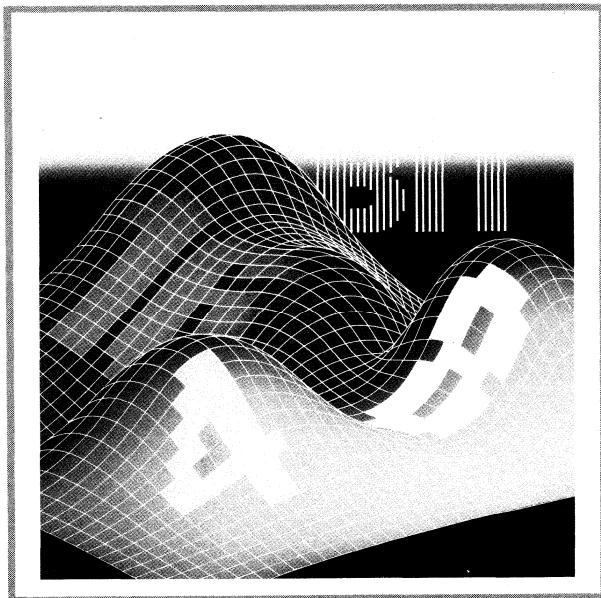


µCOM-87AD Family 8-Bit Microcomputers µPD78C1X



User's Manual

NEC



μCOM-87AD
μPD78C10/C10A/C11/C11A/
C12A/C14/C14A/CP14/C17/
C18/CP18

User's Manual

Concurrent CP/MTM and CP/M-86TM is a trademark of Digital Research Co., Ltd.

MS-DOSTM is a trademark of Microsoft Corp.

SUMMARY OF CONTENTS

CHAPTER 1	GENERAL DESCRIPTION
CHAPTER 2	PIN FUNCTION
CHAPTER 3	INTERNAL BLOCK FUNCTION
CHAPTER 4	PORT FUNCTION
CHAPTER 5	TIMER FUNCTION
CHAPTER 6	TIMER/EVENT COUNTER FUNCTION
CHAPTER 7	SERIAL INTERFACE FUNCTION
CHAPTER 8	ANALOG/DIGITAL CONVERTER FUNCTION
CHAPTER 9	INTERRUPT CONTROL FUNCTION
CHAPTER 10	CONTROL FUNCTION
CHAPTER 11	EXTERNAL DEVICE ACCESS AND TIMING
CHAPTER 12	INTERNAL EPROM ACCESS (uPD78CP14 ONLY)
CHAPTER 13	INSTRUCTION SET
APPENDIX A	INTRODUCTION OF PIGGY-BACK PRODUCT
APPENDIX B	DEVELOPMENT TOOL
APPENDIX C	INSTRUCTION INDEX

CONTENTS

	<u>PAGES</u>
CHAPTER 1 GENERAL DESCRIPTION	1-1
1.1 Features	1-3
1.2 Ordering Information	1-5
1.2.1 uPD78C10, 78C10A.....	1-5
1.2.2 uPD78C17.....	1-
1.2.3 uPD78C11, 78C11A.....	1-6
1.2.4 uPD78C12A.....	1-6
1.2.5 uPD78C14, 78C14A, 78CG14, 78CP14.....	1-7
1.2.6 uPD78C18, 78CP18.....	1-7
1.3 Pin Configuration.....	1-8
1.3.1 Shrunked-dual-in-line package, quad-in-line package straight (37), and quad-in-line package (36).....	1-8
1.3.2 QFP (1B/3BE)	1-10
1.3.3 QFP (AB8)	1-12
1.3.4 Plastic leaded chip carrier	1-13
1.4 Block Diagram	1-15
1.5 uCOM-87AD CMOS Version Function Comparison	1-16
1.6 Difference between uCOM-87AD CMOS and NMOS Versions	1-17
CHAPTER 2 PIN FUNCTION	2-1
2.1 Normal Operation Mode	2-1
2.1.1 PA7-PA0 (Port A) - 3-state input/output	2-1
2.1.2 PB7-PB0 (Port B) - 3-state input/output	2-1
2.1.3 PC7-PC0 (Port C) - 3-state input/output	2-2
2.1.4 PD7-PD0 (Port D) - 3-state input/output	2-4
2.1.5 PF7-PF0 (Port F) - 3-state input/output	2-5
2.1.6 WR (Write Strobe) - 3-state output	2-8
2.1.7 RD (Read Strobe) - 3-state output	2-8
2.1.8 ALE (Address Latch Enable) - 3-state output	2-8
2.1.9 MODE0 and MODE1 (Mode) - Input/output	2-8
2.1.10 NMI (Nonmaskable Interrupt) - Input	2-9
2.1.11 INT1 (Interrupt Request) - Input	2-10

Pages

2.1.12	AN7-AN0 (Analog Input) - Input	2-10
2.1.13	V_{AREF} (Reference Voltage) - Input	2-10
2.1.14	AV_{DD} (Analog V_{DD})	2-10
2.1.15	AV_{SS} (Analog V_{SS})	2-10
2.1.16	$STOP$ (Stop Control Input)	2-10
2.1.17	X1 and X2 (Crystal)	2-10
2.1.18	<u>RESET</u> (Reset) - Input	2-11
2.1.19	V_{DD}	2-11
2.1.20	V_{SS}	2-11
2.1.21	IC	2-11
2.2	EPROM MODE	2-12
2.2.1	A14-A0 (Address) - Input	2-12
2.2.2	07-00 (Data) - Input/output	2-12
2.2.3	\overline{CE} (Chip Enable) - Input	2-12
2.2.4	\overline{OE} (Output Enable) - Input	2-12
2.2.5	MODE1 and MODE0 (Mode) - Input	2-12
2.2.6	<u>RESET</u> (Reset) - Input	2-12
2.2.7	V_{PP}	2-13
2.2.8	V_{DD}	2-13
2.2.9	V_{SS}	2-13
2.3	Pin Input/Output Circuits	2-14
2.4	Pin Mask Option (uPD78C14A uPD78C12A and 78C11A only)	2-20
CHAPTER 3	INTERNAL BLOCK FUNCTION	3-1
3.1	Registers	3-1
3.2	Mode Registers	3-3
3.3	Arithmetic and Logic Unit (ALU)	3-5
3.4	Program Status Word (PSW)	3-5
3.5	Memory	3-8

	<u>Pages</u>
3.5.1 uPD78C14/78C14A/uPD78C12A/uPD78C11/78C11A/ uPD78C10/78C10A memory configuration	3-8
3.5.2 uPD78CP14 memory configuration	3-15
3.6 Timer	3-19
3.7 Timer/Event Counter	3-19
3.8 Serial Interface	3-19
3.9 Analog/Digital Converter	3-19
3.10 Input Control	3-20
3.11 Zero Cross Detector	3-20
CHAPTER 4 PORT FUNCTION	4-1
4.1 Port A (PA7-PA0)	4-1
4.2 Port B (PB7-PB0)	4-5
4.3 Port C (PC7-PC0)	4-6
4.4 Port D (PD7-PD0)	4-11
4.5 Port F (PF7-PF0)	4-13
CHAPTER 5 TIMER FUNCTION	5-1
5.1 Timer Configuration	5-1
5.2 Timer Mode Register (TMM)	5-2
5.3 Timer Operation	5-5
CHAPTER 6 TIMER/EVENT COUNTER FUNCTION	6-1
6.1 Timer/Event Counter Configuration	6-1
6.2 Mode Registers	6-7
6.2.1 Timer/event counter mode register (ETMM)	6-7
6.2.2 Timer/event counter output mode	6-10
6.3 Timer/Event Counter Operation	6-14
6.3.1 Interval timer mode	6-15
6.3.2 Event counter mode	6-16

	<u>Pages</u>
6.3.3 Frequency measurement mode	6-17
6.3.4 Pulse width measurement mode	6-19
6.3.5 Programmable square wave output mode	6-20
6.3.6 Timer/event program examples	6-22
CHAPTER 7 SERIAL INTERFACE FUNCTION	7-1
7.1 Serial Interface Configuration	7-1
7.2 Serial Mode Registers	7-4
7.2.1 Serial mode high register (SMH)	7-5
7.2.2 Serial mode low register (SML)	7-9
7.2.3 Serial mode register initialization	7-12
7.3 Serial Interface Operation	7-12
7.3.1 Asynchronous mode	7-12
7.3.2 Synchronous mode	7-22
7.3.3 I/O interface mode	7-26
7.3.4 Serial Interface program example	7-31
CHAPTER 8 ANALOG-TO-DIGITAL CONVERTER FUNCTION	8-1
8.1 Analog/Digital Converter Configuration	8-1
8.2 A/D Channel Mode Register (ANN)	8-4
8.3 Analog-to-Digital Converter Operation	8-7
8.3.1 Scan mode	8-7
8.3.2 Select mode	8-8
8.3.3 A/D converter operation control	8-10
8.3.4 Analog-to-digital converter program example ..	8-11
CHAPTER 9 INTERRUPT CONTROL FUNCTION	9-1
9.1 Interrupt Control Circuit Configuration	9-2
9.2 External Interrupt Sampling	9-8
9.3 Nonmaskable Interrupt Operation	9-10
9.4 Maskable Interrupt Operation	9-13

	<u>Pages</u>
9.5 SOFTI Instruction Interrupt Operation	9-18
9.6 Interrupt Wait Time	9-19
9.7 Multiple Interrupts	9-21
CHAPTER 10 CONTROL FUNCTION	10-1
10.1 Standby Function	10-1
10.1.1 HALT mode	10-1
10.1.2 HALT mode release	10-2
10.1.3 Software STOP mode	10-4
10.1.4 Software STOP mode release	10-6
10.1.5 Hardware STOP mode	10-8
10.1.6 Hardware STOP mode release	10-12
10.1.7 Low supply voltage data retention mode	10-15
10.2 Reset Function	10-16
10.3 Clock Generator	10-18
CHAPTER 11 EXTERNAL DEVICE ACCESS AND TIMING	11-1
11.1 uPD78C14/78C14A/uPD78C12A/uPD78C11/78C11A External Device Access	11-1
11.1.1 MEMORY MAPPING register (MM)	11-5
11.1.2 Memory expansion example	11-7
11.1.3 Peripheral device connection example	11-9
11.2 uPD78C10/78C10A External Device Access	11-13
11.2.1 MM register setting	11-13
11.3 Timing	11-16
CHAPTER 12 INTERNAL EPROM ACCESS (uPD75CP14 ONLY)	12-1
CHAPTER 13 INSTRUCTION SET	13-1
13.1 Operand Identifiers and Description	13-1
13.2 Explanation of Operand Code Symbols	13-4
13.3 Instruction Addressing	13-5

	<u>Pages</u>
13.3.1 Register addressing	13-5
13.3.2 Immediate addressing	13-6
13.3.3 Direct addressing	13-7
13.3.4 Relative addressing	13-8
13.3.5 Extended relative addressing	13-9
 13.4 Operand Addressing	13-10
13.4.1 Register addressing	13-10
13.4.2 Register indirect addressing	13-12
13.4.3 Auto increment addressing	13-13
13.4.4 Auto decrement addressing	13-14
13.4.5 Double auto increment addressing	13-16
13.4.6 Base addressing	13-17
13.4.7 Base index addressing	13-18
13.4.8 Working register addressing	13-19
13.4.9 Accumulator indirect addressing	13-20
13.4.10 Immediate addressing	13-21
13.4.11 Extended immediate addressing	13-21
13.4.12 Direct addressing	13-23
 13.5 Number of States Required when Instruction is Skipped	13-25
 13.6 Explanation of Instructions	13-26
13.6.1 8-bit data transfer instructions	13-26
13.6.2 16-bit data transfer instructions	13-38
13.6.3 8-bit operation instructions	13-52
13.6.4 8-bit operation instructions	13-66
13.6.5 Immediate data operation instructions	13-77
13.6.6 Working register operation instructions	13-105
13.6.7 16-bit operation instructions	13-120
13.6.8 Multiplication and division instructions	13-129
13.6.9 Increment and decrement instructions	13-130
13.6.10 Miscellaneous operation instructions	13-134
13.6.11 Rotation and shift instructions	13-136
13.6.12 Jump instructions	13-144

	<u>Pages</u>
13.6.13 Call instructions	13-149
13.6.14 Return instructions	13-154
13.6.15 Skip instructions	13-156
13.6.16 CPU control instructions	13-160
13.7 String Effect Instructions	13-163
APPENDIX A INTRODUCTION OF PIGGY-BACK PRODUCT	A-1
1. Pin Function	A-3
1.1 Lower pins	A-3
1.2 Upper pins	A-5
2. Memory Configuration	A-7
3. MEMORY MAPPING Register (MM)	A-11
4. Interface with EPROM	A-13
APPENDIX B DEVELOPMENT TOOLS	A-14
APPENDIX C INSTRUCTION INDEX	A-16

PREFACE

The 8-bit single chip microcomputer uCOM-87AD family provides the 15 products listed below.

This manual explains the CMOS version.

Product name	Memory		Instruction cycle	Structure	
	ROM	RAM			
uPD7811	Internal 4K bytes	Internal 256 bytes	1us/12 MHz	(Note 1) NMOS	
uPD7811H			0.8us/15 MHz		
uPD7810			1us/12 MHz		
uPD7810H			0.8us/15 MHz		
uPD78PG11E			1us/12MHz		
uPD78PG11E-15			0.8us/15 MHz		
uPD78C11	Internal 4K bytes	External	0.8us/15 MHz (Note 2) (1us/12 MHz)	CMOS	
uPD78C11A	External				
uPD78C10					
uPD78C10A					
uPD78C17	Internal 1K byte	Internal 256 bytes			
uPD78C12A (Note 3)					
uPD78C14					
uPD78C14A					
uPD78C18	Internal 32K bytes	Internal 1K byte	0.8us/15 MHz		
uPD78CG14	4K-, 8K-, 16K -byte EPROM is mounted (Note 3)	Internal 256 bytes			
uPD78CP14	16K-byte EPROM and one time PROM onchipped				
uPD78CP18	32K-byte EPROM	Internal 1K byte			

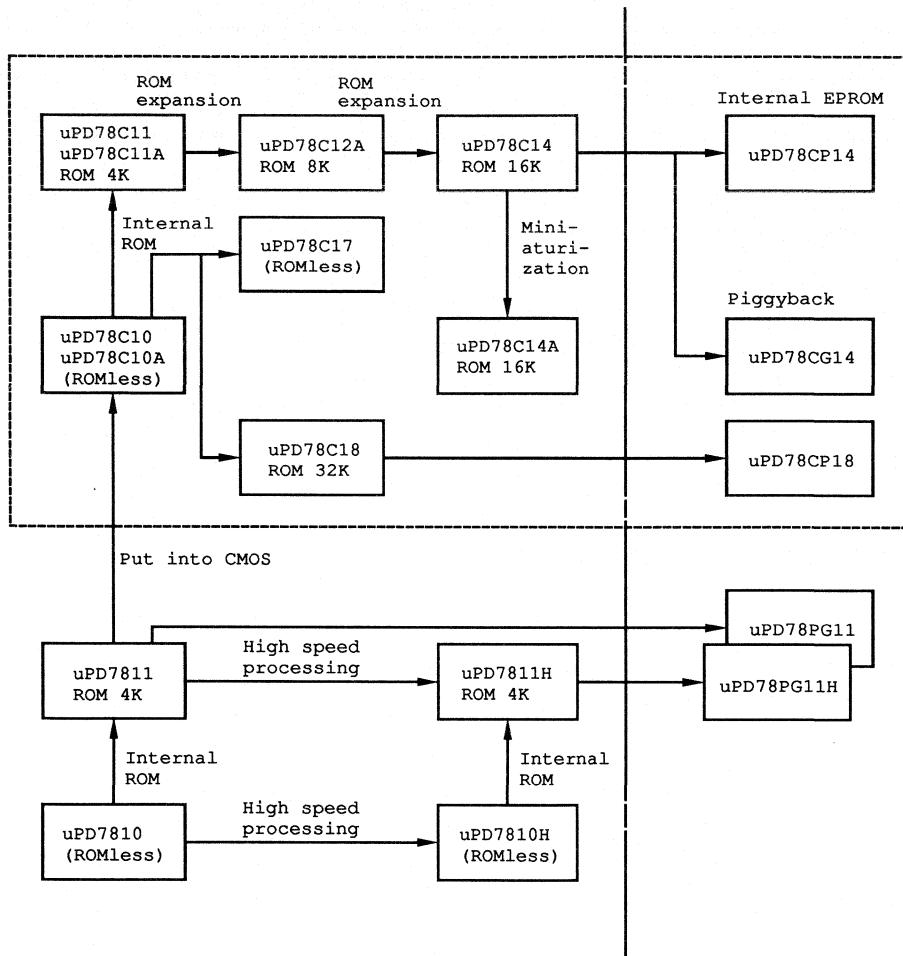
Note 1: For the NMOS version, refer to the uCOM-87AD uPD7811 User's Manual.

Note 2: uPD78C11 and uPD78C10 whose instruction cycle is 1 us at 12 MHz are also available. For details, refer to the uPD78C11 and uPD78C10 Data Sheets.

Note 3: The memory capacity can be selected by software.

The following uCOM-87Ad family data, materials, and documents are provided: (Use the appropriate according to the products that you use.)

uCOM-87AD Family Development Diagram



CHAPTER 1 GENERAL DESCRIPTION

Each product of uCOM-87AD CMOS version, the following hardware devices are integrated on one chip:

- ROM (except for uPD78C10)
- 256 x 8-bit RAM
- 16-bit arithmetic and logic unit (ALU)
- A/D converter
- Multifunctional timer/event counter
- General purpose serial interface, and others

Maintaining compatibility with the conventional NMOS products, the uCOM-87AD series CMOS version provides the enhanced standby function and contains rich packages, thus enabling miniaturization and low power consumption of the system.

The features of the products are as listed below:

Product name	Internal ROM	Expanded external memory	Remarks
uPD78C14			
uPD78C14A	16K x 8 bits	Up to 48K bytes	Internal pull-up resistor can be contained. Small package
uPD78CP14			Internal ROM is EPROM or one-time PROM
uPD78C12A	8K x 8 bits	Up to 56K bytes	Internal pull-up resistor can be contained.
uPD78C11			-
uPD78C11A	4K x 8 bits	Up to 60K bytes	Internal pull-up resistor can be contained.
uPD78C10	Not included	Up to 64K bytes	-
uPD78C10A			

The user can write or rewrite programs into the uPD78CP14 which contains internal EPROM or one-time PROM. This feature enables operation check on the system and TAT reduction during small production.

Applications

- o Business machines and office automation
 - PPC, printer, electronic typewriter, ECR, facsimile, barcode reader, and others
- o Car electronics
 - Engine control, car air conditioner, and others
- o Household electric appliances
 - Air conditioner, video tape recorder, and others
- o Miscellaneous
 - Electronic musical instrument, point of sales (POS), inverter, electronic sewing machine, auto focus camera, and others

Remarks: The manual explains the uPD78C14/78C14A/78CP14, uPD78C12A, uPD78C11/78C11A, and uPD78C10/78C10A. The uPD78CG14 is introduced in Appendix A. Unless otherwise noted, the uPD78C11 is described as a typical product. Replace the uPD78C11 with the uPD78C14/78C14A/78CP14, uPD78C11A, uPD78C12A, or uPD78C10/78C10A in the description according to the target devices.

1.1 Features

- o Single chip microcomputer (μCOM-87AD)
- o Rich 159 instructions
 - uCOM-87 upward compatible
 - Multiplication and division instructions and 16-bit operation instructions
- o Instruction cycle: 0.8 us at 15 MHz (Note 1)
- o Program (ROM) capacity: 16384 words x 8 bits (uPD78C14/78C14A)
 - 8192 words x 8 bits (uPD78C12A)
 - 4096 words x i bits (uPD78C11/78C11A)
- o Data (RAM) (Note 2) capacity: 256 words x 8 bits
- o A maximum of 64K-byte memory (ROM/RAM) space can be directly addressed.
- o High precision 8-bit A/D converter
 - 8 analog inputs
- o General purpose serial interface
 - Asynchronous mode, synchronous mode, I/O interface mode
- o Multifunctional 16-bit timer/event counter
- o Two 8-bit timers
- o Interrupt function (three external interrupts and eight internal interrupts)
 - One nonmaskable interrupt
 - 10 maskable interrupts
 - Six priority levels and six interrupt addresses
- o I/O lines
 - Input/output port: 40 bits (uPD78C14/78C14A/78C12A/78C11/78C11A)
 - : 28 bits (uPD78C10/78C10A)
- Four edge detection inputs
- o Zero cross detection function
- o Standby function
 - HALT mode
 - Software or hardware STOP mode
- o Internal pull-up resistor can be contained in port pin: PA, PB, and PC only (uPD78C14A/78C12A/78C11A)

- o Clock oscillator is contained.
- o Internal PROM: uPD78CP14R/DW (EPROM version)
uPD78CP14G/CW/L (one-time PROM version)

Note 1: Another version of the uPD78C11/uPD78C10 has instruction cycle 1 us at 12 MHz. For details, refer to the uPD78C11 and uPD78C10 documents.

Note 2: Internal RAM can be used only when the MM register RAE bit is set to 1.

- o CMOS
- o Single power supply
- o Rich package variations
 - Shrink-dual-in-line package
 - Quad-in-line package straight
 - Quad-in-line package
 - QFP
 - Plastic leaded chip carrier

1.2 Ordering Information

1.2.1 uPD78C10/78C10A

<u>Ordering code</u>	<u>Package</u>
uPD78C10CW	64-pin plastic shrink DIP
uPD78C10G-36	64-pin plastic QUIP
uPD78C10G-1B	64-pin plastic QFP (resin thickness: 2.05 mm)
uPD78C10GF-3BE	64-pin plastic QFP (resin thickness: 2.7 mm)
uPD78C10L	68-pin PLCC
uPD78C10ACW	64-pin plastic shrink DIP
uPD78C10AGQ-36	64-pin plastic QUIP
uPD78C10AGF-3BE	64-pin plastic QFP (resin thickness: 2.7 mm)
uPD78C10AL	68-pin PLCC

1.2.2 uPD78C17¹⁾

<u>Ordering code</u>	<u>Package</u>
uPD78C17CW	64-pin plastic shrink DIP
uPD78C17GQ-36	64-pin plastic QUIP
uPD78C17GF-3B	64-pin plastic QFP

Note 1) under development

1.2.3 uPD78C11, 78C11A

<u>Ordering code</u>	<u>Package</u>
uPD78C11CW-XXX	64-pin plastic shrink DIP
uPD78C11G-XXX-36	64-pin plastic QUIP
uPD78C11G-XXX-37	64-pin plastic QUIP straight
uPD78C11G-XXX-1B	64-pin plastic QFP (resin thickness: 2.05 mm)
uPD78C11GF-XXX-3BE	64-pin plastic QFP (resin thickness: 2.7 mm)
uPD78C11L-XXX	68-pin PLCC
uPD78C11ACW-XXX	64-pin plastic shrink DIP
uPD78C11AGQ-XXX-36	64-pin plastic QUIP
uPD78C11AGQ-XXX-37	64-pin plastic QUIP straight
uPD78C11AGF-XXX-3BE	64-pin plastic QFP (resin thickness: 2.7 mm)
uPD78C11AL-XXX	68-pin PLCC

XXX: ROM code

1.2.4 uPD78C12A

<u>Ordering code</u>	<u>Package</u>
uPD78C12ACW-XXX	64-pin plastic shrink DIP
uPD78C12AGQ-XXX-36	64-pin plastic QUIP
uPD78C12AGQ-XXX-37	64-pin plastic QUIP straight
uPD78C12AGF-XXX-3BE	64-pin plastic QFP (resin thickness: 2.7 mm)
uPD78C12AL-XXX	68-pin PLCC

XXX: ROM code

1.2.5 uPD78C14, 78C14A, 78CG14, 78CP14

Ordering code	Package
uPD78C14CW-XXX	64-pin plastic shrink DIP
uPD78C14G-XXX-36	64-pin plastic QUIP
uPD78C14G-XXX-37	64-pin plastic QUIP straight
uPD78C14G-XXX-1B	64-pin plastic QFP (resin thickness: 2.05 mm)
uPD78C14GF-XXX-3BE	64-pin plastic QFP (resin thickness: 2.7 mm)
uPD78C14L-XXX	68-pin PLCC
uPD78C14AG-XXX-AB8	64-pin plastic QFP (interpin pitch: 0.8 mm)
uPD78CG14E	64-pin ceramic piggy-back QUIP
uPD78CP14CW	64-pin plastic shrink DIP
uPD78CP14G-36	64-pin plastic QUIP
uPD78CP14GF-3BE	64-pin plastic QFP (resin thickness: 2.7 mm)
uPD78CP14L	68-pin PLCC
uPD78CP14DW	64-pin ceramic shrink DIP with window
uPD78CP14R	64-pin ceramic QUIP with window

1.2.6 uPD78C18¹⁾, 78CP18¹⁾

Ordering code	Package
uPD78C18CW-XXX	64-pin plastic shrink DIP
uPD78C18GQ-XXX-36	64-pin plastic QUIP
uPD78C18GF-XXX-3BE	64-pin plastic QFP
uPD78CP18GF	64-pin plastic QFP
uPD78CP18	68-pin ceramic LCC with window

Note 1) under development

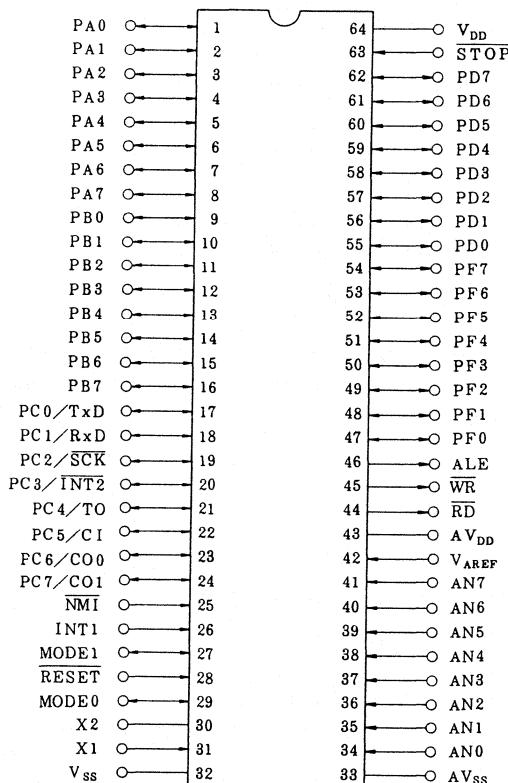
XXX: ROM code

1.3 Pin Configuration

1.3.1 Shrunked-dual-in-line package, quad-in-line package straight (37), and quad-in-line package (36)

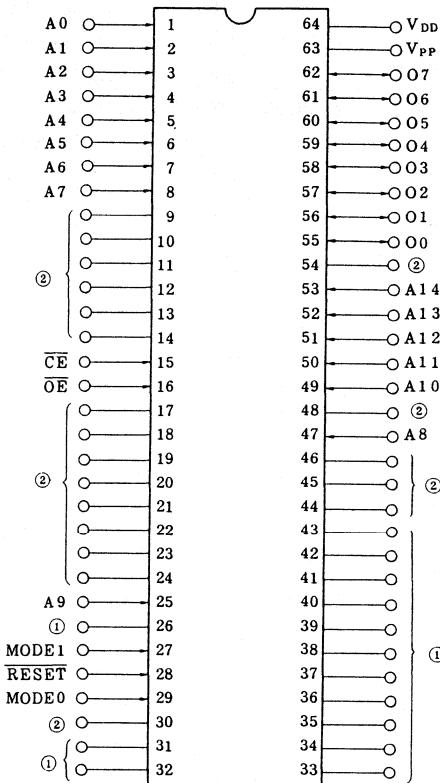
(1) Normal operation mode

Pin configuration (top view)



PA7-0	:	PortA	X1, X2	:	Crystal
PB7-0	:	PortB	AN7-0	:	Analog Input
PC7-0	:	PortC	<u>RD</u>	:	Read Strobe
PD7-0	:	PortD	<u>WR</u>	:	Write Strobe
PF7-0	:	PortF	ALE	:	Address Latch Enable
<u>NMI</u>	:	Non Maskable Interrupt	<u>RESET</u>	:	Reset
INT1	:	Interrupt Request	V _{AREF}	:	Reference Voltage
MODE0, 1	:	Mode0, 1	<u>STOP</u>	:	Stop Control Input

(2) EPROM mode (uPD78CP14 only)

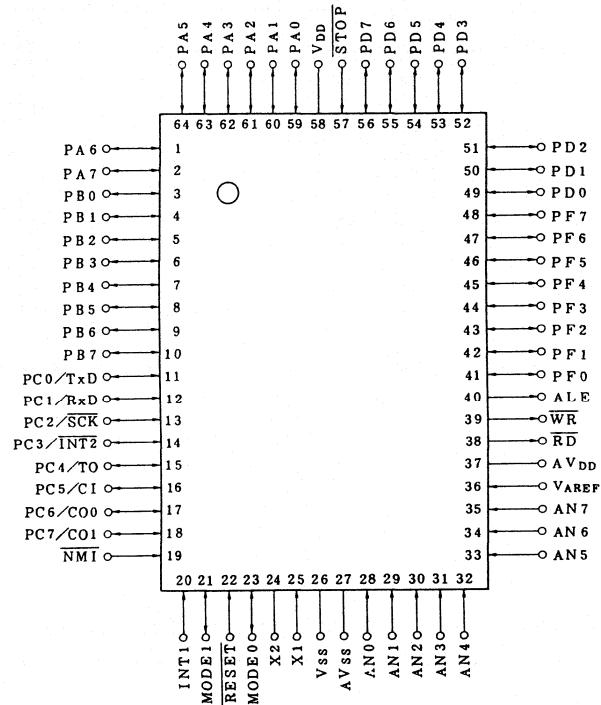


Caution: (1) : Connect the pin directly to V_{SS}.

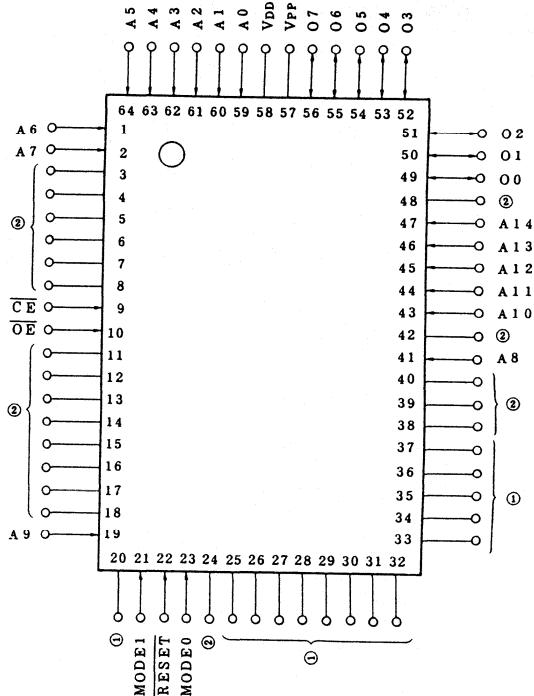
(2) : Pull low to the V_{SS} potential with a resistor for each pin. (10 kΩ)

1.3.2 QFP (1B/3BE)

(1) Normal operation mode



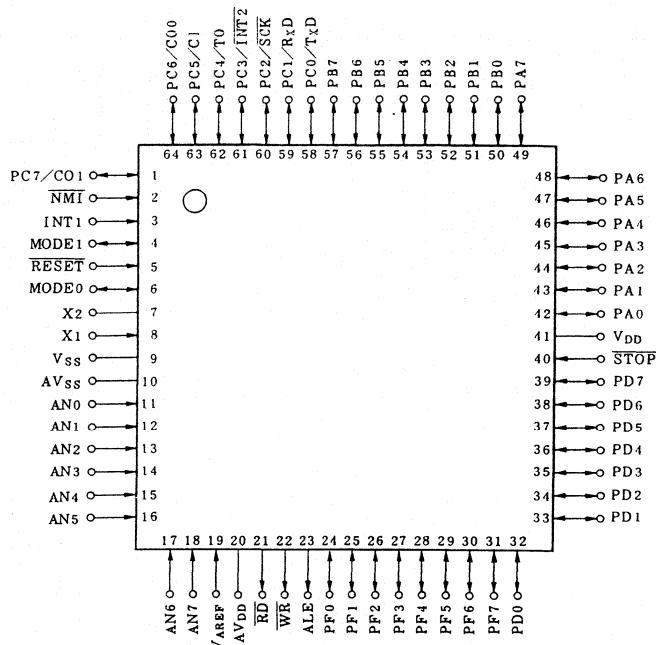
(2) EPROM mode (uPD78CP14 only)



Caution: ① : Connect the pin directly to V_{SS}.

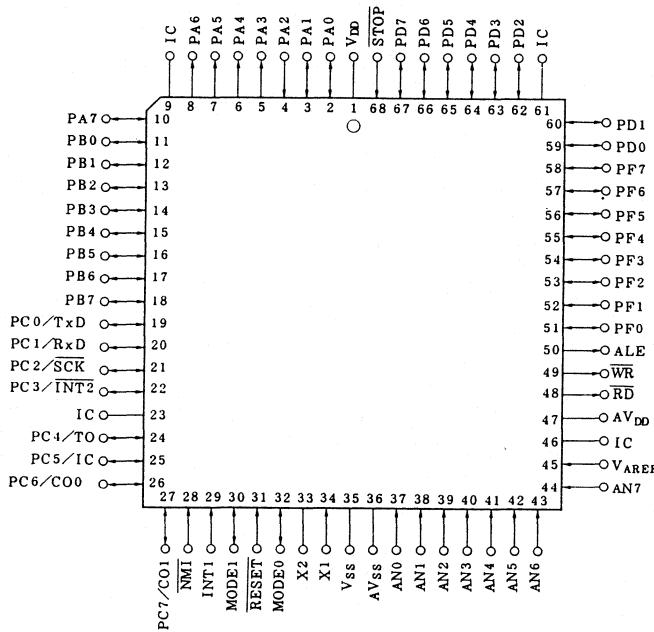
② : Pull low to the V_{SS} potential with a resistor for each pin.

1.3.3 QFP (AB8)

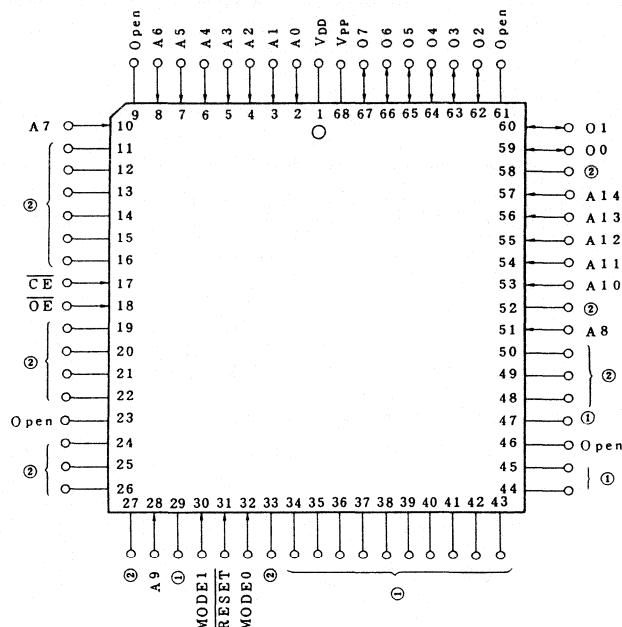


1.3.4 Plastic leaded chip carrier

(1) Normal operation mode



(2) EPROM mode (uPD78CP14 only)

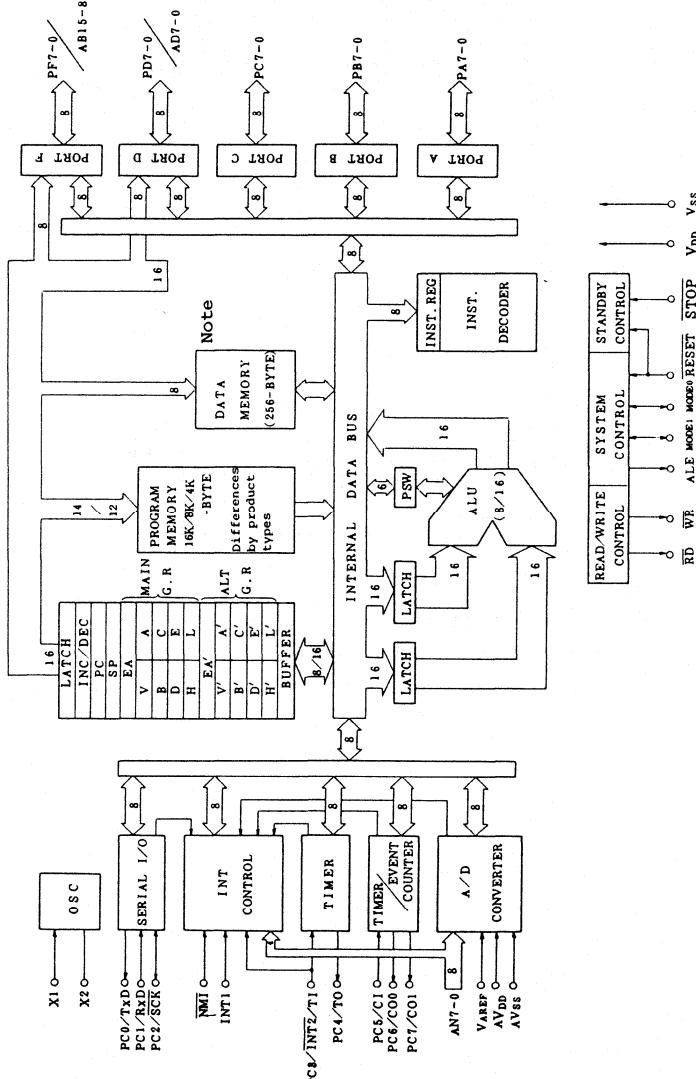


Caution: Open: Leave the pin unconnected.

① : Connect the pin directly to V_{SS}.

② : Pull low to the V_{SS} potential with a resistor for each pin.

1.4 Block Diagram



Note: When RAE bit of MM register is only set to 1, data memory is usable. When RAE bit of MM register is reset to 0, the external memory is required.

1.5 uCOM-87AD CMOS Version Function Comparison

	78C10, 78C10A	78C11, 78C11A	78C12A	78C14	78C14A	78CG14	78CP14 (Note)
Number of Instructions	159						
Minimum instruction cycle							
ROM	-	4K bytes	8K bytes	16K bytes	EPROM is mounted	Internal EPROM is contained	
RAM				256 bytes (RAE of MN register is only set to 1)			
Number of Interrupts	External			3			
	Internal			8			
Timer/counter				8 bits x two/16 bits x one			
A/D converter				8-bit x 8 analog inputs			
Serial Interface				UART (full duplex)/clock synchronization			
I/O				8-bit general purpose input/output port x 5			
Power consumption	Operation	MAX. 25mA (5 V)		MAX. 30 mA (5 V)			
	HALT mode	MAX. 13mA (5 V)		MAX. 15 mA (5 V)			
	STOP mode	MAX. 50 uA (5 V)/MAX. 15 uA (2.5 V)				Undefined	
Package				64-pin quad-in-line package - 64-pin shrinked-dual-in-line package - 64-pin quad-flat package - 68-pin plastic lead chip carrier	64-pin quad-in-line package 0.8 mm pitch 14 mm x 14 mm	64-pin quad-in-line package - 64-pin shrinked-dual-in-line package - 64-pin quad-flat package - 68-pin plastic lead chip carrier	64-pin quad-in-line package - 64-pin shrinked-dual-in-line package - 64-pin quad-flat package - 68-pin plastic lead chip carrier
Miscellaneous	Internal pull-up resistor can be contained			uPD78C11/78C12A/78C14 change by software			

1.6 Difference between μCOM-87AD CMOS and NMOS Version

Process Item	CMOS	NMOS
Product name	uPD78C14/78C14A/78C12A/78C11/78C11A 78C10/78C10A	uPD7811/7810
Number of instructions	159 (STOP instruction is added)	158
Number of special registers	28 (ZCM register is added)	27
Standby function	HALT mode, software STOP mode, or hardware STOP mode. 256-byte data in internal RAM is retained at low supply voltage (2.5 V) during the software or hardware STOP mode.	32-byte data in 256-byte internal RAM is retained at low supply voltage (3.2 V).
Number of HLT instruction states	12	11
HALT mode	CPU operation ALE	Stop Low High
Zero cross detector self-bias control	Self-bias control can be performed as specified in the ZCM register.	Self-bias control cannot be performed.
NMI, RESET noise removing method	Alalog delay	Clock sampling
A/D converter	Operation can be stopped by using the V _{AREF} pin.	Operation cannot be stopped.
Operation during RESET	RD/WR	High
	ALE	Output
	PD/PF (78C10 /7810)	0 is output to the pins selected for the address bus; others become high impedance.
Power consumption	Operation	60mW Typ. (12 MHz) (Note 1)
	Standby	5 uW Typ.
		750 mW Typ.
		4.8 mW Typ.

(to be continued)

Item \ Process	CMOS	NMOS
Package (Note 2)	64-pin plastic shrinked-dual-in-line package 64-pin plastic quad-in-line package straight 64-pin plastic quad-in-line package 64-pin plastic QFP (bent leads) 68-pin plastic leaded chip carrier	64-pin plastic shrinked-dual-in-line package 64-pin plastic quad-in-line package straight 64-pin plastic quad-in-line package
Pin connection (except for flat or PLCC)	V_{DD} (pin 64) $STOP$ (pin 63)	V_{CC} (pin 64) V_{DD} (pin 63)

Note 1: uPD78C14: 80 mW (15 MHz)

Note 2: The pin connection and pin numbers vary depending on the package type.

Caution: In addition, they differ in electrical characteristics, oscillator characteristics, and some internal operation timings. Be cautions about the matter in examination of direct replacement between the uPD78C14/78C14A/78C12A/78C11/78C11A/78C10/78C10A and uPD7811/7810.

CHAPTER 2 PIN FUNCTION

The uPD78C14/78C12A/78C11/78C11A/78C10/78C10A pin function is always executed in the normal operation mode. The uPD78CP14 pin function is executed in the normal operation or EPROM mode. The EPROM mode is selected by setting the MODE1 pin low and the MODE0 pin high.

2.1 Normal Operation Mode

2.1.1 PA7-PA0 (Port A) - 3-state input/output

PA7-PA0 are 8-bit input/output pins to/from port A (8-bit input/output port with an output latch). The input or output mode can be selected bit-wise by using the MODE A register (MA).

When RESET is input, PA7-PA0 become input port pins (output high impedance). During the hardware STOP mode, PA7-PA0 become output high impedance.

A pull-up resistor can be contained bit-wise in the uPD78C11A, uPD78C12A, and uPD78C14A.

2.1.2 PB7-PB0 (Port B) - 3-state input/output

PB7-PB0 are 8-bit input/output pins to/from port B (8-bit input/output port with an output latch). The input or output mode can be selected bit-wise by using the MODE B register (MB).

When RESET is input, PB7-PB0 become input port pins (output high impedance). During the hardware STOP mode, PA7-PA0 become output high impedance.

A pull-up resistor can be contained bit-wise in the uPD78C11A, uPD78C12A, and uPD78C14A.

2.1.3 PC7-PC0 (Port C) - 3-state input/output

PC7-PC0 serve as 8-bit input/output pins to/from port C (8-bit input/output port with an output latch). The pins also serve as control signal pins.

The port mode or control signal input/output mode can be selected bit-wise for the PC7-PC0 pins by using the MODE CONTROL C register (MCC). (See Table 2-1.)

A pull-up resistor can be contained bit-wise in the uPD78C11A, uPD78C12A, and uPD78C14A.

Table 2-1 PC7-PC0 Operation

	MCCn = 0	MCCn = 1
	Port mode	Control signal input/output mode
PC0	Input/output port	TxD output
PC1	Input/output port	RxD input
PC2	Input/output port	SCK input/output
PC3	Input/output port	INT2/TI input
PC4	Input/output port	TO output
PC5	Input/output port	CI input
PC6	Input/output port	CO0 output
PC7	Input/output port	CO1 output

(n = 0-7)

(1) Port mode

When the port mode is selected for PC7-PC0 by using the MODE CONTROL C register, the input or output port mode can be selected bit-wise for PC7-PC0 by using the MODE C register (MC).

(2) Control signal input/output mode

PC7-PC0 can be used as control pins bit-wise by setting the MODE CONTROL C register (MCC). The control pin function is described below:

(a) TxD (Transmit Data) - Output

TxD is a serial data transmission pin from which the serial register contents are output.

(b) RxD (Receive Data) - Input

RxD is a serial data reception pin. Data on RxD is loaded into the serial register.

(c) SCK (Serial Clock) - Input/output

SCK is a serial input/output data control clock pin. When internal clock is used, the pin is placed in the output mode; when external clock is used, the pin is placed in the input mode.

(d) INT2/TI (Interrupt Request/Timer Input) - Input

INT2/TI serves as an edge-triggered (falling edge) maskable interrupt input pin or a timer external clock input pin. It can also be used as an AC signal zero cross detection pin.

Caution: If an internal pull-up resistor is contained in uPD78C11A, uPD78C12A or uPD78C14A PC3, the zero cross function cannot be expected.

(e) TO (Timer Output) - Output

Square wave with one timer count time or internal clock (ϕ_3) period as a half period is output.

(f) CI (Count Input) - Input

Timer/event counter external pulses are input.

(g) CO0 and CO1 (Counter Output) - Output

CO0 and CO1 are output pins of programmable square wave generated by the timer/event counter.

When RESET is input, PC7-PC0 become input port pins (output high impedance). During the hardware STOP mode, PC7-PC0 become output high impedance.

2.1.4 PD7-PD0 (Port D) - 3-state input/output

■ uPD78C14/78C14A/78C12A/78C11/78C11A

PD7-PD0 are 8-bit input/output pins to/from port D (8-bit input/output port with an output latch). The pins also serve as multiplexed address/data bus pins to access external expanded memory.

Either of the following modes can be selected for the PD7-PD0 pins by setting the MEMORY MAPPING register:

(1) Port mode

PD7-PD0 serve as input/output pins to/from D; the input or output mode can be selected in byte (8-bit) units.

(2) Expansion mode

To use an external device (program memory, data memory, or peripheral device) in addition to internal memory, PD7-PD0 are used as multiplexed address/data bus pins (AD7-AD0). When an external device reference instruction is executed, the pins output low-order address information of the external device address at the first state of external device reference machine cycle of the instruction and become bidirectional 8-bit data bus at the second and third states. PD7-PD0 become high impedance not during execution of external device reference instruction.

Caution 1: The internal address bus contents are output to the PD7-PD0 pins used as the address/data bus as they are at the first state of every machine cycle in synchronization with ALE.

Caution 2: A program which changes the port D operation mode dynamically cannot be emulated by an emulator. Thus, do not change the once set mode to another mode.

When the RESET is input, PD7-PD0 becomes input port pins (output high impedance). During the hardware STOP mode, PD7-PD0 become output high impedance.

■ uPD78C10/78C10A

PD7-PD0 serve only as multiplexed address/data bus pins to access external memory.

The pins output the low-order eight bits of a memory address at the first state and become bidirectional 8-bit data bus at the second and third states.

When the RESET signal is low or during the hardware STOP mode or the standby mode (HALT or STOP), PD7-PD0 become output high impedance.

Caution: Port D can be used only as address/data bus pins.

2.1.5 PF7-PF0 (Port F) - 3-state input/output

■ uPD78C14/78C14A/78C12A/78C11/78C11A

PF7-PF0 are 8-bit input/output pins to/from port F (8-bit input/output port with an output latch). The pins also serve as address output pins (AB15-AB8) to access external expanded memory.

Either of the following modes can be selected for PF7-PF0 pins by setting the MEMORY MAPPING register:

(1) Port mode

PF7-PF0 serve as input/output pins to/from port F; the input or output mode can be selected bit-wise by using MODE F register.

(2) Expansion mode

To use an external device in addition to internal memory, PF7-PF0 are used as address bus pins (AB15-AB8) according to the size of the externally additional device, as listed in Table 2-2. When an external device reference instruction is executed, the pins output high-order address information of the external device in the external device reference machine cycle of the instruction.

Caution: The internal address bus contents are output to the PF7-PF0 pins used as address bus pins as they are in every machine cycle.

The pins not specified as address output pins are placed in the port mode.

Caution: A program which changes the port F operation mode dynamically cannot be emulated by an emulator. Thus, change the once set mode to another mode.

Table 2-2 PF7-PF0 Operation (uPD78C14/78C14A/78C12A/78C11/
78C11A)

PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0	External address space
Port	Within 256 bytes							
Port	Port	Port	Port	AB11	AB10	AB9	AB8	Within 4K bytes
Port	Port	AB13	AB12	AB11	AB10	AB9	AB8	Within 16K bytes
AB15	AB14	AB13	AB12	AB11	AB10	AB9	AB8	Within 48K, 56K or 60K bytes (Note)

Note: 48K = uPD78C14/78C14A, 56K = uPD78C12A, 60K = uPD78C11/78C11A

In the reset state (RESET input = low) or during the hardware STOP mode (STOP input = low), the PF7-PF0 pins become output high impedance. After this, when RESET input or STOP input is restored high, the PF7-PF0 pins are placed in the address bus or port mode according to how the MODE1 and MODE0 pins are set.

■ uPD78C10/78C10A

PF7-PF0 can be specified as address bus pins (AB15-AB8) according to the size of the external memory by setting the MODE0 and MODE1 pins. The remaining pins can be used as general purpose input/output port. (See Table 2-3.)

Table 2-3 PF7-PF0 Operation (uPD78C10/78C10A)

MODE1	MODE0	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0	External address space
0	0	Port	Port	Port	Port	AB11	AB10	AB9	AB8	4K byte
0	1	Port	Port	AB13	AB12	AB11	AB10	AB9	AB8	16K byte
1	1	AB15	AB14	AB13	AB12	AB11	AB10	AB9	AB8	64K byte

In the reset state (RESET input = low) or during the hardware STOP mode (STOP input = low), the PF7-PF0 pins become output high impedance. After this, when RESET input or STOP input is restored high, the PF7-PF0 pins are placed in the address bus or port mode according to how the MODE1 and MODE0 pins are set.

Caution: A program which changes the port F operation mode dynamically cannot be emulated by an emulator. Thus, do not change the once set mode to another mode.

2.1.6 WR (Write Strobe) - 3-state output

WR is a strobe signal output for external memory write operation. It goes high except in the external memory data write machine cycle. When the RESET signal is low or during the hardware STOP mode, WR becomes output high impedance.

2.1.7 RD (Read Strobe) - 3-state output

RD is a strobe signal output for external memory read operation. It goes high except in the external memory read machine cycle. When the RESET signal is low or during the hardware Stop mode, RD becomes output high impedance.

2.1.8 ALE (Address Latch Enable) - 3-state output

ALE is a strobe signal to externally latch low-order address information output to the PD7-PD0 pins to access external memory. When the RESET signal is low or during the hardware STOP mode, ALE becomes output high impedance.

2.1.9 MODE0 and MODE1 (Mode) - Input/output

■ uPD78C14/78C14A/78C12A/78C11/78C11A

The MODE0 pin is set to 0 (low). The MODE1 pin is pulled high (1) with a pull-up resistor. Pull-up resistor R is $4 \leq R \leq 0.4 t_{CYC}$ [$k\Omega$]. (t_{CYC} is in ns units.)

■ uPD78C10/78C10A

The size of externally installed memory can be selected among 4K, 16K, and 64K bytes by setting the MODE0 and MODE1 pins.

Table 2-4 MODE0 and MODE1 Function (uPD78C10/78C10A)

MODE1	MODE0	External address space
0	0	4K byte (addresses 0000H-0FFFH)
0	1 (Note)	16K byte (addresses 0000H-3FFFH)
1 (Note)	1 (Note)	64K byte (addresses 0000H-FEFFFH)

Note: Pull high with an pull-up resistor. Pull-up resistor R is $4 \leq R \leq 0.4 t_{CYC}$ [$k\Omega$]. (t_{CYC} is in ns units.)

If the MODE0 and MODE1 pins are pulled high (1) with pull-up resistor, control signal is output is synchronization with ALE.

The MODE0 and MODE1 pin input signals are sampled periodically, and the mode is set.

2.1.10 NMI (Nonmaskable Interrupt) - Input

NMI is an edge-triggered (falling edge) nonmaskable interrupt input pin.

2.1.11 INT1 (Interrupt Request) - Input

INT1 is an edge-triggered (rising edge) maskable interrupt input pin. It can also be used as an AC input zero cross detection pin.

2.1.12 AN7-AN0 (Analog Input) - Input

AN7-AN0 are 8-bit analog input pins to the A/D converter. AN7-AN4 can also be used as input pins for falling edge detection. When the falling edge is detected, the test flag is set to 1.

2.1.13 V_{AREF} (Reference Voltage) - Input

V_{AREF} is an A/D converter reference voltage input pin. It can also be used as an A/D converter operation control pin.

2.1.14 AV_{DD} (Analog V_{DD})

AV_{DD} is an A/D converter power supply pin.

2.1.15 AV_{SS} (Analog V_{SS})

AV_{SS} is an A/D converter GND pin.

2.1.16 STOP (Stop Control Input)

STOP is a hardware STOP mode control input pin. When the pin is set low, oscillation stops.

2.1.17 X1 and X2 (Crystal)

X1 and X2 are crystal connection pins for internal clock oscillation. When external clock is supplied, the clock is input to X1. Input the inverted clock of X1 to X2.

2.1.18 RESET (Reset) - Input

RESET is an active low reset signal input pin.

2.1.19 V_{DD}

V_{DD} is a positive power supply pin.

2.1.20 V_{SS}

V_{SS} is a GND potential pin.

2.1.21 IC

IC is an internally connected pin. Leave it unconnected.

Remarks: Plastic leaded chip carrier package only

2.2 EPROM Mode

The EPROM mode can be selected only for the uPD78CP14.

2.2.1 A14-A0 (Address) - Input

A14-A0 are 15-bit address input pins during EPROM write/verify or read.

Since the capacity of uPD78CP14 internal EPROM is 16K bytes, the memory area is addressed by using the low-order 14 bits, A13-A0. Fix A14 low.

2.2.2 O7-O0 (Data) - Input/output

O7-O0 are 8-bit data input/output pins during EPROM write/verify or read.

2.2.3 \overline{CE} (Chip Enable) - Input

\overline{CE} is a chip enable signal input pin.

2.2.4 \overline{OE} (Output Enable) - Input

\overline{OE} is an output enable signal input pin.

2.2.5 MODE1 and MODE0 (Mode) - Input

Set the MODE1 pin low (0) and the MODE0 pin high (1).

2.2.6 \overline{RESET} (Reset) - Input

Set the \overline{RESET} pin low (0).

2.2.7 V_{PP}

V_{PP} is a high voltage apply pin during EPROM write/verify.

When EPROM is read, 1 (high) is input to the pin.

2.2.8 V_{DD}

V_{DD} is a supply voltage apply pin.

2.2.9 V_{SS}

V_{SS} is a GND potential pin.

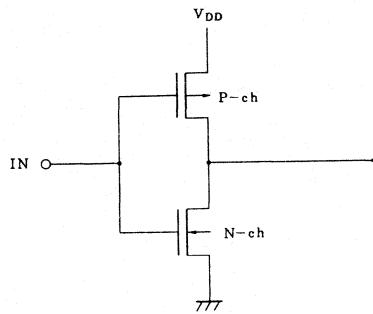
2.3 Pin Input/Output Circuits

Table 2-5 lists the pin input/output circuit type numbers. Schematic drawings of the pin input/output circuits are shown in (1) to (11).

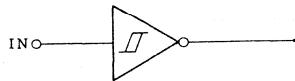
Table 2-5 Pin Type Numbers

Pin name	Type No.	
	uPD78C10/78C10A/78C11/78C14	uPD78C11A/78C12A/78C14A
PA0-7	5	5-A
PB0-7	5	5-A
PC0-1	5	5-A
PC2/SCK	8	8-A
PC3/INT2	10	10-A
PC4-7	5	5-A
PD0-7	5	
PF0-7	5	
NMI	2	
INT1	9	
RESET	2	
RD	4	
WR	4	
ALE	4	
STOP	2	
MODE0	11	
MODE1	11	
AN0-3	7	
AN4-7	12	
V _{AREF}	13	

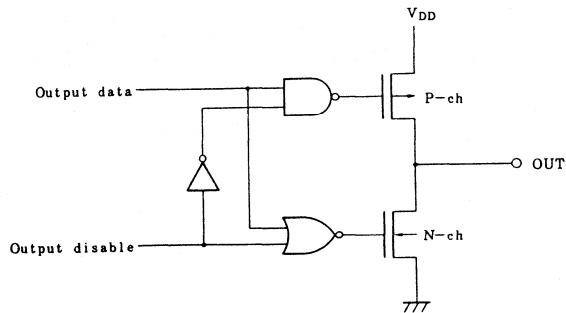
(1) Type 1



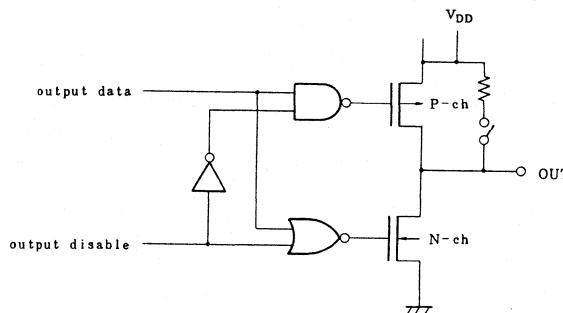
(2) Type 2



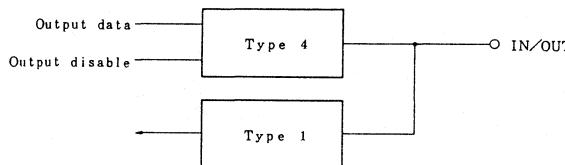
(3) Type 4



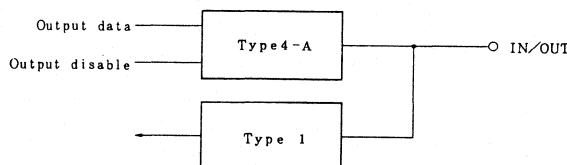
(4) Type 4-A



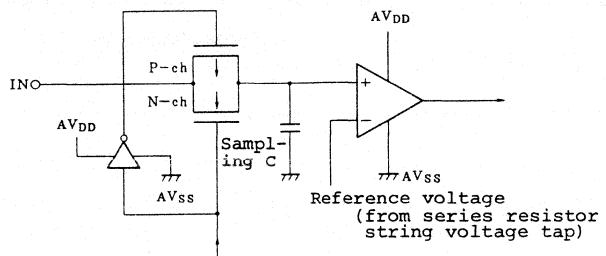
(5) Type 5



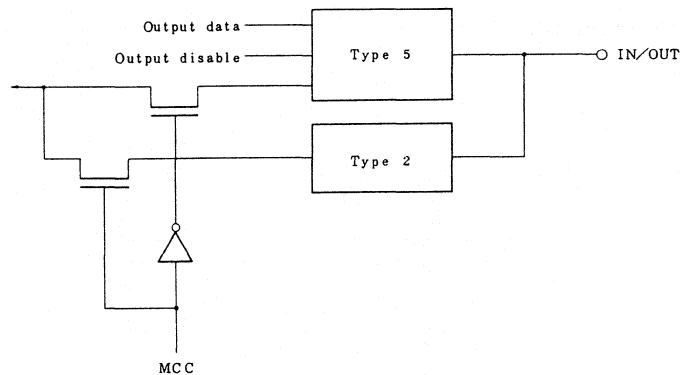
(6) Type 5-A



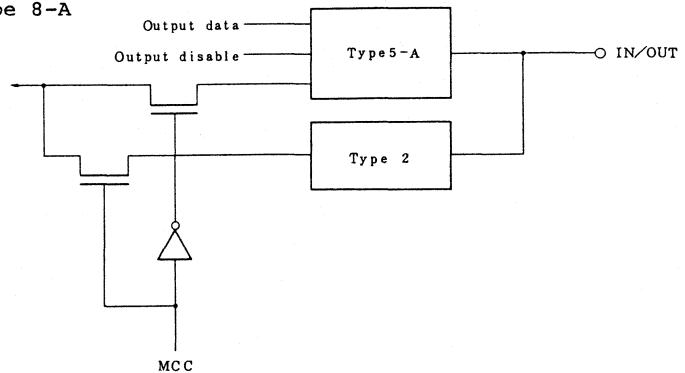
(7) Type 7



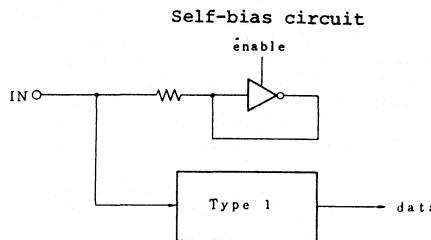
(8) Type 8



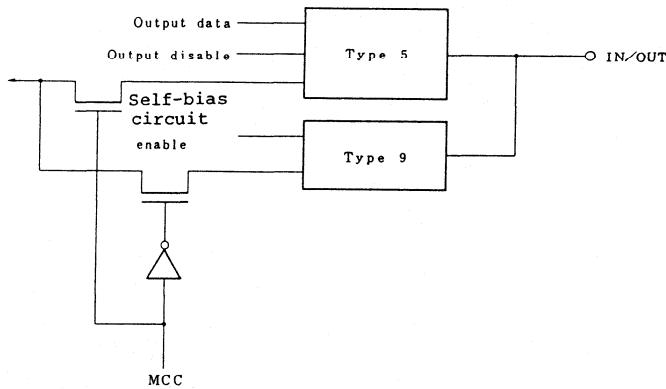
(9) Type 8-A



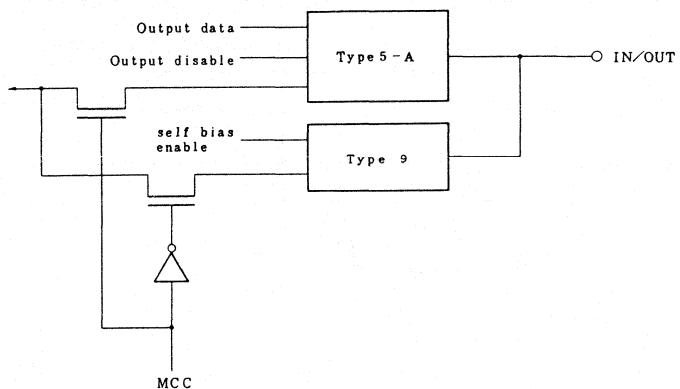
(10) Type 9



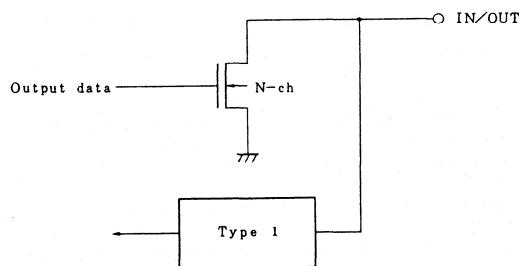
(11) Type 10



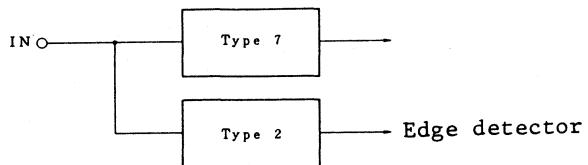
(12) Type 10-A



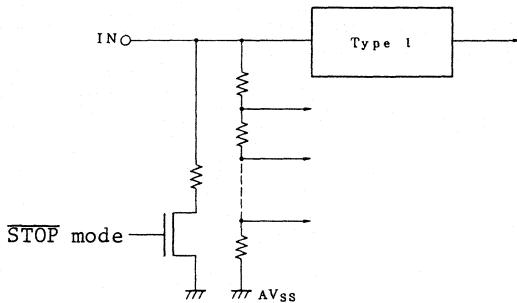
(13) Type 11



(14) Type 12



(15) Type 13



2.4 Pin Mask Option (*uPD78C14A/78C12A/78C11A only*)

The following pins contain mask option which can be selected for each bit according to the purpose:

Pin name	Mask option
PA7-0	① Pull-up resistor is contained.
PB7-0	② Pull-up resistor is not contained.
PC7-0	② Pull-up resistor is not contained.

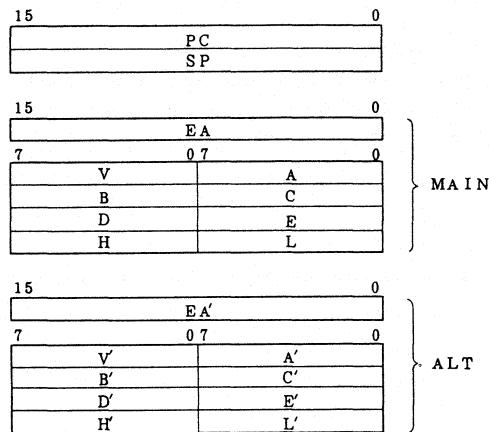
Caution: If a pull-up resistor is contained in PC3, the zero cross function cannot be expected.

CHAPTER 3 INTERNAL BLOCK FUNCTION

3.1 Registers

Central registers are 16 8-bit registers and four 16-bit registers shown in Fig. 3-1 and special registers.

Fig. 3-1 Register Configuration



(1) Accumulator (A)

Since the accumulator system architecture is adopted, data processing such as 8-bit arithmetic and logical operation instruction processing is performed centering around the accumulator. A pair of the accumulator and vector register (V) can be exchanged for a pair of A' and V' of the ALT registers by using the EXA instruction.

(2) Expansion accumulator (EA)

Data processing such as 16-bit arithmetic and logical operation instruction processing is performed centering around the expansion accumulator. The expansion accumulator can be exchanged for the ALT register EA' by using the EXA instruction.

(3) Working register vector register (V)

To set a working area on memory space, the high-order eight bits of the memory address are specified in the V register and the low-order eight bits are specified in immediate data of an instruction. Thus, the memory area specified in the V register can be used as eight working registers each consisting of 256 words.

Since the working register can be specified by using a 1-byte address field, if it can be used as a work area of software flags, parameters, counter, etc., program can be saved. A pair of the V register and accumulator can be exchanged for a pair of the ALT registers V' and A' by using the EXA instruction.

(4) General purpose registers (B, C, D, E, H, and L)

Two sets of general purpose registers, MAIN (B, C, D, E, H, and L) and ALT (B', C', D', E', H', and L') are provided. The registers have the data pointer function as register pairs BC, DE, and HL (B'C', D'E', and H'L') in addition to the accumulator auxiliary register function. Particularly, the four register pairs DE, D'E', HL, and H'L' have the base register function. If the two general purpose register sets are used, when an interrupt occurs, the register contents can be saved in another register set without saving in memory to make interrupt service. Another register set can also be

used as data pointer expansion registers. 1-step auto increment and decrement and 2-step auto increment addressing modes can be used for the register pairs DE, HL, D'E', and H'L' to reduce the processing time. BC, DE, and HL can be exchanged for the ALT registers B'C', D'E', and H'L' at the same time by using the EXX instruction. Only the HL register pair H'L' by using the EXH instruction.

(5) Program counter (PC)

The program counter (PC) is a 16-bit register which retains address information of the next program to be executed. Normally, PC is automatically incremented according to the number of bytes of the fetched instruction. When an instruction involving a branch is executed, immediate data and the register contents are loaded into PC. When RESET is input, PC is cleared to 0000H.

(6) Stack pointer (SP)

The stack pointer (SP) is a 16-bit register which retains top address information of the memory stack area (LIFO). When a call instruction or PUSH instruction is executed or an interrupt occurs, the SP contents are decremented; when a return instruction or POP instruction is executed, the SP contents are incremented.

3.2 Mode Registers

The mode registers (see Table 3-1) are provided to control the port, timer, timer/event counter, serial interface, A/D converter, and interrupt control blocks.

Table 3-1 Mode Register Function List

Mode register name		Read/Write	Function
MA	MODE A register	W	The input or output mode is selected bit-wise for port A.
MB	MODE B register	W	The input or output mode is selected bit-wise for port B.
MCC	MODE CONTROL C register	W	The port or control mode is selected bit-wise for port C.
MC	MODE C register	W	The input or output mode is selected bit-wise for port C placed in the port mode.
MM	MEMORY MAPPING register	W	The port or expansion mode is selected for ports D and F.
MF	MODE F register	W	The input or output mode is selected bit-wise for port F placed in the port mode.
TMM	Timer mode register	R/W	The timer operation mode is specified.
ETMM	Timer/event counter mode register	W	The timer/event counter operation mode is specified.
EOM	Timer/event counter output mode register	W/R	CO0 and CO1 output levels are controlled.
SML SMH	Serial mode register	W W/R	The serial interface operation mode is specified.
ANM	A/D channel mode register	W/R	The A/D converter operation mode is specified.
ZCM	Zero cross mode register	W	The zero cross detector operation is specified.

3.3 Arithmetic and Logic Unit (ALU)

The arithmetic and logic unit (ALU) performs 8-bit arithmetic and logical operations and data processing such as shift and rotation, 16-bit arithmetic and logical operations and data processing such as shift, 8-bit multiplication, and 16 bits ÷ 8 bits division.

3.4 Program Status Word (PSW)

The program status word consists of six flags which are set or reset according to the instruction execution result. The three flags Z, HC, and CY can be tested by using instructions. When an interrupt (external, internal, or SOFTI instruction) occurs, the PSW contents are automatically saved in the stack; when the RETI instruction is executed, the PSW contents are restored. When RESET is input, all the PSW bits are reset to 0.

Fig. 3-2 PSW Format

7	6	5	4	3	2	1	0
0	Z	SK	HC	L1	L0	0	CY

(1) Z (Zero)

When the operation result is zero, the Z flag is set to 1; when not zero, it is reset to 0.

(2) SK (Skip)

When the skip condition is true, the SK flag is set to 1; when not true, it is reset to 0.

(3) HC (Half Carry)

When the operation results in a carry from bit 3 or a borrow into bit 3, the HC flag is set to 1; otherwise, it is reset to 0.

(4) L1

When MVI A, byte instruction string effect is made, the L1 flag is set to 1; when not made, it is reset to 0.

(5) L0

When MVI L, byte; LXI H, word instruction string effect is made, the L0 flag is set to 1; when not made, it is reset to 0.

(6) CY (Carry)

When the operation results in a carry from bit 7 or 15 or a borrow into bit 7 or 15, the CY flag is set to 1; otherwise, it is reset to 0.

When any of the 35 ALU instructions, rotation instructions, and carry manipulation instructions is executed, the flags are affected as listed in Table 3-2.

Table 3-2 Flag Operation

Operation						D6	D5	D4	D3	D2	D0
reg, memory		immediate		skip	Z	SK	HC	L1	L0	CY	
ADD	ADDW	ADDX	ADI								
ADC	ADCW	ADCX	ACI								
SUB	SUBW	SUBX	SUI								
SBB	SBBW	SBBX	SBI								
DADD					†	0	†	0	0	†	
DADC											

(to be continued)

Table 3-2 Flag Operation (Cont'd)

Operation						D6	D5	D4	D3	D2	D0
reg, memory			immediate		skip	Z	SK	HC	L1	LO	CY
DSUB						†	0	†	0	0	†
DSBB											
EADD											
ESUB											
ANA	ANAW	ANAX	ANI	ANIW							
ORA	ORAW	ORAX	ORI	ORIW							
XRA	XRAW	XRAX	XRI			†	0	•	0	0	•
DAN											
DOR											
DXR											
ADDNC	ADDNCW	ADDNCX	ADINC								
SUBNB	SUBNBW	SUBNBX	SUINB								
GTA	GTAW	GTAX	GTI								
LTA	LTAW	LTX	LTI								
DADDNC											
DSUBNB											
DGT											
DLT											
ONA	ONAW	ONAX	ONI	ONIW							
OFFA	OFFAW	OFFAX	OFFI	OFFIW		†	†	•	0	0	•
DON											
DOFF											
NEA	NEAW	NEAX	NEI	NEIW							
EQA	EQAW	EQAX	EQI	EQIW							
DNE											
DEQ											
INR	INRW					†	†	†	0	0	•
DCR	DCRW										
DAA							0		0	0	†
RLR	RLL	SLR	SLL				•	0	•	0	0
DRLR	DRLL	DSLR	DSLL								
SLRC	SLLC						•		0	0	†
STC							•	0	•	0	1
CLC							•	0	•	0	0
	MVI A, byte					•	0	•	1	0	•
	MVI L, byte					•	0	•	0	1	•
	LXI H, word										
						BIT					
						SK					
						SKN	•	†	•	0	0
						SKIT					
						SKNIT					
						RETS	•	1	1	0	0
All other instructions							•	0	•	0	0

† : Affected (set or reset) 1: Set 0: Reset •: Not affected

3.5 Memory

3.5.1 uPD78C14/78C14A/78C12A/78C11/78C11A/78C10/78C10A memory configuration

The uPD78C14/78C14A/78C12A/78C11/78C11A/78C10/78C10A enables addressing a maximum of 64K bytes of memory. Figs. 3-3 to 3-6 show memory maps. The external memory area and internal RAM area can be used as program memory and data memory as desired. The internal and external memory access timings are the same, thus enabling high speed processing.

(1) Interrupt start addresses

All interrupt start addresses are fixed as follows:

NMI	0004H
INTT0/INTT1	0008H
INT1/ <u>INT2</u>	0010H
INTE0/INTE1	0018H
INTEIN/INTAD ...	0020H
INTSR/INTST	0028H
SOFTI	0060H

(2) Call address table

A maximum of 32 1-byte call instruction (CALT) call addresses can be stored in the 64-byte area of addresses 0080H-00BFH.

(3) Memory specific area

Note that the reset start addresses, interrupt start addresses, and call table are assigned to addresses 0000H-00BFH. The memory area of addresses 0800H-0FFFH can be directly addressed by using a 2-byte call instruction (CALF). Internal mask ROM is assigned as follows:

- uPD78C14/78C14A: Addresses 0000H-3FFFFH
- uPD78C12A: Addresses 0000H-1FFFFH
- uPD78C11/78C11A: Addresses 0000H-0FFFFH
- uPD78C10/78C10A: Internal mask ROM is not contained.

The uPD78C10/78C10A enables a specific area to be installed at the external.

(4) Internal data memory area

256-byte RAM is contained in addresses FF00H-FFFFH. The 256-byte RAM contents of addresses FF00H-FFFFH are retained during the standby operation.

To use internal RAM, be sure to set the MM register RAE bit to 1.

(5) External memory area

The following are can be expanded as external memory area:
(The area can be expanded by stages by setting the MEMORY MAPPING register.)

- uPD78C14/78C14A: 48K bytes (addresses 4000H-FEFFFH)
- uPD78C12A: 56K bytes (addresses 2000H-FEFFFH)
- uPD78C11/78C11A: 60K bytes (addresses 1000H-FEFFFH)

On the uPD78C10/78C10A, external memory can be expanded in the 64K-byte area of addresses 0000H-FEFFFH by stages by setting the MODE0 and MODE1 pins. (See Table 2-4.)

External memory is accessed by using PD7-PD0 (multiplexed address/data bus), PF7-PF0 (address bus), and RD, RW, and ALE signals. Programs and data can be stored in the external memory.

(6) Working register area

A 256-byte working register area can be installed in any desired portion of memory as specified in the V register.

Fig. 3-3 Memory Map (uPD78C14/78C14A)

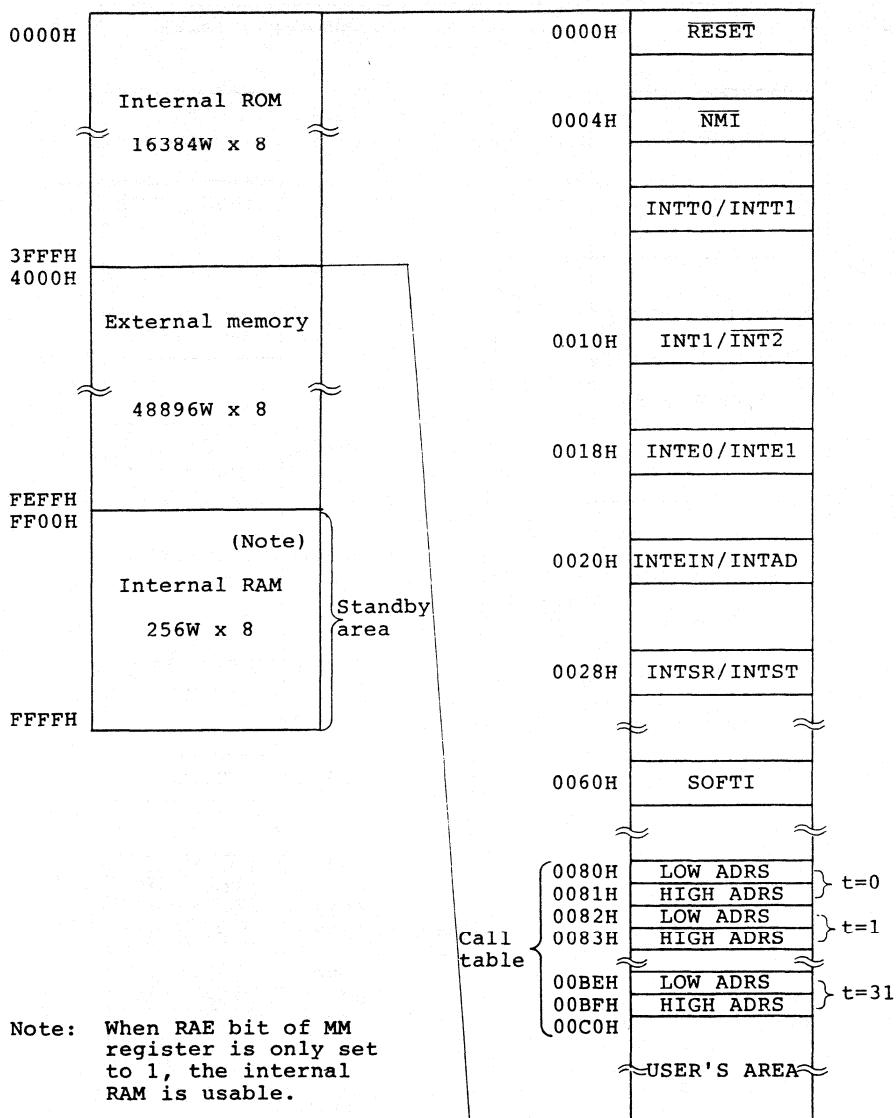


Fig. 3-4 Memory Map (uPD78C12A)

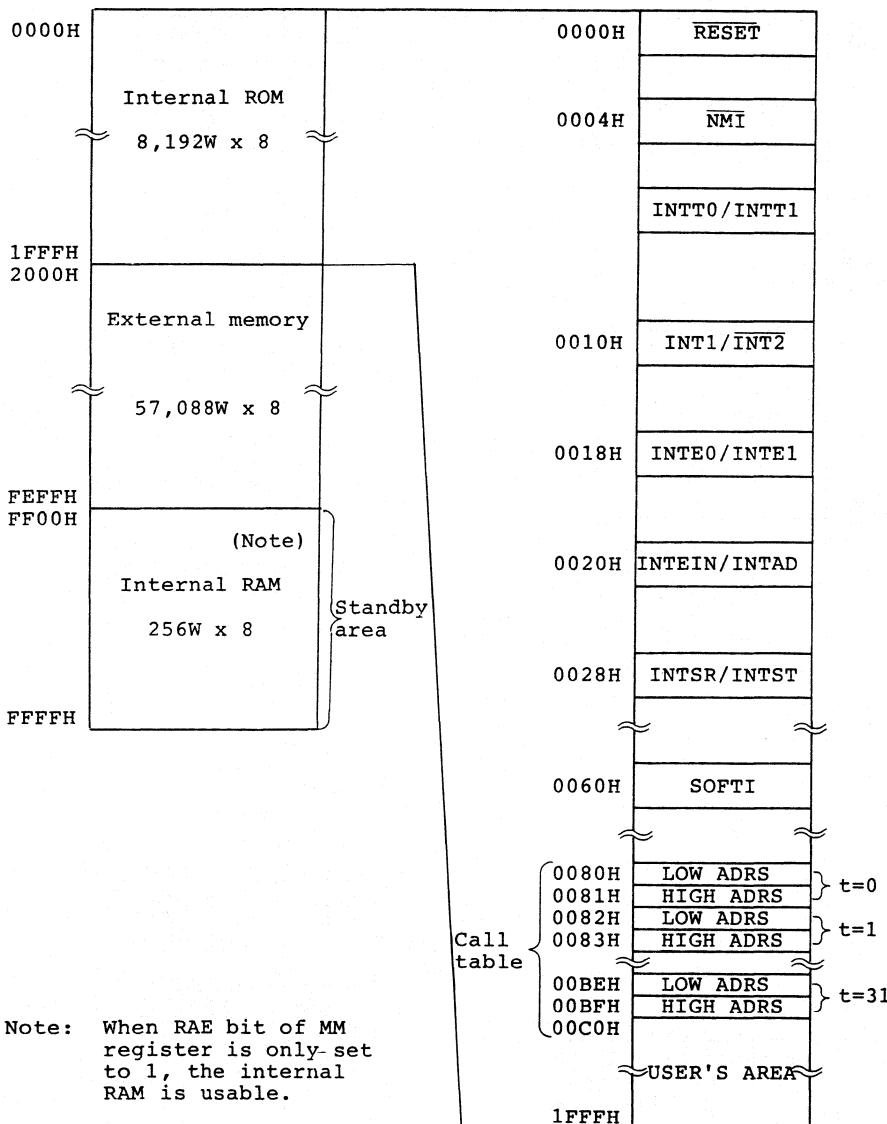
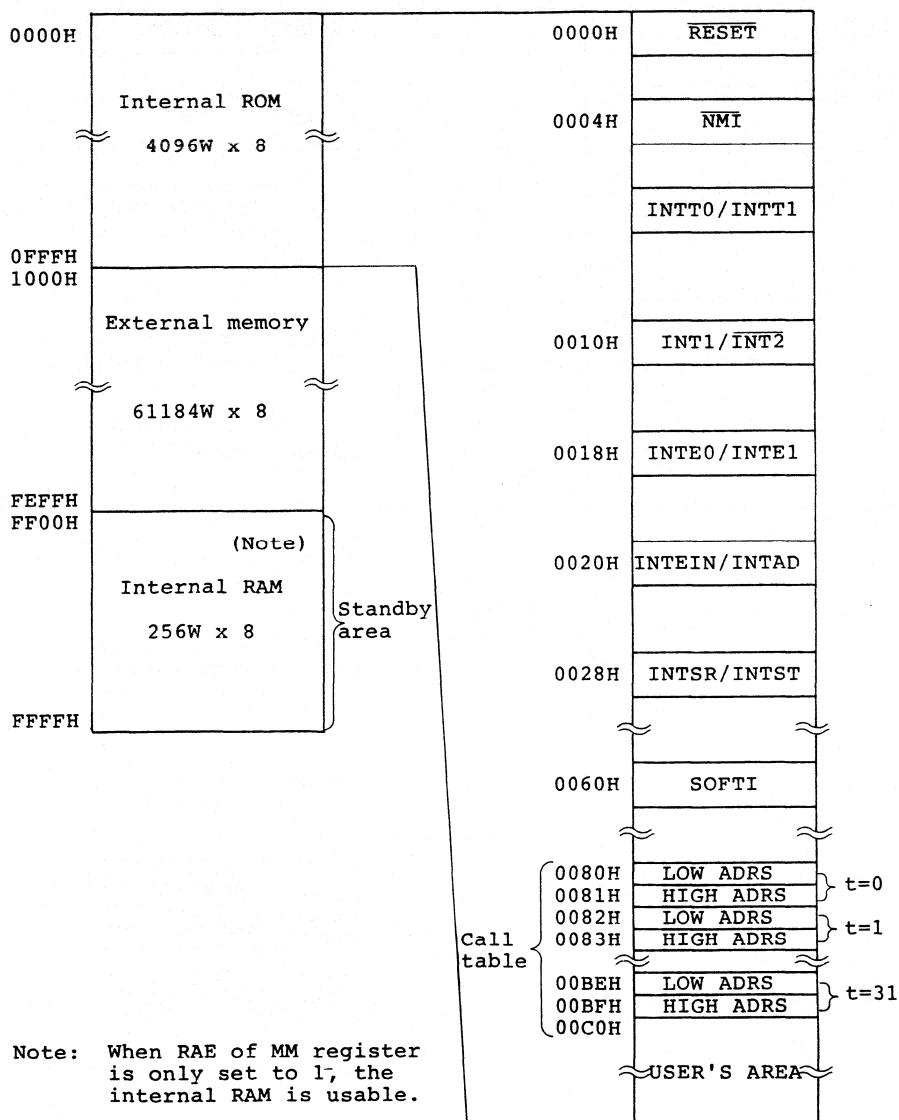
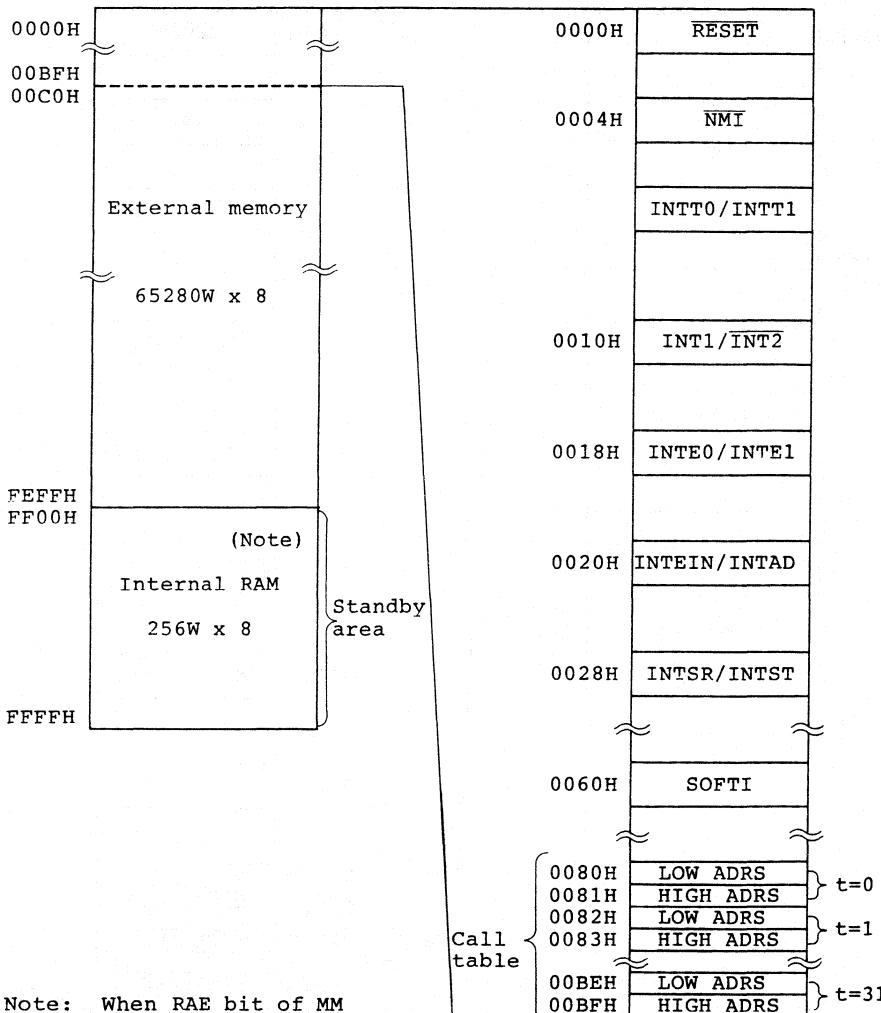


Fig. 3-5 Memory Map (uPD78C11/78C11A)



Note: When RAE of MM register is only set to 1, the internal RAM is usable.

Fig. 3-6 Memory Map (μ PD78C10/78C10A)



Note: When RAE bit of MM register is only set to 1, the internal RAM is usable.

3.5.2 uPD78CP14 memory configuration

The uPD78CP14 provides the memory function and configuration equivalent to the uPD78C11/78C11A/78C12A/78C14/78C14A. In addition, the EPROM access range can be selected by using the memory mapping register so that external memory can be set efficiently.

The vector addresses, call table area, and data memory area are common to all types of products.

Figs. 3-7 to 3-9 show memory map.

Fig. 3-7 Memory Map (uPD78C14/78C14A mode)

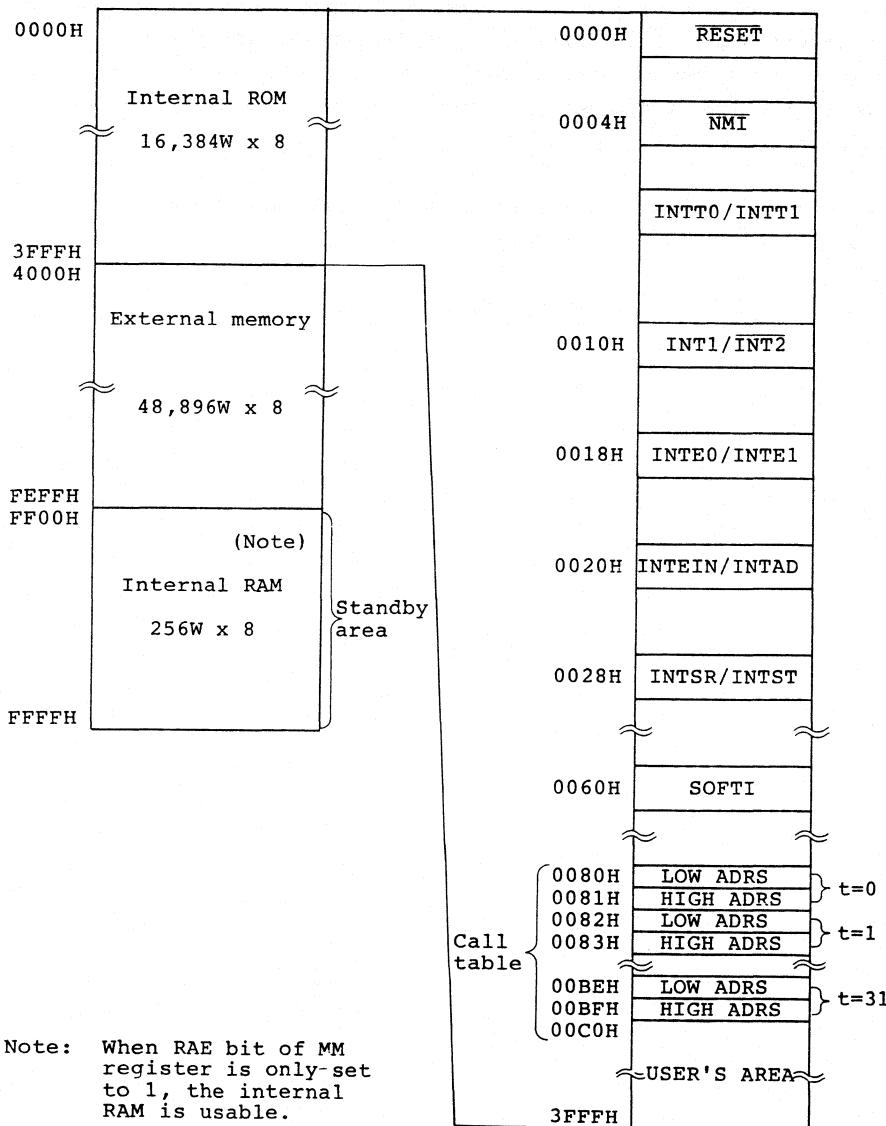


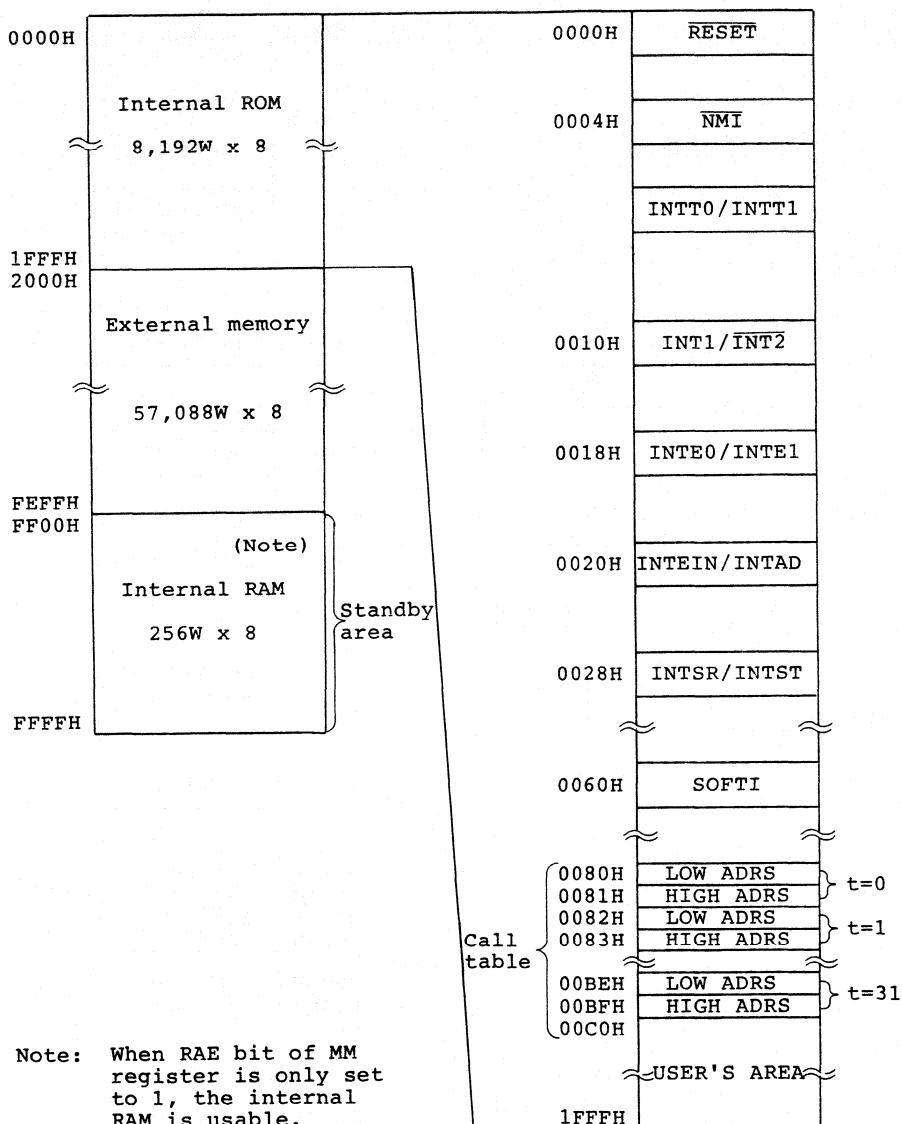
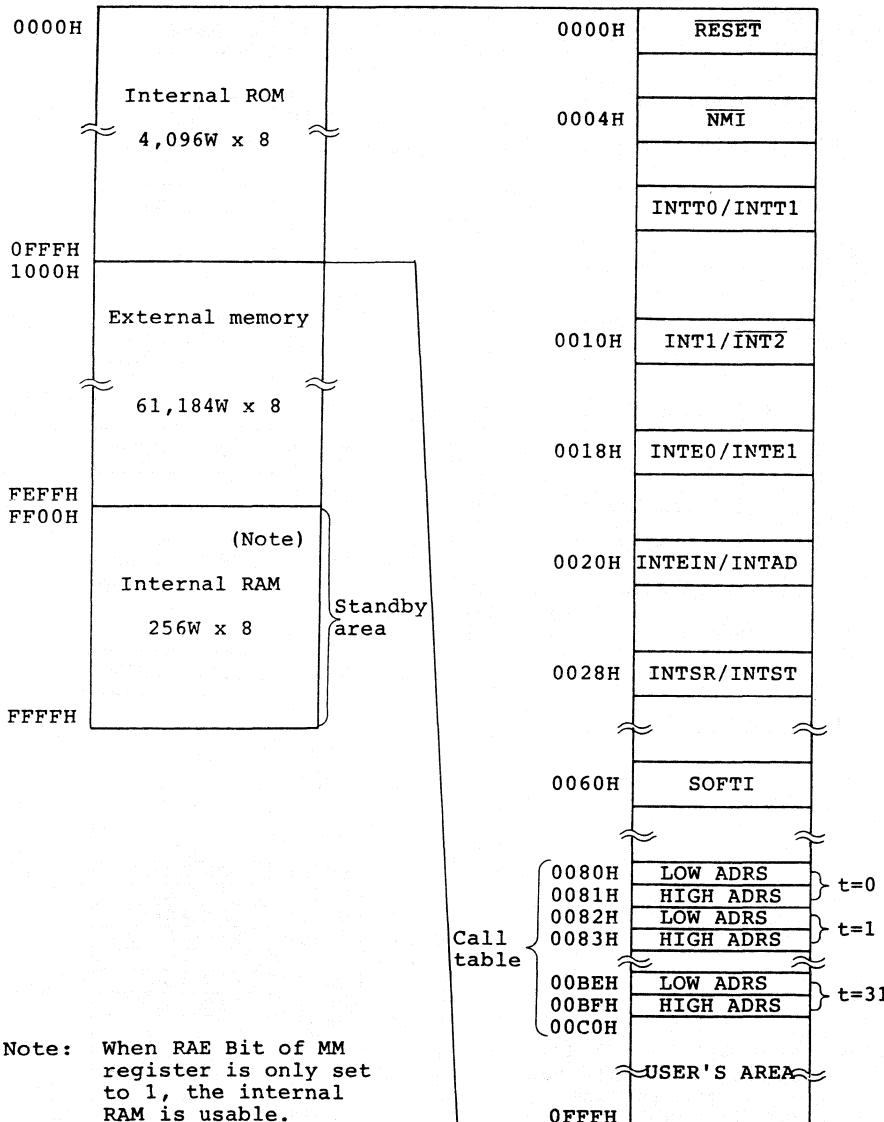
Fig. 3-8 Memory Map (μ PD78C12A mode)

Fig. 3-9 Memory Map (uPD78C11/78C11A mode)



Note: When RAE Bit of MM register is only set to 1, the internal RAM is usable.

3.6 Timer

The timer block consists of two 8-bit interval timers. The two 8-bit interval timers which are cascaded also serve as a 16-bit interval timer.

The interval time elapsed can be known when a timer interrupt occurs. Square wave with the interval time as a half period can be obtained from the TO pin. (For details, see Chapter 5.)

3.7 Timer/Event Counter

The 16-bit timer/event counter functions as follows according to the operation mode set in a program: (For details, see Chapter 6.)

- o Interval timer
- o Event counter
- o Frequency measurement
- o Pulse width measurement
- o Programmable square wave output

3.8 Serial Interface

The serial interface is used to transfer serial data to/from various terminals. It can operate in the asynchronous mode, synchronous mode, or I/O interface mode. (For details, see Chapter 7.)

3.9 Analog/Digital Converter

The analog/digital converter block consists of an 8-bit A/D converter having eight analog inputs by high-precision successive approximation and four conversion result registers which retain the conversion results, CR0-CR3.

Analog input is selected in the scan mode or select mode. The four conversion result retention registers CR0 to CR3 minimize software overhead. (For details, see Chapter 8.)

3.10 Interrupt Control

Three external interrupt requests and eight internal interrupt requests are used. They are controlled according to how the interrupt mask register is set and the priority levels.

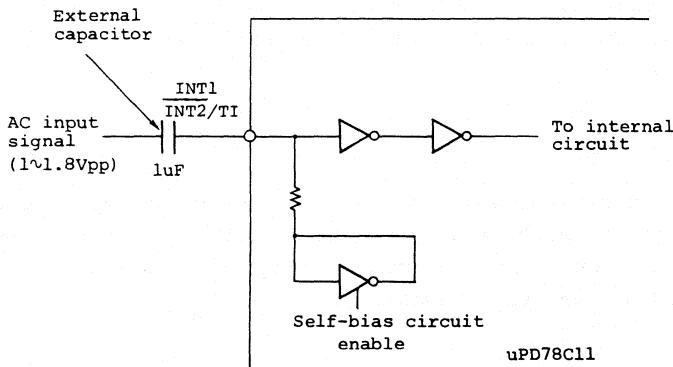
The 11 types of interrupt requests are classified into six groups each having the unique priority level and interrupt address. (For details, see Chapter 9.)

3.11 Zero Cross Detector

The INT1 pin and INT2/TI (PC3) pin can be used for zero cross detection operation by setting the zero cross mode register.

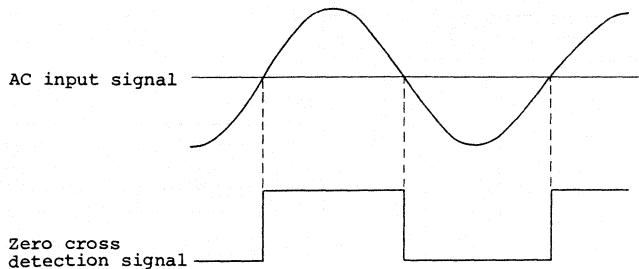
The zero cross detector consists of a high-gain amplifier of the self-bias system. Its input is biased to the switching point and the digital transition is made in response to slight input transition.

Fig. 3-10 Zero Cross Detector



The zero cross detector detects the transition of the AC signal input through an external capacitor from negative to positive or positive to negative, and generates a digital pulse which changes from 0 to 1 or 1 to 0 at the AC signal transition point.

Fig. 3-11 Zero Cross Detection Signal



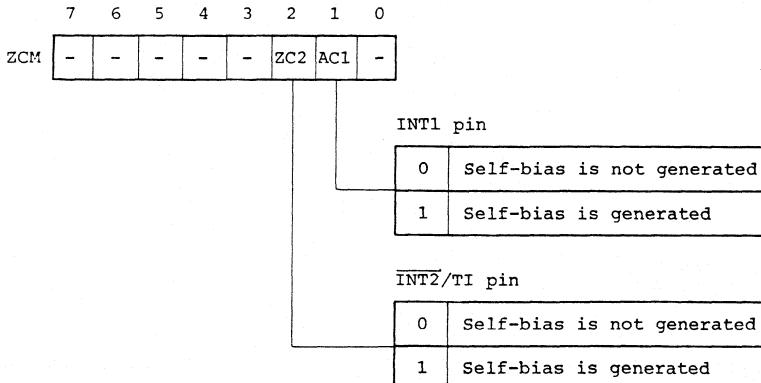
The digital pulse generated by the INT1 pin zero cross detector is sent to the interrupt control circuit. The INTF1 interrupt request flag is set to 1 at the AC signal zero cross point from negative to positive (rising edge). If INT1 interrupt is enabled, the interrupt service is started. The digital pulse generated by the $\bar{INT2}/IT$ pin zero cross detector can cause the interrupt service to be started, as with the INT1 pin, and can also be used as timer input clock.

The zero cross function enables use of 50- or 60-Hz power signal as the system timing basis. The zero cross detection function can also be used for the interrupt service at a zero voltage point. This feature enables control of voltage phase sense system devices such as a triode AC switch and SCR; it also enables use of the uPD78C11 for applications such as shaft speed and angle measurement.

The INT1 or $\bar{INT2}$ pin to which a capacitor is not connected serves as a digital input pin.

Fig. 3-12 shows the format of the zero cross mode register (ZCM) to control self bias for INT1 pin and $\overline{\text{INT2}}/\text{TI}$ pin zero cross detection.

Fig. 3-12 Zero Cross Mode Register Format



When the zero cross mode register ZC1 and ZC0 bits are set to 0, self-bias for zero cross detection by the INT1 and $\overline{\text{INT2}}$ pins is not generated, and the pins respond as normal digital input.

When the ZC1 and ZC2 bits are set to 1, self-bias is generated and AC input signal zero cross can be detected by connecting a capacitor to each of the INT1 and $\overline{\text{INT2}}$ pins. When the ZC1 and ZC2 bits are set to 1, the pins can also be directly driven not via external capacitor, in which case the pins respond as digital input. However, in this case, input load current is required and an external circuit output driver must be considered. Thus, not to make zero cross detection and use the pins as simple interrupt input and timer input, set the zero cross mode register ZC1 and ZC2 bits to 0.

When RESET is input, both the ZC1 and ZC2 are set to 1 and self-bias is generated.

In the port mode, the PC3 (INT2/TI) pin does not generate self-bias regardless or how the ZCM register is set.

Caution 1: Unlike other CMOS circuits, supply current always flows in the zero cross detector because of its operation point. The current also flows in the detector in the standby mode (HALT or software or hardware STOP mode). Thus, when the zero cross detector is operated (self bias is generated ($ZCx = 1$)), slightly much current flows as compared with the time the cross zero detector is not operated. This affection becomes large in the software STOP mode.

Caution 2: To use the PC3 pin for zero cross detection on the uPD78C11A, uPD78C12A or uPD78C14A, do not contain a pull-up resistor.

During the hardware STOP mode, automatically self-bias generation is stopped.

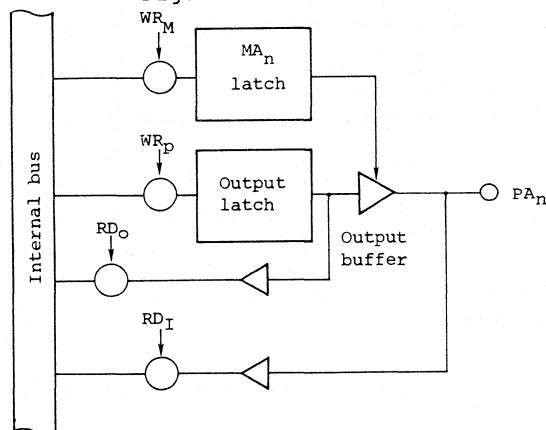
CHAPTER 4 PORT FUNCTION

4.1 Port A (PA7-PA0)

Port A is an 8-bit input/output port which has the input/output buffer and output latch function. (See Fig. 4-1.) Port A can be set to an input or output port bit-wise by using the MODE A register (MA). When it is set to an input port, output becomes high impedance.

On the uPD78C11A, uPD78C12A and uPD78C14A, a pull-up resistor can be contained bit-wise.

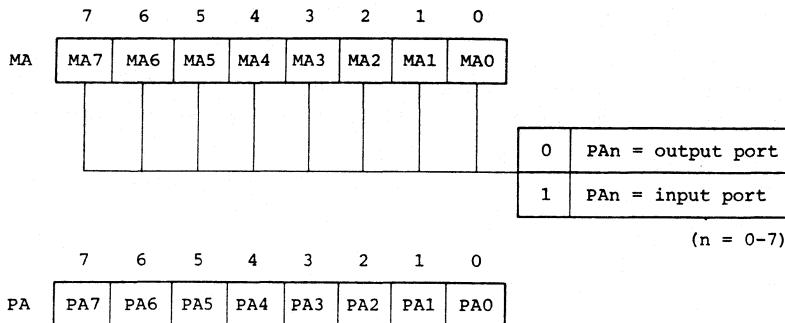
Fig. 4-1 Port A



When a MODE A register bit is set to 1, its corresponding port A pin becomes an input port; when 0, its corresponding port A pin becomes an output port. (See Fig. 4-2.)

When RESET is input or during the hardware STOP mode, all the MODE A register bits are set to 1 and port A becomes input port (output high impedance).

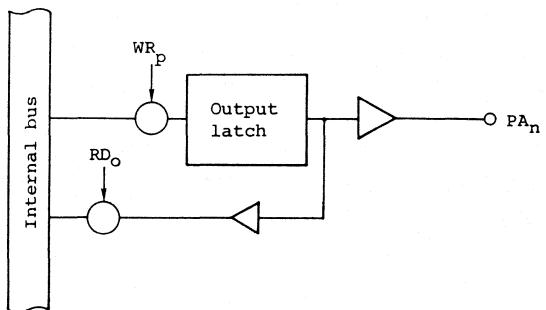
Fig. 4-2 MODE A Register Format



- (1) When output port is specified (MA_n = 0)

The output latch becomes effective. Data can be transferred between the output latch and accumulator by using transfer instructions. The output latch contents can be directly set or reset bit-wise not via the accumulator by executing instructions such as arithmetic and logical instructions. The data once written into the output latch is retained until a new port A manipulation instruction is executed or reset is made.

Fig. 4-3 Port A Specified as Output Port

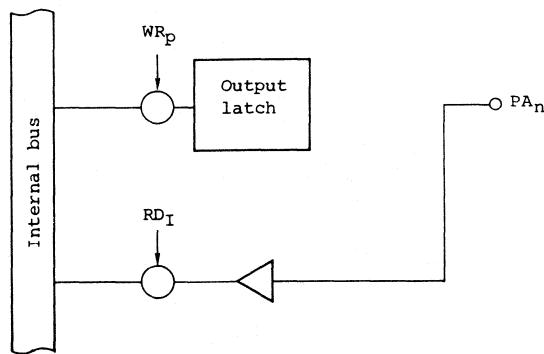


(2) When input port is specified ($MAn = 1$)

The PA line contents can be loaded into the accumulator by using transfer instructions. The PA line contents can also be directly tested bit-wise not via the accumulator by executing instructions such as arithmetic and logical instructions. In this case, a write into the output latch can also be made, and the data transferred from the accumulator by a transfer instruction is stored in the output latch regardless of the input or output mode is selected for the port. However, the output latch contents of the bit specified as an input port cannot be loaded into the accumulator, and are not output to external pin (which serves as an input pin) because the output buffer is high impedance. When the bit is changed to an output port, the data stored in the output latch is output to external pin, and can be loaded into the accumulator.

Since input data is not latched, stable input is required when a data transfer instruction or bit test is executed.

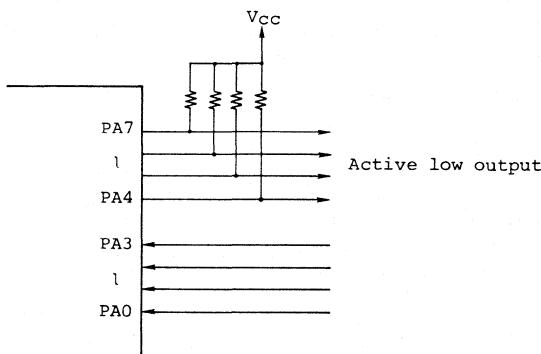
Fig. 4-4 Port A Specified as Input Port

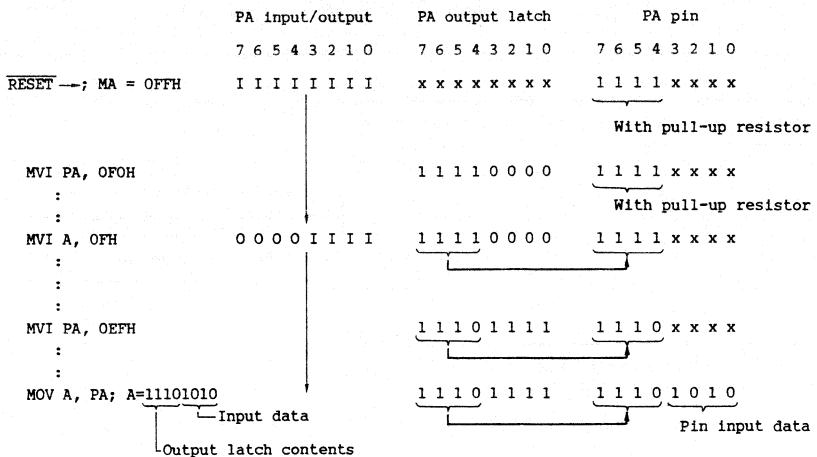


(3) Port A manipulation

An actual port A manipulation instruction is executed in 8-bit units. If a port A read instruction (MOV A,PA) is executed, the input line contents of the ports specified as input and the output latch contents of the ports specified as output are loaded into the accumulator. When a port A write instruction (MOV PA,A) is executed, both output latches of ports specified as input and output are written, but the output latch contents of the ports specified as input are not output to external pins.

Data input/output operation when the high-order four bits of port A, PA7-PA4 are used as active low output port and the low-order four bits PA3-PA0 are used as input port is explained. Since all of PA7-PA0 become input ports (output high impedance) in the initial state after reset, the PA7-PA4 output ports must previously be pulled high with pull-up resistors to deactivate the active low PA7-PA4 output ports. Since the output latch contents become undefined when reset, active low may be output when PA7-PA4 are specified as output port. Write all 1s into the PA7-PA4 output latches before specifying as output port.





4.2 Port B (PB7-PB0)

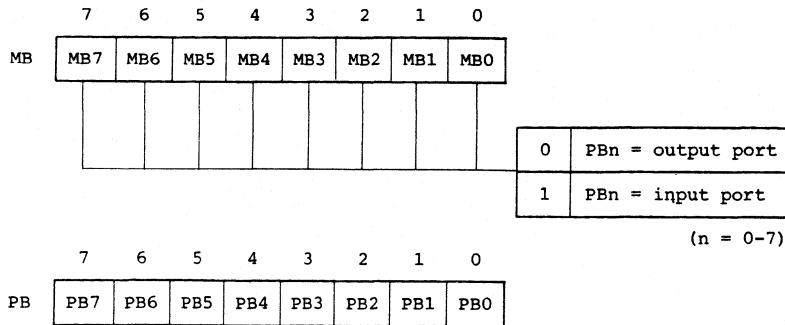
Like port A, port B is an 8-bit input/output port which has the input/output buffer and output latch function. (See Fig. 4-1.) Port B can be set to an input or output port bit-wise by using the MODE B register (MB). When it is set to an input port, output becomes high impedance.

If a MODE B register bit is set to 1, its corresponding port B pin becomes an input port; when 0, its corresponding port B pin becomes an output port. (See Fig. 4-5.)

When RESET is input or during the hardware STOP mode, all the MODE B register bits are set to 1 and port B becomes input port (output high impedance).

On the uPD78C12A and uPD78C14A, pull-up resistor can be contained bit-wise.

Fig. 4-5 MODE B Register Format



Port B operation is the same as port A operation. The port contents can be directly set, reset, and tested bit-wise not via the accumulator by executing instructions such as arithmetic and logical instructions. Data can also be transferred between port B and the accumulator.

4.3 Port C (PC7-PC0)

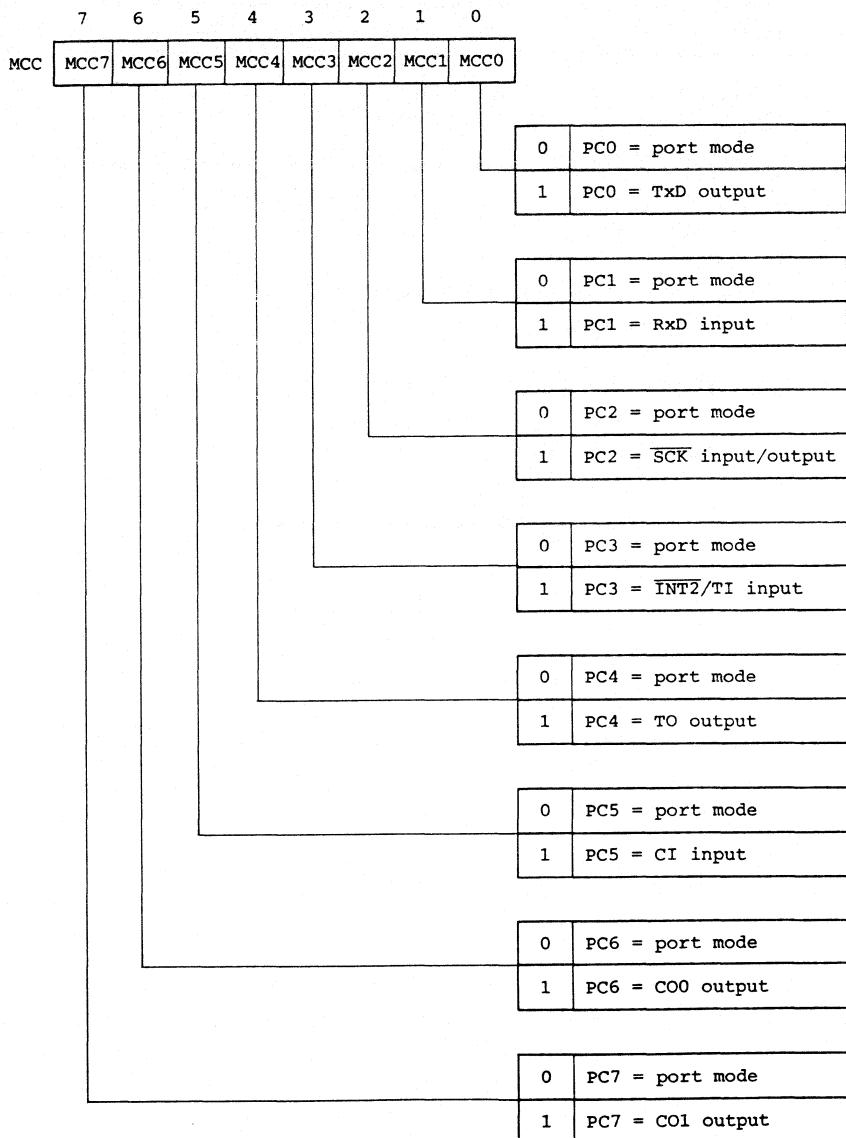
Port C (PC7-PC0) is an i-bit special input/output port which serves as the port mode or control signal input/output mode by setting the MODE CONTROL C (MCC) register.

If a MODE CONTROL C register bit is set to 1, its corresponding port C pin is placed in the control mode; when 0, its corresponding port C pin is placed in the port mode. (See Fig. 4-6.)

When RESET is input or during the hardware STOP mode, all the MODE CONTROL C register bits are reset and all the port C bits are placed in the port mode.

On the uPD78C11A, uPD78C12A and uPD78C14A, a pull-up resistor can be contained bit-wise.

Fig. 4-6 MODE CONTROL C Register Format



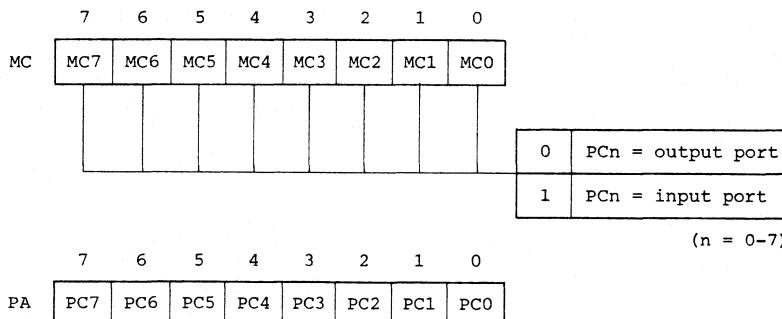
(1) Port mode

Like port A, port C is an 8-bit input/output port which has the input/output buffer and output latch function. (See Fig. 4-1.) Port C specified as the port mode in the MODE CONTROL C register can be set to an input or output port bit-wise by using the MODE C register (MC). When it is set to an input port, output becomes high impedance.

If a MODE C register bit is set to 1, its corresponding port C pin becomes an input port; when 0, its corresponding port C pin becomes an output port. (See Fig. 4-7.)

When RESET is input or during the hardware STOP mode, all the MODE C register bits are set to 1 and port C becomes input port (output high impedance).

Fig. 4-7 MODE C Register Format



Port C operation is the same as port A operation. The port contents can be directly set, reset, and tested bit-wise not via the accumulator by executing instructions such as arithmetic and logical instructions. Data can also be transferred between port C and the accumulator.

(2) Control signal input/output mode

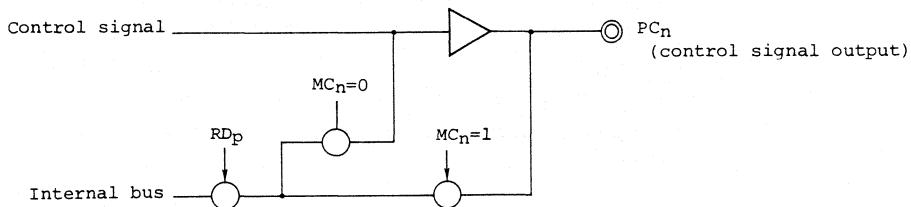
By setting a MODE CONTROL C register bit to 1, its corresponding port C input/output pin of PC7-PC0 can be used as a control signal input or output pin regardless of how the MODE C register is set. When a PCn pin is used as a control signal input or output pin ($MCCn = 1$), the control signal state can be known by executing a port C read instruction or test instruction.

(a) When PCn is used as a control signal output pin

When $MCn = 1$, the PCn pin control signal state can be read into the accumulator or tested by executing a port C read instruction or test instruction.

When $MCn = 0$, the internal control signal state can be read into the accumulator or tested by executing a port C read instruction or test instruction. (See Fig. 4-8.)

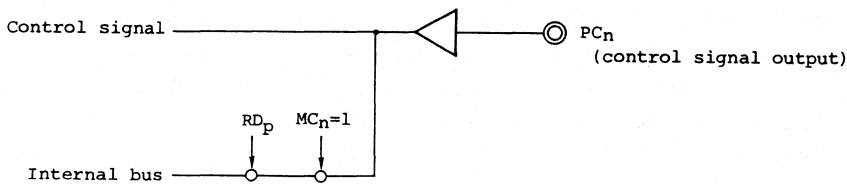
Fig. 4-8 Port C Specified as Control Signal Output



(b) When PCn is used as a control signal input pin

Only when $MCn = 1$, the PCn pin control signal state can be read into the accumulator or tested by executing a port C read instruction or test instruction. (See Fig. 4-9.)

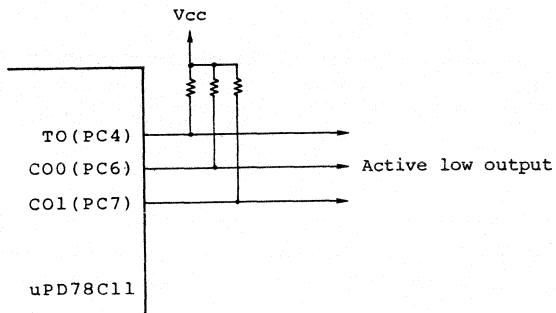
Fig. 4-9 Port C specified as control signal input



Caution 1: If MCC3 is rewritten, INTF2 may be set. After MCC3 is rewritten, reset INTF2 by executing a SKIT instruction.

Caution 2: To use TO (PC4), CO0 (PC6), and CO1 (PC7) as active low signal output pins, perform the following:

Since all port C pins become input port pins (output high impedance) in the initial state after reset, the TO, CO0, and CO1 pins must previously be pulled high with pull-up resistors to deactivate the active low TO, CO0, and CO1 pins. Before changing to the control signal output mode in the MODE CONTROL C register, write 1 into the port C output latch and make the port C output level equal to the output latch contents. Next, change the port C mode to the control signal output mode by using the MODE CONTROL C register.



```

MVI PC, OFFH ; PORT C OUTPUT LATCH = 1
MVI A, OFFH ;
MOV MCC, A ; PORT C CONTROL MODE

```

4.4 Port D (PD7-PD0)

uPD78C14/78C14A/78C12A/78C11/78C11A

Port D is an 8-bit special input.output port which serves as the multiplexed address/data bus in addition to general purpose input/output port (port mode).

The port mode or expansion mode can be selected for port D in byte units by using the MEMORY MAPPING register. (See Table 4-1.)

Table 4-1 PD7-PD0 Operation (uPD78C14/78C14A/78C12A/78C11/78C11A)

	MM2, MM1 = 0, 0	MM2, MM1 ≠ 0, 0
PD7-0	Port mode	Expansion mode

If the MEMORY MAPPING register MM2 and MM1 bits are reset to 0, port D is placed in the port mode; if not, port D is placed in the expansion mode. (See 11.1.1.)

(1) Port mode

Port D in the port mode is an 8-bit input/output port which has the input/output buffer and output latch function like port A; however, the input or output mode is selected for port D in byte (8-bit) units.

The input or output mode can be selected for port D in byte units by using the MEMORY MAPPING register MM0 bit. If the MM0 is reset to 0, port D becomes an input port; if 1, port D becomes an output port.

Port D operation is the same as port A operation except that the input or output mode is selected in byte units. The port contents can be set, reset, and tested bit-wise not via the accumulator by executing arithmetic and logical operation instructions. Data can also be transferred between port D and the accumulator.

(2) Expansion mode

In the expansion mode, port D input/output pins PD7-PD0 are used as the multiplexed address/data bus and memory of a maximum of 256 bytes can be expanded to the external. To expand furthermore large memory to the external, PF7-PF0 are used as address bus. (For details, see Chapter 11.)

■ uPD78C10/78C10A

Port D does not contain the port function and serves only as the multiplexed address/data bus (AD7-AD0).

Caution 1: In every machine cycle, the internal address bus state is output in synchronization with ALE to the port D input/output pins PD7-PD0 when port D is used as the multiplexed address/data bus (AD7-AD0).

Caution 2: A program which changes the port D operation mode dynamically cannot be emulated by an emulator. Do not change the once set mode to another mode.

4.5 Port F (PF7-PF0)

■ uPD78C14/78C14A/78C12A/78C11/78C11A

Port F is an 8-bit special input/output port which serves as an address bus in addition to general purpose input/output port (port mode).

The port or expansion mode can be selected for PF7-PF0 by using the MEMORY MAPPING register. (See 11.1.1.)

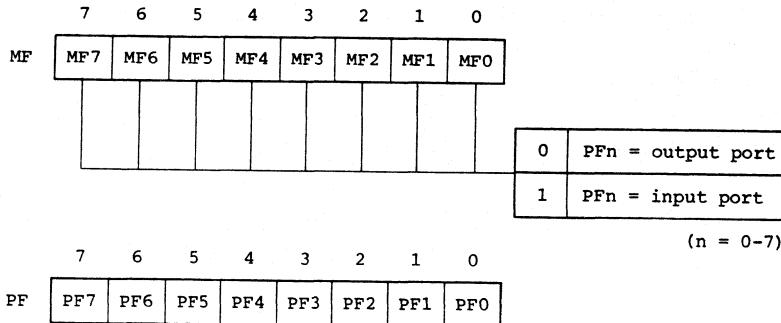
(1) Port mode

Port F in the port mode is an 8-bit input/output port which has the input/output buffer and output latch function like port A. (See Fig. 4-1.) The input or output mode can be selected bit-wise for port F by using the MODE F register (MF). When port F is set to an input port, output becomes high impedance.

If a MODE F register bit is set to 1, its corresponding port F pin becomes an input port pin; if 0, its corresponding port F pin becomes an output port pin.

When RESET is input or during the hardware STOP mode, all the MODE F register bits are set to 1.

Fig. 4-10 MODE F Register Format



Port F operation is the same as port A operation. The port contents can be directly set, reset, and tested not via the accumulator by executing arithmetic and logical operation instructions. Data can also be transferred between port F and the accumulator.

(2) Expansion mode

Port F input/output pins PF7-PF0 can be used as address output pins according to the size of externally expanded memory, as listed in Table 4-2, as set in the MEMORY MAPPING register.

The pins not used for address output are placed in the port mode.

Table 4-2 PF7-PF0 Operation (uPD78C14/78C14A/78C12A/78C11/78C11A)

PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0	External address space
Port	Within 256 bytes							
Port	Port	Port	Port	AB11	AB10	AB9	AB8	Within 4K bytes
Port	Port	AB13	AB12	AB11	AB10	AB9	AB8	Within 16K bytes
AB15	AB14	AB13	AB12	AB11	AB10	AB9	AB8	Within 48K, 56K or 60K bytes (Note)

Note: 48K = uPD78C14/78C14A, 56K = uPD78C12A, 60K = uPD78C11/78C11A

■ uPD78C10/78C10A

Port F becomes address output pins according to the size of externally installed memory.

The pins not used for address output can be used as general purpose input/output port pins having the port function like port A. The input or output mode is selected by setting the MODE F register.

Table 4-3 PF7-PF0 Operation (uPD78C10/78C10A)

MODE1	MODE0	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0	External address space
0	0	Port	Port	Port	Port	AB11	AB10	AB9	AB8	4K bytes
0	1	Port	Port	AB13	AB12	AB11	AB10	AB9	AB8	16K bytes
1	1	AB15	AB14	AB13	AB12	AB11	AB10	AB9	AB8	64K bytes

Caution 1: In every machine cycle, the internal address bus state is output from the pins used as the address bus.

Caution 2: A program which changes the port F operation mode dynamically cannot be emulated by an emulator. Do not change the once set mode to another mode.

To use the 64K-byte mode on the uPD78C10/78C10A, do not execute any port D or F output instruction. If the instruction is executed, the WR signal is output.

CHAPTER 5 TIMER FUNCTION

5.1 Timer Configuration

The uPD78C11 timer block consists of two 8-bit interval timers TIMER0 and TIMER1 and TIMER F/F. Timer operation and square wave output are controlled by using the timer mode register (TMM).

The interval timers TIMER0 and TIMER1 each consists of an 8-bit UP COUNTER, 8-bit COMPARATOR, and 8-bit TIMER REG0 (TM0), (TIMER REG1 (TM1)).

(1) UP COUNTER

The UP COUNTER is incremented according to the input clock specified in the timer mode register (TMM).

(2) TIMER REG0 (TM0) (TIMER REG1 (TM1))

Is an 8-bit register to set the interval time.

(3) COMPARATOR

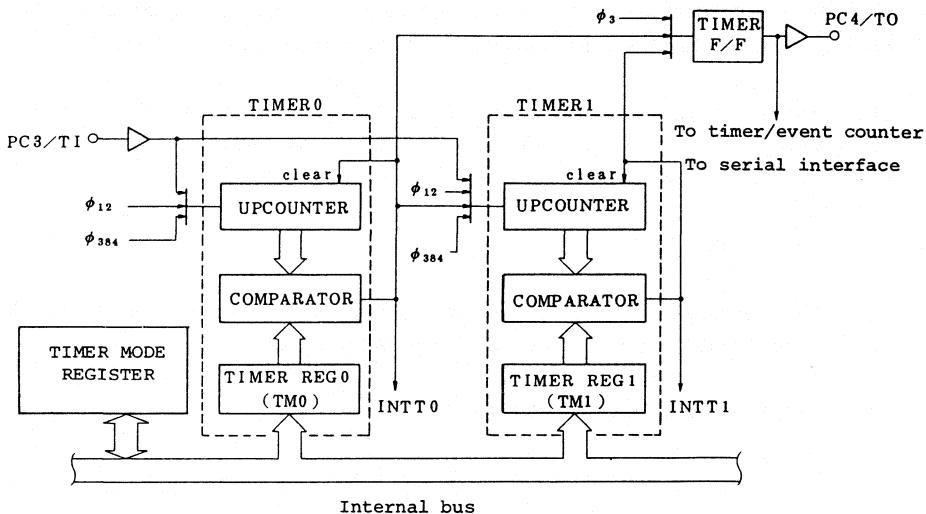
The COMPARATOR compares the UP COUNTER contents with the TIMER REG0 (TIMER REG1) contents. If they match, the COMPARATOR clears the UP COUNTER and generates an internal interrupt INTT0 or INTT1.

(4) TIMER F/F

The TIMER F/F is inverted by the TIMER0 (TIMER1) coincidence signal or internal clock (ϕ_3). TIMER F/F output can be sent to the TO pin (PC4). Regardless of the PC4 pin mode, TIMER F/F output can be used as the timer/event counter reference time as specified in the timer/event counter mode register or can be used as serial clock (\overline{SCK}) as specified in the

serial mode register. Timer is also used to generate the oscillator stable time when standby operation (STOP) is released. (For details, see 10.1).

Fig. 5-1 Timer Block Diagram



Remarks: $\phi_3 = f_{XX} \times 1/3$ f_{XX} : Oscillator frequency (MHz)
 $\phi_{12} = f_{XX} \times 1/12$
 $\phi_{384} = f_{XX} \times 1/384$

5.2 Timer Mode Register (TMM)

The timer mode register is an 8-bit register to specify the operation modes of the two interval timers TIMER0 and TIMER1 and TIMER F/F. Fig. 5-2 shows the timer mode register format.

(1) TF0 and TF1 (bits 0 and 1)

TF0 and TF1 are used to specify TIMER F/F reset or input clock. The internal clock ϕ_3 results from dividing the oscillator frequency by 3.

(2) CK00 and CK01 (bits 2 and 3)

CK00 and CK01 are used to specify TIMER0 input clock. The internal clock ϕ_{12} results from dividing the oscillator frequency by 12. The internal clock ϕ_{384} results from dividing the oscillator frequency by 384.

(3) TS0 (bit 4)

TS0 is used to control TIMER0 UPDOWN counter operation. When the bit is set to 1, UPDOWN counter is cleared to 00H, and increment operation is stopped. When the transition from 1 to 0 is made, the UPDOWN counter starts increment operation at 00H. However, if 0 is written into the bit again after the bit is set to 0 and increment operation is started, the UPDOWN counter is not cleared and continues the increment operation.

(4) CK10 and CK11 (bits 5 and 6)

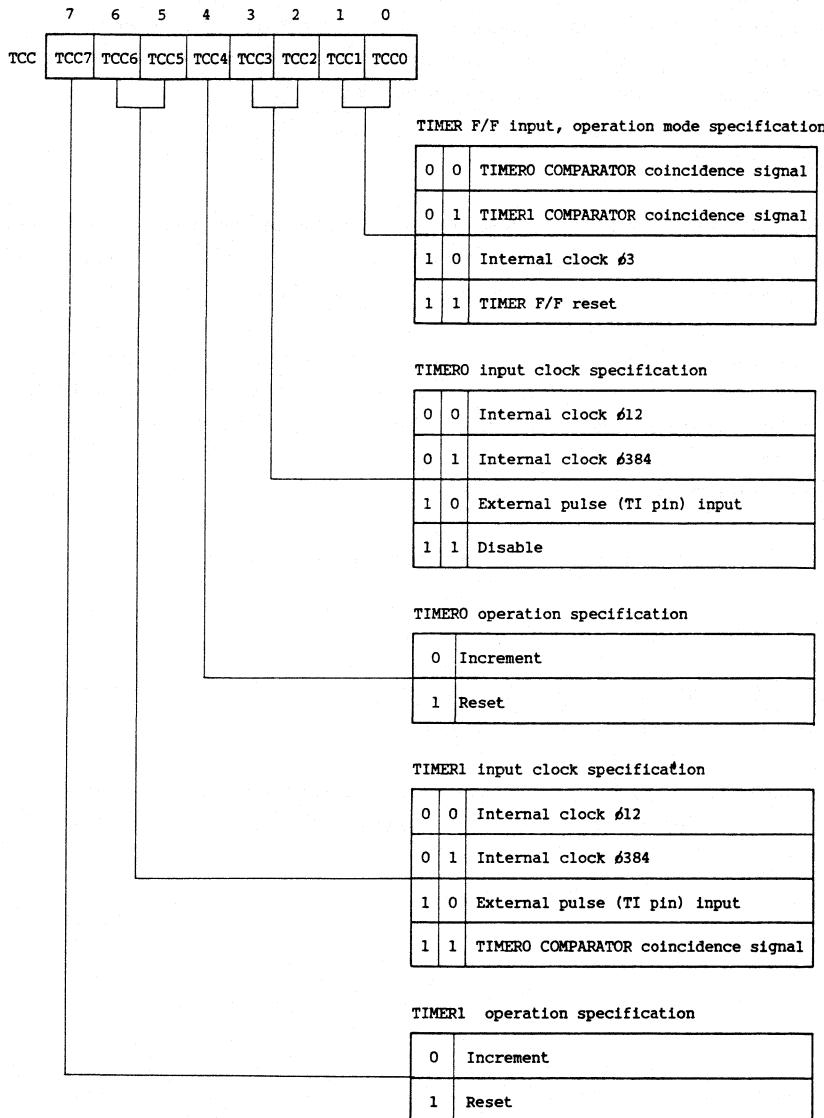
CK10 and CK11 are used to specify TIMER1 input clock.

(5) TS1 (bit 7)

TS1 is the same as TS0 except that the bit is used to control TIMER1 UPDOWN counter operation.

When RESET is input, the timer mode register is set to FFH, TIMER0 and TIMER1 UPDOWN counters are cleared and stop, and TIMER F/F is reset.

Fig. 5-2 Timer Mode Register (TMM) Format



5.3 Timer Operation

Each of the two timers performs interval timer operation according to the input clock (described below) as specified in the timer mode register (TMM):

(1) Internal clock ϕ_{12}

When the internal clock ϕ_{12} is selected for the UPOUNTER input clock, the timer operates as an interval timer in the range of 1 us to 256 us at 12 MHz with the resolution of 1 us.

(2) Internal clock ϕ_{384}

When the internal clock ϕ_{384} is selected for the UPOUNTER input clock, the interval time in the range of 32 us to 8.192 ms at 12 MHz with the resolution of 32 us can be selected.

(3) External pulse (TI)

When the external pulse (TI input) is selected for the UPOUNTER input clock, the timer operates as an interval timer with any desired resolution. It can also be used as an event counter which generates an internal timer interrupt INTT0 or INTT1 when the UPOUNTER counts the external pulses up to the value set in the TIMER REG0 (TM0) (TIMER REG1 (TM1)). However, the count data during counting (UPCOUNTER contents) cannot be read.

To prevent noise signal from causing malfunction, the TI pin is sampled with a sampling pulse of 1-state (250-ns at 12 MHz) period. Thus, an input signal of one state or less is removed, and the TI pin input signal must be held high or low in two or more states.

(4) **TIMER0 output (which can be specified only for TIMER1)**

This mode can be selected only for TIMER1. It operates as a 16-bit interval timer which counts the TIMER0 coincidence signals as TIMER1 UPCOUNTER input. The interval time in the range of 1 us to 65.536 ms or 32 us to 2.1 s at 12 MHz can be selected.

Since TIMER0 and TIMER1 are the same in operation, the TIMER0 operation is explained.

When a count value is set in TIMER REG0 and necessary data is written into the timer mode register, interval timer operation is started. Each time a clock is input, the UPCOUNTER is incremented. The COMPARATOR always compares the incremented UPCOUNTER contents with the TIMER REG0 contents. If they match, the UPCOUNTER generates an internal interrupt INTT0. When they match, the UPCOUNTER is cleared, and again incremented starting at 00H. Thus, the timer serves as an interval timer which generate an interrupt request repeatedly at intervals of the count value set in TIMER REG0. If 0 is set in TIMER REG0, an interrupt is generated at the 256th count.

Caution: When data is written into TIMER REG0, COMPARATOR coincidence signal output is disabled.

If the TIMER0 coincidence signal is selected for TIMER F/F input, when the UPCOUNTER contents match the TIMER REG0 contents, the TIMER F/F contents are inverted and a square wave can be output from the TO pin. The pulse width of the square wave output to the TO pin is determined by the count value set in TIMER REG0. If 0 is set, the TIMER F/F contents are inverted every 256 count.

Timer interrupt INTT0 is disabled by setting MKT0 (interrupt mask register MKL bit 1) to 1.

CHAPTER 6 TIMER/EVENT COUNTER FUNCTION

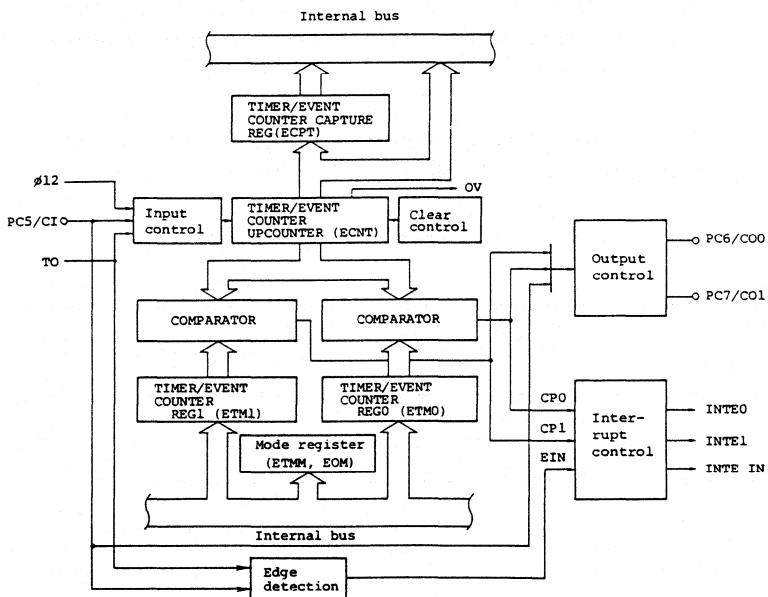
The uPD78C11 contains a multifunctional 16-bit timer/event counter which is used for the following purposes:

- o Interval timer (see 6.3.1)
- o External event counter (see 6.3.2)
- o Frequency measurement (see 6.3.3)
- o Pulse width measurement (see 6.3.4)
- o Programmable square wave output (see 6.3.5)

6.1 Timer/Event Counter Configuration

Fig. 6-1 shows the timer/event counter block diagram.

Fig. 6-1 Timer/Event Counter Block Diagram



Remarks: $\phi_{12} = f_{XX} \times 1/12$

f_{XX} : Oscillator frequency

(1) TIMER/EVENT COUNTER UPCOUNTER (ECNT)

The TIMER/EVENT COUNTER UPCOUNTER (ECNT) is a 16-bit increment counter which counts input pulses. The counter is cleared by the clear control circuit.

When the counter overflows, the OV flag is set. The OV flag can be tested by using the SKIT instruction.

(2) TIMER/EVENT COUNTER CAPTURE register (ECPT)

The TIMER/EVENT COUNTER CAPTURE register (ECPT) is a 16-bit buffer register which retains the ECNT contents. The timing at which the ECPT latches the ECNT contents is determined by input to ECNT as follows:

- On the CI input falling edge if input to ECNT is (5)
 - (i) Internal clock ϕ_{12} or (ii) Internal clock while CI input is high
- On the TO falling edge if input to ECNT is (iii) CI input or (iv) CI input while TO input is high

Table 6-1 Latch Timing in ECPT

ETMM		ECNT input	ECNT latch timing
ET1	ET0		
0	0	Internal clock ϕ_{12}	CI input falling edge
0	1	ϕ_{12} while CI is high	
1	0	CI input	TO falling edge (Note)
1	1	CI input while TO is high	

Note: When the timer F/F input is specified at the internal clock ϕ_3 , To signal is disable (see Fig. 5-1).

(3) TIMER/EVENT COUNTER REG0 and REG1 (ETM0 and ETM1)

The TIMER/EVENT COUNTER REG0 and REG1 (ETM0 and ETM1) are 16-bit registers to set count values.

Caution: If 0 is set in the register, the coincidence signal CP0 or CP1 is generated from the COMPARATOR every 65536 (10000H) count.

(4) COMPARATOR

The COMPARATOR compares the ECNT contents with the TEM0 (ETM1) contents. If they match, the COMPARATOR generates the coincidence signal CP0 (CP1).

(5) Input control circuit

The input control circuit controls input to ECNT. Input to ECNT is determined by timer/event counter mode register (ETMM) specification as follows:

- (i) Internal clock ϕ 12
- (ii) Internal clock while CI input is high
- (iii) CI input
- (iv) CI input while TO input is high

To prevent noise signal from causing malfunction, the CI pin is sampled with a sampling pulse of ϕ 3 (250 ns at 12 MHz) period. Thus, input signal of one state (250 ns at 12 MHz) or less is removed, and the CI pin input signal must be held high or low in two states (500 ns at 12 MHz) or more.

Table 6-2 ECNT Input

ETMM		ECNT input
ET1	ET0	
0	0	Internal clock ϕ_{12}
0	1	ϕ_{12} while CI is high
1	0	CI input
1	1	CI input while TO is high

(6) Clear control circuit

The clear control circuit clears ECNT. The clear mode is selected by using the timer/event counter mode register (ETMM) among the following:

- (i) Remain cleared
- (ii) Not clear
- (iii) Match between ECNT and ETM1
- (iv) CI input falling edge or TO falling edge

In (iv), the ECNT clear mode is selected as listed in Table 6-3 according to ECNT input.

Table 6-3 ECNT Clear

ETMM				ECNT input	ECNT clear
EM1	EM0	ET1	ETO		
0	0	x	x	Don't care	Stop after clear
0	1	x	x		
1	0	0	0	Internal clock ϕ_{12}	CI input falling edge
		0	1	ϕ_{12} while CI is high	
		1	0	CI input	TO falling edge (Note)
		1	1	CI input while TO is high	
1	1	x	x	Don't care	Match between ECNT and ETM1

Note: When the timer F/F input is specified at the internal clock ϕ_3 , To signal is disable (see Fig. 5-1).

If (iv) is specified in the clear mode, clear operation is performed after capture operation.

(7) Interrupt control circuit

The interrupt control circuit controls timer/event counter interrupts. There are the following interrupt sources (the corresponding request flag is set to 1):

- (i) Coincidence signal between ECNT and ETM0 → INTE0
- (ii) Coincidence signal between ECNT and ETM1 → INTE1
- (iii) CI input falling edge or TO falling edge → INTEIN

In (iii), the interrupt request flag INTEIN is set as listed in Table 6-4 according to ECNT input as in (6) (ii).

Table 6-4 Interrupt Request Flag INTEIN Setting

ETMM		ECNT input	Interrupt request flag setting
ET1	ETO		
0	0	Internal clock ϕ_{12}	
0	1	ϕ_{12} while CI input is high	CI input falling edge
1	0	CI input	
1	1	CI input while TO is high	TO falling edge (Note)

Note: When the timer F/F input is specified at the internal clock ϕ_3 , To signal is disable (see Fig. 5-1).

(8) Output control circuit

The output control circuit controls pulse outputs of two channels, CO0 and CO1. In conjunction with the timer/event counter, the output control circuit can change the pulse width and period. Pulse output changes by any of the following signals:

- (i) Match between ECNT and ETM0
- (ii) Match between ECNT and ETM1
- (iii) CI input falling edge

(9) Mode registers (ETMM and EOM)

The mode registers are 8-bit registers to specify timer/event counter and output control circuit operation. (For details, see 6.2.)

6.2 Mode Registers

The timer/event counter block contains the timer/event counter mode register (ETMM) to specify the operation mode and the timer/event counter output mode register (EOM) to specify the output control circuit operation.

6.2.1 Timer/event counter mode register (ETMM)

The timer/event counter mode register (ETMM) is an 8-bit register to specify the timer/event counter operation mode. Fig. 6-2 shows the timer/event counter mode register format.

(1) ET0 and ET1 (bits 0 and 1)

ET0 and ET1 are used to specify the TIMER/EVENT COUNTER UPDOWN COUNTER (ECNT) input clock, latch timing, INTEIN interrupt flag setting condition. When EM1 = 1 and EM0 = 0, the bits are also used to specify the clear mode.

Internal clock ϕ_{12} results from dividing the oscillator frequency by 12.

(2) EM0 and EM1 (bits 2 and 3)

EM0 and EM1 are used to control the ECNT clear mode. When the EM0 and EM1 bits are set to 00, ECNT is cleared to 0000H and not incremented.

In the EM0 and EM1 bits are not set to 00, ECNT is incremented according to the input clock, cleared under the condition shown in Fig. 6-2, and again incremented starting at 0000H. When EM0 = 0 and EM1 = 1 are set, the ECNT clear condition becomes as follows according to input clock specification:

- When ET1 = 0 and ET0 = 0 or ET1 = 1 and ET0 = 1, ECNT is cleared on the CI input falling edge. (See 6.3.4.)
- When ET1 = 1 and ET0 = 0 or ET1 = 1 and ET0 = 1, ECNT is cleared on the TO falling edge. (See 6.3.3.)

(3) CO00 and CO01 (bits 4 and 5)

CO00 and CO01 are used to specify the transfer timing of the level F/F (LV0) level to the output latch shown in Fig. 6-3. When CO00 = 0 and CO01 = 1, the LV0 level is transferred to the output latch when either a match between ECNT and ETM0 or the CI input falling edge occurs. When CO00 = 1 and CO01 = 1, the LV0 level is transferred to the output latch when either a match between ECNT and ETM0 or a match between ECNT and ETM1 occurs.

If the LD0 bit of the timer/event counter output mode register (EOM) is set to 1, the LV0 level is inverted after it is transferred to the output latch.

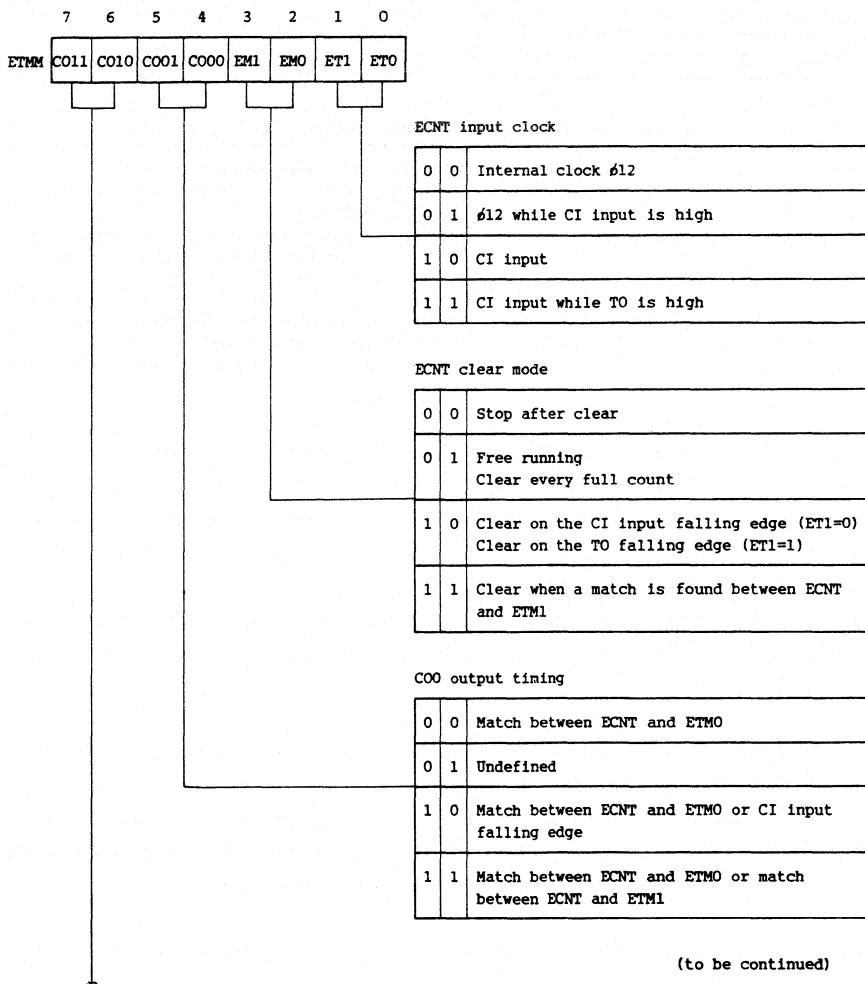
(4) CO10 and CO11 (bits 6 and 7)

Like the CO00 and CO01 bits, CO10 and CO11 are used to specify the transfer timing of the level F/F (LV1) level to the output latch. When CO10 = 0 and CO11 = 1 or CO10 = 1 and CO11 = 1, the LV1 level is transferred to the output latch under the condition shown in Fig. 6-2.

If the LD1 bit of the timer/event counter output mode register (EOM) is set to 1, the LV1 level is inverted after it is transferred to the output latch.

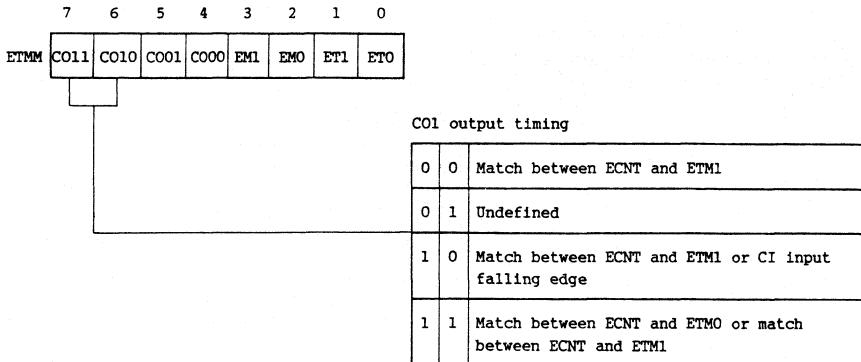
When RESET is input or during the hardware STOP mode, the timer/event counter mode register is reset to 00H.

Fig. 6-2 Timer/Event Counter Mode Register Format



(to be continued)

Fig. 6-2 Timer/Event Counter Mode Register Format (Cont'd)



6.2.2 Timer/event counter output mode register (EOM)

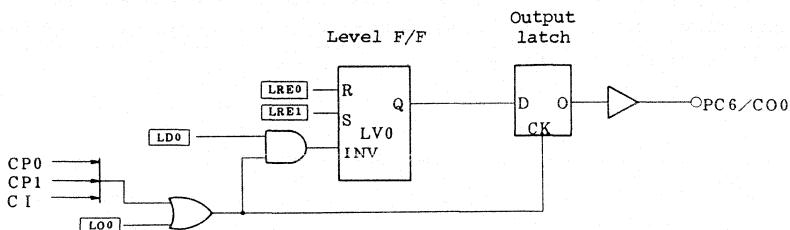
The timer/event counter output mode register (EOM) is an 8-bit register to control timer/event counter output circuit operation.

First, the output control circuit configuration and function are explained. Fig. 6-3 shows output control circuit CO0 output block diagram.

CO0 output is of the master-slave type. The level F/F (LV0) at the first stage retains the next level to be output. The output latch at the next stage is provided to output the LV0 level to the external. The CO0 output timing from the LV0 to the external is specified in the timer/event counter mode register.

The CO1 output configuration is the same as the CO0 output configuration.

Fig. 6-3 Output Control Circuit Block Diagram (CO0 output)



The timer/event counter output mode register is used to control initialization and operation of the output control circuit described above. Fig. 6-4 shows the timer/event counter output mode register format.

(1) LO0 and LO1 (bits 0 and 4)

When LO0 or LO1 is set to 1, the level F/F (LV0 or LV1) level is output to the output pin. When the level is output, automatically the bit is reset to 0.

(2) LD0 and LD1 (bits 1 and 5)

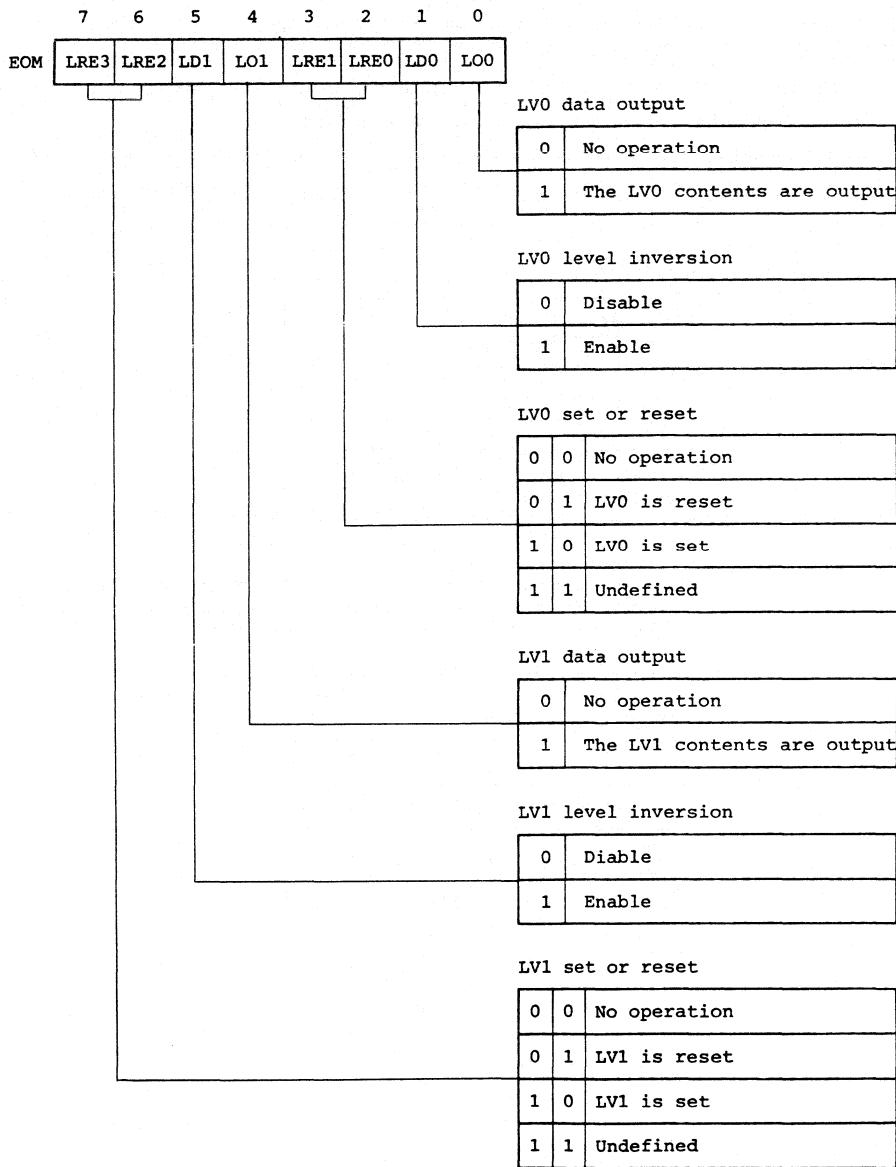
LD0 or LD1 is used to specify whether or not the LV0 or LV1 level is inverted at the output timing specified in the timer/event counter mode register. If LD0 or LD1 is set to 1, the LV0 or LV1 level is inverted at the specified output timing. If LD0 or LD1 is reset to 0, the LV0 or LV1 level is not inverted.

(3) LRE0, LRE1, LRE2, and LRE3 (bits 2, 3, 6, and 7)

These bits are used to set or reset the level F/F. If LRE0 or LRE2 is set to 1, LV0 or LV1 is reset; if LRE1 or LRE3 is set to 1, LV0 or LV1 is set.

When the level F/F is set or reset, automatically the bits are restored to 0. When RESET is input or during the hardware STOP mode, the timer/event counter output mode register is reset to 00H.

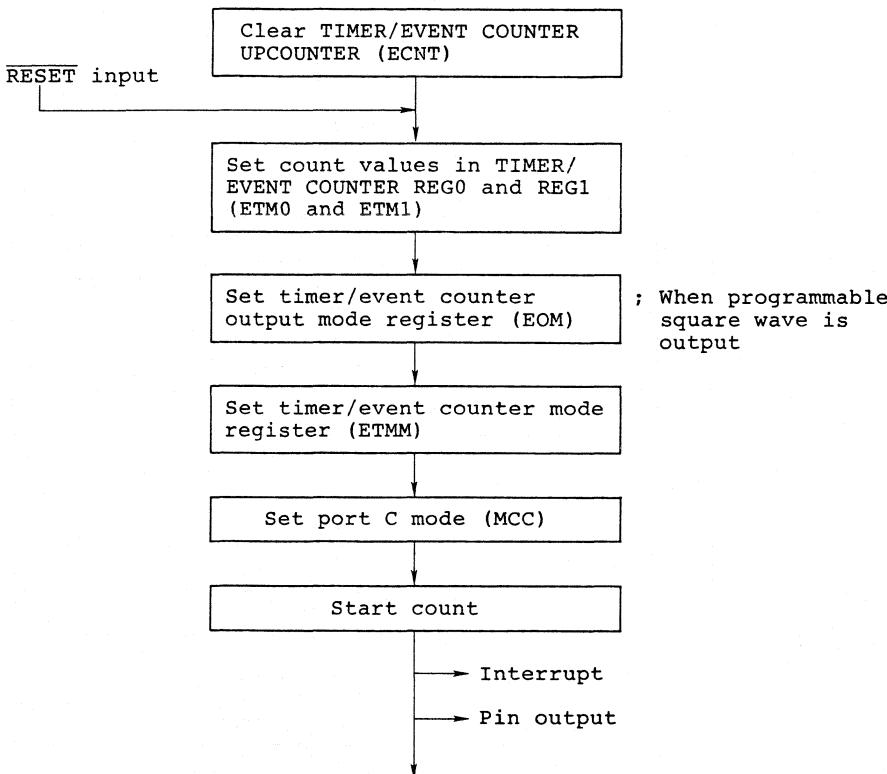
Fig. 6-4 Timer/Event Counter Output Mode Register Format



6.3 Timer/Event Counter Operation

Timer/event counter operation is started by setting the count value and operation mode as shown in Fig. 6-5. Once the operation mode is set, the timer/event counter operates in the mode unless the mode registers are again set.

Fig. 6-5 Timer Event Counter Setting Procedure

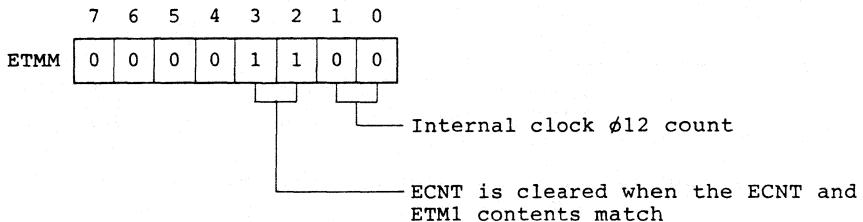


6.3.1 Interval timer mode

In the interval timer mode, the timer/event counter serves as an interval timer which generates an interrupt repeatedly at intervals of the setup count time. The interval timer can count in the range of 1 us to 65.535 ms at 12 MHz with the resolution of 1 us.

First, the TIMER/EVENT COUNTER UP COUNTER (ECNT) is cleared, then the count values are set in TIMER/EVENT COUNTER REG0 and REG1 (ETM0 and ETM1). Next, the data shown in Fig. 6-6 is set in the timer/event counter mode register (ETMM). Then, the timer/event counter serves as an interval timer with internal clock ϕ 12 as an input clock.

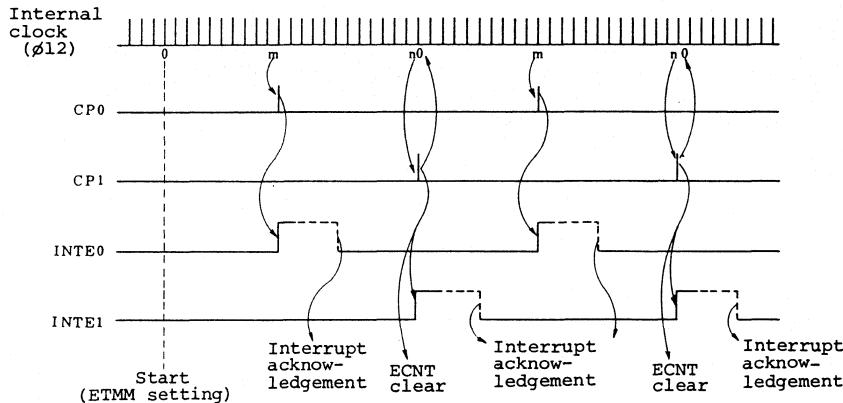
Fig. 6-6 Timer/Event Counter Mode Register Setting



ECNT is incremented every us. The COMPARATOR compares the incremented ECNT contents with the ETM0 (ETM1) contents. If they match, the COMPARATOR generates an internal interrupt INTE0 (INTE1) according to the coincidence signal CP0 (CP1). Only when a match between ECNT and ETM1 occurs, the ECNT contents are cleared, and again incremented starting at 0000H. Thus, the timer/event counter serves as an interval timer which generates an interrupt repeatedly at intervals of the count time based on the count value set in ETM1. (See Fig. 6-7.)

An internal interrupt is disabled by setting the interrupt mask register (MKL) MKE0 (MKE1) bit to 1.

Fig. 6-7 Interval Timer Mode Operation



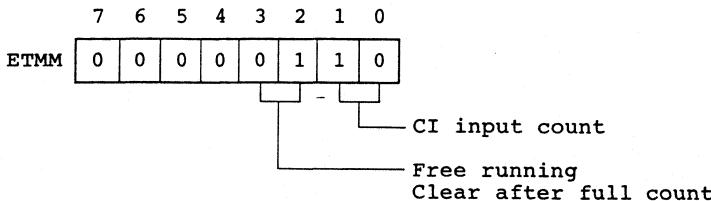
Remarks: ETM0 = m (m < n: m, n; count value)
ETM1 = n

6.3.2 Event counter mode

In the event counter mode, the timer/event counter counts external pulses input to the CI pin (PC5).

First, ECNT is cleared, then the data shown in Fig. 6-8 is set in the timer/event counter mode register. Counting external events is started.

Fig. 6-8 Timer/Event Counter Mode Register Setting



An external pulse input to the CI pin is synchronized with the internal clock, and ECNT is incremented on the falling edge.

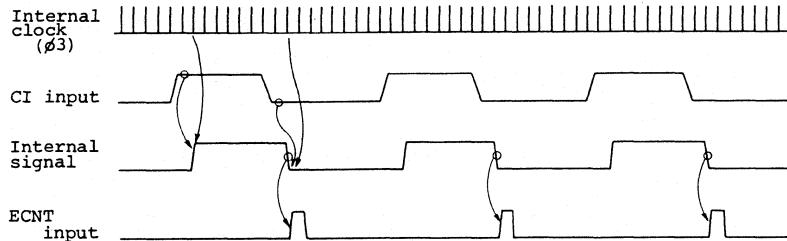
The pulse width of a pulse input to the CI pin must be 500 ns or more at 12 MHz. The pulse whose width is 250 ns or less is regarded as a noise signal and is not counted.

The count value can always be read by software.

If the timer/event counter mode register is set as shown in Fig. 6-8, when ECNT is incremented to FFFFH, the OV (overflow) flag is set and again the ECNT is set to 0000H for increment. The OV flag does not contain the interrupt function, but can be tested by using the skip instruction SKIT or SKNIT in a program.

When external events are counted up to the value set in ETM0 or ETM1, an internal interrupt INTE0 or INTE1 occurs.

Fig. 6-9 Event Counter Mode Operation

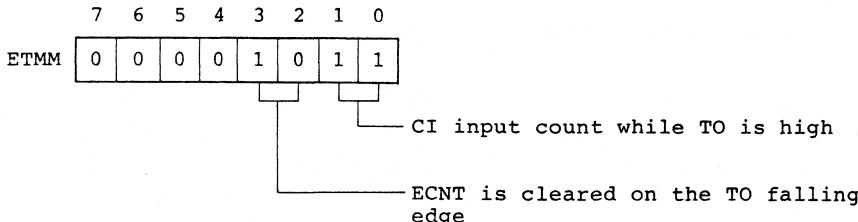


6.3.3 Frequency measurement mode

In the frequency measurement mode, the timer/event counter is used to measure the frequency of an external pulse input to the CI pin. Since the external pulses are counted while timer output (TO) is high (reference time) in the frequency measurement mode, the timer must previously be operated.

First, ECNT is cleared, then the data shown in Fig. 6-10 is set in the timer/event counter mode register. Then, frequency measurement mode operation is started.

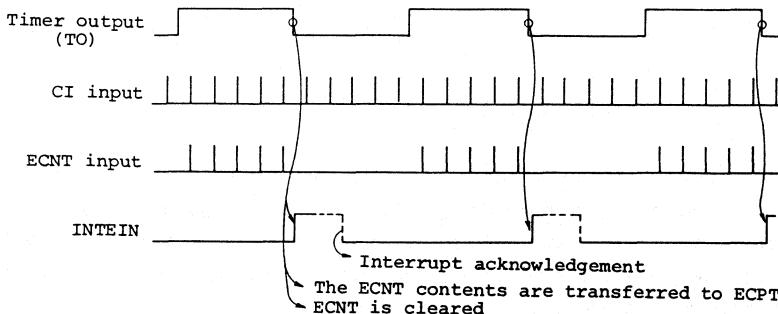
Fig. 6-10 Timer/Event Counter Mode Register Setting



ECNT counts external pulses input to the CI pin while timer output (TO) is high. On the falling edge of the timer output, the ECNT contents are transferred to the TIMER/EVENT COUNTER CAPTURE register (ECPT), the ECNT is cleared, and an internal interrupt INTEIN occurs. (See Fig. 6-11.)

The ECNT is cleared and the interrupt occurs on the TO falling edge because input to the ECNT is CI input while the TO is high. (See 6.1 (6) and (7).)

Fig. 6-11 Frequency Measurement Mode Operation

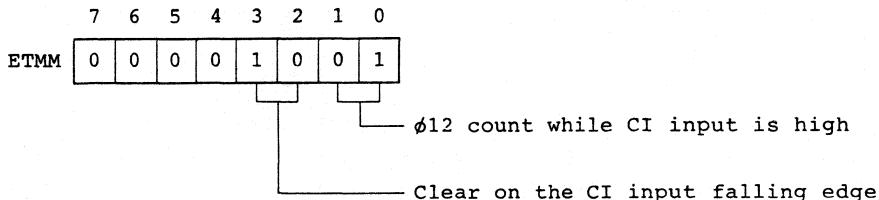


6.3.4 Pulse width measurement mode

In the pulse width measurement mode, the timer/event counter is used to measure the high width of an external pulse input to the CI pin.

First, ECNT is cleared, then the data shown in Fig. 6-12 is set in the timer/event counter mode register (TEMMP). Then, the pulse width measurement mode operation is started.

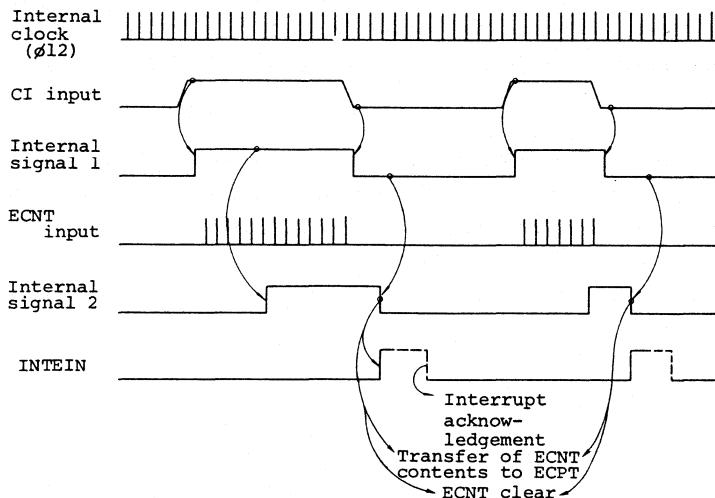
Fig. 6-12 Timer/Event Counter Mode Register Setting



On the CI input rising edge, internal clock ϕ_{12} is supplied to the ECNT and count is started. While CI input is high, the ECNT continues counting the internal clocks. On the CI input falling edge, supply of the internal clock to the ECNT is stopped; the ECNT contents are transferred to the ECPT; the ECNT is cleared; and an internal interrupt INTEIN is generated. (See Fig. 6-13.) The transfer of the ECNT contents, ECNT clear, and interrupt generation are performed on the CI input falling edge. (See 6.1 (2), (6), and (7).)

The high or low pulse width input to the CI pin in the pulse width measurement mode must be 16 states (4 us at 12 MHz) or more. If the pulse width is 12 states or less, the transfer of the ECNT contents to the ECPT, ECNT clear, and interrupt generation are not performed.

Fig. 6-13 Pulse Width Measurement Mode Operation



6.3.5 Programmable square wave output mode

The programmable square wave output mode enables output of programmable square waves to two independent inputs C00 and C01.

Since C00 and C01 are the same in operation, programmable square wave output to C00 output is explained.

First, the ECNT is cleared, then count values are set in ETM0 and ETM1. Next, the data shown in Fig. 6-14 is set in the timer/event counter output mode register (EOM) to specify output control circuit initialization and operation. The data shown in Fig. 6-15 is set in the timer/event counter mode register. Timer/event counter operation is started.

Fig. 6-14 Timer/Event Counter Output Mode Register Setting

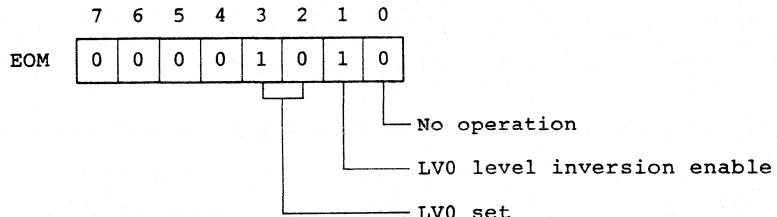
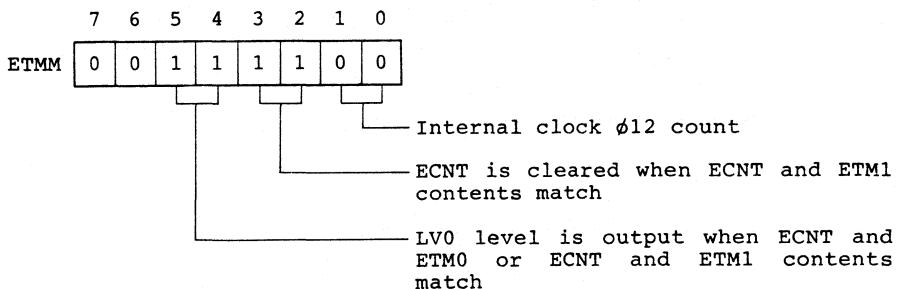


Fig. 6-15 Timer Event Counter Mode Register Setting



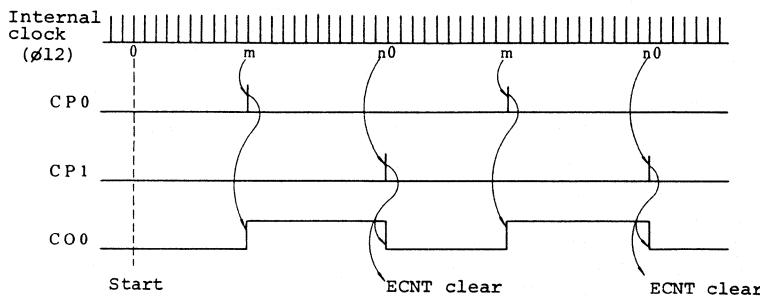
As in the interval timer mode, the ECNT is incremented every $\phi 12$. The COMPARATOR compares the incremented ECNT contents with the ETM0 (ETM1) contents. If they match, the COMPARATOR generates the coincidence signal CP0 (CP1) and internal interrupt INTE0 (INTE1). The output control circuit outputs the level F/F (LV0) contents to the CO pin according to the coincidence signal and inverts the LV0 contents.

Only when a match between ECNT and ETM1 occurs, the ECNT contents are cleared, and again incremented starting at 0000H. Thus, square wave with the count time based on the count value set in ETM0, ETM1 as the pulse width is output from C00. (See Fig. 6-16.)

An internal interrupt INTE0 (INTEL) is disabled by setting the interrupt mask register (MKL) MKE0 (MKE1) bit to 1.

By changing mode register setting, square wave output from the CO1 pin is also performed as square wave output from the CO0 pin.

Fig. 6-16 Programmable Square Wave Output Mode Operation



Remarks: ETM0 = m (m < n: m and n are count value)
ETM1 = n

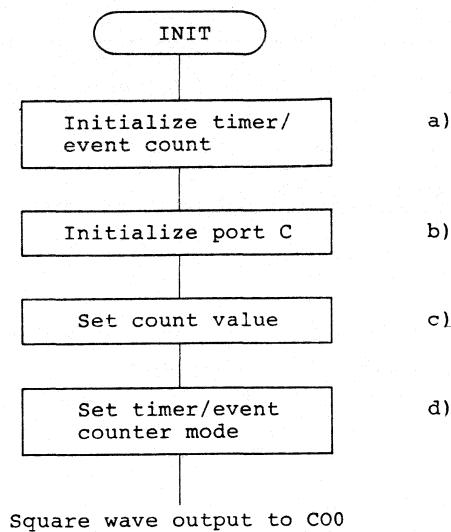
6.3.6 Timer/event counter program examples

Two timer/event counter program examples are given: Programmable square wave output and single pulse output in synchronization with the CI input falling edge.

(1) Programmable square wave output

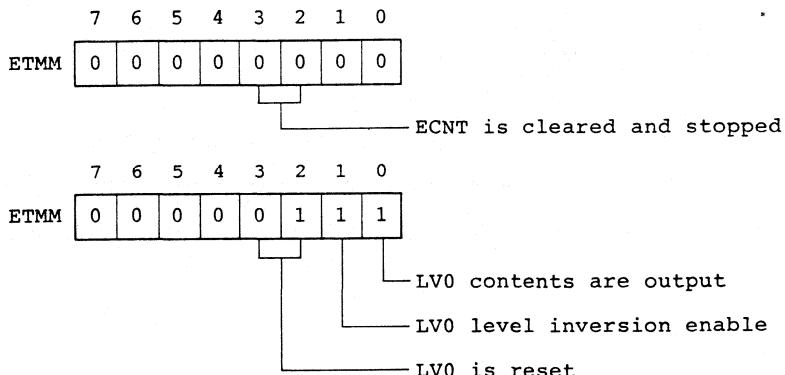
Programmable square wave is output from the CO0 pin as shown in Fig. 6-16. A program example is given where the low pulse width is 200 us and the high width is 300 us at 12 MHz.

Operation flow is shown below:



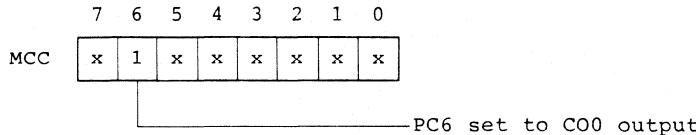
- a) The timer/event counter ECNT is cleared and CO0 output is set low. To set CO0 output low, LV0 is reset and the level is output to CO0.

Fig. 6-17 Timer/Event Counter Mode Register Setting



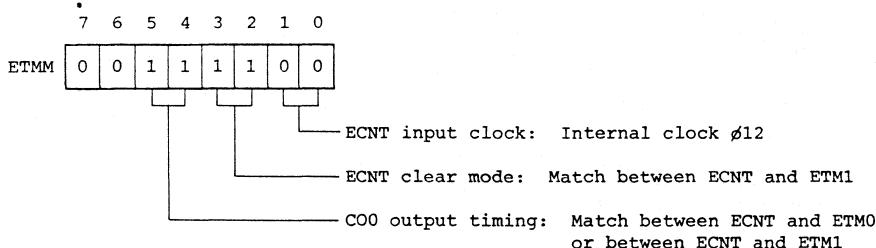
- b) The PC6 pin of port C is set to CO0 output.

Fig. 6-18 Port C Setting



- c) To determine the low width and period of square wave output to the CO0 pin, 00C8H (low width = 200 us) is set in ETM0 (TIMER/EVENT COUNTER REG0) and 01F4H (period = 500 us) is set in ETM1. (at 12 MHz)
- d) The timer/event counter operation mode is specified by using the timer/event counter mode register (ETMM). The following are set: ECNT input clock = internal clock $\phi 12$, ECNT clear mode = match between ECNT and ETM1, and CO0 output timing = match between ECNT and ETM0 or ECNT and ETM1. Timer/event counter operation is started by setting the timer/event counter mode register.

Fig. 6-19 Timer/Event Counter Mode Register Setting



- e) To output high to the CO0 pin when the first coincidence signal CP0 is generated by the comparator, LV0 is set.

;***TIMER/EVENT COUNTER INITIALIZATION*****

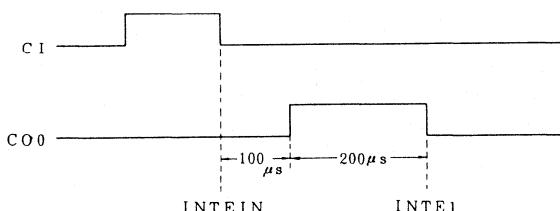
```

INIT  : MVI   A, 00H
        MOV   ETMM, A ; Clear timer/event counter }a
        MVI   EOM, 07H ; Initialize counter output 0 }
        MVI   A, 40H ; PC6 : CO0 }b
        MOV   MCC, A ; Set Port C mode control
        LXI   EZ, 00C8H ; Low level : 200 us at 12 MHz
        DMOV  ETM0, EA ; Set count value }c
        LXI   EA, 01F4H ; High level : 300 us at 12 MHz
        DMOV  ETM1, EA ; Set count value
START : MVI   A, 3CH
        MOV   ETMM, A ; Set timer/event counter mode & start }d
        ORI   EOM, 08H ; Set LV0 }e
    
```

(2) Single pulse output

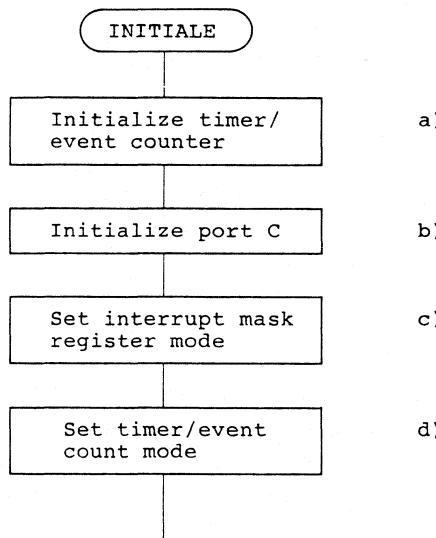
A single pulse is output to the CO0 pin in given time after the CI input falling edge as shown in Fig. 6-20. A program example is given where a pulse whose high width is 200 us is output in 100 us after the CI input falling edge. (at 12 MHz)

Fig. 6-20 Single Pulse Output



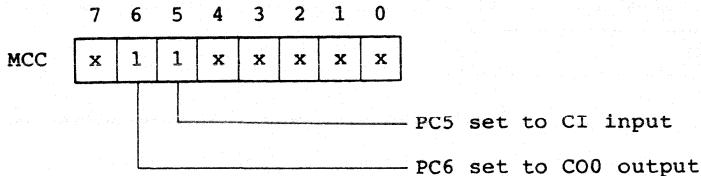
To perform this example operation, an initialization program, a service routine of internal interrupt INTEIN occurring on the CI input falling edge, and a service routine of internal interrupt INTEL1 occurring when the ECNT and ETM1 contents match are required.

First, the initialization routine is explained. Operation flow is shown below:



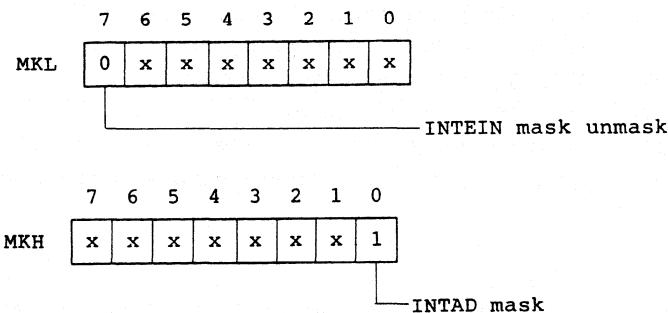
- a) The timer/event counter ECNT is cleared, CO0 output is set low, and LV0 is set, as in (1) a).
- b) The PC5 pin of port C is set to CI input and the PC6 pin is set to CO0 output.

Fig. 6-21 Port C Setting



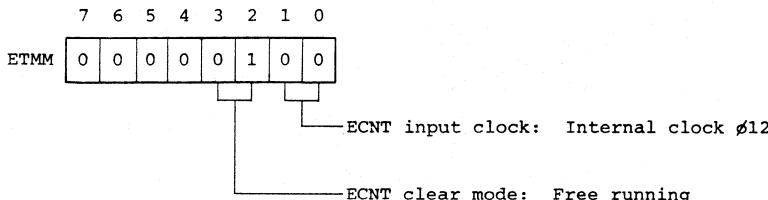
- c) Internal interrupt INTEIN mask is unmasked by using the interrupt mask register (MKL). Assume that internal interrupt INTAD assigned the same priority level as INTEIN is masked by using the interrupt mask register (MKH).

Fig. 6-22 Interrupt Mask Register Setting



- d) The timer/event counter operation mode is set by using the timer/event counter mode register (ETMM). The following are set in the timer/event counter mode register (ETMM): ECNT input clock = internal clock ϕ 12 and ECNT clear mode = free running. Timer/event counter operation is started by setting the timer/event counter mode register.

Fig. 6-23 Timer/Event Counter Mode Register Setting



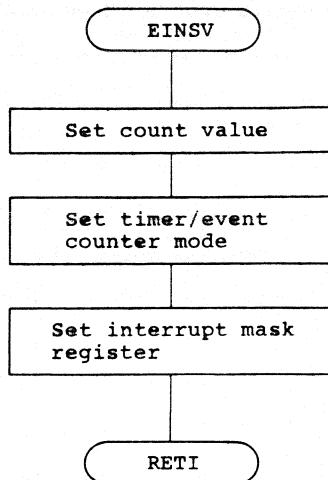
An initialization program example is given below:

;***TIMER/EVENT COUNTER INITIALIZATION*****

```
INIT  : MVI A, 00H
      MOV ETMM, A ; Clear timer/event counter } a
      MVI EOM, 05H ; Initialize counter output 0
      MVI A, 60H ; PC5 : CI, PC6 : CO0 } b
      MOV MCC, A ; Set Port mode control
      ANI MKL, 7FH ; INTEIN enable } c
      START : MVI A, 04H ; ECNT free running
      MOV ETMM, A ; Set timer/event counter mode & start } d
```

On the CI input falling edge after initialization, the ECNT value during free running (value of the CI input falling edge) is latched in ECPT (TIMER/EVENT COUNTER CAPTURE REG) and internal interrupt INTEIN is generated.

The interrupt service operation flow is shown below:



a)

b)

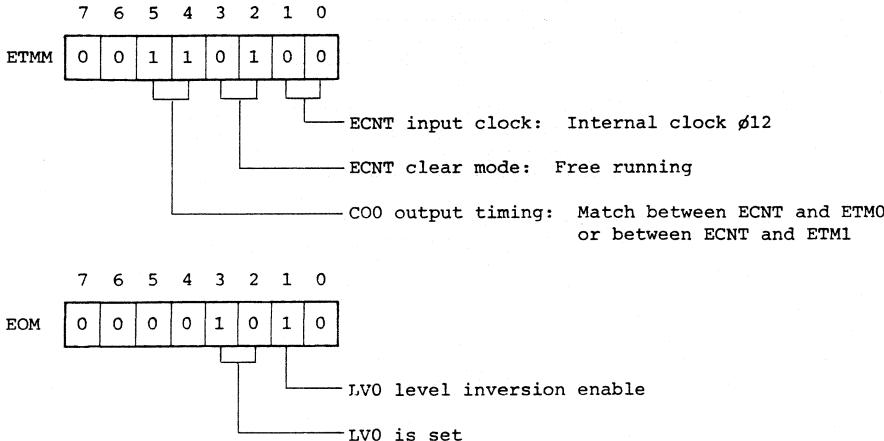
c)

- a) To output a pulse whose width is 200 us to the CO0 pin in 100 us after the CI input falling edge, 0064H (100 us) added to the ECPT value is set in ETM0 and 012CH (300 us) added to the ECPT value is set in ETM1. (at 12 MHz)
- b) CO0 output timing = match between ECNT and ETM0 or ECNT and ETM1 is set in the timer/event counter mode register (ETMM).

The ECNT input clock and ECNT clear mode remain unchanged.

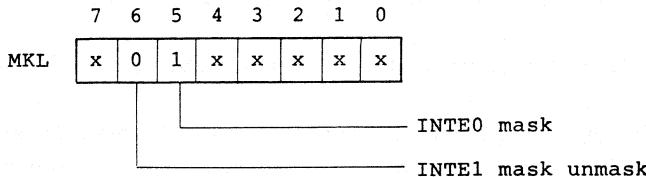
Output control circuit LV0 is set and LC0 level inversion is enabled by setting the timer/event counter output mode register (EOM).

Fig. 6-24 Timer/Event Counter Mode Register Setting



- c) Mask of internal interrupt INTE1 caused by a match between ECNT and ETM1 is unmasked by setting the interrupt mask register (MKL). Assume that INTE0 assigned the same priority level as INTE1 is masked.

Fig. 6-25 Interrupt Mask Register Setting



The INTEIN interrupt service program is shown below: (JMP EINSV instruction must be stored in the INTEIN interrupt start address 0020H.)

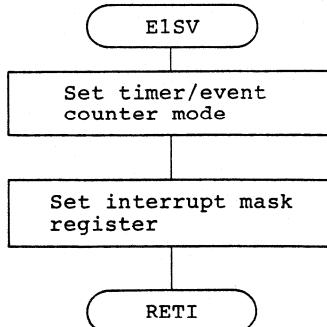
;***TIMER/EVENT COUNTER SERVICE*****

```
EINSV : EXA           ; Save accumulator
        EXX           ; Save register
        DMOV EA, EXPT
        LXI B, 0064H
        DADD EA, B    ; Low level : 100 us at 12 MHz
        DMOV TEM0, EA ; Set count value
        LXI B, 00C8H
        DADD EA, B    ; High level : 200 us at 12MHz
        DMOV ETM1, EA ; Set count value
        MVI A, 34H
        MOV ETMM, A
        ORI EOM, 0AH ; Set level F/F, inversion enable
        ANI MKL, 0BFH ; INTEIN, INTEL enable
        EXX           ; Recover register
        EXA           ; Recover accumulator
        EI
        RETI
```

} a
} b
} c

If the ECNT and ETM1 contents match after the INTEIN interrupt service program is executed, an internal interrupt INTEL occurs.

The interrupt service flow is shown below:



- a) CO0 output operation is stopped by setting the timer/event counter output mode register (EOM).
- b) INTEL interrupt is masked (disabled) by setting the interrupt mask register (MKL).

;*****TIMER/EVENT COUNTER SERVICE*****

```
E1SV : MVI EOM, 00H           a)  
       ORI MKL, 40H ; INTEL disable   b)  
       EI  
       RETI
```

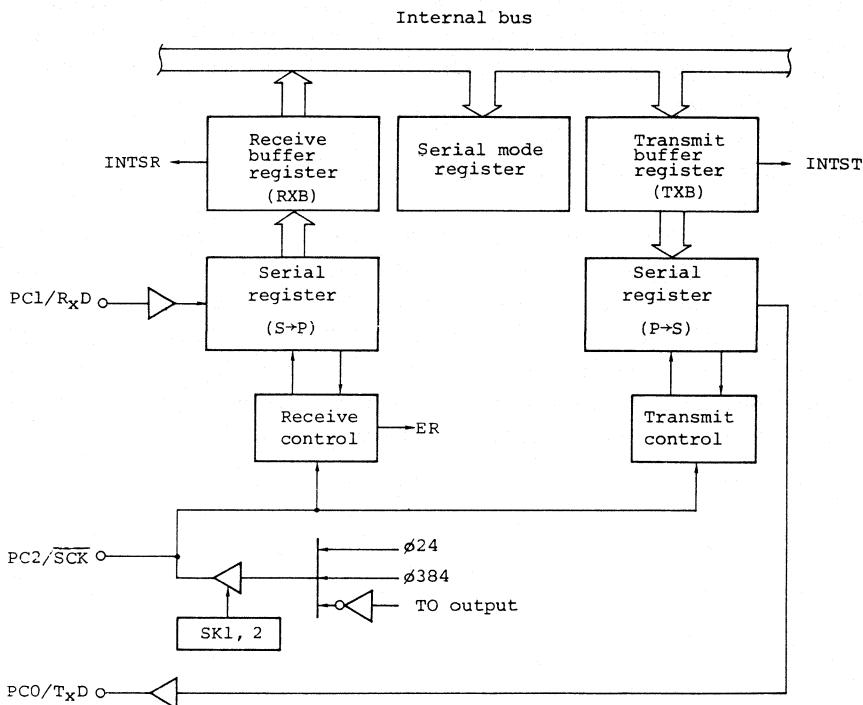
CHAPTER 7 SERIAL INTERFACE FUNCTION

The uPD78C11 contains a serial interface to enable distributed processing and connection of various terminals. The serial interface operates in the asynchronous mode, synchronous mode, or I/O interface mode.

7.1 Serial Interface Configuration

The serial interface consists of the three pins of serial data input RxD, serial data output TxD, and serial clock input/output SCK, the transmission block containing an 8-bit serial register, an 8-bit buffer register, and transmit control, the reception block containing an 8-bit serial register, an 8-bit buffer register, and receive control, and two 8-bit mode registers to specify the operation mode, as shown in Fig. 7-1.

Fig. 7-1 Serial Interface Configuration



$$\text{Remarks: } \phi_{24} = f_{xx} \times \frac{1}{24}$$

$$\phi_{384} = f_{xx} \times \frac{1}{384}$$

f_{xx} : Oscillator frequency (MHz)

(1) Transmission block

(a) Serial register (parallel to serial)

Parallel data transferred from the transmit buffer register is converted into serial data and the serial data is transmitted from the TxD pin.

(b) Transmit buffer register

The transmit buffer register is an 8-bit register into which parallel data to be transmitted is written. When data in the serial register has been transmitted, the transmit buffer register contents are transferred to the serial register. When the buffer register becomes empty, an interrupt request INTST occurs.

(c) Transmit control circuit

The transmit control circuit performs all control required to transmit serial data and generates related internal signals.

(2) Reception block

(a) Serial register (serial to parallel)

Serial data input from the RxD pin is converted into parallel data and the parallel data is transferred to the receive buffer register.

(b) Receive buffer register

The parallel data into which serial data is converted in the serial register is transferred to the receive buffer register. When the receive buffer register becomes full, an interrupt request INTSR occurs.

(c) Receive control circuit

The receive control circuit performs all control required to receive serial data. When a serial error occurs, the ER flag is set. The ER flag can be tested by using the SKIT instruction.

Although the ER flag is reset, the receive buffer is not affected.

(3) Serial mode registers

The serial mode registers are two 8-bit registers to control the serial interface operation mode. (For details, see 7.2.)

Since each of the transmission and reception blocks contains a serial register and buffer register, serial data can be transmitted and received independently (full duplex double buffer system transmitter receiver). However, since the serial clock \overline{SCK} is common to transmit and receive, half duplex is applied in the synchronous mode or I/O interface mode.

7.2 Serial Mode Registers

The serial mode registers are two 8-bit registers to specify the serial interface operation mode, serial clock, data format, etc.

7.2.1 Serial mode high register (SMH)

The serial mode high register bits are used to specify the following operation modes: (Fig. 7-2 shows the serial mode high register format.)

(1) SK1 and SK2 (bits 0 and 1)

SK1 and SK2 are used to specify which of internal clock and external clock is used for serial clock \overline{SCK} .

If internal clock is specified for serial clock, the serial clock value is determined by one of the following expressions:

When internal clock 24 is used

$$SCK = f_{XX}/24$$

When internal clock $\phi 384$ is used

$$SCK = f_{XX}/384$$

When Internal clock TO output is used

When timer input clock is $\phi 12$

$$SCK = f_{XX}/24 \times C$$

When timer input clock is $\phi 384$

$$SCK = f_{XX}/768 \times C$$

When TIMER F/F input is $\phi 3$

$$SCK = f_{XX}/6$$

Where f_{XX} is the oscillator frequency, \overline{SCK} is serial clock, and C is the timer count value. When TIMER F/F input is $\phi 3$ when internal clock TO output is used, it can only be used at the clock rate of 16 or 64 in the asynchronous mode.

(2) TxE (bit 2)

TxE is used to specify whether or not transmit operation is to be performed. When the TxE bit is reset to 0, the TxD pin is set high and data is not transmitted. When the TxE bit is set to 1, data transmission is enabled. If data is prewritten into the transmit buffer register, the data is output, or when data is written into the transmit buffer register, serial data is transmitted from the TxD pin.

However, if the transition of the TxE bit state from 1 to 0 is made, transmission is disabled after the data in the serial register has been transmitted. Thus, if both the transmit buffer register and serial register contain data, transmission is disabled after the data in the serial register has been transmitted, and the data in the transmit buffer register is not transmitted and retained. The data in the transmit buffer register is transmitted as serial data when TxE = 1 (transmit enable state) is again set. Thus, to disable transmission (TxE = 0) after all send data is transmitted, check that the serial transmit interrupt request flag INTFST is set to 1 and the transmit buffer register becomes empty before disabling transmission.

(3) RxE (bit 3)

RxE is used to specify whether or not receive operation is to be performed. When the RxE bit is reset to 0, data is not received. When the RxE bit is set to 1, data reception is enabled.

(4) SE (bit 4)

SE is used to specify whether or not the search mode is to be entered during the synchronous mode (selected in the serial mode low register SML)). If the SE bit is set to 1, each time 1-bit data is received, the serial register contents are transferred to the receive buffer register and serial receive interrupt INTSR is generated. If the SE bit is reset to 0, each time 8-bit data is received, the serial register contents are transferred to the receive buffer register and serial receive interrupt INTSR is generated.

(5) IOE (bit 5)

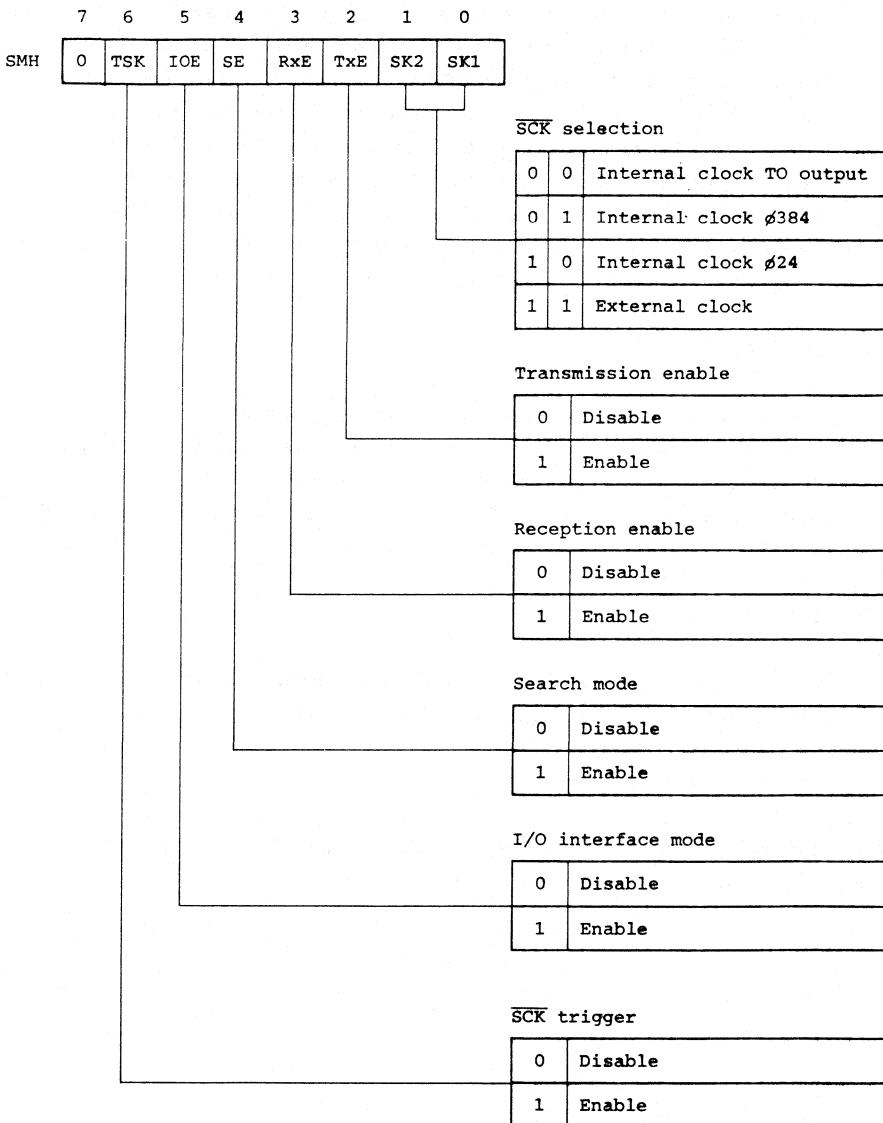
IOE is used to specify which of the synchronous mode and I/O interface mode is entered during synchronous operation (selected in the serial mode low register (SML)). When the IOE bit is reset to 0, the synchronous mode is entered; when 1, the I/O interface mode is entered.

(6) TSK (bit 6)

When data is received by using internal clock in the I/O interface mode, serial clock is started. When the TSK bit is set to 1 and the serial clock is started, automatically the bit is reset to 0.

When RESET is input or during the hardware STOP mode, the serial mode high register (SMH) is reset to 00H.

Fig. 7-2 Serial Mode High Register (SMH) Format



7.2.2 Serial mode low register (SML)

The serial mode low register bits are used to specify the following operation modes: (Fig. 7-3 shows the serial mode low register format.)

(1) B1 and B2 (bits 0 and 1)

B1 and B2 are used to select the synchronous operation or the data transfer rate during the asynchronous mode. In the asynchronous mode, the serial clock is divided at the clock rate selected in the bits, and data is transferred. To perform synchronous operation, set the B1 and B2 bits to 00.

(2) L1 and L2 (bits 2 and 3)

L1 and L2 are used to select the bit length of one character.

(3) PEN (bit 4)

PEN is used to specify whether or not an odd or even parity is added to transfer data (during transmitting) and an odd or even parity check is made on transfer data (during receiving).

When the PEN bit is set to 1, a parity bit is added to each character and the data is transmitted; a parity check is made on the received data, and if a parity error is detected, the error flag is set.

When the PEN bit is reset to 0, parity addition and parity check are not made.

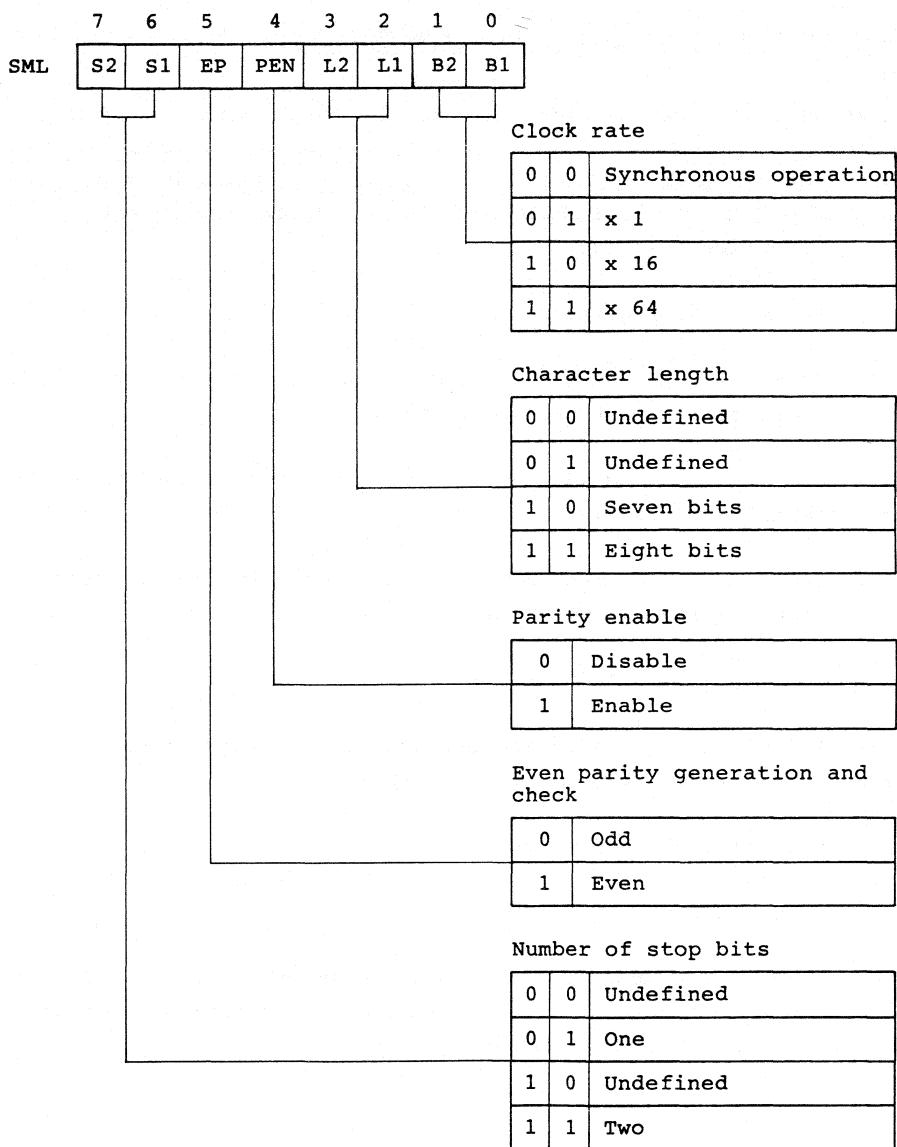
(4) EP (bit 5)

EP is used to select an odd or even parity. When the EP bit is set to 1, an even parity is selected; when 0, an odd parity is selected. The EP bit is significant when the PEN bit is set to 1.

(5) S1 and S2 (bit 6 and 7)

S1 and S2 are used to select the number of stop bits sent during the asynchronous mode.

Fig. 7-3 Serial Mode Low Register (SML) Format



When RESET is input or during the hardware STOP mode, the serial mode low register (SML) is set to 48H.

7.2.3 Serial mode register initialization

Initialize the serial mode registers in the following sequence:

- 1) Set the SMH register to the necessary operation mode with the TxE and RxE bits set to 00 (both transmission and reception disabled).
- 2) Set the SML register.
- 3) If the serial clock is TO output, set the timer mode if the timer mode is not yet specified.
- 4) Specify the control mode for the port C pins used with the serial interface.
- 5) Set the SMH register TxE or RxE bit to 1 (enable transmission or reception).

7.3 Serial Interface Operation

The uPD78C11 serial interface operates in the asynchronous mode, synchronous mode, or IO interface mode.

Each mode is explained below:

7.3.1 Asynchronous mode

The asynchronous mode is a transfer system using start and stop bits. Bit synchronization and character synchronization of data are taken by using the start bit.

To transfer data in the asynchronous mode, set transfer parameters such as the character length, clock rate, number of stop bits, odd or even parity, serial clock, and transmission and reception enable in the serial mode registers SMH and SML.

Fig. 7-4 Serial Mode Register Format during Asynchronous Mode

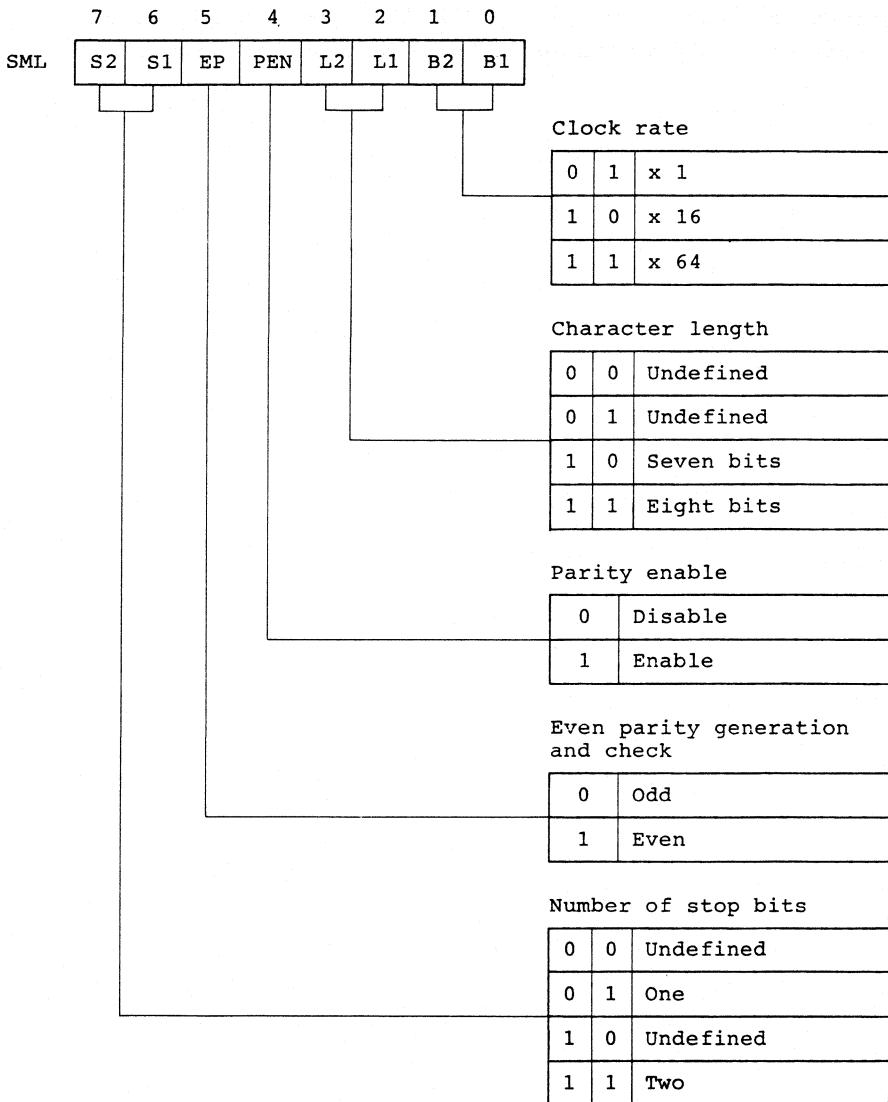
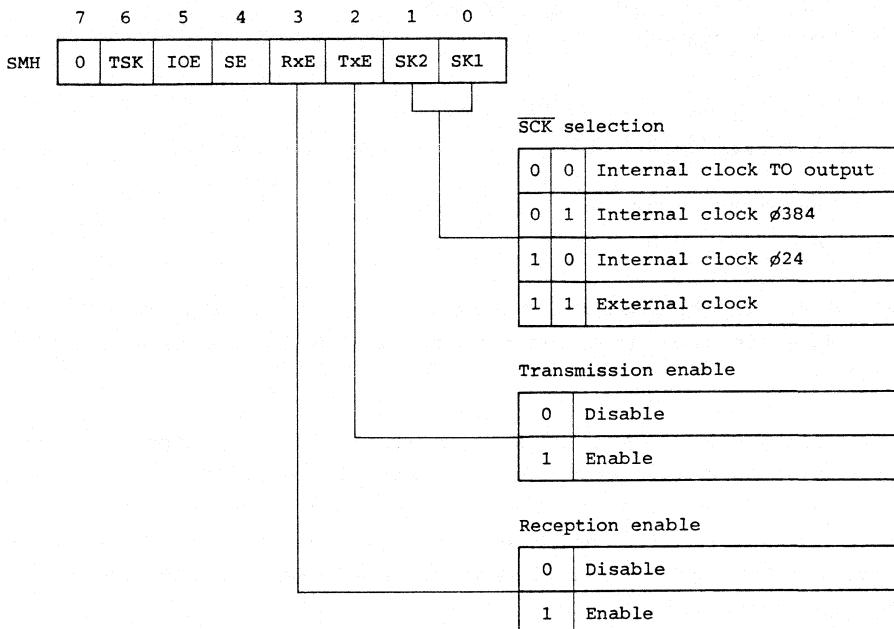


Fig. 7-4 Serial Mode Register Format during Asynchronous Mode
(Cont'd)



If internal clock is used for serial clock (SCK), the data transfer rate is determined by one of the following expressions depending on the oscillator frequency and clock rate:

When internal clock ϕ 24 is used

$$B = f_{XX}/24 \times N$$

When internal clock ϕ 384 is used

$$B = f_{XX}/384 \times N$$

When internal clock TO output is used

When timer input clock is ϕ_{12}

$$B = f_{XX}/24 \times N \times C$$

When timer input clock is ϕ_{384}

$$B = f_{XX}/786 \times N \times C$$

When TIMER F/F input is ϕ_3

$$B = f_{XX}/6 \times N$$

Where f_{XX} is the oscillator frequency, N is the clock rate 1, 16, or 64, C is the timer count value, and B is the data transfer rate.

When TIMER F/F input is ϕ_3 when internal clock TO output is used, it can be used only when the clock rate is 16 or 64.

When TIMERO is used and the clock rate is 16, the timer mode register TMM and the serial mode registers SML and SMH are set to the following values:

TMM: xxx00000B

SML: xxxx10B

SMH: 0000xx00B

(x: Set by the user)

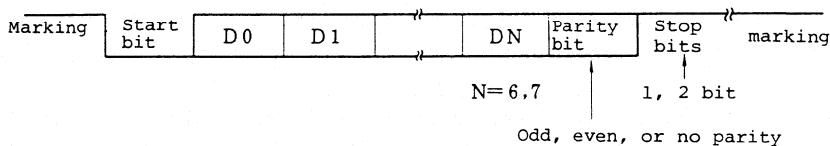
If the internal clock TO output is selected and the timer input clock is internal clock ϕ_{12} , set the timer count value as listed in Table 7-1 to transfer data at the data transfer rate of 110 to 9600 bps.

Table 7-1 Timer Setting

Oscillator frequency (MHz)	7.3728		11.0592		
Data transfer rate (bps)	N	16	64	16	64
9600	C = 2	--	C = 3	--	
4800	4	C = 1	6	--	
2400	8	2	12	C = 3	
1200	16	4	24	6	
600	32	8	48	12	
300	64	16	96	24	
150	128	32	192	48	
110	175	44	262	65	

Figure 7-5 shows the data format in the asynchronous mode.

Fig. 7-5 Asynchronous Data Format



(1) Data transmission

Transmission operation in the asynchronous mode is enabled by setting the serial mode high register (SMH) TxE bit to 1.

When data is written into the transmit buffer register by executing the MOV TXB,A instruction and sending the preceding data ends, the transmit buffer register contents are automatically transferred to the serial register. One start bit, parity bit (odd, even, or no parity), and one or two stop bits are automatically added to the data

transferred to the serial register. Then, the data is sent starting at the least significant bit (LSB) from the TxD pin. When the transmit buffer register becomes empty, a serial transmit interrupt INTST occurs.

The serial transmit interrupt is disabled by setting the interrupt mask register (MKH) MKST bit to 1.

If additional data is written into the transmit buffer register when the register is full, the preceding data is destroyed. Check that the serial transmit interrupt request flag INTFST is set to 1 and the transmit buffer register is empty before writing data into the transmit buffer register.

When the TxE bit is set to 0 or the serial register does not contain data to be transmitted, the TxD pin is placed in the mark state (1).

Send data is sent from the TxD pin on the $\overline{\text{SCK}}$ falling edge at any transfer rate of serial clock $\times 1$, $\times 1/16$, or $\times 1/64$.

Table 7-2 lists the maximum data transfer rates during transmitting at 15 MHz according to the $\overline{\text{SCK}}$ and clock rate.

Table 7-2 Maximum Data Transfer Rates

Clock rate	Internal clock		External clock	
	SCK	Data transfer rate	SCK	Data transfer rate
$\times 1$	625 kHz	625 kbps	1.25 MHz	1.25 Mbps
$\times 16$	2.5 MHz	156 kbps	2.5 MHz	156 kbps
$\times 64$		39.1 kbps		39.1 kbps

(2) Data reception

Reception operation in the asynchronous mode is enabled by setting the serial mode high register (SMH) RxE bit to 1.

The start bit is acknowledged by detecting RxD input which is low and again detecting RxD low in the 1/2 bit time. This has the effect of preventing noise in the mark state from causing malfunction. The data is received by sampling at the center of the subsequent character bits, parity bit, and stop bit or bits.

When given data is input to the serial register from the RxD pin, the data is transferred to the receive buffer register. When the receive buffer register becomes full, an interrupt request INTSR is generated. The serial receive interrupt is disabled by setting the interrupt mask register (MKH) MKSR bit to 1.

When data is received, an odd or even parity check is made when the PEN bit is set to 1. When a mismatch is found (parity error), the stop bit is low (framing error), or new data is transferred to the receive buffer register when the receive buffer register is full (overrun error), the error flag is set.

However, since the error interrupt feature is not contained, error detection must be tested by using the skip instruction SKIT or SKNIT in a program.

If RxB data is read at generation of error, overrun error is generated again at the next receive.

Table 7-3 lists the maximum data transfer rates during receiving at 15 MHz according to SCK and clock rate.

Table 7-3 Maximum Data Transfer Rates

Clock rate	Internal clock		External clock	
	SCK	Data transfer rate	SCK	Data transfer rate
x 1 (Note 2)	625 kHz	625 kbps	830 kHz 1.25 MHz	830 kbps 1.25 Mbps (Note 1)
x 16		156 kbps		156 kbps
x 64	2.5 MHz	39.1 kbps	2.5 MHz	39.1 kbps

Note 1: Two stop bits are required to receive data at the transfer rate of 830 kbps to 1.25 Mbps.

Note 2: For the clock rate x 1, signal synchronized with SCK must be input to RxD.

Caution on use: uPD78C10/78C11/78C14/78C14A/78CG14 only

If serial data is received in the asynchronous mode by using external clock for SCK, the ER flag may be set although the data is received normally.

Countermeasures:

Examine the following countermeasures:

(1) Use of software

If the ER flag is set, send a retransmit request to the transmitting party. This method is most desirable as error countermeasure.

(2) Use of internal clock for SCK

Use internal clock for SCK. In this case, do not output SCK from the PC2 pin. (MCC2 = 0, PC2 = port mode)

(3) Input of external clock to PC3/TI

Input external clock to PC3/TI and count TI by the timer. Invert TO by using the timer comparator coincidence signal and use the TO as \overline{SCK} . Do not output \overline{SCK} from the PC2 pin.
(MCC2 = 0)

Reception processing initialization program example:

```
MVI SMH, 00H      ;  $\overline{SCK}$ : TO
MVI A, xxH        ; Set SML
MOV SML, A
MVI A, xxH        ; Set TM0 (data to divide TI input)
MOV TO0, A
MVI TMM, xxxx01000B ; Timer 0: TI count. TO is inverted by
                      ; timer 0 coincidence signal.
MVI A, xxxx101xB   ; PC1: RxD input
                    ; PC2: Port mode
                    ; PC3: TI input
MOV MCC,A
:
:
ORI SMH, 08H      ; Enable           reception
```

Remarks 1: The relationship among TI input frequency f_{TI} , data transfer rate B, and clock rate N is as follows:

$$B = \frac{f_{TI}}{2 \times C \times N} \quad (C \text{ is TM0 setup value})$$

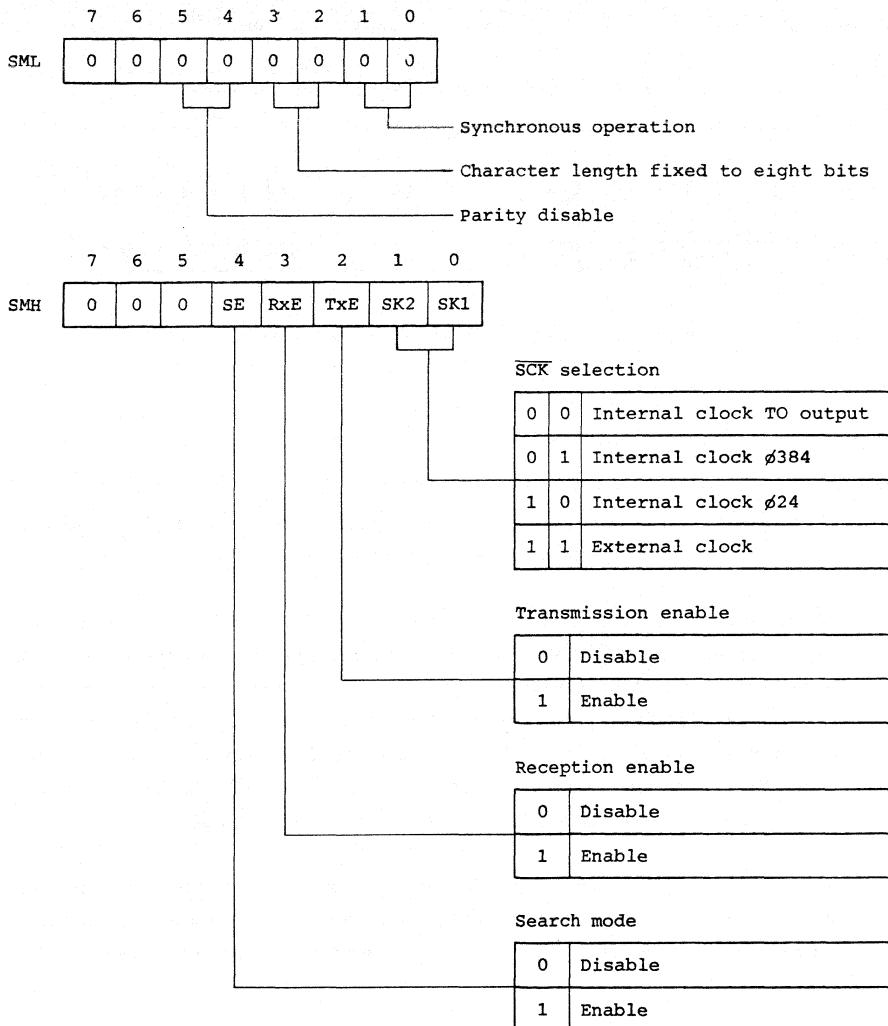
Remarks 2: The high or low width of TI input is $6/f_{xx}$ (f_{xx} is the oscillator frequency) or more. (If when $f_{xx} = 15$ MHz, the width is 0.4 us or more and duty 50%, the maximum frequency of f_{TI} becomes 1.25 MHz.)

7.3.2 Synchronous mode

In the synchronous mode, character synchronization is taken by using synchronous character and bit synchronization is taken by using serial clock.

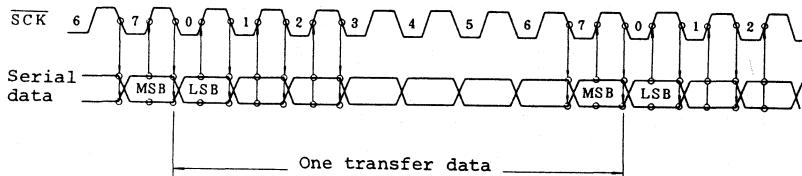
To transfer data in the synchronous mode, set the character length fixed to eight bits and no parity bit in the serial mode register, as shown in Fig. 7-6.

Fig. 7-6 Serial Mode Register Format during Synchronous Mode



In synchronous mode, send data with the character length fixed to eight bits and no parity bit is sent starting at the least significant bit (LSB) on the serial clock (\overline{SCK}) falling edge, as shown in Fig. 7-7. Receive data is input on the \overline{SCK} rising edge.

Fig. 7-7 Timing during Synchronous Mode



(1) Data transmission

Transmission operation in the synchronous mode is enabled by setting the serial mode high register (SMH) TxE bit to 1.

When data is written into the transmit buffer register by executing the MOV TXB, A instruction and the preceding data transmission terminates, the data is transferred from the transmit buffer register to the serial register and converted into serial data. The serial data is sent from the TxD pin in synchronization with the \overline{SCK} falling edge. The serial data is sent at the same rate as \overline{SCK} .

When the data is transferred from the transmit buffer register to the serial register and the transmit buffer register becomes empty, an interrupt request INTST is generated.

The serial transmission interrupt is disabled by setting the interrupt mask register MKST bit to 1.

When the TxE bit is set to 0 or the serial register does not contain data to be sent, the TxD pin is placed in the mark state (1).

However, when external clock is used, the TxD pin is placed in the mark state after a low pulse is output.

The maximum data transfer rate during transmitting is 625 kbps when internal clock is used for \overline{SCK} ; 1.25 Mbps when external clock is used for \overline{SCK} . (at 15 MHz)

(2) Data reception

Reception operation in the asynchronous mode is enabled by setting the serial mode high register RxE bit to 1. Receive data is input on the \overline{SCK} rising edge.

The synchronous mode provides two types of reception operation, either of which can be selected by using the serial mode high register SE bit.

When the SE bit is set to 1, the search mode is selected. Each time one bit is received from the RxD pin, the serial register contents are transferred to the receive buffer register and a serial reception interrupt INTSR occurs. Since the uPD78C11 does not contain a synchronous character detector of hardware, a synchronous character must be detected by software. If the synchronous character is detected and reception synchronization is taken, the SE bit is reset to 0.

When the SE bit is reset to 0, the character reception mode is selected. Each time 8-bit data is received, the serial register contents are transferred to the receive buffer register and a serial reception interrupt INTSR occurs.

The serial reception interrupt is disabled by setting the interrupt mask register (MKH) MKSR bit to 1.

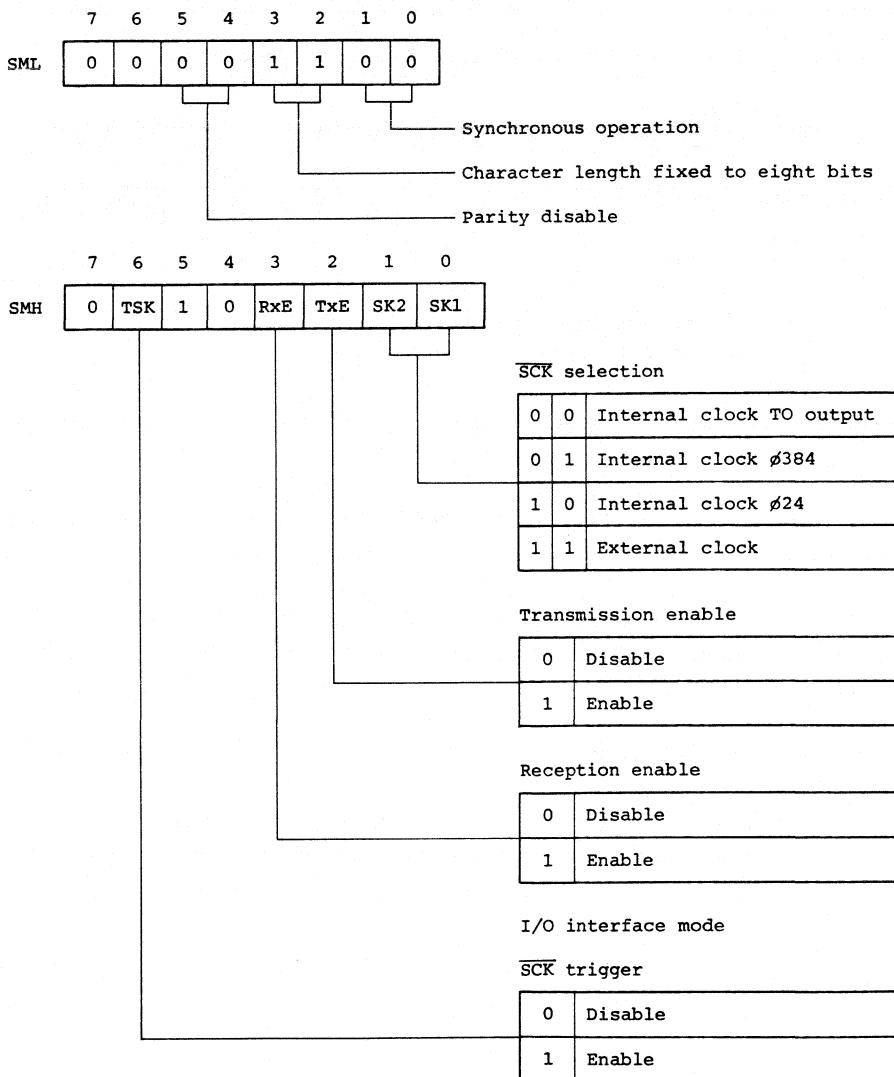
The maximum data transfer rate during receiving is 625 kbps when internal clock is used for SCK; 1.25 Mbps when external clock is used for SCK. (at 15 MHz)

7.3.3 I/O interface mode

The I/O interface mode is the same as the serial data transfer mode on the uPD7801, uPD78C06A, etc., and synchronization is taken by using the controlled serial clocks.

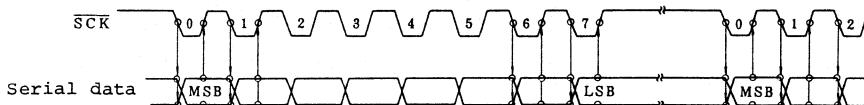
To transfer data in the I/O interface mode, set the character length fixed to eight bits and no parity bit in the serial mode registers as shown in Fig. 7-8.

Fig. 7-8 Serial Mode Register Format during I/O Interface Mode



In the I/O interface mode, send data (TxD) is sent starting at the most significant bit (MSB) on the serial clock (SCK) falling edge, and receive data is input on the SCK rising edge.

Fig. 7-9 Timing during I/O Interface Mode



In the I/O interface mode, character synchronization is taken by using controlled SCK (eight serial clock).

External or internal clock is used for SCK and can be selected in the serial mode high register.

When internal clock is used for SCK, eight controlled clocks for one data piece are output from the SCK pin.

When external clock is used for SCK, be sure to supply eight clocks for each 8-bit data transfer to SCK from the clock signal supply source.

(1) Data transmission

Transmission operation in the I/O interface mode is enabled by setting the serial mode high register TxE bit to 1.

When data is written into the transmit buffer register by executing the NOV TXB,A instruction and the preceding data transmission ends, the transmit buffer register contents are transferred to the serial register. If SCK is internal clock, when the data is transferred to the serial register, automatically eight controlled SCK clocks for one data piece are generated, and the send data is sent starting at the most significant bit (MSB) on the SCK falling edge.

If SCK is external clock, the send data is sent starting at the most significant bit (MSB) on the falling edge of the controlled SCK input to the SCK pin.

In the I/O interface mode, eight controlled SCK serial clocks are used for synchronization. Set SCK high not during data transfer.

When the transmit buffer register becomes high, a serial transmission interrupt INTST occurs.

The serial transmission interrupt is disabled by setting the interrupt mask register (MKH) MKST bit to 1.

The maximum data transfer rate during transmitting is 625 kbps when internal clock is used for SCK; 1.25 Mbps when external clock is used for SCK. (at 15 MHz)

(2) Data reception

Reception operation in the I/O interface mode is enabled by setting the serial mode high register (SMH) RxE bit to 1. Receive data (RxD) is input to serial register on the SCK rising edge.

When the serial register receives 8-bit data, the data is transferred from the serial register to the receive buffer register and a serial reception interrupt INTSR is generated.

If SCK is external clock, data transferred in synchronization with SCK is input to the serial register on the SCK rising edge.

If SCK is internal clock, SCK must be started by setting the serial mode high register (SMH) TSK bit to 1.

The serial reception interrupt is disabled by setting the interrupt mask register (MKH) MKSR bit to 1.

The maximum data transfer rate during receiving is 625 kbps when internal clock is used for SCK; 660 kbps when external clock is used for SCK. (at 15 MHz) The high width of the eighth SCK must be six states or more.

Caution 1: When the external clock under seventh is input (at sending)

When the sending and receiving are performed by I/O interface mode using external clock, the external clock under seventh is accidentally input, and data are worked down. The correction method is explained as follows.

- (1) TxE bit → reset to 0: Sending disable
- (2) Modify MCC register
MCC (2) → reset to 0: SCK pin to port mode
MCC (0) → reset to 0: TxD pin to port mode
- (3) Modify MC register
MC (2) → set to 1: To input port
MC (0) → reset to 0: Output high level pin
PC (0) → set to 1: To output port

- (4) Modify SMH
 $\left. \begin{array}{l} SK1 = 0 \\ SK2 = 1 \end{array} \right\}$ or $\left. \begin{array}{l} SK1 = 1 \\ SK2 = 0 \end{array} \right\}$: To internal clock mode
- RxE bit → reset to 0: Receiving disable
- (5) TxE bit → set to 1: Output sending enable and serial register remaining data
- (6) SK1, SK2, bit → set to 1: To external clock
- (7) The initial setting of MCC and MC

Caution 2: The external clock under seventh is input (at receiving)

- (1) RxE bit → reset to 0: Receiving disable
- (2) MCC (2) → reset to 0: \overline{SCK} pin to port mode
- (3) MC (2): Set to 1: To input port
- (4) $\left. \begin{array}{l} SK1 = 0 \\ SK2 = 1 \end{array} \right\}$ or $\left. \begin{array}{l} SK1 = 1 \\ SK2 = 0 \end{array} \right\}$: To internal clock mode
- (5) \overline{SCK} trigger bit → set to 1: Internal clock start, remaining clock count
- (6) SK1, SK2 bit → set to 1: To external clock
- (7) The initial setting of MCC, MC and ReE

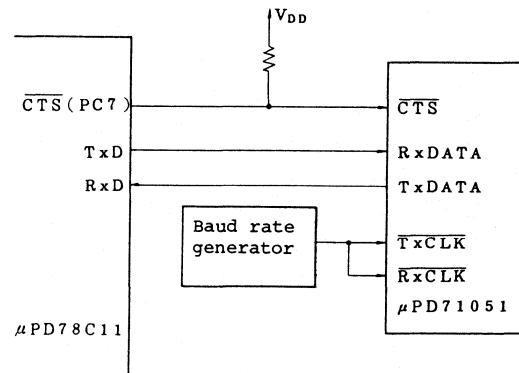
7.3.4 Serial interface program example

An example of data transfer between the uPD78C11 and uPD78051 in the asynchronous mode is given as a serial interface programming example.

Set the following: uPD78C11 oscillator frequency = 11.0592 MHz, serial clock = internal clock (T0 output), data transfer rate = 110 bps, clock rate = 16, character length = 8 bits, number of stop bits = 2, and even parity enable.

Fig. 7-10 shows a system configuration example. Three lines are required to transfer serial data; the two lines are serial data input/output lines TxD and RxD and the one line is control line CTS (Clear to Send). In this example, PC7 is used as the control line CTS; it is used for the μ PD78C11 to receive data. Since PC7 becomes input port until mode setting after reset, it is pulled high with a pull-up resistor, and 1 is written into the PC7 output latch before PC7 is set to output port.

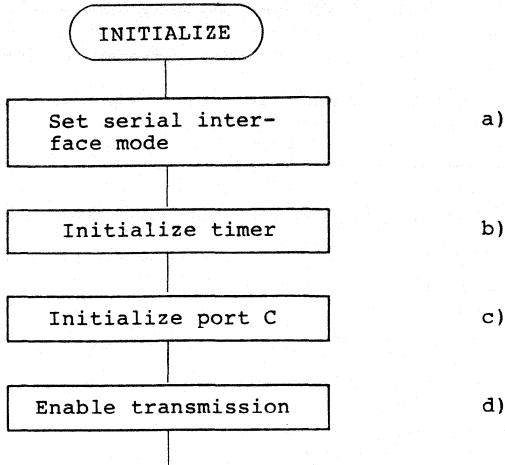
Fig. 7-10 Serial Data Transfer System Configuration Example



(1) Initialization

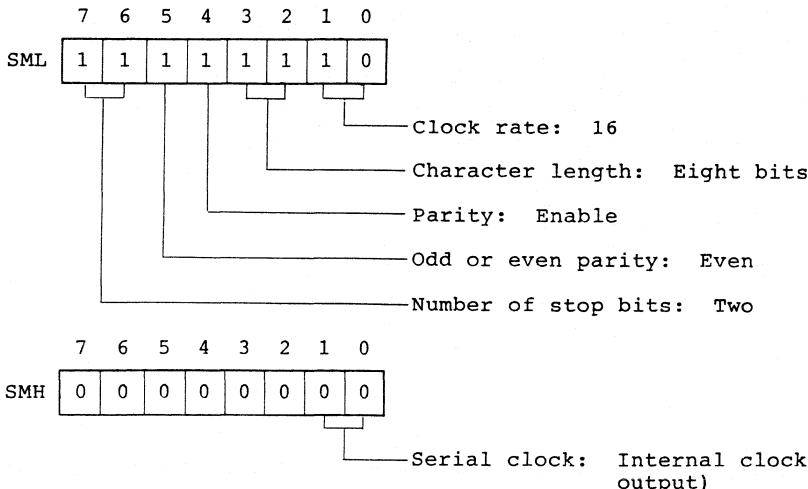
The serial mode register, timer, port C, etc., must previously be initialized so that serial data can be transferred.

The operation flow is shown below:



- a) The serial data transfer parameters of the character length, clock rate, number of stop bits, odd or even parity, and serial clock are set in the serial mode register.

Fig. 7-11 Serial Mode Register Setting



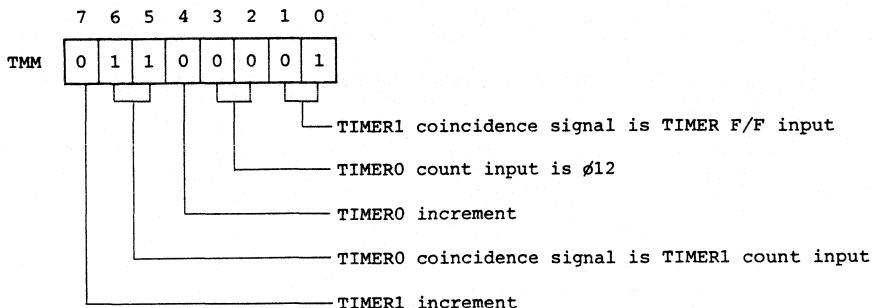
- b) Since timer TO output is used as serial clock (\overline{SCK}), the timer count value and timer operation mode are set. In the example, oscillator frequency = 11.0592 MHz, data transfer rate = 110 bps, and clock rate = 16. Thus, the count value becomes 262 from Table 7-1 or the following expression:

$$C = \frac{f_{XX}}{24 \times N \times B}$$

f_{XX} : Oscillator frequency B: Data transfer rate
 N: Clock rate C: Count value

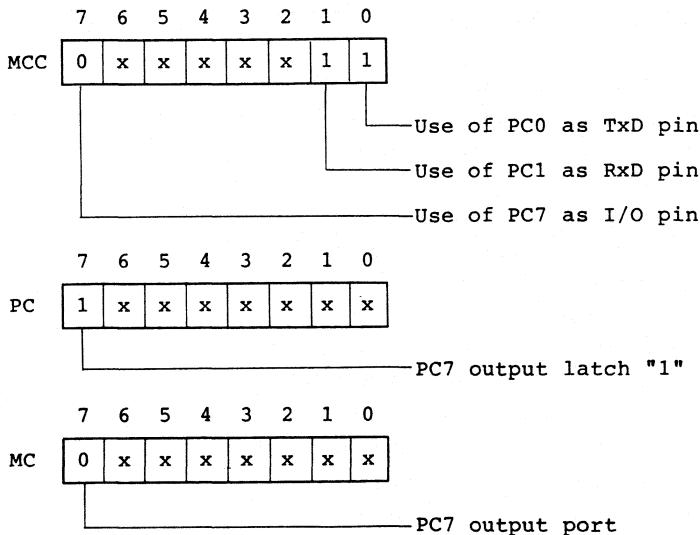
Since the count value is 255 (FFH) or more, TIMER0 and TIMER1 are cascaded for use and 131 (83H) and 2 (02H) are set in the timer registers TM0 and TM1 respectively. To cascade TIMER0 and TIMER1 for use, the timer mode register is set as shown in Fig. 7-12.

Fig. 7-12 Timer Mode Register Setting



- c) Port C is set to the following: PC0 is used as the TxD pin, PC1 is used as the RxD pin, PC7 is used as output port, and high is output from PC7.

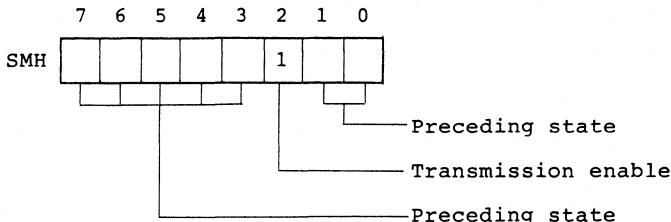
Fig. 7-13 Port C Setting



When uPD78C11 SCK is output to the external or SCK is input from the external, the PC2 pin can be used as the SCK input/output pin by setting MCC.

- d) The serial mode high register (SMH) TxE bit is set to 1 to enable transmission.

Fig. 7-14 Serial Mode High Register Setting



The initialization program is shown below:

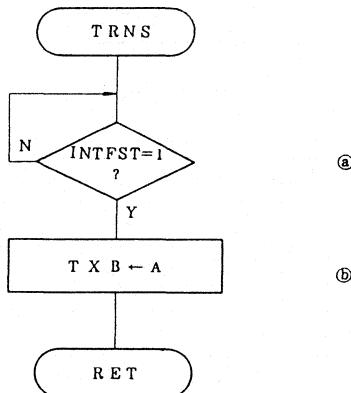
;*****SERIAL INTERFACE INITIALIZATION*****

```
SINIT: MVI SMH, 00H ; Internal serial clock (TO)
      MVI A, 0FEH ; x 16, even parity, 8-bit character, 2 stop bit
      MOV SML, A ; Set serial mode
      MVI A, 83H ;
      MOV TM0, A ; Set timer register
      MVI A, 02H ; Baud rate = 110 bps
      MOV TM1, A ;
      MVI TMM, 61H ; Set timer mode & start
      MVI A, 07H ; Set Port C mode control
      MOV MCC, A ; TxD, RxD, SCK available
      ORI PC, 80H ; PC7 output latch = 1
      MVI A, 00H ; Initialize Port C
      MOV MC, A ; Port C output mode
      ORI SMH, 04H ; Transmit enable
```

(a) (b) (c) (d)

(2) uPD78C11 data reception

A subroutine example which transmits one byte of the accumulator A contents as serial data is given. In this example, interrupt INTST operation is not used and serial data is transmitted by testing the interrupt request flag INTFST. The operation flow is shown below:



- The interrupt request flag INTFST is tested to judge whether or not data can be written into the transmit buffer TXB.
- The accumulator contents are transferred to the transmit buffer.

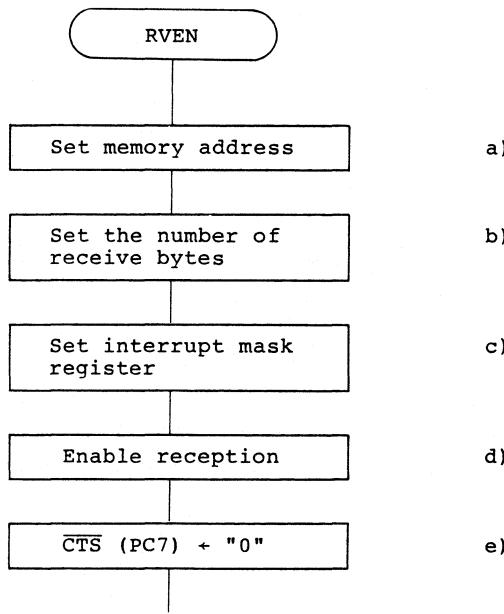
The data transmission routine is shown below: (uPD71051 must be ready to receive data to transmit data from the uPD78C11.)

;*****TRANSMIT SERVICE*****

```
TRMS : SKIT FST      ; Test FST, Skip if FST = 1 } a
       JR    TRNS      ; Wait until FST = 1
       MOV   TXB, A ; Output transmit data } b
       RET            ; Return
```

(3) uPD78C11 data reception

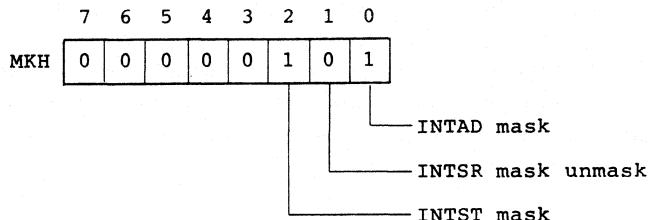
Data is received by using hardware interrupt INTSR. Thus, the receive data storing memory address, the number of receive bytes, the interrupt mask register, etc., must previously be initialized. The operation flow is shown below:



- a) The receive data storing memory address is set in the register pair HL. In this example, the receive data is stored starting at address 2000H.
- b) The number of receive data pieces is set in the B register. Here, 16 (0FH) data pieces are received.

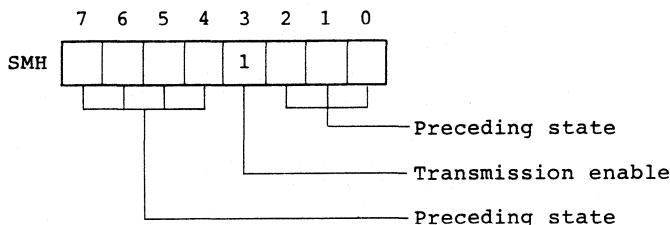
- c) The interrupt mask register (MKH) MKSR bit is reset to 0 to unmask internal interrupt INTSR mask. Assume that the interrupt mask flag of INTST assigned the same priority level as INTSR is set to 1 to mask INTST.

Fig. 7-15 Interrupt Mask Register MKH Setting



- d) The serial mode high register (SMH) RxE bit is set to 1 to enable reception.

Fig. 7-16 Serial Mode High Register SMH Setting

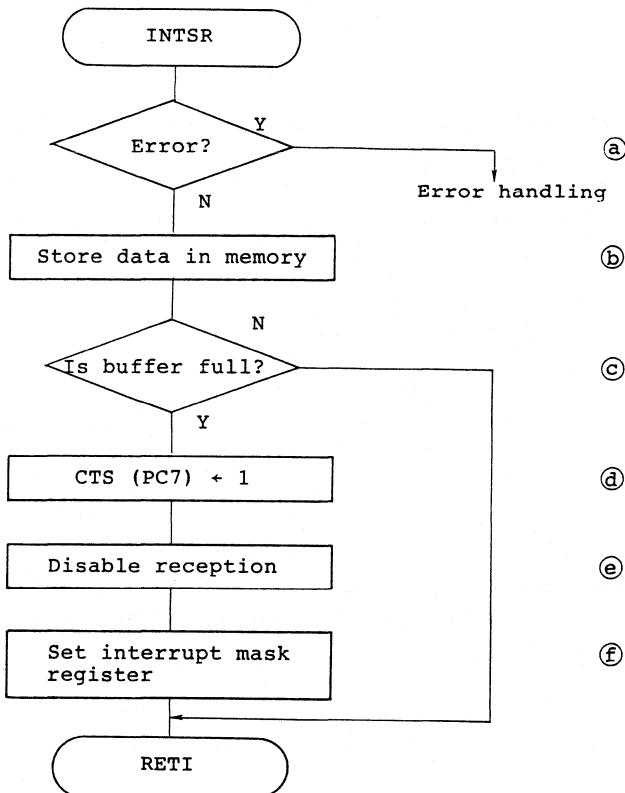


- e) PC7 output is set to 0 and \overline{CTS} is activated. The initialization program required during receiving is shown below:

;*****RECEIVE ENABLE*****

```
RVEN : LXI H, 2000H ; Set data pointer (DP = 2000H)      a)
      MVI C, 0FH    ; Set data counter (DC = 0FH)        b)
      EXX
      ANI MKH, 05H ; INTST enable                      c)
      ORI SMH, 08H ; Receive enable                    d)
      ANI PC, 7FH  ; CTS = 0                           e)
```

Each time given data is received after the settings are made, an internal interrupt INTSR occurs. The interrupt service routine operation flow is shown below:



- a) A check is made to see if the receive data contains any error. If an error is detected, control is transferred to the error handling routine.

Caution: If RxB data is read at generation of error, overrun error is generated again at the next receive.

- b) Receive data is stored in memory.
- c) A check is made to see if the data buffer becomes full. If the buffer is not full, the program flow returns to the main routine.
- d) PC7 output is set to 1, $\overline{\text{CTS}}$ is deactivated, and uPD71055 data transmission is stopped.
- e) The serial mode high register (SMH) RxE bit is reset to 0 to stop reception operation.
- f) The interrupt mask register (MKH) MKSR bit is set to 1 to disable internal interrupt INTSR. The interrupt service routine is shown below: (The interrupt service routine must be stored starting at the INTSR interrupt address 0028H or a JMP RECV instruction must be stored in 0028H.)

;*****RECEIVE SERVICE*****

```
RECV: EXA          ; Save accumulator
      EXX          ; Save register
      SKNIT ER      ; Test ERflag,skip if ER=0      } ⑧
      JMP  ERROR    ; Jump ERROR routine           }
      MOV  A ,RXB   ; Input received data        } ⑨
      STAX H+       ; Store received data to memory
      DCR  C        ; Skip if buffer full      } ⑩
      JR   RECO     ;
      ORI  PC ,80H   ; CTS←1                  } ⑪
      ANI  SMH,0F7H  ; Receive disable            } ⑫
      ORI  MKH,02H   ; INTSR disable             } ⑬
RECO : EXX          ; Recover register
      EXA          ; Recover accumulator
      EI           ; Enable interrupt
      RETI         ; Return
```

CHAPTER 8 ANALOG-TO-DIGITAL CONVERTER FUNCTION

The uPD78C11 contains a high-precision 8-bit A/D converter which has eight analog inputs and uses successive approximation and four conversion result registers CR0 to CR3 which retain the conversion result. Analog input is selected in the scan mode or select mode to minimize software overhead.

8.1 Analog/Digital Converter Configuration

The A/D converter consists of input circuit, series resistor string, voltage comparator, successive approximation logic, and CR0 to CR3 registers. (See Fig. 8-1.)

Eight analog inputs are multiplexed on the chip and selected as specified in the A/D channel mode register (ANM).

The selected analog input is sampled by the sampling and hold circuit and input to the voltage comparator. The voltage comparator amplifies the difference between the analog input and series resistor string voltage tap.

The series resistor string is placed between the A/D reference voltage pin (V_{AREF}) and A/D ground (AV_{SS}) and consists of 256 equivalent resistors to make 256 equivalent voltage steps between the two pins.

Series resistor string voltage tap is selected by the tap decoder. This decoder is driven by the 8-bit successive approximation register (SAR).

SAR is set one bit at a time starting at the most significant bit (MSB) so that the series resistor string voltage tap value matches the analog input voltage value. When conversion starts, the SAR MSB is set to 1 and the series resistor string voltage tap is set to $1/2 V_{AREF}$. It is compared with analog input. If

the analog input is greater than $1/2 V_{AREF}$, the SAR MSB remains on (1); if less than $1/2 V_{AREF}$, the MSB is reset, then comparison is made on the second most significant bit. On the bit (bit 7), series resistor string voltage tap is set to $3/4$ or $1/4 V_{AREF}$. It is compared with analog input. Such comparison is continued to the least significant bit (LSB) of SAR. (Binary search method)

At the 8-bit comparison termination, SAR retains the valid digital results which are latched in the CR0 to CR3 registers in sequence.

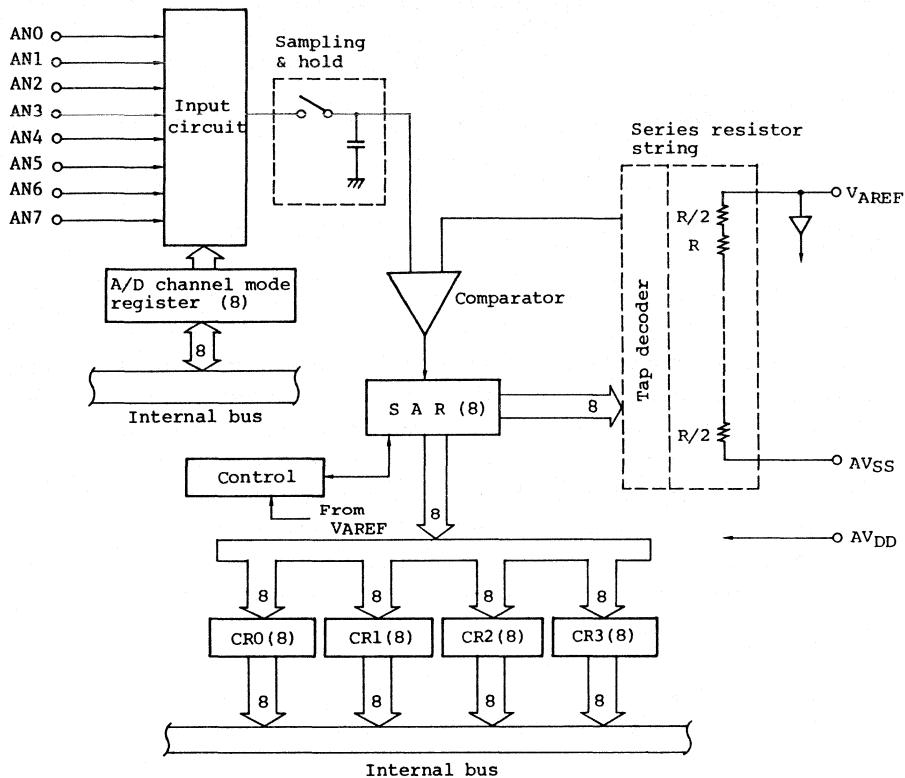
When the A/D conversion results are latched in all the CR0 to CR3 registers, an A/D conversion termination interrupt INTAD is generated.

The A/D converter contains independent power supply pins AV_{DD} and AV_{SS} . Adverse affection caused by power fluctuation or system noise can be minimized by supplying regulated power to the pins.

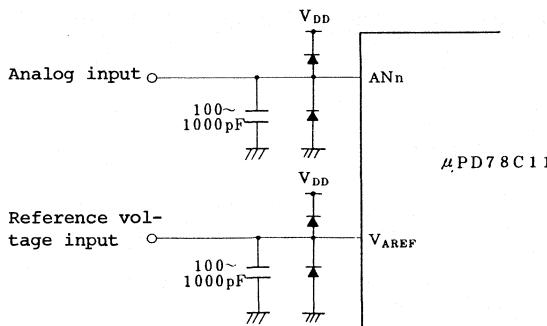
The A/D converter enables the conversion voltage range to be changed by changing V_{AREF} .

A/D converter operation can be controlled by inputting a low pulse to V_{AREF} .

Fig. 8-1 A/D Converter Block Diagram



Caution: To prevent noise from causing malfunction, connect a capacitor to the analog input pins AN7-AN0 and reference voltage input pin V_{AREF}. Do not apply voltage outside the AV_{SS}-V_{AREF} range to the unused pins or edge detection function using pins of AN7-AN0, or the conversion precision becomes worse. In that case, it is effective for noise elimination to use the clamp by V_F small-type diode, such as Schottky diode. Make the analog signal input source and reference voltage input source impedances as low as possible.



8.2 A/D Channel Mode Register (ANn)

The A/D channel mode register is an 8-bit register to control A/D converter operation. Fig. 8-2 shows the A/D channel mode register format. Bit 0 (MS) is used to control the operation mode. Bits 1-3 (ANI0-ANI2) are used to select analog input to be converted into the digital form. Bit 4 (FR) is used to maintain the appropriate conversion speed. One conversion speed can be calculated from the following expression according to the oscillator frequency and how the FR bit is set: (It is set as listed in Table 8-1.)

$$f_{XTAL} = 0: \text{Conversion speed} = 48 \times 12/f_{XTAL} \text{ (us)}$$

$$f_{XTAL} = 1: \text{Conversion speed} = 36 \times 12/f_{XTAL} \text{ (us)}$$

f_{XTAL} : Oscillator frequency (MHz)

Table 8-1 Conversion Speed Setting

Oscillator frequency	12 MHz	11 MHz	10 MHz	9 MHz	8 MHz	7 MHz	6 MHz
FR bit	0	0	0	1	1	1	1
Conversion speed	48 us	52.4 us	57.6 us	48 us	54 us	61.7 us	72 us

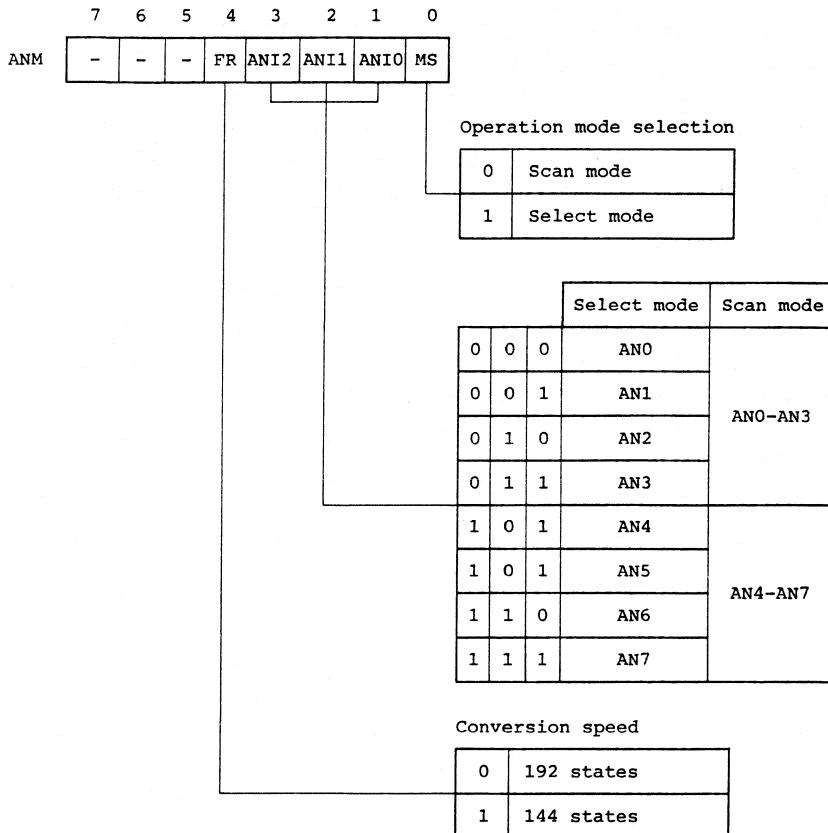
The current conversion mode can be decided by reading the A/D channel mode register contents.

When RESET is input or during the hardware STOP mode, the A/D channel mode register is reset to 00H.

When the ANM is written, the A/D converter is initialized, stops the current A/D conversion being made, and restarts the A/D conversion from the beginning according to the ANM write contents.

If the ANM is written after the INTFAD flag is cleared, A/D conversion is started according to the ANM write contents. Thus, when the INTAD flag is newly set, the results after the change are stored in the CR registers (CR0-CR3).

Fig. 8-2 A/D Channel Mode Register Format



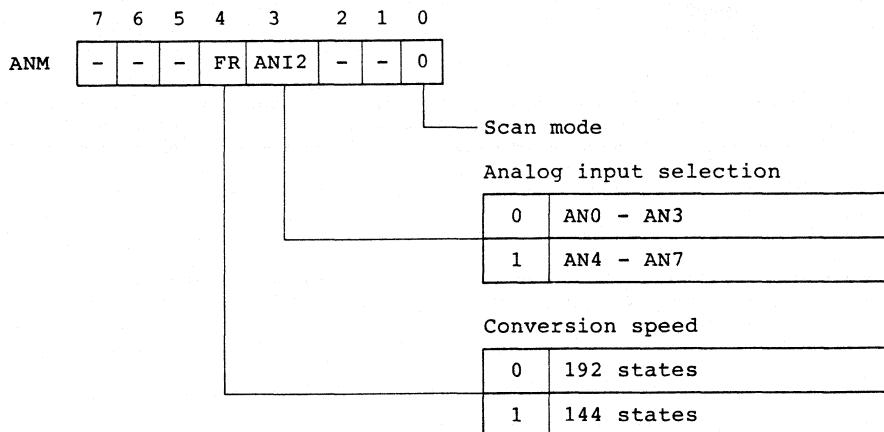
8.3 Analog-to-Digital Converter Operation

The A/D converter mode, scan or select mode can be selected by setting the A/D channel mode register (ANM) MS bit.

8.3.1 Scan mode

In the scan mode, analog inputs of AN0 to AN3 (ANI2 = 0) or AN4 to AN7 (ANI2 = 1) can be selected by using the A/D channel mode register (ANM) as shown in Fig. 8-3.

Fig. 8-3 A/D Channel Mode Register when Scan Mode is Selected



When the A/D channel mode register ANI2 bit is set to 0, analog input is selected in the order of AN0 → AN1 → AN2 → AN3, and the A/D conversion value of each input is stored in the order of CR0 → CR1 → CR2 → CR3.

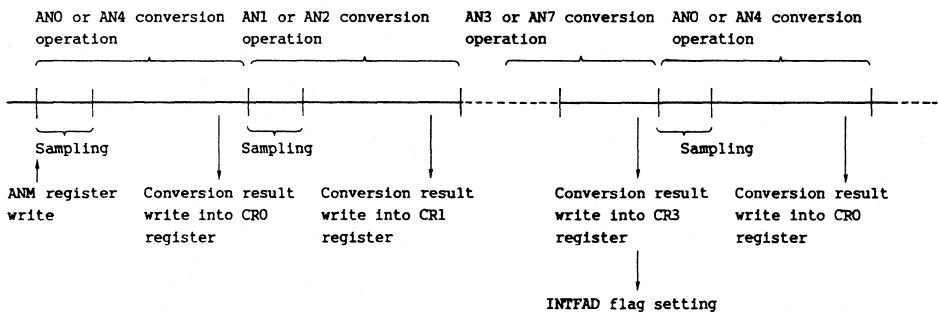
When the conversion values have been stored in the four CR registers CR0-CR3, an internal interrupt INTAD is generated.

The A/D converter again starts A/D conversion at AN0 or AN4 and stores the conversion value starting at CR0 regardless of whether or not the interrupt request is acknowledged. The A/D converter continues this operation until the A/D channel mode register is rewritten.

The scan mode enables A/D conversion of four analog inputs by minimum software.

An internal interrupt is disabled by setting the interrupt mask register (MKH) MKAD bit to 1.

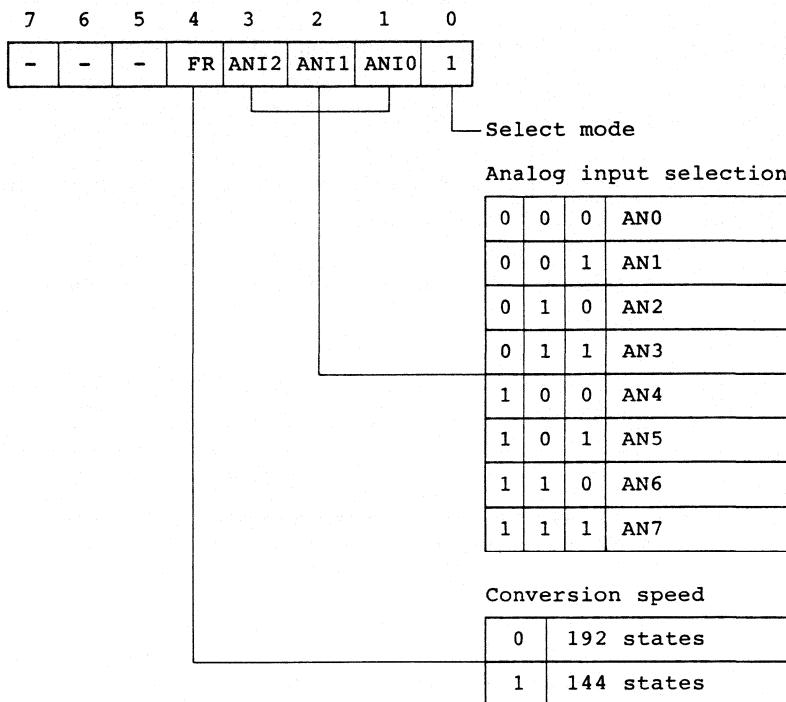
Fig. 8-4 Outline of A/D Converter Operation Timing during Scan Mode



8.3.2 Select mode

In the select mode, one of analog inputs AN0-AN7 is selected by setting the A/D channel mode register (ANM) as shown in Fig. 8-5.

Fig. 8-5 A/D Channel Mode Register when Select Mode is Selected



A/D conversion of the analog input selected in the A/D channel mode register is made and the conversion value is stored in the CR registers in the order of CR0 → CR1 → CR2 → CR3. When the conversion values have been stored in the four CR registers CR0-CR3, an internal interrupt INTAD is generated.

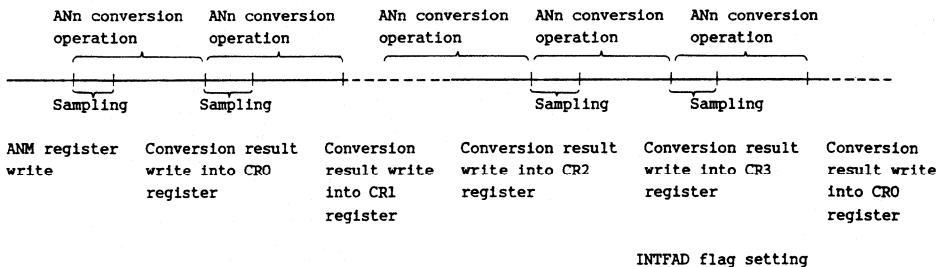
The A/D converter continues A/D conversion and stores the conversion values in the CR registers in the order of CR0 → CR1 → CR2 → CR3 regardless of whether or not the interrupt request is acknowledged. Thus, the most recent conversion value is always stored in the CR register.

The A/D converter repeats this operation until the A/D channel mode register contents are changed.

The select mode provides the most recent conversion value of the selected analog input and is useful for purposes such as conversion value average and noise input prevention.

An internal interrupt is disabled by setting the interrupt mask register (MKH) MKAD bit to 1.

Fig. 8-6 Outline of A/D Converter Operation Timing during Select Mode



8.3.3 A/D converter operation control

A/D converter conversion operation can be stopped by controlling the V_{AREF} input voltage. When voltage of V_{IH1} or more is input to the V_{AREF} pin, the A/D converter starts conversion operation and the conversion results are guaranteed at $V_{AREF} = 3.4 \text{ V}-AV_{DD}$. When the input voltage to the V_{AREF} pin is dropped to V_{IL1} or less during the conversion operation, the A/D converter conversion is stopped. The CR0-CR3 contents at the time become undefined.

Although the V_{AREF} input voltage is changed to control A/D converter stop, the A/D channel mode register(ANM) is not affected. Thus, if the V_{AREF} input voltage is risen to 3.4 V or

more to restore the A/D converter from the stop to operation state, the A/D converter restarts the A/D conversion operation storing the conversion value in CR0 in the mode immediately before the A/D converter enters the stop mode.

Although the V_{AREF} input voltage level is changed, the AN4-AN7 input edge detection function is not affected.

Caution: When V_{AREF} is set low, AN0-AN7 inputs must be placed within the AV_{SS} - AV_{DD} range.

8.3.4 Analog-to-digital converter program example

A program which stores AN0-AN7 pin A/D conversion values in the memory area of 4000H-403FH shown in Fig. 8-7 is explained as an analog-to-digital converter program example.

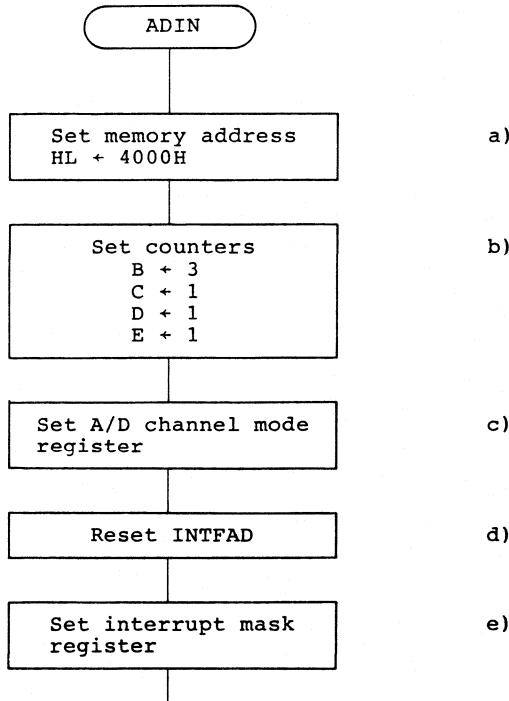
Fig. 8-7 Memory Map

	8	9	A	B	C	D	E	F
	0	1	2	3	4	5	6	7
4000H								
4008H								
4010H								
4018H								
4020H								
4028H								
4030H								
4038H								

← AN0
← AN1
← AN2
← AN3
← AN4
← AN5
← AN6
← AN7

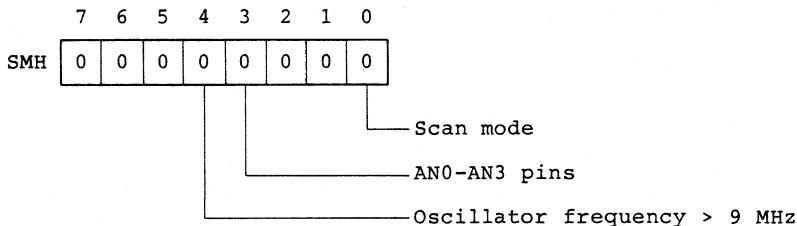
In the programming example, the A/D converter is placed in the scan mode. First, AN0-AN3 pin A/D conversion is made four times and the conversion values of AN0 to AN3 are stored in 4000H-4003H, 4008H-400BH, 4010H-4013H, and 4018H-401BH respectively. Next, AN4-AN7 pin A/D conversion is made four times and the conversion values of AN4-AN7 are stored in 4020H-4023H, 4028H-402BH, 4030H-4033H, and 4038H-403BH respectively. Next, again AN0-AN3 pin A/D conversion is made and the conversion values of AN0 to AN3 are stored in 4004H-4007H, 400CH-400FH, 4014H-4017H, and 401CH-401FH respectively. Last, AN4-AN7 A/D conversion is made and the conversion values of AN4 to AN7 are stored in 4024H-4027H, 402CH-402FH, 4034H-4037G, and 403CH-403FH respectively. An example of the program which repeats this operation sequence is given.

First, initialization operation flow is shown below:



- a) The A/D conversion value storing memory address is set in the HL register pair. Here, the conversion value is stored in memory starting at 4000H.
- b) The general purpose registers B, C, D, and E are used as counters to store the A/D conversion values in the specified memory areas. The B register is used to check whether or not AN0-AN3 or AN4-AN7 A/D conversion is made four times. Thus, 03H is set in the B register. The C, D, and E registers are used to store the conversion values in the specified memory areas: 01H is set in each of the registers.
- c) The scan mode is selected in the A/D channel mode register and input pins AN0-AN3 are selected.

Fig. 8-8 A/D Channel Mode Register Setting



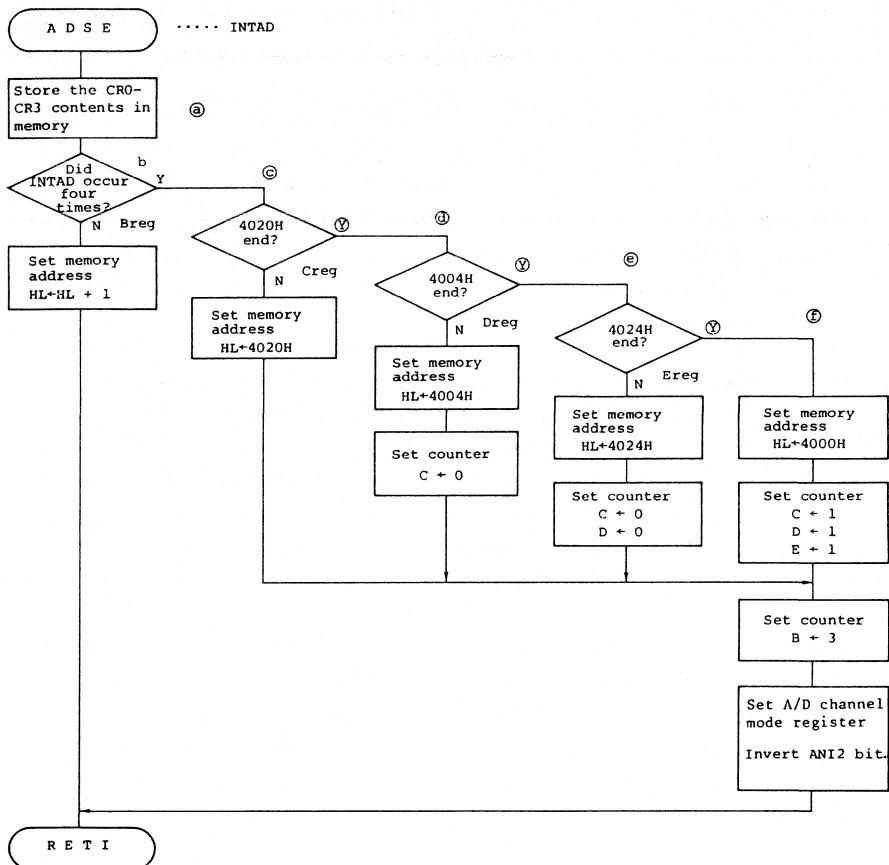
- d) The A/D channel mode register is cleared to 00H when reset. AN0-AN3 pin A/D conversion is made in the scan mode. The conversion values may be stored in the CR0-CR3 registers and the interrupt request flag INTFAD may be set to 1. Thus, the interrupt request flag is reset to 0 by executing a skip instruction before internal interrupt INTAD is unmasked by setting the interrupt mask register (MKH) MKAD bit to 0.
- e) The interrupt mask register (MKH) MKAD bit is reset to 0 and internal interrupt INTAD is unmasked.

The A/D converter initialization routine is shown below:

;*****A/D CONVERTER INITIALIZATION*****

```
ADIN : LXI H, 4000H ; Set data pointer a)
      LXI B, 0301H ; Set counter   }
      LXI D, 0101H ; Set counter   } b)
      EXX           ; Exchange register set
      MVI ANM, 00H ;                   c)
      SKIT FAD     ; Reset INTRAD   }
      NOP          ;                   d)
      ANI MKH, 0FEH; INTAD enable
```

The interrupt (INTAD) service routine stores the A/D conversion values in CR0-CR3 in given memory areas. Operation flow is shown below:



- a) The CR0-CR3 contents of the AN0-AN3 or AN4-AN7 pin A/D conversion values are stored in given memory area.
- b) A check is made to see if internal interrupt INTAD occurred four times. If it occurred less than four times. If it occurred less than four times, the HL register pair is incremented by one and a return is made. If it occurred four times, a jump is made to c). The B register is a counter to check whether or not an internal interrupt occurred four times.
- c) Since the A/D conversion values have been stored in the memory block starting at 4000H (4000H-4003H, 4008H-400BH, 4010H-4013H, 4018H-401BH), the top address of the next memory block, 4020H is set in the HL register pair, and 03H is set in the B register. To change the input pins for A/D conversion, the A/D channel mode register ANI2 bit is inverted and a return is made.

When the A/D conversion values have been stored in the memory block starting at 4020H (4020H-4023H, 4028H-402BH, 4030H-4033H, 4028H-402BH), a jump is made to c). The C register is a counter to check whether or not the A/D conversion values are stored in the memory block starting at 4020H.

- d) Since the A/D conversion values have been stored in the memory block starting at 4020H, the top address of the next memory block, 4004H is set in the HL register pair, 03H is set in the B register, and 00H is set in the C register. To change the input pins for A/D conversion, the A/D channel mode register ANI2 bit is inverted and a return is made.

When the A/D conversion values have been stored in the memory block starting at 4004H (4004H-4007H, 400CH-400FH, 4014H-4017H, 401CH-401CH), a jump is made to e). The D register is a counter to check whether or not the A/D conversion values are stored in the memory block starting at 4004H.

- e) Since the A/D conversion values have been stored in the memory block starting at 4004H, the top address of the next memory block, 4024H is set in the HL register pair, 03H is set in the B register, and 00H is set in the C and D registers. To change the input pins for A/D conversion, the A/D channel mode register ANI2 bit is inverted and return is made.

When the A/D conversion values have been stored in the memory block starting at 4024H (4024H-4027H, 402CH-402FH, 4034H-4037H, and 403CH-403FH), a jump is made to f . The E register is a counter to check whether or not the A/D conversion values are stored in the memory block starting at 4024H.

- f) The A/D conversion values are stored in the memory block starting at 4024H. The A/D conversion values are stored in all of 4000H to 403FH. Thus, initialization is made to again store the A/D conversion values in the memory block starting at 4000H.

The interrupt service routine is shown below: (JMP ADSE instruction must be stored in the INTAD interrupt address 0020H.)

;*****A/D CONVERTER SERVICE*****

```
ADSE : EXA          ; Save accumulator
       EXX          ; Save register
       MOV A, CR0    ; Store A/D conversion data in memory
       STAX H
       MOV A, CR1    ; Store A/D conversion data in memory
       STAX H+8H
       MOV A, CR2    ; Store A/D conversion data in memory
       STAX H+10H
       MOV A, CR3    ; Store A/D conversion data in memory
       STAX H+18H
       DCR B        ; Decrement counter, skip if borrow
       JR ARIN      }
       DCR C        } a
       JR ARST0     }
       MOV A, D      }
       DCR A        } b
       JR ARST1     }
       MOV A, E      }
       DCR A        } c
       JR ARST2     }
       LXI H, 4000H  ; Set data pointer
       LXI D, 0101H  ; Set counter
       MVI C, 01H    ; Set counter
       JR RET1      }
ARIN  : INX H       ; Increment HL
       JR RET2      } d
ARST0 : LXI H, 4020H ; Set data pointer
       JR RET1      } e
ARST1 : LXI H, 4004H ; Set data pointer
       MOV D, A      } f
       JR RET0      }
ARST2 : LXI H, 4024H ; Set data pointer
       MOV E, A      }
       MVI D, 00H    }
RET0  : MVI C, 00H
RET1  : MVI B, 03H
RET2  : XRI ANM, 08H ; Invert ANI2 bit
       EXA          ; Recover accumulator
       EXX          ; Recover register
       EI           ; Enable interrupt
       RETI         ; Return
```

CHAPTER 9 INTERRUPT CONTROL FUNCTION

The uPD78C11 contains three external interrupt requests NMI, INT1, and $\overline{\text{INT2}}$, eight internal interrupt requests INTT0, INTT1, INTE0, INTEL, INTEIN, INTAD, INTSR, and INTST, and one software interrupt instruction SOFTI. The interrupt requests except the SOFTI instruction are classified into six groups assigned six priority levels.

The interrupt address of the six interrupt request groups and SOFTI instruction are fixed as listed in Table 9-1.

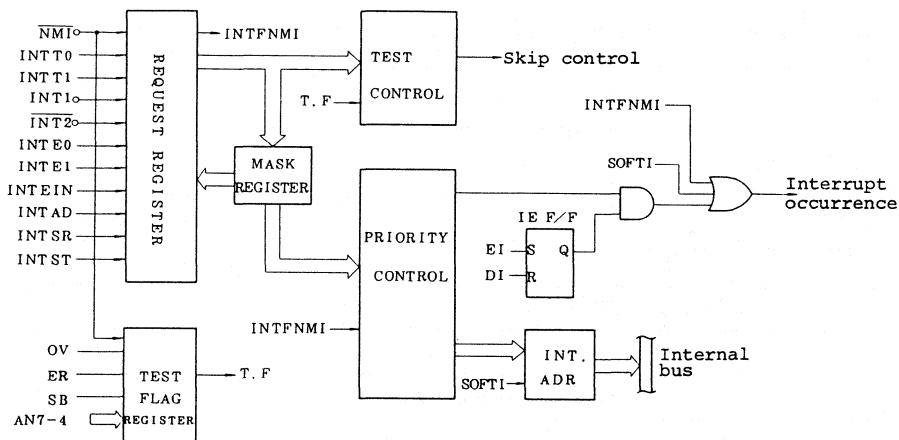
Table 9-1 Priority Levels and Interrupt Address

Priority level	Internal/ External	Interrupt request		Interrupt address		
				Decimal	Hexa	
1	External	NMI	Falling edge (nonmaskable interrupt)	4	0004	
2	Internal	INTT0	Match signal from TIMER0	8	0008	
		INTT1	Match signal from TIMER1			
3	External	INT1	Rising edge	16	0010	
		$\overline{\text{INT2}}$	Falling edge			
4	Internal	INTE0	Match signal from timer/event counter	24	0018	
		INTEL	Match signal from timer/event counter			
5		INTEIN	CI pin or T0 falling signal	32	0020	
		INTAD	A/D converter interrupt			
6		INTSR	Serial receive interrupt	40	0028	
		INTST	Serial transmit interrupt			
SOFTI instruction				96	0060	

9.1 Interrupt Control Circuit Configuration

The interrupt control circuit consists of the REQUEST REGISTER, MASK REGISTER, PRIORITY CONTROL, TEST CONTROL, INTERRUPT ENABLE F/F (IE F/F), and TEST FLAG REGISTER.

Fig. 9-1 Interrupt Control Circuit Block Diagram



(1) REQUEST REGISTER

The REQUEST REGISTER consists of 11 interrupt request flags, each of which is set when its corresponding interrupt request is made. Each interrupt request flag is reset when its corresponding interrupt request is acknowledged or a skip instruction SKIT or SKNIT is executed. When RESET is input, all the flags are reset. The interrupt request flags are not affected by setting the interrupt mask register.

- INTFNMI

Is set to 1 when the falling edge is input to the NMI pin. Unlike other interrupt request flags, the INTFNMI flag cannot be tested by executing a skip instruction. However, the NMI pin status can be tested. (See (6) TEST FLAG REGISTER.)

- INTFT0

Is set to 1 by the TIMER0 match signal.

- INTFT1

Is set to 1 by the TIMER1 match signal.

- INTF1

Is set to 1 when the rising edge is input to the INT1 pin.

- INTF2

Is set to 1 when the falling edge is input to the INT2 pin.

- INTFE0

Is set to 1 when the contents of the timer/event counter ECNT and ETM0 registers match.

- INTFE1

Is set to 1 when the contents of the timer/event counter ECNT and ETM1 registers match.

• INTFEIN

Is set to 1 by the timer/event counter input (CI input) or timer output (TO) falling edge.

• INTFAD

Is set to 1 when the A/D converter conversion values have been transferred to four registers CR0 to CR3.

• INTFSR

Is set to 1 when the receive buffer register on the serial interface becomes full.

• INTFST

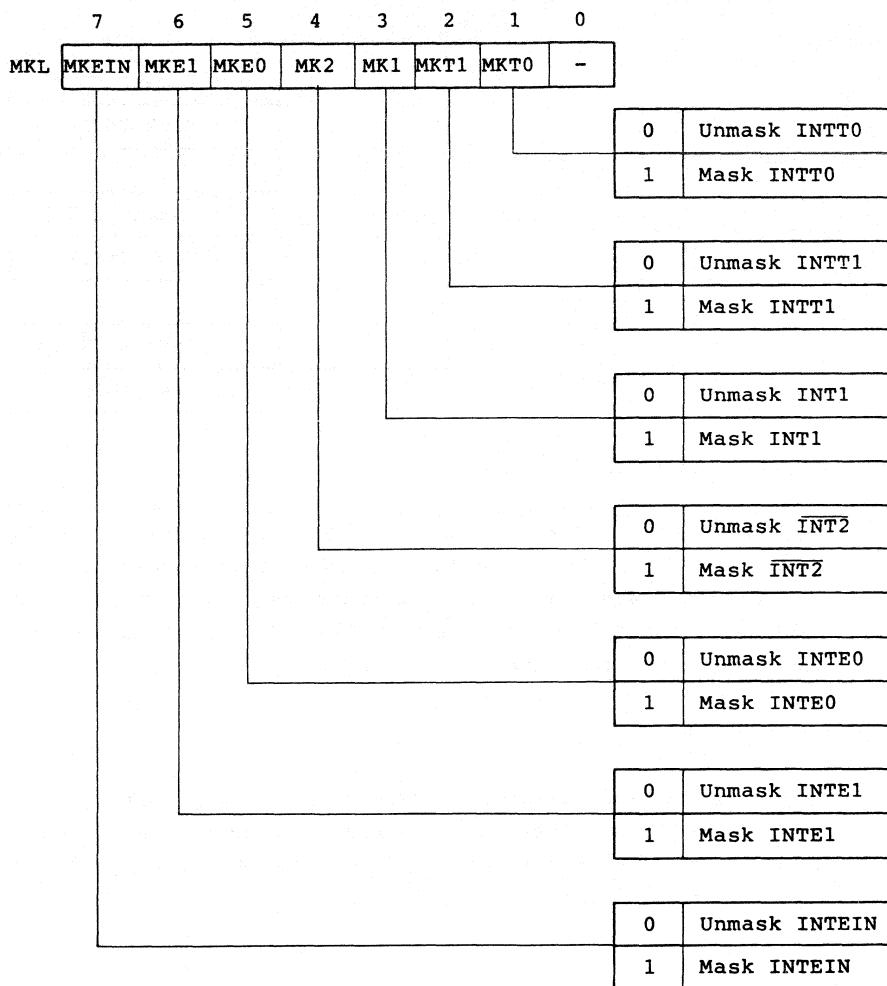
Is set to 1 when the transmit buffer register on the serial interface becomes empty.

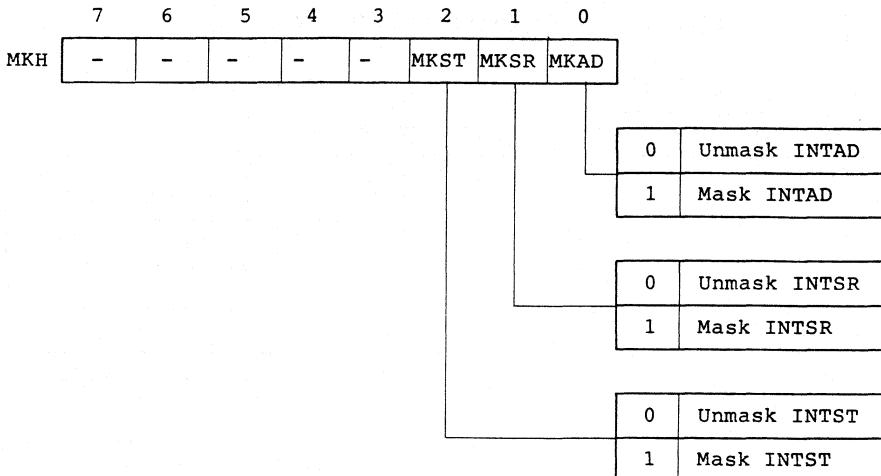
(2) MASK REGISTER

The MASK REGISTER contains 10 bits corresponding to the interrupt requests except nonmaskable interrupt NMI and can be set to 1 or reset to 0 bit-wise by executing an instruction. Each interrupt request is masked (disabled) when its corresponding bit of the mask register is set to 1; it is enabled when 0.

When RESET is input or during the hardware STOP mode, all bits of the mask register are set to 1 and interrupt requests except nonmaskable interrupt are masked.

Fig. 9-2 Mask Register (MKL, MKH) Format





(3) PRIORITY CONTROL

The PRIORITY CONTROL controls the six priority levels described above. If more than one interrupt request flag is set at the same time, the interrupt request assigned the highest priority level listed in Table 9-1 is acknowledged and other interrupt requests are held.

(4) TEST CONTROL

The TEST CONTROL operates in execution of skip instructions (SKET, SKNIT) to test the interrupt request flags except INTFNMI, the \overline{NMI} pin status, and the test flags not causing any interrupt request.

(5) INTERRUPT ENABLE F/F (IE F/F)

The INTERRUPT ENABLE F/F is a flip-flop which is set by executing the EI instruction and reset by executing the DI instruction. Once an interrupt is acknowledged, the flip-flop is reset. When RESET is input or during the hardware STOP mode, it is also reset. When the flip-flop is set, an interrupt is enabled; when reset, an interrupt is disabled. A nonmaskable interrupt is not affected by the flip-flop status and can always be acknowledged.

(6) TEST FLAG REGISTER

The TEST FLAG REGISTER consists of eight test flags not causing any interrupt request.

- NMI

The NMI pin status can be tested. If the NMI pin input level is 1, NMI is set to 1; if 0, NMI is set to 0.

- OV

Is set to 1 when the timer/event counter ECNT overflows.

- ER

Is set to 1 when a parity error, framing error, or overrun error occurs when serial data is received.

- SB

Is set to 1 when the V_{DD} pin rises from given or less low level to given or more high level.

. AN7-AN4

Are set to 1 when the falling edge is input to the AN7-AN4 pins. The falling edge is detected as with INT2.

These test flags can be tested by executing skip instruction SKIT, SKNIT. The test flags except NMI are cleared after tested. The NMI test flag is not affected by executing the instruction and the pin status can be tested as it is.

9.2 External Interrupt Sampling

The NMI, INT1, INT2 and AN7-AN4 pins have the noise removal function to prevent noise signal from causing malfunction.

(1) NMI input

NMI input is nonmaskable interrupt input (falling edge active). When the analog delay circuit detects that the NMI signal remains low for given time or longer, it is recognized as normal signal and the interrupt request flag INTFNMI is set.

INTFNMI is checked at the end of instruction. If INTFNMI is set, a jump is made to the interrupt address of the nonmaskable interrupt regardless of the EI or DI state. When an interrupt request is acknowledged, INTFNMI is automatically reset.

(2) INT1 input

INT1 input is maskable interrupt input (rising edge active). When the low-to-high transition of the INT1 signal is made and three ϕ_{12} cycle sampling pulses (12 states: 2.4 us at 15 MHz*) or more which are high are detected, the INT1 signal is recognized as normal signal and the interrupt request flag INTF1 is set.

When mask is unmasked in the EI state, if a check is made to ensure that INTF1 is set at the end of instruction and another interrupt request assigned the higher priority level does not exist, the INT1 interrupt is acknowledged and a jump is made to the interrupt address. See 9.4 for resetting the interrupt request flag.

A new INT1 interrupt is detected when the INT1 signal goes high in 12 states or more after the INT1 signal has been once restored low.

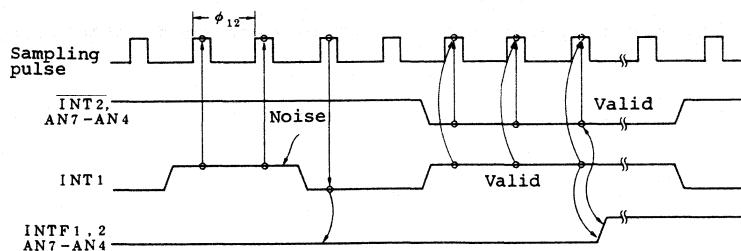
(3) INT2 input

INT2 input is maskable interrupt input (falling edge active). It has the same function as INT1 except that the active state is inverted.

(4) AN7-AN4 input

The falling edge is detected as in INT2 input and testable flag is set. The flag can be tested by executing instruction SKIT or SKNIT. It is automatically reset after tested. The testable flag is newly set when the input signal goes low in 12 states or more after the signal has been once restored high.

Fig. 9-3 Interrupt Sampling



As shown in Fig. 9-3, when three successive ϕ_{12} -cycle sampling pulses (0.8 us at 15 MHz) which are active are sampled, INT1, INT2 or AN7-AN4 is decided to be normal interrupt. Thus, noise signal of eight states (1.6 us at 15 MHz) or below is eliminated and the 12 states (2.4 us at 15 MHz) or more is input at high level or low level, the interrupt request flag is set completely.

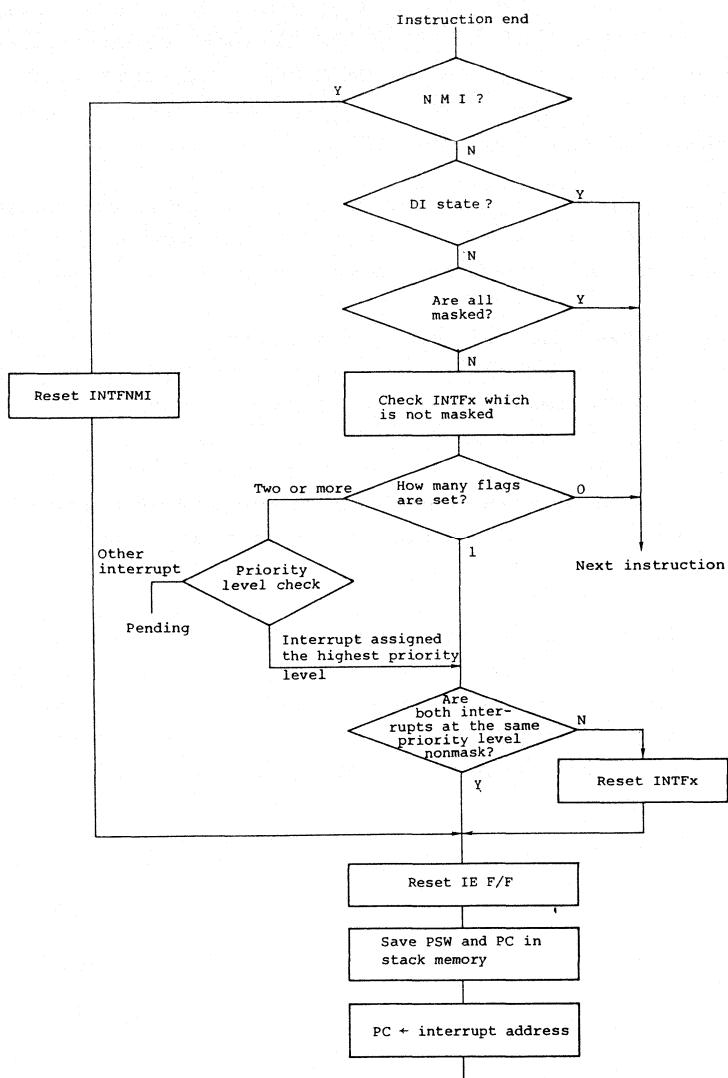
9.3 Nonmaskable Interrupt Operation

When interrupt request flag INTFNMI is set by inputting the falling edge to the NMI pin, nonmaskable interrupt is acknowledged regardless of the EI or DI state, as described below: (See Fig. 9-4.)

- (i) A check is made to see if INTFNMI is set at the last timing of each instruction. If INTFNMI is set, nonmaskable interrupt is acknowledged and INTFNMI is reset.
- (ii) When the nonmaskable interrupt is acknowledged, IE F/F is reset and all interrupt except the nonmaskable interrupt or SOFTI instruction are disabled (DI).
- (iii) PSW, the high-order byte of PC, and the low-order byte of PC are saved in stack memory in order.
- (iv) A jump is made to the interrupt address 0004H.

These interrupt operation sequence is performed automatically in 16 states.

Fig. 9-4 Interrupt Operation Sequence



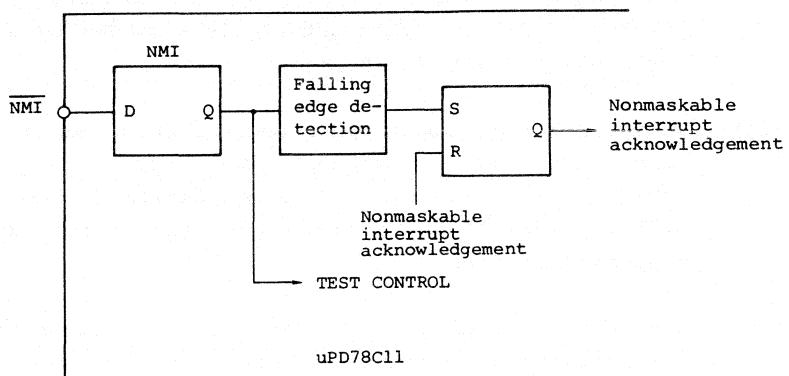
When interrupt service routine execution terminates, restore processing to the interrupt acknowledgement address is performed. First, the saved registers and flags other than PSW are restored, and IE F/F is set by executing the EI instruction as required. Next, the saved return address and PSW are restored in the order of the low-order byte of PC, the high-order byte of PC, and PSW by executing the RETI instruction.

Since the interrupt service of nonmaskable interrupt is entered regardless of the IE F/F status, the nonmaskable interrupt is useful to handle an emergency program such as power down.

Fig. 9-5 shows the NMI pin configuration. Although INTFNMI cannot be tested by executing a skip instruction, the NMI pin status can be tested by executing the skip instruction SKIT NMI or SKNIT NMI. Thus, comparatively wide noise can be removed by testing the NMI pin status by executing a skip instruction several times in the nonmaskable interrupt service routine. The NMI pin status does not change when it is tested by executing a skip instruction.

Caution: When a nonmaskable interrupt occurs, unconditionally IE F/F is reset. The IE F/F contents before the nonmaskable interrupt occurs are not stored. Thus, when a return is made to the main routine, the IE F/F contents cannot automatically be restored.

Fig. 9-5 NMI Pin Internal Configuration



9.4 Maskable Interrupt Operation

Interrupt requests other than nonmaskable interrupts or SOFTI instruction interrupts can be enabled or disabled by executing the EI or DI instruction to set or reset IE F/F; they are maskable interrupts which can be masked separately by using the mask register.

When active level of a maskable external interrupt is input for given time or longer and recognized as normal interrupt signal, its corresponding interrupt request flag is set. When an internal interrupt request occurs, immediately its corresponding interrupt request flag is set. Once an interrupt request flag is set, the interrupt is served as described below regardless of the external or internal interrupt: (See Fig. 9-3.)

- (i) If IE F/F = 1 (EI state), a check is made to see if the interrupt request flag is set at the last timing of each instruction. If the flag is set, the interrupt cycle is entered. However, interrupt requests masked by using the mask register are not checked.
- (ii) If more than one interrupt request flag is set at the same time, the interrupt requests are checked for priority level. The interrupt request assigned the highest priority level is acknowledged and other interrupt requests are held.
- (iii) When an interrupt request is acknowledged, its corresponding interrupt request flag is automatically reset. If both interrupt requests assigned the same priority level are unmasked in the mask register, the interrupt request flags are not reset because the interrupt requests will be distinguished later by software.
- (iv) When one interrupt request is acknowledged, IE F/F is reset and all interrupts other than nonmaskable interrupts or SOFTI instruction interrupts are disabled (DI).
- (v) PSW, the high-order byte of PC, and the low-order byte of PC are saved in stack memory in order.
- (vi) A jump is made to the interrupt address.

The interrupt operation sequence is automatically performed in 16 states.

When interrupts are enabled by executing the EI instruction, the pending interrupt request is acknowledged if another interrupt request assigned the higher priority level does not occur.

Two types of maskable interrupt requests are assigned the same priority level and the same interrupt address. Any of the following three can be selected by setting the mask register:

- Both interrupt requests are unmasked
- Either interrupt request is unmasked
- Both interrupt requests are masked

(1) When both interrupt requests are unmasked

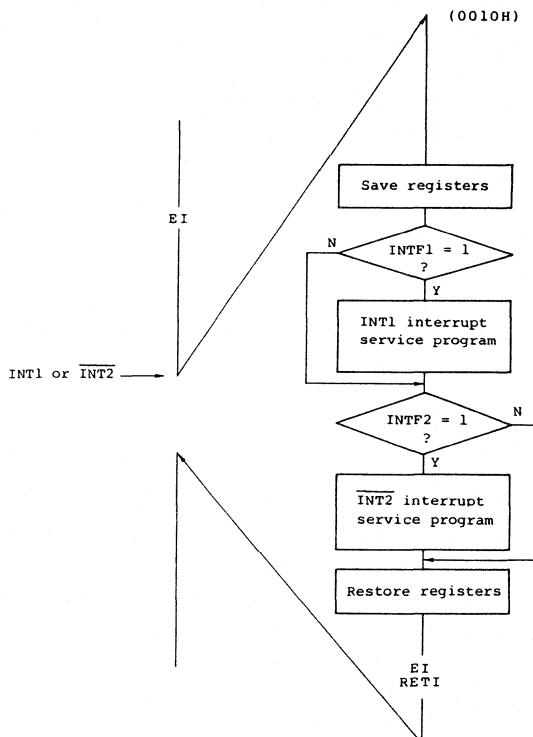
Both the mask register bits corresponding to the two interrupt requests are set to 0. In this case, an interrupt request is made according to the result of ORing two interrupt request flags.

Either or both of the interrupt request flags of the same priority level are set and the interrupt request is acknowledged according to the interrupt operation, then a jump is made to the interrupt address. In this case, however, the interrupt request flag is not reset. Thus, by executing a skip instruction to test the interrupt request flag at the beginning of the interrupt service routine, which interrupt request is decided and the interrupt request flag is reset.

The priority levels of the interrupt requests assigned the same priority level can be determined by the user as desired depending on which interrupt request the skip instruction is first executed for.

Fig. 9-6 shows the interrupt service sequence when both INT1 and INT2 are unmasked.

Fig. 9-6 Interrupt Service Sequence



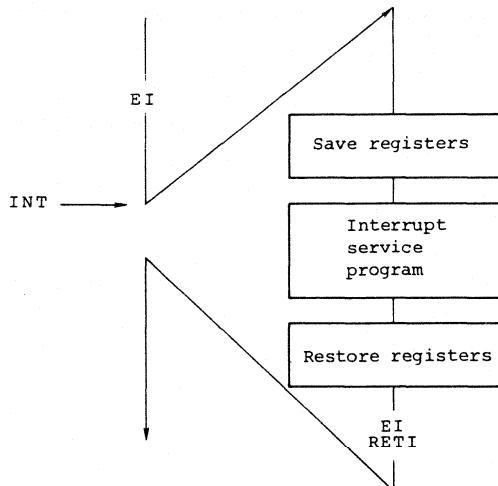
Remarks: When both interrupts assigned the same priority level, INT1 and INT2 are unmasked.

(2) When either interrupt request is unmasked

The mask register bit corresponding to the interrupt request to be unmasked of the two interrupt requests assigned the same priority level is set to 0 and the bit corresponding to the other interrupt request is set to 1. When the interrupt request flag corresponding to the unmasked interrupt request is set, interrupt request is generated.

When the interrupt request flag corresponding to the masked interrupt request is set, the interrupt request is held. When unmasked, the pending interrupt request is acknowledged if another interrupt request assigned the higher priority level does not exist in the interrupt enable state (EI). Whether or not the interrupt request flag corresponding to the acknowledged interrupt request is automatically reset is determined by how the mask register bits corresponding to the interrupt requests assigned the same priority level are set. When one interrupt request is unmasked, if the other interrupt request is masked, the interrupt request flag is automatically reset. However, if the other interrupt request remains unmasked when one interrupt request is unmasked, the interrupt request flag is not reset although interrupt request is acknowledged. (See (1) above.)

Fig. 9-7 Interrupt Service Sequence



Remarks: When either of the interrupt requests assigned the same priority level is unmasked by using the mask register.

(3) When both interrupt requests are masked

Both mask register bits corresponding to the two interrupt requests assigned the same priority level are set to 1. Although the interrupt request flag is set, interrupt request is not acknowledged and is held. When unmasked, the pending interrupt request is acknowledged if another interrupt request assigned the higher priority level does not exist in the interrupt enable state (EI).

When interrupt service routine execution terminates, restore processing to the interrupt acknowledgement address is performed. First, the saved registers and flags other than PSW are restored and IE F/F is set by executing the EI instruction. Next, the saved return address and PSW are saved in the order of the low-order byte of PC, the high-order byte of PC, and PSW by executing the RETI instruction.

9.5 SOFTI Instruction Interrupt Operation

When the SOFTI instruction is executed, unconditionally a jump is made to the interrupt address 0060H. The SOFTI instruction interrupt is not affected by IE F/F and SOFTI instruction execution does not affect IE F/F.

The SOFTI instruction interrupt is served as described below:

- (i) PSW, the high-order byte of PC, and the low-order byte of PC are saved in stack memory in order.
- (ii) A jump is made to the interrupt address 0060H.

When interrupt service routine execution terminates, restore processing to the interrupt acknowledgement address is performed. First, the saved registers and flags other than PSW are restored.

Next, the saved return address and PSW are restored in the order of the low-order byte of PC, the high-order byte of PC, and PSW by executing the RETI instruction.

Caution: If the skip condition is true in execution of an instruction just before the SOFTI instruction, such as an arithmetic or logical operation, increment or decrement, shift, skip, or RETS instruction, the SOFTI instruction is not skipped and is executed. When the SOFTI instruction is executed, the PSW SK flag which remains set to 1 is saved in the stack area. Thus, when a return is made from the SOFTI interrupt service routine, the PSW SK flag remains set and the instruction following the SOFTI instruction is skipped.

Note that the uCOM-87AD SOFTI instruction differs from the uCOM-87 SOFTI instruction in that the address contents saved in the stack memory are the top address of the next instruction.

9.6 Interrupt Wait Time

The required time until execution of the first instruction of the interrupt service routine is started after an external interrupt occurring asynchronously is acknowledged by the CPU, the interrupt wait time is the sum of wait time sources I, II, and III listed in Table 9-2.

The interrupt wait time varies depending on the type of instruction being executed when an interrupt occurs and what timing of the instruction the interrupt occurs at.

Table 9-2 lists the maximum interrupt wait time.

14 states of source I (maximum of 10 us for \overline{NMI}) mean the time required until interrupt request signal is activated and recognized as normal signal and INTFx is set to 1. Thus, this time is not required for interrupts other than \overline{NMI} , INT1 or INT2.

59 states of source II mean the execution time of the maximum one instruction. This time is required because INTFx is checked at the end of each instruction (METE). Thus, the time of source II varies depending on the type of instruction being executed at the time; it ranges from four to 59 states.

16 states of source III are the time required to save the PSW and PC contents in stack memory.

Table 9-2 Maximum Interrupt Wait Time

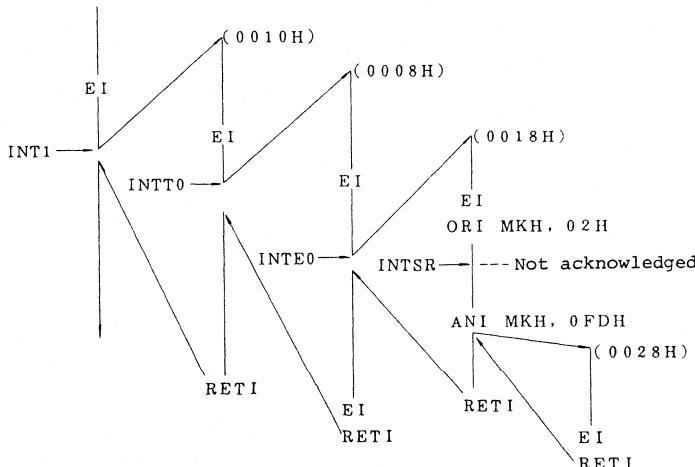
Wait time source		INT1, $\overline{INT2}$	\overline{NMI}	Others
I	Time required to remove noise signal	14 states	10 us MAX	0 state
II	Time required for instruction execution (division instruction)	59 states	59 states	59 states
II	Time required for automatic save processing	16 states	16 states	16 states
Total time		89 states (22.25 us/12 MHz)	75 states + 10 us (28.75 us/12 MHz)	75 states (18.75 us/12 MHz)

9.7 Multiple Interrupts

When the EI instruction is executed, all external and internal interrupt requests containing the interrupt being served are enabled. Thus, if the EI instruction is executed during interrupt service routine execution, the interrupt request and the interrupt requests assigned the lower priority level are also be acknowledged. When more than one interrupt request assigned the highest priority level is acknowledged and other interrupt requests assigned the lower priority are held. When interrupts are enabled, the pending interrupt request is acknowledged if another interrupt request assigned the higher priority level does not occur.

Since stack area when interrupt occurs is limited only by the memory size, interrupt nesting levels are also limited only by the memory size. (See Fig. 9-8.)

Fig. 9-8 Multiple Interrupts at Three Levels



Remarks: If both interrupt sources assigned the same priority level are unmasked by setting the mask register, which interrupt request must be decided before the EI instruction is executed at the top of the interrupt service routine.

CHAPTER 10 CONTROL FUNCTION

10.1 Standby Function

The uPD78C11 contains three standby modes (HALT, software STOP, and hardware STOP) to save power consumption during program standby.

10.1.1 HALT mode

Whenever the HLT instruction is executed, the HALT mode is entered unless the interrupt request flag of an unmasked interrupt is set. In the HALT mode, the CPU clock stops and program execution is stopped, but the contents of all registers and internal RAM just before the HALT mode is entered are retained. During the HALT mode, circuits such as the timer, timer/event counter, serial interface, A/D converter, and interrupt control circuit can operate. Table 10-1 lists the uPD78C11 output pin status during the HALT mode.

Table 10-1 Output Pin Status

Output pin	Single chip (Note 1)	External expansion
PA7-0	Data retention	Data retention
PB7-0	Data retention	Data retention
PC7-0	Data retention	Data retention
PD7-0	Data retention	High impedance
PF7-0	Data retention	Retention of next address (Note 2) Data retention (Note 3)
WR, RD	High	High
ALE	High	High

Note 1: uPD78C14/78C14A/78C12A/78C11/78C11A

2: Address output pin

3: Port data output pins

Caution: Since interrupt request flag is used to release the HALT mode, if any interrupt request flag of an unmasked interrupt is set, the HALT mode cannot be entered although the HLT instruction is executed. Thus, to set the HALT mode at a place where interrupt request flag may be set (pending interrupt may exist), first serve the pending interrupt, reset the interrupt request flag by executing a skip instruction, or mask all interrupts other than the interrupt used to release the HALT mode.

10.1.2 HALT mode release

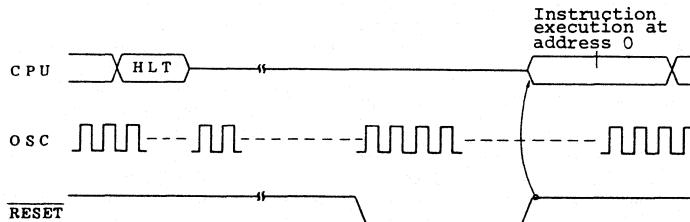
(1) Release using RESET signal

When the high-to-low transition of the RESET signal is made during the HALT mode, the HALT mode is released and the uPD78C11 is reset.

When the RESET signal is restored high, the CPU starts program execution at address 0.

When the RESET signal is input, the RAM contents are retained, but other registers become undefined.

Fig. 10-1 HALT Mode Release Timing (RESET Signal Input)



(2) Release using interrupt request flag

If one or more interrupt request flags are set when one or more of nonmaskable interrupt (\overline{NMI}) and unmasked maskable interrupts (INTT0, INTT1, INT1, $\overline{INT2}$, INTE0, INTE1, INTEIN, INTAD, INTST, and INTSR) occur during the HALT mode, the HALT mode is released.

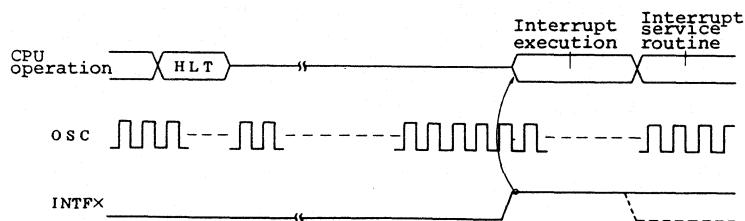
If the HALT mode is released by using a nonmaskable interrupt, the instruction following the HLT instruction is not executed and a jump is made to the interrupt address 0004H regardless of whether interrupts are enabled or disabled.

If the HALT mode is released by using a maskable interrupt, the operation after the HALT mode is released varies depending on whether interrupts are enabled or disabled.

(i) When interrupts are enabled (EI)

The instruction following the HLT instruction is not executed and a jump is made to the interrupt address.

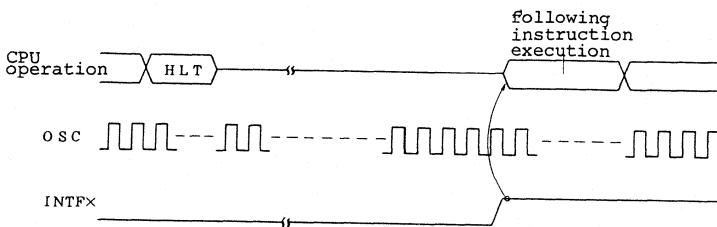
Fig. 10-2 HALT Mode Release Timing (when EI = 1)



(ii) When interrupts are disabled (DI)

Execution is restarted at the instruction following the HLT instruction. (No jump is made to the interrupt address.) The interrupt request flag used to release the HALT mode remains set. Reset the flag by executing a skip instruction as required.

Fig. 10-3 HALT Mode Release Timing (when DI = 1)



10.1.3 Software STOP mode

Whenever the STOP instruction is executed, the software STOP mode is entered unless the interrupt request flag of an unmasked external interrupt is set. In the software STOP mode, all clocks stop. When the software STOP mode is entered, program execution is stopped, the contents of all registers and internal RAM and flags other than FT0 or FT1 just before the software STOP mode is entered are retained (timer UPCOUNTER is cleared to 00H), the NMI and RESET signals used to release the software STOP mode are only valid, and all other functions stop.

The uPD78C11 output pin status during the software STOP mode is the same as that during the HALT mode, as listed in Table 10-2.

Table 10-2 Output Pin Status

Output pin	Single chip ^(Note 1)	External expansion
PA7-0	Data retention	Data retention
PB7-0	Data retention	Data retention
PC7-0	Data retention	Data retention
PD7-0	Data retention	High impedance
PF7-0	Data retention	Retention of next address (Note 2) Data retention (Note 3)
WR, RD	High	High
ALE	High	High

Note 1: uPD78C14/78C14A/78C12A/78C11/78C11A

2: Address output pin

3: Port data output pins

Caution 1: To prevent malfunction caused by an internal interrupt occurring within the oscillator operation stable time when the software STOP mode is released, mask internal interrupts before STOP instruction execution.

Caution 2: The TIMER1 match signal is used as a signal to start CPU operation to obtain the oscillator operation stable time when the software STOP mode is released by setting the interrupt request flag of a nonmaskable interrupt. Thus, set the count value considering the oscillator operation stable time in TIMER REG and set the timer mode register to the timer operation state before STOP instruction execution.

Caution 3: To use the software STOP mode, use a crystal or ceramic resonator. If external clock is input, do not use the software STOP mode.

10.1.4 Software STOP mode release

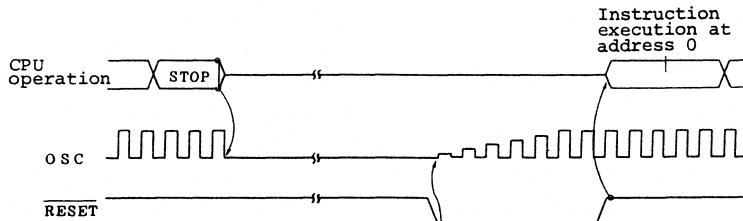
(1) Release using RESET signal

When the high-to-low transition of the RESET signal is made during the software STOP mode, the software STOP mode is released and the uPD78C11 is reset. At the same time, clock oscillation is started. If the RESET signal is set high after oscillator operation becomes stable, the CPU starts program execution at address 0.

When the high-to-low transition of the RESET signal is made, clock oscillation is started, but time is taken until oscillator operation becomes stable. Thus, the low width of the RESET signal must be taken longer than the oscillator operation stable time.

When the RESET signal is input, the RAM contents are retained, but other registers become undefined.

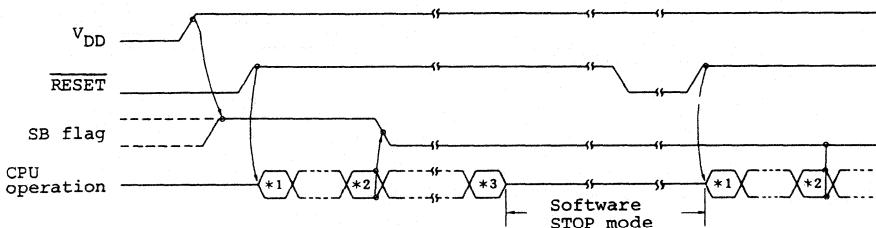
Fig. 10-4 Software STOP Mode Release Timing (RESET Signal Input)



If the software STOP mode is released by using the RESET signal, program execution is started at address 0 as with normal power on reset. To distinguish them, the standby (SB) flag can be used. When the V_{DD} pin rises from given or

less low level to given or more high level, the SB flag is set to 1; when a skip instruction is executed, the SB flag is reset to 0. Thus, power on start and software STOP mode release start can be distinguished from each other by executing a skip instruction to test the SB flag in a program after RESET is input. (See Fig. 10-5.) If the SB flag is set to 1, power on start is indicated; if the SB flag is reset to 0, software STOP mode release start is indicated.

Fig. 10-5 SB Flag Operation



*1: Instruction execution at address 0

*2: SKIT SB or SKNIT SB instruction execution

*3: STOP instruction execution

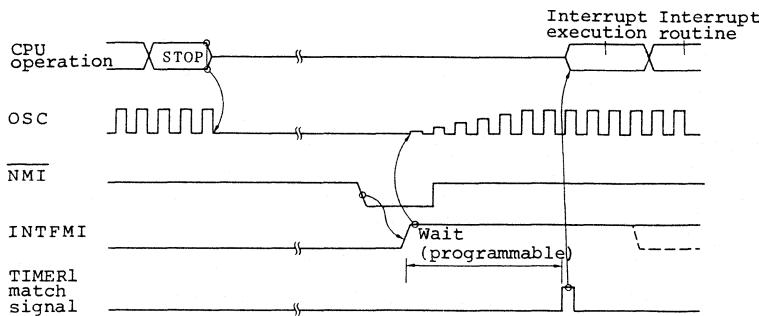
(2) Release using NMI pin input

When the nonmaskable interrupt request flag is set (NMI pin input goes from high to low) during the software STOP mode, the software STOP mode is released and clock oscillation is started at the same time. When clock oscillation is started, timer UPCOUNTER starts counting at 00H as set before the STOP instruction is executed. The CPU starts operation according to the match signal from the TIMER1 UPCOUNTER (wait time considering the oscillator operation stable time). The UPCOUNTER match signal does not set the

interrupt request flag. The timer mode register of the timer after the match signal is generated is set to FFH and the timer stops operation.

After the oscillator operation stable time has elapsed, the instruction following the STOP instruction is not executed and a jump is made to the interrupt address 0004H regardless of whether interrupts are enabled or disabled.

Fig. 10-6 Software STOP Mode Release Timing



10.1.5 Hardware STOP mode

Whenever the high-to-low transition of the STOP signal is made, the hardware STOP mode is entered. In the hardware STOP mode, all clocks stop. When the hardware STOP mode is entered, program execution is stopped; the internal RAM contents just before the hardware STOP mode is entered are retained; the STOP signal used to release the hardware STOP mode is only valid; all other functions stop; and the uPD78C11 is reset. All uPD78C11 output pins become high impedance during the hardware STOP mode. However, port output latch values are retained.

Caution 1: To use the hardware STOP mode, use a crystal or ceramic resonator. If external clock is input, do not use the hardware STOP mode.

Caution 2: The hardware STOP mode is entered on the machine cycle boundaries. Thus, the memory contents are not destroyed, but the hardware STOP mode may be entered during instruction execution. Therefore, in the 16-bit data transfer instruction, although 8 bits are completed to transfer and the other 8 bits are not completed, STOP mode is input (16-bit data transfer instruction and Call instruction).

Caution 3: If the low-to-high of the STOP signal is input even though RESET is input (RESET is low level), RESET status is transited to STOP mode.

Application for Hardware STOP Mode

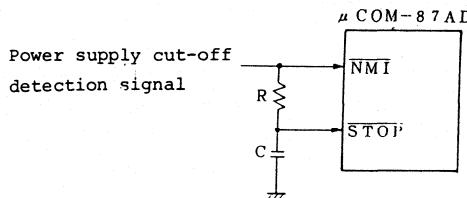
When the hardware STOP mode is executed in CPU operation and asynchronous mode, the device may consume approx. 20 mA power supply current after being input into the hardware STOP mode. Use together with STOP input and the following signals, and execute CPU operation and synchronous mode using JR\$ instruction.

- . NMI
- . RESET

(1) At used together with NMI

Input the power supply cut-off detection signal into NMI pin as nonmaskable interrupt request, and input the delay signal into STOP pin as a hardware STOP mode setting signal.

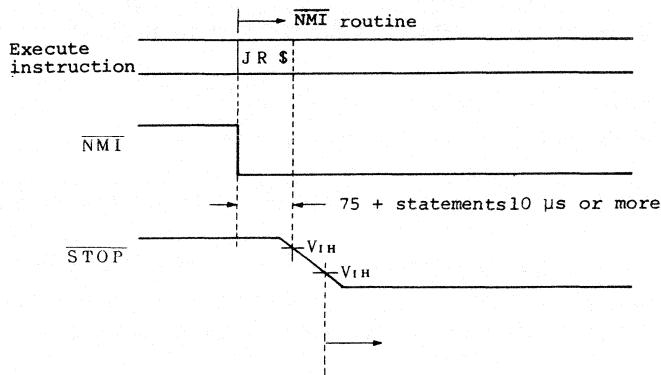
Fig. 10-7 At Used Together with NMI and STOP



Operation procedure is shown below.

- (a) When the power supply cut-off detection signal is entered, NMI routine is started.
- (b) Execute JR\$ instruction at the first of NMI routine, and wait for STOP by looping the program.

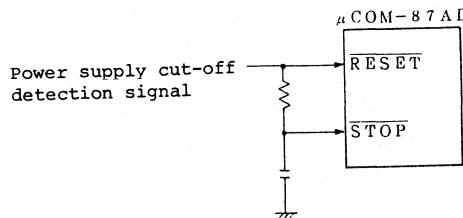
Determine R and C values so that the delay of NMI and STOP takes longer than max. interrupt wait time = 75 statements + 10 us (28.75 us: at operation of 12 MHz).

Fig. 10-8 $\overline{\text{NMI}}$ and $\overline{\text{STOP}}$ 

When the power supply cut-off is executed before execution of $\text{JR\$}$ in NMI routine, it is required for STOP delay to take longer wait corresponding with $\overline{\text{NMI}}$ routine.

(2) At used together with $\overline{\text{RESET}}$

$\overline{\text{STOP}}$ input is normally acknowledged during $\overline{\text{RESET}}$ ($\overline{\text{RESET}}$ is low level). If $\overline{\text{RESET}}$ input is active before inputting $\overline{\text{STOP}}$, hardware STOP mode is expected to be normal.

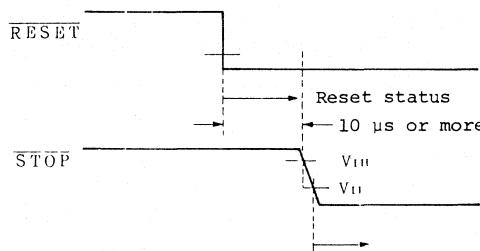
Fig. 10-9 $\overline{\text{RESET}}$ and $\overline{\text{STOP}}$ 

Operation procedure is shown below.

- (a) When power supply cut-off detection signal ($\overline{\text{RESET}}$ signal) is entered, reset status is become.
- (b) Delay $\overline{\text{RESET}}$ signal more than 10 us, use it as STOP signal.
- (c) Hardware STOP mode is entered in reset status

However, this method may break the data memory content by $\overline{\text{RESET}}$ input. When $\overline{\text{RESET}}$ input is entered during writing data into data memory by CPU, the corresponding address is undefined.

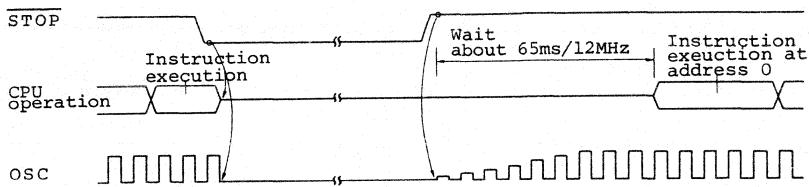
Fig. 10-10 $\overline{\text{RESET}}$ and STOP



10.1.6 Hardware STOP mode release

If the low-to-high transition of the $\overline{\text{STOP}}$ signal is made during the hardware STOP mode, the hardware STOP mode is released and clock oscillation is started at the same time. After this, when the wait time considering the oscillator operation stable time (about 65 ms at 12 MHz) has elapsed, the CPU starts program execution at address 0. (See Fig. 10-11.)

Fig. 10-11 Hardware STOP Mode Release Timing



The hardware STOP mode is not released although the high-to-low transition of the RESET signal is made. If the low-to-high transition of the STOP signal is made when the RESET signal is low, the hardware STOP mode is released and clock oscillation is started. After this, when the RESET signal is restored from low to high, the CPU starts program execution at address 0 without taking the oscillator operation stable time. (See Fig. 10-12.)

If the high-to-low transition of the RESET signal is made just after the hardware STOP mode is released (the low-to-high transition of the STOP signal is made), program execution is also started when the RESET signal is restored from low to high. (See Fig. 10-13.)

Thus, restore the RESET signal to high by considering the oscillator operation stable time.

When the RESET signal is input, the RAM contents are retained, but other registers become undefined.

Caution: Be sure to set the STOP pin high just after power on. The STOP pin may be set low after the oscillator operation becomes stable.

Fig. 10-12 Hardware STOP Mode Release Timing

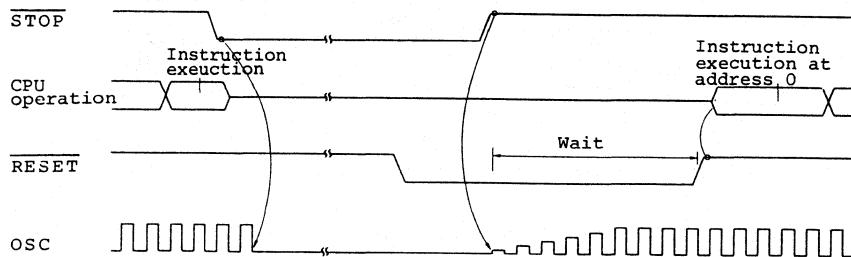
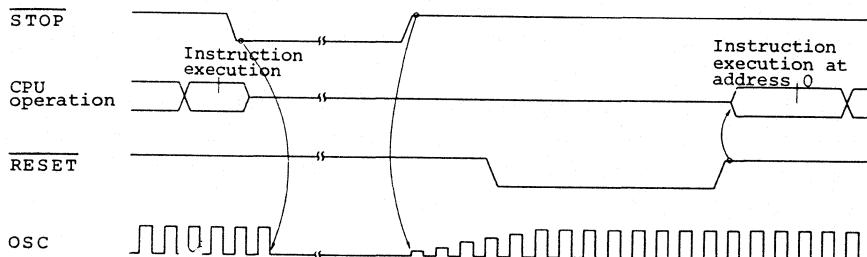


Fig. 10-13 Hardware STOP Mode Release Timing

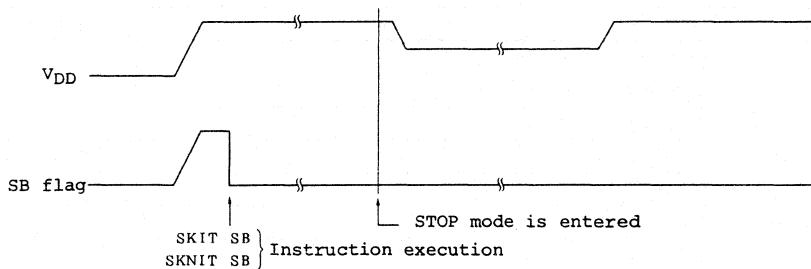


10.1.7 Low supply voltage data retention mode

The low supply voltage data retention mode can be set simply by lowering the V_{DD} supply voltage after the software or hardware STOP mode is set; the RAM contents can be retained by power consumption lower than the software or hardware STOP mode.

The SB flag is provided to distinguish return from the software or hardware STOP mode by reset from power on reset. The SB flag is set to 1 only when the supply voltage (V_{DD}) changes from given or less voltage to given or higher voltage. The SB flag can be tested by executing the SKIT SB or SKNIT SB instruction; it is automatically reset to 0 when the instruction is executed.

Fig. 10-14 Relationship between V_{DD} and SB Flag



Caution: Do not release the software or hardware STOP mode in the low supply voltage data retention mode. Be sure to rise V_{DD} to the normal operation voltage before releasing the software or hardware STOP mode.

10.2 Reset Function

When a low pulse is input to the RESET pin, the system is reset and initialized as follows:

Table 10-3 Status after each Hardware Resetting

Hardware				Status after resetting	
Internal data memory	At power-on reset			Retain the previous content	
	At reset input during normal operation	During writing by CPU	Write corresponding address data	Undefined register	
			Address data except the above	Retain the previous content	
		During except writing by CPU		Retain the previous content	
Extension accumulator (EA, EA')				Undefined register	
Accumulator (A, A')					
General purpose register (B, C, D, E, H, L, B', C', D', E', H', L')					
Program counter (PC)				0000H	
Stack Pointer (SP)				Undefined register	
Port	Mode register (MA, MB, MC, MF)			FFH	
	Mode control register (MCC)			00H	
	MM register (MM0, 1, 2)				
Output latch of each port				Undefined register	
Interrupt	INTERRUPT ENABLE F/F			0	
	Request flag			FFH	
	Mask register				
Test flag (except for SB flag)				0	
Standby flag (SB)	At power-on reset			1	
	At standby mode			Retain the previous content	
	At the RESET input during normal operation			Undefined register	
Timer	Timer mode register (TMM)			FFH	
	Timer F/F				
	Timer register (TMO, TMI)			Undefined register	
Timer event counter	Timer event counter mode register (ETMM)			00H	
	Timer event counter output mode register (EOM)				
	Timer event counter register (ETMO, ETMI)			Undefined register	
	Timer event counter capture register (ECPT)				

(to be cont'd)

Hardware		Status after resetting
Serial interface	Serial mode high register (SMH)	00H
	Serial mode low register (SML)	48H
A/D channel mode register (ANM)		00H
MM register RAE-bit (MM3)		Undefined register
Zero cross mode register (ZC)		1

Table 10-4 Status after each Pin Resetting

Pin	Status after reset
WR	
RD	
ALE	
All the ports (PA, PB, PC, PD, PF)	High impedance

When the low-to-high transition of RESET input is made, program execution is started at address 0. Initialize or reinitialize the contents of the registers in a program as required.

Caution: If V_{DD} is within the operation voltage range with no X_1 input, when the RESET signal is input,

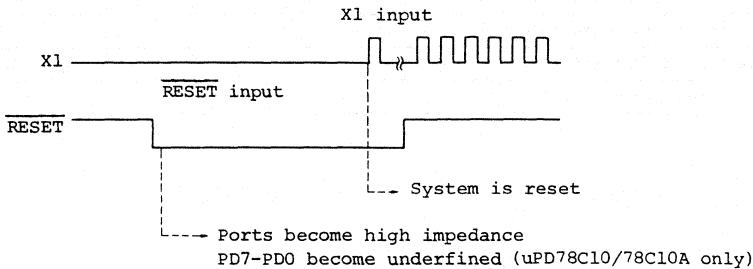
- o uPD78C14/78C14A/78C12A/78C11/78C11A:

All pins become impedance

- o uPD78C10/78C10A: The PD7-PD0 pins become undefined.

Other pins than PD7-PD0 become
high impedance.

All this, when X1 is input, the system is reset.



10.3 Clock Generator

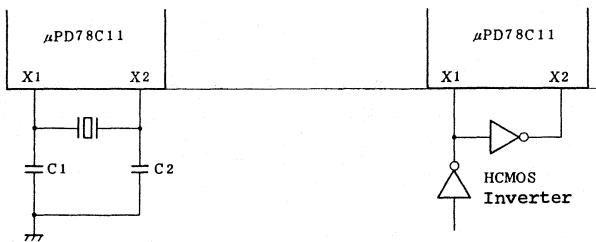
The uPD78C11 contains an internal clock generator; clocks required for operation can be generated simply by connecting a crystal or ceramic resonator and capacitors. Externally generated clocks can also be input. Fig. 10-15 shows a resonator connection circuit. Fig. 10-16 shows an external clock input circuit example.

Fig. 10-15

Resonator Connection Circuit

Fig. 10-16

External Clock Input Circuit Example



To use a crystal resonator, set C1 = C2 = 10 pF. Table 10-5 lists the recommended resonators and C1 and C2 values when a ceramic resonator is used.

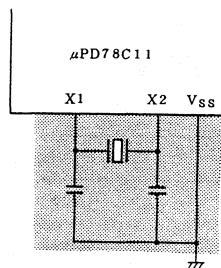
Table 10-5 Recommended Ceramic Resonators

	Manufacturer name	Product name	C1(pF)	C2(pF)
uPD78C14/ 78C14A	MURATA MANUFACTURING CO., LTD.	CSA12.0MT	30	30
		CST12.0MT	Capacitor is contained.	
uPD78C11/ 78C10	MURATA MANUFACTURING CO., LTD.	CSA12.0MT18	30	30
		CST12.0MT18	Capacitor is contained.	
		CSA7.37MT	30	30
		CST7.37MT	Capacitor is contained.	

When a resonator is connected, the oscillator becomes a high frequency analog circuit. For pattern design, note the following:

- (i) Make wiring as short as possible.
- (ii) Do not cross the circuit and other signal lines on layout.
- (iii) Do not cause current to flow on the ground pattern between the V_{SS} pin and capacitor connection part to the ground.
- (iv) Connect resonator and capacitors only to the X1 and X2 pins.

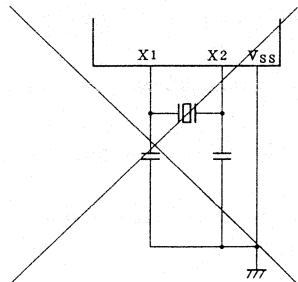
Fig. 10-17 Cautions on Resonator Connection Circuit



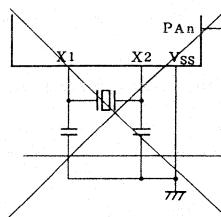
- Bring the oscillator near to the X1 and X2 pins as much as possible.
- Do not pass other signal lines through the shaded region.

Fig. 10-18 Bad Examples of Resonator Connection Circuit

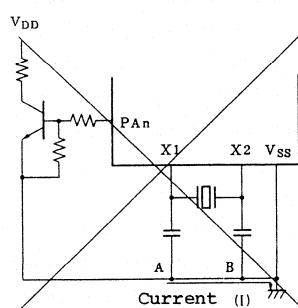
(i) Connection circuit wiring is long



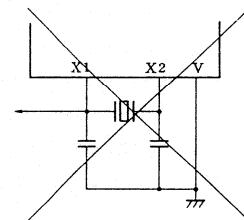
(ii) The circuit and other signal lines are crossed



(iii) Current flows on oscillator ground line (potential at A and B points fluctuates)



(iv) Signal is taken out



To input external clock, make extremely short wiring to prevent unnecessary electromagnetic wave radiation or external noise from adversely affecting the circuit.

When the hardware or software STOP mode is entered, the X1 and X2 pin levels are fixed. Thus, to use external clock, do not use the hardware or software STOP mode. To use the software or hardware STOP mode, use a crystal or ceramic resonator.

When the power is turned on or a return is made from the hardware or software STOP mode, the time until oscillation becomes stable must be taken. Generally, the time of several ms (when a crystal resonator is used) or several hundred us (when a ceramic resonator is used) is required to stabilize oscillation.

Take sufficient oscillation stable time by using

- 1) RESET input (reset period) when the power is turned on
- 2) RESET input (reset period) or automatic timer when a return is made from the hardware STOP mode
- 3) RESET input or preset timer when a return is made from the software STOP mode

Reference: Proper use of crystal and ceramic resonators

Generally, the crystal resonator oscillator frequency is extremely stable. Thus, the crystal resonator is appropriate for high-accuracy time management, such as timer or frequency measurement.

The oscillator frequency stability of a ceramic resonator is inferior to that of a crystal resonator, but the ceramic resonator has the three features of fast oscillation start time, small size, and low price. Thus, it is useful for general applications where high-accuracy time management is not required. Some ceramic resonators contain capacitors, the number of parts and the packaging area can be reduced.

CHAPTER 11 EXTERNAL DEVICE ACCESS AND TIMING

11.1 uPD78C14/78C14A/78C12A/78C11/78C11A External Device Access

The uPD78C14/78C14A/78C12A/78C11/78C11A enables external device (data memory, program memory, or peripheral device) expansion to the following area:

- uPD78C14/78C14A: Addresses 4000H-FEFFFH (48K bytes)
- uPD78C12A: Addresses 2000H-FEFFFH (56K bytes)
- uPD78C11/78C11A: Addresses 1000H-FEFFFH (60K bytes)

To expand external device, the PD7-PD0 pins are used as a multiplexed address/data bus (AD7-AD0) and the PF7-PF0 pins are used as an address bus (AB15-AB8) by setting the MEMORY MAPPING register (MM). The number of bits of the PF7-PF0 pins used as the address bus can be changed according to the externally expanded memory size; memory can be expanded by stages in the range of 256 bytes to 48 K/56 K/60 K bytes (dependent on each products). The pins not used as the address bus can be used as general purpose input/output port pins. (See Table 11-1.)

Table 11-1 PF7-PF0 Address Bus Selection

PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0	External address space
Port	Within 256 bytes							
Port	Port	Port	Port	AB11	AB10	AB9	AB8	Within 4K bytes
Port	Port	AB13	AB12	AB11	AB10	AB9	AB8	Within 16K bytes
AB15	AB14	AB13	AB12	AB11	AB10	AB9	AB8	Within 48K, 56K or 60K bytes (Note)

Note: 48K = uPD78C14/78C14A, 56K = uPD78C12A, 60K = uPD78C11/78C11A

When an external device reference instruction is executed in the 256-byte expansion mode, the uPD78C14/78C14A/78C12A/78C11/78C11A masks the high-order eight bits of a 16-bit external reference address and outputs 00H-FFH as address information from the PD7-PD0 (AD7- AD0) pins.

Likewise, in the 4K-byte expansion mode, the uPD78C14/78C14A/78C12A/78C11/78C11A masks the high-order four bits of a 16-bit external reference address and outputs 000H-FFFH as address information from the PF3-PF0 (AB11-AB8) and PD7-PD0 pins.

Likewise, in the 16K-byte expansion mode, the uPD78C14/78C14A/78C12A/78C11/78C11A masks the high-order two bits of a 16-bit external reference address and outputs 0000H-3FFFH as address information from the PF5-PF0 (AB13-AB8) and PD7-PD0 pins.

Since the high-order bits of a 16-bit address are masked in the 256-byte, 4K-byte, or 16K-byte expansion mode, external device can be placed in any desired 256-byte, 4K-byte, or 16K-byte area within the 60K-byte area. However, note that if external ROM is connected to the expansion area and the external ROM area is mapped in addresses 1000H-4FFFH following the internal ROM in the 16K-byte expansion mode, the program counter (PC) contents differ from the actual address output from the PF5-PF0 and PD7-PD0 pins as follows:

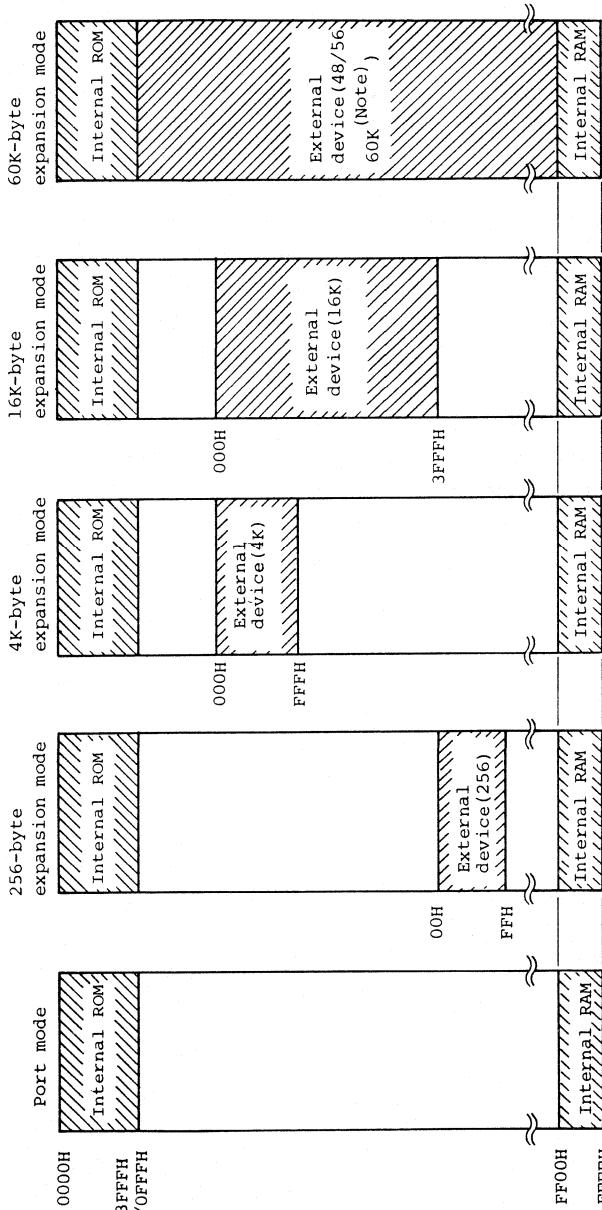
PC	PF5-0, PD7-0
1000H	1000H
:	:
3FFFH	3FFFH
4000H	0000H
:	:
4FFFH	0FFFH

To make consecutive external ROM addresses, set the external ROM area in addresses 4000H-7FFFH. In this case, the internal ROM address area is not contiguous to the external ROM address area, and a jump instruction must be used to move a program between the areas. This also applies to the case where the external ROM area is set in addresses 8000H-BFFFH.

Caution 1: The internal address bus contents are output to port D which serves as the address/data bus in every machine cycle. The internal address bus contents are output intact from the port F pins which serve as the address bus in every machine cycle. However, the \overline{RD} and \overline{WR} signals are output only in memory cycles.

Caution 2: Do not use software which changes port D, port F operation mode dynamically because it cannot be emulated by an emulator.

Fig. 11-1 External Expansion Mode Selected by Using MEMORY MAPPING Register



11.1.1 MEMORY MAPPING register (MM)

The MEMORY MAPPING register (MM) is an 8-bit register to control the following:

- Port or expansion mode selection for PD7-PD0 and PF7-PF0
- Whether or not internal RAM access is enabled
- Internal EPROM access range specification (uPD78C14 only:
See Chapter 12.)

Fig. 11-2 shows the MEMORY MAPPING register format.

(1) MM0-MM2 bits

The MM0-MM2 bits are used to control port or expansion mode selection for PD7-PD0 and PF7-PF0. As shown in Fig. 11-2, the capacity of external memory that can be connected can be selected among the following:

- 256 bytes
- 4K bytes
- 16K bytes
- 48K bytes (uPD78C14/78C14A), 56K bytes (uPD78C12A), or 60K bytes (uPD78C11/78C11A). For uPD78CP14, it depends on specification in the MM6 and MM7 bits.

The PF7-PF0 pins not used for address output can be used as general purpose port pins.

When RESET is input or during the hardware STOP mode, the bits are reset to 0 and PD7-PD0 become input port pins (output high impedance).

(2) MM3 bit (RAE)

The MM3 bit is used to specify whether internal RAM access is enabled (RAE = 1) or not (RAE = 0). During the standby operation or when external RAM rather than internal RAM is used, set the MM3 bit to 0.

If RESET is input during normal operation, the MM3 bit status at the time is retained.

Caution 1: Apparently, the memory space can be increased 256 bytes by executing a program while the RAE bit is being rewritten. However, do not perform this operation because it cannot be emulated by an emulator.

Caution 2: The RAE bit becomes undefined when power on reset. Initialize the RAE bit by executing an instruction.

In addition, MM6 and MM7 bits to specify the internal EPROM access range can be used on the uPD78CP14. For details, see Chapter 12.

Fig. 11-2 MEMORY MAPPING Register Format

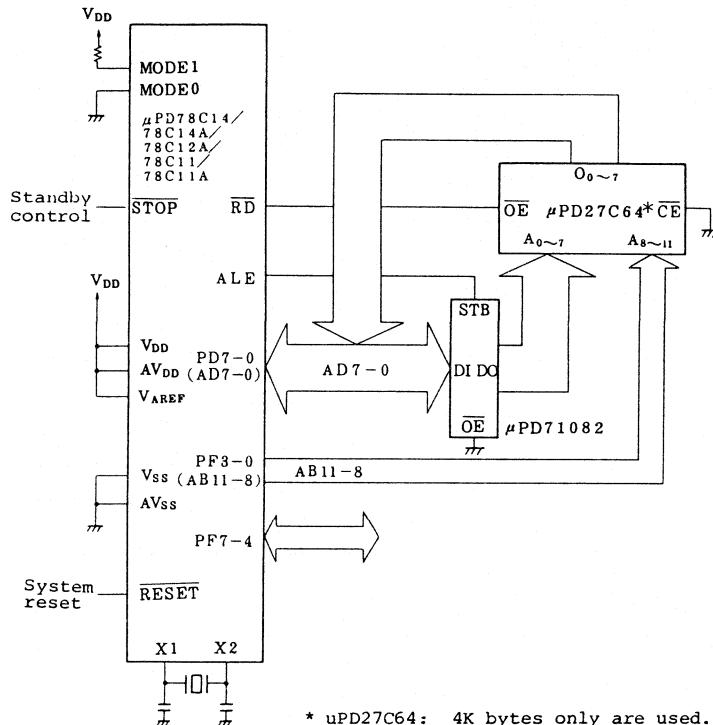
7	6	5	4	3	2	1	0	
-	-	-	-	RAE	MM2	MM1	MM0	
0	0	0						Port mode
0	0	1						
0	1	0						Expansion mode
1	0	0						
1	1	0						
1	1	1						
Internal RAM access								
0	Disable							
1	Enable							

Note: 48K bytes = uPD78C14/78C14A, 56K bytes = uPD78C12A, 60K bytes = uPD78C11/78C11A

11.1.2 Memory expansion example

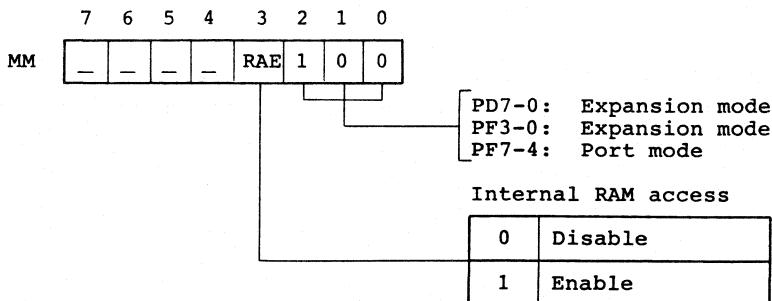
Fig. 11-3 shows a configuration example where 4K-byte external ROM is added. Fig. 11-4 shows the setup data in the MEMORY MAPPING register in the example.

Fig. 11-3 Memory Expansion Example (for reference)



* μ PD27C64: 4K bytes only are used.

Fig. 11-4 MEMORY MAPPING Register Setting



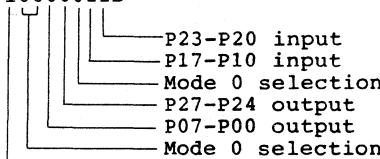
11.1.3 Peripheral device connection example

The uPD78C14/78C14A/78C12A/78C11/78C11A bus system is of the uPD8085 type (the data bus and the low-order eight bits of the address bus are multiplexed). This enables connection of most of uPD8085 peripheral devices.

To connect peripheral devices, the uPD78C14/78C14A/78C12A/78C11/78C11A does not contain I/O address space, thus all must be made memory mapped I/O. Here, connection of typical peripheral devices is shown.

Fig. 11-5 shows a configuration example where external memory and parallel interface unit (uPD71055) are connected to the uPD78C14/78C14A/78C12A/78C11/78C11A. Assume that the 4K-byte expansion mode is selected (see Fig. 11-4) and the expansion area starts at 4000H/1000H. The memory maps are as shown in Fig. 11-6 to 11-8.

A control program example for the uPD71055 is given below:

PPIST : LXI	H, 1C03H; Set Base Address
MVI	A, 10000001B
	
STAX	H - ; Set Control Word (1C03H)
MVI	A, 0F0H
STAX	H ; Port2 0F0H Output (1C02H)
MVI	A, 0C3H
MVI	L, 00H
STAX	H ; Port0 0C3H Output (2C00H)
:	
:	

Remarks: For the uPD78C14/78C14A, give LXI H, 4C03H.

Fig. 11-5 uPD71055 Connection Diagram (for reference)

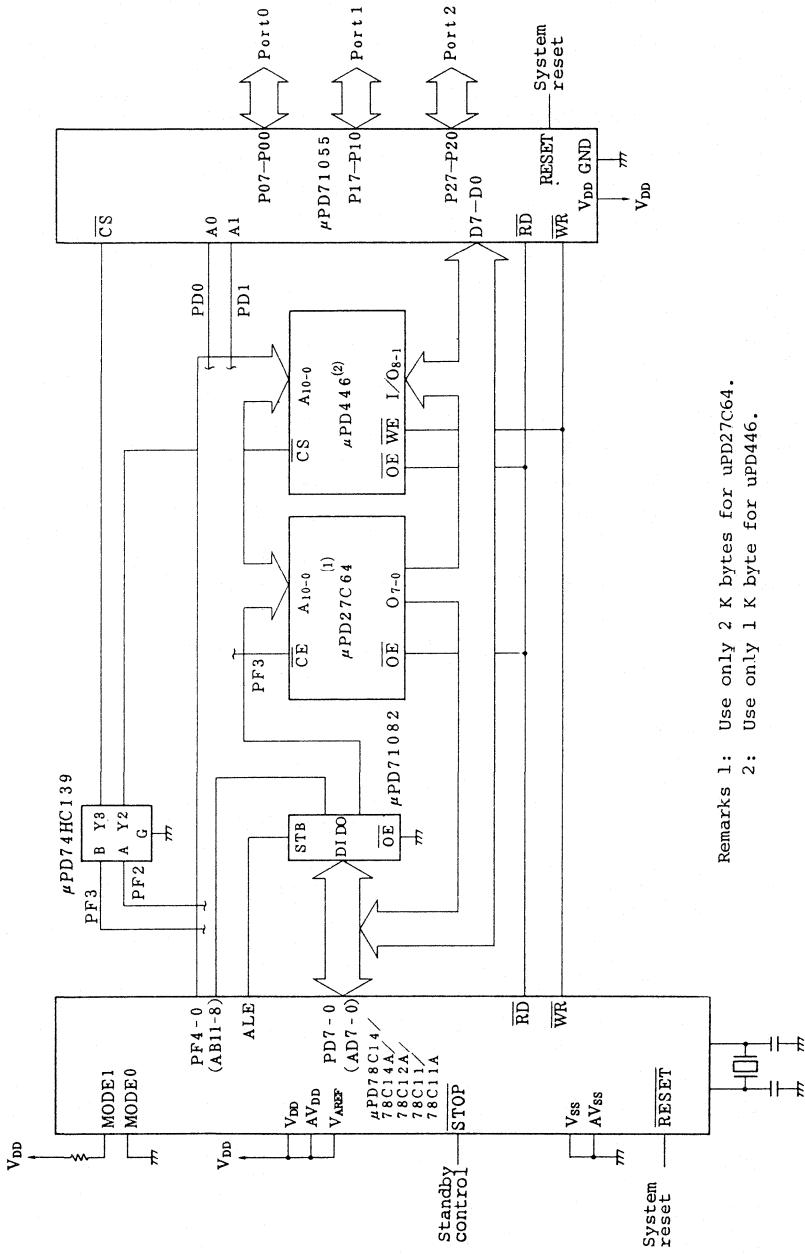


Fig. 11-6 Memory Map (uPD78C14/78C14A)

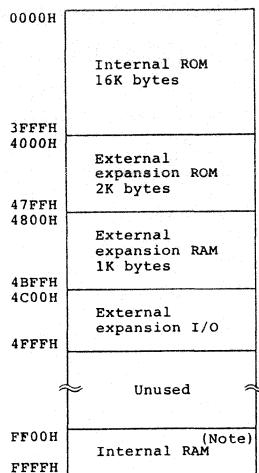
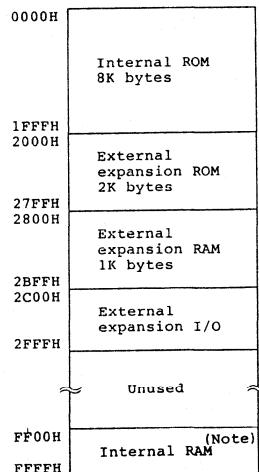
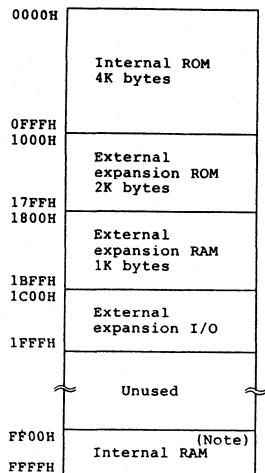


Fig. 11-7 Memory Map (uPD78C12A)



Note: When RAE bit of MM register is only set to 1, the internal RAM is usable.

Fig. 11-8 Memory Map (uPD78C11/78C11A)



Note: When RAE bit of MM register is only set to 1, the internal RAM is usable.

11.2 uPD78C10/78C10A External Device Access

Since the uPD78C10/78C10A does not contain internal ROM, external device (program memory, data memory, or peripheral device) can be placed in the external 64K-byte area (addresses 0000H-FEFFFH) in addition to internal RAM. The address space of the external device can be selected among the 4K-byte area (addresses 0000H-0FFFH), 16K-byte area (addresses 0000H-3FFFH), and 64K-byte area (addresses 0000H-FEFFFH) by setting the MODE0 and MODE1 pins.

Operation mode	Control pins		External address space	Internal RAM
	MODE1	MODE0		
4K-byte access	0	0	4K bytes (addresses 0000H-0FFFH)	Addresses FFOOH-FFFFH
16K-byte access	0	1	16K bytes (addresses 0000H-3FFFH)	Addresses FFOOH-FFFFH
64K-byte access	1	1	64K bytes (addresses 0000H-FEFFFH)	Addresses FFOOH-FFFFH

The external device is accessed by using the RD, WR, and ALE signals with the PD7-PD0 pins as the multiplexed address/data bus (AD7-AD0) and the PF7-PF0 pins as the address bus (AB15-AB8). To access the external device in the 4K-byte or 16K-byte area, the PF7-PF0 pins not used as address lines can be used as general purpose input/output port pins.

The external address space size is selected by setting the MODE0 and MODE1 pins.

11.2.1 MM register setting

Set the low-order three bits of the uPD78C10/78C10A MM register to 0. The RAE bit is used to specify whether or not internal RAM access is enabled. When internal RAM is not used and the area is used for external memory, set the RAE bit to 0 to disable internal RAM access.

If the RESET signal is input during the normal operation, the RAE bit status at the time is retained. However, the RAE bit status becomes undefined when power on reset. Initialize the RAE bit by executing an instruction.

Fig. 11-9 MM Register Format

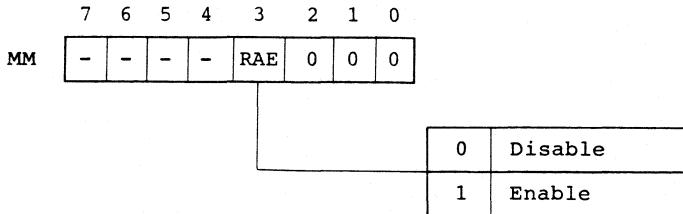
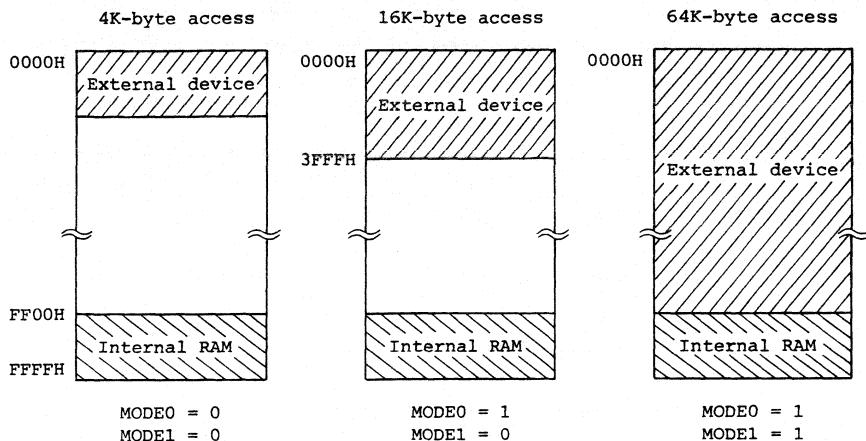


Fig. 11-10 uPD78C10/78C10A Address Space



Caution 1: Do not execute an instruction for port D or F during the 64K-byte access mode. Unexpected operation may be caused.

Caution 2: Do not use a program which changes the port F input/output mode dynamically because the program cannot be emulated by an emulator.

Caution 3: Do not execute an output instruction to port D or F during the 64K-byte access mode; if it is executed, a WR pulse is output.

Caution 4: Even if the RAE bit is not initialized by executing an instruction, normal operation may be performed on an emulator.

Caution 5: Apparently, the memory space can be increased 256 bytes by executing a program while the RAE bit is being rewritten. However, do not perform this operation because it cannot be emulated by an emulator.

11.3 Timing

Fig. 11-11 to 11-13 show the uPD78C11 operation timings. Three oscillator frequency cycles (from rising edge to falling edge) are defined to be one state which is represented by Tn.

One machine cycle of normal read or write operation consists of three states (nine clock cycles); four states (12 clock cycles) are required to fetch an OP (operation) code.

A wait state (TW) cannot be inserted.

(1) OP code fetch timing (see Fig. 11-11)

This is the timing to fetch the OP code of every instruction; it consists of four states T1-T4. The two states T1 and T2 are used to read program memory and T3 and T4 are used for internal processing (decode).

The high-order address signals of an external memory reference address are output to AB15-AB8 (PF7-PF0) from the beginning of T1 to the end of T4.

The low-order eight bits of the external memory reference address are output to AD7-AD0 (PD7-PD0) used as the multiplexed address/data bus during T1. After this, AD7-AD0 become high impedance. Address information on the AD7-AD0 bus lines is temporarily output and must be latched by the external device. The uCOM-87AD supplies the special timing signal ALE to latch AD7-AD0. The ALE signal is output in T1 of each machine cycle. The RD signal which is low is output from intermediate point of T1 to the beginning of T4.

(2) External device read timing (see Fig. 11-12)

The external device read timing is a data read machine cycle when an external device reference instruction is executed; it consists of T1 to T3. The AB15-AB8 (PF7-PF0), AD7-AD0 (PD7-PD0), and ALE timings are the same as in (1) above except that T4 does not exist. The \overline{RD} signal which is low is output from intermediate point of T1 to the beginning of T3.

(3) External device write timing (see Fig. 11-13)

The external device write timing is a data write machine cycle when an external device reference instruction is executed; it consists of three states T1 to T3.

The address output (AB15-AB8 and AD7-AD0) and ALE signal are the same as in the read timing machine cycle. Write data is output to AD7-AD0 from the beginning of T2 to the end of T3. To enable a write into the addressed device, the \overline{WR} signal which is low is output from intermediate point of T1 to the beginning of T3.

When PD7-PD0 are set to the multiplexed address/data bus (AD7-AD0) and PF7-PF0 are set to the address bus (AB15-AB8), both the \overline{RD} and \overline{WR} signals go high in a machine cycle in which an external device is not accessed. However, the ALE signal is output and the internal address bus contents are output to port D, F as they are.

Fig. 11-11 OP Code Fetch Timing

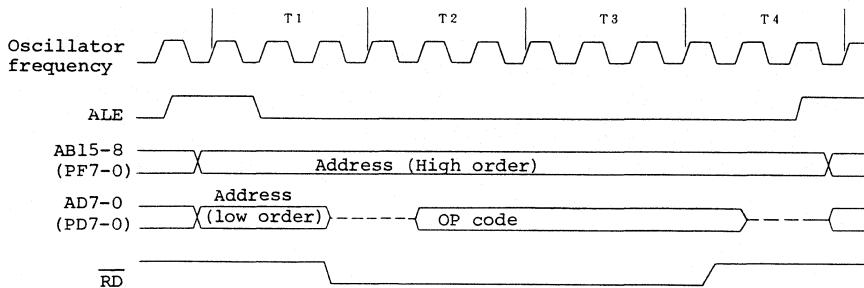


Fig. 11-12 External Device Read Timing

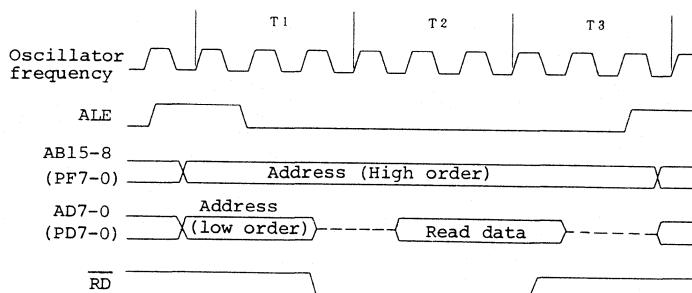
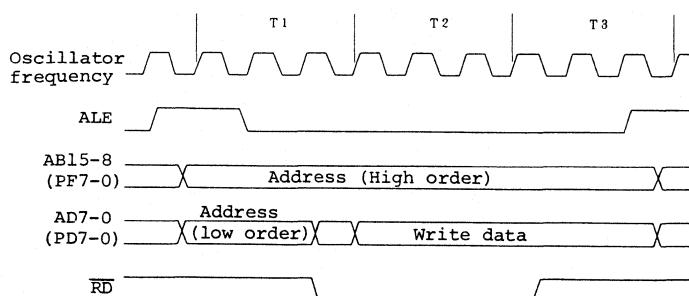


Fig. 11-13 External Device Write Timing



CHAPTER 12 INTERNAL EPROM ACCESS (uPD75CP14 ONLY)

The uPD75CP14 contains internal 16K-byte EPROM. The internal EPROM access range can be selected among the following by using the MEMORY MAPPING register MM6 and MM7 bits:

- . 4K-byte mode: The address area of 0000H-0FFFH is accessed
(uPD78C11/78C11A mode).
- . 8K-byte mode: The address area of 0000H-1FFFH is accessed (uPD78C12A mode).
- . 16K-byte mode: The address area of 0000H-3FFFH is accessed (uPD78C14/78C14A mode).

Fig. 12-1 shows the uPD78CP14 MEMORY MAPPING register format. The MEMORY MAPPING register bits are described below:

(1) MM0-MM2 bits

The MM0-MM2 bits are used to control port or expansion mode selection for PD6-PD0 and PF7-PF0.

See 11.11 for details.

(2) MM3 bit (RAE)

The MM3 bit is used to specify whether internal RAM access is enabled (RAE=1) or not (RAE=0).

See 11.11 for details.

(3) MM6 and MM7 bits

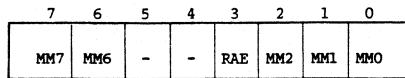
The MM6 and MM7 bits are used to select the internal EPROM access range.

When STOP or RESET is input, the MM6 and MM7 bits are reset and the 16K-byte mode (uPD78C14 mode) is selected.

The bits are significant only for the uPD78CP14 and uPD78CG14 (Note). Even if data is written into the bits on the uPD78C11/78C11A, uPD78C12A, or uPD78C14/78C14A, the CPU ignores it. Thus, programs developed on the uPD78CP14 or uPD78CG14 can be transported to mask ROM intact.

Note: The uPD78CG14 is introduced in Appendix A.

Fig. 12-1 MEMORY MAPPING Register Format



0	0	0	Port mode	Single chip	PD7-0=input mode PF7-0=port mode
0	0	1			PD7-0=output mode PF7-0=port mode
0	1	0	Expansion mode	256 bytes	PD7-0=expantion mode PF7-0=port mode
1	0	0		4K bytes	PD7-0=expantion mode PF3-0=expantion mode PF7-4=port mode
1	1	0		16K bytes	PD7-0=expantion mode PF3-0=expantion mode PF7-6=port mode
1	1	1		48K, 56K, or (Note) 60K bytes	PD7-0=expantion mode PF7-0=expantion mode

Note: The size is selected according to how the MM7 and MM6 bits are set.

Internal RAM access

0	Disable
1	Enable

Internal EPROM access

0	0	Internal EPROM area of addresses 000H-3FFFH is accessed (uPD78C14/78C14A mode)
0	1	Internal EPROM area of addresses 0000H-1FFFH is accessed (uPD78C12A mode)
1	0	Internal EPROM area of addresses 0000H-0FFFH is accessed (uPD78C11/78C11A mode)
1	1	Undefined.

CHAPTER 13 INSTRUCTION SET**13.1 Operand Identifiers and Description**

Enter operand in the operand field of each instruction according to the description for the operand identifier of the instruction. Select one of the entries in the description. Capital alphanumeric characters and + and - symbols are keywords that must be entered exactly as shown.

As immediate data, enter proper numeric value or label.

Identifier	Description
r r1 r2	V, A, B, C, D, E, H, L EAH, EAL, B, C, D, E, H, L A, B, C
sr srl sr2 sr3 sr4	PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, SML, EOM, ETMM, TMM, MM, MCC, MA, MB, MC, MF, TXB, TMO, TM1, ZCM PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, TMM, TXB, CRO, CR1, CR2, CR3 PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, TMM, ETMO, ETM1 ECNT, ECPT
rp rp1 rp2 rp3	SP, B, D, H V, B, D, H, EA SP, B, D, H, EA B, D, H,
rpa rpal rpa2 rpa3	B, D, H, D+, H+, D-, H-, B, D, H B, D, H, D+, H+, D-, H-, D+byte, H+A, H+B, H+EA, H+byte D, H, D++, H++, D+byte, H+A, H+B, H+EA, H+byte
wa	8-bit immediate data
word byte bit	16-bit immediate data 8-bit immediate data 3-bit immediate data
f	CY, HC, Z
irf	NM1*, FTO, FT1, F1, F2, FEO, FE1, FEIN, FAD, FSR, FST, ER, OV, AN4, AN5, AN6, AN7, SB

* NMI can also be described as FNMI.

Remarks:

1. sr~sr4 (special register)

2. rp~rp3 (retister pair)

4. F (flag)

PA :PORT A	ETMM:TIMER/EVENT
PB :PORT B	COUNTER MODE
PC :PORT C	EOM :TIMER/EVENT
PD :PORT D	COUNTER OUTPUT
PF :PORT F	MODE
MA :MODE A	ANM :A/D CHANNEL
MB :MODE B	MODE
MC :MODE C	CRO :A/D
MCC :MODE CONTROL C	CONVERSION
MF :MODE F	RESULT0~3
MM :MEMORY MAPPING CR3	
TMO :TIMER REG0	TXB :Tx BUFFER
TM1 :TIMER REG1	RXB :Rx BUFFER
TMM :TIMER MODE	SMH :SERIAL MODE
ETMO:TIMER/EVENT	High
COUNTER REG0	SML :SERIAL MODE
ETM1:TIMER/EVENT	Low
COUNTER REG1	MKH :MASK High
ECNT:TIMER/EVENT	MKL :MASK Low
COUNTER	XCM :XERO CROSS
UPCOUNTER	MODE
ECPT:TIMER/EVENT	
COUNTER CAPTURE	

SP:STACK POINTER
B :BC
D :DE
H :HL
V :VA
EA:EXTENDED ACCUMULATOR

CY:CARRY
HC:HALF CARRY
Z :ZERO

5. irf (interrupt flag)

3. rpa~rpa3
(rp addressing)

B :	(BC)
D :	(DE)
H :	(HL)
D+ :	(DE)+
H+ :	(HL)+
D- :	(DE)-
H- :	(HL)-
D++ :	(DE)++
H++ :	(HL)++
D+byte:	(DE+byte)
H+A :	(HL+A)
H+B :	(HL+B)
H+EA :	(HL+EA)
H+byte:	(HL+byte)

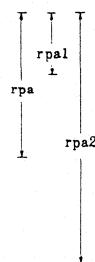
NMI :NMI INPUT
FT0 :INTFT0
FT1 :INTFT1
F1 :INTF1
F2 :INTF2
FE0 :INTFE0
FE1 :INTFE1
FEIN:INTFEIN
FAD :INTFAD
FSR :INTFSR
FST :INTFST
ER :ERROR
OV :OVERFLOW
AN4 :ANALOG
 to INPUT 4~7
AN7
SB :STANDBY

13.2 Explanation of Operation Code Symbols

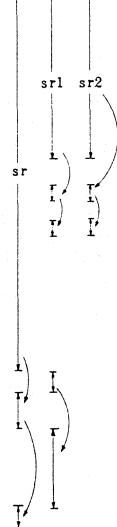
R ₂	R ₁	R ₀	reg
0	0	0	V
0	0	1	A
0	1	0	B
0	1	1	C
1	0	0	D
1	0	1	E
1	1	0	H
1	1	1	L

T ₂	T ₁	T ₀	reg
0	0	0	EAH
0	0	1	EAL
0	1	0	B
0	1	1	C
1	0	0	D
1	0	1	E
1	1	0	H
1	1	1	L

A ₃	A ₂	A ₁	A ₀	addressing
0	0	0	0	—
0	0	0	1	(BC)
0	0	1	0	(DE)
0	0	1	1	(HL)
0	1	0	0	(DE)+
0	1	0	1	(HL)+
0	1	1	0	(DE)-
0	1	1	1	(HL)-
1	0	1	1	(DE+byte)
1	1	0	0	(HL+A)
1	1	0	1	(HL+B)
1	1	1	0	(HL+EA)
1	1	1	1	(HL+byte)



S ₆	S ₅	S ₄	S ₃	S ₂	S ₁	S ₀	Special-reg
0	0	0	0	0	0	0	PA
0	0	0	0	0	1	0	PB
0	0	0	0	1	0	0	PC
0	0	0	0	1	1	0	PD
0	0	0	1	0	1	0	PF
0	0	0	1	1	0	0	MKH
0	0	0	1	1	1	0	MKL
0	0	1	0	0	0	0	ANM
0	0	1	0	0	1	0	SMH
0	0	1	0	1	0	0	SML
0	0	1	0	1	1	0	EOM
0	0	1	1	0	0	0	ETMM
0	0	1	1	0	0	1	TMM
0	1	0	0	0	0	0	MM
0	1	0	0	0	1	0	MCC
0	1	0	0	1	0	0	MA
0	1	0	0	1	1	0	MB
0	1	0	1	0	0	0	MC
0	1	0	1	1	1	0	MF
0	1	1	0	0	0	0	TXB
0	1	1	0	0	1	0	RXB
0	1	1	0	1	0	0	TMO
0	1	1	0	1	1	0	TM1
1	0	0	0	0	0	0	CR0
1	0	0	0	0	1	0	CR1
1	0	0	0	1	0	0	CR2
1	0	0	0	1	1	0	CR3
1	0	1	0	0	0	0	ZCM



C ₁	C ₂	C ₁	C ₀	addressing
0	0	1	0	(DE)
0	0	1	1	(HL)
0	1	0	0	(DE)++
0	1	0	1	(HL)++
1	0	1	1	(DE+byte)
1	1	0	0	(HL+A)
1	1	0	1	(HL+B)
1	1	1	0	(HL+EA)
1	1	1	1	(HL+byte)

I ₄	I ₃	I ₂	I ₁	I ₀	INTF
0	0	0	0	0	NMI
0	0	0	0	1	FT0
0	0	0	1	0	FT1
0	0	0	1	1	F1
0	0	1	0	0	F2
0	0	1	0	1	FE0
0	0	1	1	0	FE1
0	0	1	1	1	FEIN
0	1	0	0	0	FAD
0	1	0	0	1	FSR
0	1	0	1	0	FST
0	1	0	1	1	ER
0	1	1	0	0	OV
1	0	0	0	0	AN4
1	0	0	0	1	AN5
1	0	0	1	0	AN6
1	0	0	1	1	AN7
1	0	1	0	0	SB

U ₀	Special-reg
0	ETM0
1	ETM1

V ₀	Special-reg
0	ECNT
1	ECPT

P ₂	P ₁	P ₀	reg-pair
0	0	0	SP
0	0	1	BC
0	1	0	DE
0	1	1	HL
1	0	0	EA

Q ₂	Q ₁	Q ₀	reg-pair
0	0	0	VA
0	0	1	BC
0	1	0	DE
0	1	1	HL
1	0	0	EA

F ₂	F ₁	F ₀	フラグ
0	0	0	—
0	1	0	CY
0	1	1	HC
1	0	0	Z

13.3 Instruction addressing

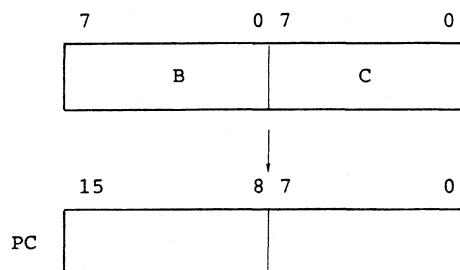
The instruction address is determined by the program counter (PC) contents. Normally, whenever one instruction is executed, PC is automatically incremented according to the number of the bytes of the fetched instruction (increment by one per byte). When an instruction involving a branch is executed, jump address is loaded into PC according to the addressing described below and a jump is made.

13.3.1 Register addressing

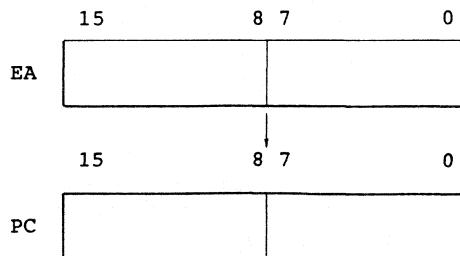
The contents of the BC register pair or expansion accumulator EA are loaded into PC and a jump is made. This is performed when the following instructions are executed:

JB

CALB



JEA

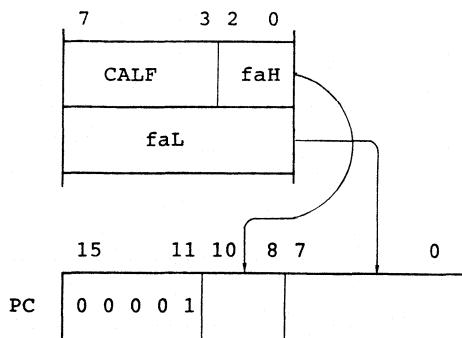
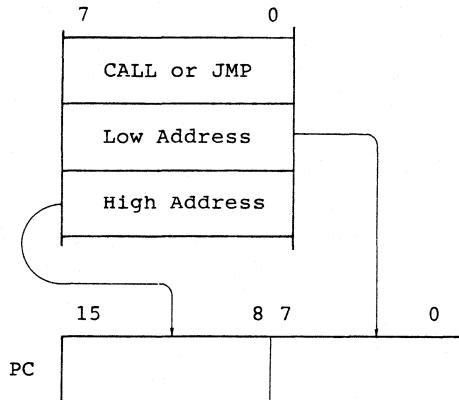


13.3.2 Immediate addressing

The immediate data of the second and third bytes of a given instruction is loaded into PC and a jump is made. This is performed when any of the following instruction is executed:

JMP word
CALL word
CALF word

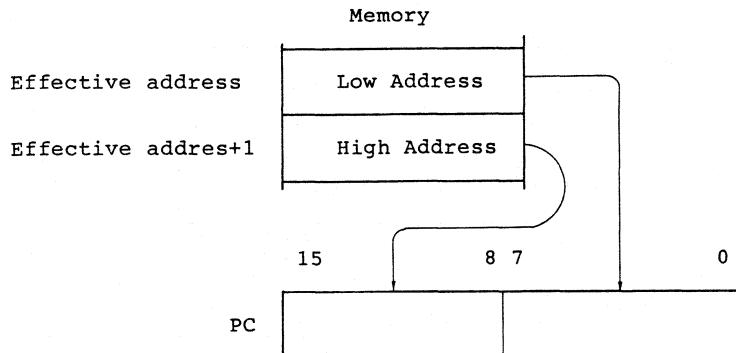
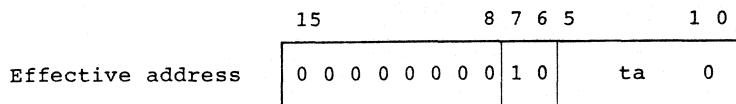
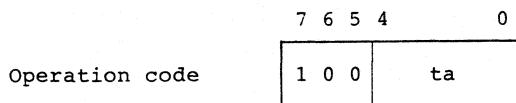
When the CALF instruction is executed, the immediate data of the low-order three bits of the first byte and the second byte is loaded into PC.



13.3.3 Direct addressing

The contents of the memory location addressed by the immediate data of the low-order five bits of operation code are loaded into PC and a jump is made. This is performed when the following instruction is executed:

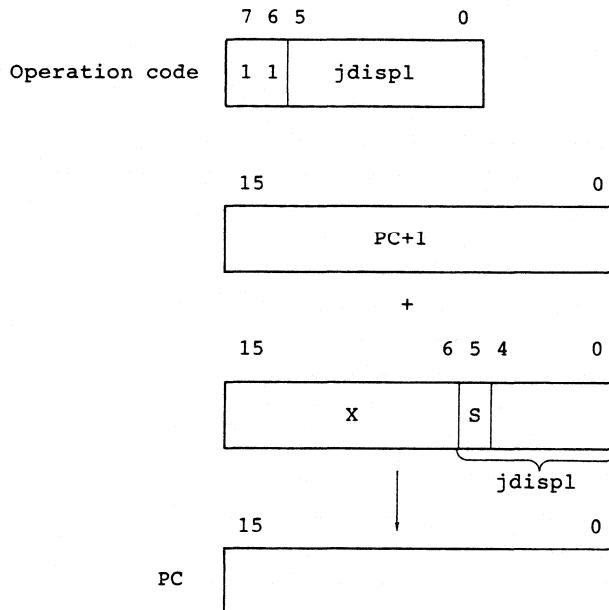
CALT word



13.3.4 Relative addressing

The result of adding the low-order 6-bit immediate data of a given operation code (displacement (jdispl)) to the top address of the next instruction is loaded into PC and a jump is made. The displacement is handled as signed two's complement data (-32 to +31) and bit 5 becomes a sign bit. This is performed when the following instruction is executed:

JR word

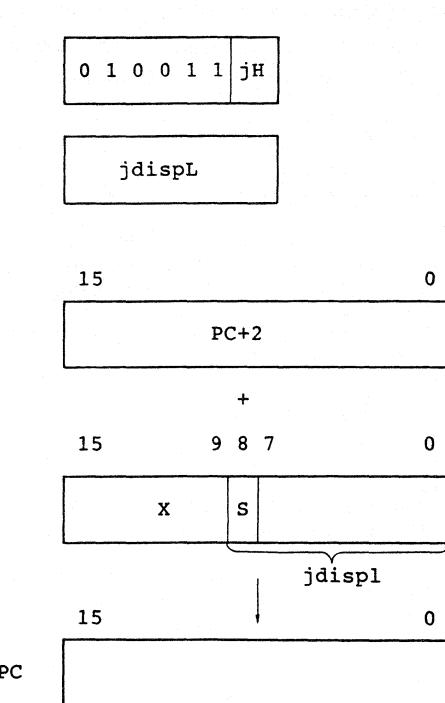


S=0: X=all 0
S=1: X=all 1

13.3.5 Extended relative addressing

The result of adding the 9-bit immediate data of a given operation code (displacement (jdisp)) to the top address of the next instruction is loaded into PC and a jump is made. The displacement is handled as signed two's complement data (-256 to +256) and bit 8 (bit 0 of the first byte of the operation code) becomes a sign bit. This is performed when the following instruction is executed:

JRE word



S=0: X=all 0
S=1: X=all 1

13.4 Operand Addressing

The addressing modes of registers and memory to be operated on in instruction execution are described below:

13.4.1 Register addressing

The register to be operated on is addressed by the contents of the register specification code such as R₂R₁R₀, T2T1T0, or S5S4S3S2S1S0 in a given instruction.

The register addressing is made when an instruction having any of the following operand identifiers is executed: (An 8-bit register of a 16-bit register pair is addressed.)

Identifier	Description
r	V, A, B, C, D, E, H, L
r1	EAH, EAL, B, C, D, E, H, L
r2	A, B, C
sr	PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, SML, EOM, ETMM, TMM, MM, MCC, MA, MB, MC, MF, TXB, TMO, TM1, ZCM
srl	PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EPM, TMM, TXB, CRO, CR1, CR2, CR3
sr2	PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, TMM, ETMO, ETM1
sr3	ETMO, ETM1
sr4	ECNT, ECPT
rp	SP, B, D, H
rp1	V, B, D, H, EA
rp2	SP, B, D, H, EA
rp3	B, D, H,
f	CY, HC, Z
irf	NM1*, FTO, FT1, F1, F2, FE0, FE1, FEIN, FAD, FSR, FST, ER, OV, AN4, AN5, AN6, AN7, SB

* NMI can also be described as FNMI.

Example 1: MOV rl, A

	7	6	5	4	3	2	1	0
Operation code	0	0	0	1	1	T ₂	T ₁	T ₀

To specify the E register in rl, enter the following:(the semicolon (;) is a separator between the instruction and a comment which does not affect instruction operation.)

MOVE E, A;E + A

The operation code of this instruction is as follows:

Operation code	0	0	0	1	1	1	0	1
----------------	---	---	---	---	---	---	---	---

Example 2: DCX rp

Operation code	0	0	P ₁	P ₀	0	0	1	1
----------------	---	---	----------------	----------------	---	---	---	---

To specify the HL register pair in rp, enter the following:

DCX H;HL + HL-1

The operation code of this instruction is as follows:

Operation code	0	0	1	1	0	0	1	1
----------------	---	---	---	---	---	---	---	---

13.4.2 Register indirect addressing

The memory location to be operated on is addressed by the contents of the register pair indicated by the register pair specification code A3A2A1A0 or C3C2C1C0 in a given instruction.

The register indirect addressing is made when an instruction having any of the following operand identifiers is executed: (Auto increment addressing, auto decrement addressing, double auto increment addressing, base addressing, and base index addressing are explained in 13.4.3 to 13.4.7.)

Identifier Description

rpa	B, D, H, D+, H+, D-, H-,
rpal	B, D, H
rpa2	B, D, H, D+, H+, D-, H-, D+byte, H+A, H+B, H+EA, H+byte
rpa3	D, H, D++, H++, D+byte, H+A, H+B, H+EA, H+byte

Example 1: LDAX rpa2

Operation code	A ₃	0	1	0	1	A ₂	A ₁	A ₀
----------------	----------------	---	---	---	---	----------------	----------------	----------------

To specify BC register pair in rpa2, enter the following:

LDAX B;A + (BC)

The operation code of this instruciton is as follows:

Operation code	0	0	1	0	1	0	0	1
----------------	---	---	---	---	---	---	---	---

13.4.3 Auto increment addressing

The auto increment addressing is a special mode of the register indirect addressing using the HL and DE register pairs. The memory location to be operated on is addressed by the contents of the register pair indicated by the addressing specification code A3A2A1A0 in a given instruction, then the register pair contents are automatically incremented by one for the next addressing.

The auto increment addressing is made when an instruction having any of the following operand identifiers is executed:

Identifier	Description
------------	-------------

Identifier	Description
rpa	D+, H+

Example 1: STAX rpa2

Operation code

A ₃	0	1	1	1	A ₂	A ₁	A ₀
----------------	---	---	---	---	----------------	----------------	----------------

To specify DE register pair auto increment mode in rpa2, enter the following:

STAX D+;(DE) + A, DE ← DE+1

The operation code of this instruction is as follows:

Operation code

0	0	1	1	1	1	0	0
---	---	---	---	---	---	---	---

Example 2: When BLOCK instruction is executed

No operand specification is made. When the BLOCK instruction is executed, the HL and DE register pairs are automatically selected for the source and destination address registers. After data is transferred from the source address to the destination address, each of the HL and DE register pairs is automatically incremented by one.

BLOCK : (DE) ← (HL), DE ← DE+1, HL ← HL+L

Example 3: After return or pop instruction is executed

No operand specification is made. When a return or pop instruction to restore data saved in a stack area is executed, the stack pointer (SP) is automatically incremented.

RET ;PC_L ← (SP), PC_H ← (SP+1), SP ← SP+2

13.4.4 Auto decrement addressing

The auto decrement addressing is a special mode of the register indirect addressing using the HL and DE register indirect addressing using the HL and DE register pairs. After the memory location to be operated on is addressed by the contents of the register pair indicated by the addressing code A3A2A1A0 in a given instruction, the register pair contents are automatically decremented by one for the next addressing.

The auto decrement addressing is made when an instruction having any of the following operand identifiers is executed:

Identifier Description

rpa D-, H-
rpa2 D-, H-

Example 1: ADDX rpa

Operation code	0	1	1	1	0	0	0	0
	1	1	0	0	0	A_2	A_1	A_0

To specify HL register pair auto decrement mode in rpa,
enter the following:

ADDX H-;A + A+(HL), HL + HL-1

The operation code of this instruction is as follows:

Operation code	0	1	1	1	0	0	0	0
	1	1	0	0	0	1	1	1

Exapmle 2: When an interrupt occurs or a call or push
instruction is executed

No operand specification is made. When an interrupt occurs
of a call or push instruction is executed to save the
register contents, etc., in a stack area, the attack pointer
(SP) is automatically decremented.

SOFTI : (SP-1) + PSW, (SP-2) + PC+1_H
(SP-3) + PC+1_L, PC + 0060H

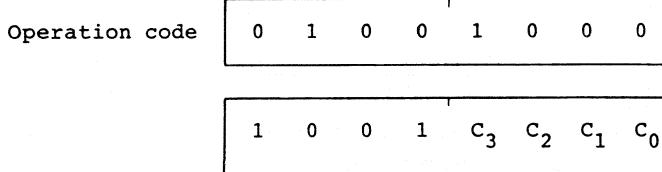
13.4.5 Double auto increment addressing

The double auto increment addressing is a special mode of the register indirect addressing using the HL and DE register pairs. It is useful for 16-bit data transfer between the expansion accumulator (EA) and memory. After the memory location to be operated on is addressed by the contents of the register pair indicated by the addressing code C3C2C1C0 in a given instruction, the register pair contents are automatically incremented by two for the next addressing.

The double auto increment addressing is made when an instruction having the following operand identifier is executed:

Identifier	Description
rpa3	D++, H++

Example 1: STEAX rpa3



To specify HL register pair double auto increment mode in rpa3, enter the following:

STEAX H++; (HL) ← EAL, (HL+1) ← EAH, HL ← HL+2

The operation code of this instruction is as follows:

Operation code

0 1 0 0 1 0 0 0

1 0 0 1 0 1 0 1

13.4.6 Base addressing

The base addressing is a special mode of the register indirect addressing using the HL and DE register pairs. The memory location to be operated on is addressed by the sum of the contents of the register pair (base register) indicated by the addressing code A3A2A1A0 or C3C2C1C0 in a given instruction and the operand immediate data (displacement). The base addressing is made when an instruction having any of the following operand identifiers is executed: (The immediate data (displacement) is handled as a nonnegative number.)

Identifier	Description
rpa2	D+byte, H+byte
rpa3	D+byte, H+byte

Example 1: STAX rpa2

A₃ 0 1 1 1 A₂ A₁ A₀

Data

To specify the base addressing of the sum of the HL register pair contents and 10H in rpa2, enter the following:

STAX H+10H; (HL+10H) ← A

The operation code of this instruction is as follows:

Operation code

1	0	1	1	1	1	1	1
---	---	---	---	---	---	---	---

0	0	0	1	0	0	0	0
---	---	---	---	---	---	---	---

13.4.7 Base index addressing

The base index addressing is a special mode of the register indirect addressing using the HL and DE register pairs. The memory location to be operated on is addressed by the sum of the contents of the register pair (base register) indicated by the addressing code A3A2A1A0 or C3C2C1C0 in a given instruction and the contents of register A, B, or EA. The base index addressing is made when an instruction having any of the following operand identifiers is executed: (Data in register A or B is handled as a nonnegative number.)

Identifier Description

rpa2 H+A, H+B, H+EA
rpa3 H+A, H+B, H+EA

Example 1: LDAX rpa2

Operation code

A ₃	0	1	0	1	A ₂	A ₁	A ₀
----------------	---	---	---	---	----------------	----------------	----------------

To specify the base index addressing of the sum of the HL register pair contents and the B register contents in rpa², enter the following:

LDAX H+B;A ← (HL+B)

The operation code of this instruction is as follows:

Operation code

1	0	1	0	1	1	0	1
---	---	---	---	---	---	---	---

13.4.8 Working register addressing

A working register in memory area to be operated on is selected by using the working register vector register (V) as the high-order eight bits of the address and the 8-bit immediate data of a given instruction as the low-order eight bits. The working register addressing is a mixture of the register indirect addressing using the V register and the direct addressing using immediate data wa.

The working register addressing is made when an instruction having the following operand identifier is executed:

Identifier Description

wa Label, numeric value of up to eight bits

Example 1: DCRW wa

Operation code

0	0	1	1	0	0	0	0
---	---	---	---	---	---	---	---

offset

To specify 77H in wa, enter the following:

DCRW 77H

The operation code of this instruction is as follows:

Operation code

0	0	1	1	0	0	0	0
---	---	---	---	---	---	---	---

0	1	1	1	0	1	1	1
---	---	---	---	---	---	---	---

Assuming that the V register contains 20H, operand address 2077H is generated and the contents of the working register at address 2077H are decremented by one.

13.4.9 Accumulator indirect addressing

The accumulator indirect addressing is a special mode of the register indirect addressing. The contents of the memory location addressed by PC+3+A are loaded into the C register, and the contents of the memory location addressed by PC+3+A+1 are loaded into the B register.

The accumulator indirect addressing is made when the TABLE instruction is executed.

Example 1: Assume that the accumulator contains 0 and PC contains 100H.

TABLE ;C ← (103H), B ← (104H)

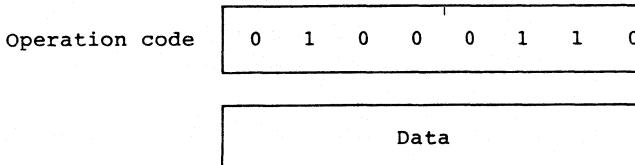
13.4.10 Immediate addressing

The 1-byte operand data to be operated on is contained in a given operation code. The immediate addressing is made when an instruction having the following operand identifier is executed:

Identifier Description

byte Label, numeric value of up to eight bits

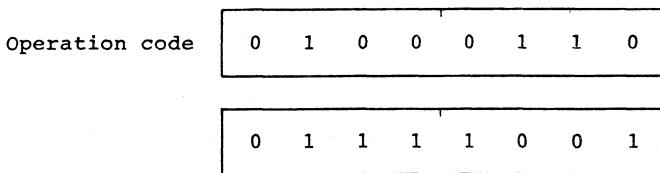
Example 1: ADI A, byte



to specify 79H in byte, enter the following:

ADI A, 79H;A ← A+79H

The operation code of this instruction is as follows:



13.4.11 Extended immediate addressing

The 2-byte operand data to be operated on is contained in a given operation code. The extended immediate addressing is made when an instruction having the following operand identifier is executed:

Identifier Description

word Label, numeric value of up to 16 bits

Example 1: LXI rp2, word

Operation code 0 P₂ P₁ P₀ 0 1 0 0

Low byte

High byte

To specify HL in rp2 and 3F54H in word, enter the following:

LXI H, 3F54H;HL ← 3F54H

The operation code of this instruction is as follows:

Operation code 0 0 1 1 0 1 0 0

0 1 0 1 0 1 0 0

0 0 1 1 1 1 1 1

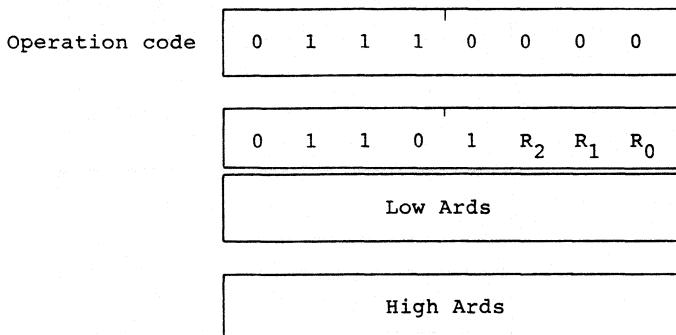
13.4.12 Direct addressing

The memory location to be operated on is addressed by the immediate data in a given instruction. The direct addressing is made when an instruction having the following operand identifier is executed:

Identifier Description

Word Label, numeric value of up to 16 bits

Example 1: MOV r, word



To specify B register in r and EFFFH in word, enter the following:

MOV B, 0EEFFH;

The operation code of this instruction is as follows:

Operation code

0	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

0	1	1	0	1	0	1	0
---	---	---	---	---	---	---	---

1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---

1	1	1	0	1	1	1	0
---	---	---	---	---	---	---	---

Example 2: SDED word

Operation code

0	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

0	0	1	0	1	1	1	0
---	---	---	---	---	---	---	---

Low Ards

High Ards

To specify label DST in word, enter the following:

SDED DST

Assuming that DST is 4000H, the operation code fo. this instruction is as follows:

Operation code	0	1	1	1	0	0	0	0
	0	0	1	0	1	1	1	0
	0	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0

13.5 Number of States Required when Instruction is Skipped

The numeric value enclosed in parentheses under 3) Number of states in 13.6 to follow is the number of idle states consumed when the instruction is skipped.

The number of idle states when the instruction is skipped is 4 for each OP code byte and 3 for immediate data.

Example: MVI sr2, byte (3-byte instruction)

0	1	1	0	0	1	0	0
S ₃	0	0	0	0	S ₂	S ₁	S ₀
Data							

Since the first and second bytes are OP code, the number of idle states is 4. Since the third byte is immediate data, the number of idle states is 3. Thus, the number of idle states consumed when the instruction is skipped is $4+4+3=11$.

13.6 Explanation of Instructions

13.6.1 8-bit data transfer instructions

MOV r1, A (Move A to Register)

- 1) Operation code:

0	0	0	1	1	T ₂	T ₁	T ₀
---	---	---	---	---	----------------	----------------	----------------
- 2) Number of bytes: 1
- 3) Number of states: 4 (4)
- 4) Function: r1 ← A

Transfer the accumulator contents to register r1 (EAH, EAL, B, C, D, E, H, or L) addressed by T2T1T0 (0-7). If EAH is specified in r1, the accumulator contents are transferred to the high-order eight bits of the expansion accumulator; if EAL is specified, the accumulator contents are transferred to the low-order eight bits of the expansion accumulator.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0
- 6) Example: MOV B, A; Transfer A to B.

MOV A, r1 (Move Register to A)

- 1) Operation code:

0	0	0	0	1	T ₂	T ₁	T ₀
---	---	---	---	---	----------------	----------------	----------------
- 2) Number of bytes: 1
- 3) Number of states: 4 (4)
- 4) Function: A ← r1

Transfer the contents of register r1 (EAH, EAL, B, C, D, E, H, or L) addressed by T2T1T0 (0-7) to the accumulator. If EAH is specified in r1, the contents of the high-order eight bits of the expansion accumulator are transferred to the accumulator if EAL is specified, the contents of the low-order eight bits of the expansion accumulator are transferred to the accumulator.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0
- 6) Example: MOV A, C; Transfer C to A.

MOV sr, A (Move A to Special Register)

1) Operation code	<table border="1"><tr><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr></table>	0	1	0	0	1	1	0	1
0	1	0	0	1	1	0	1		
	<table border="1"><tr><td>1</td><td>1</td><td>S₅</td><td>S₄</td><td>S₃</td><td>S₂</td><td>S₁</td><td>S₀</td></tr></table>	1	1	S ₅	S ₄	S ₃	S ₂	S ₁	S ₀
1	1	S ₅	S ₄	S ₃	S ₂	S ₁	S ₀		

- 2) Number of bytes: 2
- 3) Number of states: 10 (7)
- 4) Function: sr ← A

Transfer the accumulator contents to the special register sr (PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, SML, EOM, ETMM, TMM, MM, MCC, MA, MB, MC, MF, TXB, TM0, TM1, or ZCM) addressed by S5S4S3S2S1S0 (0-3, 5-D, 10-14, 16, 18, 1A, 1B, 28).

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0
- 6) Example: MOV PA, A; Transfer A to port A latch.

MOV A, srl (Move Special Register to A)

- 1) Operation code

0	1	0	0	1	1	0	1
---	---	---	---	---	---	---	---

1	1	S ₅	S ₄	S ₃	S ₂	S ₁	S ₀
---	---	----------------	----------------	----------------	----------------	----------------	----------------

- 2) Number of bytes: 2

- 3) Number of states: 10 (7)

- 4) Function: A ← srl

Transfer the contents to the special register srl (PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, TMM, RXB, CR0, CR1, CR2, or CR3) addressed by S5S4S3S2S1S0 (0-3, 5-9, B, D, 19,20-23) to the accumulator.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0

- 6) Example: MOV A, TMM; Transfer the timer mode register contents to the accumulator.

MOV r, word (Move Memory to Register)

- 1) Operation code

0	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

0	1	1	0	1	R ₂	R ₁	R ₀
---	---	---	---	---	----------------	----------------	----------------

L.Adrs

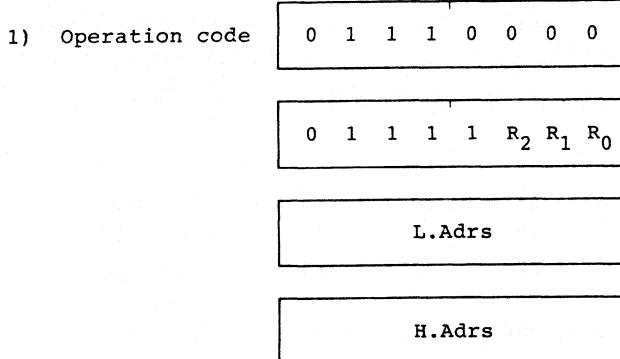
H.Adrs

- 2) Number of bytes: 4
- 3) Number of states: 17 (14)
- 4) Function: $r \leftarrow (\text{word})$

Transfer the contents of the memory location addressed by the third byte (Low Address) and the fourth byte (High Address) to register r (V, A, B, C, D, E, H, or L) addressed by $R_2 R_1 R_0$ (0-7).

- 5) Affected Flags: SK $\leftarrow 0$, L1 $\leftarrow 0$, L0 $\leftarrow 0$
- 6) Example: MOV B, 89ABH; Transfer the contents of address 89ABH to B.

MOV word, r (Move Register to Memory)



- 2) Number of bytes: 4
- 3) Number of states: 17 (14)
- 4) Function: $(\text{word}) \leftarrow r$

Transfer the contents of register r (V, A, B, C, D, E, H, or L) addressed by $R_2 R_1 R_0$ (0-7) to the memory location addressed by the third byte (Low Address) and the fourth byte (High Address).

- 5) Affected Flags: SK \leftarrow 0, L1 \leftarrow 0, L0 \leftarrow 0
- 6) Example: MOV EXAM, A; Transfer A to the memory location address by label EXAM.

MVI r, byte (Move Immediate to Register)

- 1) Operation code

0	1	1	0	1	R ₂	R ₁	R ₀
---	---	---	---	---	----------------	----------------	----------------

Data

- 2) Number of bytes: 2
- 3) Number of states: 7 (7)
- 4) Function: r \leftarrow byte

Transfer the immediate data of the second byte (Data) to register r (V, A, B, C, D, E, H, or L) addressed by R₂R₁R₀ (0-7). If A or L is specified in r, the string effect is made.

- 5) Affected Flags: SK \leftarrow 0, L1 \leftarrow 1, L0 \leftarrow 0 (when r=A)
SK \leftarrow 0, L1 \leftarrow 0, L0 \leftarrow 1 (when r=L)
SK \leftarrow 0, L1 \leftarrow 0, L0 \leftarrow 0 (other than the above)
- 6) Example: MVI D, 0AFH; Load AFH into the D register.

MVI sr2, byte (Move Immediate to Special Register)

- 1) Operation code

0	1	1	0	0	1	0	0
---	---	---	---	---	---	---	---

S₃ 0 0 0 0 S₂ S₁ S₀

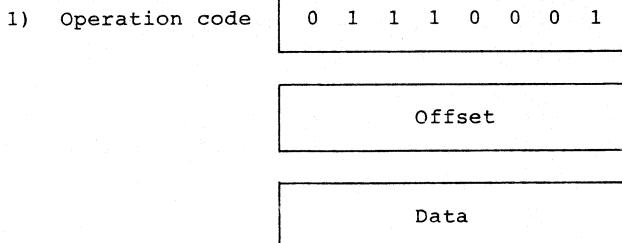
Data

- 2) Number of bytes: 3
- 3) Number of states: 14 (11)
- 4) Function: sr2 + byte

Transfer the immediate data of the third byte to special register sr2 (PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, or TMM) addressed by $S_2S_1S_0$ (0-3, 5-9, B, D).

- 5) Affected Flags: SK ← 0, L1 ← 0, L0 ← 0 (when r=A)
- 6) Example:

MVIW wa, byte (Move Immediate to Working Register)



- 2) Number of bytes: 3
- 3) Number of states: 14 (11)
- 4) Function: sr2 + byte

Transfer the immediate data of the third byte (Data) to the working register addressed by the V register (The high-order eight bits of the memory address) and the second byte (the low-order eight bits of the memory address.)

- 5) Affected Flags: SK ← 0, L1 ← 0, L0 ← 0
- 6) Example: MVI V, 40H
MVIW 00H, 20H; Load 20H into the working register at address 4000H.

MVIX rp1, byte (Move Immediate to Memory address by Register Pair)

- 1) Operation code

0	1	0	0	1	0	A ₁	A ₀
---	---	---	---	---	---	----------------	----------------

- 2) Number of bytes: 2
3) Number of states: 10 (7)
4) Function: (rp1) + byte

Transfer the immediate data of the second byte (Data) to the memory location addressed by register pair rp1 (BC, DE, or HL) indicated by A1A0 (1-3).

- 5) Affected flags: SK + 0, L1 + 0, L0 + 0
6) Example: MVIX B, 00H; Load 0 into the memory location addressed by BC register pair.

STAW wa (Store A to Working Register)

- 1) Operation code

0	1	1	0	0	0	1	1
---	---	---	---	---	---	---	---

- 2) Number of bytes: 2
3) Number of states: 10 (7)
4) Function: (V·wa) + A

Store the accumulator contents in the working register addressed by the V register (the high-order eight bits of the memory address0 and the second byte (the low-order eight bits of the memory address).

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0
- 6) Example: MVI V, 0EEH
STAW 0FFH; Store A in address EFFFH.

LDAW wa (Load A with Working Register)

- 1) Operation code

0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---
- 2) Number of bytes: 2
- 3) Number of states: 10 (7)
- 4) Function: A ← (V·wa)

Load the contents of the working register addressed by the V register (the high-order eighth bits of the memory address) and the second byte (the low-order eight bits of the memory address) into the accumulator.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0
- 6) Example:

STAX rpa2 (Store A to Memory addressed by Register Pair)

- 1) Operation code

A ₃	0	1	1	1	A ₂	A ₁	A ₀
----------------	---	---	---	---	----------------	----------------	----------------
- 2) Number of bytes: 1 or 2 (Note)
- 3) Number of states: 7 (4) or 13 (7) (Note)
- 4) Function: (rpa2) ← A

Store the accumulator contents in the memory location addressed by register pair rpa2 (BC, DE, HL, DE+, HL+, DE-, HL-, DE+byte, HL+A, HL+B, HL+EA, or HL+byte) indicated by A3A2A1A0 (1-7, B-F). If the auto increment of decrement mode is specified, the register pair (ADE or HL) contents are automatically incremented or decremented by one after the accumulator contents are stored.

If DE+byte or HL+byte is specified in rpa2, the memory location is addressed by the addition result of the DE or HL contents and the second byte of the instruction (Data). If HL+A, HL+B, or HL+EA is specified in rpa2, the memory location is addressed by the addition result of the HL contents and the contents of register A, B, or EA.

- 5) Affected flags: SK 0, L1 0, L0 0
- 6) Example: LXI D,4000H;DE 4000H

STAX D+ ;(4000H) A, DE 4001H

STAX D+ ;(4011H) A, DE 4001H

Store A in addresses 4000H and 4011H.

- 7) Note: The number of bytes and the number of states vary depending on the rpa2 specification as listed below:

rpa2	B	D	H	D+	H+	D-	H-	D+byte	H+A	H+B	H+EA	H+byte
Number of bytes				1				2		1		2
Number of states	7 (4)					13 (7)						

LDAX rpa2 (Load A with Memory addressed by Register Pair)

- 1) Operation code

A ₃	0	1	0	1	A ₂	A ₁	A ₀
----------------	---	---	---	---	----------------	----------------	----------------

- 2) Number of bytes: 1 or 2 (Note)

- 3) Number of states: 7 (4) or 13 (7) (Note)

- 4) Function: A ← (rpa2)

Data

Load the contents of the memory location addressed by register pair rpa2 (BC, DE, HL, DE+, HL+, DE-, HL-, DE+byte, HL+A, HL+B, HL+EA, or HL+byte) indicated by A3A2A1A0 (1-7, B-F) into the accumulator. If the auto increment of decrement mode is specified, the register pair (DE or HL) contents are automatically incremented by one after the memory contents are loaded into the accumultor.

If DE+byte of HL+byte is specified in rpa2, the memory location is addressed by the addition result of the DE or HL contents and the second byte of the instruction (Data). If HL+A, HL+B, or HL+EA is spedcified in rpa2, the memory location is addressed by the addition result of the HL contents and the contents of register A, B, or EA.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0

- 6) Example: LXI H,4000H;HL ← 4000H

MVI B, 20H ;B ← 20H
LDAX H+B ;A ← (4020H)

Load the contents of address 4020H into A.

- 7) Note: The number of bytes and the number of states very depending on the rpa² specification as listed below:

rpa2	B	D	H	D+	H+	D-	H-	D+byte	H+A	H+B	H+EA	H+byte
Number of bytes				1				2		1		2
Number of states				7(4)					13(7)			

EXX (Exchange Register Sets)

- 1) Operation code:

0	0	0	1	0	0	0	1
---	---	---	---	---	---	---	---
- 2) Number of bytes: 1
- 3) Number of states: 4 (4)
- 4) Function: $B \neq B'$, $C \neq C'$, $E \neq E'$, $H \neq H'$, $L \neq L'$,

Exchange the contents of registers B, C, D, E, H, and L and the contents of register B', C', D', E', H', and L'.

- 5) Affected flags: $SK \leftarrow 0$, $L1 \leftarrow 0$, $L0 \leftarrow 0$
- 6) Example:

EXA (Exchange V, A, EA and V', A', EA')

- 1) Operation code:

0	0	0	1	0	0	0	0
---	---	---	---	---	---	---	---
- 2) Number of bytes: 1
- 3) Number of states: 4 (4)
- 4) Function: $V \neq V'$, $A \neq A'$, $EA \neq EA'$

Exchange the contents of registers V, A, and EA and the contents of registers V', A', and EA'.

- 5) Affected flags: $SK \leftarrow 0$, $L1 \leftarrow 0$, $L0 \leftarrow 0$
- 6) Example:

EXH (Exchange HL and H' L')

- 1) Operation code:

0	1	0	1	0	0	0	0
---	---	---	---	---	---	---	---
- 2) Number of bytes: 1
- 3) Number of states: 4 (4)
- 4) Function: $H \rightleftarrows H'$, $L \rightleftarrows L'$

Exchange the contents of registers H and L and the contents of registers H' and L'

- 5) Affected flags: SK ← 0, L1 ← 0, LO ← 0
- 6) Example:

BLOCK (Block Data Transfer)

- 1) Operation code:

0	0	1	1	0	0	0	1
---	---	---	---	---	---	---	---
- 2) Number of bytes: 1
- 3) Number of states: $13 \times (C+1)$, (4)
- 4) Function: $(DE) \leftarrow (HL) +$, $C \leftarrow C - 1$, end if borrow

Transfer the contents of the memory location addressed by the HL register pair to the memory location addressed by the DE register pair as many bytes as determined by the contents of the C register used as a counter.

Whenever one byte is transferred, both HL and DE are automatically incremented and the C register is decremented. When the C register overflows, the instruction is terminated, and the next instruction is executed.

An interrupt can be acknowledged when data transfer is repeated by the BLOCK instruction. After a return is made from the interrupt service, the data transfer is continued.

- 5) Affected flags: SK 0, L1 0, LO 0
- 6) Example:

13.6.2 16-bit data transfer instructions

DMOV rp3, EA (Move EA to Register Pair)

- 1) Operation code:

1	0	1	1	0	1	P ₁	P ₀
---	---	---	---	---	---	----------------	----------------
- 2) Number of bytes: 1
- 3) Number of states: 4 (4)
- 4) Function: rp3L EAL, rp3H EAH

Transfer the contents of the low-order part of the expansion accumulator (EAL) to the low-order part C, E, or L of register pair rp3 (BC, DE, or HL) indicated by P1P0 (1-3). Transfer the contents of the high-order part (EAH) to the high-order part B, D, or H of the register pair.

- 5) Affected flags: SK 0, L1 0, LO 0
- 6) Example: DMOV B, EA;C EAL, B EAH

DMOV EA, rp3 (Move Register Pair to EA)

- 1) Operation code:

1	0	1	0	0	1	P ₁	P ₀
---	---	---	---	---	---	----------------	----------------
- 2) Number of bytes: 1
- 3) Number of states: 4 (4)
- 4) Function: EAL rp3L, EAH rp3H

Transfer the contents of the low-order part (C, E, or L) of register pair rp3 (BC, DE, or HL) indicated by P1P0 (!-3) to the low-order part the expansion accumulator (EAL). Transfer the contents of the high-order part of the register pair (B, D, or H) to the high-order part (EAH).

- 5) Affected flags: SK 0, L1 0, LO 0
- 6) Example: DMOV EA, B;EAL C, EAH B

DMOV sr3, EA (Move EA to Special Register)

- 1) Operation code:

0 1 0 0 1 0 0 0

1 1 0 1 0 0 1 U₀

- 2) Number of bytes: 2
- 3) Number of states: 14 (8)
- 4) Function: sr3 ← EA

Transfer the expansion accumulator contents to special register sr3 (ETM0 or ETM1) addressed by U₀ (0 or 1).

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0
- 6) Example: DMOV ETM0, EA; Transfer EA to ETM0.

DMOV EA, sr4 (Move Special Register to EA)

- 1) Operation code:

0 1 0 0 1 0 0 0

1 1 0 0 0 0 0 V₀

- 2) Number of bytes: 2
- 3) Number of states: 14 (8)
- 4) Function: EA ← sr4

Transfer the contents of special register (ECNT or ECPT) addressed by V₀ (0 or 1) to the expansion accumulator.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0
- 6) Example:

SBCD word (Store B&C Direct)

- 1) Operation code:

0	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

0	0	0	1	1	1	1	0
---	---	---	---	---	---	---	---

L.Adrs

H.Adrs

- 2) Number of bytes: 4

- 3) Number of states: 20 (14)

- 4) Function: (word) ← C, (word+1) ← B

Store the C register contents in the memory location addressed by the third byte (low address) and the fourth byte (high address) of the instruction. Store the B register contents in the next memory address.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0

- 6) Example: SBCD 4000H; Store the C register contents in address 4000H. Store the B register contents in address 4001H.

SDED word (Store D&E Direct)

- 1) Operation code:

0 1 1 1 0 0 0 0

- 2) Number of bytes: 4

- 3) Number of states: 20 (14)

- 4) Function: (word) ← E, (word+1) ← D

Store the E register contents in the memory location addressed by the third byte (low address) and the fourth byte (high address) of the instruction. Store the D register contents in the next memory address.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0

- 6) Example:

SHLD word (Store H&L Direct)

- 1) Operation code:

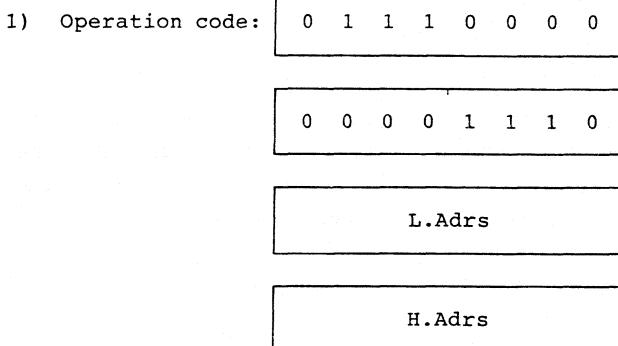
0 1 1 1 0 0 0 0

- 2) Number of bytes: 4
- 3) Number of states: 20 (14)
- 4) Function: $(\text{word}) \leftarrow L, (\text{word}+1) \leftarrow H$

Store the L register contents in the memory location addressed by the third byte (low address) and the fourth byte (high address) of the instruction. Store the H register contents in the next memory address.

- 5) Affected flags: SK $\leftarrow 0$, L1 $\leftarrow 0$, L0 $\leftarrow 0$
- 6) Example:

SSPD word (Store SP Direct)



- 2) Number of bytes: 4
- 3) Number of states: 20 (14)
- 4) Function: $(\text{word}) \leftarrow SP_L, (\text{word}+1) \leftarrow SP_H$

Store the low-order eight bits of the stack pointer (SP_L) in the memory location addressed by the third byte (low address) and the fourth byte (high address) of the instruction. Store the high-order eight bits (SP_H) in the next memory address.

- 5) Affected flags: SK $\leftarrow 0$, L1 $\leftarrow 0$, L0 $\leftarrow 0$
- 6) Example:

STEAX rpa3 (Store EA to Memory addressed by Register Pair)

- 1) Operation code

0	1	0	0	1	0	0	0
---	---	---	---	---	---	---	---

1	0	0	1	C ₃	C ₂	C ₁	C ₀
---	---	---	---	----------------	----------------	----------------	----------------

Data (Note)

- 2) Number of bytes: 2 or 3 (Note)

- 3) Number of states: 14 (8) or 20 (11) (Note)

- 4) Function: (rpa3) ← EL, (rpa3+1) ← EAH

Store the contents of the low-order eight bits of the expansion accumulator (EAL) in the memory location addressed by register pair rpa3 (DE, HL, DE++, HL++, DE+byte, HL+A, HL+B, HL+EA, or HL+byte) indicated by C3C2C1C0 (2-5, B-F). Store the contents of the high-order eight bits (EAH) in the memory location addressed by rpa3+1. If DE+byte of HL+byte is specified in rpa3, the memory location is addressed by the addition result of the DE or HL contents and the third byte of the instruction (Data). If HL+A, HL+B, or HL+EA is specified in rpa3, the memory location is addressed by the addition result of the HL contents and the contents of register A, AB, or EA.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0

- 6) Example: LXI D,4000H;DE ← 4000H

STEAX D++ ;(4000H) ← EAL, (4001H) ← EAH
;DE ← 4002H

STEAX D+10H ;(4012H) ← EAL, (4013H) ← EAH
;DE=4002H

Store the contents of the low-order eight bits of the expansion accumulator (EAL) in addresses 4000H and 4012H. Store the contents of the high-order eight bits (EAH) in addresses 4001 and 4013.

- 7) Note: The number of bytes and the number of states vary depending on the rpa3 specification as listed below:

rpa2	D	H	D++	H++	D+byte	H+A	H+B	H+EA	H+byte
Number of bytes		2			3		2		3
Number of states		14(8)				20(11)			

LBCD word (Load B&C Direct)

- 1) Operation code: 0 1 1 1 0 0 0 0 0

0 0 0 1 1 1 1 1

L.Adrs

H.Adrs

- 2) Number of bytes: 4

- 3) Number of states: 20 (14)

- 4) Function: C ← (word), B ← (word+1)

Load the contents of the memory location addressed by the third byte (low address) and the fourth byte (high address) of the instruction into the C register and the contents of the next memory address into the B register.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0

- 6) Example:

LDED word (Load D&E Direct)

- 1) Operation code:

0 1 1 1 0 0 0 0

0 0 1 0 1 1 1 1

L.Adrs

H.Adrs

- 2) Number of bytes: 4

- 3) Number of states: 20 (14)

- 4) Function: E ← (word), D ← (word+1)

Load the contents of the memory location addressed by the third byte (low address) and the fourth byte (high address) of the instruction into the E register and the contents of the next memory address into the D register.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0

- 6) Example:

LHLD word (Load H&E Direct)

- 1) Operation code:

0 1 1 1 0 0 0 0

0 0 1 1 1 1 1 1

L.Adrs

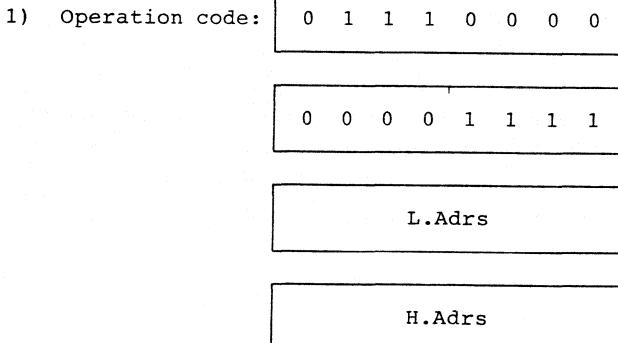
H.Adrs

- 2) Number of bytes: 4
- 3) Number of states: 20 (14)
- 4) Function: $L \leftarrow (\text{word}), H \leftarrow (\text{word}+1)$

Load the contents of the memory location addressed by the third byte (low address) and the fourth byte (high address) of the instruction into the L register contents in the next memory address into the H register.

- 5) Affected flags: $SK \leftarrow 0, L1 \leftarrow 0, L0 \leftarrow 0$
- 6) Example:

LSPD word (Load SP Direct)



- 2) Number of bytes: 4
- 3) Number of states: 20 (14)
- 4) Function: $SPL \leftarrow (\text{word}), SPH \leftarrow (\text{word}+1)$

Load the contents of the memory location addressed by the third byte (low address) and the fourth byte (high address) of the instruction into the low-order eight bits of the stack pointer (SPL) and the contents of the next memory address into the high-order eight bits (SPH).

- 5) Affected flags: $SK \leftarrow 0, L1 \leftarrow 0, L0 \leftarrow 0$
- 6) Example:

LDEAX rpa3 (Load EA with Memory addressed by Register Pair)

- 1) Operation code

0	1	0	0	1	0	0	0
---	---	---	---	---	---	---	---

1	0	0	1	C ₃	C ₂	C ₁	C ₀
---	---	---	---	----------------	----------------	----------------	----------------

Data (Note)

- 2) Number of bytes: 2 or 3 (Note)

- 3) Number of states: 14 (8) or 20 (11) (Note)

- 4) Function: EAL ← (rpa3), EAH ← (rpa3+1)

Load the contents of the memory location addressed by register pair rpa3 (DE, HL, DE++, HL++, DE+byte, HL+A, HL+B, HL+EA, or HL+byte) indicated by C3C2C1C0 (2-5, B-F) into the low-order bits of the expansion accumulator (EAL) and the memory location addressed by rpa3+1 into the high-order eight bits (EAH). If DE+byte of HL+byte is specified in rpa3, the memory location is addressed by the addition result of the DE or HL contents and the third byte of the instruction (Data).

If HL+A, HL+B, or HL+EA is specified in rpa3, the memory location is addressed by the addition result of the HL contents and the contents of register A, B, or EA.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0

- 6) Example:

- 7) Note: The number of bytes and the number of states vary depending on the rpa3 specification as listed below:

rpa2	D	H	D++	H++	D+byte	H+A	H+B	H+EA	H+byte
Number of bytes		2		3		2		3	
Number of states	14(8)				20(11)				

PUSH rpl (Push Register Pair on Stack)

- 1) Operation code

1	0	1	1	0	Q ₂	Q ₁	Q ₀
---	---	---	---	---	----------------	----------------	----------------

2) Number of bytes: 1

3) Number of states: 13 (4)

4) Function: (SP-1) + rplH, (SP-2) + rplL, SP + SP-2

Save the high-order part (V, B, D, H, or EAH) of the register pair VA, BC, DE, or HL or the expansion accumulator addressed by Q2Q1Q0 (0-4) in the stack memory addressed by (SP-1) and the low-order part (A, C, E, L, or EAL) in the stack memory addressed by (SP-2).

5) Affected flags: SK + 0, L1 + 0, L0 + 0

6) Example: ;PROGRAM START

RXI SP, 0E000H

;INTERRUPT ROUTINE

PUSH V

PUSH B

PUSH D

PUSH H

PUSH EA

POP EA

POP H

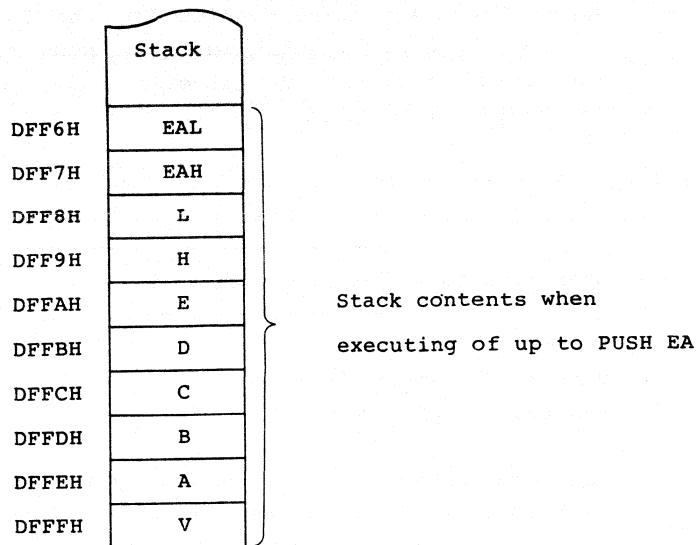
POP D

POP B

POP V

EI

RETI



POP rp1 (Pop Register Pair off stack)

1) Operation code

1	0	1	0	0	Q_2	Q_1	Q_0
---	---	---	---	---	-------	-------	-------

- 2) Number of bytes: 1
- 3) Number of states: 10 (4)
- 4) Function: $rp1_L \leftarrow (SP)$, $rp1_H \leftarrow (SP+1)$, $SP \leftarrow SP+2$

Restore the contents of the stack memory addressed by (SP) in the low-order part (A, C, E, L, or EAL) of the register pair VA, BC, DE, or HL or the expansion accumulator addressed by $Q_2Q_1Q_0$ (0-4) and the contents of the memory stack addressed by (SP+1) in the high-order part (V, B, D, H, or EAH).

- 5) Affected flags: SK $\leftarrow 0$, L1 $\leftarrow 0$, L0 $\leftarrow 0$

- 6) Example: PUSH B
PUSH D

```
POP  D
POP  B
```

Since the stack pointer points to the last saved stack address, restore the stack memory contents by executing the POP instruction in the reverse order when the PUSH instruction is executed.

LXI rp2, word (Load Register Pair with Immediate)

1) Operation code 0 P₂ P₁ P₀ 0 1 0 0

2) Number of bytes: 3

3) Number of states: 10 (10)

4) Function: rp2 ← word

Load the contents of the second byte into the low-order part (SPL, C, E, L, or EAL) of the SP, register pair BC, DE, or HL, or expansion accumulator addressed by P2P1P0 (0-4) and the contents of the third byte into the high-order part (SPH, B, D, H, or EAH). If the HL register pair is specified, the string effect is made.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 1 (when rp2=HL)
SK ← 0, 11 ← 0, 10 ← 0 (other than the
above)
- 6) Example: LXI B, 4000H; Load 40H into B register and 00H
into C register.

TABLE (Table pick up)

1) Operation code 0 1 0 0 1 0 0 0

1 0 1 0 1 0 0 0

2) Number of bytes: 2

3) Number of states: 17 (18)

- 4) Function: $C \leftarrow (PC+3+A)$, $B \leftarrow (PC+3+A+1)$

Load the contents of the table addressed by $PC+3+A$ into the C register and the contents of the table addressed by $PC+3+A+1$ into the B register

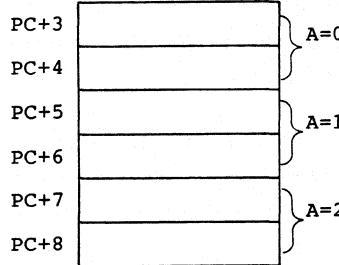
- 5) Affected flags: SK $\leftarrow 0$, L1 $\leftarrow 0$, L0 $\leftarrow 0$

- 6) Example: TB0:MVI A,0:A=0

TB1:MVI A,1;A=1

TB2:MVI A,2;A=2

SLL A ;Shift Logical Left Accumulator
PC TAB ;BC \leftarrow (Table)
PC+2 JB ;PC \leftarrow BC



13.6.3 8-bit operation instructions (registers)

ADD A, r (Add Register to A)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function: $A \leftarrow A + r$

Add the contents of register r (V, A, B, C, D, E, H, or L) addressed by $R_2 R_1 R_0$ (0-7) to the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK \leftarrow 0, HC, L1 \leftarrow 0, L0 \leftarrow 0, CY

- 6) Example: Add A, C; Add the C register contents to A and save the result in A.

ADD r, A (Add A to Register)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

0	1	0	0	0	R_2	R_1	R_0
---	---	---	---	---	-------	-------	-------

- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function: $r \leftarrow r + A$

Add the accumulator contents to register r (V, A, B, C, D, E, H, or L) addressed by $R_2 R_1 R_0$ (0-7). Save the result in the specified register.

- 5) Affected flags: Z, SK \leftarrow 0, HC, L1 \leftarrow 0, L0 \leftarrow 0, CY
- 6) Example: ADD B, A; Add the A register contents to B and save the result in B register.

ADC A, r (Add Register to A with Carry)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

1	1	0	1	0	R ₂	R ₁	R ₀
---	---	---	---	---	----------------	----------------	----------------

- 2) Number of bytes: 2
- 3) Number of states: 8 (8)
- 4) Function: A \leftarrow A+r+CY

Add the contents of register r (V, A, B, C, D, E, H, or L) addressed by R₂R₁R₀ (0-7) to the accumulator with carry flag. Save the result in the accumulator.

- 5) Affected flags: Z, SK \leftarrow 0, HC, L1 \leftarrow 0, L0 \leftarrow 0, CY
- 6) Example: ADC A, E; A \leftarrow A+E+CY

ADC r, A (Add A to Register with Carry)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

0	1	0	1	0	R ₂	R ₁	R ₀
---	---	---	---	---	----------------	----------------	----------------

- 2) Number of bytes: 2
- 3) Number of states: 8 (8)
- 4) Function: r \leftarrow r+A+CY

Add the accumulator contents to register r (V, A, B, C, D, E, H, or L) addressed by R₂R₁R₀ (0-7) with CY flag. Save the result in the specified register.

- 5) Affected flags: Z, SK \leftarrow 0, HC, L1 \leftarrow 0, L0 \leftarrow 0, CY
- 6) Example: Add register pairs HL and DE and save the result in HL.

```
MOV A,E;A  $\leftarrow$  E  
ADD L,A;L  $\leftarrow$  L+A  
MOV A,D;A  $\leftarrow$  D  
ADC H,A;H  $\leftarrow$  H+A+CY
```

ADDNC A, r (Add Register to A. Skip if No Carry)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

1	0	1	0	0	R ₂	R ₁	R ₀
---	---	---	---	---	----------------	----------------	----------------

- 2) Number of bytes: 2
- 3) Number of states: 8 (8)
- 4) Function: A + A+r; Skip if no carry

Add the contents of register r (V, A, B, C, D, E, H, or L) addressed by R₂R₁R₀ (0-7) to the accumulator. Save the result in the accumulator. Skip if there is no carry as the result of addition.

- 5) Affected flags: Z, SK, HC, L1 \leftarrow 0, L0 \leftarrow 0, CY
- 6) Example: ADDNC A, V;A \leftarrow A+V
Skip if there is no carry as the result of addition.

ADDNC r, A (Add A to Register. Skip if No Carry)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

0	0	1	0	0	R ₂	R ₁	R ₀
---	---	---	---	---	----------------	----------------	----------------

- 2) Number of bytes: 2
- 3) Number of states: 8 (8)
- 4) Function: $r \leftarrow r+A$; Skip if no carry

Add the accumulator contents of register r (V, A, B, C, D, E, H, or L) addressed by $R_2R_1R_0$ (0-7). Save the result in the specified register. Skip if there is no carry as the result of addition.

- 5) Affected flags: Z, SK, HC, L1 \leftarrow 0, L0 \leftarrow 0, CY

- 6) Example: Add A to HL register pair.

ADDNC L, A; L \leftarrow L+A, SKIP IF NO CARRY.

ADI H, 1; H \leftarrow H+1

Skip if there is no carry and terminate the addition.
If there is a carry, make carry addition to the high=order byte and terminate the addition.

SUB A, r (Subtract Register from A)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

1	1	1	0	0	R_2	R_1	R_0
---	---	---	---	---	-------	-------	-------

- 2) Number of bytes: 2
- 3) Number of states: 8 (8)
- 4) Function: $A \leftarrow A-r$

Subtract the contents of register r (V, A, B, C, D, E, H, or L) addressed by $R_2R_1R_0$ (0-7) from the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK \leftarrow 0, HC, L1 \leftarrow 0, L0 \leftarrow 0, CY
- 6) Example: SUB A, B; A \leftarrow A-B

SUB r, A (Subtract A from Register)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

0	1	1	0	0	R ₂	R ₁	R ₀
---	---	---	---	---	----------------	----------------	----------------

- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function: r ← r-A

Subtract the accumulator contents from register r (V, A, B, C, D, E, H, or L) addressed by R₂R₁R₀ (0-7). Save the result in the specified register.

- 5) Affected flags: Z, SK ← 0, HC, L1 ← 0, L0 ← 0, CY

- 6) Example: SUB A, A;A A-A=0

SBB A, r (Subtract Register from A with Borrow)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

1	1	1	1	0	R ₂	R ₁	R ₀
---	---	---	---	---	----------------	----------------	----------------

- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function: A ← A-r-CY

Subtract the contents of register r (V, A, B, C, D, E, H, or L) addressed by R₂R₁R₀ (0-7) and CY flag from the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK ← 0, HC, L1 ← 0, L0 ← 0, CY

- 6) Example: SBB A, L;A ← A-L-CY

SBB r, A (Subtract A from Register with Borrow)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function:
- $r \leftarrow r - A - CY$

Subtract the accumulator contents and CY flag from register r (V, A, B, D, E, H, or L) Addressed by $R_2 R_1 R_0$ (0-7).

Save the result in the specified register.

- 5) Affected flags: Z, SK
- $\leftarrow 0$
- , HC, L1
- $\leftarrow 0$
- , L0
- $\leftarrow 0$
- , CY

- 6) Example: SBB B, A; B
- $\leftarrow B - A - CY$

SUBNB A, r (Subtract Register from A. Skip if No Borrow)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

1	0	1	1	0	R_2	R_1	R_0
---	---	---	---	---	-------	-------	-------

- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function:
- $A \leftarrow A - r$
- ; Skip if no borrow.

Subtract the accumulator contents and CY flag from register r (V, A, B, D, E, H, or L) addressed by $R_2 R_1 R_0$ (0-7).

Save the result in the accumulator. Skip if there is no borrow as the result of subtraction.

- 5) Affected flags: Z, SK, HC, L1
- $\leftarrow 0$
- , L0
- $\leftarrow 0$
- , CY

- 6) Example: SUBNB A, D; A
- $\leftarrow A - D$

Skip if there is no borrow as the result of subtraction.

SUBNB r, A (Subtract A from Register. Skip if No Borrow)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function: $r \leftarrow r - A$; Skip if no borrow

Subtract the accumulator contents from register r (V, A, B, C, D, E, H, or L) addressed by $R_2 R_1 R_0$ (0-7). Save the result in the specified register. Skip if there is no borrow as the result of subtraction.

- 5) Affected flags: Z, SK, HC, L1 $\leftarrow 0$, L0 $\leftarrow 0$, CY

- 6) Example: Subtract A from HL register pair

SUBNB L, A; L $\leftarrow L - A$, SKIP IF NO BORROW
SUI H, 1; H $\leftarrow H - 1$

ANA A, r (And Register with A)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

1	0	0	0	1	R_2	R_1	R_0
---	---	---	---	---	-------	-------	-------

- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function: $A \leftarrow A \wedge r$

AND the contents of register r (V, A, B, C, D, E, H, or L) addressed by $R_2 R_1 R_0$ (0-7) with the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK $\leftarrow 0$, L1 $\leftarrow 0$, L0 $\leftarrow 0$, CY

- 6) Example: ANA A, L; A $\leftarrow A \wedge L$

ANA r, A (And Register with A)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

0	0	0	0	1	R ₂	R ₁	R ₀
---	---	---	---	---	----------------	----------------	----------------

- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function: r ← r ∧ A

And the accumulator with the contents from register r (V, A, B, C, D, E, H, or L) addressed by R₂R₁R₀ (0-7). Save the result in the specified register.

- 5) Affected flags: Z, SK ← 0, L1 ← 0, L0 ← 0,

- 6) Example: ANA H, A;H ← H ∧ A

ORA A, r (Or Register with A)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

1	0	0	1	1	R ₂	R ₁	R ₀
---	---	---	---	---	----------------	----------------	----------------

- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function: A ← A ∨ r

Or the accumulator with the contents of register r (V, A, B, C, D, E, H, or L) addressed by R₂R₁R₀ (0-7). Save the result in the specified register.

- 5) Affected flags: Z, SK ← 0, L1 ← 0, L0 ← 0

- 6) Example: ORA A,H ;A ← A ∨ H

ORA r, A (Or A with Register)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

0	0	0	1	1	R_2	R_1	R_0
---	---	---	---	---	-------	-------	-------

- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function: $r \leftarrow r \vee A$

OR the accumulator with the contents of register r (V, A, B, C, D, E, H, or L) addressed by $R_2R_1R_0$ (0-7). Save the result in the specified register.

- 5) Affected flags: Z, SK \leftarrow 0, L1 \leftarrow 0, L0 \leftarrow 0,

- 6) Example: ORA L, A; L \leftarrow LV A

XRA A, r (Exclusive-Or Register with A)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

1	0	0	1	0	R_2	R_1	R_0
---	---	---	---	---	-------	-------	-------

- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function: $A \leftarrow A \vee r$

Exclusive-OR the contents of register r (V, A, B, C, D, E, H, or L) addressed by $R_2R_1R_0$ (0-7) with the accumulator. Save the result in the specified register.

- 5) Affected flags: Z, SK \leftarrow 0, L1 \leftarrow 0, L0 \leftarrow 0

- 6) Example: XRA A, B; A \leftarrow A V B

XRA r, A (Exclusive-Or A with Register)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

0	0	0	1	0	R ₂	R ₁	R ₀
---	---	---	---	---	----------------	----------------	----------------

- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function:
- $r \leftarrow r \text{V} A$

Exclusive-OR the accumulator with the contents from register r (V, A, B, C, D, E, H, or L) addressed by R₂R₁R₀ (0-7). Save the result in the specified register.

- 5) Affected flags: Z, SK
- \leftarrow
- 0, L1
- \leftarrow
- 0, L0
- \leftarrow
- 0,

- 6) Example: XRA C, A; C
- \leftarrow
- CY A

GTA A, r (Greater Than Register)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

1	0	1	0	1	R ₂	R ₁	R ₀
---	---	---	---	---	----------------	----------------	----------------

- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function: A - r - 11 Skip if no borrow

Subtract the contents of register r (V, A, B, C, D, E, H, or L) addressed by R₂R₁R₀ (0-7) and 1 from the accumulator.

Skip if there is no borrow as the result of subtraction (A > r).

- 5) Affected flags: Z, SK, HC, L1
- \leftarrow
- 0, L0
- \leftarrow
- 0, CY

- 6) Example: GTA A, B; A-B-1

Skip if A is greater than B.

GTA r, A (Greater Than A)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

0	0	1	0	1	R_2	R_1	R_0
---	---	---	---	---	-------	-------	-------

- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function: $r-A-1$; Skip if no borrow

Subtract the accumulator with the contents from register r (V , A , B , C , D , E , H , or L) addressed by $R_2R_1R_0$ (0-7).

Skip if there is no borrow as the result of subtraction ($r > A$).

- 5) Affected flags: Z , SK , HC , $L1 \leftarrow 0$, $L0 \leftarrow 0$, CY

- 6) Example: GTA B, A;B-A-1

Skip if B is greater than A

LTA A, r (Less Than Register)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

1	0	1	1	1	R_2	R_1	R_0
---	---	---	---	---	-------	-------	-------

- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function: $A-r$; Skip if borrow

Subtract the contents of register r (V , A , B , C , D , E , H , or L) addressed by $R_2R_1R_0$ (0-7) from the accumulator. Skip if there is a borrow as the result of subtraction ($A < r$).

- 5) Affected flags: Z , SK , HC , $L1 \leftarrow 0$, $L0 \leftarrow 0$, CY

- 6) Example: LTA A, L;A-L
Skip if A is less than the L register.

LTA r, A (Less Than A)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

0	0	1	1	1	R ₂	R ₁	R ₀
---	---	---	---	---	----------------	----------------	----------------

- 2) Number of bytes: 2
3) Number of states: 8 (8)
4) Function: r-A; Skip if borrow

Subtract the accumulator contents from the contents of register r (V, A, B, C, D, E, H, or L) addressed by R₂R₁R₀ (0-7). Skip if there is a borrow as the result of subtraction (r < A).

- 5) Affected flags: Z, SK, HC, L1 + 0, LO + 0, CY
6) Example: LTA H, A;H-A
Skip if H register is less than A

NEA A, r (Not Equal Register with A)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

1	1	1	0	1	R ₂	R ₁	R ₀
---	---	---	---	---	----------------	----------------	----------------

- 2) Number of bytes: 2
3) Number of states: 8 (8)
4) Function: A-r; Skip if no zero

Subtract the contents of register r (V, A, B, C, D, E, H, or L) addressed by R₂R₁R₀ (0-7) from the accumulator. Skip if zero is not returned as the result of subtraction (A=r).

5) Affected flags: Z, SK, HC, L1 ← 0, L0 ← 0, CY

6) Example: NEA A, B; SKIP IF A≠B

When A < B, the CY flag is set; when
A=B, the Z flag is set.

Skip if A is less than the L register.

NEA r, A (Not Equal A with Register)

1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

0	1	1	0	1	R ₂	R ₁	R ₀
---	---	---	---	---	----------------	----------------	----------------

2) Number of bytes: 2

3) Number of states: 8 (8)

4) Function: r-A; Skip if no zero

Subtract the accumulator contents of register r (V, A, B, C, D, E, H, or L) addressed by R₂R₁R₀ (0-7). Skip if zero is not returned as the result of subtraction (A=r).

5) Affected flags: Z, SK, HC, L1 ← 0, L0 ← 0, CY

6) Example: NEA C, A; SKIP IF C≠A

When C < A, the CY flag is set; when
C=A, the Z flag is set.

EQA A,r (Equal Register with A)

1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

1	1	1	1	1	R ₂	R ₁	R ₀
---	---	---	---	---	----------------	----------------	----------------

2) Number of bytes: 2

3) Number of states: 8 (8)

4) Function: A-r; Skip if zero

Subtract the contents of register r (V, A, B, C, D, E, H, or L) addressed by $R_2 R_1 R_0$ (0-7) from the accumulator. Skip if zero is returned as the result of subtraction ($A=r$).

- 5) Affected flags: Z, SK, HC, L1 + 0, L0 + 0, CY
- 6) Example: EQA A, D;SKIP IF A=D

EQA r, A (Equal A with Register)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

0	1	1	1	1	R_2	R_1	R_0
---	---	---	---	---	-------	-------	-------

- 2) Number of bytes: 2
- 3) Number of states: 8 (8)
- 4) Function: $r-A$; Skip if zero

Subtract the accumulator contents from the contents of register r (V, A, B, C, D, E, H, or L) addressed by $R_2 R_1 R_0$ (0-7). Skip if zero is returned as the result of subtraction ($r=A$).

- 5) Affected flags: Z, SK, HC, L1 + 0, L0 + 0, CY
- 6) Example: EQA E, A;SKIP IF E=A

ONA A,r (On-Test Register with A)

- 1) Operation code

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

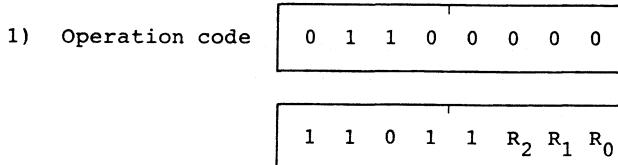
1	1	0	0	1	R_2	R_1	R_0
---	---	---	---	---	-------	-------	-------

- 2) Number of bytes: 2
- 3) Number of states: 8 (8)
- 4) Function: $A \wedge r$; Skip if no zero

And the contents of register r (V, A, B, C, D, E, H, or L) addressed by $R_2 R_1 R_0$ (0-7) with the accumulator. Skip if zero is not returned as the result of ANDing.

- 5) Affected flags: Z, SK, L1 \leftarrow 0, L0 \leftarrow 0,
- 6) Example:

OFFA A, r (Off-Test Register with A)



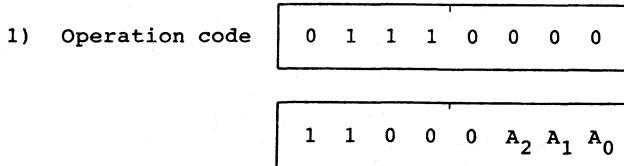
- 2) Number of bytes: 2
- 3) Number of states: 8 (8)
- 4) Function: $A \wedge r$; Skip if zero

AND the contents of register r (V, A, B, C, D, E, H, or L) addressed by $R_2 R_1 R_0$ (0-7) with the accumulator. Skip if zero is returned as the result of ANDing.

- 5) Affected flags: Z, SK, L1 \leftarrow 0, L0 \leftarrow 0,
- 6) Example:

13.6.4 8-bit operation instructions (memory)

ADDX rpa (Add Memory addressed by Register Pair to A)



- 2) Number of bytes: 2
- 3) Number of states: 11 (8)

- 4) Function: $A \leftarrow A + (rpa)$

Add the contents of the memory location addressed by register pair rpa (BC, DE, HL, DE+, HL+, DE-, or HL-) indicated by $A_2A_1A_0$ (1-7) to the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK $\leftarrow 0$, HC, L1 $\leftarrow 0$, LO $\leftarrow 0$, CY

- 6) Example: MOV A,4000H;A $\leftarrow (4000H)$
LXI H,4200H;HL $\leftarrow 4200H$
ADDX H ;A $\leftarrow A + (HL)$

Add the contents of address 4000H and the contents of address 4200H.

ADCX rpa (Add Memory addressed by Register Pair to A with Carry)

1) Operation code	0 1 1 1 0 0 0 0
	1 1 0 1 0 $A_2 A_1 A_0$

- 2) Number of bytes: 2
3) Number of states: 11 (8)
4) Function: $A \leftarrow A + (rpa) + CY$

Add the contents of the memory location addressed by register pair rpa (BC, DE, HL, DE+, HL+, DE-, or HL-) indicated by $A_2A_1A_0$ (1-7) with the CY flag to the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK $\leftarrow 0$, HC, L1 $\leftarrow 0$, LO $\leftarrow 0$, CY
6) Example: ADCX D+;A $\leftarrow A + (DE) + CY$, DE $\leftarrow DE + 1$

Add the contents of the memory location addressed by the DE register pair to A and save the result in A, then increment DE.

ADDNCX rpa (Add Memory addressd by Register Pair to A. Skip if No Carry)

- 1) Operation code

0	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

1	0	1	0	0	A_2	A_1	A_0
---	---	---	---	---	-------	-------	-------

- 2) Number of bytes: 2
3) Number of states: 11 (8)
4) Function: $A \leftarrow A + (rpa)$; Skip if no carry

Add the contents of the memory location addressed by register pair rpa (BC, DE, HL, DE+, HL+, DE-, or HL-) indicated by $A_2A_1A_0$ (1-7) to the accumulator. Save the result in the accumulator. Skip if there is no carry as the result of addition.

- 5) Affected flags: Z, SK \leftarrow HC, L1 \leftarrow 0, L0 \leftarrow 0, CY
6) Example:
- | | |
|--------|--|
| LXI | H,4200H ;HL \leftarrow 4200H |
| LXI | D,4000H ;DE \leftarrow 4000H |
| MOV | A,4100H ;A \leftarrow (4100H) |
| ADDNCX | D+ ;A $\leftarrow A + (DE)$, DE $\leftarrow DE + 1$ |
| STAX | H ;(HL) $\leftarrow A$ |
| JMP | MOTOE |

Add the contents of address 4100H and the contents of address 4000H. Save the result in address 4200H if there is no carry. If there is a carry, skip the STAX instruction, execute the JMP instruction, and jump to MOTOE.

SUBX rpa (Subtract Memory addressed by Register Pair from A)

- 1) Operation code

0	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

- 2) Number of bytes: 2
3) Number of states: 11 (8)
4) Function: $A \leftarrow A - (rpa)$

Subtract the contents of the memory location addressed by register pair rpa (BC, DE, HL, DE+, HL+, DE-, or HL-) indicated by $A_2A_1A_0$ (1-7) from the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK $\leftarrow 0$, HC, L1 $\leftarrow 0$, L0 $\leftarrow 0$, CY
6) Example: SUBX D;A $\leftarrow A - (\text{DE})$

SBBX rpa (Subtract Memory addressed by Register Pair from A)

- 1) Operation code

0	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

- 2) Number of bytes: 2
3) Number of states: 11 (8)
4) Function: $A \leftarrow A - (rpa) - CY$

Subtract the contents of the memory location addressed by register pair rpa (BC, DE, HL, DE+, HL+, DE-, or HL-) indicated by $A_2A_1A_0$ (1-7) and the CY flag from the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK $\leftarrow 0$, HC, L1 $\leftarrow 0$, L0 $\leftarrow 0$, CY

- 6) Example: SBBX D-;A \leftarrow A-(DE)-CY, DE \leftarrow DE-1

Subtract the contents of the memory location addressed by the DE register pair and the CY flag from A and save the result in A, then decrement DE.

SUBNBX rpa (Subtract Memory addressed by Register Pair from A.

Skip if No Borrow)

- 1) Operation code

0	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

1	0	1	1	0	A ₂	A ₁	A ₀
---	---	---	---	---	----------------	----------------	----------------

- 2) Number of bytes: 2

- 3) Number of states: 11 (8)

- 4) Function: A \leftarrow A-(rpa); Skip if no borrow

Subtract the contents of the memory location addressed by register pair rpa (BC, DE, HL, DE+, HL+, DE-, or HL-) indicated by A₂A₁A₀ (1-7) from the accumulator. Save the result in the accumulator. Skip if there is no borrow as the result of subtraction.

- 5) Affected flags: Z, SK \leftarrow HC, L1 \leftarrow 0, L0 \leftarrow 0, CY

- 6) Example: SUBNBX B;A \leftarrow A-(BC)

Skip if there is no borrow as the result of subtraction.

ANAX rpa (And Memory addressed by Register Pair with A)

- 1) Operation code

0	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

1	0	0	0	1	A_2	A_1	A_0
---	---	---	---	---	-------	-------	-------

- 2) Number of bytes: 2
- 3) Number of states: 11 (8)
- 4) Function: $A \leftarrow A \wedge (\text{rpa})$

AND the contents of the memory location addressed by register pair rpa (BC, DE, HL, DE+, HL+, DE-, or HL-) indicated by $A_2A_1A_0$ (1-7) with the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK + 0, L1 \leftarrow 0, L0 \leftarrow 0
- 6) Example: ANA H-;A $\leftarrow A \wedge (\text{HL})$, HL $\leftarrow \text{HL}-1$

AND the contents of the memory location addressed by the HL register pair with A and save the result in A, then decrement HL.

ORAX rpa (Or Memory addressed by Register Pair with A)

- 1) Operation code

0	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

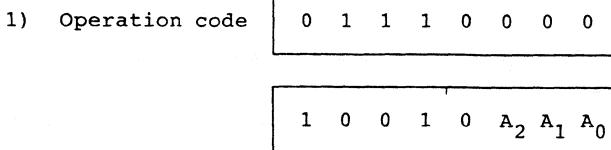
1	0	0	1	1	A_2	A_1	A_0
---	---	---	---	---	-------	-------	-------

- 2) Number of bytes: 2
- 3) Number of states: 11 (8)
- 4) Function: $A \leftarrow AV(\text{rpa})$

OR the contents of the memory location addressed by register pair rpa (BC, DE, HL, DE+, HL+, DE-, or HL-) indicated by $A_2A_1A_0$ (1-7) with the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK + 0, L1 + 0, LO + 0
- 6) Example: PRAX D;A + AV(DE)

XRAX rpa (Exclusive-Or Memory addressed by Register Pair with A)



- 2) Number of bytes: 2
- 3) Number of states: 11 (8)
- 4) Function: A + AV(rpa)

Exclusive-OR the contents of the memory location addressed by register pair rpa (BC, DE, HL, DE+, HL+, DE-, or HL-) indicated by $A_2A_1A_0$ (1-7) with the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK + 0, L1 + 0, LO + 0
- 6) Example:

```
LXI H,4000H ;HL + 4000H
MVI A,0A8H ;A + A8H
XRAX H ;A + AV(HL)
SK Z ;SKIP IF ZERO
STAX D ;(DE) + A
JMP KORED
```

10101000	A8H	Memory
▼		
10101000	A8H	4000H
+		A8H
00000000	00H	

In this example, since the contents of A match the contents of address 4000H, the result of exclusive-OR is 0 and Z flag is set. Thus, the STAX instruction is skipped as the result of execution of the SK Z instruction following the XRAX instruction, and the JMP instruction is executed.

GTAX rpa(Greater Than Memory addressed by Register Pair)

- 1) Operation code

0	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

- 2) Number of bytes: 2

- 3) Number of states: 11 (8)

- 4) Function: $A - (rpa) - 1$; Skip if no borrow

1	0	1	0	1	A_2	A_1	A_0
---	---	---	---	---	-------	-------	-------

Subtract the contents of the memory location addressed by register pair rpa (BC, DE, HL, DE+, HL+, DE-, or HL-) indicated by $A_2A_1A_0$ (1-7) and 1 from the accumulator. Skip if there is no borrow as the result of subtraction ($A > (rpa)$).

- 5) Affected flags: Z, SK, HL, L1 + 0, L1 - 0, CY

- 6) Example: GTAX D;A-(DE)-1

Skip if A is greater than the contents of the memory location addressed by the DE register pair.

LTXA rpa (Less Than Memory addressed by Register Pair)

- 1) Operation code

0	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

- 2) Number of bytes: 2

- 3) Number of states: 11 (8)

- 4) Function: A-(rpa); Skip if borrow

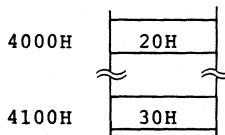
1	0	1	1	1	A ₂	A ₁	A ₀
---	---	---	---	---	----------------	----------------	----------------

Subtract the contents of the memory location addressed by register pair rpa (BC, DE, HL, DE+, HL+, DE-, or HL-) indicated by A₂A₁A₀ (1-7) from the accumulator. Skip if there is a borrow as the result of subtraction (A <(rpa)).

- 5) Affected flags: Z, SK, HC, L1 ← 0, L0 ← 0, CY

- 6) Example:
- ```
LXI D,4000H ;DE ← 4000H
LXI H,1100H ;HL ← 4100H
LDAX D ;A ← (4000H)
LTAX H ;A-(HL)
LDAX H ;A ← (HL)
JMP KORED
```

Memory



In this example, since the contents of A (contents of address 4000H=20H) are less than the contents of the memory location addressed by the HL register pair (contents of address 4100H=30H),

the LDAX instruction is skipped and the JMP instruction is executed. As the result of subtraction, the CY, SK, and HC flags are set.

NEAX rpa (Not Equal Memory addressed by Register Pair with A)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |       |       |       |
|---|---|---|---|---|-------|-------|-------|
| 1 | 1 | 1 | 0 | 1 | $A_2$ | $A_1$ | $A_0$ |
|---|---|---|---|---|-------|-------|-------|

- 2) Number of bytes: 2  
3) Number of states: 11 (8)  
4) Function:  $A - (rpa)$ ; Skip if no zero

Subtract the contents of the memory location addressed by register pair rpa (BC, DE, HL, DE+, HL+, DE-, or HL-) indicated by  $A_2A_1A_0$  (1-7) from the accumulator. Skip if zero is returned as the result of subtraction ( $A = (rpa)$ ).

- 5) Affected flags: Z, SK, HL, L1 + 0, LO + 0, CY  
6) Example: NEAX B;SKIP IF A=(BC)

EQAX rpa (Equal Memory addressed by Register Pair with A)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |       |       |       |
|---|---|---|---|---|-------|-------|-------|
| 1 | 1 | 1 | 1 | 1 | $A_2$ | $A_1$ | $A_0$ |
|---|---|---|---|---|-------|-------|-------|

- 2) Number of bytes: 2  
3) Number of states: 11 (8)  
4) Function:  $A - (rpa)$ ; Skip if no zero

Subtract the contents of the memory location addressed by register pair rpa (BC, DE, HL, DE+, HL+, DE-, or HL-) indicated by  $A_2 A_1 A_0$  (1-7) from the accumulator. Skip if zero is no returned as the result of subtraction ( $A = (rpa)$ ).

- 5) Affected flags: Z, SK, HL, L1 + 0, L1 - 0, CY
- 6) Example: EQAX D;SKIP IF A=(DE)

Skip if A equals to the contents of the memory location addressed by the DE register pair.

ONAX rpa (On-Test Memory addressed by Register Pair with A)

|                   |                                                                                                                                                                  |   |   |   |       |       |       |       |       |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|---|---|-------|-------|-------|-------|-------|
| 1) Operation code | <table border="1"><tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>                                              | 0 | 1 | 1 | 1     | 0     | 0     | 0     | 0     |
| 0                 | 1                                                                                                                                                                | 1 | 1 | 0 | 0     | 0     | 0     |       |       |
|                   | <table border="1"><tr><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td><math>A_2</math></td><td><math>A_1</math></td><td><math>A_0</math></td></tr></table> | 1 | 1 | 0 | 0     | 1     | $A_2$ | $A_1$ | $A_0$ |
| 1                 | 1                                                                                                                                                                | 0 | 0 | 1 | $A_2$ | $A_1$ | $A_0$ |       |       |

- 2) Number of bytes: 2
- 3) Number of states: 11 (8)
- 4) Function:  $A \wedge (rpa)$ ; Skip if no zero

And the contents of the memory location addressed by register pair rpa (BC, DE, HL, DE+, HL+, DE-, or HL-) indicated by  $A_2 A_1 A_0$  (1-7) with the accumulator. Skip if zero is not returned as the result of AND.

- 5) Affected flags: Z, SK, L1 + 0, L0 - 0
- 6) Example:

OFFAX rpa (Off-Test Memory addressed by Register Pair with A)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

1) Operation code

|   |   |   |   |   |       |       |       |
|---|---|---|---|---|-------|-------|-------|
| 1 | 1 | 0 | 1 | 1 | $A_2$ | $A_1$ | $A_0$ |
|---|---|---|---|---|-------|-------|-------|

- 2) Number of bytes: 2

- 3) Number of states: 11 (8)

- 4) Function:  $A \wedge (rpa); \text{Skip if zero}$

OR the contents of the memory location addressed by register pair rpa (BC, DE, HL, DE+, HL+, DE-, or HL-) indicated by  $A_2A_1A_0$  (1-7) with the accumulator. Skip if zero is returned as the result of OR.

- 5) Affected flags: Z, SK, L1  $\leftarrow$  0, L0  $\leftarrow$  0

- 6) Example:

#### 13.6.5 Immediate data operation instructions

ADI A, byte (Add Immediate to A)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

Data

- 2) Number of bytes: 2

- 3) Number of states: 7 (7)

- 4) Function:  $A \leftarrow A + \text{byte}$

Add the immediate data of the second byte to the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK  $\leftarrow$  0, HC, L1  $\leftarrow$  0, L0  $\leftarrow$  0, CY

- 6) Example:

**ADI r, byte (Add Immediate to Register)**

- 1) Operation code:

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |       |       |       |
|---|---|---|---|---|-------|-------|-------|
| 0 | 1 | 0 | 0 | 0 | $R_2$ | $R_1$ | $R_0$ |
|---|---|---|---|---|-------|-------|-------|

|      |
|------|
| Data |
|------|

- 2) Number of bytes: 3

- 3) Number of states: 11 (11)

- 4) Function:  $r \leftarrow r + \text{byte}$

Add the immediate data of the third byte to the contents of register  $r$  (V, A, B, C, D, E, H, or L) addressed by  $R_2 R_1 R_0$  (0-7). Save the result in the specified register.

- 5) Affected flags: Z, SK  $\leftarrow 0$ , HC, L1  $\leftarrow 0$ , L0  $\leftarrow 0$ , CY

- 6) Example:

**ADI sr2, byte (Add Immediate to Special Register)**

- 1) Operation code:

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|       |   |   |   |   |       |       |       |
|-------|---|---|---|---|-------|-------|-------|
| $S_3$ | 1 | 0 | 0 | 0 | $S_2$ | $S_1$ | $S_0$ |
|-------|---|---|---|---|-------|-------|-------|

|      |
|------|
| Data |
|------|

- 2) Number of bytes: 3

- 3) Number of states: 20 (11)

- 4) Function:  $sr2 \leftarrow sr2 + \text{byte}$

Add the immediate data of the third byte to the contents of special register sr2 (PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, or TMM) addressed by  $S_3S_2S_1S_0$  (0-3, 5-9, B, D). Save the result in the specified special register.

- 5) Affected flags: Z, SK ← 0, HC, L1 ← 0, L0 ← 0, CY
- 6) Example:

ACI A, byte (Add Immediate to A with Carry)

1) Operation code      

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

Data

2) Number of bytes: 2

3) Number of states: 7 (7)

4) Function: A ← A+byte+CY

Add the immediate data of the second byte with the CY flag to the accumulator. Save the result in the accumulator.

5) Affected flags: Z, SK ← 0, HC, L1 ← 0, L0 ← 0, CY

6) Example:

ACI r, byte (Add Immediate to Register with Carry)

1) Operation code      

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

0 1 0 1 0 R<sub>2</sub> R<sub>1</sub> R<sub>0</sub>

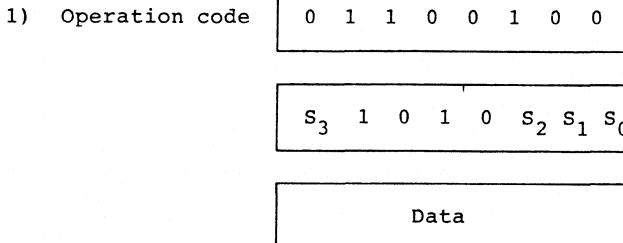
Data

- 2) Number of bytes: 3
- 3) Number of states: 11 (11)
- 4) Function:  $r \leftarrow r + \text{byte} + CY$

Add the immediate data of the third byte with the CY flag to the contents of register r (V, A, B, C, D, E, H, or L) addressed by  $R_2 R_1 R_0$  (0-7). Save the result in the specified register.

- 5) Affected flags: Z, SK  $\leftarrow$  0, HC, L1  $\leftarrow$  0, L0  $\leftarrow$  0, CY
- 6) Example:

ACI sr2, byte (Add Immediate to Special Register with Carry)



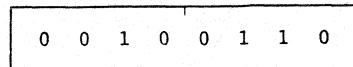
- 2) Number of bytes: 3
- 3) Number of states: 20 (11)
- 4) Function:  $sr2 \leftarrow sr2 + \text{byte} + CY$

Add the immediate data of the third byte with the CY flag to the contents of special register sr2 (PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, or TMM) addressed by  $S_3 S_2 S_1 S_0$  (0-3, 5-9, B, D). Save the result in the specified special register.

- 5) Affected flags: Z, SK  $\leftarrow$  0, HC, L1  $\leftarrow$  0, L0  $\leftarrow$  0, CY
- 6) Example:

ADINC A, byte (Add Immediate to A. Skip if No Carry)

- 1) Operation code.



Data

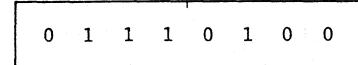
- 2) Number of bytes: 2  
3) Number of states: 7 (7)  
4) Function:  $A \leftarrow A + \text{byte}; \text{Skip if no carry}$

Add the immediate data of the second byte to the accumulator. Save the result in the accumulator. Skip if there is no carry as the result of addition.

- 5) Affected flags: Z, SK, HC, L1  $\leftarrow 0$ , L0  $\leftarrow 0$ , CY  
6) Example: ADINC A, 0A3H; A  $\leftarrow A + 0A3H$

ADINC r, byte (Add Immediate to Register. Skip if No Carry)

- 1) Operation code



0 0 1 0 0 R<sub>2</sub> R<sub>1</sub> R<sub>0</sub>

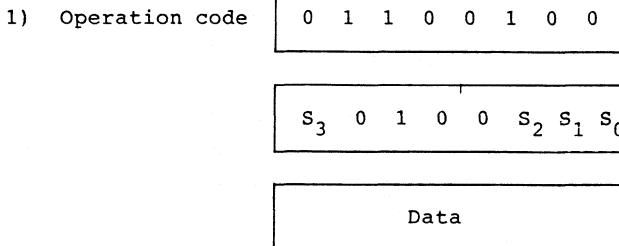
Data

- 2) Number of bytes: 3  
3) Number of states: 11 (11)  
4) Function:  $r \leftarrow r + \text{byte}; \text{Skip if no carry}$

Add the immediate data of the third byte to the contents of register r (V, A, B, C, D, E, H, or L) addressed by R<sub>2</sub>R<sub>1</sub>R<sub>0</sub> (0-7). Save the result in the specified register. Skip if there is no carry as the result of addition.

- 5) Affected flags: Z, SK, HC, L1 ← 0, L0 ← 0, CY
- 6) Example: Add immediate data to HL register pair.  
ADING L, IMM; L ← L+IMM, SKIP IF NO CARRY  
ADI H, 01H; H ← H+1

ADINC sr2, byte (Add Immediate with Special Register. Skip if No Carry)

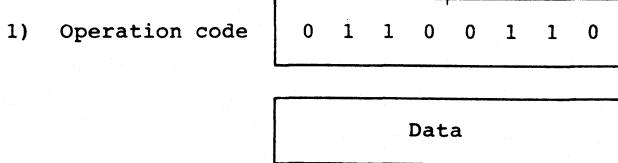


- 2) Number of bytes: 3
- 3) Number of states: 20 (11)
- 4) Function: sr2 ← sr2+byte; Skip if no carry

Add the immediate data of the third byte to the contents of special register sr2 (PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, or TMM) addressed by S<sub>3</sub>S<sub>2</sub>S<sub>1</sub>S<sub>0</sub> (0-3, 5-9, B, D). Save the result in the specified special register. Skip if there is no carry as the result of addition.

- 5) Affected flags: Z, SK, HC, L1 ← 0, L0 ← 0, CY
- 6) Example:

SUI A, byte (Subtract Immediate from A)

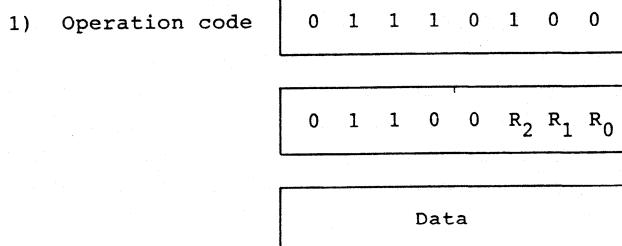


- 2) Number of bytes: 2
- 3) Number of states: 7 (7)
- 4) Function: A  $\leftarrow$  A-byte

Subtract the immediate data of the second byte from the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK  $\leftarrow$  0, HC, L1  $\leftarrow$  0, L0  $\leftarrow$  0, CY
- 6) Example:

SUI r, byte (Subtract Immediate from Register)



- 2) Number of bytes: 3
- 3) Number of states: 11 (11)
- 4) Function: r  $\leftarrow$  r-byte

Subtract the immediate data of the third byte from the contents of register r (V, A, B, C, D, E, H, or L) addressed by R<sub>2</sub>R<sub>1</sub>R<sub>0</sub> (0-7). Save the result in the specified register.

- 5) Affected flags: Z, SK  $\leftarrow$  0, HC, L1  $\leftarrow$  0, L0  $\leftarrow$  0, CY
- 6) Example:

SUI sr2, byte (Subtract Immediate from Special Register)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|       |   |   |   |   |       |       |       |
|-------|---|---|---|---|-------|-------|-------|
| $s_3$ | 1 | 1 | 0 | 0 | $s_2$ | $s_1$ | $s_0$ |
|-------|---|---|---|---|-------|-------|-------|

|      |
|------|
| Data |
|------|

- 2) Number of bytes: 3

- 3) Number of states: 20 (11)

- 4) Function:  $sr2 \leftarrow sr2 - \text{byte}$

Subtract the immediate data of the third byte from the contents of special register sr2 (PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, or TMM) addressed by  $s_3 s_2 s_1 s_0$  (0-3, 5-9, B, D). Save the result in the specified special register.

- 5) Affected flags: Z, SK  $\leftarrow$  0, HC, L1  $\leftarrow$  0, L0  $\leftarrow$  0, CY

- 6) Example:

SBI A, byte (Subtract Immediate from A with Borrow)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

|      |
|------|
| Data |
|------|

- 2) Number of bytes: 2

- 3) Number of states: 7 (7)

- 4) Function:  $A \leftarrow A - \text{byte} - CY$

Subtract the immediate data of the second byte and the CY flag from the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK ← 0, HC, L1 ← 0, L0 ← 0, CY
- 6) Example: SBI A, 30H; A ← A-30H-CY  
Subtract 30H and CY flag from A.

SBI r, byte (Subtract Immediate from Register with Borrow)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |                |                |                |
|---|---|---|---|---|----------------|----------------|----------------|
| 0 | 1 | 1 | 1 | 0 | R <sub>2</sub> | R <sub>1</sub> | R <sub>0</sub> |
|---|---|---|---|---|----------------|----------------|----------------|

|      |
|------|
| Data |
|------|

- 2) Number of bytes: 3
- 3) Number of states: 11 (11)
- 4) Function: r ← r-byte-CY

Subtract the immediate data of the third byte and the CY flag from the contents of register r (V, A, B, C, D, E, H, or L) addressed by R<sub>2</sub>R<sub>1</sub>R<sub>0</sub> (0-7). Save the result in the specified register.

- 5) Affected flags: Z, SK ← 0, HC, L1 ← 0, L0 ← 0, CY
- 6) Example:

SBI sr2, byte (Subtract Immediate from Special Register with Borrow)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

- 2) Number of bytes: 3

|       |   |   |   |   |       |       |       |
|-------|---|---|---|---|-------|-------|-------|
| $s_3$ | 1 | 1 | 1 | 0 | $s_2$ | $s_1$ | $s_0$ |
|-------|---|---|---|---|-------|-------|-------|

Data

- 3) Number of states: 20 (11)  
4) Function:  $sr2 \leftarrow sr2 - \text{byte} - CY$

Subtract the immediate data of the third byte and the CY flag from the contents of register sr2 (PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, or TMM) addressed by  $s_3 s_2 s_1 s_0$  (0-3, 5-9, B, D). Save the result in the specified special register.

- 5) Affected flags: Z, SK  $\leftarrow$  0, HC, L1  $\leftarrow$  0, L0  $\leftarrow$  0, CY  
6) Example:

SUINB A, byte (Subtract Immediate from A. Skip if No Borrow)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|

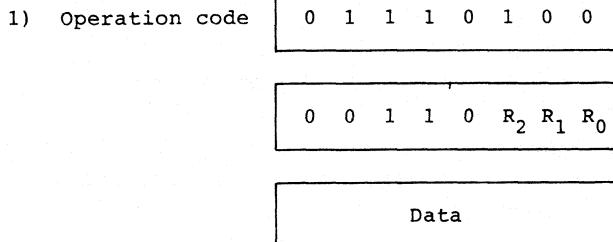
Data

- 2) Number of bytes: 2  
3) Number of states: 7 (7)  
4) Function:  $A \leftarrow A - \text{byte}; \text{Skip if no borrow}$

Subtract the immediate data of the second byte from the accumulator. Save the result in the accumulator. Skip if there is no borrow as the result of subtraction.

- 5) Affected flags: Z, SK, HC, L1 + 0, L0 + 0, CY
- 6) Example:

SUINB r, byte (Subtract Immediate from Register. Skip if No Borrow)



- 2) Number of bytes: 3
- 3) Number of states: 11 (11)
- 4) Function: r + r-byte; Skip if no borrow

Subtract the immediate data of the third byte from the contents of register r (V, A, B, C, D, E, H, or L) addressed by R<sub>2</sub>R<sub>1</sub>R<sub>0</sub> (0-7). Save the result in the specified register. Skip if There is no borrow as the result of subtraction.

- 5) Affected flags: Z, SK, HC, L1 + 0, L0 + 0, CY
- 6) Example: Subtract immediate data from HL register pair.  
SUINB L, IMM; L + L-IMM, SKIP IF NO BORROW  
SUI H, 01H; H + H-1

SUINB sr2, byte (Subtract Immediate from Special Register.)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|                |   |   |   |   |                |                |                |
|----------------|---|---|---|---|----------------|----------------|----------------|
| S <sub>3</sub> | 0 | 1 | 1 | 0 | S <sub>2</sub> | S <sub>1</sub> | S <sub>0</sub> |
|----------------|---|---|---|---|----------------|----------------|----------------|

|      |
|------|
| Data |
|------|

- 2) Number of bytes: 3

- 3) Number of states: 20 (11)

- 4) Function: sr2 ← sr2-byte; Skip if no borrow

Subtract the immediate data of the third byte from the contents of register sr2 (PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, or TMM) addressed by S<sub>3</sub>S<sub>2</sub>S<sub>1</sub>S<sub>0</sub> (0-3, 5-9, B, D). Save the result in the specified special register. Skip if there is no borrow as the result of subtraction.

- 5) Affected flags: Z, SK, HC, L1 ← 0, L0 ← 0, CY

- 6) Example: GENSU EQU 10H

SUINB PA, GENSU; PA ← PA-10H

Subtract GENSU defined in EQU from the port A contents. Save the result in port A.

ANI A, byte (And Immediate with A)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

|      |
|------|
| Data |
|------|

- 2) Number of bytes: 2

- 3) Number of states: 7 (7)

- 4) Function: A ← A&byte

AND the immediate data of the second byte with the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK ← 0, L1 ← 0, LO ← 0
- 6) Example:

ANI r, byte (And Immediate with Register)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

- 2) Number of bytes: 3

|   |   |   |   |   |                |                |                |
|---|---|---|---|---|----------------|----------------|----------------|
| 0 | 0 | 0 | 0 | 1 | R <sub>2</sub> | R <sub>1</sub> | R <sub>0</sub> |
|---|---|---|---|---|----------------|----------------|----------------|

- 3) Number of states: 11 (11)

|      |
|------|
| Data |
|------|

- 4) Function: r ← r ∧ byte

AND the immediate data of the third byte with the contents of register r (V, A, B, C, D, E, H, or L) addressed by R<sub>2</sub>R<sub>1</sub>R<sub>0</sub> (0-7). Save the result in the specified register.

- 5) Affected flags: Z, SK ← 0, L1 ← 0, LO ← 0
- 6) Example:

ANI sr2, byte (And Immediate with Special Register)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

- 2) Number of bytes: 3

|                |   |   |   |   |                |                |                |
|----------------|---|---|---|---|----------------|----------------|----------------|
| S <sub>3</sub> | 0 | 0 | 0 | 1 | S <sub>2</sub> | S <sub>1</sub> | S <sub>0</sub> |
|----------------|---|---|---|---|----------------|----------------|----------------|

- 3) Number of states: 11 (11)

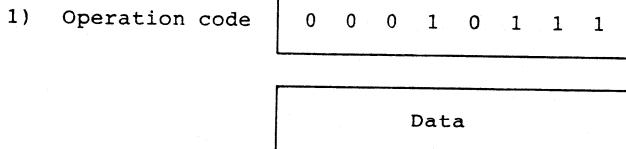
|      |
|------|
| Data |
|------|

- 2) Number of bytes: 3
- 3) Number of states: 20 (11)
- 4) Function:  $sr_2 \leftarrow sr_2 \wedge \text{byte}$

AND the immediate data of the third byte with the contents of register  $sr_2$  (PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, or TMM) addressed by  $S_3S_2S_1S_0$  (0-3, 5-9, B, D). Save the result in the specified special register.

- 5) Affected flags: Z, SK  $\leftarrow$  0, L1  $\leftarrow$  0, L0  $\leftarrow$  0
- 6) Example: Reset port B bit 2 (PB2)  
ANI PB, 0FBH; PB  $\leftarrow$  PB  $\wedge$  11111011

ORI A, byte (Or Immediate with A)



- 2) Number of bytes: 2
- 3) Number of states: 7 (7)
- 4) Function:  $A \leftarrow A \vee \text{byte}$

OR the immediate data of the second byte with the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK  $\leftarrow$  0, L1  $\leftarrow$  0, L0  $\leftarrow$  0
- 6) Example:

ORI r, byte (Or Immediate with Register)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

- 2) Number of bytes: 3

|   |   |   |   |   |                |                |                |
|---|---|---|---|---|----------------|----------------|----------------|
| 0 | 0 | 0 | 1 | 1 | R <sub>2</sub> | R <sub>1</sub> | R <sub>0</sub> |
|---|---|---|---|---|----------------|----------------|----------------|

- 3) Number of states: 11 (11)

|      |
|------|
| Data |
|------|

- 4) Function: r ← r V byte

OR the immediate data of the third byte with the contents of register r (V, A, B, C, D, E, H, or L) addressed by R<sub>2</sub>R<sub>1</sub>R<sub>0</sub> (0-7). Save the result in the specified register.

- 5) Affected flags: Z, SK ← 0, L1 ← 0, LO ← 0

- 6) Example:

ORI sr2, byte (Or Immediate with Special Register)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

- 2) Number of bytes: 3

|                |   |   |   |   |                |                |                |
|----------------|---|---|---|---|----------------|----------------|----------------|
| S <sub>3</sub> | 0 | 0 | 1 | 1 | S <sub>2</sub> | S <sub>1</sub> | S <sub>0</sub> |
|----------------|---|---|---|---|----------------|----------------|----------------|

- 3) Number of states: 20 (11)

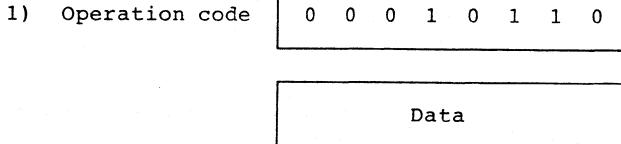
|      |
|------|
| Data |
|------|

- 4) Function: sr 2 ← sr 2 V byte

OR the immediate data of the third byte with the contents of special register sr2 (PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, or TMM) addressed by  $S_3S_2S_1S_0$  (0-3, 5-9, B, D). Save the result in the specified special register.

- 5) Affected flags: Z, SK  $\leftarrow$  0, L1  $\leftarrow$  0, L0  $\leftarrow$  0
- 6) Example: Set port C bit 1 (PC1)  
ORI PC, 02H;PC  $\leftarrow$  PC V 00000010

XRI A, byte (Exclusive-OR Immediate with A)

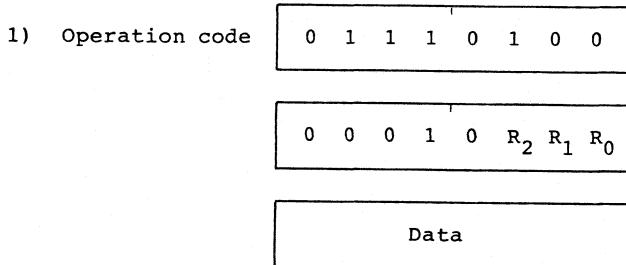


- 2) Number of bytes: 2
- 3) Number of states: 7 (7)
- 4) Function: A  $\leftarrow$  A  $\vee$  byte

Exclusive-OR the immediate data of the second byte with the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK  $\leftarrow$  0, L1  $\leftarrow$  0, L0  $\leftarrow$  0, CY
- 6) Example: XRI A, 8BH;A  $\leftarrow$  A  $\vee$  8BH

XRI r, byte (Exclusive-Or Immediate with Register)

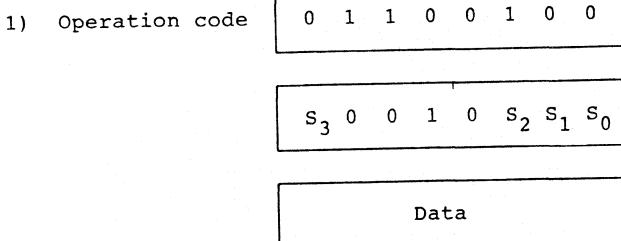


- 2) Number of bytes: 3
- 3) Number of states: 11 (11)
- 4) Function:  $r \leftarrow r \vee \text{byte}$

Exclusive-OR the immediate data of the third byte with the contents of register  $r$  (V, A, B, C, D, E, H, or L) addressed by  $R_2 R_1 R_0$  (0-7). Save the result in the specified register.

- 5) Affected flags: Z, SK  $\leftarrow$  0, L1  $\leftarrow$  0, LO  $\leftarrow$  0
- 6) Example:

XRI sr2, byte (Exclusive-Or Immediate with Special Register)



- 2) Number of bytes: 3
- 3) Number of states: 20 (11)
- 4) Function:  $sr\ 2 \leftarrow sr\ 2 \vee \text{byte}$

Exclusive-OR the immediate data of the third byte with the contents of special register sr2 (PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, or TMM) addressed by  $S_3 S_2 S_1 S_0$  (0-3, 5-9, B, D). Save the result in the specified special register.

- 5) Affected flags: Z, SK  $\leftarrow$  0, L1  $\leftarrow$  0, LO  $\leftarrow$  0
- 6) Example: Invert port A bit 2 (PA2).

XRI PA, 04H; PA  $\leftarrow$  PA  $\vee$  00000100

**GTI A, byte (Greater Than Immediate)**

- 1) Operation code      

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

Data

- 2) Number of bytes: 2  
3) Number of states: 7 (7)  
4) Function: A-byte-1; Skip if no borrow

Subtract the immediate data of the second byte and 1 from the accumulator. Skip if there is no borrow as the result of subtraction ( $A > \text{byte}$ ).

- 5) Affected flags: Z, SK, HC, L1 ← 0, L0 ← 0, CY  
6) Example:

**GTI r, byte (Greater Than Immediate)**

- 1) Operation code      

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

0 0 1 0 1 R<sub>2</sub> R<sub>1</sub> R<sub>0</sub>

Data

- 2) Number of bytes: 3  
3) Number of states: 11 (11)  
4) Function: r-byte-1; Skip if no borrow

Subtract the immediate data of the third byte and 1 from the contents of register r (V, A, B, C, D, E, H, or L) addressed by R<sub>2</sub>R<sub>1</sub>R<sub>0</sub> (0-7). Skip if there is no borrow as the result of subtraction ( $r > \text{byte}$ ).

- 5) Affected flags: z, SK, HC, L1 ← 0, L0 ← 0, CY
- 6) Example:

GTI sr2, byte (Greater Than Immediate)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|                |   |   |   |   |                |                |                |
|----------------|---|---|---|---|----------------|----------------|----------------|
| S <sub>3</sub> | 0 | 1 | 0 | 1 | S <sub>2</sub> | S <sub>1</sub> | S <sub>0</sub> |
|----------------|---|---|---|---|----------------|----------------|----------------|

|      |
|------|
| Data |
|------|

- 2) Number of bytes: 3
- 3) Number of states: 14 (11)
- 4) Function: sr2-byte-1; Skip if no borrow

Subtract the immediate data of the third byte and 1 from the contents of register sr2 (PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, or TMM) addressed by S<sub>3</sub>S<sub>2</sub>S<sub>1</sub>S<sub>0</sub> (0-3, 5-9, B, D). Skip if there is no borrow as the result of subtraction (sr2 byte).

- 5) Affected flags: z, SK, HC, L1 ← 0, L0 ← 0, CY
- 6) Example:

LTI A, byte (Less Than Immediate)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

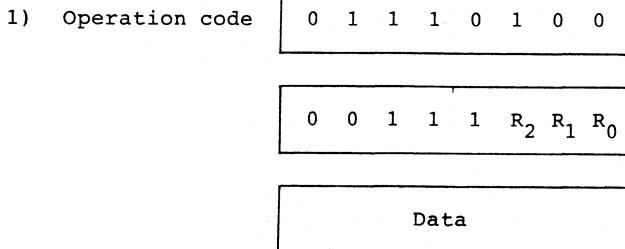
|      |
|------|
| Data |
|------|

- 2) Number of bytes: 2
- 3) Number of states: 7 (7)
- 4) Function: A-byte; Skip if borrow

Subtract the immediate data of the second byte from the accumulator. Skip if there is a borrow as the result of subtraction (A byte).

- 5) Affected flags: Z, SK, HC, L1 ← 0, L0 ← 0, CY
- 6) Example:

LTI r, byte (Less Than Immediate)



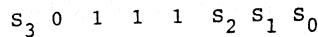
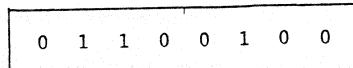
- 2) Number of bytes: 3
- 3) Number of states: 11 (11)
- 4) Function: r-byte; Skip if borrow

Subtract the immediate data of the third byte from the contents of register r (V, A, B, C, D, E, H, or L) addressed by R<sub>2</sub>R<sub>1</sub>R<sub>0</sub> (0-7). Skip if there is a borrow as the result of subtraction (r byte).

- 5) Affected flags: Z, SK, HC, L1 ← 0, L0 ← 0, CY
- 6) Example:

LTI sr2, byte (Less Than Immediate)

- 1) Operation code



Data

- 2) Number of bytes: 3

- 3) Number of states: 14 (11)

- 4) Function: sr2-byte; Skip if borrow

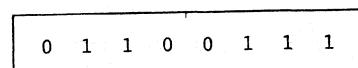
Subtract the immediate data of the third byte from the contents of special register sr2 (PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, or TMM) addressed by  $S_3S_2S_1S_0$  (0-3, 5-9, B, D). Skip if there is a borrow as the result of subtraction (sr2 byte).

- 5) Affected flags: Z, SK, HC, L1 + 0, L0 + 0, CY

- 6) Example:

NEI A, byte (Not Equal Immediate with A)

- 1) Operation code



Data

- 2) Number of bytes: 2

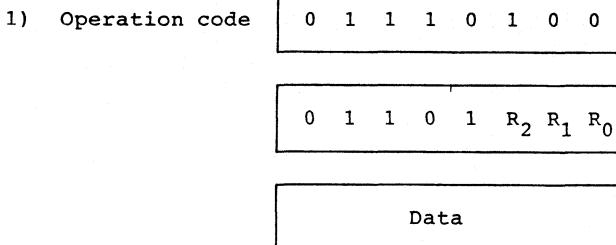
- 3) Number of states: 7 (7)

- 4) Function: A-byte; Skip if no zero

Subtract the immediate data of the second byte from the accumulator. Skip if zero is not returned as the result of subtraction (A=byte).

- 5) Affected flags: Z, SK, HC, L1 ← 0, L0 ← 0, CY
- 6) Example:

NEI r, byte (Not Equal Immediate with Register)

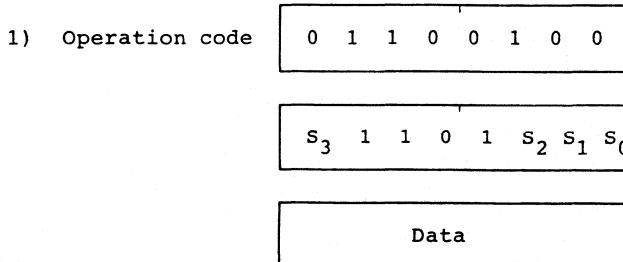


- 2) Number of bytes: 3
- 3) Number of states: 11 (11)
- 4) Function: r-byte; Skip if no zero

Subtract the immediate data of the third byte from the contents of register r (V, A, B, C, D, E, H, or L) addressed by R<sub>2</sub>R<sub>1</sub>R<sub>0</sub> (0-7). Skip if zero is not returned as the result of subtraction (r=byte).

- 5) Affected flags: Z, SK, HC, L1 ← 0, L0 ← 0, CY
- 6) Example:

NEI sr2, byte (Not Equal Immediate with Special Register)

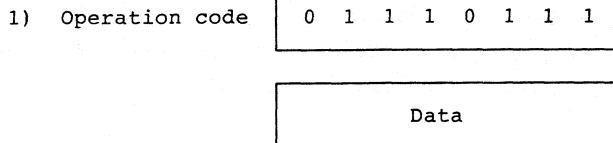


- 2) Number of bytes: 3
- 3) Number of states: 20 (11)
- 4) Function: sr2-byte; Skip if no zero

Subtract the immediate data of the third byte from the contents of special register sr2 (PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, or TMM) addressed by  $S_2S_1S_0$  (0-3, 5-9, B, D). Skip if zero is not returned as the result of subtraction (sr2=byte).

- 5) Affected flags: Z, SK, HC, L1 ← 0, L0 ← 0, CY
- 6) Example:

EQI A, byte (Equal Immediate with A)



- 2) Number of bytes: 2
- 3) Number of states: 7 (7)
- 4) Function: A-byte; Skip if zero

Subtract the immediate data of the second byte from the accumulator. Skip if zero is returned as the result of subtraction (A=byte).

- 5) Affected flags: Z, SK, HC, L1 ← 0, L0 ← 0, CY
- 6) Example:

**EQI r, byte (Equal Immediate with Register)**

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

- 2) Number of bytes: 3

|   |   |   |   |   |                |                |                |
|---|---|---|---|---|----------------|----------------|----------------|
| 0 | 1 | 1 | 1 | 1 | R <sub>2</sub> | R <sub>1</sub> | R <sub>0</sub> |
|---|---|---|---|---|----------------|----------------|----------------|

Data

- 3) Number of states: 11 (11)

- 4) Function: r-byte; Skip if zero

Subtract the immediate data of the third byte from the contents of register r (V, A, B, C, D, E, H, or L) addressed by R<sub>2</sub>R<sub>1</sub>R<sub>0</sub> (0-7). Skip if zero is returned as the result of subtraction (r=byte).

- 5) Affected flags: Z, SK, HC, L1 ← 0, LO ← 0, CY

- 6) Example:

**EQI sr2, byte (Equal Immediate with Special Register)**

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|                |   |   |   |   |                |                |                |
|----------------|---|---|---|---|----------------|----------------|----------------|
| S <sub>3</sub> | 1 | 1 | 1 | 1 | S <sub>2</sub> | S <sub>1</sub> | S <sub>0</sub> |
|----------------|---|---|---|---|----------------|----------------|----------------|

Data

- 2) Number of bytes: 3

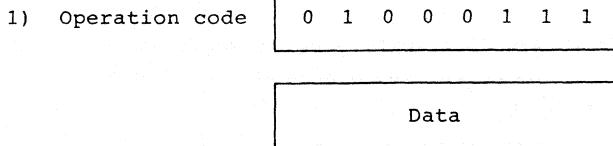
- 3) Number of states: 14 (11)

- 4) Function: sr2-byte; Skip if zero

Subtract the immediate data of the third byte from the contents of special register sr2 (PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, or TMM) addressed by  $S_3S_2S_1S_0$  (0-3, 5-9, B, D). Skip if zero is returned as the result of subtraction (sr2=byte).

- 5) Affected flags: Z, SK, HC, L1  $\leftarrow$  0, L0  $\leftarrow$  0, CY
- 6) Example:

ONI A, byte (On-Test Immediate with A)

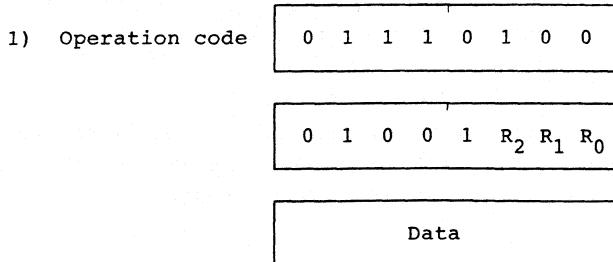


- 2) Number of bytes: 2
- 3) Number of states: 7 (7)
- 4) Function:  $A \wedge$  byte; Skip if no zero

AND the immediate data of the second byte with the accumulator. Skip if zero is not returned as the result of AND.

- 5) Affected flags: Z, SK, L1  $\leftarrow$  0, L0  $\leftarrow$  0
- 6) Example:

ONI r, byte (On-Test Immediate with Register)

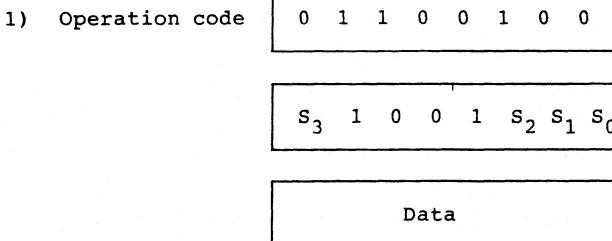


- 2) Number of bytes: 3
- 3) Number of states: 11 (11)
- 4) Function:  $r \wedge$  byte; Skip if no zero

AND the immediate data of the third byte with the contents of register r (V, A, B, C, D, E, H, or L) addressed by  $R_2 R_1 R_0$  (0-7). Skip if zero is not returned as the result of AND.

- 5) Affected flags: Z, SK, L1  $\leftarrow$  0, L0  $\leftarrow$  0
- 6) Example:

ONI sr2, byte (On-Test Immediate with Special Register)



- 2) Number of bytes: 3
- 3) Number of states: 14 (11)
- 4) Function: sr2  $\wedge$  byte; Skip if no zero

AND the immediate data of the third byte with the contents of special register sr2 (PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, or TMM) addressed by S<sub>3</sub>S<sub>2</sub>S<sub>1</sub>S<sub>0</sub> (0-3, 5-9, B, D). Skip if zero is not returned as the result of AND.

- 5) Affected flags: Z, SK, L1  $\leftarrow$  0, L0  $\leftarrow$  0
- 6) Example: Test port C bit 0 (PC0). If the bit is set to 0, jump to XX; if 1 (on), skip and execute the next instruction. ONI PC, 01H;PC 00000001  
JMP XX

**OFFI A, byte (Off-Test Immediate with A)**

- 1) Operation code

0 1 0 1 0 1 1 1

- 2) Number of bytes: 2
- 3) Number of states: 7 (7)
- 4) Function:  $A \wedge \text{byte}; \text{Skip if zero}$

AND the immediate data of the second byte from the accumulator. Skip if zero is returned as the result of AND.

- 5) Affected flags: Z, SK, L1 ← 0, L0 ← 0
- 6) Example:

**OFFI r, byte (Off-Test Immediate with Register)**

- 1) Operation code

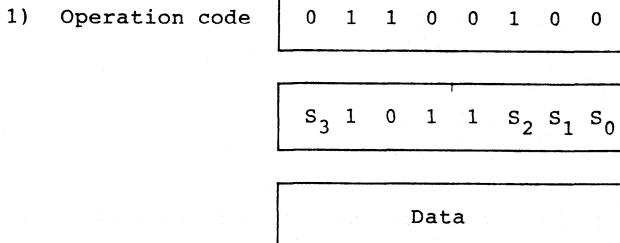
0 1 1 1 0 1 0 0

- 2) Number of bytes: 3
- 3) Number of states: 11 (11)
- 4) Function:  $r \wedge \text{byte}; \text{Skip if zero}$

AND the immediate data of the third byte with the contents of register r (V, A, B, C, D, E, H, or L) addressed by  $R_2 R_1 R_0$  (0-7). Skip if zero is returned as the result of AND.

- 5) Affected flags: Z, SK, L1 ← 0, L0 ← 0
- 6) Example:

OFFI sr2, byte (Off-Test Immediate with Special Register)



- 2) Number of bytes: 3
- 3) Number of states: 14 (11)
- 4) Function: sr2 ∧ byte; Skip if zero

AND the immediate data of the third byte with the contents of special register sr2 (PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, or TMM) addressed by S<sub>2</sub>S<sub>1</sub>S<sub>0</sub> (0-3, 5-9, B, D). Skip if zero is returned as the result of AND.

- 5) Affected flags: Z, SK, L1 ← 0, L0 ← 0
- 6) Example:

## 13.6.6 Working register operation instructions

## ADDW wa (Add Working Register to A)

1) Operation code

0 1 1 1 0 1 0 0

1 1 0 0 0 0 0 0

Offset

2) Number of bytes: 3

3) Number of states: 14 (11)

4) Function:  $A \leftarrow A + (V.wa)$ 

Add the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the third byte (low-order eight bits) to the accumulator. Save the result in the accumulator.

5) Affected flags: Z, SK  $\leftarrow$  0, HC, L1  $\leftarrow$  0, L0  $\leftarrow$  0, CY

6) Example:

## ADCW wa (Add Working Register to A with Carry)

1) Operation code

0 1 1 1 0 1 0 0

1 1 0 1 0 0 0 0

Offset

2) Number of bytes: 3

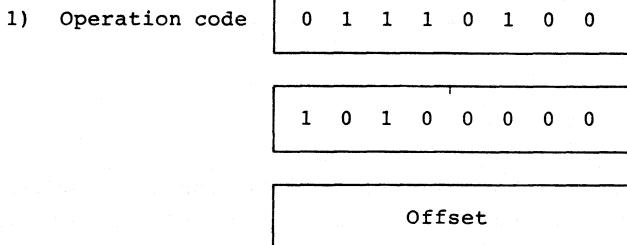
3) Number of states: 14 (11)

4) Function:  $A \leftarrow A + (V.wa) + CY$

Add the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the third byte (low-order eight bits) with the CY flag to the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK ← 0, HC, L1 ← 0, L0 ← 0, CY
- 6) Example:

ADDNCW wa (Add Working Register to A. Skip If No Carry)



- 2) Number of bytes: 3
- 3) Number of states: 14 (11)
- 4) Function: A ← A + (V.wa); Skip if no carry

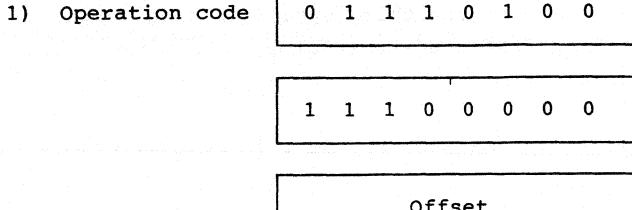
Add the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the third byte to the accumulator. Save the result in the accumulator. Skip if there is no carry as the result of addition.

- 5) Affected flags: Z, SK, HC, L1 ← 0, L0 ← 0, CY
- 6) Example:

**SUBW wa (Subtract Working Register from A)**

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|



Offset

- 2) Number of bytes: 3

- 3) Number of states: 14 (11)

- 4) Function:
- $A \leftarrow A - (V.wa)$

Subtract the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the third byte from the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK
- $\leftarrow$
- 0, HC, L1
- $\leftarrow$
- 0, L0
- $\leftarrow$
- 0, CY

- 6) Example:

**SBBW wa (Subtract Working Register From A with Borrow)**

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

Offset

- 2) Number of bytes: 3

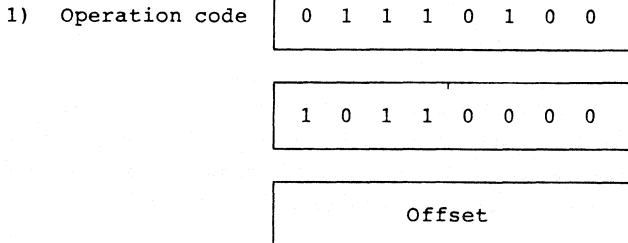
- 3) Number of states: 14 (11)

- 4) Function:
- $A \leftarrow A - (V.wa) - CY$

Subtract the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the third byte (low-order eight bits) and the CY flag from the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK ← 0, HC, L1 ← 0, L0 ← 0, CY
- 6) Example:

SUBNBW wa (Subtract Working Register from A. Skip if No Borrow)



- 2) Number of bytes: 3
- 3) Number of states: 14 (11)
- 4) Function: A ← A - (V.wa); Skip if no borrow

Subtract the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the third byte (low-order eight bits) from the accumulator. Save the result in the accumulator. Skip if there is no borrow as the result of subtraction.

- 5) Affected flags: Z, SK, HC, L1 ← 0, L0 ← 0, CY
- 6) Example:

|             |        |                |
|-------------|--------|----------------|
| WORK EQU    | 0E0H   | ;WORK=E0H      |
| LOCA EQU    | 0      | ;LOCA=00H      |
| MVI         | V,WORK | ;V ← E0H       |
| SUBNBW LOCA |        | ;A ← A0(E000H) |

To execute the instruction, the high-order eight bits of the address of the working register area must have been loaded into the V register specifying the 256-byte working register area. The SUBNBW instruction operand value is used as the low-order eight bits of the address.

**ANAW wa (And Working Register with A)**

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|



|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|        |
|--------|
| Offset |
|--------|

- 2) Number of bytes: 3

- 3) Number of states: 14 (11)

- 4) Function:  $A \leftarrow A \wedge (V.wa)$

AND the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the third byte (low-order eight bits) with the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK  $\leftarrow 0$ , L1  $\leftarrow 0$ , L0  $\leftarrow 0$

- 6) Example:

**ORAW wa (Or Woring Register with A)**

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|



|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|        |
|--------|
| Offset |
|--------|

- 2) Number of bytes: 3
- 3) Number of states: 14 (11)
- 4) Function: A ← A V(V.wa)

OR the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the third byte (low-order eight bits) with the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK ← 0, L1 ← 0, L0 ← 0
- 6) Example:

**XRAW wa (Exclusive-Or Working Register with A)**

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|        |
|--------|
| Offset |
|--------|

- 2) Number of bytes: 3
- 3) Number of states: 14 (11)
- 4) Function: A ← A V(V.wa)

Exclusive-OR the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the third byte (low-order eight bits) with the accumulator. Save the result in the accumulator.

- 5) Affected flags: Z, SK ← 0, L1 ← 0, L0 ← 0
- 6) Example:

**GTAW wa (Greater Than Working Register)**

- 1) Operation code

0 1 1 1 0 1 0 0

- 2) Number of bytes: 3

- 3) Number of states: 14 (11)

- 4) Function: A-(V.wa)-1; Skip if no borrow

1 0 1 0 1 0 0 0

Offset

- Subtract the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the third byte (low-order eight bits) and 1 from the accumulator. Skip if there is no borrow as the result of subtraction ( $A > (V.wa)$ ).
- 5) Affected flags: Z, SK, HC, L1 + 0, L0 + 0, CY  
6) Example:

**LTAW wa (Less Than Working Register)**

- 1) Operation code

0 1 1 1 0 1 0 0

- 2) Number of bytes: 3

- 3) Number of states: 14 (11)

- 4) Function: A-(V.wa); Skip if borrow

1 0 1 1 1 0 0 0

Offset

Subtract the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the third byte (low-order eight bits) from the accumulator. Skip if there is a borrow as the result of subtraction ( $A < (V.wa)$ ).

- 5) Affected flags: Z, SK, HC, L1  $\leftarrow$  0, LO  $\leftarrow$  0, CY
- 6) Example:

NEAW wa (Not Equal Working Register with A)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

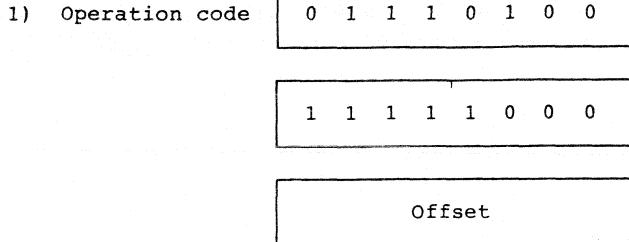
|        |
|--------|
| Offset |
|--------|

- 2) Number of bytes: 3
- 3) Number of states: 14 (11)
- 4) Function:  $A - (V.wa)$ ; Skip if not zero

Subtract the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the third byte (low-order eight bits) from the accumulator. Skip if zero is not returned as the result of subtraction ( $A \neq (V.wa)$ ).

- 5) Affected flags: Z, SK, HC, L1  $\leftarrow$  0, LO  $\leftarrow$  0, CY
- 6) Example:

## EQAW wa (Equal Working Register with A)

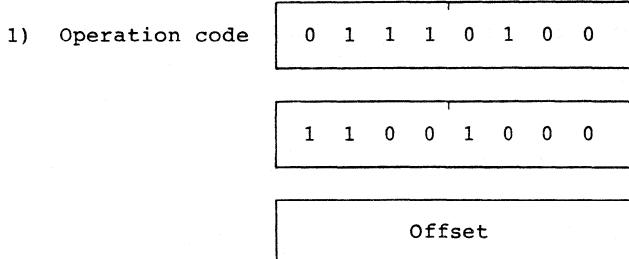


- 2) Number of bytes: 3  
3) Number of states: 14 (11)  
4) Function:  $A - (V.wa) - 1$ ; Skip if zero

Subtract the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the third byte (low-order eight bits) from the accumulator. Skip if zero is returned as the result of subtraction ( $A = (V.wa)$ ).

- 5) Affected flags: Z, SK, HC, L1  $\leftarrow$  0, L0  $\leftarrow$  0, CY  
6) Example:

## ONAW wa (On-Test Working Register with A)

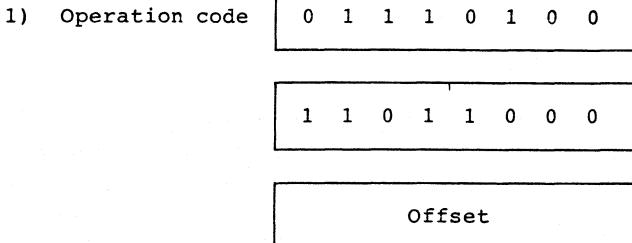


- 2) Number of bytes: 3  
3) Number of states: 14 (11)  
4) Function:  $A \Delta (V.wa)$ ; Skip if not zero

AND the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the third address (low-order four bits) with the accumulator. Skip if zero is not returned as the result of AND.

- 5) Affected flags: Z, SK, L1 ← 0, L0 ← 0
- 6) Example:

OFFAW wa (Off-Test Working Register with A)



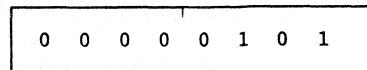
- 2) Number of bytes: 3
- 3) Number of states: 14 (11)
- 4) Function: A  $\wedge$ (V.wa); Skip if zero

AND the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the third address (low-order four bits) with the accumulator. Skip if zero is returned as the result of AND.

- 5) Affected flags: Z, SK, L1 ← 0, L0 ← 0
- 6) Example:

**ANIW wa, byte (And Immediate with Working Register)**

- 1) Operation code



Offset

Data

- 2) Number of bytes: 3

- 3) Number of states: 19 (10)

- 4) Function:
- $(V.wa) \leftarrow (V.wa) \wedge \text{byte}$

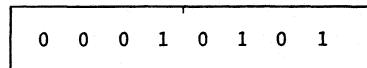
AND the immediate data of the third byte with the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the second byte (low-order eight bits). Save the result in the addressed working register.

- 5) Affected flags: Z, SK, +0, L1 +0, L0 +0

- 6) Example:

**ORIW wa, byte (Or Immediate with Working Register)**

- 1) Operation code



Offset

Data

- 2) Number of bytes: 3

- 3) Number of states: 19 (10)

- 4) Function:
- $(V.wa) \leftarrow (V.wa) \vee \text{byte}$

OR the immediate data of the third byte with the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the second byte (low-order eight bits). Save the result in the addressed working register.

- 5) Affected flags: Z, SK, ← 0, L1 ← 0, L0 ← 0
- 6) Example:

GTIW wa, byte (Greater Than Immediate)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

|         |
|---------|
| Off set |
|---------|

|      |
|------|
| Data |
|------|

- 2) Number of bytes: 3
- 3) Number of states: 13 (10)
- 4) Function: (V.wa)-byte-1; Skip if no borrow

Subtract the immediate data of the third byte and 1 from the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the second byte (low-order eight bits). Skip if there is no borrow as the result of subtraction  $((V.wa) > \text{byte})$ .

- 5) Affected flags: Z, SK, HC, L1 ← 0, L0 ← 0, CY
- 6) Example:

## LTIW wa, byte (Less Than Immediate)

- 1) Operation code

0 0 1 1 0 1 0 1

- 2) Number of bytes: 3

- 3) Number of states: 13 (10)

- 4) Function: (V.wa)-byte; Skip if borrow

Subtract the immediate data of the third byte from the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the second byte (low-order eight bits). Skip if there is a borrow as the result of subtraction  $((V.wa) < \text{byte})$ .

- 5) Affected flags: Z, SK, HC, L1
- $\leftarrow$
- 0, L0
- $\leftarrow$
- 0, CY

- 6) Example:

## NEIW WA, byte (Not Equal Immediate with Working Register)

- 1) Operation code

0 1 1 0 0 1 0 1

- 2) Number of bytes: 3

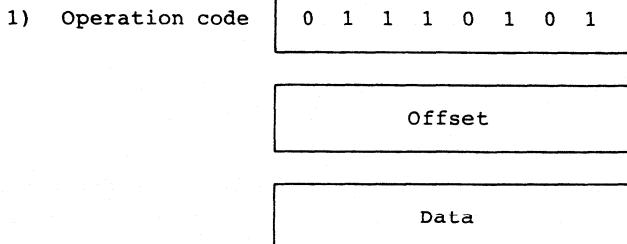
- 3) Number of states: 13 (10)

- 4) Function: (V.wa)-byte; Skip if not zero

Subtract the immediate data of the third byte with the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the second byte (low-order eight bits). Skip if zero is not returned as the result of subtraction ((V.wa)=byte).

- 5) Affected flags: Z, SK, HC, L1 + 0, L0 + 0, CY
- 6) Example:

EQIW wa,byte (Equal Immediate with Working Register)



- 2) Number of bytes: 3
- 3) Number of states: 19 (10)
- 4) Function: (V.wa)-byte; Skip if zero

Subtract the immediate data of the third byte with the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the second byte (low-order eight bits). Skip if zero is returned as the result of subtraction ((V.wa)=byte).

- 5) Affected flags: Z, SK, HC, L1 + 0, L0 + 0, CY
- 6) Example:

ONIW wa, byte (On-Test Immediate with Working Register)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

- 2) Number of bytes: 3

- 3) Number of states: 13 (10)

- 4) Function: (V.wa)  $\wedge$  byte; Skip if not zero

AND the immediate data of the third byte with the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the second byte (low-order eight bits). Skip if zero is not returned as the result of AND

- 5) Affected flags: Z, SK, L1  $\leftarrow$  0, L0  $\leftarrow$  0

- 6) Example:

OFFIW wa, byte (Off-Test Immediate with Working Register)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

- 2) Number of bytes: 3

- 3) Number of states: 13 (10)

- 4) Function: (V.wa)  $\wedge$  byte; Skip if zero

AND the immediate data of the third byte with the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the second byte (low-order eight bits). Skip if zero is returned as the result of AND.

- 5) Affected flags: Z, SK, + 0, L1 ← 0, LO ← 0
- 6) Example:

#### 13.6.7 16-bit operation instructions

EADD EA, r2 (Add Register to EA)

- 1) Operation code 

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|
- 2) Number of bytes: 2
- 3) Number of states: 11 (8)
- 4) Function: EA + EA+r2

Add the contents of register r2 (A, B, or C) addressed by R1R0 (1-3) to the contents of the low-order eight bits of the expansion accumulator. Save the result in the expansion accumulator.

- 5) Affected flags: Z, SK, + 0, HC, L1 ← 0, LO ← 0, CY
- 6) Example:

DADD EA, rp3 (Add Register Pair to EA)

- 1) Operation code 

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|
- 2) Number of bytes: 2
- 3) Number of states: 11 (8)
- 4) Function: EA + EA+rp3

- 2) Number of bytes: 2
- 3) Number of states: 11 (8)
- 4) Function: EA + EA+rp3

Add the contents of register pair rp3 (BC, DE, or HL ) addressed by P1P0 (1-3) to the expansion accumulator. Save the result in the expansion accumulator.

- 5) Affected flags: Z, SK + 0, HC, L1 + 0, L0 + 0, CY
- 6) Example:

DADC EA, rp3 (Add Register Pair to EA with Carry)

|                   |                                                                                                                                             |   |   |   |   |                |                |                |                |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------|---|---|---|---|----------------|----------------|----------------|----------------|
| 1) Operation code | <table border="1"><tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td></tr></table>                         | 0 | 1 | 1 | 1 | 0              | 1              | 0              | 0              |
| 0                 | 1                                                                                                                                           | 1 | 1 | 0 | 1 | 0              | 0              |                |                |
|                   | <table border="1"><tr><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>P<sub>1</sub></td><td>P<sub>0</sub></td></tr></table> | 1 | 1 | 0 | 1 | 0              | 1              | P <sub>1</sub> | P <sub>0</sub> |
| 1                 | 1                                                                                                                                           | 0 | 1 | 0 | 1 | P <sub>1</sub> | P <sub>0</sub> |                |                |

- 2) Number of bytes: 2
- 3) Number of states: 11 (8)
- 4) Function: EA + EA+rp3+CY

Add the contents of register pair rp3 (BC, DE, or HL ) addressed by P1P0 (1-3) with the CY flag to the expansion accumulator. Save the result in the expansion accumulator.

- 5) Affected flags: Z, SK + 0, HC, L1 + 0, L0 + 0, CY
- 6) Example:

DADDNC EA, rp3 (Add Register Pair to EA. Skip if no Carry)

|                   |                                                                                                                                             |   |   |   |   |                |                |                |                |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------|---|---|---|---|----------------|----------------|----------------|----------------|
| 1) Operation code | <table border="1"><tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td></tr></table>                         | 0 | 1 | 1 | 1 | 0              | 1              | 0              | 0              |
| 0                 | 1                                                                                                                                           | 1 | 1 | 0 | 1 | 0              | 0              |                |                |
|                   | <table border="1"><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>P<sub>1</sub></td><td>P<sub>0</sub></td></tr></table> | 1 | 0 | 1 | 0 | 0              | 1              | P <sub>1</sub> | P <sub>0</sub> |
| 1                 | 0                                                                                                                                           | 1 | 0 | 0 | 1 | P <sub>1</sub> | P <sub>0</sub> |                |                |

- 2) Number of bytes: 2
- 3) Number of states: 11 (8)
- 4) Function: EA ← EA+rp3; Skip if no Carry

Add the contents of register pair rp3 (BC, DE, or HL) addressed by P1P0 (1-3) to the expansion accumulator. Save the result in the expansion accumulator. Skip if there is no carry as the result of addition.

- 5) Affected flags: Z, SK, HC, L1 ← 0, L0 ← 0, CY
- 6) Example:

ESUB EA, r2 (Subtract Register from EA)

|                   |                                                                                                                                             |   |   |   |   |                |                |                |                |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------|---|---|---|---|----------------|----------------|----------------|----------------|
| 1) Operation code | <table border="1"><tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>                         | 0 | 1 | 1 | 1 | 0              | 0              | 0              | 0              |
| 0                 | 1                                                                                                                                           | 1 | 1 | 0 | 0 | 0              | 0              |                |                |
|                   | <table border="1"><tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>R<sub>1</sub></td><td>R<sub>0</sub></td></tr></table> | 0 | 1 | 1 | 0 | 0              | 0              | R <sub>1</sub> | R <sub>0</sub> |
| 0                 | 1                                                                                                                                           | 1 | 0 | 0 | 0 | R <sub>1</sub> | R <sub>0</sub> |                |                |

- 2) Number of bytes: 2
- 3) Number of states: 11 (8)
- 4) Function: EA ← EA-r2

Subtract the contents of register r2 (A, B, or C) addressed by R1R0 (1-3) from the expansion accumulator. Save the result in the expansion accumulator.

- 5) Affected flags: Z, SK ← 0, HC, L1 ← 0, L0 ← 0, CY
- 6) Example:

DSUB EA, rp3 (Subtract Register Pair from EA)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

- 2) Number of bytes: 2  
3) Number of states: 11 (8)  
4) Function: EA + EA - rp3

Subtract the contents of register pair (BC, DE, or HL) addressed by P1P0 (1-3) from the expansion accumulator. Save the result in the expansion accumulator.

- 5) Affected flags: Z, SK + 0, HC, L1 + 0, L0 + 0, CY  
6) Example:

DSBB EA, rp3 (Subtract Register Pair from EA with Borrow)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

- 2) Number of bytes: 2  
3) Number of states: 11 (8)  
4) Function: EA + EA - rp3 - CY

Subtract the contents of register pair (BC, DE, or HL) addressed by P1P0 (1-3) and the CY flag from the expansion accumulator. Save the result in the expansion accumulator.

- 5) Affected flags: Z, SK + 0, HC, L1 + 0, L0 + 0, CY  
6) Example:

DSUBNB EA, rp3 (Subtract Register Pair from EA. Skip if No Borrow)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |                |                |
|---|---|---|---|---|---|----------------|----------------|
| 1 | 0 | 1 | 1 | 0 | 1 | P <sub>1</sub> | P <sub>0</sub> |
|---|---|---|---|---|---|----------------|----------------|

- 2) Number of bytes: 2  
3) Number of states: 11 (8)  
4) Function: EA ← EA - rp3; Skip if no borrow

Subtract the contents of register pair rp3 (BC, DE, or HL) addressed by P1P0 (1-3) from the expansion accumulator. Save the result in the expansion accumulator. Skip if there is no borrow as the result of subtraction.

- 5) Affected flags: Z, SK, HC, L1 ← 0, L0 ← 0, CY  
6) Example:

DAN EA, rp3 (And Register pair with EA)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |                |                |
|---|---|---|---|---|---|----------------|----------------|
| 1 | 0 | 0 | 0 | 1 | 1 | P <sub>1</sub> | P <sub>0</sub> |
|---|---|---|---|---|---|----------------|----------------|

- 2) Number of bytes: 2  
3) Number of states: 11 (8)  
4) Function: EA ← EA ∧ rp3

AND the contents of register pair rp3 (BC, DE, or HL) addressed by P1P0 (1-3) with the expansion accumulator. Save the result in the expansion accumulator.

- 5) Affected flags: Z, SK ← 0, L1 ← 0, L0 ← 0  
6) Example:

DOR EA rp3 (Or Register pair with EA)

- 1) Operation code

0 1 1 1 0 1 0 0

- 2) Number of bytes: 2  
3) Number of states: 11 (8)  
4) Function: EA + EA Vrp3

OR the contents of register pair (BC, DE, or HL) addressed by P1P0 (1-3) with the expansion accumulator. Save the result in the expansion accumulator.

- 5) Affected flags: Z, SK, + 0, L1 + 0, L0 + 0
- 
- 6) Example:

DXR EA, rp3 (Exclusive-Or Register Pair with EA)

- 1) Operation code

0 1 1 1 0 1 0 0

- 2) Number of bytes: 2  
3) Number of states: 11 (8)  
4) Function: EA + EA V rp3

Exclusive-OR the contents of register pair rp3 (BC, DE, or HL) addressed by P1P0 (1-3) with the expansion accumulator. Save the result in the expansion accumulator.

- 5) Affected flags: Z, SK, + 0, L1 + 0, L0 + 0
- 
- 6) Example:

DGT EA, rp3 (Greater Than Register Pair)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |                |                |
|---|---|---|---|---|---|----------------|----------------|
| 1 | 0 | 1 | 0 | 1 | 1 | P <sub>1</sub> | P <sub>0</sub> |
|---|---|---|---|---|---|----------------|----------------|

- 2) Number of bytes: 2  
3) Number of states: 11 (8)  
4) Function: EA-rp3-1; Skip if no borrow

Subtract the contents of register pair (BC, DE, or HL) addressed by P1P0 (1-3) and 1 from the expansion accumulator. Skip if there is no borrow as the result of subtraction (EA rp3)

- 5) Affected flags: Z, SK, HC, L1 ← 0, LO ← 0, CY  
6) Example: DGT EA, B;EA-BC-1  
              Skip if EA is greater than BC.

DLT EA, rp3 (Less Than Register Pair)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |                |                |
|---|---|---|---|---|---|----------------|----------------|
| 1 | 0 | 1 | 1 | 1 | 1 | P <sub>1</sub> | P <sub>0</sub> |
|---|---|---|---|---|---|----------------|----------------|

- 2) Number of bytes: 2  
3) Number of states: 11 (8)  
4) Function: EA-rp3; Skip if borrow

Subtract the contents of register pair (BC, DE, or HL) addressed by P1P0 (1-3) from the expansion accumulator. Skip if there is a borrow as the result of subtraction (EA < rp3).

- 5) Affected flags: Z, SK, HC, L1 + 0, L0 + 0, CY
- 6) Example: DLT EA, B;EA-BC  
Skip if BC is greater than EA.

DNE EA, rp3 (Not Equal Register Pair with EA)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |                |                |
|---|---|---|---|---|---|----------------|----------------|
| 1 | 1 | 1 | 0 | 1 | 1 | P <sub>1</sub> | P <sub>0</sub> |
|---|---|---|---|---|---|----------------|----------------|

- 2) Number of bytes: 2
- 3) Number of states: 11 (8)
- 4) Function: EA-rp3; Skip if not zero

Subtract the contents of register pair (BC, DE, or HL) addressed by P1P0 (1-3) from the expansion accumulator. Skip if zero is return as the result of subtraction (EA=rp3).

- 5) Affected flags: Z, SK, HC, L1 + 0, L0 + 0, CY
- 6) Example: DNE EA, B;EA-BC  
Skip if EA does not equal BC.

DEQ EA, rp3 (Equal Register Pair with EA)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |                |                |
|---|---|---|---|---|---|----------------|----------------|
| 1 | 1 | 1 | 1 | 1 | 1 | P <sub>1</sub> | P <sub>0</sub> |
|---|---|---|---|---|---|----------------|----------------|

- 2) Number of bytes: 2
- 3) Number of states: 11 (8)
- 4) Function: EA-rp3; Skip if zero

Subtract the contents of register pair (BC, DE, or HL) addressed by P1P0 (1-3) from the expansion accumulator. Skip if zero is returned as the result of subtraction (EA=rp3).

- 5) Affected flags: Z, SK, HC, L1  $\leftarrow$  0, L0  $\leftarrow$  0, CY
- 6) Example: DEQ EA, B;EA-BC  
Skip if EA equals BC.

DON EA, rp3 (On-Test Register Pair with EA)

1) Operation code      

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |                |                |
|---|---|---|---|---|---|----------------|----------------|
| 1 | 1 | 0 | 0 | 1 | 1 | P <sub>1</sub> | P <sub>0</sub> |
|---|---|---|---|---|---|----------------|----------------|

- 2) Number of bytes: 2
- 3) Number of states: 11 (8)
- 4) Function: EA  $\wedge$  rp3; Skip if not zero

AND the contents of register pair rp3 (BC, DE, or HL) addressed by P1P0 (1-3) with the expansion accumulator. Skip if zero is returned as the result of AND.

- 5) Affected flags: Z, SK, L1  $\leftarrow$  0, L0  $\leftarrow$  0
- 6) Example:

DOFF EA, rp3 (Off-Test Register Pair with EA)

1) Operation code      

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |                |                |
|---|---|---|---|---|---|----------------|----------------|
| 1 | 1 | 0 | 1 | 1 | 1 | P <sub>1</sub> | P <sub>0</sub> |
|---|---|---|---|---|---|----------------|----------------|

- 2) Number of bytes: 2
- 3) Number of states: 11 (8)
- 4) Function: EA  $\wedge$  rp3; Skip if zero

AND the contents of register pair rp3 (BC, DE, or HL) addressed by P1P0 (1-3) with the expansion accumulator. Skip if zero is returned as the result of AND.

- 5) Affected flags: Z, SK, L1 ← 0, L0 ← 0
- 6) Example:

#### 13.6.8 Multiplication and division instructions

MUL r2 (Multiply A by Register)

1) Operation code      

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |                |                |
|---|---|---|---|---|---|----------------|----------------|
| 0 | 0 | 1 | 0 | 1 | 1 | R <sub>1</sub> | R <sub>0</sub> |
|---|---|---|---|---|---|----------------|----------------|

- 2) Number of bytes: 2
- 3) Number of states: 32 (8)
- 4) Function: EA    Axr2

Multiply the accumulator by the contents of register r2 (A, B, or C) addressed by R1R0 (1-3) with no sign. Save the result in the expansion accumulator.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0
- 6) Example:

DIV r2 (Divide EA by Register)

1) Operation code      

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |                |                |
|---|---|---|---|---|---|----------------|----------------|
| 0 | 0 | 1 | 1 | 1 | 1 | R <sub>1</sub> | R <sub>0</sub> |
|---|---|---|---|---|---|----------------|----------------|

- 2) Number of bytes: 2
- 3) Number of states: 59 (8)
- 4) Function: EA ← EA:r2, r2 ← remainder

Divide the expansion accumulator by the contents of register r2 (A, B, or C) addressed by R1R0 (1-3) with no sign. Save the quotient in the expansion accumulator and the remainder in register r2.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0
- 6) Example:

#### 13.6.9 Increment and decrement instructions

##### INR r2 (Increment Register)

- 1) Operation code 

|   |   |   |   |   |   |                |                |
|---|---|---|---|---|---|----------------|----------------|
| 0 | 1 | 0 | 0 | 0 | 0 | R <sub>1</sub> | R <sub>0</sub> |
|---|---|---|---|---|---|----------------|----------------|
- 2) Number of bytes: 1
- 3) Number of states: 4 (4)
- 4) Function: r2 ← r2+1; Skip if carry

Increment the contents of register r2 (A, B, or C) addressed by R1R0 (1-3). Skip if there is a carry as the result of increment.

- 5) Affected flags: X, SK, HC, L1 ← 0, L0 ← 0
- 6) Example: INR A ; A+A + 1

##### INRW wa (Increment Working Register)

- 1) Operation code 

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|
- 2) Number of bytes: 2
- 3) Number of states: 16 (7)
- 4) Function: (V.wa) ← (V.wa)+1; Skip if carry

Increment the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the second byte (low-order eight bits). Skip if there is a carry as the result of increment.

5) Affected flags: z, SK, HC, L1 ← 0, LO ← 0

6) Example: MVI V, 0FFH

INRW 0FFH; (FFFF) ← (FFFF)+1

Increment the contents of the working register at address FFFFH.

INX rp (Increment Register Pair)

1) Operation code      

|   |   |                |                |   |   |   |   |
|---|---|----------------|----------------|---|---|---|---|
| 0 | 0 | P <sub>1</sub> | P <sub>0</sub> | 0 | 0 | 1 | 0 |
|---|---|----------------|----------------|---|---|---|---|

2) Number of bytes: 1

3) Number of states: 7 (4)

4) Function: rp ← rp+1

Increment SP or register pair rp (BC, DE, or HL) addressed by P1P0 (0-3).

5) Affected flags: SK ← 0, L1 ← 0, LO ← 0

6) Example: INX D;DE ← DE+1

Increment 16-bit register (D=high-order eight bits and E=low-order eight bits).

INX EA (Increment EA)

1) Operation code      

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

2) Number of bytes: 1

3) Number of states: 7 (4)

- 4) Function: EA  $\leftarrow$  EA+1

Increment the expansion accumulator.

- 5) Affected flags: SK  $\leftarrow$  0, L1  $\leftarrow$  0, L0  $\leftarrow$  0

- 6) Example:

**DCR r2 (Decrement Register)**

- 1) Operation code 

|   |   |   |   |   |   |                |                |
|---|---|---|---|---|---|----------------|----------------|
| 0 | 1 | 0 | 1 | 0 | 0 | R <sub>1</sub> | R <sub>0</sub> |
|---|---|---|---|---|---|----------------|----------------|

- 2) Number of bytes: 1

- 3) Number of states: 4 (4)

- 4) Function: r2  $\leftarrow$  r2-1; Skip if borrow

Decrement the contents of register r2 (A, B, or C) addressed by R1R0 (1-3). Skip if there is a borrow as the result of decrement.

- 5) Affected flags: Z, SK, HC, L1  $\leftarrow$  0, L0  $\leftarrow$  0

- 6) Example: DCR B; B  $\leftarrow$  B-1

**DCRW wa (Decrement Working Register)**

- 1) Operation code 

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

Offset

- 2) Number of bytes: 2

- 3) Number of states: 16 (7)

- 4) Function: (V.wa)  $\leftarrow$  (V.wa)-1; Skip if borrow

Decrement the contents of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the second byte (low-order eight bits). Skip if there is a borrow as the result of decrement.

5) Affected flags: Z, SK, HC, L1 ← 0, L0 ← 0

6) Example: MVI V, 50H

DCRW 0 ;(5000H) ← (5000H)-1

Decrement the contents of the working register at address 5000H.

DCX rp (Decrement Register Pair)

1) Operation code 0 0 P<sub>1</sub> P<sub>0</sub> 0 0 1 1

2) Number of bytes: 1

3) Number of states: 7 (4)

4) Function: rp ← rp-1

Decrement SP or register pair rp (BC, DE, or HL) addressed by P1P0 (1-3)

5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0

6) Example: DCX H:HL ← HL-1

DCX EA (Decrement EA)

1) Operation code 1 0 1 0 1 0 0 1

2) Number of bytes: 1

3) Number of states: 7 (4)

4) Function: EA ← EA-1

Decrement the expansion accumulator.

5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0

6) Example:

### 13.6.10 Miscellaneous operation instructions

DAA (Decimal Adjust A)

|                   |                 |
|-------------------|-----------------|
| 1) Operation code | 0 1 1 0 0 0 0 1 |
|-------------------|-----------------|

2) Number of bytes: 1

3) Number of states: 4 (4)

4) Function:

Decide the accumulator, CY flag, and HC flag contents.

Make decimal adjustment as follows:

|                           | Condition                 | Operation                   |
|---------------------------|---------------------------|-----------------------------|
| $A_{3-0} \leq 9$<br>HC=0  | $A_{7-4} \leq 9$ and CY=0 | $A \leftarrow A$            |
|                           | $A_{7-4} \geq 10$ or CY=0 | $A \leftarrow A + 01100000$ |
| $A_{3-0} \geq 10$<br>HC=0 | $A_{7-4} < 9$ and CY=0    | $A \leftarrow A + 00000110$ |
|                           | $A_{7-4} \geq 9$ or CY=0  | $A \leftarrow A + 01100110$ |
| HC=1                      | $A_{7-4} \leq 9$ and CY=0 | $A \leftarrow A + 00000110$ |
|                           | $A_{7-4} \geq 10$ or CY=0 | $A \leftarrow A + 01100110$ |

This instruction is significant only after operation is performed on decimal (BCD) data pieces.

5) Affected Flags: X, SK  $\leftarrow$  0, HC, L1  $\leftarrow$  0, L0  $\leftarrow$  0, CY

6) Example: MVI A, 88H

ADI A, 79H; A=01H, CY=1, HC=1

DAA ;A  $\leftarrow$  A+66H, A=67H, CY=1

;88 + 79 = 167

$$\begin{array}{r}
 \text{ADI} \\
 \left. \begin{array}{r}
 10001000 \quad 88H \\
 +)01111001 \quad 79H \\
 \hline
 00000001 \quad 01H
 \end{array} \right\} \text{Carry HC} \\
 \text{DAA} \\
 \left. \begin{array}{r}
 +)01100110 \quad 66H \\
 \hline
 01100111 \quad 67H
 \end{array} \right\}
 \end{array}$$

- 7) Note: The instruction cannot be used for adjustment after subtraction is executed. To subtract decimal (BCD) data, perform complement operation.

### STC (Set Carry)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function: CY ← 1

Set the CY flag to 1.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0, CY ← 1

- 6) Example:

### CLC (Clear Carry)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|

- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function: CY ← 0

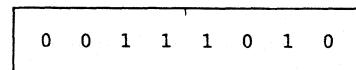
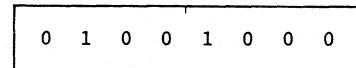
Clear the CY flag.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0, CY ← 0

- 6) Example:

**NEGA (Negate A)**

- 1) Operation code



- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function:  $A \leftarrow A + 1$

Two's complement of the accumulator.

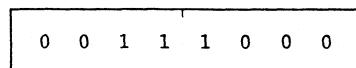
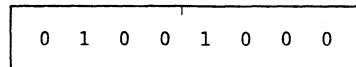
- 5) Affected flags: SK  $\leftarrow 0$ , L1  $\leftarrow 0$ , L0  $\leftarrow 0$

- 6) Example:

**13.6.11 Rotation and shift instructions**

**RLD (Rotate Left digit)**

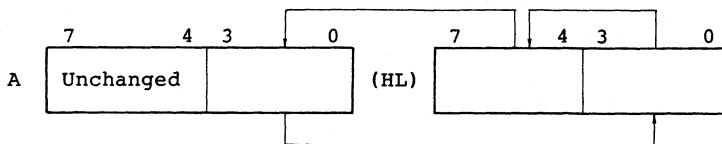
- 1) Operation code



- 2) Number of bytes: 2

- 3) Number of states: 17 (8)

- 4) Function:  $A_{3-0} \leftarrow (HL)_{7-4} \leftarrow (HL)_{3-0}$ ,  $(HL)_{3-0} \leftarrow A_{3-0}$



Rotate the low-order four bits of the accumulator and the high-order four bits and low-order four bits of the memory location addressed by the HL register pair left four bits (digit). Accumulator bits 7-4 are not affected.

5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0

6) Example:

|                  | A |   |   |   | (HL) |   |   |   |
|------------------|---|---|---|---|------|---|---|---|
|                  | 7 | 4 | 3 | 0 | 7    | 4 | 3 | 0 |
| Before execution | 0 | 0 | 0 | 0 | 0    | 1 | 0 | 1 |
|                  |   |   |   |   |      |   |   |   |
| After execution  | 0 | 0 | 0 | 0 | 0    | 1 | 0 | 1 |
|                  |   |   |   |   |      |   |   |   |

#### RRD (Rotate Right Digeit)

1) Operation code      

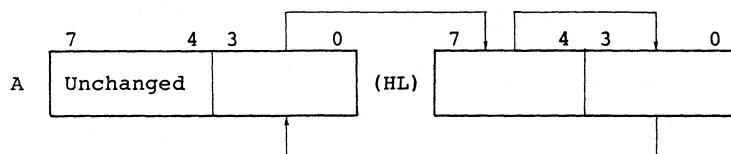
|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

2) Number of bytes: 2

3) Number of states: 17 (8)

4) Function:  $(HL)_{7-4} \leftarrow A_{3-0}, (HL)_{3-0}, A_{3-0} \leftarrow (HL)_{3-0}$



Rotate the low-order four bits of the accumulator and the high-order four bits and low-order four bits if the memory location addressed by the HL register pair left four bits (digit). Accumulator bits 7-4 are not affected.

5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0

6) Example:

|                  | A                                                                                                                                                                                                                        | (HL) |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                                                                                          |   |   |   |   |   |   |   |   |   |   |   |   |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|---|---|---|---|---|---|---|---|---|---|---|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|---|---|---|---|---|---|---|---|---|---|---|
| Before execution | <table border="1" style="width: 100px; border-collapse: collapse;"> <tr><td>7</td><td>4</td><td>3</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr> </table> | 7    | 4 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | <table border="1" style="width: 100px; border-collapse: collapse;"> <tr><td>7</td><td>4</td><td>3</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td></tr> </table> | 7 | 4 | 3 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 7                | 4                                                                                                                                                                                                                        | 3    | 0 |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                                                                                          |   |   |   |   |   |   |   |   |   |   |   |   |
| 0                | 0                                                                                                                                                                                                                        | 0    | 0 | 0 | 0 | 0 | 1 |   |   |   |   |   |   |                                                                                                                                                                                                                          |   |   |   |   |   |   |   |   |   |   |   |   |
| 7                | 4                                                                                                                                                                                                                        | 3    | 0 |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                                                                                          |   |   |   |   |   |   |   |   |   |   |   |   |
| 0                | 1                                                                                                                                                                                                                        | 0    | 1 | 0 | 0 | 1 | 1 |   |   |   |   |   |   |                                                                                                                                                                                                                          |   |   |   |   |   |   |   |   |   |   |   |   |
| After execution  | <table border="1" style="width: 100px; border-collapse: collapse;"> <tr><td>7</td><td>4</td><td>3</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td></td></tr> </table>  | 7    | 4 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |   | <table border="1" style="width: 100px; border-collapse: collapse;"> <tr><td>7</td><td>4</td><td>3</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> </table> | 7 | 4 | 3 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 7                | 4                                                                                                                                                                                                                        | 3    | 0 |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                                                                                          |   |   |   |   |   |   |   |   |   |   |   |   |
| 0                | 0                                                                                                                                                                                                                        | 0    | 0 | 0 | 1 | 1 |   |   |   |   |   |   |   |                                                                                                                                                                                                                          |   |   |   |   |   |   |   |   |   |   |   |   |
| 7                | 4                                                                                                                                                                                                                        | 3    | 0 |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                                                                                          |   |   |   |   |   |   |   |   |   |   |   |   |
| 0                | 0                                                                                                                                                                                                                        | 0    | 1 | 0 | 1 | 0 | 1 |   |   |   |   |   |   |                                                                                                                                                                                                                          |   |   |   |   |   |   |   |   |   |   |   |   |

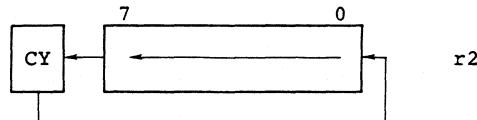
RLL r2 (Rotate Logical Left Register)

|                   |                                                                                                                                                                                                |   |   |   |   |                |                |                |                |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|---|---|---|----------------|----------------|----------------|----------------|
| 1) Operation code | <table border="1" style="width: 100px; border-collapse: collapse;"> <tr><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td></tr> </table>                         | 0 | 1 | 0 | 0 | 1              | 0              | 0              | 0              |
| 0                 | 1                                                                                                                                                                                              | 0 | 0 | 1 | 0 | 0              | 0              |                |                |
|                   | <table border="1" style="width: 100px; border-collapse: collapse;"> <tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>R<sub>1</sub></td><td>R<sub>0</sub></td></tr> </table> | 0 | 0 | 1 | 1 | 0              | 1              | R <sub>1</sub> | R <sub>0</sub> |
| 0                 | 0                                                                                                                                                                                              | 1 | 1 | 0 | 1 | R <sub>1</sub> | R <sub>0</sub> |                |                |

2) Number of bytes: 2

3) Number of states: 8 (8)

4) Function: r2<sub>m+1</sub> ← r2<sub>m</sub>, r2<sub>0</sub> ← CY, CY ← r2<sub>7</sub>



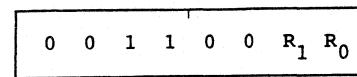
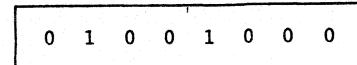
Rotate the contents of register 42 (A, B, or C) indicated by R1R0 (1-3) left one bit through CY flag.

5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0, CY

6) Example:

**RLR r2 (Rotate Logical Right Register)**

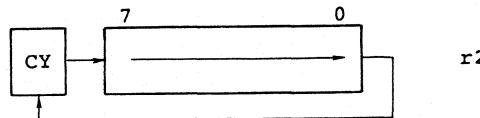
- 1) Operation code



- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function:
- $r2_{m-1} \leftarrow r2_m, r2_7 \leftarrow CY, CY \leftarrow r2_0$



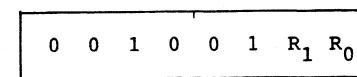
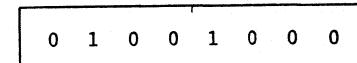
Rotate the contents of register r2 (A, B, or C) indicated by R1R0 (1-3) right one bit through CY flag.

- 5) Affected flags: SK
- $\leftarrow 0$
- , L1
- $\leftarrow 0$
- , L0
- $\leftarrow 0$
- , CY

- 6) Example:

**SLL r2 (Shift Logical Left Register)**

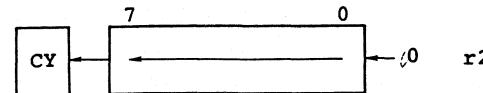
- 1) Operation code



- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

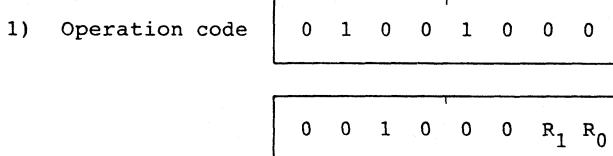
- 4) Function:
- $r2_{m+1} \leftarrow r2_m, r2_0 \leftarrow 0, CY \leftarrow r2_7$



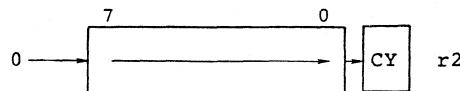
Shift left one bit the contents of register r2 (A, B, or C) indicated by R1R0 (1-3). R27 is shifted to the CY flag and 0 is loaded into r20.

- 5) Affected flags: SK  $\leftarrow$  0, L1  $\leftarrow$  0, L0  $\leftarrow$  0, CY
- 6) Example:

SLR r2 (Shift Logical Left Register)



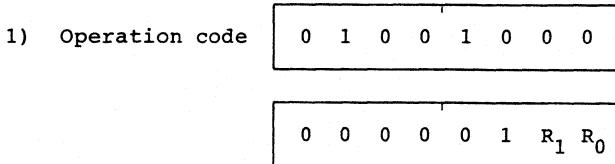
- 2) Number of bytes: 2
- 3) Number of states: 8 (8)
- 4) Function: r<sub>2m-1</sub>  $\leftarrow$  r<sub>2m</sub>, r<sub>27</sub>  $\leftarrow$  0, CY  $\leftarrow$  r<sub>20</sub>



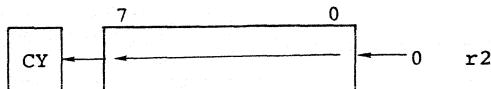
Shift right one bit the contents of register r2 (A, B, or C) indicated by R1R0 (1-3). r20 is shifted to the CY flag and 0 is loaded into r27.

- 5) Affected flags: SK  $\leftarrow$  0, L1  $\leftarrow$  0, L0  $\leftarrow$  0, CY
- 6) Example:

SLLC r2 (Shift Logical Left Register. Skip if Carry)



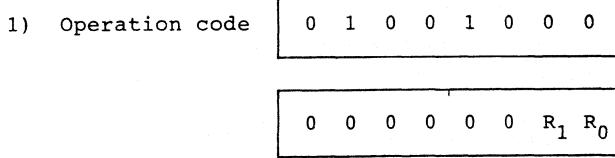
- 2) Number of bytes: 2
- 3) Number of states: 8 (8)
- 4) Function:  $r2_{m+1} \leftarrow r2_m$ ,  $r2_0 \leftarrow 0$ , CY  $\leftarrow r2_7$ ; skip if carry



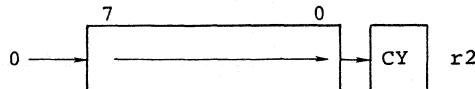
Shift left one bit the contents of register r2 (A, B, or C) indicated by R1R0 (1-3). r27 is shifted to the CY flag and 0 is loaded into R20. Skip if there is a carry as the result of shift.

- 5) Affected flags: SK, L1  $\leftarrow 0$ , L0  $\leftarrow 0$ , CY
- 6) Example:

SLRC r2 (Shift Logical Right Register. Skip if Carry)



- 2) Number of bytes: 2
- 3) Number of states: 8 (8)
- 4) Function:  $r2_{m-1} \leftarrow r2_m$ ,  $r2_7 \leftarrow 0$ , CY  $\leftarrow r2_0$



Shift right one bit the contents of register r2 (A, B, or C) indicated by R1R0 (1-3). r27 is shifted to the CY flag and 0 is loaded into r27. Skip if there is a carry as the result of shift.

- 5) Affected flags: SK, L1 ← 0, L0 ← 0, CY
- 6) Example:

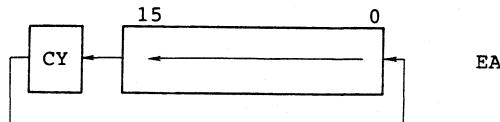
DRLL EA (Rotate Logical Left EA)

1) Operation code      

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|

- 2) Number of bytes: 2
- 3) Number of states: 8 (8)
- 4) Function:  $EA_{n+1} \leftarrow EA_n$ ,  $EA_0 \leftarrow CY$ ,  $CY \leftarrow EA_{15}$



Rotate the expansion accumulator contents left one bit through CY flag.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0, CY
- 6) Example:

DRLR EA (Rotate Logical Right EA)

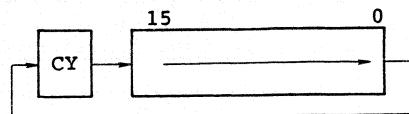
1) Operation code      

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

- 2) Number of bytes: 2
- 3) Number of states: 8 (8)

- 4) Function:  $EA_{n-1} \leftarrow EA_n$ ,  $EA_{15} \leftarrow CY$ ,  $CY \leftarrow EA_0$



Rotate the expansion accumulator contents right one bit through CY flag.

- 5) Affected flags: SK  $\leftarrow 0$ , L1  $\leftarrow 0$ , L0  $\leftarrow 0$ , CY  
6) Example:

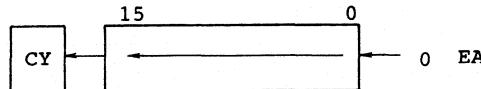
DSLL EA (Shift Logical Left EA)

- 1) Operation code      

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|
- 2) Number of bytes: 2  
3) Number of states: 8 (8)  
4) Function:  $EA_{n-1} \leftarrow EA_n$ ,  $EA_0 \leftarrow CY$ ,  $CY \leftarrow EA_{15}$

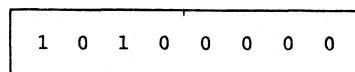
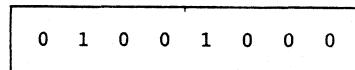


Shift left one bit the expansion accumulator contents. EA15 is shifted to the CY flag and 0 is loaded into EA0.

- 5) Affected flags: SK  $\leftarrow 0$ , L1  $\leftarrow 0$ , L0  $\leftarrow 0$ , CY  
6) Example:

DSLR EA (Shift Logical Right EA)

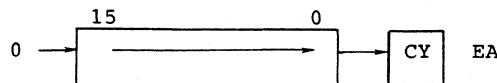
- 1) Operation code



- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function:  $EA_{n-1} \leftarrow EA_n$ ,  $EA_{15} \leftarrow 0$ ,  $CY \leftarrow EA_0$



Shift right one bit the expansion accumulator contents.  
EA0 is shifted to the CY flag and 0 is loaded into EA15.

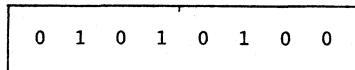
- 5) Affected flags: SK  $\leftarrow 0$ , L1  $\leftarrow 0$ , L0  $\leftarrow 0$ , CY

- 6) Example:

#### 13.6.12 Jump instructions

JMP word (Jump direct)

- 1) Operation code



Low Address

High Address

- 2) Number of bytes: 3

- 3) Number of states: 10 (10)

- 4) Function: PC ← word

Load the immediate data of the second byte into the low-order eight bits of the program counter (PC7-PC0) and the immediate data of the third byte into the high-order eight bits (PC15-PC8). Jump to the address indicated by the immediate data.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0

- 6) Example: MVI C,7FH ;C=127

LXI H,4000H ;HL=4000H

LXI D,5000H ;DE=5000H

BLOCK ;(DE) + ← (HL) +, C ← C - 1

JMP OWARI ;PC ← OWARI

;JUMP TO OWARI

At the termination of 128-byte block transfer, jump to the address indicated by label OWARI.

JB (Jump BC indirect)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

- 2) Number of bytes: 1

- 3) Number of States: 4 (4)

- 4) Function: PC15-8 ← B, PC7-0 ← C

Load the B register contents into the high-order eight bits of the program counter (PC15-PC8) and the C register contents into the low-order eight bits (PC7-PC0). Jump to the address indicated by the BC register pair.

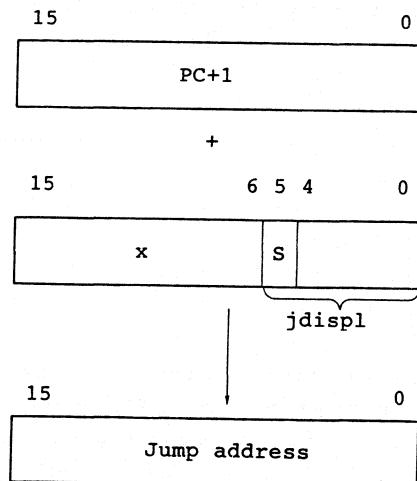
For example, the JB instruction can be used for jump after address information is loaded into BC by executing the TABLE instruction.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0
- 6) Example:

JR word (Jump Relative)

- 1) Operation code      1 1      jdispl
- 2) Number of bytes: 1
- 3) Number of states: 10 (4)
- 4) Function: PC ← PC+1+jdispl

Jump to the address provided by adding 6-bit displacement jdispl to the top address of the next instruction. jdispl is handled as signed two's complement data (-32 to +31) and bit 5 becomes a sign bit.



When S=0: x=all 0

When S=1: x=all 1

The jump destination address or label should be directly entered in the JR instruction operand by considering the jump range. For example, if the JR instruction is executed at address 1000, a jump can be made within the range of addresses 969 to 1032.

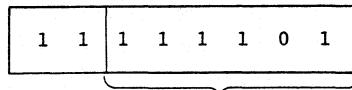
5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0

6) Example: CLWR:LXI D,2000H;DE=2000H

```
 MVI C,7 ;C-7,LOOP COUNTER
 XRA A,A ;CLEAR A
98 LOOP:STAX D+ ;(DE) ← 0, DE ← DE+1
99 DCR C ;C ← C-1,SKIP IF BORROW
100 JR LOOP ;JUMP TO LOOP
 RET ;
```

Make a loop repeatedly by executing the JR instruction until the contents of eight addresses from memory addresses 2000H to 2007H are cleared.

In this example, the displacement becomes -3, thus the actual operation code is as follows:



Two's complement of -3

JRE word (Jump Relative Extended)

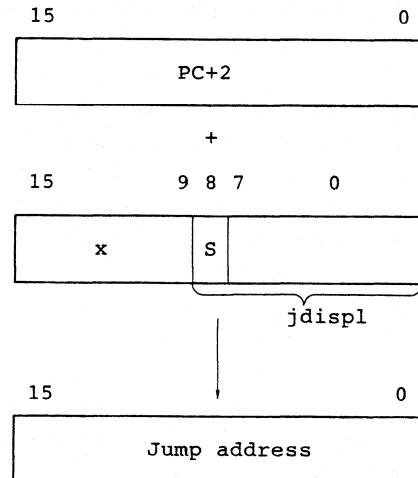
1) Operation code    

|               |                |
|---------------|----------------|
| 0 1 0 0 1 1 1 | j <sub>H</sub> |
|---------------|----------------|

|        |
|--------|
| jdispL |
|--------|

2) Number of bytes: 2

- 3) Number of states: 10 (7)
- 4) Function:  $PC \leftarrow PC + 2 + jdisp$



When S=0: x=all 0  
When S=1: x=all 1

Jump to the address provided by adding 9-bit displacement jdisp to the top address of the next instruction. jdisp is handled as signed two's complement data (-256 to +255) and bit 8 (bit 0 of the first byte) becomes a sign bit.

The jump destination address or label should be directly entered in the JRE instruction operand by considering the jump range. For example, if the JRE instruction is executed at address 1000, a jump can be made within the range of addresses 746 to 1257.

- 5) Affected flags: SK  $\leftarrow 0$ , L1  $\leftarrow 0$ , L0  $\leftarrow 0$
- 6) Example:

## JEA (Jump EA indirect)

- 1) Operation code

0 1 0 0 1 0 0 0

- 2) Number of bytes: 2  
3) Number of states: 8 (8)  
4) Function: PC + EA

Load the high-order eight bits of the expansion accumulator (EAH) into the high-order eight bits of the program counter (P15-P8) and the low-order eight bits of the expansion accumulator (EAL) into the low-order eight bits (PC7-PC0). Jump to the address indicated by the expansion accumulator.

- 5) Affected flags: SK + 0, L1 + 0, L0 + 0
- 
- 6) Example:

## 13.6.13 Call instructions

## CALL word (Call subroutine direct)

- 1) Operation code

0 1 0 0 0 0 0 0

Low Address

High Address

- 2) Number of bytes: 3
- 
- 3) Number of states: 16 (10)

- 4) Function:  $(SP-1) \leftarrow PC+3_{15-8}, (SP-2) \leftarrow PC+3_{7-0}$   
 $SP \leftarrow SP-2, PC \leftarrow \text{word}$

Store the high-order eight bits of the top address of the instruction following the CALL in the stack memory addressed by SP-1 and the low-order eight bits in the stack memory address by SP-2. Then, load the immediate data of the second byte of the CALL instruction into the low-order eight bits of the program counter (PC7-PC0) and the immediate data of the third byte into the high-order eight bits (PC15-PC8). Jump to the address indicated by the immediate data.

- 5) Affected flags: SK  $\leftarrow 0$ , L1  $\leftarrow 0$ , L0  $\leftarrow 0$   
6) Example:

**CALB (Call subroutine BC indirect)**

1) Operation code      

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

- 2) Number of bytes: 2  
3) Number of states: 17 (8)  
4) Function:  $(SP-1) \leftarrow PC+2_{15-8}, (SP-2) \leftarrow PC+2_{7-0}$   
 $SP \leftarrow SP-2, PC_{15-8} \leftarrow B, PC_{7-0} \leftarrow C$

Store the high-order eight bits of the top address of the instruction following the CALB in the stack memory addressed by SP-1 and the low-order eight bits in the stack memory address by SP-2. Then, load the B register contents into the high-order eight bits of the program counter (PC15-PC8) and the C register contents into the low-order eight bits (PC-PC0). Jump to the address indicated by the BC register.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0
- 6) Example:

CALF word (Call subroutine in fixed area)

- 1) Operation code

|   |   |   |   |   |        |
|---|---|---|---|---|--------|
| 0 | 1 | 1 | 1 | 1 | $fa_H$ |
|---|---|---|---|---|--------|

$fa_L$

- 2) Number of bytes: 2
- 3) Number of states: 13 (7)
- 4) Function:  $(SP-1) \leftarrow PC+3_{15-8}$ ,  $(SP-2) \leftarrow PC+2_{7-0}$   
 $SP \leftarrow SP-2$ ,  $PC_{15-11} \leftarrow 00001$ ,  $PC_{10-0} \leftarrow fa$

Store the high-order eight bits of the top address of the instruction following the CALF in the stack memory addressed by SP-1 and the low-order eight bits in the stack memory address by SP-2. Then, load 00001 into the high-order five bits of the program counter (PC15-PC11) and 11-bit immediate data fa into the low-order 11 bits (PC10-PC0). Jump to the address indicated by the immediate data.

The call destination address (label or numeric value ranging from 800H to FFFF (2K-byte range)) should be entered in the CALF instruction operand.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0
- 6) Example:

CALT word (Call Table Address)

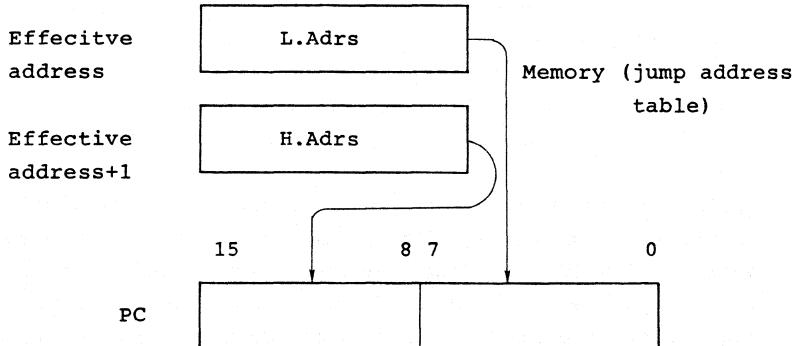
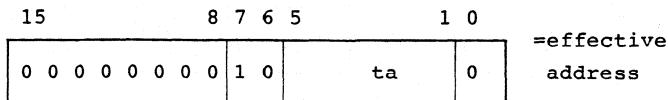
- 1) Operation code

|   |   |   |      |
|---|---|---|------|
| 1 | 0 | 0 | $ta$ |
|---|---|---|------|

- 2) Number of bytes: 1

3) Number of states: 16 (4)

4) Function:  $(SP-1) \leftarrow PC+3_{15-8}$ ,  $(SP-2) \leftarrow PC+1_{7-0}$   
 $SP \leftarrow SP-2$ ,  $PC_{7-0} \leftarrow (128+2ta)$ ,  $PC15-8 \leftarrow (129+2ta)$



Store the high-order eight bits of the top address of the instruction following the CALT in the stack memory addressed by SP-1 and the low-order eight bits in the stack memory address by SP-2. Then, load the contents of the memory location addressed by the effective address made up of 5-bit immediate data ta into the low-order eight bits of the program counter (PC7-PC0) and the contents of the memory location addressed by effective address+1 into the high-order eight bits (PC15-PC8). Jump to the address indicated by the memory location contents.

The jump address table must be located in addresses 128-191.

The table address (label or a numeric value of up to 16 bits) should be entered in the CALT instruction operand.

- 5) Affected flags: SK 0, L1 0, L0 0
- 6) Example:

#### SOFTI (Software Interrupt)

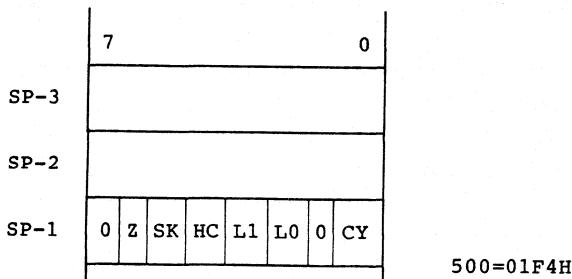
- 1) Operation code 

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
- 2) Number of bytes: 1
- 3) Number of states: 16
- 4) Function:  $(SP-1) \leftarrow PSW, (SP-2) \leftarrow PC+3_{15-8}, (SP-3) \leftarrow PC+1_{7-0}, SP \leftarrow SP-3$   
 $PC \leftarrow 0060H$

Store the PSW (X, SK, HC, L1, L0, and CY) contents in the stack memory addressed by SP-1, the high-order eight bits of the top address of the instruction following the SOFTI in the stack memory addressed by SP-2, and the low-order eight bits in the stack memory addressed by SP-3. Then load 0060H into the program counter. Jump to address 0060H.

- 5) Affected flags: SK  $\leftarrow$  0, L1  $\leftarrow$  0, L0  $\leftarrow$  0
- 6) Example: Execute SOFTI instruction at address 500  
500 SOFTI  
501  
502

## Memory



- 7) Note: Note ethat the difference between the uCOM-87 and uCOM-87AD SOFTI instructions is that the address of the uCOM-87 SOFTI instruction is saved in the stack memory; whereas, the top address of the instruction following the uCOM-87AD SOFTI instruction is saved in the stack memory. Even if the skip condition is true when the instruction just preceding the SOPFI instruction (arithmetic or logical operation, increment or decrement, skip or RETS) is executed, the SOFTI instruction is not skipped and executed. (See 9.4)

## 13.6.14 Return instructions

RET (Return from subroutine)

- |                   |                 |
|-------------------|-----------------|
| 1) Operation code | 1 0 1 1 1 0 0 0 |
|-------------------|-----------------|
- 2) Number of bytes: 1  
 3) Number of states: 10 (4)  
 4) Function:  $PC_{7-0} + (SP)$ ,  $PC_{15-8} + (SP+1)$ ,  $SP \leftarrow SP+2$

Restore the contents of the stack memory addressed by SP in the low-order eight bits of the program counter (PC7-PC0) and the contents of the stack memory addressed by SP-1 in the high-order eight bits (PC15-PC8).

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0
- 6) Example:

RETS (Return from subroutine and Skip)

- 1) Operation code 

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|
- 2) Number of bytes: 1
- 3) Number of states: 10 (4)
- 4) Function: PC<sub>7-0</sub> ← (SP), PC<sub>15-8</sub> ← (SP+1), SP ← SP+2  
PC ← PC+n'

n': Number of bytes of skipped instruction

Restore the contents of the stack memory addressed by SP in the low-order eight bits of the program counter (PC7-PC0) and the contents of the stack memory addressed by SP+1 in the high-order eight bits (PC15-PC8). Then, skip the instruction following the CALL instruction unconditionally.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0
- 6) Example: EXAM EQU 0600H  

|      |      |       |        |       |
|------|------|-------|--------|-------|
| 0500 | CALL | EXAM  | ;STACK | 0503H |
| 0503 | JMP  | 0700H |        |       |
| 0506 | JMP  | 0800H |        |       |
| 0509 |      |       |        |       |

  
0600 EXAM:PUSH V  
PUSH B  
  
POP B  
POP V  
RETS ;PC ← (STACK), PC ← PC+3

In return from the subroutine EXAM, skip JMP 0700H and execute JMP 0800H.

**RETI (Return from Interrupt)**

- 1) Operation code 

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|

- 2) Number of bytes: 1

- 3) Number of states: 13 (4)

- 4) Function:  $PC_{7-0} \leftarrow (SP)$ ,  $PC_{15-8} \leftarrow (SP+1)$ ,  $SPW \leftarrow (SP+2)$   
 $SP \leftarrow SP+3$

Restore the contents of the stack memory addressed by SP in the low-order eight bits of the program counter (PC7-PC0), the contents of the stack memory addressed by SP+1 in the high-order eight bits (PC15-PC8), and the contents of the stack memory addressed by SP+2 in PSW.

The RETI instruction is used to return from the interrupt service routine of an external interrupt (NMI, INT1, or INT2), internal interrupt such as timer or serial transfer, or SPFTI instruction interrupt.

- 5) Affected flags: SK, L1, L0

- 6) Example:

**13.6.15 Skip instructions**

**BIT bit, wa (Bit Test Working Register)**

- 1) Operation code 

|   |   |   |   |   |       |       |       |
|---|---|---|---|---|-------|-------|-------|
| 0 | 1 | 0 | 1 | 1 | $B_2$ | $B_1$ | $B_0$ |
|---|---|---|---|---|-------|-------|-------|

Offset

- 2) Number of bytes: 2

- 3) Number of states: 10 (7)

- 4) Function: Skip if bit on.

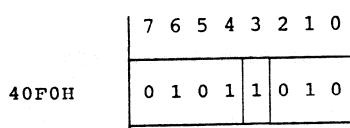
Skip if the bit of the working register addressed by the V register (high-order eight bits of the address) and the immediate data of the second byte (low-order eight bits), indicated by B2B1B0 (0-7) is set to 1.

- 5) Affected flags: SK, L1 ← 0, L0 ← 0

- 6) Example: When address 10F0H contains 5AH

```
MVI V, 40H
BIT 3, 0FOH
JR $+2
RET
```

Working register



In this example, since the specified bit of the specified address is set to 1, skip the JR instruction and execute the RET instruction.

SK f (Skip if flag)

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |                |                |                |
|---|---|---|---|---|----------------|----------------|----------------|
| 0 | 0 | 0 | 0 | 1 | F <sub>2</sub> | F <sub>1</sub> | F <sub>0</sub> |
|---|---|---|---|---|----------------|----------------|----------------|

- 2) Number of bytes: 2

- 3) Number of states: 8 (8)

- 4) Function: Skip if f=1

Skip if the status of flag f (CY, HC, or Z) indicated by F2F1F0 (2-4) equals 1.

- 5) Affected flags: SK, L1 ← 0, L0 ← 0
- 6) Example:

**SKN f (Skip if No Flag)**

- 1) Operation code 

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|
- 2) Number of bytes: 2
- 3) Number of states: 8 (8)
- 4) Function: Skip if f=0

Skip if the status of flag f (CY, HC, or Z) indicated by F2F1F0 (2-4) equals to 0.

- 5) Affected flags: SK, L1 ← 0, L0 ← 0
- 6) Example:

**SKIT irf (Skip if Interrupt)**

- 1) Operation code 

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|
- 2) Number of bytes: 2
- 3) Number of states: 8 (8)
- 4) Function: Skip if irf=1, then reset irf.

Skip if the status of the interrupt request flag or test flag (NMI, FT0, FT1, F1, F2, FE0, FE1, FEIN, FAD, FSR, FST, ER, OV, AN4, AN5, AN6, AN7, or SB) indicated by I4I3I2I0 (0-C, 10-14) equals 1. Then, the interrupt request flag is reset. The NMI flag is not affected.

- 5) Affected flags: SK, L1 ← 0, L0 ← 0
- 6) Example:

SKINT irf (Skip if No Interrupt)

- 1) Operation code 

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|
  - 2) Number of bytes: 2
  - 3) Number of states: 8 (8)
  - 4) Function: Skip if irf=0
- |   |   |   |                |                |                |                |                |
|---|---|---|----------------|----------------|----------------|----------------|----------------|
| 0 | 1 | 1 | I <sub>4</sub> | I <sub>3</sub> | I <sub>2</sub> | I <sub>1</sub> | I <sub>0</sub> |
|---|---|---|----------------|----------------|----------------|----------------|----------------|

Skip if the status of the interrupt request flag or test flag (NMI, FT0, FT1, F1, F2, FE0, FE1, FEIN, FAD, FSR, FST, ER, OV, AN4, AN5, AN6, AN7, or SB) indicated by I4I3I2I0 (0-C, 10-14) equals 0.

If the interrupt request flag is set to 1, the interrupt request flag is reset. NMI is not affected.

- 5) Affected flags: SK, L1 ← 0, L0 ← 0
- 6) Example:

### 13.6.16 CPU control instructions

#### NOP (No Operation)

1) Operation code      

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

2) Number of bytes: 1

3) Number of states: 4 (4)

4) Function:

Performs no operation and consumes four states.

5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0

6) Example:

#### EI (Enable Interrupt)

1) Operation code      

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|

2) Number of bytes: 1

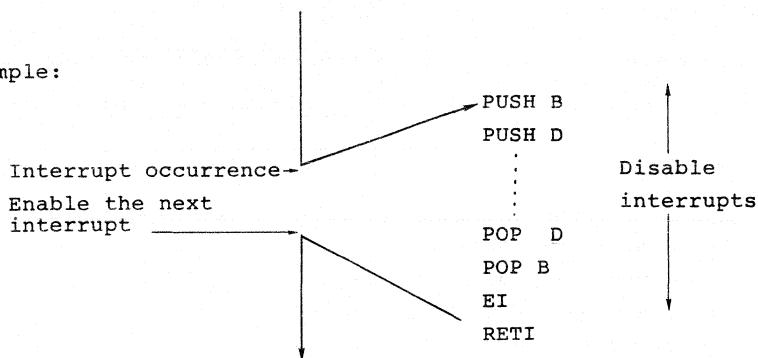
3) Number of states: 4 (4)

4) Function:

Enable interrupts. However, an actual interrupt is enabled after the instruction following the EI instruction, such as a return instruction, is executed. Extra stack is not used for subsequent interrupts that will occur. Nonmaskable and SOFTI instruction insterrupts are always enabled regardless of the EI instruction.

5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0

## 6) Example:



DI (Disable Interrupt)

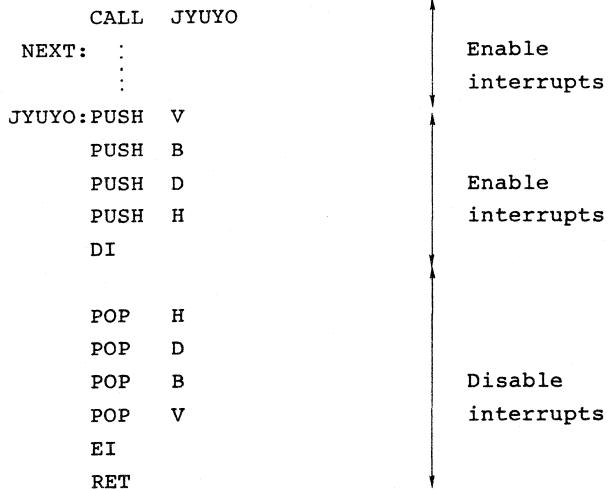
- 1) Operation code      

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|

- 2) Number of bytes: 1  
 3) Number of states: 4 (4)  
 4) Function:

Disable interrupts other than a nomaskable or SOFTI instruction interrupt. Interrupts are disabled during DI instruction execution.

- 5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0  
 6) Example:



**HLT (Halt)**

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

- 2) Number of bytes: 2  
3) Number of states: 12 (8)  
4) Function: Set the HALT mode.  
5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0  
6) Example:

**STOP (Stop)**

- 1) Operation code

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

- 2) Number of bytes: 2  
3) Number of states: 12 (8)  
4) Function: Set the software STOP mode.  
5) Affected flags: SK ← 0, L1 ← 0, L0 ← 0  
6) Example:

### 13.7 String Effect Instructions

If the instructions of group A or B shown below are located in two or more contiguous addresses in a program, the instruction at the start of the string effect instruction group is executed and after this, no operation is performed as long as the number of states required for execution of the instructions:

|                              |           |
|------------------------------|-----------|
| Group A: MVI A, byte         | (L1 flag) |
| Group B: L, byte;LXI H, word | (L0 flag) |

When group A instruction is executed, the L1 flag is set. When group B instruction is executed, the L0 flag is set. A check is made to see if instructions of the same group are located in contiguous addresses.

Although interrupts are not disabled during string effect instruction execution, the L1 and L0 flags are automatically saved when an interrupt is made, thus whether or not the next instruction is to be given the string effect can be decided when a return is made from the interrupt service routine.

When the string effect instruction MVI is used, a program which clears registers of four bytes located in a specific memory area (in this case, the high-order byte of the address is 10H) can be written as follows:

|      | 8     | 9 | A     | B | C | D | E | F |
|------|-------|---|-------|---|---|---|---|---|
|      | 0     | 1 | 2     | 3 | 4 | 5 | 6 | 7 |
| 1000 | X-REG |   | Y-REG |   |   |   |   |   |
| 1008 | Z-REG |   | W-REG |   |   |   |   |   |

```
CLX : MVI L,00H ; CLEAR X-REG
CLY : MVI L,04H ; CLEAR Y-REG
CLX : MVI L,08H ; CLEAR Z-REG
CLW : MVI L,0CH ; CLEAR W-REG
 MVI H,40H
 MVI C,3 ; SET COUNTER
 XRA A,A ; CLEAR A
LOOP: STAX H+ ; (HL) ← 0, HL ← HL+1
 DCR C ; SKIP IF BORROW
 JR LOOP
 RET
```

For example, when CLY is called, 04H is loaded into the L register; the subsequent two string effect instructions MVI L, 08H and MVI L, 0CH are replaced with idle cycle of 14 states (NOP); the high-order byte of the address is determined by the MVI H, 40H instruction following the MVI L, 0CH instruction; and the top address of Y-REG, 4004H is loaded into the HL register pair.

## APPENDIX A INTRODUCTION OF PIGGY-BACK PRODUCT

## uPD78CG14

8-bit microcomputer (with A/D converter) on which EPROM is mounted

The uPD78CG14 is an 8-bit microcomputer to which piggy-back program memory C standard (EPROM27C256/27C256A) can be connected.

The uPD78CG14 is pin-compatible with the QUIP type of single chip 8-bit microcomputer uPD78C11/78C11A/78C12A/78C14/78C14A containing on-chip mask ROM and they are the same in function.

uPD78CG14 programs can be changed by rewriting EPROM; the uPD78CG14 is appropriate for evaluation or small production of uPD78C11/78C11A/78C12A/78C14/78C14A.

## Features

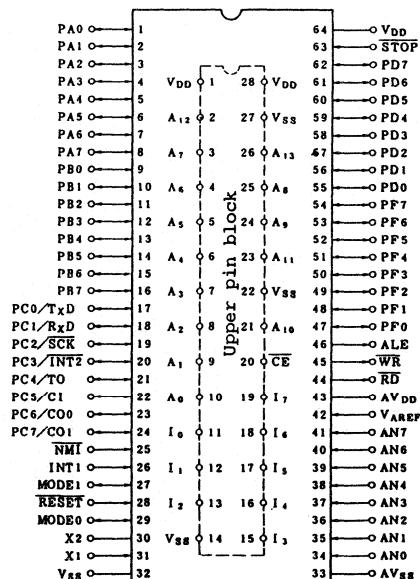
- o uPD78C11/78C11A/78C12A/78C14/78C14A (QUIP type) compatible
- o The capacity that can be accessed as piggy-back memory can be selected among 16K, 8K, and 4K bytes by software.
- o Program memory addressing space: 65280 x 8 bits
- o Internal RAM capacity: 256 x 8 bits
- o Standby function: HALT mode, hardware STOP mode, or software STOP mode
- o CMOS
- o Single 5 V ± 10% power supply

## Ordering Information

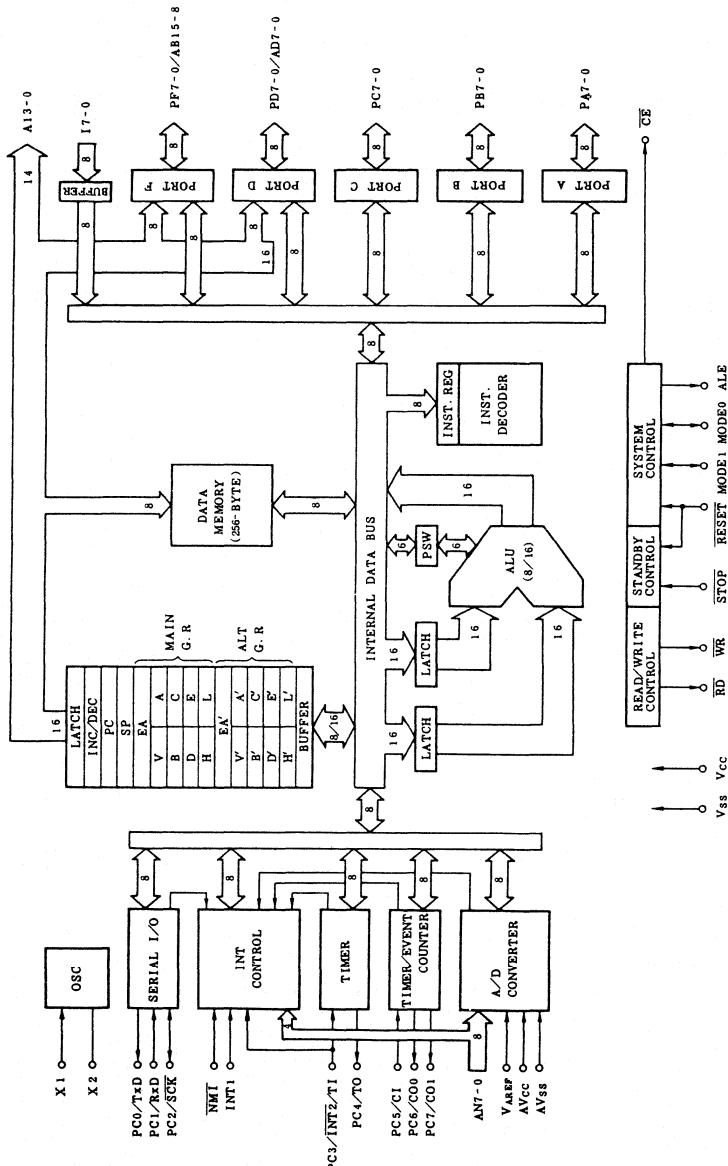
| Ordering code | Package                                        |
|---------------|------------------------------------------------|
| uPD78CG14E    | 64-pin ceramic piggy-back quad-in-line package |

## Pin configuration

(Top View)



uPD78CG14 Block Diagram



## 1. Pin Function

1.1 Lower pins (uPD78C11/78C11A/78C12A/78C14/78C14A QUIP type compatible)

| Pin name           | I/O     | Function                                                                                                                                                                                      |
|--------------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PA7-0<br>(Port A)  | I/O     | 8-bit input/output port. The output mode can be selected bit-wise.                                                                                                                            |
| PB7-0<br>(Port B)  | I/O     | 8-bit input/output port. The input or output mode can be selected bit-wise.                                                                                                                   |
| PC0/TxD            | I/O/O   | Port C<br>8-bit input/output port.<br>The input or output mode can be selected bit-wise.                                                                                                      |
| PC1/RxD            | I/O/I   | Transmit Data<br>Serial data output pin.                                                                                                                                                      |
| PC2/SCK            | I/O/O   | Receive Data<br>Serial data input pin.                                                                                                                                                        |
| PC3/<br>INT2/TI    | I/O/I/I | Serial Clock<br>Serial clock input/output pin. When internal clock is used, the pin becomes an output pin. When external clock is used, the pin becomes an input pin.                         |
| PC4/TO             | I/O/O   | Interrupt Request/<br>Timer Input<br>Edge trigger (falling edge) maskable interrupt input pin or timer external clock input pin. It can also be used as an AC input zero cross detection pin. |
| PC5/CI             | I/O/I   | Timer output<br>Square wave with one period of internal clock or timer count time as a half period is output.                                                                                 |
| PC6/CO0<br>PC7/CO1 | I/O/O   | Counter Input<br>External pulse input pin to the timer/event counter.                                                                                                                         |
|                    |         | Counter Output 0,1<br>Programmable square wave by the timer/event counter is output.                                                                                                          |

| Pin name                               | I/O     | Function                                                                                                                                                                                                                                     |
|----------------------------------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PD7-0/<br>AD7-0                        | I/O/I/O | Port D<br>8-bit input/output port.<br>The input or output mode<br>can be selected bit-wise.                                                                                                                                                  |
| PF7-0/                                 | I/O/O   | Port F<br>8-bit input/output port.<br>The input or output mode<br>can be selected bit-wise.                                                                                                                                                  |
| WR<br>(Write<br>Strobe)                | O       | WR is a strobe signal output for external memory write<br>operation. WR goes high except in external memory data<br>write machine cycle. When the RESET signal is low or<br>during the hardware STOP mode, output becomes high<br>impedance. |
| RD<br>(Read<br>Strobe)                 | O       | RD is a strobe signal output for external memory read<br>operation. RD goes high except in external memory read<br>machine cycle. When the RESET signal is low or during<br>the hardware STOP mode, output becomes high impedance.           |
| ALE<br>(Address<br>Latch<br>Enable)    | O       | ALE is strobe signal used to externally latch low-order<br>address information output to the PD7-PD0 pins to access<br>external memory. When the RESET signal is low or during<br>the hardware STOP mode, output becomes high impedance.     |
| MODE0<br>MODE1<br>(Mode)               | I/O     | The MODE0 pin is set to 0 (low) and the MODE1 pin is set<br>to 1 (high).<br><small>(Note)</small><br>When the MODE0 and MODE1 pins are set to 1<br><small>(Note)</small> , control signal is output synchronization with ALE.                |
| NMI<br>(Non-<br>Maskable<br>Interrupt) | I       | Edge trigger (falling edge) nonmaskable interrupt input<br>pin.                                                                                                                                                                              |
| INT 1<br>(Interrupt<br>Request)        | I       | Edge trigger (rising edge) maskable interrupt input pin.<br>It can also be used as an AC input zero cross<br>detection pin.                                                                                                                  |
| AN7-0<br>(Analog<br>Input)             | I       | Eight analog input pins to the A/D converter. AN7-AN4<br>can be used as edge detection (Falling edge) input.                                                                                                                                 |

Note: Pull high with a pull up resistor ( $4 \leq R \leq 0.4$  CYC [ kΩ ].  
 $t_{CYC}$  is ns units.)

| Pin name                             | I/O | Function                                                                                                    |
|--------------------------------------|-----|-------------------------------------------------------------------------------------------------------------|
| $V_{AREF}$<br>(Reference<br>Voltage) | I   | This pin is used as both A/D converter reference voltage input pin and A/D converter operation control pin. |
| $AV_{DD}$<br>(Analog<br>$V_{DD}$ )   |     | A/D converter power supply pin.                                                                             |
| $AV_{SS}$<br>(Analog<br>$V_{SS}$ )   |     | A/D converter GND pin.                                                                                      |
| X1,X2<br>(Crystal)                   |     | System clock oscillation crystal connection pins. When external clock is supplied, it is input to X1.       |
| RESET<br>(Reset)                     | I   | System reset input which is active low.                                                                     |
| STOP<br>(Stop)                       | I   | Hardware STOP mode control signal input pin. When a low pulse is input to STOP, oscillation stops.          |
| $V_{DD}$                             |     | Positive power supply pin.                                                                                  |
| $V_{SS}$                             |     | GND pin.                                                                                                    |

## 1.2 Upper pins (27C256/27C256A compatible)

| Pin name       | I/O | Function                                                                                                                                                                             | When reset |
|----------------|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|
| $A_0-A_{13}$   | O   | 14 bits of the program counter (PC0-PC13) used as 27C256/27C256A address signal (AO-A13) are output.                                                                                 | Undefined  |
| $I_0-I_{13}$   | I   | Data read from 27C246/27C256A (00-07) is input.                                                                                                                                      |            |
| CE(20)         | O   | Chip enable signal is supplied to the 27C256/27C256A CE pin. During the standby mode (HALT or hardware or software STOP), a high pulse is output; else a low pulse is always output. |            |
| $V_{DD}^{(1)}$ |     | $V_{DD}$ is the same potential as the lower pin $V_{DD}$ . $V_{CC}$ power ( $V_{PP}$ ) is supplied to 27C256/27C256A.                                                                |            |

| Pin name      | I/O | Function                                                                                                                                   | When reset |
|---------------|-----|--------------------------------------------------------------------------------------------------------------------------------------------|------------|
| $V_{DD}$ (28) |     | $V_{DD}$ is the same potential as the lower pin $V_{DD}$ . $V_{CC}$ power ( $V_{CC}$ ) is supplied to 27C256/27C256A.                      |            |
| $V_{SS}$ (14) |     | $V_{SS}$ is the same potential as the lower pin $V_{SS}$ . It is connected to the 27C256/27C256A. GND pin.                                 |            |
| $V_{SS}$ (22) |     | $V_{SS}$ is the same potential as the lower pin $V_{SS}$ . The $\overline{OE}$ signal, which is always low, is supplied to 27C256/27C256A. |            |
| $V_{SS}$ (27) |     | $V_{SS}$ is the same potential as the lower pin $V_{SS}$ . The A14 signal, which is always low, is supplied to 27C256/27C256A.             |            |

## 2. Memory Configuration

The memory of uPD78CG14 can performe the same function and configuration as the memory of uPD78C11/78C11A/78C12A/78C14/78C14A.

EPROM address area can be selected by using memory mapping register.

Three type products are common in vector address, call table area, and data memory area.

Figs. A-1 to A-3 show memory map.

Fig. A-1 Memory map (uPD78C14/78C14A mode)

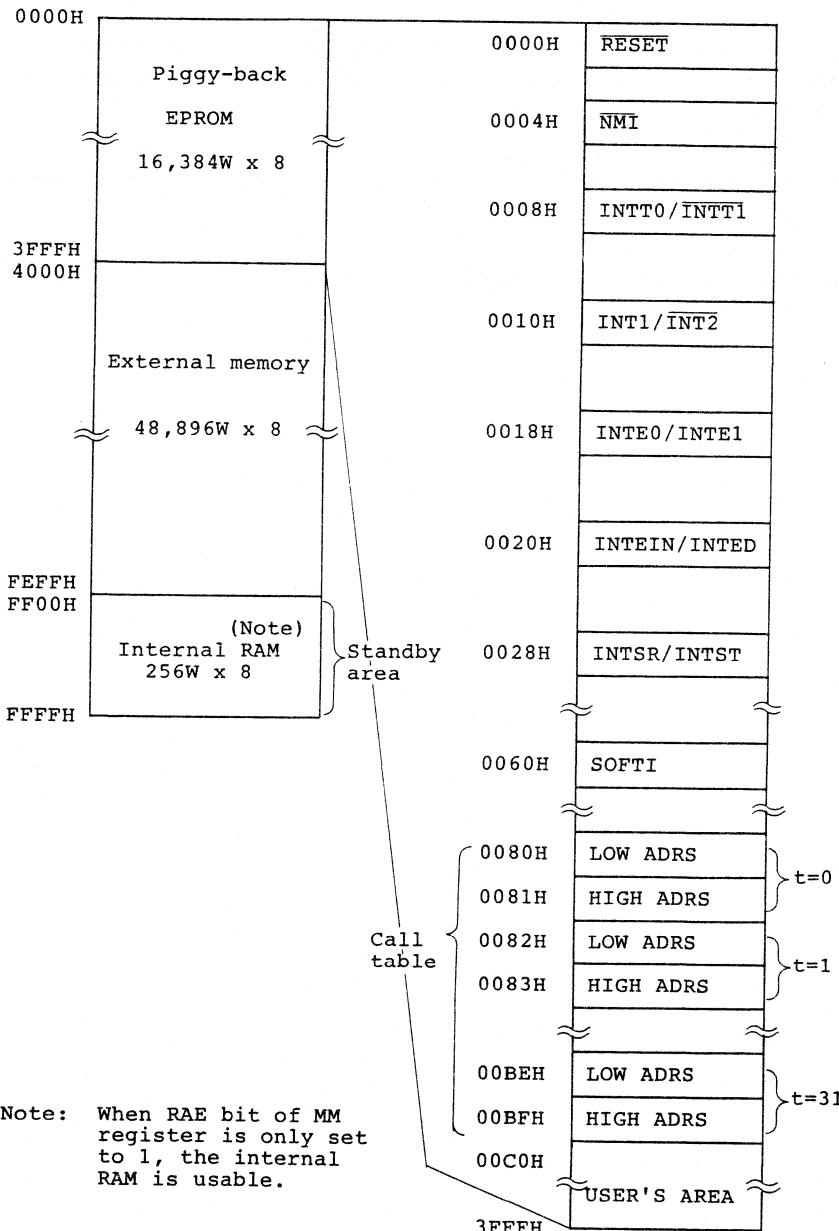


Fig. A-2 Memory map (uPD78C12A mode)

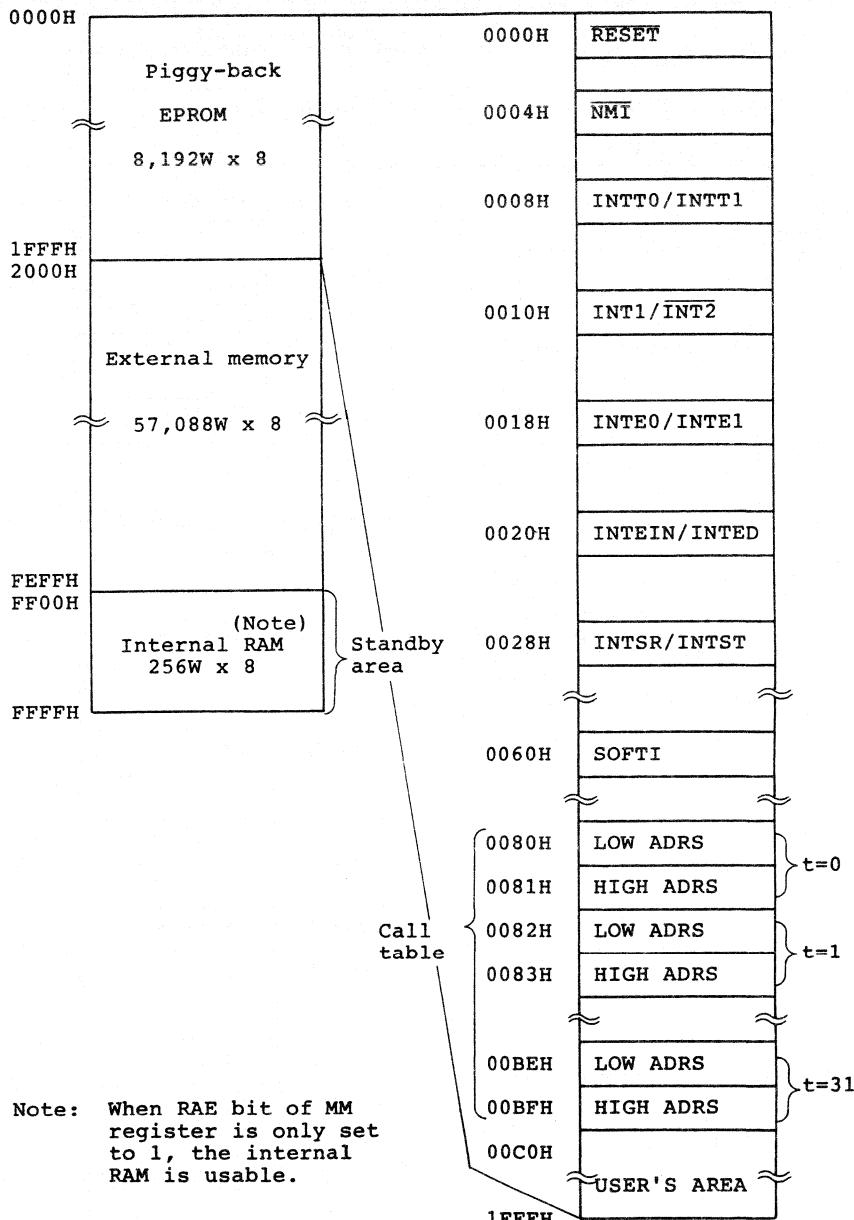
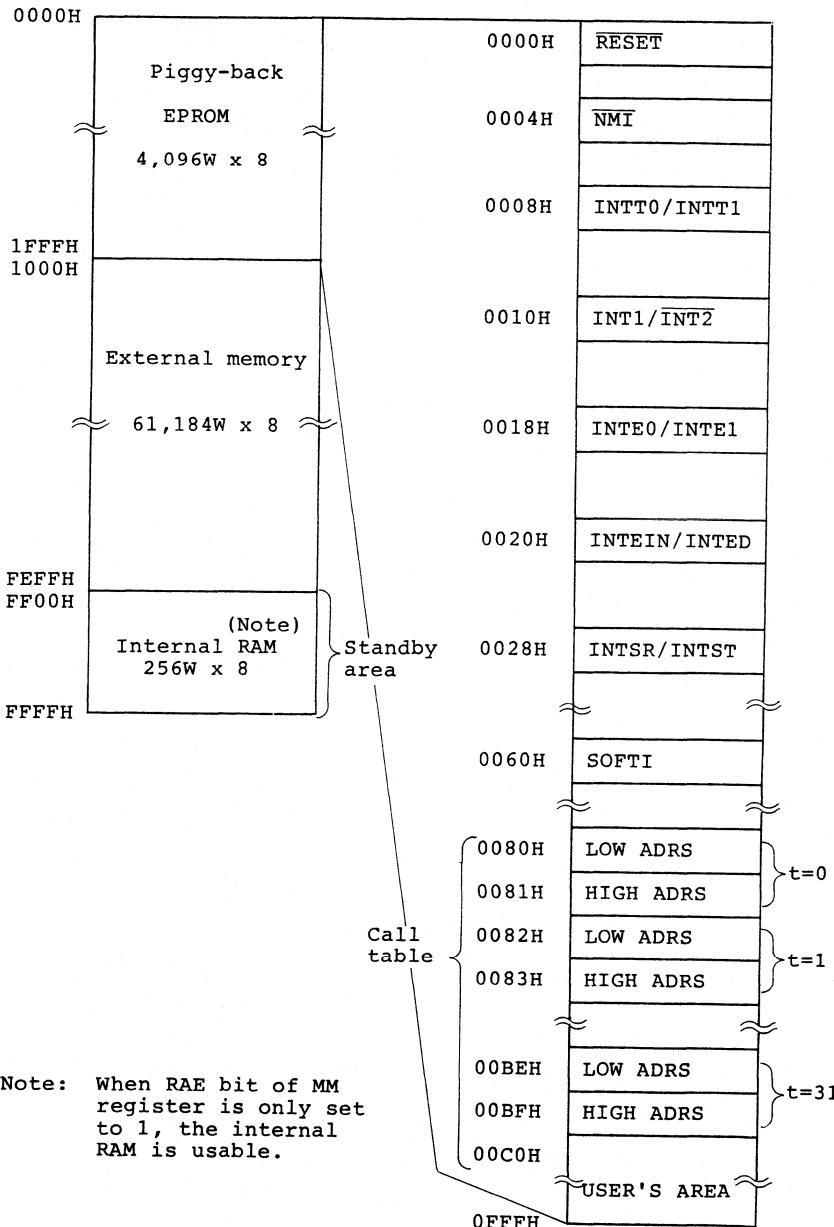


Fig. A-3 Memory map (uPD78C11/78C11A mode)



**Note:** When RAE bit of MM register is only set to 1, the internal RAM is usable.

### 3. MEMORY MAPPING Register (MM)

The MEMORY MAPPING register (MM) is an 8-bit register that contains the piggy-back EPROM access addressing function (MM6 and MM7) as well as the uPD78C11/78C11A/78C12A/78C14/78C14A control function.

Fig. A-4 shows the MEMORY MAPPING register format.

When MM7 and MM6 are set to 00, piggy-back EPROM addresses 0000H-3FFFFH (16K-byte area) are accessed and the external memory capacity becomes 48K bytes. When MM7 and MM6 are set to 10, addresses 0000H-0FFFH (4K-byte area) are accessed and the external memory capacity becomes 60K bytes.

The MM7 (and MM6) bits are significant only for uPD78CG14/uPD78CP14 (Note). Even if data is written into the bits on uPD78C14/78C14A/78C12A/78C11/78C11A, the CPU ignores it. Thus, programs developed on piggy-back EPROM can be transported to mask ROM as they are.

When RESET is input, uPD78CG14 MM7, MM6, MM2, MM1, and MM0 are initialized to 0. Thus, the CPU starts operation in the uPD78C14 single chip mode.

When RESET is input, the RAE bit becomes undefined. Be sure to initialize at the beginning of the program.

Note: See Chapter 12.

Fig. A-4 MEMORY MAPPING Register Format

| 7   | 6   | 5 | 4 | 3   | 2   | 1   | 0   |
|-----|-----|---|---|-----|-----|-----|-----|
| MM7 | MM6 | - | - | RAE | MM2 | MM1 | MM0 |

|   |   |   |                |                                     |                                                                 |
|---|---|---|----------------|-------------------------------------|-----------------------------------------------------------------|
| 0 | 0 | 0 | Port mode      | Single chip                         | PD7=0=input mode<br>PF7=0=port mode                             |
| 0 | 0 | 1 |                |                                     | PD7=0=output mode<br>PF7=0=port mode                            |
| 0 | 1 | 0 | Expansion mode | 256 bytes                           | PD7=0=expantion mode<br>PF7=0=port mode                         |
| 1 | 1 | 0 |                | 4K byts                             | PD7=0=expansion mode<br>PF3=0=expansion mode<br>PF7=4=port mode |
| 1 | 1 | 0 |                | 16K byts                            | PD7=0=expansion mode<br>PF5=0=expansion mode<br>PF7=6=port mode |
| 1 | 1 | 1 |                | 48K, 56K, or<br>(Note)<br>60K bytes | PD7=0=expansion mode<br>PF7=0=expansion mode                    |

Note: The size is selected according to how the MM7 and MM6 bits are set.

Internal RAM access

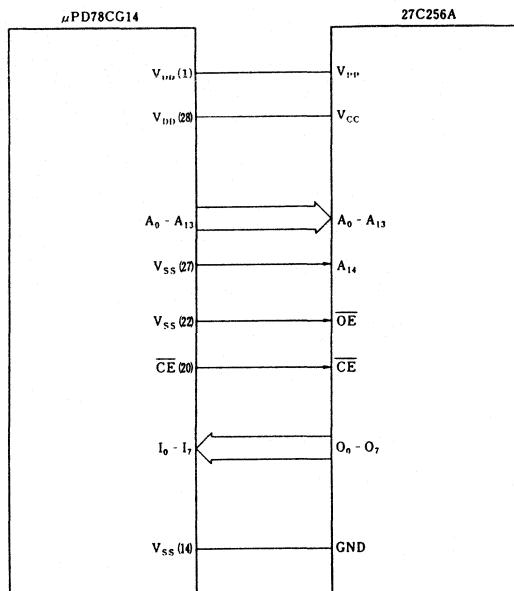
|   |         |
|---|---------|
| 0 | Disable |
| 1 | Enable  |

Piggy-back memory access

|   |   |                                                                                                       |
|---|---|-------------------------------------------------------------------------------------------------------|
| 0 | 0 | EPROM area of addresses 0000H-3FFFH mounted on the upper pin block is accessed (uPD78C14/78C14A mode) |
| 0 | 1 | EPROM area of addresses 0000H-1FFFH mounted on the upper pin block is accessed (uPD78C12A mode)       |
| 1 | 0 | EPROM area of addresses 0000H-1FFFH mounted on the upper pin block is accessed (uPD78C11/78C11A mode) |
| 1 | 1 | Undefined.                                                                                            |

## 4. Interface with EPROM

Fig. A-5 Connection to 27C256A



**Caution:** To use uPD2764, uPD27C64, or uPD27128, a high pulse must be input to pin 27 (PGM). Do not insert only pin 27 in the socket and input external high pulse to the pin.



**APPENDIX B DEVELOPMENT TOOLS**

The following development tools are provided to develop a system using the µCOM-87AD:

|                |                                                                                                                                                            |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Hardware tools |                                                                                                                                                            |
| IE-78C11-M     | In circuit Emulator<br>to be connected via serial interface<br>(RS232) to Terminal/PC                                                                      |
| EB-78C10       | Evaluation Board<br>to be connected via serial interface<br>(RS 232) to Terminal/PC                                                                        |
| Software tools |                                                                                                                                                            |
| CCMSD-I5DD-87  | Software package for MS-Dos incl.<br>C-Compiler<br>Assembler (relocatable)<br>Linker-Locater<br>Library master<br>(IBM-PC format N 5 1/4" DS/DD)           |
| CCVMS-OT16-87  | Software package for VMS incl.<br>C-Compiler<br>Assembler (relocatable)<br>Linker-Locater<br>Library master<br>(VAX/VMS 1600 bpi magtape backup format)    |
| CCUNX-OT16-87  | Software package for Ultrix incl.<br>C-Compiler<br>Assembler (relocatable)<br>Linker-Locater<br>Library master<br>(VAX/Ultrix 1600 bpi magtape Tar format) |
| RAMSD-I5DD-87  | Software package for MS-Dos incl.<br>Assembler (relocatable)<br>Linker-Locater<br>Library master<br>(IBM PC format N 5 1/4" DS/DD)                         |
| RAVMS-OT16-87  | Software package for VMS incl.<br>Assembler (relocatable)<br>Linker-Locater<br>Library master<br>(VAX/VMS 1600 bpi magtape backup format)                  |
| RAUNX-OT16-87  | Software package for Ultrix incl.<br>Assembler (relocatable)<br>Linker-Locater<br>Library master<br>(VAX/Ultrix 1600 bpi magtape Tar format)               |

all Hardware and Software tools include an users manual to help for installation and giving maximum benefit to the customer

## APPENDIX C INSTRUCTION INDEX

| Instruction     | Page   | Instruction    | Page   | Instruction    | Page   |
|-----------------|--------|----------------|--------|----------------|--------|
| ACI A, byte     | 13-79  | BLOCK          | 13-37  | DSBB EA, rp3   | 13-123 |
| ACI r, byte     | 13-79  | CALB           | 13-150 | DSLL EA        | 13-143 |
| ACI sr2, byte   | 13-80  | CALF word      | 13-151 | DSLR EA        | 13-144 |
| ADC A, r        | 13-53  | CALL word      | 13-149 | DSUB DA, rp3   | 13-123 |
| ADC r, A        | 13-53  | CALT word      | 13-151 | DSUBNB EA, rp3 | 13-124 |
| ADCW wa         | 13-105 | CLC            | 13-135 | DXR DA, rp3    | 13-125 |
| ADCX rpa        | 13-67  | DAA            | 13-134 | EADD EA, r2    | 13-120 |
| ADD A, r        | 13-52  | DADC DA, rp3   | 13-121 | EI             | 13-160 |
| ADD r, A        | 13-52  | DADD EA, rp3   | 13-120 | EQA A, r       | 13-64  |
| ADDNC A, r      | 13-54  | DADDNC EA, rp3 | 13-121 | EQA r, A       | 13-65  |
| ADDNC r, A      | 13-54  | DAN EA, rp3    | 13-124 | EQAW wa        | 13-113 |
| ADDNCW Wa       | 13-106 | DCR r2         | 13-132 | EQAX rpa       | 13-75  |
| ADDNCX rqa      | 13-68  | DCRW wa        | 13-132 | EQI A, byte    | 13-99  |
| ADDW wa         | 13-105 | DCX EA         | 13-133 | EQI r, byte    | 13-100 |
| ADDX rpa        | 13-66  | DCX rp         | 13-133 | EQI sr2, byte  | 13-100 |
| ADI A, byte     | 13-77  | DEQ EA, rp3    | 13-127 | EQIW wa, byte  | 13-118 |
| ADI r, byte     | 13-78  | DGT EA, rp3    | 13-126 | ESUB EA, r2    | 13-122 |
| ADI sr2, byte   | 13-78  | DI             | 13-161 | EXA            | 13-36  |
| ADINC A, byte   | 13-81  | DIV r2         | 13-129 | EXH            | 13-37  |
| ADINC r, byte   | 13-81  | DLT EA, rp3    | 13-126 | EXX            | 13-36  |
| ADINC sr2, byte | 13-82  | DMOV EA, rp3   | 13-38  | GTA A, r       | 13-61  |
| ANA A, r        | 13-58  | DMOV EA, sr4   | 13-39  | GTA r, A       | 13-62  |
| ANA r, A        | 13-59  | DMOV rp3, EA   | 13-38  | GTAW wa        | 13-111 |
| ANAW wa         | 13-109 | DMOV sr3, EA   | 13-39  | GTAX rpa       | 13-73  |
| ANAX rpa        | 13-71  | DNE EA, rp3    | 13-127 | GTI A, byte    | 13-94  |
| ANI A, byte     | 13-88  | DOFF EA, rp3   | 13-128 | GTI r, byte    | 13-94  |
| ANI r, byte     | 13-89  | DON EA, rp3    | 13-128 | GTI sr2, byte  | 13-95  |
| ANI sr2, byte   | 13-89  | DOR EA, rp3    | 13-125 | GTIW wa, byte  | 13-116 |
| ANIW wa, byte   | 13-115 | DRLL EA        | 13-142 | HLT            | 13-162 |
| BIT bit, wa     | 13-156 | DRLR EA        | 13-142 | INR r2         | 13-130 |

| Instruction   | Page   | Instruction     | Page   | Instruction   | Page   |
|---------------|--------|-----------------|--------|---------------|--------|
| INRW wa       | 13-130 | MUL r2          | 13-129 | ORA r, A      | 13-60  |
| INX EA        | 13-131 | MVI sr2, byte   | 13-30  | ORAW wa       | 13-109 |
| INX rp        | 13-131 | MVI r, byte     | 13-30  | ORAX rpa      | 13-71  |
| JB            | 13-145 | MVIW wa, byte   | 13-31  | ORI A, byte   | 13-90  |
| JEA           | 13-149 | MVIX rpa1, byte | 13-32  | ORI r, byte   | 13-91  |
| JMP word      | 13-144 | NEA A, r        | 13-63  | ORI sr2, byte | 13-91  |
| JR word       | 13-146 | NEA r, A        | 13-64  | ORIW wa, byte | 13-115 |
| JRE word      | 13-147 | NEAW wa         | 13-112 | POP rpl       | 13-49  |
| LBCD word     | 13-44  | NEAX rpa        | 13-75  | PUSH rpl      | 13-48  |
| LDAW wa       | 13-33  | NEGA            | 13-136 | RET           | 13-154 |
| LDAX rpa2     | 13-35  | NEI A, byte     | 13-97  | RETI          | 13-156 |
| LDEAX rpa3    | 13-46  | NEI r, byte     | 13-98  | RETS          | 13-155 |
| LDED word     | 13-45  | NEI sr2, byte   | 13-98  | RLD           | 13-136 |
| LHLD word     | 13-45  | NEIW wa, byte   | 13-117 | RLL r2        | 13-138 |
| LSPD word     | 13-46  | NOP             | 13-160 | RLR r2        | 13-139 |
| LTA A, r      | 13-62  | OFFA A, r       | 13-66  | RRD           | 13-137 |
| LTA r, A      | 13-63  | OFFAW wa        | 13-114 | SBB A, r      | 13-56  |
| LTAW wa       | 13-111 | OFFAX rpa       | 13-77  | SBB r, A      | 13-57  |
| LTAX rpa      | 13-74  | OFFI A, byte    | 13-103 | SBBW wa       | 13-107 |
| LTI a, byte   | 13-95  | OFFI r, byte    | 13-103 | SBBX rpa      | 13-69  |
| LTI r, byte   | 13-96  | OFFI sr2, byte  | 13-104 | SBCD word     | 13-40  |
| LTI sr2, byte | 13-97  | OFFIW wa, byte  | 13-119 | SBI A, byte   | 13-84  |
| LTIW wa, byte | 13-117 | ONA A, r        | 13-65  | SBI r, byte   | 13-85  |
| LXI rp2, word | 13-50  | ONAW wa         | 13-113 | SBI sr2, byte | 13-86  |
| MOV r1, A     | 13-26  | ONAX rpa        | 13-76  | SDED word     | 13-41  |
| MOV A, r1     | 13-26  | ONI A, byte     | 13-101 | SHLD word     | 13-41  |
| MOV sr, A     | 13-27  | ONI r, byte     | 13-101 | SK f          | 13-157 |
| MOV A, srl    | 13-28  | ONI sr2, byte   | 13-102 | SKIT irf      | 13-158 |
| MOV r, word   | 13-28  | ONIW wa, byte   | 13-109 | SKN f         | 13-158 |
| MOV word, r   | 13-29  | ORA A, r        | 13-59  | SKNIT irf     | 13-159 |

| Instruction     | Page   |
|-----------------|--------|
| SLL r2          | 13-139 |
| SLLC r2         | 13-140 |
| SLR r2          | 13-140 |
| SLRC r2         | 13-141 |
| SOFTI           | 13-153 |
| SSPD word       | 13-42  |
| STAW wa         | 13-32  |
| STAX rpa2       | 13-33  |
| STEAX rpa3      | 13-43  |
| STC             | 13-135 |
| STOP            | 13-162 |
| SUB A, r        | 13-55  |
| SUB r, A        | 13-56  |
| SUBNB A, r      | 13-57  |
| SUBNB r, A      | 13-58  |
| SUBNEW wa       | 13-108 |
| SUBNBX rpa      | 13-70  |
| SUBW wa         | 13-107 |
| SUBX rpa        | 13-69  |
| SUI A, byte     | 13-82  |
| SUI r, byte     | 13-83  |
| SUI sr2, byte   | 13-84  |
| SUINB A, byte   | 13-86  |
| SUINB r, byte   | 13-87  |
| SUINB sr2, byte | 13-88  |
| TABLE           | 13-50  |
| XRA A, r        | 13-60  |
| XRA r, A        | 13-61  |
| XRAW wa         | 13-110 |
| XRAZ rpa        | 13-72  |
| XRI A, byte     | 13-92  |
| XRI r, byte     | 13-92  |
| XRI sr2, byte   | 13-93  |

# NEC Customer Services

## NEC's Commitment to Information

Our offices throughout Europe are always at your service for comprehensive support. Here are some of the technical services we provide:

- INSECT
- Seminars
- Update service
- Hotline
- Mailing list
- University program

### INSECT

#### Information System and Electronic CaTalog.

This is an on-line information service. Via a telephone link you can call up the latest data on all VLSI devices available from NEC. This includes enhancements, news and the most recent application know-how.

The service is free to our customers and other interested parties. For a menu-driven guest session, you can dial in to INSECT via the international packet switching network – in Germany this is DATEX-P – using of these numbers

45 21 10 13 020/030  
and responding to the request for USERNAME and PASSWORD simply with "customer".

### Seminars

No-one is more aware than NEC of the difference that a brief intensive training course can make to your mastery of advanced and often complex devices. We hold regular workshops and seminars at local NEC offices, in our Düsseldorf headquarters, or on customer premises. For information on NEC workshops and seminars, please contact your nearest office.

## Update Service

When you buy an evaluation package from NEC, you become automatically entitled to one year's free updates for both hardware and software. All updates reach you fast and reliably via a courier service. In a field where rapid changes are the norm, you can be thus be sure of working with the most up-to-date development tools.

## Hotlines

NEC's offices located throughout Europe are responsible for technical support and customer services. On the back cover of this brochure you can check which office is most convenient for you to contact. You are also welcome to contact our European headquarters in Düsseldorf directly.

## Mailing list

Our engineering staff produces frequent additions to the available technical documentation in the form of application notes, product news and technical letters. If you would like to be included on our mailing list for this documentation, please inform us.

## University Program

Many of our products, because of their complexity and dedicated application support through EB tools, are interesting subjects for graduate studies. NEC is always ready to discuss this possibility.



## EUROPEAN DISTRIBUTORS

### AUSTRIA

G. DEMEL  
HANDELSGESELLSCHAFT MBH  
HOFFMEISTERGASSE 8-10/1  
1120 WIEN  
TEL.: (222) 8132 50 70

### BELGIUM

INTRA ELECTRONICS PVBA  
BOSUUL 80, BUS 1  
2100 DEURNE  
TEL.: (03) 3 25 23 20  
TLX: 31102

MALCHUS ELECTRONICS B.V.B.A.  
PLANTIN EN MORETUSLEI 172  
2018 ANTWERPEN  
TEL.: (03) 2 35 32 72  
TLX: 33 637

INNOCIRCUIT BENELUX B.V.B.A.  
PLANTIN EN MORETUSLEI 172  
2018 ANTWERPEN  
TEL.: (03) 2 35 32 72 (B)  
TEL.: (0031) 10-4 27 77 80  
TLX: 33 637

DENMARK  
MER-EL A/S  
VED KLAEDEBØ 18  
2970 HOERSHOLM  
TEL.: (2) 57 10 00  
TLX: 37 360

FINLAND  
OY COMDAX AB  
ITAELAHDENKUTA 23A  
PI 1

00210 HELSINKI  
TEL.: (0) 67 02 77  
TLX: 57 125 876

### FRANCE

ASAP

2 AVENUE DES CHAUMES  
78180 MONTIGNY LE BRETONNEUX  
TEL.: (1) 30 43 82 33  
TLX: 698 887

### ITALY

C.C.I.  
5, RUE MARCELIN BERTHELOT  
BP 92  
92164 ANTONY  
TEL.: (1) 46 66 21 82  
TLX: 203 881

### CGE COMPOSANTS

32 RUE GRANGE DAME ROSE  
92360 MEUDON  
TEL.: (1) 46 30 24 25  
TLX: 632 118

PANTEC (GROUPE SCIENTECH)  
29 RUE MARCEL DASSAULT  
92100 BOULOGNE  
TEL.: (1) 46 20 06 74  
TLX: 633 048

### GERMANY

BIT-ELECTRONIC AG  
DINGOLFINGER STRASSE 6  
8000 MÜNCHEN 80  
TEL.: (089) 41 80 07 0  
TLX: 5 212 931

GLEICHMANN + CO ELECTRONICS  
GMBH  
WORMSER STRASSE 34  
6710 FRANKENTHAL  
TEL.: (0623) 250 56  
TLX: 4 65 270

GLYN GMBH  
SCHÖNE AUSSICHT 30  
6272 NIEDERNAHUSEN  
TEL.: (06127) 80 77  
TLX: 4 186 911

H3W ELEKTRONIK VERTRIEBS GMBH  
STAHLGRUBERRING 12  
8000 MÜNCHEN 82  
TEL.: (089) 42 92 71  
TLX: 2 124 514

MICROSCAN GMBH  
ÜBERSEERING 31  
2000 HAMBURG 60  
TEL.: (040) 6 32 00 30  
TLX: 2 132 288

REIN ELEKTRONIK GMBH  
LÖTSCHERWEG 66  
4054 NETTETAL 1  
TEL.: (02153) 73 31 11  
TLX: 854 251

SYSTEM ELEKTRONIK VERTRIEBS  
GMBH  
HEESFELD 4  
3300 BRAUNSCHWEIG  
TEL.: (0511) 31 40 95  
TLX: 952 351

ULTRATRONIK GMBH  
GEWERBESTRASSE 4  
8036 HERRSCHING  
TEL.: (08152) 37 09-0  
TLX: 5 26 459

UNIELECTRONIC VERTRIEBS GMBH  
IM GEFIERTH 11A  
6072 DREIEICH 1 B. FRANKFURT  
TEL.: (06103) 35 175  
TLX: 4 112 13

ITALY  
ADELSY S.R.L.  
VIA DEL FONDITORE, 5  
LOCALITA' ROVERI  
40127 BOLOGNA  
TEL.: (51) 53 219  
TLX: 510 226

CLAIRTON S.P.A.  
VIA GALLARATE, 211  
20151 MILANO  
TEL.: (1) 33 40 4000  
TLX: 313 843

DIS EL. S.P.A.  
VIA ALA DI STURA 71  
10148 TORINO  
TEL.: (011) 22 01 522  
TLX: 215 118

MELCHIONI S.P.A.  
VIA COLLETTA, 37  
20135 MILANO  
TEL.: (01) 57 941  
TLX: 315 293

### NETHERLANDS

MALCHUS B.V.  
FOKKERSTRASSE 511  
3125 BD SCHIEDAM  
TEL.: (10) 4 27 77 77  
TLX: 215 98

INNOCIRCUIT BENELUX B.V.  
POSTBUS 48  
3100 AA SCHIEDAM  
TEL.: (10) 4 27 77 80  
TLX: 215 98

INTRA ELECTRONICS B.V.  
GULBERG 33  
5674 TE NUENEN  
TEL.: (40) 83 80 09  
TLX: 59 418

### NORWAY

JAKOB HATTELAND ELECTRONIC A/S  
PB. 25  
5578 NEDRE VATS  
TEL.: (47) 63 111  
TLX: 428 50

### PORTUGAL

AMPEREL S.A.  
AV. FONTES PEREIRA DE MELO 47, 4D  
1000 LISBOA  
TEL.: (1) 53 26 98  
TLX: 185 98

### SPAIN

COMELTA SA  
EMILIO MUÑOZ 41, NAVE 1-1-2  
28037 MADRID  
TEL.: (1) 7 54 30 77  
TLX: 42 007

### VENCO

CALLE GALILEO 249  
08028 BARCELONA  
TEL.: (3) 330 97 51  
TLX: 98 266

### SWEDEN

NAX AB KOMPONENTBOLAGET  
BOX 1415  
17104 SOLNA  
TEL.: (8) 50 51 40  
TLX: 179 12

NORDQVIST & BERG  
BOX 1458  
17128 SOLNA  
TEL.: (8) 76 67 10  
TLX: 10 407

TH'S ELEKTRONIK  
BOX 3027  
16303 SPAANGA  
TEL.: (8) 36 29 70  
TLX: 11145

SWITZERLAND  
MEMOTEC AG  
GASWERKSTRASSE 32  
4901 LANGENTHAL  
TEL.: (63) 28 11 22  
TLX: 9 62 550

### UNITED KINGDOM

2001 ELECTRONIC COMPONENTS  
WOOLMERS WAY  
STEVENAGE  
HERTS  
SG1 3AJ  
TEL.: (438) 74 20 01  
TLX: 827 701

ANZAC COMPONENTS LIMITED  
822, YEOVIL ROAD  
SLOUGH TRADING ESTATE  
SLOUGH  
BERKS  
SL1 4JA  
TEL.: (62 86) 44 11  
TLX: 847 949

CELDIS LIMITED  
37 LOVEROCK ROAD  
READING  
BERKS  
RG3 1EL  
TEL.: (7 34) 58 51 71  
TLX: 848 370

IMPULSE ELECTRONICS LIMITED  
HAMMOND HOUSE  
CATHERHAM  
SURREY  
CR3 6XG  
TEL.: (8 83) 4 70 11  
TLX: 291 496

S.T.C. MULTICOMPONENTS LIMITED  
EDINBURGH WAY  
HARLOW  
ESSEX  
CM20 2DF  
TEL.: (2 79) 44 11 44  
TLX: 818 763

V.S.I. ELECTRONICS (UK) LIMITED  
ROYDONBURY INDUSTRIAL PARK

HORSCROFT ROAD  
HARLOW  
ESSEX  
CM19 5BY  
TEL.: (2 79) 2 96 66  
TLX: 813 87





## **NEC OFFICES**

NEC Electronics (Europe) GmbH, Oberrather Str. 4, 4000 Düsseldorf 30, W. Germany.  
Tel. (02 11) 65 03 01, Telex 8 58 996-0

NEC Electronics (Germany) GmbH, Oberrather Str. 4, 4000 Düsseldorf 30,  
Tel. (02 11) 65 03 02, Telex 8 58 996-0  
- Königstr. 12, 3000 Hannover 1, Tel. (05 11) 31 60 91, Telex 9 230 109  
- Arabellastr. 17, 8000 München 81, Tel. (0 89) 92 10 03-0, Telex 5 22 971  
- Heilbronner Str. 314, 7000 Stuttgart 30, Tel. (07 11) 89 09 10, Telex 7 252 220

NEC Electronics (BNL) - Boschdijk 187a, NL-5612 HB Eindhoven, Tel. (0 40) 44 58 45,  
Telex 51 923

NEC Electronics (Scandinavia) - Svärdvägen 25 B, S-18233 Danderyd,  
Tel. (08) 75 36 020, Telex 13 839

NEC Electronics (France) S.A., 9, rue Paul Dautier, B.P. 187,  
F-78142 Vélizy Villacoublay Cedex, Tél. (1) 39 46 96 17, Télex 699 499

NEC Electronics (France) S.A., Representacion en Espana, Edificio «La Caixa»,  
Paseo de la Castellana 51, E-28046 Madrid, Tél. (1) 41 94 150, Télex 41 316

NEC Electronics Italiana S.R.L., Via Fabio Filzi, 25A, I-20124 Milano, Tel. (02) 67 09 108,  
Telex 315 355  
- Rome Office, Via Monte Cervialto, 131, I-00139 Rome,  
Tel. (06) 8 11 12 91, Telex 623 323

NEC Electronics (UK) Ltd., Cygnus House, Sunrise Park Way, Milton Keynes, MK14 6NP,  
Tel. (09 08) 69 11 33, Telex 826 791  
- Dublin Office, 34/35 South William Street, Dublin 2, Ireland, Tel. (00 01) 79 42 00  
- Scotland Office, Block 3, Carfin Industrial Estate Motherwell ML1 4UL, Scotland,  
Tel. (6 98) 73 22 21, Telex 777 565

NEC does not assume any responsibility for any circuits shown or  
claim that they are free from patent infringement.  
NEC reserves the right to make changes any time without notice.  
© by NEC Electronics (Europe) GmbH