers. These bits are reset automatically when either the baud rate generator or serial control register contents are written. The AS (All Sent) bit is set when both the transmit buffer and the transmit shift register are empty.

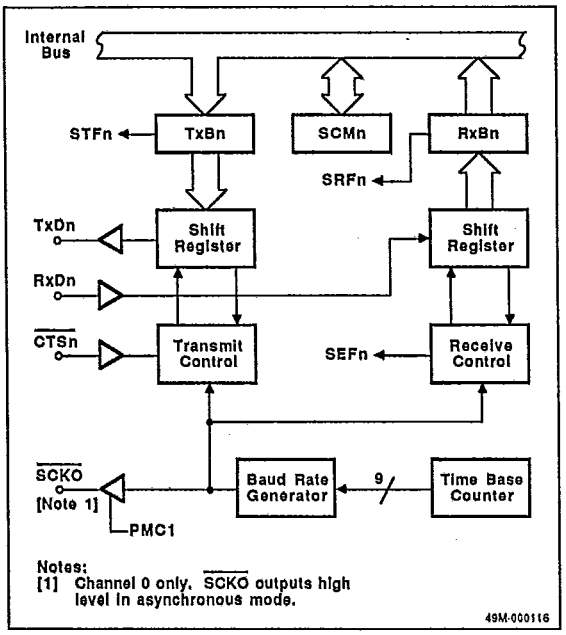
Figure 23. Serial Status Register (SST)

RxDn	AS	TxBE	RxBF	0	ERP	ERF	ERO
7							0

RxDn	Receive Terminal State
0, 1	Status of RxD pin
AS	**All Sent Flag**
0	Data has been written in transmit buffer
1	Data in transmit buffer and shift register has been sent
TxBE	**Transmit Buffer Empty Flag**
0	Data has been written in transmit buffer
1	Data in buffer has been sent to shift register
RxBF	**Receive Buffer Full Flag**
0	Data has been read from receive buffer
1	Data has been sent from shift register to buffer
ERP	**Parity Error Flag**
0	No error
1	Transmit and receive parity are different
ERF	**Framing Error Flag**
0	No error
1	Stop bit not detected
ERO	**Overrun Error Flag**
0	No error
1	Data is received before receive buffer outputs previous data

Figures 24 and 25 show the serial interface block diagram for asynchronous and I/O interface modes, respectively.

Figure 24. Serial Channels 0 and 1; Asynchronous Mode



DMA CONTROLLER

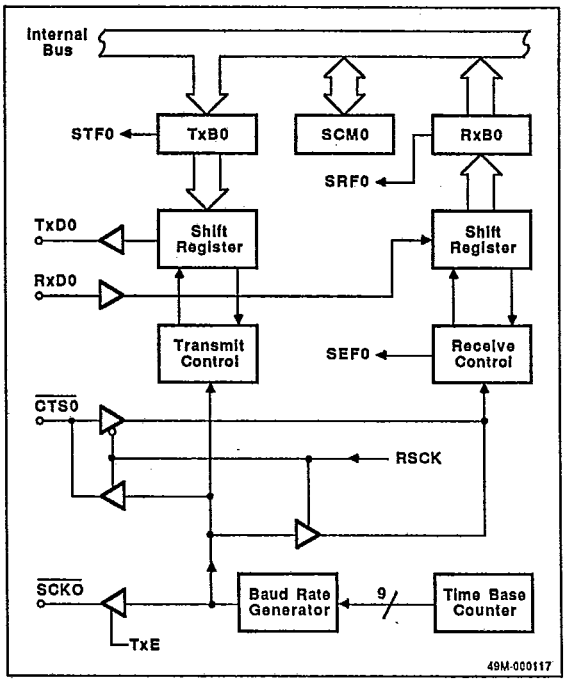
The μPD70325 has an on-chip, two-channel DMA controller capable of supporting data transfer at full bus bandwidth. Although the operating modes of the μPD70325 DMA unit are identical to those of the V25, the programming and data transfer rates are different.

Six I/O pins support the operation of the DMA unit.

- Two active-high request inputs
- Two active-low acknowledge outputs valid only in I/O-to-memory and memory-to-I/O transfer modes
- Two terminal count active-low outputs, driven low when the transfer count register is decremented from 0 (borrow occurs).

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.

Figure 25. Serial Channel 0; I/O Interface Mode



Memory-to-Memory DMA Transfers

Single-Step Mode. The single-step mode allows direct memory access for memory-to-memory transfers. These transfers take two bus cycles (nominally two clock cycles each), one to read the data and the other to write it back to memory. The data is stored inside the μPD70325 in a dedicated temporary register between the two bus cycles.

When a DMA request occurs in this mode, DMA and CPU bus cycles alternate until the transfer count is reached. This mode is provided to allow the user program to continue executing at 50% of its normal speed when DMA is operational. The single request thus generates a full block of data transfer; that is, until terminal count is reached.

Burst Mode. This mode is also a memory-to-memory transfer running at full bus bandwidth. Once a request is recognized, the DMA unit takes control of the bus and continuously transfers data until the terminal count is reached for that channel. This mode forces all other lower priority bus masters (including the CPU) to hold their bus requests until the specified DMA block is transferred.

Table 4. DMA Unit Functional Interaction

	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 time of DMA request, specified times of DMA transmission are executed.	One DMA transmission 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	None	Depends on software	Halted at low level during DMA transmission
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 times of DMA transmission are executed consecutively	Same as usual	Same as usual
DMA request during DMA transmission	DMA at channel 1 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.

I/O-to-Memory DMA Transfers

I/O-to-memory (source synchronized) and memory-to-I/O (destination synchronized) transfers are performed in one bus cycle (nominally two clock states). These transfers drive a single memory address and strobe the IORD or IOWR and $\overline{\text{DMAAK}}$ signals. Thus the appropriate I/O device and memory location are selected without the processor driving the data bus. These transfers are designed for high-speed, I/O data transfer reaching 5 megabytes/ second at 10 MHz.

Single Transfer. Single-transfer mode responds with one DMA transfer per rising edge of the DMA request line. After one DMA cycle is performed, control is transferred back to the CPU until another request is recognized. Transfers continue, one-per-request, until the terminal count is reached.

Care must be taken in the single-transfer mode to adhere to the t_{WIQH} and t_{WIQL} specifications. Although the DMA request is latched internally, if the above values are not met, the μPD70325 may not detect the following rising edge of the request input, and the subsequent DMA transfer will not take place.

Demand Release. Demand release mode continues to perform DMA transfers until either terminal count is reached or the DMA request is removed. When the DMARQ is removed during DMA operation, the DMA channel finishes the current transfer and releases the bus.

Note that the t_{WIQH} and t_{WIQL} parameters must also be taken into design consideration for the demand release mode. The t_{WIQH} parameter insures that the DMA request is held past the internal sampling point. When the $\overline{\text{DMAAK}}$ signal is asserted, the μPD70325 has obviously accepted the request. Thus, the t_{WIQH} parameter may not need to be fully satisfied; that is, the request may be removed after the $\overline{\text{DMAAK}}$ signal has been asserted. This should not present a problem because it is specified that the request must be active until the acknowledge is output.

Demand Release Termination. The demand release mode may be released in one of two ways: the programmed terminal count is reached; or the DMARQ signal is removed. These two conditions differ considerably in operational characteristics.

When terminating demand release DMA by the terminal count method, there are no restrictions placed on the transfer bandwidth, and the transfer rate will maintain 4 megabytes/second at 8 MHz or 5 megabytes/second at 10 MHz. This rate is achieved by programming the DMA to operate at zero wait states and by sustaining the DMARQ until the terminal count signal is asserted.

If demand release mode DMA is terminated by deassertion of the DMARQ line, then several specifications are at issue. The $\overline{DMAAK}$ signal is triggered from the falling clock edge in the middle of B1 with a propagation delay of 80 ns max. The DMA request line is sampled on the next rising edge of the CPU clock, which is ideally 62.5 ns later (minus the 10-ns DMARQ setup time). Therefore, on a worst-case device, the $\overline{DMAAK}$ signal is not asserted until after the μPD70325 has already sampled the DMARQ line. If $\overline{DMAAK}$ is used to negate DMARQ, the request may not be sampled correctly. This situation may be avoided by inserting wait states in the DMA transfer to delay the DMARQ sampling point.

A second issue is the internal propagation delay of the $\overline{DMAAK}$ signal. If the DMARQ input is gated with the $\overline{DMAAK}$ signal in a zero wait-state system, $\overline{DMAAK}$ is not held low long enough to allow internal timing signals to fully propagate through the device.

The result of the above situation must be broken into two cases: zero wait-state systems and two wait-state systems. The zero wait-state system using DMARQ to terminate the transfer may "overrun." This would produce one more DMA transfer than the desired number (unless terminal count was reached on the transfer during which DMARQ was removed).

The two wait-state system will function normally in the above situation since the wait states assure that the DMARQ line will be sampled and accepted, thereby terminating the transfer without any slippage. In this case, the t_{WIQL} specification may be taken at 3T (min), the width of the $\overline{DMAAK}$ pulse.

Addresses. All DMA addresses are mapped to external memory space on the μPD70325. When a location corresponding to an internal data area is addressed, external memory with the same address will be accessed.

Priority. Bus hold and refresh control maintain higher bus priority than the DMA unit and are active during DMA transmission. If an HLDRQ or $\overline{REFRQ}$ signal is presented during DMA operation, the DMA unit releases its $\overline{DMAAK}$ signal and relinquishes control of the bus to the requesting bus master. DMA transmission is also temporarily halted during an interrupt acknowledge cycle.

HALT State. The μPD70325 will acknowledge a DMA request while the CPU is in the HALT state, and the processor will return to HALT after the transfer is completed. When the DMA terminal count interrupt is presented in the HALT state, the processor releases the HALT state and returns to normal mode.

Latency. The μPD70325 DMA latency time is substantially faster than the standard V25. Since the DMA controller is hardwired, it responds to requests within instruction execution. The resulting worst-case latency is 14+2W clocks from request to acknowledge and this typical value will be 4 system clocks.

DMA Registers

The μPD70325 DMA registers differ from those on the standard V25. The control blocks for each channel are located in the special function register area and thus do not overlap the internal RAM register banks.

DMA mode registers are provided for each DMA channel. These registers, shown in figure 26, specify the transmission mode, byte/word data size, and the enable state of each DMA channel.

Figure 26. DMA Mode Registers (DMAM)

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	Reserved
100	Burst (memory to memory)
101	Single-transfer (I/O to memory)
110	Single-transfer (memory to I/O)
111	Reserved

W	Transfer Method
0	Byte transfer
1	Word transfer

EDMA	TDMA	Transfer Condition
0	0	Disabled
1	0	Maintain condition
1	1	Start DMA transfer

The μPD70325 performs two consecutive 8-bit transfers to accomplish word data transfers with the lower addressed byte transferred first. Upon reaching terminal count, the DMA channel is automatically disabled by the EDMA bit of the mode register. A DMA request to a disabled channel is ignored. The DMA transfer bit (TDMA) has meaning only in the single-step and burst modes and allows software to initiate the DMA operation. This software request is valid only when the channel is enabled. The TDMA bit is write only and is always read as a 0.

The DMA address control registers (figure 27) specify the method of address pointer update. Two bits specify the source address and two specify the destination address update mode. The μPD70325 adjusts the address pointers by 1 for byte transfers and by 2 for word transfers.

Figure 27. DMA Address Control Registers (DMAC)

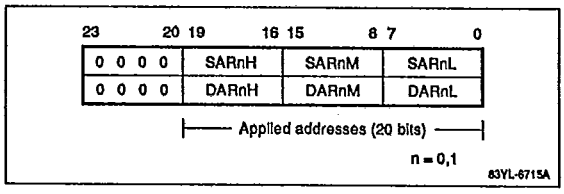
0	0	PD_1	PD_0	0	0	PS_1	PS_0
7							0

PD_1-PD_0	Destination Address Offset
0 0	No modification
0 1	Increment
1 0	Decrement
1 1	No modification
PS_1-PS_0	**Source Address Offset**
0 0	No modification
0 1	Increment
1 0	Decrement
1 1	No modification

Figure 28 shows the address pointer registers that specify the source and destination of the DMA transfer. Unlike the standard V25, the address registers are linear. This allows the 20-bit address to be completely specified in three byte-wide registers. The SARnL and DARnL registers contain the low 8 bits of the address; SARnM and DARnM contain the middle 8 bits; and SARnH and DARnH contain the high 4 bits of the address in the low nibble (the high nibble is set to 0).

All of these registers may be read or written with either 8- or 16-bit transfers and are updated as specified in the DMAC registers.

Figure 28. DMA Address Registers



The terminal count registers (TCnH and TCnL) are dual 8-bit registers that hold the current number of transfers remaining in the DMA block. These registers are read/write in 8- or 16-bit operations. They must be initially programmed to the desired number of transfers minus one. This is because the terminal count interrupt is generated by a borrow out of these registers; this borrow is generated by the decrement performed after each DMA transfer.

DMA Timing

DMA operation on the μPD70325 is considerably faster than on the standard V25. This speedup is realized by converting the DMA operation from a microcoded process to a hardwired one. As a result, the DMA latency times on the standard V25 do not occur on the μPD70325. However, bus controller latency is still present as is the nominal transfer time: 1 bus cycle for I/O-to-memory transfers and 2 bus cycles for memory-to-memory transfers.

Programmable wait state control is active for DMA operations with the programmed number of states added to both source and destination addresses even if these numbers are different. Memory-to-I/O transfers insert the number of wait states required by either the I/O device or the memory location (whichever is slower). I/O-to-memory transfers insert the memory wait states for the memory write cycle.

Figures 29 to 32 are examples of cycles for DMA operations. Figure 33 is a block diagram of the μPD70325 DMA controller and its internal registers.

Figure 29. DMA Single-Step Mode Timing

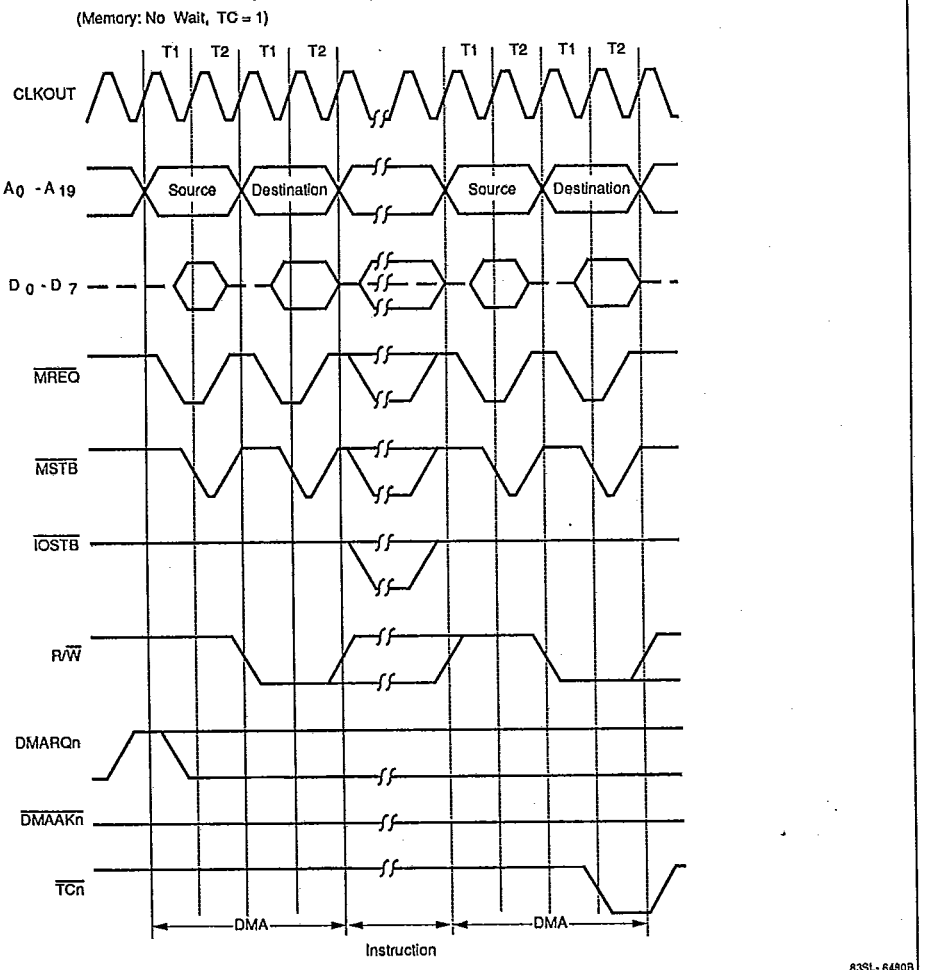
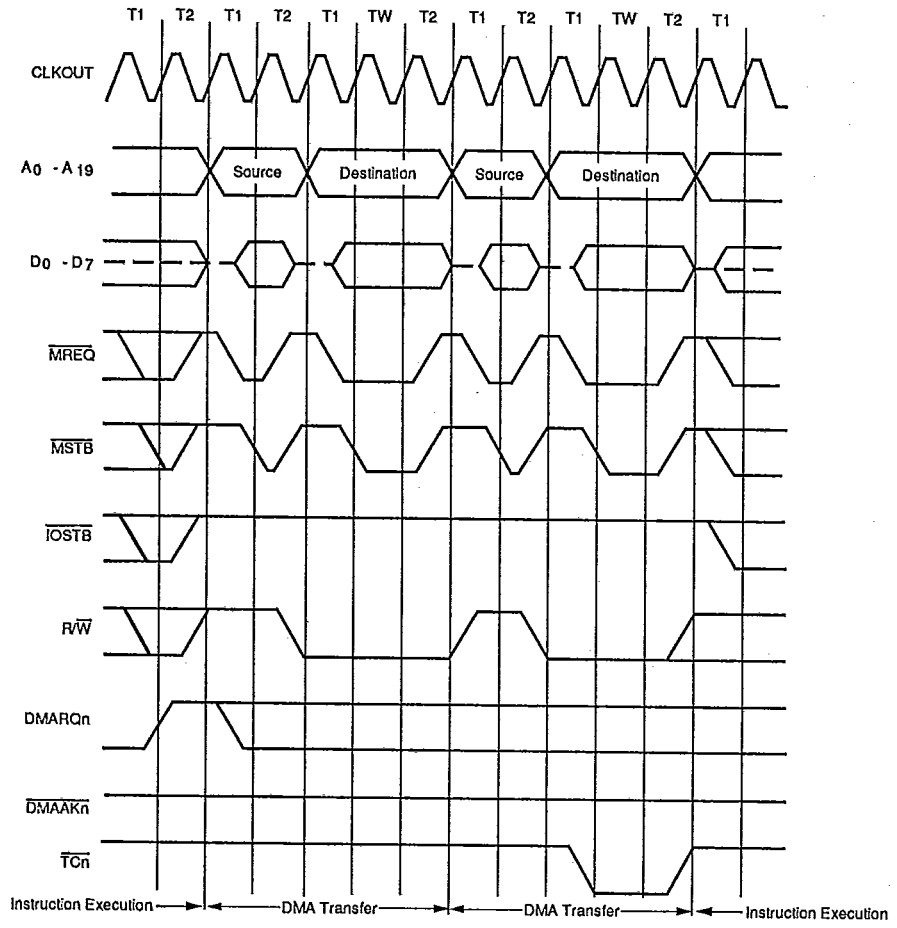


Figure 30. DMA Burst Mode Timing



Burst mode with no-wait for transfer source memory bank and insertion of one wait state for transfer destination when DMA is started by DMARQ signal when TC=1.

83SL-6478B

Figure 31. DMA Single-Transfer Mode Timing

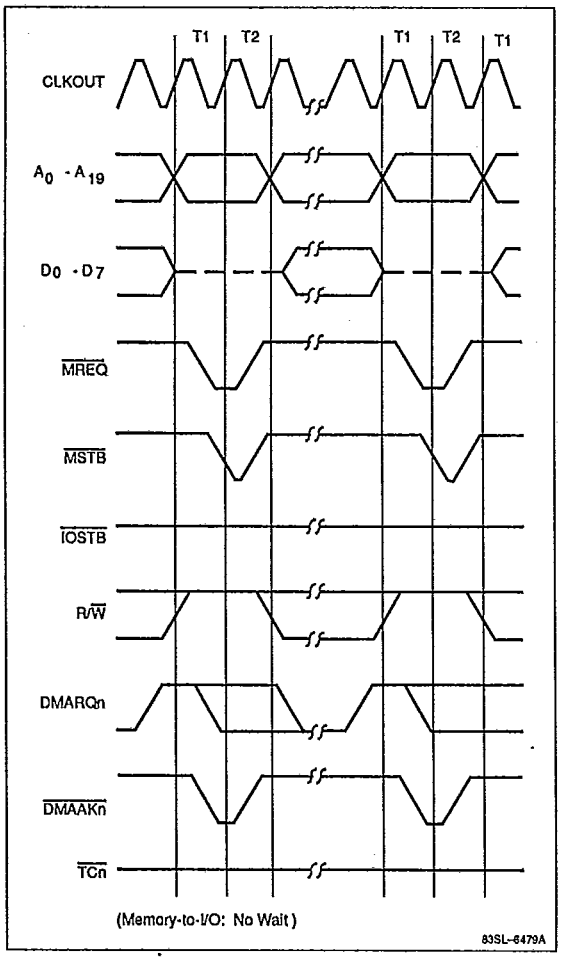


Figure 32. DMA Demand Release Timing

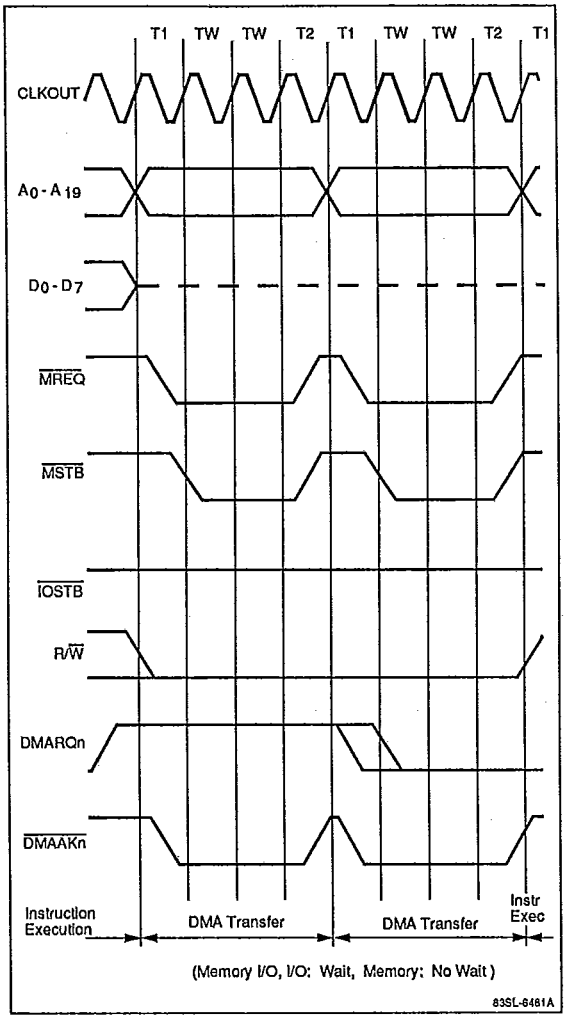
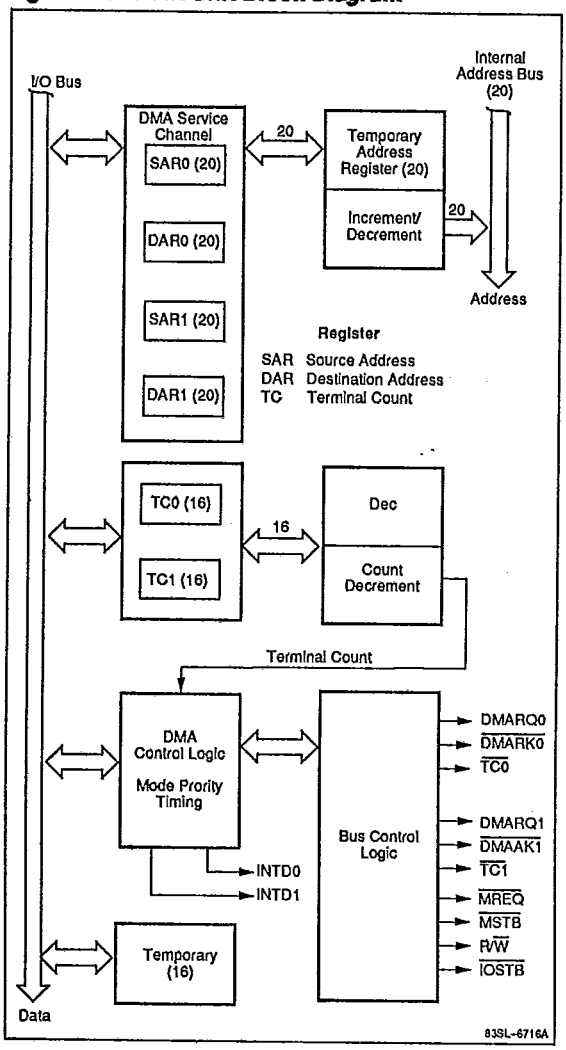


Figure 33. DMA Unit Block Diagram



PARALLEL I/O PORTS

The μPD70325 has three 8-bit parallel I/O ports: P0, P1, and P2. Associated registers are shown in figures 34, 35, and 36. Special-function register (SFR) locations can access these ports as memory. 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 control (PMC) and port mode (PM) registers to select the function for a given I/O line.

Figure 34. Port 0 Registers (PMC0, PM0)

$PMC0_7$	0	0	0	0	0	0	0
7			PMC0 Register				0

$PM0_7$	$PM0_6$	$PM0_5$	$PM0_4$	$PM0_3$	$PM0_2$	$PM0_1$	$PM0_0$
7			PM0 Register				0

		$PMC0_n = 0$	
Port Pin	$PMC0_7 = 1$	$PM0_n = 1$	$PM0_n = 0$
$P0_7$	CLKOUT	Input port	Output port
$P0_6$	—	Input port	Output port
$P0_5$	—	Input port	Output port
$P0_4$	—	Input port	Output port
$P0_3$	—	Input port	Output port
$P0_2$	—	Input port	Output port
$P0_1$	—	Input port	Output port
$P0_0$	—	Input port	Output port

Figure 35. Port 1 Registers (PMC1, PM1)

$PMC1_7$	$PMC1_6$	$PMC1_5$	$PMC1_4$	$PMC1_3$	0	0	0
7			PMC1 Register				0

$PM1_7$	$PM1_6$	$PM1_5$	$PM1_4$	1	1	1	1
7			PM1 Register				0

		$PMC1_n = 0$	
Port Pin	$PMC1_7 = 1$	$PM1_n = 1$	$PM1_n = 0$
$P1_7$	READY Input	Input port	Output port
$P1_6$	$\overline{SCKO}$ output	Input port	Output port
$P1_5$	TOUT output	Input port	Output port
$P1_4$	INT Input	$\overline{POLL}$ Input	Output port
$P1_3$	$\overline{INTAK}$ output	INTP2 Input	—
$P1_2$	—	INTP1 Input	—
$P1_1$	—	INTP0 Input	—
$P1_0$	—	NMI Input	—

Figure 36. Port 2 Registers (PMC2, PM2)

$PMC2_7$	$PMC2_6$	$PMC2_5$	$PMC2_4$	$PMC2_3$	$PMC2_2$	$PMC2_1$	$PMC2_0$
7			PMC2 Register				0

$PM2_7$	$PM2_6$	$PM2_5$	$PM2_4$	$PM2_3$	$PM2_3$	$PM2_2$	$PM2_1$	$PM2_0$
7			PM2 Register					0

		$PMC2_n$ = 0	
Port Pin	$PMC2_n$ = 1	$PM2_n$ = 1	$PM2_n$ = 0
$P2_7$	HLDRQ Input	Input port	Output port
$P2_6$	$\overline{HLDAK}$ output	Input port	Output port
$P2_5$	$\overline{TC1}$ output	Input port	Output port
$P2_4$	$\overline{DMAAK1}$ output	Input port	Output port
$P2_3$	DMARQ1 Input	Input port	Output port
$P2_2$	$\overline{TC0}$ output	Input port	Output port
$P2_1$	$\overline{DMAAK0}$ output	Input port	Output port
$P2_0$	DMARQ0 Input	Input port	Output port

The analog comparator port (PT) compares each input line to a reference voltage. The reference voltage can be programmed to the V_{TH} input x n/16, where n = 1 to 16. See figure 37.

Figure 37. Port T Mode Register (PMT)

0	0	0	0	PMT_3	PMT_2	PMT_1	PMT_0
7							0

Comparator Reference Voltage (V_{REF})	PMT_3	PMT_2	PMT_1	PMT_0
V_{TH} x 16/16	0	0	0	0
1/16	0	0	0	1
2/16	0	0	1	0
3/16	0	0	1	1
4/16	0	1	0	0
5/16	0	1	0	1
6/16	0	1	1	0
7/16	0	1	1	1
8/16	1	0	0	0
9/16	1	0	0	1
10/16	1	0	1	0
11/16	1	0	1	1
12/16	1	1	0	0
13/16	1	1	0	1
14/16	1	1	1	0
15/16	1	1	1	1

PROGRAMMABLE WAIT STATES

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 1 megabyte of 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 sets the wait-state conditions for the entire I/O address space. Figure 38 shows the memory map for programmable wait-state generation.

Figure 39 diagrams the wait control word. Note that READY pin control is enabled only when two internally generated wait states are selected by the "11" option.

Figure 38. Programmable Wait State Generation

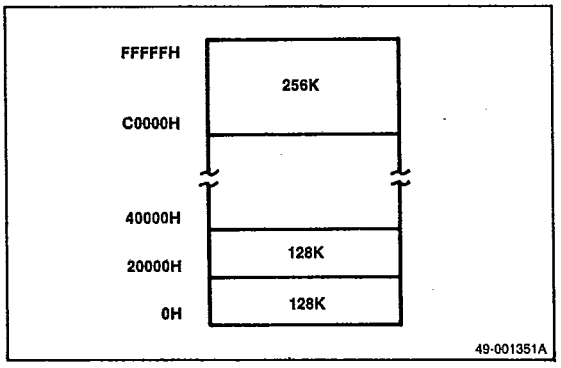


Figure 39. Wait Control Word (WTC)

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 READYpin)	1	1

n = 0 thru 6

STANDBY MODES

The two low-power standby modes are HALT and STOP. Both modes are entered under software control.

HALT Mode

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

STOP Mode

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

A standby flag in the SFR area is reset by rises in the supply voltage. Its status is maintained during normal operation and standby. The STBC register (figure 40) is not initialized by $\overline{\text{RESET}}$. Use the standby flag to determine whether program execution is returning from standby or from a cold start by setting this flag before entering STOP mode.

Figure 40. Standby Register (STBC)

0	0	0	0	0	0	0	SBF
7			Address xxFE0H				0

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

SPECIAL-FUNCTION REGISTERS

Table 5 lists the special-function registers . The 8 high-order bits of each register address (denoted by. xx in table 5) is specified by the IDB register. This allows the special function register bank to be dynamically relocated in memory.

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

The Read/Write column in table 5 shows the legal data movement operations used to address these registers. Data size can be specified as: bit (1), byte (8), and/or word (16).

Table 5. Special-Function Registers

Address	Register Name	Label	Read/Write	Reset Condition
xxF00H	Port 0 data	P0	R/W 8/1	Undefined
xxF01H	Port 0 mode	PM0	W 8	0FFH
xxF02H	Port 0 control	PMC0	W 8	00H
xxF08H	Port 1 data	P1	R/W 8/1	Undefined
xxF09H	Port 1 mode	PM1	W 8	0FFH
xxF0AH	Port 1 control	PMC1	W 8	00H
xxF10H	Port 2 data	P2	R/W 8/1	Undefined
xxF11H	Port 2 mode	PM2	W 8	0FFH
xxF12H	Port 2 control	PMC2	W 8	00H
xxF38H	Threshold port data	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 macroservice control channel 0 (Note 4)	EMS0	R/W 8/1	Undefined
xxF45H	External interrupt macroservice control channel 1 (Note 4)	EMS1	R/W 8/1	Undefined
xxF46H	External interrupt macroservice control channel 2 (Note 4)	EMS2	R/W 8/1	Undefined
xxF4CH	External interrupt request control 0 (Note 4)	EXIC0	R/W 8/1	47H
xxF4DH	External interrupt request control 1 (Note 4)	EXIC1	R/W 8/1	47H
xxF4EH	External interrupt request control 2 (Note 4)	EXIC2	R/W 8/1	47H
xxF60H	Serial receive buffer channel 0	RxB0	R 8	Undefined
xxF62H	Serial transmit buffer channel 0	TxB0	W 8	Undefined
xxF65H	Serial receive macroservice register 0 (Note 4)	SRMS0	R/W 8/1	Undefined
xxF66H	Serial transmit macroservice register 0 (Note 4)	STMS0	R/W 8/1	Undefined
xxF68H	Serial mode register 0	SCM0	R/W 8/1	00H
xxF69H	Serial control register 0	SCC0	R/W 8/1	00H
xxF6AH	Baud rate generator 0	BRG0	R/W 8/1	00H
xxF6BH	Serial status register 0	SCS0	R 8	60H
xxF6CH	Serial error interrupt request register 0 (Note 4)	SEIC0	R/W 8/1	47H
xxF6DH	Serial receive interrupt request register 0 (Note 4)	SRIC0	R/W 8/1	47H
xF6EH	Serial transmit interrupt request register 0 (Note 4)	STIC0	R/W 8/1	47H
xxF70H	Serial receive buffer 1	RxB1	R 8	Undefined
xxF72H	Serial transmit buffer 1	TxB1	W 8	Undefined
xxF75H	Serial receive macroservice register 1 (Note 4)	SRMS1	R/W 8/1	Undefined
xxF76H	Serial transmit macroservice register 1 (Note 4)	STMS1	R/W 8/1	Undefined
xxF78H	Serial mode register 1	SCM1	R/W 8/1	00H
xxF79H	Serial control register 1	SCC1	R/W 8/1	00H
xxF7AH	Baud rate generator channel 1	BRG1	R/W 8/1	00H
xxF7BH	Serial status register 1	SCS1	R 8	60H
xxF7CH	Serial error interrupt request register 1 (Note 4)	SEIC1	R/W 8/1	47H

Table 5. Special-Function Registers (cont)

Address	Register Name	Label	Read/Write	Reset Condition
xxF7DH	Serial receive interrupt request register 1 (Note 4)	SRIC1	R/W 8/1	47H
xxF7EH	Serial transmit interrupt request register 1 (Note 4)	STIC1	R/W 8/1	47H
xxF80H	Timer register 0 (Note 5)	TM0	R/W 16	Undefined
xxF82H	Timer 0 modulo register (Note 5)	MD0	R/W 16	Undefined
xxF88H	Timer register 1 (Note 5)	TM1	R/W 16	Undefined
xxF8AH	Timer 1 modulo register (Note 5)	MD1	R/W 16	Undefined
xxF90H	Timer 0 control register (Note 5)	TMC0	R/W 8/1	00H
xxF91H	Timer 1 control register (Note 5)	TMC1	R/W 8/1	00H
xxF94H	Timer unit 0 macroservice register (Note 4)	TMMS0	R/W 8/1	Undefined
xxF95H	Timer unit 1 macroservice register (Note 4)	TMMS1	R/W 8/1	Undefined
xxF96H	Timer unit 2 macroservice register (Note 4)	TMMS2	R/W 8/1	Undefined
xxF9CH	Timer unit 0 interrupt request register (Note 4)	TMIC0	R/W 8/1	47H
xxF9DH	Timer unit 1 interrupt request register (Note 4)	TMIC1	R/W 8/1	47H
xxF9EH	Timer unit 2 interrupt request register (Note 4)	TMIC2	R/W 8/1	47H
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 4)	DIC0	R/W 8/1	47H
xxFADH	DMA interrupt request control register 1 (Note 4)	DIC1	R/W 8/1	47H
xxFC0H	DMA channel 0 source address pointer low	SAR0L	R/W 16/8	Undefined
xxFC1H	DMA channel 0 source address pointer mid	SAR0M	R/W 16/8	Undefined
xxFC2H	DMA channel 0 source address pointer high	SAR0H	R/W 8	Undefined
xxFC4H	DMA channel 0 destination address pointer low	DAR0L	R/W 16/8	Undefined
xxFC5H	DMA channel 0 destination address pointer mid	DAR0M	R/W 16/8	Undefined
xxFC6H	DMA channel 0 destination address pointer high	DAR0H	R/W 8	Undefined
xxFC8H	DMA channel 0 count register	TC0	R/W 16/8	Undefined
xxFD0H	DMA channel 1 source address pointer low	SAR1L	R/W 16/8	Undefined
xxFD1H	DMA channel 1 source address pointer mid	SAR1M	R/W 16/8	Undefined
xxFD2H	DMA channel 1 source address pointer high	SAR1H	R/W 8	Undefined
xxFD4H	DMA channel 1 destination address pointer low	DAR1L	R/W 16/8	Undefined
xxFD5H	DMA channel 1 destination address pointer mid	DAR1M	R/W 16/8	Undefined
xxFD6H	DMA channel 1 destination address pointer high	DAR1H	R/W 8	Undefined
xxFD8H	DMA channel 1 terminal count register	TC1	R/W 16/8	Undefined
xxFE0H	Standby control register (Notes 1, 2)	STBC	R/W 8/1	Undefined
xxFE1H	Refresh mode register	RFM	R/W 8/1	0FCH
xxFE8H	Wait state control register	WTC	R/W 16/8	0FFFFH
xxFEAH	User flag register (Note 3)	FLAG	R/W 8/1	00H

Table 5. Special-Function Registers (cont)

Address	Register Name	Label	Read/Write	Reset Condition
xxFEBH	Processor control register	PRC	R/W 8/1	4EH
xxFECH	Time base interrupt request control register (Note 4)	TBIC	R/W 8/1	47H
xxFEFH	Interrupt factor register (Note 4)	IRQS	R 8	Undefined
xxFFCH	Interrupt priority control register (Note 4)	ISPR	R 8	00H
xxFFFH	Internal data area base address register (Note 4)	IDB	R/W 8/1	0FFH

Notes:

(1) Standby control register can be set by instruction but not cleared.

(2) At power-on reset: 00H. Otherwise not changed.

(3) Bit operations on FLAG register bits other than 3 and 5 have no meaning.

(4) One wait state is inserted in accesses to these registers.

(5) A maximum of six wait states are added into accesses to these registers.

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 ≤ +7.0 V
Output voltage, V_O	−0.5 to V_{DD}+0.5 ≤ +7.0 V
Threshold voltage, V_{TH}	−0.5 to V_{DD}+0.5 ≤ +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.

Comparator Characteristics

T_A = −10 to +70°C; V_{DD} = +5.0 V ± 10%

Parameter	Symbol	Min	Max	Unit
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

T_A = 25°C; V_{DD} = 0 V

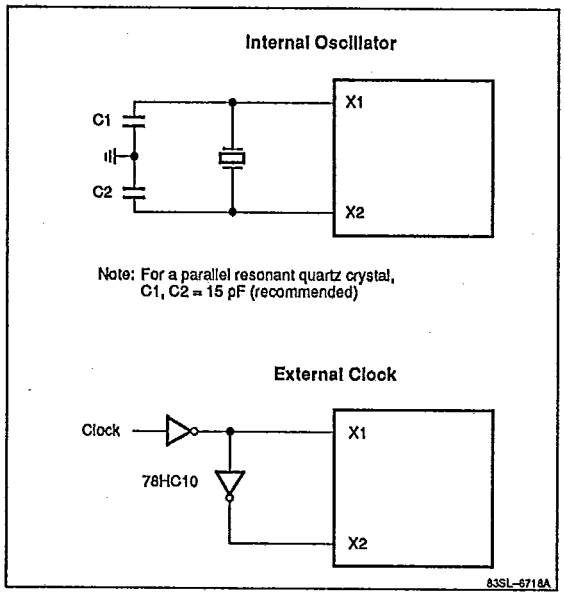
Parameter	Symbol	Max	Unit	Conditions
Input capacitance	C_I	10	pF	fc = 1 MHz; unmeasured pins returned to ground
Output capacitance	C_O	20	pF	
I/O capacitance	C_{IO}	20	pF	

DC Characteristics; μPD70325-8

T_A = −10 to +70°C; V_{DD} = +5.0 V ±10%

Parameter	Symbol	Min	Typ	Max	Unit	Conditions
Input voltage, low	V_{IL}	0		0.8	V	
Input voltage, high	V_{IH1}	2.2		V_{DD}	V	All except $\overline{RESET}$, $P1_0$/NMI, X1, X2
	V_{IH2}	0.8 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; 0≤ V_{IN} ≤ V_{DD}
Input leakage current	I_{LI}			±10	μA	All except $\overline{EA}$, $P1_0$/NMI; 0 ≤ V_{IN} ≤ V_{DD}
Output leakage current	I_{LO}			±10	μA	0 ≤ V_O ≤ V_{DD}
V_{TH} supply current	I_{TH}		0.5	1.0	mA	0 ≤ V_{TH} ≤ V_{DD}
V_{DD} supply current	I_{DD1}		65	120	mA	Operation mode
	I_{DD2}		25	50	mA	HALT mode
	I_{DD3}		10	30	uA	STOP mode

External System Clock Control Source



Recommended Oscillator Components

Ceramic Resonator (Note 1)		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
Crystal (Note 2)		**Capacitors**	
Manufacturer	**Product No.**	**C1 (pF)**	**C2 (pF)**
Kinseki	HC-49/U	15	15
	HC-43/U	15	15

Notes:

(1) Ceramic resonator product no. includes the frequency: 10.0 or 16.0 MHz.

(2) Crystal frequencies: 10, 16, 20 MHz.

AC Characteristics; μPD70325-8

T_A = −10 to +70°; V_{DD} = +5.0 V ±10%; C_L = 100 pF max; T = t_{CYK}; n = number of wait states inserted

Parameter	Symbol	Min	Max	Unit	Conditions
Input rise, fall times	t_{IR}, t_{IF}		20	ns	Except X1, X2, RESET, NMI
Input rise, fall times (Schmitt)	t_{IRS}, t_{IFS}		30	ns	RESET, NMI
Output rise, fall times	t_{OR}, t_{OF}		20	ns	Except CLKOUT
X_1 cycle time	t_{CYX}	62	250	ns	
X_1 width, low	t_{WXL}	20		ns	
X_1 width, high	t_{WXH}	20		ns	
X_1 rise, fall times	t_{XR}, t_{XF}		20	ns	
CLKOUT cycle time	t_{CYK}	125	2000	ns	CLKOUT = $f_X/2$
CLKOUT width, low	t_{WKL}	0.5T − 15		ns	
CLKOUT width, high	t_{WKH}	0.5T − 15		ns	
CLKOUT rise, fall times	t_{KR}, t_{KF}		15	ns	
Address delay time	t_{DKA}	15	90	ns	
Address valid to input data valid	t_{DADR}		(n+1.5)T − 70	ns	
MREQ to data delay time	t_{DMRD}		(n+1)T − 60	ns	
MSTB to data delay time	t_{DMSD}		(n+0.5)T − 60	ns	
MREQ to TC delay time	t_{DMRTC}		0.5T + 50	ns	
MREQ to MSTB delay time	t_{DMRMS}	0.5T − 35	0.5T + 35	ns	
MREQ width, low	t_{WMRL}	(n+1)T − 30		ns	
Address hold time	t_{HKA}	0.5T − 30		ns	
Input data hold time	t_{HKDR}	0		ns	
Next control setup time	t_{SCC}	T − 25		ns	
TC width, low	t_{WTCL}	(n+2)T − 30		ns	
Address data output	t_{DADW}		0.5T + 50	ns	
MREQ delay time	t_{DAMR}	0.5T − 30		ns	
MSTB delay time	t_{DAMS}	T − 30		ns	
MSTB width, low	t_{WMSL}	(n+0.5)T − 30		ns	
Data output setup time	t_{SDM}	(n+1)T − 50		ns	
Data output hold time	t_{HMDW}	0.5T − 30		ns	
IOSTB delay time	t_{DAIS}	0.5T − 30		ns	
IOSTB to data input	t_{DISD}		(n+1)T − 60	ns	
IOSTB width, low	t_{WISL}	(n+1)T − 30		ns	
Address hold time	t_{HISA}	0.5T − 30		ns	
Data input hold time	t_{HISDR}	0		ns	
Output data setup time	t_{SDIS}	(n+1)T − 50		ns	
Output data hold time	t_{HISDW}	0.5T − 30		ns	
Next DMARQ setup time	t_{SDADQ}		T − 50	ns	Demand mode
DMARQ hold time	t_{HDARQ}	0		ns	Demand mode
DMAAK read width, low	t_{WDMRL}	(n+1.5)T − 30		ns	
DMAAK to TC delay time	t_{DDATC}		0.5T + 50	ns	
DMAAK write width, low	t_{WDMWL}	(n+1)T − 30		ns	

AC Characteristics; μPD70325-8 (cont)

Parameter	Symbol	Min	Max	Unit	Conditions
$\overline{REFRQ}$ delay time	t_{DARF}	0.5T – 30		ns	
$\overline{REFRQ}$ width, low	t_{WRFL}	(n+1)T – 30		ns	
Address hold time	t_{HRFA}	0.5T – 30		ns	
$\overline{RESET}$ width, low	t_{WRSL1}	30		ms	Crystal osc; STOP/Power-On RESET
$\overline{RESET}$ width, low	t_{WRSL2}	5		μs	System reset
$\overline{MREQ}$, $\overline{IOSTB}$ to READY setup time	t_{SCRY}		(n – 1)T – 100	ns	n ≥ 2
$\overline{MREQ}$, $\overline{IOSTB}$ to READY hold time	t_{HCRY}	(n – 1)T		ns	n ≥ 2
$\overline{HLDAK}$ 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 control output time	t_{DHQC}	3T + 30		ns	
$\overline{HLDAK}$ width, low	t_{WHAL}	T		ns	
HLDRQ setup time	t_{SHQK}	30		ns	
HLDRQ to $\overline{HLDAK}$ delay time	t_{DHQHA}		3T +160	ns	
HLDRQ width, low	t_{WHQL}	1.5T		ns	
INTP, DMARQ setup time	t_{SIQK}	30		ns	
INTP, DMARQ width, high	t_{WIQH}	8T		ns	
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	
INT hold time	t_{HIAQ}	0		ns	
$\overline{INTAK}$ width, low	t_{WIAL}	2T – 30		ns	
$\overline{INTAK}$ delay time	t_{DKIA}	15	80	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	
$\overline{SCKO}$ 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	
$\overline{CTS0}$ (RSCK) width, low	t_{WSRL}	420		ns	
RxD setup time	t_{SRDK}	80		ns	
RxD hold time	t_{HKRD}	80		ns	

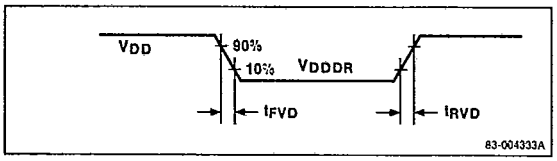
STOP Mode Data Retention Characteristics

T_A = −10 to +70°C

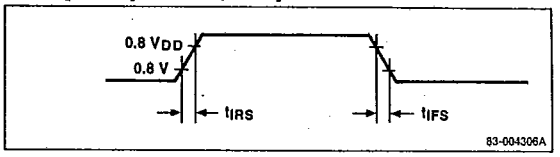
Parameter	Symbol	Min	Max	Unit
Data retention voltage	V_{DDDR}	2.4	5.5	V
V_{DD} rise time	t_{RVD}	200		μs
V_{DD} fall time	t_{FVD}	200		μs

Timing Waveforms

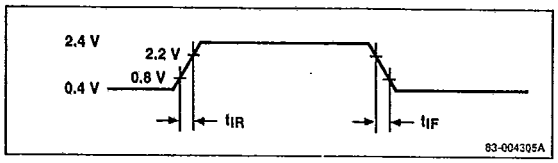
Stop Mode Data Retention Timing



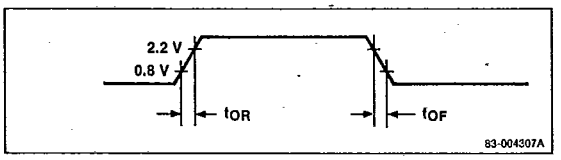
AC Input 2 ($\overline{RESET}$, NMI)



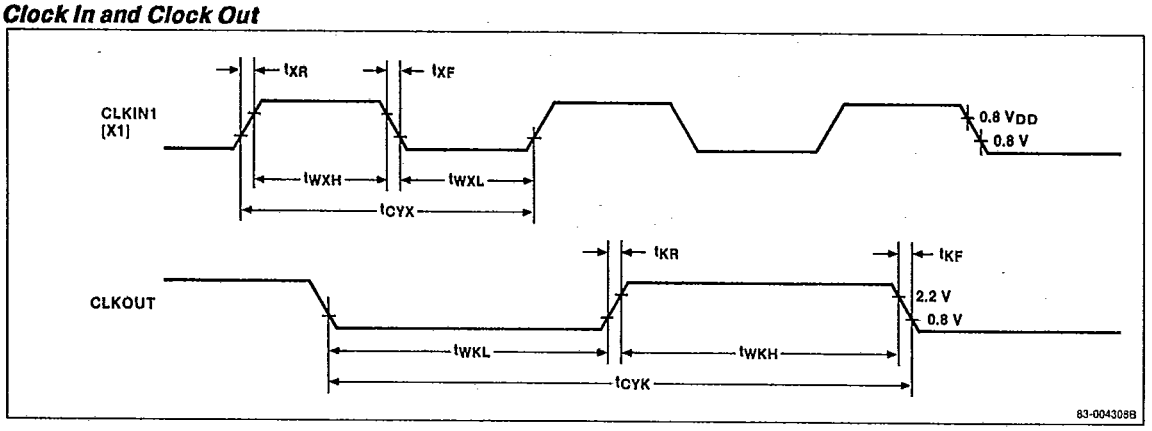
AC Input 1 (Except X1, X2, $\overline{RESET}$, NMI)



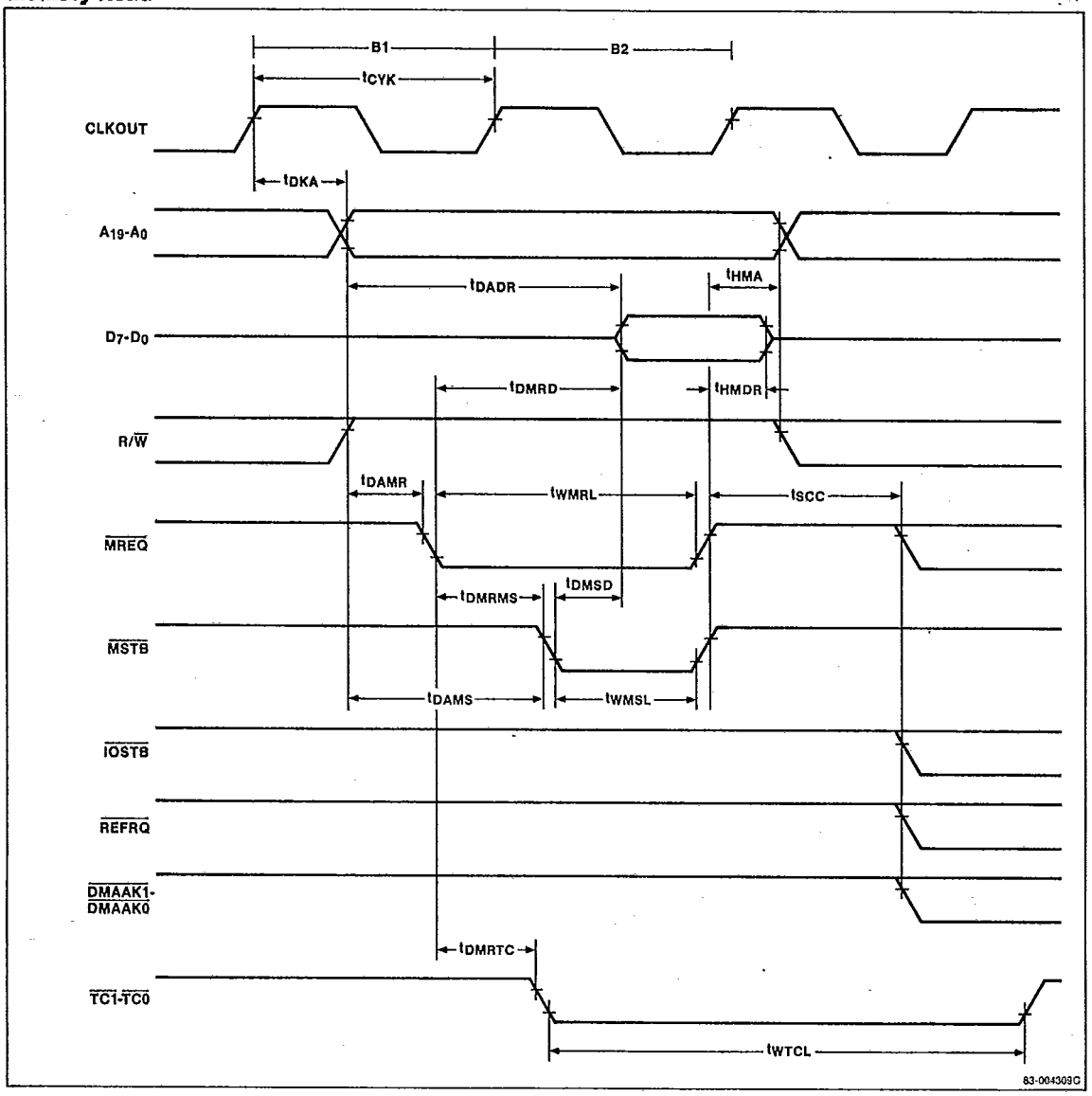
AC Output (Except CLKOUT)



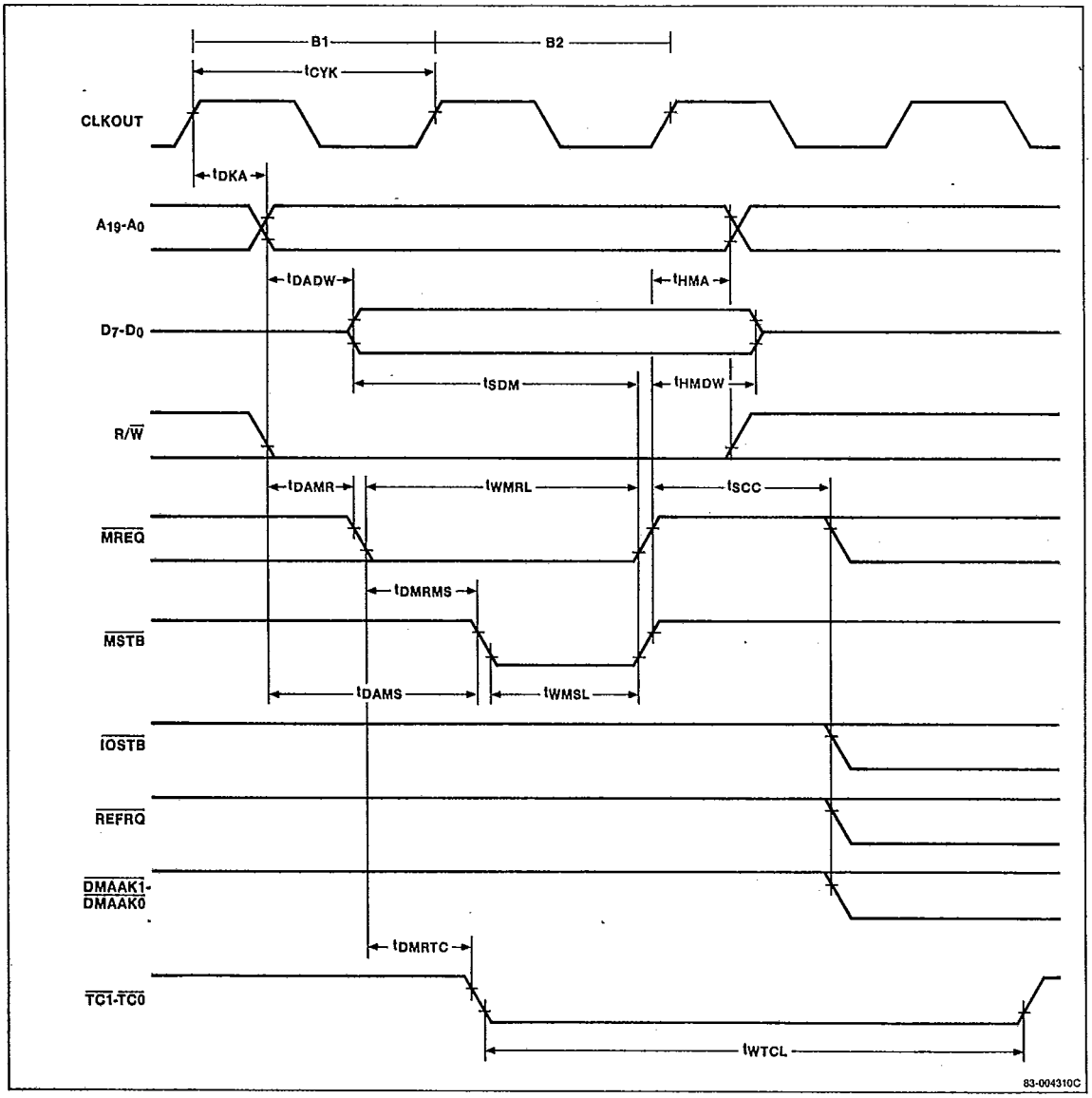
Clock In and Clock Out



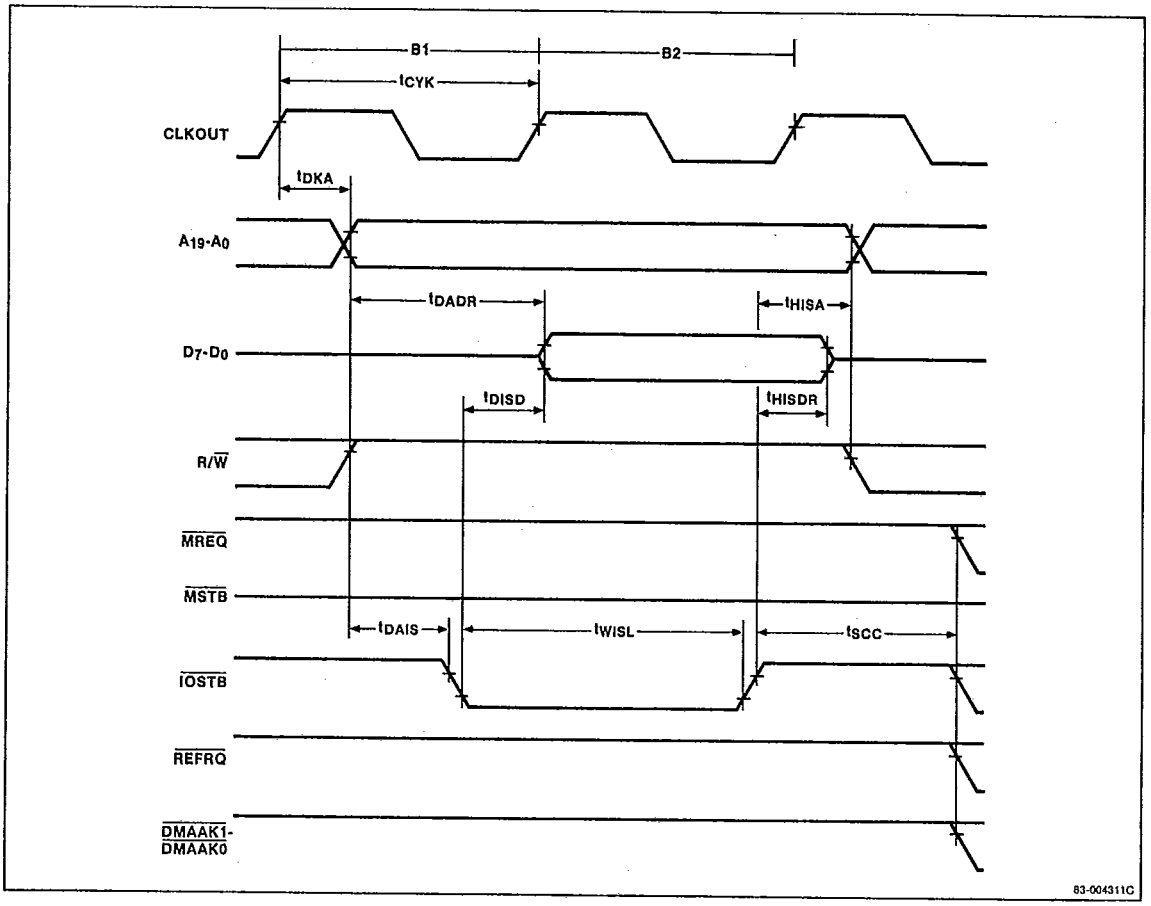
Memory Read


B1
B2
tCYK
CLKOUT
tDKA
A19-A0
tDADR
tHMA
D7-D0
tDMRD
tHMDR
R/W
tDAMR
tWMRL
tSCC
MREQ
tDMRMS
tDMSD
MSTB
tDAMS
tWMSL
IOSTB
REFRQ
DMAAK1-
DMAAK0
tDMRTC
TC1-TC0
tWTCL
83-004309C

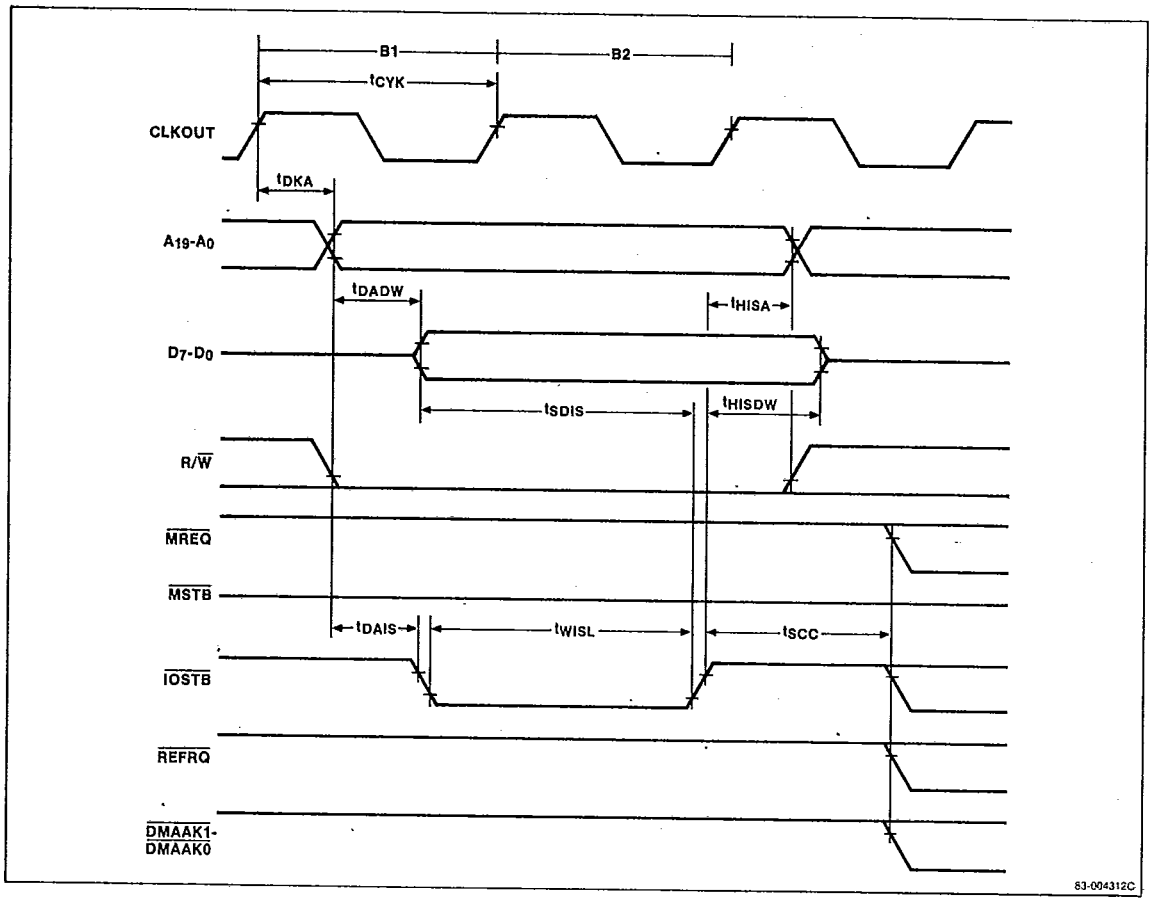
Memory Write



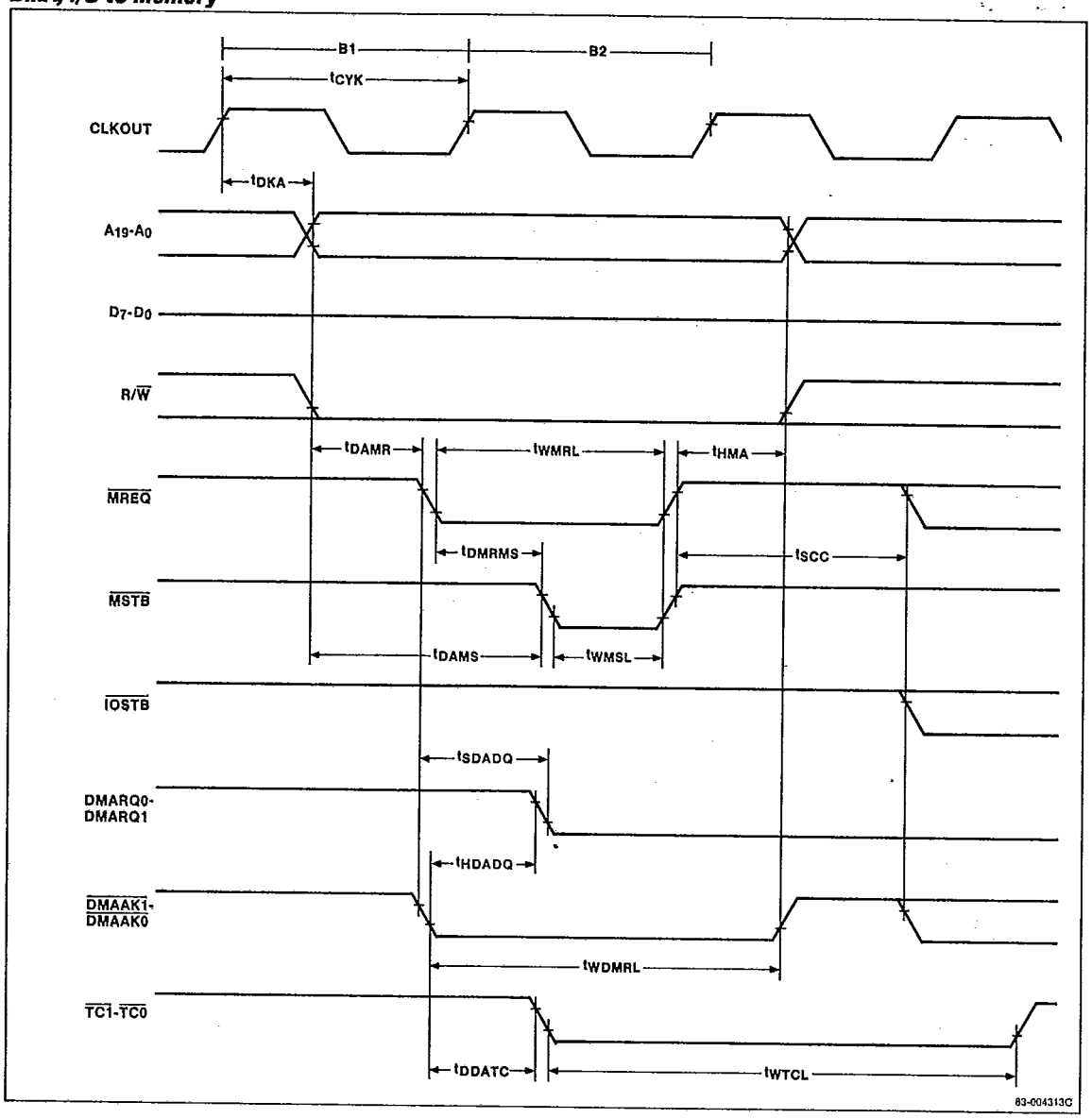
I/O Read


B1
B2
tCYK
CLKOUT
tDKA
A19-A0
tDADR
tHISA
D7-D0
tDISD
tHISDR
R/W
MREQ
MSTB
tDAIS
tWISL
tSCC
IOSTB
REFRQ
DMAAK1-
DMAAK0
83-004311C

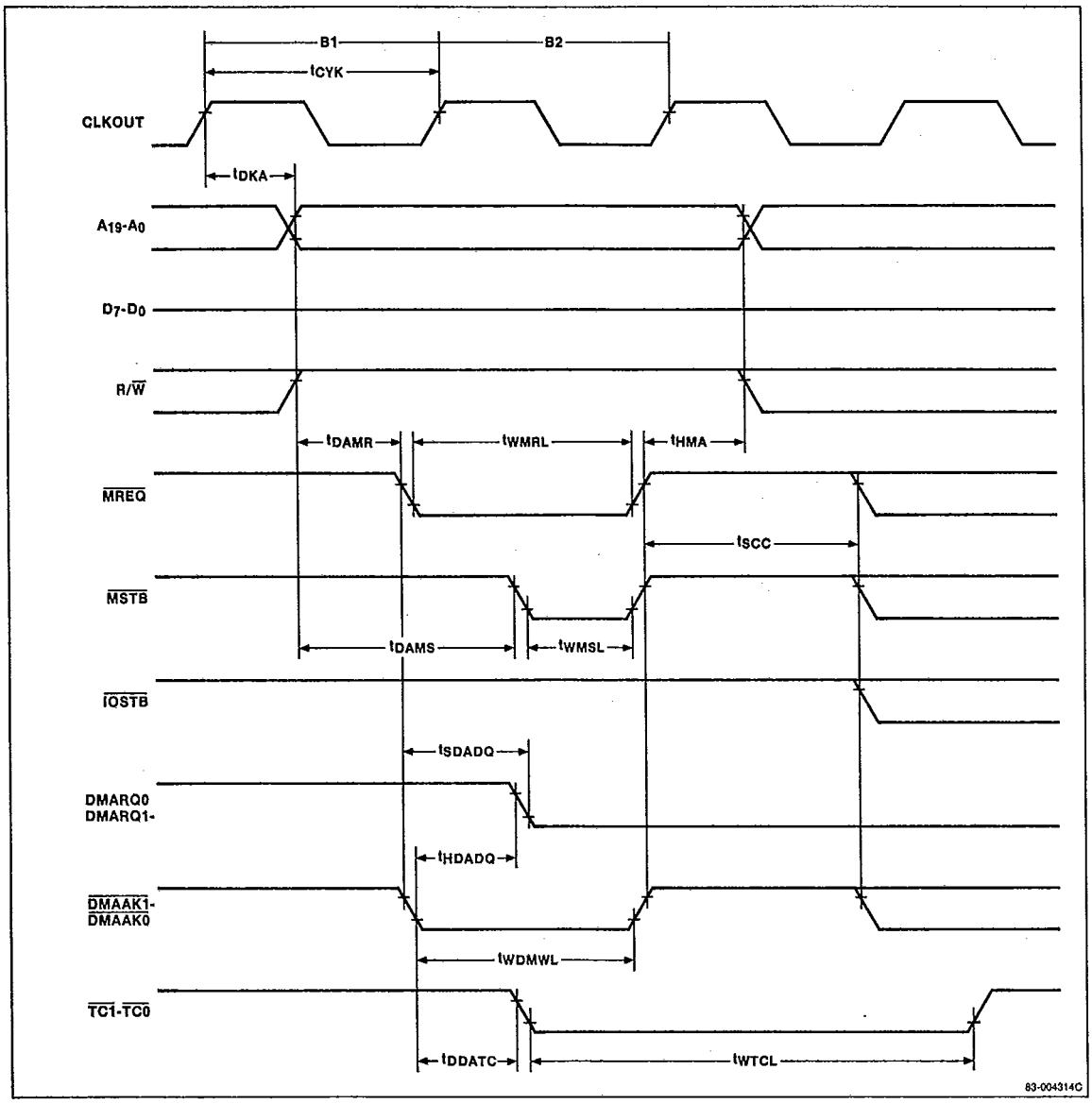
I/O Write


B1
B2
tCYK
CLKOUT
tDKA
A19-A0
tDADW
tHISA
D7-D0
tSDIS
tHISDW
R/W
MREQ
MSTB
tDAIS
tWISL
tSCC
IOSTB
REFRQ
DMAAK1-
DMAAK0
83-004312C

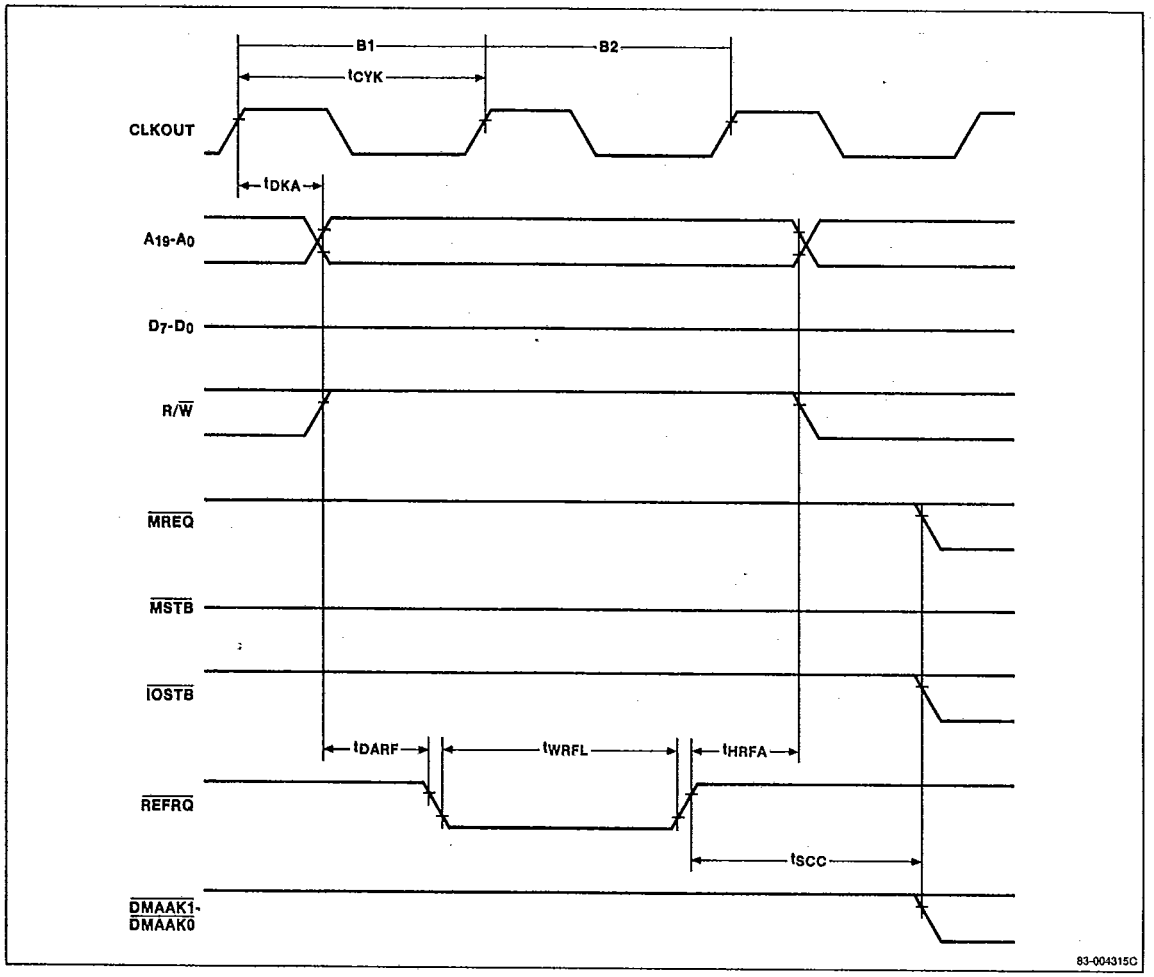
DMA, I/O to Memory



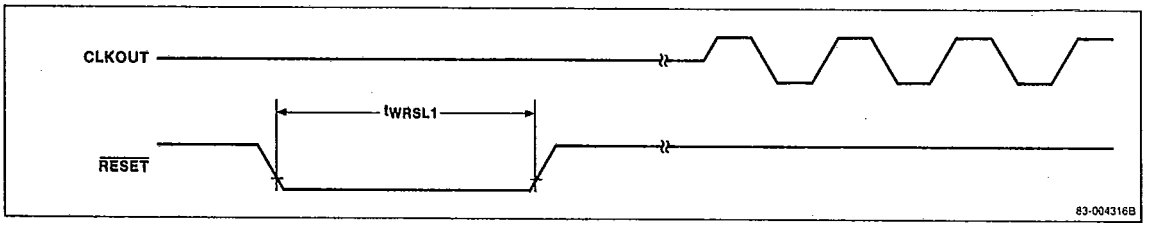
DMA, Memory to I/O



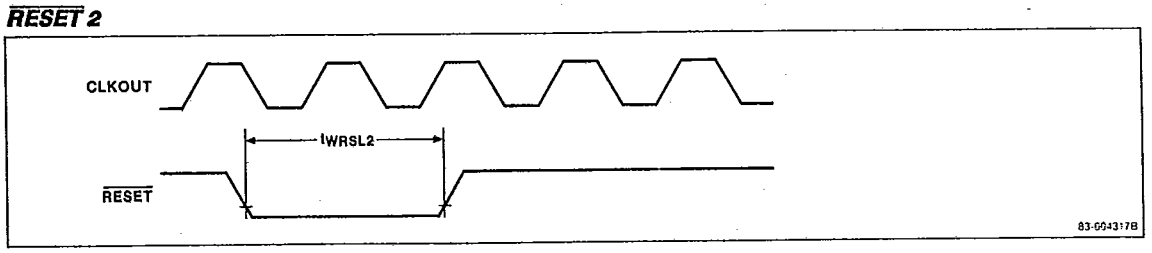
Refresh



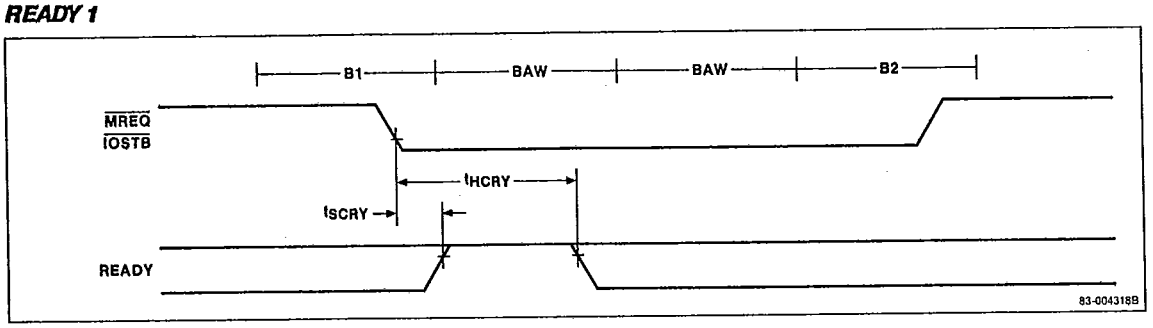
RESET 1



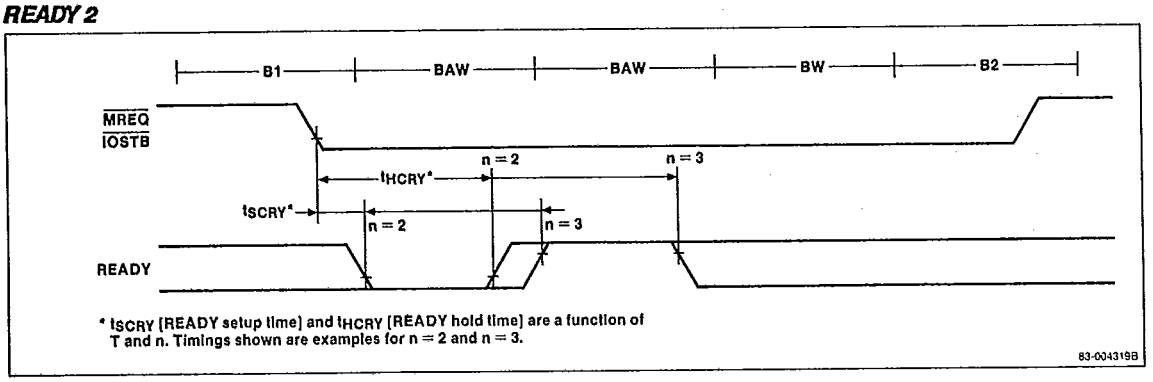
RESET 2



READY 1

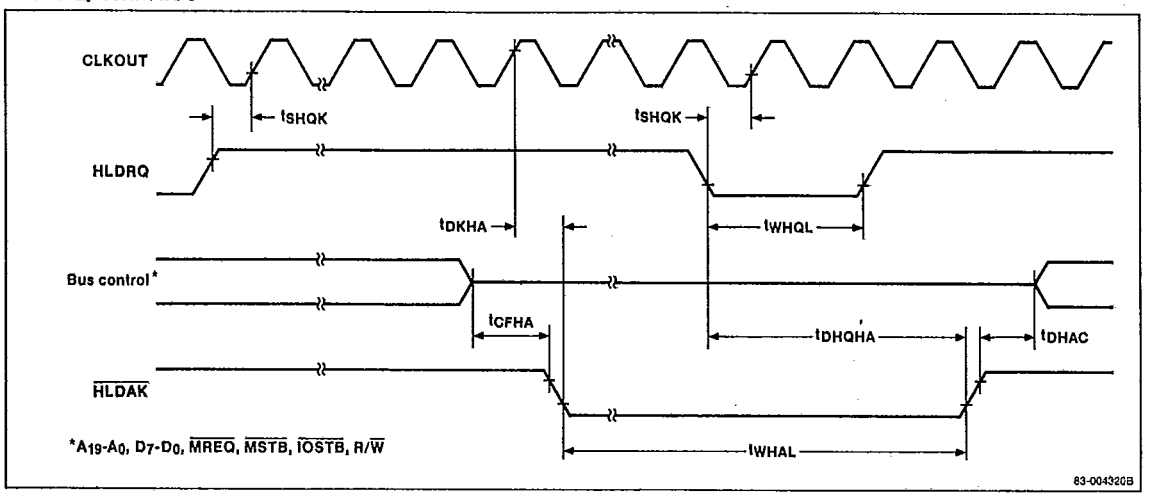


READY 2

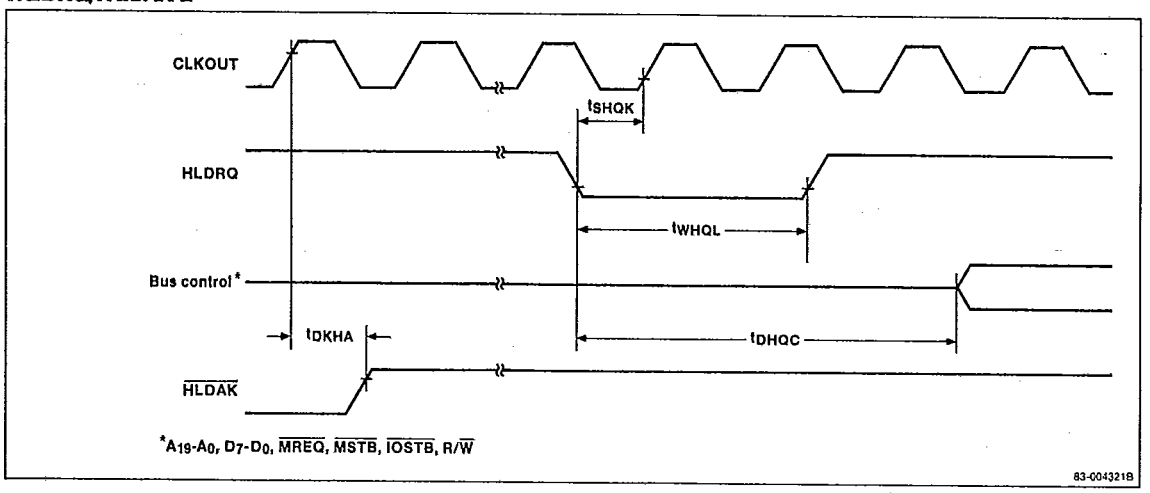


* t_{SCRY} [READY setup time] and t_{HCRY} [READY hold time] are a function of T and n. Timings shown are examples for n = 2 and n = 3.

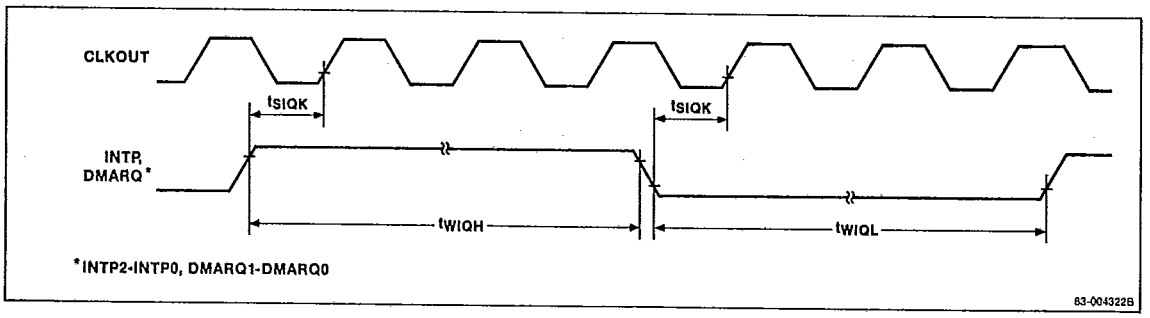
HLDRQ/$\overline{\text{HLDAK}}$ 1



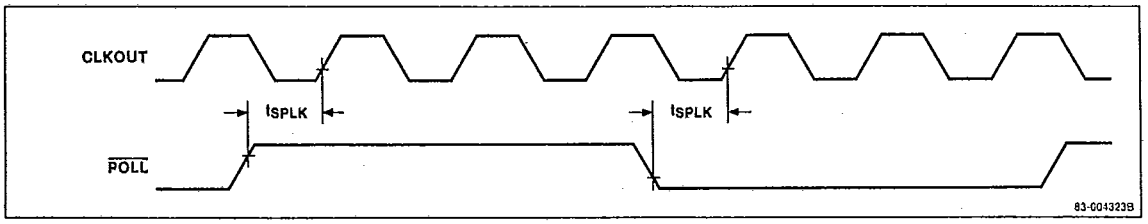
HLDRQ/$\overline{\text{HLDAK}}$ 2



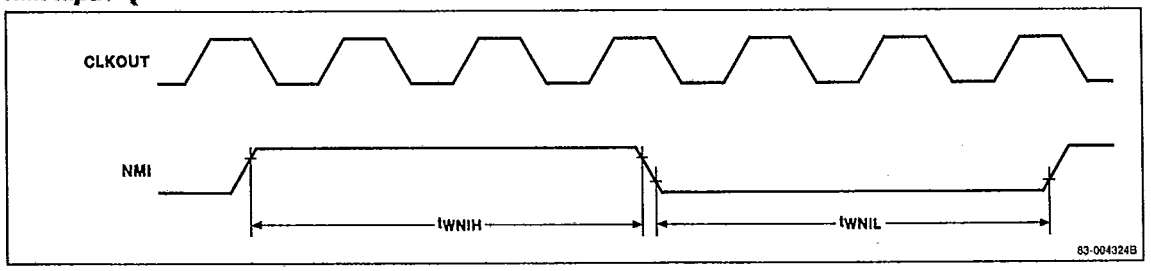
INTP, DMARQ Input



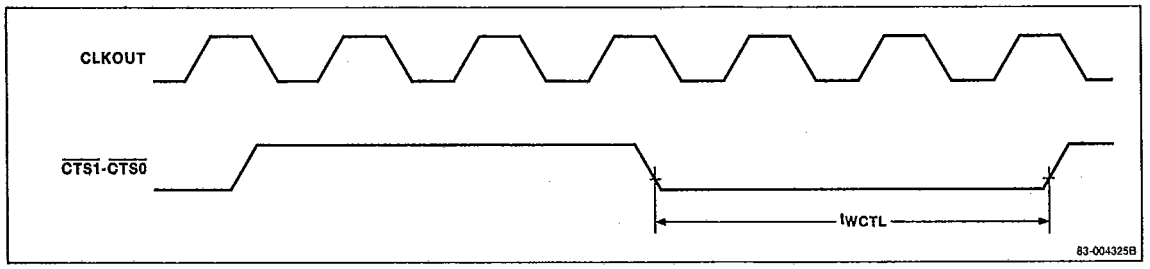
$\overline{\text{POLL}}$ Input



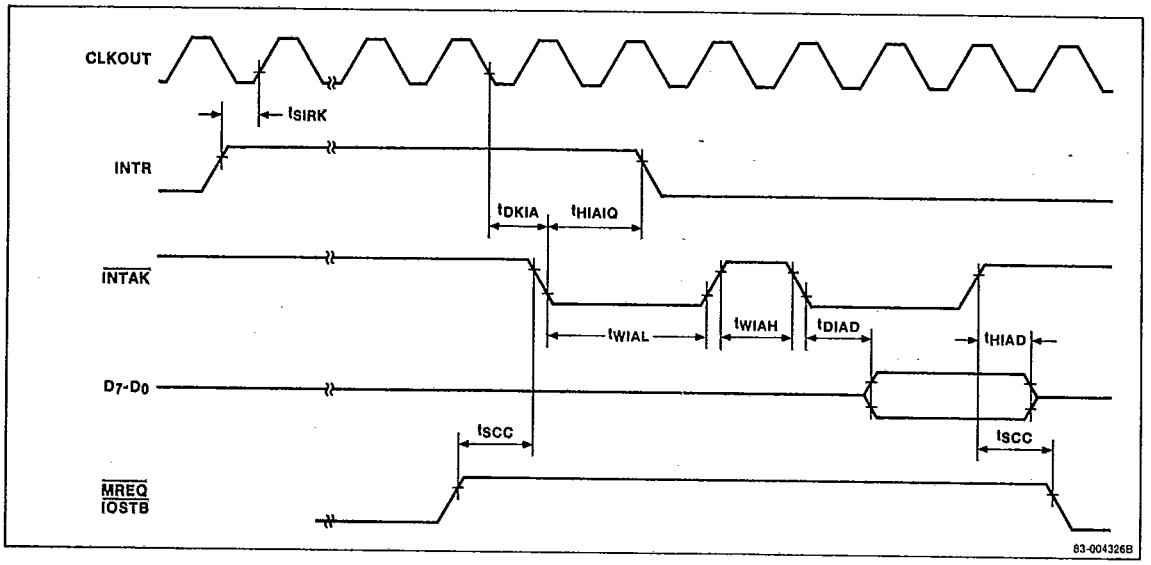
NMI Input



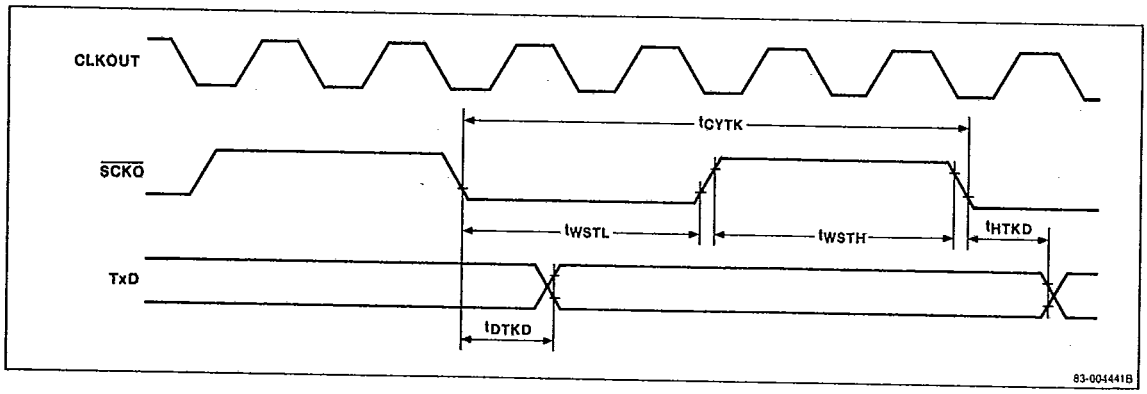
$\overline{\text{CTS}}$ Input



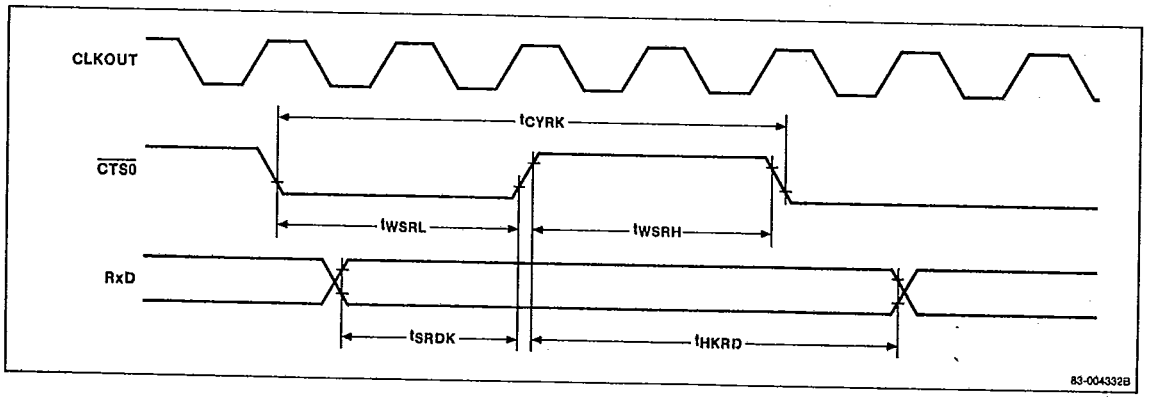
INTR/INTAK



Serial Transmit



Serial Receive



INSTRUCTION SET

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

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

Symbols and Abbreviations

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

Symbols and Abbreviations (cont)

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

Flag Symbols

Identifier	Description
(blank)	No change
0	Cleared to 0
1	Set to 1
x	Set or cleared according to the result
u	Undefined
r	Value saved earlier is restored

8- and 16-Bit Registers (mod = 11)

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

Segment Registers

sreg	Register
00	DS_1
01	PS
10	SS
11	DS_0

Memory Addressing

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

Instruction Clock Counts

Mnemonic	Operand	Clocks
ADD	reg8, reg8	2
	reg16, reg16	2
	reg8, mem8	EA+6+W
	reg16, mem16	EA+8+2W
	mem8, reg8	EA+8+2W [EA+6+W]
	reg16, mem16	EA+12+4W [EA+8+2W]
	reg8, imm8	5
	reg16, imm8	5
	mem16, imm16	6
	mem8,imm8	EA+9+2W [EA+7+2W]
	mem16, imm8	EA+9+2W [EA+7+2W]
	mem16, imm16	EA+14+4W [EA+10+4W]
	AL, imm8	5
	AW, imm16	6
ADD4S		22+(27+3W)n [22+(25+3W)n]
ADDC	Same as ADD	
ADJ4A		9
ADJ4S		9
ADJBA		17
ADJBS		17
AND	reg8, reg8	2
	reg16, reg16	2
	reg8, mem8	EA+6+W
	reg16, mem16	EA+8+2W
	mem8, reg8	EA+8+2W [EA+6+W]
	mem16, reg16	EA+12+4W [EA+8+2W]
	reg8, imm8	5
	reg16, imm16	6
	mem8, imm8	EA+9+2W [EA+7+2W]
	mem16, imm16	EA+14+4W [EA+10+4W]
Bcond (conditional branch)		8 or 15
BCWZ		8 or 15
BR	near-label	12
	short-label	12
	regptr16	13
	memptr16	EA+17+2W
	far-label	15
	memptr32	EA+25+4W
BRK	3	55+10W [43+10W]
	imm8	56+10W [44+10W]
BRKCS		15
BRKV		55+10W [43+10W]
BTCLR		29
BUSLOCK		2

Mnemonic	Operand	Clocks
CALL	near-proc	22+2W [18+2W]
	regptr16	22+2W [18+2W]
	memptr16	EA+26+4W [EA+24+4W]
	far-proc	36+4W [34+4W]
	memptr32	EA+36+8W [EA+24+8W]
CHKIND		EA+26+4W
CLR1	CY	2
	DIR	2
	reg8, CL	8
	reg16, CL	8
	mem8, CL	EA+14+2W [EA+12+W]
	mem16, CL	EA+18+4W [EA+14+2W]
	reg8, imm3	7
	reg16, imm4	7
	mem8, imm3	EA+11+2W [EA+9+W]
	mem16, imm4	EA+15+4W [EA+10+2W]
CMP	reg8, reg8	2
	reg16, reg16	2
	reg8, mem8	EA+6+W
	reg16, mem16	EA+8+2W
	mem8, reg8	EA+6+W
	mem16, reg16	EA+8+2W
	reg8, imm8	5
	reg16, imm8	5
	reg16, imm16	6
	mem8, imm8	EA+7+W
	mem16, imm8	EA+10+2W
	mem16, imm16	EA+10+2W
	AL, imm8	5
	AW, imm16	6
CMP4S		22+(23+2W)n
CMPBK	mem8, mem8	23+2W [19+2W]
	mem16, mem16	27+4W [21+2W]
CMPBKB		16+(21+2W)n
CMPBKW		16+(25+4W)n
CMPM	mem8	17+W
	mem16	19+2W
CMPMB		16+(15+W)n
CMPMW		16+(17+2W)n
CVTBD		19
CVTBW		3
CVTDB		20
CVTWL		8

Instruction Clock Counts (cont)

Mnemonic	Operand	Clocks
DBNZ		8 or 17
DBNZE		8 or 17
DBNZNE		8 or 17
DEC	reg8	5
	reg16	2
	mem8	EA+11+2W [EA+9+2W]
	mem16	EA+15+4W [EA+11+4W]
DI		4
DISPOSE		12+2W
DIV	AW, reg8	46-56
	AW, mem8	EA+48+W to EA+58+W
	DW:AW, reg16	54-64
	DW; AW, mem16	EA+58+2W to EA+68+2W
DIVU	AW, reg8	31
	AW, mem8	EA+33+W
	DW:AW, reg16	39
	DW: AW, mem16	EA+43+2W
DS0:		2
DS1:		2
EI		12
EXT	reg8, reg8	41-121
	reg8, Imm4	42-122
FINT		2
FP01		60+10W [48+10W]
FP02		60+10W [48+10W]
HALT		0
IN	AL, Imm8	14+W
	AW, Imm8	16+2W
	AL, DW	13+W
	AW, DW	15+2W
INC	reg8	5
	reg16	2
	mem8	EA+11+2W [EA+9+2W]
	mem16	EA+15+4W [EA+11+4W]
INM	mem8, DW	19+2W [17+2W]
	mem16, DW	21+4W [17+4W]
	mem8, DW	18+(13+2W)n [18+(11+2W)n]
	mem16, DW	18+(15+4W)n [18+(11+4W)n]
INS	reg8, reg8	63-155
	reg8, Imm4	64-156
LDEA		EA+2
LDM	mem8	12+W
	mem16	16+(12+2W)n
LDMB	mem16	14+2W
LDMW	mem8	16+(10+W)n

Mnemonic	Operand	Clocks
MOV	reg8, reg8	2
	reg16, reg16	2
	reg8, mem8	EA+6+W
	reg16, mem16	EA+8+2W
	mem8, reg8	EA+4+W [EA+2]
	mem16, reg16	EA+6+2W [EA+2]
	reg8, Imm8	5
	reg16, Imm16	6
	mem8, Imm8	EA+5+W
	mem16, Imm16	EA+5+2W
	AL, dmem8	9+W
	AW, dmem16	11+2W
	dmem8, AL	7+W [5]
	dmem16, AW	9+2W [5]
	sreg, reg16	4
	sreg, mem16	EA+10+2W
	reg16, sreg	3
	mem16, sreg	EA+7+2W [EA+3]
	AH, PSW	2
	PSW, AH	3
	DS0, reg16, memptr32	EA+19+4W
	DS1, reg16, memptr32	EA+19+4W
MOVBK	mem8, mem8	20+2W [16+W]
	mem16, mem16	16+(20+4W)n [16+(12+2W)n]
MOVBKB	mem8, mem8	16+(16+2W)n [16+(12+W)n]
MOVBKW	mem16, mem16	24+4W [20+2W]
MOVSPA		16
MOVSPB		11
MUL	AW, AL, reg8	31-40
	AW, AL, mem8	EA+33+W to EA+42+W
	DW:AW, AW, reg16	39-48
	DW:AW, AW, mem16	EA+43+2W to EA+52+2W
	reg16, reg16, Imm8	39-49
	reg16, mem16, Imm8	EA+43+2W to EA+53+2W
	reg16, reg16, Imm16	40-50
	reg16, mem16, Imm16	EA+44+2W to EA+54+2W
MULU	reg8	24
	mem8	EA+26+W
	reg16	32
	mem16	EA+34+2W

4d

Instruction Clock Counts (cont)

Mnemonic	Operand	Clocks
NEG	reg8	5
	reg16	5
	mem8	EA+11+2W [EA+9+W]
	mem16	EA+15+4W [EA+11+2W]
NOP		3
NOT	reg8	5
	reg16	5
	mem8	EA+11+2W [EA+9+W]
	mem16	EA+15+4W [EA+11+2W]
NOT1	CY	2
	reg8, CL	7
	reg16, CL	7
	mem8, CL	EA+13+2W [EA+11+W]
	mem16, CL	EA+17+4W [EA+13+2W]
	reg8, imm3	6
	reg16, imm4	6
	mem8, imm3	EA+10+2W [EA+8+W]
	mem16, imm4	EA+14+4W [EA+10+2W]
OR	reg8, reg8	2
	reg16, reg16	2
	reg8, mem8	EA+6+W
	reg16, mem16	EA+8+2W
	mem8, reg8	EA+8+2W [EA+6+W]
	mem16, reg16	EA+12+4W [EA+8+2W]
	reg8, imm8	5
	reg16, imm16	6
	mem8, imm8	EA+9+W [EA+7+W]
	mem16, imm16	EA+14+4W [EA+10+4W]
	AL, imm8	5
	AW, imm16	6
OUT	imm8, AL	10+W
	imm8, AW	10+2W
	DW, AL	9+W
	DW, AW	9+2W
OUTM	DW, mem8	19+2W [17+2W]
	DW, mem16	21+4W [17+4W]
	DW, mem8	18+(13+2W)n [18+(11+2W)n]
	DW, mem16	18+(15+4W)n [18+(11+4W)n]
POLL		0
POP	reg16	12+2W
	mem16	EA+16+4W [EA+12+2W]
	DS1	13+2W
	SS	13+2W
	DS0	13+2W
	PSW	14+2W
	R	82+16W [58]
PREPARE	imm16, imm8	imm8 = 0:27+2W imm8 = 1:39+4W imm8 = n > 1:46+19 (n−1)+4W
PS:		2
PUSH	reg16	10+2W [6]
	mem16	EA+18+4W [EA+14+4W]
	DS1	11+2W [7]
	PS	11+2W [7]
	SS	11+2W [7]
	DS0	11+2W [7]
	PSW	10+2W [6]
	R	82+16W [50]
	imm8	13+2W [9]
	imm16	14+2W [10]
REP		2
REPE		2
REPZ		2
REPC		2
REPNC		2
REPNE		2
REPNZ		2
RET	null	20+2W
	pop-value	20+2W
	null	29+4W
	pop-value	30+4W
RETI		43+6W [35+2W]
RETRBI		12
ROL	reg8, 1	8
	reg16, 1	8
	mem8, 1	EA+14+2W [EA+12+W]
	mem16, 1	EA+18+4W [EA+14+2W]
	reg8, CL	11+2n
	reg16, CL	11+2n
	mem8, CL	EA+17+2W+2n [EA+15+W+2n]
	mem16, CL	EA+21+4W+2n [EA+17+2W+2n]
	reg8, imm8	9+2n
	reg16, imm8	9+2n
	mem8, imm8	EA+13+2W+2n [EA+11+W+2n]
	mem16, imm8	EA+17+4W+2n [EA+13+2W+2n]
ROL4	reg8	17
	mem8	EA+18+2W [EA+16+2W]

Instruction Clock Counts (cont)

Mnemonic	Operand	Clocks
ROLC	Same as ROL	
ROR	Same as ROL	
ROR4	reg8	21
	mem8	EA+24+2W [EA+22+2W]
RORC	Same as ROL	
SET1	CY	2
	DIR	2
	reg8, CL	7
	reg16, CL	7
	mem8, CL	EA+13+2W [EA+11+W]
	mem16, CL	EA+17+4W [EA+13+2W]
	reg8, imm3	6
	reg16, imm4	6
	mem8, imm3	EA+10+2W [EA+8+W]
	mem16, imm4	EA+14+4W [EA+10+2W]
SHL	Same as ROL	
SHR	Same as ROL	
SHRA	Same as ROL	
SS:		2
STM	mem8	12+2 [10]
	mem16	16+(10+2W)n [16+(6+2W)n]
STMB	mem8	16+(8+W)n [16+(6+W)n]
STMW	mem16	14+2W [10]
STOP		0
SUB	Same as ADD	
SUB4S		22+(27+3W)n [22+(25+3W)n]
SUBC	Same as ADD	
TEST	reg8, reg8	4
	reg16, reg16	4
	reg8, mem8	EA+8+W
	reg16, mem16	EA+10+2W
	mem8, reg8	EA+8+W
	mem16, reg16	EA+10+W
	reg8, imm8	7
	reg16, imm16	8
	mem8, imm8	EA+11+W
	mem16, imm16	EA+11+2W
	AL, imm8	5
	AW, imm16	6
TEST1	reg8, CL	7
	reg16, CL	7
	mem8, CL	EA+11+W
	mem16, CL	EA+13+2W
	reg8, imm3	6
	reg16, imm4	6
	mem8, imm3	EA+8+W
	mem16, imm4	EA+10+2W
TRANS		10+W
TRANSB		10+W
TSKSW		11
XCH	reg8, reg8	3
	reg16, reg16	3
	reg8, mem8	EA+10+2W [EA+8+2W]
	reg16, mem16	EA+14+4W [EA+10+4W]
	mem8, reg8	EA+10+2W [EA+8+2W]
	mem16, reg16	EA+14+4W [EA+10+4W]
	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

Execution Clock Counts for Operations

	Byte		Word	
	RAM Enable	RAM Disable	RAM Enable	RAM Disable
Context switch interrupt (Note 1)	—	—	28	28
DMA (Single-step mode) (Note 2)	4+2W	4+2W	8+4W	8+4W
DMA (Demand release mode)	2+W	2+W	4+W	4+2W
DMA (Burst mode)	(4+2W)n	(4+2W)n	(8+4W)n	(8+4W)n
DMA (Single-transfer mode)	2+W	2+W	4+2W	4+2W
Interrupt (INT pin) (Note 3)	—	—	62+10W	50+10W
Macroservice, sfr ← mem (Note 2)	23+W	21+W	27+2W	25+2W
Macroservice, mem ← sfr	22+W	20+W	26+2W	24+2W
Macroservice (Search char mode), sfr ← mem	37+W	37+W	—	—
Macroservice (Search char mode), mem ← sfr	37+W	37+W	—	—
Priority interrupt (Vectored) (Note 1)	—	—	58+10W	46+10W

N = number of clocks to complete the instruction currently executing.

Notes:

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

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

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

Bus Controller Latency

		Clocks	
Latency	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 release	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.

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						

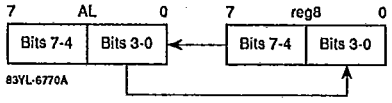
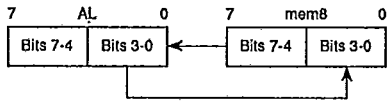
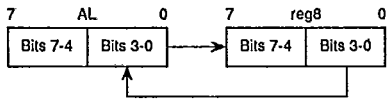
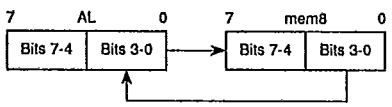
Instruction Set (cont)

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	reg8		0	0	0	0	1	1	1	1	3						
			0	0	1	0	1	0	0	0							
			1	1	0	0	0		reg								
	mem8		0	0	0	0	1	1	1	1	3-5						
			0	0	1	0	1	0	0	0							
			mod		0	0	0		mem								
ROR4	reg8		0	0	0	0	1	1	1	1	3						
			0	0	1	0	1	0	1	0							
			1	1	0	0	0		reg								
	mem8		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								

Instruction Set (cont)

Mnemonic	Operand	Operation	Operation Code 7	6	5	4	3	2	1	0	Bytes	Flags 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	2	u	u	u	u	u	u
			1	1	1	1	1		reg								
	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	2-4	u	u	u	u	u	u
			mod		1	1	1		mem								
Data Conversion																	
CVTBD		AH ← AL ÷ 0AH, AL ← AL % 0AH	1	1	0	1	0	1	0	0	2	u	u	u	x	x	x
			0	0	0	0	1	0	1	0							
CVTDB		AH ← 0, AL ← AH x 0AH + AL	1	1	0	1	0	1	0	1	2	u	u	u	x	x	x
			0	0	0	0	1	0	1	0							
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	2	x	x	x	x	x	x
			1	1		reg			reg								
	mem, reg	(mem)−reg	0	0	1	1	1	0	0	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, mem	reg−(mem)	0	0	1	1	1	0	1	W	2-4	x	x	x	x	x	x
			mod			reg			mem								
	reg, imm	reg−imm	1	0	0	0	0	0	S	W	3-4	x	x	x	x	x	x
			1	1	1	1	1		reg								
	mem, imm	(mem)−imm	1	0	0	0	0	0	S	W	3-6	x	x	x	x	x	x
			mod		1	1	1		mem								
	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 7	6	5	4	3	2	1	0	Bytes	Flags 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 1 1 reg reg	2	u	0	0	x	x	x
	mem, reg	(mem) ← (mem) OR reg	0 0 0 0 1 0 0 W mod reg mem	2-4	u	0	0	x	x	x
	reg, mem	reg ← reg OR (mem)	0 0 0 0 1 0 1 W mod reg mem	2-4	u	0	0	x	x	x
	reg, imm	reg ← reg OR imm	1 0 0 0 0 0 0 W 1 1 0 0 1 reg	3-4	u	0	0	x	x	x
	mem, imm	(mem) ← (mem) OR imm	1 0 0 0 0 0 0 W mod 0 0 1 mem	3-6	u	0	0	x	x	x
	acc, imm	When W = 0: AL ← AL OR imm8 When W = 1: AW ← AW OR imm16	0 0 0 0 1 1 0 W	2-3	u	0	0	x	x	x
XOR	reg, reg	reg ← reg XOR reg	0 0 1 1 0 0 1 W 1 1 reg reg	2	u	0	0	x	x	x
	mem, reg	(mem) ← (mem) XOR reg	0 0 1 1 0 0 0 W mod reg mem	2-4	u	0	0	x	x	x
	reg, mem	reg ← reg XOR (mem)	0 0 1 1 0 0 1 W mod reg mem	2-4	u	0	0	x	x	x
	reg, imm	reg ← reg XOR imm	1 0 0 0 0 0 0 W 1 1 1 1 0 reg	3-4	u	0	0	x	x	x
	mem, imm	(mem) ← (mem) XOR imm	1 0 0 0 0 0 0 W mod 1 1 0 mem	3-6	u	0	0	x	x	x
	acc, imm	When W = 0: AL ← AL XOR imm8 When W = 1: AW ← AW XOR imm16	0 0 1 1 0 1 0 W	2-3	u	0	0	x	x	x
Bit Operation										
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 0 0 0 1 0 0 0 0 1 1 0 0 0 reg	3	u	0	0	u	u	x
	mem8, CL	(mem8) bit no. CL = 0: Z ← 1 (mem8) bit no. CL = 1: Z ← 0	0 0 0 0 1 1 1 1 0 0 0 1 0 0 0 0 mod 0 0 0 mem	3-5	u	0	0	u	u	x
	reg16, CL	reg16 bit no. CL = 0: Z ← 1 reg16 bit no. CL = 1: Z ← 0	0 0 0 0 1 1 1 1 0 0 0 1 0 0 0 1 1 1 0 0 0 reg	3	u	0	0	u	u	x
	mem16, CL	(mem16) bit no. CL = 0: Z ← 1 (mem16) bit no. CL = 1: Z ← 0	0 0 0 0 1 1 1 1 0 0 0 1 0 0 0 1 mod 0 0 0 mem	3-5	u	0	0	u	u	x
	reg8, imm3	reg8 bit no. imm3 = 0: Z ← 1 reg8 bit no. imm3 = 1: Z ← 0	0 0 0 0 1 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 reg	4	u	0	0	u	u	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)																	
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				

4d

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 0 0 0 1 0 0 1 0 1 1 0 0 0 reg	3	
	mem8, CL	(mem8) bit no. CL ← 0	0 0 0 0 1 1 1 1 0 0 0 1 0 0 1 0 mod 0 0 0 mem	3-5	
	reg16, CL	reg16 bit no. CL ← 0	0 0 0 0 1 1 1 1 0 0 0 1 0 0 1 1 1 1 0 0 0 reg	3	
	mem16, CL	(mem16) bit no. CL ← 0	0 0 0 0 1 1 1 1 0 0 0 1 0 0 1 1 mod 0 0 0 mem	3-5	
	reg8, imm3	reg8 bit no. imm3 ← 0	0 0 0 0 1 1 1 1 0 0 0 1 1 0 1 0 1 1 0 0 0 reg	4	
	mem8, imm3	(mem8) bit no. imm3 ← 0	0 0 0 0 1 1 1 1 0 0 0 1 1 0 1 0 mod 0 0 0 mem	4-6	
	reg16, imm4	reg16 bit no. imm4 ← 0	0 0 0 0 1 1 1 1 0 0 0 1 1 0 1 1 1 1 0 0 0 reg	4	
	mem16, imm4	(mem16) bit no. imm4 ← 0	0 0 0 0 1 1 1 1 0 0 0 1 1 0 1 1 mod 0 0 0 mem	4-6	
	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 0 0 0 1 0 1 0 0 1 1 0 0 0 reg	3	
	mem8, CL	(mem8) bit no. CL ← 1	0 0 0 0 1 1 1 1 0 0 0 1 0 1 0 0 mod 0 0 0 mem	3-5	
	reg16, CL	reg16 bit no. CL ← 1	0 0 0 0 1 1 1 1 0 0 0 1 0 1 0 1 1 1 0 0 0 reg	3	
	mem16, CL	(mem16) bit no. CL ← 1	0 0 0 0 1 1 1 1 0 0 0 1 0 1 0 1 mod 0 0 0 mem	3-5	

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								

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 1 1 1 0 1 reg	2	u	x	x	x	x	x
	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2 When MSB of (mem) ≠ bit following MSB of (mem): V ← 1 When MSB of (mem) = bit following MSB of (mem): V ← 0	1 1 0 1 0 0 0 W mod 1 0 1 mem	2-4	u	x	x	x	x	x
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1	1 1 0 1 0 0 1 W 1 1 1 0 1 reg	2	u	x	u	x	x	x
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1	1 1 0 1 0 0 1 W mod 1 0 1 mem	2-4	u	x	u	x	x	x
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1	1 1 0 0 0 0 0 W 1 1 1 0 1 reg	3	u	x	u	x	x	x
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1	1 1 0 0 0 0 0 W mod 1 0 1 mem	3-5	u	x	u	x	x	x
SHRA	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2, V ← 0 MSB of operand does not change	1 1 0 1 0 0 0 W 1 1 1 1 1 reg	2	u	x	0	x	x	x
	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2, V ← 0, MSB of operand does not change	1 1 0 1 0 0 0 W mod 1 1 1 mem	2-4	u	x	0	x	x	x
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1 MSB of operand does not change	1 1 0 1 0 0 1 W 1 1 1 1 1 reg	2	u	x	u	x	x	x
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1 MSB of operand does not change	1 1 0 1 0 0 1 W mod 1 1 1 mem	2-4	u	x	u	x	x	x
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp − 1 MSB of operand does not change	1 1 0 0 0 0 0 W 1 1 1 1 1 reg	3	u	x	u	x	x	x
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp − 1 MSB of operand does not change	1 1 0 0 0 0 0 W mod 1 1 1 mem	3-5	u	x	u	x	x	x
Rotation										
ROL	reg, 1	CY ← MSB of reg, reg ← reg x 2 + CY MSB of reg ≠ CY: V ← 1 MSB of reg = CY: V ← 0	1 1 0 1 0 0 0 W 1 1 0 0 0 reg	2		x	x			
	mem, 1	CY ← MSB of (mem), (mem) ← (mem) x 2 + CY MSB of (mem) ≠ CY: V ← 1 MSB of (mem) = CY: V ← 0	1 1 0 1 0 0 0 W mod 0 0 0 mem	2-4		x	x			

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									

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 1 1 0 1 0 reg	2	x x
	mem, 1	tmpcy ← CY, CY ← MSB of (mem), (mem) ← (mem) x 2 + tmpcy, MSB of (mem) = CY: V ← 0, MSB of (mem) ≠ CY: V ← 1	1 1 0 1 0 0 0 W mod 0 1 0 mem	2-4	x x
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of reg, reg ← reg x 2 + tmpcy, temp ← temp − 1	1 1 0 1 0 0 1 W 1 1 0 1 0 reg	2	x u
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of (mem), (mem) ← (mem) x 2 + tmpcy, temp ← temp − 1	1 1 0 1 0 0 1 W mod 0 1 0 mem	2-4	x u
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of reg, reg ← reg x 2 + tmpcy, temp ← temp − 1	1 1 0 0 0 0 0 W 1 1 0 1 0 reg	3	x u
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← MSB of (mem), (mem) ← (mem) x 2 + tmpcy, temp ← temp − 1	1 1 0 0 0 0 0 W mod 0 1 0 mem	3-5	x u
RORC	reg, 1	tmpcy ← CY, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← tmpcy, MSB of reg ≠ bit following MSB of reg: V ← 1, MSB of reg = bit following MSB of reg: V ← 0	1 1 0 1 0 0 0 W 1 1 0 1 1 reg	2	x x
	mem, 1	tmpcy ← CY, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, MSB of (mem) ← tmpcy, MSB of (mem) ≠ bit following MSB of (mem): V ← 1, MSB of (mem) = bit following MSB of (mem): V ← 0	1 1 0 1 0 0 0 W mod 0 1 1 mem	2-4	x x
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← tmpcy, temp ← temp − 1	1 1 0 1 0 0 1 W 1 1 0 1 1 reg	2	x u
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, MSB of (mem) ← tmpcy, temp ← temp − 1	1 1 0 1 0 0 1 W mod 0 1 1 mem	2-4	x u
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← tmpcy, temp ← temp − 1	1 1 0 0 0 0 0 W 1 1 0 1 1 reg	3	x u
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation, tmpcy ← CY, CY ← LSB of (mem), (mem) ← (mem) ÷ 2, MSB of (mem) ← tmpcy, temp ← temp − 1	1 1 0 0 0 0 0 W mod 0 1 1 mem	3-5	x u

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								