

**USER'S MANUAL**

**NEC**

**$\mu$ PD78312A**

**16/8-BIT SINGLE-CHIP MICROCOMPUTER**

**$\mu$ PD78310A**

**$\mu$ PD78312A**

**$\mu$ PD78P312A**

**$\mu$ PD78310A(A)**

**$\mu$ PD78312A(A)**

Document No. IEU-1265D  
(O. D. No. IEM-5086F)  
Date Published October 1993 P  
Printed in Japan

**$\mu$ PD78312A**

**16/8-BIT SINGLE-CHIP MICROCOMPUTER**

**$\mu$ PD78310A**

**$\mu$ PD78312A**

**$\mu$ PD78P312A**

**$\mu$ PD78310A(A)**

**$\mu$ PD78312A(A)**

Major Revisions in This Version

Section	Description
4-19	Correction of numeric failure in Figure 4-12 "Count Unit Input Pin Noise Removal Example"
4-99	Correction of number of system clocks in 4.7 "External Interrupt Request Function"
4-99	Correction of numeric failure in Figure 4-64 "External Interrupt Request Pin Input Noise Removal"
9-6	Addition of 9.5 "One-Time PROM Product Screening"
12-9	Addition of 12.4 "Precautions on POP PSW Instructions"
13-1	Addition of Chapter 13. "Cautions in Chapters"

## PREFACE

**Target** : This manual is intended for the user engineer who understands the uPD78310A, 78310A(A), 78312A, 78312A(A) and 78P312A functions and designs an application system using the uPD78310A, 78310A(A), 78312A, 78312A(A) and 78P312A.

**Purpose** : The manual is intended for the user to understand the uPD78310A, 78310A(A), 78312A, 78312A(A) and 78P312A hardware functions listed in Configuration.

**Configuration:** The manual consists of the following:

- . General description
- . Pin function
- . Internal block function
- . Interrupts
- . Other on-chip peripheral functions
- . Instruction function

**Use** : The manual assumes that the reader has general knowledge of electricity, logical circuits, and microcomputers.

For the user who uses the manual for uPD78310A, 78310A(A), 78312A(A) or 78P312A

+ Unless otherwise noted, the manual explains the uPD78312A as a typical product. To use the manual for uPD78310A, 78310A(A), 78312A(A) or 78P312A, change uPD78312A to uPD78310A, 78310A(A), 78312A(A) or 78P312A.



Application examples in this manual are intended for the use of standard quality grade products in general electrical applications.

If the examples in this manual are to be used in applications where the special quality grade is required, please investigate the quality grade of each part and circuit used.

For the user who has experienced uPD78310,  
78312

- + Check the differences referring to "1.6 DIFFERENCES AMONG FAMILY PRODUCTS" and read the manual centering around the explanation of them.

To look up the instruction function when you know the mnemonics

- + Use the index of instruction (alphabetical order) in Appendix A.

To look up instruction when you do not know its mnemonic, but know the rough function

- + Look up the mnemonic of the instruction in "10.1 INSTRUCTION SET AND ITS OPERATION" and the instruction function in "10.6 DESCRIPTION OF INSTRUCTIONS".

To understand the uPD78310A, 78310A(A), 78312A, 78312A(A) and 78P312A functions in general

- + Read the manual according to the contents.

Remarks: The uPD78310, 78312 and 78P312 are discontinuation products.

Legend : Data representation  
weight : High-order and low-order digits are indicated from left to right.

Active low  
representation :  $\overline{\text{xxx}}$  (pin or signal name is overlined)

Memory map address : Upper part; lower address,  
Lower part; higher address

\* : Explanation of \* in the text

NOTE : Description to which you should pay attention

Remarks : Supplementary explanation to the text

No. representation : Binary number .... xxxxB or  
xxxHB  
Decimal number ... xxxxD  
Hexadecimal  
number ..... xxxxH

**Relevant**

**Documents** : The following documents should also be referred to together.

Any number occurring on the table is of a document.

Document Product Name	Data Sheet	User's Manual	Application Note(I)	Application Note(II)	Special-Function Register Use Table	Instruction Use Table	Instruction Set
uPD78310A	IC-7920	This Manual	IEM-964 (Introductory Volume)	IEA-628 (Floating point operation program Volume)	IEM-5118	IEM-5115	IEM-5116
uPD78312A							
uPD78P312A	IC-7772						
uPD78310A(A)	IC-8272						
uPD78312A(A)							

- Documents for Series Product Selection  
Data Book (16-bit single-chip microcomputer) (IA-118)

## CONTENTS

CHAPTER 1.	GENERAL DESCRIPTION .....	1-1
1.1	Features .....	1-2
1.2	Ordering Information and Quality Grade .....	1-3
1.2.1	Ordering Information .....	1-3
1.2.2	Quality Grade .....	1-4
1.3	Pin Configuration (Top View) .....	1-5
1.3.1	uPD78312A, uPD78310A Pin Configuration .....	1-5
1.3.2	uPD78P312A Pin Configuration .....	1-9
1.4	Block Diagrams .....	1-16
1.4.1	uPD78312A, uPD78310A Block Diagram .....	1-16
1.4.2	uPD78P312A Block Diagram .....	1-17
1.5	Function List .....	1-18
1.6	Differences among Family Products .....	1-20
1.6.1	Differences among uPD78312A, uPD78310A, and uPD78P312A .....	1-20
1.6.2	Differences between uPD78312A and uPD78312 ...	1-21
1.6.3	Differences between uPD78310A & uPD78312A and uPD78310A(A) & uPD78312A(A) .....	1-22
1.7	Application Examples .....	1-23
CHAPTER 2.	PIN FUNCTION .....	2-1
2.1	Normal Operating Mode .....	2-2
2.1.1	P00 to P07 (Port 0) .....	2-2
2.1.2	P10 to P17 (Port 1) .....	2-2
2.1.3	P20 to P27 (Port 2) .....	2-3
2.1.4	P30 to P37 (Port 3) .....	2-5
2.1.5	P40 to P47 (Port 4) .....	2-7
2.1.6	P50 to P57 (Port 5) .....	2-8
2.1.7	$\overline{WR}$ (Write Strobe) .....	2-8
2.1.8	$\overline{RD}$ (Read Strobe) .....	2-8
2.1.9	ALE (Address Latch Enable) .....	2-8
2.1.10	$\overline{EA}$ (External Access) .....	2-9
2.1.11	AN0 to AN3 (Analog Input) .....	2-9
2.1.12	$AV_{REF}$ (Reference Voltage) .....	2-9

2.1.13	AV <sub>SS</sub> (Analog V <sub>SS</sub> ) .....	2-9
2.1.14	X1 and X2 (Crystal) .....	2-9
2.1.15	$\overline{\text{RFSH}}$ (Refresh) .....	2-9
2.1.16	$\overline{\text{RESET}}$ (Reset) .....	2-9
2.1.17	V <sub>DD</sub> .....	2-9
2.1.18	V <sub>SS</sub> .....	2-10
2.1.19	IC .....	2-10
2.2	PROM Mode .....	2-11
2.2.1	A0 to A12 (Address) .....	2-11
2.2.2	D0 to D7 (Data) .....	2-11
2.2.3	$\overline{\text{CE}}$ (Chip Enable) .....	2-11
2.2.4	$\overline{\text{OE}}$ (Output Enable) .....	2-11
2.2.5	PROG (Program) .....	2-11
2.2.6	V <sub>PP</sub> (PROM Power Supply) .....	2-11
2.2.7	V <sub>DD</sub> (Power Supply) .....	2-11
2.2.8	V <sub>SS</sub> (Ground) .....	2-11
2.2.9	IC (Internally Connected) .....	2-12
2.3	Pin Input/Output Circuits .....	2-13
2.4	Recommended Connection of Unused Pins .....	2-15
 CHAPTER 3. CPU ARCHITECTURE .....		 3-1
3.1	Memory Space .....	3-1
3.2	Processor Registers .....	3-5
3.2.1	Control Registers .....	3-6
3.2.2	General Registers .....	3-10
3.2.3	Special Function Registers (SFRs) .....	3-12
3.3	Data Memory Addressing .....	3-20
3.3.1	General Register Addressing .....	3-20
3.3.2	Short Direct Addressing .....	3-21
3.3.3	Special Function Register (SFR) Addressing ...	3-21
 CHAPTER 4. PERIPHERAL HARDWARE FUNCTION .....		 4-1
4.1	Port Function .....	4-1
4.1.1	Hardware Configuration .....	4-1
4.1.2	Port Function .....	4-5
4.2	Clock Generator .....	4-15

4.3	Pulse Input/Output Unit .....	4-17
4.3.1	Count Unit .....	4-17
4.3.2	Capture Unit .....	4-42
4.3.3	PWM Unit .....	4-46
4.3.4	Timer Unit .....	4-50
4.3.5	Real-Time Output Port .....	4-61
4.4	Serial Communication Interface .....	4-67
4.4.1	Serial Communication Interface Configuration .....	4-67
4.4.2	Mode Registers .....	4-69
4.4.3	Baud Rate Generator (BRG) .....	4-75
4.4.4	Serial Communication Interface Operation .....	4-79
4.4.5	Serial Communication Interface Interrupt Requests .....	4-84
4.5	Analog-to-Digital Converter .....	4-86
4.5.1	Analog-to-Digital Converter Configuration .....	4-86
4.5.2	Analog-to-Digital Converter Mode Register (ADM) .....	4-88
4.5.3	Analog-to-Digital Converter Operation .....	4-90
4.5.4	Analog-to-Digital Converter Interrupt Requests .....	4-91
4.6	Time Base Counter and Watchdog Timer .....	4-93
4.6.1	Time Base Counter and Watchdog Timer Configuration .....	4-93
4.6.2	Time Base Counter (TBC) .....	4-95
4.6.3	Watchdog Timer (WDT) .....	4-96
4.7	External Interrupt Request Function .....	4-99
CHAPTER 5. INTERRUPT FUNCTION .....		5-1
5.1	Interrupt Requests .....	5-7
5.1.1	Interrupt Control Hardware Devices .....	5-7
5.1.2	Nonmaskable Interrupt Requests .....	5-11
5.1.3	Maskable Interrupt Requests .....	5-11
5.1.4	Software Interrupt Requests .....	5-13
5.1.5	Multiple Interrupts .....	5-14
5.2	Macro Service Function .....	5-15
5.2.1	Macro Service Function Use Example .....	5-15

5.2.2	Macro Service Function Features .....	5-18
5.2.3	Macro Service Function Operation .....	5-18
5.2.4	Macro Service Channels .....	5-21
5.2.5	Macro Service Control Register .....	5-22
5.3	Context Switching Function .....	5-25
5.3.1	Context Switching Function when Interrupt Request Occurs .....	5-25
5.3.2	Context Switching Function when BRKCS Instruction is Executed .....	5-27
5.3.3	Return from Context Switching Function Branch Address .....	5-28
<b>CHAPTER 6. STANDBY FUNCTION .....</b>		<b>6-1</b>
6.1	Standby Mode Setting and Operation State .....	6-1
6.1.1	Standby Control Register (STBC) .....	6-1
6.1.2	Use of SBF Bit .....	6-4
6.1.3	Clock Varying Mode .....	6-5
6.1.4	HALT Mode .....	6-5
6.1.5	STOP Mode .....	6-6
6.2	Standby Mode Release .....	6-8
6.2.1	HALT Mode Release .....	6-8
6.2.2	STOP Mode Release .....	6-9
6.3	Operation after Standby Mode is Released .....	6-11
<b>CHAPTER 7. RESET FUNCTION .....</b>		<b>7-1</b>
<b>CHAPTER 8. LOCAL BUS INTERFACE FUNCTION .....</b>		<b>8-1</b>
8.1	uPD78312A, uPD78P312A External Device Expansion Function .....	8-1
8.1.1	Memory Expansion Mode Register (MM) .....	8-4
8.1.2	Memory Expansion Example .....	8-6
8.2	uPD78310A External Device Access .....	8-8
8.3	Pseudo-Static RAM Refresh Function .....	8-9
8.3.1	Pulse Refresh Operation .....	8-9
8.3.2	Power Down Self-Refresh Operation .....	8-11
8.3.3	Refresh Mode Register (RFM) .....	8-13

8.3.4	Pseudo-Static Memory Expansion Examples .....	8-15
CHAPTER 9.	uPD78P312A PROGRAMMING .....	9-1
9.1	Operating Mode for PROM Programming .....	9-2
9.2	PROM Writing Procedure .....	9-4
9.3	PROM Reading Procedure .....	9-5
9.4	Data Erasure Procedure (EPROM Product Only) .....	9-6
9.5	One-Time PROM Product Screening .....	9-7
CHAPTER 10.	INSTRUCTION SET .....	10-1
10.1	Instruction Set and Its Operation .....	10-1
10.2	Instruction Execution State Count Estimation ....	10-33
10.3	OP Code of Each Instruction .....	10-38
10.4	Instruction Address Addressing .....	10-63
10.4.1	Relative Addressing .....	10-63
10.4.2	Immediate Addressing .....	10-64
10.4.3	Table Indirect Addressing .....	10-65
10.4.4	Register Addressing .....	10-65
10.4.5	Register Indirect Addressing .....	10-66
10.5	Operand Address Addressing .....	10-67
10.5.1	Register Addressing .....	10-67
10.5.2	Immediate Addressing .....	10-69
10.5.3	Direct Addressing .....	10-70
10.5.4	Short Direct Addressing .....	10-71
10.5.5	Special Function Register (SFR) Addressing ...	10-73
10.5.6	Memory Indirect Addressing .....	10-74
10.5.7	Register Indirect Addressing .....	10-76
10.5.8	Base Addressing .....	10-78
10.5.9	Index Addressing .....	10-79
10.5.10	Base Index Addressing .....	10-80
10.6	Description of Instructions .....	10-81
10.6.1	8-Bit Data Transfer Instructions .....	10-81
10.6.2	16-Bit Data Transfer Instructions .....	10-91
10.6.3	8-Bit Operation Instructions .....	10-96
10.6.4	16-Bit Operation Instructions .....	10-131
10.6.5	Multiply/Divide Instructions .....	10-142



10.6.6	Increment/Decrement Instructions .....	10-144
10.6.7	Shift/Rotation Instructions .....	10-147
10.6.8	BCD Adjustment Instruction .....	10-154
10.6.9	Bit Operation Instructions .....	10-156
10.6.10	Call/Return Instructions .....	10-177
10.6.11	Stack Manipulation Instructions .....	10-182
10.6.12	Unconditional Branch Instructions .....	10-186
10.6.13	Conditional Branch Instructions .....	10-188
10.6.14	Context Switch Instructions .....	10-210
10.6.15	String Instructions .....	10-212
10.6.16	CPU Control Instructions .....	10-220
CHAPTER 11.	SPECIFICATIONS .....	11-1
11.1	Electrical Specifications (uPD78310A, uPD78312A) .....	11-1
11.2	Electrical Specifications (uPD78P312A) .....	11-17
11.3	Electrical Specifications (uPD78310A(A), uPD78312A(A)).....	11-36
11.4	Package Information .....	11-51
CHAPTER 12.	PRECAUTIONS FOR USE .....	12-1
12.1	Precautions Regarding Serial Communication Interface Function .....	12-1
12.2	Precautions for Expansion by the Use of External Devices .....	12-3
12.3	Procautions Regarding Instruction Combinations ..	12-5
12.4	POP PSW Instruction Related Caution .....	12-9
CHAPTER 13.	CAUTIONS IN CHAPTERS .....	13-1
13.1	Chapter 3 "CPU Architecture" Related Caution ....	13-1
13.2	Chapter 4 "Peripheral Hardware Functions" Related Caution .....	13-1
13.3	Chapter 6 "Standby Function" Related Caution ....	13-4
13.4	Chapter 8 "Local Bus Interface" Related Caution .....	13-4



13.5	Chapter 9 "uPD78P312A Programming" Related Caution .....	13-5
13.6	Chapter 10 "Instruction Set" Related Caution ....	13-6
13.7	Chapter 11 "Specifications" Related Caution .....	13-7
APPENDIX A. INSTRUCTION INDEX (ALPHABETIC ORDER) .....		A-1
APPENDIX B. DEVELOPMENT TOOLS .....		B-1

## List of Figures

Figure No.	Title	Page
2-1	Pin Input/Output Circuit List .....	2-14
3-1	Memory Map .....	3-4
3-2	Register Configuration .....	3-5
3-3	PSW Format .....	3-6
3-4	CCW Format .....	3-9
3-5	Memory Locations of General Registers .....	3-10
3-6	Addressing Space of Data Memory .....	3-20
4-1	Basic Structure of Port .....	4-1
4-2	Port Set to Output Port .....	4-2
4-3	Port Set to Input Port .....	4-3
4-4	When Port is Set to Control Mode .....	4-3
4-5	Port Mode Register Format .....	4-7
4-6	Memory Expansion Mode Register Format .....	4-8
4-7	Port 2 Mode Control Register Format .....	4-9
4-8	Port 3 Mode Control Register Format .....	4-10
4-9	Clock Generator Configuration .....	4-15
4-10	System Clock Oscillator External Circuit .....	4-16
4-11	Count Unit Block Diagram .....	4-18
4-12	Count Unit Input Pin Noise Removal Example .....	4-19
4-13	Up/Down Counter Control Register Format .....	4-21
4-14	Capture/Compare Register Control Register Format .	4-24
4-15	Count Unit Input Mode Register Format .....	4-26
4-16	Operation when Compare Preset Mode is Specified ..	4-28
4-17	Count Operation during Up/Down Modulo Mode .....	4-29
4-18	Count Operation Example (Down Count) (1) .....	4-30
4-19	Count Operation Example (Down Count) (2) .....	4-31
4-20	Count Operation Example (Down Count) (3) .....	4-32
4-21	Operation when Capture Mode is Specified .....	4-34
4-22	Mode 1 Operation .....	4-35
4-23	Mode 2 Operation when Internal Clock is Selected for Count Clock .....	4-36

Figure No.	Title	Page
4-24	Mode 2 Operation when External Clock is Selected for Count Clock .....	4-37
4-25	Mode 3 Down Count Operation .....	4-38
4-26	Mode 3 Up Count Operation .....	4-38
4-27	Mode 4 Count Operation Example .....	4-39
4-28	Count Unit Interrupt Request Control Register Formats .....	4-40
4-29	Macro Service Control Register Formats .....	4-42
4-30	Capture Unit Block Diagram .....	4-42
4-31	Capture Mode Register Format .....	4-43
4-32	Free Running Counter Control Register Format .....	4-44
4-33	Free Running Counter Specification .....	4-45
4-34	PWM Unit Block Diagram .....	4-46
4-35	PWM Mode Register Format .....	4-47
4-36	PWM Output Operation Timing .....	4-49
4-37	Timer Unit Block Diagram .....	4-50
4-38	Timer Control Register (TMC0) Format .....	4-52
4-39	Timer Control Register (TMC1) Format .....	4-53
4-40	Timer Unit Output Timing .....	4-56
4-41	Interrupt Requests from Timer Unit .....	4-58
4-42	Timer Unit Interrupt Request Control Register Format .....	4-59
4-43	Macro Service Control Register Format .....	4-60
4-44	Real-Time Output Port Block Diagram .....	4-61
4-45	Real-Time Output Port Control Register Format .....	4-63
4-46	Serial Communication Interface Block Diagram .....	4-68
4-47	Serial Communication Mode Register Format (a) .....	4-72
4-48	Serial Communication Mode Register Format (b) .....	4-73
4-49	Serial Communication Control Register Format .....	4-74
4-50	Baud Rate Generator Configuration .....	4-76
4-51	Transmission Timing during Asynchronous Mode .....	4-80
4-52	Reception Timing during Asynchronous Mode .....	4-81
4-53	Transmission Timing during I/O Interface Mode .....	4-83
4-54	Reception Timing during I/O Interface Mode .....	4-84

Figure No.	Title	Page
4-55	Interrupt Request Control register Format .....	4-84
4-56	Macro Service Control Register Formats .....	4-85
4-57	Analog-to-Digital Converter Block Diagram .....	4-87
4-58	Analog-to-Digital Converter Mode Register Format .	4-89
4-59	Interrupt Request Control Register Formats .....	4-91
4-60	Macro Service Control Register Format .....	4-92
4-61	Time Base Counter and Watchdog Timer Block Diagram .....	4-94
4-62	Time Base Mode Register Format .....	4-95
4-63	Watchdog Timer Mode Register Format .....	4-97
4-64	External Interrupt Request Pin Input Noise Removal .....	4-99
4-65	External Interrupt Mode Register Format .....	4-100
4-66	Interrupt Request Control Register Formats .....	4-101
4-67	Macro Service Control Register Formats .....	4-102
5-1	Interrupt Request Processing Modes .....	5-1
5-2	Automatically Saved/Restored Data by Interrupt Request Service .....	5-3
5-3	Interrupt Service Sequence .....	5-4
5-4	Interrupt Request Control Register Formats .....	5-9
5-5	In-Service Priority Register Format .....	5-10
5-6	3-Level Multiple Interrupts .....	5-14
5-7	Macro Service Operation Example .....	5-16
5-8	Macro Service Operation Example Flowchart .....	5-17
5-9	Macro Service Operation Flow .....	5-19
5-10	Macro Service Channel Configuration .....	5-21
5-11	Macro Service Channel Mapping .....	5-22
5-12	Macro Service Control Register Format .....	5-23
5-13	Context Switching Operation when Interrupt Request Occurs .....	5-26
5-14	Context Switching Operation when BRKCS Instruction is Executed .....	5-27
6-1	Standby Control Register (STBC) Format .....	6-2
6-2	Standby Function Block Diagram .....	6-3

Figure No.	Title	Page
6-3	SBF Bit Processing Routine .....	6-5
6-4	HALT Mode Release when Interrupt Request Occurs ..	6-8
6-5	Macro Service Start during HALT Mode .....	6-9
6-6	STOP Mode Release when a Valid Edge is Input to NMI Pin .....	6-10
7-1	Reset Signal Acknowledgment .....	7-1
7-2	Reset when Power is Turned on .....	7-1
8-1	External Expansion Mode Selected by Setting Memory Expansion Mode Register (uPD78312A, 78P312A) .....	8-3
8-2	MM Register Format .....	8-5
8-3	Memory Expansion Example (for Reference) .....	8-6
8-4	Memory Expansion Register Setting (in Memory Expansion Example) .....	8-7
8-5	uPD78310A Address Space .....	8-8
8-6	Pulse Refresh Operation when Memory is Accessed ..	8-10
8-7	Return from Power Down Self-Refresh Operation ....	8-12
8-8	Refresh Mode Register (RFM) Format .....	8-14
8-9	Pseudo-Static Memory Expansion Example (for Reference) .....	8-15
8-10	Memory Expansion Mode Register Setting (in Pseudo-Static Memory Expansion Example) .....	8-16
9-1	PROM Write/Verify Timings .....	9-4
9-2	PROM Read Timings .....	9-5
12-1	Flowchart of Polling Processing .....	12-2
12-2	Glitch Output of ALE Pin .....	12-3
12-3	Example of Countermeasure Circuit .....	12-4

## List of Tables

Table No.	Title	Page
1-1	Differences among uPD78312A, uPD78310A, and uPD78P312A .....	1-20
1-2	Differences between uPD78312A and uPD78312 .....	1-21
1-3	Differences between uPD78310A & uPD78312A and uPD78310A(A) & uPD78312A(A) .....	1-22
2-1	uPD78P312A Operating Mode .....	2-1
2-2	P20 to p27 Operation .....	2-3
2-3	P30 to P37 Operation .....	2-5
2-4	Pin Input/Output Circuit Types .....	2-13
2-5	Recommended Connection of Unused Pins .....	2-15
3-1	Vector Table List .....	3-1
3-2	General Register Configuration .....	3-11
3-3	Special Function Register (SFR) List .....	3-14
4-1	Port Function and Features .....	4-5
4-2	Port 0, 1 Operation when Read/Write Instruction is Executed (n = 0 to 7) .....	4-11
4-3	Port 2 Operation when Read/Write Instruction is Executed (n = 0 to 7) .....	4-12
4-4	Port 3 Operation when Read/Write Instruction is Executed (n = 0 to 7) .....	4-13
4-5	Port 4, 5 Operation when Read/Write Instruction is Executed (uPD78312A, uPD78P312A) .....	4-13
4-6	Port 4, 5 Operation (uPD78312A, uPD78P312A) .....	4-14
4-7	Internal System Clock Specification .....	4-16
4-8	Operation when Compare Preset Mode is Specified ..	4-28
4-9	Operation when Capture Mode is Specified .....	4-33
4-10	Count Unit Interrupt Request Flag Set Conditions .	4-40
4-11	Count Time during Interval Timer Operating Mode ..	4-55
4-12	Count Time during One-Shot Timer Operating Mode ..	4-57
4-13	Port 0 Access Targets .....	4-64
4-14	Real-Time Output Port Output Timings .....	4-66
4-15	Baud Rate Generator Set Values (for Reference) ...	4-77
4-16	Internal System Clock and FR Bit Setting .....	4-88

Table No.	Title	Page
4-17	Watchdog Timer Count Clock and Overflow Time .....	4-98
5-1	Maximum Wait Time until Interrupt is Acknowledged .....	5-2
5-2	Interrupt Request Source List .....	5-6
5-3	Intergroup Priority Levels .....	5-12
5-4	Number of Basic States Required for Macro Service Operation .....	5-20
5-5	Number of Added States Required for Macro Service Operation .....	5-20
6-1	Operation State during HALT/STOP Mode .....	6-7
6-2	Operation after HALT Mode is Released when Interrupt Request Occurs .....	6-11
7-1	Hardware State after Reset .....	7-2
8-1	P57 to P50 Address Bus Selection .....	8-1
9-1	Pin Functions in Programming Mode .....	9-1
9-2	Operating Mode for PROM Programming .....	9-2
10-1	8-Bit Register Absolute Name/Function Name Correspondence .....	10-3
10-2	16-Bit Register Pair Absolute Name/Function Name Correspondence .....	10-4
10-3	State Count Increase/Instruction Execution .....	10-34
10-4	Number of SFR Accesses of Each Instruction .....	10-35
10-5	Number of Memory Accesses of Each Instruction ....	10-36
12-1	Relevant sfr's and sfrp's .....	12-7
12-2	Relevant Instructions .....	12-7
12-3	saddr Instructions .....	12-8



## CHAPTER 1. GENERAL DESCRIPTION

The uPD78312A, uPD78310A, and uPD78P312A are 16-/8-bit single chip microcomputers belonging to the 78K/III series.

The uPD78312A contains a high-performance 16-bit CPU; the internal operation performance is improved greatly. In addition, a multifunctional pulse input/output unit, general purpose serial interface, and high resolution analog-to-digital converter are integrated on a single chip.

The uPD78312A is applicable to machine system control. External memory of up to 56 Kbytes can be expanded. The external bus consists of eight bits.

The uPD78310A is internal mask ROM-less version of the uPD78312A. External memory of up to 64 Kbytes can be directly accessed.

The uPD78P312A is a product provided by replacing uPD78312A internal mask ROM with PROM. The uPD78P312A is appropriate for production during system development or multiple device small production.

The uPD78312A, uPD78310A, and uPD78P312A are uPD78312, uPD78310, and uPD78P312 function expansion products. The following functions are added:

- . 16-bit data transfer instruction between memory and register pair
- . Counter unit mode 4 (incremental/decremental count by 2-phased inputs)
- . Count start function by using interval timer external trigger

The uPD78312A(A) and uPD78310A(A) are special quality grade versions of the uPD78312A and uPD78310A, respectively.



## 1.1 FEATURES

- o 78K/III series
- o High speed instruction execution by prefetching instruction
  - . 3-byte instruction queue is contained.
  - . Instruction cycle: 500 ns at 12 MHz
- o 96 basic instructions appropriate for control purpose
  - . 16-bit operation instructions
  - . Multiplication and division instructions (16 bits x 16 bits and 32 bits 16 bits)
  - . Bit manipulation instructions
  - . String instructions
- o High-performance interrupt controller contained.
  - . Interrupt acknowledge priority levels can be programmed.
  - . Three processing modes  
(Vectored interrupt function, macro service function, and context switching function)
- o Various on-chip peripheral hardware devices appropriate for machine system control
  - . Multi-functional pulse input/output unit
  - . 8-bit general purpose serial interface
  - . 8-bit resolution analog-to-digital converter
  - . Pseudo-static RAM refresh function
- o On-chip peripheral hardware devices (special function registers) are mapped in memory space

## 1.2 ORDERING INFORMATION AND QUALITY GRADE

### 1.2.1 ORDERING INFORMATION

Ordering Code	Package	On-Chip ROM
uPD78310ACW	64-pin plastic shrink DIP (750 mil)	None
uPD78310AGF-3BE	64-pin plastic QFP (14 x 20 mm)	None
uPD78310AGQ-36	64-pin plastic QUIP	None
uPD78310AL	68-pin plastic QFJ (□950 mil)	None
uPD78312ACW-xxx	64-pin plastic shrink DIP (750 mil)	Mask ROM
uPD78312AGF-xxx-3BE	64-pin plastic QFP (14 x 20 mm)	Mask ROM
uPD78312AGQ-xxx-36	64-pin plastic QUIP	Mask ROM
uPD78312AL-xxx	68-pin plastic QFJ (□950 mil)	Mask ROM
uPD78P312ACW	64-pin plastic shrink DIP (750 mil)	One-time PROM
uPD78P312AGF-3BE	64-pin plastic QFP (14 x 20 mm)	One-time PROM
uPD78P312AGQ-36	64-pin plastic QUIP	One-time PROM
uPD78P312AL	68-pin plastic QFJ (□950 mil)	One-time PROM
uPD78P312ADW	64-pin ceramic shrink DIP with window (750 mil)	EPROM
uPD78P312AR	64-pin ceramic QUIP with window	EPROM
uPD78310ACW(A)	64-pin plastic shrink DIP (750 mil)	None
uPD78310AGF(A)-3BE	64-pin plastic QFP (14 x 20 mm)	None
uPD78310AGQ(A)-36	64-pin plastic QUIP	None
uPD78310AL(A)	68-pin plastic QFJ (□950 mil)	None
uPD78312ACW(A)-xxx	64-pin plastic shrink DIP (750 mil)	Mask ROM
uPD78312AGF(A)-xxx-3BE	64-pin plastic QFP (14 x 20 mm)	Mask ROM
uPD78312AGQ(A)-xxx-36	64-pin plastic QUIP	Mask ROM
uPD78312AL(A)-xxx	68-pin plastic QFJ (□950 mil)	Mask ROM

Remarks: xxx: Code number

## 1.2.2 QUALITY GRADE

Ordering Code	Package	Quality Grade
uPD78310ACW	64-pin plastic shrink DIP (750 mil)	Standard
uPD78310AGF-3BE	64-pin plastic QFP (14 x 20 mm)	Standard
uPD78310AGQ-36	64-pin plastic QUIP	Standard
uPD78310AL	68-pin plastic QFJ (□950 mil)	Standard
uPD78312ACW-xxx	64-pin plastic shrink DIP (750 mil)	Standard
uPD78312AGF-xxx-3BE	64-pin plastic QFP (14 x 20 mm)	Standard
uPD78312AGQ-xxx-36	64-pin plastic QUIP	Standard
uPD78312AL-xxx	68-pin plastic QFJ (□950 mil)	Standard
uPD78P312ACW	64-pin plastic shrink DIP (750 mil)	Standard
uPD78P312AGF-3BE	64-pin plastic QFP (14 x 20 mm)	Standard
uPD78P312AGQ-36	64-pin plastic QUIP	Standard
uPD78P312AL	68-pin plastic QFJ (□950 mil)	Standard
uPD78P312ADW	64-pin ceramic shrink DIP with window (750 mil)	Standard
uPD78P312AR	64-pin ceramic QUIP with window	Standard
uPD78310ACW(A)	64-pin plastic shrink DIP (750 mil)	Special
uPD78310AGF(A)-3BE	64-pin plastic QFP (14 x 20 mm)	Special
uPD78310AGQ(A)-36	64-pin plastic QUIP	Special
uPD78310AL(A)	68-pin plastic QFJ (□950 mil)	Special
uPD78312ACW(A)-xxx	64-pin plastic shrink DIP (750 mil)	Special
uPD78312AGF(A)-xxx-3BE	64-pin plastic QFP (14 x 20 mm)	Special
uPD78312AGQ(A)-xxx-36	64-pin plastic QUIP	Special
uPD78312AL(A)-xxx	68-pin plastic QFJ (□950 mil)	Special

Remarks: xxx: Code number

Please refer to "Quality grade on NEC Semiconductor Devices" (Document number IEI-1209) published by NEC Corporation to know the specification of quality grade on the devices and its recommended applications.

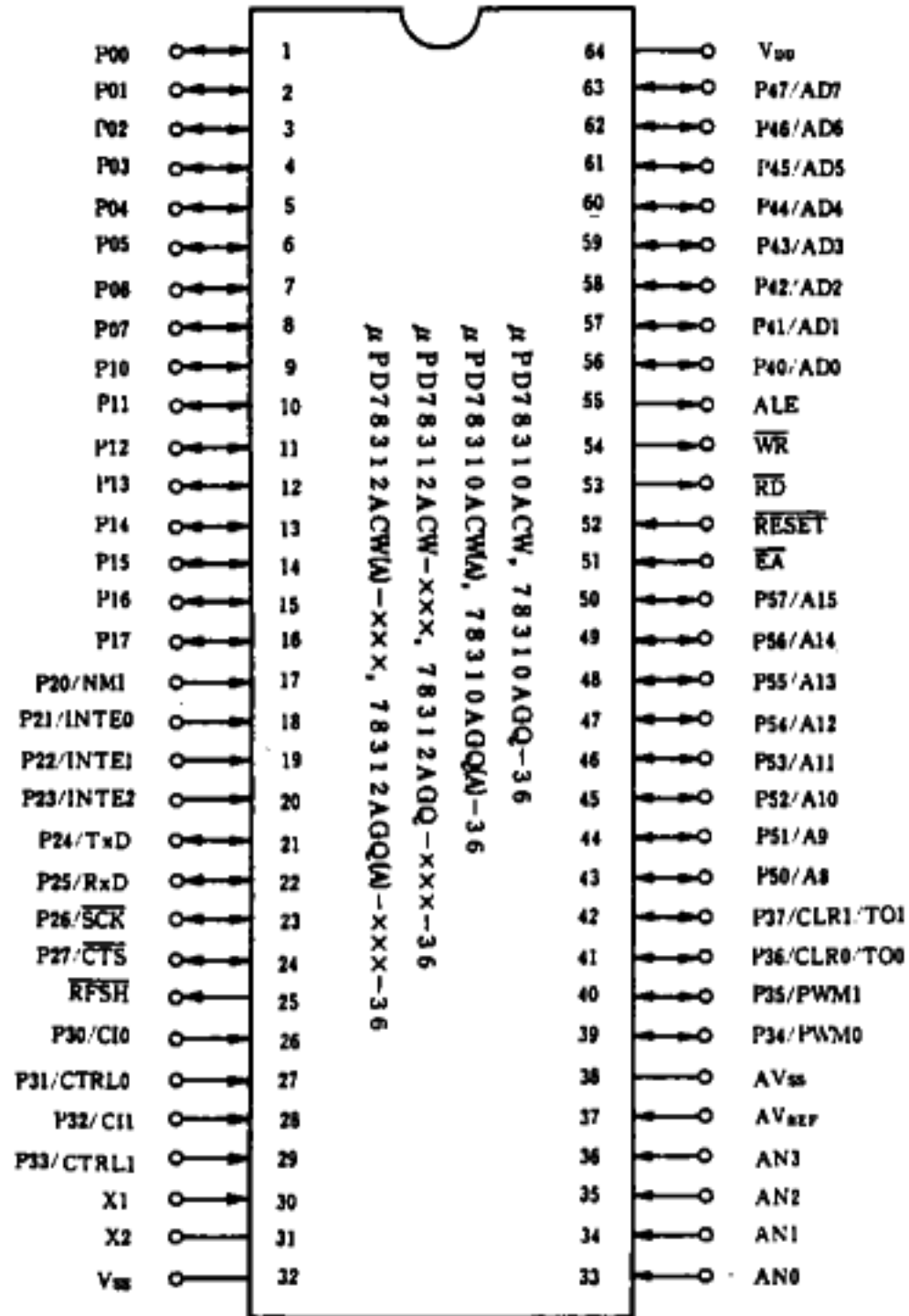
### 1.3 PIN CONFIGURATION (TOP VIEW)

#### 1.3.1 uPD78312A, uPD78310A PIN CONFIGURATION

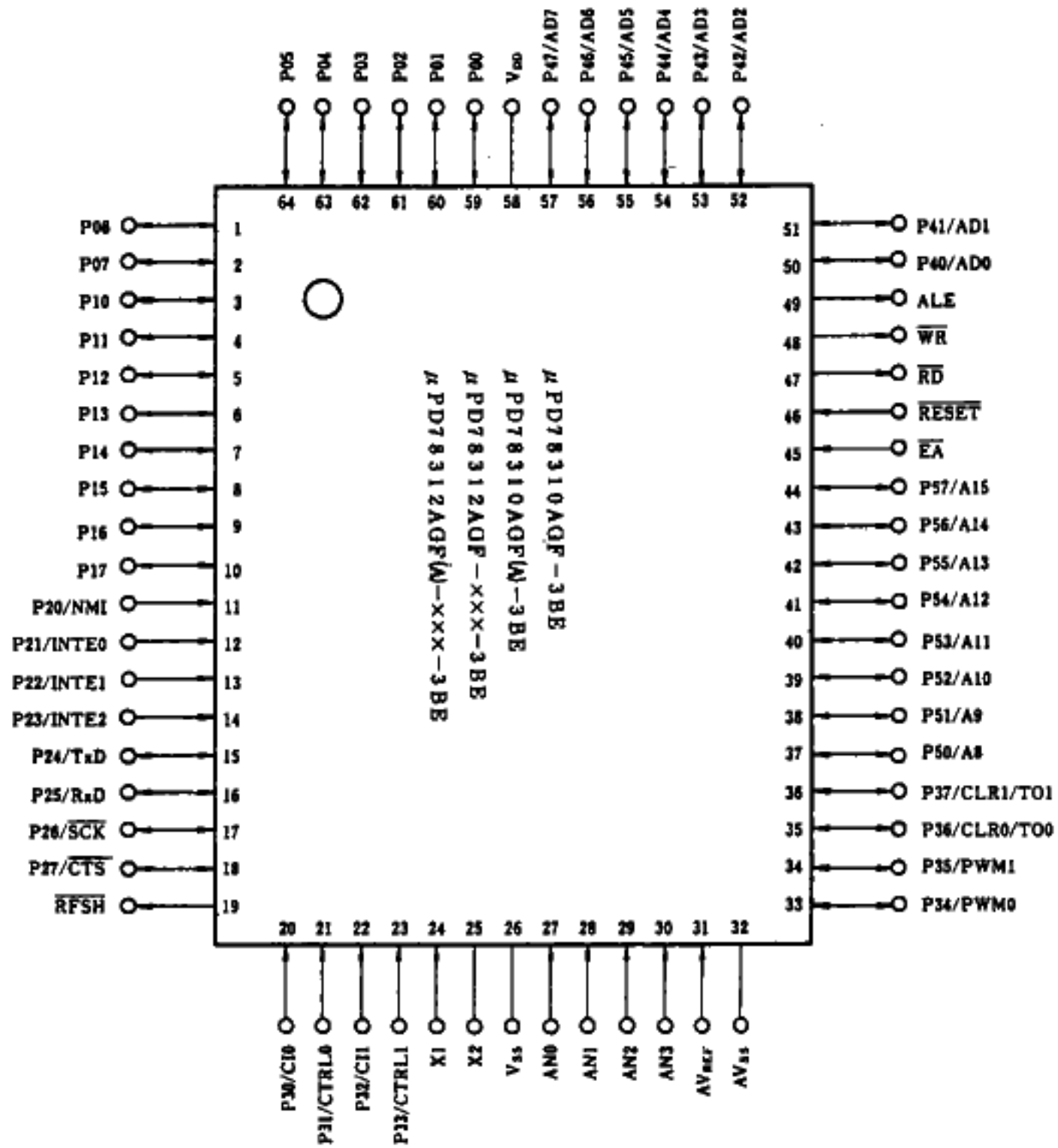
##### Pin IDs

P00-P07:Port0	CI0, CI1	:Count Pulse Input
P10-P17:Port1	CTRL0, CTRL1	:Control Pulse Input
P20-P27:Port2	CLR0, CLR1	:Timer Clear Input
P30-P37:Port3	PWM0, PWM1	:Pulse Width Modulation
P40-P47:Port4		Output
P50-P57:Port5	TO0, TO1	:Timer Output
AD0-AD7:Address/Data Bus	NMI	:Nonmaskable Interrupt
A8-A15 :Address Bus	INTE0, INTE2	:Interrupt From Externals
$\overline{RD}$ :Read Strobe	AN0-AN7	:Analog Input
$\overline{WR}$ :Write Strobe	AVREF	:Reference Voltage
ALE :Address Latch Enable	AVss	:Analog Vss
$\overline{EA}$ :External Access	RxD	:Receive Serial Data
$\overline{RFSH}$ :Refresh	TxD	:Transfer Serial Data
X1, X2 :Crystal	$\overline{SCK}$	:Serial Clock
$\overline{RESET}$ :Reset	$\overline{CTS}$	:Clear To Send
	IC	:Internally Connected

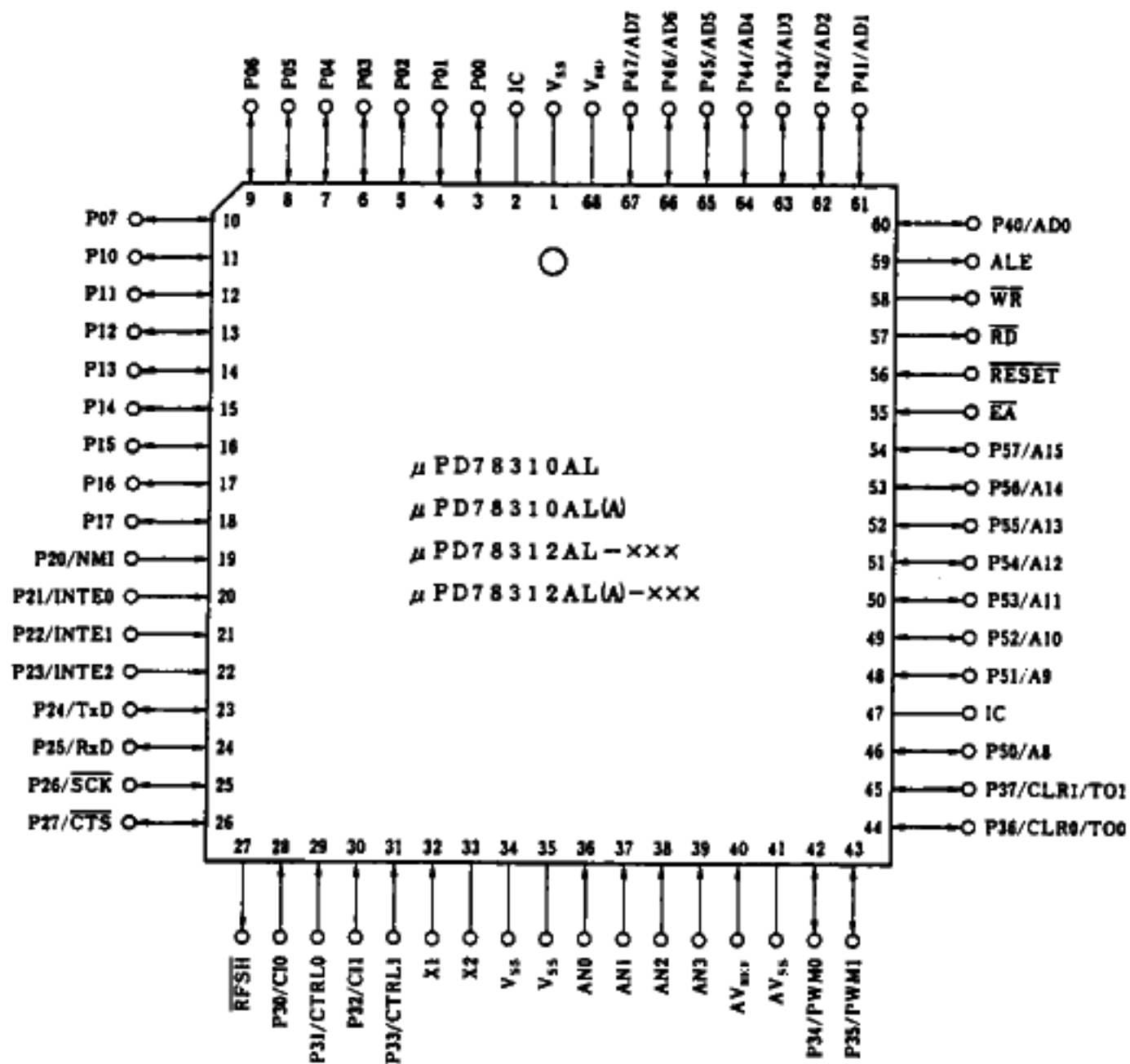
(1) 64-pin plastic shrink DIP/QUIP



(2) 64-pin plastic QFP (14 x 20 mm)



(3) 68-pin plastic QFJ



### 1.3.2 uPD78P312A PIN CONFIGURATION

#### Pin IDs

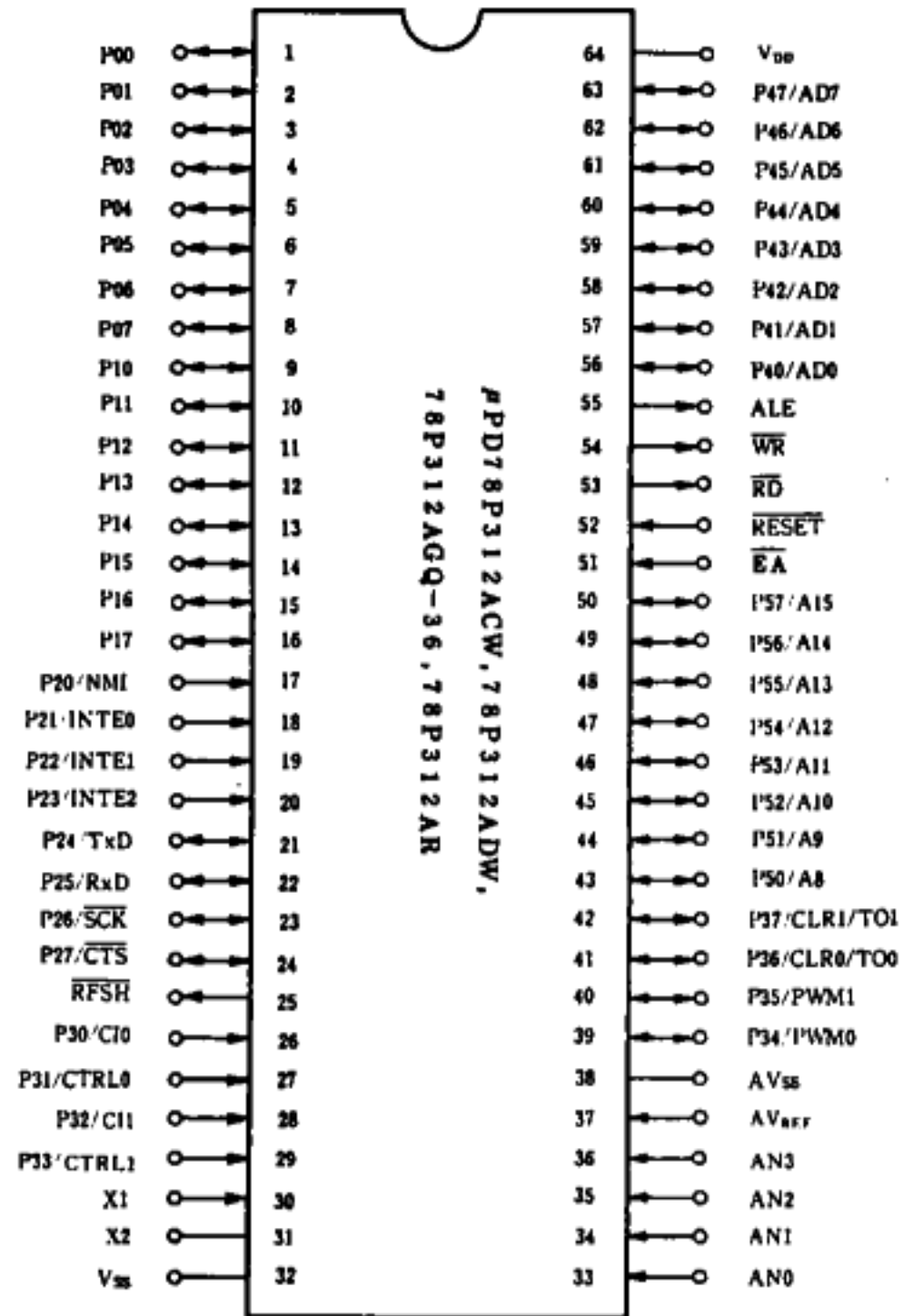
P00-P07: Port0	CI0, CI1	: Count Pulse Input
P10-P17: Port1	CTRL0, CTRL1	: Control Pulse Input
P20-P27: Port2	CLR0, CLR1	: Timer Clear Input
P30-P37: Port3	PWM0, PWM1	: Pulse Width Modulation Output
P40-P47: Port4		
P50-P57: Port5	TO0, TO1	: Timer Output
	NMI	: Nonmaskable Interrupt
AD0-AD7: Address/Data Bus	INTE0-INTE2	: Interrupt From Externals
A8-A15 : Address Bus	AN0-AN7	: Analog Input
$\overline{RD}$ : Read Strobe	$V_{REF}$	: Reference Voltage
$\overline{WR}$ : Write Strobe	$V_{SS}$	: Analog Vss
ALE : Address Latch Enable	RxD	: Receive Serial Data
$\overline{EA}$ : External Access	TxD	: Transfer Serial Data
$\overline{RFSH}$ : Refresh	$\overline{SCK}$	: Serial Clock
X1, X2 : Crystal	$\overline{CTS}$	: Clear To Send
$\overline{RESET}$ : Reset	IC	: Internally Connected
A0-A12 : Address Bus	$\overline{CE}$	: Chip Enable
D0-D7 : Data Bus	$\overline{OE}$	: Output Enable
PROG : Program	$V_{PP}$	: Program Power Supply



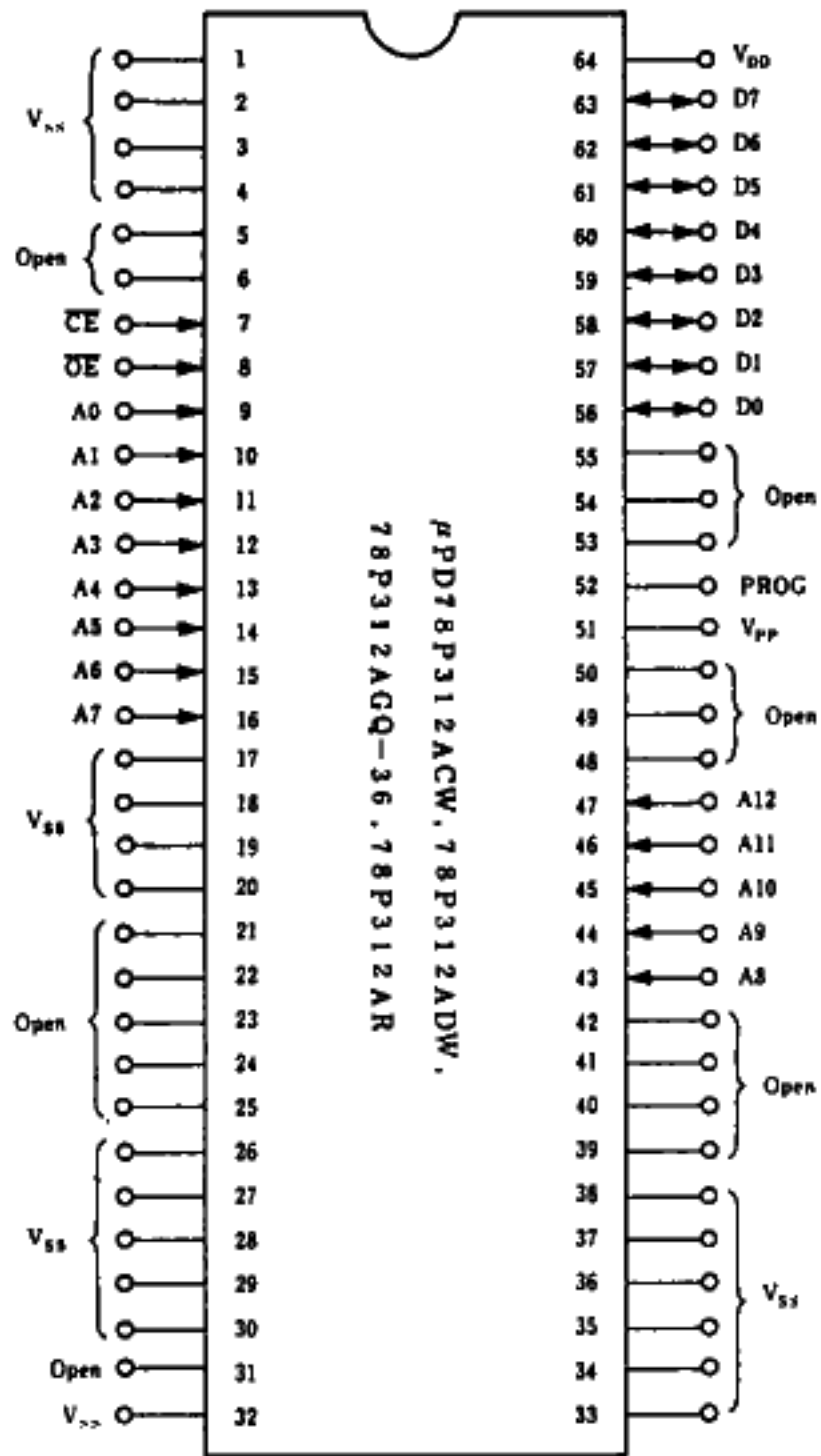
(1) 64-pin plastic shrink DIP/QUIP

64-pin ceramic shrink DIP/QUIP with window

(a) Normal operating mode



(b) PROM mode

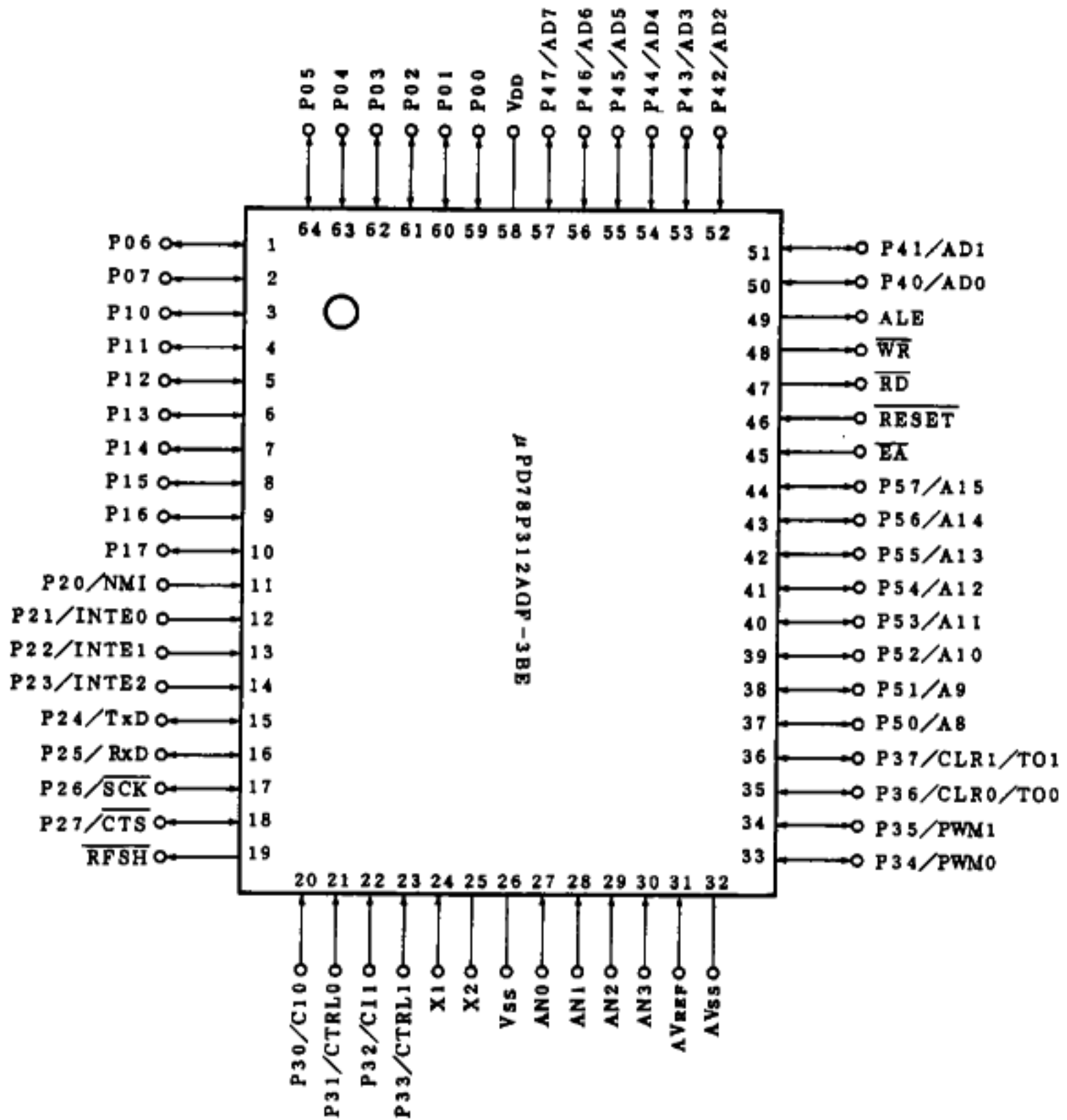


NOTE 1:  $V_{SS}$  : Connect the pin to ground.

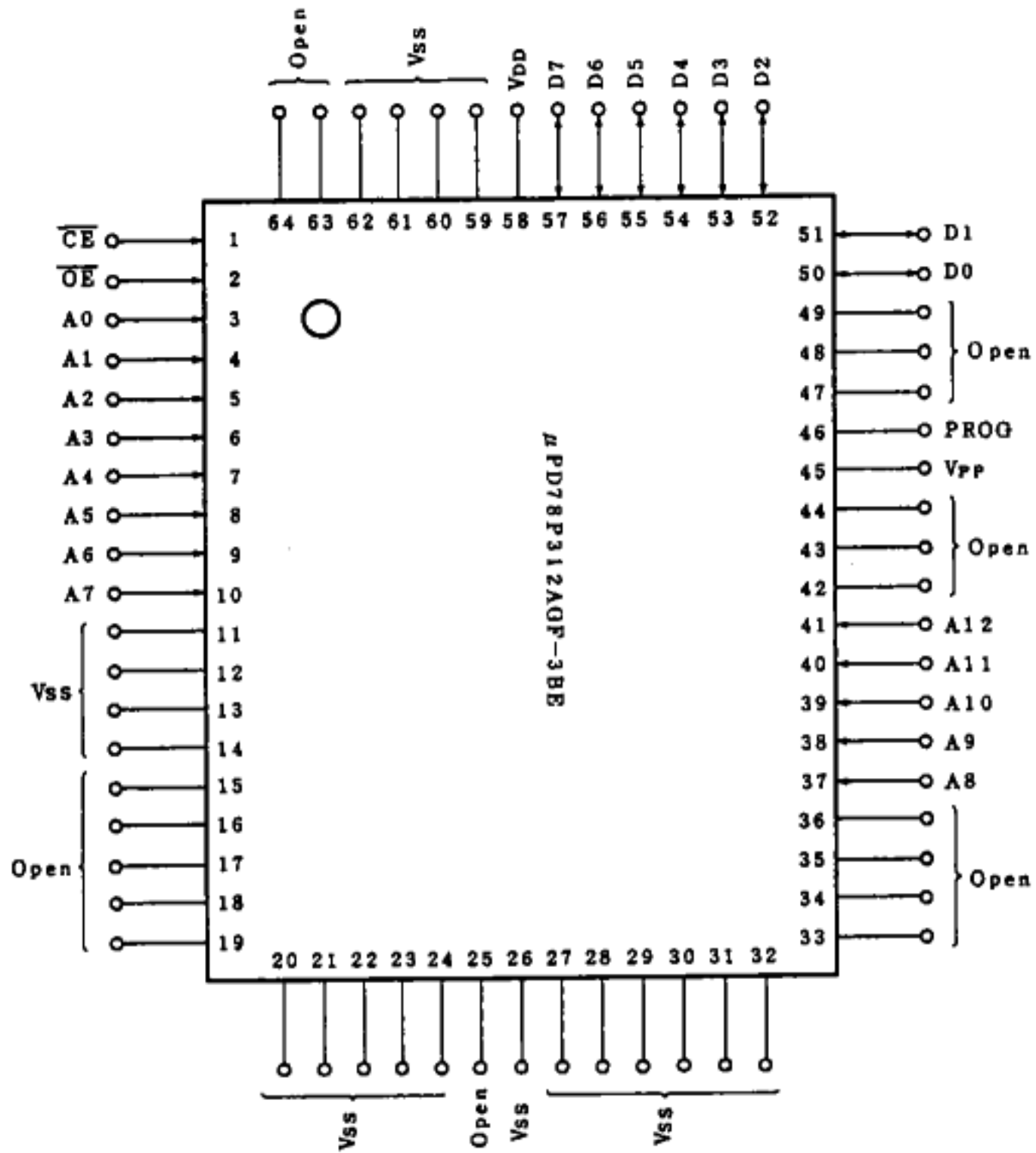
2: Open: Leave open.

(2) 64-pin plastic QFP (14 x 20 mm)

(a) Normal operating mode



(b) PROM mode

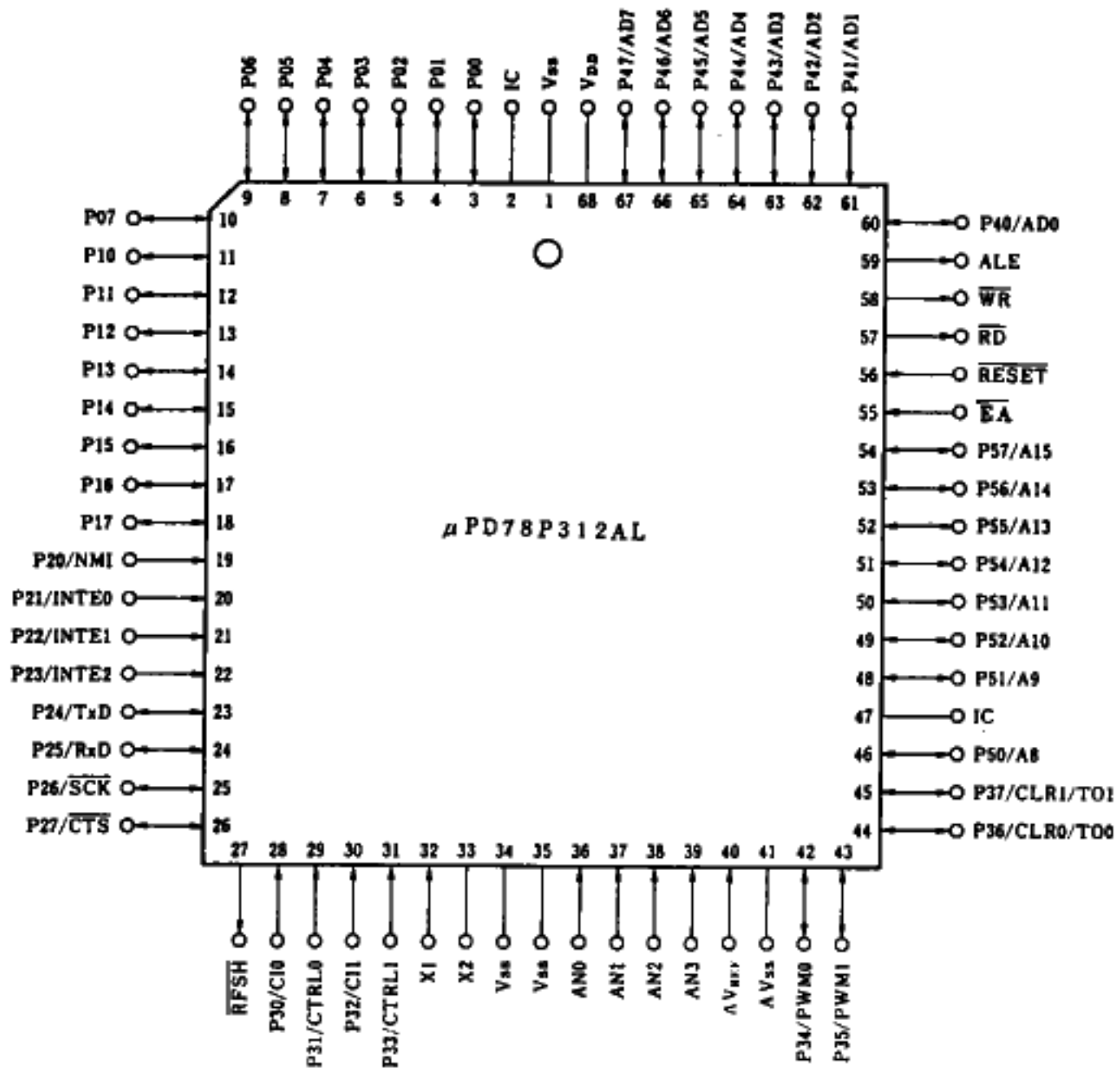


NOTE 1:  $V_{SS}$  : Connect the pin to ground.

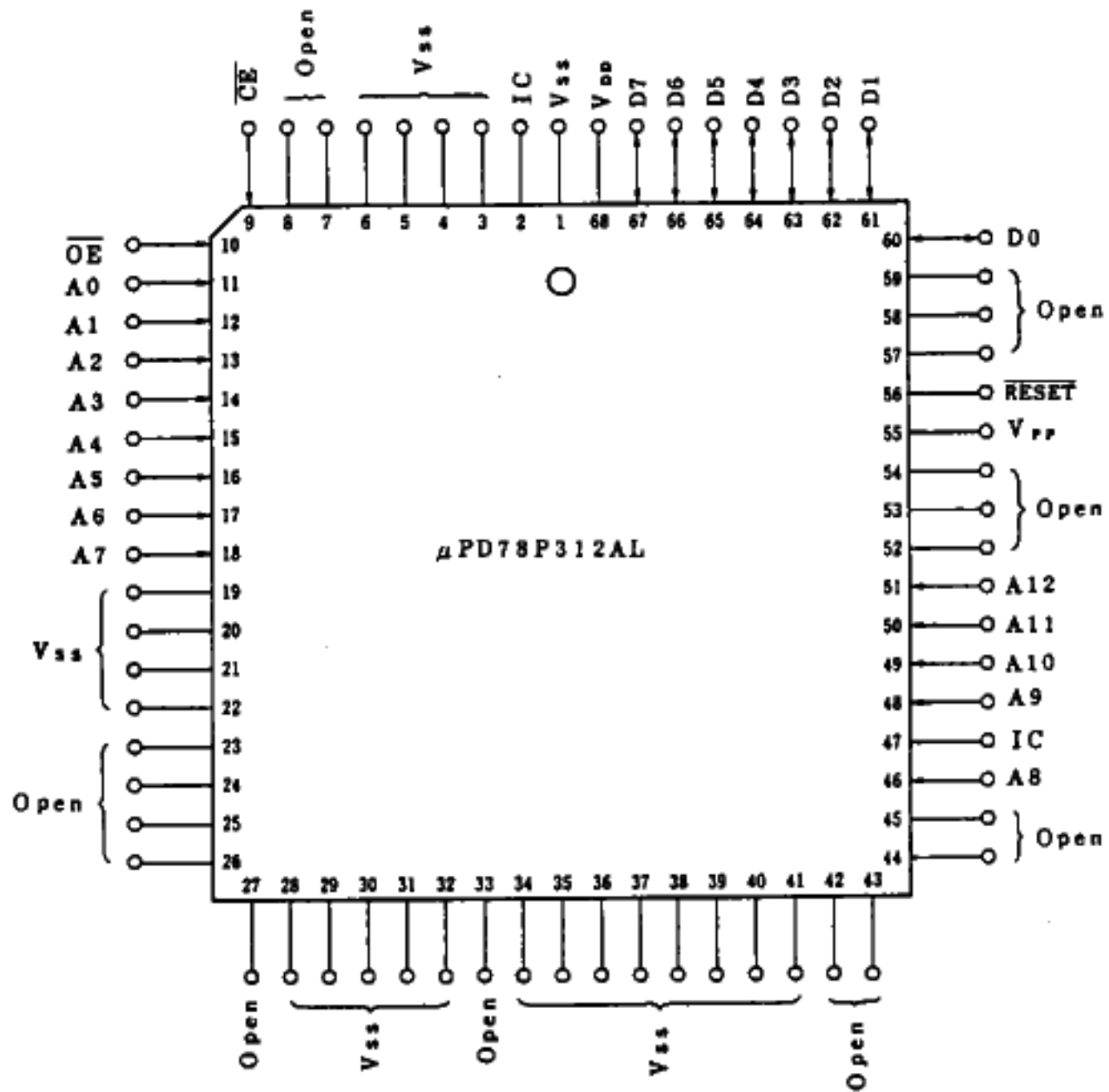
2: Open: Leave open.

(3) 68-pin plastic QFJ

(a) Normal operating mode



(b) PROM mode

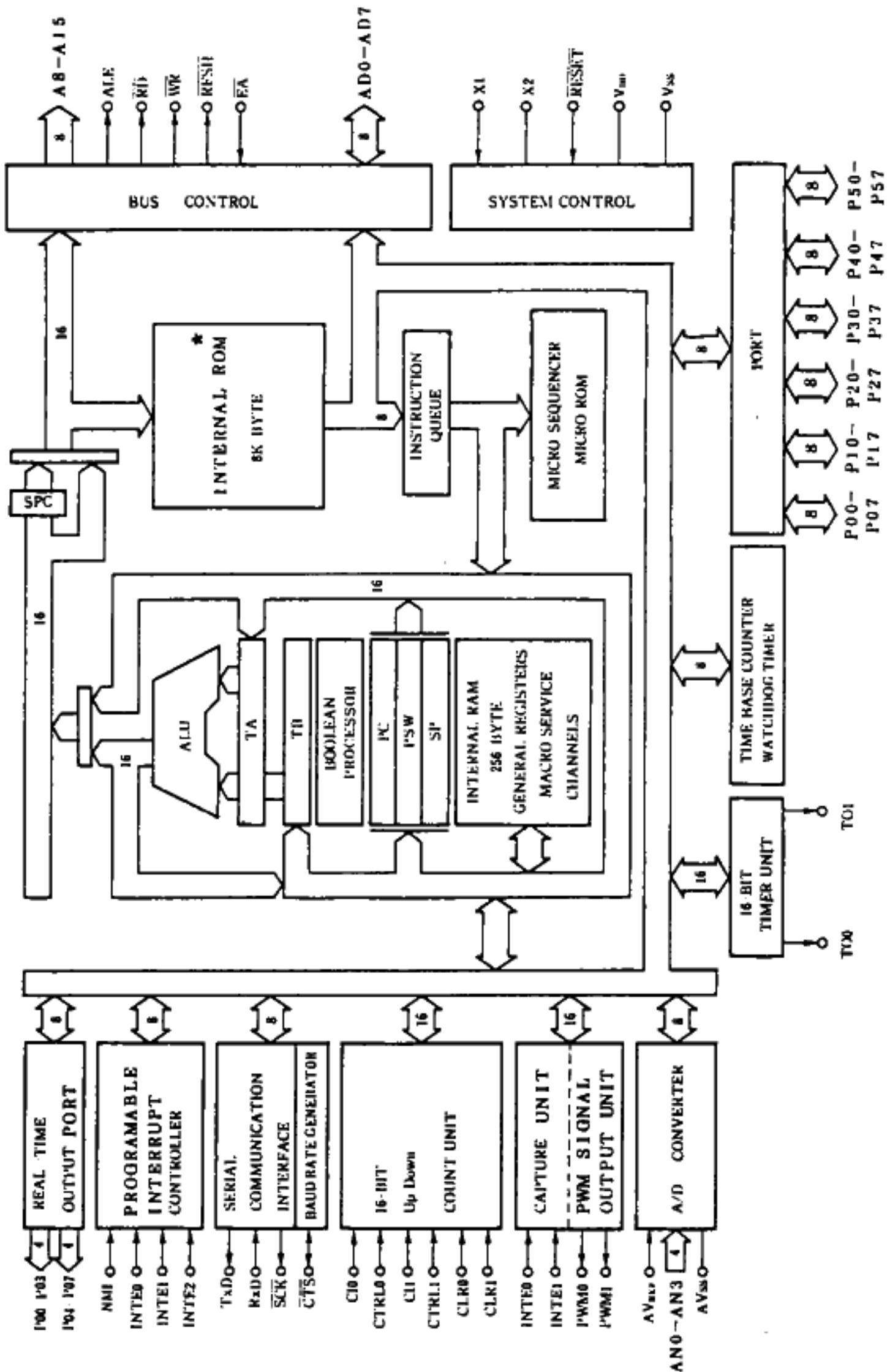


NOTE 1: V<sub>SS</sub> : Connect the pin to ground.

2: Open: Leave open.

1.4 BLOCK DIAGRAMS

1.4.1 uPD78312A, uPD78310A BLOCK DIAGRAM



\*: The uPD78310A does not contain internal ROM.





## 1.5 FUNCTION LIST

No. of basic instructions	96
Minimum instruction execution time	500 ns at 12 MHz
Internal memory	<ul style="list-style-type: none"> <li>. ROM : 8192 x 8 (uPD78312A only)</li> <li>. PROM: 8192 x 8 (uPD78P312A only)</li> <li>. RAM : 256 x 8</li> </ul>
Memory space	64 Kbytes
General registers	8-bit x 16 x 8 banks (memory mapping)
I/O lines	<ul style="list-style-type: none"> <li>. Input ports : 8</li> <li>. Input/output ports: 40 (uPD78312A, 78P312A), 24 (uPD78310A)</li> <li>. Output port : 1</li> <li>. Analog input : 4</li> </ul>
Multifunctional pulse input/output unit	<ul style="list-style-type: none"> <li>. 16-bit presetable up/down counter x 2</li> <li>. 16-bit free running counter capture function x 2</li> <li>. 16-bit interval timer x 2</li> <li>. High resolution PWM output x 2</li> <li>. 4-bit real-time output port x 2</li> </ul>
Serial communication interface	<ul style="list-style-type: none"> <li>. Eight bits (full duplex transmission/reception)</li> <li>. Contains a dedicated baud rate generator.</li> <li>. Two transfer modes (asynchronous mode and I/O interface mode)</li> </ul>
Analog-to-digital converter	<ul style="list-style-type: none"> <li>. 8-bit resolution (four analog inputs)</li> </ul>

(to be continued)

Interrupts	<ul style="list-style-type: none"><li>. 17 sources (four external and 13 internal sources)</li><li>. Eight priority levels can be programmed.</li><li>. Three processing modes (vectored interrupt, macro service, and context switching)</li></ul>
Standby	STOP or HALT mode
Main instructions	16-bit operation, multiplication and division, bit manipulation, BCD adjustment, user stack operation, and string
Miscellaneous	<ul style="list-style-type: none"><li>. Watchdog timer contained.</li><li>. 20-bit time base counter contained.</li><li>. Pseudo-static RAM refresh function contained.</li></ul>

## 1.6 DIFFERENCES AMONG FAMILY PRODUCTS

### 1.6.1 DIFFERENCES AMONG uPD78312A, uPD78310A, AND uPD78P312A

The uPD78312A, uPD78310A, and uPD78P312A are the same in function except for the differences listed in Table 1-1. When the packages are the same, they are pin-compatible with each other.

Table 1-1 Differences among uPD78312A, uPD78310A, and uPD78P312A

Item		uPD78312A uPD78312A(A)	uPD78310A uPD78310A(A)	uPD78P312A
Program memory		. Mask ROM . 8192 x 8 bits	. Not contained	. PROM . 8192 x 8 bits
Pin function	PROM mode	No	No	Yes
	Ports 4 and 5	Yes	No (always function as address bus and data bus)	Yes
	EA	Yes	Yes (be sure to use EA low.)	Yes
External memory access		External memory can be expanded by stages (256 bytes, 4 Kbytes, 16 Kbytes, and 56 Kbytes) by setting the memory expansion mode register (MM).	64-Kbyte external memory is always accessed regardless of how the memory expansion mode register (MM) is set.	Same as uPD78312A.
Pack-ages	Without window	. 64-pin plastic shrink DIP (750 mil) . 64-pin plastic QUIP . 64-pin plastic QFP (14 x 20 mm) . 64-pin plastic QFJ (□ 950 mil)		
	With window	None		. 64-pin ceramic DIP with window (750 mil) . 64-pin ceramic QUIP with window

### 1.6.2 DIFFERENCES BETWEEN uPD78312A AND uPD78312

The uPD78312A and uPD78312 are the same in function except for the differences listed in Table 1-2. They are pin-compatible with each other.

The differences between ROM-less uPD78310A and uPD78310 and between uPD78P312A and uPD78P312 which contain PROM are also as listed in Table 1-2.

Table 1-2 Differences between uPD78312A and uPD78312

Item	uPD78312A uPD78312A(A)	uPD78312 uPD78312(A)*
Count unit mode 4 (incremental/ decremental count by 2-phased inputs)	Yes	No
Count start by using interval timer external trigger	Yes	No
16-bit data transfer instructions between memory and register pair . MOVW rpl, !addr16 instruction . MOVW !addr16, rpl instruction	Yes	No

\*: Discontinuation product

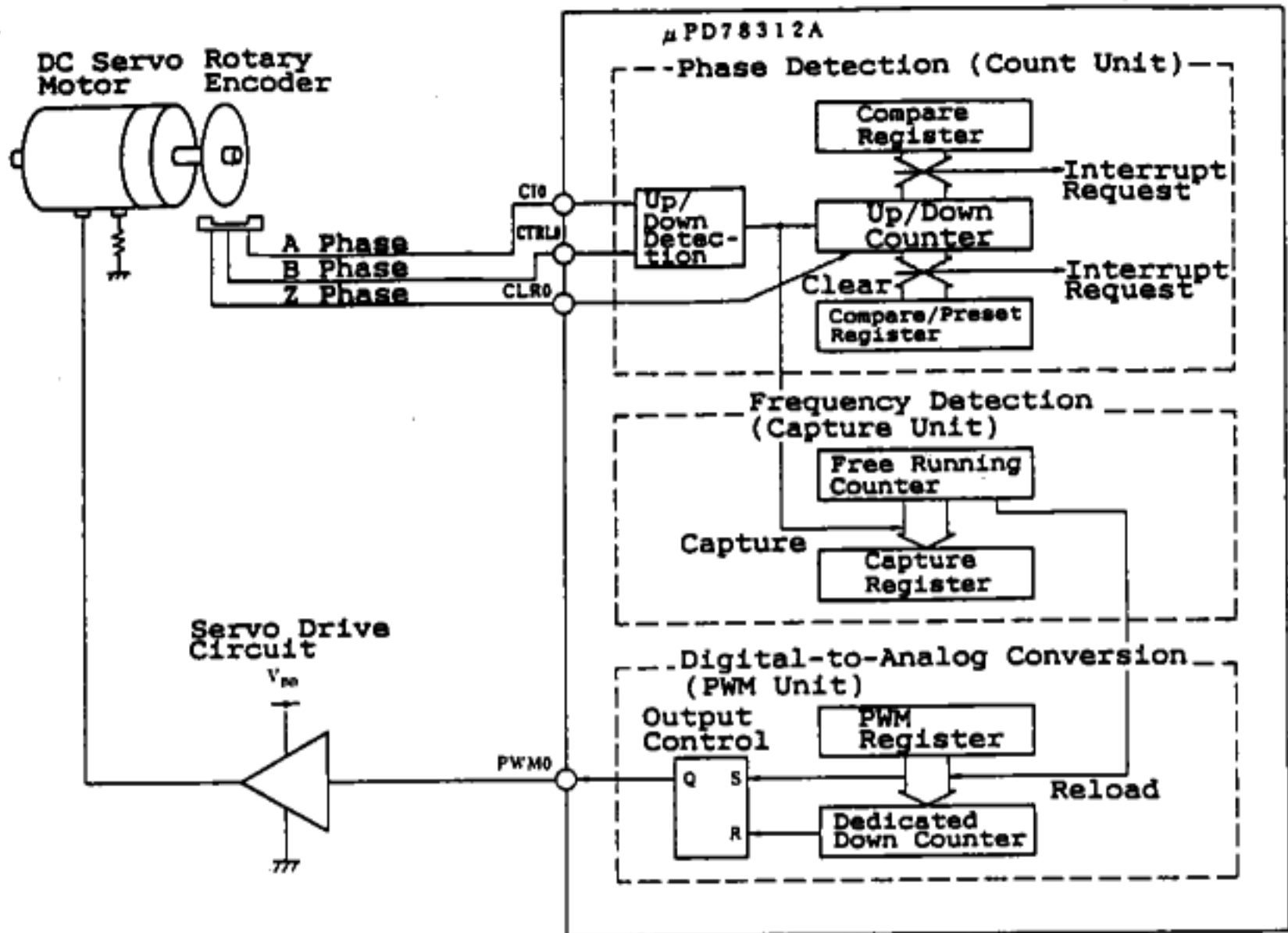
1.6.3 DIFFERENCES BETWEEN  $\mu$ PD78310A &  $\mu$ PD78312A AND  $\mu$ PD78310A(A) &  $\mu$ PD78312A(A)

Table 1-3 Differences between  $\mu$ PD78310A &  $\mu$ PD78312A and  $\mu$ PD78310A(A) &  $\mu$ PD78312A(A)

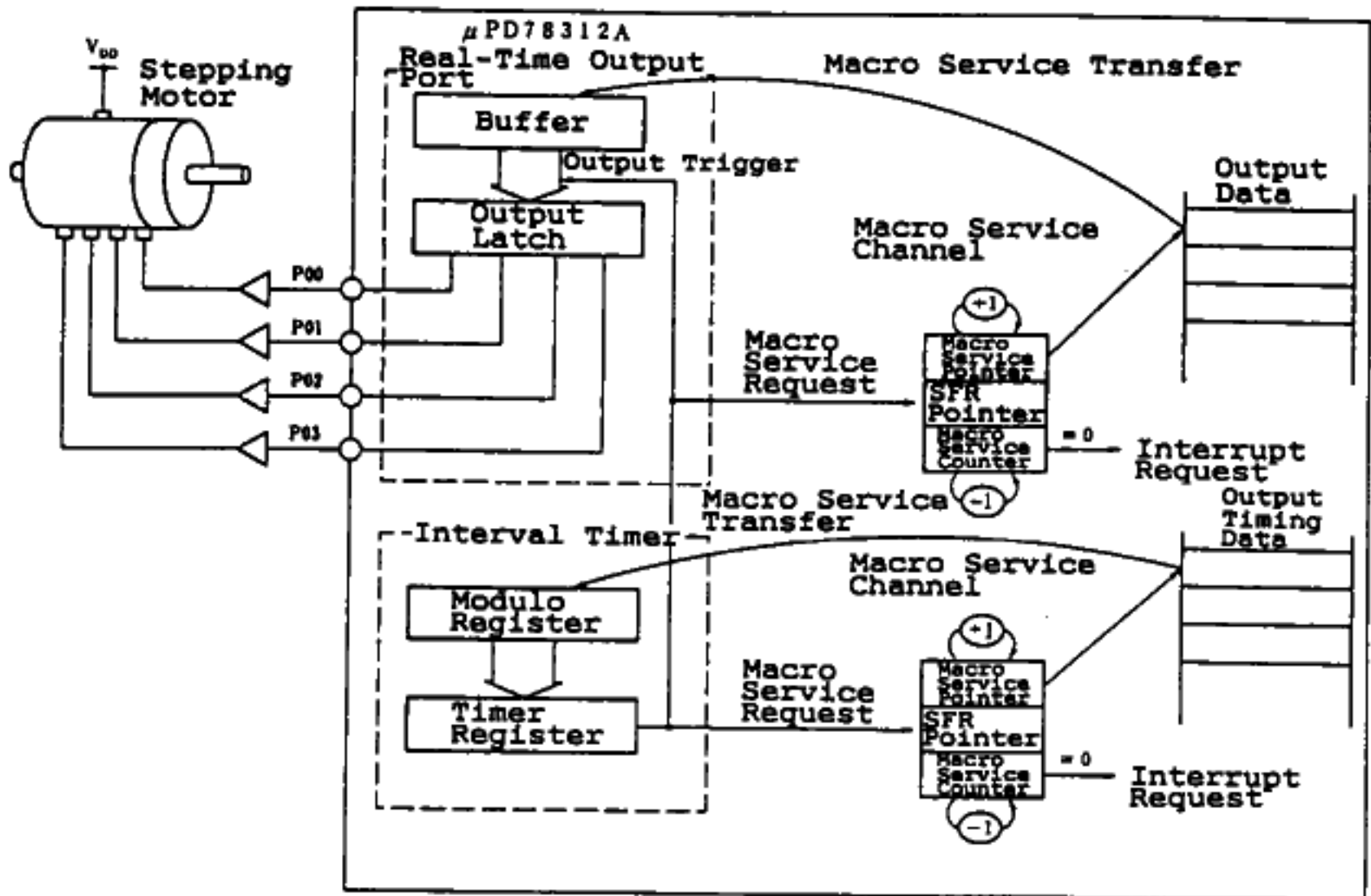
Item		Product	$\mu$ PD78310A & $\mu$ PD78312A	$\mu$ PD78310A(A) & $\mu$ PD78312A(A)
Quality grade			Standard	Special
Electrical specifications	Absolute maximum ratings		Different operating temperatures.	
	Recommended operating conditions		Different ambient temperatures ( $T_a$ ).	
	DC characteristics		Different data hold current.	
	A/D converter characteristics		Different analog input voltage.	

## 1.7 APPLICATION EXAMPLES

### (1) Closed loop control configuration example of DC servo motor



(2) Stepping motor control configuration example



## CHAPTER 2. PIN FUNCTION

The uPD78312A and uPD78310A operate in the normal operation mode of the pin function.

The uPD78P312A operates in the normal operating mode (uPD78312A mode) or PROM mode of the pin function. The mode is selected as listed in Table 2-1.

Table 2-1 uPD78P312A Operating Mode

$V_{DD}$	$V_{PP}$	PROG/ $\overline{\text{RESET}}$	Operating mode	
6 V	12.5 V	12.5 V	PROM mode	PROM write/verify mode
$V_{DD} = V_{PP} = 5 \text{ V}$				PROM read mode
		H/L	Normal operating mode	

Remarks: H = High, L = Low



## 2.1 NORMAL OPERATING MODE

### 2.1.1 P00 TO P07 (PORT 0) ... 3-STATE INPUT/OUTPUT

P00 to P07 are 8-bit input/output pins of port 0 (8-bit input/output port with output latch). The input or output mode can be selected bit-wise by setting port 0 mode register (PM0).

Port 0 can also serve as a real-time output port which outputs the buffer register contents at programmable intervals. (See 4.3.5.)

When  $\overline{\text{RESET}}$  is input, port 0 becomes an input port (output high impedance) and the output latch contents become undefined.

### 2.1.2 P10 TO P17 (PORT 1) ... 3-STATE INPUT/OUTPUT

P10 to P17 are 8-bit input/output pins of port 1 (8-bit input/output port with output latch). The input or output mode can be selected bit-wise by setting the port 1 mode register (PM1).

When  $\overline{\text{RESET}}$  is input, port 1 becomes an input port (output high impedance) and the output latch contents become undefined.

### 2.1.3 P20 TO P27 (PORT 2) ... INPUT, 3-STATE INPUT/OUTPUT

P20 to P27 are 8-bit input/output pins of port 2 (P20 to P23 are input port pins and P24 to P27 are 4-bit input/output port pins with output latch).

In addition to the input/output port function, the control pin function is provided.

The P20 to P23 pins are fixed to the control signal mode. The pin level can always be read or tested regardless of the dual function pin operation.

The operating mode can be selected bit-wise for the P24 to P27 pins by setting the port 2 mode control register (PMC2), as listed in Table 2-2.

Table 2-2 P20 to P27 Operation

Pin \ Mode	PMC2n = 0	PMC2n = 1
	Port Mode	Control Signal Mode
P20	-	NMI input
P21	-	INTE0 input
P22	-	INTE1 input
P23	-	INTE2 input
P24	Input/output port	TxD output
P25	Input/output port	RxD input
P26	Input/output port	$\overline{\text{SCK}}$ input/output
P27	Input/output port	$\overline{\text{CTS}}$ input/output

Remarks: PMC2n is PMC2 register bit n.

#### (1) Port mode

When the port mode is selected for the P24 to P27 pins by setting the PMC2 register, the input or output mode can be selected bit-wise by setting the port 2 mode register (PM2).

Although the P20 to P23 pins are fixed to the control signal mode, the pin level can be read or tested.

(2) Control signal mode

The P24 to P27 pins can be set to the control pins bit-wise by setting the port 2 mode control register (PMC2). The P20 to P23 pins are fixed to the control signal mode.

(a) NMI

NMI is an external nonmaskable interrupt request input pin. The rising or falling edge can be selected for NMI detection edge by setting the external interrupt mode register (INTM).

(b) INTE0 to INTE2

INTE0 to INTE2 are external interrupt request input pins. The detection edge can be selected by using the external interrupt mode register (INTM) as follows:

- . INTE0, INTE1: Falling edge or both rising and falling edges
- . INTE2 : Rising or falling edge

(c) TxD

TxD is a serial data output pin.

(d) RxD

RxD is a serial data input pin.

(e)  $\overline{SCK}$

$\overline{SCK}$  is a serial transmit clock output pin. It is effective only when the serial interface is in the I/O interface mode.

(f)  $\overline{CTS}$

The  $\overline{CTS}$  function varies depending on the serial interface operating mode.

- . In the asynchronous mode,  $\overline{CTS}$  is a clear-to-send control input pin.
- . In the I/O interface mode,  $\overline{CTS}$  is a serial receive clock input/output pin.

#### 2.1.4 P30 TO P37 (PORT 3) ... INPUT, 3-STATE INPUT/OUTPUT

P30 to P37 are 8-bit input/output pins of port 3 (P30 to P33 are input port pins and P34 to P37 are 4-bit input/output port pins with output latch).

In addition to the input/output port function, the control signal pin function is provided.

The P30 to P33 pins are fixed to the control signal mode. The pin level can always be read or tested regardless of dual function pin operation.

The operating mode can be selected bit-wise for the P34 to P37 pins by setting the port 3 mode control register (PMC3), as listed in Table 2-3.

The P36 and P37 pins function as counter clear pins by setting up/down counter control register (UDCC0, UDCC1) bit 2 to (1) (see Figure 4-13); the pins function as capture trigger input pins by setting capture/compare register (CRC) bits 0 and 4 to (1) (see Figure 4-14).

When RESET is input, port 3 becomes an input port (output high impedance) and the output latch contents become undefined.

Table 2-3 P30 to P37 Operation

Pin \ Mode	PMC3n = 0	PMC3n = 1
	Port Mode	Control Signal Mode
P30	-	CIO input
P31	-	CTRL0 input
P32	-	CI1 input
P33	-	CTRL1 input
P34	Input/output port	PMW0 output
P35	Input/output port	PMW1 output
P36	Input/output port	T00 output
P37	Input/output port	T01 output

Remarks: The P36 and P37 pins can also be set to the CLR0 and CLR1 pins.

(1) Port mode

When the port mode is selected for the P34 to P37 pins by setting the PMC3 register, the input or output mode can be selected bit-wise by setting port 3 mode register (PM3).

Although the P30 to P33 pins are fixed to the control signal mode, the pin level can be read or tested.

(2) Control signal mode

The P34 to P37 pins can be set to the control pins bit-wise by setting the port 3 mode control register (PMC3). The P30 to P33 pins are fixed to the control signal mode.

(a) CIO and CI1

CIO and CI1 are external count clock input pins to the count unit.

(b) CTRL0 and CTRL1

CTRL0 and CTRL1 are count operation (up/down) change control signal input pins to the count unit.

(c) PWM0 and PWM1

PWM0 and PWM1 are PWM pulse output pins from the PWM output unit.

(d) T00 and T01

T00 and T01 are programmable timer output pins from the timer unit.

(3) Counter clear/capture trigger operation

The P36 and P37 pins can be set to the CLR0 and CLR1 pins; up/down counter (UDC0, UDC1) clear operation can be specified by setting UDCC0 and UDCC1 register bit 2 to (1) and capture register (CR00, CR10) capture trigger operation can be specified by setting CRC register bits 0 and 4 to (1). (See Figures 4-13 and 4-14).

To perform clear/capture trigger operation by using external input, be sure to select the input port mode for the P36 and P37 pins by the PM3 and PMC3 registers.

## 2.1.5 P40 TO P47 (PORT 4) ... 3-STATE INPUT/OUTPUT

### (1) uPD78312A, uPD78P312A

P40 to P47 are 8-bit input/output pins of port 4 (8-bit input/output port with output latch). In addition to the input/output port function, the multiplexed address output/data input/output pin (multiplexed address/data bus) function is provided to access external expansion memory.

When  $\overline{\text{RESET}}$  is input, port 4 becomes an input port (output high impedance) and the output latch contents become undefined.

Port 4 can be set to either of the following modes by setting the memory expansion mode register (MM):

#### (a) Port mode

The P40 to P47 pins serve as port 4 input/output pins. The input or output mode can be selected in 8-bit units by setting the memory expansion mode register (MM).

#### (b) Expansion mode

When external memory is added to internal memory, the P40 to P47 pins function as multiplexed address data bus (AD0 to AD7).

### (2) uPD78310A

The P40 to P47 pins always function as multiplexed address/data bus and do not provide the port 4 function.

## 2.1.6 P50 TO P57 (PORT 5) ... 3-STATE INPUT/OUTPUT

### (1) uPD78312A, uPD78P312A

P50 to P57 are 8-bit input/output pins of port 5 (8-bit input/output port with output latch). In addition to the input/output port function, the address output pin (address bus) function is provided to access external expansion memory.

When  $\overline{\text{RESET}}$  is input, port 5 becomes an input port (output high impedance) and the output latch contents become undefined.

Port 5 can be set to either of the following modes by setting the memory expansion mode register (MM):

#### (a) Port mode

The P50 to P57 pins serve as port 5 input/output pins. The input or output mode can be selected bit-wise by setting port 5 mode register (PM5).

#### (b) Expansion mode

When external memory is added to internal memory, the P50 to P57 pins can be set to address output (A8 to A15) pins by stages according to the external memory expansion size.

### (2) uPD78310A

The P50 to P57 pins always function as address bus and do not provide the port 5 function.

## 2.1.7 $\overline{\text{WR}}$ (WRITE STROBE)

$\overline{\text{WR}}$  is an active low strobe signal output for external memory write operation. It goes high except in external memory data write machine cycles. When reset,  $\overline{\text{WR}}$  also goes high.

## 2.1.8 $\overline{\text{RD}}$ (READ STROBE) ... OUTPUT

$\overline{\text{RD}}$  is an active low strobe signal output for external memory read operation. It goes high except in external memory data read machine cycles. When reset,  $\overline{\text{RD}}$  also goes high.

## 2.1.9 ALE (ADDRESS LATCH ENABLE) ... OUTPUT

ALE is a strobe signal to externally latch low-order 8-bit address signals (P40 to P47 output) output when external memory is accessed. It is activated except when internal ROM is accessed.

#### 2.1.10 $\overline{EA}$ (EXTERNAL ACCESS)

##### (1) uPD78312A, 78P312A

$\overline{EA}$  functions as a ROM-less mode specification pin to access external memory instead of internal ROM as program memory. When a high pulse is input, internal ROM is accessed; when a low pulse is input, external memory is accessed. Normally, fix  $\overline{EA}$  high.

##### (2) uPD78310A

Be sure to fix  $\overline{EA}$  low.

#### 2.1.11 AN0 TO AN3 (ANALOG INPUT)

AN0 to AN3 are four analog signal input pins to the analog-to-digital converter.

#### 2.1.12 $AV_{REF}$ (REFERENCE VOLTAGE)

$AV_{REF}$  is an analog-to-digital converter reference voltage input and analog-to-digital converter power pin.

#### 2.1.13 $AV_{SS}$ (ANALOG $V_{SS}$ )

$AV_{SS}$  is an analog-to-digital converter ground pin.

#### 2.1.14 X1 AND X2 (CRYSTAL)

X1 and X2 are internal clock oscillation crystal connection pins. To supply external clock, input it to X1 and its inverted phase to X2.

#### 2.1.15 $\overline{RFSH}$ (REFRESH) ... OUTPUT

$\overline{RFSH}$  is an output pin of refresh pulses to external pseudo-static memory when external pseudo-static memory is connected.

If  $\overline{RFSH}$  is not used as the refresh pulse output pin, it can be used as a 1-bit output port.

#### 2.1.16 $\overline{RESET}$ (RESET) ... INPUT

$\overline{RESET}$  is a reset input pin.

#### 2.1.17 $V_{DD}$

$V_{DD}$  is a positive power supply pin.



#### 2.1.18 V<sub>SS</sub>

V<sub>SS</sub> is a ground potential pin.

#### 2.1.19 IC

IC is an internally connected pin. Leave the pin unconnected.

## 2.2 PROM MODE

The PROM mode can be selected only for the uPD78P312A.

### 2.2.1 A0 to A12 (ADDRESS) ... INPUT

A0 to A12 are 13-bit address input pins when PROM is written and verified.

The A0 to A7 pins are also used for the P10 to P17 pins. The A8 to A12 pins are also used for the P50 to P54 pins.

### 2.2.2 D0 to D7 (DATA) ... INPUT/OUTPUT

D0 to D7 are 8-bit data input/output pins when PROM is written and verified.

The D0 to D7 pins are also used for the P40 to P47 pins.

### 2.2.3 $\overline{CE}$ (CHIP ENABLE) ... INPUT

$\overline{CE}$  is a program pulse input pin.

The pin is also used for the P06 pin.

### 2.2.4 $\overline{OE}$ (OUTPUT ENABLE) ... INPUT

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

The pin is also used for the P07 pin.

### 2.2.5 PROG (PROGRAM)

PROG is a high-voltage apply pin when the PROM mode is set.

The pin is also used for the  $\overline{RESET}$  pin.

### 2.2.6 $V_{PP}$ (PROM POWER SUPPLY)

$V_{PP}$  is a high-voltage apply pin when PROM is written and verified.

During the normal operating mode, be sure to connect the pin to  $V_{DD}$ .

### 2.2.7 $V_{DD}$ (POWER SUPPLY)

$V_{DD}$  is a positive power supply pin. During the PROM mode, apply 6 V to the pin.

### 2.2.8 $V_{SS}$ (GROUND)

$V_{SS}$  is a GND potential pin.

2.2.9 IC (INTERNALLY CONNECTED)

Leave the IC pin unconnected.

### 2.3 PIN INPUT/OUTPUT CIRCUITS

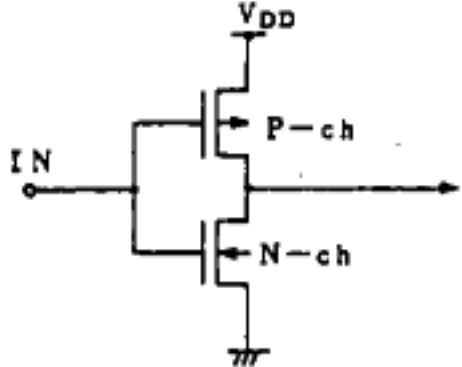
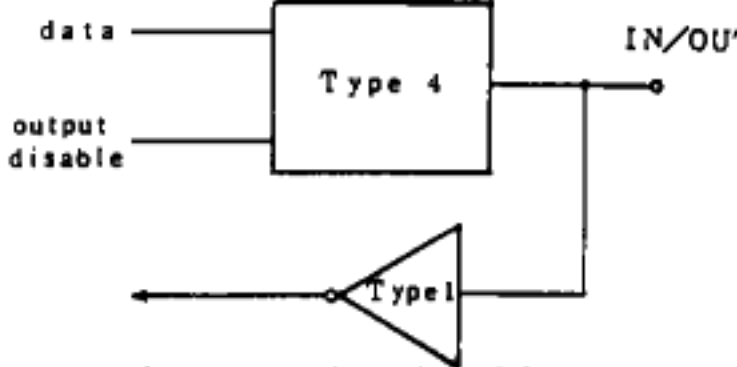
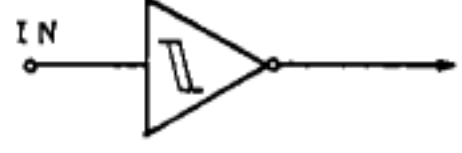
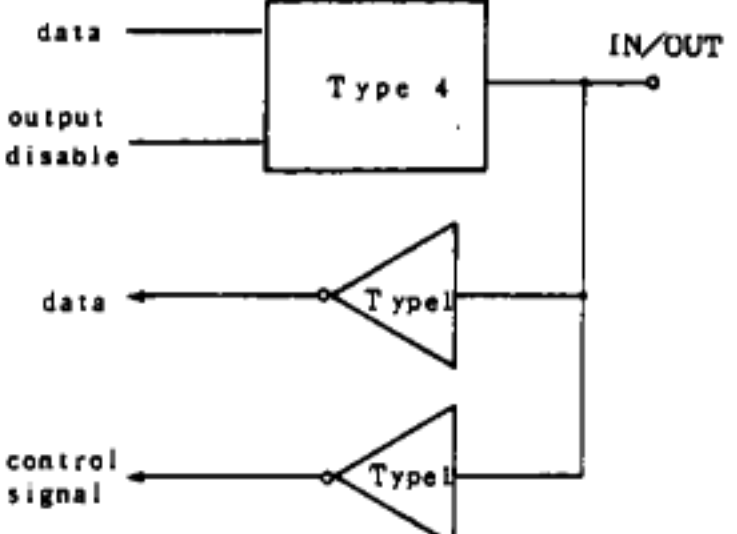
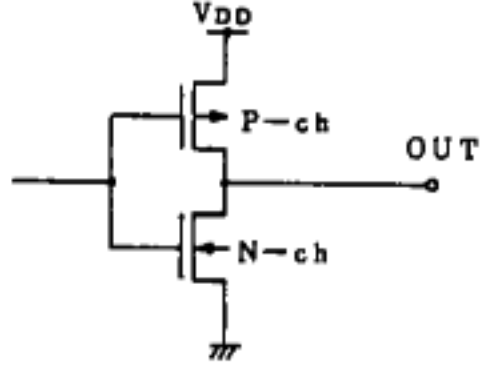
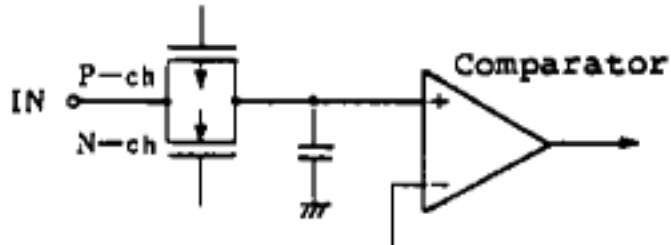
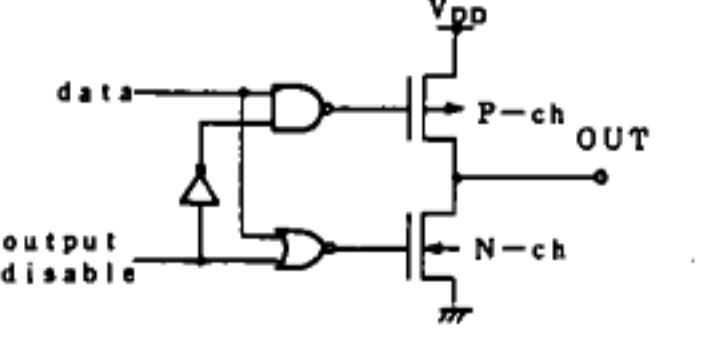
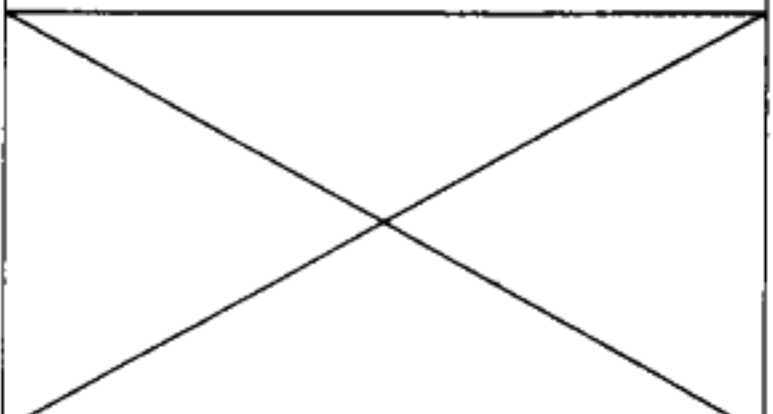
Table 2-4 lists the pin input/output circuit types. Figure 2-1 shows the pin input/output circuit types schematically.

The pin names when the normal operating mode is set are used.

Table 2-4 Pin Input/Output Circuit Types

Pin	I/O Circuit Type	Pin	I/O Circuit Type
P00 to P07	5	P34/PWM0	5
P10 to P17	5	P35/PWM1	
P20/NMI	2	P36/TO0/CLR0	6
P21/INTE0	1	P37/TO1/CLR1	
P22/INTE1		P40 to P47/AD0 to AD7	5
P23/INTE2		P50 to P57/A8 to A15	5
P24/TxD	5	$\overline{WR}$	3
P25/RxD		$\overline{RD}$	
P26/ $\overline{SCK}$		ALE	
P27/ $\overline{CTS}$		$\overline{EA}$	1
P30/CIO	1	AN0 to AN3	7
P31/CTRL0		$\overline{RFSH}$	3
P32/CI1		$\overline{RESET}$	2
P33/CTRL1			

Figure 2-1 Pin Input/Output Circuit List

<p><b>TYPE 1</b></p> 	<p><b>TYPE 5</b></p>  <p>Input/output circuit which consists of Type 4 push-pull output and Type 1 input buffer</p>
<p><b>TYPE 2</b></p>  <p>Schmitt-triggered input having hysteresis characteristic</p>	<p><b>TYPE 6</b></p>  <p>Input/output circuit which consists of Type 4 push-pull output and Type 1 input buffer</p>
<p><b>TYPE 3</b></p> 	<p><b>TYPE 7</b></p> 
<p><b>TYPE 4</b></p>  <p>Push-pull output where output can be placed in high impedance (both P-ch and N-ch are off)</p>	

## 2.4 RECOMMENDED CONNECTION OF UNUSED PINS

Table 2-5 Recommended Connection of Unused Pins

Pin	Recommended Connection
P00 to P07, P10 to P17	Input state : Connect the pins to $V_{DD}$ with pull-up resistor.  Output state: No connection required
P20 to P23	Connect the pins to $V_{SS}$ .
P30 to P33	Connect the pins to $V_{DD}$ or $V_{SS}$ .
P24 to P27, P34 to P37, P40 to P47, P50 to P57	Input state : Connect the pins to $V_{DD}$ with pull-up resistor.  Output state: No connection required
$\overline{WR}$ , $\overline{RD}$ , ALE, $\overline{RFSH}$	No connection required
AN0 to AN3	Connect the pins to $V_{DD}$ or $V_{SS}$ .
$AV_{REF}$ , $AV_{SS}$	Connect the pins to $V_{SS}$ .

## CHAPTER 3. CPU ARCHITECTURE

### 3.1 MEMORY SPACE

The uPD78312A enables the user to address memory of a maximum of 64 Kbytes. (See Figure 3-1.)

Programs can be fetched from the 0000H to FEFFH area of the 64-Kbyte address space. The 256-byte area of FF00H to FFFFH is reserved as a special function register area.

#### (1) Vector table area

Peripheral hardware interrupt requests, reset input, external interrupt requests, and break instruction interrupt branch addresses are stored in the 64-byte area of 0000H to 003FH.

When an interrupt request occurs, the vector table contents (2 words x 8 bits) are set in the program counter (PC) for branch; the even address contents of the vector table are set in the low-order eight bits of the PC and the odd address contents are set in the high-order eight bits of the PC.

When CPU control word (CCW) bit 1 (TPF) is set to (1), 8000H to 803FH of the external memory area can be used instead of 0000H to 003FH as interrupt vector table.

Table 3-1 Vector Table List

Interrupt Request Source	Interrupt Request Flag	Vector Table Address
Reset input (RESET)	-	0000H
NMI pin input (NMI)	-	0002H
INTE0 pin input (INTE0)	EXIF0	0004H
INTE1 pin input (INTE1)	EXIF1	0006H
INTE2 pin input (INTE2)	EXIF2	0008H
Watchdog timer (WDT)	-	000AH
Time base counter	TBF	000CH
Timer unit	TMF0	000EH

(to be continued)

Table 3-1 Vector Table List (cont'd)

Interrupt Request Source	Interrupt Request Flag	Vector Table Address
Timer unit	TMF1	0010H
Timer unit	TMF2	0012H
Count unit	CRF00	001AH
Count unit	CRF01	001CH
Count unit	CRF10	001EH
Count unit	CRF11	0020H
Serial reception error	SEF	0022H
Serial reception completion	SRF	0024H
Serial transmission completion	STF	0026H
Analog-to-digital converter	ADF	0028H
Break instruction	-	003EH

(2) CALLT instruction table area

1-byte call instruction (CALLT) call addresses can be stored in the 64-byte area of 0040H to 007FH.

When CPU control word (CCW) bit 1 (TPF) is set to (1), 8040H to 807FH of the external memory area can be used instead of 0040H to 007FH as CALLT instruction table.

(3) CALLF instruction entry area

The area of 0800H to 0FFFH can be directly addressed in a 2-byte call instruction (CALLF).

(4) Internal RAM area

256-byte RAM is mapped in FE00H to FEFFH. General registers of eight banks are mapped in the 128-byte FE80H to FEFFH are of the internal RAM area.



(5) Special function register area

Special function registers such as the mode registers and control registers of on-chip peripheral hardware are mapped in the FF00H to FFFFH area. Addresses in which no registers are mapped cannot be accessed.

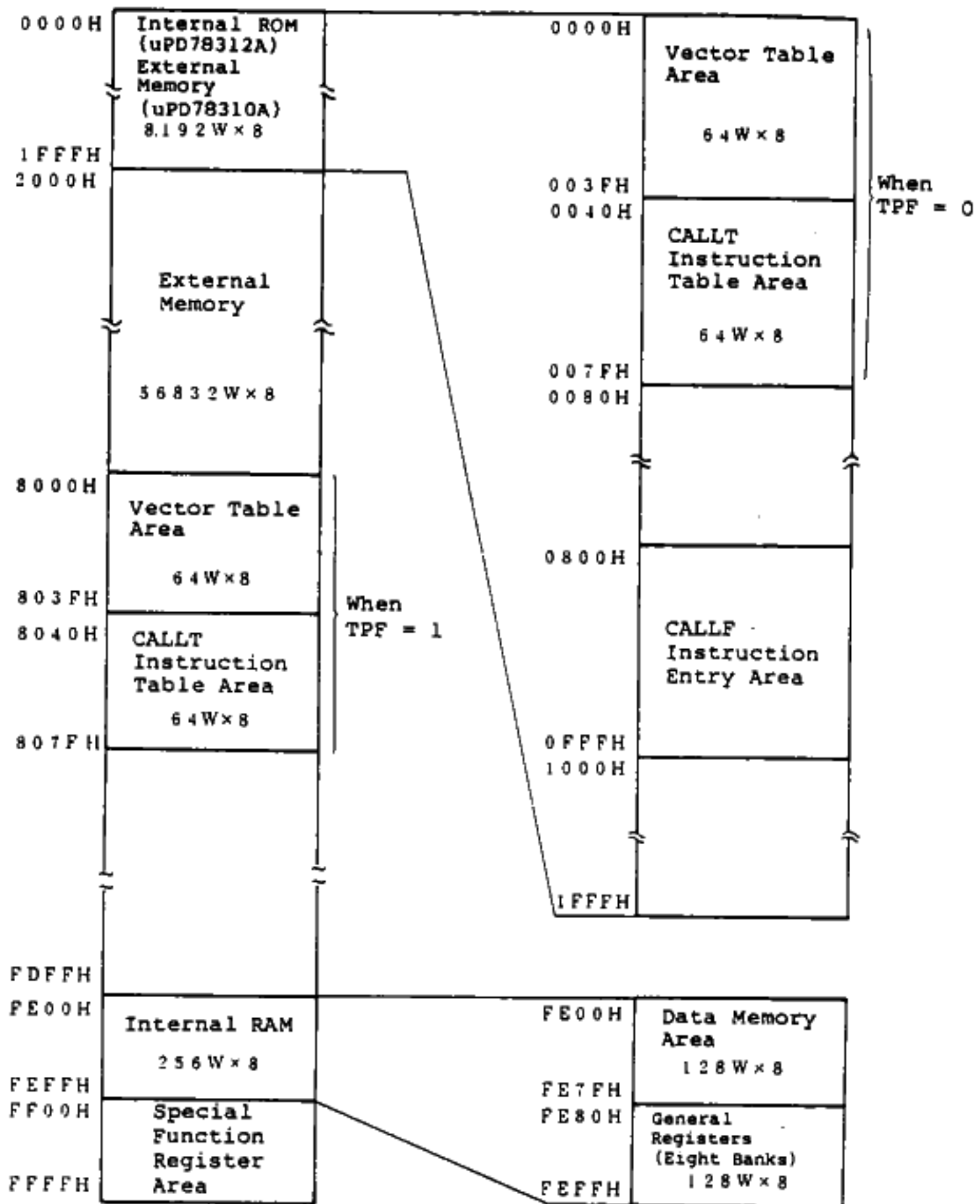
(6) External memory area

When the uPD78312A or uPD78P312A is used, external memory (ROM, RAM) can be expanded by stages in the 56-Kbyte area of 2000H to FDFFH.

When the uPD78310A is used, external memory (ROM, RAM) can be connected to the 64-Kbyte area of 0000H to FDFFH.

The external memory is accessed by using P47 to P40 (multiplexed address/data bus), P57 to P50 (address bus), and the  $\overline{RD}$ ,  $\overline{WR}$ , and ALE signals. Pseudo-static memory refresh pulse output port ( $\overline{RFSH}$ ) is provided and pseudo-static memory equivalent to uPD428128 can be easily connected.

Figure 3-1 Memory Map



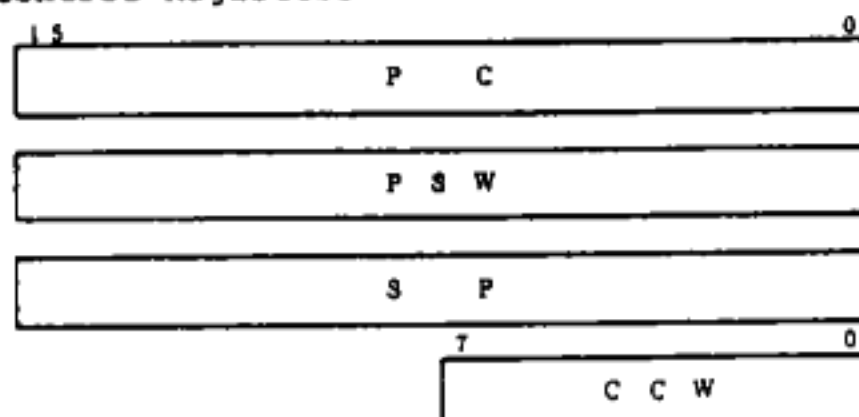
Remarks: TPF is CPU control word (CCW) bit 1.

### 3.2 PROCESSOR REGISTERS

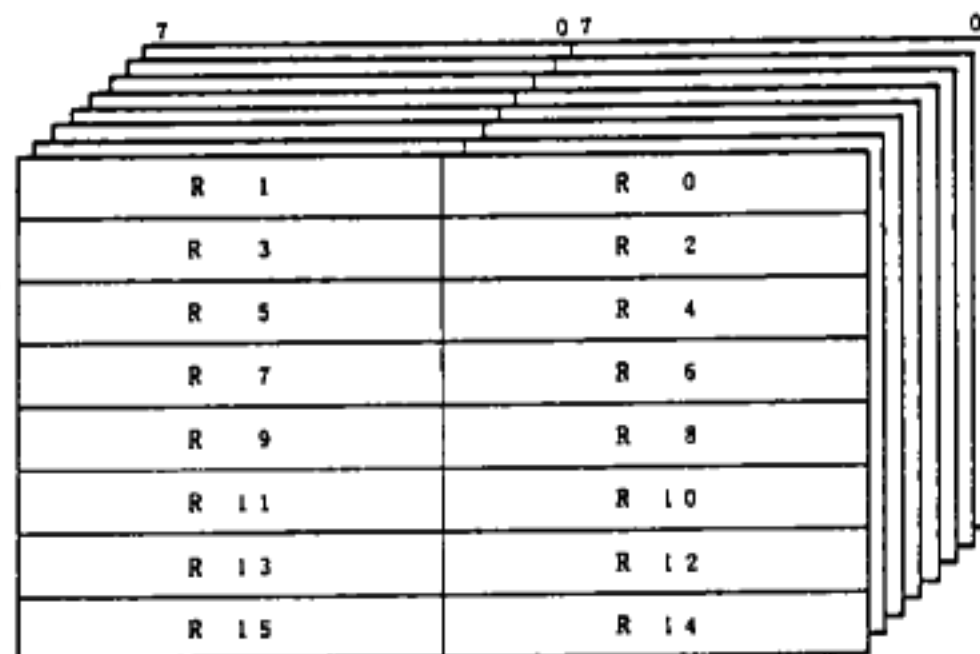
The main registers are eight banks each consisting of 16 8-bit general registers, control registers of one 8-bit register and three 16-bit registers, and special function registers such as peripheral hardware I/O mode registers.

Figure 3-2 Register Configuration

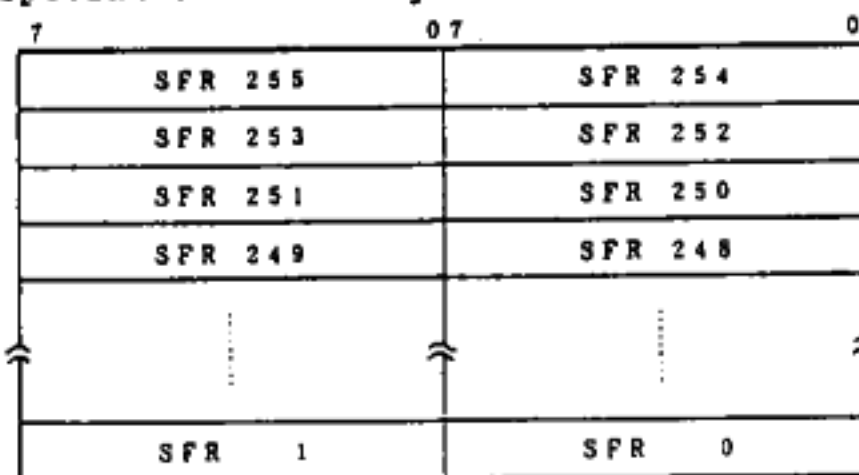
#### Control Registers



#### General Registers



#### Special Function Registers



Remarks: The control register CCW (CPU control word) is mapped in the special function register (SFR) area.

### 3.2.1 CONTROL REGISTERS

The control registers are three 16-bit registers and an 8-bit register which have dedicated functions such as program sequence, status, and stack memory control and operand address modification.

#### (1) Program counter (PC)

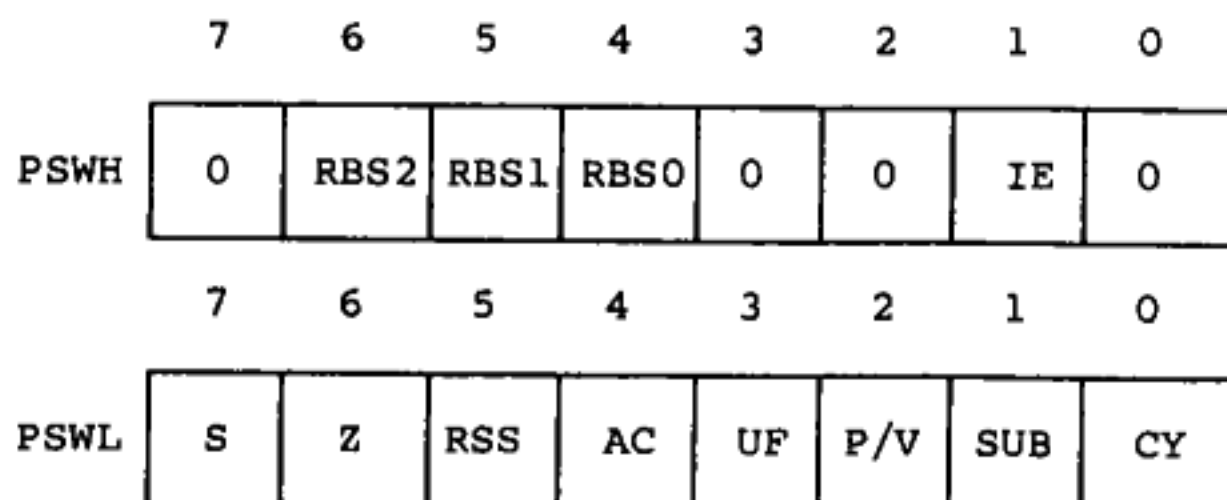
The program counter (PC) is a 16-bit register which retains address information of the next program to be executed. Normally, it is automatically incremented according to the number of the bytes of the fetched instruction. When an instruction involving a branch is executed, immediate data or the register contents are set. When RESET is input, the reset vector data at 00H and 01H is set in the PC for a branch.

#### (2) Program status word (PSW)

The program status word (PSW) is a 16-bit register which consists of flags set or reset according to the instruction execution result. It is read/written in 8-bit units (high-order eight bits (PSWH), low-order eight bits (PSWL)). Each flag can also be manipulated by executing a 1-bit manipulation instruction. When an interrupt request occurs or the BRK instruction is executed, automatically the PSW contents are saved in a stack, and restored by executing the RETI instruction.

When RESET is input, all the bits are reset to 0.

Figure 3-3 PSW Format



(a) Subtraction flag (SUB)

When the arithmetic and logic unit (ALU) performs subtraction operation, the subtraction flag (SUB) is set to (1); else, the flag is reset to (0). It becomes effective when a decimal adjustment instruction is executed after BCD subtraction is made.

(b) Carry flag (CY)

When a carry occurs from bit 7 or 15 or a borrow occurs into bit 7 or 15 as a result of operation instruction execution, the carry flag (CY) is set to (1); else, the flag is reset to (0). It can be tested by executing a conditional branch instruction.

When a 1-bit manipulation instruction is executed, the flag functions as a 1-bit accumulator.

(c) Zero flag (Z)

When the operation result is zero, the zero flag (Z) is set to (1); else, the flag is reset to (0). It can be tested by executing a conditional branch instruction.

(d) Sign flag (S)

When the most significant bit (MSB) of the operation result is set to "1", the sign flag (S) is set to (1); when "0", the flag is reset to (0). It can be tested by executing a conditional branch instruction.

(e) Parity/overflow flag (P/V)

Only when an overflow or underflow occurs as two's complement during arithmetic operation instruction execution, the parity/overflow flag (P/V) is set to (1); else, the flag is reset to (0). (Overflow flag operation)

When the number of 1 bits of the operation result is even during logical operation instruction execution, the parity/overflow flag (P/V) is set to (1); when odd, the flag is reset to (0). However, this is effective only for lower 8 bits of the operation result, irrespective of 16-bit operation or 8-bit operation.

(Parity flag operation)

It can be tested by executing a conditional branch instruction.

(f) Auxiliary carry flag (AC)

When a carry occurs from bit 3 or a borrow occurs into bit 3 as a result of operation, the auxiliary carry flag (AC) is set to (1); else, the flag is reset to (0). It can be tested by executing a conditional branch instruction.

(g) Register set selection flag (RSS)

The register set selection flag is used to specify the general registers which function as X, A, C, and B. The correspondence between function registers and absolute registers is changed depending on the RSS contents, as listed in Table 3-2.

(h) Interrupt request enable flag (IE)

The interrupt request enable flag (IE) indicates whether an interrupt request is enabled or disabled. When the EI instruction is executed, the flag is set to (1); when the DI instruction is executed or an interrupt is acknowledged, the flag is reset to (0).

(i) Register bank selection flag (RBS0 to RBS2)

RBS0 to RBS2 are three bits flags to select one eight register banks (registers banks 0 to 7).

(j) User flag (UF)

The user flag (UF) can be used for program control by setting or resetting on user program.

(3) Stack pointer (SP)

The stack pointer (SP) is a 16-bit register which retains the top address of stack memory area (LIFO). It is handled by executing a dedicated instruction.

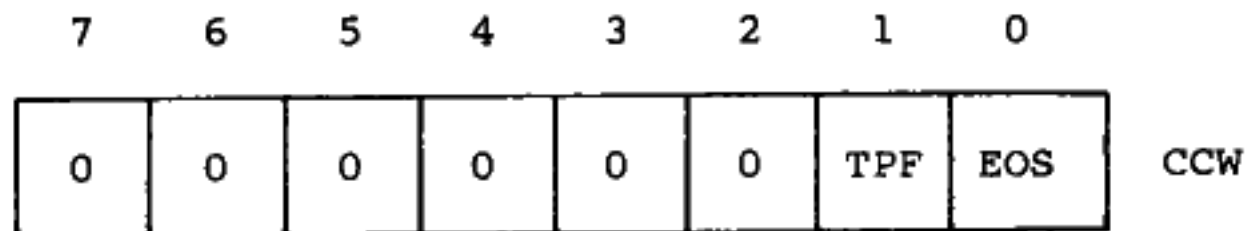
The SP is decremented before write operation into the stack memory (save); it is incremented after read operation from the stack memory (restore).

When  $\overline{\text{RESET}}$  is input, the SP contents become undefined.

(4) CPU control word (CCW)

The CPU control word (CCW) is an 8-bit register which consists of CPU control flags. It is mapped in the special function register area and can be controlled by software. When  $\overline{\text{RESET}}$  is input, all the bits are reset to (0).

Figure 3-4 CCW Format



(a) Table position flag (TPF)

The table position flag (TPF) is used to specify the memory area used as an interrupt vector table area and CALLT instruction table area.

Since TPF is reset to (0) after  $\overline{\text{RESET}}$  is input, addresses 0000H to 007FH are used as the interrupt vector table area and CALLT instruction table area. Addresses 8000H to 807FH of external memory area can be used as the table areas instead of addresses 0000H to 007FH by setting TPF to (1) by software.

(b) End-of-software interrupt flag (EOS)

The end-of-software interrupt flag (EOS) is used to disable RETI or RETCS instruction execution from resetting the in-service priority register (ISPR). When the EOS is set to (1), ISPR resetting is disabled.

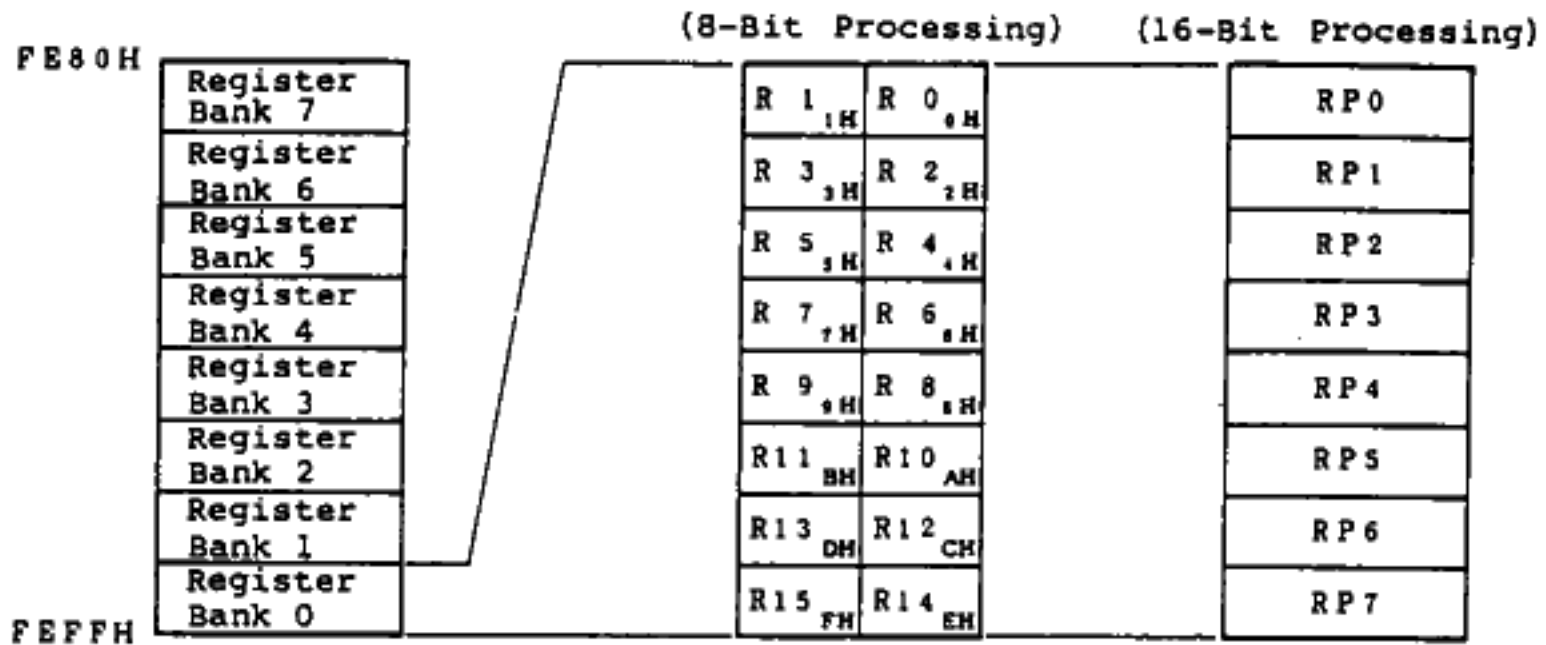
If ISPR is reset when a return is made from a software interrupt, interrupt nesting control is destroyed. (See 5.1.4.) To return from the interrupt service routine started by executing the BRK or BRKCS instruction, be sure to set the EOS to (1) just before the RETI or RETCS instruction is executed.

When the RETI or RETCS instruction is executed, the EOS is cleared (0).

### 3.2.2 GENERAL REGISTERS

The general registers are mapped in specific area of internal RAM space, FE80H to FEFFH (128 bytes in total). Eight register banks are provided. Each register bank consists of 16 8-bit general registers.

Figure 3-5 Memory Locations of General Registers



The 2 8-bit registers are also paired as eight 16-bit register pairs (RP0 to RP7).

The 16 8-bit registers are characterized by the function names as listed in Table 3-2. The X register functions as the low-order byte of a 16-bit accumulator. The A register functions as an 8-bit accumulator or the high-order byte of a 16-bit accumulator. The B and C registers function as counters. The register pairs DE, HL, VP, and UP function as address registers. Particularly, the VP register pair has the base register function and the UP register pair has the user stack pointer function.



Table 3-2 General Register Configuration

Absolute Name	Function Name	
	RSS = 0	RSS = 1
R0	X	
R1	A	
R2	C	
R3	B	
R4		X
R5		A
R6		C
R7		B
R8	VPL	VPL
R9	VPH	VPH
R10	UPL	UPL
R11	UPH	UPH
R12	E	E
R13	D	D
R14	L	L
R15	H	H

Absolute Name	Function Name	
	RSS = 0	RSS = 1
RP0	AX	
RP1	BC	
RP2		AX
RP3		BC
RP4	VP	VP
RP5	UP	UP
RP6	DE	DE
RP7	HL	HL

The registers having the unique functions change according to the PSW register set selection flag (RSS) value as listed in Table 3-2.

As process data addressing, the uPD78312A enables implied addressing by using the function names emphasizing the unique function to each register and register addressing by using the absolute names for high speed processing where data transfer is made not frequently and preparation of highly descriptive programs.

### 3.2.3 SPECIAL FUNCTION REGISTERS (SFRs)

Unlike the general registers, the special function registers (SFRs) have special functions. The SFRs are mapped in memory address space of FF00H to FFFFH (256 bytes).

Short direct addressing is applicable to the 32-byte area of FF00H to FF1FH. Thus, the SFRs mapped in this area can be processed with shorter word length and less clocks than SFRs in other locations. Frequently accessed SFRs such as timer compare registers, capture registers, and ports are mapped in the area.

SFRs can be handled as general registers by executing instructions such as operation, transfer, or bit manipulation. SFRs are handled in 1-, 8-, or 16-bit units. However, SFRs that can be handled in 16-bit units are limited. (See Table 3-3.) The SFR specification method is described below:

#### . 1-bit manipulation

Describe the symbol and bit in the 1-bit manipulation instruction operand (sfr.bit). SFR can also be specified by using the address.

#### . 8-bit manipulation

Describe the symbol in the 8-bit manipulation instruction operand (sfr). SFR can also be specified by using the address.

#### . 16-bit manipulation

Describe the symbol in the 16-bit manipulation instruction operand (sfrp). SFR that can be handled in 16-bit units is mapped in a contiguous 2-byte area (even and odd addresses). To specify the SFR by using the address, describe the even address.

Table 3-3 lists the SFRs. The meanings of the items in the table are as follows:

- . Symbol : Symbol which indicates the internal SFR address.  
  
It can be described as instruction operand.
- . R/W : Indicates whether or not the SFR can be read/written.  
  
R/W: The SFR can be read/written.  
R : The SFR can only be read.  
W : The SFR can only be written.
- . 16-bit manipulation: Indicates whether or not the SFR can be handled in 16-bit units.  
○ denotes that the SFR can be handled in 16-bit units.
- . When reset : Indicates the register state when RESET is input.

NOTE: In the F00H to FFFFH area, the addresses in which no SFR is mapped cannot be accessed. If the address is accessed, malfunction may occur.

Table 3-3 Special Function Register (SFR) List

Address	Special Function Register (SFR) Name	Symbol		R/W	16-bit Manipulation	When Reset
FF00H	Port 0	P 0		R/W	-	Un-defined
FF01H	Port 1	P 1				
FF02H	Port 2	P 2		(*) R/W	-	
FF03H	Port 3	P 3				
FF04H	Port 4	P 4		R/W	-	
FF05H	Port 5	P 5				
FF08H	Capture/compare register 00	CR00L	CR00	R/W	o	
FF09H		CR00H				
FF0AH	Capture/compare register 01	CR01L	CR01		o	
FF0BH		CR01H				
FF0CH	Capture/compare register 10	CR10L	CR10		o	
FF0DH		CR10H				
FF0EH	Capture/compare register 11	CR11L	CR11		o	
FF0FH		CR11H				
FF10H	Capture register 0	CPT0L	CPT0		o	
FF11H		CPT0H				
FF12H	Capture register 1	CPT1L	CPT1		o	
FF13H		CPT1H				
FF14H	PWM register 0	PWM0L	PWM0		o	
FF15H		PWM0H				
FF16H	PWM register 1	PWM1L	PWM1		o	
FF17H		PWM1H				

(to be continued)

\*: Bits 0 to 3 of P2 and P3 are read only.

Table 3-3 Special Function Register (SFR) List (cont'd)

Address	Special Function Register (SFR) Name	Symbol		R/W	16-bit Manipulation	When Reset
FF1CH	Presettable up/down count register 0	UDC0L	UDC0	R/W	0	Un-defined
FF1DH		UDC0H				
FF1EH	Presettable up/down count register 1	UDC1L	UDC1		0	
FF1FH		UDC1H				
FF20H	Port 0 mode register	PM0		R/W	-	FFH
FF21H	Port 1 mode register	PM1			FFH	
FF22H	Port 2 mode register	PM2			-	FFH
FF23H	Port 3 mode register	PM3			FFH	
FF25H	Port 5 mode register	PM5			-	FFH
FF32H	Port 2 mode control register	PMC2		R/W	-	0FH
FF33H	Port 3 mode control register	PMC3			0FH	
FF38H	Real time output port control register	RTPC			-	08H
FF3AH	Port 0 buffer register(*1)	POL		R/W	-	Un-defined
FF3BH		POH				
FF40H	Memory expansion mode register	MM		R/W	-	30H
FF41H	Refresh mode register	RFM			10H	
FF42H	Watchdog timer mode register	WDM			-	00H
FF44H	Standby control register	STBC			-	(*2) 2 x H

(to be continued)

\*1: POH and POL are 4-bit buffers; POH is the high-order four bits and POL is the low-order four bits. POH and POL can also be paired for 8-bit manipulation. (See Table 4-1.)

\*2: Bit 3 is not affected by  $\overline{\text{RESET}}$  input, thus the low-order four bits are set to 0 or 8.

Table 3-3 Special Function Register (SFR) List (cont'd)

Address	Special Function Register (SFR) Name	Symbol	R/W	16-bit Manipulation	When Reset
FF46H	Time base mode register	TBM	R/W	-	OOH
FF48H	External interrupt mode register	INTM		-	OOH
FF4AH	In-service priority register	ISPR	R	-	OOH
FF4EH	CPU control word	CCW	R/W	-	OOH
FF50H	Serial communication mode register	SCM		-	OOH
FF52H	Serial communication control register	SCC		-	OOH
FF53H	Baud rate generator	BRG		-	OOH
FF56H	Serial communication receive buffer	RxB	R	-	Un-defined
FF57H	Serial communication transmit buffer	TxB	W	-	
FF60H	Free running counter control register	FRCC	R/W	-	OOH
FF64H	Capture mode register	CPTM		-	OOH
FF66H	PWM mode register	PWMM		-	OOH
FF68H	Analog-to-digital converter mode register	ADM		-	OOH
FF6AH	Analog-to-digital conversion result register	ADCR	R	-	Un-defined
FF70H	Count unit input mode register	CUIM	R/W	-	OOH
FF72H	Up/down counter control register 0	UDCC0		-	OOH
FF74H	Capture compare register control register	CRC		-	OOH
FF7AH	Up/down counter control register 1	UDCC1		-	OOH

(to be continued)

Table 3-3 Special Function Register (SFR) List (cont'd)

Address	Special Function Register (SFR) Name	Symbol		R/W	16-bit Manipulation	When Reset			
FF80H	Timer control register 0	TMC0		R/W	-	00H			
FF82H	Timer control register 1	TMC1			-	00H			
FF88H	Timer register 0	TM0L	TM0		o	Un-defined			
FF89H		TM0H							
FF8AH	Modulo/timer register 0	MD0L	MD0						
FF8BH		MD0H							
FF8CH	Timer register 1	TM1L	TM1						
FF8DH		TM1H							
FF8EH	Modulo/timer register 1	MD1L	MD1						
FF8FH		MD1H							
FFB0H to FFBFH	External access area (*)						-		
FFC0H	Count unit 0 interrupt request control register 0	CRIC00					R/W	-	47H
FFC1H	Count unit 0 macro service control register 0	CRMS00						Un-defined	
FFC2H	Count unit 0 interrupt request control register 1	CRIC01						-	47H
FFC4H	Count unit 1 interrupt request control register 0	CRIC10		-				47H	
FFC5H	Count unit 1 macro service control register 0	CRMS10		Un-defined					
FFC6H	Count unit 1 interrupt request control register 1	CRIC11		-	47H				

(to be continued)

\*: The external memory space can be accessed by SFR addressing.

Table 3-3 Special Function Register (SFR) List (cont'd)

Address	Special Function Register (SFR) Name	Symbol	R/W	16-bit Manipulation	When Reset
FFC8H	External interrupt pin interrupt request control register 0	EXIC0	R/W	-	47H
FFC9H	External interrupt pin macro service control register 0	EXMS0			Un-defined
FFCAH	External interrupt pin interrupt request control register 1	EXIC1		-	47H
FFCBH	External interrupt pin macro service control register 1	EXMS1			Un-defined
FFCCH	External interrupt pin interrupt request control register 2	EXIC2		-	47H
FFCDH	External interrupt pin macro service control register 2	EXMS2			Un-defined
FFCEH	Timer unit interrupt request control register 0	TMIC0		-	47H
FFCFH	Timer unit macro service control register 0	TMMS0			Un-defined
FFD0H	Timer unit interrupt request control register 1	TMIC1	R/W	-	47H
FFD1H	Timer unit macro service control register 1	TMMS1			Un-defined
FFD2H	Timer unit interrupt request control register 2	TMIC2		-	47H
FFD3H	Timer unit macro service control register 2	TMMS2			Un-defined
FFDAH	Serial communication reception error interrupt request control register	SEIC		-	47H
FFDCH	Serial communication reception completion interrupt request control register	SRIC			-

(to be continued)



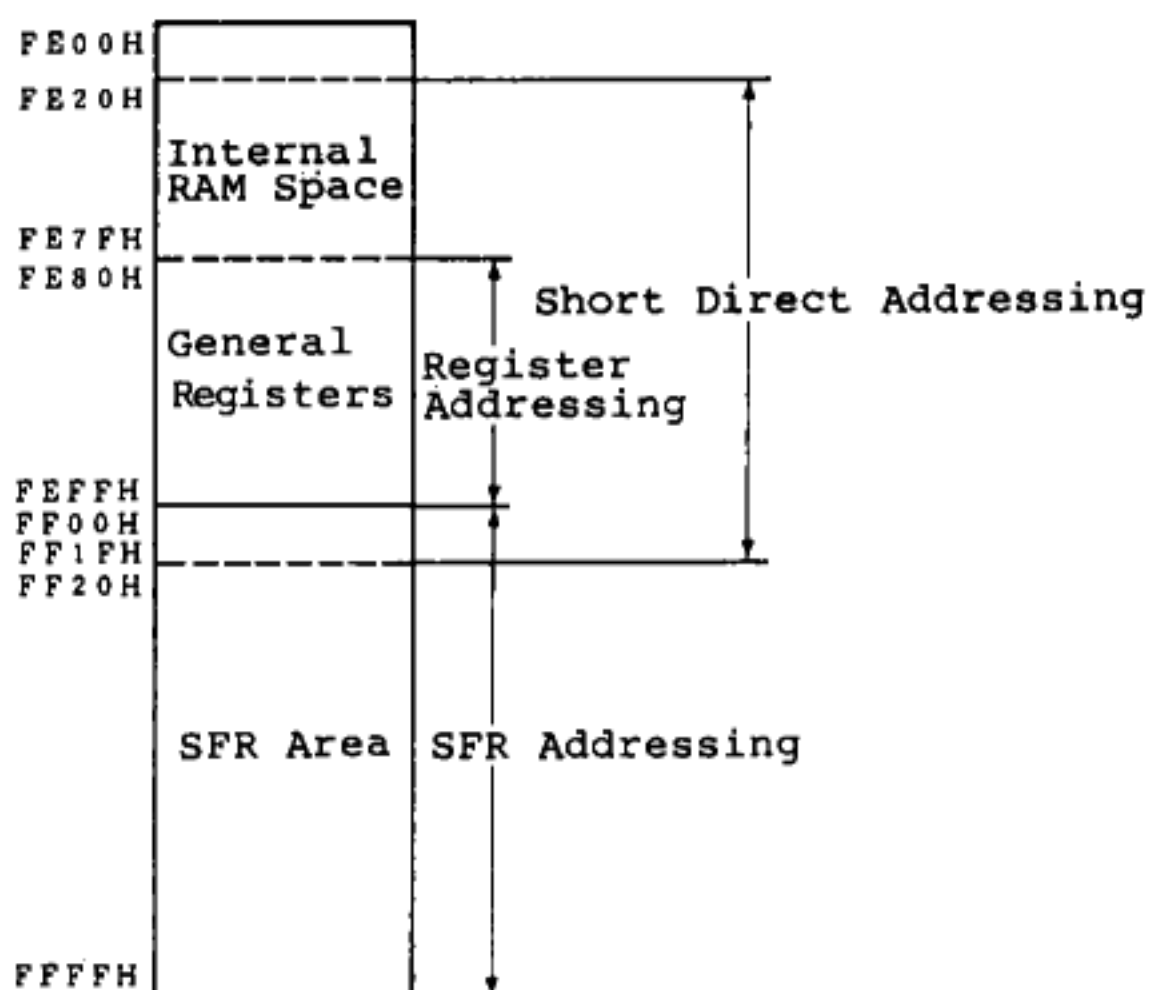
Table 3-3 Special Function Register (SFR) List (cont'd)

Address	Special Function Register (SFR) Name	Symbol	R/W	16-bit Manipulation	When Reset
FFDDH	Serial communication reception completion macro service control register	SRMS	R/W	-	Un-defined
FFDEH	Serial communication transmission completion interrupt request control register	STIC		-	47H
FFDFH	Serial communication transmission completion macro service control register	STMS		-	Un-defined
FFE0H	Analog-to-digital converter interrupt request control register	ADIC	R/W	-	47H
FFE1H	Analog-to-digital converter macro service control register	ADMS		-	Un-defined
FFE2H	Time base counter interrupt request register	TBIC		-	47H

### 3.3 DATA MEMORY ADDRESSING

On the uPD78312A, internal RAM space (FE00H to FEFFH) and special function register area (FF00H to FFFFH) are mapped in the FE00H to FFFFH area. Short direct addressing is used for a part of data memory space (FE20H to FF1FH); direct addressing is enabled with 1-byte data in the instruction word.

Figure 3-6 Addressing Space of Data Memory



#### 3.3.1 GENERAL REGISTER ADDRESSING

The uPD78312A contains eight register banks each consisting of 16 8-bit general registers or eight 16-bit general registers.

A general register is addressed by using 3-bit or 4-bit register specification field supplied from instruction field and the register bank selection flag (RBS0 to RBS2) and the register set selection flag (RSS) in PSW.

### 3.3.2 SHORT DIRECT ADDRESSING

The short direct addressing which enables direct addressing with 1-byte data in instruction word is applicable to the internal RAM space of FE20H to FEFFH and the SFR area of FF00H to FF1FH. Short direct memory is accessed as 8-bit data or 16-bit data. When it is accessed as 16-bit data, the 2-byte data addressed by contiguous addresses of even and odd addresses is accessed regardless of whether 1-byte addressing data is odd or even. (The least significant bit of the addressing data is ignored.)

### 3.3.3 SPECIAL FUNCTION REGISTER (SFR) ADDRESSING

The special function register (SFR) addressing is used to handle the special function registers (SFRs) mapped in the SFR area of FF00H to FFFFH. A special function register is addressed by the 1-byte data in instruction word corresponding to the low-order eight bits of the special function register address.

When SFR that can be handled in 16-bit units is accessed in 16-bit units, the 2-byte data addressed by the contiguous addresses of even and odd addresses is accessed as with the short direct addressing.

## CHAPTER 4. PERIPHERAL HARDWARE FUNCTION

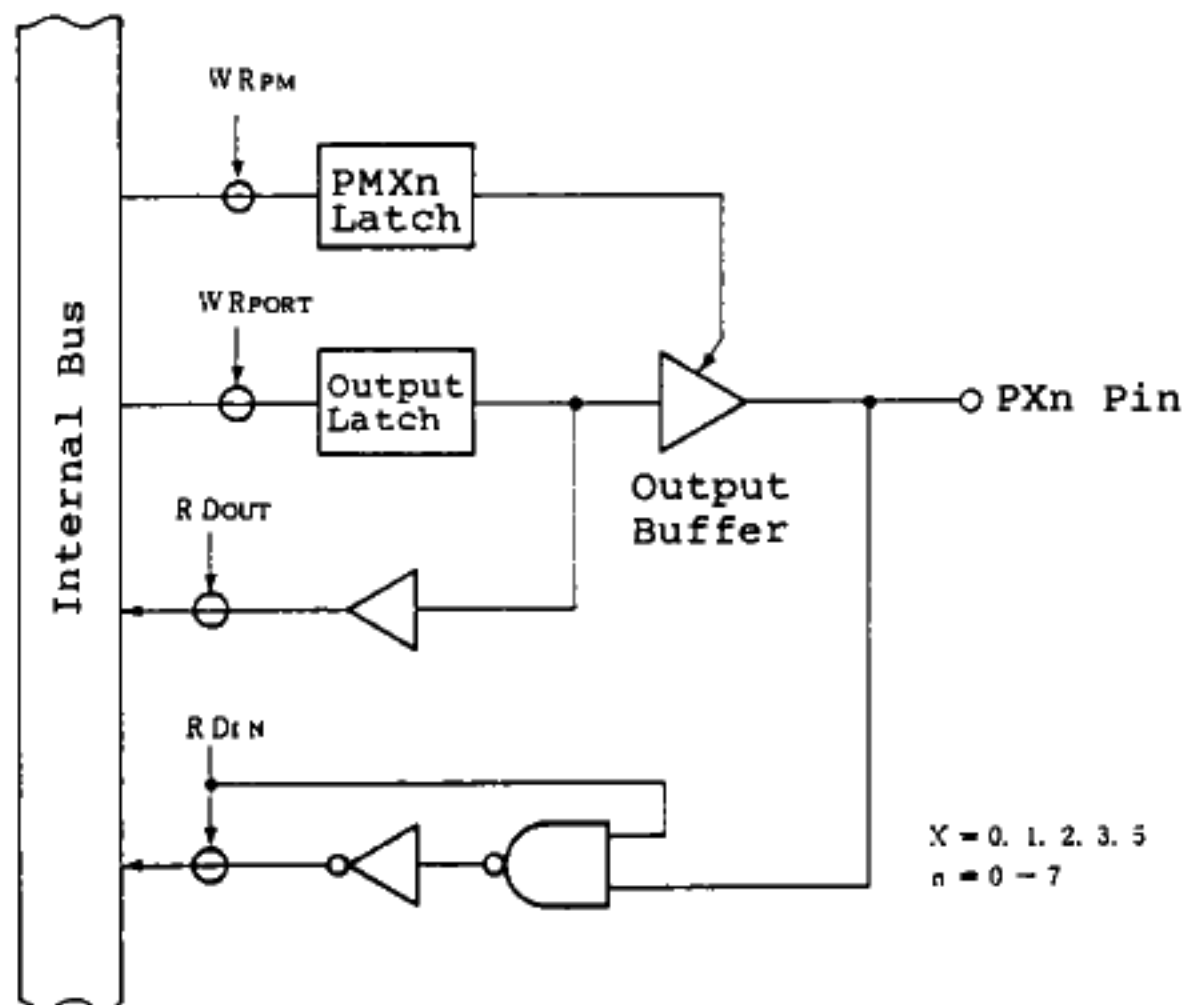
### 4.1 PORT FUNCTION

#### 4.1.1 HARDWARE CONFIGURATION

The uPD78312A ports basically are 3-state bidirectional ports shown in Figure 4-1.

When  $\overline{\text{RESET}}$  is input, each bit of the port mode register is set to (1) and input ports are specified. All the port pins become high impedance. When  $\overline{\text{RESET}}$  is input, the output latch contents become undefined.

Figure 4-1 Basic Structure of Port

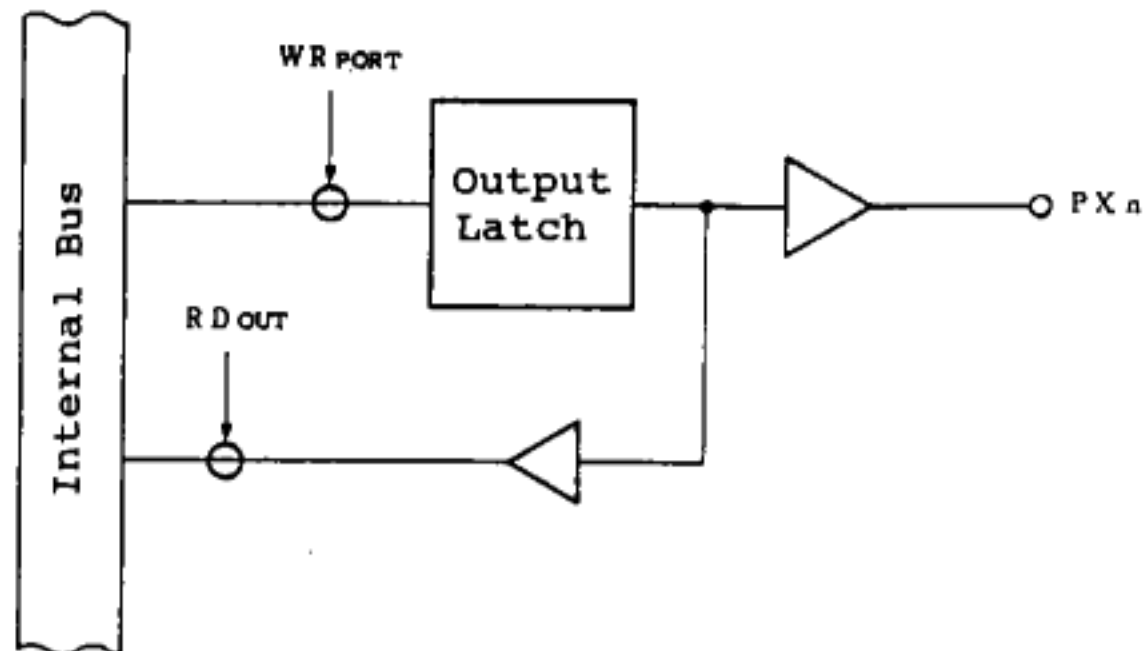


Remarks: PMXn latch: Port mode register PMX bit n

- (1) When port is set to output port ( $PMX_n = 0$ )

Output latch is activated. Data can be transferred between the output latch and accumulator by executing a transfer instruction. The output latch contents can be set/reset bit-wise. Data once written into the output latch is retained until a new instruction handling the port is executed.

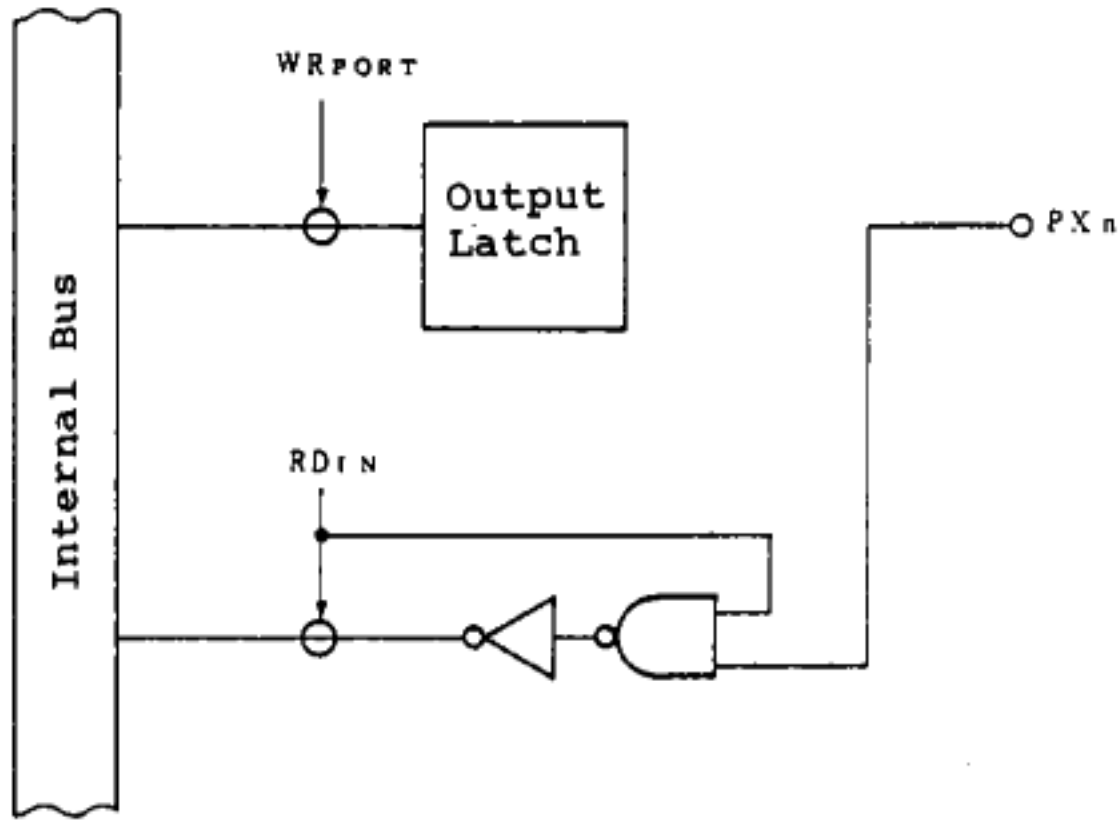
Figure 4-2 Port Set to Output Port



- (2) When port is set to input port ( $PMX_n = 1$ )

The port pin level can be loaded into a given accumulator by executing a transfer instruction. In this case, data can also be written into the output latch; data transferred from the accumulator by executing a transfer instruction is stored in output latch regardless of whether the port is set to input or output port. However, since the output buffer of the bit set to input port is high impedance, no output is made to the port pin. (When the bit set to input port is changed to output port, the output latch contents are output to the port pin.) The contents of the output latch of the bit set to input port cannot be read. (See Figure 4-3.)

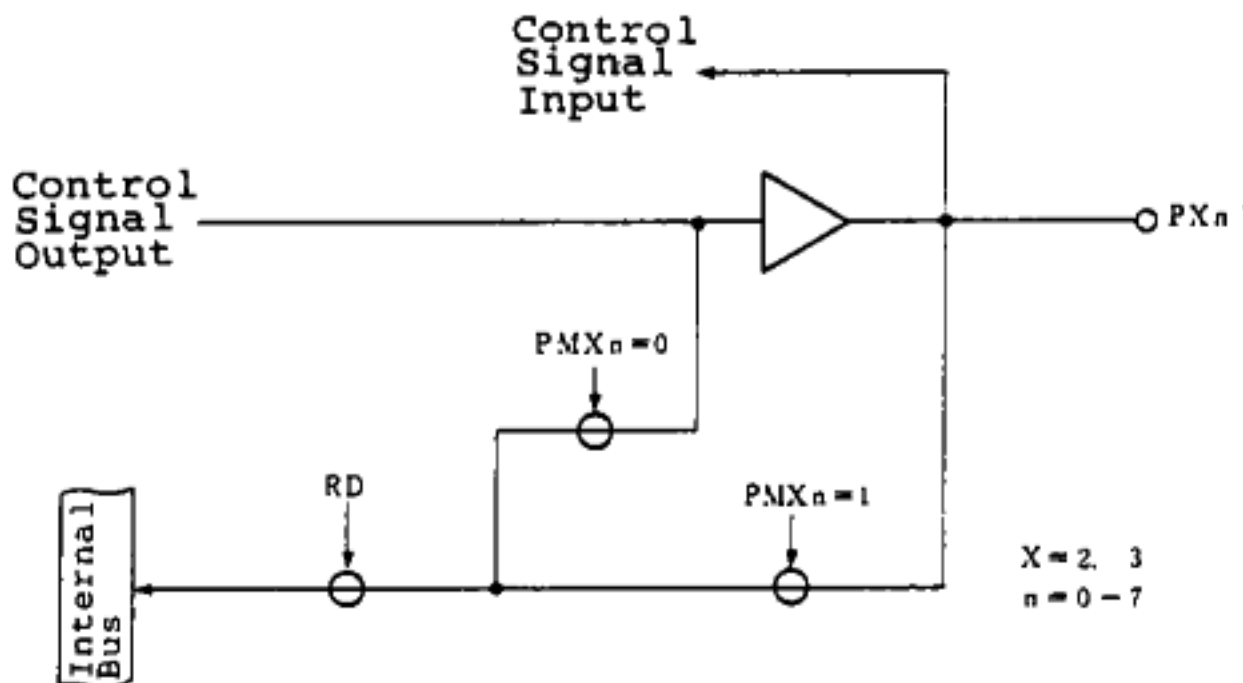
Figure 4-3 Port Set to Input Port



- (3) When port is set to control signal mode (PMC2n, PMC3n = 1)

When port mode control register (PMC2, PMC3) bits are set to (1), ports 2 and 3 can be used bit-wise as control signal input or output regardless of how the port mode register (PM2, PM3) is set. When each pin is used as control signal, the control signal state can be read by executing a port access instruction.

Figure 4-4 When Port is Set to Control Mode



(a) When port outputs control signal

If a port read instruction is executed when the port mode register (PM2n, PM3n) is set to (1), the control signal pin state can be read.

If a port read instruction is executed when the port mode register is reset to (0), the internal control signal state can be read.

(b) When port inputs control signal

If a port read instruction is executed only when the port mode register is set to (1), the control signal pin state can be read.

#### 4.1.2 PORT FUNCTION

(1) Digital input/output port function and features

Table 4-1 lists the digital input/output ports.

On every port, input/output data can be handled in 1-bit units as well as 8-bit units, and very diversified control can be performed.

Table 4-1 Port Function and Features

Port Name	Function	Operation and Features	Remarks
Port 0	8-bit input/output	The input or output mode can be selected bit-wise.  The port can be set to real time output port in 4-bit units.	See 4.3.5 for the real time output port function.
Port 1	8-bit input/output	The input or output mode can be selected bit-wise.	-
Port 2	8-bit input/output (P20 to P23 are input only)	The port 2 pins can be set to port or control pins bit-wise.  The input or output mode can be selected bit-wise for P24 to P27.	The pins are also used for NMI, INTE0 to INTE2, $\overline{\text{TxD}}$ , $\overline{\text{RxD}}$ , $\overline{\text{SCK}}$ , and CTS.
Port 3	8-bit input/output (P30 to P33 are input only)	The port 3 pins can be set to port or control pins bit-wise.  The input or output mode can be selected bit-wise for P34 to P37.	The pins are also used for CIO, CIL, CTRL0, CTRL1, PWM0, PWM1, T00, T01, CLR0, and CLR1.
Port 4	8-bit input/output	The input or output mode can be selected in 8-bit units.  Port 4 functions as multiplexed address/data bus (AD0 to AD7) during the external memory expansion mode.	On the uPD78310A, port 4 always function as the multiplexed address/data bus.

(to be continued)



Table 4-1 Port Function and Features (cont'd)

Port Name	Function	Operation and Features	Remarks
Port 5	8-bit input/output	<p>The input or output mode can be selected bit-wise.</p> <p>Port 5 functions as the address bus (A8 to A15) during the external memory expansion mode.</p> <p>The pins not used as the address bus can be used as port pins.</p>	<p>On the uPD78310A, port 5 always functions as the address bus.</p>

(2) Input/output mode and control mode setting

The input or output mode is selected for each port bit-wise by setting each port mode register, as shown in Figure 4-5. The input or output mode is selected in 8-bit units only for port 4 by setting the memory expansion mode register (MM), as shown in Figure 4-6.

The control mode of each pin of ports 2 and 3 can be specified bit-wise by setting the port mode control registers (PMC2 and PMC3), as shown in Figures 4-7 and 4-8.

When a PMC2 or PMC3 bit is set to (1), the port 2 or 3 pin corresponding to the bit functions as the control pin regardless of P2, PM2, P3, PM3. Although the P20 to P23 and P30 to P33 pins are fixed to the control mode, the pin level can be read by executing a read instruction.

Data can be set in the mode registers PM0 to PM3, PM5, MM, PMC2, and PMC3 in 8-bit units; they can also be handled bit-wise.

When  $\overline{\text{RESET}}$  is input, PM0 to PM3 and PM5 are set to FFH; PMC2 and PMC3 are set to 0FH; and MM is set to 30H.

Figure 4-5 Port Mode Register Format

	7	6	5	4	3	2	1	0	Address	After Reset
PM0	PM07	PM06	PM05	PM04	PM03	PM02	PM01	PM00	FF20H	FFH
PM1	PM17	PM16	PM15	PM14	PM13	PM12	PM11	PM10	FF21H	FFH
PM2	PM27	PM26	PM25	PM24	1	1	1	1	FF22H	FFH
PM3	PM37	PM36	PM35	PM34	1	1	1	1	FF23H	FFH
PM5	PM57	PM56	PM55	PM54	PM53	PM52	PM51	PM50	FF25H	FFH

PM <sub>m</sub> n	Input or output mode selection for P <sub>m</sub> n pin (m = 0 to 3, 5, n = 0 to 7)									
0	Output mode (output buffer on)									
1	Input mode (output buffer off)									

Figure 4-6 Memory Expansion Mode Register Format

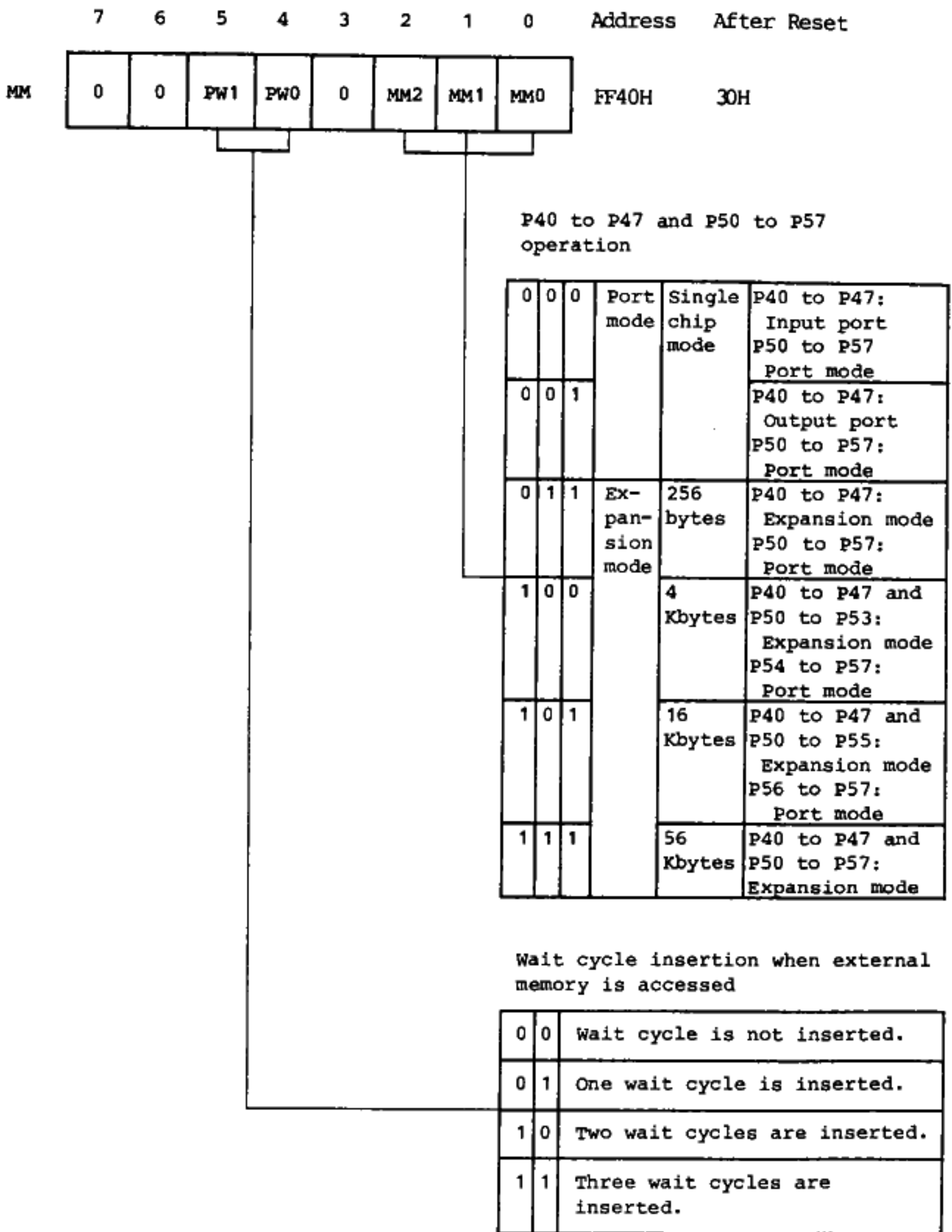


Figure 4-7 Port 2 Mode Control Register Format

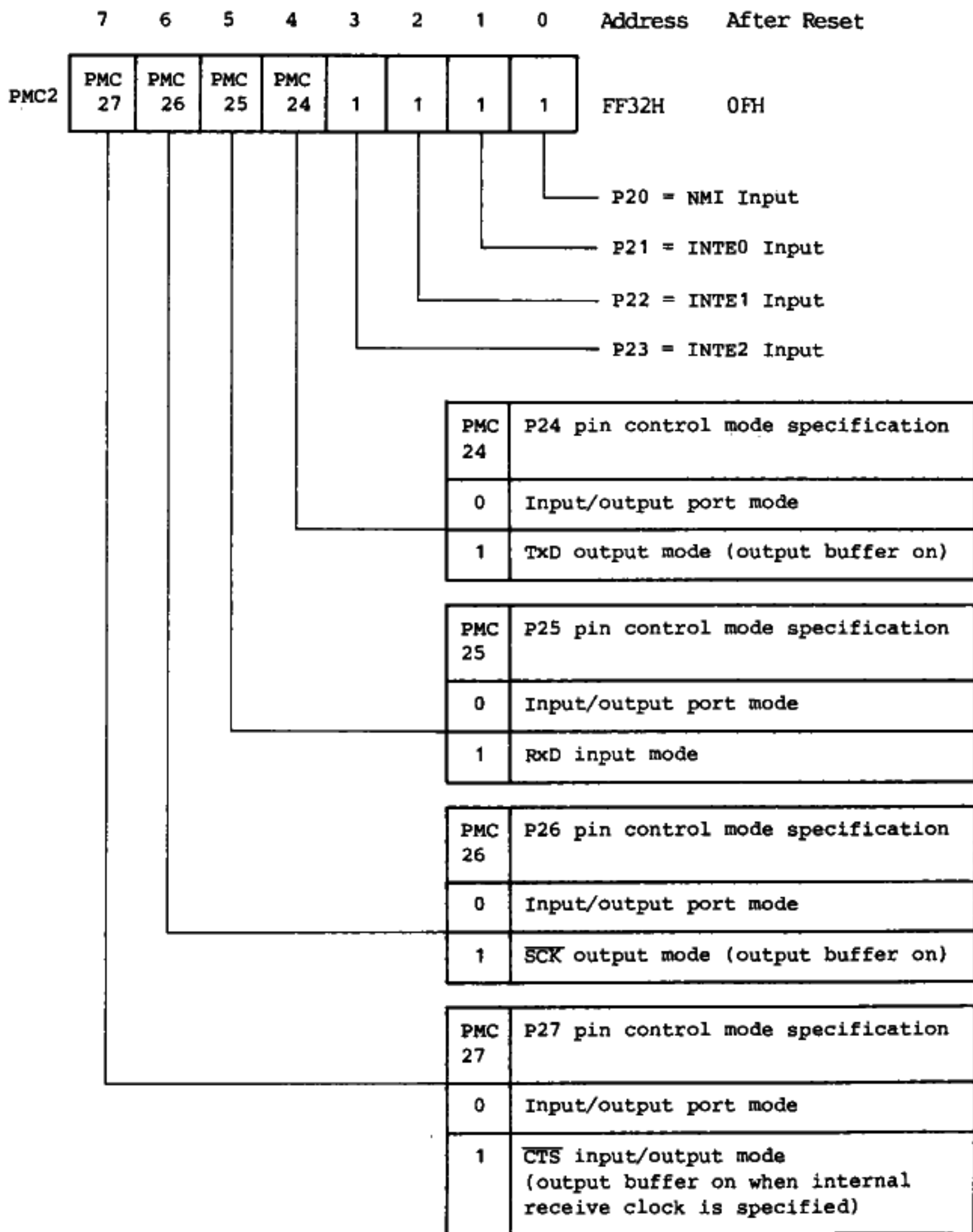
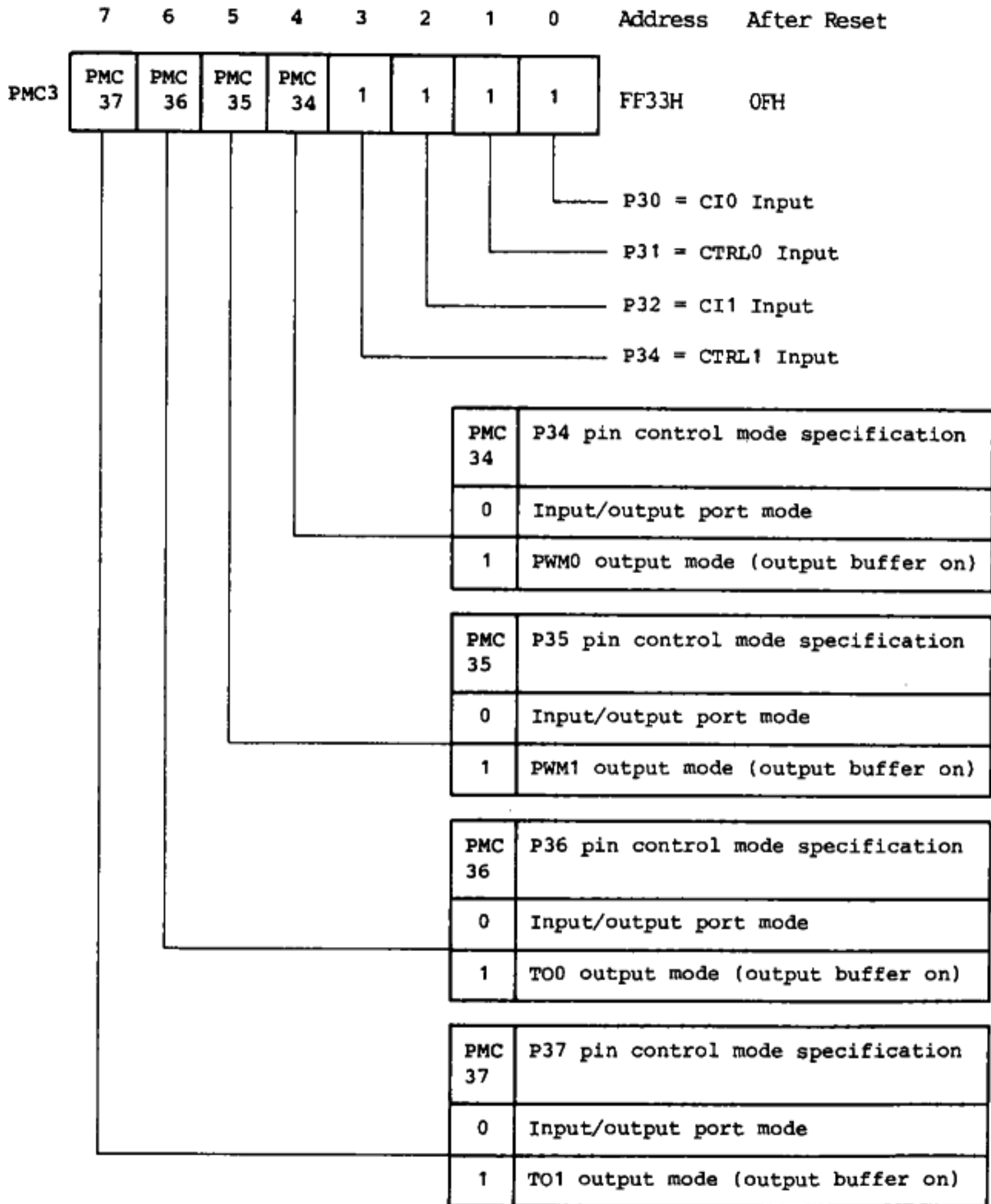


Figure 4-8 Port 3 Mode Control Register Format



### (3) Digital input/output port operation

When a read/write instruction is executed, operation of each port varies depending on the specified mode, as listed in Tables 4-2 to 4-5. P20 to P23 and P30 to P33 are input-only ports also used for control signal input. The pin level can always be read or tested.

The P36 and P37 pins also function as count unit counter clear/capture trigger pins (CLR0 and CLR1), as specified in the up/down counter control register (UDCC0, UDCC1) ENCLR bit and capture/compare register control register (CRC) CM00 and CM10 bits. (See Figures 4-13 and 4-14.)

Ports 4 and 5 of the uPD78312A, uPD78P312A operate as listed in Table 4-6 according to MM register specification. When the expansion mode is specified, ports 4 and 5 do not function as ports. Port 4 of the uPD78310A always operates as multiplexed address/data bus (AD0 to AD7) and port 5, as address bus (A8 to A15); they do not function as ports.

When the output mode is selected for a port pin, immediately the output latch contents are output to the port pin. To output data, prewrite the data into output latch before selecting the output mode.

When  $\overline{\text{RESET}}$  is input, each port enters the input port mode and all the port pins become high impedance. When  $\overline{\text{RESET}}$  is input, the output latch contents become undefined.

NOTE: If data is written into output latch by executing a bit manipulation instruction during the input mode or control signal input/output mode, all the output latch contents become undefined.

Table 4-2 Port 0, 1 Operation when Read/Write Instruction is Executed (n = 0 to 7)

Port 0, 1 Operating Mode	Read Instruction Execution	Write Instruction Execution
Output port mode	Output latch data is input.	Data is output to output pin.
Input port mode	Pin level is input.	Data is set in output latch.

Table 4-3 Port 2 Operation when Read/Write Instruction is Executed (n = 0 to 7)

Pin	PMC2n	PM2n	Read Instruction Execution	Write Instruction Execution
P20 to P23	1	1	Pin level is input.	Written data is invalid.
P24, P26	0	0	Output latch data is input.	Data is output to output pin.
	0	1	Pin level (port input) is input.	Data is set in output latch.
	1	0	Internal control signal is input.	
	1	1	Pin level (control signal) is input.	
P25, P27	0	0	Output latch data is input.	Data is output to output pin.
	0	1	Pin level (port input) is input.	Data is set in output latch.
	1	0	Output latch data is input.	
	1	1	Pin level (control signal) is input.	

**Table 4-4 Port 3 Operation when Read/Write Instruction is Executed (n = 0 to 7)**

Pin	PMC3n	PM3n	Read Instruction Execution	Write Instruction Execution
P30 to P33	1	1	Pin level is input.	Written data is invalid.
P34 to P37	0	0	Output latch data is input.	Data is output to output pin.
	0	1	Pin level (port input) is input.	
	1	0	Internal PWM and TO output signals are input.	
	1	1	Pin levels (PWM and TO output) are input.	

**Table 4-5 Port 4, 5 Operation when Read/Write Instruction is Executed (uPD78312A, uPD78P312A)**

Port 4, 5 Operating Mode		Read Instruction Execution	Write Instruction Execution
Output port mode		Output latch data is input.	Data is output to output pin.
Input port mode		Pin level is input.	Data is set in output latch.
Expansion mode	Port 4	Pin level is input.	Written data is invalid.
	Port 5	0 is input.	



Table 4-6 Port 4, 5 Operation (uPD78312A, uPD78P312A)

Mode Specification in MM Register		Port 4	Port 5							
			P57	P56	P55	P54	P53	P52	P51	P50
Port mode	Single chip	Input port	Port mode							
		Output port	Port mode							
Expansion mode	256-byte expansion	Multiplexed address/ data bus	Port mode							
	4-Kbyte expansion	Multiplexed address/ data bus	Port mode				A11	A10	A9	A8
	16-Kbyte expansion	Multiplexed address/ data bus	Port mode		A13	A12	A11	A10	A9	A8
	56-Kbyte expansion	Multiplexed address/ data bus	A15	A14	A13	A12	A11	A10	A9	A8

An: Address bus

## 4.2 CLOCK GENERATOR

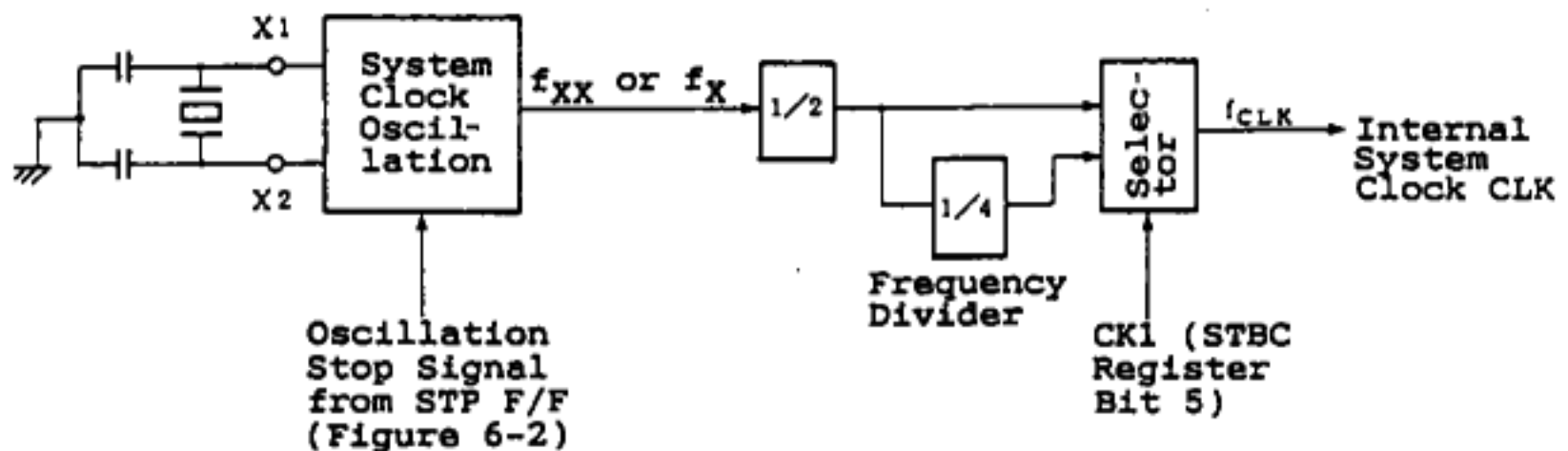
The clock generator generates and controls internal system clock supplied to the CPU and peripheral hardware.

Figure 4-9 shows the clock generator configuration.

The system clock oscillator oscillates by a crystal vibrator or ceramic oscillator connected to the X1 and X2 pins.

External clock can also be input. In this case, input the clock signal to the X1 pin and its inverted phase to the X2 pin.

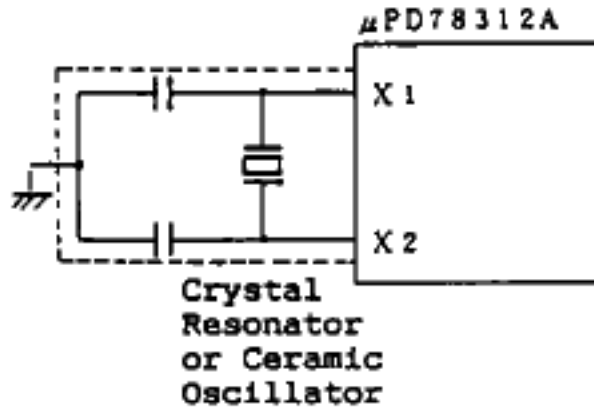
Figure 4-9 Clock Generator Configuration



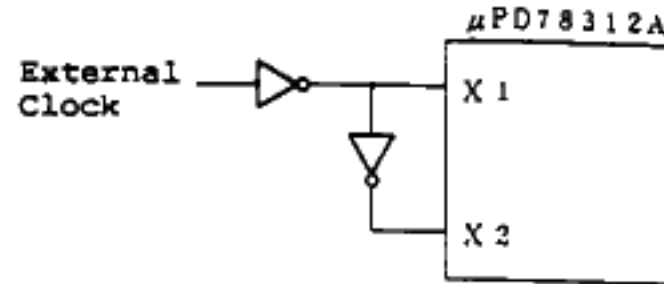
- Remarks 1:  $f_{XX}$  = crystal or ceramic oscillation frequency
- 2:  $f_X$  = external clock frequency
- 3:  $f_{CLK}$  = internal system clock frequency
- 4: CK1 is STBC register bit 5 (see Figure 6-1)

Figure 4-10 System Clock Oscillator External Circuit

(a) Crystal or ceramic oscillation



(b) External clock



NOTE: When the system clock oscillator is used, wiring within the shaded area should be carried out as follows to avoid the effects of wiring capacitance, etc.

- . Keep wiring as short as possible.
- . Do not cross other signal lines and keep clear of lines carrying a variable high current.
- . The ground point of the oscillator capacitors should be such that the potential is always the same as  $V_{SS}$ . Do not ground them in a ground pattern in which a high current flows.
- . Do not take signals from the oscillator.

The internal system clock (CLK) is changed by setting standby control register (STBC) CK1 bit, as listed in Table 4-7. When RESET is input, the CK1 bit is set to (1) and  $f_{CLK}$  equals to  $f_{XX}/8$  (low speed operating mode.)

Table 4-7 Internal System Clock Specification

CK1	Internal System Clock Dividing Ratio (when $f_{XX} = 12$ MHz)
0	$f_{CLK} = f_{XX} \times 1/2$ (6 MHz)
1	$f_{CLK} = f_{XX} \times 1/8$ (1.5 MHz)

Remarks: To use external clock, replace  $f_{XX}$  with  $f_X$ .

NOTE: In a system which uses external clock, do not set the STOP mode. (See 6.1.5.)

### 4.3 PULSE INPUT/OUTPUT UNIT

The pulse input/output unit consists of the following five blocks:

- . Count unit
- . Capture unit
- . PWM unit
- . Timer unit
- . Real time output port

#### 4.3.1 COUNT UNIT

The count unit can count external event input and measure external event input intervals.

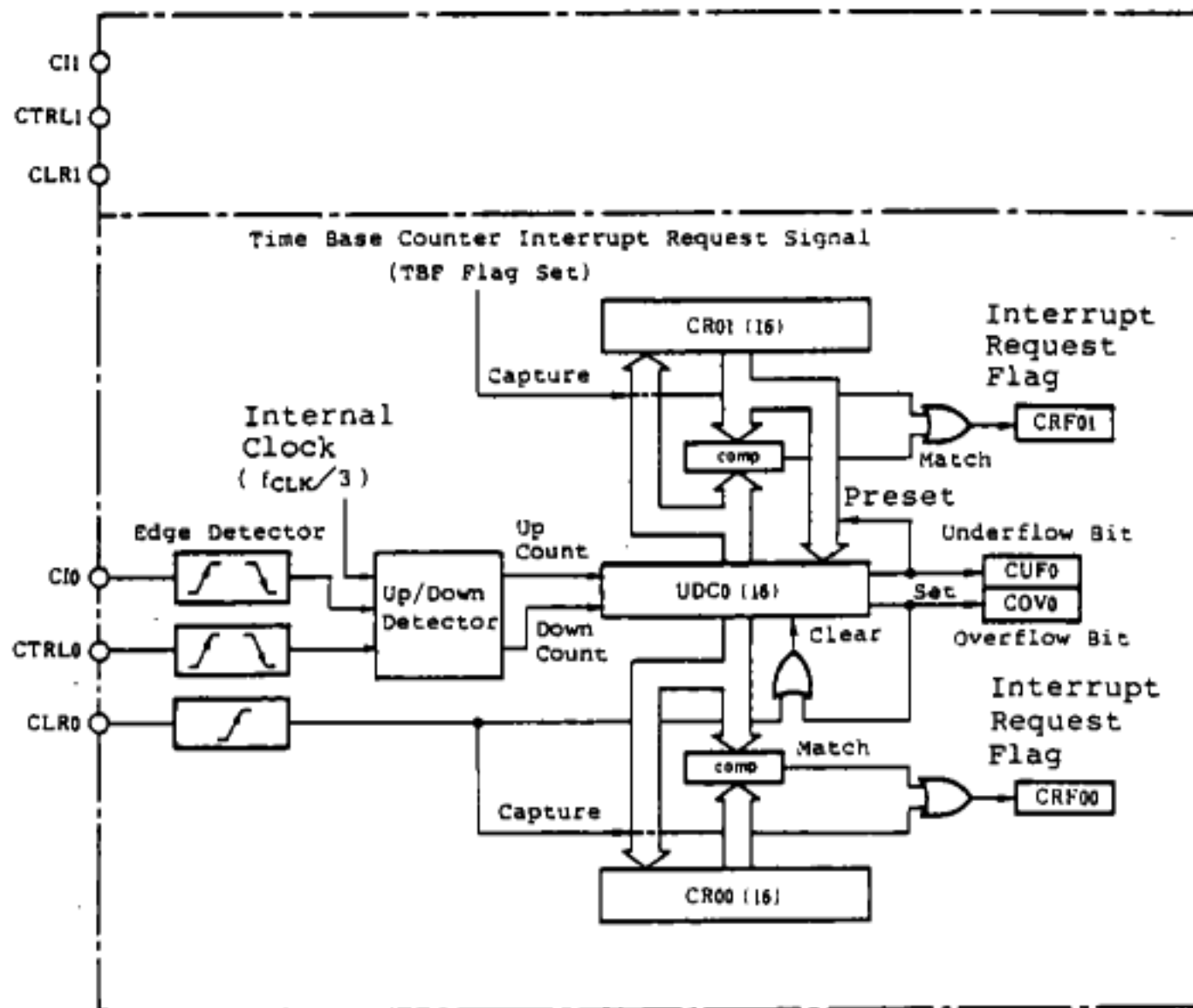
##### (1) Count unit configuration

The count unit consists of the following registers:

- . 16-bit presetable up/down count registers (UDC0 and UDC1)
- . 16-bit capture/compare registers (CR00, CR01, CR10, and CR11)
- . Up/down counter control registers (UDCC0 and UDCC1)
- . Capture/compare register control register (CRC)
- . Count unit input mode register (CUIM)
- . Interrupt request control registers (CRIC00, CRIC01, CRIC10, and CRIC11)
- . Macro service control registers (CRMS00 and CRMS10)

Two blocks xxx0 and xxx1 are the same in structure and function.

Figure 4-11 Count Unit Block Diagram



(a) 16-bit presettable up/down count registers (UDC0 and UDC1)

Each of UDC0 and UDC1 is an up/down counter which counts internal clock ( $f_{CLK}/3$ ) or external clock (C10, C11 pin input). ( $f_{CLK}$  is the internal system clock frequency.)

UDC0 or UDC1 up count/down count operation can be controlled by software or external pin (CTRL0 or CTRL1). The clear function and data preset function are also contained.

UDC0 operation and UDC1 operation are controlled by the UDCC0 and UDCC1 registers.

When  $\overline{RESET}$  is input, UDC0 and UDC1 are not affected.

- (b) Capture/compare registers (CR00, CR01, CR10, and CR11)

The capture/compare register functions as a register which retains the preset value in the UDC0 (UDC1) register, a compare register which compares with the UDC0 (UDC1) register count value, or a capture register which captures the UDC0 (UDC1) register count value at a given timing, as specified in the capture/compare register control register (CRC).

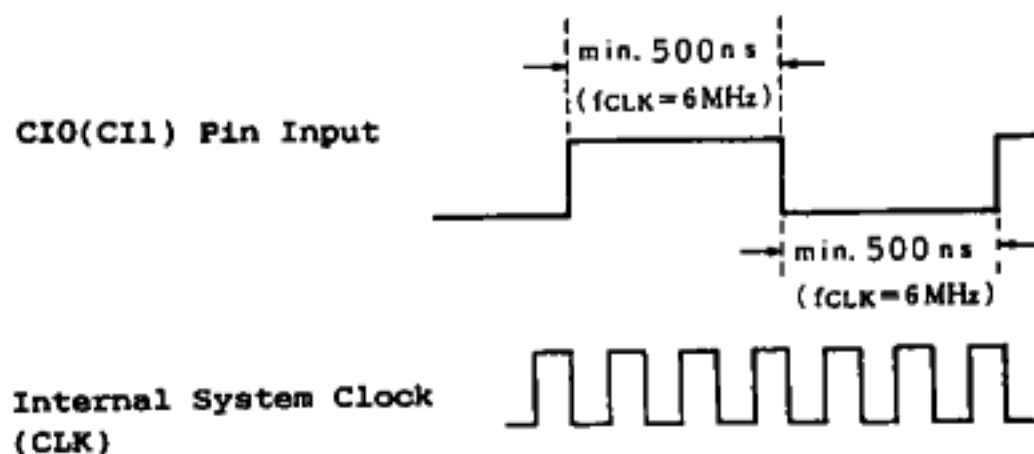
When the capture/compare register functions as a capture register, the CR00 or CR10 register captures the count value when rising edge is input to the external pin (CLR0 or CLR1); the CR01 or CR11 register captures the count value when the time base counter interrupt request flag (TBF) is set to (1).

When  $\overline{\text{RESET}}$  is input, the capture/compare registers are not affected.

- (c) Count unit input pins (CIO, CI1, CTRL0, CTRL1, CLR0, and CLR1)

On the CIO, CI1, CTRL0, CTRL1, CLR0, and CLR1 pins, digital noise removal is made by using internal system clock (CLK) to prevent noise from causing malfunction. To prevent signal from being removed as noise, signal having the width of three system clocks or more must be input.

Figure 4-12 Count Unit Input Pin Noise Removal Example □



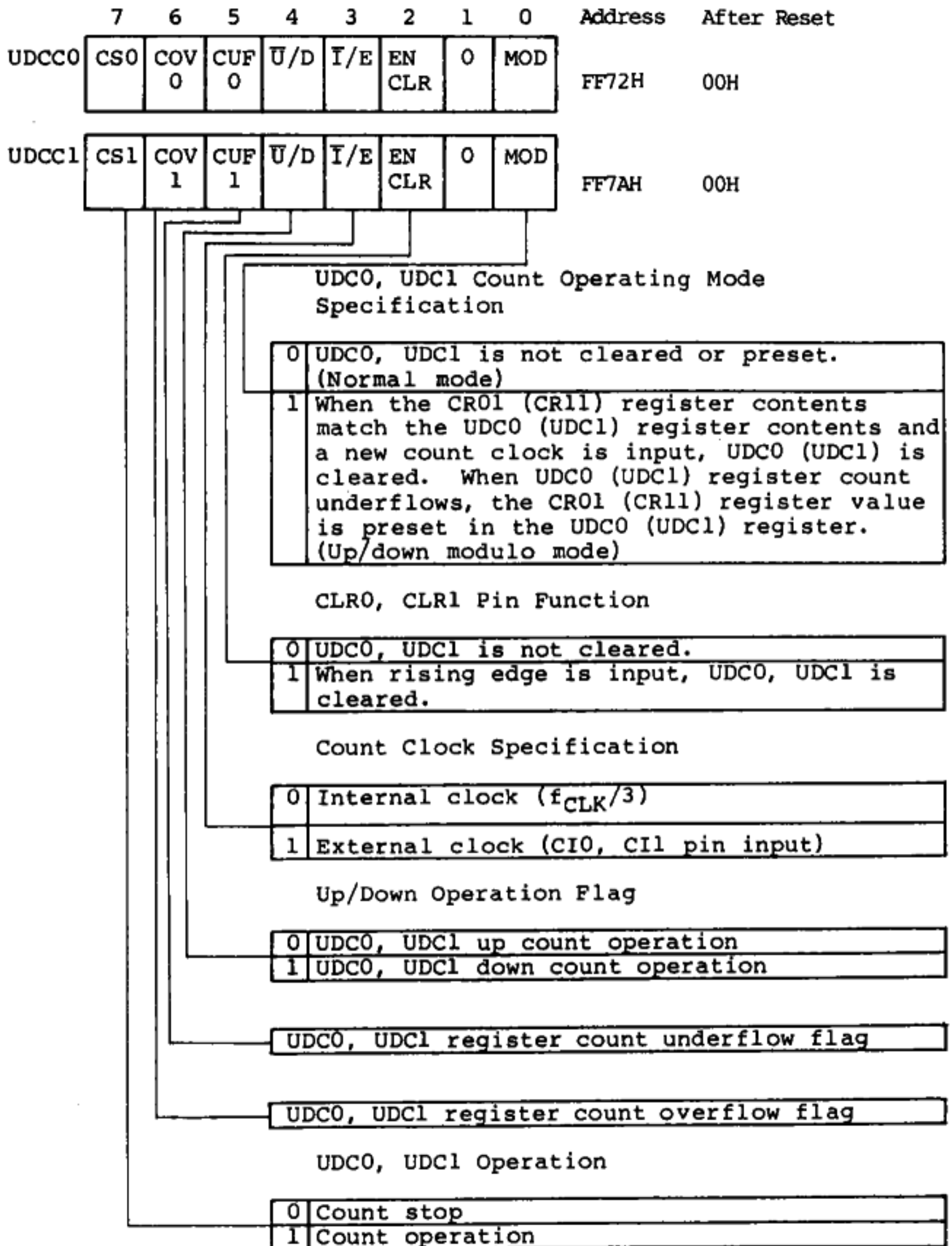
(2) Mode registers

- (a) Up/down counter control registers (UDCC0 and UDCC1)

The UDCC0 and UDCC1 registers are 8-bit registers to control UDC0 and UDC1 operating modes and count operation. Figure 4-13 shows the UDCC0 and UDCC1 register formats.

When  $\overline{\text{RESET}}$  is input, all bits of the UDCC0 and UDCC1 registers are cleared to "0".

Figure 4-13 Up/Down Counter Control Register Format



$f_{CLK}$ : Internal system clock frequency



(i) MOD bit (bit 0)

The UDC0 (UDC1) count operating mode is specified.

When the MOD bit is cleared, UDC0 (UDC1) is set to the normal mode. UDC0 (UDC1) is not cleared or preset and functions as a 16-bit binary counter.

When the MOD bit is set to (1), UDC0 (UDC1) is set to the up/down modulo mode. When the CR01 (CR11) contents match the UDC0 (UDC1) contents during up counting and a new count clock is input, UDC0 (UDC1) is cleared. When UDC0 (UDC1) contains 0000H during down counting and a new count clock is input, the CR01 (CR11) contents are preset in UDC0 (UDC1).

UDC0 (UDC1) set to the capture mode by setting the capture/compare register control register (CRC) performs capture operation regardless of how the MOD bit is set.

(ii) ENCLR bit (bit 2)

The ENCLR bit is used to control the clear function of the UDC0 (UDC1) register to 0000H according to CLR0 (CLR1) pin input.

When rising edge is input to the CLR0 (CLR1) pin with the ENCLR bit set to (1), UDC0 (UDC1) is cleared.

(iii)  $\bar{I}/E$  bit (bit 3)

The I/E bit is used to select UDC0 (UDC1) count clock. When the I/E bit is set to (0), internal clock ( $f_{CLK}/3$ ) is selected; when (1), external clock (CI0, CI1 pin input clock) is selected.

(iv)  $\overline{U/D}$  bit (bit 4)

The  $\overline{U/D}$  bit indicates the initial state of count operation. When the bit is set to (0), up count operation is indicated; when (1), down count operation is indicated. The count operation specified in the  $\overline{U/D}$  bit is performed unless edge input (CTRL0, CTRL1 pin input) specified in the count unit input mode register (CUIM) occurs or the  $\overline{U/D}$  bit is inverted by software.

When the counter counts up, the  $\overline{U/D}$  bit is reset to (0); when the counter counts down, the bit is set to (1). The up or down count operation can be detected by software which decides whether the  $\overline{U/D}$  bit is set or reset.

(v) CUFO, CUF1 bit (bit 5)

The CUFO, CUF1 bit indicates a count underflow. In the normal mode, the CUFO (CUF1) bit is set to (1) when UDC0 (UDC1) counts down to FFFFH from 0000H. In the up/down modulo mode, the bit is set to (1) when the CR01 (CR11) register value is preset in UDC0 (UDC1).

(vi) COV0, COV1 bit (bit6)

The COV0, COV1 bit indicates a count overflow. In the normal mode, the COV0 (COV1) bit is set to (1) when UDC0 (UDC1) counts up to 0000H from FFFFH. In the up/down modulo mode, the bit is set to (1) when the CR01 (CR11) register value matches the UDC0 (UDC1) value.

(vii) CS0, CS1 bit (bit 7)

The CS0 (CS1) bit is used to control UDC0 (UDC1) count operation. When the bit is set to (1), count operation is started. When the bit is reset to (0), count operation is stopped.

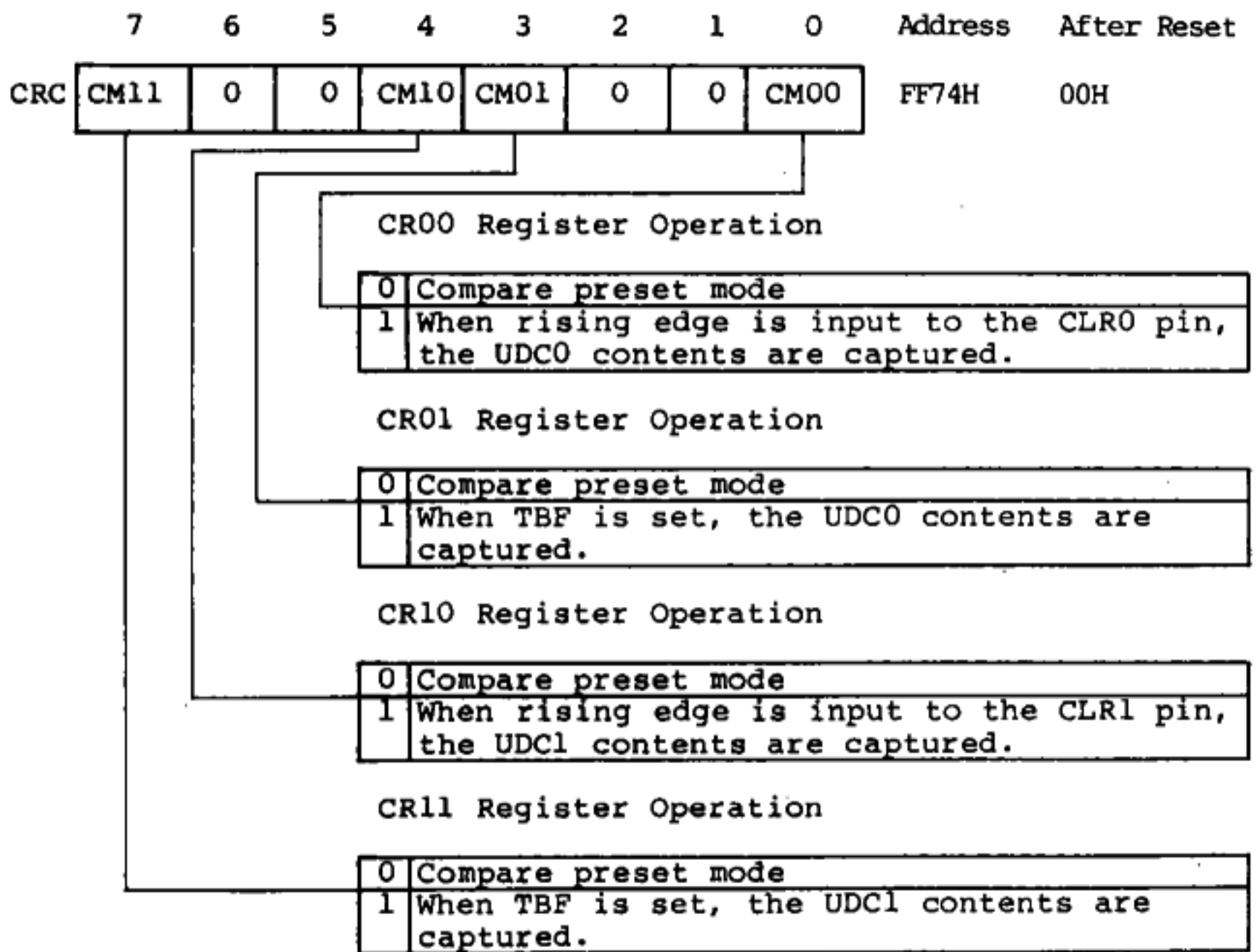
(b) Capture/compare register control register (CRC)

The CRC register is an 8-bit register to control operation of the capture/compare registers (CR00, CR01, CR10, and CR11).

When a CRC register bit is cleared, the capture/compare register corresponding to the bit is set to the compare preset mode. When a CRC register bit is set to (1), the capture/compare register corresponding to the bit is set to the capture mode.

When **RESET** is input, all the CRC register bits are cleared.

Figure 4-14 Capture/Compare Register Control Register Format



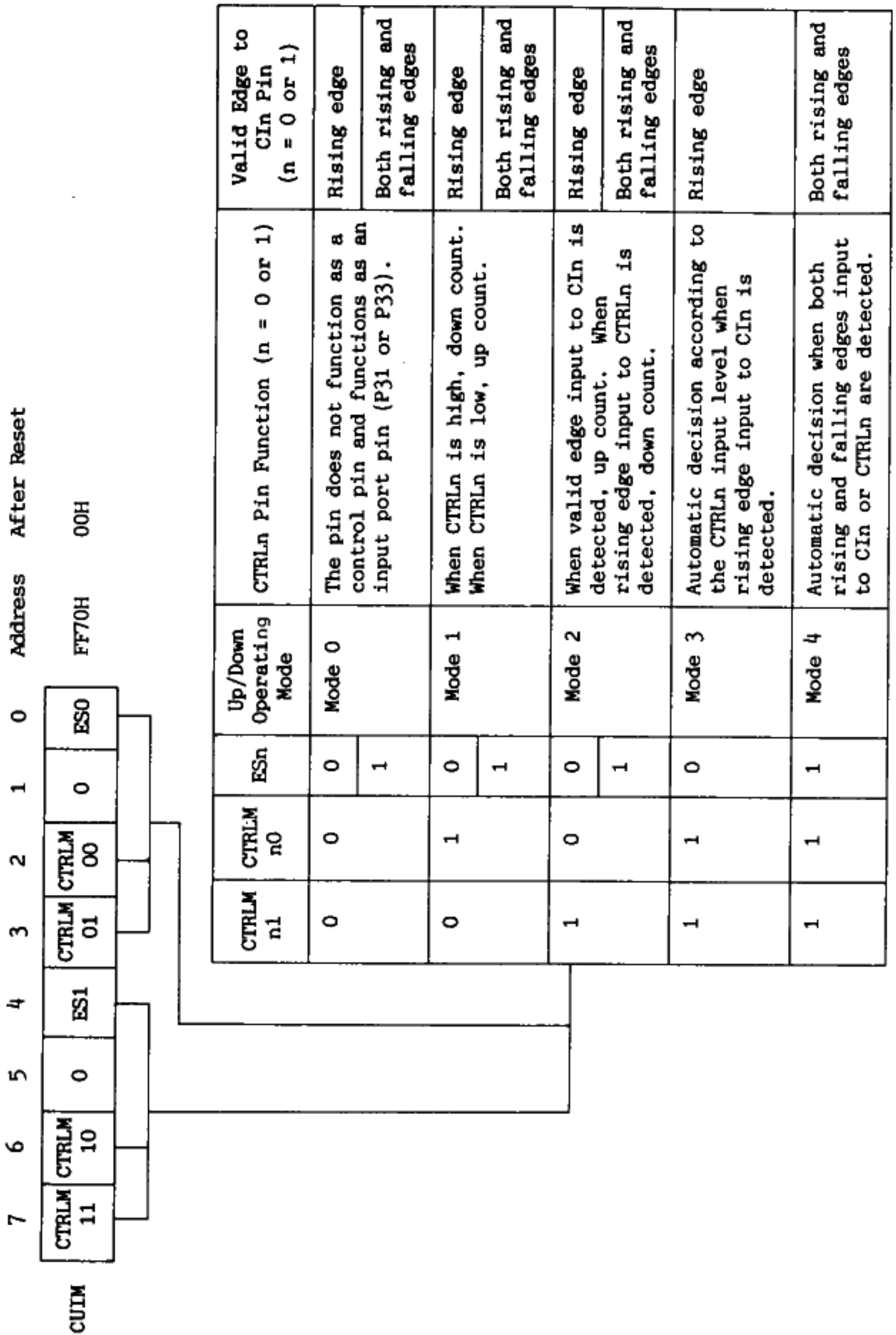
TBF: Time base counter interrupt request flag

(c) Count unit input mode register (CUIM)

The CUIM register is an 8-bit register to control the function of the count unit input pins (CIO, CI1, CTRL0, and CTRL1). Figure 4-15 shows the CUIM register format.

When RESET is input, all the CUIM register bits are cleared.

Figure 4-15 Count Unit Input Mode Register Format



### (3) Count unit operating mode

UDCO (UDC1) register count operation is set to the normal mode by resetting the UDCC0 (UDCC1) register MOD bit to (0); it is set to the up/down modulo mode by setting the MOD bit to (1).

The CRO0 (CR10), CRO1 (CR11) register operating mode is set to the compare preset mode by resetting the CRC register CM0 (CM1) bit to (0); it is set to the capture mode by setting the bit to (1).

Count clock and up or down count are selected by using the UDCC0 (UDCC1) register  $\bar{I}/E$  bit and CUIM register. (See below.)

The UDC0 (UDC1) clear function depending on CLR0 (CLR1) pin input is controlled by using the UDCC0 (UDCC1) register ENCLR bit.

When each operating mode is specified, the count unit operates as follows:

NOTE: The CRO0 (CR10) and CRO1 (CR11) registers can be set to the compare preset mode or capture mode separately by setting the CRC register.

- (a) When CRO0 (CR10), CRO1 (CR11) is set to compare preset mode

Count unit operation varies depending on whether UDC0 (UDC1) is set to the normal mode or up/down modulo mode.

Table 4-8 Operation when Compare Preset Mode is Specified

	UDC0 (UDC1) Operation	
	Normal Mode	Up/Down Modulo Mode
CR00 (CR10) operation	The value retained in CR00 (CR10) is compared with the UDC0 (UDC1) value. If a match is found, interrupt request CRF00 (CRF10) is generated when a new count clock is input (Compare operation). UDC0 (UDC1) functions as a 16-bit binary counter.	
CR01 (CR11) operation	Same as CR00 (CR10)	When UDC0 (UDC1) counts up, the value retained in CR01 (CR11) is compared with the UDC0 (UDC1) value. If a match is found, when a new count clock is input, interrupt request CRF01 (CRF11) is generated, the counter overflow flag (COV) is set, and UDC0 (UDC1) is cleared (Clear operation). When UDC0 (UDC1) counts down, if a count underflow occurs from UDC0 (UDC1), interrupt request CRF01 (CRF11) is generated, the count underflow flag (CUF) is set, and the value retained in CR01 (CR11) is preset in UDC0 (UDC1) (Preset operation).

Figure 4-16 Operation when Compare Preset Mode is Specified

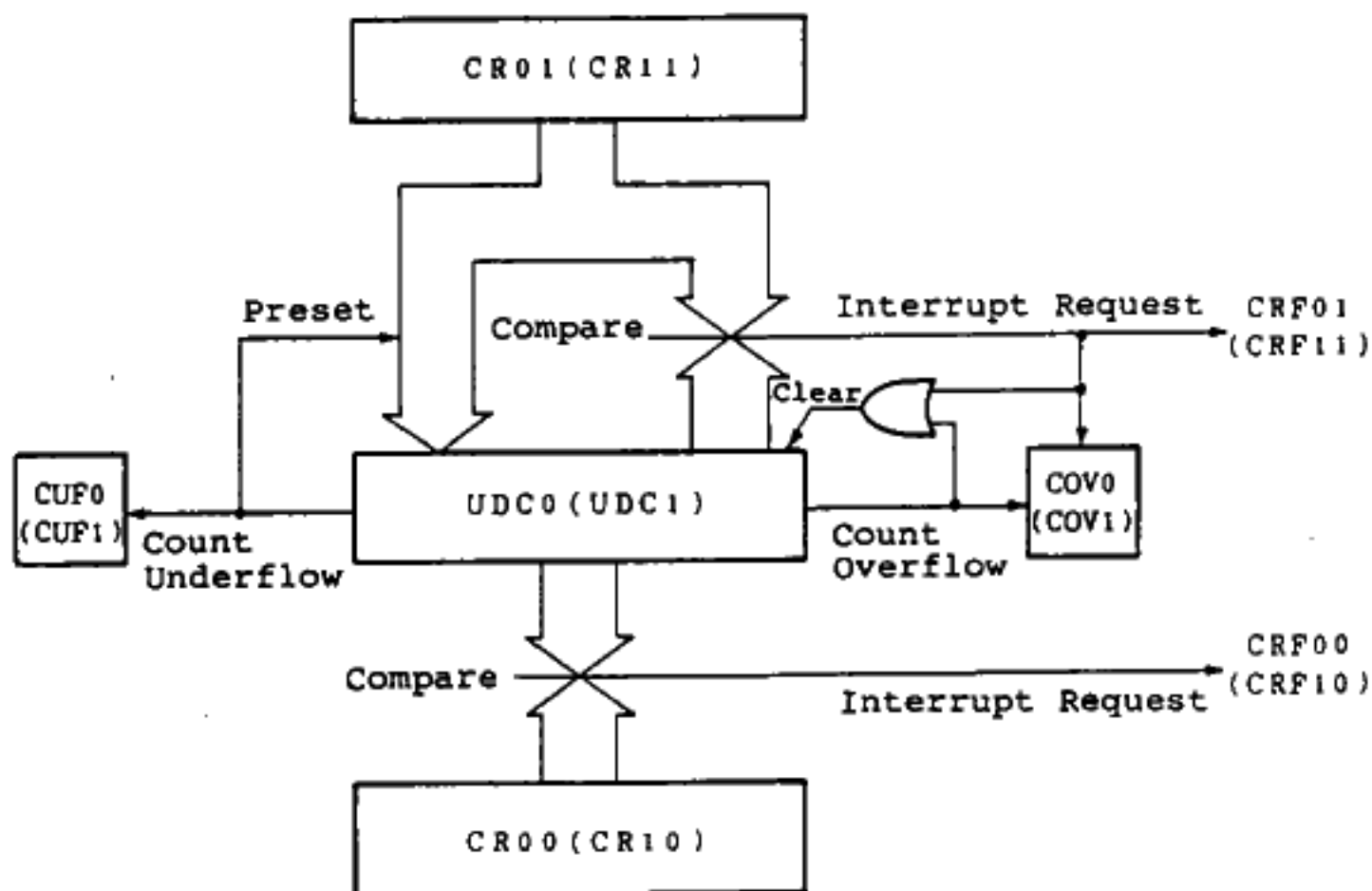
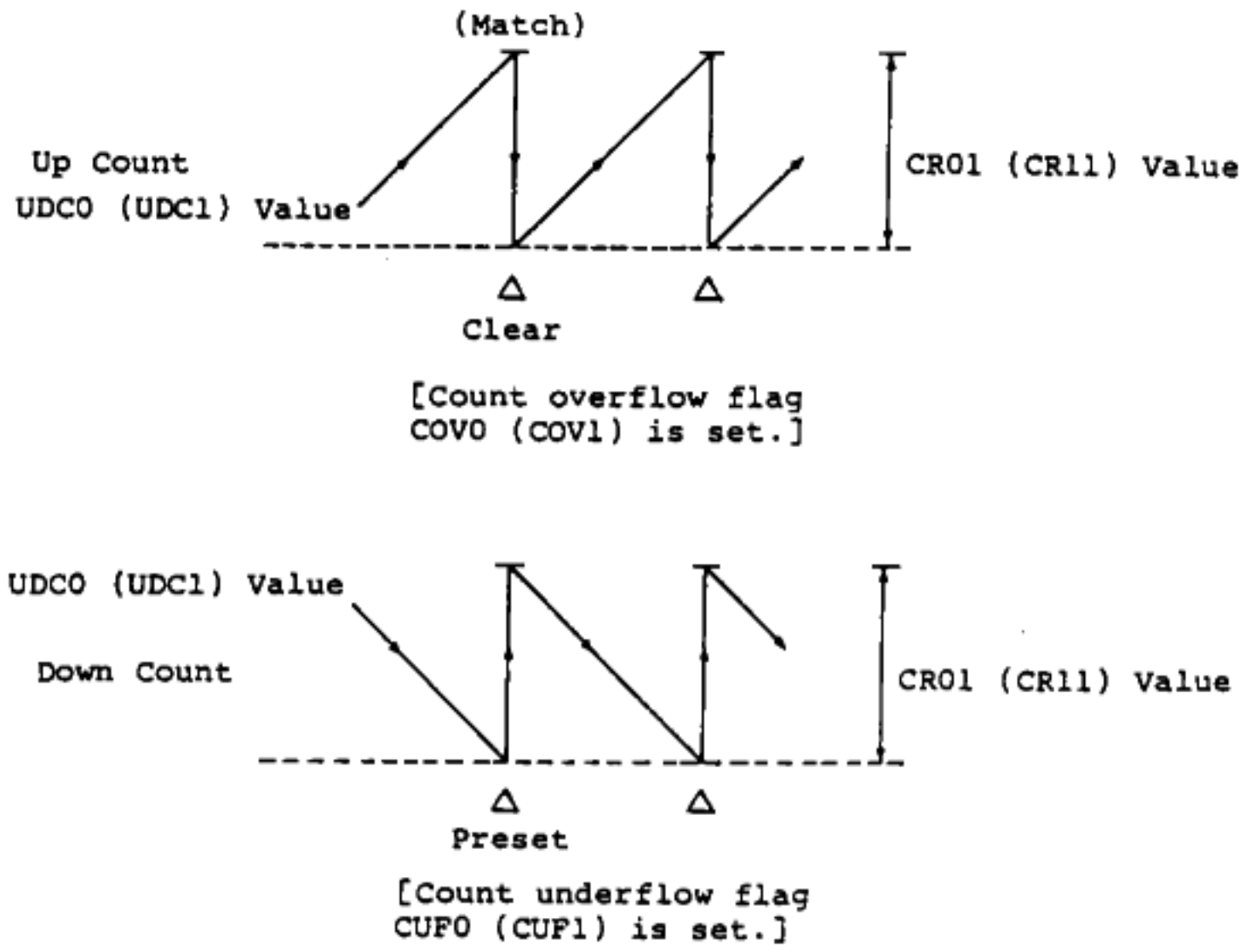


Figure 4-17 Count Operation during Up/Down Modulo Mode



- (i) When counter count operation is stopped and restored

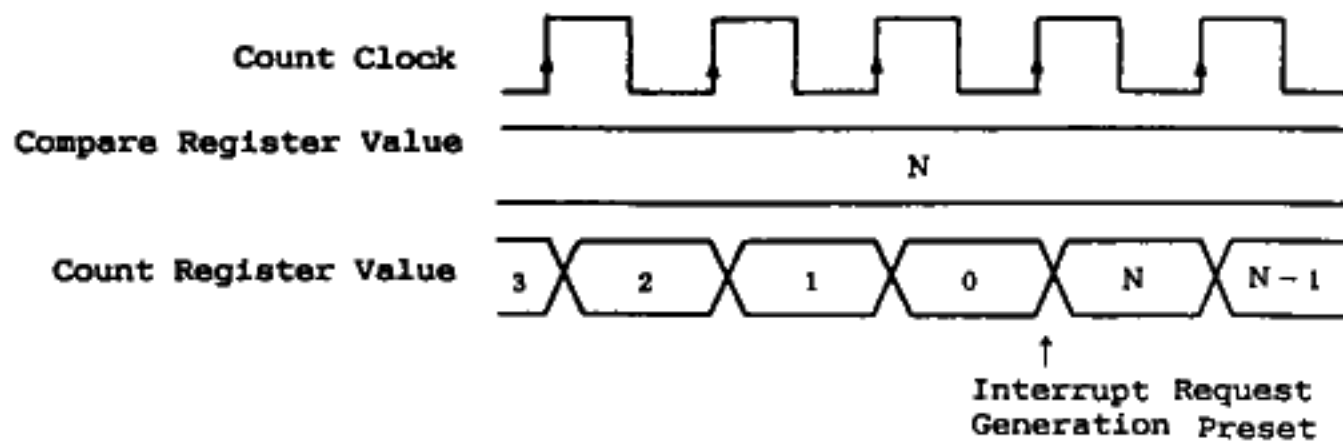
When the count operation is stopped(count disabled) after the count register value reaches the compare register value or OH, and then the count is restarted (count enable), presetting and clearing of the count register value is performed on the first count clock pulse, and an interrupt request is generated.

When the count register value is rewritten during the count disabled period, preset and clear operations are not performed, and an interrupt request is not generated.

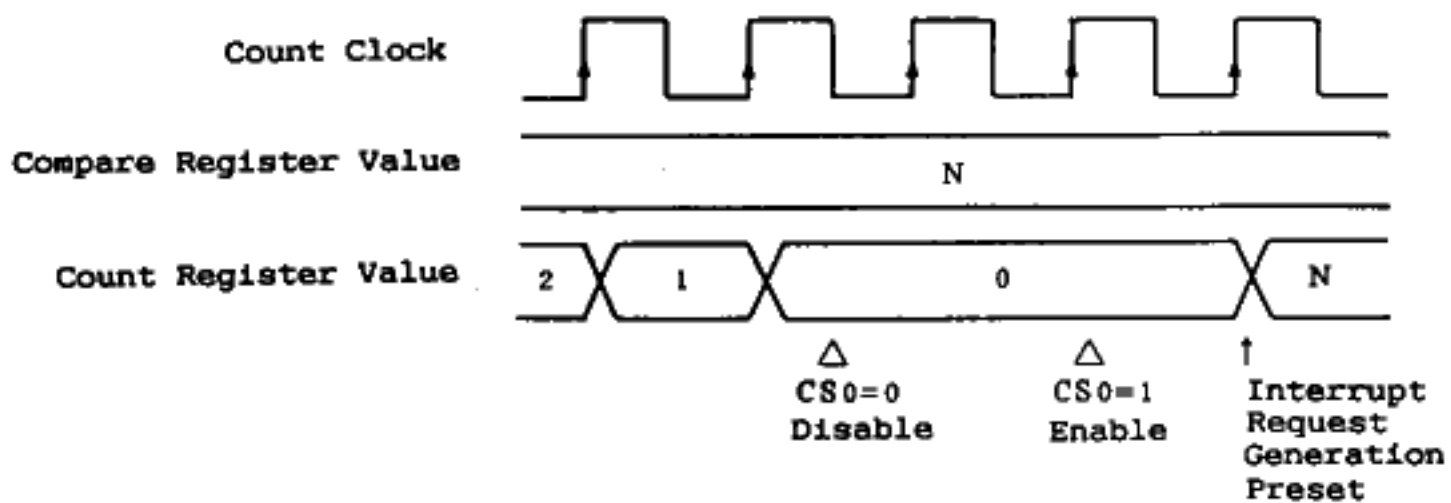


Figure 4-18 Count Operation Examples (Down Count) (1)

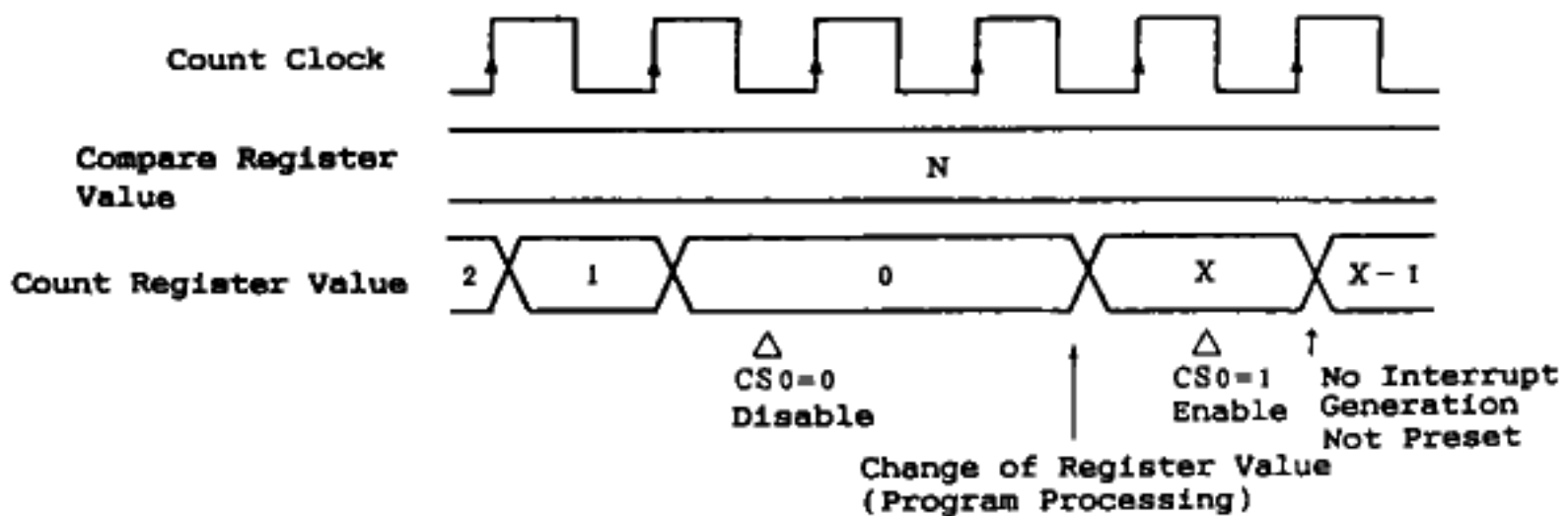
(a) In normal operation



(b) When counter count operation is stopped and restarted



(c) When count register value is rewritten during count disable



- (ii) When count register value is changed during count operation

If the count register value is changed by program processing during the count operation, count register value preset and clear operations may not be performed normally, and an interrupt request may not be generated normally. Therefore, the count register value should not be rewritten during the count operation. Next, a concrete example is shown below.

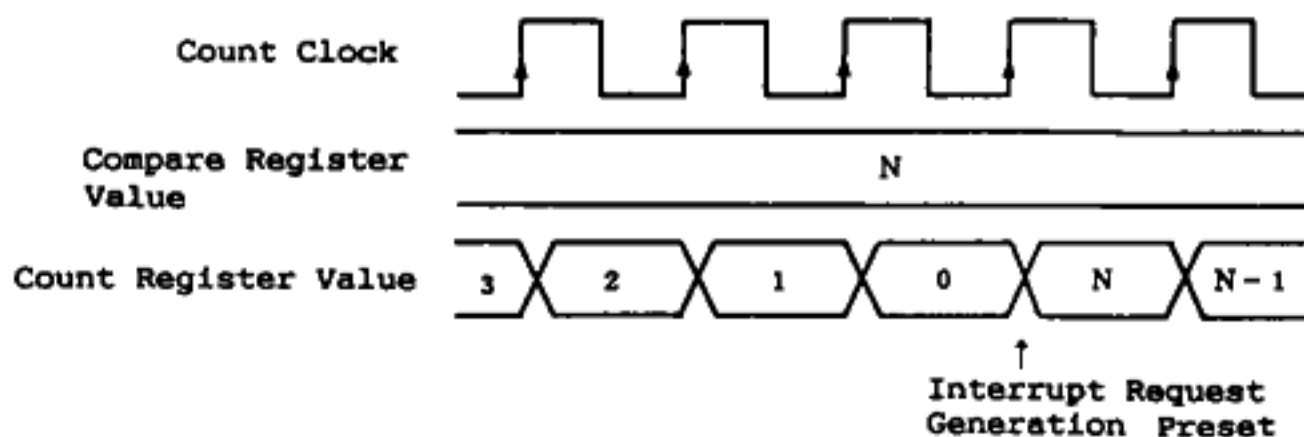
If the same value as that of the compare register, or 0H, is written to the count register by program processing, preset and clear operations are not performed on the first count clock pulse. As a result, the count register value is as follows:

- . Up-count : Compare register value + 1
- . Down-count : FFFFH

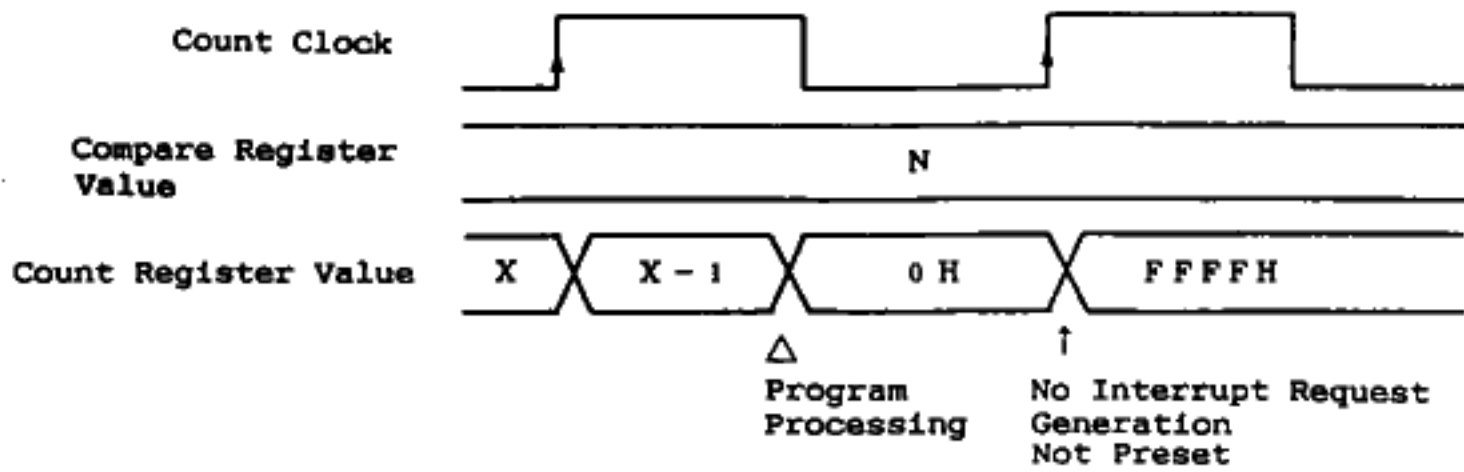
However, if the count register value is cleared by CLR0(CLR1) signal input, the preset operation is performed normally.

Figure 4-19 Count Operation Examples (Down Count) (2)

(a) In normal operation

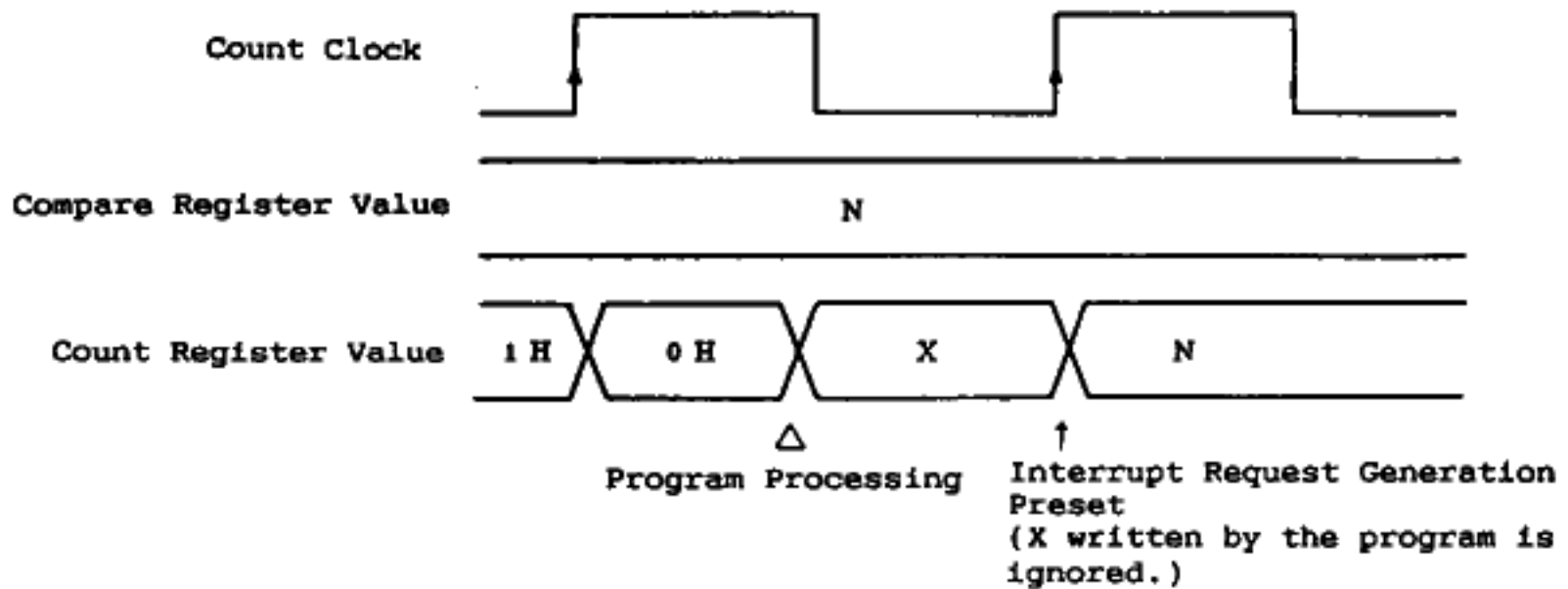


(b) When 0H is written to count register



When the count register value is changed before the next down-count after the count register reaches 0H during the count operation, the compare register value is preset by the next down-count input.

Figure 4-20 Count Operation Example (Down Count) (3)



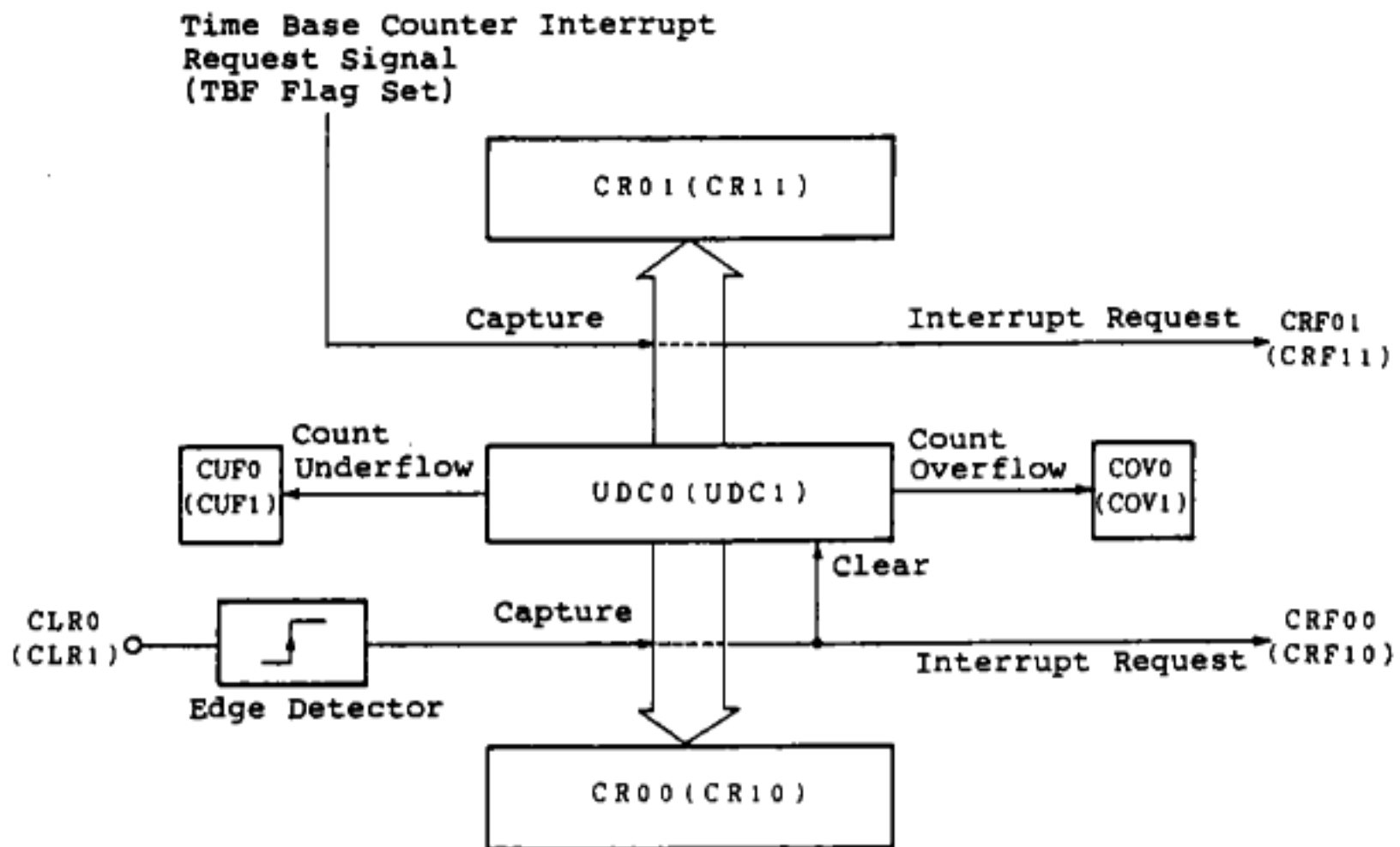
- (b) When CR00 (CR10), CR01 (CR11) is set to capture mode

UDCO (UDC1) operates in the normal mode regardless of how the MOD bit is set. Preset operation is not performed.

Table 4-9 Operation when Capture Mode is Specified

	UDCO (UDC1) Operation
CR00 (CR10) operation	<p>When rising edge is input to the CLR0 (CLR1) pin, the UDC0 (UDC1) value is captured in CR00 (CR10) and interrupt request CRF00 (CRF10) is generated (Capture operation). If UDCC0 (UDCC1) bit 2 (ENCLR) is set to (1) at the time, UDC0 (UDC1) is cleared after capture operation is performed.</p> <p>When CR00 (CR10) is set to the capture mode, the CLR0/P36 (CLR1/P37) pin functions as a capture trigger input pin.</p>
CR01 (CR11) operation	<p>When a time base counter interrupt request is generated (TBF is set), the UDC0 (UDC1) value is captured in CR01 (CR11) and interrupt request CRF01 (CRF11) is generated (Capture operation).</p> <p>The UDC0 (UDC1) clear function after capture operation is performed is not contained.</p>

Figure 4-21 Operation when Capture Mode is Specified



(4) Up/down count change control by using count unit input pins

Up/down count change operation during UDC0 (UDC1) register counting is controlled according to input to the count unit input pins.

One of change operating modes 0 to 4 is selected by setting the count unit input mode register (CUIM) ES, CTRLM0, and CTRLM1 bits.

(a) Mode 0 (CTRLM1 = 0 and CTRLM0 = 0)

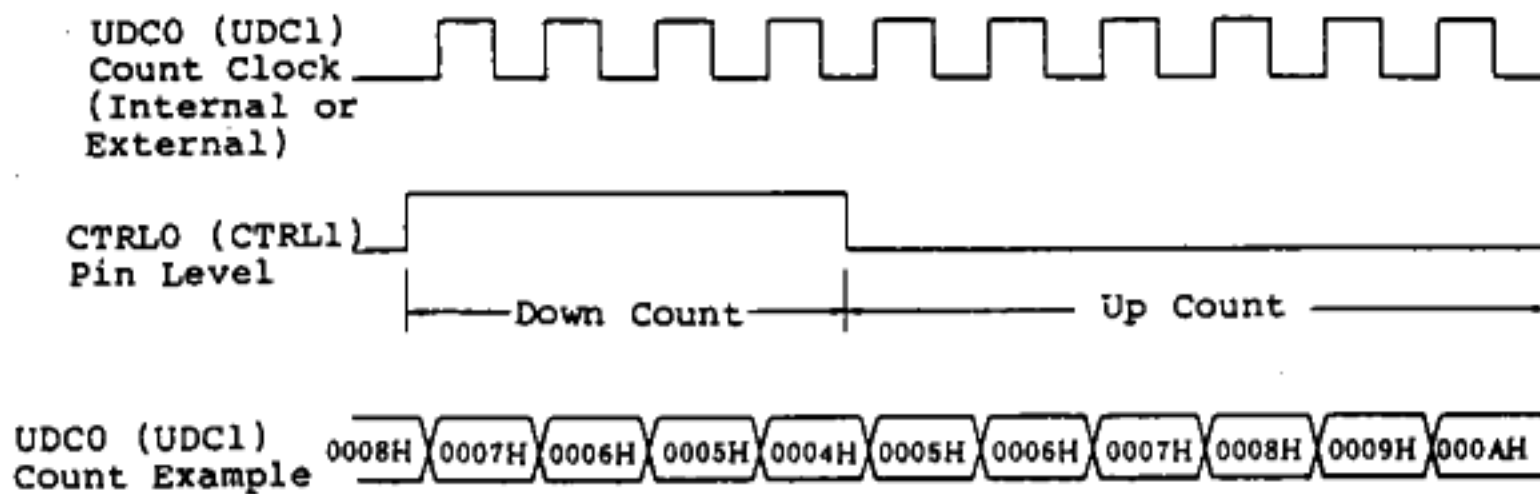
In mode 0, the CTRL0 (CTRL1) pin functions as general purpose input port pin (P33, P31). UDC0 (UDC1) count operation is not affected.

(b) Mode 1 (CTRLM1 = 0 and CTRLM0 = 1)

In mode 1, when the CTRL0 (CTRL1) pin is high, count clocks are counted down; when the pin is low, count clocks are counted up.

The count clock is selected by using UDCC0 (UDCC1) register bit 3 (I/E).

Figure 4-22 Mode 1 Operation



(c) Mode 2 (CTRLM1 = 1 and CTRLM0 = 0)

In mode 2, operation varies depending on which count clock internal or external clock is selected.

(1) When internal clock is selected

When valid edge specified in the ES bit is input to the CI0 (CI1) pin, the up/down counter is set to the count up operating mode and counts up internal clocks.

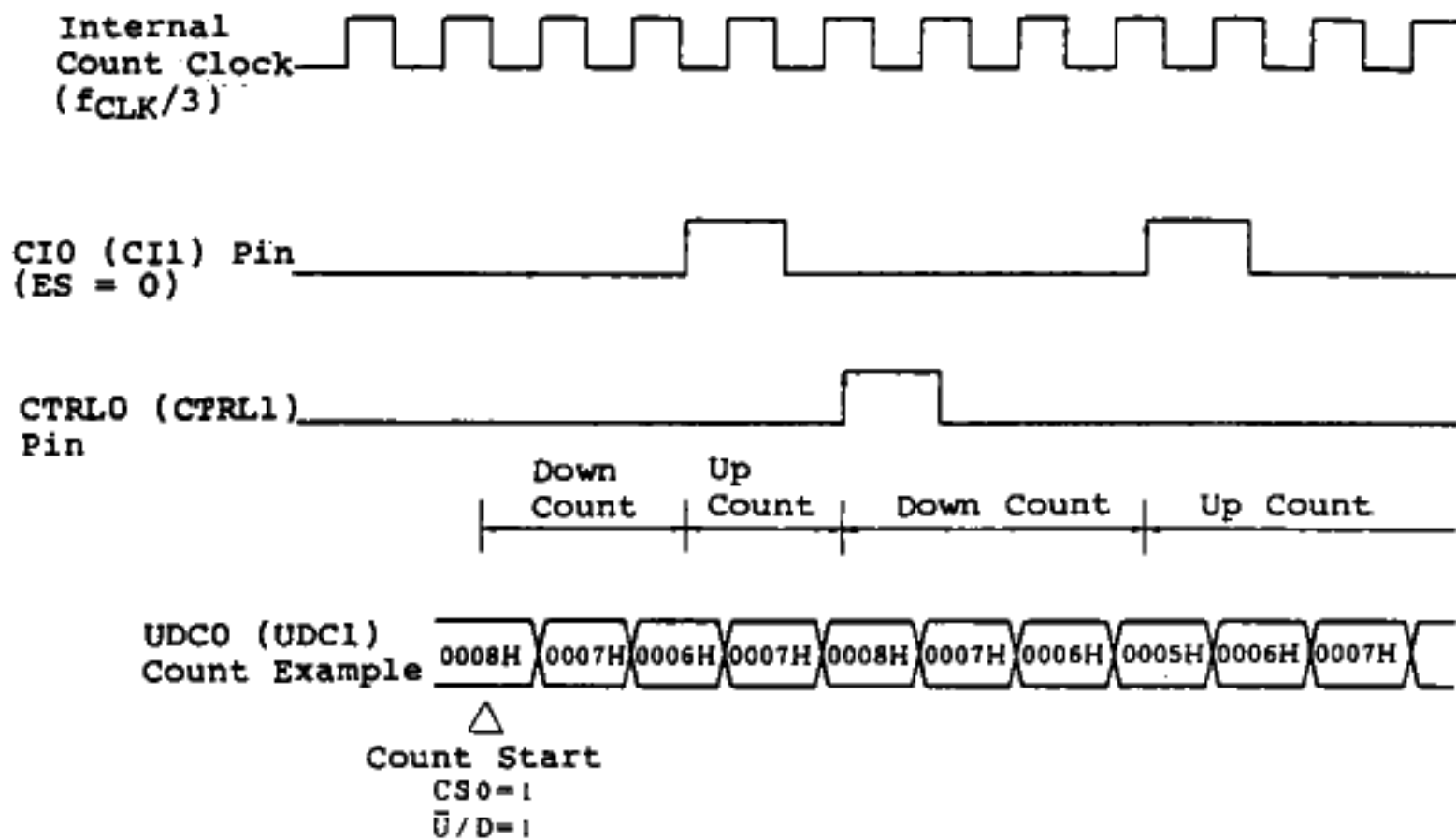
When rising edge is input to the CTRL0 (CTRL1) pin, the up/down counter is set to the count down operating mode and counts down internal clocks.

If mode 2 is set by using the count unit input mode register (CUIM) and the up/down counter control register CS bit is set to (1), the up/down counter counts up or down clocks in the mode specified in the up/down counter control register (UDCC0, UDCC1).

After this, when valid edge is input to the CI0 (CI1) pin or CTRL0 (CTRL1) pin, count operation is automatically changed to up or down operation.

**Example:** When the count unit input mode register (CUIM) ES bit is reset to (0) and rising edge is selected for the valid edge input to the CI0 (CI1) pin, up/down change operation is shown (Figure 4-23).

Figure 4-23 Mode 2 Operation when Internal Clock is Selected for Count Clock



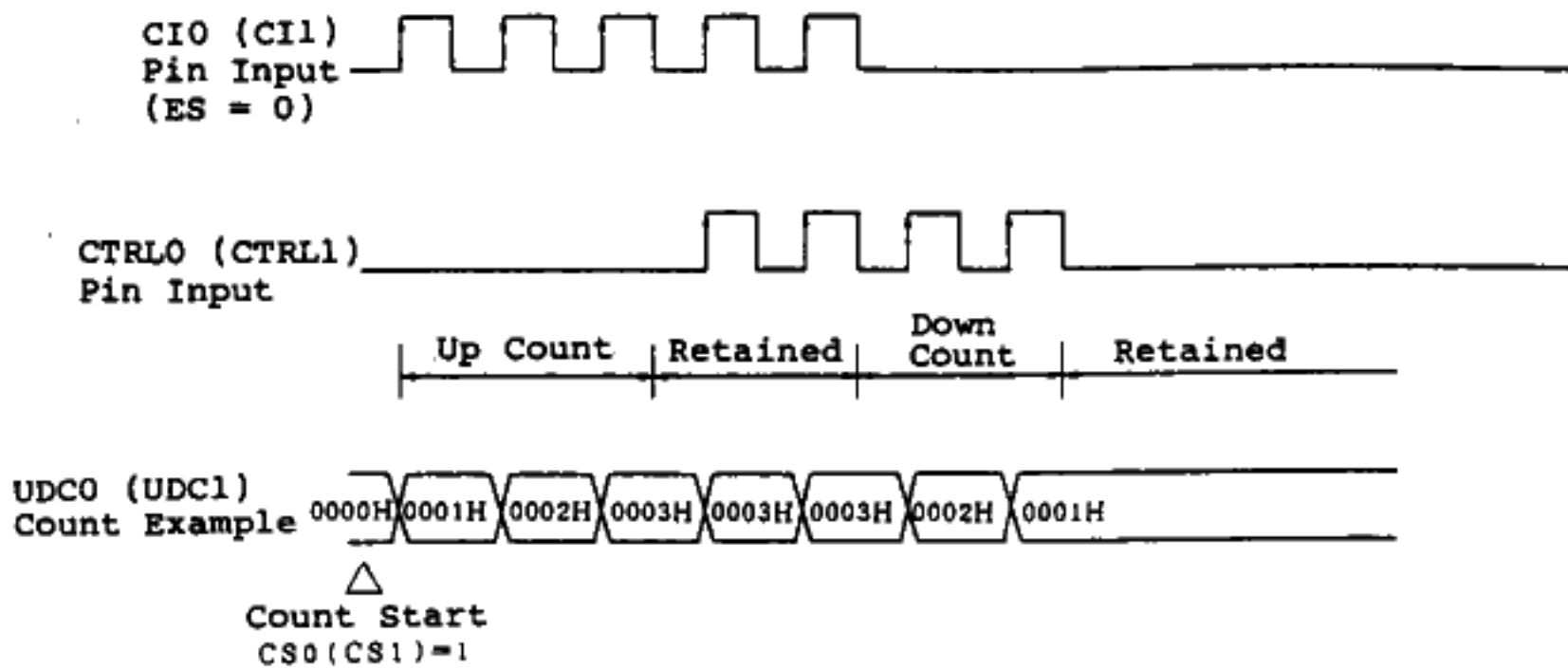
(ii) When external clock is selected

Valid edges (specified in the ES bit) input to the CIO (CI1) pin are counted up and rising edges input to the CTRL0 (CTRL1) pin are counted down.

This mode is set by using the count input mode register (CUIM). The up/down counter starts counting in the mode specified in the up/down counter control register. After this, count clocks input to the CIO (CI1) pin are counted up and count clocks input to the CTRL0 (CTRL1) pin are counted down. If a count clock is input to the CIO (CI1) pin and CTRL0 (CTRL1) pin at the same time, count operation is not performed and the immediately preceding count value is retained.

**Example:** When the count unit input mode register (CUIM) ES bit is reset to (0) and the rising edge is selected for the valid edge input to the CIO pin, up/down counter operation is shown (Figure 4-24).

Figure 4-24 Mode 2 Operation when External Clock is Selected for Count Clock



(d) Mode 3 (ES = 0, CTRLM1 = 1, and CTRLM0 = 1)

Mode 3 is the most useful mode when 2-phase signal with phase shifted  $90^\circ$  like servo motor shaft encoder output is input to the CIO (CI1) pin and CTRL0 (CTRL1) pin as count clock.

The count unit detects relative phase lead or lag of the 2-phase signal and automatically changes up/down counter up/down count operation.

When 2-phase signal having  $90^\circ$  phase difference is input to the CIO (CI1) pin and CTRL0 (CTRL1) pin, the CTRL0 (CTRL1) pin level is sampled when rising edge is input to the CIO (CI1) pin.

When the CTRL0 (CTRL1) pin level sampled on the rising edge input to the CIO (CI1) pin is low, the up/down counter counts down when rising edge is input to the CIO (CI1) pin.

The CTRL0 (CTRL1) pin level sampled on the rising edge input to the CIO (CI1) pin is high, the up/down counter counts up when rising edge is input to the CIO (CI1) pin.



Figure 4-25 Mode 3 Down Count Operation

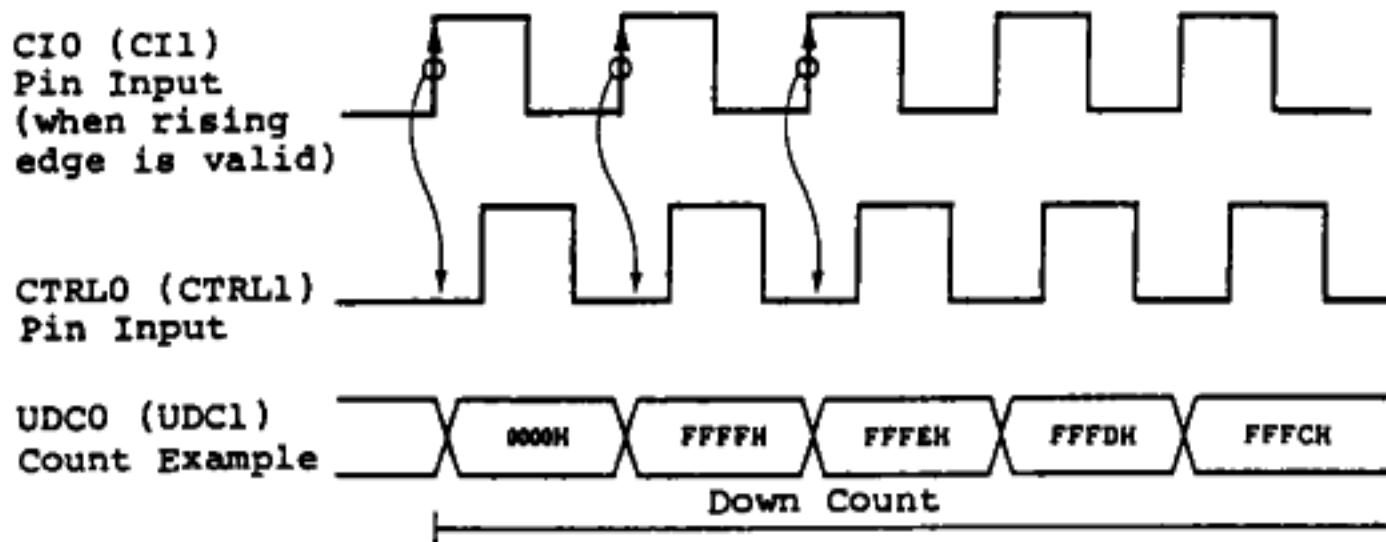
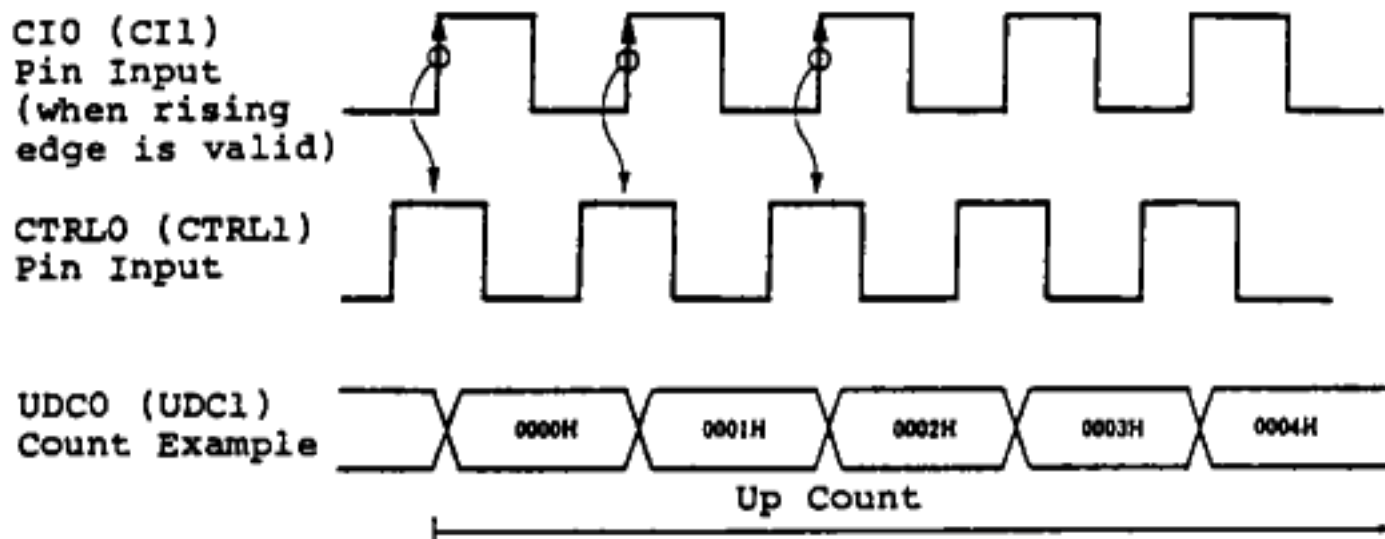


Figure 4-26 Mode 3 Up Count Operation



(e) Mode 4 (ES = 1, CTRLM = 1, and CTRLMO = 1)

Like mode 3, mode 4 is useful when 2-phase signals with phase shifted  $90^\circ$  like servo motor shaft encoder output are counted as count clocks.

When 2-phase signal with phase shifted  $90^\circ$  is input to the CIO (CI1) pin and CTRL0 (CTRL1) pin, automatically up or down count operation is decided and count is made at the timing as shown in Figure 4-27.

In mode 4, both the rising edge and falling edges of each 2-phase signal input to the CIO (CI1) pin and CTRL0 (CTRL1) pin are counted. Thus, the up/down counter counts four per input signal cycle (4 times count).



Figure 4-28 Count Unit Interrupt Request Control Register Formats

	7	6	5	4	3	2	1	0	Address	After Reset
CRIC00	CRF 00	CRMK 00	CRISM 00	CRCSE 00	0	PR2	PR1	PRO	FFC0H	47H
CRIC01	CRF 01	CRMK 01	0	CRCSE 01	0	-	-	-	FFC2H	47H
CRIC10	CRF 10	CRMK 10	CRISM 10	CRCSE 10	0	-	-	-	FFC4H	47H
CRIC11	CRF 11	CRMK 11	0	CRCSE 11	0	-	-	-	FFC6H	47H

Remarks: "-" denotes that write cannot be made. If the bit is read, "1" is read.

Table 4-10 Count Unit Interrupt Request Flag Set Conditions

Inter-rupt Request Flag	Operat-ing Mode	Up/Down Counter Operat-ing Mode	Count Opera-tion	Interrupt Request Flag Set Condition	
CRF00	Capture	-	-	When the UDCO value is captured in CRO0 when rising edge is input to the CLRO pin	
	Compare preset	-	-	When the CRO0 contents match the UDCO value and a new count clock is input	
CRF01	Capture	-	-	When the UDCO value is captured in CRO1 when a time base interrupt request occurs (TBF is set)	
	Compare preset	Normal mode	-	When the CRO1 contents match the UDCO value and a new count clock is input	
		Up/down modulo mode	Down count		When the CRO1 contents are preset in UDCO
			Up count		When the CRO1 contents match the UDCO value and a new count clock is input (After this, the UDCO value is cleared.)

(to be continued)

**Table 4-10 Count Unit Interrupt Request Flag Set Conditions (cont'd)**

Interrupt Request Flag	Operating Mode	Up/Down Counter Operating Mode	Count Operation	Interrupt Request Flag Set Condition	
CRF10	Capture	-	-	When the UDC1 value is captured in CR10 when rising edge is input to the CLR1 pin	
	Compare preset	-	-	When the CR10 contents match the UDC1 value and a new count clock is input	
CRF11	Capture	-	-	When the UDC1 value is captured in CR11 when a time base interrupt request occurs (TBF is set)	
	Compare preset	Normal mode	-	When the CR11 contents match the UDC1 value and a new count clock is input	
		Up/down modulo mode	Down count		When the CR11 contents are preset in UDC1
			Up count		When the CR11 contents match the UDC1 value and a new count clock is input (After this, the UDC1 value is cleared.)

**(6) Count unit macro service control registers (CRMS00 and CRMS10)**

CRF00 and CRF10 of the four interrupt requests occurring from the count unit can cause macro service request.

CRMS00 is an 8-bit register to control macro service started when CRF00 is set to (1); CRMS10 is an 8-bit register to control macro service started when CRF10 is set to (1). The function of the bits is described in "5.2 MACRO SERVICE FUNCTION".

Figure 4-29 Macro Service Control Register Formats

	7	6	5	4	3	2	1	0	Address	After Reset
CRMS00	MSM 2	MSM 1	MSM 0	DIR	0	CH2	CH1	CH0	FFC1H	Undefined
CRMS10	MSM 2	MSM 1	MSM 0	DIR	0	CH2	CH1	CH0	FFC5H	Undefined

#### 4.3.2 CAPTURE UNIT

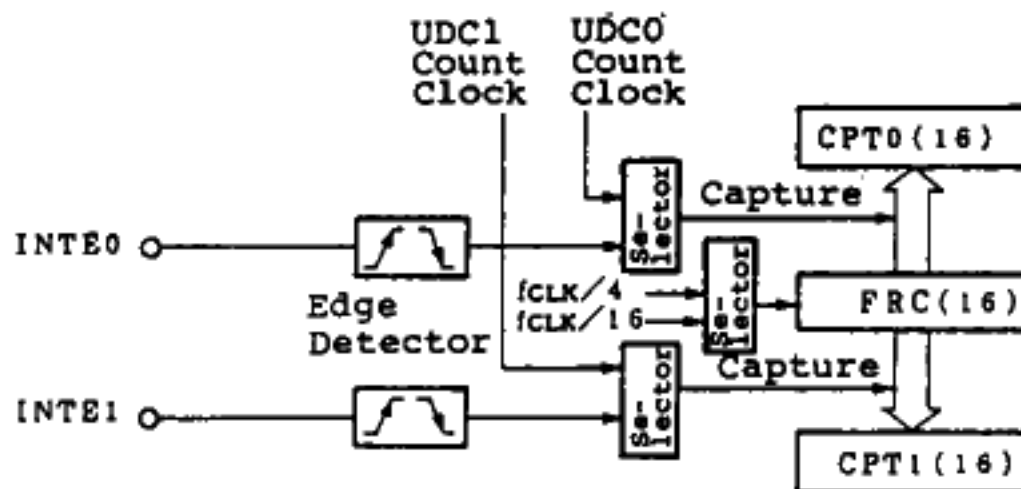
##### (1) Capture unit configuration

The capture unit has the free running counter capture function. Figure 4-30 shows the capture unit block diagram.

The free running counter (FRC) is also used for a 20-bit time base counter which counts internal system clock ( $f_{CLK}$ ); it is cleared to 00H only when  $\overline{RESET}$  is input.

The capture registers (CPT0 and CPT1) function as the FRC capture registers. When a count clock is input to the count unit or a valid edge is input to the INTE0 (INTE1) pin, the FRC count value is captured in the CPT0 (CPT1) register. When  $\overline{RESET}$  is input, the unit becomes undefined.

Figure 4-30 Capture Unit Block Diagram

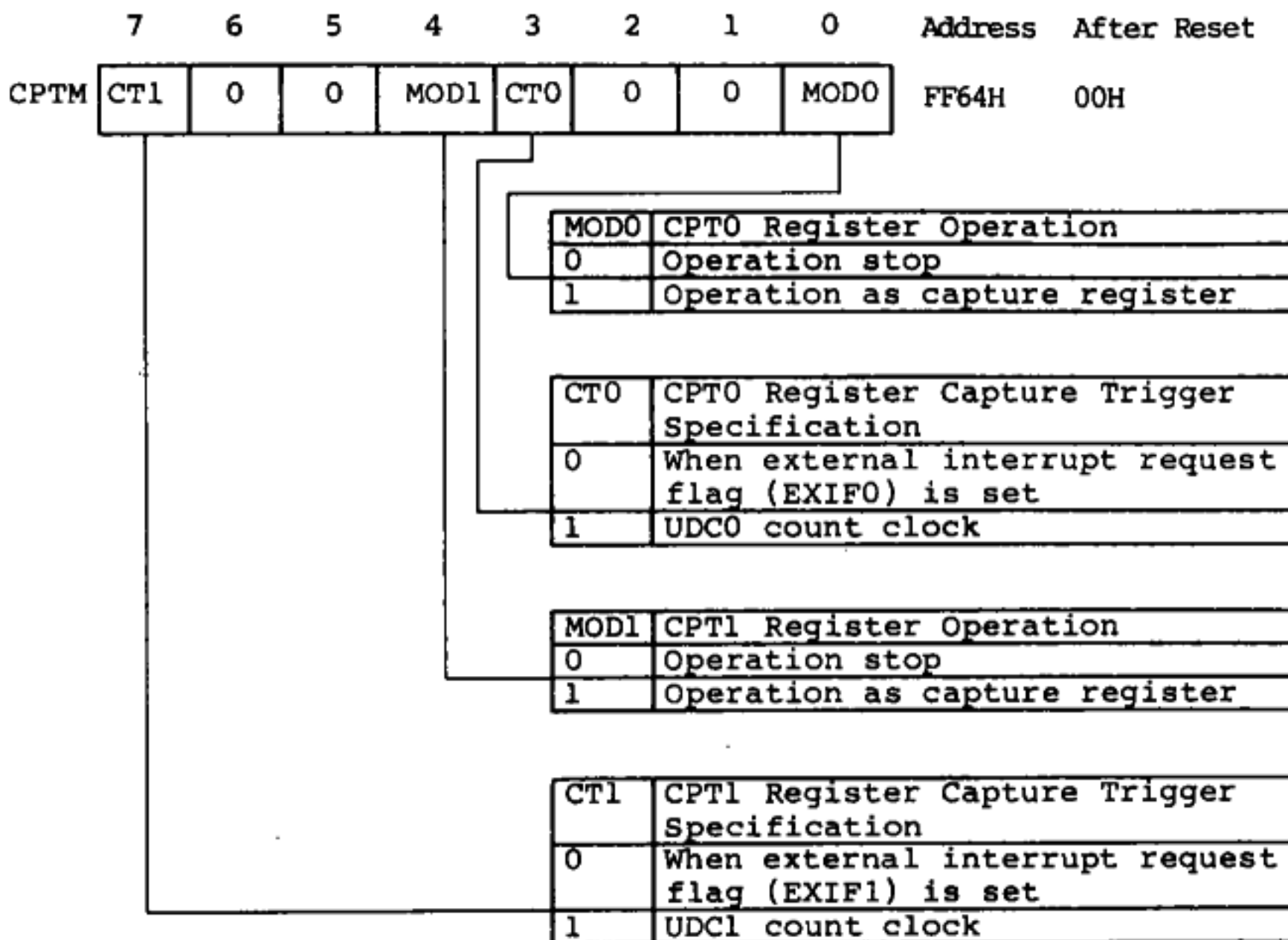


Remarks:  $f_{CLK}$ : Internal system clock frequency

(2) Capture mode register (CPTM)

The capture mode register (CPTM) is an 8-bit register to specify the capture register (CPT0, CPT1) operating mode and capture trigger. Figure 4-31 shows the capture mode register (CPTM) format. When RESET is input, all the CPTM bits are cleared to "0".

Figure 4-31 Capture Mode Register Format

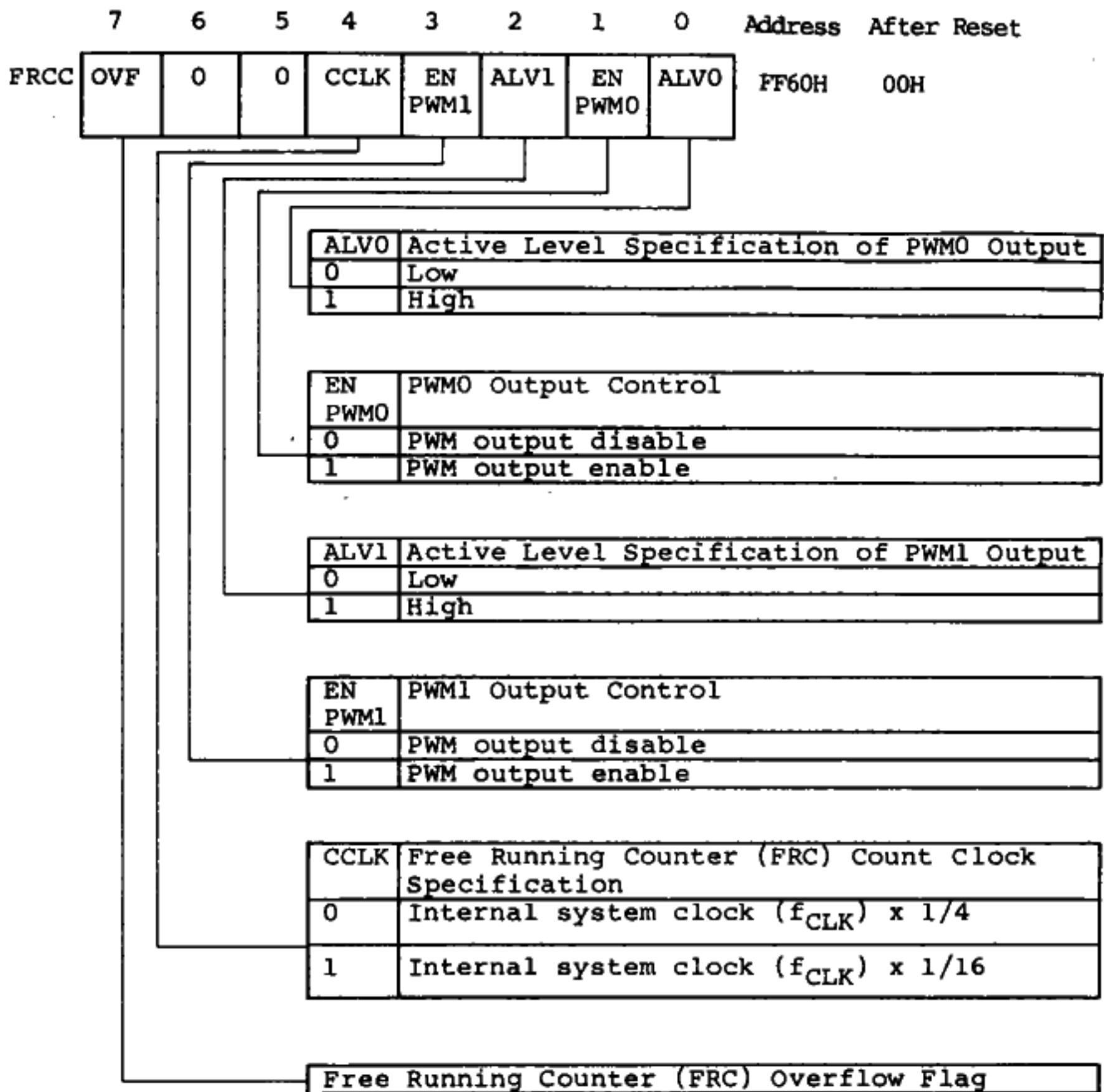


(3) Free running counter control register (FRCC)

The FRCC register is an 8-bit register to control the free running counter (FRC) and PWM output. Figure 4-32 shows the FRCC register format.

When RESET is input, all the FRCC register bits are cleared to "0".

Figure 4-32 Free Running Counter Control Register Format



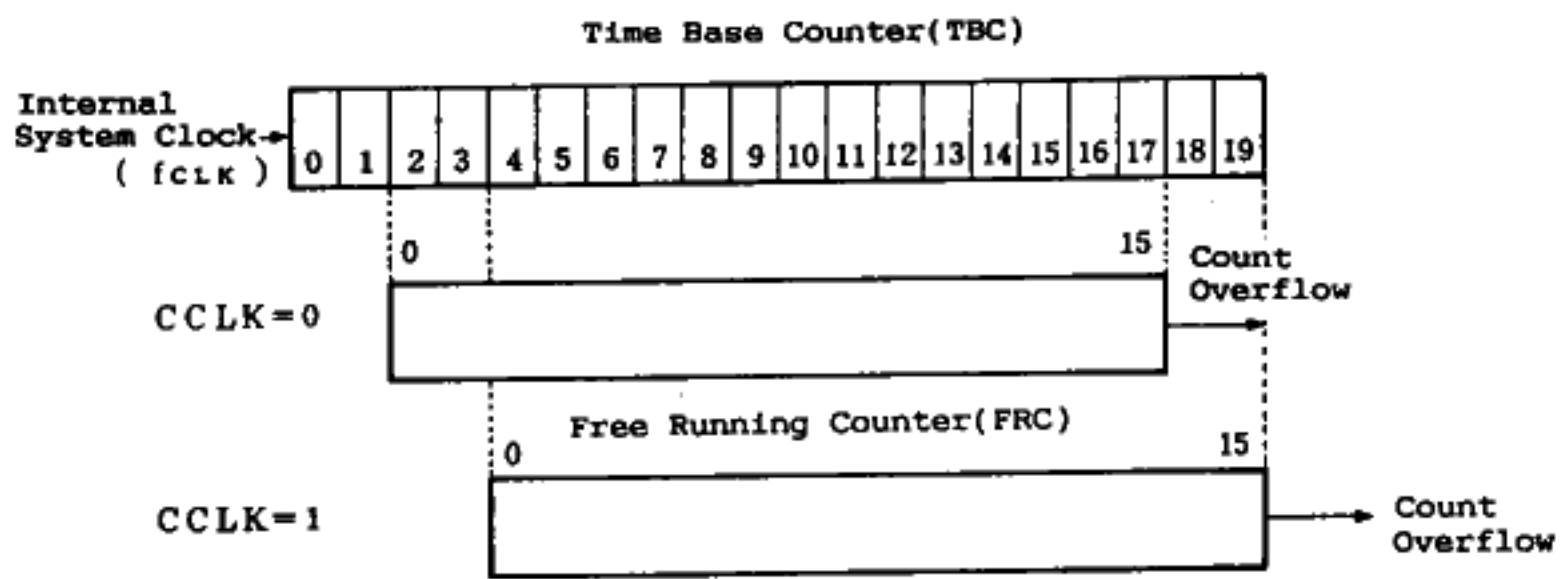
(4) Capture unit operation

When the trigger specified in the CPTM register CT0 (CT1) bit capture occurs, 16-bit data in the free running counter (FRC) is captured in the CPT0 (CPT1) register.

The captured 16-bit free running counter (FRC) part of the 20-bit time base counter (TBC) is specified by using free running counter control register (FRCC) bit 4 (CCLK). The 16-bit data captured in the capture register (CPT0, CPT1) is changed depending on whether the CCLK bit is set to (1) or reset to (0), as shown in Figure 4-33.

On software, free running counter (FRC) count clock appears to be changed.

Figure 4-33 Free Running Counter Specification



FRCC bit 7 (OVF) indicates an FRC overflow. If an overflow from TBC bit 17 (when CCLK = 0) or from TBC bit 19 (when CCLK = 1) occurs, the OVF bit is set to (1). Once OVF bit is set to (1), the bit is not reset to (0) by hardware even if the CCLK bit setting is changed. The OVF bit is always reset to (0) only by software.

If the external interrupt flag (EXIF0, EXIF1) is selected for the trigger of capturing data in the capture register, capture operation is performed each time a valid edge is input to the external interrupt pin (INTE0, INTE1).

If the up/down counter count clock is selected, count clock input to UDC0 (UDC1) is used as the capture trigger as it is independently of count clock source (internal or external) selection or count operation (up or down count).

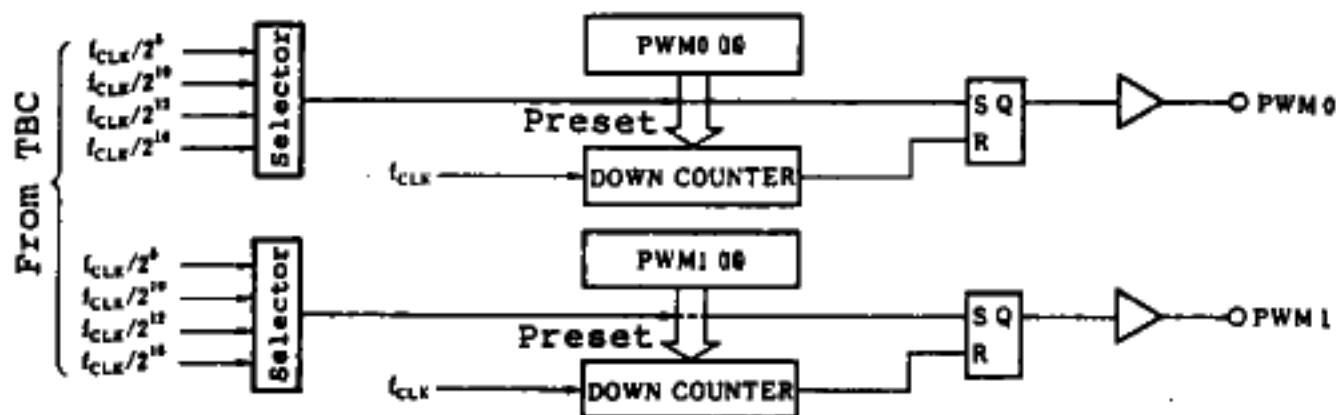


### 4.3.3 PWM UNIT

#### (1) PWM unit configuration

The PWM unit has the PWM output function. Figure 4-34 shows the PWM unit block diagram.

Figure 4-34 PWM Unit Block Diagram



Remarks:  $f_{CLK}$ : Internal system clock frequency

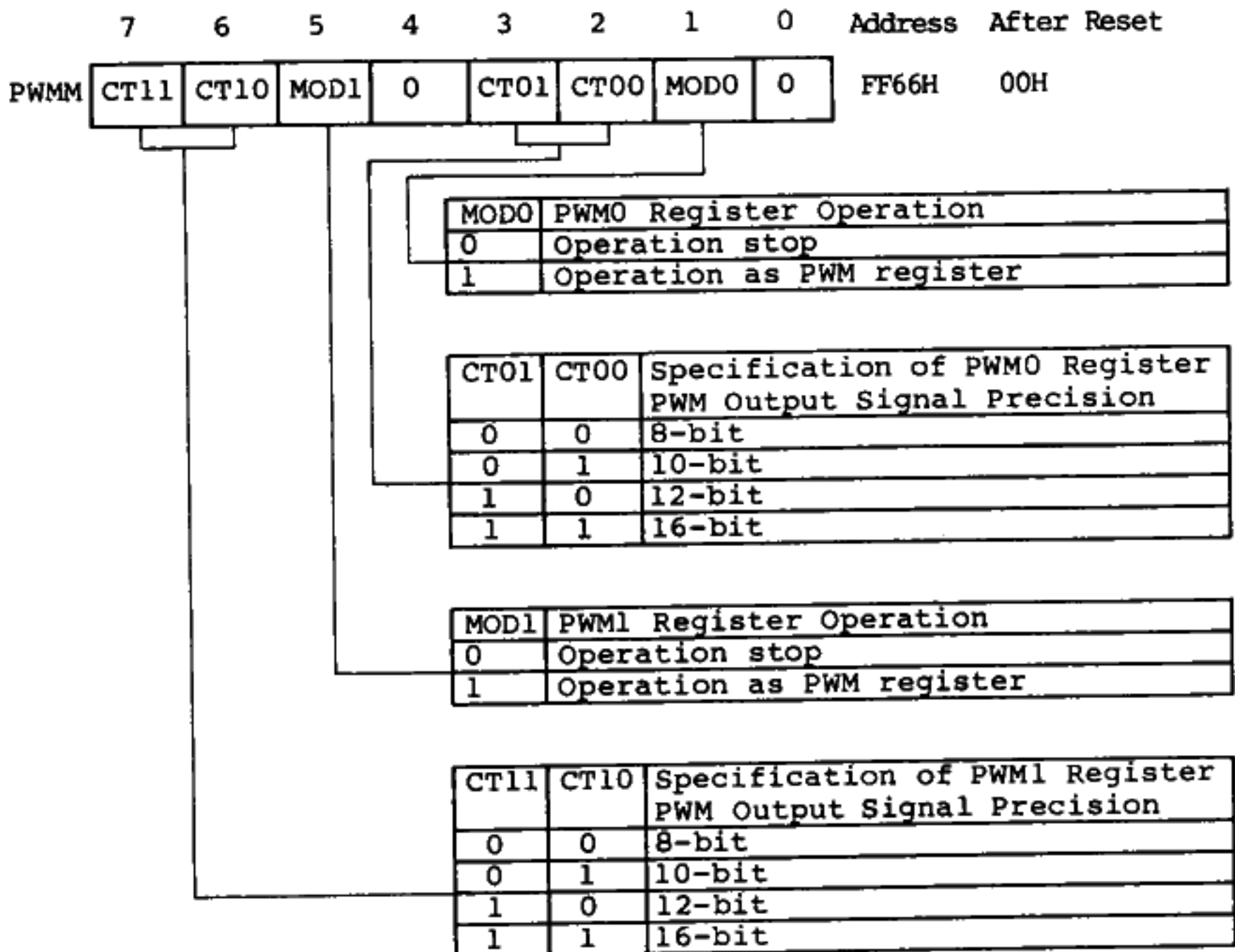
The PWM registers (PWM0 and PWM1) are used to control the PWM output pulse width (duty). When  $\overline{RESET}$  is input, the PWM registers become undefined.

#### (2) PWM mode register (PWMM)

The PWM mode register (PWMM) is an 8-bit register to specify the PWM registers (PWM0 and PWM1) operating modes and the number of significant PWM output bits (PWM resolution). Figure 4-35 shows the PWM mode register format.

When  $\overline{RESET}$  is input, all the PWM mode register bits are cleared to "0".

Figure 4-35 PWM Mode Register Format



### (3) PWM unit operation

PWM unit PWM operation is enabled by setting the PWM mode register (PWMM) MOD bit 1.

To perform PWM output, port 3 mode control register (PMC3) bits 4 and 5 must be set to (1) to set the P34/PWM0 and P35/PWM1 pins to the control mode.

When time base counter tap output (TBC7, TBC9, TBC11, or TBC15) goes high according to the number of significant bits specified in the PWMM register CT0 and CT1 (CT010 and CT011) bits, PWM output is activated, the PWM0 (PWM1) register value is preset in the down counter, and down count is made. When the down count reaches 0000H, the down counter stops and PWM output is deactivated. The operation controls the PWM output pulse width (duty).

The active level of PWM output is specified by using the FRCC register ALV bit. When the bit is set to (1), active high is selected; when (0), active low is selected.

The active level width of PWM output,  $T_w$  is determined by the following expression:

$$T_w = (\text{PWMn register value}) \times t_{\text{cyc}} \quad (n = 0 \text{ or } 1)$$

$t_{\text{cyc}}$  is the internal system clock cycle time and  $\frac{1}{f_{\text{CLK}}}$ .

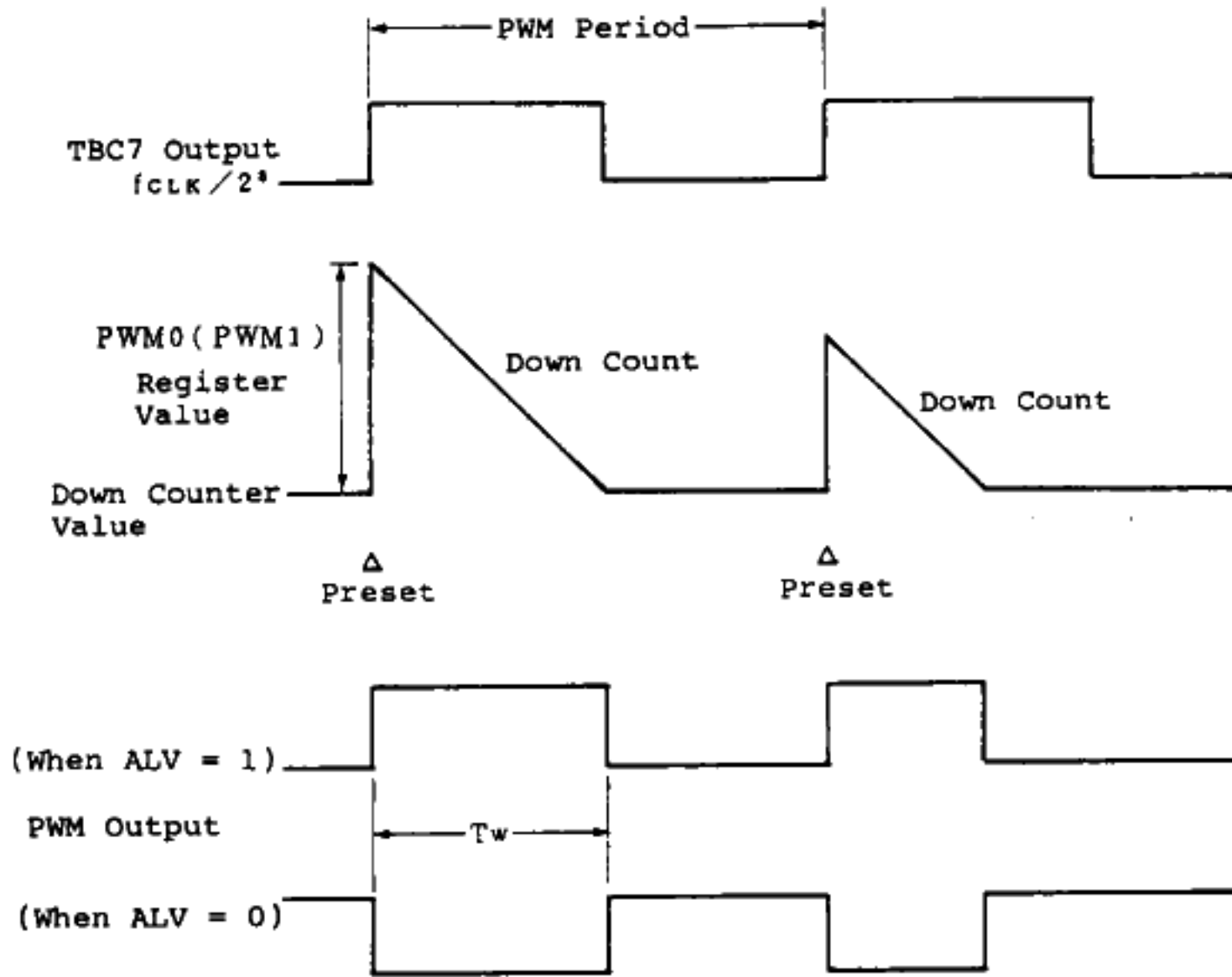
Therefore, if 0000H is written into the PWMn register, the PWM output pin is always deactivated.

Whether PWM output is enabled or disabled is specified by using the FRCC register ENPWM0 (ENPWM1) bit. When the bit is reset to (0), immediately PWM output is deactivated (ALV).

When the PWMM register MOD1 bit is reset to (0), PWM operation stops and the down counter stops counting. If the MOD1 bit is reset to 0 during PWM operation, the down counter stops with the value retained; when the bit is set to 1, the down counter restarts counting at the value. When operation stops, the pin level remains unchanged. However, if the MOD1 bit is set to 1 when TBC tap output selected for input clock is high, the PWM0 (PWM1) register value is again preset and count is started.

Figure 4-36 PWM Output Operation Timing

Example: PWM output operation when time base counter output tap TBC7 is specified.



PWM Period Setting

CT1	CT0	PWM Precision	Frequency	PWM Period
0	0	8-bit PWM output	23.4 kHz	42 us
0	1	10-bit PWM output	5.9 kHz	171 us
1	0	12-bit PWM output	1.5 kHz	683 us
1	1	16-bit PWM output	91.6 Hz	10.9 ms

$$f_{CLK} = 6 \text{ MHz}$$

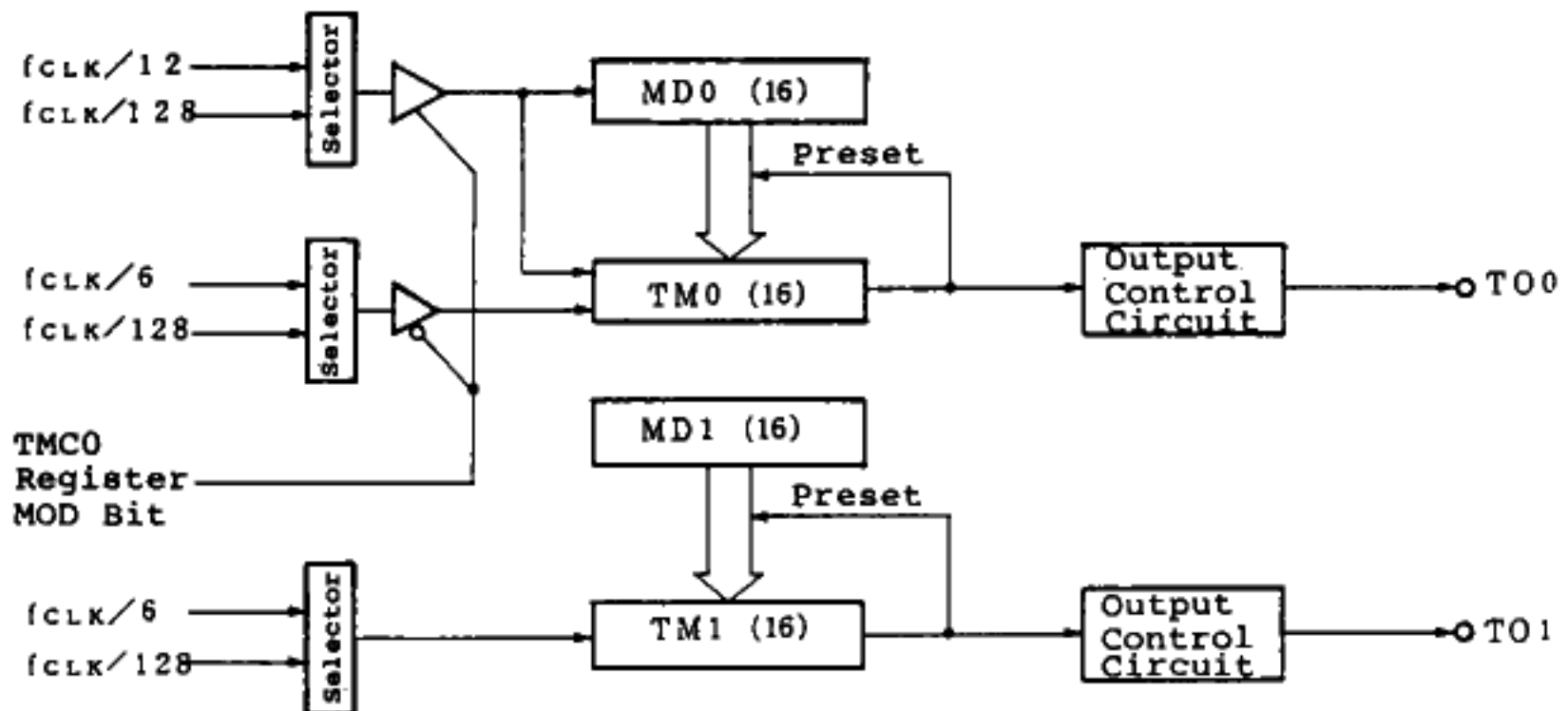
#### 4.3.4 TIMER UNIT

The timer unit has the square wave output function and the timer function for real-time output port output control in addition to the interval timer function and one-shot timer function.

##### (1) Timer unit configuration

The timer unit consists of the 16-bit timer registers (TM0 and TM1), 16-bit modulo/timer registers (MD0 and MD1), 8-bit timer control registers (TMC0 and TMC1), and output control circuit as shown in Figure 4-37.

Figure 4-37 Timer Unit Block Diagram



Remarks:  $f_{CLK}$ : Internal system clock frequency

(2) Timer control registers (TMC0 and TMC1)

The TMC0 register is an 8-bit register to control operation of the TM0 and MD0 registers. The TMC1 register is an 8-bit register to control operation of the TM1 and MD1 registers.

When  $\overline{\text{RESET}}$  is input, the TMC0 and TMC1 registers are cleared (00H).

The TMC0 and TMC1 registers differ in format as shown in Figures 4-38 and 4-39.

The TMC0 register MCLK0 and MS0 bits do not affect count operation when the interval timer operating mode is specified.

NOTE: External trigger control of timer operation specified in the TRG bit is effective only in the interval timer mode. The one-shot timer cannot be started by external trigger.

Figure 4-38 Timer Control Register (TMC0) Format

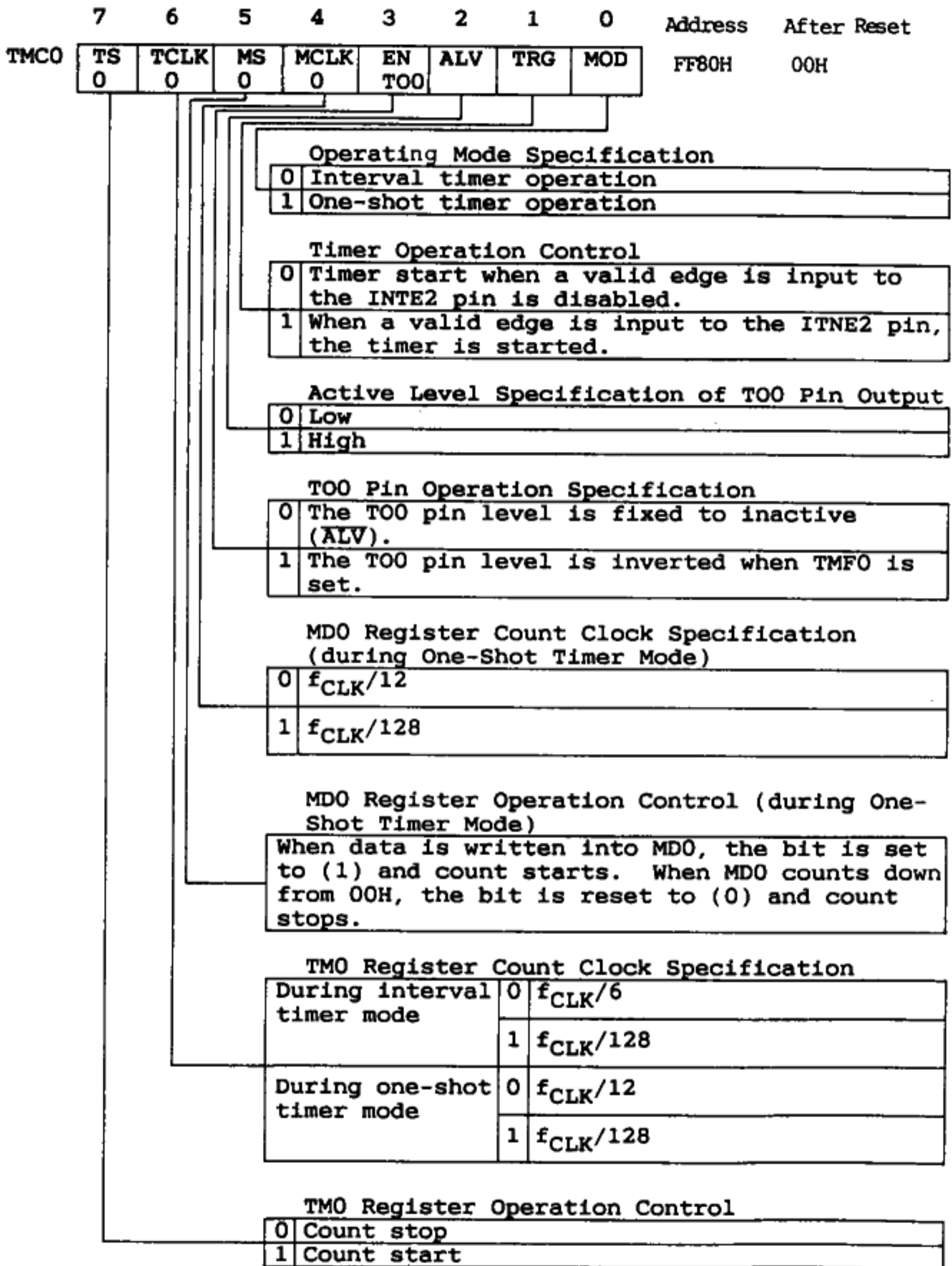
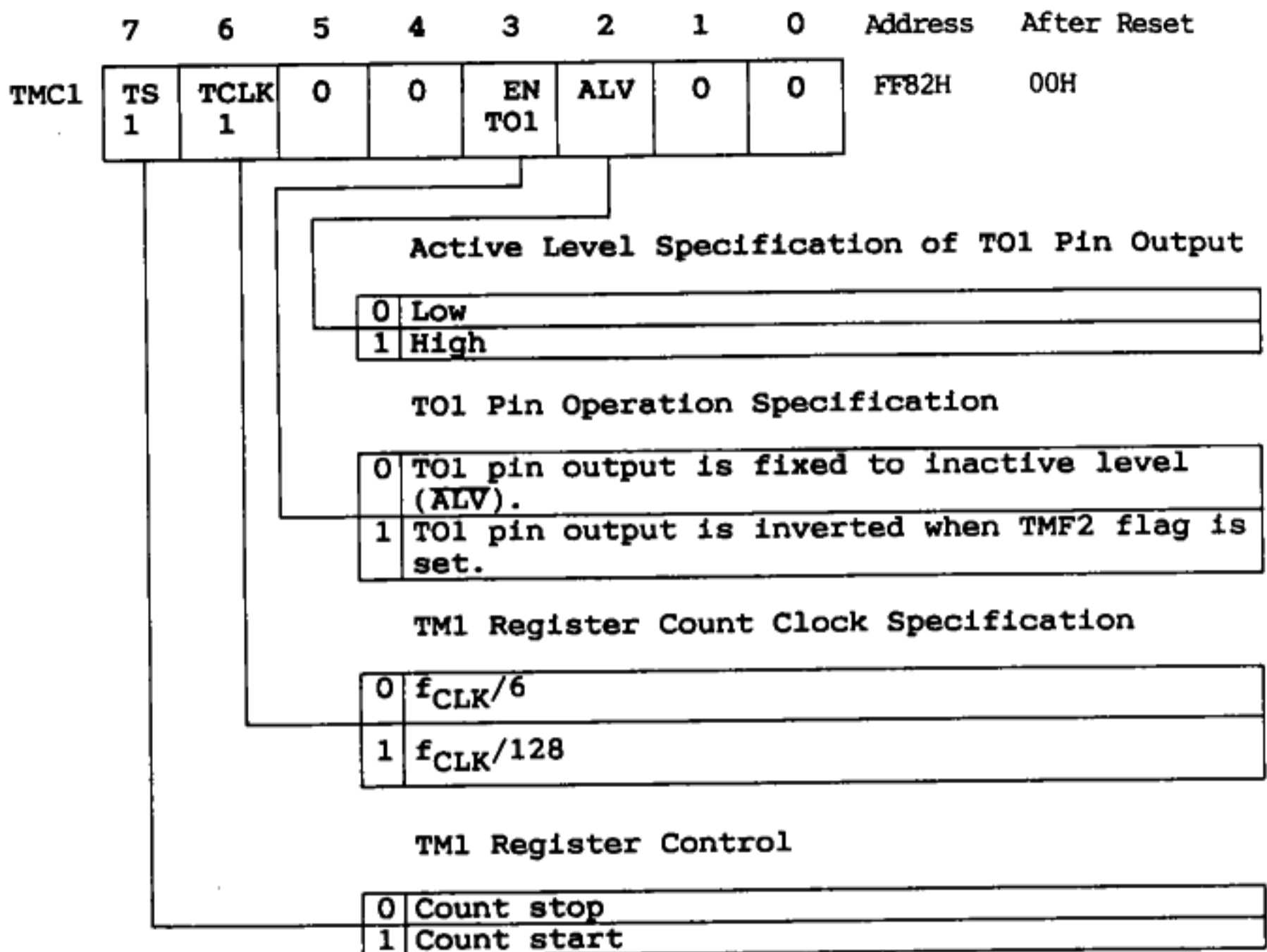


Figure 4-39 Timer Control Register (TMC1) Format



### (3) Timer unit operation

When the TMC0 register MOD bit (bit 0) is reset to 0, the TMO and MD0 registers are set to the interval timer operating mode; when the bit is set to (1), the TMO and MD0 registers are set to the one-shot timer operating mode.

The TM1 and MD1 registers always operate in the interval timer operating mode and are not set to the one-shot timer operating mode.



(a) Interval timer operating mode

When the interval timer operating mode is specified, TMO (TM1) operates as a timer register to count down the setup value and MD0 (MD1) operates as a modulo register which retains the interval set value preset in TMO (TM1).

Interval timer start instruction is given by setting TMC0 (TMC1) register bit 7 (TS0 (TS1)) to (1).

The TMO and MD0 registers contain the interval timer start instruction function when external trigger is input. If TMC0 register bit 1 (TRG) is set to (1), when a valid edge is input to the INTE2 pin, the TS0 bit is set to (1) by hardware and the timer is started.

When the TS0 (TS1) bit is set to (1), the value retained in MD0 (MD1) is preset in TMO (TM1) and count down is made by using clock specified in the TCLK0 (TCLK1) bit. When an underflow occurs as a result of count down, an interrupt request occurs and the TMF0 (TMF1, TMF2) flag is set to (1). The value retained in MD0 (MD1) is again preset in TMO (TM1). This operation sequence is repeated.

The interrupt request continues to occur at the given intervals determined by the TCLK0 (TCLK1) bit and MD0 (MD1) set value.

If the TS0 (TS1) bit is reset to (0) during TMO (TM1) counting down, count operation stops and TMO (TM1) retains the value.

NOTE 1: When the TS0 (TS1) bit is set to (1), if the bit is again set to (1), again the MD0 (MD1) register value is set in the TMO (TM1) register and the timer restarts.

NOTE 2: When the unit is used as an interval timer, if the timer is started with 0000H written into the TMO (TM1) register, the next operation occurs at the first down count.

- . Timer underflow request generated
- . T00(T01) pin output level inverted (ENT00(ENT01) = 1)

To set the interval, prewrite any value other than 0000H into the timer register (TMO, TM1) before starting the timer.

Note that the TMO (TM1) register is undefined after RESET is input.

Table 4-11 Count Time during Interval Timer Operating Mode

Timer register (TMO, TM1) count time

$f_{CLK} = 6 \text{ MHz}$

TCLK0 (TCLK1) Bit	Count Clock	Resolution	Full Count
0	$f_{CLK}/6$	1.0 us	65.5 ms
1	$f_{CLK}/128$	21.3 us	1.40 s

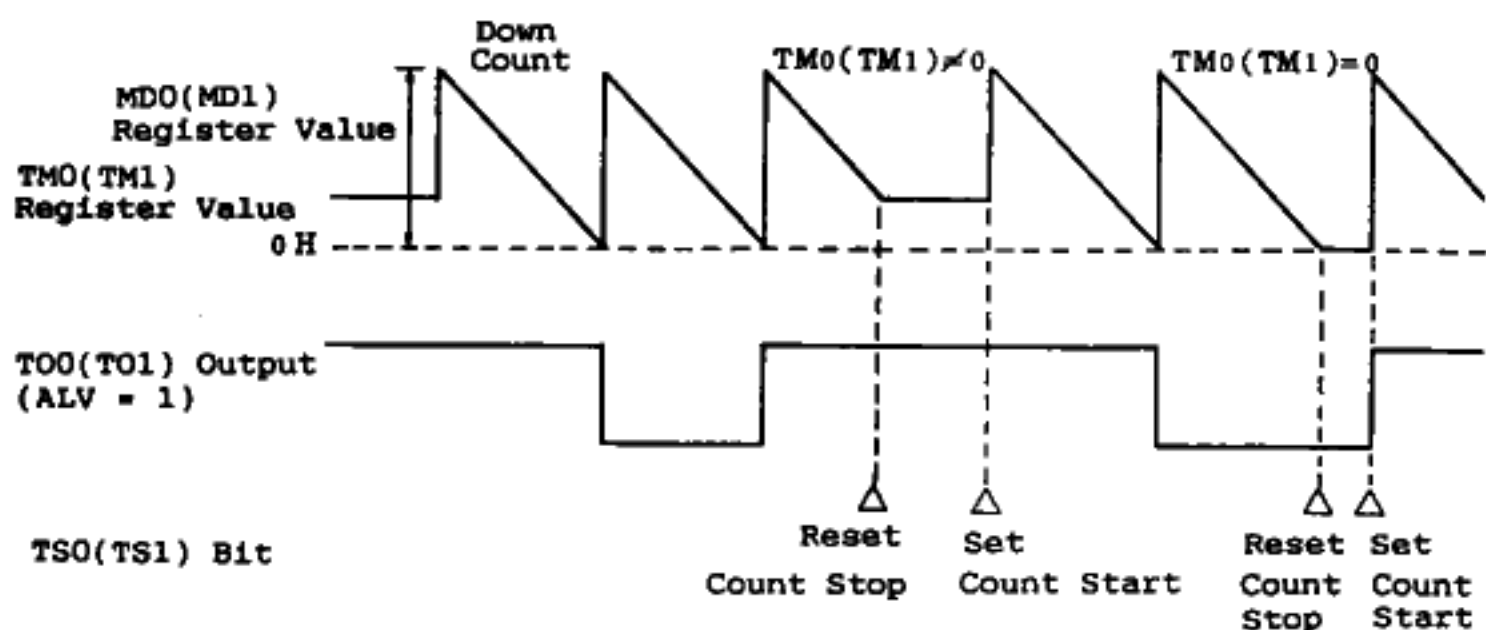
Square wave output to the T00 (T01) pin is controlled by using TMC0 (TMC1) register bit 3 (ENT00 (ENT01)) and bit 2 (ALV).

The active level of T00 (T01) pin output is specified by the ALV bit. When the ALV bit is set to (1), active high is selected; when the bit is reset to (0), active low is selected.

When the ENT00 (ENT01) bit is reset to 0, the T00 (T01) pin level is inactive (ALV). When the bit is set to (1), the T00 (T01) pin level is inverted when the timer unit interrupt request flag TMF0 (TMF2) is set to (1).

To output square wave from the T00 (T01) pin, port 3 mode control register (PMC3) bits 6 and 7 must be set to (1) to set the P36/T00 and P37/T01 pins to the control mode.

Figure 4-40 Timer Unit Output Timing



(b) One-shot timer operating mode

Only TMO and MD0 can be set to the one-shot timer operating mode, in which case TM1 and MD1 operate in the interval timer operating mode.

When the one-shot timer operating mode is specified, the TMO and MD0 registers operate as timer registers to count down the setup value.

One-shot timer start instruction is given by writing data into the TMO and MD0 registers. When the TMC0 register TSO bit (bit 7) or MS0 bit (bit 5) is set to (1) by hardware, count down is started.

TMO and MD0 count down by using clocks specified in the TMC0 register TCLK0 and MCLK0 bits. If the count value reaches "0" as a result of count down, the TSO or MS0 bit is reset to (0) by hardware and count is stopped. At the time, the 0000H value is retained in the TMO or MD0 register.

When the TMO or MD0 count value reaches "0", an interrupt request occurs and TMF0 for TMO or TMF1 for MD0 is set to (1). In this case, when the TM1 register which operates as an interval timer underflows as a result of count down, TMF2 is set to (1).

Table 4-12 Count Time during One-Shot Timer Operating Mode

TMO, MDO register count time

$f_{CLK} = 6 \text{ MHz}$

TCLK0, MCLK0 Bit	Count Clock	Resolution	Full Count
0	$f_{CLK}/12$	2.0 us	131.1 ms
1	$f_{CLK}/128$	21.3 us	1.40 s

(4) Timer unit interrupt request

Three interrupt requests (TMF0 to TMF2) occur from the timer unit. The occurrence conditions of the interrupt requests from the timer unit vary depending on the timer operating mode specification.

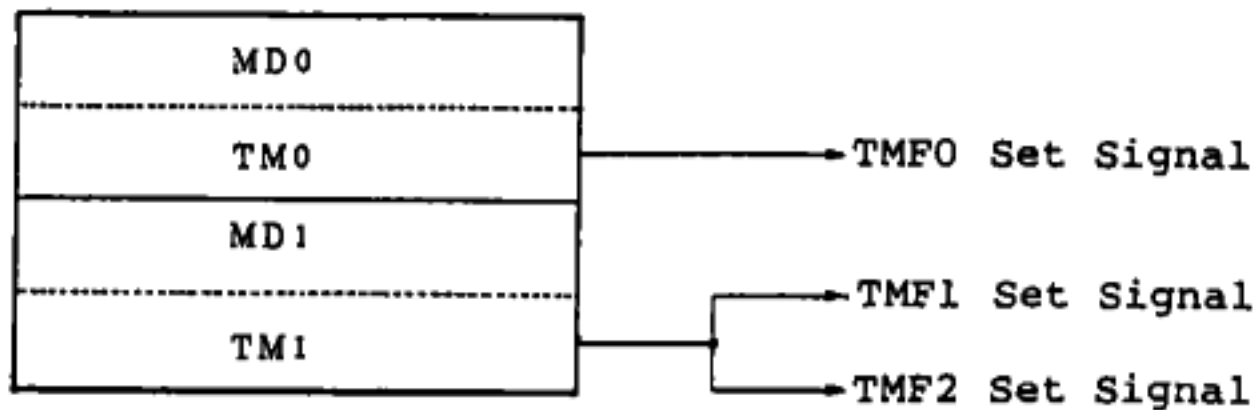
If the interval timer mode is specified, when the TMO register underflows as a result of count down, TMF0 is set to (1); when the TM1 register underflows as a result of count down, TMF1, TMF2 is set to (1). (Figure 4-41 (a))

If the TMO and MDO registers are set to the one-shot timer mode, when the MDO register underflows as a result of count down, TMF1 is set to (1); when the TMO register underflows as a result of count down, TMF0 is set to (1). In this case, when the TM1 register which operates as an interval timer underflows as a result of count down, TMF2 is set to (1). (Figure 4-41 (b))

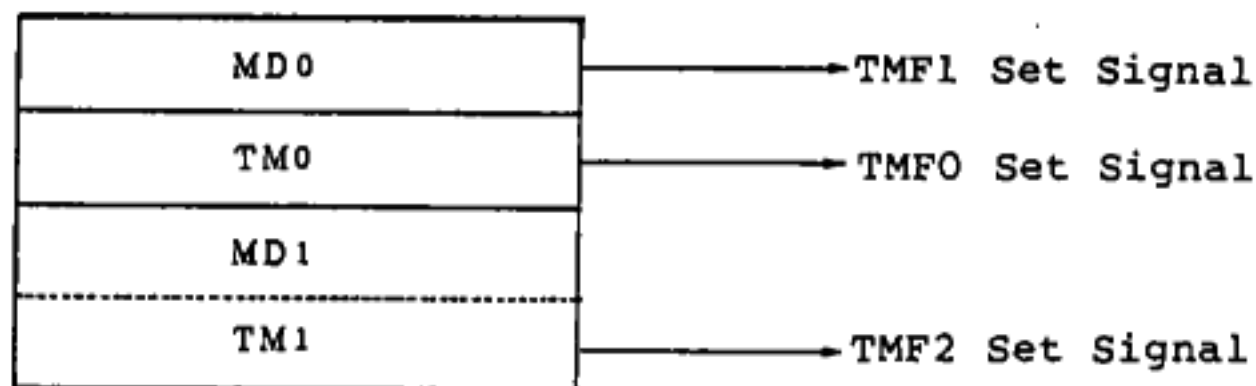
The TMF0 and TMF1 set signals can be selected for real-time output port output triggers.

Figure 4-41 Interrupt Requests from Timer Unit

(a) When TM0 and MD0 are set to interval timer mode



(b) When TM0 and MD0 are set to one-shot timer mode



TMF0 to TMF2: Timer unit interrupt request flags 0 to 2

(a) Timer unit interrupt request control registers (TMIC0, TMIC1, and TMIC2)

The TMIC0 to TMIC2 registers are used to control three interrupt requests occurring from the timer unit. The three interrupt requests make up one group and the timer unit interrupt request priority level is programmable by using the PR2 to PR0 bits of the TMIC0 register. In the group, the priority levels are fixed by hardware as follows:

TMF0 > TMF1 > TMF2

The TMIC0 to TMIC2 register bits are described in "CHAPTER 5. INTERRUPT FUNCTION".

Figure 4-42 Timer Unit Interrupt Request Control Register Format

	7	6	5	4	3	2	1	0	Address	After Reset
TMIC0	TMF0	TM MK0	TM ISM0	TM CSE0	0	PR2	PR1	PR0	FFCEH	47H
TMIC1	TMF1	TM MK1	TM ISM1	TM CSE1	0	-	-	-	FFD0H	47H
TMIC2	TMF2	TM MK2	TM ISM2	TM CSE2	0	-	-	-	FFD2H	47H

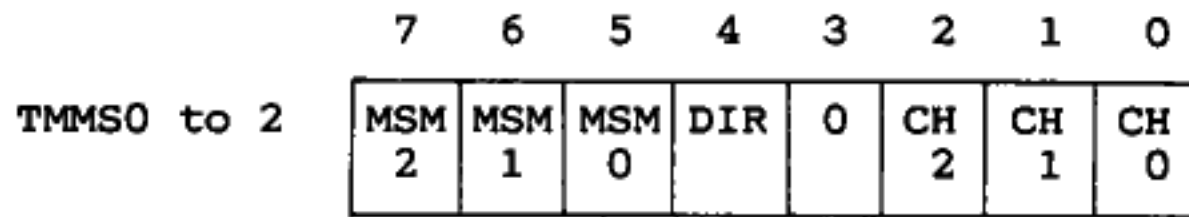
Remarks: "-" denotes that write cannot be made. If the bit is read, "1" is read.

(b) Timer unit macro service control registers (TMMS0 to TMMS2)

The TMMS0 to TMMS2 registers are used to control macro services started when the three interrupt requests occur from the timer unit.

The TMMS0 register is used to control the macro service when the TMF0 flag is set to (1); the TMMS1 register is used to control the macro service when the TMF1 flag is set to (1); and the TMMS2 register is used to control the macro service started when the TMF2 flag is set to (1).

**Figure 4-43 Macro Service Control Register Format**



**Remarks:**

	Address	After Reset
TMMS0 :	FFCFH	Undefined
TMMS1 :	FFD1H	Undefined
TMMS2 :	FFD3H	Undefined

The TMMS0 to TMMS2 register bits are described in "5.2 MACRO SERVICE FUNCTION".

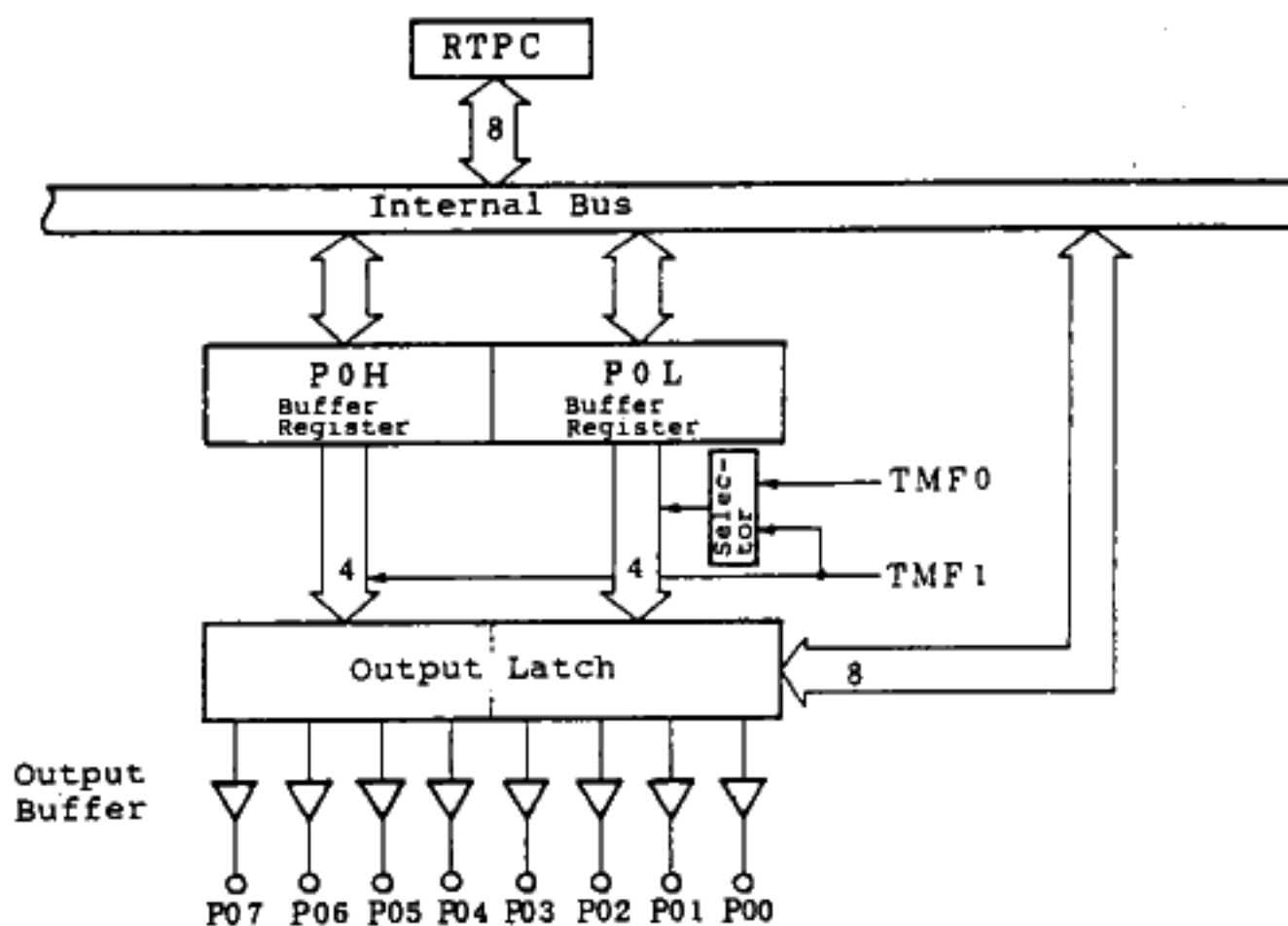
### 4.3.5 REAL-TIME OUTPUT PORT

Port 0 contains the real-time output port function and can output the port 0 buffer register contents in 4-bit or 8-bit units at programmable intervals.

#### (1) Real-time output port configuration

The real-time output port takes the master slave configuration and consists of the buffer register, output latch, and real-time output port control register (RTPC), as shown in Figure 4-44.

Figure 4-44 Real-Time Output Port Block Diagram



#### (a) Port 0 buffer register (POH, POL)

The buffer register retains the next data to be output when port 0 is set to the real-time output port mode. The buffer register contents are transferred to the output latch at the output timing listed in Table 4-14.

The buffer register contents are not affected by RESET input.

#### (b) Output latch

The output latch retains port 0 output data.



(2) Real-time output port control register (RTPC)

The RTPC register is an 8-bit register to specify the port 0 function and real-time output port operating mode. Figure 4-45 shows the RTPC register format.

The POLM and POHM bits are used to specify the operating modes of the low-order four bits and high-order four bits of port 0. When the POLM and POHM bits are reset to (0), port 0 serves as a normal 3-state input/output port. When the bits are set to (1), port 0 functions as a real-time output port.

If port 0 is used as a 1-byte (8-bit) real-time output port rather than separate 4-bit real-time output ports (low-order four bits and high-order four bits of port 0), the BYTE bit is set to 1.

The TRG bit is used to specify the transfer trigger of the buffer register value to the output latch. It is effective only for the port bits set to the real-time output port.

When  $\overline{\text{RESET}}$  is input, the RTPC register is set to 08H and port 0 is set to the general-purpose input/output port mode.

Figure 4-45 Real-Time Output Port Control Register Format

	7	6	5	4	3	2	1	0	Address	After Reset
RTPC	0	0	0	0	BYTE	TRG	POHM	POLM	FF38H	08H

POHM	POLM	Port 0 Function Specification
0	0	P00 to P07: 3-state input/output port
0	1	P00 to P03: Real-time output port P04 to P07: 3-state input/output port
1	0	P00 to P03: 3-state input/output port P04 to P07: Real-time output port
1	1	P00 to P07: Real-time output port

TRG	Real-Time Output Port Transfer Timing Specification
0	When the TMFn flag is set to 1, the buffer register contents are transferred. (When TMF0 is set, the POL contents are transferred; when TMF1 is set, the POH contents are transferred; when TMF1 is set, the POL and POH contents are transferred.)
1	When 1 is written into the TRG bit, data is transferred. (After data is transferred, 1 is retained.)

BYTE	Real-Time Output Port Operating Mode
0	4-bit separate real-time output ports
1	8-bit real-time output port

TMFn: Time unit interrupt request flag (n = 0 to 2)

(3) Real-time output port operating mode

The operating mode can be selected in 4-bit units for port 0 by setting RTPC register bits 3, 1, and 0 (BYTE, POMH, and POML).

Table 4-13 lists the read and write access targets for each operating mode.

Port 0 addresses that can be accessed are PO, POH, and POL which can be described in the sfr operand.

Table 4-13 Port 0 Access Targets

RTPC			Operating Mode	Access Address	To be Accessed (Read)		To be Accessed (Write)	
BYTE	POMH	POML			High-Order Four Bits	Low-Order Four Bits	High-Order Four Bits	Low-Order Four Bits
x	0	0	8-bit port mode	PO	Port access*		Output latch	
				POH	POH	POL	POH	-
				POL	POH	POL	-	POL
x	0	1	P07 to P04: Port mode P03 to P00: Real-time output port	PO	Port access*	Output latch	Output latch	-
				POH	POH	POL	POH	-
				POL	POH	POL	-	POL
x	1	0	P07 to P04: Real-time output port P03 to P00: Port mode	PO	Output latch	Port access*	-	Output latch
				POH	POH	POL	POH	-
				POL	POH	POL	-	POL
0	1	1	4-bit separate real-time output port	PO	Output latch		-	
				POH	POH	POL	POH	-
				POL	POH	POL	-	POL

(to be continued)

Table 4-13 Port 0 Access Targets (cont'd)

RTPC			Operating Mode	Access Address	To be Accessed (Read)		To be Accessed (Write)	
BYTE	POMH	POML			High-Order Four Bits	Low-Order Four Bits	High-Order Four Bits	Low-Order Four Bits
1	1	1	8-bit real time output port	P0	Output latch		-	
				POH	POH	POL	POH	POL
				POL	POH	POL	POH	POL

\*: The output latch of the bit set to the output mode by using the port 0 mode register (PM0) is accessed; the pin level of the bit set to the input mode is accessed.

Remarks 1: -: Write access is not made. Output latch and buffer register (POH, POL) are not affected.

2: x: Don't care

NOTE: If port 0 is set to 8-bit real time output port, when either POH or POL is accessed, 8-bit data is always written into POH and POL.

(4) Real time output port output timing

Table 4-14 lists the transfer timing of the buffer register (POH, POL) contents to the output latch when port 0 is set to real time output port according to how the RTPC register BYTE and TRG bits are set.

Table 4-14 Real Time Output Port Output Timings

BYTE	TRG	POH	POL
0	0	When TMF1 is set	When TMF0 is set
0	1	When 1 is written into the TRG bit	When 1 is written into the TRG bit
1	0	When TMF1 is set	
1	1	When 1 is written into the TRG bit	

Remarks: If data is output to the output latch by writing 1 into the TRG bit, the TRG bit also contains 1 after the data is transferred.

#### 4.4 SERIAL COMMUNICATION INTERFACE

The uPD78312A has a serial communication interface which contains a dedicated baud rate generator.

The serial communication interface operates in the asynchronous mode or I/O interface mode.

In the asynchronous mode, bit synchronization and character synchronization are taken by using start and stop bits for transfer. In the I/O interface mode, serial data is transferred in synchronization with controlled serial clock as in the 87 AD series.

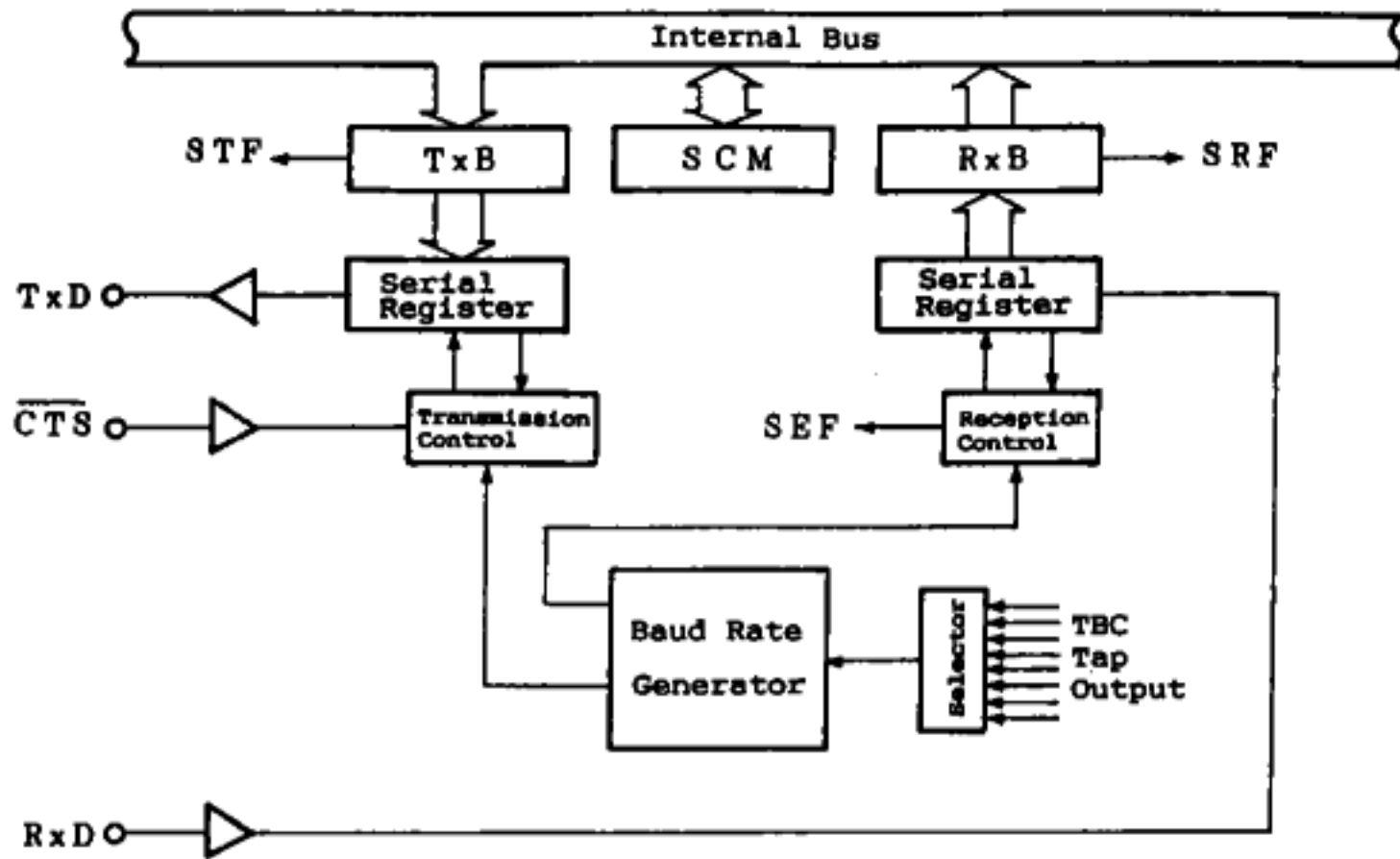
##### 4.4.1 SERIAL COMMUNICATION INTERFACE CONFIGURATION

Figure 4-46 shows the serial communication interface configuration.

The serial communication interface which contains serial registers and buffers for transmission and reception enables data to be transmitted and received independently. Since the  $\overline{\text{CTS}}$  pin has the receive clock input/output pin function in the I/O interface mode, serial communication is also enabled in full duplex in the I/O interface mode.

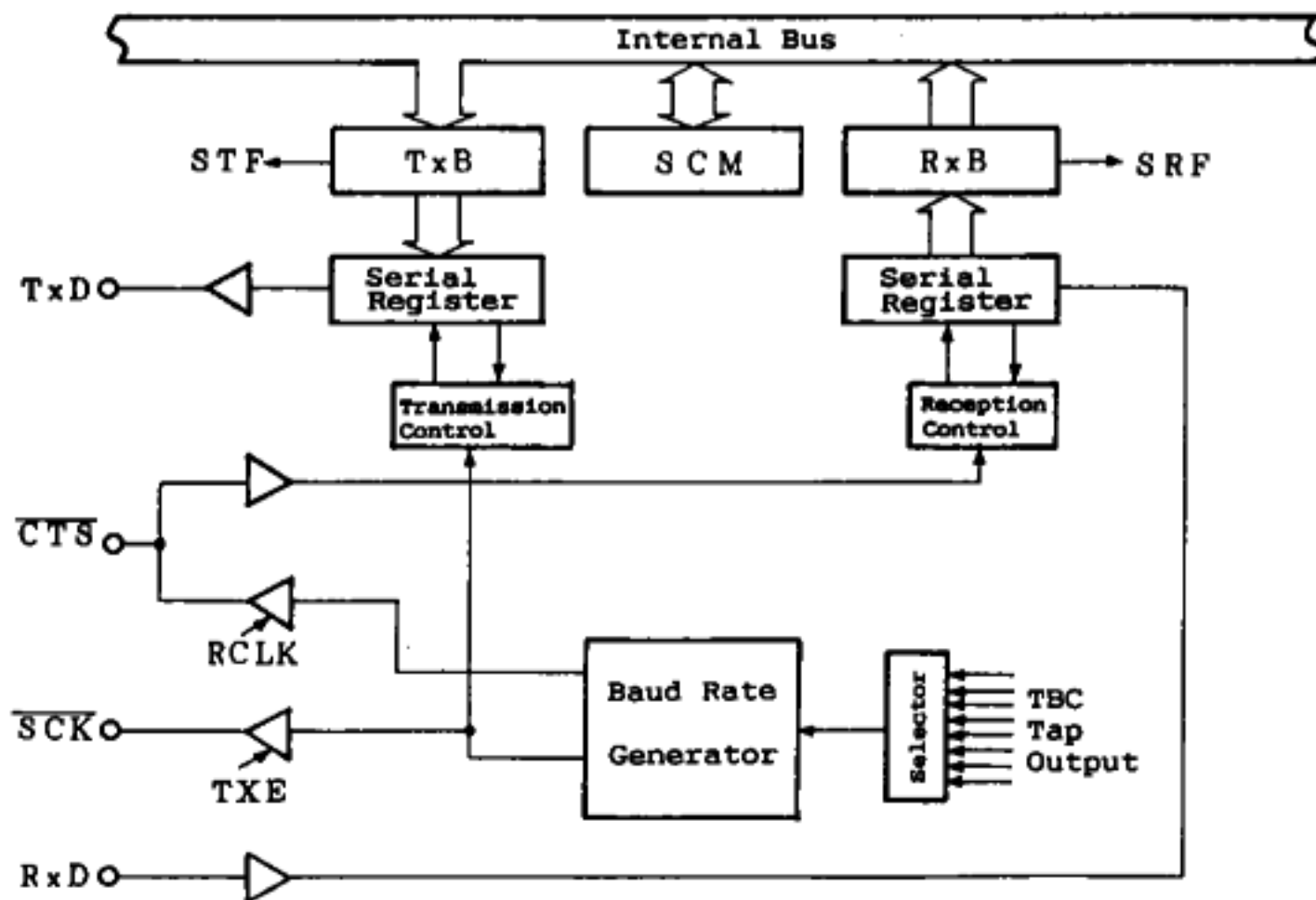
Figure 4-46 Serial Communication Interface Block Diagram

(a) When asynchronous mode is set



Remarks: The  $\overline{SCK}$  pin is fixed high.

(b) When I/O interface mode



#### 4.4.2 MODE REGISTERS

##### (1) Serial communication mode register (SCM)

The serial communication mode register (SCM) is an 8-bit register to specify the serial interface transfer mode. The meanings of SCM register bits 2 to 7 vary depending on how bits 1 and 0 (MD1 and MD0) are set.

7	6	5	4	3	2	1	0
						MD1	MD0

MD1, MD0 = 0, 1 (asynchronous mode)

7	6	5	4	3	2	1	0
TXRDY	RXE	PEN	EP	CL	SL	0	1

MD1, MD0 = 0, 0 (I/O interface mode)

7	6	5	4	3	2	1	0
TXE	RXE	0	0	TSK	RSCK	0	0

The MD1 and MD0 bits are a bit field to specify the serial interface transfer mode. When MD1 and MD0 are set to 0 and 1 (01), the asynchronous mode is selected; when 0 and 0 (00), the I/O interface mode is selected.

When  $\overline{\text{RESET}}$  is input, the SCM register is cleared (00H) and the I/O interface mode is set.

**NOTE:** Before changing the serial interface transfer mode between the I/O interface and asynchronous, be sure to once set the RXE bit to 0 (reception disable).



**Example:** Change from I/O interface mode to asynchronous mode

```

; I/O interface mode
; reception is enabled
MOV SCM, #x0xxxx01B; Disable asynchronous
mode reception
MOV SCM, #x1xxxx01B; Enable asynchronous
mode reception
```

(a) When asynchronous mode is set

The RXE bit is used to control reception enable in the asynchronous or I/O interface mode. When the RXE bit is set to 0 (reception disable) during reception operation, reception processing is stopped at the time and a reception completion interrupt request does not occur.

The SL bit is used to specify the stop bit length. When the SL bit is reset to 0, the stop bit length becomes one bit; when set to 1, the stop bit length becomes two bits.

The CL bit is used to specify the character length. When the CL bit is reset to 0, the character length becomes seven bits; when set to 1, the character length becomes eight bits.

The PEN bit is used to specify parity enable. When the PEN bit is reset to 0, no parity is specified; when set to 1, parity bit is added. When parity bit is added, the EP bit is used to select odd or even parity. When the EP bit is reset to 0, odd parity is selected; when set to 1, even parity is selected. The EP bit is significant only when the PEN bit is set to 1.

The TXRDY bit is used to control transmission enable. When the  $\overline{\text{CTS}}$  pin is set low and TXRDY is set to 1, transmission is enabled.

**NOTE:** When setting to asynchronous mode, the  $\overline{\text{CTS}}$ /P27 pin should always be set to control mode. (Bit 7(PMC27) of port 2 mode control register(PMC2) is set to 1.) If setting to port mode, the serial communication interface cannot be implemented.

(b) When I/O interface mode is set

The RSCK bit is used to specify the serial receive clock source. When the RSCK bit is reset to 0, external receive clock is used for reception operation; when set to 1, internal receive clock is used for reception operation. The receive clock is input/output through the  $\overline{\text{CTS}}$  pin.

The TSK bit has the receive clock output trigger bit function. The bit is significant only when the RSCK bit is set to 1. When 1 is written into the TSK bit, eight shift clocks for reception are output from the  $\overline{\text{CTS}}$  pin.

The RXE bit is used to control reception enable. When the RXE bit is set to 1, reception is enabled; when reset to 0, reception is disabled. If the RXE bit is reset to 0 (reception disable) during reception operation, reception processing is stopped at the time and a reception completion interrupt request does not occur.

The TXE bit is used to control transmission enable. When the TXE bit is set to 1, transmission is enabled; when reset to 0, transmission is disabled.

NOTE: When the RSCK bit is set to 0 (external clock reception mode), do not set the receive clock output trigger bit (TSK) to 1. If the TSK bit is set to 1, the counter which counts receive clocks is unconditionally reset, and external clocks cannot be counted accurately.

Figure 4-47 Serial Communication Mode Register Format (a)

When asynchronous mode is set (bit 0 = 1)

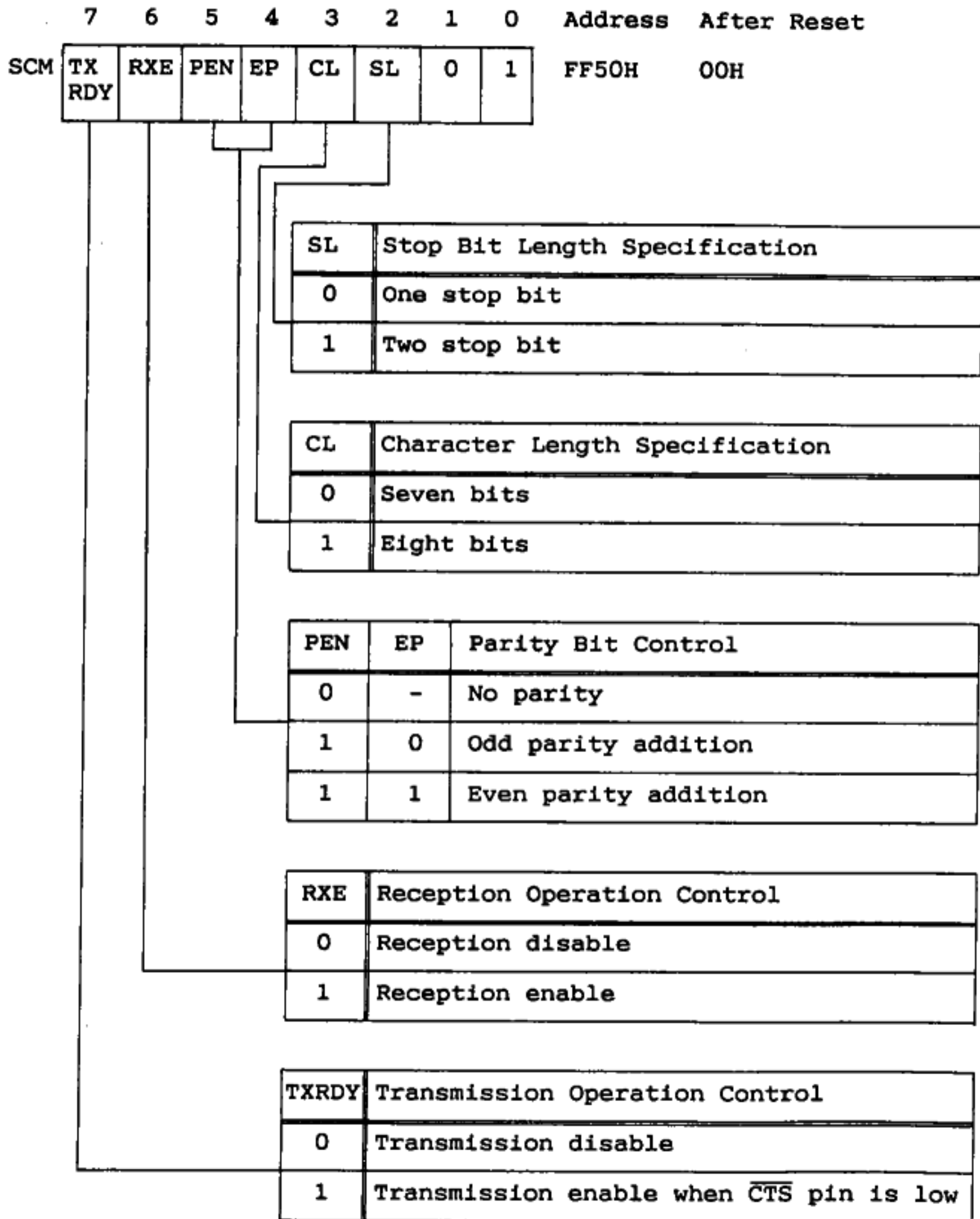


Figure 4-48 Serial Communication Mode Register Format (b)

When I/O interface mode is set (bit 0 = 0)

	7	6	5	4	3	2	1	0	Address	After Reset
SCM	TXE	RXE	0	0	TSK	RSCK	0	0	FF50H	00H

RSCK	Synchronization Clock Specification
0	External receive clock is used for reception (CTS pin input mode)
1	Internal receive clock is used for reception (CTS pin output mode)

When 1 is written into this bit, eight shift clocks are output (effective only when RSCK = 1)

RXE	Reception Operation Control
0	Reception disable
1	Reception enable

TXE	Transmission Operation Control
0	Reception disable
1	Reception enable

(2) Serial communication control register (SCC)

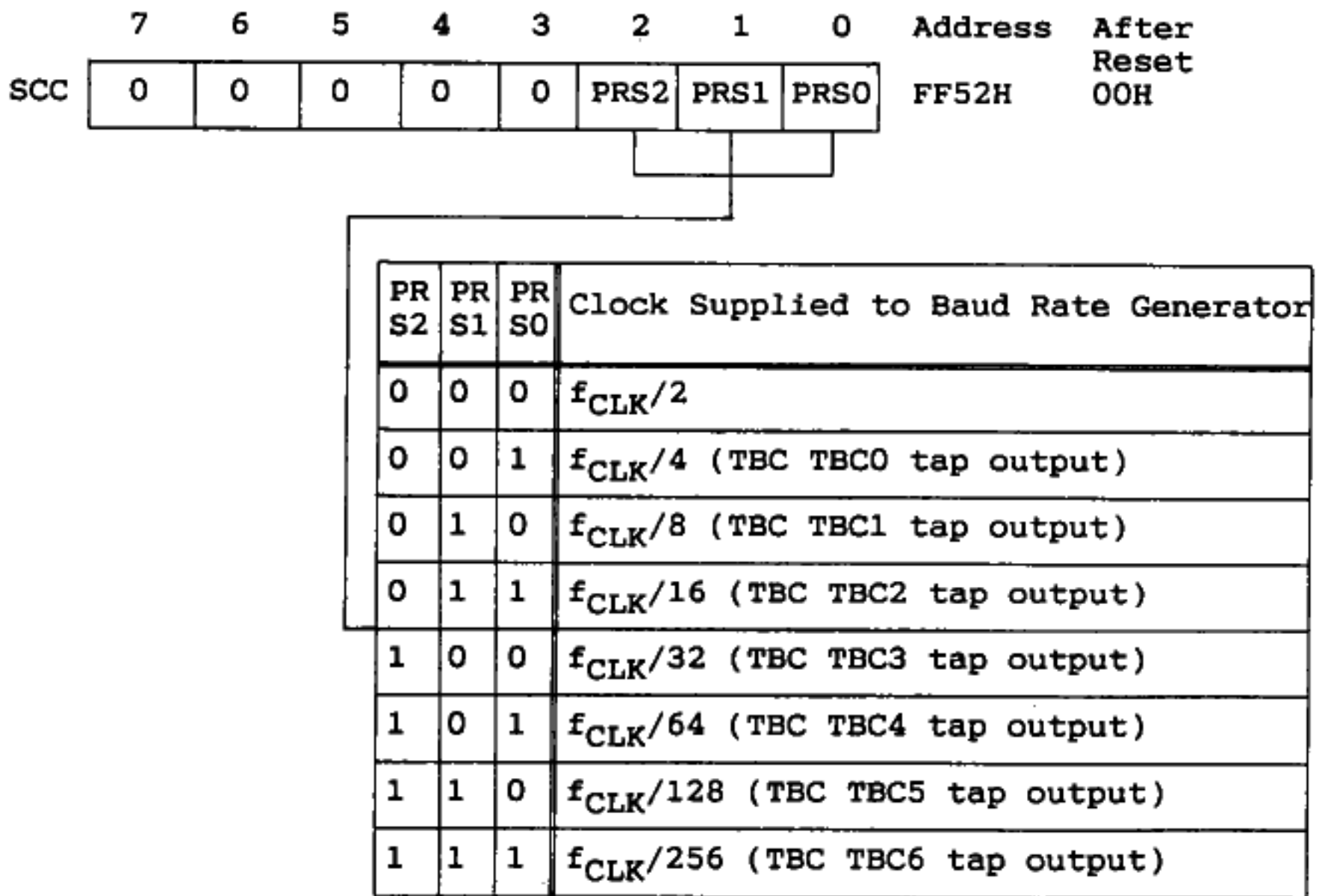
The serial communication control register (SCC) is an 8-bit register to control the serial communication interface transfer rate.

Bits 2 to 0 (PRS2 to PRS0) are used to specify time base counter tap output input to the baud rate generator.

When  $\overline{\text{RESET}}$  is input, the SCC register is cleared (00H).

Figure 4-49 shows the SCC register format.

Figure 4-49 Serial Communication Control Register Format



#### 4.4.3 BAUD RATE GENERATOR (BRG)

The baud rate generator is an 8-bit timer which is dedicated to the serial communication interface and generates transmit or receive shift clock. The transmission and reception baud rate generators are provided. (See Figure 4-50.)

##### (1) Baud rate generator setting

Input clock to the baud rate generator is specified by selecting time base counter output tap in the serial communication control register (SCC) PRS2 to PRS0 bits. Serial communication interface shift clock is provided by furthermore dividing the baud rate generator output signal by two. Set the parameter values in the baud rate generator for the transfer rate so as to satisfy the following expression:

$$B G = 10 \times \frac{f_{\text{CLK}}}{2^{n+2}}$$

Where:

B : Transfer baud rate [bps]

B = 110, 150, ....., 9600, 19200, .....

G : Baud rate generator set value ( $1 \leq G \leq 255$ )

n : Input clock specification number to the baud rate generator specified in SCC register PRS2 to PRS0 bits ( $0 \leq n \leq 7$ )

$f_{\text{CLK}}$ : Internal system clock frequency [MHz]

Table 4-15 lists the baud rate generator set values for each standard transfer baud rate when internal system clock 6 MHz is used based on the expression shown above.

Figure 4-50 Baud Rate Generator Configuration

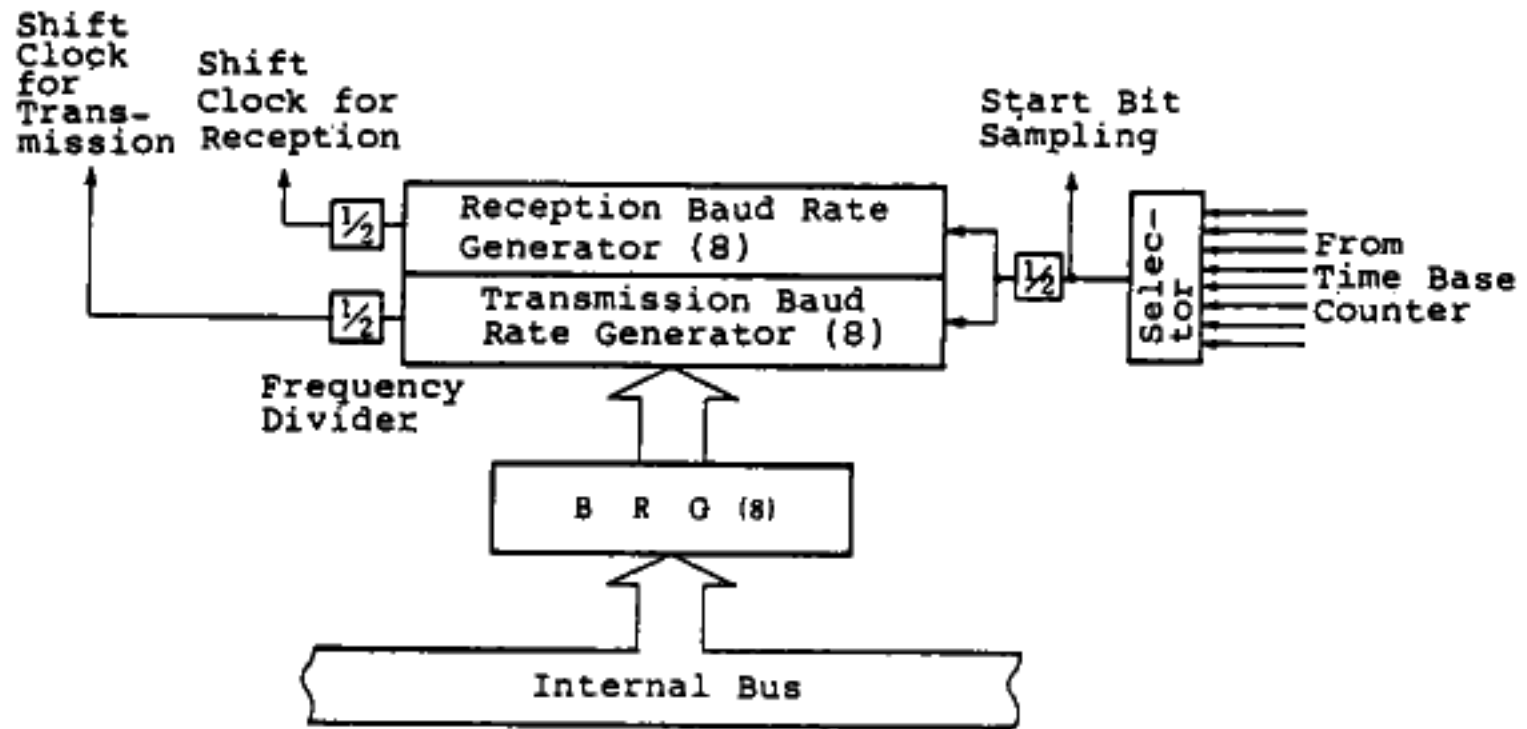


Table 4-15 Baud Rate Generator Set Values (for Reference)

$f_{CLK} = 6 \text{ MHz}$

Transfer Baud Rate	SCC Register PRS2 to PRS0 Set Value n	Baud Rate Generator Set Value G	Error (%)
110	7	107	0.43
150	7	78	0.16
300	6	78	0.16
600	5	78	0.16
1200	4	78	0.16
2400	3	78	0.16
4800	2	78	0.16
9600	1	78	0.16
19200	0	78	0.16
38400	0	39	0.16
1.5M	0	1	0

NOTE: Set a value of 2 or more for the baud rate generator (BRG) during the I/O interface mode. (When internal system clock is set to 6 MHz and BRG = 2 is set, the transfer baud rate becomes 750 kbps.)

Remarks: Error is calculated as shown below:

Error (%)

$$= \frac{|(\text{baud rate calculation result}) - (\text{expected baud rate})|}{(\text{expected baud rate})} \times 100$$

$$\text{Where (baud rate calculation result)} = \frac{f_{CLK}}{2^{n+2} \times G}$$



(2) Note when baud rate is set

The baud rate generator samples a start bit by using input clock to the baud rate generator. To lessen an error during start bit sampling, clock of frequency as high as possible should be supplied to the baud rate generator.

Thus, set a value as low as possible in the low-order three bits of the SCC register (PRS0 to PRS2).

For example, to set the transfer baud rate to 9600 bps, set either of the following:

(a)  $n = 0$  and  $G = 156 (= 9CH)$

(b)  $n = 1$  and  $G = 78 (= 4EH)$

In either setting (a) or (b), equal error 0.16 occurs. Since the frequency of clock supplied from the time base counter in (a) is higher than that in (b), shift during start bit sampling in (a) is smaller than that in (b).

#### 4.4.4 SERIAL COMMUNICATION INTERFACE OPERATION

##### (1) Asynchronous mode

In the asynchronous mode, the character length, the number of stop bits, parity enable, and even or odd parity can be specified in the serial communication mode register (SCM).

##### (a) Transmission

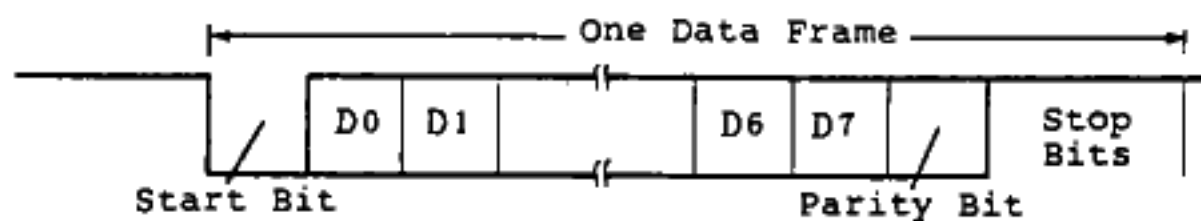
When serial communication mode register (SCM) bit 7 (TXRDY) is set to 1 and the  $\overline{\text{CTS}}$  pin is active low (0), transmission is enabled. In this case, transmission can be enabled by first setting TXRDY to 1, then activating the  $\overline{\text{CTS}}$  pin or first activating the  $\overline{\text{CTS}}$  pin, then setting TXRDY to 1.

Transmission is started by any of the following three methods:

- (i) By setting the transmission enable state when the transmit buffer (TxB) is empty, a transmission completion interrupt request is generated and send data is written into the transmit buffer in the interrupt service.
- (ii) When send data is transferred to the transmit buffer in the transmission enable state, the send data is transferred consecutively after the preceding transmission operation terminates.
- (iii) When send data is prewritten into the transmit buffer in the transmission disable state and then the transmission enable state is set, the data retained in the transmit buffer is transmitted.

The send data format is shown below. The start bit, character bits, and stop bit or bits make up one data frame. Send data is sent from the TxD pin starting at the least significant bit (LSB). When transmission is disabled or the serial register does not contain data to be transmitted, the TxD pin becomes mark state (1).

Figure 4-51 Transmission Timing during Asynchronous Mode



Number of start bits : One  
Number of character bits: Seven or eight  
Parity bit : Odd, even, or no parity  
Number of stop bits : One or two

When the transmit buffer (TxB) becomes empty, a transmission completion interrupt request occurs immediately. When  $\overline{\text{RESET}}$  is input, the transmit buffer (TxB) is cleared. If transmission is enabled at the time, a transmission completion interrupt request occurs. When transmission operation starts and send data in the transmit buffer is transferred to the shift register, the transmit buffer becomes empty and a transmission completion interrupt request occurs.

If send data is written into the transmit buffer each time a transmission completion interrupt request occurs, consecutive data transmission is enabled without insertion of mark state (1).

If a change is made to the transmission disable state when transmission operation is being performed, the entire frame of the data being transmitted is transmitted. However, if new send data is already written into the transmit buffer, data transfer from the transmit buffer to the shift register is disabled and the transmit buffer contents are retained intact. When transmission is again enabled, the transmit buffer contents are transferred to the shift register. A transmission completion interrupt request occurs at the same time as transmission is started.

Remarks: When the character length is set to seven bits, send data is written into the low-order seven bits of TxB. The most significant bit (MSB) contents are not transmitted.

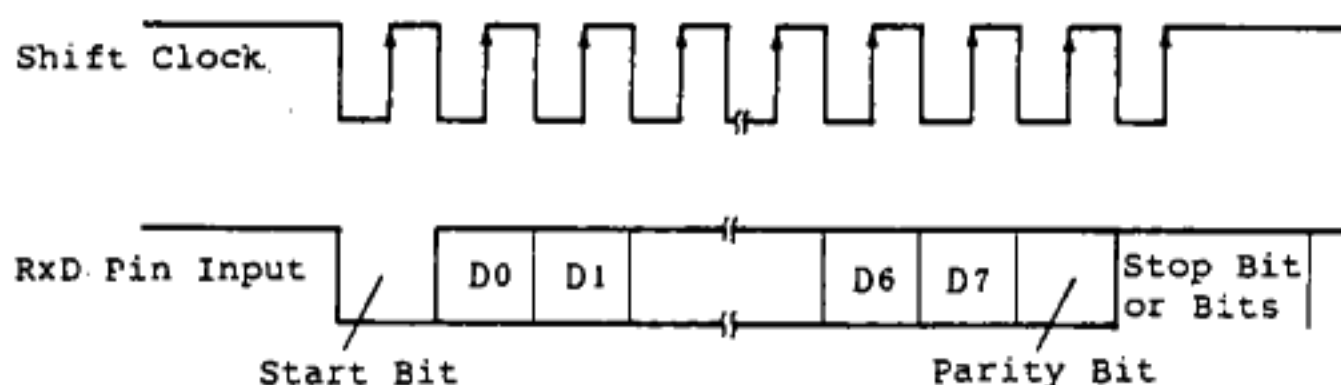
(b) Reception

When serial communication mode register (SCM) bit 6 (RXE) is set to 1, reception is enabled. (When reception is disabled (RXE = 0), reception hardware stands by in the initial state.)

When RxD pin input is sampled by using input clock to the baud rate generator and falling edge is detected, reception operation is started and the reception baud rate generator starts counting. If RxD pin input is detected low on the first timing signal from the reception baud rate generator, it is recognized as a start bit and subsequent reception operation is performed. If RxD pin input is detected high on the first timing signal, it is not recognized as a start bit and the baud rate generator is initialized and operation is stopped.

Receive data is sampled in synchronization with the shift clock rising edge after the start bit is detected.

Figure 4-52 Reception Timing during Asynchronous Mode



When data of the character length specified in serial communication mode register bit 3 (CL) has been received, receive data in the shift register is transferred to the receive buffer (RxB) and a reception completion interrupt request is generated.

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

Remarks: When the character length is set to seven bits, receive data is set in the low-order seven bits of RxB and 0 is set in the MSB.

(2) I/O interface mode

The I/O interface mode is the same as the 87AD series serial interface; it is useful when external I/O expansion is made or I/O controller such as analog-to-digital converter or liquid crystal controller is connected.

In the I/O interface mode, data is transferred starting at the most significant bit (MSB) with the character length fixed to eight bits and no parity bit.

(a) Transmission

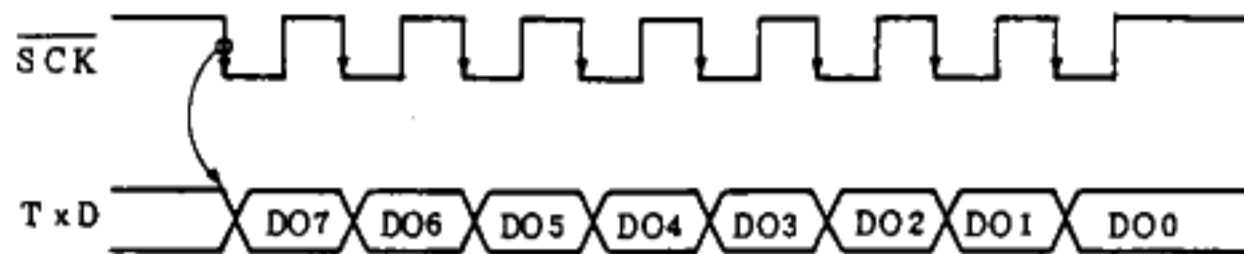
When serial communication mode register bit 7 (TXE) is set to 1, transmission is enabled. The SCK pin is used as a transmit clock output pin during the I/O interface mode.

As in the asynchronous mode, transmission is started by any of the following three methods:

- (i) By setting the transmission enable state when the transmit buffer (TxB) is empty, a transmission completion interrupt request is generated and send data is written into the transmit buffer in the interrupt service.
- (ii) When send data is transferred to the transmit buffer (TxB) in the transmission enable state, the send data is transmitted consecutively after the preceding transmission operation terminates.
- (iii) When send data is prewritten into the transmit buffer (TxB) in the transmission disable state and then the transmission enable state is set, the data retained in the transmit buffer (TxB) is transmitted.

When the transmit enable state is set, the send data in the transmit buffer (TxB) is sent starting at the most significant bit (MSB) on the transmit clock falling edge.

Figure 4-53 Transmission Timing during I/O Interface Mode



When the transmit buffer (TxB) becomes empty, a transmission completion interrupt request occurs immediately. When  $\overline{\text{RESET}}$  is input, the transmit buffer (TxB) is cleared. If the transmission enable state is set at the time, a transmission completion interrupt request occurs. When transmission operation starts and the send data in the transmit buffer (TxB) is transferred to the shift register, the transmit buffer (TxB) becomes empty and a transmission completion interrupt request occurs.

(b) Reception

When serial communication mode register (SCM) bit 6 (RXE) is set to 1, reception is enabled. Receive data is input to the serial register on the receive clock rising edge. When the serial register receives 8-bit data, the data is transferred from the serial register to the receive buffer (RxB) and a reception completion interrupt request is generated.

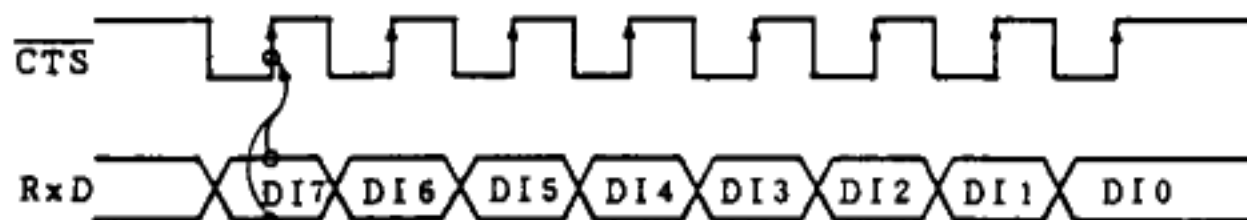
Receive clock in the I/O interface mode can be selected among external and internal receive clocks as specified in serial communication mode register (SCM) bit 2 (RSCK).

The  $\overline{\text{CTS}}$  pin functions as a receive clock input/output pin during the I/O interface mode.

When data is received, if new data is transferred to the receive buffer (RxB) when the receive buffer is full (overrun error), the reception error flag is set to 1 and a reception error interrupt request occurs.

If RXE is set to 0 (reception disable) during reception operation, the receive clock counting counter stops with the value retained. To receive new 8-bit data, set the TSK bit to 1 and clear the receive clock counting counter.

Figure 4-54 Reception Timing during I/O Interface Mode



#### 4.4.5 SERIAL COMMUNICATION INTERFACE INTERRUPT REQUESTS

There are three types of interrupt requests occurring from the serial communication interface: Transmission completion interrupt request, reception completion interrupt request, and reception error interrupt request.

- (1) Serial communication interrupt request control registers (SEIC, SRIC, and STIC)

The SEIC, SRIC, and STIC registers are used to control transmission completion interrupt request (STF), reception completion interrupt request (SRF), and reception error interrupt request (SEF). The three interrupt request control registers form a group and the serial communication interface interrupt request priority levels are programmable by using SEIC register PR2 to PR0 bits. In the group, the priority levels are determined by the hardware as follows:

SEF > SRF > STF

Figure 4-55 Interrupt Request Control Register Format

	7	6	5	4	3	2	1	0	Address	After Reset
SEIC	SEF	SEMK	0	SE CSE	0	PR2	PR1	PR0	FFDAH	47H
SRIC	SRF	SRMK	SR ISM	SR CSE	0	-	-	-	FFDCH	47H
STIC	STF	SEMK	ST ISM	ST CSE	0	-	-	-	FFDEH	47H

Remarks: "-" denotes that write cannot be made. If the bit is read, 1 is read.



The SEF, SRF, and STF bits are interrupt request flags. When a reception error occurs, the SEF bit is set to 1; when reception is complete, the SRF bit is set to 1; and when transmission is complete, the STF bit is set to 1. When an interrupt request is acknowledged or the clear instruction is executed, the bits are reset to 0.

Other bit fields are described in "CHAPTER 5. INTERRUPT FUNCTION".

(2) Serial communication macro service control registers (SRMS and STMS)

SRMS is an 8-bit register to specify the macro service processing mode and channel accompanying the serial communication interface reception completion. STMS is an 8-bit register to specify the macro service processing mode and channel accompanying the serial communication interface transmission completion. The macro service control register bits are described in "5.2 MACRO SERVICE FUNCTION".

Figure 4-56 Macro Service Control Register Formats

	7	6	5	4	3	2	1	0	Address	After Reset
SRMS	MSM2	MSM1	MSM0	DIR	0	CH2	CH1	CH0	FFDDH	Undefined
STMS	MSM2	MSM1	MSM0	DIR	0	CH2	CH1	CH0	FFDFH	Undefined



## 4.5 ANALOG-TO-DIGITAL CONVERTER

The uPD78312A contains an 8-bit high-speed, high-resolution analog-to-digital (A/D) converter which has four multiplexed analog inputs (AN3 to AN0). The analog-to-digital converter contains an analog-to-digital conversion result register (ADCR) which adopts successive approximation and retains the conversion result.

### 4.5.1 ANALOG-TO-DIGITAL CONVERTER CONFIGURATION

Figure 4-57 shows the analog-to-digital converter configuration.

Four analog inputs are multiplexed on the chip and one is selected as specified in the analog-to-digital converter mode register (ADM).

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

The series resistor string is connected between the A/D reference voltage pin ( $AV_{REF}$ ) and A/D ground ( $AV_{SS}$ ); it consists of 255 equivalent resistors and two resistors of the half resistance value of the resistor to make 256 equivalent voltage steps between the two pins.

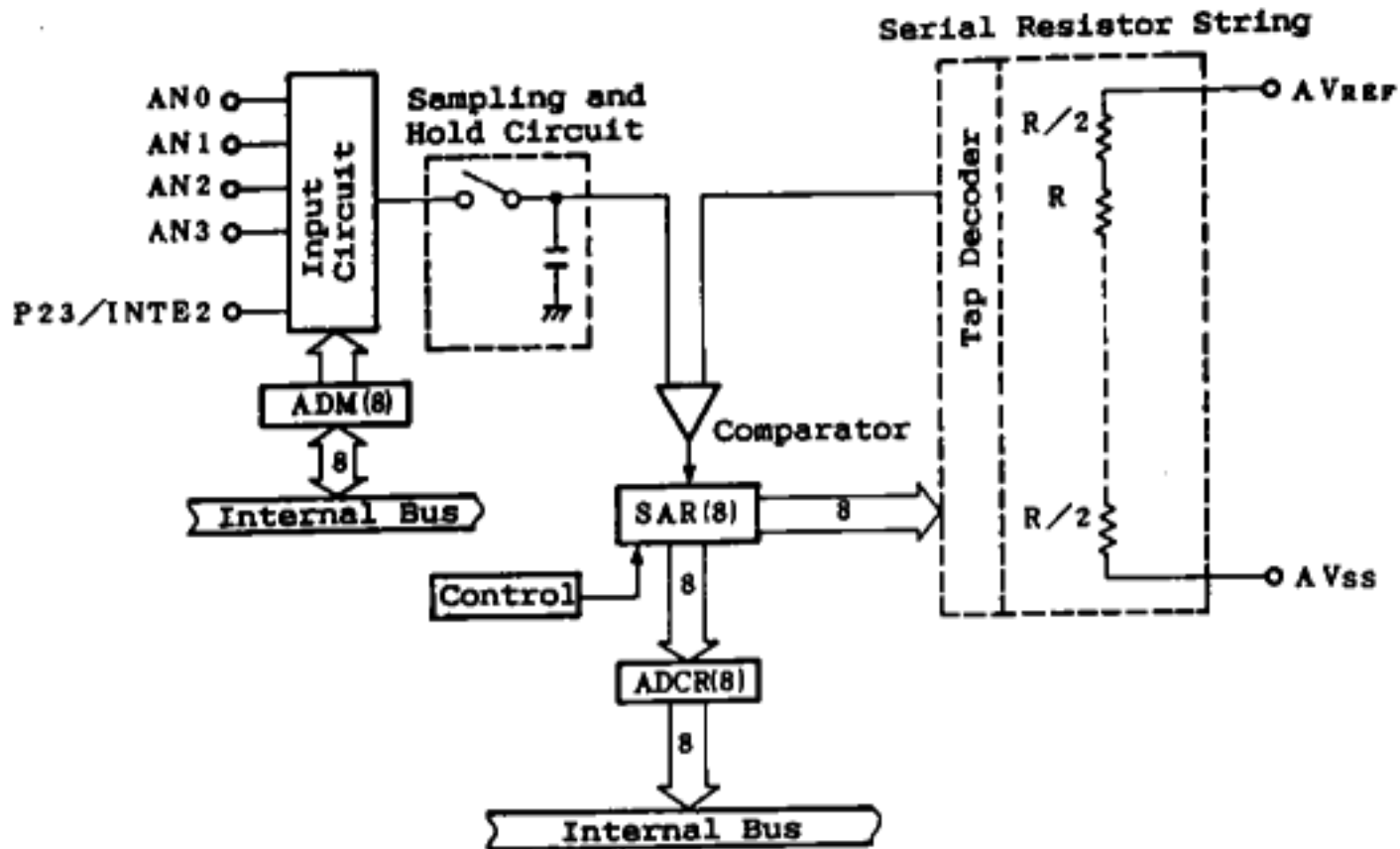
Series resistor string voltage tap is selected by the tap decoder. This decoder is driven by the 8-bit successive approximation register (SAR).

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. That is, when conversion starts, the MSB of SAR is set to 1 and the series resistor string voltage tap is set to  $1/2 AV_{REF}$ . It is compared with analog input. If analog input is greater than  $1/2 AV_{REF}$ , the MSB of SAR remains set and the next most significant bit (bit 7) is compared. If analog input is less than  $1/2 AV_{REF}$ , the MSB is reset to 0 and the next most significant bit (bit 7) is compared. On bit 7, the series resistor string voltage tap is set to  $3/4 AV_{REF}$  or  $1/4 AV_{REF}$  and it is compared with analog input. Such comparison is continued until the least significant bit of SAR (binary search method).

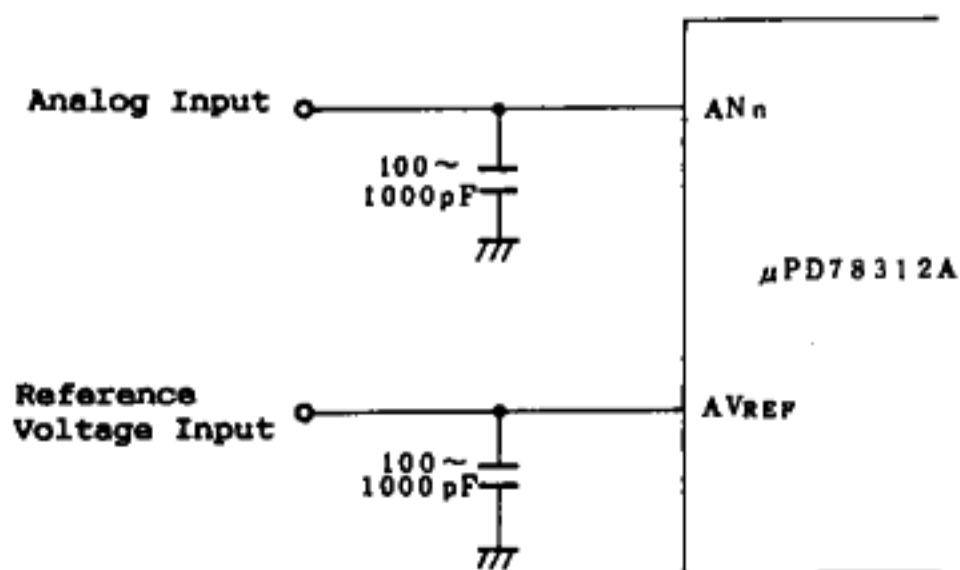
When 8-bit comparison terminates, SAR retains the valid digital result. The result is latch-input to the ADCR register and the interrupt request flag (ADF) is set to 1.

NOTE: Do not apply voltage to the AN3 to AN0 pins beyond the range of  $AV_{SS}$  to  $AV_{REF}$  regardless of whether or not the analog-to-digital converter is used.

Figure 4-57 A/D Converter Block Diagram



NOTE: Connect a capacitor to the analog input pins (AN3 to AN0) and reference voltage input pin ( $AV_{REF}$ ) to prevent noise from causing malfunction.



#### 4.5.2 ANALOG-TO-DIGITAL CONVERTER MODE REGISTER (ADM)

The analog-to-digital converter mode register (ADM) is used to control analog-to-digital converter operation. Figure 4-58 shows the ADM register format.

Bit 0 (MS) is used to control the operating mode. Bits 1 and 2 (ANIO and ANI1) are used to select analog input to be converted into digital form.

Bit 4 (FR) is a control bit to suppress large change of the analog-to-digital conversion time if the oscillation frequency is changed. One conversion speed determined by the internal system clock ( $f_{CLK}$ ) and FR bit can be calculated from the following expression. Table 4-16 lists the conversion speed for each setting.

- When  $f_{CLK} \leq 4$  MHz, the FR bit is set to 1.
- FR = 0: Conversion speed =  $180/f_{CLK}$  (us)
- FR = 1: Conversion speed =  $120/f_{CLK}$  (us)
- $f_{CLK}$  : Internal system clock frequency (MHz)

Table 4-16 Internal System Clock and FR Bit Setting

Internal System Clock	6 MHz	5 MHz	4 MHz	3 MHz	2 MHz
FR bit	0	0	1	1	1
Conversion speed	30 us	36 us	30 us	40 us	60 us

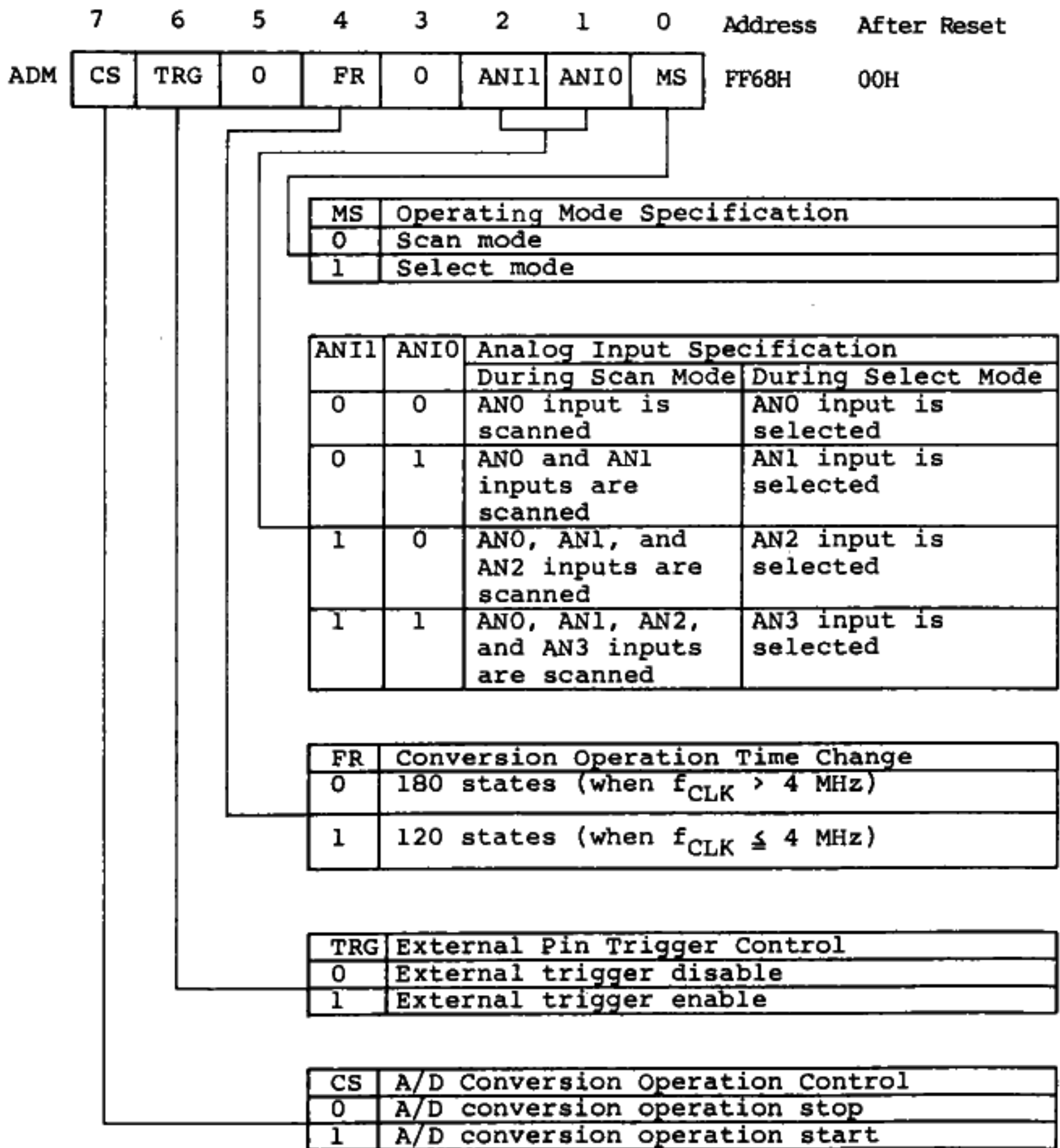
Bit 6 (TRG) is used to enable external synchronization of analog-to-digital conversion operation. When the TRG bit is reset to 0, each time one analog-to-digital conversion operation terminates, the next conversion operation is started. When the TRG bit is set to 1, analog-to-digital operation is stopped. When a valid edge is input to the external trigger pin (INTE2), conversion operation is initialized and performed consecutively. After this, each time a valid edge is input to the INTE2 pin, the conversion operation sequence is initialized.

Bit 7 (CS) is used to control analog-to-digital conversion operation. When the CS bit is set to 1, conversion operation is started; when the CS bit is reset to 0, every conversion operation is stopped although conversion operation is being performed.

At this time, the ADCR register is not updated and the ADF flag is not set.

When **RESET** is input, the ADM register is cleared (00H).

Figure 4-58 Analog-to-Digital Converter Mode Register Format



### 4.5.3 ANALOG-TO-DIGITAL CONVERTER OPERATION

When ADM register bit 7 (CS) is set to 1, analog-to-digital conversion is started. If the ADM register is rewritten during conversion operation, the conversion operation is stopped, then initialized. In this case, the ADF flag is not set and the ADCR register is not updated.

If external pin trigger is enabled by setting ADM register bit 6 (TRG) to 1, conversion operation is stopped. When a valid edge is input to the external trigger pin (INTE2), conversion operation is initialized, then started. Once the conversion operation is started, it is continued as specified in the ADM register. After this, each time a valid edge is input to the INTE2 pin, conversion operation is initialized.

The ADCR register is not affected by  $\overline{\text{RESET}}$  input.

The scan mode or select mode can be selected for the analog-to-digital converter by using analog channel mode register (ADM) bit 0 (MS).

#### (1) Scan mode

In the scan mode, analog inputs specified by using ADM register bits 2 and 1 (ANI1 and ANI0) are selected and converted in order. For example, when AN1 and AN0 are set to 0 and 1 (01), AN0 and AN1 are scanned repeatedly (ANO → AN1 → AN0 → AN1 → .....).

When conversion operation of the selected analog input terminates, the conversion result is stored in the ADCR register and the interrupt request flag (ADF) is set. Then, the next analog input conversion operation is started regardless of whether or not an interrupt request is acknowledged. This operation sequence is continued until the ADM register is rewritten.

#### (2) Select mode

In the select mode, one of four analog inputs is selected in ADM register bits 2 and 1 (ANI1 and ANI0) and is converted.

When conversion operation of the selected analog input terminates, the conversion result is stored in the ADCR register and the interrupt request flag (ADF) is set. Then, conversion operation of the same analog input is started again regardless of whether or not an interrupt request is acknowledged. This operation sequence is continued until the ADM register is rewritten.

#### 4.5.4 ANALOG-TO-DIGITAL CONVERTER INTERRUPT REQUESTS

Interrupt requests occurring from the analog-to-digital converter are controlled by using the analog-to-digital converter interrupt request control register and analog-to-digital converter macro service control register.

- (1) Analog-to-digital converter interrupt request control register (ADIC)

The ADIC register is an 8-bit register to control an interrupt request occurring from the analog-to-digital converter. The ADIC register and the time base counter interrupt control register (TBIC) form a group and the priority level is programmable by using the ADIC register PR2 to PR0 bits in the group unit. In the group, the order of priority is fixed by hardware as follows:

ADF (A/D conversion interrupt request)  
> TBF (time base counter interrupt request)

The interrupt request control register bits are described in "CHAPTER 5. INTERRUPT FUNCTION".

For the time base counter interrupt request, see 4.6.

Figure 4-59 Interrupt Request Control Register Formats

	7	6	5	4	3	2	1	0	Address	After Reset
ADIC	ADF	AD MK	AD ISM	AD CSE	0	PR2	PR1	PR0	FFE0H	47H
TBIC	TBF	TB MK	0	TB CSE	0	-	-	-	FFE2H	47H

Remarks: "-" denotes that write cannot be made.  
If the bit is read, 1 is read.

- (2) Analog-to-digital converter macro service control register (ADMS)

The ADMS register is an 8-bit register to control the macro service started when an interrupt request (ADF) occurs from the analog-to-digital converter. The ADMS register bits are described in "5.2 MACRO SERVICE FUNCTION".

Figure 4-60 Macro Service Control Register Format

	7	6	5	4	3	2	1	0	Address	After Reset
ADMS	MSM2	MSM1	MSM0	DIR	0	CH2	CH1	CH0	FFE1H	Undefined

## 4.6 TIME BASE COUNTER AND WATCHDOG TIMER

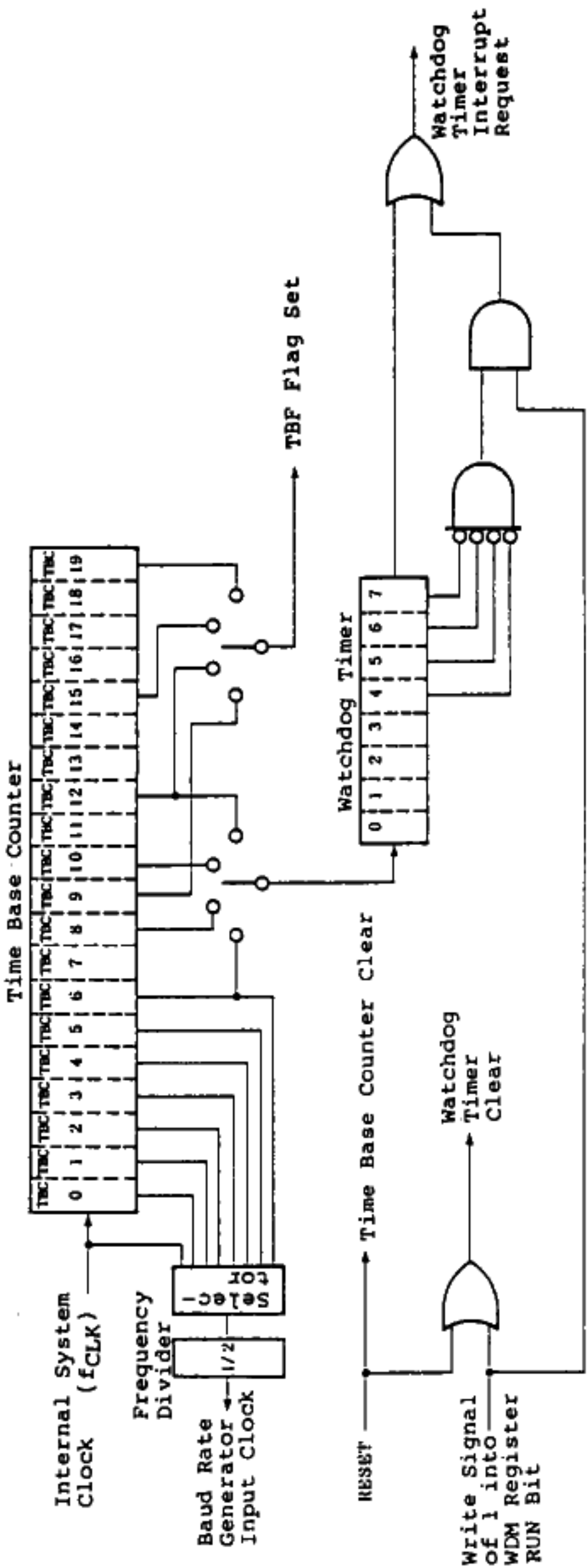
The uPD78312A contains a time base counter to provide time base output used as reference for various types of processing and a watchdog timer to detect software upset.

### 4.6.1 TIME BASE COUNTER AND WATCHDOG TIMER CONFIGURATION

This block consists of a 20-bit time base counter and an 8-bit watchdog timer which uses time base counter tap output as clock source as shown in Figure 4-61.



Figure 4-61 Time Base Counter and Watchdog Timer Block Diagram



#### 4.6.2 TIME BASE COUNTER (TBC)

##### (1) Time base counter operation

The time base counter (TBC) is a 20-bit counter which divides internal system clock ( $f_{CLK}$ ).

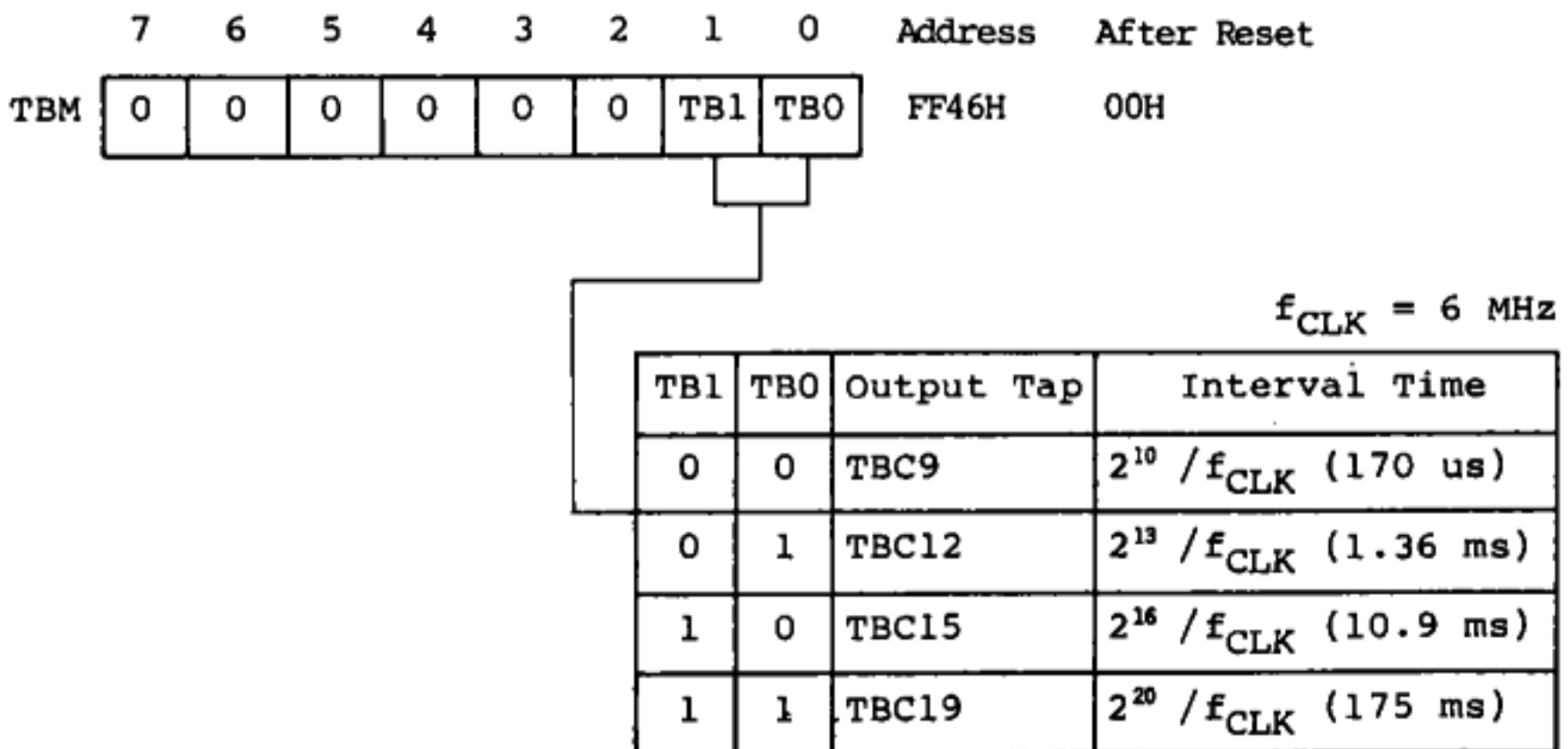
The interrupt request flag (TBF) is set to 1 on the falling edge of time base counter tap output specified by using the time base mode register (TBM). Figure 4-62 shows the TBM register format.

The interval time of an interrupt request (TBF flag set) occurring from the time base counter can be selected among four types shown in Figure 4-62 by setting the TBM register.

The time base counter is also used as capture or, PWM unit free running counter (FRC). (See "4.3.2 CAPTURE UNIT".)

The time base counter is cleared to 0 only when  $\overline{RESET}$  is input. After this, the time base counter continues to be incremented; it cannot be stopped.

Figure 4-62 Time Base Mode Register Format



Remarks:  $f_{CLK}$ : Internal system clock frequency

NOTE: The time to the first interrupt request occurrence just after the TBM register is set becomes undefined.

(2) Time base counter interrupt request control register (TBIC)

The TBIC register is an 8-bit register to control interrupt requests occurring from the time base counter. The TBIC register and the A/D converter interrupt request control register (ADIC) form a group and the order of priority is programmable in the group unit. In the group, the priority levels are fixed by hardware as follows:

ADF (A/D conversion interrupt request)  
> TBF (time base counter interrupt request)

TBIC register bit 7 (TBF) is a time base counter interrupt request flag. When the TBC tap output selected in the TBM register is once set to 1 and then falls, the flag is set to 1.

When an interrupt request is acknowledged or a clear instruction is executed, the TBF flag is reset to 0.

#### 4.6.3 WATCHDOG TIMER (WDT)

The watchdog timer is an 8-bit timer which uses time base counter (TBC) tap output as clock source. It generates a nonmaskable interrupt at the specified intervals. The watchdog timer is controlled by the watchdog timer mode register (WDM).

(1) Watchdog timer mode register (WDM)

The watchdog timer mode register (WDM) is an 8-bit register to control watchdog timer operation. The WDM register is written only by executing a special instruction to prevent the WDM register contents from being easily rewritten due to software upset. When RESET is input, the WDM register is cleared (00H).

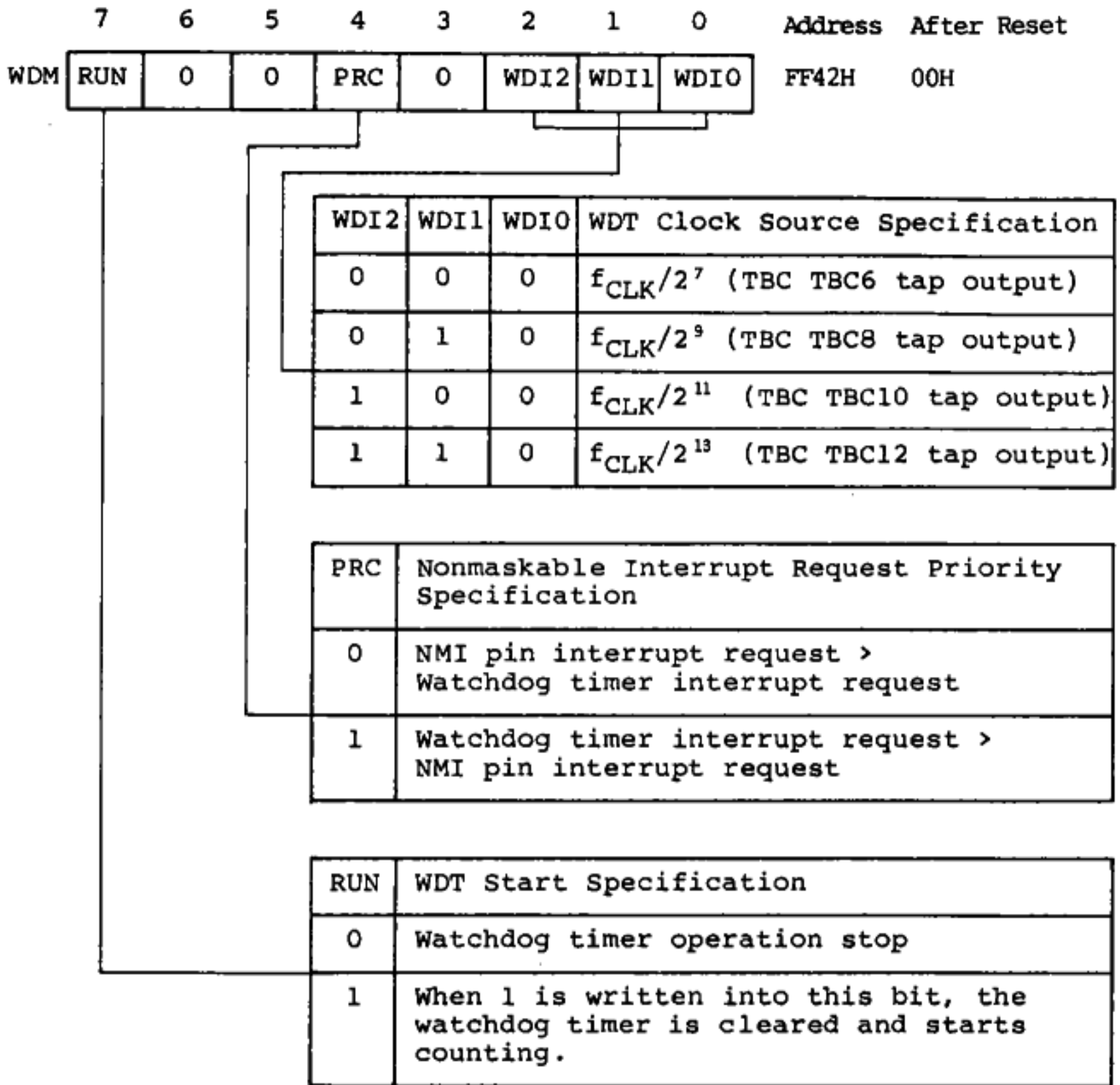
Bit 7 (RUN) is used to control the watchdog timer count start. When the RUN bit is reset to 0, the watchdog timer stops; when 1 is written into the RUN bit, the watchdog timer is cleared (00H) and starts counting.

Bits 2 to 0 (WDI2 to WDI0) are used to select watchdog timer count clock; time base counter output tap is specified.

Bit 4 (PRC) is used to specify the priority levels of nonmaskable interrupt request occurring from the watchdog timer and nonmaskable interrupt request caused when a valid edge is input to the NMI pin.

Figure 4-63 shows the WDM register format.

Figure 4-63 Watchdog Timer Mode Register Format



(2) Watchdog timer operation

When 1 is written into WDM register bit 7 (RUN), the watchdog timer is cleared (00H) and starts counting the TBC tap output specified in WDM register bits 2 to 0 (WDI2 to WDIO).

A nonmaskable interrupt request occurs from the watchdog timer when clear operation is not performed until the watchdog timer overflows or when the watchdog timer is cleared (00H) by the time watchdog timer bit 4 is first set to 1 (except just after the 0 to 1 transition of the WDM RUN bit is made).

The priority levels between nonmaskable interrupt request occurring from the watchdog timer and nonmaskable interrupt request when a valid edge is input to the NMI pin can be specified by setting watchdog timer mode register (WDM) bit 4 (PRC) as follows:

PRC = 0: NMI > WDT

PRC = 1: WDT > NMI

Table 4-17 lists the watchdog timer overflow time and the time until bit 4 is first set to 1 after count starts (6.25% of the overflow time) when internal system clock is set to 6 MHz.

Table 4-17 Watchdog Timer Count Clock and Overflow Time

( $f_{CLK} = 6 \text{ MHz}$ )

Watchdog Timer Count Clock Frequency	Overflow Time	6.25% of Overflow Time
$f_{CLK}/2^7$ (TBC6 tap output)	5.5 ms	343 us
$f_{CLK}/2^9$ (TBC8 tap output)	21.8 ms	1.36 ms
$f_{CLK}/2^{11}$ (TBC10 tap output)	87.4 ms	5.46 ms
$f_{CLK}/2^{13}$ (TBC12 tap output)	349.5 ms	22 ms

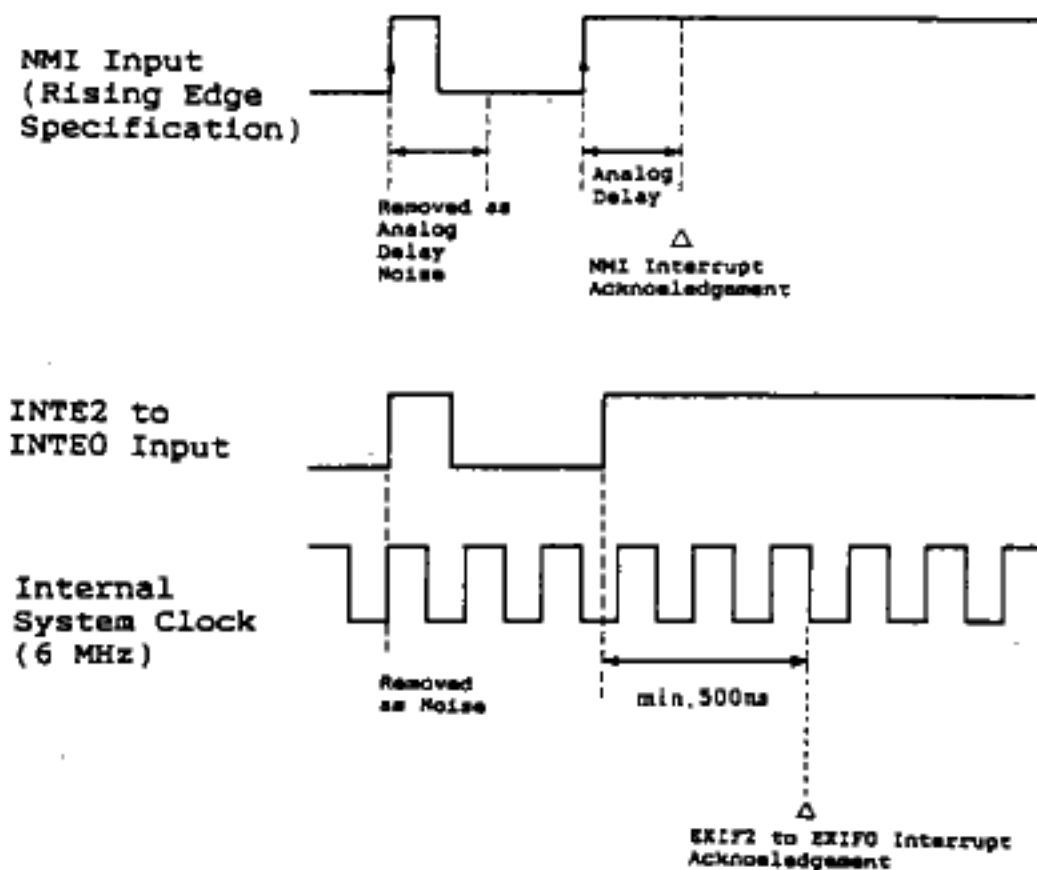
#### 4.7 EXTERNAL INTERRUPT REQUEST FUNCTION

When a valid edge specified in the external interrupt mode register (INTM) is input to one of the INTE0 to INTE2 pins or the NMI pin, an external interrupt request occurs.

The external interrupt pins contain noise removal circuits to prevent noise from causing malfunction.

The NMI input pin contains noise removal circuit using analog delay. Other external interrupt input pins, INTE2 to INTE0 contain digital noise removal circuits using internal system clocks. To prevent removal of signal as noise, signal whose width is wider than analog delay must be input to the NMI pin; signal whose width is wider than three system clocks must be input to the INTE2 to INTE0 pins after a valid edge is input.

Figure 4-64 External Interrupt Request Pin Input Noise Removal



##### (1) External interrupt mode register (INTM)

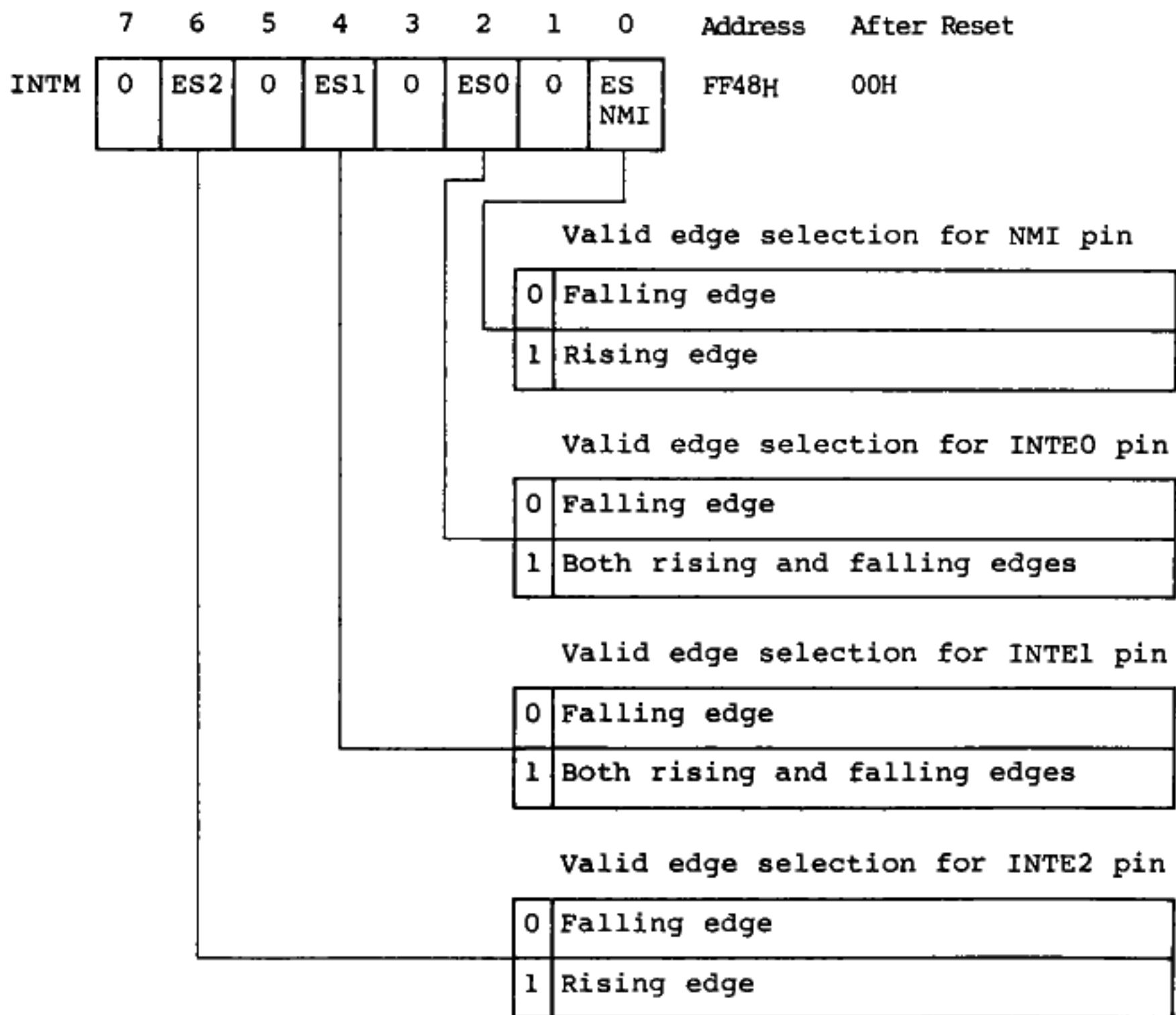
The external interrupt mode register (INTM) is an 8-bit register to specify valid edges input to the INTE0 to INTE2 pins and NMI pin. When RESET is input, the INTM register is cleared (00H).

The INTM register ESNMI bit is used to specify the valid edge input to the NMI pin. When the ESNMI bit is reset to 0, the falling edge is selected; when the bit is set to 1, the rising edge is selected for the valid edge.

The ES0 and ES1 bits are used to specify valid edges input to the INTE0 and INTE1 pins. When the ES0 (ES1) bit is reset to 0, the falling edge is selected; when the bit is set to 1, both rising and falling edges are selected for valid edges.

The ES2 bit is used to specify the valid edge input to the INTE2 pin. When the ES2 bit is reset to 0, the falling edge is selected; when the bit is set to 1, the rising edge is selected for the valid edge.

Figure 4-65 External Interrupt Mode Register Format



(2) External interrupt pin interrupt request control registers (EXIC0 to EXIC2)

The EXIC0 to EXIC2 registers are used to control interrupt requests occurring from the external interrupt pins INTE0 to INTE2. The three interrupt request registers form a group and the external interrupt pin interrupt request priority levels are programmable by using the EXIC0 register PR2 to PR0 bits. In the group, the priority levels are determined by hardware as follows:

$$\text{INTE0} > \text{INTE1} > \text{INTE2}$$

Figure 4-66 Interrupt Request Control Register Formats

	7	6	5	4	3	2	1	0	Address	After Reset
EXIC0	EXIF0	EXI MK0	EXI ISM0	EXI CSE0	0	PR2	PR1	PR0	FFC8H	47H
EXIC1	EXIF1	EXI MK1	EXI ISM1	EXI CSE1	0	-	-	-	FFCAH	47H
EXIC2	EXIF2	EXI MK2	EXI ISM2	EXI CSE2	0	-	-	-	FFCCH	47H

Remarks: "-" denotes that write cannot be made. If the bit is read, 1 is read.

The EXIF0, EXIF1, and EXIF2 bits are interrupt request flags. When a valid edge is input to the INTE0, INTE1, or INTE2 pin, the EXIF0, EXIF1, or EXIF2 bit is set to 1. The bit is reset to 0 when interrupt request is acknowledged or by software. Other bits are explained in "CHAPTER 5 INTERRUPT FUNCTION".

(3) External interrupt pin macro service control registers (EXMS0 to EXMS2)

Each of EXMS0 to EXMS2 is an 8-bit register to specify the operating mode of the macro service executed when external interrupt pin interrupt request occurs.



Figure 4-67 Macro Service Control Register Formats

	7	6	5	4	3	2	1	0	Address	After Reset
EXMS0	MSM 2	MSM 1	MSM 0	DIR	0	CH2	CH1	CH0	FFC9H	Undefined
EXMS1	MSM 2	MSM 1	MSM 0	DIR	0	CH2	CH1	CH0	FFCBH	Undefined
EXMS2	MSM 2	MSM 1	MSM 0	DIR	0	CH2	CH1	CH0	FFCDH	Undefined

## CHAPTER 5. INTERRUPT FUNCTION

The uPD78312A handles interrupt requests occurring from on-chip peripheral hardware and the external by executing vectored interrupt service centering around software processing or macro service function processing in which data is only transferred by hardware.

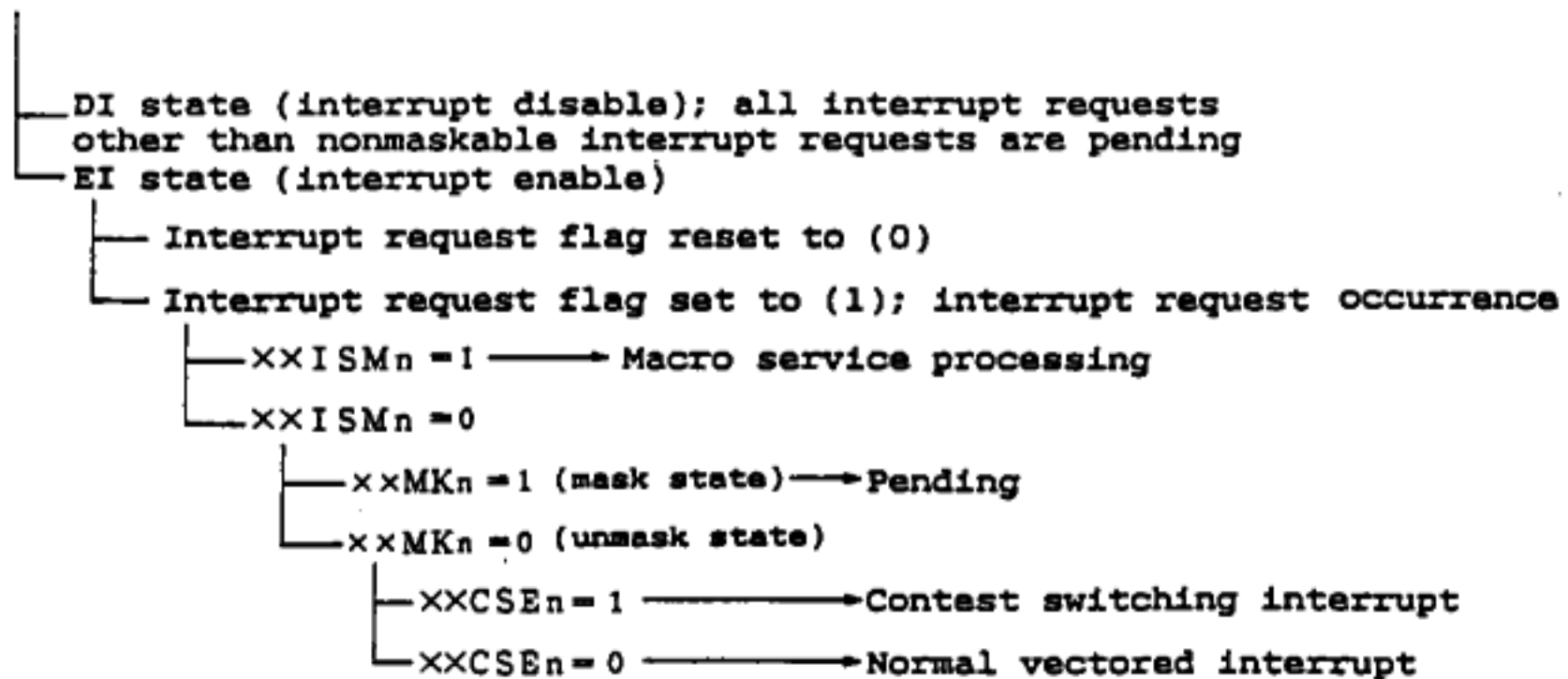
The uPD78312A also contains the context switching function which performs register bank switching by hardware, exchanges the PC contents, and saves the PSW contents.

Interrupt requests are classified into the following three types:

- o Nonmaskable interrupt requests
- o Maskable interrupt requests
- o Software interrupt requests

Figure 5-1 shows the interrupt request processing modes. Table 5-2 lists the interrupt request sources.

Figure 5-1 Interrupt Request Processing Modes



Remarks: xxMKn, xxISMn, and xxCSEn are interrupt request register bits 6, 5, and 4.

Table 5-1 lists the maximum wait time until operation code prefetch of the first instruction of the interrupt service routine starts after an interrupt request occurs.

In the table, the number of vectored interrupt automatic save processing states equals to 16 when stack memory and branch destination vector table are set in internal memory. If stack memory is set in external memory,  $8 + 4m$  states increase; if the branch destination vector table is set in external memory,  $16 + 4m$  states increase. (m is the number of states inserted as specified in the MM register.)

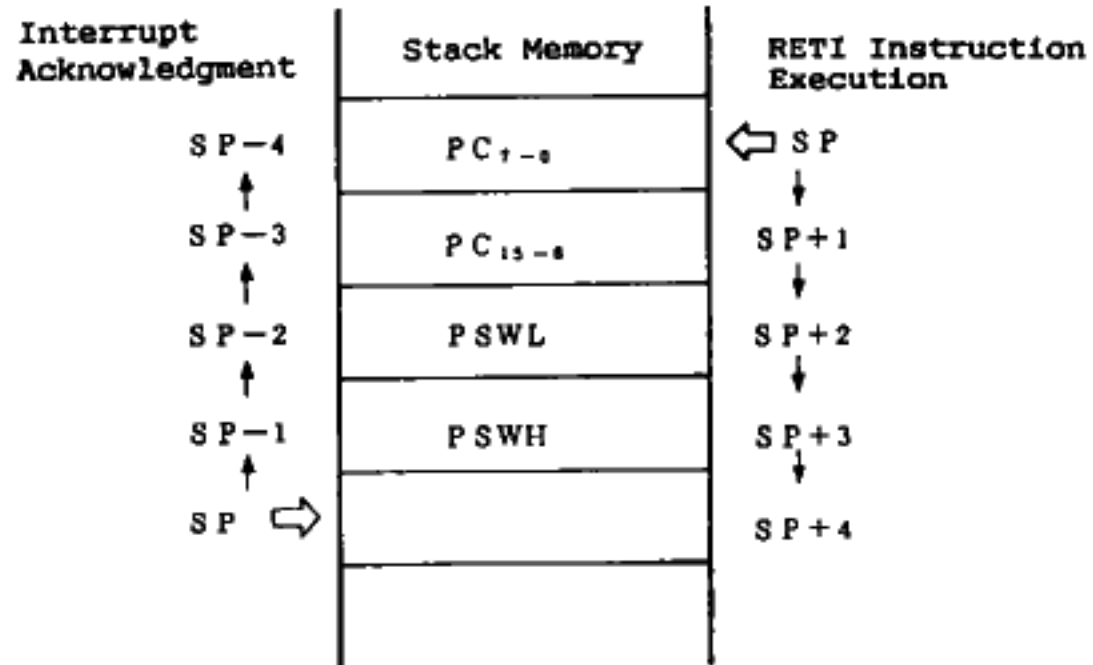
Table 5-1 Maximum Wait Time Until Interrupt is Acknowledged

Wait Time Source	Number of States	
	Vectored Interrupt	Context Switching
Interrupt priority level scan	max. 16	
Execution instruction processing time when interrupt is acknowledged	max. 50 (DIVUX instruction)	
Time required for automatic save processing	16 (64)	12
Total	82 (130)	78

Remarks 1: The value enclosed in parentheses is the number of required states when stack memory and branch destination vector table are set in external memory and three wait states are inserted.

2: For an interrupt request caused when a valid edge is input to an external interrupt pin, the time required for noise removal is added to the value in Table 5-1. (A maximum of two states)

Figure 5-2 Automatically Saved/Restored Data by Interrupt Request Service

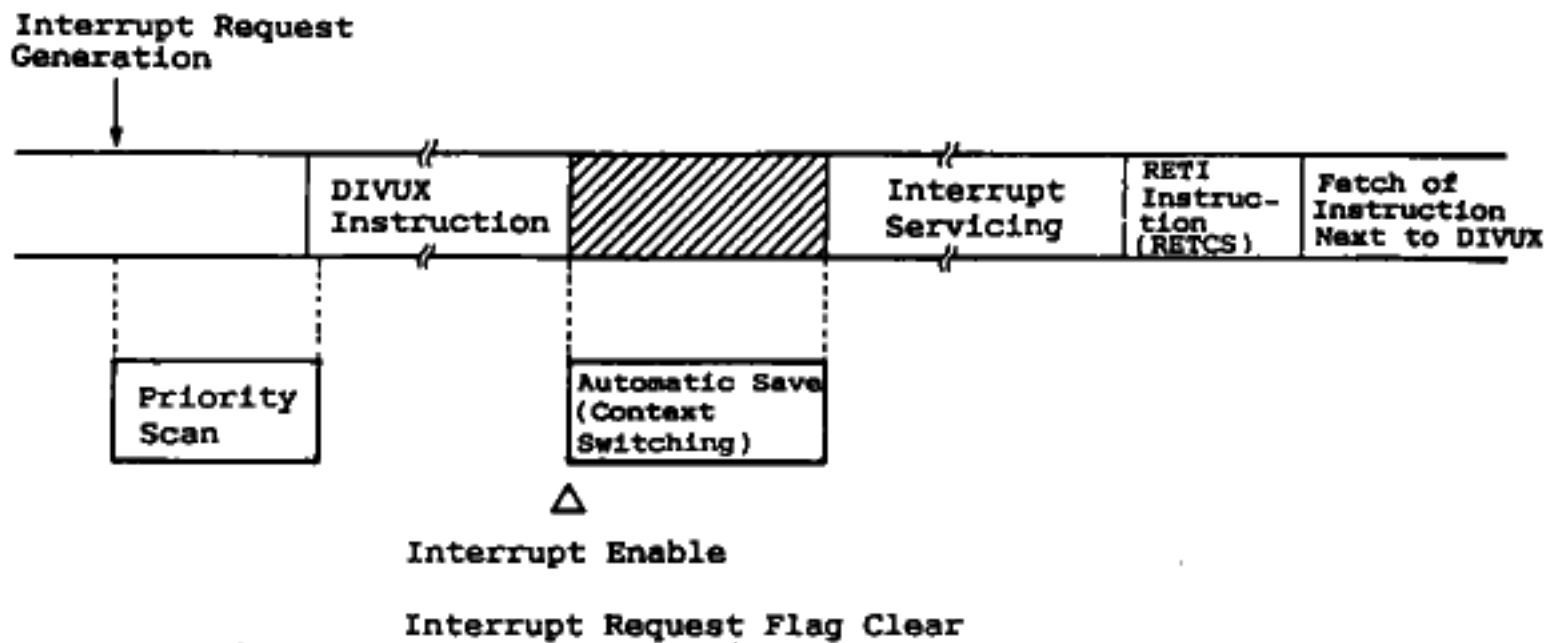


Remarks: SP = Stack Pointer

Figure 5-3 shows interrupt request service and macro service processing sequences in which the DIVUX instruction is executed when an interrupt is acknowledged as an example.

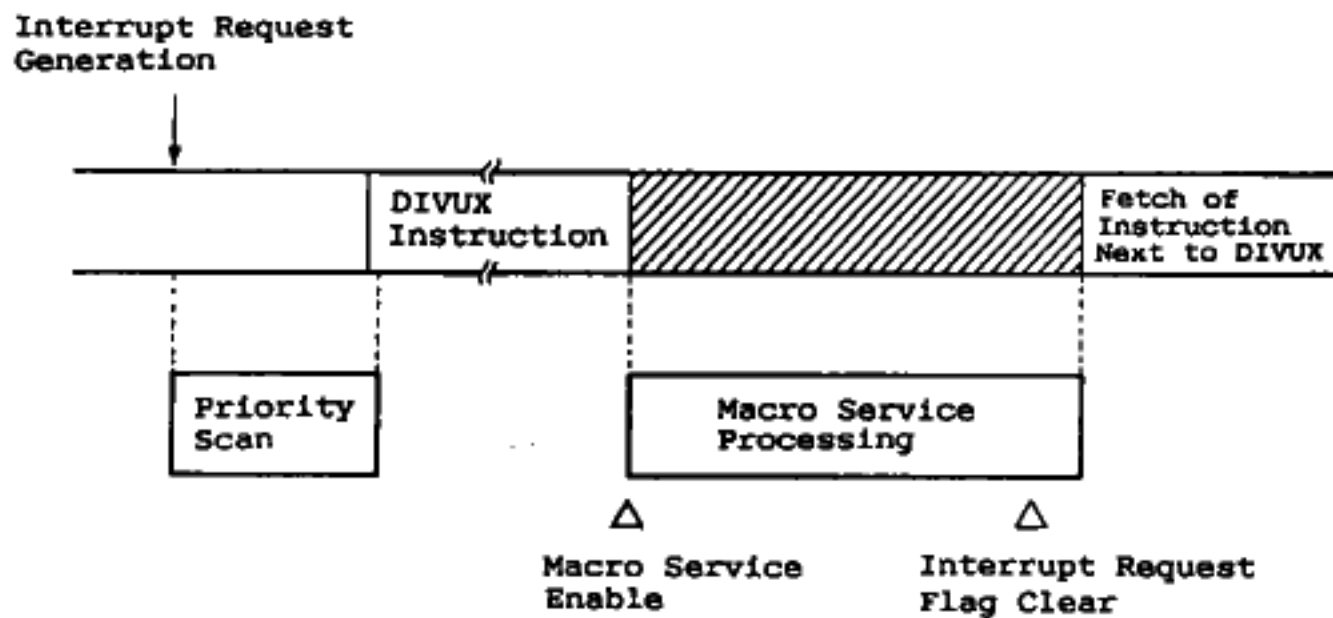
Figure 5-3 Interrupt Service Sequence

(a) Interrupt request service sequence



Remarks: ( ): When interrupt service is made by the context switching function.

(b) Macro service processing sequence



Remarks: Refer to Table 5-4 for the number of states required for macro service processing.

Interrupt priority level scan is made by the 3-bit scan counter. The scan counter performs count operation as 0 + 1 + ...7 + 0 + ... After an interrupt request occurs, if the scan counter value matches the priority level of the interrupt, the scan counter stops and the interrupt is acknowledged. Unless they match, the interrupt request is held. When the interrupt is acknowledged, after the instruction being executed terminates, automatic save operation is performed and the interrupt service routine is entered. Since the scan counter counts one when two system clocks are input, two to 16 states are required for priority level scan. When the interrupt request control register or macro service control register is rewritten, the ISPR register is reset when a return is made from the interrupt service routine, or the IE flag is set to (1), the scan counter is cleared and again starts counting at "0".

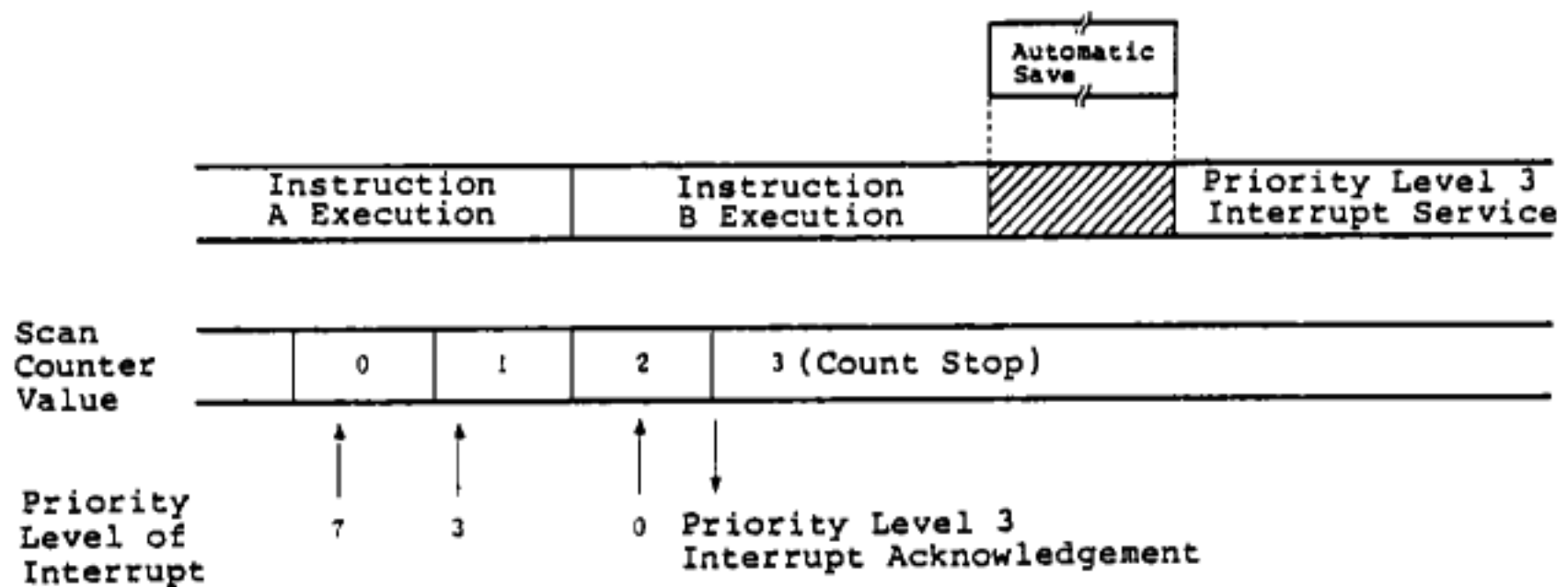


Table 5-2 Interrupt Request Source List

Interrupt Request Type	Default Priority	Interrupt Request Source		Macro Service	Vector Table Address
Software	-	BRK	Break instruction	-	003EH
Nonmaskable	-	NMI	NMI pin input	-	0002H
	-	WDT	Watchdog timer	-	000AH
Maskable	0	CRF00	Count unit	o	001AH
	1	CRF01	Count unit	-	001CH
	2	CRF10	Count unit	o	001EH
	3	CRF11	Count unit	-	0020H
Maskable	4	EXIF0	INTE0 pin input	o	0004H
	5	EXIF1	INTE1 pin input	o	0006H
	6	EXIF2	INTE2 pin input	o	0008H
Maskable	7	TMF0	Timer unit	o	000EH
	8	TMF1	Timer unit	o	0010H
	9	TMF2	Timer unit	o	0012H
Maskable	10	SEF	Serial interface error	-	0022H
	11	SRF	Serial reception	o	0024H
	12	STF	Serial transmission	o	0026H
Maskable	13	ADF	A/D converter	o	0028H
	14	TBF	Time base counter	-	000CH
Reset	-	RESET	Reset input	-	0000H

Remarks: Default priority: Priority fixed by hardware

## 5.1 INTERRUPT REQUESTS

### 5.1.1 INTERRUPT CONTROL HARDWARE DEVICES

#### (1) Interrupt request control registers

The interrupt request control registers are provided for interrupt request sources such as the serial communication interface, pulse input/output unit, and external interrupt pins and are used for interrupt request priority level control and interrupt mask control. Figure 5-4 shows the interrupt request control register formats.

The PR2 to PR0 bits are a bit field to specify the group priority level for maskable interrupts classified into five groups. The bit field of the interrupt request control register assigned the highest default priority in the group is significant. The bit fields (bits 2 to 0) of other interrupt request control registers cannot be written. If the bit is read, "1" is read. Up to eight priority levels can be specified and the same priority level can also be assigned to more than one group. Level 0 is the highest priority level of maskable interrupt requests.

If more than one interrupt request occurs in a single group or groups assigned the same priority level, the interrupt requests are acknowledged in the default priority order listed in Table 5-2.

The xxISMn bit is used to specify macro service or normal interrupt service for interrupt request processing. The bit is fixed to "0" for interrupt requests not causing a macro service request (normal vectored interrupt service).

The xxMKn bit is an interrupt mask bit of the corresponding interrupt request flag. When the bit is set to (1), its corresponding interrupt request is masked.

The xxFn bit is a peripheral hardware interrupt request flag. The interrupt request flag remains set until it is reset to (0) by software or interrupt request is acknowledged.



The `xxCSEn` bit is used to enable the context switching function (register bank switching). If the `xxCSEn` bit is set to (1), when interrupt service is started, a change is made to the register bank of the same value as the priority level specified in the PR2 to PRO bit field by hardware. (See "5.3 CONTEXT SWITCHING FUNCTION".) When the `xxCSEn` bit is reset to (0), the context switching function is disabled.

When  $\overline{\text{RESET}}$  is input, every interrupt request control register is set to 47H.

Figure 5-4 Interrupt Request Control Register Formats

	7	6	5	4	3	2	1	0	Address	After Reset
CRIC00	CRF00	CRMK00	CRSM00	CRCE00	0	PR2	PR1	PRO	FFC0H	47H
CRIC01	CRF01	CRMK01	0	CRCE01	0	-	-	-	FFC2H	47H
CRIC10	CRF10	CRMK10	CRSM10	CRCE10	0	-	-	-	FFC4H	47H
CRIC11	CRF11	CRMK11	0	CRCE11	0	-	-	-	FFC6H	47H
EXIC0	EXPF0	EXMK0	EXSM0	EXCE0	0	PR2	PR1	PRO	FFC8H	47H
EXIC1	EXPF1	EXMK1	EXSM1	EXCE1	0	-	-	-	FFCAH	47H
EXIC2	EXPF2	EXMK2	EXSM2	EXCE2	0	-	-	-	FFCCH	47H
TMIC0	TMPF0	TMMK0	TMSM0	TMCCE0	0	PR2	PR1	PRO	FFCEH	47H
TMIC1	TMPF1	TMMK1	TMSM1	TMCCE1	0	-	-	-	FFD0H	47H
TMIC2	TMPF2	TMMK2	TMSM2	TMCCE2	0	-	-	-	FFD2H	47H
SEIC	SEF	SEMK	0	SECE	0	PR2	PR1	PRO	FFDAH	47H
SRIC	SRF	SRMK	SRSM	SRCCE	0	-	-	-	FFDCH	47H
STIC	STF	STMK	STSM	STCE	0	-	-	-	FFDEH	47H
ADIC	ADF	ADMK	ADSM	ADCE	0	PR2	PR1	PRO	FFE0H	47H
TBIC	TBF	TBMK	0	TBCE	0	-	-	-	FFE2H	47H

PR2	PR1	PRO	Group Interrupt Request Priority Level Specification
0	0	0	Priority level 0 (highest)
:	:	:	:
1	1	1	Priority level 7 (lowest)

xx	Context Switching Function Control
0	Context switching function disable
1	Context switching function enable

xx	Macro Service Function Control
0	Interrupt service
1	Macro service

xx	Interrupt Request Mask control
0	Interrupt request enable
1	Interrupt request disable (however, the macro service function is not disabled)

xx	Interrupt Request Flag
0	No interrupt request
1	Interrupt is requested

Remarks: -: Write cannot be made. If the bit is read, "1" is read.



### 5.1.2 NONMASKABLE INTERRUPT REQUESTS

A nonmaskable interrupt request is unconditionally acknowledged although DI is set. A nonmaskable interrupt request occurs from the watchdog timer or when a valid edge is input to the NMI pin. Unlike other maskable interrupt requests, programmable priority level control is not applied to the nonmaskable interrupt requests. The relative priority levels between interrupt request caused when a valid edge is input to the NMI pin and interrupt request occurring from the watchdog timer can be specified by using the watchdog timer mode register. (See "4.6 TIME BASE COUNTER, WATCHDOG TIMER FUNCTIONS".)

### 5.1.3 MASKABLE INTERRUPT REQUESTS

Maskable interrupt requests are classified into five groups and priority levels are programmable in group units by using the interrupt request register (eight levels). In one group, default priority levels are determined by hardware. (See Table 5-2.)

When a number of maskable interrupt requests occur, the interrupt request assigned the highest priority level is selected by the priority level judgement circuit and interrupt service or macro service processing is performed for the interrupt request. Unselected interrupt requests are held.

For example, if the same priority level is assigned to two or more groups, the priority levels are determined by the intergroup priority levels listed in Table 5-3.

When an interrupt request is acknowledged, the DI state is set and the subsequent maskable interrupts are disabled. If the EI instruction is executed within the interrupt service, the EI state is set. The interrupt request assigned the higher priority level than the priority level of the interrupt request being acknowledged is enabled. The same or lower priority level of interrupt is not acknowledged.

The maskable interrupt requests are classified into interrupt requests not causing a macro service interrupt and those causing a macro service interrupt.

- (1) Maskable interrupt requests not causing a macro service request

The type of maskable interrupt request causes vectored interrupt service only. Interrupt request control register is provided for each interrupt request source.

(2) Interrupt requests causing a macro service request

This type of maskable interrupt request can cause vectored interrupt and macro service request. Interrupt request control register and macro service control register are provided for each interrupt request source.

Table 5-3 Intergroup Priority Levels

Intragroup Priority Levels (Fixed by Hardware)	Intergroup Priority Level
CRIC00 > CRIC01 > CRIC10 > CRIC11	1
EXIC0 > EXIC1 > EXIC2	2
TMIC0 > TMIC1 > TMIC2	3
SEIC > SRIC > STIC	4
ADIC > TBIC	5

#### 5.1.4 SOFTWARE INTERRUPT REQUESTS

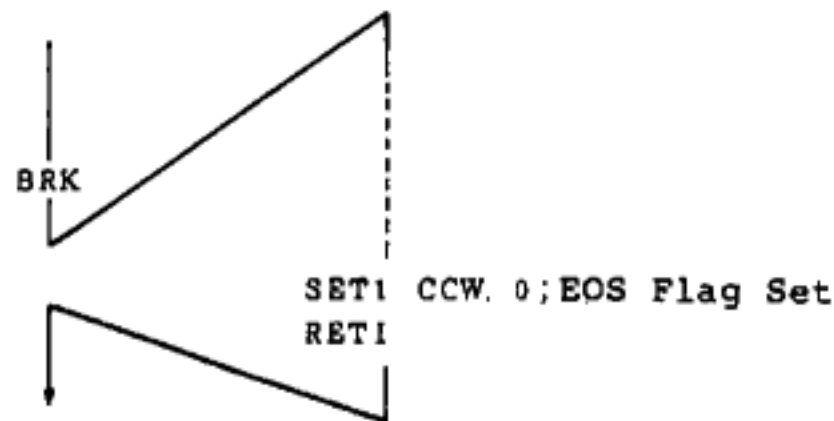
A software interrupt request is caused by BRK instruction execution causing a vectored interrupt or by BRKCS instruction execution starting the context switching function (register bank switching). The BRKCS instruction is described in "5.3.2 CONTEXT SWITCHING FUNCTION WHEN BRKCS INSTRUCTION IS EXECUTED".

The interrupt request caused by BRK instruction execution is also acknowledged in the DI state. Interrupt request priority level control is not applied to the interrupt request.

When the BRK instruction is executed, unconditionally the vector table contents are set in the PC for a branch.

If the BRK instruction is executed within the BRK instruction service routine, nesting can also be made in its own routine.

If the in-service priority register (ISPR) is reset to (0) when a return is made from the BRK instruction service routine, interrupt nesting control is destroyed. Be sure to set the CPU control word (CCW) EOS flag to (1) just before the RETI instruction.

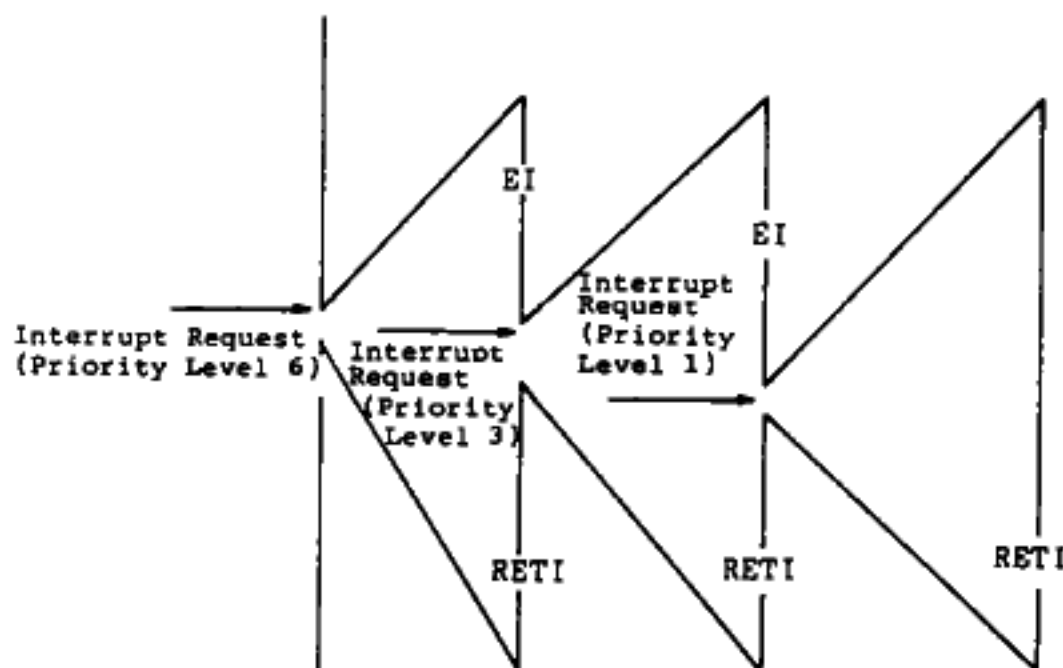


### 5.1.5 MULTIPLE INTERRUPTS

When the EI instruction is executed, the EI state is set and all external and internal unmasked interrupt requests are enabled. If the EI instruction is executed during interrupt service routine execution, only interrupt requests assigned the higher priority level than the interrupt request can be acknowledged. In this case, when a number of interrupt requests occur at the same time, the interrupt request of the higher priority level is acknowledged and the interrupt request of the same or lower priority level is held. When the EI state is set later, the pending interrupt request is acknowledged unless any other higher priority level interrupt request occurs. However, nesting cannot be made in the higher priority level interrupt service routine than the interrupt request.

Other interrupts cannot be nested in the nonmaskable interrupt (NMI or WDT) service routine. The priority level of the nonmaskable interrupt request is set by the watchdog timer mode register (WDM) (see Figure 4-63).

Figure 5-6 3-Level Multiple Interrupts



## 5.2 MACRO SERVICE FUNCTION

The macro service function is provided to lessen the occurrence frequency of interrupts processed mainly by software as much as possible, suppress such overheads as interrupt service, register save, register restore, and return from interrupt service routine, and improve the CPU service time.

When a macro service request occurs, the CPU temporarily stops program processing and 1- or 2-byte data transfer processing is automatically performed between a given special function register (SFR) and memory. A macro service function operation example is described in 5.2.1.

### 5.2.1 MACRO SERVICE FUNCTION USE EXAMPLE

A use example in which analog-to-digital conversion is performed six times by using an analog-to-digital converter and the results are stored in the internal RAM area of FE40H to FE45H is given below. In this example, macro service channel 4 is used.

After data is transferred six times, the transferred data is processed by the interrupt service program.



Figure 5-7 Macro Service Operation Example

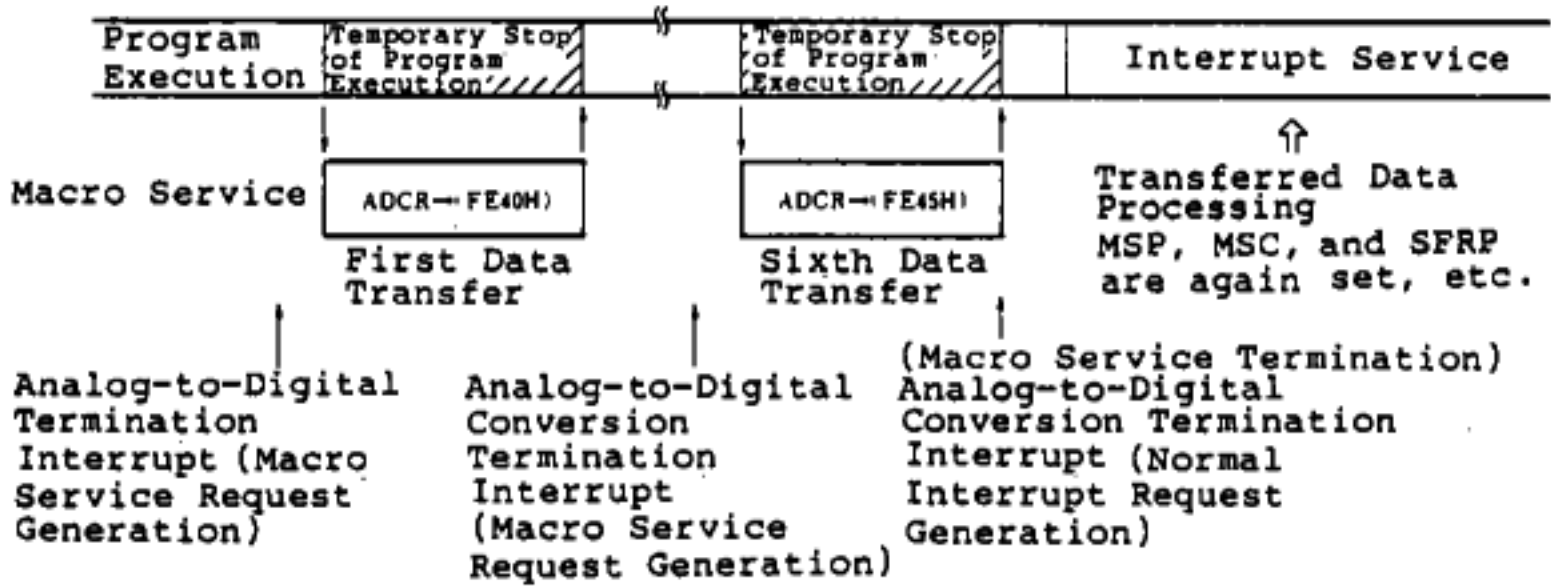
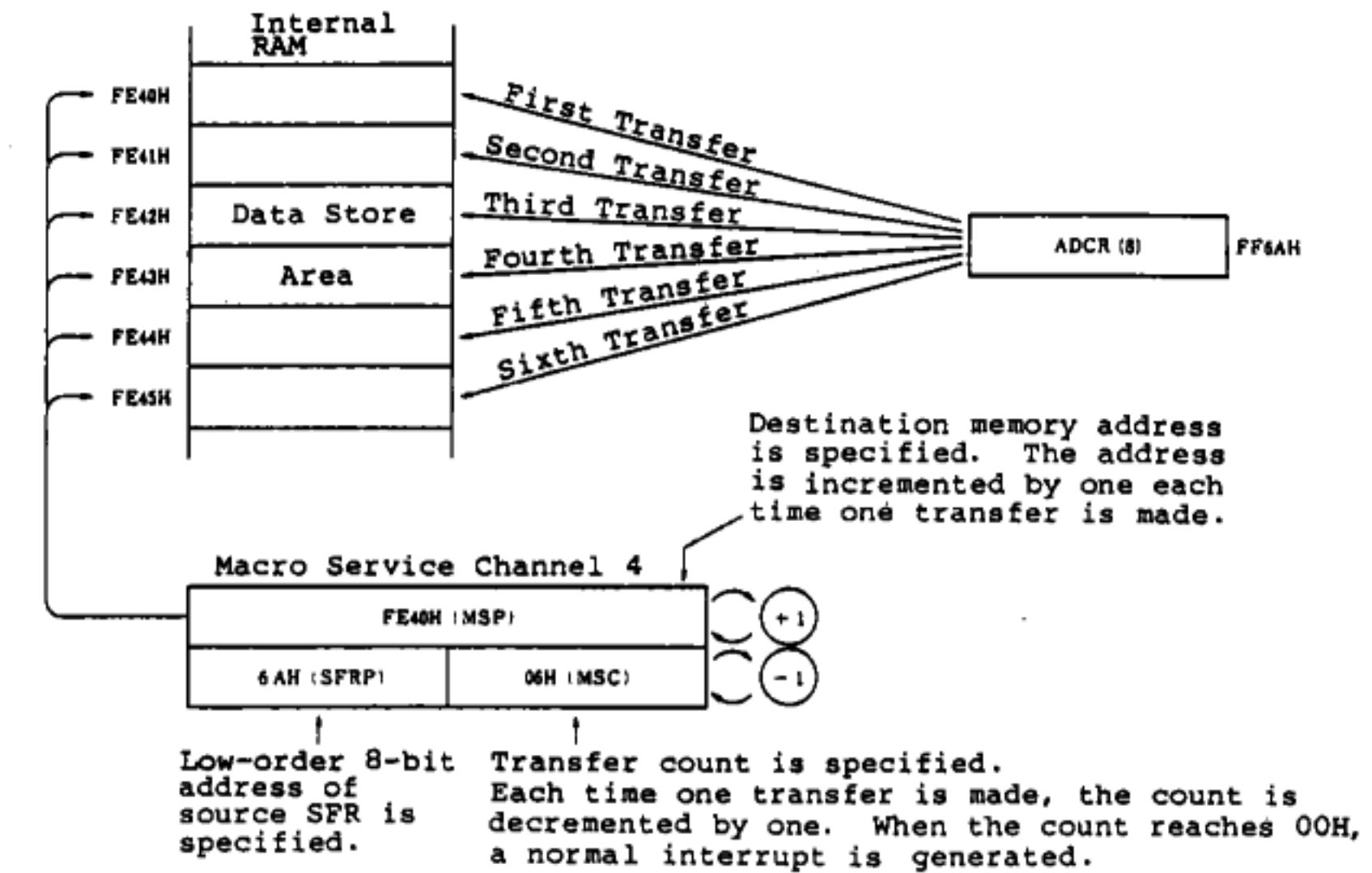
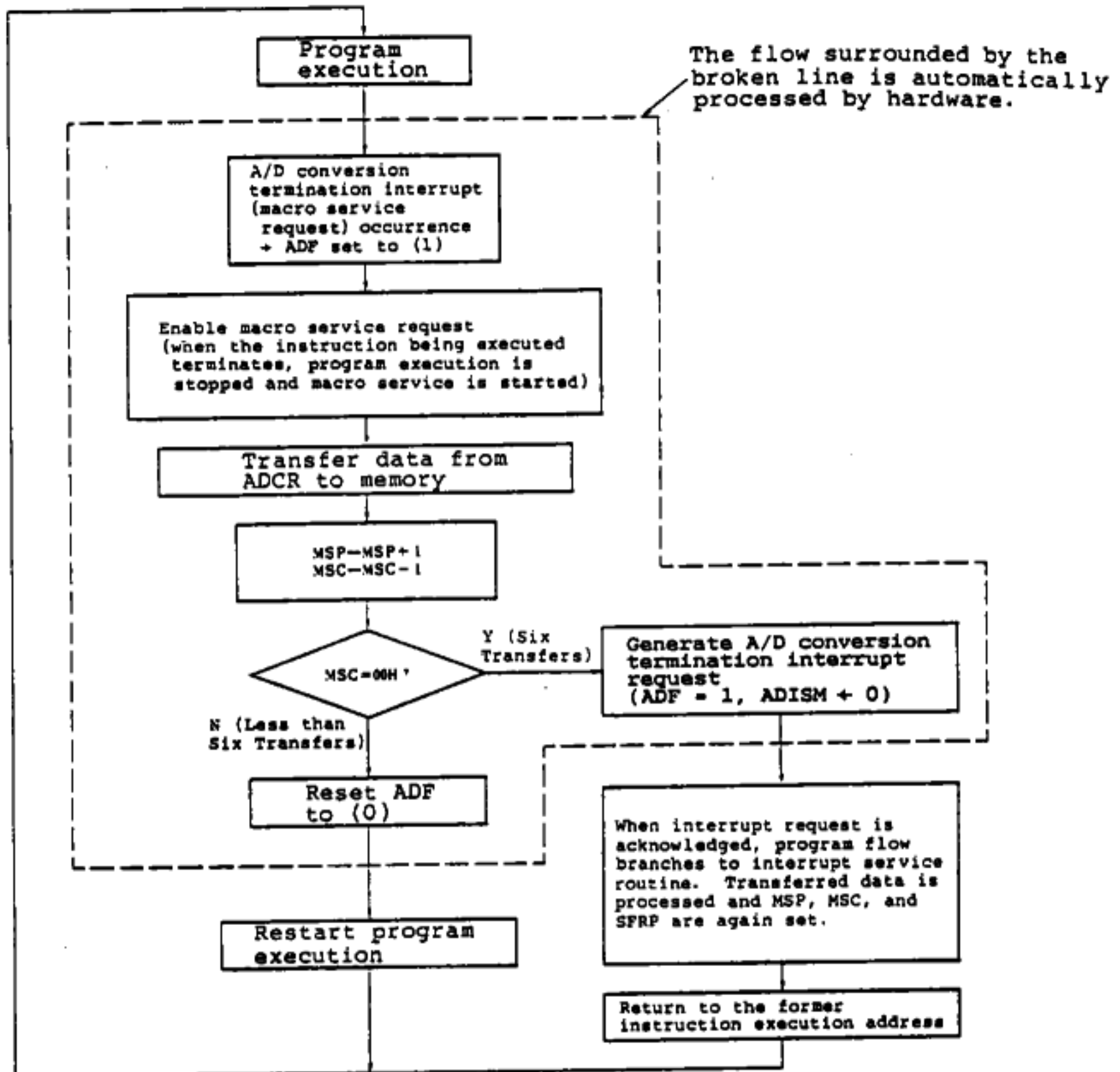


Figure 5-8 Macro Service Operation Example Flowchart



Remarks: ADF: A/D conversion interrupt request flag

### 5.2.2 MACRO SERVICE FUNCTION FEATURES

Unlike normal interrupt processing, macro service function processing is automatically performed without starting an interrupt service program. A sequence of steps such as branch to the interrupt service routine by rewriting the program counter (PC) and register save are not performed. Thus, the CPU service time can be improved and the number of program steps can be reduced. During the macro service execution, the general purpose registers and instruction queue in the CPU retain the state. The macro service function is useful for simple data transfer by interrupt request generation, such as A/D conversion result store and serial transfer data setting.

Since a normal interrupt request is generated automatically after data is transferred as many times as specified, the transferred data can be processed by the interrupt service program in batch.

### 5.2.3 MACRO SERVICE FUNCTION OPERATION

The interrupt request generated by setting the  $xxISM_n$  bit of the interrupt request register provided for each interrupt request source to (1) is handled as a macro service request.

When a macro service request occurs, the CPU temporarily stops program execution and 1-byte or 2-byte data is transferred automatically between memory and special function register (SFR). When the data transfer terminates, the interrupt request flag is reset to (0) and again the CPU returns to program execution. (See Figure 5-7.) Thus, user program execution is temporarily stopped during data transfer processing started by macro service request.

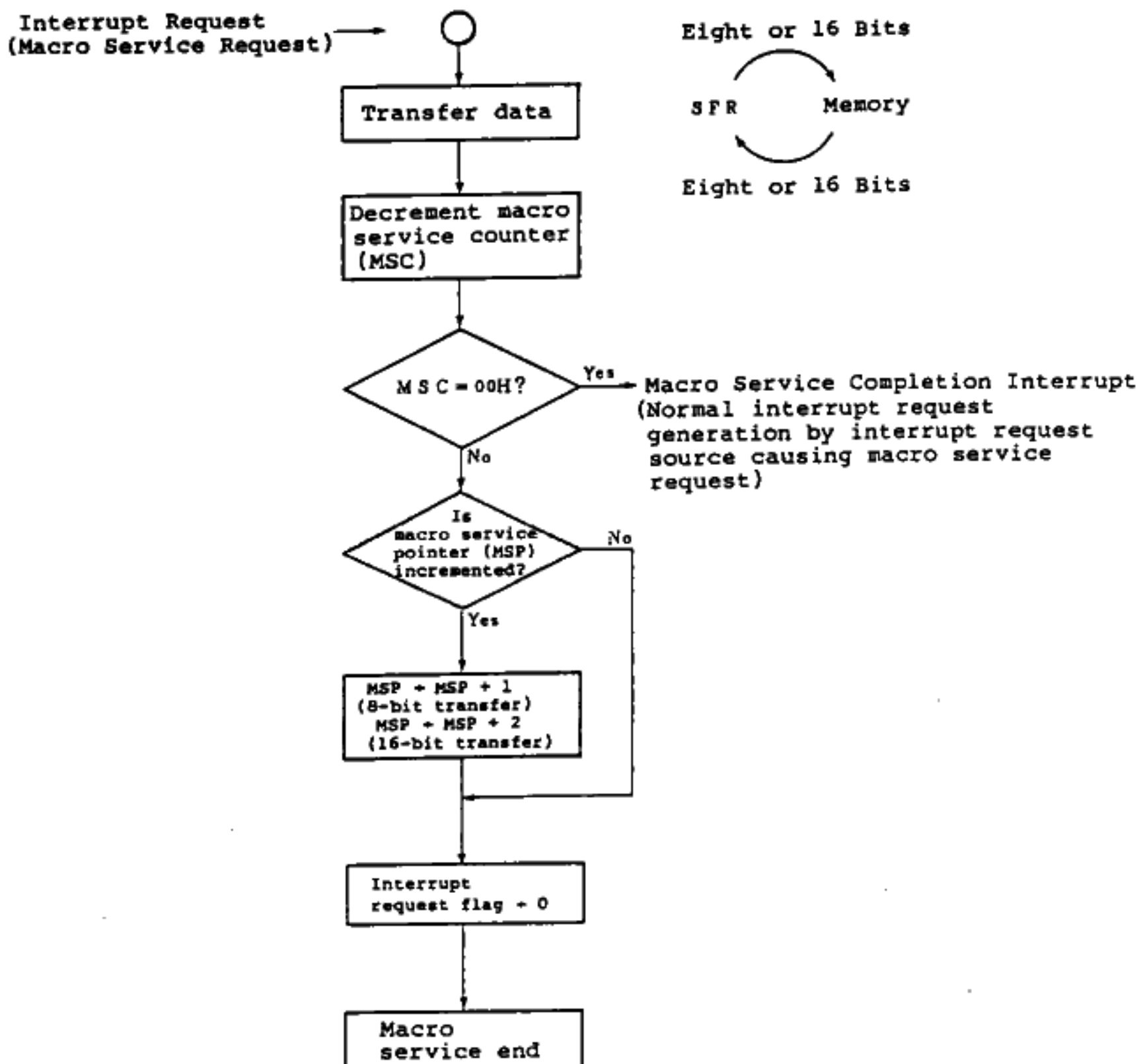
Whenever an interrupt request occurs, the destination/source memory address is incremented and SFR data can be stored in contiguous addresses of memory in sequence or data stored in memory can be set in SFR in sequence. With the destination/source memory address fixed, data at the same address can be set in SFR repeatedly or SFR data can be overwritten into the same address to make the most recent data only valid.

Source and destination SFR and memory addressing and transfer count specification are made on macro service channel mapped in internal RAM. (See 5.2.4.)

Macro service selection, channel transfer direction specification, and transfer mode specification are made by using the macro service control register. (See Figure 5-12.)

Whenever data is transferred, the macro service counter (MSC) is decremented by one. When the MSC is set to 00H, that is, data is transferred as many times as the count set in the MSC according to macro service request, the interrupt request flag is not reset to (0), the xxISMn bit is reset to (0), and a normal interrupt request of the same source as the macro service request is generated. (The vectored interrupt processing or context switching interrupt processing is performed by the xxCSEn bit.) In the interrupt service, process the transferred data and again set MSP, MSC, and SFRP as macro service completion interrupt.

Figure 5-9 Macro Service Operation Flow



NOTE: This flow is a general flow of macro service function operation and differs from actual operation flow.

The number of states required for macro service processing varies depending on the number of transfer data bits and whether or not the MSP is incremented, as listed in Table 5-4. If the MPS points to external memory area, the number of required states increases to the value enclosed in parentheses in Table 5-4.

If the SFRP (SFR pointer) points to the timer unit, count unit counter, or external access area, the number of required state furthermore increases as listed in Table 5-5.

Table 5-4 Number of Basic States Required for Macro Service Operation

	8-Bit Transfer	16-Bit Transfer
MSP is incremented after data is transferred	12 states (14 + m)	16 states (20 + 2m)
MSP is retained after data is transferred	11 states (13 + m)	15 states (19 + 2m)

Remarks: The value enclosed in parentheses is the number of required states when external memory is accessed as pointed to by MSP. m is the number of wait states inserted as specified in the MM register.

Table 5-5 Number of Added States Required for Macro Service Operation

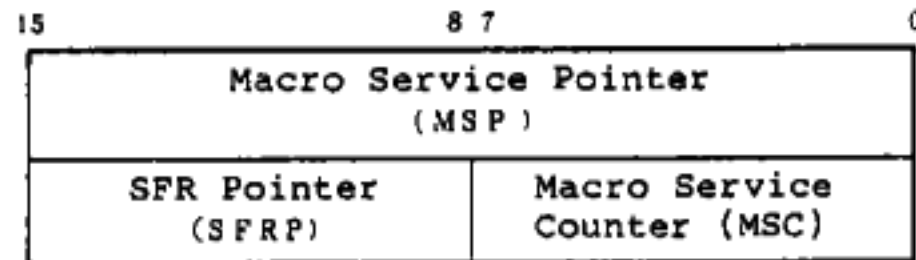
Accessed SFR (Special Function Register)	Number of Added States
Timer unit (TM0, TM1, MD0, MD1)	0 to 5*
Counter unit (UDC0, UDC1, CR00, CR01, CR10, CR11)	0 to 2*
External access area (FFB0H to FFBFH)	1 + m
SFR other than the above	0

\*: The number varies depending on the state when the counter is accessed; it is not defined.

#### 5.2.4 MACRO SERVICE CHANNELS

Each macro service channel consists of pointers to control macro service transfer processing and an 8-bit counter. Eight macro service channels are provided in the internal RAM area of FEE0H to FEFFH (also used for register banks 0 and 1).

Figure 5-10 Macro Service Channel Configuration



- . Macro service pointer (MSP) ..... 16 bits

The macro service pointer (MSP) points to the destination or source memory address. After data is transferred, the macro service pointer is incremented by one or two according to the number of bytes of the transferred data or retains the same value, as specified in the macro service control register (see Figure 5-12).

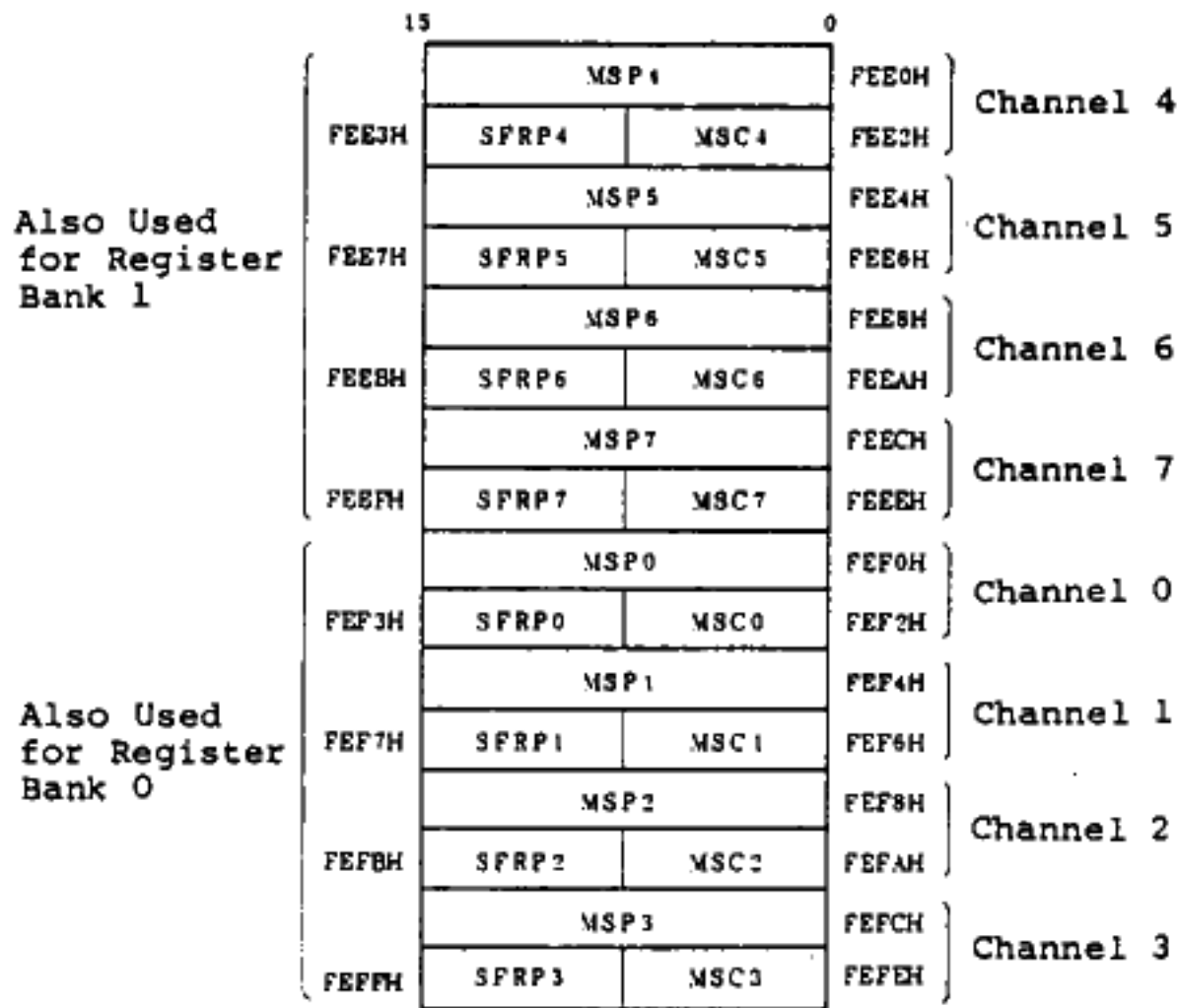
- . Macro service counter (MSC) ..... 8 bits

The macro service counter (MSC) is used to control the transfer count. Whenever data is transferred, the counter is decremented by one. When 01H is set in the counter, data is transferred once; when 00H is set, data is transferred 256 times.

- . SFR pointer (SFRP) ..... 8 bits

The SFR pointer (SFRP) points to the low-order 8-bit address of the destination or source SFR, but cannot point to any interrupt request control register or macro service control register.

Figure 5-11 Macro Service Channel Mapping



### 5.2.5 MACRO SERVICE CONTROL REGISTER

The macro service control register is used to control the macro service transfer mode and macro service channel selection. The register is provided for each maskable interrupt request source causing a macro service request. Figure 5-12 shows the macro service control register format.

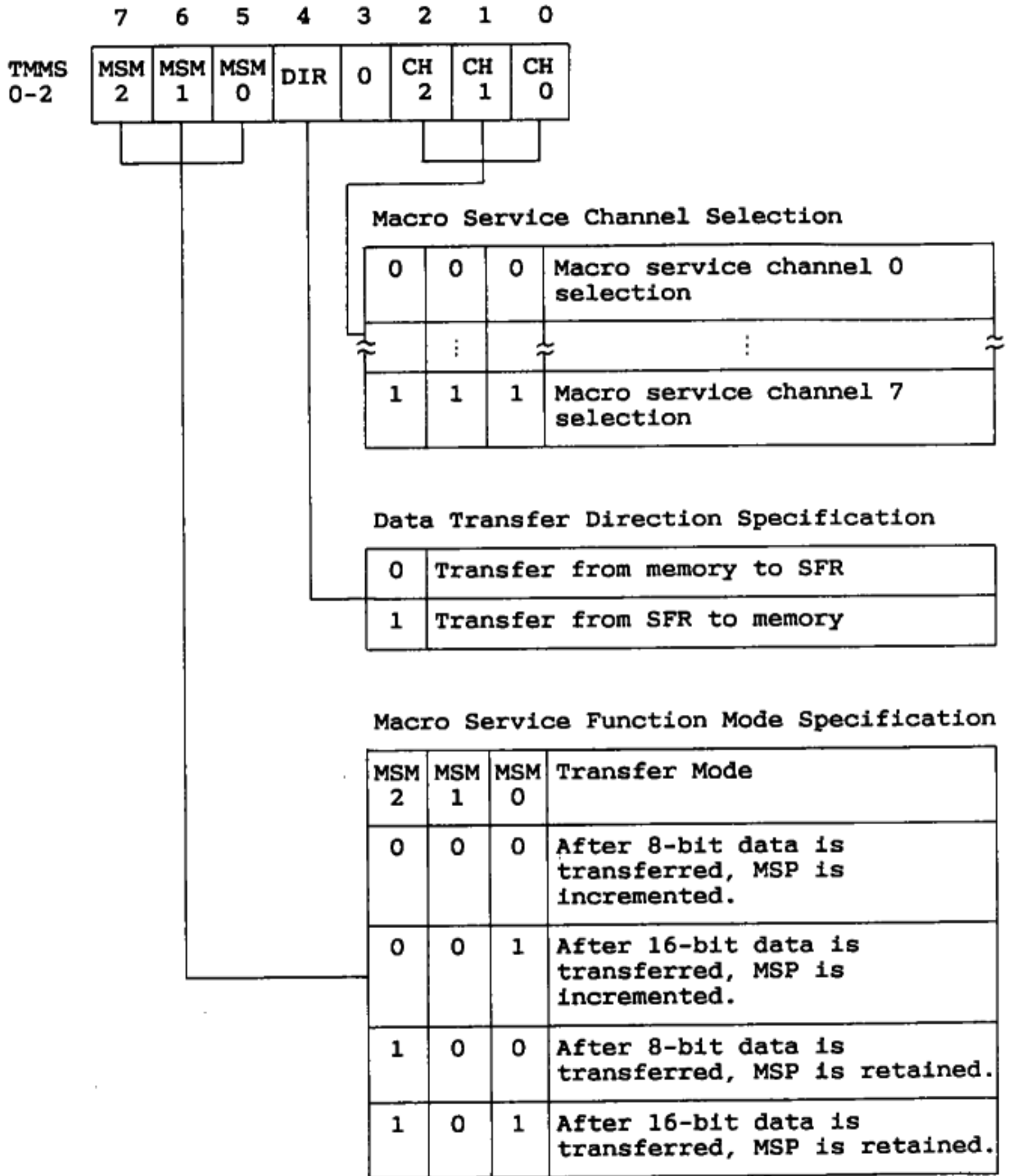
The macro service control register CH0 to CH2 bits are a bit field to select one of eight macro service channels on internal RAM.

The DIR bit is used to specify the data transfer direction. When the DIR bit is reset to (0), data is transferred from memory to a given special function register (SFR); when the bit is set to (1), data is transferred from SFR to memory.

The MSM0 to MSM2 bits are a bit field to specify the macro service transfer mode; 8-bit or 16-bit data transfer is specified and whether the macro service pointer is incremented or retained after data is transferred is specified.

When  $\overline{\text{RESET}}$  is input, the macro service control register becomes undefined.

Figure 5-12 Macro Service Control Register Format





**Remarks:**

	<b>Address</b>	<b>After Reset</b>
TMMS0 :	FFCFH	Undefined
TMMS1 :	FFD1H	Undefined
TMMS2 :	FFD3H	Undefined

### 5.3 CONTEXT SWITCHING FUNCTION

When an interrupt request occurs or the BRKCS instruction is executed, the context switching function selects a given register bank by hardware and stacks the current PC and PSW contents in the register bank at the same time as branch to the vector address prestored in the register bank.

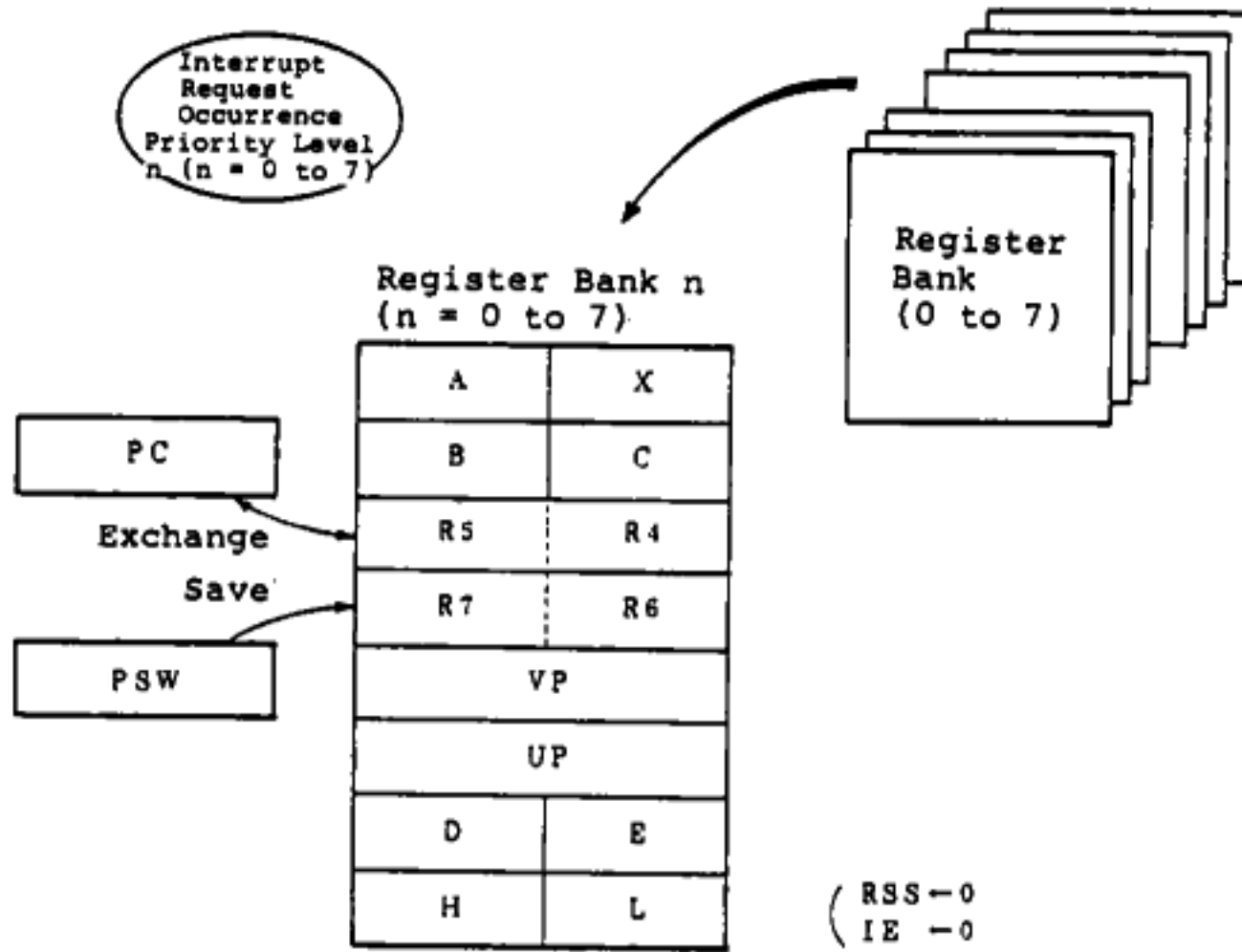
The context switching function can be specified only for maskable interrupts.

#### 5.3.1 CONTEXT SWITCHING FUNCTION WHEN INTERRUPT REQUEST OCCURS

Context switching function start is enabled by setting the xxCSEn bit of the interrupt request control register provided for each interrupt request source to (1). When an unmasked interrupt request for which the context switching function is enabled occurs in the EI state, the register bank corresponding to the priority level of the group containing the interrupt request is selected. The vector address prestored in the selected register bank is transferred to the program counter (PC). At the same time, the current PC and program status word (PSW) contents are saved in the register bank and control branches to the interrupt service routine.

11 states are required until branch to the interrupt service routine after an interrupt request for which the context switching function is enabled is enabled.

Figure 5-13 Context Switching Operation when Interrupt Request Occurs

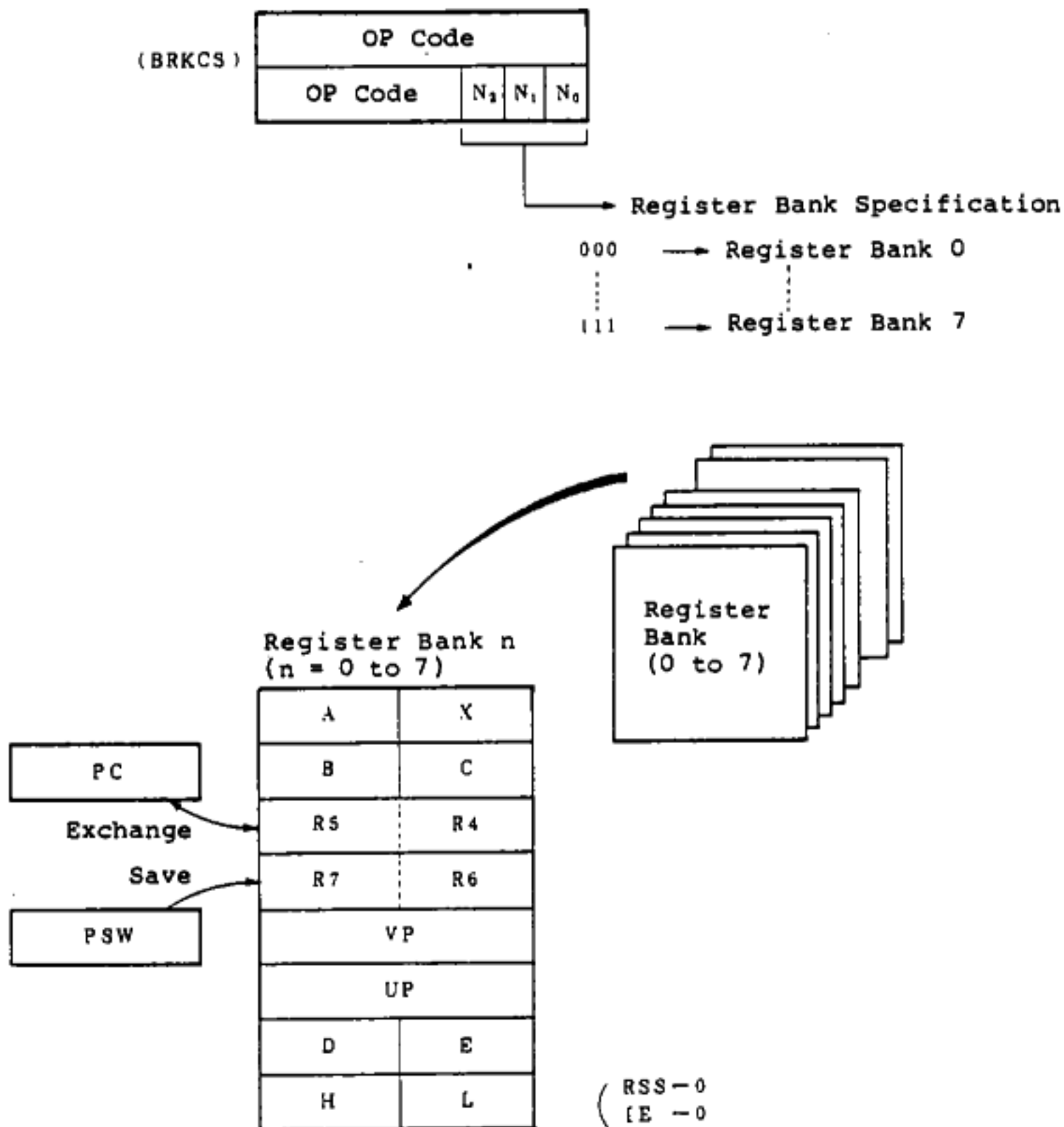


### 5.3.2 CONTEXT SWITCHING FUNCTION WHEN BRKCS INSTRUCTION IS EXECUTED

The context switching function can be started by executing the BRKCS instruction.

Specify the register bank selected by the context switching function in the BRKCS instruction operand RBn. When the BRKCS instruction is executed, the register bank indicated by the 3-bit immediate data (N<sub>2</sub> to N<sub>0</sub>) in the operation code is selected and control branches to the vector address prestored in the register bank. At the same time, the current PC and PSW contents are saved in the register bank.

Figure 5-14 Context Switching Operation when BRKCS Instruction is Executed





## CHAPTER 6. STANDBY FUNCTION

The uPD78312A provides three operation clock control modes for the standby function.

- o Clock varying mode: The uPD78312A can operate in the wide supply voltage range by changing the internal system clock (CLK) dividing ratio for low speed operation.
- o HALT mode : The CPU operation clock is stopped. The HALT mode and the normal operating mode can be used in combination for intermittent operation to reduce the total power consumption of the system.
- o STOP mode : The oscillator is stopped.  
Data can be retained by consuming low power.

### 6.1 STANDBY MODE SETTING AND OPERATION STATE

The standby mode is selected by using the standby control register (STBC). Three states are required until the standby state is entered after the standby mode STOP or HALT is selected by executing an STBC register write instruction.

#### 6.1.1 STANDBY CONTROL REGISTER (STBC)

The standby control register (STBC) is used to control the standby function mode. Figure 6-1 shows the STBC register format.

Like the watchdog timer mode register (WDM), the STBC register can be written only by executing a special instruction to prevent carelessly entering the standby mode because of program upset, etc.

The HLT bit is used to select the HALT mode. In the normal operation state, the HLT bit is reset to 0. When the HLT bit is set to 1, the HALT mode is selected.

The STP bit is used to select the STOP mode. In the normal operation state, the STP bit is reset to 0. When the STP bit is set to 1, the STOP mode is selected.

The CK0 and CK1 bits are a bit field to specify the internal system clock dividing ratio. Oscillator output is divided by the value specified in the CK0 and CK1 bits, then supplied as internal system clock (CLK).

The SBF bit is a standby flag that can be used for return decision from the STOP mode. It cannot be reset to 0 by software. If the  $\overline{\text{RESET}}$  signal is input, the standby flag is not affected.

When  $\overline{\text{RESET}}$  is input, the STBC register is set to 0010 x 000B, selecting the low speed operating mode.

Figure 6-1 Standby Control Register (STBC) Format

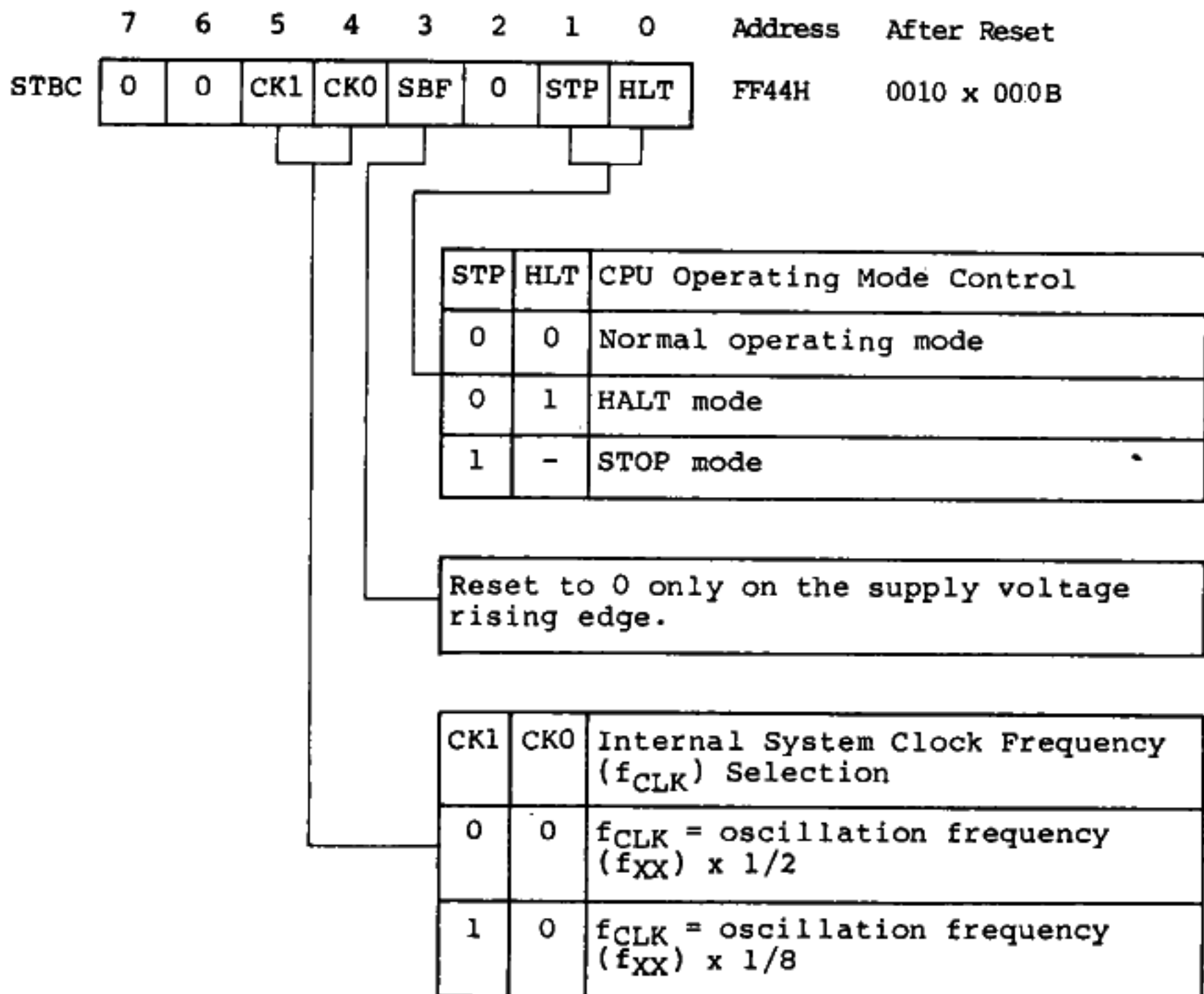
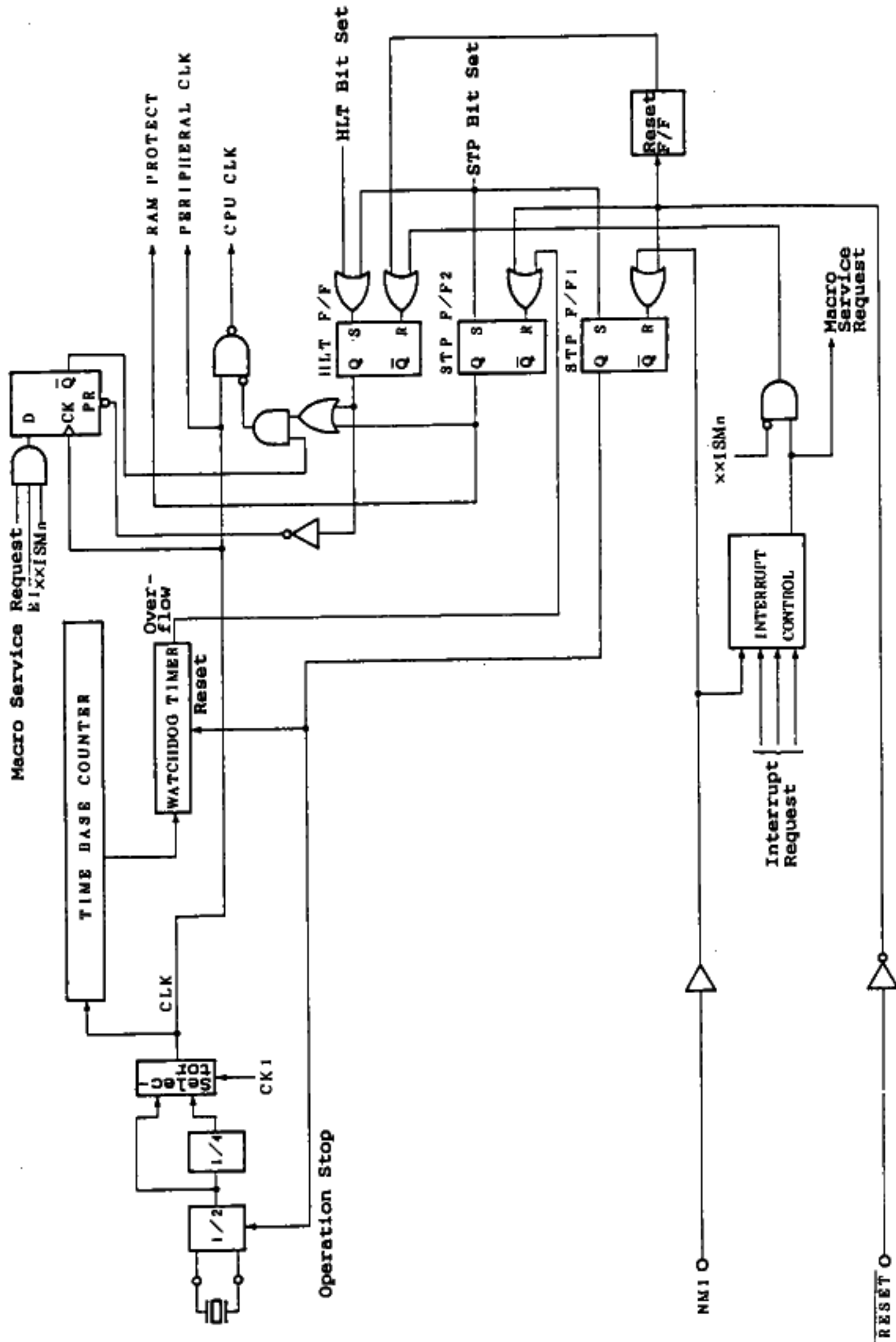


Figure 6-2 Standby Function Block Diagram



Remarks: CLK: Internal System Clock



### 6.1.2 USE OF SBF BIT

The SBF bit is reset (0) automatically when the uPD78312A is powered on (when the  $V_{DD}$  power supply rises from 0 V). It cannot be reset by software.

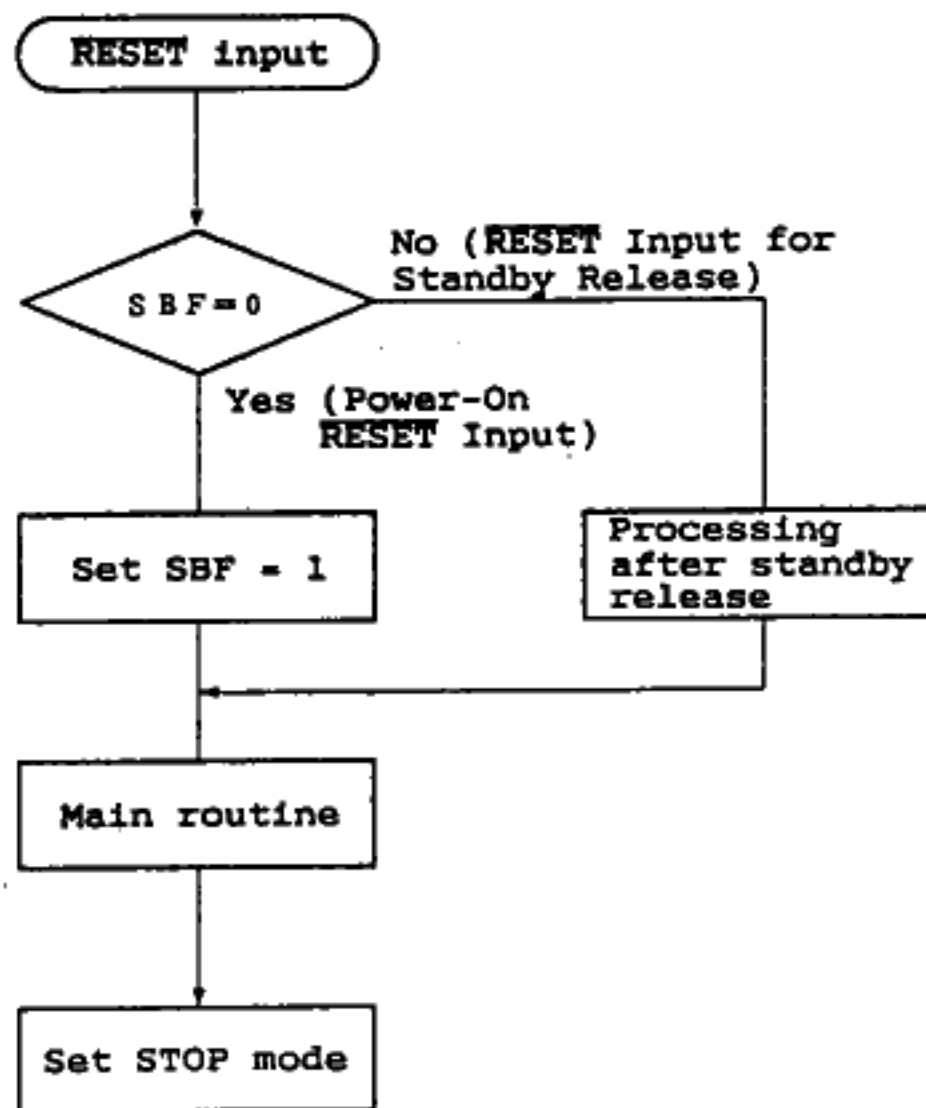
The SBF bit can only be set (1) by software. This bit is not affected by  $\overline{\text{RESET}}$  input (its contents are retained).

SBF	0	Rise of $V_{DD}$ power supply from 0 V
	1	"1" written by MOV STBC, #byte instruction

The above functions enable identification of power-on or standby release  $\overline{\text{RESET}}$  input by testing the SBF bit by software after  $\overline{\text{RESET}}$  input. The SBF bit processing routine is shown in Figure 6-3.

**NOTE:** The SBF bit may not be cleared if the supply voltage rise time after powering on exceeds the range given in the specifications. When the SBF bit is used, therefore, the supply voltage rise time must be kept within this range (see Chapter 11 "Specifications").

Figure 6-3 SBF Bit Processing Routine



### 6.1.3 CLOCK VARYING MODE

When the clock varying mode is selected, the uPD78312A can be operated at the low speed by furthermore dividing the internal system clock by four.

In the clock varying mode, the internal system clock frequency ( $f_{CLK}$ ) can be set to 1/2 or 1/8 times the oscillation frequency by setting standby control register (STBC) bits 5 and 4 (CK1 and CK0).

### 6.1.4 HALT MODE

In the HALT mode, the CPU operation clock is stopped.

The total power consumption of the system can be reduced by selecting the HALT mode for the CPU idle time. The HALT mode is selected by setting standby control register (STBC) bit 0 (HLT) to 1.

In the HALT mode, the CPU clock stops and program execution is stopped, but all the register and internal RAM contents just before the HALT mode is entered are retained. Table 6-1 lists the hardware state.

#### 6.1.5 STOP MODE

In the STOP mode, the oscillator is stopped.

When the entire application system stops, the STOP mode enables very low power consumption. The STOP mode is selected by setting standby control register (STBC) bit 1 (STP) to 1. In the STOP mode, all clocks stop. Although program execution is stopped, all the register and internal RAM contents just before the STOP mode is entered are retained. Table 6-1 lists the hardware state.

NOTE: When the STOP mode is set, the X1 pin is internally short-circuited to  $V_{SS}$  (ground potential) to suppress clock generator leakage. Thus, do not use the STOP mode in a system which uses external clock.

Table 6-1 Operation State during HALT/STOP Mode

Item		HALT Mode	STOP Mode
Oscillator		Operation	Stop
Internal system clock			
CPU clock		Stop	
Pulse input/output unit		Operation	
Time base counter			
Watchdog timer			
Serial interface			
Interrupt request control			
A/D converter			
I/O lines		Retention	Retention
Bus lines	A8 to A15	Retention	Retention
	AD0 to AD7	Retention	Retention
RD, WR output		High	High
ALE output		Low	Low
RFSH output		Stop*	Stop*
Internal CPU status and internal RAM contents		All retained	All retained

\*: Set the power down refresh mode before setting the standby mode (see 8.3.2).

## 6.2 STANDBY MODE RELEASE

### 6.2.1 HALT MODE RELEASE

When a nonmaskable interrupt request, unmasked maskable interrupt request, macro service request occurs, or  $\overline{\text{RESET}}$  is input, the HALT mode is released.

#### (1) Release when interrupt request occurs

##### (i) When HALT mode is set in interrupt service routine

When an unmasked maskable interrupt assigned the higher priority level than the current interrupt in service or a nonmaskable interrupt request occurs, the HALT mode is released.

##### (ii) Other than (i)

When a nonmaskable interrupt request or an unmasked maskable interrupt request occurs regardless of the priority level, the HALT mode is released.

#### (2) Release when $\overline{\text{RESET}}$ is input

This is the same as normal reset operation except that the internal RAM contents before the HALT mode is set are retained.

Figure 6-4 HALT Mode Release when Interrupt Request Occurs

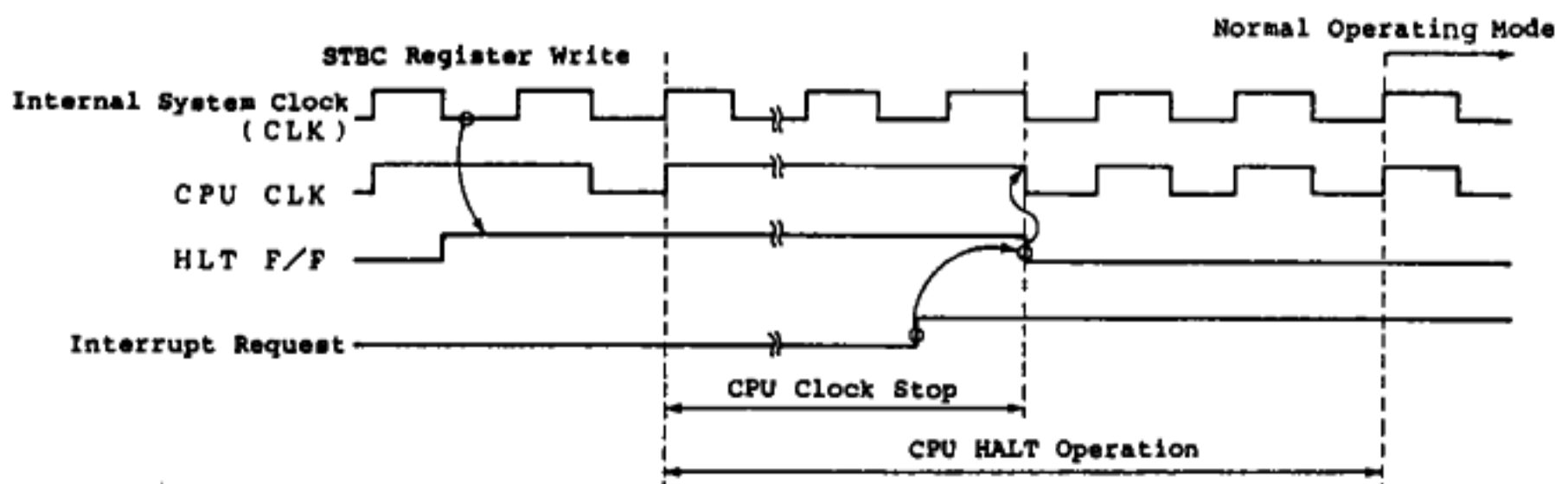
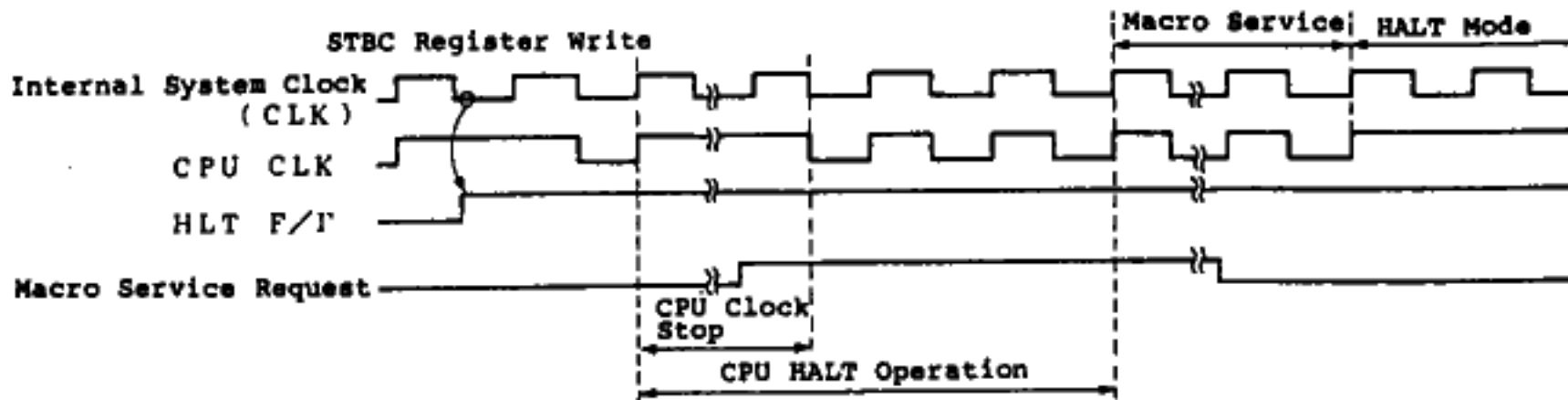


Figure 6-5 Macro Service Start during HALT Mode



### 6.2.2 STOP MODE RELEASE

When an NMI pin input interrupt request occurs or  $\overline{\text{RESET}}$  is input, the STOP mode is released.

- (1) Release when a valid edge is input to NMI pin

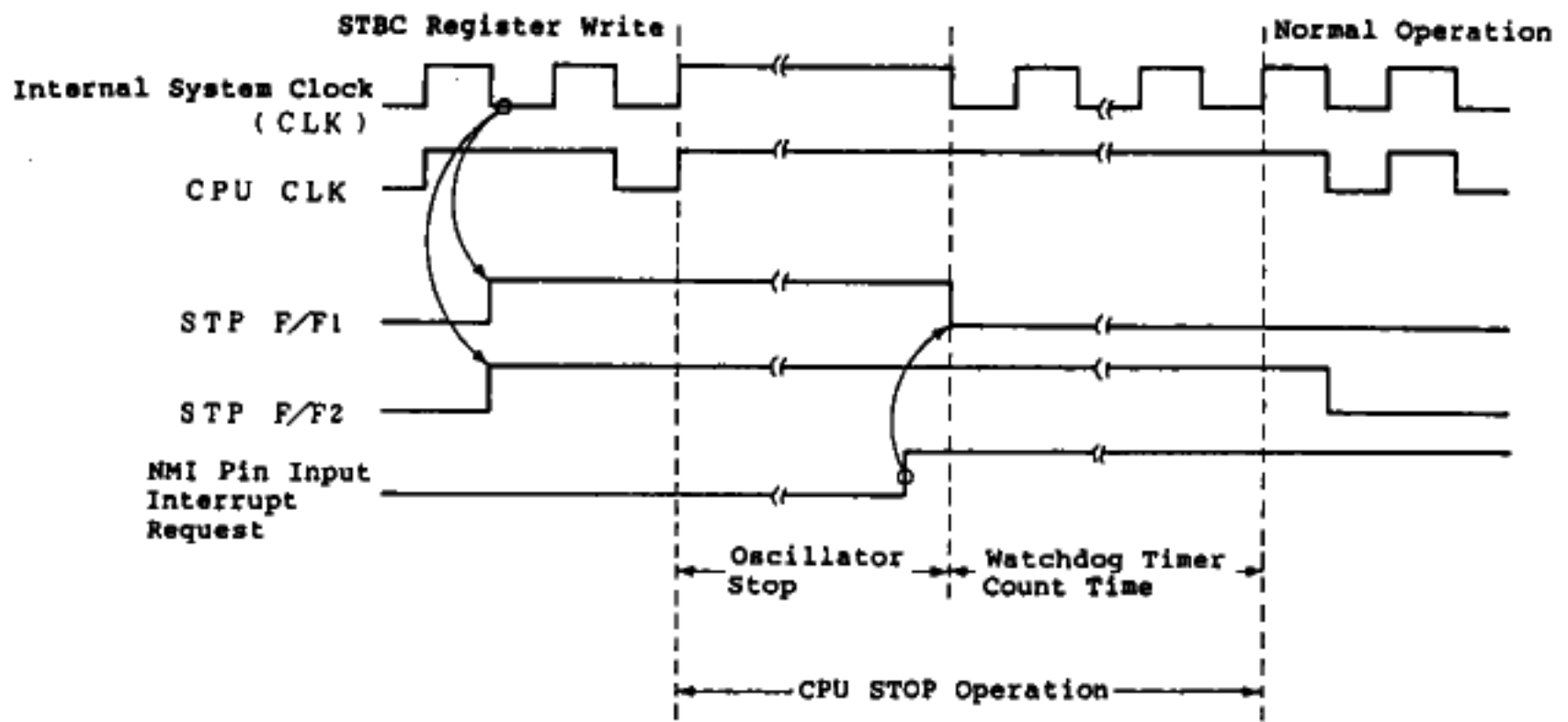
When a valid edge is input to the NMI pin, oscillator oscillation is restarted. The time base counter and watchdog timer start operation. The watchdog timer counts time base counter tap output. After the watchdog timer counts the time specified in the watchdog timer mode register (WDM), that is, when the watchdog timer overflows, CPU clock supply is started.

The same state as the HALT mode is once entered, then the same operation as the HALT mode release operation is performed.

- (2) Release when  $\overline{\text{RESET}}$  is input

This is the same as normal reset operation except that the internal RAM contents before the STOP mode is set are retained.

**Figure 6-6 STOP Mode Release when a Valid Edge is Input to NMI Pin**



### 6.3 OPERATION AFTER STANDBY MODE IS RELEASED

- (1) If the STOP or HALT mode is released when  $\overline{\text{RESET}}$  is input, normal reset operation is performed except that the internal RAM contents before the standby mode is set are retained.
- (2) If the STOP mode is released when a valid edge is input to the NMI pin (NMI pin input interrupt request), vectored interrupt is executed after the STOP mode is released.

The operation after the HALT mode is released when an interrupt request occurs varies depending on whether interrupts are enabled (EI) or disabled (DI) when the CPU restarts instruction execution, as listed in Table 6-2.

Table 6-2 Operation after HALT Mode is Released when Interrupt Request Occurs

Release Source	EI State	DI State
Nonmaskable interrupt request	After standby mode is released, control branches to vector address.	After standby mode is released, control branches to vector address.
Maskable interrupt request	After standby mode is released, control branches to vector address.	After standby mode is released, the next instruction is executed.
Macro service request	When macro service starts and the macro service counter reaches 0H, control branches to vector address. If the macro service counter does not reach 0H, the HALT mode is entered again.	Macro service does not start. The next instruction is executed.

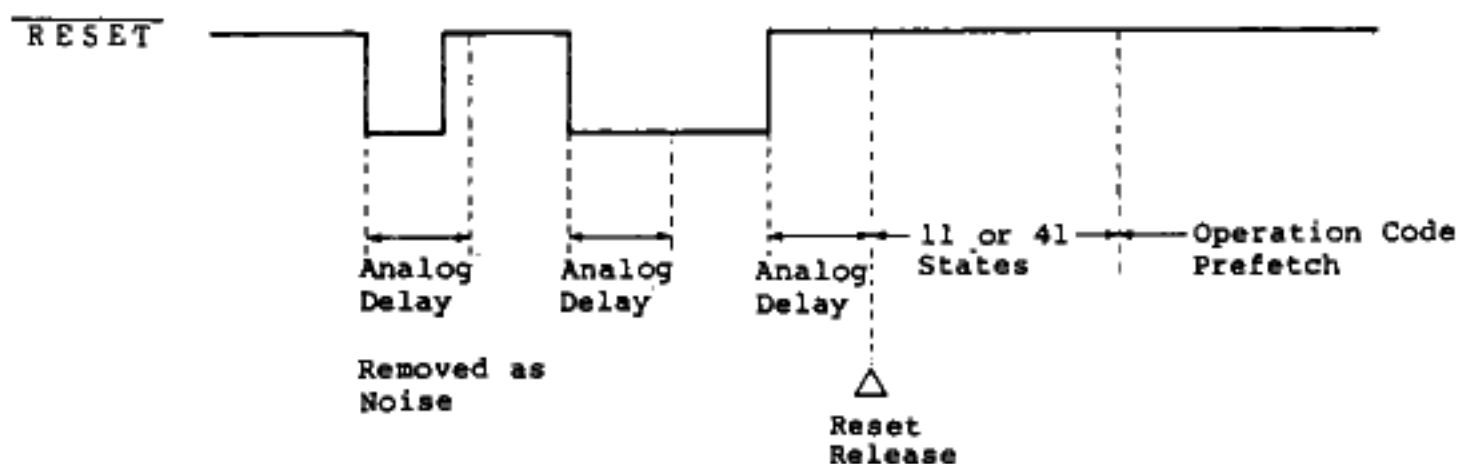


## CHAPTER 7. RESET FUNCTION

When a low pulse is input to the  $\overline{\text{RESET}}$  input pin, the system is reset and the hardware devices are placed in the state as listed in Table 7-1. When the low-to-high transition of  $\overline{\text{RESET}}$  input is made, the reset state is released and the contents of reset vector table (addresses 0000H and 0001H) are loaded into the program counter (PC) for branch. Program execution starts at the branch destination address. The  $\overline{\text{RESET}}$  input pin contains a noise removal circuit using analog delay. The number of states required until operation code prefetch is started after  $\overline{\text{RESET}}$  input rises and reset is released is 11 when the reset vector table is set in internal ROM or 41 when the table is set in external memory.

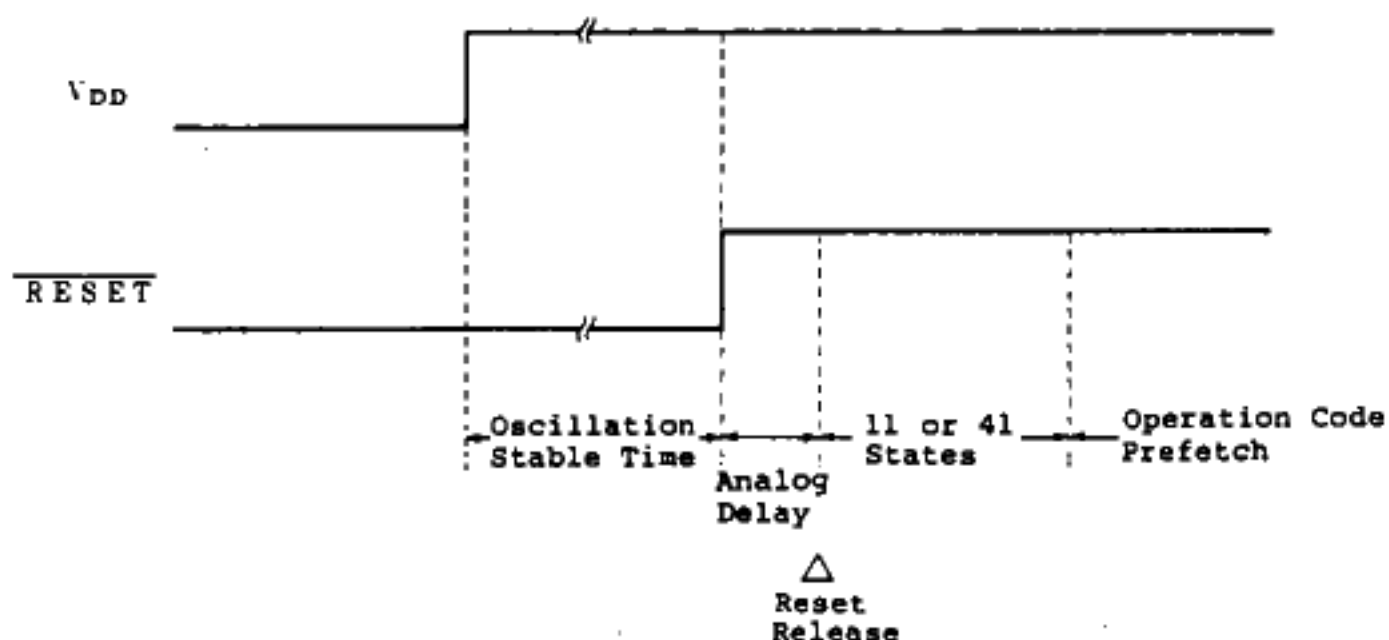
Initialize the register contents in a program as required.

Figure 7-1 Reset Signal Acknowledgement



On reset operation when the power is turned on, take the oscillation stable time of about 40 ms from power on to reset acknowledgement, as shown in Figure 7-2.

Figure 7-2 Reset when Power is Turned on



After reset operation,

- . MM register PW0, PW1 bits = 11
- . RFM register RFEN bit = 1
- . STBC register CK1 bit = 1

are set as listed in Table 7-1. Thus, the 3-wait cycle insertion mode, refresh pulse generation mode, and low speed operation mode ( $f_{CLK} = f_{XX} \times 1/8$ ) are selected. To operate the uPD78312A at the high speed, clear the bits to 0 shown above at the beginning of a program.

Table 7-1 Hardware State after Reset

Hardware		State after Reset
Program counter (PC)		00H
Stack pointer (SP)		Undefined
Program status word (PSW)		00H
CPU control word (CCW)		00H
Internal RAM	Data memory	Undefined*
	General register (R0 to R15)	
Ports	Port register (P0 to P5)	Undefined
	Mode registers (PM0 to PM3 and PM5)	FFH (input mode)
	Mode control registers (PMC2 and PMC3)	0FH
Count unit	Capture compare registers (CR00, CR01, CR10, and CR11)	Undefined
	Up/down count registers (UDC0 and UDC1)	Undefined

(to be continued)

\*: When the standby mode is released, the state before the standby mode is set is retained.

Table 7-1 Hardware State after Reset (cont'd)

Hardware		State after Reset
Count unit (cont'd)	Input mode register (CUIM)	00H
	UDC control registers (UDCC0 and UDCC1)	00H
	Capture compare register control register (CRC)	00H
Capture PWM unit	Capture registers (CPT0 and CPT1)	Undefined
	PWM registers (PWM0 and PWM1)	Undefined
	FRC control registers (FRCC)	00H
	Capture mode register (CPTM)	00H
	PWM mode register (PWMM)	00H
Real time output port	Control register (RTPC)	08H
	Port 0 L buffer register (POL)	Undefined
	Port 0 H buffer register (POH)	Undefined
Timer unit	Timer registers (TM0 and TM1)	Undefined
	Modulo/timer registers (MDO and MD1)	Undefined
	Timer control registers (TMC0 and TMC1)	00H
A/D converter	Mode register (ADM)	00H
	Conversion result register (ADCR)	Undefined

(to be continued)

Table 7-1 Hardware State after Reset (cont'd)

Hardware		State after Reset
Serial communication interface	Serial mode register (SCM)	00H
	Serial control register (SCC)	00H
	Baud rate generator setup value (BRG)	00H
	Receive buffer register (RxB)	Undefined
	Transmit buffer register (TxB)	Undefined
Time base counter		00H
Time base mode register (TBM)		00H
Standby control register (STBC)		2 x H *
Watchdog timer mode register (WDM)		00H
Memory expansion mode register (MM)		30H
Refresh mode register (RFM)		10H
Interrupt request	External interrupt mode register (INTM)	00H
	In-service priority register (ISPR)	00H
	Interrupt request control register	47H
	Macro service control register	Undefined

\*: The STBC bits are not affected by  $\overline{\text{RESET}}$  input. The low-order four bits are set to 0 or 8.

## CHAPTER 8. LOCAL BUS INTERFACE FUNCTION

The local bus interface function is provided to connect external memory (ROM, RAM) and I/O in addition to internal memory.

### 8.1 uPD78312A, uPD78P312A EXTERNAL DEVICE EXPANSION FUNCTION

In addition to internal ROM and RAM, external device (data memory, program memory, or peripheral device) can be added to the uPD78312A, uPD78P312A external 56-Kbyte area (addresses 2000H to FDFH). To add external device, port 4 (P47 to P40) is used as multiplexed address/data bus (AD7 to AD0) and port 5 (P57 to P50) is used as address bus (A7 to A0) by setting the memory expansion mode register (MM), and the external device is accessed by using the  $\overline{RD}$ ,  $\overline{WR}$ , and ALE signals.

The number of bits of the P57 to P50 pins which serve as the address bus can be changed according to the external memory size. External memory can be expanded by stages from 256 bytes to 56 Kbytes. The pins not used as the address bus can be used as general purpose input/output port pins (see Table 8-1).

Table 8-1 P57 to P50 Address Bus Selection

P57	P56	P55	P54	P53	P52	P51	P50	External Address Space
Port	Port	Port	Port	Port	Port	Port	Port	Within 256 bytes
Port	Port	Port	Port	All	A10	A9	A8	Within 4 Kbytes
Port	Port	A13	A12	A11	A10	A9	A8	Within 16 Kbytes
A15	A14	A13	A12	A11	A10	A9	A8	Within 56 Kbytes

When an external device reference instruction is executed in the 256-byte expansion mode, the high-order eight bits of the 16-bit external reference address are masked and address information in the range of 00H to FFH is output from the P47 to P40 (AD7 to AD0) pins.

Likewise, in the 4-Kbyte expansion mode, the high-order four bits of the 16-bit external reference address are masked and address information in the range of 000H to FFFH is output from the P53 to P50 (A11 to A8) and P47 to P40 pins.

Likewise, in the 16-Kbyte expansion mode, the high-order two bits of the 16-bit external reference address are masked and address information in the range of 0000H to 3FFFH is output from the P55 to P50 (A13 to A8) and P47 to P40 pins.

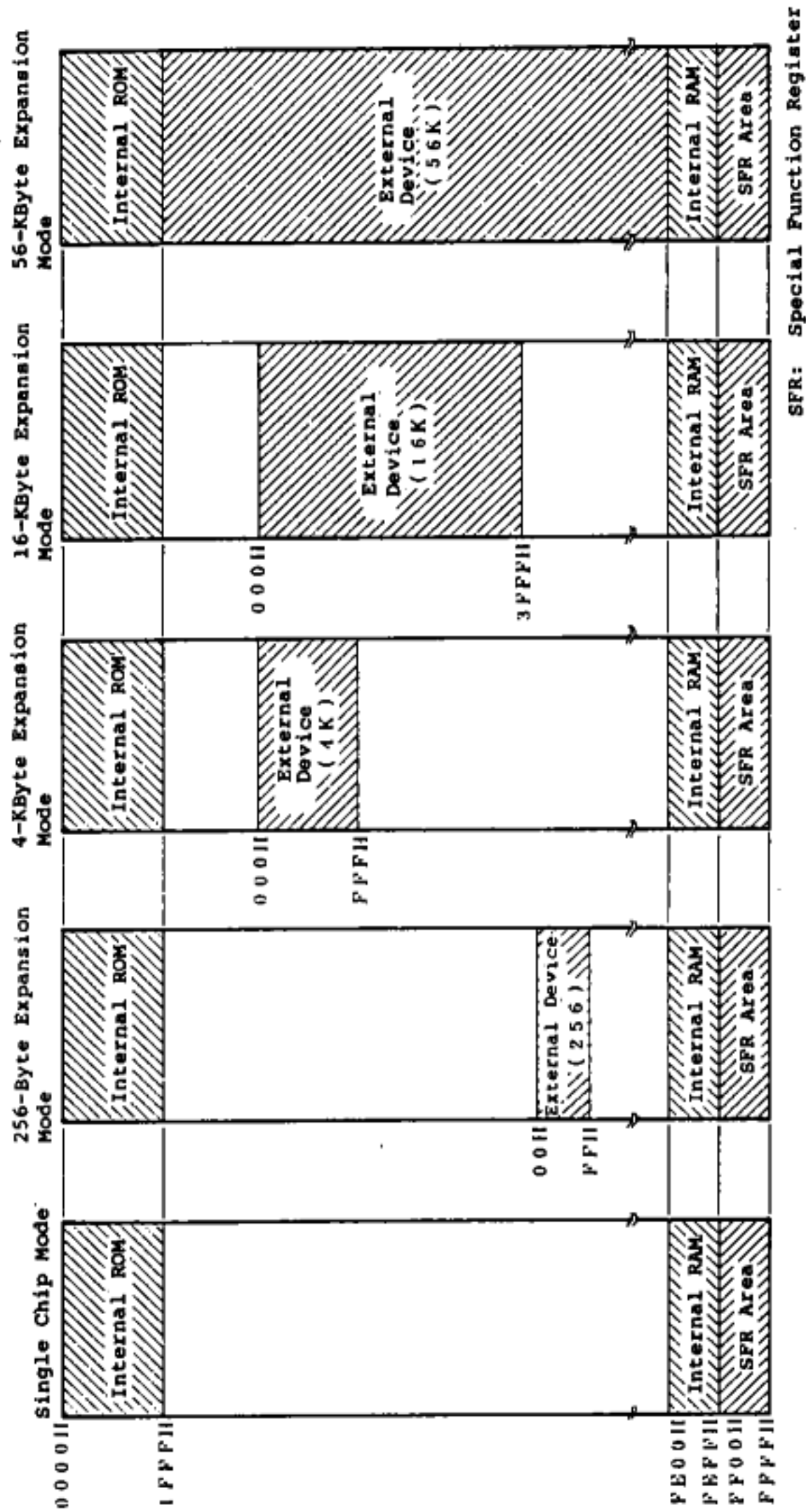
Thus, in the 256-byte, 4-Kbyte, or 16-Kbyte expansion mode, the high-order bits of the 16-bit address are masked, and external device can be located in any 256-byte, 4-Kbyte, or 16-Kbyte area within the external 56-Kbyte area. However, note that if in the 16-Kbyte expansion mode external ROM is connected to an expansion area and the external ROM area is located in addresses 2000H to 5FFFH following the internal ROM, the program counter (PC) contents differ from the actually output address from the P55 to P50 and P47 to P40 pins as follows:

PC Contents    P55 to P50, P47 to P40 Output Address

2000H	2000H
⋮	⋮
3FFFH	3FFFH
4000H	0000H
⋮	⋮
5FFFH	1FFFH

To use the external ROM addresses as consecutive addresses, locate external ROM area in addresses 4000H to 7FFFH. In this case, external ROM and internal ROM addresses are not consecutive, thus a branch instruction must be used to move a program between the internal and external ROM areas. This is also applied when external ROM area is located in addresses 8000H to BFFFH.

Figure 8-1 External Expansion Mode Selected by Setting Memory Expansion Mode Register (uPD78312A, uPD78P312A)



### 8.1.1 MEMORY EXPANSION MODE REGISTER (MM)

The memory expansion mode register (MM) is an 8-bit register to control the address bus, address/data bus, and control signals such as  $\overline{RD}$  and  $\overline{WR}$  when memory or I/O is expanded to the external.

Figure 8-2 shows the memory expansion mode register (MM) format.

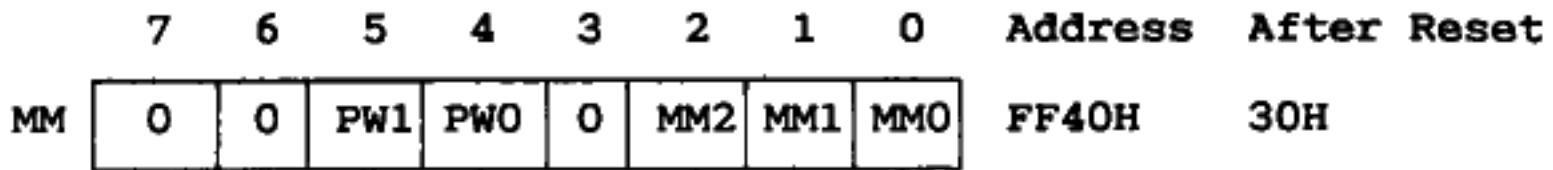
The MM2 to MM0 bits are effective when the  $\overline{EA}$  pin is high (1); the bits are used to specify P47 to P40 pin port or expansion mode and input or output and P57 to P50 pin address output.

The PW1 and PW0 bits are used to specify the number of waits inserted in external access cycle regardless of the  $\overline{EA}$  pin level.

When  $\overline{RESET}$  is input, the MM register is set to 30H; P47 to P40 are used as input port (output high impedance) and three waits are inserted in external access cycle.



Figure 8-2 MM Register Format



P47 to P40 and P57 to P50 Operation

0	0	0	Port mode	Single chip mode	P47 to P40: Input port P57 to P50: Port mode
0	0	1			P47 to P40: Output port P57 to P50: Port mode
0	1	1	Expansion mode	256 bytes	P47 to P40: Expansion mode P57 to P50: Port mode
1	0	0		4K bytes	P47 to P40 & P53 to P50: Expansion mode P57 to P54: Port mode
1	0	1		16K bytes	P47 to P40 & P55 to P50: Expansion mode P57 and P56: Port mode
1	1	1		56K bytes	P47 to P40 & P57 to P50: Expansion mode

Wait Insertion when External Access is Made

0	0	No wait cycle is inserted in external access cycle.
0	1	One wait cycle is inserted in external access cycle.
1	0	Two wait cycles are inserted in external access cycle.
1	1	Three wait cycles are inserted in external access cycle.

### 8.1.2 MEMORY EXPANSION EXAMPLE

Figure 8-3 shows an expansion example of 16K-byte memory when the 64K-byte PROM product uPD27C512\* is connected. Figure 8-4 shows the data set in the memory expansion mode register in the example.

\*: Maintenance product

Figure 8-3 Memory Expansion Example (for Reference)

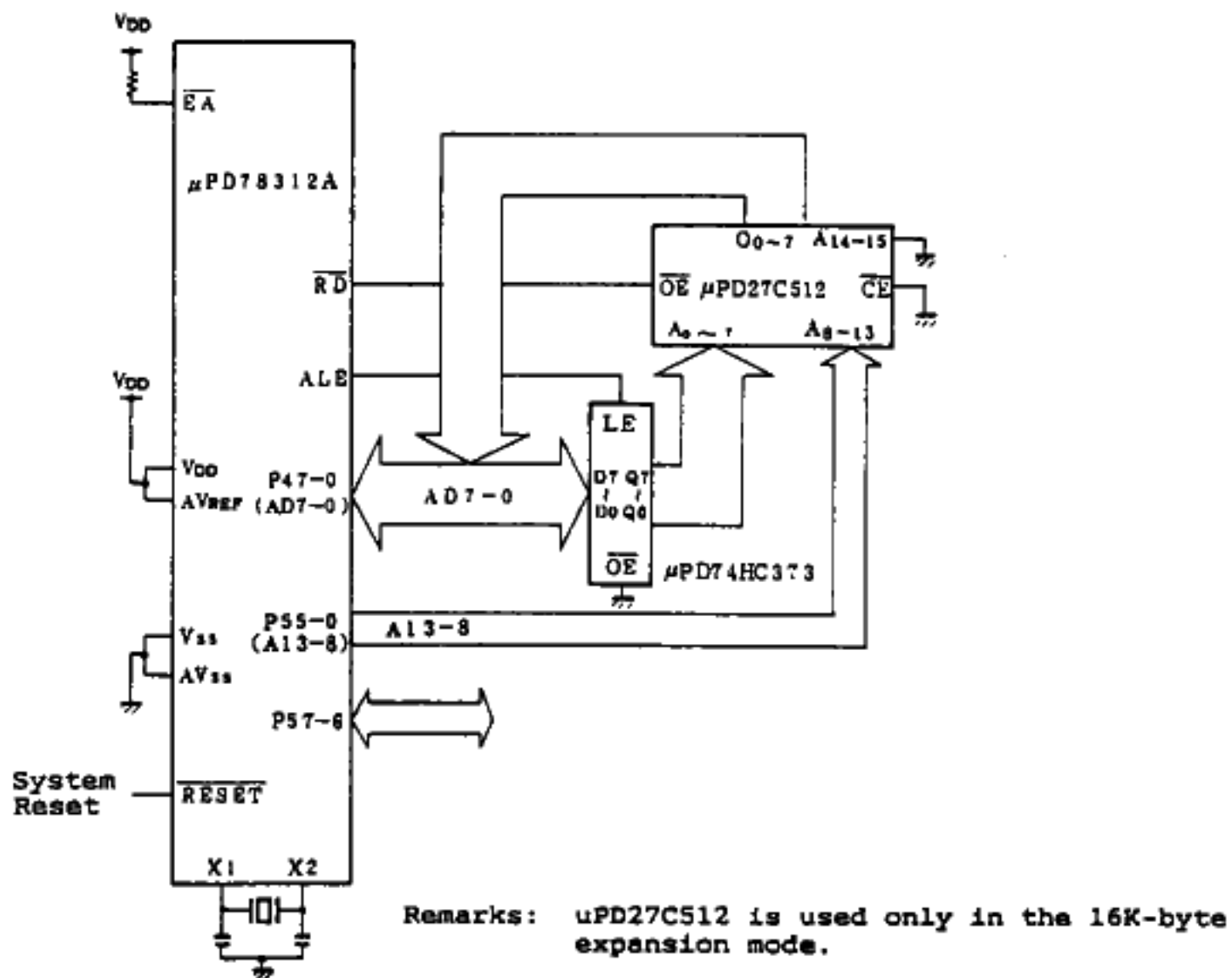
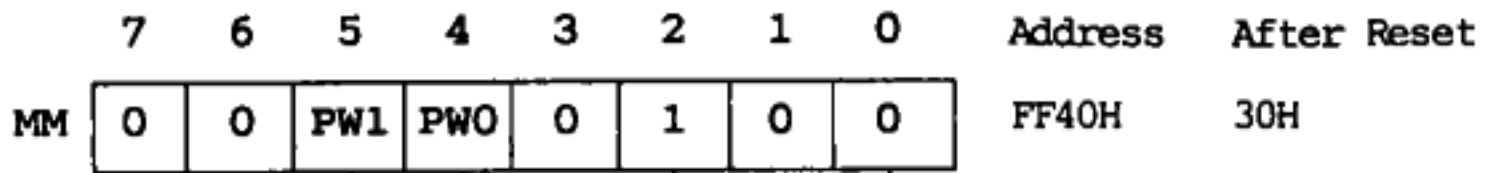


Figure 8-4 Memory Expansion Register Setting  
(in Memory Expansion Example)



[ P47 to P40 : Expansion Mode  
 P55 to P50 : Expansion Mode  
 P57 and P56: Port Mode

PW1	PW0	Wait Cycle Insertion when External Memory is Accessed
0	0	No wait cycle is inserted
0	1	One wait cycle is inserted
1	0	Two wait cycles are inserted
1	1	Three wait cycles are inserted

## 8.2 uPD78310A EXTERNAL DEVICE ACCESS

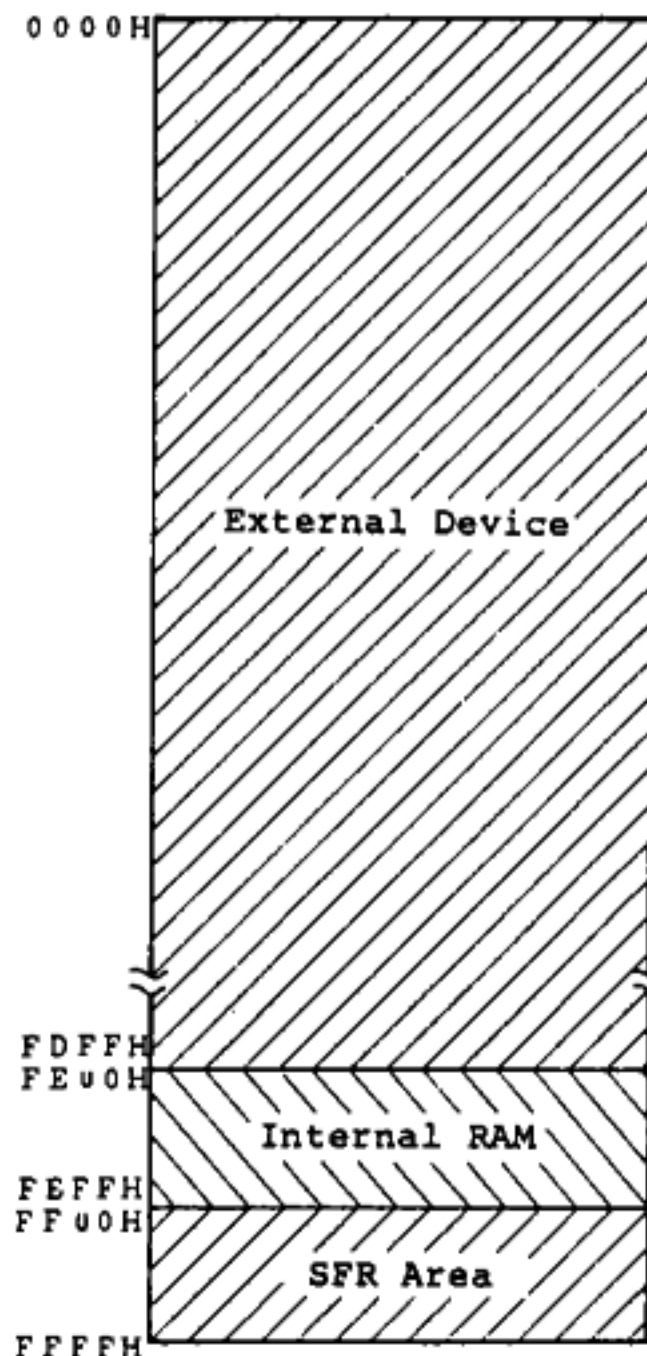
Since the uPD78310A does not contain ROM, external device (data memory, program memory, or peripheral device) can be located in the external 64-Kbyte area (0000H to FDFFH) in addition to internal RAM.

External device is accessed by using the P47 to P40 as multiplexed address/data bus (AD7 to AD0) and the P57 to P50 pins as the address bus (A15 to A8) pins and using the  $\overline{RD}$ ,  $\overline{WR}$ , and ALE signals. Fix the  $\overline{EA}$  pin low.

On the uPD78310A, specification in the memory expansion mode register (MM) MM2 to MM0 bits is ineffective and the P47 to P40 and P57 to P50 pins always function as the AD7 to AD0 and A15 to A8 pins.

The external expansion mode as with the uPD78312A cannot be selected.

Figure 8-5 uPD78310A Address Space



### 8.3 PSEUDO-STATIC RAM REFRESH FUNCTION

The uPD78312A contains the pseudo-static RAM refresh function to directly connect pseudo-static RAM equivalent to uPD42818\*. The refresh pulse output intervals are specified by setting the refresh mode register (RFM), and the external access cycle is changed to the refresh bus cycle corresponding to the uPD428128 bus cycle.

The uPD78312A contains the following function to support uPD428128 pulse refresh operation and power down self-refresh operation:

\*: Maintenance product

#### 8.3.1 PULSE REFRESH OPERATION

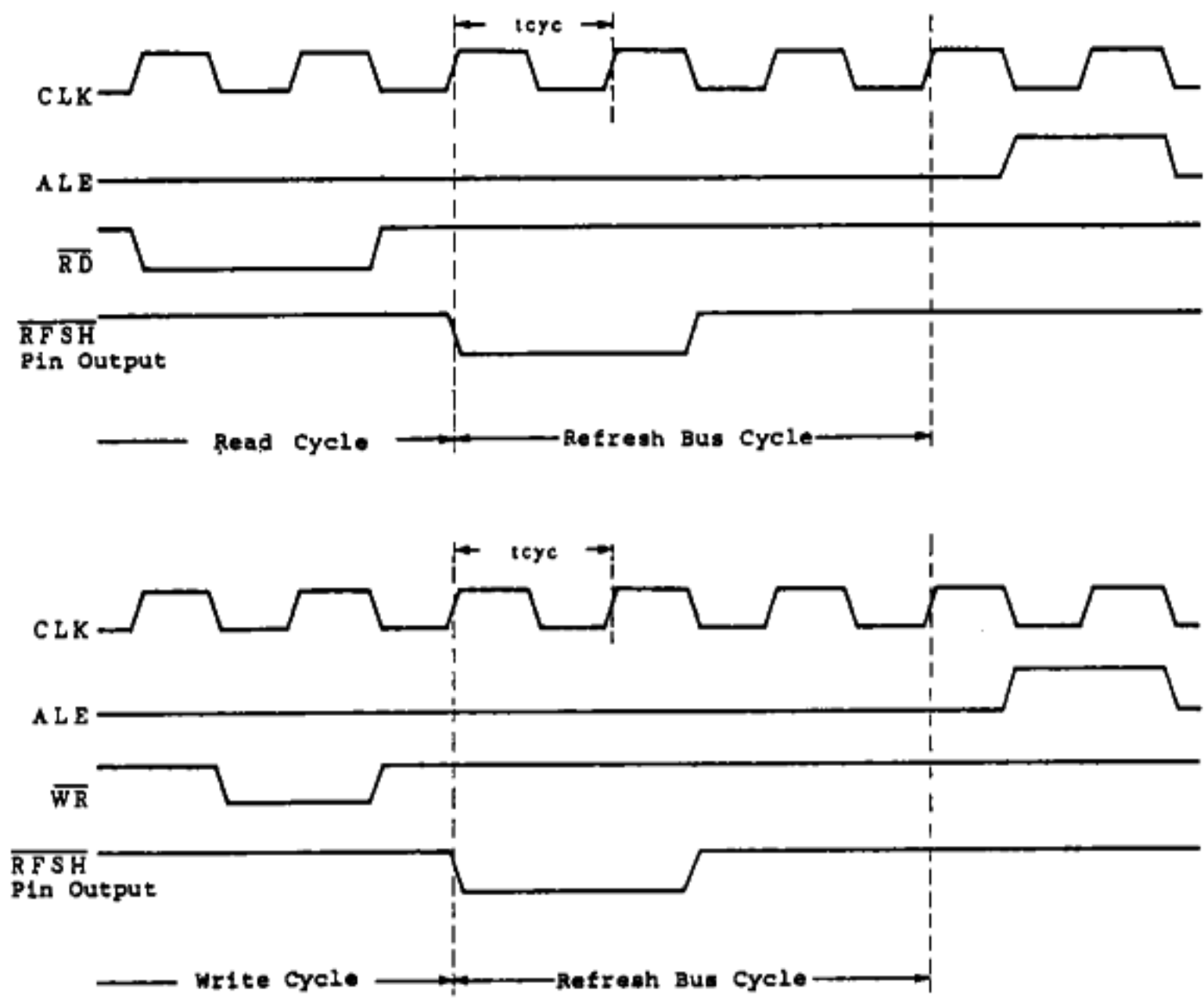
To support uPD428128 pulse refresh cycle, a refresh pulse is output from the  $\overline{\text{RFSH}}$  pin in synchronization with the bus cycle.

If the access timing overlaps with the refresh pulse output timing, the uPD428128 may malfunction. Thus, the uPD78312A generates a refresh pulse in synchronization with the bus cycle. If external memory access cycle concurs with refresh bus cycle, the refresh bus cycle takes precedence over the external memory access cycle and external memory access operation is made to wait.

In the refresh bus cycle, 3-state wait condition occurs.

A refresh bus cycle is generated at the intervals specified in the refresh mode register (RFM).

Figure 8-6 Pulse Refresh Operation when Memory is Accessed



### 8.3.2 POWER DOWN SELF-REFRESH OPERATION

When  $\overline{\text{RFSH}}$  pin output is set low by handling the refresh mode register (RFM) on software, the power down self-refresh mode is entered.

A return from power down self-refresh operation is made by setting the  $\overline{\text{RFSH}}$  pin high on software.

#### (1) Power down self-refresh operation specification

The power down self-refresh operating mode is set by resetting refresh mode register (RFM) bit 7 (RFLV) to 0 on software.

Example:

```

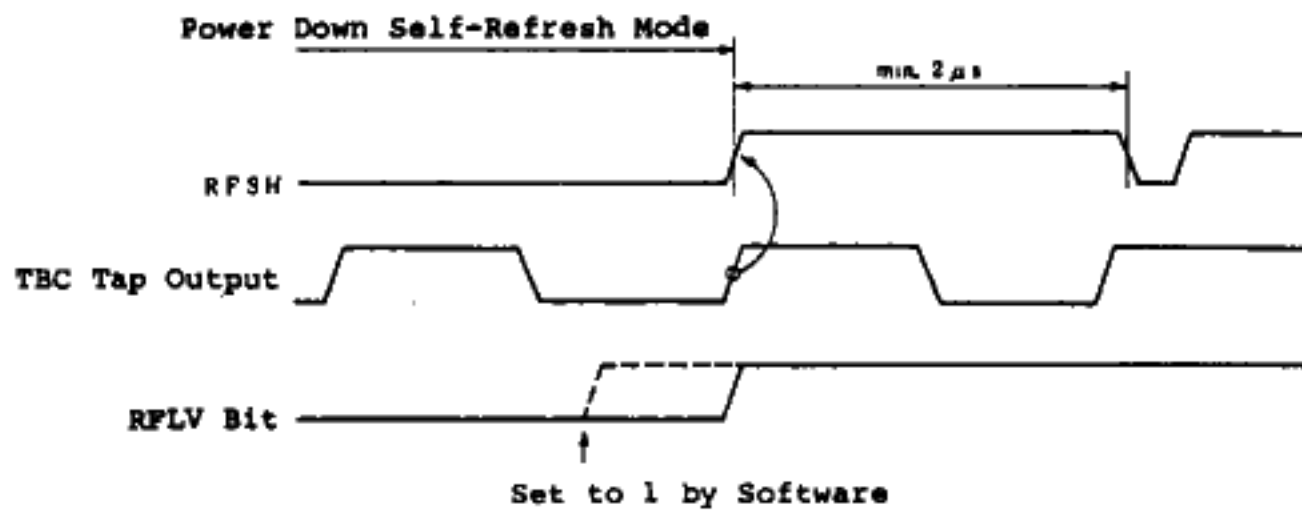
      :
      :
CLR1 RFM.7      ; Reset RFLV bit
MOV  STBC, #0AH; Set STOP mode and set standby flag
      :
      :
```

NOTE: When the power down self-refresh mode is set, be careful so as not to activate the pseudo-static RAM CE pin.

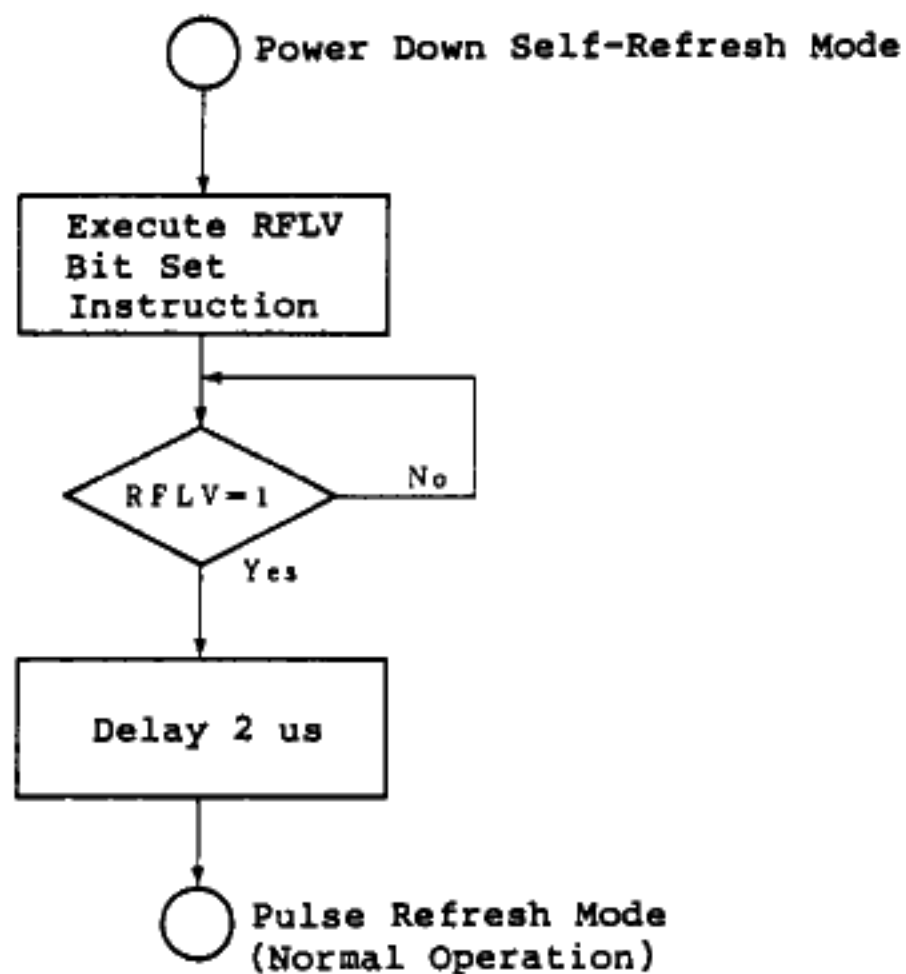
#### (2) Return from power down self-refresh operation

Since refresh pulse to the uPD428128 is disabled for 2 us after the  $\overline{\text{RFSH}}$  line rises, the  $\overline{\text{RFSH}}$  pin rises in synchronization with the time base counter.

Figure 8-7 Return from Power Down Self-Refresh Operation



NOTE: Although the RFLV bit is set to 1 by software, the internal hardware sets the RFLV bit to 1 in synchronization with time base counter tap output. In return from power down self-refresh operation, the following operation is required on software:





### 8.3.3 REFRESH MODE REGISTER (RFM)

The refresh mode register (RFM) is an 8-bit register to control pseudo-static RAM refresh intervals and change to power down self-refresh cycle. Figure 8-8 shows the RFM register format.

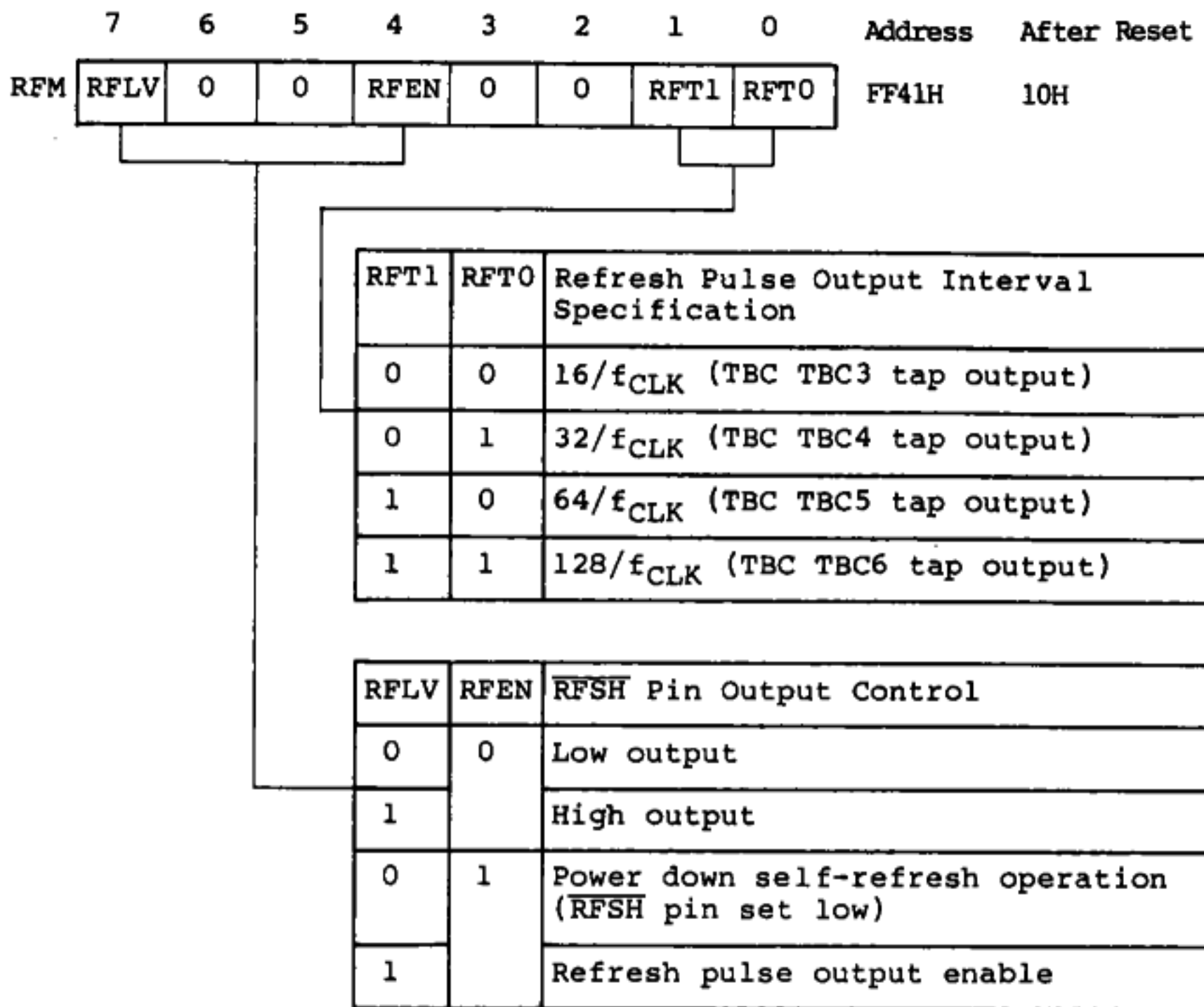
The RFT0 and RFT1 (bits 0 and 1) are a bit field to specify the refresh pulse interval. One of four time base counter (TBC) tap outputs can be selected.

The RFEN bit (bit 4) is used to control refresh pulse output. When the RFEN bit is reset to 0, no refresh pulse is output and the  $\overline{\text{RFSH}}$  pin functions as a 1-bit output port; when the RFEN bit is set to 1, a refresh pulse is output at the specified intervals.

The RFLV bit (bit 7) specifies the  $\overline{\text{RFSH}}$  pin level when the RFEN bit is reset to 0. When the RFLV bit is set to 1, the  $\overline{\text{RFSH}}$  pin is set high; when the RFLV bit is reset to 0, the  $\overline{\text{RFSH}}$  pin is set low. If the RFEN bit is set to 1, the RFLV bit is used to set the power down self-refresh mode. When the RFLV bit is set to 1, pulse refresh operation in operation state is performed; when the RFLV bit is reset to 0, power down self-refresh mode is set. When the standby mode is set, the RFLV bit is reset to 0 by hardware.

When  $\overline{\text{RESET}}$  is input, the RFM register is set to 10H, setting the power down self-refresh mode.

Figure 8-8 Refresh Mode Register (RFM) Format



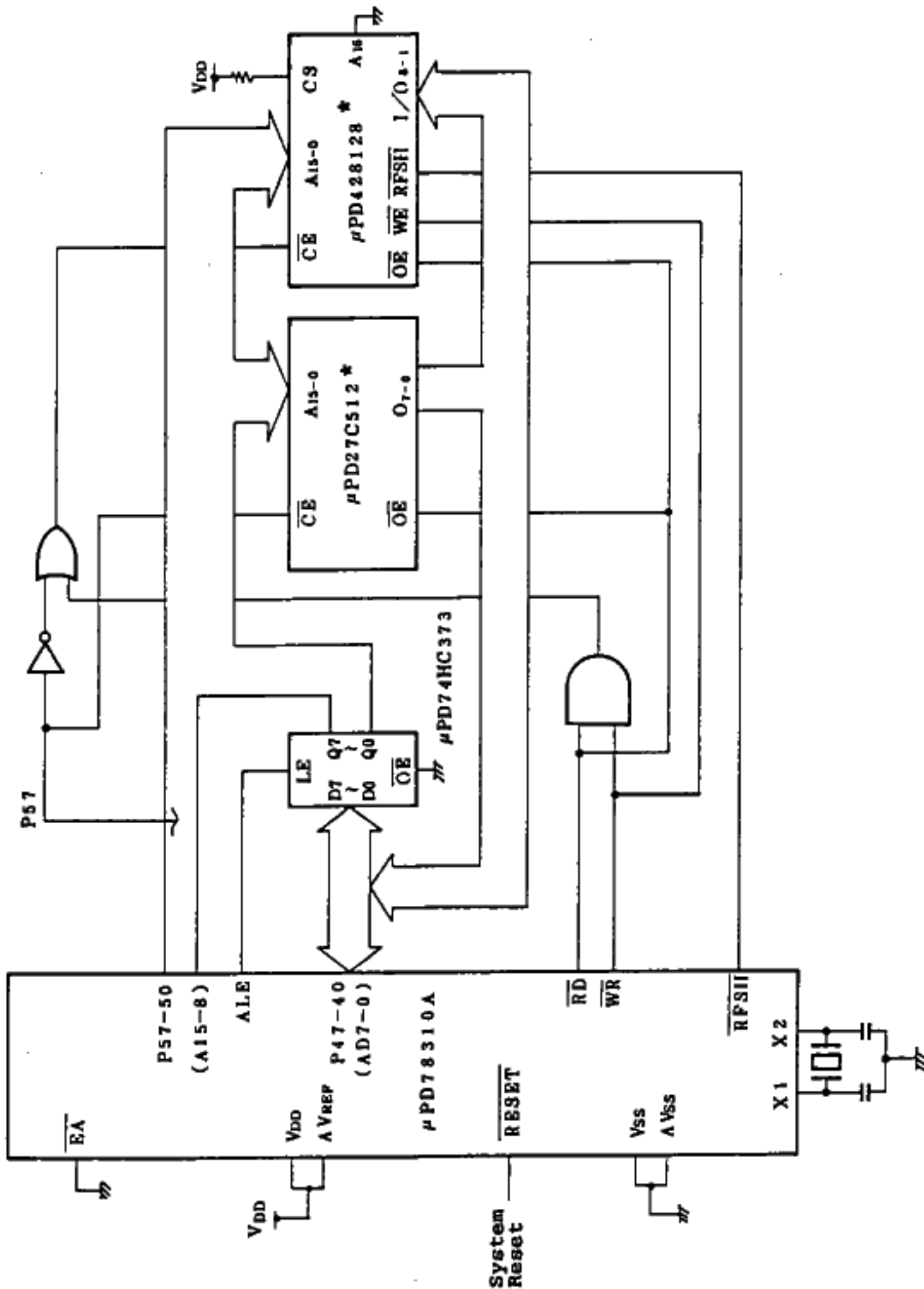
If the refresh function is not used, disable the refresh function during program initialization.

Example:   MOV RFM, #00H  
           MOV STBC, #00H

### 8.3.4 PSEUDO-STATIC MEMORY EXPANSION EXAMPLES

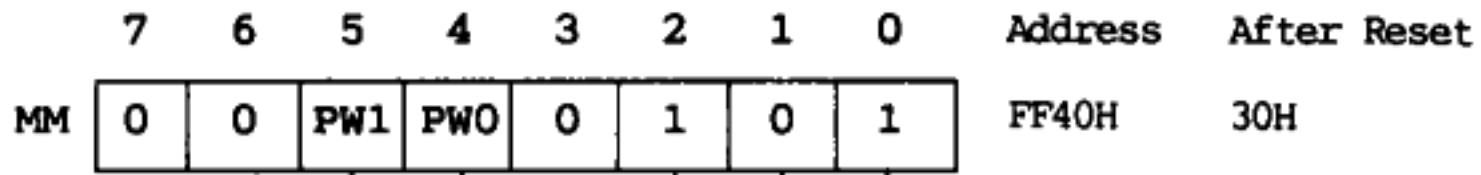
Figure 8-9 shows a configuration example when uPD27C512 of 64K-byte PROM product and 128K-byte pseudo-static RAM uPD428128 are connected to the external. Figure 8-10 shows the data set in the memory expansion mode register in the example.

Figure 8-9 Pseudo-Static Memory Expansion Example (for Reference)



\*: Maintenance product

Figure 8-10 Memory Expansion Mode Register Setting  
(in Pseudo-Static Memory Expansion Example)



P47 to P40 : Expansion Mode  
P57 to P50 : Expansion Mode

PW1	PW0	Wait Cycle Insertion when External Memory is Accessed
0	0	No wait cycle is inserted
0	1	One wait cycle is inserted
1	0	Two wait cycles are inserted
1	1	Three wait cycles are inserted

## CHAPTER 9. uPD78P312A PROGRAMMING

The on-chip ROM of uPD78P312A is a 8192 x 8 bits electrically programmable PROM. The pins listed in the Table 9-1 are used for the programming of the PROM.

In the normal operating mode, 5 V  $\pm 10\%$  is applied to  $V_{DD}$  and  $V_{PP}$ , and a voltage no more than  $V_{DD}$  is applied to all other pins.

The programming characteristics of uPD78P312A are compatible with uPD27C256A.

Table 9-1 Pin Functions in Programming Mode

Pin Name	Functions
$V_{PP}$	High-voltage input (for write/verify) and high-level input (for read)
PROG	High-voltage input (for write/verify and read)
A0 to A7	Address input (lower 8 bits)
A8 to A12	Address input (for upper 5 bits)
D0 to D7	Data input (for write) and data output (for verify)
CE	Program pulse input
OE	Output enable input
$V_{DD}$	Supply voltage input

NOTE 1: Be sure to mask the window of the uPD78P312A with a light-shielding cover film except when erasing EPROM data.

2: Data erasure by ultra-violet light is not possible in the uPD78P312A one-time PROM versions which have no window.

## 9.1 OPERATING MODE FOR PROM PROGRAMMING

If +6 V is applied to  $V_{DD}$  pin and +12.5 V is applied to PROG and  $V_{PP}$  pins, uPD78P312A is set to the program write/verify mode. In this mode, the operating modes described in Table 9-2 are available by setting  $\overline{CE}$  and  $\overline{OE}$  pins.

In the read mode, uPD78P312A can read the PROM contents.

Table 9-2 Operating Mode for PROM Programming

Operating Mode Setting					Operating Mode	
$V_{PP}$	$V_{DD}$	$\overline{CE}$	$\overline{OE}$	PROG		
+12.5 V	+6 V	L	H	+12.5 V	Write mode	
		H	L		Verify mode	
		H	H		Program inhibit mode	
$V_{PP} = V_{DD} = +5 V$		L/H	L		Read mode	Data output from pins D0 to D7
			H			Pins D0 to D7 are in high-impedance state.

Remarks: In the above table, H indicates high level and L, low level.

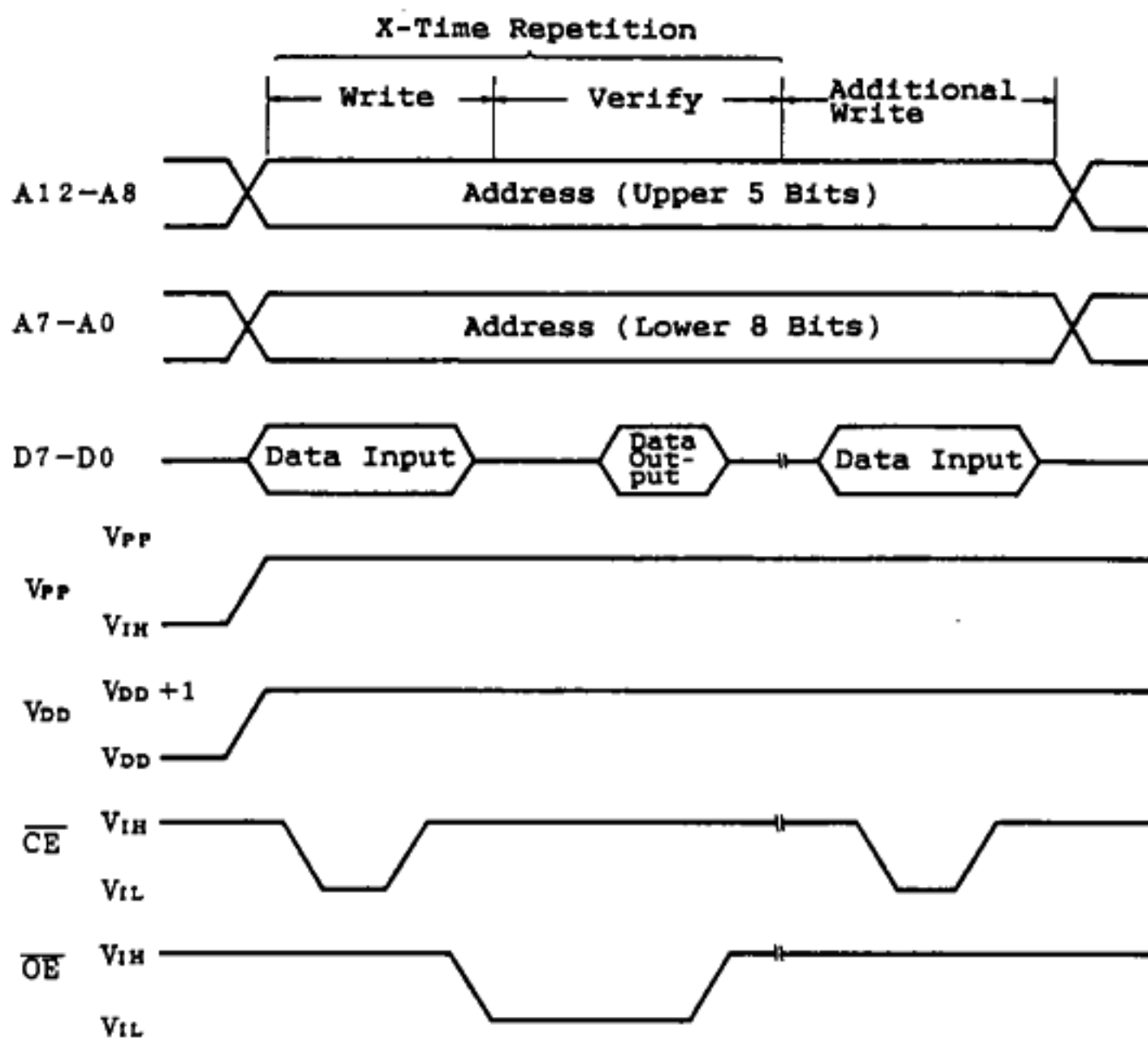
NOTE: When  $V_{PP}$  and  $V_{DD}$  are set to +12.5 V and +6 V, respectively, setting both  $\overline{CE}$  and  $\overline{OE}$  to L is prohibited.

## 9.2 PROM WRITING PROCEDURE

Data write onto PROM can be carried out at high speed as follows:

- (1) Treat the unused pins as shown in directed in 1.3.2 "uPD78P312A Pin Configuration" and supply +6 V to  $V_{DD}$  and +12.5 V to  $V_{PP}$ .
- (2) Supply an initial address.
- (3) Supply write data.
- (4) Supply CE pin with a 1 ms program pulse (active low).
- (5) Set the verify mode. Proceed to step (7) if data has been written or repeat steps (3) to (5) if data has not been written. Proceed to step (6) if data still cannot be written after repeating 25 times.
- (6) Stop the write operation by judging the device to be defective.
- (7) Supply write data and supply program pulses of (X: The number of repetitions of steps (3) to (5)) x 3 ms (additional write).
- (8) Increase the address by increments.
- (9) Repeat steps (3) to (8) up to the last address.

Figure 9-1 PROM Write/Verify Timings





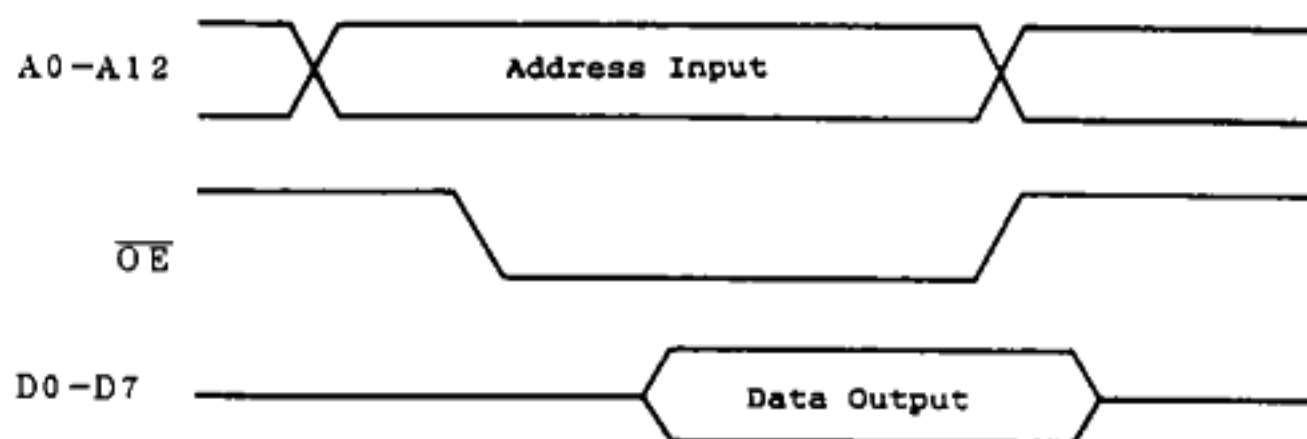
### 9.3 PROM READING PROCEDURE

The PROM contents can be read into the external data buses (D0 to D7) by using the following procedure:

- (1) Treat the unused pins as directed in 1.3.2 "uPD78P312A Pin Configuration".
- (2) Supply +5 V to  $V_{DD}$  and  $V_{PP}$  pins and +12.5 V to the PROG pin.
- (3) Input the address of data to be read to pins A0 to A12.
- (4) Set the read mode.
- (5) Output data to pins D0 to D7.

Figure 9-2 shows the timings for steps (2) to (5).

Figure 9-2 PROM Read Timings



#### 9.4 DATA ERASURE PROCEDURE (EPROM PRODUCT ONLY)

The programmed EPROM data contents of the uPD78P312ADW/R can be erased by irradiation with light having a wavelength of less than approximately 400 nm. The EPROM data contents may be erased if the uPD78P312ADW/R is exposed to direct sunlight or light from a fluorescent lamp. To protect the data contents, mask the uPD78P312ADW/R with light-shielding cover film to prevent ultraviolet light from entering through the upper window. Quality-guaranteed light-shielding cover film is available from NEC together with on-chip EPROM products using a package with a window.

To erase data, irradiate the uPD78P312ADW/R window with ultraviolet light of 254 nm. A minimum of  $15 \text{ W}\cdot\text{s}/\text{cm}^2$  (ultraviolet intensity  $\times$  erasure time) is required to completely erase the EPROM contents of the uPD78P312ADW/R. For example, it will take about 15 to 20 minutes to erase data using a  $12000 \text{ uW}/\text{cm}^2$  ultraviolet lamp. The erasure time may become slightly longer, depending on the life of the ultraviolet lamp or dirt, etc. on the package window. Allow a distance of no more than 2.5 cm between the ultraviolet lamp and the uPD78312ADW/R window.

9.5 ONE-TIME PRODUCTS SCREENING



The one-time product (uPD78P312ACW, 78P312AGF-3BE, 78P312AGQ-36, 78P312AL) cannot be tested completely by NEC before it is shipped, because of its structure. It is recommended to perform screening to verify PROM after writing necessary data and performing storage under the condition below.

Storage Temperature	Storage Time
125°C	24 hours

NEC is offering a pay service, called QTOP™ microcomputer, which comprises one-time PROM write, marking, screening and verify operations. For details, contact our sales personnel.

## CHAPTER 10. INSTRUCTION SET

### 10.1 INSTRUCTION SET AND ITS OPERATION

#### (1) Operand identifier and description

Operands are described in the operand field of each instruction in accordance with the description for the operand identifier of the instruction.

(Details conform to the assembler specification.)

When there are multiple elements in the description, one of the elements is selected. Upper case alphabetic characters and the symbols +, -, #, \$, !, and [ ] are key words and are described unchanged.

For immediate data, a suitable value or label is described. Also always describe the #, \$, !, and [ ] symbols when describing immediate data by label.

Identifier	Description
r	R0, R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15
r1	R0, R1, R2, R3, R4, R5, R6, R7
r2	C, B
rp	RP0, RP1, RP2, RP3, RP4, RP5, RP6, RP7
rp1	RP0, RP1, RP2, RP3, RP4, RP5, RP6, RP7
rp2	DE, HL, VP, UP
sfr	Special function register name (See Table 3-3.)
sfrp	Special function register name (16-bit handleable register; see Table 3-3.)
post	RP0, RP1, RP2, RP3, RP4, RP5/PSW, RP6, RP7  [Multiple descriptions possible. However, RP5 is limited to PUSH and POP instructions and PSW is limited to PUSHU and POPU instructions.]

(to be continued)

Identifier	Description
mem	<p>[DE], [HL], [DE+], [HL+], [DE-], [HL-], [VP], [UP]: Register indirect mode</p> <p>[DE+A], [HL+A], [DE+B], [HL+B], [VP+DE], [VP+HL]: Base index mode</p> <p>[DE+byte], [HL+byte], [VP+byte], [UP+byte], [SP+byte]: Base mode</p> <p>word[A], word[B], word[DE], word[HL]: Index mode</p>
saddr	FE20H to FF1FH Immediate data or label
saddrp	FE20H to FF1EH Immediate data (however, bit 0 = 0) or label (with 16 bit handling)
\$addr16	0000H to FEFFH Immediate data or label; relative addressing
laddr16	<p>0000H to FEFFH Immediate data or label; immediate addressing</p> <p>(However, for MOV instructions, description up to FFFFH is possible.)</p>
addr11	800H to FFFH Immediate data or label
addr5	40H to 7EH Immediate data (however, bit 0 = 0) or label
word	16-bit immediate data or label
byte	8-bit immediate data or label
bit	3-bit immediate data or label
n	3-bit immediate data (0 to 7)

Eight-bit register identifiers *r* and *r1* and 16-bit register pair identifiers *rp*, *rpl*, and *post* can be described by function name, as well as absolute name (R0 to R15, RP0 to RP7). The absolute name function name correspondence is shown in Table 10-1 and Table 10-2.

Table 10-1 8-Bit Register Absolute Name/  
Function Name Correspondence

Absolute Name	Function Name		Absolute Name	Function Name	
	RSS = 0	RSS = 1		RSS = 0	RSS = 1
R0	X		R8	VP <sub>L</sub>	VP <sub>L</sub>
R1	A		R9	VP <sub>H</sub>	VP <sub>H</sub>
R2	C		R10	UP <sub>L</sub>	UP <sub>L</sub>
R3	B		R11	UP <sub>H</sub>	UP <sub>H</sub>
R4		X	R12	E	E
R5		A	R13	D	D
R6		C	R14	L	L
R7		B	R15	H	H

**Table 10-2 16-Bit Register Pair Absolute Name/  
Function Name Correspondence**

Absolute Name	Function Name	
	RSS = 0	RSS = 1
RP0	AX	
RP1	BC	
RP2		AX
RP3		BC
RP4	VP	VP
RP5	UP	UP
RP6	DE	DE
RP7	HL	HL

RSS is the register set select flag (PSW bit 5). The absolute name and function name correspondence is switched by setting and resetting this bit.

**(2) Operation description legend**

- A : A register; 8-bit accumulator
- X : X register
- B : B register
- C : C register
- D : D register
- E : E register
- H : H register
- L : L register
- R0 to R15 : Register 0 to register 15  
(absolute name)
- AX : Register pair (AX); 16-bit accumulator
- BC : Register pair (BC)
- DE : Register pair (DE)

HL : Register pair (HL)  
 RP0 to RP7: Register pair 0 to register pair 7  
 (absolute name)  
 PC : Program counter  
 SP : Stack pointer  
 UP : User stack pointer  
 PSW : Program status word  
 CY : Carry flag  
 AC : Auxiliary carry flag  
 Z : Zero flag  
 P/V : Parity/overflow flag  
 S : Sign flag  
 SUB : Subtraction flag  
 TPF : Table position flag  
 RBS : Register bank select flag  
 RSS : Register set select flag  
 IE : Interrupt enable flag  
 EOS : End of software interrupt flag  
 STBC : Standby control register  
 WDM : Watchdog timer mode register  
 ( ) : Contents of memory indicated by the  
 contents of the address or register in  
 ( ).  
 For ( +) and ( -), after instruction  
 execution, the contents in ( ) are  
 incremented/decremented by one.  
 (( )) : Contents of memory indicated by the  
 contents of memory indicated by the  
 address in (( )).  
 xxH : Hexadecimal number  
 xH, xL : High-order 8 bits, low-order 8 bits of  
 16-bit register.



(3) State field symbols

- (1) When n is described in the state field, its value is determined as follows:
- . Stack manipulation instruction : Number of registers to be saved/restored
  - . Shift rotate instruction : Number of bits to be shifted
  - . String instruction : Number of instruction executions until the condition is satisfied and the program exits from the loop
- (ii) The number in ( ) of the state field of a conditional branch instruction is the minimum number of execution states when not branched.
- (iii) When a SFR is accessed by describing FF00H to FF1FH at saddr, saddrp in an instruction with saddr, saddrp in the operand, the state count is the latter separated by a /.
- (iv) When an interrupt request is accepted during execution of a string instruction, the number in ( ) in the state field of a string instruction is the number of execution states until rewriting the program counter.

- (4) Byte count and state count of instruction containing mem in the operand

The byte count and state count differs with the mode of the contents described at mem as follows:

mem Mode			Register Indirect Mode	Base Index Mode	Base Mode	Index Mode
Byte Count			1*/2	2	3	4
State count (differs with the instruction)	MOV	A, mem	5	6	6	6
		mem, A				
	XCH	A, mem	7	8	8	8
		mem, A				
	ADD, ADDC, SUB, SUBC, AND, OR, XOR	A, mem	6	7	7	7
		mem, A	7	8	8	8
	CMP	A, mem	6	7	7	7
		mem, A				

\*: Becomes a dedicated 1-byte instruction only when [DE], [HL], [DE+], [HL+], [DE-], or [HL-] was described at mem in a MOV instruction.

- (5) Description of symbols of flag operation field

Symbol	Description
(blank)	No change
0	Cleared to 0
1	Set to 1
x	Set/cleared according to result
P	P/V flag operates as parity flag
V	P/V flag operates as overflow flag
U	Undefined
R	Value saved previously restored

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag					
							S	Z	AC	P/V	SUB	CY
8-bit data transfer instruction	MOV	r1, #byte	2	3	3	r1 + byte						
		saddr, #byte	3	3/4	0	(saddr) + byte						
		sfr, * #byte	3	4	0	sfr + byte						
		r, r1	2	3	3	r + r1						
		A, r1	1	3	3	A + r1						
		A, saddr	2	3/4	1	A + (saddr)						
		saddr, A	2	3/4	0	(saddr) + A						
		saddr, saddr	3	4/6	0	(saddr) + (saddr)						
		A, sfr	2	4	1	A + sfr						
		sfr, A	2	4	0	sfr + A						
		A, mem	1 to 4	5, 6	3, 4	A + (mem)						
		mem, A	1 to 4	5, 6	2	(mem) + A						
		A, [saddrp]	2	5/6	1	A + ((saddrp))						
		[saddrp], A	2	4/5	0	((saddrp)) + A						
		A, !addr16	4	5	3	A + (addr16)						
!addr16, A	4	4	2	(addr16) + A								

(to be continued)

\*: When STBC or WDM is described at sfr, the instructions becomes a separate dedicated instructions. The byte count and state count are different from this instruction.

(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag					
							S	Z	AC	P/V	SUB	CY
8-bit data transfer instruction (cont'd)	MOV	PSWL, #byte	3	4	0	PSWL + byte	x	x	x	x	x	x
		PSWH, #byte	3	4	0	PSWH + byte						
		PSWL, A	2	4	0	PSWL + A	x	x	x	x	x	x
		PSWH, A	2	4	0	PSWH + A						
		A, PSWL	2	4	1	A + PSWL						
		A, PSWH	2	4	1	A + PSWH						
	XCH	A, r1	1	4	4	A ↔ r1						
		r, r1	2	4	4	r ↔ r1						
		A, mem	2 to 4	7, 8	3, 4	A ↔ (mem)						
		A, saddr	2	4/6	0	A ↔ (saddr)						
		A, sfr	3	8	3	A ↔ sfr						
		A, [saddrp]	2	6/7	0	A ↔ ((saddrp))						
		saddr, saddr	3	8/12	0	(saddr) ↔ (saddr)						
16-bit data transfer instruction	MOVW	rp1, #word	3	3	3	rp1 + word						
		saddrp, #word	4	3/4	0	(saddrp) + word						
		sfrp, #word	4	4	0	sfrp + word						
		rp, rp1	2	3	3	rp + rp1						
		AX, saddrp	2	3/4	1	AX + (saddrp)						
		saddrp, AX	2	3/4	0	(saddrp) + AX						

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag					
							S	Z	AC	P/V	SUB	CY
16-bit data transfer instruction (cont'd)	MOVW	saddrp, saddrp	3	4/6	0	(saddrp) + (saddrp)						
		AX, sfrp	2	4	1	AX + sfrp						
		sfrp, AX	2	4	0	sfrp + AX						
		rp1, !addr16	4	10	6	rp1 + (addr16)						
		!addr16 rp1	4	8	4	(addr16) + rp1						
	XCHW	AX, saddrp	2	4/6	0	AX ↔ (saddrp)						
		AX, sfrp	3	9	3	AX ↔ sfrp						
		saddrp, saddrp	3	8/12	0	(saddrp) ↔ (saddrp)						
		rp, rp1	2	5	5	rp ↔ rp1						
	8-bit operation instruction	ADD	A, #byte	2	3	3	A, CY + A+byte	x	x	x	V	0
saddr, #byte			3	5/7	0	(saddr), CY + (saddr)+byte	x	x	x	V	0	x
sfr, #byte			4	10	3	sfr, CY + sfr+byte	x	x	x	V	0	x
r, r1			2	3	3	r, CY + r+r1	x	x	x	V	0	x
A, saddr			2	3/4	1	A, CY + A+(saddr)	x	x	x	V	0	x
A, sfr			3	7	4	A, CY + A+sfr	x	x	x	V	0	x
saddr, saddr			3	6/9	0	(saddr), CY + (saddr)+(saddr)	x	x	x	V	0	x
A, mem			2 to 4	6, 7	4, 5	A, CY + A+(mem)	x	x	x	V	0	x

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag					
							S	Z	AC	P/V	SUB	CY
8-bit operation instruction (cont'd)	ADD	mem, A	2 to 4	7, 8	2, 3	(mem), CY + (mem)+A	x	x	x	V	0	x
	ADDC	A, #byte	2	3	3	A, CY + A+byte+CY	x	x	x	V	0	x
		saddr, #byte	3	5/7	0	(saddr), CY + (saddr)+byte+CY	x	x	x	V	0	x
		sfr, #byte	4	10	3	sfr, CY + sfr+byte+CY	x	x	x	V	0	x
		r, r1	2	3	3	r, CY + r+r1+CY	x	x	x	V	0	x
		A, saddr	2	3/4	1	A, CY + A+(saddr)+CY	x	x	x	V	0	x
		A, sfr	3	7	4	A, CY + A+sfr+CY	x	x	x	V	0	x
		saddr, saddr	3	6/9	0	(saddr), CY + (saddr)+(saddr)+CY	x	x	x	V	0	x
		A, mem	2 to 4	6, 7	4, 5	A, CY + A+(mem)+CY	x	x	x	V	0	x
		mem, A	2 to 4	7, 8	2, 3	(mem), CY + (mem)+A+CY	x	x	x	V	0	x
	SUB	A, #byte	2	3	3	A, CY + A-byte	x	x	x	V	1	x
		saddr, #byte	3	5/7	0	(saddr), CY + (saddr)-byte	x	x	x	V	1	x
		sfr, #byte	4	10	3	sfr, CY + sfr-byte	x	x	x	V	1	x
		r, r1	2	3	3	r, CY + r-r1	x	x	x	V	1	x
		A, saddr	2	3/4	1	A, CY + A-(saddr)	x	x	x	V	1	x
A, sfr		3	7	4	A, CY + A-sfr	x	x	x	V	1	x	

(to be continued)

(cont'd)

Instruc- tion Group	Mne- monic	Operand	Bytes	State	Idle State	Operation	Flag					
							S	Z	AC	P/V	SUB	CY
8-bit opera- tion instruc- tion (cont')	SUB	saddr, saddr	3	6/9	0	(saddr), CY + (saddr)- (saddr)	x	x	x	V	1	x
		A, mem	2 to 4	6, 7	4, 5	A, CY + A-(mem)	x	x	x	V	1	x
		mem, A	2 to 4	7, 8	2, 3	(mem), CY + (mem)-A	x	x	x	V	1	x
	SUBC	A, #byte	2	3	3	A, CY + A- byte-CY	x	x	x	V	1	x
		saddr, #byte	3	5/7	0	(saddr), CY + (saddr)- byte-CY	x	x	x	V	1	x
		sfr, #byte	4	10	3	sfr, CY + sfr-byte-CY	x	x	x	V	1	x
		r, r1	2	3	3	r, CY + r- r1-CY	x	x	x	V	1	x
		A, saddr	2	3/4	1	A, CY + A-(saddr)-CY	x	x	x	V	1	x
		A, sfr	3	7	4	A, CY + A- sfr-CY	x	x	x	V	1	x
		saddr, saddr	3	6/9	0	(saddr), CY + (saddr)- (saddr)-CY	x	x	x	V	1	x
		A, mem	2 to 4	6, 7	4, 5	A, CY + A- (mem)-CY	x	x	x	V	1	x
		mem, A	2 to 4	7, 8	2, 3	(mem), CY + (mem)-A-CY	x	x	x	V	1	x
	AND	A, #byte	2	3	3	A + A^byte	x	x		P	0	
		saddr, #byte	3	5/7	0	(saddr) + (saddr)^byte	x	x		P	0	
		sfr, #byte	4	10	3	sfr + sfr^byte	x	x		P	0	

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag					
							S	Z	AC	P/V	SUB	CY
8-bit operation instruction (cont'd)	AND	r, r1	2	3	3	$r + r \wedge r1$	x	x		P	0	
		A, saddr	2	3/4	1	$A + A \wedge (\text{saddr})$	x	x		P	0	
		A, sfr	3	7	4	$A + A \wedge \text{sfr}$	x	x		P	0	
		saddr, saddr	3	6/9	0	$(\text{saddr}) + (\text{saddr}) \wedge (\text{saddr})$	x	x		P	0	
		A, mem	2 to 4	6, 7	4, 5	$A + A \wedge (\text{mem})$	x	x		P	0	
		mem, A	2 to 4	7, 8	2, 3	$(\text{mem}) + (\text{mem}) \wedge A$	x	x		P	0	
	OR	A, #byte	2	3	3	$A + A \vee \text{byte}$	x	x		P	0	
		saddr, #byte	3	5/7	0	$(\text{saddr}) + (\text{saddr}) \vee \text{byte}$	x	x		P	0	
		sfr, #byte	4	10	3	$\text{sfr} + \text{sfr} \vee \text{byte}$	x	x		P	0	
		r, r1	2	3	3	$r + r \vee r1$	x	x		P	0	
		A, saddr	2	3/4	1	$A + A \vee (\text{saddr})$	x	x		P	0	
		A, sfr	3	7	4	$A + A \vee \text{sfr}$	x	x		P	0	
		saddr, saddr	3	6/9	0	$(\text{saddr}) + (\text{saddr}) \vee (\text{saddr})$	x	x		P	0	
		A, mem	2 to 4	6, 7	4, 5	$A + A \vee (\text{mem})$	x	x		P	0	
	mem, A	2 to 4	7, 8	2, 3	$(\text{mem}) + (\text{mem}) \vee A$	x	x		P	0		
	XOR	A, #byte	2	3	3	$A + A \vee \text{byte}$	x	x		P	0	
		saddr, #byte	3	5/7	0	$(\text{saddr}) + (\text{saddr}) \vee \text{byte}$	x	x		P	0	
		sfr, #byte	4	10	3	$\text{sfr} + \text{sfr} \vee \text{byte}$	x	x		P	0	

(to be continued)



(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag					
							S	Z	AC	P/V	SUB	CY
8-bit operation instruction (cont'd)	XOR	r, r1	2	3	3	$r + r \vee r1$	x	x		P	0	
		A, saddr	2	3/4	1	$A + A \vee (saddr)$	x	x		P	0	
		A, sfr	3	7	4	$A + A \vee sfr$	x	x		P	0	
		saddr, saddr	3	6/9	0	$(saddr) + (saddr) \vee (saddr)$	x	x		P	0	
		A, mem	2 to 4	6, 7	4, 5	$A + A \vee (mem)$	x	x		P	0	
		mem, A	2 to 4	7, 8	2, 3	$(mem) + (mem) \vee A$	x	x		P	0	
	CMP	A, #byte	2	3	3	A - byte	x	x	x	V	1	x
		saddr, #byte	3	5/7	1	$(saddr) - \text{byte}$	x	x	x	V	1	x
		sfr, #byte	4	10	4	sfr - byte	x	x	x	V	1	x
		r, r1	2	3	3	$r - r1$	x	x	x	V	1	x
		A, saddr	2	3/4	1	$A - (saddr)$	x	x	x	V	1	x
		A, sfr	3	7	4	$A - sfr$	x	x	x	V	1	x
		saddr, saddr	3	6/8	1	$(saddr) - (saddr)$	x	x	x	V	1	x
		A, mem	2 to 4	6, 7	4, 5	$A - (mem)$	x	x	x	V	1	x
mem, A	2 to 4	6, 7	3, 4	$(mem) - A$	x	x	x	V	1	x		
16-bit operation instruction	ADDW	AX, #word	3	4	4	AX, CY + AX+word	x	x	x	V	0	x
		saddrp, #word	4	5/7	0	$(saddrp), CY + (saddrp) + \text{word}$	x	x	x	V	0	x

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag					
							S	Z	AC	P/V	SUB	CY
16-bit operation instruction (cont'd)	ADDW	sfrp, #word	5	10	3	sfrp, CY + sfrp+word	x	x	x	V	0	x
		rp, rp1	2	4	4	rp, CY + rp+rp1	x	x	x	V	0	x
		AX, saddrp	2	4/5	2	AX, CY + AX+(saddrp)	x	x	x	V	0	x
		AX, sfrp	3	8	5	AX, CY + AX+sfrp	x	x	x	V	0	x
		saddrp, saddrp	3	6/9	0	(saddrp), CY + (saddrp)+(saddrp)	x	x	x	V	0	x
	SUBW	AX, #word	3	4	3	AX, CY + AX-word	x	x	x	V	1	x
		saddrp, #word	4	5/7	0	(saddrp), CY + (saddrp)-word	x	x	x	V	1	x
		sfrp, #word	5	10	3	sfrp, CY + sfrp-word	x	x	x	V	1	x
		rp, rp1	2	4	4	rp, CY + rp-rp1	x	x	x	V	1	x
		AX, saddrp	2	4/5	2	AX, CY + AX-(saddrp)	x	x	x	V	1	x
		AX, sfrp	3	8	5	AX, CY + AX-sfrp	x	x	x	V	1	x
		saddrp, saddrp	3	6/9	0	(saddrp), CY + (saddrp)-(saddrp)	x	x	x	V	1	x
	CMPW	AX, #word	3	4	3	AX - word	x	x	x	V	1	x
		saddrp, #word	4	4/5	1	(saddrp) - word	x	x	x	V	1	x
		sfrp, #word	5	8	4	sfrp - word	x	x	x	V	1	x

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag					
							S	Z	AC	P/V	SUB	CY
16-bit operation instruction (cont'd)	CMPW	rp, rp1	2	4	4	rp - rp1	x	x	x	V	1	x
		AX, saddrp	2	4/5	1	AX - (saddrp)	x	x	x	V	1	x
		AX, sfrp	3	8	4	AX - sfrp	x	x	x	V	1	x
		saddrp, saddrp	3	5/7	1	(saddrp) - (saddrp)	x	x	x	V	1	x
Multiply/divide instruction	MULU	r1	2	18	18	AX + Axr1						
	DIVUW	r1	2	26	26	AX(quotient), r1(remainder) + AX÷r1						
	MULUW	rp1	2	27	27	AX(high-order 16 bits), rp1 (low-order 16 bits) + AXxrpl						
	DIVUX	rp1	2	50	50	AXDE(quotient), rp1 (remainder) + AXDE÷rp1						
Increment/decrement instruction	INC	r1	1	3	3	r1 + r1+1	x	x	x	V	0	
		saddr	2	4/6	0	(saddr) + (saddr)+1	x	x	x	V	0	
	DEC	r1	1	3	3	r1 + r1-1	x	x	x	V	1	
		saddr	2	4/6	0	(saddr) + (saddr)-1	x	x	x	V	1	
	INCW	rp2	1	3	3	rp2 + rp2+1						
		saddrp	3	6/8	2	(saddrp) + (saddrp)+1						
	DECW	rp2	1	3	3	rp2 + rp2-1						
		saddrp	3	6/8	2	(saddrp) + (saddrp)-1						

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag					
							S	Z	AC	P/V	SUB	CY
Shift/rotate instruction	ROR	r1, n	2	4+3n	4+3n	(CY, r1 <sub>7</sub> + r1 <sub>0</sub> , r1 <sub>m-1</sub> + r1 <sub>m</sub> )xn times				P	0	x
	ROL	r1, n	2	4+3n	4+3n	(CY, r1 <sub>0</sub> + r1 <sub>7</sub> , r1 <sub>m+1</sub> + r1 <sub>m</sub> )xn times				P	0	x
	RORC	r1, n	2	4+3n	4+3n	(CY + r1 <sub>0</sub> , r1 <sub>7</sub> + CY, r1 <sub>m-1</sub> + r1 <sub>m</sub> )xn times				P	0	x
	ROLC	r1, n	2	4+3n	4+3n	(CY + r1 <sub>7</sub> , r1 <sub>7</sub> + CY, r1 <sub>m+1</sub> + r1 <sub>m</sub> )xn times				P	0	x
	SHR	r1, n	2	4+3n	4+3n	(CY + r1 <sub>0</sub> , r1 <sub>0</sub> + 0, r1 <sub>m-1</sub> + r1 <sub>m</sub> )xn times	x	x	0	P	0	x
	SHL	r1, n	2	4+3n	4+3n	(CY + r1 <sub>7</sub> , r1 <sub>0</sub> + 0, r1 <sub>m+1</sub> + r1 <sub>m</sub> )xn times	x	x	0	P	0	x
	SHRW	rp1, n	2	4+3n	4+3n	(CY + rp1 <sub>0</sub> , rp1 <sub>15</sub> + 0, rp1 <sub>m-1</sub> + rp1 <sub>m</sub> )xn times	x	x	0	P	0	x
	SHLW	rp1, n	2	4+3n	4+3n	(CY + rp1 <sub>15</sub> , rp1 <sub>0</sub> + 0, rp1 <sub>m+1</sub> + rp1 <sub>m</sub> )xn times	x	x	0	P	0	x
	ROR4	[rp1]	2	7	3	A <sub>3</sub> to A <sub>0</sub> + (rp1) <sub>3</sub> to (rp1) <sub>0</sub> , (rp1) <sub>7</sub> to (rp1) <sub>4</sub> , + A <sub>3</sub> to A <sub>0</sub> , (rp1) <sub>3</sub> to (rp1) <sub>0</sub> + (rp1) <sub>7</sub> to (rp1) <sub>4</sub>						

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag						
							S	Z	AC	P/V	SUB	CY	
Shift/rotate instruction (cont'd)	ROL4	[rp1]	2	7	3	A <sub>3</sub> to A <sub>0</sub> + (rp1) <sub>7</sub> to (rp1) <sub>4</sub> , (rp1) <sub>3</sub> to (rp1) <sub>0</sub> , + A <sub>3</sub> to A <sub>0</sub> , (rp1) <sub>7</sub> to (rp1) <sub>4</sub> + (rp1) <sub>3</sub> to (rp1) <sub>0</sub>							
BCD adjustment instruction	ADJ4		1	3	3	Decimal Adjust Accumulator	x	x	x	P		x	
Bit operation instruction	MOV1	CY, saddr.bit	3	6/7	4	CY + (saddr.bit)						x	
		CY, sfr.bit	3	7	4	CY + sfr.bit						x	
		CY, A.bit	2	6	6	CY + A.bit						x	
		CY, X.bit	2	6	6	CY + X.bit						x	
		CY, PSWH.bit	2	6	6	CY + PSWH.bit						x	
		CY, PSWL.bit	2	6	6	CY + PSWL.bit						x	
		saddr.bit, CY	3	7/8	3	(saddr.bit) + CY							
		sfr.bit, CY	3	8	3	sfr.bit + CY							
		A.bit, CY	2	8	8	A.bit + CY							
X.bit, CY	2	8	8	X.bit + CY									

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag					
							S	Z	AC	P/V	SUB	CY
Bit operation instruction (cont'd)	MOV1	PSWH. bit, CY	2	9	9	PSWH.bit + CY						
		PSWL. bit, CY	2	9	9	PSWL.bit + CY	x	x	x	x	x	
	AND1	CY, saddr. bit	3	6/7	4	CY + CY $\wedge$ (saddr.bit)						x
		CY, /saddr. bit	3	6/7	4	CY + CY $\wedge$ (saddr.bit)						x
		CY, sfr.bit	3	7	4	CY + CY $\wedge$ sfr.bit						x
		CY, /sfr.bit	3	7	4	CY + CY $\wedge$ sfr.bit						x
		CY, A.bit	2	6	6	CY + CY $\wedge$ A.bit						x
		CY, /A.bit	2	6	6	CY + CY $\wedge$ A.bit						x
		CY, X.bit	2	6	6	CY + CY $\wedge$ X.bit						x
		CY, /X.bit	2	6	6	CY + CY $\wedge$ X.bit						x
		CY, PSWH.bit	2	6	6	CY + CY $\wedge$ PSWH.bit						x
		CY, /PSWH.bit	2	6	6	CY + CY $\wedge$ PSWH.bit						x
		CY, PSWL.bit	2	6	6	CY + CY $\wedge$ PSWL.bit						x
		CY, /PSWL.bit	2	6	6	CY + CY $\wedge$ PSWL.bit						x
		OR1	CY, saddr. bit	3	6/7	4	CY + CY V(saddr.bit)					

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag					
							S	Z	AC	P/V	SUB	CY
Bit operation instruction (cont'd)	OR1	CY,/ saddr. bit	3	6/7	4	$CY + CY$ $V(saddr.bit)$						x
		CY, sfr.bit	3	7	4	$CY + CY$ $Vsfr.bit$						x
		CY,/ sfr.bit	3	7	4	$CY + CY$ $Vsfr.bit$						x
		CY, A.bit	2	6	6	$CY + CY$ $VA.bit$						x
		CY,/ A.bit	2	6	6	$CY + CY$ $VA.bit$						x
		CY, X.bit	2	6	6	$CY + CY$ $VX.bit$						x
		CY,/ X.bit	2	6	6	$CY + CY$ $VX.bit$						x
		CY, PSWH.bit	2	6	6	$CY + CYV$ $PSWH.bit$						x
		CY,/ PSWH.bit	2	6	6	$CY + CYV$ $PSWH.bit$						x
		CY, PSWL.bit	2	6	6	$CY + CYV$ $PSWL.bit$						x
	CY,/ PSWL.bit	2	6	6	$CY + CYV$ $PSWL.bit$						x	
	XOR1	CY, saddr. bit	3	6/7	4	$CY + CY$ $\nabla(saddr.bit)$						x
		CY, sfr.bit	3	7	4	$CY + CY$ $\nabla sfr.bit$						x
		CY, A.bit	2	6	6	$CY + CY$ $\nabla A.bit$						x
		CY, X.bit	2	6	6	$CY + CY$ $\nabla X.bit$						x

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag							
							S	Z	AC	P/V	SUB	CY		
Bit operation instruction (cont'd)	XOR1	CY, PSWH.bit	2	6	6	$CY + \overline{CY}$ $\overline{PSWH.bit}$							x	
		CY, PSWL.bit	2	6	6	$CY + \overline{CY}$ $\overline{PSWL.bit}$							x	
	SET1	saddr.bit	2	5/7	1	$(saddr.bit) + 1$								
		sfr.bit	3	8	2	$sfr.bit + 1$								
		A.bit	2	7	7	$A.bit + 1$								
		X.bit	2	7	7	$X.bit + 1$								
		PSWH.bit	2	8	8	$PSWH.bit + 1$								
		PSWL.bit	2	8	8	$PSWL.bit + 1$	x	x	x	x	x	x	x	
	CLR1	saddr.bit	2	5/7	1	$(saddr.bit) + 0$								
		sfr.bit	3	8	2	$sfr.bit + 0$								
		A.bit	2	7	7	$A.bit + 0$								
		X.bit	2	7	7	$X.bit + 0$								
		PSWH.bit	2	8	8	$PSWH.bit + 0$								
		PSWL.bit	2	8	8	$PSWL.bit + 0$	x	x	x	x	x	x	x	
	NOT1	saddr.bit	3	6/8	2	$(saddr.bit) + \overline{(saddr.bit)}$								
		sfr.bit	3	8	2	$sfr.bit + \overline{sfr.bit}$								
		A.bit	2	7	7	$A.bit + \overline{A.bit}$								
		X.bit	2	7	7	$X.bit + \overline{X.bit}$								
		PSWH.bit	2	8	8	$PSWH.bit + \overline{PSWH.bit}$								
		PSWL.bit	2	8	8	$PSWL.bit + \overline{PSWL.bit}$	x	x	x	x	x	x	x	

(to be continued)



(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag					
							S	Z	AC	P/V	SUB	CY
Bit operation instruction (cont'd)	SET1	CY	1	3	3	CY + 1						1
	CLR1	CY	1	3	3	CY + 0						0
	NOT1	CY	1	3	3	CY + $\overline{\text{CY}}$						x
Call/return instruction	CALL	!addr16	3	8	0	(SP-1) + (PC+3) <sub>H</sub> , (SP-2) + (PC+3) <sub>L</sub> , PC + !addr16, SP + SP-2						
	CALLF	!addr11	2	8	0	(SP-1) + (PC+2) <sub>H</sub> , (SP-2) + (PC+2) <sub>L</sub> , PC <sub>15</sub> to PC <sub>11</sub> + 00001, PC <sub>10</sub> to PC <sub>0</sub> + !addr11, SP + SP-2						
	CALLT	[addr5]	1	13	0	(SP-1) + (PC+1) <sub>H</sub> , (SP-2) + (PC+1) <sub>L</sub> , PC <sub>H</sub> + (TPF, 00000000, addr5+1), PC <sub>L</sub> + (TPF, 00000000, addr5), SP + SP-2						
	CALL	rp1	2	9	0	(SP-1) + (PC+2) <sub>H</sub> , (SP-2) + (PC+2) <sub>L</sub> , PC <sub>H</sub> + rp1 <sub>H</sub> , PC <sub>L</sub> + rp1 <sub>L</sub> , SP + SP-2						

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag						
							S	Z	AC	P/V	SUB	CY	
Call/ return instruc- tion (cont'd)	CALL	[rp1]	2	11	0	(SP-1) + (PC+2) <sub>H</sub> , (SP-2) + (PC+2) <sub>L</sub> , PC <sub>H</sub> + (rp1+1), PC <sub>L</sub> + (rp1), SP + SP-2							
	BRK		1	20	0	(SP-1) + PSW <sub>H</sub> , (SP-2) + PSW <sub>L</sub> , (SP-3) + (PC+1) <sub>H</sub> , (SP-4) + (PC+1) <sub>L</sub> , PC <sub>L</sub> + (003EH), PC <sub>H</sub> + (003FH), SP + SP-4, IE + 0							
	RET		1	8	0	PC <sub>L</sub> + (SP), PC <sub>H</sub> + (SP+1), SP + SP+2							
	RET1		1	14	0	PC <sub>L</sub> + (SP), PC <sub>H</sub> + (SP+1), PSW <sub>L</sub> + (SP+2), PSW <sub>H</sub> + (SP+3), SP + SP+4, EOS + 0	R	R	R	R	R	R	
Stack opera- tion instruc- tion	PUSH	post	2	41+4n	41	{(SP-1) + post <sub>H</sub> , (SP-2) + post <sub>L</sub> , SP + SP-2}xn times*							

(to be continued)

\*: n is the number of registers described as post.

(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag						
							S	Z	AC	P/V	SUB	CY	
Stack operation instruction (cont'd)	PUSH	PSW	1	5	1	(SP-1) + PSW <sub>H</sub> , (SP-2) + PSW <sub>L</sub> , SP + SP-2							
	PUSHU	post	2	42+4n	42	{(UP-1) + post <sub>H</sub> , (UP-2) + post <sub>L</sub> , UP + UP-2} × n times*							
	POP	post	2	41+5n	41+n	{post <sub>L</sub> + (SP), post <sub>H</sub> + (SP+1), SP + SP+2 × n times*							
		PSW	1	6	2	PSW <sub>L</sub> + (SP), PSW <sub>H</sub> + (SP+1), SP + SP+2	R	R	R	R	R	R	
	POPU	post	2	42+5n	42+n	{post <sub>L</sub> + (UP), post <sub>H</sub> + (UP+1), UP + UP+2} × n times*							
	MOVW	SP, #word	4	4	4	0	SP + word						
		SP, AX	2	4	4	0	SP + AX						
		AX, SP	2	4	4	1	AX + SP						
	INCW	SP	2	5	5	5	SP + SP+1						
	DECW	SP	2	5	5	5	SP + SP-1						
	Unconditional branch instruction	BR	laddr16	3	4	4	0	PC + laddr16					
rp1			2	5	5	0	PC <sub>H</sub> + rp1 <sub>H</sub> , PC <sub>L</sub> + rp1 <sub>L</sub>						

(to be continued)

\*: n is the number of registers described as post.

(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag					
							S	Z	AC	P/V	SUB	CY
Conditional branch instruction (cont'd)	BLE	\$addr16	3	9(5)	0(5)	PC + \$addr16 if (P/V $\bar{V}$ S)V Z=1						
	BH	\$addr16	3	9(5)	0(5)	PC + \$addr16 if ZVCY=0						
	BNH	\$addr16	3	9(5)	0(5)	PC + \$addr16 if ZVCY=1						
	BT	saddr. bit, \$addr16	3	9(6) /10(7)	0(4)	PC + \$addr16 if (saddr. bit)=1						
		sfr.bit, \$addr16	4	11(8)	0(5)	PC + \$addr16 if sfr.bit=1						
		A.bit, \$addr16	3	10(7)	0(7)	PC + \$addr16 if A.bit=1						
		X.bit, \$addr16	3	10(7)	0(7)	PC + \$addr16 if X.bit=1						
		PSWH. bit, \$addr16	3	10(7)	0(7)	PC + \$addr16 if PSW <sub>H</sub> , bit=1						
		PSWL. bit, \$addr16	3	10(7)	0(7)	PC + \$addr16 if PSW <sub>L</sub> , bit=1						
	BF	saddr. bit, \$addr16	4	10(7) /11(8)	0(5)	PC + \$addr16 if (saddr. bit)=0						
		sfr.bit, \$addr16	4	11(8)	0(5)	PC + \$addr16 if sfr.bit=0						
		A.bit, \$addr16	3	10(7)	0(7)	PC + \$addr16 if A.bit=0						
		X.bit, \$addr16	3	10(7)	0(7)	PC + \$addr16 if X.bit=0						
		PSWH. bit, \$addr16	3	10(7)	0(7)	PC + \$addr16 if PSW <sub>H</sub> . bit=0						

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag					
							S	Z	AC	P/V	SUB	CY
Unconditional branch instruction (cont'd)	BR	[rp1]	2	8	0	PC <sub>H</sub> + (rp1+1), PC <sub>L</sub> + (rp1)						
		\$addr16	2	7	0	PC + \$addr16						
Conditional branch instruction	BC	\$addr16	2	7(3)	0(3)	PC + \$addr16 if CY=1						
	BL											
	BNC	\$addr16	2	7(3)	0(3)	PC + \$addr16 if CY=0						
	BNL											
	BZ	\$addr16	2	7(3)	0(3)	PC + \$addr16 if Z=1						
	BE											
	BNZ	\$addr16	2	7(3)	0(3)	PC + \$addr16 if Z=0						
	BNE											
	BV	\$addr16	2	7(3)	0(3)	PC + \$addr16 if P/V=1						
	BPE											
	BNV	\$addr16	2	7(3)	0(3)	PC + \$addr16 if P/V=0						
	BPO											
	BN	\$addr16	2	7(3)	0(3)	PC + \$addr16 if S=1						
	BP	\$addr16	2	7(3)	0(3)	PC + \$addr16 if S=0						
	BGT	\$addr16	3	9(5)	0(5)	PC + \$addr16 if (P/V'S) VZ=0						
	BGE	\$addr16	3	9(5)	0(5)	PC + \$addr16 if P/V'S=0						
BLT	\$addr16	3	9(5)	0(5)	PC + \$addr16 if P/V'S=1							

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag					
							S	Z	AC	P/V	SUB	CY
Conditional branch instruction (cont'd)	BF	PSWL. bit, \$addr16	3	10(7)	0(7)	PC + \$addr16 if PSWL. bit=0						
	BTCLR	saddr. bit, \$addr16	4	12(7) /14(8)	0(5)	PC + \$addr16 if (saddr. bit)=1 then reset (saddr.bit)						
		sfr.bit, \$addr16	4	14(8)	0(5)	PC + \$addr16 if sfr.bit=1 then reset sfr.bit						
		A.bit, \$addr16	3	11(7)	0(7)	PC + \$addr16 if A.bit=1 then reset A.bit						
		X.bit, \$addr16	3	11(7)	0(7)	PC + \$addr16 if X.bit=1 then reset X.bit						
		PSWH. bit, \$addr16	3	12(7)	0(7)	PC + \$addr16 if PSWH. bit=1 then reset PSWH.bit						
		PSWL. bit, \$addr16	3	12(7)	0(7)	PC + \$addr16 if PSWL. bit=1 then reset PSWL.bit						
	BFSET	saddr. bit, \$addr16	4	12(7) /14(8)	0(5)	PC + \$addr16 if (saddr. bit)=0 then set (saddr.bit)						
		sfr.bit, \$addr16	4	14(8)	0(5)	PC + \$addr16 if sfr.bit=0 then set sfr.bit						

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag					
							S	Z	AC	P/V	SUB	CY
Conditional branch instruction (cont'd)	BFSET	A.bit, \$addr16	3	11(7)	0(7)	PC + \$addr16 if A.bit=0 then set A.bit						
		X.bit, \$addr16	3	11 7)	0(7)	PC + \$addr16 if X.bit=0 then set X.bit						
		PSWH.bit, \$addr16	3	12(7)	0(7)	PC + \$addr16 if PSWH.bit=0 then set PSWH.bit						
		PSWL.bit, \$addr16	3	12(7)	0(7)	PC + \$addr16 if PSWL.bit=0 then set PSWL.bit						
	DBNZ	r2, \$addr16	2	8(5)	0(5)	r2 + r2-1, then PC + \$addr16 if r2≠0						
		saddr, \$addr16	3	9(6) /11(8)	0(2)	(saddr) + (saddr)-1, then PC + \$addr16 if (saddr)≠0						
Context switching instruction	BRKCS	R <sub>n</sub>	2	12	0	PC <sub>H</sub> ↔ R <sub>5</sub> , PC <sub>L</sub> ↔ R <sub>4</sub> , R <sub>7</sub> + PSW <sub>H</sub> , R <sub>6</sub> + PSW <sub>L</sub> , RBS <sub>2</sub> to RBS <sub>0</sub> + n, RSS + 0, IE + 0						

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag					
							S	Z	AC	P/V	SUB	CY
Context switching instruction (cont'd)	RETCS	!addr16	3	6	0	PC <sub>H</sub> + R5, PC <sub>L</sub> + R4, R5, R4 + !addr16, PSW <sub>H</sub> + R7, PSW <sub>L</sub> + R6, EOS + 0	R	R	R	R	R	R
String instruction	MOVMM	[DE+], A	2	2+7n (4+7n)	2+5n (3+5n)	(DE+) + A, C + C-1 End if C=0						
		[DE-], A	2	2+7n (4+7n)	2+5n (3+5n)	(DE-) + A, C + C-1 End if C=0						
	MOVBK	[DE+], [HL+]	2	2+10n (4+10n)	2+6n (3+6n)	(DE+) + (HL+), C + C-1 End if C=0						
		[DE-], [HL-]	2	2+10n (4+10n)	2+6n (3+6n)	(DE-) + (HL-), C + C-1 End if C=0						
	XCHM	[DE+], A	2	2+12n (4+12n)	2+6n (3+6n)	(DE+) ↔ A, C + C-1 End if C=0						
		[DE-], A	2	2+12n (4+12n)	2+6n (3+6n)	(DE-) ↔ A, C + C-1 End if C=0						
	XCHBK	[DE+], [HL+]	2	2+15n (4+15n)	2+7n (3+7n)	(DE+) ↔ (HL+), C + C-1 End if C=0						
		[DE-], [HL-]	2	2+15n (4+15n)	2+7n (3+7n)	(DE-) ↔ (HL-), C + C-1 End if C=0						

(to be continued)



(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag					
							S	Z	AC	P/V	SUB	CY
String instruction (cont'd)	CMPME	[DE+], A	2	2+7n (4+7n)	2+5n (3+5n)	(DE+) - A, C + C-1 End if C=0 or Z=0	x	x	x	V	1	x
		[DE-], A	2	2+7n (4+7n)	2+5n (3+5n)	(DE-) - A, C + C-1 End if C=0 or Z=0	x	x	x	V	1	x
	CMP BKE	[DE+], [HL+]	2	2+10n (4+10n)	2+6n (3+6n)	(DE+) - (HL+), C + C-1 End if C=0 or Z=0	x	x	x	V	1	x
		[DE-], [HL-]	2	2+10n (4+10n)	2+6n (3+6n)	(DE-) - (HL-), C + C-1 End if C=0 or Z=0	x	x	x	V	1	x
	CMP MNE	[DE+], A	2	2+7n (4+7n)	2+5n (3+5n)	(DE+) - A, C + C-1 End if C=0 or Z=1	x	x	x	V	1	x
		[DE-], A	2	2+7n (4+7n)	2+5n (3+5n)	(DE-) - A, C + C-1 End if C=0 or Z=1	x	x	x	V	1	x
	CMPB BKE	[DE+], [HL+]	2	2+10n (4+10n)	2+6n (3+6n)	(DE+) - (HL+), C + C-1 End if C=0 or Z=1	x	x	x	V	1	x
		[DE-], [HL-]	2	2+10n (4+10n)	2+6n (3+6n)	(DE-) - (HL-), C + C-1 End if C=0 or Z=1	x	x	x	V	1	x

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag					
							S	Z	AC	P/V	SUB	CY
String instruction (cont'd)	CMPMC	[DE+], A	2	2+7n (4+7n)	2+5n (3+5n)	(DE+) - A, C + C-1 End if C=0 or CY=0	x	x	x	V	1	x
		[DE-], A	2	2+7n (4+7n)	2+5n (3+5n)	(DE-) - A, C + C-1 End if C=0 or CY=0	x	x	x	V	1	x
	CMPB BKC	[DE+], [HL+]	2	2+10n (4+10n)	2+6n (3+6n)	(DE+) - (HL+), C + C-1 End if C=0 or CY=0	x	x	x	V	1	x
		[DE-], [HL-]	2	2+10n (4+10n)	2+6n (3+6n)	(DE-) - (HL-), C + C-1 End if C=0 or CY=0	x	x	x	V	1	x
	CMP MNC	[DE+], A	2	2+7n (4+7n)	2+5n (3+5n)	(DE+) - A, C + C-1 End if C=0 or CY=1	x	x	x	V	1	x
		[DE-], A	2	2+7n (4+7n)	2+5n (3+5n)	(DE-) - A, C + C-1 End if C=0 or CY=1	x	x	x	V	1	x
	CMPB KNC	[DE+], [HL+]	2	2+10n (4+10n)	2+6n (3+6n)	(DE+) - (HL+), C + C-1 End if C=0 or CY=1	x	x	x	V	1	x
		[DE-], [HL-]	2	2+10n (4+10n)	2+6n (3+6n)	(DE-) - (HL-), C + C-1 End if C=0 or CY=1	x	x	x	V	1	x

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	Bytes	State	Idle State	Operation	Flag					
							S	Z	AC	P/V	SUB	CY
CPU control instruction	MOV	STBC, #byte	4	6	1	STBC + byte						
		WDM, #byte	4	6	1	WDM + byte						
	SWRS		1	3	3	RSS + $\overline{\text{RSS}}$						
	SEL	RBn	2	4	4	RBS2 to RBS0 + n, RSS + 0						
		RBn, ALT	2	4	4	RBS2 to RBS0 + n, RSS + 1						
CPU control instruction (cont'd)	NOP		1	3	3	No Operation						
	EI		1	3	3	IE + 1 (Enable Interrupt)						
	DI		1	3	3	IE + 0 (Disable Interrupt)						

## 10.2 INSTRUCTION EXECUTION STATE COUNT ESTIMATION

The uPD78312A has a 3-byte instruction prefetch queue. Instructions are speeded up by pipelining. The OP code is fetched via a peripheral bus which accesses the on-chip hardware. At instruction execution, when the peripheral bus is idle, the OP code is fetched over the peripheral bus concurrently with instruction execution.

The state count of each instruction shown in the table of the preceding section is the instruction execution state count excluding the fetch cycle. The idle state count is the number of instruction execution state at which the peripheral bus was not accessed.

The instruction execution state count is estimated by first computing the basic value from the equation given below.

However, when computed value < state count, the state count becomes the basic value of the state count required at instruction execution.

- . When program in internal ROM  
(state count) + 1 x (execution instruction byte count)  
- (idle state count)
  
- . When program in external ROM  
(state count) + (4 + m) x (execution instruction byte  
count) - (idle state count)
  
- m: Number of waits inserted by MM register  
specification
  
- . When program in internal RAM  
(state count) + 3 x (execution instruction byte count)

Next, whether or not the instruction to be computed corresponds to the following instructions is checked.

- . When vector table and CALLT table in external memory,  
CALLT and BRK instructions
  
- . Instruction which access the special function register  
(SFR) and external memory

When it does not correspond to these instructions, the basic value becomes the instruction execution state count. When it corresponds to these instructions, the instruction execution state is increased over the basic value. The state count which is added is shown on the next page.

- (i) When vector table and CALLT table in external memory

Since the CALLT and BRK instructions fetch the branch destination address from external memory, the instruction execution state count is increased by  $(12+4m)$  states. ( $m$  is the number of waits specified by the MM register.)

- (ii) When special function register (SFR) and external memory accessed

When a special function register (SFR) and external memory were described in the operands, the instruction execution state count is increased above the equation above. The increase of the state count per instruction execution is shown in Table 10-3.

(The number of times the SFR and memory are accessed by each instruction is shown in Table 10-4 and Table 10-5.)

Table 10-3 State Count Increase/Instruction Execution

Access Objective	State Count Increase/ Instruction Execution
Internal memory (ROM, RAM)	0
Special function register (SFR)	Number of SFR accesses $\times k$
External memory	Number of external memory accesses $\times (2+m)$

$k$ : Number of waits generated at timer unit and count unit counter access. It is not fixed, but changes within the range shown below with the state of the counter at the time of access.

Accessed SFR	Value of $k$
Timer unit (TM $n$ , MD $n$ )	0 to 5
Counter unit (UDC $n$ , CR $nn$ )	0 to 2
Other SFR	0

( $n = 0, 1$ )

$m$ : Number of waits at external memory access specified by MM register

When the internal system clock is specified as 6 MHz, 1 state time becomes approximately 167 ns.

Table 10-4 Number of SFR Accesses of Each Instruction

Mnemonic	Operands	No. of SFR Accesses
MOV	sfr, #byte	1
	A, sfr	
	sfr, A	
XCH	A, sfr	2
MOVW	sfrp, #word	1
	AX, sfrp	
	sfrp, AX	
XCHW	AX, sfrp	2
ADD, ADDC, SUB, SUBC, AND, OR, XOR	sfr, #byte	2
	A, sfr	1
CMP	sfr, #byte	1
	A, sfr	
ADDW, SUBW	sfrp, #word	2
	AX, sfrp	1
CMPW	sfrp, #word	1
	AX, sfrp	
MOV1	CY, sfr.bit	1
	sfr.bit, CY	2
AND1, OR1, XOR1	CY, sfr.bit	1
SET1, CLR1, NOT1	sfr.bit	2
BT, BF	sfr.bit, \$addr16	1

(to be continued)

(cont'd)



Mnemonic	Operands	No. of SFR Accesses
BTCLR, BFSET	sfr.bit, \$addr16	1/2*

\*: When the condition is satisfied and the program branches, the SFR is accessed twice.

Table 10-5 Number of Memory Accesses of Each Instruction

Mnemonic	Operands	No. of Memory Accesses
MOV	A, mem	1
	mem, A	
	A, [saddr]	
	[saddr], A	
	A, !addr16	
	!addr16, A	
XCH	A, mem	2
	mem, A	
MOVW	rpl, !addr16	2
	!addr16, rpl	
ADD, ADDC, SUB, SUBC, AND, OR, XOR	A, mem	1
	mem, A	2
CMP	A, mem	1
	mem, A	
ROR4, ROL4	[rpl]	2
CALL	!addr16	2
	rpl	
	[rpl]	4

(to be continued)

(cont'd)

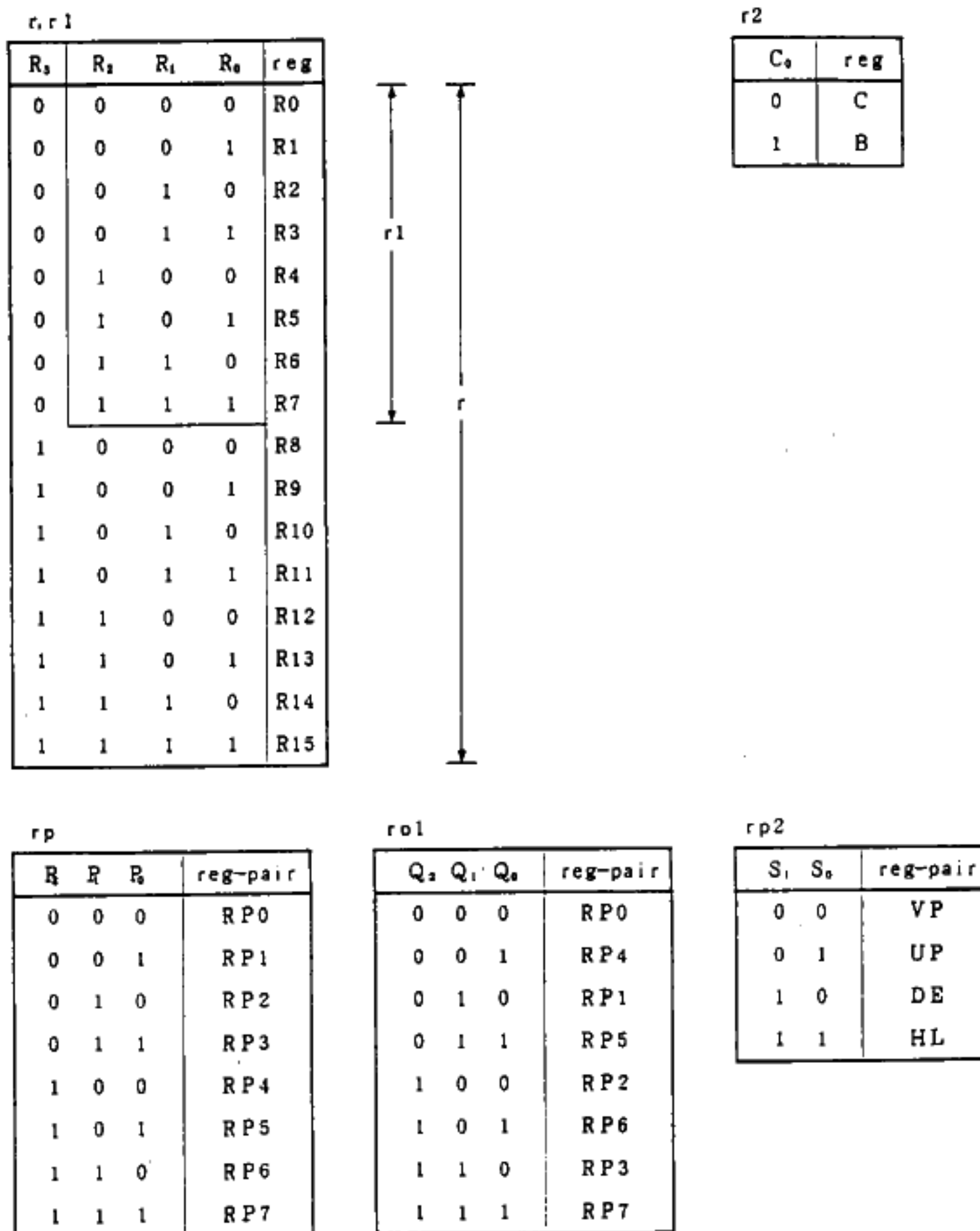
Mnemonic	Operands	No. of Memory Accesses
CALLF	!addr11	2
CALLT	[addr5]	2
BRK		4
RET		2
RETI		4
PUSH, POP	post	2 n
	PSW	2
PUSHU, POPU	post	2 n
BR	[rp1]	2
MOVM, CMPME, CPMNE, CMPMC, CPMNC	[DE+], A	1 n
	[DE-], A	
MOVBK, CMPBKE, CMPBKNE, CMPBKC, CMPBKNC	[DE+], [HL+]	2 n
	[DE-], [HL-]	
XCHM	[DE+], A	2 n
	[DE-], A	
XCHBK	[DE+], [HL+]	4 n
	[DE-], [HL-]	

n: For PUSH/POP instructions, it is the register count described at post. For a string instruction, it is the number of iterations.



### 10.3 OP CODE OF EACH INSTRUCTION

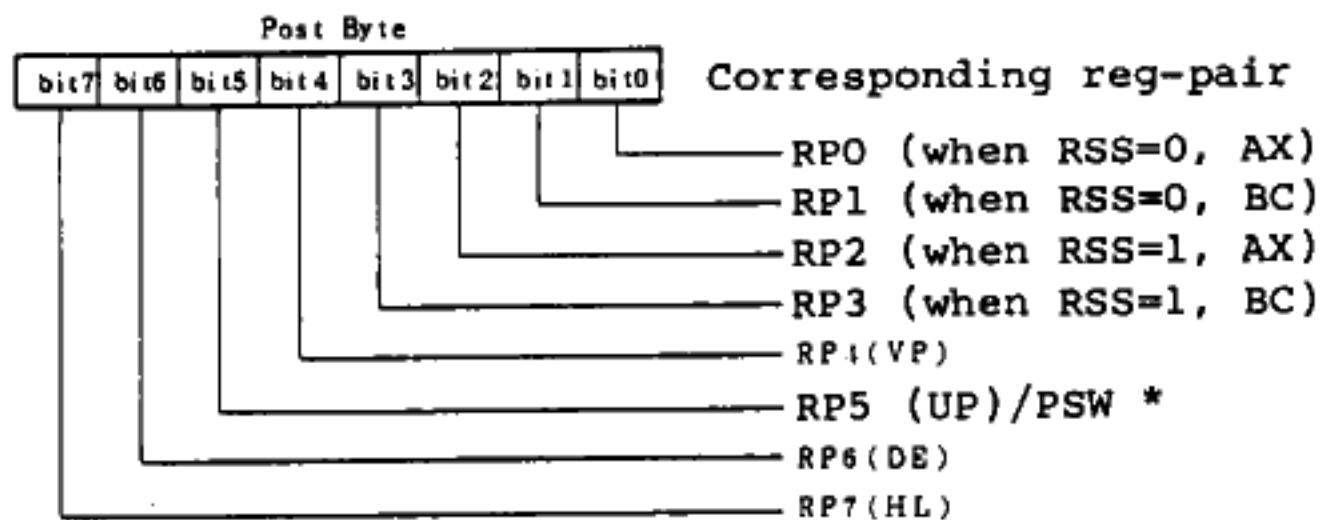
#### (1) Description of OP code symbols



- B<sub>n</sub> : Immediate data for bit
- N<sub>n</sub> : Immediate data for n
- Data : 8-bit immediate data corresponding to byte

- Low/High Byte : 16-bit immediate data corresponding to word
- Saddr-offset : Low-order 8 bits offset data of 16-bit address corresponding to saddr
- Sfr-offset : Low-order 8 bits data of 16-bit address of special function register (sfr)
- Low/High offset: 8/16 bit offset data at base mode/index mode memory addressing
- Low/High Addr. : 16-bit immediate data corresponding to addr16
- jdisp : Signed two's complement data (8 bits) of relative address distance with start address and branch destination address of next instruction
- fa : Low-order 11 bits of immediate data corresponding to addr11
- ta : Low-order 5 bits of immediate data corresponding to  $\text{addr5} \times 1/2$
- Post Byte : 8-bit data which specifies reg-pair which performs stack manipulation

Each reg-pair is assigned to a bit and is specified by its contents (0/1).  
(See the figure below.)



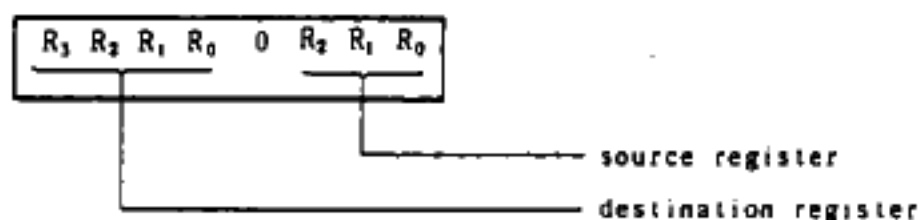
0	Do not perform stack memory save/restore operation
1	Perform stack memory save/restore operation

\*: Becomes RP5 (UP) for PUSH/POP instruction and PSW for PUSHU/POPU instruction.

NOTE 1: When the source and destination at the MOV r or rl and ADD saddr or saddr, etc. operand field are both registers or both are saddr, saddrp, the code becomes:

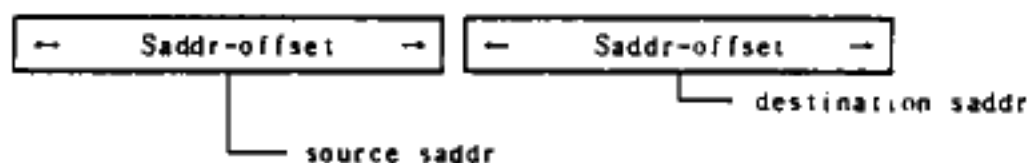
- . When the source and destination are both registers, the destination code comes first and the source code comes last. (This also applies to register pairs.)

Example



- . When the source and destination are both saddr, saddrp, the first 1 byte data is the offset data which specifies the source and the last 1 byte data is the offset data which specifies the destination.

Example



- 2: When a special function register (SFR) mapped to FFO0H to FF1FH was described as the sfr, sfrp operands, short direct addressing is used instead of SFR addressing and the generated OP code becomes the OP code of the saddr, saddrp instruction.

Examples AND A, P5

OP code 1 0 0 1 1 1 0 0 0 0 0 0 0 1 0 1

AND A, PM5

OP code 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0

0 0 1 0 0 1 0 1

In this case, since the AND A, P5 instruction uses short direct addressing, the OP code is shorter than for SFR addressing.

(2) OP code of each memory addressing mode

The code of the mod and mem parts of the OP code field corresponding to the contents described at mem of the operand field is shown below.

mod \ mem		1 0 1 1 0	1 0 1 1 1	0 0 1 1 0	0 1 0 1 0
		Register Indirect Mode	Base Index Mode	Base Mode	Index Mode
0 0 0	[DE+] *	[DE+A]	[DE+byte]	word [DE]	
0 0 1	[HL+] *	[HL+A]	[SP+byte]	word [A]	
0 1 0	[DE-] *	[DE+B]	[HL+byte]	word [HL]	
0 1 1	[HL-] *	[HL+B]	[UP+byte]	word [B]	
1 0 0	[DE] *	[VP+DE]	[VP+byte]	-	
1 0 1	[HL] *	[VP+HL]	-	-	
1 1 0	[VP]	-	-	-	
1 1 1	[UP]	-	-	-	

Remarks 1: For a MOV instruction, when the register indirect mode marked with an asterisk is described at mem, it becomes a dedicated 1 byte instruction.

2: When base mode/index mode is described at mem, 8-bit/16-bit offset data corresponding to byte/word is added, starting from byte 3.

Instruction Group	Mnemonic	Operand	OP Code		
			B1	B2	B3
			B4	B5	
8-bit data transfer instruction	MOV	r1, #byte	1 0 1 1 1 R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	+ Data +	
		saddr, #byte	0 0 1 1 1 0 1 0	+ Saddr-offset +	+ Data +
		sfr, #byte	0 0 1 0 1 0 1 1	+ Sfr-offset +	+ Data +
		r, r1	0 0 1 0 0 1 0 0	R <sub>3</sub> R <sub>2</sub> R <sub>1</sub> R <sub>0</sub> 0 R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	
		A, r1	1 1 0 1 0 R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>		
		A, saddr	0 0 1 0 0 0 0 0	+ Saddr-offset +	
		saddr, A	0 0 1 0 0 0 1 0	+ Saddr-offset +	
		saddr, saddr	0 0 1 1 1 0 0 0	+ Saddr-offset +	+ Saddr-offset +
		A, sfr	0 0 0 1 0 0 0 0	+ Sfr-offset +	
		sfr, A	0 0 0 1 0 0 1 0	+ Sfr-offset +	
		A, mem *	0 1 0 1 1 mem		
			0 0 0 mod	0 mem 0 0 0 0	+ Low Offset +
			+ High Offset +		
		mem, A *	0 1 0 1 0 mem		
			0 0 0 mod	1 mem 0 0 0 0	+ Low Offset +
			+ High Offset +		
A, [saddrp]	0 0 0 1 1 0 0 0	+ Saddr-offset +			

(to be continued)

\*: When [DE], [HL], [DE+], [DE-], [HL+], or [HL-] is described at mem, it becomes this 1 byte code.

(cont'd)

Instruction Group	Mnemonic	Operand	OP Code		
			B1	B2	B3
			B4	B5	
8-bit data transfer instruction (cont'd)	MOV	[saddrp], A	0 0 0 1 1 0 0 1	+ Saddr-offset →	
		A, !addr16	0 0 0 0 1 0 0 1	1 1 1 1 0 0 0 0	+ Low Addr. →
			+ High Addr. →		
		!addr16, A	0 0 0 0 1 0 0 1	1 1 1 1 0 0 0 1	+ Low Addr. →
			+ High Addr. →		
		PSWL, #byte	0 0 1 0 1 0 1 1	1 1 1 1 1 1 1 0	+ Data →
		PSWH, #byte	0 0 1 0 1 0 1 1	1 1 1 1 1 1 1 1	+ Data →
		PSWL, A	0 0 0 1 0 0 1 0	1 1 1 1 1 1 1 0	
		PSWH, A	0 0 0 1 0 0 1 0	1 1 1 1 1 1 1 1	
		A, PSWL	0 0 0 1 0 0 0 0	1 1 1 1 1 1 1 0	
	A, PSWH	0 0 0 1 0 0 0 0	1 1 1 1 1 1 1 1		
	XCH	A, r1	1 1 0 1 1 R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>		
		r, r1	0 0 1 0 0 1 0 1	R <sub>3</sub> R <sub>2</sub> R <sub>1</sub> R <sub>0</sub> 0 R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	
		A, mem	0 0 0 mod	0 mem 0 1 0 0	+ Low Offset →
			+ High Offset →		
		A, saddr	0 0 1 0 0 0 0 1	+ Saddr-offset →	
		A, sfr	0 0 0 0 0 0 0 1	0 0 1 0 0 0 0 1	+ Sfr-offset →
		A, [saddrp]	0 0 1 0 0 0 1 1	+ Saddr-offset →	
		saddr, saddr	0 0 1 1 1 0 0 1	+ Saddr-offset →	+ Saddr-offset →

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	OP Code		
			B1	B2	B3
			B4	B5	
16-bit data transfer instruction	MOVW	rp1, #word	0 1 1 0 0 Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub>	+ Low Byte +	+ High Byte +
		saddrp, #word	0 0 0 0 1 1 0 0	+ Saddr-offset +	+ Low Byte +
			+ High Byte +		
		sfrp, #word	0 0 0 0 1 0 1 1	+ Sfr-offset +	+ Low Byte +
			+ High Byte +		
		rp, rp1	0 0 1 0 0 1 0 0	P <sub>2</sub> P <sub>1</sub> P <sub>0</sub> 0 1 Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub>	
		AX, saddrp	0 0 0 1 1 1 0 0	+ Saddr-offset +	
		saddrp, AX	0 0 0 1 1 0 1 0	+ Saddr-offset +	
		saddrp, saddrp	0 0 1 1 1 1 0 0	+ Saddr-offset +	+ Saddr-offset +
		AX, sfrp	0 0 0 1 0 0 0 1	+ Sfr-offset +	
		sfrp, AX	0 0 0 1 0 0 1 1	+ Sfr-offset +	
		rp1, !addr16	0 0 0 0 1 0 0 1	1 0 0 0 0 Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub>	+ Low Addr. +
	+ High Addr. +				
	!addr16, rp1	0 0 0 0 1 0 0 1	1 0 0 1 0 Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub>	+ Low Addr. +	
		+ High Addr. +			
	XCHW	AX, saddrp	0 0 0 1 1 0 1 1	+ Saddr-offset +	
AX, sfrp		0 0 0 0 0 0 0 1	0 0 0 1 1 0 1 1	+ Sfr-offset +	
saddrp, saddrp		0 0 1 0 1 0 1 0	+ Saddr-offset +	+ Saddr-offset +	
rp, rp1		0 0 1 0 0 1 0 1	P <sub>2</sub> P <sub>1</sub> P <sub>0</sub> 0 1 Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub>		

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	OP Code		
			B1	B2	B3
			B4	B5	
8-bit operation instruction	ADD	A, #byte	1 0 1 0 1 0 0 0	+ Data +	
		saddr, #byte	0 1 1 0 1 0 0 0	+ Saddr-offset +	+ Data +
		sfr, #byte	0 0 0 0 0 0 0 1	0 1 1 0 1 0 0 0	+ Sfr-offset +
			+ Data +		
		r, r1	1 0 0 0 1 0 0 0	R <sub>3</sub> R <sub>2</sub> R <sub>1</sub> R <sub>0</sub> 0 R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	
		A, saddr	1 0 0 1 1 0 0 0	+ Saddr-offset +	
		A, sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 0 0 0	+ Sfr-offset +
		saddr, saddr	0 1 1 1 1 0 0 0	+ Saddr-offset +	+ Saddr-offset +
		A, mem	0 0 0 mod	0 mem 1 0 0 0	+ Low Offset +
	+ High Offset +				
	mem, A	0 0 0 mod	1 mem 1 0 0 0	+ Low Offset +	
		+ High Offset +			
	ADDC	A, #byte	1 0 1 0 1 0 0 1	+ Data +	
		saddr, #byte	0 1 1 0 1 0 0 1	+ Saddr-offset +	+ Data +
		sfr, #byte	0 0 0 0 0 0 0 1	0 1 1 0 1 0 0 1	+ Sfr-offset +
			+ Data +		
		r, r1	1 0 0 0 1 0 0 1	R <sub>3</sub> R <sub>2</sub> R <sub>1</sub> R <sub>0</sub> 0 R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	
		A, saddr	1 0 0 1 1 0 0 1	+ Saddr-offset +	
A, sfr		0 0 0 0 0 0 0 1	1 0 0 1 1 0 0 1	+ Sfr-offset +	
saddr, saddr		0 1 1 1 1 0 0 1	+ Saddr-offset +	+ Saddr-offset +	

(to be continued)



(cont'd)

Instruction Group	Mnemonic	Operand	OP Code		
			B1	B2	B3
			B4	B5	
8-bit operation instruction (cont'd)	ADDC	A, mem	0 0 0 mod	0 mem 1 0 0 1	+ Low Offset +
			+ High Offset +		
		mem, A	0 0 0 mod	1 mem 1 0 0 1	+ Low Offset +
			+ High Offset +		
	SUB	A, #byte	1 0 1 0 1 0 1 0	+ Data +	
		saddr, #byte	0 1 1 0 1 0 1 0	+ Saddr-offset +	+ Data +
		sfr, #byte	0 0 0 0 0 0 0 1	0 1 1 0 1 0 1 0	+ Sfr-offset +
			+ Data +		
		r, r1	1 0 0 0 1 0 1 0	R <sub>3</sub> R <sub>2</sub> R <sub>1</sub> R <sub>0</sub> 0 R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	
		A, saddr	1 0 0 1 1 0 1 0	+ Saddr-offset +	
		A, sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 0 1 0	+ Sfr-offset +
		saddr, saddr	0 1 1 1 1 0 1 0	+ Saddr-offset +	+ Saddr-offset +
		A, mem	0 0 0 mod	0 mem 1 0 1 0	+ Low Offset +
			+ High Offset +		
		mem, A	0 0 0 mode	1 mem 1 0 1 0	+ Low Offset +
			+ High Offset +		
	SUBC	A, #byte	1 0 1 0 1 0 1 1	+ Data +	
		saddr, #byte	0 1 1 0 1 0 1 1	+ Saddr-offset +	+ Data +
		sfr, #byte	0 0 0 0 0 0 0 1	0 1 1 0 1 0 1 1	+ Sfr-offset +
			+ Data +		
r, r1		1 0 0 0 1 0 1 1	R <sub>3</sub> R <sub>2</sub> R <sub>1</sub> R <sub>0</sub> 0 R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>		

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	OP Code		
			B1	B2	B3
			B4	B5	
8-bit operation instruction (cont'd)	SUBC	A, saddr	1 0 0 1 1 0 1 1	+ Saddr-offset →	
		A, sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 0 1 1	+ Sfr-offset →
		saddr, saddr	0 1 1 1 1 0 1 1	+ Saddr-offset →	+ Saddr-offset →
		A, mem	0 0 0 mod	0 mem 1 0 1 1	+ Low Offset →
			+ High Offset →		
		mem, A	0 0 0 mod	1 mem 1 0 1 1	+ Low Offset →
	+ High Offset →				
	AND	A, #byte	1 0 1 0 1 1 0 0	+ Data →	
		saddr, #byte	0 1 1 0 1 1 0 0	+ Saddr-offset →	+ Data →
		sfr, #byte	0 0 0 0 0 0 0 1	0 1 1 0 1 1 0 0	+ Sfr-offset →
			+ Data →		
		r, r1	1 0 0 0 1 1 0 0	R <sub>3</sub> R <sub>2</sub> R <sub>1</sub> R <sub>0</sub> 0 R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	
		A, saddr	1 0 0 1 1 1 0 0	+ Saddr-offset →	
		A, sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 1 0 0	+ Sfr-offset →
		saddr, saddr	0 1 1 1 1 1 0 0	+ Saddr-offset →	+ Saddr-offset →
		A, mem	0 0 0 mod	0 mem 1 1 0 0	+ Low Offset →
			+ High Offset →		
		mem, A	0 0 0 mod	1 mem 1 1 0 0	+ Low Offset →
+ High Offset →					

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	OP Code		
			B1	B2	B3
			B4	B5	
8-bit operation instruction (cont'd)	OR	A, #byte	1 0 1 0 1 1 1 0	+ Data +	
		saddr, #byte	0 1 1 0 1 1 1 0	+ Saddr-offset +	+ Data +
		sfr, #byte	0 0 0 0 0 0 0 1	0 1 1 0 1 1 1 0	+ Sfr-offset +
			+ Data +		
		r, r1	1 0 0 0 1 1 1 0	R <sub>3</sub> R <sub>2</sub> R <sub>1</sub> R <sub>0</sub> 0 R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	
		A, saddr	1 0 0 1 1 1 1 0	+ Saddr-offset +	
		A, sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 1 1 0	+ Sfr-offset +
		saddr, saddr	0 1 1 1 1 1 1 0	+ Saddr-offset +	+ Saddr-offset +
		A, mem	0 0 0 mod	0 mem 1 1 1 0	+ Low Offset +
			+ High Offset +		
	mem, A	0 0 0 mod	1 mem 1 1 1 0	+ Low Offset +	
		+ High Offset +			
	XOR	A, #byte	1 0 1 0 1 1 0 1	+ Data +	
		saddr, #byte	0 1 1 0 1 1 0 1	+ Saddr-offset +	+ Data +
		sfr, #byte	0 0 0 0 0 0 0 1	0 1 1 0 1 1 0 1	+ Sfr-offset +
			+ Data +		
		r, r1	1 0 0 0 1 1 0 1	R <sub>3</sub> R <sub>2</sub> R <sub>1</sub> R <sub>0</sub> 0 R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	
		A, saddr	1 0 0 1 1 1 0 1	+ Saddr-offset +	
A, sfr		0 0 0 0 0 0 0 1	1 0 0 1 1 1 0 1	+ Sfr-offset +	
saddr, saddr		0 1 1 1 1 1 0 1	+ Saddr-offset +	+ Saddr-offset +	

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	OP Code		
			B1	B2	B3
			B4	B5	
8-bit operation instruction (cont'd)	XOR	A, mem	0 0 0 mod	0 mem 1 1 0 1	+ Low Offset →
			← High Offset →		
		mem, A	0 0 0 mod	1 mem 1 1 0 1	+ Low Offset →
			← High Offset →		
	CMP	A, #byte	1 0 1 0 1 1 1 1	+ Data →	
		saddr, #byte	0 1 1 0 1 1 1 1	+ Saddr-offset →	+ Data →
		sfr, #byte	0 0 0 0 0 0 0 1	0 1 1 0 1 1 1 1	+ Sfr-offset →
			← Data →		
		r, r1	1 0 0 0 1 1 1 1	R <sub>3</sub> R <sub>2</sub> R <sub>1</sub> R <sub>0</sub> 0 R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	
		A, saddr	1 0 0 1 1 1 1 1	+ Saddr-offset →	
		A, sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 1 1 1	+ Sfr-offset →
		saddr, saddr	0 1 1 1 1 1 1 1	+ Saddr-offset →	+ Saddr-offset →
		A, mem	0 0 0 mod	0 mem 1 1 1 1	+ Low Offset →
			← High Offset →		
mem, A	0 0 0 mod	1 mem 1 1 1 1	+ Low Offset →		
	← High Offset →				
16-bit operation instruction	ADDW	AX, #word	0 0 1 0 1 1 0 1	+ Low Byte →	+ High Byte →
		saddrp, #word	0 0 0 0 1 1 0 1	+ Saddr-offset →	+ Low Byte →
			← High Offset →		
		sfrp, #word	0 0 0 0 0 0 0 1	0 0 0 0 1 1 0 1	+ Sfr-offset →
← Low Byte →			← High Byte →		

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	OP Code		
			B1	B2	B3
			B4	B5	
16-bit operation instruction (cont'd)	ADDW	rp, rp1	1 0 0 0 1 0 0 0	P <sub>2</sub> P <sub>1</sub> P <sub>0</sub> 0 1 Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub>	
		AX, saddrp	0 0 0 1 1 1 0 1	+ Saddr-offset +	
		AX, sfrp	0 0 0 0 0 0 0 1	0 0 0 1 1 1 0 1	+ Sfr-offset +
		saddrp, saddrp	0 0 1 1 1 1 0 1	+ Saddr-offset +	+ Saddr-offset +
	SUBW	AX, #word	0 0 1 0 1 1 1 0	+ Low Byte +	+ High Byte +
		saddrp, #word	0 0 0 0 1 1 1 0	+ Saddr-offset +	+ Low Byte +
			+ High Byte +		
		sfrp, #word	0 0 0 0 0 0 0 1	0 0 0 0 1 1 1 0	+ Sfr-offset +
			+ Low Byte +	+ High Byte +	
		rp, rp1	1 0 0 0 1 0 1 0	P <sub>2</sub> P <sub>1</sub> P <sub>0</sub> 0 1 Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub>	
		AX, saddrp	0 0 0 1 1 1 1 0	+ Saddr-offset +	
		AX, sfrp	0 0 0 0 0 0 0 1	0 0 0 1 1 1 1 0	+ Sfr-offset +
	saddrp, saddrp	0 0 1 1 1 1 1 0	+ Saddr-offset +	+ Saddr-offset +	
	CMPW	AX, #word	0 0 1 0 1 1 1 1	+ Low Byte +	+ High Byte +
		saddrp, #word	0 0 0 0 1 1 1 1	+ Saddr-offset +	+ Low Byte +
			+ High Byte +		
		sfrp, #word	0 0 0 0 0 0 0 1	0 0 0 0 1 1 1 1	+ Sfr-offset +
			+ Low Byte +	+ High Byte +	
		rp, rp1	1 0 0 0 1 1 1 1	P <sub>2</sub> P <sub>1</sub> P <sub>0</sub> 0 1 Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub>	

(to be continued)

Instruction Group	Mnemonic	Operand	OP Code		
			B1	B2	B3
			B4	B5	
16-bit operation instruction (cont'd)	CMPW	AX, saddrp	0 0 0 1 1 1 1 1	+ Saddr-offset +	
		AX, sfrp	0 0 0 0 0 0 0 1	0 0 0 1 1 1 1 1	+ Sfr-offset +
		saddrp, saddrp	0 0 1 1 1 1 1 1	+ Saddr-offset +	+ Saddr-offset +
Multiply/divide instruction	MULU	r1	0 0 0 0 0 1 0 1	0 0 0 0 1 R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	
	DIVUW	r1		0 0 0 1 1 R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	
	MULUW	rp1		0 0 1 0 1 Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub>	
	DIVUX	rp1		1 1 1 0 1 Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub>	
Increment/decrement instruction	INC	r1	1 1 0 0 0 R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>		
		saddr	0 0 1 0 0 1 1 0	+ Saddr-offset +	
	DEC	r1	1 1 0 0 1 R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>		
		saddr	0 0 1 0 0 1 1 1	+ Saddr-offset +	
	INCW	rp2	0 1 0 0 0 1 S <sub>1</sub> S <sub>0</sub>		
		saddrp	0 0 0 0 0 1 1 1	1 1 1 0 1 0 0 0	+ Saddr-offset +
DECW	rp2	0 1 0 0 1 1 S <sub>1</sub> S <sub>0</sub>			
	saddrp	0 0 0 0 0 1 1 1	1 1 1 0 1 0 0 1	+ Saddr-offset +	
Shift/rotate instruction	ROR	r1, n	0 0 1 1 0 0 0 0	0 1 N <sub>2</sub> N <sub>1</sub> N <sub>0</sub> R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	
	ROL	r1, n		0 0 0 1 0 1 N <sub>2</sub> N <sub>1</sub> N <sub>0</sub> R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	
	RORC	r1, n		0 0 0 0 0 0 N <sub>2</sub> N <sub>1</sub> N <sub>0</sub> R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	
	ROLC	r1, n		0 0 0 1 0 0 N <sub>2</sub> N <sub>1</sub> N <sub>0</sub> R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	
	SHR	r1, n		0 0 0 0 1 0 N <sub>2</sub> N <sub>1</sub> N <sub>0</sub> R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	
	SHL	r1, n		0 0 0 1 1 0 N <sub>2</sub> N <sub>1</sub> N <sub>0</sub> R <sub>2</sub> R <sub>1</sub> R <sub>0</sub>	

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	OP Code		
			B1	B2	B3
			B4	B5	
Shift/rotate instruction (cont'd)	SHRW	rp1, n	0 0 1 1 0 0 0 0	1 1 N <sub>2</sub> N <sub>1</sub> N <sub>0</sub> Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub>	
	SHLW	rp1, n	0 0 1 1 0 0 0 1	1 1 N <sub>2</sub> N <sub>1</sub> N <sub>0</sub> Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub>	
	ROR4	[rp1]	0 0 0 0 0 1 0 1	1 0 0 0 1 Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub>	
	ROL4	[rp1]	0 0 0 0 0 1 0 1	1 0 0 1 1 Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub>	
BCD adjustment instruction	ADJ4		0 0 0 0 1 0 0 0		
Bit manipulation instruction	MOV1	CY, saddr.bit	0 0 0 0 1 0 0 0	0 0 0 0 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ Saddr-offset +
		CY, sfr.bit	1 0 0 0	1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ Sfr-offset +
		CY, A.bit	0 0 1 1	1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		CY, X.bit	0 0 1 1	0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		CY, PSWH.bit	0 0 1 0	1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		CY, PSWL.bit	0 0 1 0	0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		saddr.bit, CY	1 0 0 0	0 0 0 1 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ Saddr-offset +
		sfr.bit, CY	1 0 0 0	1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ Sfr-offset +
		A.bit, CY	0 0 1 1	1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		X.bit, CY	0 0 1 1	0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	OP Code		
			B1	B2	B3
			B4	B5	
Bit manipulation instruction (cont'd)	MOV1	PSWH. bit, CY	0 0 0 0 0 0 1 0	0 0 0 1 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		PSWL. bit, CY	0 0 0 0 0 0 1 0	0 0 0 1 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
	AND1	CY, saddr. bit	0 0 0 0 1 0 0 0	0 0 1 0 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ Saddr-offset +
		CY, /saddr. bit		0 0 1 1 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ Saddr-offset +
		CY, sfr. bit		0 0 1 0 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ Sfr-offset +
		CY, /sfr. bit		0 0 1 1 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ Sfr-offset +
		CY, A. bit		0 0 1 1 0 0 1 0 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		CY, /A. bit		0 0 1 1 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		CY, X. bit		0 0 1 0 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		CY, /X. bit		0 0 1 1 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		CY, PSWH. bit		0 0 1 0 0 0 1 0 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		CY, /PSWH. bit		0 0 1 1 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		CY, PSWL. bit		0 0 1 0 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		CY, /PSWL. bit		0 0 1 1 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	

(to be continued)



(cont'd)

Instruction Group	Mnemonic	Operand	OP Code		
			B1	B2	B3
			B4	B5	
Bit manipulation instruction (cont'd)	XOR1	CY, A.bit	0 0 0 0 0 0 1 1	0 1 1 0 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		CY, X.bit	0 0 1 1	0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		CY, PSWH.bit	0 0 1 0	1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		CY, PSWL.bit	0 0 1 0	0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
	SET1	saddr. bit	1 0 1 1 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	← Saddr-offset →	
		sfr.bit	0 0 0 0 1 0 0 0	1 0 0 0 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	← Sfr-offset →
		A.bit	0 0 1 1	1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		X.bit	0 0 1 1	0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		PSWH.bit	0 0 1 0	1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		PSWL.bit	0 0 1 0	0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
	CLR1	saddr. bit	1 0 1 0 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	← Saddr-offset →	
		sfr.bit	0 0 0 0 1 0 0 0	1 0 0 1 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	← Sfr-offset →
		A.bit	0 0 1 1	1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		X.bit	0 0 1 1	0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		PSWH.bit	0 0 1 0	1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
		PSWL.bit	0 0 1 0	0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	
	NOT1	saddr. bit	0 0 0 0 1 0 0 0	0 1 1 1 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	← Saddr-offset →
		sfr.bit	1 0 0 0	1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	← Sfr-offset →
		A.bit	0 0 1 1	1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	OP Code			
			B1	B2	B3	
			B4	B5		
Bit manipulation instruction (cont'd)	OR1	CY, saddr. bit	0 0 0 0 1 0 0 0	0 1 0 0 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ Saddr-offset +	
		CY, /saddr. bit		0 1 0 1 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ Saddr-offset +	
		CY, sfr. bit		0 1 0 0 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ Sfr-offset +	
		CY, /sfr. bit		0 1 0 1 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ Sfr-offset +	
		CY, A. bit	0 0 1 1	0 1 0 0 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>		
		CY, /A. bit		0 1 0 1 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>		
		CY, X. bit		0 1 0 0 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>		
		CY, /X. bit		0 1 0 1 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>		
		CY, PSWH. bit	0 0 1 0	0 1 0 0 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>		
		CY, /PSWH. bit		0 1 0 1 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>		
		CY, PSWL. bit		0 1 0 0 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>		
		CY, /PSWL. bit		0 1 0 1 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>		
		XOR1	CY, saddr. bit	0 0 0 0 1 0 0 0	0 1 1 0 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ Saddr-offset +
				1 0 0 0	1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ Sfr-offset +

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	OP Code																			
			B1		B2		B3															
			B4		B5																	
Bit manipulation instruction (cont'd)	NOT1	X.bit	0	0	0	0	0	0	1	1	0	1	1	1	0	B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>						
		PSWH.bit			0	0	1	0			1	B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>										
		PSWL.bit			0	0	1	0			0	B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>										
	SET1	CY	0	1	0	0	0	0	0	0	1											
	CLR1	CY			0	0	0	0														
	NOT1	CY			0	0	1	0														
Call/return instruction	CALL	!addr16	0	0	1	0	1	0	0	0		+	Low Addr.	+		+	High Addr.	+				
	CALLF	!addr11	1	0	0	1	0	+			fa			+								
	CALLT	[addr5]	1	1	1	+		ta		+												
	CALL	rp1	0	0	0	0	0	1	0	1	0	1	0	1	1	Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub>						
		[rp1]	0	0	0	0	0	1	0	1	0	1	1	1	1	Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub>						
	BRK		0	1	0	1	1	1	1	0												
	RET		0	1	0	1	0	1	1	0												
	RETI		0	1	0	1	0	1	1	1												
Stack operation instruction	PUSH	post	0	0	1	1	0	1	0	1		+	Post Byte	+								
		PSW	0	1	0	0	1	0	0	1												
	PUSHU	post	0	0	1	1	0	1	1	1		+	Post Byte	+								
	POP	post	0	0	1	1	0	1	0	0		+	Post Byte	+								
		PSW	0	1	0	0	1	0	0	0												
	POPU	post	0	0	1	1	0	1	1	0		+	Post Byte	+								
	MOVW	SP, #word		0	0	0	0	1	0	1	1	1	1	1	1	1	1	0	0	+	Low Byte	+
													+	High Byte	+							
SP, AX		0	0	0	1	0	0	1	1	1	1	1	1	1	1	0	0					

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	OP Code		
			B1	B2	B3
			B4	B5	
Stack operation instruction (cont'd)	MOVW	AX, SP	0 0 0 1 0 0 0 1	1 1 1 1 1 1 0 0	
	INCW	SP	0 0 0 0 0 1 0 1	1 1 0 0 1 0 0 0	
	DFCW	SP	0 0 0 0 0 1 0 1	1 1 0 0 1 0 0 1	
Unconditional branch instruction	BR	!addr16	0 0 1 0 1 1 0 0	← Low Addr. →	+ High Addr. →
		rp1	0 0 0 0 0 1 0 1	0 1 0 0 1 Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub>	
		[rp1]	0 0 0 0 0 1 0 1	0 1 1 0 1 Q <sub>2</sub> Q <sub>1</sub> Q <sub>0</sub>	
		\$addr16	0 0 0 1 0 1 0 0	+ jdisp →	
Conditional branch instruction	BC	\$addr16	1 0 0 0 0 0 1 1	+ jdisp →	
	BL				
	BNC	\$addr16	0 0 1 0	+ jdisp →	
	BNL				
	BZ	\$addr16	0 0 0 1	+ jdisp →	
	BE				
	BNZ	\$addr16	0 0 0 0	+ jdisp →	
	BNE				
	BV	\$addr16	0 1 0 1	+ jdisp →	
	BPE				
	BNV	\$addr16	0 1 0 0	+ jdisp →	
	BPO				
	BN	\$addr16	0 1 1 1	+ jdisp →	
BP	\$addr16	0 1 1 0	+ jdisp →		
BGT	\$addr16	0 0 0 0 0 1 1 1	1 1 1 1 1 0 1 1	+ jdisp →	

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	OP Code																
			B1		B2		B3												
			B4		B5														
Conditional branch instruction (cont'd)	BGE	\$addr16	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	1	+ jdisp +
	BLT	\$addr16													1	0	0	0	+ jdisp +
	BLE	\$addr16													1	0	1	0	+ jdisp +
	BH	\$addr16													1	1	0	1	+ jdisp +
	BNH	\$addr16													1	1	0	0	+ jdisp +
	BT	saddr. bit, \$addr16	0	1	1	1	0	B <sub>2</sub>	B <sub>1</sub>	B <sub>0</sub>	+ Saddr-offset +	+ jdisp +							
		sfr.bit, \$addr16	0	0	0	0	1	0	0	0	1	0	1	1	1	B <sub>2</sub>	B <sub>1</sub>	B <sub>0</sub>	+ Sfr-offset +
			+ jdisp +																
		A.bit, \$addr16	0	0	0	0	0	0	1	1	1	0	1	1	1	B <sub>2</sub>	B <sub>1</sub>	B <sub>0</sub>	+ jdisp +
		X.bit, \$addr16					0	0	1	1					0	B <sub>2</sub>	B <sub>1</sub>	B <sub>0</sub>	+ jdisp +
		PSWH. bit, \$addr16					0	0	1	0					1	B <sub>2</sub>	B <sub>1</sub>	B <sub>0</sub>	+ jdisp +
		PSWL. bit, \$addr16					0	0	1	0					0	B <sub>2</sub>	B <sub>1</sub>	B <sub>0</sub>	+ jdisp +
	BF	saddr. bit, \$addr16	0	0	0	0	1	0	0	0	1	0	1	0	0	B <sub>2</sub>	B <sub>1</sub>	B <sub>0</sub>	+ Saddr-offset +
			+ jdisp +																
		sfr.bit, \$addr16	0	0	0	0	1	0	0	0	1	0	1	0	1	B <sub>2</sub>	B <sub>1</sub>	B <sub>0</sub>	+ Sfr-offset +
+ jdisp +																			
A.bit, \$addr16		0	0	0	0	0	0	1	1	1	0	1	0	1	B <sub>2</sub>	B <sub>1</sub>	B <sub>0</sub>	+ jdisp +	
X.bit, \$addr16					0	0	1	1					0	B <sub>2</sub>	B <sub>1</sub>	B <sub>0</sub>	+ jdisp +		

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	OP Code			
			B1	B2	B3	
			B4	B5		
Conditional branch instruction (cont'd)	BF	PSWH. bit, \$addr16	0 0 0 0 0 0 1 0	1 0 1 0 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ jdisp →	
		PSWL. bit, \$addr16	0 0 0 0 0 0 1 0	1 0 1 0 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ jdisp →	
	BTCLR	saddr. bit, \$addr16	0 0 0 0 1 0 0 0	1 1 0 1 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ Saddr-offset →	
			+ jdisp →			
		sfr.bit, \$addr16	0 0 0 0 1 0 0 0	1 1 0 1 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ Sfr-offset →	
			+ jdisp →			
		A.bit, \$addr16	0 0 0 0 0 0 1 1	1 1 0 1 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ jdisp →	
		X.bit, \$addr16		0 0 1 1	0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ jdisp →
		PSWH. bit, \$addr16		0 0 1 0	1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ jdisp →
		PSWL. bit, \$addr16		0 0 1 0	0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ jdisp →
	BFSET	saddr. bit, \$addr16	0 0 0 0 1 0 0 0	1 1 0 0 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ Saddr-offset →	
			+ jdisp →			
		sfr.bit, \$addr16	0 0 0 0 1 0 0 0	1 1 0 0 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ Sfr-offset →	
			+ jdisp →			
A.bit, \$addr16		0 0 0 0 0 0 1 1	1 1 0 0 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ jdisp →		
X.bit, \$addr16		0 0 0 0 0 0 1 1	1 1 0 0 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ jdisp →		

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	OP Code		
			B1	B2	B3
			B4	B5	
Conditional branch instruction (cont'd)	BFSET	PSWH. bit, \$addr16	0 0 0 0 0 0 1 0	1 1 0 0 1 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ jdisp +
		PSWL. bit, \$addr16	0 0 0 0 0 0 1 0	1 1 0 0 0 B <sub>2</sub> B <sub>1</sub> B <sub>0</sub>	+ jdisp +
	DBNZ	r2, \$addr16	0 0 1 1 0 0 1 C <sub>0</sub>		+ jdisp +
		saddr, \$addr16	0 0 1 1 1 0 1 1	+ saddr-offset +	+ jdisp +
Context switching instruction	BRKCS	R <sub>Bn</sub>	0 0 0 0 0 1 0 1	1 1 0 1 1 N <sub>2</sub> N <sub>1</sub> N <sub>0</sub>	
	RETCS	!addr16	0 0 1 0 1 0 0 1	+ Low Addr. +	+ High Addr. +
String instruction	MOVM	[DE+], A	0 0 0 1 0 1 0 1	0 0 0 0 0 0 0 0	
		[DE-], A		0 0 0 1	
	MOVBK	[DE+], [HL+]		0 0 1 0	
		[DE-], [HL-]		0 0 1 1	
	XCHM	[DE+], A		0 0 0 0 0 0 0 1	
		[DE-], A		0 0 0 1	
	XCHBK	[DE+], [HL+]		0 0 1 0	
		[DE-], [HL-]		0 0 1 1	
	CMPME	[DE+], A		0 0 0 0 0 1 0 0	
		[DE-], A		0 0 0 1 0 1 0 0	

(to be continued)

(cont'd)

Instruction Group	Mnemonic	Operand	OP Code			
			B1	B2	B3	
			B4	B5		
String instruction (cont'd)	CMP BKE	[DE+], [HL+]	0 0 0 1 0 1 0 1	0 0 1 0 0 1 0 0		
		[DE-], [HL-]		0 0 1 1 0 1 0 0		
	CMP MNE	[DE+], A		0 0 0 0 0 1 0 1		
		[DE-], A		0 0 0 1		
	CMPB KNE	[DE+], [HL+]		0 0 1 0		
		[DE-], [HL-]		0 0 1 1		
	CMPMC	[DE+], A		0 0 0 0 0 1 1 1		
		[DE-], A		0 0 0 1		
	CMP BKC	[DE+], [HL+]		0 0 1 0		
		[DE-], [HL-]		0 0 1 1		
	CMP MNC	[DE+], A		0 0 0 0 0 1 1 0		
		[DE-], A		0 0 0 1		
	CMPB KNC	[DE+], [HL+]		0 0 1 0		
		[DE-], [HL-]		0 0 1 1		
	CPU control instruction	MOV	STBC, #byte	0 0 0 0 1 0 0 1	0 1 0 0 0 1 0 0	+ Data +
				+ Data +		
WDM, #byte			0 0 0 0 1 0 0 1	0 1 0 0 0 0 1 0	+ Data +	
			+ Data +			
SWRS		0 1 0 0 0 0 1 1				

(to be continued)



(cont'd)

Instruction Group	Mnemonic	Operand	OP Code		
			B1	B2	B3
			B4	B5	
CPU control instruction	SEL	RBn	0 0 0 0 0 1 0 1	1 0 1 0 1 N <sub>2</sub> N <sub>1</sub> N <sub>0</sub>	
		RBn, ALT	0 0 0 0 0 1 0 1	1 0 1 1 1 N <sub>2</sub> N <sub>1</sub> N <sub>0</sub>	
	NOP		0 0 0 0 0 0 0 0		
	EI		0 1 0 0 1 0 1 1		
	DI		0 1 0 0 1 0 1 0		

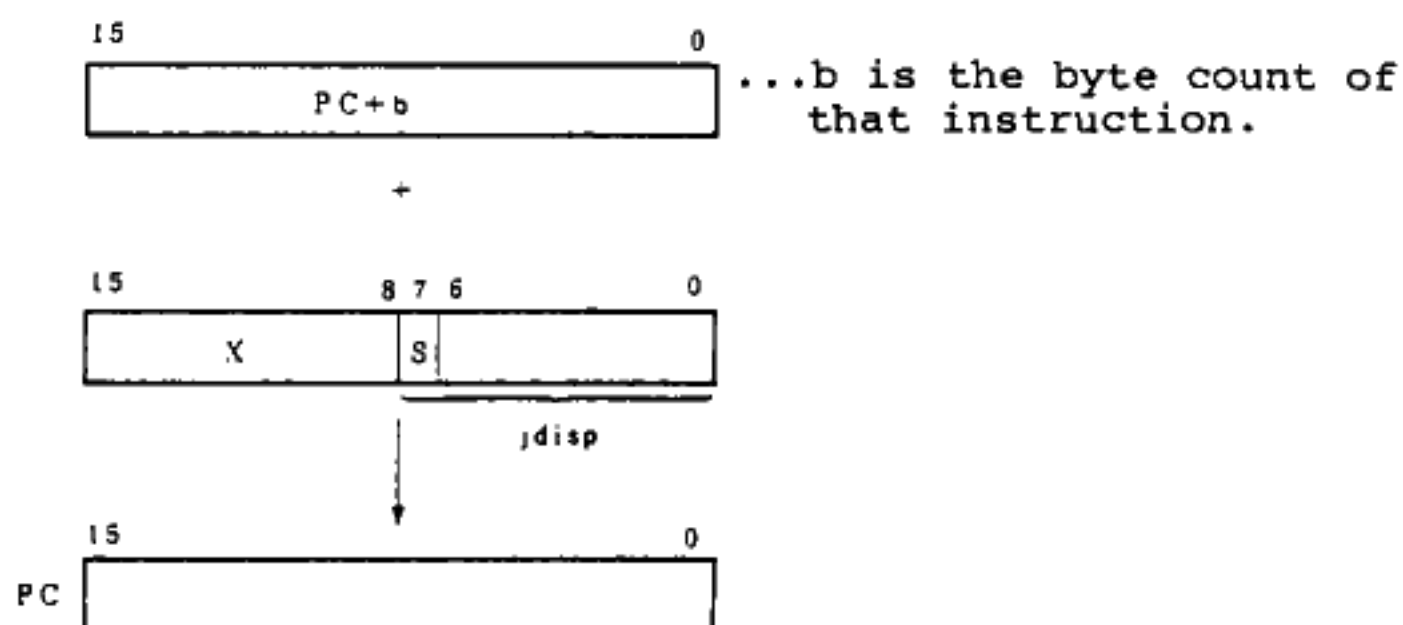
## 10.4 INSTRUCTION ADDRESS ADDRESSING

The contents of the program counter (PC) determine the instruction address and the address is usually incremented (by one for one byte) automatically according to the byte count of the instruction fetched each time an instruction is executed. However, when an instruction with a branch is executed, the branch destination address information is set in the PC and the program is branched by the addressing described below.

### 10.4.1 RELATIVE ADDRESSING

The sum of the 8-bit immediate data (displacement:  $jdisp$ ) of the OP code and the start address of the next instruction is transferred to the program counter (PC) and the branch is executed. The displacement is treated as two's complement data (-128 to +127). Bit 7 is the sign bit.

Relative addressing is performed when executing a BR  $\$addr16$  instruction or conditional branch instruction.



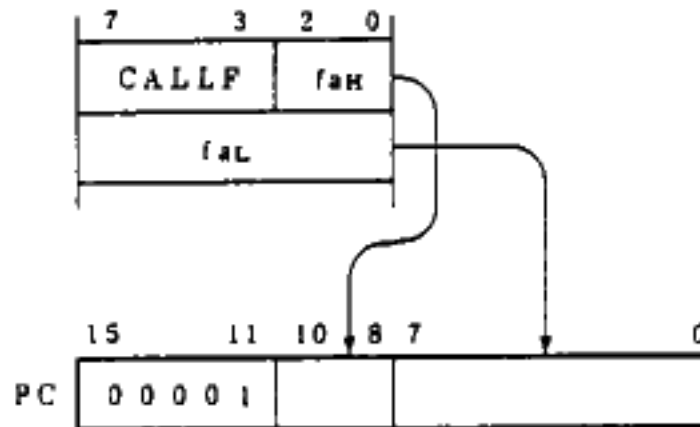
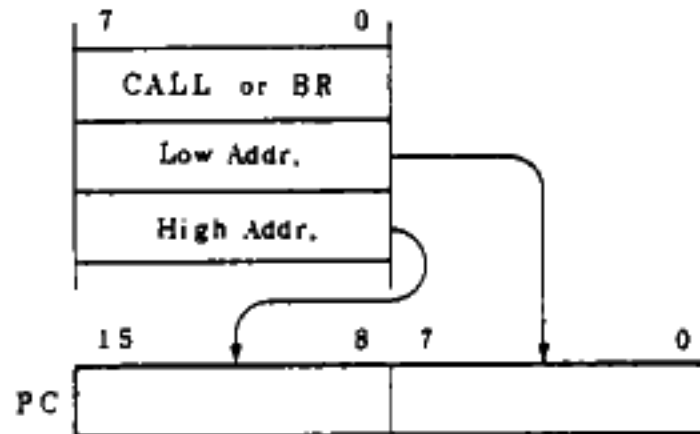
- When S=0, X is all bits 0.
- When S=1, X is all bits 1.

### 10.4.2 IMMEDIATE ADDRESSING

The immediate data in the instruction word is transferred to the program counter (PC) and the branch is executed.

Immediate addressing is performed when executing a CALL !addr16, BR !addr16, or CALLF !addr11 instruction.

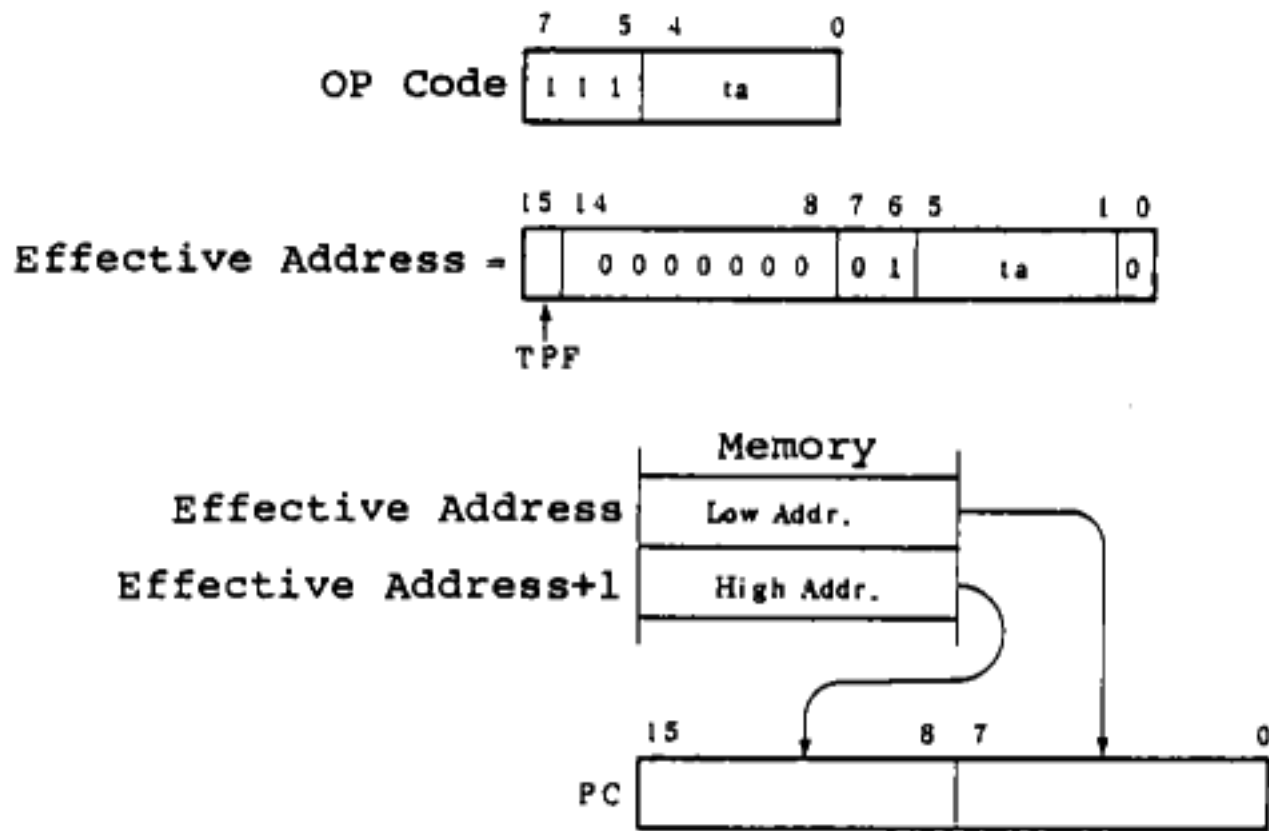
For a CALLF !addr11 instruction, the program branches to the fixed address determined by the high-order 5 bits address.



### 10.4.3 TABLE INDIRECT ADDRESSING

The contents (branch destination address) of the table of the specific location addressed by the immediate data of the low-order 5 bits of the OP code are transferred to the program counter (PC) and the branch is executed.

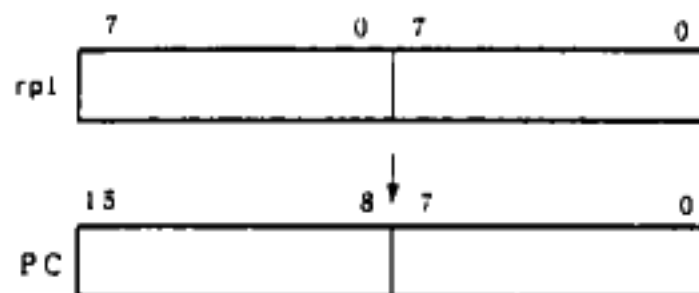
Table indirect addressing is performed when executing a CALLT [addr5] instruction.



### 10.4.4 REGISTER ADDRESSING

The contents of the register pair (RP7 to RP0) specified by the instruction word are transferred to the program counter (PC) and the branch is executed.

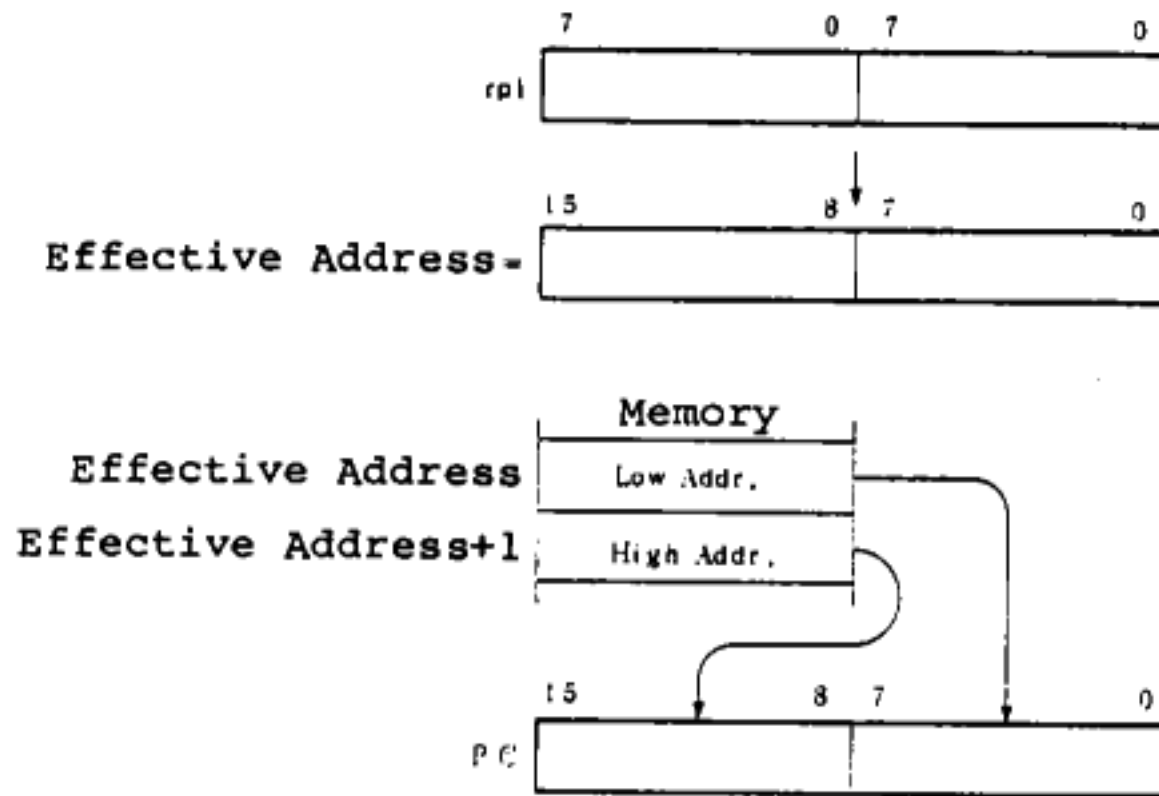
Register addressing is performed when executing a BR rpl or CALL rpl instruction.



#### 10.4.5 REGISTER INDIRECT ADDRESSING

The continuous 2 byte data of the memory addressed by the contents of the register pair (RP7 to RP0) specified by the instruction word are transferred to the program counter (PC) and the branch is executed.

Register indirect addressing is performed when executing a BR [rpl] or CALL [rpl] instruction.



## 10.5 OPERAND ADDRESS ADDRESSING

When executing an instruction, the objective registers and memory can be addressed by the methods described below.

### 10.5.1 REGISTER ADDRESSING

This addressing accesses the general register specified by the register set select flag (RSS) in the register bank specified by the register bank select flag (RBS2 to RBS0) and the register select code (Rn, Pn, Qn) in the instruction word as the operand.

Register addressing is performed when executing an instruction having the operand format shown below. When addressing an 8-bit register, 8 is specified by 3 bits in the OP code and 16 is specified by 4 bits. When addressing a 16-bit register pair, 8 are specified by 3 bits in the OP code.

Identifier	Description
r	R0, R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15
r1	R0, R1, R2, R3, R4, R5, R6, R7
r2	C, B
rp	RP0, RP1, RP2, RP3, RP4, RP5, RP6, RP7
rpl	RP0, RP1, RP2, RP3, RP4, RP5, RP6, RP7
rp2	DE, HL, VP, UP

r, r1, rp, and rpl can be described by function name (X, A, C, B, E, D, L, H, AX, BC, DE, HL, VP, UP) as well as absolute name (R0 to R15, RP0 to RP7). The function name corresponding to each absolute name is shown in Table 10-1 and Table 10-2.

Example 1: MOV A, r1

OP code  $\boxed{1\ 1\ 0\ 1\ '0\ R_2\ R_1\ R_0}$

When the R2 register is selected as r1, it is described as shown below. (When RSS = 0, the R2 register becomes the C register.)

MOV A, R2

The OP code for this is:

OP code  $\boxed{1\ 1\ 0\ 1\ '0\ 0\ 1\ 0}$

Example 2: INCW rp2

OP code  $\boxed{0\ 1\ 0\ 0\ '0\ 1\ S_1\ S_0}$

When the DE register pair is selected as rp2, it is described follows:

INCW DE

The OP code for this is:

OP code  $\boxed{0\ 1\ 0\ 0\ '0\ 1\ 1\ 0}$

## 10.5.2 IMMEDIATE ADDRESSING

This addressing has 8-bit data and 16-bit data as the operation objective in the OP code.

Immediate addressing is performed when executing an instruction having the operands shown below.

Identifier	Description
byte	Label, numerical value within 8 bits
word	Label, numerical value within 16 bits

Example 1: ADD A, #byte

OP code 

1	0	1	0	1	0	0	0
---	---	---	---	---	---	---	---

Data
------

When 77H is taken as byte, it is described as follows:

ADD A, #77H

The OP code for this is:

OP code 

1	0	1	0	1	0	0	0
---	---	---	---	---	---	---	---

0	1	1	1	0	1	1	1
---	---	---	---	---	---	---	---



### 10.5.3 DIRECT ADDRESSING

With this addressing, the immediate data in the instruction word becomes the operand address and the memory to be manipulated is accessed.

Direct addressing is performed when executing an instruction having the following operands:

Identifier	Description
addr16	Label, numerical value within 16 bits

Example 1: MOV A, laddr16

OP code 

0	0	0	0	1	0	0	1
---	---	---	---	---	---	---	---

1	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

Low	Addr.
-----	-------

High	Addr.
------	-------

When FE00H is taken as addr16, it is described as follows:

MOV A, l0FE00H

The OP code for this is:

OP code 

0	0	0	0	1	0	0	1
---	---	---	---	---	---	---	---

1	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

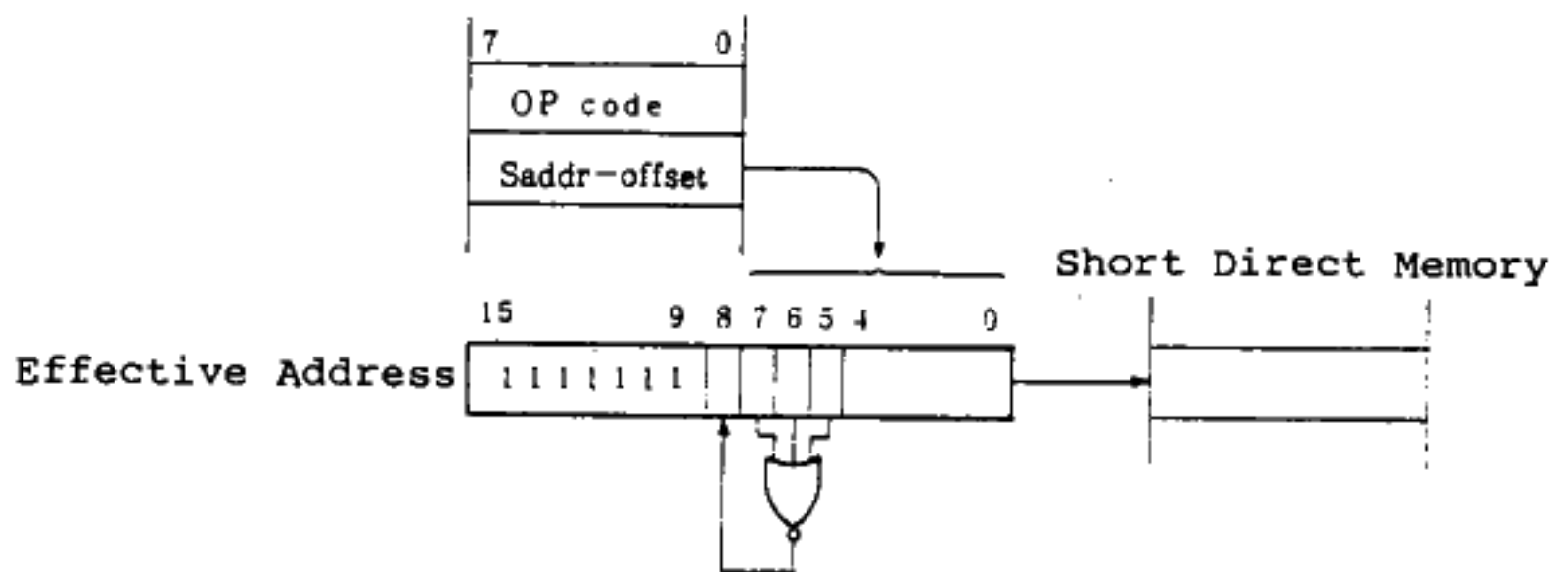
1	1	1	1	1	1	1	0
---	---	---	---	---	---	---	---

#### 10.5.4 SHORT DIRECT ADDRESSING

This addressing directly accesses the objective memory of the fixed space by the 8-bit immediate data in the instruction word.

The addressing is applicable to the FE20H to FF1FH 256-byte space. The internal RAM (short direct memory) is mapped to FE20H to FEFFH and the special function register (SFR) is mapped to FFO0H to FF1FH.

When the 8-bit immediate data is 20H to FFH, bit 8 of the effective address becomes 0 and when the 8-bit immediate data is 00H to 1FH, bit 8 of the effective address becomes 1.



This addressing is performed when executing instructions with `saddr`, `saddrp` in the operand. For an instruction with `saddrp`, the 2-byte data (data of even number minus odd number address, disregarding the least significant bit of the effective address) of the memory addressed by effective address and the memory of the next address is accessed.

Identifier	Description
<code>saddr</code>	Label, numerical value within FE20H to FF1FH
<code>saddrp</code>	Label, numerical value within FE20H to FF1EH (even number)

Example 1: MOV saddr, saddr

OP code 

0	0	1	1	1	0	0	0
---	---	---	---	---	---	---	---

Saddr-offset
--------------

Saddr-offset
--------------

When FE30H is taken as saddr of the first operand and FE50H is taken as saddr of the second operand, they are described as follows:

MOV OFE20H, OFE50H

The OP code for this is:

OP code 

0	0	1	1	1	0	0	0
---	---	---	---	---	---	---	---

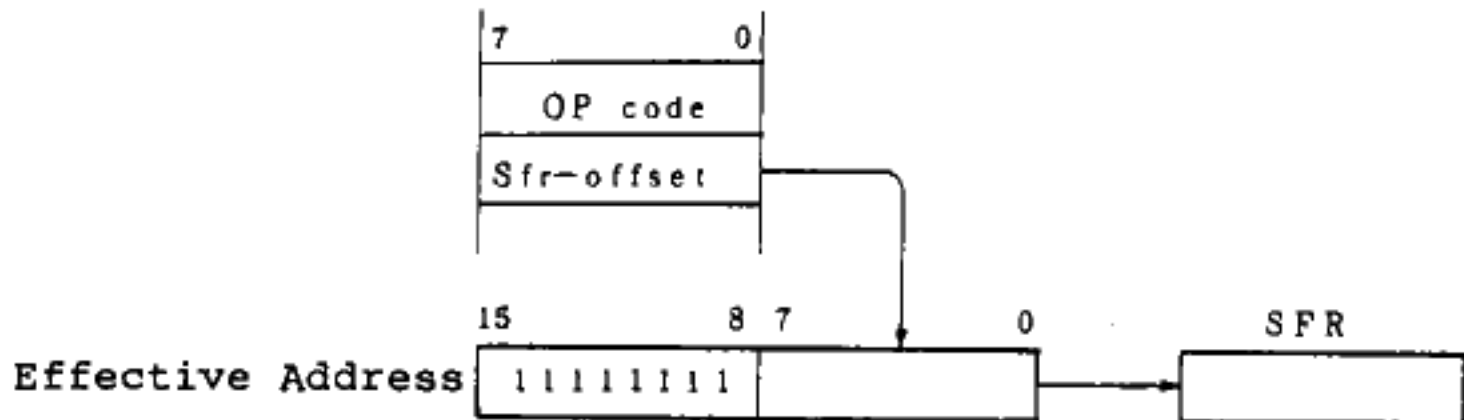
0	0	1	0	0	0	0	0
---	---	---	---	---	---	---	---

0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

### 10.5.5 SPECIAL FUNCTION REGISTER (SFR) ADDRESSING

This addressing addresses the special function register (SFR) memory mapped by the 8-bit immediate data in the instruction word.

The mapping space of the SFR which can be addressed by this addressing is the 256 bytes from FF00H to FFFFH. However, the SFR mapped to FF00H through FF1FH is accessed by short direct addressing instead of SFR addressing.



Identifier	Description
sfr	Special function register abbreviation
sfrp	16-bit handleable special function register abbreviation

Example 1: MOV sfr, A

OP code 0 0 0 1 ' 0 0 1 0

Sfr-offset

When PM0 is specified as sfr, it is described as follows:

MOV PM0, A

The OP code for this is:

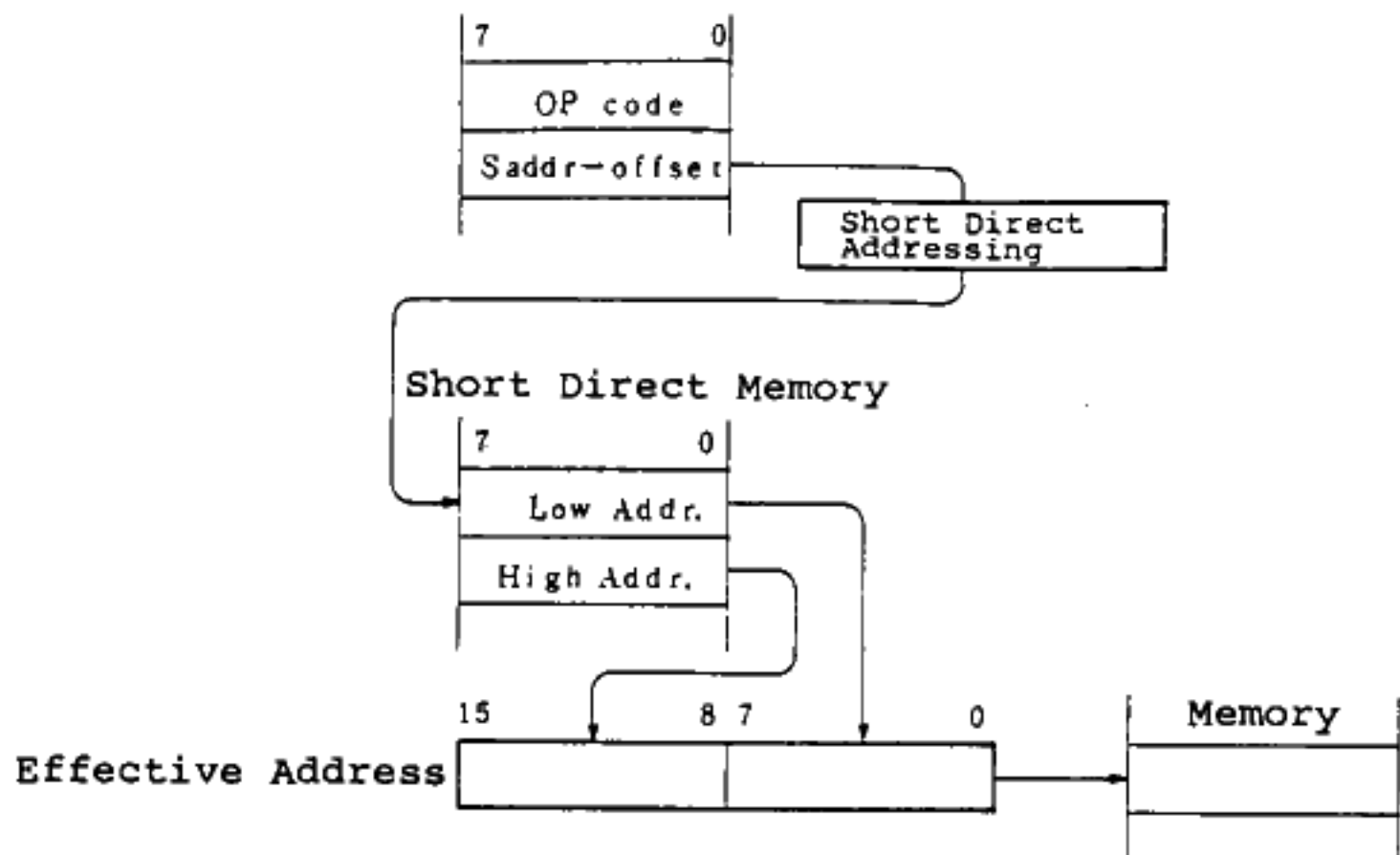
OP code 0 0 0 1 ' 0 0 1 0

0 0 1 0 ' 0 0 0 0

### 10.5.6 MEMORY INDIRECT ADDRESSING

This addressing addresses the objective memory with the contents of the 2-byte continuous short direct memory addressed by the 8-bit immediate data in the instruction word as the operand address.

This addressing is performed when executing an instruction with [saddrp] in the operands.



Identifier	Description
[saddrp]	[Label, numerical value within FE20H to FF1FH (even number)]

Example 1: XCH A, [saddrp]

OP code 

0	0	1	0	0	0	1	1
---	---	---	---	---	---	---	---

Saddr-offset
--------------

When FEAOH is taken as saddrp, it is described as follows:

XCH A, [OFEAOH]

The OP code for this is:

OP code 

0	0	1	0	0	0	1	1
---	---	---	---	---	---	---	---

1	0	1	0	0	0	0	0
---	---	---	---	---	---	---	---

### 10.5.7 REGISTER INDIRECT ADDRESSING

This addressing addresses the objective memory with the contents of the register set select flag (RSS) in the register bank specified by the register bank select flag (RBS2 to RBS0) and the register pair specified by the register pair code in the instruction word as the operand address.

This addressing is performed when executing an instruction having the operand format shown below.

Identifier	Description
mem	[DE], [HL], [DE+], [HL+], [DE-], [HL-], [VP], [UP]
[rp1]	[RP0], [RP1], [RP2], [RP3], [RP4], [RP5], [RP6], [RP7]

Register indirect addressing by register pair DE, HL is addressing with a function which prepares for the next addressing by incrementing/decrementing the contents of the register pair by one after memory is addressed.

In this case, [DE+], [HL+], [DE-], [HL-] is described at mem of the operand field.

Example 1: MOV A, mem

OP code . When register indirect mode [DE], [HL], [DE+], [HL+], [DE-], [HL-] described as mem

0 1 0 1 ' 1 mem
-----------------

. When a register indirect mode other than the above described as mem

0 0 0 1 ' 0 1 1 0
-------------------

0 mem ' 0 0 0 0
-----------------

When [DE] is specified as mem, the instruction is described as follows:

MOV A, [DE]

The OP code for this is:

OP code 

0 1 0 1 ' 1 1 0 0
-------------------

Example 2: ROR4 [rpl]

OP code 

0	0	0	0	0	1	0	1
---	---	---	---	---	---	---	---

1	0	0	0	1	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>
---	---	---	---	---	----------------	----------------	----------------

When RPO is selected as rpl, it is described as follows:

ROR4 [RPO]

The OP code for this is:

OP code 

0	0	0	0	0	1	0	1
---	---	---	---	---	---	---	---

1	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

Example 3: ADD A, mem

OP code (when register indirect mode described)

0	0	0	1	0	1	1	0
---	---	---	---	---	---	---	---

0	mem	1	0	0	0	0	0
---	-----	---	---	---	---	---	---

When [HL+] is specified as mem, it is described as follows:

ADD A, [HL+]

The OP code for this is:

OP code 

0	0	0	1	0	1	1	0
---	---	---	---	---	---	---	---

0	0	0	1	1	0	0	0
---	---	---	---	---	---	---	---



### 10.5.8 BASE ADDRESSING

This addressing addresses the objective memory with the sum of the contents of the 16-bit register and register pair (DE, SP, HL, UP, VP) specified by the register set select flag (RSS) in the register bank specified by the register bank select flag (RBS2 to RBS0) and the addressing code (mem) in the instruction word and the 8-bit immediate data of the operand as the operand address.

This addressing is performed when executing an instruction having the operand format shown below.

Identifier	Description
mem	[DE+byte], [SP+byte], [HL+byte], [UP+byte], [VP+byte]

Example 1: AND A, mem

OP code (when base mode described)

0 0 0 0 ' 0 1 1 0

0 mem ' 1 1 0 0

Offset

When base addressing of the sum of register pair VP and 10H is selected as mem, it is described as follows:

AND A, [VP+10H]

The OP code for this is:

OP code 0 0 0 0 ' 0 1 1 0

0 1 0 0 ' 1 1 0 0

0 0 0 1 ' 0 0 0 0

### 10.5.9 INDEX ADDRESSING

This addressing addresses the objective memory with the sum of the contents of specified by the register set select flag (RSS) in the register bank specified by the register bank select flag (RBS2 to RBS0) and the 8-bit register and 16-bit register pair (A, B, DE, HL) specified by the addressing code (mem) in the instruction word and the 16-bit immediate data of the operand as the operand address.

This addressing is performed when executing an instruction having the operand format shown below.

Identifier	Description
mem	word[DE], word[A], word[HL], word[B]

Example 1: ADDC A, mem

OP code (when index mode described)

0 0 0 0 ' 1 0 1 0

0 mem ' 1 0 0 1

Low ' Offset

High ' Offset

When index addressing of the sum of register pair DE and 4010H is selected as mem, it is described as follows:

ADDC A, 4010H[DE]

The OP code for this is:

OP code 0 0 0 0 ' 1 0 1 0

0 0 0 0 ' 1 0 0 1

0 0 0 1 ' 0 0 0 0

0 1 0 0 ' 0 0 0 0

### 10.5.10 BASE INDEX ADDRESSING

This addressing addresses the objective memory with the sum of the contents of the 16-bit register (DE, HL, VP) specified by the register set select flag (RSS) in the register bank specified by the register bank select flag (RBS2 to RBS0) and the addressing code (mem) in the instruction word and the 8/16 bit register (A, B, DE, HL) as the operand address.

This addressing is performed when executing an instruction having the operand format shown below.

Identifier	Description
mem	[DE+A], [HL+A], [DE+B], [HL+B], [VP+DE], [VP+HL]

Example 1: OR A, mem

OP code (when base index mode described)

0 0 0 1 0 1 1 1

0 mem 1 0 0 1

When base index addressing of the sum of register pair HL and register B is selected as mem, it is described as follows:

SUBC A, [HL+B]

The OP for this is:

OP code 0 0 0 1 0 1 1 1

0 0 1 1 1 0 1 1

## 10.6 DESCRIPTION OF INSTRUCTIONS

### 10.6.1 8-BIT DATA TRANSFER INSTRUCTIONS

**MOV r1, #byte**

**Function:** r1 + byte                      byte = 00H to FFH

Transfers the 8-bit immediate data specified by the second operand to the 8-bit register addressed by the first operand.

**Flag operation:** Unchanged

**Coding example:** MOV R1, #4DH; Set 4DH in register R1.

**MOV saddr, #byte**

**Function:** (saddr) + byte              saddr = FE20H to FF1FH  
byte = 00H to FFH

Transfers the 8-bit immediate data specified by the second operand to the short direct memory addressed by the first operand.

Describe the short direct memory address or label directly at saddr of the first operand.

**Flag operation:** Unchanged

**Coding example:** MOV OFE40H, #40H; Store 40H in address FE40H.

**MOV sfr, #byte**

**Function:** sfr + byte                      byte = 00H to FFH

Transfers the 8-bit immediate data specified by the second operand to the special function register sfr specified by the first operand.

**NOTE:** When STBC, WDM is described as sfr, a dedicated OP code different from this instruction is generated. (See paragraph 10.6.16 CPU control instructions.)

**Flag operation:** Unchanged

**Coding example:** MOV PM0, #0H; Specifies port 0 as an output port.

MOV r, rl

Function: r ← rl

Transfers the contents of the 8-bit register specified by the second operand to the 8-bit register specified by the first operand.

Flag operation: Unchanged

Coding examples: SEL RB0 ; Select bank 0.

MOV R15, R1; Transfer contents of the R1 (A) register to the R15 (H) register.

MOV A, rl

Function: A ← rl

Transfers the contents of the 8-bit register specified by the second operand to the A register.

Flag operation: Unchanged

Coding example:

MOV A, saddr

Function: A ← (saddr)                   saddr = FE20H to FF1FH

Transfers the contents of the short direct memory addressed by the second operand to the A register.

Describe the short direct memory address or label directly at saddr of the second operand.

Flag operation: Unchanged

Coding example: MOV A, OFE40H; Transfer the contents of address FE40H to the A register of the specified bank.

MOV saddr, A

Function: (saddr) + A                    saddr = FE20H to FF1FH

Transfers the contents of the A register to the short direct memory addressed by the first operand.

Describe the short direct memory address or label directly at saddr of the first operand.

Flag operation: Unchanged

Coding examples: SEL RB2                ; Select bank 2.

MOV R1, R0            ; Transfer the contents of  
   the X register to memory  
MOV 0FE30H, A        address FE30H.

MOV saddr, saddr

Function: (saddr) + (saddr)            saddr = FE20H to FF1FH

Transfers the contents of the short direct memory (source) addressed by the second operand to the short direct memory (destination) addressed by the first operand.

Describe the short direct memory address or label directly at saddr of the first and the second operands.

Flag operation: Unchanged

Coding example: MOV 0FE18H, 0FEC2H; Transfer the contents  
   of address FEC2H to  
   address FE18H.

MOV A, sfr

Function: A + sfr

Transfers the contents of the special function register specified by the second operand to the A register.

Flag operation: Unchanged

Coding example: MOV A, ADCR; Transfer the A/D converted  
   result to the A register of  
   the currently selected  
   register bank.

MOV sfr, A

Function: sfr ← A

Transfers the contents of the A register to the special function register specified by the first operand.

Flag operation: Unchanged

Coding examples: MOV P1, #00H; Specify port 1 as output port mode.

MOV P1, A ; Output the contents of the A register from port 1.

MOV A, mem

Function: A ← (mem)

Transfers the contents of memory addressed by the memory addressing specified in the second operand to the A register.

When auto increment ([DE+], [HL+])/auto decrement ([DE-], [HL-]) was specified as mem, the contents of register pair DE or register pair HL are automatically incremented or decremented by one after data transfer.

Flag operation: Unchanged

Coding examples: MOVW RP6, #3000H; DE (RP6) ← 3000H

MOV A, [DE+] ; Transfer the contents of memory address 3000H to the A register. (After transfer, increment the contents of DE by one.)

MOV mem, A

Function: (mem) + A

Transfers the contents of the A register to the memory addressed by the memory addressing specified in the first operand.

When auto increment ([DE+], [HL+])/auto decrement ([DE-], [HL-]) was specified as mem, the contents of register pair DE or register pair HL are automatically incremented/decremented by one after data transfer.

Flag operation: Unchanged

Coding examples: MOV R2, #0FFH ; Set FFH into the C (R2) register.

MOVW RP6, #3000H; Set 3000H into register pair DE.

MOV R1, #0H ; Set 0H into the A (R1) register.

LOOP:MOV [DE+], A ; Set the contents of the A register into address 3000H. (After transfer, increments the contents of DE by one.)

DBNZ C, \$LOOP ; Initialize the contents of memory addresses 3000H to 30FFH to 0H.

MOV A, [saddrp]

Function: A + ((saddrp)) saddrp = FE20H to FF1EH

Transfers the contents of memory addressed by the contents of short direct memory addressed by the second operand to the A register.

Describe the short direct memory address or label directly at saddrp of the second operand. However, this is limited to even numbered addresses.

Flag operation: Unchanged

Coding example:



MOV [saddrp], A

Function: ((saddrp)) + A            saddrp = FE20H to FF1EH

Transfers the contents of the A register to the memory addressed by the contents of the short direct memory addressed by the first operand.

Describe the short direct memory address or label directly at saddrp of the first operand. However, this is limited to even numbered addresses.

Flag operation: Unchanged

Coding example:

MOV A, !addr16

Function: A + (addr16)            addr16 = 0000H to FFFFH

Transfers the contents of memory addressed by the 16-bit immediate data specified by the second operand to the A register.

Flag operation: Unchanged

Coding example: MOV A, EXAM; Transfer the contents of memory indicated by label EXAM to the A register.

MOV !addr16, A

Function: (addr16) + A            addr16 = 0000H to FFFFH

Transfers the contents of the A register to the memory addressed by the 16-bit immediate data specified by the first operand.

Flag operation: Unchanged

Coding example:

MOV PSWL #byte

Function:  $PSW_L + \text{byte}$                       byte = 00H to FFH

Transfers the 8-bit immediate data specified by the second operand to the low-order 8-bits of the PSW.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	x	x	x

Coding example:

MOV PSWH, #byte

Function:  $PSW_H + \text{byte}$                       byte = 00H to FFH

Transfers the 8-bit immediate data specified by the second operand to the high-order 8 bits of the PSW.

Flag operation: Unchanged

Coding operation:

MOV PSWL, A

Function:  $PSW_L + A$

Transfers the contents of the A register to the low-order 8 bits of the PSW.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	x	x	x

Coding example:

MOV PSWH, A

Function:  $PSW_H + A$

Transfers the contents of the A register to the high-order 8 bits of the PSW.

Flag operation: Unchanged

Coding example:

MOV A, PSWL

Function:  $A \leftarrow PSW_L$

Transfers the contents of the low-order 8 bits of the PSW to the A register.

Flag operation: Unchanged

Coding example:

MOV A, PSWH

Function:  $A \leftarrow PSW_H$

Transfers the contents of the high-order 8 bits of the PSW to the A register.

Flag operation: Unchanged

Coding example:

XCH A, r1

Function:  $A \leftrightarrow r1$

Swaps the contents of the A register and the contents of the 8-bit register specified by the second operand.

Flag operation: Unchanged

Coding example:

XCH r, r1

Function:  $r \leftrightarrow r1$

Swaps the contents of the 8-bit register specified by the first operand and the contents of the 8-bit register specified by the second operand.

Flag operation: Unchanged

Coding example:

XCH A, mem

Function: A ↔ (mem)

Swaps the contents of the A register and the contents of the memory addressed by the memory addressing specified in the second operand.

When auto increment ([DE+], [HL+])/auto decrement ([DE-], [HL-]) was specified as mem, the contents of register pair DE or register pair HL are automatically incremented or decremented by one after the data is swapped.

Flag operation: Unchanged

Coding examples: MOV R2, #0FH ; C (R2) + 10H

MOVW RP7, #FE10H; HL (RP7) + FE10H

MOV R1, #0H ; A (R1) + 00H

LOOP:XCH [HL-], A ; Swap the contents of the A register and the contents of address FE10H. (Decrement the contents of HL after data transfer.)

DBNZ C, \$LOOP ; Shift the contents of FE01H to FE10H forward one address at a time. (Address FE10H: 00H)

XCH A, saddr

Function: A ↔ (saddr) saddr = FE20H to FF1FH

Swap the contents of the A register and the contents of the short direct memory addressed by the second operand.

Describe the short direct memory address or label directly at saddr of the second operand.

Flag operation: Unchanged

Coding example:

XCH A, sfr

Function: A ↔ sfr

Swaps the contents of the A register and the contents of the special function register specified by the second operand.

Flag operation: Unchanged

Coding example: XCH A, RXB; Swap the contents of the A register and the contents of the serial receive buffer.

XCH A, [saddrp]

Function: A ↔ ((saddrp))            saddrp = FE20H to FF1EH

Swaps the contents of the A register and the contents of memory addressed by the contents of the short direct memory addressed by the second operand.

Describe the short direct memory address or label directly at saddrp of the second operand. However, this is limited to even numbered addresses.

Flag operation: Unchanged

Coding example:

XCH saddr, saddr

Function: (saddr) ↔ (saddr)            saddr = FE20H to FF1FH

Swaps the contents of the short direct memory addressed by the first operand and the contents of the short direct memory addressed by the second operand.

Describe the short direct memory address or label directly at saddr of the first and second operands.

Flag operation: Unchanged

Coding example:

## 10.6.2 16-BIT DATA TRANSFER INSTRUCTIONS

MOVW rpl, #word

Function: rpl + word                      word = 0000H to FFFFH

Transfers the 16-bit immediate data specified by the second operand to the 16-bit register pair specified by the first operand.

Flag operation: Unchanged

Coding example: MOVW RPO, #0AA55H; Transfer AA55H to the AX register pair.

MOVW saddrp, #word

Function: (saddrp) + word              saddrp = FE20H to FF1EH  
word = 0000H to FFFFH

Transfers the 16-bit immediate data specified by the second operand to the 2-byte area of the short direct memory addressed by the first operand.

Describe the short direct memory address or label directly at saddrp of the first operand. However, this is limited to even numbered addresses.

Flag operation: Unchanged

Coding example:

MOVW sfrp, #word

Function: sfrp + word                      word = 0000H to FFFFH

Transfers the 16-bit immediate data specified by the second operand to the 16-bit special function register specified by the first operand.

Flag operation: Unchanged

Coding example: MOVW PWM0, #0FF00H; Set FF00H into the PWM0 register.



AND1 CY, /saddr.bit

Function:  $CY + \overline{CY \wedge (\text{saddr.bit})}$     saddr = FE20H to FF1FH  
bit = 0 to 7

ANDs the inverted contents of the short direct memory bit addressed by the second operand and the contents of the carry flag and places the result into the carry flag.

Specify the short direct memory bit address or label directly at the saddr.bit operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:

AND1 CY, sfr.bit

Function:  $CY + \overline{CY \wedge \text{sfr.bit}}$     bit = 0 to 7

ANDs the contents of the bit addressed by the 3-bit immediate data of the second operand of the special function register specified by the second operand and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:



MOVl CY, X.bit

Function: CY + X.bit                      bit = 0 to 7

Transfers the contents of the bit addressed by the 3-bit immediate data of the second operand of the X register to the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:

MOVl CY, PSWH.bit

Function: CY + PSW<sub>H</sub>.bit                      bit = 0 to 7

Transfers the contents of the bit addressed by the 3-bit immediate data of the second operand of the high-order 8 bits of the program status word (PSW) to the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:

MOVl CY, PSWL.bit

Function: CY + PSW<sub>L</sub>.bit                      bit = 0 to 7

Transfers the contents of the bit addressed by the 3-bit immediate data of the second operand of the low-order 8 bits of the program status word (PSW) to the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:



Coding example: MOV A, #88H

ADD A, #79H; A = 01H, CY = 1, AC = 1

ADJ4 ; A ← A+66H, A = 67H, CY = 1

; 88+79 = 167

ADD	{	10001000	88H
		+ ) 01111001	79H
		<hr/>	
		00000001	
		CY AC	
ADJ4	{	+ ) 01100110	66H
		01100111	67H

### 10.6.9 BIT MANIPULATION INSTRUCTIONS

MOVl CY, saddr.bit

Function: CY + (saddr.bit)      saddr = FE20H to FF1FH  
bit = 0 to 7

Transfers the contents of the short direct memory bit addressed by the second operand to the carry flag.

Specify the short direct memory bit address or label directly in the saddr.bit operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:

MOVl CY, sfr.bit

Function: CY + sfr.bit      bit = 0 to 7

Transfers the contents of the bit addressed by the 3-bit immediate data of the special function register specified by the second operand to the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:

MOVl CY, A.bit

Function: CY + A.bit      bit = 0 to 7

Transfers the contents of the bit addressed by the 3-bit immediate data of the second operand of the A register to the carry flag.

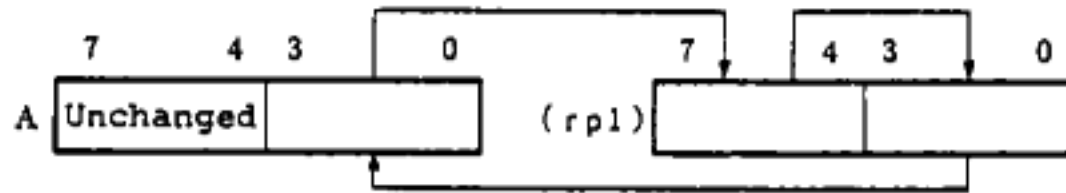
Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:

ROR4 [rpl]

Function:  $A_3$  to  $A_0 + (rpl)_3$  to  $(rpl)_0$ ,  $(rpl)_7$  to  $(rpl)_4 + A_3$  to  $A_0$ ,  $(rpl)_3$  to  $(rpl)_0 + (rpl)_7$  to  $(rpl)_4$



Rotates the low-order 4 bits of the A register and the high-order 4 bits and low-order 4 bits of the memory addressed by the operand to the right in 4 bit units. Bits 7 to 4 of the A register are not affected.

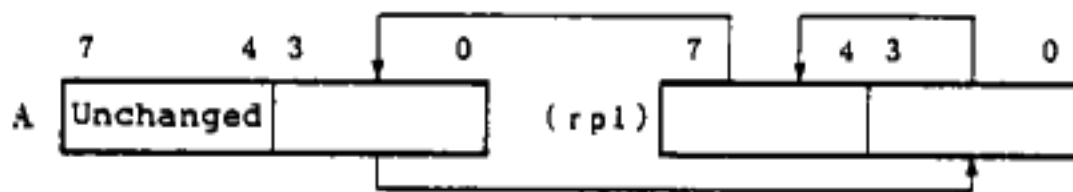
Flag operation: Unchanged

Coding example: ROR4 [HL]; A (HL)

	7	4	3	0	7	4	3	0
Before Execution	0	0	0	0	0	0	1	0
After Execution	0	0	0	0	0	0	1	1

ROL4 [rpl]

Function:  $A_3$  to  $A_0 + (rpl)_7$  to  $(rpl)_4$ ,  $(rpl)_3$  to  $(rpl)_0 + A_3$  to  $A_0$ ,  $(rpl)_7$  to  $(rpl)_4 + (rpl)_3$  to  $(rpl)_0$



Rotates the low-order 4 bits of the A register and the high-order 4 bits and low-order 4 bits of the memory addressed by the operand to the left in 4 bit units. Bits 7 to 4 of the A register are not affected.

Flag operation: Unchanged

Coding example:

### 10.6.8 BCD ADJUSTMENT INSTRUCTION

ADJ4

Function:

Judges the contents of the A register, carry flag (CY), auxiliary carry flag (AC), and subtraction flag (SUB) and performs decimal adjustment as shown below. This instruction is meaningful only after operation between decimal (BCD) data was executed.

Condition			Operation
SUB = 0	A <sub>3</sub> to A <sub>0</sub> ≤ 9 AC = 0	A <sub>7</sub> to A <sub>4</sub> ≤ 9 and CY = 0	A + A
		A <sub>7</sub> to A <sub>4</sub> ≥ 10 or CY = 1	A + A+01100000B
	A <sub>3</sub> to A <sub>0</sub> ≥ 10 AC = 0	A <sub>7</sub> to A <sub>4</sub> < 9 and CY = 0	A + A+00000110B
		A <sub>7</sub> to A <sub>4</sub> ≥ 9 or CY = 1	A + A+01100110B
	AC = 1	A <sub>7</sub> to A <sub>4</sub> ≤ 9 and CY = 0	A + A+00000110B
		A <sub>7</sub> to A <sub>4</sub> ≥ 10 or CY = 1	A + A+01100110B
SUB = 1	AC = 0	CY = 0	A + A
		CY = 1	A + A-01100000B
	AC = 1	CY = 0	A + A-00000110B
		CY = 1	A + A-01100110B

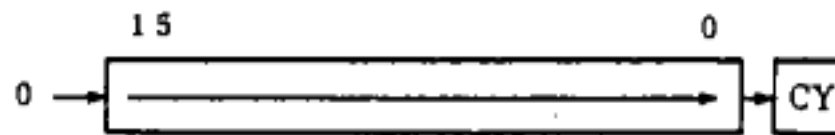
Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	P		x

SHRW rpl, n

Function: (CY + rpl<sub>0</sub>, rpl<sub>15</sub> + 0, rpl<sub>m-1</sub> + rpl<sub>m</sub>) x n times

n = 0 to 7



Shifts the contents of the 16-bit register pair specified by the first operand to the right the number of bits specified by the 3-bit immediate data given as the second operand. The contents of the LSB of the 16-bit register pair are shifted to the carry flag and 0 is set in the MSB. However, when n = 0, the above operation is not performed.

Flag operation:

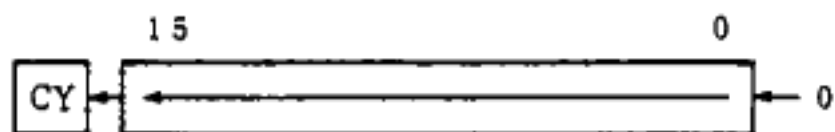
S	Z	AC	P/V	SUB	CY
x	x	0	P	0	x

Coding example:

SHLW rpl, n

Function: (CY + rpl<sub>15</sub>, rpl<sub>0</sub> + 0, rpl<sub>m+1</sub> + rpl<sub>m</sub>) x n times

n = 0 to 7



Shifts the contents of the 16-bit register pair specified by the first operand to the left the number of bits specified by the 3-bit immediate data given as the second operand. The contents of the MSB of the 16-bit register pair are shifted to the carry flag and 0 is set in the LSB. However, when n = 0, the operation above is not performed.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	0	P	0	x

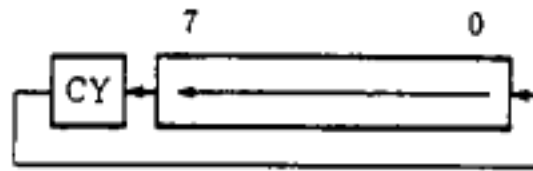
Coding example: SHLW RPO, 1; Double the contents of the AX register pair.



ROLC r1, n

Function: (CY + r1<sub>7</sub>, r1<sub>0</sub> + CY, r1<sub>m+1</sub> + r1<sub>m</sub>) x n times

n = 0 to 7



Rotates the contents, including the carry flag, of the 8-bit register specified by the first operand to the left the number of bits specified by the 3-bit immediate data given as the second operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
			P	0	x

Coding example:

SHR r1, n

Function: (CY + r1<sub>0</sub>, r1<sub>7</sub> + 0, r1<sub>m-1</sub> + r1<sub>m</sub>) x n times

n = 0 to 7



Shifts the contents of the 8-bit register specified by the first operand to the right the number of bits specified by the 3-bit immediate data given as the second operand. The contents of the LSB of the 8-bit register are shifted to the carry flag and the 0 is set in the MSB. However, when n = 0, the above operation is not performed.

Flag operation:

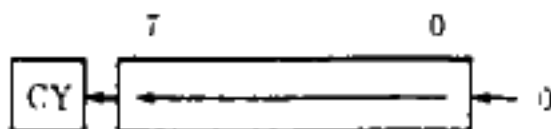
S	Z	AC	P/V	SUB	CY
x	x	0	P	0	x

Coding example: SHR R1, 1; Half the contents of the A register. (Remainder is placed into CY.)

SHL r1, n

Function:  $(CY \leftarrow r1_7, r1_0 \leftarrow 0, r1_{m+1} \leftarrow r1_m) \times n$  times

n = 0 to 7



Shifts the contents of the 8-bit register specified by the first operand to the left the number of bits specified by the 3-bit immediate data given as the second operand. The contents of the MSB of the 8 bit register are shifted to the carry flag and the 0 is set in the LSB. However, when n = 0, the operation above is not performed.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	0	P	0	x

Coding example: SHL R14, 3; Shift the contents of the L register 3 bits to the left. Set the contents of bit 5 before the shift into the carry flag.

### 10.6.7 SHIFT/ROTATION INSTRUCTIONS

ROR  $rl, n$

Function:  $(CY, rl_7 + rl_0, rl_{m-1} + rl_m) \times n$  times

$n = 0$  to  $7$



Rotates the contents of the 8-bit register specified by the first operand to the right the number of bits specified by the 3-bit immediate data written as the second operand. The contents of the LSB of the 8-bit register are transferred to the MSB and are set in the carry flag. However, when  $n = 0$ , the operation above is not performed.

Flag operation:

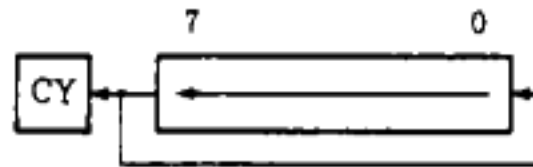
S	Z	AC	P/V	SUB	CY
			P	0	x

Coding example:

ROL r1, n

Function: (CY, r1<sub>0</sub> + r1<sub>7</sub>, r1<sub>m+1</sub> + r1<sub>m</sub>) x n times

n = 0 to 7



Rotates the contents of the 8-bit register specified by the first operand to the left the number of bits specified by the 3-bit immediate data written as the second operand. The contents of the MSB of the 8-bit register are transferred to the LSB and are set in the carry flag. However, when n = 0, the operation above is not performed.

Flag operation:

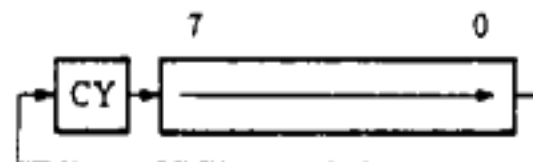
S	Z	AC	P/V	SUB	CY
			P	0	x

Coding example:

RORC r1, n

Function: (CY + r1<sub>0</sub>, r1<sub>7</sub> + CY, r1<sub>m-1</sub> + r1<sub>m</sub>) x n times

n = 0 to 7



Rotates the contents, including the carry flag, of the 8-bit register specified by the first operand are shifted to the right the number of bits specified by the 3-bit immediate data given as the second operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
			P	0	x

Coding example:

DEC saddr

Function: (saddr) + (saddr)-1 saddr = FE20H to FF1FH

Decrements the contents of the short direct memory addressed by the operand.

Specify the short direct memory address or label directly in the saddr operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	

Coding example:

INCW rp2

Function: rp2 + rp2+1

Increments the contents of the 16-bit register pair specified by the operand.

Flag operation: Unchanged

Coding example: INCW DE; DE DE+1

INCW saddrp

Function: (saddrp) + (saddrp)+1

saddrp = FE20H to FF1EH

Increments the contents of the 2-byte area of the short direct memory addressed by the operand.

Specify the short direct memory address or label directly in the saddrp operand. However, this is limited to even numbered addresses.

Flag operation: Unchanged

Coding example:

DECW rp2

Function:  $rp2 + rp2 - 1$

Decrements the contents of the 16-bit register pair specified by the operand.

Flag operation: Unchanged

Coding example: DECW HL; HL + HL-1

DECW saddrp

Function:  $(saddrp) + (saddrp) - 1$

$saddrp = FE20H \text{ to } FF1EH$

Decrements the contents of the 2-byte area of the short direct memory addressed by the operand.

Specify the short direct memory address or label directly in the saddrp operand. However, this is limited to even numbered addresses.

Flag operation: Unchanged

Coding example:

DIVUX rpl

Function: AXDE, rpl  $\leftarrow$  AXDE $\div$ rpl

Divides the contents (32-bit data) of register pair AX (high-order 16 bits) and register pair DE (low-order 16 bits) by the contents of the 16 bit register pair specified by the operand and places the quotient into register pair AX (high-order 16 bits) and register pair DE (low-order 16 bits) and the remainder into the 16-bit register specified by the operand.

Flag operation: Unchanged

Coding example:

## 10.6.6 INCREMENT/DECREMENT INSTRUCTIONS

INC r1

Function:  $r1 + r1+1$

Increments the contents of the 8-bit register specified by the operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	

Coding example:

INC saddr

Function:  $(saddr) + (saddr)+1$   $saddr = FE20H$  to  $FF1FH$

Increments the contents of the short direct memory addressed by the operand.

Specify the short direct memory address or label directly in the saddr operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	

Coding example: INC TB1; Increments by 1 the contents of the label TB1 short direct memory.

DEC r1

Function:  $r1 + r1-1$

Decrements the contents of the 8-bit register specified by the operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	

Coding example:



CMPW saddrp, saddrp

Function: (saddrp)-(saddrp) saddrp = FE20H to FF1EH

Subtracts the contents of the 2-byte area of the short direct memory addressed by the second operand from the contents of the 2-byte area of the short direct memory addressed by the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

After instruction execution, the contents of the short direct memory are unchanged.

Specify the short direct memory address or label directly in the first and second saddrp operands. However, this is limited to even numbered addresses.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

### 10.6.5 MULTIPLY/DIVIDE INSTRUCTIONS

MULU rl

Function:  $AX \leftarrow A \times rl$

Multiplies the contents of the A register and the contents of the 8-bit register specified by the operand and places the product into register pair AX.

Flag operation: Unchanged

Coding example:

DIVUW rl

Function:  $AX, rl \leftarrow AX \div rl$

Divides the contents of register pair AX by the contents of the 8-bit register specified by the operand and places the quotient into register pair AX and the remainder into the register specified by the operand.

Flag operation: Unchanged

Coding example:

MULUW rpl

Function:  $AX, rpl \leftarrow AX \times rpl$

Multiplies the contents of register pair AX and the contents of the 16-bit register pair specified by the operand and places the high-order 16 bits of the product into register pair AX and the lower-order 16 bits into the 16-bit register pair specified by the operand.

Flag operation: Unchanged

Coding example:

CMPW sfrp, #word

Function: sfrp-word

word = 0000H to FFFFH

Subtracts the 16-bit immediate data specified by the second operand from the contents of the 16-bit special function register specified by the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

After instruction execution, the contents of the 16-bit special function register specified by the first operand are unchanged.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

CMPW rp, rpl

Function: rp-rpl

Subtracts the contents of the 16-bit register pair specified by the second operand from the contents of the 16-bit register pair specified by the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

After instruction execution, the contents of the register pair specified by the first and second operands are unchanged.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

CMPW AX, saddrp

Function: AX-(saddrp)

saddrp = FE20H to FF1EH

Subtracts the contents of the 2-byte area of the short direct memory addressed by the second operand from the contents of the register pair AX. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

After instruction execution, the contents of register pair AX and the short direct memory are unchanged.

Specify the short direct memory address or label directly in the second saddrp operand. However, this is limited to even numbered addresses.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

CMPW AX, sfrp

Function: AX-sfrp

Subtracts the contents of the 16-bit special function register specified by the second operand from the contents of register pair AX. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

After instruction execution, the contents of register pair AX and the 16-bit special function register specified by the second operand are unchanged.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

**SUBW AX, sfrp**

**Function:** AX, CY ← AX-sfrp

Subtracts the contents of the 16-bit special function register specified by the second operand from the contents of register pair AX. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

**Flag operation:**

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

**Coding example:**

**SUBW saddrp, saddrp**

**Function:** (saddrp), CY ← (saddrp)-(saddrp)

saddrp = FE20H to FF1EH

Subtracts the contents of the 2-byte area of the short direct memory addressed by the second operand from the contents of the 2-byte area of the short direct memory addressed by the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

Specify the short direct memory address and label directly in the first and second saddrp operands. However, this is limited to even numbered addresses.

**Flag operation:**

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

**Coding example:**

CMPW AX, #word

Function: AX-word

word = 0000H to FFFFH

Subtracts the 16-bit immediate data specified by the second operand from the contents of register pair AX. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

After instruction execution, the contents of register pair AX are unchanged.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

CMPW saddrp, #word

Function: (saddrp)-word

saddrp = FE20H to FF1EH

word = 0000H to FFFFH

Subtracts the 16-bit immediate data specified by the second operand from the contents of the short direct memory addressed by the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

After instruction execution, the contents of the short direct memory are unchanged.

Specify the short direct memory address or label directly in the first saddrp operand. However, this is limited to even numbered addresses.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example: CMPW OFE50H, #8000H  
BGT \$JMP

; When the contents of memory addresses FE51H, FE50H are greater than those branch of 8000H to the address indicated by label JMP.

SUBW saddrp, #word

Function: (saddrp), CY + (saddrp)-word

saddrp = FE20H to FF1EH  
word = 0000H to FFFFH

Subtracts the 16-bit immediate data specified by the second operand from the contents of the 2-byte area of the short direct memory addressed by the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

Specify the short direct memory address or label directly in the first saddrp operand. However, this is limited to even numbered addresses.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

SUBW sfrp, #word

Function: sfrp, CY + sfrp-word

word = 0000H to FFFFH

Subtracts the 16-bit immediate data specified by the second operand from the contents of the 16-bit special function register specified by the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

SUBW rp, rpl

Function: rp, CY + rp-rpl

Subtracts the contents of the 16-bit register pair specified by the second operand from the contents of the 16-bit register pair specified by the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

SUBW AX, saddrp

Function: AX, CY + AX-(saddrp)

saddrp = FE20H to FF1EH

Subtracts the contents of the 2-byte area of the short direct memory addressed by the second operand from the contents of register pair AX. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

Specify the short direct memory address or label directly in the second saddrp operand. However, this is limited to even numbered addresses.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:



**ADDW AX, saddrp**

**Function:** AX, CY + AX+(saddrp)

saddrp = FE20H to FF1EH

Binary adds the contents of the 2-byte area of the short direct memory addressed by the second operand to the contents of register pair AX. When an addition result carry occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

Describe the short direct memory address or label directly in the saddrp second operand. However, this is limited to even numbered addresses.

**Flag operation:**

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

**Coding example:**

**ADDW AX, sfrp**

**Function:** AX, CY + AX+sfrp

Binary adds the contents of the 16-bit special function register specified by the second operand to the contents of register pair AX. When an addition result carry occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

**Flag operation:**

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

**Coding example:**

ADDW saddrp, saddrp

Function: (saddrp), CY + (saddrp)+(saddrp)

saddrp = FE20H to FF1EH

Binary adds the contents of the 2-byte area of the short direct memory addressed by the second operand to the contents of the 2-byte area of the short direct memory addressed by the first operand. When an addition result carry occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

Specify the short direct memory address or label directly in the first and second saddrp operands. However, this is limited to even numbered addresses.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

Coding example:

SUBW AX, #word

Function: AX CY + AX-word      word = 0000H to FFFFH

Subtracts the 16-bit immediate data specified by the second operand from the contents of register pair AX. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

#### 10.6.4 16-BIT OPERATION INSTRUCTIONS

ADDW AX, #word

Function: AX, CY + AX+word      word = 0000H to FFFFH

Binary adds the 16-bit immediate data specified by the second operand to the contents of register pair AX. When an addition result carry occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

Coding example:

ADDW saddrp, #word

Function: (saddrp), CY + (saddrp)+word

saddrp = FE20H to FF1EH

word = 0000H to FFFFH

Binary adds the 16-bit immediate data specified by the second operand to the contents of the 2-byte area of the short direct memory addressed by the first operand. When an addition result carry occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

Specify the short direct memory address or label directly in the saddrp first operand. However, this is limited to even addresses.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

Coding example:

**ADDW sfrp, #word**

**Function: sfrp, CY + sfrp+word**

**word = 0000H to FFFFH**

Binary adds the 16-bit immediate data specified by the second operand to the contents of the 16-bit special function register specified by the first operand. When an addition result carry occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

**Flag operation:**

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

**Coding example:**

**ADDW rp, rpl**

**Function: rp, CY + rp+rpl**

Binary adds the contents of the 16-bit register pair specified by the second operand to the contents of the 16-bit register pair specified by the first operand. When an addition result carry occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

**Flag operation:**

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

**Coding example:**

CMP A, sfr

Function: A-sfr

Subtracts the contents of the special function register specified by the second operand from the contents of the A register. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

After instruction execution, the contents of the A register and special function register are unchanged.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

CMP saddr, saddr

Function: (saddr)-(saddr)      saddr = FE20H to FF1FH

Subtracts the contents of the short direct memory addressed by the second operand from the contents of the short direct memory addressed by the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

After instruction execution, the contents of the short direct memory are unchanged.

Specify the short direct memory address or label directly in the first and second saddr operands.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

CMP A, mem

Function: A-(mem)

Subtracts the contents of the memory addressed by the addressing memory described at the second operand from the contents of the A register. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

When auto increment ([DE+], [HL+])/auto decrement ([DE-], [HL-]) was specified as mem, the contents of register pair DE or register pair HL are automatically incremented or decremented by 1 after the operation is executed.

After instruction execution, the contents of the A register and memory are unchanged.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

CMP mem, A

Function: (mem)-A

Subtracts the contents of the A registers from the contents of the memory addressed by the memory addressing given as the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

When auto increment ([DE+], [HL+])/auto decrement ([DE-], [HL-]) was specified as mem, the contents of register pair DE or register pair HL are automatically incremented or decremented by 1 after the operation is executed.

After instruction execution, the contents of the A register and memory are unchanged.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

CMP saddr, #byte

Function: (saddr)-byte

saddr = FE20H to FF1FH  
byte = 00H to FFH

Subtracts the 8-bit immediate data specified by the second operand from the contents of the short direct memory addressed by the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

After instruction execution, the contents of the short direct memory are unchanged.

Described the short direct memory address or label directly at saddr of the first operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

CMP sfr, #byte

Function: sfr-byte

byte = 00H to FFH

Subtracts the 8-bit immediate data specified by the second operand from the contents of the special function register specified by the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

After instruction execution, the contents of the special function register are unchanged.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

CMP r, r1

Function: r-r1

Subtracts the contents of the 8-bit register specified by the second operand from the contents of the 8-bit register specified by the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

After instruction execution, the contents of 8-bit registers r and r1 are unchanged.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

CMP A, saddr

Function: A-(saddr)

saddr = FE20H to FF1FH

Subtracts the contents of the short direct memory addressed by the second operand from the contents of the A register. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

After instruction execution, the contents of the A register and the short direct memory are unchanged.

Describe the short direct memory address or label directly at saddr of the second operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:



XOR saddr, saddr

Function: (saddr) + (saddr)⊕(saddr)

saddr = FE20H to FF1FH

EXCLUSIVE-ORs the contents of the short direct memory addressed by the first operand and the contents of the short direct memory addressed by the second operand and places the result into the short direct memory addressed by the first operand.

Describe the short direct memory address or label directly at saddr of the first and second operands.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P		0

Coding example:

XOR A, mem

Function: A + A⊕(mem)

EXCLUSIVE-ORs the contents of the A register and the contents of the memory addressed by the memory addressing specified in the second operand and places the result into the A register.

When auto increment ([DE+], [HL+])/auto decrement ([DE-], [HL-]) was specified as mem, the contents of register pair DE or register pair HL are automatically incremented or decremented by one after the operation is executed.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P		0

Coding example:

XOR mem, A

Function: (mem) + (mem)⊕A

EXCLUSIVE-ORs the contents of the memory addressed by the memory addressing specified in the first operand and the contents of the A register and places the result into the memory addressed by the first operand.

When auto increment ([DE+], [HL+])/auto decrement ([DE-], [HL-],) was specified as mem, the contents of register pair DE or register pair HL are automatically incremented or decremented by one after the operation is executed.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P	0	

Coding example:

CMP A, #byte

Function: A-byte

byte = 00H to FFH

Subtracts the 8-bit immediate data specified by the second operand from the contents of the A register. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

After instruction execution, the contents of the A register are unchanged.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

XOR saddr, #byte

Function: (saddr) + (saddr)#byte

saddr = FE20H to FF1FH  
byte = 00H to FFH

EXCLUSIVE-ORs the contents of the short direct memory addressed by the first operand and the 8-bit immediate data specified by the second operand and places the result into the short direct memory addressed by the first operand.

Describe the short direct memory address or label directly at saddr of the first operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P		0

Coding example:

XOR sfr, #byte

Function: sfr + sfr#byte      byte = 00H to FFH

EXCLUSIVE-ORs the contents of the special function register specified by the first operand and the 8-bit immediate data specified by the second operand and places the result into the special function register specified by the first operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P		0

Coding example:

XOR r, r1

Function:  $r \leftarrow r \oplus r1$

EXCLUSIVE-ORs the contents of the 8-bit register specified by the first operand and the contents of the 8-bit register specified by the second operand and places the result into the 8-bit register specified by the first operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P	0	

Coding example:

XOR A, saddr

Function:  $A \leftarrow A \oplus (\text{saddr})$       saddr = FE20H to FF1FH

EXCLUSIVE-ORs the contents of the A register and the contents of the short direct memory addressed by the second operand and places the result into the A register.

Describe the short direct memory address or label directly at saddr of the second operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P	0	

Coding example:

XOR A, sfr

Function:  $A \leftarrow A \oplus \text{sfr}$

EXCLUSIVE-ORs the contents of the A register and the contents of the special function register specified by the second operand and places the result into the A register.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P	0	

Coding example:

OR saddr, saddr

Function: (saddr) + (saddr)V(saddr)

saddr = FE20H to FF1FH

ORs the contents of the short direct memory addressed by the first operand and the contents of the short direct memory addressed by the second operand and places the result into the short direct memory addressed by the first operand.

Describe the short direct memory address or label directly at saddr of the first and second operands.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P		0

Coding example:

OR A, mem

Function: A + AV(mem)

ORs the contents of the A register and the contents of the memory addressed by the memory addressing specified in the second operand and places the result into the A register.

When auto increment ([DE+], [HL+])/auto decrement ([DE-], [HL-]) was specified as mem, the contents of register pair DE or register pair HL are automatically incremented or decremented by one after the operation is executed.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P		0

Coding example:

OR mem, A

Function: (mem) ← (mem)VA

ORs the contents of the memory addressed by the memory addressing specified in the first operand and the contents of the A register and places the result into the memory specified by the first operand.

When auto increment ([DE+], [HL+])/auto decrement ([DE-], [HL-]) was specified as mem, the contents of register pair DE or register pair HL are automatically incremented or decremented by one after the operation is executed.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P	0	

Coding example:

XOR A, #byte

Function: A ← A#byte                      byte = 00H to FFH

EXCLUSIVE-ORs the contents of the A register and the 8-bit immediate data specified by the second operand and places the result into the A register.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P	0	

Coding example: XOR A, #0FFH; Invert the contents of the A register.

OR saddr, #byte

Function: (saddr) + (saddr)Vbyte

saddr = FE20H to FF1FH  
byte = 00H to FFH

ORs the contents of the short direct memory addressed by the first operand and the 8-bit immediate data specified by the second operand and places the result into the short direct memory addressed by the first operand.

Describe the short direct memory address or label directly at saddr of the first operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P		0

Coding example:

OR sfr, #byte

Function: sfr + sfrVbyte            byte = 00H to FFH

ORs the contents of the special function register specified by the first operand and 8-bit immediate data specified by the second operand and places the result into the special function register specified by the first operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P		0

Coding example: MOV Pm1, #00H  
OR P1, #F0H ; Output 1 from the high-order 4 bits of port 1.  
  
(Low-order 4 bits are unchanged)

OR r, r1

Function:  $r + rVr1$

ORs the contents of the 8-bit register specified by the first operand and the contents of the 8-bit register specified by the second operand and places the result into the 8-bit register specified by the first operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P		0

Coding example:

OR A, saddr

Function:  $A + AV(saddr)$                        $saddr = FE20H \text{ to } FF1FH$

ORs the contents of the A register and the contents of the short direct memory addressed by the second operand and places the result into the A register.

Describe the short direct memory address or label directly at saddr of the second operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P		0

Coding example:

OR A, sfr

Function:  $A + AVsfr$

ORs the contents of the A register and the contents of the special function register specified by the second operand and places the result into the A register.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P		0

Coding example:



AND saddr, saddr

Function:  $(saddr) + (saddr) \wedge (saddr)$

saddr = FE20H to FF1FH

ANDs the contents of the short direct memory addressed by the first operand and the contents of the short direct memory addressed by the second operand and places the result into the short direct memory addressed by the first operand.

Describe the short direct memory address or label directly at saddr of the first and second operands.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P		0

Coding example:

AND A, mem

Function:  $A + A \wedge (\text{mem})$

ANDs the contents of the A register and the contents of the memory addressed by the memory addressing specified in the second operand and places the result into the A register.

When auto increment ([DE+], [HL+])/auto decrement ([DE-], [HL-]) was specified as mem, the contents of register pair DE or register pair HL are automatically incremented or decremented by one after the operation is executed.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P		0

Coding example:

AND mem, A

Function: (mem) + (mem)∧A

ANDs the contents of the memory addressed by the memory addressing specified in the first operand and the contents of the A register and places the result into the memory specified by the first operand.

When auto increment ([DE+], [HL+])/auto decrement ([DE-], [HL-]) was specified as mem, the contents of register pair DE or register pair HL are automatically incremented or decremented by one after the operation is executed.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P		0

Coding example:

OR A, #byte

Function: A + AVbyte                      byte = 00H to FFH

ORs the contents of the A register and the 8-bit immediate data specified by the second operand and places the result into the A register.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P		0

Coding example:

AND sfr, #byte

Function: sfr + sfr^byte      byte = 00H to FFH

ANDs the contents of the special function register specified by the first operand and the 8-bit immediate data specified by the second operand and places the result into the special function register specified by the first operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P		0

Coding example: AND MM, #0FH;      Reset only the high-order 4 bits of the MM register.

(Low-order 4 bits are not changed.)

AND r, r1

Function: r + r^r1

ANDs the contents of the 8-bit register specified by the first operand and the contents of the 8-bit register specified by the second operand and places the result into the 8-bit register specified by the first operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P		0

Coding example:

AND A, saddr

Function:  $A \leftarrow A \wedge (saddr)$       saddr = FE20H to FF1FH

ANDs the contents of the A register and the contents of the short direct memory addressed by the second operand and places the result into the A register.

Describe the short direct memory address or label directly at saddr of the second operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P		0

Coding example:

AND A, sfr

Function:  $A \leftarrow A \wedge sfr$

ANDs the contents of the A register and the contents of the special function register specified by the second operand and places the result into the A register.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P		0

Coding example:

SUBC A, mem

Function:  $A, CY \leftarrow A - (\text{mem}) - CY$

Subtracts the contents, including the carry flag, of the memory addressed by the memory addressing specified in the second operand from the contents of the A register. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

When auto increment ([DE+], [HL+])/auto decrement ([DE-], [HL-]) was specified as mem, the contents of register pair DE or register pair HL are automatically incremented or decremented by one after the subtraction is executed.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

SUBC mem, A

Function:  $(\text{mem}), CY \leftarrow (\text{mem}) - A - CY$

Subtracts the contents, including the carry flag, of the A register from the memory addressed by the memory addressing specified in the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

When auto increment ([DE+], [HL+])/auto decrement ([DE-], [HL-]) was specified as mem, the contents of register pair DE or register pair HL are automatically incremented or decremented by one after the subtraction is executed.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

AND A, #byte

Function:  $A + A \wedge \text{byte}$

byte = 00H to FFH

ANDs the contents of the A register and the 8-bit immediate data specified by the second operand and places the result into the A register.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P		0

Coding example:

AND saddr, #byte

Function:  $(\text{saddr}) + (\text{saddr}) \wedge \text{byte}$

saddr = FE20H to FF1FH

byte = 00H to FFH

ANDs the contents of the short direct memory addressed by the first operand and the 8-bit immediate data specified by the second operand and places the result into the short direct memory addressed by the first operand.

Describe the short direct memory address or label directly at saddr of the first operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x		P		0

Coding example:

SUBC r, r1

Function: r, CY + r-r1-CY

Subtracts the contents, including the carry flag, of the 8-bit register specified by the second operand from the contents of the 8-bit register specified by the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

SUBC A, saddr

Function: A, CY + A-(saddr)-CY

saddr = FE20H to FF1FH

Subtracts the contents, including the carry flag, of the short direct memory addressed by the second operand from the contents of the A register. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

Describe the short direct memory address or label directly at saddr of the second operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

SUBC A, sfr

Function:  $A, CY \leftarrow A - \text{sfr} - CY$

Subtracts the contents, including the carry flag, of the special function register specified by the second operand from the contents of the A register. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

SUBC saddr, saddr

Function:  $(\text{saddr}), CY \leftarrow (\text{saddr}) - (\text{saddr}) - CY$

saddr = FE20H to FF1FH

Subtracts the contents, including the carry flag, of the short direct memory addressed by the second operand from the contents of the short direct memory addressed by the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

Describe the short direct memory address or label directly at saddr of the first and second operands.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:



SUB mem, A

Function: (mem), CY + (mem)-A

Subtracts the contents of the A register from the contents of the memory addressed by the memory addressing specified in the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

When auto increment ([DE+], [HL+])/auto decrement ([DE-], [HL-]) was specified as mem, the contents of register pair DE or register pair HL are automatically incremented or decremented by one after the subtraction is executed.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

SUBC A, #byte

Function: A, CY + A-byte-CY      byte = 00H to FFH

Subtracts the 8-bit immediate data, including the carry flag, specified by the second operand from the contents of the A register. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

SUBC saddr, #byte

Function: (saddr), CY + (saddr)-byte-CY

saddr = FE20H to FF1FH  
byte = 00H to FFH

Subtracts the 8-bit immediate data, including the carry flag, specified by the second operand from the contents of the short direct memory addressed by the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

Describe the short direct memory address or label directly at saddr of the first operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

SUBC sfr, #byte

Function: sfr, CY + sfr-byte-CY

byte = 00H to FFH

Subtracts the 8-bit immediate data, including the carry flag, specified by the second operand from the contents of the special function register specified by the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

SUB A, saddr

Function: A, CY ← A-(saddr)      saddr = FE20H to FF1FH

Subtracts the contents of the short direct memory addressed by the second operand from the contents of the A register. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

Describe the short direct memory address or label directly at saddr of the second operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

SUB A, sfr

Function: A, CY ← A-sfr

Subtracts the contents of the special function register specified by the second operand from the contents of the A register. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

SUB saddr, saddr

Function: (saddr), CY + (saddr)-(saddr)

saddr = FE20H to FF1FH

Subtracts the contents of the short direct memory addressed by the second operand from the contents of the short direct memory addressed by the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

Describe the short direct memory address or label directly at saddr of the first and second operands.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

SUB A, mem

Function: A, CY + A-(mem)

Subtracts the contents of the memory addressed by the memory addressing specified in the second operand from the contents of the A register. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

When auto increment ([DE+], [HL+])/auto decrement ([DE-], [HL-]) was specified as mem, the contents of register pair DE or register pair HL are automatically incremented or decremented by one after subtraction is executed.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

SUB A, #byte

Function: A, CY + A-byte                      byte = 00H to FFH

Subtracts the 8-bit immediate data specified by the second operand from the contents of the A register. If a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

SUB saddr, #byte

Function: (saddr), CY + (saddr)-byte

saddr = FE20H to FF1FH  
byte = 00H to FFH

Subtracts the 8-bit immediate data specified by the second operand from the contents of the short direct memory addressed by the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

Describe the short direct memory address or label directly at saddr of the first operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

SUB sfr, #byte

Function: sfr, CY ← sfr-byte    byte = 00H to FFH

Subtracts the 8-bit immediate data specified by the second operand from the contents of the special function register specified by the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

SUB r, r1

Function: r, CY ← r-r1

Subtracts the contents of the 8-bit register specified by the second operand from the contents of the 8-bit register specified by the first operand. When a subtraction result borrow occurs, the carry flag is set. When a borrow does not occur, the carry flag is reset.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

ADDC A, sfr

Function:  $A, CY + A+sfr+CY$

Binary adds the contents, including the carry flag, of the special function register specified by the second operand to the contents of the A register. When an addition result carry occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

Coding example:

ADDC saddr, saddr

Function:  $(saddr), CY + (saddr)+(saddr)+CY$

saddr = FE20H to FF1FH

Binary adds the contents, including the carry flag, of the short direct memory addressed by the second operand to the contents of the short direct memory addressed by the first operand. When an additional result carry out occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

Describe the short direct memory address or label directly at saddr of the first and second operands.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

Coding example:

ADDC A, mem

Function:  $A, CY \leftarrow A + (\text{mem}) + CY$

Binary adds the contents, including the carry flag, of the memory addressed by the memory addressing specified in the second operand to the contents of the A register. When an addition result carry occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

When auto increment ([DE+], [HL+])/auto decrement ([DE-], [HL-]) was specified as mem, the contents of register pair DE or register pair HL are automatically incremented or decremented by one after the addition is executed.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

Coding example:

ADDC mem, A

Function:  $(\text{mem}), CY \leftarrow (\text{mem}) + A + CY$

Binary adds the contents, including the carry flag, of the A register to the contents of the memory addressed by the memory addressing specified in the first operand. When an addition result carry occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

When auto increment ([DE+], [HL+])/auto decrement ([DE-], [HL-]) was specified as mem, the contents of register pair DE or register pair HL are automatically incremented or decremented by one after the addition is executed.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

Coding example:



ADDC saddr, #byte

Function: (saddr), CY + (saddr)+byte+CY

saddr = FE20H to FF1FH  
byte = 00H to FFH

Binary adds the 8-bit immediate data, including the carry flag, specified by the second operand to the contents of the short direct memory addressed by the first operand. When an addition result carry occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

Describe the short direct memory address or label directly at saddr of the first operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

Coding example:

ADDC sfr, #byte

Function: sfr, CY + sfr+byte+CY

byte = 00H to FFH

Binary adds the 8-bit immediate data, including the carry flag, specified by the second operand to the contents of the special function register specified by the first operand. When an addition result carry occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

Coding example:

ADDC r, r1

Function: r, CY + r+r1+CY

Binary adds the contents, including the carry flag, of the 8-bit register specified by the second operand to the contents of the 8-bit register specified by the first operand. When an addition result carry occurs, the carry flag is set. When a carry flag does not occur, the carry flag is reset.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

Coding example:

ADDC A, saddr

Function: A, CY + A+(saddr)+CY

saddr = FE20H to FF1FH

Binary adds the contents, including the carry flag, of the short direct memory addressed by the second operand to the contents of the A register. When an addition result carry occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

Describe the short direct memory address or label directly at saddr of the second operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

Coding example:

ADD saddr, saddr

Function: (saddr), CY + (saddr)+(saddr)

saddr = FE20H to FF1FH

Binary adds the contents of the short direct memory addressed by the second operand to the contents of the short direct memory addressed by the first operand. When an addition result carry occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

Describe the short direct memory address or label directly at saddr of the first and second operands.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

Coding example:

ADD A, mem

Function: A, CY + A+(mem)

Binary adds the contents of the memory addressed by the memory addressing specified in the second operand to the contents of the A register. When an addition result carry occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

When auto increment ([DE+], [HL+])/auto decrement ([DE-], [HL-]) was specified as mem, the contents of register pair DE or register pair HL are automatically incremented or decremented by one after the addition is executed.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

Coding example:

ADD mem, A

Function: (mem), CY + (mem)+A

Binary adds the contents of the A register to the contents of the memory addressed by the memory addressing specified in the first operand. When an addition result carry out occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

When auto increment ([DE+], [HL+])/auto decrement ([DE-], [HL-]) was specified as mem, the contents of register pair DE or register pair HL are automatically incremented or decremented by one after the addition is executed.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

Coding examples: MOV R2, #0FH ; C + 0FH

MOV RP7, #FE20H; HL + FE20H

LOOP:MOV A, #10H

ADD [HL+80H], A; Add 10H to the  
contents of FEAOH to  
FEAFH.

DBNZ C, \$LOOP

ADDC A, #byte

Function: A, CY + A+byte+CY    byte = 00H to FFH

Binary adds the 8-bit immediate data, including the carry flag, specified by the second operand, to the contents of the A register. When an additional result carry occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

Coding example:

ADD sfr, #byte

Function: sfr, CY + sfr+byte    byte = 00H to FFH

Binary adds the 8-bit immediate data specified by the second operand to the contents of the special function register specified by the first operand. When an addition result carry occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

Coding example:    ADD CROOL #1H;    Binary add the low-order 8 bits of the CRO0 register and 1H and place the result into the CRO0 register.

ADD r, r1

Function: r, CY + r+r1

Binary adds the contents of the register specified by the second operand to the contents of the register specified by the first operand. When an additional result carry occurs, the carry flag is set. When a carry out does not occur, the carry flag is reset.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

Coding example:

ADD A, saddr

Function: A, CY + A+(saddr) saddr = FE20H to FF1FH

Binary adds the contents of the short direct memory addressed by the second operand to the contents of the A register. When an addition result carry occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

Describe the short direct memory address of label directly at saddr of the second operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

Coding example:

ADD A, sfr

Function: A, CY + A+sfr

Binary adds the contents of the special function register specified by the second operand to the contents of the A register. When an addition result carry occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

Coding example:

XCHW AX, sfrp

Function: AX ↔ sfrp

Swaps the contents of register pair AX and the contents of the 16-bit special function register specified by the second operand.

Flag operation: Unchanged

Coding example: XCHW AX, PWMOL; Swap the contents of the PWM0 register and the AX register pair.

XCHW saddrp, saddrp

Function: (saddrp) ↔ (saddrp) saddrp = FE20H to FF1EH

Swaps the contents of the 2-byte area of the short direct memory addressed by the first operand and the contents of the 2-byte area of the short direct memory addressed by the second operand.

Describe the short direct memory address or label directly at saddrp of the first and second operands. However, this is limited to even numbered addresses.

Flag operation: Unchanged

Coding example:

XCHW rp, rpl

Function: rp ↔ rpl

Swaps the contents of the 16-bit register pair specified by the first operand and the contents of the 16-bit register pair specified by the second operand.

Flag operation: Unchanged

Coding example:

### 10.6.3 8-BIT OPERATION INSTRUCTIONS

ADD A, #byte

Function: A, CY + A+byte                      byte = 00H to FFH

Binary adds the 8-bit immediate data specified by the second operand to the contents of the A register. When an addition result carry occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

Coding example: ADD A, #40H; Binary add 40H to the contents of the A register.

ADD saddr, #byte

Function: (saddr), CY + (saddr)+byte

saddr = FE20H to FF1FH  
byte = 00H to FFH

Binary adds the 8-bit immediate data specified by the second operand to the contents of the short direct memory addressed by the first operand. When an addition result carry occurs, the carry flag is set. When a carry does not occur, the carry flag is reset.

Describe the short direct memory address or label directly at saddr of the first operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	0	x

Coding example:



MOVW saddrp, saddrp

Function: (saddrp) + (saddrp) saddrp = FE20H to FF1EH

Transfer the 2-byte area of the short direct memory addressed by the second operand to the contents of the 2-byte area of the short direct memory addressed by the first operand.

Describe the short direct memory address or label directly at saddrp of the first and second operands. However, this is limited to even numbered addresses.

Flag operation: Unchanged

Coding example:

MOVW AX, sfrp

Function: AX + sfrp

Transfers the contents of the 16-bit special function register specified by the second operand to register pair AX.

Flag operation: Unchanged

Coding example: MOVW AX, CPT0; Transfer the contents of the CPT0 register to register pair AX.

MOVW sfrp, AX

Function: sfrp + AX

Transfers the contents of register pair AX to the 16-bit special function register specified by the first operand.

Flag operation: Unchanged

Coding example:



## 10.6.2 16-BIT DATA TRANSFER INSTRUCTIONS

MOVW rpl, #word

Function: rpl + word                      word = 0000H to FFFFH

Transfers the 16-bit immediate data specified by the second operand to the 16-bit register pair specified by the first operand.

Flag operation: Unchanged

Coding example: MOVW RPO, #0AA55H; Transfer AA55H to the AX register pair.

MOVW saddrp, #word

Function: (saddrp) + word              saddrp = FE20H to FF1EH  
word = 0000H to FFFFH

Transfers the 16-bit immediate data specified by the second operand to the 2-byte area of the short direct memory addressed by the first operand.

Describe the short direct memory address or label directly at saddrp of the first operand. However, this is limited to even numbered addresses.

Flag operation: Unchanged

Coding example:

MOVW sfrp, #word

Function: sfrp + word                      word = 0000H to FFFFH

Transfers the 16-bit immediate data specified by the second operand to the 16-bit special function register specified by the first operand.

Flag operation: Unchanged

Coding example: MOVW PWM0, #0FF00H; Set FF00H into the PWM0 register.

AND1 CY, /sfr.bit

Function:  $CY + \overline{CY \wedge \text{sfr.bit}}$  bit = 0 to 7

ANDs the inverted contents of the bit addressed by the 3-bit immediate data of the second operand of the special function register specified by the second operand and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					X

Coding example:

AND1 CY, A.bit

Function:  $CY + \overline{CY \wedge A.bit}$  bit = 0 to 7

ANDs the contents of the bit addressed by the 3-bit immediate data of the second operand of the A register and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					X

Coding example:

AND1 CY, /A.bit

Function:  $CY + \overline{A.bit}$  bit = 0 to 7

ANDs the inverted contents of the A register bit addressed by the 3-bit immediate data of the second operand and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					X

Coding example:

AND1 CY, X.bit

Function:  $CY + CY \wedge X.bit$  bit = 0 to 7

ANDs the contents of the X register bit addressed by the 3-bit immediate data of the second operand and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					X

Coding example:

AND1 CY, /X.bit

Function:  $CY + CY \wedge \overline{X.bit}$  bit = 0 to 7

ANDs the inverted contents of the X register bit addressed by the 3-bit immediate data of the second operand and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					X

Coding example:

AND1 CY, PSWH.bit

Function:  $CY + CY \wedge PSW_H.bit$  bit = 0 to 7

ANDs the contents of the bit addressed by the 3-bit immediate data of the second operand of the high-order 8 bits of the program status word (PSW) and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					X

Coding example:

AND1 CY, /PSWH.bit

Function:  $CY + \overline{CY \wedge PSW_H.bit}$  bit = 0 to 7

ANDs the inverted contents of the bit addressed by the 3-bit immediate data of the second operand of the high-order 8 bits of the program status word (PSW) and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:

AND1 CY, PSWL.bit

Function:  $CY + CY \wedge PSW_L.bit$  bit = 0 to 7

ANDs the contents of the bit addressed by the 3-bit immediate data of the second operand of the low-order 8 bits of the program status word (PSW) and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:

AND1 CY, /PSWL.bit

Function:  $CY + \overline{CY \wedge PSW_L.bit}$  bit = 0 to 7

ANDs the inverted contents of the bit addressed by the 3-bit immediate data of the second operand of the low-order 8 bits of the program status word (PSW) and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:



ORl CY, sfr.bit

Function:  $CY + CY \vee \text{sfr.bit}$       bit = 0 to 7

ORs the contents of the bit addressed by the 3-bit immediate data of the second operand of the special function register specified by the second operand and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:

ORl CY, /sfr.bit

Function:  $CY + \overline{CY \vee \text{sfr.bit}}$       bit = 0 to 7

ORs the inverted contents of the bit, addressed by the 3-bit immediate data of the second operand, specified by the second operand of the special function register and the contents of the carry flag, and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:

ORl CY, A.bit

Function:  $CY + CY \vee A.\text{bit}$       bit = 0 to 7

ORs the contents of the A register bit addressed by the 3-bit immediate data of the second operand and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:



ORl CY, /A.bit

Function:  $CY + \overline{CYVA.bit}$  bit = 0 to 7

ORs the inverted contents of the A register bit addressed by the 3-bit immediate data of the second operand and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					X

Coding example:

ORl CY, X.bit

Function:  $CY + \overline{CYVX.bit}$  bit = 0 to 7

ORs the contents of the X register bit addressed by the 3-bit immediate data of the second operand and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					X

Coding example:

ORl, CY, /X.bit

Function:  $CY + \overline{CYVX.bit}$  bit = 0 to 7

ORs the inverted contents of the X register bit addressed by the 3-bit immediate data of the second operand and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					X

Coding example:

ORl, CY, PSWH.bit

Function:  $CY + CY \vee PSW_H.bit$  bit = 0 to 7

ORs the contents of the bit addressed by the 3-bit immediate data of the second operand of the high-order 8 bits of the program status word (PSW) and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:

ORl CY, /PSWH.bit

Function:  $CY + CY \vee \overline{PSW_H.bit}$  bit = 0 to 7

ORs the inverted contents of the bit addressed by the 3-bit immediate data of the second operand of the high-order 8 bits of the program status word (PSW) and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:

ORl CY, PSWL.bit

Function:  $CY + CY \vee PSW_L.bit$  bit = 0 to 7

ORs the contents of the bit addressed by the 3-bit immediate data of the second operand of the low-order 8 bits of the program status word (PSW) and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:

ORl CY, /PSWL.bit

Function:  $CY \leftarrow CY \vee \overline{PSW_L.bit}$  bit = 0 to 7

ORs the inverted contents of the bit addressed by the 3-bit immediate data of the second operand of the low-order 8 bits of the program status word (PSW) and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					X

Coding example:

XORl CY, saddr.bit

Function:  $CY \leftarrow CY \oplus (saddr.bit)$  saddr = FE20H to FF1FH  
bit = 0 to 7

EXCLUSIVE-ORs the contents of the short direct memory bit addressed by the second operand and the contents of the carry flag and places the result into the carry flag.

Specify the short direct memory bit address or label directly in the saddr.bit operand.

Flag operation:

S	Z	AC	P/V	SUB	CY
					X

Coding example:

XORl CY, sfr.bit

Function:  $CY + CY \nabla \text{sfr.bit}$  bit = 0 to 7

EXCLUSIVE-ORs the contents of the bit addressed by the 3-bit immediate data of the second operand of the special function register specified by the second operand addressed and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:

XORl CY A.bit

Function:  $CY + CY \nabla A.bit$  bit = 0 to 7

EXCLUSIVE-ORs the contents of the A register bit addressed by the 3-bit immediate data of the second operand and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:

XORl CY, X.bit

Function:  $CY + CY \nabla X.bit$  bit = 0 to 7

EXCLUSIVE-ORs the contents of the X register bit addressed by the 3-bit immediate data of the second operand and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:

XORl CY, PSWH.bit

Function:  $CY \oplus PSW_H.bit$  bit = 0 to 7

EXCLUSIVE-ORs the contents of the bit addressed by the 3-bit immediate data of the second operand of the high-order 8 bits of the program status word (PSW) and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:

XORl CY, PSWL.bit

Function:  $CY \oplus PSW_L.bit$  bit = 0 to 7

EXCLUSIVE-ORs the contents of the bit addressed by the 3-bit immediate data of the second operand of the low-order 8 bits of the program status word (PSW) and the contents of the carry flag and places the result into the carry flag.

Flag operation:

S	Z	AC	P/V	SUB	CY
					x

Coding example:

SETl saddr.bit

Function:  $(saddr.bit) \oplus 1$  saddr = FE20H to FF1FH  
bit = 0 to 7

Sets the short direct memory bit addressed by the operand to 1.

Specify the short direct memory bit address or label directly in the saddr.bit operand.

Flag operation: Unchanged

Coding example:

SET1 sfr.bit

Function: sfr.bit + 1                      bit = 0 to 7

Sets the bit of the special function register specified by the operand, addressed by the 3-bit immediate data of the operand to 1.

Flag operation: Unchanged

Coding example:

SET1 A.bit

Function: A.bit + 1                      bit = 0 to 7

Sets the A register bit addressed by the 3-bit immediate data of the operand to 1.

Flag operation: Unchanged

Coding example:

SET1 X.bit

Function: X.bit + 1                      bit = 0 to 7

Sets the X register bit addressed by the 3-bit immediate data of the operand to 1.

Flag operation: Unchanged

Coding example:

SET1 PSWH.bit

Function: PSW<sub>H</sub>.bit + 1                      bit = 0 to 7

Sets the bit addressed by the 3-bit immediate data of the operand of the high-order 8 bits of the program status word (PSW) to 1.

Flag operation: Unchanged

Coding example:



CLRl X.bit

Function: X.bit + 0                      bit = 0 to 7

Clears the X register bit addressed by the 3-bit immediate data of the operand to 0.

Flag operation: Unchanged

Coding example:

CLRl PSWH.bit

Function: PSW<sub>H</sub>.bit + 0                      bit = 0 to 7

Clears the bit addressed by the 3-bit immediate data of the operand of the high-order 8 bits of the program status word (PSW) to 0.

Flag operation: Unchanged

Coding example:

CLRl PSWL.bit

Function: PSW<sub>L</sub>.bit + 0                      bit = 0 to 7

Clears the bit addressed by the 3-bit immediate data of the operand of the low-order 8 bits of the program status word (PSW) to 0.

Flag operation: The flag addressed by the operand is cleared to 0.

Coding example:

NOTl saddr.bit

Function: (saddr.bit) +  $\overline{\text{(saddr.bit)}}$

saddr = FE20H to FF1FH  
bit = 0 to 7

Inverts the contents of the short direct memory bit addressed by the operand.

Specify the short direct memory address or label directly in the saddr.bit operand.

Flag operation: Unchanged

Coding example:



NOT1 sfr.bit

Function:  $\text{sfr.bit} + \overline{\text{sfr.bit}}$  bit = 0 to 7

Inverts the contents of the bit addressed by the 3-bit immediate data of the operand of the special function register specified by the operand.

Flag operation: Unchanged

Coding example:

NOT1 A.bit

Function:  $\text{A.bit} + \overline{\text{A.bit}}$  bit = 0 to 7

Inverts the contents of the A register bit addressed by the 3-bit immediate data of the operand.

Flag operation: Unchanged

Coding example:

NOT1 X.bit

Function:  $\text{X.bit} + \overline{\text{X.bit}}$  bit = 0 to 7

Inverts the contents of the X register bit addressed by the 3-bit immediate data of the operand.

Flag operation: Unchanged

Coding example:

NOT1 PSWH.bit

Function:  $\text{PSW}_H.\text{bit} + \overline{\text{PSW}_H.\text{bit}}$  bit = 0 to 7

Inverts the contents of the bit addressed by the 3-bit immediate data of the operand of the high-order 8 bits of the program status word (PSW).

Flag operation: Unchanged

Coding example:

NOT1 PSWL.bit

Function:  $PSW_L.bit \leftarrow \overline{PSW_L.bit}$  bit = 0 to 7

Inverts the contents of the bit addressed by the 3-bit immediate data of the operand of the low-order 8 bits of the program status word (PSW).

Flag operation: The contents of the flag addressed by the operand are inverted.

Coding example:

SET1 CY

Function:  $CY + 1$

Sets the carry flag to 1.

Flag operation:

S	Z	AC	P/V	SUB	CY
					1

Coding example:

CLR1 CY

Function:  $CY + 0$

Clears the carry flag to 0.

Flag operation:

S	Z	AC	P/V	SUB	CY
					0

Coding example:

NOT1 CY

Function:  $CY + \overline{CY}$

Inverts the contents of the carry flag.

Flag operation: 

S	Z	AC	P/V	SUB	CY
					x

Coding example:

## 10.6.10 CALL/RETURN INSTRUCTIONS

**CALL !addr16**

**Function:**  $(SP-1) + (PC+3)_H$ ,  $(SP-2) + (PC+3)_L$ ,  
 $PC + \text{addr16}$ ,  $SP + SP-2$

**addr16 = 0000H to FEFFH**

After saving the start address (return address) of the next instruction to the memory (stack) addressed by the stack pointer (SP) and decrementing the SP, branches to the location addressed by the 16-bit immediate data specified by the operand.

**NOTE:** Addresses FF00H to FFFFH cannot be instruction-fetched. Therefore, do not write them in addr16.

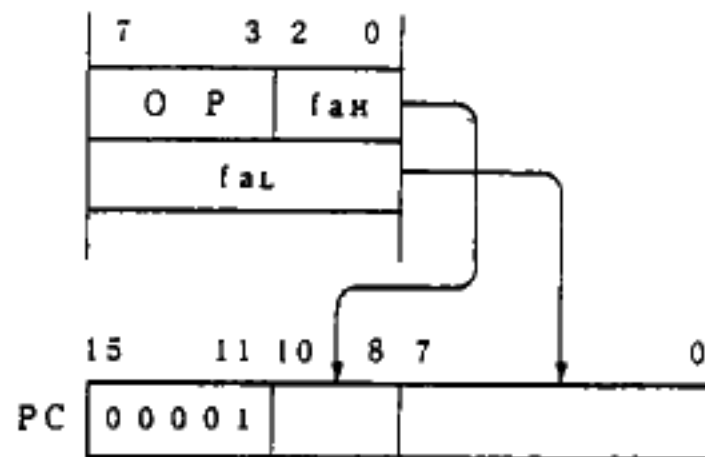
**Flag operation:** Unchanged

**Coding example:**

**CALLF !addr11**

Function:  $(SP-1) + (PC+2)_H$ ,  $(SP-2) + (PC+2)_L$ ,  
 $PC_{15}$  to  $PC_{11} + 00001$ ,  $PC_{10}$  to  $PC_0 + fa$ ,  $SP + SP-2$

addr11 = 0800H to 0FFFH



After saving the start address (return address) of the next instruction to the memory (stack) addressed by the stack pointer (SP) and decrementing the SP, branches to the location addressed by the effective address consisting of 11-bit immediate data fa in the OP code.

The range which can be called is limited to addresses 0800H to 0FFFH. Consider the entry address range and write the branch destination address directly in the addr11 operand by label or numeric.

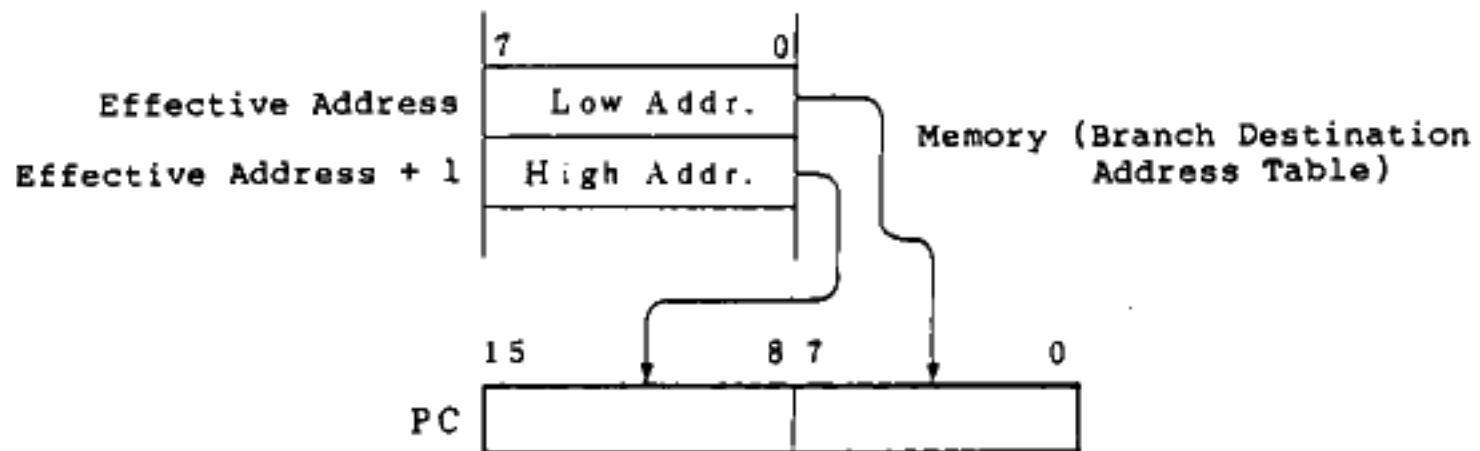
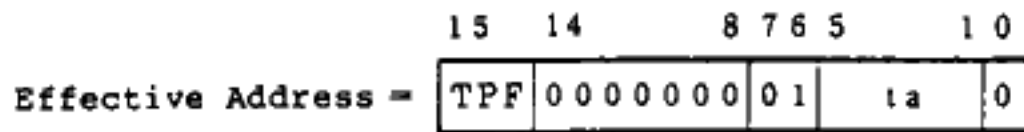
Flag operation: Unchanged

Coding example:

CALLT [addr5]

Function:  $(SP-1) + (PC+1)_H, (SP-2) + (PC+1)_L,$   
 $PC_H + (TPF, 000000001, ta, 1)$   
 $PC_L + (TPF, 000000001, ta, 0), SP + SP-2$

addr5 = 40H to 7EH



After saving the start address (return address) of the next instruction to the memory (stack) addressed by the stack pointer (SP) and decrementing the SP, sets the contents of the memory (branch destination address table) addressed by the effective address consisting of the 5-bit immediate data ta in the OP code into the program counter (PC) and branches to the address specified by its contents.

The branch destination address table must be located at addresses 0040H to 007FH. Write the branch destination address table directly in the addr5 operand by label or numeric.

Remarks: The branch destination address table can be installed at external memory area (8040H to 807FH) by setting the TPF flag to 1.

Flag operation: Unchanged

Coding example: CALL [TBL1]; Branch to the location addressed by the contents of the table specified by label TBL1.

CALL rpl

Function:  $(SP-1) + (PC+2)_H$ ,  $(SP-2) + (PC+2)_L$ ,  $PC_H + rpl_H$ ,  
 $PC_L + rpl_L$ ,  $SP + SP-2$

After saving the start address (return address) of the next instruction to the memory (stack) addressed by the stack pointer (SP) and decrementing the SP, sets the contents of the 16-bit register pair specified by the operand into the program counter (PC) and branches.

Flag operation: Unchanged

Coding example:

CALL [rpl]

Function:  $(SP-1) + (PC+2)_H$ ,  $(SP-2) + (PC+2)_L$ ,  
 $PC_H + (rpl+1)$ ,  $PC_L + (rpl)$ ,  $SP + SP-2$

After saving the start address (return address) of the next instruction to the memory (stack) addressed by the stack pointer (SP) and decrementing the SP, sets the contents of the 2-byte area of memory addressed by the contents of the 16-bit register pair specified by the operand into the program counter (PC) and branches.

Flag operation: Unchanged

Coding example:

BRK

Function:  $(SP-1) + PSW_H$ ,  $(SP-2) + PSW_L$ ,  
 $(SP-3) + (PC+1)_H$ ,  $(SP-4) + (PC+1)_L$ ,  
 $PC_L + (003EH)$ ,  $PC_H + (003FH)$ ,  $SP + SP-4$ ,  $IE + 0$

Saves the start address (return address) of the next instruction and the program status word (PSW) to the memory (stack) addressed by the stack pointer (SP) and decrements the SP sets the contents of the BRK instruction branch destination address table (003EH, 003FH) in the program counter (PC) and branches. The IE flag is reset to 0 and subsequent maskable interrupts are disabled. The BRK instruction is also accepted in the DI state (IE = 0).

Flag operation: Unchanged

Coding example:

## RET

Function:  $PC_L + (SP)$ ,  $PC_H + (SP+1)$ ,  $SP + SP+2$

Restores the contents of the memory (stack) addressed by the stack pointer (SP) to the program counter (PC) and increments the SP.

Flag operation: Unchanged

Coding example:

## RETI

Function:  $PC_L + (SP)$ ,  $PC_H + (SP+1)$ ,  $PSW_L + (SP+2)$ ,  
 $PSW_H + (SP+3)$ ,  $SP + SP+4$ ,  $EOS + 0$

Restores the contents of the memory (stack) addressed by the stack pointer (SP) to the program counter (PC) and the program status word, then increments the SP. It also clears the EOS flag.

This instruction is used when returning from an interrupt service routine.

NOTE: When returning from an interrupt service routine by BRK instruction, always set the EOS flag by SET1 EOS instruction immediately before the RETI instruction.

Flag operation:

S	Z	AC	P/V	SUB	CY
R	R	R	R	R	R

Coding example:



## 10.6.11 STACK MANIPULATION INSTRUCTIONS

### PUSH post

Function:  $\{(SP-1) + post_H, (SP-2) + post_L, SP + SP-2\}$   
x n times

(n; number of register pairs described as post)

Saves the contents of the 16-bit register pair specified by the operand to memory (stack) and decrements the SP.

Multiple register pairs can be specified in the post operand.

The save operation is performed sequentially from the register pair assigned to bit 7 of the 8-bit immediate data of the second byte (Post Byte). The high-order side of the register pair is saved to the stack addressed by  $(SP-2_{n+})$  and the low-order side is saved to the stack addressed by  $(SP-2_n)$ .

Flag operation: Unchanged

Coding example:

### PUSH PSW

Function:  $(SP-1) + PSW_H, (SP-2) + PSW_L, SP + SP-2$

Saves the contents of the program status word (PSW) to the memory (stack) addressed by the stack pointer (SP) and decrements the SP.

Flag operation: Unchanged

Coding example:

PUSHU post

Function:  $\{(UP-1) + post_H, (UP-2) + post_L, UP + UP-2\}$   
x n times

(n; number of register pairs written as post)

Saves the contents of the 16-bit register pair specified by the operand to the memory addressed by the user stack pointer (UP) and decrements the UP.

Multiple register pair names can be specified in the post operand.

Save operation is performed sequentially from the register pair assigned to bit 7 of the 8-bit immediate data of the second byte (Post Byte). The high-order side of the register pair is saved to the memory addressed by  $(UP-2_{n+1})$  and the low-order side is saved to the memory addressed by  $(UP-2_n)$ .

Flag operation: Unchanged

Coding example:

POP post

Function:  $\{post_L + (SP), post_H + (SP+1), SP + SP+2\}$  x  
n times

(n; number of register pairs written as post)

Restores the contents of the memory (stack) addressed by the stack pointer (SP) to the 16-bit register pair specified by the operand and increments the SP.

Multiple register pair names can be specified in the post operand.

Transfer is performed sequentially from the register pair assigned to bit 0 of the 8 bit immediate data of the second byte (Post Byte). The contents of the stack addressed by  $(SP+2_{n-2})$  are restored to the low-order side of the register pair and the contents of the stack addressed by  $(SP+2_{n-1})$  are restored to the high-order side.

Flag operation: Unchanged

Coding example:

## POP PSW

Function:  $PSW_L + (SP), PSW_H + (SP+1), SP + SP+2$

Restores the contents of the memory (stack) addressed by the stack pointer (SP) to the program status word (PSW) and decrements the SP.

Flag operation:

S	Z	AC	P/V	SUB	CY
R	R	R	R	R	R

Coding example:

## POPU post

Function:  $\{post_L + (UP), post_H + (UP+1), UP + UP+2\} \times n$  times

(n; number of register pairs written as post)

Restores the contents of the memory (stack) addressed by the user stack pointer (UP) to the register pair specified by operand and increments the UP.

Multiple register pair names can be specified in the post operand.

Transfer is performed sequentially from the register pair assigned to bit 0 of the 8-bit immediate data of the second byte (Post Byte). The contents of the stack addressed by  $(UP+2_{n-2})$  are restored to the low-order side of the register pair and the contents of the stack addressed by  $(UP+2_{n-1})$  are restored to the high-order side.

Flag operation: Unchanged

Coding example:

## MOVW SP, #word

Function:  $SP + word$

word = 0000H to FDFEH  
(enable to any data)  
word = FE00H to FFFE H  
(limited to even data)

Transfers the 16-bit immediate data specified by the second operand to the stack pointer (SP).

Flag operation: Unchanged

Coding example:

**MOVW SP, AX**

**Function: SP + AX**

Transfers the contents of the 16-bit register pair AX to the stack pointer (SP). Any data is possible when the contents of AX are 0000H to FDFEH, but only even data is possible when the contents are FE00H to FFFEH.

**Flag operation: Unchanged**

**Coding example:**

**MOVW AX, SP**

**Function: AX + SP**

Transfers the contents of the stack pointer (SP) to the 16-bit register pair AX.

**Flag operation: Unchanged**

**Coding example:**

**INCW SP**

**Function: SP + SP+1**

Increments the contents of the stack pointer (SP).

**Flag operation: Unchanged**

**Coding example:**

**DECW SP**

**Function: SP + SP-1**

Decrements the contents of the stack pointer (SP).

**Flag operation: Unchanged**

**Coding example:**

## 10.6.12 UNCONDITIONAL BRANCH INSTRUCTIONS

**BR !addr16**

**Function:** PC + addr16                      addr16 = 0000H to FEFFH

Transfers the 16-bit immediate data specified by the operand to the program counter (PC) and branches to the location addressed by the PC.

Branching to memory addresses 0000H to FEFFH is possible.

**Flag operation:** Unchanged

**NOTE:** Since addresses FF00H to FFFFH cannot be instruction-fetched, do not write them at addr16.

**Coding example:** BR BLK3; Branch to the address specified specified by label BLK3.

**BR rpl**

**Function:** PC<sub>H</sub> + rpl<sub>H</sub>, PC<sub>L</sub> + rpl<sub>L</sub>

Transfers the contents of the 16-bit register pair specified by the operand to the program counter (PC) and branches to the location addressed by the PC.

Branching to memory addresses 0000H to FEFFH is possible.

**Flag operation:** Unchanged

**NOTE:** Since addresses FF00H to FFFFH cannot be instruction-fetched, do not set them are rpl.

**Coding example:**



### 10.6.13 CONDITIONAL BRANCH INSTRUCTIONS

BC \$addr16  
BL \$addr16

Function:  $PC + PC+2+jdisp$  if  $CY = 1$

addr16 = (PC-126) to  
(PC+129)

When the carry flag is 1, transfers the sum of the 8-bit displacement value *jdisp* of the second byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

*jdisp* is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the *addr16* operand by label or numeric.

Flag operation: Unchanged

Coding example:

BNC \$addr16  
BNL \$addr16

Function:  $PC + PC+2+jdisp$  if  $CY = 0$

addr16 = (PC-126) to  
(PC+129)

When the carry flag is 0, transfers the sum of the 8-bit displacement value *jdisp* of the second byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

*jdisp* is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly the *addr16* operand by label or numeric.

Flag operation: Unchanged

Coding example:

BZ \$addr16  
BE \$addr16

Function:  $PC + PC+2+jdisp$  if  $Z = 1$

addr16 = (PC-126) to  
(PC+129)

When the zero flag is 1, transfers the sum of the 8-bit displacement value *jdisp* of the second byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

*jdisp* is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination directly in the *addr16* operand by label or numeric.

Flag operation: Unchanged

Coding example: DEC OFE20H  
BZ \$JMP ; Decrement by 1 the contents of the memory addressed by FE20H and when the contents reach 0, branch to the address indicated by the label JMP. (However, the branch destination is limited to the (-128 to +127) range from the start address of the next instruction.)



BNZ \$addr16  
BNE \$addr16

Function:  $PC + PC+2+jdisp$  if  $Z = 0$

addr16 = (PC-126) to  
(PC+129)

When the zero flag is 0, transfers the sum of the 8-bit displacement value *jdisp* of the second byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

*jdisp* is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and write the branch destination address directly in the addr16 operand by label or numeric.

Flag operation: Unchanged

Coding example:

BV \$addr16  
BPE \$addr16

Function:  $PC + PC+2+jdisp$  if  $P/V = 1$

addr16 = (PC-126) to  
(PC+129)

When the parity/overflow flag is 1, transfers the sum of the 8-bit displacement value *jdisp* of the second byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

*jdisp* is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the addr16 operand by label or numeric.

Flag operation: Unchanged

Coding example:

BNV \$addr16  
BPO \$addr16

Function:  $PC + PC+2+jdisp$  if  $P/V = 0$

$addr16 = (PC-126)$  to  
 $(PC+129)$

When the parity/overflow flag is 0, transfers the sum of the 8-bit displacement value *jdisp* of the second byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

*jdisp* is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and write the branch destination address directly in the *addr16* operand by label or numeric.

Flag operation: Unchanged

Coding example:

BN \$addr16

Function:  $PC + PC+2+jdisp$  if  $S = 1$

$addr16 = (PC-126)$  to  
 $(PC+129)$

When the sign flag is 1, transfers the sum of the 8-bit displacement value *jdisp* of the second byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addresses by the PC.

*jdisp* is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and written the branch destination address directly in the *addr16* operand by label or numeric.

Flag operation: Unchanged

Coding example:

### BP \$addr16

Function:  $PC + PC+2+jdisp$  if  $S = 0$

$addr16 = (PC-126)$  to  
 $(PC+129)$

When the sign flag is 0, transfers the sum of the 8-bit displacement value  $jdisp$  of the second byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

$jdisp$  is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and written the branch destination address directly in the  $addr16$  operand by label or numeric.

Flag operation: Unchanged

Coding example:

### BGT \$addr16

Function:  $PC + PC+3+jdisp$  if  $(P/V\psi S)VZ = 0$

$addr16 = (PC-125)$  to  
 $(PC+130)$

EXCLUSIVE-ORS the contents of the parity/overflow flag and the contents of the sign flag and when the OR of the result and the contents of the zero flag is 0, transfers the sum of the 8-bit displacement value  $jdisp$  of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

$jdisp$  is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and write the branch destination address directly in the  $addr16$  operand by label or numeric.

Flag operation: Unchanged

Coding example: `CMP A, #0FH;` If the two's complement data  
`BGT $MR2` in the A register is larger  
than FH, branch to the  
address specified by the  
label MR2.

BGE \$addr16

Function:  $PC + PC+3+jdisp$  if  $P/V\neq S = 0$

addr16 = (PC-125) to  
(PC+130)

EXCLUSIVE-ORs the contents of the parity/overflow flag and the contents of the sign flag and when the result is 0, transfers the sum of the 8-bit displacement value *jdisp* of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

*jdisp* is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the *addr16* operand by label or numeric.

Flag operation: Unchanged

Coding example:

BLT \$addr16

Function:  $PC + PC+3+jdisp$  if  $P/V\neq S = 1$

addr16 = (PC-125) to  
(PC+130)

EXCLUSIVE-ORs the contents of the parity/overflow flag and the contents of the sign flag and when the result is 1, transfers the sum of the 8-bit displacement value *jdisp* of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

*jdisp* is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the *addr16* operand by label or numeric.

Flag operation: Unchanged

Coding example:

## BNH \$addr16

Function:  $PC + PC+3+jdisp$  if ZVCY = 1

addr16 = (PC-125) to  
(PC+130)

ORs the contents of the zero flag and the contents of the carry flag and when the result is 1, transfers the sum of the 8-bit displacement value *jdisp* of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

*jdisp* is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the *addr16* operand by label or numeric.

Flag operation: Unchanged

Coding example:

## BT *saddr.bit*, \$addr16

Function:  $PC + PC+3+jdisp$  if (*saddr.bit*) = 1

*addr16* = (PC-125) to  
(PC+130)  
*saddr* = FE20H to FF1FH  
*bit* = 0 to 7

When the contents of the short direct memory bit addressed by the first operand are 1, transfers the sum of the 8-bit displacement value *jdisp* of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

*jdisp* is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Specify the short direct memory bit address or label directly at *saddr.bit* of the first operand and consider the branch range and specify the branch destination directly in the second operand *addr16* by label or numeric.

Flag operation: Unchanged

Coding example:

BLE \$addr16

Function:  $PC + PC+3+jdisp$  if  $(P/V\checkmark S)VZ = 1$

addr16 = (PC-125) to  
(PC+130)

EXCLUSIVE-ORs the contents of the parity/overflow flag and the contents of the sign flag and when the OR of the result and the contents of the zero flag is 1, transfers the sum of the 8-bit displacement value *jdisp* of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

*jdisp* is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the *addr16* operand by label or numeric.

Flag operation: Unchanged

Coding example:

BH \$addr16

Function:  $PC + PC+3+jdisp$  if  $ZVCY = 0$

addr16 = (PC-125) to  
(PC+130)

ORs the contents of the zero flag and the contents of the carry flag and when the result is 0, transfers the sum of the 8-bit displacement value of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

*jdisp* is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the *addr16* operand by label or numeric.

Flag operation: Unchanged

Coding example:

BT sfr.bit, \$addr16

Function: PC + PC+4+jdisp if sfr.bit = 1

addr16 = (PC-124) to  
(PC+131)  
bit = 0 to 7

When the contents of the bit addressed by the 3-bit immediate data of the first operand of the special function register specified by the first operand are 1, transfers the sum of the 8-bit displacement value jdisp of the fourth bit of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the program counter.

jdisp is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the second operand addr16 by label or numeric.

Flag operation: Unchanged

Coding example:

BT A.bit, \$addr16

Function: PC + PC+3+jdisp if A.bit = 1

addr16 = (PC-125) to  
(PC+130)  
bit = 0 to 7

When the contents of the A register bit addressed by the 3-bit immediate data of the first operand are 1, transfers the sum of the 8-bit displacement value jdisp of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

jdisp is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand addr16 by label or numeric.

Flag operation: Unchanged

Coding example: BT A.3, ; If A register bit 3 is "1",  
\$JMP1 branch to the address specified  
by the label JMP1.



BT X.bit, \$addr16

Function:  $PC + PC+3+jdisp$  if X.bit = 1

addr16 = (PC-125) to  
(PC+130)  
bit = 0 to 7

When the contents of the X register bit addressed by the 3-bit immediate data of the first operand are 1, transfers the sum of the 8-bit displacement value jdisp of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

jdisp is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand addr16 by label or numeric.

Flag operation: Unchanged

Coding example:

BT PSWH.bit, \$addr16

Function:  $PC + PC+3+jdisp$  if PSWH bit = 1

addr16 = (PC-125) to  
(PC+130)  
bit = 0 to 7

When the contents of the bit addressed by the 3-bit immediate data of the first operand of the high-order 8 bits of the program status word are 1, transfers the sum of the 8-bit displacement value jdisp of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

jdisp is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand addr16 by label or numeric.

Flag operation: Unchanged

Coding example:



BT PSWL.bit, \$addr16

Function:  $PC \leftarrow PC+3+jdisp$  if PSWL.bit = 1

addr16 = (PC-125) to  
(PC+130)  
bit = 0 to 7

When the contents of the bit addressed by the 3-bit immediate data of the first operand of the low-order 8 bits of the program status word are 1, transfers the sum of the 8-bit displacement value *jdisp* of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

*jdisp* is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand *addr16* by label or numeric.

Flag operation: Unchanged

Coding example:

BF saddr.bit, \$addr16

Function:  $PC + PC+4+jdisp$  if (saddr.bit) = 0

addr16 = (PC-124) to  
(PC+131)  
saddr = FE20H to FF1FH  
bit = 0 to 7

When the contents of the short direct memory bit addressed by the first operand are 0, transfers the sum of the 8-bit displacement value of the fourth byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

jdisp is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Specify the short direct memory bit address or label directly at saddr.bit of the first operand and consider the branch range and specify the branch destination address directly in the second operand addr16 by label or numeric.

Flag operation: Unchanged

Coding example:

BF sfr.bit, \$addr16

Function:  $PC + PC+4+jdisp$  if sfr.bit = 0

addr16 = (PC-124) to  
(PC+131)  
bit = 0 to 7

When the contents of the bit addressed by the 3-bit immediate data of the first operand of the special function register specified by the first operand are 0, transfers the sum of the 8-bit displacement value of the fourth byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

jdisp is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand addr16 by label or numeric.

Flag operation: Unchanged

Coding example:

BF A.bit \$addr16

Function:  $PC + PC+3+jdisp$  if A.bit = 0

addr16 = (PC-125) to  
(PC+130)  
bit = 0 to 7

When the contents of the A register bit addressed by the 3-bit immediate data of the first operand are 0, transfers the sum of the 8-bit displacement value *jdisp* of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

*jdisp* is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand *addr16* by label or numeric.

Flag operation: Unchanged

Coding example:

BF X.bit, \$addr16

Function:  $PC + PC+3+jdisp$  if X.bit = 0

addr16 = (PC-125) to  
(PC+130)  
bit = 0 to 7

When the contents of the X register bit addressed by the 3-bit immediate data of the first operand are 0, transfers the sum of the 8-bit displacement value *jdisp* of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

*jdisp* is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand *addr16* by label or numeric.

Flag operation: Unchanged

Coding example:

BF PSWH.bit, \$addr16

Function:  $PC + PC+3+jdisp$  if PSWH.bit = 0

addr16 = (PC-125) to  
(PC+130)  
bit = 0 to 7

When the contents of the bit addressed by the 3-bit immediate data of the first operand of the high-order 8 bits of the program status word are 0, transfers the sum of the 3-bit displacement value jdisp of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

jdisp is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand addr16 by label or numeric.

Flag operation: Unchanged

Coding example:

BF PSWL.bit, \$addr16

Function:  $PC + PC+3+jdisp$  if PSWL.bit = 0

addr16 = (PC-125) to  
(PC+130)  
bit = 0 to 7

When the contents of the bit addressed by the 3-bit immediate data of the first operand of the low-order 8 bits of the program status word are 0, transfers the sum of the 8-bit displacement value jdisp of third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

jdisp is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand addr16 by label or numeric.

Flag operation: Unchanged

Coding example:

BTCLR saddr.bit \$addr16

Function:  $PC \leftarrow PC+4+jdisp$  if (saddr.bit) = 1 then clear

addr16 = (PC-124) to  
(PC+131)

saddr = FE20H to FF1FH

bit = 0 to 7

When the contents of the short direct memory bit addressed by the first operand are 1, transfers the sum of the 8-bit displacement value jdisp of the fourth byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC.

jdisp is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Specify the short direct memory bit address or label directly at saddr.bit of the first operand and consider the branch range and specify the branch destination address directly in the second operand addr16 by label or numeric.

Flag operation: Unchanged

Coding example:

BTCLR sfr.bit, \$saddr16

Function:  $PC + PC+4+jdisp$  if sfr.bit = 1 then clear

addr16 = (PC-124) to  
(PC+131)  
bit = 0 to 7

When the contents of the bit addressed by the 3-bit immediate data of the first operand of the special function register specified by the first operand are 1, transfers the sum of the 8-bit displacement value jdisp of the fourth byte of the OP code and the start address of the next address to the program counter (PC) and branches to the location addressed by the program counter and clears that bit to 0.

jdisp is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand addr16 by label or numeric.

Flag operation: Unchanged

Coding example:

BTCLR A.bit, \$addr16

Function:  $PC + PC+3+jdisp$  if A.bit = 1 then clear

addr16 = (PC-125) to  
(PC+130)  
bit = 0 to 7

When the contents of the A register bit addressed by the 3-bit immediate data of the first operand are 1, transfers the sum of the 8-bit displacement value jdisp of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC and clears that bit to 0.

jdisp is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand addr16 by label or numeric.

Flag operation: Unchanged

Coding example:

BTCLR X.bit, \$addr16

Function:  $PC + PC+3+jdisp$  if X.bit = 1 then clear

addr16 = (PC-125) to  
(PC+130)  
bit = 0 to 7

When the contents of the X register bit addressed by the 3-bit immediate data of the first operand are 1, transfers the sum of the 8-bit displacement value of the third byte of the OP code and the start address of the next address to the program counter (PC) and branches to the location addressed by the PC and clears that bit to 0.

jdisp is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand addr16 by label or numeric.

Flag operation: Unchanged

Coding example:

BTCLR PSWH.bit, \$addr16

Function:  $PC + PC+3+jdisp$  if PSWH.bit = 1 then clear

addr16 = (PC-125) to  
(PC+130)  
bit = 0 to 7

When the contents of the bit addressed by the 3-bit immediate data of the first operand of the high-order 8 bits of the program status word are 1, transfers the sum of the 8-bit displacement value jdisp of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the program counter and clears that bit to 0.

jdisp is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand addr16 by label or numeric.

Flag operation: Unchanged

Coding example:

BTCLR PSWL.bit, \$addr16

Function: PC + PC+3+jdisp if PSWL.bit = 1 then clear

addr16 = (PC-125) to  
(PC+130)  
bit = 0 to 7

When the contents of the bit addressed by the 3-bit immediate data of the first operand of the low-order 8 bits of the program status word are 1, transfers the sum of the 8-bit displacement value jdisp of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the program counter and clears that bit to 0.

jdisp is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand addr16 by label or numeric.

Flag operation: If the specified flag is "1", it is reset (0).

Coding example: BTCLR PSWL.3, \$OF6EH; If the UF flag is "1", reset (0) and branch to address F6EH.



BFSET saddr.bit, \$addr16

Function: PC + PC+4+jdisp if (saddr.bit) = 0 then set

addr16 = (PC-124) to  
(PC+131)  
saddr = FE20H to FF1FH  
bit = 0 to 7

When the contents of the short direct memory bit addressed by the first operand are 0, transfers the sum of the 8-bit displacement value jdisp of the fourth byte of the OP code and the start address of the next address to the program counter (PC) and branches to the location addressed by the PC and sets the addressed bit to 1.

jdisp is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand addr16 by label or numeric.

Flag operation: Unchanged

Coding example:

BFSET sfr.bit, \$addr16

Function: PC + PC+4+jdisp if sfr.bit = 0 then set

addr16 = (PC-124) to  
(PC+131)  
bit = 0 to 7

When the contents of the bit addressed by the 3-bit immediate data of the first operand of the special function register specified by the first operand are 0, transfers the sum of the 8-bit bit displacement value jdisp of the fourth byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the program counter and sets that bit to 1.

jdisp is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand addr16 by label or numeric.

Flag operation: Unchanged

Coding example:

BFSET A.bit, \$addr16

Function: PC + PC+3+jdisp if A.bit = 0 then set

addr16 = (PC-125) to  
(PC+130)  
bit = 0 to 7

When the contents of the A register bit addressed by the 3-bit immediate data of the first operand are 0, transfers the sum of the 8-bit displacement value of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC and sets the addressed bit to 1.

jdisp is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand addr16 by label or numeric.

Flag operation: Unchanged

Coding example:

BFSET X.bit, \$addr16

Function: PC + PC+3+jdisp if X.bit = 0 then set

addr16 = (PC-125) to  
(PC+130)  
bit = 0 to 7

When the contents of the X register bit addressed by the 3-bit immediate data of the first operand are 0, transfers the sum of the 8-bit displacement value jdisp of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC and sets the addressed bit to 1.

jdisp is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand addr16 by label or numeric.

Flag operation: Unchanged

Coding example:

DBNZ r2, \$addr16

Function:  $r2 + r2 - 1$ , then  $PC + PC + 2 + jdisp$  if  $r2 \neq 0$

addr16 = (PC-126) to  
(PC+129)  
bit = 0 to 7

Decrements the contents of the 8-bit register specified by the first operand and if the result is not 0, transfers the sum of the 8-bit displacement value *jdisp* of the second byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the program counter.

*jdisp* is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand *addr16* by label or numeric.

Flag operation: Unchanged

Coding example:

DBNZ saddr, \$addr16

Function:  $(saddr) + (saddr) - 1$ , then  $PC + PC + 3 + jdisp$  if  
 $(saddr) \neq 0$

addr16 = (PC-125) to  
(PC+130)  
saddr = FE20H to FF1FH

Decrements the contents of the short direct memory addressed by the first operand and if the result is not 0, transfers the sum of the 8-bit displacement value *jdisp* of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the program counter.

*jdisp* is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand *addr16* by label or numeric.

Flag operation: Unchanged

Coding example:

BFSET PSWH.bit, \$addr16

Function: PC + PC+3+jdisp if PSWH.bit = 0 then set

addr16 = (PC-125) to  
(PC+130)  
bit = 0 to 7

When the contents of the bit specified by the 3-bit immediate data of the first operand of the high-order 8 bits of the program status word are 0, transfers the contents of the 8-bit displacement value jdisp of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC and sets the addressed bit to 1.

jdisp is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand addr16 by label or numeric.

Flag operation: Unchanged

Coding example:

BFSET PSWL.bit, \$addr16

Function: PC + PC+3+jdisp if PSWL.bit = 0 then set

addr16 = (PC-125) to  
(PC+130)  
bit = 0 to 7

When the contents of the bit specified by the 3-bit immediate data of the first operand of the low-order 8 bits of the program status word are 0, transfers the contents of the 8-bit displacement value jdisp of the third byte of the OP code and the start address of the next instruction to the program counter (PC) and branches to the location addressed by the PC and sets the addressed bit to 1.

jdisp is treated as signed two's complement data (-128 to +127). Bit 7 is the sign bit.

Consider the branch range and specify the branch destination address directly in the operand addr16 by label or numeric.

Flag operation: If the specified flag is "0", it is set (1).

Coding example:

RETCS laddr16

Function:  $PC_H + R5$ ,  $PC_L + R4$ ,  $R5 + addr16_H$ ,  $R4 + addr16_L$ ,  
 $PSW_H + R7$ ,  $PSW_L + R6$ ,  $EOS + 0$

addr16 = 0000H to FFFFH

Transfers the contents of 8-bit registers R7, R6, R5, R4 in the register bank specified at execution of this instruction to each program status word (PSW) and the program counter (PC) and returns to the address set at R5, R4. The 16-bit immediate data specified by the operand is then transferred to R5, R4 and the EOS flag is cleared to 0.

The RETCS instruction is used when returning from branch processing by context switching. addr16 specified at the operand becomes the branch destination address when the same register bank was specified again by the context switch function.

NOTE: At return from an interrupt by BRKCS instruction, always set the EOS flag by SET1 EOS instruction immediately before the RETCS instruction.

Flag operation:

S	Z	AC	P/V	SUB	CY
R	R	R	R	R	R

NOTE: Since addresses F000H to FFFFH cannot be instruction-fetched, do not specify them at addr16.

Coding example:

#### 10.6.14 CONTEXT SWITCH INSTRUCTIONS

BRKCS R<sub>n</sub>

Function: RBS2 to RBS0 + n, PC<sub>H</sub> ↔ R5, PC<sub>L</sub> ↔ R4,  
R7 + PSW<sub>H</sub>, R6 + PSW<sub>L</sub>, RSS + 0, IE + 0

n = 0 to 7

Sets 3-bit immediate data N<sub>2</sub> to N<sub>0</sub> in the OP code into the register bank select flag (RBS2 to RBS0) and selects the register bank n specified in the operand and swaps the contents of 8-bit registers R5, R4 of that register bank and the contents of the program counter (PC) and saves the contents of the program status word (PSW) to 8-bit registers R7, R6 and branches to the address set at R5, R4 and then clears the RSS flag and IE flag.

Flag operation: Unchanged

Coding example:

### 10.6.15 STRING INSTRUCTIONS

MOVM [DE+], A  
MOVM [DE-], A

Function:  $\{(DE) + A, DE + DE+1/-1, C + C-1\}$

End if C = 0

Transfers the contents of the A register to the memory addressed by register pair DE and increments/decrements the contents of register pair DE. Then it decrements the contents of the C register and repeats the operation above until the contents of the C register reach 0.

Flag operation: Unchanged

Coding example: MOV R2, #00H ; C + 00H  
MOV R1, #00H ; A + 00H  
MOVW RP6, #FE00H; DE + FE00H  
MOVM [DE+], A ; Clear the FE00H to  
FEFFH RAM.

MOVBK [DE+], [HL+]  
MOVBK [DE-], [HL-]

Function:  $\{(DE) + (HL), DE + DE+1/-1, HL + HL+1/-1, C + C-1\}$

End if C = 0

Transfers the contents of memory addressed by register pair HL to the memory addressed by register pair DE and increments/decrements the contents of register pairs DE and HL. Then it decrements the contents of the C register and repeats the operation above until the contents of the C register reach 0.

Flag operation: Unchanged

Coding example: MOV R2, #10H ; C + 10H  
MOVW RP6, #3000H ; DE + 3000H  
MOVW RP7, #5000H ; HL + 5000H  
MOVBK [DE+], [HL+]; Transfer the  
contents of memory  
5000H to 500FH to  
memory 3000H to  
300FH.

XCHM [DE+], A  
XCHM [DE-], A

Function:  $\{(DE) \leftrightarrow A, DE \pm DE+1/-1, C \pm C-1\}$

End if C = 0

Swaps the contents of the A register and the contents of the memory addressed by register pair DE and increments/decrements the contents of register pair DE. Then it decrements the contents of the C register and repeats the operation above until the contents of the C register reach 0.

Flag operation: Unchanged

Coding example: MOV R2, #10H ; C + 10H

MOV R1, #00H ; A + 00H

MOVW RP6, #3050H; DE + 3050H

XCHM [DE+], A ; Shift the contents of memory 3050H to 305FH back one address at the time. (Contents of address 3050H become 0.)

XCHBK [DE+], [HL+]  
XCHBK [DE-], [HL-]

Function:  $\{(DE) \leftrightarrow (HL), DE \pm DE+1/-1, HL \pm HL+1/-1, C \pm C-1\}$

End if C = 0

Swaps the contents of the memory addressed by register pair HL and the contents of memory addressed by register pair DE and increments/decrements the contents of register pairs DE and HL. Then it decrements the contents of the C register and repeats the operation above until the contents of the C register reach 0.

Flag operation: Unchanged

Coding example:



CMPME [DE+], A  
CMPME [DE-], A

Function:  $\{(DE)-A, DE + DE+1/-1, C + C-1\}$

End if  $C = 0$  or  $Z = 0$

Compares the contents of the A register and the contents of the memory addressed by register pair DE and increments/decrements the contents of register pair DE and decrements the contents of the C register. This operation is repeated until the result of comparison is not a match or the contents of the C register become 0.

The contents of the A register and the contents of the memory address of register pair DE are not changed by execution of this instruction.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

CMPBKE [DE+], [HL+]  
CMPBKE [DE-], [HL-]

Function:  $\{(DE)-(HL), DE + DE+1/-1, HL + HL+1/-1, C + C-1\}$

End if  $C = 0$  or  $Z = 0$

Compares the contents of the memory addressed by register pair HL and the contents of the memory addressed by the register pair DE and increments/decrements the contents of register pairs DE and HL and decrements the contents of the C register. This operation is repeated until the result of comparison is not a match or the contents of the C register become 0.

The contents of the memory addressed by register pairs DE and HL are not changed by execution of this instruction.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

CMPMNE [DE+], A  
CMPMNE [DE-], A

Function:  $\{(DE)-A, DE + DE+1/-1, C + C-1\}$

End if  $C = 0$  or  $Z = 1$

Compares the contents of the A register and the contents of the memory addressed by register pair DE and increments/decrements the contents of register pair DE and decrements the contents of the C register. This operation is repeated until the result of comparison is a match or the contents of the C register become 0.

The contents of the A register and the contents of the memory addressed by register pair DE are not changed by execution of this instruction.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example: MOV R2, #00H ; C + 00H

MOVW RP6, #3000H; DE + 3000H

CMPMNE [DE+], A ; If the contents of  
BZ \$JMP 3000H to 30FFH are the  
same value as the A  
registers, branch to  
the address indicated  
by the label JMP.

CMPBKNE [DE+], [HL+]  
 CMPBKNE [DE-], [HL-]

Function:  $\{(DE)-(HL), DE + DE+1/-1, HL + HL+1/-1,$   
 $C + C-1\}$

End if C = 0 or Z = 1

Compares the contents of the memory addressed by register pair HL and the contents of the memory addressed by register pair DE and increments/decrements the contents of register pairs DE and HL and decrements the contents of the C register. This operation is repeated until the result of comparison is a match or the contents of the C register become 0.

The contents of the memory addressed by register pairs DE and HL are not changed by execution of this instruction.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

CMPMC [DE+], A  
 CMPMC [DE-], A

Function:  $\{(DE)-A, DE + DE+1/-1, C + C-1\}$

End if C = 0 or CY = 0

Compares the contents of the A register and the contents of the memory addressed by pair register DE and increments/decrements the contents of register pair DE and decrements the contents of the C register. This operation is repeated until the result of comparison is that the contents of the memory addressed by register pair DE are larger or the contents of the C register become 0.

The contents of the A register and the contents of the memory addressed by register pair are not changed by execution of this instruction.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

CMPBKC [DE+], [HL+]  
CMPBKC [DE-], [HL-]

Function:  $\{(DE)-(HL), DE \pm DE+1/-1, HL \pm HL+1/-1,$   
 $C \pm C-1\}$

End if  $C = 0$  or  $CY = 0$

Compares the contents of the memory addressed by register pair HL and the contents of the memory addressed by register pair DE and increments/decrements the contents of register pairs DE and HL and decrements the contents of the C register. This operation is repeated until the result of comparison is that the contents of the memory addressed by register pair DE are larger or the contents of the C register become 0.

The contents of the memory addressed by register pairs DE and HL are not changed by execution of this instruction.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

```
CMPMNC [DE+], A
CMPMNC [DE-], A
```

Function:  $\{(DE)-A, DE + DE+1/-1, C + C-1\}$

End if  $C = 0$  or  $CY = 1$

Compares the contents of the A register and the contents of the memory addressed by register pair DE and increments/decrements the contents of register pair DE and decrements the contents of the C register. This operation is repeated until the result of comparison is that the contents of the A register are larger or the contents of the C register become 0.

The contents of the A register and the contents of the memory addressed by register pair DE are not changed by execution of this instruction.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

```
Coding example: MOV    R2, #00H    ; C + 00H
                MOVW   RP6, #8000H; DE + 8000H
                CLR1   CY          ; CY + 0
                CMPMNC [DE+], A
                BC     $JMP        ; If the value of 8000H
                                   to 80FFH is larger
                                   than the contents of
                                   the A register, jump
                                   to the address
                                   indicated by the label
                                   JMP.
```

CMPBKNC [DE+], [HL+]  
CMPBKNC [DE-], [HL-]

Function:  $\{(DE)-(HL), DE \pm DE+1/-1, HL \pm HL+1/-1,$   
 $C \pm C-1\}$

End if  $C = 0$  or  $CY = 1$

Compares the contents of the memory addressed by register pair HL and the contents of the memory addressed by register pair DE and increments/decrements the contents of register pairs DE and HL and decrements the contents of the C register. This operation is repeated until the result of comparison is that the contents of the memory addressed by register pair HL are larger or the contents of the C register become 0.

The contents of the memory addressed by register pairs HL and DE are not changed by execution of this instruction.

Flag operation:

S	Z	AC	P/V	SUB	CY
x	x	x	V	1	x

Coding example:

### SEL R<sub>n</sub>, ALT

Function: RBS2 to RBS0 + n, RSS + 1 n = 0 to 7

Selects the register bank specified at the operand by setting the 3-bit immediate data N<sub>2</sub> to N<sub>0</sub> in the OP code into the register bank select flag (RBS2 to RBS0) and sets the register set selection flag (1).

Flag operation: Unchanged

Coding example:

### NOP

Function:

Expend three states without performing any operation.

Flag operation: Unchanged

Coding example:

### EI

Function: IE + 1

Sets (1) the interrupt request enable flag (IE). Reception of maskable interrupts is controlled by each interrupt request control register.

Flag operation: Unchanged

Coding example:

### DI

Function: IE + 0

Clears the interrupt request enable flag (IE). Reception of maskable interrupts is disabled.

Flag operation: Unchanged

Coding example:

## 10.6.16 CPU CONTROL INSTRUCTIONS

**MOV STBC, #byte**

**Function:** STBC + byte                      byte = 00H to FFH

Sets the 8-bit immediate data specified by the second operand into the standby control register (STBC).

This instruction is a special OP code for setting the STBC register.

**Flag operation:** Unchanged

**Coding example:** MOV STBC, #01H; Set the HALT mode.

**MOV WDM, #byte**

**Function:** WDM + byte                      byte = 00H to FFH

Sets the 8-bit immediate data specified by the second operand into the watchdog timer mode register (WDM).

This instruction is a special OP code for setting the WDM register.

**Flag operation:** Unchanged

**Coding example:**

**SWRS**

**Function:** RSS +  $\overline{\text{RSS}}$

Inverts the contents of the register set flag (RSS).

**Flag operation:** Unchanged

**Coding example:**

**SEL RBn**

**Function:** RBS2 to RBS0 + n, RSS + 0    n = 0 to 7

Selects the register bank specified at the operand by setting the 3-bit immediate data  $N_2$  to  $N_0$  in the OP code into the register bank select flag (RBS2 to RBS0) and clears (0) the register set selection flag (RSS).

**Flag operation:** Unchanged

**Coding example:**



## CHAPTER 11. SPECIFICATIONS

### 11.1 ELECTRICAL SPECIFICATIONS (uPD78310A, uPD78312A)

Absolute Maximum Ratings ( $T_a = 25^{\circ}\text{C}$ )

Parameter	Symbol	Test Conditions	Rating	Unit
Power supply voltage	$V_{DD}$		-0.5 to +7.0	V
	$AV_{REF}$		-0.5 to $V_{DD}$ +0.3	V
	$AV_{SS}$		-0.5 to +0.5	V
Input voltage	$V_I$		-0.5 to $+V_{DD}$ +0.5	V
Output voltage	$V_O$		-0.5 to $+V_{DD}$ +0.5	V
Output current low	$I_{OL}$	1 pin	4.0	mA
		All output pins total	100	mA
Output current high	$I_{OH}$	1 pin	-2	mA
		All output pins total	-25	mA
Operating temperature	$T_{opt}$		-10 to +70	$^{\circ}\text{C}$
Storage temperature	$T_{stg}$		-65 to +150	$^{\circ}\text{C}$

#### Recommended Operating Conditions

Oscillation Frequency	Parameter	$T_a$	$V_{DD}$
$4 \text{ MHz} \leq f_{XX} \leq 12 \text{ MHz}$		-10 to $+70^{\circ}\text{C}$	+5.0 V $\pm 10\%$

Capacitance ( $T_a = 25^{\circ}\text{C}$ ,  $V_{DD} = V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions	MIN.	TYP.	MAX.	Unit
Input capacitance	$C_I$	$f = 1\text{ MHz}$			10	pF
Output capacitance	$C_O$	Unmeasured pins returned to 0 V			20	pF
I/O capacitance	$C_{IO}$				20	pF

Oscillator Characteristics ( $T_a = -10$  to  $+70^{\circ}\text{C}$ ,  $V_{DD} = +5.0\text{ V} \pm 10\%$ ,  $V_{SS} = AV_{SS} = 0\text{ V}$ ,  $4.0\text{ V} \leq AV_{REF} \leq V_{DD}$ )

Oscillator	Recommended Circuit	Parameter	MIN.	TYP.	MAX.	Unit
Ceramic or crystal resonator		Oscillation frequency ( $f_{XX}$ )	4		12	MHz
External clock		X1 input frequency ( $f_X$ )	4		12	MHz
		X1 input rise and fall times ( $t_{XR}$ , $t_{XF}$ )	0		30	ns
		X1 input high, low-level width ( $t_{WXH}$ , $t_{WXL}$ )	30		130	ns

NOTE 1: Place the oscillator circuit as close to the X1, X2 pins as possible.

2: Do not pass other signals within a range of dotted area.

## Recommended Oscillator Constants

### Ceramic Resonator

Manufacturer	Product Name	Frequency [MHz]	External Capacitance [pF]	
			C1	C2
Murata Mfg. Co., Ltd.	CSA8.00MT CSA10.0MT CSA12.0MT	8.0 10.0 12.0	30	30
	CST10.0MT CST10.0MT CST12.0MT	8.0 10.0 12.0	On-chip	On-chip
Kyocera Corp.	KBR-8.0M KBR-10.0M KBR-12.0M	8.0 10.0 12.0	33	33
TDK	FCR10.0MC FCR12.0MC	10.0 12.0	On-chip	On-chip

### Crystal Resonator

Manufacturer	Product Name	Frequency [MHz]	External Capacitance [pF]	
			C1	C2
Kinseki Ltd.	HC-49U	8.0 10.0 12.0	22	22

DC Characteristics ( $T_a = -10$  to  $+70^{\circ}\text{C}$ ,  $V_{DD} = +5.0\text{ V} \pm 10\%$ ,  
 $V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions	MIN.	TYP.	MAX.	Unit	
Input voltage low	$V_{IL1}$	Except for $\overline{EA}$	0		0.8	V	
	$V_{IL2}$	$\overline{EA}$	0		0.5	V	
Input voltage high	$V_{IH1}$	Except for P20/NMI, X1, X2, $\overline{RESET}$	2.2		$V_{DD}$	V	
	$V_{IH2}$	P20/NMI, X1, X2, $\overline{RESET}$	3.8		$V_{DD}$	V	
Output voltage low	$V_{OL}$	$I_{OL} = 2.0\text{ mA}$			0.45	V	
Output voltage high	$V_{OH}$	$I_{OH} = -1.0\text{ mA}$	$V_{DD} - 1$			V	
Input current	$I_I$	P20/NMI, $\overline{RESET}$ $0.45\text{ V} < V_I < V_{DD}$			$\pm 10$	$\mu\text{A}$	
Input leakage current	$I_{LI}$				$\pm 10$	$\mu\text{A}$	
I/O leakage current	$I_{LO}$				$\pm 10$	$\mu\text{A}$	
$AV_{REF}$ current	$AI_{REF}$	$f_{CLK} = 6\text{ MHz}$		1.5	5	mA	
$V_{DD}$ power supply current	$I_{DD1}$	Operating mode, $f_{CLK} = 6\text{ MHz}$		30	60	mA	
	$I_{DD2}$	HALT mode, $f_{CLK} = 6\text{ MHz}$		5	15	mA	
Data hold voltage	$V_{DDDR}$	STOP mode	2.5			V	
Data hold current	$I_{DDDR}$	STOP mode	$V_{DDDR} = 2.5\text{ V}$		3	15	$\mu\text{A}$
			$V_{DDDR} = 5.0\text{ V} \pm 10\%$		10	50	$\mu\text{A}$

## AC Characteristics

Read/Write Operation ( $T_a = -10$  to  $+70^\circ\text{C}$ ,  $V_{DD} = +5.0$  V  
 $\pm 10\%$ ,  $V_{SS} = 0$  V)

Parameter	Symbol	Test Conditions	MIN.	MAX.	Unit
Internal system clock cycle time *1	$t_{CYK}$		166	1000	ns
Address setup time (to ALE $\uparrow$ )	$t_{SAL}$		150		ns
Address hold time (from ALE $\uparrow$ )	$t_{HLA}$	$C_L = 100$ pF, $R_L = 2$ k $\Omega$ *4	30		ns
$\overline{RD}\uparrow$ delay time from address	$t_{DAR}$		233		ns
Address float time from $\overline{RD}\downarrow$	$t_{FRA}$			0	ns
Data input time from address	$t_{DAID}$			413	ns
Data input time from ALE $\uparrow$	$t_{DLID}$			233	ns
Data input time from $\overline{RD}\uparrow$	$t_{DRID}$			180	ns
$\overline{RD}\downarrow$ delay time from ALE $\uparrow$	$t_{DLR}$		63		ns
Data hold time (from $\overline{RD}\uparrow$ )	$t_{HRID}$		0		ns
Address active time from $\overline{RD}\uparrow$	$t_{DRA}$		53		ns
ALE $\uparrow$ delay time from $\overline{RD}\uparrow$	$t_{DRL}$		116		ns
$\overline{RD}$ low-level width	$t_{WRL}$		200		ns
ALE high-level width	$t_{WLH}$		126		ns
$\overline{WR}\uparrow$ delay time from address	$t_{DAW}$		233		ns

(to be continued)

(cont'd)

Parameter	Symbol	Test Conditions	MIN.	MAX.	Unit
Data output time from ALE↑	$t_{DLOD}$			193	ns
Data output time from $\overline{WR}$ ↑	$t_{DWOD}$			100	ns
$\overline{WR}$ ↑ delay time from ALE↑ *2	$t_{DLW}$		63		ns
		Refresh mode	116		ns
Data setup time (to $\overline{WR}$ ↑)	$t_{SODWR}$		150		ns
Data setup time (to $\overline{WR}$ ↑) *3	$t_{SODWF}$	Refresh mode	33		ns
Data hold time (from $\overline{WR}$ ↑)	$t_{HWOD}$		20		ns
ALE↑ delay time from $\overline{WR}$ ↑	$t_{DWL}$		116		ns
$\overline{WR}$ low-level width	$t_{WWL}$		200		ns
		Refresh mode	116		ns

- \*1: Internal system clock ( $f_{CLK}$ ) is the oscillation clock ( $f_{XX}$ ) divided by 2 or 8 as specified by STBC register. The value in this table is the value when  $f_{XX} = 12$  MHz,  $f_{CLK} = f_{XX}/2$ .
- 2: At pulse refresh operation,  $t_{DLW}$  is one stage lower because  $\overline{WR}$  signal falls after a 1/2 clock delay.
- 3: For a pseudo-static RAM (uPD428128 etc.) of a type that reads data by the falling edge of the  $\overline{WR}$  signal, the data setup time is not  $t_{SODWR}$ , but  $t_{SODWF}$ .
- 4: The hold time includes the time for holding  $V_{OH}$  and  $V_{OL}$  at  $C_L = 100$  pF,  $R_L = 2$  k $\Omega$  load conditions.

Remarks: This table shows the AC characteristics when the number of wait cycles is 0.

Serial Operation ( $T_a = -10$  to  $+70^{\circ}\text{C}$ ,  $V_{DD} = +5.0\text{ V} \pm 10\%$ ,  
 $V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions		MIN.	MAX.	Unit
Serial clock cycle time	$t_{\text{CYSK}}$	Output	$\overline{\text{SCK}}$ *1	1.33		us
			$\overline{\text{CTS}}$ *2	1.33		us
		Input	$\overline{\text{CTS}}$ *3	1		us
Serial clock low-level width	$t_{\text{WSKL}}$	Output	$\overline{\text{SCK}}$ *1	580		ns
			$\overline{\text{CTS}}$ *2	580		ns
		Input	$\overline{\text{CTS}}$ *3	420		ns
Serial clock high-level width	$t_{\text{WSKH}}$	Output	$\overline{\text{SCK}}$ *1	580		ns
			$\overline{\text{CTS}}$ *2	580		ns
		Input	$\overline{\text{CTS}}$ *3	420		ns
$\overline{\text{CTS}}$ high, low-level width	$t_{\text{WCSH}}$ , $t_{\text{WC SL}}$		*4	3		$t_{\text{CYK}}$
RxD setup time (to $\overline{\text{CTS}}\uparrow$ )	$t_{\text{SRXSK}}$			80		ns
RxD hold time (from $\overline{\text{CTS}}\uparrow$ )	$t_{\text{HSKRX}}$			80		ns
TxD delay time from $\overline{\text{SCK}}\downarrow$	$t_{\text{DSKTX}}$				210	ns

- \*1: I/O interface mode transmission, data transfer speed 750 kbps
- 2: I/O interface mode reception, data transfer speed 750 kbps
- 3: I/O interface mode reception, data transfer speed 1 Mbps
- 4: Asynchronous mode

A/D Converter Characteristics ( $T_a = -10$  to  $+70^{\circ}\text{C}$ ,  $V_{DD} = +5\text{ V} \pm 10\%$ ,  $4.0\text{ V} \leq AV_{REF} \leq V_{DD}$ ,  $AV_{SS} = V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions	MIN.	TYP.	MAX.	Unit
Resolution			8			bit
Total error*		$4.0\text{ V} \leq AV_{REF} \leq V_{DD}$ , $166\text{ ns} \leq t_{CYK} \leq 500\text{ ns}$			0.4	%
Quantization error					$\pm 1/2$	LSB
Conversion time	$t_{CONV}$	$166\text{ ns} \leq t_{CYK} \leq 250\text{ ns}$	180			$t_{CYK}$
		$250\text{ ns} \leq t_{CYK} \leq 500\text{ ns}$	120			$t_{CYK}$
Sampling time	$t_{SAMP}$	$166\text{ ns} \leq t_{CYK} \leq 250\text{ ns}$	36			$t_{CYK}$
		$250\text{ ns} \leq t_{CYK} \leq 500\text{ ns}$	24			$t_{CYK}$
Analog input voltage	$V_{IAN}$		-0.3		$AV_{REF} + 0.3$	V
Analog input impedance	$R_{AN}$			1000		$M\Omega$
Reference voltage	$AV_{REF}$		4.0		$V_{DD}$	V
$AV_{REF}$ current	$AI_{REF}$	$f_{CLK} = 6\text{ MHz}$		1.5	5.0	mA

\*: Quantization error not included. Expressed as a percentage of the full-scale value.



Count Unit Operation ( $T_a = -10$  to  $+70^{\circ}\text{C}$ ,  $V_{DD} = +5.0\text{ V}$   
 $\pm 10\%$ ,  $V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions	MIN.	MAX.	Unit
CI0, CI1 high, low-level width	$t_{WCIH}$ , $t_{WCIL}$		3		$t_{CYK}$
CTRL0, CTRL1 high, low-level width	$t_{WCTH}$ , $t_{WCTL}$		3		$t_{CYK}$
CTRL0, CTRL1 setup time (to CI $\uparrow$ )	$t_{SCTCI}$	Specifies count unit operation mode 3, and CI pin input rising edge effective.	2		$t_{CYK}$
CTRL0, CTRL1 hold time (from CI $\uparrow$ )	$t_{HCICT}$	Specifies count unit operation mode 3, and CI pin input rising edge effective.	5		$t_{CYK}$
CLR0, CLR1 high, low-level width	$t_{WCRH}$ , $t_{WCRL}$		3		$t_{CYK}$
CI0, CI1 setup time (to CI $\uparrow$ )	$t_{S4CTCI}$	Specifies count unit operation mode 4	6		$t_{CYK}$
CI0, CI1 hold time (from CI $\uparrow$ )	$t_{H4CICT}$	Specifies count unit operation mode 4	6		$t_{CYK}$
CI0, CI1, CTRL0, CTRL1 cycle time	$t_{CYC4}$	Specifies count unit operation mode 4	4		us

Other Operations ( $T_a = -10$  to  $+70^{\circ}\text{C}$ ,  $V_{DD} = +5.0\text{ V} \pm 10\%$ ,  
 $V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions	MIN.	MAX.	Unit
NMI high, low-level width	$t_{WNIH}$ , $t_{WNIL}$		10		us
INTE0 high, low-level width	$t_{WIOH}$ , $t_{WIOI}$		3		$t_{CYK}$
INTE1 high, low-level width	$t_{WI1H}$ , $t_{WI1L}$		3		$t_{CYK}$
INTE2 high, low-level width	$t_{WI2H}$ , $t_{WI2L}$		3		$t_{CYK}$
$\overline{\text{RESET}}$ high, low-level width	$t_{WRSH}$ , $t_{WRSL}$		10		us
$V_{DD}$ rise time (for using SBF bit)	$t_{ROVD}$		4		ms
$V_{DD}$ rise, fall time	$t_{RVD}$ , $t_{FVD}$		200		us

External Clock Timing ( $T_a = -10$  to  $+70^{\circ}\text{C}$ ,  $V_{DD} = +5.0\text{ V} \pm 10\%$ ,  
 $V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions	MIN.	MAX.	Unit
X1 input high-level width	$t_{WXH}$		30	130	ns
X1 input low-level width	$t_{WXL}$		30	130	ns
X1 input rise time	$t_{XR}$		0	30	ns
X1 input fall time	$t_{XF}$		0	30	ns
X1 input cycle time	$t_{CYX}$		83	250	ns

$t_{CYK}$  Dependent Bus Timing Definition

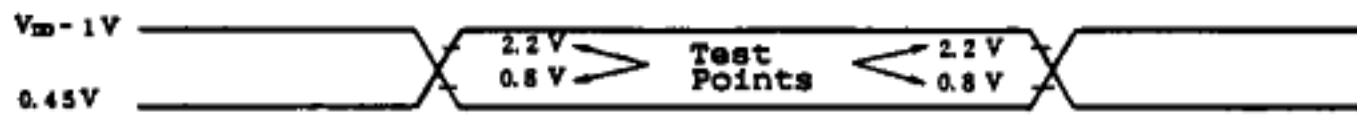
Parameter	Expression	MIN./MAX.	Unit
$t_{SAL}$	1.5 T-100	MIN.	ns
$t_{DAR}$	2 T-100	MIN.	ns
$t_{DAID}$	(3.5+n) T-170	MAX.	ns
$t_{DLID}$	(2+n) T-100	MAX.	ns
$t_{DRID}$	(1.5+n) T-70	MAX.	ns
$t_{DLR}$	0.5 T-20	MIN.	ns
$t_{DRL}$	T-50	MIN.	ns
$t_{DRA}$	0.5 T-30	MIN.	ns
$t_{WRL}$	(1.5+n) T-50	MIN.	ns
$t_{WLH}$	T-40	MIN.	ns
$t_{DAW}$	2 T-100	MIN.	ns
$t_{DLOD}$	0.5 T+110	MAX.	ns
$t_{DLW}$	0.5 T-20 (normal operation)	MIN.	ns
	T-50 (refresh mode)	MIN.	ns
$t_{SODWR}$	(1.5+n) T-100	MIN.	ns
$t_{SODWF}$	0.5 T-50	MIN.	ns
$t_{DWL}$	T-50	MIN.	ns
$t_{WWL}$	(1.5+n) T-50 (normal operation)	MIN.	ns
	(1+n) T-50 (refresh mode)	MIN.	ns

Remarks 1: n is the number of wait cycles inserted by MM register specification.

2:  $T = t_{CYK} = 1/f_{CLK}$  ( $f_{CLK}$ : Internal system clock frequency)

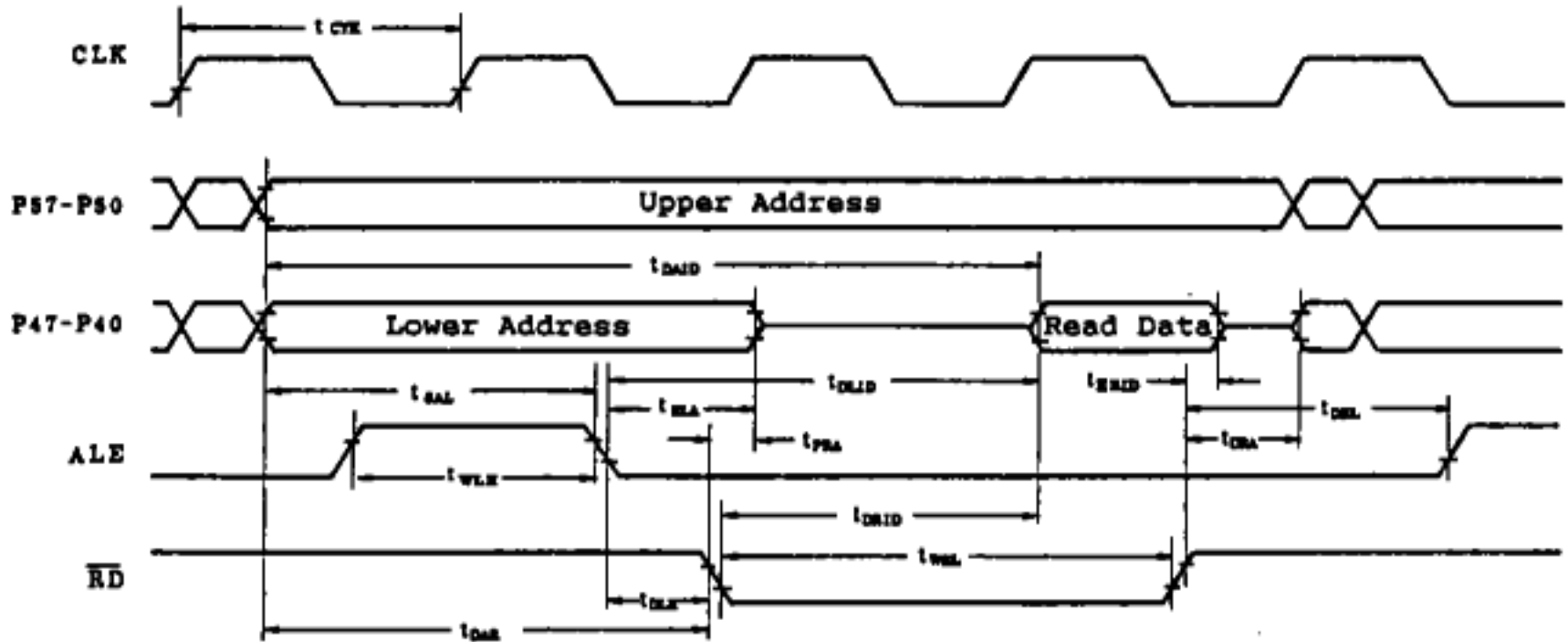
3: Items not in this table does not depend on the internal system clock frequency ( $f_{CLK}$ ).

## AC Timing Test Point

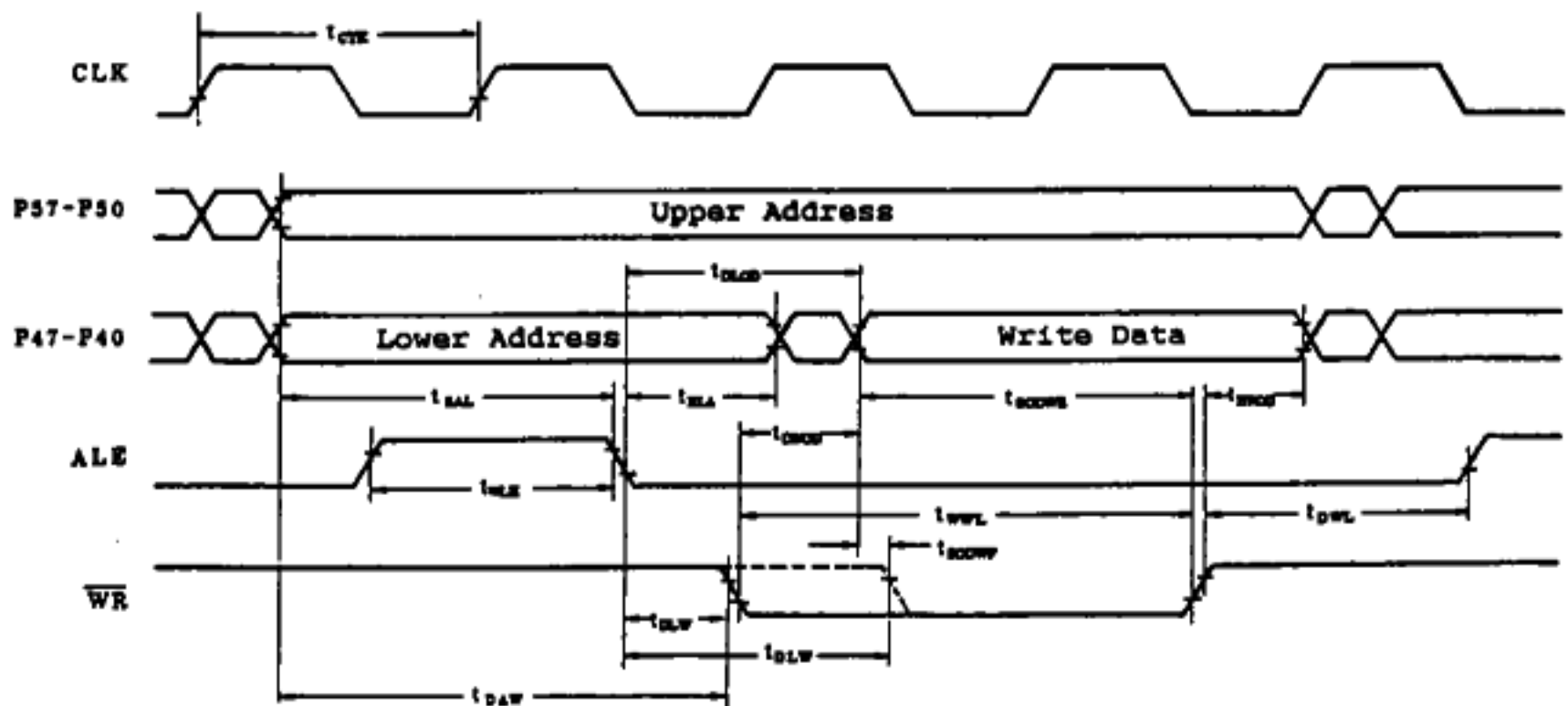


## Timing Waveform

### Read operation:

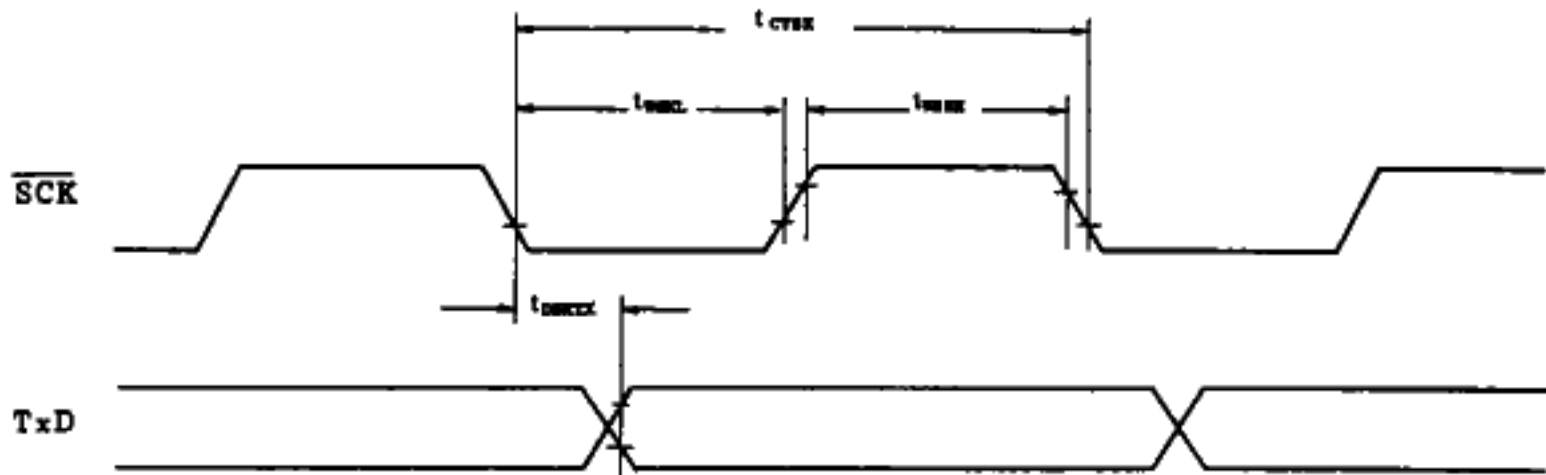


### Write operation:

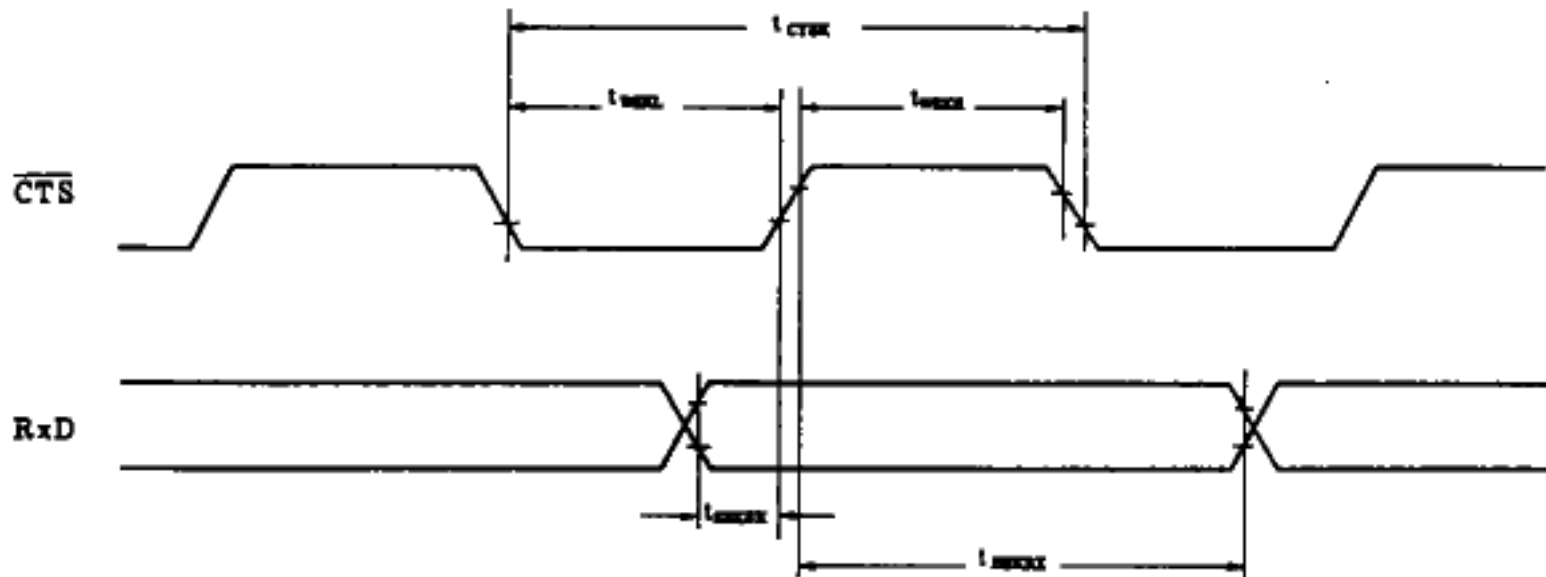


## Serial Operation

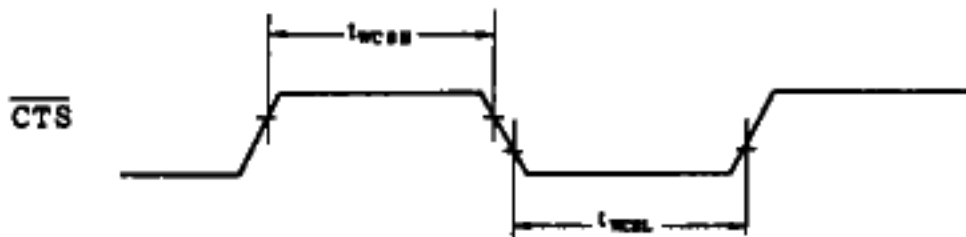
### I/O interface mode transmission:



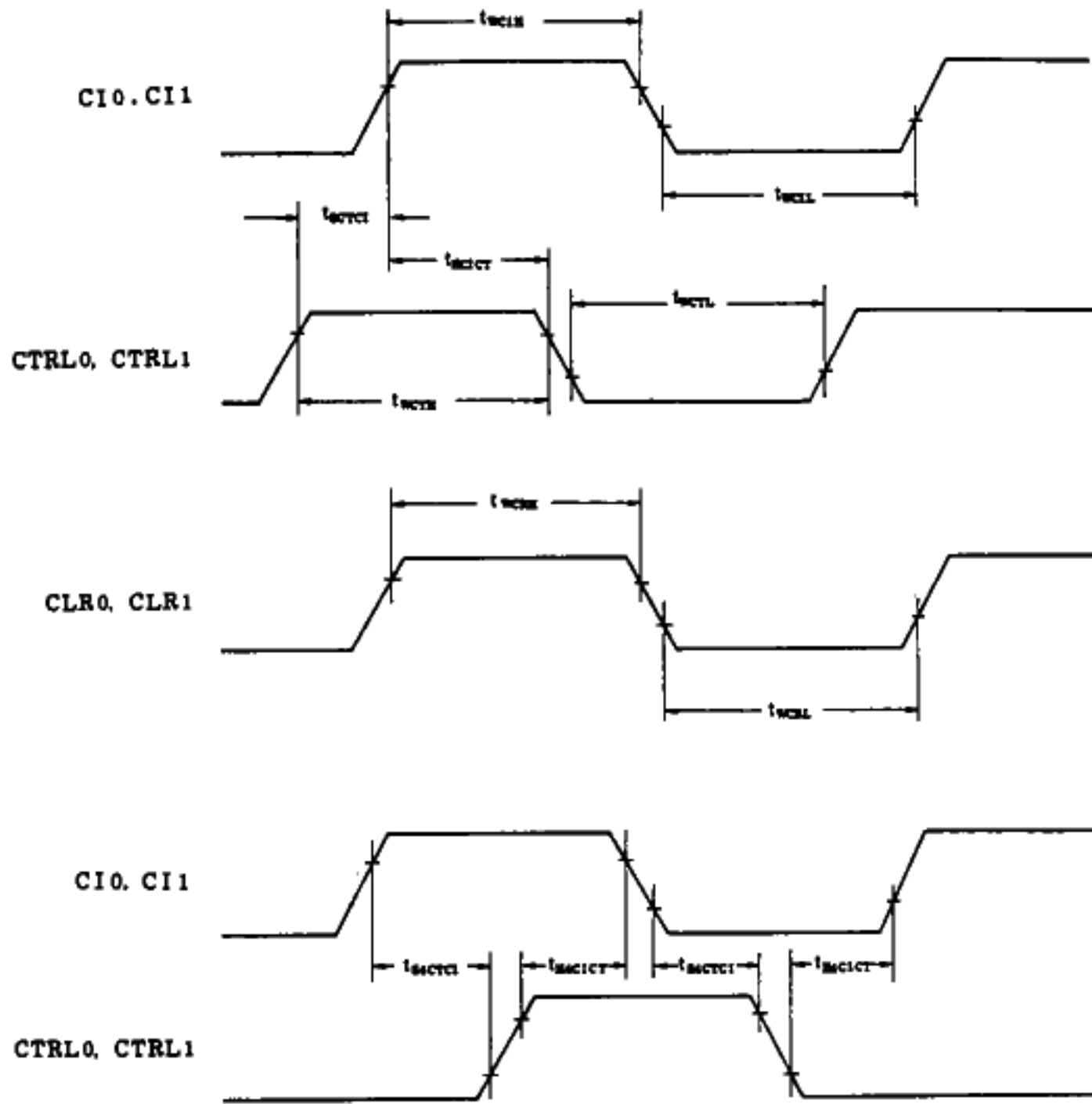
### I/O interface mode reception:



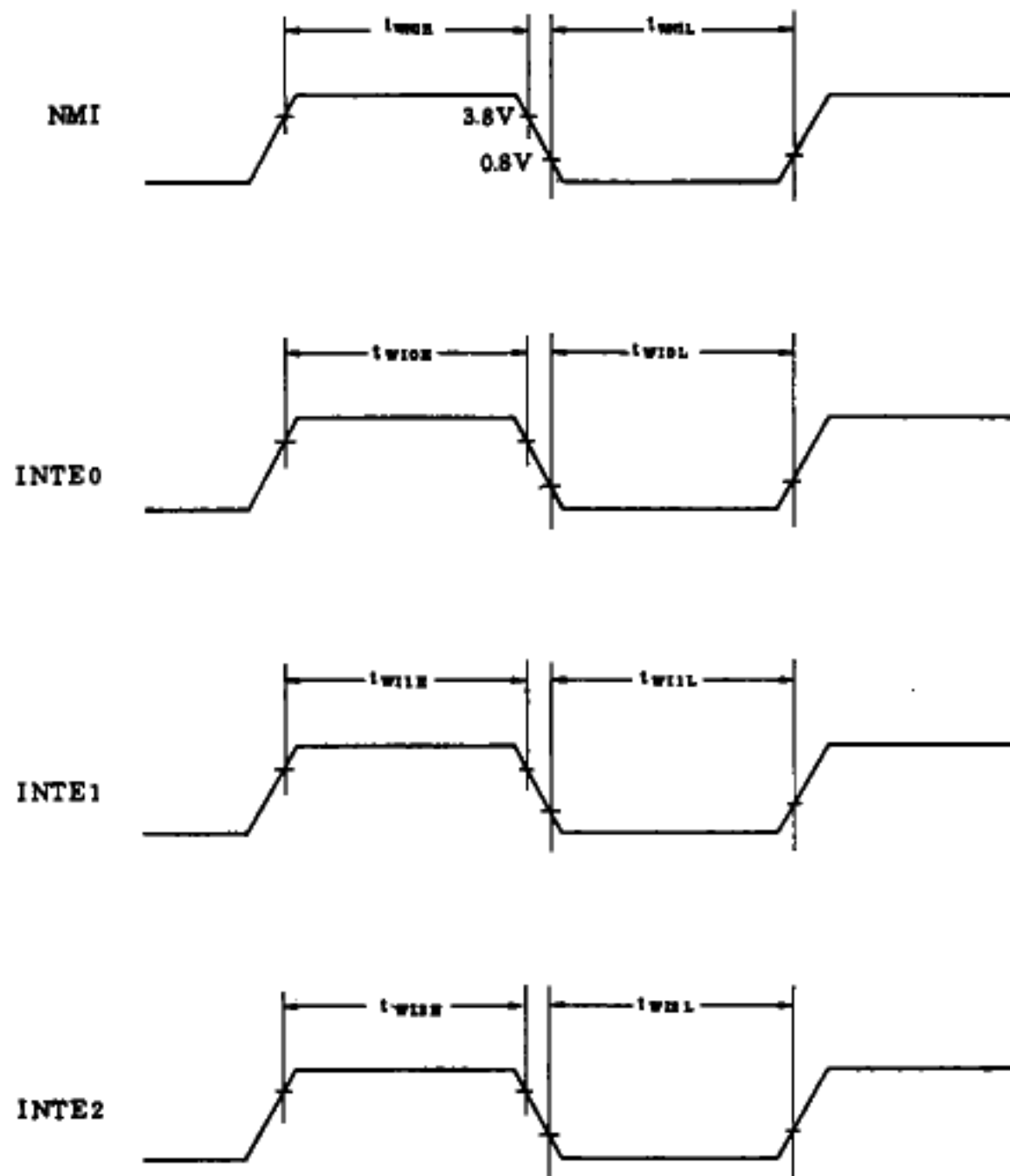
### Transmit enable input timing (asynchronous mode):



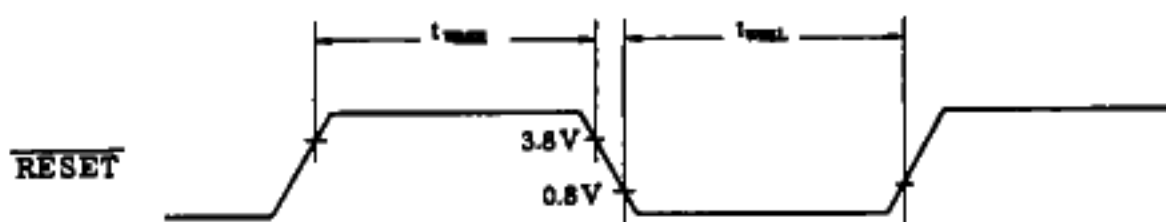
# Count Unit Input Timing



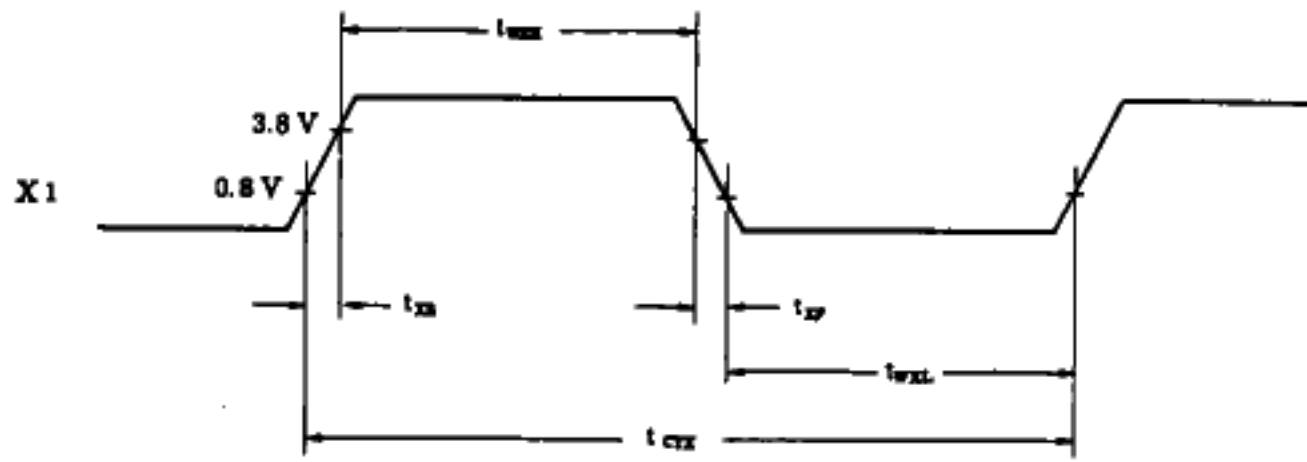
## Interrupt Input Timing



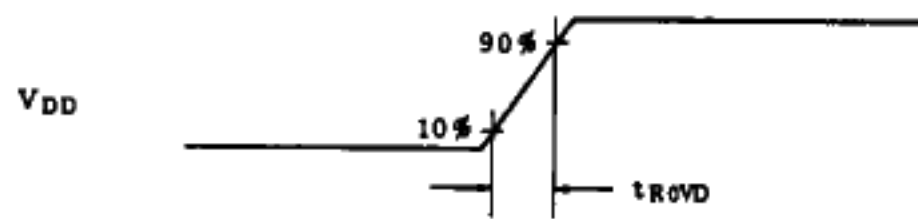
## Reset Input Timing



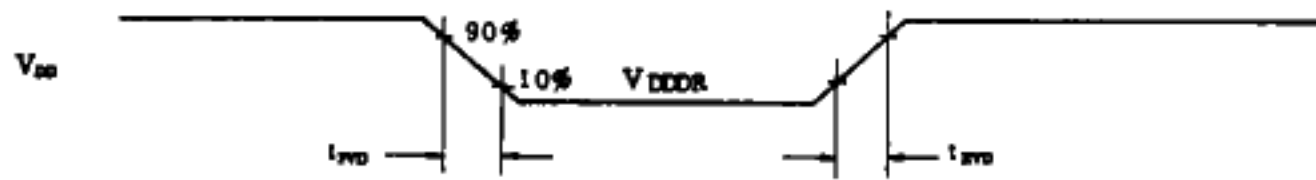
### External Clock Timing



### Power On Timing



### Data Hold Timing





## 11.2 ELECTRICAL SPECIFICATIONS (uPD78P312A)

### Absolute Maximum Ratings (Ta = 25°C)

Parameter	Symbol	Test Conditions	Rating	Unit
Power supply voltage	$V_{DD}$		-0.5 to +7.0	V
	$AV_{REF}$		-0.5 to $V_{DD}$ +0.3	V
	$AV_{SS}$		-0.5 to +0.5	V
	$V_{PP}$		-0.5 to +13.5	V
Input voltage	$V_{I1}$	Except for <u>RESET</u>	-0.5 to $+V_{DD}$ +0.5	V
	$V_{I2}$	<u>RESET</u>	-0.5 to +13.5	V
Output voltage	$V_O$		-0.5 to $+V_{DD}$ +0.5	V
Output current low	$V_{OL}$	1 pin	4.0	mA
		All output pins total	60	mA
Output current high	$I_{OH}$	1 pin	-2	mA
		All output pins total	-15	mA
Operating temperature	$T_{opt}$		-10 to +70	°C
Storage temperature	$T_{stg}$		-65 to +150	°C

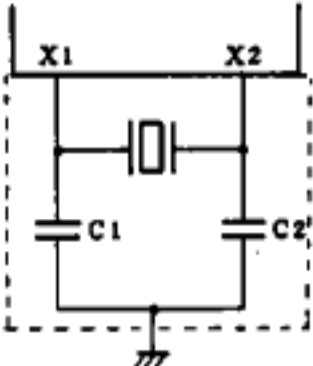
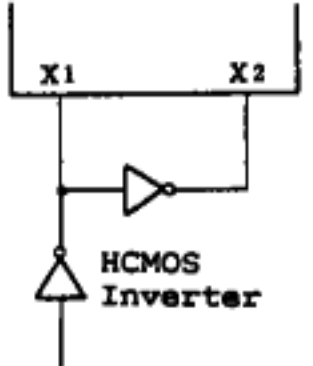
### Recommended Operating Conditions

Oscillation Frequency	Parameter	Ta	$V_{DD}$
$4 \text{ MHz} \leq f_{XX} \leq 12 \text{ MHz}$		-10 to +70°C	+5.0 V $\pm 10\%$

Capacitance ( $T_a = 25^{\circ}\text{C}$ ,  $V_{DD} = V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions	MIN.	TYP.	MAX.	Unit
Input capacitance	$C_I$	$f = 1\text{ MHz}$			10	pF
Output capacitance	$C_O$	Unmeasured pins returned to 0 V			20	pF
I/O capacitance	$C_{IO}$				20	pF

Oscillator Characteristics ( $T_a = -10$  to  $+70^{\circ}\text{C}$ ,  $V_{DD} = +5.0\text{ V} \pm 10\%$ ,  $V_{SS} = AV_{SS} = 0\text{ V}$ ,  $4.0\text{ V} \leq AV_{REF} \leq V_{DD}$ )

Oscillator	Recommended Circuit	Parameter	MIN.	TYP.	MAX.	Unit
Ceramic or crystal resonator		Oscillation frequency ( $f_{XX}$ )	4		12	MHz
External clock		X1 input frequency ( $f_X$ )	4		12	MHz
		X1 input rise and fall times ( $t_{XR}$ , $t_{XF}$ )	0		30	ns
		X1 input high, low-level width ( $t_{WXH}$ , $t_{WXL}$ )	30		130	ns

NOTE 1: Place the oscillator circuit as close to the X1, X2 pins as possible.

2: Do not pass other signals within a range of dotted area.

## Recommended Oscillator Constants

### Ceramic Resonator

Manufacturer	Product Name	Frequency [MHz]	External Capacitance [pF]	
			C1	C2
Murata Mfg. Co., Ltd.	CSA8.00MT CSA10.0MT CSA12.0MT	8.0 10.0 12.0	30	30
	CST10.0MT CST10.0MT CST12.0MT	8.0 10.0 12.0	On-chip	On-chip
Kyocera Corp.	KBR-8.0M KBR-10.0M KBR-12.0M	8.0 10.0 12.0	33	33
TDK	FCR10.0MC FCR12.0MC	10.0 12.0	On-chip	On-chip

### Crystal Resonator

Manufacturer	Product Name	Frequency [MHz]	External Capacitance [pF]	
			C1	C2
Kinseki Ltd.	HC-49U	8.0 10.0 12.0	22	22

DC Characteristics ( $T_a = -10$  to  $+70^\circ\text{C}$ ,  $V_{DD} = +5.0\text{ V} \pm 10\%$ ,  
 $V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions	MIN.	TYP.	MAX.	Unit	
Input voltage low	$V_{IL1}$	Other than $\overline{EA}$	0		0.8	V	
	$V_{IL2}$	$\overline{EA}$	0		0.5	V	
Input voltage high	$V_{IH1}$	Except for P20/NMI, X1, X2, $\overline{RESET}$	2.2		$V_{DD}$	V	
	$V_{IH2}$	P20/NMI, X1, X2, $\overline{RESET}$	3.8		$V_{DD}$	V	
Output voltage low	$V_{OL}$	$I_{OL} = 2.0\text{ mA}$			0.45	V	
Output voltage high	$V_{OH}$	$I_{OH} = -1.0\text{ mA}$	$V_{DD} - 1$			V	
Input current	$I_I$	P20/NMI, $\overline{RESET}$ $0.45\text{ V} < V_I < V_{DD}$			$\pm 10$	$\mu\text{A}$	
Input leakage current	$I_{LI}$				$\pm 10$	$\mu\text{A}$	
I/O leakage current	$I_{LO}$				$\pm 10$	$\mu\text{A}$	
$AV_{REF}$ current	$AI_{REF}$	$f_{CLK} = 6\text{ MHz}$		1.5	5	mA	
$V_{DD}$ power supply current	$I_{DD1}$	Operating mode, $f_{CLK} = 6\text{ MHz}$		30	60	mA	
	$I_{DD2}$	HALT mode, $f_{CLK} = 6\text{ MHz}$		5	15	mA	
Data hold voltage	$V_{DDDR}$	STOP mode	2.5			V	
Data hold current	$I_{DDDR}$	STOP mode	$V_{DDDR} = 2.5\text{ V}$		3	15	$\mu\text{A}$
			$V_{DDDR} = 5.0\text{ V} \pm 10\%$		10	50	$\mu\text{A}$

## AC Characteristics

Read/Write Operation ( $T_a = -10$  to  $+70^{\circ}\text{C}$ ,  $V_{DD} = +5.0$  V  
 $\pm 10\%$ ,  $V_{SS} = 0$  V)

Parameter	Symbol	Test Conditions	MIN.	MAX.	Unit
Internal system clock cycle time *1	$t_{CYK}$		166	1000	ns
Address setup time (to ALE $\uparrow$ )	$t_{SAL}$		150		ns
Address hold time (from ALE $\uparrow$ )	$t_{HLA}$		30		ns
$\overline{RD}\downarrow$ delay time from address	$t_{DAR}$		233		ns
Address float time from $\overline{RD}\downarrow$	$t_{FRA}$			0	ns
Data input time from address	$t_{DAID}$			413	ns
Data input time from ALE $\uparrow$	$t_{DLID}$			233	ns
Data input time from $\overline{RD}\downarrow$	$t_{DRID}$			180	ns
$\overline{RD}\downarrow$ delay time from ALE $\uparrow$	$t_{DLR}$		63		ns
Data hold time (from $\overline{RD}\downarrow$ )	$t_{HRID}$		0		ns
Address active time from $\overline{RD}\downarrow$	$t_{DRA}$		53		ns
ALE $\uparrow$ delay time from $\overline{RD}\downarrow$	$t_{DRL}$		116		ns
$\overline{RD}$ low-level width	$t_{WRL}$		200		ns
ALE high-level width	$t_{WLH}$		126		ns
$\overline{WR}\downarrow$ delay time from address	$t_{DAW}$		233		ns

(to be continued)

(cont'd)

Parameter	Symbol	Test Conditions	MIN.	MAX.	Unit
Data output time from ALE $\uparrow$	$t_{DLOD}$			193	ns
Data output time from $\overline{WR}\uparrow$	$t_{DWOD}$			100	ns
$\overline{WR}\uparrow$ delay time from ALE $\uparrow$ *2	$t_{DLW}$		63		ns
		Refresh mode	116		ns
Data setup time (to $\overline{WR}\uparrow$ )	$t_{SODWR}$		150		ns
Data setup time (to $\overline{WR}\uparrow$ ) *3	$t_{SODWF}$	Refresh mode	33		ns
Data hold time (from $\overline{WR}\uparrow$ )	$t_{HWOD}$		20		ns
ALE $\uparrow$ delay time from $\overline{WR}\uparrow$	$t_{DWL}$		116		ns
$\overline{WR}$ low-level width	$t_{WWL}$		200		ns
		Refresh mode	116		ns

- \*1: Internal system clock ( $f_{CLK}$ ) is the oscillation clock ( $f_{XX}$ ) divided by 2 or 8 as specified by STBC register. The value in this table is the value when  $f_{XX} = 12$  MHz,  $f_{CLK} = f_{XX}/2$ .
- 2: At pulse refresh operation,  $t_{DLW}$  is one stage lower because  $\overline{WR}$  signal falls after a 1/2 clock delay.
- 3: For a pseudo-static RAM (uPD428128 etc.) of a type that reads data by the falling edge of the  $\overline{WR}$  signal, the data setup time is not  $t_{SODWR}$ , but  $t_{SODWF}$ .

Remarks: This table shows the AC characteristics when the number of wait cycles is 0.

Serial Operation ( $T_a = -10$  to  $+70^{\circ}\text{C}$ ,  $V_{DD} = +5.0\text{ V} \pm 10\%$ ,  
 $V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions		MIN.	MAX.	Unit
Serial clock cycle time	$t_{\text{CYSK}}$	Output	$\overline{\text{SCK}}$ *1	1.33		us
			$\overline{\text{CTS}}$ *2	1.33		us
		Input	$\overline{\text{CTS}}$ *3	1		us
Serial clock low-level width	$t_{\text{WSKL}}$	Output	$\overline{\text{SCK}}$ *1	580		ns
			$\overline{\text{CTS}}$ *2	580		ns
		Input	$\overline{\text{CTS}}$ *3	420		ns
Serial clock high-level width	$t_{\text{WSKH}}$	Output	$\overline{\text{SCK}}$ *1	580		ns
			$\overline{\text{CTS}}$ *2	580		ns
		Input	$\overline{\text{CTS}}$ *3	420		ns
$\overline{\text{CTS}}$ high, low-level width	$t_{\text{WCSH}}$ , $t_{\text{WC SL}}$		*4	3		$t_{\text{CYK}}$
RxD setup time (to $\overline{\text{CTS}}\uparrow$ )	$t_{\text{SRXSK}}$			80		ns
RxD hold time (from $\overline{\text{CTS}}\uparrow$ )	$t_{\text{HSKRX}}$			80		ns
TxD delay time from $\overline{\text{SCK}}\uparrow$	$t_{\text{DSKTX}}$				210	ns

- \*1: I/O interface mode transmission, data transfer speed 750 kbps
- 2: I/O interface mode reception, data transfer speed 750 kbps
- 3: I/O interface mode reception, data transfer speed 1 Mbps
- 4: Asynchronous mode

A/D Converter Characteristics ( $T_a = -10$  to  $+70^\circ\text{C}$ ,  $V_{DD} = +5\text{ V} \pm 10\%$ ,  $4.0\text{ V} \leq AV_{REF} \leq V_{DD}$ ,  $AV_{SS} = V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions	MIN.	TYP.	MAX.	Unit
Resolution			8			bit
Total error*		$4.0\text{ V} \leq AV_{REF} \leq V_{DD}$ , $166\text{ ns} \leq t_{CYK} \leq 500\text{ ns}$			0.4	%
Quantization error					$\pm 1/2$	LSB
Conversion time	$t_{CONV}$	$166\text{ ns} \leq t_{CYK} \leq 250\text{ ns}$	180			$t_{CYK}$
		$250\text{ ns} \leq t_{CYK} \leq 500\text{ ns}$	120			$t_{CYK}$
Sampling time	$t_{SAMP}$	$166\text{ ns} \leq t_{CYK} \leq 250\text{ ns}$	36			$t_{CYK}$
		$250\text{ ns} \leq t_{CYK} \leq 500\text{ ns}$	24			$t_{CYK}$
Analog input voltage	$V_{IAN}$		-0.3		$AV_{REF} + 0.3$	V
Analog input impedance	$R_{AN}$			1000		$M\Omega$
Reference voltage	$AV_{REF}$		4.0		$V_{DD}$	V
$AV_{REF}$ current	$AI_{REF}$	$f_{CLK} = 6\text{ MHz}$		1.5	5.0	mA

\*: Quantization error not included. Expressed as a percentage of the full-scale value.



Count Unit Operation ( $T_a = -10$  to  $+70^{\circ}\text{C}$ ,  $V_{DD} = +5.0\text{ V}$   
 $\pm 10\%$ ,  $V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions	MIN.	MAX.	Unit
CI0, CI1 high, low-level width	$t_{WC1H}$ , $t_{WC1L}$		3		$t_{CYK}$
CTRL0, CTRL1 high, low-level width	$t_{WCTH}$ , $t_{WCTL}$		3		$t_{CYK}$
CTRL0, CTRL1 setup time (to CI $\uparrow$ )	$t_{SCTCI}$	Specifies count unit operation mode 3, and CI pin input rising edge effective.	2		$t_{CYK}$
CTRL0, CTRL1 hold time (from CI $\uparrow$ )	$t_{HCICT}$	Specifies count unit operation mode 3, and CI pin input rising edge effective.	5		$t_{CYK}$
CLR0, CLR1 high, low-level width	$t_{WCRH}$ , $t_{WCRL}$		3		$t_{CYK}$
CI0, CI1 setup time (to CI $\uparrow$ )	$t_{S4CTCI}$	Specifies count unit operation mode 4	6		$t_{CYK}$
CI0, CI1 hold time (from CI $\uparrow$ )	$t_{H4CICT}$	Specifies count unit operation mode 4	6		$t_{CYK}$
CI0, CI1, CTRL0, CTRL1 cycle time	$t_{CYC4}$	Specifies count unit operation mode 4	4		us

Other Operations ( $T_a = -10$  to  $+70^{\circ}\text{C}$ ,  $V_{DD} = +5.0\text{ V} \pm 10\%$ ,  
 $V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions	MIN.	MAX.	Unit
NMI high, low-level width	$t_{WNIH}$ , $t_{WNIL}$		10		us
INTE0 high, low-level width	$t_{WIOH}$ , $t_{WIO L}$		3		$t_{CYK}$
INTE1 high, low-level width	$t_{WI1H}$ , $t_{WI1L}$		3		$t_{CYK}$
INTE2 high, low-level width	$t_{WI2H}$ , $t_{WI2L}$		3		$t_{CYK}$
$\overline{\text{RESET}}$ high, low-level width	$t_{WRSH}$ , $t_{WRSL}$		10		us
$V_{DD}$ rise time (for using SBF bit)	$t_{ROVD}$		4		ms
$V_{DD}$ rise, fall time	$t_{RVD}$ , $t_{FVD}$		200		us

External Clock Timing ( $T_a = -10$  to  $+70^{\circ}\text{C}$ ,  $V_{DD} = +5.0\text{ V} \pm 10\%$ ,  
 $V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions	MIN.	MAX.	Unit
X1 input high-level width	$t_{WXH}$		30	130	ns
X1 input low-level width	$t_{WXL}$		30	130	ns
X1 input rise time	$t_{XR}$		0	30	ns
X1 input fall time	$t_{XF}$		0	30	ns
X1 input cycle time	$t_{CYX}$		83	250	ns

$t_{CYK}$  Dependent Bus Timing Definition

Parameter	Expression	MIN./MAX.	Unit
$t_{SAL}$	1.5 T-100	MIN.	ns
$t_{DAR}$	2 T-100	MIN.	ns
$t_{DAID}$	(3.5+n) T-170	MAX.	ns
$t_{DLID}$	(2+n) T-100	MAX.	ns
$t_{DRID}$	(1.5+n) T-70	MAX.	ns
$t_{DLR}$	0.5 T-20	MIN.	ns
$t_{DRL}$	T-50	MIN.	ns
$t_{DRA}$	0.5 T-30	MIN.	ns
$t_{WRL}$	(1.5+n) T-50	MIN.	ns
$t_{WLH}$	T-40	MIN.	ns
$t_{DAW}$	2 T-100	MIN.	ns
$t_{DLOD}$	0.5 T+110	MAX.	ns
$t_{DLW}$	0.5 T-20 (normal operation)	MIN.	ns
	T-50 (refresh mode)	MIN.	ns
$t_{SODWR}$	(1.5+n) T-100	MIN.	ns
$t_{SODWF}$	0.5 T-50	MIN.	ns
$t_{DWL}$	T-50	MIN.	ns
$t_{WWL}$	(1.5+n) T-50 (normal operation)	MIN.	ns
	(1+n) T-50 (refresh mode)	MIN.	ns

Remarks 1: n is the number of wait cycles inserted by MM register specification.

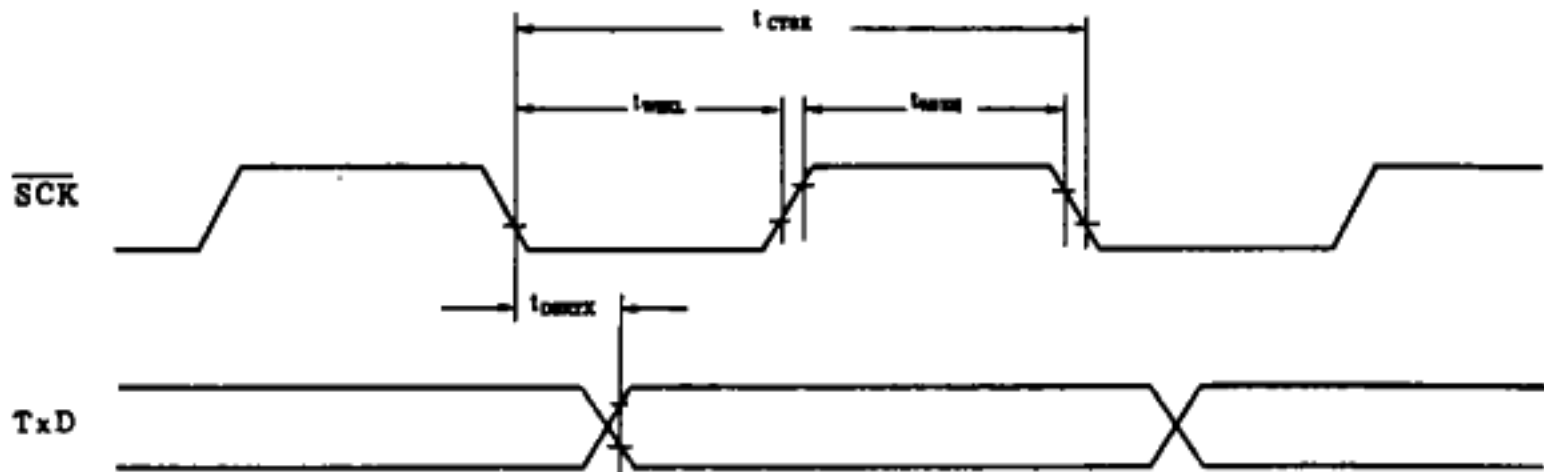
2:  $T = t_{CYK} = 1/f_{CLK}$  ( $f_{CLK}$ : Internal system clock frequency)

3: Items not in this table does not depend on the internal system clock frequency ( $f_{CLK}$ ).

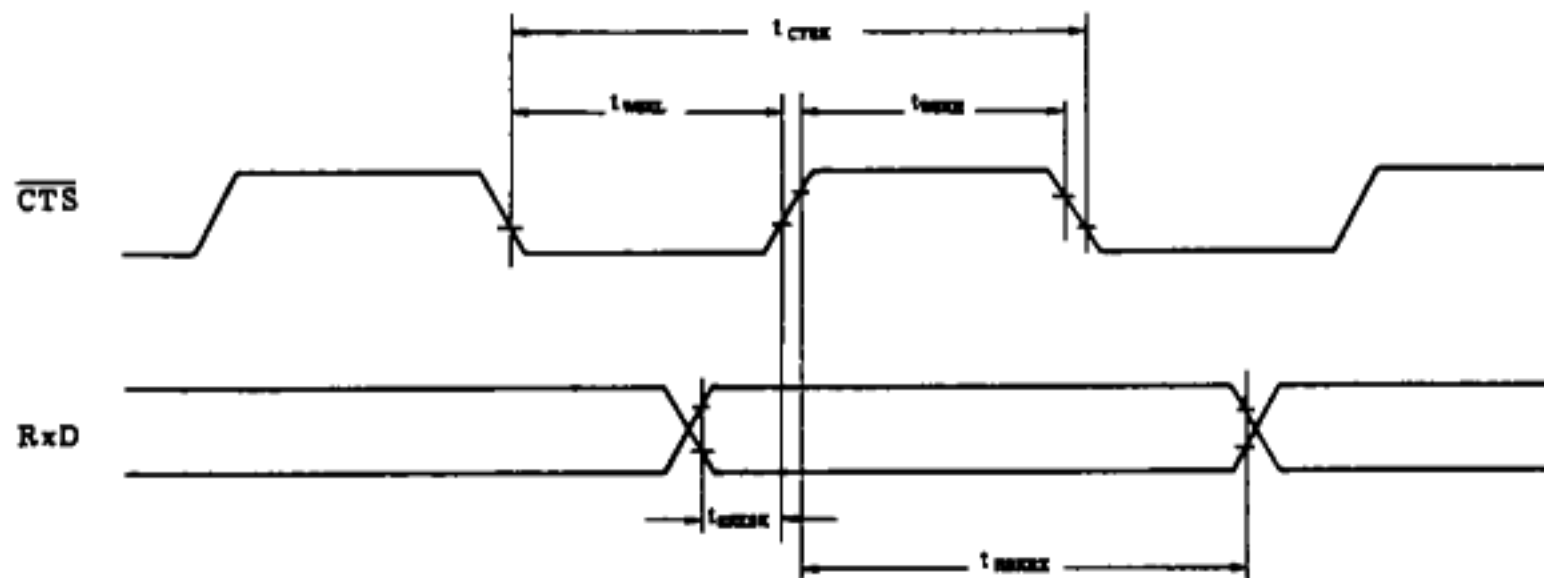


## Serial Operation

### I/O interface mode transmission:



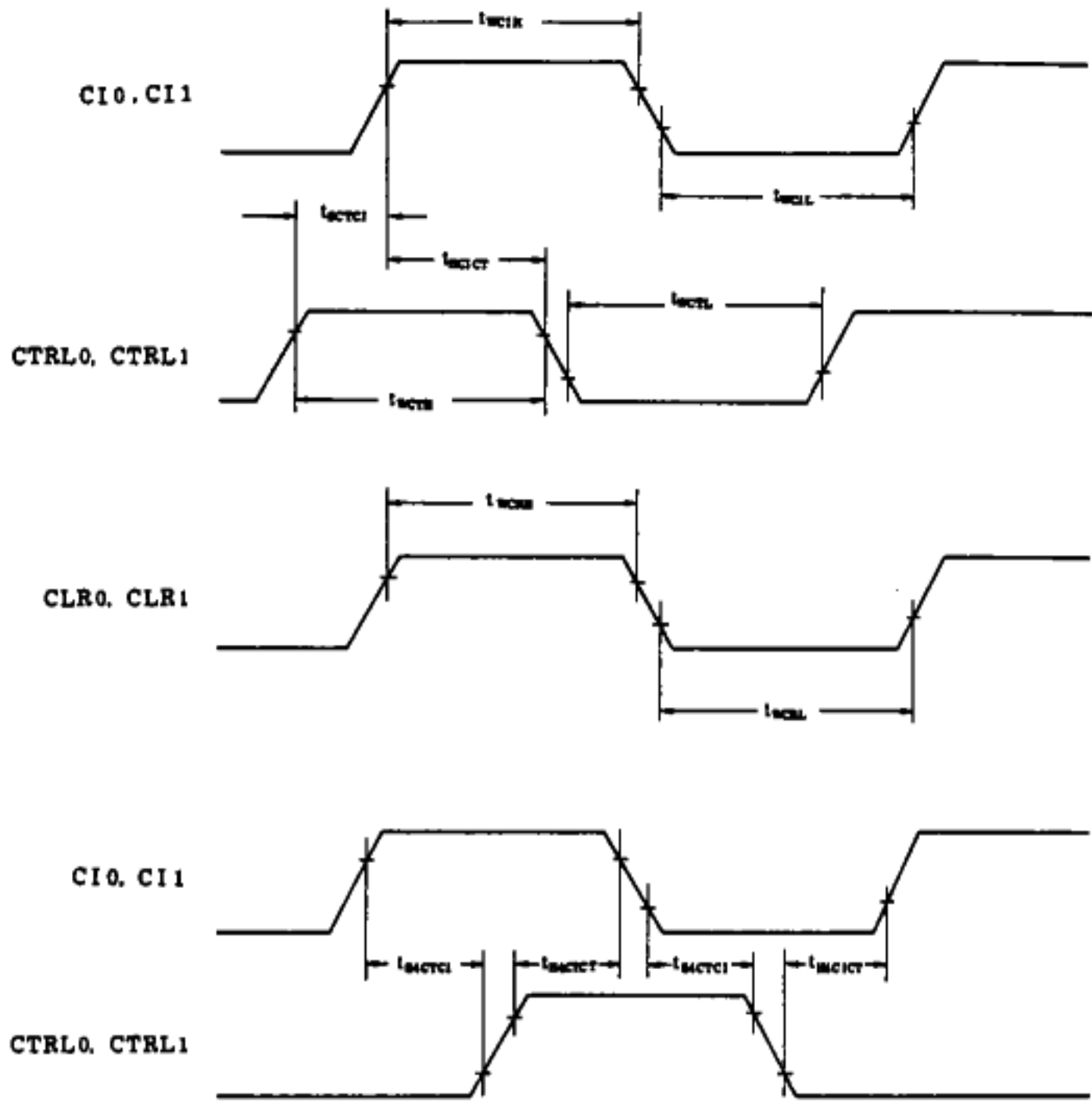
### I/O interface mode reception:



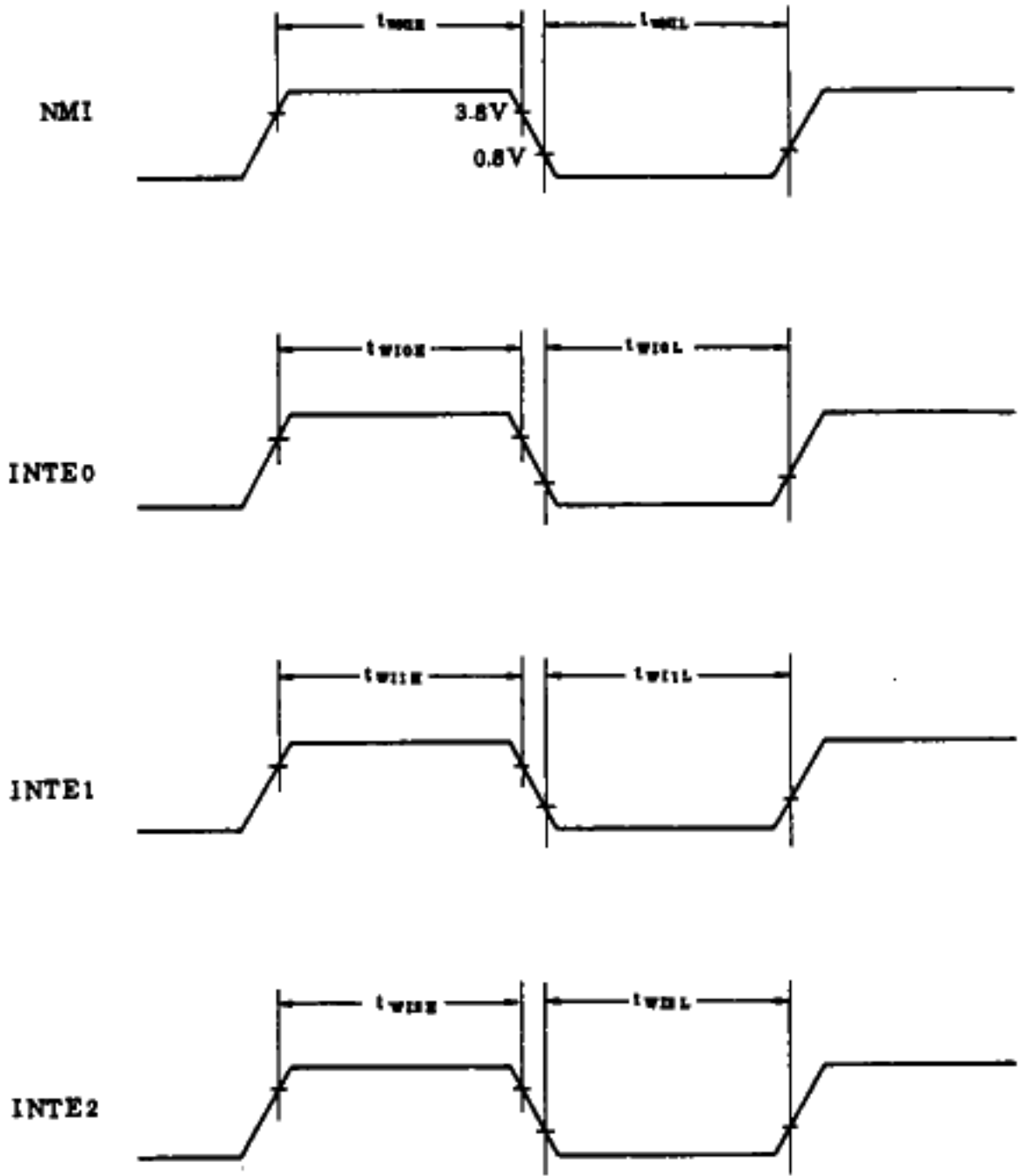
### Transmit enable input timing (asynchronous mode):



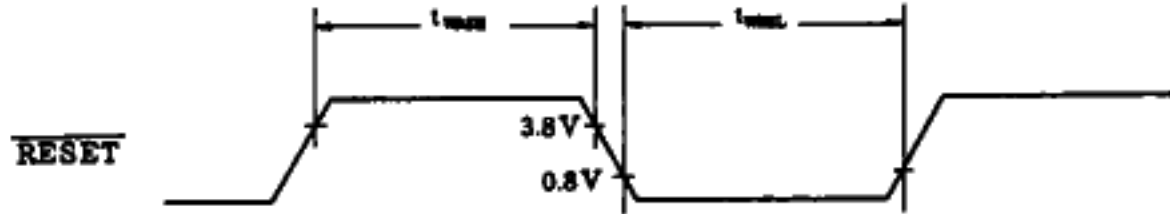
# Count Unit Input Timing



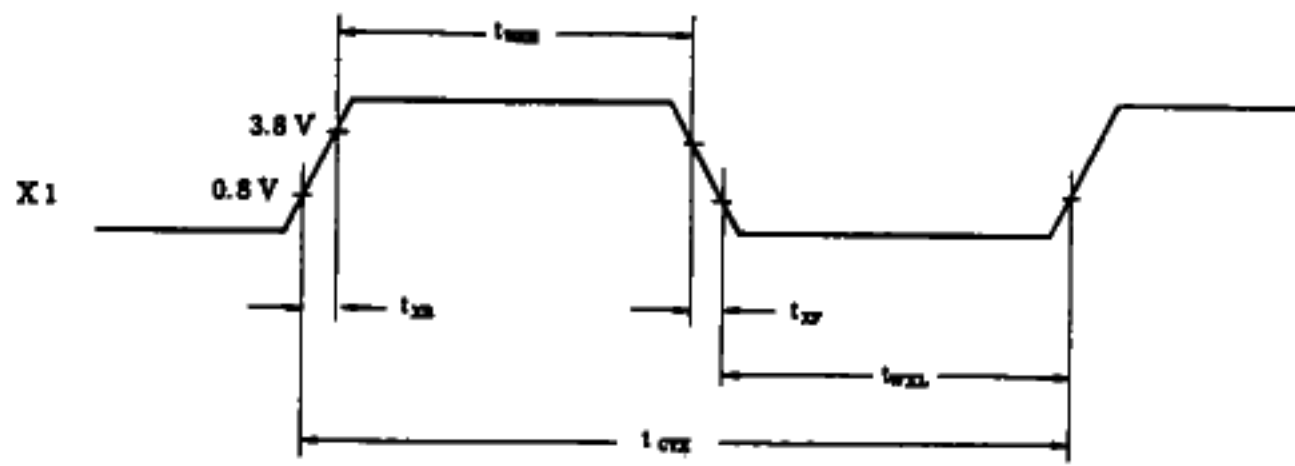
**Interrupt Input Timing**



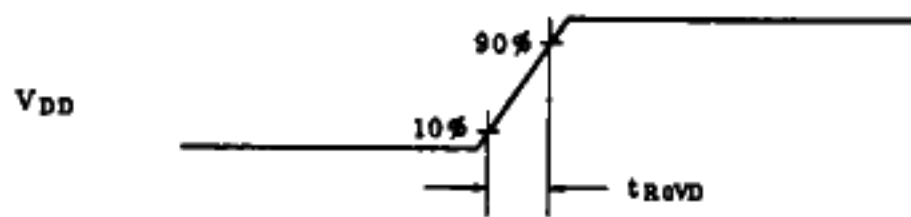
**Reset Input Timing**



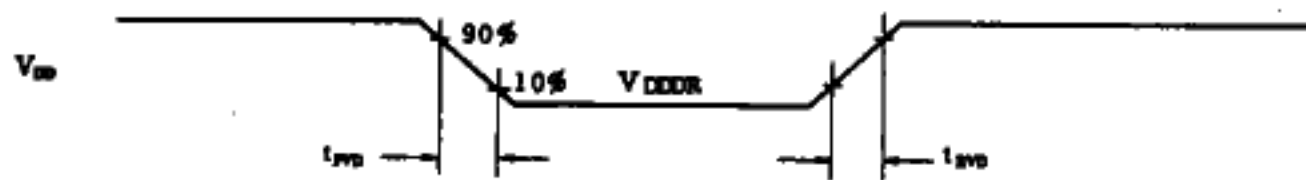
## External Clock Timing



## Power On Timing



## Data Hold Timing





DC Programming Characteristics ( $T_a = 25 \pm 5^\circ\text{C}$ ,  $V_{IP} = 12.0 \pm 0.5 \text{ V}$ ,  $V_{SS} = 0 \text{ V}$ )

Parameter	Symbol	Symbol*	Test Conditions	MIN.	TYP.	MAX.	Unit
Input voltage high	$V_{IH}$	$V_{IH}$		2.2		$V_{DDP} + 0.3$	V
Input voltage low	$V_{IL}$	$V_{IL}$		-0.3		0.8	V
Input leakage current	$I_{LIP}$	$I_{LI}$	$0 \leq V_I \leq V_{DDP}$			10	$\mu\text{A}$
Output voltage high	$V_{OH}$	$V_{OH}$	$I_{OH} = -1.0 \text{ mA}$	$V_{DD} - 1$			V
Output voltage low	$V_{OL}$	$V_{OL}$	$I_{OL} = 2.0 \text{ mA}$			0.45	V
Output leakage current	$I_{LO}$	—	$0 \leq V_O \leq V_{DDP}$ , $\overline{OE} = V_{IH}$			10	$\mu\text{A}$
PROG pin input current high	$I_{IP}$	—				$\pm 10$	$\mu\text{A}$
$V_{DDP}$ supply voltage	$V_{DDP}$	$V_{DD}$	Program memory write mode	5.75	6.0	6.25	V
			Program memory read mode	4.5	5.0	5.5	V
$V_{pp}$ supply voltage	$V_{pp}$	$V_{pp}$	Program memory write mode	12.2	12.5	12.8	V
			Program memory read mode	$V_{pp} = V_{DDP}$			V
$V_{DDP}$ supply current	$I_{DD}$	$I_{DD}$	Program memory write mode		10	30	mA
			Program memory read mode $\overline{CE} = V_{IL}$ , $V_I = V_{IH}$		10	30	mA
$V_{pp}$ supply current	$I_{pp}$	$I_{pp}$	Program memory write mode $\overline{CE} = V_{IL}$ , $\overline{OE} = V_{IH}$		10	30	mA
			Program memory read mode		1	100	$\mu\text{A}$

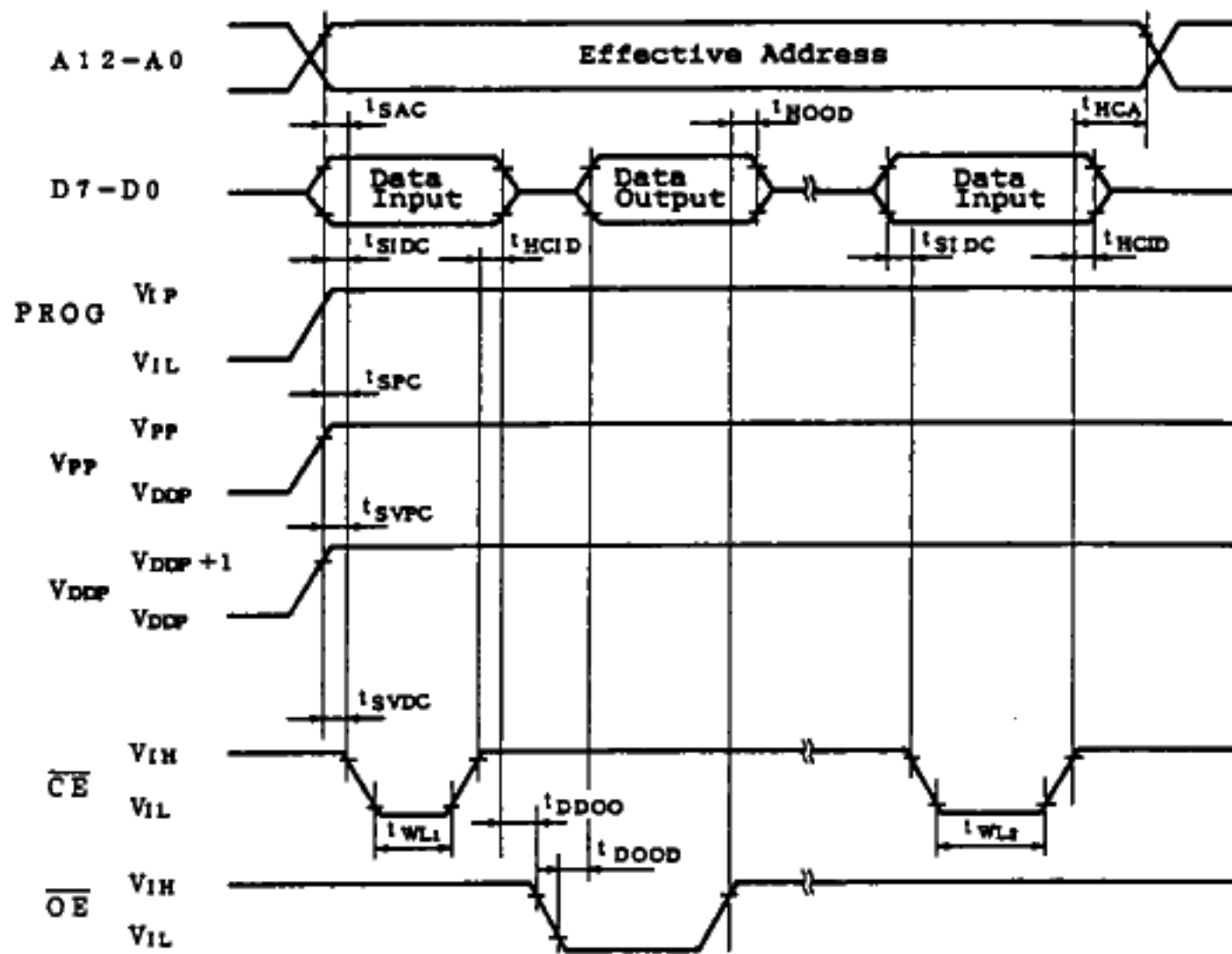
\*: Symbol of corresponding uPD27C256A

AC Programming Characteristics ( $T_a = 25 \pm 5^\circ\text{C}$ ,  $V_{IP} = 12.0 \pm 0.5 \text{ V}$ ,  $V_{SS} = 0 \text{ V}$ )

Parameter	Symbol	Symbol*	Test Conditions	MIN.	TYP.	MAX.	Unit
Address setup time (to $\overline{\text{CE}}\uparrow$ )	$t_{\text{SAC}}$	$t_{\text{AS}}$		2			us
$\overline{\text{OE}}\uparrow$ delay time from data	$t_{\text{DDOO}}$	$t_{\text{OES}}$		2			us
Input data setup time (to $\overline{\text{CE}}\uparrow$ )	$t_{\text{SIDC}}$	$t_{\text{DS}}$		2			us
Address hold time (from $\overline{\text{CE}}\uparrow$ )	$t_{\text{HCA}}$	$t_{\text{AH}}$		2			us
Input data hold time (from $\overline{\text{CE}}\uparrow$ )	$t_{\text{HCID}}$	$t_{\text{DH}}$		2			us
Output data hold time (from $\overline{\text{OE}}\uparrow$ )	$t_{\text{HOOD}}$	$t_{\text{DF}}$		0		130	ns
$V_{\text{PP}}$ setup time (to $\overline{\text{CE}}\uparrow$ )	$t_{\text{SVPC}}$	$t_{\text{VPS}}$		2			us
$V_{\text{DDP}}$ setup time (to $\overline{\text{CE}}\uparrow$ )	$t_{\text{SVDC}}$	$t_{\text{VDS}}$		2			us
Initial program pulse width	$t_{\text{WL1}}$	$t_{\text{PW}}$		0.95	1.0	1.05	ms
Additional program pulse width	$t_{\text{WL2}}$	$t_{\text{OPW}}$		2.85		78.75	ms
PROG high-voltage input setup time (to $\overline{\text{CE}}\uparrow$ )	$t_{\text{SPC}}$	—		2			us
Data output time from address	$t_{\text{DAOD}}$	$t_{\text{ACC}}$	$\overline{\text{OE}} = V_{\text{IL}}$			2	us
Data output time from $\overline{\text{OE}}\uparrow$	$t_{\text{DOOD}}$	$t_{\text{OE}}$				1	us
Data hold time (from $\overline{\text{OE}}\uparrow$ )	$t_{\text{HCOD}}$	$t_{\text{DF}}$		0		130	ns
Data hold time (from address)	$t_{\text{HAOD}}$	$t_{\text{OH}}$	$\overline{\text{OE}} = V_{\text{IL}}$	0			ns

\*: Symbol of corresponding uPD27C256A

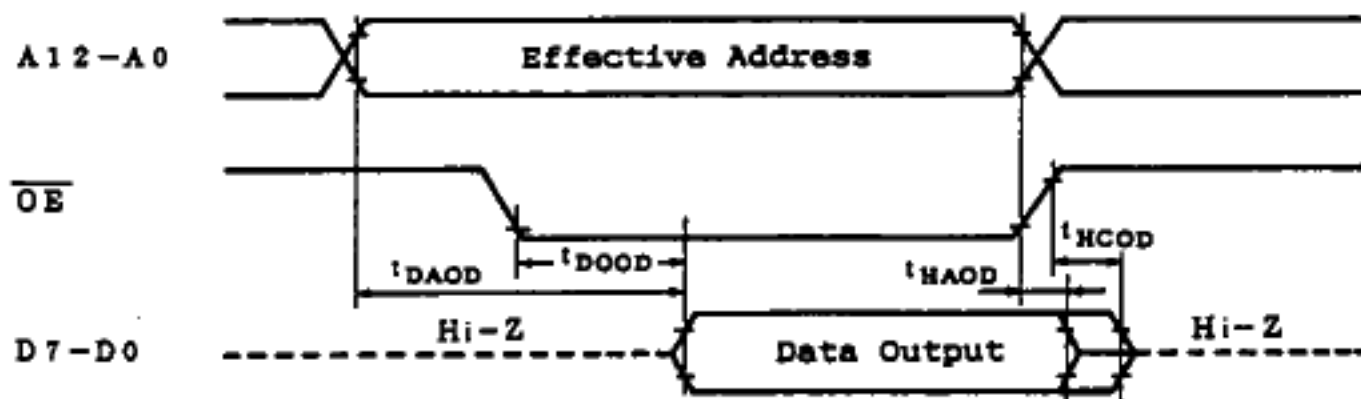
## PROM Write Mode Timings



NOTE 1: Apply  $V_{DDP}$  before  $V_{pp}$  and cut it off after  $V_{pp}$ .

2: Set  $V_{pp}$  to +13 V or less including an overshoot.

## PROM Read Mode Timings



### 11.3 ELECTRICAL SPECIFICATIONS (uPD78310A(A), uPD78312A(A))

#### Absolute Maximum Ratings (Ta = 25°C)

Parameter	Symbol	Test Conditions	Rating	Unit
Power supply voltage	$V_{DD}$		-0.5 to +7.0	V
	$AV_{REF}$		-0.5 to $V_{DD}$ +0.3	V
	$AV_{SS}$		-0.5 to +0.5	V
Input voltage	$V_I$		-0.5 to $+V_{DD}$ +0.5	V
Output voltage	$V_O$		-0.5 to $+V_{DD}$ +0.5	V
Output current low	$I_{OL}$	1 pin	4.0	mA
		All output pins total	100	mA
Output current high	$I_{OH}$	1 pin	-2	mA
		All output pins total	-25	mA
Operating temperature	$T_{opt}$		-40 to +85	°C
Storage temperature	$T_{stg}$		-65 to +150	°C

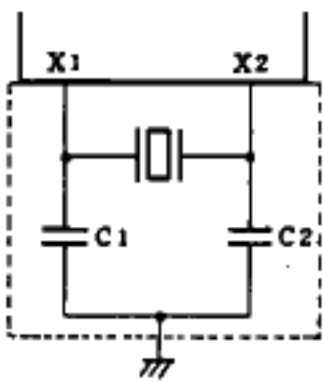
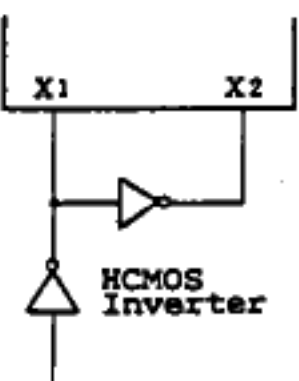
#### Recommended Operating Conditions

Oscillation Frequency	Parameter	Ta	$V_{DD}$
$4 \text{ MHz} \leq f_{XX} \leq 12 \text{ MHz}$		-40 to +85°C	+5.0 V $\pm 10\%$

Capacitance ( $T_a = 25^{\circ}\text{C}$ ,  $V_{DD} = V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions	MIN.	TYP.	MAX.	Unit
Input capacitance	$C_I$	$f = 1\text{ MHz}$			10	pF
Output capacitance	$C_O$	Unmeasured pins returned to 0 V			20	pF
I/O capacitance	$C_{IO}$				20	pF

Oscillator Characteristics ( $T_a = -40\text{ to }+85^{\circ}\text{C}$ ,  $V_{DD} = +5.0\text{ V} \pm 10\%$ ,  $V_{SS} = AV_{SS} = 0\text{ V}$ ,  $4.0\text{ V} \leq AV_{REF} \leq V_{DD}$ )

Oscillator	Recommended Circuit	Parameter	MIN.	TYP.	MAX.	Unit
Ceramic or crystal resonator		Oscillation frequency ( $f_{XX}$ )	4		12	MHz
External clock		X1 input frequency ( $f_X$ )	4		12	MHz
		X1 input rise and fall times ( $t_{XR}$ , $t_{XF}$ )	0		30	ns
		X1 input high, low-level width ( $t_{WXH}$ , $t_{WXL}$ )	30		130	ns

NOTE 1: Place the oscillator circuit as close to the X1, X2 pins as possible.

2: Do not pass other signals within a range of shaded area.

DC Characteristics ( $T_a = -40$  to  $+85^\circ\text{C}$ ,  $V_{DD} = +5.0\text{ V} \pm 10\%$ ,  
 $V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions	MIN.	TYP.	MAX.	Unit	
Input voltage low	$V_{IL1}$	Except for $\overline{EA}$	0		0.8	V	
	$V_{IL2}$	$\overline{EA}$	0		0.5	V	
Input voltage high	$V_{IH1}$	Except for P20/NMI, X1, X2, $\overline{RESET}$	2.2		$V_{DD}$	V	
	$V_{IH2}$	P20/NMI, X1, X2, $\overline{RESET}$	3.8		$V_{DD}$	V	
Output voltage low	$V_{OL}$	$I_{OL} = 2.0\text{ mA}$			0.45	V	
Output voltage high	$V_{OH}$	$I_{OH} = -1.0\text{ mA}$	$V_{DD} - 1$			V	
Input current	$I_I$	P20/NMI, $\overline{RESET}$ $0.45\text{ V} < V_I < V_{DD}$			$\pm 10$	$\mu\text{A}$	
Input leakage current	$I_{LI}$				$\pm 10$	$\mu\text{A}$	
I/O leakage current	$I_{LO}$				$\pm 10$	$\mu\text{A}$	
$AV_{REF}$ current	$AI_{REF}$	$f_{CLK} = 6\text{ MHz}$		1.5	5	mA	
$V_{DD}$ power supply current	$I_{DD1}$	Operating mode, $f_{CLK} = 6\text{ MHz}$		30	60	mA	
	$I_{DD2}$	HALT mode, $f_{CLK} = 6\text{ MHz}$		5	15	mA	
Data hold voltage	$V_{DDDR}$	STOP mode	2.5			V	
Data hold current	$I_{DDDR}$	STOP mode	$V_{DDDR} = 2.5\text{ V}$		3	30	$\mu\text{A}$
			$V_{DDDR} = 5.0\text{ V} \pm 10\%$		10	100	$\mu\text{A}$

## AC Characteristics

Read/Write Operation ( $T_a = -40$  to  $+85^{\circ}\text{C}$ ,  $V_{DD} = +5.0\text{ V}$   
 $\pm 10\%$ ,  $V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions	MIN.	MAX.	Unit
Internal system clock cycle time *1	$t_{CYK}$		166	1000	ns
Address setup time (to ALE $\uparrow$ )	$t_{SAL}$		150		ns
Address hold time (from ALE $\uparrow$ )	$t_{HLA}$	$C_L = 100\text{ pF}$ , $R_L = 2\text{ k}\Omega$ *4	30		ns
$\overline{RD}\downarrow$ delay time from address	$t_{DAR}$		233		ns
Address float time from $\overline{RD}\downarrow$	$t_{FRA}$			0	ns
Data input time from address	$t_{DAID}$			413	ns
Data input time from ALE $\uparrow$	$t_{DLID}$			233	ns
Data input time from $\overline{RD}\downarrow$	$t_{DRID}$			180	ns
$\overline{RD}\downarrow$ delay time from ALE $\uparrow$	$t_{DLR}$		63		ns
Data hold time (from $\overline{RD}\downarrow$ )	$t_{HRID}$		0		ns
Address active time from $\overline{RD}\downarrow$	$t_{DRA}$		53		ns
ALE $\uparrow$ delay time from $\overline{RD}\downarrow$	$t_{DRL}$		116		ns
$\overline{RD}$ low-level width	$t_{WRL}$		200		ns
ALE high-level width	$t_{WLH}$		126		ns
$\overline{WR}\downarrow$ delay time from address	$t_{DAW}$		233		ns

(to be continued)

(cont'd)

Parameter	Symbol	Test Conditions	MIN.	MAX.	Unit
Data output time from ALE $\downarrow$	$t_{DLOD}$			193	ns
Data output time from $\overline{WR}\downarrow$	$t_{DWOD}$			100	ns
$\overline{WR}\downarrow$ delay time from ALE $\downarrow$ *2	$t_{DLW}$		63		ns
		Refresh mode	116		ns
Data setup time (to $\overline{WR}\downarrow$ )	$t_{SODWR}$		150		ns
Data setup time (to $\overline{WR}\downarrow$ ) *3	$t_{SODWF}$	Refresh mode	33		ns
Data hold time (from $\overline{WR}\downarrow$ )	$t_{HWOD}$		20		ns
ALE $\downarrow$ delay time from $\overline{WR}\downarrow$	$t_{DWL}$		116		ns
$\overline{WR}$ low-level width	$t_{WWL}$		200		ns
		Refresh mode	116		ns

- \*1: Internal system clock ( $f_{CLK}$ ) is the oscillation clock ( $f_{XX}$ ) divided by 2 or 8 as specified by STBC register. The value in this table is the value when  $f_{XX} = 12$  MHz,  $f_{CLK} = f_{XX}/2$ .
- 2: At pulse refresh operation,  $t_{DLW}$  is one stage lower because  $\overline{WR}$  signal falls after a 1/2 clock delay.
- 3: For a pseudo-static RAM of a type that reads data by the falling edge of the  $\overline{WR}$  signal, the data setup time is not  $t_{SODWR}$ , but  $t_{SODWF}$ .
- 4: The hold time includes the time for holding  $V_{OH}$  and  $V_{OL}$  at  $C_L = 100$  pF,  $R_L = 2$  k $\Omega$  load conditions.

Remarks: This table shows the AC characteristics when the number of wait cycles is 0.



Serial Operation ( $T_a = -40$  to  $+85^{\circ}\text{C}$ ,  $V_{DD} = +5.0\text{ V} \pm 10\%$ ,  
 $V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions		MIN.	MAX.	Unit
Serial clock cycle time	$t_{\text{CYSK}}$	Output	$\overline{\text{SCK}}$ *1	1.33		us
			$\overline{\text{CTS}}$ *2	1.33		us
		Input	$\overline{\text{CTS}}$ *3	1		us
Serial clock low-level width	$t_{\text{WSKL}}$	Output	$\overline{\text{SCK}}$ *1	580		ns
			$\overline{\text{CTS}}$ *2	580		ns
		Input	$\overline{\text{CTS}}$ *3	420		ns
Serial clock high-level width	$t_{\text{WSKH}}$	Output	$\overline{\text{SCK}}$ *1	580		ns
			$\overline{\text{CTS}}$ *2	580		ns
		Input	$\overline{\text{CTS}}$ *3	420		ns
CTS high, low-level width	$t_{\text{WCSH}}$ , $t_{\text{WC SL}}$		*4	3		$t_{\text{CYK}}$
RxD setup time (to $\overline{\text{CTS}}\uparrow$ )	$t_{\text{SRXSK}}$			80		ns
RxD hold time (from $\overline{\text{CTS}}\uparrow$ )	$t_{\text{HSKRX}}$			80		ns
TxD delay time from $\overline{\text{SCK}}\uparrow$	$t_{\text{DSKTX}}$				210	ns

- \*1: I/O interface mode transmission, data transfer speed 750 kbps
- 2: I/O interface mode reception, data transfer speed 750 kbps
- 3: I/O interface mode reception, data transfer speed 1 Mbps
- 4: Asynchronous mode

A/D Converter Characteristics ( $T_a = -40$  to  $+85^\circ\text{C}$ ,  $V_{DD} = +5\text{ V} \pm 10\%$ ,  $4.0\text{ V} \leq AV_{REF} \leq V_{DD}$ ,  $AV_{SS} = V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions	MIN.	TYP.	MAX.	Unit
Resolution			8			bit
Total error*					0.4	%
Quantization error					$\pm 1/2$	LSB
Conversion time	$t_{CONV}$	$166\text{ ns} \leq t_{CYK} \leq 250\text{ ns}$	180			$t_{CYK}$
		$250\text{ ns} \leq t_{CYK} \leq 500\text{ ns}$	120			$t_{CYK}$
Sampling time	$t_{SAMP}$	$166\text{ ns} \leq t_{CYK} \leq 250\text{ ns}$	36			$t_{CYK}$
		$250\text{ ns} \leq t_{CYK} \leq 500\text{ ns}$	24			$t_{CYK}$
Analog input voltage	$V_{IAN}$		-0.3		$AV_{REF} + 0.3$	V
Analog input impedance	$R_{AN}$			1000		$M\Omega$
Reference voltage	$AV_{REF}$		4.0		$V_{DD}$	V
$AV_{REF}$ current	$AI_{REF}$	$f_{CLK} = 6\text{ MHz}$		1.5	5.0	mA

\*: Quantization error not included. Expressed as a percentage of the full-scale value.

Count Unit Operation ( $T_a = -40$  to  $+85^{\circ}\text{C}$ ,  $V_{DD} = +5.0\text{ V}$   
 $\pm 10\%$ ,  $V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions	MIN.	MAX.	Unit
CI0, CI1 high, low-level width	$t_{WCIH}$ , $t_{WCIL}$		3		$t_{CYK}$
CTRL0, CTRL1 high, low-level width	$t_{WCTH}$ , $t_{WCTL}$		3		$t_{CYK}$
CTRL0, CTRL1 setup time (to CI $\uparrow$ )	$t_{SCTCI}$	Specifies count unit operation mode 3, and CI pin input rising edge effective.	2		$t_{CYK}$
CTRL0, CTRL1 hold time (from CI $\uparrow$ )	$t_{HCICT}$	Specifies count unit operation mode 3, and CI pin input rising edge effective.	5		$t_{CYK}$
CLR0, CLR1 high, low-level width	$t_{WCRH}$ , $t_{WCRL}$		3		$t_{CYK}$
CI0, CI1 setup time (to CI $\uparrow$ )	$t_{S4CTCI}$	Specifies count unit operation mode 4	6		$t_{CYK}$
CI0, CI1 hold time (from CI $\uparrow$ )	$t_{H4CICT}$	Specifies count unit operation mode 4	6		$t_{CYK}$
CI0, CI1, CTRL0, CTRL1 cycle time	$t_{CYC4}$	Specifies count unit operation mode 4	4		us

Other Operations ( $T_a = -40$  to  $+85^{\circ}\text{C}$ ,  $V_{DD} = +5.0\text{ V} \pm 10\%$ ,  
 $V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions	MIN.	MAX.	Unit
NMI high, low-level width	$t_{WNIH}$ , $t_{WNIL}$		10		us
INTE0 high, low-level width	$t_{WIOH}$ , $t_{WIOI}$		3		$t_{CYK}$
INTE1 high, low-level width	$t_{WI1H}$ , $t_{WI1L}$		3		$t_{CYK}$
INTE2 high, low-level width	$t_{WI2H}$ , $t_{WI2L}$		3		$t_{CYK}$
$\overline{\text{RESET}}$ high, low-level width	$t_{WRSH}$ , $t_{WRSL}$		10		us
$V_{DD}$ rise time (for using SBF bit)	$t_{ROVD}$		4		ms
$V_{DD}$ rise, fall time	$t_{RVD}$ , $t_{FVD}$		200		us

External Clock Timing ( $T_a = -40$  to  $+85^{\circ}\text{C}$ ,  $V_{DD} = +5.0\text{ V} \pm 10\%$ ,  
 $V_{SS} = 0\text{ V}$ )

Parameter	Symbol	Test Conditions	MIN.	MAX.	Unit
X1 input high-level width	$t_{WXH}$		30	130	ns
X1 input low-level width	$t_{WXL}$		30	130	ns
X1 input rise time	$t_{XR}$		0	30	ns
X1 input fall time	$t_{XF}$		0	30	ns
X1 input cycle time	$t_{CYX}$		83	250	ns

$t_{CYK}$  Dependent Bus Timing Definition

Parameter	Expression	MIN./MAX.	Unit
$t_{SAL}$	1.5 T-100	MIN.	ns
$t_{DAR}$	2 T-100	MIN.	ns
$t_{DAID}$	(3.5+n) T-170	MAX.	ns
$t_{DLID}$	(2+n) T-100	MAX.	ns
$t_{DRID}$	(1.5+n) T-70	MAX.	ns
$t_{DLR}$	0.5 T-20	MIN.	ns
$t_{DRL}$	T-50	MIN.	ns
$t_{DRA}$	0.5 T-30	MIN.	ns
$t_{WRL}$	(1.5+n) T-50	MIN.	ns
$t_{WLH}$	T-40	MIN.	ns
$t_{DAW}$	2 T-100	MIN.	ns
$t_{DLOD}$	0.5 T+110	MAX.	ns
$t_{DLW}$	0.5 T-20 (normal operation)	MIN.	ns
	T-50 (refresh mode)	MIN.	ns
$t_{SODWR}$	(1.5+n) T-100	MIN.	ns
$t_{SODWF}$	0.5 T-50	MIN.	ns
$t_{DWL}$	T-50	MIN.	ns
$t_{WWL}$	(1.5+n) T-50 (normal operation)	MIN.	ns
	(1+n) T-50 (refresh mode)	MIN.	ns

Remarks 1: n is the number of wait cycles inserted by MM register specification.

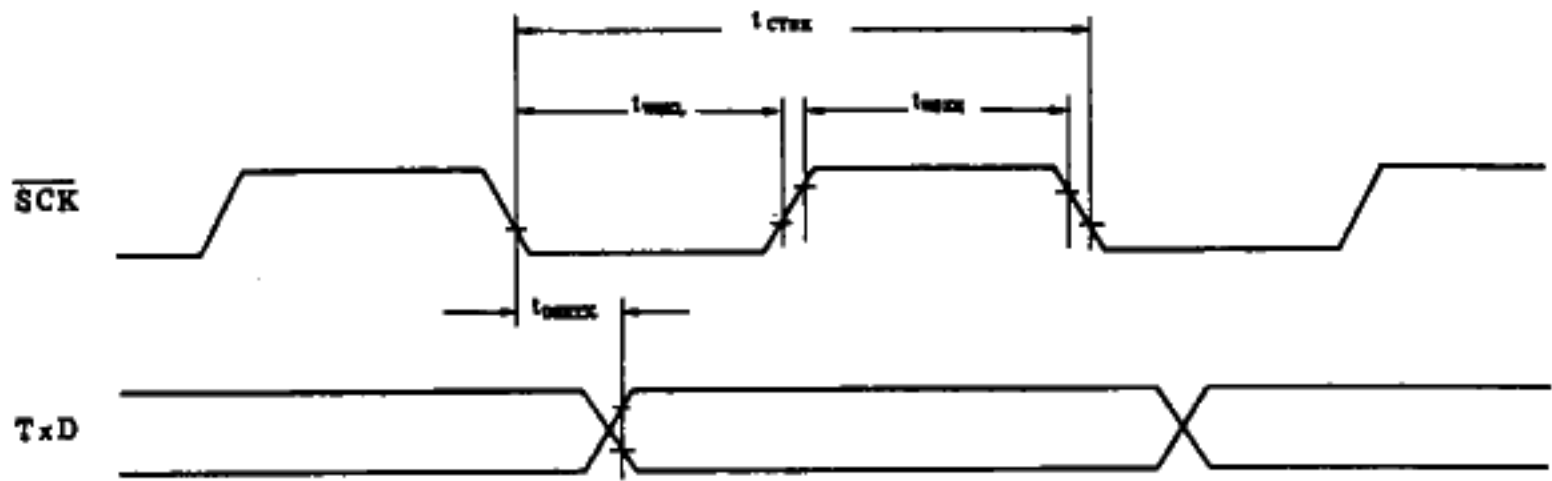
2:  $T = t_{CYK} = 1/f_{CLK}$  ( $f_{CLK}$ : Internal system clock frequency)

3: Items not in this table does not depend on the internal system clock frequency ( $f_{CLK}$ ).

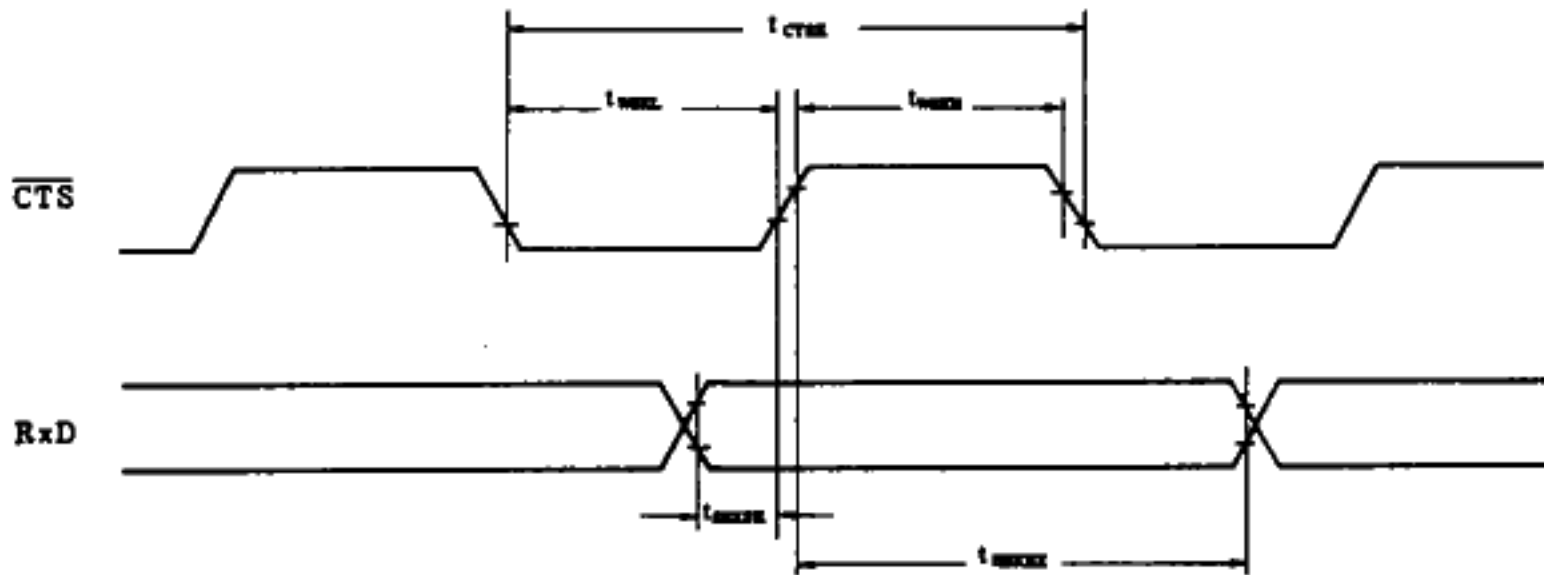


## Serial Operation

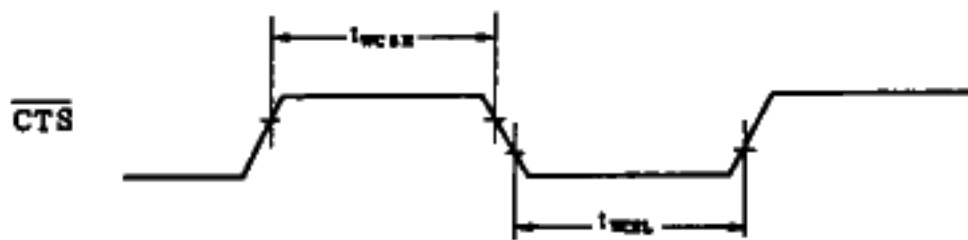
### I/O interface mode transmission:



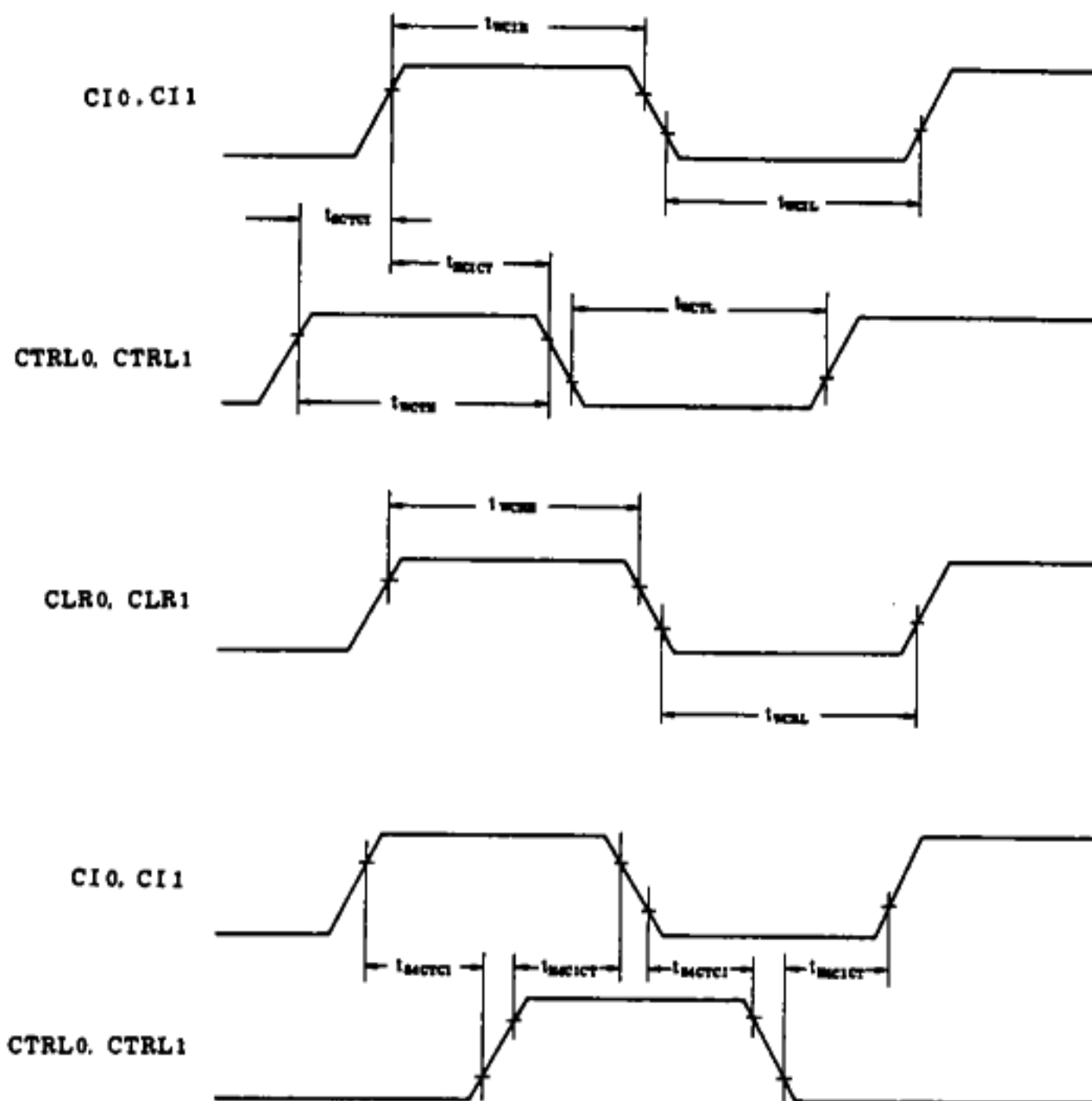
### I/O interface mode reception:



### Transmit enable input timing (asynchronous mode):

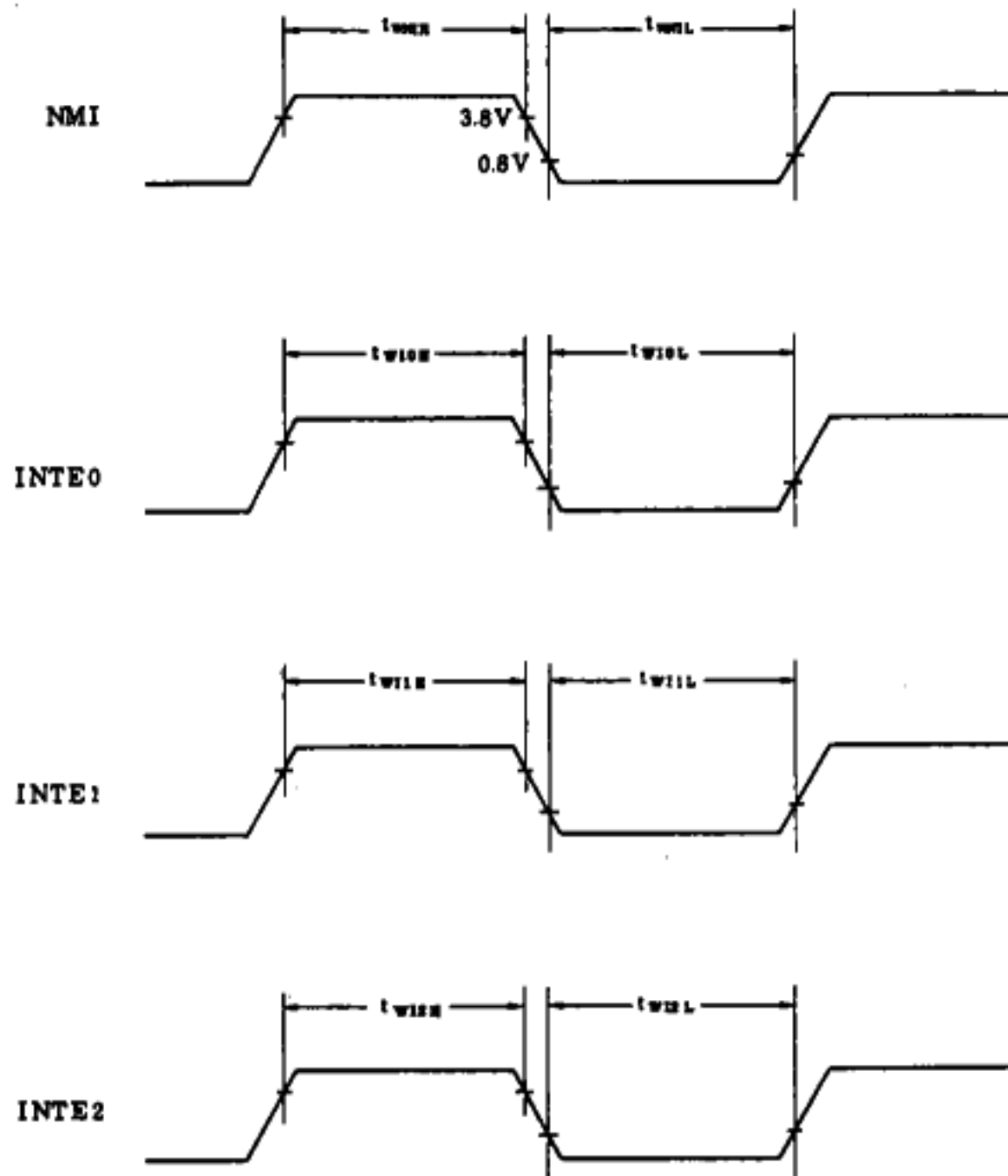


# Count Unit Input Timing

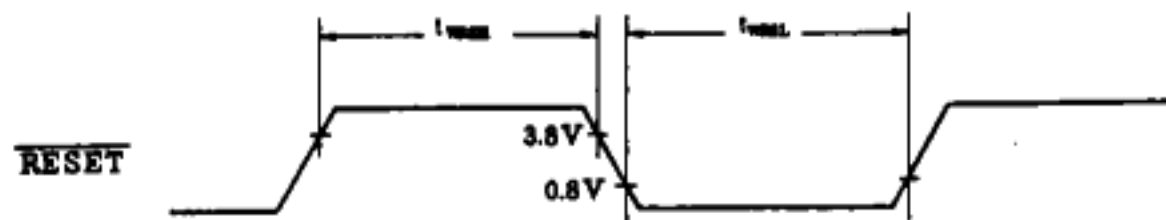




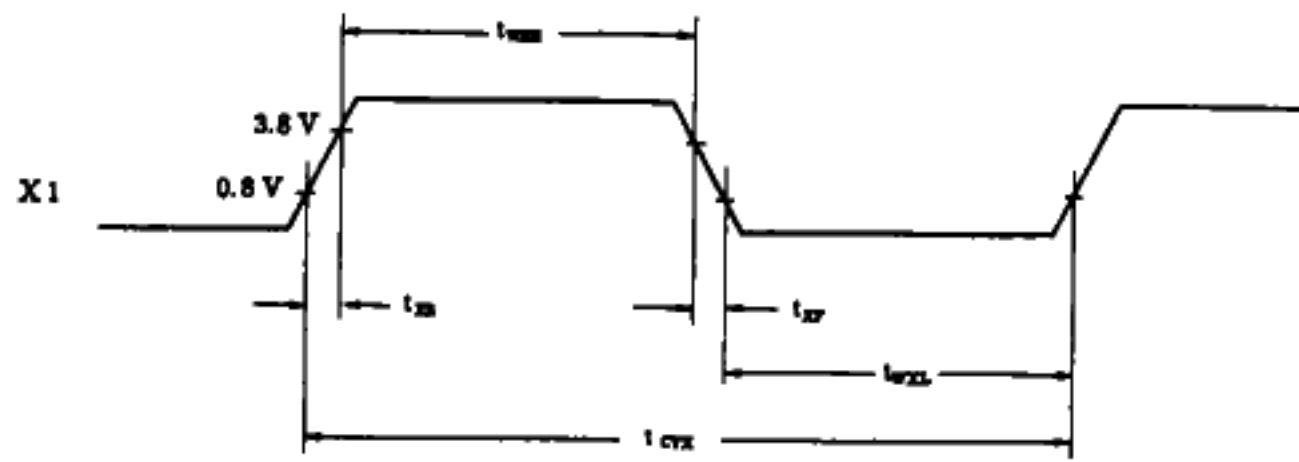
## Interrupt Input Timing



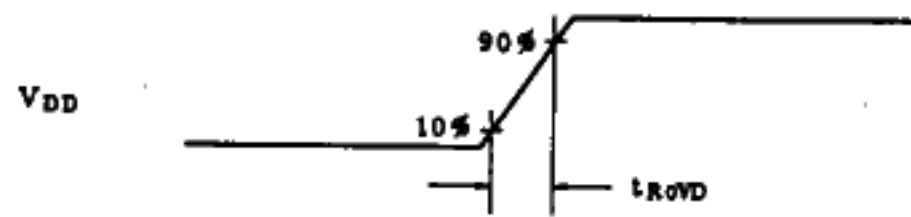
## Reset Input Timing



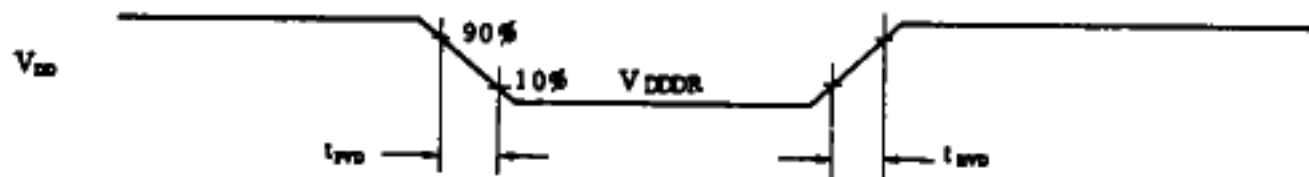
## External Clock Timing



## Power On Timing

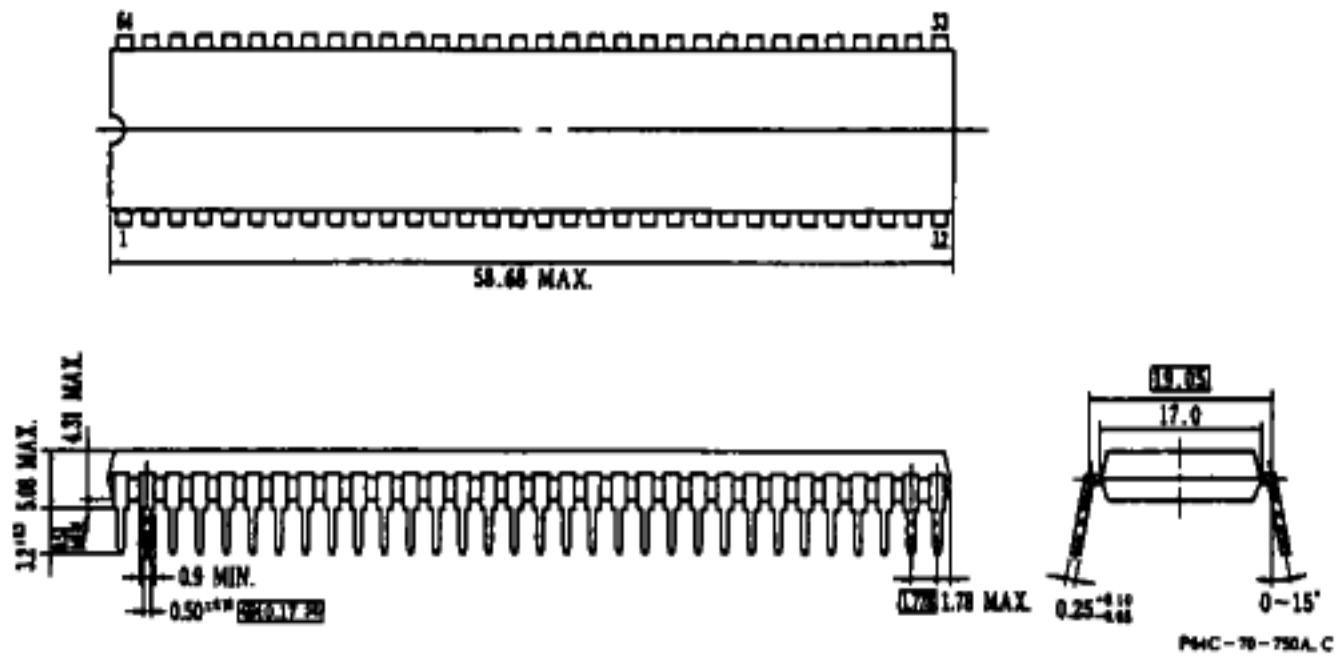


## Data Hold Timing

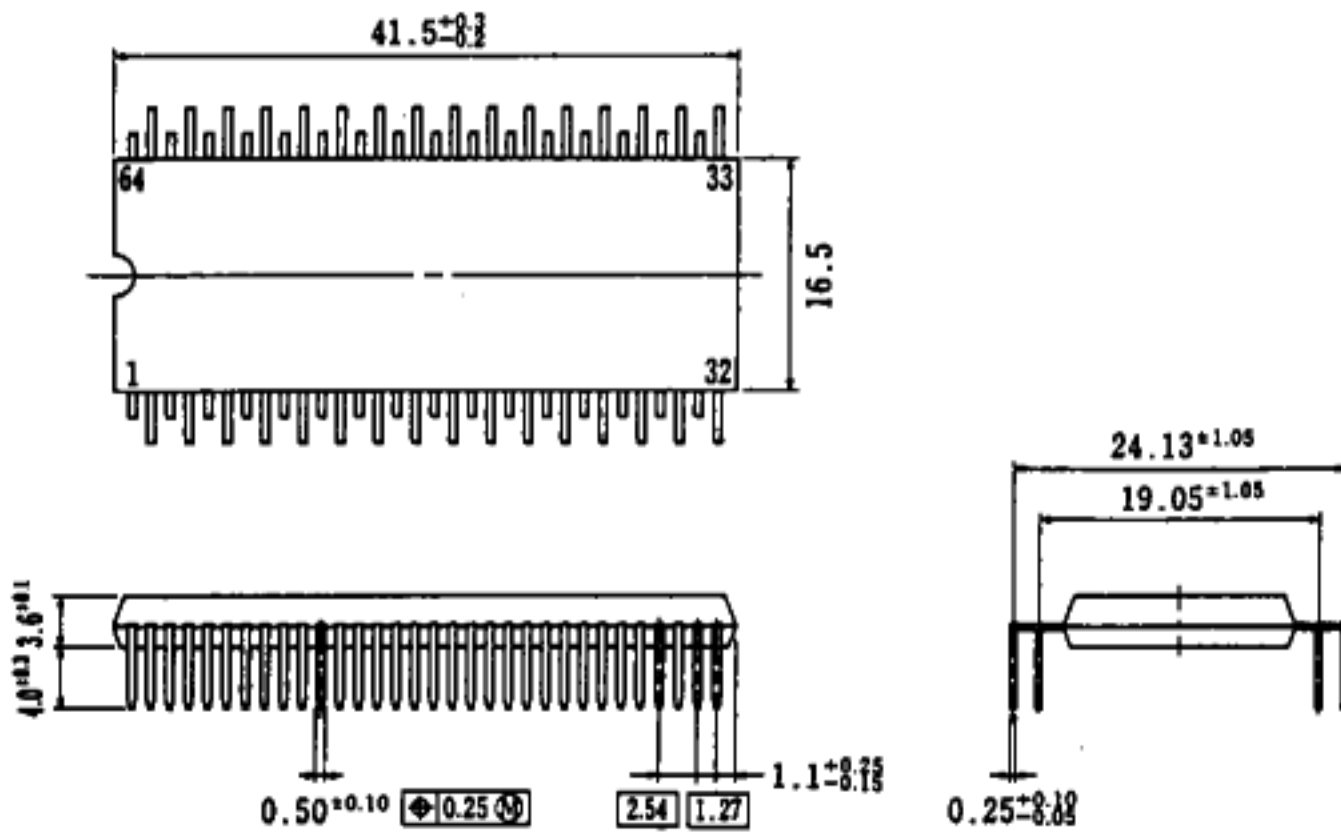


11.4 PACKAGE INFORMATION

64-pin plastic shrink DIP (750 mil) (unit: mm)

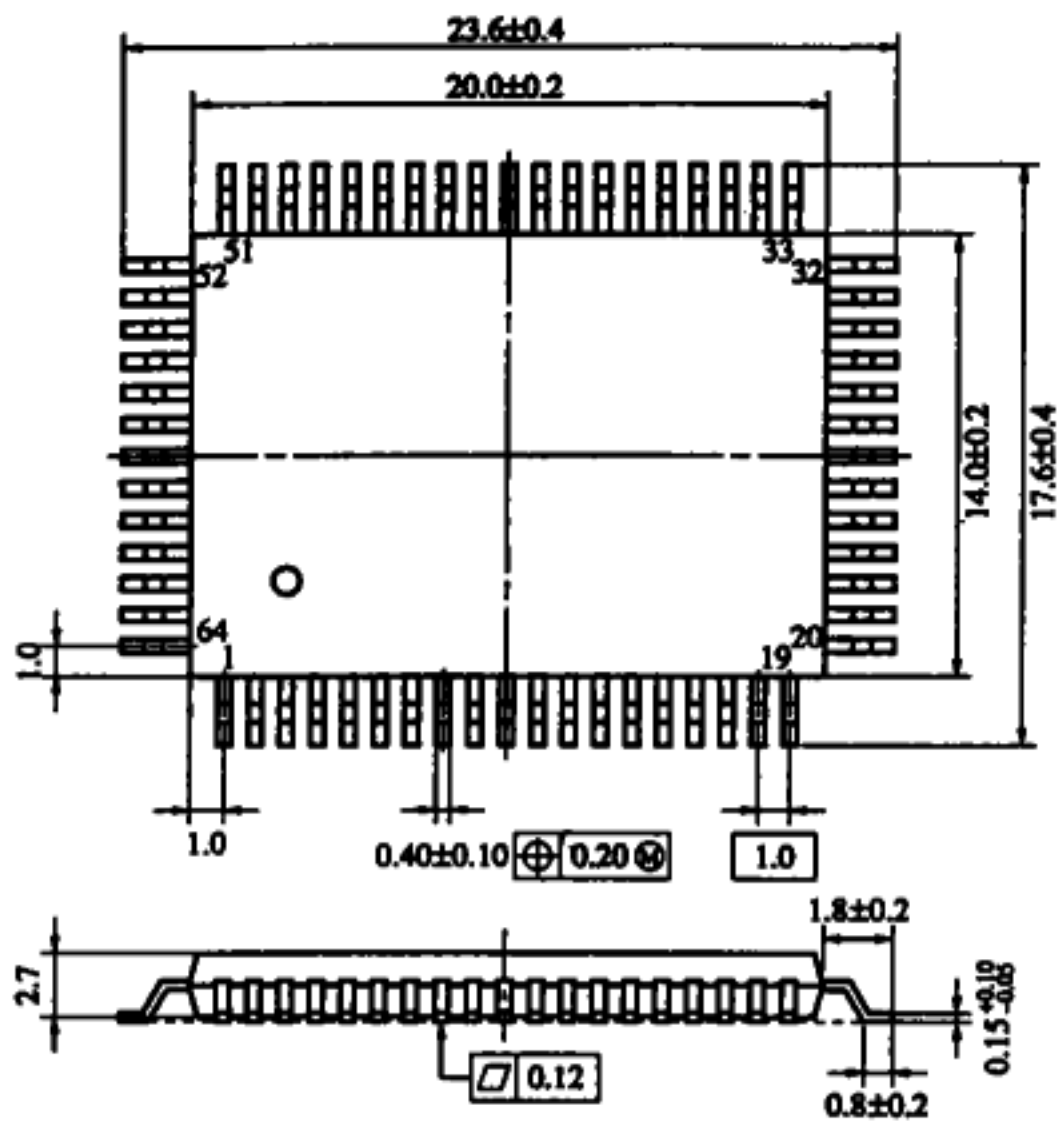


64-pin plastic QUIP (unit: mm)

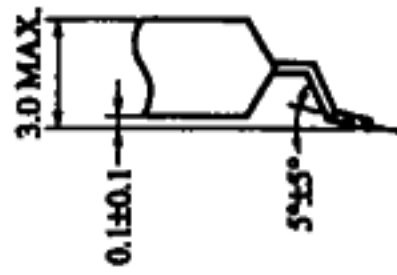


P64CQ-100-36

64-pin plastic QFP (14 x 20) (unit: mm)

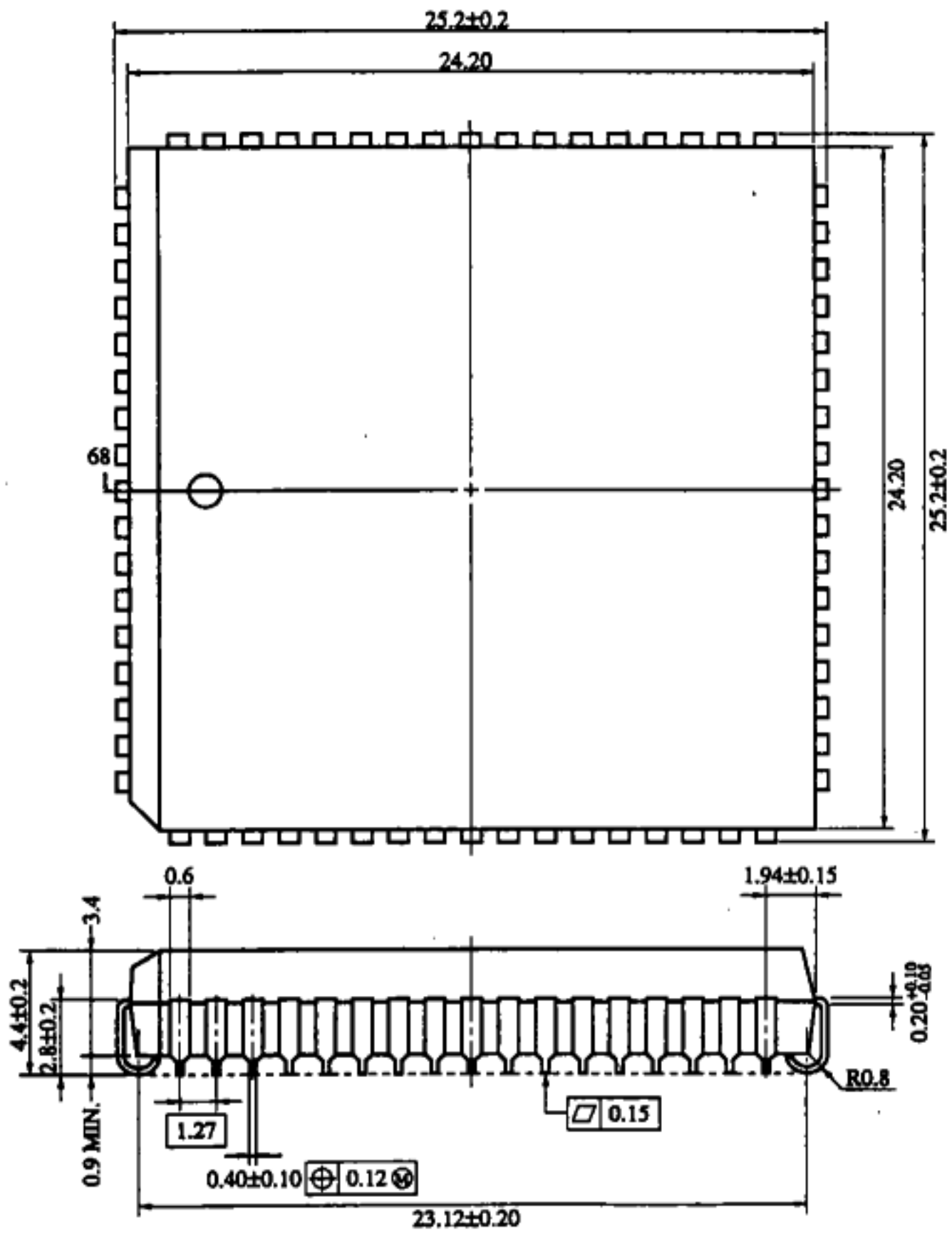


Detail Drawing of Pin Tip Shape



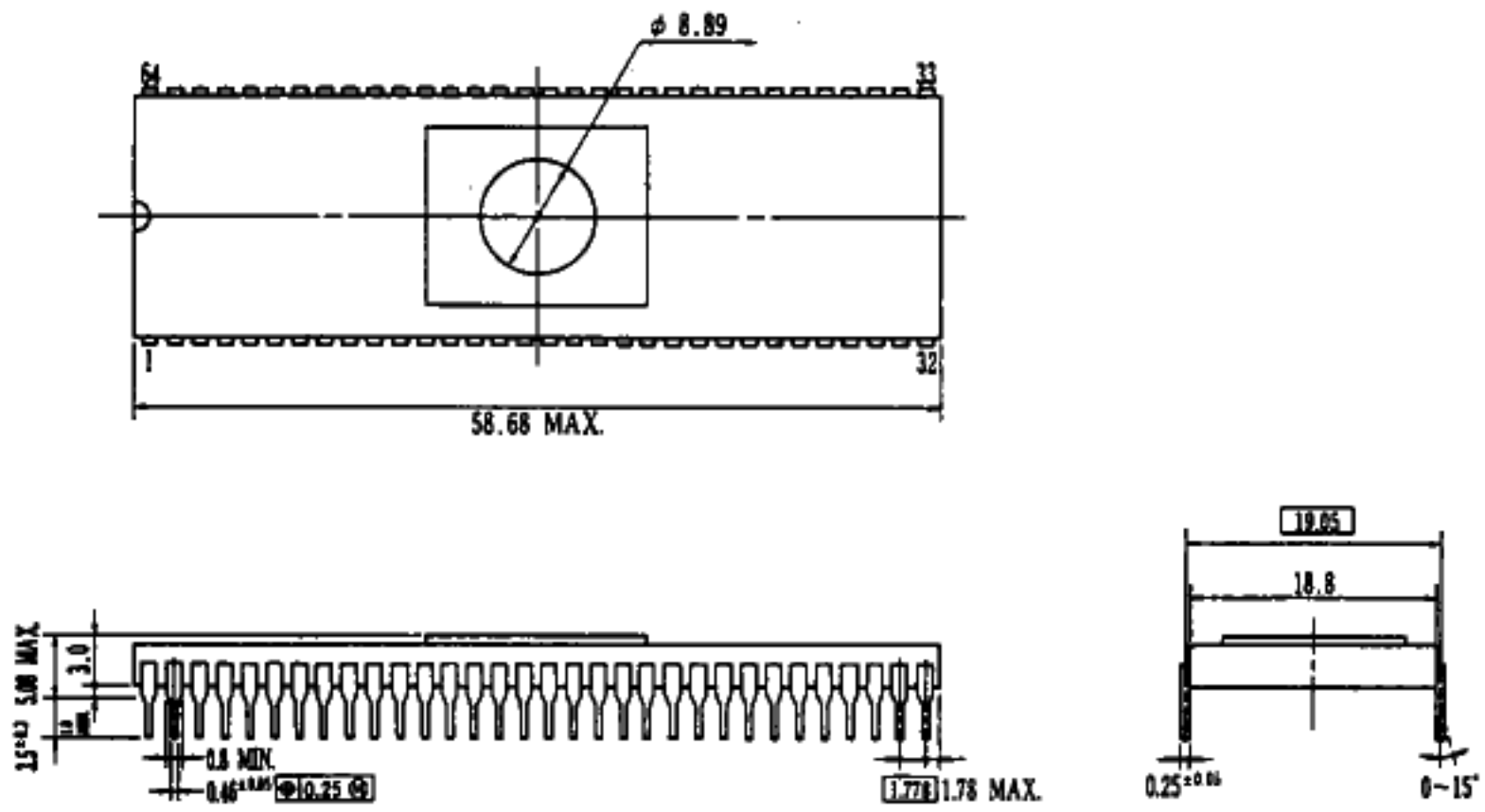
P64GF-100-3B8,3BE,3BR-1

68-pin plastic QFJ ( $\square$  950 mil) (unit: mm)



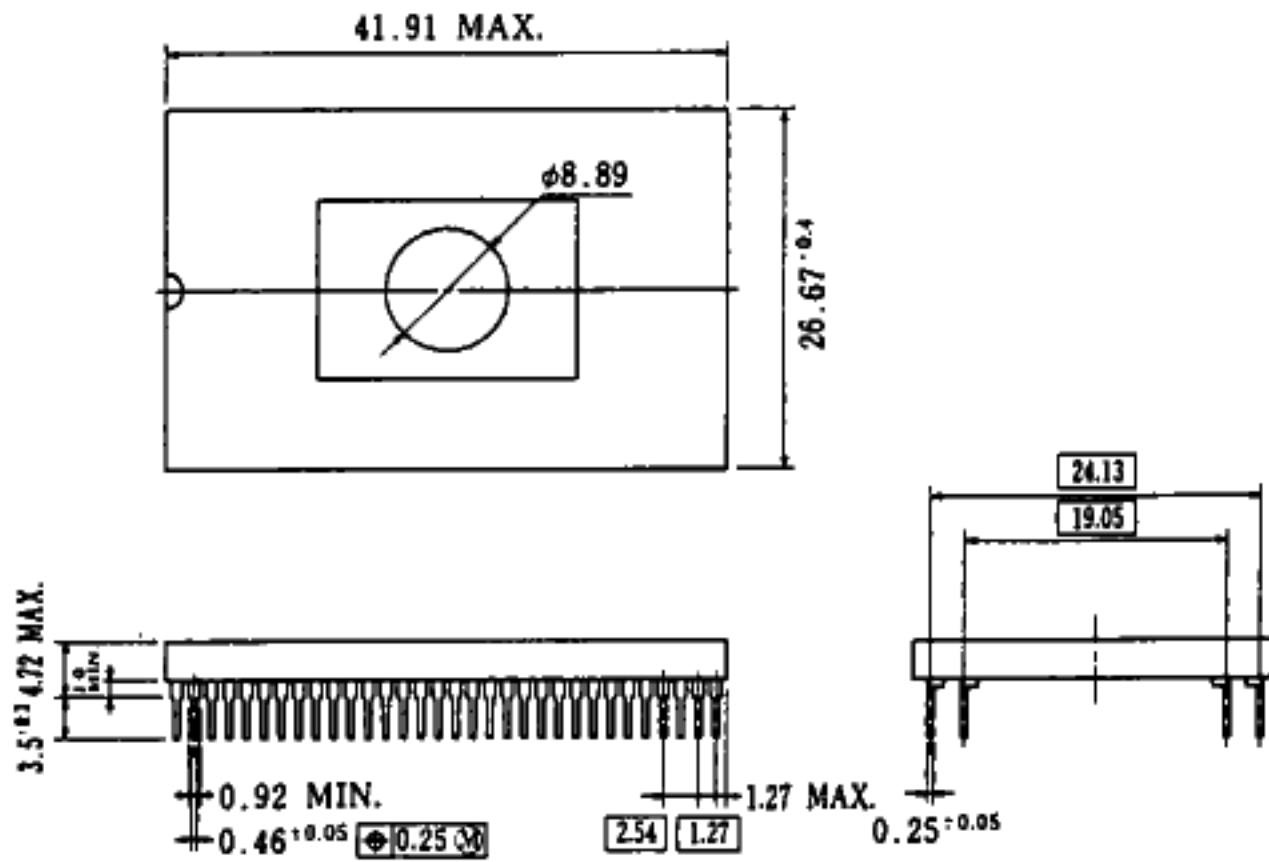
P68L-50A1-2

64-pin ceramic shrink DIP (750 mil) (unit: mm)



P64DW-70-750A

64-pin ceramic QUIP (with window) (unit: mm)



P64RQ-100-A

## CHAPTER 12. PRECAUTIONS FOR USE

### 12.1 PRECAUTIONS REGARDING SERIAL COMMUNICATION INTERFACE FUNCTION

When transmitting data in asynchronous mode, be sure not to change the  $\overline{\text{CTS}}$  pin input signal level or the TXRDY bit (bit 7) in the serial communication mode register.

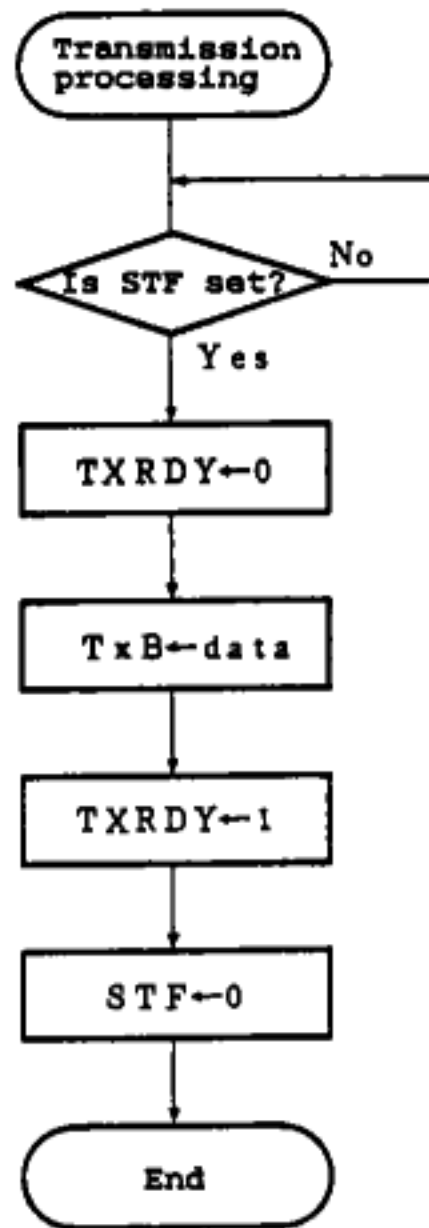
When the  $\overline{\text{CTS}}$  pin input level changes from low to high, or when the TXRDY is set (1), extra transmission complete interrupt may be generated, resulting in duplicate writing of data to the transmit buffer register (TxB) and loss of the transmit data.

The following procedure is recommended to avoid this problem.

#### [Remedies]

- (1) Fix the  $\overline{\text{CTS}}$  pin input level low and leave the TXRDY bit set (1) during serial communication.
- (2) Perform polling of the transmission complete interrupt by software.

Figure 12-1 Flowchart of Polling Processing





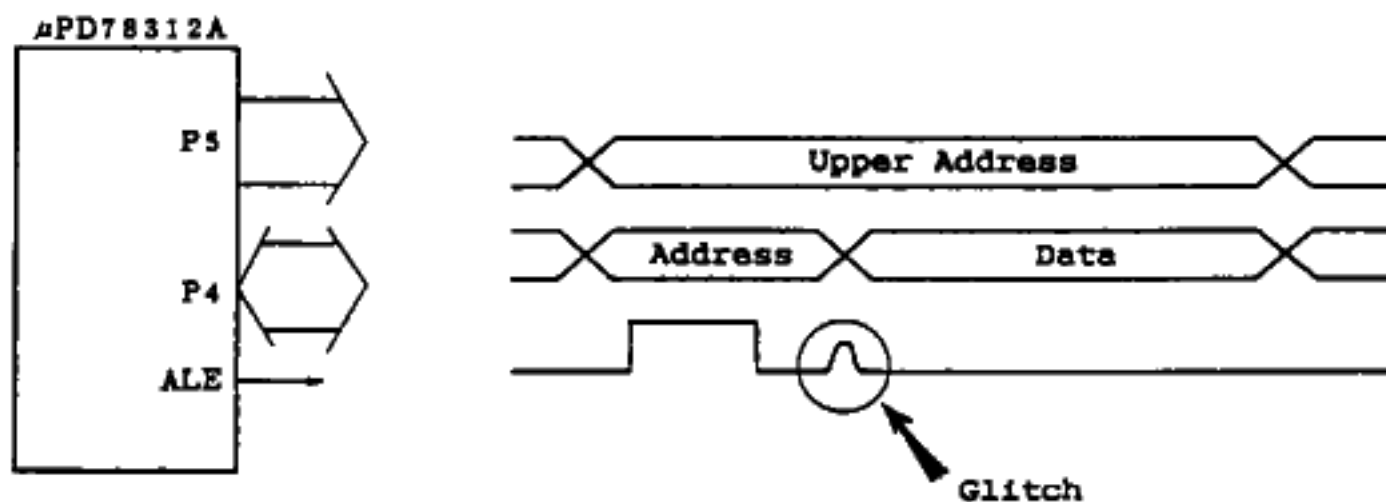
## 12.2 PRECAUTIONS FOR EXPANSION BY THE USE OF EXTERNAL DEVICES

In an access to an external memory using  $\mu$ PD78310A, 78312A or 78P312A, a glitch output of around 2.0 V max. may come out of the ALE pin depending upon the type of the application system.

### (1) Glitch output

The glitch output is likely to occur at a timing at which the address/data multiplexed bus (port 4) changes from the output state of address FFH to that of data 00H. Due to this glitch, the address latch circuit may malfunction to latch in the data as the low-order address part, thus causing a defect in the application product.

Figure 12-2 Glitch Output of ALE Pin



### (2) Measures

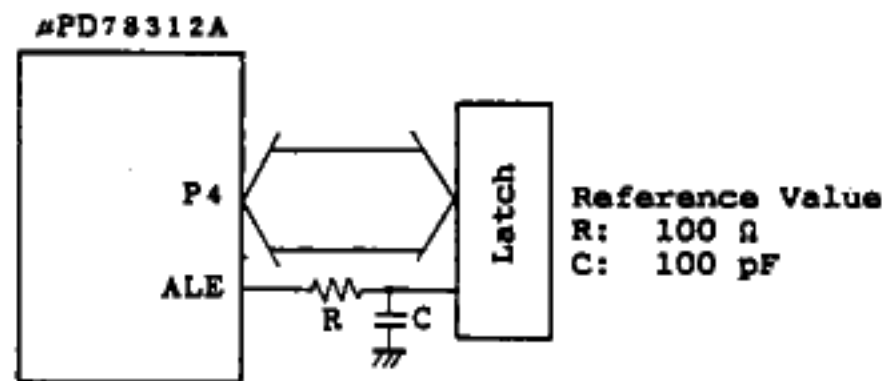
In order to suppress the glitch output down to a nonproblematic level for the application system, care should be paid to the following points in device installation.

- ① Reinforcement of power supply grounding pattern (e.g. use of multilayer board)

- ② Direct installation of device to board, not socket insertion
- ③ Reduction of load capacitance on bus

What is more, as the most direct measure, use of a countermeasure circuit as shown in Figure 12-3 is effective to avoid any malfunctioning.

Figure 12-3 Example of Countermeasure Circuit



(3) Possible causes

A glitch becomes more likely to occur mainly on account of following factors.

① Factor due to device

Switching noise becomes more likely to occur due to higher operation speed.

② Factor due to system

- . The greater the load capacitance on the bus, the more the momentary charge shift. Thus, glitch becomes more likely to occur.
- . The higher the impedance of the power line, the more likely the glitch output.

### 12.3 Precautions Regarding Instruction Combinations

If an instruction which accesses an saddr area is executed immediately after a specific special function register(SFR) has been manipulated by an arithmetic/logic operation instruction, the SFR area(FF20H to FFFFH) is accessed instead of the saddr area(FE20H to FEFFH) specified by the operand.

This operation is not caused by a specific condition such as the supply voltage or operating frequency, but only by a combination of instructions.

For instance, if instructions are structured as shown in the example, an access is made to 0FF21H.0H, and not to 0FE21H.0H. To avoid this problem, an NOP instruction must be inserted after an arithmetic/logic operation instructions.

As one remedy, use of a Ver. 4.10 or later assembler is recommended. Ver. 4.10 and later assemblers detect the instruction combination, and insert an automatically generated NOP instruction after an arithmetic/logic operation instruction (see the example below).

```

Example: Source file      ⋮

                        ADDW CR11, #01H
                        BF  OFE21H.OH, $JMP
                        ⋮

                        JMP:
                        ⋮

                        Assembler
                        (Ver.4.10)  ↓

Object module file      ⋮

                        ADDW CR11, #01H
                        NOP                               ; Automatically inserted
                        BF  OFE21H.OH$JMP                 NOP instruction
                        ⋮

                        JMP:
                        ⋮

```

When the assembler executes this remedial action, a message is displayed which indicates that an NOP instruction has been inserted in the assembly list file(\*.PRN). An indication is also given of the total number of NOP instructions inserted.

Tables 12-1 to 12-3 show the SFRs, arithmetic/logic operation instructions, and saddr instructions concerned.

Table 12-1 Relevant sfr's and sfrp's

sfr	CR11L	(OFF0EH)
	UDC1L	(OFF1EH)
	CCW	(OFF4EH)
	MD1L	(OFF8EH)
	EXTSFR14*	(OFFBEH)
	TMICO	(OFFCEH)
	STIC	(OFFDEH)
sfrp	CR11	(OFF0EH)
	UDC1	(OFF1EH)
	MD1	(OFF8EH)

\*: External SFR

Remarks: Addresses are shown in ( ).

Table 12-2 Relevant Instructions

Mnemonic	Operands
XCH	A, sfr
ADD ADDC SUB SUBC AND OR XOR	sfr, #byte
XCHW	AX, sfrp
ADDW SUBW	sfrp, #word

Table 12-3 saddr Instructions

Mnemonic	Operand
MOV ADD ADDC SUB SUBC AND OR XOR CMP	saddr, #byte
MOV XCH ADD ADDC SUB SUBC AND OR XOR CMP	A, saddr
MOV	saddr, A
MOV XCH ADD ADDC SUB SUBC AND OR XOR CMP	saddr, saddr
MOV XCH	A, [saddrp]
MOV	[saddrp], A
MOVW ADDW SUBW CMPW	saddrp, #word

Mnemonic	Operand
MOVW XCHW ADDW SUBW CMPW	AX, saddrp
MOVW	saddrp, AX
MOVW XCHW ADDW SUBW CMPW	saddrp, saddrp
INC DEC	saddr
INCW DECW	saddrp
MOV1 AND1 OR1 XOR1	CY, saddr. bit
AND1 OR1	CY, /saddr. bit
MOV1	saddr. bit, CY
SET1 CLR1 NOT1	saddr. bit
BT BF BTCLR BFSET	saddr. bit, \$saddr16
DBNZ	saddr, \$saddr16

12.4 POP PSW INSTRUCTION RELATED CAUTION □

If an interrupt is generated (an interrupt is acknowledged and an interrupt request is issued to the CPU) immediately after execution of the POP PSW instruction, and the next instruction is of 2 bytes or more, the CPU may overrun. In order to avoid this, insert a 1-byte instruction such as an NOP instruction after the POP PSW instruction.

Whether or not this bug occurs depends on the internal state of the instruction prefetch queue before execution of the POP PSW instruction.

Example:

```

      :
      :
MOVW  RP2, #1234H
POP   PSW      ← Even if an interrupt request is
                generated, the interrupt is not
                serviced.

CLR1  CRIC11.6 ← An illegal sfr address is
                generated and executed without
                accessing CRIC11 (FFC6H).
      :
      :
```



<Remedy>

```

      :
      :
MOVW  RP2, #1234H
POP   PSW      ← If an interrupt request is
                generated, the interrupt is
                serviced.

NOP      ← A 1-byte instruction such as an
                NOP instruction is inserted.

CLR1  CRIC11.6 ← Normal sfr access is performed.
      :
      :
```

This chapter puts together the cautions described in each chapter. Please use this for application product design. The page number in the parentheses indicates the page relevant to each caution.

### 13.1 CHAPTER 3 "CPU ARCHITECTURE" RELATED CAUTION

- (1) Any of addresses FF00H to FFFFH to which SFR is not allocated cannot be accessed. Attempting to access it may cause misoperation. (3-13)

### 13.2 CHAPTER 4 "PERIPHERAL HARDWARE FUNCTIONS" RELATED CAUTION

- (1) When the digital input/output port is used in the input mode or control signal input/output mode, writing to the output latch by a bit manipulation instruction makes the output latch contents undefined for all bits. (4-11)
- (2) When using the system clock oscillator, wiring should be performed as follows to avoid effects of wiring capacitance, etc. (4-16)
  - . Use the shortest possible wiring.
  - . Wiring should not cross other signal lines or be placed close to a varying high current.
  - . The grounding point potential of the oscillator capacitor should always be the same as  $V_{SS}$ . It should not be grounded to a ground pattern where a varying high current flows.
  - . Do not fetch a signal from the oscillator.



- (3) The STOP mode should not be set in a system using the external clock. (4-16)
- (4) The CR00 (CR10) register and CR01 (CR11) register can be set to the compare preset mode/capture mode independently by specification of the capture/compare register control register (CRC). (4-27)
- (5) Timer operation external trigger control by the timer control register (TMC0) TRG bit is valid only in the interval timer mode. The one-shot timer cannot be started by an external trigger. (4-51)
- (6) When the TMC0 (TMC1) register TS0 (TS1) bit is set (1), if the TS0 (TS1) bit is set again, the MD0 (MD1) register value is set in the TMO (TM1) register again and the timer is restarted. (4-54)
- (7) When the TMO (TM1) register is used as an interval timer, if the timer is started with 0000H written to the TMO (TM1) register, the following operation occurs in the first countdown.

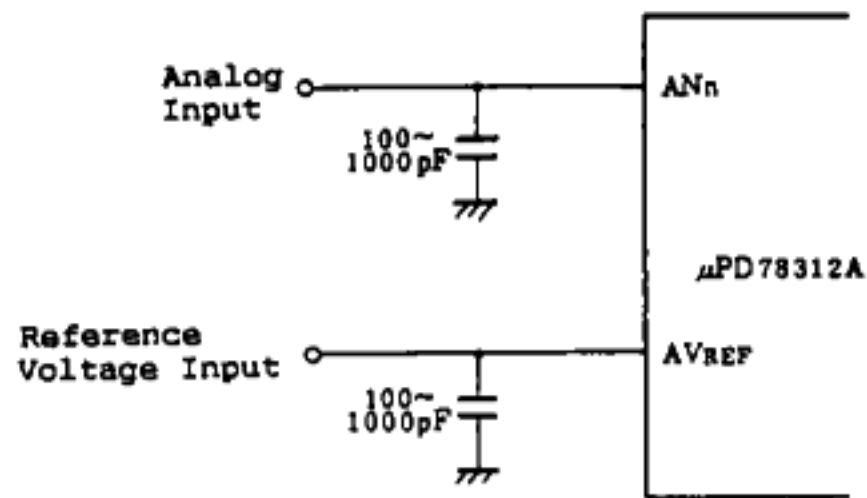
- . Generation of a timer underflow interrupt request
- . Inversion of the output level of the T00 (T01) pin  
(ENT00 (ENT01) = 1)

Therefore, when setting intervals, it is necessary to write a value other than 0000H to the timer register (TMO, TM1) before the timer is started.

Note that after RESET input, the TMO (TM1) register is undefined. (4-55)

- (8) If port 0 is specified as an 8-bit real-time output port, 8-bit data is written to POH or POL by accessing either POH or POL. (4-65)
- (9) If the serial interface transfer mode is switched from the I/O interface mode to the asynchronous mode, or from the asynchronous mode to the I/O interface mode, ensure that the reception is disabled (RXE bit = 0) before switching the mode. (4-69)
- (10) If the serial interface transfer mode is set to the asynchronous mode, be sure to set the CTS/P27 pin to the control mode (set port 2 mode control register (PMC2) bit 7 (PMC27) to 1). If it is set to the port mode, the serial communication interface cannot be achieved. (4-70)
- (11) When the serial communication mode register (SCM) (in the I/O interface mode) RSCK bit is 0 (external clock reception mode), do not set the reception clock output trigger bit (TSK) to 1. If the TSK bit is set to 1, the reception clock counter is reset unconditionally, preventing a normal count of the external clock. (4-71)
- (12) If the serial interface transfer mode is set to the I/O interface mode, set two or more values in the baud rate generator (BRG). (The transfer baud rate is 750 Kbps with the internal system clock of 6 MHz, BRG = 2). (4-77)
- (13) Ensure that a voltage outside the range of  $AV_{SS}$  to  $AV_{REF}$  is not applied to pins AN3 to AN0 irrespective of whether A/D conversion is used or not. (4-87)

- (14) Connect a capacitor to the analog input pins (AN3 to AN0) and reference voltage input pin (AV<sub>REF</sub>) to avoid misoperation due to noise. (4-87)



- (15) The time immediately after the time base mode register (TBM) setting until generation of the first interrupt request is undefined. (4-95)

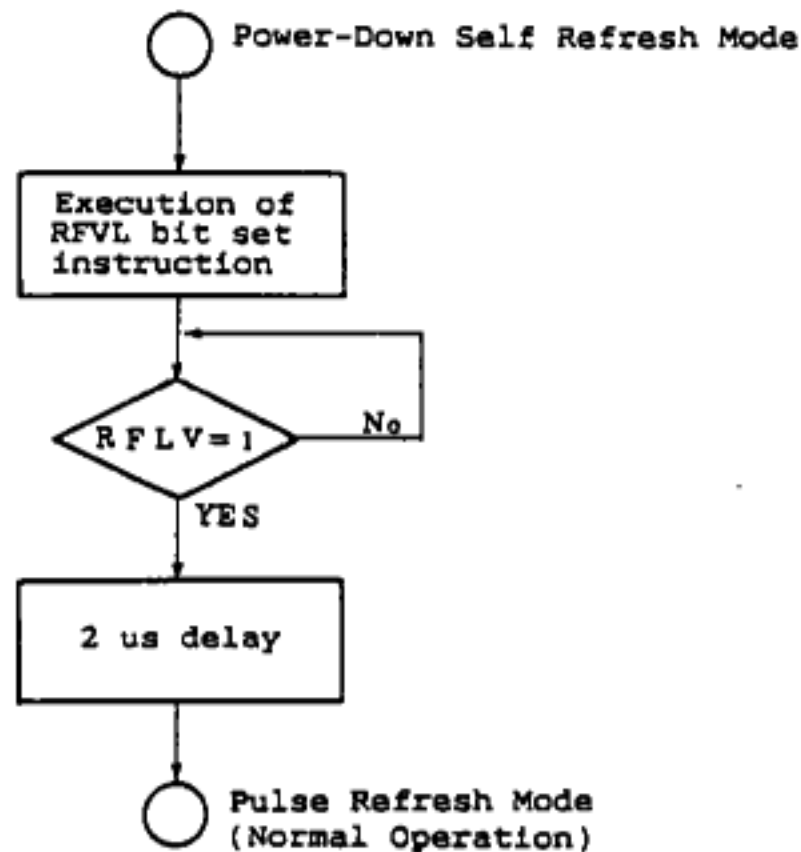
### 13.3 CHAPTER 6 "STANDBY FUNCTION" RELATED CAUTION

- (1) If the supply voltage rise time upon powering-on exceeds the specified range, the standby control register (STBC) SBF bit may not be cleared. Therefore, when using the SBF bit, ensure that the supply voltage rise time is within the specified range. (6-4)
- (2) If the STOP mode is set, the X1 pin is internally shorted to V<sub>SS</sub> (ground potential) in order to suppress the leakage in the clock generator. Therefore, the use of the STOP mode is prohibited in a system using the external clock. (6-6)

### 13.4 CHAPTER 8 "LOCAL BUS INTERFACE" RELATED CAUTION

- (1) When setting the power down self refresh mode, ensure that the pseudo-static RAM CE pin is not activated. (8-11)

- (2) In the internal hardware, the RFLV bit is set (1) in synchronization with the time base counter tap output even if the refresh mode register (RFM) RFLV bit is set (1). Therefore, when restoring from the power down refresh operation, the following operation by software is required. (8-12)



### 13.5 CHAPTER 9 "uPD78P312A PROGRAMMING" RELATED CAUTION

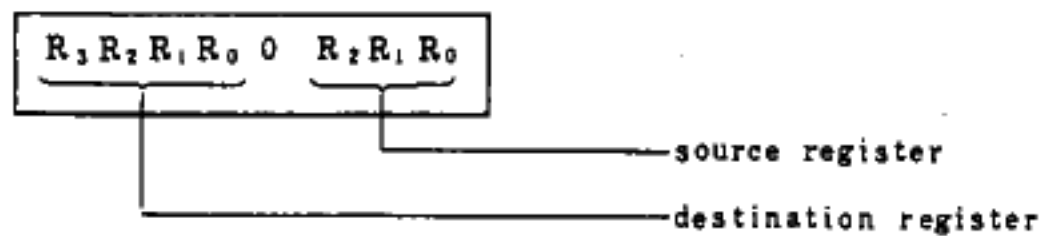
- (1) Extend a shielding cover film over the uPD78P312A with an erasure window except for EPROM erasure. (9-1)
- (2) UV erasure cannot be applied to the one-time PROM version uPD78P312A which is not provided with an erasure window. (9-1)
- (3) Driving both CE and OE low is prohibited if  $V_{pp}$  and  $V_{DD}$  are set to +12.5 V and +6 V, respectively during PROM programming. (9-2)

13.6 CHAPTER 10 "INSTRUCTION SET" RELATED CAUTION

(1) If both the source and destination are registers or `saddr`, `saddrp` in an operand field such as `MOV r, r1` or `ADD saddr, saddr`, the code is as follows. (10-40)

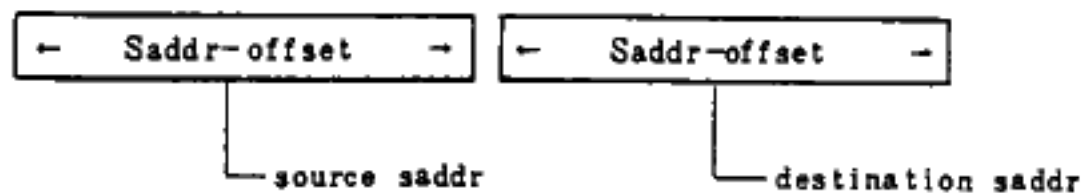
- . In the case of registers, a destination specification code comes first, followed by a source specification code (the same applies to a register pair).

Example:



- . In the case of `saddr`, `saddrp`, the code comprises offset data whose first 1-byte data specifies a source, and second 1-byte, a destination.

Example:



(2) If the special function register (SFR) mapped at `FF00H` to `FF1FH` is written as operand `sfr`, `sfrp`, not the SFR addressing but the short direct addressing is applied and the operation code generated is the one for an instruction whose operand is `saddr`, `saddrp`. (10-40)

**Example:**

AND A, P5

operation codes

10011100

00000101

AND A, PM5

operation codes

00000001

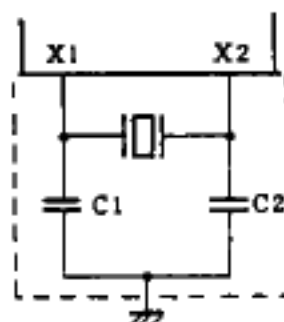
10011100

00100101

In this case, the short direct addressing is applied to the AND A, P5 instructions and therefore the operation code is shorter than that in the SFR addressing.

**13.7 CHAPTER 11 "SPECIFICATIONS" RELATED CAUTION**

- (1) The oscillator should be located as close as possible to the X1 and X2 pins. (11-2, 11-18, 11-37)
- (2) No other signal lines should cross the area enclosed by a dotted line. (11-2, 11-18, 11-37)



- (3) In the PROM write mode,  $V_{DDP}$  should be applied before  $V_{PP}$  and cut after  $V_{PP}$ . (11-35)
- (4) In the PROM write mode,  $V_{PP}$ , including overshoot should not exceed +13 V. (11-35)

APPENDIX A. INSTRUCTION INDEX (ALPHABETIC ORDER)

Instruction	Page	Instruction	Page
ADD A, mem	10-99	BE \$addr16	10-189
ADD A, saddr	10-98	BF A.bit, \$addr16	10-200
ADD A, sfr	10-98	BF PSWH.bit, \$addr16	10-201
ADD A, #byte	10-96	BF PSWL.bit, \$addr16	10-201
ADD mem, A	10-100	BF saddr.bit, \$addr16	10-199
ADD saddr, saddr	10-99	BF sfr.bit, \$addr16	10-199
ADD saddr, #byte	10-96	BF X.bit, \$addr16	10-200
ADD sfr, #byte	10-97	BFSET A.bit, \$addr16	10-207
ADD r, r1	10-97	BFSET PSWH.bit, \$addr16	10-208
ADDC A, mem	10-104	BFSET PSWL.bit, \$addr16	10-208
ADDC A, saddr	10-102	BFSET saddr.bit, \$addr16	10-206
ADDC A, sfr	10-103	BFSET sfr.bit, \$addr16	10-206
ADDC A, #byte	10-100	BFSET X.bit, \$addr16	10-207
ADDC mem, A	10-104	BGE \$addr16	10-193
ADDC saddr, saddr	10-103	BGT \$addr16	10-192
ADDC saddr, #byte	10-101	BH \$addr16	10-194
ADDC sfr, #byte	10-101	BL \$addr16	10-188
ADDC r, r1	10-102	BLE \$addr16	10-194
ADDW AX, saddrp	10-133	BLT \$addr16	10-193
ADDW AX, sfrp	10-133	BN \$addr16	10-191
ADDW AX, #word	10-131	BNC \$addr16	10-188
ADDW saddrp, saddrp	10-134	BNE \$addr16	10-190
ADDW saddrp, #word	10-131	BNH \$addr16	10-195
ADDW sfrp, #word	10-132	BNL \$addr16	10-188
ADDW rp, rp1	10-132	BNV \$addr16	10-191
ADJ4	10-154	BNZ \$addr16	10-190
AND A, mem	10-117	BP \$addr16	10-192
AND A, saddr	10-116	BPE \$addr16	10-190
AND A, sfr	10-116	BPO \$addr16	10-191
AND A, #byte	10-114	BR rp1	10-186
AND mem, A	10-118	BR !addr16	10-186
AND saddr, saddr	10-117	BR \$addr16	10-187
AND saddr, #byte	10-114	BR [rp1]	10-187
AND sfr, #byte	10-115	BRK	10-180
AND r, r1	10-115	BRKCS RBn	10-210
AND1 CY, A.bit	10-161	BT A.bit, \$addr16	10-196
AND1 CY, PSWH.bit	10-162	BT PSWH.bit, \$addr16	10-197
AND1 CY, PSWL.bit	10-163	BT PSWL.bit, \$addr16	10-198
AND1 CY, saddr.bit	10-159	BT saddr.bit, \$addr16	10-195
AND1 CY, sfr.bit	10-160	BT sfr.bit, \$addr16	10-196
AND1 CY, X.bit	10-162	BT X.bit, \$addr16	10-197
AND1 CY, /A.bit	10-161	BTCLR A.bit, \$addr16	10-203
AND1 CY, /PSWH.bit	10-163	BTCLR PSWH.bit \$addr16	10-204
AND1 CY, /PSWL.bit	10-163	BTCLR PSWL.bit \$addr16	10-205
AND1 CY, /saddr.bit	10-160	BTCLR saddr.bit \$addr16	10-202
AND1 CY, /sfr.bit	10-161	BTCLR sfr.bit, \$addr16	10-203
AND1 CY, /X.bit	10-162	BTCLR X.bit, \$addr16	10-204
BC \$addr16	10-188	BV \$addr16	10-190
		BZ \$addr16	10-189

(to be continued)

(cont'd)

Instruction	Page	Instruction	Page
CALL rpl	10-180	DECW saddrp	10-146
CALL !addr16	10-177	DECW SP	10-185
CALL [rpl]	10-180	DI	10-221
CALLF !addr11	10-178	DIVUW r1	10-142
CALLT [addr5]	10-179	DIVUX rpl	10-143
CLR1 A.bit	10-172	EI	10-221
CLR1 CY	10-175	INC r1	10-144
CLR1 PSWH.bit	10-173	INC saddr	10-144
CLR1 PSWL.bit	10-173	INCW rp2	10-145
CLR1 saddr.bit	10-172	INCW saddrp	10-145
CLR1 sfr.bit	10-172	INCW SP	10-185
CLR1 X.bit	10-173	MOV A, !addr16	10-86
CMP A, mem	10-130	MOV A, mem	10-84
CMP A, saddr	10-128	MOV A, PSWH	10-88
CMP A, sfr	10-129	MOV A, PSWL	10-88
CMP A, #byte	10-126	MOV A, r1	10-82
CMP mem, A	10-130	MOV A, saddr	10-82
CMP saddr, saddr	10-129	MOV A, sfr	10-83
CMP saddr, #byte	10-127	MOV A, [saddrp]	10-85
CMP sfr, #byte	10-127	MOV mem, A	10-85
CMP r, r1	10-128	MOV PSWH, A	10-87
CMPBKC [DE+], [HL+]	10-217	MOV PSWH, #byte	10-87
CMPBKC [DE-], [HL-]	10-217	MOV PSWL, A	10-87
CMPBKE [DE+], [HL+]	10-214	MOV PSWL, #byte	10-87
CMPBKE [DE-], [HL-]	10-214	MOV r1, #byte	10-81
CMPBKNC [DE+], [HL+]	10-219	MOV r, r1	10-82
CMPBKNC [DE-], [HL-]	10-219	MOV saddr, A	10-83
CMPBKNE [DE+], [HL+]	10-216	MOV saddr, saddr	10-83
CMPBKNE [DE-], [HL-]	10-216	MOV saddr, #byte	10-81
CMPMC [DE+], A	10-216	MOV sfr, A	10-84
CMPMC [DE-], A	10-216	MOV sfr, #byte	10-81
CMPME [DE+], A	10-214	MOV STBC, #byte	10-220
CMPME [DE-], A	10-214	MOV WDM, #byte	10-220
CMPMNC [DE+], A	10-218	MOV !addr16, A	10-86
CMPMNC [DE-], A	10-218	MOV [saddrp], A	10-86
CMPMNE [DE+], A	10-215	MOVBK [DE+], [HL+]	10-212
CMPMNE [DE-], A	10-215	MOVBK [DE-], [HL-]	10-212
CMPW AX, saddrp	10-140	MOVW [DE+], A	10-212
CMPW AX, sfrp	10-140	MOVW [DE-], A	10-212
CMPW AX, #word	10-138	MOVW AX, saddrp	10-92
CMPW saddrp, saddrp	10-141	MOVW AX, sfrp	10-93
CMPW saddrp, #word	10-138	MOVW AX, SP	10-185
CMPW sfrp, #word	10-139	MOVW rp, rp1	10-92
CMPW rp, rp1	10-139	MOVW rp1, #word	10-91
DBNZ r2, \$addr16	10-209	MOVW rp1, !addr16	10-94
DBNZ saddr, \$addr16	10-209	MOVW saddrp, AX	10-92
DEC r1	10-144	MOVW saddrp, saddrp	10-93
DEC saddr	10-145		
DECW rp2	10-146		

(to be continued)



(cont'd)

Instruction	Page	Instruction	Page
MOVW saddrp, #word	10-91	OR1 CY,/X.bit	10-166
MOVW sfrp, AX	10-93		
MOVW sfrp, #word	10-91	POP post	10-183
MOVW SP, AX	10-185	POP PSW	10-184
MOVW SP, #word	10-184	POPU post	10-184
MOVW !addr16, rpl	10-94	PUSH post	10-182
MOV1 A.bit, CY	10-158	PUSH PSW	10-182
MOV1 CY, A.bit	10-156	PUSHU post	10-183
MOV1 CY, PSWH.bit	10-157		
MOV1 CY, PSWL.bit	10-157	RET	10-181
MOV1 CY, saddr.bit	10-156	RETI	10-181
MOV1 CY, sfr.bit	10-156	RETCS !addr16	10-211
MOV1 CY, X.bit	10-157	ROL r1, n	10-148
MOV1 PSWH.bit, CY	10-159	ROLC r1, n	10-149
MOV1 PSWL.bit, CY	10-159	ROL4 [rpl]	10-153
MOV1 saddr.bit, CY	10-158	ROR r1, n	10-147
MOV1 sfr.bit, CY	10-158	RORC r1, n	10-148
MOV1 X.bit, CY	10-158	ROR4 [rpl]	10-153
MULU r1	10-142		
MULW rpl	10-142	SEL RBn	10-220
		SEL RBn, ALT	10-221
NOP	10-221	SET1 A.bit	10-171
NOT1 A.bit	10-174	SET1 CY	10-175
NOT1 CY	10-176	SET1 PSWH.bit	10-171
NOT1 PSWH.bit	10-174	SET1 PSWL.bit	10-172
NOT1 PSWL.bit	10-175	SET1 saddr.bit	10-170
NOT1 saddr.bit	10-173	SET1 sfr.bit	10-170
NOT1 sfr.bit	10-174	SET1 X.bit	10-171
NOT1 X.bit	10-174	SHL r1, n	10-150
		SHLW rpl, n	10-152
OR A, mem	10-121	SHR r1, n	10-149
OR A, saddr	10-120	SHRW rpl, n	10-151
OR A, sfr	10-120	SUB A, mem	10-108
OR A, #byte	10-118	SUB A, saddr	10-107
OR mem, A	10-122	SUB A, sfr	10-107
OR saddr, saddr	10-121	SUB A, #byte	10-105
OR saddr, #byte	10-119	SUB mem, A	10-109
OR sfr, #byte	10-119	SUB saddr, saddr	10-108
OR r, r1	10-120	SUB saddr, #byte	10-105
OR1 CY, A.bit	10-165	SUB sfr, #byte	10-106
OR1 CY, PSWH.bit	10-167	SUB r, r1	10-106
OR1 CY, PSWL.bit	10-167	SUBC A, mem	10-113
OR1 CY, saddr.bit	10-164	SUBC A, saddr	10-111
OR1 CY, sfr.bit	10-165	SUBC A, sfr	10-112
OR1 CY, X.bit	10-166	SUBC A, #byte	10-109
OR1 CY,/A.bit	10-166	SUBC mem, A	10-113
OR1 CY,/PSWH.bit	10-167	SUBC saddr, saddr	10-112
OR1 CY,/PSWL.bit	10-168	SUBC saddr, #byte	10-110
OR1 CY,/saddr.bit	10-164	SUBC sfr, #byte	10-110
OR1 CY,/sfr.bit	10-165	SUBC r, r1	10-111

(to be continued)

(cont'd)

Instruction		Page	Instruction		Page
SUBW	AX, saddrp	10-136	XCHW	AX, saddrp	10-94
SUBW	AX, sfrp	10-137	XCHW	AX, sfrp	10-95
SUBW	AX, #word	10-134	XCHW	rp, rpl	10-95
SUBW	saddrp, saddrp	10-137	XCHW	saddrp, saddrp	10-95
SUBW	saddrp, #word	10-135	XOR	A, mem	10-125
SUBW	sfrp, #word	10-135	XOR	A, saddr	10-124
SUBW	rp, rpl	10-136	XOR	A, sfr	10-124
SWRS		10-220	XOR	A, #byte	10-122
			XOR	mem, A	10-126
XCH	A, mem	10-89	XOR	saddr, saddr	10-125
XCH	A, r1	10-88	XOR	saddr, #byte	10-123
XCH	A, saddr	10-89	XOR	sfr, #byte	10-123
XCH	A, sfr	10-90	XOR	r, r1	10-124
XCH	A, [saddrp]	10-90	XOR1	CY, A.bit	10-169
XCH	r, r1	10-88	XOR1	CY, PSWH.bit	10-170
XCH	saddr, saddr	10-90	XOR1	CY, PSWL.bit	10-170
XCHBK	[DE+], [HL+]	10-213	XOR1	CY, saddr.bit	10-168
XCHBK	[DE-], [HL-]	10-213	XOR1	CY, sfr.bit	10-169
XCHM	[DE+], A	10-213	XOR1	CY, X.bit	10-169
XCHM	[DE-], A	10-213			

## APPENDIX B. DEVELOPMENT TOOLS

The following development tools are available for system development using the uPD78310A, 78312A and 78P312A.

### Hardware

IE-78310A-R	The IE-78310A-R is an in-circuit emulator which can be used for application system development and debugging. Debugging is performed with the IE-78310A-R by connecting to a host machine. Object file transfer to/from the host machine is possible, enabling debugging to be performed efficiently.
EP-78310CW EP-78310GF EP-78310GQ EP-78310L	Emulation probe for connecting the IE-78310A-R to the user's system.
PG-1500	The PG-1500 is a PROM programmer which allows programming, in standalone or via operation from a host computer, of a single-chip microcomputer with on-chip PROM by connection of the board provided and a separately available PROM programmer adapter. It can also program typical 256K-bit to 4 M-bit PROMs.
PA-78P312CW PA-78P312GF PA-78P312GQ PA-78P312L	PROM programmer adapter for writing a program to the uPD78P312A with a general-purpose PROM program such as the PG-1500. PA-78P312CW ... For uPD78P312ACW & 78P312ADW PA-78P312GF ... For uPD78P312AGF-3BE PA-78P312GQ ... For uPD78P312AGQ-36 & 78P312AR PA-78P312L .... For uPD78P312AL

### Other PROM Programmer

The following PROM programmer can also be used to uPD78P213A programming.

Manufacturer	Product Name
Data IO Japan	UNISITE 2900

Software

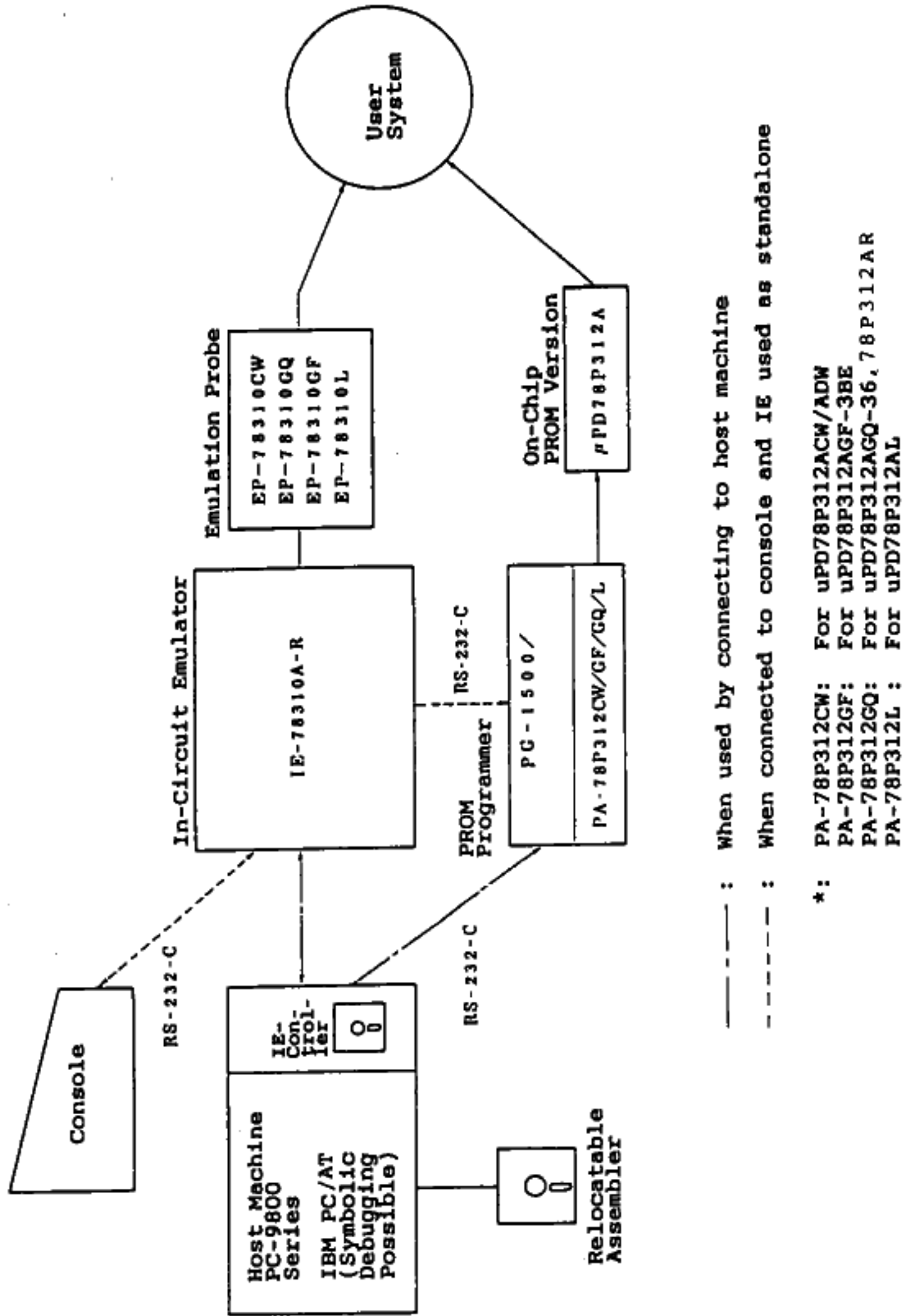
IE-78310A-R control program (IE controller)	Host Machine	OS	Supply Medium	Ordering Code (Product Name)
	PC-9800 series	MS-DOS™	8-inch 2D*	uS5A1IE78310-P01
			3.5-inch 2HD	uS5A13IE78310
			5-inch 2HD	uS5A10IE78310-P01
IBM PC/AT™	PC DOS™	5-inch 2HC	uS7B10IE78310	
78K/III Series relocatable assembler	Host Machine	OS	Supply Medium	Ordering Code (Product Name)
	PC-9800 series	MS-DOS	8-inch 2D*	uS5A1RA78K3
			3.5-inch 2HD	uS5A13RA78K3
			5-inch 2HD	uS5A10RA78K3
IBM PC/AT	PC DOS	5-inch 2HC	uS7B10RA78K3	
PC-1500 controller	Host Machine	OS	Supply Medium	Ordering Code (Product Name)
	PC-9800 series	MS-DOS	3.5-inch 2HD	uS5A13PG1500
			5-inch 2HD	uS5A10PG1500
IBM PC/AT	PC DOS	5-inch 2HC	uS7B10PG1500	

\*: No new shipments are currently being made on 8-inch 2D medium. Either 5-inch 2HD or 3.5-inch 2HD should be selected.

If you have already purchased 8-inch 2D software, future version upgrades will be issued on 5-inch 2HD floppies.

Remarks: Software operation is guaranteed only with the host machines and operating systems listed above.

# Development Tools Configuration



----- : When used by connecting to host machine

- - - - - : When connected to console and IE used as standalone

- \*: PA-78P312CW: For uPD78P312ACW/ADM  
 PA-78P312GF: For uPD78P312AGF-3BE  
 PA-78P312GQ: For uPD78P312AGQ-36, 78P312AR  
 PA-78P312L : For uPD78P312AL



Reference Materials on Development Tools

Reference Material		Document Number
IE-78310A-R User's Manual	Hardware	EEU-645
	Software	EEU-637
IE-78310-R System Software Instruction Manual	PC-9800 series (CP/M-86, MS-DOS based)	EEM-646
	IBM PC series (PC DOS based)	EEM-756
<input type="checkbox"/> RA78K Series Assembler Package User's Manual	Operation	EEU-809
<input type="checkbox"/>	Language	EEU-815