

μ PD78214 SUB-SERIES

8-BIT SINGLE-CHIP MICROCOMPUTER

HARDWARE

μ PD78212
 μ PD78213
 μ PD78214
 μ PD78P214
 μ PD78212 (A)
 μ PD78213 (A)
 μ PD78214 (A)
 μ PD78P214 (A)

GENERAL	1
PIN FUNCTIONS	2
CPU FUNCTION	3
CLOCK GENERATOR	4
PORT FUNCTIONS	5
REAL-TIME OUTPUT FUNCTION	6
TIMER/COUNTER UNITS	7
A/D CONVERTER	8
ASYNCHRONOUS SERIAL INTERFACE	9
CLOCK SYNCHRONOUS SERIAL INTERFACE	10
EDGE DETECTION FUNCTION	11
INTERRUPT FUNCTIONS	12
LOCAL BUS INTERFACE FUNCTION	13
STANDBY FUNCTION	14
RESET FUNCTION	15
APPLICATION EXAMPLES	16
PROGRAMMING FOR THE μ PD78214	17
INSTRUCTION OPERATIONS	18
78K/II SERIES PRODUCT LIST	A
DEVELOPMENT TOOLS	B
SOFTWARE FOR EMBEDDED APPLICATIONS	C
REGISTER INDEX	D
INDEX	E

Cautions on CMOS Devices

① Countermeasures against static electricity for all MOSs

Caution When handling MOS devices, take care so that they are not electrostatically charged.

Strong static electricity may cause dielectric breakdown in gates. When transporting or storing MOS devices, use conductive trays, magazine cases, shock absorbers, or metal cases that NEC uses for packaging and shipping. Be sure to ground MOS devices during assembling. Do not allow MOS devices to stand on plastic plates or do not touch pins. Also handle boards on which MOS devices are mounted in the same way.

② CMOS-specific handling of unused input pins

Caution Hold CMOS devices at a fixed input level.

Unlike bipolar or NMOS devices, if a CMOS device is operated with no input, an intermediate-level input may be caused by noise. This allows current to flow in the CMOS device, resulting in a malfunction. Use a pull-up or pull-down resistor to hold a fixed input level. Since unused pins may function as output pins at unexpected times, each unused pin should be separately connected to the V_{DD} or GND pin through a resistor. If handling of unused pins is documented, follow the instructions in the document.

③ Statuses of all MOS devices at initialization

Caution The initial status of a MOS device is unpredictable when power is turned on.

Since characteristics of a MOS device are determined by the amount of ions implanted in molecules, the initial status cannot be determined in the manufacture process. NEC has no responsibility for the output statuses of pins, input and output settings, and the contents of registers at power on. However, NEC assures operation after reset and items for mode setting if they are defined.

When you turn on a device having a reset function, be sure to reset the device first.

EWS-4800 Series, EWS-UX/V, and QTOP are trademarks of NEC Corporation.

MS-DOS is a trademark of Microsoft Corporation.

IBM DOS, PC/AT, and PC DOS are trademarks of IBM Corporation.

SPARCstation is a trademark of SPARC International, Inc.

Sun OS is a trademark of Sun Microsystems Inc.

HP9000 Series 300 and HP-UX are trademarks of Hewlett-Packard.

The information in this document is subject to change without notice.

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Corporation. NEC Corporation assumes no responsibility for any errors which may appear in this document.

NEC Corporation does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from use of a device described herein or any other liability arising from use of such device. No license, either express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Corporation or others.

The devices listed in this document are not suitable for use in aerospace equipment, submarine cables, nuclear reactor control systems and life support systems. If customers intend to use NEC devices for above applications or they intend to use "Standard" quality grade NEC devices for applications not intended by NEC, please contact our sales people in advance.

Application examples recommended by NEC Corporation

Standard: Computer, Office equipment, Communication equipment, Test and Measurement equipment, Machine tools, Industrial robots, Audio and Visual equipment, Other consumer products, etc.

Special: Automotive and Transportation equipment, Traffic control systems, Antidisaster systems, Anticrime systems, etc.

Main Revisions in This Edition

Page	Description
P.55	V_{SS} and "Caution" have been added in (a) of Fig. 4-2 .
P.329	"Caution" has been added in (2) of Section 12.4.6 .
P.383	"Caution" has been added in (b) of Section 14.4.2 .
P.429	Appendix B has been modified as follows: <ul style="list-style-type: none">• "IBM PC series" has been changed to "IBM PC/AT."• Upgraded versions of MS-DOS are now supported by some development tools designed for the PC-9800 series.• 3.5" 2HC has been added to as a PC DOS distribution media.• MS DOS and IBM DOS have been added as supported operating systems for the IBM PC/AT.• The description of real-time OS has been deleted.
P.441	Appendix C has been added.

Major changes in this revision are indicated by stars (★) in the margins.

PREFACE

Users:

This manual is aimed at engineers who need to be familiar with the capabilities of the μ PD78214 sub-series for application program development purposes.

Purpose:

The purpose of this manual is to help users understand the hardware capabilities of the μ PD78214 sub-series.

Organization:

Two manuals are available for the μ PD78214 sub-series: The hardware manual (this manual) and instruction manual (common to all 78K/II series products). The contents of the manuals are:

Hardware	Instruction
Pin functions	CPU functions
Internal block functions	Addressing
Interrupt functions	Instruction set
Other built-in functions	

Important information related to using the products described in this manual is provided in the form of “Caution” notes, appearing in appropriate places in each chapter. Each “Caution” is repeated at the end of the chapter. Be careful to observe these notes when using the products.

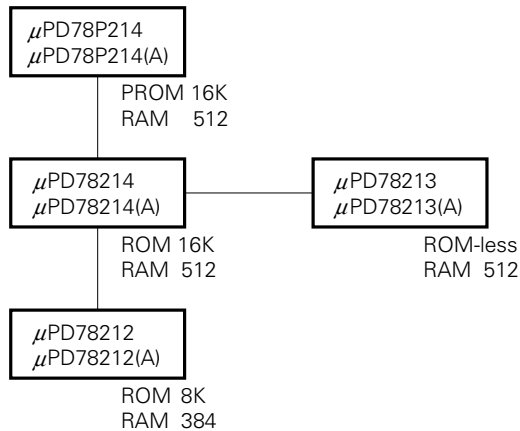
Guidance:

Readers of this manual are assumed to have a general knowledge of electronics, logical circuits, and microcomputers.

When using this manual with the μ PD78212, μ PD78213, μ PD78P214, μ PD78212(A), μ PD78213(A), μ PD78214(A), or μ PD78P214(A):

This manual describes the functions of the μ PD78212, μ PD78213, μ PD78214, μ PD78P214, μ PD78212(A), μ PD78213(A), μ PD78214(A), and μ PD78P214(A). The relationships between these products are shown in the figure on the next page. Where there is no functional difference between the products, only the μ PD78214 is described, that description also being applicable to the μ PD78212, μ PD78213, μ PD78P214, μ PD78212(A), μ PD78213(A), μ PD78214(A), and μ PD78P214(A).

The examples given in this manual are prepared for “Standard” quality products for general electronics devices. If customers intend to use the examples in this manual in fields where the “Special” quality is required, note the quality grade of the parts and circuits actually used.



To check the details of a register when you know the name of the register:

See **Appendix D**.

To check the differences between the μ PD78214 sub-series and other models of the 78K/II series:

First see **Appendix A** to determine the differences between the models then see **Appendix E** for details.

To check the details of a function when you know the name of the function:

See **Appendix E**.

If the microcomputer does not operate correctly during debugging:

See the cautions at the end of the chapter related to the erroneous function.

To become familiar with the general functions of the μ PD78214 sub-series:

Read the entire manual in the order of the table of contents.

To determine the instructions supported by the μ PD78214 sub-series in detail:

Refer to **78K/II Series User's Manual, Instruction (IEU-1311)**.

To determine the electrical characteristics of the μ PD78214 sub-series:

Refer to the separate data sheet.

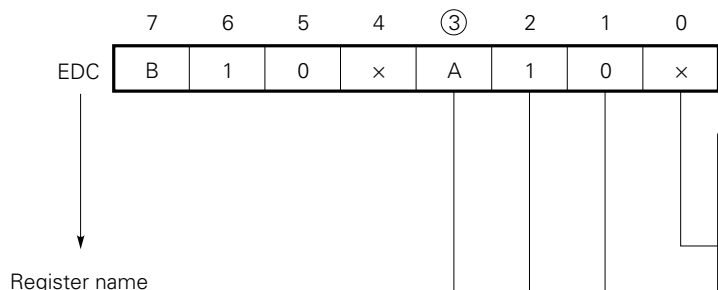
Application examples of the μ PD78214 sub-series:

Refer to the separate application note.

Notation:

- Data weight: High-order digits on the left side
Low-order digits on the right side
- Active low: $\overline{\text{xxx}}$ (Pins and signal names are overscored.)
- Note: Explanation of a noted part of text
- Caution: Information demanding the user's special attention
- Remarks: Supplementary information
- Numeric value: Binary : xxxxB or xxxx
Decimal : xxxx
Hexadecimal : xxxxH

Register representation



The encircled bit number indicates that the bit name is used as reserved word by the NEC assembler and defined by the header file, sfrbit.h, by C compiler.

Write operation	Read operation
Either 0 or 1 can be written to this bit without affecting the register operation.	0 or 1 is read from these bits.
Write 0 to this bit.	
Write 1 to this bit.	
Write a value according to the necessary function.	A value is read according to the operation.

Never use the code combinations indicated "Not to be set" in the register descriptions.

Characters likely to be confused: 0 (zero) and O (uppercase "O")

1 (one), l (lowercase "L"), and I (uppercase "I")

Related documents:

The following reference documents are also available.

- **Documents related to the μ PD78214 sub-series**

Document		Product	μ PD78212	μ PD78212(A)	μ PD78P214(A)
			μ PD78213	μ PD78213(A)	
Data Sheet			IC-2526	IC-2831	IC-3095
User's Manual	Hardware	This manual			
	Instruction	IEU-1311			
Application Note	Basic	IEA-1220			
	Application	IEA-1282			
	Floating-Point Arithmetic Operation Programs	IEA-1273			

- **Serial Bus Interface (SBI) User's Manual (IEM-1303)**

- **Documents related to development tools**

Document name		Document No.
IE-78240-R-A In-Circuit Emulator User's Manual		EEU-1395
IE-78240-R In-Circuit Emulator User's Manual	Hardware	EEU-1322
	Software	EEU-1331
IE-78210-R In-Circuit Emulator Hardware Operator's Manual		EEP-1027
IE-78210-R In-Circuit Emulator Software Operator's Manual		EEM-1024
IE-78210-R In-Circuit Emulator System Software Operator's Manual	For Operation on the PC-9800 Series (MS-DOS)	EEM-1260
	For Operation on the IBM PC Series (PC DOS)	EEM-1027
RA78K Series Assembler Package User's Manual	Language	EEU-1283
	Operation	EEU-1273
78K Series Structured Assembler Preprocessor User's Manual		EEU-1254
CC78K Series C Compiler User's Manual	Language	EEU-1289
	Operation	EEU-1280
SD78K/II Screen Debugger User's Manual for Operation under MS-DOS		EEU-1447
	Tutorial	EEU-1413
78K/II Series Development Tools Selection Guide		EF-1114

- **Documents related to software to be incorporated into the product**

Document name		Document No.
Fuzzy Inference Development Support System Pamphlet		EF-1113
78K/II Series Fuzzy Inference Development Support System User's Manual	Fuzzy Inference Module	EEU-1448
78K/0, 78K/II and 87AD Series Fuzzy Inference Development Support System User's Manual	Translator	EEU-1444
User's Manual for Tool for Creating Fuzzy Knowledge Data		EEU-1438

- **Other documents**

Document name	Document No.
Package Manual	IEI-1213
SMD Surface Mount Technology Manual	IEI-1207
Quality Grades on NEC Semiconductor Devices	IEI-1209
NEC Semiconductor Device Reliability/Quality Control System	IEI-1203
Guide to Quality Assurance for Semiconductor Devices	MEI-1202

Caution The above documents may be revised without notice. Use the latest versions when you designing an application system.

CONTENTS

CHAPTER 1	GENERAL	1
1.1	FEATURES	3
1.2	ORDERING INFORMATION AND QUALITY GRADE	4
1.2.1	Ordering Information	4
1.2.2	Quality Grade	5
1.3	PIN CONFIGURATION (TOP VIEW)	6
1.3.1	Normal Operating Mode	6
1.3.2	PROM Programming Mode	11
1.4	EXAMPLE APPLICATION SYSTEM (PRINTER)	16
1.5	BLOCK DIAGRAM	17
1.6	FUNCTIONS	18
1.7	DIFFERENCES BETWEEN THE μ PD78210 AND μ PD78213	20
1.8	DIFFERENCES BETWEEN THE μ PD78214 SUB-SERIES AND μ PD78218A SUB-SERIES	21
1.9	DIFFERENCES BETWEEN THE μ PD78212 AND μ PD78212(A)	22
1.10	DIFFERENCES BETWEEN THE μ PD78213 AND μ PD78214, AND THE μ PD78213(A) AND μ PD78214(A)	22
1.11	DIFFERENCES BETWEEN THE μ PD78P214 AND μ PD78P214(A)	22
1.12	DIFFERENCES BETWEEN THE μ PD78212, μ PD78213, μ PD78214, AND μ PD78P214 ...	23
1.12.1	Functional Differences	23
1.12.2	Package Differences	23
CHAPTER 2	PIN FUNCTIONS	25
2.1	PIN FUNCTION LIST	25
2.1.1	Normal Operating Mode	25
2.1.2	PROM Programming Mode	27
2.2	PIN FUNCTIONS	27
2.2.1	Normal Operating Mode	27
2.2.2	PROM Programming Mode	31
2.3	I/O CIRCUITS AND UNUSED-PIN HANDLING	33
2.4	NOTES	35
CHAPTER 3	CPU FUNCTION	37
3.1	MEMORY SPACE	37
3.1.1	Internal Program Memory Area	42
3.1.2	Internal RAM Area	43
3.1.3	Special Function Register (SFR) Area	43
3.1.4	External SFR Area	43
3.1.5	External Memory Space	43
3.1.6	External Extension Data Memory Space	43
3.2	REGISTERS	45
3.2.1	Program Counter (PC)	45
3.2.2	Program Status Word (PSW)	45
3.2.3	Stack Pointer (SP)	46
3.2.4	General-Purpose Registers	47
3.2.5	Special Function Registers (SFR)	50

3.3	NOTES	53
CHAPTER 4	CLOCK GENERATOR	55
4.1	CONFIGURATION AND FUNCTION	55
4.2	NOTES	56
4.2.1	Inputting an external clock	56
4.2.2	Using the Crystal/Ceramic Oscillator	56
CHAPTER 5	PORT FUNCTIONS	59
5.1	DIGITAL I/O PORTS	59
5.2	PORT 0.....	60
5.2.1	Hardware Configuration	61
5.2.2	Setting the Input/Output Mode and Control Mode	61
5.2.3	Operation	62
5.2.4	Built-In Pull-Up Resistor	62
5.2.5	Driving Transistors	62
5.3	PORT 2.....	63
5.3.1	Hardware Configuration	64
5.3.2	Setting the Input Mode and Control Mode	64
5.3.3	Operation	64
5.3.4	Built-In Pull-Up Resistor	65
5.4	PORT 3.....	66
5.4.1	Hardware Configuration	68
5.4.2	Setting the I/O Mode and Control Mode	71
5.4.3	Operation	73
5.4.4	Built-In Pull-Up Resistor	74
5.5	PORT 4.....	75
5.5.1	Hardware Configuration	76
5.5.2	Setting the I/O Mode and Control Mode	76
5.5.3	Operation	77
5.5.4	Built-In Pull-Up Resistor	78
5.5.5	Driving LEDs Directly	79
5.6	PORT 5.....	80
5.6.1	Hardware Configuration	80
5.6.2	Setting the I/O Mode and Control Mode	80
5.6.3	Operation	81
5.6.4	Built-In Pull-Up Resistor	82
5.6.5	Driving LEDs Directly	83
5.7	PORT 6.....	84
5.7.1	Hardware Configuration	85
5.7.2	Setting the I/O Mode and Control Mode	88
5.7.3	Operation	90
5.7.4	Built-In Pull-Up Resistor	91
5.7.5	Note	92
5.8	PORT 7.....	92
5.8.1	Hardware Configuration	92
5.8.2	Setting the I/O Mode and Control Mode	92
5.8.3	Operation	93

	5.8.4	Built-In Pull-Up Resistor	93
	5.8.5	Notes	93
	5.9	NOTES	93
CHAPTER 6		REAL-TIME OUTPUT FUNCTION	95
	6.1	CONFIGURATION AND FUNCTION	95
	6.2	REAL-TIME OUTPUT CONTROL REGISTER (RTPC)	97
	6.3	ACCESS TO THE REAL-TIME OUTPUT PORT	97
	6.4	OPERATION	99
	6.5	APPLICATION EXAMPLE	102
	6.6	NOTES	104
CHAPTER 7		TIMER/COUNTER UNITS	107
	7.1	16-BIT TIMER/COUNTER	109
	7.1.1	Functions	109
	7.1.2	Configuration	109
	7.1.3	16-Bit Timer/Counter Control Registers	111
	7.1.4	Operation of 16-Bit Timer 0 (TM0)	114
	7.1.5	Compare Register and Capture Register Operations	117
	7.1.6	Basic Operation of Output Control Circuit	119
	7.1.7	PWM Output	122
	7.1.8	PPG Output	125
	7.1.9	Sample Applications	129
	7.2	8-BIT TIMER/COUNTER 1	139
	7.2.1	Functions	139
	7.2.2	Configuration	140
	7.2.3	8-Bit Timer/Counter 1 Control Registers	143
	7.2.4	Operation of 8-Bit Timer 1 (TM1)	145
	7.2.5	Compare Register and Capture/Compare Register Operations	148
	7.2.6	Sample Applications	151
	7.3	8-BIT TIMER/COUNTER 2	159
	7.3.1	Functions	159
	7.3.2	Configuration	161
	7.3.3	8-Bit Timer/Counter 2 Control Registers	163
	7.3.4	Operation of 8-Bit Timer 2 (TM2)	167
	7.3.5	External Event Counter Function	170
	7.3.6	One-Shot Timer Function	175
	7.3.7	Compare Register and Capture Register Operations	176
	7.3.8	Basic Operation of Output Control Circuit	179
	7.3.9	PWM Output	182
	7.3.10	PPG Output	185
	7.3.11	Sample Applications	190
	7.4	8-BIT TIMER/COUNTER 3	205
	7.4.1	Functions	205
	7.4.2	Configuration	205
	7.4.3	8-Bit Timer/Counter 3 Control Registers	207
	7.4.4	Operation of 8-Bit Timer 3 (TM3)	208
	7.4.5	Compare Register Operation	210

	7.4.6 Sample Applications	211
	7.5 NOTES	212
	7.5.1 Common Notes on All Timers/Counters	212
★	7.5.2 Notes on 16-Bit Timer/Counter	219
	7.5.3 Notes on 8-Bit Timer/Counter 2	219
	7.5.4 Notes on Using In-Circuit Emulators	222
CHAPTER 8	A/D CONVERTER	225
	8.1 CONFIGURATION	225
	8.2 A/D CONVERTER MODE REGISTER (ADM)	228
	8.3 OPERATION	230
	8.3.1 Basic A/D Converter Operation	230
	8.3.2 Select Mode	232
	8.3.3 Scan Mode	233
	8.3.4 A/D Conversion Activated by Software Start	234
	8.3.5 A/D Conversion Activated by Hardware Start	235
	8.4 INTERRUPT REQUEST FROM THE A/D CONVERTER	239
	8.5 SETTING FOR USE OF AN6 AND AN7	239
	8.6 NOTES	239
CHAPTER 9	ASYNCHRONOUS SERIAL INTERFACE	243
	9.1 CONFIGURATION	243
	9.2 ASYNCHRONOUS SERIAL INTERFACE CONTROL REGISTER	245
	9.3 ASYNCHRONOUS SERIAL INTERFACE OPERATIONS	247
	9.3.1 Data Format	247
	9.3.2 Parity Types and Operations	247
	9.3.3 Transmission	248
	9.3.4 Reception	249
	9.3.5 Reception Error	249
	9.4 BAUD RATE GENERATOR	251
	9.4.1 Configuration of the Baud Rate Generator for UART	251
	9.4.2 Baud Rate Generator Control Register (BRGC)	251
	9.4.3 Operation of the Baud Rate Generator for UART	253
	9.5 BAUD RATE SETTING	254
	9.5.1 Example of Setting the BRGC Register When the Baud Rate Generator for UART Is Used	254
	9.5.2 Example of Setting the Baud Rate When 8-bit Timer/Counter 3 Is Used	256
	9.5.3 Example of Setting the BRGC When the External Baud Rate Input (ASCK) Is Used	258
	9.6 NOTES	258
CHAPTER 10	CLOCK SYNCHRONOUS SERIAL INTERFACE	259
	10.1 FUNCTION	259
	10.2 CONFIGURATION	259
	10.3 CONTROL REGISTERS	262
	10.3.1 Clock Synchronous Serial Interface Mode Register (CSIM)	262
	10.3.2 Serial Bus Interface Control Register (SBIC)	263

10.4	OPERATIONS IN THE THREE-WIRE SERIAL I/O MODE	265
10.4.1	Basic Operation Timing	265
10.4.2	Operation When Only Transmission Is Permitted.....	267
10.4.3	Operation When Only Reception Is Permitted.....	267
10.4.4	Operation When Both Transmission and Reception Are Permitted	267
10.4.5	Action to Be Taken When the Serial Clock and Shift Become Asynchronous	268
10.5	SBI MODE	268
10.5.1	Features of SBI	268
10.5.2	Configuration of the Serial Interface	270
10.5.3	Detecting an Address Match	272
10.5.4	Control Registers in SBI Mode.....	272
10.6	SBI COMMUNICATION AND SIGNALS.....	277
10.6.1	Bus Release Signal (REL)	277
10.6.2	Command Signal (CMD)	278
10.6.3	Address	278
10.6.4	Command and Data	279
10.6.5	Acknowledge Signal (\overline{ACK}).....	279
10.6.6	Busy Signal (\overline{BUSY}) and Ready Signal (READY)	280
10.6.7	Signals.....	280
10.6.8	Communication	287
10.6.9	Releasing the Busy State	287
10.6.10	Setting Wake-Up	287
10.6.11	Starting Transmission and Reception.....	287
10.7	NOTES	292
CHAPTER 11	EDGE DETECTION FUNCTION	293
11.1	EXTERNAL INTERRUPT MODE REGISTERS (INTM0, INTM1)	293
11.2	EDGE DETECTION ON PIN P20	296
11.3	EDGE DETECTION ON PINS P21 TO P26	297
11.4	NOTES	298
CHAPTER 12	INTERRUPT FUNCTIONS	301
12.1	INTERRUPT REQUEST SOURCES.....	302
12.1.1	Software Interrupt Request	302
12.1.2	Nonmaskable Interrupt Request	303
12.1.3	Maskable Interrupt Request	303
12.1.4	Selecting an Interrupt Source	303
12.2	INTERRUPT HANDLING CONTROL REGISTERS.....	304
12.2.1	Interrupt Request Flag Register (IF0)	305
12.2.2	Interrupt Mask Register (MK0)	306
12.2.3	Interrupt Service Mode Register (ISM0).....	306
12.2.4	Priority Specification Flag Register (PR0).....	306
12.2.5	Interrupt Status Register (IST)	307
12.2.6	Program Status Word (PSW)	308
12.3	INTERRUPT HANDLING	308
12.3.1	Accepting Software Interrupts	308
12.3.2	Accepting Nonmaskable Interrupts	308
12.3.3	Accepting Maskable Interrupts	311

12.3.4	Multiplexed-Interrupt Handling.....	313
12.3.5	Interrupt Request and Macro Service Pending.....	316
12.3.6	Interrupt and Macro Service Operation Timing	317
12.4	MACRO SERVICE FUNCTION	319
12.4.1	Macro Service Outline.....	319
12.4.2	Macro Service Types	320
12.4.3	Macro Service Basic Operation.....	321
12.4.4	Macro Service Control Register	322
12.4.5	Macro Service Type A	323
12.4.6	Type B Macro Service	327
12.4.7	Macro Service Type C	331
12.5	NOTES	343
CHAPTER 13	LOCAL BUS INTERFACE FUNCTION	345
13.1	CONTROL REGISTERS	346
13.1.1	Memory Expansion Mode Register (MM)	346
13.1.2	Programmable Wait Control Register (PW)	347
13.2	MEMORY EXPANSION FUNCTION	347
13.2.1	External Memory Expansion Function	347
13.2.2	1M-Byte Expansion Function	348
13.2.3	Memory Mapping with Expanded Memory	350
13.2.4	Example of Connecting Memories	355
13.3	INTERNAL ROM HIGH-SPEED FETCH FUNCTION	357
13.4	WAIT FUNCTION.....	357
13.5	PSEUDO STATIC RAM REFRESH FUNCTION	367
13.5.1	Function	367
13.5.2	Refresh Mode Register (RFM)	367
13.5.3	Operation	368
13.5.4	Example of Connecting Pseudo Static RAM	372
13.6	NOTES	372
CHAPTER 14	STANDBY FUNCTION	377
14.1	FUNCTION OVERVIEW	377
14.2	STANDBY CONTROL REGISTER (STBC)	379
14.3	HALT MODE	379
14.3.1	Specifying HALT Mode and Operation States in HALT Mode.....	379
14.3.2	Releasing HALT Mode.....	380
14.4	STOP MODE	382
14.4.1	Specifying STOP Mode and Operation States in STOP Mode	382
14.4.2	Releasing STOP Mode.....	382
14.4.3	Notes on Using STOP Mode	384
14.5	NOTES	386
CHAPTER 15	RESET FUNCTION	389
15.1	RESET FUNCTION.....	389
15.2	NOTE	393

CHAPTER 16	APPLICATION EXAMPLES	395	
	16.1 OPEN-LOOP CONTROL OF STEPPER MOTORS	395	
	16.2 SERIAL COMMUNICATION WITH MULTIPLE DEVICES	397	
CHAPTER 17	PROGRAMMING FOR THE μPD78P214	399	
	17.1 OPERATING MODE	399	
	17.2 PROCEDURE FOR WRITING INTO PROM	399	
	17.3 PROCEDURE FOR READING FROM PROM	401	
	17.4 NOTE	402	
CHAPTER 18	INSTRUCTION OPERATIONS	403	
	18.1 LEGEND	403	
	18.1.1 Operand Field	403	
	18.1.2 Operation Field	404	
	18.1.3 Flag Field	405	
	18.2 LIST OF OPERATIONS	406	
	18.3 INSTRUCTION LISTS FOR EACH ADDRESSING TYPE	416	
APPENDIX A	78K/II SERIES PRODUCT LIST	421	
APPENDIX B	DEVELOPMENT TOOLS	429	
	B.1 HARDWARE	431	
	B.2 SOFTWARE	433	
	B.2.1 Language Processing Software	433	
	B.2.2 Software for the In-Circuit Emulator	435	
	B.2.3 Software for the PROM Programmer	437	
	B.2.4 OS for the IBM PC	437	★
	B.3 UPGRADING OTHER IN-CIRCUIT EMULATORS TO 78K/II SERIES LEVEL	438	
	B.3.1 Upgrading to IE-78240-R-A Level	438	
	B.3.2 Upgrading to IE-78240-R Level	439	
	B.3.3 Upgrading to IE-78210-R Level	440	
APPENDIX C	SOFTWARE FOR EMBEDDED APPLICATIONS	441	★
	C.1 FUZZY INFERENCE DEVELOPMENT SUPPORT SYSTEM	441	
APPENDIX D	REGISTER INDEX	443	
	D.1 REGISTER INDEX	443	
	D.2 REGISTER SYMBOL INDEX	445	
APPENDIX E	INDEX	447	
	E.1 INDEX	447	
	E.2 SYMBOL INDEX	452	

LIST OF FIGURES

Fig. No.	Title, Page
2-1	I/O Circuits Provided for Pins 34
3-1	Memory Map of μ PD78212 (\overline{EA} Pin Driven High) 38
3-2	Memory Map of μ PD78212 (\overline{EA} Pin Driven Low) 39
3-3	Memory Map of μ PD78213, μ PD78214, or μ PD78P214 (\overline{EA} Pin Driven Low) 40
3-4	Memory Map of μ PD78214, μ PD78P214 (\overline{EA} Pin Driven High) 41
3-5	Sample Data Transfer between Banks 44
3-6	Configuration of the Program Counter 45
3-7	Configuration of the Program Status Word 45
3-8	Configuration of the Stack Pointer 46
3-9	Data Saved to the Stack Area 47
3-10	Data Restored from the Stack Area 47
3-11	Configuration of General-Purpose Registers 48
4-1	Block Diagram of Clock Generator 55
4-2	External Circuit for the Clock Oscillator 55
4-3	Point from Which Signals Can Be Drawn When an External Clock Is Input 56
4-4	Notes on Connection of the Oscillator 57
4-5	Incorrect Oscillator Connections 57
5-1	Port Configuration 59
5-2	Configuration of Port 0 61
5-3	Port 0 Mode Register Format 61
5-4	Port Specified as an Output Port 62
5-5	Example of Driving a Transistor 62
5-6	Block Diagram of Port 2 64
5-7	Port Specified as an Input Port 65
5-8	Built-In Pull-Up Resistor Format 65
5-9	Connection of Pull-Up Resistors (Port 2) 66
5-10	Block Diagram of P30 (Port 3) 68
5-11	Block Diagram of P31, and P34 through P37 (Port 3) 69
5-12	Block Diagram of P32 (Port 3) 70
5-13	Block Diagram of P33 (Port 3) 71
5-14	Port 3 Mode Register Format 72
5-15	Port 3 Mode Control Register (PMC3) Format 72
5-16	Port Specified as an Output Port 73
5-17	Port Specified as an Input Port 73
5-18	Port Specified as a Control Signal Input or Output 74
5-19	Pull-Up-Resistor-Option Register Format 74
5-20	Connection of Pull-Up Resistors (Port 3) 75
5-21	Block Diagram of Port 4 76
5-22	Port Specified as an Output Port 77
5-23	Port Specified as an Input Port 77
5-24	Pull-Up-Resistor-Option Register Format 78

Fig. No.	Title, Page
5-25	Connection of Pull-Up Resistors (Port 4) 79
5-26	Example of Driving an LED Directly 79
5-27	Block Diagram of Port 5..... 80
5-28	Port 5 Mode Register Format..... 81
5-29	Port Specified as an Output Port..... 81
5-30	Port Specified as an Input Port..... 82
5-31	Pull-Up-Resistor-Option Register Format..... 82
5-32	Connection of Pull-Up Resistors (Port 5) 83
5-33	Example of Driving an LED Directly 83
5-34	Block Diagram of P60 through P63 (Port 6) 85
5-35	Block Diagram of P64 and P65 (Port 6) 86
5-36	Block Diagram of P66 (Port 6)..... 87
5-37	Block Diagram of P67 (Port 6)..... 88
5-38	Port 6 Mode Register Format..... 89
5-39	Port Specified as an Output Port..... 90
5-40	Port Specified as an Input Port..... 90
5-41	Pull-Up-Resistor-Option Register Format..... 91
5-42	Connection of Pull-Up Resistors (Port 6) 91
5-43	Block Diagram of Port 7..... 92
5-44	Port Specified as an Input Port..... 93
6-1	Block Diagram of the Real-Time Output Port 96
6-2	Real-Time Output Port Control Register (RTPC) Format..... 97
6-3	Configuration of the Buffer Registers (P0H and P0L) 97
6-4	Real-Time Output Port Operation Timing 100
6-5	Real-Time Output Port Operation Timing (Controlling 2 Channels Independently of Each Other) 101
6-6	Real-Time Output Port Operation Timing 102
6-7	Contents of the Control Register for the Real-Time Output Function..... 103
6-8	Real-Time Output Function Setting Procedure 103
6-9	Interrupt Request Handling When the Real-Time Output Function Is Used 104
7-1	Block Diagrams of Timer/Counter Units 108
7-2	Block Diagram of 16-Bit Timer/Counter 110
7-3	Format of Timer Control Register 0 (TMC0) 112
7-4	Format of Capture/Compare Control Register 0 (CRC0) 112
7-5	Format of Timer Output Control Register (TOC)..... 113
7-6	Basic Operation of 16-Bit Timer 0 (TM0)..... 114
7-7	TM0 Cleared by a Coincidence with Compare Register (CR01) 115
7-8	Clear Operation When the CE0 Bit Is Reset to 0 116
7-9	Compare Operation..... 117
7-10	TM0 Cleared After a Coincidence Is Detected 118
7-11	Capture Operation 119
7-12	Toggle Output Operation 121
7-13	PWM Pulse Output 122
7-14	Example of PWM Output Using TM0 123
7-15	PWM Output When CR00 = FFFFH 123

Fig. No.	Title, Page
7-16	Example of Rewriting Compare Register CR00 124
7-17	Example of PWM Output Signal with a 100% Duty Factor 124
7-18	Example of PPG Output Using TM0 125
7-19	PPG Output When CR00 = CR01 126
7-20	PPG Output When CR00 = 0000H 126
7-21	Example of Rewriting Compare Register CR00 127
7-22	Example of PPG Output Signal with a 100% Duty Factor 127
7-23	Example of PPG Output Period Made Longer 128
7-24	Timing of Interval Timer Operation (1) 129
7-25	Setting of Control Registers for Interval Timer Operation (1) 130
7-26	Setting Procedure for Interval Timer Operation (1) 130
7-27	Interrupt Request Handling for Interval Timer Operation (1) 131
7-28	Timing of Interval Timer Operation (2) 131
7-29	Setting of Control Registers for Interval Timer Operation (2) 132
7-30	Setting Procedure for Interval Timer Operation (2) 132
7-31	Timing of Pulse Width Measurement 133
7-32	Setting of Control Registers for Pulse Width Measurement 133
7-33	Setting Procedure for Pulse Width Measurement 134
7-34	Interrupt Request Handling for Pulse Width Calculation 134
7-35	Example of PWM Signal Output by 16-Bit Timer/Counter 135
7-36	Setting of Control Registers for PWM Output Operation 135
7-37	Setting Procedure for PWM Output 136
7-38	Changing Duty Factor of PWM Output 136
7-39	Example of PPG Signal Output by 16-Bit Timer/Counter 137
7-40	Setting of Control Registers for PPG Output Operation 137
7-41	Setting Procedure for PPG Output 138
7-42	Changing Duty Factor of PPG Output 138
7-43	Block Diagram of 8-Bit Timer/Counter 1 141
7-44	Format of Timer Control Register 1 (TMC1) 143
7-45	Format of Prescaler Mode Register 1 (PRM1) 144
7-46	Format of Capture/Compare Control Register 1 (CRC1) 144
7-47	Basic Operation of 8-Bit Timer 1 (TM1) 145
7-48	TM1 Cleared by a Coincidence with Compare Register (CR1m) 146
7-49	TM1 Cleared after Capture Operation 147
7-50	Clear Operation When the CE1 Bit Is Reset to 0 147
7-51	Compare Operation 149
7-52	TM1 Cleared After a Coincidence Is Detected 149
7-53	Capture Operation 150
7-54	TM1 Cleared after Capture Operations 151
7-55	Timing of Interval Timer Operation (1) 152
7-56	Setting of Control Registers for Interval Timer Operation (1) 152
7-57	Setting Procedure for Interval Timer Operation (1) 153
7-58	Interrupt Request Handling for Interval Timer Operation (1) 153
7-59	Timing of Interval Timer Operation (2) 154
7-60	Setting of Control Registers for Interval Timer Operation (2) 154
7-61	Setting Procedure for Interval Timer Operation (2) 155

Fig. No.	Title, Page
7-62	Timing of Pulse Width Measurement 156
7-63	Setting of Control Registers for Pulse Width Measurement 157
7-64	Setting Procedure for Pulse Width Measurement 158
7-65	Interrupt Request Handling for Pulse Width Calculation 158
7-66	Block Diagram of 8-Bit Timer/Counter 2 162
7-67	Format of Timer Control Register 1 (TMC1) 163
7-68	Format of Prescaler Mode Register 1 (PRM1) 164
7-69	Format of Capture/Compare Control Register 2 (CRC2) 165
7-70	Format of Timer Output Control Register (TOC) 166
7-71	Basic Operation of 8-Bit Timer 2 (TM2) 167
7-72	TM2 Cleared by a Coincidence with Compare Register (CR21) 168
7-73	TM2 Cleared After Capture Operation 168
7-74	Clear Operation When the CE2 Bit Is Reset to 0 169
7-75	External Event Count Timing of 8-Bit Timer/Counter 2 171
7-76	Interrupt Request Generation Using External Event Counter 172
7-77	Example Where Input of No Valid Edge Cannot Be Distinguished from Input of Only One Valid Edge with External Event Counter 173
7-78	How to Distinguish Input of No Valid Edge from Input of Only One Valid Edge with External Event Counter 173
7-79	One-Shot Timer Operation 175
7-80	Compare Operation 176
7-81	TM2 Cleared After a Coincidence Is Detected 177
7-82	Capture Operation 178
7-83	TM2 Cleared After Capture Operation 179
7-84	Toggle Output Operation 181
7-85	PWM Pulse Output 182
7-86	Example of PWM Output Using TM2 183
7-87	PWM Output When CR20 = FFH 184
7-88	Example of Rewriting a Compare Register 184
7-89	Example of PWM Output Signal with a 100% Duty Factor 185
7-90	Example of PPG Output Using TM2 186
7-91	PPG Output When CR20 = CR21 187
7-92	PPG Output When CR20 = 00H 187
7-93	Example of Rewriting Compare Register CR20 188
7-94	Example of PPG Output Signal with a 100% Duty Factor 188
7-95	Example of PPG Output Period Made Longer 189
7-96	Timing of Interval Timer Operation (1) 190
7-97	Setting of Control Registers for Interval Timer Operation (1) 191
7-98	Setting Procedure for Interval Timer Operation (1) 191
7-99	Interrupt Request Handling for Interval Timer Operation (1) 192
7-100	Timing of Interval Timer Operation (2) 192
7-101	Setting of Control Registers for Interval Timer Operation (2) 193
7-102	Setting Procedure for Interval Timer Operation (2) 194
7-103	Timing of Pulse Width Measurement 195
7-104	Setting of Control Registers for Pulse Width Measurement 195
7-105	Setting Procedure for Pulse Width Measurement 196

Fig. No.	Title, Page
7-106	Interrupt Request Handling for Pulse Width Calculation 197
7-107	Example of PWM Signal Output by 8-Bit Timer/Counter 2 197
7-108	Setting of Control Registers for PWM Output Operation 198
7-109	Setting Procedure for PWM Output 199
7-110	Changing Duty Factor of PWM Output 199
7-111	Example of PPG Signal Output by 8-Bit Timer/Counter 2 199
7-112	Setting of Control Registers for PPG Output Operation 200
7-113	Setting Procedure for PPG Output 201
7-114	Changing Duty Factor of PPG Output 201
7-115	External Event Counter Operation 201
7-116	Setting of Control Registers for External Event Counter Operation 202
7-117	Setting Procedure for External Event Counter Operation 202
7-118	One-Shot Timer Operation 203
7-119	Setting of Control Registers for One-Shot Timer Operation 203
7-120	Setting Procedure for One-Shot Timer Operation 204
7-121	Procedure for Starting an Additional One-Shot Timer Operation 204
7-122	Block Diagram of 8-Bit Timer/Counter 3 206
7-123	Format of Timer Control Register 0 (TMC0) 207
7-124	Format of Prescaler Mode Register 0 (PRM0) 207
7-125	Basic Operation of 8-Bit Timer 3 (TM3) 208
7-126	TM3 Cleared by a Coincidence with Compare Register (CR30) 209
7-127	Clear Operation When the CE3 Bit Is Reset to 0 209
7-128	Compare Operation 211
7-129	Timing of Interval Timer Operation 211
7-130	Setting of Control Registers for Interval Timer Operation 212
7-131	Setting Procedure for Interval Timer Operation 212
7-132	Count Start Operation 214
7-133	Count Operation Stop 214
7-134	Timing of Count Operation Stop and Restart 214
7-135	Example of PWM Output Signal with a 100% Duty Factor 216
7-136	Example of PPG Output Signal with a 100% Duty Factor 217
7-137	Example of PPG Output Period Made Longer 218
7-138	Interrupt Request Generation Using External Event Counter 220
7-139	Example Where Input of No Valid Edge Cannot Be Distinguished from Input of Only One Valid Edge with External Event Counter 220
7-140	How to Distinguish Input of No Valid Edge from Input of Only One Valid Edge with External Event Counter 221
7-141	Interrupt Generation Timing Change by an Erroneously Detected Edge 223
8-1	A/D Converter Configuration 226
8-2	Example of Capacitors Connected to the A/D Converter Pins 227
8-3	A/D Converter Mode Register (ADM) Format 229
8-4	Basic A/D Converter Operation 230
8-5	Relations between Analog Input Voltages and A/D Conversion Results 231
8-6	Select Mode Operation Timing 232
8-7	Scan Mode Operation Timing 233
8-8	Software-Started Select-Mode A/D Conversion 234

Fig. No.	Title, Page
8-9	Software-Started Scan-Mode A/D Conversion 235
8-10	Example of Malfunction in a Hardware-Started A/D Conversion 236
8-11	Select-Mode A/D Conversion Started by Hardware 237
8-12	Scan-Mode A/D Conversion Started by Hardware 238
8-13	Example of Capacitors Connected to the A/D Converter Pins 240
8-14	Example of Malfunction in a Hardware-Started A/D Conversion 241
9-1	Asynchronous Serial Interface Configuration 244
9-2	Format of the Asynchronous Serial Interface Mode Register (ASIM)..... 246
9-3	Format of the Asynchronous Serial Interface Status Register (ASIS) 247
9-4	Format of the Transmission/Reception Data at the Asynchronous Serial Interface 247
9-5	Asynchronous Serial Interface Transmission Completion Interrupt Timing 248
9-6	Asynchronous Serial Interface Reception Completion Interrupt Timing 249
9-7	Reception Error Timing 250
9-8	Baud Rate Generator Clock Configuration 251
9-9	Baud Rate Generator Control Register (BRGC) Format 252
10-1	Block Diagram of the Clock Synchronous Serial Interface 260
10-2	Format of the Clock Synchronous Serial Interface Mode Register (CSIM) 262
10-3	Format of Serial Bus Interface Control Register (SBIC) 264
10-4	Sample System Configuration with Three-Wire Serial I/O 265
10-5	Timing in Three-Wire Serial I/O Mode 266
10-6	Sample Connection with a Device Having Two-Wire Serial I/O 266
10-7	Sample Serial Bus Configured with SBI..... 269
10-8	Pin Configuration..... 270
10-9	Block Diagram of Clock Synchronous Serial Interface..... 271
10-10	Format of Clock Synchronous Serial Interface Mode Register (CSIM) 273
10-11	Format of SBIC Register 274
10-12	Configuration of Shift Register and Related Components 276
10-13	SBI Transfer Timing 277
10-14	Bus Release Signal 277
10-15	Command Signal 278
10-16	Address 278
10-17	Selecting a Slave Device by Its Address 278
10-18	Command 279
10-19	Data 279
10-20	Acknowledge Signal 279
10-21	Busy Signal and Ready Signal 280
10-22	Operation of RELT, CMDT, RELD, and CMDD 280
10-23	ACKT Operation 281
10-24	ACKE Operations 281
10-25	ACKD Operations..... 282
10-26	BSYE Operation 283
10-27	Sending an Address from Master Device to Slave Device 288
10-28	Sending a Command from Master Device to Slave Device 289
10-29	Sending Data from Master Device to Slave Device 290
10-30	Sending Data from the Slave Device to the Master Device 291

Fig. No.	Title, Page
11-1	Format of External Interrupt Mode Register 0 (INTM0) 294
11-2	Format of External Interrupt Mode Register 1 (INTM1) 295
11-3	Edge Detection on Pin P20 296
11-4	Edge Detection on Pins P21 to P26 297
11-5	Erroneously Detected Edges 297
11-6	Erroneously Detected Edges 299
12-1	INTM1 Register Format 303
12-2	ADM Register Format 304
12-3	Interrupt Request Flag Register (IF0) Format 305
12-4	Interrupt Mask Register (MK0) Format 306
12-5	Interrupt Service Mode Register (ISM0) Format 306
12-6	Priority Specification Flag Register (PR0) Format 307
12-7	Interrupt Status Register (IST) Format 307
12-8	Program Status Word Format 308
12-9	Accepting an NMI Interrupt Request 309
12-10	Interrupt Handling Algorithm 312
12-11	Example of Handling an Interrupt Request When an Interrupt Is Already Being Handled 314
12-12	Example of Handling Interrupts That Occur Simultaneously 316
12-13	Interrupt Request Generation and Acceptance 317
12-14	Differences between a Vectored Interrupt and Macro Service 319
12-15	Macro Service Processing Sequence 321
12-16	Macro Service Control Word Configuration 322
12-17	Macro Service Mode Register Format 323
12-18	Flow of Data Transfer by Macro Service (Type A) 325
12-19	Type A Macro Service Channel 326
12-20	Asynchronous Serial Reception 327
12-21	Flow of Data Transfer by Macro Service (Type B) 328
12-22	Type B Macro Service Channel 329
12-23	Parallel Data Input in Synchronization with an External Interrupt 330
12-24	Parallel Data Input Timing 330
12-25	Flow of Data Transfer by Macro Service (Type C) 332
12-26	Type C Macro Service Channel 334
12-27	Open-Loop Control for a Stepper Motor by the Real-Time Output Port 336
12-28	Data Transfer Control Timing 337
12-29	Four-Phase Stepping Motor with Phase 1 Excitation 338
12-30	Four-Phase Stepping Motor with Phases 1 and 2 Excitation 338
12-31	Block Diagram 1 for Automatic Addition Control Plus Ring Control (Constant-Speed Rotation with Phases 1 and 2 Excitation) 339
12-32	Timing Chart 1 for Automatic Addition Control Plus Ring Control (Constant-Speed Rotation with Phases 1 and 2 Excitation) 340
12-33	Block Diagram 2 for Automatic Addition Control Plus Ring Control (with the Output Timing Varied by Phase 2 Excitation) 341
12-34	Timing Chart 2 for Automatic Addition Control Plus Ring Control (with the Output Timing Varied by Phase 2 Excitation) 342

Fig. No.	Title, Page
13-1	Format of the Memory Expansion Mode Register (MM) 346
13-2	Format of Programmable Wait Control Register (PW) 347
13-3	Read Timing 348
13-4	Write Timing 348
13-5	Accessing Expansion Data Memory 349
13-6	Data Memory Expansion for μ PD78212 (When $\overline{EA} = L$) 351
13-7	Data Memory Expansion for μ PD78212 (When $\overline{EA} = H$) 352
13-8	Data Memory Expansion for μ PD78213 and μ PD78214 (When $\overline{EA} = L$) 353
13-9	Data Memory Expansion for μ PD78214 and μ PD78P214 (When $\overline{EA} = H$) 354
13-10	Example of Connecting Memories to μ PD78214 356
13-11	Wait Control Space of μ PD78212 (When $\overline{EA} = L$) 358
13-12	Wait Control Space of μ PD78212 (When $\overline{EA} = H$) 359
13-13	Wait Control Space of μ PD78213 and μ PD78214 (When $\overline{EA} = L$) 360
13-14	Wait Control Space of μ PD78214 and μ PD78P214 361
13-15	Read Timing of Programmable Wait Function 362
13-16	Write Timing of Programmable Wait Function 364
13-17	Timing When External Wait Signal Is Used 366
13-18	Format of Refresh Mode Register (RFM) 367
13-19	Pulse Refresh When Internal Memory Is Accessed 368
13-20	Pulse Refresh When External Memory Is Accessed 369
13-21	Restoration Timing from Self-Refresh 370
13-22	Return from Self-Refresh 371
13-23	Example of Connecting Pseudo Static RAM to μ PD78214 372
13-24	Return from Self-Refresh 374
13-25	Glitch Observed on Pins A16 to A19 during Emulation 374
13-26	Insufficient Address Hold Time during Emulation 374
13-27	Preventing Problems That May Occur during Emulation 375
14-1	Transition Diagram for the Standby Modes 377
14-2	Standby Function Block 378
14-3	Configuration of the Standby Control Register (STBC) 379
14-4	Releasing STOP Mode with an NMI Signal 383
14-5	Example of Longer Oscillation Settling Time 383
14-6	Example of Address Bus Arrangement 385
14-7	Example Address/Data Bus Arrangement 385
14-8	Example Arrangement for Analog Input Pin 386
14-9	Example of Longer Oscillation Settling Time 387
15-1	Acceptance of the RESET Signal 389
15-2	Reset Operation at Power-On 389
15-3	Timing Charts for Reset Operation 393
16-1	Example of Controlling Two Stepper Motors 396
16-2	Example System Configuration Using the Serial Bus Interface 397
16-3	Example of Communication with SBI 398
16-4	Serial Bus Communication Timing 398

Fig. No.	Title, Page
17-1	Timing Chart for PROM Write and Verify 400
17-2	Write Operation Flowchart401
17-3	PROM Read Timing Chart402

LIST OF TABLES

Table No.	Title, Page
2-1	Port 2 Functions 27
2-2	Port 3 Operating Mode 29
2-3	Port 6 Operating Mode 30
2-4	Types of I/O Circuits and Unused-Pin Handling 33
3-1	Vector Table 42
3-2	Selecting a Register Bank 46
3-3	Function Names and Absolute Names 49
3-4	Special Function Registers (SFR) 51
5-1	Port Functions 60
5-2	Number of I/O Ports 60
5-3	Functions of Port 2 63
5-4	Port 3 Operating Modes 67
5-5	Port 4 Operating Modes 76
5-6	Port 5 Operating Modes 81
5-7	Port 6 Operating Modes 84
5-8	Port 6 Control Pin Functions and the Required Operations 88
6-1	Port 0 Operating Modes and Operations Needed for the Port 0 Buffer Registers 98
6-2	Output Trigger for the Real-Time Output Port 99
7-1	Timer/Counter Types and Functions 107
7-2	Intervals of 16-Bit Timer/Counter 109
7-3	Programmable Square Wave Output Setting Range of 16-Bit Timer/Counter 109
7-4	Pulse Width Measurement Range of 16-Bit Timer/Counter 109
7-5	Timer Output (TO0, TO1) Operation 120
7-6	TO0 and TO1 Toggle Output 121
7-7	PWM Output on TO0 and TO1 122
7-8	PPG Output on TO0 125
7-9	Intervals of 8-Bit Timer/Counter 1 139
7-10	Pulse Width Measurement Range of 8-Bit Timer/Counter 1 139
7-11	Intervals of 8-Bit Timer/Counter 2 159
7-12	Programmable Square Wave Output Setting Range of 8-Bit Timer/Counter 2 160
7-13	Pulse Width Measurement Range of 8-Bit Timer/Counter 2 160
7-14	Clock Signals That Can Be Applied to 8-Bit Timer/Counter 2 161
7-15	Timer Output (TO2, TO3) Operation 180
7-16	TO2 and TO3 Toggle Output 182
7-17	PWM Output on TO2 and TO3 183
7-18	PPG Output on TO2 186
7-19	Intervals of 8-Bit Timer/Counter 3 205
7-20	Maximum Number of Wait States Inserted When Registers Associated with Timers/Counters Are Accessed 215

Table No.	Title, Page
8-1	Modes Generating the INTAD..... 225
8-2	A/D Conversion Time 232
8-3	Conditions to Generate Interrupt Requests in Each A/D Converter Operating Mode 239
9-1	Causes of Reception Errors 250
9-2	Baud Rate Setting 254
9-3	Example of Setting the BRGC Register When the Baud Rate Generator for UART Is Used 255
9-4	Example of Setting the Baud Rate When 8-Bit Timer/Counter 3 Is Used (Asynchronous Serial Interface) 257
9-5	Examples of Setting the BRGC When an External Baud Rate Input (ASCK) Is Used 258
10-1	Reading/Writing the Contents of the SBIC Register 263
10-2	Signals in SBI Mode 284
10-3	Conditions Governing Release of BUSY 287
11-1	Pins P20 to P26 and Use of Detected Edge 293
12-1	Interrupt Request Handling Modes 301
12-2	Interrupt Request Sources 302
12-3	Flags for Interrupt Request Sources 305
12-4	Multiple-Interrupt Handling 313
12-5	Interrupt Request Acceptance Processing Time 317
12-6	Macro Service Processing Time 318
12-7	Interrupts That Can Use a Macro Service 320
12-8	Interrupt Requests That Can Specify Macro Service and Related SFRs (Type A) 324
12-9	Illegal Write Access Conditions and Corresponding Operations 324
12-10	Interrupt Requests That Can Specify Macro Service and SFRs (Type C) 331
12-11	Illegal Write Access Conditions and Corresponding Operations 331
12-12	Illegal Write Access Conditions and Corresponding Operations 344
13-1	Conditions and Operations for Illegal Write Access 350
13-2	System Clock Frequency and Refresh Pulse Output Cycle When Pseudo Static RAM Is Used 368
13-3	Conditions and Operations for Illegal Write Access 373
14-1	Operation States in HALT Mode 379
14-2	Sources for Releasing HALT Mode and Operations Performed After Release 380
14-3	Release of HALT Mode by a Maskable Interrupt Request 381
14-4	Operation States in STOP Mode 382
15-1	Pin States during Reset and After Reset State Is Released 390
15-2	Hardware States after Reset 391
17-1	Operating Modes for PROM Programming 399
18-1	8-Bit Instructions for Each Addressing Type 416
18-2	16-Bit Instructions for Each Addressing Type 417
18-3	Bit Manipulation Instructions for Each Addressing Type 418
18-4	Call Instructions and Branch Instructions for Each Addressing Type 419

CHAPTER 1 GENERAL

1

The μ PD78214 sub-series is part of the 78K/II series of eight-bit single-chip microcomputers capable of accessing an expanded memory space of 1 megabyte. This sub-series consists of the following products.

The μ PD78214 offers a 16-KB masked ROM, 512-byte RAM, highly functional timers/counters, a high-precision A/D converter, and two independent serial interfaces.

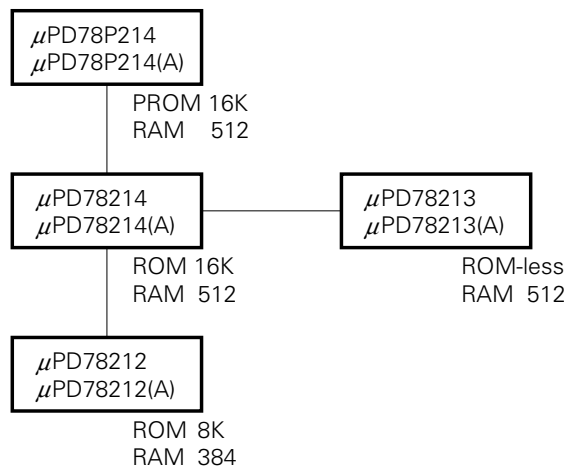
The μ PD78212 is the same as the μ PD78214 except that it offers an 8-KB ROM and 384-byte RAM.

The μ PD78213 is the same as the μ PD78214 except that it has no built-in ROM.

The μ PD78P214 is the PROM version (used in place of masked ROM) of the μ PD78214.

- μ PD78P214DW: Programs can be written repeatedly (suitable for evaluating an application system).
- Others : A program can be written once (suitable for application systems produced in small lots).

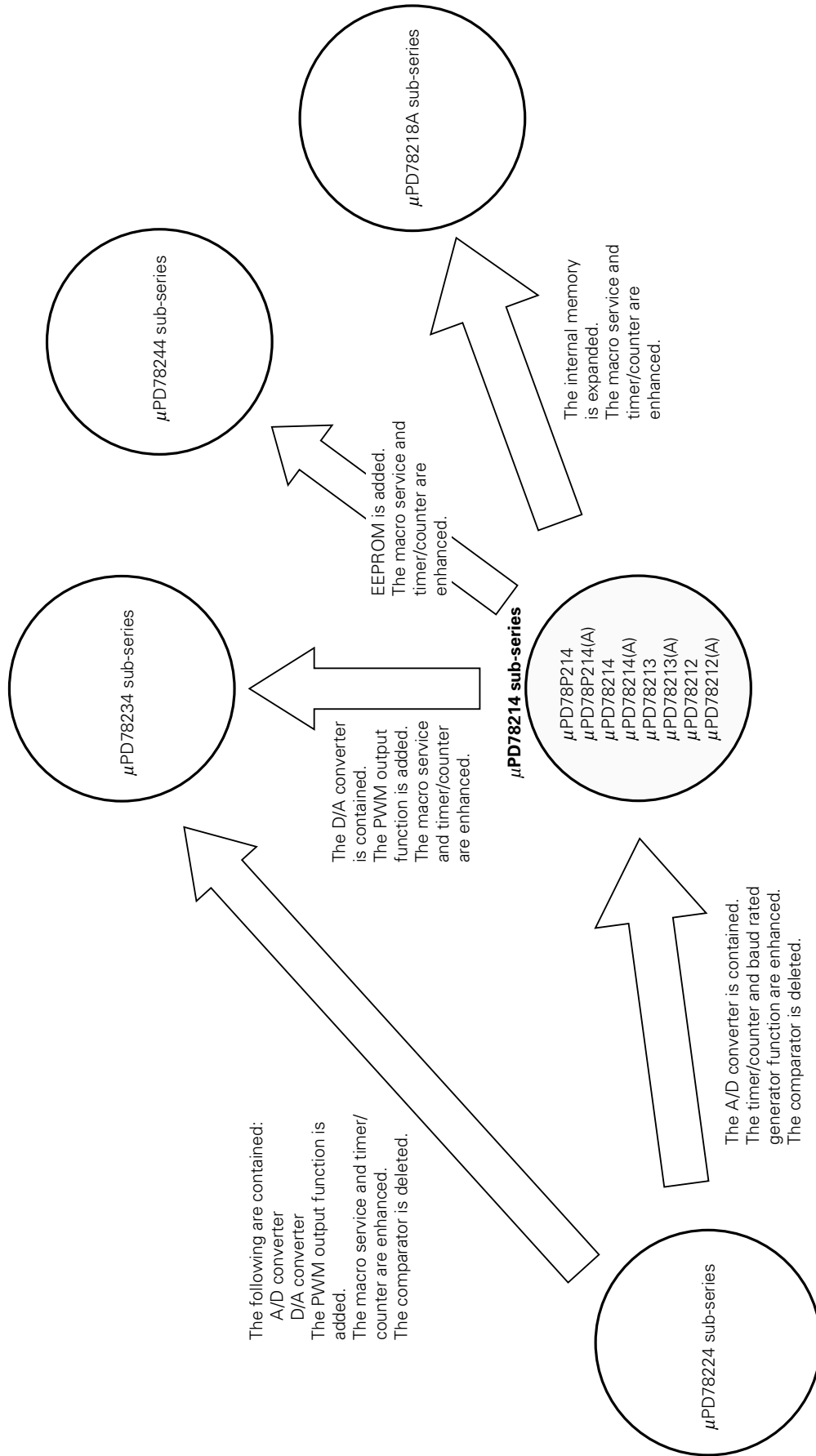
The μ PD78212(A), μ PD78213(A), μ PD78214(A), and μ PD78P214(A) are the special quality versions of the μ PD78212, μ PD78213, μ PD78214, and μ PD78P214, respectively.



This sub-series can be applied to the following:

- Standard-quality products
 - Printers
 - Electronic typewriters
 - Electronic cash registers (ECRs)
 - Plain paper copiers (PPCs)
 - Electronic musical instruments
 - Air conditioners
 - Cellular phones
 - Cameras
- Special-quality products
 - Vehicle-mounted electrical equipment
 - Combustion control

78K/II Products



1.1 FEATURES

- 78K/II series
- Multiplexed internal bus (faster execution of instructions)
 - Minimum instruction cycle (operating at 12 MHz): 333 ns (μ PD78212, μ PD78214, and μ PD78P214), or 500 ns (μ PD78213)
- Instruction set suitable for control applications
- Data memory expansion function (memory space of 1MB with two bank designation pointers)
- Interrupt controller (with two priority levels)
 - Vectored interrupt handling
 - Macro service
- Internal memory
 - ROM
 - Masked ROM : 16KB (μ PD78214), 8KB (μ PD78212), or none (μ PD78213)
 - PROM : 16KB (μ PD78P214)
 - RAM: 512 bytes (μ PD78213, μ PD78214, and μ PD78P214), or 384 bytes (μ PD78212)
- Number of I/O pins: 54 (μ PD78212, μ PD78214, and μ PD78P214), or 36 (μ PD78213)
 - Number of pins with software-programmable pull-up resistors: 16 (μ PD78213 only), or 34 (other than μ PD78213)
 - Number of LED direct-drive pins: 16 (μ PD78212, μ PD78214, and μ PD78P214)
 - Number of transistor direct-drive pins: 8
- Serial interface
 - UART (baud rate generator included)
 - Synchronous serial interface (three-wire serial I/O, serial bus interface)
- Real-time output ports (two stepper motors can be independently controlled by combining the output ports with an eight-bit timer/counter.)
- Eight-bit A/D converter (analog eight-bit input)
- Highly functional timer/counter units
 - 16-bit unit
 - Three eight-bit units

1.2 ORDERING INFORMATION AND QUALITY GRADE

1.2.1 Ordering Information

Ordering code	Package	Internal ROM
μPD78212CW-xxx	64-pin plastic shrink DIP (750 mil)	Masked ROM
μPD78212GC-xxx-AB8	64-pin plastic QFP (14 × 14 mm)	Masked ROM
μPD78212GJ-xxx-5BJ	74-pin plastic QFP (20 × 20 mm)	Masked ROM
μPD78213CW	64-pin plastic shrink DIP (750 mil)	None
μPD78213GC-AB8	64-pin plastic QFP (14 × 14 mm)	None
μPD78213GJ-5BJ	74-pin plastic QFP (20 × 20 mm)	None
μPD78213GQ-36	64-pin plastic QUIP	None
μPD78213L	68-pin plastic QFJ	None
μPD78214CW-xxx	64-pin plastic shrink DIP (750 mil)	Masked ROM
μPD78214GC-xxx-AB8	64-pin plastic QFP (14 × 14 mm)	Masked ROM
μPD78214GJ-xxx-5BJ	74-pin plastic QFP (20 × 20 mm)	Masked ROM
μPD78214GQ-xxx-36	64-pin plastic QUIP	Masked ROM
μPD78214L-xxx	68-pin plastic QFJ	Masked ROM
μPD78P214CW	64-pin plastic shrink DIP (750 mil)	One-time PROM
μPD78P214GC-AB8	64-pin plastic QFP (14 × 14 mm)	One-time PROM
μPD78P214GJ-5BJ	74-pin plastic QFP (20 × 20 mm)	One-time PROM
μPD78P214GQ-36	64-pin plastic QUIP	One-time PROM
μPD78P214L	68-pin plastic QFJ	One-time PROM
μPD78P214DW	64-pin ceramic shrink DIP with window (750 mil)	EPROM
μPD78P214CW-xxx Note	64-pin plastic shrink DIP (750 mil)	Program-written one-time PROM
μPD78P214GC-xxx-AB8 Note	64-pin plastic QFP (14 × 14 mm)	Program-written one-time PROM
μPD78P214GJ-xxx-5BJ Note	74-pin plastic QFP (20 × 20 mm)	Program-written one-time PROM
μPD78P214GQ-xxx-36 Note	64-pin plastic QUIP	Program-written one-time PROM
μPD78P214L-xxx Note	68-pin plastic QFJ	Program-written one-time PROM
μPD78212CW (A)-xxx	64-pin plastic shrink DIP (750 mil)	Masked ROM
μPD78212GC (A)-xxx-AB8	64-pin plastic QFP (14 × 14 mm)	Masked ROM
μPD78213CW (A)	64-pin plastic shrink DIP (750 mil)	None
μPD78213GQ (A)-36	64-pin plastic QUIP	None
μPD78214CW (A)-xxx	64-pin plastic shrink DIP (750 mil)	Masked ROM
μPD78214GC (A)-xxx-AB8	64-pin plastic QFP (14 × 14 mm)	Masked ROM
μPD78214GJ (A)-xxx-5BJ	74-pin plastic QFP (20 × 20 mm)	Masked ROM
μPD78214GQ (A)-xxx-36	64-pin plastic QUIP	Masked ROM
μPD78214L (A)-xxx	68-pin plastic QFJ	Masked ROM
μPD78P214CW (A)	64-pin plastic shrink DIP (750 mil)	One-time PROM
μPD78P214GC (A)-AB8	64-pin plastic QFP (14 × 14 mm)	One-time PROM
μPD78P214CW (A)-xxx Note	64-pin plastic shrink DIP (750 mil)	Program-written one-time PROM

Note QTOPT™ microcomputers. QTOP microcomputers are a line of one-time PROM single-chip microcomputers that are fully supported by NEC, from writing of the program, through marking and screening, to verifying.

Remark xxx indicates the ROM code number.

1.2.2 Quality Grade

Ordering code	Package	Quality grade
μ PD78212CW-xxx	64-pin plastic shrink DIP (750 mil)	Standard
μ PD78212GC-xxx-AB8	64-pin plastic QFP (14 × 14 mm)	Standard
μ PD78212GJ-xxx-5BJ	74-pin plastic QFP (20 × 20 mm)	Standard
μ PD78213CW	64-pin plastic shrink DIP (750 mil)	Standard
μ PD78213GC-AB8	64-pin plastic QFP (14 × 14 mm)	Standard
μ PD78213GJ-5BJ	74-pin plastic QFP (20 × 20 mm)	Standard
μ PD78213GQ-36	64-pin plastic QUIP	Standard
μ PD78213L	68-pin plastic QFJ	Standard
μ PD78214CW-xxx	64-pin plastic shrink DIP (750 mil)	Standard
μ PD78214GC-xxx-AB8	64-pin plastic QFP (14 × 14 mm)	Standard
μ PD78214GJ-xxx-5BJ	74-pin plastic QFP (20 × 20 mm)	Standard
μ PD78214GQ-xxx-36	64-pin plastic QUIP	Standard
μ PD78214L-xxx	68-pin plastic QFJ	Standard
μ PD78P214CW	64-pin plastic shrink DIP (750 mil)	Standard
μ PD78P214GC-AB8	64-pin plastic QFP (14 × 14 mm)	Standard
μ PD78P214GJ-5BJ	74-pin plastic QFP (20 × 20 mm)	Standard
μ PD78P214GQ-36	64-pin plastic QUIP	Standard
μ PD78P214L	68-pin plastic QFJ	Standard
μ PD78P214DW	64-pin ceramic shrink DIP with window (750 mil)	Standard
μ PD78P214CW-xxx Note	64-pin plastic shrink DIP (750 mil)	Standard
μ PD78P214GC-xxx-AB8 Note	64-pin plastic QFP (14 × 14 mm)	Standard
μ PD78P214GJ-xxx-5BJ Note	74-pin plastic QFP (20 × 20 mm)	Standard
μ PD78P214GQ-xxx-36 Note	64-pin plastic QUIP	Standard
μ PD78P214L-xxx Note	68-pin plastic QFJ	Standard
μ PD78212CW (A)-xxx	64-pin plastic shrink DIP (750 mil)	Special
μ PD78212GC (A)-xxx-AB8	64-pin plastic QFP (14 × 14 mm)	Special
μ PD78213CW (A)	64-pin plastic shrink DIP (750 mil)	Special
μ PD78213GQ (A)-36	64-pin plastic QUIP	Special
μ PD78214CW (A)-xxx	64-pin plastic shrink DIP (750 mil)	Special
μ PD78214GC (A)-xxx-AB8	64-pin plastic QFP (14 × 14 mm)	Special
μ PD78214GJ (A)-xxx-5BJ	74-pin plastic QFP (20 × 20 mm)	Special
μ PD78214GQ (A)-xxx-36	64-pin plastic QUIP	Special
μ PD78214L (A)-xxx	68-pin plastic QFJ	Special
μ PD78P214CW (A)	64-pin plastic shrink DIP (750 mil)	Special
μ PD78P214GC (A)-AB8	64-pin plastic QFP (14 × 14 mm)	Special
μ PD78P214CW (A)-xxx Note	64-pin plastic shrink DIP (750 mil)	Special

Note QTOP™ microcomputers. QTOP microcomputers are a line of one-time PROM single-chip microcomputers that are fully supported by NEC, from writing of the program, through marking and screening, to verifying.

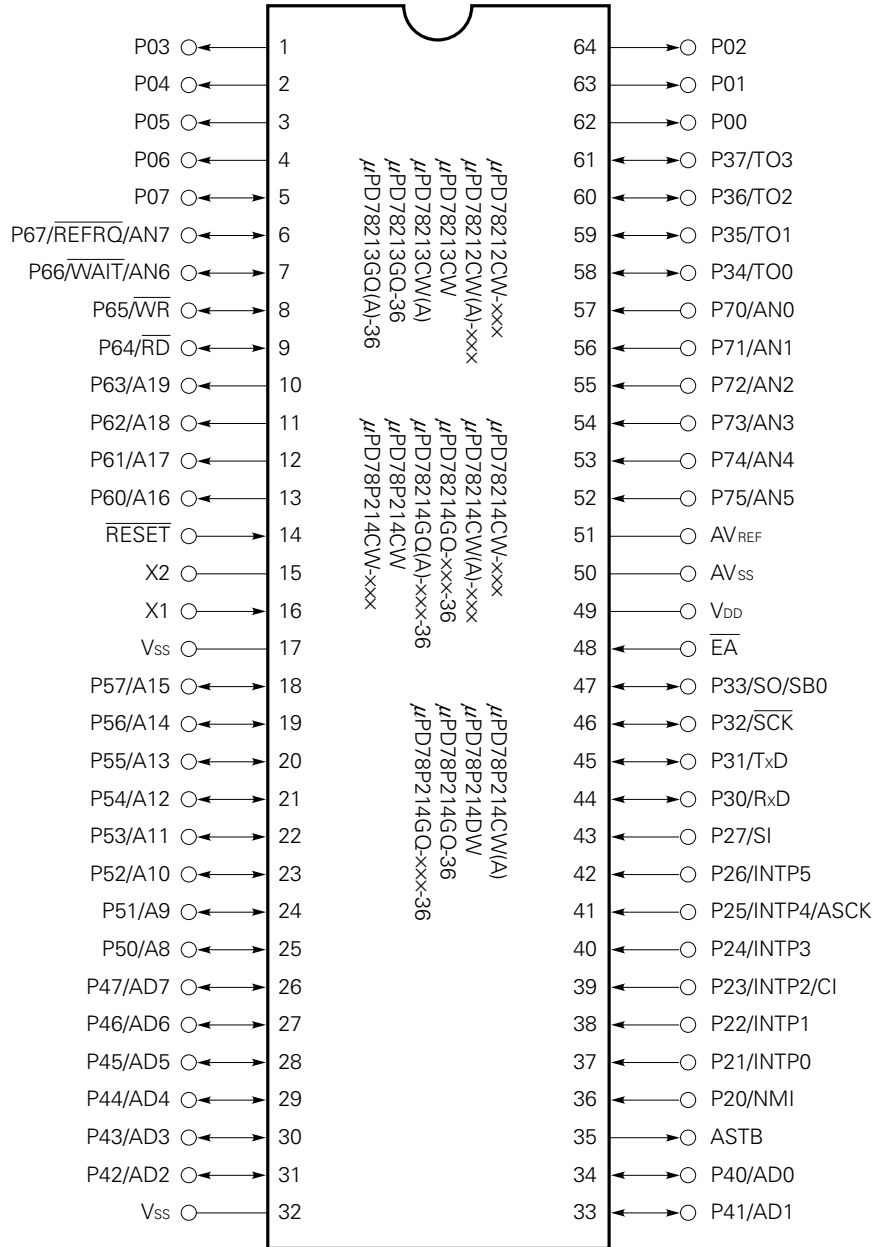
Remark xxx indicates the ROM code number.

Refer to "Quality grade on NEC Semiconductor Devices" (Document number IEI-1209), published by NEC Corporation, for the specifications of the quality grade of the devices and the recommended applications.

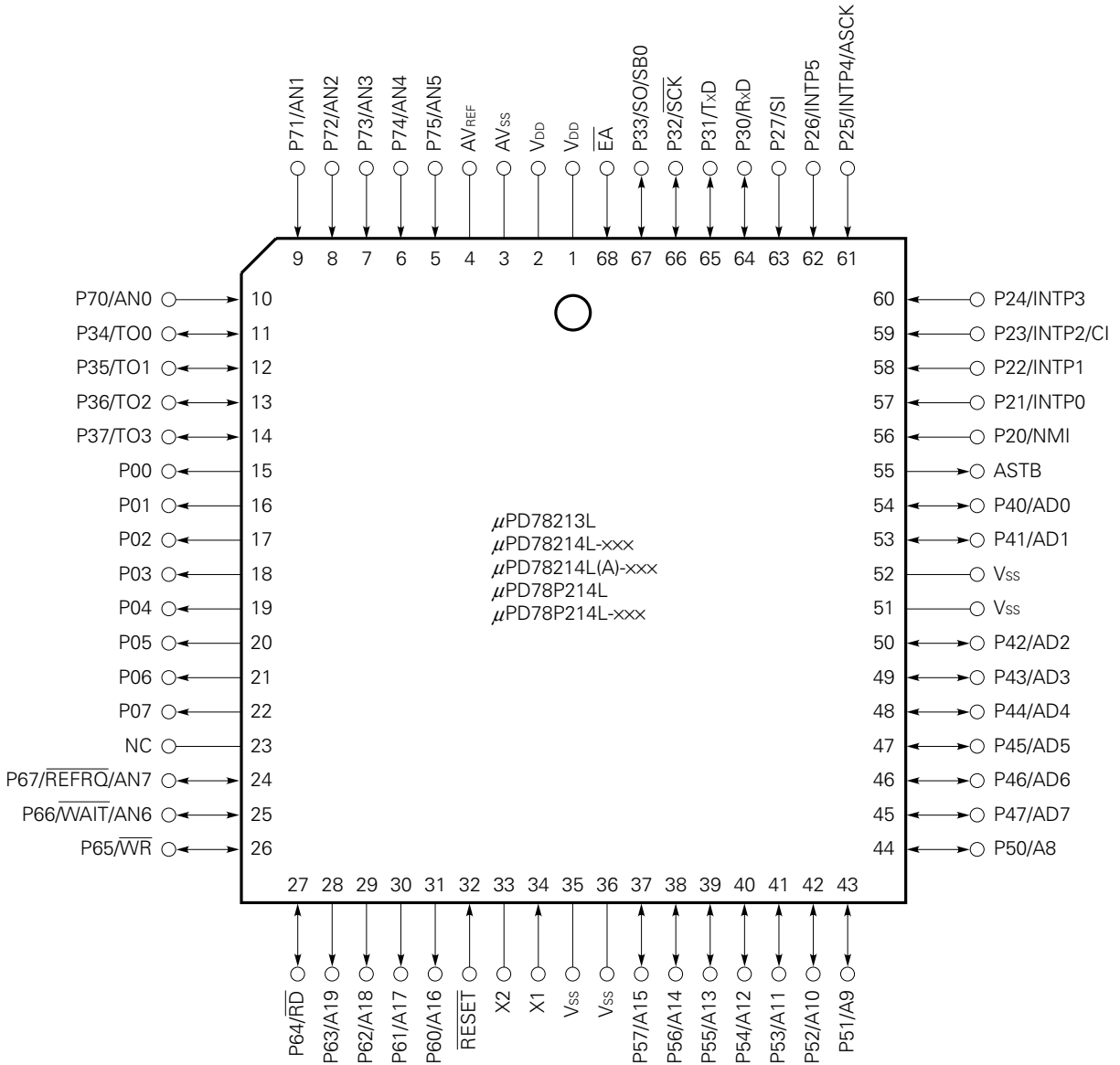
1.3 PIN CONFIGURATION (TOP VIEW)

1.3.1 Normal Operating Mode

(1) 64-pin plastic shrink DIP, 64-pin plastic QUIP, 64-pin ceramic shrink DIP with window

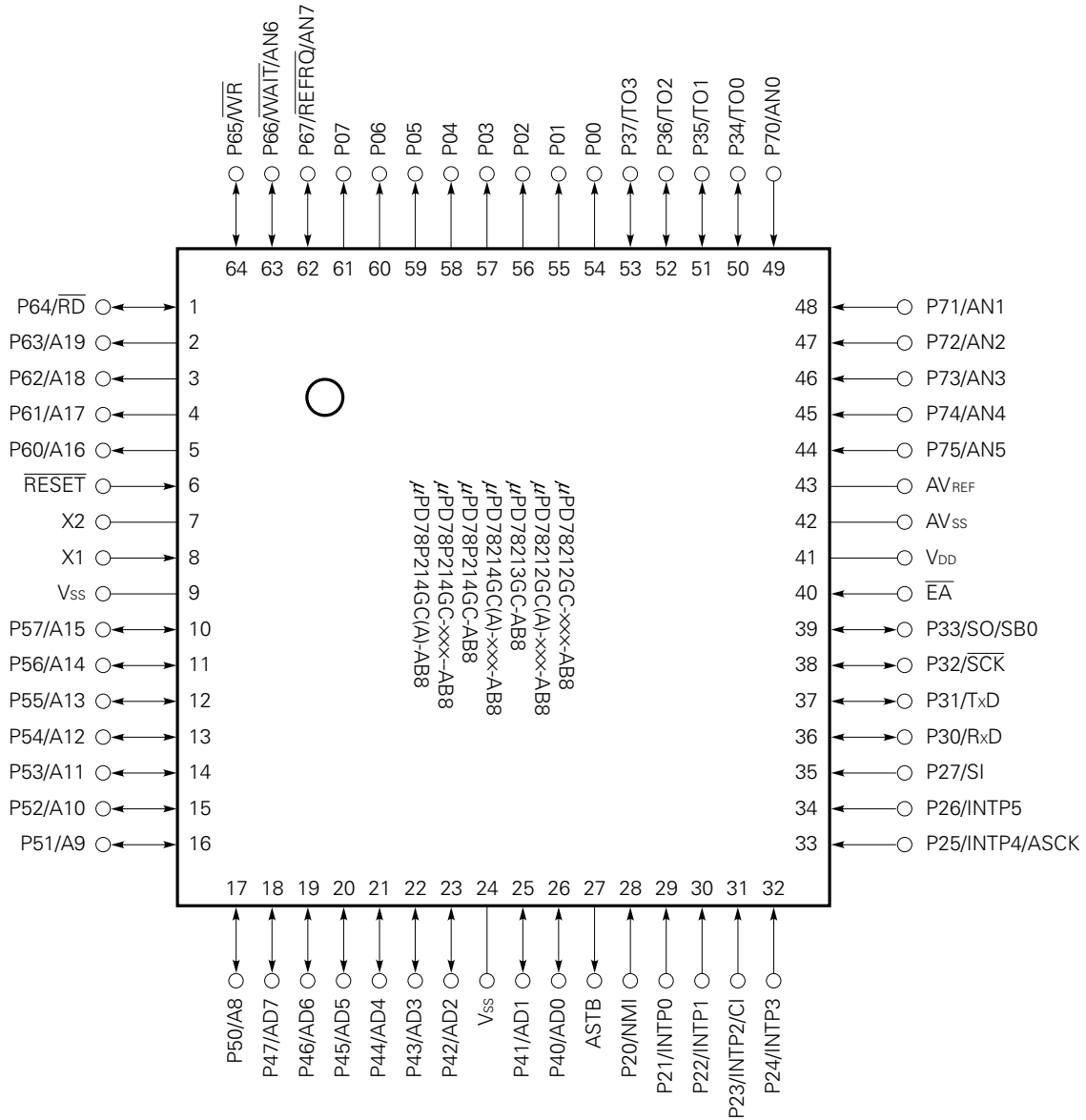


(2) 68-pin plastic QFJ

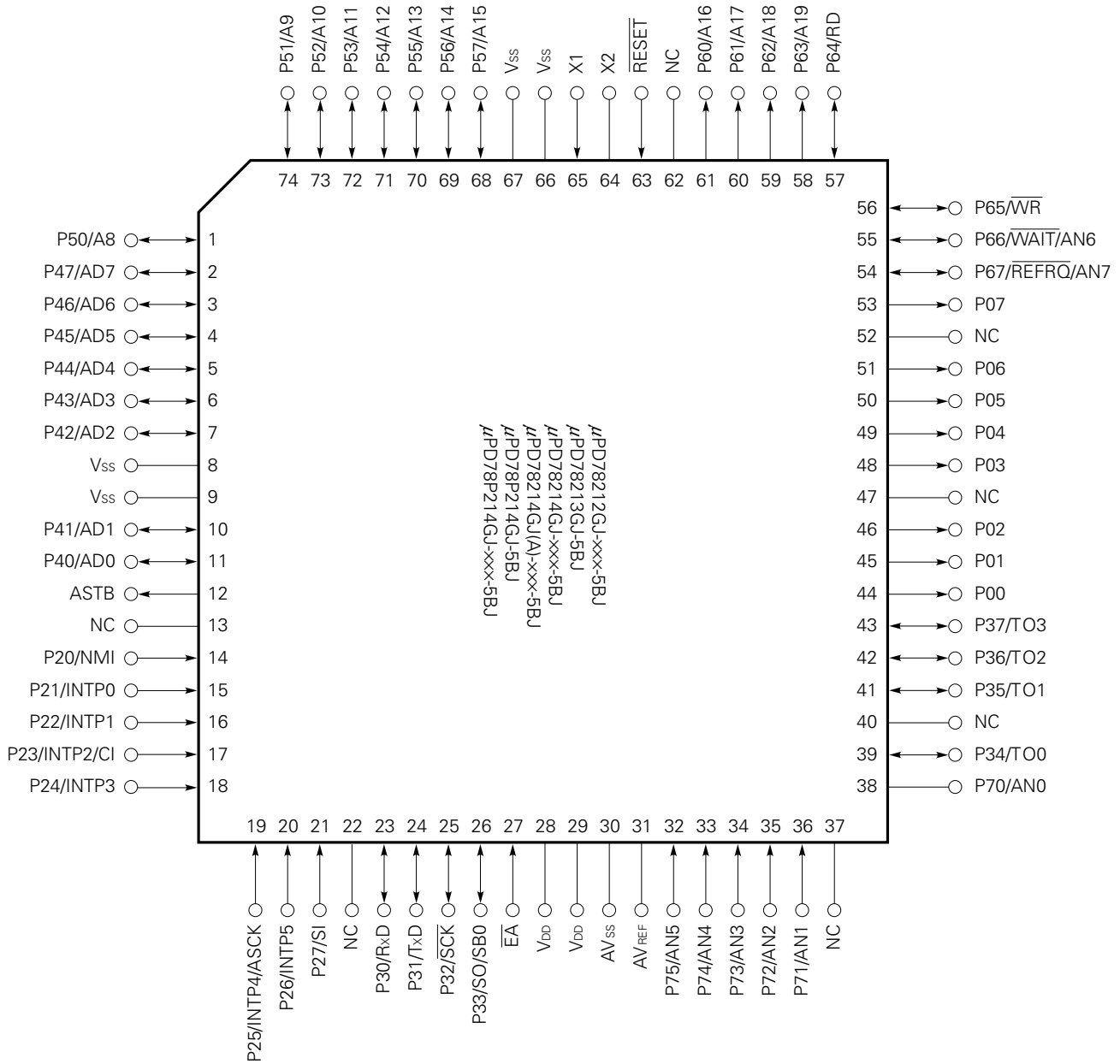


Remark The NC pin is not connected inside the chip.

(3) 64-pin plastic QFP (14 × 14 mm)



(4) 74-pin plastic QFP (20 × 20 mm)

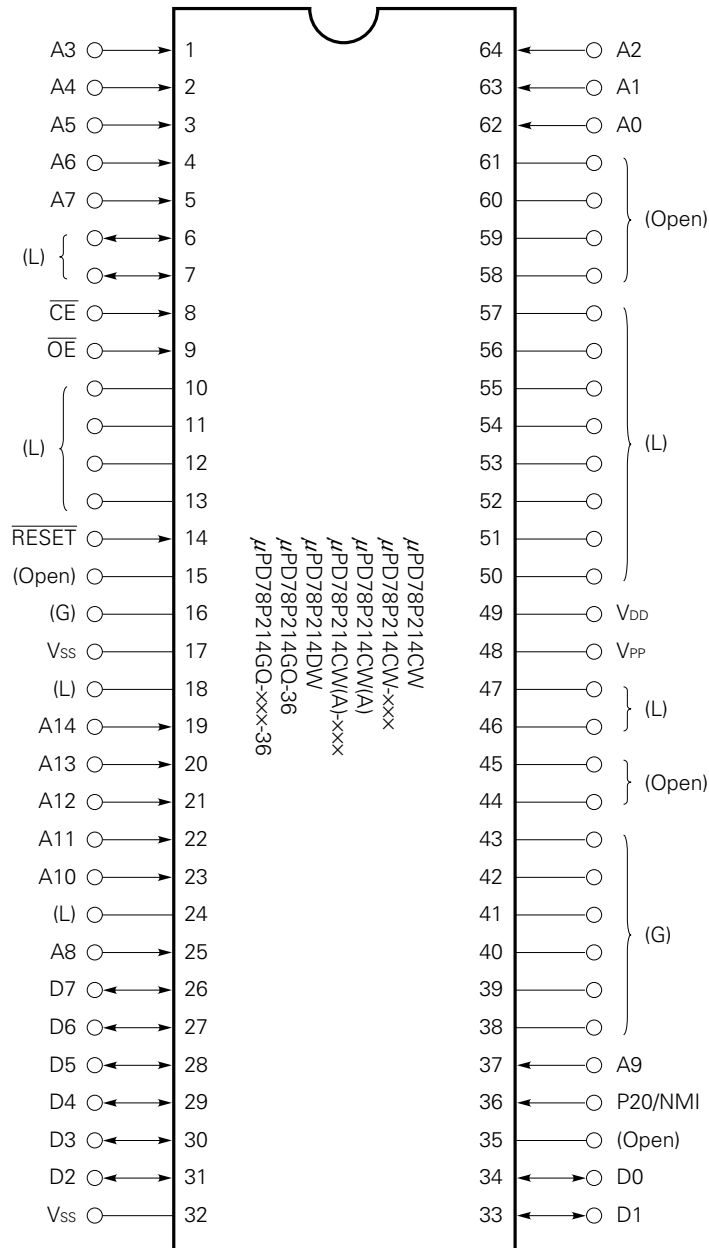


Remark The NC pins are not connected inside the chip.

P00-P07	: Port 0
P20-P27	: Port 2
P30-P37	: Port 3
P40-P47	: Port 4
P50-P57	: Port 5
P60-P67	: Port 6
P70-P75	: Port 7
TO0-TO3	: Timer output
Cl	: Clock input
RxD	: Receive data
TxD	: Transmit data
$\overline{\text{SCK}}$: Serial clock
ASCK	: Asynchronous serial clock
SBO	: Serial bus
SI	: Serial input
SO	: Serial output
NMI	: Non-maskable interrupt
INTP0-INTP5	: Interrupt from peripherals
AD0-AD7	: Address/data bus
A8-A19	: Address bus
$\overline{\text{RD}}$: Read strobe
$\overline{\text{WR}}$: Write strobe
$\overline{\text{WAIT}}$: Wait
ASTB	: Address strobe
$\overline{\text{REFRQ}}$: Refresh request
$\overline{\text{RESET}}$: Reset
X1, X2	: Crystal
$\overline{\text{EA}}$: External access
AN0-AN7	: Analog input
AV _{REF}	: Reference voltage
AV _{SS}	: Analog ground
V _{DD}	: Power supply
V _{SS}	: Ground
NC	: Non-connection

1.3.2 PROM Programming Mode (P20/NMI = 12.5 V, $\overline{\text{RESET}} = \text{L}$)

(1) 64-pin plastic shrink DIP, 64-pin plastic QUIP, 64-pin ceramic shrink DIP with window



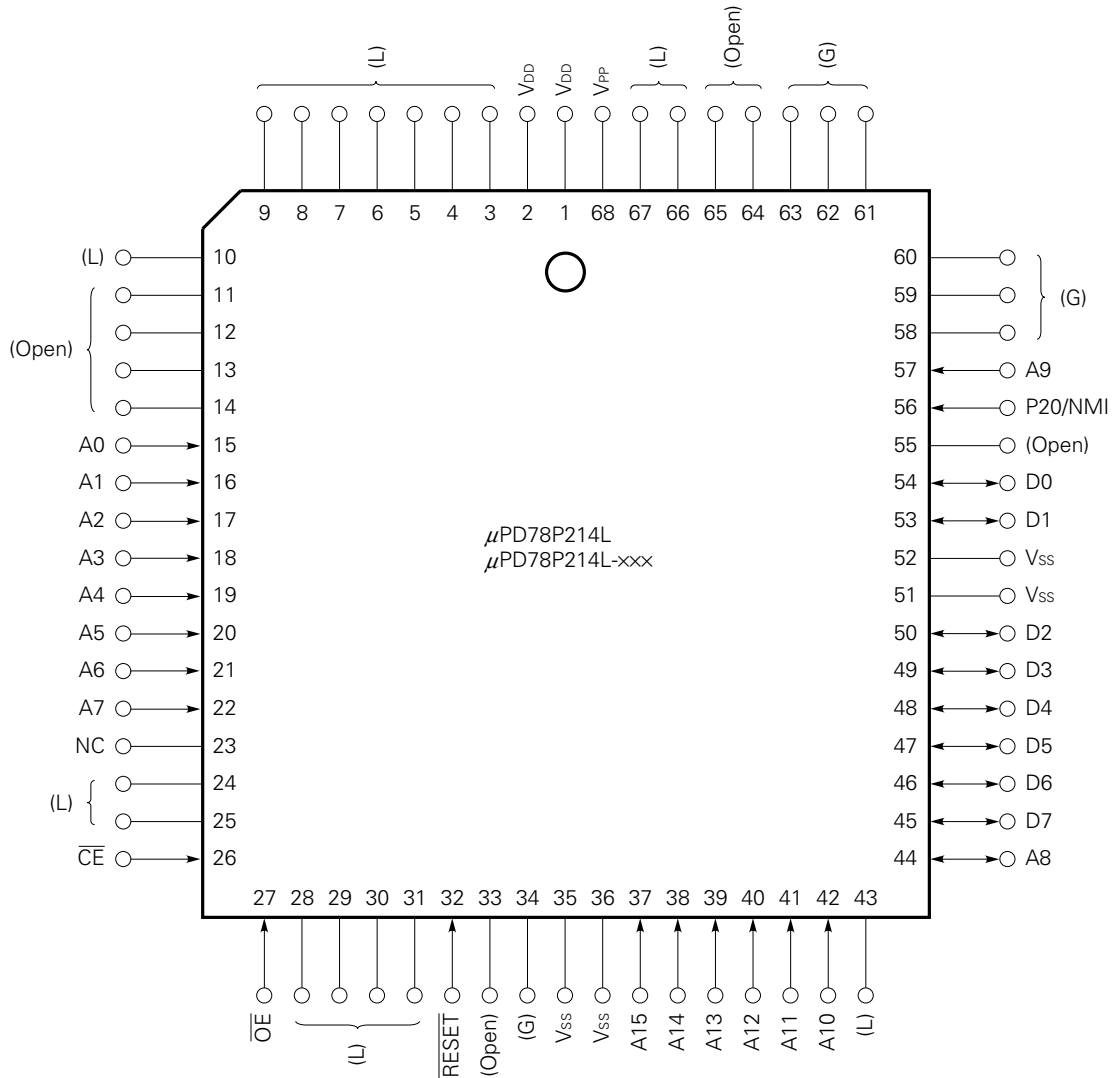
Caution The symbols enclosed in parentheses indicate that the corresponding pins, not used in PROM programming mode, shall be handled as follows:

L : Connect the corresponding pin independently to V_{SS} , through a 10-k Ω resistor. ★

G : Connect the corresponding pin to V_{SS} .

Open : Leave the corresponding pin unconnected.

(2) 68-pin plastic QFJ

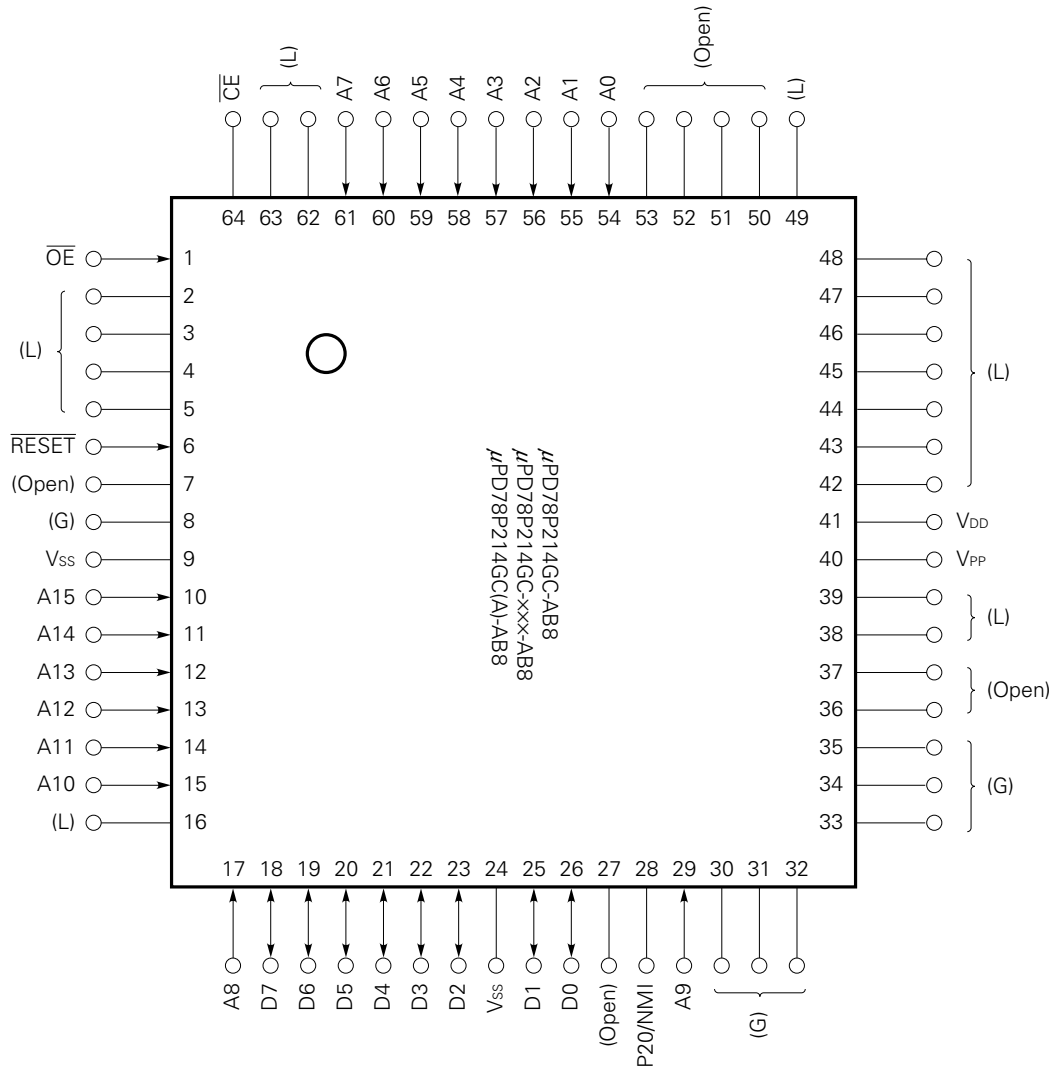


Caution The symbols enclosed in parentheses indicate that the corresponding pins, not used in PROM programming mode, shall be handled as follows:

- ★ **L** : Connect the corresponding pin independently to V_{SS}, through a 10-kΩ resistor.
- G** : Connect the corresponding pin to V_{SS}.
- Open** : Leave the corresponding pin unconnected.

Remark The NC pin is not connected inside the chip.

(3) 64-pin plastic QFP (14 × 14 mm)



Caution The symbols enclosed in parentheses indicate that the corresponding pins, not used in PROM programming mode, shall be handled as follows:

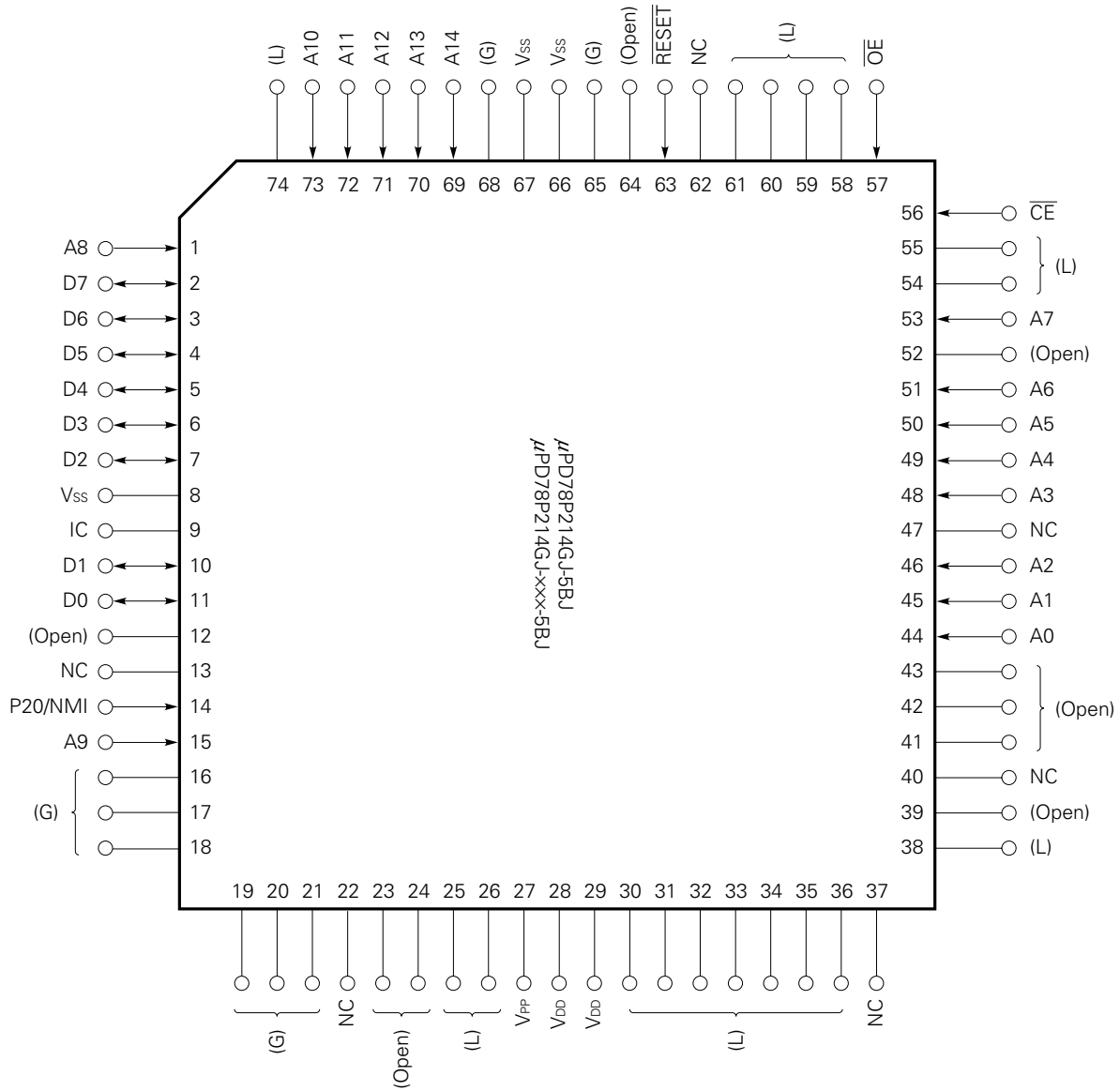
L : Connect the corresponding pin independently to V_{SS} , through a 10-k Ω resistor.

G : Connect the corresponding pin to V_{SS} .

Open : Leave the corresponding pin unconnected.



(4) 74-pin plastic QFP (20 × 20 mm)



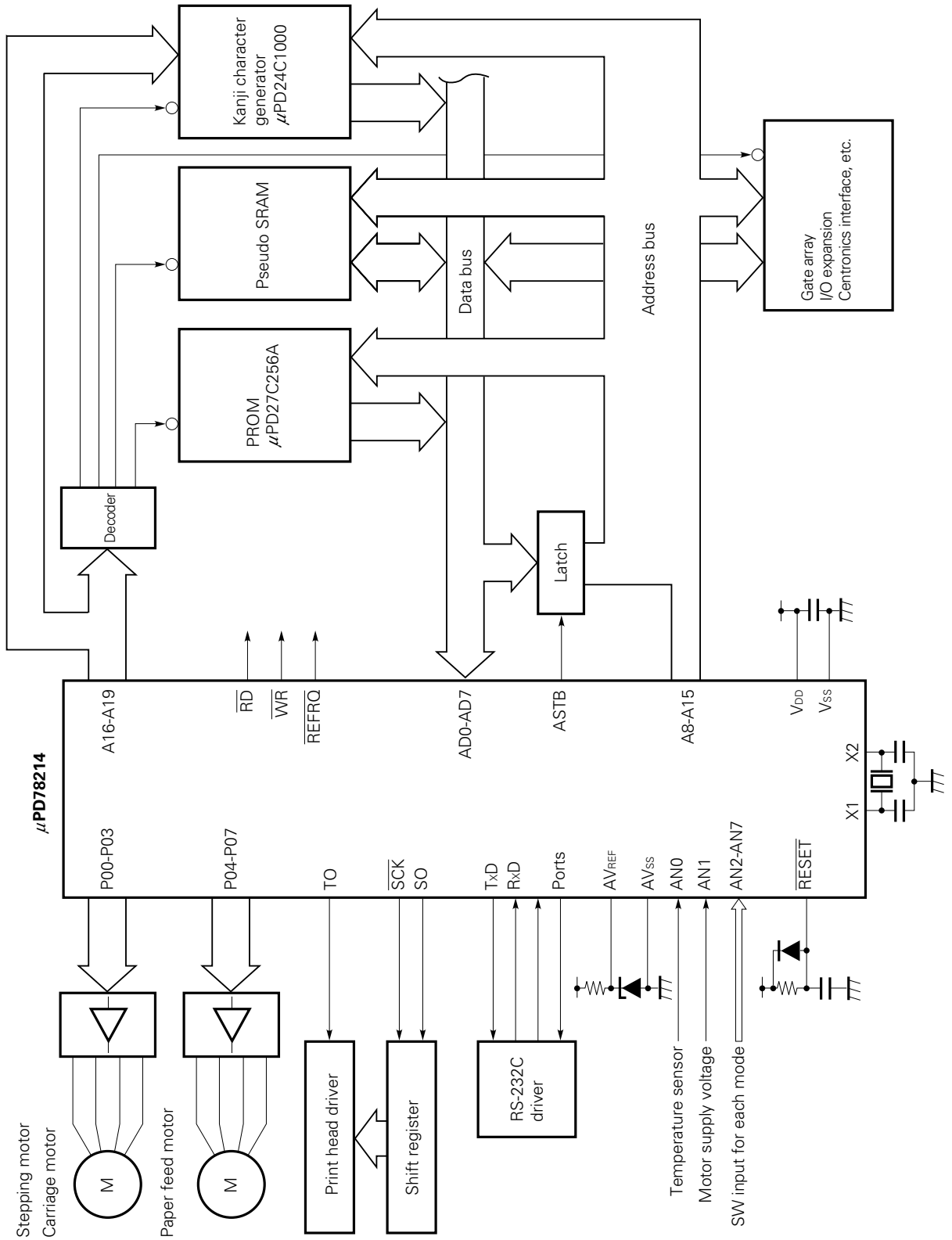
Caution The symbols enclosed in parentheses indicate that the corresponding pins, not used in PROM programming mode, shall be handled as follows:

- ★ **L** : Connect the corresponding pin independently to V_{SS} , through a 10-kΩ resistor.
- G** : Connect the corresponding pin to V_{SS} .
- Open** : Leave the corresponding pin unconnected.

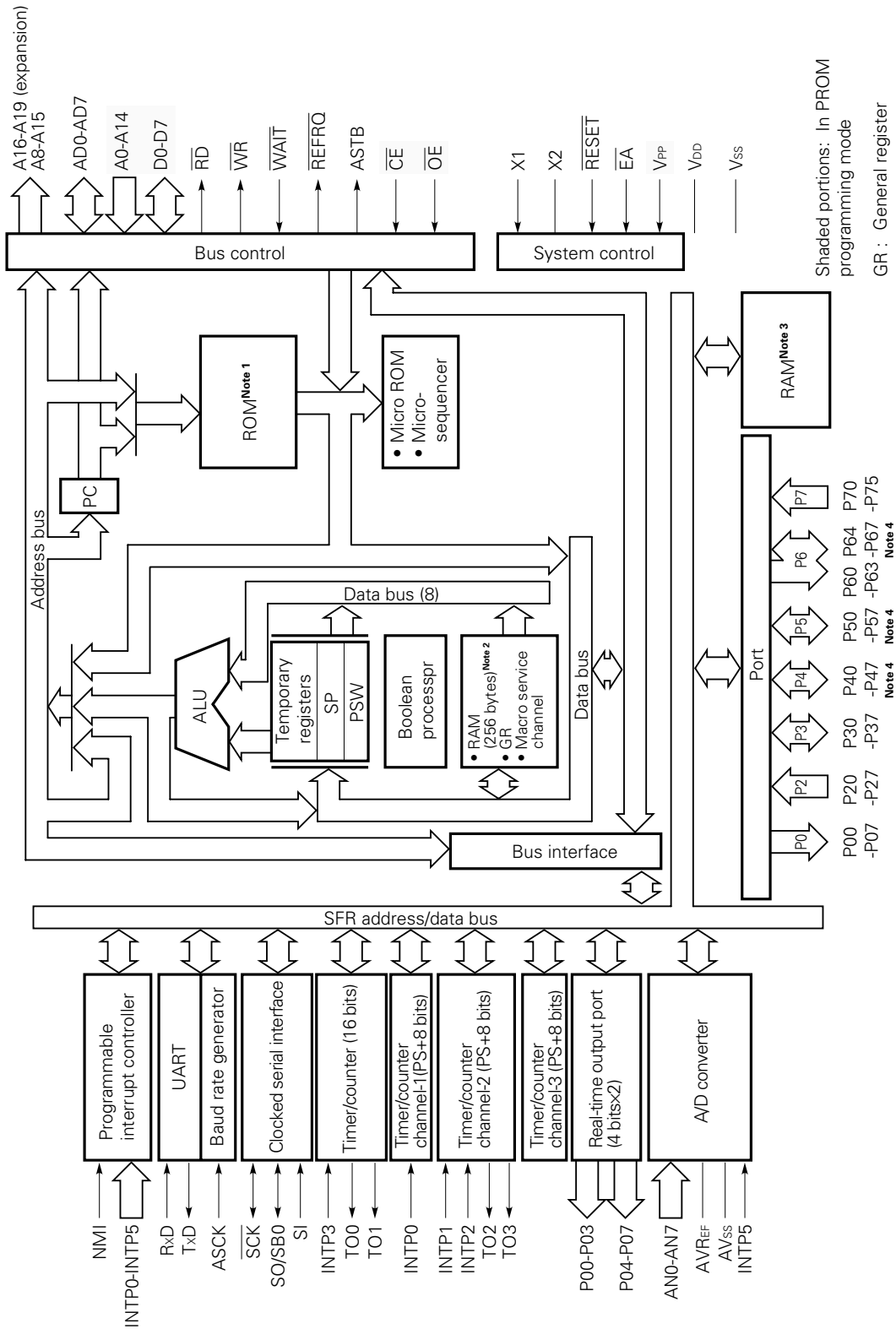
Remark The NC pins are not connected inside the chip.

V_{PP} : Programming power supply
 \overline{RESET} : Reset
D0-D7 : Data bus
A0-A14 : Address bus
 V_{SS} : Ground
 \overline{OE} : Output enable
 V_{DD} : Power supply
 \overline{CE} : Chip enable
P20/NMI : Port 2/non-maskable interrupt
NC : Non-connection

1.4 EXAMPLE APPLICATION SYSTEM (PRINTER)



1.5 BLOCK DIAGRAM



- Notes**
- None for μ PD78213(A), 8KB for μ PD78212 and μ PD78212(A), 16KB for μ PD78214, μ PD78214(A)
 - Internal dual-port RAM
 - Peripheral RAM (PRAM). 128 bytes for μ PD78212 and μ PD78212(A), 256 bytes for μ PD78213, μ PD78214, μ PD78214(A), and μ PD78214(A)
 - For the μ PD78213, P40-P47, P50-P57, P64, or P65 do not operate as ports.

1.6 FUNCTIONS

Item		μPD78212	μPD78214	μPD78P214	μPD78213
Number of basic instructions (mnemonics)		65			
Minimum instruction execution time (when operating at 12 MHz)		333 ns			500 ns
Capacity of internal memory	ROM	8K bytes	16K bytes		None
	RAM	384 bytes	512 bytes		
Memory area		64KB for program and 1MB for data			
Number of I/O pins	Input pins	14			
	Output pins	12			
	I/O pins	28			10
	Total	54			36
Special-function pins <small>Note</small>	Connected to a pull-up resistor	34			16
	Driving a LED directly	16			0
	Driving a transistor directly	8			
Real-time output ports		Two four-bit ports or one eight-bit port			
General-purpose registers		Four banks of eight eight-bit registers (memory mapping)			
Timer/counters		16-bit timer/counter, consisting of one timer register, one capture register, and two compare registers.		Pulse output possible (toggle output or PWM/PPG output)	
		8-bit timer/counter unit 1, consisting of one timer register, one capture/compare register, and one compare register		Pulse output possible (two four-bit, real-time outputs)	
		8-bit timer/counter unit 2, consisting of one timer register, one capture register, and two compare registers.		Pulse output possible (toggle output or PWM/PPG output)	
		8-bit timer/counter unit 3, consisting of one timer register and one compare register		————	
Serial interface		One-channel UART (special baud rate generator incorporated) One-channel CSI (three-wire serial I/O, SBI)			

Note The number of I/O pins includes special-function pins.

Item	μ PD78212	μ PD78214	μ PD78P214	μ PD78213
A/D converter	Eight channels, each having a resolution of eight bits			
Interrupt	<ul style="list-style-type: none"> • 19 interrupts (seven external and 12 internal) plus those caused by BRK instructions • Two programmable priority levels • Two types of interrupt handling, vectored interrupt and macro service 			
Instruction set	<ul style="list-style-type: none"> • 16-bit calculation • Multiplication (8 bits by 8 bits) and division (16 bits by 8 bits) • Bit manipulation • BCD conversion • Others 			
Package	<ul style="list-style-type: none"> • 64-pin plastic shrink DIP (750 mil) for all products • 64-pin plastic QUIP for all products other than μPD78212, μPD78212(A), and μPD78P214(A) • 68-pin plastic QFJ for all products other than μPD78212, μPD78212(A), μPD78213(A), and μPD78P214(A) • 64-pin plastic QFP (14 x 14 mm)Note for all products other than μPD78213(A) • 74-pin plastic QFP (20 x 20 mm) for all products other than μPD78212(A), μPD78213(A), and μPD78P214(A) • 64-pin ceramic shrink DIP with window (750 mil) for μPD78P214 only 			

Note Small package with a lead pitch of 0.8 mm, suitable for cameras.

1.7 DIFFERENCES BETWEEN THE μPD78210^{Note} AND μPD78213

Note For maintenance purposes only.

Item \ Product	μPD78210	μPD78213
RAM capacity	128 bytes	512 bytes
I/O pins	<ul style="list-style-type: none"> • Software programmable pull-up resistors: Not supported • Transistor direct drive outputs: Not supported 	<ul style="list-style-type: none"> • Software programmable pull-up resistors: Supported • Transistor direct drive outputs: Supported
Timer/counter	PWM/PPG output: Not supported	PWM/PPG output: Supported
Serial interface	Scaler for the baud rate generator output: Not supported	Scaler for the baud rate generator output: Supported
Interrupt	Macro service can be applied to some interrupt requests.	Macro service can be applied to all interrupt requests with the exception of that caused by a serial receive error.
A/D converter	<ul style="list-style-type: none"> • Six input pins. • A voltage ranging from 0 V to AV_{REF} can be applied to the input pins. 	<ul style="list-style-type: none"> • Eight input pins. • A voltage ranging from 0 V to AV_{REF} can be applied only to those pins for which A/D conversion is being performed, as well as the pins selected by the ANI0 to ANI3 bits of the ADM register.
Package	64-pin QFP: Not supported	64-pin QFP: Supported
Others	The area at addresses 0FE20H to 0FE7FH can only be accessed in saddr addressing mode.	The area at addresses 0FE20H to 0FE7FH can be accessed in any addressing mode.

1.8 DIFFERENCES BETWEEN THE μ PD78214 SUB-SERIES AND μ PD78218A SUB-SERIES

Series name		μ PD78214 Sub-Series				μ PD78218A Sub-Series			
Product		μ PD78212 μ PD78212(A)	μ PD78213 μ PD78213(A)	μ PD78214 μ PD78214(A)	μ PD78P214 μ PD78P214(A)	μ PD78217A	μ PD78218A	μ PD78P218A	
Minimum instruction cycle (when operating at 12 MHz)		333 ns	500 ns	333 ns		500 ns	333 ns		
Operating voltage range		$V_{DD} = +5\text{ V} \pm 10\%$						$V_{DD} = +5\text{ V} \pm 0.3\text{ V}$	
Internal memory	ROM	8K bytes (masked ROM)	ROM-less	16K bytes (masked ROM)	16K bytes (PROM)	ROM-less	32K bytes (masked ROM)	32K bytes (PROM)	
	RAM	384 bytes	512 bytes			1024 bytes			
Number of I/O pins		54	36	54		36	54		
Execution time (number of clocks) required for PUSH PSW instruction		<ul style="list-style-type: none"> Five or seven when internal dual-port RAM is used for the stack area Seven or nine in all other cases 				<ul style="list-style-type: none"> Six when internal dual-port RAM is used for the stack area Eight in all other cases 			
One-shot output of 16-bit timer/counter		None				Supported			
Bit width of macro-service counter		Eight bits				Eight bits or 16 bits (selectable, except for macro service type A)			
MPD or MPT increment in macro-service type C		Only the eight lower-order bits are incremented. (The eight high-order bits remain as is.)				The 16 bits are incremented.			
Restriction imposed on data transfer from memory to SFR in macro-service type A		Transfer data shall not be D0H to DFH.				Transfer source buffer (memory) addresses shall not be 0FED0H to 0FEDFH.			
Macro-service execution time		The macro-service execution time depends on the macro-service mode and other factors, and also varies with the sub-series. For details, refer to the table of macro-service execution times in the relevant user's manual.							
A/D converter	Restriction imposed on input voltage	A voltage ranging from 0 V to AV_{REF} can be applied only to those pins for which A/D conversion is performed and the pins selected by the ANI0 to ANI3 bits of the ADM register.				A voltage ranging from 0 V to AV_{REF} can be applied only to those pins for which A/D conversion is being performed.			
	Restriction imposed on the AV_{REF} voltage	3.4 V to V_{DD}				3.6 V to V_{DD}			
Oscillation settling time when STOP mode is released		Pulse width of NMI at its active level, plus 16 counts on the corresponding counter				15 counts on the corresponding counter or pulse width of NMI at its active level, plus 16 counts on the corresponding counter			
Package		<ul style="list-style-type: none"> 64-pin plastic shrink DIP (750 mil) for all products 64-pin plastic QUIP for all products other than μPD78212, μPD78212(A), and μPD78P214(A) 68-pin plastic QFJ for all products other than μPD78212, μPD78212(A), μPD78213(A), and μPD78P214(A) 64-pin plastic QFP (14 × 14 mm) for all products other than μPD78213(A) 74-pin plastic QFP (20 × 20 mm) for all products other than μPD78212(A), μPD78213(A), and μPD78P214(A) 64-pin ceramic shrink DIP with window (750 mil) for μPD78P214 only 				<ul style="list-style-type: none"> 64-pin plastic shrink DIP (750 mil) for all products 64-pin plastic QFP (14 × 14 mm) for all products 64-pin ceramic shrink DIP with window (750 mil) for the μPD78P218A only 			

1.9 DIFFERENCES BETWEEN THE μPD78212 AND μPD78212(A)

Item \ Product	μPD78212	μPD78212(A)
Quality grade	Standard	Special
Package	<ul style="list-style-type: none"> • 64-pin plastic shrink DIP • 64-pin plastic QFP • 74-pin plastic QFP 	<ul style="list-style-type: none"> • 64-pin plastic shrink DIP • 64-pin plastic QFP

1.10 DIFFERENCES BETWEEN THE μPD78213 AND μPD78214, AND THE μPD78213(A) AND μPD78214(A)

Item \ Product	μPD78213, μPD78214	μPD78213(A), μPD78214(A)
Quality grade	Standard	Special
Maximum period in which 74-pin plastic QFPs can be soldered satisfactorily, after their sealed packaging has been opened	No limit	Seven days
Package	<ul style="list-style-type: none"> • 64-pin plastic shrink DIP • 64-pin plastic QFP^{Note} • 74-pin plastic QFP^{Note} • 64-pin plastic QUIP • 68-pin plastic QFJ^{Note} 	

Note Not supported for the μPD78213(A)

1.11 DIFFERENCES BETWEEN THE μPD78P214 AND μPD78P214(A)

Item \ Product	μPD78P214	μPD78P214(A)
Quality grade	Standard	Special
Package	<ul style="list-style-type: none"> • 64-pin plastic shrink DIP • 64-pin ceramic shrink DIP with window • 64-pin plastic QFP • 74-pin plastic QFP • 64-pin plastic QUIP • 68-pin plastic QFJ 	<ul style="list-style-type: none"> • 64-pin plastic shrink DIP • 64-pin plastic QFP

1.12 DIFFERENCES BETWEEN THE μ PD78212, μ PD78213, μ PD78214, AND μ PD78P214

1.12.1 Functional Differences

Product name Parameter	μ PD78212	μ PD78213	μ PD78214	μ PD78P214
Internal ROM	8KB masked ROM at 00000H to 01FFFFH	None	16KB masked ROM at 00000H to 03FFFFH	16KB PROM at 00000H to 03FFFFH
Internal RAM	384 bytes at 0FD80H to 0FEFFH	512 bytes at 0FD00H to 0FEFFH		
Port 4	Used as both general-purpose I/O port (P40 to P47) and address/data bus (AD0 to AD7)	Used only as address/data bus (AD0 to AD7)	Used as both general-purpose I/O port (P40 to P47) and address/data bus (AD0 to AD7)	
Port 5	Used as both general-purpose I/O port (P50 to P57) and address bus (A8 to A15)	Used only as address bus (A8 to A15)	Used as both general-purpose I/O port (P50 to P57) and address bus (A8 to A15)	
Port 6	P64 and P65 are used as both general-purpose I/O ports, and the \overline{RD} and \overline{WR} pins, respectively.	P64 and P65 are used only as the \overline{RD} and \overline{WR} pins, respectively.	P64 and P65 are used as both general-purpose I/O ports, and the \overline{RD} and \overline{WR} pins, respectively.	
Others	—	—	—	PROM programming mode is supported

1.12.2 Package Differences

These products are available with either of two types of QFP (64-pin and 74-pin). Taking differences such as their soldering conditions, listed below, into consideration, select an appropriate QFP.

Item	64-pin QFP	74-pin QFP	
Package size	14 × 14 mm, lead pitch: 0.8 mm	20 × 20 mm, lead pitch: 1.0 mm	
Soldering conditions	Infrared reflow soldering or vapor-phase soldering (VPS)	Supported by μ PD78212, μ 78213 ^{Note} , μ 78214, and μ 78P214	Supported by μ PD78212 ^{Note} , μ 78213 ^{Note} , μ 78214, and μ 78P214 ^{Note}
	Humidity control for infrared reflow soldering or VPS	The μ PD78212, μ PD78213 ^{Note} , μ PD78214, and μ PD78P214 shall be soldered within two days of their sealed package being opened.	The μ PD78P214 ^{Note} shall be soldered within seven days of its sealed package being opened. The μ PD78212 ^{Note} , μ PD78213 ^{Note} , and μ PD78214 can be soldered at any time after their sealed package has been opened.

Note The corresponding package is not available for the special quality grade versions of these products, indicated by suffix (A).

CHAPTER 2 PIN FUNCTIONS

2.1 PIN FUNCTION LIST

2.1.1 Normal Operating mode

(1) Ports

Pin	Input/Output	Pin name for secondary function	Function
P00-P07	Output	—	Port 0 (P0): Can be used as two four-bit, real-time output ports. Can drive transistors directly.
P20	Input	NMI	Port 2 (P2): Cannot be used as a general-purpose port (non-maskable interrupt). The input level can be checked as part of the interrupt routine. Pins P22 to P27 can be collectively connected to internal pull-up resistors by means of software.
P21		INTP0	
P22		INTP1	
P23		INTP2/CI	
P24		INTP3	
P25		INTP4/ASCK	
P26		INTP5	
P27		SI	
P30	Input/Output	RxD	Port 3 (P3): Each pin can be assigned to be either an input or output pin. Input pins can be collectively connected to internal pull-up resistors by means of software.
P31		TxD	
P32		\overline{SCK}	
P33		SO/SB0	
P34-P37		TO0-TO3	
P40-P47 ^{Note 1}	Input/Output	AD0-AD7	Port 4 (P4): Can be simultaneously assigned to be either input or output pins. Can be collectively connected to internal pull-up resistors by means of software. Can directly drive LEDs.
P50-P57 ^{Note 1}	Input/Output	A8-A15	Port 5 (P5): Each pin can be assigned to be either an input or output pin. Input pins can be collectively connected to internal pull-up resistors by means of software. Can directly drive LEDs.
P60-P63	Output	A16-A19	Port 6 (P6): Each of pins P64 to P67 can be assigned to be either an input or output pin. Input pins P64 to P67 can be collectively connected to internal pull-up resistors by means of software.
P64 ^{Note 2}	Input/Output	\overline{RD}	
P65 ^{Note 2}		\overline{WR}	
P66		$\overline{WAIT}/AN6$	
P67		$\overline{REFRQ}/AN7$	
P70-P75	Input	AN0-AN5	Port 7 (P7)

Notes 1. In the case of the μ PD78213, these pins do not function as ports.

2. In the case of the μ PD78213, neither P64 nor P65 functions as a port.

(2) Pins other than those which function as ports

Pin	Input/output	Function	Pin name for secondary function
TO0-TO3	Output	Timer output	P34-P37
CI	Input	Count clock supplied to eight-bit timer/counter unit 2	P23/INTP2
RxD	Input	Serial data input (UART)	P30
TxD	Output	Serial data output (UART)	P31
ASCK	Input	Baud rate clock input (UART)	P25/INTP4
SB0	Input/output	Serial data input/output (SB2)	P33/SO
SI	Input	Serial data input (three-wire serial I/O)	P27
SO	Output	Serial data output (three-wire serial I/O)	P33/SB0
$\overline{\text{SCK}}$	Input/output	Serial clock input/output (SBI, three-wire serial I/O)	P32
NMI	Input	External interrupt requests	P20
INTP0			P21
INTP1			P22
INTP2			P23/CI
INTP3			P24
INTP4			P25/ASCK
INTP5			P26
AD0-AD7	Input/output	Time-multiplexed address/data bus (external memory connected)	P40-P47Note
A8-A15	Output	High-order address bus (external memory connected)	P50-P57Note
A16-A19	Output	High-order address when the addresses are expanded (external memory connected)	P60-P63
$\overline{\text{RD}}$	Output	Read strobe to external memory	P64Note
$\overline{\text{WR}}$	Output	Write strobe to external memory	P65Note
$\overline{\text{WAIT}}$	Input	Wait insertion	P66/AN6
ASTB	Output	Latch timing for addresses A0 to A7 (when external memory is accessed)	—
$\overline{\text{REFRQ}}$	Output	Refresh pulse to external pseudo-static memory	P67/AN7
$\overline{\text{RESET}}$	Input	Chip reset	—
X1	Input	Connected to the crystal oscillator used for the system clock (a clock signal can be input to X1)	—
X2	—		
$\overline{\text{EA}}$	Input	Designating ROM-less operation (access to the external memory mapped to the same area as the internal ROM). Set this pin to low for the μPD78213, or to high for the μPD78212 and μPD78214.	—
AN0-AN5	Input	Analog voltage input for A/D converter	P70-P75
AN6, AN7			P66/ $\overline{\text{WAIT}}$, P67/ $\overline{\text{REFRQ}}$
AV_{REF}	—	A/D converter reference voltage	—
AV_{SS}		A/D converter ground	
V_{DD}		Main power	
V_{SS}		Ground	
NC		—	

Note These pins do not function as ports in the case of the μPD78213.

2.1.2 PROM Programming Mode (only for the μ PD78P214, P20/NMI = 12.5 V, $\overline{\text{RESET}} = \text{L}$)

Pin	Input/output	Function
P20/NMI	Input	Setting PROM programming mode
$\overline{\text{RESET}}$		
A0-A14		
D0-D7	Input/output	Data bus
$\overline{\text{CE}}$	Input	PROM enable input
$\overline{\text{OE}}$		Read strobe to PROM
V_{PP}	—	Power for programming
V_{DD}		Main power
V_{SS}		Ground
NC		—

2.2 PIN FUNCTIONS

2.2.1 Normal Operating mode

(1) P00 to P07 (Port 0): Tristate outputs

Port 0, an eight-bit output port with output latches, can directly drive transistors. Its eight bits can be simultaneously set to either output mode or high impedance mode by specifying the port-0 mode register (PM0) accordingly.

Port 0 can output the contents of the buffer registers (P0L, P0H) in real-time at any interval, in units of eight or four bits (P00 to P03 and P04 to P07). The real-time output port control register (RTPC) determines whether port 0 is used as a normal output port or real-time output port.

When the $\overline{\text{RESET}}$ signal is input, the output of port 0 becomes high impedance, resulting in the contents of the output latches becoming undefined.

(2) P20 to P27 (port 2): Inputs

Port 2 is an eight-bit input port. P22 to P27 are provided with software-programmable pull-up resistors. P20 to P27 also act as input pins for control signals such as external interrupts (see **Table 2-1**). To prevent malfunctions caused by noise, port 2 provides Schmitt-triggered input circuits.

Table 2-1 Port 2 Functions

Port	Function
P20	Input port/NMI input ^{Note}
P21	Input port/INTP0 input/CR11 capture trigger input/trigger signal for real-time output port
P22	Input port/INTP1 input/CR22 capture trigger input
P23	Input port/INTP2 input/CI input
P24	Input port/INTP3 input/CR02 capture trigger input
P25	Input port/INTP4 input/ASCK input
P26	Input port/INTP5 input/external trigger signal for A/D converter
P27	Input port/SI input

Note An NMI input is received regardless of whether interrupts are enabled or disabled.

(a) When functioning as a port

Signals applied to these pins can be read and these pins can be tested, regardless of whether these pins are acting as secondary function pins.

(b) When functioning as control-signal input pins

(i) NMI (non-maskable interrupt)

Apply an external non-maskable interrupt request signal to this pin. The external interrupt mode register (IMTM0) specifies whether an interrupt is detected at its rising or falling edge.

(ii) INTP0 to INTP5 (interrupts from peripherals)

Apply external interrupt request signals to these pins. When an interrupt request signal is detected at the edge specified by the external interrupt mode registers (INTM0 and INTM1), at any of the INTP0 to INTP5 pins, an interrupt is generated. (For details, see **Chapter 11**.)

The INTP0 to INTP3 and INTP5 pins can also be used as external trigger input pins for a range of functions, as described below.

- INTP0: Capture trigger input for eight-bit timer/counter unit 1, and trigger input for real-time output port
- INTP1: Capture trigger input for eight-bit timer/counter unit 2
- INTP2: External count clock input for eight-bit timer/counter unit 2
- INTP3: Capture trigger input for 16-bit timer/counter
- INTP5: External trigger input for A/D converter

(iii) CI (clock input)

External clock input for eight-bit timer/counter unit 2

(iv) ASCK (asynchronous serial clock)

External baud-rate clock input.

(v) SI (serial input)

Serial data input (when three-wire serial input mode is used)

(3) P30-P37 (port 3): Tristate inputs/outputs

Port 3, an eight-bit input and output port with output latches, is provided with software-programmable pull-up resistors. Each pin can be used as an input or output pin by specifying the port-3 mode register (PM3).

It also acts as a control signal pin.

It can be used in either of the following two operating modes, as listed in Table 2-2, by specifying the port-3 mode control register (PMC3). The signals applied to these pins can be read and these pins can be tested regardless of whether these pins are acting as secondary function pins.

When the $\overline{\text{RESET}}$ signal is input, the output of port 3 becomes high impedance, such that it functions as an input port, resulting in the contents of the output latches becoming undefined.

Table 2-2 Port 3 Operating Mode (n = 0 to 7)

Mode	Port mode	Control signal I/O mode
PMC3 setting	PMC3n = 0	PMC3n = 1
P30	I/O port	RxD input
P31		TxD output
P32		$\overline{\text{SCK}}$ input/output
P33		SO output/SB0 input/output
P34		TO0 output
P35		TO1 output
P36		TO2 output
P37		TO3 output

(a) Port mode

Pins for which port mode is specified by the PMC3 can be used independently as input or output pins by specifying the port-3 mode register (PM3).

(b) Control signal I/O mode

Each pin can be used as a control signal pin by specifying the PMC3 register.

(i) RxD (receive data)

Serial data input for asynchronous serial interface

(ii) TxD (transmit data)

Serial data output for asynchronous serial interface

(iii) $\overline{\text{SCK}}$ (serial clock)

Serial clock input or output for synchronous serial interface

(iv) SO (serial output)/SBO (serial bus)

Serial data output (when three-wire serial I/O mode is used), or serial bus input or output in SBI mode

(v) TO0 to TO3 (timer output)

Timer outputs

(4) P40-P47 (port 4): Tristate inputs/outputs

Port 4, an eight-bit I/O port with output latches, is provided with software-programmable pull-up resistors and can directly drive LEDs. Its pins can be collectively used as input or output pins by specifying the memory expansion mode register (MM).

It acts as a time-multiplexed address/data bus (AD0 to AD7) when external memory or I/O devices are expanded.

In the case of the $\mu\text{PD78213}$, it acts as a time-multiplexed address/data bus (AD0 to AD7).

When the $\overline{\text{RESET}}$ signal is input, the output of port 4 becomes high impedance, such that it functions as an input port, resulting in the contents of the output latches becoming undefined.

(5) P50 to P57 (port 5): Tristate inputs/outputs

Port 5, an eight-bit I/O port with output latches, is provided with software-programmable pull-up resistors and can directly drive LEDs. Its pins can be used independently as input or output pins by specifying the port-5 mode register (PM5) accordingly.

It acts as an address bus (A8 to A15) when external memory or I/O devices are expanded.

In the case of the $\mu\text{PD78213}$, it functions as an address bus (A8 to A15).

When the $\overline{\text{RESET}}$ signal is input, the output of port 5 becomes high impedance, such that it functions as an input port, resulting in the contents of the output latches becoming undefined.

(6) P60 to P67 (port 6): Output (P60 to P63) and tristate inputs/outputs (P64 to P67)

Port 6 is an eight-bit I/O port with output latches. Pins P64 to P67 are provided with software-programmable pull-up resistors.

The pins of port 6 also function as control signal input pins, as listed in Table 2-3. To use these pins as control signal input pins, set up is required.

In the case of the μPD78213, P64 and P65 act as an \overline{RD} output and \overline{WR} output, respectively.

When the \overline{RESET} signal is applied, P60 to P63 go low and P64 to P67 function as an input port (the outputs become high impedance). The contents of the output latches become undefined at the four high-order bits and 0H at the four low-order bits.

Table 2-3 Port 6 Operating Mode

Pin	Port mode	Control signal I/O mode	Operation required to assign pins as control signal pins
P60-P63	Output port	A16-A19 output	Set the MM6 bit of the MM register to 1.
P64	Input/output port	\overline{RD} output	For the μPD78213, specify external memory expansion mode using bits MM2 to MM0 of the MM register.
P65		\overline{WR} output	
P66		\overline{WAIT} input/AN6 input	Set the PW register, or bits PWN1 and PWN0 (n = 2 and 3) of the MM register and P66 to input mode.
P67		\overline{REFRQ} output/AN7 input	Set the RFEN bit of the RFM register to 1.

Caution While the \overline{RESET} signal is being applied, P60 to P63 is high impedance. When the \overline{RESET} signal is released, the output of these pins is low level. Design the peripheral circuit so that it operates normally when pins P60 to P63 initially output low level.

Remark For details, see Chapter 13.

(a) Port mode

P60 to P63 are an output port. Each of pins P64 to P67 can be used for input or output by specifying the port-6 mode register (PM6) accordingly.

(b) Control signal I/O mode

(i) A16 to A19 (address bus)

High-order address bus output when the external memory area is expanded (10000H to FFFFFH). The memory expansion register (MM) controls these pins.

(ii) \overline{RD} (read strobe)

Strobe signal output used for reading the external memory. In the case of the μPD78213, the MM register controls this pin.

(iii) \overline{WR} (write strobe)

Strobe signal output used for writing to the external memory. In the case of the μPD78213, the MM register controls this pin.

(iv) \overline{WAIT} (wait)

Wait signal input. The programmable wait control (PW) register or MM register controls this pin.

(v) \overline{REFRQ} (refresh request)

Used for outputting refresh pulses to the external pseudo-static memory. The refresh mode register (RFM) controls this pin.

(vi) AN6 and AN7 (analog input)

Analog inputs to the A/D converter.

(7) P70 to P75 (port 7): Input

Port 7 is a six-bit input port. Its pins also function as analog input pins (AN0 to AN5) for the A/D converter. Signals applied to these pins can be read and these pins can be tested regardless of whether these pins are acting as secondary function pins.

(8) ASTB (address strobe): Output

Timing signal output used for latching addresses externally to enable access to external memory.

(9) \overline{EA} (external access): Input

Control signal input used for switching the program memory from the internal ROM to the external memory. When this signal is high, the internal ROM is accessed. When low, the external memory is accessed in ROM-less mode. In the case of the μ PD78212, μ PD78214, and μ PD78P214, always apply a high-level signal to this pin. In the case of the μ PD78213, apply a low-level signal.

(10) X1 and X2 (crystal)

Pins for connecting the crystal used for internal clock oscillation. When an external clock signal is supplied, apply it to pin X1. Also, apply the signal having the inverted phase of this clock signal to pin X2.

(11) \overline{RESET} (reset): Input

Low-active reset input.

(12) AV_{REF}

Input of the reference voltage and power for the A/D converter.

(13) AV_{SS}

A/D converter ground.

(14) V_{DD}

Main power input. Connect all V_{DD} pins to the power.

(15) V_{SS}

Ground. Connect all V_{SS} pins to the ground.

(16) NC (non-connection)

Not connected inside the chip.

2.2.2 PROM Programming Mode (for the μ PD78P214)**(1) P20/NMI: Input**

Input used for setting the μ PD78P214 to PROM programming mode. When a voltage of 12.5 V is applied to this pin and the \overline{RESET} pin goes low, the μ PD78P214 enters PROM programming mode.

(2) \overline{RESET} : Input

Input used for setting the μ PD78P214 to PROM programming mode. When a low-level signal is applied to this pin and a voltage of 12.5 V is applied to the P20/NMI pin, the μ PD78P214 enters PROM programming mode.

(3) A0 to A14 (address bus): Inputs

Address bus used for the internal PROM (0000H to 3FFFFH). Apply a low-level signal to pin A14.

(4) D0 to D7 (data bus): Inputs/outputs

Data bus, through which programs are read or written to and from the internal PROM.

(5) \overline{CE} (chip enable): Input

Enable signal input for the internal PROM. When this signal is active, programs can be either written or read.

(6) \overline{OE} (output enable): Input

Read strobe signal input for the internal PROM. When this input goes active while the \overline{CE} is low, one byte of the program at the address specified by pins A0 to A14 in the internal PROM is read through pins D0 to D7.

(7) V_{PP} (programming power supply)

Power supply for writing programs. When the \overline{CE} goes low while the V_{PP} is 12.5 V and the \overline{OE} is high, the data on pins D0 to D7 is written into the internal PROM at the address specified by pins A0 to A14.

(8) V_{DD}

Main power input.

(9) V_{SS}

Ground.

(10) NC (non-connection)

Not connected inside the chip.

2.3 I/O CIRCUITS AND UNUSED-PIN HANDLING

Table 2-4 lists the types of I/O circuits provided for each pin and describes how pins are handled when not used. Fig. 2-1 illustrates the I/O circuit types.

Table 2-4 Types of I/O Circuits and Unused-Pin Handling

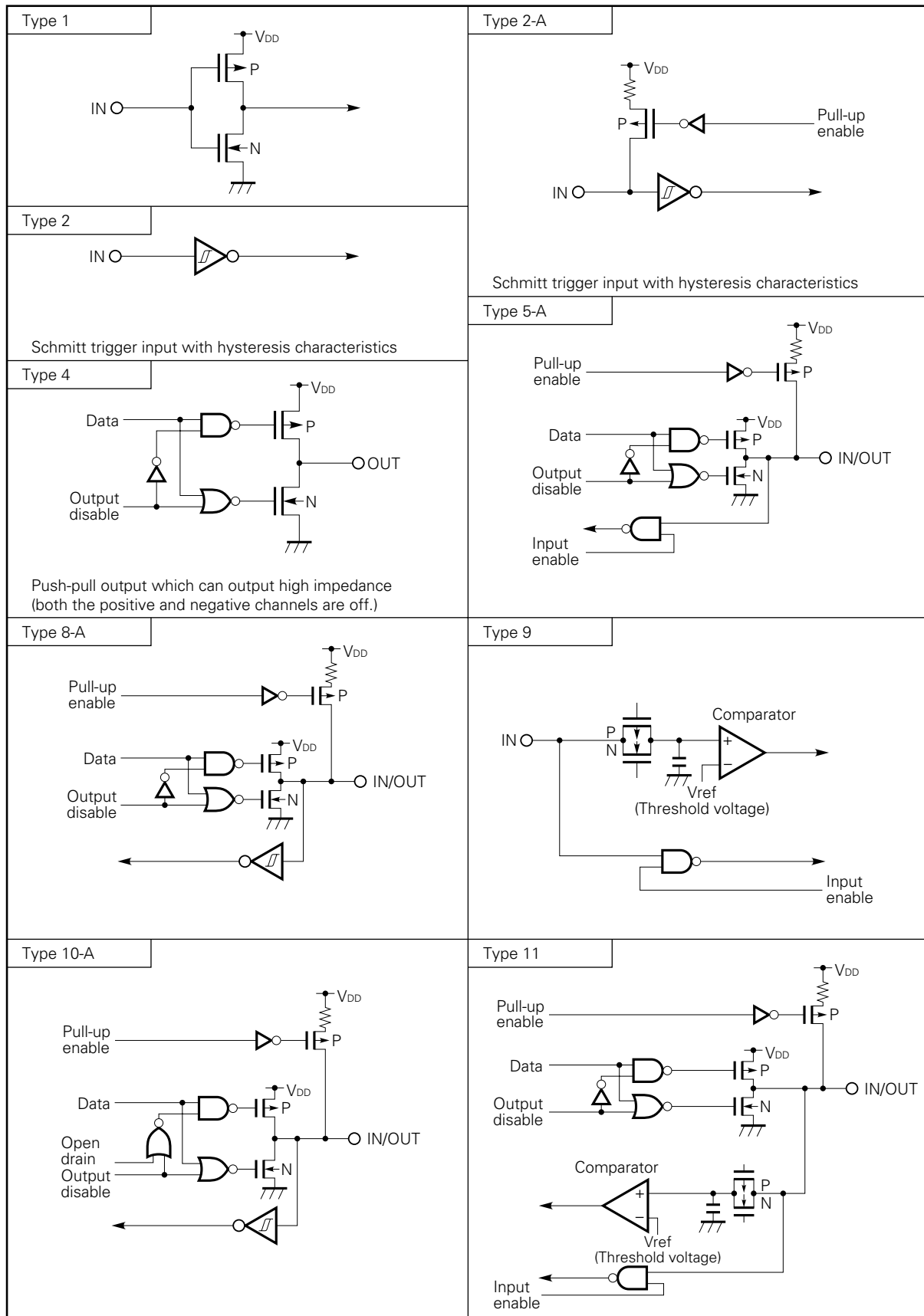
Pin	Type of I/O circuit	Input/output	Recommended unused-pin handling	
P00-P07	4	Output	Leave open.	
P20/NMI	2	Input	Connect to V_{DD} or V_{SS} .	
P21/INTP0				
P22/INTP1	2-A	Input	Connect to V_{DD} .	
P23/INTP2/CI				
P24/INTP3				
P25/INTP4/ASCK				
P26/INTP5				
P27/SI				
P30/RxD	5-A	Input/output	Connect to V_{DD} when used as an input pin. Leave open when used as an output pin.	
P31/TxD				
P32/ \overline{SCK}	8-A	Input/output	Connect to V_{DD} when used as an input pin. Leave open when used as an output pin.	
P33/SB0/SO	10-A			
P34/TO0-P37/TO3	5-A			
P40/AD0-P47/AD7				
P50/A8-P57/A15				
P60/A16-P63/A19				4
P64/ \overline{RD}	5-A	Input/output	Connect to V_{DD} when used as an input pin. Leave open when used as an output pin.	
P65/ \overline{WR}				
P66/ \overline{WAIT} /AN6	11	Input/output	Connect to V_{DD} ^{Note} when used as an input pin. Leave open when used as an output pin.	
P67/ \overline{REFRQ} /AN7				
P70/AN0-P75/AN5	9	Input	Connect to V_{SS} .	
ASTB	4	Output	Leave open.	
\overline{RESET}	2	Input	—	
\overline{EA}	1			
AV_{REF}	—			Connect to V_{DD} or V_{SS} ^{Note} .
AV_{SS}				Connect to V_{SS} .

Note See Section 8.6.

Remark Since the type numbers of I/O circuits are numbered in the 78K series, they may not be serial in a certain product. (A product may not contain some of these I/O circuits.)

Caution When an I/O pin is used as both an input and output pin, connect the pin to the V_{DD} pin through a resistor of less than 100 kilohms. (Especially, when the RESET pin goes to a voltage higher than the low level upon power on, or when an I/O pin is switched with software.)

Fig. 2-1 I/O Circuits Provided for Pins



2.4 NOTES

- (1) While the $\overline{\text{RESET}}$ signal is being applied, pins P60 to P63 are high impedance. When the $\overline{\text{RESET}}$ signal is released, the output of these pins is low level. Design the peripheral circuit so that it operates satisfactorily when pins P60 to P63 initially output the low level.
- (2) When an I/O pin is used as both an input and output pin, connect the pin to the V_{DD} pin through a resistor of less than 100 kilohms. (Especially, when the $\overline{\text{RESET}}$ pin goes to a voltage higher than the low level upon power on, or when an I/O pin is switched with software.)

CHAPTER 3 CPU FUNCTION

3.1 MEMORY SPACE

The μ PD78214 can access a memory space of up to 1M byte. Figs. 3-1 to 3-4 show the corresponding memory maps. The mapping of program memory depends on the status of the \overline{EA} pin. The \overline{EA} pin of the μ PD78213 must be tied low.

(1) μ PD78212

Program memory is mapped to the internal ROM (8K bytes: 00000H to 01FFFH) and external memory (56704 bytes: 02000H to 0FD7FH). External memory is accessed in external memory expansion mode. The area mapped as external memory can also be used as data memory.

Data memory is mapped to internal RAM (384 bytes: 0FD80H to 0FEFFH). In 1M-byte expansion mode, external memory (960K bytes: 10000H to FFFFFH) is mapped as expanded data memory.

(2) μ PD78213

Program memory is mapped to external memory (64768 bytes: 00000H to 0FCFFH). This area can also be used as data memory.

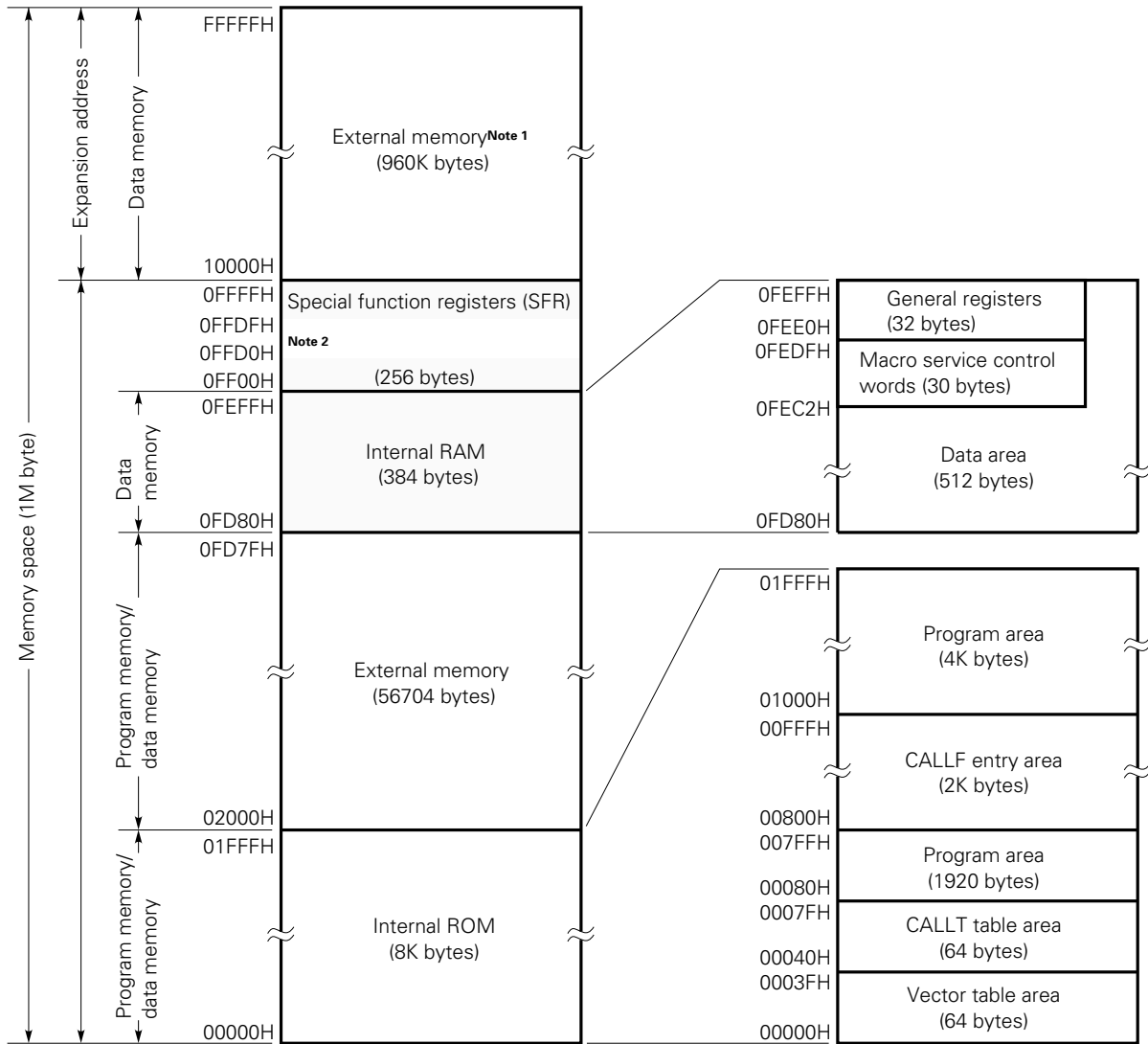
Data memory is mapped to internal RAM (512 bytes: 0FD00H to 0FEFFH). In 1M-byte expansion mode, external memory (960K bytes: 10000H to FFFFFH) is mapped as expanded data memory.

(3) μ PD78214, μ PD78P214

Program memory is mapped to internal ROM (16K bytes: 00000H to 03FFFH) and external memory (48384 bytes: 04000H to 0FCFFH). External memory is accessed in external memory expansion mode. The area mapped as external memory can also be used as data memory.

Data memory is mapped to internal RAM (512 bytes: 0FD00H to 0FEFFH). In 1M-byte expansion mode, external memory (960K bytes: 10000H to FFFFFH) is mapped as expanded data memory.

Fig. 3-1 Memory Map of μPD78212 (\overline{EA} Pin Driven High)

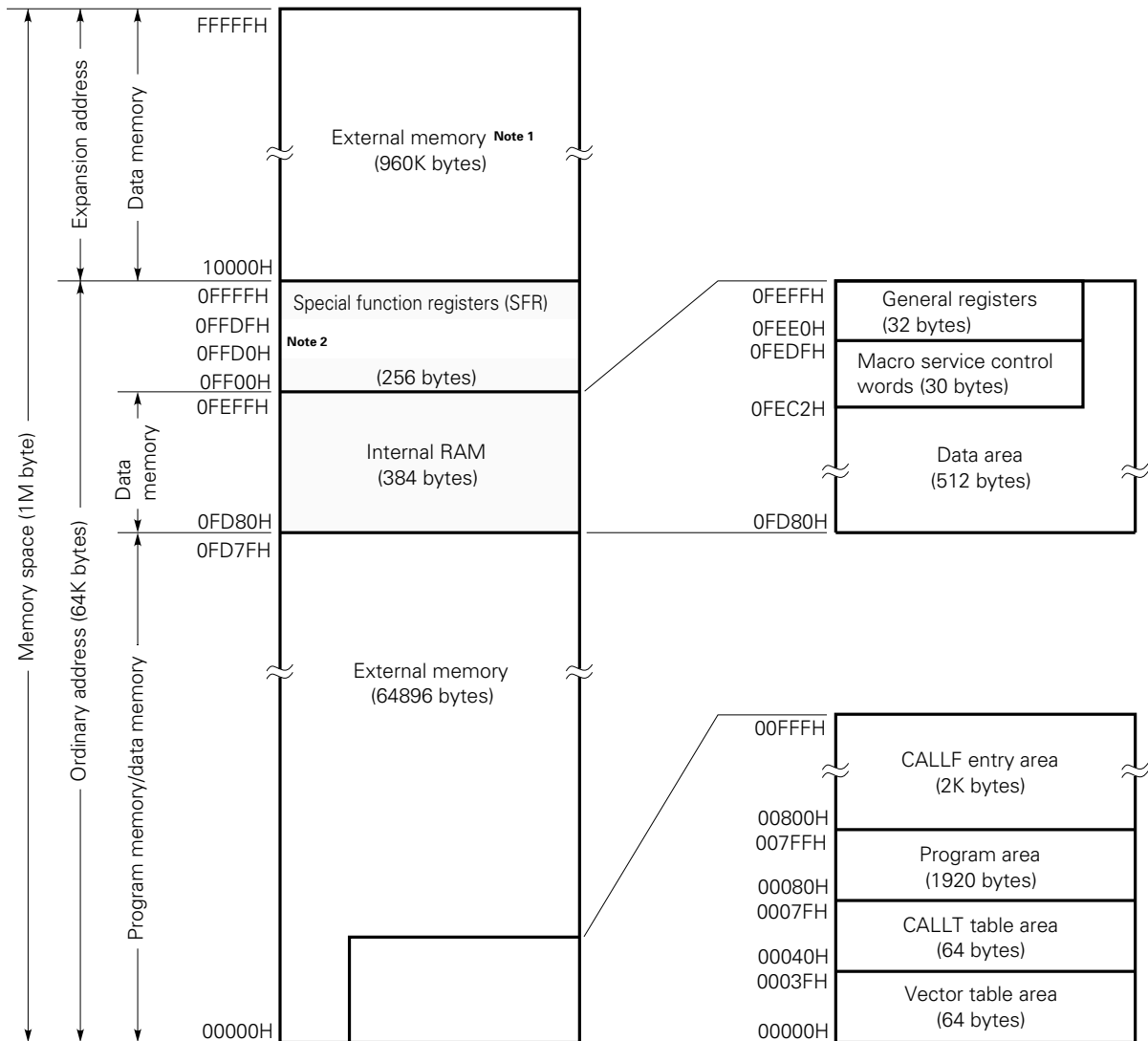


Notes 1. Accessed in 1M-byte expansion mode.

2. External SFR area

Remark The shaded areas indicate internal memory.

Fig. 3-2 Memory Map of μ PD78212 (\overline{EA} Pin Driven Low)

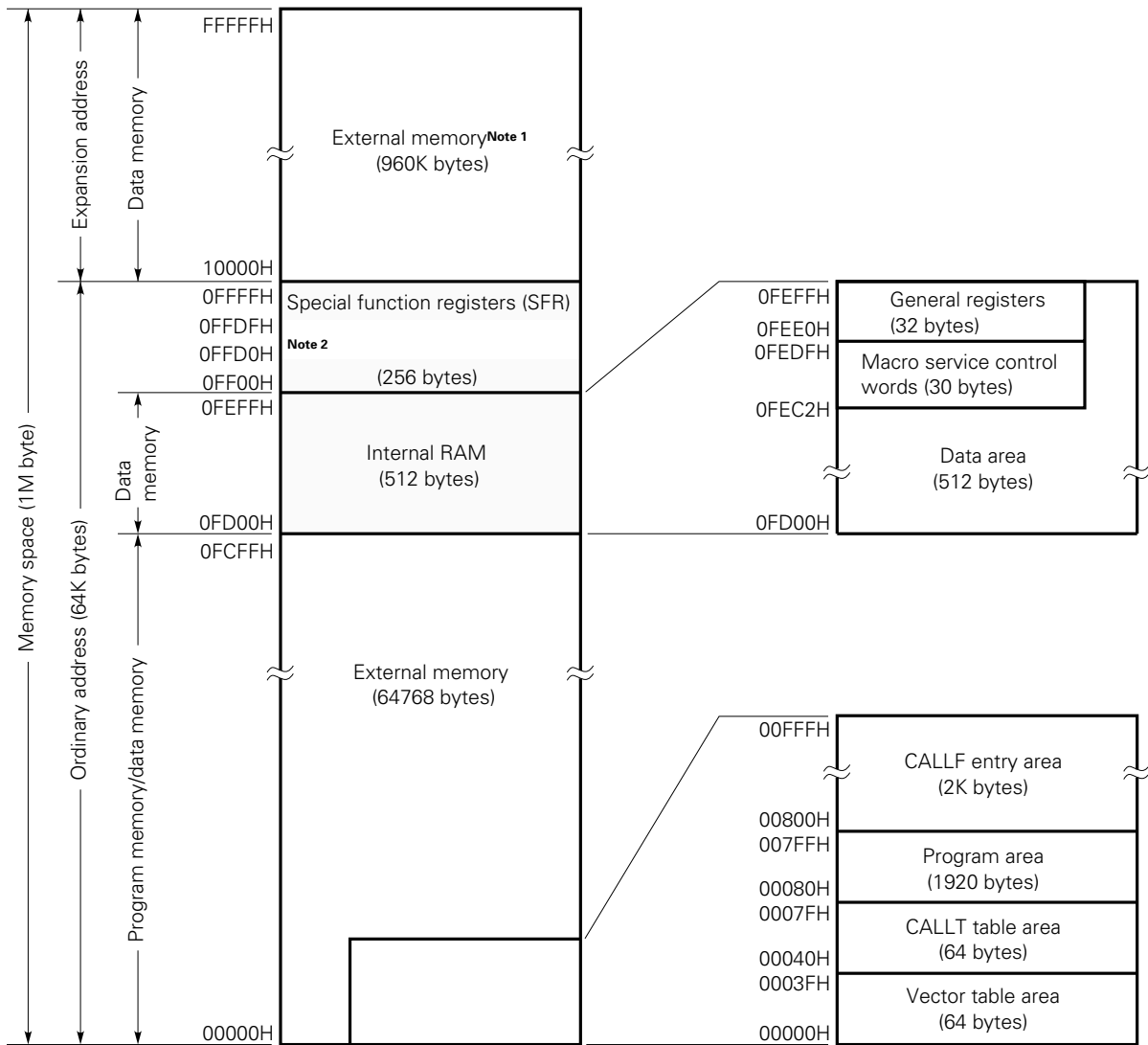


Notes 1. Accessed in 1M-byte expansion mode.

2. External SFR area

Remark The shaded areas indicate internal memory.

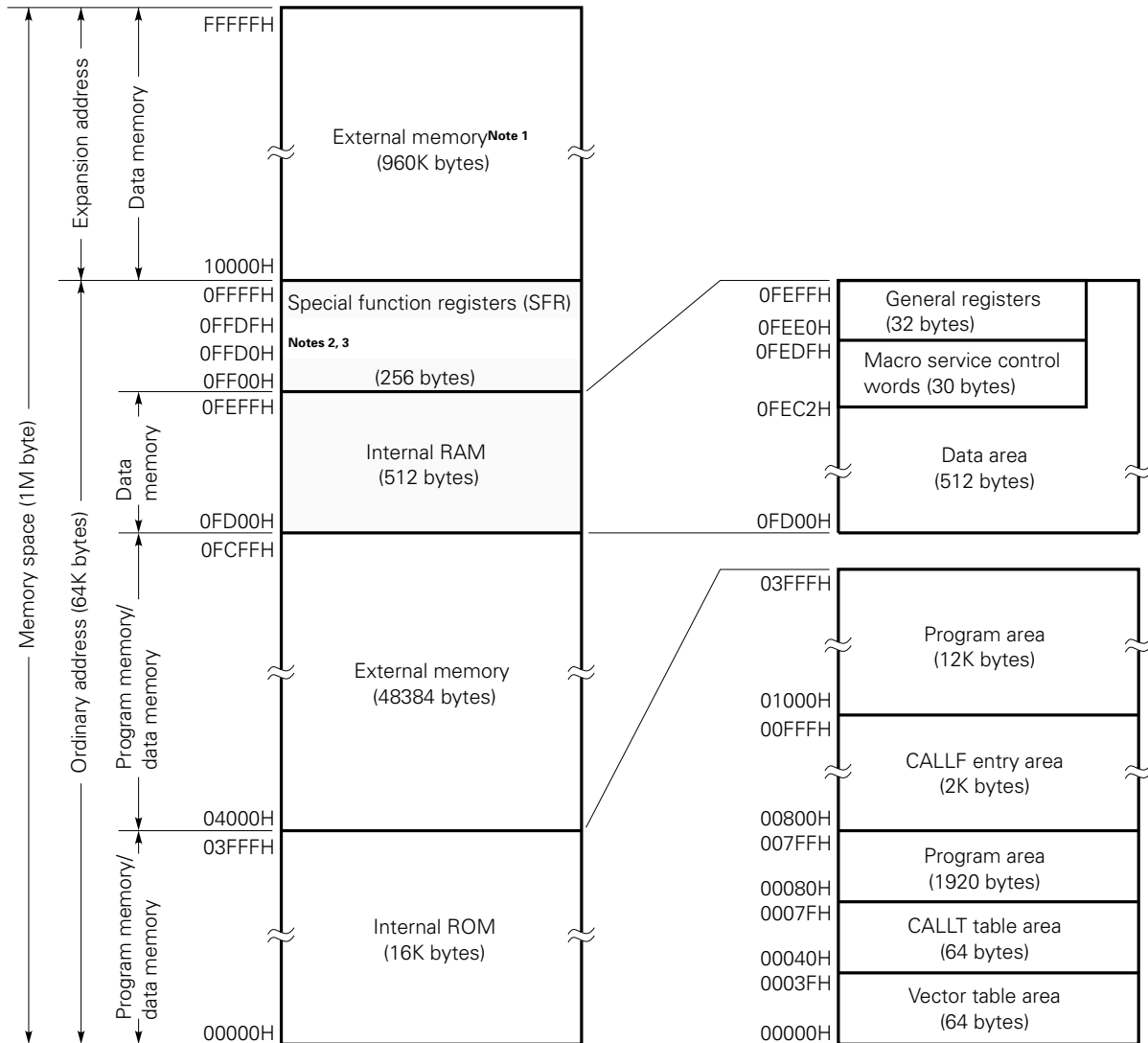
Fig. 3-3 Memory Map of μPD78213, μPD78214, or μPD78P214 (\overline{EA} Pin Driven Low)



Notes 1. Accessed in 1M-byte expansion mode.

2. External SFR area

Remark The shaded areas indicate internal memory.

Fig. 3-4 Memory Map of μ PD78214, μ PD78P214 (\overline{EA} Pin Driven High)

- Notes**
1. Accessed in 1M-byte expansion mode.
 2. Accessed in external memory expansion mode
 3. External SFR area

Remark The shaded areas indicate internal memory.

3.1.1 Internal Program Memory Area

In the area from 00000H to 03FFFH (00000H to 01FFFH for the μPD78212), a 16K × 8 bit ROM (8K × 8 bit ROM for the μPD78212) is incorporated. Programs and table data are stored in this area. Usually, the program counter (PC) is used for addressing.

If the μPD78213 is used or if the \overline{EA} pin is driven low, this area becomes external memory (ROM-less operation).

(1) Vector table area

The 64-byte area from 00000H to 0003FH is reserved as a vector table area. Stored in the vector table area is the program start address to which a program is branched if RESET is input or if an interrupt request occurs. The low-order eight bits of the 16-bit address are stored at even-numbered addresses, and the high-order eight bits at odd-numbered addresses.

Table 3-1 Vector Table

Vector table address	Interrupt request
00000H	Reset (\overline{RESET} input)
00002H	NMI
00006H	INTP0
00008H	INTP1
0000AH	INTP2
0000CH	INTP3
0000EH	INTP4/INTC30
00010H	INTP5/INTAD
00012H	INTC20
00014H	INTC00
00016H	INTC01
00018H	INTC10
0001AH	INTC11
0001CH	INTC21
00020H	INTSER
00022H	INTSR
00024H	INTST
00026H	INTCSI
0003EH	BRK

(2) CALLT instruction table area

In the 64-byte area from 00040H to 0007FH, the subroutine entry address of a one-byte call instruction (CALLT) can be stored.

(3) CALLF instruction entry area

A two-byte call instruction (CALLF) can directly call a subroutine, starting from an address between 00800H and 00FFFH.

3.1.2 Internal RAM Area

A 512-byte (384-byte for the μ PD78212) general-purpose static RAM is incorporated into the area from 0FD00H to 0FEFFH.

This area consists of the following two RAMs:

- Peripheral RAM (PRAM) : 0FD00H to 0FDFFH (0FD80H to 0FDFFH for the μ PD78212)
- Internal dual-port RAM (IRAM): 0FE00H to 0FEFFH

The internal dual-port RAM (IRAM) can be accessed at high speed.

Short direct addressing mode for high-speed access can be used for the area from 0FE20H to 0FEFFH. (Refer to **Chapter 6, Instruction, "78K/II Series User's Manual."**)

General-purpose registers of four banks are mapped into the 32-byte area from 0FEE0H to 0FEFFH. Macro service control words are mapped into the 30-byte area from 0FEC2H to 0FEDFH.

Caution Program fetch from the internal RAM area is prohibited.

Remark It is convenient to store data, work areas, and status flags that are frequently accessed in the area from 0FE20H to 0FEC1H. The area from 0FE00H to 0FE1FH can be accessed at high speed. If this area is used as a stack area, macro service channel, or a data transfer area for a macro service, system throughput can be improved. (For this area, short direct addressing cannot be used. This area is addressed in the same way as other memory spaces. The area, however, can be accessed faster than other memory spaces. In terms of the efficiency of the entire system, it is advantageous to use the area as a stack area, macro service channel, or a data transfer area for a macro service channel.)

3.1.3 Special Function Register (SFR) Area

Special function registers (SFR), which are on-chip hardware peripherals, are mapped into the area from 0FF00H to 0FFFFH (see **Section 3.2.5**).

The area from 0FFD0H to 0FFDFH is mapped as an external SFR area. This area allows the μ PD78214 in external memory expansion mode (selected by memory expansion mode register MM) and μ PD78213 (ROM-less) to access external peripheral I/O.

Caution Never access an address to which no SFR is mapped in this area. If this is attempted, the μ PD78214 may enter a deadlock. To restore the device from the deadlock, a reset signal must be input.

3.1.4 External SFR Area

In the SFR area, the 16-byte area from 0FFD0H to 0FFDFH is mapped as an external SFR area. This area allows the μ PD78214 in external memory expansion mode (selected by memory expansion mode register MM) and the μ PD78213 (ROM-less) to access external peripheral I/O through the address bus or address/data bus.

The external SFR area can be accessed by the SFR addressing method. The area features the following: Peripheral I/O can be easily manipulated and the object size can be compressed. This area can also be specified as an SFR of macro service type B.

When the external SFR area is accessed, the bus operates as in ordinary memory access (see **Chapter 13**).

3.1.5 External Memory Space

The area from 04000H to 0FCFFH (02000H to 0FD7FH for the μ PD78212) is an external memory space that can be accessed by setting a memory expansion mode register (MM). In this area, programs and table data can be stored and peripheral I/O devices can be mapped.

For the μ PD78213, the area from 00000H to 0FCFFH can be accessed at any time.

3.1.6 External Extension Data Memory Space

The area from 10000H to FFFFFH can be accessed if 1M-byte expansion mode is selected by the memory expansion mode register (MM). In this mode, pins P60 to P63 of port 6 function as the expansion address bus of four bits (A16 to A19). The data memory space is manipulated as 16 banks of 64K bytes. The low-order four bits of registers P6 and PM6 function as a bank register for selecting a bank. This memory space is useful if the kanji character generator or a large amount of data is used.

To access the space, specify the bank to be used (high-order four bits of address, A16 to A19) in the bank register (P60 to P63 of register P6, or PM60 to PM63 of register PM6). Then, execute an instruction which allows extended addressing. The high-order four bits of address output from pins P60 to P63 are valid only while an instruction that allows extended addressing is being executed.

Because two bank registers are provided, two data banks can always be used. To select either of the two bank registers, specify the operand of the instruction with or without &. If & is added, register P6 is selected as the bank register. If & is omitted, register PM6 is selected as the bank register.

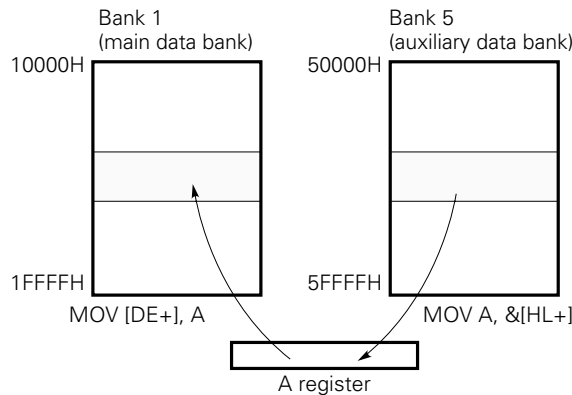
If one of the two banks is specified as the main data bank in a RAM area, and if the other bank is specified as the sub-data bank in a data ROM area, for example, the data read from the data ROM (for example, character data for the printer) can easily be enlarged or reduced in size and stored in the RAM.

Example Specifying bank 1 as the main bank and bank 5 as the sub-bank and transferring the data from bank 5 to bank 1

```

MOV    MM, #47H    ; Selects memory expansion mode.
MOV    PM6, #1H    ; Sets the main bank register (PM6).
MOV    P6, #5H     ; Sets the sub-bank register (P6).
MOV    B, #0FFH    ; Sets the loop counter.
      ⋮
LOOP : ⋮
      ⋮
MOV    A, &[HL+]   ; Reads data from bank 5. (The contents of register P6 are added as the most significant address.)
MOV    [DE+], A    ; Stores the data in bank 1. (The contents of register PM6 are added as the most significant address.)
      ⋮
      ⋮
DBNZ   B, $LOOP    ; Repetition
    
```

Fig. 3-5 Sample Data Transfer between Banks



- Remarks**
1. The MOV [DE+], A and MOV A, &[HL+] instructions are both held in bank 0.
 2. The instruction that uses register PM6 as the bank register requires a shorter instruction code and shorter execution time than that which uses register P6. The instruction that manipulates register P6 requires a shorter instruction code and shorter execution time than that which manipulates register PM6. It is, therefore, efficient to use PM6 as the main bank register that specifies the bank accessed most frequently and P6 as the sub-bank register that frequently specifies different banks.

3.2 REGISTERS

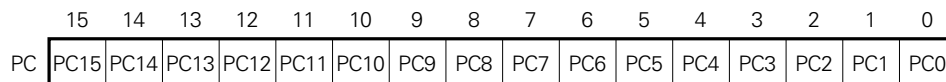
3.2.1 Program Counter (PC)

This 16-bit binary counter holds the address of the program to be executed next (see **Fig. 3-6**).

Usually, the address is automatically incremented according to the number of bytes of the instruction to be fetched. If an instruction causing a branch is executed, the contents of the register or immediate data are set.

When $\overline{\text{RESET}}$ is input, the contents at address 00000H of the internal ROM (external memory for the $\mu\text{PD78213}$) are specified in the low-order eight bits of the PC and the contents at address 00001H are specified in the high-order eight bits of the PC.

Fig. 3-6 Configuration of the Program Counter



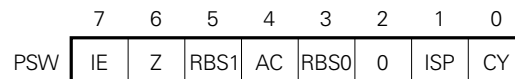
3.2.2 Program Status Word (PSW)

This 8-bit register consists of flags that are set or reset according to the execution results of an instruction (see **Fig. 3-7**).

The contents can be read and written in units of eight bits. Each flag can be manipulated by a bit manipulation instruction. The PSW is saved to a stack when a vectored interrupt request is acknowledged or when the BRK or PUSH PSW instruction is executed. The PSW is restored when an RETI, RETB, or POP PSW instruction is executed.

When $\overline{\text{RESET}}$ is input, the PSW is set to 02H. (In this state, no interrupt requests can be acknowledged.)

Fig. 3-7 Configuration of the Program Status Word



(1) Carry flag (CY)

This flag indicates whether an overflow or underflow occurs when an add/subtract instruction is executed.

If a shift/rotate instruction is executed, the flag holds a shift-out value. If a bit arithmetic/logical instruction is executed, the flag functions as a bit accumulator.

(2) Interrupt priority status flag (ISP)

This flag manages the priority of maskable vectored interrupts that can currently be acknowledged. If a maskable interrupt is acknowledged, the contents of the priority designation flag of the acknowledged interrupt are transferred to the ISP flag. If a non-maskable interrupt (NMI) is acknowledged, this flag is set to 0.

If the ISP flag is set to 0, vectored interrupts assigned a lower priority by the priority designation flag register (PR0) cannot be acknowledged. If the ISP flag is set to 1, interrupts can be acknowledged, independent of the priority. The actual acknowledgment of interrupts is controlled according to the status of the IE flag.

The contents are updated each time a maskable vectored interrupt is acknowledged.

For details, see **Chapter 12**.

(3) Register bank selection flags (RBS0, RBS1)

These two flags are used to select one of four register banks (see **Table 3-2**). The flags hold two-bit information indicating the register bank selected by the SEL RBn instruction.

Table 3-2 Selecting a Register Bank

RBS1	RBS0	Specified register bank
0	0	Register bank 0
0	1	Register bank 1
1	0	Register bank 2
1	1	Register bank 3

(4) Auxiliary carry flag (AC)

If an operation generates a carry from bit 3 or a borrow into bit 3, this flag is set (1). Otherwise, the flag is reset (0).

The flag is used when a BCD conversion instruction is executed.

(5) Zero flag (Z)

If an operation results in zero, this flag is set (1). Otherwise, the flag is reset (0).

(6) Interrupt request enable flag (IE)

This flag controls the CPU’s acknowledgment of interrupt requests.

If the flag is set to 0, interrupts are inhibited. Only nonmaskable interrupts and unmasked macro services can be acknowledged. All other interrupts are prohibited.

If the flag is set to 1, interrupts are permitted. The ISP flag, interrupt mask flag corresponding to each interrupt request, and priority designation flag control the acknowledgment of an interrupt request.

When the EI instruction is executed, the IE flag is set (1). When the DI instruction is executed or when an interrupt is acknowledged, the flag is reset (0).

3.2.3 Stack Pointer (SP)

This 16-bit register holds the first address of a stack area (LIFO: 00000H to 0FFFFH) (see **Fig. 3-8**). The stack pointer is used to address a stack area when a subroutine is executed or when an interrupt is handled.

The contents of the SP are decremented before data is written into the stack area and incremented after data is read from the stack area (see **Figs. 3-9 and 3-10**).

A special instruction can access the SP.

The contents of the SP become undefined when $\overline{\text{RESET}}$ is input. Immediately after a reset is released (before a subroutine is called or before an interrupt is acknowledged), run an initialization program to initialize the SP.

Example Initializing the SP

```
MOVW SP, #0FEE0H; SP ← 0FEE0H (if used from FEDFH)
```

Fig. 3-8 Configuration of the Stack Pointer

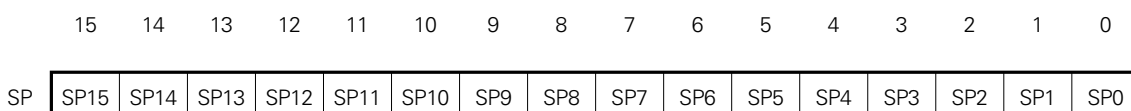
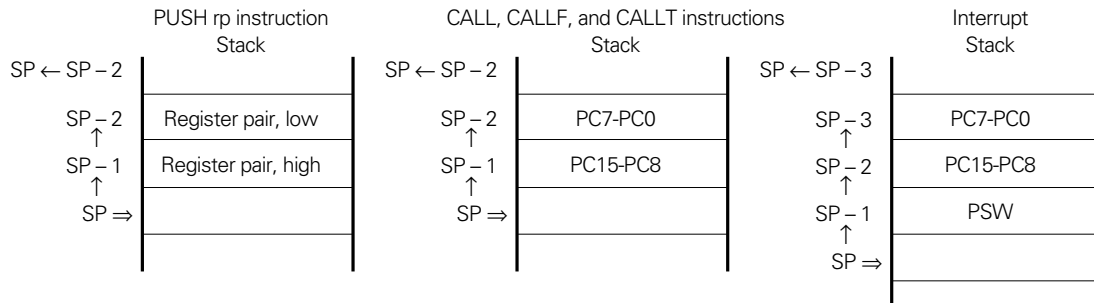
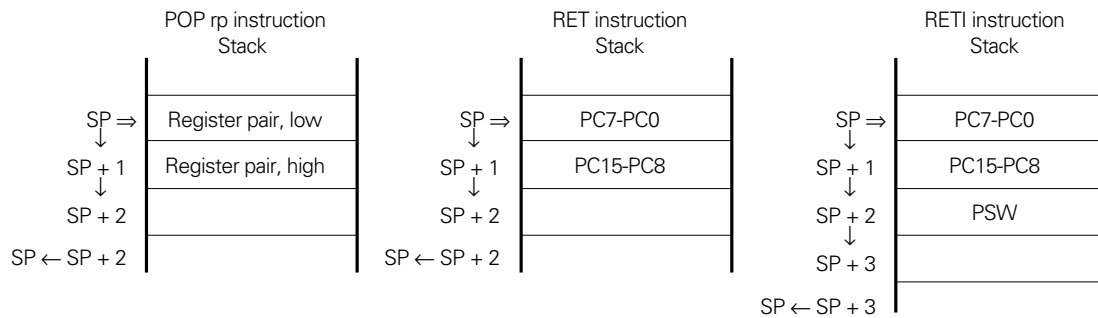


Fig. 3-9 Data Saved to the Stack Area**Fig. 3-10 Data Restored from the Stack Area**

- Cautions**
1. In stack addressing, the entire 64K bytes can be accessed. A stack area cannot be mapped in the SFR area or internal ROM area.
 2. The SP becomes undefined when $\overline{\text{RESET}}$ is input. Meanwhile, nonmaskable interrupts can be acknowledged immediately after a reset is released. If a nonmaskable interrupt request occurs, while the SP is undefined, immediately after a reset is released, an unpredictable operation may be carried out. To minimize this danger, initialize the SP immediately after a reset is released. For details, see Section 12.3.2.

3.2.4 General-Purpose Registers

(1) Configuration

General-purpose registers are mapped to special addresses (0FEE0H to 0FEFFH) in data memory. The registers are grouped into four banks, each of which consists of eight 8-bit registers (X, A, C, B, E, D, L, H) (see Fig. 3-11).

Fig. 3-11 Configuration of General-Purpose Registers

		(8-bit processing)				(16-bit processing)		
0FEE0H	A E1H	X E0H	↑ Register bank 3 (RBS1, 0 = 11) ↓	AX	E0H	↑ Register bank 2 (RBS1, 0 = 10) ↓	AX	E8H
	B E3H	C E2H		BC	E2H		BC	EAH
	D E5H	E E4H		DE	E4H		DE	ECH
	H E7H	L E6H		HL	E6H		HL	EEH
	A E9H	X E8H	↑ Register bank 1 (RBS1, 0 = 01) ↓	AX	F0H	↑ Register bank 0 (RBS1, 0 = 00) ↓	AX	F8H
	B EBH	C EAH		BC	EAH		BC	FAH
	D EDH	E ECH		DE	E4H		DE	FCH
	H E7H	L E6H		HL	F6H		HL	FEH
0FEFFH	A F1H	X F0H	↑ Register bank 0 (RBS1, 0 = 00) ↓	AX	F0H		AX	F8H
	B F3H	C F2H		BC	F2H	BC	FAH	
	D F5H	E F4H		DE	F4H	DE	FCH	
	H F7H	L F6H		HL	F6H	HL	FEH	

To specify the register bank to be used to execute an instruction, use the CPU control instruction (SEL RBn).

When $\overline{\text{RESET}}$ is input, register bank 0 is specified.

The current register bank can be checked by reading the register bank selection flags (RBS0, RBS1) in the PSW.

The area from 0FEE0H to 0FEFFH can be addressed and accessed as an ordinary data memory area, irrespective of whether the area is used for general-purpose registers.

Remark To restore the current register bank after it is changed to a different register bank, execute the PUSH PSW instruction to save the PSW to a stack, then execute the SEL RBn instruction. The POP PSW instruction can restore the register bank if the stack position is not changed. If an interrupt handling program changes the register bank, the PSW is automatically saved to a stack when an interrupt is acknowledged. The PSW is restored by the RETI or RETB instruction. If a single register bank is predetermined for the interrupt handling routine, only the SEL RBn instruction need be executed. The PUSH PSW instruction need not be executed.

Examples 1. Changing the register bank by an ordinary program

Specifying register bank 2

```

    ⋮
    PUSH PSW
    SEL  RB2
    ⋮
    POP  PSW
    ⋮

```

} Register bank 2 is used.

} The previous register bank is used.

2. Changing the register bank by an interrupt handling program

Selecting register bank 1

```

    SEL  RB1
    ⋮
    RETI

```

} Register bank 1 is used.

} The previous register bank is automatically restored upon return from the interrupt handling program.

(2) Function

General-purpose registers can be operated in units of eight bits. They can also be operated in units of 16 bits, that is, a pair of eight-bit registers can be operated as a single unit (AX, BC, DE, HL).

Each register can temporarily hold operation results or can be used as an operand of an arithmetic/logical instruction between registers.

General-purpose registers are grouped into four register banks. By separating the register banks to be used for ordinary operation from those for interrupt handling, an efficient program can be created.

The registers have different functions as described below:

A (R1) : Central register for the transfer of 8-bit data or arithmetic/logical operations. This register can also hold bit data.

The register can also hold the offset value for indexed addressing.

AX (RP0) : Central register for the transfer of 16-bit data or arithmetic/logical operations

X (R0) : Register that can hold bit data

B (R3) : Register with the functions of a loop counter. The register can be used by the DBNZ instruction.

This register can also hold the offset value for indexed addressing.

C (R2) : Register with the functions of a loop counter. The register can be used by the DBNZ instruction.

DE (RP2), HL (RP3) : Registers with the functions of a pointer. The registers specify a base address for register indirect addressing and base addressing.

This register can also hold the offset value for indexed addressing.

Each register can be identified by a name representing its function (X, A, C, B, E, D, L, H, AX, BC, DE, HL) or by an absolute name (R0 to R7, RP0 to RP3). For details of the relationship between the function names and absolute names, see **Table 3-3**.

Table 3-3 Function Names and Absolute Names

Function name	Absolute name	Function name	Absolute name
X	R0	AX	RP0
A	R1	BC	RP1
C	R2	DE	RP2
B	R3	HL	RP3
E	R4		
D	R5		
L	R6		
H	R7		

3.2.5 Special Function Registers (SFR)

A mode register, control register, and other registers with special functions, which are built-in hardware peripherals, are mapped into the 256-byte space from 0FF00H to 0FFFFH.

Caution Never access an address to which no SFR is mapped in this area. If this is attempted, the μPD78214 may enter a deadlock. To clear the deadlock, a reset signal must be input to the device.

Table 3-4 lists the special function registers (SFR). The following column headings are used:

- **Symbol** : Symbol indicating the built-in SFR. This is a reserved word for NEC's assembler (RA78K/II). For the C compiler (CC78K/II), the #pragma sfr instruction allows this symbol to be used as an sfr variable.
- **R/W** : Whether the contents of the SFR can be read or written
 - R/W : Read/write
 - R : Read-only
 - W : Write-only
- **Bit unit** : Number of bits that can be manipulated when the SFR is operated. The SFR that can be manipulated in units of 16 bits can be coded in the sfrp operand or specified by an even address. The SFR that can be manipulated in one-bit units can be coded in the bit manipulation instruction.
- **At reset** : Status of the register when RESET is input

Table 3-4 Special Function Registers (SFR) (1/2)

Address	Name of special function register (SFR)		Symbol	R/W	Bit unit			At reset	
					1 bit	8 bits	16 bits		
0FF00H	Port 0		P0	R/W	○	○	—	Not defined	
0FF02H	Port 2		P2	R	○	○	—		
0FF03H	Port 3		P3	R/W	○	○	—		
0FF04H	Port 4		P4		○	○	—		
0FF05H	Port 5		P5		○	○	—		
0FF06H	Port 6		P6	R	○	○	—		×0H
0FF07H	Port 7		P7		○	○	—		
0FF0AH		Port 0 buffer register	P0L	R/W	○	○	—	Not defined	
0FF0BH	Port 0 buffer register		P0H		○	○	—		
0FF0CH	Real-time output port control register		RTPC		○	○	—	00H	
0FF10H	16-bit compare register 0 (16-bit timer/counter)		CR00	R/W	—	—	○	Not defined	
0FF11H					—	—	○		
0FF12H	16-bit compare register 1 (16-bit timer/counter)		CR01		—	—	○		
0FF13H					—	—	○		
0FF14H	8-bit compare register (8-bit timer/counter 1)		CR10		—	○	—		
0FF15H	8-bit compare register (8-bit timer/counter 2)		CR20		—	○	—		
0FF16H	8-bit compare register (8-bit timer/counter 2)		CR21		—	○	—		
0FF17H	8-bit compare register (8-bit timer/counter 3)		CR30	—	○	—			
0FF18H	16-bit capture register (16-bit timer/counter)		CR02	R	—	—	○		
0FF19H					—	—	○		
0FF1AH	8-bit capture register (8-bit timer/counter 2)		CR22	R/W	—	○	—		
0FF1CH	8-bit capture/compare register (8-bit timer/counter 1)		CR11		—	○	—		
0FF20H	Port 0 mode register		PM0	W	—	○	—	FFH	
0FF23H	Port 3 mode register		PM3		—	○	—		
0FF25H	Port 5 mode register		PM5		—	○	—		
0FF26H	Port 6 mode register		PM6	R/W	○	○	—	F×H	
0FF30H	Capture/compare control register 0		CRC0	W	—	○	—	10H	
0FF31H	Timer output control register		TOC		—	○	—		
0FF32H	Capture/compare control register 1		CRC1		—	○	—		
0FF34H	Capture/compare control register 2		CRC2		—	○	—		
0FF40H	Pull-up-resistor-option register		PUO	R/W	○	○	—	00H	
0FF43H	Port 3 mode control register		PMC3		○	○	—		

Table 3-4 Special Function Registers (SFR) (2/2)

Address	Name of special function register (SFR)	Symbol	R/W	Bit unit			At reset	
				1 bit	8 bits	16 bits		
0FF50H	16-bit timer register 0	TM0	R	—	—	○	0000H	
0FF51H				—	—			
0FF52H	8-bit timer register 1	TM1		—	○	—	00H	
0FF54H	8-bit timer register 2	TM2		—	○			
0FF56H	8-bit timer register 3	TM3	—	○				
0FF5CH	Prescaler mode register 0	PRM0	W	—	○	—	00H	
0FF5DH	Timer control register 0	TMC0	R/W	—	○			
0FF5EH	Prescaler mode register 1	PRM1	W	—	○			
0FF5FH	Timer control register 1	TMC1	R/W	—	○			
0FF68H	A/D converter mode register	ADM	R	○	○	—	Not defined	
0FF6AH	A/D conversion result register	ADCR		—	○			
0FF80H	Clock synchronous serial interface mode register	CSIM	R/W	○	○	—	00H	
0FF82H	Serial bus interface control register	SBIC		○	○			
0FF86H	Serial shift register	SIO		—	○	—	Not defined	
0FF88H	Asynchronous serial interface mode register	ASIM		○	○	—	80H	
0FF8AH	Asynchronous serial interface status register	ASIS	R	○	○	—	00H	
0FF8CH	Serial reception buffer: UART	RXB		—	○			—
0FF8EH	Serial transmission shift register: UART	TXS	W	—	○	—	00H	
0FF90H	Baud rate generator control register	BRGC		—	○			—
0FFC0H	Standby control register	STBC	R/W	—	○	—	0000 × 000B	
0FFC4H	Memory expansion mode register	MM	R/W	○	○	—	20H	
0FFC5H	Programmable weight control register	PW		○	○	—	80H	
0FFC6H	Refresh mode register	RFM		○	○	—	00H	
0FFD0H 0FFDFH	External SFR area	—	R/W	○	○	—	Not defined	
0FFE0H	Interrupt request flag register L	IF0L		IF0	○	○	○	0000H
0FFE1H	Interrupt request flag register H	IF0H			○	○		
0FFE4H	Interrupt mask flag register L	MK0L		MK0	○	○	○	FFFFH
0FFE5H	Interrupt mask flag register H	MK0H			○	○		
0FFE8H	Priority designation flag register L	PR0L		PR0	○	○	○	FFFFH
0FFE9H	Priority designation flag register H	PR0H			○	○		
0FFECH	Interrupt service mode register L	ISM0L		ISM0	○	○	○	0000H
0FFEDH	Interrupt service mode register H	ISM0H			○	○		
0FFF4H	External interrupt mode register 0	INTM0		R/W	○	○	—	00H
0FFF5H	External interrupt mode register 1	INTM1			○	○		
0FFF8H	Interrupt status register	IST			○	○		

3.3 NOTES

(1) A program fetch from the internal RAM area is prohibited.

(2) **Operation of the stack pointer**

In stack addressing, the entire 64K bytes can be accessed. No stack area can be mapped into the SFR area or internal ROM area.

(3) **Special function register (SFR)**

Never access an address to which no SFR is mapped in the area from 0FF00H to 0FFFFH. If this is attempted, the μ PD78214 may enter a deadlock. To clear the deadlock, a reset signal must be input to the device.

(4) **Initializing the stack pointer**

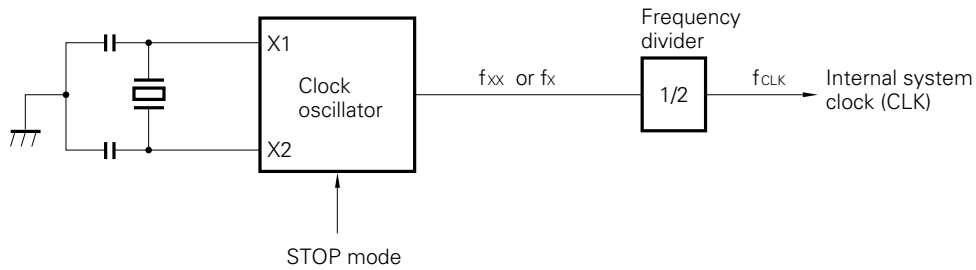
The SP becomes undefined when RESET is input. Meanwhile, nonmaskable interrupts can be acknowledged immediately after a reset is released. If a nonmaskable interrupt request occurs while the SP is undefined immediately after a reset is released, an unpredictable operation may be carried out. To minimize this danger, initialize the SP immediately after a reset is released. For details, see **Section 12.3.2**.

CHAPTER 4 CLOCK GENERATOR

4.1 CONFIGURATION AND FUNCTION

A clock generator generates and controls the internal system clock (CLK) sent to the CPU. Fig. 4-1 shows the configuration of the clock generator.

Fig. 4-1 Block Diagram of Clock Generator



Remarks f_{XX} : Crystal/ceramic oscillation frequency

f_X : External clock frequency

f_{CLK} : Internal system clock frequency (= $1/2 \cdot f_{XX}$ or $1/2 \cdot f_X$)

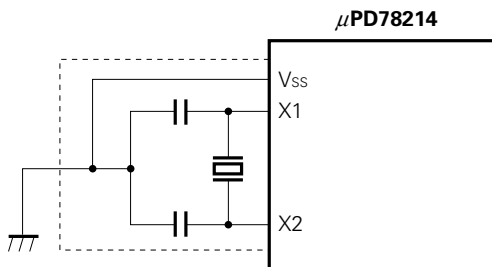
The clock oscillator oscillates according to a crystal or ceramic resonator connected to pins X1 and X2. In standby (STOP) mode, the oscillation is stopped (see **Chapter 14**).

The external clock can also be input. To input the external clock, input the clock signal to pin X1 and the inverted signal to pin X2. If the external clock is input, STOP mode cannot be selected.

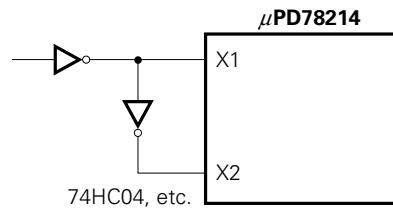
The frequency divider divides the output from the clock oscillator (f_{XX} for the crystal/ceramic oscillation, f_X for the external clock) by two and generates the internal system clock (CLK).

Fig. 4-2 External Circuit for the Clock Oscillator

(a) Crystal/ceramic oscillation



(b) External clock



Caution When using the clock oscillator, the adverse influence of stray capacitance must be avoided. When configuring the circuit enclosed in a dashed line, observe the following: ★

- Minimize the wiring length.
- Do not let other signal lines cross this circuit.
- Keep this circuit away from lines through which a varying high current flows.
- Always ground the capacitors of the oscillator at the same potential as V_{SS} . Do not ground the capacitors to a ground pattern through which a high current flows.
- Do not draw signals from the oscillator.

Remark Different uses of the crystal and ceramic resonator

Generally, a crystal's oscillation frequency is quite stable. Crystals are ideal for high-precision time management (for example, clock or frequency measurement). In comparison with crystals, ceramic resonators are less stable but offer three advantages: a shorter oscillation start time, smaller dimensions, and lower price. Ceramic resonators are suitable for general applications (which do not require high-precision time management). If a ceramic resonator incorporating capacitors is used, the number of components and the mounting area can be reduced.

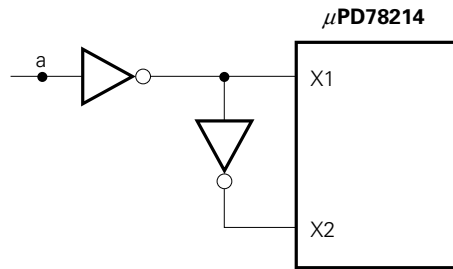
4.2 NOTES

Regarding the clock generator, note the following:

4.2.1 Inputting an External Clock

- (1) When inputting an external clock, do not select STOP mode. The clock generator may be damaged. At least, its reliability will be adversely affected.
- (2) When inputting an external clock, input the clock signal to pin X1 and the inverted signal to pin X2. If the inverted signal is not input to the pin X2, malfunctions may readily occur due to noise.
- (3) When inputting an external clock, use HCMOS or a device having equivalent drive capability.
- (4) Do not draw signals from pins X1 and X2. Draw signals from point a shown in Fig. 4-3.

Fig. 4-3 Point from Which Signals Can Be Drawn When an External Clock Is Input

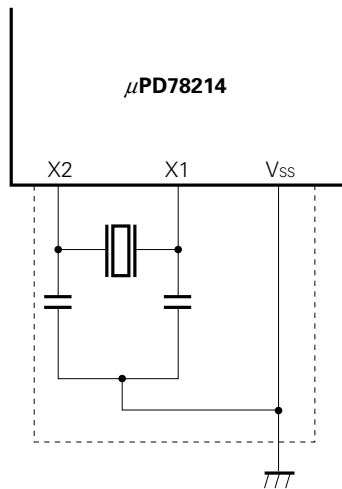


- (5) Minimize the length of the line connecting pin X1 to pin X2 through the inverter.

4.2.2 Using the Crystal/Ceramic Oscillator

- (1) The oscillator is a high-frequency analog circuit. Use the oscillator carefully. Special notes are given below:
 - Minimize the lengths of the wiring.
 - Do not let other signal lines cross this circuit.
 - Keep the oscillator away from a line through which a varying high current flows.
 - Always ground the capacitors of the oscillator at the same potential as pin V_{SS}. Do not ground the capacitors to a ground pattern through which a high current flows.
 - Do not draw signals from the oscillator.

The microcomputer is capable of normal, stable operation only when the oscillation is normal and stable. If a high-precision oscillation frequency is required, consult with the oscillator manufacturer.

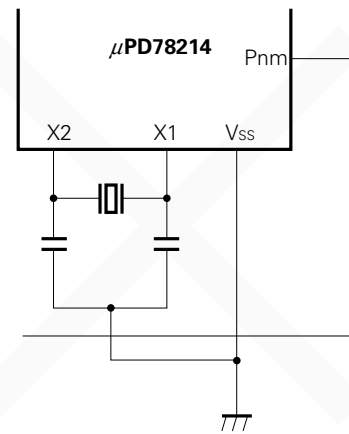
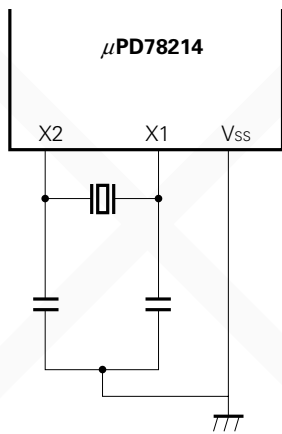
Fig. 4-4 Notes on Connection of the Oscillator

4

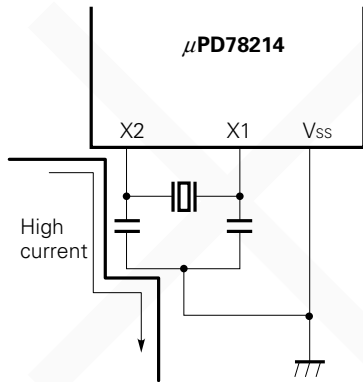
- Cautions**
1. Place the oscillator as close as possible to pins X1 and X2.
 2. Do not let other signal lines cross the circuit enclosed in a dashed line.

Fig. 4-5 Incorrect Oscillator Connections

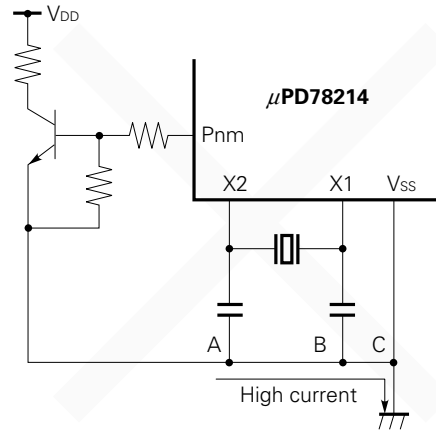
- (a) The wiring length of the external circuit is too long. (b) A signal line is allowed to cross the oscillator.



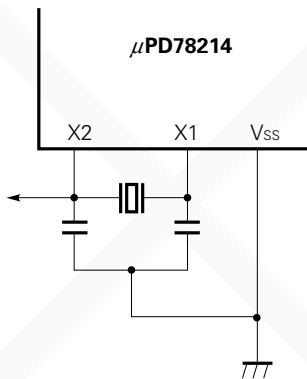
(c) A varying high current flows too close to the signal line.



(d) A current flows through the ground line of the oscillator. (The potentials vary at points A, B, and C.)



(e) A signal is being drawn from the oscillator.



(2) At power-on or return from STOP mode, some time is required for the oscillation to settle. Generally, a crystal requires a few milliseconds, and a ceramic resonator several hundreds of microseconds, for the oscillation to settle.

The oscillation settling time is determined as described below. Ensure that sufficient time is allowed for the oscillation to settle.

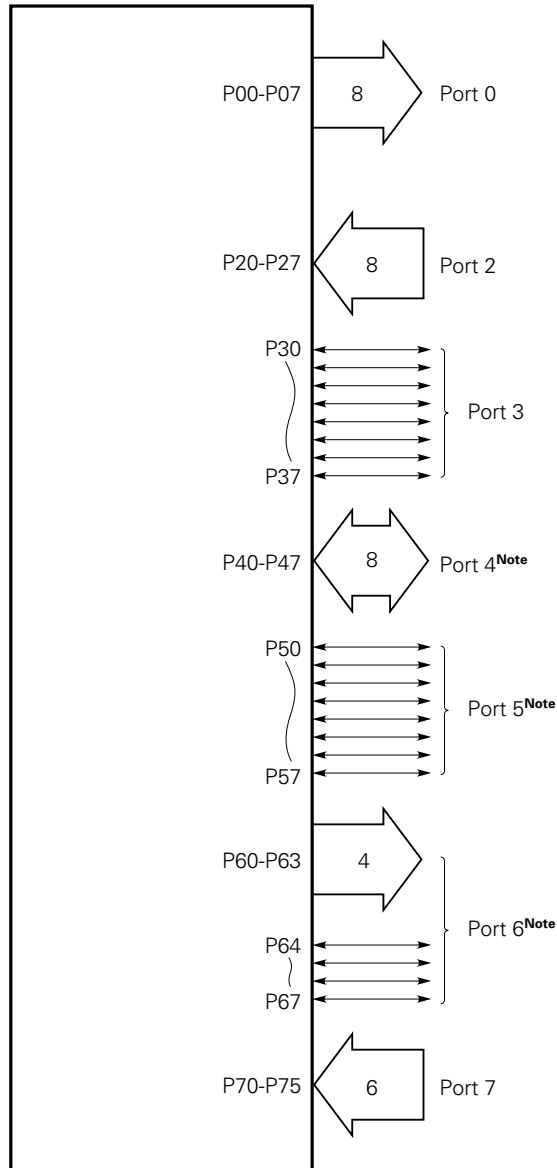
- ① At power on : $\overline{\text{RESET}}$ input (reset period)
- ② At return from STOP mode : (i) $\overline{\text{RESET}}$ input (reset period)
 (ii) (Period in which the NMI signal is active) + (period specified for the timer for automatic start)

CHAPTER 5 PORT FUNCTIONS

5.1 DIGITAL I/O PORTS

The μ PD78214 has the ports shown in Fig. 5-1. These ports can be used for various types of control. Table 5-1 lists the function of each port. For ports 2 through 6, software can specify whether to use a built-in pull-up resistor for inputs.

Fig. 5-1 Port Configuration



Note For μ PD78213, P40 through P47, P50 through P57, P64, or P65 does not function as ports.

Table 5-1 Port Functions

Name	Pin name	Function	Software-specified pull-up resistor
Port 0	P00-P07	Can be specified for either output in 8-bit units or high impedance. Can also function as 4-bit real-time output port (P00-P03 and P04-P07). Can drive transistors directly.	—
Port 2	P20-P27	Input port	In 6-bit units (P22-P27)
Port 3	P30-P37	Can be specified for either input or output in bit units.	For all input pins at a time
Port 4 ^{Note}	P40-P47	Can be specified for either input or output in 8-bit units Can drive LEDs directly.	In 8-bit units
Port 5 ^{Note}	P50-P57	Can be specified for either input or output in bit units Can drive LEDs directly.	For all input pins at a time
Port 6 ^{Note}	P60-P63	Output port	—
	P64-P67	Can be specified for either input or output in bit units	For all input pins at a time
Port 7	P70-P75	Input port	—

Note For μPD78213, P40 through P47, P50 through P57, P64, or P65 does not function as ports.

Table 5-2 Number of I/O Ports

I/O Port	Total	Input mode	Output mode	
		Software-specified pull-up resistors	Directly driven LEDs	Direct driven transistor
Input port	14 (14)	6 (6)	—	—
I/O port	28 (10)	28 (10)	16 (0)	0 (0)
Output port	12 (12)	—	0 (0)	8 (8)
Total	54 (36)	34 (16)	16 (0)	8 (8)

Values enclosed in parentheses apply to the μPD78213.

5.2 PORT 0

Port 0 is an 8-bit output-only port with an output latch and can drive transistors directly. The port 0 mode register (PM0) can specify that port 0 be in either the output mode or high-impedance state in 8-bit units.

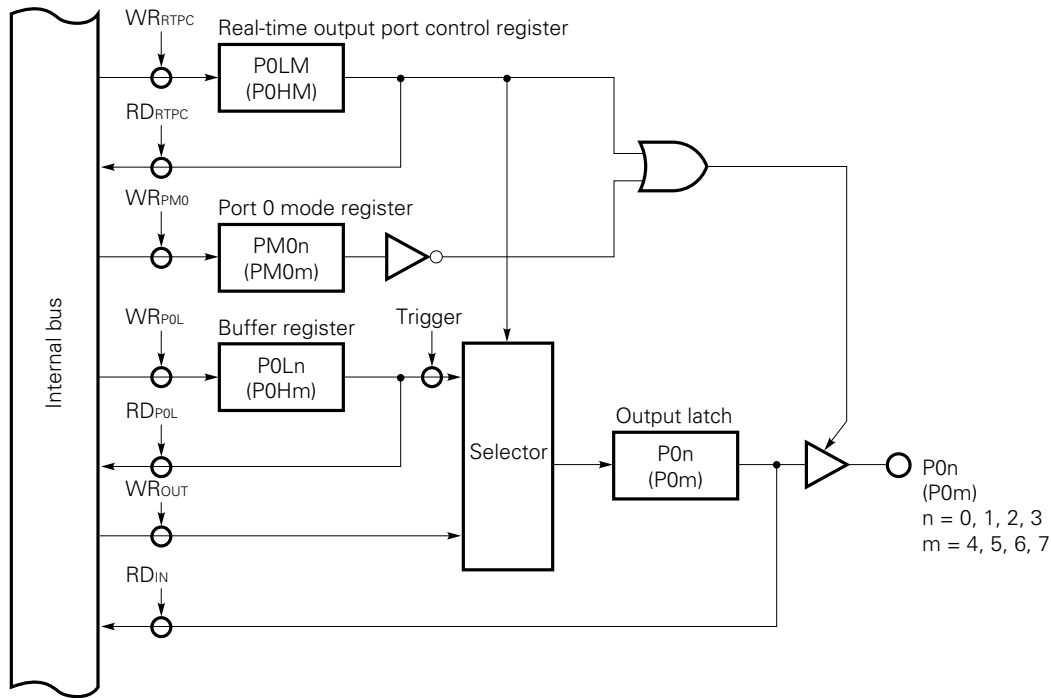
A set of P00 through P03 and a set of P04 through P07 function as a 4-bit real-time output port. Similarly, a set of P00 through P07 functions as an 8-bit real-time output port. These ports can output the contents of the P0L and P0H buffers at arbitrary intervals. The real-time output trigger control register (RTPC) specifies whether port 0 is to function as an ordinary output port or real-time output port.

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

5.2.1 Hardware Configuration

Fig. 5-2 shows the hardware configuration of port 0.

Fig. 5-2 Configuration of Port 0

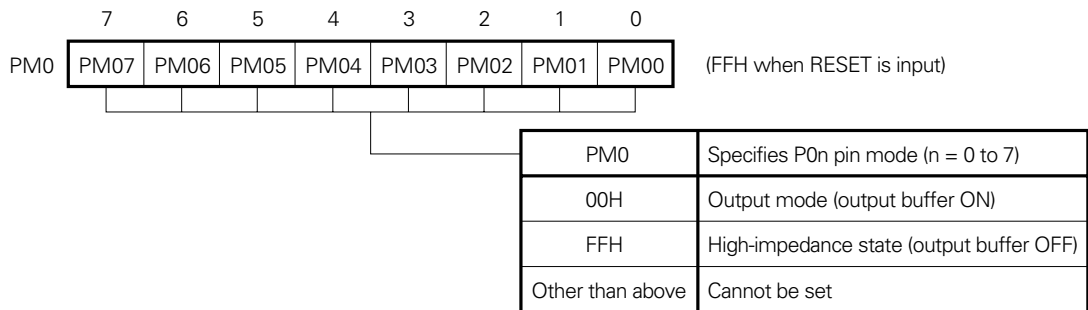


5

5.2.2 Setting the Input/Output Mode and Control Mode

The port 0 mode register (PM0) sets the I/O mode of port 0, as shown in Fig. 5-3. This register is set by an 8-bit data transfer instruction. (It can neither manipulated in bit units nor read-accessed).

Fig. 5-3 Port 0 Mode Register Format



To use port 0 as a real-time output port, it is necessary to set the P0LM and P0HM bits of the real-time output port control register (RTPC) to 1.

When the P0LM and P0HM bits are set, the output buffer for each pin is turned on, and the contents of the output latch are output to the pin, regardless of the contents of the PM0.

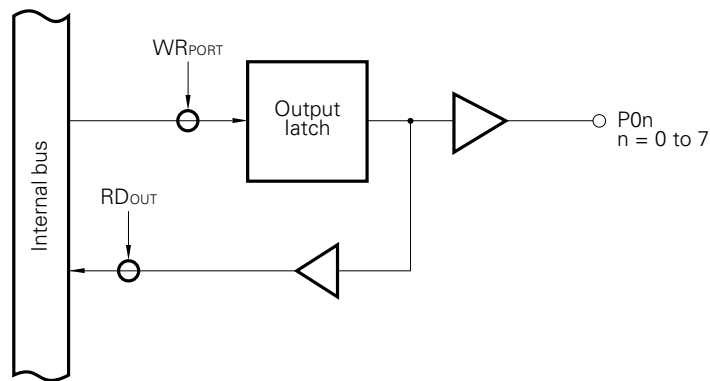
5.2.3 Operation

Port 0 is an output-only port.

Once port 0 is put in the output mode, the output latch becomes operable, enabling data transfer between the output latch and accumulator according to a transfer instruction. The output latch can be loaded with any data by a logical operation instruction. Once the output latch is loaded with some data, it retains the data until it is loaded with other data.

If port 0 is specified to be a real-time output port, no data can be written to the output latch. However, it is possible to read the contents of the output latch if it is in the real-time output port mode.

Fig. 5-4 Port Specified as an Output Port



5.2.4 Built-In Pull-Up Resistor

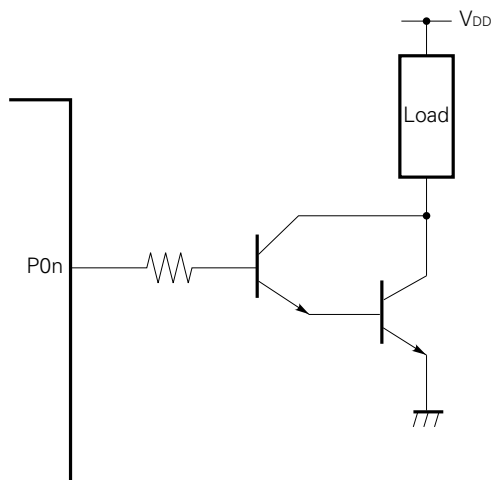
Port 0 has no built-in pull-up resistor.

5.2.5 Driving Transistors

Because port 0 has an enhanced driving capacity for the high level side of the output buffer, it can drive a transistor directly on an active-high signal.

Fig. 5-5 shows an example of connecting a transistor to the port.

Fig. 5-5 Example of Driving a Transistor



5.3 PORT 2

Port 2 is an 8-bit input-only port. P22 through P27 have a software-programmable built-in pull-up resistor. In addition to functioning as an input port, port 2 functions as a control signal input pin such as for external interrupts (see **Table 5-3**). All the 8 input pins of port 2 are configured as Schmitt trigger circuits in order to prevent malfunction due to noise.

Table 5-3 Functions of Port 2

Port	Function
P20	Input port/NMI input ^{Note}
P21	Input port/INTP0 input/CR11 capture trigger input/real-time output port trigger signal
P22	Input port/INTP1 input/CR22 capture trigger input
P23	Input port/INTP2 input/CI input
P24	Input port/INTP3 input/CR02 capture trigger input
P25	Input port/INTP4 input/ASCK input
P26	Input port/INTP5 input/A/D controller external trigger input
P27	Input port/SI input

Note NMI inputs are accepted regardless of whether interrupts are enabled.

(a) Function as a port pin

Although the pins of port 0 are shared by more than one function, the level of each pin can be read and tested.

(b) Function as a control signal input pin

(i) NMI (nonmaskable interrupt)

The NMI pin receives a nonmaskable interrupt request from the outside. The external interrupt mode register (INTM0) specifies which edge, rising or falling, is valid as an interrupt request signal.

(ii) INTP0 through INTP5 (interrupt from peripherals)

The INTP0 through INTP5 pins receive interrupt requests from the outside. An interrupt occurs when one of these pin receives an edge specified as valid by the external interrupt mode register (INTM0 and INTM1). (See **Chapter 11**.)

The INTP0 through INTP3 and INTP5 pins are also used to receive external triggers, as listed below:

- INTP0: 8-bit timer/counter 1 capture trigger input and real-time output port trigger signal
- INTP1: 8-bit timer/counter 2 capture trigger input
- INTP2: 8-bit timer/counter 2 external count clock input
- INTP3: 16-bit timer/counter 2 capture trigger input
- INTP5: A/D converter external trigger input

(iii) CI (clock input)

The CI pin receives external clock inputs for 8-bit timer/counter 2.

(iv) ASCK (asynchronous serial clock)

The ASCK pin receives a baud rate clock from the outside.

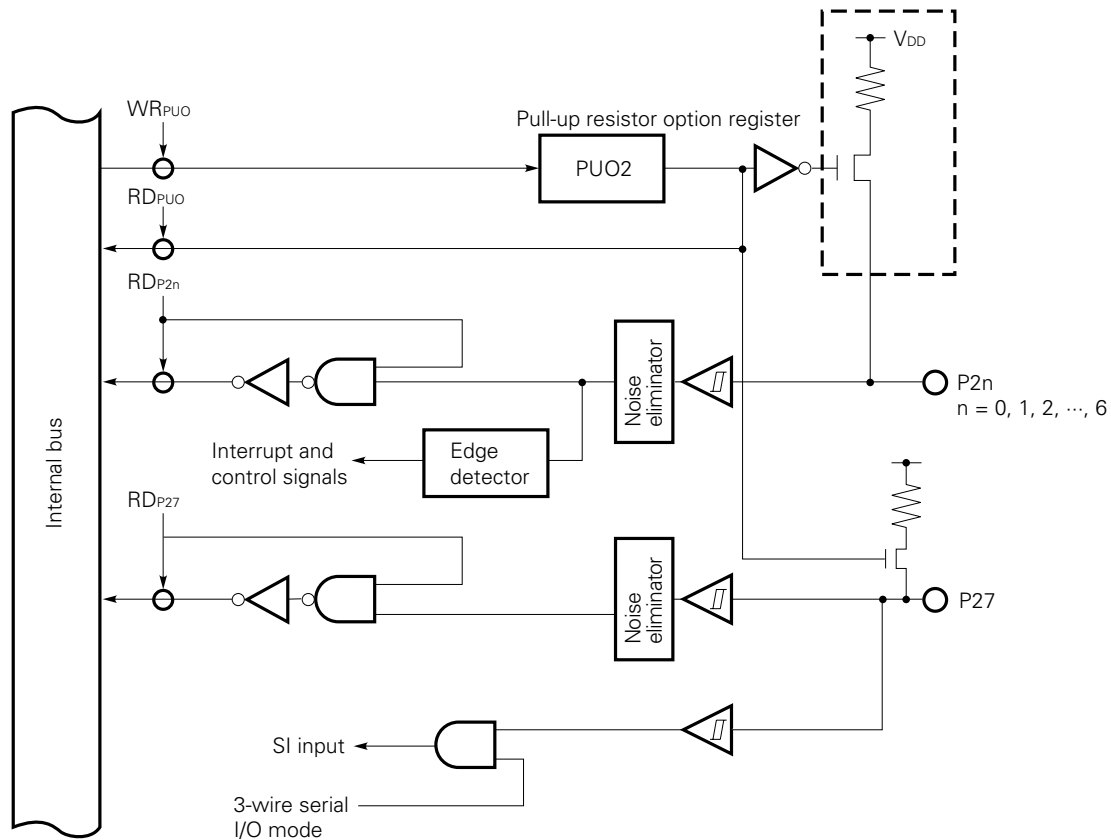
(v) SI (serial input)

The SI pin receives serial data (during three-wire serial I/O mode).

5.3.1 Hardware Configuration

Fig. 5-6 shows the configuration of port 2

Fig. 5-6 Block Diagram of Port 2



Note P20 or P21 does not have a circuit enclosed in a dotted box.

5.3.2 Setting the Input Mode and Control Mode

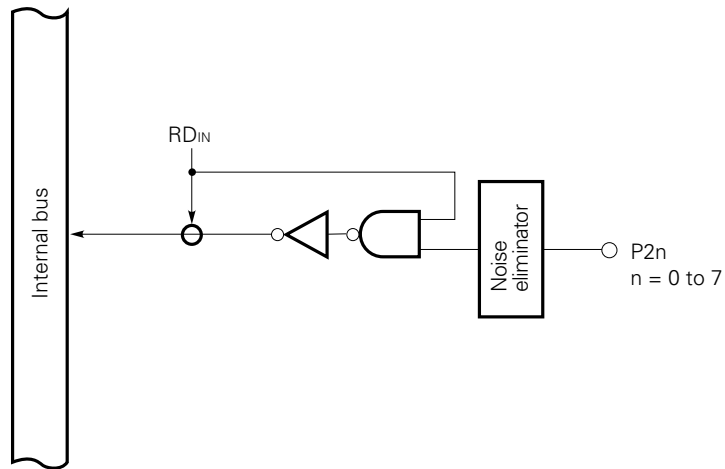
Port 2 is an input-only port.

There is no register to specify an input mode for port 2. Port 2 is always ready to receive control signals. A register such as a control register in the hardware is used to specify what control signal to receive.

5.3.3 Operation

Port 2 is an input-only port, and the level of each pin of it can be read and tested.

For P20 through P27, the level from which noise has been removed can be read and tested. See **Chapter 11** for removing noise.

Fig. 5-7 Port Specified as an Input Port

5

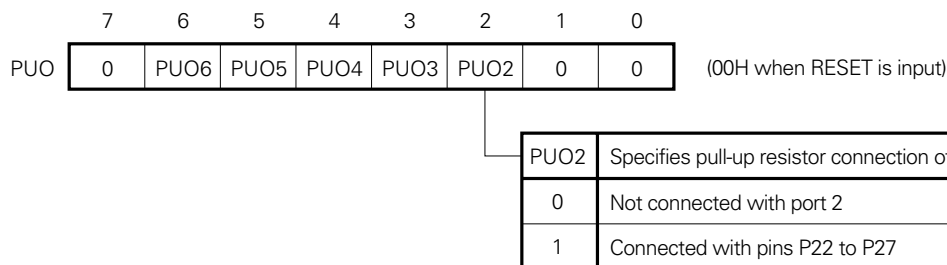
Caution For the in-circuit emulator, the level of each port 2 pin from which noise has not been removed can be read and tested.

5.3.4 Built-In Pull-Up Resistor

P22 through P27 have built-in pull-up resistors. When they must be pulled up, the built-in pull-up resistors should be used. Use of the built-in pull-up resistors can reduce the number of the required components and the required installation space.

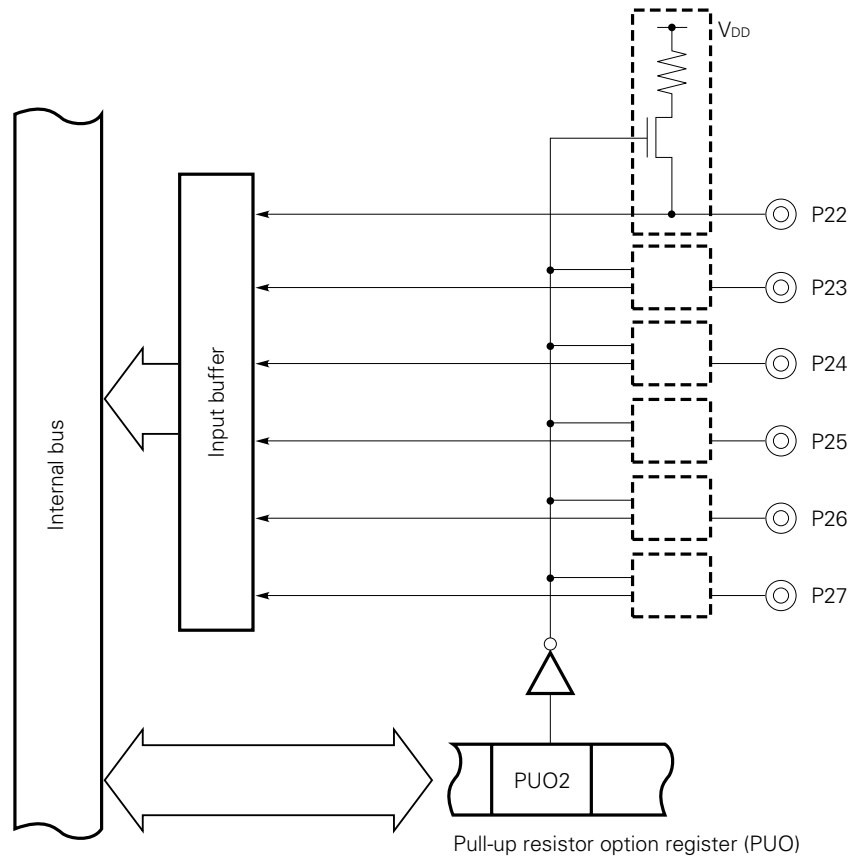
The PUO2 bit of the pull-up-resistor-option register (PUO) can specify whether to use the built-in pull-up resistors at P22 through P27, for all six pins at one time. (It is impossible to specify use of the built-in pull-up resistor for an individual bit independently of the other bits.)

P20 or P21 does not have a built-in pull-up resistor.

Fig. 5-8 Built-In Pull-Up Resistor Format

Remark Resetting the PUO2 bit to 00H can reduce the required current in the STOP mode.

Fig. 5-9 Connection of Pull-Up Resistors (Port 2)



Caution P22 through P26 are not pulled up immediately after a reset. In this case, INT_{P1} through INT_{P5} (one of the multiple functions assigned to P22 to P26) may set interrupt request flags. To avoid this problem, specify use of the pull-up resistors in the initialization routine, before clearing the interrupt flags.

5.4 PORT 3

Port 3 is an 8-bit I/O port with an output latch. The port 3 mode register (PM3) can put each bit of this port in either the input or output mode, separately from the other bits. Each pin has a software-programmable built-in pull-up resistor.

In addition to I/O functions, each pin works as control signal pin.

The port 3 mode control register (PMC3) can specify the mode of operation for each pin separately from the other pins, as listed in Table 5-4. The level of any pin can be read and tested, regardless of what function the pin is performing.

When the $\overline{\text{RESET}}$ signal is input, port 3 becomes an input port (in the high output impedance state), and the contents of the output latch become undefined.

Table 5-4 Port 3 Operating Modes (n = 0 through 7)

Mode	Port mode	Control signal I/O mode
Condition	PMC3n = 0	PMC3n = 1
P30	I/O port	RxD input
P31		TxD output
P32		$\overline{\text{SCK}}$ I/O
P33		SO output or SB0 I/O
P34		TO0 output
P35		TO1 output
P36		TO2 output
P37		TO3 output

(a) Port mode

If a port is put in a port mode by the PMC3 register, the port mode register (PM3) can put each bit of the port in either the input or output mode independently of the other bits.

(b) Control signal I/O mode

The PMC3 register can specify each pin of port 3 as a control pin independently of the other pins, as described below.

(i) RxD (receive data)

The RxD pin receives serial data from the asynchronous serial interface.

(ii) TxD (transmit data)

The TxD pin outputs serial data to the asynchronous serial interface.

(iii) $\overline{\text{SCK}}$ (serial clock)

The SCK pin is a serial clock I/O pin for the clock-synchronized serial interface.

(iv) SO (serial output)/SB0 (serial bus)

The SO pin outputs serial data (during three-wire serial I/O mode). The SB0 pin is a serial bus I/O pin (during the SBI mode).

Remark For bit 3 (P33) of port 3, "SB0" is a reserved word in the NEC assembly program package. The bit is also defined in a header file named sfrbit.h by the C compiler.

(v) TO0 through TO3 (timer output)

The TO0 through TO3 pins are timer output pins.

5.4.1 Hardware Configuration

Fig. 5-10 through 5-13 show the configuration of port 3.

Fig. 5-10 Block Diagram of P30 (Port 3)

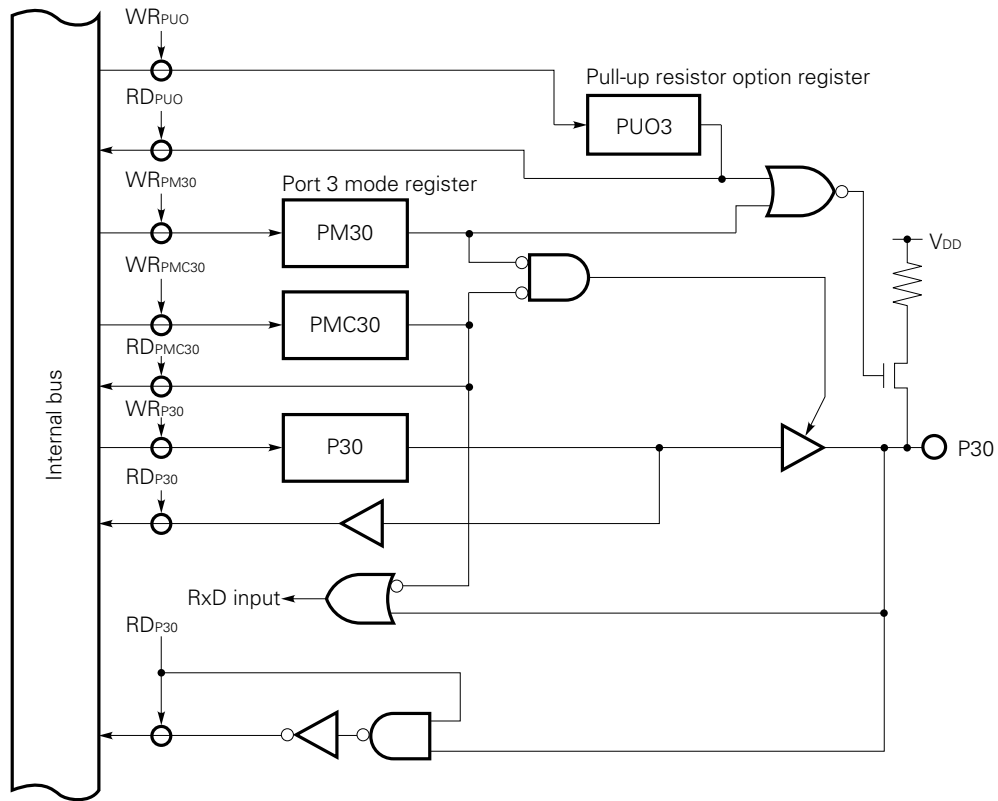
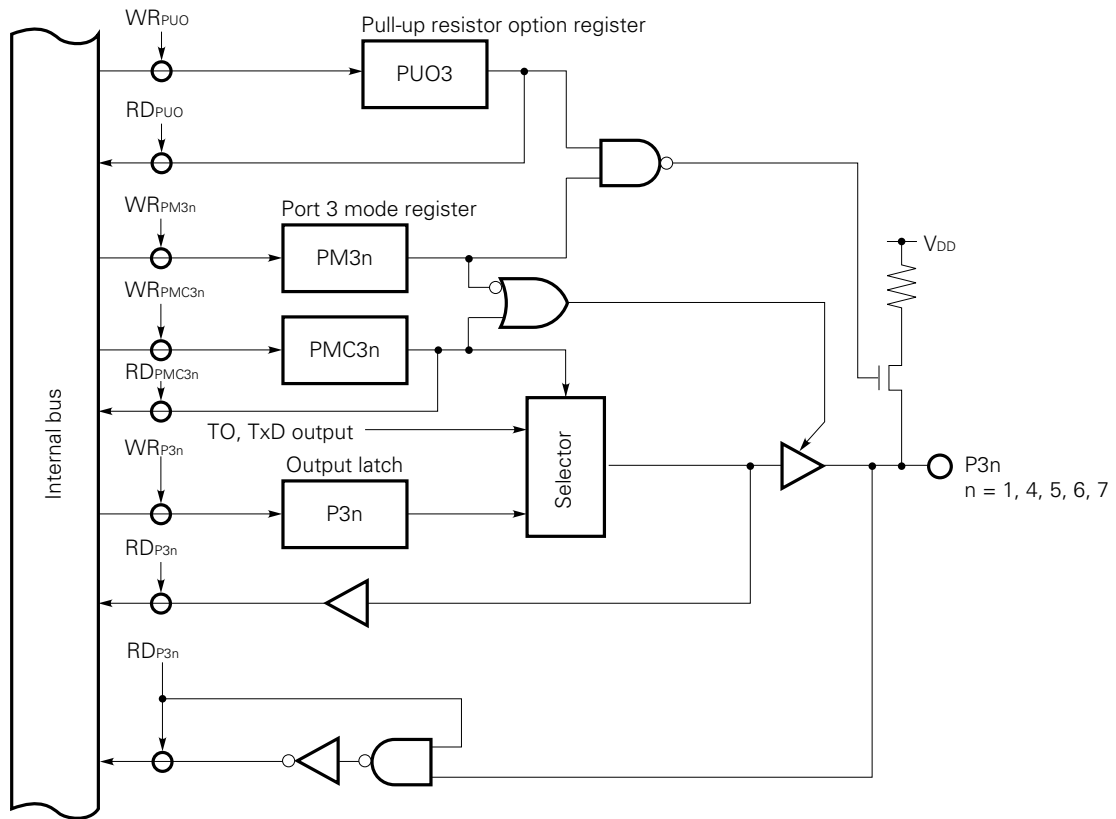


Fig. 5-11 Block Diagram of P31, and P34 through P37 (Port 3)



★

Fig. 5-12 Block Diagram of P32 (Port 3)

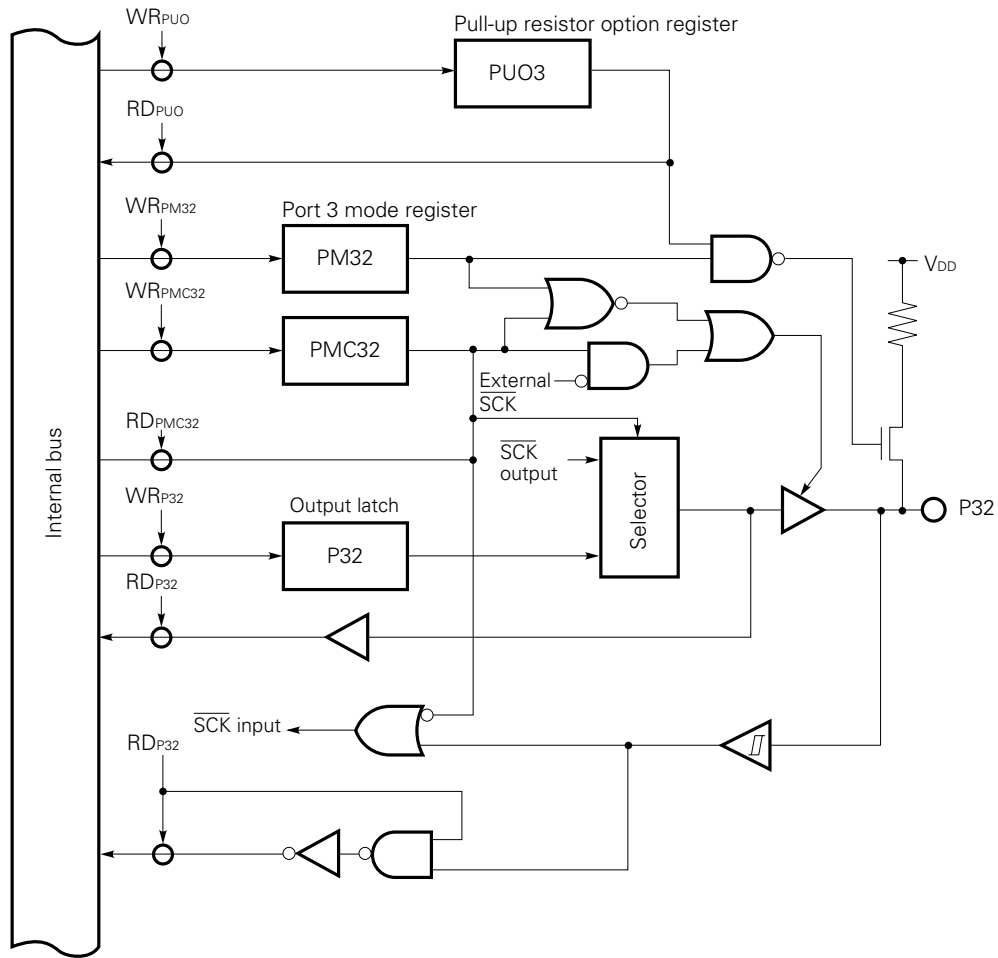
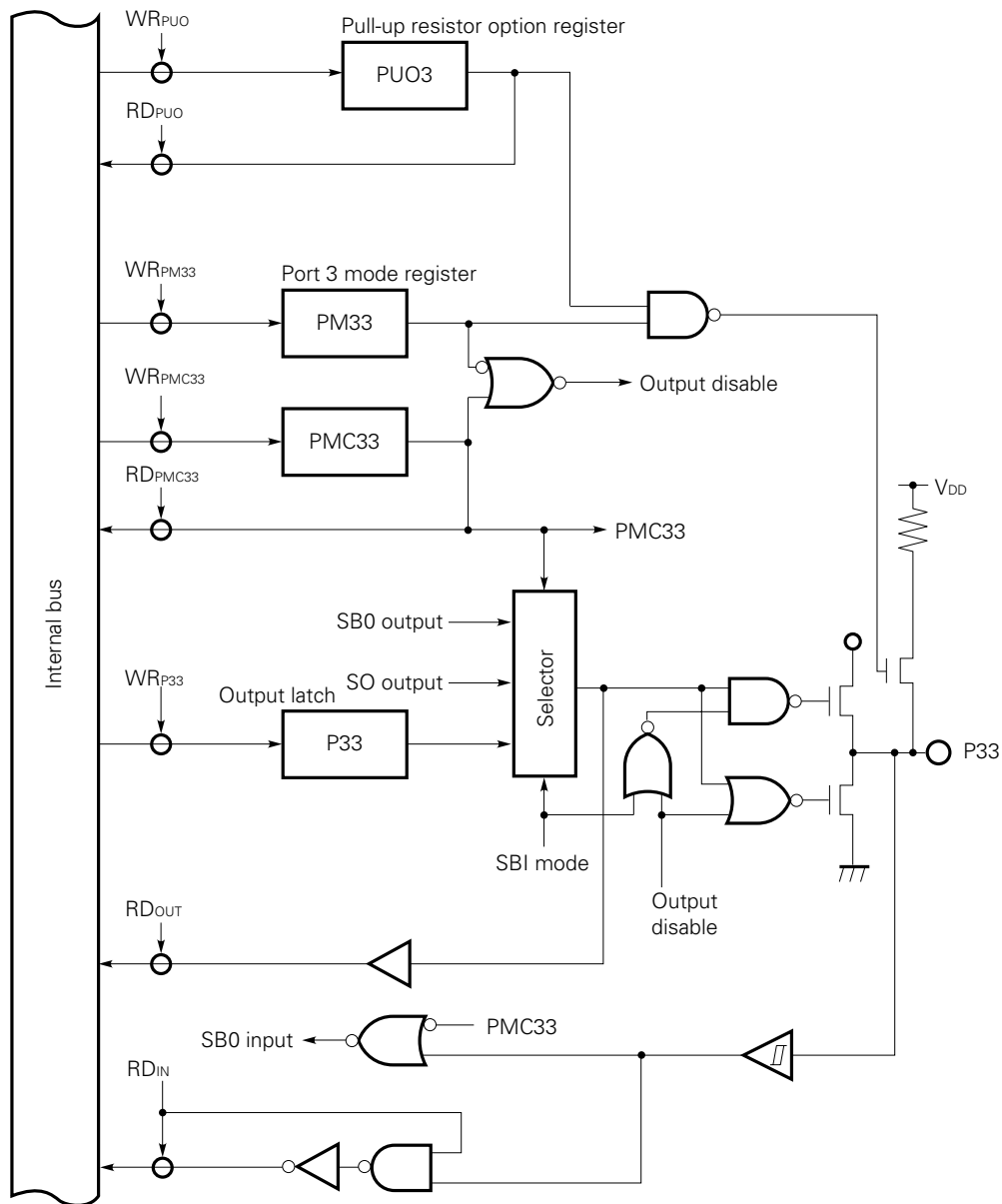


Fig. 5-13 Block Diagram of P33 (Port 3)

★



5

5.4.2 Setting the I/O Mode and Control Mode

The port 3 mode register (PM3) can put each pin of port 3 in either the input or output mode independently of the other pins, as shown in Fig. 5-14.

The PM3 register is loaded with data using an 8-bit data transfer instruction; it cannot be bit-manipulated or read-accessed.

In addition to I/O port functions, each pin of port 3 works as a control signal pin. As shown in Fig. 5-15, the port 3 mode control register (PMC3) can specify the mode of control for each pin.

Fig. 5-14 Port 3 Mode Register Format

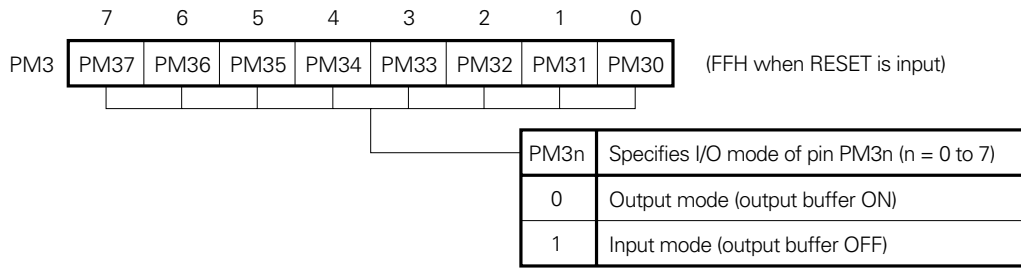
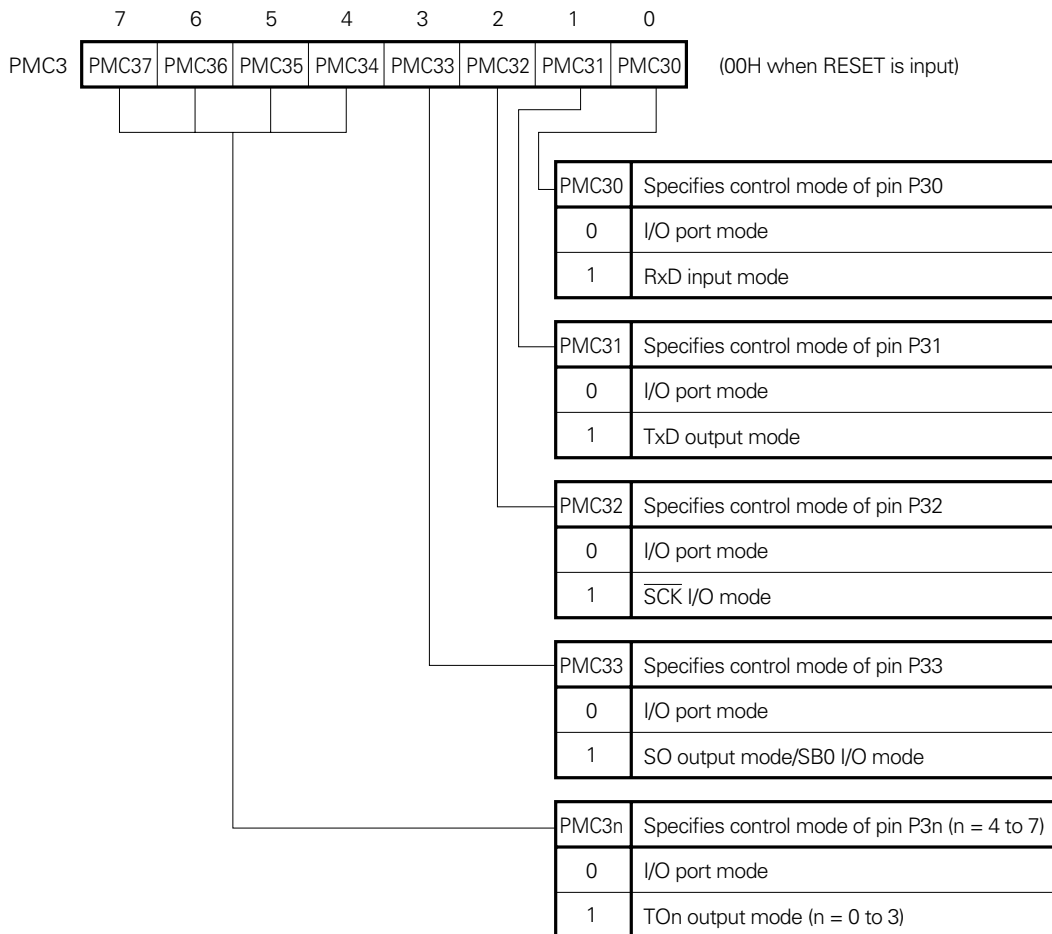


Fig. 5-15 Port 3 Mode Control Register (PMC3) Format



5.4.3 Operation

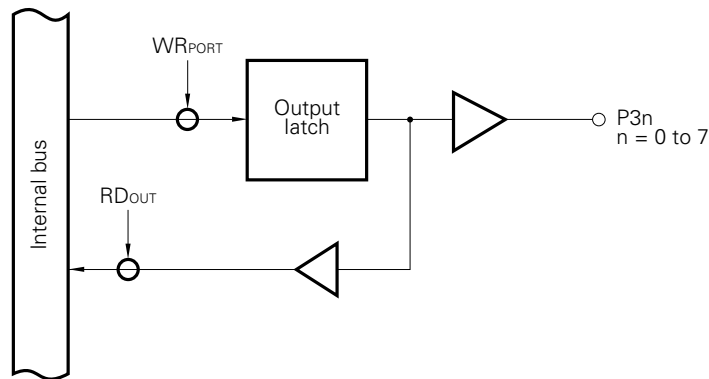
Port 3 is an I/O port. Its pins also function as control signal pins.

(1) Output port

When port 3 is in the output mode, its output latch is operable. Once the output latch becomes operable, data can be transferred between the output latch and the accumulator using a transfer instruction. The output latch can be loaded with any data by a logical operation instruction. Once the output latch is loaded with some data, it retains the data until it is loaded with other data.

Note This includes a case in which any other bit of the same port is manipulated using a bit manipulation instruction.

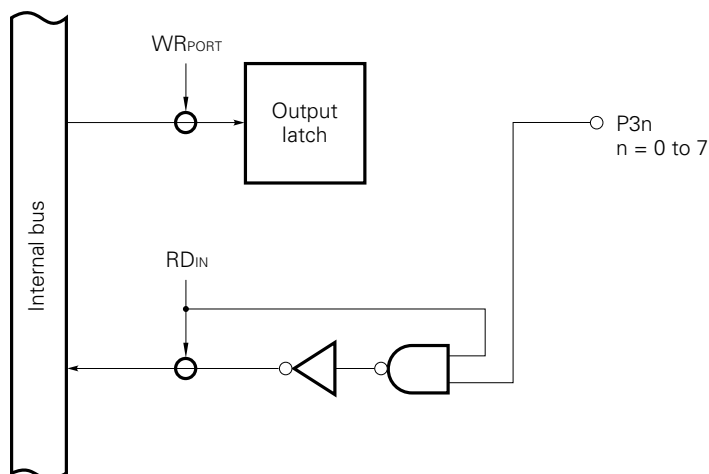
Fig. 5-16 Port Specified as an Output Port



(2) Input port

The level of each pin of port 3 can be transferred to the accumulator by a transfer instruction. Also in this case, data can be written to the output latches, and all output latches store data transferred from the accumulator by a transfer instruction or other similar instruction, regardless of the current mode of the port operation. If a pin is specified as an input port, however, the latched data is not output to the port pin because the output buffer at the pin is in the high-impedance state. (When the pin is switched to the output mode, the contents of the output latch are output to the port pin.) If a pin is specified as an input port, the contents of the output latch for the pin cannot be transferred to the accumulator.

Fig. 5-17 Port Specified as an Input Port



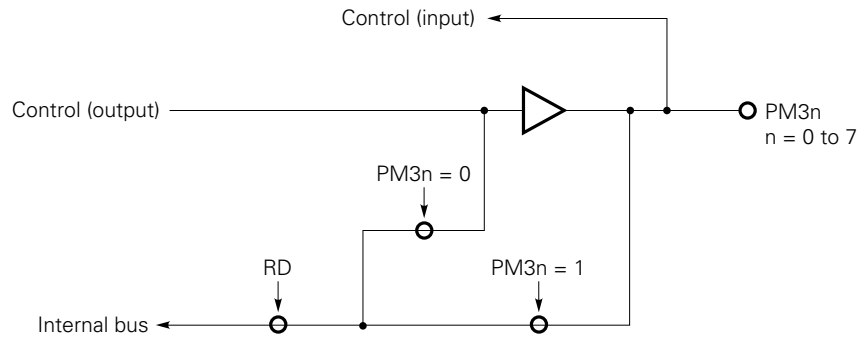
Caution Although its ultimate purpose is to manipulate only 1 bit, a bit manipulation instruction accesses a port in 8-bit units. If a bit manipulation instruction is used for a port some pins of which are in the output mode and the other pins of which are in the input mode, the contents of the output latch corresponding to the pins in the input mode or the control mode become undefined (except for the bits manipulated by the SET1 or CLR1 instruction). Special care should be taken if bits are switched between the input and output modes.

The same holds true when the port is manipulated using 8-bit arithmetic/logical instructions.

(3) Control signal input or output

Regardless of setting of the port mode 3 register (PM3), each bit of port 3 can be used to input or output a control signal, independently of the other bits, by setting the corresponding bit of the port mode control register (PMC3) to 1. When a pin is used for a control signal, executing a read instruction for the port can detect the state of the control signal.

Fig. 5-18 Port Specified as a Control Signal Input or Output



(a) Control signal output

When a bit of the port mode register (PM3n) is 1, executing a read instruction for port 3 can read the level of the corresponding control signal pin.

When a bit of the port mode register (PM3n) is 0, executing a read instruction for port 3 can check the corresponding control signal in the μPD78214.

Remark For bit 3 (P33) of port 3, "SB0" is a reserved word in the NEC assembly program package. The bit is also defined in a header file named sfrbit.h by the C compiler.

(b) Control signal input

Only when a bit of the port mode register (PM3n) is 1, executing a read instruction for port 3 can read the level of the corresponding control signal pin.

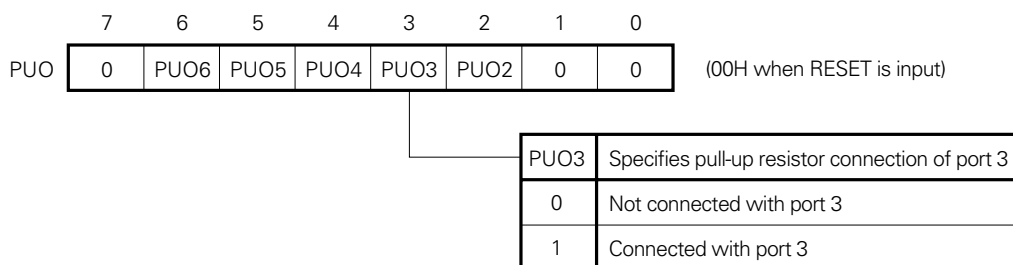
5.4.4 Built-In Pull-Up Resistor

Port 3 has built-in pull-up resistors. When port 3 must be pulled up, the built-in pull-up resistors should be used. Use of the built-in pull-up resistors can reduce the number of the required components and the required installation space.

Use of a built-in pull-up resistor can be specified for each bit of port 3, independently of the other bits, by the PU03 bit of pull-up-resistor-option register (PUO) and the port 3 mode register (PM3). When the PU03 bit is 1, the built-in pull-up resistor for a pin specified by the PM3 (PM3n = 1, n = 0 to 7) is connected.

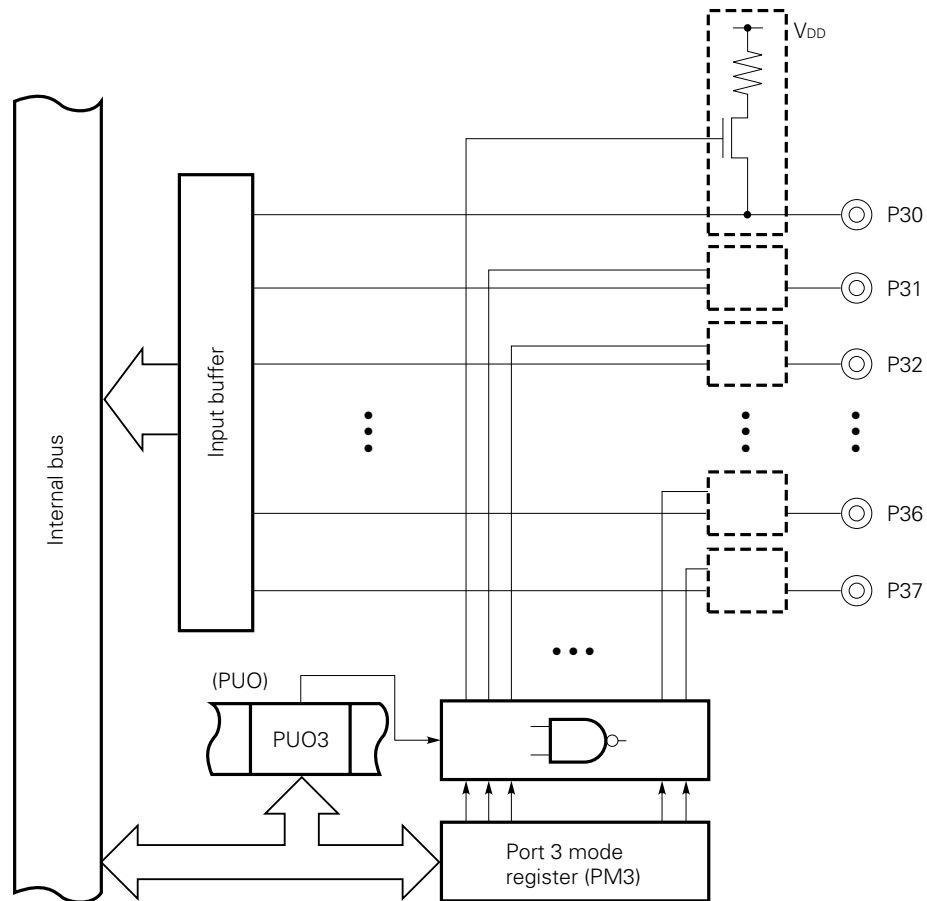
If a pin is specified to be in the control mode, the built-in pull-up resistor for the pin is connected. (A built-in pull-up resistor is connected to a pin that becomes an output pin during the control mode.) If you do not want to use the built-in pull-up resistor for a pin in the control mode, reset the corresponding bit of the PM3 to 0 (output mode).

Fig. 5-19 Pull-Up-Resistor-Option Register Format



Remark Resetting the PU02 bit to 00H can reduce the required current in the STOP mode.

Fig. 5-20 Connection of Pull-Up Resistors (Port 3)



5.5 PORT 4

Port 4 is an 8-bit I/O port with an output latch. The memory expansion mode register (MM) can put all 8 bits of this port in either the input or output mode at one time. Each pin has a software-programmable built-in pull-up resistor, and can drive an LED directly.

When an external memory or I/O device is expanded, port 4 functions as a time-division address/data bus (AD0 through AD7).

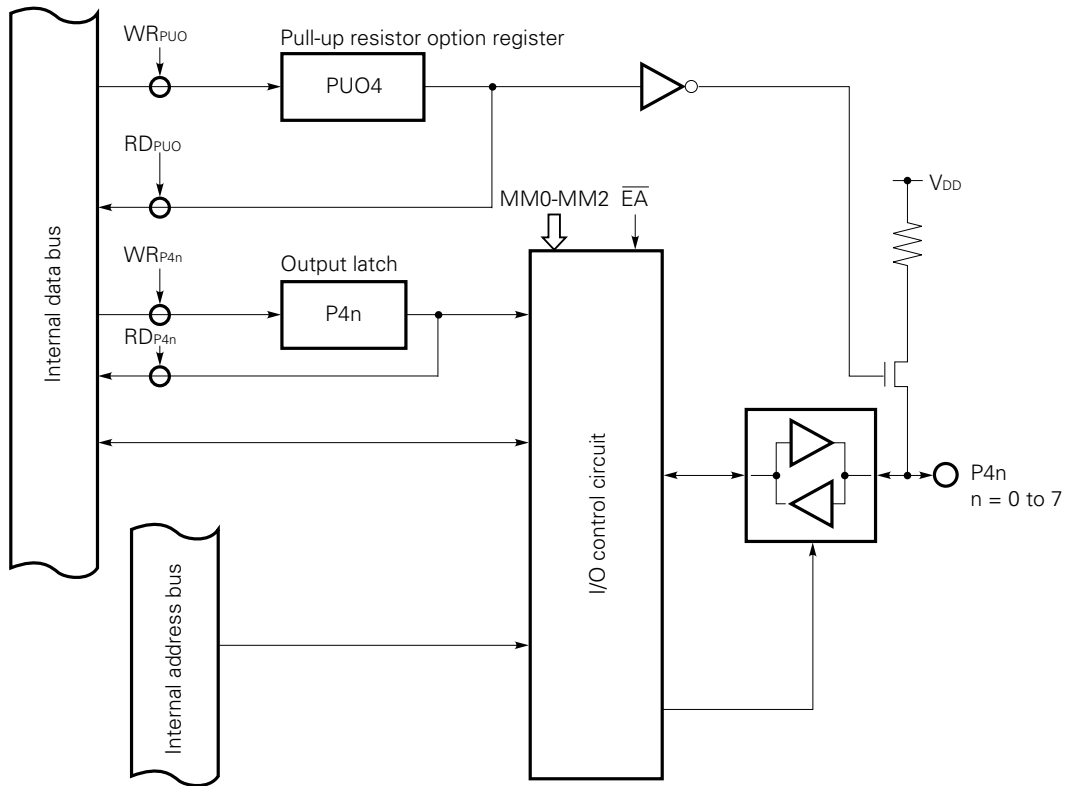
For the μ PD78213, port 4 functions only as a time-division address/data bus (AD0 through AD7).

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

5.5.1 Hardware Configuration

Fig. 5-21 shows the hardware configuration of port 4.

Fig. 5-21 Block Diagram of Port 4



5.5.2 Setting the I/O Mode and Control Mode

The memory expansion mode register (MM, see Fig 13-1) specifies the operating mode of port 4, as listed in Table 5-5.

Table 5-5 Port 4 Operating Modes

EA pin	MM register bit			Operation mode
	MM2	MM1	MM0	
1	0	0	0	Input port
1	0	0	1	Output port
1	1	1	1	Address/data bus (AD0-AD7)
0	×	×	×	

For the μPD78213, port 4 functions only as the address/data bus (AD0 through AD7).

5.5.3 Operation

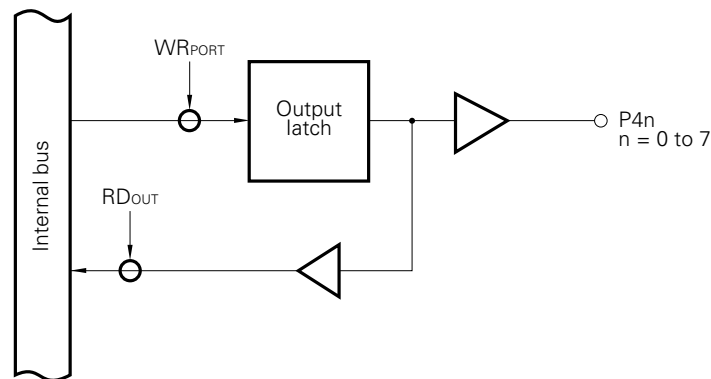
Port 4 is an I/O port. It functions also as an address/data bus (AD0 through AD7).

(1) Output port

When port 4 is in the output mode, its output latch is operable. Once the output latch becomes operable, data can be transferred between the output latch and the accumulator using a transfer instruction. The output latch can be loaded with any data by a logical operation instruction. Once the output latch is loaded with some data, it retains the data until it is loaded with other data.

Note This includes a case in which any other bit of the same port is manipulated using a bit manipulation instruction.

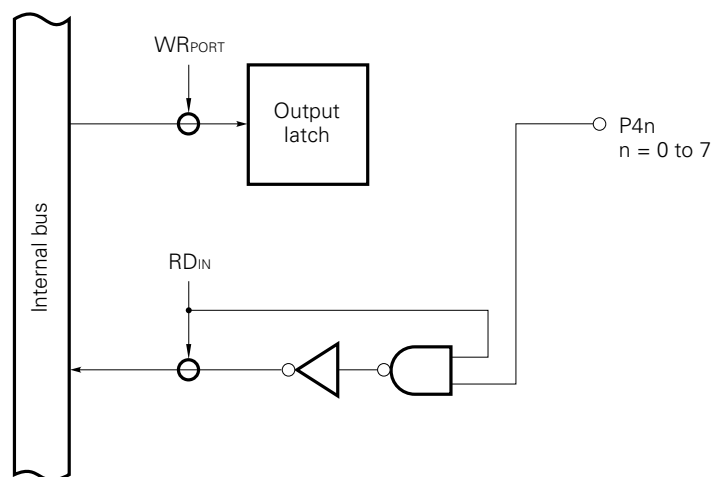
Fig. 5-22 Port Specified as an Output Port



(2) Input port

The level of each pin of port 3 can be transferred to the accumulator by a transfer instruction. Also in this case, data can be written to the output latches, and all output latches hold data transferred from the accumulator by a transfer instruction or other similar instruction, regardless of the current mode of the port operation. If a pin is specified as an input port, however, the latched data is not output to the port pin because the output buffer at the pin is in the high-impedance state. (When the pin is switched to the output mode, the contents of the output latch are output to the port pin.) If a pin is specified as an input port, the contents of the output latch for that pin cannot be transferred to the accumulator.

Fig. 5-23 Port Specified as an Input Port



Caution Although its ultimate purpose is to manipulate only 1 bit, a bit manipulation instruction accesses a port in 8-bit units. If a bit manipulation instruction is used for a port specified to be in the input mode, the contents of the output latch become undefined (except for the bits manipulated by the SET1 or CLR1 instruction). Special care should be taken if bits are switched between the input and output modes. The same holds true when the port is manipulated using 8-bit arithmetic/logical instructions.

(3) Address/data bus (AD0 through AD7)

Port 4 is used as the address/data automatically for external access.

Do not execute I/O instructions for port 4.

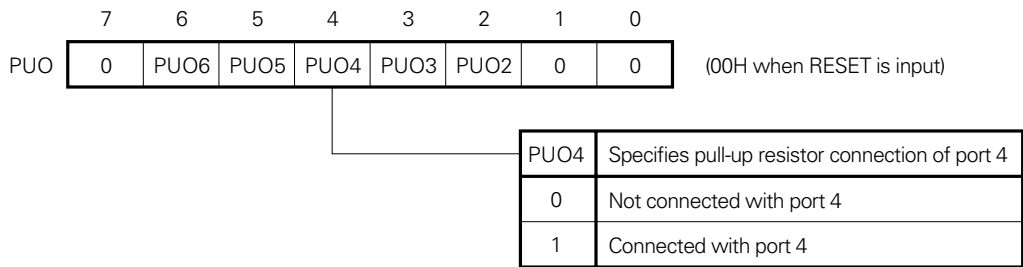
5.5.4 Built-In Pull-Up Resistor

Port 4 has built-in pull-up resistors. When port 4 must be pulled up, the built-in pull-up resistors should be used. Use of the built-in pull-up resistors can reduce the number of the required components and the required installation space.

Use of built-in pull-up resistors can be specified for all 8 bits of port 4 at one time (not independently of each other) by the PU04 bit of the pull-up-resistor-option register (PUO).

Use of built-in pull-up resistors can be specified for this port, regardless of the current mode (input or output) of the port.

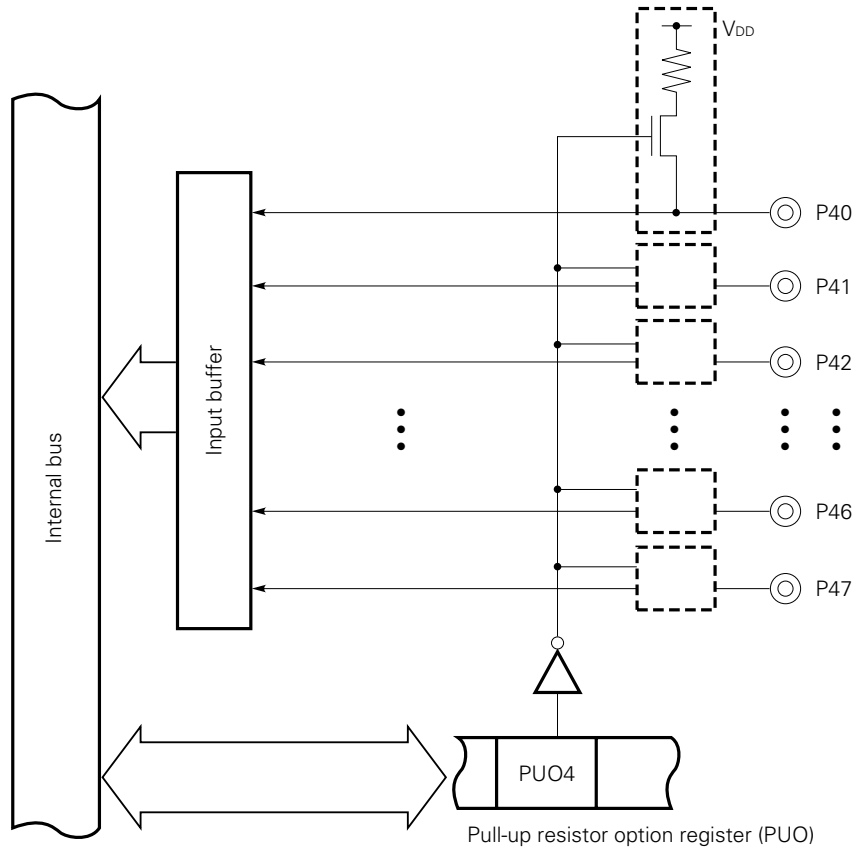
Fig. 5-24 Pull-Up-Resistor-Option Register Format



Caution For the μPD78213, port 4 is used as an address/data bus. Therefore, the PU04 bit must be kept to be 0, and a built-in pull-up resistor must not be connected. For the μPD78214, the same conditions must be maintained if port 4 is used as an address/data bus.

Remark Resetting the PUO to 00H can reduce the required current in the STOP mode.

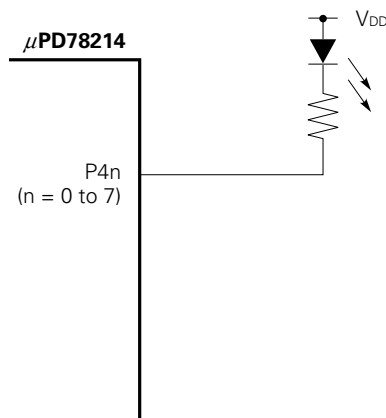
Fig. 5-25 Connection of Pull-Up Resistors (Port 4)



5.5.5 Driving LEDs Directly

For port 4, the low level side of the output buffer has an enhanced driving capacity so that it can drive an LED directly on an active-low signal. Fig. 5-26 is an example of such an output buffer.

Fig. 5-26 Example of Driving an LED Directly



5.6 PORT 5

Port 5 is an 8-bit I/O port with an output latch. The port 5 mode register (PM5) can put each bit of this port in either the input or output mode, independently of the other bits. Each pin has a software-programmable built-in pull-up resistor, and can drive an LED directly.

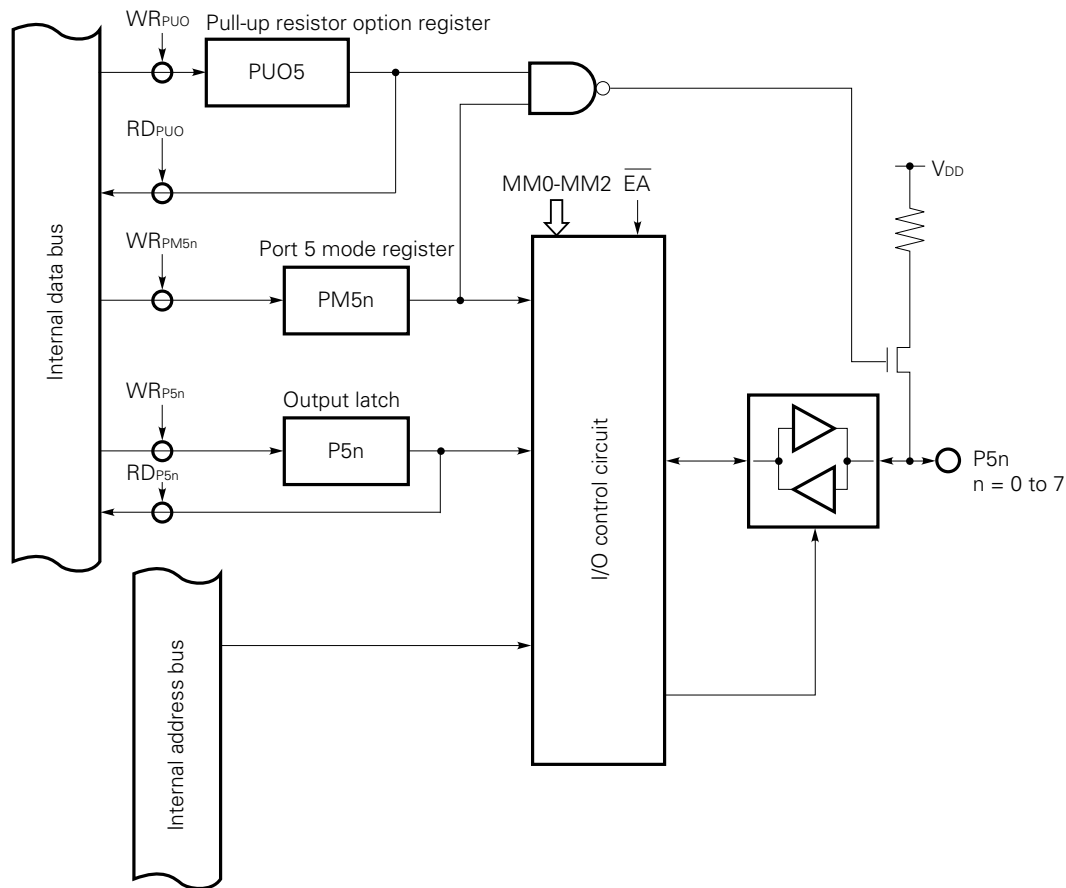
When an external memory or I/O device is expanded, P50 through P57 function as an address bus (AD8 through AD15). For the μPD78213, P50 through P57 function only as an address bus (AD8 through AD15).

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

5.6.1 Hardware Configuration

Fig. 5-27 shows the hardware configuration of port 5.

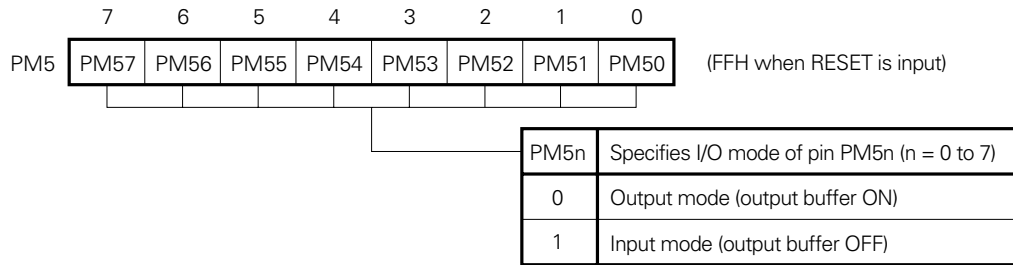
Fig. 5-27 Block Diagram of Port 5



5.6.2 Setting the I/O Mode and Control Mode

The port 5 mode register (PM5) can put each pin of port 5 in either the input or output mode independently of the other pins, as shown in Fig. 5-28. The PM5 register is loaded with data using an 8-bit data transfer instruction; it cannot be bit-manipulated or read-accessed.

The memory expansion mode register (MM, see Fig. 13-1) can specify the control mode of port 5, as listed in Table 5-6.

Fig. 5-28 Port 5 Mode Register Format**Table 5-6 Port 5 Operating Modes**

\overline{EA} pin	MM register bit			Operation mode
	MM2	MM1	MM0	
1	0	0	×	I/O port
1	1	1	1	Address/data bus (A8-A15)
0	×	×	×	

For the μ PD78213, port 4 functions only as the address/data bus (AD8 through AD15).

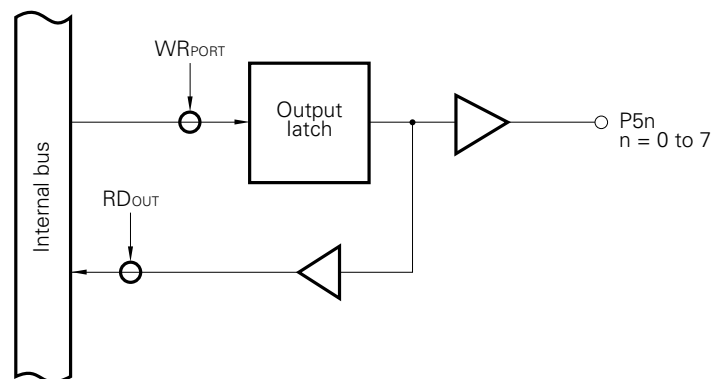
5.6.3 Operation

Port 5 is an I/O port. Its pins also function as control signal pins.

(1) Output port

When port 5 is in the output mode, its output latch is operable. Once the output latch becomes operable, data can be transferred between the output latch and the accumulator using a transfer instruction. The output latch can be loaded with any data by a logical operation instruction. Once the output latch is loaded with some data, it retains the data until it is loaded **Note** with other data.

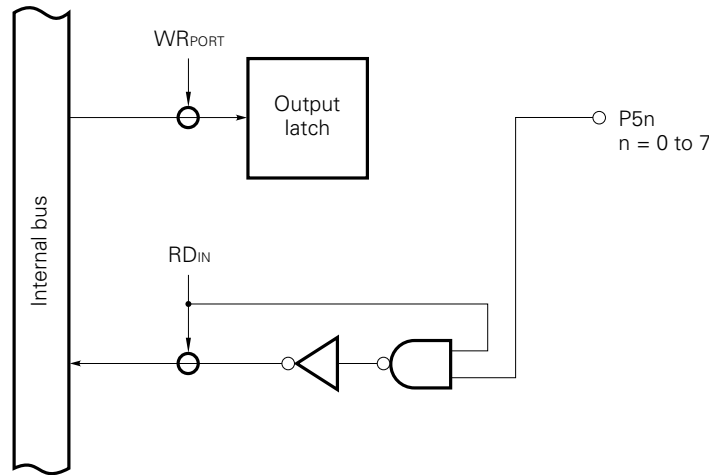
Note This includes a case in which any other bit of the same port is manipulated using a bit manipulation instruction.

Fig. 5-29 Port Specified as an Output Port

(2) Input port

The level of each pin of port 5 can be transferred to the accumulator by a transfer instruction. Also in this case, data can be written to the output latches, and all output latches store data transferred from the accumulator by a transfer instruction or other similar instruction, regardless of the current mode of the port operation. If a bit is specified as an input port, however, the latched data is not output to the port pin because the output buffer at the pin is in the high-impedance state. (When the pin is switched from the input mode to the output mode, the contents of the output latch are output to the port pin.) If a bit is specified as an input port, the contents of the output latch for the pin cannot be transferred to the accumulator.

Fig. 5-30 Port Specified as an Input Port



Caution Although its ultimate purpose is to manipulate only 1 bit, a bit manipulation instruction accesses a port in 8-bit units. If a bit manipulation instruction is used for a port some pins of which are in the output mode and the other pins of which are in the input mode, the contents of the output latch corresponding to the pin in the input mode become undefined (except for the bits manipulated by the SET1 or CLR1 instruction). Special care should be taken if bits are switched between the input and output modes.

The same holds true when the port is manipulated using 8-bit arithmetic/logical instructions.

(3) Address bus (A8 through A15)

Port 5 is used as the address bus automatically for external addresses. Do not execute I/O instructions for port 5.

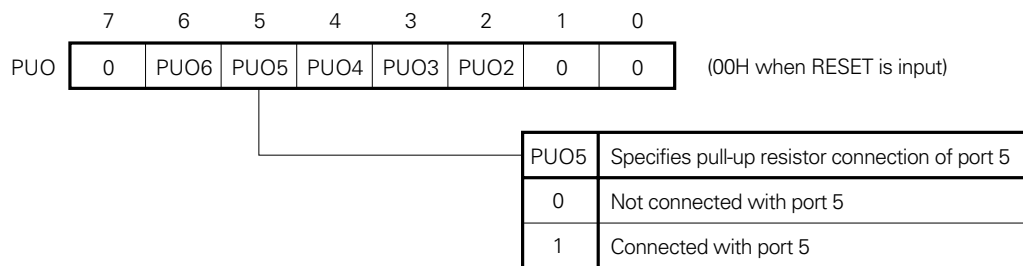
5.6.4 Built-In Pull-Up Resistor

Port 5 has built-in pull-up resistors. When port 5 must be pulled up, the built-in pull-up resistors should be used. Use of the built-in pull-up resistors can reduce the number of the required components and the required installation space.

Use of a built-in pull-up resistor can be specified for each pin of port 5, independently of the other pins, by the PU05 bit of pull-up-resistor-option register (PUO) and the port 5 mode register (PM5).

When the PU05 bit is 1, the built-in pull-up resistor for a pin specified by the PM5 (PM5n = 1, n = 0 to 7) is connected.

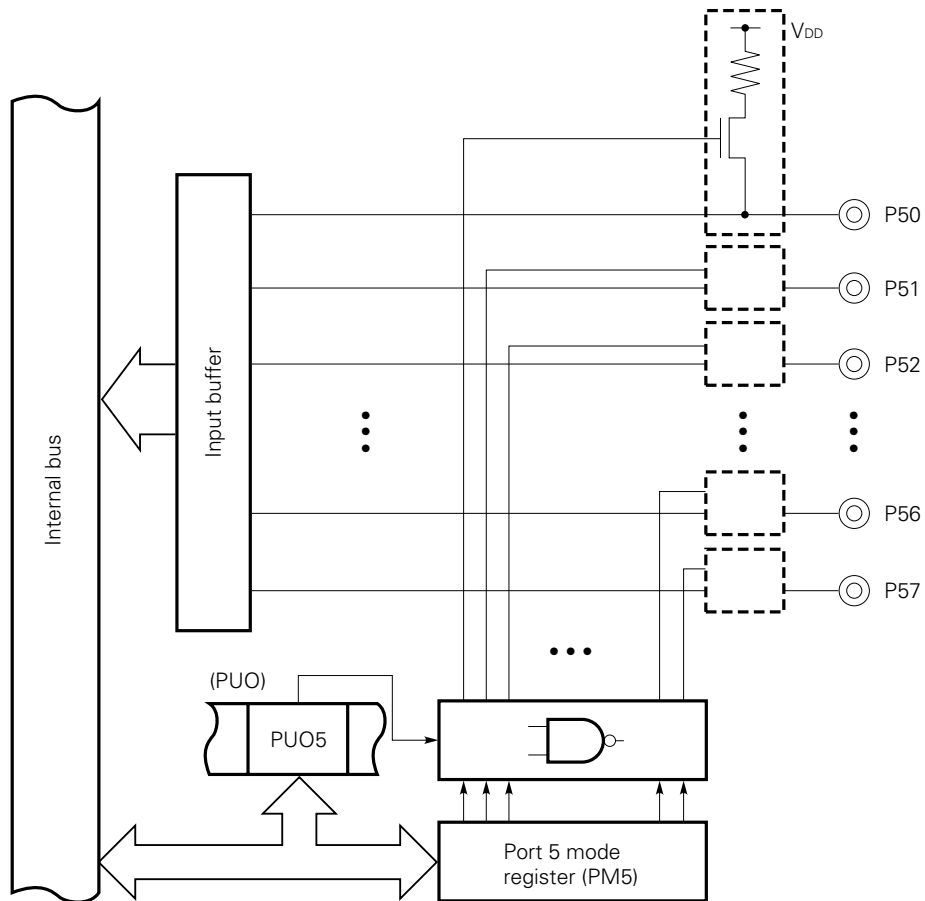
Fig. 5-31 Pull-Up-Resistor-Option Register Format



Caution For the μPD78213, port 5 is used as an address bus. Therefore, the PU05 bit must be kept to be 0, and a built-in pull-up resistor must not be connected. For the μPD78214, the same conditions must be maintained if port 5 is used as an address bus.

Remark Resetting the PUO to 00H can reduce the required current in the STOP mode.

Fig. 5-32 Connection of Pull-Up Resistors (Port 5)

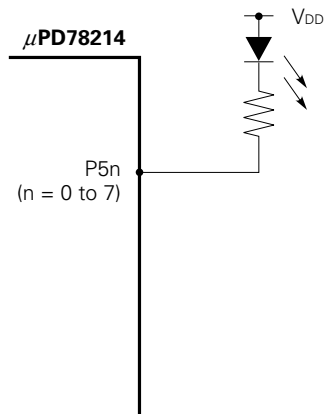


5

5.6.5 Driving LEDs Directly

For port 5, the low level side of the output buffer has an enhanced driving capacity so that it can drive an LED directly on an active-low signal. Fig. 5-33 is an example of such an output buffer.

Fig. 5-33 Example of Driving an LED Directly



5.7 PORT 6

Port 6 is an 8-bit I/O port with an output latch. P64 through P67 have a software-programmable built-in pull-up resistor.

In addition to the port functions, port 5 works as I/O pins for various control signals as listed in Table 5-7. Each control pin is operated by the corresponding function.

For the μPD78213, P64 and P65 function only as \overline{RD} and \overline{WR} output pins, respectively.

When the \overline{RESET} signal is input, P60 through P63 go low, and P64 through P67 are put in the input port mode (high output impedance state). At the same time, the contents of the higher 4 bits of the output latch become undefined, and the lower 4 bits are reset to 0H.

Table 5-7 Port 6 Operating Modes

Pin	Port mode	Control signal I/O mode	Operation needed to make the pin function as a control signal
P60-P63	Output port	A16-A19 output	Set the MM6 bit of the MM register to 1.
P64	I/O port	\overline{RD} output	For the μPD78213, no special operation is needed. For the other models, specify the memory expansion mode by the MM2 through MM0 bits of the MM register.
P65		\overline{WR} output	
P66		\overline{WAIT} input/AN6 input	Specified by the PW register, or the PWN1 and PWN0 bits (n = 2 or 3) of the MM register or by putting P66 in the input mode.
P67		\overline{REFRQ} output/AN7 input	Set the RFEN bit of the RFM register to 1.

Caution When the \overline{RESET} signal exists, P60 through P63 are kept in the high impedance state. When the \overline{RESET} signal disappears, they output a low level. So, it is necessary to design the external circuit so that P60 through P63 are allowed to output a low level during the initial state.

Remark See Chapter 13 for details.

(a) Port mode

P60 through P63 are output-only circuits. Each of P64 through P67 can be specified to be in either the input or output mode by the port 6 mode register (PM6), independently of the other bits.

(b) Control signal I/O mode

(i) A16 through A19 (address bus)

These pins function as the upper address bus output pins when the external memory space is expanded (10000H through FFFFFH). They operate according to the setting of the memory expansion register (MM).

(ii) \overline{RD} (read strobe)

This pin outputs a strobe signal to read from external memory. For the μPD78213, it always operates. For the other models, it operates when the external memory is expanded.

(iii) \overline{WR} (write strobe)

This pin outputs a strobe signal to write to external memory. For the μPD78213, it always operates. For the other models, it operates according to the setting of the MM register.

(iv) \overline{WAIT} (wait)

This pin receives a wait signal. It operates according to the setting of the programmable wait control (PW) register or the MM register.

(v) \overline{REFRQ} (refresh request)

When a pseudo-static memory is connected to externally, this pin outputs a refresh pulse to the pseudo-static memory. It operates according to the setting of the refresh mode register (RFM).

(vi) AN6 and AN7 (analog input)

These pins receive analog signals for the A/D converter.

5.7.1 Hardware Configuration

Fig. 5-34 through 5-37 show the hardware configuration of port 6.

Fig. 5-34 Block Diagram of P60 through P63 (Port 6)

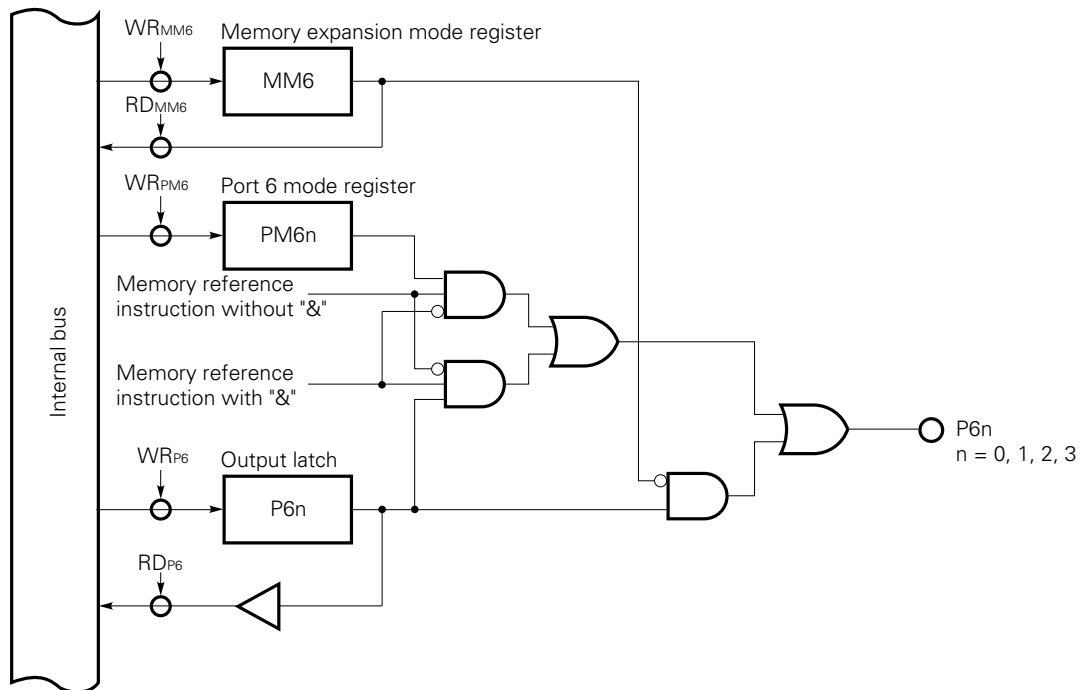


Fig. 5-35 Block Diagram of P64 and P65 (Port 6)

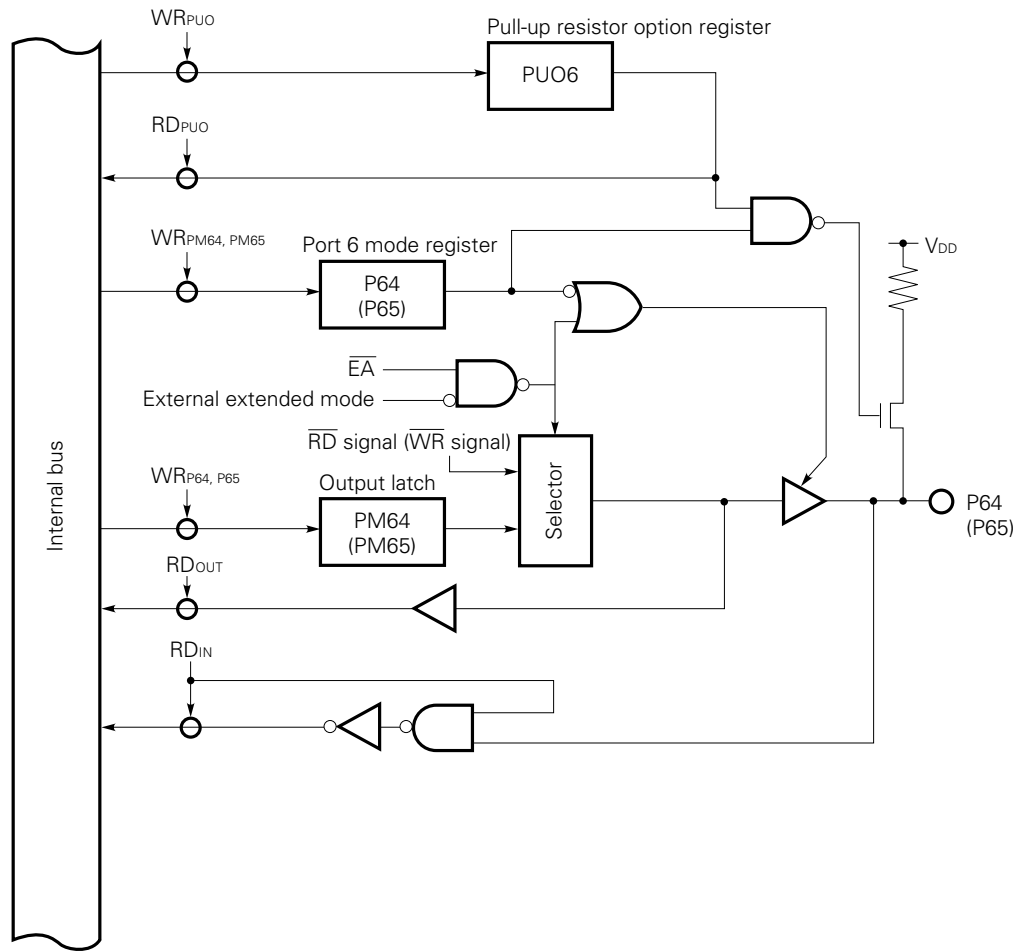


Fig. 5-36 Block Diagram of P66 (Port 6)

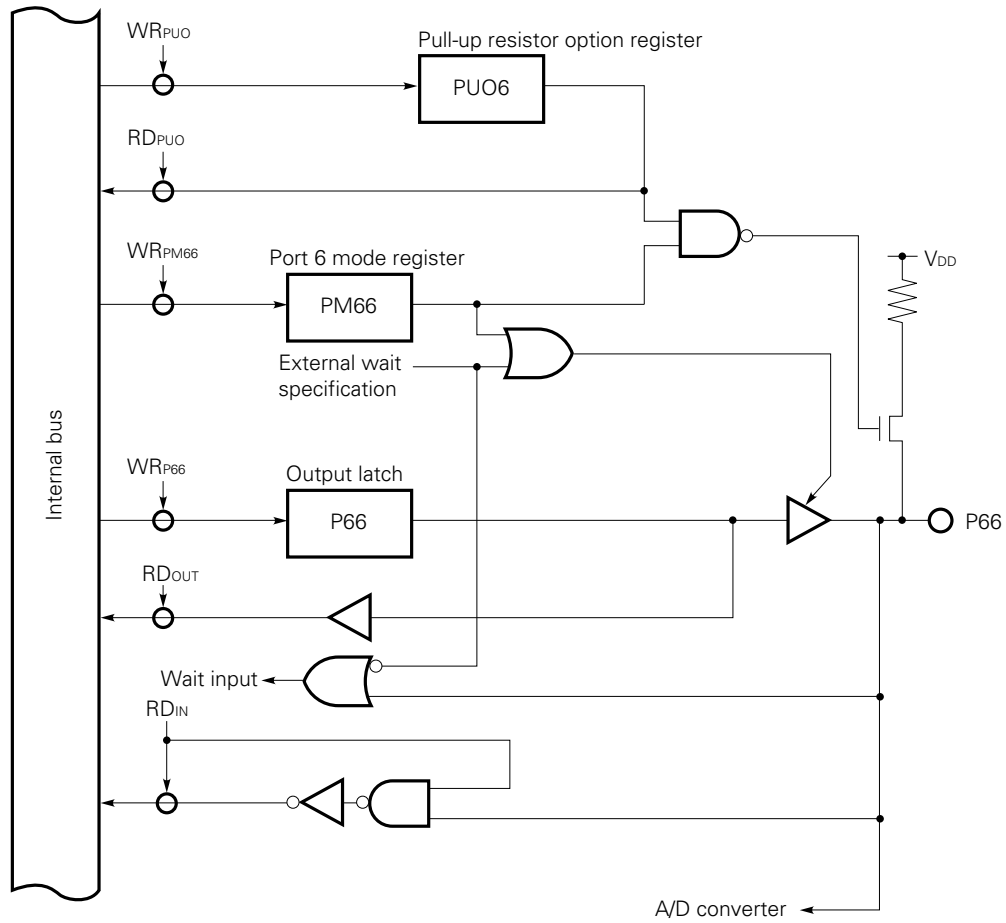
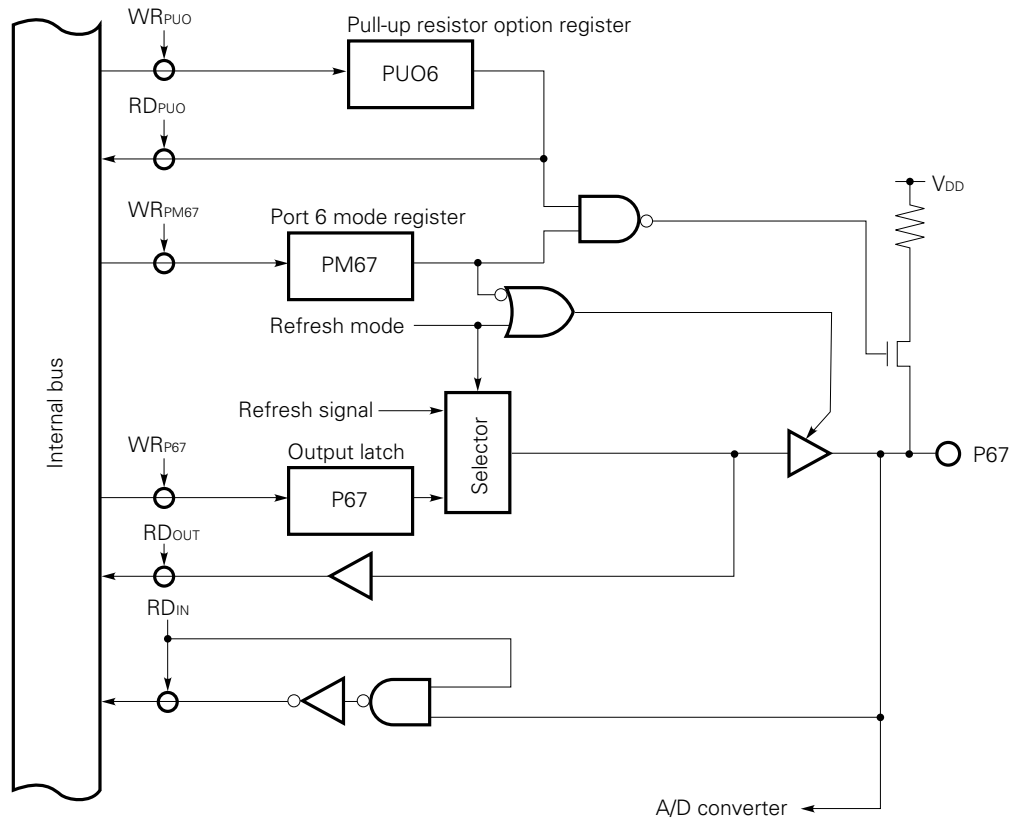


Fig. 5-37 Block Diagram of P67 (Port 6)



5.7.2 Setting the I/O Mode and Control Mode

The port 6 mode register (PM6) can put port 6 in either the input or output mode as shown in Fig. 5-38. Table 5-8 lists the operations needed to make port 6 function as control pins.

P66 and P67 can always receive analog signals.

The ADM of the A/D converter specifies the mode of the A/D converter operation. (See **Chapter 8.**) To use port 6 as AN6 and AN7, however, it is necessary to specify the input mode using the port 6 mode register (PM6) and disconnection of a pull-up resistor using the PUO6 bit (= 0) of the pull-up-resistor-option register (PUO).

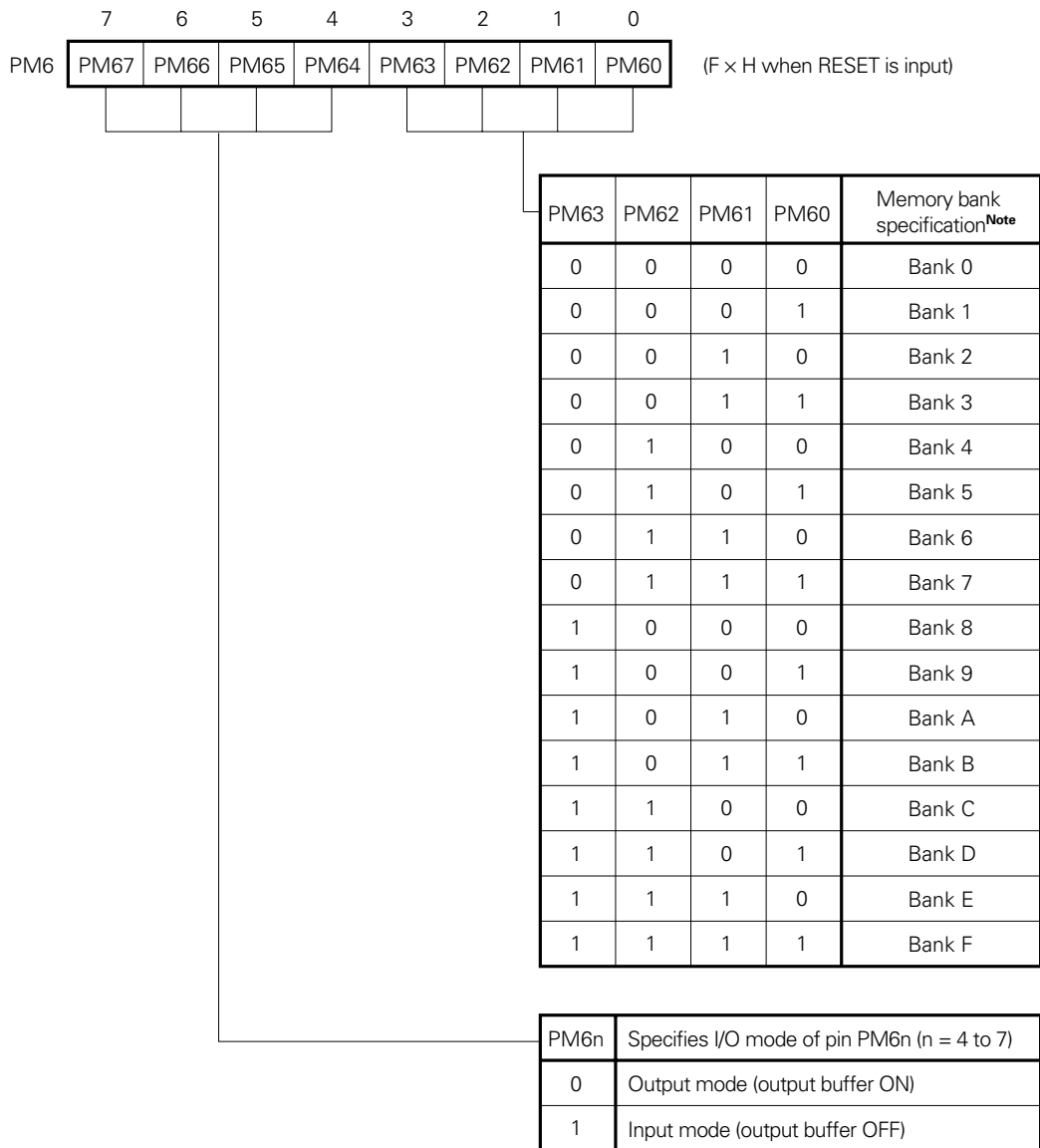
For the μPD78213, P64 and P65 function only as output pins for \overline{RD} and \overline{WR} , respectively.

Table 5-8 Port 6 Control Pin Functions and the Required Operations

Pin	Function	I/O	Operation needed to make port 6 function as control pins
P60	A16	Output	Set the MM6 bit of the MM register to 1.
P61	A17	Output	
P62	A18	Output	
P63	A19	Output	
P64	\overline{RD}	Output	For the μPD78213, no special operation is needed. For the other models, specify the memory expansion mode by the MM2 through MM0 bits of the MM register.
P65	\overline{WR}	Output	
P66	\overline{WAIT}	Input	Specify insertion of external wait using the PW register or the PWN1 and PWN0 bits (n = 2 or 3) of the MM register or by putting P66 in the input mode.
P67	\overline{REFRQ}	Output	Set the RFEN bit of the RFM register to 1.

- Cautions**
1. To use P60 through P63 as an output port, it is necessary to reset the PM60 through PM63 bits to 0. If they are not 0, the in-circuit emulator may not work.
 2. To use the P66/ $\overline{\text{WAIT}}$ pin as the $\overline{\text{WAIT}}$ pin, it is necessary to put P66 in the input mode using the PM6 register.

Fig. 5-38 Port 6 Mode Register Format



Note When the MM6 bit of the memory expansion mode register (MM) is set to 1, these bits function as a bank specification register used to execute memory reference instructions without "&". When the 1M-byte expansion function is not used, reset PM63 through PM60 to all 0s.

Caution To use analog inputs AN6 and AN7, put PM66 and PM67 in the input mode.

Remark The lower four bits (P60 through P63) of port 6 are an output-only port.

5.7.3 Operation

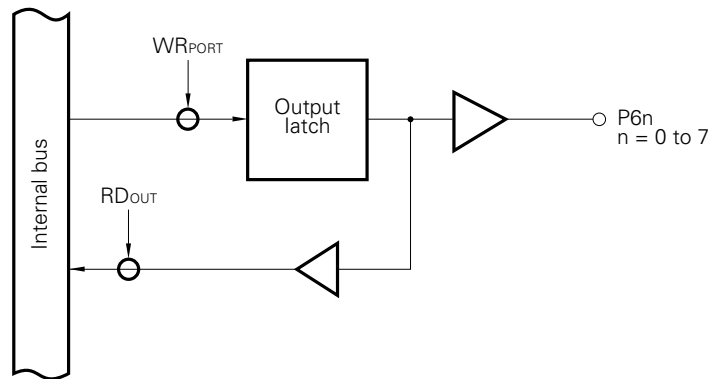
Port 6 is an I/O port. Its pins also function as control signal pins.

(1) Output port

When port 6 is in the output mode, the contents of its output latch are output, and data can be transferred between the output latch and the accumulator using a transfer instruction. The output latch can be loaded with any data by a logical operation instruction. Once the output latch is loaded with some data, it retains the data until it is loaded with other data.

Note This includes a case in which any other bit of the same port is manipulated using a bit manipulation instruction.

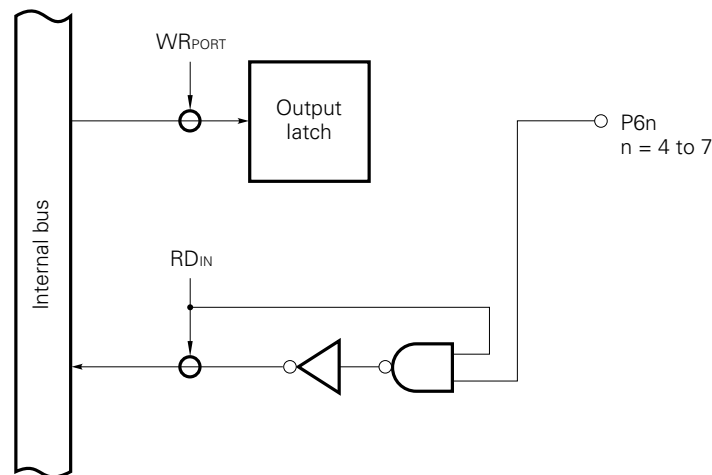
Fig. 5-39 Port Specified as an Output Port



(2) Input port

The level of each pin of port 3 can be transferred to the accumulator by a transfer instruction. Also in this case, data can be written to the output latches, and all output latches store data transferred from the accumulator by a transfer instruction or other similar instruction, regardless of the current mode of the port operation. If a pin is specified as an input port, however, the latched data is not output to the port pin because the output buffer at the pin is in the high-impedance state. (When the bit is switched from the input mode to the output mode, the contents of the output latch are output to the corresponding port pin.) If a bit is specified as an input port, the contents of the output latch for the bit cannot be transferred to the accumulator.

Fig. 5-40 Port Specified as an Input Port



Caution Although its ultimate purpose is to manipulate only 1 bit, a bit manipulation instruction accesses a port in 8-bit units. If a bit manipulation instruction is used for a port some pins of which are in the output mode and the other pins of which are in the input mode, the contents of the output latch corresponding to the pins in the input mode become undefined (except for the bits manipulated by the SET1 or CLR1 instruction). Special care should be taken if bits are switched between the input and output modes.

The same holds true when the port is manipulated using 8-bit arithmetic/logical instructions.

(3) Control pins

When port 6 function as control pins, they cannot be manipulated or tested by software.

(4) Analog inputs (P66 and P67 only)

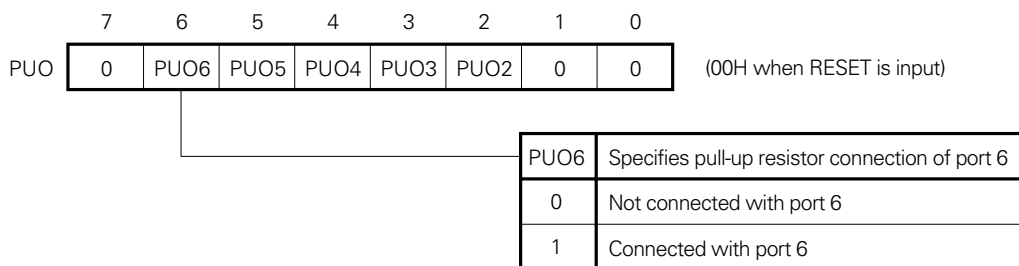
When port 6 is used as analog input pins (AN6 and AN7), the level of each pin can be read and tested.

5.7.4 Built-In Pull-Up Resistor

P64 through P67 have built-in pull-up resistors. When they must be pulled up, the built-in pull-up resistors should be used. Use of the built-in pull-up resistors can reduce the number of the required components and the required installation space.

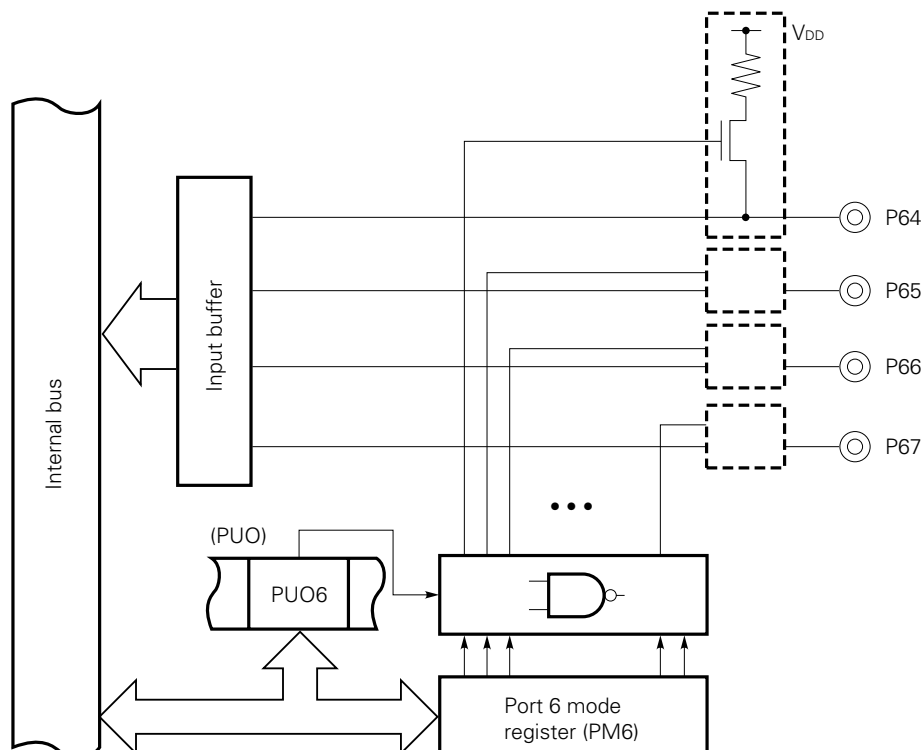
Use of a built-in pull-up resistor can be specified for each of these pins, independently of the other pins, by the PUO6 bit of pull-up-resistor-option register (PUO) and the port 6 mode register (PM6).

When the PUO6 bit is 1, the built-in pull-up resistor for a pin specified by the PM6 ($PM6n = 1, n = 4$ to 7) is connected. P60 through P63 do not have a built-in pull-up resistor.

Fig. 5-41 Pull-Up-Resistor-Option Register Format

Remark Resetting the PUO to 00H can reduce the required current in the STOP mode.

Caution To use P66 and P67 as AN6 and AN7, respectively, it is necessary to reset the PUO6 bit to 0; do not specify to use built-in pull-up resistors for port 6.

Fig. 5-42 Connection of Pull-Up Resistors (Port 6)

5.7.5 Note

When P66 and P67 are used as analog input pins AN6 and AN7 respectively or when A/D conversion is not performed, do not apply a voltage out of the range AV_{SS} through AV_{REF} to these pins, if AN6 and AN7 are selected for ANI0 through ANI2 of the A/D converter mode register (ADM).

See **Chapter 8** for details.

5.8 PORT 7

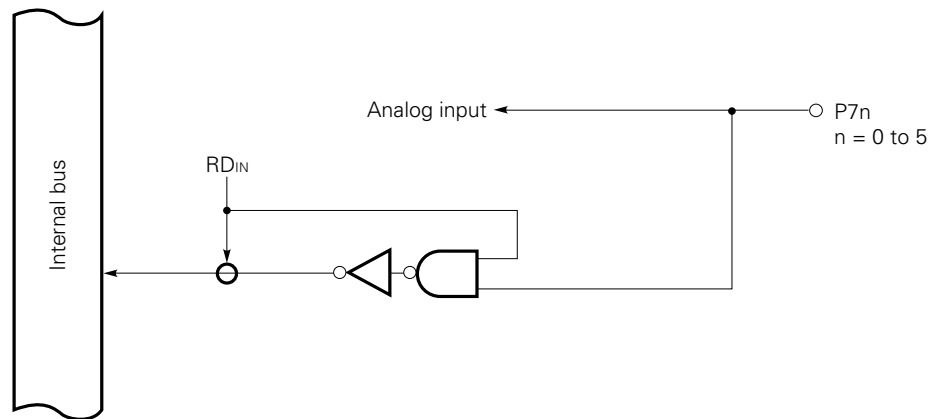
Port 7 is a 6-bit input-only port. It functions as A/D converter analog input pins (AN0 through AN5) as well as input port pins.

The level of each pin of port 7 can be read and tested, although it is a dual-function pin.

5.8.1 Hardware Configuration

Fig. 5-43 shows the hardware configuration of port 7.

Fig. 5-43 Block Diagram of Port 7



5.8.2 Setting the I/O Mode and Control Mode

Port 7 is an input-only port.

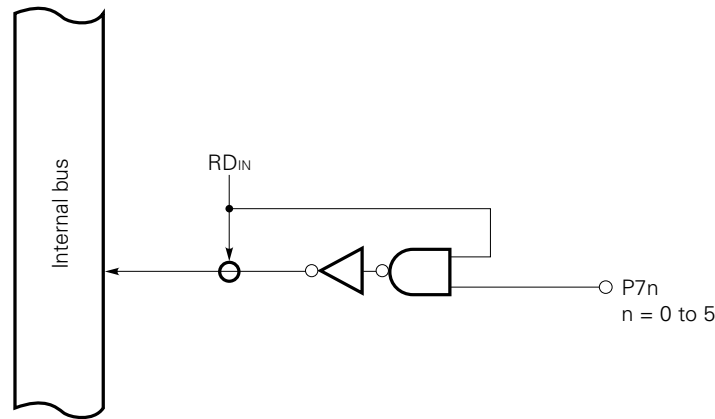
Port 7 is designed so that it can always receive analog inputs. Therefore, it is unnecessary to specify a mode.

The ADM register for the A/D converter is used to specify the mode of A/D converter operation. (See **Chapter 8** for details.)

5.8.3 Operation

Port 7 is an input-only port, and the level of its pins can be read and tested.

Fig. 5-44 Port Specified as an Input Port



5.8.4 Built-In Pull-Up Resistor

Port 0 has no built-in pull-up resistor.

5.8.5 Notes

- (1) When P70 through P75 are used as analog input pins AN0 and AN5 respectively or when A/D conversion is not performed, do not apply a voltage out of the range AV_{SS} through AV_{REF} to the pins that are selected for ANI0 through ANI2 of the A/D converter mode register (ADM).

See **Chapter 8** for details.

- (2) Port 7 is a 6-bit input port. The upper 2 bits of "8-bit" data from port 7 are undefined.

5.9 NOTES

- (1) When the $\overline{\text{RESET}}$ signal is input, all the port pins are set to high-impedance (disconnected from their built-in pull-up resistors).

If it is necessary to prevent a port pin from being set to high-impedance state when the $\overline{\text{RESET}}$ signal is being supplied, take an appropriate action using an external circuit.

- (2) Some operations of the pull-up-resistor-option register (PUO), which is used to specify connection of built-in pull-up resistors, cannot be emulated by an in-circuit emulator, because of the following restrictions:

- The contents of the PUO register cannot be read correctly.
- Bit manipulation instructions or logical/arithmetic instructions do not work for the PUO register; an SFR illegal access break may occur, aborting emulation.
- No built-in pull-up resistor is provided.

Therefore, observe the following two points.

- To set the PUO register, use only an 8-bit data transfer instruction (MOV).
- When debugging the target board, use pull-up resistors provided on it.

- (3) Supplying the $\overline{\text{RESET}}$ signal does not initialize the contents of the output latch. When using a port in the output mode, initialize the output latch before turning on the output buffer. Otherwise, data output from the output ports is unpredictable.

Similarly, to use a port as a control pin, always initialize the internal hardware before specifying the control pin.

- (4) P22 through P26 are not pulled up immediately after a reset, and the interrupt request flag may be set depending on the function of a dual-function pin (INTP1 through INTP5). Therefore, specify connection of a pull-up resistor in the initialization routine, before clearing the interrupt request flag.
- (5) With an in-circuit emulator, the level of each pin of port 2 can be read and tested before noise is removed.
- (6) For the μ PD78214, to use P40 through P47 and P50 through P57 as an address/data bus and an address bus respectively, always reset the PU04 and PU05 bits of the PU0 register to 0 so that a built-in pull-up resistor is not used.

For the μ PD78213, P40 through P47 and P50 through P57 are always used as an address/data bus and an address bus respectively, and the PU04 and PU05 bits of the PU0 register must always be reset to 0, and a built-in pull-up resistor must not be connected.

- (7) To use P60 through P63 as an output port, always reset to the PM60 through P63 bits to 0. If they are not 0, P60 through P63 cannot be emulated normally by an in-circuit emulator.
- (8) To use analog inputs AN6 and AN7, put P66 and P67 in the input mode, respectively.
- (9) To use P66 and P67 as AN6 and AN7 respectively, reset the PU06 to 0; do not specify connection of built-in pull-up resistors for port 6.
- (10) When P66 and P67 are used as analog input pins AN6 and AN7 respectively or when A/D conversion is not performed, do not apply a voltage out of the range AV_{SS} through AV_{REF} to these pins, if AN6 and AN7 are selected for ANI0 through ANI2 of the A/D converter mode register (ADM).

See **Chapter 8** for details.

- (11) To use the P66/ \overline{WAIT} pin as the \overline{WAIT} pin, it is necessary to put P66 in the input mode using the PM6 register.
- (12) When P70 through P75 are used as analog input pins AN0 and AN5 respectively or when A/D conversion is not performed, do not apply a voltage out of the range AV_{SS} through AV_{REF} to the pins that are selected for ANI0 through ANI2 of the A/D converter mode register (ADM).

See **Chapter 8** for details.

- (13) Although its ultimate purpose is to manipulate only 1 bit, a bit manipulation instruction accesses a port in 8-bit units. If a bit manipulation instruction is used for a port some pins of which are in the output mode and the other pins of which are in the input mode, the contents of the output latch corresponding to the pins in the input mode or the control mode become undefined (except for the bits manipulated by the SET1 or CLR1 instruction). Special care should be taken if bits are switched between the input and output modes. The same applies when the port is manipulated using 8-bit arithmetic/logical instructions.
- (14) Port 7 is a 6-bit input port. The upper 2 bits of "8-bit" data from port 7 are undefined.

CHAPTER 6 REAL-TIME OUTPUT FUNCTION

6.1 CONFIGURATION AND FUNCTION

The real-time output function is implemented by the hardware centering around port 0 and the buffer register (POH and POL) as shown in Fig. 6-1.

The term real-time output function refers to a function that transfers data in the buffer register to the output latch by hardware for output to the outside simultaneously when a timer interrupt or external interrupt occurs. The term real-time output port refers to a pin used to output such data to the outside.

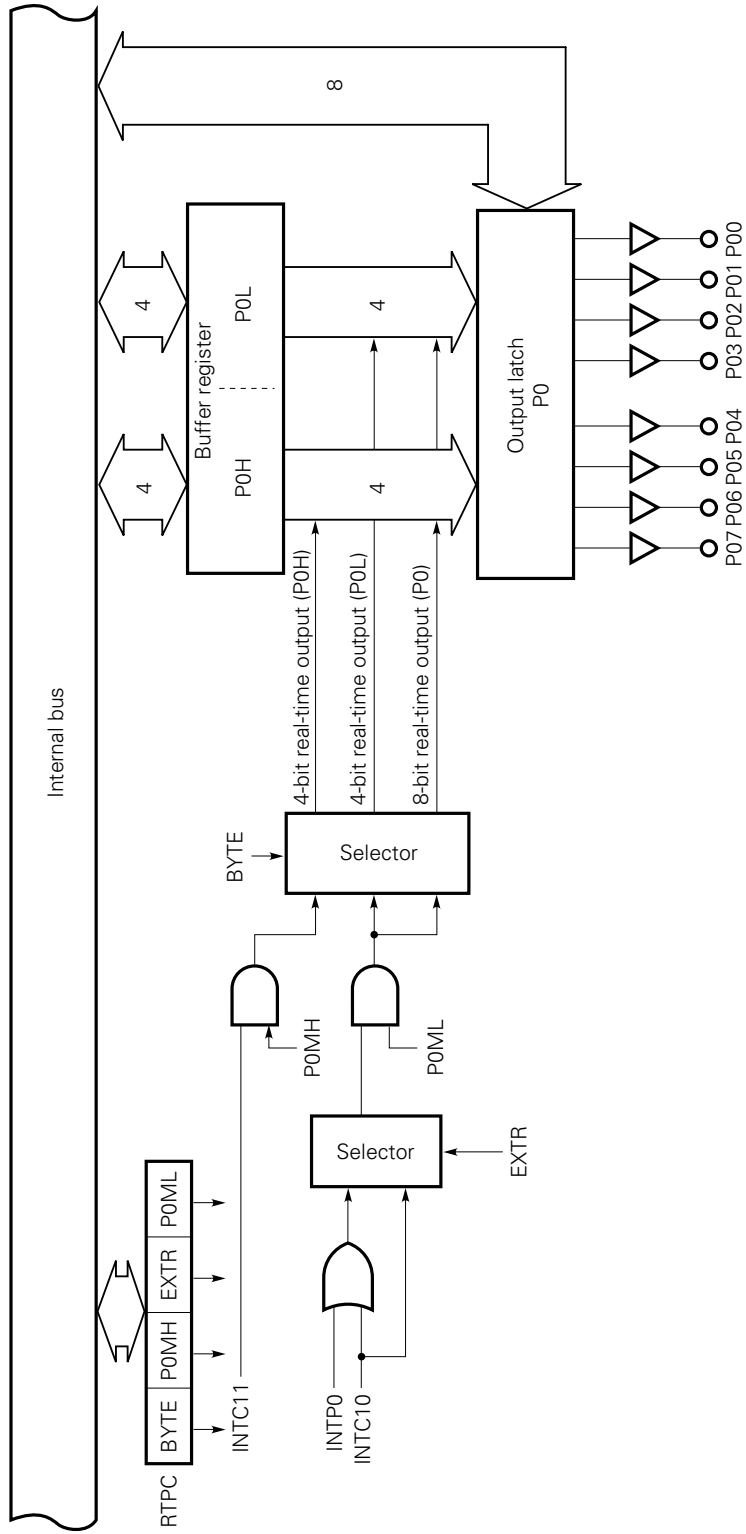
The real-time output function handles the following two types of real-time output data:

- 4 bits × 2 channels
- 8 bits × 1 channel

Combined use of the real-time output function and the macro service function (described later) implements a pattern generator function with programmable timing without intervention by software.

The pattern generator function is suitable to control stepper motors.

Fig. 6-1 Block Diagram of the Real-Time Output Port

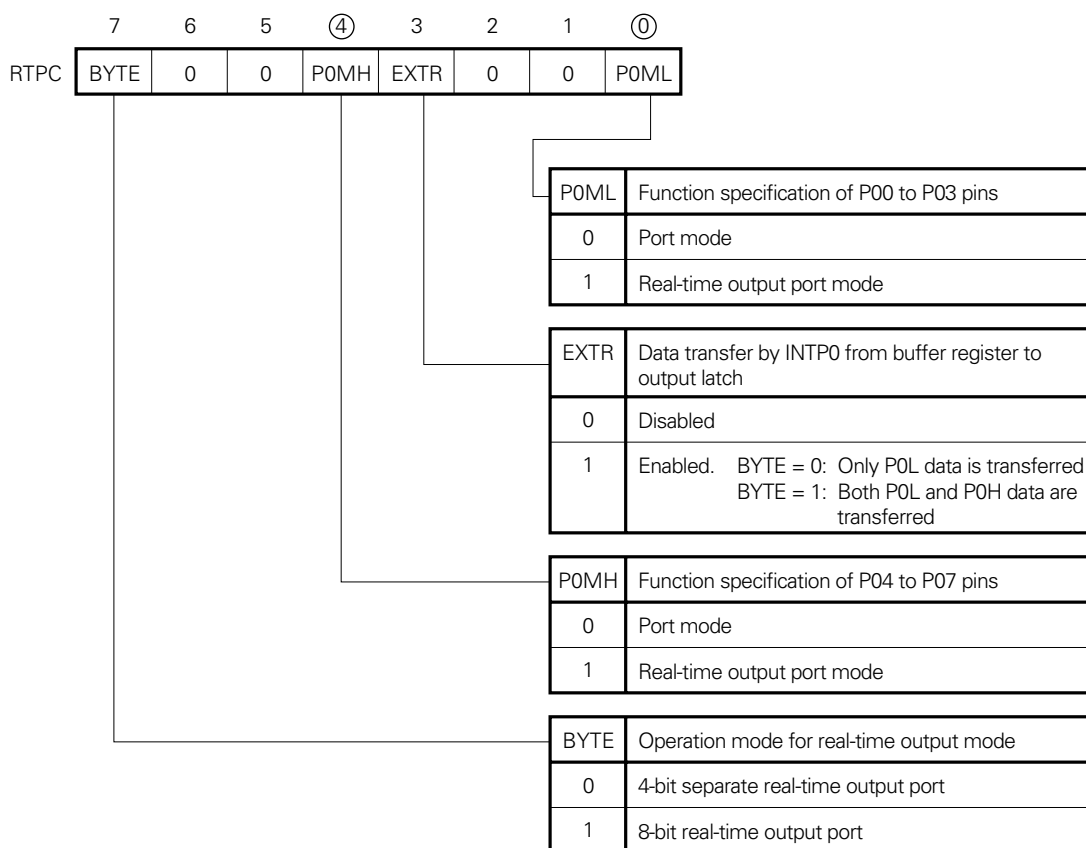


6.2 REAL-TIME OUTPUT CONTROL REGISTER (RTPC)

The real-time output control register (RTPC) is an 8-bit register to specify the functions of port 0. An 8-bit manipulation instruction and a bit manipulation instruction can be used to read data from and write data to the RTPC register. Fig. 6-2 shows the format of this register.

When the $\overline{\text{RESET}}$ signal is input, the RTPC is reset to 00H.

Fig. 6-2 Real-Time Output Port Control Register (RTPC) Format



Caution When the P0ML or P0MH is set to 1, the output buffer for the corresponding output port is turned on to output the contents of the port 0 output latch, regardless of the contents of the port 0 mode register (PM0). Therefore, initialize the contents of the output latch before specifying the real-time output port.

6.3 ACCESS TO THE REAL-TIME OUTPUT PORT

The buffer registers (P0H and P0L) are mapped to independent addresses in the SFR area, as shown in Fig. 6-3. When the 4-bit \times 2-channel real-time output function is specified, data can be set in each buffer register (P0H or P0L) independently.

When the 8-bit \times 1-channel real-time output function is specified, 8-bit data can be set in the buffer registers by writing to either one (P0H or P0L).

Table 6-1 lists the operating modes of port 0 and the operations needed for the port 0 buffer registers.

Fig. 6-3 Configuration of the Buffer Registers (P0H and P0L)

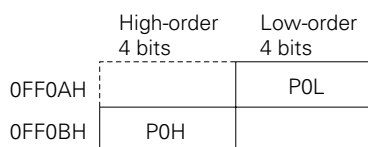


Table 6-1 Port 0 Operating Modes and Operations Needed for the Port 0 Buffer Registers

Operating mode	Register	Read operation		Write operation	
		High-order 4 bits	Low-order 4 bits	High-order 4 bits	Low-order 4 bits
8-bit port mode	P0	Output latch		Output latch	
	P0L	Buffer register ^{Note}		—	Buffer register
	P0H	Buffer register ^{Note}		Buffer register	—
8-bit real-time output port mode	P0	Output latch		—	
	P0L	Buffer register ^{Note}		Buffer register	
	P0H	Buffer register ^{Note}		Buffer register	
4-bit separate real-time output port mode	P0	Output latch		—	
	P0L	Buffer register ^{Note}		—	Buffer register
	P0H	Buffer register ^{Note}		Buffer register	—
P00-P03: Port P04-P07: Real-time output port mode	P0	Output latch		—	Output latch
	P0L	Buffer register ^{Note}		—	Buffer register
	P0H	Buffer register ^{Note}		Buffer register	—
P00-P03: Real-time output port mode P04-P07: Port	P0	Output latch		Output latch	—
	P0L	Buffer register ^{Note}		—	Buffer register
	P0H	Buffer register ^{Note}		Buffer register	—

Note The contents of the P0H are read to the high-order 4 bits, and the contents of the P0L are read to the low-order 4 bits.

Remark —: The output latches or buffer registers are not affected.

Example of setting data in the buffer registers

- 4-bit × 2-channel operation
 - MOV P0L, #05H ; Sets 0101B in the P0L register
 - MOV P0H, #0C0H ; Sets 1100B in the P0H register
- 8-bit × 1-channel operation
 - MOV P0L, #0C5H ; Sets 0101B in the P0L register
and 1100B in the P0H register

Or,

MOV P0H, #0C5H

The following three sources can determine the timing at which data is output to an output latch.

- Interrupt from 8-bit timer/counter 1 (INTC10 or INTC11)
- INTP0 external interrupt

6.4 OPERATION

When port 0 is in the real-time output port mode, the contents of the buffer registers (P0H and P0L) are sent to the output latches for output to the pins of port 0 in synchronization with the occurrence of a trigger condition listed in Table 6-2.

For example, let's select, as an output trigger source, a signal (INTC10 or INTC11) indicating that timer 1 (TM1) of 8-bit timer/counter 1 coincides with the compare register (CR10 or CR11). In this case, the output data at the pins of port 0 can be made to reflect the contents of the buffer register at intervals of a value preset in each compare register. Combined use of the real-time output port function and the macro service function can output data at each pin of port 0 sequentially at arbitrary intervals (see **Section 12.4**).

If external interrupt INTP0 is selected as an external output trigger source, data can be output from port 0 in synchronization with an external event.

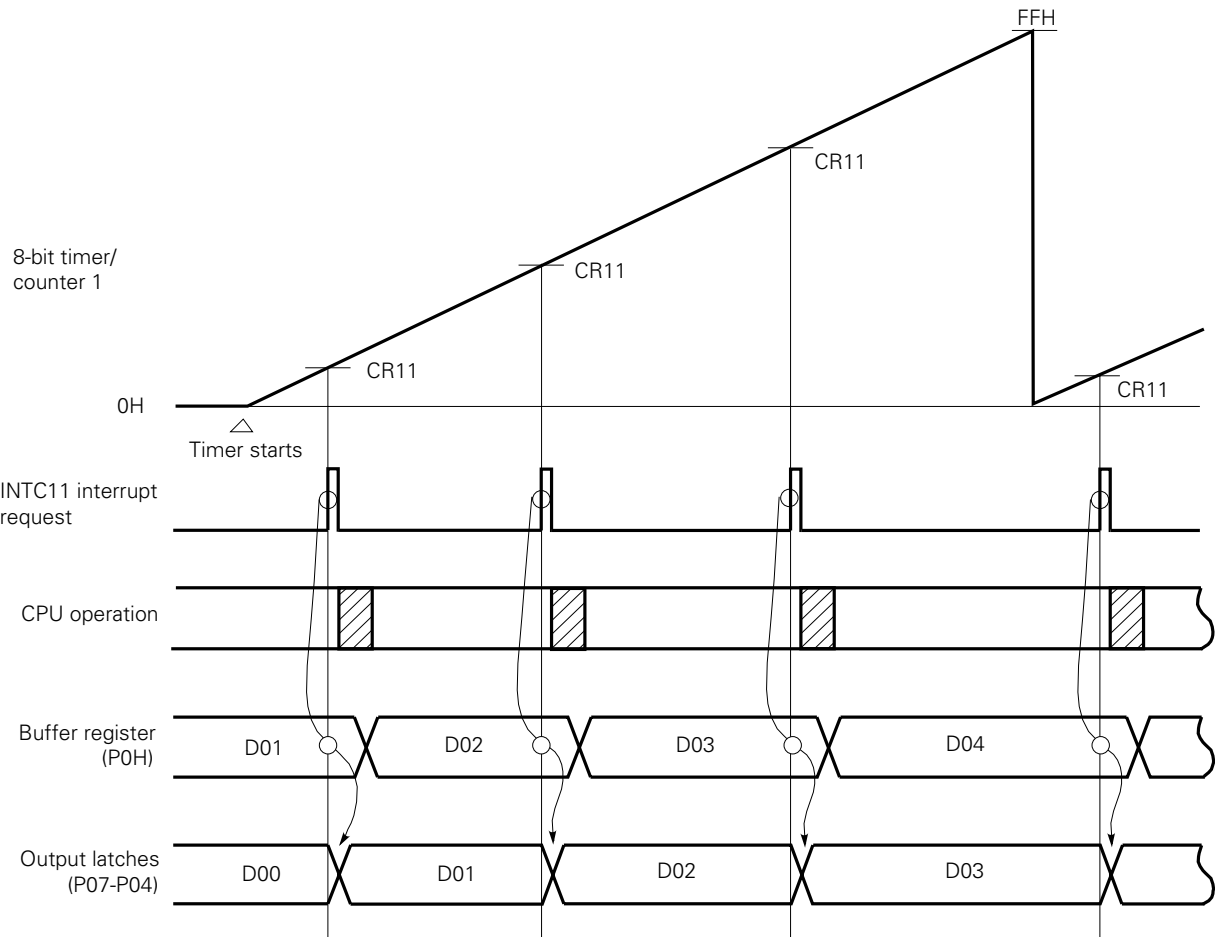
Table 6-2 Output Trigger for the Real-Time Output Port (When P0MH = P0ML = 1)

RTPC register		Output mode	P0H register	P0L register
BYTE	EXTR			
0	0	4-bit real-time output	INTC11	INTC10
	1		INTC11	INTC10/INTP0
1	0	8-bit real-time output	INTC10	
	1		INTC10/INTP0	

Caution With an in-circuit emulator, digital noise cannot be eliminated normally from the INTP0 pin. When it is specified that data transfer from the buffer register to the output latch be performed according to a signal from the INTP0 pin, data transfer may occur according to an erroneously detected edge. Keep in mind this characteristic when using the in-circuit emulator.

See the notes in Chapter 11 for details of erroneous detection of edges.

Fig. 6-4 Real-Time Output Port Operation Timing




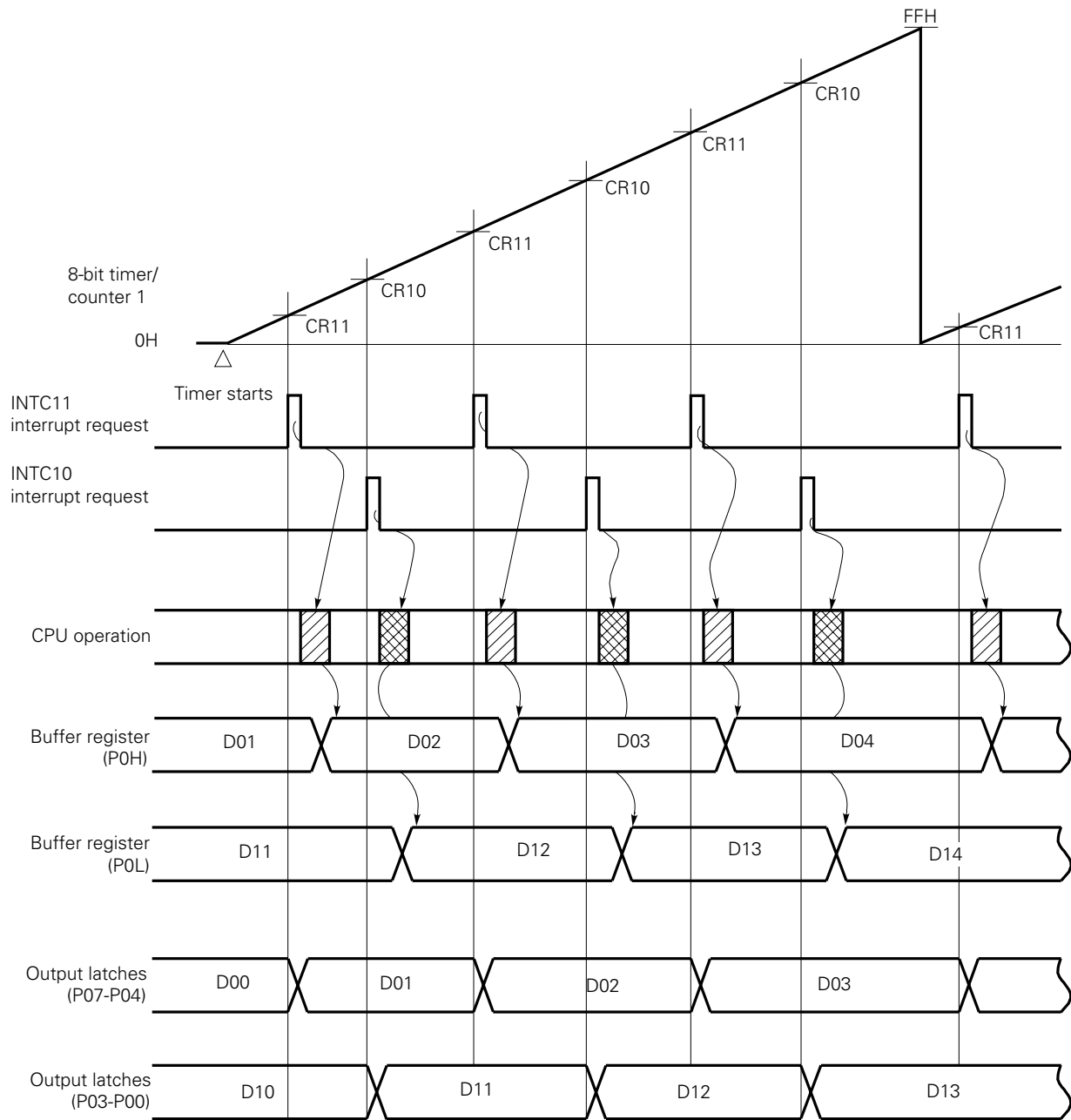


 The contents of the buffer register and compare register are rewritten by software processing or macro service (see **Section 12.4**).

Fig. 6-5 Real-Time Output Port Operation Timing (Controlling 2 Channels Independently of Each Other)



  The contents of the buffer register and compare register are rewritten by software processing or macro service (see Section 12.4).

6.5 APPLICATION EXAMPLE

This section describes an example of application in which P00 through P03 are used as a 4-bit real-time output port. Each time TM1 for 8-bit timer/counter 1 coincides with the contents of CR10, the contents of the P0L are output to P00 through P03. At this point, an interrupt occurs, and the interrupt handling routine for this interrupt sets the next data to be output and determines the timing at which the output is to change (see Fig 6-6).

See Section 7.2 for how to use timer/counter 1.

Fig. 6-7 shows the data to be set in the control register, Fig. 6-8 illustrates the procedure to set the data, and Fig. 6-9 is a flowchart of the interrupt handling routine.

Fig. 6-6 Real-Time Output Port Operation Timing

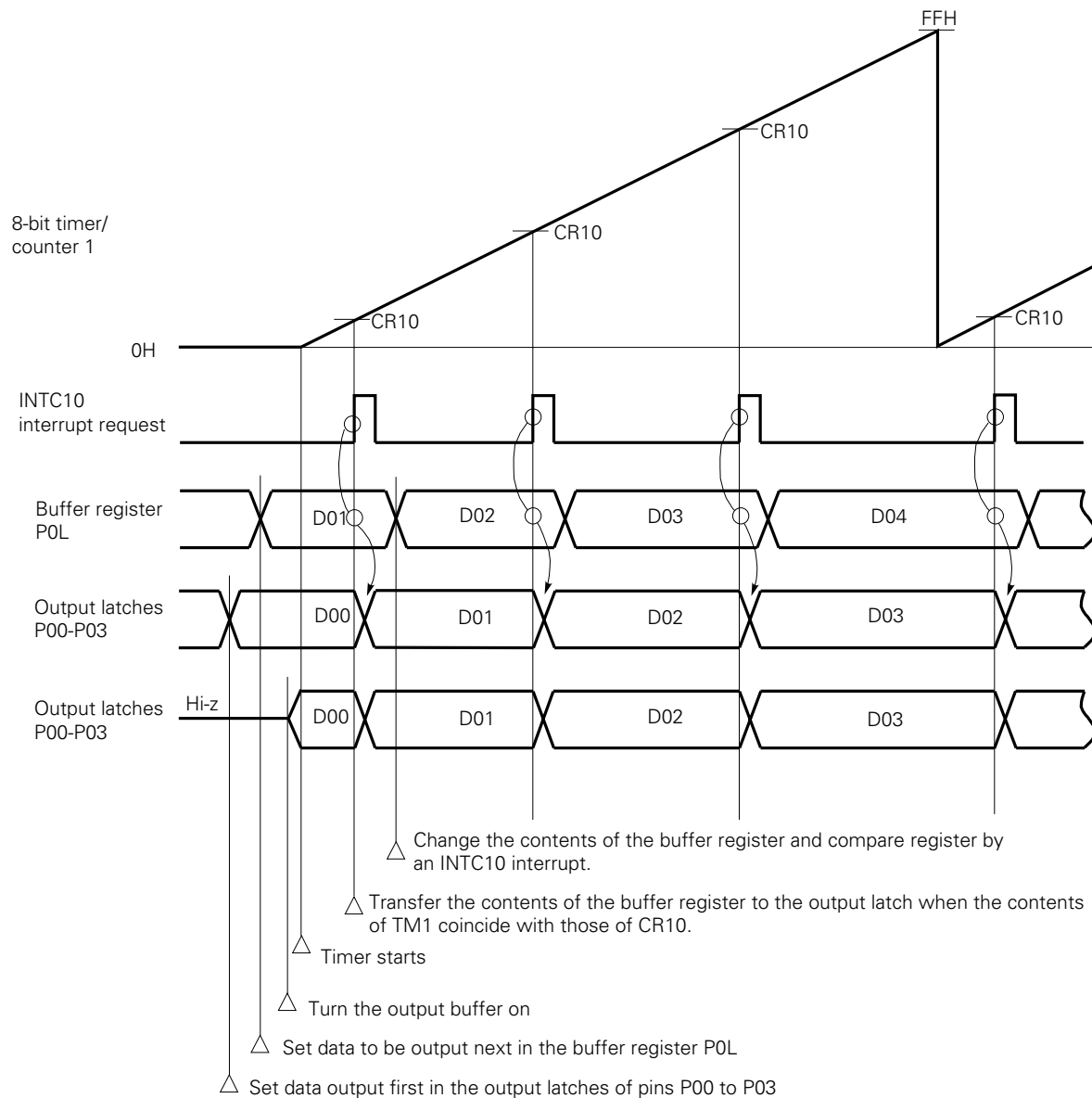


Fig. 6-7 Contents of the Control Register for the Real-Time Output Function

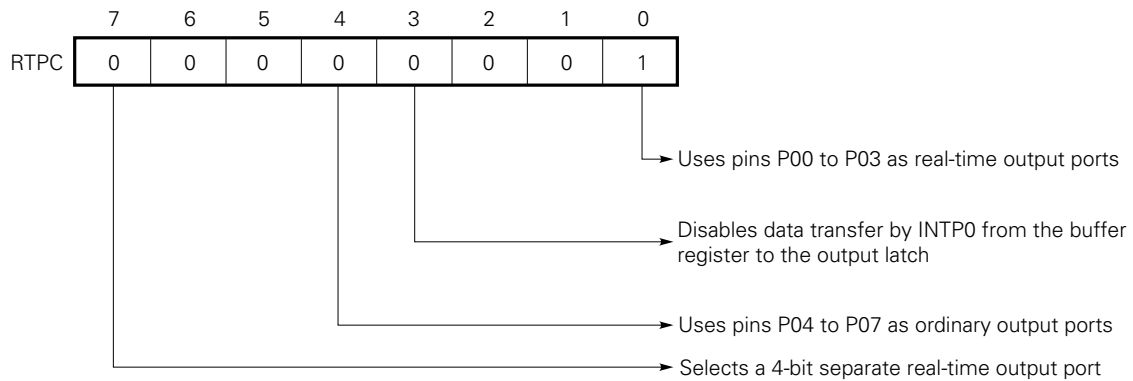


Fig. 6-8 Real-Time Output Function Setting Procedure

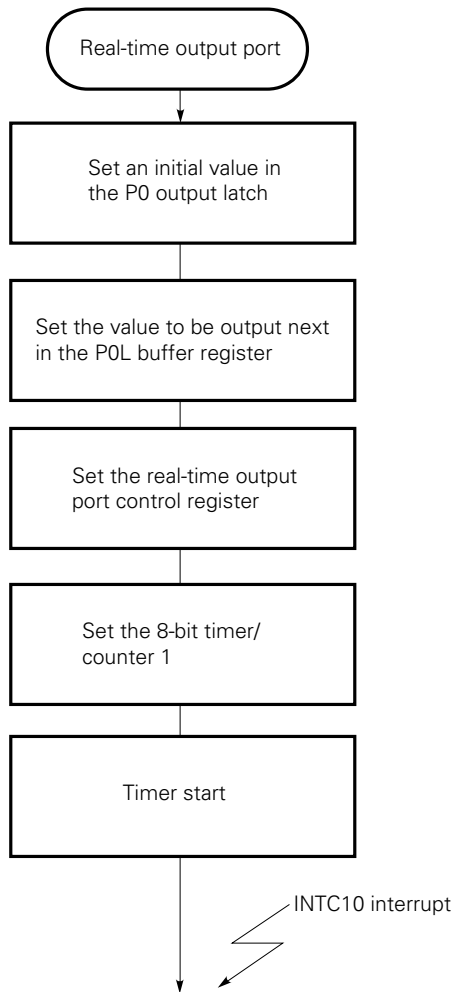
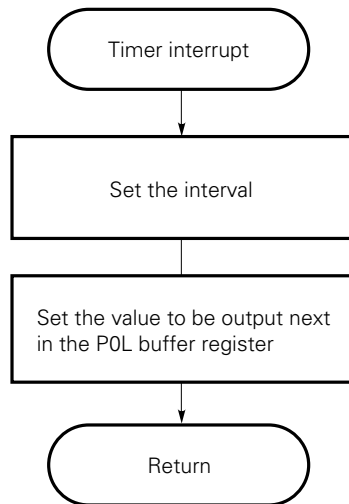


Fig. 6-9 Interrupt Request Handling When the Real-Time Output Function Is Used



6.6 NOTES

- (1) When the P0ML or P0MH is set to 1, the output buffer for the corresponding output port is turned on to output the contents of the port 0 output latch, regardless of the contents of the port 0 mode register (PM0). Therefore, initialize the contents of the output latch before specifying the real-time output port.
- (2) When port 0 is used as a real-time output port, values cannot be written directly to the output latch by software. Therefore, the output latch must be set with the initial value by software beforehand, if necessary.

If it is necessary to forcibly change the output data into a constant value when port 0 is used as a real-time output port, put port 0 in an ordinary output mode by manipulating the RTPC, before writing the desired value to the output latch.

- (3) Even if it is specified that data transfer from the buffer register to the output latch is to occur according to a signal from the INTP0 pin, data transfer from the buffer register to the output latch occurs when the contents of timer/counter 1 (TM1) coincide with the contents of the compare register (CR10).

To perform data transfer from the buffer register to the output latch only according to a signal from the INTP0 pin, perform one of the following points. Use of any of these methods does not allow use of the compare register (CR10) for 8-bit timer/counter 1.

- (a) Do not use 8-bit timer/counter 1.
- (b) When using the capture/compare register (CR11) for 8-bit timer/counter 1 as the compare register, use the compare register as an interval timer in a mode in which clearing occurs when the contents of the capture/compare register (CR11) coincide with the contents of 8-bit timer 1 (TM1).

In this case, however, make sure that the contents of the compare register (CR10) are greater than those in the CR11 register.

- (c) When using the capture/compare register (CR11) for 8-bit timer/counter 1 as the capture register, use the capture register only when it is guaranteed that the period of the valid edge of a signal input at INTP0 is sufficiently shorter than the time required for the value in 8-bit timer 1 (TM1) to change from 0 to FEH.

In this case, set the compare register (CR10) to FFH, and clear timer/counter 1 after captured.

- (d) If there is no problem, even if data transfer from the buffer register to the output latch is delayed by at least one clock of 8-bit timer (TM1), and it is guaranteed that TM1 does not overflow, set the following:
 - Specify that 8-bit timer/counter 1 is cleared after captured by the INTP0 signal.
 - Specify that the INTP0 signal is not used as a trigger signal for data transfer through the real-time output port.
 - Specify that the compare register (CR10) is reset to 0H.

- (4) With an in-circuit emulator, digital noise cannot be eliminated normally from the INTPO pin. When it is specified that data transfer from the buffer register to the output latch be performed according to a signal from the INTPO pin, data transfer may occur according to an erroneously detected edge. Keep in mind this characteristic when using the in-circuit emulator.

See the notes in **Chapter 11** for details of erroneous detection of edges.

CHAPTER 7 TIMER/COUNTER UNITS

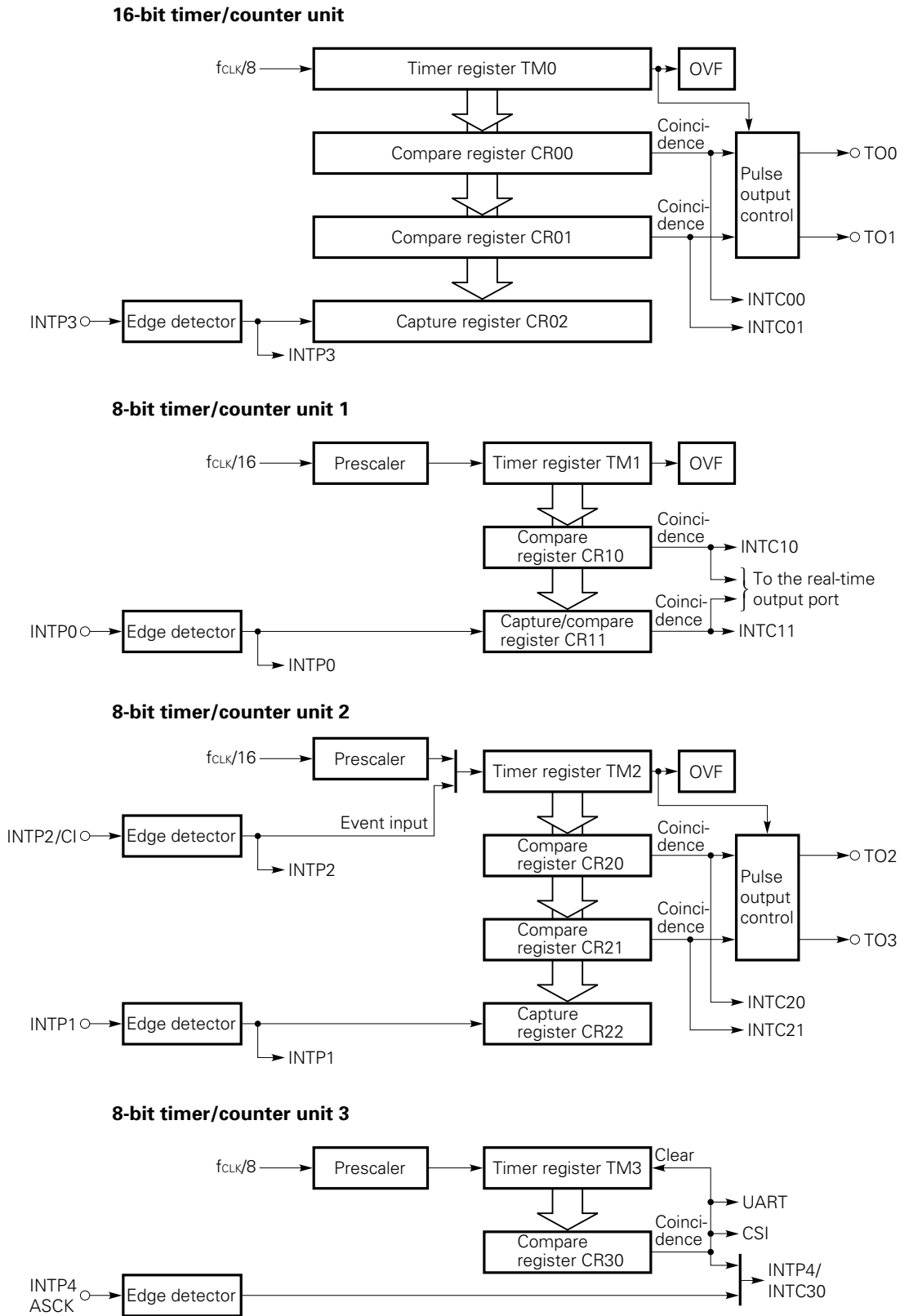
The μ PD78214 contains one 16-bit timer/counter unit (channel) and three 8-bit timer/counter units (channels).

Table 7-1 Timer/Counter Types and Functions

Unit		16-bit timer/ counter	8-bit timer/ counter 1	8-bit timer/ counter 2	8-bit timer/ counter 3
Types and functions					
Types	Interval timer	2 ch	2 ch	2 ch	1 ch
	External event counter	—	—	o	—
	One-shot timer	—	—	o	—
Functions	Timer output	2 ch	—	2 ch	—
	Toggle output	o	—	o	—
	PWM/PPG output	o	—	o	—
	Real-time output	—	o	—	—
	Pulse width measurement	o	o	o	—
	Number of interrupt requests	2	2	2	1
	Serial interface clock source	—	—	—	o

These timer/counter units can be used as a 7-channel timer because seven interrupt requests are supported in total.

Fig. 7-1 Block Diagrams of Timer/Counter Units



OVF: Overflow flag

7.1 16-BIT TIMER/COUNTER

7.1.1 Functions

The 16-bit timer/counter can function as an interval timer and can also be used for programmable square wave output and pulse width measurement. In addition to these basic functions, the 16-bit timer/counter can be used for the following:

- PWM output
- Period measurement

(1) Interval timer

When operating as an interval timer, the 16-bit timer/counter generates an internal interrupt at specified intervals.

Table 7-2 Intervals of 16-Bit Timer/Counter

Minimum interval	Maximum interval	Resolution
$8/f_{\text{CLK}}$ (1.3 μs)	$2^{16} \times 8/f_{\text{CLK}}$ (87.4 ms)	$8/f_{\text{CLK}}$ (1.3 μs)

The values in parentheses are based on $f_{\text{CLK}} = 6 \text{ MHz}$.

(2) Programmable square wave output

The 16-bit timer/counter outputs a square wave separately on the TO0 pin and TO1 pin.

Table 7-3 Programmable Square Wave Output Setting Range of 16-Bit Timer/Counter

Minimum pulse width	Maximum pulse width
$8/f_{\text{CLK}}$ (1.3 μs)	$2^{16} \times 8/f_{\text{CLK}}$ (87.4 ms)

The values in parentheses are based on $f_{\text{CLK}} = 6 \text{ MHz}$.

(3) Pulse width measurement

The 16-bit timer/counter measures the pulse width of a signal applied to the external interrupt pin INTP3.

Table 7-4 Pulse Width Measurement Range of 16-Bit Timer/Counter

Measurable pulse width	Resolution
$\leq 2^{16} \times 8/f_{\text{CLK}}$ ($\leq 87.4 \text{ ms}$)	$8/f_{\text{CLK}}$ (1.3 μs)

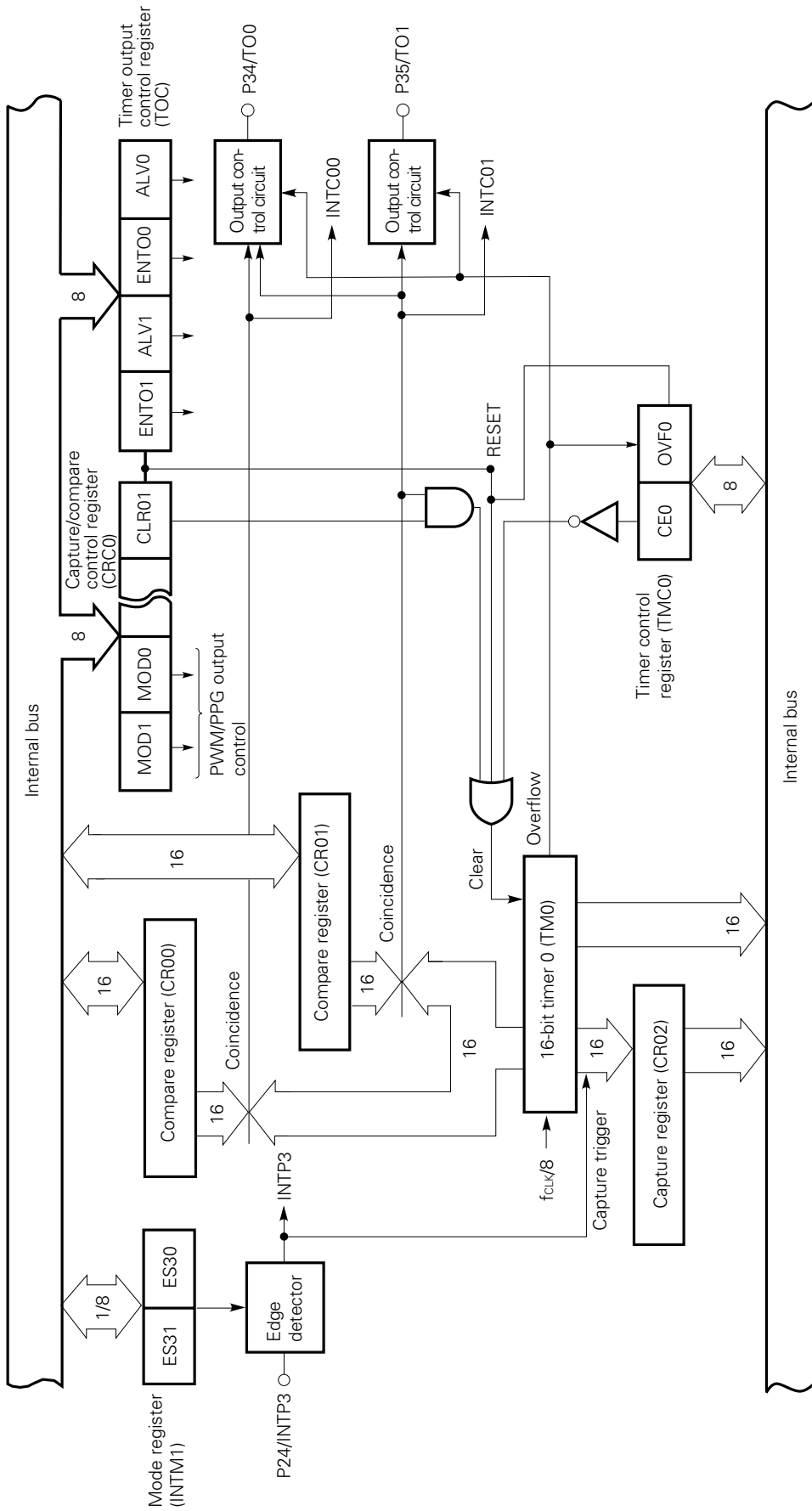
The values in parentheses are based on $f_{\text{CLK}} = 6 \text{ MHz}$.

7.1.2 Configuration

The 16-bit timer/counter consists of one 16-bit timer 0 (TM0), two 16-bit compare registers (CR00, CR01), and one 16-bit capture register (CR02).

Fig. 7-2 shows the block diagram of the 16-bit timer/counter.

Fig. 7-2 Block Diagram of 16-Bit Timer/Counter



(1) 16-bit timer 0 (TM0)

TM0 is a count-up timer using a count clock of $f_{CLK}/8$.

The count operation of TM0 can be enabled or disabled by timer control register 0 (TMC0).

TM0 allows only read operation using a 16-bit manipulation instruction. When the \overline{RESET} signal is applied, TM0 is cleared to 0000H, and count operation stops.

(2) Compare registers (CR00, CR01)

The CR00 and CR01 registers are 16-bit registers for holding a value that determines the period of the interval timer.

When the values of the CR00 and CR01 registers coincide with the value of TM0, interrupt requests (INTC00, INTC01) and timer output control signals are generated. Count value clear operation can also be performed when the value of CR01 coincides with the value of TM0.

The compare registers allow both read and write operations using a 16-bit manipulation instruction. When the \overline{RESET} signal is applied, the compare registers become undefined.

(3) Capture register (CR02)

The CR02 register is a 16-bit register for capturing the value of TM0.

Capture operation is performed on a valid edge (capture trigger) occurring on the external interrupt request (INTP3) input pin. The value of CR02 register is held until the next capture trigger occurs.

The CR02 register allows only read operation using a 16-bit manipulation instruction. When the \overline{RESET} signal is applied, the CR02 register becomes undefined,

(4) Edge detector

The edge detector detects a valid edge of an external input signal.

When the edge detector detects, on the INTP3 input pin, a valid edge specified in external interrupt mode register 1 (INTM1), INTP3 and a capture trigger are generated. (See Fig. 11-2 for information about the INTM1 register.)

(5) Output control circuit

When the value of CR00 or CR01 coincides with the value of TM0, timer output can be inverted. By setting the lower 4 bits of the timer output control register (TOC), a square wave can be output on a timer output pin (TO0, TO1). At this time, PWM/PPG output is possible, depending on the setting of capture/compare control register 0 (CRC0).

Timer output can be disabled or enabled by the TOC register. When timer output is disabled, the inactive level is output on the TO0 and TO1 pins. (The active level is set using the TOC register.)

7.1.3 16-Bit Timer/Counter Control Registers**(1) Timer control register 0 (TMC0)**

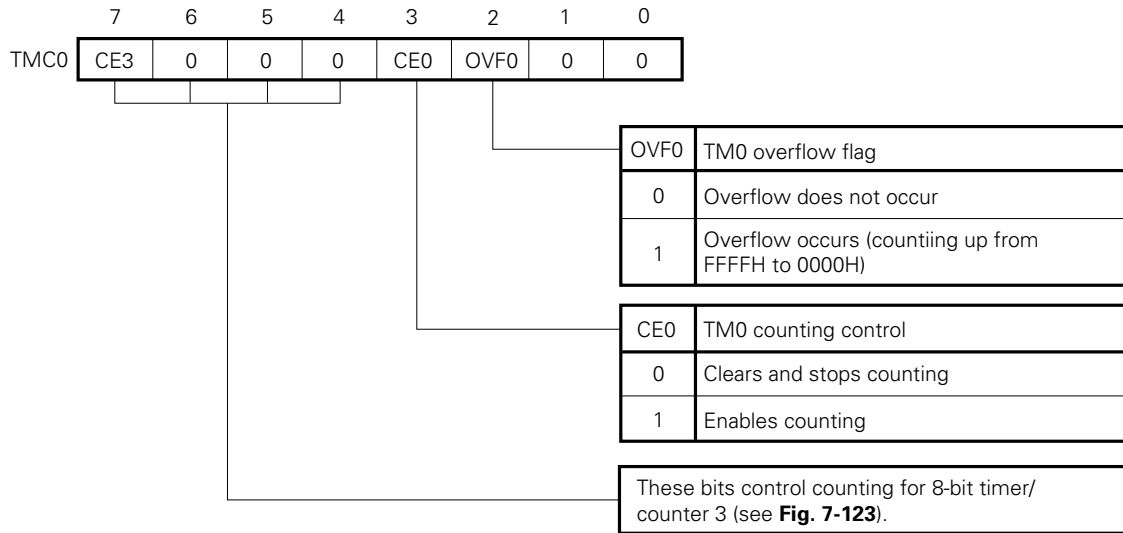
The TMC0 register is an 8-bit register for controlling the count operation of 16-bit timer 0 (TM0).

The lower 4 bits control the count operation of TM0. (The higher 4 bits control the count operation of 8-bit timer/counter 3.)

The TMC0 register allows both read and write operations using an 8-bit manipulation instruction. Fig. 7-3 shows the format of the TMC0 register.

When the \overline{RESET} signal is applied, the TMC0 register is cleared to 00H.

Fig. 7-3 Format of Timer Control Register 0 (TMC0)



Remark The OVF0 bit can be reset only by software.

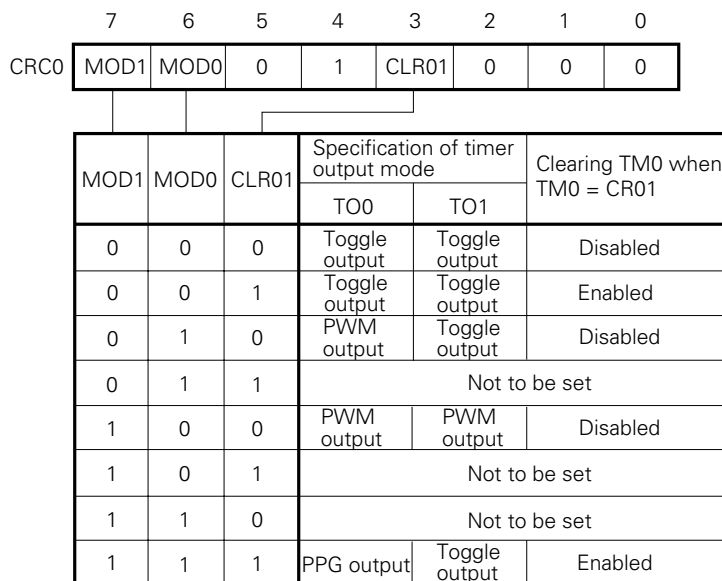
(2) Capture/compare control register 0 (CRC0)

The CRC0 register is used to specify the condition for enabling the clear operation of TM0 to be performed by a coincidence between the value of the CR01 compare register and the count value of TM0. The CRC0 register is also used to specify a timer output (TO0, TO1) mode.

The CRC0 register allows only write operation using an 8-bit manipulation instruction. Fig. 7-4 shows the format of the CRC0 register.

When the $\overline{\text{RESET}}$ signal is applied, the CRC0 register is cleared to 10H.

Fig. 7-4 Format of Capture/Compare Control Register 0 (CRC0)



(3) Timer output control register (TOC)

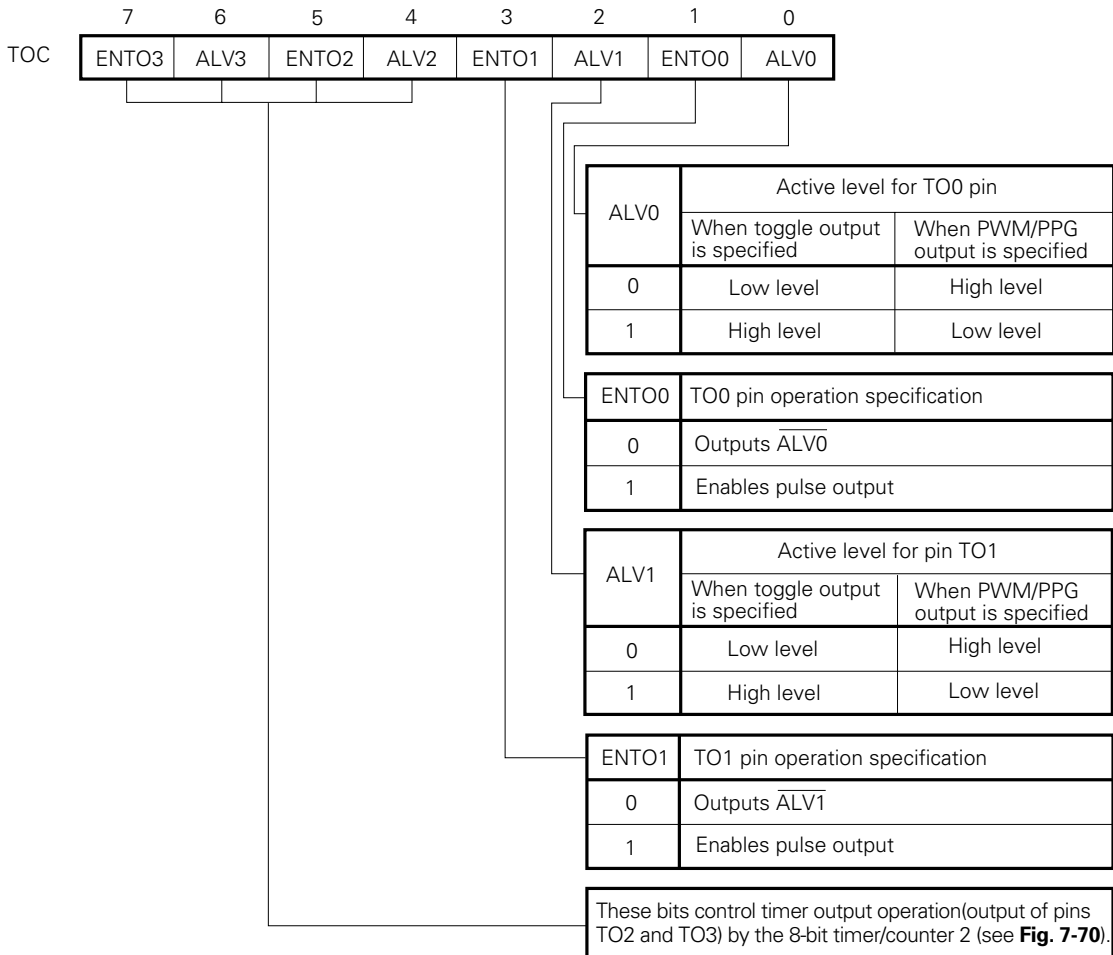
The TOC register is an 8-bit register for specifying the active level of timer output and for enabling/disabling timer output.

The lower 4 bits control the timer output operation (on the TO0 and TO1 pins) of the 16-bit timer/counter. (The higher 4 bits control the timer output operation (on the TO2 and TO3 pins) of 8-bit timer/counter 2.)

The TOC register allows only write operation using an 8-bit manipulation instruction. Fig. 7-5 shows the format of the TOC register.

When the $\overline{\text{RESET}}$ signal is applied, the TOC register is cleared to 00H.

Fig. 7-5 Format of Timer Output Control Register (TOC)



7.1.4 Operation of 16-Bit Timer 0 (TM0)

(1) Basic operation

The 16-bit timer/counter performs count operation by counting up with a count clock of $f_{CLK}/8$.

When the \overline{RESET} signal is applied, TM0 is cleared to 0000H, and count operation stops.

Bit 3 (CE0) of timer control register 0 (TMC0) is used to enable/disable count operation. When the CE0 bit is reset to 1 by software, TM0 is cleared to 0000H by the first count clock pulse, then count-up operation starts.

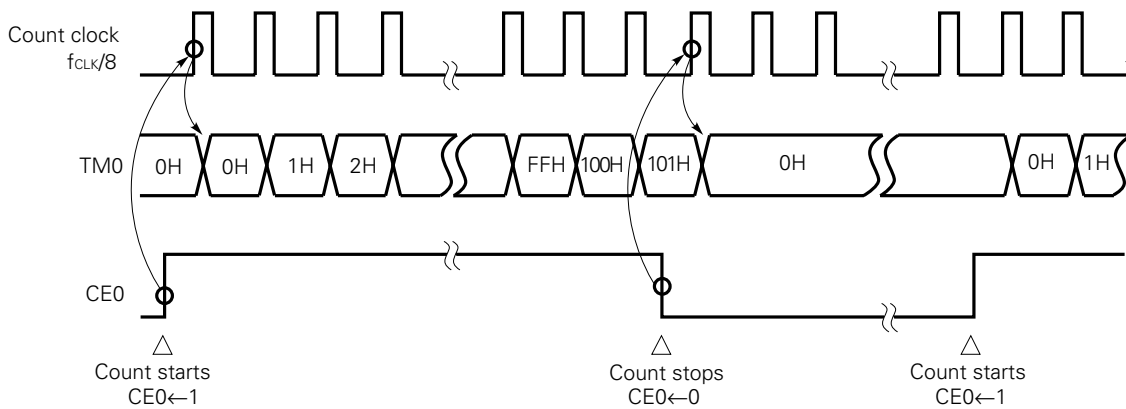
When the CE0 bit is reset to 0, TM0 is cleared to 0000H by the next count clock pulse, then capture operation and coincide with signal generation stop.

If the CE0 bit is set to 1 when the CE0 bit is already set to 1, TM0 is not cleared, but continues count operation.

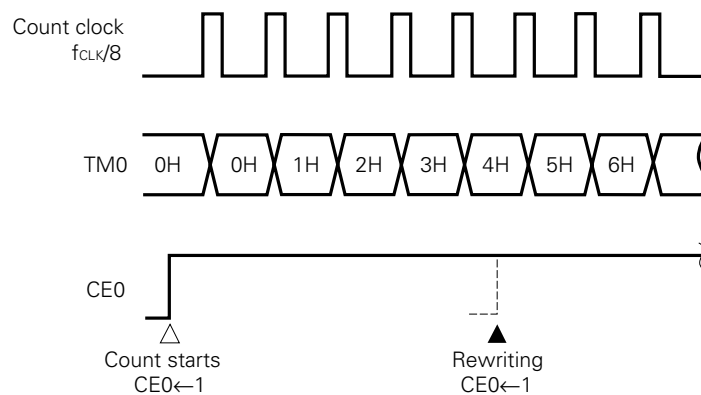
If the count clock is applied when the value of TM0 is FFFFH, TM0 is set to 0000H. At this time, OVFO is set to send the overflow signal to the output control circuit. OVFO can be cleared only by software. Count operation continues.

Fig. 7-6 Basic Operation of 16-Bit Timer 0 (TM0)

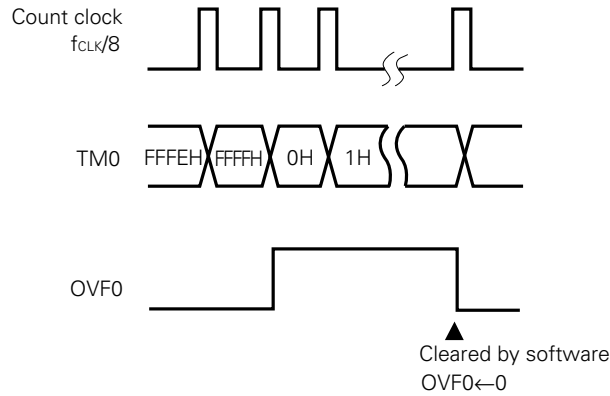
(a) Count start → count stop → count start



(b) When the CE0 bit is set to 1 again after count operation starts



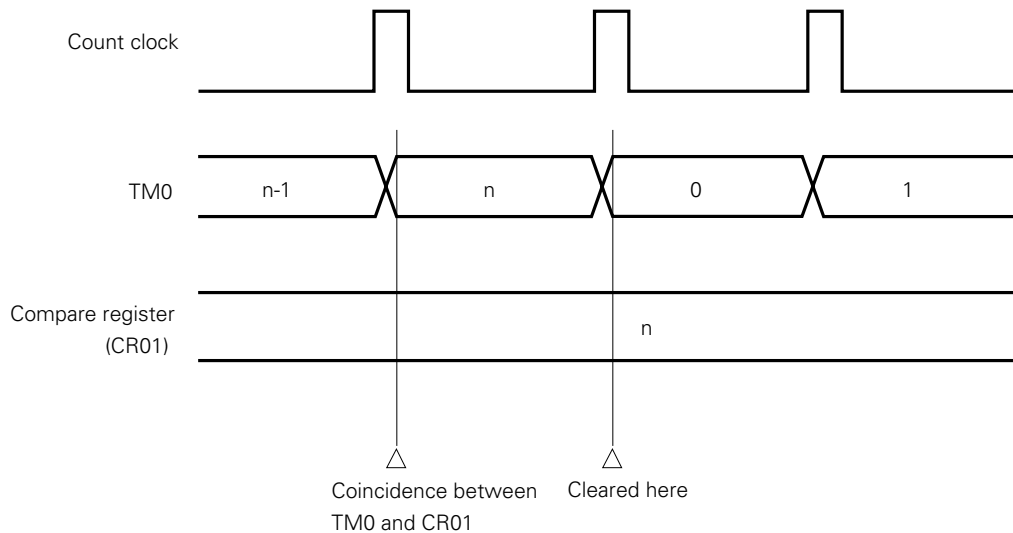
(c) When the value of TM0 is FFFFH



(2) Clear operation

After a coincidence with the CR01 compare register, 16-bit timer 0 (TM0) can be automatically cleared. If a TM0 clear cause occurs, TM0 is cleared to 0000H by the next count clock pulse. This means that even if a TM0 clear cause occurs, TM0 holds the value existing at that time until the next count clock pulse is applied.

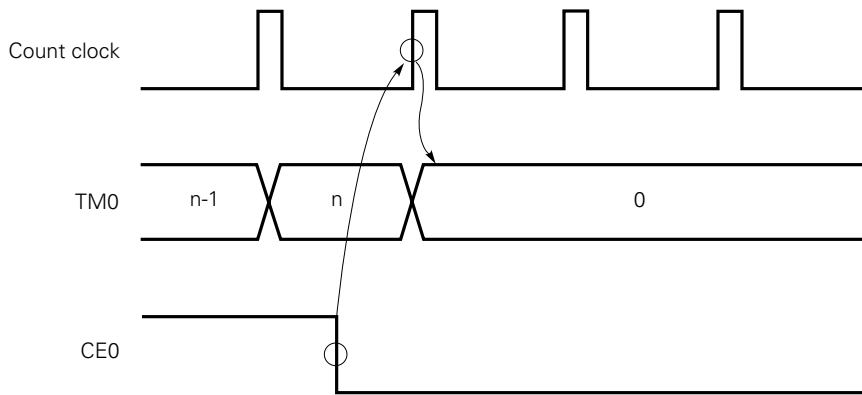
Fig. 7-7 TM0 Cleared by a Coincidence with Compare Register (CR01)



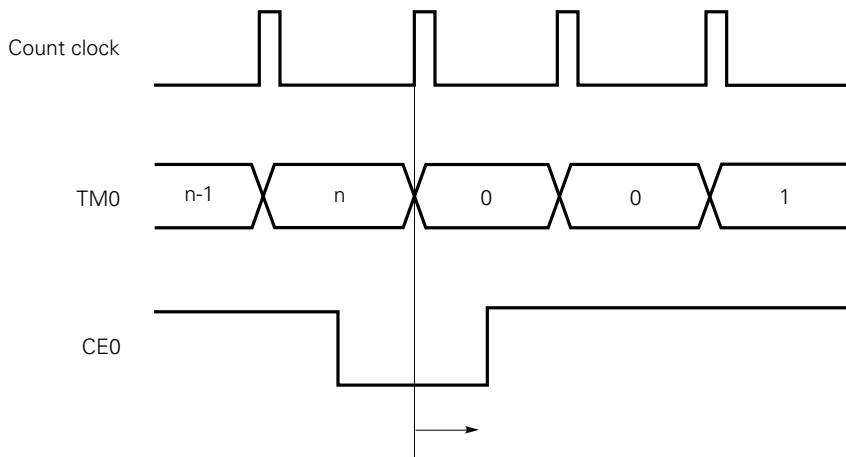
TM0 can also be cleared by software when the CE0 bit of the timer control register (TMC0) is reset to 0. Similarly, clear operation is performed by the count clock pulse following the resetting of CE0 bit to 0. If the CE0 bit is set to 1 before TM0 is reset to 0 by the resetting of the CE0 bit to 0 (that is, before the first count clock pulse is applied after the CE0 bit is reset to 0), two operations are simultaneously performed: one operation is an operation to clear TM0 to 0, and the other operation is a count operation starting with the counting of 0.

Fig. 7-8 Clear Operation When the CE0 Bit Is Reset to 0

(a) Basic operation

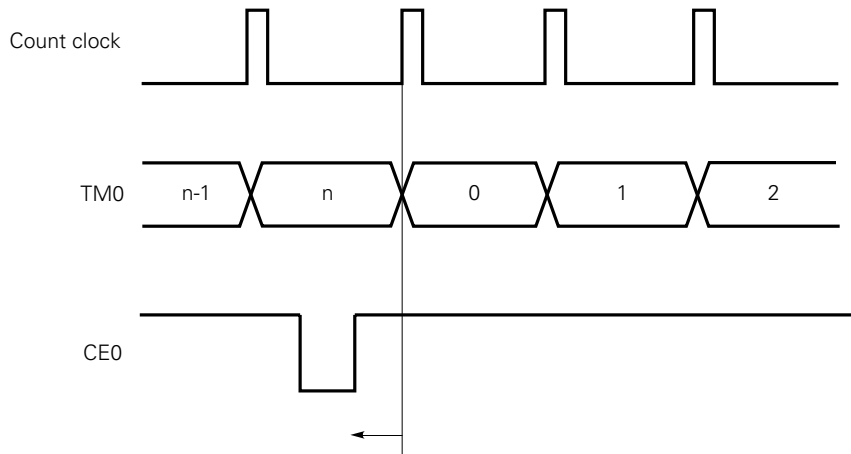


(b) Restart after 0 is set in TM0 cleared



When the CE0 bit is set to 1 after this count clock, counting starts from 0 on the count clock input after the CE0 bit has been set.

(c) Restart before 0 is set in TM0 cleared



When the CE0 bit is set to 1 before this count clock, Clearing TM0 by $CE0 \leftarrow 0$ and counting by $CE0 \leftarrow 1$ are performed simultaneously.

7.1.5 Compare Register and Capture Register Operations

(1) Compare operation

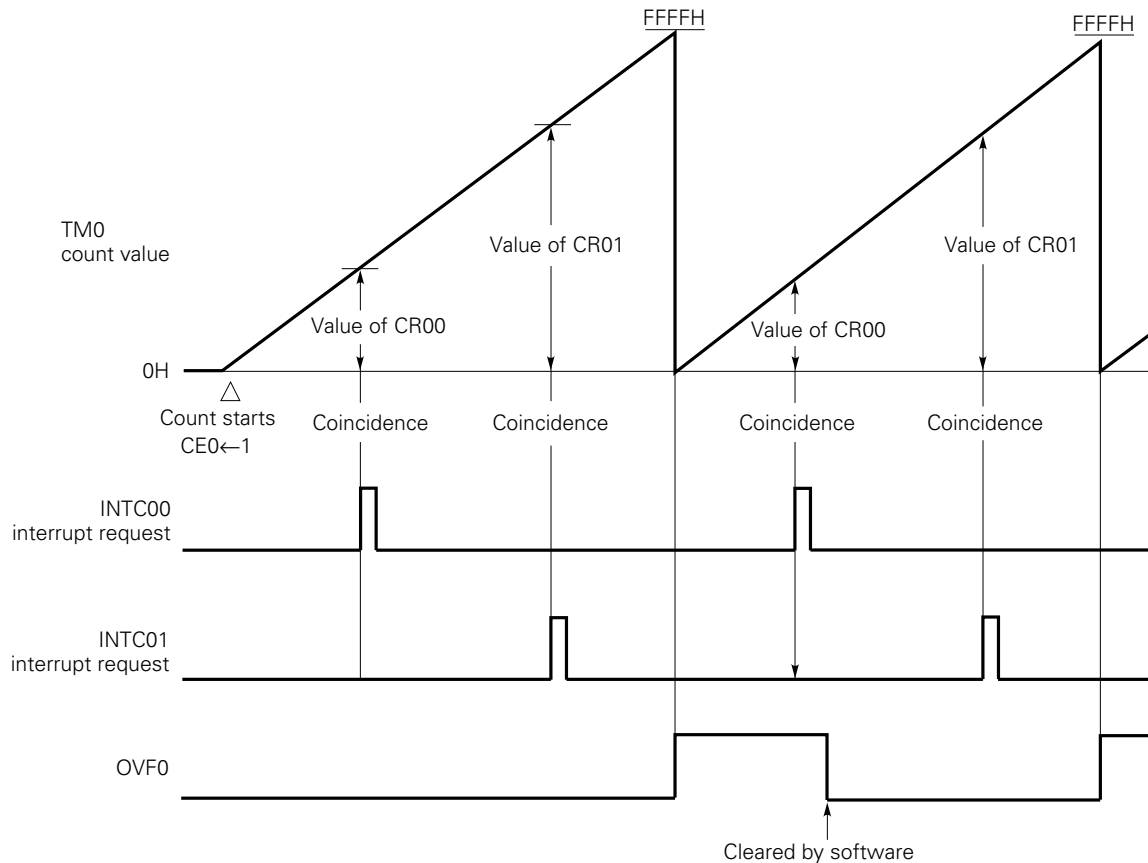
The 16-bit timer/counter performs an operation to compare the values set in the compare registers with timer count values.

When the values set in the compare registers (CR00, CR01) coincide with count values of 16-bit timer 0 (TM0), the coincidence signal is sent to the output control circuit. At the same time, the interrupt requests (INTC00, INTC01) are generated.

After the value of the CR01 register coincides with a count value of TM0, the count value of TM0 can be cleared. Thus, the 16-bit timer/counter can operate as an interval timer for repeatedly counting up to the value set in the CR01 register.

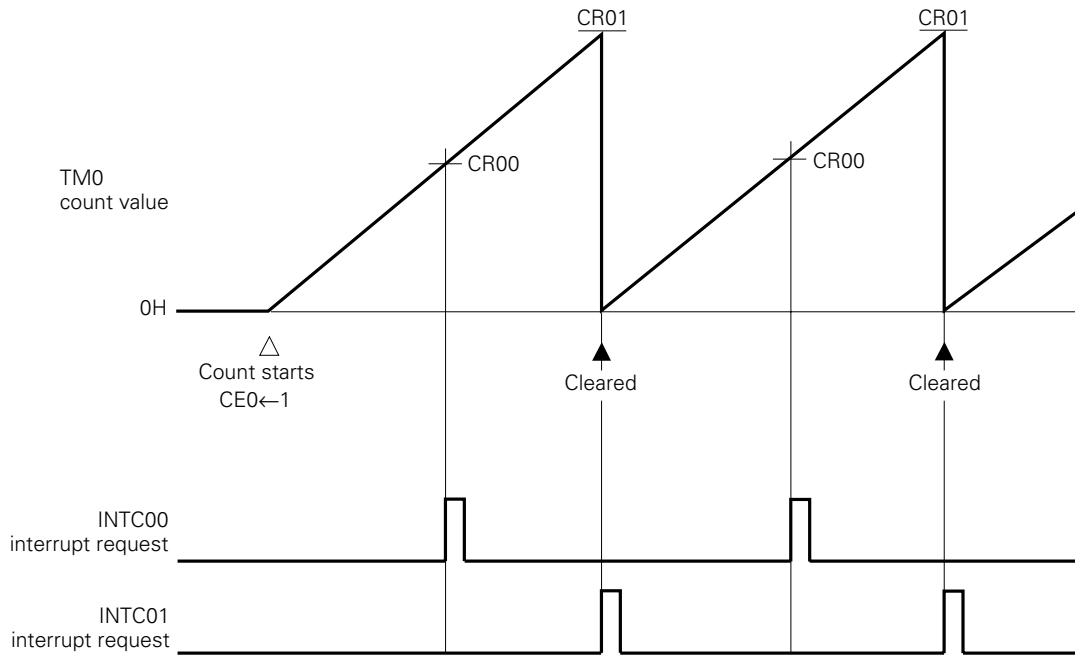
Caution Before the 16-bit timer/counter can perform a compare operation, a value must be loaded into the compare register or registers used.

Fig. 7-9 Compare Operation



Remark CLR01 = 0

Fig. 7-10 TM0 Cleared After a Coincidence Is Detected



Remark CLR01 = 1

(2) Capture operation

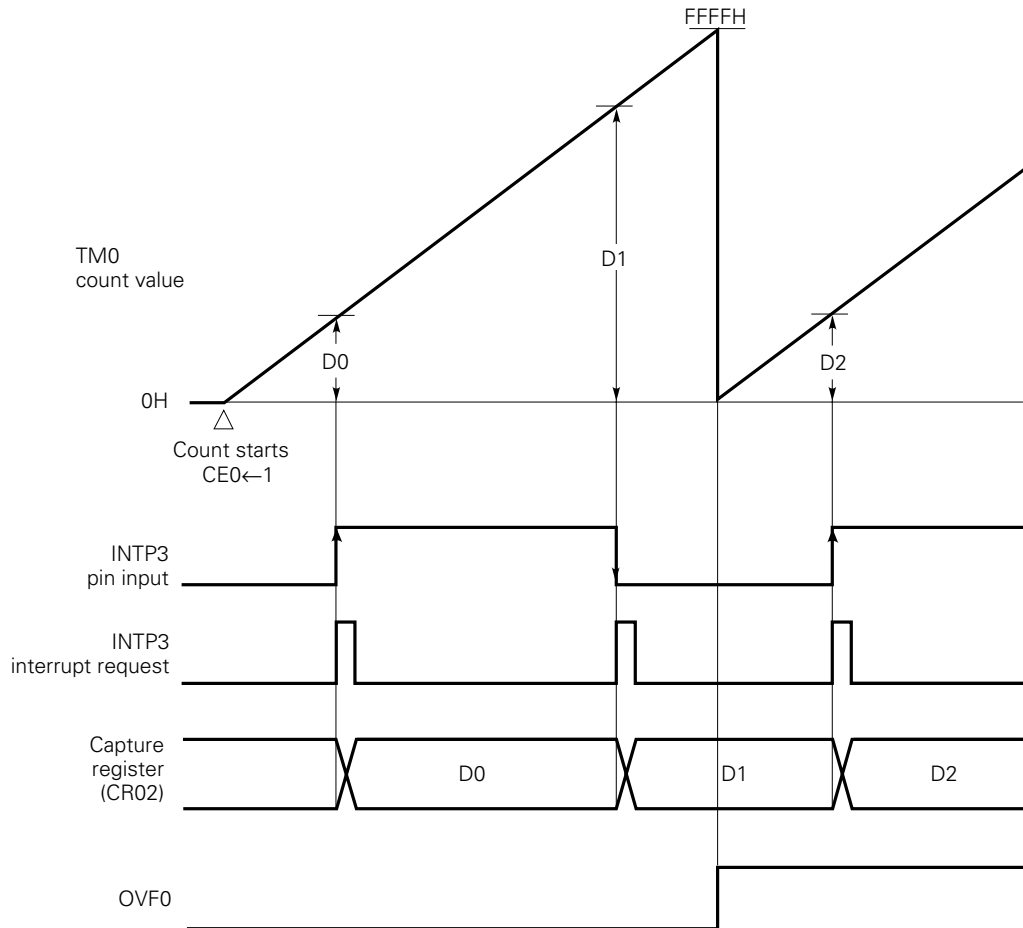
The 16-bit timer/counter performs a capture operation to load the count value of the timer into the capture register in synchronism with an external trigger.

As an external trigger, a valid edge detected on the external interrupt request (INTP3) input pin is used (capture trigger). In synchronism with a capture trigger, the count value of 16-bit timer 0 (TM0) in count operation is loaded and held in the CR02 capture register. Until the next capture trigger occurs, the value of the CR02 register is held.

A valid edge used as a capture trigger is set using external interrupt mode register 1 (INTM1). When both a rising edge and falling edge are set as capture triggers, the pulse width of an applied external signal can be measured. When a capture trigger is generated using one edge, the period of an input pulse signal can be measured.

For the detailed format of the INTM1 register, see **Fig. 11-2 in Chapter 11**.

Fig. 7-11 Capture Operation



7

Remark Dn: TM0 count value (n = 0, 1, 2, ...)
CLR01 = 0

Caution With an in-circuit emulator, digital noise on the INTP3 pin cannot be removed correctly. When the capture function is used, the operation described below is performed if an edge is detected erroneously.

- When IE-78210-R is used
Capture operation is performed on an erroneously detected edge. At this time, an interrupt request is also generated on the edge.
- When other in-circuit emulators are used
Capture operation is not performed on an erroneously detected edge. However, an interrupt request is generated on the edge. Accordingly, the capture value must be used after checking by software to see if the INTP3 interrupt was generated normally or generated on an erroneously detected edge. For detailed information about erroneous edge detection, see Section 11.4.

7.1.6 Basic Operation of Output Control Circuit

The output control circuit controls the levels of the timer outputs (TO0, TO1) according to the overflow signal or the coincidence signal from the compare registers. The operation of the output control circuit is determined by the timer output control register (TOC) and capture/compare control register 0 (CRC0). (See **Table 7-5**.)

Before a timer output (TO0 or TO1) signal can be output on a pin, the pin must be placed in the control mode by the PMC3 register.

Table 7-5 Timer Output (TO0, TO1) Operation

TOC			CRCO				TO1		TO0
ENTO1	ALV1	ENTO0	ALV0	MOD1	MOD0	CLR01	TO1	TO0	
0	0/1	0	0/1	x	x	x	Tied high/low	Tied high/low	
0	0/1	1	0/1	0	0	x	Tied high/low	Toggle output (low/high active)	
1	0/1	0	0/1	0	0	x	Toggle output (low/high active)	Tied high/low	
1	0/1	1	0/1	0	0	x	Toggle output (low/high active)	Toggle output (low/high active)	
0	0/1	1	0/1	0	1	0	Tied high/low	PWM output (high/low active)	
1	0/1	0	0/1	0	1	x	Toggle output (low/high active)	Tied high/low	
1	0/1	1	0/1	0	1	0	Toggle output (low/high active)	PWM output (high/low active)	
0	0/1	1	0/1	1	0	0	Tied high/low	PWM output (high/low active)	
1	0/1	0	0/1	1	0	0	PWM output (high/low active)	Tied high/low	
1	0/1	1	0/1	1	0	0	PWM output (high/low active)	PWM output (high/low active)	
0	0/1	1	0/1	1	1	1	Tied high/low	PPG output (high/low active)	
1	0/1	0	0/1	1	1	x	Toggle output (low/high active)	Tied high/low	
1	0/1	1	0/1	1	1	1	Toggle output (low/high active)	PPG output (high/low active)	

- Remarks**
1. The numbers 0 and 1 of "0/1" in the ALV1 and ALV0 columns correspond to the words high and low of "high/low" in the TO1 and TO0 columns, respectively.
 2. The character x represents the value 0 or 1.
 3. No combinations other than those indicated in this table are allowed.

(1) Basic operation

By setting ENTOn (n = 0, 1) of the timer output control register (TOC) to 1, the timer outputs (TO1, TO0) can be changed with the timing determined by MOD0, MOD1, and CLR01 of capture/compare control register 0 (CRC0).

In addition, by clearing ENTOn (n = 0, 1) to 0, the levels of the timer outputs (TO1, TO0) can be tied. The level where an output is tied is determined by ALVn (n = 0, 1) of the timer output control register (TOC). When ALVn (n = 0, 1) is 0, the output is tied high; when ALVn (n = 0, 1) is 1, the output is tied low.

(2) Toggle output

Toggle output is an operation mode where the level of output is inverted each time the value of a compare register (CR00, CR01) coincides with the value of 16-bit timer 0 (TM0). The output level of TO0 is inverted when the value of CR00 coincides with the value of TM0. The output level of TO1 is inverted when the value of CR01 coincides with the value of TM0.

Fig. 7-12 Toggle Output Operation

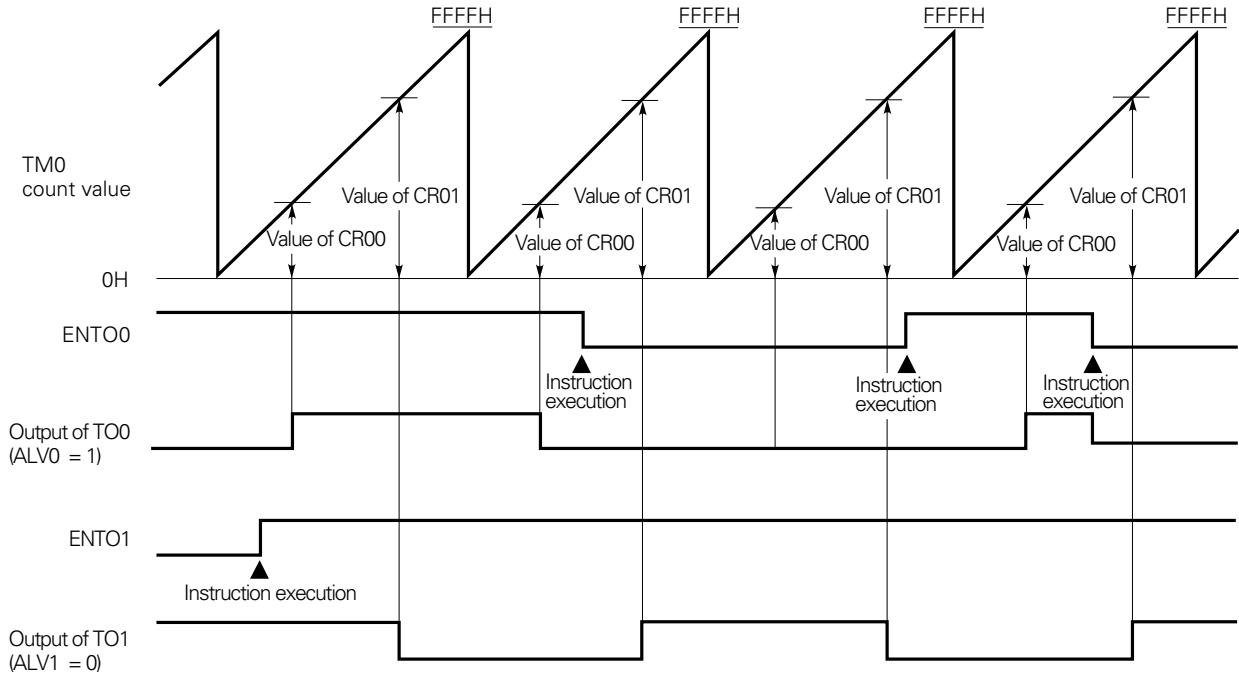


Table 7-6 TO0 and TO1 Toggle Output ($f_{CLK} = 6 \text{ MHz}$)

Count clock	Minimum pulse width	Maximum interval
$f_{CLK}/8$	1.3 μs	87.4 ms

7.1.7 PWM Output

The PWM output function outputs a PWM signal whose period coincides with the full-count period of 16-bit timer 0 (TM0). The pulse width of TO0 is determined by the value of CR00, and the pulse width of TO1 is determined by the value of CR01. Before this function can be used, the CLR01 bit of capture/compare control register 0 (CRC0) must be set to 0.

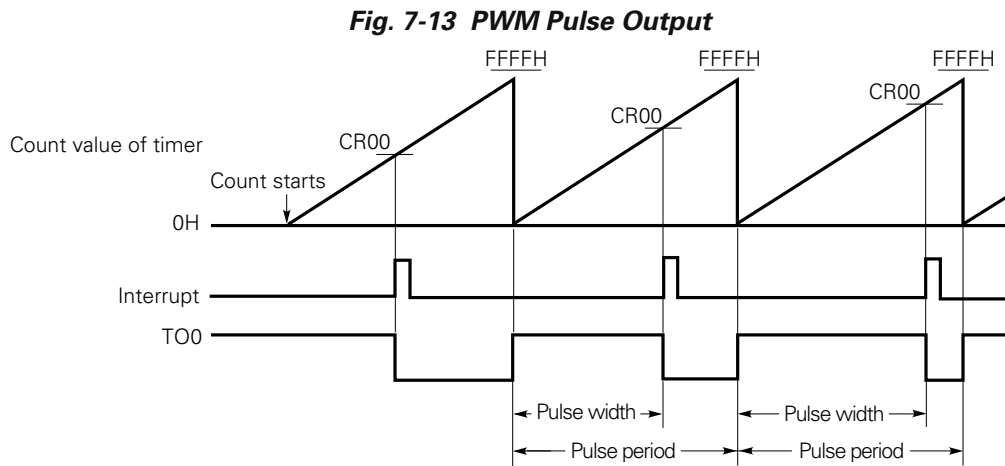
The pulse period and pulse width are as follows:

- PWM period = $524288/f_{CLK}$
- PWM pulse width = $(\text{value set in compare register} \times 8 + 2)/f_{CLK} \approx (\text{value set in compare register}) \times 8/f_{CLK}$

Note Zero cannot be set in the compare registers.

- Duty factor = $(\text{PWM pulse width})/(\text{PWM period}) = ((\text{value set in compare register}) \times 8 + 2)/(65536 \times 8) \approx (\text{value set in compare register})/65536$

Caution In PWM output, the actual pulse width is longer than a value obtained with the approximate expression by two clock pulses of f_{CLK} for the active level, and is shorter than such an approximate value by two clock pulses of f_{CLK} for the inactive level. Take this point into consideration when high-precision output is required.



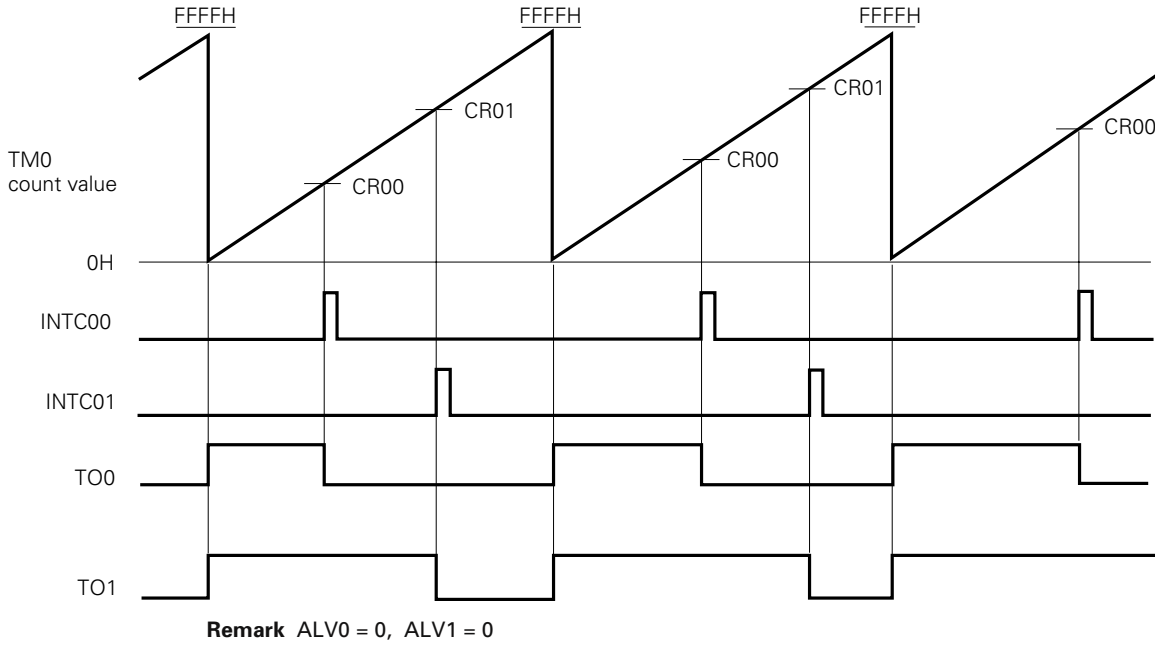
Remark ALV0 = 0

Table 7-7 PWM Output on TO0 and TO1 ($f_{CLK} = 6 \text{ MHz}$)

Count clock	Minimum pulse width	PWM period	PWM frequency
$f_{CLK}/8$	1.3 μs	87.4 ms	11.4 Hz

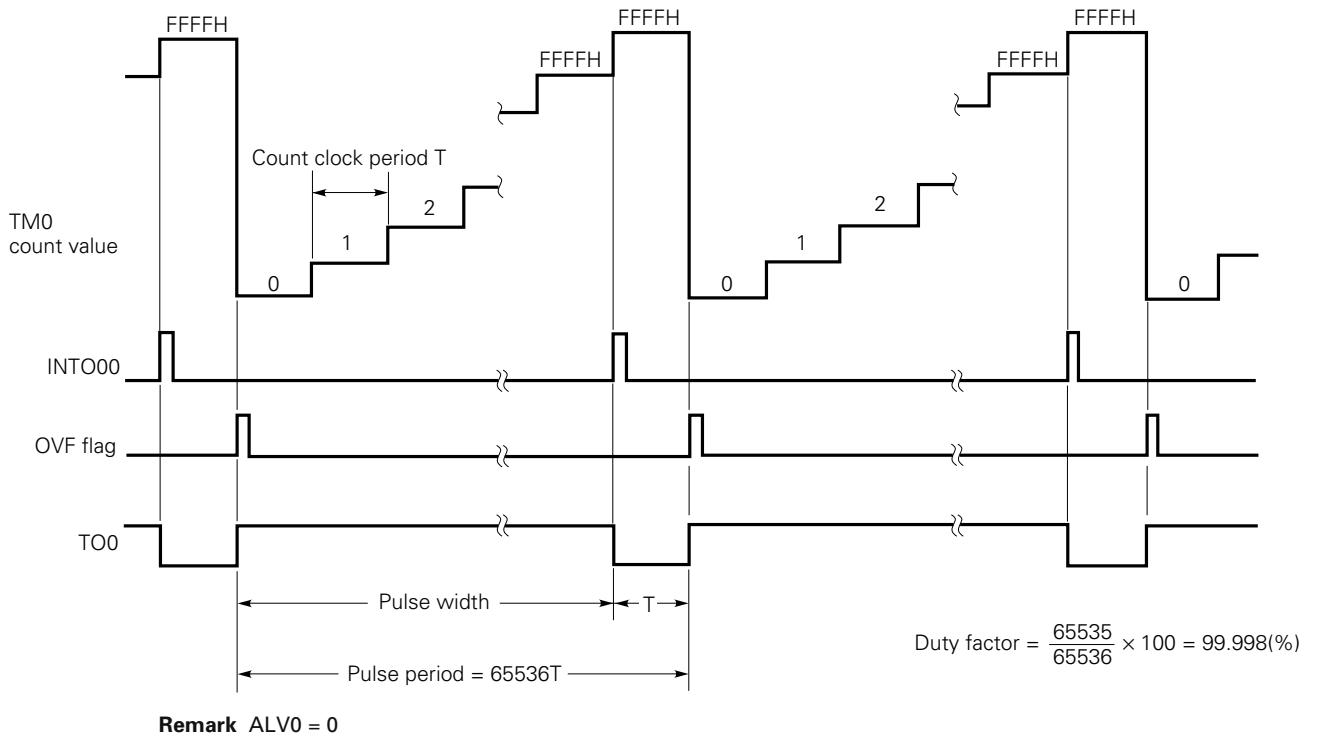
Fig. 7-14 shows an example of 2-channel PWM output. Fig. 7-15 shows PWM output when FFFFH is set in the CR00 compare register.

Fig. 7-14 Example of PWM Output Using TM0



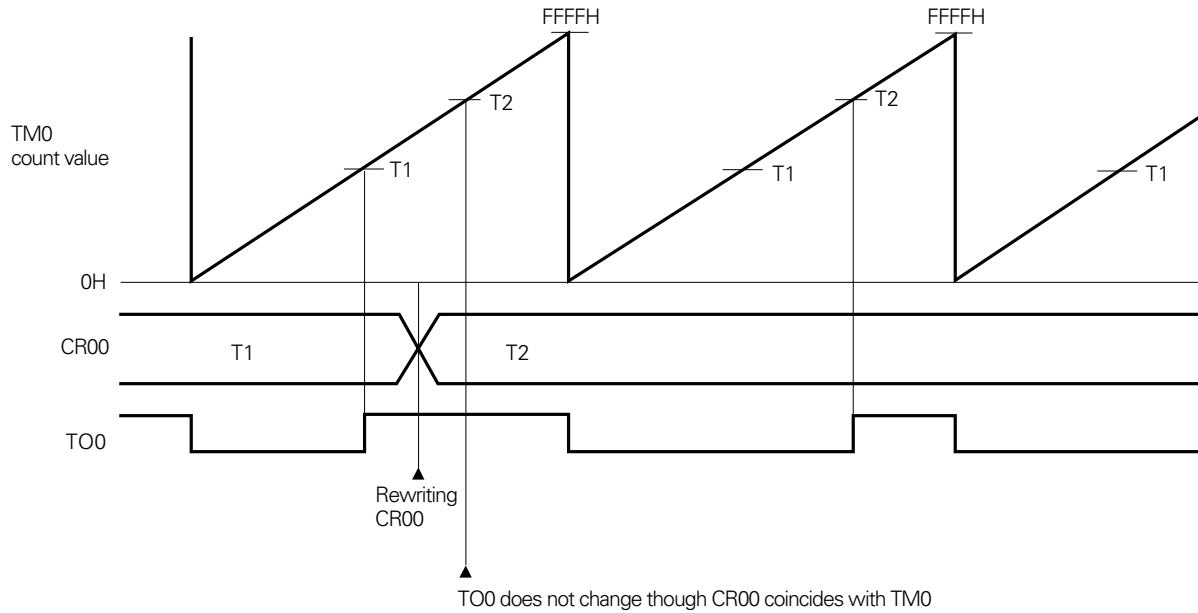
7

Fig. 7-15 PWM Output When CR00 = FFFFH



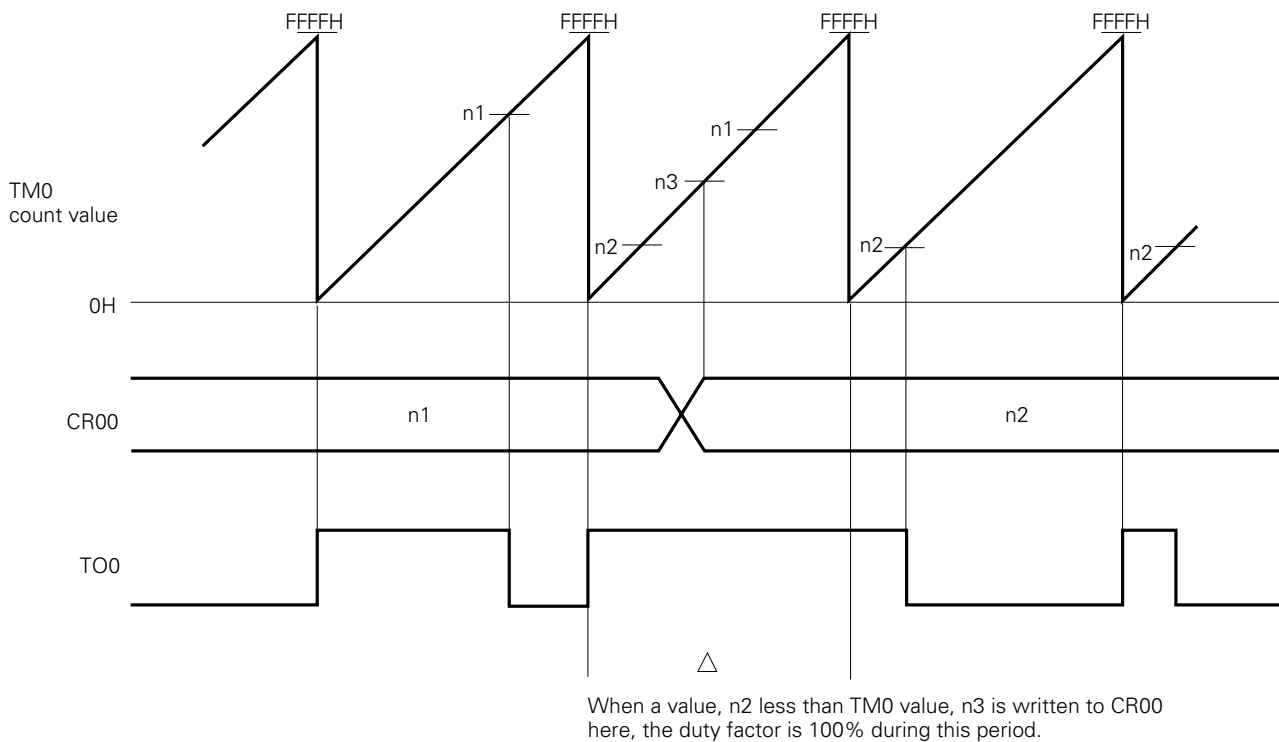
Even if the value of a compare register (CR00, CR01) coincides with the value of 16-bit timer 0 (TM0) more than once during one period of PWM output, the output levels on the timer outputs (TO0, TO1) do not change.

Fig. 7-16 Example of Rewriting Compare Register CR00



Cautions 1. If a value less than the value of 16-bit timer 0 (TM0) is set in a compare register (CR00, CR01), a PWM signal with a 100% duty factor is output. Rewrite the CR00 or CR01 compare register, if required, by using an interrupt generated by a coincidence between TM0 and the compare register.

Fig. 7-17 Example of PWM Output Signal with a 100% Duty Factor



Remark ALV0 = 0

2. If timer output is disabled (ENTOn = 0: n = 0, 1), the output level on the TOn (n = 0, 1) pin is the inverted value of the value set in ALVn (n = 0, 1). Accordingly, note that if timer output is disabled when the PWM output function is selected, the active level is output.

7.1.8 PPG Output

The PPG output function outputs a square wave that has a period determined by the CR01 compare register, and has a pulse width determined by the CR00 compare register. PPG output is PWM output whose period is made variable. This output signal can be output on the TO0 pin only.

Before this function can be used, the CLR01 bit of capture/compare control register 0 (CRC0) must be set to 1.

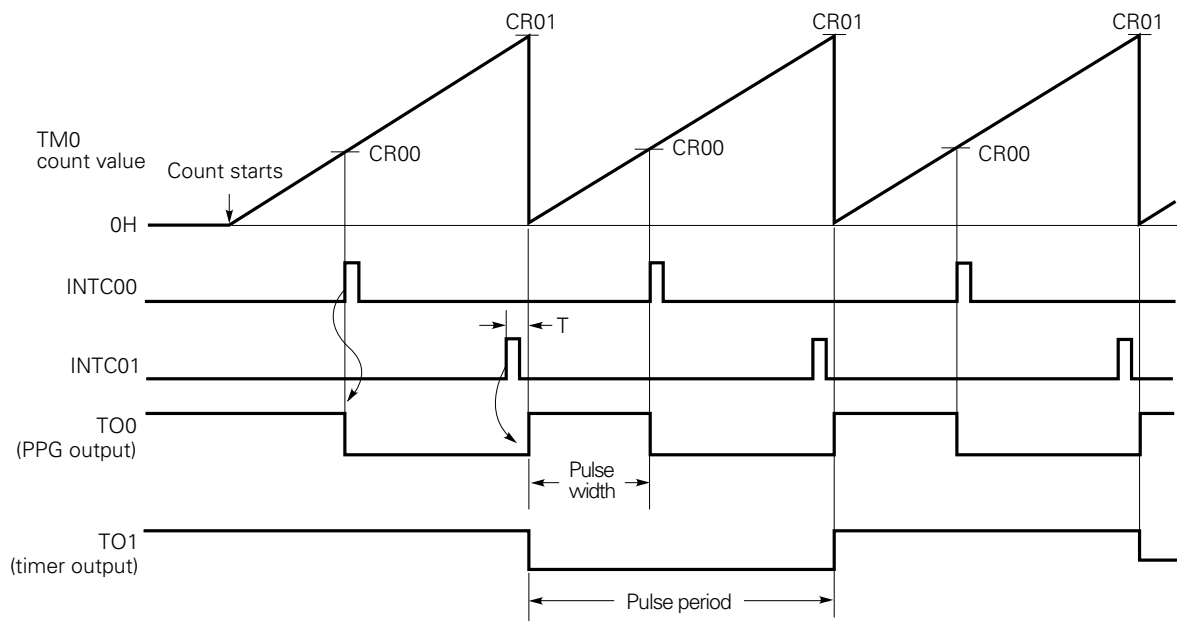
The pulse period and pulse width are as follows:

- PPG period = ((value set in CR01 compare register) + 1) × 8/f_{CLK}
- PPG pulse width = ((value set in CR00 compare register) × 8 + 2)/f_{CLK} ≐ (value set in CR00) × 8/f_{CLK}
where, CR00 ≤ CR01
- Duty factor = (PPG pulse width)/(PPG period) = ((value set in CR00) × 8 + 2)/(((value set in CR01) + 1) × 8) ≐ (value set in CR00)/((value set in CR01) + 1)

Caution In PPG output, the actual pulse width is longer than a value obtained with the approximate expression by two clock pulses of f_{CLK} for the active level, and is shorter than such an approximate value by two clock pulses of f_{CLK} for the inactive level. Take this point into consideration when high-precision output is required or the PPG pulse period is short.

Fig. 7-18 shows an example of PPG output using 16-bit timer 0. Fig. 7-19 shows an example of PPG output when CR00 = CR01. Fig. 7-20 shows an example of PPG output when CR00 = 0000H.

Fig. 7-18 Example of PPG Output Using TM0



Remark ALV0 = 0, ALV1 = 0

Table 7-8 PPG Output on TO0 (f_{CLK} = 6 MHz)

Count clock	Minimum pulse width ^{Note}	PPG period	PPG frequency
f _{CLK} /8	1.3 μs	2.6 μs - 87.4 ms	385 kHz - 11.4 Hz

Note The case where CR00 = 0 is excluded.

Fig. 7-19 PPG Output When CR00 = CR01

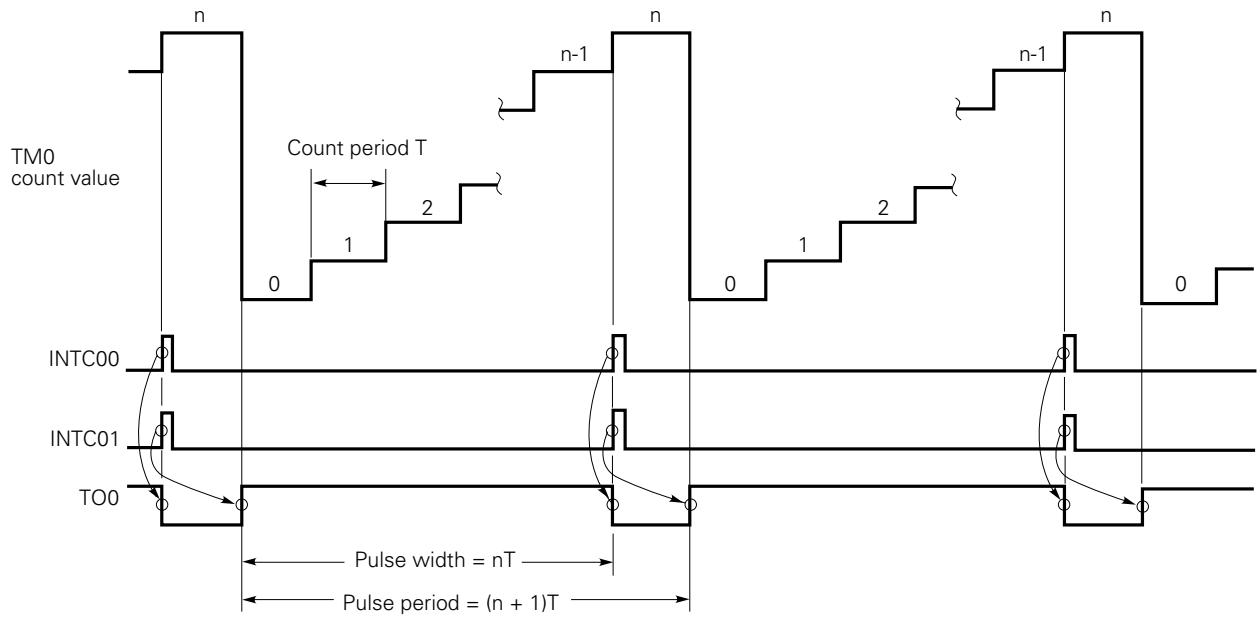
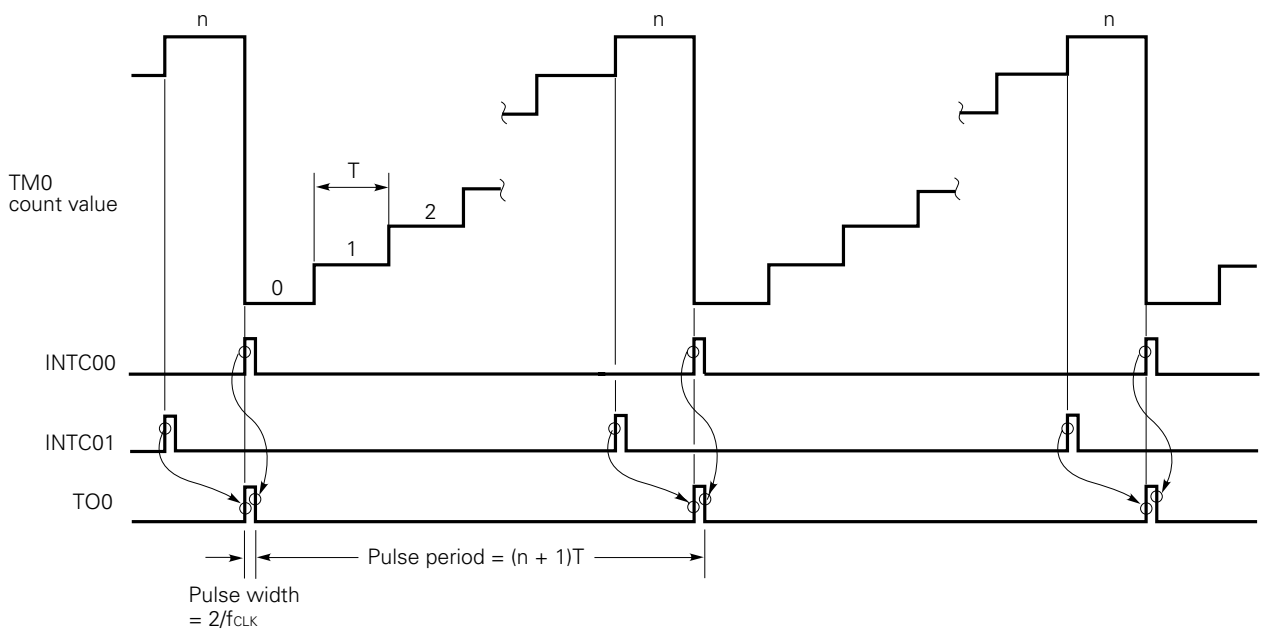


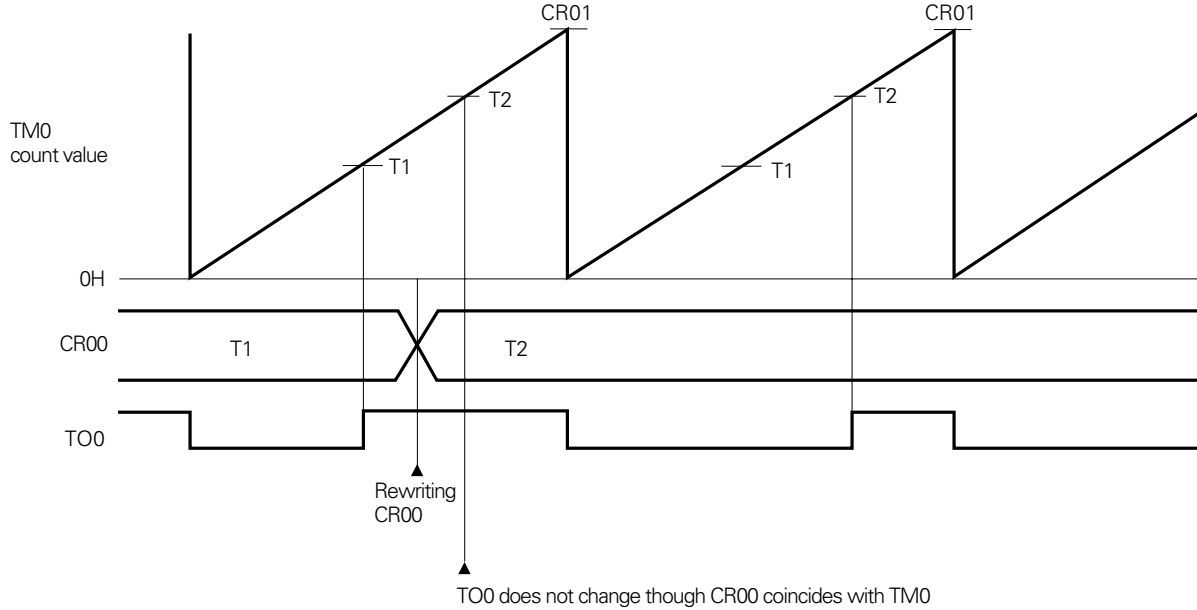
Fig. 7-20 PPG Output When CR00 = 0000H



Remark ALV0 = 0

Even if the value of the CR00 compare register coincides with the value of 16-bit timer 0 (TM0) more than once during one period of PPG output, the output levels on the timer outputs (TO0, TO1) are not inverted.

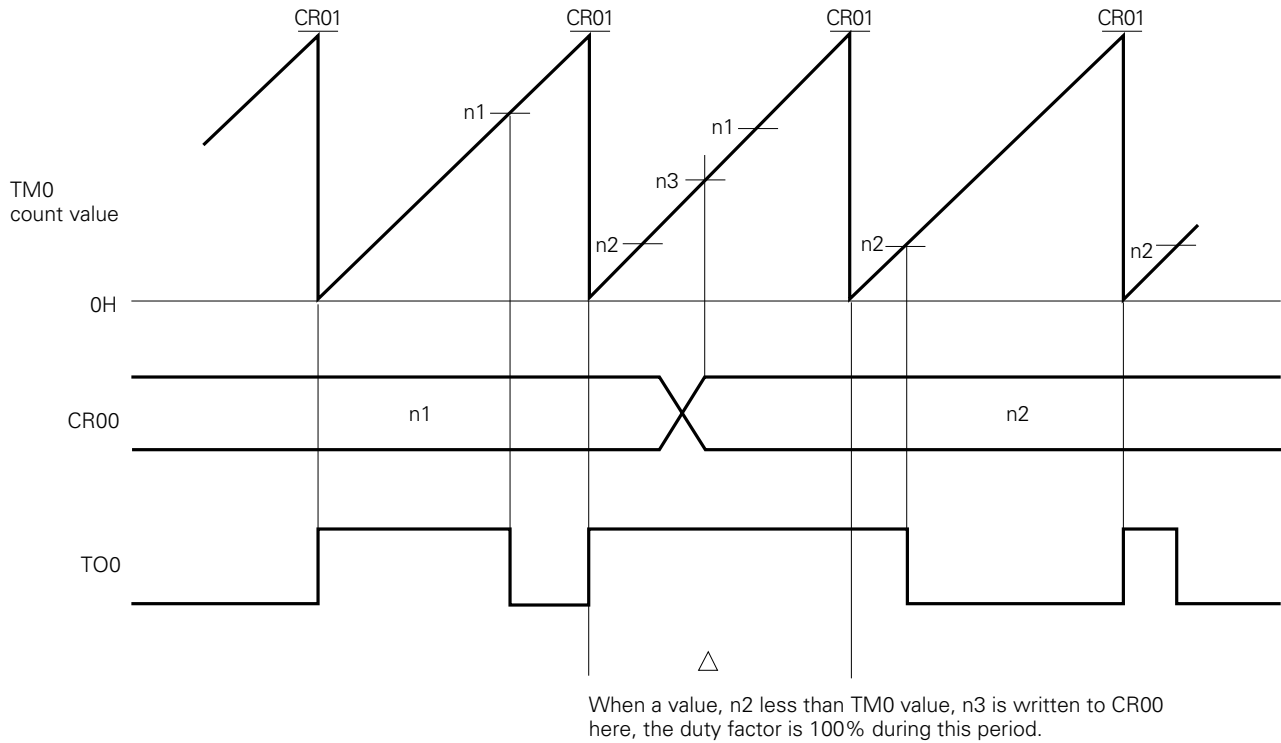
Fig. 7-21 Example of Rewriting Compare Register CR00



Remark ALV0 = 1

Cautions 1. If a value less than the value of 16-bit timer 0 (TM0) is written into the CR00 compare register before the value of CR00 coincides with the value of TM0, a PPG signal with a 100% duty factor is output in that period. Rewrite CR00, if required, by using an interrupt generated by a coincidence between TM0 and CR00.

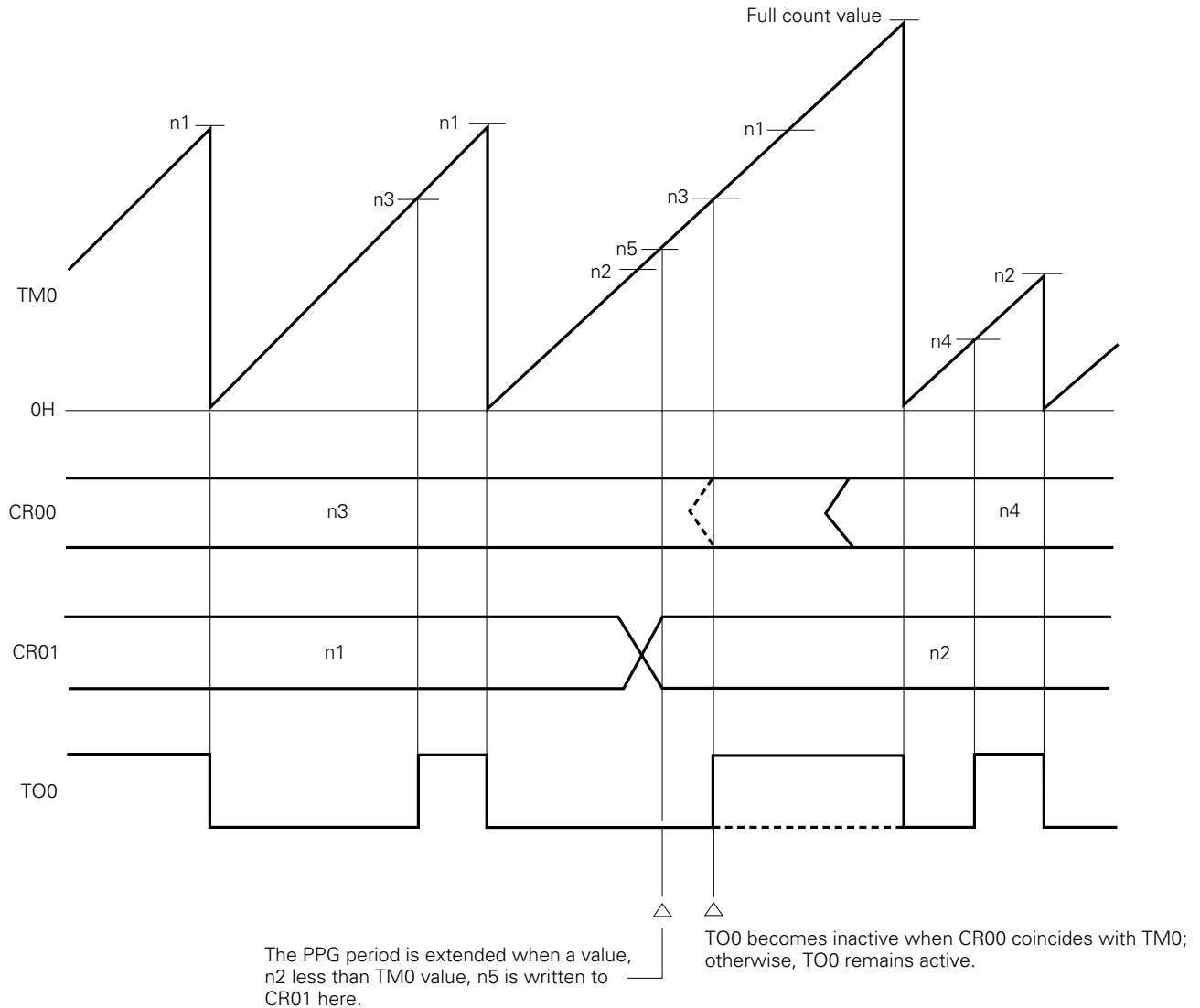
Fig. 7-22 Example of PPG Output Signal with a 100% Duty Factor



Remark ALV0 = 0

2. If the current value of the CR01 compare register is decreased below the value of 16-bit timer 0 (TM0), the PPG period becomes as long as the full-count time of TM0. At this time, if CR01 is rewritten after the value of the CR00 compare register coincides with the value of TM0, the inactive level is output until TM0 overflows to 0, then normal PPG output is resumed. If CR01 is rewritten before the value of CR00 coincides with the value of TM0, the active level is output until the value of CR00 coincides with the value of TM0. When the value of CR00 coincides with the value of TM0 before TM0 overflows to 0, the inactive level is output at that time. When TM0 overflows to 0, the active level is output, and normal PPG output is resumed. Rewrite CR01, if required, by using an interrupt generated by a coincidence between TM0 and CR01.

Fig. 7-23 Example of PPG Output Period Made Longer



Remark ALV0 = 1

3. If the PPG period is too short for interrupt acceptance, the measures described in Cautions 1 and 2 above do not lead to solution. Consider other measures (such as masking all interrupts and polling interrupt request flags by software).
4. If timer output is disabled (ENTOn = 0: n = 0, 1), the output level on the TOn (n = 0, 1) pin is the inverted value of the value set in ALVn (n = 0, 1). Accordingly, note that if timer output is disabled when the PPG output function is selected, the active level is output.

7.1.9 Sample Applications

(1) Interval timer operation (1)

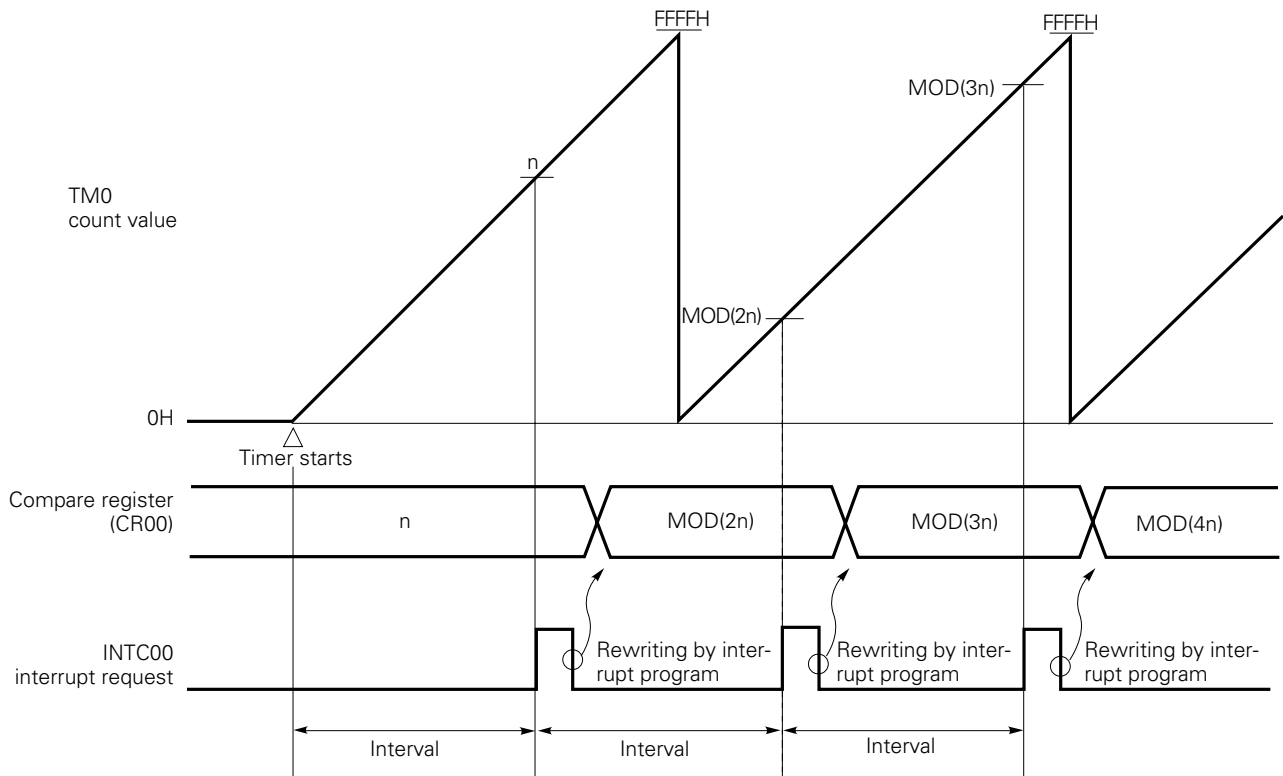
By free running 16-bit timer 0 (TM0), and adding a value to a compare register (CR00, CR01) in an interrupt handling routine, the 16-bit timer/counter can be used as an interval timer whose period is as long as the added value. (See Fig. 7-24.)

This interval timer has a resolution of 1.3 μ s, and can count up to 87.4 ms (at internal system clock $f_{CLK} = 6$ MHz).

In addition, 16-bit timer 0 (TM0) has two compare registers, so that interval timers with two types of periods can be produced.

Fig. 7-25 shows the setting of control registers. Fig. 7-26 shows the setting procedure. Fig. 7-27 shows interrupt handling.

Fig. 7-24 Timing of Interval Timer Operation (1)



Remark Interval = $n \times 8/f_{CLK}$, $1 \leq n \leq FFFFH$

Fig. 7-25 Setting of Control Registers for Interval Timer Operation (1)

★

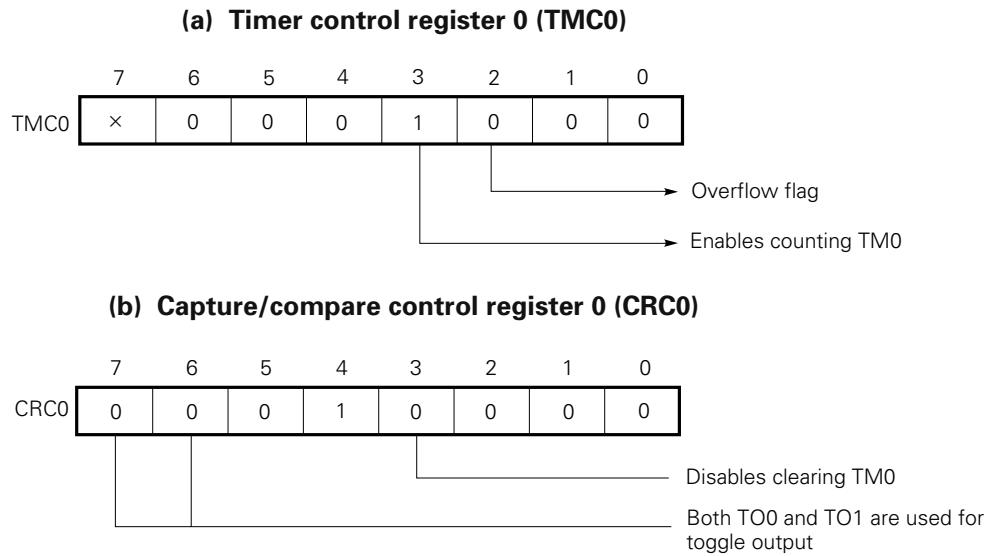


Fig. 7-26 Setting Procedure for Interval Timer Operation (1)

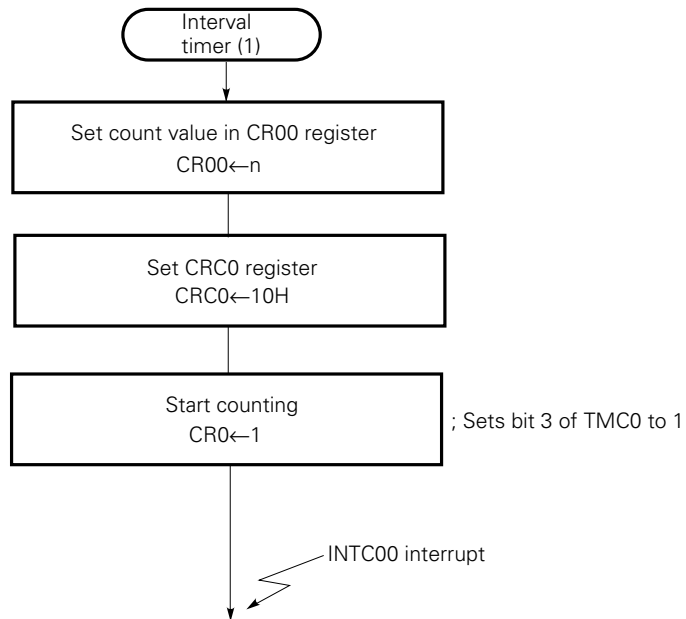
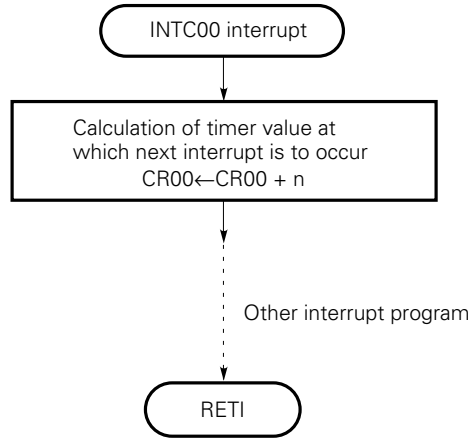


Fig. 7-27 Interrupt Request Handling for Interval Timer Operation (1)



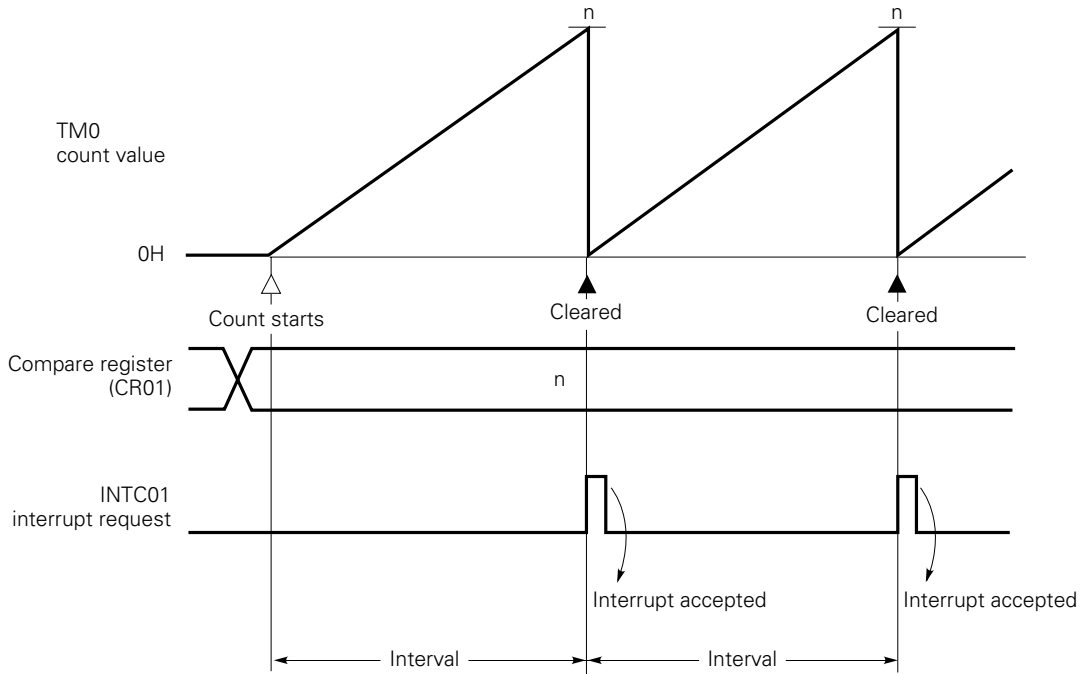
(2) Interval timer operation (2)

The 16-bit timer/counter can be used as an interval timer that generates an interrupt at intervals of a count time specified beforehand. (See Fig. 7-28.)

This interval timer has a resolution of 1.3 μs, and can count up from 1.3 μs to 87.4 ms (at internal system clock f_{CLK} = 6 MHz).

Fig. 7-29 shows the setting of control registers, and Fig. 7-30 shows the setting procedure.

Fig. 7-28 Timing of Interval Timer Operation (2)



Remark Interval = (n + 1) × 8/f_{CLK}, 0 ≤ n ≤ FFFFH

Fig. 7-29 Setting of Control Registers for Interval Timer Operation (2)

★

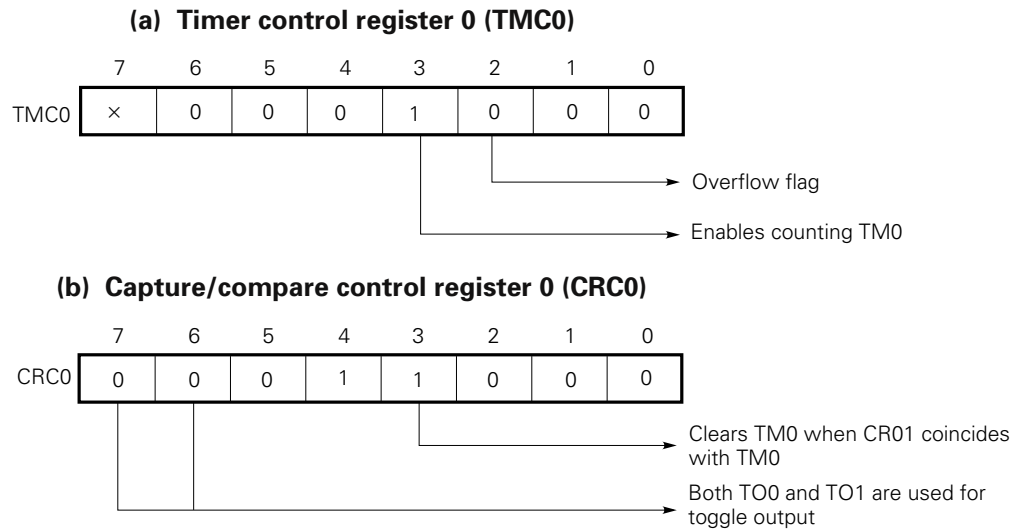
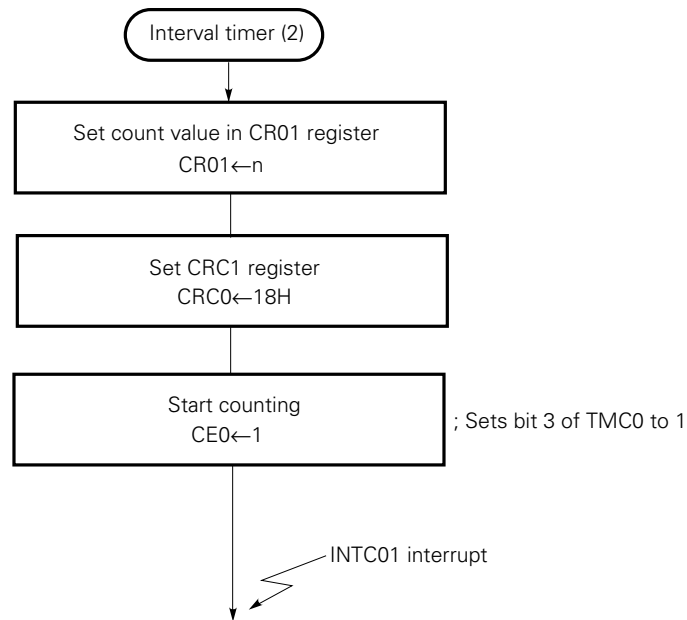


Fig. 7-30 Setting Procedure for Interval Timer Operation (2)



(3) Pulse width measurement operation

In pulse width measurement, the width of the high level or low level of an external pulse signal applied to the external interrupt request (INTP3) input pin is measured.

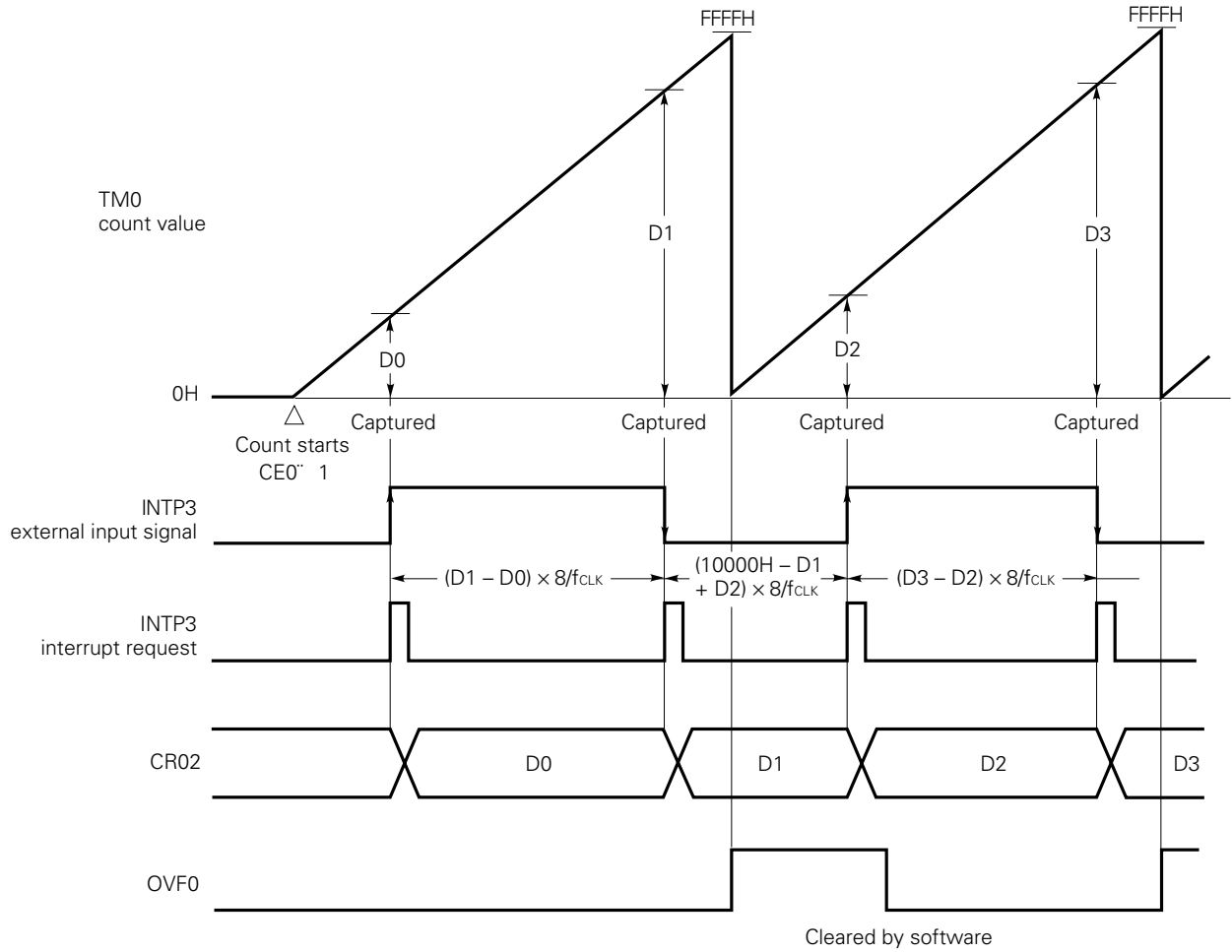
A pulse signal applied to the INTP3 pin must have a pulse width of 12 system clock pulses (2 μs: f_{CLK} = 6 MHz) or more for both the high level and the low level. If the pulse width is less than this value, no valid edge can be detected, thus resulting in a failure to perform capture operation.

This pulse width measurement allows a pulse width of 2.6 μs to 87.4 ms to be measured with a resolution of 1.3 μs (at f_{CLK} = 6 MHz).

As shown in Fig. 7-31, the value of 16-bit timer 0 (TM0) in count operation is loaded and held in the CR02 capture register on a valid edge (either a rising edge or falling edge) of a signal applied to the INTP3 pin. The pulse width of the input signal is found by multiplying the count clock (8/f_{CLK}) by the difference between the TM0 count value (D_n) loaded and held in the CR02 register on the n-th valid edge detected and the TM0 count value (D_{n-1}) loaded and held in the CR02 register on the (n - 1)-th valid edge detected.

Fig. 7-32 shows the setting of control registers, and Fig. 7-33 shows the setting procedure.

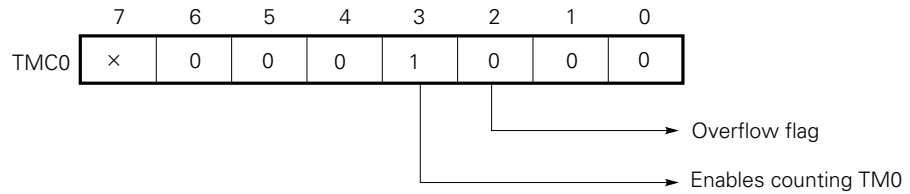
Fig. 7-31 Timing of Pulse Width Measurement



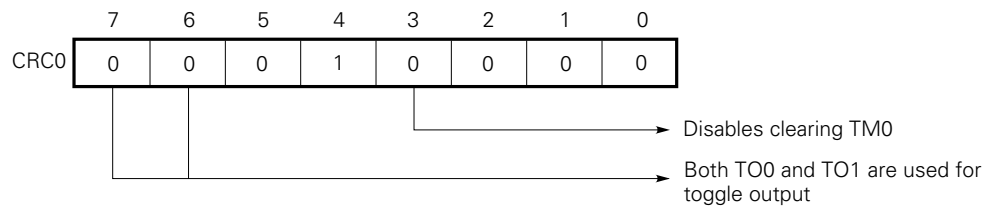
Remark D_n : TM0 count value ($n = 0, 1, 2, \dots$)

Fig. 7-32 Setting of Control Registers for Pulse Width Measurement

(a) Timer control register 0 (TMC0)



(b) Capture/compare control register 0 (CRC0)



(c) External interrupt mode register 1 (INTM1)

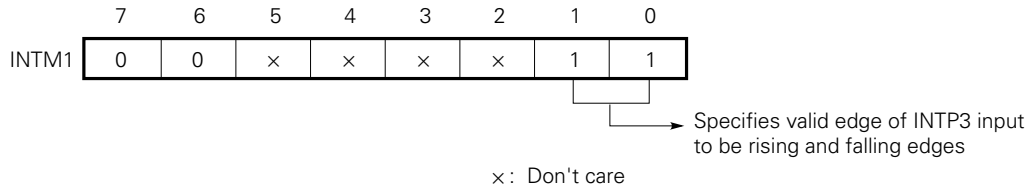


Fig. 7-33 Setting Procedure for Pulse Width Measurement

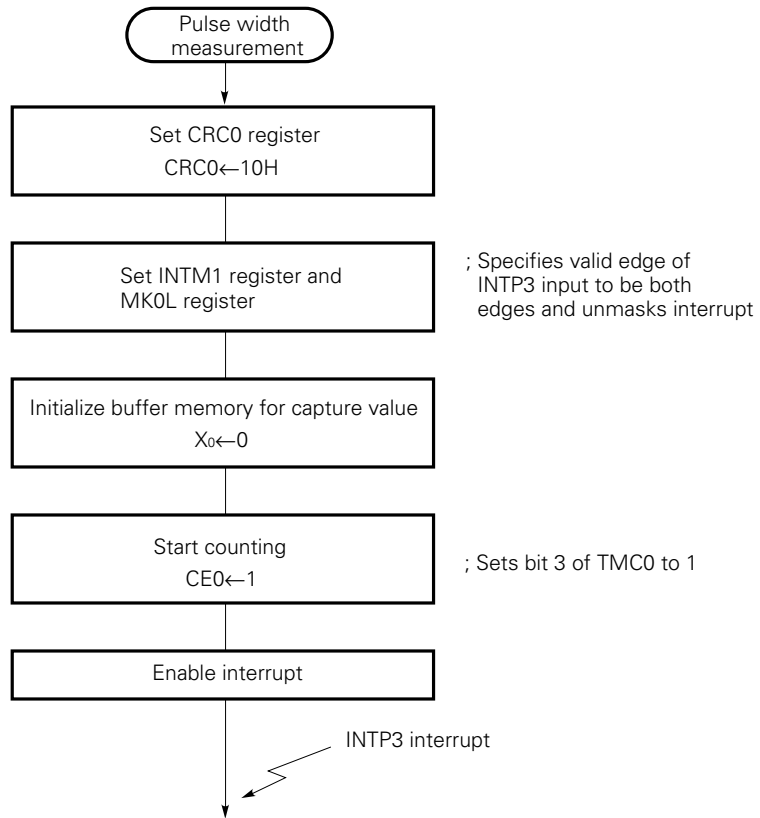
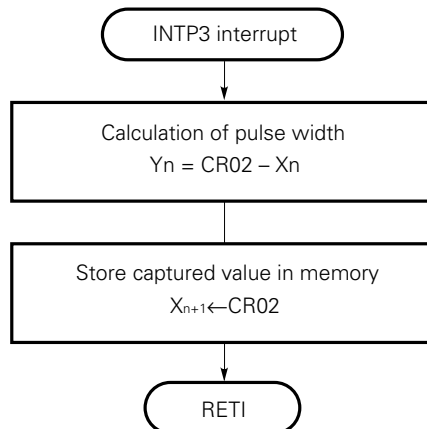


Fig. 7-34 Interrupt Request Handling for Pulse Width Calculation



(4) PWM output operation

In PWM output operation, a pulse signal with a duty factor determined by the value set in a compare register is output. (See Fig. 7-35.)

The duty factor of a PWM output signal can be changed in steps of 1/65536 from 1/65536 to 65535/65536. In addition, 16-bit timer 0 (TM0) has two compare registers, so that two types of PWM signals can be output. Fig. 7-36 shows the setting of control registers. Fig. 7-37 shows the setting procedure. Fig. 7-38 shows the procedure for changing the duty factor of PWM output.

Fig. 7-35 Example of PWM Signal Output by 16-Bit Timer/Counter

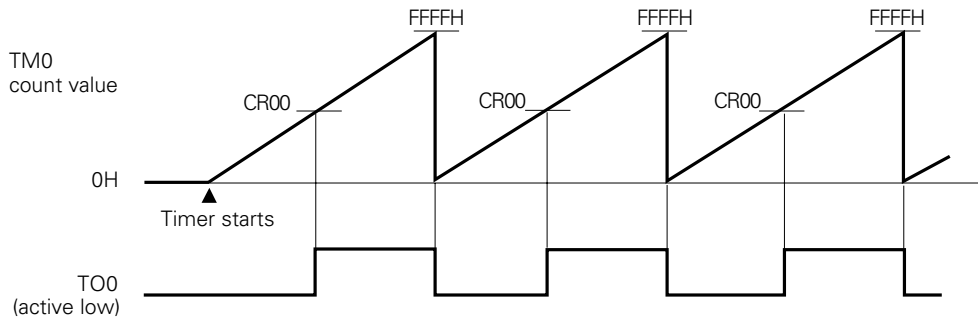


Fig. 7-36 Setting of Control Registers for PWM Output Operation

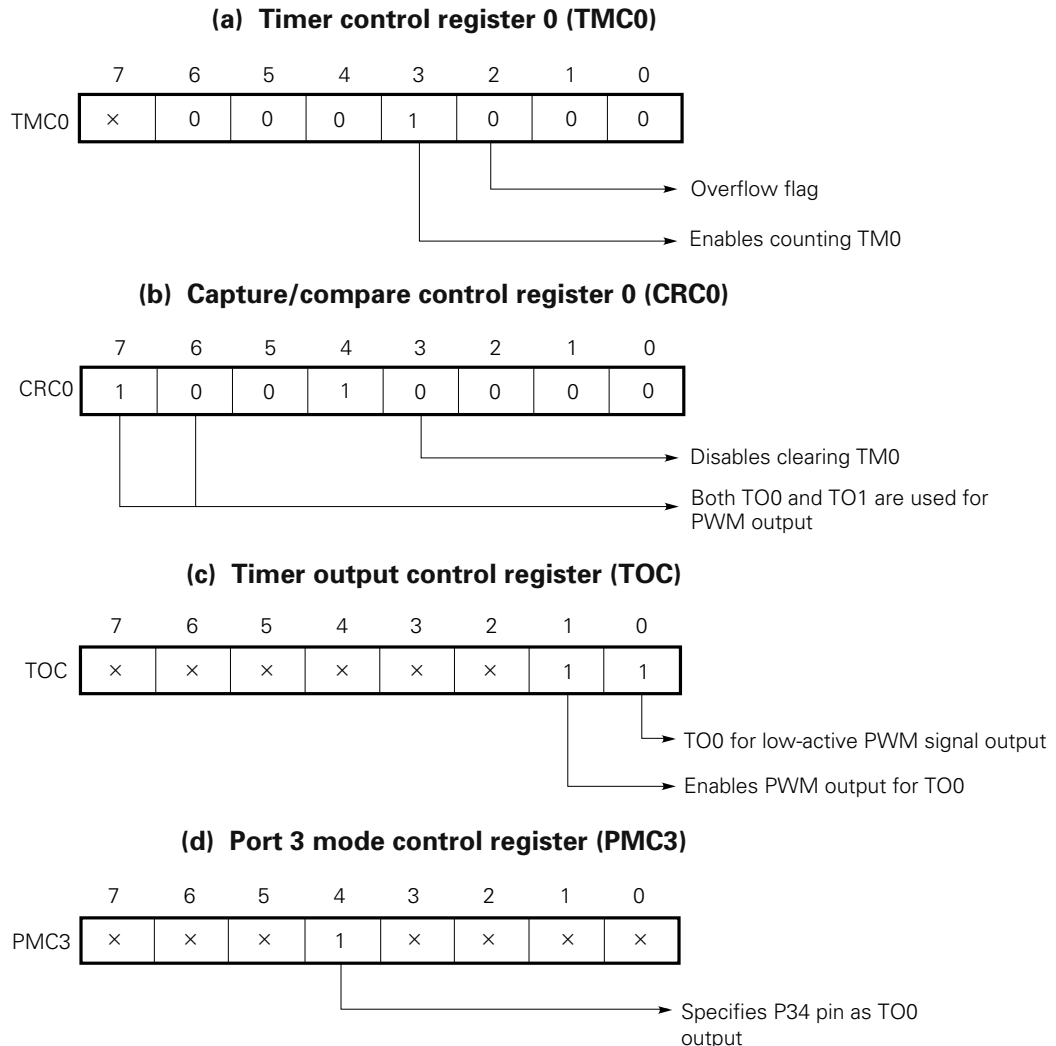


Fig. 7-37 Setting Procedure for PWM Output

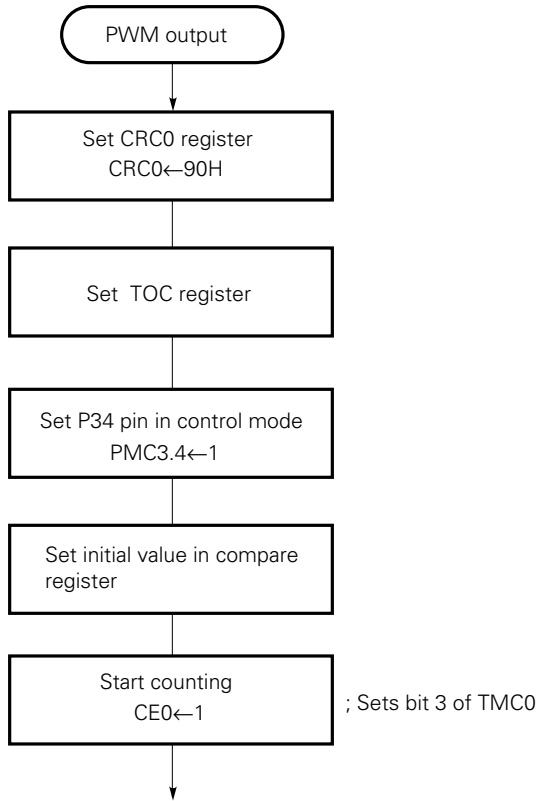
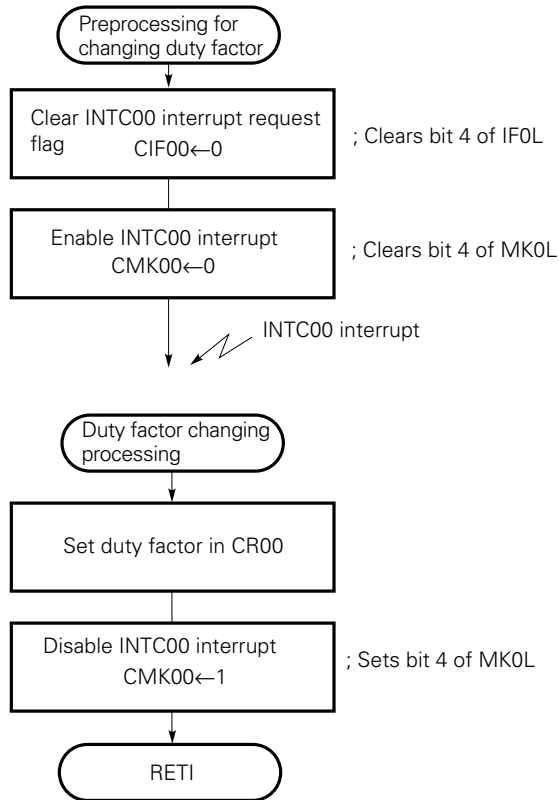


Fig. 7-38 Changing Duty Factor of PWM Output



(5) PPG output operation

In PPG output operation, a pulse signal with a period and duty factor determined by the values set in the compare registers is output. (See Fig. 7-39.)

Fig. 7-40 shows the setting of control registers. Fig. 7-41 shows the setting procedure. Fig. 7-42 shows the procedure for changing the duty factor of PPG output.

Fig. 7-39 Example of PPG Signal Output by 16-Bit Timer/Counter

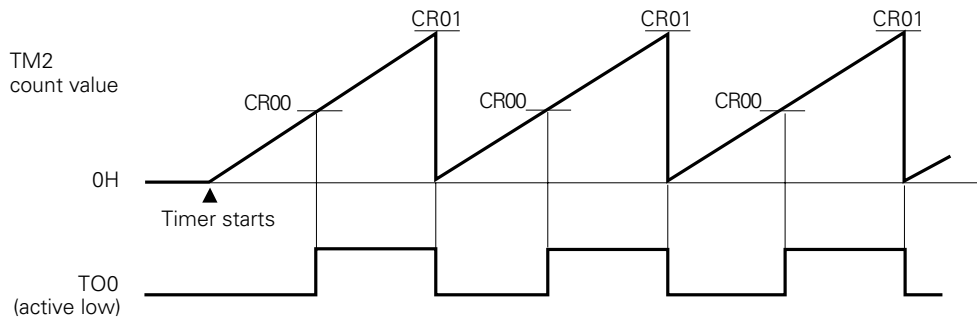
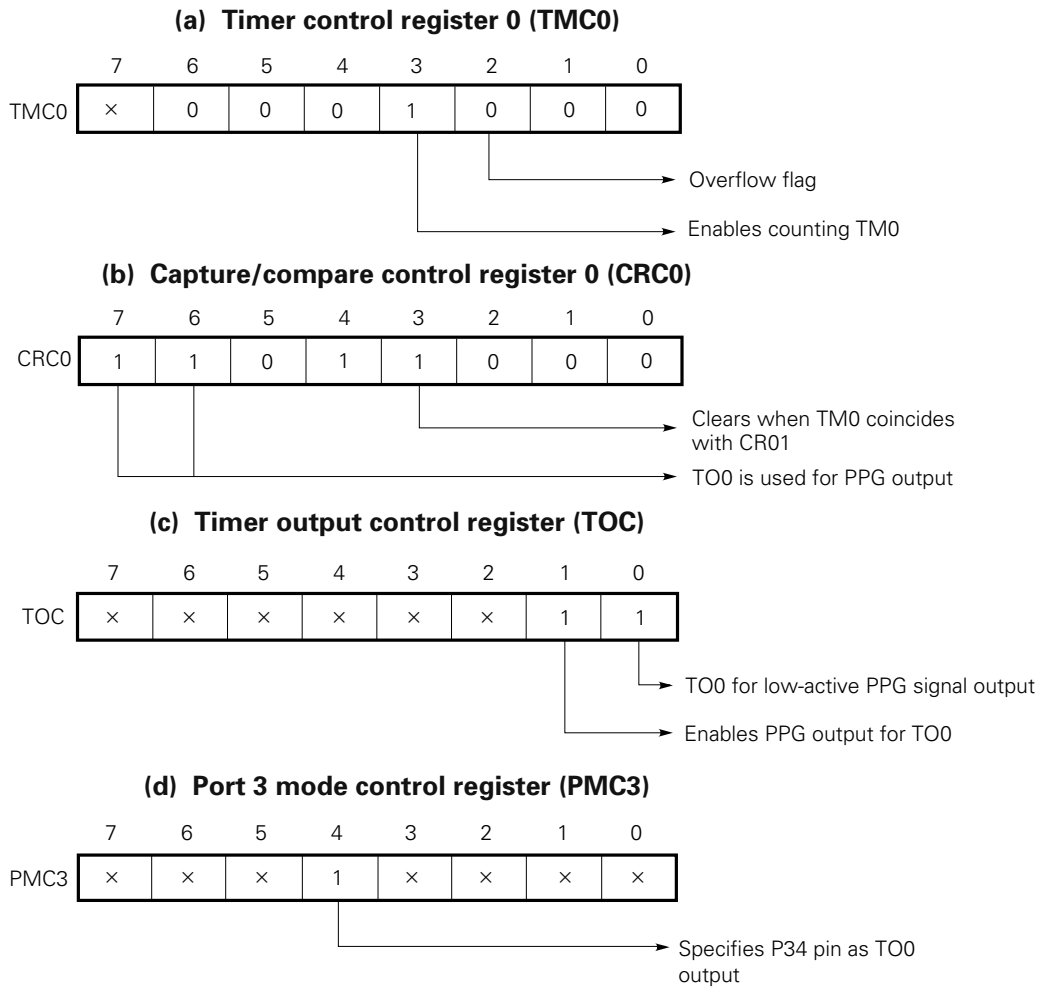


Fig. 7-40 Setting of Control Registers for PPG Output Operation



7

★

Fig. 7-41 Setting Procedure for PPG Output

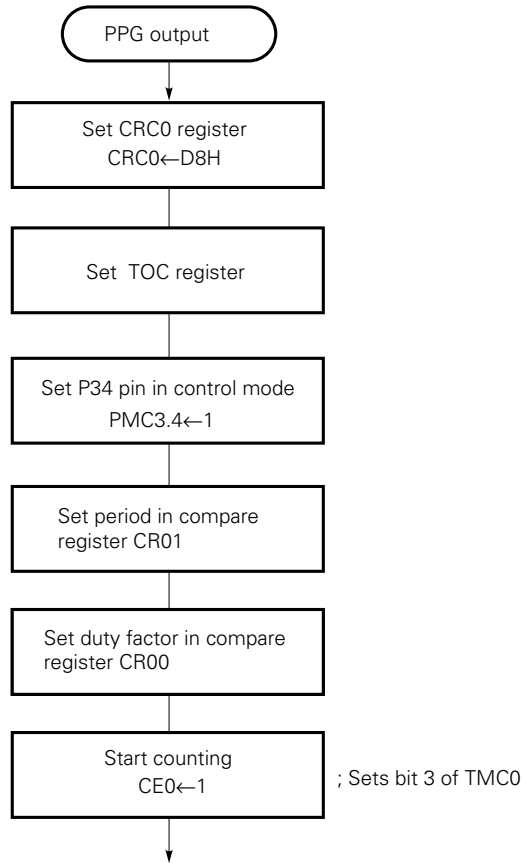
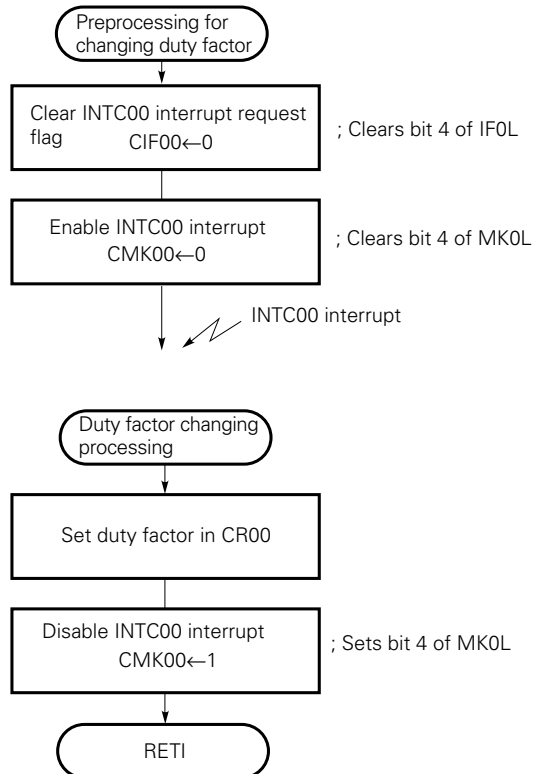


Fig. 7-42 Changing Duty Factor of PPG Output



7.2 8-BIT TIMER/COUNTER 1

7.2.1 Functions

Eight-bit timer/counter 1 can function as an interval timer and can also be used for pulse width measurement. In addition to these basic functions, 8-bit timer/counter 1 can be used as a timer for generating an output trigger on a real-time output port.

(1) Interval timer

When operating as an interval timer, 8-bit timer/counter 1 generates an internal interrupt at specified intervals.

Table 7-9 Intervals of 8-Bit Timer/Counter 1

Resolution	Minimum interval	Maximum interval
$16/f_{\text{CLK}}$ (2.6 μs)	$16/f_{\text{CLK}}$ (2.6 μs)	$2^8 \times 16/f_{\text{CLK}}$ (683 μs)
$32/f_{\text{CLK}}$ (5.3 μs)	$32/f_{\text{CLK}}$ (5.3 μs)	$2^8 \times 32/f_{\text{CLK}}$ (1.37 ms)
$64/f_{\text{CLK}}$ (10.7 μs)	$64/f_{\text{CLK}}$ (10.7 μs)	$2^8 \times 64/f_{\text{CLK}}$ (2.73 ms)
$128/f_{\text{CLK}}$ (21.3 μs)	$128/f_{\text{CLK}}$ (21.3 μs)	$2^8 \times 128/f_{\text{CLK}}$ (5.46 ms)
$256/f_{\text{CLK}}$ (42.7 μs)	$256/f_{\text{CLK}}$ (42.7 μs)	$2^8 \times 256/f_{\text{CLK}}$ (10.9 ms)
$512/f_{\text{CLK}}$ (85.3 μs)	$512/f_{\text{CLK}}$ (85.3 μs)	$2^8 \times 512/f_{\text{CLK}}$ (21.8 ms)

The values in parentheses are based on $f_{\text{CLK}} = 6 \text{ MHz}$.

(2) Pulse width measurement

Eight-bit timer/counter 1 measures the pulse width of a signal applied to the external interrupt request input pin INTPO.

Table 7-10 Pulse Width Measurement Range of 8-Bit Timer/Counter 1

Measurable pulse width	Resolution
$\leq 2^8 \times 16/f_{\text{CLK}}$ (683 μs)	$16/f_{\text{CLK}}$ (2.6 μs)
$\leq 2^8 \times 32/f_{\text{CLK}}$ (1.37 ms)	$32/f_{\text{CLK}}$ (5.3 μs)
$\leq 2^8 \times 64/f_{\text{CLK}}$ (2.73 ms)	$64/f_{\text{CLK}}$ (10.7 μs)
$\leq 2^8 \times 128/f_{\text{CLK}}$ (5.46 ms)	$128/f_{\text{CLK}}$ (21.3 μs)
$\leq 2^8 \times 256/f_{\text{CLK}}$ (10.9 ms)	$256/f_{\text{CLK}}$ (42.7 μs)
$\leq 2^8 \times 512/f_{\text{CLK}}$ (21.8 ms)	$512/f_{\text{CLK}}$ (85.3 μs)

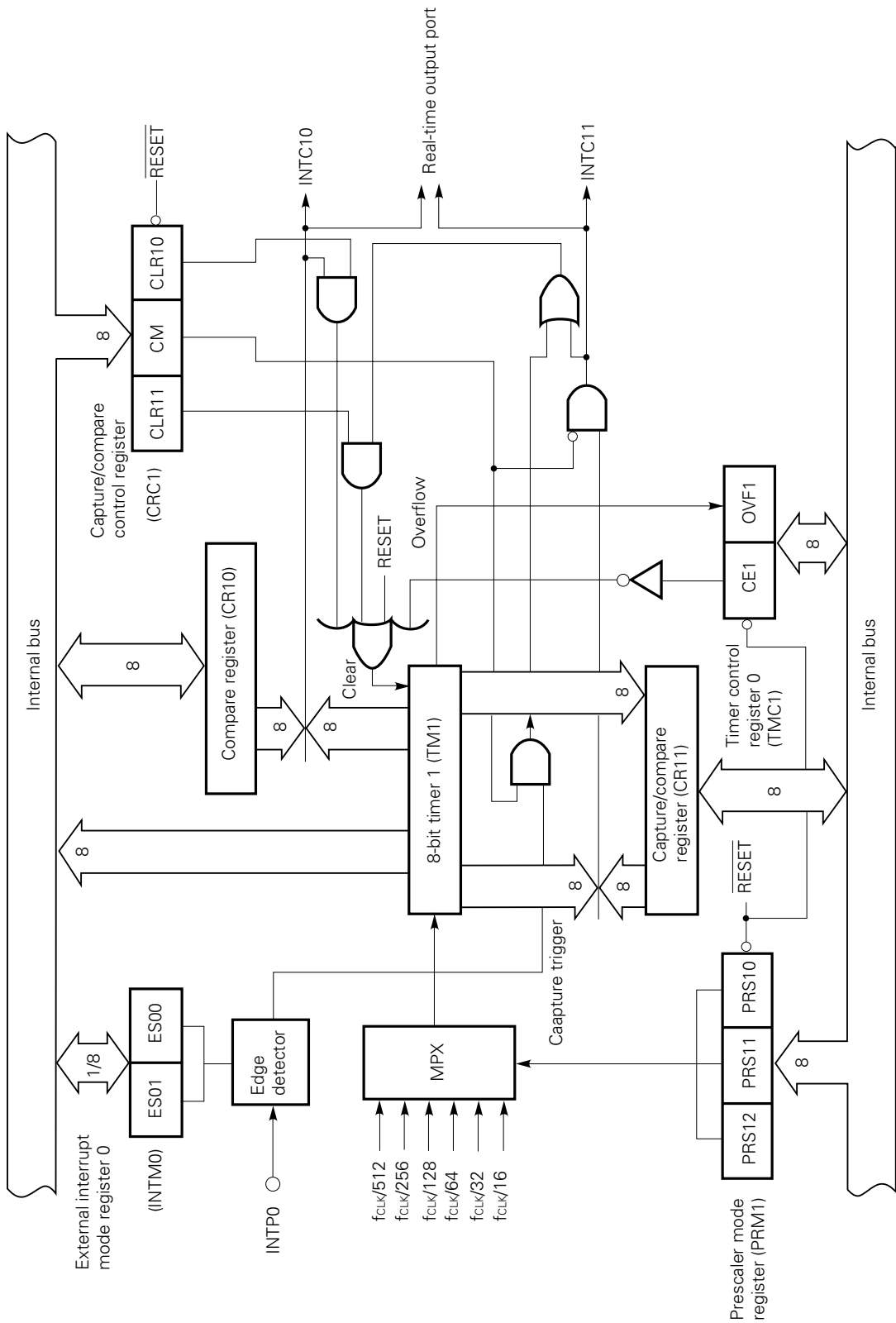
The values in parentheses are based on $f_{\text{CLK}} = 6 \text{ MHz}$.

7.2.2 Configuration

Eight-bit timer/counter 1 consists of one 8-bit timer 1 (TM1), one 8-bit compare register (CR10), and one 8-bit capture/compare register (CR11).

Fig. 7-43 shows the block diagram of 8-bit timer/counter 1.

Fig. 7-43 Block Diagram of 8-Bit Timer/Counter 1



(1) 8-bit timer 1 (TM1)

TM1 is a timer for counting up with the count clock specified by the lower 4 bits of prescaler mode register 1 (PRM1).

The count operation of TM1 can be enabled or disabled by timer control register 1 (TMC1).

TM1 allows only read operation using an 8-bit manipulation instruction. When the $\overline{\text{RESET}}$ signal is applied, TM1 is cleared to 00H, and count operation stops.

(2) Compare register (CR10)

The CR10 register is an 8-bit register for holding a value that determines the period of interval timer operation.

When the value of the CR10 register coincides with the value of TM1, an interrupt request (INTC10) is generated. This coincide with signal functions also as a real-time output port trigger signal. Count value clear operation can also be performed when the value of CR10 coincides with the value of TM1.

The compare register allows both read and write operations using an 8-bit manipulation instruction. When the $\overline{\text{RESET}}$ signal is applied, the compare register becomes undefined.

(3) Capture/compare register (CR11)

The CR11 register is an 8-bit register that can be set by capture/compare control register 1 (CRC1) as a compare register used to detect a coincidence with the count value of TM1 or as a capture register used to capture the count value of TM1.

(a) When CR11 is set as a compare register

The CR11 register functions as an 8-bit register for holding a value that determines the period of interval timer operation.

When the value of the CR11 register coincides with the value of TM1, an interrupt request (INTC11) is generated.

In addition, this coincide with signal functions also as a real-time output port trigger signal. Count value clear operation can also be performed when the value of CR11 coincides with the value of TM1.

(b) When CR11 is set as a capture register

The CR11 register functions as an 8-bit register that captures the value of TM1 on a valid edge (capture trigger) appearing on the external interrupt request (INTP0) input pin.

The value of the CR11 register is held until the next capture trigger occurs. After capture operation, TM1 can be cleared.

The CR11 register allows both read and write operations using an 8-bit manipulation instruction. When the $\overline{\text{RESET}}$ signal is applied, the CR11 register becomes undefined.

(4) Edge detector

The edge detector detects a valid edge of an external input signal.

When the edge detector detects, on the INTP0 input pin, a valid edge specified in external interrupt mode register 0 (INTM0), INTP0 and a capture trigger are generated. (See Fig. 11-1 for information about the INTM0 register.)

(5) Prescaler

The prescaler generates count clocks from the internal system clock. From these count clocks generated by the prescaler, a count clock is selected with the selector for the timer to perform count operation.

7.2.3 8-Bit Timer/Counter 1 Control Registers

(1) Timer control register 1 (TMC1)

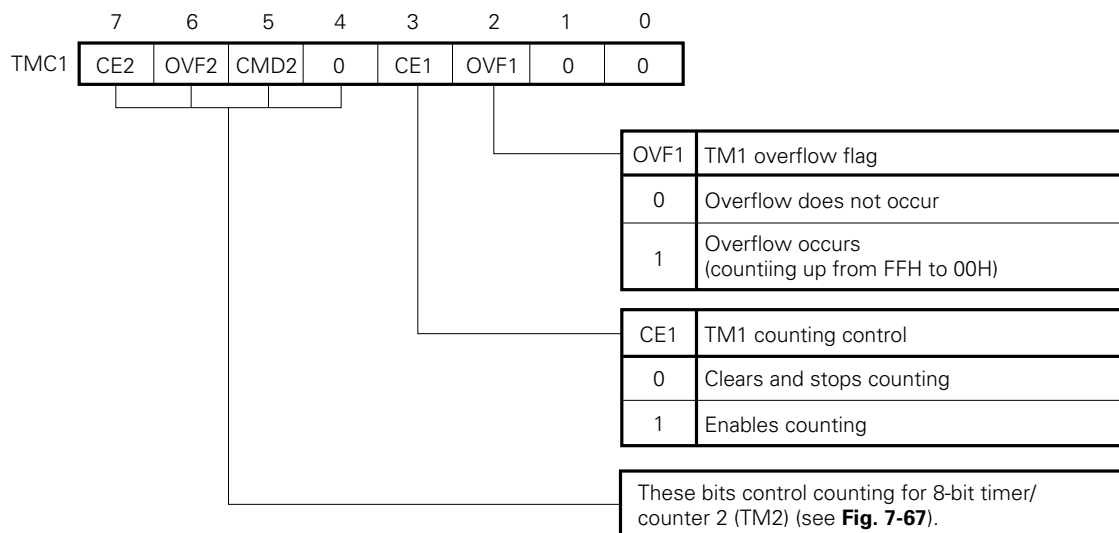
The TMC1 register is an 8-bit register for controlling the count operations of 8-bit timer 1 (TM1) and 8-bit timer 2 (TM2).

The lower 4 bits control the count operation of TM1 of 8-bit timer/counter 1. (The higher 4 bits control the count operation of TM2 of 8-bit timer/counter 2.)

The TMC1 register allows both read and write operations using an 8-bit manipulation instruction. Fig. 7-44 shows the format of the TMC1 register.

When the $\overline{\text{RESET}}$ signal is applied, the TMC1 register is cleared to 00H.

Fig. 7-44 Format of Timer Control Register 1 (TMC1)



Remark The OVF1 bit can be reset only by software.

(2) Prescaler mode register 1 (PRM1)

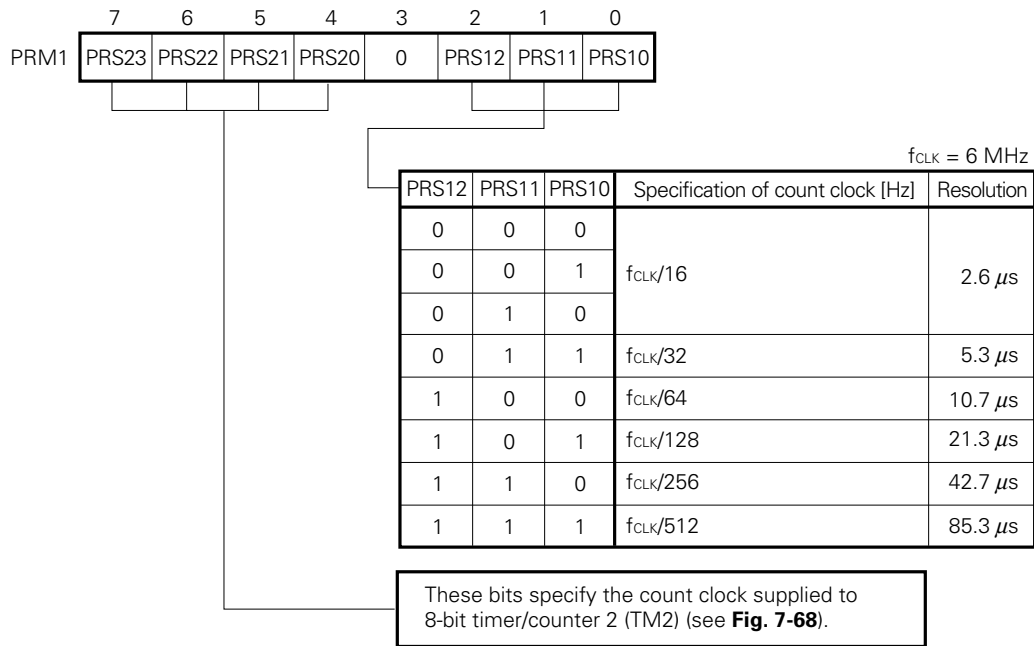
The PRM1 register is an 8-bit register used to specify a count clock for 8-bit timer 1 (TM1) and 8-bit timer 2 (TM2).

The lower 4 bits are used to specify a count clock for TM1 of 8-bit timer/counter 1. (The higher 4 bits are used to specify a count clock for TM2 of 8-bit timer/counter 2.)

The PRM1 register allows only write operation using an 8-bit manipulation instruction. Fig. 7-45 shows the format of the PRM1 register.

When the $\overline{\text{RESET}}$ signal is applied, the PRM1 register is cleared to 00H.

Fig. 7-45 Format of Prescaler Mode Register 1 (PRM1)



Remark f_{CLK}: System clock frequency

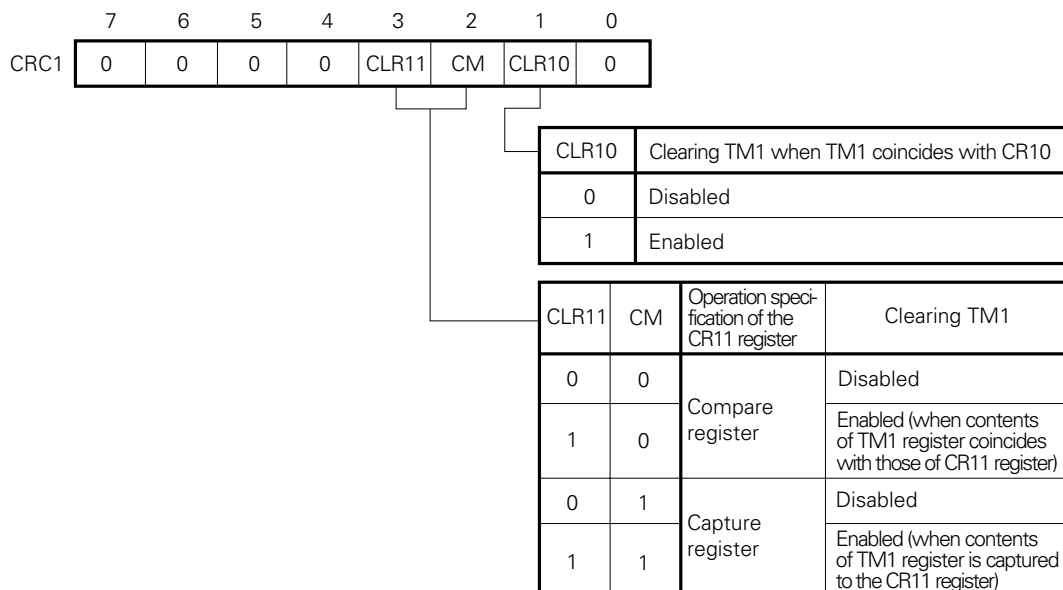
(3) Capture/compare control register 1 (CRC1)

The CRC1 register is used to specify the operation of the CR11 capture/compare register and the condition for enabling the clear operation of 8-bit timer 1 (TM1).

The CRC1 register allows only write operation using an 8-bit manipulation instruction. Fig. 7-46 shows the format of the CRC1 register.

When the $\overline{\text{RESET}}$ signal is applied, the CRC1 register is cleared to 00H.

Fig. 7-46 Format of Capture/Compare Control Register 1 (CRC1)



7.2.4 Operation of 8-Bit Timer 1 (TM1)

(1) Basic operation

Eight-bit timer/counter 1 performs count operation by counting up with the count clock specified by the lower 4 bits of prescaler mode register 1 (PRM1).

When the $\overline{\text{RESET}}$ signal is applied, TM1 is cleared to 00H, and count operation stops.

Bit 3 (CE1) of timer control register 1 (TMC1) is used to enable/disable count operation. (The lower 4 bits of the TMC1 register are used to control the operation of 8-bit timer/counter 1.) When the CE1 bit is set to 1 by software, TM1 is cleared to 00H by the first count clock pulse, then count-up operation starts.

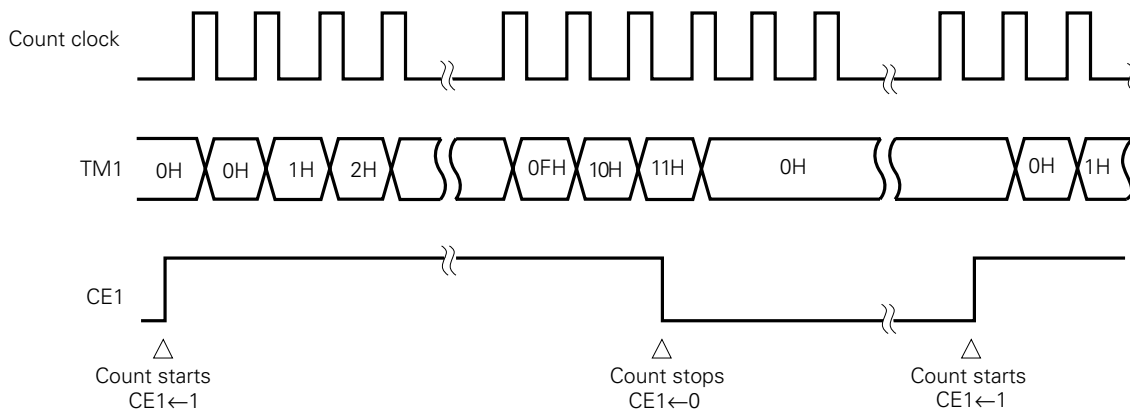
When the CE1 bit is reset to 0, TM1 is cleared to 00H by the next count clock pulse, then capture operation and coincide with signal generation stop.

If the CE1 bit is set to 1 when the CE1 bit is already set to 1, TM1 is not cleared, but continues count operation.

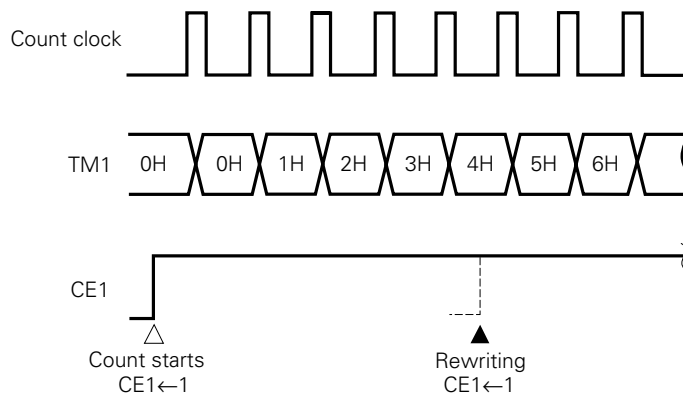
If the count clock signal is applied when the value of TM1 is FFH, TM1 is set to 00H. At this time, OVF1 is set. OVF1 can be cleared only by software. Count operation continues.

Fig. 7-47 Basic Operation of 8-Bit Timer 1 (TM1)

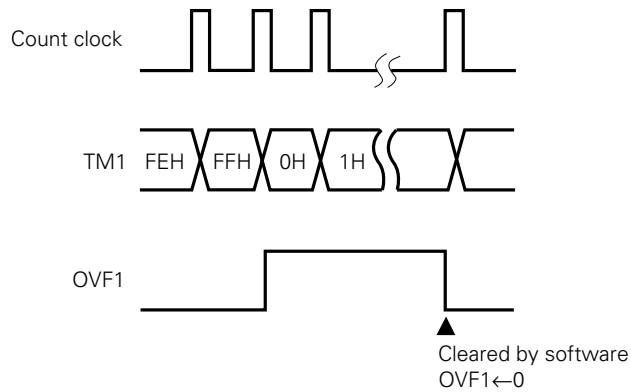
(a) Count start → count stop → count start



(b) When the CE1 bit is set to 1 again after count operation starts



(c) When the value of TM1 is FFH



(2) Clear operation

After a coincidence with a compare register (CR1m: m = 0, 1) or capture operation, 8-bit timer 1 (TM1) can be automatically cleared. If a TM1 clear cause occurs, TM1 is cleared to 00H by the next count clock pulse. This means that even if a TM1 clear cause occurs, TM1 holds the value existing at that time until the next count clock pulse is applied.

Fig. 7-48 TM1 Cleared by a Coincidence with Compare Register (CR1m)

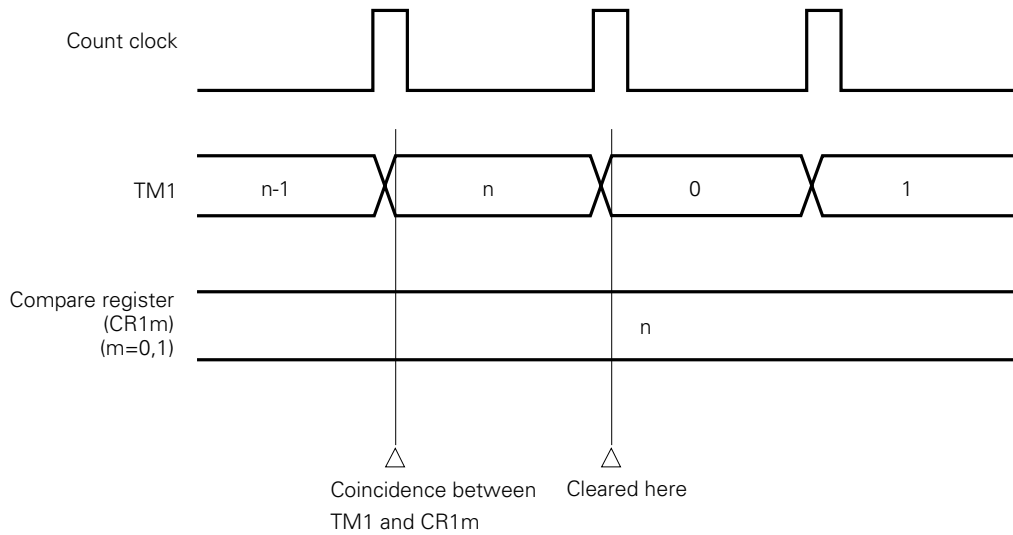
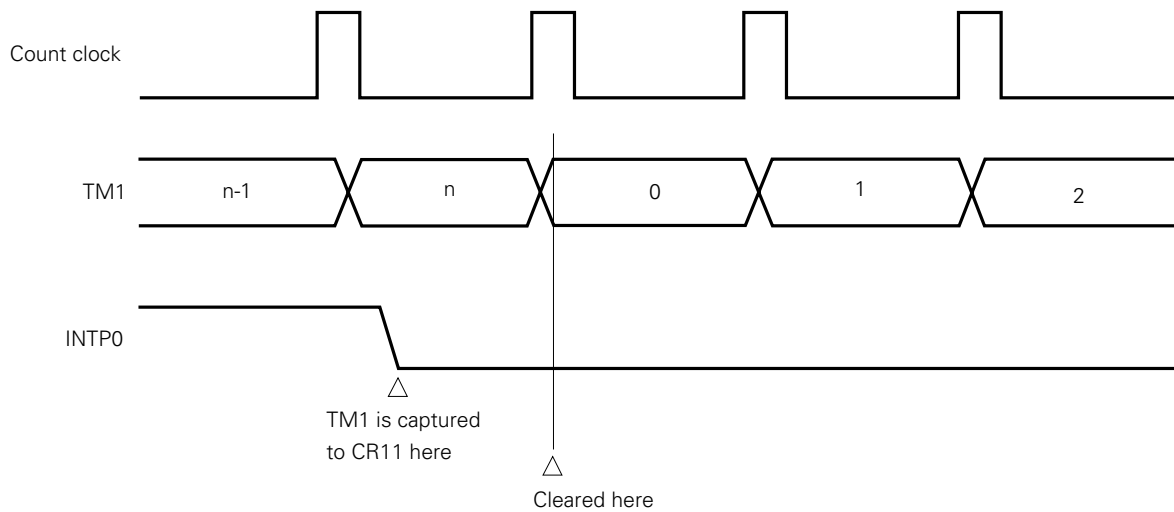
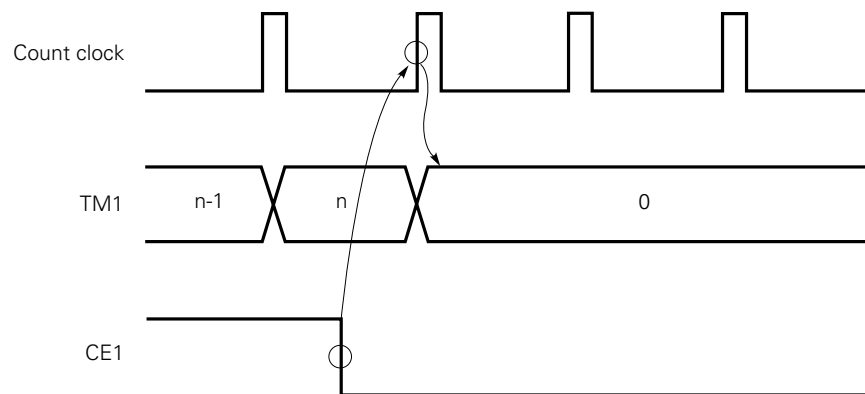
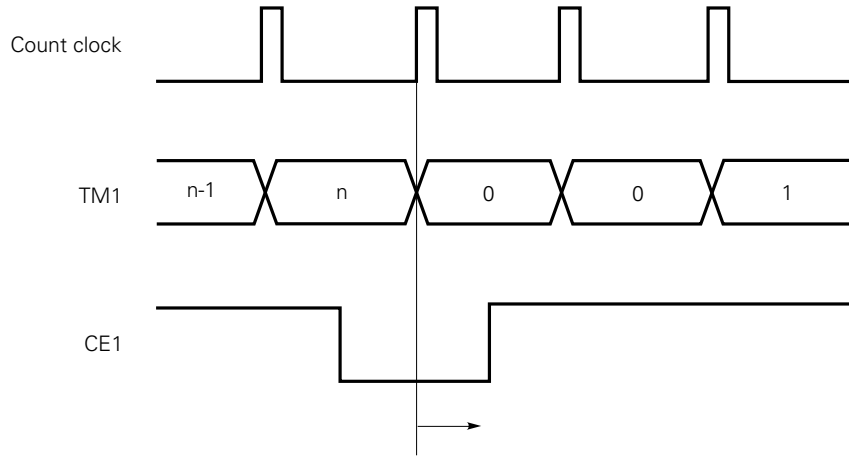


Fig. 7-49 TM1 Cleared after Capture Operation

TM1 can also be cleared by software when the CE1 bit of the timer control register (TMC1) is reset to 0. Similarly, clear operation is performed by the count clock pulse following the resetting of CE1 bit to 0. If the CE1 bit is set to 1 before TM1 is reset to 0 by the resetting of the CE1 bit to 0 (that is, before the first count clock pulse is applied after the CE1 bit is reset to 0), two operations are simultaneously performed: one operation is an operation to clear TM1 to 0, and the other operation is a count operation starting with the counting of 0.

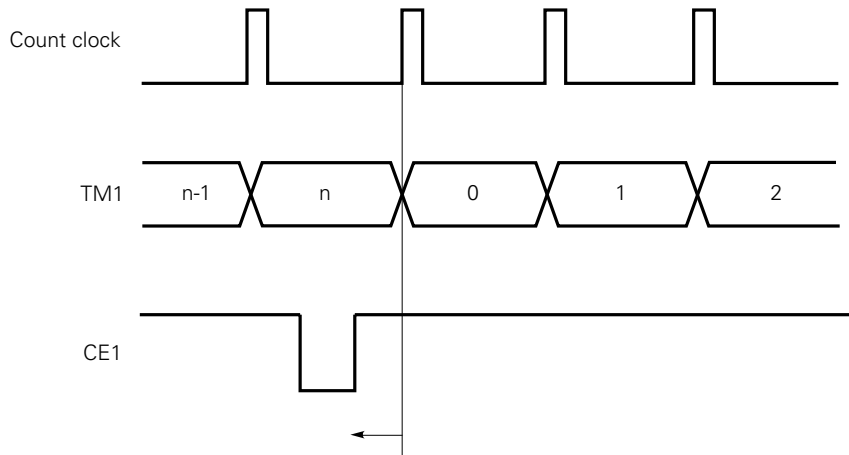
Fig. 7-50 Clear Operation When the CE1 Bit Is Reset to 0**(a) Basic operation**

(b) Restart after 0 is set in TM1 cleared



When the CE1 bit is set to 1 after this count clock, counting starts from 0 on the count clock input after the CE1 bit has been set.

(c) Restart before 0 is set in TM1 cleared



When the CE1 bit is set to 1 before this count clock, Clearing TM1 by CE1←0 and counting by CE1←1 are performed simultaneously.

7.2.5 Compare Register and Capture/Compare Register Operations

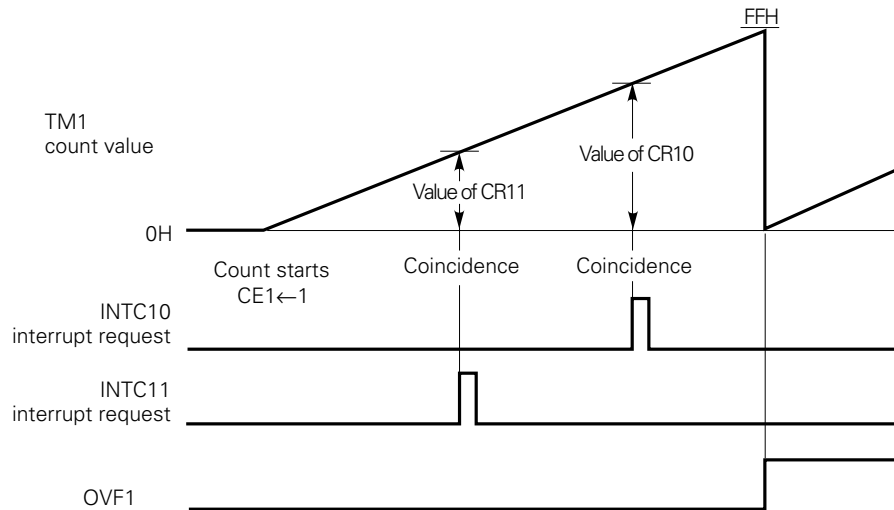
(1) Compare operation

Eight-bit timer/counter 1 performs an operation to compare the values set in the compare registers with timer count values.

When the value set in the compare register (CR10) and the value set in the capture/compare register (CR11) specified to perform compare operation coincide with count values of 8-bit timer 1 (TM1), the interrupt request signals (INTC10 for the CR10 register and INTC11 for the CR11 register) are generated.

After the value of the CR10 register or CR11 register coincides with a count value of TM1, the count value of TM1 can be cleared. Thus, 8-bit timer/counter 1 can operate as an interval timer for repeatedly counting up to the value set in the CR10 register or CR11 register.

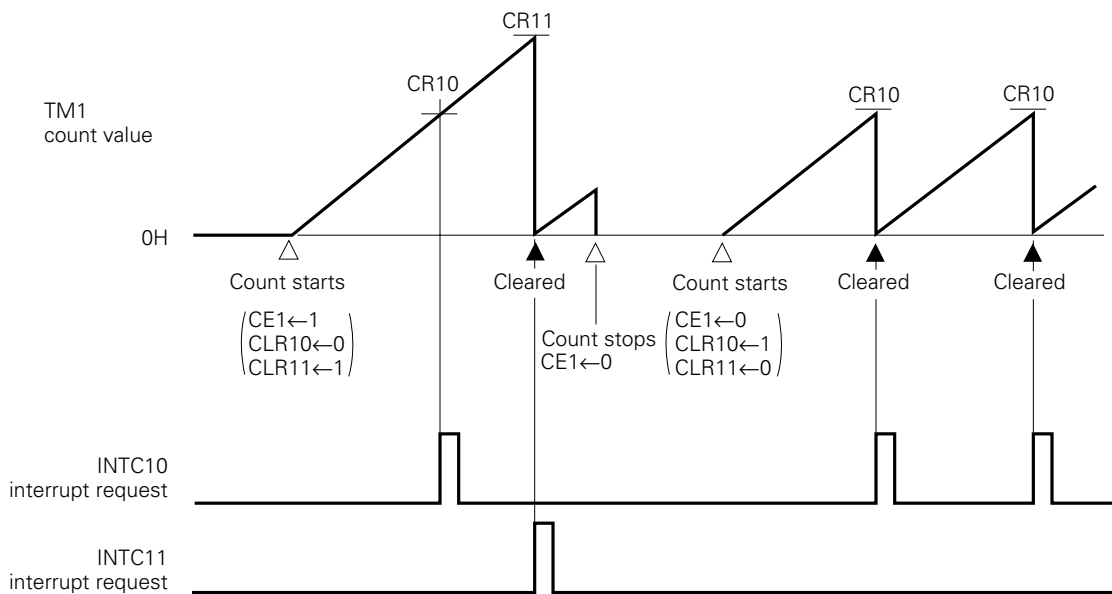
Fig. 7-51 Compare Operation



Remark CLR10 = 0, CLR11 = 0, CM = 0

Caution When using an in-circuit emulator, see the notes described in Section 7.5.4.

Fig. 7-52 TM1 Cleared After a Coincidence Is Detected



(2) Capture operation

Eight-bit timer/counter 1 performs a capture operation to load the count value of the timer into the capture register in synchronism with an external trigger.

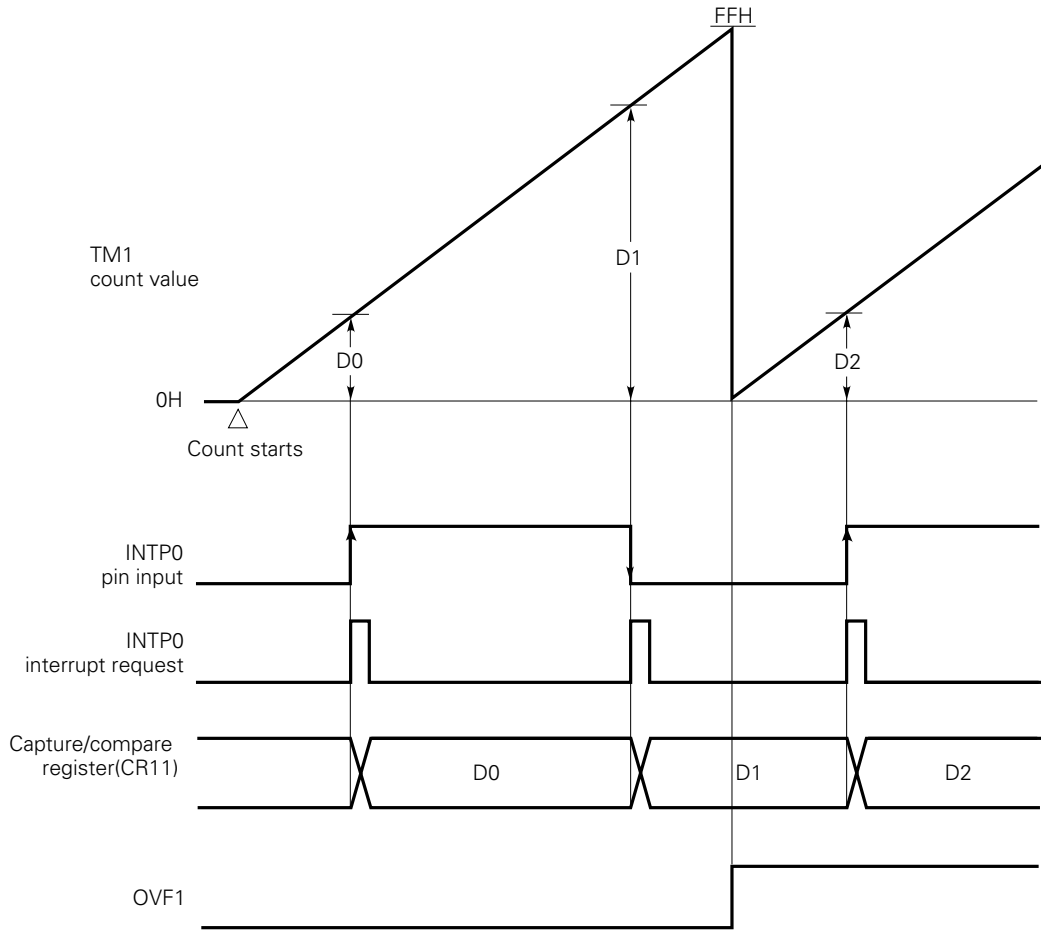
As an external trigger, a valid edge detected on the external interrupt request (INTP0) input pin is used (capture trigger). In synchronism with a capture trigger, the count value of 8-bit timer 1 (TM1) in count operation is loaded and held in the capture/compare register (CR11) specified to perform capture operation. Until the next capture trigger occurs, the value of the CR11 register is held.

A valid edge used as a capture trigger is set using external interrupt mode register 0 (INTM0). When both a rising edge and falling edge are set as capture triggers, the pulse width of an applied external signal can be measured. When a capture trigger is generated using one edge, the period of an input pulse signal can be measured.

For the detailed format of the INTM0 register, see Fig. 11-1 in Chapter 11.

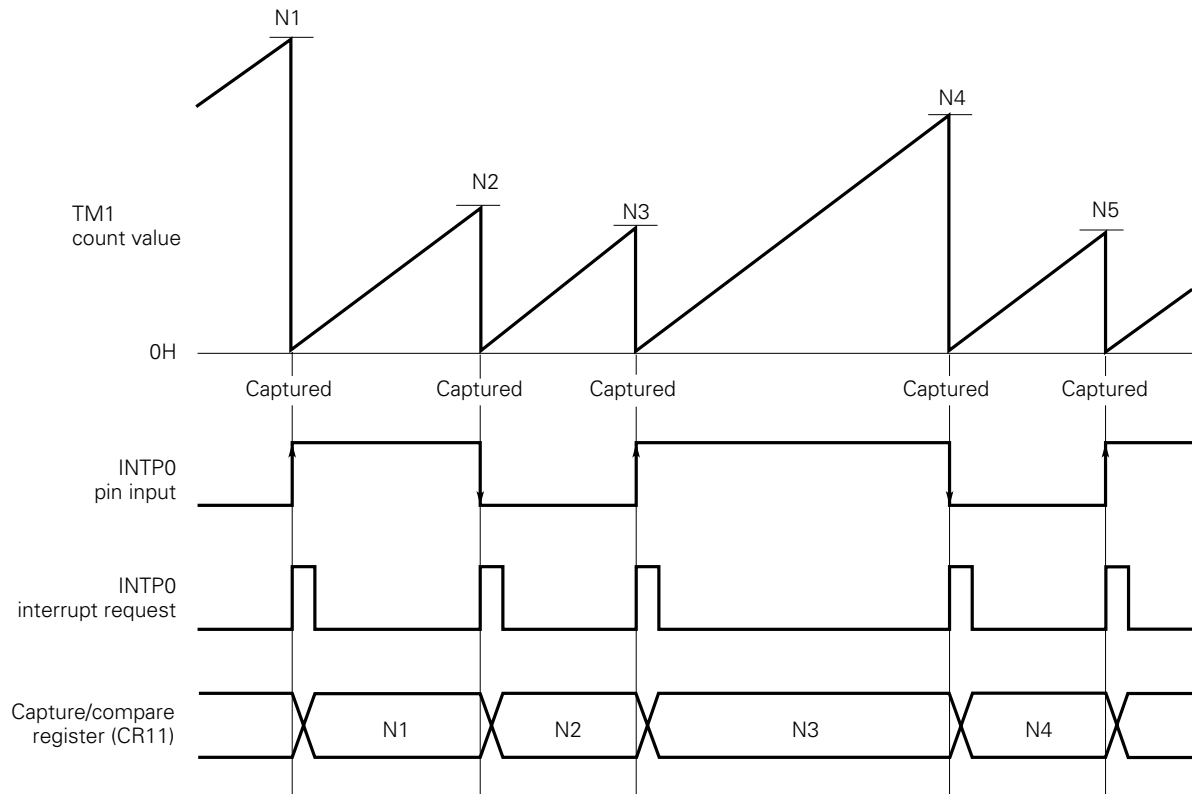
Caution When using an in-circuit emulator, see the notes described in Section 7.5.4.

Fig. 7-53 Capture Operation



Remark D_n : TM1 count value ($n = 0, 1, 2, \dots$)
 CLR10 = 0, CLR11 = 0, CM = 1

Fig. 7-54 TM1 Cleared after Capture Operations



Remark D_n : TM1 count value ($n = 0, 1, 2, \dots$)
 $CLR10 = 0, CLR11 = 0, CM = 1$

7.2.6 Sample Applications

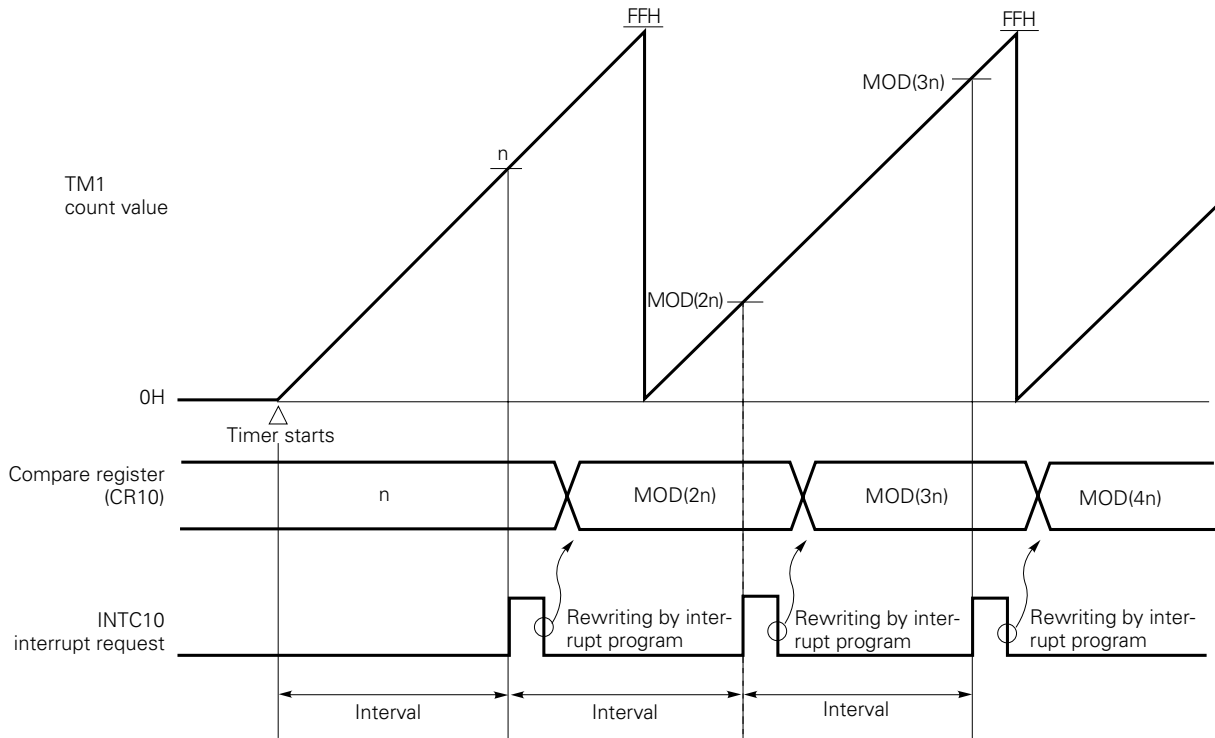
(1) Interval timer operation (1)

By free running 8-bit timer 1 (TM1), and adding a value to a compare register (CR10, CR11) in an interrupt handling routine, 8-bit timer/counter 1 can be used as an interval timer whose period is as long as the added value. (See Fig. 7-55.)

In addition, 8-bit timer 1 (TM1) has two compare registers, so that interval timers with two types of periods can be produced.

Fig. 7-56 shows the setting of control registers. Fig. 7-57 shows the setting procedure. Fig. 7-58 shows interrupt handling.

Fig. 7-55 Timing of Interval Timer Operation (1)



Remark Interval = $n \times x / f_{CLK}$, $1 \leq n \leq FFH$
 $x = 16, 32, 64, 128, 256, 512$

Fig. 7-56 Setting of Control Registers for Interval Timer Operation (1)

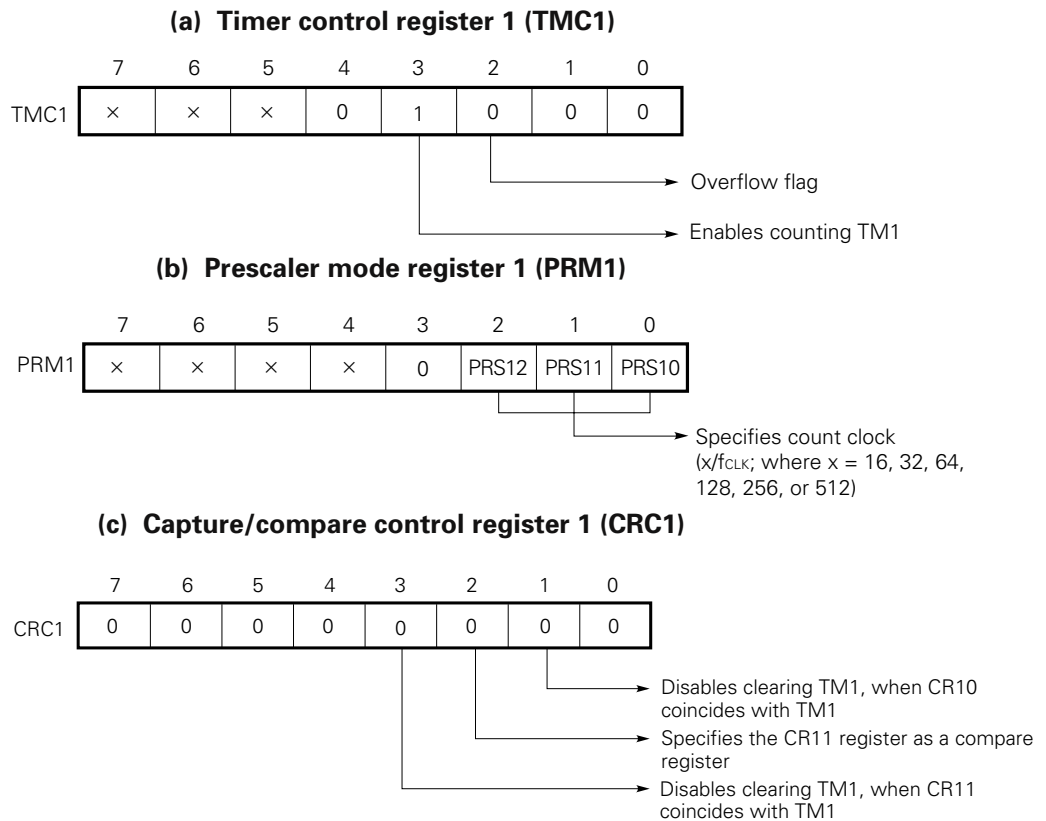
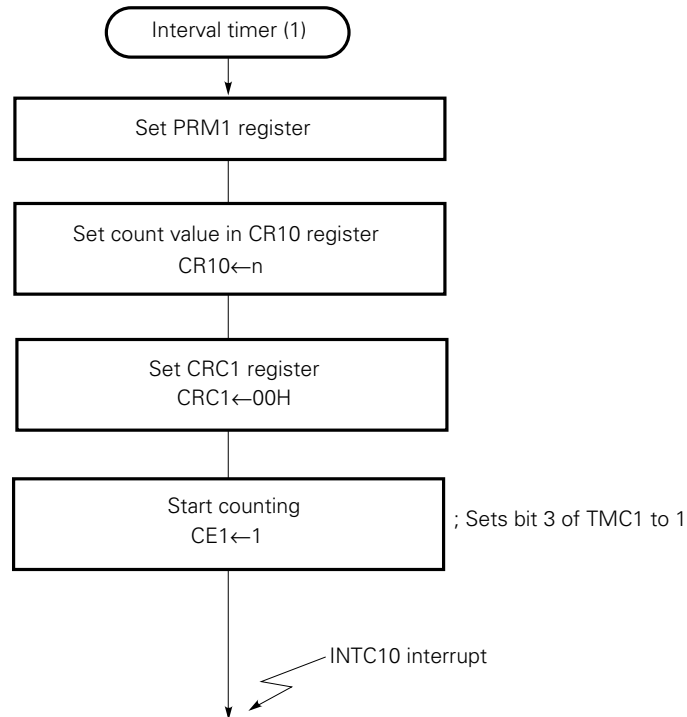
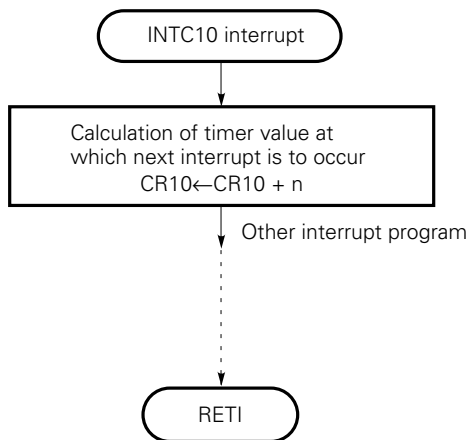
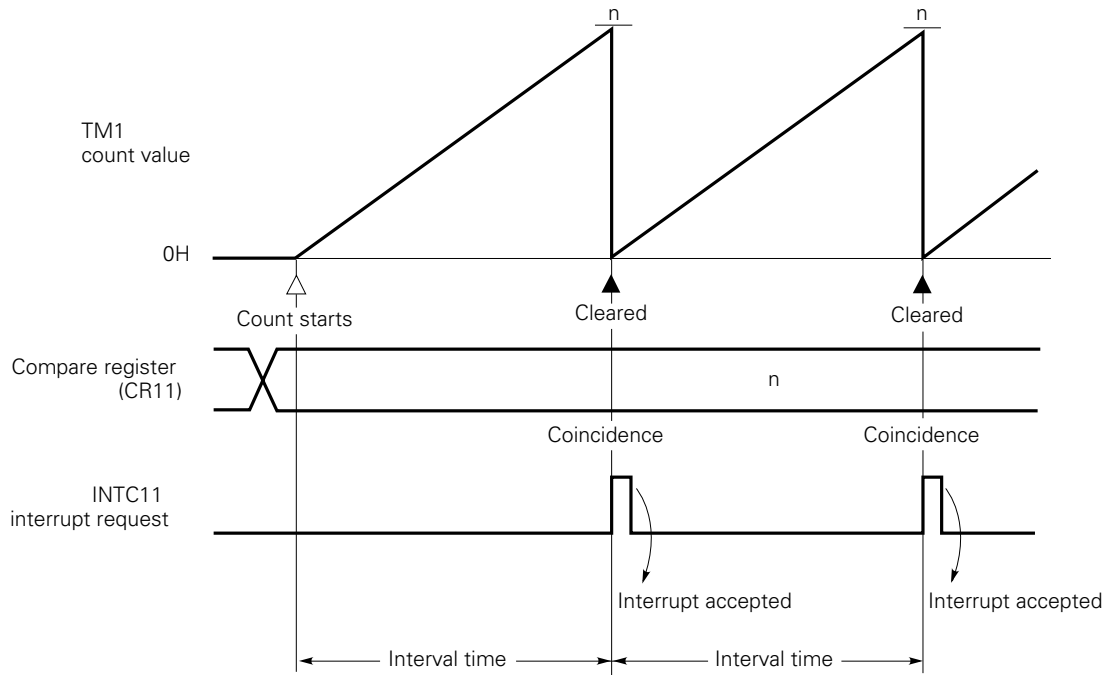


Fig. 7-57 Setting Procedure for Interval Timer Operation (1)**Fig. 7-58 Interrupt Request Handling for Interval Timer Operation (1)****(2) Interval timer operation (2)**

Eight-bit timer/counter 1 can be used as an interval timer that generates an interrupt at intervals of a count time specified beforehand. (See Fig. 7-59.)

Fig. 7-60 shows the setting of control registers, and Fig. 7-61 shows the setting procedure.

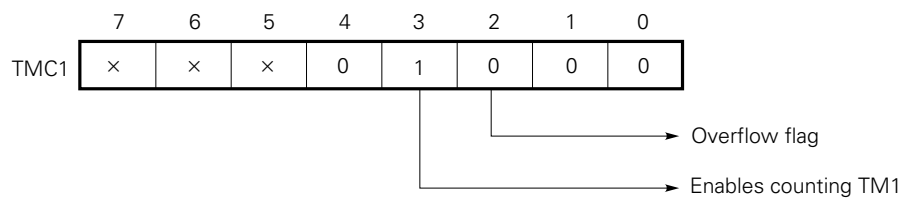
Fig. 7-59 Timing of Interval Timer Operation (2) (When CR11 Is Used As a Compare Register)



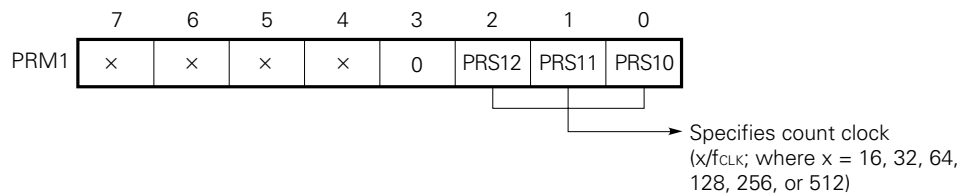
Remark Interval = $(n + 1) \times x/f_{CLK}$, $0 \leq n \leq FFH$
 $x = 16, 32, 64, 128, 256, 512$

Fig. 7-60 Setting of Control Registers for Interval Timer Operation (2)

(a) Timer control register 1 (TMC1)



(b) Prescaler mode register 1 (PRM1)



(c) Capture/compare control register 1 (CRC1)

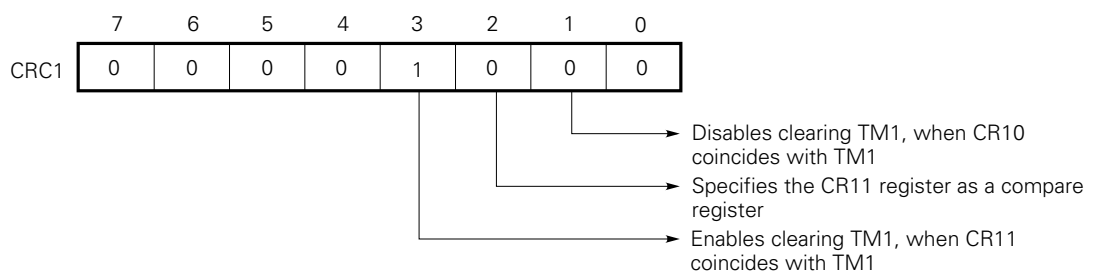
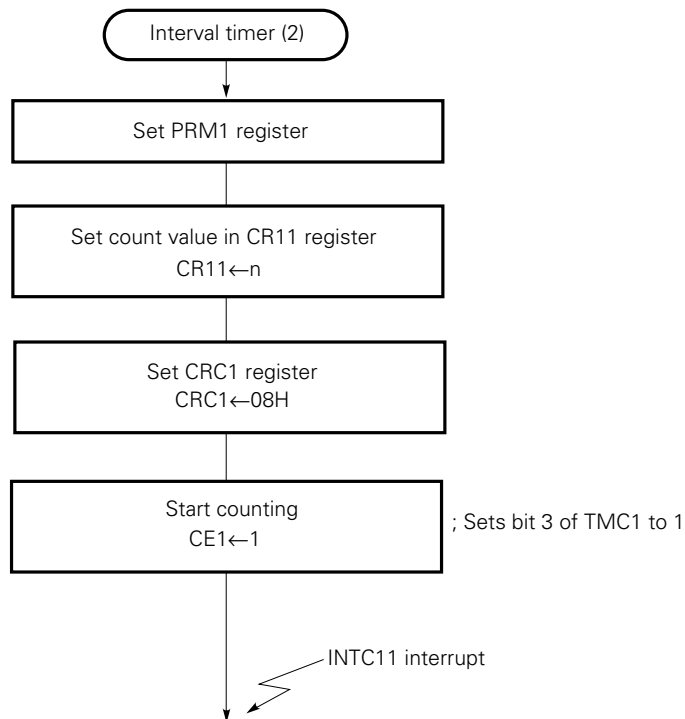


Fig. 7-61 Setting Procedure for Interval Timer Operation (2)



(3) Pulse width measurement operation

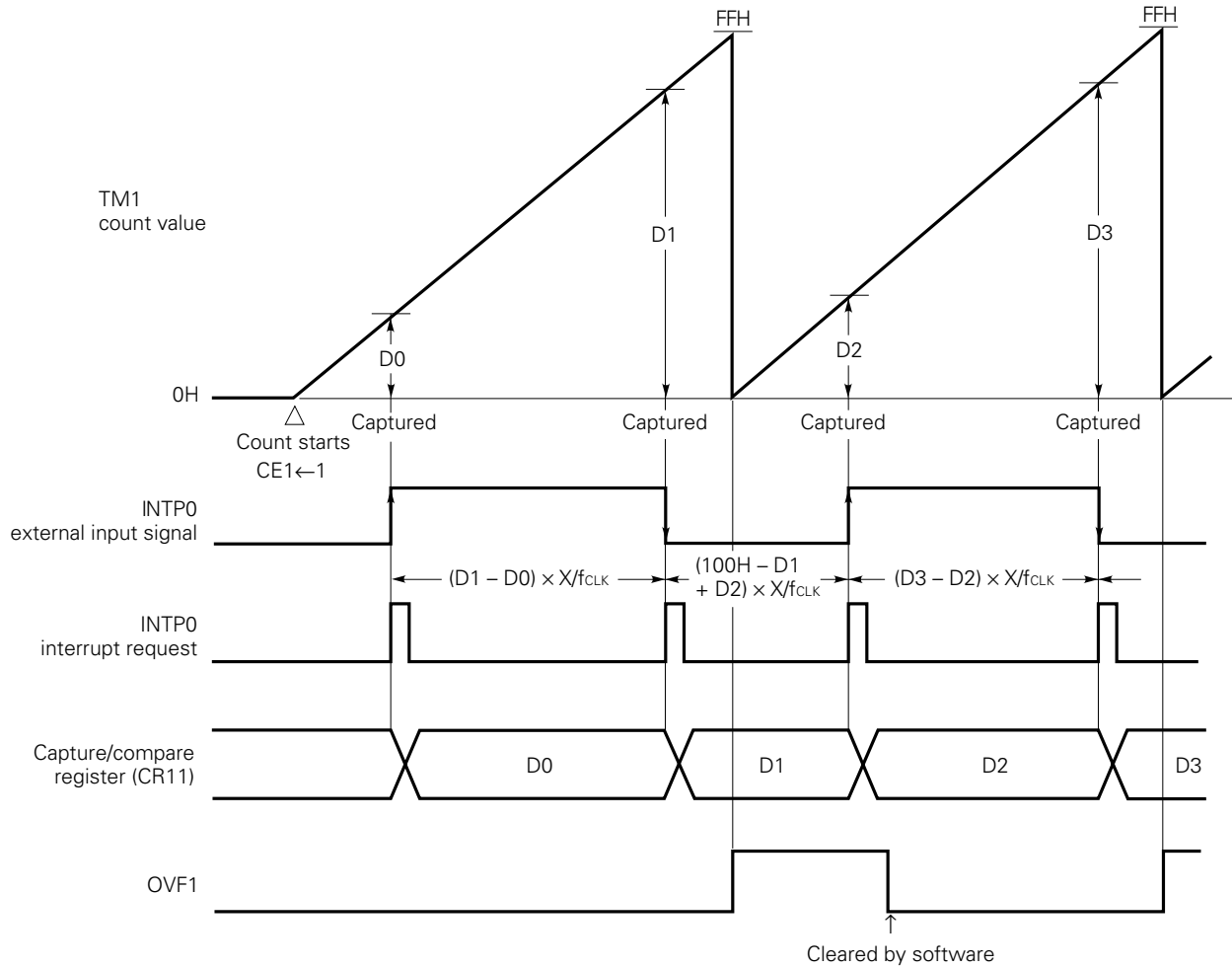
In pulse width measurement, the width of the high level or low level of an external pulse signal applied to the external interrupt request (INTP0) input pin is measured.

A pulse signal applied to the INTP0 pin must have a pulse width of 12 system clock pulses ($2 \mu\text{s}$; $f_{\text{CLK}} = 6 \text{ MHz}$) or more for both the high level and the low level. If the pulse width is less than this value, no valid edge can be detected, thus resulting in a failure to perform capture operation.

As shown in Fig. 7-62, the value of 8-bit timer 1 (TM1) in count operation is loaded and held in the CR11 capture/compare register specified as a capture register on a valid edge (either a rising edge or falling edge) of a signal applied to the INTP0 pin. The pulse width of the input signal is found by multiplying the count clock (X/f_{CLK} ; $X = 16, 32, 64, 128, 256, 512$) by the difference between the TM1 count value (D_n) loaded and held in the CR11 register on the n -th valid edge detected and the TM1 count value (D_{n-1}) loaded and held in the CR11 register on the $(n - 1)$ -th valid edge detected.

Fig. 7-63 shows the setting of control registers, and Fig. 7-64 shows the setting procedure.

Fig. 7-62 Timing of Pulse Width Measurement (When CR11 Is Used As a Capture Register)



Remark D_n : TM1 count value ($n = 0, 1, 2, \dots$)
 $X = 16, 32, 64, 128, 256, 512$

Fig. 7-63 Setting of Control Registers for Pulse Width Measurement

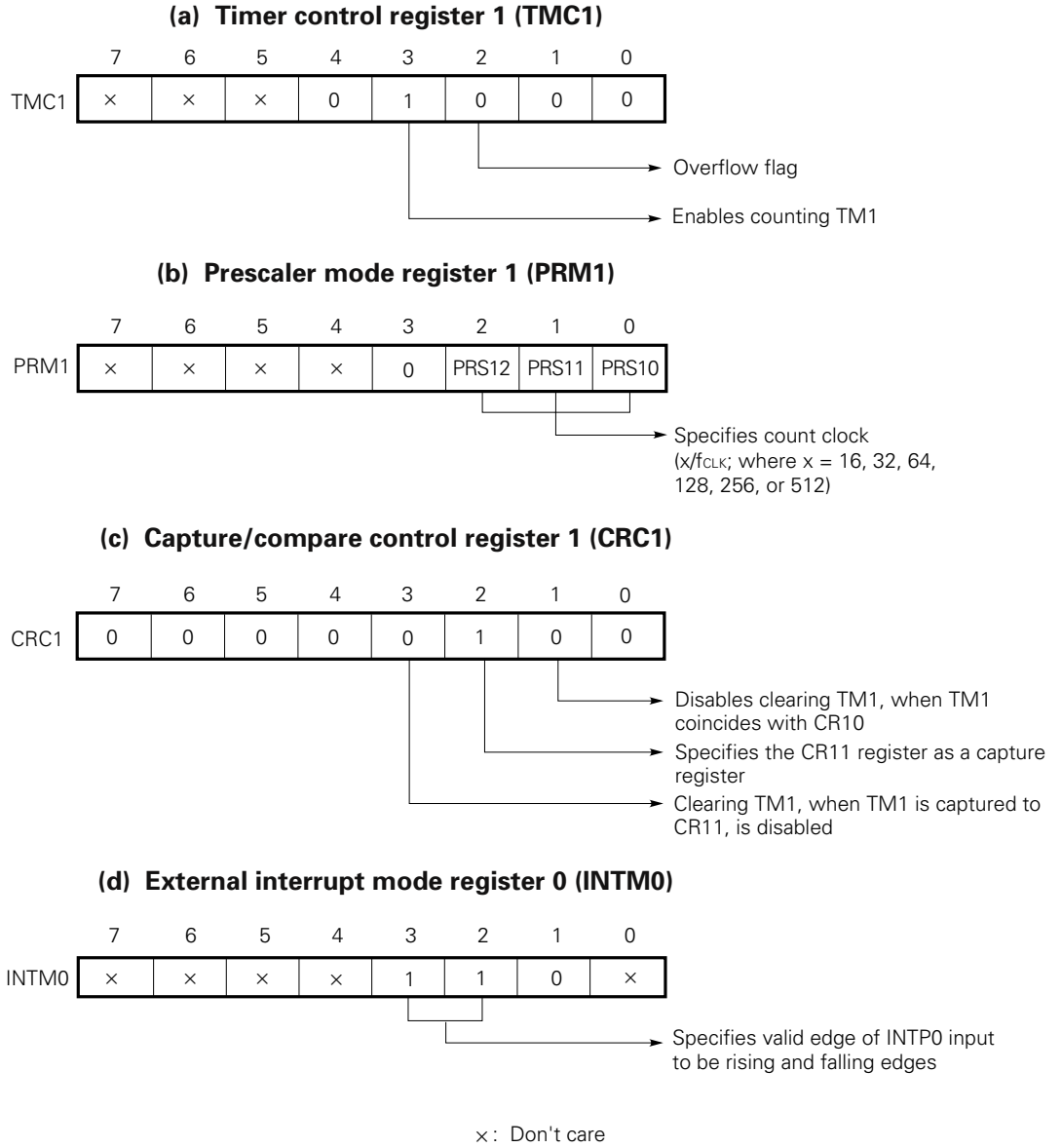


Fig. 7-64 Setting Procedure for Pulse Width Measurement

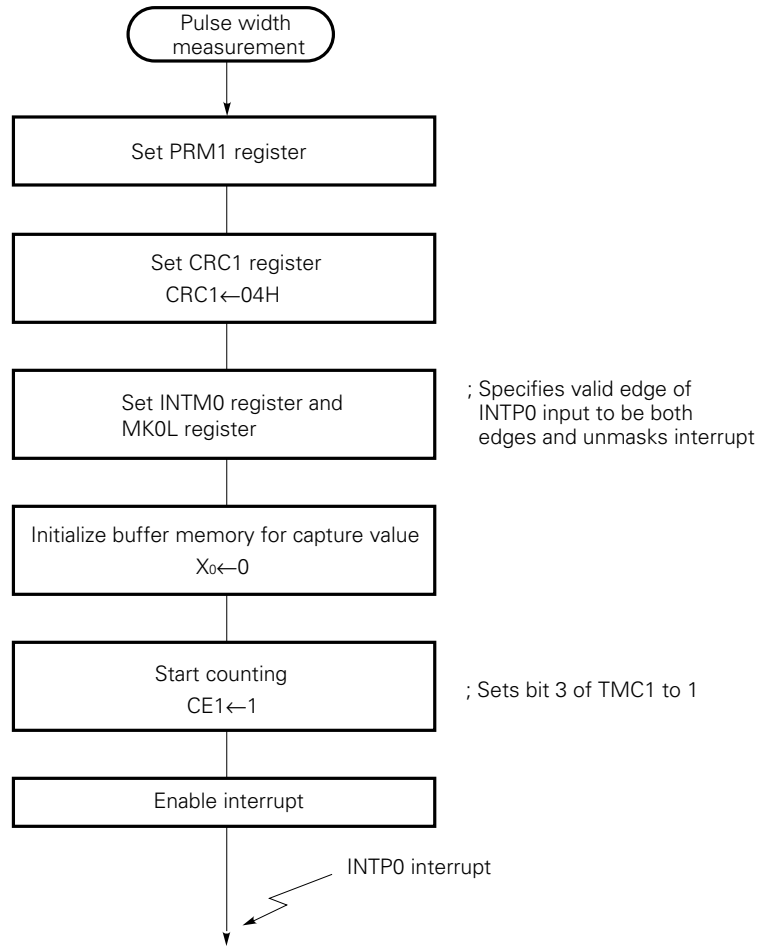
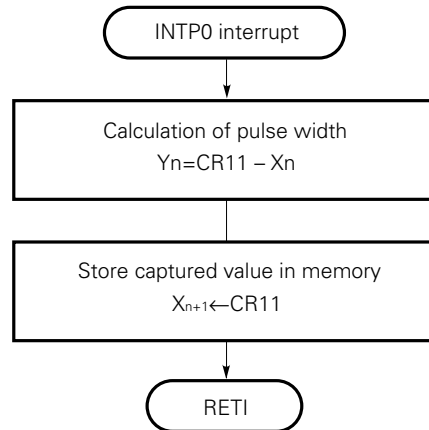


Fig. 7-65 Interrupt Request Handling for Pulse Width Calculation



7.3 8-BIT TIMER/COUNTER 2

7.3.1 Functions

Eight-bit timer/counter 2 has two functions not available with the other three timers/counters:

- External event counter
- One-shot timer

This section describes the following four basic functions in sequence:

- Interval timer
- Programmable square wave output
- Pulse width measurement
- External event counter

(1) Interval timer

When operating as an interval timer, 8-bit timer/counter 2 generates an internal interrupt at specified intervals.

Table 7-11 Intervals of 8-Bit Timer/Counter 2

Resolution	Minimum interval	Maximum interval
$16/f_{\text{CLK}}$ (2.6 μs)	$16/f_{\text{CLK}}$ (2.6 μs)	$2^8 \times 16/f_{\text{CLK}}$ (683 μs)
$32/f_{\text{CLK}}$ (5.3 μs)	$32/f_{\text{CLK}}$ (5.3 μs)	$2^8 \times 32/f_{\text{CLK}}$ (1.37 ms)
$64/f_{\text{CLK}}$ (10.7 μs)	$64/f_{\text{CLK}}$ (10.7 μs)	$2^8 \times 64/f_{\text{CLK}}$ (2.73 ms)
$128/f_{\text{CLK}}$ (21.3 μs)	$128/f_{\text{CLK}}$ (21.3 μs)	$2^8 \times 128/f_{\text{CLK}}$ (5.46 ms)
$256/f_{\text{CLK}}$ (42.7 μs)	$256/f_{\text{CLK}}$ (42.7 μs)	$2^8 \times 256/f_{\text{CLK}}$ (10.9 ms)
$512/f_{\text{CLK}}$ (85.3 μs)	$512/f_{\text{CLK}}$ (85.3 μs)	$2^8 \times 512/f_{\text{CLK}}$ (21.8 ms)

The values in parentheses are based on $f_{\text{CLK}} = 6 \text{ MHz}$.

(2) Programmable square wave output

Eight-bit timer/counter 2 outputs a square wave separately on the TO2 and TO3 timer output pins.

Table 7-12 Programmable Square Wave Output Setting Range of 8-Bit Timer/Counter 2

Minimum pulse width	Maximum pulse width
$16/f_{CLK}$ (2.6 μs)	$2^8 \times 16/f_{CLK}$ (683 μs)
$32/f_{CLK}$ (5.3 μs)	$2^8 \times 32/f_{CLK}$ (1.37 ms)
$64/f_{CLK}$ (10.7 μs)	$2^8 \times 64/f_{CLK}$ (2.73 ms)
$128/f_{CLK}$ (21.3 μs)	$2^8 \times 128/f_{CLK}$ (5.46 ms)
$256/f_{CLK}$ (42.7 μs)	$2^8 \times 256/f_{CLK}$ (10.9 ms)
$512/f_{CLK}$ (85.3 μs)	$2^8 \times 512/f_{CLK}$ (21.8 ms)

The values in parentheses are based on $f_{CLK} = 6$ MHz.

Caution The values in Table 7-12 assume the use of an internal clock.

(3) Pulse width measurement

Eight-bit timer/counter 2 measures the pulse width of a signal applied to the external interrupt input pin INTP1.

Table 7-13 Pulse Width Measurement Range of 8-Bit Timer/Counter 2

Measurable pulse width	Resolution
$\leq 2^8 \times 16/f_{CLK}$ (683 μs)	$16/f_{CLK}$ (2.6 μs)
$\leq 2^8 \times 32/f_{CLK}$ (1.37 ms)	$32/f_{CLK}$ (5.3 μs)
$\leq 2^8 \times 64/f_{CLK}$ (2.73 ms)	$64/f_{CLK}$ (10.7 μs)
$\leq 2^8 \times 128/f_{CLK}$ (5.46 ms)	$128/f_{CLK}$ (21.3 μs)
$\leq 2^8 \times 256/f_{CLK}$ (10.9 ms)	$256/f_{CLK}$ (42.7 μs)
$\leq 2^8 \times 512/f_{CLK}$ (21.8 ms)	$512/f_{CLK}$ (85.3 μs)

The values in parentheses are based on $f_{CLK} = 6$ MHz.

(4) External event counter

Eight-bit timer/counter 2 counts clock pulses (CI pin input pulses) applied to the external interrupt input pin (INTP2).

Table 7-14 indicates the clock signals that can be applied to 8-bit timer/counter 2.

Table 7-14 Clock Signals That Can Be Applied to 8-Bit Timer/Counter 2

	When occurrences of one edge only are counted	When occurrences of both edges are counted
Maximum frequency	$f_{CLK}/24$ (250 kHz)	$f_{CLK}/32$ (187.5 kHz)
Minimum pulse width ^{Note} (high and low level)	$12/f_{CLK}$ (2 μ s)	$16/f_{CLK}$ (2.67 μ s)

Note The values in parentheses are based on $f_{CLK} = 6$ MHz.

7.3.2 Configuration

Eight-bit timer/counter 2 consists of one 8-bit timer 2 (TM2), two 8-bit compare registers (CR20, CR21), and one 8-bit capture register (CR22).

Fig. 7-66 shows the block diagram of 8-bit timer/counter 2.

(1) 8-bit timer 2 (TM2)

TM2 is a timer for counting up with the count clock specified by external interrupt mode register 0 (INTM0) or the lower 4 bits of prescaler mode register 1 (PRM1). Either an internal count clock or an external count clock can be selected.

TM2 allows only read operation using an 8-bit manipulation instruction. The count operation of TM2 can be enabled or disabled by timer control register 1 (TMC1).

When the $\overline{\text{RESET}}$ signal is applied, TM2 is cleared to 00H, and count operation stops.

(2) Compare registers (CR20, CR21)

The CR20 and CR21 registers are 8-bit registers for holding a value that determines the period of interval timer operation.

When the values of the CR20 and CR21 registers coincide with the value of TM2, interrupt requests (INTC20, INTC21) and timer output control signals are generated. Count value clear operation can also be performed when the value of CR21 coincides with the value of TM2.

The CR20 and CR21 registers allow both read and write operations using an 8-bit manipulation instruction. When the $\overline{\text{RESET}}$ signal is applied, the CR20 and CR21 registers become undefined.

(3) Capture register (CR22)

The CR22 register is an 8-bit register for capturing the value of TM2.

Capture operation is performed on a valid edge (capture trigger) occurring on the external interrupt request (INTP1) input pin. The value of CR22 register is held until the next capture trigger occurs or the value of the CR22 register is read. After being read, the CR22 register is undefined until a value is set in the CR22 register when the next capture trigger occurs. After a capture operation, TM2 can be cleared.

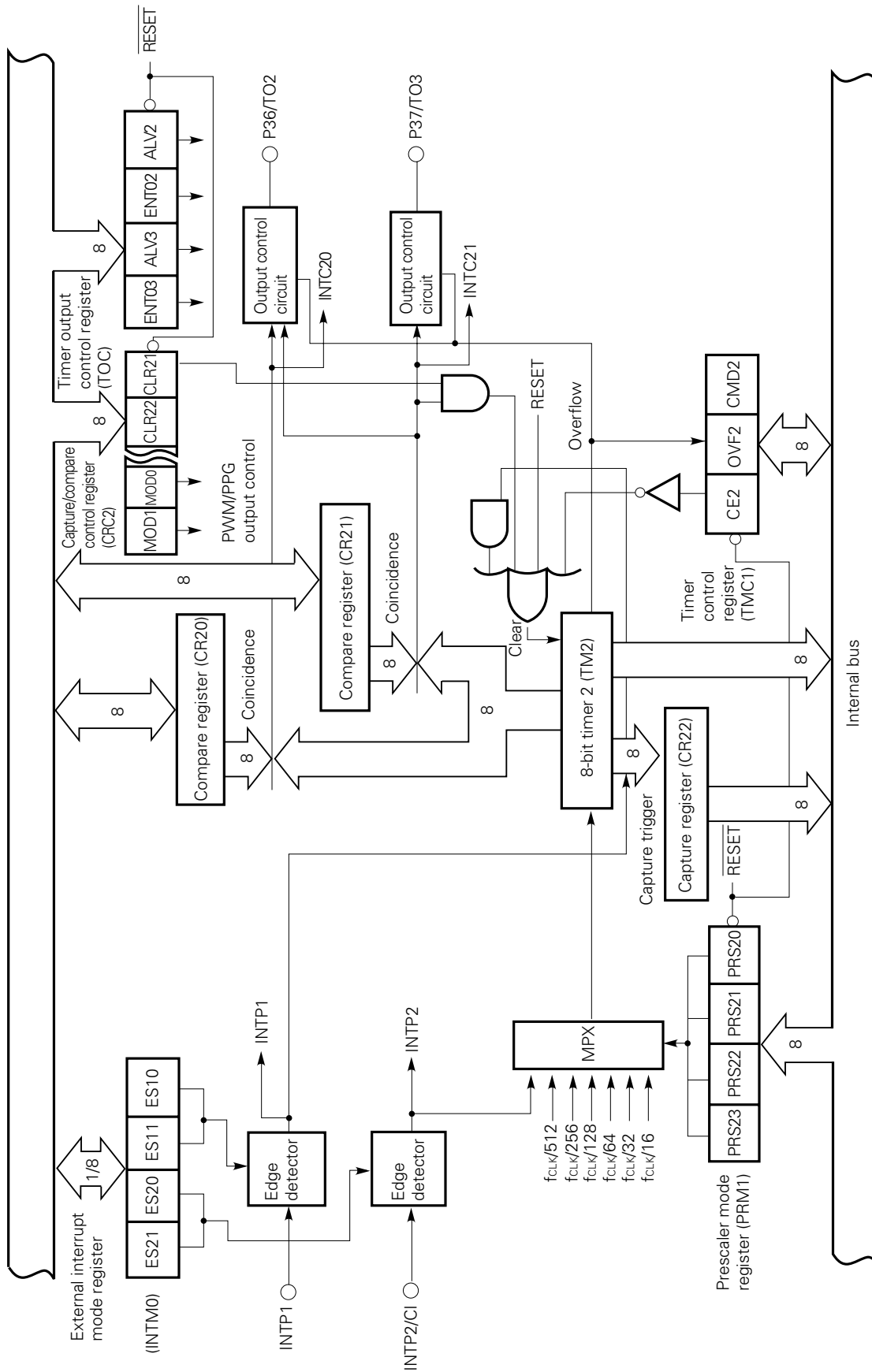
The CR22 register allows only read operation using an 8-bit manipulation instruction. When the $\overline{\text{RESET}}$ signal is applied, the CR22 register becomes undefined,

(4) Edge detector

The edge detector detects a valid edge of an external input signal.

When the edge detector detects, on the INTP1 input pin, a valid edge specified in external interrupt mode register 0 (INTM0), INTP1 and a capture trigger are generated. When the edge detector detects a valid edge on the external interrupt request (INTP2) input pin, a TM2 count clock and INTP2 are generated.

Fig. 7-66 Block Diagram of 8-Bit Timer/Counter 2



(5) Output control circuit

When the value of CR20 or CR21 coincides with the value of TM2, timer output can be inverted.

By setting the higher 4 bits of the timer output control register (TOC), a square wave can be output on a timer output pin (TO2, TO3). At this time, PWM/PPG output is possible, depending on the setting of capture/compare control register 2 (CRC2).

Timer output can be disabled or enabled by the TOC register. When timer output is disabled, a fixed level is output on the TO2 and TO3 pins. (An output level is set using the TOC register.)

(6) Prescaler

The prescaler generates count clocks from the internal system clock. From these count clocks generated by the prescaler, a count clock is selected with the selector for the timer to perform count operation.

7.3.3 8-Bit Timer/Counter 2 Control Registers**(1) Timer control register 1 (TMC1)**

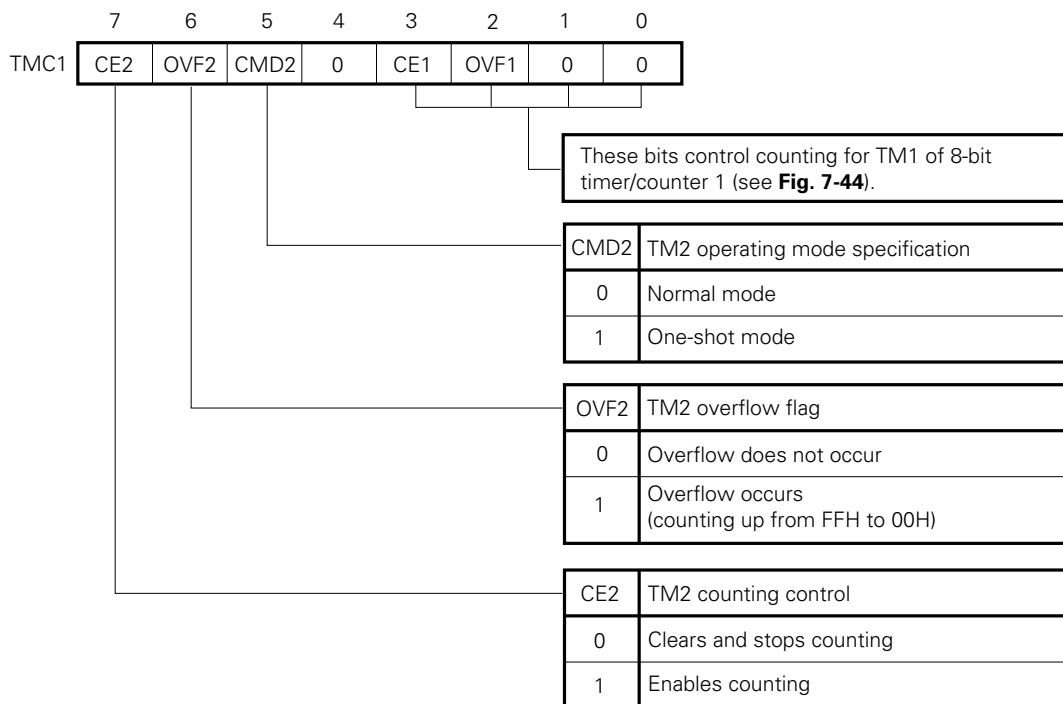
The TMC1 register is an 8-bit register for controlling the count operations of 8-bit timer 1 (TM1) and 8-bit timer 2 (TM2).

The higher 4 bits control the count operation of TM2 of 8-bit timer/counter 2. (The lower 4 bits control the count operation of TM1 of 8-bit timer/counter 1.)

The TMC1 register allows both read and write operations using an 8-bit manipulation instruction. Fig. 7-67 shows the format of the TMC1 register.

When the $\overline{\text{RESET}}$ signal is applied, the TMC1 register is cleared to 00H.

Fig. 7-67 Format of Timer Control Register 1 (TMC1)



Remark The OVF2 bit can be reset only by software.

(2) Prescaler mode register 1 (PRM1)

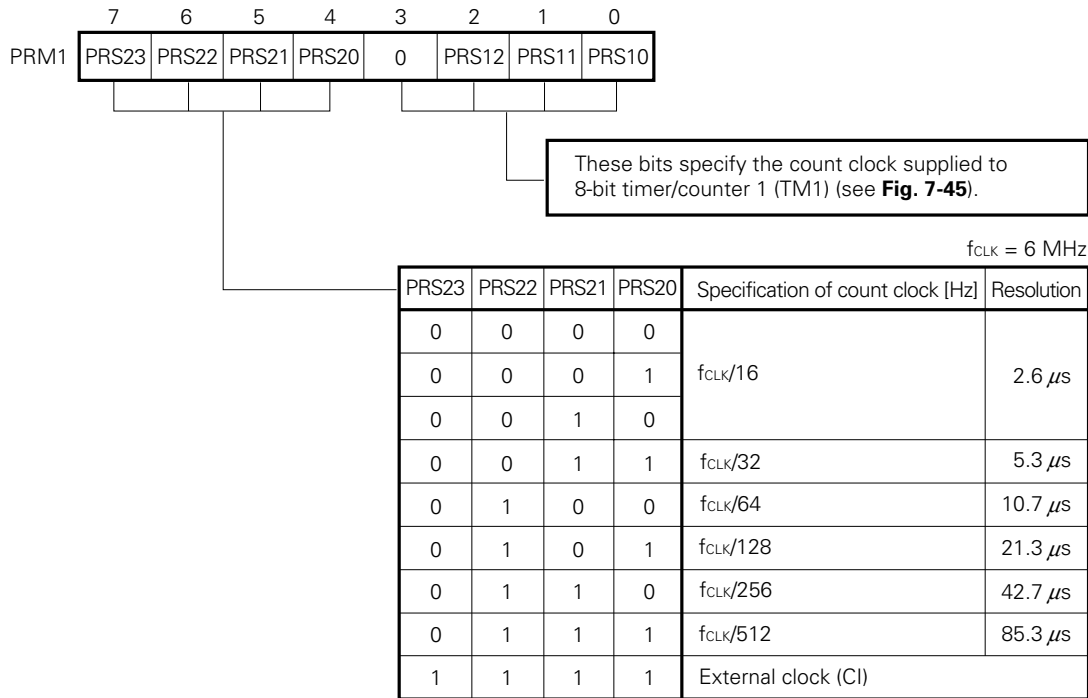
The PRM1 register is an 8-bit register used to specify a count clock for 8-bit timer 1 (TM1) and 8-bit timer 2 (TM2).

The higher 4 bits are used to specify a count clock for TM2 of 8-bit timer/counter 2. (The lower 4 bits are used to specify a count clock for TM1 of 8-bit timer/counter 1.)

The PRM1 register allows only write operation using an 8-bit manipulation instruction. Fig. 7-68 shows the format of the PRM1 register.

When the $\overline{\text{RESET}}$ signal is applied, the PRM1 register is cleared to 00H.

Fig. 7-68 Format of Prescaler Mode Register 1 (PRM1)



Remark f_{CLK}: System clock frequency

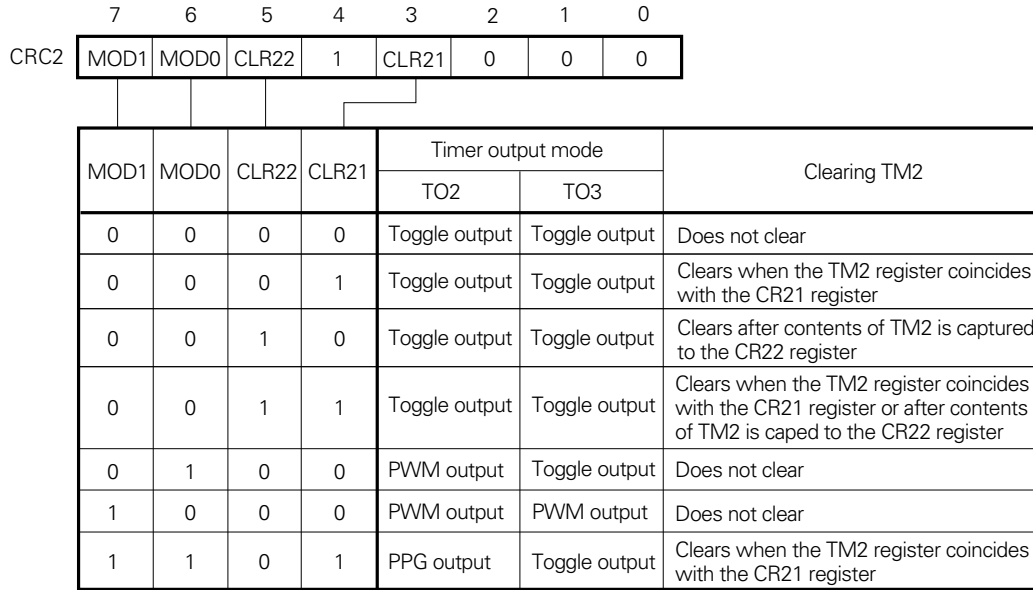
(3) Capture/compare control register 2 (CRC2)

The CRC2 register is used to specify the condition for enabling the clear operation of 8-bit timer 2 (TM2) with the CR21 compare register or CR22 capture register, and also specify a timer output (TO2, TO3) mode.

The CRC2 register allows only write operation using an 8-bit manipulation instruction. Fig. 7-69 shows the format of the CRC2 register.

When the $\overline{\text{RESET}}$ signal is applied, the CRC2 register is cleared to 10H.

Fig. 7-69 Format of Capture/Compare Control Register 2 (CRC2)



Caution Combinations other than those indicated above are prohibited.

(4) Timer output control register (TOC)

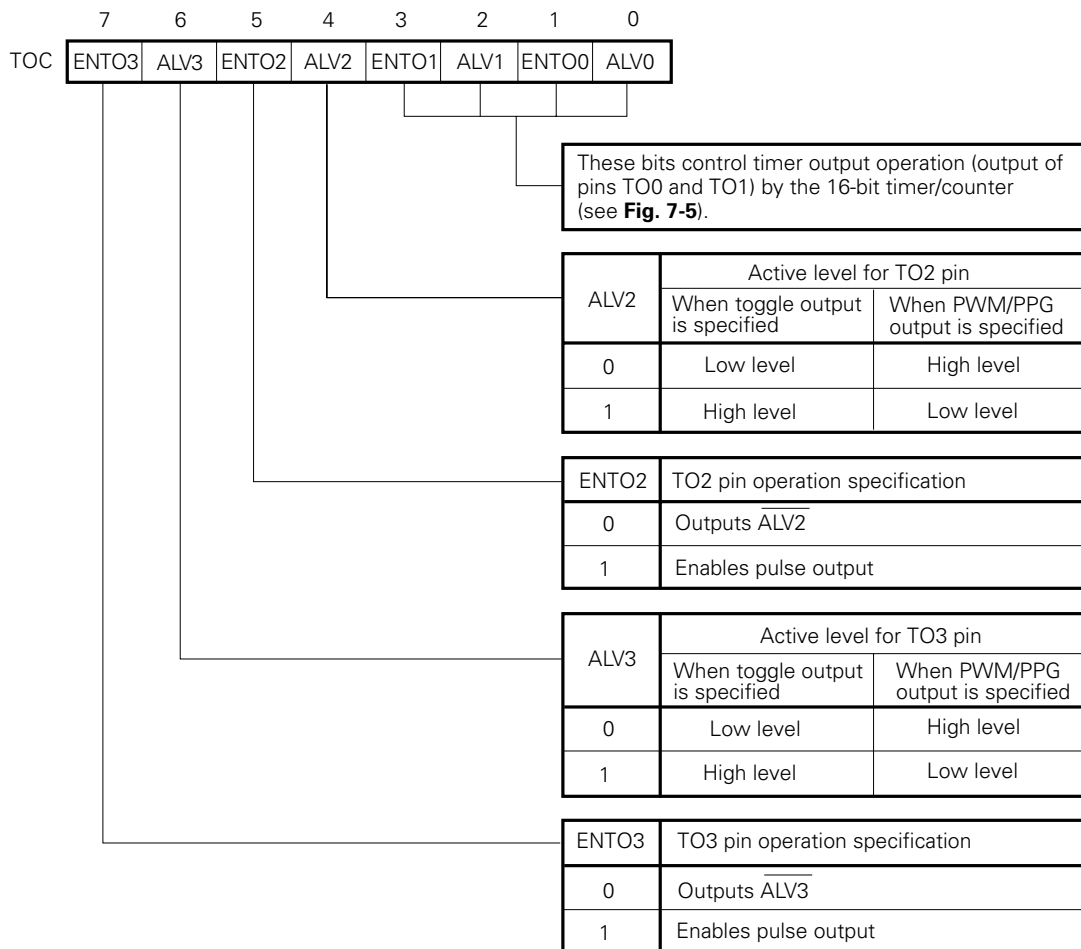
The TOC register is an 8-bit register for controlling the active level of timer output and for enabling/disabling timer output.

The higher 4 bits control the timer output operation (on the TO2 and TO3 pins) of 8-bit timer/counter 2. (The lower 4 bits control the timer output operation (on the TO0 and TO1 pins) of the 16-bit timer/counter.)

The TOC register allows only write operation using an 8-bit measurement instruction. Fig. 7-70 shows the format of the TOC register.

When the $\overline{\text{RESET}}$ signal is applied, the TOC register is cleared to 00H.

Fig. 7-70 Format of Timer Output Control Register (TOC)



7.3.4 Operation of 8-Bit Timer 2 (TM2)

(1) Basic operation

Eight-bit timer/counter 2 performs count operation by counting up with the count clock specified by the higher 4 bits of prescaler mode register 1 (PRM1).

Bit 7 (CE2) of timer control register 1 (TMC1) is used to enable/disable count operation. (The higher 4 bits of the TMC1 register are used to control the operation of 8-bit timer/counter 2.) When the CE2 bit is set to 1 by software, TM2 is cleared to 00H by the first count clock pulse, then count-up operation starts.

When the CE2 bit is reset to 0 by software, TM2 is cleared to 00H by the next count clock pulse, then capture operation and coincide with signal generation stop.

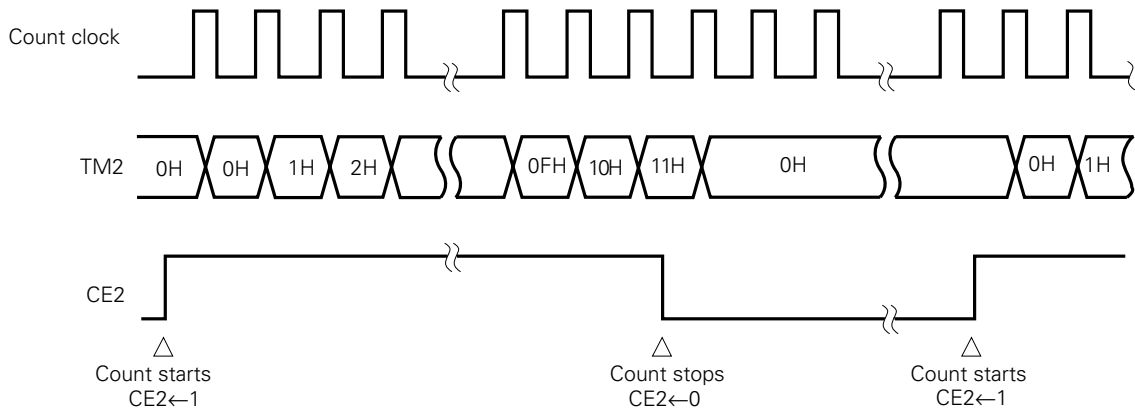
If the CE2 bit is set to 1 when the CE2 bit is already set to 1, TM2 count operation is not affected. (See Fig. 7-71 (b).)

If the count clock signal is applied when the value of TM2 is FFH, TM2 is set to 00H. At this time, OVF2 is set, and the overflow signal is sent to the output control circuit. OVF2 can be cleared only by software. Count operation continues.

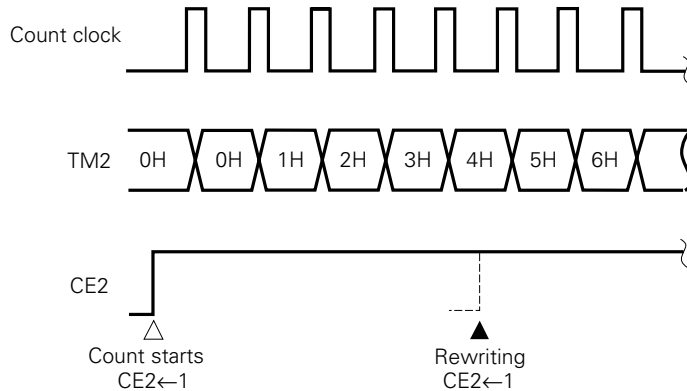
When the RESET signal is applied, TM2 is cleared to 00H, and count operation stops.

Fig. 7-71 Basic Operation of 8-Bit Timer 2 (TM2)

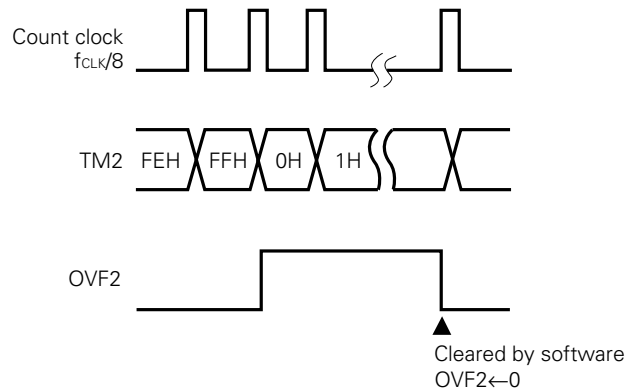
(a) Count start → count stop → count start



(b) When the CE2 bit is set to 1 again after count operation starts



(c) When the value of TM2 is FFH



(2) Clear operation

After a coincidence with the CR21 compare register or capture operation, 8-bit timer 2 (TM2) can be automatically cleared. If a TM2 clear cause occurs, TM2 is cleared to 00H by the next count clock pulse. This means that even if a TM2 clear cause occurs, TM2 holds the value existing at that time until the next count clock pulse is applied.

Fig. 7-72 TM2 Cleared by a Coincidence with Compare Register (CR21)

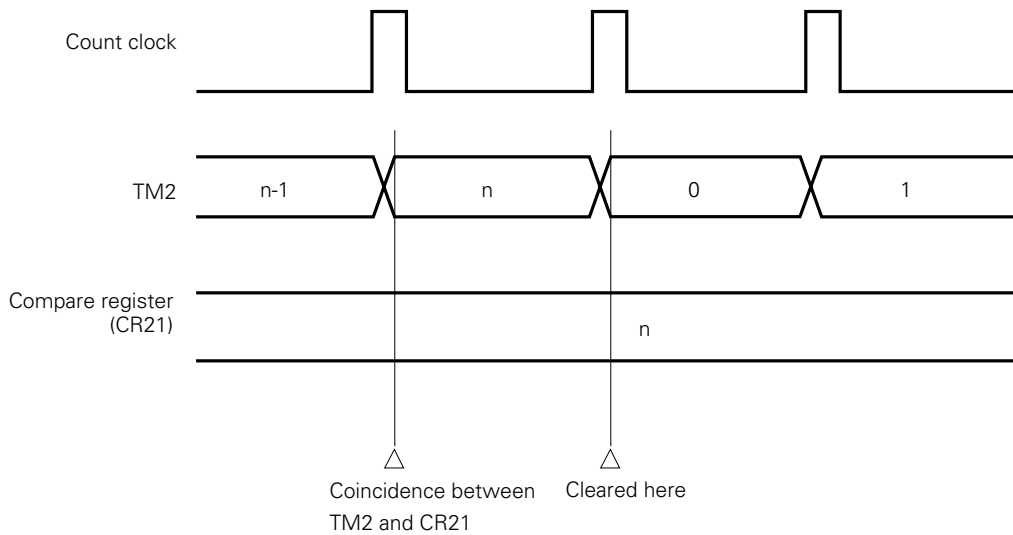
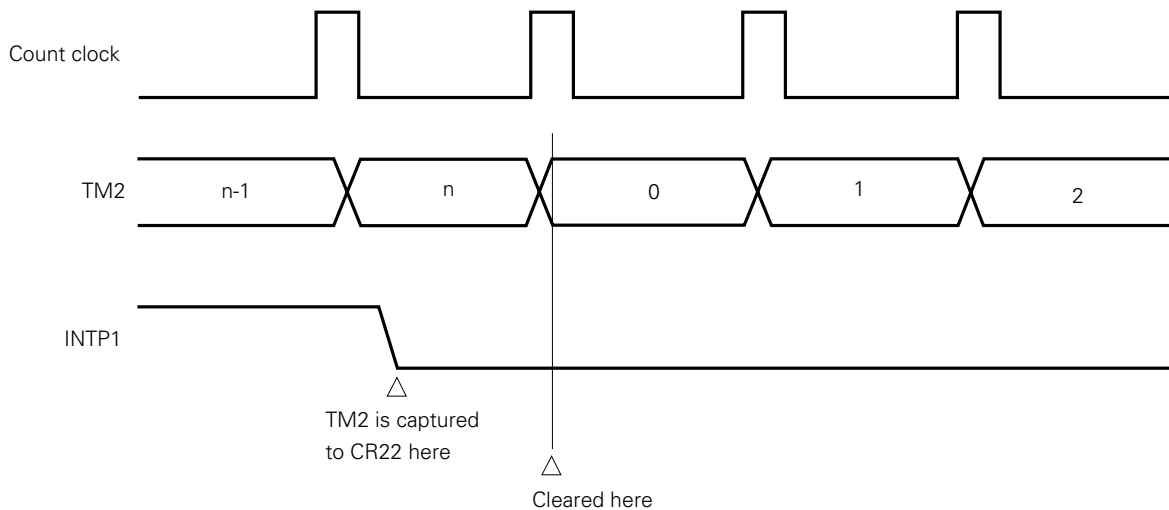


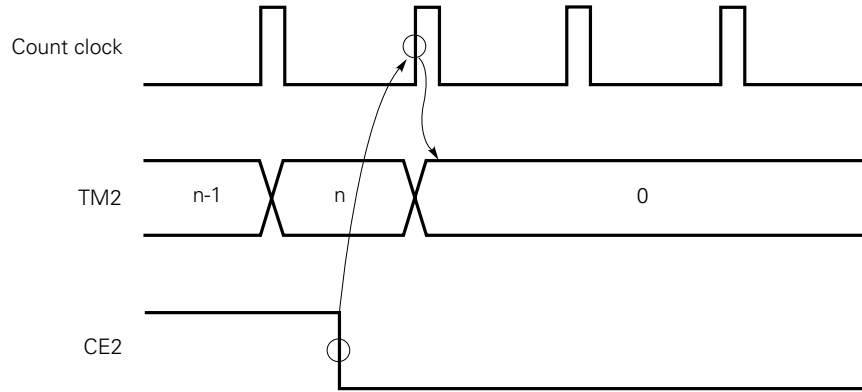
Fig. 7-73 TM2 Cleared after Capture Operation



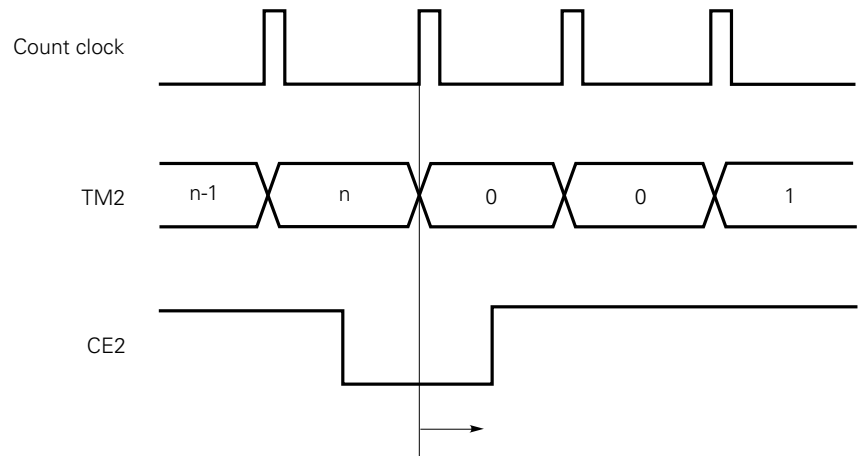
TM2 can also be cleared by software when the CE2 bit of the timer control register (TMC1) is reset to 0. Similarly, clear operation is performed by the count clock pulse following the resetting of CE2 bit to 0. If the CE2 bit is set to 1 before TM2 is reset to 0 by the resetting of the CE2 bit to 0 (that is, before the first count clock pulse is applied after the CE2 bit is reset to 0), two operations are simultaneously performed: one operation is an operation to clear TM2 to 0, and the other operation is a count operation starting with the counting of 0.

Fig. 7-74 Clear Operation When the CE2 Bit Is Reset to 0

(a) Basic operation

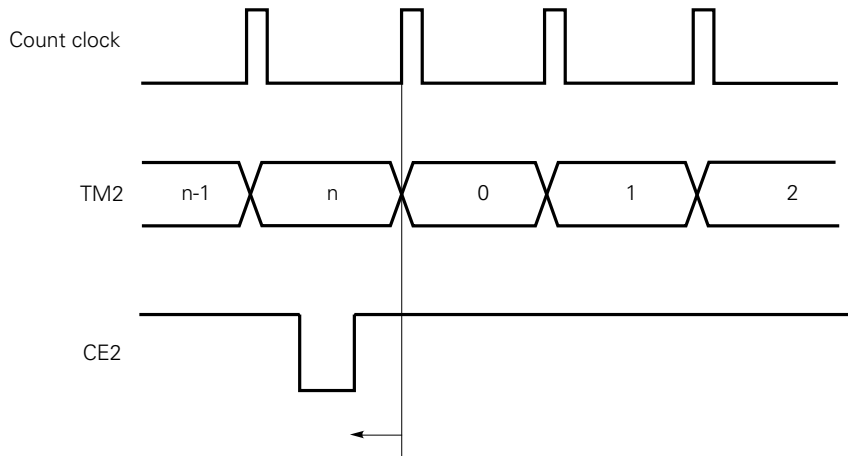


(b) Restart after 0 is set in TM2 cleared



When the CE2 bit is set to 1 after this count clock, counting starts from 0 on the count clock input after the CE2 bit has been set.

(c) Restart before 0 is set in TM2 cleared



When the CE2 bit is set to 1 before this count clock, Clearing TM2 by CE2←0 and counting by CE2←1 are performed simultaneously.

7.3.5 External Event Counter Function

Eight-bit timer/counter 2 can count clock pulses externally applied to the CI pin.

The external event counter operation mode requires no particular selection method. TM2 functions as an external event counter when external count input is specified for the count clock of TM2 by setting the higher 4 bits of prescaler mode register 1 (PRM1).

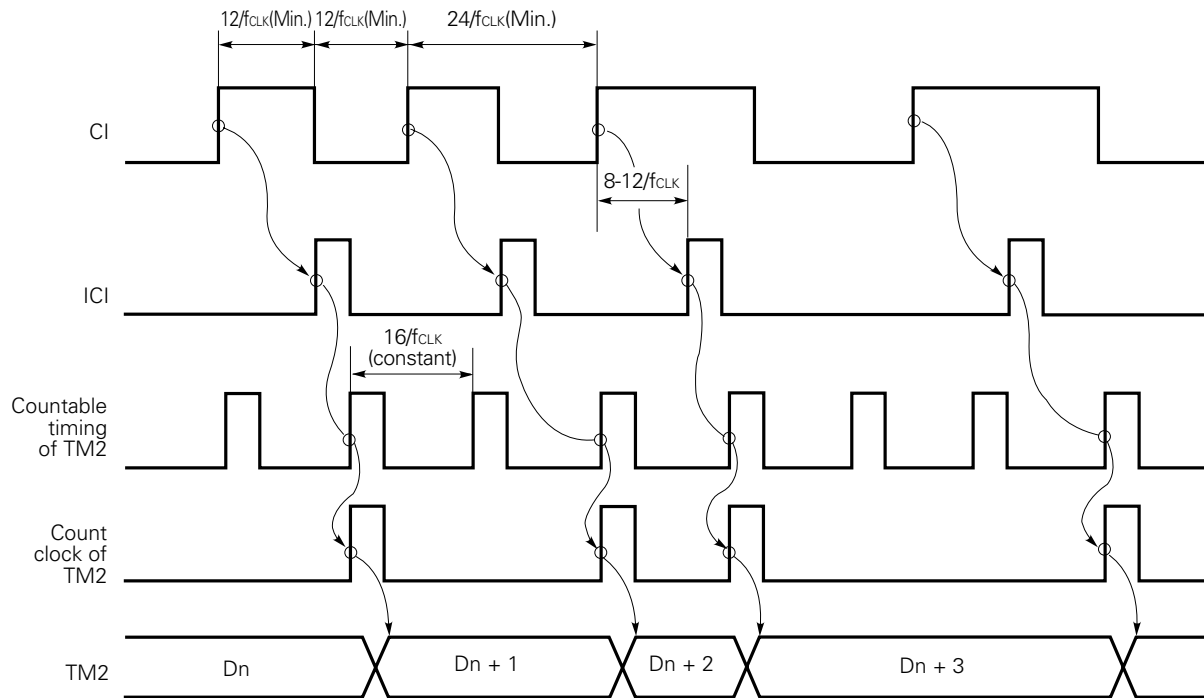
The maximum frequency of an external clock pulse that can be counted by TM2 functioning as an external event counter is 187.5 kHz when occurrences of both CI input signal edges are counted, and is 250 kHz when occurrences of only one edge are counted ($f_{CLK} = 6 \text{ MHz}$).

When occurrences of both edges are counted, the external clock pulse signal must have a pulse width of 16 system clock pulses ($2.67 \mu\text{s}$; $f_{CLK} = 6 \text{ MHz}$) or more for both the high level and the low level. If the pulse width is less than this value, no edges may be counted. When occurrences of only one edge are counted, the external clock pulse signal must have a pulse width of 12 system clock pulses ($2 \mu\text{s}$; $f_{CLK} = 6 \text{ MHz}$) or more for both the high level and the low level. If the pulse width is less than this value, no edges may be counted.

Fig. 7-75 shows the external event count timing of 8-bit timer/counter 2.

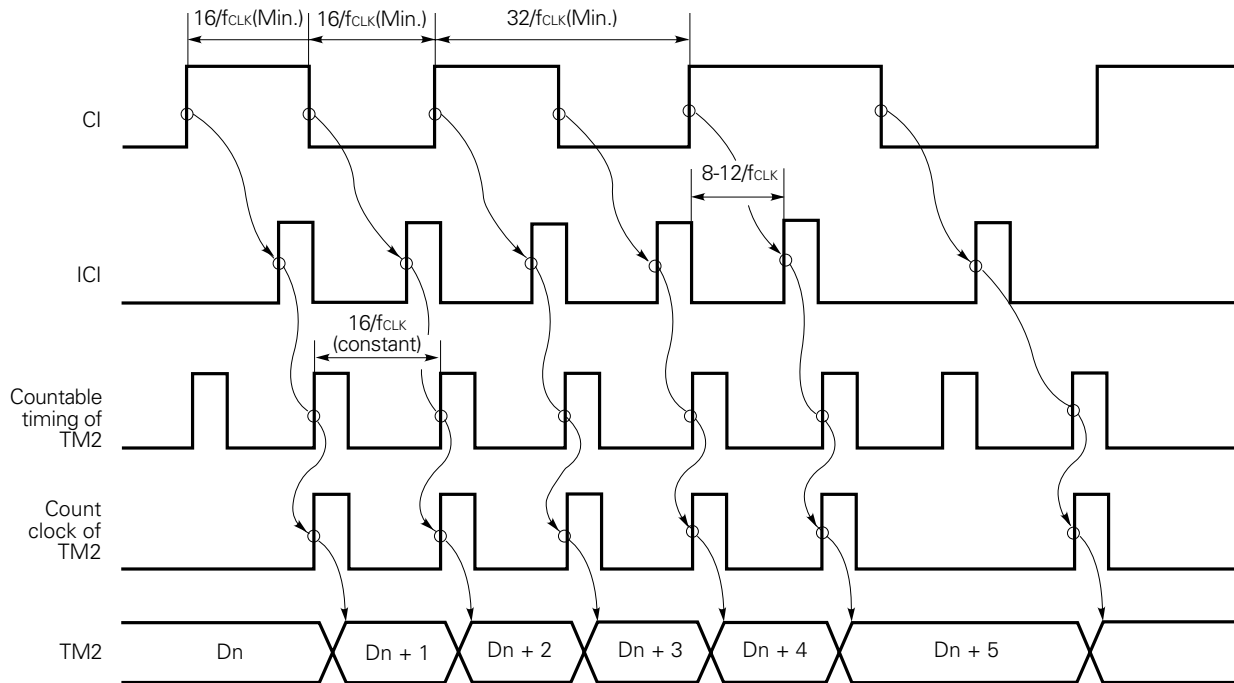
Fig. 7-75 External Event Count Timing of 8-Bit Timer/Counter 2

(1) When occurrences of one edge are counted (maximum frequency = $f_{CLK}/24$)



Remark ICI: CI input signal after passing through the edge detector

(2) When occurrences of both edges are counted (maximum frequency = $f_{CLK}/32$)



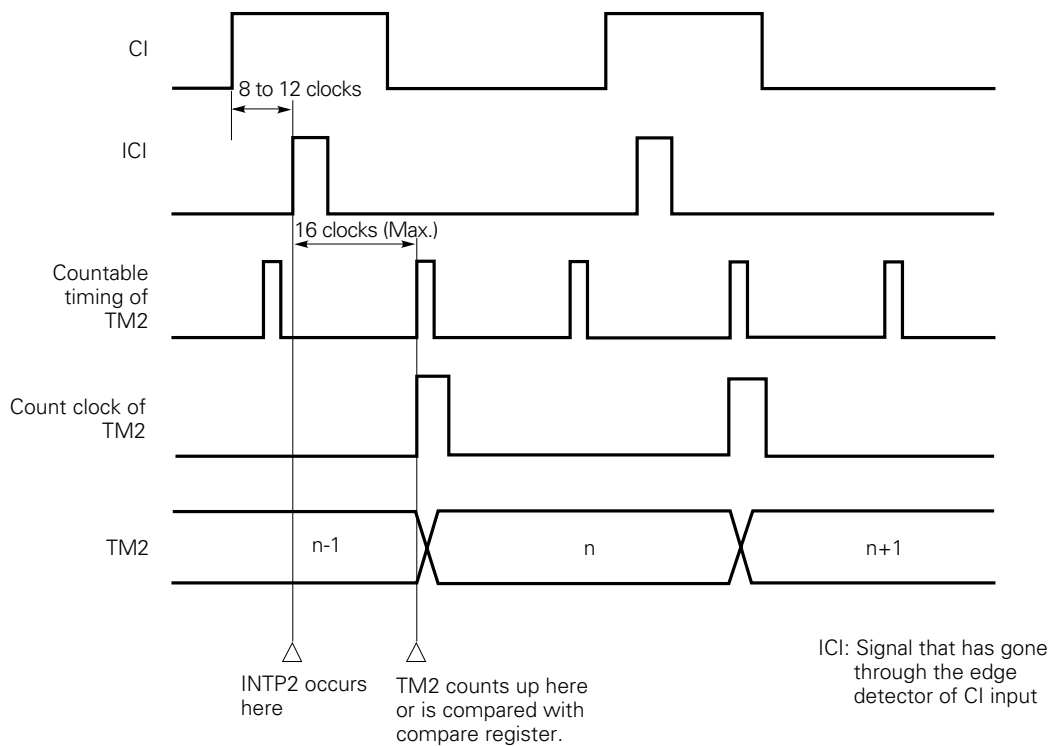
Remark ICI: CI input signal after passing through the edge detector

The count operation of TM2 is controlled by the CE2 bit of the TMC1 register as in the case of basic operation. When the CE2 bit is set to 1 by software, TM2 is cleared to 00H by the first count clock pulse, then count-up operation starts.

When the CE2 bit is set to 0 by software during TM2 count operation, TM2 is cleared to 00H by the next count clock pulse, and count operation stops. If the CE2 bit is set to 1 by software when the CE2 bit is already set to 1, TM2 count operation is not affected.

Cautions 1. When 8-bit timer/counter 2 is used as an external event counter, the increment of TM2 lags the input of a valid edge to the CI pin by a maximum of 28 system clock pulses (4.67 μs: $f_{CLK} = 6MHz$). This means that TM2 may not be incremented yet when read immediately after an edge is detected. In addition, the generation of an interrupt request by a coincidence with a compare register (CR20, CR21) lags the input of an edge. Take this point into consideration when short-period timing control is required after input of an edge.

Fig. 7-76 Interrupt Request Generation Using External Event Counter



2. When 8-bit timer/counter 2 is used as an external event counter, TM2 alone cannot distinguish between the state where no valid edge is applied and the state where only one valid edge has been applied. (See Fig. 7-77.) In either case, the value of TM2 is 0. When the states need to be distinguished from each other, use the INTP2 interrupt request flag. (The same pin is used as the INTP2 pin as well as the CI pin, so that the functions can be used at the same time.) Fig. 7-78 shows an example of distinction.

Fig. 7-77 Example Where Input of No Valid Edge Cannot Be Distinguished from Input of Only One Valid Edge with External Event Counter

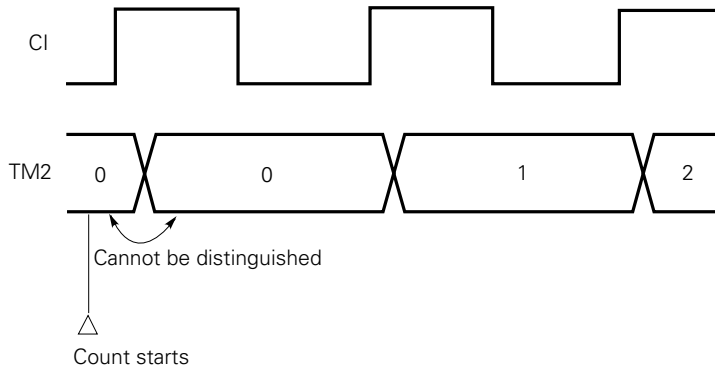
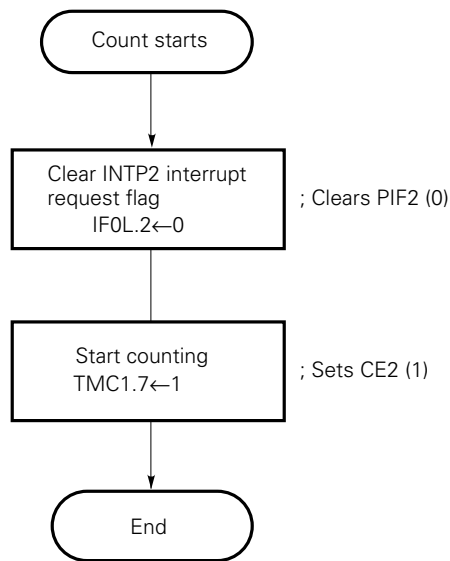
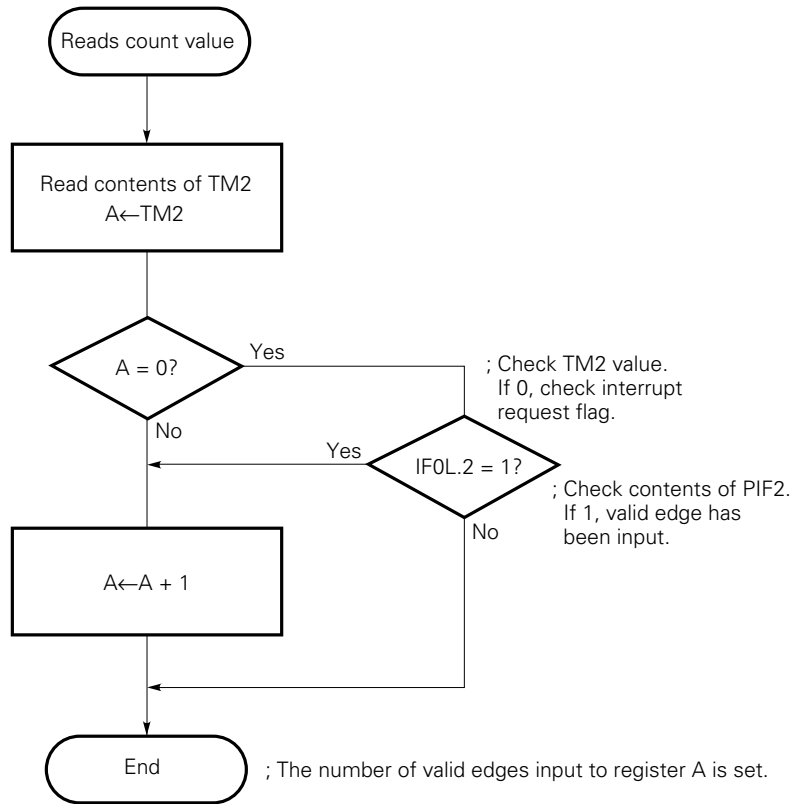


Fig. 7-78 How to Distinguish Input of No Valid Edge from Input of Only One Valid Edge with External Event Counter

(a) Count start processing



(b) Count value read processing

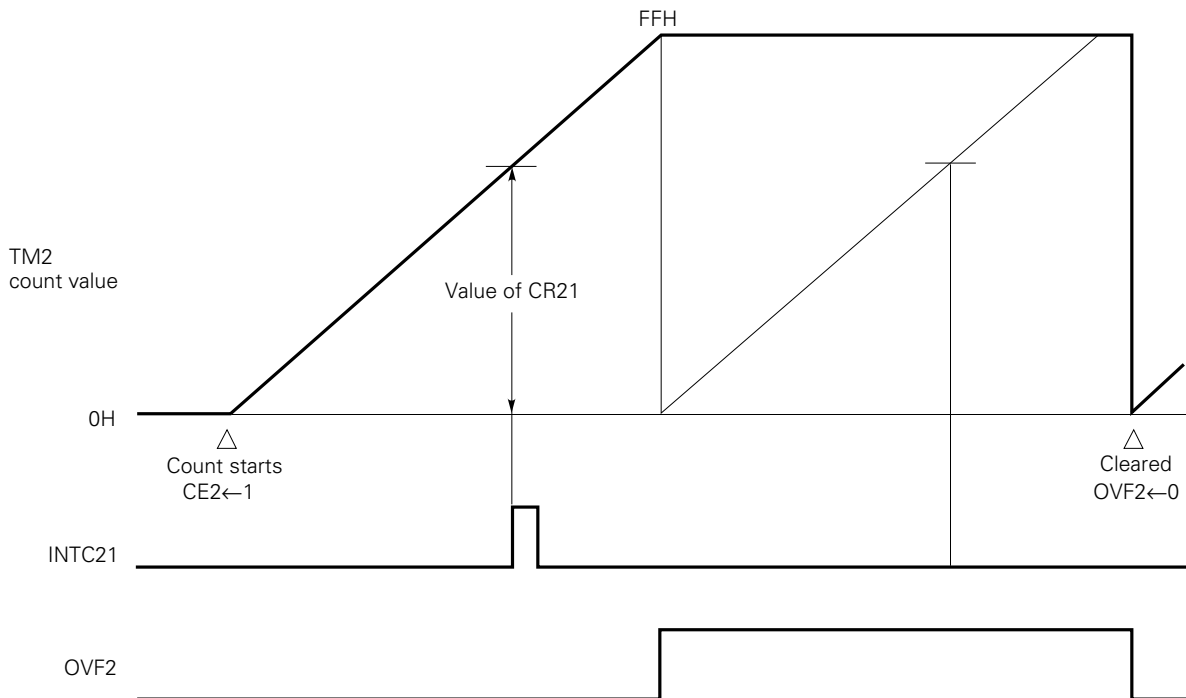


3. With an in-circuit emulator, digital noise on the CI/INTP2 pin cannot be removed correctly. When the event counter function is used, the operation described below is performed if an edge is detected erroneously.
- When IE-78210-R is used
Count operation is performed on an erroneously detected edge. At this time, an interrupt request is also generated on the edge.
 - When other in-circuit emulators are used
Count operation is not performed on an erroneously detected edge. However, an interrupt request is generated on the edge.
- For detailed information about erroneous edge detection, see Section 11.4.

7.3.6 One-Shot Timer Function

Eight-bit timer/counter 2 has an operation mode in which the full-count (FFH) is reached as the result of count operation.

Fig. 7-79 One-Shot Timer Operation



As shown in Fig. 7-79, a one-shot interrupt is generated when the value (00H-FFH) set in the CR20 or CR21 register coincides with the value of TM2.

The one-shot timer operation mode can be specified by setting bit 5 (CMD2) of timer control register 1 (TMC1) to 1 by software.

The count operation of TM2 is controlled using the CE2 bit of the TMC1 register as in the case of basic operation. When the CE2 bit is set to 1 by software, TM2 is cleared to 00H by the first count clock pulse, then count-up operation starts.

When the value of TM2 reaches FFH (full-count) as the result of count operation, bit 6 (OVF2) of the TMC1 register is set to 1, and TM2 stops its count operation with the count value of FFH held.

From the count stop state, one-shot timer operation can be started again by resetting the OVF2 bit to 0 by software. When the OVF2 bit is reset to 0, TM2 is cleared to 00H by the next count clock pulse, then count-up operation restarts.

When the CE2 bit is reset to 0 by software during TM2 count operation, TM2 is cleared to 00H by the next count clock pulse, and count operation stops. If the CE2 bit is set to 1 by software when the CE2 bit is already set to 1, TM2 count operation is not affected.

7.3.7 Compare Register and Capture Register Operations

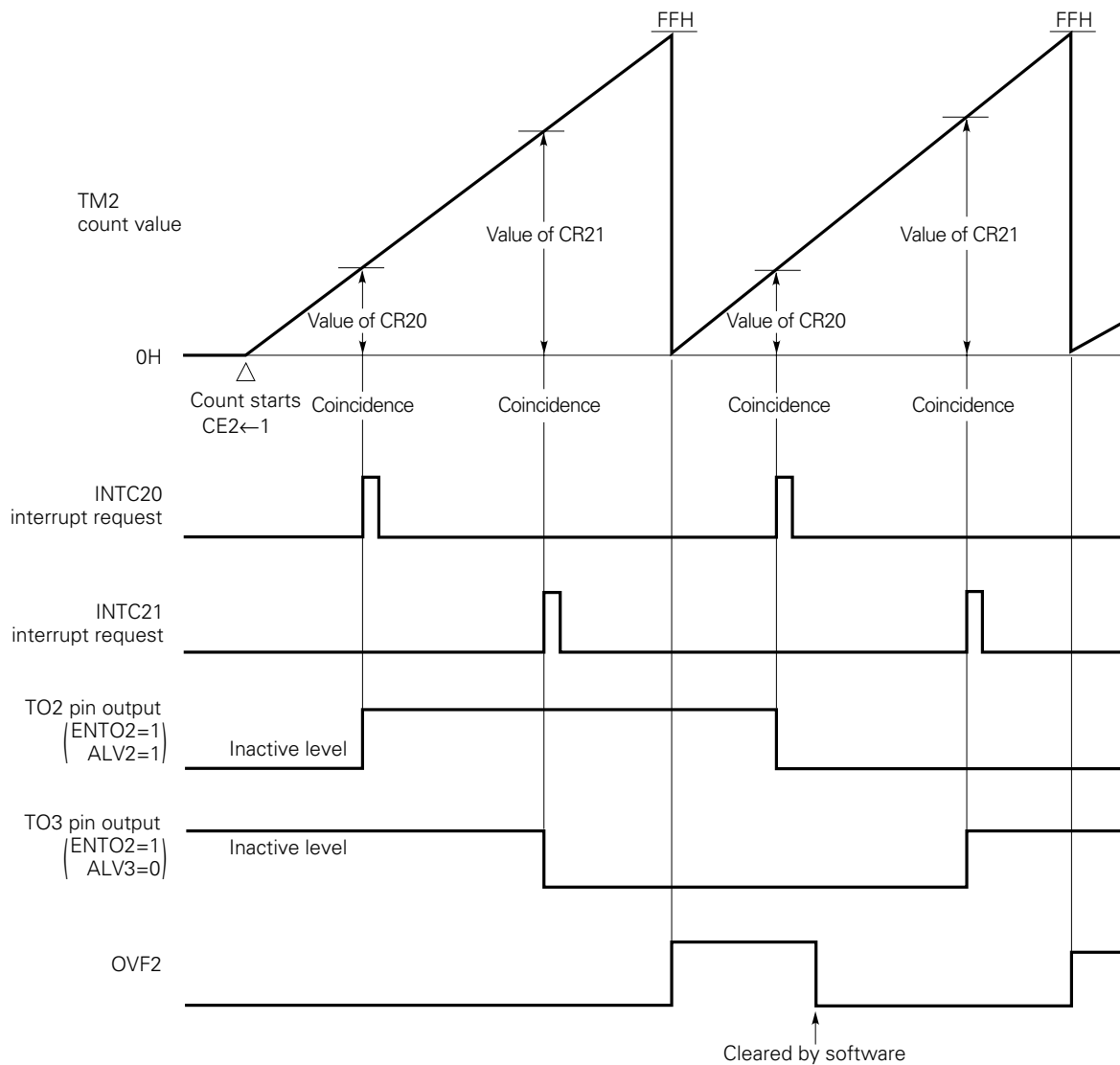
(1) Compare operation

Eight-bit timer/counter 2 performs an operation to compare the values set in the compare registers with timer count values.

When the values set in the compare registers (CR20, CR21) coincide with count values of 8-bit timer 2 (TM2), the coincidence signal is sent to the output control circuit. At the same time, the interrupt requests (INTC20, INTC21) are generated.

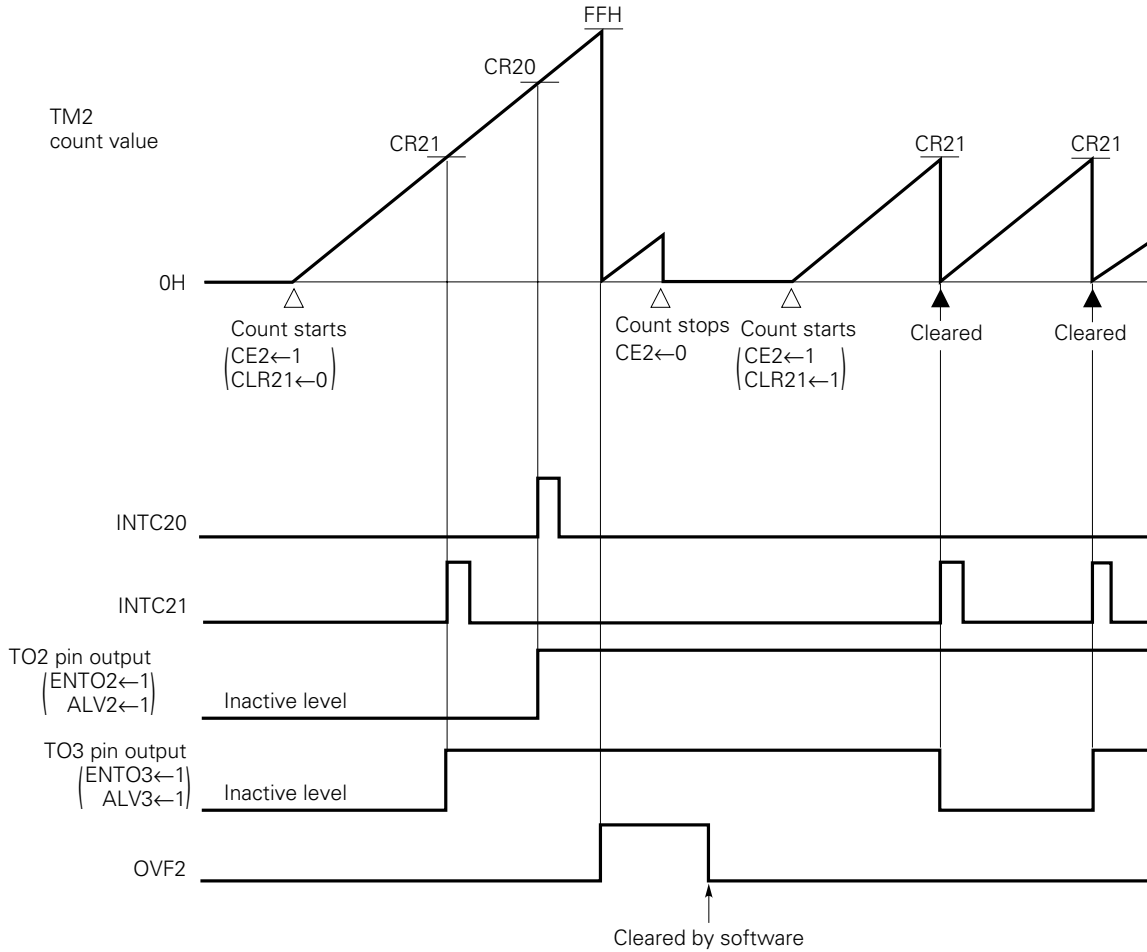
After the value of the CR21 register coincides with a count value of TM2, the count value of TM2 can be cleared. Thus, 8-bit timer/counter 2 can operate as an interval timer for repeatedly counting up to the value set in the CR21 register.

Fig. 7-80 Compare Operation



Remark CLR21 = 0, CLR22 = 0

Fig. 7-81 TM2 Cleared After a Coincidence Is Detected



7

Remark CLR22 = 0

Caution When using an in-circuit emulator, see the notes described in Section 7.5.4.

(2) Capture operation

Eight-bit timer/counter 2 performs a capture operation to load the count value of the timer into the capture register in synchronism with an external trigger.

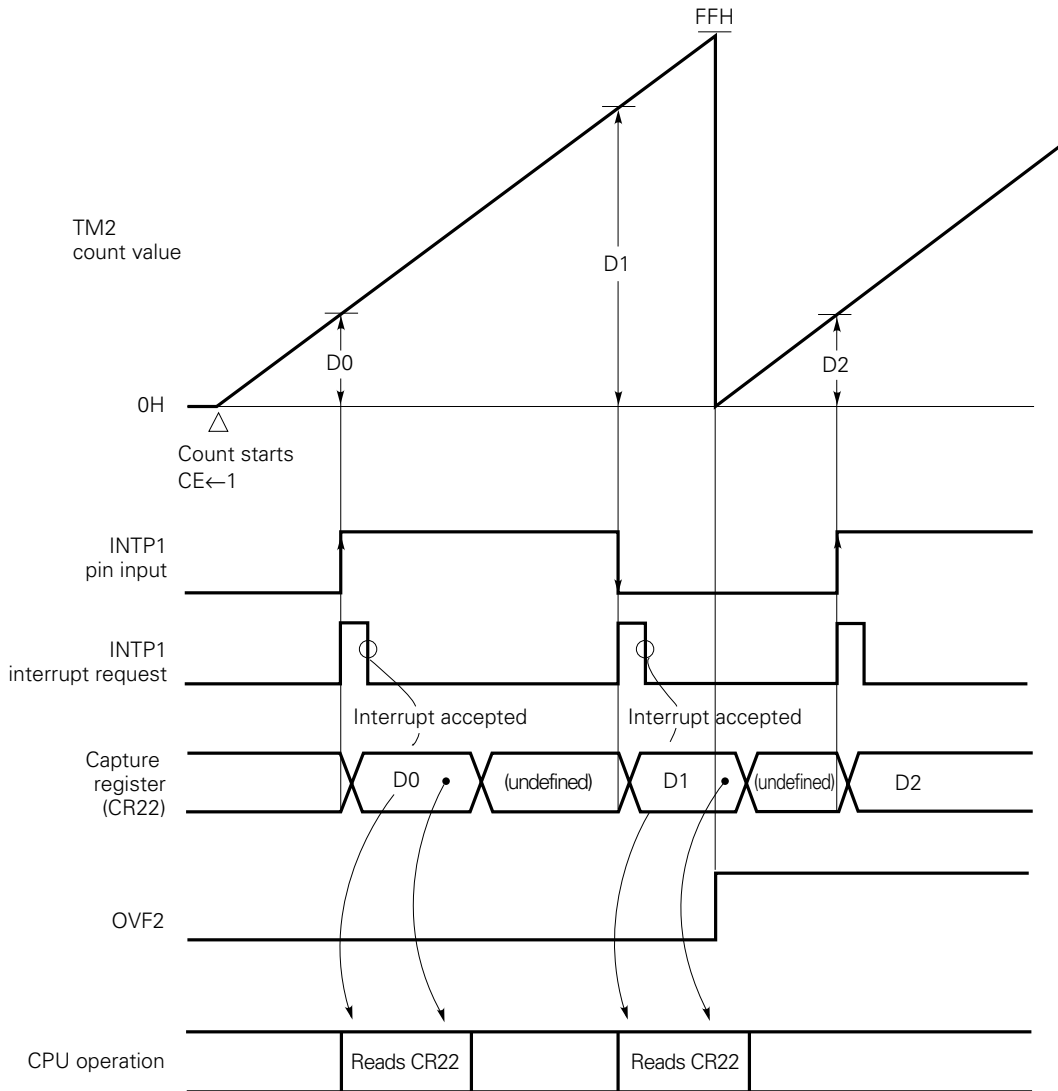
As an external trigger, a valid edge detected on the external interrupt request (INTP1) input pin is used (capture trigger). In synchronism with a capture trigger, the count value of 8-bit timer 2 (TM2) in count operation is loaded and held in the CR22 capture register. After the value captured in the CR22 register is read by the program, the value of the CR22 register becomes undefined.

A valid edge used as a capture trigger is set using external interrupt mode register 0 (INTM0). When both a rising edge and falling edge are set as capture triggers, the pulse width of an applied external signal can be measured. When a capture trigger is generated using one edge, the period of an input pulse signal can be measured.

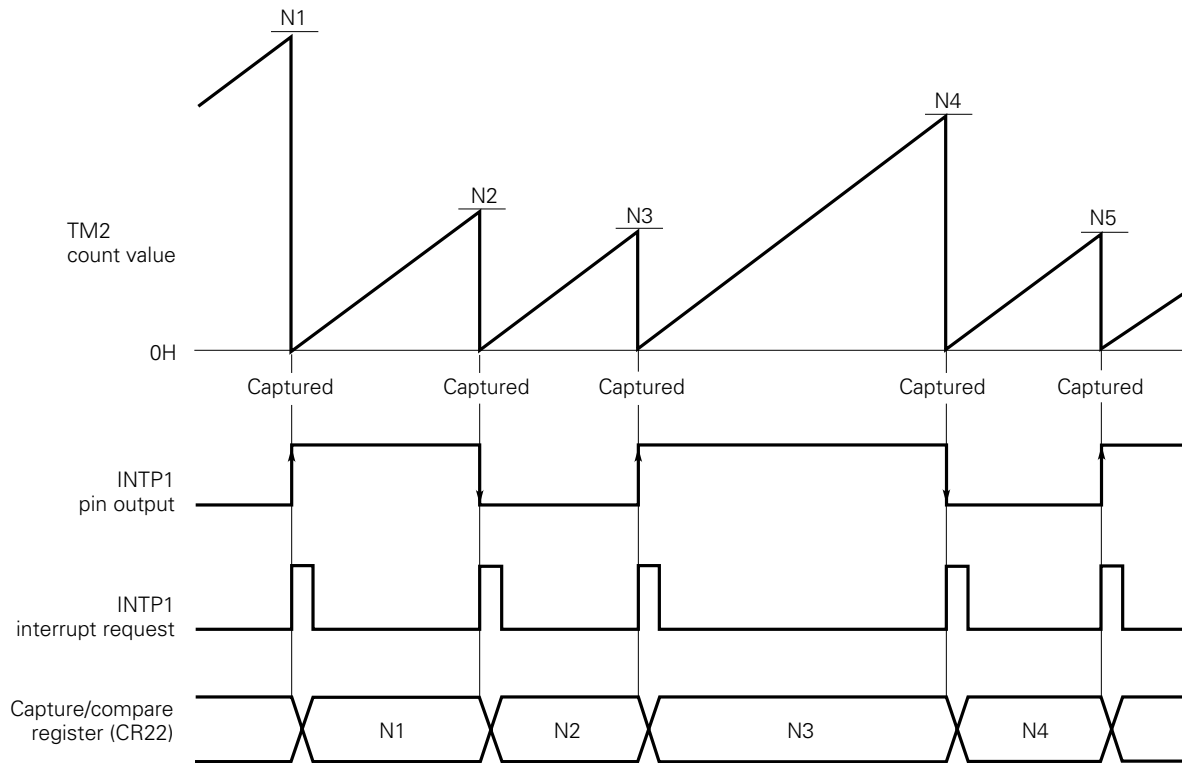
For the detailed format of the INTM0 register, see **Fig. 11-1 in Chapter 11**.

- Cautions**
1. The value of the CR22 register, after being read, becomes undefined. A captured value can be used more than once by saving the captured value to a register or memory.
 2. When using an in-circuit emulator, see the notes described in Section 7.5.4.

Fig. 7-82 Capture Operation



Remark D_n : TM2 count value ($n = 0, 1, 2, \dots$)
 CLR21 = 0, CLR22 = 0

Fig. 7-83 TM2 Cleared after Capture Operation

Remark CLR21 = 0, CLR22 = 1

7.3.8 Basic Operation of Output Control Circuit

The output control circuit controls the levels of the timer outputs (TO2, TO3) according to the coincidence signal from the compare registers. The operation of the output control circuit is determined by timer output control register (TOC) and capture/compare control register 2 (CRC2). (See **Table 7-15**.) Before a timer output (TO0 or TO1) signal can be output on a pin, the pin must be placed in the control mode by the PMC3 register.

Table 7-15 Timer Output (TO2, TO3) Operation

TOC			CRC0				TMC1		TO3	TO2
ENTO3	ALV3	ENTO2	ALV2	MOD1	MOD0	CLR22	CLR21	CMD2		
0	0/1	0	0/1	x	x	x	x	x	Tied high/low	Tied high/low
0	0/1	1	0/1	0	0	x ^{Note}	x	x	Tied high/low	Toggle output (low/high active)
1	0/1	0	0/1	0	0	x ^{Note}	x	x	Toggle output (low/high active)	Tied high/low
1	0/1	1	0/1	0	0	x ^{Note}	x	x	Toggle output (low/high active)	Toggle output (low/high active)
0	0/1	1	0/1	0	1	0	0	0	Tied high/low	PWM output (high/low active)
1	0/1	0	0/1	0	1	0	0	0	Toggle output (low/high active)	Tied high/low
1	0/1	1	0/1	0	1	0	0	0	Toggle output (low/high active)	PWM output (high/low active)
0	0/1	1	0/1	1	0	0	0	0	Tied high/low	PWM output (high/low active)
1	0/1	0	0/1	1	0	0	0	0	PWM output (high/low active)	Tied high/low
1	0/1	1	0/1	1	0	0	0	0	PWM output (high/low active)	PWM output (high/low active)
0	0/1	1	0/1	1	1	0	1	0	Tied high/low	PPG output (high/low active)
1	0/1	0	0/1	1	1	0	1	0	Toggle output (low/high active)	Tied high/low
1	0/1	1	0/1	1	1	0	1	0	Toggle output (low/high active)	PPG output (high/low active)

Note Usually, CLR22 = 0 for these cases.

- Remarks**
1. The numbers 0 and 1 of "0/1" in the ALV3 and ALV2 columns correspond to the words high and low of "high/low" in the TO3 and TO2 columns, respectively.
 2. The character x represents the value 0 or 1.
 3. No combinations other than those indicated in this table are allowed.

(1) Basic operation

By setting ENTOn (n = 2, 3) of the timer output control register (TOC) to 1, the timer outputs (TO2, TO3) can be changed with the timing determined by MOD0, MOD1, and CLR21 of capture/compare control register 2 (CRC2).

In addition, by clearing ENTOn (n = 2, 3) to 0, the levels of the timer outputs (TO2, TO3) can be tied. The level where an output is tied is determined by ALVn (n = 2, 3) of the timer output control register (TOC). When ALVn (n = 2, 3) is 0, the output is tied high; when ALVn (n = 2, 3) is 1, the output is tied low.

(2) Toggle output

Toggle output is an operation mode where the level of output is inverted each time the value of a compare register (CR20, CR21) coincides with the value of 8-bit timer 2 (TM2). The output level of TO2 is inverted when the value of CR20 coincides with the value of TM2. The output level of TO3 is inverted when the value of CR21 coincides with the value of TM2.

When 8-bit timer/counter 2 is stopped by resetting the CE2 bit of the TMC1 register to 0, the output level present at that time is held.

Fig. 7-84 Toggle Output Operation

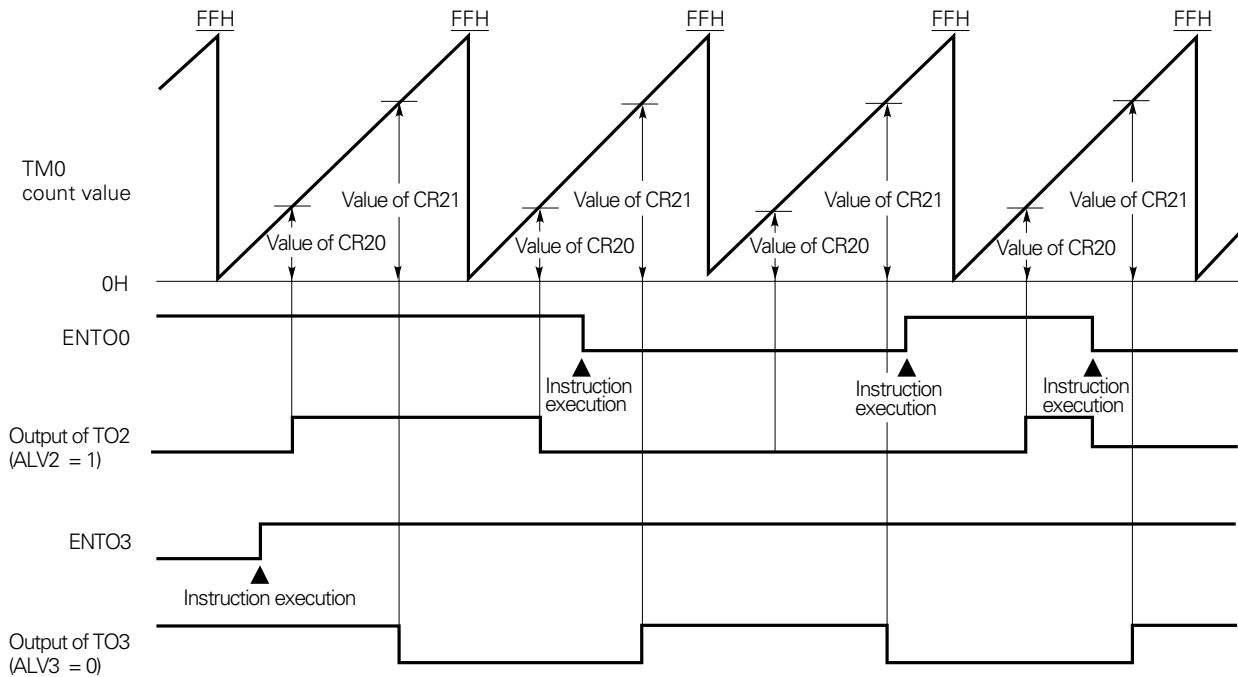


Table 7-16 TO2 and TO3 Toggle Output ($f_{CLK} = 6\text{ MHz}$)

Count clock	Minimum pulse width	Maximum pulse width
$f_{CLK}/16$	2.6 μs	$2^8 \times 16/f_{CLK}$ (683 μs)
$f_{CLK}/32$	5.3 μs	$2^8 \times 32/f_{CLK}$ (1.37 ms)
$f_{CLK}/64$	10.7 μs	$2^8 \times 64/f_{CLK}$ (2.73 ms)
$f_{CLK}/128$	21.3 μs	$2^8 \times 128/f_{CLK}$ (5.46 ms)
$f_{CLK}/256$	42.7 μs	$2^8 \times 256/f_{CLK}$ (10.9 ms)
$f_{CLK}/512$	85.3 μs	$2^8 \times 512/f_{CLK}$ (21.75 ms)

Caution When using an in-circuit emulator, see the notes described in Section 7.5.4.

7.3.9 PWM Output

The PWM output function outputs a PWM signal whose period coincides with the full-count period of 8-bit timer 2 (TM2). The pulse width of TO2 is determined by the value of CR20, and the pulse width of TO3 is determined by the value of CR21. Before this function can be used, the CLR21 and CLR22 bits of capture/compare control register 2 (CRC2) must be set to 0, and the CMD2 bit of timer control register 1 (TMC1) must be set to 0.

The pulse period and pulse width are as follows:

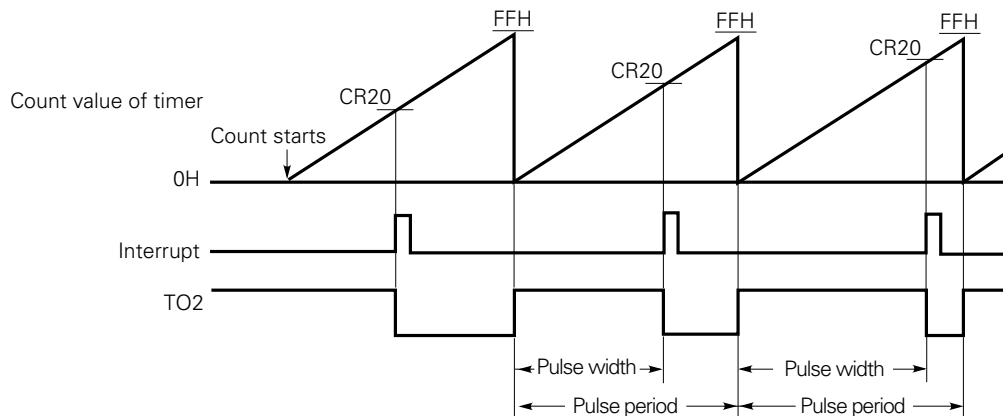
- PWM period = $256 \times x/f_{CLK}$
- PWM pulse width = $((\text{value set in compare register}^{\text{Note}}) \times x + 2)/f_{CLK}$
; $x = 16, 32, 64, 128, 256, 512$

Note Zero cannot be set in the compare registers.

- Duty factor = $(\text{PWM pulse width})/(\text{PWM period}) = ((\text{value set in compare register}) \times x + 2)/(256 \times x) \approx (\text{value set in compare register})/256$

Caution In PWM output, the actual pulse width is longer than a value obtained with the approximate expression by two clock pulses of f_{CLK} for the active level, and is shorter than such an approximate value by two clock pulses of f_{CLK} for the inactive level. Take this point into consideration when high-precision output is required or a high-speed count clock is used.

Fig. 7-85 PWM Pulse Output



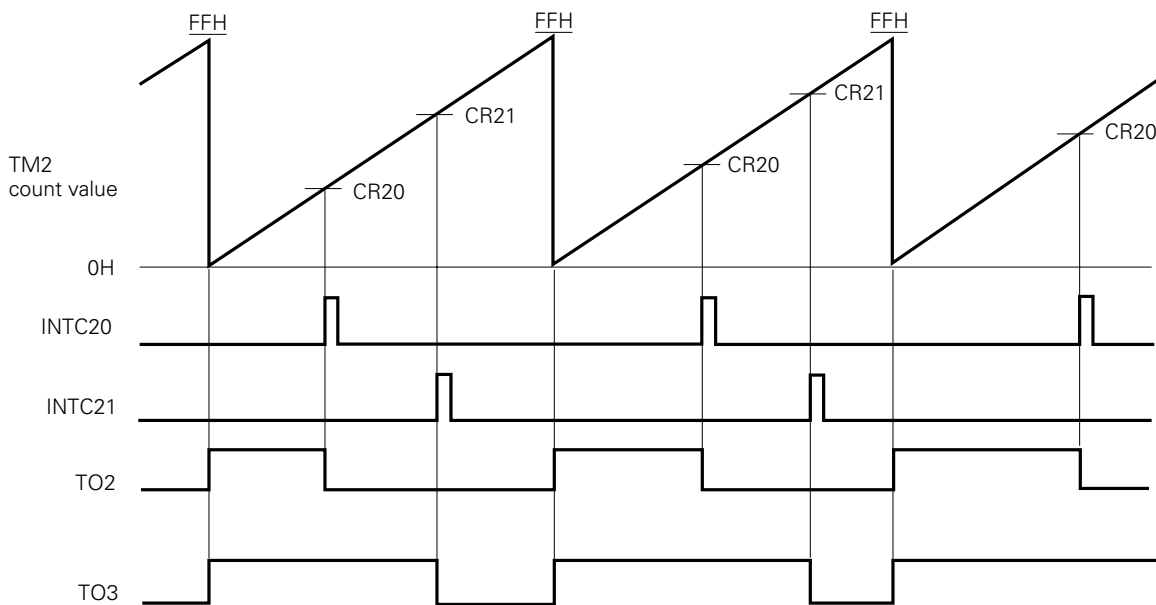
Remark ALV2 = 0

Table 7-17 PWM Output on TO2 and TO3 ($f_{CLK} = 6\text{ MHz}$)

Count clock	Minimum pulse width	PWM period (ms)	PWM frequency (Hz)
$f_{CLK}/16$	2.7	0.7	1465
$f_{CLK}/32$	5.3	1.4	732
$f_{CLK}/64$	10.7	2.7	366
$f_{CLK}/128$	21.3	5.5	183
$f_{CLK}/256$	42.7	10.9	92
$f_{CLK}/512$	85.3	21.8	46

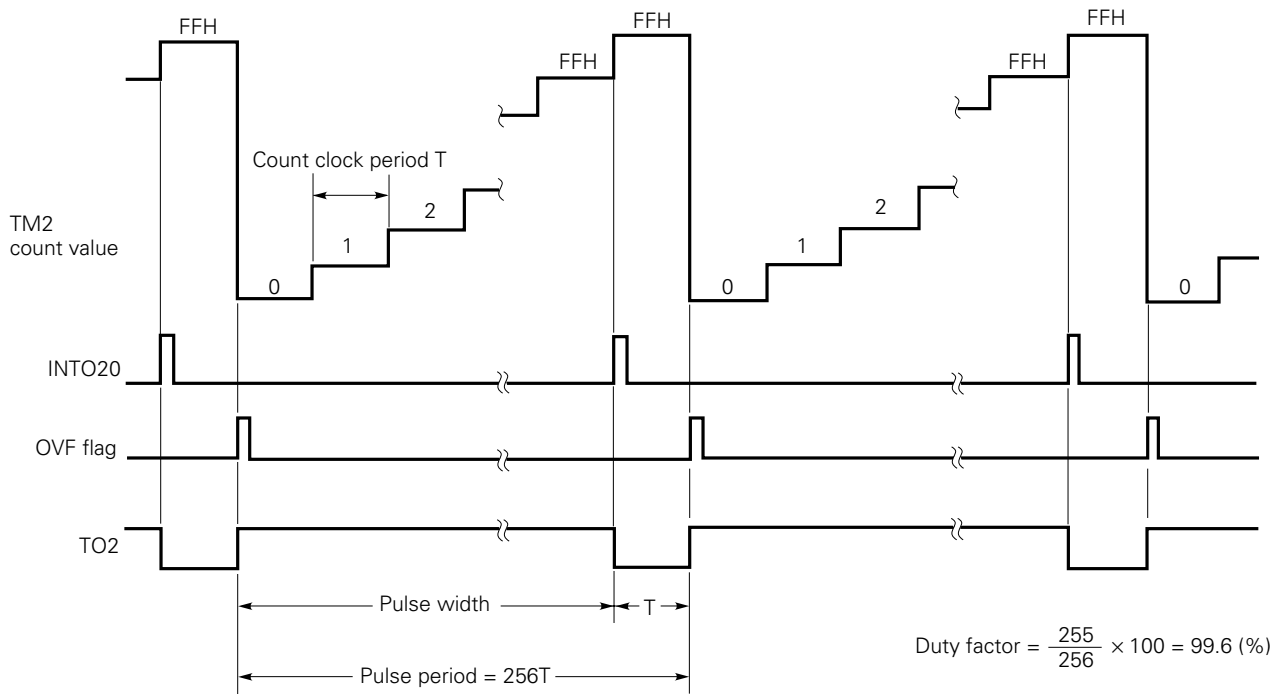
Fig. 7-86 shows an example of 2-channel PWM output. Fig. 7-87 shows PWM output when FFH is set in the CR20 compare register.

Fig. 7-86 Example of PWM Output Using TM2



Remark ALV2 = 0, ALV3 = 0

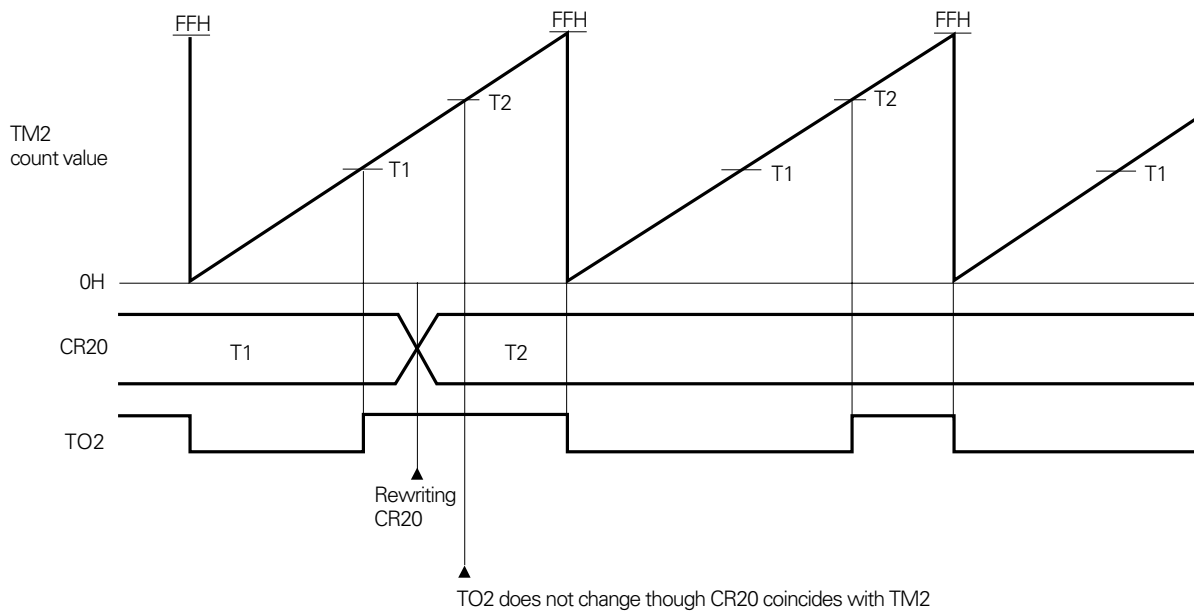
Fig. 7-87 PWM Output When CR20 = FFH



Remark ALV2 = 0

Even if the value of a compare register (CR20, CR21) coincides with the value of 8-bit timer 2 (TM2) more than once during one period of PWM output, the output levels on the timer outputs (TO2, TO3) are not inverted.

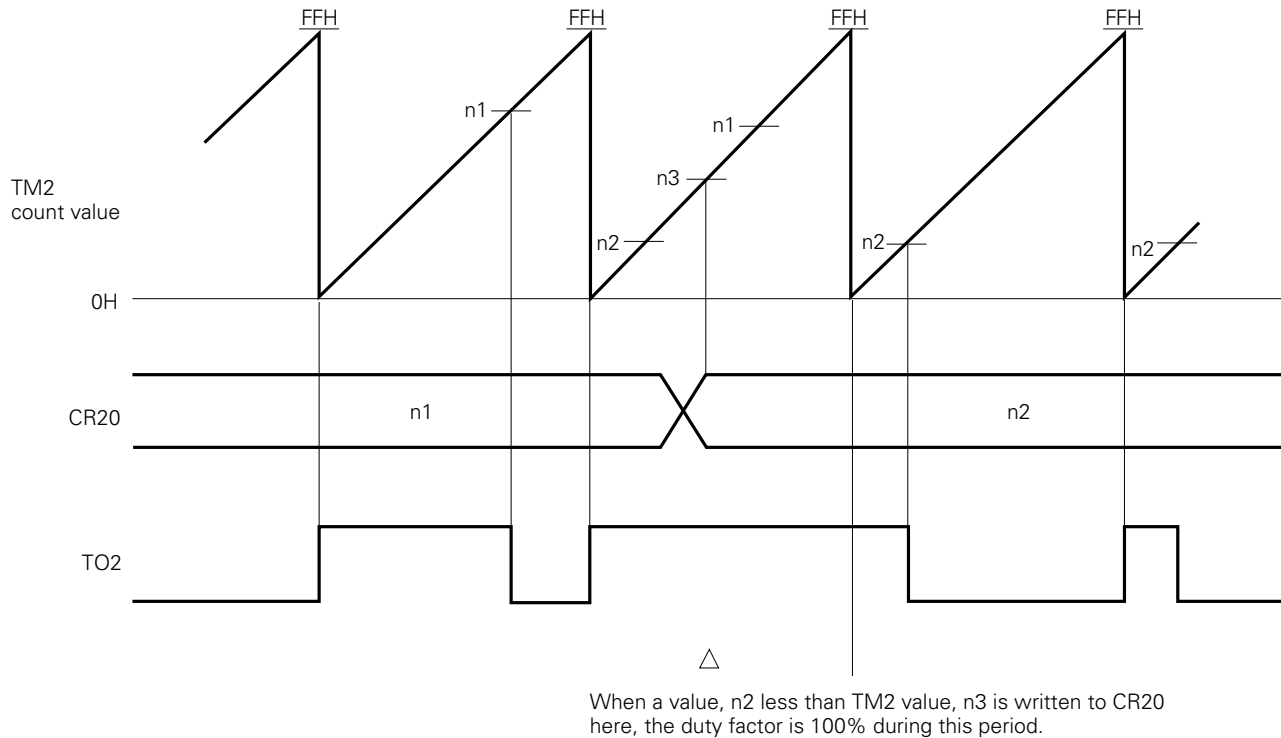
Fig. 7-88 Example of Rewriting a Compare Register



Remark ALV2 = 1

Cautions 1. If a value less than the value of 8-bit timer 2 (TM2) is set in a compare register (CR20, CR21), a PWM signal with a 100% duty factor is output. Rewrite the CR20 or CR21 compare register, if required, by using an interrupt generated by a coincidence between TM2 and the compare register.

Fig. 7-89 Example of PWM Output Signal with a 100% Duty Factor



Remark ALV2 = 0

2. If timer output is disabled (ENTOn = 0: n = 2, 3), the output level on the TOn (n = 2, 3) is the inverted value of the value set in ALVn (n = 2, 3). Accordingly, note that if timer output is disabled when the PWM output function is selected, the active level is output.

7.3.10 PPG Output

The PPG output function outputs a square wave that has a period determined by the CR21 compare register, and has a pulse width determined by the CR20 compare register. PPG output is PWM output whose period is made variable. This output signal can be output on the TO0 pin only.

Before this function can be used, the CLR21 bit of capture/compare control register 2 (CRC2) must be set to 1, the CLR22 bit of the same register must be set to 0, and the CMD2 bit of timer control register 1 (TMC1) must be set to 0.

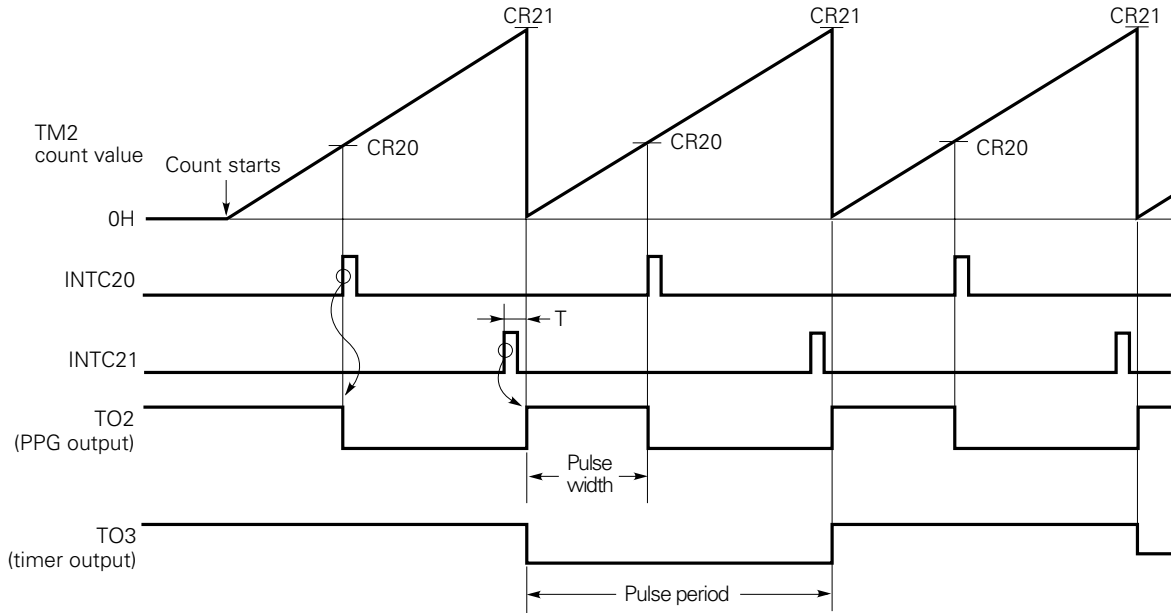
The pulse period and pulse width are as follows:

- PPG period = ((value set in CR21 compare register) + 1) × x / f_{CLK}
; x = 16, 32, 64, 128, 256, 512
- PPG pulse width = ((value set in CR20 compare register) × x + 2) / f_{CLK} ≐ (value set in CR20) × x / f_{CLK}
where, CR20 ≤ CR21
- Duty factor = (PPG pulse width) / (PPG period) = ((value set in CR20) × x + 2) / (((value set in CR21) + 1) × x) ≐ (value set in CR20) / ((value set in CR21) + 1)

Caution In PPG output, the actual pulse width is longer than a value obtained with the approximate expression by two clock pulses of f_{CLK} for the active level, and is shorter than such an approximate value by two clock pulses of f_{CLK} for the inactive level. Take this point into consideration when high-precision output is required, the PPG pulse period is short, or a high-speed count clock is used.

Fig. 7-90 shows an example of PPG output using 8-bit timer 2 (TM2). Fig. 7-91 shows an example of PPG output when CR20 = CR21. Fig. 7-92 shows an example of PPG output when CR20 = 00H.

Fig. 7-90 Example of PPG Output Using TM2



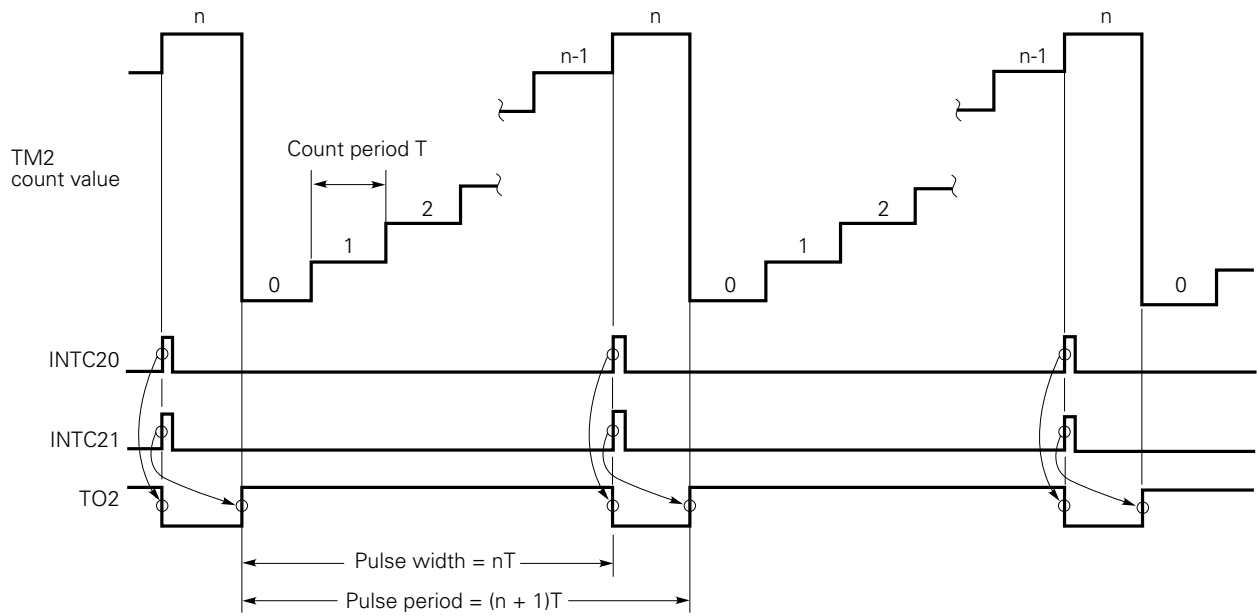
Remark ALV2 = 0, ALV3 = 0

Table 7-18 PPG Output on TO2 ($f_{CLK} = 6\text{ MHz}$)

Count clock	Minimum pulse width ^{Note}	PPG period	PPG frequency
$f_{CLK}/16$	2.67 μs	5.33 μs to 683 μs	187.5 kHz to 1.46 kHz
$f_{CLK}/32$	5.33 μs	10.7 μs to 1.37 ms	93.75 kHz to 732 Hz
$f_{CLK}/64$	10.7 μs	21.3 μs to 2.73 ms	46.9 kHz to 366 Hz
$f_{CLK}/128$	21.3 μs	42.7 μs to 5.46 ms	23.4 kHz to 183 Hz
$f_{CLK}/256$	42.7 μs	85.3 μs to 10.9 ms	11.7 kHz to 91.6 Hz
$f_{CLK}/512$	85.3 μs	171 μs to 21.8 ms	5.86 kHz to 45.8 Hz

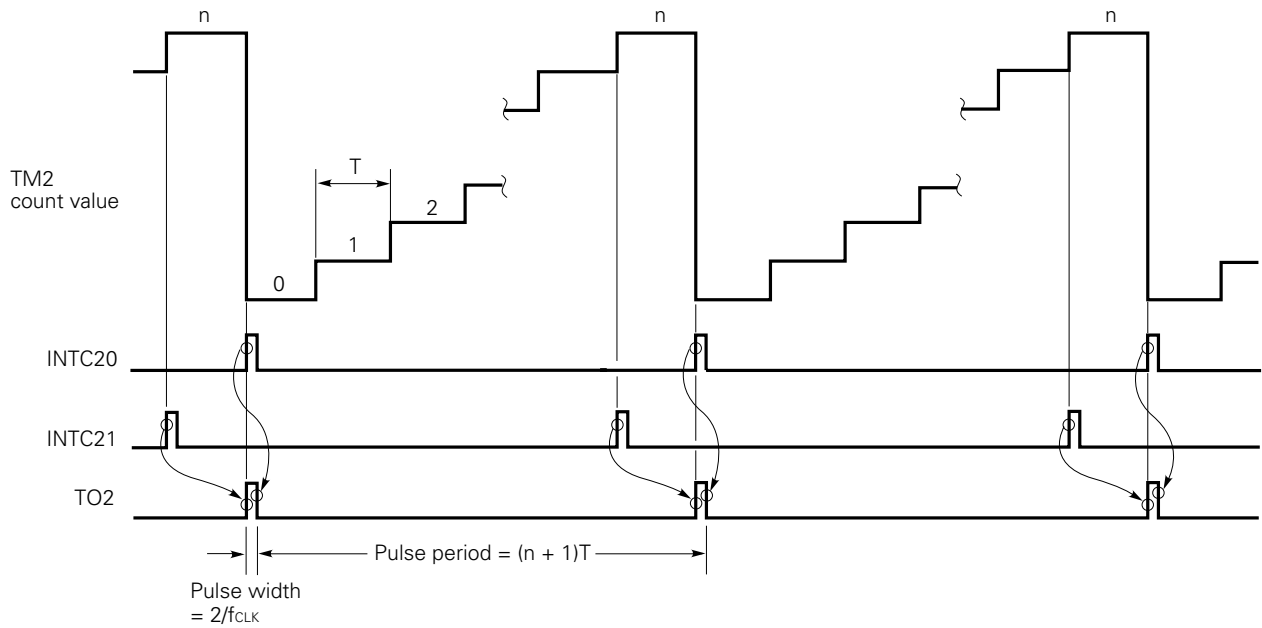
Note The case where CR20 = 0 is excluded.

Fig. 7-91 PPG Output When CR20 = CR21



Remark ALV2 = 0

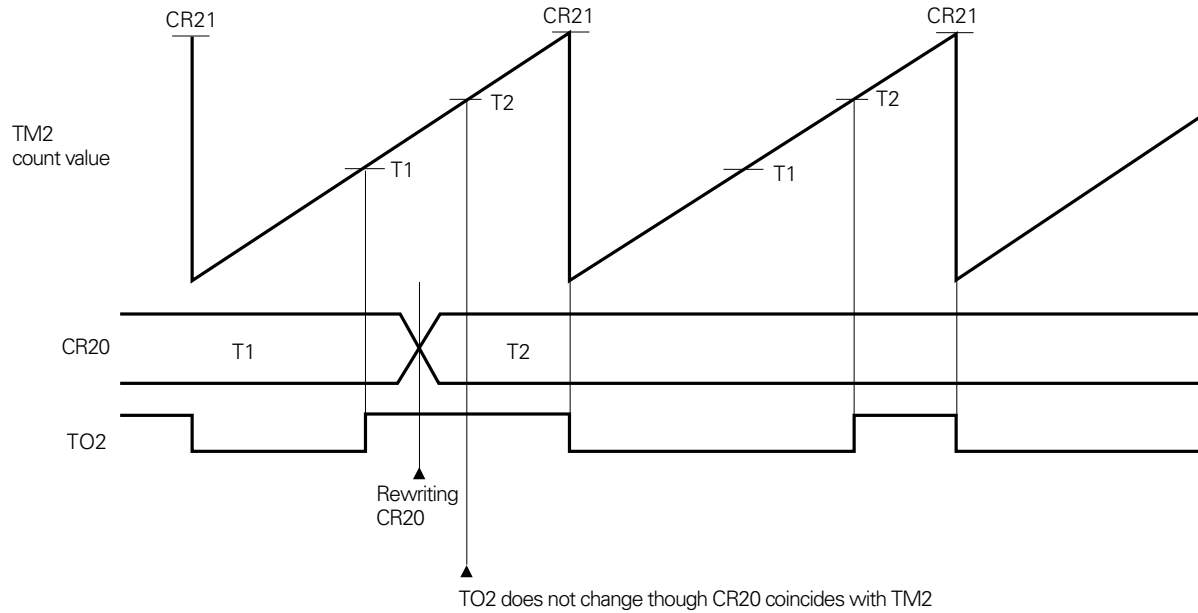
Fig. 7-92 PPG Output When CR20 = 00H



Remark ALV2 = 0

Even if the value of the CR20 compare register coincides with the value of 8-bit timer 2 (TM2) more than once during one period of PPG output, the output level on the timer output (TO2) is not inverted.

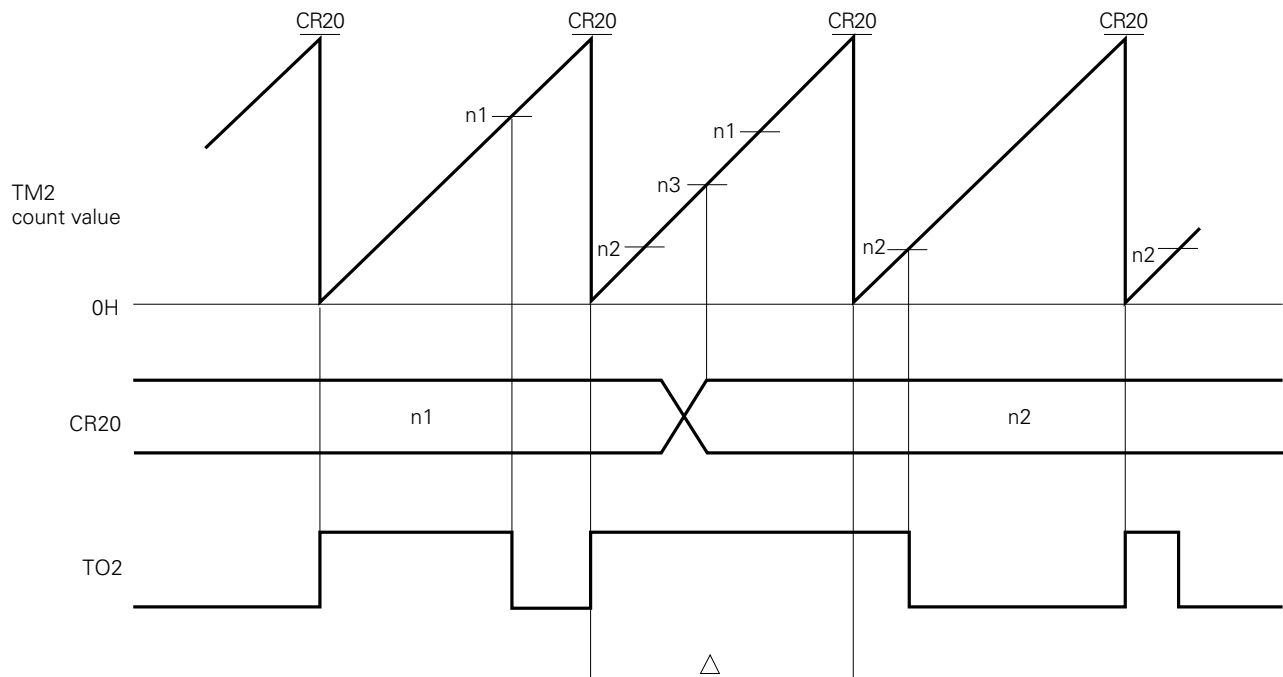
Fig. 7-93 Example of Rewriting Compare Register CR20



Remark ALV2 = 1

Cautions 1. If a value less than the value of 8-bit timer 2 (TM2) is written into the CR20 compare register before the value of CR20 coincides with the value of TM2, a PPG signal with a 100% duty factor is output in that period. Rewrite CR20, if required, by using an interrupt generated by a coincidence between TM2 and CR20.

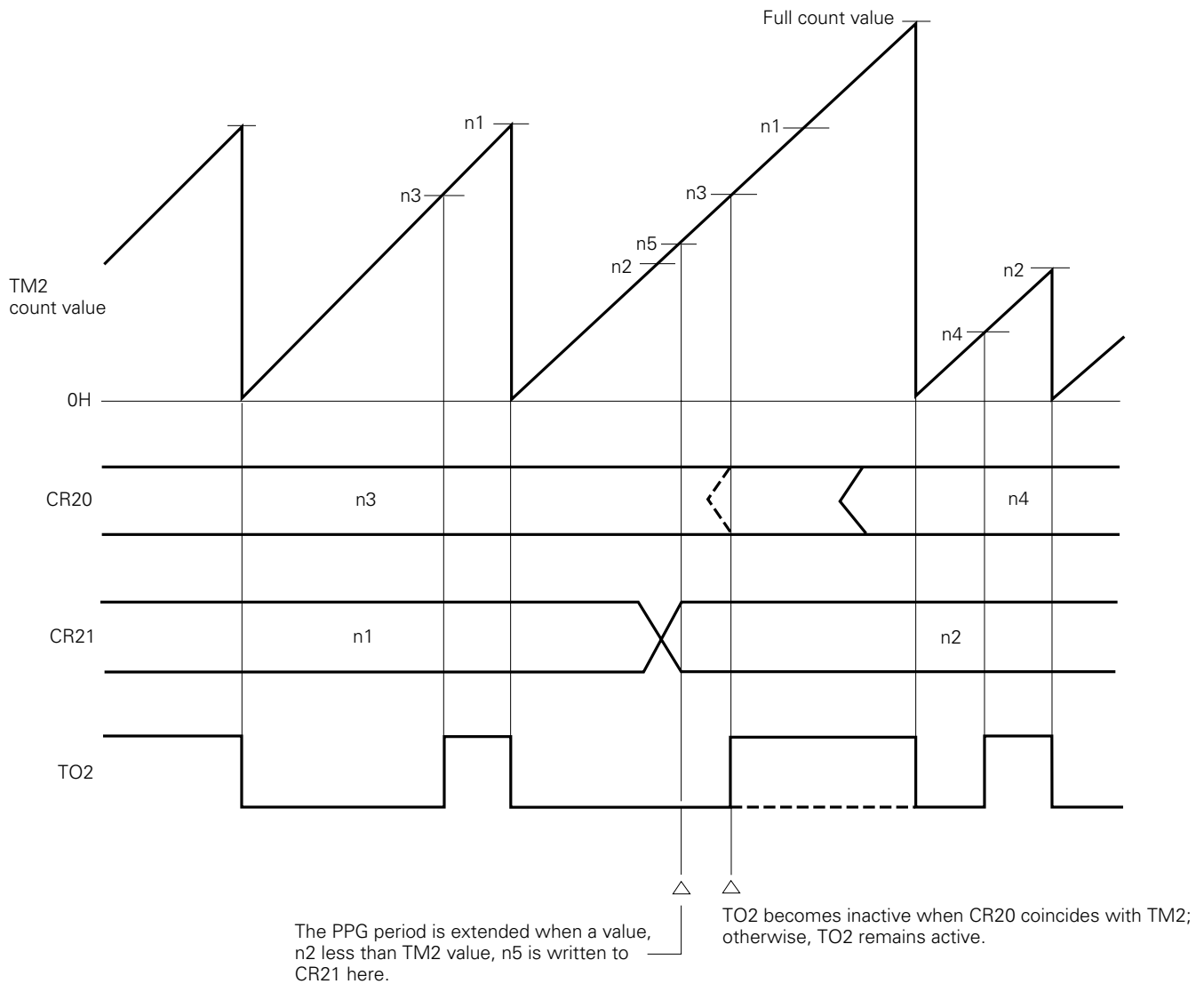
Fig. 7-94 Example of PPG Output Signal with a 100% Duty Factor



Remark ALV2 = 0

2. If the current value of the CR21 compare register is decreased below the value of 8-bit timer 2 (TM2), the PPG period becomes as long as the full-count time of TM2. At this time, if CR21 is rewritten after the value of the CR20 compare register coincides with the value of TM2, the inactive level is output until TM2 overflows to 0, then normal PPG output is resumed. If CR21 is rewritten before the value of CR20 coincides with the value of TM2, the active level is output until the value of CR20 coincides with the value of TM2. When the value of CR20 coincides with the value of TM2 before TM2 overflows to 0, the inactive level is output at that time. When TM2 overflows to 0, the active level is output, and normal PPG output is resumed. Rewrite CR21, if required, by using an interrupt generated by a coincidence between TM2 and CR21.

Fig. 7-95 Example of PPG Output Period Made Longer



Remark ALV2 = 1

3. If the PPG period is too short for interrupt acceptance, the measures described in Cautions 1 and 2 above do not lead to solution. Consider other measures (such as masking all interrupts and polling interrupt request flags by software).
4. If timer output is disabled (ENTOn = 0: n = 2, 3), the output level on the TOn (n = 2, 3) is the inverted value of the value set in ALVn (n = 2, 3). Accordingly, note that if timer output is disabled when the PPG output function is selected, the active level is output.

7.3.11 Sample Applications

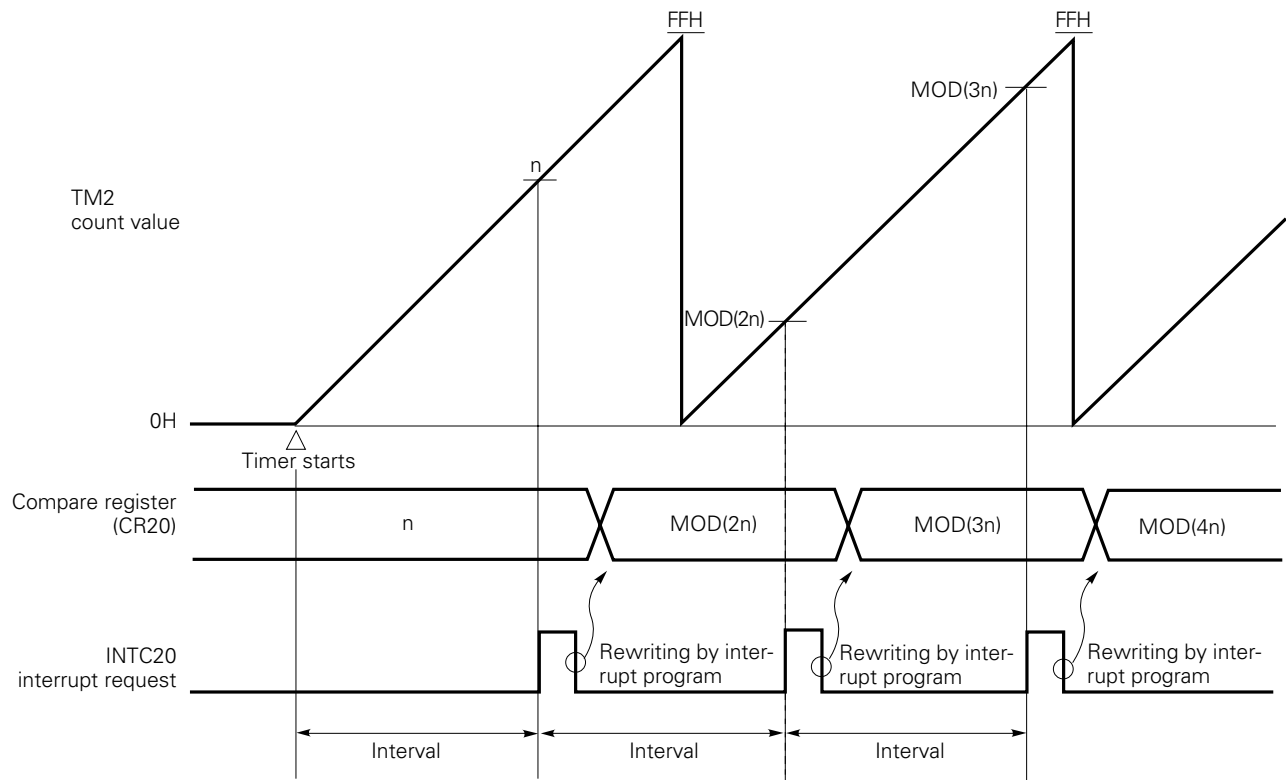
(1) Interval timer operation (1)

By free running 8-bit timer 2 (TM2), and adding a value to a compare register (CR20, CR21) in an interrupt handling routine, 8-bit timer/counter 2 can be used as an interval timer whose period is as long as the added value. (See Fig. 7-96.)

In addition, 8-bit timer 2 (TM2) has two compare registers, so that interval timers with two types of periods can be produced.

Fig. 7-97 shows the setting of control registers. Fig. 7-98 shows the setting procedure. Fig. 7-99 shows interrupt handling.

Fig. 7-96 Timing of Interval Timer Operation (1)



Remark Interval = $n \times x / f_{CLK}$, $1 \leq n \leq FFH$
 $x = 16, 32, 64, 128, 256, 512$

Fig. 7-97 Setting of Control Registers for Interval Timer Operation (1)

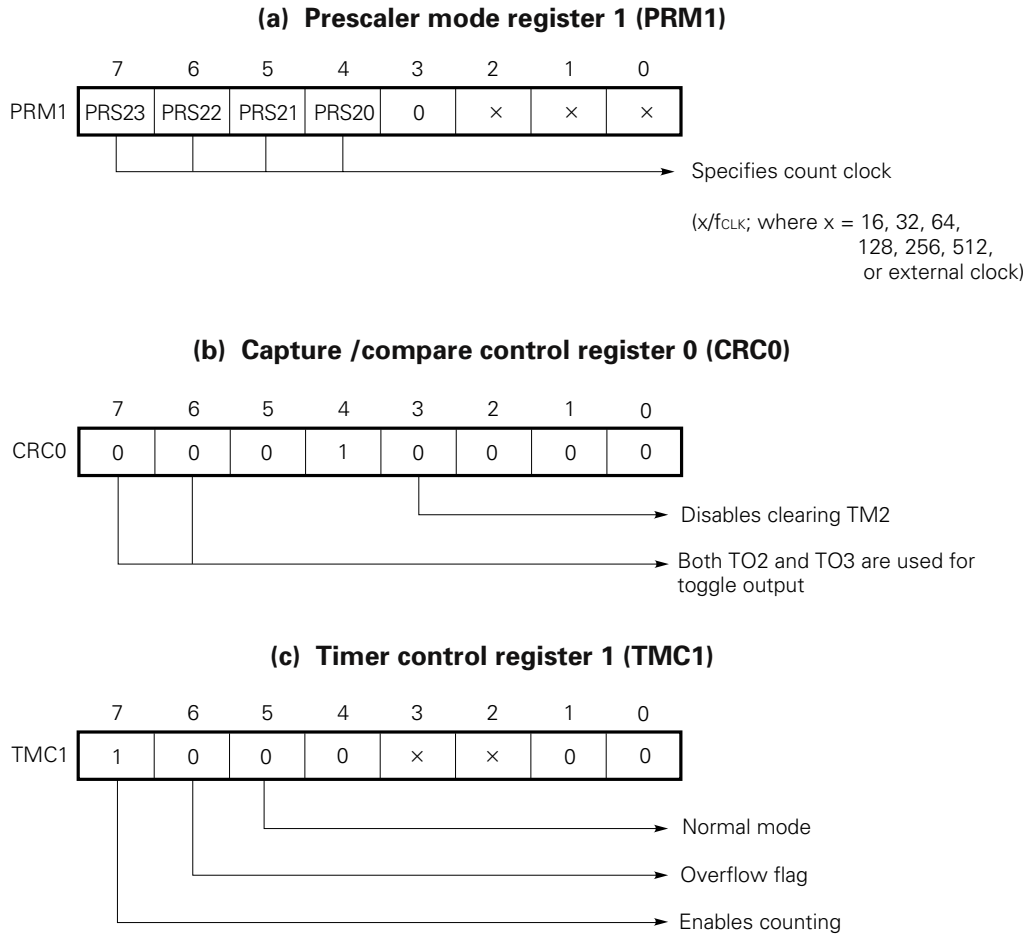


Fig. 7-98 Setting Procedure for Interval Timer Operation (1)

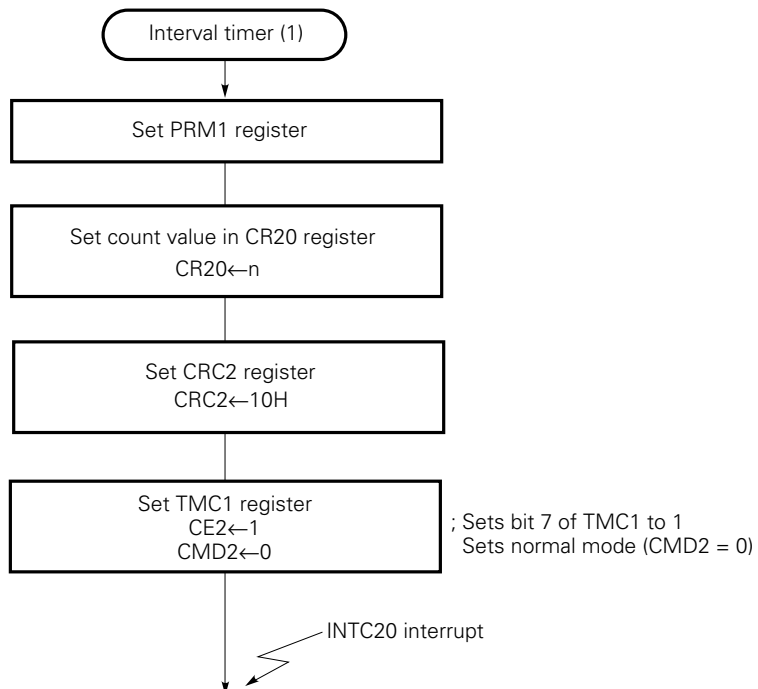
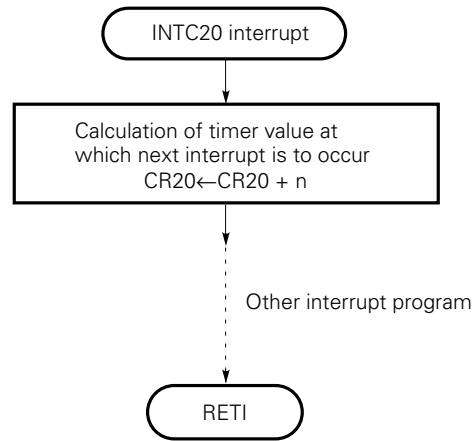


Fig. 7-99 Interrupt Request Handling for Interval Timer Operation (1)

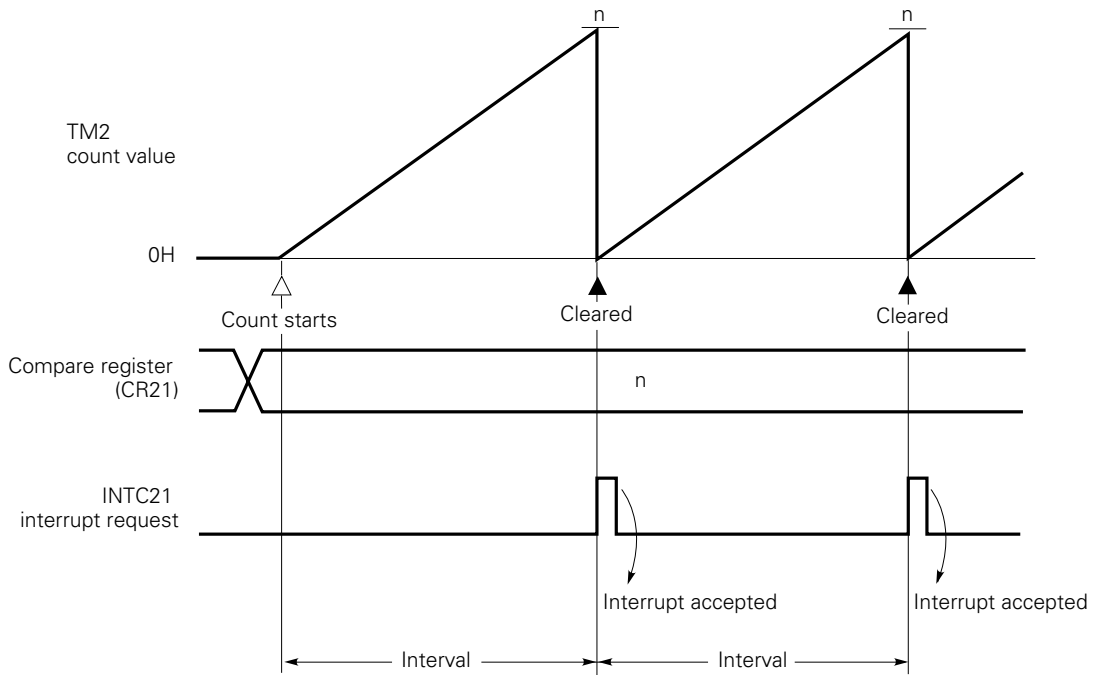


(2) Interval timer operation (2)

Eight-bit timer/counter 2 can be used as an interval timer that generates an interrupt at intervals of a count time specified beforehand. (See Fig. 7-100.)

Fig. 7-101 shows the setting of control registers, and Fig. 7-102 shows the setting procedure.

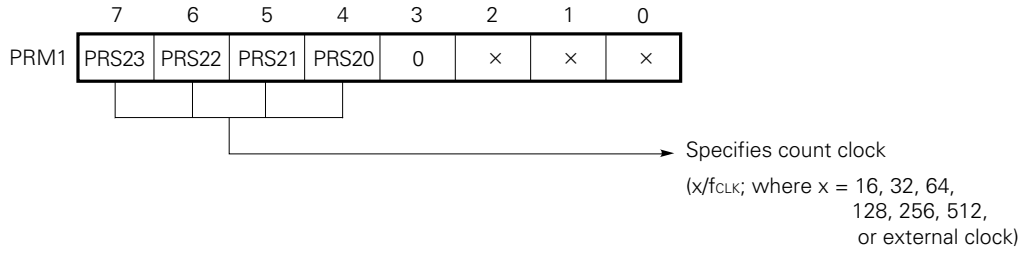
Fig. 7-100 Timing of Interval Timer Operation (2)



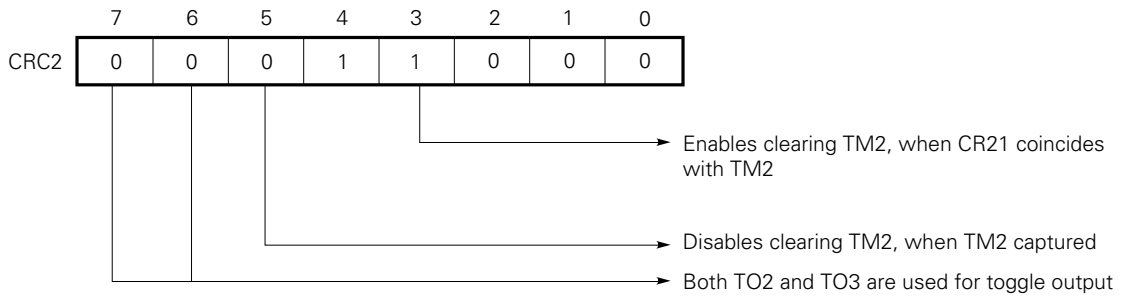
Remark Interval = $(n + 1) \times x / f_{CLK}$, $0 \leq n \leq FFH$
 $x = 16, 32, 64, 128, 256, 512$

Fig. 7-101 Setting of Control Registers for Interval Timer Operation (2)

(a) Prescaler mode register 1 (PRM1)



(b) Capture/compare control register 2 (CRC2)



(c) Timer control register 1 (TMC1)

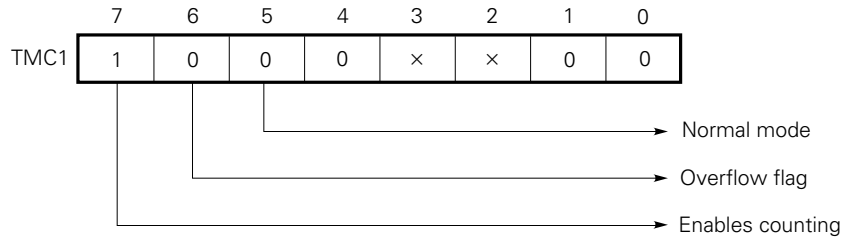
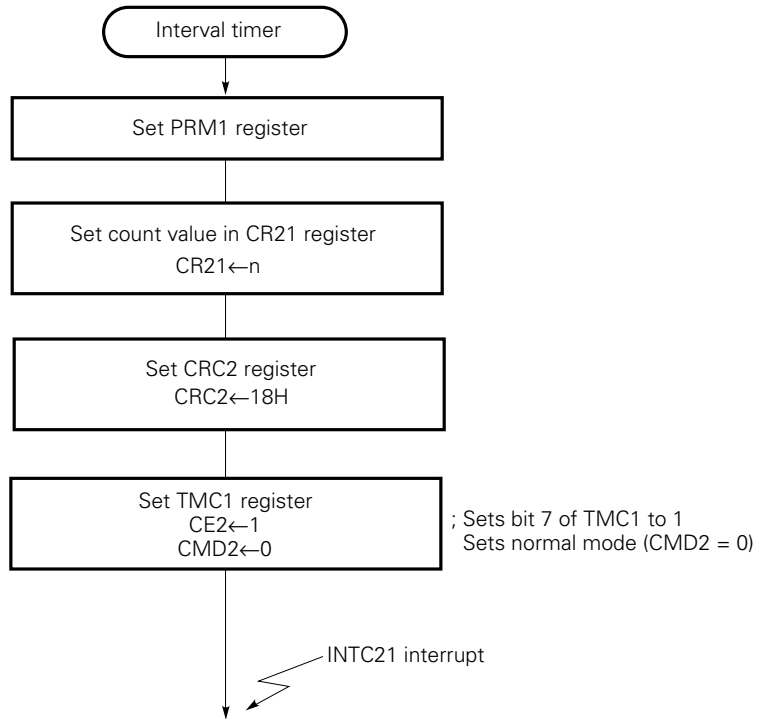


Fig. 7-102 Setting Procedure for Interval Timer Operation (2)



(3) Pulse width measurement operation

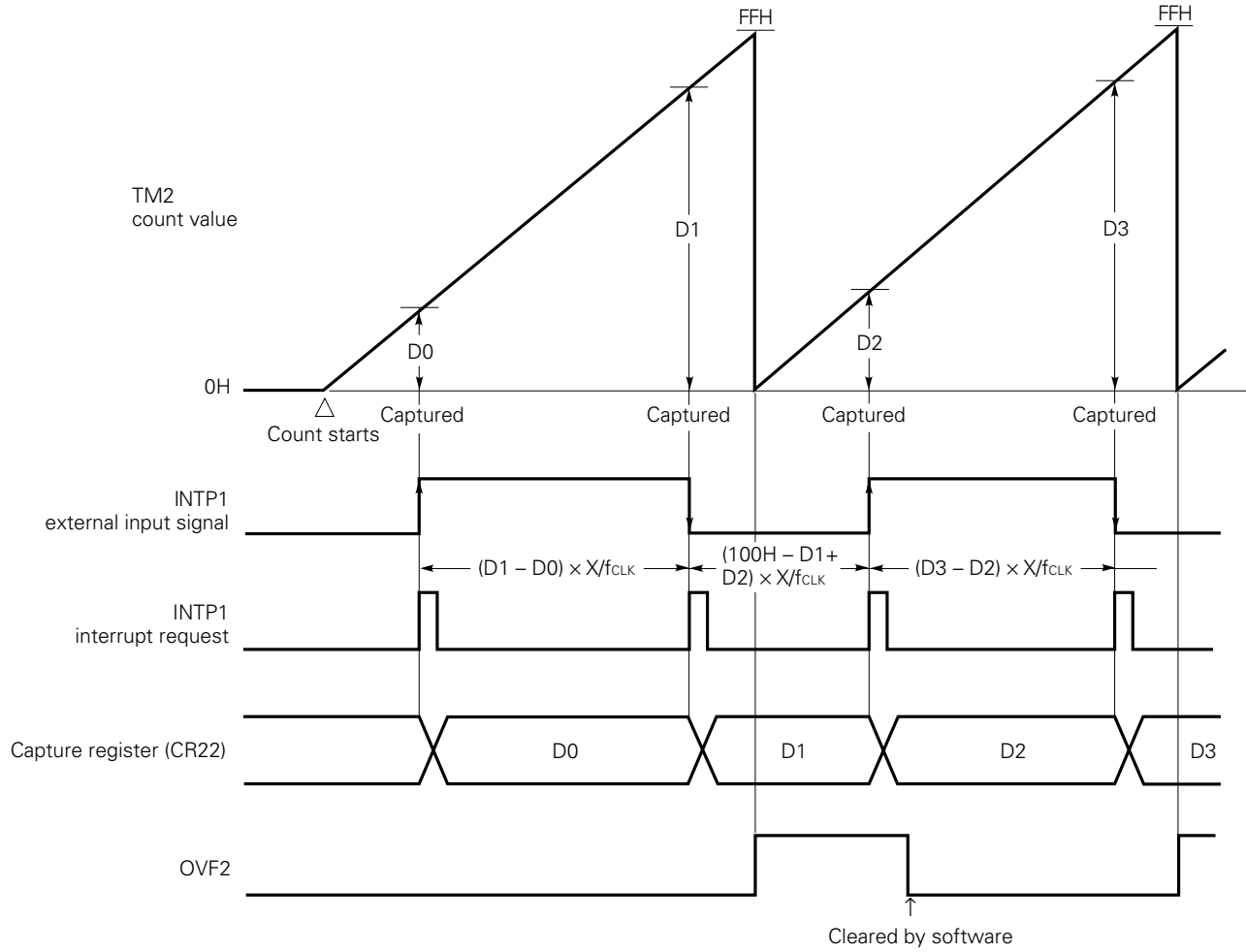
In pulse width measurement, the width of the high level or low level of an external pulse signal applied to the INTP1 pin is measured.

A pulse signal applied to the INTP1 pin must have a pulse width of 12 system clock pulses ($2 \mu\text{s}$: $f_{\text{CLK}} = 6 \text{ MHz}$) or more for both the high level and the low level. If the pulse width is less than this value, no valid edge can be detected, thus resulting in a failure to perform capture operation.

As shown in Fig. 7-103, the value of 8-bit timer 2 (TM2) in count operation is loaded and held in the CR22 capture register on a valid edge (either a rising edge or falling edge) of a signal applied to the INTP1 pin. The pulse width of the input signal is found by multiplying the count clock by the difference between the TM2 count value (D_n) loaded and held in the CR22 register on the n -th valid edge detected and the TM2 count value (D_{n-1}) loaded and held in the CR22 register on the $(n - 1)$ -th valid edge detected.

Fig. 7-104 shows the setting of control registers, and Fig. 7-105 shows the setting procedure.

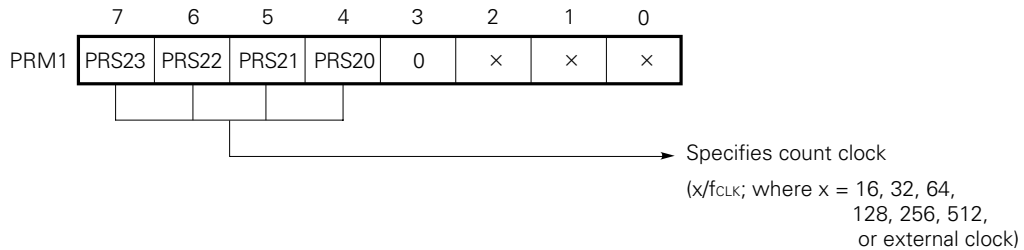
Fig. 7-103 Timing of Pulse Width Measurement



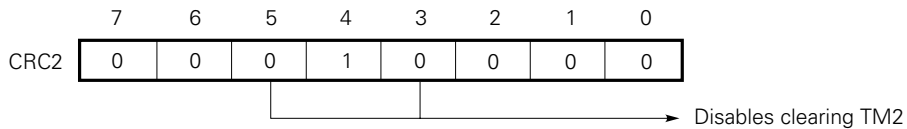
Remark D_n : TM2 count value ($n = 0, 1, 2, \dots$)

Fig. 7-104 Setting of Control Registers for Pulse Width Measurement

(a) Prescaler mode register 1 (PRM1)



(b) Capture/compare control register 2 (CRC2)



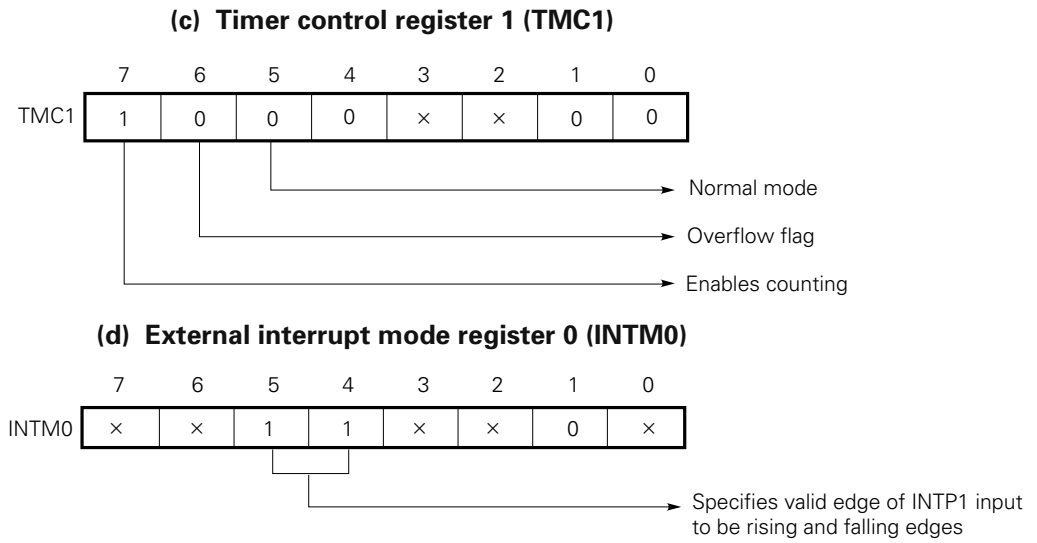


Fig. 7-105 Setting Procedure for Pulse Width Measurement

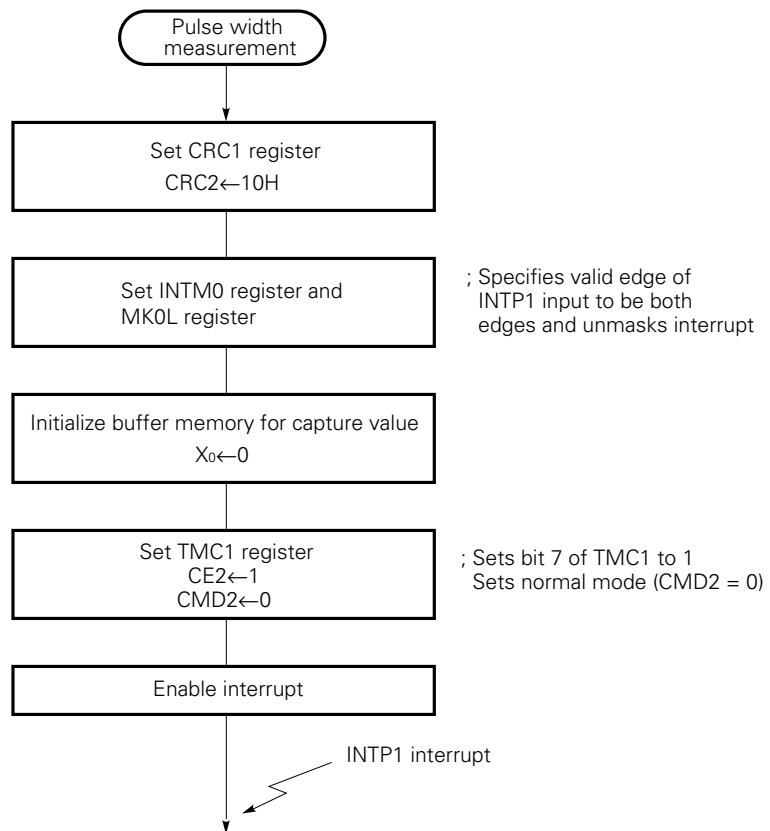
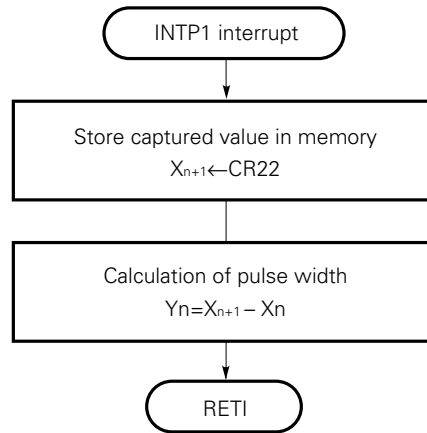


Fig. 7-106 Interrupt Request Handling for Pulse Width Calculation**(4) PWM output operation**

In PWM output operation, a pulse signal with a duty factor determined by the value set in a compare register is output. (See Fig. 7-107.)

The duty factor of a PWM output signal can be changed in steps of 1/256 from 1/256 to 255/256.

In addition, 8-bit timer 2 (TM2) has two compare registers, so that two types of PWM signals can be output.

Fig. 7-108 shows the setting of control registers. Fig. 7-109 shows the setting procedure. Fig. 7-110 shows the procedure for changing the duty factor of PWM output.

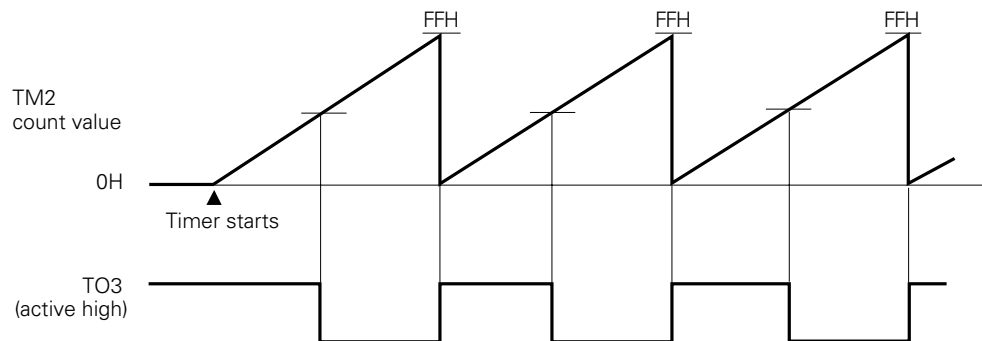
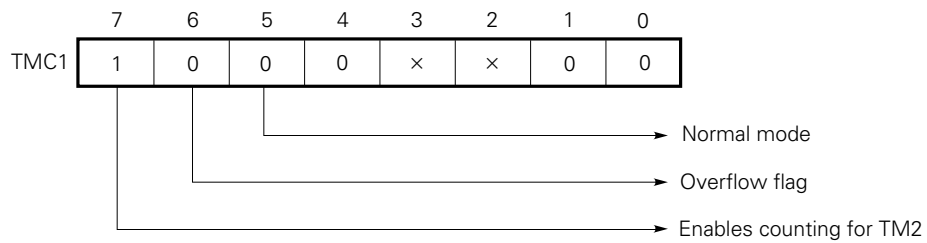
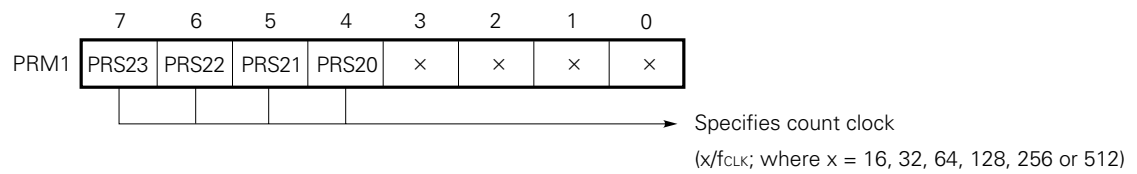
Fig. 7-107 Example of PWM Signal Output by 8-Bit Timer/Counter 2

Fig. 7-108 Setting of Control Registers for PWM Output Operation

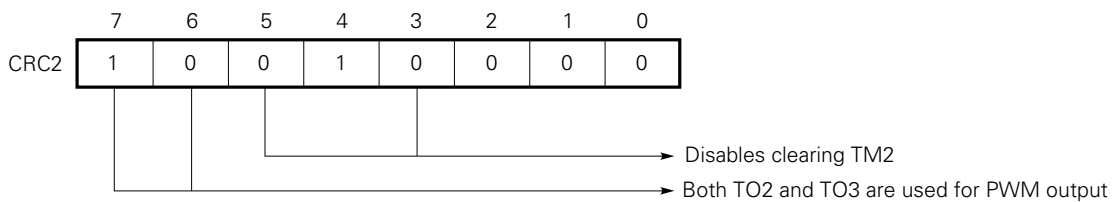
(a) Timer control register 1 (TMC1)



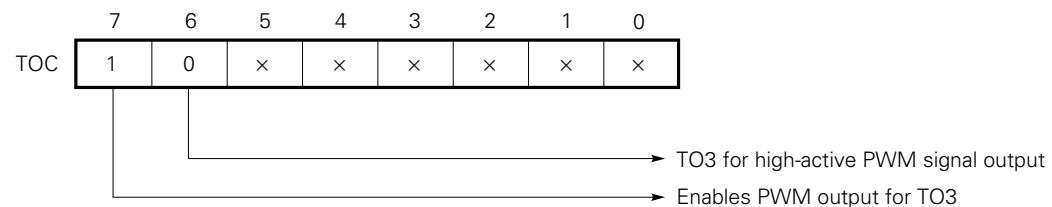
(b) Prescaler mode register 1 (PRM1)



(c) Capture/compare control register 2 (CRC2)



(d) Timer output control register (TOC)



(e) Port 3 mode control register (PMC3)

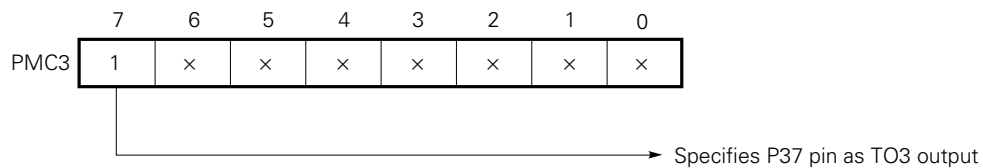


Fig. 7-109 Setting Procedure for PWM Output

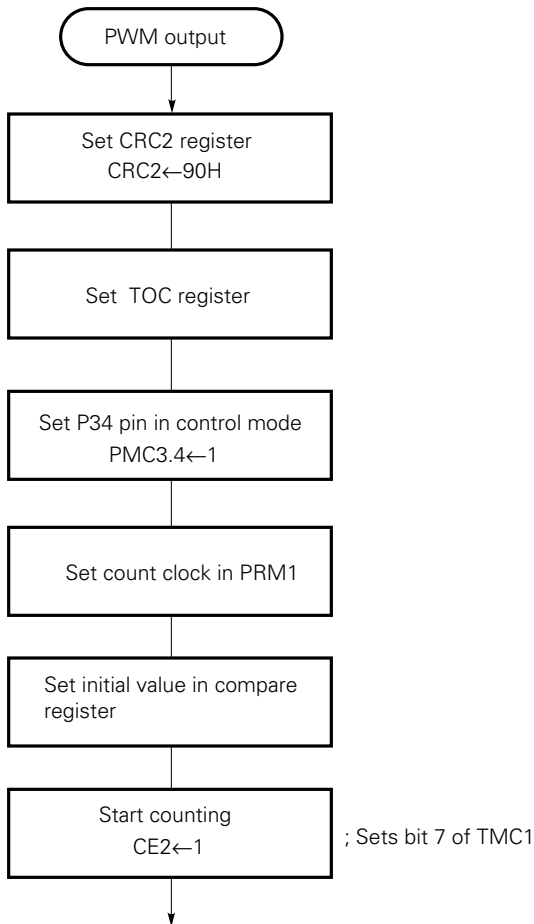
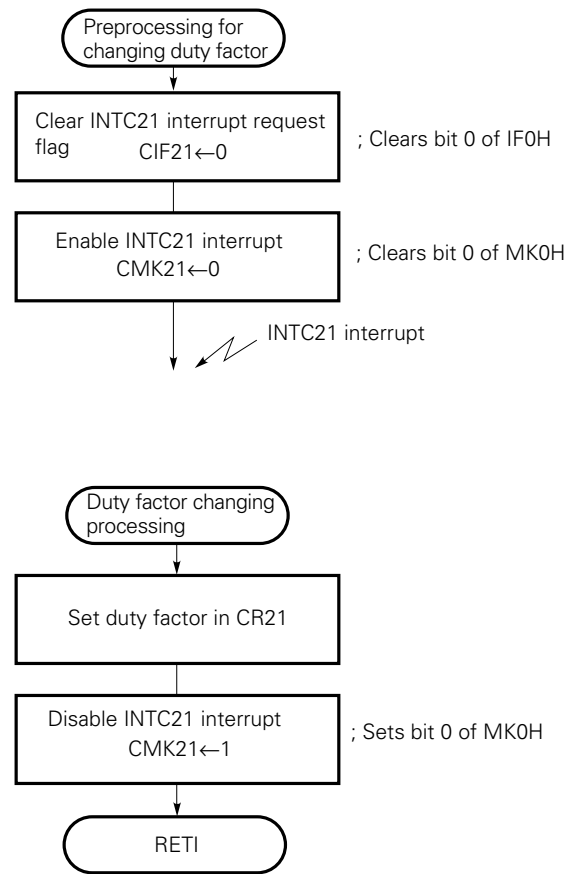


Fig. 7-110 Changing Duty Factor of PWM Output



(5) PPG output operation

In PPG output operation, a pulse signal with a period and duty factor determined by the values set in the compare registers is output. (See Fig. 7-111.)

Fig. 7-112 shows the setting of control registers. Fig. 7-113 shows the setting procedure. Fig. 7-114 shows the procedure for changing the duty factor of PPG output.

Fig. 7-111 Example of PPG Signal Output by 8-Bit Timer/Counter 2

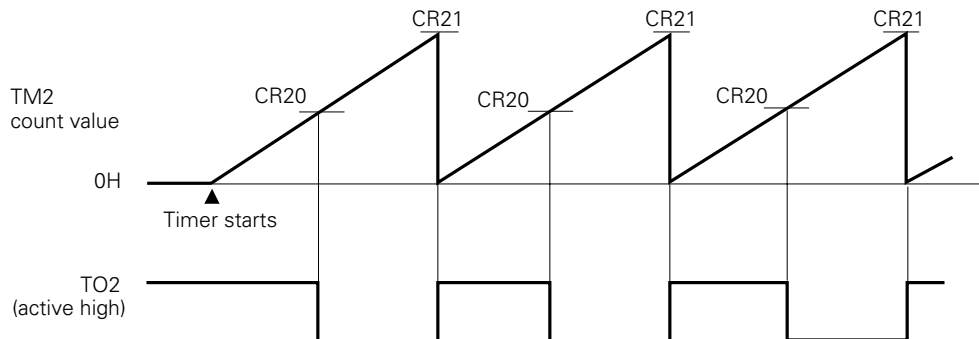
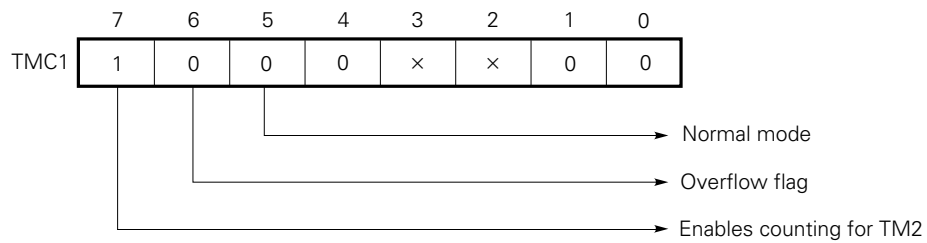
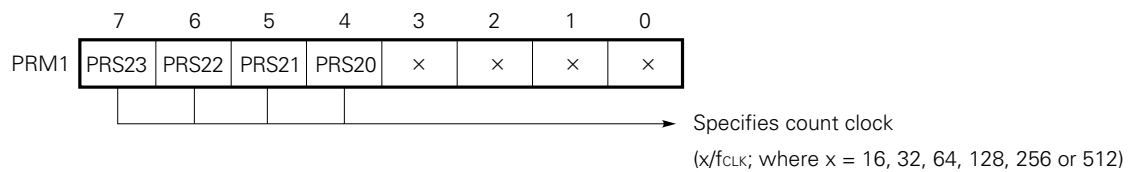


Fig. 7-112 Setting of Control Registers for PPG Output Operation

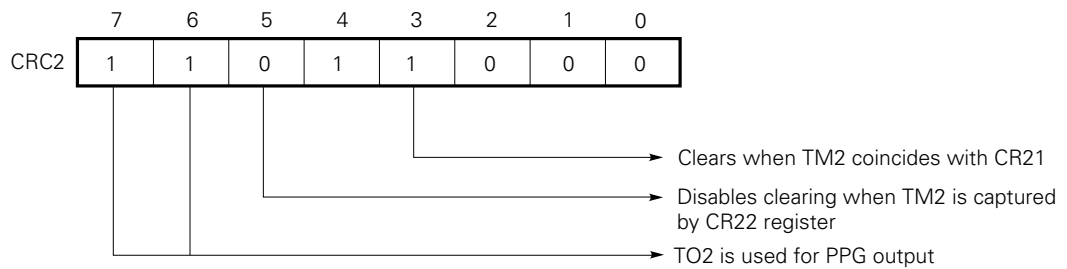
(a) Timer control register 1 (TMC1)



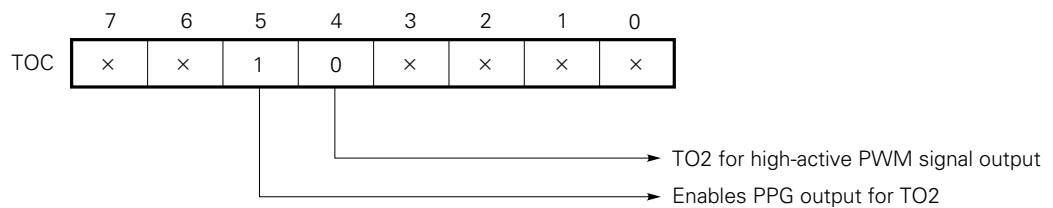
(b) Prescaler mode register 1 (PRM1)



(c) Capture/compare control register 2 (CRC2)



(d) Timer output control register (TOC)



(e) Port 3 mode control register (PMC3)

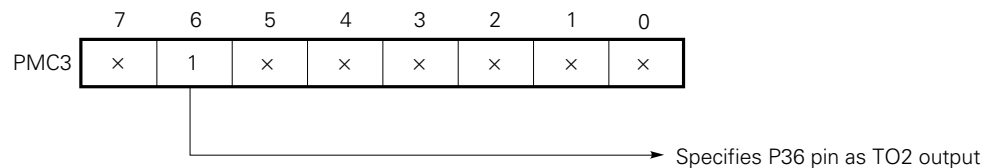


Fig. 7-113 Setting Procedure for PPG Output

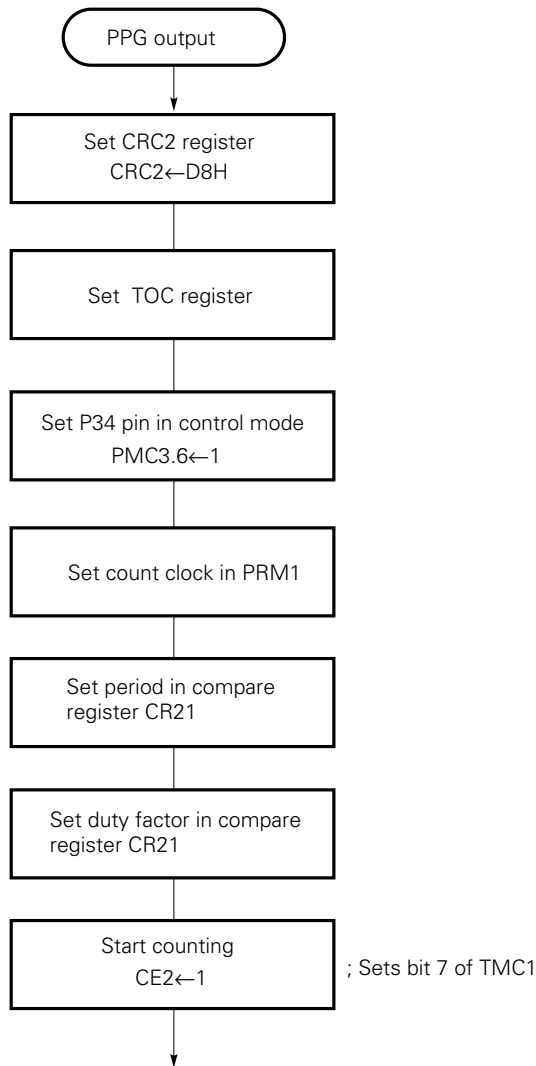
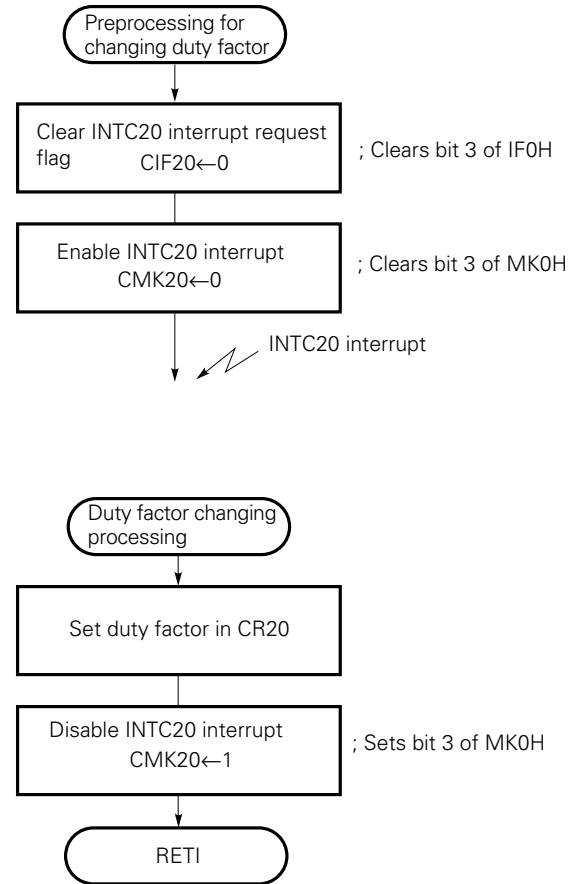


Fig. 7-114 Changing Duty Factor of PPG Output



(6) External event counter operation

When functioning as an external event counter, 8-bit timer/counter 2 counts clock pulses externally applied to the CI pin.

As shown in Fig. 7-115, the value of 8-bit timer 2 (TM2) is incremented on each valid edge specified (rising edge only in this case) of a pulse signal applied to the CI pin.

Fig. 7-115 External Event Counter Operation (When Only One Edge Is Used)

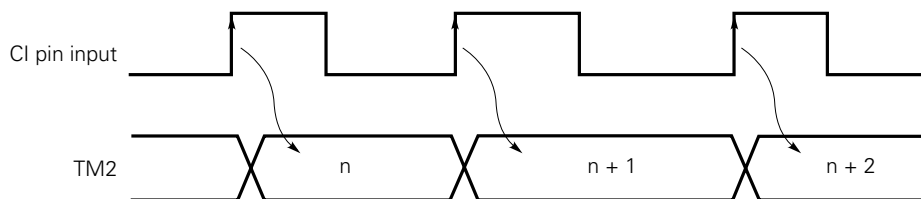


Fig. 7-116 shows the setting of control registers, and Fig. 7-117 shows the setting procedure when 8-bit timer/counter 2 functions as an external event counter.

Remark The value of TM2 is less than the number of input clock pulses by 1.

Fig. 7-116 Setting of Control Registers for External Event Counter Operation

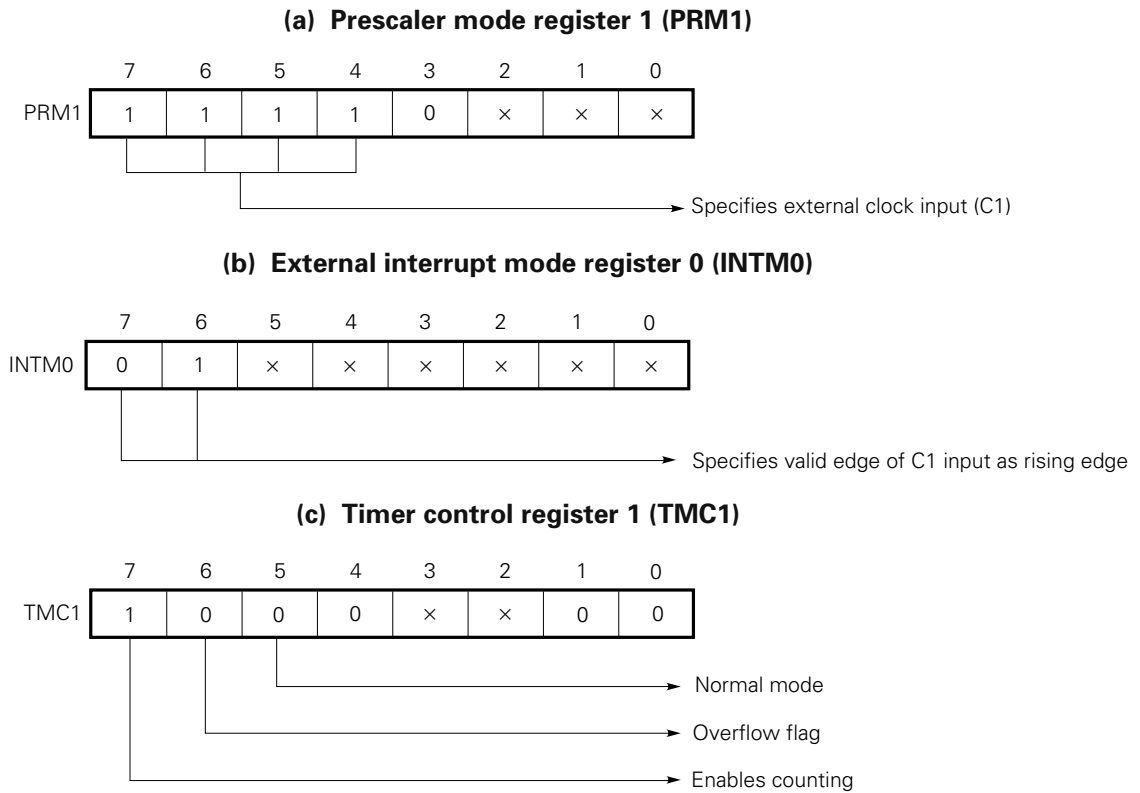
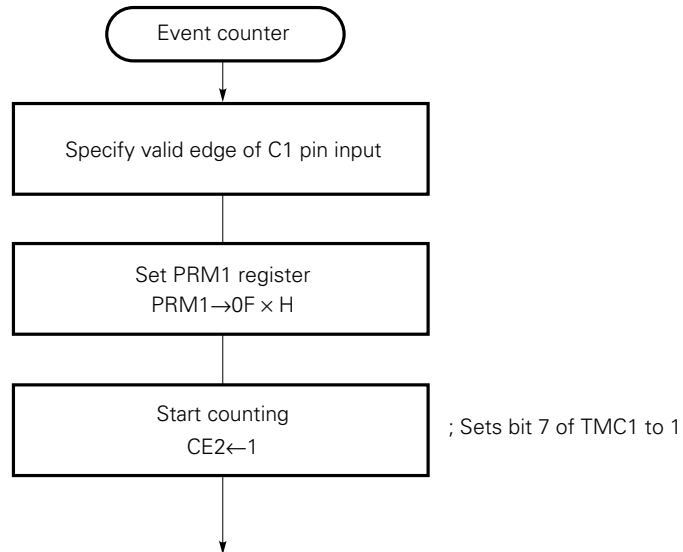


Fig. 7-117 Setting Procedure for External Event Counter Operation



(7) One-shot timer operation

When functioning as a one-shot timer, 8-bit timer/counter 2 generates only one interrupt when a specified count time has elapsed after the start of 8-bit timer 2 (TM2). (See Fig. 7-118.)

An additional one-shot timer operation can be started by clearing the OVF2 bit of timer control register 1 (TMC1).

Fig. 7-119 shows the setting of control registers. Fig. 7-120 shows the setting procedure. Fig. 7-121 shows the procedure for starting an additional one-shot operation.

Fig. 7-118 One-Shot Timer Operation

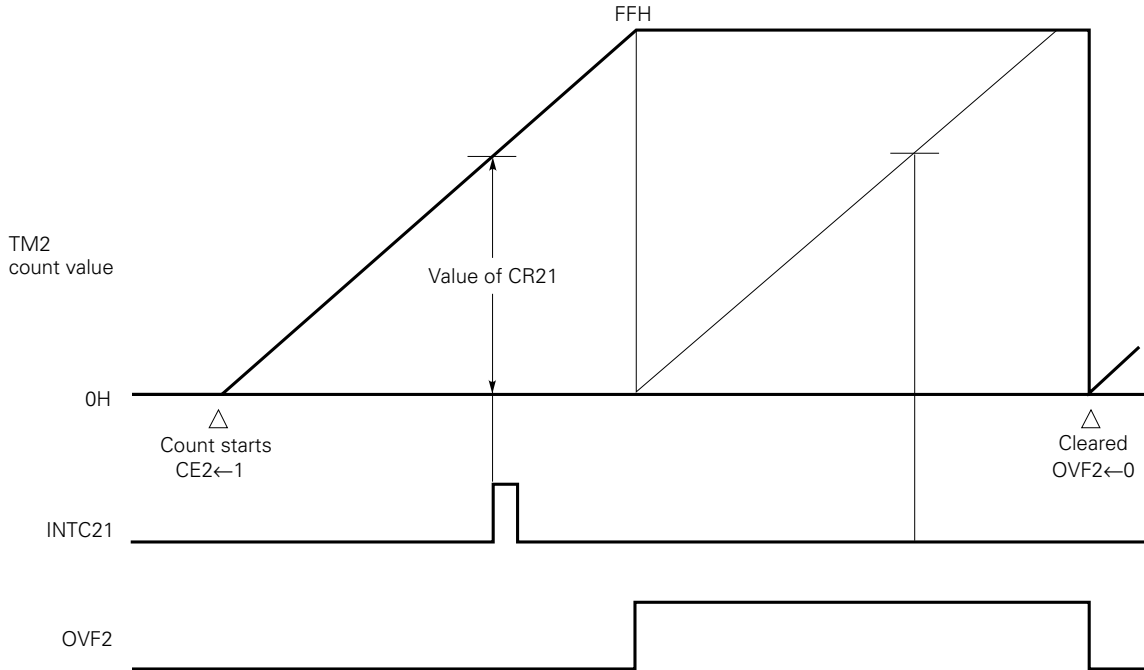
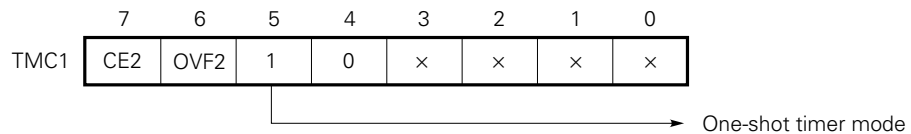
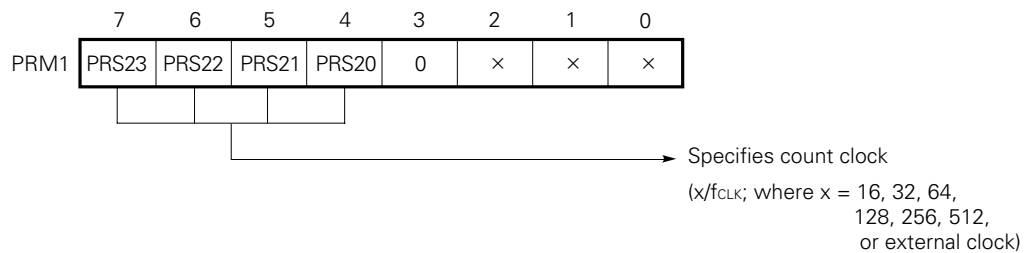


Fig. 7-119 Setting of Control Registers for One-Shot Timer Operation

(a) Timer control register 1 (TMC1)



(b) Prescaler mode register 1 (PRM1)



(c) Capture/compare control register 2 (CRC2)

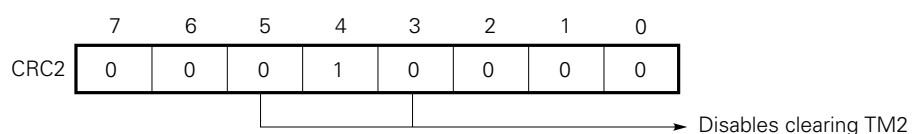


Fig. 7-120 Setting Procedure for One-Shot Timer Operation

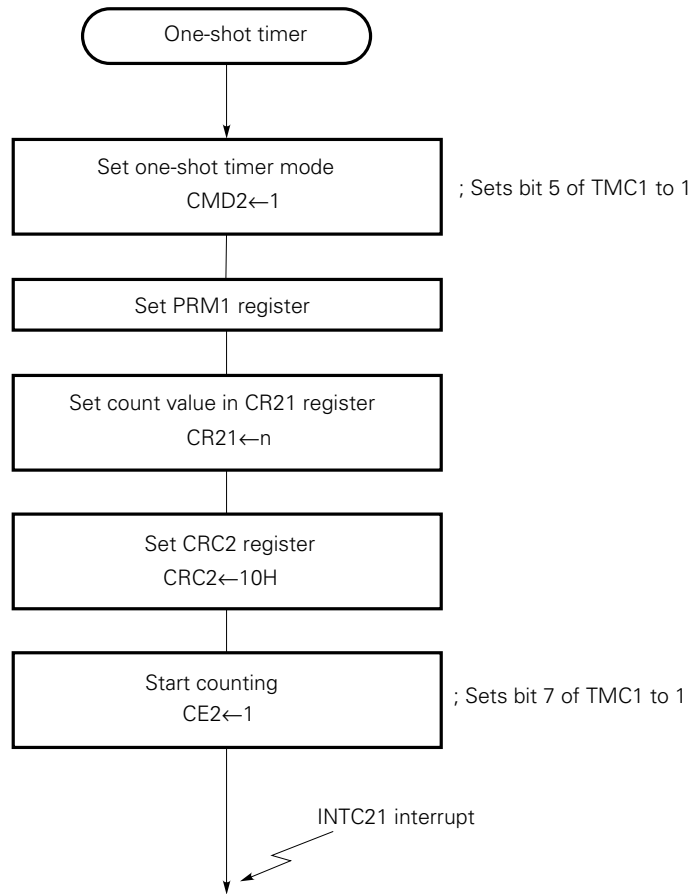
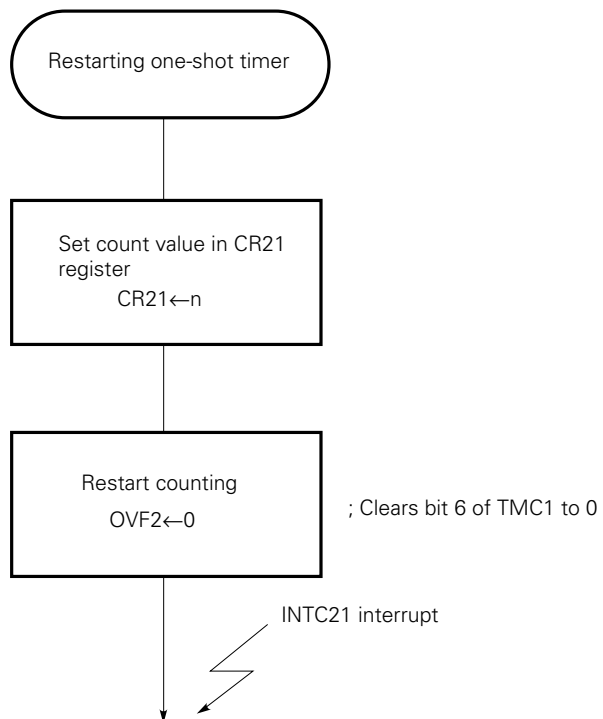


Fig. 7-121 Procedure for Starting an Additional One-Shot Timer Operation



7.4 8-BIT TIMER/COUNTER 3

7.4.1 Functions

Eight-bit timer/counter 3 can be used as an interval timer, and also as a counter for generating a clock signal used with the baud rate generator.

When operating as an interval timer, 8-bit timer/counter 3 generates an internal interrupt at specified intervals. Table 7-19 indicates the interval setting range.

Table 7-19 Intervals of 8-Bit Timer/Counter 3

Resolution	Minimum interval	Maximum interval
$8/f_{\text{CLK}}$ (1.3 μs)	$8/f_{\text{CLK}}$ (1.3 μs)	$2^8 \times 8/f_{\text{CLK}}$ (341 μs)
$16/f_{\text{CLK}}$ (2.6 μs)	$16/f_{\text{CLK}}$ (2.6 μs)	$2^8 \times 16/f_{\text{CLK}}$ (683 μs)
$32/f_{\text{CLK}}$ (5.3 μs)	$32/f_{\text{CLK}}$ (5.3 μs)	$2^8 \times 32/f_{\text{CLK}}$ (1.37 ms)
$64/f_{\text{CLK}}$ (10.7 μs)	$64/f_{\text{CLK}}$ (10.7 μs)	$2^8 \times 64/f_{\text{CLK}}$ (2.73 ms)
$128/f_{\text{CLK}}$ (21.3 μs)	$128/f_{\text{CLK}}$ (21.3 μs)	$2^8 \times 128/f_{\text{CLK}}$ (5.46 ms)
$256/f_{\text{CLK}}$ (42.7 μs)	$256/f_{\text{CLK}}$ (42.7 μs)	$2^8 \times 256/f_{\text{CLK}}$ (10.9 ms)
$512/f_{\text{CLK}}$ (85.3 μs)	$512/f_{\text{CLK}}$ (85.3 μs)	$2^8 \times 512/f_{\text{CLK}}$ (21.8 ms)

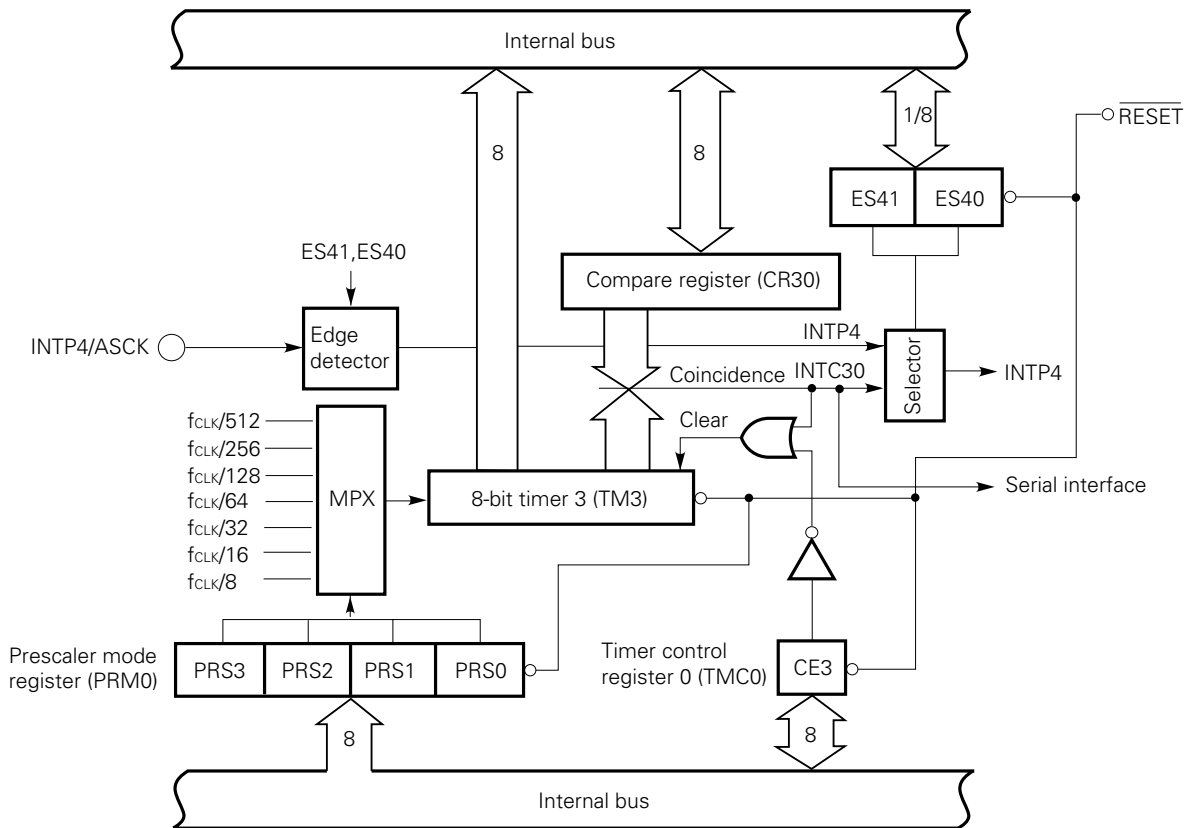
The values in parentheses are based on $f_{\text{CLK}} = 6 \text{ MHz}$.

7.4.2 Configuration

Eight-bit timer/counter 3 consists of one 8-bit timer 3 (TM3) and one 8-bit compare register (CR30).

Fig. 7-122 shows the block diagram of 8-bit timer/counter 3.

Fig. 7-122 Block Diagram of 8-Bit Timer/Counter 3



(1) 8-bit timer 3 (TM3)

TM3 is a timer for counting up with the count clock specified by the lower 4 bits of prescaler mode register 0 (PRM0).

TM3 allows only read operation using an 8-bit manipulation instruction. The count operation of TM3 can be enabled or disabled by timer control register 0 (TMC0).

When the $\overline{\text{RESET}}$ signal is applied, TM3 is cleared to 00H, and count operation stops.

(2) Compare registers (CR30)

The CR30 register is an 8-bit registers for holding a value that determines the period of interval timer operation.

When the value of the CR30 register coincides with the value of TM3, the value of TM3 is automatically cleared, and an interrupt request (INTC30) is generated.

The CR30 register allows both read and write operations using an 8-bit manipulation instruction. When the $\overline{\text{RESET}}$ signal is applied, the CR30 register becomes undefined.

(3) Prescaler

The prescaler generates count clocks from the internal system clock. From these count clocks generated by the prescaler, a count clock is selected with the selector for the timer to perform count operation.

7.4.3 8-Bit Timer/Counter 3 Control Registers

(1) Timer control register 0 (TMC0)

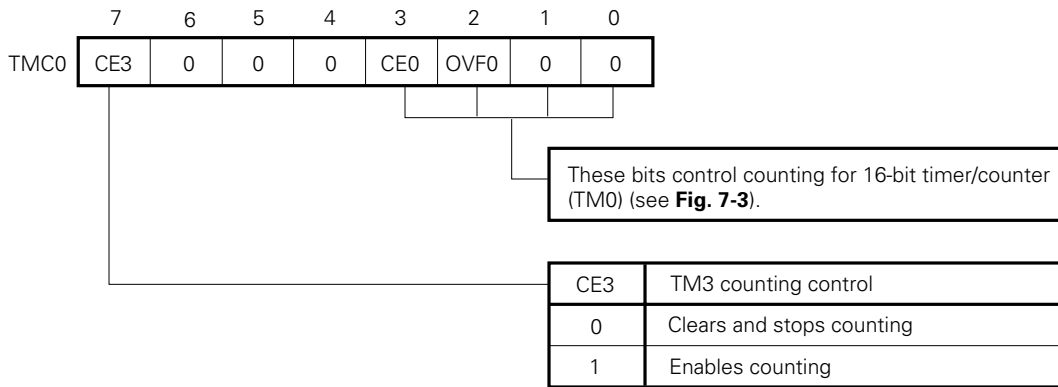
The TMC0 register is an 8-bit register for controlling the count operation of 8-bit timer 3 (TM3).

The higher 4 bits control the count operation of TM3 of 8-bit timer/counter 3. (The lower 4 bits control the count operation of TM0 of the 16-bit timer/counter.)

The TMC0 register allows both read and write operations using an 8-bit manipulation instruction. Fig. 7-123 shows the format of the TMC0 register.

When the $\overline{\text{RESET}}$ signal is applied, the TMC0 register is cleared to 00H.

Fig. 7-123 Format of Timer Control Register 0 (TMC0)



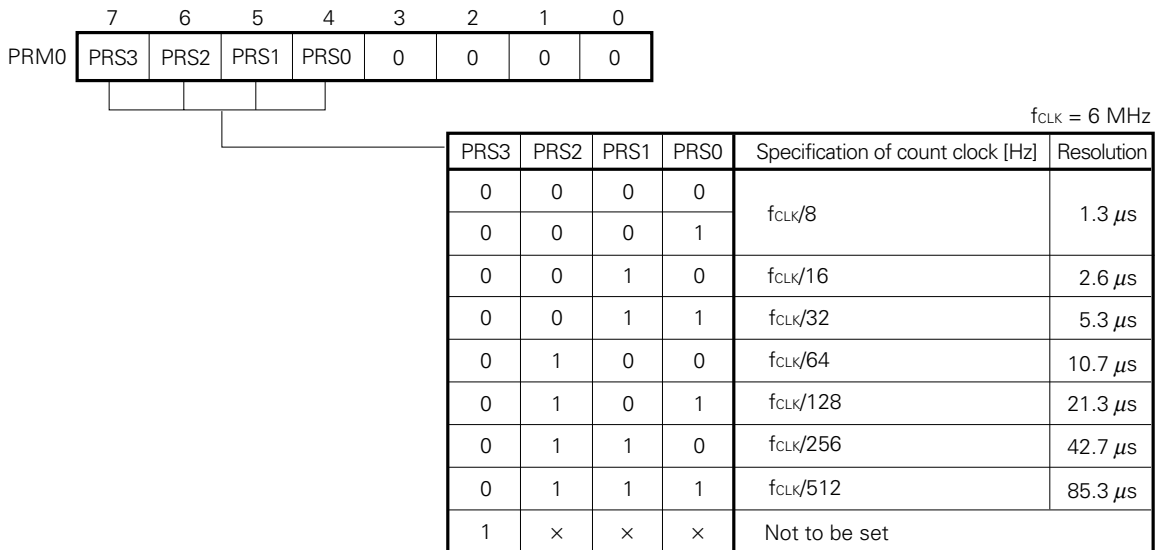
(2) Prescaler mode register 0 (PRM0)

The PRM0 register is an 8-bit register used to specify a count clock for 8-bit timer 3 (TM3).

The PRM0 register allows only write operation using an 8-bit manipulation instruction. Fig. 7-124 shows the format of the PRM0 register.

When the $\overline{\text{RESET}}$ signal is applied, the PRM0 register is cleared to 00H.

Fig. 7-124 Format of Prescaler Mode Register 0 (PRM0)



Remark f_{CLK} : System clock frequency
 x: 1 or 2

7.4.4 Operation of 8-Bit Timer 3 (TM3)

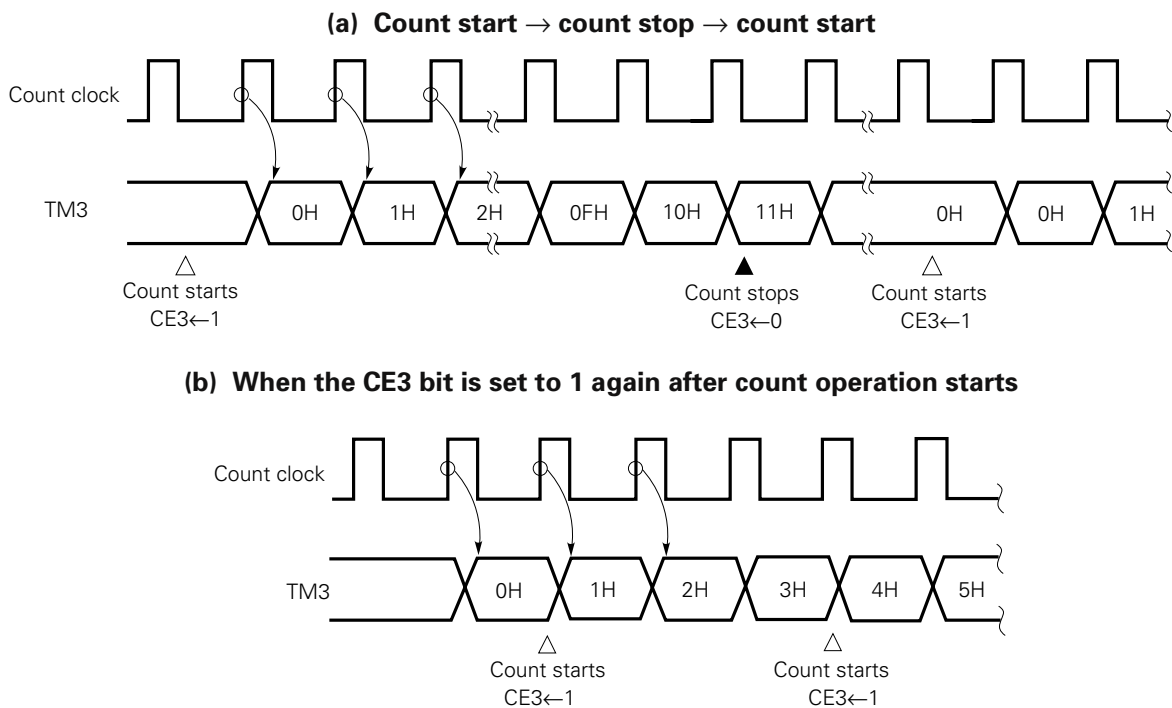
(1) Basic operation

Eight-bit timer/counter 3 performs count operation by counting up with the count clock specified by the higher 4 bits of prescaler mode register 0 (PRM0).

When the $\overline{\text{RESET}}$ signal is applied, TM3 is cleared to 00H, and count operation stops.

Bit 7 (CE3) of timer control register 0 (TMC0) is used to enable/disable count operation. (The higher 4 bits of the TMC0 register are used to control the operation of 8-bit timer/counter 3.) When the CE3 bit is set to 1 by software, TM3 is cleared to 00H by the first count clock pulse, then count-up operation starts. When the CE3 bit is reset to 0, TM3 is cleared to 00H by the next count clock pulse, then coincide with signal generation stops. If the CE3 bit is set to 1 when the CE3 bit is already set to 1, TM3 is not cleared, but continues count operation.

Fig. 7-125 Basic Operation of 8-Bit Timer 3 (TM3)

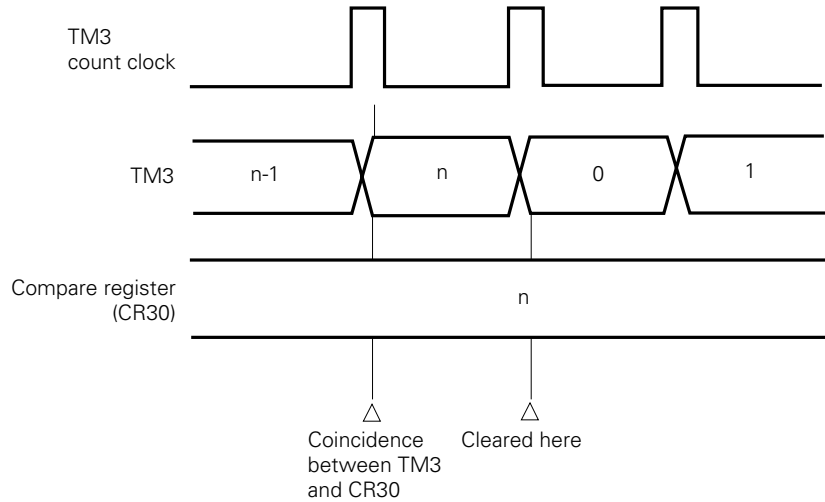


(2) Clear operation

After a coincidence with the CR30 compare register, 8-bit timer 3 (TM3) can be automatically cleared.

If a TM3 clear cause occurs, TM3 is cleared to 00H by the next count clock pulse. This means that even if a TM3 clear cause occurs, TM3 holds the value existing at that time until the next count clock pulse is applied.

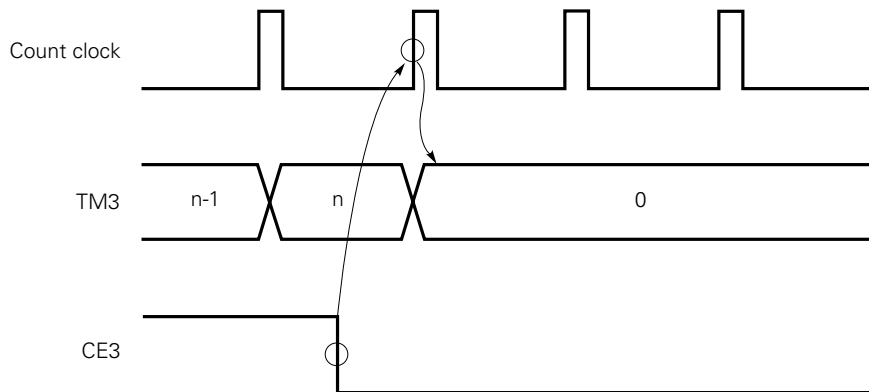
Fig. 7-126 TM3 Cleared by a Coincidence with Compare Register (CR30)



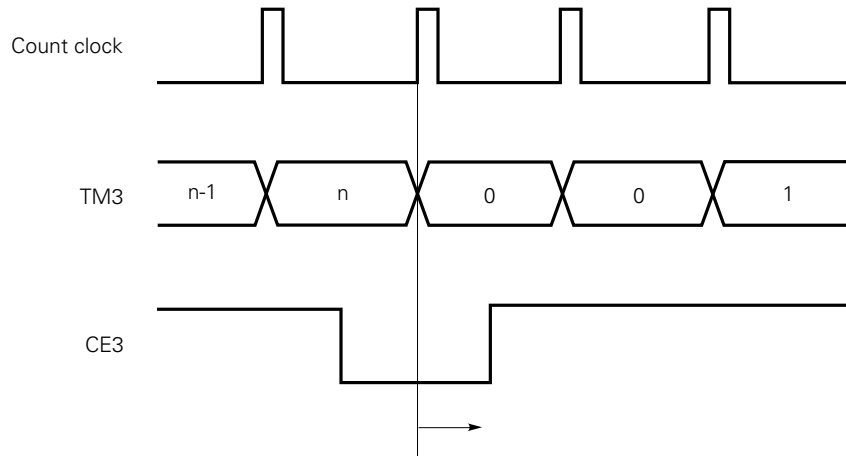
TM3 can also be cleared by software when the CE3 bit of the timer control register (TMC0) is reset to 0. Similarly, clear operation is performed by the count clock pulse following the resetting of CE3 bit to 0. If the CE3 bit is set to 1 before TM3 is reset to 0 by the resetting of the CE3 bit to 0 (that is, before the first count clock pulse is applied after the CE3 bit is reset to 0), two operations are simultaneously performed: one operation is an operation to clear TM3 to 0, and the other operation is a count operation starting with the counting of 0.

Fig. 7-127 Clear Operation When the CE3 Bit Is Reset to 0

(a) Basic operation

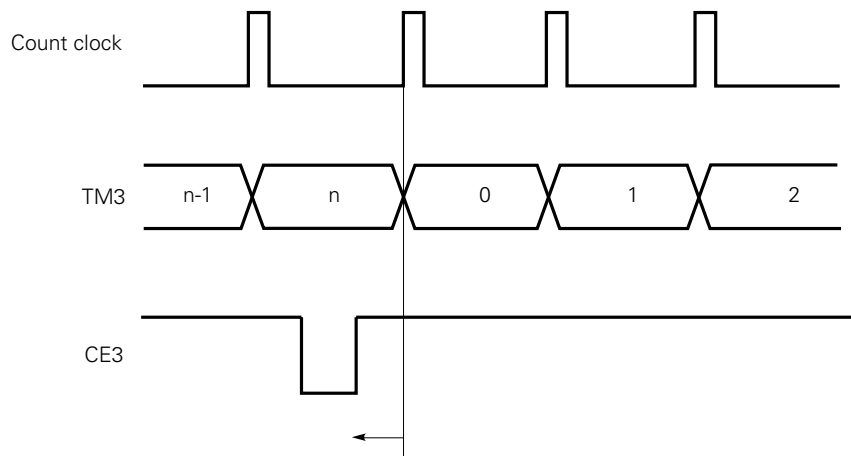


(b) Restart after 0 is set in TM3 cleared



When the CE3 bit is set to 1 after this count clock, counting starts from 0 on the count clock input after the CE3 bit has been set.

(c) Restart before 0 is set in TM3 cleared



When the CE3 bit is set to 1 before this count clock, Clearing TM3 by $CE3 \leftarrow 0$ and counting by $CE3 \leftarrow 1$ are performed simultaneously.

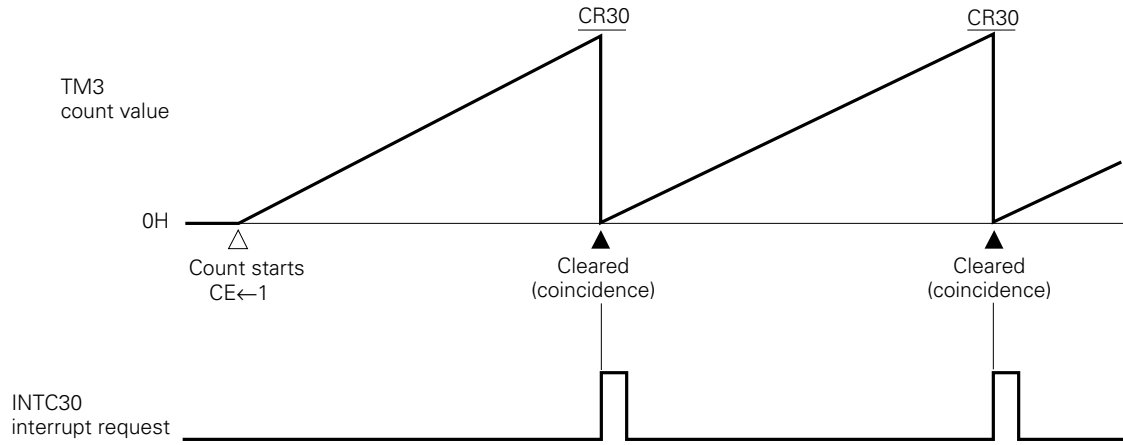
7.4.5 Compare Register Operation

Eight-bit timer/counter 3 performs a compare operation to compare the value set in the compare register with a timer count value.

When the value set in the compare register (CR30) coincides with a count value of 8-bit timer 3 (TM3), the interrupt request (INTC30) is generated.

After the value of the CR30 register coincides with a count value of TM3, the value of TM3 is automatically cleared. Thus, 8-bit timer/counter 3 can operate as an interval timer for repeatedly counting up to the value set in the CR30 register.

Fig. 7-128 Compare Operation



7.4.6 Sample Applications

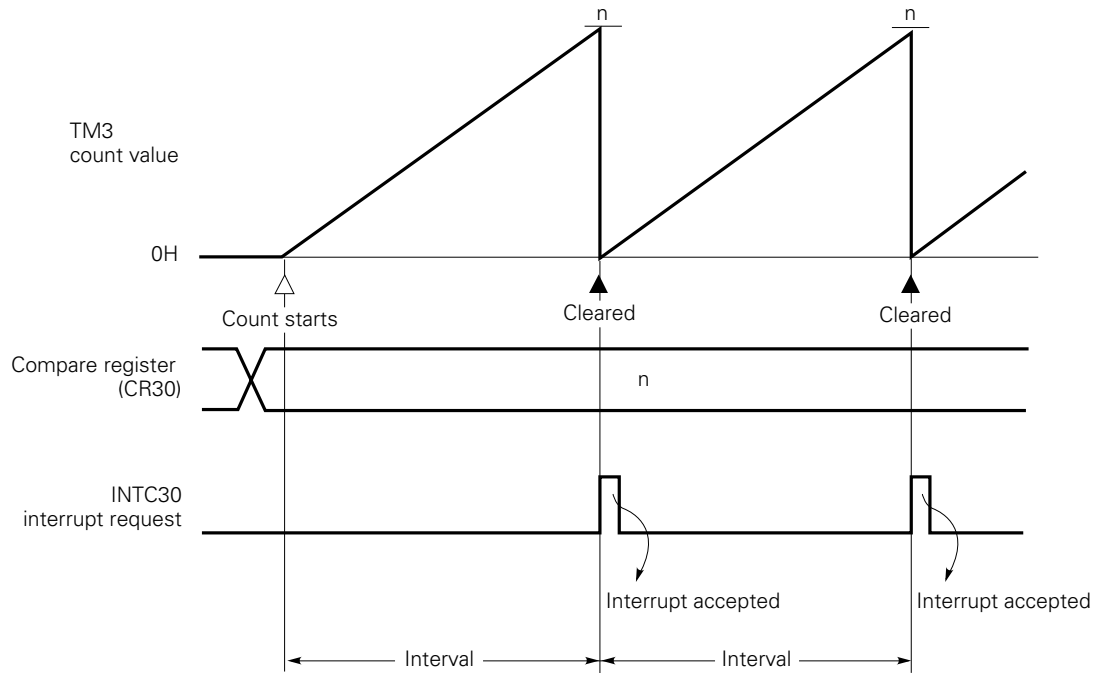
(1) Interval timer operation

Eight-bit timer/counter 3 can be used as an interval timer that generates an interrupt at intervals of a count time specified beforehand. Eight-bit timer/counter 3 can also be used for baud rate generation.

This interval timer has a resolution from 1.3 μs to 85.3 μs, and can count up to 341 μs and 21.8 ms, respectively (at internal system clock $f_{CLK} = 6$ MHz).

Fig. 7-129 shows the timing of interval timer operation. Fig. 7-130 shows the setting of control registers for interval timer operation. Fig. 7-131 shows the setting procedure.

Fig. 7-129 Timing of Interval Timer Operation



Remark Interval = $(n + 1) \times x / f_{CLK}$, $0 \leq n \leq FFH$
 $x = 8, 16, 32, 64, 128, 256, 512$

Fig. 7-130 Setting of Control Registers for Interval Timer Operation

★

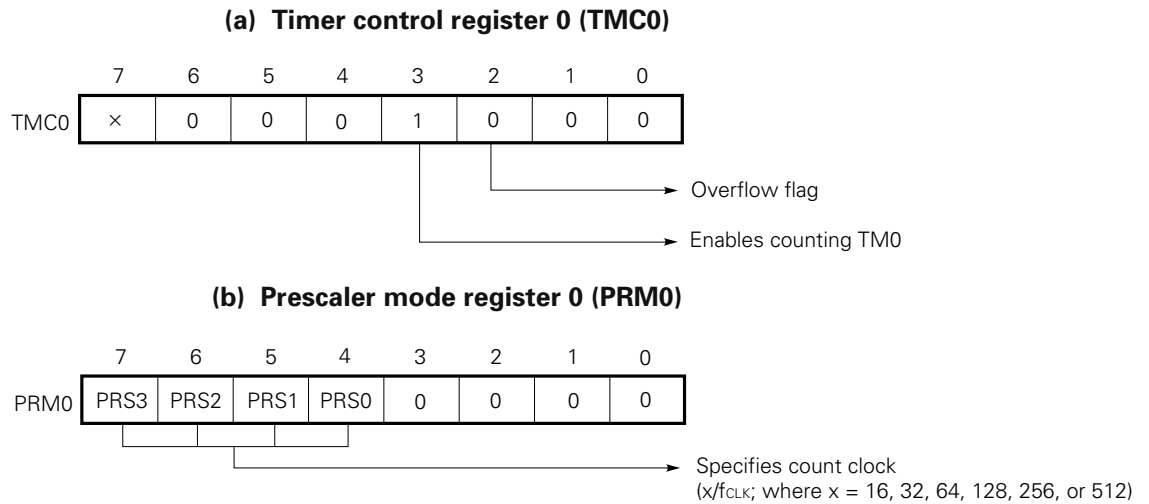
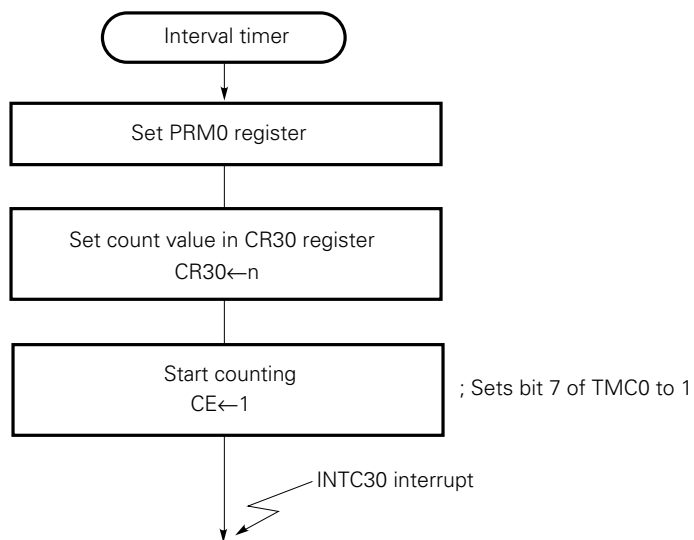


Fig. 7-131 Setting Procedure for Interval Timer Operation



7.5 NOTES

7.5.1 Common Notes on All Timers/Counters

(1) When the registers listed below are rewritten while a counter is operating (with the CEm bit of register TMCn is set), the counter can malfunction. The cause of such a malfunction is that when a hardware function change made by the rewriting of a register conflicts with a state change of the function before the rewriting, which change is to have priority is undefined.

Before rewriting these registers, be sure to stop counter operation for safety.

- Prescaler mode register (PRMn)
- Capture/compare control register (CRCn)
- Timer output control register (TOC)
- CMD2 bit of timer control register 1 (TMC1)

- (2) The OVFm flag for holding an overflow from a timer/counter is contained in register TMCn used to control the operation of the timer/counter. When a read/modify/write instruction (such as AND TMCn,#7FH) is executed, for example, the OVFm flag may be cleared. (Even if the OVFm flag is 0 when the OVFm flag is read by the CPU, the OVFm flag may already be set to 1 by a timer/counter overflow when the OVFm flag is rewritten to by the CPU. However, the OVFm flag was 0 when being read by the CPU, so that the OVFm flag is cleared to 0.)

To prevent the OVFm flag of a timer/counter controlled by register TMCn from being cleared, use the method below.

1. Read the TMm register of the timer/counter using the OVFm flag.
2. After a timer/counter operation, read the TMm register again.
3. Make a comparison between two read values. Set the OVFm flag if the value read later is smaller than the value read first.

When this method is used, the OVFm flag must not be manipulated. In addition, the time between the first TMm read operation and the next TMm read operation must be shorter than the full-count time of TMm.

Example: To prevent the OVF1 flag of timer/counter 1 from being cleared

```

MOV  A, TM1
MOV  TMC1, #xxx01000B ; xxx depends on the manipulation of timer/counter 2.
CMP  A, TM1           ; Checks the timer value.
BL   $NEXT
BE   $NEXT
MOV  TMC1, #xxx011000B ; Sets OVF1.

```

NEXT:

- (3) If the value of a compare register coincides with the value of a timer register when an instruction for stopping timer operation is executed, the count operation of the timer stops, but an interrupt request is generated.

To prevent an interrupt request from being generated when timer operation is stopped, mask the interrupt by using the mask register before stopping the timer.

Program that can generate an interrupt request

```

:
:
Example: MOV  TMC1, #6CH ← An interrupt request
             AND  MK0H, #0F6H is generated from the
             :           timer/counter be-
             :           tween these instruc-
             :           tions.

```

Program that generates no interrupt request

```

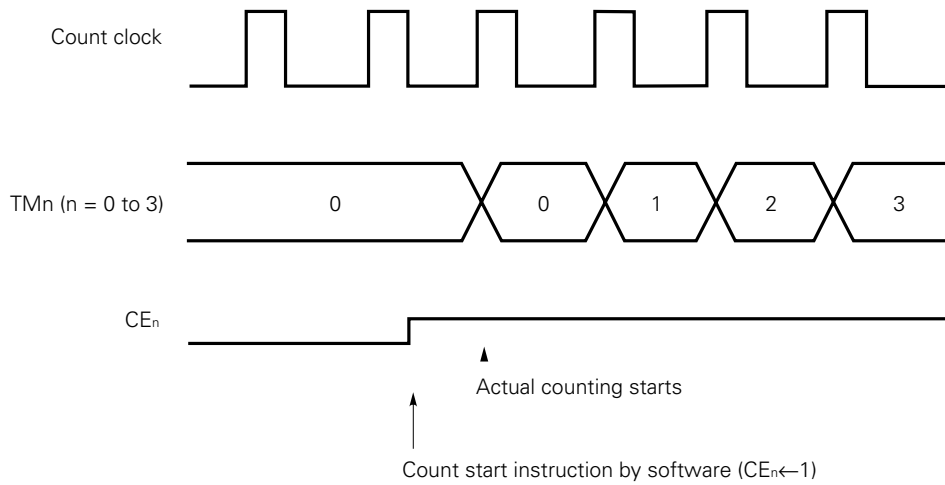
:
:
AND  MK0H, #0F6H ← An interrupt from
MOV  TMC1, #6CH   timer/counter 2 is disa-
AND  IF0H, #0F6H ← The interrupt request
:               flag for timer/counter
:               2 is cleared.

```

- (4) After an operation for starting a timer/counter is performed ($CE_n \leftarrow 1$; $n = 0$ to 3), a maximum of 1 count clock pulse is required until the timer/counter actually starts. (See **Fig. 7-132**.)

When a timer/counter is used as an interval timer, for example, the first interval is longer by a maximum of 1 clock pulse. After the first interval, the specified interval occurs.

Fig. 7-132 Count Start Operation



- (5) Even when an instruction is executed to stop a timer ($CEn \leftarrow 0$), the value of TMn is not cleared to 0 immediately. Instead, the value of TMn is cleared to 0 by the count clock pulse occurring after an instruction is executed ($CEn \leftarrow 0$) to stop the timer.

Even if a timer is started ($CEn \leftarrow 1$) before the timer is cleared to 0 immediately after the timer is stopped, the timer counts up starting with 0.

Fig. 7-133 Count Operation Stop

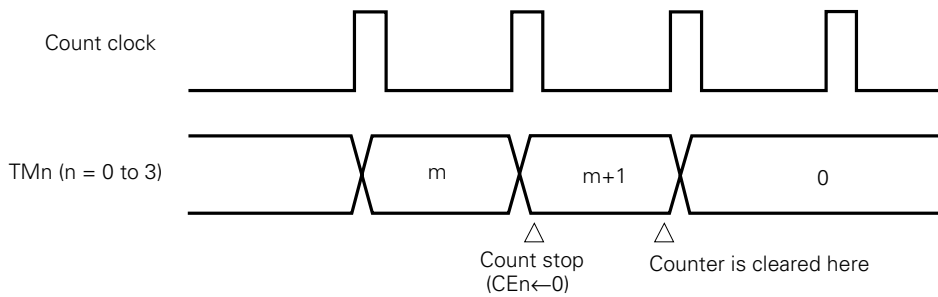
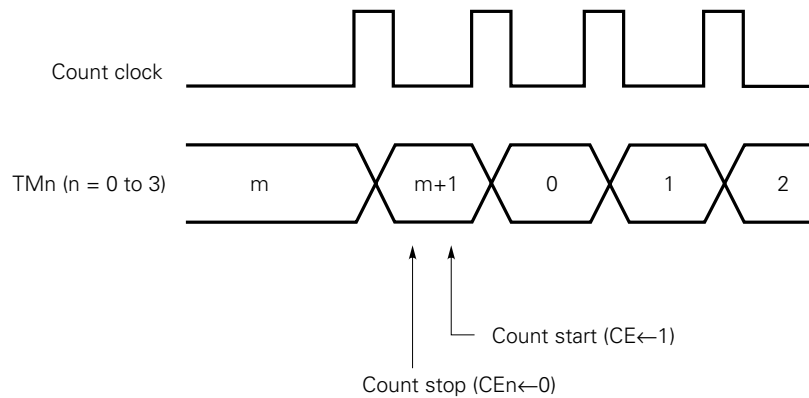


Fig. 7-134 Timing of Count Operation Stop and Restart



- (6) When a register associated with a timer/counter is accessed, wait states as many as the maximum number of clock pulses^{Note} indicated below are automatically inserted.

Note One wait state: $1/f_{CLK}$

Table 7-20 *Maximum Number of Wait States Inserted When Registers Associated with Timers/Counters Are Accessed*

Register	Number of wait states inserted	
	For read	For write
TMCn	0	1
PRMn	—	1
CRCn	—	1
TOC	—	1
CRnm	1	1
TM0	7	—
TM1	15	—
TM2	15	—
TM3	7	—

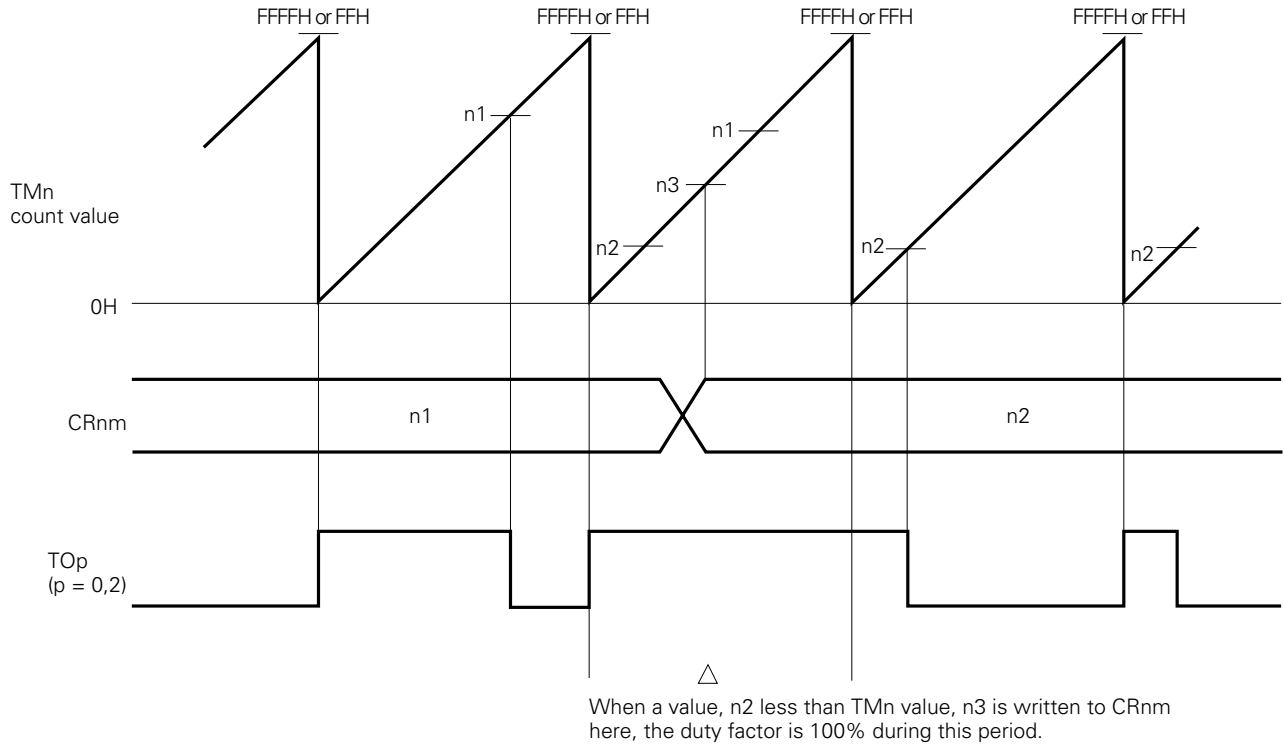
- (7) While an instruction for writing to compare register CRnm ($n = 0$ to 3 , $m = 0, 1$) is being executed, no coincidence is detected between the CRnm register being written to and TMn ($n = 0$ to 3). For example, if the value of the CRnm register remains unchanged after the CRnm is written to, no interrupt request is generated, and timer output (TON: $n = 0$ to 3) does not change even when the value of TMn coincides with the value of the CRnm register.

While a timer/counter is performing count operation, CRnm register write operation must be performed when the value of TMn does not coincide with the value of the CRnm register before or after CRnm register write operation. (CRnm register write operation must be performed, for example, immediately after an interrupt request is generated by a coincidence between TMn and CRnm.)

- (8) A coincidence between TMn and CRnm is detected only when TMn is incremented. This means that no interrupt request is generated, and no timer output change is made even if the same value as of TMn is written to CRnm.

- (9) When PWM is used, a PWM signal with a 100% duty factor is output if a value less than the value of TMn (n = 0, 2) is set in compare register CRnm (n = 0, 2, m = 0, 1). CRnm rewrite operation must be performed using an interrupt generated by a coincidence between TMn and CRnm to be rewritten.

Fig. 7-135 Example of PWM Output Signal with a 100% Duty Factor

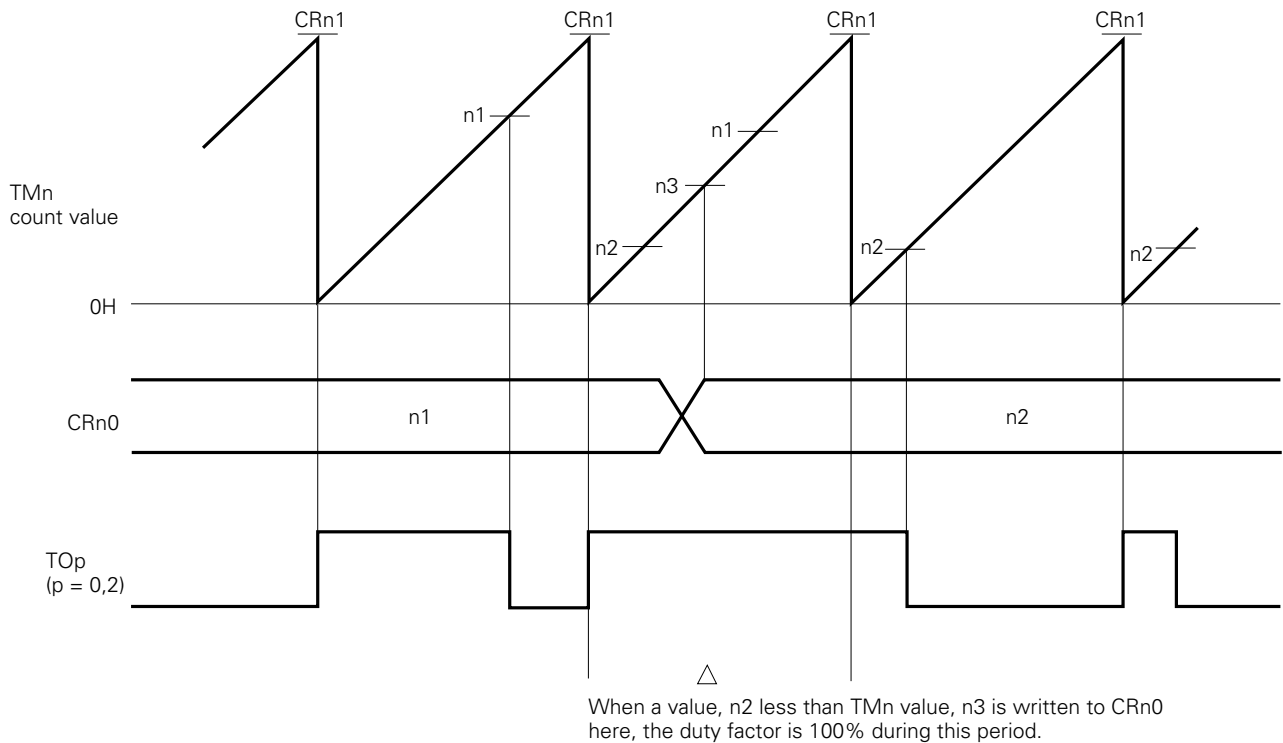


Remark ALVp = 0

(10) Notes on compare register rewrite operation when PPG output is used

- (a) If a value less than the value of TM_n is written into compare register CR_n0 ($n = 0, 2$) before the value of the CR_n0 register coincides with the value of TM_n ($n = 0, 2$), a PPG signal with a 100% duty factor is output in that period. CR_n0 rewrite operation must be performed using an interrupt generated by a coincidence between TM_n and CR_n0 .

Fig. 7-136 Example of PPG Output Signal with a 100% Duty Factor

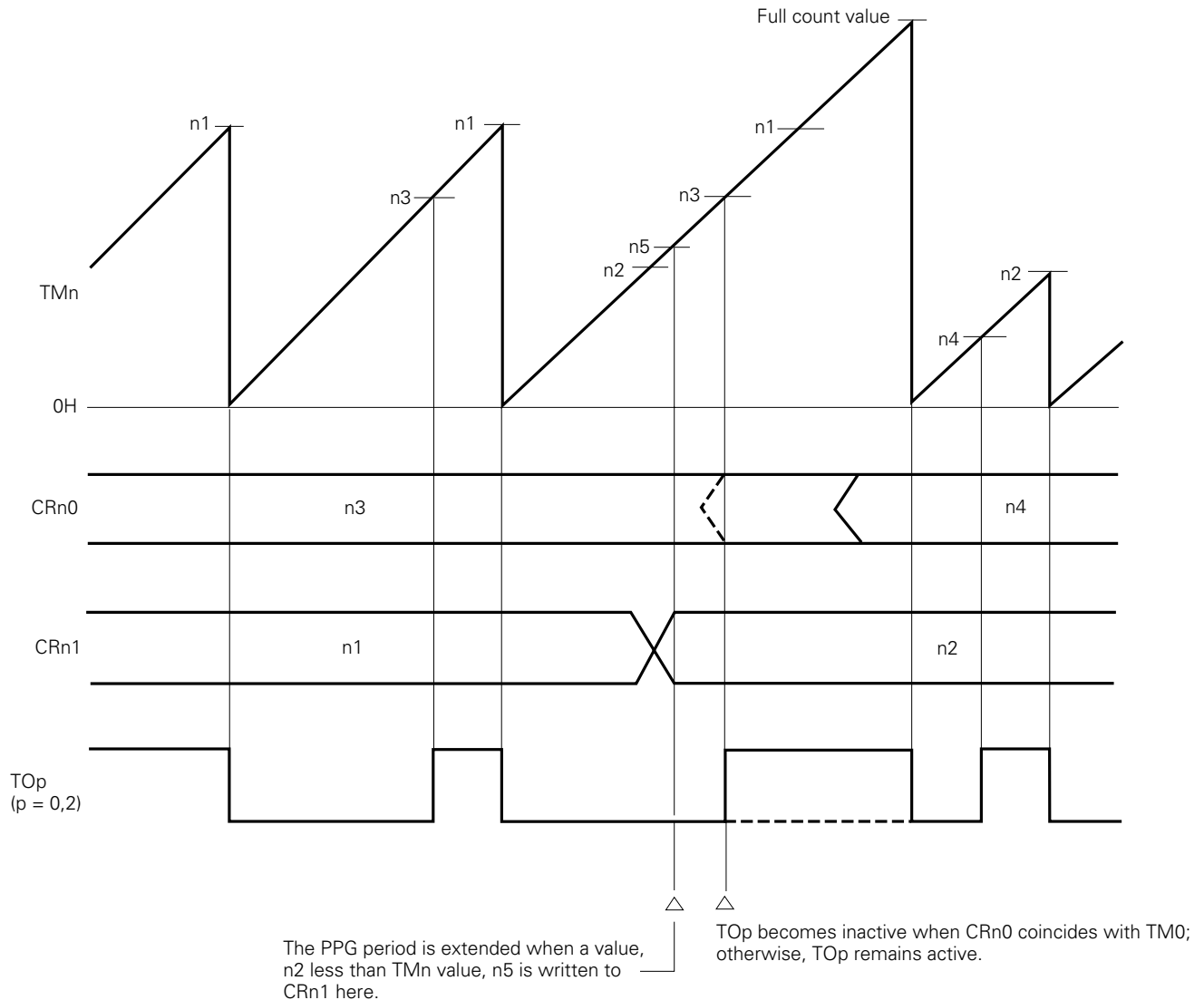


Remark $ALV_p = 0$

- (b) If the current value of the CR_n1 compare register is decreased below the value of TM_n , the PPG period becomes as long as the full-count time of TM_2 . At this time, if CR_n1 is rewritten after the value of the CR_n0 compare register coincides with the value of TM_n , the inactive level is output until TM_n overflows to 0, then normal PPG output is resumed. If CR_n1 is rewritten before the value of CR_n0 coincides with the value of TM_n , the active level is output until the value of CR_n0 coincides with the value of TM_n . When the value of CR_n0 coincides with the value of TM_n before TM_n overflows to 0, the inactive level is output at that time. When TM_n overflows to 0, the active level is output, and normal PPG output is resumed.

CR_n1 rewrite operation must be performed using an interrupt generated by a coincidence between TM_n and CR_n1 .

Fig. 7-137 Example of PPG Output Period Made Longer



Remark ALVp = 1

(c) If the PPG period is too short for interrupt acceptance, the measures described in (a) and (b) above do not lead to solution. Consider other measures (such as masking all interrupts and polling interrupt request flags by software).

- ★ (11) In PWM output, the actual pulse width is longer than a value obtained with the approximate expression by two clock pulses of f_{CLK} for the active level, and is shorter than such an approximate value by two clock pulses of f_{CLK} for the inactive level. Take this point into consideration when high-precision output is required or a high-speed count clock is used. For details, see **Section 7.1.7** and **Section 7.3.9**.
- (12) If timer output is disabled (ENTOn = 0: n = 0, 1, or 2, 3), the output level on the TOn (n = 0, 1, or 2, 3) is the inverted value of the value set in ALVn (n = 0, 1, or 2, 3). Accordingly, note that if timer output is disabled when the PWM output function or PPG output function is selected, the active level is output.
- ★ (13) In PPG output, the actual pulse width is longer than a value obtained with the approximate expression by two clock pulses of f_{CLK} for the active level, and is shorter than such an approximate value by two clock pulses of f_{CLK} for the inactive level. Take this point into consideration when high-precision output is required, the PPG pulse period is short, or a high-speed count clock is used.

For details, see **Section 7.1.8** and **Section 7.3.10**.

(14) With an in-circuit emulator, digital noise cannot be removed correctly. When a timer/counter is used together with edge detection function, note the point below.

(a) When IE-78210-R is used

Operations are performed on an erroneously detected edge.

(b) When other in-circuit emulators are used

- Timer/counter capture operation and clear operation

An erroneously detected edge has no effect. Accordingly, even if an interrupt is generated by an erroneously detected edge, the capture value is not updated. Particularly, note that the value of CR22 after being read is undefined.

- Timer/counter compare operation

If a mode is set which performs a clear operation after a capture operation, an erroneously detected edge changes the timing of coincidence-based interrupt generation. As the result, an interrupt is repeatedly generated at a time when the value of the timer/counter does not coincide with the value of the compare register. Normal coincidence-based interrupt generation is resumed when a normal edge is applied or the timer/counter is stopped. Timer output is not affected by an erroneously detected edge, but is performed with the normal timing.

For details of erroneous edge detection, see **Section 11.4**.

When using an in-circuit emulator, see also **Section 7.5.4**.

7.5.2 Notes on 16-Bit Timer/Counter

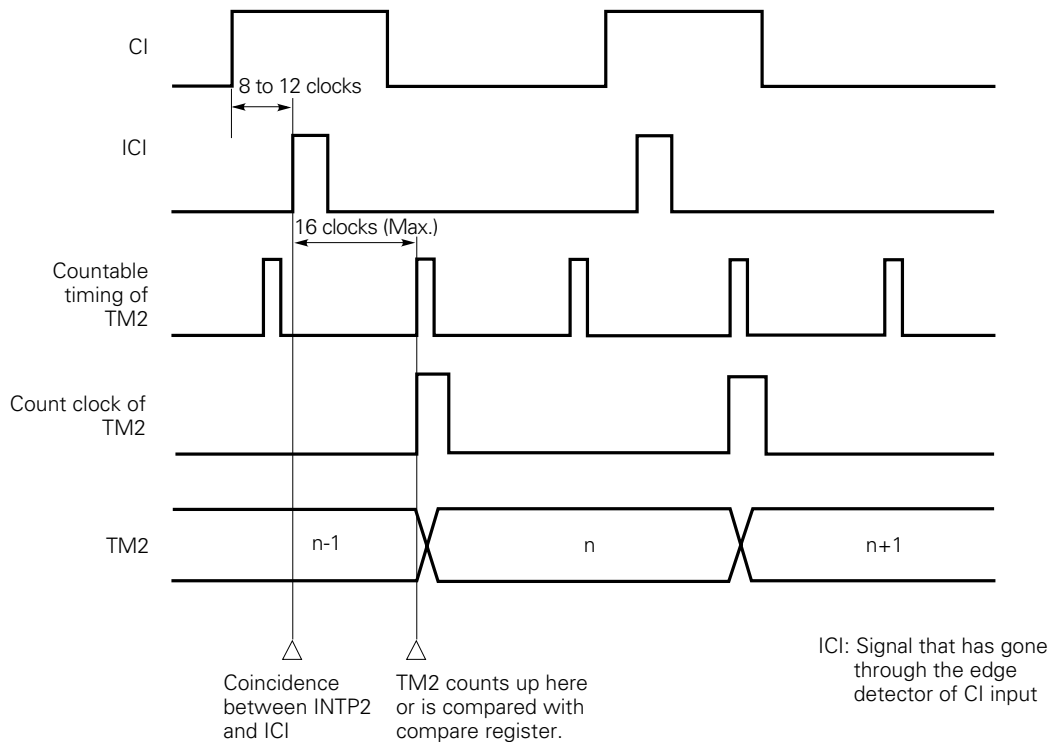


Before compare operation can be performed with the 16-bit timer/counter, an appropriate value must be written into a compare register to be used.

7.5.3 Notes on 8-Bit Timer/Counter 2

- (1) When 8-bit timer/counter 2 is used as an external event counter, the increment of TM2 lags the input of a valid edge to the CI pin by a maximum of 28 system clock pulses ($4.67 \mu\text{s}$: $f_{\text{CLK}} = 6 \text{ MHz}$). This means that TM2 may not be incremented when read immediately after an edge is detected. In addition, the generation of an interrupt request by a coincidence with a compare register (CR20, CR21) lags the input of an edge. Take this point into consideration when fine timing control is required.

Fig. 7-138 Interrupt Request Generation Using External Event Counter



- (2) When 8-bit timer/counter 2 is used as an external event counter, TM2 alone cannot distinguish between the state where no valid edge is applied and the state where only one valid edge has been applied. (See Fig. 7-139.) In either case, the value of TM2 is 0. When the states need to be distinguished from each other, use the INTTP2 interrupt request flag. (The same pin is used as the INTTP2 pin as well as the CI pin, so that the functions can be used at the same time.) Fig. 7-140 shows an example of distinction.

Fig. 7-139 Example Where Input of No Valid Edge Cannot Be Distinguished from Input of Only One Valid Edge with External Event Counter

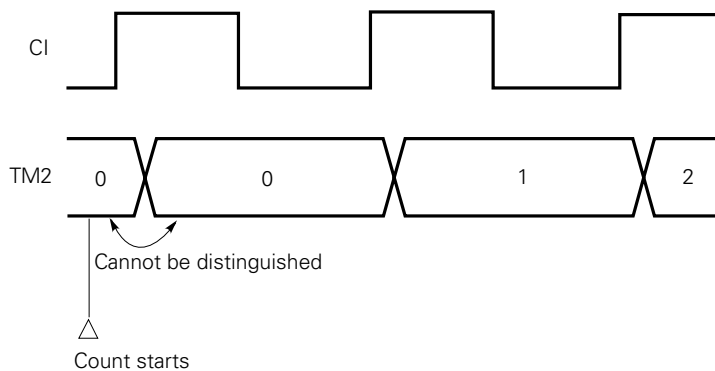
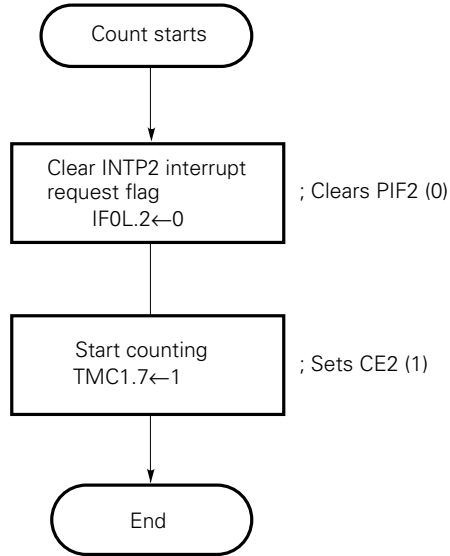
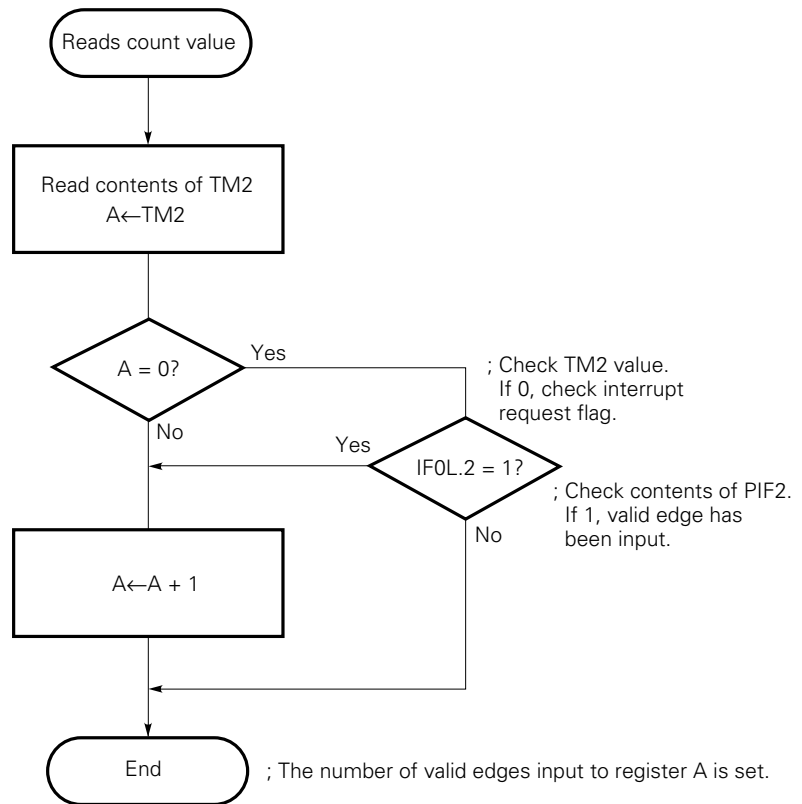


Fig. 7-140 How to Distinguish Input of No Valid Edge from Input of Only One Valid Edge with External Event Counter

(a) Count start processing



(b) Count value read processing



- (3) With an in-circuit emulator, digital noise cannot be removed correctly. When the timer/counter is used together with edge detection function, note the point below.
- When IE-78210-R is used
All functions are performed on an erroneously detected edge.
 - When other in-circuit emulators are used
When 8-bit timer/counter 2 is used as an external event counter, an erroneously detected edge changes the timing of coincidence-based interrupt generation. As the result, an interrupt is repeatedly generated at a time when the value of the timer/counter does not coincide with the value of the compare register.
Normal coincidence-based interrupt generation is resumed when the timer/counter is stopped.
Timer output is not affected by an erroneously detected edge, but is performed with the normal timing.
For details of erroneous edge detection, see **Section 11.4**.
When using an in-circuit emulator, see also **Section 7.5.4**.
- (4) The value of the CR22 register, after being read, becomes undefined. A captured value can be used more than once by saving the captured value to a register or memory.

7.5.4 Notes on Using In-Circuit Emulators

When an in-circuit emulator is used, noise removal operation for INTP0, INTP1, INTP2/CI, and INTP3 may not be performed normally, thus resulting in noise detected erroneously as an edge. For details of erroneous edge detection, see **Section 11.4**. How a timer/counter operates with an erroneously detected edge is described below.

(1) When IE-78210-R is used

All timer/counter-related operations are performed on erroneously detected edges in the same way as on normal edges.

(2) When other in-circuit emulators are used

(a) Capture operation

Capture operation is not performed on an erroneously detected edge. However, an interrupt is generated on an erroneously detected edge. The value of a capture register read during interrupt handling performed on an erroneously detected edge is as follows:

- For CR02 and CR11
Value captured on the immediately preceding normal edge
- For CR22
Undefined value

(b) Clear operation after capture operation (with only 8-bit timer/counter 1 and 8-bit timer/counter 2)

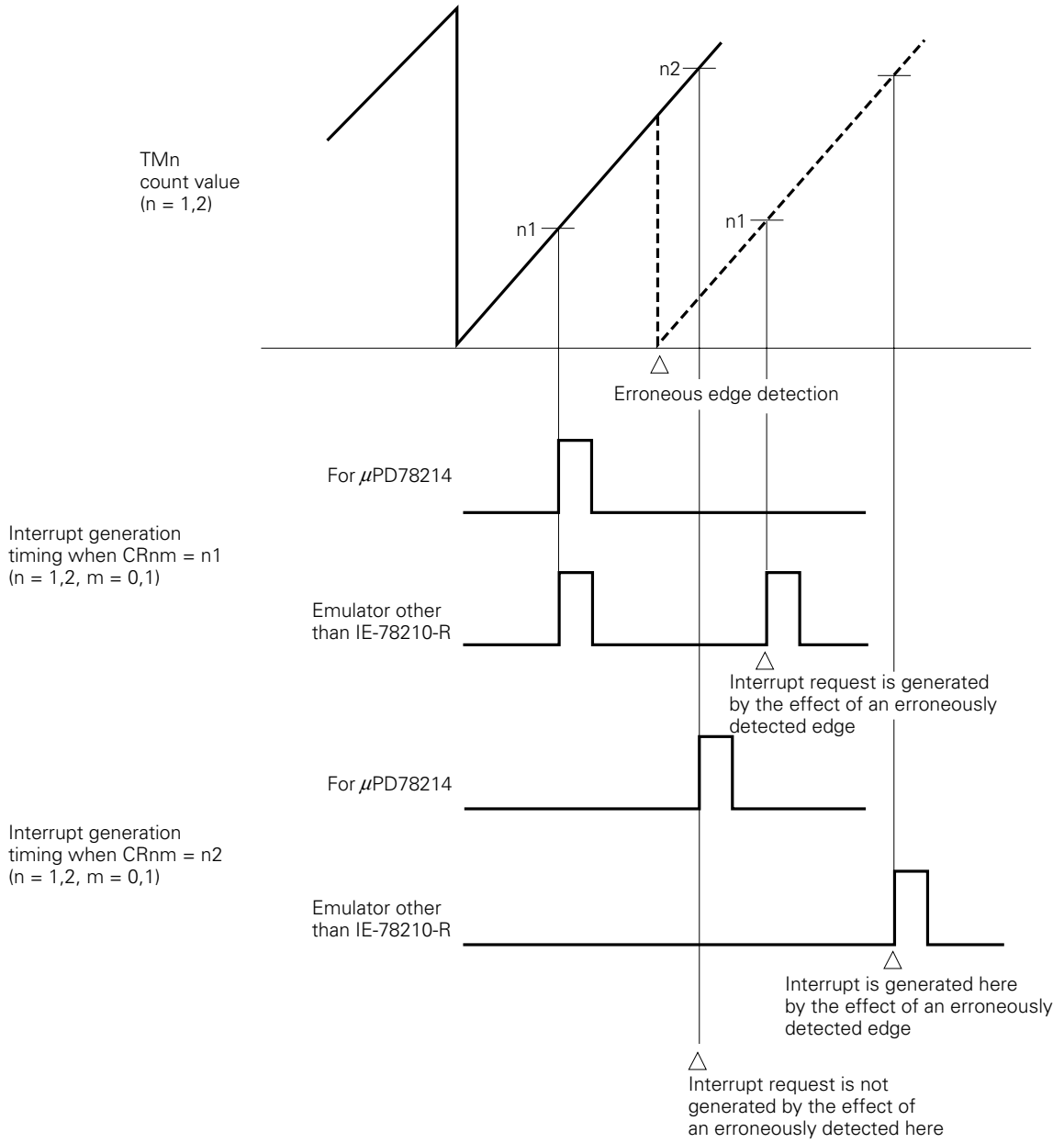
Clear operation is not performed on an erroneously detected edge. After erroneous edge detection, however, an interrupt request to be generated when the value of the timer/counter coincides with the value of a compare register is generated with the timing not based on the values of the timer/counter and compare register. This interrupt generation timing is the timing assuming that the timer/counter is cleared. (See **Fig. 7-141**.)

When a coincidence between the value of 8-bit timer/counter 1 and the value of a compare register is used as an output trigger for a real-time output port, such a deviated timing is used as an output trigger for the real-time output port.

The timer output function of 8-bit timer/counter 2 is not affected by an erroneously detected edge, but operates with the correct timing. Such an interrupt generation timing deviation as described above can be corrected by the following operations:

- Clear operation on a normal edge
- Clearing bit CEn (n = 1, 2) for the 8-bit timer/counter in timer control register 1 (TMC1) to 0

Fig. 7-141 Interrupt Generation Timing Change by an Erroneously Detected Edge



(c) Event counter function (with only 8-bit timer/counter 2)

An erroneously detected edge causes no change in the value of the timer/counter. However, the timing for generating an interrupt by a coincidence between the value of the timer/counter and the value of a compare register becomes faster by the number of edges detected erroneously.

The timer output function is not affected by an erroneously detected edge, but operates with the correct timing.

Such an interrupt generation timing deviation as described above can be corrected by the following operations:

- Clear operation on a normal edge when the clear function following capture operation is used
- Clearing the CE2 bit of timer control register 1 (TMC1) to 0

CHAPTER 8 A/D CONVERTER

The μ PD78214 contains an analog-to-digital (A/D) converter with eight multiplexed analog input pins (AN0 through AN7).

This A/D converter uses successive approximation. The conversion result is stored in an 8-bit A/D conversion result register (ADCR). Conversion can be performed at high speed (with conversion time of 30 μ s and at $f_{CLK} = 6$ MHz) and with high accuracy.

The A/D converter can be started in the following two modes:

- Hardware start: Conversion starts on a trigger input (INTP5).
- Software start: Conversion starts as specified in the A/D converter mode register (ADM).

Once started, the A/D converter operates in either of the following modes:

- Scan mode: Analog inputs are sequentially selected for A/D conversion from all the input pins.
- Select mode: Only one input pin is used. Analog signals are successively input from this pin to the converter.

These start and operating modes are specified by the ADM register. The ADM register is also used to stop conversion.

When the conversion result is sent to the ADCR register, an interrupt request (INTAD) is generated (except in the select mode started by software). Therefore, the conversion result can be successively sent to the memory by using a macro service.

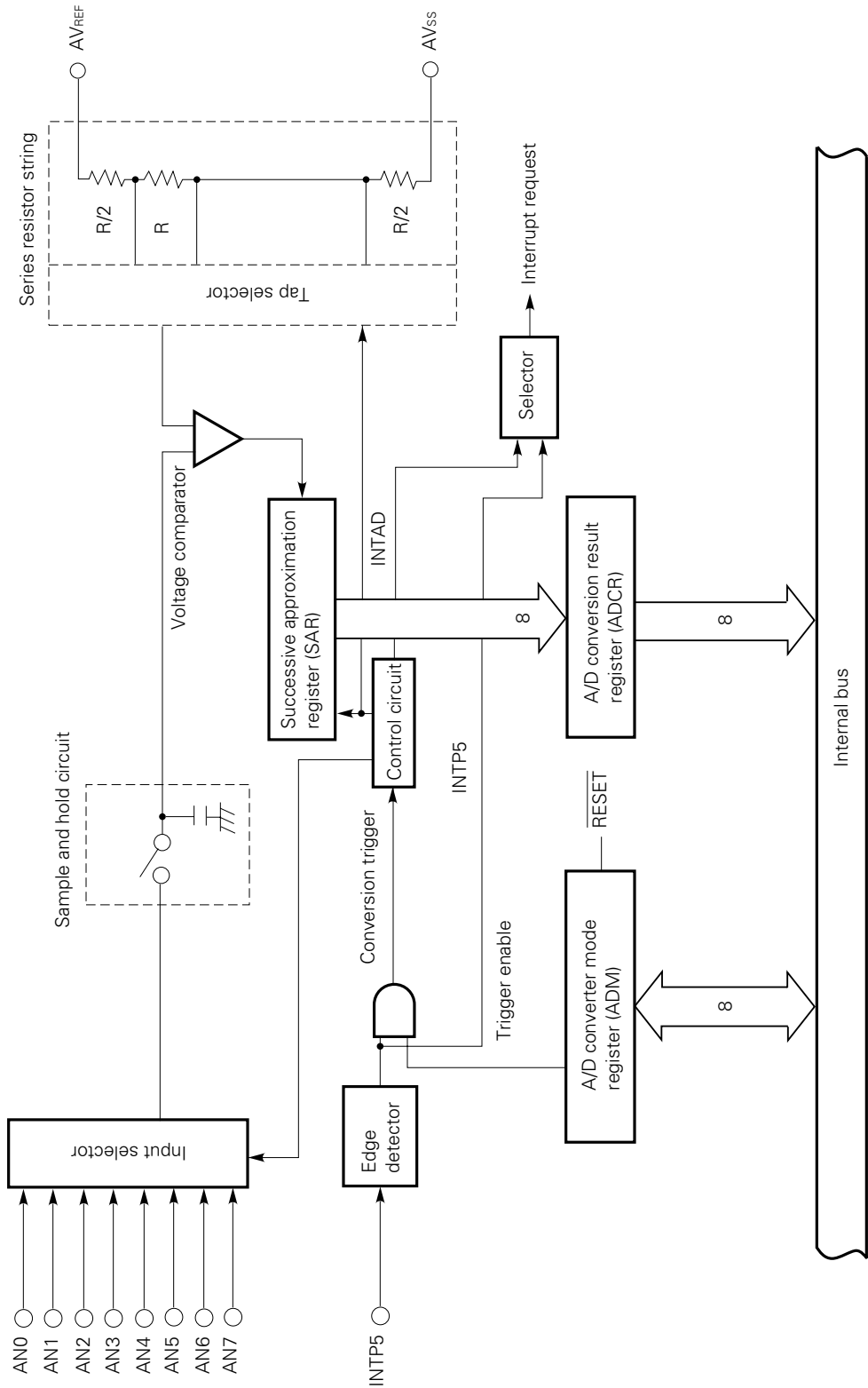
Table 8-1 Modes Generating the INTAD

	Scan mode	Select mode
Hardware start	○	○
Software start	○	—

8.1 CONFIGURATION

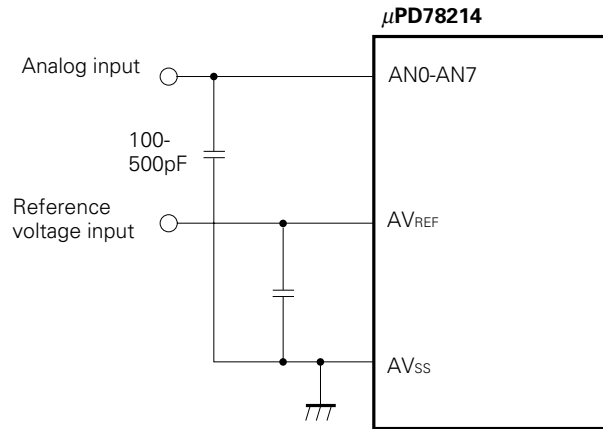
Fig. 8-1 shows the configuration of the A/D converter.

Fig. 8-1 A/D Converter Configuration



Cautions 1. To prevent malfunction due to noise, insert a capacitor between each analog input pins (AN0 through AN7) and the AV_{SS} pin and between the reference voltage input pin (AV_{REF}) and the AV_{SS} pin.

Fig. 8-2 Example of Capacitors Connected to the A/D Converter Pins



2. Do not apply a voltage out of the rated voltage range (AV_{SS} through AV_{REF}) to the A/D converter input pins. See Section 8.6 for details.

(1) Input circuit

The input circuit selects an input analog signal as specified by the A/D converter mode register (ADM) and sends the signal to the sample and hold circuit in accordance with a specified operating mode.

(2) Sample and hold circuit

The sample and hold circuit samples each of the analog signals successively sent from the input circuit and retains the analog signals during A/D conversion.

(3) Voltage comparator

The voltage comparator compares the potential differences between the analog inputs and the voltage taps of the serial resistor string.

(4) Series resistor string

The series resistor string generates a voltage that matches the input analog signal voltage.

The series resistor string is connected between the reference voltage pin (AV_{REF}) and the GND pin (AV_{SS}) of the A/D converter. It consists of 255 resistors, each having equal resistance, and two resistors, each having half the resistance of the other 255 resistors. This configuration enables the voltage between the AV_{REF} and AV_{SS} pins to be divided into 256 steps.

A voltage tap of the resistor string is selected by a tap selector, which is controlled by the SAR register.

(5) Successive approximation register (SAR)

When the voltage at one of the voltage taps of the serial resistor string matches the analog input voltage, this 8-bit register is set with the corresponding data on a bit-by-bit basis, starting at the most significant bit (MSB).

When the SAR is set up to the least significant bit (LSB), the SAR contents (conversion result) are sent to the A/D conversion result register (ADCR) and held there.

(6) A/D conversion result register (ADCR)

This 8-bit register holds the A/D conversion result. Each time A/D conversion ends, this register is loaded with the conversion result received from the SAR.

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

(7) Edge detector

The edge detector detects the valid edge of an input at the interrupt request input pin (INTP5) and generates an external interrupt request signal (INTP5) and an external trigger for A/D conversion.

The valid edge of an input at the INTP5 pin is specified by external interrupt mode register 1 (INTM1) (see **Fig. 11-2**). The external trigger is enabled or disabled by the ADM register (see **Section 8.2**).

8.2 A/D CONVERTER MODE REGISTER (ADM)

This 8-bit register controls A/D converter operations.

A bit manipulation instruction or an 8-bit manipulation instruction can be used to read data from or write data to this register. Fig. 8-3 shows the ADM format.

Bit 0 (MS) of the ADM register controls the A/D converter operating mode.

Bits 1, 2, and 3 (ANI0, ANI1, and ANI2) select analog input signals to be converted to digital form.

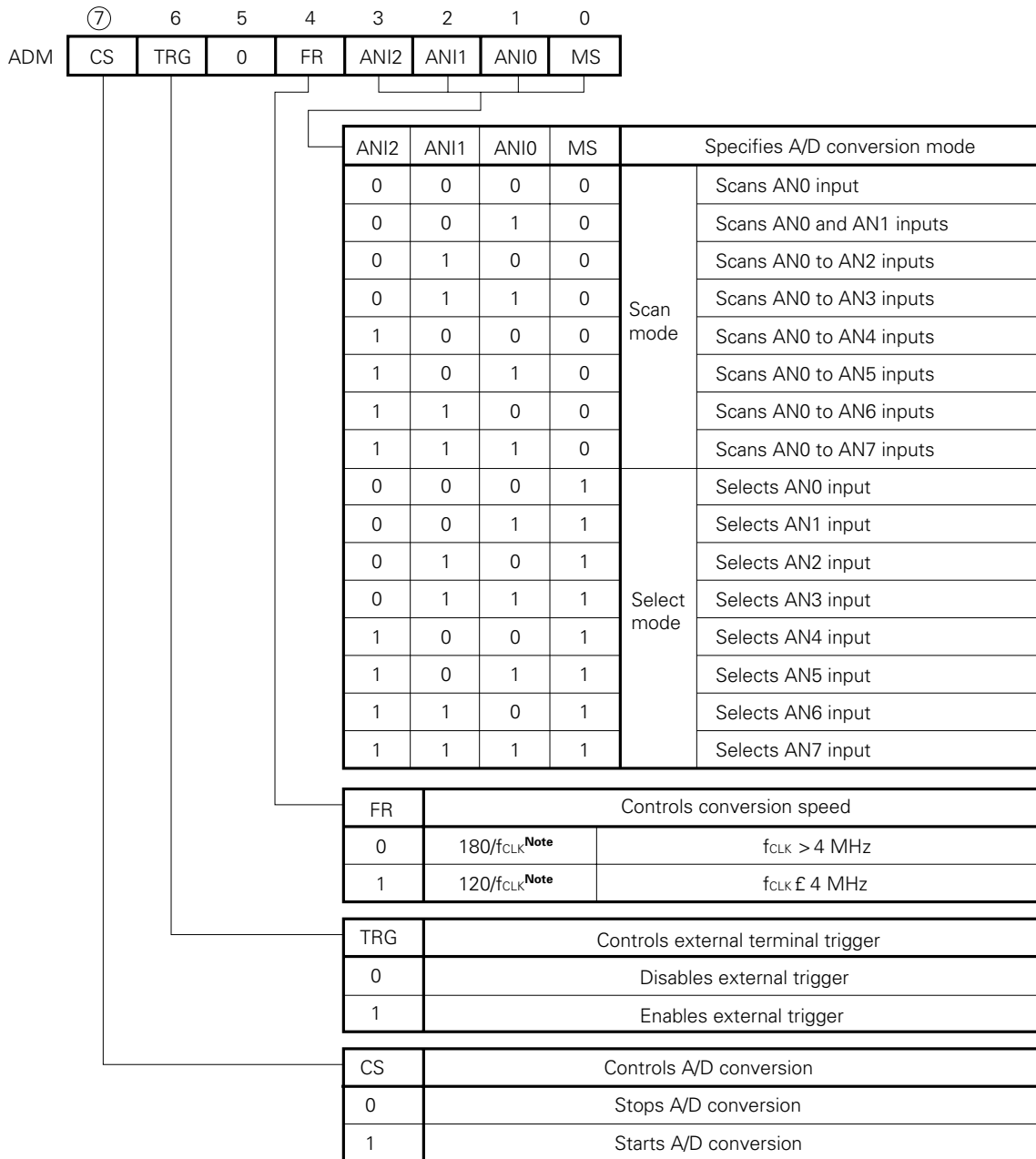
Bit 6 (TRG) is used to enable external synchronization for A/D conversion. When the TRG bit is set to 1, if the CS bit is already 1, the conversion is initialized each time a valid edge arrives at the INTP5 pin as an external trigger. When the TRG bits is reset to 0, conversion is carried out regardless of the state of the INTP5 pin.

Bit 7 (CS) controls A/D conversion. When this bit is set to 1, the A/D converter starts operating. When the CS bit is reset to 0, the converter stops, even if it is in the middle of conversion. At this point, however, the ADCR register contents are not updated, nor does an INTPAD interrupt occurs. Power supply to the voltage comparator is stopped to reduce supply current to the A/D converter.

When the $\overline{\text{RESET}}$ signal is input, the ADM register is reset to 00H.

Caution When using the STOP mode, reset the CS bit to 0 beforehand to reduce supply current. If the CS bit remains set to 1, conversion do stops when the STOP mode is selected, but power supply to the voltage comparator is not stopped. Consequently, the supply current to the A/D converter does not decrease.

Fig. 8-3 A/D Converter Mode Register (ADM) Format



Note f_{CLK} : System clock frequency

8.3 OPERATION

8.3.1 Basic A/D Converter Operation

(1) A/D conversion sequence

The A/D converter operates as follows:

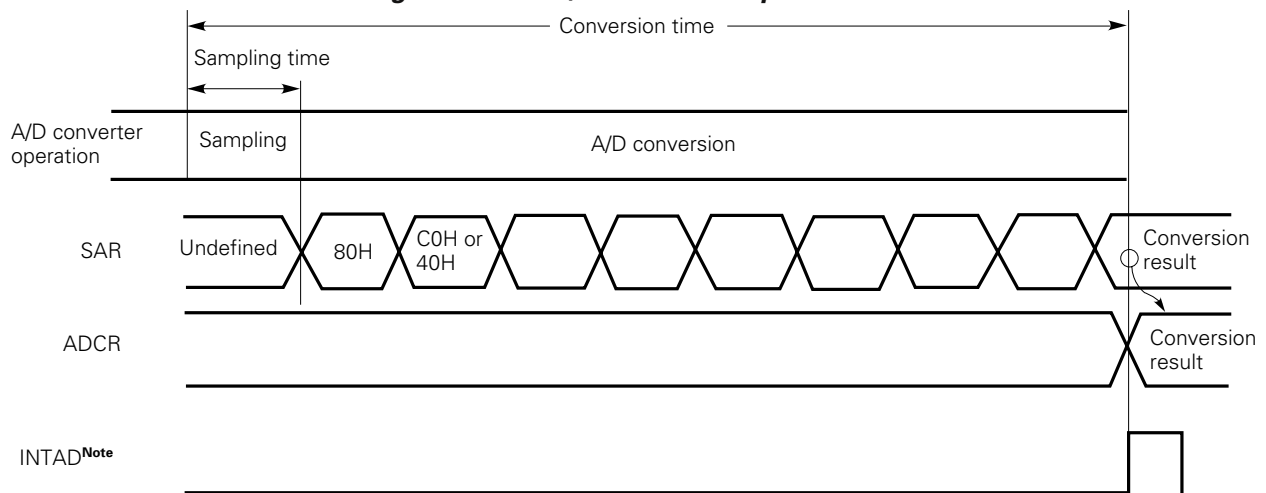
- (a) The input selector selects one of the analog input pins (AN0 through AN7) according to the mode of operation specified in the A/D converter mode register (ADM).
- (b) The sample and hold circuit samples the voltage at the analog input pin selected by the input selector.
- (c) After a certain sampling period, the sample and hold circuit goes on hold and retains the input analog voltage until A/D conversion ends.
- (d) When bit 7 of the SAR register is set, the tap selector sets the voltage tap of the serial resistor string to $(255/512)AV_{REF} (\cong (1/2)AV_{REF})$.
- (e) The voltage comparator compares the voltage at the serial resistor string with an input analog signal voltage. If the analog input signal voltage is greater than $(1/2)AV_{REF}$, the MSB of the SAR register remains set. If it is less than $(1/2)AV_{REF}$, the MSB is reset.
- (f) Bit 6 of the SAR register is set to 1 automatically, and the voltage comparator starts comparing the next analog input signal voltage. A voltage tap of the serial resistor string is selected as follows, according to the state of bit 7, which has already been set according to the result.
 - Bit 7 = 1 ... $(383/512)AV_{REF} \cong (3/4)AV_{REF}$
 - Bit 7 = 0 ... $(127/512)AV_{REF} \cong (1/4)AV_{REF}$

The voltage at this tap is compared with the input analog voltage. According to the comparison result, bit 6 of the SAR register is set or reset as follows:

- Analog input voltage \geq voltage tap voltage: Bit 6 = 1
 - Analog input voltage $<$ voltage tap voltage: Bit 6 = 0
- (g) These comparisons are carried out successively until the least significant bit (bit 0) of the SAR register is compared (binary search method).
 - (h) When all the 8 bits of the SAR register have been compared, a valid digital number is left in the SAR register. This digital number is sent to and latched in the ADCR register.

At the same time, an A/D conversion end interrupt (INTAD) can be generated (except in software-started select mode). This INTAD interrupt should be handled as either a vectored interrupt or by a macro service (described later).

Fig. 8-4 Basic A/D Converter Operation



Note Except for software-started select mode

A/D conversion continues until the CS bit is reset by software.

If data is written to the ADM register during conversion, conversion is initialized. If the CS bit is 1, conversion is started from the beginning.

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

(2) Input voltage and conversion result

The analog voltages input to the analog input pins (AN0 through AN7) are related to the A/D conversion result (value held in the ADCR), as follows:

$$\text{ADCR} = \text{INT}\left(\frac{V_{\text{IN}}}{AV_{\text{REF}}} \times 256 + 0.5\right)$$

or,

$$(\text{ADCR} - 0.5) \times \frac{AV_{\text{REF}}}{256} \leq V_{\text{IN}} < (\text{ADCR} + 0.5) \times \frac{AV_{\text{REF}}}{256}$$

Remark INT() : Function returning the integer part of a value specified in ()

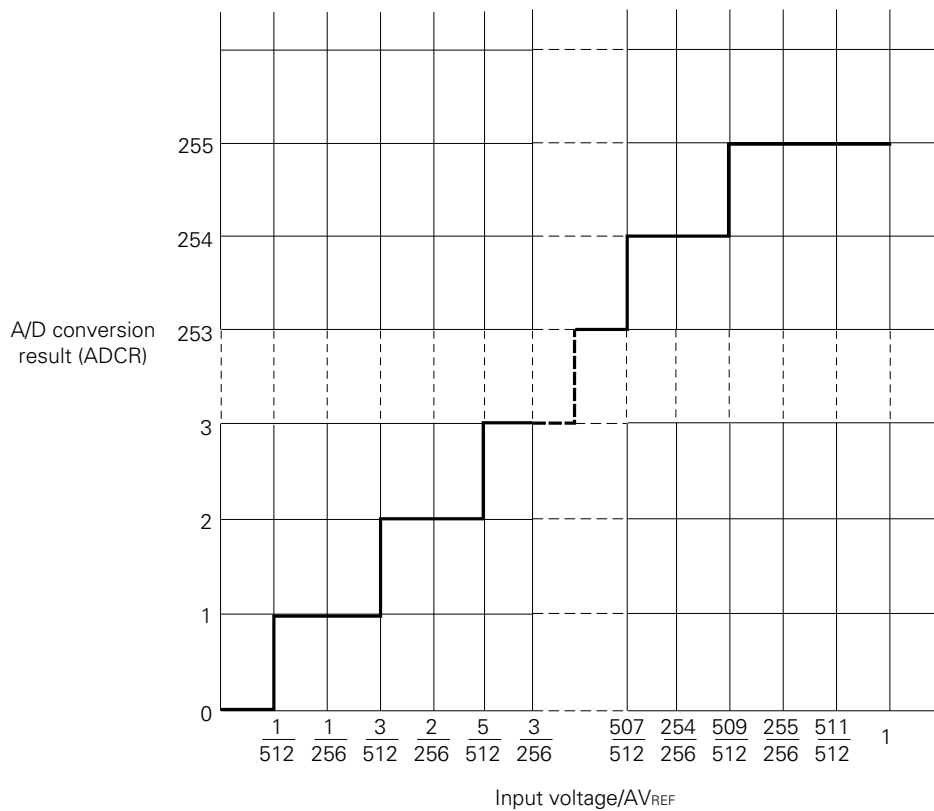
V_{IN} : Analog input voltage

AV_{REF} : AV_{REF} pin voltage

ADCR : Value in the ADCR register

Fig. 8-5 shows the relations between the analog voltages and the A/D conversion results.

Fig. 8-5 Relations between Analog Input Voltages and A/D Conversion Results



(3) A/D conversion time

The time required for A/D conversion is determined by the system clock frequency (f_{CLK}) and the FR bit of the ADM register. To maintain A/D conversion accuracy above a certain level, it is necessary to set the FR bit as listed in Table 8-2 according to the system clock frequency.

This A/D conversion time includes all the time required for one A/D conversion sequence and the sampling time.

Table 8-2 shows the conversion time and sampling time.

Table 8-2 A/D Conversion Time

System clock (f_{CLK}) range	FR bit	Conversion time	Sampling time
$4 \text{ MHz} < f_{CLK} \leq 6 \text{ MHz}$	0	$180/f_{CLK}$ (30 μs to 45 μs)	$36/f_{CLK}$ (6 μs to 9 μs)
$2 \text{ MHz} \leq f_{CLK} < 4 \text{ MHz}$	1	$120/f_{CLK}$ (30 μs to 60 μs)	$24/f_{CLK}$ (6 μs to 12 μs)

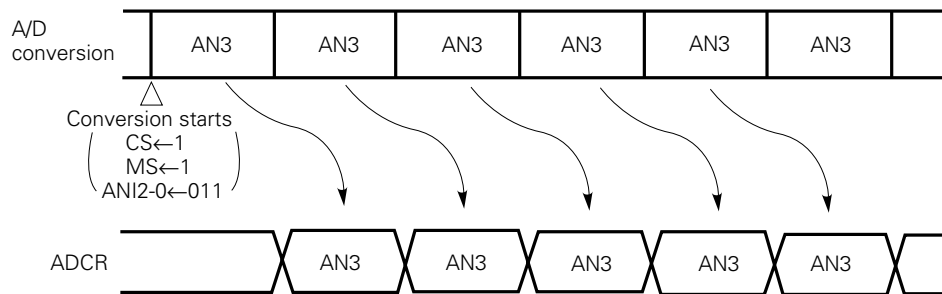
8.3.2 Select Mode

Bits 1 through 3 (ANI0 through ANI2) of the ADM register specify one analog input pin. A/D conversion is repeated for the specified pin. The resultant digital data is stored in the A/D conversion result register (ADCR).

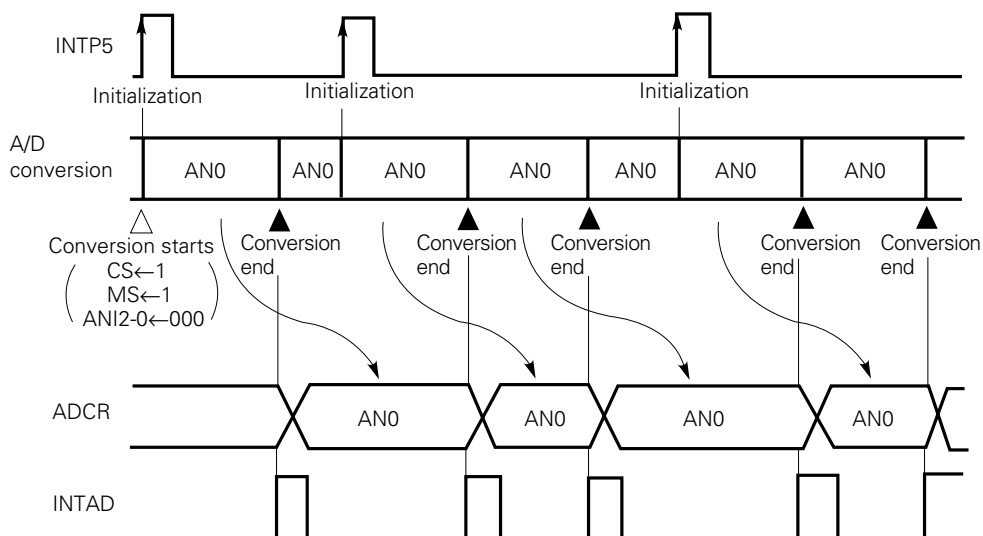
If bit 6 (TRG) of the ADM register is set to enable an external trigger, an A/D conversion end interrupt request (INTAD) is generated.

Fig. 8-6 Select Mode Operation Timing

(a) TRG bit ← 0



(b) TRG bit ← 1

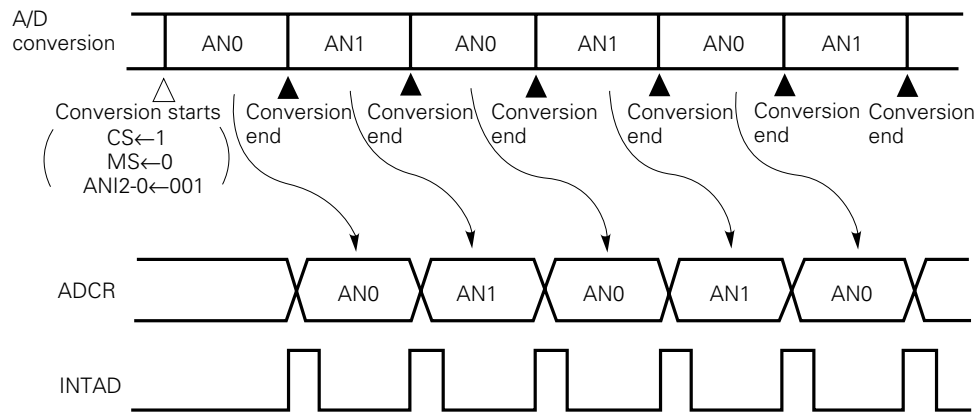


8.3.3 Scan Mode

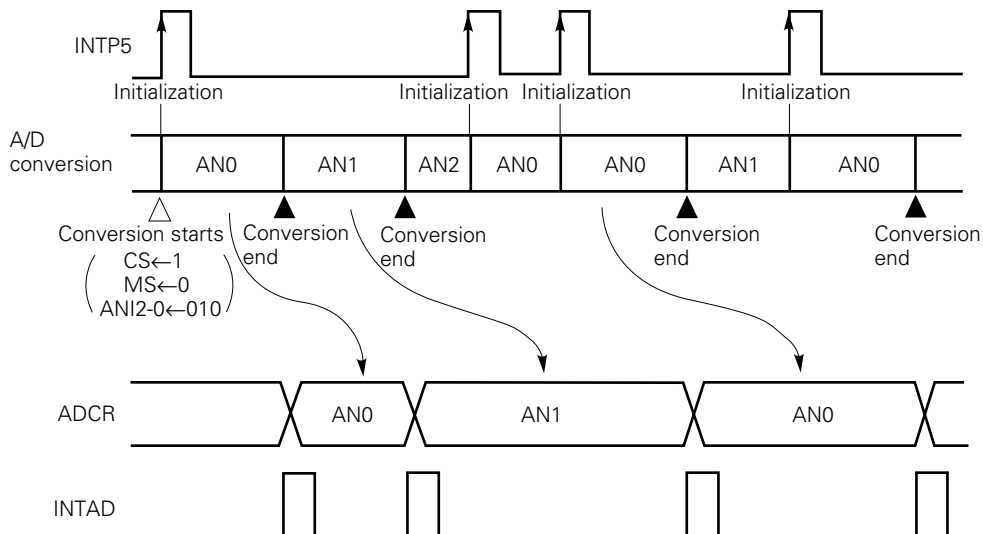
In the scan mode, signals input from the analog input pins, specified by bits 1 through 3 (ANI0 through ANI2) of the A/D converter mode register (ADM), are selected successively for conversion.

For example, when the ANI2 through ANI0 bits of the ADM register are 001, the AN0 and AN1 pins are scanned repeatedly, starting at the ANI0 pin in the sequence: AN0 → AN1 → AN0 → AN1 →... In this mode, each time conversion is completed, the conversion result is stored in the ADCR register, and an A/D conversion end interrupt request (INTAD) is generated.

Fig. 8-7 Scan Mode Operation Timing
(a) TRG bit ← 0



(b) TRG bit ← 1



Cautions 1. When the result of A/D conversion is read by using a vectored interrupt during the scan mode, if the A/D conversion end interrupt is kept pending for a prolonged time because of other interrupts being handled (at least 180 clocks if the FR bit is 0 or 120 clocks if the FR bit is 1), the conversion result cannot be accurately measured. To measure the conversion result accurately, take the following measures:

- Keep the time required to handle other interrupts adequately shorter than the required A/D conversion time.
- Use the multiplexed interrupt mode so that the A/D conversion end interrupt can be accepted even when other interrupts are being handled.
- Use a macro service to handle the A/D conversion end interrupt.

Note that the A/D conversion end interrupt may also be kept pending by the causes described in Section 12.3.5.

Of the measures described above, the macro service might be the simplest method for you application.

2. If the ADM register is set after registers related to interrupts have been set during the scan mode, an unwanted interrupt may occur, thus causing the storage location of the conversion result to appear to have shifted. To prevent this, take the actions listed below in the stated order.
 - Write to the ADM register.
 - Reset the interrupt request flag (PIF5) to 0.
 - Set the interrupt mask flag or interrupt service mode flag.

8.3.4 A/D Conversion Activated by Software Start

Software can start A/D conversion by writing such a value to the ADM register that the TRG bit is reset to 0 and the CS bit is set to 1.

If such a value is written to the ADM register again during A/D conversion (the CS bit is 1) that the TRG is reset to 0 and the CS bit is set to 1, the ongoing A/D conversion sequence is stopped, and another A/D conversion sequence (that matches the newly written value) is started immediately.

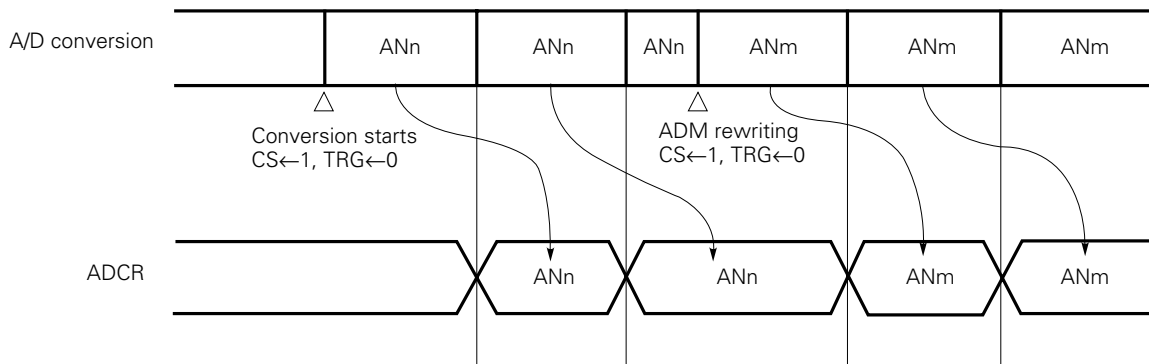
Once A/D conversion is started, the conversion sequence of the new data is started according to the mode of operation specified in the ADM register immediately when the conversion sequence of the previous data is completed. A/D conversion continues until a write instruction is executed for the ADM register.

If A/D conversion is started by software (the TRG bit is 0), an input to the INTP5 (pin P26) does not affect the conversion.

(1) Select-mode A/D conversion

In the select mode, the analog signal, input to the pin selected by the ADM register, is converted to digital form. When conversion is completed, the analog signal at the same pin is converted again. An interrupt request (INTAD) does not occur when conversion is completed.

Fig. 8-8 Software-Started Select-Mode A/D Conversion

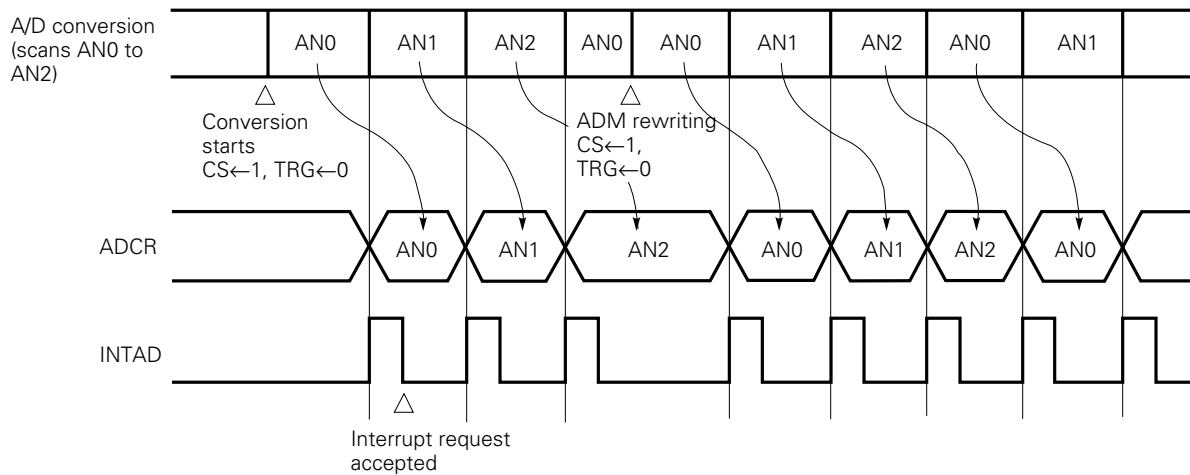


Remark n = 0, 1, ..., 7
 m = 0, 1, ..., 7

(2) Scan-mode A/D conversion

When triggered, conversion begins with the signal input to the AN0 pin. When the conversion sequence for the AN0 pin is completed, the signal at the next analog input pin is converted. Each time a conversion sequence is completed, an interrupt request (INTAD) is generated.

Fig. 8-9 Software-Started Scan-Mode A/D Conversion



8

8.3.5 A/D Conversion Activated by Hardware Start

Hardware can start A/D conversion by setting both the TRG and CS bits of the ADM register to 1. When they are set to 1, the A/D converter is ready to receive an external signal. A/D conversion begins when a valid edge arrives at the INT5 pin (pin P26).

After A/D conversion is started, if another valid edge arrives at the INT5 pin, the current sequence of A/D conversion is stopped, and another sequence of conversion is started from the beginning, in accordance with the current ADM register contents.

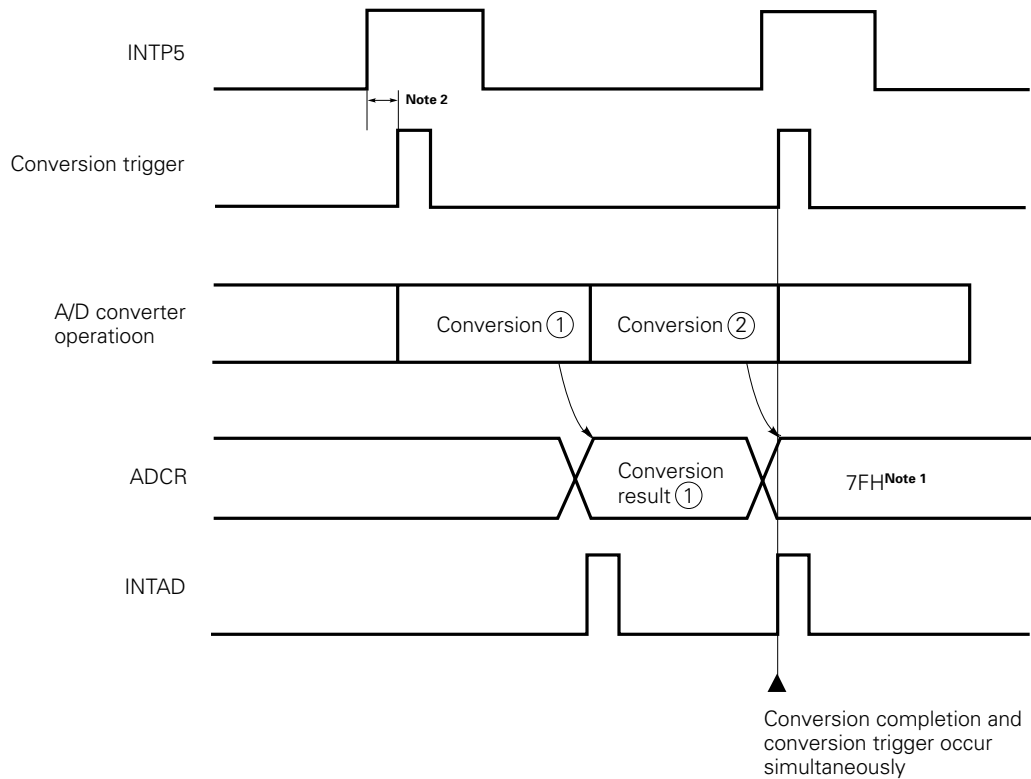
If such a value that both the TRG and CS bits are set to 1 is written to the ADM register again during A/D conversion (the CS bit is 1), the current conversion sequence (including a wait period for an external signal) is stopped. The converter stands by, until a valid edge arrives at the INTP5 pin in the mode of A/D conversion specified by the written value. When a valid edge arrives, conversion starts.

By using this function, A/D conversion can be synchronized with an external signal.

When one A/D conversion sequence is completed, another conversion sequence is started immediately in an operating mode set in the ADM register (the converter does not wait for the input from the INTP5 pin). Conversion continues until an instruction that writes to the ADM register is executed or until a valid edge arrives at the INTP5 pin.

- Cautions**
1. Eight to twelve system clocks are required from when a valid edge appears at the INTP5 pin until A/D conversion is actually started. Take this delay into consideration when designing your application. See Chapter 11 for details on the edge detection function.
 2. When A/D conversion is already activated by hardware start (by a valid edge at the INTP5 pin), if another valid edge arrives at the INTP5 pin, the A/D converter may malfunction. To be specific, the malfunction occurs if the valid edge arrives at the INTP5 pin when the previous conversion result is being stored in the A/D conversion result register (ADCR). In this case, an A/D conversion end interrupt (INTAD) is generated. However, the value stored in the ADCR register is not the conversion result. Instead, it is always 7FH (see Fig. 8-10).

Fig. 8-10 Example of Malfunction in a Hardware-Started A/D Conversion

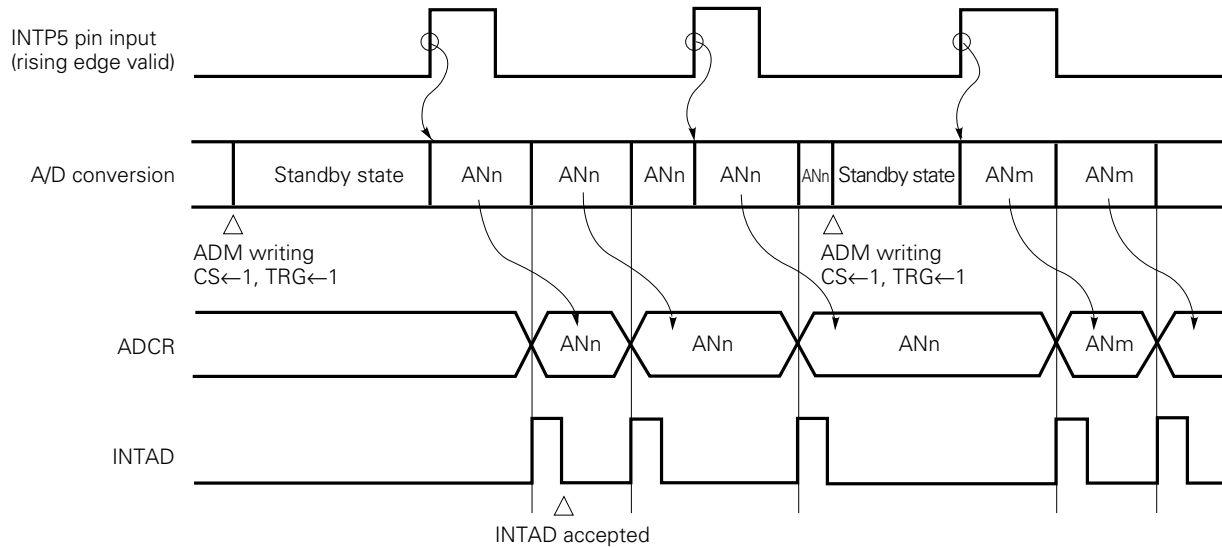


- Notes**
1. When the operation is normal, the result of conversion ② is stored. If a malfunction occurs, however, value 7FH is stored.
 2. Time from when an input to the INT5 pin changes to when its edge is asserted. See **Chapter 11** for details.
- In order to solve this problem, the A/D converter mode register (ADM) must be set again after the necessary A/D conversion is hardware-started and completed. A similar problem occurs during in-circuit emulation.**
3. Digital noise at the INTP6 pin cannot be eliminated normally with an in-circuit emulator. If it has been specified that A/D conversion be hardware-started, A/D conversion may be started by an erroneously detected edge. See Section 11.4 for detail on erroneous detection of an edge.

(1) Select-mode A/D conversion

The analog signal at the input pin, specified in the ADM register, is converted to digital form. When one A/D conversion sequence is completed, the analog signal at the same input pin is converted again. Each time a conversion sequence is completed, an interrupt request (INTAD) is generated.

If a valid edge arrives at the INTP5 pin during A/D conversion, the current conversion sequence is stopped, and another conversion session is started.

Fig. 8-11 Select-Mode A/D Conversion Started by Hardware

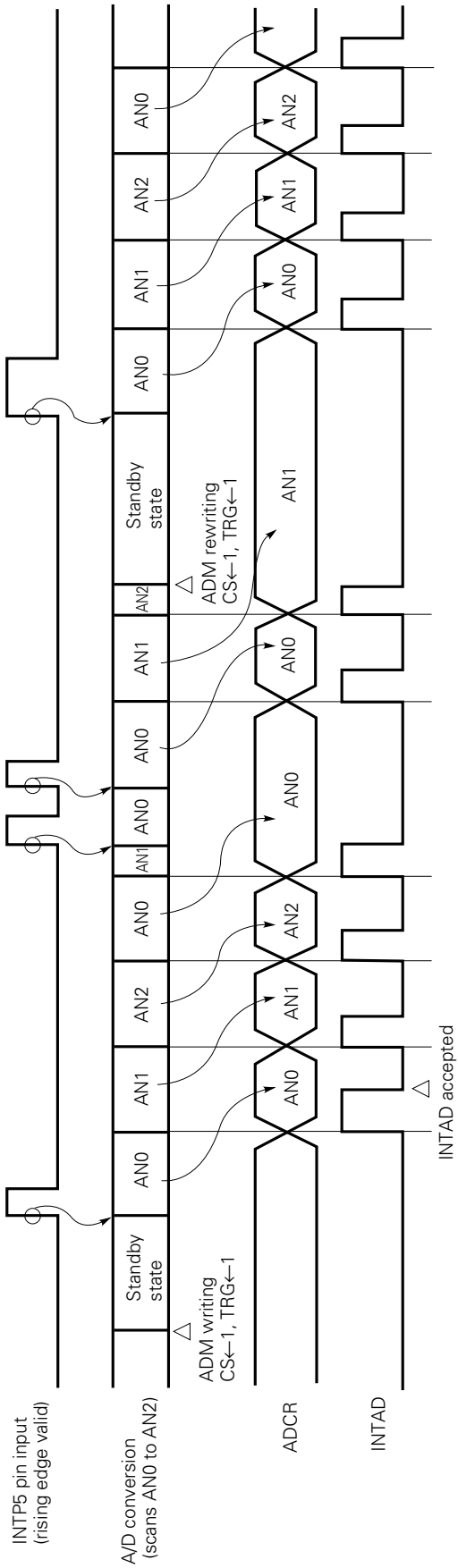
Remark $n = 0, 1, \dots, 7$
 $m = 0, 1, \dots, 7$

(2) Scan-mode A/D conversion

When A/D conversion is started, the analog signal input to the AN0 pin is converted. When one conversion sequence is completed, the signal from the next analog input pin is converted. Each time a conversion sequence is completed, an interrupt request (INTAD) is generated.

When a valid edge arrives at the INTP5 pin during A/D conversion, the current conversion sequence is stopped, and another conversion session is started for the signal at the AN0 pin.

Fig. 8-12 Scan-Mode A/D Conversion Started by Hardware



8.4 INTERRUPT REQUEST FROM THE A/D CONVERTER

The A/D converter generates an A/D conversion end interrupt request (INTAD), each time a conversion sequence is completed, except for the select mode.

The interrupt control flags are shared by the INTAD interrupt and the INTP5 external interrupt. Therefore, the timing at which an interrupt request occurs varies depending on the mode of A/D conversion specified in the ADM register, as listed in Table 8-3.

The INTAD interrupt is handled using the interrupt control register in the same manner as the INTP5 interrupt. See Chapter 12 for details.

Table 8-3 Conditions to Generate Interrupt Requests in Each A/D Converter Operating Mode

A/D converter operation	Interrupt request flag	Mask flag	Interrupt request	Condition to generate interrupt requests
Stop	PIF5	PMK5	INTP5	Valid edge input to the INTP5 pin
Scan mode			INTAD	A/D conversion end
Select mode			INTP5	Valid edge input to the INTP5 pin
Hardware-started A/D conversion			INTAD	A/D conversion end

8.5 SETTING FOR USE OF AN6 AND AN7

When using AN6 or AN7, set up as follows:

(1) When using AN6

- Specify an internal weight (according to the MM and PW registers)
- Specify P66 as an input port (PM66 = 1)
- Do not use an internal pull-up resistor (PUO6 = 0)

(2) When using AN7

- Inhibit refresh (RFEN = 0)
- Specify P67 as an input port (PM67 = 1)
- Do not use an internal pull-up resistor (PUO6 = 0)

8.6 NOTES

(1) Range of voltages applied to analog input pints

When using the A/D converter input pins AN0 through AN7 (P66, P67, P70 through P75, observe the following points. Otherwise, the μ PD78214 may be damaged.

- Do not apply voltages out of the range of AV_{SS} to AV_{REF} to the pin subjected to A/D conversion.
- If the A/D converter is not in use (not operating), do not apply voltages out of the range of AV_{SS} through AV_{REF} to the pin **Note** selected by the A/D converter mode register (ADM).

Especially after the \overline{RESET} signal is input, observe the following points, because AN0 (P70) is selected automatically.

- When you clamp the AV_{REF} pin to the V_{SS} level, also clamp the AN0 (P70) pin to the V_{SS} level.
- When you use the AN0 (P70) pin, clamp the AV_{REF} to the V_{DD} level or keep the input to the AN0 (P70) pin below the potential at the AV_{REF} pin.

Note If the MS bit is 1, a pin selected by the ADM register is a pin subjected to A/D conversion. If the MS bit = 0, a pin selected by the ADM register is the AN0 pin.

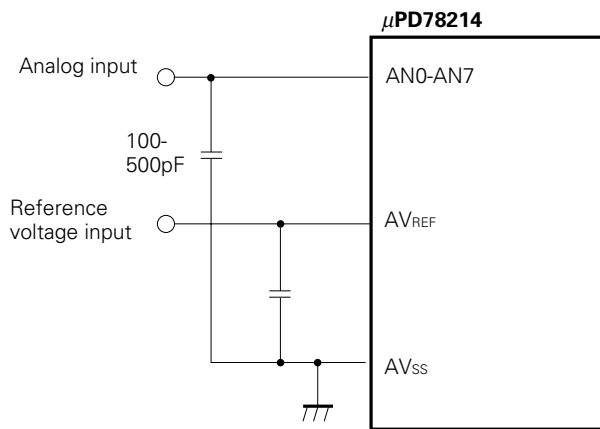
(2) About hardware-started A/D conversion

- (a) Eight to twelve system clocks are required from when a valid edge appears at the INTP5 pin until A/D conversion is actually started. Take this delay into consideration when designing your application. See **Chapter 11** for details of the edge detection function.
- (b) Digital noise at the INTP6 pin cannot be eliminated normally with an in-circuit emulator. If it has been specified that A/D conversion be hardware-started, A/D conversion may be started by an erroneously detected edge. See **Section 11.4** for detail of erroneous detection of an edge.

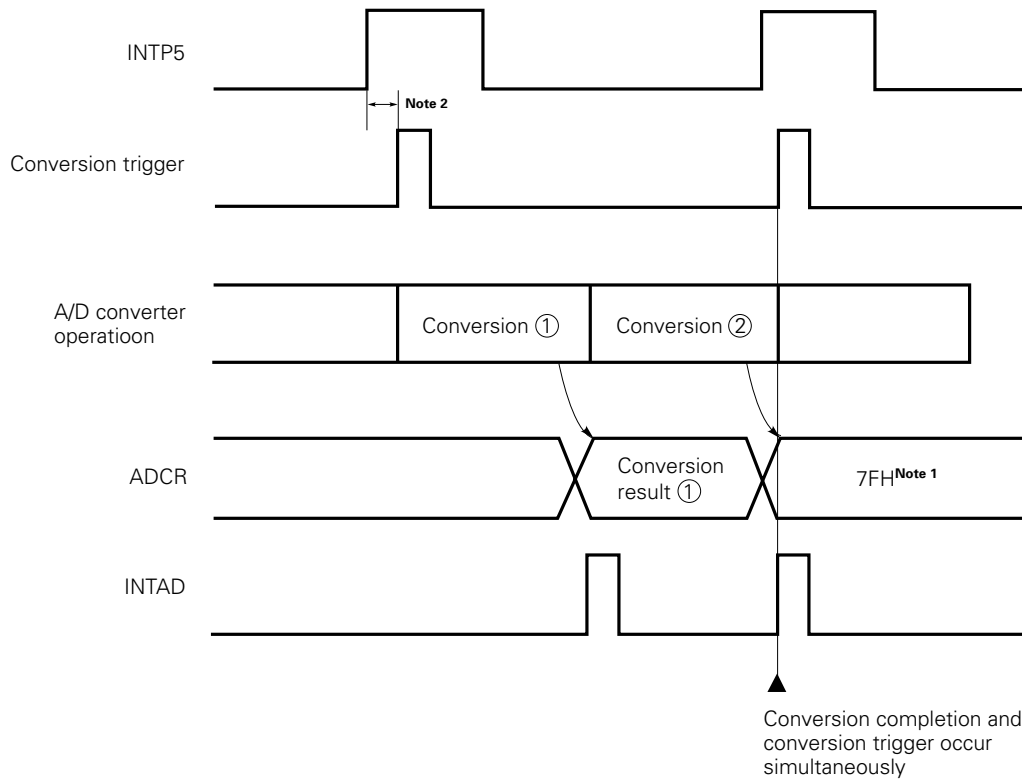
(3) Capacitors connected to analog input pins

To prevent malfunction due to noise, insert a capacitor between each analog input pins (AN0 through AN7) and the AV_{SS} pin and between the reference voltage input pin (AV_{REF}) and the AV_{SS} pin.

Fig. 8-13 Example of Capacitors Connected to the A/D Converter Pins



- (4) When using the STOP mode, reset the CS bit to 0 beforehand to reduce supply current. If the CS bit remains set to 1, conversion do stops when the STOP mode is selected, but power supply to the voltage comparator is not stopped. Consequently, the supply current to the A/D converter does not decrease.
- (5) When A/D conversion is already activated by hardware start (by a valid edge at the INTP5 pin), if another valid edge arrives at the INTP5 pin, the A/D converter may malfunction. To be specific, a malfunction occurs if the valid edge arrives at the INTP5 pin when the previous conversion result is being stored in the A/D conversion result register (ADCR). In this case, an A/D conversion end interrupt (INTAD) is generated. However, the value stored in the ADCR register is not the conversion result. Instead, it is always 7FH (see **Fig. 8-14**).

Fig. 8-14 Example of Malfunction in a Hardware-Started A/D Conversion

- Notes** 1. When the operation is normal, the result of conversion ② is stored. If a malfunction occurs, however, value 7FH is stored.
 2. Time from when an input to the INT5 pin changes to when its edge is asserted. See **Chapter 11** for details.

In order to solve this problem, the A/D converter mode register (ADM) must be set again after the necessary A/D conversion is hardware-started and completed. A similar problem occurs during in-circuit emulation.

- (6) When the result of A/D conversion is read by using a vectored interrupt during the scan mode, if the A/D conversion end interrupt is kept pending for a prolonged time because of other interrupts being handled (at least 180 clocks if the FR bit is 0 or 120 clocks if the FR bit is 1), the conversion result cannot be accurately measured. To measure the conversion result accurately, take the following measures:
- Keep the time required to handle other interrupts adequately shorter than the required A/D conversion time.
 - Use the multiplexed interrupt mode so that the A/D conversion end interrupt can be accepted even when other interrupts are being handled.
 - Use a macro service to handle the A/D conversion end interrupt.
- Note that the A/D conversion end interrupt may also be kept pending by the causes described in Section 12.3.5. Of the measures described above, the macro service might be the simplest method for you application.
- (7) If the ADM register is set after registers related to interrupts have been set during the scan mode, an unwanted interrupt may occur, thus causing the storage location of the conversion result to appear to have shifted. To prevent this, take the actions listed below in the stated order.
- Write to the ADM register.
 - Reset the interrupt request flag (PIF5) to 0.
 - Set the interrupt mask flag or interrupt service mode flag.

CHAPTER 9 ASYNCHRONOUS SERIAL INTERFACE

The μ PD78214 contains an asynchronous serial interface, UART (Universal Asynchronous Receiver Transmitter). This interface transmits 1-byte data following a start bit and is capable of full-duplex transmission.

The μ PD78214 also contains a baud rate generator for UART, which allows data to be transmitted at a wide baud rate range.

In addition, a baud rate can also be specified by dividing the frequency of the clock input to the ASCK pin.

Moreover, 8-bit timer/counter 3 can be used to generate a baud rate.

When the baud rate generator for UART is used, the MIDI standard baud rate (31.25 kbps) can also be obtained.

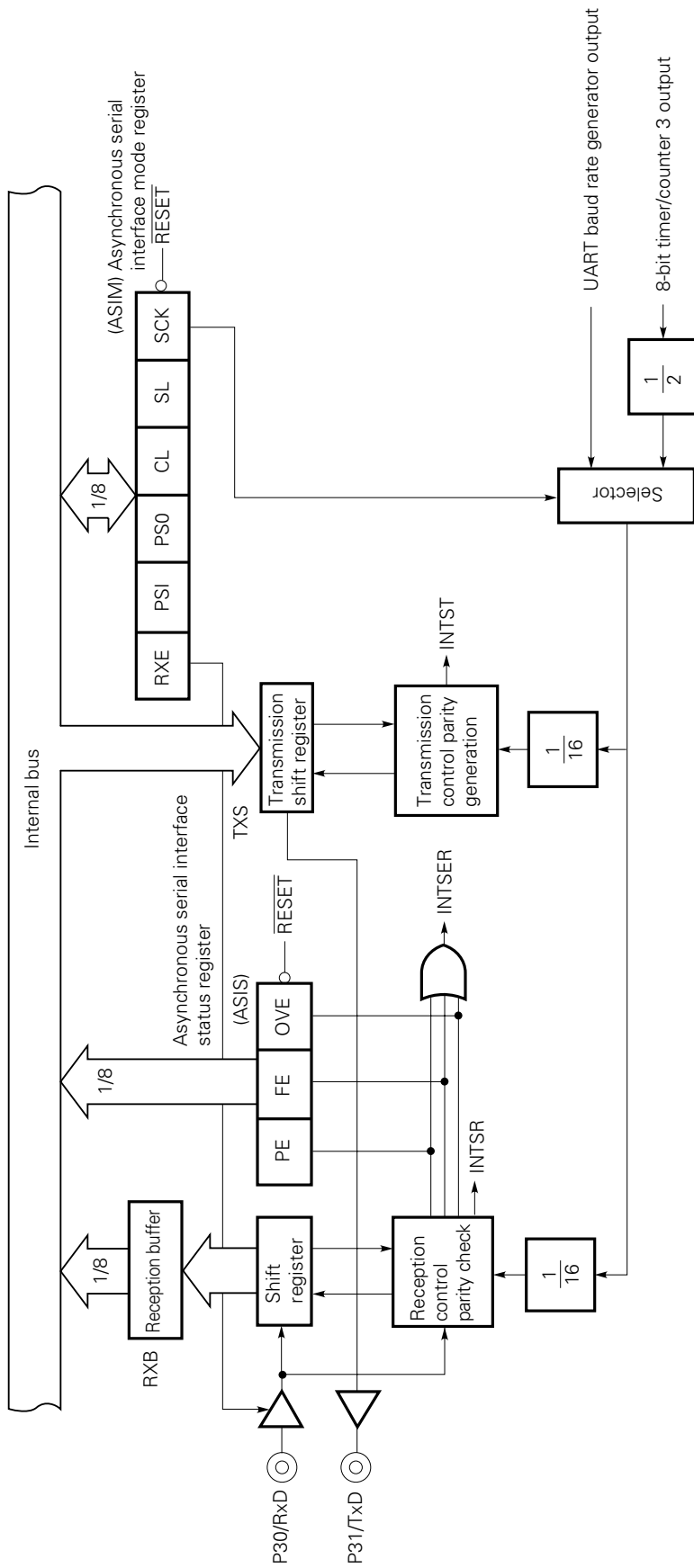
The asynchronous serial interface operates independently of the clock-synchronized serial interface.

9.1 CONFIGURATION

This section describes the configuration of the asynchronous serial interface.

See **Section 9.4** for details of the baud rate generator.

Fig. 9-1 Asynchronous Serial Interface Configuration



(1) Reception buffer (RXB)

The reception buffer holds the receive data. Each time the shift register receives 1 byte of data, it sends it to this reception buffer.

If the data length is specified to be 7 bits, the receive data is sent to bits 0 through 6 of the RXB. The MSB of the RXB is always kept as 0.

Only an 8-bit manipulation instruction can be used for the reception buffer, and its use is limited to read operations. When the $\overline{\text{RESET}}$ signal is input, the contents of the RXB become undefined.

(2) Transmission shift register (TXS)

The transmission shift register holds the data to be transmitted. The data written to the TXS is transmitted as serial data.

If the data length is specified to be 7, bits 0 through 6 of the data written to the TXS register are treated as the transmit data. Writing data to the TXS register triggers transmission. Do not write to the TXS register when transmission is in progress.

Only an 8-bit manipulation instruction can be used for the transmission shift register, and its use is limited to write operations. When the $\overline{\text{RESET}}$ signal is input, the contents of the TXS become undefined.

(3) Shift register

The shift register converts the serial data input to the RxD pin into parallel data. When it receives 1 byte of data, it sends it to the reception buffer.

The shift register cannot be manipulated directly from the CPU.

(4) Reception control parity check

Reception is controlled according to the contents of the asynchronous serial interface mode register (ASIM). In addition, error checks such as parity error check are also performed during reception. If an error is detected, a value corresponding to the error is set in the asynchronous serial interface status register (ASIS).

(5) Transmission control parity generation

Transmission is controlled by appending a start bit, parity bit, and one or two stop bits to the data written to the TXS register according to the contents of the ASIM register.

(6) Selector

The selector selects a baud rate clock source.

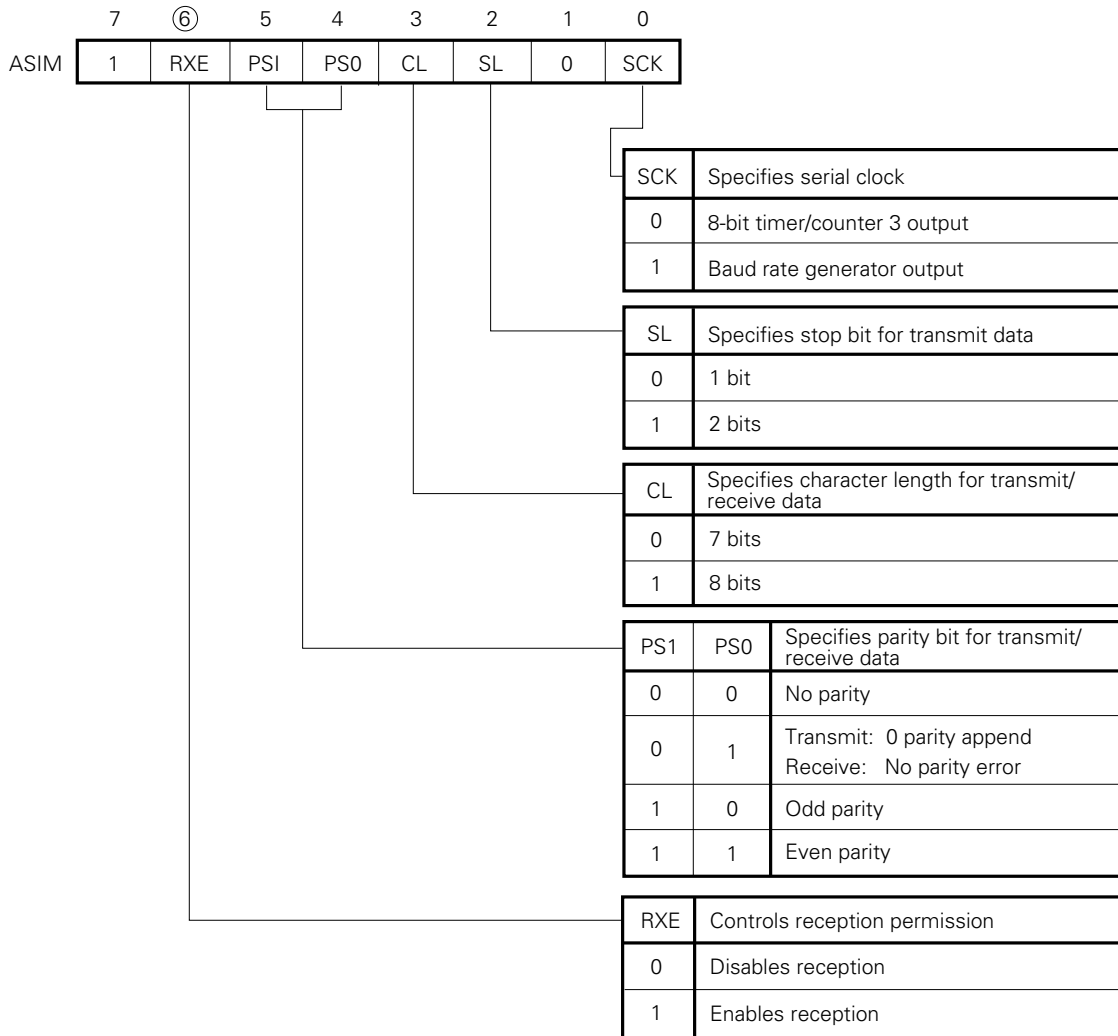
9.2 ASYNCHRONOUS SERIAL INTERFACE CONTROL REGISTER**(1) Asynchronous serial interface mode register (ASIM)**

This 8-bit register specifies the asynchronous serial interface operations.

Either 8-bit manipulation instruction or a bit manipulation instruction can be used to read data from or write data to this register. Fig. 9-2 shows the format of the asynchronous serial interface control register.

When the $\overline{\text{RESET}}$ signal is input, the ASIM register is set to 80H.

Fig. 9-2 Format of the Asynchronous Serial Interface Mode Register (ASIM)



Cautions 1. The asynchronous serial interface mode register (ASIM) must not be modified during transmission. If the ASIM register is modified during transmission, further transmission becomes impossible (inputting the $\overline{\text{RESET}}$ signal resumes normal operation).

Software can determine whether transmission is in progress, using the transmission completion interrupt (INTST) or the interrupt request flag (STIF), which is set by the INTST.

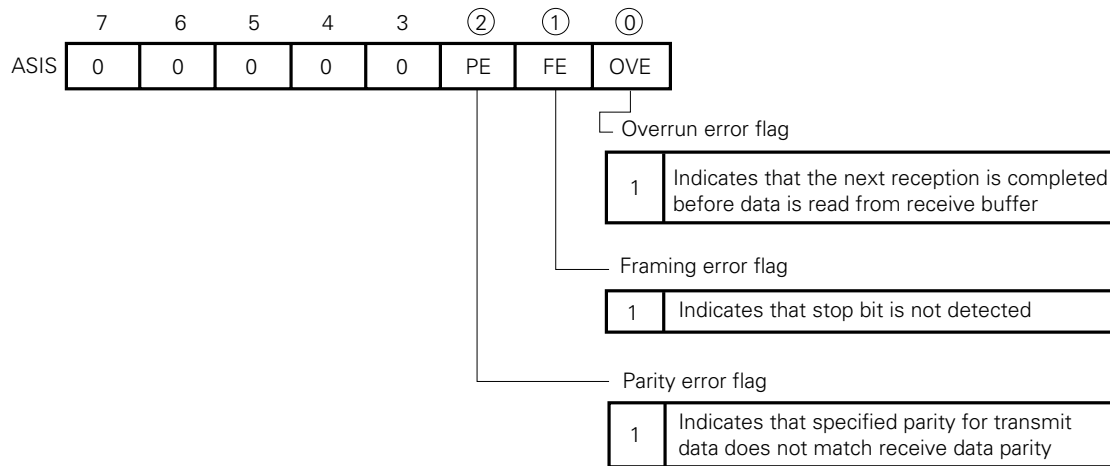
2. If the ASIM register is modified during reception, the current and next receive data may be damaged. When changing the mode, disable reception beforehand.

(2) Asynchronous serial interface status register (ASIS)

The ASIS register is a collection of flags that describe reception errors. A flag is set to 1 when a reception error occurs. It is reset to 0 by reading data from the reception buffer. When the next data is received, the overrun error flag (OVE) is set to 1, and the other error flags are reset to 0 (if this new data also contains an error, the error flag corresponding to that error is set to 1).

Both 8-bit manipulation instruction and bit manipulation instruction can be used for the ASIS register, but their use is limited to read operations.

When the $\overline{\text{RESET}}$ signal is input, the ASIS register is reset to 00H.

Fig. 9-3 Format of the Asynchronous Serial Interface Status Register (ASIS)

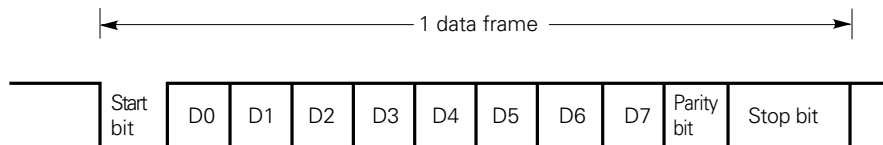
Caution Be sure to read the reception buffer (RXB) contents, even if a reception error occurs. Otherwise, an overrun error will occur when the next data is received, and the error status will persist.

9.3 ASYNCHRONOUS SERIAL INTERFACE OPERATIONS

9.3.1 Data Format

Fig. 9-4 shows the format of the transmit/receive data. One data frame consists of a start bit, character bits, a parity bit, and one or two stop bits.

The asynchronous serial interface mode register (ASIM) specifies the number of character and stop bits, and whether to use a parity bit.

Fig. 9-4 Format of the Transmission/Reception Data at the Asynchronous Serial Interface

- Start bit : 1 bit
- Character bits : 7 or 8 bits
- Parity bit : Even parity, odd parity, 0 parity, or no parity
- Stop bit : 1 or 2 bits

The serial transmission rate can range from 1.43 bps to 93.75 kbps according to the setting of the asynchronous serial interface mode register and baud rate generator or timer/counter 3.

If an error occurs during reception of serial data, the error can be identified by reading the contents of the asynchronous serial interface status register (ASIS).

9.3.2 Parity Types and Operations

The parity bit is used to detect a bit error in transmit/receive data. Usually, the same parity bit is used at both the transmission and reception ends. When even or odd parity is used, a 1-bit (the odd number of bits) error can be detected. When 0 parity or no parity is used, no error can be detected.

• Even parity

When the transmit data has an odd (or even) number of 1 bits, the parity bit is set to 1 (or 0), so that the number of 1 bits in the data becomes even. When data is received, the number of 1 bits in it is counted, and if the number of 1 bits is odd, a parity error is detected.

- **Odd parity**

In contrast to even parity, the parity bit for odd parity is controlled so that the number of 1 bits in the transmit data becomes odd. When data is received, the number of 1 bits in it is counted, and if the number of 1 bits is even, a parity error is detected.

- **0 parity**

When data is transmitted, the parity bit is reset to 0, regardless of what the transmit data is like. During reception, the parity bit is not checked. Therefore, a parity error is not detected, regardless of whether the parity bit is 0 or 1.

- **No parity**

No parity bit is attached to the transmit data. The reception end assumes that there is no parity bit. Because no parity bit is used, no parity error is detected.

9.3.3 Transmission

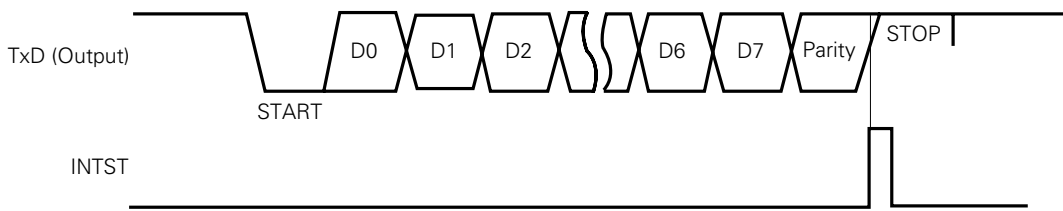
The asynchronous serial interface for the μPD78214 is always ready to transmit data. Writing transmit data to the transmission shift register (TXS) triggers transmission. The start bit, parity bit, and stop bit(s) are attached automatically.

When transmission is triggered, the transmission shift register (TXS) shifts out its contents. When the register becomes empty, a transmission completion interrupt (INTST) occurs.

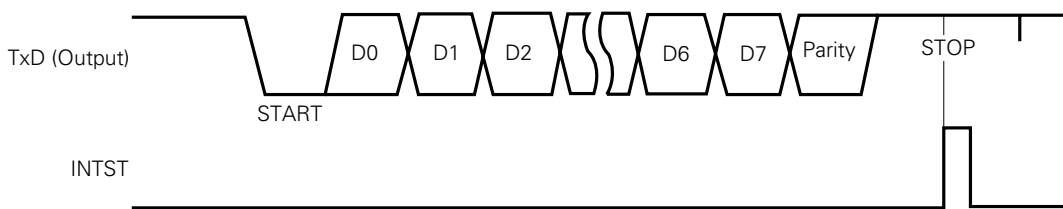
If no further transmission data is written to the transmission shift register (TXS), transmission breaks.

Fig. 9-5 Asynchronous Serial Interface Transmission Completion Interrupt Timing

(a) Stop bit length: 1



(b) Stop bit length: 2



- Cautions**
1. When the **RESET** signal is input, the transmission shift register becomes empty, but no transmission completion interrupt occurs. Transmission is triggered by writing the transmit data to the transmission shift register.
 2. The asynchronous serial interface mode register (ASIM) must not be modified during transmission. If the ASIM register is modified during transmission, further transmission becomes impossible (inputting the **RESET** signal resumes normal operation).
Software can determine whether transmission is in progress, using the transmission completion interrupt (INTST) or the interrupt request flag (STIF), which is set by the INTST.

9.3.4 Reception

When the RXE bit of the asynchronous serial interface mode register (ASIM) is set to 1, reception is enabled, and the input to the RxD pin is sampled.

Sampling at the RxD pin is performed using the serial clock specified in the ASIM register.

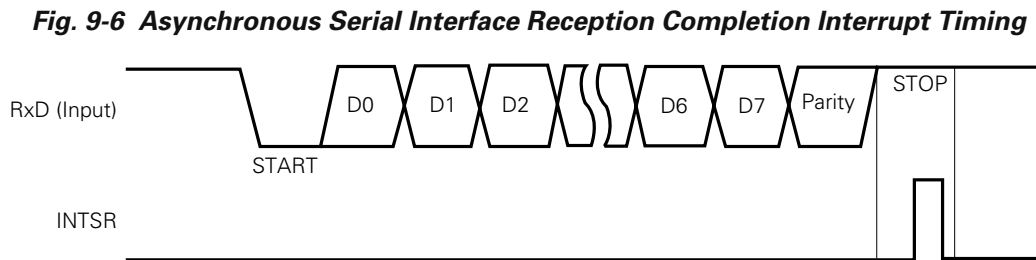
When the input to the RxD pin becomes low, the 1/16 frequency division counter starts counting. When the counter reaches eight counts, it outputs the start timing signal for data sampling. The RxD pin is sampled with this start timing signal again. If the pin is found to be at low level, it is recognized as a start bit, then initializing the 1/16 frequency division counter and causing it to start counting again for data sampling. When a start bit, data bits, parity bit, and a stop bit are detected, reception of one frame of data is completed.

When one frame of data is received, the receive data in the shift register is sent to the reception buffer (RXB), eventually generating a reception completion interrupt (INTSR).

If an error occurs, the erroneous receive data is sent to the RXB and causes an INTSR to be generated.

Resetting the RXE bit to 0 immediately stops reception. In this case, the contents of the RXB or ASIS are not affected, and neither INTSR nor INTSER is generated. Setting the RXE bit to 1 triggers sampling for a start bit.

Note Reception assumes there is only one stop bit, regardless of whether the SL bit of the ASIM register is 1.



Cautions 1. If the ASIM register is modified during reception, the current and next receive data may be damaged. When changing the mode, disable reception beforehand.

2. Be sure to read the reception buffer (RXB) contents, even if a reception error occurs. Otherwise, an overrun error will occur when the next data is received, and the error status will persist.

9.3.5 Reception Error

Three types of errors may occur during reception; parity error, framing error, and overrun error. If any one of these errors occurs, the corresponding error flag in the asynchronous serial interface register (ASIS) is set, and a reception error interrupt (INTSER) is generated. Table 9-1 lists causes of reception errors.

The type of the reception error can be identified by the reception error interrupt routine (INTSER), which reads and checks the contents of the asynchronous serial interface register (ASIS). (See Fig. 9-3 and Fig. 9-7.)

The ASIS is reset to 0 by reading data from the reception buffer (RXB) or receiving the next data. (If the next data again contains an error, the corresponding error flag will be set.)

★

9

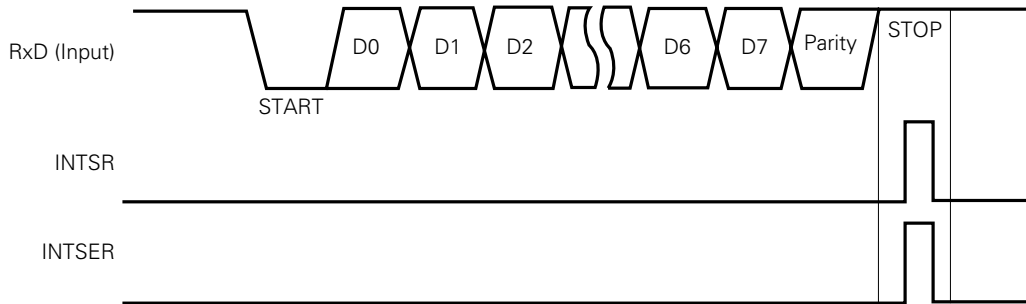
Table 9-1 Causes of Reception Errors

Reception error	Cause
Parity error	The parity of the receive data does not match the type of parity specified at transmission.
Framing error	No stop bit is detected ^{Note} .
Overrun error	Before the receive data is read out from the reception buffer, the next data is received.

★

Note Reception assumes that only one stop bit is used. Therefore, if the transmission end uses a two-stop bit format, and the second stop bit is low when received, no reception error is detected; it is recognized as the start bit of the next data.

Fig. 9-7 Reception Error Timing



Remark With the μPD78214, no break signal can be detected by hardware. Because a break signal consists of two characters' worth of low level, software identifies it by detecting that a framing error has occurred twice consecutively where the receive data is 00H. Software can differentiate the break signal from a sequence of two accidental framing errors. This is done by reading the level of the RxD pin (by reading port 3 (P3) by setting bit 0 of the port 3 mode register to 1) and checking if it is 0.

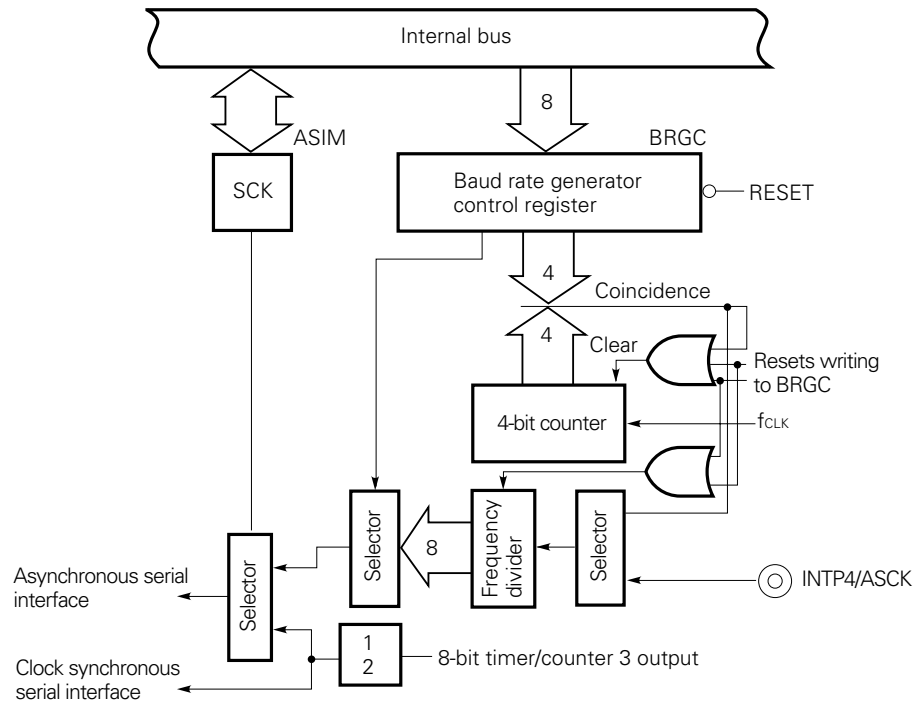
- Cautions**
1. The ASIS register is reset to 0 when the reception buffer (RXB) is read-accessed or receives the next data. To identify the error, be sure to check the ASIS before reading data from the RXB. If a macro service is used during reception, it is impossible to identify the error; it is only possible to know an error has occurred (an INTSER has occurred or the reception error interrupt request flag (SERIF) is set to 1). Make sure that this poses no problem for your application.
 2. Be sure to read the reception buffer (RXB) contents, even if a reception error occurs. Otherwise, an overrun error will occur when the next data is received, and the error status will persist.

9.4 BAUD RATE GENERATOR

9.4.1 Configuration of the Baud Rate Generator for UART

Fig. 9-8 shows the configuration of the baud rate generator.

Fig. 9-8 Baud Rate Generator Clock Configuration



(1) 4-bit counter

The 4-bit counter counts the internal system clock (f_{CLK}). It generates a signal having the frequency selected by the lower four bits of the baud rate generator control register (BRGC).

(2) Frequency divider

The frequency divider divides the signal input from the 4-bit counter or an external baud rate input (ASCK), and allows the selector at the next stage to select the clock for the baud rate.

(3) Both-edge detector

The both-edge detector detects either edge of the signal input to the ASCK pin and generates a signal having a frequency two times as high as the ASCK input clock frequency. See **Chapter 11** for details of edge detection.

9.4.2 Baud Rate Generator Control Register (BRGC)

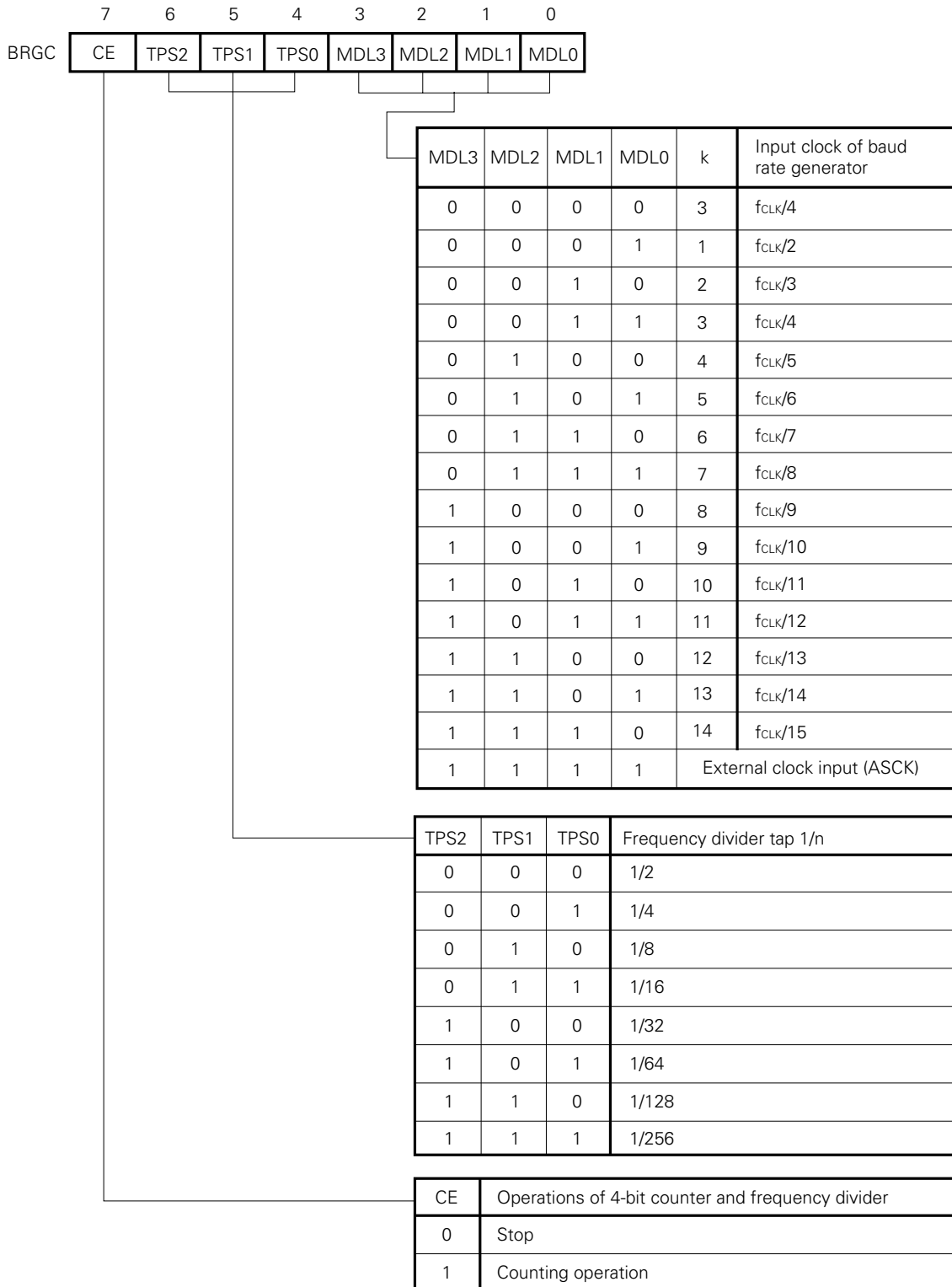
The BRGC register is an 8-bit register that holds the clock for baud rate generation controlled according to the internal system clock (f_{CLK}).

Only an 8-bit manipulation instruction can be used for this register, and its use is limited to write operations. Fig. 9-9 shows the format of the register.

When the \overline{RESET} signal is input, the BRGC register is reset to 00H.

Caution When a BRGC register write instruction is executed, the 4-bit counter and the frequency divider are reset. If the BRGC register is write-accessed during transmission, the baud rate being generated is disrupted, hampering normal communication. For this reason, do not write to the BRGC register during transmission.

Fig. 9-9 Baud Rate Generator Control Register (BRGC) Format



9.4.3 Operation of the Baud Rate Generator for UART

The baud rate generator for UART starts operating, when the CE bit of the baud rate generator control register (BRGC) is set to 1. The baud rate clock to be generated is a signal obtained by dividing either the internal system clock (f_{CLK}) or the clock input from the external baud rate input (ASCK) pin.

Resetting the CE bit to 0 stops the operation of the baud rate generator.

When the CE is 1, if an attempt is made to set it to 1 again, the 4-bit counter and frequency divider for baud rate generation are reset, then baud rate clock generation starts again.

Caution When a BRGC register write instruction is executed, the 4-bit counter and the frequency divider are reset. If the BRGC register is write-accessed during transmission, the baud rate being generated is disrupted, hampering normal communication. For this reason, do not write to the BRGC register during transmission.

(1) Generating the baud rate clock from the internal system clock (f_{CLK})

The internal system clock (f_{CLK}) is divided by the 4-bit counter. The resultant signal is further divided by the frequency divider to generate the baud rate clock.

The baud rate generated from the internal system clock (f_{CLK}) is determined by the following formula:

$$(\text{Baud rate}) = f_{CLK} / (k + 1) \times 1/n \times 1/16$$

where, f_{CLK} : Internal system clock frequency

k : Value set in the MDL3 to MDL0 bits of the BRGC register ($k = 1$ through 14; see **Fig. 9-9**.)

$1/n$: Frequency divider tap

16 : Serial data sampling rate

(2) Generating the baud rate clock from the ASCK input

Both edges of an input to the ASCK pin are detected to generate a clock having the frequency two times as high as the frequency at the ASCK pin. The resultant clock is then divided at the frequency divider. This function makes it possible to generate more than one baud rate from one external input clock.

The baud rate generated from the input to the ASCK pin is determined by the following formula:

$$(\text{Baud rate}) = f_{ASCK} \times 2/n \times 1/16$$

where, f_{ASCK} : ASCK input clock frequency

Note that the ASCK input cannot be higher than $f_{CLK}/24$ (250 kHz for $f_{CLK} = 6$ MHz).

9.5 BAUD RATE SETTING

The baud rate can be set by three methods listed in Table 9-2.

The table indicates the ranges of baud rates that can be generated by each method, the baud rate calculation formulas, and the selection methods.

Table 9-2 Baud Rate Setting

Baud rate clock source		Selection method		Calculation formula	Baud rate range
Baud rate generator for UART	Internal system clock	SCK of the ASIM register = 1	MDL0 through MDL3 of the BRGC register = 0H to EH	$\frac{f_{CLK}}{K + 1} \times \frac{1}{n} \times \frac{1}{16}$	$\frac{f_{CLK}}{61440} - \frac{f_{CLK}}{64}$
	ASCK input	CE of the BRGC register = 1	MDL0 through MDL3 of the BRGC register = FH	$f_{ASCK} \times \frac{2}{n} \times \frac{1}{16}$	$\frac{f_{ASCK}}{2048} - \frac{f_{ASCK}}{16}$ ^{Note}
8-bit timer/counter 3		SCK of the ASIM register = 0		$\frac{f_{CLK}}{2^{j+3}} \times \frac{1}{m+1} \times \frac{1}{16} \times \frac{1}{2}$	$\frac{f_{CLK}}{4194304} - \frac{f_{CLK}}{256}$

f_{CLK} : Internal system clock frequency

k : Value set in the MDL3 through MDL0 bits of the BRGC register ($k = 1$ through 14; see **Fig. 9-9.**)

$1/n$: Frequency divider tap ($n = 2, 4, 8, 16, 32, 64, 128, 256$)

f_{ASCK} : Frequency of the ASCK input clock ($0 - f_{CLK}/24$)

$1/16$: Serial data sampling rate

j : Value set in the PRS3 through PRS0 bits of prescaler mode register 0 ($j = 0$ through 6)

PRS3-PRS0	0H	1H	2H	3H	4H	5H	6H	7H
j	0	1	2	3	4	5	6	

m : Value set in the 8-bit compare register (CR30); $m = 0$ through 255

Note $0 - f_{CLK}/384$ if the f_{ASCK} input range is included.

9.5.1 Example of Setting the BRGC Register When the Baud Rate Generator for UART Is Used

This section shows examples of setting the BRGC register when the baud rate generator for UART is used.

To use the baud rate generator, set the SCK bit of the asynchronous serial interface mode register (ASIM) to 1.

Table 9-3 Example of Setting the BRGC Register When the Baud Rate Generator for UART Is Used

Oscillation frequency (f _{xx}) or external clock input (fx)	12 MHz		11.0592 MHz		10.0 MHz		9.8304 MHz		7.3728 MHz	
	BRGC value	Baud rate error (%)	BRGC value	Baud rate error (%)	BRGC value	Baud rate error (%)	BRGC value	Baud rate error (%)	BRGC value	Baud rate error (%)
Internal system clock (f _{CLK})	6 MHz		5.5296 MHz		5.0 MHz		4.9152 MHz		3.6864 MHz	
Baud rate [bps]	BRGC value	Baud rate error (%)	BRGC value	Baud rate error (%)	BRGC value	Baud rate error (%)	BRGC value	Baud rate error (%)	BRGC value	Baud rate error (%)
75	—	—	—	—	—	—	—	—	FBH	0.00
110	FCH	2.44	FBH	2.27	FAH	0.89	FAH	0.83	F7H	2.27
150	F9H	2.34	F8H	0.00	F7H	1.73	F7H	0.00	EBH	0.00
300	E9H	2.34	E8H	0.00	E7H	1.73	E7H	0.00	DBH	0.00
600	D9H	2.34	D8H	0.00	D7H	1.73	D7H	0.00	CBH	0.00
1200	C9H	2.34	C8H	0.00	C7H	1.73	C7H	0.00	BBH	0.00
2400	B9H	2.34	B8H	0.00	B7H	1.73	B7H	0.00	ABH	0.00
4800	A9H	2.34	A8H	0.00	A7H	1.73	A7H	0.00	9BH	0.00
9600	99H	2.34	98H	0.00	97H	1.73	97H	0.00	8BH	0.00
19200	89H	2.34	88H	0.00	87H	1.73	87H	0.00	85H	0.00
31250	92H	0.00	—	—	84H	0.00	84H	1.70	—	—
38400	84H	2.34	—	—	83H	1.73	83H	0.00	82H	0.00

9.5.2 Example of Setting the Baud Rate When 8-bit Timer/Counter 3 Is Used

Table 9-4 lists examples of setting the baud rate when 8-bit timer/counter 3 is used. When using 8-bit timer/counter 3, reset the SCK bit of the asynchronous serial interface mode register (ASIM) to 0.

See **Section 7.4** for how to use 8-bit timer/counter 3.

Table 9-4 Example of Setting the Baud Rate When 8-Bit Timer/Counter 3 Is Used (Asynchronous Serial Interface)

Oscillation frequency fosc (MHz)	12 MHz			11.0592 MHz			10.0 MHz			9.8304 MHz			7.3728 MHz		
	Count clock	m	Baud rate error (%)	Count clock	m	Baud rate error (%)	Count clock	m	Baud rate error (%)	Count clock	m	Baud rate error (%)	Count clock	m	Baud rate error (%)
Internal system clock (fCLK) MHz	6 MHz			5.5296 MHz			5.0 MHz			4.9152 MHz			3.6864 MHz		
Baud rate [bps]															
75	fCLK/16	155	0.16	fCLK/16	143	0.00	fCLK/16	129	0.16	fCLK/16	127	0.00	fCLK/16	191	0.00
110	fCLK/8	212	0.03	fCLK/8	195	0.00	fCLK/8	177	0.25	fCLK/8	174	0.26	fCLK/8	130	0.07
150	fCLK/8	155	0.16	fCLK/8	143	0.00	fCLK/8	129	0.16	fCLK/8	127	0.00	fCLK/8	95	0.00
300	fCLK/8	77	0.16	fCLK/8	71	0.00	fCLK/8	64	0.16	fCLK/8	63	0.00	fCLK/8	47	0.00
600	fCLK/8	38	0.16	fCLK/8	35	0.00	fCLK/8	32	1.36	fCLK/8	31	0.00	fCLK/8	23	0.00
1200	fCLK/8	19	2.34	fCLK/8	17	0.00	fCLK/8	15	1.73	fCLK/8	15	0.00	fCLK/8	11	0.00
2400	fCLK/8	9	2.34	fCLK/8	8	0.00	fCLK/8	7	1.73	fCLK/8	7	0.00	fCLK/8	5	0.00
4800	fCLK/8	4	2.34	—	—	—	fCLK/8	3	1.73	fCLK/8	3	0.00	fCLK/8	2	0.00
9600	—	—	—	—	—	—	fCLK/8	1	1.73	fCLK/8	1	0.00	fCLK/8	—	—
19200	—	—	—	—	—	—	fCLK/8	0	1.73	fCLK/8	0	0.00	fCLK/8	—	—

9.5.3 Example of Setting the BRGC When the External Baud Rate Input (ASCK) Is Used

Table 9-5 lists examples of setting the BRGC register when an external baud rate input (ASCK) is used. To use the ASCK input, set the SCK bit of the asynchronous serial interface mode register (ASIM) to 1.

Table 9-5 Examples of Setting the BRGC When an External Baud Rate Input (ASCK) Is Used

f _{ASCK} (ASCK input frequency)	153.6 kHz
Baud rate [bps]	BRGC value
75	FFH
150	EFH
300	DFH
600	CFH
1200	BFH
2400	AFH
4800	9FH
9600	8FH

9.6 NOTES

- (1) The asynchronous serial interface mode register (ASIM) must not be modified during transmission. If the ASIM register is modified during transmission, further transmission becomes impossible (inputting the RESET signal resumes normal operation).
Software can determine whether transmission is in progress, using the transmission completion interrupt (INTST) or the interrupt request flag (STIF), which is set by the INTST.
- (2) When the RESET signal is input, the transmission shift register becomes empty, but no transmission completion interrupt occurs. Transmission is triggered by writing the transmit data to the transmission shift register.
- (3) The ASIS register is reset to 0 when the reception buffer (RXB) is read-accessed or receives the next data. To identify the error, be sure to check the ASIS before reading data from the RXB. If a macro service is used during reception, it is impossible to identify the error; it is only possible to detect that an error has occurred (an INTSER has occurs or the reception error interrupt request flag (SERIF) is set to 1). Make sure that this poses no problem for your application.
- (4) If the ASIM register is modified during reception, the current and next receive data may become undefined. When changing the mode, disable reception beforehand.
- (5) Be sure to read the reception buffer (RXB) contents, even if a reception error occurs. Otherwise, an overrun error will occur when the next data is received, and the error status will persist.
- (6) Do not write to the BRGC register during communication. If a write instruction is executed for the BRGC register, the 4-bit counter and frequency divider are reset, and the baud rate clock generated is disturbed, hampering normal communication.

CHAPTER 10 CLOCK SYNCHRONOUS SERIAL INTERFACE

10.1 FUNCTION

The clock synchronous serial interface of the μ PD78214 is configured as shown in Fig. 10-1. The clock synchronous serial interface supports the following two operation modes:

(1) Three-wire serial I/O mode (MSB first)

Three lines, serial clock ($\overline{\text{SCK}}$) and serial bus lines (SO, SI), are used to transfer 8-bit data. This mode is suitable for connecting a display controller or peripheral I/O device having a conventional clock synchronous serial interface.

(2) Serial bus interface (SBI) mode (MSB first)

In this mode, the device can communicate with two or more devices via two lines, the serial clock ($\overline{\text{SCK}}$) and serial data bus line (SB0).

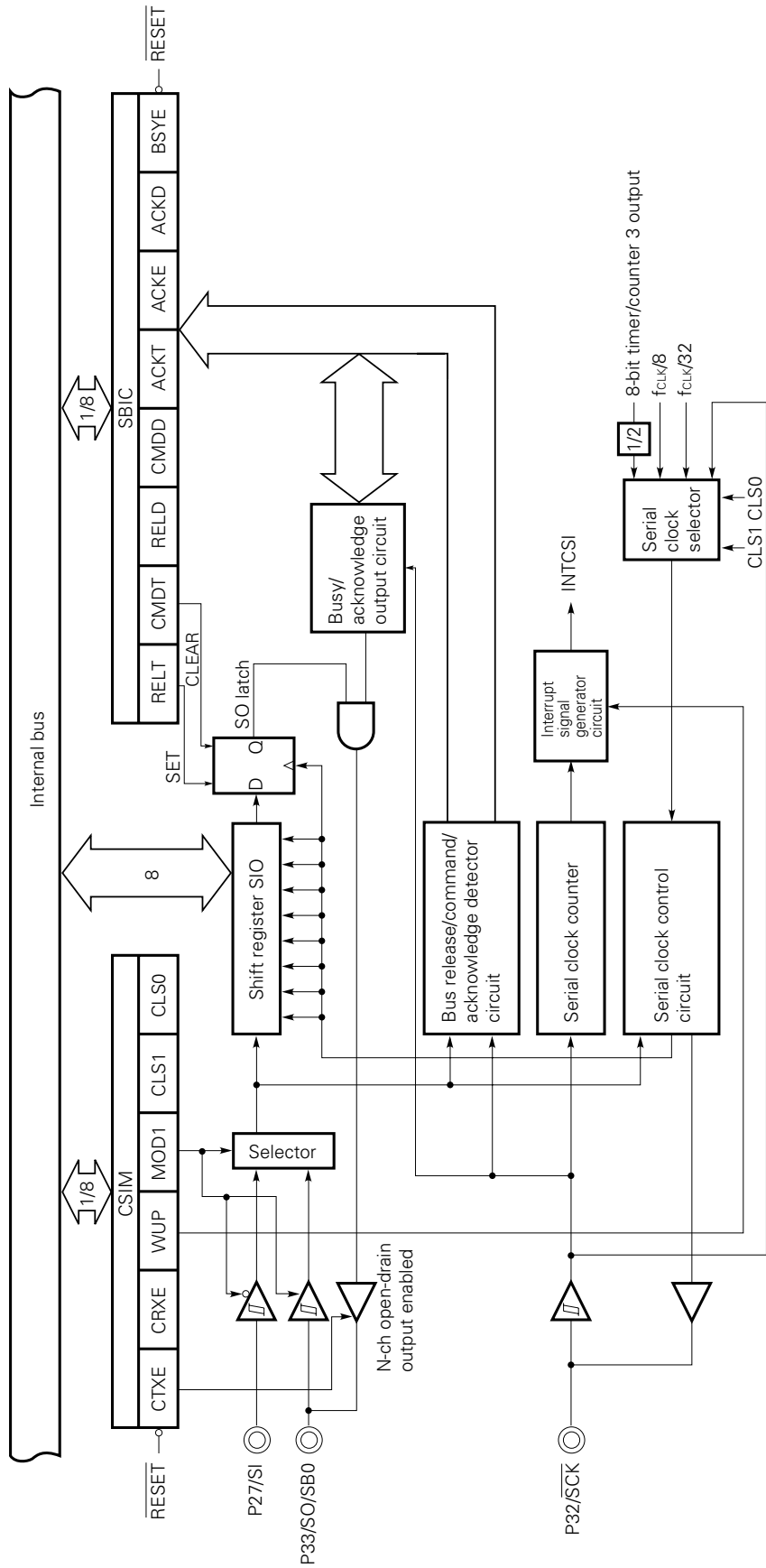
This mode conforms to the NEC serial bus format.

In SBI mode, an address for selecting a target device for serial communication, a command specifying the operation of the target device, and actual data can be output on the serial data bus. This mode eliminates the need for a handshaking line, which would otherwise be required to connect two or more devices through a conventional clock synchronous serial interface. The input/output ports can thus be used efficiently.

10.2 CONFIGURATION

This section describes the configuration of the clock synchronous serial interface.

Fig. 10-1 Block Diagram of the Clock Synchronous Serial Interface



(1) Shift register (SIO)

Converts 8-bit serial data into 8-bit parallel data and vice versa. The SIO is used for both transmission and reception.

Data is shifted in (received) or shifted out (transmitted) from the MSB.

The actual transmission/reception is controlled by writing or reading the contents of the SIO.

The 8-bit manipulation instruction can read or write the contents of this register. The contents become undefined when $\overline{\text{RESET}}$ is input.

(2) SO latch

Retains the output level of the SO/SB0 pin. In serial bus interface (SBI) mode, the software can directly control the latch.

(3) Serial clock selector

Selects the serial clock to be used.

(4) Serial clock counter

Counts the number of serial clock pulses output or input during transmission or reception and checks whether 8-bit data is transmitted or received.

(5) Interrupt signal generator

Controls whether an interrupt request is generated when the serial clock counter counts eight serial clock pulses. In three-wire serial I/O mode, an interrupt request is generated each time eight pulses are counted. In SBI mode, an interrupt request is generated whenever the conditions are satisfied.

(6) Serial clock controller

Controls the supply of the serial clock to the shift register. If the internal clock is used, the controller also controls the clock output to the $\overline{\text{SCK}}$ pin.

(7) Busy/acknowledge output circuit, bus release/command/acknowledge detector

Output and detect control signals in SBI mode. These circuits do not operate in three-wire serial I/O mode.

10.3 CONTROL REGISTERS

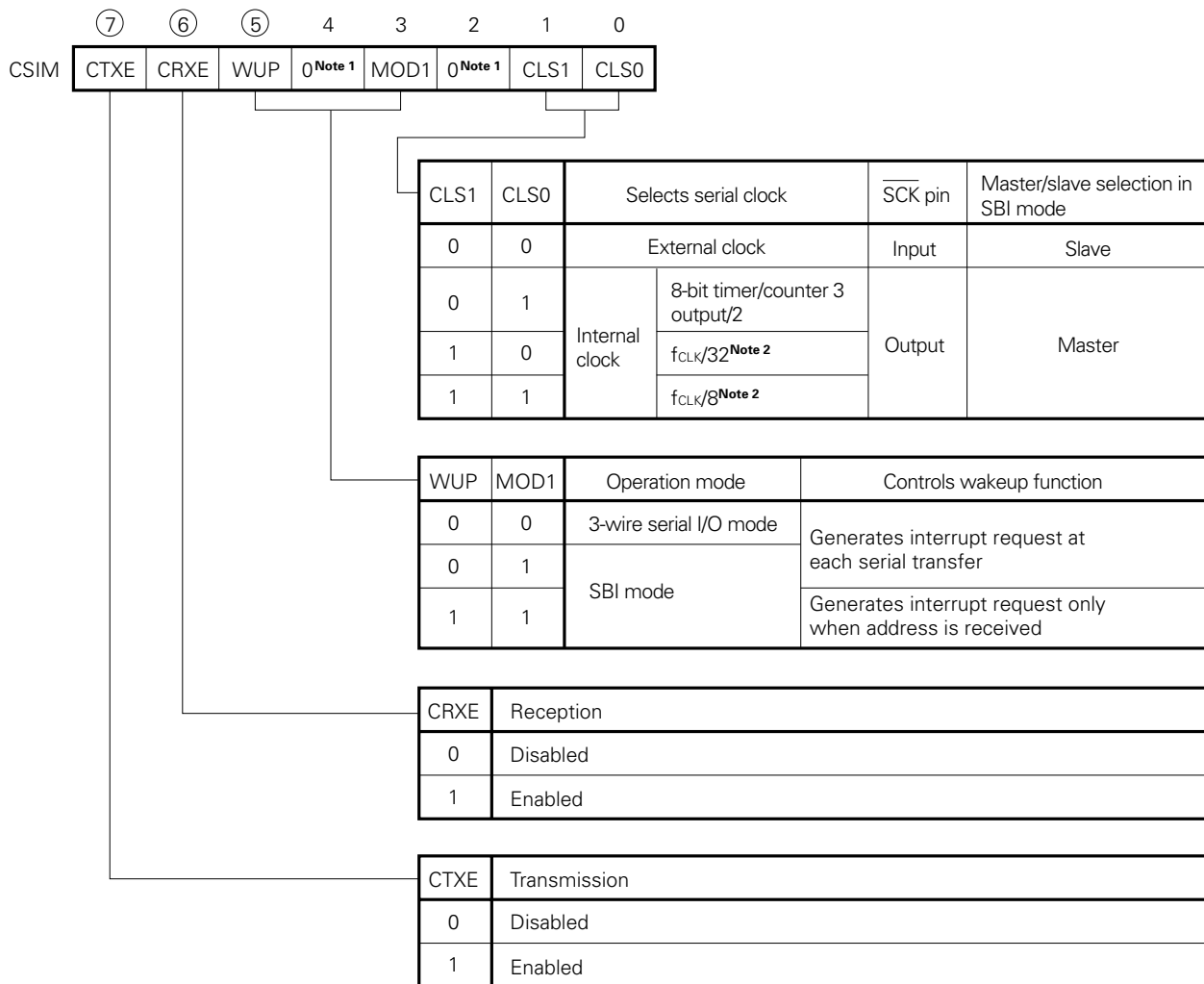
10.3.1 Clock Synchronous Serial Interface Mode Register (CSIM)

This 8-bit register specifies a serial interface operation mode, serial clock and wake-up function.

The 8-bit manipulation instruction and bit manipulation instruction can read and write the contents of the CSIM register. Fig. 10-2 shows the format.

The register is set to 00H when $\overline{\text{RESET}}$ is input.

Fig. 10-2 Format of the Clock Synchronous Serial Interface Mode Register (CSIM)



Notes 1. Always write 0 into bits 2 and 4.

2. f_{CLK} : Internal system clock

Caution Do not change CTXE from 0 to 1 and CRXE from 1 to 0, or vice versa, by means of a single instruction. If this is attempted, the serial clock counter will malfunction and the first communication after the change will be terminated before the eighth bit is sent. To change these statuses, use two instructions, as shown below:

Example Changing CTXE from 1 to 0 and CRXE from 0 to 1

CLR1 CTXE

SET1 CRXE

10.3.2 Serial Bus Interface Control Register (SBIC)

The SBIC register consists of bits that control the status of the serial bus, as well as bits that indicate the status of the data input from the serial bus. This 8-bit register can be used only in SBI mode, not in three-wire serial I/O mode.

The 8-bit manipulation instruction and bit manipulation instruction manipulate the contents of the register. These bits have different read/write attributes, listed in Table 10-1. When the contents are read, 0 is read from the write-only bits. The format is shown in Fig. 10-3.

The register is set to 00H when $\overline{\text{RESET}}$ is input.

Detection flags ACKD, CMDD, and RELD are cleared when transmission and reception are inhibited (both CTXE and CRXE are set to 0).

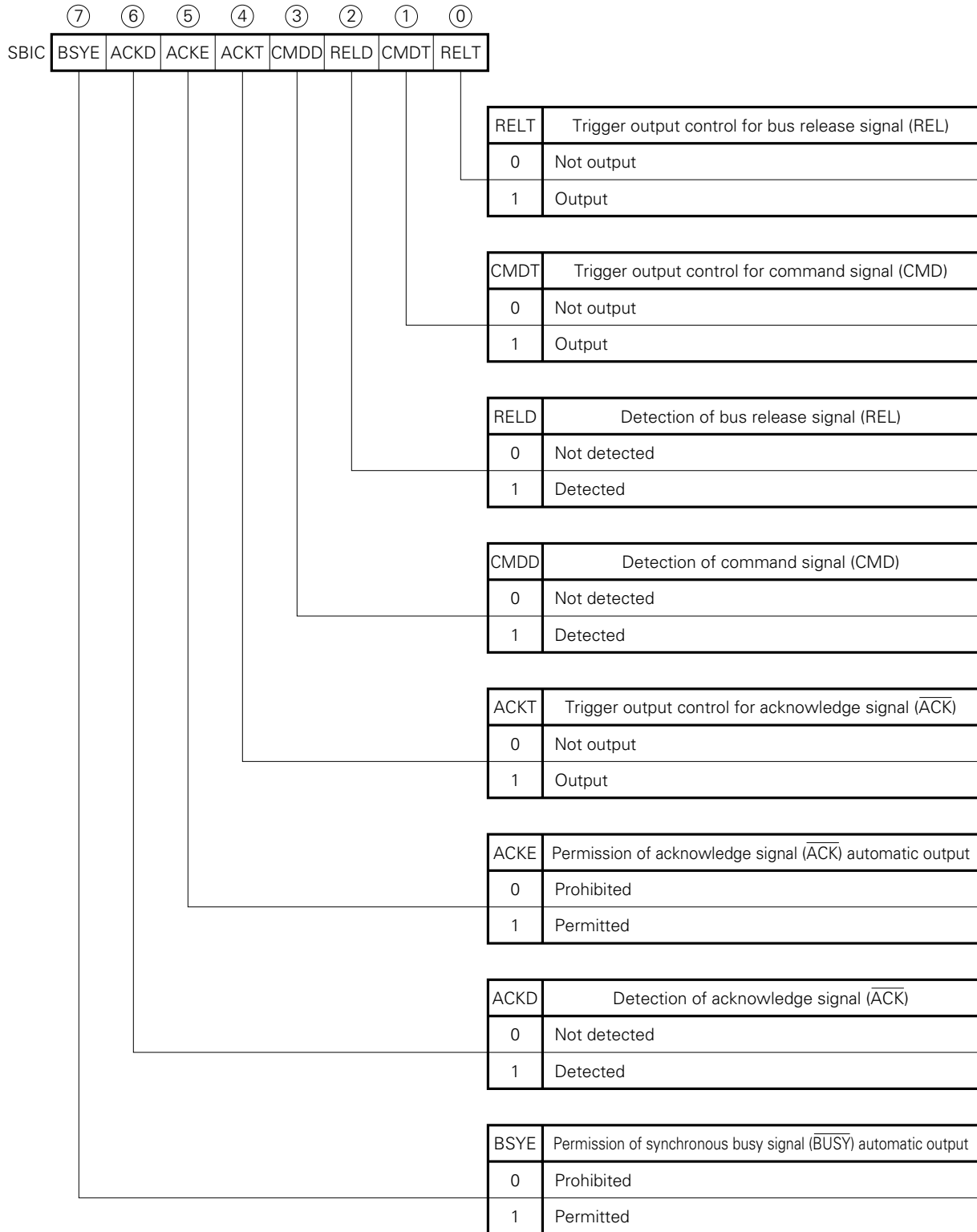
Table 10-1 Reading/Writing the Contents of the SBIC Register

	⑦	⑥	⑤	④	③	②	①	①
SBIC	BSYE	ACKD	ACKE	ACKT	CMDD	RELD	CMDT	RELT
	R/W	R	R/W	W	R	R	W	W

Remarks R/W : Read/write
 R : Read-only
 W : Write-only

Fig. 10-3 Format of Serial Bus Interface Control Register (SBIC)

★

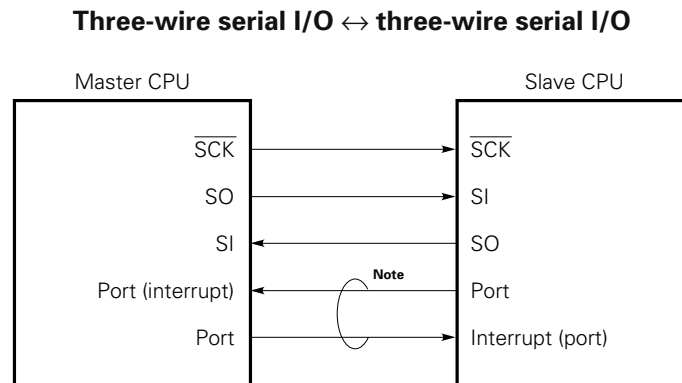


10.4 OPERATIONS IN THE THREE-WIRE SERIAL I/O MODE

In three-wire serial I/O mode, the device can communicate with a device having a conventional clock synchronous serial interface.

Basically, communication is performed over three lines of serial clock ($\overline{\text{SCK}}$), serial data output (SO) and serial data input (SI). A handshaking line is required to connect the device to two or more devices.

Fig. 10-4 Sample System Configuration with Three-Wire Serial I/O



Note Handshaking line

10

10.4.1 Basic Operation Timing

In three-wire serial I/O mode, data is transmitted and received in units of eight bits. The data is transmitted and received one bit at a time by means of the MSB-first method, in synchronization with the serial clock.

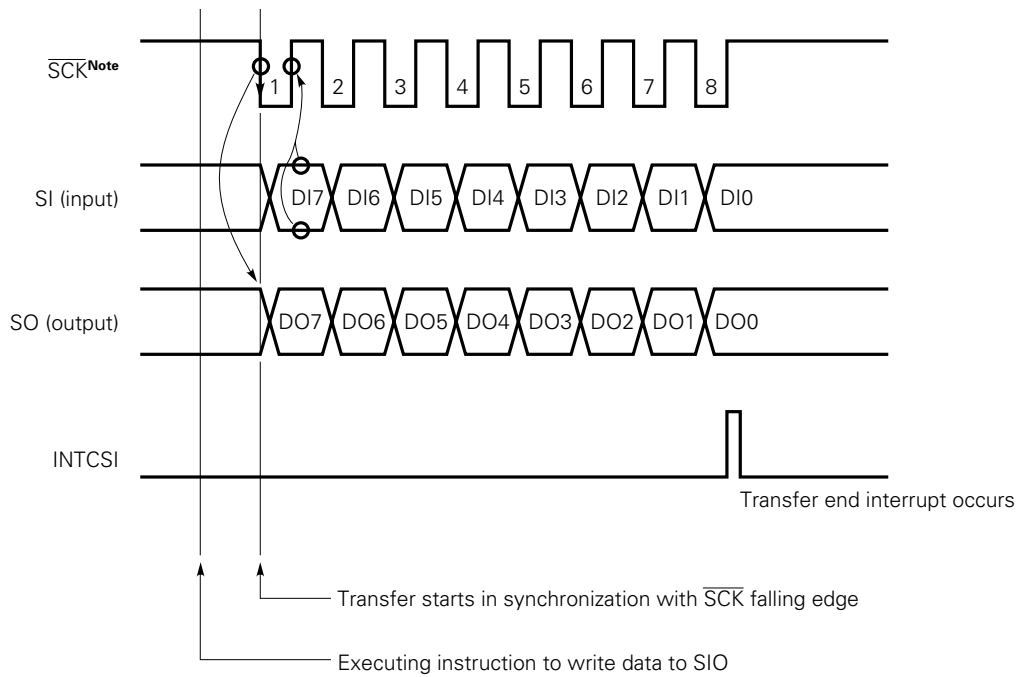
The transmission data is output in synchronization with the falling edge of $\overline{\text{SCK}}$. The reception data is sampled at the rising edge of $\overline{\text{SCK}}$.

At the eighth rising edge of $\overline{\text{SCK}}$, interrupt request INTCSI is issued.

If $\overline{\text{SCK}}$ is used as the internal clock, the output of $\overline{\text{SCK}}$ is stopped at the eighth rising edge of $\overline{\text{SCK}}$. $\overline{\text{SCK}}$ is tied high until transmission or reception of the subsequent data is started.

Fig. 10-5 shows the timing chart for three-wire serial I/O mode.

Fig. 10-5 Timing in Three-Wire Serial I/O Mode

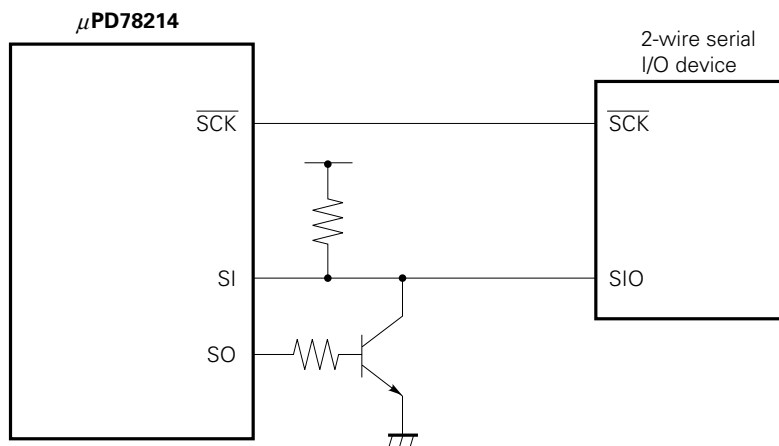


Notes Master CPU : Output
Slave CPU : Input

In three-wire serial I/O mode, the SO pin sends a CMOS push-pull output.

Remark When connecting the device to a device having two-wire serial I/O, connect a buffer to the SO pin as shown in Fig. 10-6. In the example shown in Fig. 10-6, the buffer inverts the output level. Invert the data to be output and write the inverted data into the SIO. Do not connect the built-in pull-up resistor to pin P33/SO.

Fig. 10-6 Sample Connection with a Device Having Two-Wire Serial I/O



If transmission and reception are time-shared, and if the μPD78214 can control the output level of the SIO pin of a device containing two-wire serial I/O, the SI pin and SO pin can be connected directly. To do this, disable transmission by other devices while the μPD78214 is transmitting data.

10.4.2 Operation When Only Transmission Is Permitted

Transmission is enabled when the CTXE bit of the clock synchronous serial interface mode register (CSIM) is set (1). If the CTXE bit is set, writing the contents of the shift register (SIO) invokes the start of transmission.

If the CTXE bit is reset (0), the output from the SO pin goes to the high-impedance state.

(1) Selecting the internal clock as the serial clock

When transmission is started, the serial clock is output from the $\overline{\text{SCK}}$ pin. At the same time, data is sequentially output from the SIO to the SO pin in synchronization with the falling edge of the serial clock. In synchronization with the rising edge of the serial clock, the signal from the SI pin is shifted into the SIO.

It takes up to one cycle of the $\overline{\text{SCK}}$ clock to drive $\overline{\text{SCK}}$ low for the first time after transmission is started.

If transmission is inhibited (the CTXE bit is reset to 0) during transmission, the output of the $\overline{\text{SCK}}$ clock is stopped and transmission is halted at the next rising edge of $\overline{\text{SCK}}$. At this time, no interrupt request (INTCSI) occurs. The output from the SO pin goes to the high-impedance state.

(2) Selecting the external clock as the serial clock

When transmission is started, data is sequentially output from the SIO to the SO pin in synchronization with the falling edge of the serial clock input to the $\overline{\text{SCK}}$ pin, after the start of transmission. At the same time, the signal from the SI pin is shifted into the SIO in synchronization with the rising edge of the input to the $\overline{\text{SCK}}$ pin. If the serial clock is input to the $\overline{\text{SCK}}$ pin before transmission has been started, no shift occurs. The output level of the SO pin does not change.

If transmission is inhibited (the CTXE bit is reset to 0) during transmission, the transmission is halted and any subsequent $\overline{\text{SCK}}$ input is ignored. At this time, no interrupt request (INTCSI) occurs. The output from the SO pin goes to the high-impedance state.

10.4.3 Operation When Only Reception Is Permitted

Reception is performed when the CRXE bit of the CSIM register is set (1). When the CRXE bit is changed from 0 to 1 or when the contents of the SIO are read, reception is started.

(1) Selecting the internal clock as the serial clock

When reception is started, the serial clock is output from the $\overline{\text{SCK}}$ pin. In synchronization with the rising edge of the serial clock, data is sequentially sent from the SI pin to the SIO.

It takes up to one cycle of the $\overline{\text{SCK}}$ clock to drive $\overline{\text{SCK}}$ low for the first time, after the reception is started.

If reception is inhibited (the CRXE bit is reset to 0) during reception, the output of the $\overline{\text{SCK}}$ clock is stopped and reception is halted at the next rising edge of $\overline{\text{SCK}}$. At this time, no interrupt request (INTCSI) is issued. The contents of the SIO become undefined.

(2) Selecting the external clock as the serial clock

When reception is started, data is sent sequentially from the SI pin to the SIO in synchronization with the rising edge of the serial clock input to the $\overline{\text{SCK}}$ pin after the start of reception. If the serial clock is input to the $\overline{\text{SCK}}$ pin before reception has been started, no shift occurs.

If reception is inhibited (the CRXE bit is reset to 0) during reception, the reception is halted and subsequent $\overline{\text{SCK}}$ input is ignored. At this time, no interrupt request (INTCSI) is issued.

10.4.4 Operation When Both Transmission and Reception Are Permitted

Transmission and reception can be simultaneously performed when both the CTXE bit and CRXE bit of the CSIM register are set (1) (transmission and reception). Transmission and reception are started when the CRXE bit is changed from 0 to 1 or when the contents of the SIO are written.

When transmission and reception are started for the first time, the CRXE bit is changed from 0 to 1. Because transmission and reception are started immediately, undefined data may be output. To prevent this, write the first transmission data into the SIO while both transmission and reception are inhibited (the CTXE and CRXE bits are both reset to 0), then permit transmission and reception.

If transmission and reception are inhibited (both CTXE and CRXE are set to 0), the output from the SO pin goes to the high-impedance state.

(1) Selecting the internal clock as the serial clock

When transmission and reception are started, the serial clock is output from the $\overline{\text{SCK}}$ pin. In synchronization with the falling edge of the serial clock, data is sequentially output from the SIO to the SO pin. In synchronization with the rising edge of the serial clock, data is sequentially shifted in from the SI pin to the SIO.

It takes up to one cycle of the $\overline{\text{SCK}}$ clock to drive $\overline{\text{SCK}}$ low for the first time, after the transmission and reception are started.

If either transmission or reception is inhibited during transmission and reception, only the inhibited operation is halted. If transmission is inhibited, the output from the SO pin goes to the high-impedance state. If reception is inhibited, the contents of the SIO register become undefined.

If transmission and reception are simultaneously inhibited, the output of the $\overline{\text{SCK}}$ clock is stopped and transmission and reception are halted at the next rising edge of $\overline{\text{SCK}}$. If this occurs, the contents of the SIO become undefined. No interrupt request (INTCSI) is issued. The output from the SO pin goes to the high-impedance state.

(2) Selecting an external clock as the serial clock

When transmission and reception are started, data is sequentially output from the SIO to the SO pin in synchronization with the falling edge of the serial clock input to the $\overline{\text{SCK}}$ pin, after the start of the transmission and reception. In synchronization with the rising edge of the serial clock, data is sequentially shifted in from the SI pin to the SIO. If the serial clock is input to the $\overline{\text{SCK}}$ pin before transmission and reception have been started, shift into the SIO does not occur. The output level of the SO pin does not change.

If either transmission or reception is inhibited during transmission and reception, only the inhibited operation is halted. If transmission is inhibited, the output from the SO pin goes to the high-impedance state. If reception is inhibited, the contents of the SIO become undefined.

If transmission and reception are simultaneously inhibited, transmission and reception are halted and the subsequent $\overline{\text{SCK}}$ input is ignored. If this occurs, the contents of the SIO become undefined. No interrupt request (INTCSI) is issued. The output from the SO pin goes to the high-impedance state.

10.4.5 Action to Be Taken When the Serial Clock and Shift Become Asynchronous

If the external clock is selected as the serial clock, the serial clock and shift may become asynchronous because of noise. If this occurs, inhibit both transmission and reception (reset the CTXE and CRXE bits to 0). This initializes the serial clock counter. Then, the shift and serial clock are synchronized again at the first serial clock pulse input, after either transmission or reception is permitted.

10.5 SBI MODE

SBI (serial bus interface) is a high-speed serial interface conforming to the NEC serial bus format.

SBI is a high-speed serial bus with a single master, consisting of a clock synchronous serial I/O and a bus configuration function. In SBI mode, the device can communicate with two or more devices over two signal lines. If the serial bus is configured with two or more microcomputers and peripheral ICs, the number of ports and lines on the board can be reduced.

For details of the SBI functions, refer also to “**Serial Bus Interface (SBI) User’s Manual (IEM-1303)**”.

10.5.1 Features of SBI

Conventional serial I/O supports only data transfer. If the serial bus is configured with two or more devices, many ports and lines are required for the chip select signal, the separation of commands from data, and judgment of the busy state. If they are controlled by the software, an excessive load will be applied to the software.

If SBI is used, two signal lines, serial clock $\overline{\text{SCK}}$ and serial data bus line SB0, can form the serial bus. The number of ports for the microcomputers and lines on the printed circuit board can thus be reduced.

The SBI functions are described below:

(1) Function to separate address, command, and data

This function separates serial data into an address, command, and data.

(2) Function to select a chip by its address

The master sends an address to select a slave chip.

(3) Wake-up function

Using the wake-up function (which can be set or released by software), a slave device can easily detect whether it receives the address (chip select).

If the wake-up function is set, a serial reception interrupt (INTCSI) occurs only when the address is received.

While the master device is communicating with two or more devices, the CPUs of those slave devices which are not selected can operate, independent of the serial communication.

(4) Acknowledge signal ($\overline{\text{ACK}}$) control function

This function controls the acknowledge signal to check whether the serial data has been received.

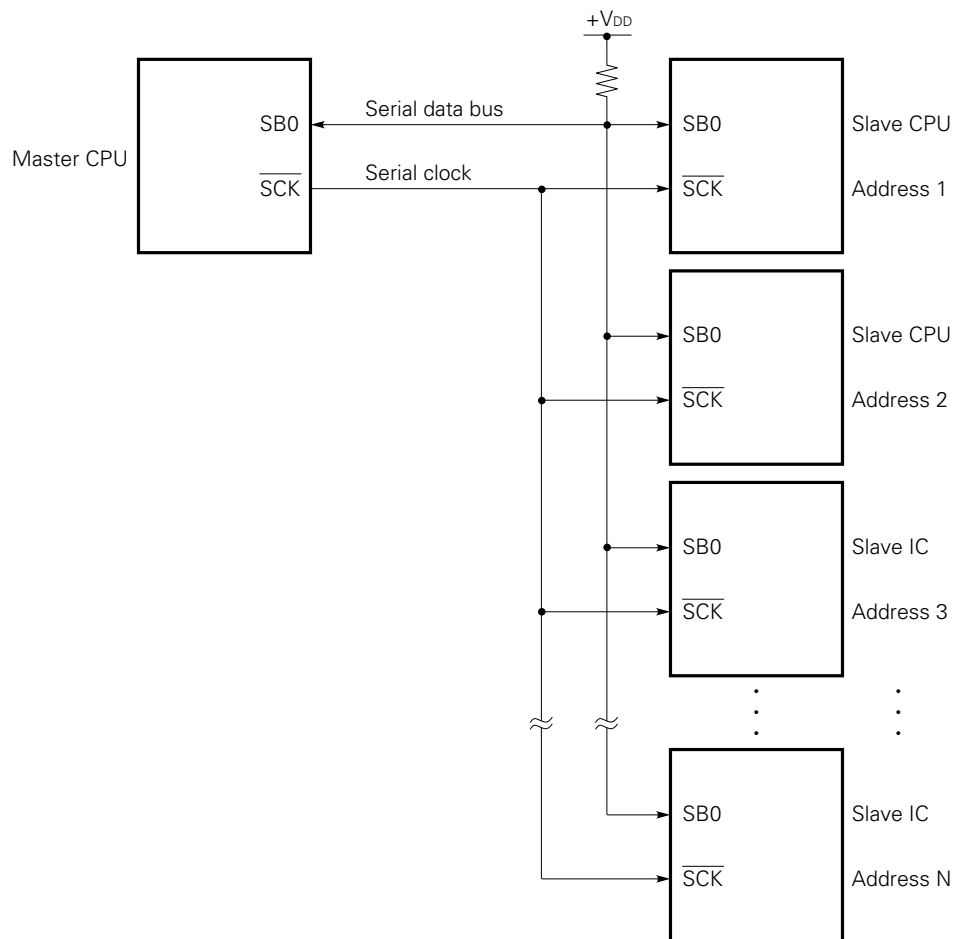
(5) Busy signal ($\overline{\text{BUSY}}$) control function

This function controls the busy signal that indicates that a slave device is busy.

Fig. 10-7 shows a sample serial bus configured with CPUs having a serial interface conforming to SBI and peripheral ICs.

In SBI mode, serial data bus pin SB0 functions as an open-drain output pin. The serial data bus lines are wired-ORed. The serial data bus line requires a pull-up resistor.

Fig. 10-7 Sample Serial Bus Configured with SBI



Caution When switching the master and slave, the input and output of the serial clock line ($\overline{\text{SCK}}$) are asynchronously switched between the master and slave. The serial clock line ($\overline{\text{SCK}}$) requires a pull-up resistor.

10.5.2 Configuration of the Serial Interface

Fig. 10-9 is a block diagram of the μPD78214.

The serial clock pin ($\overline{\text{SCK}}$) and serial data bus pin SB0 are configured as shown in Fig. 10-8.

(1) $\overline{\text{SCK}}$: Pin to input/output the serial clock

- Master : CMOS push-pull output
- Slave : Schmitt input

(2) SB0: Input/output pin for serial data

For both master and slave, N-ch open-drain output or Schmitt input

The serial data bus line requires an external pull-up resistor because of the N-ch open-drain output pin.

Fig. 10-8 Pin Configuration

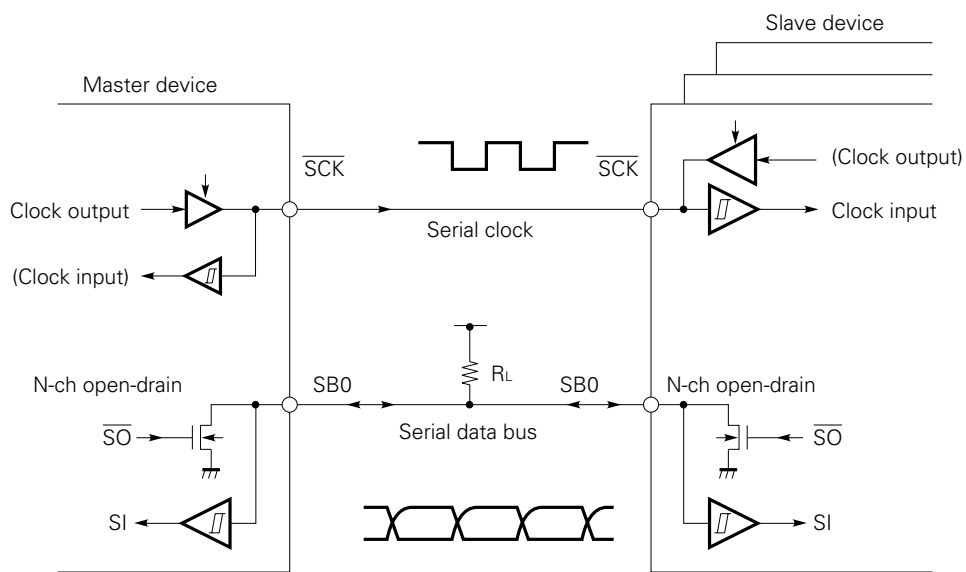
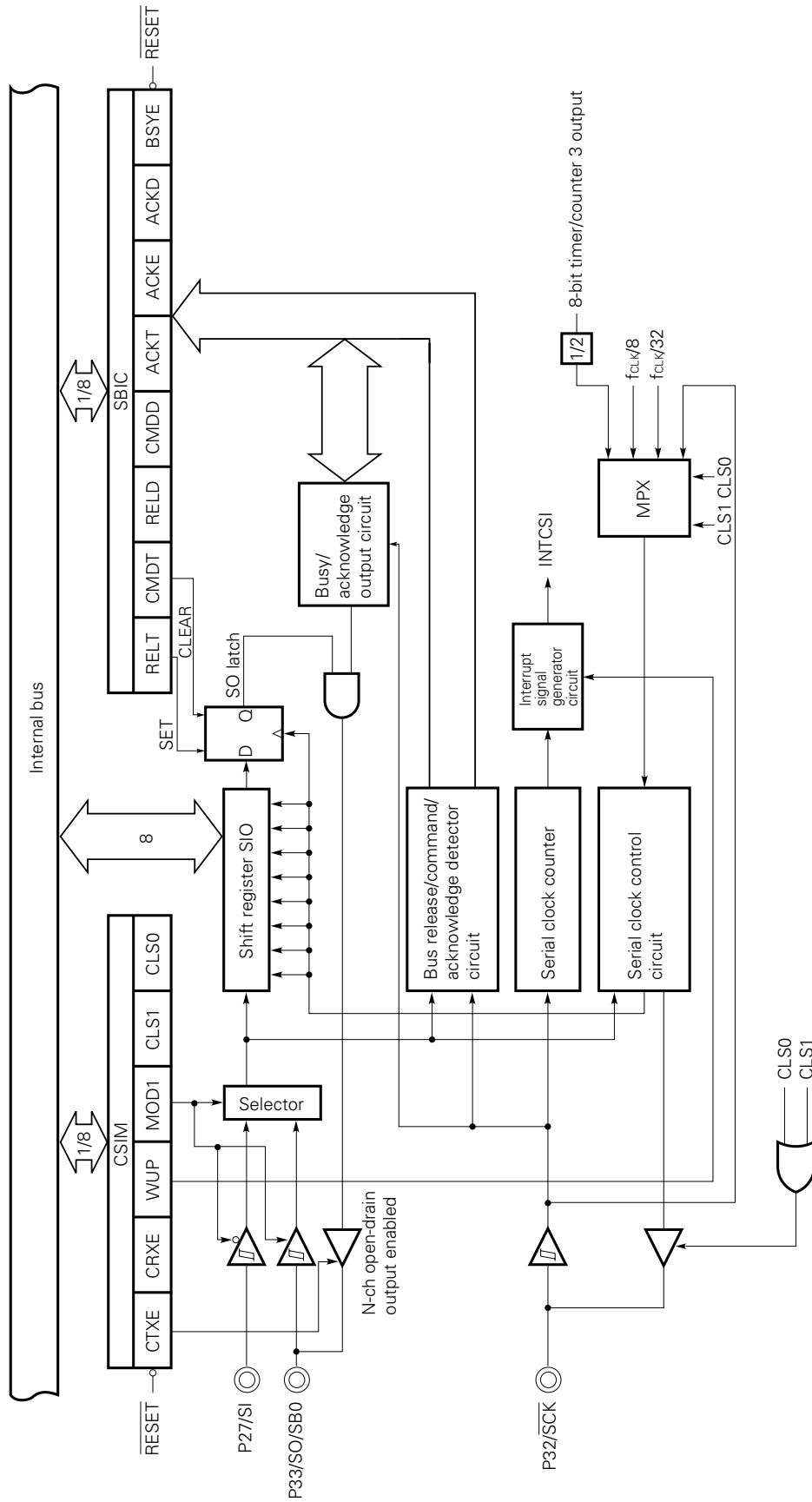


Fig. 10-9 Block Diagram of Clock Synchronous Serial Interface



10.5.3 Detecting an Address Match

SBI communication is started when a slave device is selected according to the address sent by the master device. The software detects whether the address of a slave device matches the sent address. In the wake-up state (WUP set to 1), the slave device generates a serial transfer completion interrupt request only when it receives the address. Upon detecting the address match, the software releases the wake-up state (sets WUP to 0) and prepares for the reception of the subsequent command and data.

10.5.4 Control Registers in SBI Mode

(1) Clock synchronous serial interface mode register (CSIM)

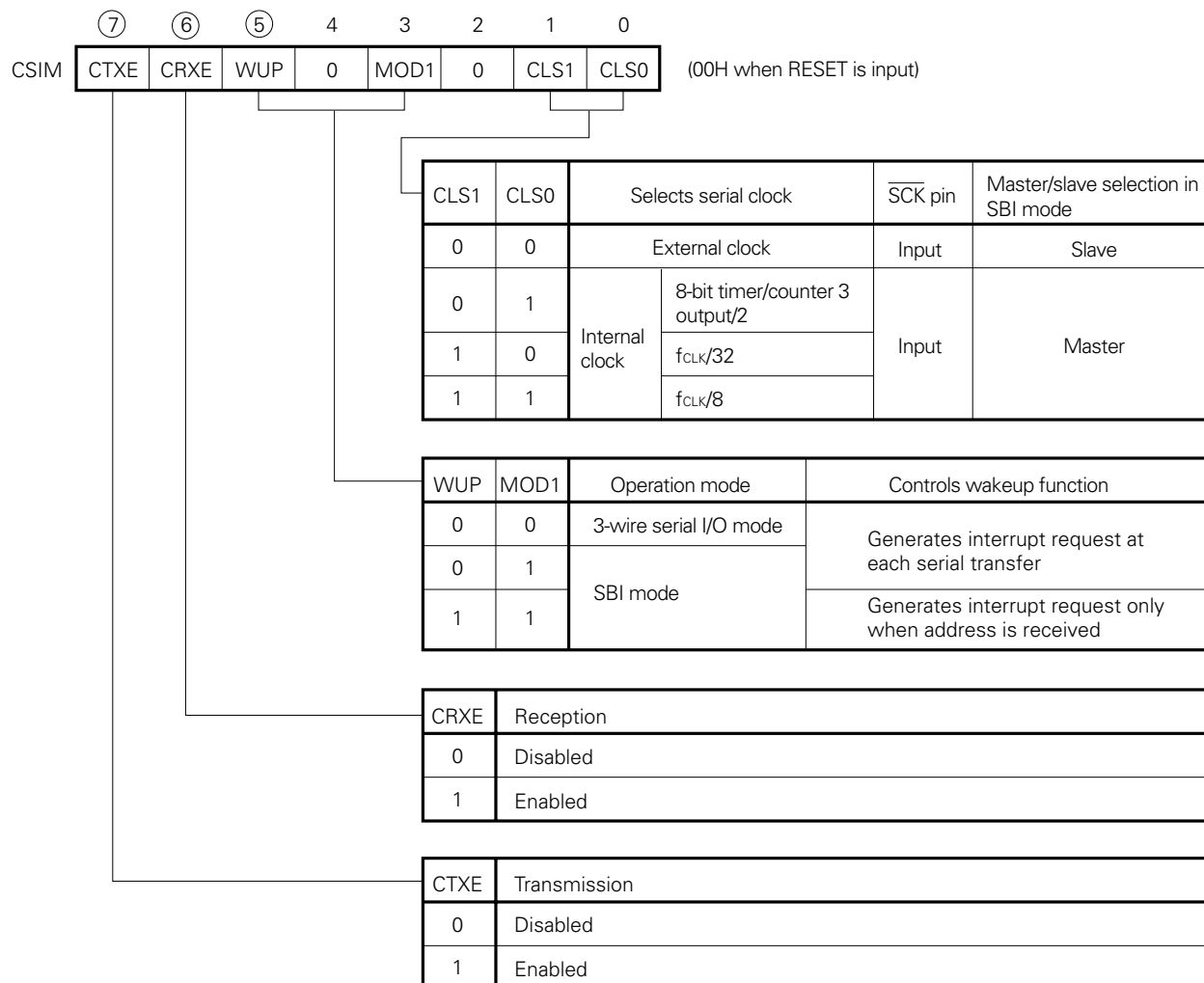
This 8-bit register specifies a serial interface operation mode, serial clock, and wake-up function.

Fig. 10-10 shows the format of the CSIM register.

The 8-bit manipulation instruction and bit manipulation instruction can read and write the contents of the register.

The bits of the register have different read/write attributes.

The value of the CSIM register is set to 00H when $\overline{\text{RESET}}$ is input.

Fig. 10-10 Format of Clock Synchronous Serial Interface Mode Register (CSIM)

Caution Do not change CTXE from 0 to 1 or CRXE from 1 to 0, or vice versa, by means of a single instruction. If this is attempted, the serial clock counter will malfunction and the first communication after the change will be terminated before the eighth bit is sent. To change those statuses, use two instructions as shown below:

Example Changing CTXE from 1 to 0 and CRXE from 0 to 1

```
CLR1 CTXE
SET1 CRXE
```

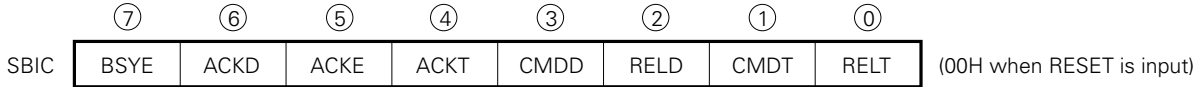
(2) Serial bus interface control register (SBIC)

This 8-bit register consists of bits controlling the serial bus statuses and flags indicating the statuses of data input from the serial bus.

The 8-bit manipulation instruction and bit manipulation instruction can read and write the contents of the register. The bits have different read/write attributes. Fig. 10-11 shows the format.

The value of the SBIC register is set to 00H when $\overline{\text{RESET}}$ is input.

Fig. 10-11 Format of SBIC Register (1/2)



Bus release trigger bit (W)

RELT	Trigger output control bit for the bus release signal (REL). When this bit is set, the SO latch is set to 1, then the RELT bit is automatically cleared to 0.
------	---

Command trigger bit (W)

CMDT	Trigger output control bit for the command signal (CMD). When this bit is set, the SO latch is cleared to 0, then the CMDT bit is automatically cleared to 0.
------	---

Bus release detection flag (R)

RELD	Clearing condition (RELD = 0)	Setting condition (RELD = 1)
	<ul style="list-style-type: none"> ① When transfer start instruction is executed ② When RESET signal is input ③ CTXE = CRXE = 0 	When bus release signal (REL) is detected

Command detection flag (R)

CMDD	Clearing condition (CMDD = 0)	Setting condition (CMDD = 1)
	<ul style="list-style-type: none"> ① When transfer start instruction is executed ② When bus release signal (REL) is detected ③ When RESET signal is input ④ CTXE = CRXE = 0 	When command signal (CMD) is detected

Fig. 10-11 Format of SBIC Register (2/2)

Acknowledge trigger bit (W)

ACKT	<p>When this bit is set after transfer, \overline{ACK} is output in synchronization with the next \overline{SCK}. After the \overline{ACK} signal is output, this bit is automatically cleared to 0.</p> <p>Cautions:</p> <ul style="list-style-type: none"> ① Do not set this bit to 1 before serial transfer is completed. ② ACKT cannot be cleared by software. ③ Set ACKT when ACKE = 0.
------	--

Acknowledge enable bit (R/W)

ACKE	0	Disables automatic output of acknowledge signal	
	1	Before transfer	Outputs \overline{ACK} in synchronization with the 9th \overline{SCK}
		After transfer	Outputs \overline{ACK} in synchronization with \overline{SCK} immediately after set instruction execution

Acknowledge detection flag (R)

ACKD	Clearing condition (ACKD = 0)		Setting condition (ACKD = 1)
	<ul style="list-style-type: none"> ① When \overline{SCK} falls first time after releasing busy after the transfer start instruction has been executed ② When RESET signal is input ③ CTXE = CRXE = 0 ④ When bus release is detected (in slave mode only) 	When acknowledge signal (\overline{ACK}) is detected	

Busy enable bit (R/W)

BSYE	0	<ul style="list-style-type: none"> ① Disables automatic output of busy signal ② Stops busy signal output in synchronization with \overline{SCK} falling immediately after clear instruction execution
	1	Outputs busy signal in synchronization with \overline{SCK} falling after acknowledge signal

Remarks (R) : Read-only
 (W) : Write-only
 (R/W): Read/write

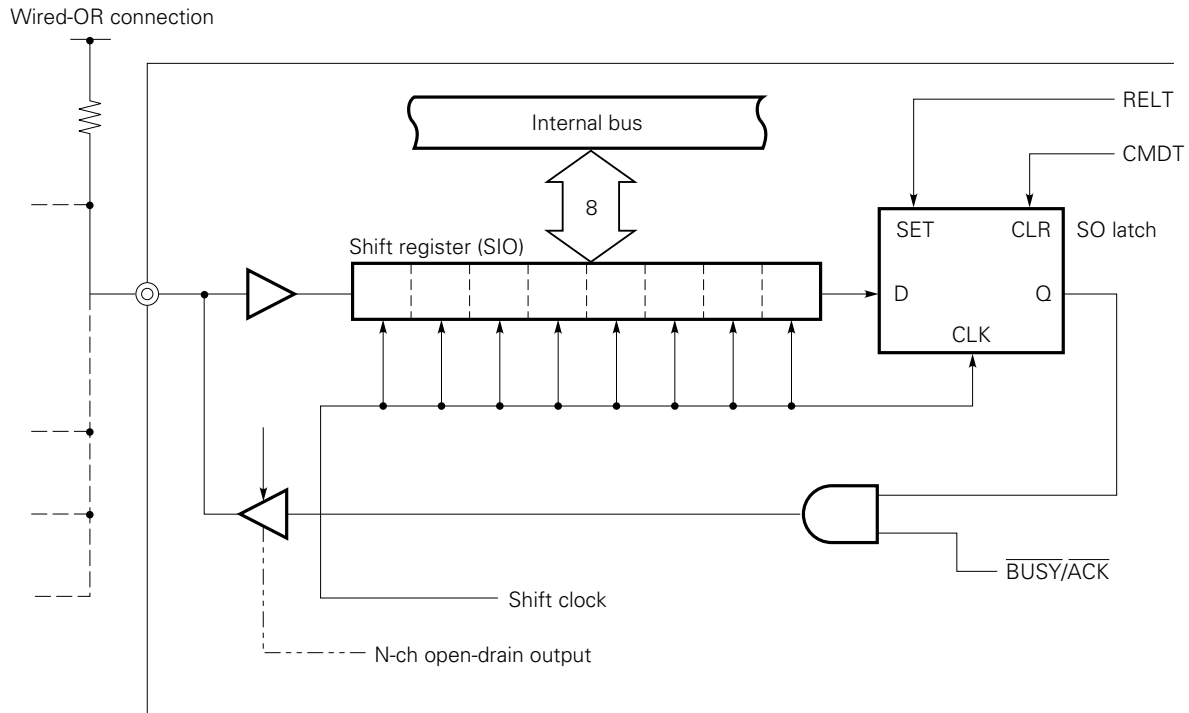


(3) Shift register (SIO)

This 8-bit shift register is used for parallel-serial conversion.

The data written into the SIO is output to the serial data bus. The data on the serial data bus is read into the SIO. Fig. 10-12 shows the configuration of the shift register and related components.

Fig. 10-12 Configuration of Shift Register and Related Components



In the SBI data bus configuration, the same pins are used for both input and output. The output pin functions as an N-ch open-drain output pin. With an external pull-up resistor, the output pin has a wired-OR configuration. For a device that is going to receive data, set the shift register (SIO) to FFH. Alternatively, disable transmission by the device.

10.6 SBI COMMUNICATION AND SIGNALS

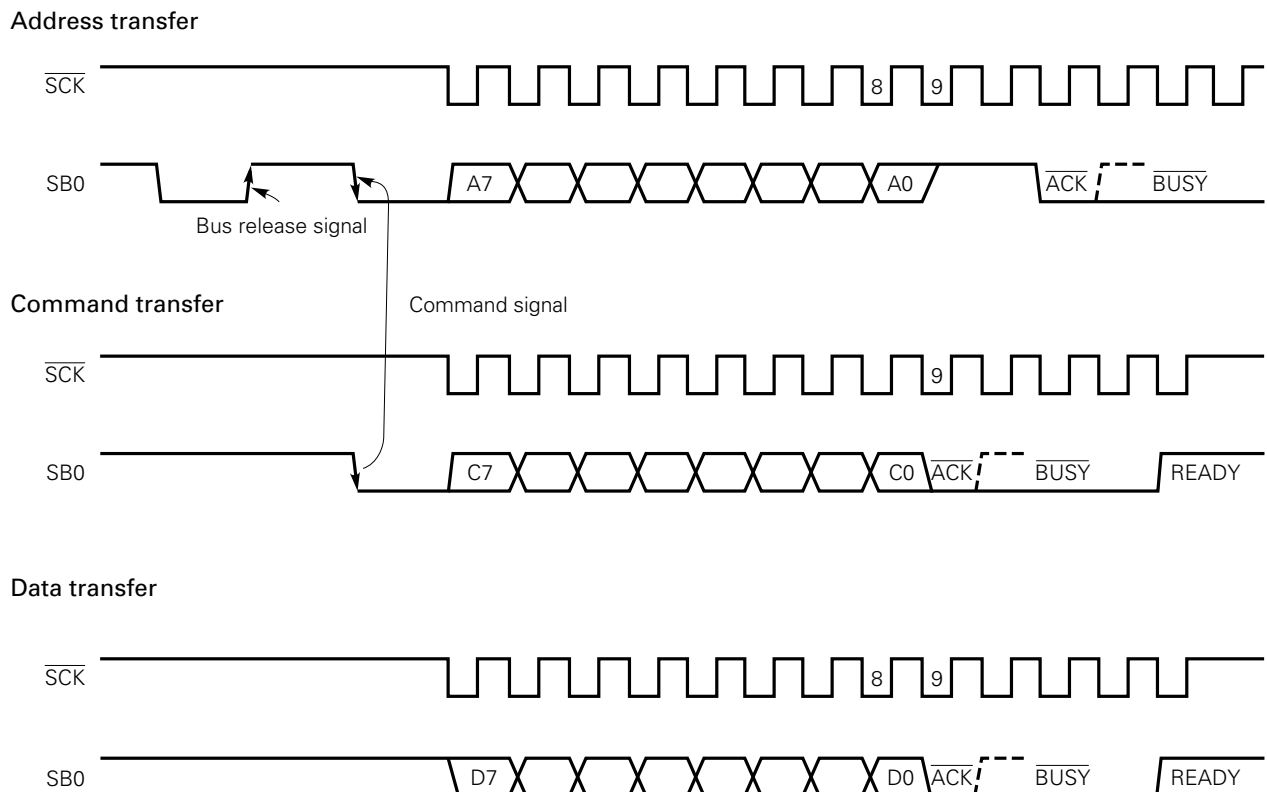
This section describes the format of the SBI serial data and signals to be used.

Serial data transferred via SBI can be divided into three groups: address, command, and data. Each frame of serial data is formed as shown below:

$$\text{(bus release signal) + (command signal) + 8-bit data + } \overline{\text{ACK}} \text{ + } (\overline{\text{BUSY}})$$

Fig. 10-13 shows the transfer timing of the address, command, and data.

Fig. 10-13 SBI Transfer Timing



10

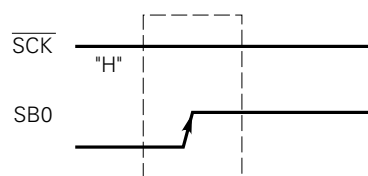
The master device outputs the bus release signal and command signal. The slave device outputs $\overline{\text{BUSY}}$. Either the master or slave device can output $\overline{\text{ACK}}$. (Usually, the device receiving 8-bit data outputs $\overline{\text{ACK}}$.)

The master device continues serial clock output during the period from the start of 8-bit data transfer to the release of $\overline{\text{BUSY}}$.

10.6.1 Bus Release Signal (REL)

The bus release signal is the SB0 line going from low to high while the $\overline{\text{SCK}}$ line is high (the serial clock is not output). The master device outputs this signal.

Fig. 10-14 Bus Release Signal

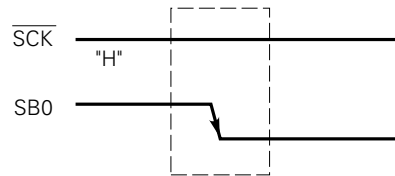


The bus release signal indicates that the master device is going to send an address to a slave device. The slave device contains hardware to detect the bus release signal.

10.6.2 Command Signal (CMD)

The command signal is the SB0 line going from high to low while the $\overline{\text{SCK}}$ line is high (the serial clock is not output). The master device outputs this signal.

Fig. 10-15 Command Signal

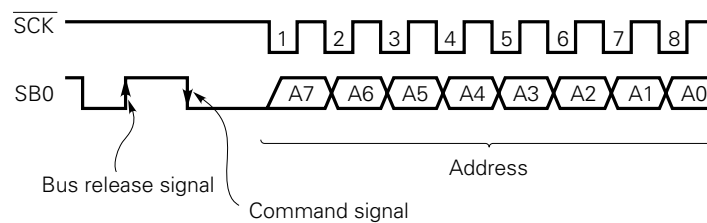


The slave device contains the hardware to detect the command signal.

10.6.3 Address

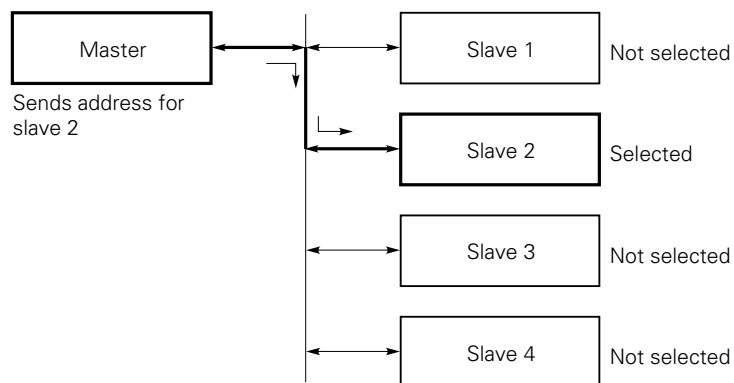
The master device outputs an address, which is 8-bit data, to select one of the slave devices connected to the bus line.

Fig. 10-16 Address



The 8-bit data output following the bus release signal and command signal is defined as an address. In a slave device, the hardware detects the condition and the hardware or software checks whether the 8-bit data matches its own number (slave address). If the 8-bit data matches a slave address, the slave device is selected. This slave device communicates with the master device until the slave is disconnected from the master.

Fig. 10-17 Selecting a Slave Device by Its Address



10.6.4 Command and Data

The master device sends commands to, and sends or receives data to or from, the slave device selected according to the specified address.

Fig. 10-18 Command

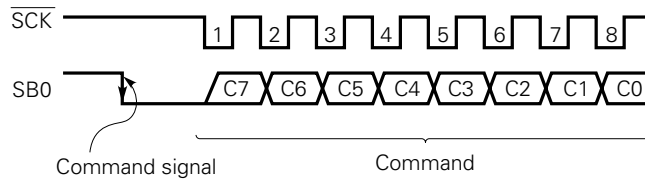
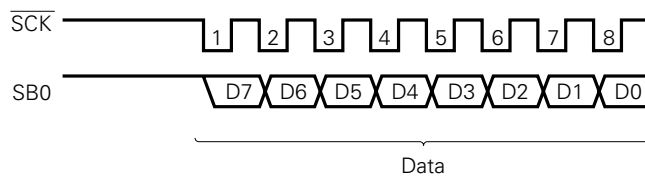


Fig. 10-19 Data



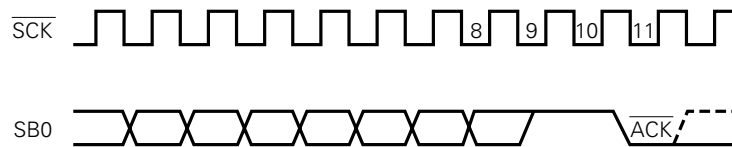
The 8-bit data following the command signal is defined as a command. The 8-bit data without the command signal is defined as data. The use of the command and data can be determined according to the communication specifications.

10.6.5 Acknowledge Signal ($\overline{\text{ACK}}$)

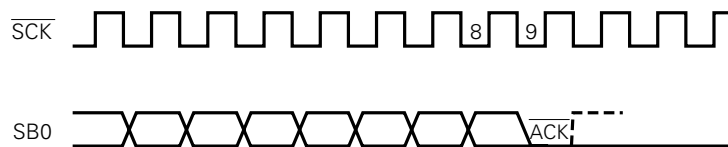
The transmitting device and receiving device use the acknowledge signal to check whether the data is successfully received.

Fig. 10-20 Acknowledge Signal

[Output in synchronization with the eleventh pulse of the $\overline{\text{SCK}}$ clock]



[Output in synchronization with the ninth pulse of the $\overline{\text{SCK}}$ clock]



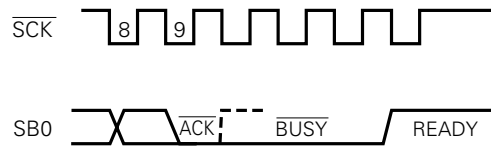
The acknowledge signal is a one-shot pulse, synchronized with the falling edge of $\overline{\text{SCK}}$ after the 8-bit data is transferred. The signal can be synchronized with the $\overline{\text{SCK}}$ clock pulse at a desired position.

After sending the 8-bit data, the transmitting device checks whether the receiving device returns the acknowledge signal. If the receiving device does not return the acknowledge signal within a certain period after the data is sent, the device has not successfully received the data.

10.6.6 Busy Signal ($\overline{\text{BUSY}}$) and Ready Signal (READY)

The busy signal informs the master device that the slave device is preparing for data transmission or reception. The ready signal informs the master device that the slave device is ready for data transmission or reception.

Fig. 10-21 Busy Signal and Ready Signal



In SBI mode, the slave device drives the SB0 line low to inform the master device that the slave is busy. The busy signal is output after the acknowledge signal is output by the master or slave device. The busy signal is set or released in synchronization with the falling edge of SCK. When the busy signal is released, the master device automatically terminates the output of the SCK serial clock. The master device can start the next transfer when the busy signal is released and the ready signal is set.

10.6.7 Signals

Figs. 10-22 to 10-26 show the signals in SBI mode and the flag operations of the SBIC. Table 10-2 lists the SBI signals.

Fig. 10-22 Operation of RELT, CMDT, RELD, and CMDD

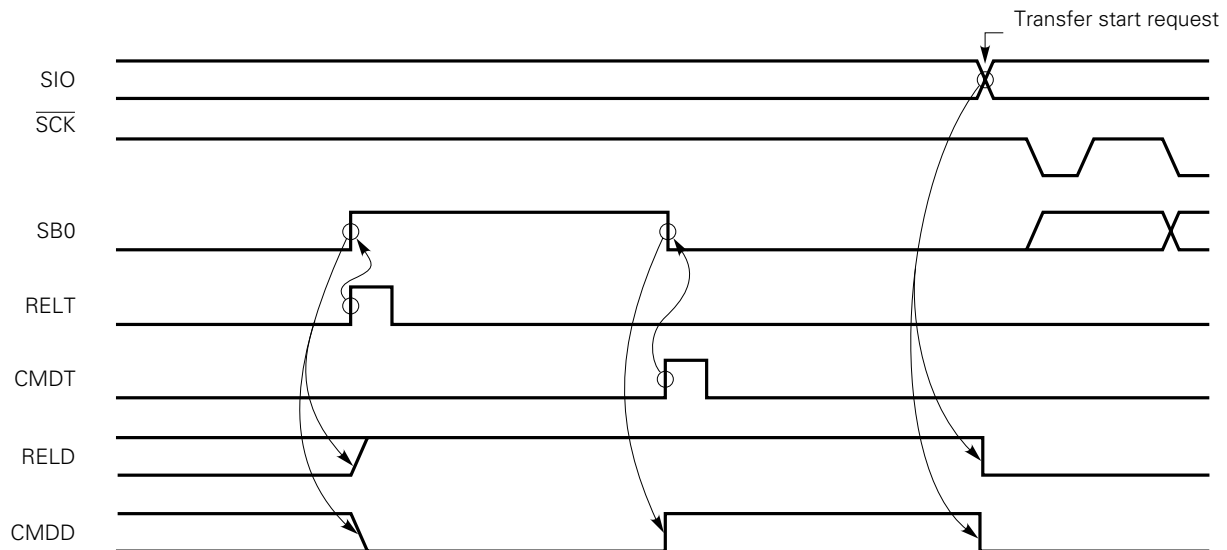
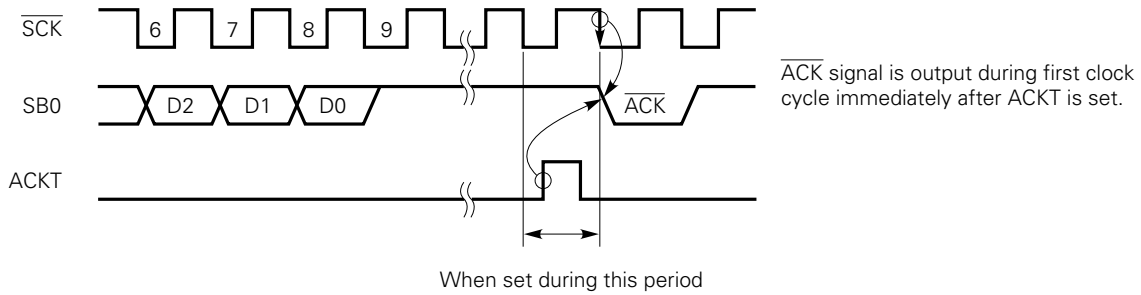


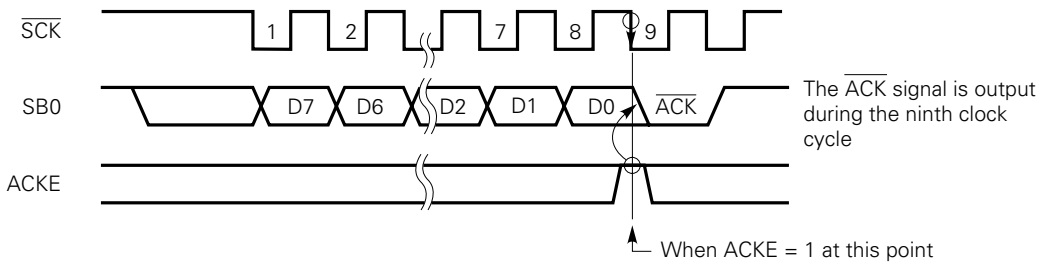
Fig. 10-23 ACKT Operation



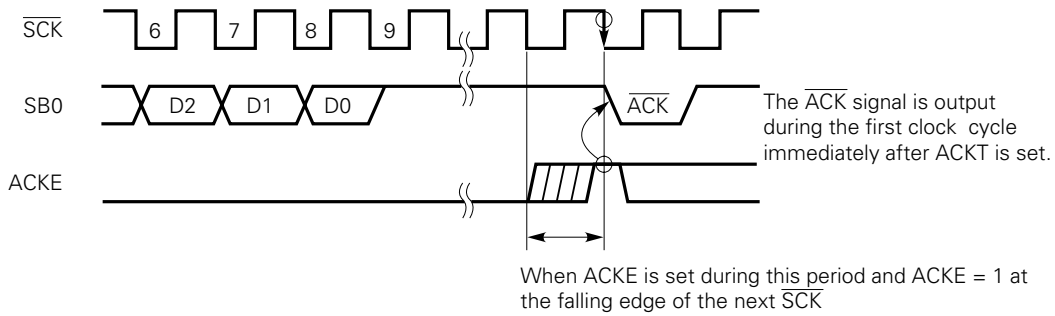
Caution Do not set ACKT before transfer has been completed.

Fig. 10-24 ACKE Operations

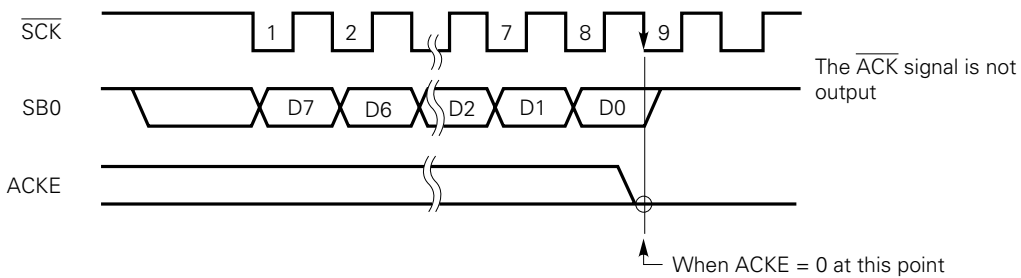
(a) When ACKE is set to 1 at the end of transfer



(b) When ACKE is set after transfer has been completed



(c) When ACKE is set to 0 at the end of transfer



(d) When ACKE is set to 1 for a short period of time

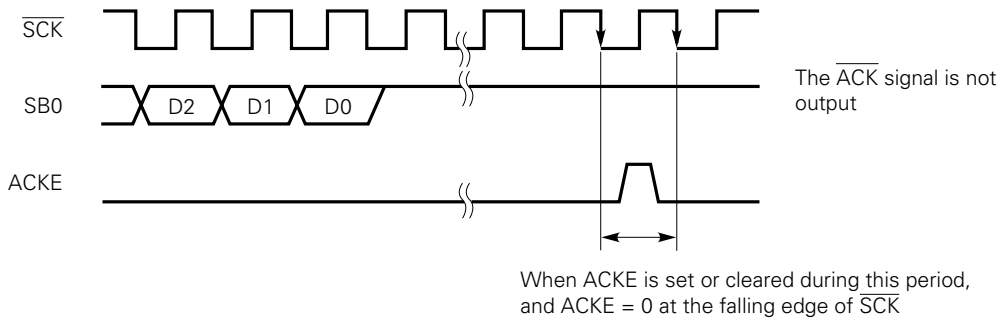
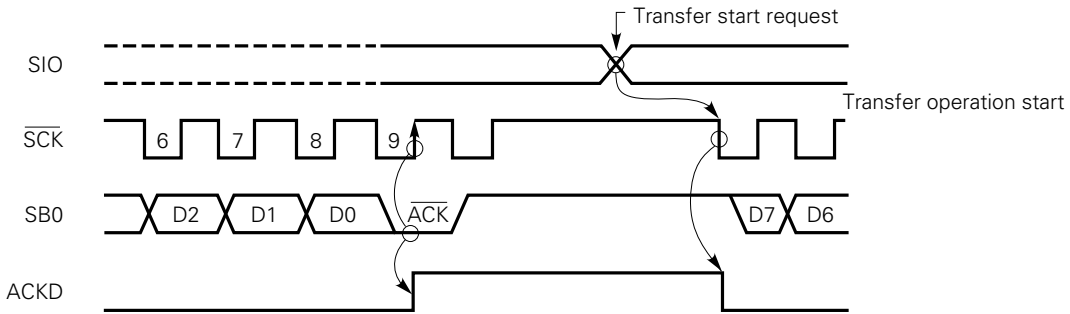
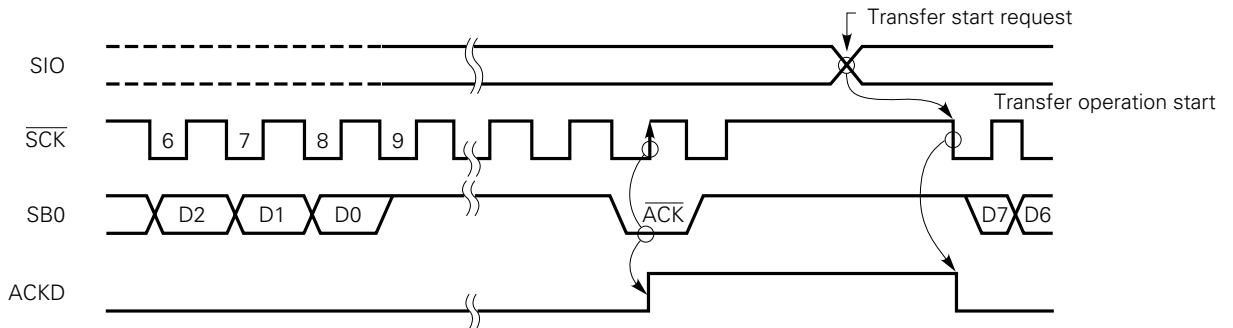


Fig. 10-25 ACKD Operations

(a) When the ACK signal is output during the ninth cycle of the SCK clock



(b) When the ACK signal is output after the ninth pulse of the SCK clock



(c) Clear timing when a transfer start is specified in the busy state

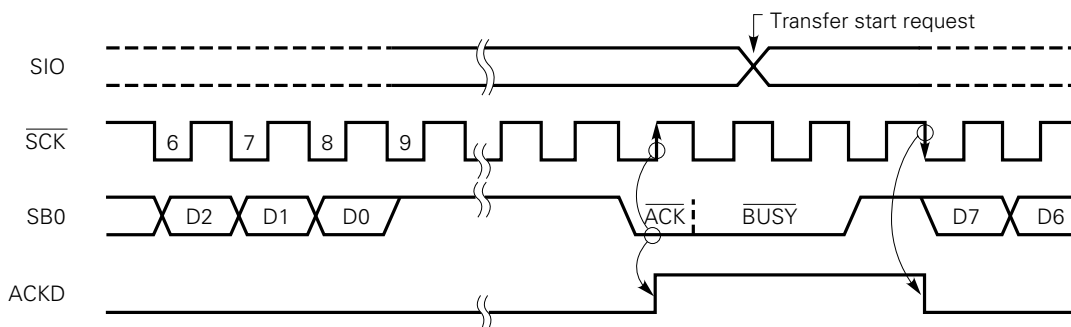


Fig. 10-26 BSYE Operation

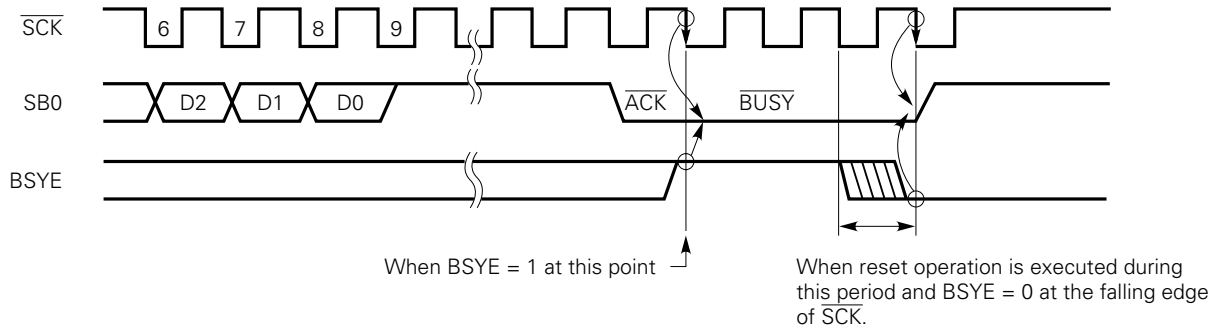


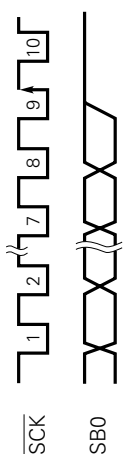
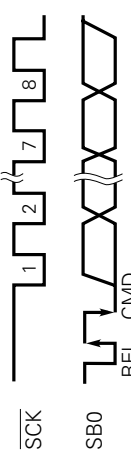
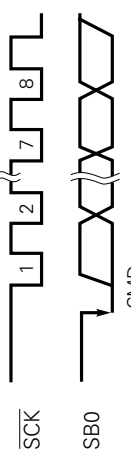
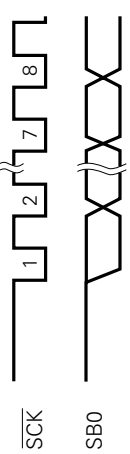
Table 10-2 Signals in SBI Mode (1/3)

Signal name	Output device	Definition	Timing chart	Output condition	Influence on flag	Description
Bus release signal (REL)	Master	Rising edge of SB0 while SCK is set to 1	<p>The timing chart shows two signals: SCK and SB0. SCK is a high-level signal. SB0 is also high, then it transitions to low. An arrow points to the falling edge of SB0, which is labeled as the signal event.</p>	<ul style="list-style-type: none"> RELT is set. 	<ul style="list-style-type: none"> RELD is set. CMDD is cleared. 	Following this signal, the CMD signal is output to indicate that the transmission data is an address.
Command signal (CMD)	Master	Falling edge of SB0 while SCK is set to 1	<p>The timing chart shows two signals: SCK and SB0. SCK is a high-level signal. SB0 is also high, then it transitions to low. An arrow points to the falling edge of SB0, which is labeled as the signal event.</p>	<ul style="list-style-type: none"> CMDT is set. 	<ul style="list-style-type: none"> CMDD is set. 	<p>(1) If this signal is output after the REL signal, the transmission data is an address.</p> <p>(2) If this signal is output without the REL signal, the transmission data is a command.</p>

Table 10-2 Signals in SBI Mode (2/3)

Signal name	Output device	Definition	Timing chart	Output condition	Influence on flag	Description	
Acknowledge signal (ACK)	Master/slave	Low signal output to SB0 within a single cycle of the SCK clock after serial reception has been completed	<p>The timing chart shows three signals: SCK (Serial Clock), SB0 (Slave Busy/Ready), and ACK (Acknowledge). SCK is a periodic clock signal. SB0 is a signal that transitions from high to low during the ACK pulse and returns to high during the BUSY period. ACK is a narrow pulse that occurs while SB0 is low. BUSY is a period where SB0 is high, indicating that a serial transfer is in progress.</p>	<ol style="list-style-type: none"> ① ACKE is set to 1. ② ACKT is set. 	<ul style="list-style-type: none"> • ACKD is set. 	Reception is completed.	
Busy signal (BUSY)	Slave	Low signal output to SB0, following the acknowledge signal		<p>The timing chart shows three signals: SCK (Serial Clock), SB0 (Slave Busy/Ready), and ACK (Acknowledge). SCK is a periodic clock signal. SB0 is a signal that transitions from high to low during the ACK pulse and returns to high during the BUSY period. ACK is a narrow pulse that occurs while SB0 is low. BUSY is a period where SB0 is high, indicating that a serial transfer is in progress.</p>	<ul style="list-style-type: none"> • BSYE is set to 1. 	—	Because an operation is currently in progress, serial transmission or reception is disabled.
Ready signal (READY)	Slave	High signal output to SB0 before serial transfer is started and after serial transfer is completed			<p>The timing chart shows three signals: SCK (Serial Clock), SB0 (Slave Busy/Ready), and ACK (Acknowledge). SCK is a periodic clock signal. SB0 is a signal that transitions from high to low during the ACK pulse and returns to high during the BUSY period. ACK is a narrow pulse that occurs while SB0 is low. BUSY is a period where SB0 is high, indicating that a serial transfer is in progress.</p>	<ol style="list-style-type: none"> ① BSYE is set to 0. ② When CTXE is set to 1, data is written into the SIO (a serial transfer start is specified). Note 2 ③ When CTXE is set to 0 and CRXE is set to 1, the instruction to read data from the SIO is executed. ④ The CRXE bit is changed from 0 to 1. 	—

Table 10-2 Signals in SBI Mode (3/3)

Signal name	Output device	Definition	Timing chart	Output condition	Influence on flag	Description
Serial clock (\overline{SCK})	Master	Synchronization clock for output address, command, or data, \overline{ACK} signal, and synchronous \overline{BUSY} signal. The address, command, or data is transferred during the first eight cycles.		① When CTXE is set to 1, the instruction to write data into the SIO is executed (a serial transfer start is specified). Note 2 ② When CTXE is set to 0 and CRXE is set to 1, the instruction to read data from the SIO is executed. ③ The CRXE bit is changed from 0 to 1.	CSIF is set (at the rising edge of the eighth clock pulse). Note 1	Timing of signal output to the serial data bus
Address (A7 to A0)	Master	8-bit data transferred in synchronization with \overline{SCK} after the REL and CMD signals are output				Address of a slave device on the serial bus
Command (C7 to C0)	Master	8-bit data transferred in synchronization with \overline{SCK} after the CMD signal is output without the REL signal				Instruction and message to a slave device
Data (D7 to D0)	Master/ slave	8-bit data transferred in synchronization with \overline{SCK} when neither the REL signal nor CMD signal is output				Data to be processed by the slave or master device

Notes 1. If WUP is set to 0, CSIF is always set at the rising edge of the eighth pulse of the \overline{SCK} clock.
 If WUP is set to 1, CSIF is set at the rising edge of the eighth pulse of the \overline{SCK} clock only when an address is received.
 2. Data is neither sent nor received in the \overline{BUSY} state. Data transfer is started once the READY state has been established.

10.6.8 Communication

In SBI communication, the master device outputs an address on the serial bus and, usually, one target slave device is selected out of two or more devices according to the address.

Once the target device has been determined, commands and data are transferred between the master device and slave device to implement serial communication.

Figs. 10-27 to 10-30 show the timing charts for data communication.

10.6.9 Releasing the Busy State

The conditions governing the release of the busy state depend on whether transmission or reception is permitted. They can be applied to high-speed transfer using the SBI macro service. Table 10-3 lists the methods of releasing the \overline{BUSY} state.

Table 10-3 Conditions Governing Release of \overline{BUSY}

Divice and condition		Conditions governing the release of \overline{BUSY}
CTxE	CRxE	
0	0	
0	1	BSYE←0, SIO read access
1	0	BSYE←0, SIO write access ^{Note}
1	1	

Note Write FFH into the SIO if the next operation is reception.

10.6.10 Setting Wake-Up

If WUP is set to 1 in the busy state, the wake-up state is set as soon as the device enters the ready state.

In the wake-up state, an interrupt (INTCSI) is issued only when an address is received. The acknowledge (\overline{ACK}) signal is not detected in this state.

10.6.11 Starting Transmission and Reception

Start transmission and reception in the same way as the busy state is released. Even if start of transmission and reception is specified, transmission and reception are not started while the slave device is outputting the \overline{BUSY} signal. Transmission and reception are started when the busy state is released.

Fig. 10-27 Sending an Address from Master Device to Slave Device

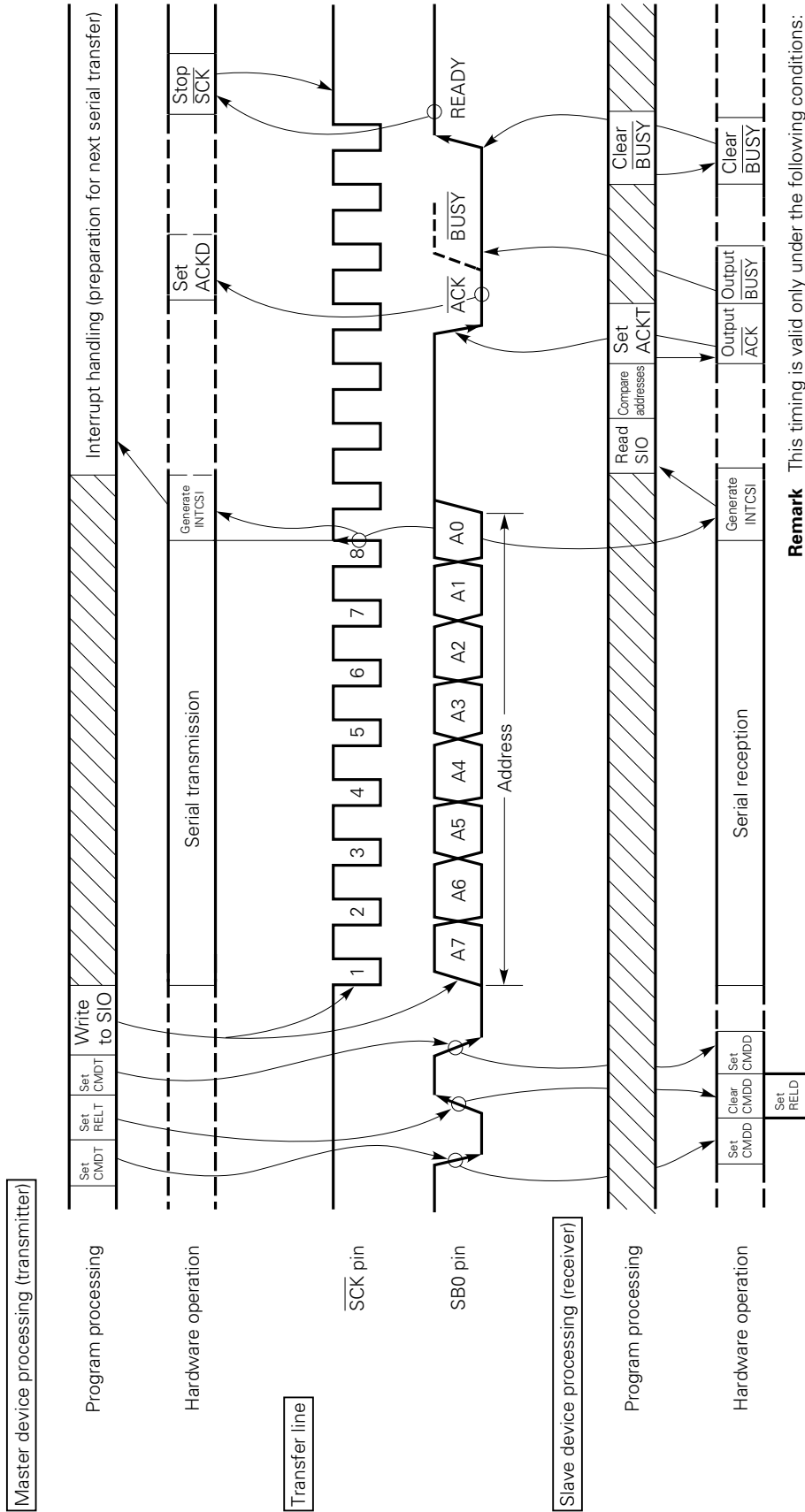


Fig. 10-28 Sending a Command from Master Device to Slave Device

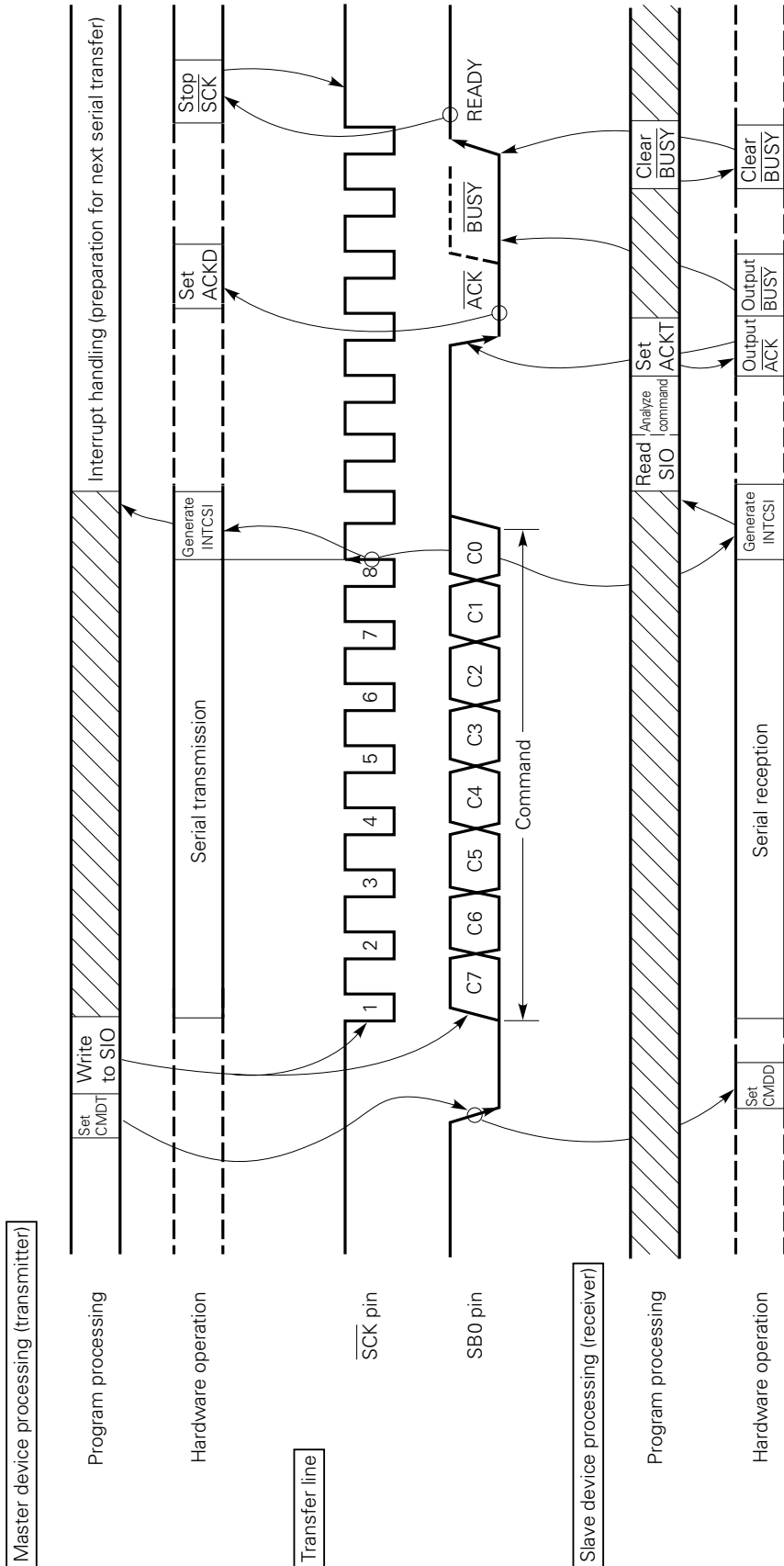


Fig. 10-29 Sending Data from Master Device to Slave Device

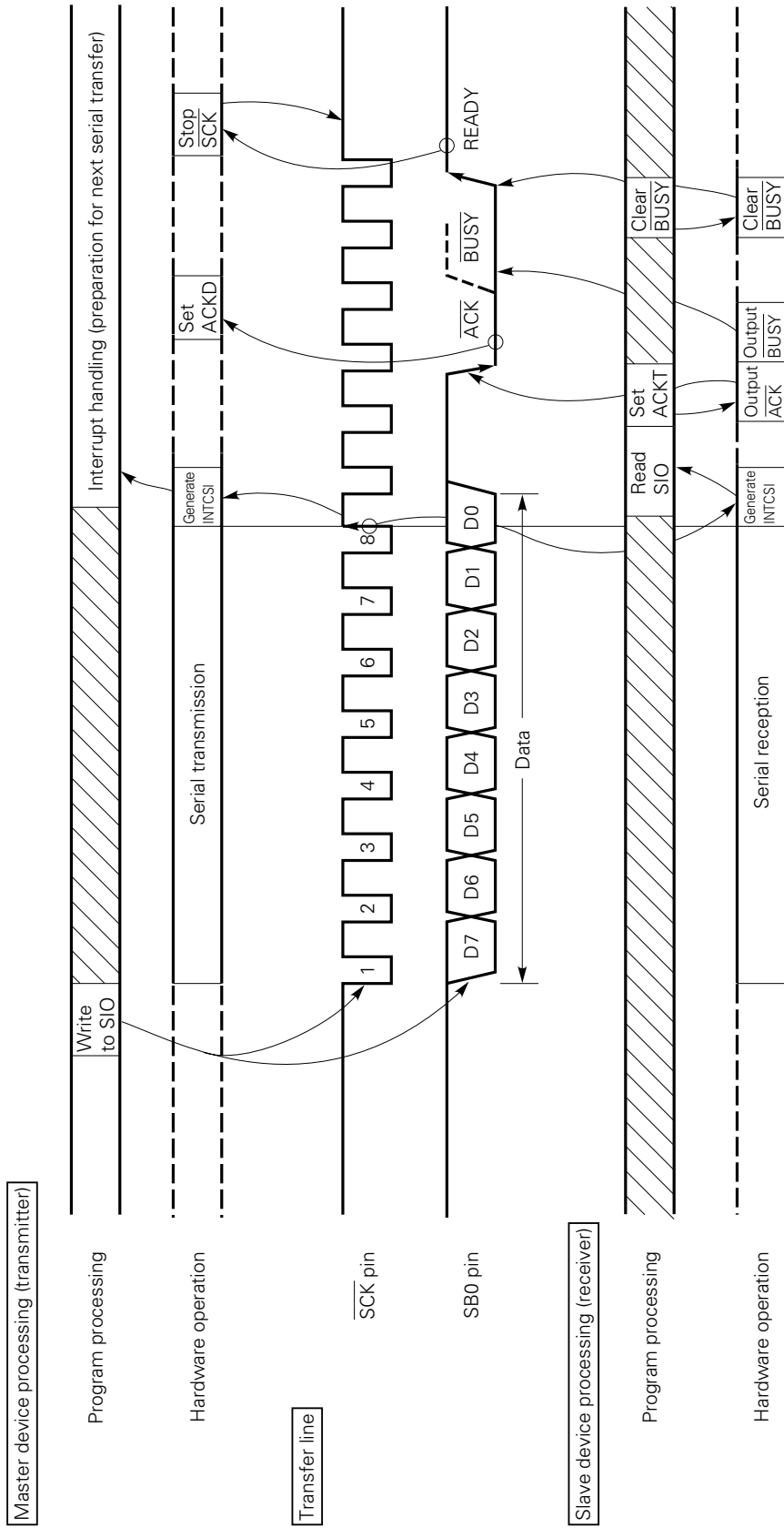
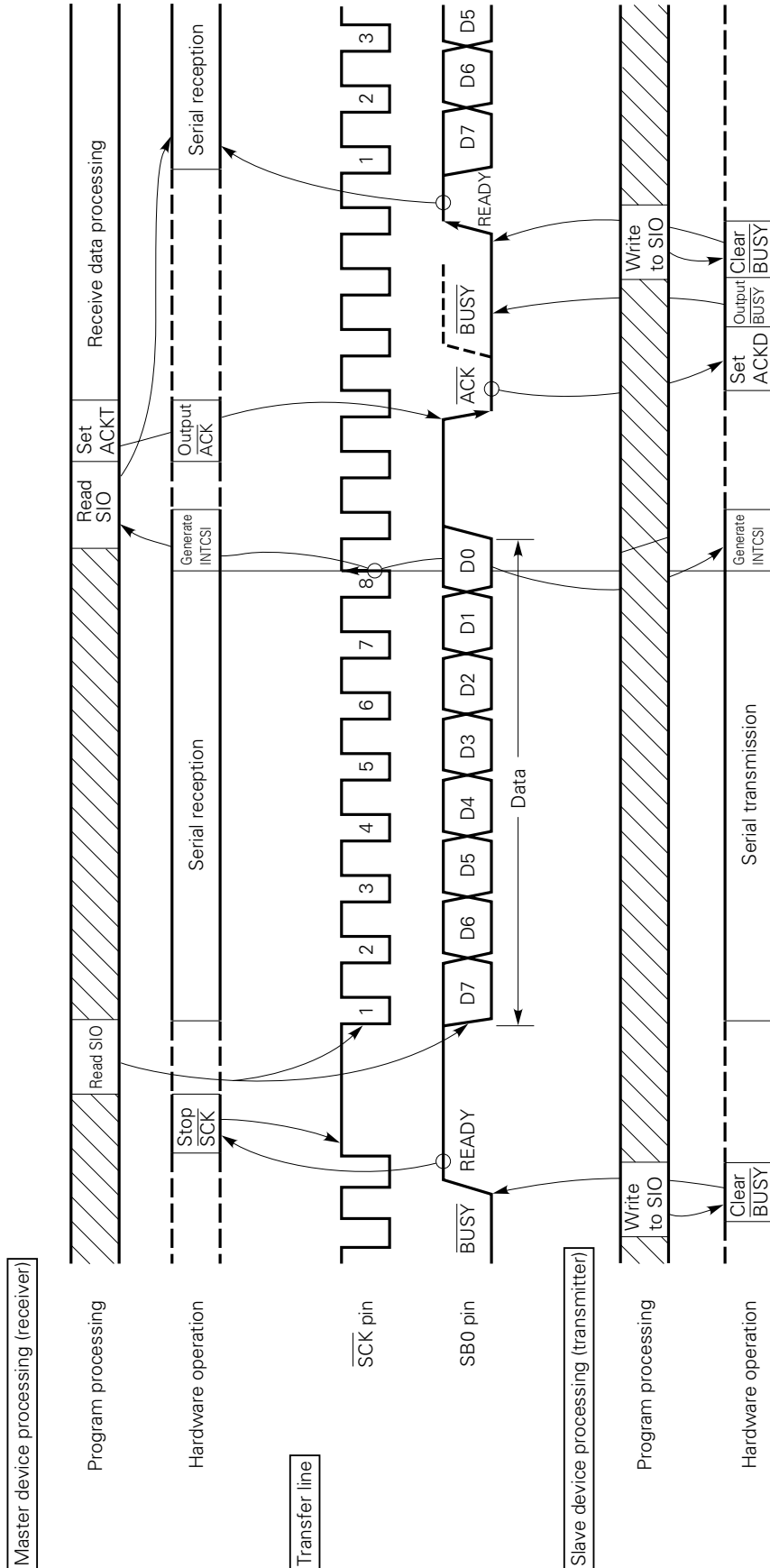


Fig. 10-30 Sending Data from the Slave Device to the Master Device



10.7 NOTES

- (1) Do not change CTXE from 0 to 1 and CRXE from 1 to 0, or vice versa, by means of a single instruction. If this is attempted, the serial clock counter will malfunction and the first communication after the change will be terminated before the eighth bit is sent. To change these statuses, use two instructions as shown below:

Example Changing CTXE from 1 to 0 and CRXE from 0 to 1

CLR1 CTXE

SET1 CRXE

- (2) When switching the master and slave, the input and output of the serial clock line (\overline{SCK}) are asynchronously switched between the master and slave. The serial clock line (\overline{SCK}) requires a pull-up resistor.
- (3) Do not set ACKT before the transfer has been completed.

CHAPTER 11 EDGE DETECTION FUNCTION

Pins P20 to P26 support an edge detection function to program a rising or falling edge. The detected edge is sent to the internal hardware. Table 11-1 shows the relationship between pins P20 to P26, and the use of the detected edge.

Table 11-1 Pins P20 to P26 and Use of Detected Edge

Pins	Use	Register specifying the edge to be detected
P20	NMI, control of the standby circuit	INTM0
P21	INTP0, capture signal of 8-bit timer/counter 1, trigger signal of real-time output port	
P22	INTP1, capture signal of 8-bit timer/counter 2	
P23	INTP2, CI (count clock of 8-bit timer/counter 2)	
P24	INTP3, capture signal of 16-bit timer/counter	INTM1
P25	INTP4, ASCK (external baud rate input of UART)	
P26	INTP5, conversion start signal of the A/D converter	

The edge detection function is always enabled except in STOP mode. (The edge detection function of pin P20 is enabled even in STOP mode.)

11

11.1 EXTERNAL INTERRUPT MODE REGISTERS (INTM0, INTM1)

The external interrupt mode registers (INTM0, INTM1) specify the valid edge to be detected on pins P20 to P26. The INTM0 register specifies the valid edge for pins P20 to P23, while the INTM1 register specifies that for pins P24 to P26.

The INTM1 register can also switch the INTP4 interrupt and INTC30 interrupt (for details see **Chapter 12**). When ASCK is input, both edges are detected, independent of the values of these registers.

The 8-bit manipulation instruction and bit manipulation instruction can read and write the contents of the INTM0 and INTM1 registers. Figs. 11-1 and 11-2 show their formats.

When $\overline{\text{RESET}}$ is input, both these registers are set to 00H.

Fig. 11-1 Format of External Interrupt Mode Register 0 (INTM0)

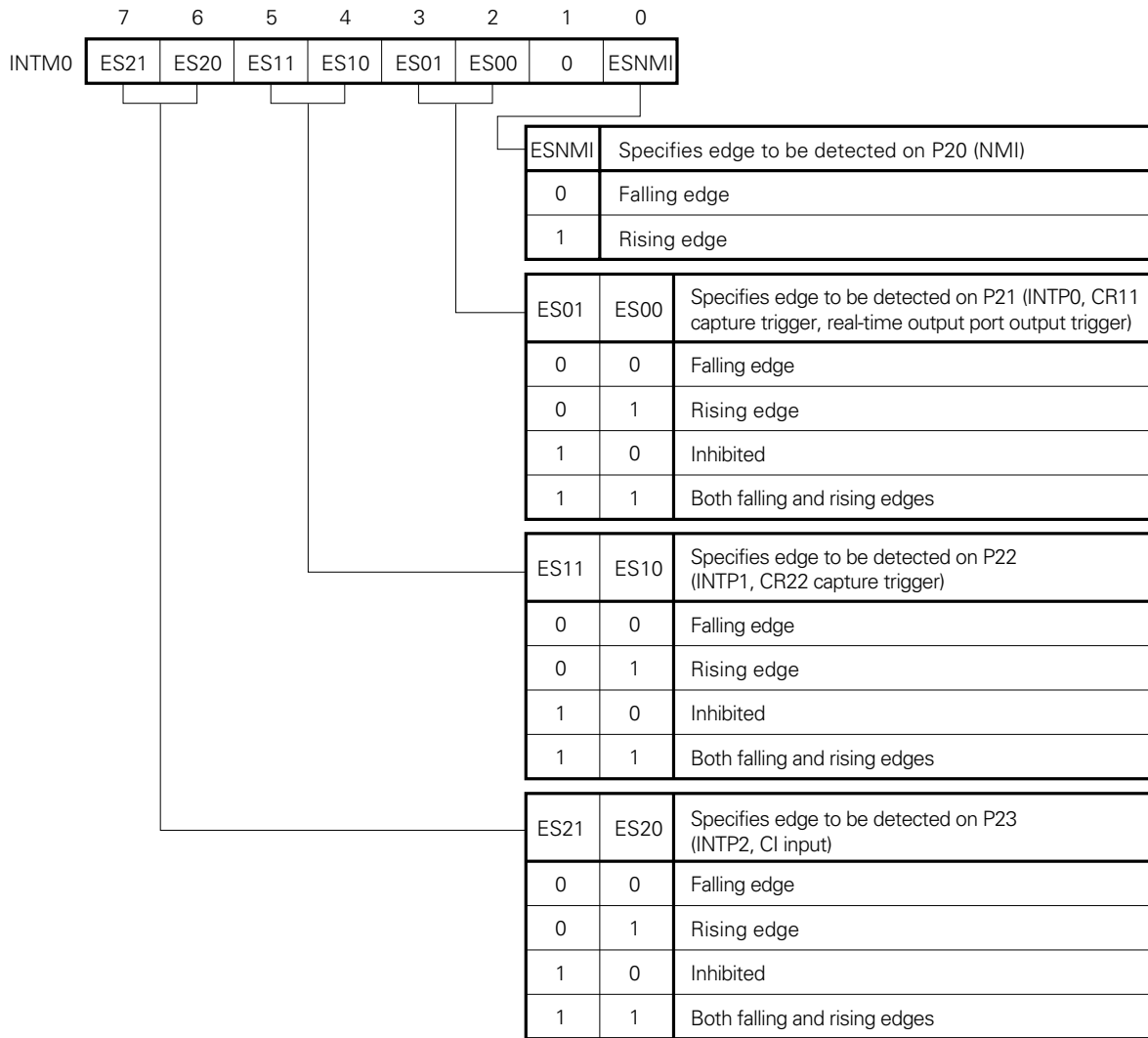
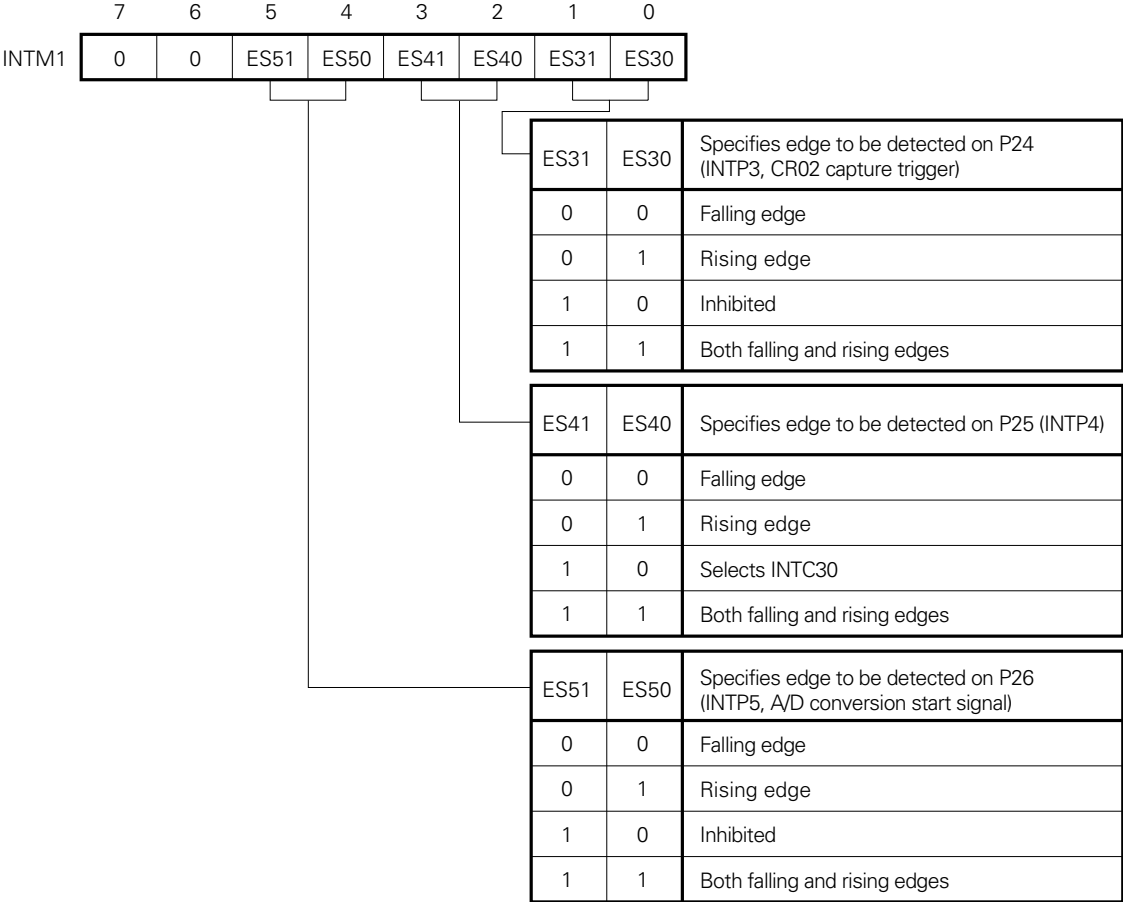


Fig. 11-2 Format of External Interrupt Mode Register 1 (INTM1)

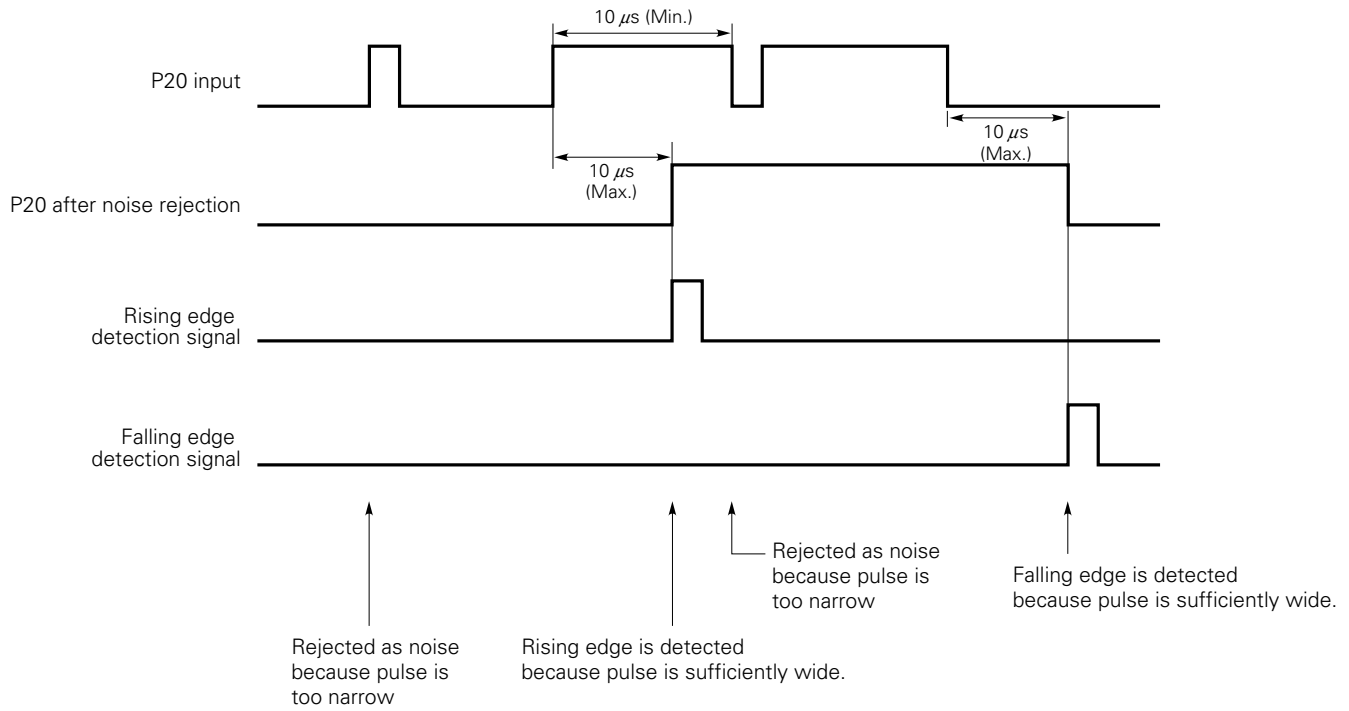


Caution If an edge is input while a valid edge is changing, it is not known whether the new edge is judged as being a valid edge.

11.2 EDGE DETECTION ON PIN P20

An edge on pin P20 is detected after noise elimination by means of analog delay. A pulse width of at least 10 μs is required to detect the edge.

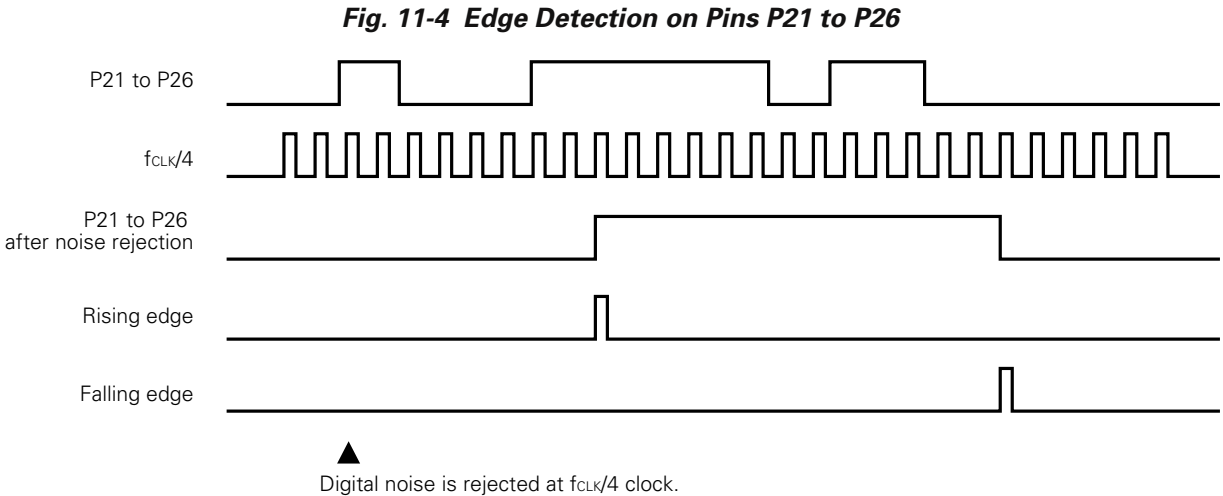
Fig. 11-3 Edge Detection on Pin P20



Caution Because noise elimination by analog delay is performed by pin P20, an edge is detected up to 10 μs after it is actually input. This pin differs from pins P21 to P26 in that the delay depends on the characteristics of the device.

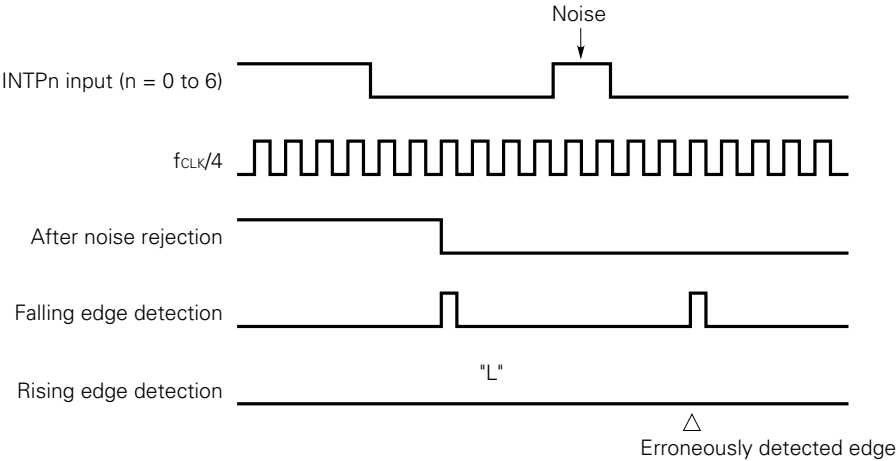
11.3 EDGE DETECTION ON PINS P21 TO P26

An edge on pins P21 to P26 is detected after digital noise elimination by means of clock sampling. The digital noise elimination is performed by means of sampling with the $f_{CLK}/4$ clock. The input signal is eliminated as noise if an identical level is not obtained three or more times in a row (even if an identical level is consecutively obtained twice). The input signal is detected as a valid edge only when its level remains identical for three or more cycles of the $f_{CLK}/4$ clock ($2 \mu s$: $f_{CLK} = 6 \text{ MHz}$).

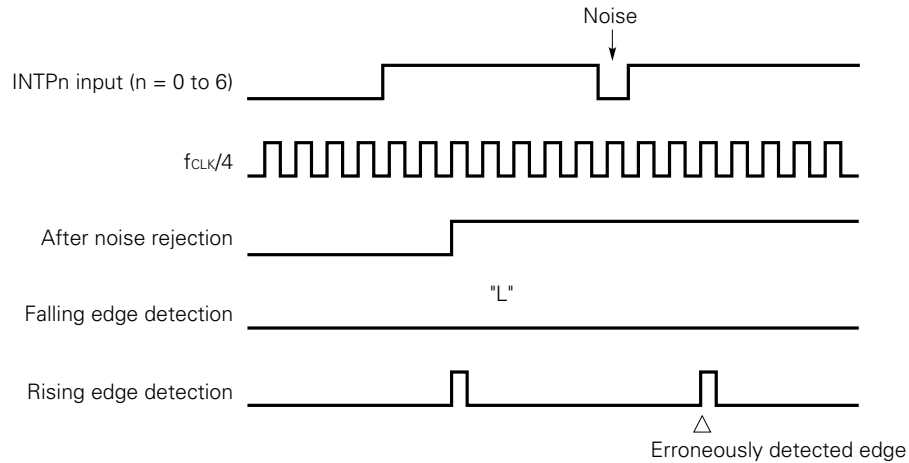


- Cautions**
1. Because the $f_{CLK}/4$ clock is used for digital noise elimination, it takes about 8 to 12 cycles of the f_{CLK} clock to detect an edge after the edge is actually input.
 2. If the width of an input pulse corresponds to 8 to 12 cycles of the f_{CLK} clock, it cannot be determined whether the pulse is detected as a valid edge. To ensure the accurate detection of a pulse, hold the pulse at an identical level for 12 clock cycles or longer.
 3. If noise input to a pin is synchronized with the $f_{CLK}/4$ clock of the $\mu PD78214$, it may not be judged as being noise. If input of such noise is possible, add a filter to the input pin to eliminate the noise.
 4. An in-circuit emulator cannot successfully eliminate digital noise. It may erroneously detect a falling edge due to noise during the input of a low signal and a rising edge due to noise during the input of a high signal (see Fig. 11-5). When data is read from port 2, noise is not eliminated, being read instead.

Fig. 11-5 Erroneously Detected Edges
(a) Erroneously detected edge during input of a low signal



(b) Erroneously detected edge during input of a high signal



If the IE-78210-R is used, the real-time output port, timer/counter, and A/D converter operate according to the erroneously detected edge. If another in-circuit emulator is used, these components operate as described below:

- **Real-time output port** : Operates according to the erroneously detected edge.
- **A/D converter** : Operates according to the erroneously detected edge.
- **Capture or clear operation of the timer/counter** : Carried out independently of the erroneously detected edge. Even if the erroneously detected edge causes an interrupt, the capture value is not updated. The value of CR22 becomes undefined after being read by the CPU.
- **Compare operation of the timer/counter** : If a mode for performing a clear operation after a capture operation is selected, or if timer/counter 2 is used as an external event counter, the erroneously detected edge causes the timing of match interrupt generation to be changed. As a result, the timing of match interrupt generation will disagree with that when the values of the timer/counter and compare register match. If the mode for performing a clear operation after a capture operation is selected, the timing of match interrupt generation can be corrected by inputting a correct edge or by stopping the timer/counter. If timer/counter 2 is used as an external event counter, the timing of the match interrupt generation can be corrected by stopping the timer/counter. Timer output is not affected by the erroneously detected edge and operates according to the correct timing.

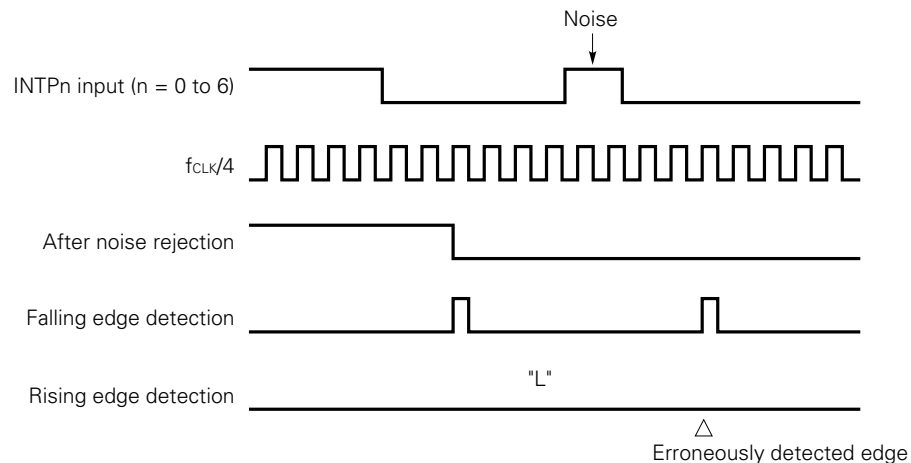
11.4 NOTES

- (1) If an edge is input while a valid edge is changed, it cannot be determined whether the new edge is judged as being a valid edge.
- (2) Noise elimination by analog delay is carried out on pin P20. An edge is detected up to 10 μs after the edge is actually input. Pin P20 differs from pins P21 to P26 in that the delay time depends on the characteristics of the device.
- (3) On pins P21 to P26, digital noise elimination is carried out with the f_{CLK}/4 clock. It takes about 8 to 12 cycles of the f_{CLK} clock to detect an edge after it is actually input.
- (4) If the width of a pulse input to pins P21 to P26 corresponds to 8 to 12 cycles of the f_{CLK} clock, it cannot be determined whether the pulse is detected as being a valid edge. To ensure the accurate detection of a pulse, hold the pulse at an identical level for 12 clock cycles or longer.

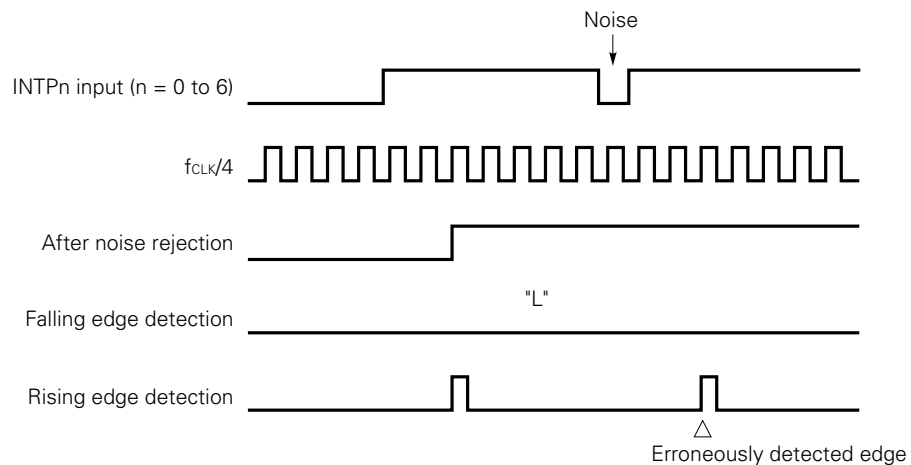
- (5) If noise input to pins P21 to P26 is synchronized with the $f_{CLK}/4$ clock of the μ PD78214, it may not be judged as being noise. If the input of such noise is possible, add a filter to the input pin so that the noise can be eliminated.
- (6) An in-circuit emulator cannot successfully eliminate digital noise. It may erroneously detect a falling edge due to noise during the input of a low signal and a rising edge due to the noise during the input of a high signal (see Fig. 11-6). When data is read from port 2, noise is not eliminated, being read instead.

Fig. 11-6 Erroneously Detected Edges

(a) Erroneously detected edge during input of low signal



(b) Erroneously detected edge during input of high signal



If the IE-78210-R is used, the real-time output port, timer/counter, and A/D converter operate according to the erroneously detected edge. If another in-circuit emulator is used, these components operate as described below:

- Real-time output port : Operates according to the erroneously detected edge.
- A/D converter : Operates according to the erroneously detected edge.
- Capture or clear operation of the timer/counter : Carried out independently of the erroneously detected edge. Even if the erroneously detected edge causes an interrupt, the capture value is not updated. The value of CR22 becomes undefined after it has been read by the CPU.

- Compare operation of the timer/counter : If the mode for carrying out a clear operation after a capture operation is selected, or if timer/counter 2 is used as an external event counter, the erroneously detected edge causes the timing of match interrupt generation to be changed. As a result, the timing of match interrupt generation will disagree with that when the values of the timer/counter and compare register match. If the mode for performing a clear operation after a capture operation is selected, the timing of match interrupt generation can be corrected by inputting a correct edge or by stopping the timer/counter. If timer/counter 2 is used as an external event counter, the timing of match interrupt generation can be corrected by stopping the timer/counter. The timer output is not affected by the erroneously detected edge, operating according to the correct timing.

CHAPTER 12 INTERRUPT FUNCTIONS

The μ PD78214 has the following two interrupt handling modes. Either mode can be selected by the program. Interrupt handling by a macro service is limited to the interrupt request sources provided with a macro service handling mode listed in Table 12-1.

Table 12-1 Interrupt Request Handling Modes

Interrupt request handling mode	Processed by:	PC and PSW contents	Processing
Vectored interrupt	Software	Saved and restored	Program control branches to a specified service program.
Macro service	Hardware (firmware)	Retained	Predetermined processing, such as data transfer between memory and I/O, is performed.

Maskable vectored interrupts can easily be controlled by multiple-interrupt handling with two priority levels.

12.1 INTERRUPT REQUEST SOURCES

The μPD78214 has 19 interrupt request sources shown in Table 12-2. Each of these sources is assigned an interrupt vector table.

Table 12-2 Interrupt Request Sources

Interrupt request type	Default priority	Interrupt request source	Generating unit	Macro service type	Vector table address
Software	None	BRK instruction execution		None	003EH
Nonmaskable	None	NMI (edge input to the pin is detected)		None	0002H
Maskable	0	INTP0 (edge input to the pin is detected)	Edge detection	A, B	0006H
	1	INTP1 (edge input to the pin is detected)		A, B	0008H
	2	INTP2 (edge input to the pin is detected)		A, B	000AH
	3	INTP3 (edge input to the pin is detected)		B	000CH
	4	INTC00 (TM0-CR00 coincidence signal generation)	16-bit timer/counter	B	0014H
	5	INTC01 (TM0-CR01 coincidence signal generation)		B	0016H
	6	INTC10 (TM1-CR10 coincidence signal generation)	8-bit timer/counter 1	A, B, C	0018H
	7	INTC11 (TM1-CR11 coincidence signal generation)		A, B, C	001AH
	8	INTC21 (TM2-CR21 coincidence signal generation)	8-bit timer/counter 2	A, B	001CH
	9	INTP4 (edge input to the pin is detected)	Edge detection	B	000EH
		INTC30 (TM3-CR30 coincidence signal generation)	8-bit timer/counter 3	A, B	
	10	INTP5 (edge input to the pin is detected)	Edge detection	B	0010H
		INTAD (A/D conversion end)	A/D converter	A, B	
	11	INTC20 (TM2-CR20 coincidence signal generation)	8-bit timer/counter 2	A, B	0012H
	12	INTSER (asynchronous serial interface reception error occurrence)	Asynchronous serial interface	None	0020H
13	INTSR (asynchronous serial interface reception end)	A, B		0022H	
14	INTST (asynchronous serial interface transmission end)	A, B		0024H	
15	INTCSI (clock-synchronized serial interface transmission end)	Clock-synchronized serial interface	A, B	0026H	

Remark The default priority is fixed by hardware. If two or more interrupts having the same priority occur simultaneously, they are handled according to their default priority.

12.1.1 Software Interrupt Request

A software interrupt request is issued by the BRK instruction, which eventually causes a vectored interrupt. Interrupt requests issued by the BRK instruction are accepted even in an interrupt disabled state. In this case, interrupt priority control is not applied.

When the BRK instruction is executed, the vector table contents are unconditionally set in the PC to cause a branch. By executing the BRK instruction in a BRK service routine, the service routine can nest itself.

To exit the BRK service routine, execute the RETB instruction.

12.1.2 Nonmaskable Interrupt Request

A nonmaskable interrupt request is input to the NMI pin. When a valid edge, specified by bit 0 (ESNMI) of external interrupt mode register 0 (INTM0), is input to the NMI pin, an interrupt request is generated.

A nonmaskable interrupt request is accepted unconditionally, even in an interrupt disabled state. In this case, interrupt priority control is not applied; the nonmaskable interrupt request takes precedence over all other interrupts.

12.1.3 Maskable Interrupt Request

Maskable interrupt requests can be masked by setting the interrupt mask register (MK0). The IE flag in the PSW can specify whether to enable or disable all the maskable interrupts simultaneously.

A default priority is assigned to each maskable interrupt request as shown in Table 12-2, so that, when two or more interrupts having the same priority occur at the same time, which interrupt takes precedence is determined. The interrupts can be divided into two groups by the priority specification flag register (PR0); a group of interrupts with higher priority and a group of interrupts with lower priority, so that multiple-interrupt handling can be achieved. However, the macro service is accepted independently of the priority control and the IE flag.

12.1.4 Selecting an Interrupt Source

Interrupts INTP4 and INTC30 cannot be used at the same time, because these interrupts share the same vector table, interrupt request flags, and other control flags. Therefore, either INTP4 or INTC30 must be selected by software. The same holds true of a pair of INTP5 and INTAD. A selected interrupt request source is given the right to use the vector table, interrupt request flags (PIF_n; n = 4 or 5), interrupt mask flags (PMK_n; n = 4 or 5), interrupt service mode flags (PISM_n; n = 4 or 5), and priority specification flags (PPR_n; n = 4 or 5) exclusively, thus generating the corresponding interrupt and macro service. The other interrupt request sources cannot use these resources, and therefore cannot generate an interrupt or macro service.

The other types of interrupts have a dedicated vector table and control flags, and therefore need not be selected.

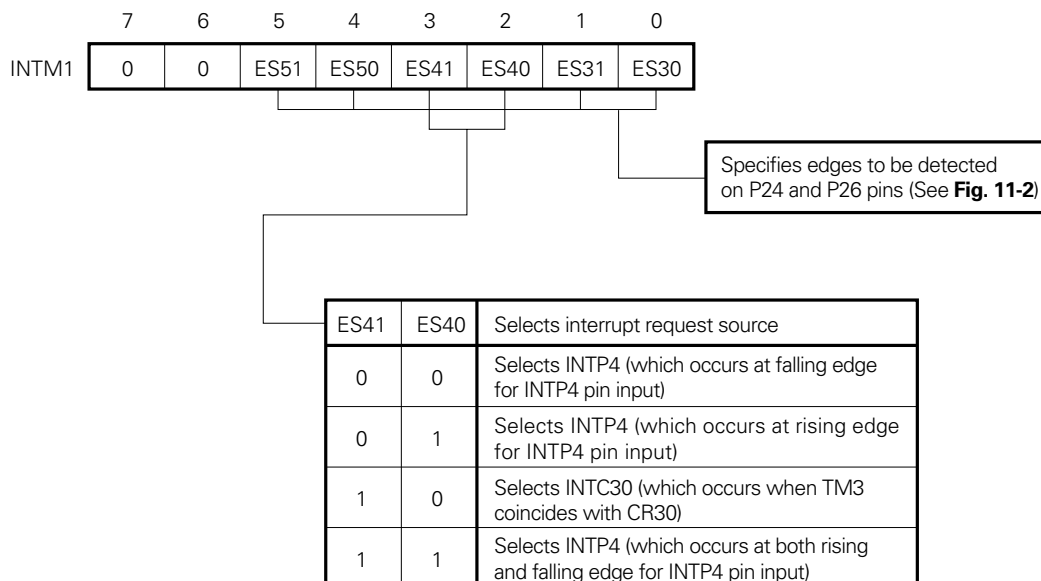
(1) Selecting INTP4 or INTC30

Interrupt INTP4 or INTC30 is selected by the ES40 and ES41 bits of external mode register 1 (INTM1).

Both 8-bit manipulation instruction and bit manipulation instruction can be used to read data from and write data to the INTM1 register. The format of this register is shown in Fig. 12-1.

When the RESET signal is input, the register is reset to 00H, and INTP4 occurs on the falling edge at the INTP4 pin.

Fig. 12-1 INTM1 Register Format



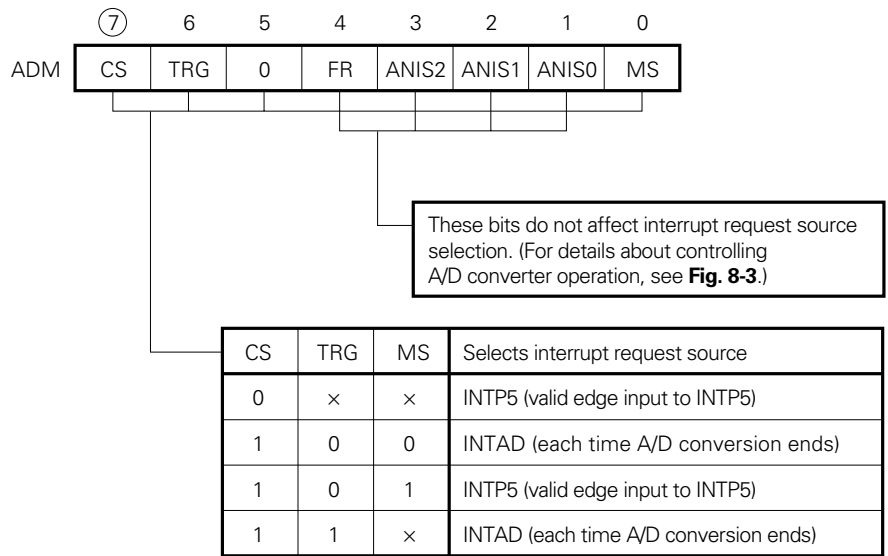
(2) Selecting INTP5 or INTAD

Interrupt INTP5 or INTAD is selected by the A/D converter mode register (ADM). (Either of these interrupts is selected automatically, according to the mode of operation specified for the A/D converter.)

Both 8-bit manipulation instruction and bit manipulation instruction can be used to read data from and write data to the ADM register. The format of this register is shown in Fig. 12-2. See **Chapter 8** for control of the A/D converter.

When the $\overline{\text{RESET}}$ signal is input, the register is reset to 00H.

Fig. 12-2 ADM Register Format



12.2 INTERRUPT HANDLING CONTROL REGISTERS

The following six registers control interrupt handling.

- Interrupt request flag register (IF0)
- Interrupt mask register (MK0)
- Interrupt service mode register (ISM0)
- Priority specification flag register (PR0)
- Interrupt status register (IST)
- Program status word (PSW)

The IF0, MK0, ISM0, and PR0 are 16-bit read/write registers. The contents of these registers can be manipulated in either 16- or 8-bit units. In addition, each bit of these registers can be set and reset by a bit manipulation instruction independently of the other bits. The IST and PSW are 8-bit read/write registers, whose contents can be manipulated in either 8- or 1-bit units. The IE flag in the PSW can be manipulated by a dedicated instruction. Fig. 12-3 through 12-8 show the formats of these registers.

Table 12-3 lists the interrupt request flags, interrupt mask flags, interrupt service mode flags, and priority specification flags corresponding to each interrupt request source.

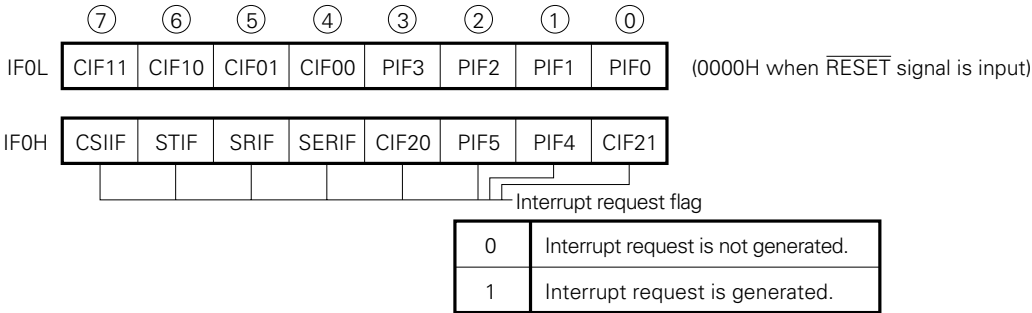
Table 12-3 Flags for Interrupt Request Sources

Interrupt request source	Interrupt request flag	Interrupt mask flag	Interrupt service mode flag	Priority specification flag
INTP0	PIF0	PMK0	PISM0	PPR0
INTP1	PIF1	PMK1	PISM1	PPR1
INTP2	PIF2	PMK2	PISM2	PPR2
INTP3	PIF3	PMK3	PISM3	PPR3
INTC00	CIF00	CMK00	CISM00	CPR00
INTC01	CIF01	CMK01	CISM01	CPR01
INTC10	CIF10	CMK10	CISM10	CPR10
INTC11	CIF11	CMK11	CISM11	CPR11
INTC21	CIF21	CMK21	CISM21	CPR21
INTP4)	PIF4	PMK4	PISM4	PPR4
INTC30)				
INTP5)	PIF5	PMK5	PISM5	PPR5
INTAD)				
INTC20	CIF20	CMK20	CISM20	CPR20
INTSER	SERIF	SERMK	—	SERPR
INTSR	SRIF	SRMK	SRISM	SRPR
INTST	STIF	STMK	STISM	STPR
INTCSI	CSIIF	CSIMK	CSISM	CSIPR

12.2.1 Interrupt Request Flag Register (IF0)

The IF0 register is a 16-bit register consisting of interrupt request flags. Each interrupt request flag is set to 1, when the corresponding interrupt request occurs. It is reset to 0, when a vectored interrupt is accepted or macro service processing is performed. When the $\overline{\text{RESET}}$ signal is input, this register is reset to 0000H.

Fig. 12-3 Interrupt Request Flag Register (IF0) Format



12.2.2 Interrupt Mask Register (MK0)

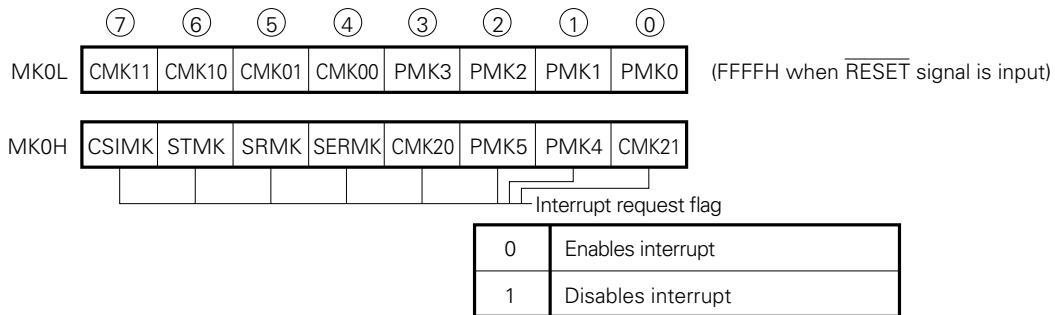
The MK0 register is a 16-bit register consisting of interrupt mask flags. Each interrupt mask flag enables or disables the corresponding interrupt request.

When the $\overline{\text{RESET}}$ signal is input, the register is set to FFFFH, thus disabling all maskable interrupts.

If an interrupt mask flag is set to 1, it inhibits acceptance of the corresponding interrupt request.

If an interrupt mask flag is reset to 0, it enables the corresponding interrupt request to be accepted as a vectored interrupt or macro service.

Fig. 12-4 Interrupt Mask Register (MK0) Format



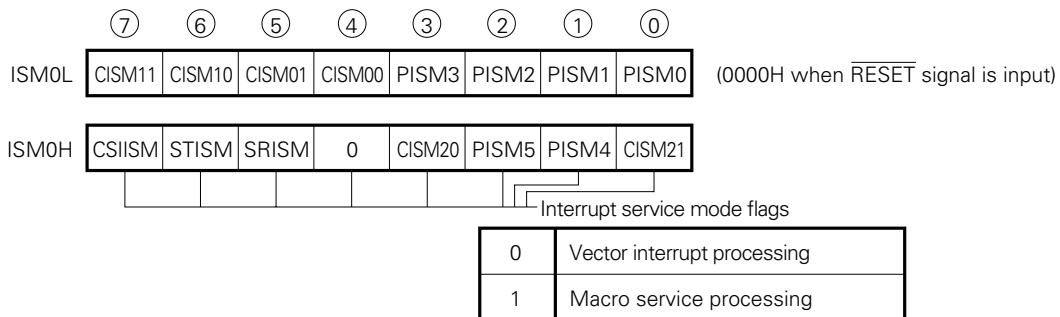
12.2.3 Interrupt Service Mode Register (ISM0)

The ISM0 register is a 16-bit register consisting of interrupt service mode flags.

If an interrupt service mode flag is 0, the corresponding interrupt request is handled as a vectored interrupt. If it is 1, the corresponding interrupt request is processed by a macro service. When a macro service request is executed a specified number of times, the interrupt service mode flag is reset to 0.

When the $\overline{\text{RESET}}$ signal is input, the register is reset to 0000H, thereby specifying vectored interrupt handling.

Fig. 12-5 Interrupt Service Mode Register (ISM0) Format



12.2.4 Priority Specification Flag Register (PR0)

The PR0 register is a 16-bit register consisting of interrupt priority specification flags that determine priority with which each interrupt is accepted. They are used to control multiple-interrupt handling.

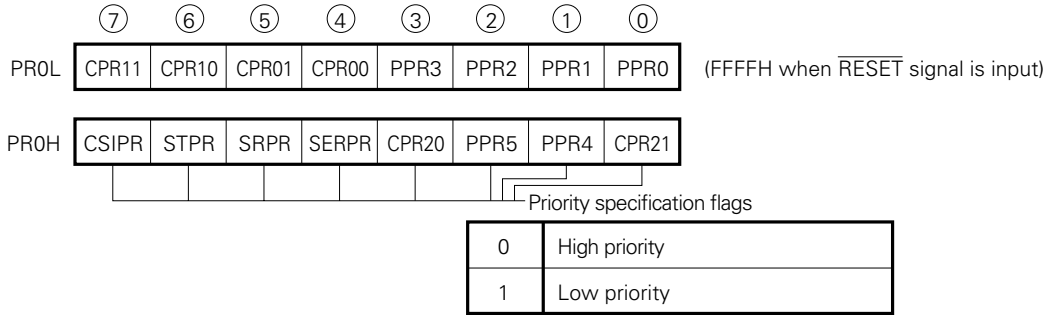
There are two interrupt groups with respect to priority; one having higher priority and one having lower priority. When a priority specification flag is 0, the corresponding interrupt request is specified to belong to the high-priority group; when a priority specification flag is 1, the corresponding interrupt request is specified to belong to the lower-priority group.

When an interrupt is accepted, the corresponding priority specification flag is sent to the ISP bit of the PSW.

When a low-priority vectored interrupt is being handled, vectored interrupt requests with lower and higher priorities are accepted for multiple-interrupt handling provided that interrupts are enabled. When a high-priority interrupt is being handled, high-priority vectored interrupts are accepted for multiple-interrupt handling provided that interrupts are enabled. Moreover, any interrupt requests specifying a macro service are accepted regardless of their priority.

When the $\overline{\text{RESET}}$ signal is input, this register is set to FFFFH, thereby specifying that all interrupts be in the low-priority group.

Fig. 12-6 Priority Specification Flag Register (PR0) Format



12.2.5 Interrupt Status Register (IST)

The IST register is an 8-bit register that controls multiple-interrupt handling for nonmaskable interrupt requests (input to the NMI pin) and indicates whether a nonmaskable interrupt request has been accepted.

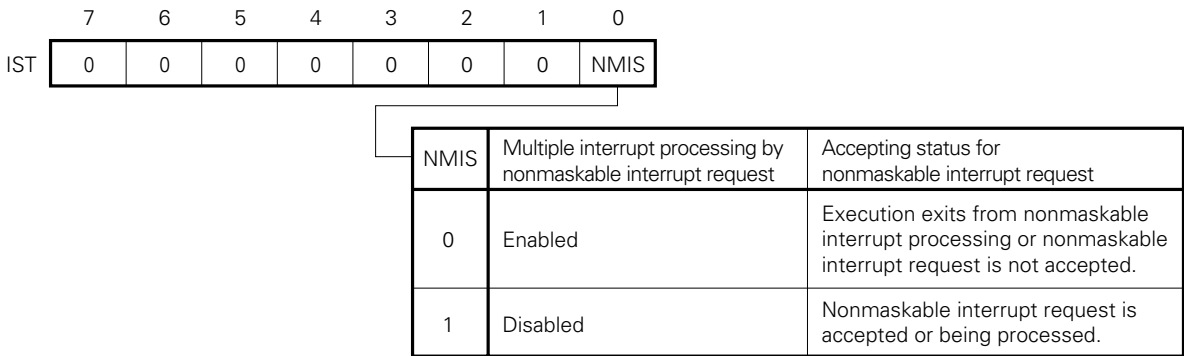
When a nonmaskable interrupt is being handled, another nonmaskable interrupt request may occur. In such a case, the nonmaskable interrupt request is accepted if the NMIS bit is 0; it is not accepted if the NMIS bit is 1.

The NMIS bit is set to 1 when a nonmaskable interrupt request is accepted. It is reset to 0, when a return (execution of the RETI instruction) from the interrupt handling for the nonmaskable interrupt request occurs.

Both an 8-bit manipulation instruction and bit manipulation instruction can be used to read data from and write data to the IST register. The NMIS flag is set to 1 when a nonmaskable interrupt is accepted. It is reset to 0 by the RETI instruction. Fig. 12-7 shows the format of the IST register.

When the $\overline{\text{RESET}}$ signal is input, the register is reset to 00H.

Fig. 12-7 Interrupt Status Register (IST) Format



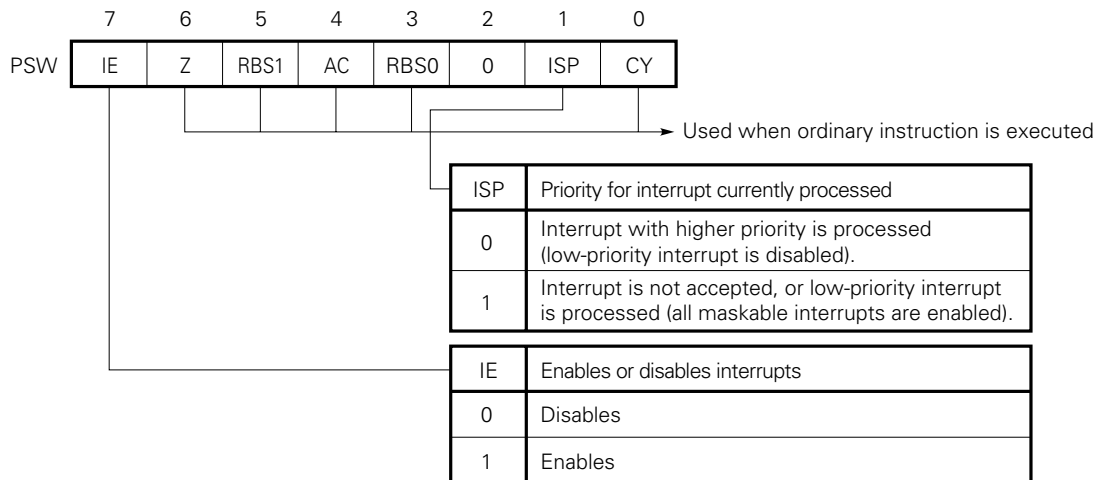
12.2.6 Program Status Word (PSW)

The PSW is a register that holds the result of instruction execution and the current status of interrupt requests. The register is mapped with the IE flag that specifies whether to enable maskable interrupts and the ISP flag to control multiple-interrupt handling.

The PSW can be read and written to in 8-bit units. It can also be manipulated by a bit manipulation instruction and dedicated instructions (EI and DI). When a vectored interrupt request is accepted, and the BRK instruction is executed, the PSW is saved in the stack, and the IE flag is reset to 0. When a maskable interrupt request is accepted, the priority specification flag for the corresponding interrupt is transferred to the ISP flag. When a nonmaskable interrupt request is accepted, the ISP flag is reset to 0. Also when the PUSH PSW instruction is executed, the PSW is saved in the stack. Executing the RETI, RETB, or POP PSW instruction restores the PSW from the stack.

When the $\overline{\text{RESET}}$ signal is input, the PSW is set to 02H.

Fig. 12-8 Program Status Word Format



12.3 INTERRUPT HANDLING

12.3.1 Accepting Software Interrupts

A software interrupt request is accepted by executing the BRK instruction. Software interrupts cannot be disabled. When a software interrupt request is accepted, the PSW and PC are saved in the stack in the stated order, the IE flag is reset to 0, and the PC is loaded with the contents of the vector table (at 003EH and 003FH) to cause a branch. The RETB instruction is used to return from software interrupt handling.

Caution Do not use the RETI instruction to return from software interrupt handling.

12.3.2 Accepting Nonmaskable Interrupts

A nonmaskable interrupt request is accepted regardless of whether interrupts are enabled. It is not subjected to priority control. Instead it has precedence over all other interrupt requests.

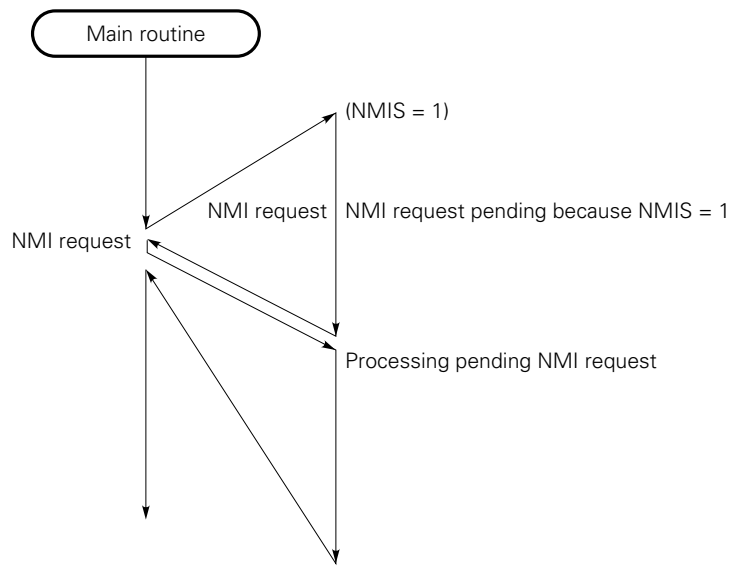
When a nonmaskable interrupt request is accepted, the NMIS bit of the interrupt status register (IST) is set to 1, the PSW and PC are saved in the stack in the stated order, the IE and ISP flags are reset to 0, and the PC is loaded with the contents of the vector table to cause a branch. When the RETI instruction is executed, the NMIS bit is reset to 0.

If the NMIS bit is 1, a new nonmaskable interrupt request is not accepted. It is kept pending until the NMIS bit is reset to 0. In other words, when a nonmaskable interrupt service program is running, a new nonmaskable interrupt request is not accepted. If a new nonmaskable interrupt request occurs when a nonmaskable interrupt service program is already running, it is accepted after the service program ends (an RETI instruction is executed). In this case, if two nonmaskable interrupt requests occur, only one request is accepted after the current service program ends.

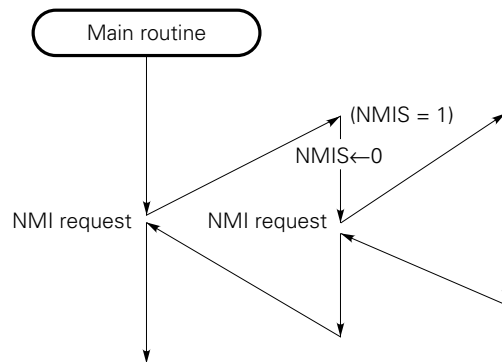
Resetting the NMIS bit to 0 during execution of a nonmaskable interrupt service program enables multiple-interrupt handling for nonmaskable interrupt requests. If the NMIS bit is 0, a new nonmaskable interrupt request is accepted even when a nonmaskable interrupt service program is running.

Fig. 12-9 Accepting an NMI Interrupt Request

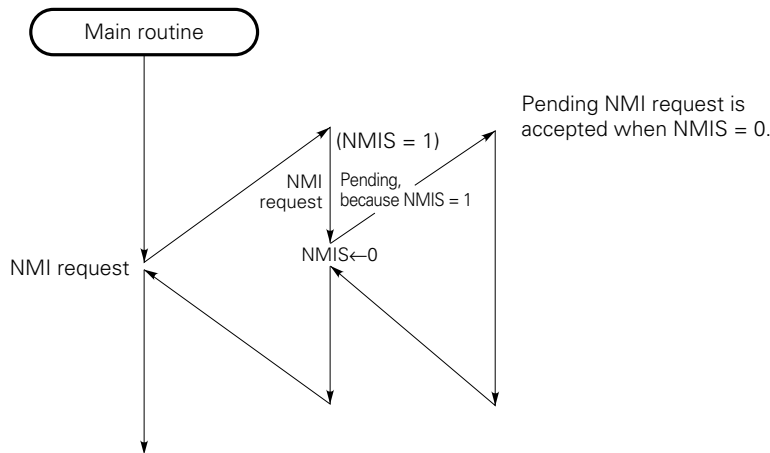
- (a) If a new NMI request occurs during execution of an NMI service program (when the IST register is not manipulated)



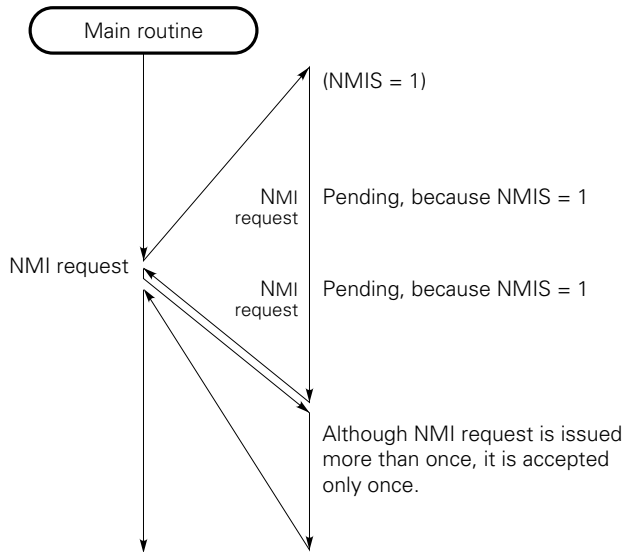
- (b) If a new NMI request occurs during execution of an NMI service program (when the NMIS bit is reset to 0 by the current NMI service program)



- (c) If a new NMI request occurs during execution of an NMI service program (when the NMIS bit is reset to 0 by the current NMI service program after the NMI request occurs)



- (d) If two new NMI requests occur during execution of an NMI service program (when the NMIS bit is not manipulated by the current NMI service program)



- Cautions**
1. A macro service request is accepted and processed even when a nonmaskable interrupt service program is running. To disable macro service processing during execution of the nonmaskable interrupt service program, cause the nonmaskable interrupt service program to manipulate the mask register so that no macro service will not occur.
 2. If the IE bit of the PSW is set to 1, for example, by executing the EI instruction in the nonmaskable interrupt service program, maskable interrupt requests assigned high priority are made acceptable. If a maskable interrupt with high priority occurs during execution of the nonmaskable interrupt service program, a service program for the maskable interrupt runs. If the IE and ISP bits of the PSW are set to 1, interrupt requests with low priority will also occur, thus causing the interrupt service program to run. An RETI instruction will be used to return from the maskable interrupt service program. The RETI instruction resets the NMIS bit to 0, thus enabling nonmaskable interrupt requests even when multiple-interrupt handling should not be performed for nonmaskable interrupts during execution of the nonmaskable interrupt service program to which a return was just made. To inhibit multiple-interrupt handling for nonmaskable interrupt requests, do not enable interrupts during execution of the nonmaskable interrupt service program.

3. Nonmaskable interrupts are always accepted except during execution of the nonmaskable interrupt handling program (except when multiple-interrupt handling for nonmaskable interrupts have been enabled by resetting the NMIS bit of the IST register to 0 during execution of the nonmaskable interrupt handling program) and except a period between a special instruction described in Section 12.3.5 and an instruction that follows that special instruction. Therefore, nonmaskable interrupts are accepted, even if the contents of the stack pointer are undefined, for example, right after a reset occurs. At this point, the contents of the PC and PSW may be transferred to addresses (see Table 3-4 in Section 3.2.5) where writing to any special-function register is inhibited, depending on the value in the stack pointer. If this occurs, the CPU may hang, unexpected signals may be output from pins, or an attempt may be made to transfer the contents of the PC or PSW to a location where no RAM has been installed, thereby making it impossible to return from the nonmaskable interrupt handling program to the main routine, hence a program crash.

If a falling edge (valid edge of the NMI input after a reset) arrives at the NMI pin at much the same time when a rising edge is supplied to the RESET pin, a branch occurs to the nonmaskable interrupt handling program without executing a single instruction after a reset, resulting in a program crash almost with no exception. To avoid these problems, initialize the stack pointer after a reset, and design the hardware so that the NMI signal does not drop within $10 \mu\text{s} + 20/f_{\text{CLK}}$ after the RESET signal rises. ★

12.3.3 Accepting Maskable Interrupts

A maskable interrupt is accepted when the corresponding interrupt request flag is set to 1, if the corresponding interrupt mask flag is 0. When a macro service is used, the interrupt is accepted immediately when the interrupt flag is set to 1. If it is a vectored interrupt, it is accepted when interrupts are enabled (when the IE flag is 1). However, low-priority interrupts are not accepted if a high-priority interrupt is already being serviced (when the ISP flag is 0).

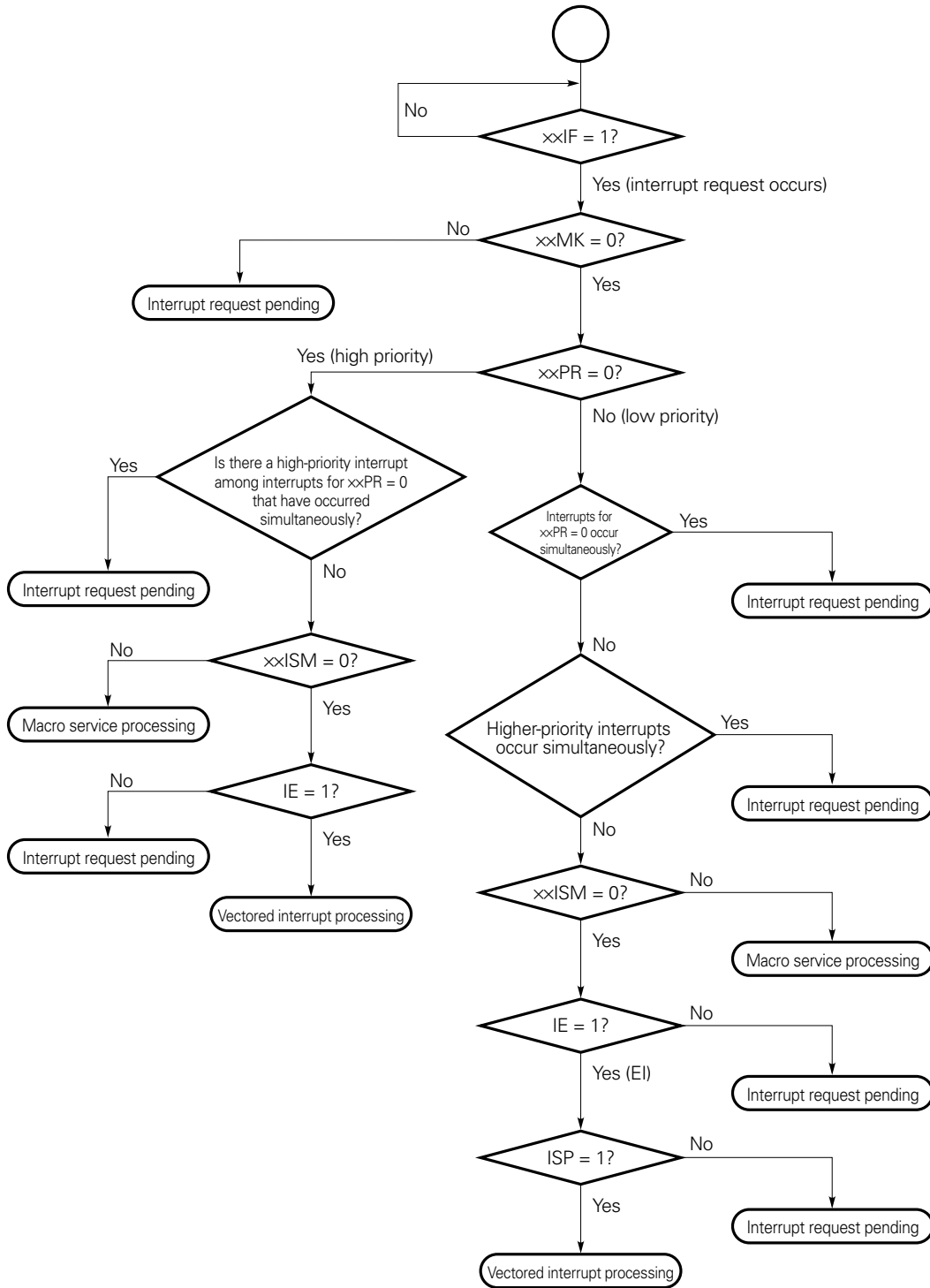
If two or more maskable interrupt requests occur simultaneously, they are accepted according to their priority (as specified in the priority specification flag). If interrupt requests have the same priority, an interrupt request with the highest default priority is accepted first.

A pending interrupt request is accepted when it is enabled.

Fig. 12-10 shows the algorithm for accepting interrupt requests.

When a maskable interrupt request is accepted, the contents of the PSW and PC are saved in the stack in the stated order, the IE flag is reset to 0 (interrupt disabled state), the content of the priority specification flag corresponding to the accepted interrupt request is transferred to the ISP. Moreover, the PC is loaded with the contents of the vector table corresponding to the accepted interrupt request, thus causing a branch. The RETI instruction is used to return from the interrupt.

Fig. 12-10 Interrupt Handling Algorithm



12.3.4 Multiple-Interrupt Handling

The μ PD78214 performs multiple-interrupt handling in which another interrupt request is accepted during one interrupt is already being handled. Multiple-interrupt handling runs according to priority.

Priority control is based on either default priority or programmable priority specified in the priority specification flag register (PR0). Priority control by default priority handles interrupt requests according to the default priority assigned to each interrupt request (see **Table 12-2**). Priority control by programmable priority divides interrupt requests into a high-priority group and a low-priority group according to the corresponding bit of the PR0 register. Table 12-4 lists interrupt requests that can be subjected to multiple-interrupt handling.

Table 12-4 Multiple-Interrupt Handling

Interrupt request accepted	IE flag	Interrupt request source that can be subjected to multiple-interrupt handling
Interrupts assigned low programmable priority	0	<ul style="list-style-type: none"> • Nonmaskable interrupt • Maskable interrupt by macro service processing
	1 ^{Note 1}	<ul style="list-style-type: none"> • Nonmaskable interrupt • All maskable interrupts
Interrupts assigned high programmable priority	0	<ul style="list-style-type: none"> • Nonmaskable interrupt • Maskable interrupt by macro service processing
	1 ^{Note 1}	<ul style="list-style-type: none"> • Nonmaskable interrupt • Maskable interrupt by macro service processing • Maskable interrupt assigned high programmable priority
Nonmaskable interrupt	0	<ul style="list-style-type: none"> • Nonmaskable interrupt^{Note 2} • Maskable interrupt by macro service processing
	1 ^{Note 1}	<ul style="list-style-type: none"> • Nonmaskable interrupt^{Note 2} • Maskable interrupt by macro service processing • Maskable interrupt assigned high programmable priority^{Note 3}

- Notes**
1. Immediately after an interrupt is accepted, interrupts are disabled (IE = 0) automatically. To enable interrupts (IE = 1), execute the EI instruction.
 2. When a nonmaskable interrupt request is being accepted, bit 0 (NMIS) of the interrupt status register (IST) is 1. When the NMIS bit is 1, nonmaskable interrupt requests are not accepted. To enable multiple-interrupt handling for nonmaskable interrupt requests, reset the NMIS flag to 0 by software.
 3. When the ISP flag is 1, low priority interrupt requests are accepted.

Fig. 12-11 Example of Handling an Interrupt Request When an Interrupt Is Already Being Handled (1/2)

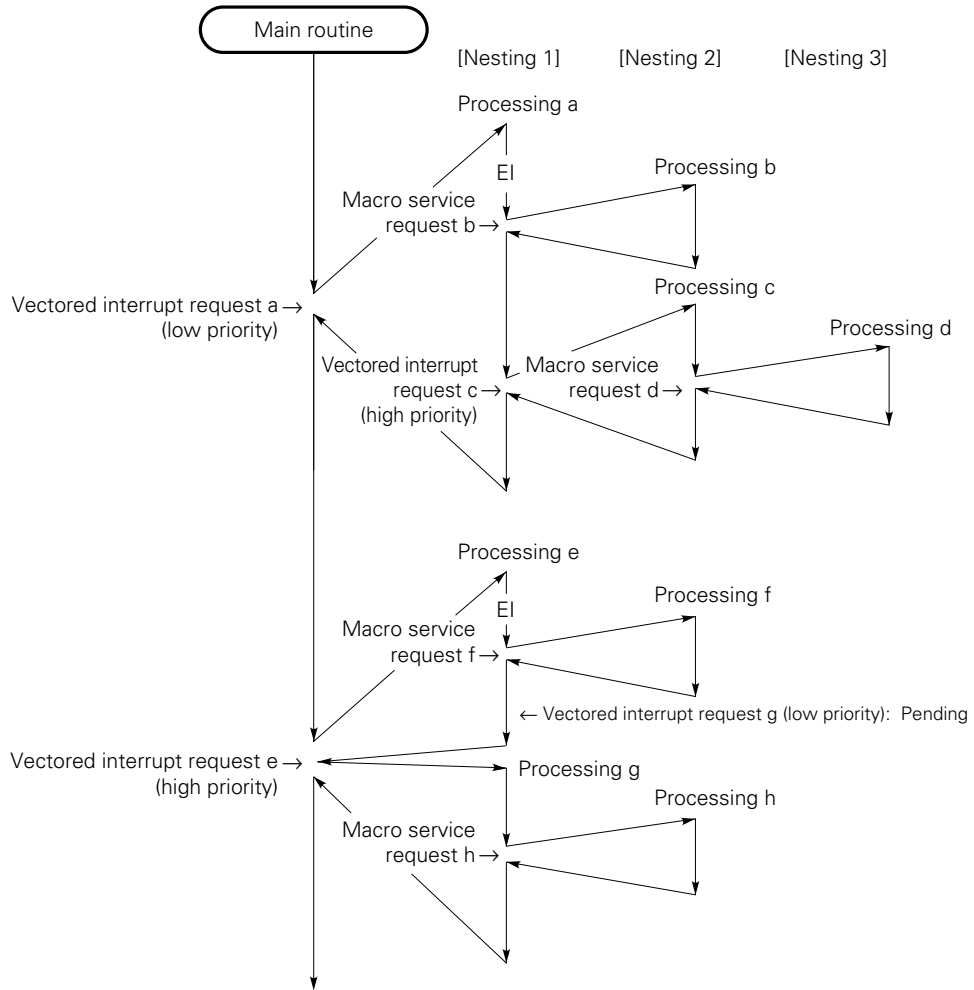


Fig. 12-11 Example of Handling an Interrupt Request When an Interrupt Is Already Being Handled (2/2)

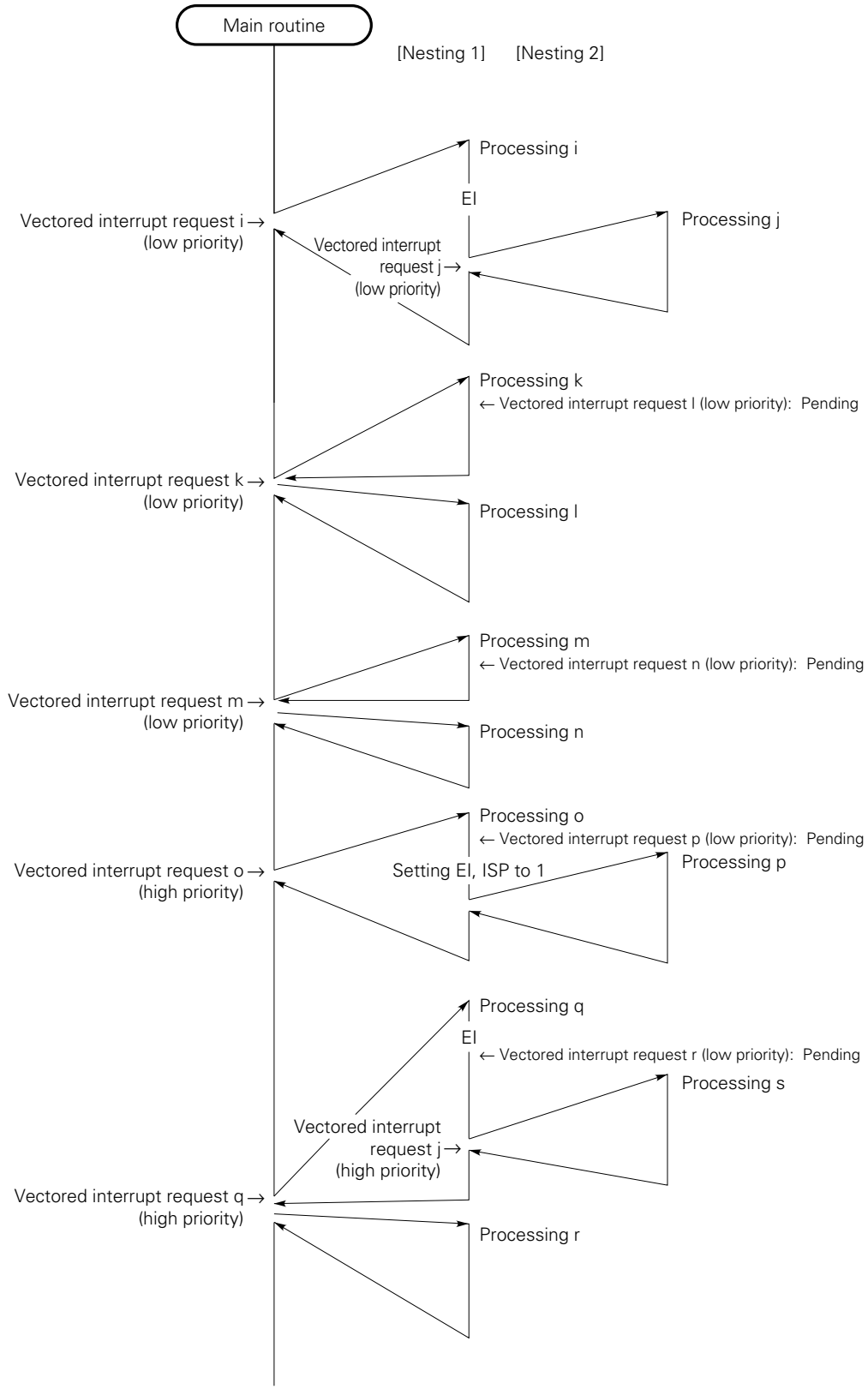
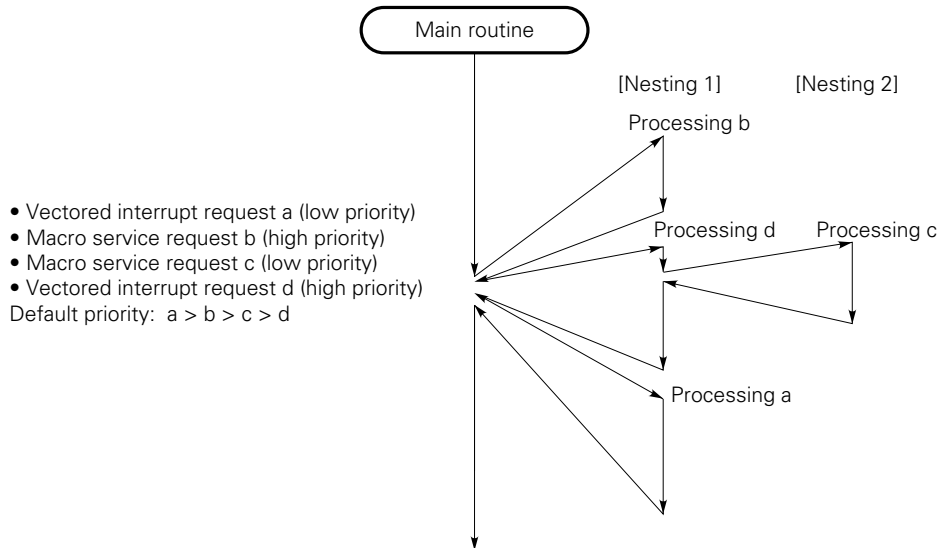


Fig. 12-12 Example of Handling Interrupts That Occur Simultaneously



12.3.5 Interrupt Request and Macro Service Pending

When any of the following instructions is executed, all interrupts (including nonmaskable interrupts) and macro services are kept pending. The pending state continues until another instruction is executed. Software interrupt requests are not kept pending, however.

- EI
- DI
- RETI
- RETB
- POP PSW
- MOV PSW,A
- MOV PSW,#byte
- IST, MK0, IF0, PR0, and ISM0 manipulation instructions
- PSW bit manipulation instructions (excluding BT PSW.bit, \$addr16, BF PSW.bit, \$addr16, SET1 CY, NOT1 CY, and CLR1 CY instructions)

Cautions 1. When a BF instruction is used to poll registers related to interrupts, do not specify this BF instruction as the branch destination. Otherwise, all interrupts and macro services are kept pending until a condition that inhibits a branch is met during execution of the instruction.

```

Example of incorrect coding
:
:
LOOP:  BF IF0H.3, $LOOP    ← All interrupts and macro services are kept pending, until
:                               IF0H.3 is set to 1. The pending state continues until the
:                               instruction next to the BF is executed.
:                               ×××
:
:
Example of correct coding (1)
:
:
LOOP:  NOP                ← Interrupts or macro services will not be kept pending long,
:                               because they are processed after the NOP is executed.
:                               BF IF0H.3, $LOOP
:
:

```

Example of correct coding (2)

```

:
:
LOOP: BT IF0H.3, $NEXT
      BR $LOOP
NEXT:
:
:

```

Remark The BTCLR would be more convenient than the BT, because it clears the flags automatically.

← Interrupts or macro services will not be kept pending long, because they are processed after the BR is executed.

2. In addition, when you have to use a coding of the instructions listed above consecutively, yet expect frequent occurrence of interrupts and macro services, insert NOP instructions in the coding to allow time during which interrupts and macro service can be accepted.

12.3.6 Interrupt and Macro Service Operation Timing

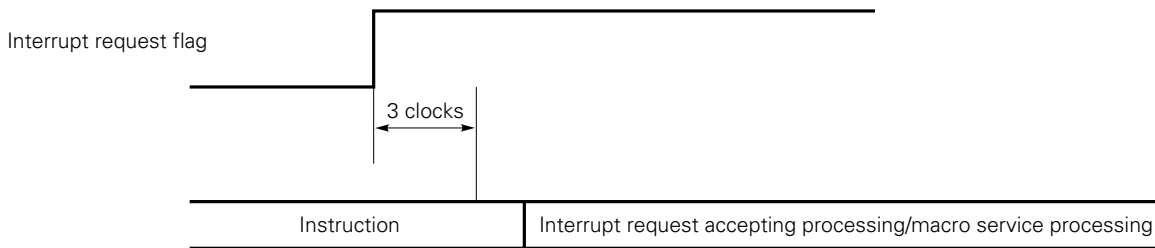
(1) Generation and acceptance of an interrupt request

An interrupt request is generated by hardware. The generated interrupt request sets the corresponding interrupt request flag to 1.

When the interrupt request flag is set to 1, three clocks (0.5 μs at f_{CLK} = 6 MHz) are required to identify the priority of the interrupt request.

If the acceptance of the interrupt request is allowed when the current instruction has been executed, the interrupt request is accepted. If the current instruction is one that keeps interrupt requests and macro services pending, the interrupt request is accepted after the instruction next to that instruction has been executed. (See Section 12.3.5 for instructions that keep interrupt requests and macro services pending.)

Fig. 12-13 Interrupt Request Generation and Acceptance (Unit: Clock)



(2) Interrupt request acceptance time

The time listed in Table 12-5 is required to accept each interrupt request. The interrupt handling program starts running after the time listed in Table 12-5 has elapsed.

Table 12-5 Interrupt Request Acceptance Processing Time^{Note}

(Unit: Clock)

Stack area	Internal RAM	Peripheral RAM	External memory
Program fetch			
Internal ROM fetch	18	24	24 + w × 3
External ROM fetch	24 + w × 3	30 + w × 3	30 + w × 6

(w = number of wait cycles)

Note The time listed here does not include the time that elapses before the current instruction is completed or the time required to identify the priority of the interrupt request.

- Remarks**
1. The values on the "Internal ROM fetch" row apply when the program is fetched from the internal ROM with IFCH bit of the memory expansion mode register (MM) set to 1. If the IFCH bit is 0, the same values as when the program is fetched from an external ROM apply.
 2. "Internal RAM" is located at addresses 0FE00H through 0FEFFH.

3. "Peripheral RAM" corresponds to the internal RAM at addresses 0FC80H through 0FDFFH (for the μPD78212, 0FD80H through 0FDFFH).
4. 1 clock = 1/f_{CLK} (167 ns at 12 MHz).

(3) Macro service processing time

The time required to process a macro service varies, depending on the type of the macro service, as listed in Table 12-6.

Table 12-6 Macro Service Processing Time^{Note}

(Unit: Clock)

macro service processing type		Program fetch	Internal ROM fetch			External ROM fetch		
		Memory	Internal ROM	Internal RAM	External memory	Internal ROM	Internal RAM	External memory
A	Memory to SFR	—	19	—	—	21 + w	—	
	SFR to memory	—	20	—	—	22 + w	—	
B	Memory to SFR	30	29	31 + w	32 + w	31 + w	33 + 2w	
	SFR to memory	—	31	33 + w	—	33 + w	35 + 2w	
C	Without ring control	Data transfer	37	35	39 + 2w	39 + w	37 + w	41 + 3w
		Automatic addition	41	39	43 + 2w	43 + w	41 + w	45 + 3w
	With ring control	Data transfer	42/47	40/45	44 + 2w/ 49 + 2w	44 + w/ 49 + w	42 + w/ 47 + w	46 + 3w/ 51 + 3w
		Automatic addition	46/51	44/49	48 + 2w/ 53 + 2w	48 + w/ 53 + w	46 + w/ 51 + w	50 + 3w/ 55 + 3w

(w = number of wait cycles)

Note The time listed here does not include the time that elapses before the current instruction is completed or the time required to identify the priority of the interrupt request.

- Remarks**
1. The values in the "Internal ROM fetch" column apply when the IFCH bit of the memory expansion mode register (MM) is 1. If the IFCH bit is 0, see the "External ROM fetch" column.
 2. The values in the "Internal RAM" column apply when an internal RAM at 0FE00H through 0FEFFH is used. If other areas in the internal RAM are used, the values in the "External memory" column apply after w is reset to 0.
 3. The values on the right of "/" apply when the ring counter is 0. The values on the left of "/" apply when the ring counter is other than 0.
 4. 1 clock = 1/f_{CLK} (167 ns at 12 MHz)
 5. For types A and B, an additional wait time of up to 15 clocks is inserted. See (8) of Section 7.5.1 and Table 7-20 in Chapter 7 for details.
 6. The number of clocks for type C does not include wait states inserted for access to the CR10 or CR11. If the wait time is inserted, the values in the list must be increased by one clock.

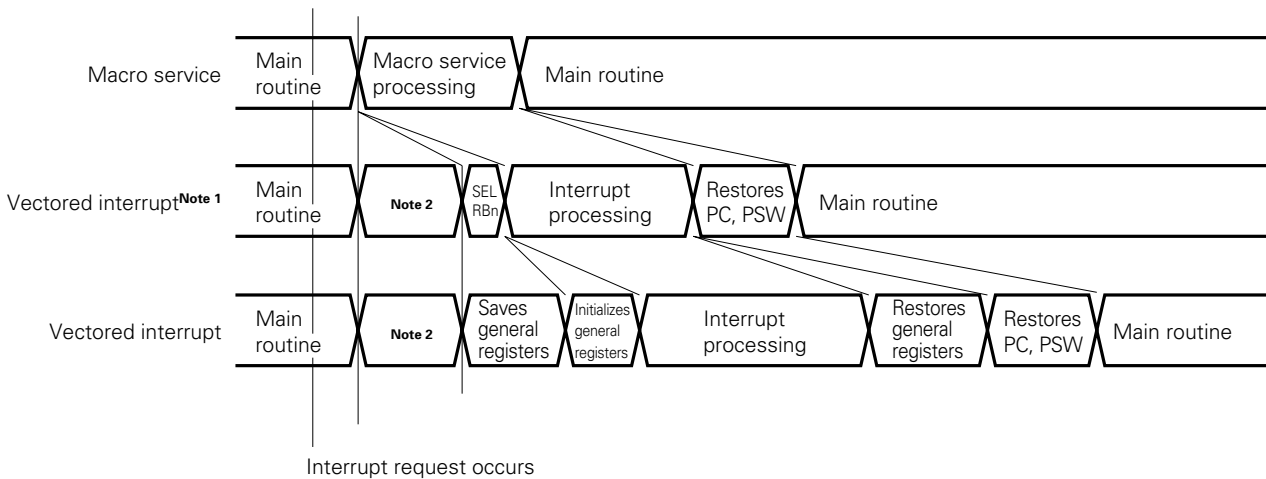
12.4 MACRO SERVICE FUNCTION

12.4.1 Macro Service Outline

Macro service is one of the interrupt handling methods. When a vectored interrupt is processed, the contents of the program counter (PC) and the program status word (PSW) are saved in the stack and the PC is loaded with the vector address retrieved from the vector table. With the macro service function, a different type of processing (mainly data transfer) is performed. This type of processing can respond to an interrupt request quickly. Moreover, it transfers data much faster than the program does, shortening the required processing time significantly.

With the macro service function, a vectored interrupt can be generated after processing has been performed a specified number of times, so that the vectored interrupt program can be simplified.

Fig. 12-14 Differences between a Vectored Interrupt and Macro Service



- Notes 1. This timing chart applies when the register bank switching function is used and the registers are loaded with the initial values in advance.
- 2. The PC and PSW contents are saved in the stack and the PC is loaded with the vector address.

12.4.2 Macro Service Types

The macro service can be used by the 17 types of interrupts listed in Table 12-7 (of which, 15 types can use macro services simultaneously). In addition, three modes of operation are available, and each should be selected according to the application.

Table 12-7 Interrupts That Can Use a Macro Service

Interrupt request source	Generating unit	Macro service type	Special function register used for type A
INTP0 (edge input to the pin is detected)	Edge detection	A, B	CR11
INTP1 (edge input to the pin is detected)		A, B	CR22
INTP2 (edge input to the pin is detected)		A, B	TM2
INTP3 (edge input to the pin is detected)		B	—
INTC00 (TM0-CR00 coincidence signal generation)	16-bit timer/counter	B	—
INTC01 (TM0-CR01 coincidence signal generation)		B	—
INTC10 (TM1-CR10 coincidence signal generation)	8-bit timer/counter 1	A, B, C	CR10
INTC11 (TM1-CR11 coincidence signal generation)		A, B, C	CR11
INTC21 (TM2-CR21 coincidence signal generation)	8-bit timer/counter 2	A, B	CR21
INTP4 (edge input to the pin is detected)	Edge detection	B	—
INTC30 (TM3-CR30 coincidence signal generation)	8-bit timer/counter 3	A, B	CR30
INTP5 (edge input to the pin is detected)	Edge detection	B	—
INTAD (A/D conversion end)	A/D converter	A, B	ACDR
INTC20 (TM2-CR20 coincidence signal generation)	8-bit timer/counter 2	A, B	CR20
INTSR (asynchronous serial interface reception end)	Asynchronous serial interface	A, B	RxB
INTST (asynchronous serial interface transmission end)		A, B	TxB
INTCSI (clock-synchronized serial interface transmission end)	Clock-synchronized serial interface	A, B	SIO

The following three types of macro services are available:

(1) Type A

Transfers 1-byte data between a special function register (SFR) and memory upon each interrupt request. When a specified number of data transfers are performed, a vectored interrupt request is generated.

The SFR with which data is to be transferred is predetermined for each interrupt request. In addition, the memory to be used is fixed at addresses 0FE00H through 0FEFFH in the internal RAM.

This macro service is easily specified and is usable for transfer of a small amount of data at high speed.

(2) Type B

Similarly to type A, transfers 1-byte data between an SFR and memory upon each interrupt request. When a specified number of data transfers are performed, a vectored interrupt request is generated.

The SFR and memory between which data is to be transferred are specified by the macro service channel (the memory is a 64K-byte area at addresses 0000H through FFFFH).

This macro service is a general-purpose version of type A. It is suitable for transfer of a large amount of data.

(3) Type C

Transfers 1-byte data from memory to the real-time output port and the compare register for 8-bit timer/counter 1 upon each interrupt request. When a specified number of data transfers are performed, a vectored interrupt request is generated.

Type C macro service transfers data to two locations upon one interrupt request. In addition, it can be used together with output data ring control and automatic addition of the compare register contents to data.

Use of type C is limited to INTC10 and INTC11. The SFRs to which data can be transferred are also limited. A 64K-byte memory area, addresses 0000H through FEFFH, can be used.

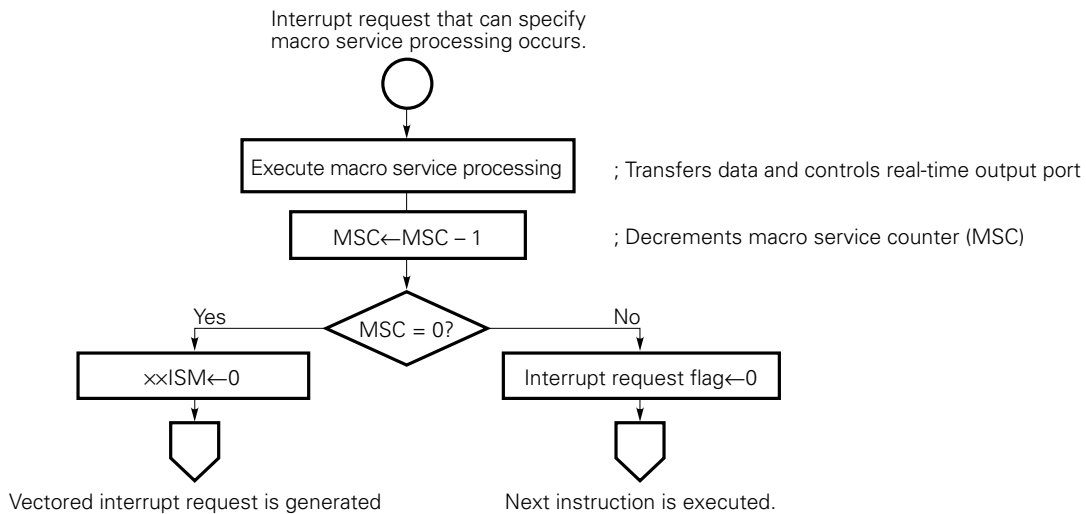
Type C is a macro service for real-time output port and is suitable for controlling stepper motors.

12.4.3 Macro Service Basic Operation

The interrupt request that can specify the macro service processing and that is generated by the algorithm shown in Fig. 12-10 is handled basically by using the sequence shown in Fig. 12-15.

An interrupt request that can specify a macro service is not affected by the state of the IE flag. It is disabled by setting the interrupt mask flag in the interrupt mask register (MK0) to 1. The macro service processing is performed regardless of whether interrupts are disabled and whether a macro service processing is already being performed.

Fig. 12-15 Macro Service Processing Sequence



The type of a macro service and the direction of its data transfer are determined by the value in the mode register in the macro service control word. Then the macro service channel specified by a channel pointer is used according to the type of macro service for data transfer.

A macro service channel contains the macro service counter to hold the number of data transfers to be performed, transfer destination and source pointers, and data buffers. It is mapped to locations from FE00H through FEFFH in the internal RAM.

12.4.4 Macro Service Control Register

(1) Macro service control word

The macro service function of the μPD78214 is controlled using the macro service mode registers and macro service channel pointers. The macro service mode registers specify the mode of macro service processing, and the macro service channel pointer specifies the address of a macro service channel to be used.

The macro service mode registers and macro service channel pointers are mapped in the internal RAM as macro service control words for individual macro services as shown in Fig. 12-16.

To perform macro service, it is necessary to set values in the macro service mode register and channel pointer corresponding to an interrupt request that can be processed by macro service.

Fig. 12-16 Macro Service Control Word Configuration

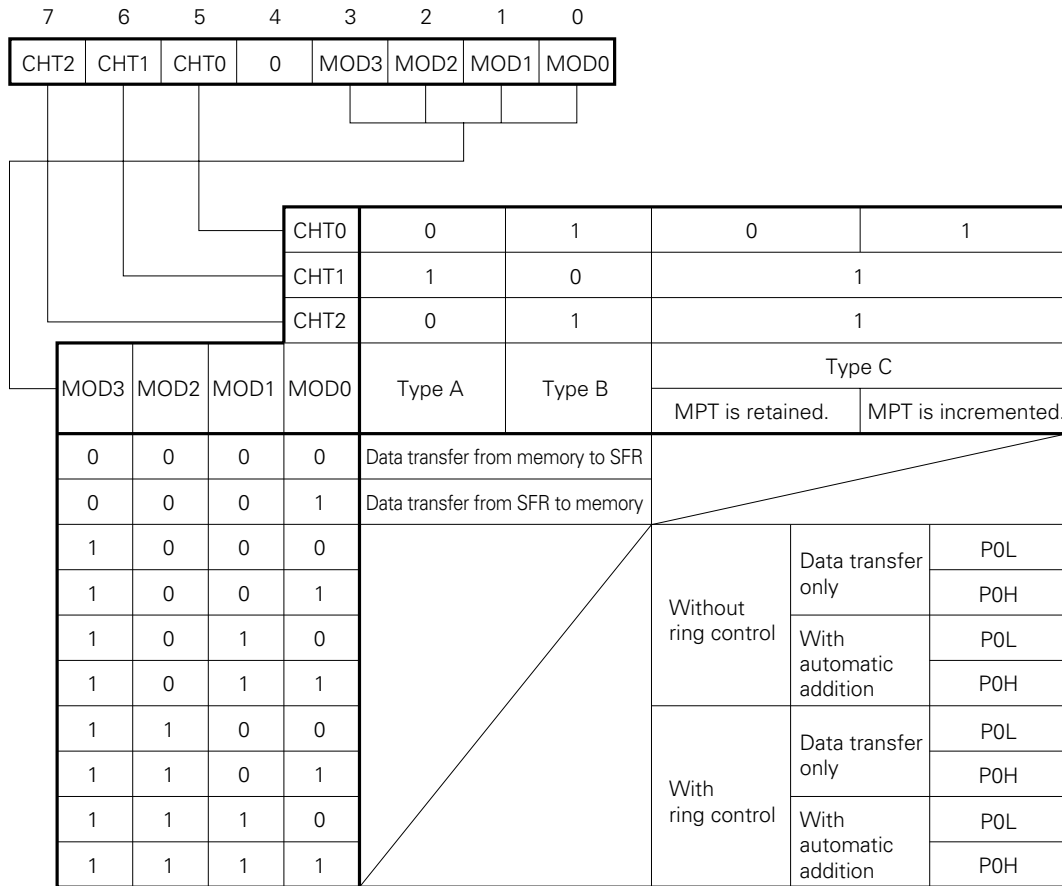
0FEDFH	Channel pointer	} INTSR
0FEDEH	Mode register	
0FEDDH	Channel pointer	} INTST
0FEDCH	Mode register	
0FEDBH	Channel pointer	} INTCSI
0FEDA H	Mode register	
0FED9H	Channel pointer	} INTC10
0FED8H	Mode register	
0FED7H	Channel pointer	} INTC11
0FED6H	Mode register	
0FED5H	Channel pointer	} INTP4/INTC30
0FED4H	Mode register	
0FED3H	Channel pointer	} INTP5/INTAD
0FED2H	Mode register	
0FED1H	Channel pointer	} INTC00
0FED0H	Mode register	
0FECFH	Channel pointer	} INTC01
0FECEH	Mode register	
0FECDH	Channel pointer	} INTC20
0FECCH	Mode register	
0FECBH	Channel pointer	} INTC21
0FECAH	Mode register	
0FEC9H	Channel pointer	} INTP0
0FEC8H	Mode register	
0FEC7H	Channel pointer	} INTP1
0FEC6H	Mode register	
0FEC5H	Channel pointer	} INTP2
0FEC4H	Mode register	
0FEC3H	Channel pointer	} INTP3
0FEC2H	Mode register	

(2) Macro service mode register

A macro service mode register is an 8-bit register that specifies the mode of macro service operation. It is mapped in internal RAM as part of macro service control word (see Fig 12-16).

Fig. 12-17 shows the format of the macro service mode register.

Fig. 12-17 Macro Service Mode Register Format



(3) Macro service channel pointer

A macro service channel pointer specifies the address of a macro service channel. The macro service channel can be located in a 256-byte area in the internal RAM at addresses FE00H through FEFFH. The higher 8 bits of the address are fixed. Therefore, the macro service channel pointer specifies the lower 8 bits of the highest address of the macro service channel.

12.4.5 Macro Service Type A

(1) Operation

The type A macro service transfers data between the buffer memory in the macro service channel and the SFR predetermined for an individual interrupt request.

For this macro service, the direction of data transfer can be specified as either “from memory to SFR” or “from SFR to memory.”

Data transfer is repeated as many times as previously specified in the macro service counter. The macro service transfers 8-bit data.

The type A macro service is suitable for transfer of a limited amount of data at high speed.

The 12 types of interrupt requests listed in Table 12-8 can be specified for the type A macro service. Table 12-8 also lists the SFR registers that can be specified as transfer destinations and sources.

Table 12-8 Interrupt Requests That Can Specify Macro Service and Related SFRs (Type A)

Interrupt request specifying the type A macro service	Transfer source/destination SFR
INTC10	CR10 register
INTC11	CR11 register
INTC20	CR20 register
INTC21	CR21 register
INTC30	CR30 register
INTSR	RXB register
INTST	TXS register
INTCSI	SIO register
INTAD	ADCR register
INTP0	CR11 register
INTP1	CR22 register
INTP2	TM2 register

Caution When the external memory is expanded (or always with the μPD78213), an illegal write access operation may occur during the type A macro service.

This illegal write access occurs when either of the following two conditions is satisfied.

- (1) When data D0H through DFH is transferred from memory to an SFR.
- (2) When macro service transfers data from an SFR to a buffer (memory) at 0FED0H through 0FEDFH.

An illegal write access is processed in the same manner as the normal memory access. In addition, wait states may be inserted according to the setting of the PW20 and PW21 bits of the memory expansion mode register (MM). Table 12-9 lists the conditions under which an illegal write access occurs and the corresponding operations.

Table 12-9 Illegal Write Access Conditions and Corresponding Operations

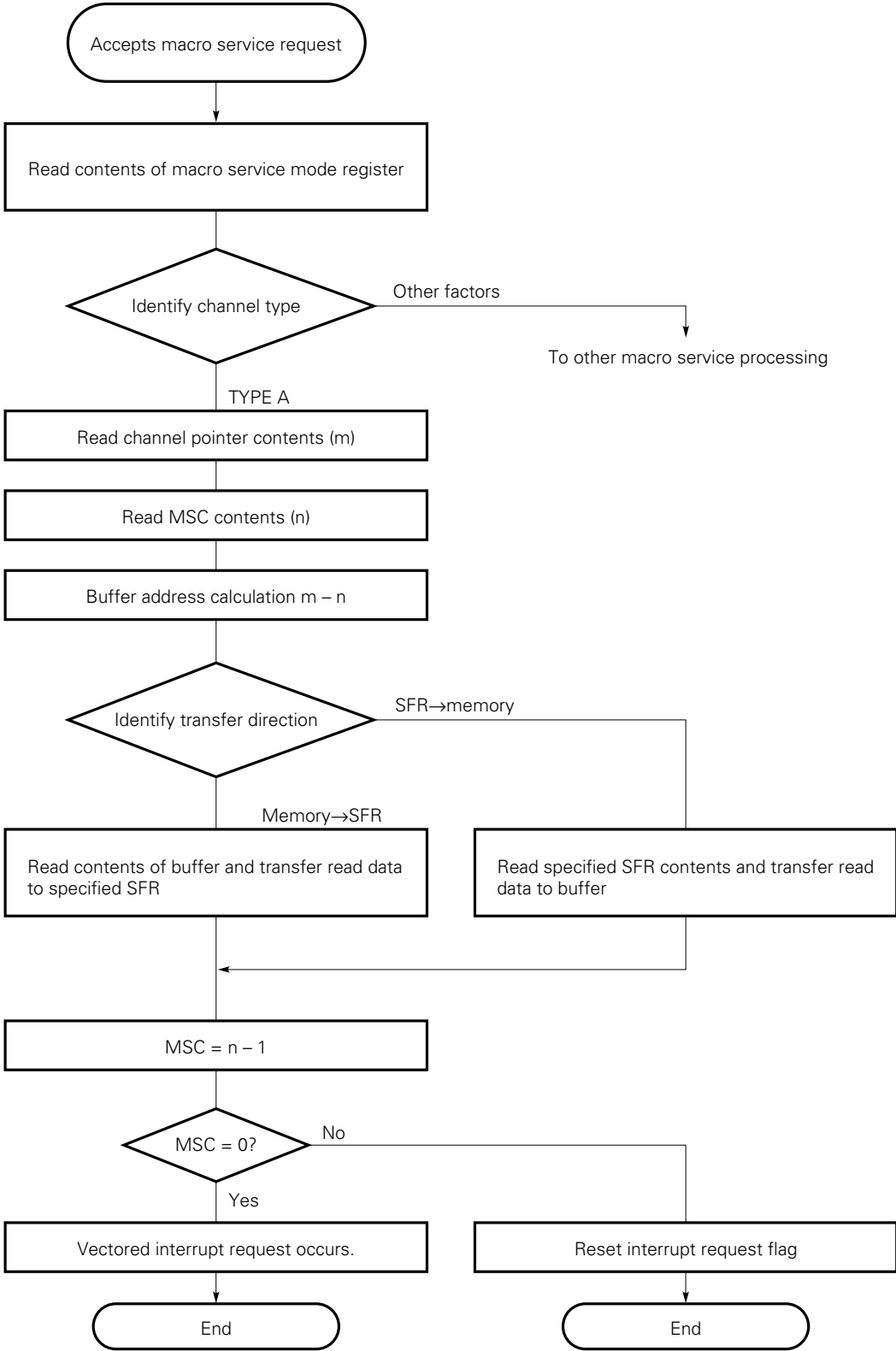
Condition	Illegal write access	
	Address	Data
1	Address of a destination SFR	Data transferred by macro service
2	Address of a source SFR	Lower 8 bits of the address of the destination buffer (memory)

This problem may be solved by either of the following two methods.

- (1) It is difficult for software to solve the problem if it occurs under condition 1, because it depends on the transfer data. Therefore, use an external address decoder circuit to keep the image in the area of 0FF00H through 0FFFFH from overlapping the memory addresses of the external circuit.
- (2) If the macro service to be used does not satisfy condition 1 (i.e., if data is not transferred from an SFR to memory), and under condition 2, locate the buffer area so that its addresses are not 0FED0H through 0FEDFH.

The above problem also occurs with an in-circuit emulator.

Fig. 12-18 Flow of Data Transfer by Macro Service (Type A)

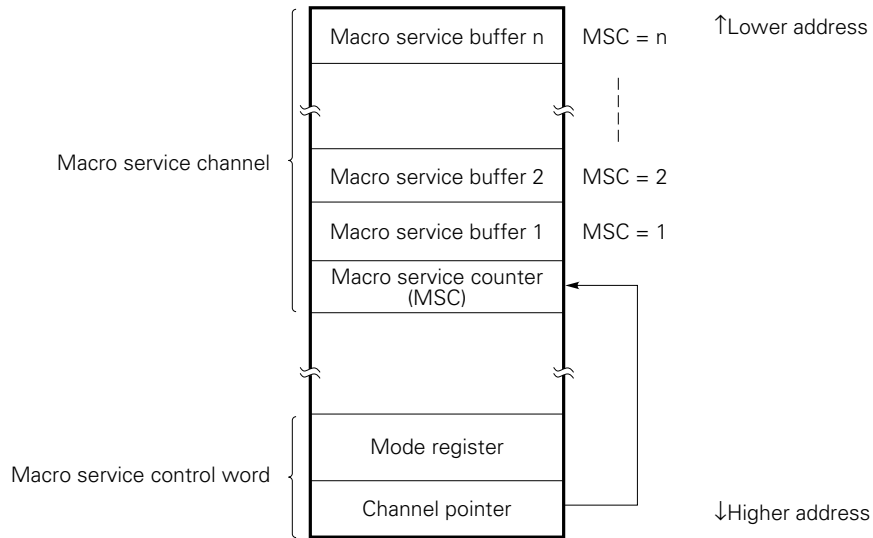


(2) Macro service channel configuration

A channel pointer and a macro service counter (MSC) specify the addresses of transfer source and destination buffers in the internal RAM (at FE00H through FEFFH). (See **Fig. 12-19.**)

The SFR to be accessed is predetermined for each interrupt request. (See **Table 12-8.**)

Fig. 12-19 Type A Macro Service Channel

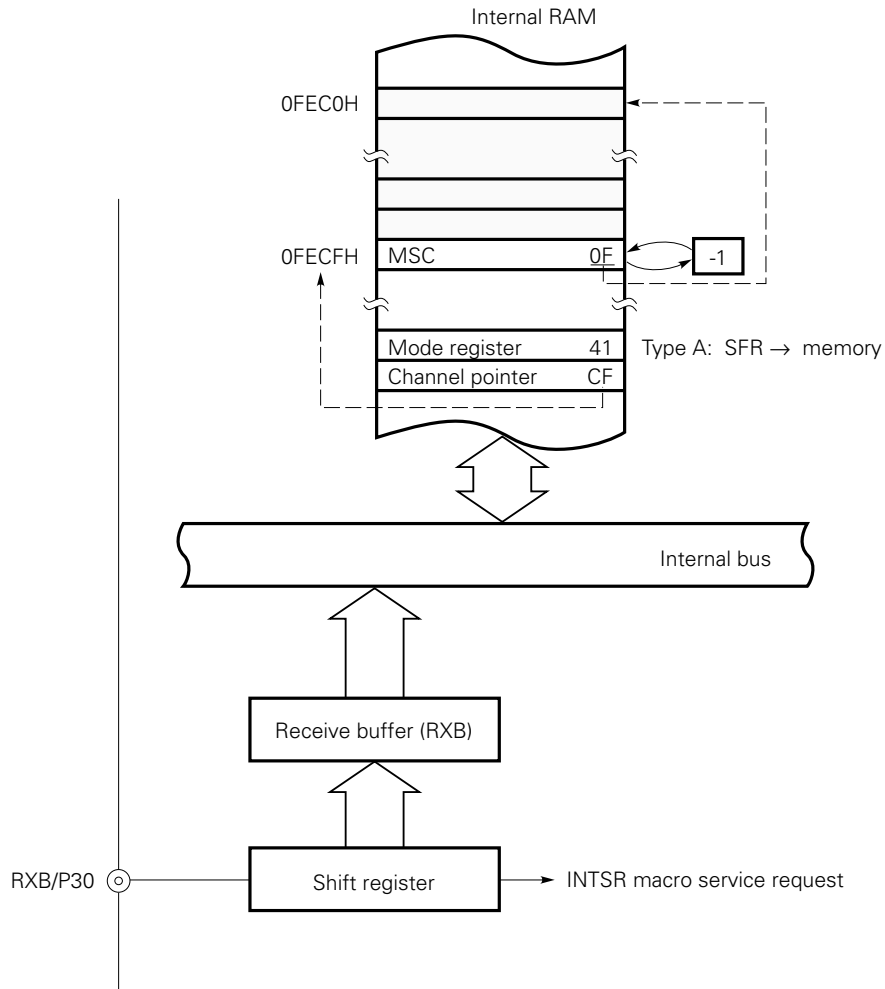


$$(\text{Macro service buffer address}) = (\text{Channel pointer}) - (\text{Macro service counter})$$

(3) Example of using the type A macro service

The following example shows how data received through an asynchronous serial interface is transferred to a buffer area in the internal RAM.

Fig. 12-20 Asynchronous Serial Reception

**12.4.6 Type B Macro Service****(1) Operation**

The type B macro service transfers data between a data area in the memory specified by the macro service channel and an SFR.

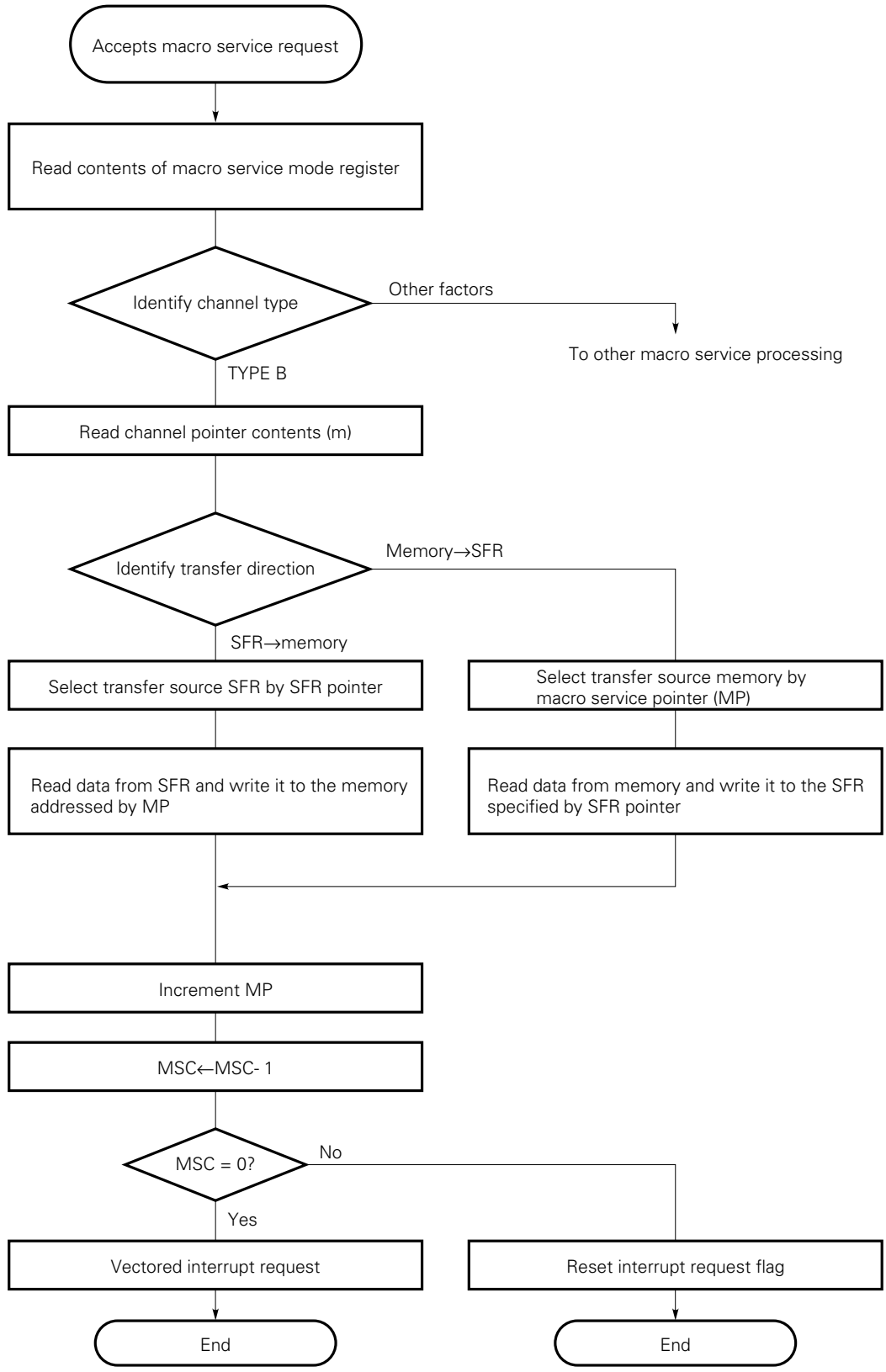
For this macro service, the direction of data transfer can be specified as either “from memory to SFR” or “from SFR to memory.”

Data transfer is repeated as many times as previously specified in the macro service counter. The macro service transfers 8-bit data.

The type B macro service can be specified for all interrupt requests of the μ PD78214 that can activate a macro service. For the type B macro service, the SFR pointers can specify any SFRs for data transfer sources and destinations.

This macro service is a general-purpose version of the type A macro service. It has a data buffer area in a 64K-byte address space and therefore is suitable for transfer of a large amount of data.

Fig. 12-21 Flow of Data Transfer by Macro Service (Type B)



(2) Macro service channel configuration

The macro service pointer (MP) indicates a data buffer area in the 64K memory space as a transfer source or destination.

The SFR pointer (SFRP) is set with the lower 8 bits of the address of an SFR used as a transfer source or destination.

The macro service counter (MSC) specifies the number of data transfers to be performed.

The macro service channel, which holds the macro service pointer, SFR pointer, and macro service counter, is located at addresses 0FE00H through 0FEFFH in the internal RAM space.

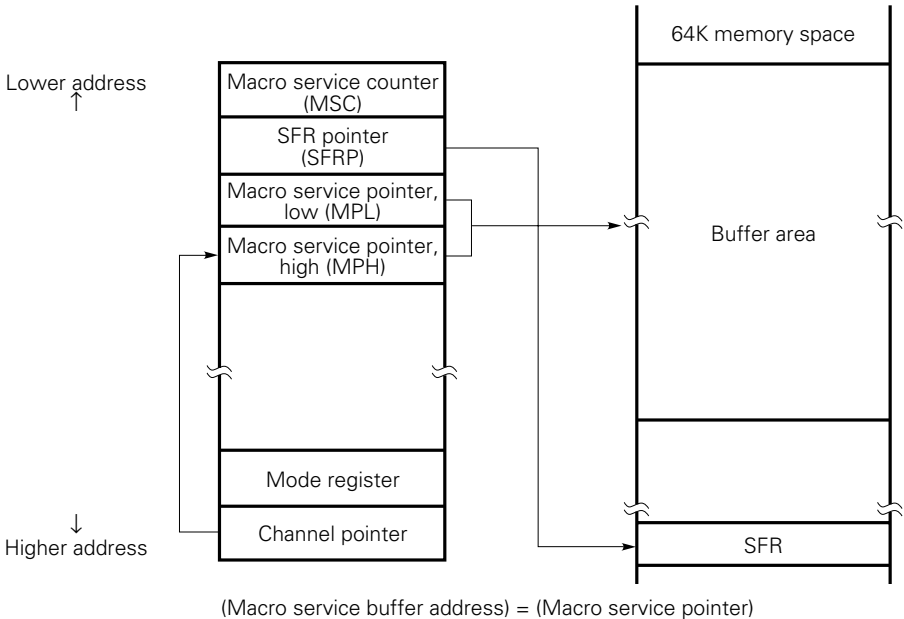
The channel pointer indicates the macro service channel, as shown in Fig. 12-22. The channel pointer holds the lower 8-bits of the address.

Caution The following registers cannot be used as SFRs.

IF0L, IF0H, MK0L, MK0H, PR0L, PR0H, ISM0L, ISM0H, and IST



Fig. 12-22 Type B Macro Service Channel



(3) Example of using the type B macro service

The following example shows how parallel data is input from port 3 in synchronization with an external signal. The external signal is input to the external interrupt pin (INTP4).

Fig. 12-23 Parallel Data Input in Synchronization with an External Interrupt

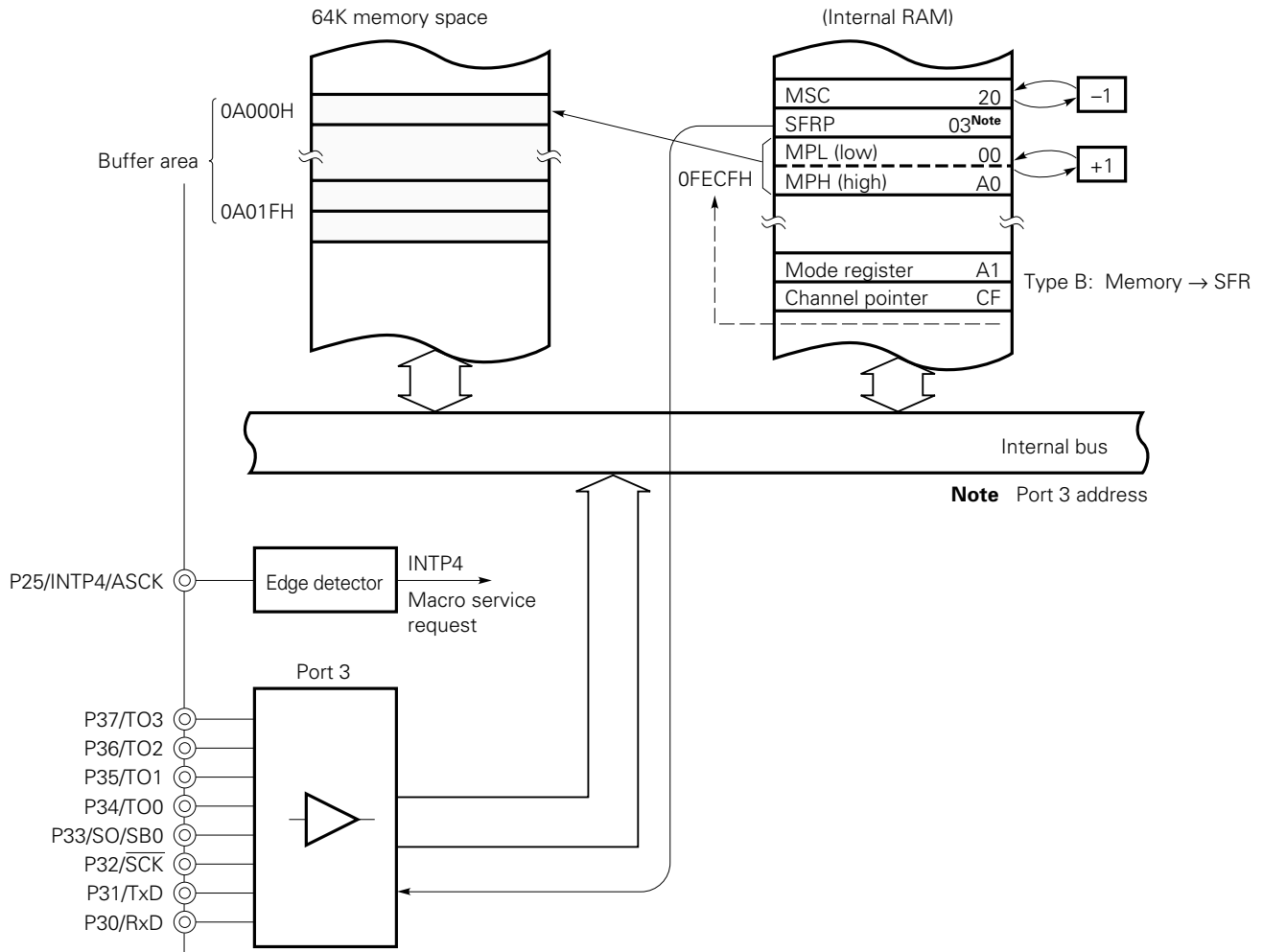
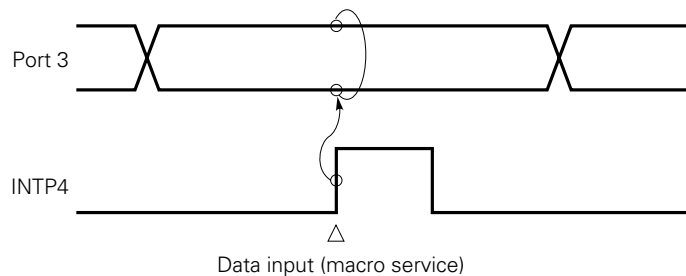


Fig. 12-24 Parallel Data Input Timing



12.4.7 Macro Service Type C

(1) Operation

The type C macro service controls 8-bit timer/counter 1 and the real-time output port simultaneously. This macro service transfers data to both the compare register for 8-bit timer/counter 1 and the buffer register for the real-time output port upon one interrupt request.

Only INTC10 and INTC11 can use the type C macro service. Their transfer destination registers are predetermined as listed in Table 12-8.

Table 12-10 Interrupt Requests That Can Specify Macro Service and SFRs (Type C)

Interrupt request	Data transfer destination addressed by the MPT	Data transfer destination addressed by the MPD
INTC10	CR10	P0L or P0H (specified by the mode register)
INTC11	CR11	P0L or P0H (specified by the mode register)

Besides the basic data transfer function described above, the type C macro service can have the following ancillary functions to reduce the required buffer area and load on the software.

(a) Retention of the timer macro service pointer

It can be specified whether to retain or increment the timer macro service pointer (MPT).

(b) Automatic addition

This function adds data addressed by the timer macro service pointer (MPT) to the value in the compare register, and transfers the sum to the compare register.

If this function is not used, data addressed by the MPT is transferred to the compare register.

(c) Ring control

This function repeats to output a pattern of data with a predetermined length automatically.

These ancillary functions are specified in the mode register in the macro service control word.

- Cautions**
1. With the type C macro service, the MPT and MPD are incremented only at the lower 8 bits. If a carry occurs at bit 7 of the MPTL or MPDL, it is ignored; so the higher 8 bits are not affected.
 2. When the external memory is expanded (or always with the μ PD78213), an illegal write access operation may occur during the type C macro service. This illegal write access occurs when the following condition is satisfied.
 - When the MPTL address is 0FED0H through 0FEDFH.

An illegal write access is performed in the same manner as the normal memory access. In addition, wait states may be inserted according to the setting of the PW20 and PW21 bits of the memory expansion mode register (MM). Table 12-11 lists the conditions under which an illegal write access occurs and the corresponding operations.

Table 12-11 Illegal Write Access Conditions and Corresponding Operations

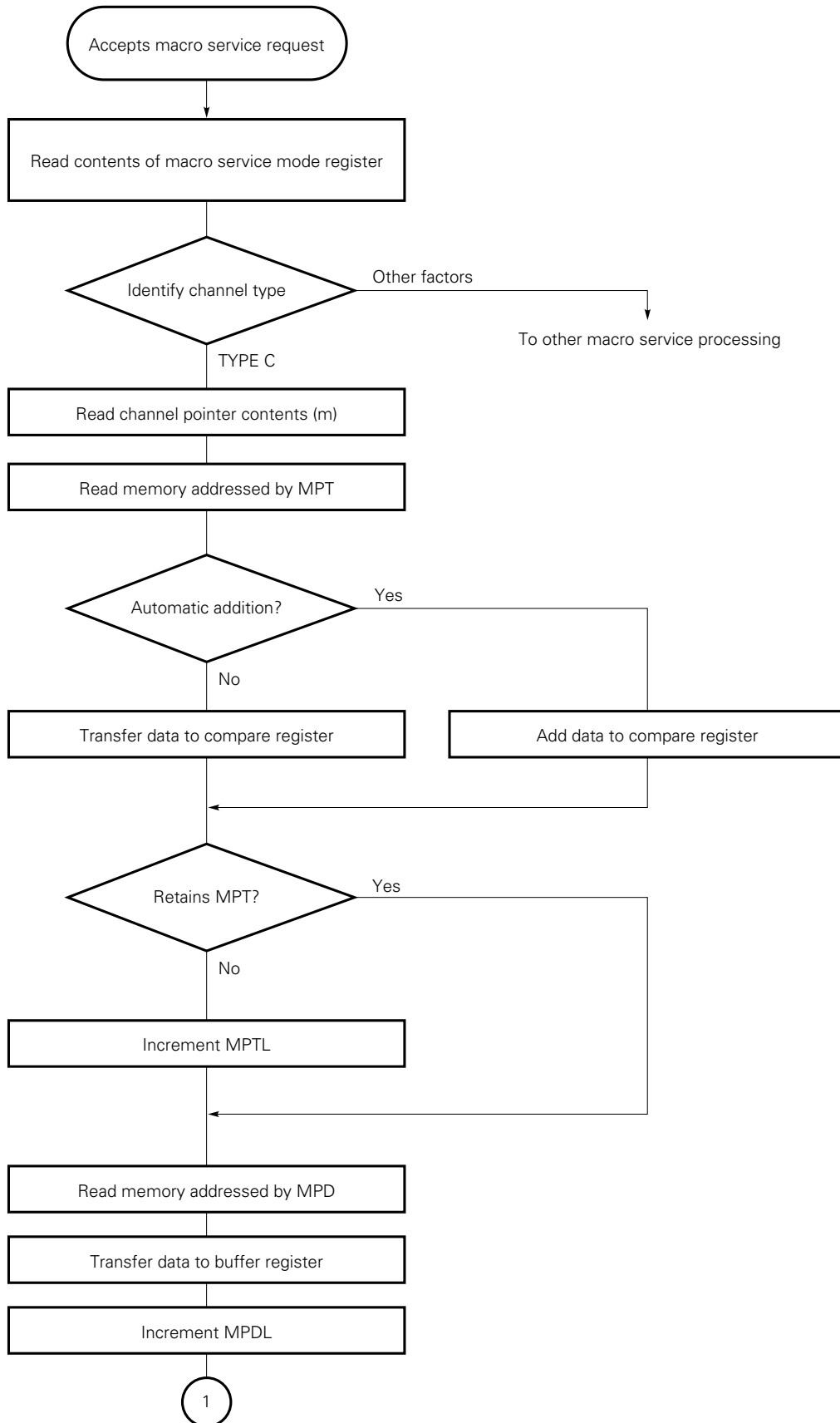
Illegal write access	
Address	Data
Address of a destination SFR (CR10 or CR11)	Lower 8 bits of the address of the MPTL

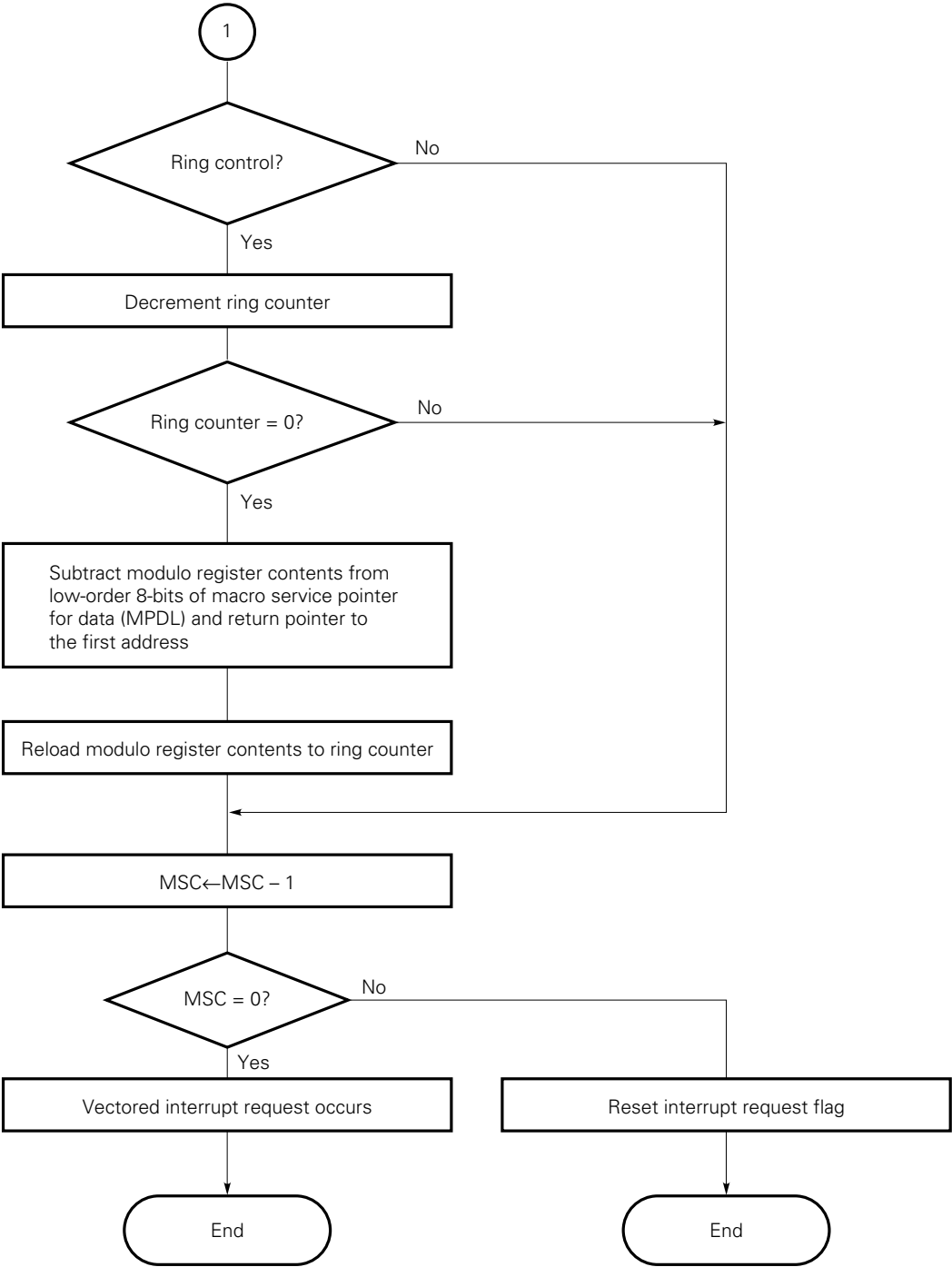
This problem may be solved by the following method.

- Locate the MPTL so that its address does not coincide with an address from 0FED0H through 0FEDFH.

The above problem also occurs with an in-circuit emulator.

Fig. 12-25 Flow of Data Transfer by Macro Service (Type C)





(2) Macro service channel configuration

There are two types of type C macro service channels, as shown in Fig. 12-26.

The timer macro service pointer (MPT) indicates a data buffer area in the 64K memory space from which data is transferred to, or added to the contents of, the compare register for 8-bit timer/counter 1.

The data macro service pointer (MPD) indicates a data buffer area in the 64K memory space from which data is transferred to the real-time output port.

The modulo register (MR) specifies the number of repetition patterns for ring control (if used).

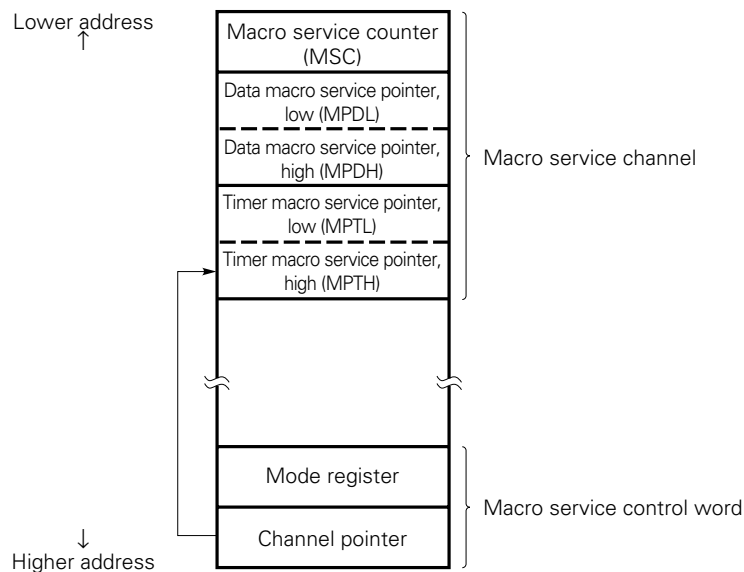
The ring counter (RC) retains the steps in a pattern for ring control (if used). Usually, it is initialized with the same value as for the modulo register.

The macro service counter (MSC) specifies the number of data transfers to occur.

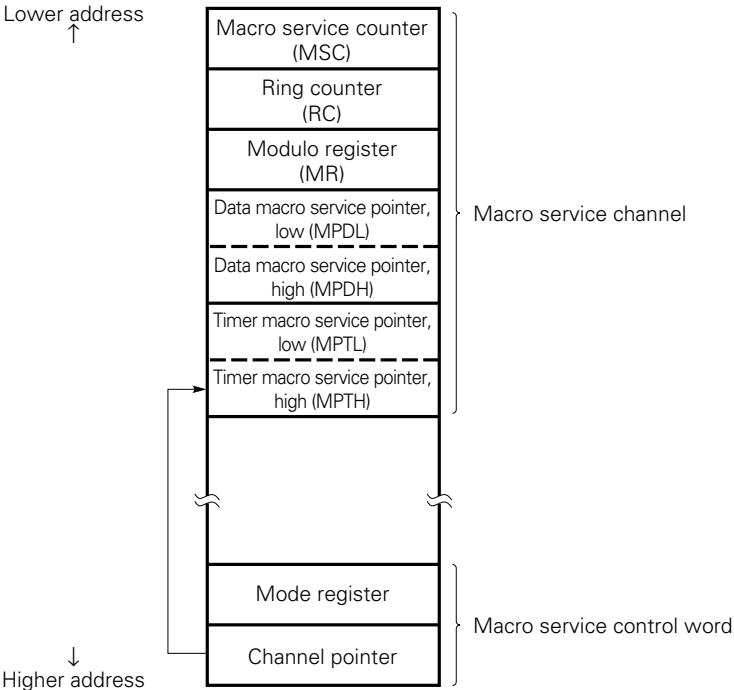
The macro service channel, holding these pointers and counter, is located at addresses 0FE00H through 0FEFFH in the internal RAM space. The macro service channel is indicated by the channel pointer as shown in Fig. 12-26. The channel pointer is set with the lower 8 bits of the address of the macro service channel.

Fig. 12-26 Type C Macro Service Channel

(a) Without ring control



(b) With ring control



(3) Example of using the type C macro service

The following example shows a pattern output to the real-time output port and how the output interval is controlled directly.

Update data is transferred from two data areas previously set in the 64K-byte space to the buffer registers (P0H and P0L) and compare registers (CR10 and CR11) for the real-time output port.

Fig. 12-27 Open-Loop Control for a Stepper Motor by the Real-Time Output Port

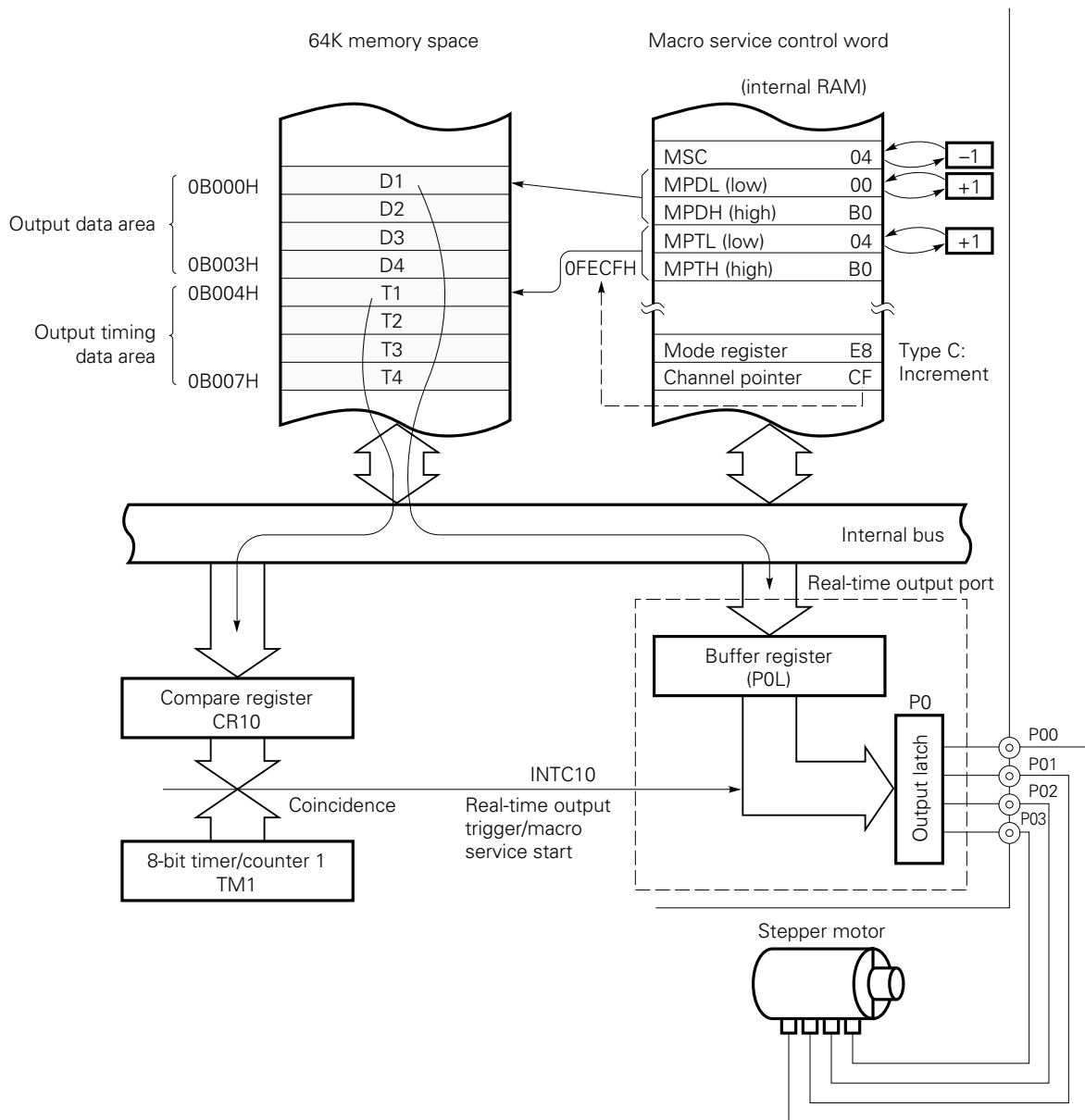
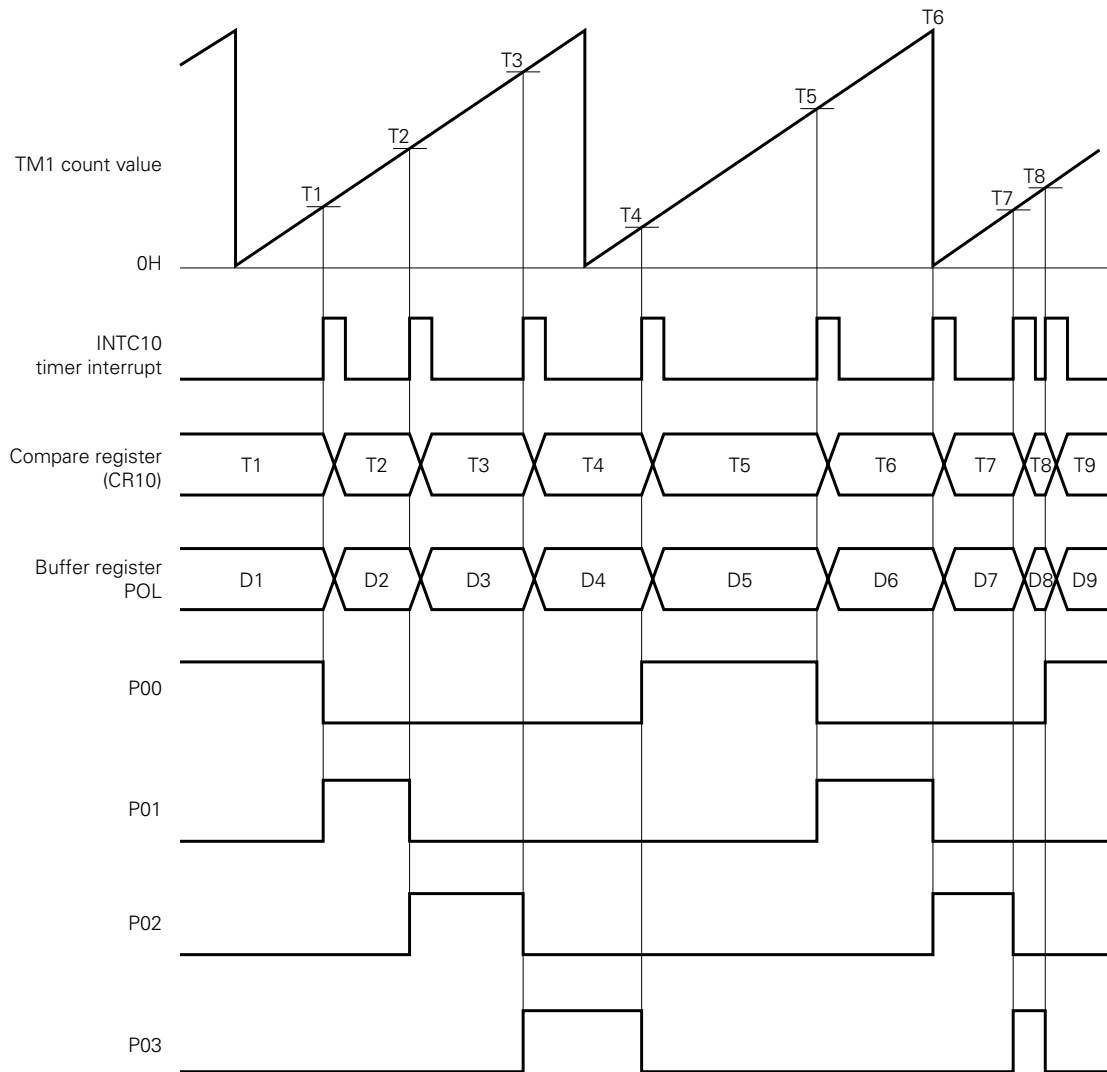


Fig. 12-28 Data Transfer Control Timing



(4) Example of using automatic addition control and ring control

(a) Automatic addition control

The automatic addition control function adds the output timing data (Δt) specified by a macro service pointer (MPT) to the contents of a compare register and writes the sum to the compare register.

Using this function makes it unnecessary for software to calculate the setting for the compare register each time the compare register has to be set.

(b) Ring control

The ring control function cyclically outputs a pattern of predetermined output data.

When this function is used, only one cycle of output data pattern must be prepared even if more data patterns are required. Therefore, the required data ROM area can be kept small.

The macro service counter (MSC) is decremented each time one data transfer is performed.

The ring control function also generates an interrupt request when $MSC = 0$.

Let's take an example of controlling a stepping motor. The data pattern output to the stepping motor varies with the structure and excitation method (single-phase or double-phase excitation) of the motor. In any case, the output pattern is repeated. Fig. 12-29 and 12-30 show examples of controlling a 4-phase stepping motor with the phase 1 excitation method and the phases 1 and 2 excitation method, respectively.

Fig. 12-29 Four-Phase Stepping Motor with Phase 1 Excitation

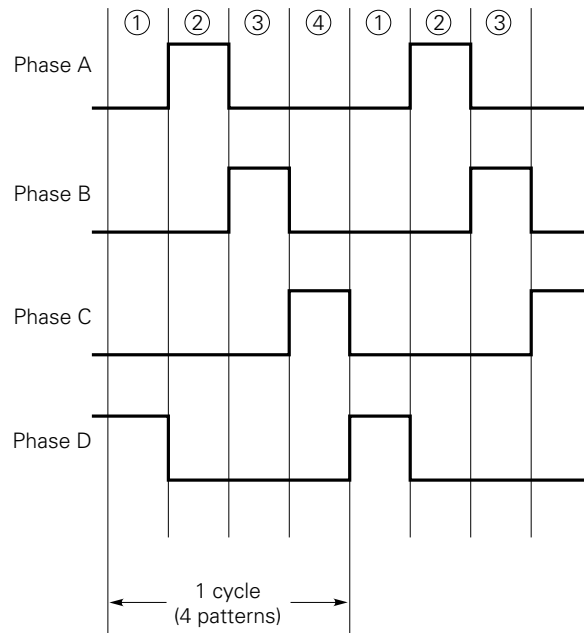


Fig. 12-30 Four-Phase Stepping Motor with Phases 1 and 2 Excitation

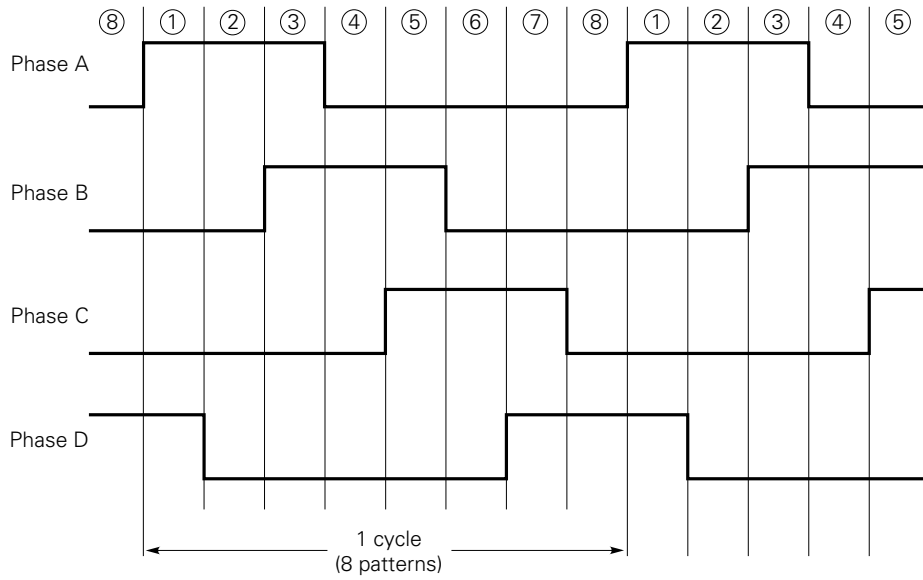


Fig. 12-31 Block Diagram 1 for Automatic Addition Control Plus Ring Control (Constant-Speed Rotation with Phases 1 and 2 Excitation)

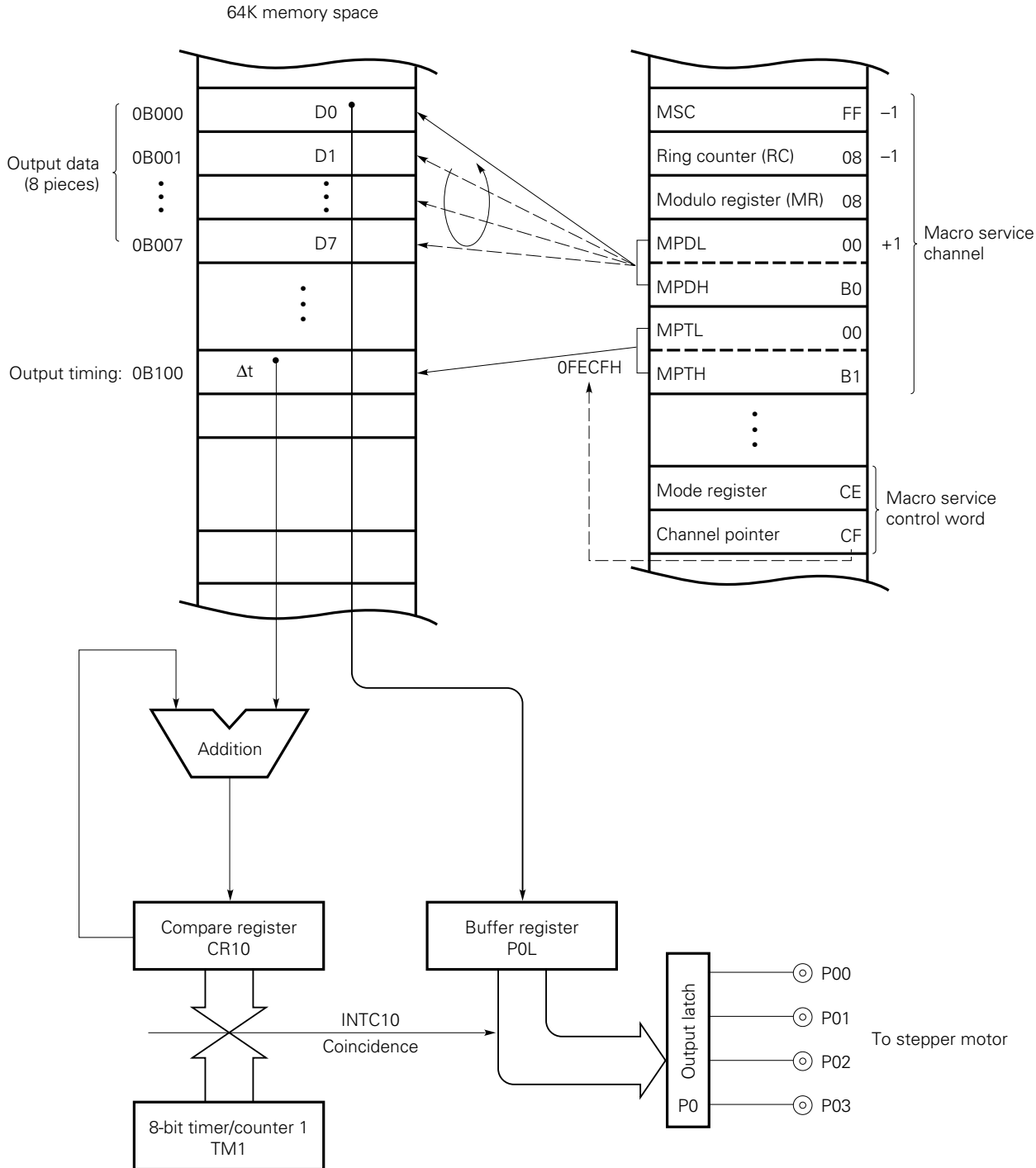
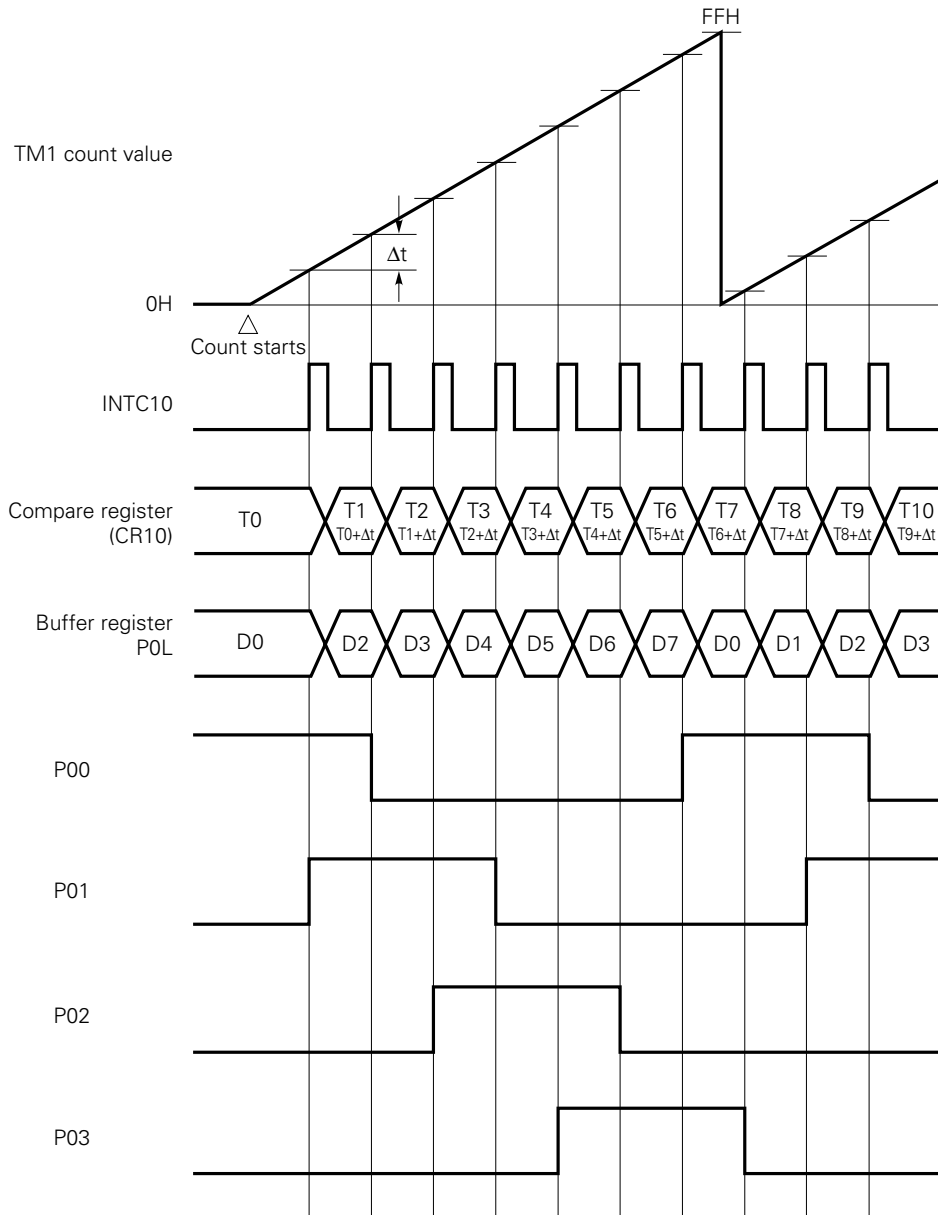


Fig. 12-32 Timing Chart 1 for Automatic Addition Control Plus Ring Control (Constant-Speed Rotation with Phases 1 and 2 Excitation)



Remark Select a mode in which the MPT contents are retained.

Fig. 12-33 Block Diagram 2 for Automatic Addition Control Plus Ring Control (with the Output Timing Varied by Phase 2 Excitation)

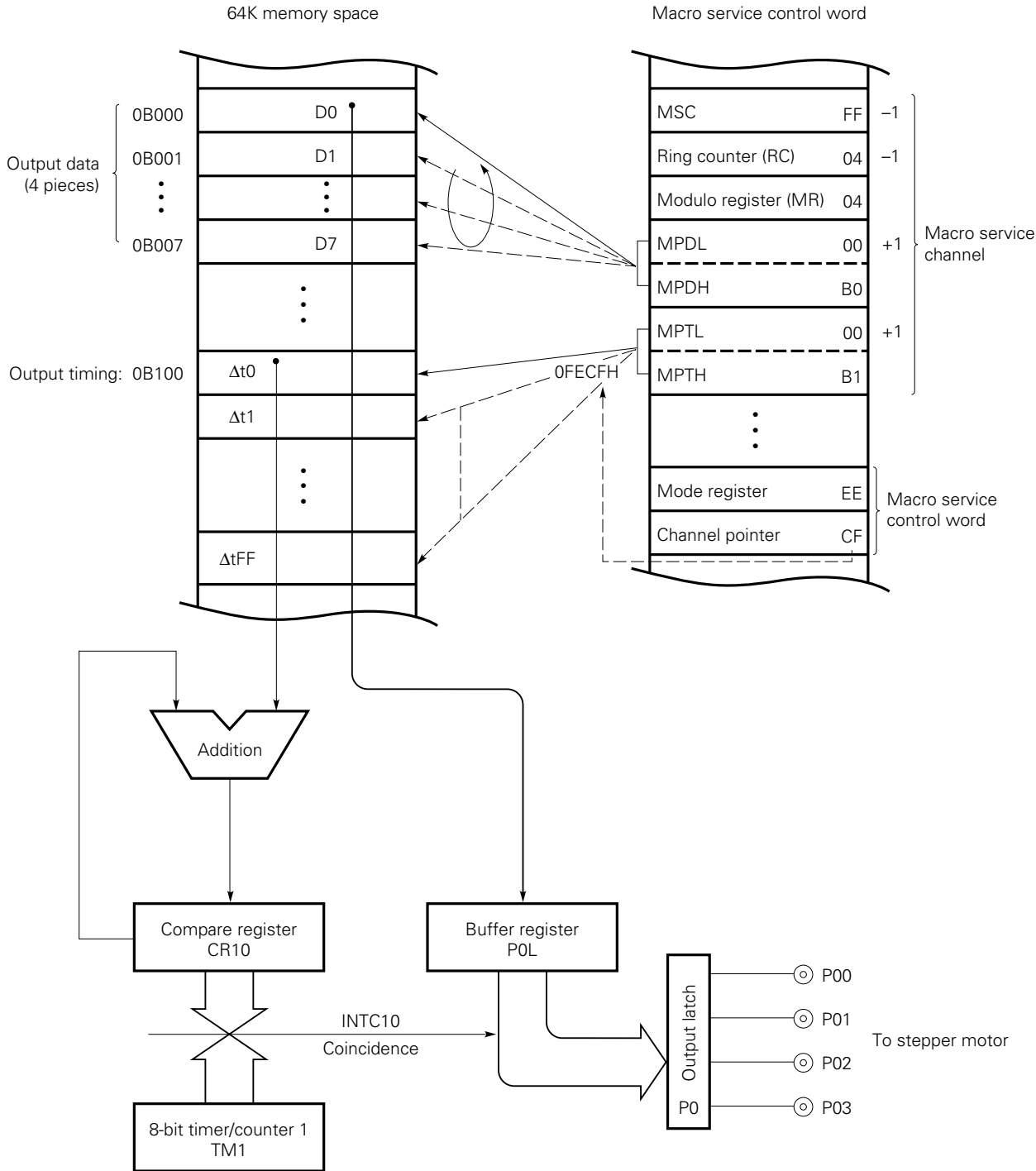
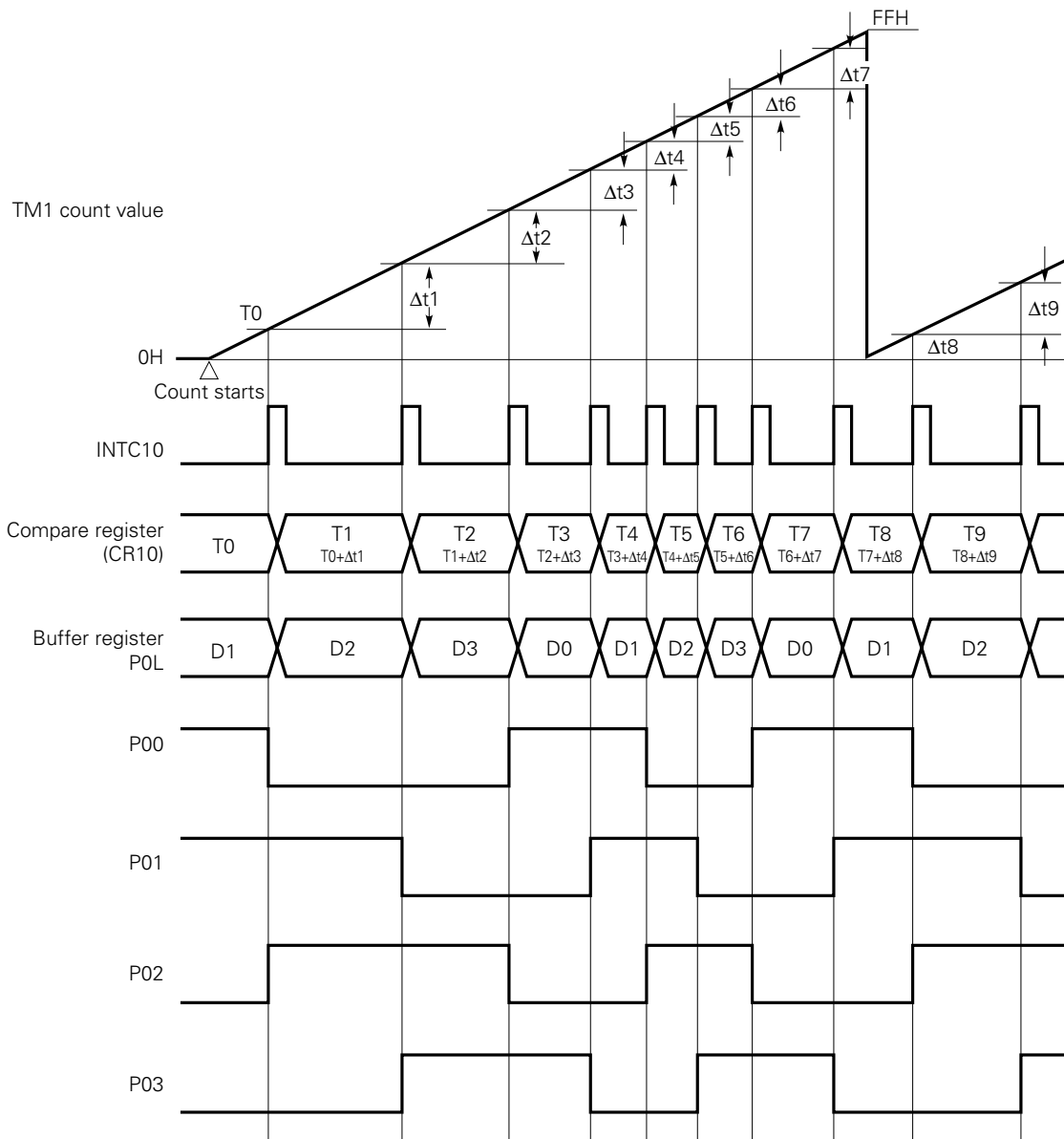


Fig. 12-34 Timing Chart 2 for Automatic Addition Control Plus Ring Control (with the Output Timing Varied by Phase 2 Excitation)



Remark Select a mode which increments the MPT.

12.5 NOTES

- (1) Do not use the RETI instruction to return from the software interrupt.
- (2) A macro service request is accepted and processed even when a nonmaskable interrupt service program is running. To disable macro service processing during execution of the nonmaskable interrupt service program, cause the nonmaskable interrupt service program to manipulate the mask register so that no macro service will not occur.
- (3) If the IE bit of the PSW is set to 1, for example, by executing the EI instruction in the nonmaskable interrupt service program, maskable interrupt requests assigned high priority are made acceptable. If a maskable interrupt with high priority occurs during execution of the nonmaskable interrupt service program, a service program for the maskable interrupt runs. If the IE and ISP bits of the PSW are set to 1, interrupt requests with low priority will also occur, thus causing the interrupt service program to run. An RETI instruction will be used to return from the maskable interrupt service program. The RETI instruction resets the NMIS bit to 0, thus enabling nonmaskable interrupt requests even when multiple-interrupt handling should not be performed for nonmaskable interrupts during execution of the nonmaskable interrupt service program to which a return was just made. To inhibit multiple-interrupt handling for nonmaskable interrupt requests, do not enable interrupts during execution of the nonmaskable interrupt service program.
- (4) Nonmaskable interrupts are always accepted except during execution of the nonmaskable interrupt handling program (except when multiple-interrupt handling for nonmaskable interrupts have been enabled by resetting the NMIS bit of the IST register to 0 during execution of the nonmaskable interrupt handling program) and except a period between a special instruction described in **Section 12.3.5** and an instruction that follows that special instruction. Therefore, nonmaskable interrupts are accepted, even if the contents of the stack pointer are undefined, for example, right after a reset occurs. At this point, the contents of the PC and PSW may be transferred to addresses (see **Table 3-4 in Section 3.2.5**) where writing to any special-function register is inhibited, depending on the value in the stack pointer. If this occurs, the CPU may hang, unexpected signals may be output from pins, or an attempt may be made to transfer the contents of the PC or PSW to a location where no RAM has been installed, thereby making it impossible to return from the nonmaskable interrupt handling program to the main routine, hence a program crash.

If a falling edge (valid edge of the NMI input after a reset) arrives at the NMI pin at much the same time when a rising edge is supplied to the RESET pin, a branch occurs to the nonmaskable interrupt handling program without executing a single instruction after a reset, resulting in a program crash almost with no exception. To avoid these problems, initialize the stack pointer after a reset, and design the hardware so that the NMI signal does not drop within $10 \mu\text{s} + 20/f_{\text{CLK}}$ after the RESET signal rises. ★

- (5) When a BF instruction is used to poll registers related to interrupts, do not specify this BF instruction as the branch destination. Otherwise, all interrupts and macro services are kept pending until a condition that inhibits a branch is met during execution of the instruction.

Example of incorrect coding

```

      :
      :
LOOP: BF IF0H.3, $LOOP ← All interrupts and macro services are kept pending, until
      :               IF0H.3 is set to 1. The pending state continues until the
      :               instruction next to the BF is executed.
      :               ×××
      :

```

Example of correct coding (1)

```

      :
      :
LOOP: NOP ← Interrupts or macro services will not be kept pending long,
      :               because they are processed after the NOP is executed.
      :               BF IF0H.3, $LOOP
      :

```

Example of correct coding (2)

```

:
LOOP:  BT IF0H.3, $NEXT
      BR $LOOP

NEXT:
:

```

Remark The BTCLR would be more convenient than the BT, because it clears the flags automatically.

← Interrupts or macro services will not be kept pending long, because they are processed after the BR is executed.

- (6) In addition, when you have to use a coding of the instructions listed in **Section 12.3.5** consecutively, yet expect frequent occurrence of interrupts and macro services, insert NOP instructions in the coding to allow time during which interrupts and macro service are accepted.
- (7) With the type C macro service, the MPT and MPD are incremented only at the lower bits. If a carry occurs at bit 7 of the MPTL or MPDL, it is ignored; so the higher 8 bits are not affected.
- (8) When the external memory is expanded (or always with the μPD78213), an illegal write access operation may occur during the type A macro service. This illegal write access occurs when any of the following three conditions is satisfied.
 1. For the type A macro service, when data D0H through DFH is transferred from memory to an SFR.
 2. For the type A macro service, when macro service transfers data from an SFR to a buffer (memory) at 0FED0H through 0FEDFH.
 3. For the type C macro service, when the MPTL address is 0FED0H through 0FEDFH.

An illegal write access is processed in the same manner as the normal memory access. In addition, wait states may be inserted according to the setting of the PW20 and PW21 bits of the memory expansion mode register (MM). Table 12-9 lists the conditions under which an illegal write access occurs and the corresponding operations.

Table 12-12 Illegal Write Access Conditions and Corresponding Operations

Condition	Macro service type	Illegal write access	
		Address	Data
1	A	Address of a destination SFR	Data transferred by macro service
2	A	Address of a source SFR	Lower 8 bits of the address of the destination buffer (memory)
3	C	Address of a destination SFR (CR10 or CR11)	Lower 8 bits of the address of the MPTL

This problem may be solved by the following methods.

1. It is difficult for software to solve the problem if it occurs under condition 1, because it depends on the transfer data. Therefore, use an external address decoder circuit to keep the image in the area of 0FF00H through 0FFFFH from overlapping the memory addresses of the external circuit.
2. If the macro service to be used does not satisfy condition 1 (i.e., if data is not transferred from an SFR to memory), and under condition 2, locate the buffer area so that its addresses are not 0FED0H through 0FEDFH.

The above problem also occurs with an in-circuit emulator.

- ★ (9) For the type B service macro, the following registers cannot be used as SFRs.
IF0L, IF0H, MK0L, MK0H, PR0L, PR0H, ISM0L, ISM0H, and IST

CHAPTER 13 LOCAL BUS INTERFACE FUNCTION

The local bus interface function is provided to connect external memories (ROM and RAM) and I/Os.

External memories (ROM and RAM) and I/Os are accessed by using the \overline{RD} , \overline{WR} , and ASTB signals, a multiplexed address/data bus consisting of lines AD0 to AD7, and an address bus consisting of lines A8 to A19. Figs. 13-3 and 13-4 show the basic bus interface timing diagrams.

In addition, a wait function for interfacing with low-speed memory, and a refresh signal output function for refreshing pseudo static RAM, are provided.

13.1 CONTROL REGISTERS

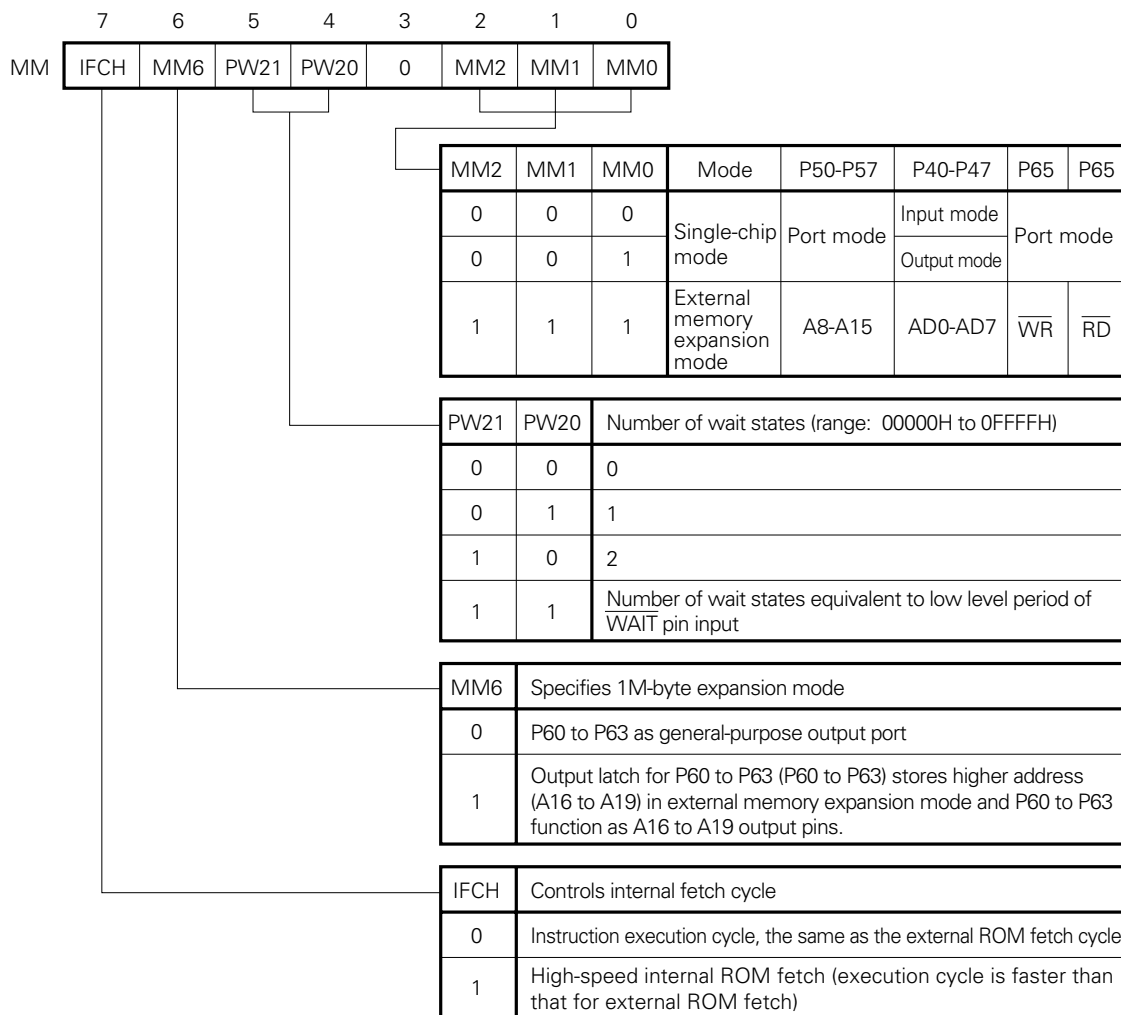
13.1.1 Memory Expansion Mode Register (MM)

The MM register is an 8-bit register for controlling externally expanded memory, specifying the number of wait states (address space: 00000H to 0FFFFH), and controlling the internal fetch cycle.

The MM register can be read and written with 8-bit manipulation instructions and bit manipulation instructions. Fig. 13-1 shows the format of the MM register.

When the $\overline{\text{RESET}}$ signal is applied, the register is set to 20H.

Fig. 13-1 Format of the Memory Expansion Mode Register (MM)

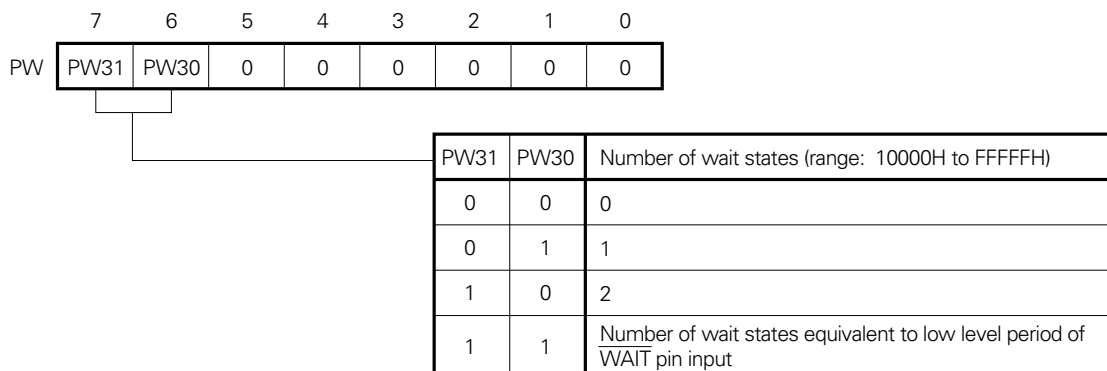


13.1.2 Programmable Wait Control Register (PW)

The PW register is an 8-bit register for specifying the number of wait states for external expansion data memory space 10000H to FFFFFH. The PW register can be read and written with both 8-bit manipulation instructions and bit manipulation instructions. Fig. 13-2 shows the format of the PW register.

When the $\overline{\text{RESET}}$ signal is applied, the register is set to 80H.

Fig. 13-2 Format of Programmable Wait Control Register (PW)



13.2 MEMORY EXPANSION FUNCTION

13.2.1 External Memory Expansion Function

The $\mu\text{PD78214}$ allows additional 48384-byte memory (for the $\mu\text{PD78212}$, 56704-byte memory) and I/Os to be connected according to the setting of the memory expansion mode register (MM).

When the memory space is expanded externally, P50 to P57 function as an address bus, and P40 to P47 function as a multiplexed address/data bus.

When the $\overline{\text{EA}}$ signal is tied low, the $\mu\text{PD78214}$ operates in ROM-less mode. In ROM-less mode, 64768 bytes (65536 bytes for the $\mu\text{PD78212}$) of memory or I/Os can be connected.

The $\mu\text{PD78213}$ is a ROM-less device, hence always uses external memory.

Caution Address information output on P50/A8 to P57/A15 and P60/A16 to P63/A19 is valid from the time the ASTB signal goes high until the $\overline{\text{RD}}$ or $\overline{\text{WR}}$ signal goes high. Other than during this period, the output levels of P50/A8 to P57/A15 and P60/A16 to P63/A19 are undefined. When designing circuits, take this into consideration, and make sure that the output of an undefined value will not cause any problems. The data sheet of the relevant product gives the specification of the valid period for address output.

Fig. 13-3 Read Timing

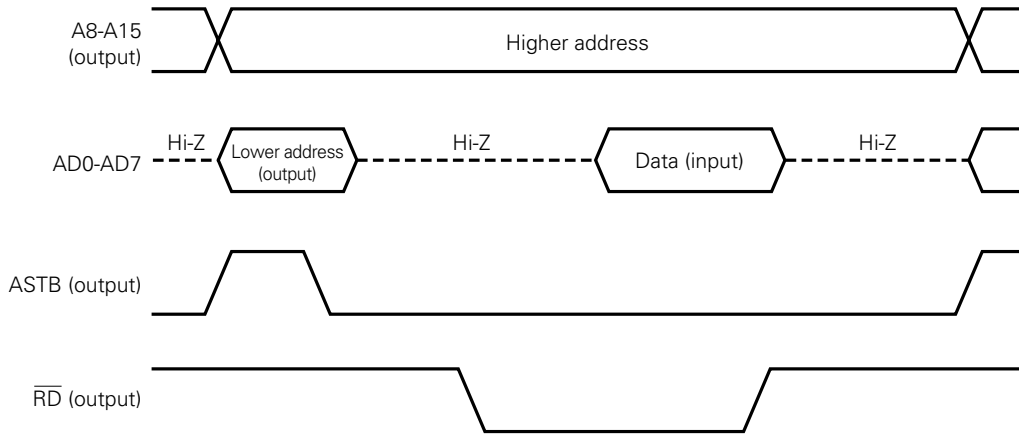
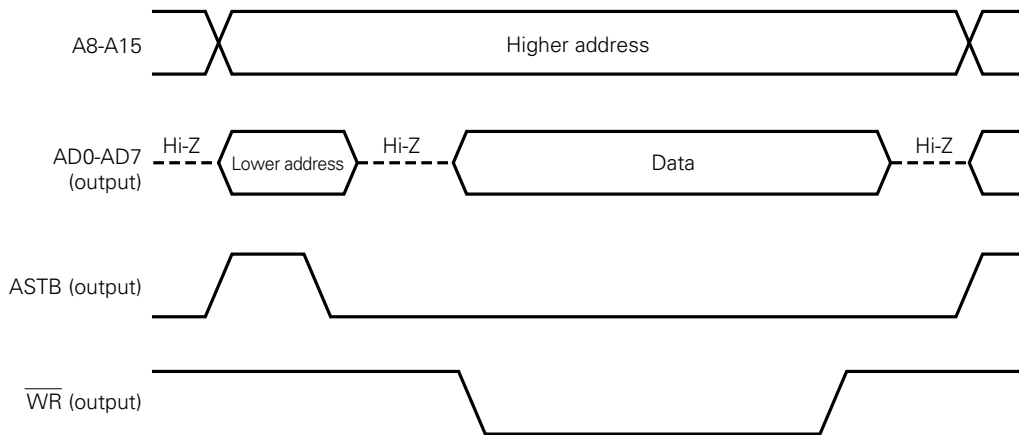


Fig. 13-4 Write Timing



Caution External devices cannot be mapped to the same addresses as those of the internal RAM area (μPD78213, μPD78214, and μPD78P214: 0FD00H to 0FEFFH, μPD78212: 0FD80H to 0FEFFH) and SFR area (0FF00H to 0FFFH, excluding the external SFR area (0FFD0H to 0FFDFH)).

When manipulating a space in which external device addresses overlap internal RAM or SFR addresses, the internal RAM or SFR area is accessed automatically. In this case, the address signal is output, but the ASTB, RD, and WR signals are not output (these signals remain inactive).

13.2.2 1M-Byte Expansion Function

When bit MM6 of the MM register is set to 1, an additional 960K-bytes of data memory can be used. Therefore, a 1M-byte memory space is available. In this case, pins P60 to P63 output the highest address bits, A16 to A19.

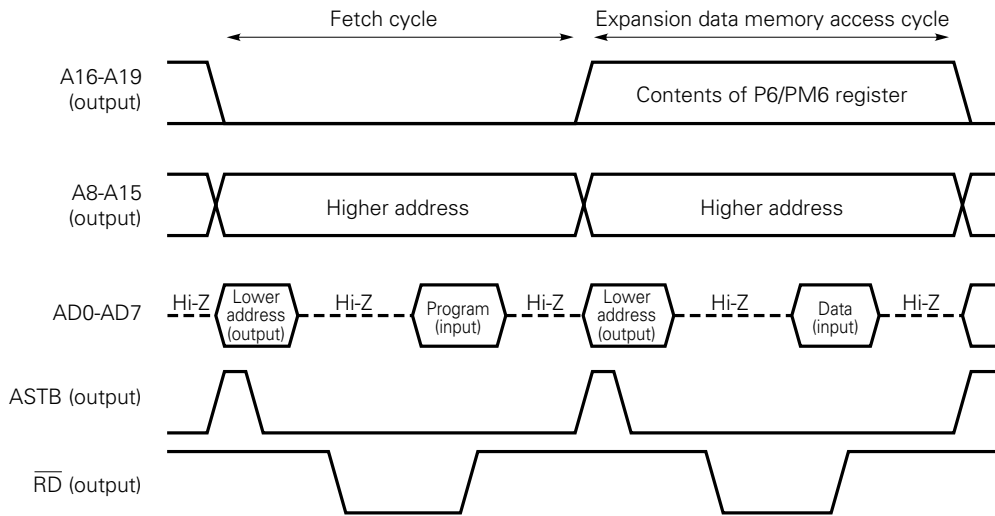
Example: MOV MM, #47H ; Expansion to 1M bytes

```

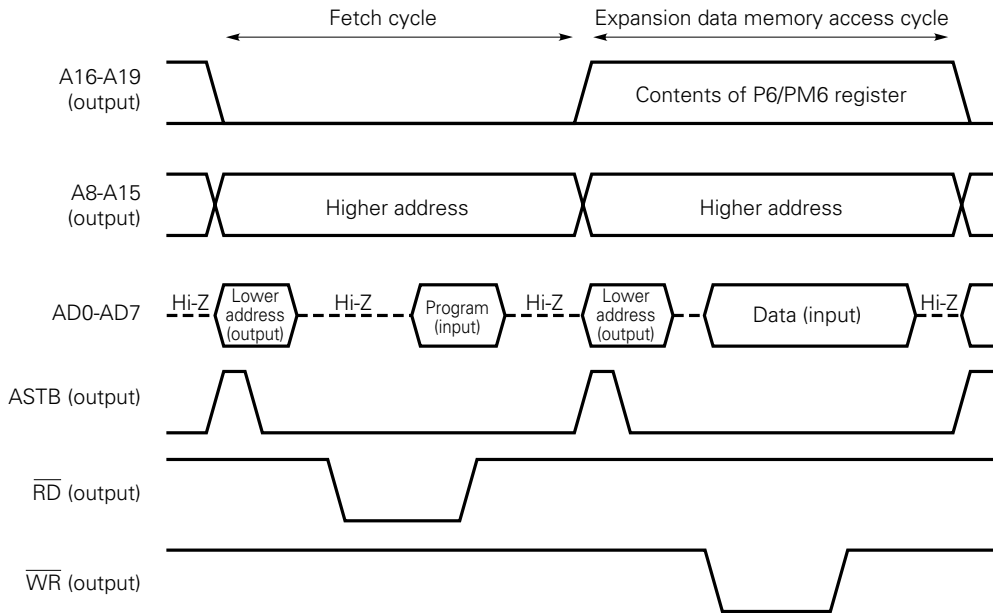
MOV P6, #3H ; Latches the highest address information (select bank 3)
⋮
MOV A, &!2000H ; Load the contents of memory at 32000H into register A.
    
```

Fig. 13-5 Accessing Expansion Data Memory

(a) Read cycle



(b) Write cycle



13.2.3 Memory Mapping with Expanded Memory

Figs. 13-6 to 13-9 show the memory maps when the memory has been expanded. Even when the memory has been expanded, external devices at the same addresses as those of the internal ROM area, internal RAM area, or SFR area (excluding the external SFR area (0FFD0H to 0FFDFH)) cannot be accessed. When these addresses are accessed, the memory and SFRs of the μPD78214 take precedence and are accessed. The ASTB, \overline{RD} , and \overline{WR} signals are not output (these signals remain inactive). The output levels of the address bus and address/data bus become undefined. However, when the memory is expanded, and the same execution cycle as the external ROM fetch cycle is specified for internal ROM fetching (by setting the IFCH bit of the memory expansion mode register (MM) to 0), the address signal and ASTB and \overline{RD} signals are output upon access to internal ROM. Information on the address/data bus at this time, however, is not read, and the CPU reads data from internal ROM. (When the \overline{RD} signal is active, the address/data bus of the μPD78214 becomes high-impedance.)

The bus cycle in this case is the same as the normal read cycle. So, the programmable wait function can insert wait states into the bus cycle.

Caution When macro service Type A or Type C is used in external memory expansion mode (the μPD78213 always uses external memory), an illegal write access may occur. This occurs when any of the following three conditions is satisfied:

- (1) Data is transferred from memory to an SFR using macro service Type A, and the transfer data is D0H to DFH.
- (2) Data is transferred from an SFR to memory using macro service Type A, and the transfer destination buffer (memory) address is 0FED0H to 0FEDFH when the macro service is executed.
- (3) The MPTL address is 0FED0H to 0FEDFH when macro service Type C is used.

An illegal write access is performed in the same way as normal memory access. In addition, wait states are inserted according to the setting of bits PW20 and PW21 of the memory expansion mode register (MM). Table 13-1 lists the conditions and operations for illegal write access.

Table 13-1 Conditions and Operations for Illegal Write Access

Condition	Macro service type	Illegal write access	
		Address	Data
1	A	Transfer destination SFR address	Data transferred by macro service
2	A	Transfer target SFR address	Low-order 8 bits of transfer destination buffer (memory) address
3	C	Transfer destination SFR (CR10 or CR11) address	Low-order 8 bits of MPTL address

This problem can be avoided by using the following methods:

1. The problem caused by condition 1 is difficult to avoid by means of software (because whether an illegal access occurs depends on the transfer data). The problem must be avoided by using the external address decoder so that the area at addresses 0FF00H to 0FFFFH does not overlap the addresses of the external circuits.
2. When the macro service being used does not satisfy condition 1 (transfer from memory to an SFR is not performed by macro service Type A), or when condition 2 is satisfied, the buffer area must be mapped to an address other than addresses 0FED0H to 0FEDFH. In the case of condition 3, the MPTL must be mapped to an address other than addresses 0FED0H to 0FEDFH.

This problem can also occur in the in-circuit emulator.

Fig. 13-6 Data Memory Expansion for μ PD78212 (When $\overline{EA} = L$)

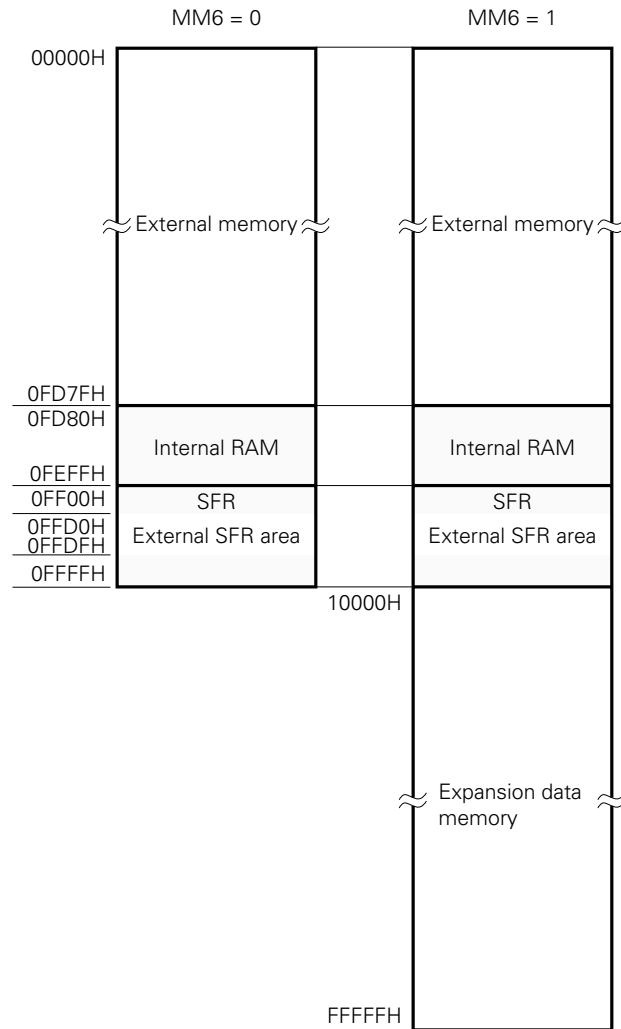


Fig. 13-7 Data Memory Expansion for μPD78212 (When $\overline{EA} = H$)

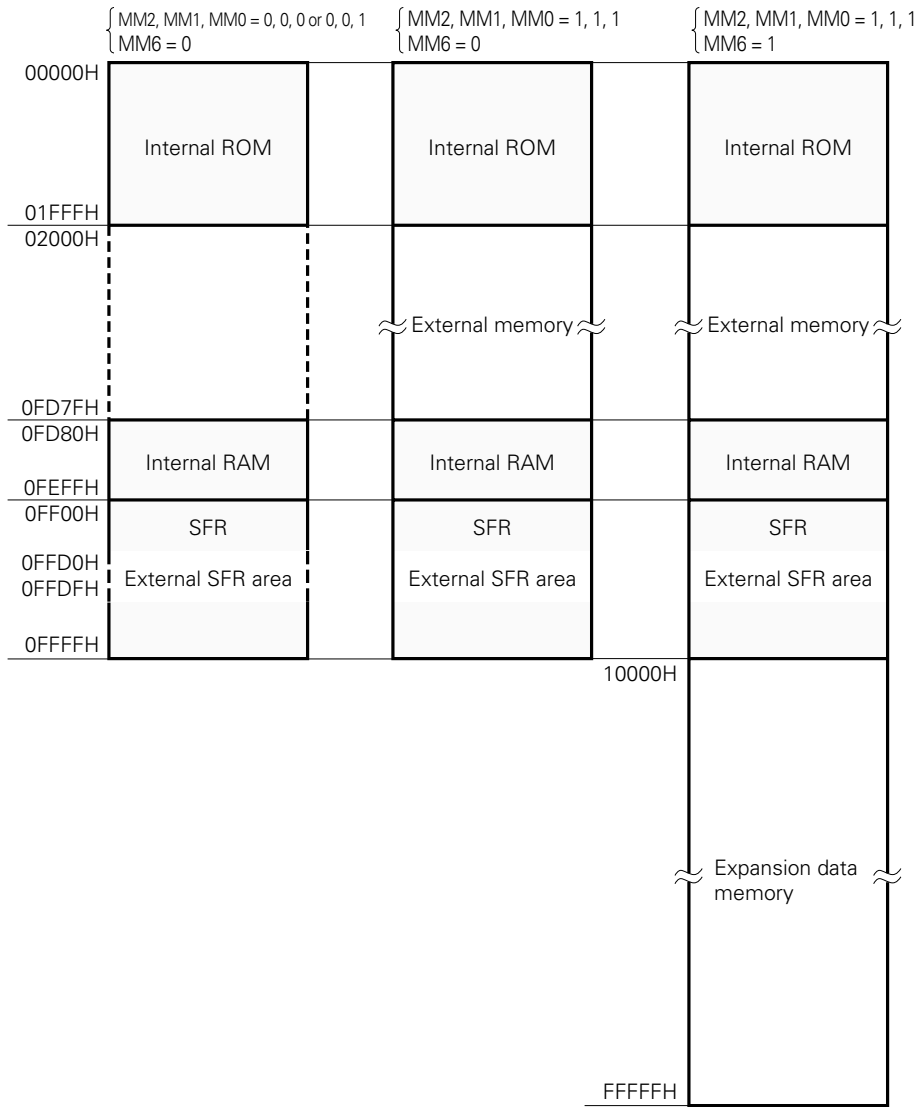


Fig. 13-8 Data Memory Expansion for μ PD78213 and μ PD78214 (When $\overline{EA} = L$)

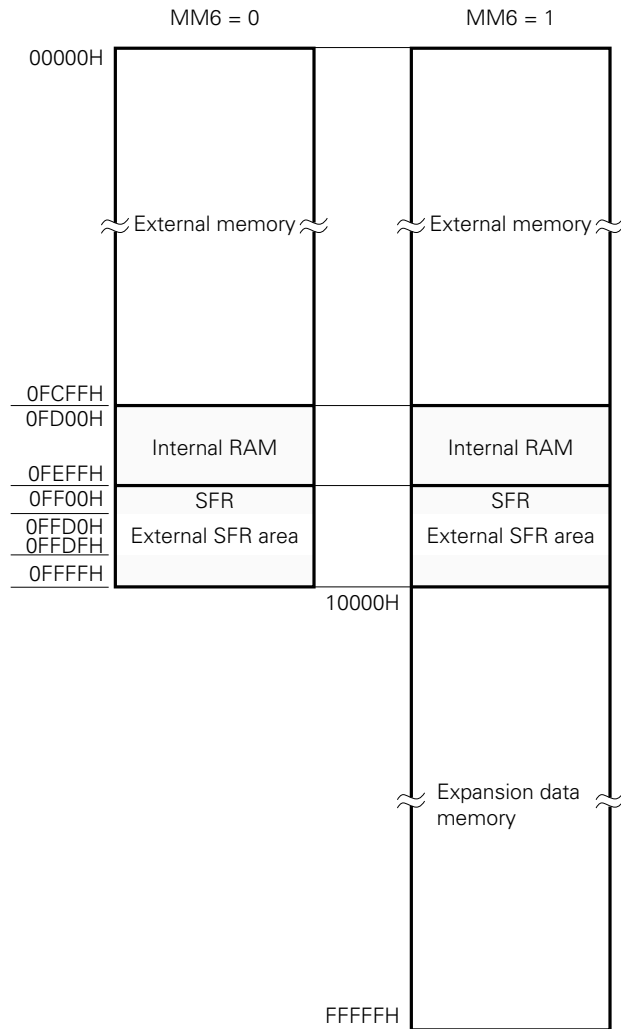
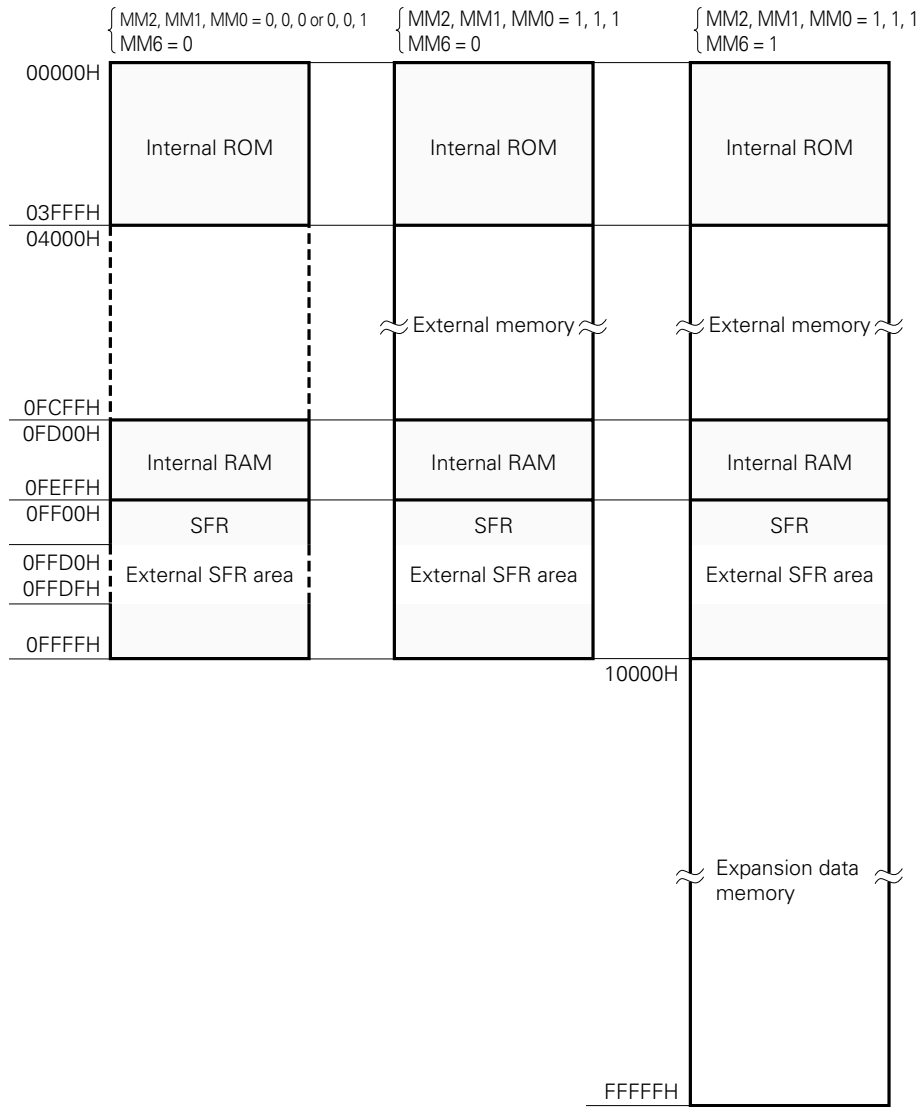


Fig. 13-9 Data Memory Expansion for μPD78214 and μPD78P214 (When EA = H)



13.2.4 Example of Connecting Memories

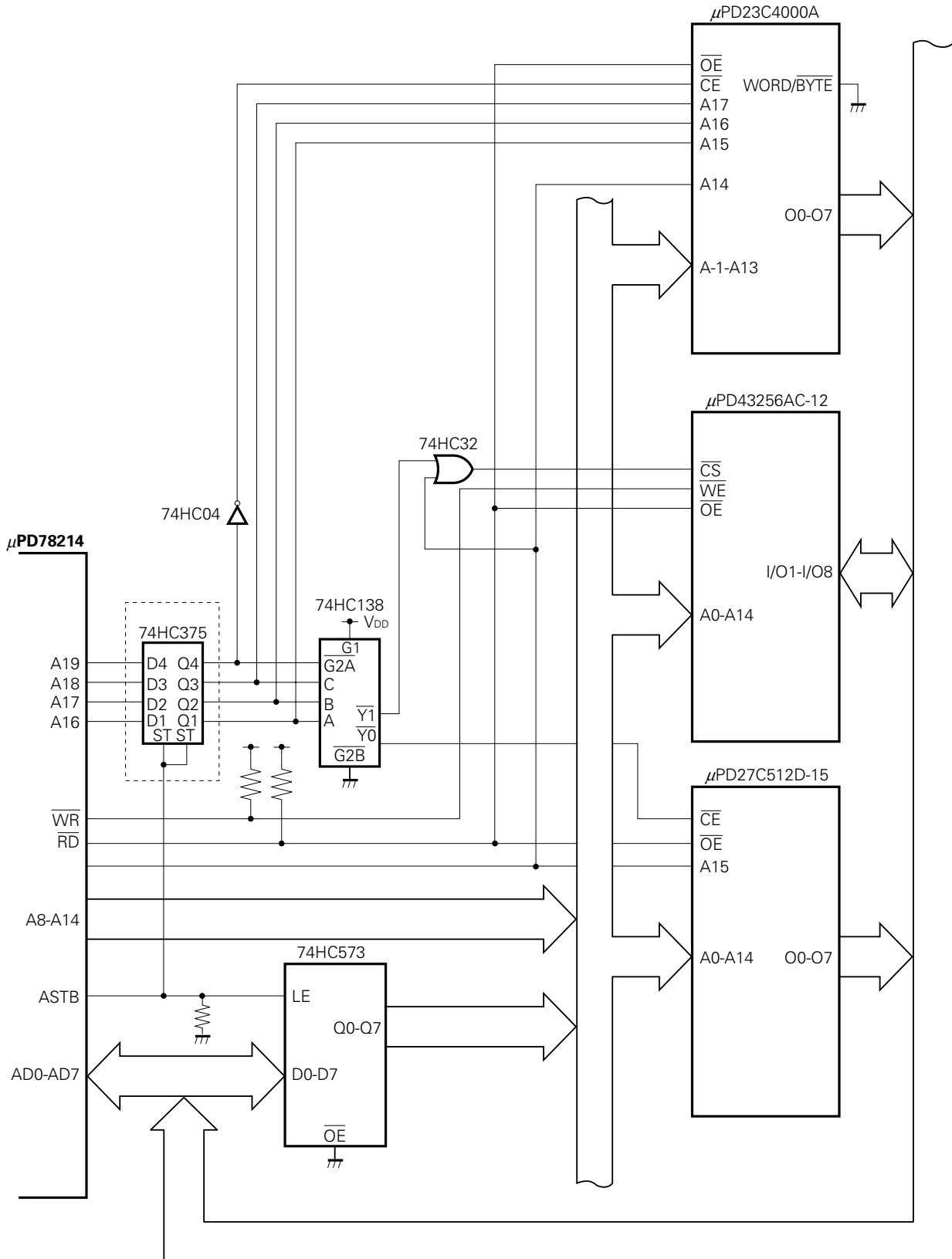
Fig. 13-10 shows an example of connecting memories to the μ PD78214. In this example, a PROM, SRAM, and mask-programmable ROM are connected to the μ PD78214. Addresses are assigned to these memories as follows:

- PROM (μ PD27C512D-15) : 0000H-FCFFH (μ PD78213)
2000H-0FD7FH (μ PD78212)
4000H-FCFFH (μ PD78214)
FFD0H-FFDFH as an external SFR area
- RAM (μ PD43256AC-12) : 10000H-17FFFH
- Mask-programmable ROM (μ PD23C4000A) : 80000H-FFFFFFH

Since the μ PD23C4000A mask-programmable ROM requires a long access time, insert one wait state by means of the programmable wait function (See **Section 13.4**).

The circuit enclosed in dotted lines (74HC375) is required only when an in-circuit emulator is used. When the emulator is not used, remove the 74HC375, and connect Dn and Qn (n = 1 to 4), respectively. If the emulator is used without this circuit, malfunctions may occur. (Usually, however, normal operation is possible because of the delays introduced by the address decoder and other circuitry.) When this circuit is not used, the 150-ns version, the μ PD43256AC-15, can be used as RAM in place of the μ PD43256AC-12.

Fig. 13-10 Example of Connecting Memories to μPD78214



Remark Pull-up resistors must be connected to the address and address/data bus lines.

13.3 INTERNAL ROM HIGH-SPEED FETCH FUNCTION

The μ PD78212, μ PD78214, and μ PD78P214 contain an internal ROM. The internal ROM can be accessed quickly without having to use the bus control circuit. Usually, internal ROM is fetched at the same speed as external ROM. When the IFCH bit of the memory expansion mode register (MM) is set to 1, the high-speed fetch function is enabled, which can speed up internal ROM fetching.

When the same instruction execution cycle as that for external ROM fetching is selected, wait states are inserted into the cycle by the wait function. When high-speed fetching is performed, however, no wait state is inserted when accessing internal ROM.

When the $\overline{\text{RESET}}$ signal is applied, the instruction execution cycle for internal fetching is the same as that for the external ROM fetch cycle.

13.4 WAIT FUNCTION

When a slow memory or I/O is connected to the μ PD78214, wait states can be inserted into the external memory access cycle.

Wait states are inserted while the $\overline{\text{RD}}$ or $\overline{\text{WR}}$ signal is low. The low level period of the signal is extended by $1/f_{\text{CLK}}$ (167 ns with $f_{\text{CLK}} = 6$ MHz) for each wait state.

There are two methods of inserting wait states: Using the programmable wait function to automatically insert a predefined number of wait states, and using an external wait signal to control wait insertion.

Wait state insertion is controlled with the memory expansion mode register (MM) for space 00000H to 0FFFFH and the programmable wait control register (PW) for space 10000H to FFFFFH. No wait state is inserted when internal ROM and RAM are accessed upon high-speed fetching. When the internal SFR area is accessed, wait states are inserted at the appropriate timing, regardless of the register specification.

When an access to internal ROM is specified such that it will be performed in the same execution cycle as an access to external ROM, wait states are also inserted into the internal ROM access cycle according to the MM register setting.

When control with an external wait signal is specified by the MM register and/or the PW register, the $\overline{\text{WAIT}}$ signal is applied to pin P66.

When the $\overline{\text{RESET}}$ signal is applied, pin P66 functions as a general-purpose I/O port pin.

Figs. 13-15 to 13-17 show the bus timings when wait states are inserted.

Caution When using the external wait signal, set bit 6 of the PM6 register to 1 to set pin P66/ $\overline{\text{WAIT}}$ to input mode.

Fig. 13-11 Wait Control Space of μPD78212 (When $\overline{EA} = L$)

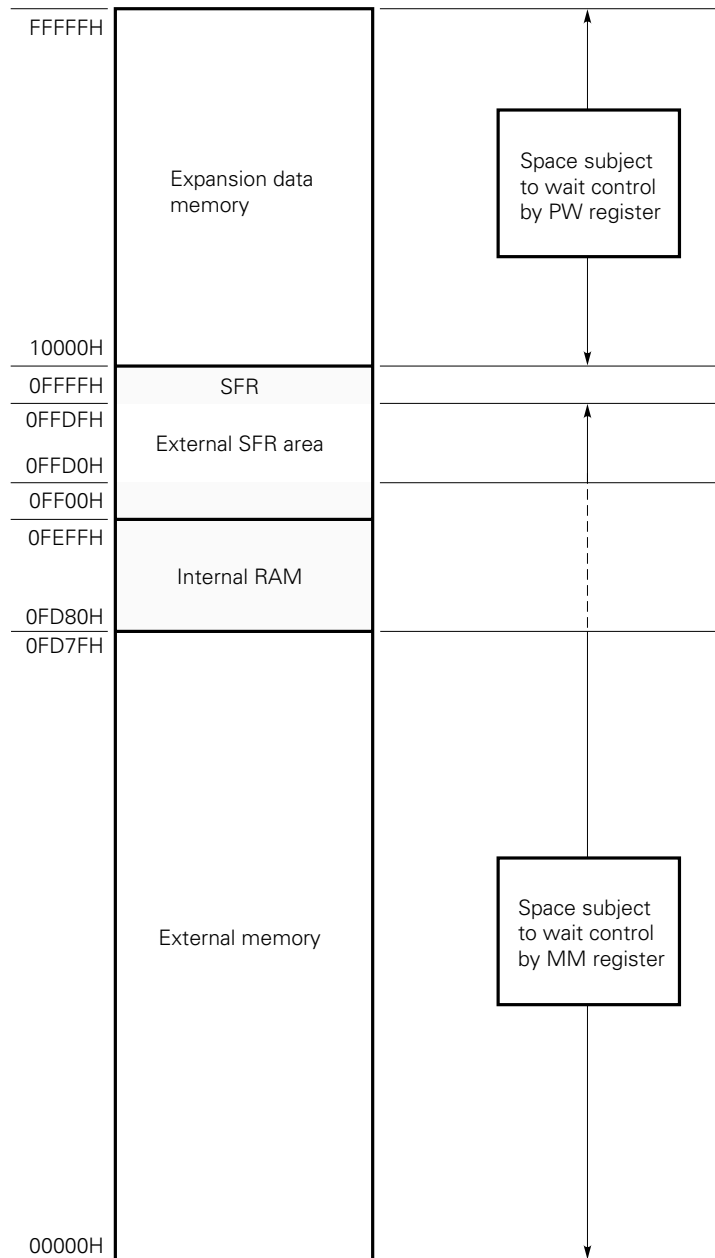


Fig. 13-12 Wait Control Space of μ PD78212 (When $\overline{EA} = H$)

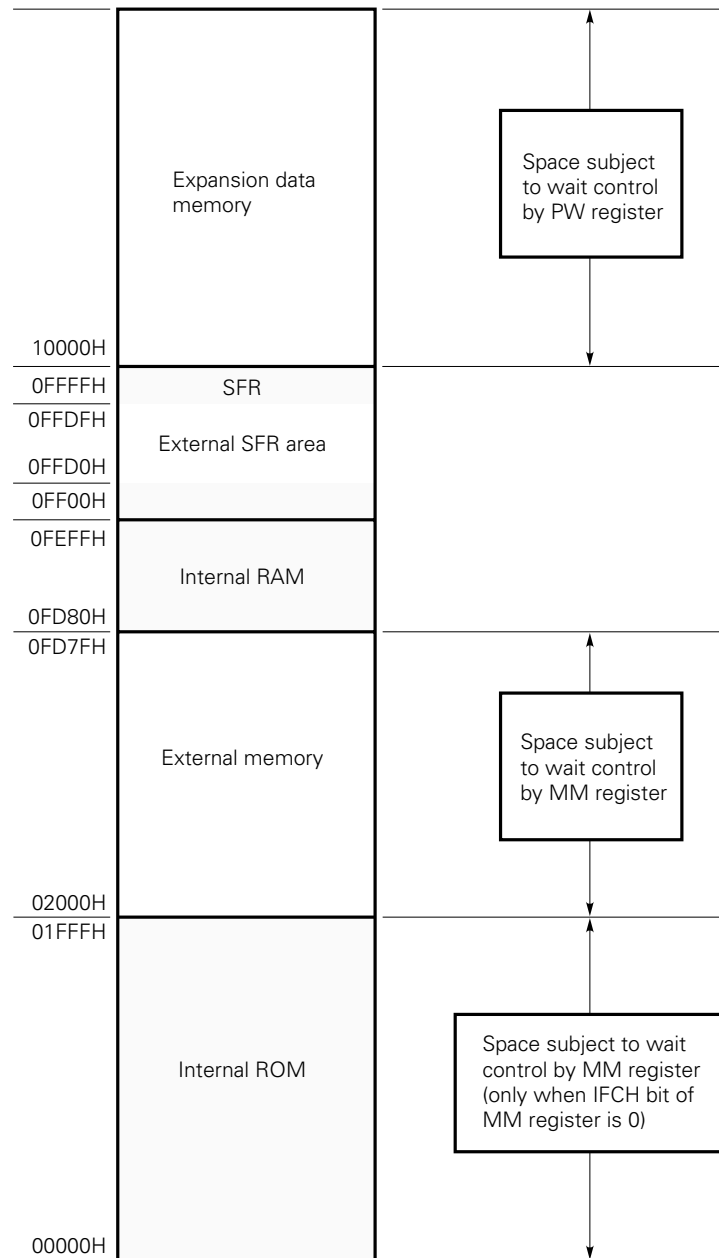


Fig. 13-13 Wait Control Space of μPD78213 and μPD78214 (When $\overline{EA} = L$)

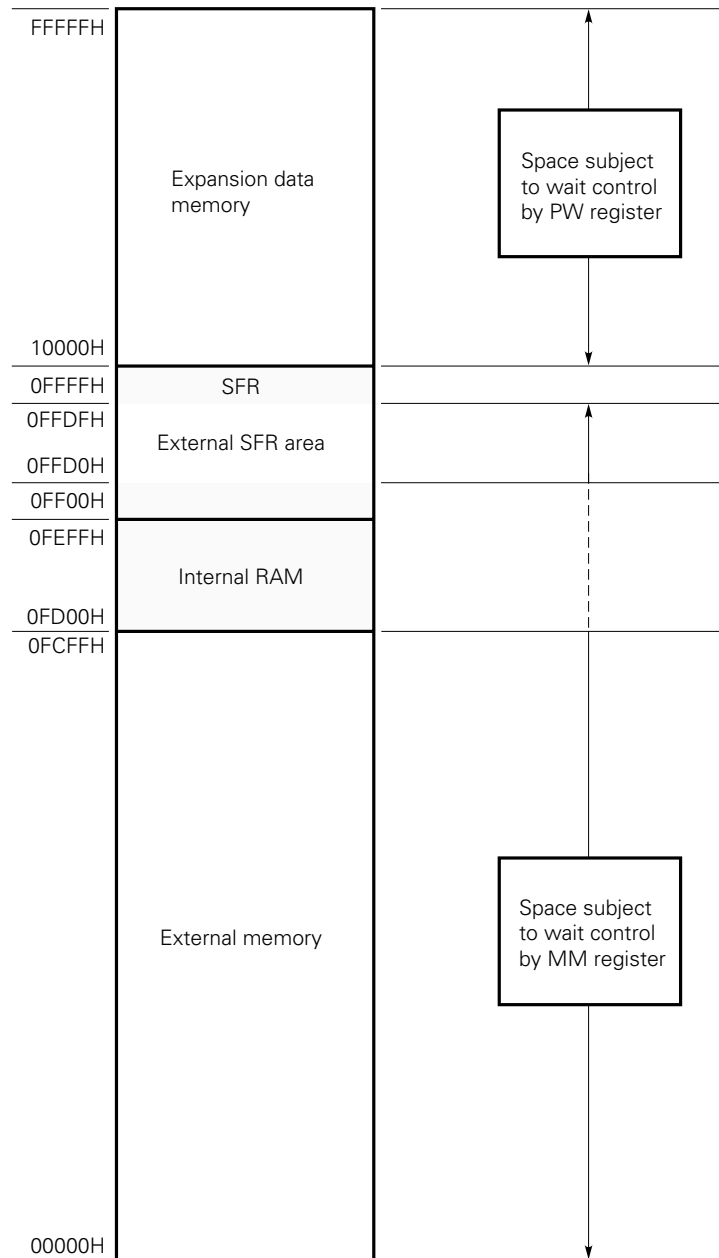


Fig. 13-14 Wait Control Space of μ PD78214 and μ PD78P214

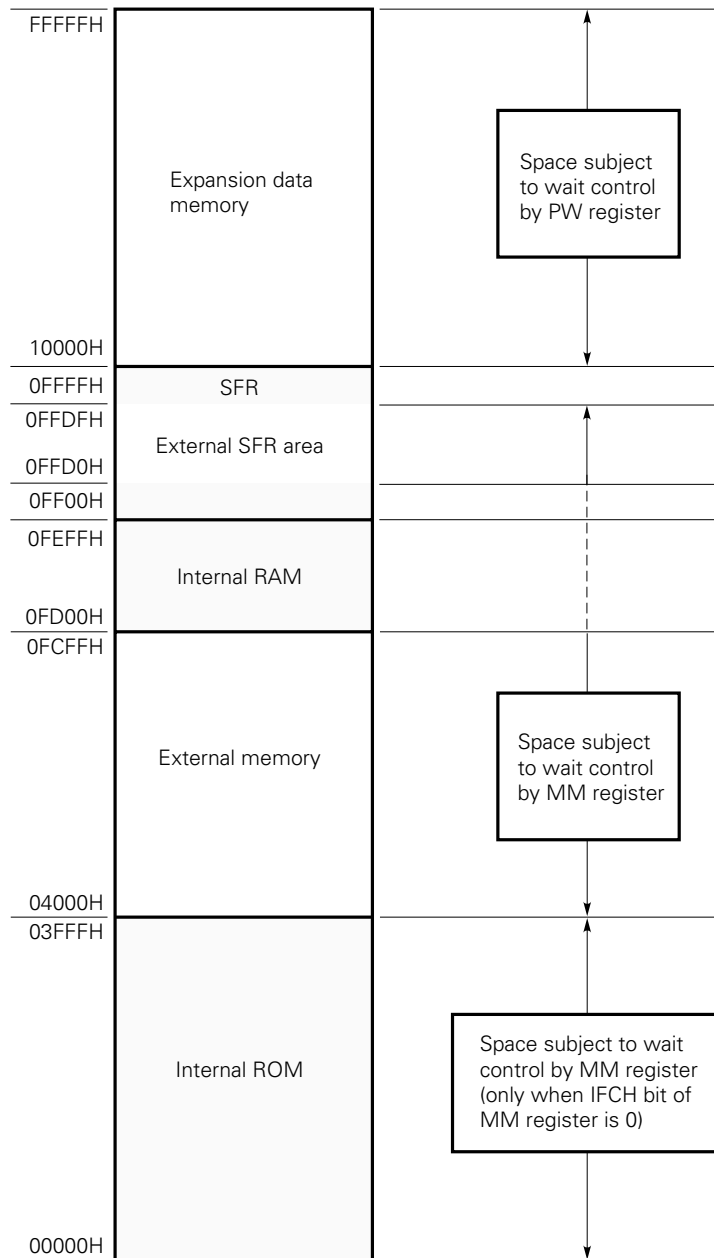
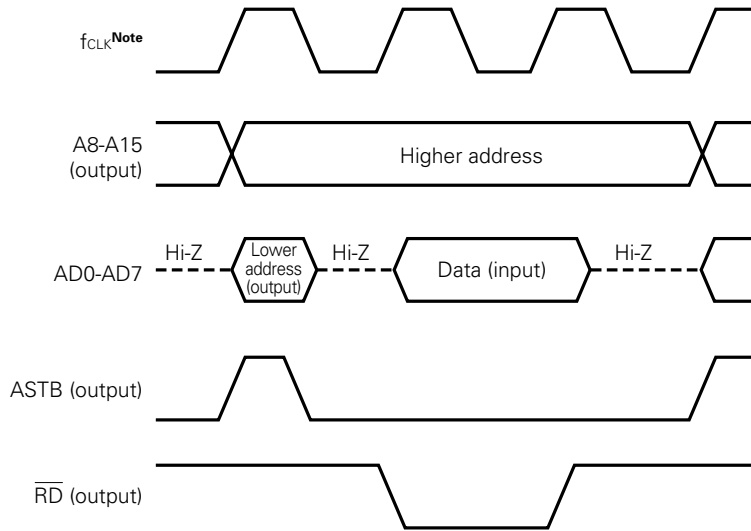


Fig. 13-15 Read Timing of Programmable Wait Function (1/2)

(a) When zero wait states are set



(b) When one wait state is set

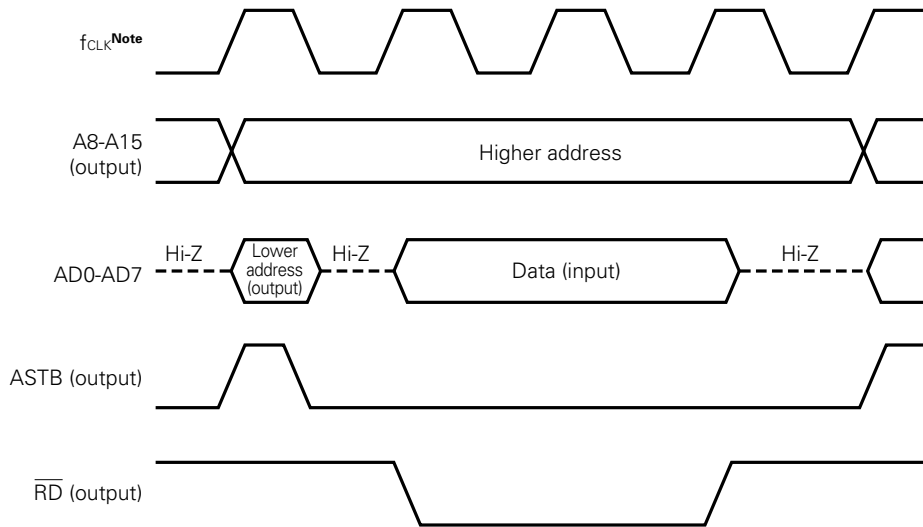
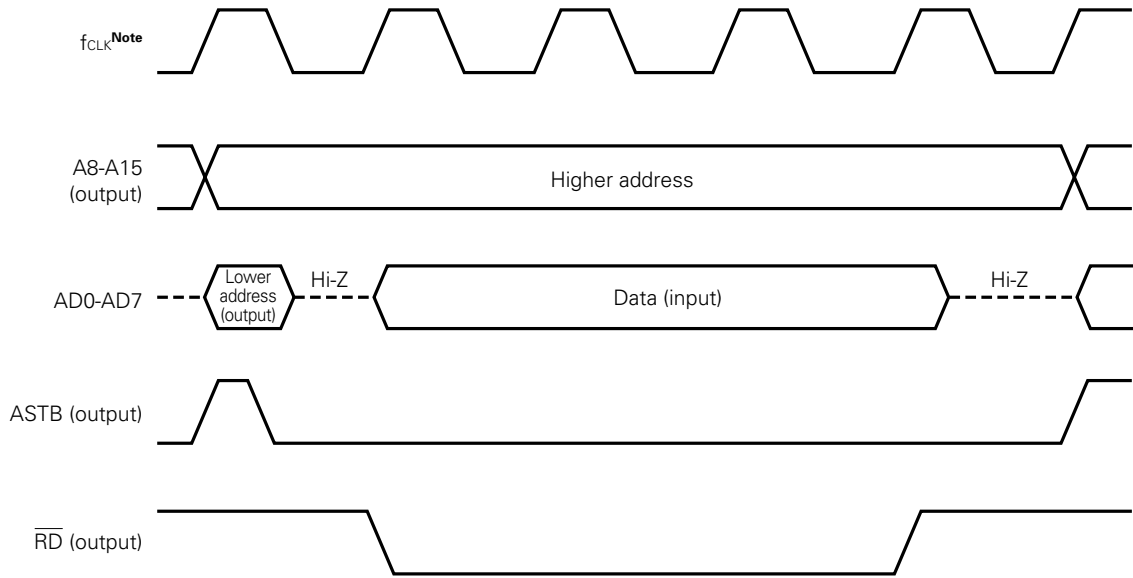


Fig. 13-15 Read Timing of Programmable Wait Function (2/2)

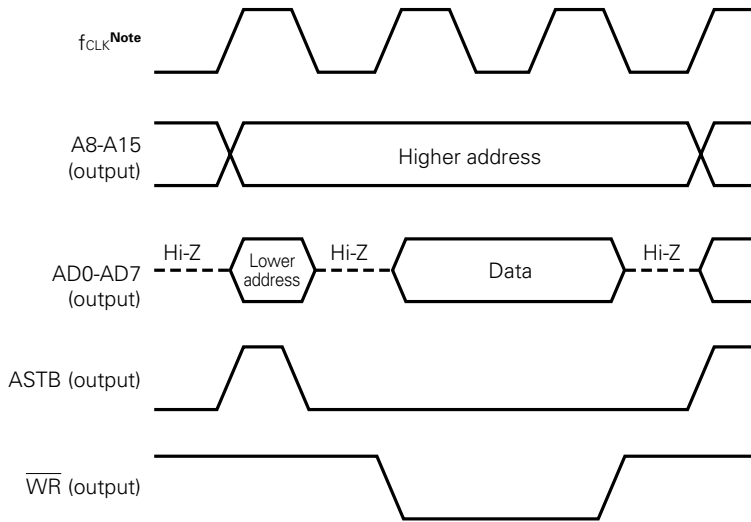
(c) When two wait states are set



Note f_{CLK}: System clock frequency (f_{XX}/2)

Fig. 13-16 Write Timing of Programmable Wait Function (1/2)

(a) When zero wait states are set



(b) When one wait state is set

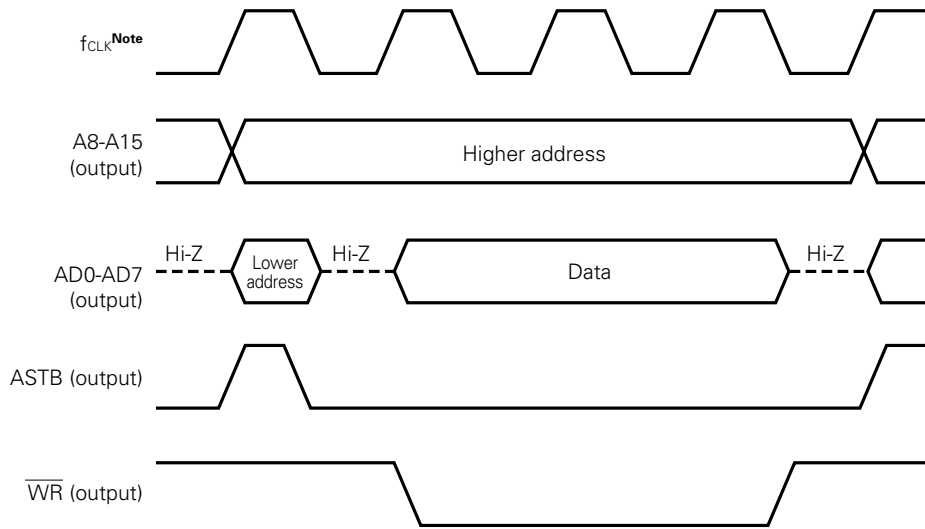
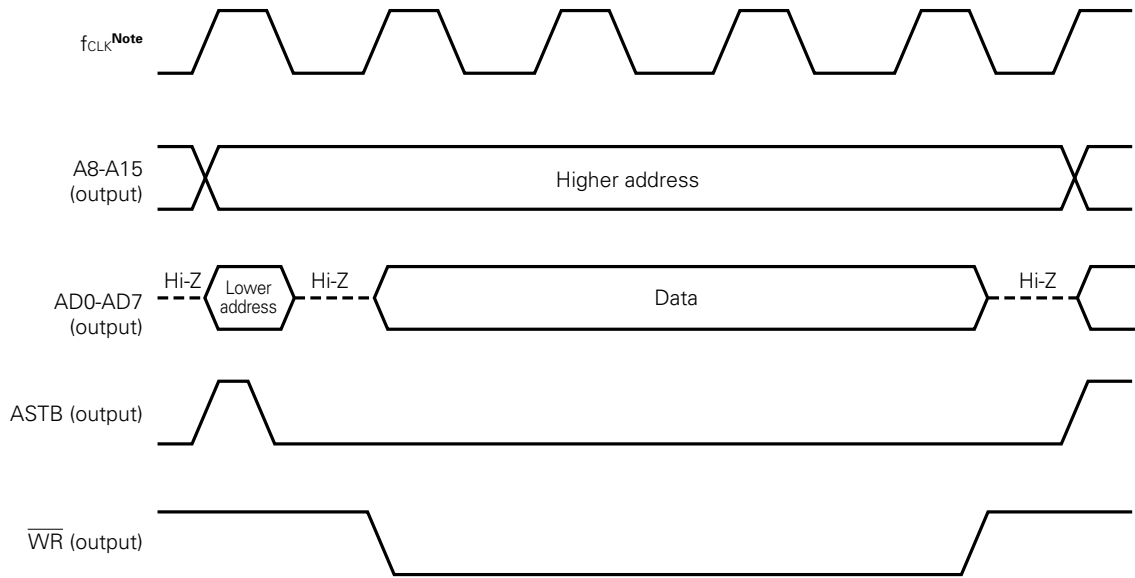


Fig. 13-16 Write Timing of Programmable Wait Function (2/2)

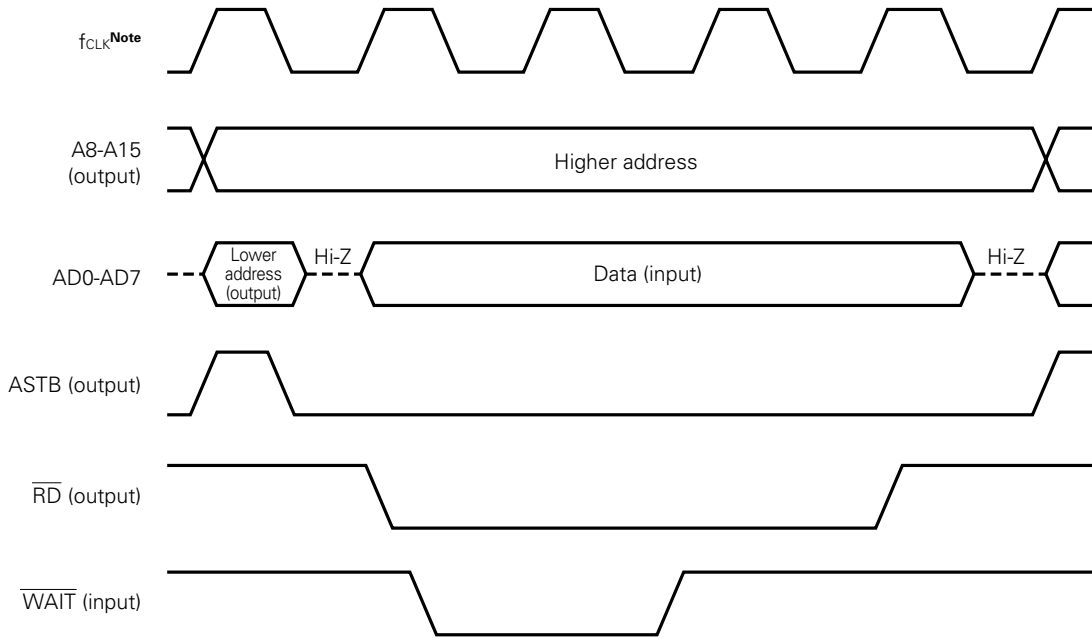
(c) When two wait states are set



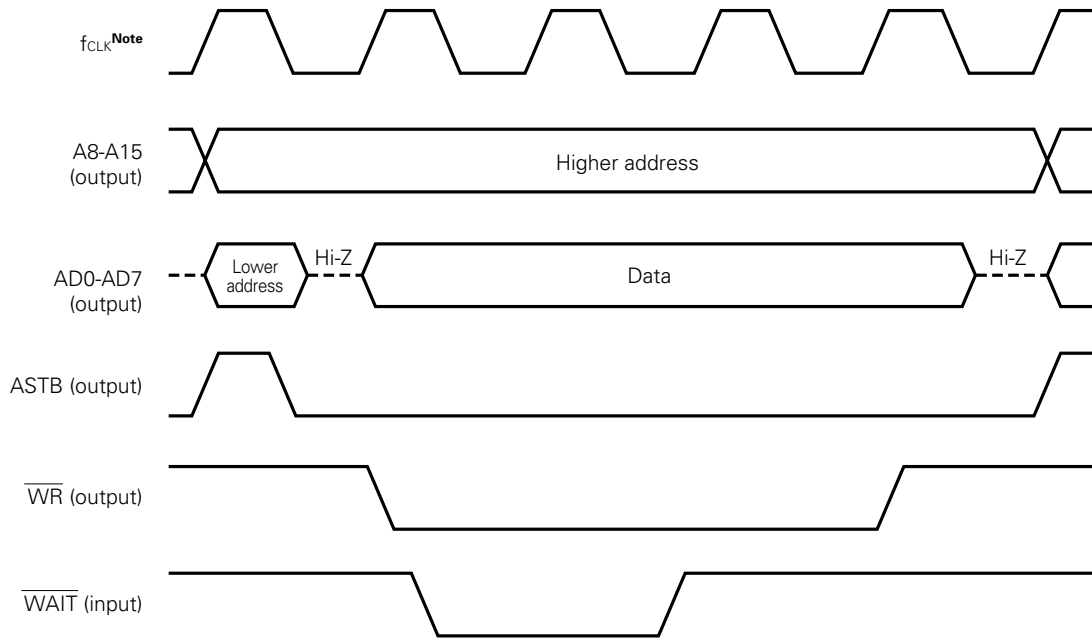
Note f_{CLK}: System clock frequency (f_{XX}/2)

Fig. 13-17 Timing When External Wait Signal Is Used

(a) Read timing



(b) Write timing



Note f_{CLK}: System clock frequency (f_{XX}/2)

13.5 PSEUDO STATIC RAM REFRESH FUNCTION

13.5.1 Function

The μ PD78214 provides the pseudo static RAM refresh function to enable pseudo static RAM to be connected directly.

The pseudo static RAM refresh function outputs refresh pulses at arbitrary intervals. The refresh pulse output cycle period is specified in the refresh mode register (RFM), and the external access cycle is changed to the refresh bus cycle that matches the pseudo static RAM bus cycle. (The write pulse width becomes shorter by half a clock pulse than that observed when pseudo static RAM is not connected.)

The μ PD78214 provides a function to support self-refresh and thus reduce the power dissipation of pseudo static RAM application systems.

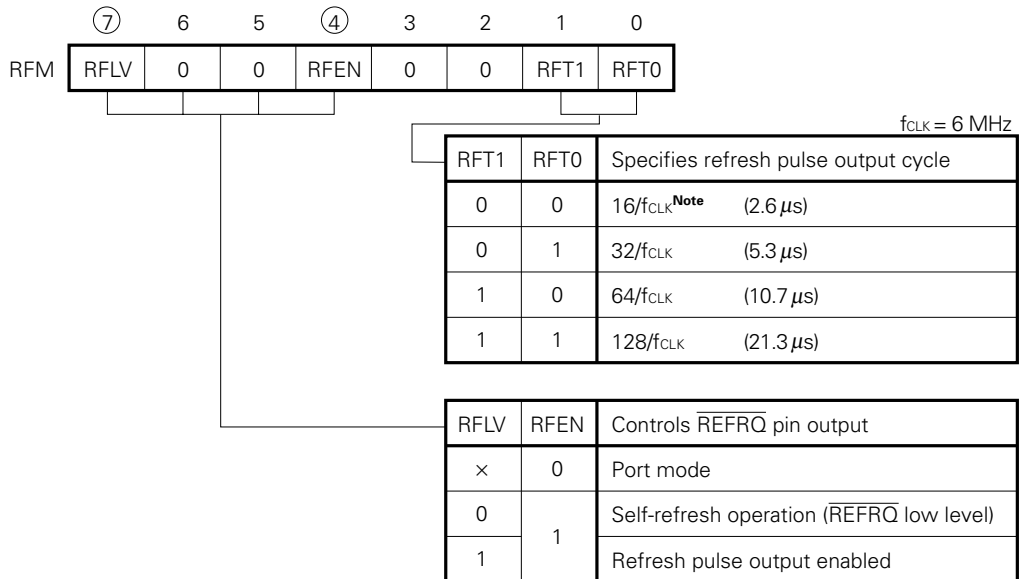
13.5.2 Refresh Mode Register (RFM)

The RFM register is an 8-bit register that controls the refresh cycle of pseudo static RAM and switching to self-refresh.

The register can be read and written with 8-bit manipulation instructions and bit manipulation instructions. Fig. 13-18 shows the format of the register.

When the $\overline{\text{RESET}}$ signal is applied, the register is set to 00H. The $\overline{\text{REFRQ}}$ pin is set to port mode such that it functions as pin P67.

Fig. 13-18 Format of Refresh Mode Register (RFM)



Remark x: 0 or 1

Note f_{CLK} : System clock frequency ($f_{\text{XX}}/2$)

13.5.3 Operation

(1) Pulse refresh operation

To support the pulse refresh cycle of pseudo static RAM, the $\overline{\text{REFRQ}}$ pin outputs refresh pulses, synchronized with the bus cycle.

Adjust the oscillator frequency and bits 1 and 0 (RFT1 and RFT0) of the refresh mode register (RFM) so that at least 512 refresh pulses are output in 8 ms.

Table 13-2 System Clock Frequency and Refresh Pulse Output Cycle When Pseudo Static RAM Is Used

System clock frequency (f_{CLK}) MHz	Refresh pulse output cycle	RFT1	RFT0
$4.096 < f_{\text{CLK}} \leq 6$ ($8.192 < f_{\text{XX}} \leq 12$)	$64/f_{\text{CLK}}$	1	0
$2.048 < f_{\text{CLK}} \leq 4.096$ ($4.096 < f_{\text{XX}} \leq 8.192$)	$32/f_{\text{CLK}}$	0	1
$2 < f_{\text{CLK}} \leq 2.048$ ($4 \leq f_{\text{XX}} \leq 4.096$)	$16/f_{\text{CLK}}$	0	0

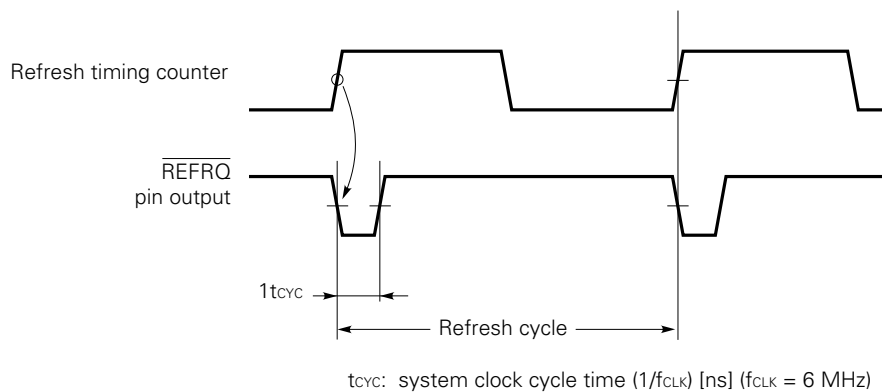
Pulse refresh is controlled so that it does not overlap external memory access. During the refresh cycle, the external memory access cycle is held pending ($\overline{\text{ASTB}}$, $\overline{\text{RD}}$, and $\overline{\text{WR}}$ are inactive). During the external memory access cycle, the refresh cycle is held pending.

When pulse refresh does not cause a contention with external memory access, the refresh cycle can be executed without affecting instruction execution by the CPU.

(a) Accessing internal memory

Even when external pseudo static RAM is not accessed, and internal memory is accessed, the refresh bus cycle is output at the intervals specified by the RFM register, ensuring that the contents of pseudo static RAM are retained.

Fig. 13-19 Pulse Refresh When Internal Memory Is Accessed

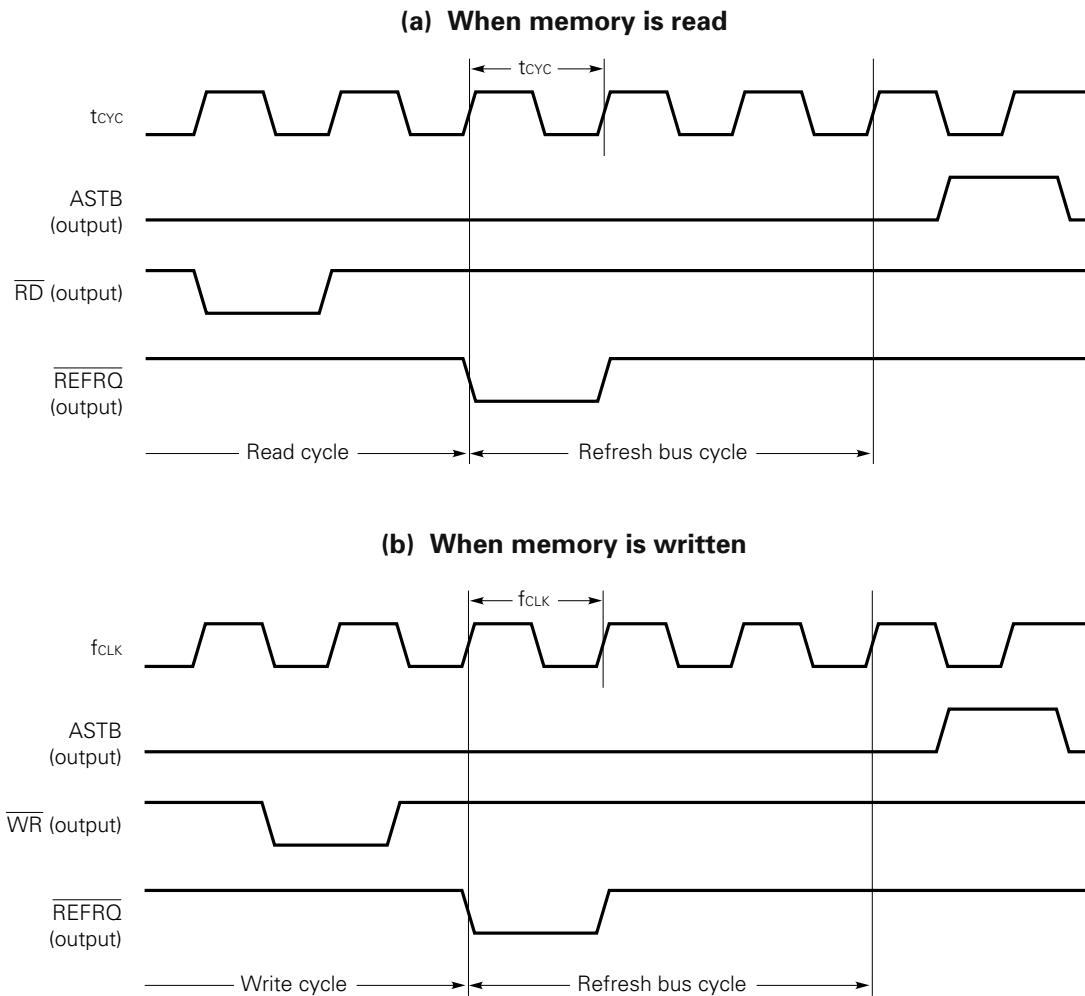


(b) Accessing External Memory

The refresh bus cycle is generated at the intervals specified with the refresh mode register (RFM).

Pseudo static RAM may malfunction if the access timing overlaps the refresh pulse output timing; therefore, the μ PD78214 generates a refresh bus cycle of three clock pulses, synchronized with the bus cycle.

Fig. 13-20 Pulse Refresh When External Memory Is Accessed



(2) Self-refresh

Self-refresh is performed to retain the contents of pseudo static RAM when in standby mode.

(a) Setting self-refresh mode

When bit 4 (RFEN) of the RFM register is set to 1, and bit 7 (RFLV) is set to 0, pin $\overline{\text{REFRQ}}$ outputs a low-level signal, requesting pseudo static RAM to enter self-refresh mode.

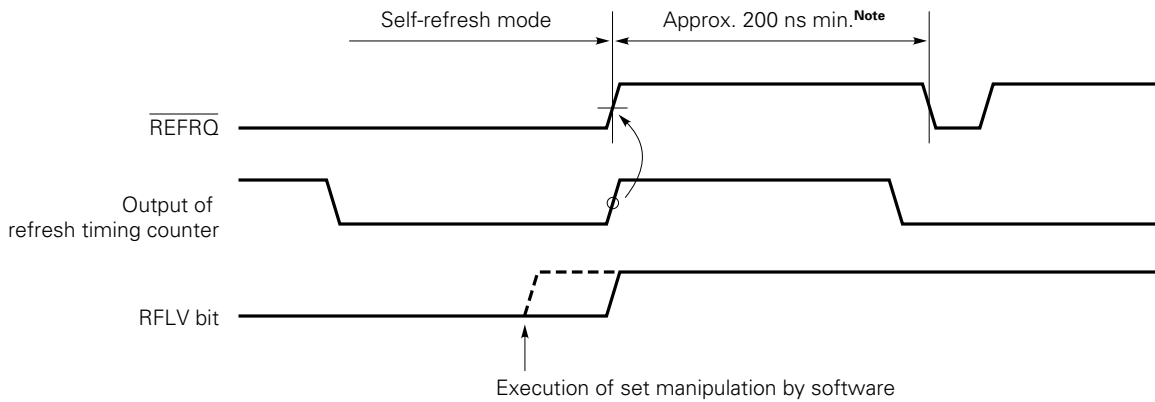
(b) Restoration from self-refresh

Refresh pulse output to pseudo static RAM is disabled for approximately 200 ns^{Note} after the output level of pin $\overline{\text{REFRQ}}$ goes high. The μPD78214 drives pin $\overline{\text{REFRQ}}$ high, synchronized with the refresh timing counter, so that refresh pulses are not output during a refresh disabled period.

To detect pin $\overline{\text{REFRQ}}$ going high, the read out level of bit RFLV is set to 1 when pin $\overline{\text{REFRQ}}$ goes high.

Note This time varies with the speed of the pseudo static RAM.

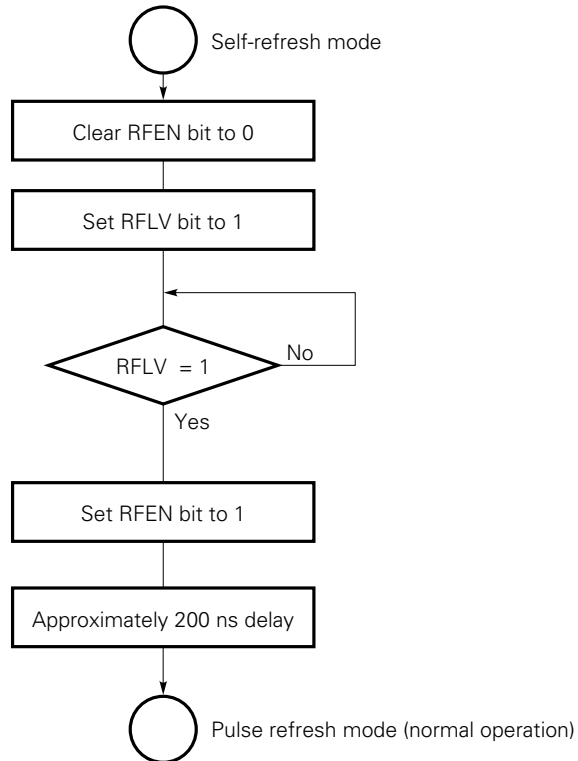
Fig. 13-21 Restoration Timing from Self-Refresh



Note Refresh disabled period

Caution If the RFEN bit of the refresh mode register (RFM) is already set to 1 (or is simultaneously set to 1) when the RFLV bit is changed from 0 to 1, pin $\overline{\text{REFRQ}}$ may output a glitch, having a peak level of approximately 2.6 V, for approximately 10 ns. When setting the RFLV bit to 1, follow the steps shown in Fig. 13-22. The 200-ns delay after setting RFEN assures the access disabled time when pseudo static RAM returns from self-refresh mode.

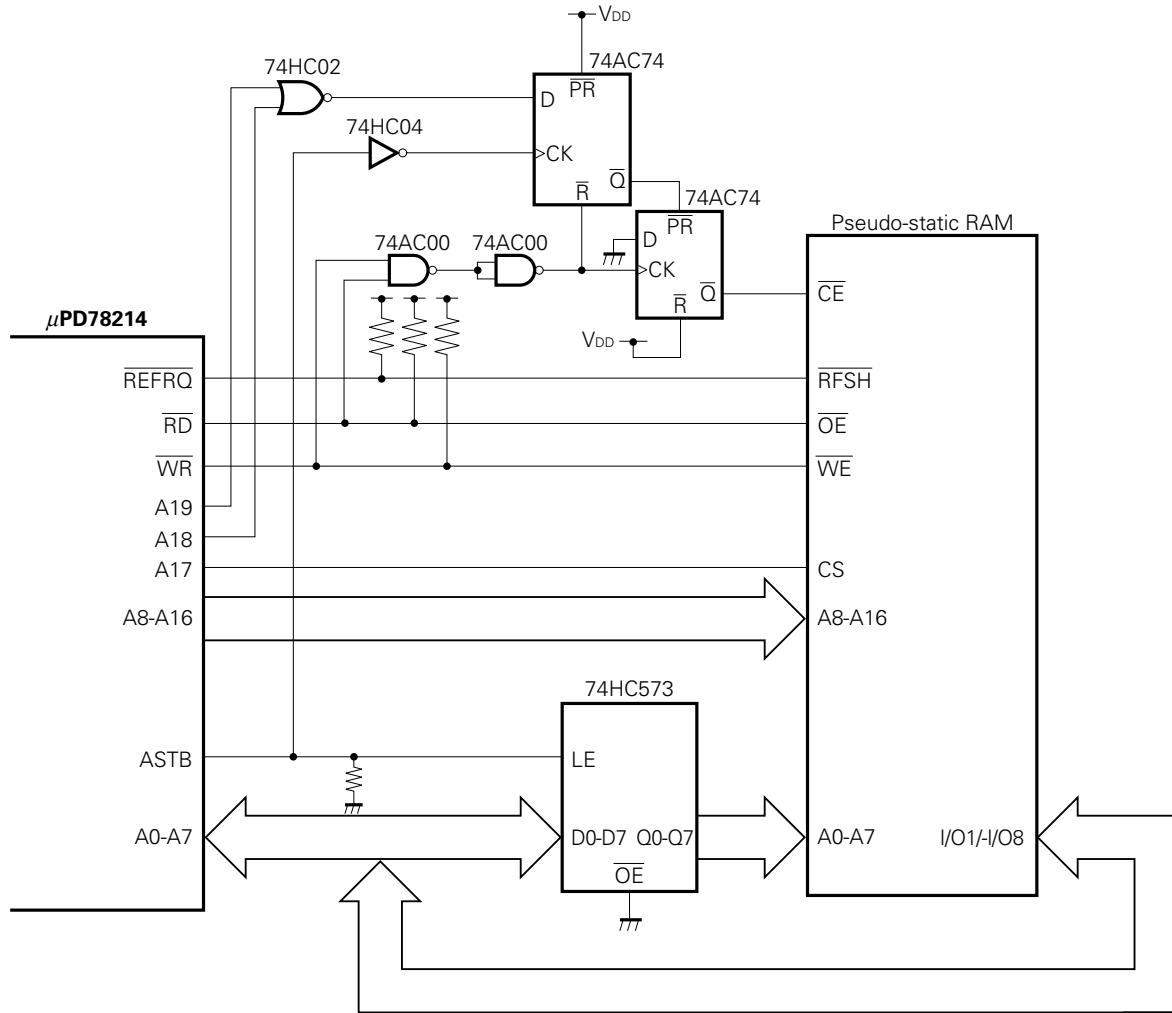
Fig. 13-22 Return from Self-Refresh



13.5.4 Example of Connecting Pseudo Static RAM

Fig. 13-23 shows an example of connecting pseudo static RAM to the μPD78214. In this example, pseudo static RAM is assigned to addresses 20000H to 3FFFFH.

Fig. 13-23 Example of Connecting Pseudo Static RAM to μPD78214



- Remarks**
1. To ensure the precharge and access times for pseudo static RAM, devices of a sufficiently high speed must be used for those devices identified as 74ACxx. To satisfy the precharge time for pseudo static RAM, use a high-speed product.
 2. Pull-up resistors must be connected to the address and address/data bus lines.

13.6 NOTES

- (1) Address information output on P50/A8 to P57/A15 and on P60/A16 to P63/A19 is valid from when the ASTB signal goes high until the \overline{RD} or \overline{WR} signal goes high. Except during this period, the output levels of P50/A8 to P57/A15 and of P60/A16 to P63/A19 are undefined. When designing circuits, take this into consideration, and make sure that the output of an undefined value will not cause any problems.

The data sheet of the relevant product gives the specification of the valid period for address output.

- (2) External devices cannot be mapped onto the same addresses as those of the internal RAM area (μPD78213, μPD78214, and μPD78P214: 0FD00H to 0FEFFH, μPD78212: 0FD80H to 0FEFFH) and SFR area (0FF00H to 0FFFFH, excluding the external SFR area (0FF00H to 0FFDFH)).

Upon manipulating the space where external device addresses overlap with internal RAM or SFR addresses, the internal RAM or SFR area is accessed automatically. In this case, the address signal is output, but the ASTB, \overline{RD} , and \overline{WR} signals are not output (these signals remain inactive).

- (3) When macro service Type A or Type C is used in external memory expansion mode (the μ PD78213 always uses external memory), an illegal write access may occur.

This occurs when any of the following three conditions is satisfied:

- (a) Data is transferred from memory to an SFR using macro service Type A, and the transfer data is D0H to DFH.
- (b) Data is transferred from an SFR to memory by using macro service Type A, and the transfer destination buffer (memory) address is 0FED0H to 0FEDFH upon execution of the macro service.
- (c) The MPTL address is 0FED0H to 0FEDFH when macro service Type C is used.

An illegal write access is performed in the same way as during normal memory access. In addition, wait states are inserted according to the setting of bits PW20 and PW21 of the memory expansion mode register (MM). Table 13-3 lists the conditions and operations for illegal write access.

Table 13-3 Conditions and Operations for Illegal Write Access

Condition	Macro service type	Illegal write access	
		Address	Data
1	A	Transfer destination SFR address	Data transferred by macro service
2	A	Transfer target SFR address	Low-order 8 bits of transfer destination buffer (memory) address
3	C	Transfer destination SFR (CR10 or CR11) address	Low-order 8 bits of MPTL address

This problem can be avoided by applying the following methods:

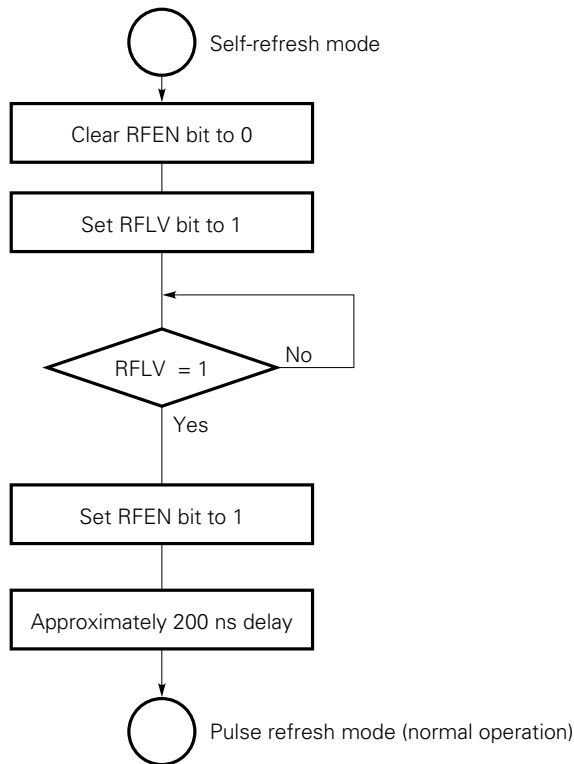
1. The problem caused by condition 1 is difficult to avoid by means of software (whether an illegal access occurs depends on the transfer data). The problem must be avoided by using the external address decoder so that the area at addresses 0FF00H to 0FFFFH does not overlap the addresses of the external circuits.
2. When the macro service being used does not satisfy condition 1 (transfer from memory to an SFR is not performed by macro service Type A), or when condition 2 is satisfied, the buffer area must be mapped to an address other than addresses 0FED0H to 0FEDFH. In the case of condition 3, the MPTL must be mapped to an address other than addresses 0FED0H to 0FEDFH.

This problem can also occur in the in-circuit emulator.

- (4) When using the external wait signal, set bit 6 of register PM6 to 1 to set pin P66/ $\overline{\text{WAIT}}$ to input mode.
- (5) If the RFEN bit of the refresh mode register (RFM) is already set to 1 (or is simultaneously set to 1) when the RFLV bit is changed from 0 to 1, pin $\overline{\text{REFRQ}}$ may output a glitch, having a peak level of approximately 2.6 V, for approximately 10 ns.

When setting the RFLV bit to 1, follow the steps shown in Fig. 13-24. The 200-ns delay after setting RFEN assures the access disabled time when pseudo static RAM returns from self-refresh mode.

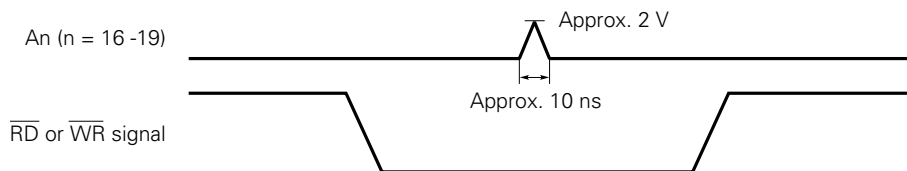
Fig. 13-24 Return from Self-Refresh



(6) When using the in-circuit emulator, note the following points:

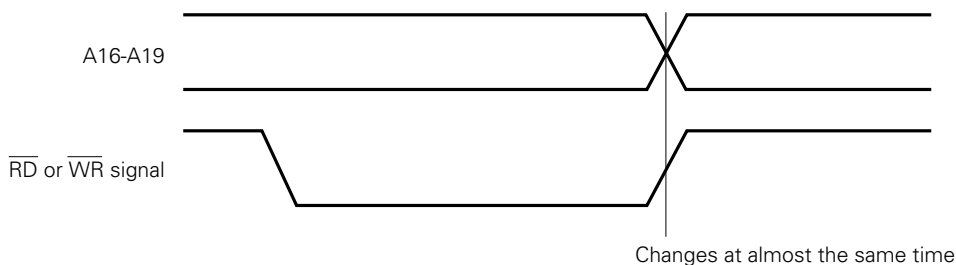
- When the \overline{RD} signal or \overline{WR} signal is active, a glitch may occur on pins A16 to A19.

Fig. 13-25 Glitch Observed on Pins A16 to A19 during Emulation

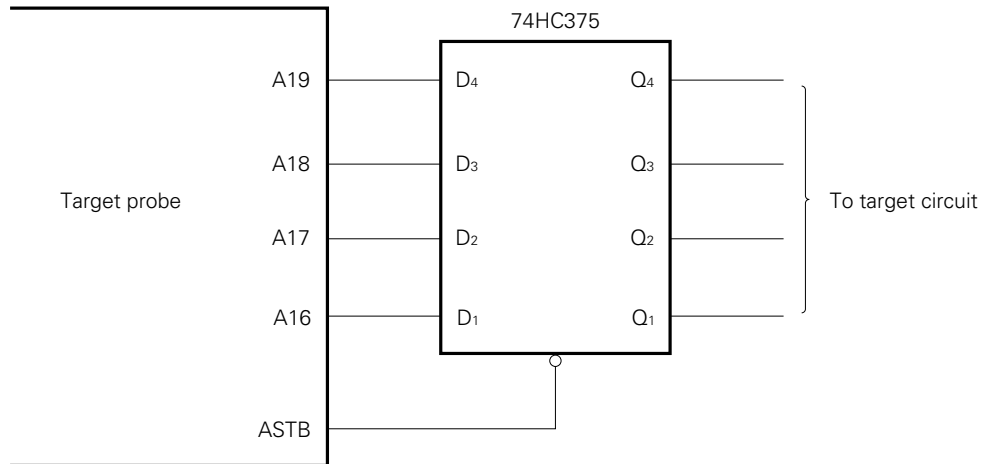


- For the \overline{RD} and \overline{WR} signals, the hold time of the address signals on pins A16 to A19 is almost 0 ns.

Fig. 13-26 Insufficient Address Hold Time during Emulation



To prevent these problems, it is recommended that a latch be provided for pins A16 to A19 when emulation is performed (the latch is not necessary for the device).

Fig. 13-27 Preventing Problems That May Occur during Emulation

CHAPTER 14 STANDBY FUNCTION

14.1 FUNCTION OVERVIEW

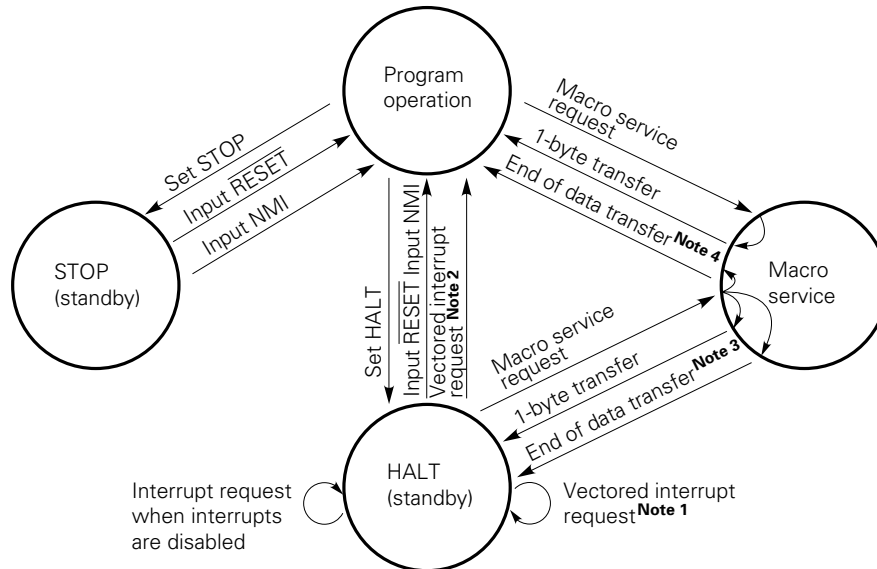
The μ PD78214 supports a standby function to reduce the system's power consumption. With the standby function, two modes are available:

- HALT mode: In this mode, only the CPU clock is stopped. Intermittent operation, when combined with normal operating mode, can reduce overall system power consumption.
- STOP mode: In this mode, the oscillator is stopped to stop the entire system. Since only leakage currents are allowed to flow in this mode, the system's power consumption is minimized.

Each mode is specified by software. Fig. 14-1 is the transition diagram for the standby modes (STOP and HALT modes).

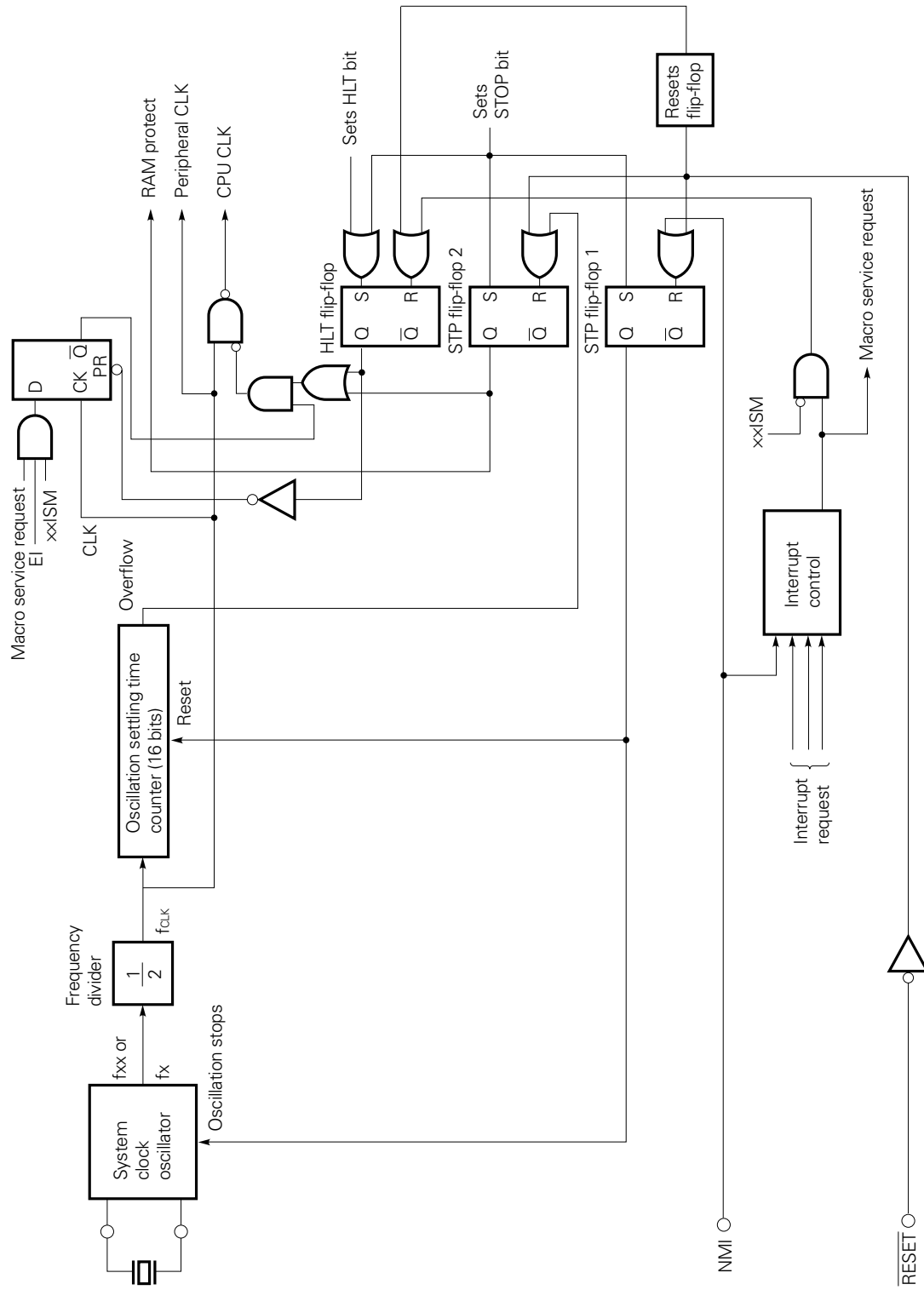
Fig. 14-2 shows the functional block that implements the standby functions.

Fig. 14-1 Transition Diagram for the Standby Modes



- Notes**
1. When a vectored interrupt with a low priority is requested when it is disabled at the start of HALT mode
 2. When a vectored interrupt with a high priority is requested, or when a vectored interrupt with a low priority is requested when it is enabled at the start of HALT mode
 3. When a macro service with a low priority is requested when low-priority interrupts are disabled at the start of HALT mode
 4. When a macro service with a high priority is requested, or when a macro service with a low priority is requested while low-priority interrupts are enabled at the start of HALT mode

Fig. 14-2 Standby Function Block

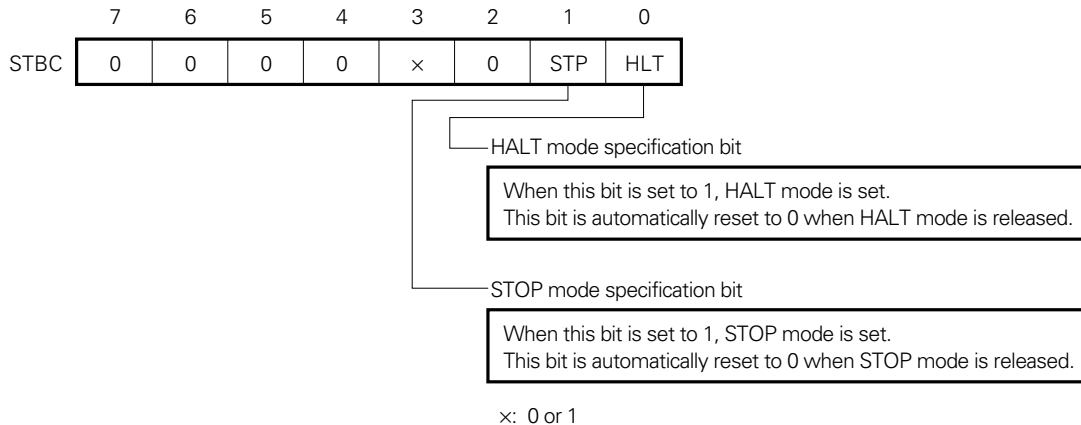


14.2 STANDBY CONTROL REGISTER (STBC)

The standby control register (STBC) is an 8-bit register which controls standby mode. The STBC register can be both read and written. Only a specified instruction (MOV STBC, #byte), however, can be used for writing to the register, to prevent the application system stopping unintentionally as a result of a program crash. Fig. 14-3 shows the format of the STBC register.

When the $\overline{\text{RESET}}$ signal is input, the register is set to 0000x00B.

Fig. 14-3 Configuration of the Standby Control Register (STBC)



14.3 HALT MODE

14.3.1 Specifying HALT Mode and Operation States in HALT Mode

The system enters HALT mode when the HLT bit of the STBC register is set to 1.

The STBC register can be written only with a specified 8-bit data write instruction. When specifying HALT mode, execute the "MOV STBC, #01H" instruction.

Caution If HALT mode is specified under the conditions for releasing HALT mode, the system does not enter HALT mode, instead executing the next instruction or branching to the vectored interrupt service program. Clear any interrupt requests before specifying HALT mode to ensure that the system enters HALT mode correctly.

Table 14-1 Operation States in HALT Mode

Clock oscillator	Operating	
Internal system clock	Operating	
CPU	Stopped ^{Note}	
I/O lines	Same as before HALT mode	
Peripheral functions	Operating	
Internal RAM	Contents maintained	
Bus lines	AD0-AD7	High-impedance
	A8-A15	States maintained
	A16-A19	Low
$\overline{\text{RD}}$, $\overline{\text{WR}}$ output	High	
ASTB output	Low	

Note Macro services are executed.

14.3.2 Releasing HALT Mode

HALT mode can be released by any of the following three sources:

- Nonmaskable interrupt request (NMI)
- Maskable interrupt request (vectored interrupt or macro service)
- $\overline{\text{RESET}}$ input

Table 14-2 lists the sources used for releasing HALT mode and the operations that are performed after HALT mode is released by each source.

Table 14-2 Sources for Releasing HALT Mode and Operations Performed after Release

Releasing source	MK _{xx}	PR _{xx}	IE	ISP	Operation
$\overline{\text{RESET}}$ input	×	×	×	×	Ordinary reset operation
Nonmaskable interrupt request	—	—	×	×	Handling a vectored interrupt ^{Note}
Maskable vectored interrupt request	0	0	1	×	Handling a vectored interrupt
	0	×	1	1	
	0	0	0	×	Executing the instruction at the next address (holding the interrupt request)
	0	×	0	1	
	0	1	×	0	Continuing HALT mode (holding the interrupt request)
	1	×	×	×	
Macro service request	0	0	1	×	Executing the macro service When the end conditions are satisfied, a vectored interrupt is handled.
	0	×	1	1	When the end conditions are not satisfied, HALT mode is resumed.
	0	0	0	×	Executing the macro service When the end conditions are satisfied, the instruction at the next address is executed.
	0	×	0	1	When the end conditions are not satisfied, HALT mode is resumed.
	0	1	×	0	Executing the macro service then resuming HALT mode (holding the interrupt request)
	1	×	×	×	Continuing HALT mode

Note When the NMIS bit of the interrupt status register (LST) is set to 1, the instruction at the next address is executed. Processing then branches to the NMI interrupt service program when NMIS is cleared to 0.

Remark MK_{xx} : Interrupt mask flag
 PR_{xx} : Priority designation flag
 IE : Interrupt request enable flag
 ISP : Interrupt priority status flag

(1) Release by a nonmaskable interrupt (NMI) request

When a nonmaskable interrupt (NMI) is requested, HALT mode is released regardless of whether interrupts are enabled (EI) or disabled (DI).

Once HALT mode has been released, processing branches to the NMI service program if the NMIS bit of the interrupt status register (IST) is set to 0. If the NMIS bit is set to 1 (for example, when HALT mode is specified in the NMI interrupt service program), processing is resumed from the instruction subsequent the one that has specified HALT mode. Processing then branches to the NMI interrupt service program when the NMIS bit is set to 0 (for example, with a RETI instruction).

(2) Release by a maskable interrupt request

Only maskable interrupts with 0 in the interrupt mask flag can be used to release HALT mode. If the interrupt priority status flag (ISP) is set to 0 (only high-priority interrupts are enabled), only interrupts with 0 in the priority designation flag (high priority) can release HALT mode. Macro services are, however, executed regardless of their priorities (an interrupt to be generated at the end of each macro service is subject to priority control).

Once HALT mode has been released, processing branches to the interrupt handling program if the interrupt request enable flag (IE) is set to 1. If the IE flag is set to 0, processing is resumed from the instruction subsequent to that which specified HALT mode.

When a macro service is requested, HALT mode is temporarily released to execute the macro service. After the macro service is executed, HALT mode is resumed. Once the macro service has been executed the specified number of times, subsequent operations depend on the IE flag, ISP flag, and priority designation flag.

Table 14-3 Release of HALT Mode by a Maskable Interrupt Request

Releasing source	MK $\times\times$	PR $\times\times$	IE	ISP	Operation
Maskable vectored interrupt request	0	0	1	\times	Handling a vectored interrupt
	0	\times	1	1	
	0	0	0	\times	Executing the instruction at the next address (holding the interrupt request)
	0	\times	0	1	
	0	1	\times	0	Continuing HALT mode
	1	\times	\times	\times	
Macro service request	0	0	1	\times	Executing the macro service When the end conditions are satisfied, a vectored interrupt is handled.
	0	\times	1	1	When the end conditions are not satisfied, HALT mode is resumed.
	0	0	0	\times	Executing the macro service When the end conditions are satisfied, the instruction at the next address is executed.
	0	\times	0	1	When the end conditions are not satisfied, HALT mode is resumed.
	0	1	\times	0	Executing the macro service then resuming HALT mode
	1	\times	\times	\times	Continuing HALT mode

Remark MK $\times\times$: Interrupt mask flag
 PR $\times\times$: Priority designation flag
 IE : Interrupt request enable flag
 ISP : Interrupt priority status flag

(3) Release by $\overline{\text{RESET}}$ input

The program is resumed from the reset vector address in the same way as for an ordinary reset operation, except that the contents of internal RAM are the same as those existing immediately before entering HALT mode.

14.4 STOP MODE

14.4.1 Specifying STOP Mode and Operation States in STOP Mode

The system enters STOP mode when the STP bit of the STBC register is set to 1.

The STBC register can be written only with a specified 8-bit data write instruction. To specify STOP mode, execute the “MOV STBC, #02H” instruction.

Table 14-4 Operation States in STOP Mode

Clock oscillator		Operating
Internal system clock		Operating
CPU		Stopped
I/O lines		Same as before entering STOP mode
Peripheral functions		Stopped ^{Note}
Internal RAM		Contents maintained
Bus lines	AD0-AD7	High-impedance
	A8-A15	States maintained
	A16-A19	Low
\overline{RD} , \overline{WR} output		High
ASTB output		Low

Note The A/D converter stops, but its current dissipation is not reduced if the CS bit of the A/D converter mode register (ADM) is set.

Cautions 1. In STOP mode, the X1 pin is internally short-circuited to V_{SS} (ground potential) to prevent current leakage from the clock oscillator circuit. STOP mode must not, therefore, be specified for a system using an external clock.

2. Reset the CS bit for the A/D converter.

3. The system enters STOP mode even if an NMI request is being held when STOP mode is specified. When using an NMI request to release STOP mode, input the NMI signal again.

14.4.2 Releasing STOP Mode

STOP mode can be released by inputting an NMI or \overline{RESET} signal.

(1) Releasing STOP mode by NMI input

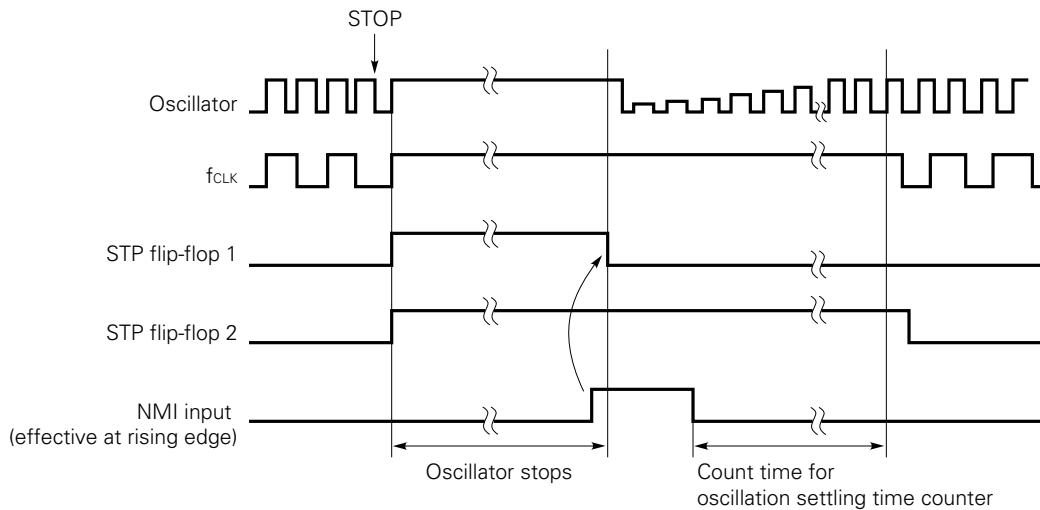
(a) Operation

The oscillator restarts when an effective edge, specified with the external interrupt mode register (INTM0), is detected at the NMI pin. Then, STOP mode is released after the specified time, required to allow the oscillation to settle, has elapsed.

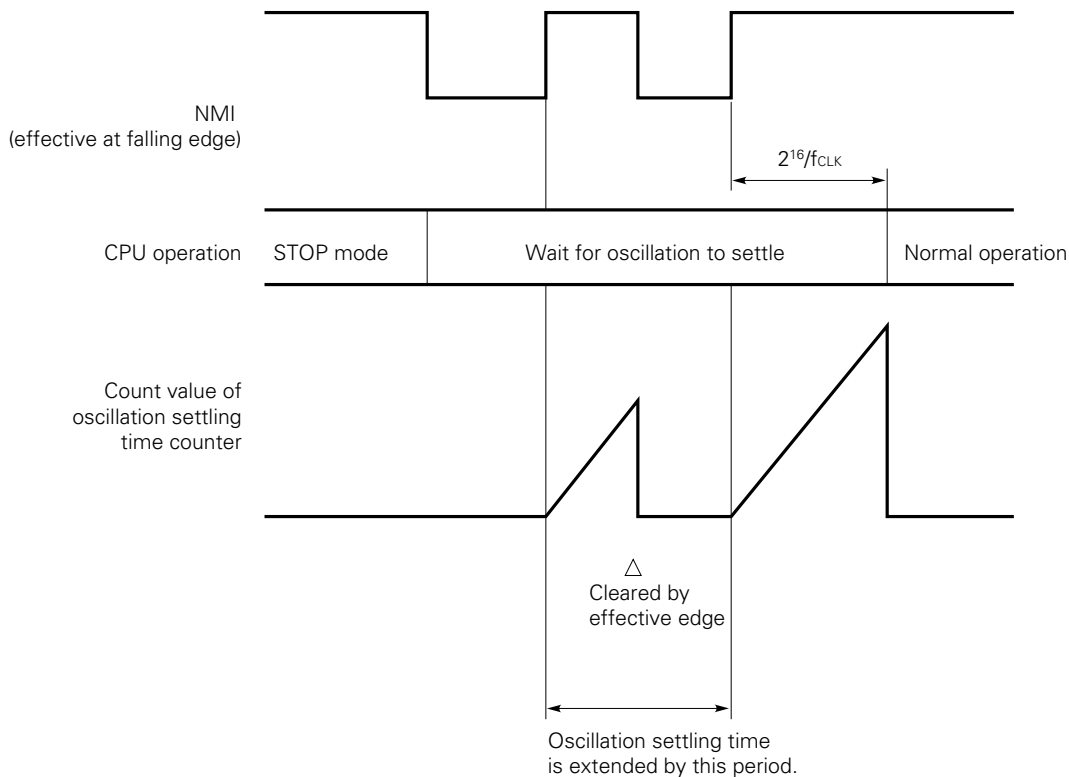
Once STOP mode has been released, processing branches to the NMI service program if the NMIS bit of the interrupt status register (IST) is set to 0. If the NMIS bit is set to 1 (for example, when STOP mode is specified in the NMI interrupt service program), processing is resumed from the instruction subsequent to that which specified STOP mode. Processing then branches to the NMI interrupt service program when the NMIS bit is set to 0 (for example, with a RETI instruction).

(b) Oscillation settling time

The oscillator restarts when an effective, edge specified with the external interrupt mode register (INTM0), is detected at the NMI pin. Then, when the input level of the NMI signal returns to its original level, the oscillation settling time counter starts counting. When the 16-bit counter overflows (in 11 ms when $f_{XX} = 12$ MHz), the internal system clock is started. The system therefore waits, from the detection of the effective edge to the start of the internal clock, a total time equal to the high- or low-level width of the NMI signal after detecting the effective edge and the time needed for the 16-bit oscillation settling time counter to overflow.

Fig. 14-4 Releasing STOP Mode with an NMI Signal

Caution If another effective edge of the NMI signal is detected during the oscillation settling time, the oscillation settling time counter is cleared and restarts counting, resulting in a longer wait time than usual. The extra wait time is the total of the time from the end of the first active level to detection of the next effective edge and the width of the second active level, starting from its effective edge. Fix the NMI signal at the inactive level during the oscillation settling time, to ensure the release of STOP mode. ★

Fig. 14-5 Example of Longer Oscillation Settling Time

(2) Releasing STOP mode by $\overline{\text{RESET}}$ input

The oscillator restarts when the $\overline{\text{RESET}}$ signal is changed from high to low to set the system to the reset state. Keep the $\overline{\text{RESET}}$ signal active until the oscillation settling time has elapsed. When the $\overline{\text{RESET}}$ signal goes high, normal operation starts.

Unlike ordinary reset, the contents of data memory will be the same as those existing immediately before the system enters STOP mode.

14.4.3 Notes on Using STOP Mode

Check the following items to ensure that current consumption is appropriately reduced in STOP mode:

(1) Is the output level of each output pin appropriate?

The appropriate output level of each pin depends on the circuit of the next stage. Select an output level that minimizes current consumption.

- When the circuit of the next stage has a low input impedance, high output causes current to flow from the power supply to the port, thus increasing current consumption. A CMOS IC is an example of such a circuit. CMOS ICs having low impedance when the power is turned off. Apply a low voltage to pins used for output to CMOS ICs, to reduce the current consumption and minimize any degradation in the reliability of the CMOS ICs. A high output may result in latch-up when the power is next turned on.
- Low output may increase the current consumption, depending on the circuit of the next stage. In such a case, apply a high voltage to the relevant pins or place the pins in the high-impedance state to reduce current consumption.

The procedure for setting the output level for each port depends on the state of the port.

- When a port is set to control mode, the output level depends on the state of the internal hardware. The state of the internal hardware must be considered when setting the output level.
- When a port is set to port mode, the output level can be set by writing appropriate data to the output latch for that port, as well as the port mode register, by means of a program.

When a port is set to control mode, setting of the output level is facilitated by changing the setting of the port to port mode.

(2) Is the input level of each input pin appropriate?

The voltage input to each pin must be maintained at a level between V_{SS} and V_{DD} . A voltage that falls outside this range will result in increased current consumption as well as adversely affecting the reliability of the microcomputer. Any intermediate voltage must also be avoided.

(3) Is the built-in pull-up resistor necessary?

Any unnecessary pull-up resistors may increase current consumption and result in the latch-up of other devices. Specify the use of only necessary pull-up resistors.

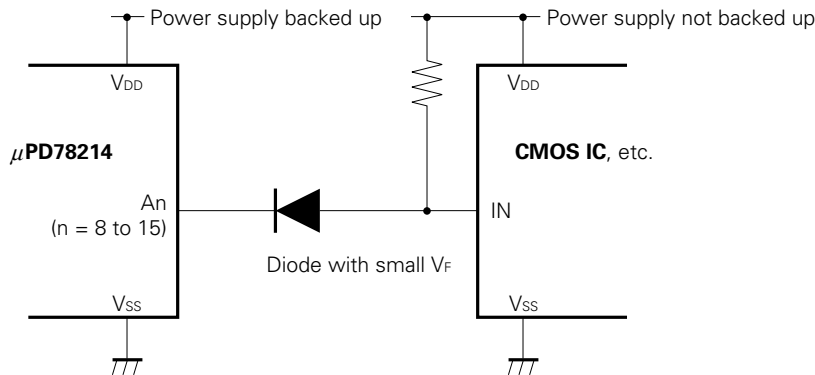
If a port consists both of pins that require pull-up resistors and those that do not require pull-up resistors, connect external pull-up resistors to those pins that require them and specify internal pull-up resistors for a port that is not to be used.

(4) Are the address bus and address/data bus pins arranged appropriately?

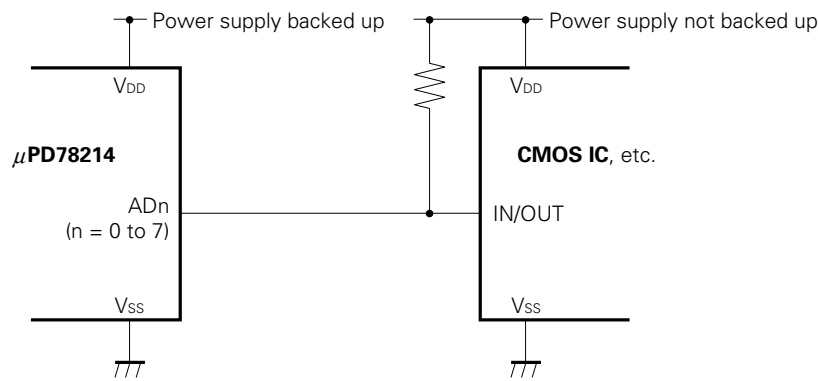
The outputs of the address bus pins are unpredictable (high or low) in STOP mode. If the external circuit has a low input impedance, current flows from the address bus pins, thus increasing the current consumption. The input impedance of the external circuits connected to the address bus pins must therefore be high, even in STOP mode.

In particular, CMOS ICs have a low input impedance when their power is turned off. Therefore, the following arrangements are necessary to prevent current consumption from increasing or the reliability of the ICs from being degraded:

- Do not turn off the CMOS IC (always supply power, even in STOP mode).
- Connect a diode with a small V_F to prevent current from flowing, as shown in Fig. 14-6.

Fig. 14-6 Example of Address Bus Arrangement

The outputs of the address/data bus pins are high-impedance in STOP mode. The address/data bus pins are usually pulled up with pull-up resistors. If a pull-up resistor is connected to a power supply which is backed up, current flows through the pull-up resistor to an external circuit that is connected to a power supply that is not backed up, if the circuit has low input impedance. As a result, current consumption increases. To prevent this, connect pull-up resistors to any power supply that is not backed up, as shown in Fig. 14-7.

Fig. 14-7 Example Address/Data Bus Arrangement

The outputs of the \overline{RD} , \overline{WR} , \overline{ASTB} , and \overline{REFRQ} pins are fixed in STOP mode and, therefore, require the same arrangements as the address bus pins. The \overline{ASTB} pin, which outputs a low voltage, usually does not require any external parts.

The level of the voltage input to the \overline{WAIT} pin must be maintained between V_{SS} and V_{DD} . Any voltage falling outside this range increases the current consumption as well as adversely affecting the reliability of the microcomputer.

For the $\mu PD78214$, you can prevent problems related to address/data bus pins simply by specifying port mode for the pins. The $\mu PD78213$, however, requires that the above arrangements be implemented.

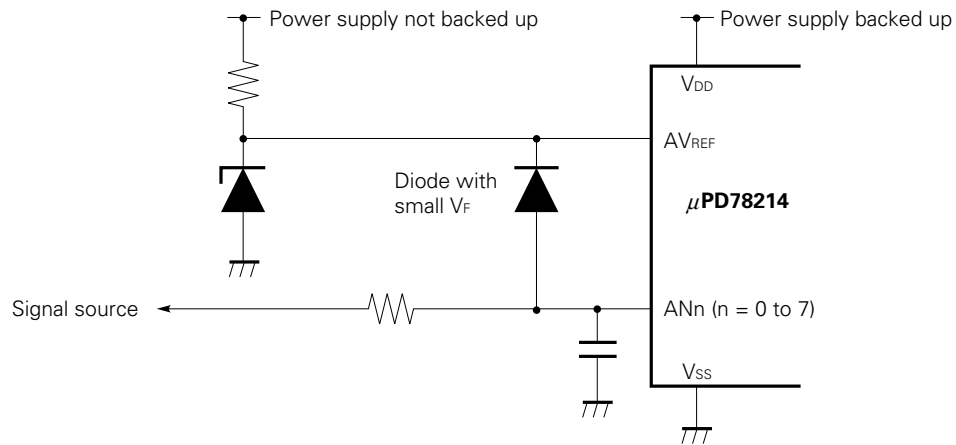
(5) A/D converter

When the CS bit (bit 7) of the A/D converter mode register (ADM) is reset to 0, the current flowing from the AV_{REF} pin is reduced. To reduce the current further, cut off the current supplied to the AV_{REF} pin by means of an external circuit. In this case, however, a voltage higher than that of the AV_{REF} pin must not be applied to the following pins:

- Pin selected with bits ANI0 to ANI2 when the MS bit of the ADM register is 0
- AN0 pin when the MS bit of the ADM register is 1

Fig. 14-8 shows an example arrangement for preventing voltages higher than that of the AV_{REF} pin. Note, however, that in this case, response to changes in the input signal may become slow due to the time constant derived from C and R on the input line.

Fig. 14-8 Example Arrangement for Analog Input Pin

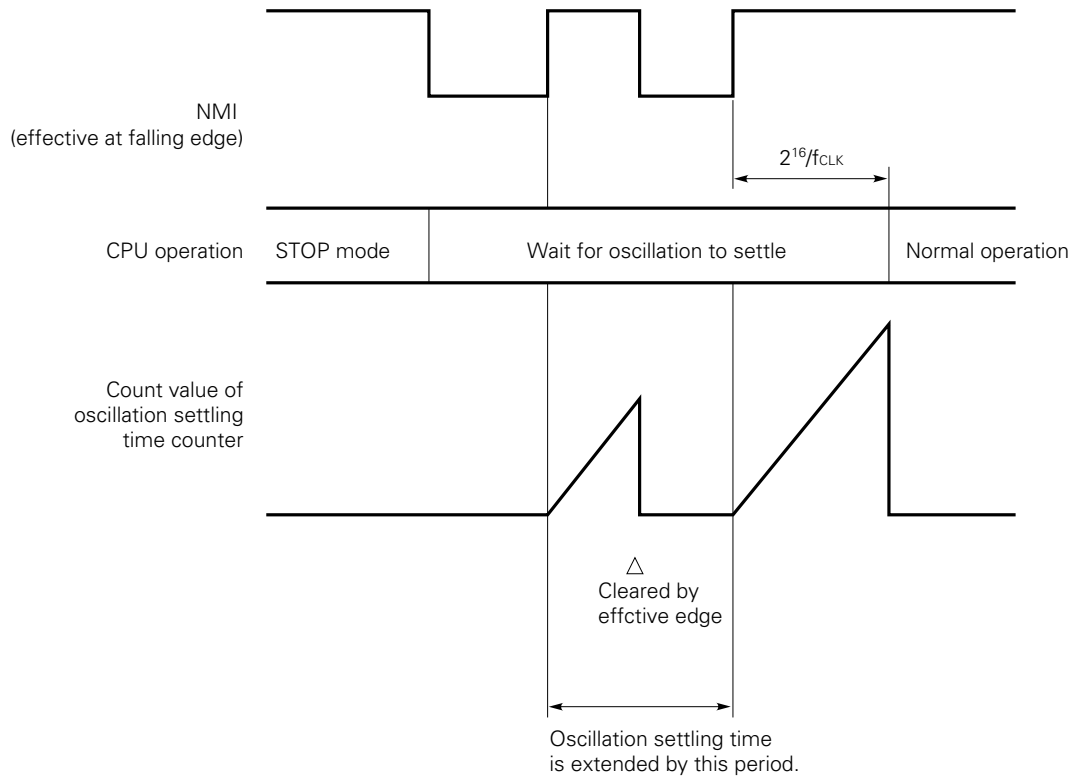


The voltage input to the AN0 to AN7 pins must be maintained at a level between V_{SS} and V_{DD} . Any voltage falling outside this range increases the current consumption as well as adversely affecting the reliability of the microcomputer.

14.5 NOTES

- (1) If HALT mode is specified under the conditions for releasing HALT mode, the system does not enter HALT mode, instead executing the next instruction or branching to the vectored interrupt service program. Clear any interrupt requests before specifying HALT mode to ensure that the system correctly enters HALT mode.
- (2) In STOP mode, the X1 pin is internally short-circuited to V_{SS} (ground potential) to prevent current leakage from the clock oscillator circuit. STOP mode must not, therefore, be specified for a system using an external clock.
- (3) Reset the CS bit for the A/D converter before specifying STOP mode.
- (4) The system enters STOP mode even if an NMI request is held when STOP mode is specified. When using an NMI request to release STOP mode, input the NMI signal again.
- ★ (5) If another effective edge of the NMI signal is detected during the oscillation settling time, the oscillation settling time counter is cleared and restarts counting, resulting in a longer wait time than usual. The extra wait time is the total of the time from the end of the first active level to detection of the next effective edge and the width of the second active level, starting from its effective edge. Fix the NMI signal at the inactive level during the oscillation settling time, to ensure the correct release of STOP mode.

Fig. 14-9 Example of Longer Oscillation Settling Time



CHAPTER 15 RESET FUNCTION

15.1 RESET FUNCTION

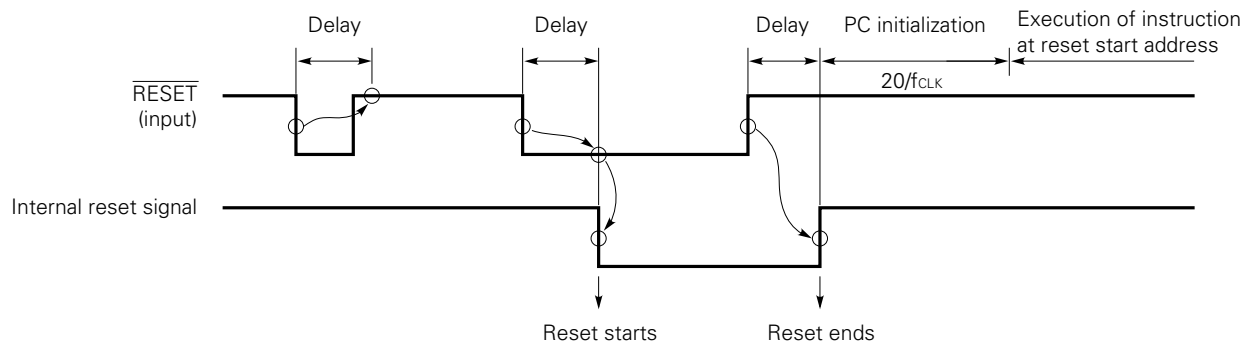
When the signal applied to the $\overline{\text{RESET}}$ input pin is low, the system is reset, and each hardware component is set to the state indicated in Table 15-2. All pins, except the power supply pin, assume the high-impedance state. Table 15-1 lists the states of pins during reset and after the reset state is released.

When the signal applied to the $\overline{\text{RESET}}$ input pin goes high, the reset state is released and program execution branches as follows: The contents of address 00000H in the reset vector table are set in bits 0 to 7 of the program counter (PC), while the contents of 00001H are set in bits 8 to 15 of the PC. The program is resumed from the address set in the PC. You can thus resume the program from an arbitrary address.

Initialize registers in the program as required.

The $\overline{\text{RESET}}$ pin contains a noise eliminator based on analog delays to prevent abnormal operation due to noise (see Fig. 15-1).

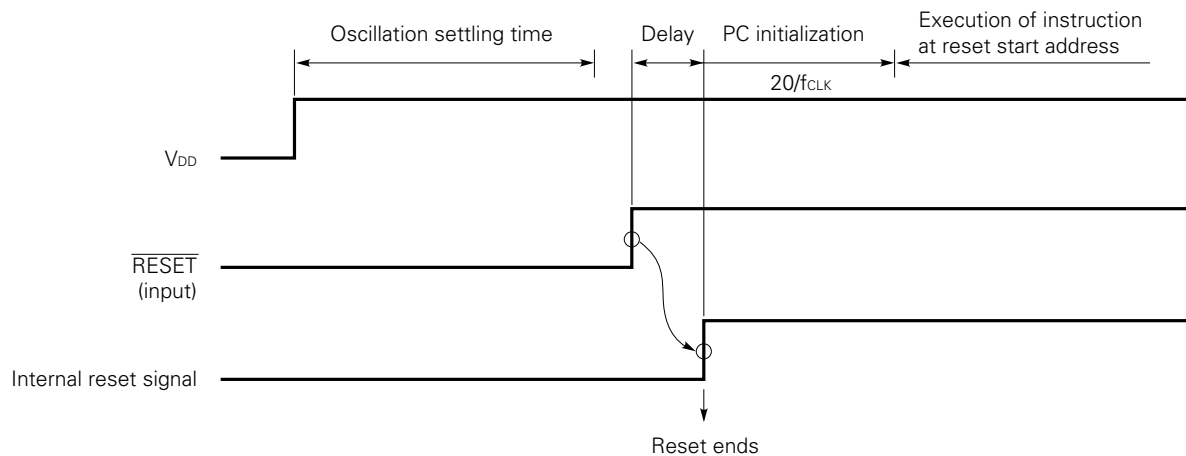
Fig. 15-1 Acceptance of the RESET Signal



Remark f_{CLK} : System clock frequency ($f_{\text{XX}}/2$)

When resetting the system at power-on, keep the $\overline{\text{RESET}}$ signal active until the oscillation settling time (approx. 40 ms, depending on the resonator being used) elapses.

Fig. 15-2 Reset Operation at Power-On



Remark f_{CLK} : System clock frequency ($f_{\text{XX}}/2$)

Table 15-1 Pin States during Reset and After Reset State Is Released

Pin name	I/O	During reset	After reset state is released
P00-P07	Output	Hi-Z	Hi-Z
P20/NMI-27/SI	Input	Hi-Z	Hi-Z (input port)
P30/RxD-P37/T03	I/O	Hi-Z	Hi-Z (input port mode)
P40/AD0-P47/AD7	I/O	Hi-Z	Hi-Z (input port mode) Note
P50/A8-P57/A15	I/O	Hi-Z	Hi-Z (input port mode) Note
P60/A16-P63/A19	Output	Hi-Z	0
P64/ \overline{RD} , P65/ \overline{WR}	I/O	Hi-Z	Hi-Z (input port mode) Note
P66/ \overline{WAIT} /AN6, P67/ \overline{REFRQ} /AN7	I/O	Hi-Z	Hi-Z (input port mode)
P70/AN0-P75/AN5	Input	Hi-Z	Hi-Z (input port mode)
ASTB	Output	Hi-Z	0

Note When ROM-less mode is specified (\overline{EA} pin = 0), these pins function as an address/data bus and output signals for fetching the reset vector address from address 0000H (see **Fig. 15-3 (a)**).

Table 15-2 Hardware States after Reset (1/2)

Hardware		State after reset	
Program counter (PC)		The contents of the reset vector table (0000H and 0001H) are set.	
Stack pointer (SP)		Undefined	
Program status word (PSW)		02H	
Built-in RAM	Data memory	Undefined ^{Note}	
	General registers (X, A, C, B, E, D, L, H)		
Ports	Port 0, 2, 3, 4, 5, and 7	Undefined (high-impedance)	
	Port 6	×0H	
Port mode registers	PM0, PM3, PM5	FFH	
	PM6	F×H	
Port 3 mode control register (PMC3)		00H	
Pull-up-resistor-option register (PUO)		00H	
Memory expansion mode register (MM)		20H	
Timer/ counter unit	16-bit timer/ counters	Timer (TM0)	0000H
		Compare registers (CR00, CR01)	Undefined
		Capture register (CR02)	
	8-bit timer/ counters	Timer (TM1, TM2, TM3)	00H
		Compare registers (CR10, CR20, CR21, CR30)	Undefined
		Capture register (CR22)	
		Capture/compare register (CR11)	
	Timer control registers (TMC0, TMC1)		00H
	Timer output control register (TOC)		
	Capture/compare control registers	CRC0	10H
		CRC1, CRC2	00H
	Prescaler mode registers (PRM0, PRM1)		00H
A/D converter	Mode register (ADM)	00H	
	A/D conversion result register (ADCR)	Undefined	

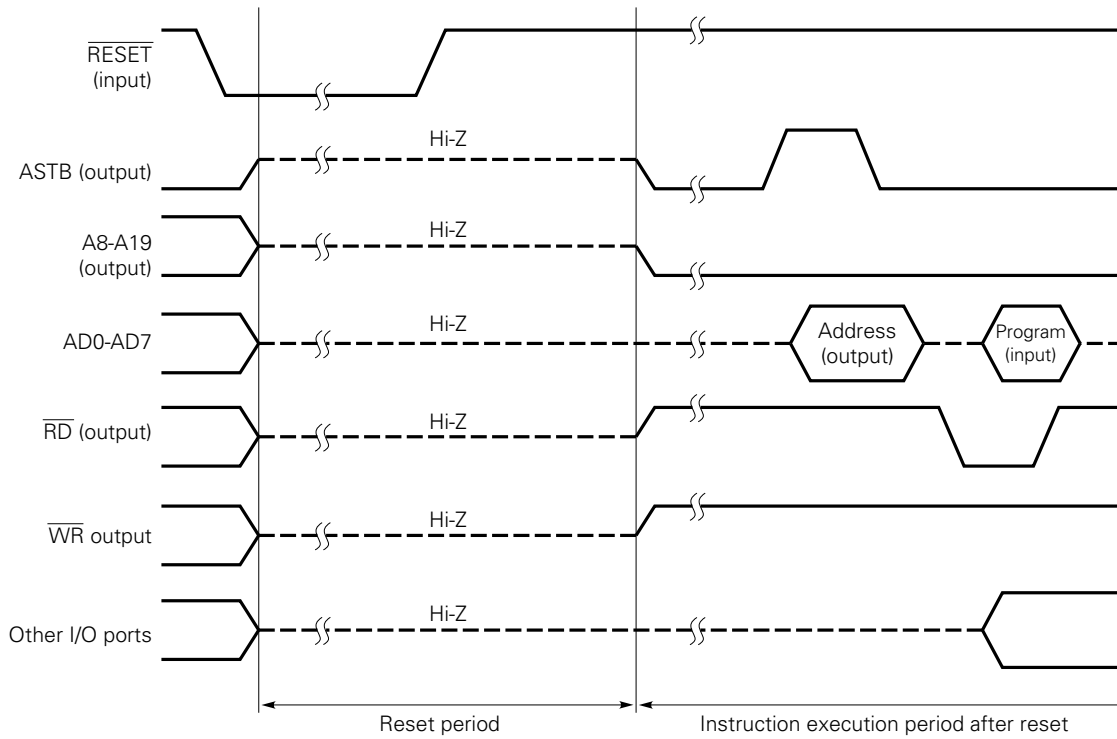
Note When STOP mode is released by a $\overline{\text{RESET}}$ signal, the values stored immediately before setting STOP mode are maintained.

Table 15-2 Hardware States after Reset (2/2)

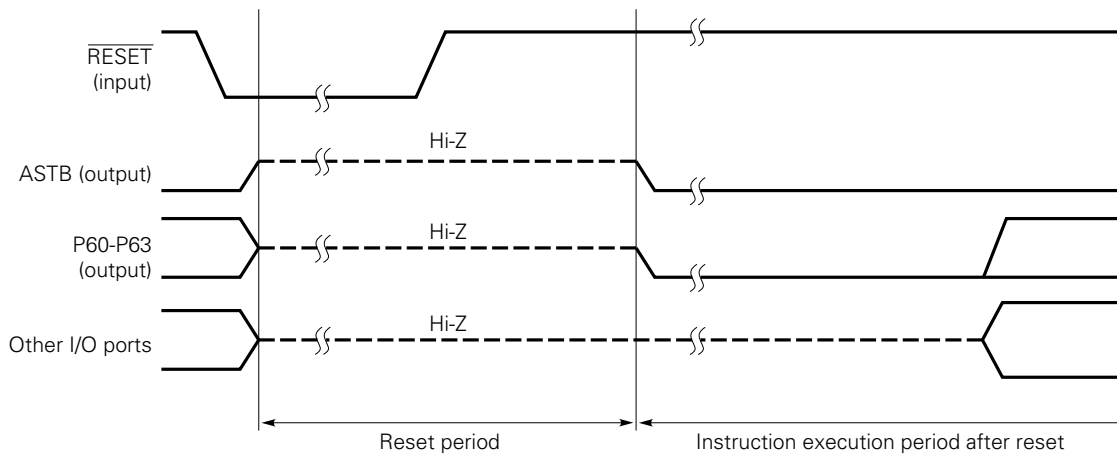
Hardware		State after reset
Serial interface	Mode register (CSIM)	00H
	Shift register (SIO)	Undefined
	Asynchronous mode register (ASIM)	80H
	Asynchronous status register (ASIS)	00H
	Serial bus control register (SBIC)	00H
	Serial reception buffer (RXB)	Undefined
	Serial transmission buffer (TXS)	Undefined
	Baud rate generator control register (BRGC)	00H
Real-time output port control register (RTPC)		00H
Programmable wait control register (PW)		80H
Refresh mode register (RFM)		00H
Interrupts	Interrupt request flag register (IF0)	0000H
	Interrupt mask register (MK0)	FFFFH
	Priority designation flag register (PR0)	FFFFH
	Interrupt service mode register (ISM0)	0000H
	Interrupt status register (IST)	00H
External interrupt mode registers (INTM0 and INTM1)		00H
Standby control register (STBC)		0000 × 000B

Fig. 15-3 Timing Charts for Reset Operation

(a) For μ PD78213



(b) For μ PD78214



15.2 NOTE

When resetting the system at power-on, do not set the RESET signal high immediately after the supply voltage reaches the specified level. Keep the signal low until oscillation has settled.

CHAPTER 16 APPLICATION EXAMPLES

16.1 OPEN-LOOP CONTROL OF STEPPER MOTORS

This section provides an example of controlling stepper motors with the real-time output function, 8-bit timer/counter 1, and the macro service function of the μ PD78214.

Fig. 16-1 shows the functional blocks for controlling two stepper motors.

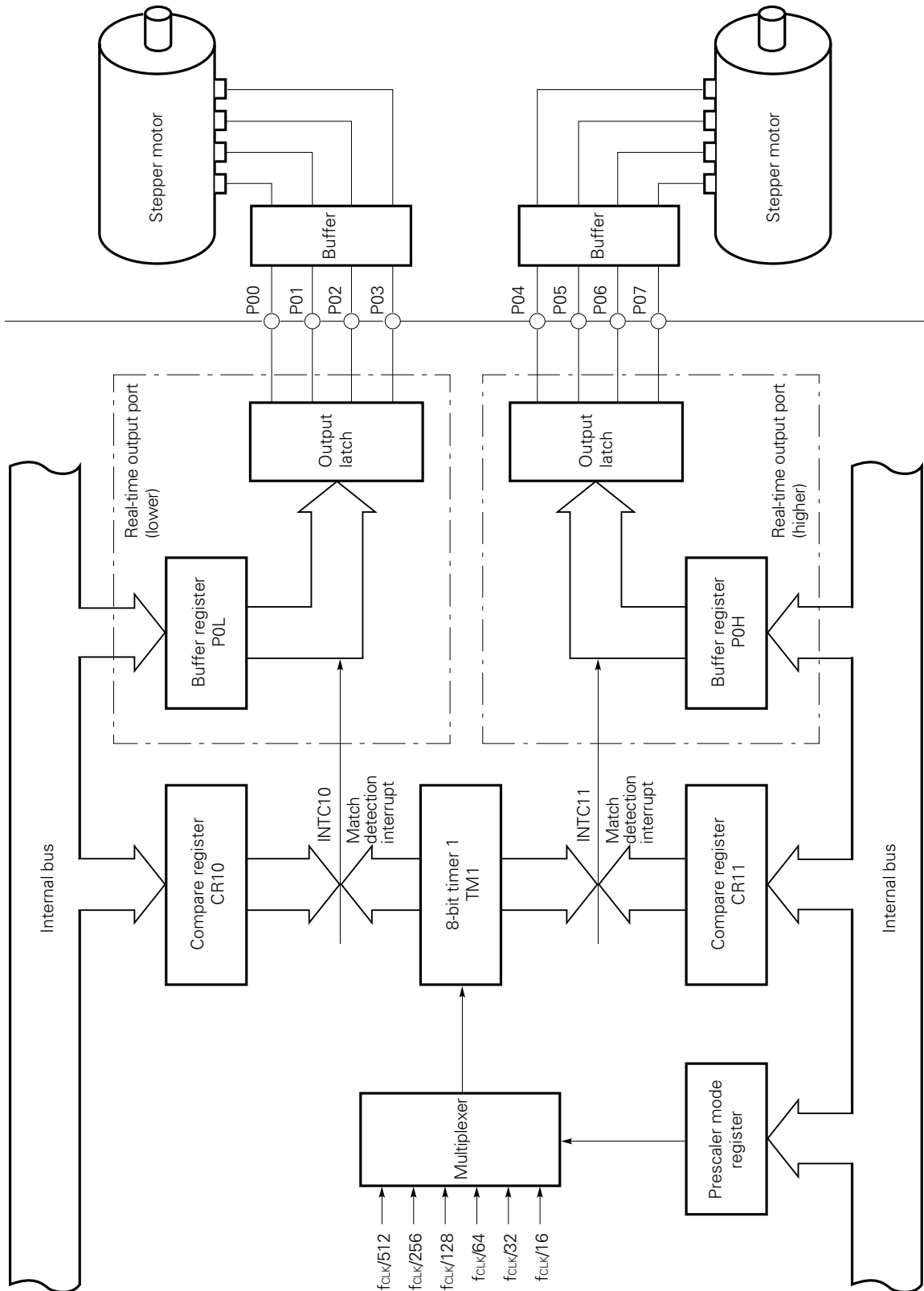
An interrupt signal is generated when the value in 8-bit timer/counter 1 (TM1) is matched with the value in compare register CR10 or CR11. The interrupt signal triggers a macro service in which the CPU automatically transfers table data prepared in memory to the compare register and the buffer register for the real-time output port. Data transferred to the compare register is used for the interval before the generation of the next interrupt.

Data transferred to the buffer register is output from port 0.

Open-loop control of stepper motors with the μ PD78214 has the following advantages:

- (1) The real-time output function provides accurate control (output) signals based on a specified interval.
- (2) Two stepper motors can be independently controlled with an 8-bit timer/counter and two compare registers. Hardware can thus be efficiently used.
- (3) The macro service function can perform complicated control, such as the acceleration/deceleration of stepper motors, without the need for software.

Fig. 16-1 Example of Controlling Two Stepper Motors



16.2 SERIAL COMMUNICATION WITH MULTIPLE DEVICES

Fig. 16-2 shows an example of a system configured with a serial bus interface. The serial bus interface can transfer addresses (for selecting devices), commands, and data, as well as acknowledge and busy signals, using only two lines: The serial clock and serial bus lines.

When a master device communicates with multiple slave devices, the master device outputs an address for selecting a slave device on the serial bus line. Each slave device checks whether the received address is the same as the address assigned to the device, using software. Only a slave device whose address is the same as the received address returns an acknowledge signal to the master device, after which it receives commands from the master device or transfers data to and from the master device. Fig. 16-3 shows an example of communication using the serial bus interface (SBI).

Fig. 16-2 Example System Configuration Using the Serial Bus Interface

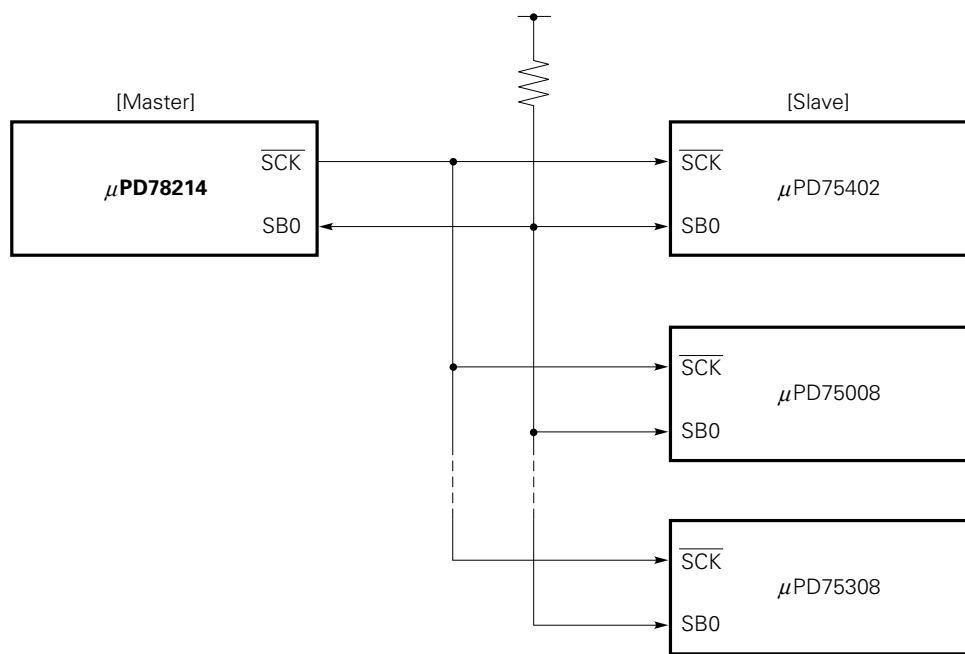


Fig. 16-3 Example of Communication with SBI

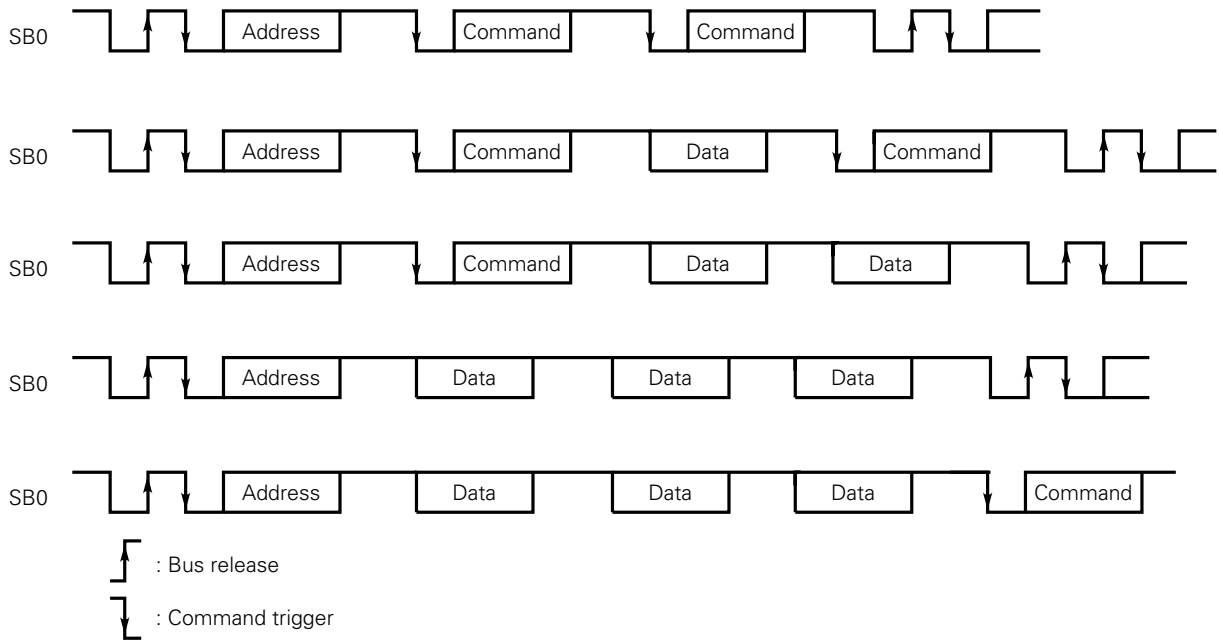
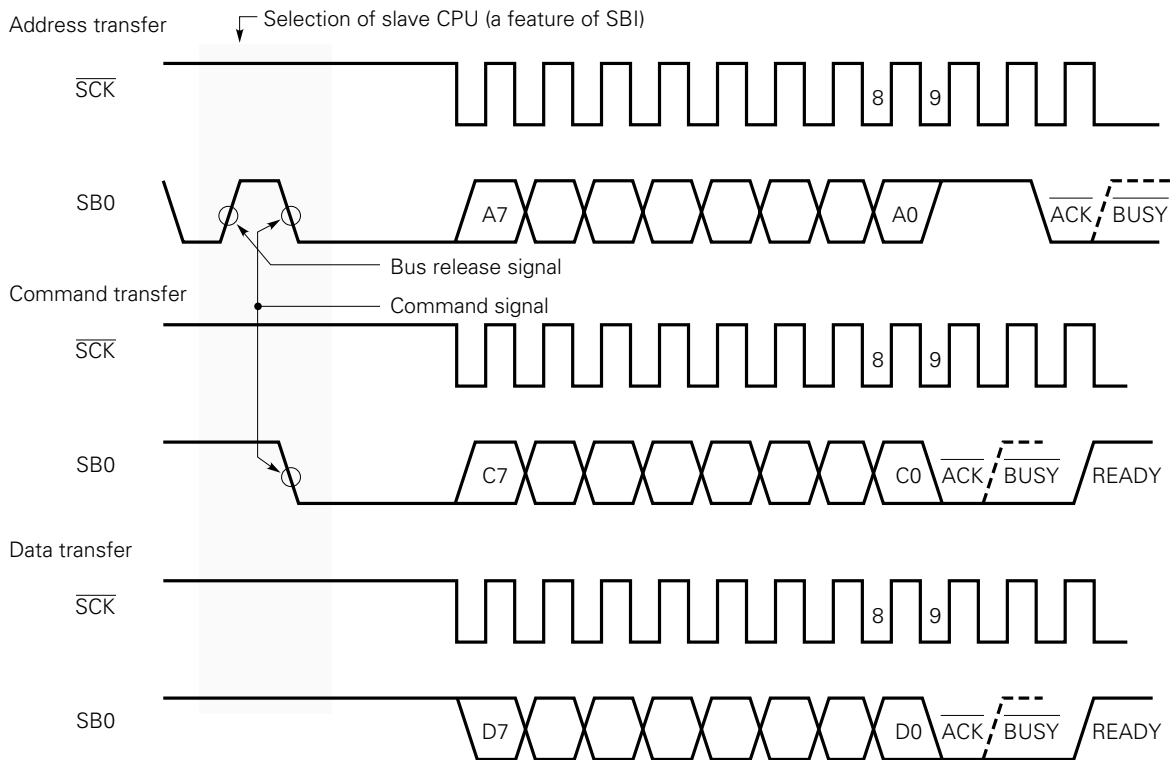


Fig. 16-4 Serial Bus Communication Timing



CHAPTER 17 PROGRAMMING FOR THE μ PD78P214

The μ PD78P214 employs an electrically writable PROM of 16384×8 bits for program memory. Use the NMI and $\overline{\text{RESET}}$ pins to set the μ PD78P214 to PROM programming mode when programming the PROM.

The μ PD78P214 provides programming characteristics compatible with the μ PD27C256A^{Note}.

Note 100 μ s program pulses are not supported.

★

17.1 OPERATING MODE

When +6 V is applied to the V_{DD} pin and +12.5 V to the V_{PP} pin, the μ PD78P214 enters PROM programming mode. This mode can be changed to each of the operating modes shown in Table 17-1 according to the settings of the $\overline{\text{CE}}$ and $\overline{\text{OE}}$ pins.

Setting the μ PD78P214 to read mode enables it to read the contents of PROM.

Table 17-1 Operating Modes for PROM Programming

Mode \ Pin	NMI	$\overline{\text{RESET}}$	V_{PP}	V_{DD}	$\overline{\text{CE}}$	$\overline{\text{OE}}$	D0-D7
Program write	+12.5 V	L	+12.5 V	+6 V	L	H	Data input
Program verify					H	L	Data output
Program inhibit					H	H	High-impedance
Read	+12.5 V	L	+5 V	+5 V	L	L	Data output
Output disable					L	H	High-impedance
Standby					H	L/H	High-impedance

Caution When V_{PP} is +12.5 V and V_{DD} is +6 V, $\overline{\text{CE}}$ and $\overline{\text{OE}}$ must not be set to low at the same time.

17.2 PROCEDURE FOR WRITING INTO PROM

Data can be written into PROM at high speed by following the procedure below:

- (1) Fix the $\overline{\text{RESET}}$ pin to the low level. Apply +12.5 V to pin NMI. Handle unused pins as described in **Section 1.3.2**.
- (2) Apply +6 V to the V_{DD} pin and +12.5 V to the V_{PP} pin.
- (3) Input an initial address.
- (4) Input the write data.
- (5) Input a program pulse (active low), having a period of 1 ms, to the $\overline{\text{CE}}$ pin.
- (6) Check that data has been written into the PROM (verify mode). When the data has been written correctly, go to step (8). Otherwise, repeat steps (4) to (6). If data has still not been written successfully after repeating this part of the procedure 25 times, go to step (7).
- (7) Assume the device to be defective and abandon the write operation.
- (8) Input the write data, then input a program pulse which has a period of (number of times steps (4) to (6) have been repeated) \times 3 ms (additional write).
- (9) Increment the address.
- (10) Repeat steps (4) to (9) until the address exceeds the previous address.

Fig. 17-1 is the timing chart for steps (2) to (8) above.

Fig. 17-1 Timing Chart for PROM Write and Verify

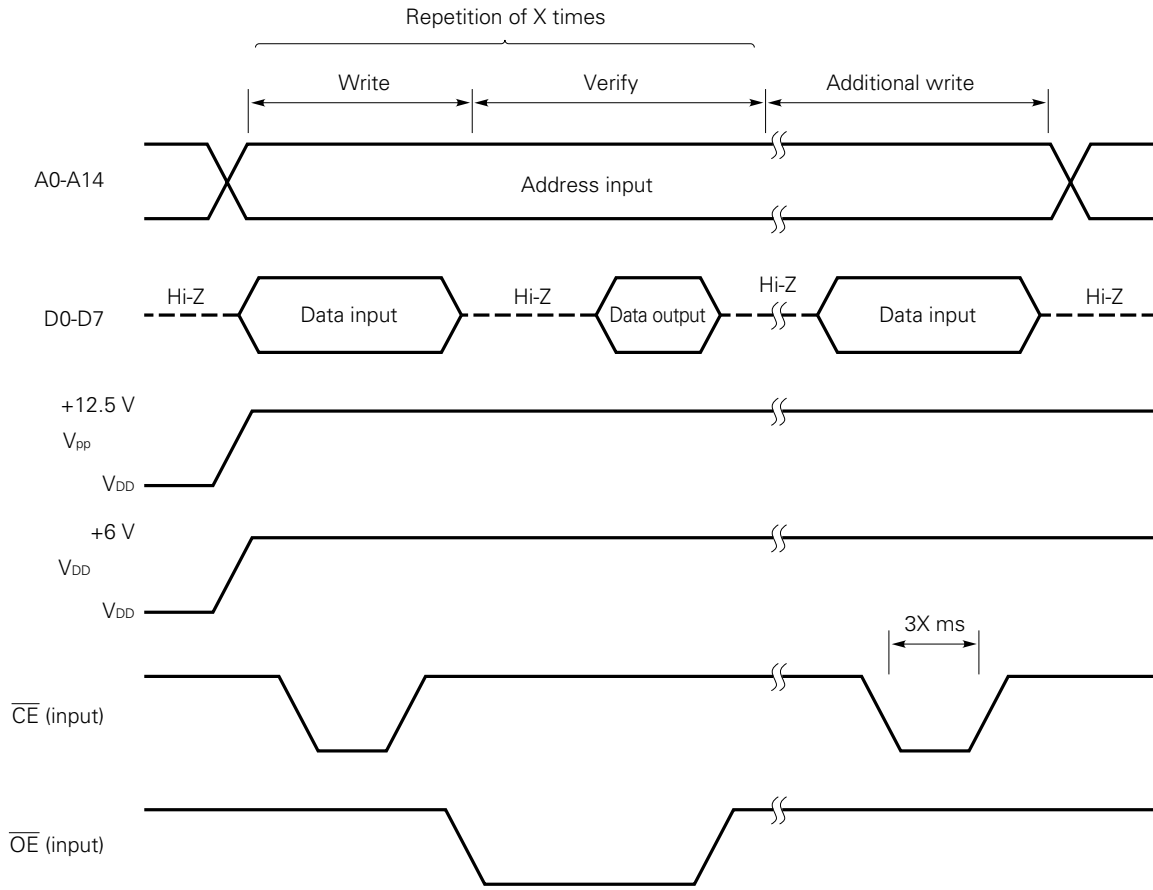
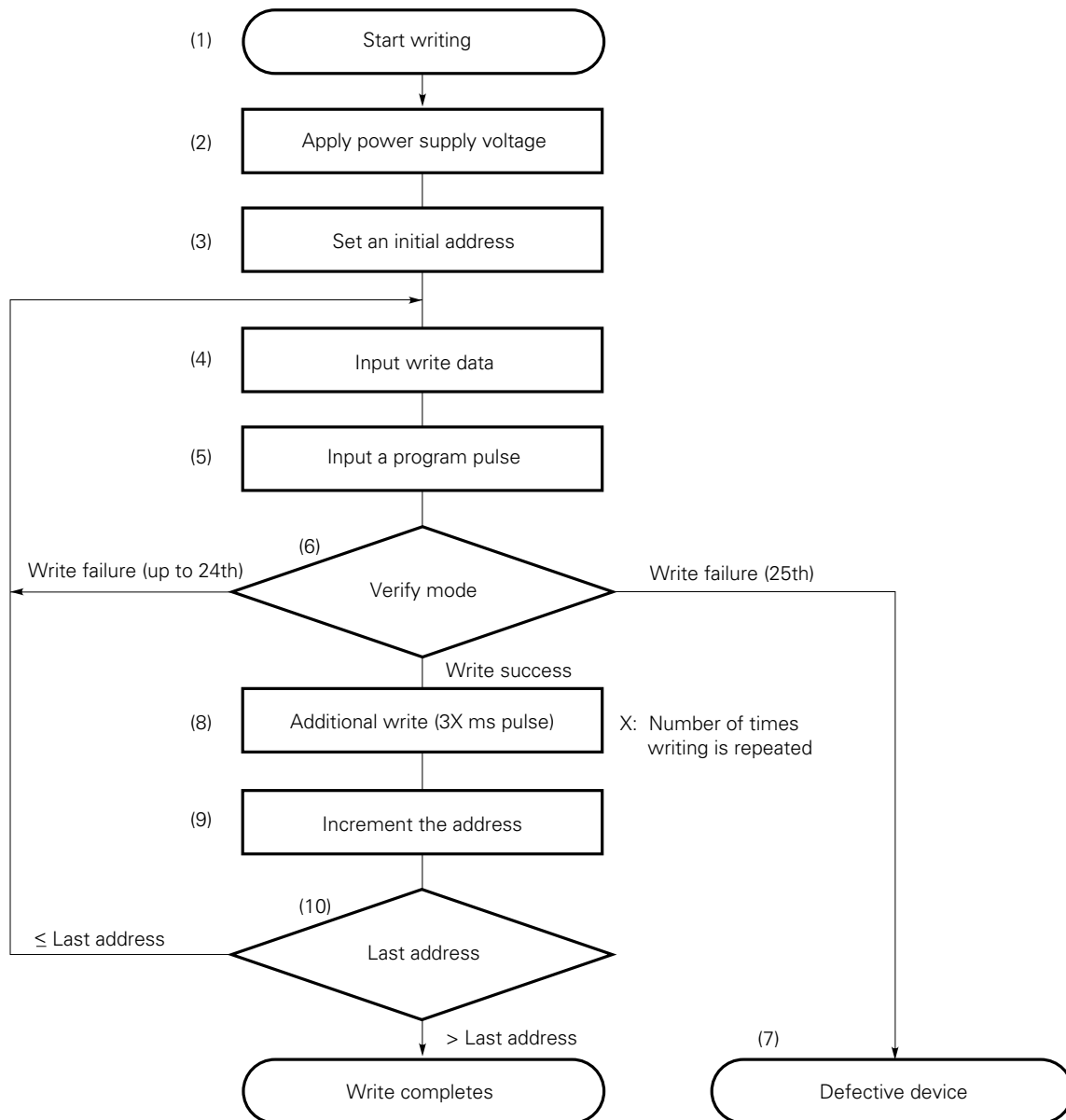


Fig. 17-2 Write Operation Flowchart

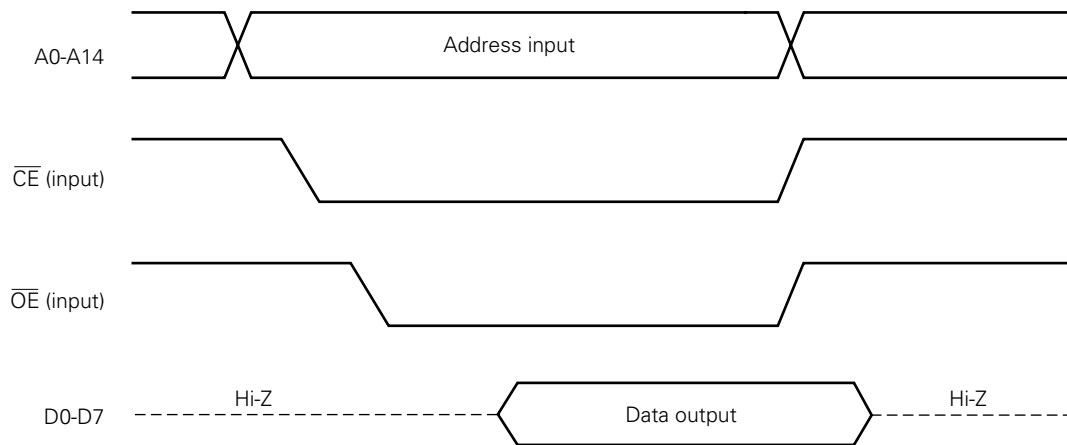
17.3 PROCEDURE FOR READING FROM PROM

The contents of PROM can be read out to the external data bus (D0 to D7) by following the procedure below:

- (1) Fix the $\overline{\text{RESET}}$ pin to the low level. Apply +12.5 V to pin NMI. Handle unused pins as described in **Section 1.3.2**.
- (2) Apply +5 V to the V_{DD} and V_{PP} pins.
- (3) Input the address of the data to be read into the A0 to A14 pins.
- (4) Set read mode.
- (5) Output the data on the D0 to D7 pins.

Fig. 17-3 is the timing chart for steps (2) to (5).

Fig. 17-3 PROM Read Timing Chart



17.4 NOTE

When V_{PP} is +12.5 V and V_{DD} is +6 V, \overline{CE} and \overline{OE} must not be set to low at the same time.

CHAPTER 18 INSTRUCTION OPERATIONS

This chapter describes the operation of each instruction of the μ PD78214 sub-series. Refer to the **78K/II Series User's Manual, Instructions (IEU-1311)** for details of each operation, the corresponding machine language code (instruction code), and the number of clock states for each instruction.

18.1 LEGEND

18.1.1 Operand Field

Code operands in the operand field for each instruction, using the specified operand representation format (for details, refer to the relevant assembler specifications). When several coding forms are presented, any one can be used. Since uppercase letters and symbols +, -, #, !, \$, /, [], and & are keywords, write any symbols as is.

Do not omit symbols +, -, #, !, \$, /, [], and & when writing immediate data with labels. r and rp can be written with any functional and absolute names.

+	: Auto increment
-	: Auto decrement
#	: Immediate data
!	: Absolute addressing
\$: Relative addressing
/	: Bit inversion
[]	: Indirect addressing
&	: Sub-bank specification
r,r'	: Registers; Functional name: X, A, C, B, E, D, L, H Absolute name : R0-R7
r1	: Register group 1; B, C
rp, rp'	: Register pairs; Functional name: AX, BC, DE, HL Absolute name : RP0-RP3
sfr	: Special function registers; P0, P2, P3, P4, P5, P6, P7, P0H, P0L, RTPC, CR10, CR11, CR20, CR21, CR22, CR30, PM0, PM3, PM5, PM6, PMC3, PUO, CRC0, CRC1, CRC2, TOC, TM1, TM2, TM3, TMC0, TMC1, PRM0, PRM1, ADM, ADCR, CSIM, SBIC, SIO, ASIM, ASIS, RXB, TXS, BRGC, STBC (only for dedicated instruction), MM, PW, RFM, IF0L, IF0H, MK0L, MK0H, PR0L, PR0H, ISM0L, ISM0H, INTM0, INTM1, IST
sfrp	: Special function register pairs; CR00, CR01, CR02, TM0, IF0, MK0, PR0, ISM0
mem	: Memory address indicated in indirect addressing mode; Register indirect mode : [DE], [HL], [DE+], [HL+], [DE-], [HL-] Base mode : [DE+byte], [HL+byte], [SP+byte] Indexed mode : word[A], word[B], word[DE], word[HL]
mem1	: Memory address indicated in indirect addressing group 1 mode; [DE], [HL]

saddr, saddr'	: Memory address indicated in short direct addressing mode; FE20H-FF1FH immediate data or label
saddrp	: Memory address indicated in short direct addressing pair mode; FE20H-FF1EH immediate data or label
addr16	: 16-bit address; 0000H-FEFFFH immediate data or label
addr11	: 11-bit address; 800H-FFFH immediate data or label
addr5	: 5-bit address; 40H-7EH immediate data or label
word	: 16-bit data; 16-bit immediate data or label
byte	: 8-bit data; 8-bit immediate data or label
bit	: 3-bit data; 3-bit immediate data or label
n	: Number of shift bits; 3-bit immediate data (0-7)
RBn	: Register bank; RB0-RB3

18.1.2 Operation Field

A	: Register A; 8-bit accumulator
X	: Register X
B	: Register B
C	: Register C
D	: Register D
E	: Register E
H	: Register H
L	: Register L
R0-R7	: Register 0 to register 7 (absolute name)
AX	: Register pair (AX); 16-bit accumulator
BC	: Register pair (BC)
DE	: Register pair (DE)
HL	: Register pair (HL)
RP0-RP3	: Register pair 0 to register pair 3 (absolute name)
PC	: Program counter
SP	: Stack pointer
PSW	: Program status word
CY	: Carry flag
AC	: Auxiliary carry flag

- Z : Zero flag
- RBS1-RBS0 : Register bank selection flag
- IE : Interrupt request enable flag
- STBC : Standby control register
- jdisp8 : Signed 8-bit data (displacement: -128 to +127)
- () : Contents at address enclosed in parentheses or at address indicated in register enclosed in parentheses
- xxH : Hexadecimal number
- x_H, x_L : Eight high-order bits and eight low-order bits of 16-bit register pair

18.1.3 Flag Field

- Blank : No change
- 0 : Cleared to zero.
- 1 : Set to 1.
- × : Set or cleared according to the result.
- R : Saved values are restored.

18.2 LIST OF OPERATIONS

(1) 8-bit data transfer instructions: MOV, XCH

Mnemonic	Operand	No. of bytes	Operation	Flags		
				Z	AC	CY
MOV	r, #byte	2	$r \leftarrow \text{byte}$			
	saddr, #byte	3	$(\text{saddr}) \leftarrow \text{byte}$			
	sfr, #byte	3	$\text{sfr} \leftarrow \text{byte}$			
	r, r'	2	$r \leftarrow r'$			
	A, r	1	$A \leftarrow r$			
	A, saddr	2	$A \leftarrow (\text{saddr})$			
	saddr, A	2	$(\text{saddr}) \leftarrow A$			
	saddr, saddr	3	$(\text{saddr}) \leftarrow (\text{saddr})$			
	A, sfr	2	$A \leftarrow \text{sfr}$			
	sfr, A	2	$\text{sfr} \leftarrow A$			
	A, mem	1-4	$A \leftarrow (\text{mem})$			
	A, & mem	2-5	$A \leftarrow (\& \text{mem})$			
	mem, A	1-4	$(\text{mem}) \leftarrow A$			
	& mem, A	2-5	$(\& \text{mem}) \leftarrow A$			
	A, !addr16	4	$A \leftarrow (!\text{addr}16)$			
	A, & !addr16	5	$A \leftarrow (\& !\text{addr}16)$			
	!addr16, A	4	$(!\text{addr}16) \leftarrow A$			
	& !addr16, A	5	$(\& !\text{addr}16) \leftarrow A$			
	PSW, #byte	3	$\text{PSW} \leftarrow \text{byte}$	×	×	×
	PSW, A	2	$\text{PSW} \leftarrow A$	×	×	×
A, PSW	2	$A \leftarrow \text{PSW}$				
XCH	A, r	1	$A \leftrightarrow r$			
	r, r'	2	$r \leftrightarrow r'$			
	A, mem	2-4	$A \leftrightarrow (\text{mem})$			
	A, & mem	3-5	$A \leftrightarrow (\& \text{mem})$			
	A, saddr	2	$A \leftrightarrow (\text{saddr})$			
	A, sfr	3	$A \leftrightarrow \text{sfr}$			
	saddr, saddr'	3	$(\text{saddr}) \leftrightarrow (\text{saddr}')$			

(2) 16-bit data transfer instructions: MOVW

Mnemonic	Operand	No. of bytes	Operation	Flags		
				Z	AC	CY
MOVW	rp, #word	3	rp ← word			
	saddrp, #word	4	(saddrp) ← word			
	sfrp, #word	4	sfrp ← word			
	rp, rp'	2	rp ← rp'			
	AX, saddrp	2	AX ← (saddrp)			
	saddrp, AX	2	(saddrp) ← AX			
	AX, sfrp	2	AX ← sfrp			
	sfrp, AX	2	sfrp ← AX			
	AX, mem1	2	AX ← (mem1)			
	AX, & mem1	3	AX ← (& mem1)			
	mem1, AX	2	(mem1) ← AX			
	& mem1, AX	3	(& mem1) ← AX			

(3) 8-bit arithmetic/logical instructions: ADD, ADDC, SUB, SUBC, AND, OR, XOR, CMP

Mnemonic	Operand	No. of bytes	Operation	Flags		
				Z	AC	CY
ADD	A, #byte	2	A, CY ← A + byte	×	×	×
	saddr, #byte	3	(saddr), CY ← (saddr) + byte	×	×	×
	sfr, #byte	4	sfr, CY ← sfr + byte	×	×	×
	r, r'	2	r, CY ← r + r'	×	×	×
	A, saddr	2	A, CY ← A + (saddr)	×	×	×
	A, sfr	3	A, CY ← A + sfr	×	×	×
	saddr, saddr'	3	(saddr), CY ← (saddr) + (saddr')	×	×	×
	A, mem	2-4	A, CY ← A + (mem)	×	×	×
	A, & mem	3-5	A, CY ← A + (& mem)	×	×	×
ADDC	A, #byte	2	A, CY ← A + byte + CY	×	×	×
	saddr, #byte	3	(saddr), CY ← (saddr) + byte + CY	×	×	×
	sfr, #byte	4	sfr, CY ← sfr + byte + CY	×	×	×
	r, r'	2	r, CY ← r + r' + CY	×	×	×
	A, saddr	2	A, CY ← A + (saddr) + CY	×	×	×
	A, sfr	3	A, CY ← A + sfr + CY	×	×	×
	saddr, saddr'	3	(saddr), CY ← (saddr) + (saddr') + CY	×	×	×
	A, mem	2-4	A, CY ← A + (mem) + CY	×	×	×
	A, & mem	3-5	A, CY ← A + (& mem) + CY	×	×	×

(Continued)

Mnemonic	Operand	No. of bytes	Operation	Flags		
				Z	AC	CY
SUB	A, #byte	2	A, CY ← A – byte	×	×	×
	saddr, #byte	3	(saddr), CY ← (saddr) – byte	×	×	×
	sfr, #byte	4	sfr, CY ← sfr – byte	×	×	×
	r, r'	2	r, CY ← r – r'	×	×	×
	A, saddr	2	A, CY ← A – (saddr)	×	×	×
	A, sfr	3	A, CY ← A – sfr	×	×	×
	saddr, saddr'	3	(saddr), CY ← (saddr) – (saddr')	×	×	×
	A, mem	2-4	A, CY ← A – (mem)	×	×	×
	A, & mem	3-5	A, CY ← A – (& mem)	×	×	×
SUBC	A, #byte	2	A, CY ← A – byte – CY	×	×	×
	saddr, #byte	3	(saddr), CY ← (saddr) – byte – CY	×	×	×
	sfr, #byte	4	sfr, CY ← sfr – byte – CY	×	×	×
	r, r'	2	r, CY ← r – r' – CY	×	×	×
	A, saddr	2	A, CY ← A – (saddr) – CY	×	×	×
	A, sfr	3	A, CY ← A – sfr – CY	×	×	×
	saddr, saddr'	3	(saddr), CY ← (saddr) – (saddr') – CY	×	×	×
	A, mem	2-4	A, CY ← A – (mem) – CY	×	×	×
	A, & mem	3-5	A, CY ← A – (& mem) – CY	×	×	×
AND	A, #byte	2	A ← A ∧ byte	×		
	saddr, #byte	3	(saddr) ← (saddr) ∧ byte	×		
	sfr, #byte	4	sfr ← sfr ∧ byte	×		
	r, r'	2	r ← r ∧ r'	×		
	A, saddr	2	A ← A ∧ (saddr)	×		
	A, sfr	3	A ← A ∧ sfr	×		
	saddr, saddr'	3	(saddr) ← (saddr) ∧ (saddr')	×		
	A, mem	2-4	A ← A ∧ (mem)	×		
	A, & mem	3-5	A ← A ∧ (& mem)	×		
OR	A, #byte	2	A ← A ∨ byte	×		
	saddr, #byte	3	(saddr) ← (saddr) ∨ byte	×		
	sfr, #byte	4	sfr ← sfr ∨ byte	×		
	r, r'	2	r ← r ∨ r'	×		
	A, saddr	2	A ← A ∨ (saddr)	×		
	A, sfr	3	A ← A ∨ sfr	×		
	saddr, saddr'	3	(saddr) ← (saddr) ∨ (saddr')	×		
	A, mem	2-4	A ← A ∨ (mem)	×		
	A, & mem	3-5	A ← A ∨ (& mem)	×		

(Continued)

Mnemonic	Operand	No. of bytes	Operation	Flags		
				Z	AC	CY
XOR	A, #byte	2	$A \leftarrow A \vee \text{byte}$	×		
	saddr, #byte	3	$(\text{saddr}) \leftarrow (\text{saddr}) \vee \text{byte}$	×		
	sfr, #byte	4	$\text{sfr} \leftarrow \text{sfr} \vee \text{byte}$	×		
	r, r'	2	$r \leftarrow r \vee r'$	×		
	A, saddr	2	$A \leftarrow A \vee (\text{saddr})$	×		
	A, sfr	3	$A \leftarrow A \vee \text{sfr}$	×		
	saddr, saddr'	3	$(\text{saddr}) \leftarrow (\text{saddr}) \vee (\text{saddr}')$	×		
	A, mem	2-4	$A \leftarrow A \vee (\text{mem})$	×		
	A, & mem	3-5	$A \leftarrow A \vee (\& \text{mem})$	×		
CMP	A, #byte	2	$A - \text{byte}$	×	×	×
	saddr, #byte	3	$(\text{saddr}) - \text{byte}$	×	×	×
	sfr, #byte	4	$\text{sfr} - \text{byte}$	×	×	×
	r, r'	2	$r - r'$	×	×	×
	A, saddr	2	$A - (\text{saddr})$	×	×	×
	A, sfr	3	$A - \text{sfr}$	×	×	×
	saddr, saddr'	3	$(\text{saddr}) - (\text{saddr}')$	×	×	×
	A, mem	2-4	$A - (\text{mem})$	×	×	×
	A, & mem	3-5	$A - (\& \text{mem})$	×	×	×

(4) 16-bit arithmetic/logical instructions: ADDW, SUBW, CMPW

Mnemonic	Operand	No. of bytes	Operation	Flags		
				Z	AC	CY
ADDW	AX, #word	3	$AX, CY \leftarrow AX + \text{word}$	×	×	×
	AX, rp	2	$AX, CY \leftarrow AX + rp$	×	×	×
	AX, saddrp	2	$AX, CY \leftarrow AX + (\text{saddrp})$	×	×	×
	AX, sfrp	3	$AX, CY \leftarrow AX + \text{sfrp}$	×	×	×
SUBW	AX, #word	3	$AX, CY \leftarrow AX - \text{word}$	×	×	×
	AX, rp	2	$AX, CY \leftarrow AX - rp$	×	×	×
	AX, saddrp	2	$AX, CY \leftarrow AX - (\text{saddrp})$	×	×	×
	AX, sfrp	3	$AX, CY \leftarrow AX - \text{sfrp}$	×	×	×
CMPW	AX, #word	3	$AX - \text{word}$	×	×	×
	AX, rp	2	$AX - rp$	×	×	×
	AX, saddrp	2	$AX - (\text{saddrp})$	×	×	×
	AX, sfrp	3	$AX - \text{sfrp}$	×	×	×

(5) Multiply/divide instructions: MULU, DIVUW

Mnemonic	Operand	No. of bytes	Operation	Flags		
				Z	AC	CY
MULU	r	2	$AX \leftarrow A \times r1$			
DIVUW	r	2	AX (quotient), r (remainder) $\leftarrow AX \div r$ When $r = 0$, $r \leftarrow X$, $AX \leftarrow 0FFFFH$			

(6) Increment/decrement instructions: INC, DEC, INCW, DECW

Mnemonic	Operand	No. of bytes	Operation	Flags		
				Z	AC	CY
INC	r	1	$r \leftarrow r + 1$	×	×	
	saddr	2	$(saddr) \leftarrow (saddr) + 1$	×	×	
DEC	r	1	$r \leftarrow r - 1$	×	×	
	saddr	2	$(saddr) \leftarrow (saddr) - 1$	×	×	
INCW	rp	1	$rp \leftarrow rp + 1$			
DECW	rp	1	$rp \leftarrow rp - 1$			

(7) Shift/rotate instructions: ROR, ROL, RORC, ROLC, SHR, SHL, SHRW, SHLW, ROR4, ROL4

Mnemonic	Operand	No. of bytes	Operation	Flags		
				Z	AC	CY
ROR	r, n	2	$(CY, r_7 \leftarrow r_0, r_{m-1} \leftarrow r_m) \times n$ times $n=0$ to 7			×
ROL	r, n	2	$(CY, r_0 \leftarrow r_7, r_{m+1} \leftarrow r_m) \times n$ times $n=0$ to 7			×
RORC	r, n	2	$(CY \leftarrow r_0, r_7 \leftarrow CY, r_{m-1} \leftarrow r_m) \times n$ times $n=0$ to 7			×
ROLC	r, n	2	$(CY \leftarrow r_7, r_0 \leftarrow CY, r_{m+1} \leftarrow r_m) \times n$ times $n=0$ to 7			×
SHR	r, n	2	$(CY \leftarrow r_0, r_7 \leftarrow 0, r_{m-1} \leftarrow r_m) \times n$ times $n=0$ to 7	×	0	×
SHL	r, n	2	$(CY \leftarrow r_7, r_0 \leftarrow 0, r_{m+1} \leftarrow r_m) \times n$ times $n=0$ to 7	×	0	×
SHRW	rp, n	2	$(CY \leftarrow rp_0, rp_{15} \leftarrow 0, rp_{m-1} \leftarrow rp_m) \times n$ times $n=0$ to 7	×	0	×
SHLW	rp, n	2	$(CY \leftarrow rp_{15}, rp_0 \leftarrow 0, rp_{m+1} \leftarrow rp_m) \times n$ times $n=0$ to 7	×	0	×
ROR4	mem1	2	$A_{3-0} \leftarrow (mem1)_{3-0}, (mem1)_{7-4} \leftarrow A_{3-0},$ $(mem1)_{3-0} \leftarrow (mem1)_{7-4}$			
	& mem1	3	$A_{3-0} \leftarrow (& mem1)_{3-0}, (& mem1)_{7-4} \leftarrow A_{3-0},$ $(& mem1)_{3-0} \leftarrow (& mem1)_{7-4}$			
ROL4	mem1	2	$A_{3-0} \leftarrow (mem1)_{7-4}, (mem1)_{3-0} \leftarrow A_{3-0},$ $(mem1)_{7-4} \leftarrow (mem1)_{3-0}$			
	& mem1	3	$A_{3-0} \leftarrow (& mem1)_{7-4}, (& mem1)_{3-0} \leftarrow A_{3-0},$ $(& mem1)_{7-4} \leftarrow (& mem1)_{3-0}$			

(8) BCD conversion instructions: **ADJBA, ADJBS**

Mnemonic	Operand	No. of bytes	Operation	Flags		
				Z	AC	CY
ADJBA		1	Use the decimal adjust accumulator after addition.	×	×	×
ADJBS		1	Use the decimal adjust accumulator after subtraction.	×	×	×

(9) Bit manipulation instructions: **MOV1, AND1, OR1, XOR1, SET1, CLR1, NOT1**

Mnemonic	Operand	No. of bytes	Operation	Flags		
				Z	AC	CY
MOV1	CY, saddr.bit	3	$CY \leftarrow (\text{saddr.bit})$			×
	CY, sfr.bit	3	$CY \leftarrow \text{sfr.bit}$			×
	CY, A.bit	2	$CY \leftarrow \text{A.bit}$			×
	CY, X.bit	2	$CY \leftarrow \text{X.bit}$			×
	CY, PSW.bit	2	$CY \leftarrow \text{PSW.bit}$			×
	saddr.bit, CY	3	$(\text{saddr.bit}) \leftarrow CY$			
	sfr.bit, CY	3	$\text{sfr.bit} \leftarrow CY$			
	A.bit, CY	2	$\text{A.bit} \leftarrow CY$			
	X.bit, CY	2	$\text{X.bit} \leftarrow CY$			
	PSW.bit, CY	2	$\text{PSW.bit} \leftarrow CY$	×	×	
AND1	CY, saddr.bit	3	$CY \leftarrow CY \wedge (\text{saddr.bit})$			×
	CY, /saddr.bit	3	$CY \leftarrow CY \wedge \overline{(\text{saddr.bit})}$			×
	CY, sfr.bit	3	$CY \leftarrow CY \wedge \text{sfr.bit}$			×
	CY, /sfr.bit	3	$CY \leftarrow CY \wedge \overline{\text{sfr.bit}}$			×
	CY, A.bit	2	$CY \leftarrow CY \wedge \text{A.bit}$			×
	CY, /A.bit	2	$CY \leftarrow CY \wedge \overline{\text{A.bit}}$			×
	CY, X.bit	2	$CY \leftarrow CY \wedge \text{X.bit}$			×
	CY, /X.bit	2	$CY \leftarrow CY \wedge \overline{\text{X.bit}}$			×
	CY, PSW.bit	2	$CY \leftarrow CY \wedge \text{PSW.bit}$			×
	CY, /PSW.bit	2	$CY \leftarrow CY \wedge \overline{\text{PSW.bit}}$			×
OR1	CY, saddr.bit	3	$CY \leftarrow CY \vee (\text{saddr.bit})$			×
	CY, /saddr.bit	3	$CY \leftarrow CY \vee \overline{(\text{saddr.bit})}$			×
	CY, sfr.bit	3	$CY \leftarrow CY \vee \text{sfr.bit}$			×
	CY, /sfr.bit	3	$CY \leftarrow CY \vee \overline{\text{sfr.bit}}$			×
	CY, A.bit	2	$CY \leftarrow CY \vee \text{A.bit}$			×
	CY, /A.bit	2	$CY \leftarrow CY \vee \overline{\text{A.bit}}$			×
	CY, X.bit	2	$CY \leftarrow CY \vee \text{X.bit}$			×
	CY, /X.bit	2	$CY \leftarrow CY \vee \overline{\text{X.bit}}$			×
	CY, PSW.bit	2	$CY \leftarrow CY \vee \text{PSW.bit}$			×
	CY, /PSW.bit	2	$CY \leftarrow CY \vee \overline{\text{PSW.bit}}$			×

(Continued)

Mnemonic	Operand	No. of bytes	Operation	Flags		
				Z	AC	CY
XOR1	CY, saddr.bit	3	$CY \leftarrow CY \vee (\text{saddr.bit})$			×
	CY, sfr.bit	3	$CY \leftarrow CY \vee \text{sfr.bit}$			×
	CY, A.bit	2	$CY \leftarrow CY \vee \text{A.bit}$			×
	CY, X.bit	2	$CY \leftarrow CY \vee \text{X.bit}$			×
	CY, PSW.bit	2	$CY \leftarrow CY \vee \text{PSW.bit}$			×
SET1	saddr.bit	2	$(\text{saddr.bit}) \leftarrow 1$			
	sfr.bit	3	$\text{sfr.bit} \leftarrow 1$			
	A.bit	2	$\text{A.bit} \leftarrow 1$			
	X.bit	2	$\text{X.bit} \leftarrow 1$			
	PSW.bit	2	$\text{PSW.bit} \leftarrow 1$	×	×	×
	CY	1	$CY \leftarrow 1$			1
CLR1	saddr.bit	2	$(\text{saddr.bit}) \leftarrow 0$			
	sfr.bit	3	$\text{sfr.bit} \leftarrow 0$			
	A.bit	2	$\text{A.bit} \leftarrow 0$			
	X.bit	2	$\text{X.bit} \leftarrow 0$			
	PSW.bit	2	$\text{PSW.bit} \leftarrow 0$	×	×	×
	CY	1	$CY \leftarrow 0$			0
NOT1	saddr.bit	3	$(\text{saddr.bit}) \leftarrow \overline{(\text{saddr.bit})}$			
	sfr.bit	3	$\text{sfr.bit} \leftarrow \overline{\text{sfr.bit}}$			
	A.bit	2	$\text{A.bit} \leftarrow \overline{\text{A.bit}}$			
	X.bit	2	$\text{X.bit} \leftarrow \overline{\text{X.bit}}$			
	PSW.bit	2	$\text{PSW.bit} \leftarrow \overline{\text{PSW.bit}}$	×	×	×
	CY	1	$CY \leftarrow \overline{CY}$			×

(10) Call/return instructions: **CALL, CALLF, CALLT, BRK, RET, RETI, RETB**

Mnemonic	Operand	No. of bytes	Operation	Flags		
				Z	AC	CY
CALL	!addr16	3	$(SP - 1) \leftarrow (PC + 3)_H, (SP - 2) \leftarrow (PC + 3)_L,$ $PC \leftarrow \text{addr16}, SP \leftarrow SP - 2$			
	rp	2	$(SP - 1) \leftarrow (PC + 2)_H, (SP - 2) \leftarrow (PC + 2)_L,$ $PC_H \leftarrow rp_H, PC_L \leftarrow rp_L, SP \leftarrow SP - 2$			
CALLF	!addr11	2	$(SP - 1) \leftarrow (PC + 2)_H, (SP - 2) \leftarrow (PC + 2)_L,$ $PC_{15-11} \leftarrow 00001, PC_{10-0} \leftarrow \text{addr11}, SP \leftarrow SP - 2$			
CALLT	[addr5]	1	$(SP - 1) \leftarrow (PC + 1)_H, (SP - 2) \leftarrow (PC + 1)_L,$ $PC_H \leftarrow (00000000, \text{addr5} + 1),$ $PC_L \leftarrow (00000000, \text{addr5}), SP \leftarrow SP - 2$			
BRK		1	$(SP - 1) \leftarrow \text{PSW}, (SP - 2) \leftarrow (PC + 1)_H$ $(SP - 3) \leftarrow (PC + 1)_L, PC_L \leftarrow (003EH),$ $PC_H \leftarrow (003FH), SP \leftarrow SP - 3, IE \leftarrow 0$			
RET		1	$PC_L \leftarrow (SP), PC_H \leftarrow (SP + 1), SP \leftarrow SP + 2$			
RETI		1	$PC_L \leftarrow (SP), PC_H \leftarrow \text{PSW} \leftarrow (SP + 2),$ $SP \leftarrow (SP + 3), \text{NMIS} \leftarrow 0$	R	R	R
RETB		1	$PC_L \leftarrow (SP), PC_H \leftarrow \text{PSW} \leftarrow (SP + 2),$ $SP \leftarrow (SP + 3)$	R	R	R

(11) Stack manipulation instructions: **PUSH, POP, MOVW, INCW, DECW**

Mnemonic	Operand	No. of bytes	Operation	Flags		
				Z	AC	CY
PUSH	PSW	1	$(SP - 1) \leftarrow \text{PSW}, SP \leftarrow SP - 1$			
	sfr	2	$(SP - 1) \leftarrow \text{sfr}, SP \leftarrow SP - 1$			
	rp	1	$(SP - 1) \leftarrow rp_H, (SP - 2) \leftarrow rp_L, SP \leftarrow SP - 2$			
POP	PSW	1	$\text{PSW} \leftarrow (SP), SP \leftarrow SP + 1$	R	R	R
	sfr	2	$\text{sfr} \leftarrow (SP), SP \leftarrow SP + 1$			
	rp	1	$rp_L \leftarrow (SP), rp_H \leftarrow (SP + 1), SP \leftarrow SP + 2$			
MOVW	SP, #word	4	$SP \leftarrow \text{word}$			
	SP, AX	2	$SP \leftarrow \text{AX}$			
	AX, SP	2	$\text{AX} \leftarrow \text{SP}$			
INCW	SP	2	$SP \leftarrow SP + 1$			
DECW	SP	2	$SP \leftarrow SP - 1$			

(12) Unconditional branch instruction: **BR**

Mnemonic	Operand	No. of bytes	Operation	Flags		
				Z	AC	CY
BR	!addr16	3	$PC \leftarrow \text{addr16}$			
	rp1	2	$PC_H \leftarrow rp_H, PC_L \leftarrow rp_L$			
	\$ addr16	2	$PC \leftarrow PC + 2 + \text{jdisp8}$			

(13) Conditional branch instructions: BC, BL, BNC, BNL, BZ, BE, BNZ, BNE, BT, BF, BTCLR, DBNZ

Mnemonic	Operand	No. of bytes	Operation	Flags		
				Z	AC	CY
BC	\$ addr16	2	PC ← PC + 2 + jdisp8 if CY = 1			
BL						
BNC	addr16	2	PC ← PC + 2 + jdisp8 if CY = 0			
BNL						
BZ	\$ addr16	2	PC ← PC + 2 + jdisp8 if Z = 1			
BE						
BNZ	\$ addr16	2	PC ← PC + 2 + jdisp8 if Z = 0			
BNE						
BT	saddr.bit, \$ addr16	3	PC ← PC + 3 + jdisp8 if (saddr.bit) = 1			
	sfr.bit, \$ addr16	4	PC ← PC + 4 + jdisp8 if sfr.bit = 1			
	A.bit, \$ addr16	3	PC ← PC + 3 + jdisp8 if A.bit = 1			
	X.bit, \$ addr16	3	PC ← PC + 3 + jdisp8 if X.bit = 1			
	PSW.bit, \$ addr16	3	PC ← PC + 3 + jdisp8 if PSW.bit = 1			
BF	saddr.bit, \$ addr16	4	PC ← PC + 4 + jdisp8 if (saddr.bit) = 0			
	sfr.bit, \$ addr16	4	PC ← PC + 4 + jdisp8 if sfr.bit = 0			
	A.bit, \$ addr16	3	PC ← PC + 3 + jdisp8 if A.bit = 0			
	X.bit, \$ addr16	3	PC ← PC + 3 + jdisp8 if X.bit = 0			
	PSW.bit, \$ addr16	3	PC ← PC + 3 + jdisp8 if PSW.bit = 0			
BTCLR	saddr.bit, \$ addr16	4	PC ← PC + 4 + jdisp8 if (saddr.bit) = 1 then reset (saddr.bit)			
	sfr.bit, \$ addr16	4	PC ← PC + 4 + jdisp8 if sfr.bit = 1 then reset sfr.bit			
	A.bit, \$ addr16	3	PC ← PC + 3 + jdisp8 if A.bit = 1 then reset A.bit			
	X.bit, \$ addr16	3	PC ← PC + 3 + jdisp8 if X.bit = 1 then reset X.bit			
	PSW.bit, \$ addr16	3	PC ← PC + 3 + jdisp8 if PSW.bit = 1 then reset PSWH.bit	×	×	×
DBNZ	r1, \$ addr16	2	r1 ← r1 - 1, then PC ← PC + 2 + jdisp8 if r1 ≠ 0			
	saddr, \$ addr16	3	(saddr) ← (saddr) - 1, then PC ← PC + 3 + jdisp8 if (saddr) ≠ 0			

(14) CPU control instructions: MOV, SEL, NOP, EI, DI

Mnemonic	Operand	No. of bytes	Operation	Flags		
				Z	AC	CY
MOV	STBC, #byte	4	STBC \leftarrow byte			
SEL	RBn	2	RBS1 - 0 \leftarrow n, n = 0 - 3			
NOP		1	No operation			
EI		1	IE \leftarrow 1 (Enable interrupts)			
DI		1	IE \leftarrow 0 (Disable interrupts)			

18.3 INSTRUCTION LISTS FOR EACH ADDRESSING TYPE

(1) 8-bit instructions

MOV, XCH, ADD, ADDC, SUB, SUBC, AND, OR, XOR, CMP, MULU, DIVUW, INC, DEC, ROR, ROL, RORC, ROLC, SHR, SHL, ROR4, ROL4, and DBNZ

Table 18-1 8-Bit Instructions for Each Addressing Type

First operand \ Second operand	# byte	A	r r'	saddr saddr'	sfr	mem	& mem	!addr16	& !addr16	PSW	n	None ^{Note 2}
A	ADD ^{Note 1}		MOV XCH	MOV XCH ADD ^{Note 1}	MOV XCH ADD ^{Note 1}	MOV XCH ADD ^{Note 1}	MOV XCH ADD ^{Note 1}	MOV	MOV	MOV		
r	MOV		MOV XCH ADD ^{Note 1}								ROR RORC ROL ROLC SHR SHL	MULU DIVUW DEC INC
rl												DBNZ
saddr	MOV ADD ^{Note 1}	MOV		MOV XCH ADD ^{Note 1}								DEC INC DBNZ
sfr	MOV ADD ^{Note 1}	MOV										POP PUSH
mem & mem		MOV										
mem1 & mem1												ROR4 ROL4
!addr16 & !addr16		MOV										
PSW	MOV	MOV										POP PUSH
STBC	MOV											

Notes 1. ADDC, SUB, SUBC, AND, OR, XOR, and CMP are the same as ADD.

2. The second operand does not exist or is not an operand address.

(2) 16-bit instructions

MOVW, ADDW, SUBW, CMPW, INCW, DECW, SHRW, and SHLW

Table 18-2 16-Bit Instructions for Each Addressing Type

First operand \ Second operand	# word	AX	rp rp'	saddrp	sfr	mem1	& mem1	SP	n	None
AX	ADDW SUBW CMPW		ADDW SUBW CMPW	MOVW ADDW SUBW CMPW	MOVW ADDW SUBW CMPW	MOVW	MOVW	MOVW		
rp	MOVW		MOVW						SHLW SHRW	DECW INCW PUSH POP
saddrp	MOVW	MOVW								
sfrp	MOVW	MOVW								
mem1 & mem1		MOVW								
SP	MOVW	MOVW								DECW INCW

(3) Bit manipulation instructions

MOV1, AND1, OR1, XOR1, SET1, CLR1, NOT1, BT, BF, and BTCLR

Table 18-3 Bit Manipulation Instructions for Each Addressing Type

Second operand First operand	CY	A. bit	/A. bit	X. bit	/X. bit	saddr. bit	/saddr. bit	sfr. bit	/sfr. bit	PSW. bit	/PSW. bit	None ^{Note}
CY		MOV1 AND1 OR1 XOR1	AND1 OR1	MOV1 AND1 OR1 XOR1	AND1 OR1	MOV1 AND1 OR1 XOR1	AND1 OR1	MOV1 AND1 OR1 XOR1	AND1 OR1	MOV1 AND1 OR1 XOR1	AND1 OR1	CLR1 NOT1 SET1
A. bit	MOV1											CLR1 NOT1 SET1 BF BT BTCLR
X. bit	MOV1											CLR1 NOT1 SET1 BF BT BTCLR
saddr. bit	MOV1											CLR1 NOT1 SET1 BF BT BTCLR
sfr. bit	MOV1											CLR1 NOT1 SET1 BF BT BTCLR
PSW. bit	MOV1											CLR1 NOT1 SET1 BF BT BTCLR

Note The second operand does not exist or is not an operand address.

(4) Call instructions and branch instructions

CALL, CALLF, CALLT, BR, BC, BT, BF, BTCLR, DBNZ, BL, BNC, BNL, BZ, BE, BNZ, and BNE

★

Table 18-4 Call Instructions and Branch Instructions for Each Addressing Type

Instruction addressing operand	\$addr16	!addr16	rp	!addr11	[addr5]
Basic instruction	BR BC ^{Note}	CALL BR	CALL BR	MOV1	MOV1
Composite instruction	BT BF BTCLR DBNZ				

Note BL, BNC, BNL, BZ, BF, BNZ, and BNE are the same as BC.**(5) Other instructions**

ADJBA, ADJBS, BRK, RET, RETI, RETB, NOP, EI, DI, and SEL

★

APPENDIX A 78K/II SERIES PRODUCT LIST

The following pages list the 78K/II series products.
For details, refer to each User's Manual.

μPD78214 Sub-Series

Series name		μPD78214 Sub-Series			μPD78218A Sub-Series		μPD78224 Sub-Series		
Item	Product name	μPD78212	μPD78213	μPD78214 (μPD78P214)	μPD78217A	μPD78218A (μPD78P218A)	μPD78220	μPD78224 (μPD78P224)	
	Number of basic instructions		65 (instructions common to all 78K/II series products)						
Minimum instruction execution time		333 ns	500 ns	333 ns	500 ns	333 ns	500 ns	333 ns	
PUSH PSW instruction execution time (number of clocks)		When the stack area is configured in the internal dual-port RAM: 5 or 7 Other cases: 7 or 9			When the stack area is configured in the internal dual-port RAM: 6 Other cases: 8		When the stack area is configured in the internal dual-port RAM: 5 or 7 Other cases: 7 or 9		
Operating temperature and voltage ranges		Other than μPD78P2148A: -40 to +85°C, V _{DD} = +5 V ±10%					-40 to +85°C, V _{DD} = +5 V ±5%		μPD78P2148A : -40 to +85°C, V _{DD} = +5 V ±0.3 V
General-purpose registers		8 bits × 8 × 4 banks							
Bank registers		P6 and PM6					P6 only		
Internal memory	ROM (bytes)	8K	None	16K	None	32K	None	16K	
	EEPROM	None							
	RAM (bytes)	384	512	1024	1024	640	640		
Memory space		Program memory space: 64K bytes, Data memory space: 1M byte							
I/O pins	Input	14					8		
	Output	12					12	20	
	I/O	28	10	28	10	28	25	35	
	Total	54	36	54	36	54	45	63	
Ancillary function pins ^{Note}	With a pull-up resistor	34	16	34	16	34	None		
	LED direct drive output	16	0	16	0	16	8		
	Transistor direct drive output	8					None		
P0		8-bit output port							
P1		—					8-bit I/O port		
P2		8-bit input port							
P3		8-bit I/O port							
P4		8-bit I/O port	—	8-bit I/O port	—	8-bit I/O port	—	8-bit I/O port	
P5		8-bit I/O port	—	8-bit I/O port	—	8-bit I/O port	—	8-bit output port	
P6		4-bit output port + 4-bit I/O port	4-bit output port + 2-bit I/O port	4-bit output port + 4-bit I/O port	4-bit output port + 2-bit I/O port	4-bit output port + 4-bit I/O port	4-bit output port + 2-bit I/O port	4-bit output port + 4-bit I/O port	
P7		6-bit input port					7-bit I/O port		

Note The ancillary function pins are included in the I/O pins.

(1/3)

μPD78234 Sub-Series				μPD78244 Sub-Series	
μPD78233	μPD78234	μPD78237	μPD78238 (μPD78P238)	μPD78243	μPD78244
65 (instructions common to all 78K/II series products)					
500 ns	333 ns	500 ns	333 ns	500 ns	333 ns
When the stack area is configured in the internal dual-port RAM: 6 Other cases: 8					
-40 to +85°C, V _{DD} = +5 V ±10%				-10 to +70°C, V _{DD} = +5 V ±10%	
8 bits × 8 × 4 banks					
P6 and PM6					
None	16K	None	32K (32/16K ^{Note})	None	16K
None				512	
640		1024		1024K (1024/640K ^{Note})	
Program memory space: 64K bytes/Data memory space: 1M byte					
16				14	
12					
18	36	18	36	10	28
46	64	46	64	36	54
24	42	24	42	16	34
8	24	8	24	0	16
8					
8-bit output port					
8-bit I/O port				—	
8-bit input port					
8-bit I/O port					
—	8-bit I/O port	—	8-bit I/O port	—	8-bit I/O port
—	8-bit I/O port	—	8-bit I/O port	—	8-bit I/O port
4-bit output port +	4-bit output port +	4-bit output port +	4-bit output port +	4-bit output port +	4-bit output port +
2-bit I/O port	4-bit I/O port	2-bit I/O port	4-bit I/O port	2-bit I/O port	4-bit I/O port
8-bit input port				6-bit input port	

Note Set by software.

μPD78214 Sub-Series

Series name		μPD78214 Sub-Series			μPD78218A Sub-Series		μPD78224 Sub-Series	
Item	Product name	μPD78212	μPD78213	μPD78214 (μPD78P214)	μPD78217A	μPD78218A (μPD78P218A)	μPD78220	μPD78224 (μPD78P224)
PWM output		None						
Comparator		None					4 bits × 8	
A/D converter		8 bits × 8					None	
Selection of conversion time		Selected according to operating frequency					—	
AV _{REF} input voltage range		3.4 V to V _{DD}			3.6 V to V _{DD}		—	
Restrictions on input voltage		Pins selected by bits ANI0 to ANI2 of the ADM register. Pin voltage is always 0 V to AV _{REF} .			Pins subject to A/D conversion. Pin voltage is 0 V to AV _{REF} during A/D conversion.		—	
D/A converter		None						
Timer/counter	16-bit timer/counter	1						
	8-bit timer/counter	3					2	
	Timer output	4						
	PWM/PPG output	Provided					None	
	One-shot pulse	None			Provided		None	
	Interrupt source	7					5	
External SFR area		16 bytes (0FFD0H to 0FFDFH)					None	
Serial interface	UART	1 channel						
	CSI	1 channel (SBI)						
	BRG timer	Provided (shared with timer/counter 3)					Provided	
	Baud rate generator	Provided					None	
	External baud rate clock input	Provided					None	
Real-time output port		4 bits × 2 or 8 bits × 1						

(2/3)

μ PD78234 Sub-Series				μ PD78244 Sub-Series	
μ PD78233	μ PD78234	μ PD78237	μ PD78238 (μ PD78P238)	μ PD78243	μ PD78244
12 bits \times 2				None	
None					
8 bits \times 8					
Selected freely				Selected according to operating frequency	
3.4 V to V_{DD}				3.6 V to V_{DD}	
Pins subject to A/D conversion. Pin voltage is 0 V to AV_{REF} during A/D conversion.					
8 bits \times 2				None	
1					
3					
4					
Provided					
Provided					
7					
16 bytes (0FFD0H to 0FFDFH)					
1 channel					
1 channel (SBI)					
Provided (shared with timer/counter 3)					
Provided					
Provided					
4 bits \times 2 or 8 bits \times 1					

μPD78214 Sub-Series

Series name	μPD78214 Sub-Series			μPD78218A Sub-Series		μPD78224 Sub-Series		
Item	Product name	μPD78212	μPD78213	μPD78214 (μPD78P214)	μPD78217A	μPD78218A (μPD78P218A)	μPD78220	μPD78224 (μPD78P224)
Interrupt	2 levels (programmable), vector/macro service							
External	7						8	
Internal	12						9	
Interrupts that can use macro service	15						6	
Bits of macro service counter	8 bits only			8/16 bits selectable (except type A)		8 bits only		
Incrementing MPD and MPT of type C macro service	Increments lower 8 bits only (higher bits remain as is)			Increments 16 bits		Increments lower 8 bits only (higher bits remain as is)		
Constraints on memory-to-SFR data transfer by type A macro service	Occurs when transferred data is D0H to DFH			Occurs when transfer source buffer (memory) address is 0FED0H to 0FEDFH		Occurs when transferred data is D0H to DFH		
Macro service execution time	Depends on mode. Refer to the relevant user's manual.							
Standby function	HALT/STOP mode							
Oscillation setting time upon releasing STOP mode	Fixed			Selected from two options		Fixed		
Pseudo SRAM refresh function	Provided (refresh pulse width: $1/f_{CLK}$)					Provided (refresh pulse width: $1.5/f_{CLK}$)		
Constraints on memory access	None					FC80H to FDFH cannot be accessed during refresh		
ROM-less mode setting	\overline{EA} pin = low level	—	\overline{EA} pin = low level	—	\overline{EA} pin = low level	—	\overline{EA} pin = low level	
Package	<ul style="list-style-type: none"> • 64-pin plastic shrink DIP (750 mil) • 68-pin plastic QFJ (except μPD78212) • 64-pin plastic QFP (14 × 14 mm) • 74-pin plastic QFP (20 × 20 mm) • 64-pin plastic QUIP (except μPD78212) • 64-pin ceramic shrink DIP with window (for μPD78P214 only) 			<ul style="list-style-type: none"> • 64-pin plastic shrink DIP (750 mil) • 64-pin plastic QFP (14 × 14 mm) • 64-pin ceramic shrink DIP with window (for μPD78P218A only) 		<ul style="list-style-type: none"> • 84-pin plastic QFJ • 94-pin plastic QFP (20 × 20 mm) 		

★

(3/3)

μPD78234 Sub-Series				μPD78244 Sub-Series	
μPD78233	μPD78234	μPD78237	μPD78238 (μPD78P238)	μPD78243	μPD78244
2 levels (programmable), vector/macro service					
7					
12			14		
15					
8/16 bits selectable (except type A)					
Increments 16 bits					
Occurs when transferred data is D0H to DFH				Occurs when transfer source buffer (memory) address is 0FED0H to 0FEDFH	
Depends on mode. Refer to the relevant user's manual.					
HALT/STOP mode					
Selected from two options					
Provided (refresh pulse width: $1/f_{CLK}$)					
None					
—	MODE pin = high level	—	MODE pin = high level (cannot be set)	—	\overline{EA} pin = low level
<ul style="list-style-type: none"> • 84-pin plastic QFJ • 80-pin plastic QFP (14 × 14 mm) • 94-pin plastic QFP (20 × 20 mm) • 94-pin ceramic WQFN (for μPD78P238 only) 				<ul style="list-style-type: none"> • 64-pin plastic shrink DIP (750 mil) • 64-pin plastic QFP (14 × 14 mm) 	

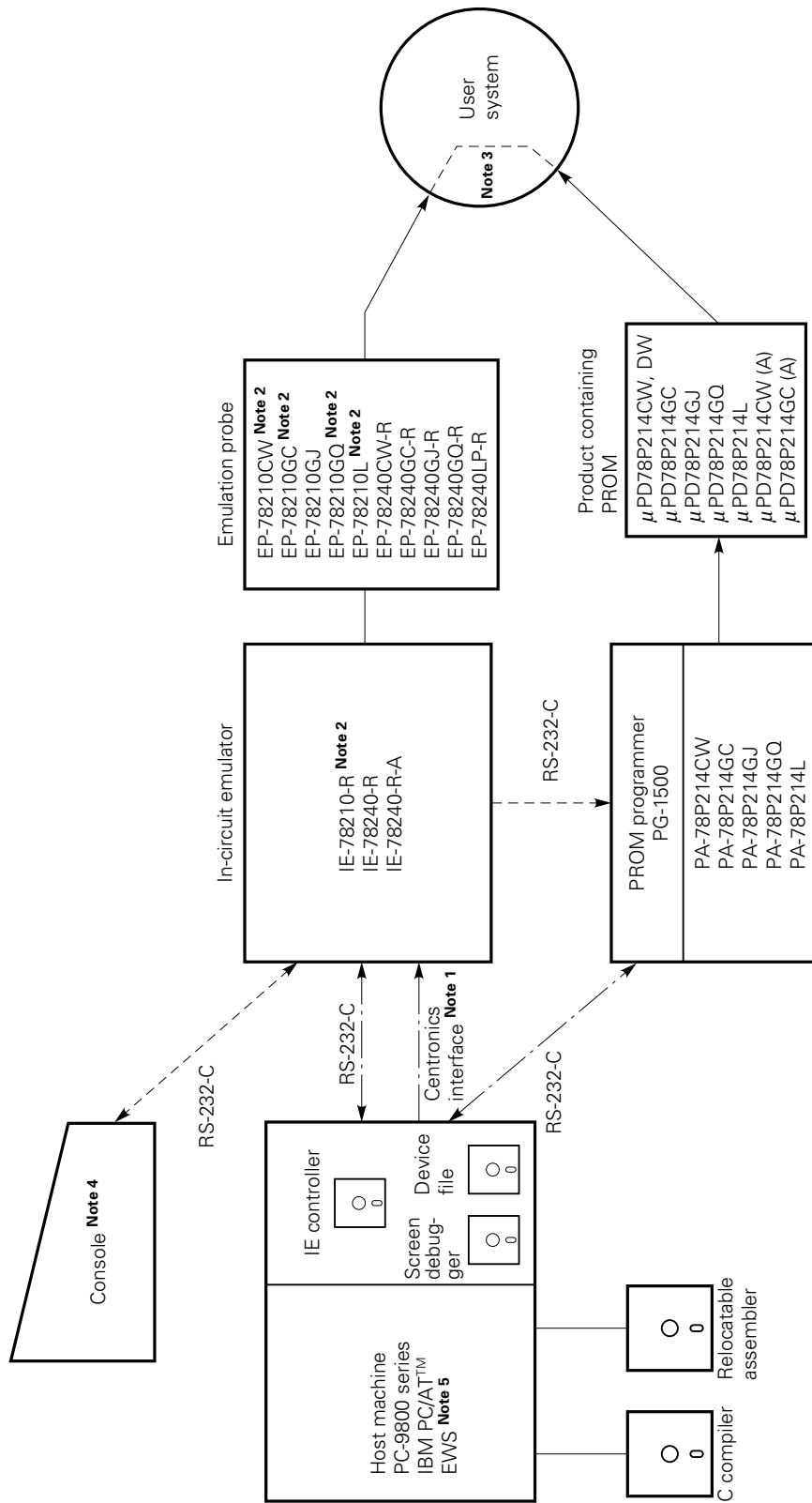
★

A

APPENDIX B DEVELOPMENT TOOLS

The development tools described on the following pages are available for the development of systems using μ PD78214 sub-series.

DEVELOPMENT ENVIRONMENT



-----: When the in-circuit emulator is connected to the host machine

-----: When the in-circuit emulator is connected to the console so that it is used as a stand-alone emulator.

- Notes**
1. Used for high-speed file transfer (for IE-78240-R only during download)
 2. Not supported
 3. EV-9200G-74 or EV-9220GC-64
 4. Only when IE-78240-R is used
 5. EWS may be the HP9000 series 300, SUN4/3900, or EWS-4800/200 series. EWS cannot be connected with an in-circuit emulator.

B.1 HARDWARE (1/2)

IE-78240-R-A	The IE-78240-R-A is an enhanced version of the IE-78210-R and IE-78240-R. This in-circuit emulator can be used for any model of the μ PD78214 sub-series. It operates with a PC-9800 series or IBM PC/AT host machine. By using this emulator together with the optional screen debugger and device file, programs written in C or a structured assembly language can be debugged at the source program level. Features such as the simultaneous tracing of data access and program fetch, as well as C0 coverage, enable efficient debugging and program testing. For those already using the IE-78210-R or IE-78240-R, the emulator can be upgraded to a level equivalent to the IE-78240-R-A simply by adding an optional board (IE-78200-R-BK).
IE-78240-R ^{Note 2} IE-78210-R ^{Note 2}	These in-circuit emulators can be used for any model of the μ PD78214 sub-series ^{Note 1} . They are connected to a host machine or a console to enable debugging. When connected to a host machine, they enable symbolic debugging and allow object files to be exchanged with the host machine, thus making the debugging process more efficient. These emulators each have two RS-232C ports, to which the PC-1500 PROM programmer can be connected.
IE-78240-R-EM IE-78210-R-EM ^{Note 2} IE-78200-R-EM ^{Note 2} IE-78200-R-BK	These boards are used to upgrade the 75X and 78K series in-circuit emulators to a level equivalent to the IE-78240-R-A, IE-78240-R, or IE-78210-R ^{Note 2} . For details, see B.3 .
EP-78240CW-R EP-78210CW ^{Note 2}	These emulation probes are used for the μ PD78210CW, μ PD78212CW-xxx, μ PD78213CW, μ PD78214CW-xxx, μ PD78P214CW, μ PD78212CW(A)-xxx, μ PD78213CW(A), and μ PD78214CW(A)-xxx. The EP-78240CW-R is essentially the same as the EP-78210CW, the only difference being that the former has a longer cable.
EP-78240GC-R EP-78210GC ^{Note 2}	These emulation probes are used for the μ PD78212GC-xxx-AB8, μ PD78213GC-AB8, μ PD78214GC-xxx-AB8, μ PD78P214GC-AB8, μ PD78212GC(A)-xxx-AB8, and μ PD78214GC(A)-xxx-AB8. Both are used together with the EV-9200GC-64. The EP-78240GC-R is essentially the same product as the EP-78210GC, the only difference being that the former has a longer cable.
EP-78210GJ ^{Note 2}	This emulation probe is used for the μ PD78210GJ-5BJ, μ PD78212GJ-xxx-5BJ, μ PD78213GJ-5BJ, μ PD78214GJ-xxx-5BJ, μ PD78P214GJ-5BJ, and μ PD78214GJ(A)-xxx-5BJ. It is used together with the EV-9200G-74 and EP-78210L or EP-78240LP-R.
EP-78240GJ-R	This emulation probe is used for the μ PD78210GJ-5BJ, μ PD78212GJ-xxx-5BJ, μ PD78213GJ-5BJ, μ PD78214GJ-xxx-5BJ, μ PD78P214GJ-5BJ, and μ PD78214GJ(A)-xxx-5BJ. It is used together with the EV-9200G-74. Unlike the EP-78210GJ, this probe is of single-unit type, and is thus easier to use.
EP-78240GQ-R EP-78210GQ ^{Note 2}	These emulation probes are used for the μ PD78213GQ-36, μ PD78214GQ-xxx-36, μ PD78P214GQ-36, μ PD78213GQ(A)-36, and μ PD78214GQ(A)-xxx-36. The EP-78240GQ-R is essentially the same product as the EP-78210GQ, the only difference being that the former has a longer cable.
EP-78240LP-R EP-78210L ^{Note 2}	These emulation probes are used for the μ PD78210L, μ PD78213L, μ PD78214L-xxx, μ PD78P214L, and μ PD78214L(A)-xxx. The EP-78240LP-R is essentially the same product as the EP-78210L, the only difference being that the former has a longer cable.

Notes 1. μ PD78212, μ PD78213, μ PD78214, μ PD78P214, μ PD78212(A), μ PD78213(A), and μ PD78214(A)

2. These products are no longer produced and are not available from NEC.

HARDWARE (2/2)

EV-9200G-74	This socket is mounted on the board of the user system developed for the 74-pin QFP. It is used together with the EP-78210GJ or EP-78240GJ-R.
EV-9200GC-64	This socket is mounted on the board of the user system developed for the 64-pin QFP. It is used together with the EP-78210GC or EP-78240GC-R.
PG-1500	This PROM programmer, when connected to an accessory board and the optional programmer adapter, enables programming of the PROM built into a single-chip microcomputer, either in its stand-alone mode or under the control of the host machine. PROMs having typical capacities, from 256K to 4M bits, can be programmed.
PA-78P214CW	This PROM programmer adapter is used with the μPD78P214CW and μPD78P214DW. It is used together with a PROM programmer such as the PG-1500.
PA-78P214GC	This PROM programmer adapter is used with the μPD78P214GC-AB8. It is used together with a PROM programmer such as the PG-1500.
PA-78P214GJ	This PROM programmer adapter is used with the μPD78P214GJ-5BJ. It is used together with a PROM programmer such as the PG-1500.
PA-78P214GQ	This PROM programmer adapter is used with the μPD78P214GQ-36. It is used together with a PROM programmer such as the PG-1500.
PA-78P214L	This PROM programmer adapter is for the μPD78P214L. It is used together with a PROM programmer such as the PG-1500.

- ★
- Remarks**
1. The EP-78210GC, EP-78210GJ, EP-78240GC-R, and EP-78240GJ-R are provided with one EV-9200G-74 or EV-9200GC-64.
 2. The EV-9200G-74 and EV-9200GC-64 are supplied in batches of five.

B.2 SOFTWARE

B.2.1 Language Processing Software (1/3)

78K/II series relocatable assembler (RA78K/II)	<p>This relocatable assembler can be used for all the 78K/II series products. Its macro functions enhance efficiency in software development. It also includes a structured assembler, which makes the program control structure more comprehensive, thus improving software productivity and maintainability. The relocatable assembler consists of the following programs:</p>
Structured assembler preprocessor (program name: ST78K2)	Converts a source program written in the structured assembler language into a form that can be input into the relocatable assembler.
Relocatable assembler (program name: RA78K2)	Converts a source program written in assembly language into a machine language program, enabling the generation of a relocatable object module file.
Linker (program name: LK78K2)	Links an object module file, generated by the relocatable assembler, with a library file, to determine the absolute address of the program and to generate a load module file.
Object converter (program name: OC78K2)	Converts a load module file, generated by the linker, into a suitable form for downloading to the in-circuit emulator and PROM programmer.
Librarian (program name: LB78K2)	Links object module files, generated by the relocatable assembler, to create a single library file. It also updates the library files.
List converter (program name: LCNV78K2)	Creates an assemble list from the assemble list file output by the relocatable assembler, using the object and load module files. The assemble list contains absolute values.

Language Processing Software (2/3)

78K/II series relocatable assembler (RA78K/II)	Host machine	OS	Distribution medium	Part number
	PC-9800 series	MS-DOSTM (Ver.3.30 to Ver.5.00ANote 3)	8-inch 2DNote1	μS5A1RA78K2
			5.25-inch 2HD	μS5A10RA78K2
			3.5-inch 2HD	μS5A13RA78K2
	IBM PC/AT or compatibles	See Section B.2.4.	5.25-inch 2DNote2	μS7B11RA78K2
			5.25-inch 2HC	μS7B10RA78K2
			3.5-inch 2HC	μS7B13RA78K2
	HP9000 series300TM	HP-UXTM (rel.7.05B)	Cartridge tape (QIC-24)	μS3H15RA78K2
SPARCstationTM	Sun OSTM (rel.4.1.1)	μS3K15RA78K2		
EWS-4800 seriesTM (RISC)	EWS-UX/VTM (rel.4.0)	μS3M15RA78K2		
78K/II series C compiler (CC78K/II)	This C compiler can be used with all 78K/II series products. Its language specifications conform to ANSI standards, and compiled programs can be written into ROM. The compiler offers such features as special function register manipulation, bit manipulation, variables using short direct addressing, and interrupt control. The use of these features ensures effective programming and high object efficiency. It also has a start-up routine sample program and a standard function object library. To use this compiler, the 78K/II series relocatable assembler (RA78K/II) is necessary.			
	Host machine	OS	Distribution medium	Part number
	PC-9800 series	MS-DOS (Ver.3.30 to Ver.5.00ANote 3)	5.25-inch 2HD	μS5A10CC78K2
			3.5-inch 2HD	μS5A13CC78K2
	IBM PC/AT or compatibles	See Section B.2.4.	5.25-inch 2DNote2	μS7B11CC78K2
			5.25-inch 2HC	μS7B10CC78K2
			3.5-inch 2HC	μS7B13CC78K2
	HP9000 series300	HP-UJ (rel.7.05B)	Cartridge tape (QIC-24)	μS3H15CC78K2
SPARCstation	Sun OS (rel.4.1.1)	μS3K15CC78K2		
EWS-4800 series (RISC)	EWS-UJ/V (rel.4.0)	μS3M15CC78K2		

- Notes**
1. The 8-inch 2D model has been superseded by the 5.25-inch 2HD and 3.5-inch 2HD models. Those users who have already purchased an 8-inch 2D model will be supplied with a 5.25-inch 2HC model when the product is next upgraded.
 2. The 5.25-inch 2D model is no longer available. Those users who have already purchased a 5.25-inch 2D model will be supplied with a 5.25-inch 2HC model when the product is next upgraded.
 3. Versions 5.00 and 5.00A feature a the task swap function. However, the task swap function cannot be used with this software.

Language Processing Software (3/3)

78K/II series C compiler library source file	This source program is used to modify the libraries supplied with CC78K/II to satisfy user specifications.			
	Host machine	OS	Distribution medium	Part number
	PC-9800 series	MS-DOS (Ver.3.30 to Ver.5.00A ^{Note})	5.25-inch 2HD	μS5A10CC78K2-L
			3.5-inch 2HD	μS5A13CC78K2-L
	IBM PC/AT or compatible	See Section B.2.4.	5.25-inch 2HC	μS7B10CC78K2-L
			3.5-inch 2HC	μS7B13CC78K2-L
	HP9000 series300	HP-UX (rel.7.05B)	Cartridge tape (QIC-24)	μS3H15CC78K2-L
	SPARCstation	Sun OS (rel.4.1.1)		μS3K15CC78K2-L
EWS-4800 series (RISC)	EWS-UX/V (rel.4.0)	μS3M15CC78K2-L		

Note Versions 5.00 and 5.00A feature a task swap function. However, the task swap function cannot be used with this software.

B.2.2 Software for the In-Circuit Emulator (1/2)

Screen debugger (SD78K/II)	This program controls the in-circuit emulator for the 78K/II series when used together with the device file (DF78210). It can be used when the in-circuit emulator has been upgraded to IE-78240-R-A class and a PC-9800 series or IBM PC/AT computer is being used as a host computer. This debugger can debug source programs written in C, structured assembly language, and assembly language. Its split screen function, by which the screen is split into sections to enable the simultaneous display of different information, makes debugging more efficient.			
	Host machine	OS	Distribution medium	Part number
	PC-9800 series	MS-DOS (Ver.3.30 to Ver.5.00A ^{Note})	5.25-inch 2HD	μS5A10SD78K2
			3.5-inch 2HD	μS5A13SD78K2
	IBM PC/AT or compatible	See Section B.2.4.	5.25-inch 2HC	μS7B10SD78K2
			3.5-inch 2HC	μS7B13SD78K2
Device file (DF78210)	This is used together with the screen debugger (SD78K/II) to debug programs of the μPD78214 sub-series.			
	Host machine	OS	Distribution medium	Part number
	PC-9800 series	MS-DOS (Ver.3.30 to Ver.5.00A ^{Note})	5.25-inch 2HD	μS5A10DF78210
			3.5-inch 2HD	μS5A13DF78210
	IBM PC/AT or compatible	See Section B.2.4.	5.25-inch 2HC	μS7B10DF78210
			3.5-inch 2HC	μS7B13DF78210

Note Versions 5.00 and 5.00A feature a task swap function. However, the task swap function cannot be used with this software.

★

★

★

B

★

Software for the In-Circuit Emulator (2/2)

In-circuit emulator control program (IE78210) Note 1 (IE78240)	This program enables control of the in-circuit emulator for the 78K/II series from the host machine. It can automatically execute commands, thus enhancing efficiency in debugging. The following programs are available, depending on the type of in-circuit emulator:				
	Emulator	Host machine	OS	Distribution medium	Part number
	IE-78210-R IE-78210-R-EM	PC-9800 series (Ver.3.10 to Ver5.00A Note 2)	MS-DOS	8-inch 2D Note 3	μS5A11E78210-P01
				5.25-inch 2HD	μS5A10IE78210-P01
				3.5-inch 2HD	μS5A13IE78210
		IBM PC/AT or compatible	See Section B.2.4.	5.25-inch 2D Note 4	μS7B11IE78210-P02
				5.25-inch 2HC	μS7B10IE78210
				3.5-inch 2HC	μS7B13IE78210
	IE-78240-R IE-78240-R-EM	PC-9800 series (Ver.3.10 to Ver5.00A Note 2)	MS-DOS	8-inch 2D Note 3	μS5A11IE78240
				5.25-inch 2HD	μS5A10IE78240
3.5-inch 2HD				μS5A13IE78240	
IBM PC/AT or compatible		See Section B.2.4.	5.25-inch 2D Note 4	μS7B11IE78240	
			5.25-inch 2HC	μS7B10IE78240	
			3.5-inch 2HC	μS7B13IE78240	

- Notes**
1. When using the IE-78210-R or IE-78210-R-EM, the IE-78210 is also necessary. For the IE-78240-R or IE-78240-R-EM, the IE-78240 is necessary. When using the IE-78210-R together with the IE-78240-R-EM, the IE-78210 is necessary.
 2. Versions 5.00 and 5.00A feature a task swap function. However, the task swap function cannot be used with this software.
 3. The 8-inch 2D model has been superseded by the 5.25-inch 2HD and 3.5-inch 2HD models. Those users who have already purchased an 8-inch 2D model will be supplied with a 5.25-inch 2HD model when the product is next upgraded.
 4. The 5.25-inch 2D model is no longer available. Those users who have already purchased a 5.25-inch 2D model will be supplied with a 5.25-inch 2HC model when the product is next upgraded.



B.2.3 Software for the PROM Programmer

PG-1500 controller	This program provides the serial and parallel interfaces between PG-1500 and the host machine, enabling the host machine to control the PG-1500.			
	Host machine	OS	Distribution medium	Part number
	PC-9800 series	MS-DOS (Ver.3.10 to Ver.5.00A ^{Note 1})	5-inch 2HD	μS5A10PG1500
			3.5-inch 2HD	μS5A13PG1500
	IBM PC/AT or compatible	See Section B.2.4.	5-inch 2D ^{Note 2}	μS7B11PG1500
			5-inch 2HC	μS7B10PG1500
			3.5-inch 2HC	μS5A13PG1500

- Notes**
1. Versions 5.00 and 5.00A feature a task swap function. However, the task swap function cannot be used with this software. ★
 2. The 5.25-inch 2D model is no longer available. Those users who have already purchased a 5.25-inch 2D model will be supplied with a 5.25-inch 2HC model when the product is next upgraded.

B.2.4 OS for the IBM PC ★

The following OSs are supported for the IBM PC:

OS	Version
PC DOS	Ver.3.1 to Ver.6.1
Windows ^{Note 1}	Ver.3.1
MS-DOS	Ver.5.0 to Ver.6.0 5.0/V ^{Note 2}
IBM DOS	J5.02/V ^{Note 2}

- Notes**
1. PC DOS and Windows are used together for the fuzzy knowledge-data creation tool.
 2. Only English-version systems are supported.

Caution Versions 5.00 and later feature a task swap function. However, the task swap function cannot be used with this software.

B.3 UPGRADING OTHER IN-CIRCUIT EMULATORS TO 78K/II SERIES LEVEL

The 78K series and 75X series in-circuit emulators can be upgraded to the level of the 78K/II series by replacing their internal boards with an optional board.

Note that the upgraded in-circuit emulator requires an appropriate new control program.

B.3.1 Upgrading to IE-78240-R-A Level

Emulator	IE Group Number	Required Board	Remarks
IE-78230-R-A IE-78140-R	1	IE-78240-R-EM	—
IE-78240-R ^{Note 1}	2	IE-78200-R-BK	—
IE-78112-R ^{Note 1} IE-78210-R ^{Note 1} IE-78220-R ^{Note 1} IE-78310-R ^{Note 1} IE-78310A-R	3	IE-78200-R-BK IE-78240-R-EM ^{Note 2}	The high-speed download function is not supported. Those users who are also using an in-circuit emulator of IE group 1, 2, or 4, are recommended to upgrade these emulators also. Those users with an in-circuit emulator of IE group 1 do not need to purchase the IE-78200-R-BK (the IE-78200-R-BK board is built into the IE group 1 in-circuit emulator).
IE-75000-R IE-75001-R IE-78000-R IE-78130-R IE-78230-R IE-78320-R ^{Note 1} IE-78327-R IE-78330-R IE-78350-R IE-78600-R ^{Note 1}	4	IE-78200-R-BK IE-78240-R-EM	Those users with an in-circuit emulator of IE group 1 do not need to purchase the IE-78200-R-BK (the IE-78200-R-BK board is built into the IE group 1 in-circuit emulator).

- Notes**
1. This product is no longer produced, and is not available from NEC.
 2. This board is used for emulation for the μPD78214 series. Those users who already have the IE-78210-R-EM^{Note 1} do not have to purchase this board.

B.3.2 Upgrading to IE-78240-R Level^{Note 1} (Upgrading to IE-78240-R-A Level is recommended.)

Emulator	IE Group Number	Required Board	Remarks
IE-78112-R ^{Note 1} IE-78210-R ^{Note 1} IE-78220-R ^{Note 1}	1	IE-78240-R-EM ^{Note 2}	The high-speed download function is not supported. Those users who are also using an in-circuit emulator of IE group 4, are recommended to upgrade to IE group 4 level.
IE-78130-R IE-78230-R ^{Note 1}	2	IE-78240-R-EM	—
IE-78310-R ^{Note 1} IE-78310A-R	3	IE-78200-R-EM ^{Note 1} IE-78240-R-EM ^{Note 2}	The high-speed download function is not supported. Those users who have an in-circuit emulator of IE group 1 do not need to purchase the IE-78200-R-EM (the IE-78200-R-EM board is built into the IE group 1 in-circuit emulator).
IE-75000-R IE-75001-R IE-78000-R IE-78320-R ^{Note 1} IE-78327-R IE-78330-R IE-78350-R IE-78600-R ^{Note 1}	4	IE-78200-R-EM ^{Note 1} IE-78240-R-EM	Those users who have an in-circuit emulator of IE group 1 do not need to purchase the IE-78200-R-EM (the IE-78200-R-EM board is built into the IE group 1 in-circuit emulator).
IE-78140-R IE-78230-R-A	5	IE-78200-R-EM ^{Note 1} IE-78240-R-EM	Upgrading to IE-78240-R-A level is recommended.

Notes 1. This product is no longer produced, and is not available from NEC.

2. This board is used for emulation for the μ PD78214 series. Those users who already have the IE-78210-R-EM^{Note 1} do not have to purchase this board.

B.3.3 Upgrading to IE-78210-R Level^{Note 2} (Upgrading to IE-78240-R-A level is recommended.)

Emulator	IE Group Number	Required Board	Remarks
IE-78112-R ^{Note 2} IE-78220-R ^{Note 2}	1	IE-78210-R-EM ^{Note 1}	—
IE-78310-R ^{Note 2} IE-78310A-R	2	IE-78200-R-EM ^{Note 2} IE-78210-R-EM ^{Note 1}	Those users who have an in-circuit emulator of IE group 1 do not need to purchase the IE-78200-R-EM (the IE-78200-R-EM board is built into the IE group 1 in-circuit emulator).
IE-75000-R IE-75001-R IE-78000-R IE-78130-R IE-78140-R IE-78230-R IE-78230-R-A IE-78240-R IE-78240-R-A IE-78320-R ^{Note 2} IE-78327-R IE-78330-R IE-78350-R IE-78600-R	3	—	Upgrading to IE-78210-R level is not allowed. Upgrading to IE-78240-R or IE-78240-R-A is recommended.

- Notes**
1. This board is no longer produced, and is not available from NEC. Those users who do not have the IE-78210-R-EM, are recommended to upgrade to the IE-78240-R-A level, which includes the functions of IE-78240-R.
 2. This product is no longer produced, and is not available from NEC.

APPENDIX C SOFTWARE FOR EMBEDDED APPLICATIONS



C.1 FUZZY INFERENCE DEVELOPMENT SUPPORT SYSTEM

Fuzzy knowledge-data creation tool	Supports input and editing, as well as the evaluation (simulation) of fuzzy knowledge-data (fuzzy rules and membership functions).			
	Host machine	OS	Distribution medium	Part number
	PC-9800 series	MS-DOS (Ver.3.30 to Ver.5.00A ^{Note})	5.25-inch 2HD	μS5A10FE9000
			3.5-inch 2HD	μS5A13FE9000
	IBM PC/AT or compatible	See Section B.2.4.	5.25-inch 2HC	μS7B10FE9200
3.5-inch 2HC			μS7B13FE9200	
Translator	This program converts fuzzy knowledge-data, obtained with the fuzzy knowledge data creation tool, into an assembler source program for the RA78K/II.			
	Host machine	OS	Distribution medium	Part number
	PC-9800 series	MS-DOS (Ver.3.30 to Ver.5.00A ^{Note})	5.25-inch 2HD	μS5A10FT9080
			3.5-inch 2HD	μS5A13FT9080
	IBM PC/AT or compatible	See Section B.2.4.	5.25-inch 2HC	μS7B10FT9085
3.5-inch 2HC			μS7B13FT9085	
Fuzzy inference module (FI78K/II)	This program performs fuzzy inference by linking with the fuzzy knowledge data converted by the translator.			
	Host machine	OS	Distribution medium	Part number
	PC-9800 series	MS-DOS (Ver.3.30 to Ver.5.00A ^{Note})	5.25-inch 2HD	μS5A10FI78K2
			3.5-inch 2HD	μS5A13FI78K2
	IBM PC/AT or compatible	See Section B.2.4.	5.25-inch 2HC	μS7B10FI78K2
3.5-inch 2HC			μS7B13FI78K2	
Fuzzy inference debugger (FD78K/II)	This program is used to evaluate and adjust fuzzy knowledge-data at the hardware level, using the in-circuit emulator.			
	Host machine	OS	Distribution medium	Part number
	PC-9800 series	MS-DOS (Ver.3.30 to Ver.5.00A ^{Note})	5.25-inch 2HD	μS5A10FD78K2
			3.5-inch 2HD	μS5A13FD78K2
	IBM PC/AT or compatible	See Section B.2.4.	5.25-inch 2HC	μS7B10FD78K2
3.5-inch 2HC			μS7B13FD78K2	

Note Versions 5.00 and 5.00A feature a task swap function. However, the task swap function cannot be used with this software.

APPENDIX D REGISTER INDEX

D.1 REGISTER INDEX

16-bit capture register (CR02) ... 111
16-bit compare register (CR00,CR01) ... 111
16-bit timer 0 (TM0) ... 111
8-bit capture/compare register (CR11) ... 142
8-bit capture register (CR22) ... 161
8-bit compare register (CR10) ... 142
8-bit compare register (CR20) ... 161
8-bit compare register (CR21) ... 161
8-bit compare register (CR30) ... 206
8-bit timer 1 (TM1) ... 142
8-bit timer 2 (TM2) ... 161
8-bit timer 3 (TM3) ... 206

A

A/D conversion result register (ADCR) ... 227
A/D converter mode register (ADM) ... 229, 304
Asynchronous serial interface mode register (ASIM) ... 245
Asynchronous serial interface status register (ASIS) ... 246

B

Baud rate generator control register (BRGC) ... 251

C

Capture/compare control register 0 (CRC0) ... 112
Capture/compare control register 1 (CRC1) ... 144
Capture/compare control register 2 (CRC2) ... 165
Clock synchronous serial interface mode register (CSIM) ... 262, 273

E

External interrupt mode register 0 (INTM0) ... 294
External interrupt mode register 1 (INTM1) ... 295, 303

I

Interrupt mask register (MK0) ... 306
Interrupt request flag register (IF0) ... 305
Interrupt service mode register (ISM0) ... 306
Interrupt status register (IST) ... 307

M

Macro service mode register ... 323
Memory expansion mode register (MM) ... 346

P

Port 0 (P0) ... 60
Port 2 (P2) ... 63

Port 3 (P3) ... 66
Port 4 (P4) ... 75
Port 5 (P5) ... 80
Port 6 (P6) ... 84
Port 7 (P7) ... 92
Port 0 buffer register (P0L, P0H) ... 97
Port 0 mode register (PM0) ... 61
Port 3 mode control register (PMC3) ... 72
Port 3 mode register (PM3) ... 72
Port 5 mode register (PM5) ... 80
Port 6 mode register (PM6) ... 89
Prescaler mode register 0 (PRM0) ... 207
Prescaler mode register 1 (PRM1) ... 143, 164
Priority specification flag register (PR0) ... 306
Programmable wait control register (PW) ... 347
Pull-up-resistor-option register (PUO) ... 65, 74, 78, 82, 91

R

Real-time output port control register (RTPC) ... 97
Refresh mode register (RFM) ... 367

S

Serial bus interface control register (SBIC) ... 264
Serial reception buffer (RXB) ... 245
Serial shift register (SIO) ... 276
Serial transmission shift register (TXS) ... 245
Standby control register (STBC) ... 379

T

Timer control register 0 (TMC0) ... 111, 207
Timer control register 1 (TMC1) ... 143, 163
Timer output control register (TOC) ... 113, 166

D.2 REGISTER SYMBOL INDEX**A**

- ADCR: A/D conversion result register ... 227
ADM: A/D converter mode register ... 229, 304
ASIM: Asynchronous serial interface mode register ... 245
ASIS: Asynchronous serial interface status register ... 246

B

- BRGC: Baud rate generator control register ... 251

C

- CR00: 16-bit compare register ... 111
CR01: 16-bit compare register ... 111
CR02: 16-bit capture register ... 111
CR10: 8-bit compare register ... 142
CR11: 8-bit capture/compare register ... 142
CR20: 8-bit compare register ... 161
CR21: 8-bit compare register ... 161
CR22: 8-bit capture register ... 161
CR30: 8-bit compare register ... 206
CRC0: Capture/compare control register 0 ... 112
CRC1: Capture/compare control register 1 ... 144
CRC2: Capture/compare control register 2 ... 165
CSIM: Clock synchronous serial interface mode register ... 262, 273

I

- IF0: Interrupt request flag register ... 305
INTM0: External interrupt mode register 0 ... 294
INTM1: External interrupt mode register 1 ... 295, 303
ISM0: Interrupt service mode register ... 306
IST: Interrupt status register ... 307

M

- MK0: Interrupt mask register ... 306
MM: Memory expansion mode register ... 346

P

- P0: Port 0 ... 60
P2: Port 2 ... 63
P3: Port 3 ... 66
P4: Port 4 ... 75
P5: Port 5 ... 80
P6: Port 6 ... 84
P7: Port 7 ... 92
P0H: Port 0 buffer register ... 97
P0L: Port 0 buffer register ... 97

PM0: Port 0 mode register ... 61
PM3: Port 3 mode register ... 72
PM5: Port 5 mode register ... 80
PM6: Port 6 mode register ... 89
PMC3: Port 3 mode control register ... 72
PR0: Priority specification flag register ... 306
PRM0: Prescaler mode register 0 ... 207
PRM1: Prescaler mode register 1 ... 143, 164
PUO: Pull-up-resistor-option register ... 65, 74, 78, 82, 91
PW: Programmable wait control register ... 347

R

RFM: Refresh mode register ... 367
RTPC: Real-time output port control register ... 97
RXB: Serial reception buffer ... 245

S

SBIC: Serial bus interface control register ... 264
SIO: Serial shift register ... 276
STBC: Standby control register ... 379

T

TM0: 16-bit timer 0 ... 111
TM1: 8-bit timer 1 ... 142
TM2: 8-bit timer 2 ... 161
TM3: 8-bit timer 3 ... 206
TMC0: Timer control register 0 ... 111, 207
TMC1: Timer control register 1 ... 143, 163
TOC: Timer output control register ... 113, 166
TXS: Serial transmission shift register ... 245

APPENDIX E INDEX

E.1 INDEX

0 parity ... 247, 248
16-bit timer 0 ... 111, 114
16-bit timer/counter ... 109
1M-byte expansion function ... 348
4-bit counter ... 251
4-bit separate real-time output port ... 97
64-pin ceramic shrink DIP with window ... 6
64-pin plastic QFP ... 8
64-pin plastic QUIP ... 6
64-pin plastic shrink DIP ... 6
68-pin plastic QFJ ... 7
74-pin plastic QFP ... 9
78K/II products ... 2
8-bit timer 1 ... 142, 145
8-bit timer 2 ... 161, 167
8-bit timer 3 ... 206, 208
8-bit timer/counter 1 ... 99, 139
8-bit timer/counter 2 ... 159
8-bit timer/counter 3 ... 205

A

Accepting maskable interrupt ... 311
Accepting nonmaskable interrupt ... 308
Accepting software interrupt ... 308
Acknowledge detection flag ... 275
Acknowledge enable bit ... 275
Acknowledge signal ... 269
Acknowledge trigger bit ... 275
Active level ... 113, 166
Address ... 272, 278
Address bus ... 29, 345
Analog delay ... 296, 389
Analog-to-digital (A/D) converter ... 225
Asynchronous serial interface ... 243
Asynchronous serial interface mode register ... 245, 247
Asynchronous serial interface operations ... 247
Asynchronous serial interface status register ... 246
Automatic addition ... 331
Auxiliary carry flag ... 46

B

Bank register ... 43
Basic operation ... 121
Baud rate ... 251, 254
Baud rate generator ... 243, 251
Baud rate generator control register ... 251
Baud rate generator for UART ... 243
Baud rate setting ... 254
Both edges ... 294
Both-edge detector ... 251
Buffer register ... 97
Built-in pull-up resistor ... 59, 65, 74, 78, 82, 91
Bus interface function ... 345
Bus release detection flag ... 274
Bus release signal ... 277
Bus release trigger bit ... 274
Bus release/command/acknowledge detector ... 261
Busy enable bit ... 275
Busy signal ... 269, 280
Busy/acknowledge output circuit ... 261

C

Capture operation ... 145, 149, 167, 177
Capture register ... 111, 142, 161
Capture trigger ... 118, 149, 177
Capture/compare control register 0 ... 112, 119
Capture/compare control register 1 ... 144
Capture/compare control register 2 ... 165, 179
Capture/compare register ... 142, 148, 149
Carry flag ... 45
Ceramic oscillation ... 55
Character bit ... 247
Clear operation ... 112, 144, 165
Clear ... 117, 148, 176, 210
Clock for baud rate generation ... 251
Clock generator ... 55
Clock pulse ... 161, 170
Clock sampling ... 297
Clock synchronous serial interface ... 259

- Clock synchronous serial interface mode register ... 262, 273
- Command ... 279
- Command detection flag ... 274
- Command signal ... 278
- Command trigger bit ... 274
- Compare operation ... 117, 148, 176, 210
- Compare register ... 111, 161, 142, 148, 206
- Controlling multiple-interrupt handling ... 301
- Conversion result ... 231
- Conversion time ... 232
- Count clock ... 114, 143, 164, 207
- Count operation ... 114, 145, 167, 208
- Count-up operation ... 114, 145, 167, 208
- Counting up ... 142, 161
- Crystal oscillation ... 55
- D**
- Data ... 279
- Data buffer area ... 329
- Data format ... 247
- Data frame ... 247
- Data macro service pointer ... 334
- Default priority ... 302
- Digital noise elimination ... 297
- Driving a transistor directly ... 62
- E**
- Edge detection ... 293, 296
- Edge detection function ... 293
- Edge detector ... 111, 142, 161, 228
- Edge to be detected ... 294
- Error flag ... 249
- Error ... 249
- Even parity ... 247
- Expansion address bus ... 43
- External baud rate input ... 253, 258
- External clock ... 55
- External clock pulse ... 170
- External event counter ... 161, 170, 201
- External event counter function ... 170
- External event counter operation mode ... 170
- External extension data memory ... 43
- External interrupt mode register ... 293, 294
- External memory ... 43
- External memory expansion mode ... 346
- External SFR area ... 43
- External trigger ... 118, 149, 177
- F**
- Framing error ... 249
- Frequency divider ... 251, 253
- Full-count ... 175
- Full-duplex transmission ... 243
- G**
- General-purpose register ... 47
- H**
- Hardware start ... 225, 235
- I**
- I/O circuit ... 33
- I/O port ... 60
- Input circuit ... 227
- Input port ... 60
- Input voltage ... 231
- Instruction ... 403
- Instructin lists for each addressing type ... 416
- Internal clock ... 262
- Internal dual-port RAM ... 43
- Internal RAM ... 43
- Internal ROM ... 37, 357
- Internal system clock ... 55
- Interrupt ... 301
- Interrupt function ... 301
- Interrupt handling ... 308
- Interrupt handling control register ... 304
- Interrupt mask flag ... 306
- Interrupt mask register ... 304, 306
- Interrupt operation timing ... 317
- Interrupt priority status flag ... 45
- Interrupt request ... 239
- Interrupt request acceptance time ... 317
- Interrupt request pending ... 316
- Interrupt request enable flag ... 46
- Interrupt request flag register ... 304, 305
- Interrupt request flag ... 305
- Interrupt request source ... 302
- interrupt service mode flag ... 306
- Interrupt service mode register ... 304, 306

-
- Interrupt status register ... 304, 307
 - Interval timer ... 129, 151, 190, 192, 210, 211
 - Interval ... 109, 139, 159, 205
 - L**
 - Local bus interface function ... 345
 - Loop counter ... 49
 - M**
 - Macro service ... 301, 319
 - Macro service channel ... 321, 326, 329
 - Macro service channel pointer ... 322
 - Macro service control register ... 322
 - Macro service control word ... 321
 - Macro service counter ... 326, 329
 - Macro service function ... 319
 - Macro service mode register ... 323
 - Macro service operation timing ... 317
 - Macro service pending ... 316
 - Macro service pointer ... 329
 - Macro service processing time ... 318
 - Macro service type ... 320
 - Macro service type A ... 323
 - Macro service type B ... 327
 - Macro service type C ... 331
 - Main data bank ... 44
 - Maskable interrupt request ... 303
 - Master ... 262
 - Master device ... 287
 - Match signal ... 117, 119, 176, 179
 - Maximum frequency ... 161, 170
 - Maximum interval ... 109, 139, 159, 205
 - Maximum pulse width ... 109, 160
 - Measurable pulse width ... 139, 160
 - Memory expansion functin ... 347
 - Memory expansion mode register ... 76, 80, 346
 - Memory map ... 38
 - Memory space ... 37
 - Minimum interval ... 109, 139, 159, 205
 - Minimum pulse width ... 109, 160, 161
 - Modulo register ... 334
 - Multiple interrupt ... 307, 313
 - Multiple-interrupt handling ... 306
 - Multiplexed address/data bus ... 345
 - Multiplexed analog ... 225
 - N**
 - Noise elimination ... 296
 - Noise eliminator ... 389
 - Nonmaskable interrupt ... 308
 - Nonmaskable interrupt request ... 303, 307
 - Number of I/O ports ... 60
 - Number of wait states ... 346
 - O**
 - Odd parity ... 247, 248
 - One-shot ... 175
 - One-shot timer ... 175, 203
 - One-shot timer function ... 175
 - One-shot timer operation ... 175
 - One-shot timer operation mode ... 175
 - Operating mode ... 399
 - Operations ... 406
 - Operations in the three-wire serial I/O mode ... 265
 - Oscillation settling time ... 58, 382
 - Output control circuit ... 111, 163
 - Output port ... 60
 - Output trigger ... 99
 - Overflow ... 114, 167
 - Overflow flag ... 112, 143, 163
 - Overrun error ... 249
 - P**
 - Parallel data ... 261
 - Parity bit ... 247
 - Parity error ... 247, 249
 - Pattern generator ... 95
 - Pending interrupt request ... 311
 - Peripheral RAM ... 43
 - Pin 25
 - Pin state ... 390
 - Pin state after reset state is released ... 390
 - Pin state during reset ... 390
 - Pointer ... 46, 49
 - Port ... 59
 - Port 0 ... 60
 - Port 0 mode register ... 61
 - Port 2 ... 63
 - Port 3 ... 66

- Port 3 mode control register ... 72
- Port 3 mode register ... 71
- Port 4 ... 75
- Port 5 ... 80
- Port 5 mode register ... 80
- Port 6 ... 84
- Port 6 mode register ... 89
- Port 7 ... 92
- Port mode ... 346, 367
- Prescaler ... 142, 163, 206
- Prescaler mode register 0 ... 207
- Prescaler mode register 1 ... 143, 164
- Priority ... 306, 308
- Priority specification flag register ... 304, 306
- Priority specification flag ... 306
- Procedure for reading ... 401
- Procedure for writing ... 399
- Program counter ... 45
- Program status word ... 45, 304, 308
- Programmable priority ... 313
- Programmable wait control register ... 347
- Programming ... 399
- Pseudo static RAM ... 367
- Pseudo static RAM refresh function ... 367
- Pull-up-resistor-option register ... 65, 74, 78, 82, 91
- Pulse refresh operation ... 368
- Pulse width ... 170
- Pulse width measurement ... 132, 155, 194
- R**
- Read timing ... 348
- Ready signal ... 280
- Real-time output function ... 95
- Real-time output port ... 95, 331
- Real-time output port control register ... 61, 97
- Receive data ... 249, 265
- Reception ... 249, 267
- Reception buffer ... 245
- Reception control parity check ... 245
- Reception error ... 246, 249
- Reference voltage pin ... 227
- Refresh bus cycle ... 367
- Refresh function ... 367
- Refresh mode register ... 367
- Refresh pulse ... 367
- Register bank selection flag ... 46
- Register ... 45
- Release detection flag ... 274
- Release trigger bit ... 274
- Releasing the busy state ... 287
- Reset ... 389
- Reset function ... 389
- Reset vector table ... 389
- Resolution ... 109, 139, 159, 160, 205
- Ring control ... 331
- Ring counter ... 334
- S**
- Sample and hold circuit ... 227
- Scan mode ... 225, 233
- Select mode ... 225, 232
- Selector ... 245
- Self-refresh ... 370
- Sending a Command ... 289
- Sending an Address ... 288
- Sending Data ... 290
- Serial bus interface mode ... 259
- Serial bus interface control register ... 263, 274
- Serial clock ... 267, 270
- Serial clock controller ... 261
- Serial clock counter ... 261
- Serial clock pin ... 270
- Serial clock selector ... 261
- Serial data ... 261, 270
- Serial data bus ... 270
- Serial data bus pin ... 270
- Serial data input ... 265
- Serial data output ... 265
- Series resistor string ... 227
- Shift ... 267
- Shift register ... 245, 261, 267, 276
- Single-chip mode ... 346
- Slave ... 262
- Slave device ... 287
- Software interrupt request ... 302
- Software start ... 225, 234

-
- Special function register ... 43, 50
 - Specifying 1M-byte expansion mode ... 346
 - Specifying the operation of the capture/compare register ... 144
 - Stack pointer ... 46
 - Standby control register ... 379
 - Standby function ... 377
 - Standby mode ... 377
 - Start bit ... 247
 - Stop bit ... 247
 - Sub-data bank ... 44
 - Successive approximation ... 225
 - System clock ... 55
 - System reset ... 389
 - T**
 - Three-wire serial I/O mode ... 259
 - Time-multiplexed address/data bus ... 29
 - Timer control register 0 ... 111, 207
 - Timer control register 1 ... 143, 163
 - Timer macro service pointer ... 334
 - Timer output mode ... 112, 165
 - Timer output control register ... 113, 119, 166, 179
 - Timer output ... 119, 179
 - Timer/counter unit ... 107
 - Timing in three-wire serial I/O mode ... 266
 - Transmission ... 248, 267
 - Transmission and reception ... 267
 - Transmission control parity generation ... 245
 - Transmission data ... 248, 265
 - Transmission shift register ... 245
 - Trigger input ... 225
 - Type A ... 320
 - Type B ... 320
 - Type C ... 321
 - U**
 - Unused pin ... 33
 - V**
 - Valid edge ... 293
 - Vector table ... 42, 302
 - Vectored interrupt ... 301, 302
 - Voltage comparator ... 227
 - W**
 - Wait function ... 357
 - Wake up ... 269, 287
 - Wake-up function ... 269
 - Write timing ... 348
 - Z**
 - Zero flag ... 46

E.2 SYMBOL INDEX

A

A ... 48
A0 ... 31
A1 ... 31
A2 ... 31
A3 ... 31
A4 ... 31
A5 ... 31
A6 ... 31
A7 ... 31
A8 ... 29, 31, 345
A9 ... 29, 31, 345
A10 ... 29, 31, 345
A11 ... 29, 31, 345
A12 ... 29, 31, 345
A13 ... 29, 31, 345
A14 ... 29, 31, 345
A15 ... 29, 345
A16 ... 30, 345
A17 ... 30, 345
A18 ... 30, 345
A19 ... 30, 345
AC ... 46, 308
ACKD ... 282
ACKE ... 281
ACKT ... 281
AD0 ... 29, 345
AD1 ... 29, 345
AD2 ... 29, 345
AD3 ... 29, 345
AD4 ... 29, 345
AD5 ... 29, 345
AD6 ... 29, 345
AD7 ... 29, 345
A/D conversion ... 234
A/D conversion result ... 230, 231, 235
A/D conversion result register ... 225
A/D conversion start signal ... 295
A/D converter ... 225
A/D converter mode register ... 228

ADCR ... 225, 227
ADM ... 228, 304
ALV0 ... 113
ALV1 ... 113
ALV2 ... 166
ALV3 ... 166
AN0 ... 225
AN1 ... 225
AN2 ... 225
AN3 ... 225
AN4 ... 225
AN5 ... 225
AN6 ... 30, 225, 239
AN7 ... 30, 225, 239
ANI0 ... 229
ANI1 ... 229
ANI2 ... 229
ASCK ... 28, 253, 258
ASIM ... 245, 247
ASIS ... 246
ASTB ... 31, 345
AV_{REF} ... 31, 227
AV_{SS} ... 31, 227
AX ... 48

B

B ... 48
Baud rate generator for UART ... 243
BC ... 48
BRGC ... 251
BRK ... 302
BSYE ... 283
BYTE ... 97

C

C ... 48
CALLF instruction entry ... 42
CALLT instruction table ... 42
CE ... 252
 \overline{CE} ... 31
CE0 ... 112, 114
CE1 ... 143

CE2 ... 163, 167
CE3 ... 207, 208
CHT0 ... 323
CHT1 ... 323
CHT2 ... 323
CI ... 28, 161, 170
CIF00 ... 305
CIF01 ... 305
CIF10 ... 305
CIF11 ... 305
CIF20 ... 305
CIF21 ... 305
CISM00 ... 306
CISM01 ... 306
CISM10 ... 306
CISM11 ... 306
CISM20 ... 306
CISM21 ... 306
CL ... 246
CLR01 ... 112
CLR10 ... 144
CLR11 ... 144
CLR21 ... 165
CLR22 ... 165
CLS0 ... 262, 273
CLS1 ... 262, 273
CM ... 144
CMD2 ... 163
CMDD ... 280
CMDT ... 280
CMK00 ... 306
CMK01 ... 306
CMK10 ... 306
CMK11 ... 306
CMK20 ... 306
CMK21 ... 306
CPR00 ... 307
CPR01 ... 307
CPR10 ... 307
CPR11 ... 307
CPR20 ... 307
CPR21 ... 307
CR00 ... 111, 117
CR01 ... 111, 117
CR02 ... 111, 118
CR10 ... 142, 148
CR11 ... 142, 148
CR20 ... 161, 176
CR21 ... 161, 176
CR22 ... 161, 177
CR30 ... 206, 210
CRC0 ... 112, 119
CRC1 ... 144
CRC2 ... 165, 179
CRXE ... 262, 273
CS ... 229, 304
CSIIF ... 305
CSIISM ... 306
CSIM ... 262, 273
CSIMK ... 306
CSIPR ... 307
CTXE ... 262, 273
CY ... 45, 308
D
D ... 48
D0 ... 31
D1 ... 31
D2 ... 31
D3 ... 31
D4 ... 31
D5 ... 31
D6 ... 31
D7 ... 31
DE ... 48
DI ... 308
Driving LED directly ... 79, 83
E
E ... 48
 \overline{EA} ... 31, 37
EI ... 308
ENTO0 ... 113
ENTO1 ... 113
ENTO2 ... 166
ENTO3 ... 166

ES00 ... 294
ES01 ... 294
ES10 ... 294
ES11 ... 294
ES20 ... 294
ES21 ... 294
ES30 ... 295
ES31 ... 295
ES40 ... 295, 303
ES41 ... 295, 303
ES50 ... 295
ES51 ... 295
ESNMI ... 294
EXTR ... 97
F
f_{CLK} ... 55
FE ... 247
FR ... 229
H
H ... 48
HALT mode ... 377, 379
HL ... 48
HLT ... 379
I
IE ... 46, 308
IE flag ... 308
IF0 ... 304, 305
IF0H ... 305
IF0L ... 305
IFCH ... 346
INTAD ... 239, 302, 304
INTC00 ... 117, 302
INTC01 ... 117, 302
INTC10 ... 99, 148, 302, 331
INTC11 ... 99, 148, 302, 331
INTC20 ... 176, 302
INTC21 ... 176, 302
INTC30 ... 210, 302
INTCSI ... 302
Interrupt request from the A/D converter ... 239
INTM0 ... 293
INTM1 ... 293
INTP0 ... 28, 99, 149, 294, 302
INTP1 ... 28, 177, 294, 302
INTP2 ... 28, 161, 294, 302
INTP3 ... 28, 118, 295, 302
INTP4 ... 28, 295, 302, 303
INTP5 ... 28, 225, 239, 295, 302, 303, 304
INTSER ... 249, 302
INTSR ... 249, 302
INTST ... 248, 302
IRAM ... 43
ISM0 ... 304, 306
ISM0H ... 306
ISM0L ... 306
ISP ... 45, 308
ISP flag ... 311
IST ... 304, 307
L
L ... 48
M
MDL0 ... 252
MDL1 ... 252
MDL2 ... 252
MDL3 ... 252
MK0 ... 304, 306
MK0H ... 306
MK0L ... 306
MM ... 346
MM0 ... 346
MM1 ... 346
MM2 ... 346
MM6 ... 346
MOD0 ... 112, 165, 323
MOD1 ... 112, 165, 262, 273, 323
MOD2 ... 323
MOD3 ... 323
MP ... 329
MPD ... 334
MPT ... 334
MR ... 334
MS ... 229, 304
MSB first ... 265
MSC ... 326

-
- μ PD78210 ... 20
 - μ PD78212 ... 1
 - μ PD78213 ... 1
 - μ PD78214 ... 1
 - μ PD78p214 ... 1, 399
 - N**
 - NC ... 31, 32
 - NMI ... 428, 302, 303
 - NMIS ... 307
 - O**
 - \overline{OE} ... 31
 - OVE ... 247
 - OVF0 ... 112
 - OVF1 ... 143
 - OVF2 ... 163
 - P**
 - POH ... 97
 - POL ... 97
 - POHM ... 61
 - POLM ... 61
 - POMH ... 97
 - POML ... 97
 - P20 ... 294
 - P20/NMI ... 31
 - P21 ... 294
 - P22 ... 294
 - P23 ... 294
 - P24 ... 295
 - P25 ... 295
 - P26 ... 295
 - PC ... 45
 - PE ... 247
 - PIF0 ... 305
 - PIF1 ... 305
 - PIF2 ... 305
 - PIF3 ... 305
 - PIF4 ... 305
 - PIF5 ... 305
 - PISM0 ... 306
 - PISM1 ... 306
 - PISM2 ... 306
 - PISM3 ... 306
 - PISM4 ... 306
 - PISM5 ... 306
 - PM0 ... 61
 - PM3 ... 72
 - PM5 ... 81
 - PM6 ... 89
 - PMC3 ... 72
 - PMK0 ... 306
 - PMK1 ... 306
 - PMK2 ... 306
 - PMK3 ... 306
 - PMK4 ... 306
 - PMK5 ... 306
 - Port 0 ... 60
 - Port 2 ... 63
 - Port 3 ... 66
 - Port 4 ... 75
 - Port 5 ... 80
 - Port 6 ... 84
 - Port 7 ... 92
 - PPG frequency ... 125, 186
 - PPG output ... 125, 185
 - PPG period ... 125, 185
 - PPG pulse width ... 125, 185
 - PPR0 ... 307
 - PPR1 ... 307
 - PPR2 ... 307
 - PPR3 ... 307
 - PPR4 ... 307
 - PPR5 ... 307
 - PR0 ... 304, 307
 - PROH ... 307
 - PROL ... 307
 - PRAM ... 43
 - PRM0 ... 207
 - PRM1 ... 143, 164
 - Procedure for reading from PROM ... 401
 - Procedure for writing into PROM ... 399
 - PROM ... 399
 - PROM programming mode ... 27, 399
 - PRS0 ... 207
 - PRS1 ... 207

PRS2 ... 207
PRS3 ... 207
PRS10 ... 144
PRS11 ... 144
PRS12 ... 144
PRS20 ... 164
PRS21 ... 164
PRS22 ... 164
PRS23 ... 164
PS0 ... 246
PS1 ... 246
PSW ... 45, 304, 308
PUO ... 65, 74, 78, 82, 91
PUO2 ... 65
PUO3 ... 74
PUO4 ... 78
PUO5 ... 82
PUO6 ... 91
PW ... 347
PW20 ... 346
PW21 ... 346
PW30 ... 347
PW31 ... 347
PWM frequency ... 122, 183
PWM output ... 122, 182
PWM period ... 122, 182
PWM pulse width ... 122, 182
PWM signal ... 122, 182

R
R0 ... 49
R1 ... 49
R2 ... 49
R3 ... 49
R4 ... 49
R5 ... 49
R6 ... 49
R7 ... 49
RBS0 ... 46, 308
RBS1 ... 46, 308
RC ... 334
 \overline{RD} ... 30, 345
 \overline{REFRQ} ... 30

RELD ... 280
Releasing HALT mode ... 485
Releasing STOP mode ... 382
RELT ... 280
 \overline{RESET} ... 31, 389
RETB ... 308
RETB instruction ... 308
RETI ... 308
RETI instruction ... 308
RFEN ... 367
RFLV ... 367
RFM ... 367
RFT0 ... 367
RFT1 ... 367
ROM less ... 42
RP0 ... 49
RP1 ... 49
RP2 ... 49
RP3 ... 49
RTPC ... 61, 97
RXB ... 245
RxD ... 29
RXE ... 246

S
SAR ... 227
SB0 ... 270
SBI ... 272
SBIC ... 263
SBI mode ... 259, 268
SCK ... 246
 \overline{SCK} ... 267, 270
 \overline{SCK} pin ... 267
SERIF ... 305
SERMK ... 306
SERPR ... 307
SFR ... 43, 50
SFR pointer ... 329
SFRP ... 329
SI ... 28, 265
SI pin ... 267
SIO ... 261, 267, 276
SL ... 246

SO ... 29, 265
SO pin ... 267
SO latch ... 261
SP ... 46
Specifying HALT mode ... 379
Specifying STOP mode ... 382
SRIF ... 305
SRISM ... 306
SRMK ... 306
SRPR ... 307
STBC ... 379
STIF ... 305
STISM ... 306
STMK ... 306
STOP mode ... 377, 382
STP ... 379
STPR ... 307

T

TM0 ... 111, 114
TM1 ... 142, 145
TM2 ... 161, 167
TM3 ... 206, 208
TMC0 ... 111, 207
TMC1 ... 143, 163
TO0 ... 29, 119
TO1 ... 29, 119
TO2 ... 29, 179
TO3 ... 29, 179
TOC ... 113, 119, 166, 179
TPS0 ... 252
TPS1 ... 252
TPS2 ... 252
TRG ... 229, 304
TxD ... 29
TXS ... 245

V

V_{DD} ... 31
V_{PP} ... 31
V_{SS} ... 31, 32

W

$\overline{\text{WAIT}}$... 30
 $\overline{\text{WR}}$... 30, 345

WUP ... 262, 273

X

X ... 48
X1 ... 31, 55
X2 ... 31, 55

Z

Z ... 46, 308

