To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

RENESAS

USER'S MANUAL

RENESAS

μPD78234 SUBSERIES

8-BIT SINGLE CHIP MICROCOMPUTER

HARDWARE

μPD78233
μPD78234
μPD78237
μPD78238
μPD78P238
μPD78234 (A)
μPD78238 (A)

Document No. IEU-1290H
(O. D. No. IEU-718H)
Date Published January 1995 P
Printed in Japan

GENERAL NOTES ON CMOS DEVICES

① **STATIC ELECTRICITY (ALL MOS DEVICES)**

Exercise care so that MOS devices are not adversely influenced by static electricity while being handled.

The insulation of the gates of the MOS device may be destroyed by a strong static charge. Therefore, when transporting or storing the MOS device, use a conductive tray, magazine case, or conductive buffer materials, or the metal case NEC uses for packaging and shipment, and use grounding when assembling the MOS device system. Do not leave the MOS device on a plastic plate and do not touch the pins of the device.

Handle boards on which MOS devices are mounted similarly.

② **PROCESSING OF UNUSED PINS (CMOS DEVICES ONLY)**

Fix the input level of CMOS devices.

Unlike bipolar or NMOS devices, if a CMOS device is operated with nothing connected to its input pin, intermediate level input may be generated due to noise, and an inrush current may flow through the device, causing the device to malfunction. Therefore, fix the input level of the device by using a pull-down or pull-up resistor. If there is a possibility that an unused pin serves as an output pin (whose timing is not specified), each pin should be connected to V_{DD} or GND through a resistor.

Refer to "Processing of Unused Pins" in the documents of each devices.

③ **STATUS BEFORE INITIALIZATION (ALL MOS DEVICES)**

The initial status of MOS devices is undefined upon power application.

Since the characteristics of an MOS device are determined by the quantity Since the characteristics of an MOS device are determined by the quantity of injection at the molecular level, the initial status of the device is not controlled during the production process. The output status of pins, I/O setting, and register contents upon power application are not guaranteed. However, the items defined for reset operation and mode setting are subject to guarantee after the respective operations have been executed.

When using a device with a reset function, be sure to reset the device after power application.

M7 92.6

Contents Updated in This Edition

Page	Contents
P. 364	Fig. 12-1 Clock-Synchronized Serial Interface Configuration has been modified.
P. 524	Some part of the description of (5) A/D Converter in 16.4.3 Notes on using STOP mode in CHAPTER 6 STANDBY FUNCTIONS has been deleted. An example of Processing of Analog Input Pin has been deleted.
P. 527, 528	Some explanation has been added to Fig. 17-1 Accepting RESET Signal and Fig. 17-2 Reset Operation on Power Application in CHAPTER 17 RESET FUNCTION.
 P. 574-577 P. 577	APPENDIX B DEVELOPMENT TOOL • Descriptions concerning 3.5" 2HC has been added to the supply media and part number for IBM PC/AT. • B.2.4 OS for IBM PC has been added.

The mark ★ shows major revised points.

INTRODUCTION

Target reader This manual is prepared for engineers who understand the functions of μPD78234 sub-series, and try to design the application systems of the microcomputers.

Objectives This manual describes the functions of the internal hardware devices and instructions of μPD78234 sub-series.

Outline Two manuals are available for μPD78234 sub-series: the hardware Manual (this manual) and the Instruction Manual (which can be commonly used with the 78K/II series products). The contents of these manuals are:

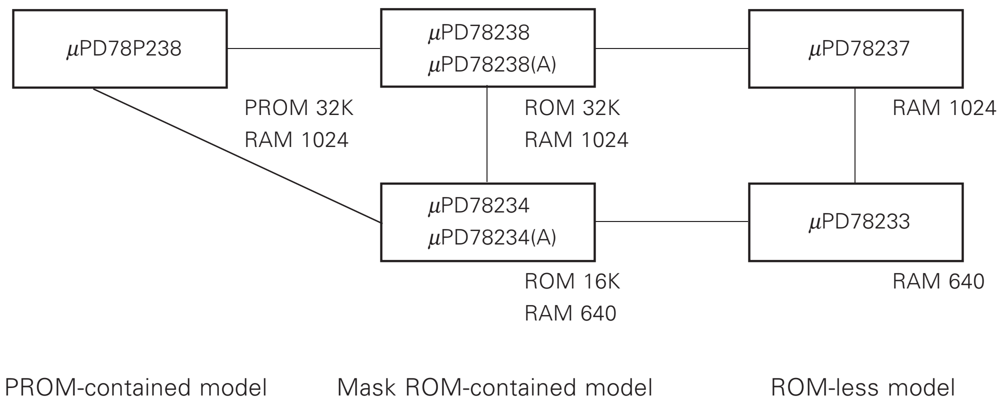
Hardware	Instruction
Pin functions	CPU functions
Internal block functions	Addressing
Interrupt	Instruction set
Other internal peripherals	

The important information in using the products described in this manual are provided in the form of "Caution" in appropriate places in each Chapter of this manual. This information is repeated at the end of the Chapter. Be sure to read it before using the products.

How to read this manual

Readers of this manual must have general knowledge on electric engineering, logic circuits, and microcomputers.

This manual describes the functions for μPD78233, 78234, 78237, 78238, 78P238, 78234(A), and 78238(A). The relations among these products are shown in the next figure.



Typical circuit examples described in this manual are designed for "Standard" quality standard products for general electronics devices. When using the circuits for applications requiring "Special" quality standard, the quality standard for each component and circuit actually used must be confirmed before using.

- When there are no functional differences among the products
  μPD78234 is described as the representative product.

- When there are functional differences
  The functions of an individual product are described.
  However, if the functional difference is whether ROM is provided or not, μPD78233 is described as the representative ROM-less model.

For the detailed description of a register whose name is known, refer to **APPENDIX D REGISTER INDEX**.

For the differences between μPD78234 and the other models in the 78K/II series, first refer to **APPENDIX A 78K/II SERIES PRODUCT LIST** to learn what kinds of differences exist, and then refer to **APPENDIX E GENERAL INDEX** for details.

For the detailed description of a function whose name is known, refer to **APPENDIX E GENERAL INDEX**.

If μPD78234 does not operate correctly, while the microcomputer or its software is being debugged, refer to the end of each chapter where notes on each function are presented.

To understand the functions of μPD78234 sub-series read the manual according to the CONTENTS.

For a detailed description on instruction functions, refer to the **78K/II SERIES USER'S MANUAL, INSTRUCTIONS (IEU-1311)** separately available.

Refer to the data sheet provided with this manual as an appendix for the electrical characteristics of the μPD78234 sub-series.

Refer to the application note provided with this manual as an appendix for application examples of each function of the μPD78234 sub-series.

Legend

Data significance	: Data on the left has higher significance, while data on the right has lower significance
Active low signal	: $\overline{\text{xxx}}$ (top bar on pin name or signal name)
*	: Footnote
Caution	: Information calling for your particular attention
Remarks	: Supplemental information
Number representation	: Binary: ×××× or ××××B Decimal: ×××× Hexadecimal: ××××H

Related documents Refer to the following documents as well as this manual.

- **Documents related to μPD78234 sub-series**

Document \ Product		μPD78233 μPD78234 μPD78237 μPD78238	μPD78P238	μPD78234(A) μPD78238(A)
Data sheet		IC-2476	IC-2607	IC-2984
User's manual	Hardware	This manual		
	Instruction	IEU-1311		
Application note	Application	IEA-1280		
	Floating point operation program	IEA-1273		

- **Serial bus interface (SBI) user's manual**
- **Documents related to development tools**

Document		Document no.
IE-78230-R in-circuit emulator user's manual	Hardware	EEU-1327
	Software	EEU-1296
IE-78230-R-A in-circuit emulator user's manual		EEU-1392
RA78K series assembler package user's manual	Language	EEU-1404
	Operation	EEU-1399
78K series structured assembler preprocessor user's manual		EEU-1402
CC78K series C compiler user's manual	Language	EEU-1284
	Operation	EEU-1280
SD78K/II screen debugger user's manual MS-DOS base	Beginner's guide	EEU-1447
	Reference	EEU-1413

• **Documents related to software for embedded applications**

Document		Document no.
78K/0, 78K/II, 87AD series fuzzy inference development support system user's manual	Translator	EEU-1444

• **Other related documents**

Document	Document no.
Package manual	IEI-1213
Semiconductor device mounting technology manual	IEI-1207
Quality grade on NEC semiconductor devices	IEI-1209
Static electricity discharge (ESD) test	IEI-1201

Caution: Be sure to use the latest document for designing.

Register representation:

	7	6	5	4	③	2	1	0
EDC	B	1	0	×	A	1	0	×

Encircled bit numbers are used as reserved word by NEC assembler and defined by sfrbit.h by C compiler.

Bit	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.
0	Write 0 to this bit.	
1	Write 1 to this bit.	
A	Write a value according to the necessary function.	A value according to the operation is read.

Never use the combination of codes indicated "Inhibited" in the descriptions of registers.

Confusing characters : 0 (zero), O (uppercase "O")
: 1 (one), l (lowercase "L"),
I (uppercase "I")

CONTENTS

FIGURE (1/10)

FIGURE (2/10)

FIGURE (3/10)

FIGURE (4/10)

FIGURE (5/10)

FIGURE (6/10)

FIGURE (7/10)

FIGURE (8/10)

FIGURE (9/10)

FIGURE (10/10)

TABLE (1/3)

TABLE (2/3)

TABLE (3/3)

CHAPTER 1 GENERAL

μPD78234 is a member of the NEC's 78K/II series products and can directly input/output analog signals. The 78K/II series consists of 8-bit single-chip microcomputers, each equipped with a high-performance CPU having functions such as the one to access a 1M-byte data memory space.

μPD78234 is provided with internal 16K-byte mask ROM and 640-byte RAM, as well as high-performance timer/counter, 8-bit A/D converter, 8-bit D/A converter, PWM output function, and two channels of serial interfaces independent from each other.

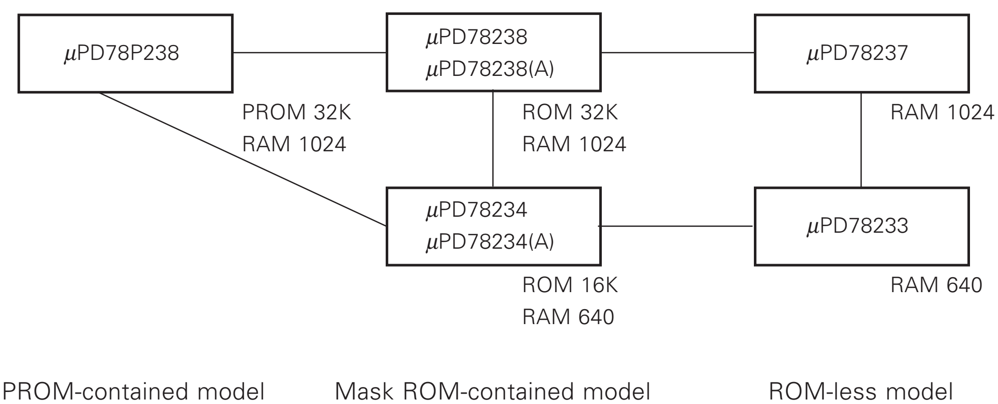
μPD78233 has the same functions as those for μPD78234, except the mask ROM.

μPD78238 is provided with a 32K-byte mask ROM and 1024-byte RAM, expanded from those for μPD78234.

μPD78237 has the same functions as those for μPD78238, except the mask ROM.

μPD78P238 has a PROM in place of the mask ROM in the μPD78238.

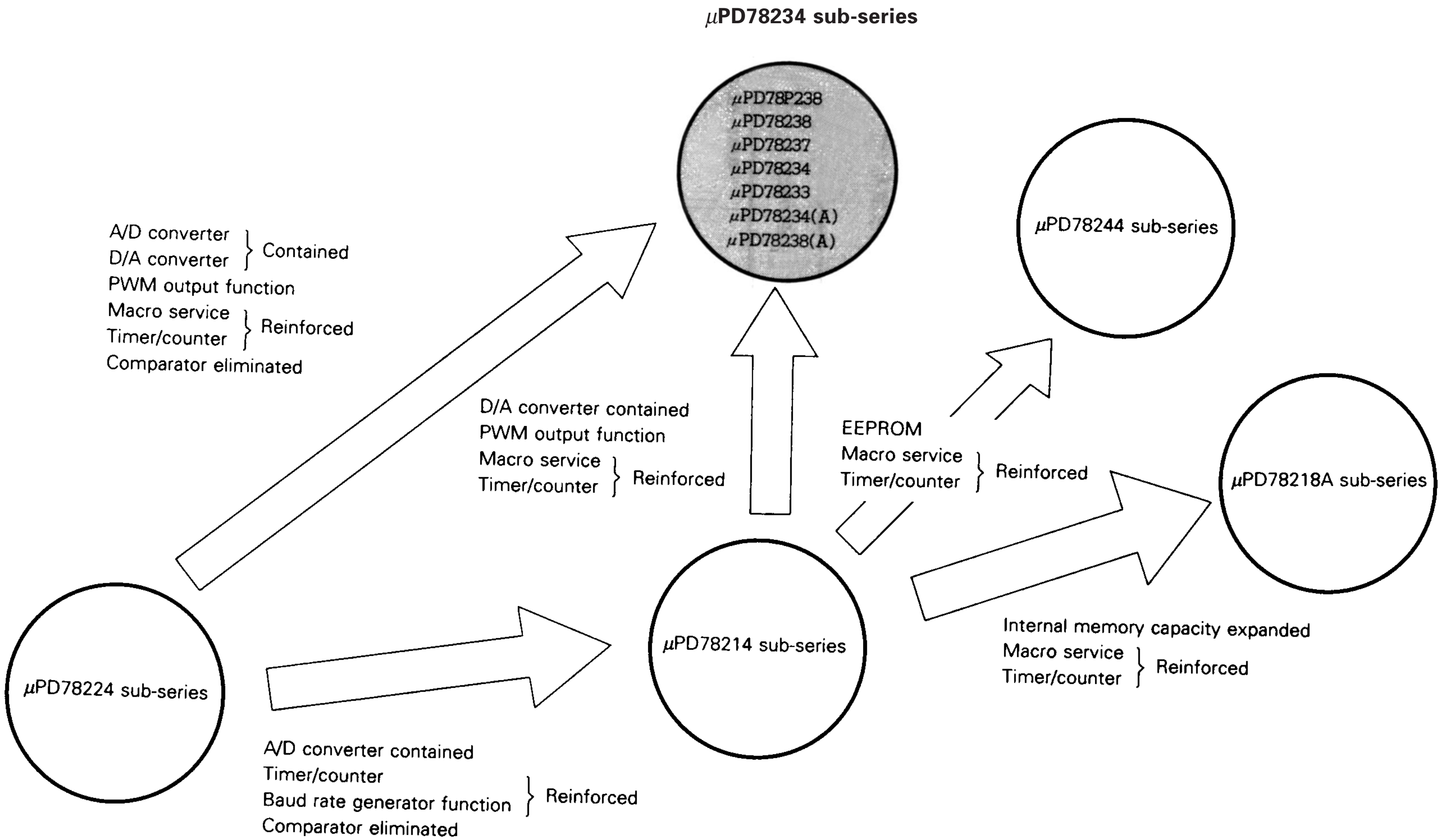
The μPD78234(A) and the μPD78238(A) are "Special" quality standard versions of the μPD78234 and μPD78238, respectively.



The application fields of these products are as follows:

- Standard products
  - Printer
  - Electronic typewriter
  - ECR (Electronic Cash Register)
  - PPC (Plain Paper Copier)
  - FDD (Floppy Disk Drive)
  - HDD (Hard Disk Drive)
  - Electronic musical instrument
  - Air conditioner (household appliance)
  - Automobile telephone (communication)
  - Camera
- Special products
  - Automotive electronics
  - Combustion control

78K/II Products



1.1 Features

- 78K/II series
- Multiplexed internal bus (high instruction execution speed)
  - Minimum instruction cycle (at 12 MHz) : 333 ns (μPD78234, 78238, 78P238)
500 ns (μPD78233, 78237)
- Instruction set ideal for control applications
- Data memory expansion function (1M-byte memory space: bank select pointer x 2)
- Interrupt controller (2-level priority)
  - Vector interrupt processing
  - Macro service
- Internal memory
  - ROM
Mask ROM : 16K bytes (μPD78234)
32K bytes (μPD78238)
No internal memory (μPD78233, 78237)
PROM : 32K bytes (μPD78P238)
  - RAM : 640 bytes (μPD78233, 78234)
1024 bytes (μPD78237, 78238, 78P238)
- I/O pins : 64 (μPD78234, 78238, 78P238)
46 (μPD78233, 78237)
  - Software programmable pull-up : 42 inputs (μPD78234, 78238, 78P238)
24 inputs (μPD78233, 78237)
  - LED direct drive pins : 24 outputs (μPD78234, 78238, 78P238)
8 outputs (μPD78233, 78237)
  - Transistor direct drive pins : 8 outputs
- Serial interface
  - UART (with baud rate generator)
  - Clock-synchronized serial interface (3-line serial I/O, serial bus interface)
- Real-time output port (two stepping motors can be controlled independently from each other when the output port is used in combination with an 8-bit timer/counter)
- 8-bit A/D converter (8 analog inputs)
- 8-bit D/A converter (2 analog outputs)
- 12-bit PWM output (2 outputs)
- High-performance timer/counter unit
  - 16 bits × 1
  - 8 bits × 3

1.2 Ordering Information and Quality Grade

1.2.1 Ordering information

Ordering code	Package	Internal ROM
μPD78233GC-3B9	80-pin plastic QFP (14 × 14 mm)	None
μPD78233GJ-5BG	94-pin plastic QFP (20 × 20 mm)	None
μPD78233LQ	84-pin plastic QFJ (□1150 mil)	None
μPD78234GC-xxx-3B9	80-pin plastic QFP (14 × 14 mm)	Mask ROM
μPD78234GJ-xxx-5BG	94-pin plastic QFP (20 × 20 mm)	Mask ROM
μPD78234LQ-xxx	84-pin plastic QFJ (□1150 mil)	Mask ROM
μPD78237GC-3B9	80-pin plastic QFP (14 × 14 mm)	None
μPD78237GJ-5BG	94-pin plastic QFP (20 × 20 mm)	None
μPD78237LQ	84-pin plastic QFJ (□1150 mil)	None
μPD78238GC-xxx-3B9	80-pin plastic QFP (14 × 14 mm)	Mask ROM
μPD78238GJ-xxx-5BG	94-pin plastic QFP (20 × 20 mm)	Mask ROM
μPD78238LQ-xxx	84-pin plastic QFJ (□1150 mil)	Mask ROM
μPD78P238GC-3B9	80-pin plastic QFP (14 × 14 mm)	One-time PROM
μPD78P238GJ-5BG	94-pin plastic QFP (20 × 20 mm)	One-time PROM
μPD78P238LQ	84-pin plastic QFJ (□1150 mil)	One-time PROM
μPD78P238KF*1	94-pin ceramic WQFN	EPROM
μPD78P238GC-xxx-3B9*2	80-pin plastic QFP (14 × 14 mm)	Written One-time PROM
μPD78P238GJ-xxx-5BG*2	94-pin plastic QFP (20 × 20 mm)	Written One-time PROM
μPD78P238LQ-xxx*2	84-pin plastic QFJ (□1150 mil)	Written One-time PROM
μPD78234GC(A)-xxx-3B9	80-pin plastic QFP (14 × 14 mm)	Mask ROM
μPD78234GJ(A)-xxx-5BG	94-pin plastic QFP (20 × 20 mm)	Mask ROM
μPD78238GC(A)-xxx-3B9	80-pin plastic QFP (14 × 14 mm)	Mask ROM

***1:** can be mounted on a PC board designed for the 94-pin plastic QFP in combination with EV-9200G-94 (socket).

***2:** This is a QTOP microcomputer.
A QTOP microcomputer is a single-chip microcomputer with one-time PROM for which program writing, marking, screening, and verifying is completely supported by NEC.

Remarks: xxx indicates a ROM code number.

1.2.2 Quality grade

Ordering code	Package	Quality grade
μPD78233GC-3B9	80-pin plastic QFP (14 × 14 mm)	Standard
μPD78233GJ-5BG	94-pin plastic QFP (20 × 20 mm)	Standard
μPD78233LQ	84-pin plastic QFJ (□1150 mil)	Standard
μPD78234GC-xxx-3B9	80-pin plastic QFP (14 × 14 mm)	Standard
μPD78234GJ-xxx-5BG	94-pin plastic QFP (20 × 20 mm)	Standard
μPD78234LQ-xxx	84-pin plastic QFJ (□1150 mil)	Standard
μPD78237GC-3B9	80-pin plastic QFP (14 × 14 mm)	Standard
μPD78237GJ-5BG	94-pin plastic QFP (20 × 20 mm)	Standard
μPD78237LQ	84-pin plastic QFJ (□1150 mil)	Standard
μPD78238GC-xxx-3B9	80-pin plastic QFP (14 × 14 mm)	Standard
μPD78238GJ-xxx-5BG	94-pin plastic QFP (20 × 20 mm)	Standard
μPD78238LQ-xxx	84-pin plastic QFJ (□1150 mil)	Standard
μPD78P238GC-3B9	80-pin plastic QFP (14 × 14 mm)	Standard
μPD78P238GJ-5BG	94-pin plastic QFP (20 × 20 mm)	Standard
μPD78P238LQ	84-pin plastic QFJ (□1150 mil)	Standard
μPD78P238KF*1	94-pin ceramic WQFN	Standard
μPD78P238GC-xxx-3B9*2	80-pin plastic QFP (14 × 14 mm)	Standard
μPD78P238GJ-xxx-5BG*2	94-pin plastic QFP (20 × 20 mm)	Standard
μPD78P238LQ-xxx*2	84-pin plastic QFJ (□1150 mil)	Standard
μPD78234GC(A)-xxx-3B9	80-pin plastic QFP (14 × 14 mm)	Special
μPD78234GJ(A)-xxx-5BG	94-pin plastic QFP (20 × 20 mm)	Special
μPD78238GC(A)-xxx-3B9	80-pin plastic QFP (14 × 14 mm)	Special

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

***1:** Can be mounted on a printed circuit board designed for the 94-pin plastic QFP, in combination with EV-9200G-94 (socket).

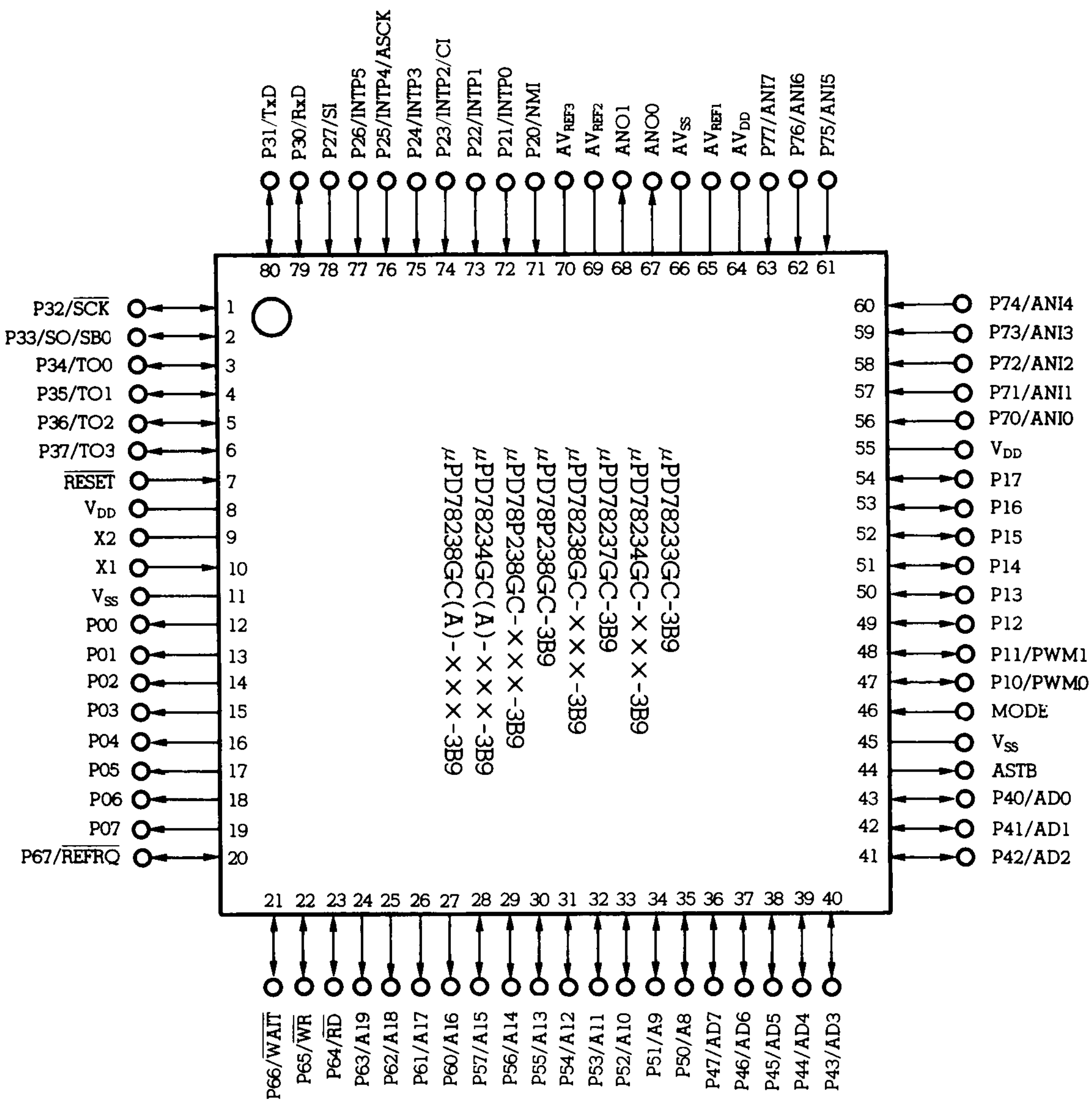
***2:** This is a QTOP microcomputer.
A QTOP microcomputer is a single-chip microcomputer with one-time PROM for which program writing, marking, screening, and verifying is completely supported by NEC.

Remarks: xxx indicates a ROM code number.

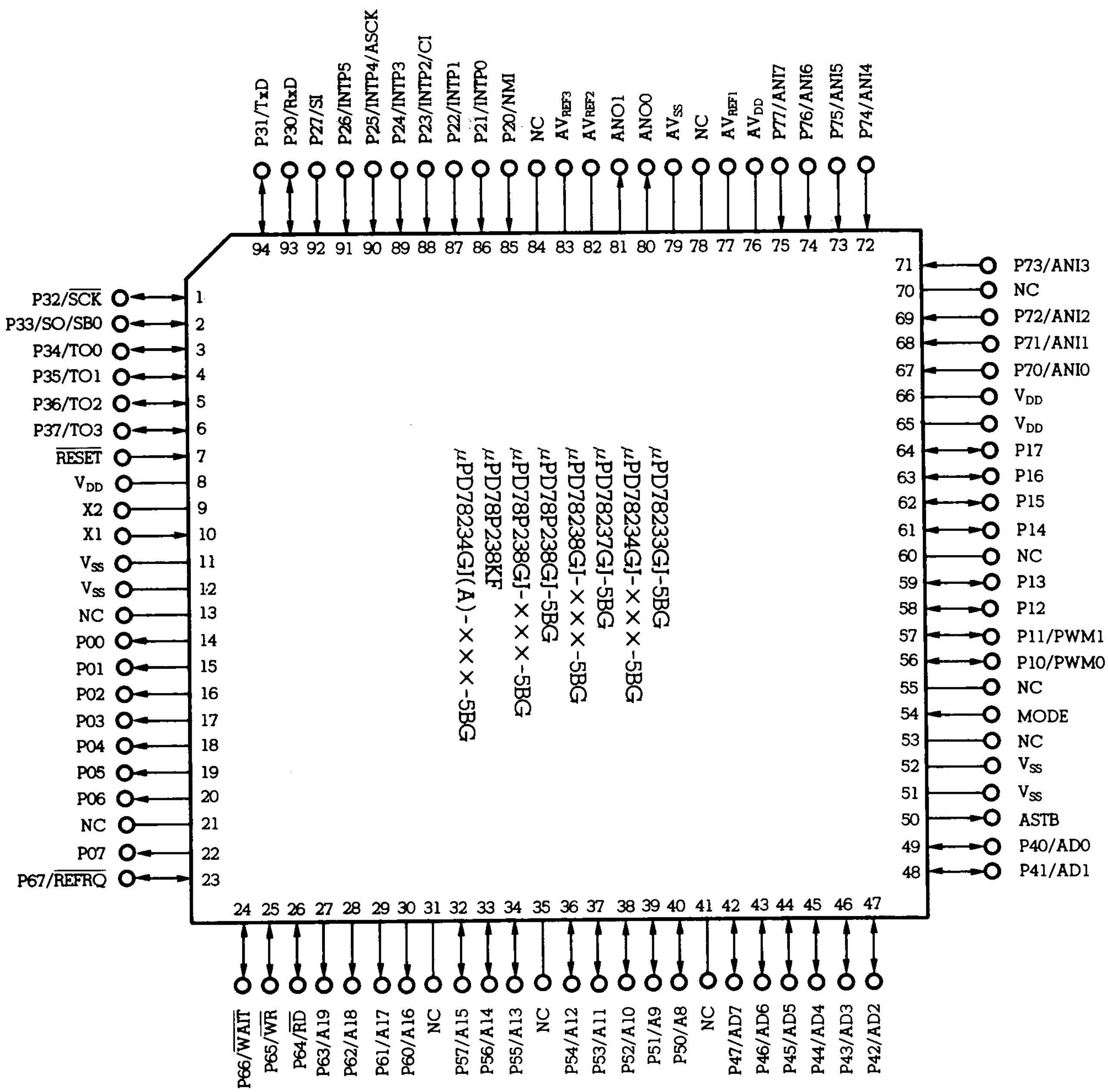
1.3 Pin Configuration (Top View)

1.3.1 Ordinary operation mode

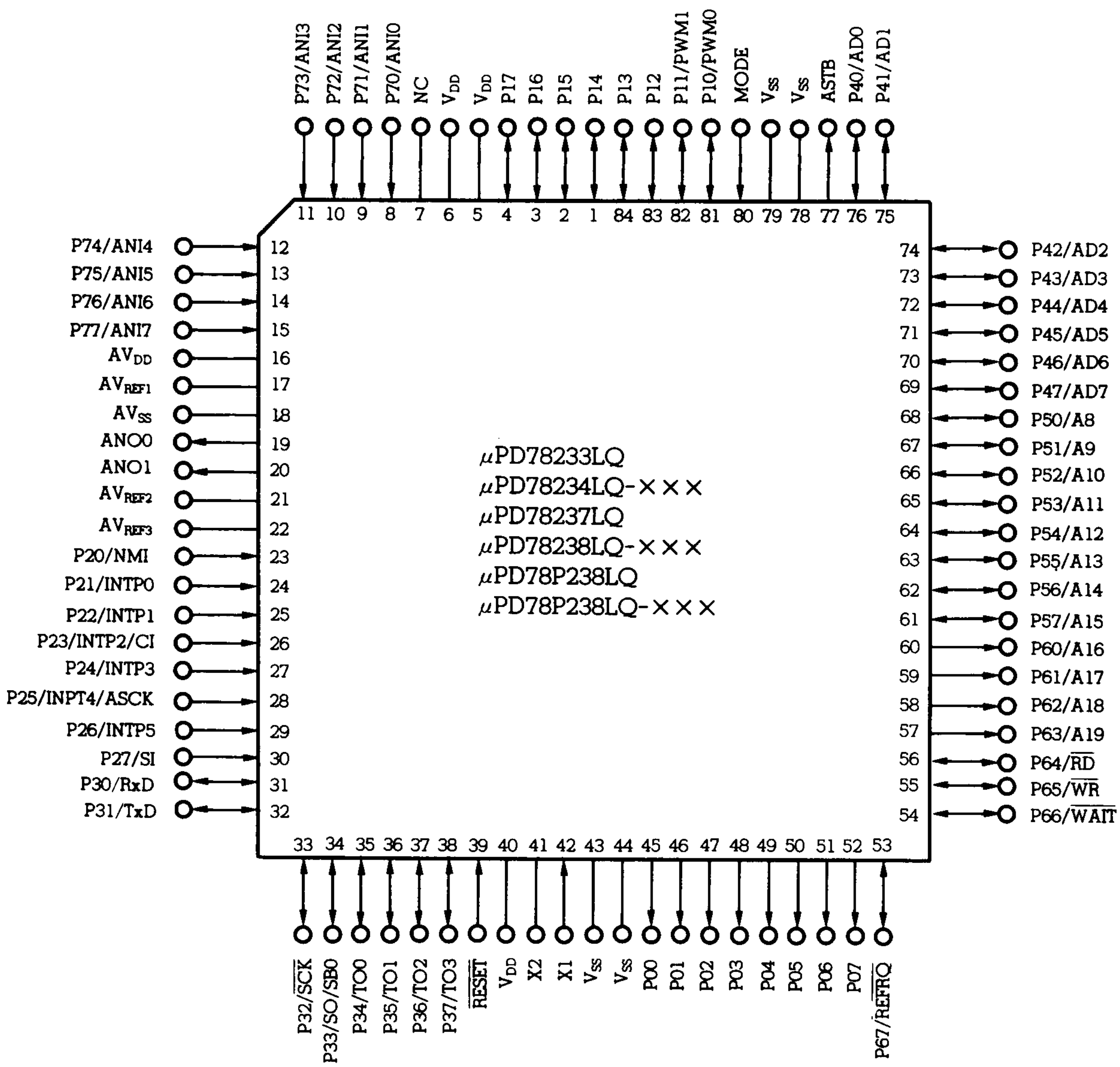
(1) 80-pin plastic QFP (chip size: 14 × 14 mm)



(2) 94-pin plastic QFP, 94-pin ceramic WQFN (chip size: 20 × 20 mm)



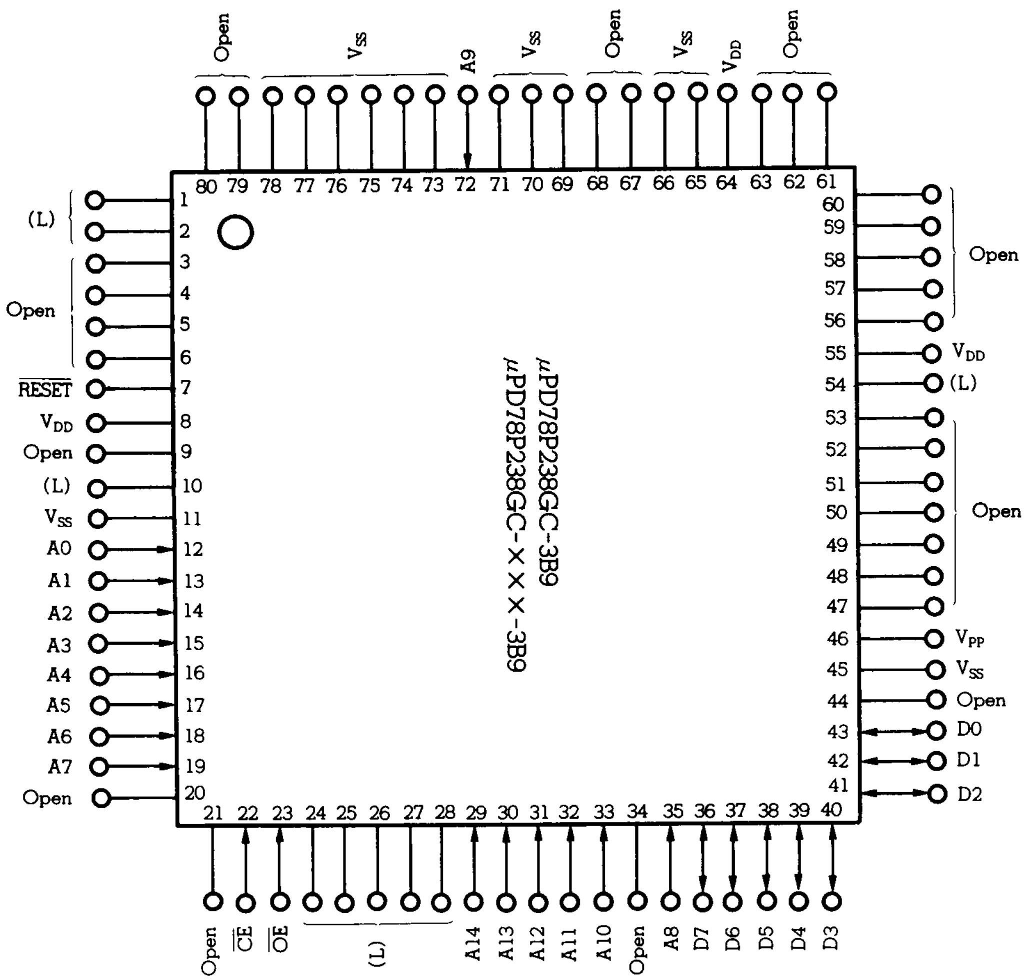
(3) 84-pin plastic QFJ (□1150 mil)



P00-P07 : Port 0
P10-P17 : Port 1
P20-P27 : Port 2
P30-P37 : Port 3
P40-P47 : Port 4
P50-P57 : Port 5
P60-P67 : Port 6
P70-P77 : Port 7
TO0-TO3 : Timer Output
CI : Clock Input
RxD : Receive Data
TxD : Transmit Data
$\overline{\text{SCK}}$: Serial Clock
ASCK : Asynchronous Serial Clock
SB0 : Serial Bus
SI : Serial Input
SO : Serial Output
PWM0, PWM1 : Pulse Width Modulation 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
MODE : Mode
X1, X2 : Crystal
ANI0-ANI7 : Analog Input
ANO0, ANO1 : Analog Output
AV_{REF1}-AV_{REF3} : Reference Voltage
AV_{DD} : Analog Power Supply
AV_{SS} : Analog Ground
V_{DD} : Power Supply
V_{SS} : Ground
NC : Non-connection

1.3.2 PROM programming mode ($V_{PP} \geqq 5$ V, $\overline{\text{RESET}}$ = L)

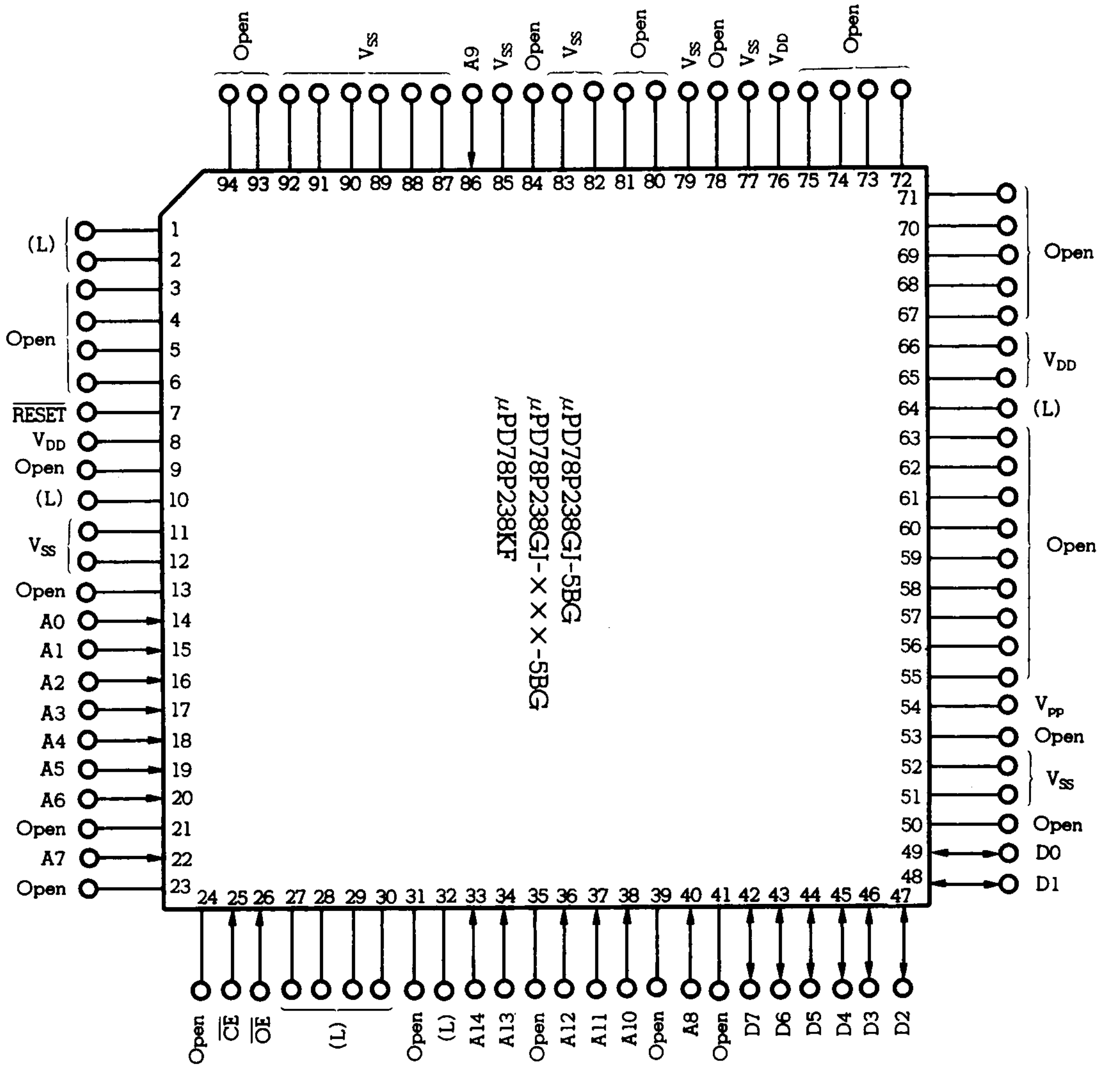
(1) 80-pin plastic QFP (chip size: 14 × 14 mm)



Caution:

L	:	Connect each of these pins to V_{SS} via a 10kΩ pull-down resistor.
V_{SS}	:	Ground these pins.
Open	:	Do not connect these pins.
$\overline{\text{RESET}}$	:	Keep this pin at low level.

(2) 94-pin plastic QFP, 94-pin ceramic WQFN (chip size: 20 × 20 mm)



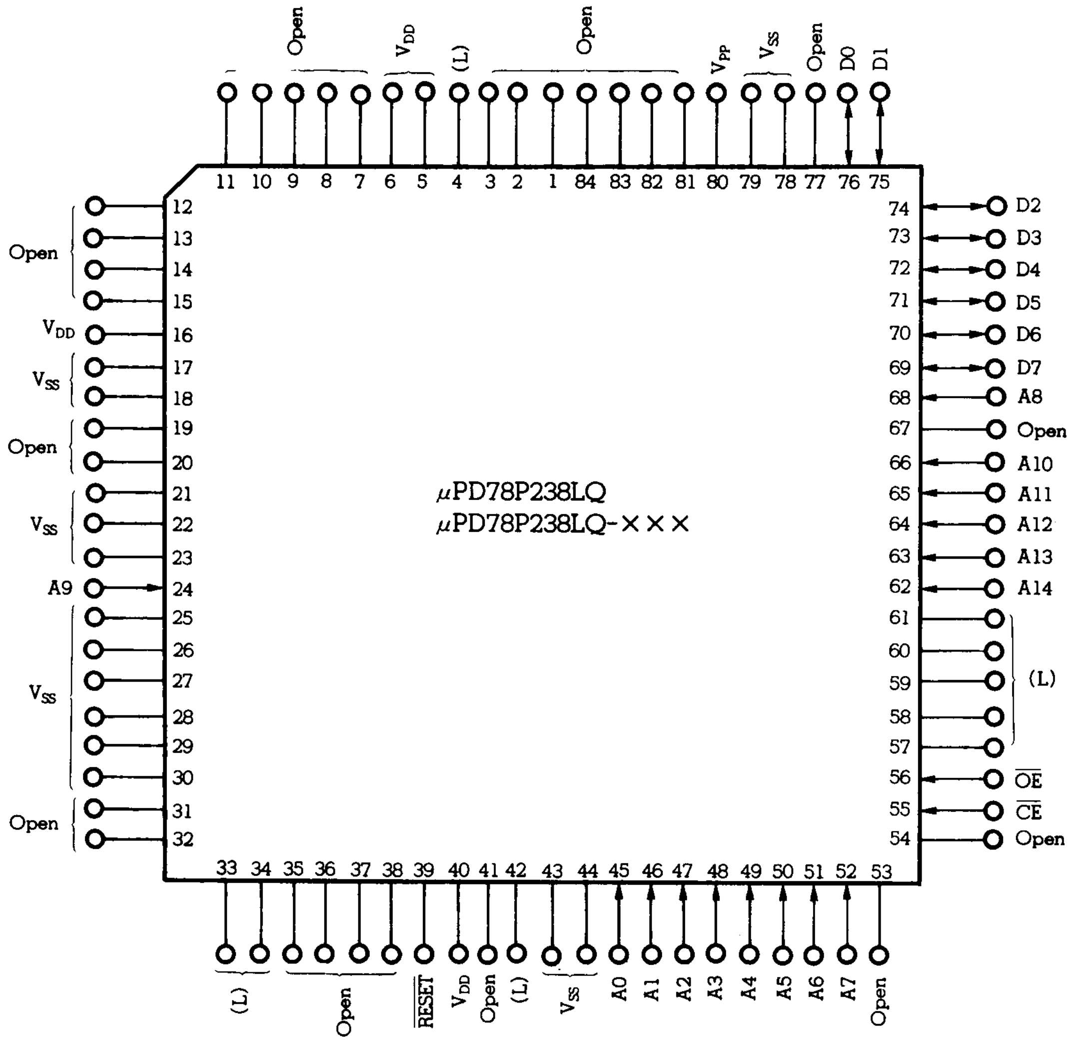
Caution: L : Connect each of these pins to V_{SS} via a 10kΩ pull-down resistor.

V_{SS} : Ground these pins.

Open : Do not connect these pins.

$\overline{\text{RESET}}$: Keep this pin at low level.

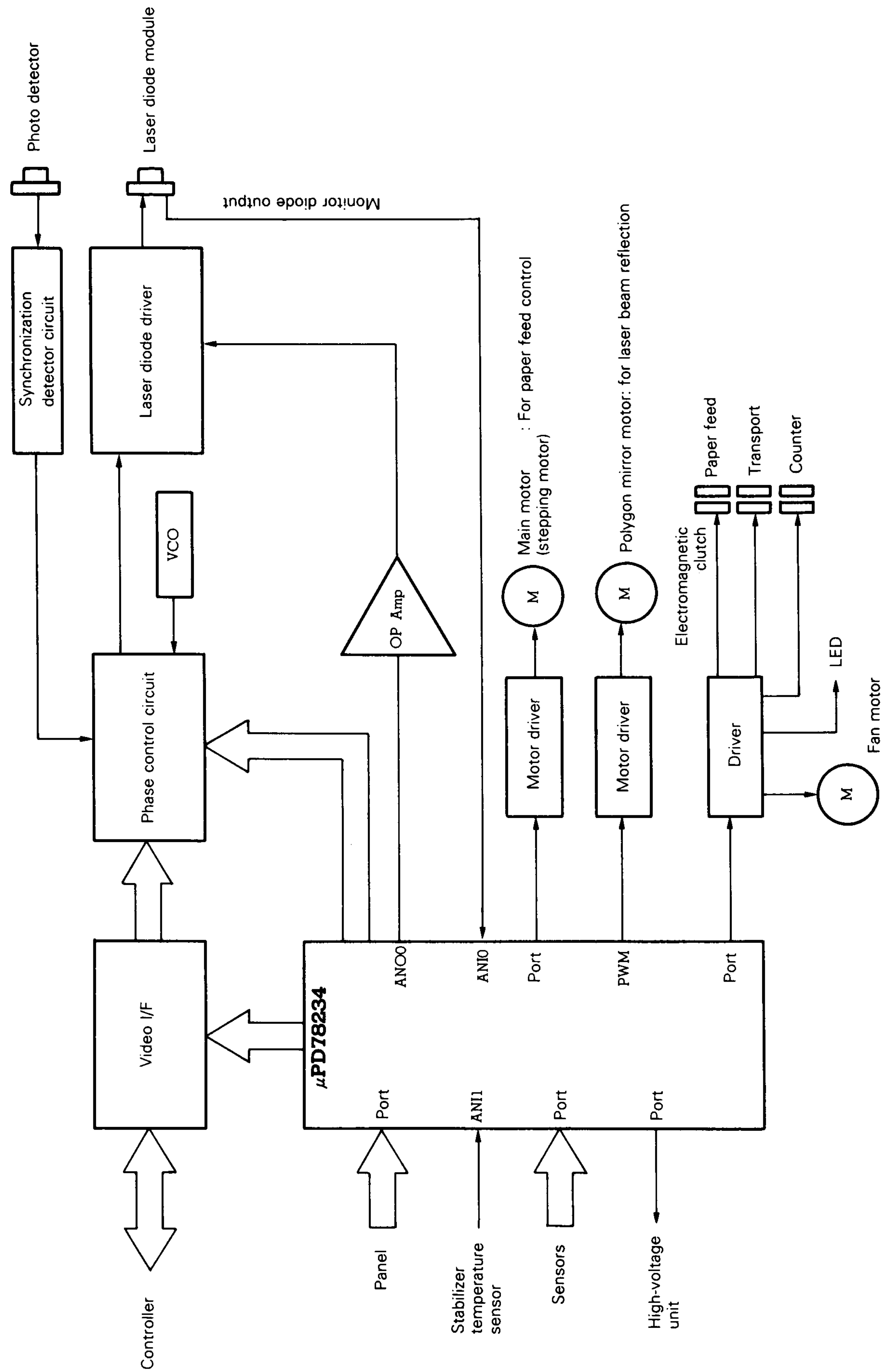
(3) 84-pin plastic QFJ (1150 mil)



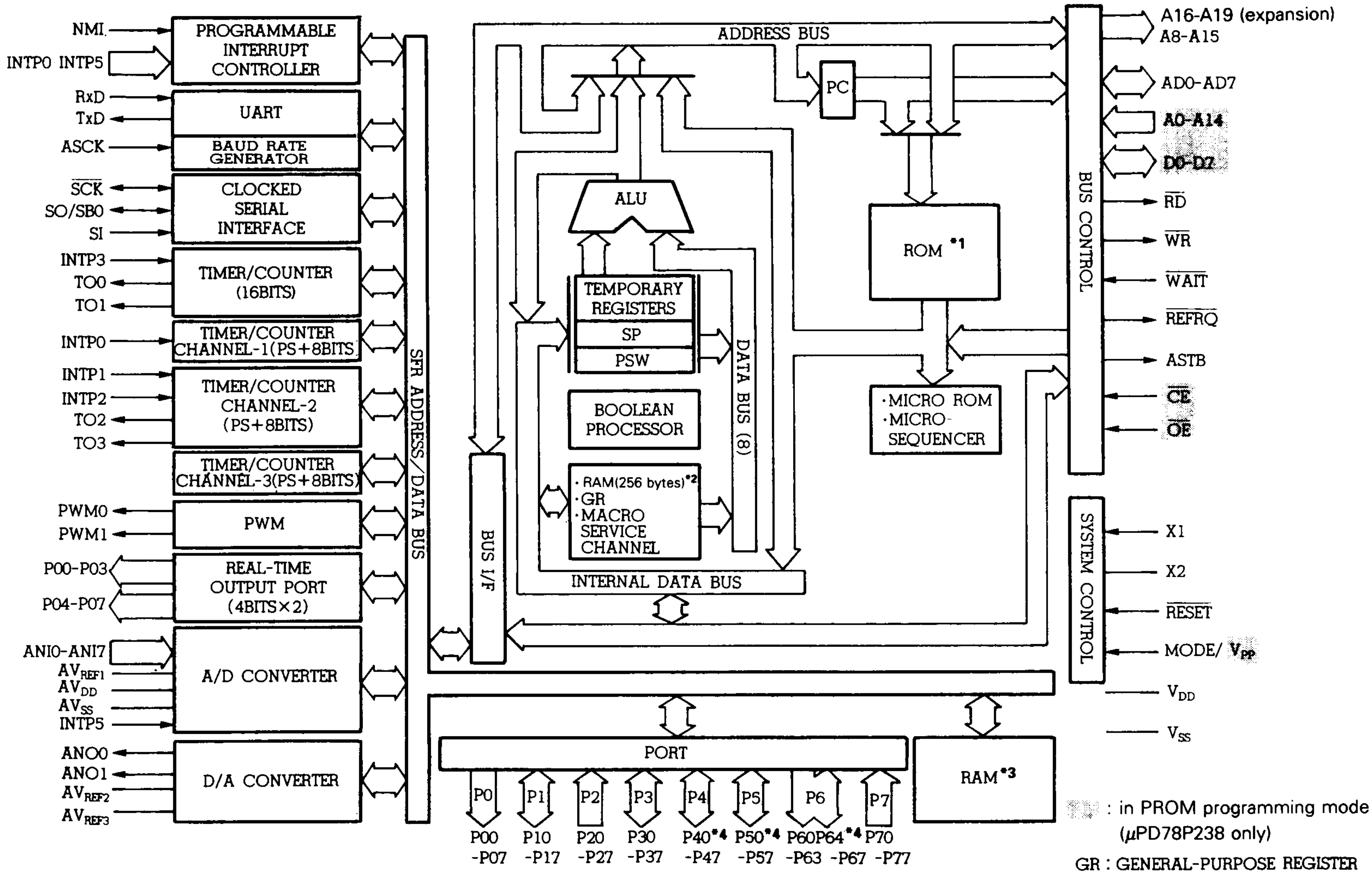
Caution: L : Connect each of these pins to V_{SS} via a 10kΩ pull-down resistor.
V_{SS} : Ground these pins.
Open : Do not connect these pins.
$\overline{\text{RESET}}$: Keep this pin at low level.

V_{PP}	: Programming Power Supply	$\overline{OE}$	: Output Enable
$\overline{RESET}$	: Reset	V_{DD}	: Power Supply
A0-A14	: Address Bus	V_{SS}	: Ground
D0-D7	: Data Bus		
$\overline{CE}$	: Chip Enable		

1.4 Application System Configuration Example (LBP engine)



1.5 Internal Block Diagram



***1:** μPD78233, 78237: Not provided, μPD78234: 16K bytes, μPD78238, 78P238: 32K bytes

***2:** Internal dual port RAM

***3:** Peripheral RAM (PRAM), μPD78233, 78234: 384 bytes, μPD78237, 78238, 78P238: 768 bytes

***4:** P40 to P47, P50 to P57, P64, and P65 of μPD78233 and 78237 cannot be used as port pins.

1.6 Specifications

Item			μPD78233	μPD78237	μPD78234	μPD78238	μPD78P238
Number of basic instructions (mnemonics)			65				
Minimum instruction execution time (at 12 MHz)			500ns		333ns		
Internal memory capacity	ROM		None		16K bytes	32K bytes	32K/16K bytes*1
	RAM		640 bytes	1024 bytes	640 bytes	1024 bytes	1024/640 bytes*1
Address area			Program memory area: 64K bytes, data memory area: 1M bytes				
I/O Pins		Input	16				
		Output	12				
		I/O	18		36		
		Total	46		64		
	Ancillary-function pins*2	w/pull-up register	24		42		
		LED direct drive output	8		24		
		Transistor direct drive output	8				
Real-time output port			4 bits × 2, or 8 bits × 1				
General-purpose register			8 bits × 8 × 4 banks (memory mapping)				
Timer/counter			16-bit timer/counter : timer register × 1 capture register × 1 compare register × 2 Pulse output possible (Toggle output PWM/PPG output One-shot pulse output)				
			8-bit timer/counter 1 : timer register × 1 capture/compare register × 1 compare register × 2 Pulse output possible real-time output : 4 bits × 2				
			8-bit timer/counter 2 : timer register × 1 capture register × 1 compare register × 2 Pulse output possible (Toggle output PWM/PPG output)				
			8-bit timer/counter 3 : timer register × 1 compare register × 1 –				
PWM output			12-bit resolution × 2 channels				
Serial interface			UART: 1 channel (with baud rate generator) Clock-synchronized serial I/O: 1 channel				

***1:** Set by software

***2:** The ancillary functions are included in the functions of the I/O pins.

Item	μPD78233	μPD78237	μPD78234	μPD78238	μPD78P238
A/D converter	8-bit resolution × 8 channels				
D/A converter	8-bit resolution × 2 channels				
Interrupts	19 sources (7 external and 12 internal) + BRK instruction Two priority levels (programmable) Two processing formats (vector interrupt, macro service)				
Instruction set	16-bit arithmetic operation instructions, multiplication/division instructions (8 bits × 8 bits,16 bits/8 bits), bit manipulation instructions, BCD adjustment, etc.				
Package	80-pin plastic QFP (14 × 14 mm, excluding pins) 94-pin plastic QFP (20 × 20 mm, excluding pins) 84-pin plastic QFJ (□1150 mil) 94-pin ceramic WQFN (μPD78P238 only)				

1.7 Differences among μPD78234 Sub-Series Products

1.7.1 Functional differences

Item			μPD78233	μPD78237	μPD78234	μPD78238	μPD78P238
Minimum instruction execution time (at 12 MHz)			500ns		333ns		
Internal memory capacity		ROM	None		16K bytes	32K bytes	32K/16K bytes*1
		RAM	640 bytes	1024 bytes	640 bytes	1024 bytes	1024/640 bytes*1
Internal memory size selection			Impossible				Possible
I/O Pins		Input	16				
		Output	12				
		I/O	18		36		
		Total	46		64		
	Ancillary-function pins*2	w/pull-up register	24		42		
		LED direct drive output	8		24		
		Transistor direct drive output	8				
ROM-less mode setting			ROM-less product		MODE pin = high level		Impossible

***1:** Set by software
***2:** The ancillary functions are included in the functions of the I/O pins.

1.7.2 Differences in package

Two types of QFPs are available: 94-pin and 80-pin packages. The differences in these packages are shown in the following table, depending on the development condition of the user system and mounting. Take this into consideration when selecting a package.

Item		94-pin QFP	80-pin QFP
Package size		20 × 20 mm	14 × 14 mm
PROM with window		μPD78P238KF (evaluation with PROM model is easy)	None
Mounting method	Moisture content control for infrared reflow and VPS	Refer to the data sheets for details.	

1.7.3 Differences between μPD78234/μPD78238 and μPD78234(A)/μPD78238(A)

Item \ Product	μPD78234, 78238	μPD78234(A), 78238(A)
Quality grade	Standard	Special
Package	• 80-pin plastic QFP • 94-pin plastic QFP • 84-pin plastic QFJ • 94-pin ceramic WQFN (μPD78P238 only)	• 80-pin plastic QFP • 94-pin plastic QFP*

***:** μPD78234(A) only

CHAPTER 2 PIN FUNCTIONS

2.1 Pin Function List

2.1.1 Ordinary operation mode

(1) Port

Pin name	I/O	Multiplexed pin	Function
P00-P07	Output	—	Port 0 (P0): This port can be used as a real-time output port (4-bit x 2) and can directly drive transistors.
P10	I/O	PWM0	Port 1 (P1): This port can be set in input or output mode in bit units. Pins in input mode can be specified collectively by software to be connected to the internal pull-up resistor. This port can directly drive LEDs.
P11		PWM1	
P12-P17		—	
P20	Input	NMI	Port 2 (P2): P20 cannot be used as a general-purpose port pin (because this pin inputs a nonmaskable interrupt signal). However, the input level for P20 can be checked by an interrupt routine. Pins P22 through P27 can be specified collectively by software to be connected to the internal pulled-up resistor in 6-bit units.
P21		INTP0	
P22		INTP1	
P23		INTP2/CI	
P24		INTP3	
P25		INTP4/ASCK	
P26		INTP5	
P27		SI	
P30	I/O	RxD	Port 3 (P3): This port can be set in input or output mode in bit units*. Pins in input mode can be specified collectively by software to be connected to the internal resistor.
P31		TxD	
P32		$\overline{SCK}$	
P33		SO/SB0	
P34-P37		TO0-TO3	
P40-P47*	I/O	AD0-AD7	Port 4 (P4): This port can be set in input or output mode in units of 8 bit. Pins in input mode can be specified collectively by software to be connected to the internal pull-up resistor in 8 bits unit. These ports can directly drive LEDs.
P50-P57*	I/O	A8-A15	Port 5 (P5): This port can be set in input or output mode in bit units. Pins in input mode can be specified collectively by software to be connected to the internal pull-up resistor.
P60-P63	Output	A16-A19	Port 6 (P6): Pins P64 through P67 can be specified in input or output mode in bit units. Pins P64 through P67 can be specified collectively by software to be connected to the internal pull-up resistor.
P64*	I/O	$\overline{RD}$	
P65*		$\overline{WR}$	
P66		$\overline{WAIT}$	
P67		$\overline{REFRQ}$	
P70-P77	Input	ANI0-ANI7	Port 7 (P7)

***:** These μPD78233 pins cannot be used as port pins.

(2) Pins other than for ports

Pin name	I/O	Function	Multiplexed pin
TO0-TO3	Output	Timer output	P34-P37
CI	Input	Count clock input to 8-bit timer/counter 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	I/O	Serial data I/O (SBI)	P33/SO
SI	Input	Serial data input (3-line serial I/O)	P27
SO	Output	Serial data output (3-line serial I/O)	P33/SB0
$\overline{SCK}$	I/O	Serial clock I/O (SBI, 3-line serial I/O)	P32
NMI	Input	External interrupt request	P20
INTP0			P21
INTP1			P22
INTP2			P23/CI
INTP3			P24
INTP4			P25/ASCK
INTP5			P26
AD0-AD7	I/O	Time-division address/data bus (for external memory connection)	P40-P47*
A8-A15	Output	Higher address bus (for external memory connection)	P50-P57*
A16-A19	Output	Higher address of expanded address (for external memory connection)	P60-P63
$\overline{RD}$	Output	Read strobe to external memory	P64*
$\overline{WR}$	Output	Write strobe to external memory	P65*
$\overline{WAIT}$	Input	Wait insertion	P66
ASTB	Output	Latch timing output (when external memory is accessed) for time-division address (A0-A7)	—
$\overline{REFRQ}$	Output	Refresh pulse output to external pseudo static memory	P67
$\overline{RESET}$	Input	Chip select	—
X1	Input	For connection of system clock oscillator crystal (clock can also be input to X1)	—
X2	—		
MODE	Input	For ROM-less mode setting (external access of same space as internal ROM). Keep this pin for μPD78233 at high level, and that for μPD78234 at low level.	—
ANI0-ANI7	Input	A/D converter analog voltage input	P70-P77
ANO0, ANO1	Output	D/A converter analog voltage output	—
AV_{REF1}	—	A/D converter reference voltage input	—
AV_{REF2}, AV_{REF3}		D/A converter reference voltage input	
AV_{DD}		A/D converter power source	
AV_{SS}		A/D converter GND	
V_{DD}		Power source	
V_{SS}		GND	
NC		No connection	

*: These μPD78233 pins cannot be used as port pins.

2.1.2 PROM programming mode (μPD78P238 only: $V_{PP} \geqq 5$ V, $\overline{\text{RESET}}$ = L)

Pin name	I/O	Function
V_{PP}	Input	Set PROM programming mode. Apply high voltage when program is written or verified.
$\overline{\text{RESET}}$		Set PROM programming mode
A0-A14		Address bus
D0-D7	I/O	Data bus
$\overline{\text{CE}}$	Input	PROM enable input/program pulse input
$\overline{\text{OE}}$		Read strobe input to PROM
V_{DD}	—	Power source
V_{SS}		GND

2.2 Pin Functions

2.2.1 Ordinary operation mode

(1) P00-P07 (Port 0) ... Tri-state output

Port 0 is an 8-bit output port with an output latch, which can directly drive transistors. This port can be set in the output mode or high-impedance mode by port 0 mode register (PM0).

Pins P00 through P03 and P04 through P07 can respectively constitute 4-bit real-time output ports or an 8-bit real-time port, through which the contents of buffer registers (P0L and P0H) can be output at specified time intervals. Whether these pins are used to constitute a general-purpose output port or a real-time output port is specified by real-time output port control register (RTPC).

When the $\overline{\text{RESET}}$ signal is input, all the pins enter the high-impedance status, and the output latch contents become undefined.

(2) P10-P17 (Port 1) ... Tri-state I/O

Port 1 is an 8-bit I/O port with output latch which can be specified in the input or output mode in bit units by port 1 mode register (PM1). Each pin is internally connected to a programmable pull-up resistor, and can directly drive an LED.

Pins P10 and P11 can also be used as PWM output pins when so specified by PWM control register (PWMC).

When the $\overline{\text{RESET}}$ signal is input, all the pins enter the output high-impedance state, being set in the input mode, and the output latch contents become undefined.

Table 2-1 Port 1 Operation Mode

Pin	Port mode	Control signal output mode	To use control signal pin function:
P10	I/O port	PWM0 output	Set EN0 bit for PWMC register to 1
P11		PWM1 output	Set EN1 bit for PWMC register to 1
P12-P17		—	—

(a) Port mode

Pins P10 and P11 are set in the port mode, when the EN0 and EN1 bits for the PWMC register are cleared to 0. Pins P12 through P17 are always in the port mode. Each bit for port 1 can be set in the input or output mode by port 1 mode register (PM1).

(b) Control signal output mode

Pins P10 and P11 respectively output PWM signals when the EN0 and EN1 bits for the PWMC register are set to 1.

(3) P20-P27 (Port 2) ... Input

Port 2 is an 8-bit input port. Pins P22 through P27 are internally connected to a software programmable pull-up resistor. This port functions not only as an input port, but also to input control signals such as external interrupt signals (see Table 2-2 below). All the eight pins for this port are provided with a Schmitt trigger circuit to prevent malfunctioning due to noise.

Table 2-2 Port 2 Operation Mode

Pin	Function
P20	Input port/NMI input*
P21	Input port/INTP0 input/CR11 capture trigger input/trigger signal of 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/A/D converter external trigger input
P27	Input port/SI input

*: NMI input can be accepted, regardless of whether interrupts are enabled or disabled.

(a) Function as port pins

The port 2 pin levels can always be read or tested, regardless of the multiplexed pins operation.

(b) Function as control signal input pins

(i) NMI (Non-maskable interrupt)

This pin inputs an external nonmaskable interrupt request signal. Whether the interrupt request signal is detected at the rising or falling edge can be specified by external interrupt mode register (INTM0).

(ii) INTP0 to INTP5 (Interrupt from peripherals)

These pins input external interrupt request signals. An interrupt occurs when a valid edge specified by external interrupt mode registers (INTM0 and INTM1), has been input to any of the INTP0 to INTP5 pins (refer to **Chapter 13 Edge Detection Function**).

Pins INTP0 through INTP3 and INTP5 can also be used to input external trigger signals for the following functions:

- INTP0 ... Capture trigger input pin for 8-bit timer/counter 1
  Trigger signal for real-time output port
- INTP1 ... Capture trigger input pin for 8-bit timer/counter 2
- INTP2 ... External count clock input pin for 8-bit timer/counter 2
- INTP3 ... Capture trigger input pin for 16-bit timer/counter
- INTP5 ... External capture trigger input pin for A/D converter

(iii) CI (Clock Input)

This pin inputs an external clock for 8-bit timer/counter 2.

(iv) ASCK (Asynchronous serial clock)

This pin inputs an external baud rate clock.

(v) SI (Serial input)

This pin inputs serial data (in 3-line serial I/O mode).

(4) P30-P37 (Port 3) ... Tri-state I/O

Port 3 is an 8-bit I/O port which can be specified in the input or output mode by port 3 mode register (PM3). Each of these pins is internally connected with a software-programmable pull-up resistor.

In addition to constituting a general-purpose I/O port, these pins can also input or output control signals. Whether these pins are set in the port mode or control signal mode can be specified by port 3 mode control register (PMC3) in bit units, as shown in Table 2-3. The level for each pin can always be read and tested, regardless of the multiplexed pin function.

When the $\overline{\text{RESET}}$ signal is input, these pins enter the output high-impedance state, being set in the input mode, and the output latch contents become undefined.

Table 2-3 Port 3 Operation Mode (n = 0–7)

<table>
<tr><td>Mode</td><td>Port mode</td><td>Control signal mode</td></tr>
<tr><td>Condition</td><td>PMC3n = 0</td><td>PMC3n = 1</td></tr>
<tr><td>P30</td><td rowspan="8">I/O port</td><td>RxD input</td></tr>
<tr><td>P31</td><td>TxD output</td></tr>
<tr><td>P32</td><td>$\overline{\text{SCK}}$ I/O</td></tr>
<tr><td>P33</td><td>SO output/SB0 I/O</td></tr>
<tr><td>P34</td><td>TO0 output</td></tr>
<tr><td>P35</td><td>TO1 output</td></tr>
<tr><td>P36</td><td>TO2 output</td></tr>
<tr><td>P37</td><td>TO3 output</td></tr>
</table>

(a) Port mode

Each port pin set in the port mode by the PMC3 register can be set in the input or output mode by port 3 mode register (PM3).

(b) Control mode

Each pin for port 3 can be set in the control signal mode by the PMC3 register, in which case the following control signal pin functions are available:

(i) RxD (Receive data)

This pin inputs serial data for the asynchronous serial interface.

(ii) TxD (Transmit data)

This pin outputs serial data for the asynchronous serial interface.

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

This pin inputs/outputs serial clock for the clock-synchronized serial interface.

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

The SO pins output serial data (in the 3-line serial I/O mode). The SBO pin is the I/O pin for the serial bus in the SBI mode.

(v) TO0-TO3 (Timer output)

These are timer output pins.

(5) P40-P47 (Port 4) ... Tri-state I/O

Port 4 is an 8-bit I/O port with an output latch which can be specified in the input or output mode in 8 bit units by memory expansion mode register (MM). Each of these pins is internally connected to a software programmable pull-up resistor and can directly drive an LED.

Port 4 also functions as a time-division address/data bus (AD0-AD7), when an external memory or I/O is connected to the microcomputer.

Note that port 4 for μPD78233 functions only as a time-division address/data bus (AD0-AD7).

When the $\overline{\text{RESET}}$ signal is input, port 4 is set in the input mode (output high-impedance state) and the output latch contents become undefined.

(6) P50-P57 (Port 5) ... Tri-state I/O

Port 5 is an 8-bit I/O port with an output latch which can be specified in the input or output mode in bit units by port 5 mode register (PM5). Each of these pins is internally connected to a software programmable pull-up resistor and can directly drive an LED.

Port 5 also functions as an address bus (A8-A15), when an external memory or I/O is connected to the microcomputer.

Note that port 5 for μPD78233 functions only as an address bus (A8-A15).

When the $\overline{\text{RESET}}$ signal is input, port 5 is set in the input mode (output high-impedance state) and the output latch contents become undefined.

(7) P60-P67 (Port 6) ... P60-P63: output, P64-P67: tri-state I/O

Port 6 is an 8-bit I/O port with an output latch. Pins P64 through P67 are internally connected to a software programmable pull-up resistor.

In addition to port pin functions, these pins also have control signal I/O pin functions, as shown in Table 2-4. When an operation shown in this table is performed, each of these pins functions as a control signal pin. Pins P64 and P65 for μPD78233 for function only as the $\overline{\text{RD}}$ output and $\overline{\text{WR}}$ output pins.

When the $\overline{\text{RESET}}$ signal is input, pins P60 through P63 become output high-impedance state, and then enter into low level after the $\overline{\text{RESET}}$ signal is released. Additionally, when the $\overline{\text{RESET}}$ signal is input, pins P64 through P67 are set in the input port mode (output high-impedance state). The output latch contents for the higher 4 bits become undefined, and those for the lower 4 bits become 0H.

Table 2-4 Port 6 Operation Mode

Pin	Port mode	Control signal I/O mode	To use control signal pin function:
P60-P63	Output port	A16-A19 output	Set MM6 bit of MM register to 1
P64	I/O port	$\overline{\text{RD}}$ output	Set external memory expansion mode by MM2-0 bits for MM register. For μPD78233, only these control signal pin functions are available from these pins.
P65		$\overline{\text{WR}}$ output	
P66		$\overline{\text{WAIT}}$ input	Specified by PWn1 and PWn0 bits (n=2, 3) for PW or MM register, or by setting P66 to input mode
P67		$\overline{\text{REFRQ}}$ output	Set RFEN bit for RFM register to 1

Caution: P60-P63 go into a high-impedance state during $\overline{\text{RESET}}$ input, but go low after $\overline{\text{RESET}}$ signal is removed. Therefore, design the external circuit so that these pins may go low under initial status.

Remarks: For details, refer to **Chapter 15 Local Bus Interface**.

(a) Port mode

Pins P60 through P63 are output pins, while P64 through P67 can be set in the input or output mode in bit units by port 6 mode register (PM6).

(b) Control signal I/O mode

(i) A16-A19 (Address Bus)

These pins constitute the higher 4 bits for the address bus, when an external memory space is expanded (10000H-FFFFFH) and can be manipulated by memory expansion mode register (MM).

(ii) $\overline{\text{RD}}$ (Read strobe)

This pin outputs a read strobe signal, which is used to read data from an external memory, when the MODE pin is 1 or an external memory is expanded.

(iii) $\overline{\text{WR}}$ (Write strobe)

This pin outputs a write strobe signal, which is used to write data to an external memory, when the MODE pin is 1 or when so specified by the MM register.

(iv) $\overline{\text{WAIT}}$ (Wait)

This pin inputs a wait signal and is manipulated by programmable wait control (PW) register or MM register.

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

This pin outputs a refresh pulse to an external pseudo static memory and is manipulated by refresh mode register (RFM).

(8) P70-P77 (Port 7) ... Input

Port 7 is an 8-bit input port, which also inputs analog voltages (ANI0-ANI7) to the internal A/D converter. The levels for these pins can always be read or tested, regardless of the multiplexed pins operation.

(9) ASTB (Address strobe) ... Output

This pin outputs a timing signal that allows external latches address information to access an external memory.

(10) MODE (Mode) ... Input

This pin inputs a control signal that accesses an external program memory, instead of the internal ROM. When a low-level signal is input to this pin, the internal ROM is accessed. When a high-level signal is input, the microcomputer is set in the ROM-less mode, in which the external memory is accessed. This pin for μPD78233 is fixed to the high level. With μPD78234, this pin is fixed to the low level.

Caution: Be sure to keep the MODE pin for μPD78P238 at the low level. μPD78P238 cannot be set in the ROM-less mode, even when the MODE pin is made high.

(11) X1 and X2 (Crystal)

A crystal oscillator that oscillates the system clocks is connected across these pins. When an external clock is to be supplied, input the clock signal to the X1 pin and a signal 180° out of phase with the clock signal to the X2 pin.

(12) $\overline{\text{RESET}}$ (Reset) ... Input

This pin inputs an active-low system reset signal.

(13) ANO0, ANO1 ... Output

These pins output the analog voltage for the D/A converter.

(14) AV_{REF1}

This pin inputs a reference voltage to the A/D converter.

(15) AV_{REF2}

This pin inputs a reference voltage (positive) to the D/A converter.

(16) AV_{REF3}

This pin inputs a reference voltage (negative) to the D/A converter.

(17) AV_{DD}

This pin supplies power to the A/D converter. Make sure that the potential placed on this pin is the same as that on the V_{DD} pin.

(18) AV_{SS}

This is the ground pin for the A/D converter. Make sure that the potential placed on this pin is the same as that on the V_{SS} pin.

(19) V_{DD}

Connect this pin to the positive polarity of a power source.

(20) V_{SS}

Connect this pin to the ground potential.

(21) NC (Non-connection)

This pin is not internally connected.

2.2.2 PROM programming mode (μPD78P238)

(1) V_{PP} (Programming power supply) ... Input

This pin sets μPD78P238 in the PROM programming mode. When 5V or higher is applied to this pin and the $\overline{\text{RESET}}$ pin goes low, μPD78P238 is set in the PROM programming mode.

When the $\overline{\text{OE}}$ pin is made high and the $\overline{\text{CE}}$ pin is made low with the V_{PP} pin at 12.5 V, the program data on pins D0 through D7 can be written to an internal PROM cell, selected by pins A0 through A14.

(2) $\overline{\text{RESET}}$ ... Input

This input pin sets μPD78P238 in the PROM programming mode. When this pin is at low level, and the voltage placed on the V_{PP} pin is 5 V or more, μPD78P238 is set in the PROM programming mode.

(3) A0-A14 (Address bus) ... Input

These pins constitute an address bus, which selects an address for the internal PROM (0000H-7FFFH).

(4) D0-D7 (Data bus) ... I/O

These pins constitute a data bus, through which the program is written to or read from the internal PROM.

(5) $\overline{\text{CE}}$ (Chip enable) ... Input

This pin inputs an enable signal for the internal PROM. When this signal is active, the program can be read from or written to the internal PROM.

(6) $\overline{\text{OE}}$ (Output enable) ... Input

This pin inputs a read strobe signal to the internal PROM. If this signal is made active, when the $\overline{\text{CE}}$ pin is low, the program in the internal PROM cell selected by the A0 through A14 pins can be read out to pins D0 through D07.

(7) V_{DD}

Connect this pin to the positive polarity for a power source.

(8) V_{SS}

Connect this pin to the ground potential.

2.3 I/O Circuits and Processing Unused Pins

Table 2-5 identifies the input/output circuit category internally connected to each pin. Aschematic view of each input/output circuit is shown in Fig. 2-1.

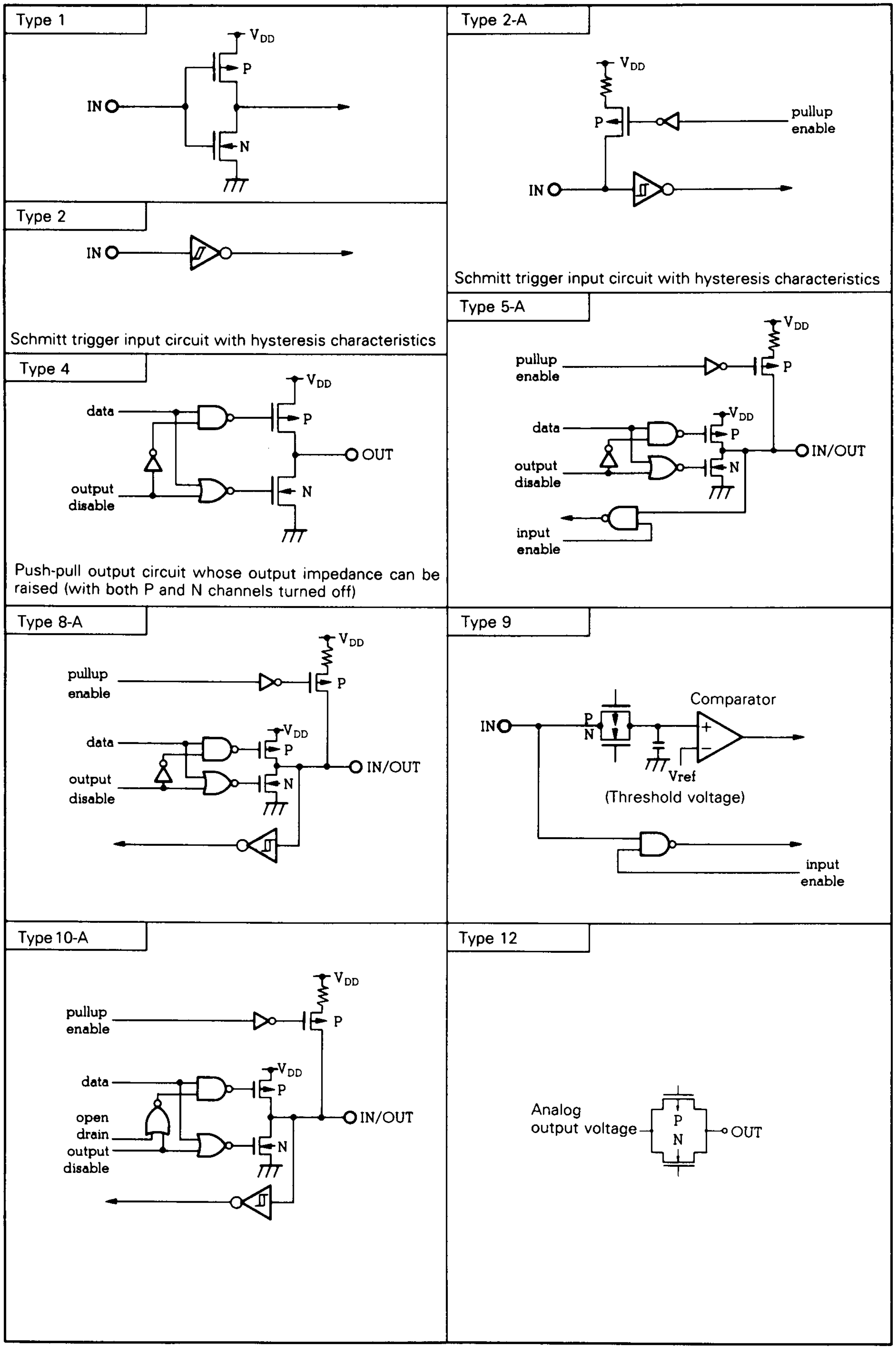
Table 2-5 I/O Circuit Category and Processing Unused Pins

Pin	I/O circuit type	I/O	Recommended connections for unused pins
P00-P07	4	Output	Open
P10-P17	5-A	I/O	Input : Connect to V_{DD} Output : Open
P20/NMI	2	Input	Connect to V_{DD} or V_{SS}
P21/INTP0			
P22/INTP1	2-A		Connect to V_{DD}
P23/INTP2/CI			
P24/INTP3			
P25/INTP4/ASCK			
P26/INTP5			
P27/SI			
P30/RxD	5-A	I/O	Input : Connect to V_{DD} Output : Open
P31/TxD			
P32/$\overline{SCK}$	8-A		
P33/SB0/SO	10-A		
P34/TO0-P37/TO3	5-A		
P40/AD0-P47/AD7			
P50/A8-P57/A15			
P60/A16-P63/A19	4	Output	Open
P64/$\overline{RD}$	5-A	I/O	Input : Connect to V_{DD} Output : Open
P65/$\overline{WR}$			
P66/$\overline{WAIT}$			
P67/$\overline{REFRQ}$			
P70/ANI0-P77/ANI7	9	Input	Connect to V_{SS}
ANO0, ANO1	12	Output	Open
ASTB	4		
$\overline{RESET}$	2	Input	—
MODE	1		
AV_{REF1}-AV_{REF3}	—		Connect to V_{SS}
AV_{SS}			
V_{DD}			Connect to V_{DD}

Remarks: The same circuit category numbers are used among the products in the 78K series and the circuit category numbers used in this product are not necessarily serial numbers (because some circuits are not provided to this product).

Caution: Connect I/O pins to V_{DD} via a resistor in the 10 to 99 KΩ range, when I/O mode is undefined. (This is especially important when the low-level input voltage for the RESET input pin exceeds the rated voltage on power application, or when the input or output function for the pins is selected by software).

Fig. 2-1 I/O Circuits List



2.4 Notes

(1) When processing the unused pins, connect I/O pins to V_{DD} through a resistor in the 10 to 99 kΩ range when I/O mode is undefined (especially when the voltage on the RESET input pin exceeds the low-level input voltage on power application, or when I/O is selected by software).

(2) Be sure to make the MODE pin for μPD78P238 low. The ROM-less mode is not set, even when the MODE pin is made high.

(3) P60-P63 go into a high-impedance state during $\overline{\text{RESET}}$ input, but go low after $\overline{\text{RESET}}$ signal is removed. Therefore, design the external circuit so that these pins may go low under initial status.

CHAPTER 3 CPU FUNCTION

3.1 Memory Space

μPD78234 can access 1M byte of memory space address up to 64K bytes of memory. The program memory is mapped differently, depending on the MODE pin state. μPD78233 is used with MODE = H.

When using μPD78P238, mapping the memory for μPD78234 can be exchanged to that for μPD78238, depending on the specification of the memory size select register. The μPD78P238 can not be used with MODE = H.

(1) μPD78233 (MODE = H)

The program memory is mapped on an external memory (64640 bytes: 00000H-0FC7FH). This area can also be used as a data memory.

The data memory is mapped in the internal RAM (640 bytes: 0FC80H-0FEFFH). In the 1M-byte expansion mode, an expansion data memory is mapped on an external memory (960K bytes: 10000H-FFFFFH).

(2) μPD78234 (MODE = L)

The program memory is mapped on the internal ROM (16K bytes: 00000H-03FFFH) and external memory (48256 bytes: 04000H-0FC7FH). The external memory is accessed in the external memory expansion mode. The area to be mapped on the external memory can also be shared with the data memory.

The data memory is mapped on the internal RAM (640 bytes: 0FC80H-0FEFFH). In the 1M-byte expansion mode, the external memory (960 bytes: 10000H-FFFFFH) is mapped as an expansion data memory.

(3) μPD78237 (MODE = H)

The μPD78237 program memory is mapped on an external memory (64256 bytes: 00000H through 0FAFFH). This area can also be shared with the data memory.

The data memory is mapped on the internal RAM (1024 bytes: 0FB00H through 0FEFFH). In the 1M-byte expansion mode, the external memory (960K bytes: 10000H through FFFFFH) is mapped as an expansion data memory.

(4) μPD78238 (MODE = L)

The program memory is mapped on the internal ROM (32K bytes: 00000H through 07FFFH) and external memory (31488 bytes: 08000H through 0FAFFH). The external memory is accessed in the external memory expansion mode. The area to be mapped on the external memory can also be shared with the data memory.

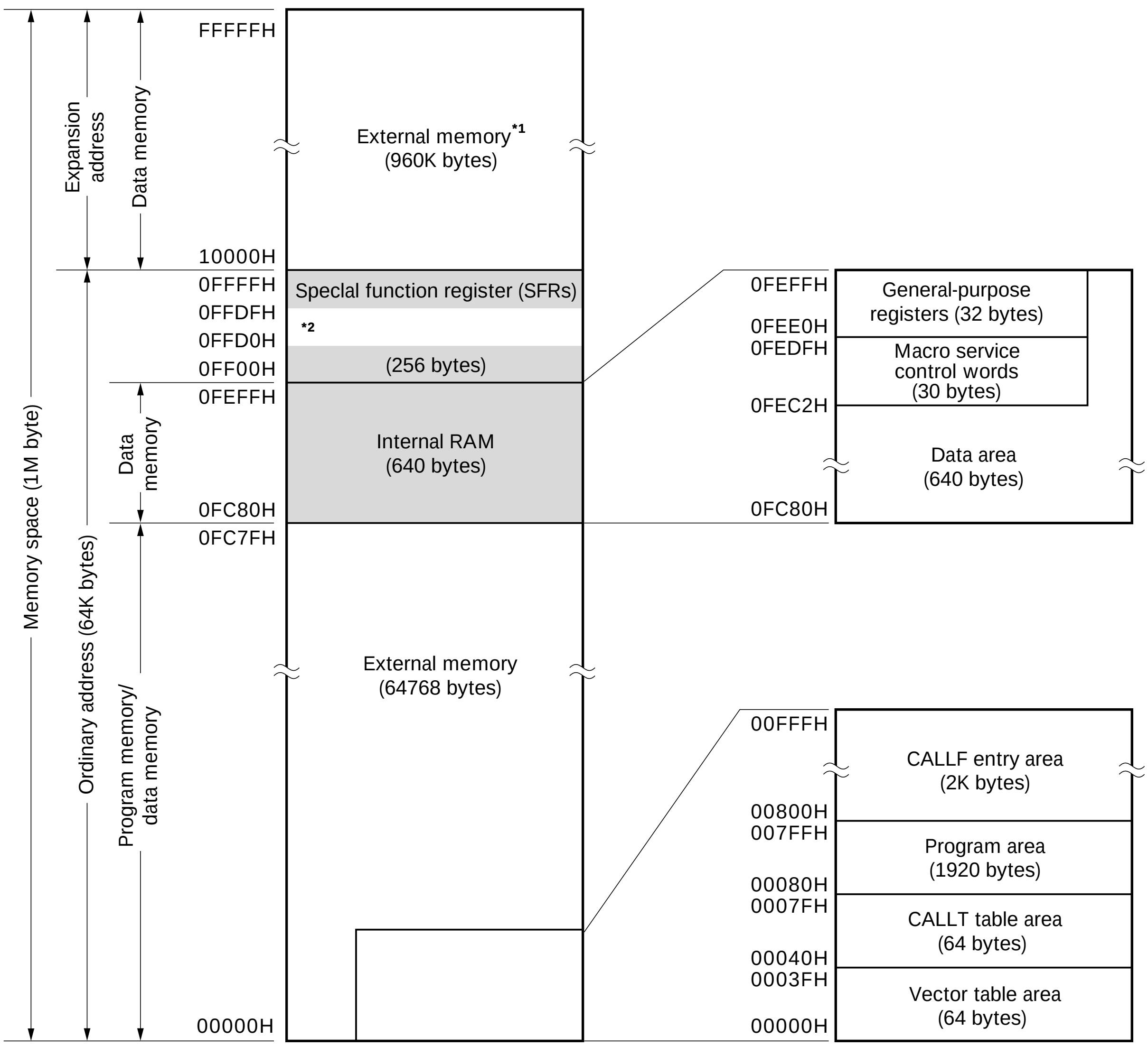
The data memory is mapped on the internal RAM (1024 bytes: 0FB00H through 0FEFFH). In the 1M-byte expansion mode, the external memory (960K bytes: 10000H through FFFFFH) is mapped as an expansion data memory.

(5) μPD78P238 (MODE = L)

Whether the μPD78P238 memory is mapped in the same manner as the μPD78234 memory or whether the μPD78238 memory, is specified by the memory size select register (IMS). When the $\overline{\text{RESET}}$ signal has been input, the memory mapping is the same as that for μPD78238.

Caution: μPD78P238 cannot be used with its MODE pin made high (ROM-less operation cannot be specified).

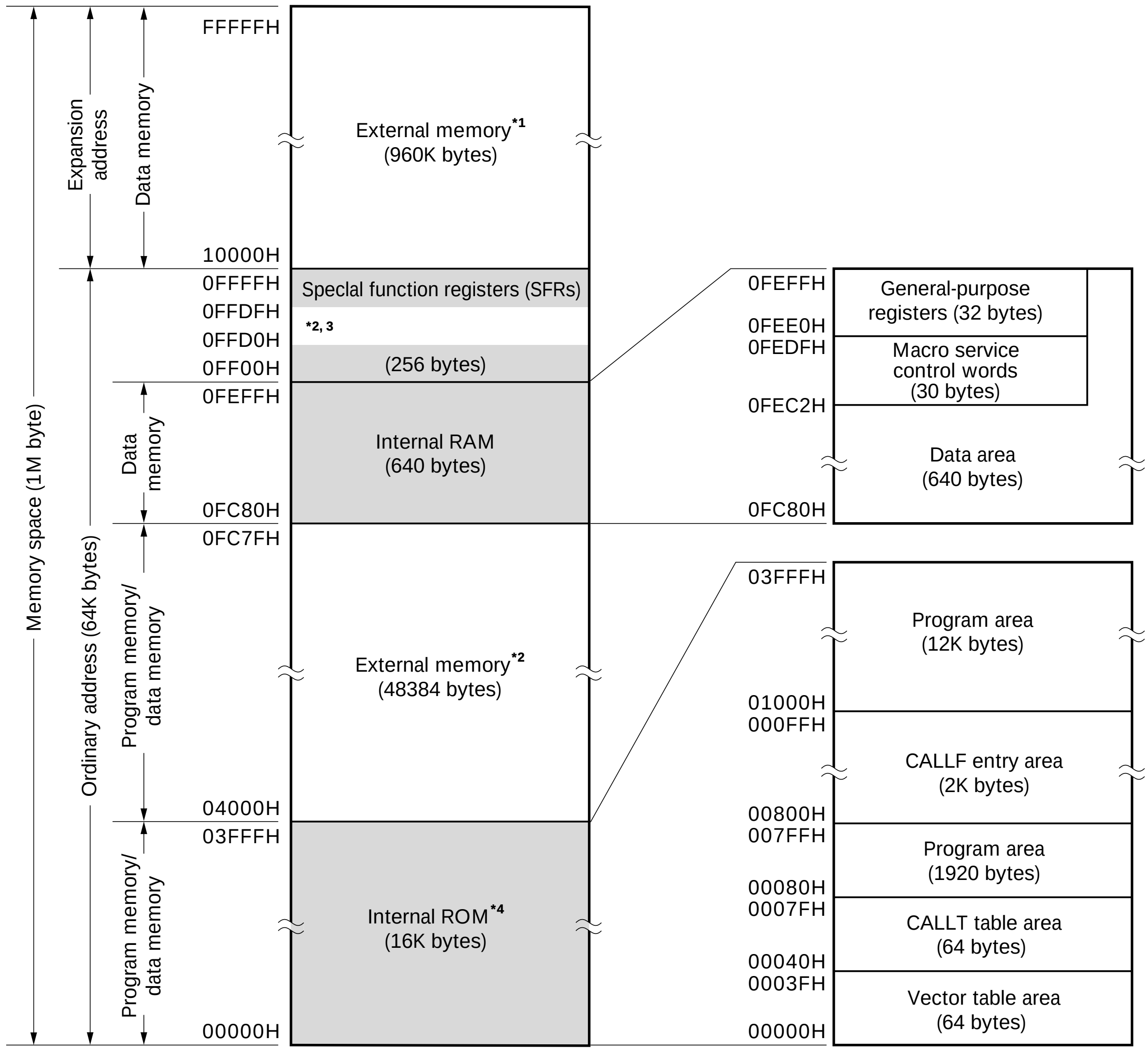
Fig. 3-1 μPD78233 Memory Map



***1:** Accessed in 1M-byte expansion mode. The shaded area in the figure is in the internal memory.

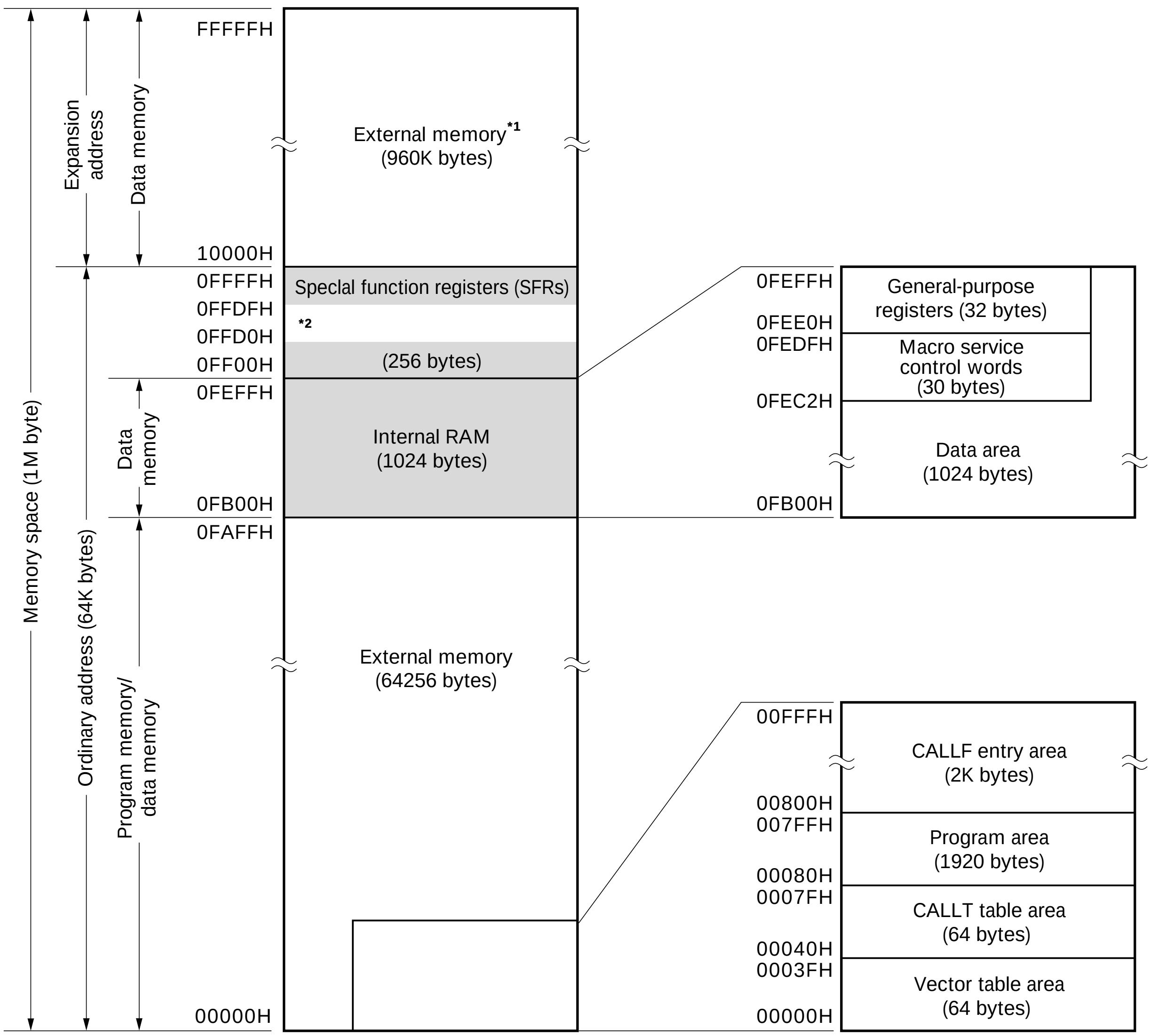
***2:** External SFR area

Fig. 3-2 μPD78234 Memory Map



***1:** Accessed in 1M-byte expansion mode. The shaded area in the figure is in the internal memory.
***2:** Accessed in external memory expansion mode
***3:** External SFR area
***4:** For μPD78P238, internal PROM

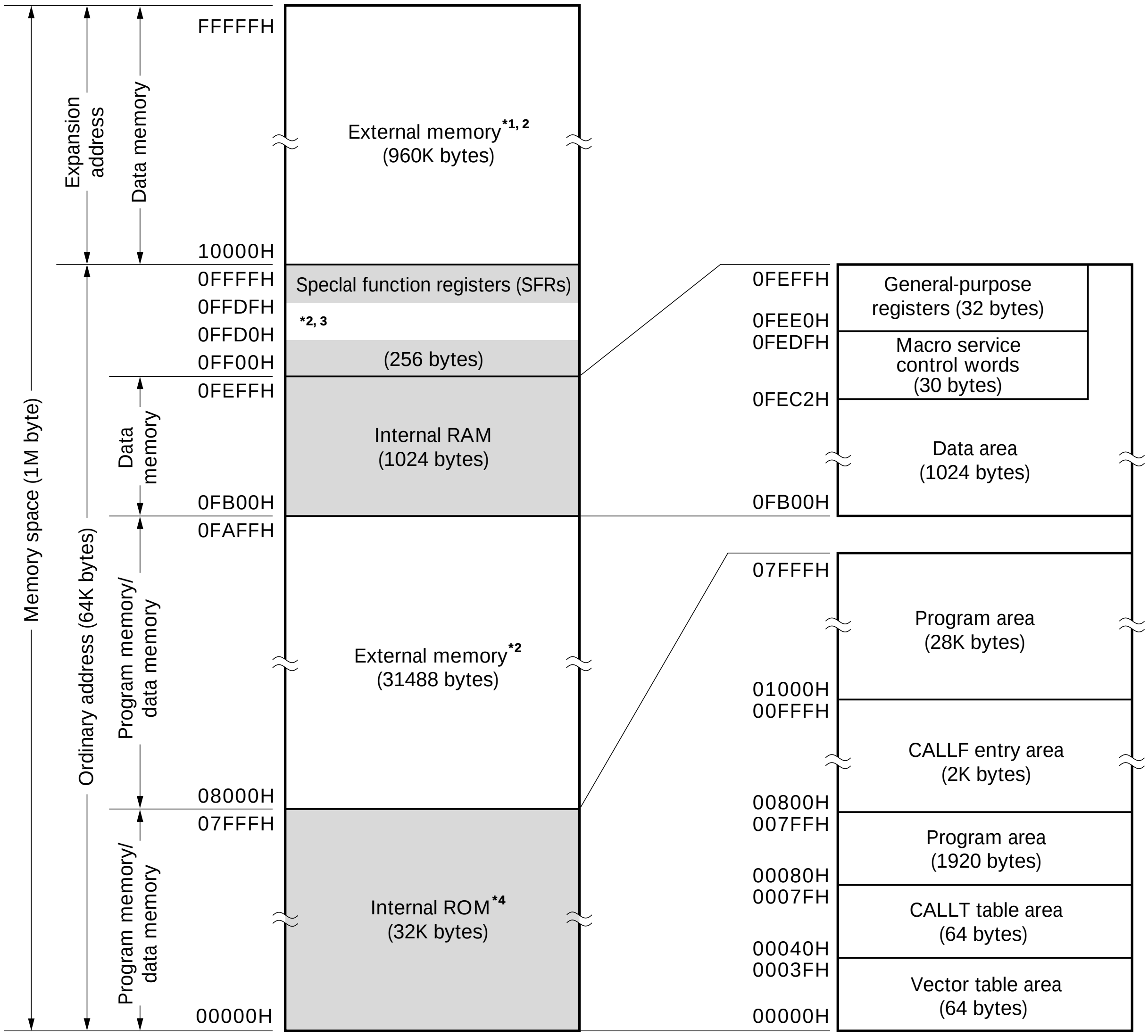
Fig. 3-3 μPD78237 Memory Map



***1:** Accessed in 1M-byte expansion mode. The shaded area indicates the internal memory.

***2:** External SFR area

Fig. 3-4 μPD78238 Memory Map



***1:** Accessed in 1M-byte expansion mode. The shaded area in the figure is in the internal memory.

***2:** Accessed in external memory expansion mode

***3:** External SFR area

***4:** For μPD78P238, internal PROM

3.1.1 Internal program memory area

The 16K × 8 bits ROM (32K × 8 bits in the case of μPD78238) is mapped on an 00000H through 03FFFH address space (00000H through 07FFFH in the case of μPD78238) in which programs and table data are stored. This ROM is usually addressed by program counter (PC).

This address area for μPD78233 is in an external memory (ROM-less operation).

(1) Vector table area

The 64-byte area, consisting of 00000H through 0003FH, is reserved as a vector table area, in which a program starts addresses to which program execution is to branch, when the $\overline{\text{RESET}}$ signal is input or when an interrupt occurs. Of a 16-bit address, the lower 8 bits are stored in an even address, while the higher 8 bits are stored in an odd address.

Table 3-1 Vector Table

Vector Table Address	Interrupt Request
00000H	Reset ($\overline{\text{RESET}}$ 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	NTCSI
0003EH	BRK

(2) CALLT instruction table area

In a 64-byte area, consisting of 00040H through 0007FH, the subroutine entry addresses for a 1-byte call instruction (CALLT) can be stored.

(3) CALLF instruction entry area

From addresses 00800H to 00FFFH, a subroutine can be directly called by a 2-byte call (CALLF) instruction.

3.1.2 Internal RAM area

A 640-byte general-purpose static RAM (1024-byte for μPD78238) is mapped on addresses 0FC80H through 0FEFFH (0FB00H through 0FEFFH in the case of μPD78238).

This area consists of the following two RAMs:

- Peripheral RAM (PRAM) : 0FC80H-0FDFFH (0FB00H-0FDFFH in the case of μPD78238)
- Internal dual-port RAM (IRAM) : 0FE00H-0FEFFH

The internal dual-port RAM (IRAM) can be accessed at high speeds. Especially, the area consisting of 0FE20H through 0FEFFH can be accessed in short direct addressing mode (refer to **78K/II Series User's Manual-Instruction part CHAPTER 6 ADDRESSING**).

In a 32-byte area consisting of 0FEE0H through 0FEFFH, four banks of general-purpose registers are mapped, while macro service control words are mapped on a 30-byte area of 0FEC2H through 0FEDFH.

Caution: Programs cannot be fetched from the internal RAM area.

Remarks: Store data, work areas, and status flags which are frequently used in the 0FE20H through 0FEC1H area.

When the area, consisting of addresses 0FE00H through 0FE1FH, is used as a stack area or data transfer area for macro service channel and macro service, it can be accessed at high speeds, so that the system throughput is effectively improved. (This area cannot be accessed in the short-direct addressing mode, but can be manipulated in the same manner as the other memory space. However, since this area can be accessed more quickly than the other memory area, it is considered to be more advantageous if the area is used as a stack area or a transfer area for macro service channel and macro service data, rather than when it is used in any other addressing modes.)

3.1.3 Special function register (SFR) area

The memory areas for addresses 0FF00H to 0FFFFH are mapped on-chip hardware peripherals, special function registers (SFRs). Refer to 3.2.5 Special function registers.

The 0FFD0H through 0FFDFH area is mapped as an external SFR area, so that external peripheral I/Os can be accessed in the ROM-less mode for μPD78233 or in the external memory expansion mode for μPD78234 (which is set by memory expansion mode register (MM)).

Caution: Do not access addresses to which special function registers are not mapped; otherwise, μPD78234 may be dead-locked. The deadlock can be released only by the RESET signal.

3.1.4 External SFR area

The 16-byte area, consisting of addresses 0FFD0H through 0FFDFH in the SFR area, is mapped as an external SFR area. When μPD78233 (ROM-less) is used, or when μPD78234 is used in the external memory expansion mode (set by the memory expansion mode register (MM)), the external peripheral I/Os can be accessed through the address bus and address/data bus.

The external SFR area can be accessed in the SFR addressing mode. Therefore, its feature is that the peripheral I/Os can be easily manipulated and that the object size can be compressed. In addition, the external SFR area can be specified as macro service type B SFRs.

The bus operation, when the external SFR area is accessed, is the same as that when the ordinary memory is accessed (see **CHAPTER 15 LOCAL BUS INTERFACE FUNCTIONS**).

3.1.5 External memory area

The area consisting of addresses 04000H through 0FC7FH (08000H through 0FAFFH in the case of μPD78P238) is an external memory space which can be accessed when so specified by the memory expansion mode register (MM). In this area, programs and table data can be stored and peripheral I/O devices can be mapped.

An 00000H through 0FC7FH area for μPD78233 can always be accessed (00000H through 0FAFFH in the case of μPD78237).

3.1.6 External expansion data memory space

An area consisting of addresses 10000H through FFFFFH can be accessed when the 1M-byte expansion mode is specified by the memory expansion mode register (MM), in which case the P60 through P63 pins for port 6 function as a 4-bit expansion address bus (A16 through A19). The data memory space is treated as 16 banks, each consisting of 64K bytes. The lower 4 bits of the P6 and PM6 registers function as bank registers, that select a bank. This data memory space is useful for storing high-capacity data, such as for a character generator.

This space can be accessed by executing an instruction capable of expansion addressing with a bank to be used (the higher 4 bits (A16 through A19) for the address information) set by a bank register (P60 through P63 for the P6 register or PM60 through PM63 for the PM6 register). The higher 4-bit address output from the P60 through P63 pins is valid, only when an instruction capable of expansion addressing is executed.

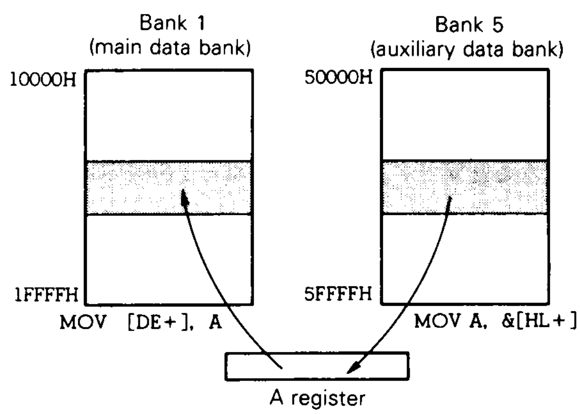
Since two bank registers are available, two data banks can be usually be manipulated. The two bank registers are selected, depending on whether "&" is appended to the operand of an instruction. When "&" is appended, the P6 register serves as a bank register. When "&" is not appended, the PM6 register serves as a bank register.

Therefore, for example, if one of the two banks is used as a main data bank to specify an RAM area, while the other bank is used as an auxiliary data bank, to specify a data ROM area, the data read out from the data ROM (e.g., character data for a printer) can be expanded or reduced and then stored in the RAM.

Example: To transfer data from bank 5 to bank 1, where bank 1 is the main bank and bank 5 is the auxiliary bank

```
        MOV MM,#47H   ; Sets memory expansion mode
        MOV PM6,#1H   ; Sets main bank register (PM6)
        MOV P6,#5H    ; Sets auxiliary bank register (P6)
        MOV B,#0FFH   ; Sets loop counter
          :
LOOP:     :
          :
        MOV A,&[HL+]  ; Reads data from bank 5 (P6 register contents are appended as highest address
                        information)
        MOV [DE+],A   ; Stores data in bank 1 (PM6 register contents are appended as highest address
                        information)
          :
          :
        DBNZ B,$LOOP ; Repetitive processing
```

Fig. 3-5 Data Transfer between Banks



Remarks 1: Both the MOV [DE+],A and MOV A,&[HL+] instructions are stored in bank 0.

2: The OP code and execution time for an instruction, that uses the PM6 register as a bank register, are shorter than those for an instruction that uses the P6 register. An instruction that uses the P6 register is shorter than an instruction that uses the PM6 register, and the execution time for the former instruction is accordingly shorter than that for the latter.
Therefore, use the PM6 register as a main bank register that specifies a bank which is frequently accessed, and the P6 register as an auxiliary bank register, because the bank specified by the auxiliary bank register often changes.

3.1.7 Memory mapping of μPD78P238

μPD78P238 is provided with a 32K-byte internal ROM and 1024-byte internal RAM. Therefore, its memory mapping is slightly different from that for μPD78234. μPD78P238 is provided with a function (memory size selection function) so as not to use a part of the internal memory to make up for the difference.

To select memory size, the memory size select register (IMS) is used. To map the μPD78P238 memory in the same manner as mapping the μPD78234 memory, be sure to write this register immediately after reset.

The IMS register can be written by an 8-bit manipulation instruction. The register format is shown in Fig. 3-6.

When the $\overline{\text{RESET}}$ signal has been input, the IMS register contents are set to FFH.

Fig. 3-6 Memory Size Select Register

	7	6	5	4	3	2	1	0
IMS	IMS7	IMS6	IMS5	IMS4	IMS3	IMS2	IMS1	IMS0

IMS0-7	Memory size
FFH	Same as μPD78238
DDH	Same as μPD78234

This register is not provided to μPD78234 and 78238. However, even when an instruction to write this register is executed with μPD78234 or 78238, the operations are not affected.

3.2 Registers

3.2.1 Program counter (PC)

This 16-bit binary counter holds the address for an instruction in the program to be executed next (see **Fig. 3-7**).

Usually, the contents of this counter are automatically incremented, according to the number of bytes in the instruction to be fetched. When an instruction that causes program execution to branch has been executed, the immediate data for that instruction or the contents of a register are set in PC.

When the $\overline{RESET}$ signal has been input, the internal ROM (external memory contents in the case of μPD78233) at address 00000H are stored in the lower 8 bits for the PC, while the contents of address 00001H are stored in the higher 8 bits.

Fig. 3-7 Program Counter Configuration

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC	PC15	PC14	PC13	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0

3.2.2 Program status word (PSW)

This 8-bit register is a collection of flags that are set or reset as a result of an instruction execution (see **Fig. 3-8**).

Data can be read from or written to the PSW in 8 bit units. In addition, each flag for the PSW can be manipulated by a bit manipulation instruction. When an vector interrupt is accepted, or when the BRK or PUSH PSW instruction is executed, the PSW contents are saved to the stack. To restore the PSW contents from the stack, execute the RETI, RETB, or POP PSW instruction.

When the $\overline{RESET}$ signal is input, the PSW contents are initialized to 02H, disabling interrupts.

Fig. 3-8 Program Status Word Configuration

	7	6	5	4	3	2	1	0
PSW	IE	Z	RBS1	AC	RBS0	0	ISP	CY

(1) Carry flag (CY)

This flag indicates whether an overflow or underflow has occurred as a result of executing an addition or subtraction instruction.

This flag also retains the value shifted out as a result of executing a shift or rotate instruction, and functions as a bit accumulator when a bit arithmetic operation instruction is executed.

(2) Interrupt priority status flag (ISP)

This flag controls the priority for the maskable vector interrupt that can be currently accepted.

When a maskable interrupt has been accepted, the content of the interrupt priority specification flag, corresponding to the accepted interrupt, is transferred to ISP. When the non-maskable interrupt (NMI) has been accepted, the ISP content is cleared to "0".

When this flag is 0, a vector interrupt having a low priority specified by the priority flag (PR0), is disabled. When this flag is 1, the interrupt is enabled, regardless of the priority of the interrupt. Accepting an interrupt is actually controlled according to the IE flag status.

The content of this flag is updated each time a maskable interrupt has been accepted.

For details, refer to **CHAPTER 14 INTERRUPT FUNCTION**.

(3) Register bank selector flags (RBS0 and RBS1)

These two flags select one of the four register banks (see Table 3-2), and store 2-bit information that indicates a register bank selected as a result of executing the SEL RBn instruction.

Table 3-2 Selecting 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)

This flag is set to 1, when a carry from or borrow to data bit 3 has occurred as a result of executing an arithmetic operation; otherwise, the flag remains 0. This flag is used when a BCD adjustment instruction is executed.

(5) Zero flag (Z)

This flag is set to 1, when the result of an arithmetic operation executed is 0; otherwise, it remains 0.

(6) Interrupt request enable flag (IE)

This flag controls acceptance of an interrupt request by the CPU.

When this flag is cleared to 0, the interrupt is disabled, only the nonmaskable interrupt and unmasked macro service can be accepted, and any other interrupts are disabled.

When this flag is set to 1, the interrupt is enabled, and accepting an interrupt request is controlled by the ISP flag, an interrupt mask flag corresponding to the interrupt request generated, and interrupt priority flag.

When the EI instruction is executed, this flag is set to 1; when the DI instruction is executed or when an interrupt is accepted, it is cleared to 0.

3.2.3 Stack pointer (SP)

This is a 16-bit register that retains the first address for the stack area (LIFO format: 00000H through 0FFFFH) (see Fig. 3-9) and is used to address the stack area when a subroutine or interrupt is processed.

The stack pointer contents are decremented, when data is written (saved) to stack area and incremented when the data is read (restored) from stack area (see Figs. 3-10 and 3-11).

When the $\overline{\text{RESET}}$ signal has been input, the stack pointer contents become undefined; so, be sure to initialize stack pointer immediately after reset has been released and before calling a subroutine or accepting an interrupt, by executing an initialization program.

Example: To initialize SP
MOVW SP,#0FEE0H ; SP ← 0FEE0H (when used from FEDFH)

Fig. 3-9 Stack Pointer Configuration

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SP	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0

Fig. 3-10 Data Saved to Stack Area

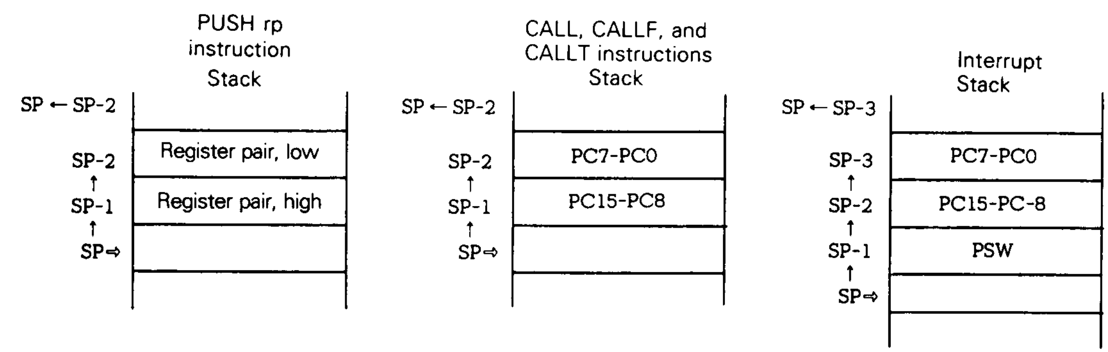
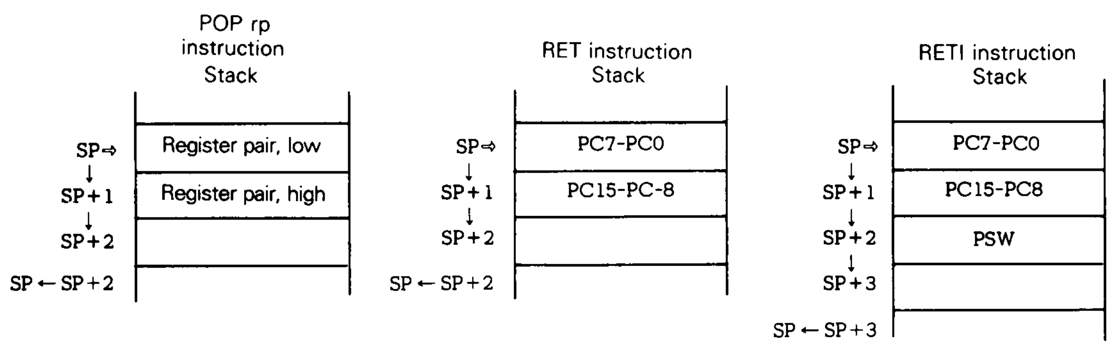


Fig. 3-11 Data Restored from Stack Area



Caution 1: The whole 64K-byte area can be accessed in the stack addressing mode, but a stack area connot be reserved on the SFR area and internal ROM area.

2: SP becomes undefined when the $\overline{\text{RESET}}$ signal has been input. The nonmaskable interrupt can be accepted immediately after reset has been released. Therefore, if the nonmaskable interrupt occurs while SP is undefined immediately after reset has been released, an unexpected operation may be performed. To prevent this, set the initial value of SP immediately after reset is released. For details, refer to 14.3.2 Accepting nonmaskable interrupt.

3.2.4 General-purpose registers

(1) Configuration

The general-purpose registers are mapped on a specific area (0FEE0H through 0FEFFH) in the data memory. On this area, four banks of registers are mapped, with one bank consisting of eight 8-bit registers: X, A, C, B, E, D, L, and H (see **Fig. 3-12**).

Fig. 3-12 General-Purpose Registers Configuration

	(8-bit processing)			(16-bit processing)
0FEE0H	A E1H	X E0H	↑	AX E0H
	B E3H	C E2H	Register bank 3	BC E2H
	D E5H	E E4H	(RBS1, 0 = 11)	DE E4H
	H E7H	L E6H	↓	HL E6H
	A E9H	X E8H	↑	AX E8H
	B EBH	C EAH	Register bank 2	BC EAH
	D EDH	E ECH	(RBS1, 0 = 10)	DE ECH
	H EFH	L EEH	↓	HL EEH
	A F1H	X F0H	↑	AX F0H
	B F3H	C F2H	Register bank 1	BC F2H
	D F5H	E F4H	(RBS1, 0 = 01)	DE F4H
	H F7H	L F6H	↓	HL F6H
	A F9H	X F8H	↑	AX F8H
	B FBH	C FAH	Register bank 0	BC FAH
	D FDH	E FCH	(RBS1, 0 = 00)	DE FCH
0FEFFH	H FFH	L FEH	↓	HL FEH

A register bank to be used when an instruction is executed is specified by a CPU control instruction (SEL RBn). Register bank 0 is selected on $\overline{\text{RESET}}$.

The register bank under execution can be identified by reading the register bank selector flags (RBS0, RBS1) in the PSW.

The 0FEE0H-0FEFFH area can be addressed and accessed as the ordinary data memory, regardless of whether this area is used as a general-purpose register area or not.

Remarks: If the original register bank needs to be specified again when the register bank is changed, save the PSW contents to the stack by executing the PUSH PSW instruction and then execute the SEL RBn instruction. The original register bank can be specified again by the POP PSW instruction, unless the stack position has been changed.
To change the register bank by an interrupt routine, the PSW contents are automatically saved to the stack, when the interrupt is accepted, and restored by the RETI or RETB instruction. Therefore, if only one register bank is used by the interrupt routine, only the SEL RBn instruction has to be executed and the PUSH PSW instruction does not have to be executed.

Example 1: To change the register bank with ordinary program:
To specify register bank 2:

```
    :
PUSH PSW
SEL   RB2  ⎫
    :      ⎬ Operates with register bank 2
POP  PSW   ⎭
    :        Operates with original register bank
```

2: To change the register bank with interrupt routine:
To select register bank 1:

```
SEL   RB1  ⎫
    :      ⎬ Operates with register bank 1.
RETI       ⎭ Original register bank is automatically
             restored when execution returns from interrupt routine
```

(2) Function

Two 8-bit registers can form a 16-bit register pair. Therefore, four 16-bit register pairs, AX, BC, DE, and HL, are available.

Each register can also be used to temporarily store the result of an arithmetic operation and as the operand of an instruction that performs an arithmetic operation between registers.

Since the general-purpose register is provided with four register banks, efficient programs can be developed by using one register bank for interrupt processing and the others for ordinary processing.

Each register has its own functions, as follows:

A (R1):

This register plays the central role in 8-bit data transfer and arithmetic operations. It can also be used to store bit data.

This register can also be used to store an offset value in the indexed addressing mode.

AX (RP0):

This register plays the central role in 16-bit data transfer and arithmetic operations.

X (R0):

This register can store bit data.

B (R3):

These registers have loop counter functions and can be used with the DBNZ instruction.

The B register can also be used to store offset values in the indexed addressing mode.

C (R2):

This register fucntions as a loop counter and can be used by the DBNZ instruction.

DE (RP2), HL (RP3):

These registers function as pointers and specify a base address in the register indirect addressing and base addressing modes. In the indexed addressing mode, these registers store offset values.

Each register can be described in a function name that indicates the function of the register (such as X, A, C, B, E, D, L, H, AX, BC, DE, or HL) or in an absolute name (R0-R7, RP0-RP3). For the correspondence between the function names and absolute names, refer to Table 3-3.

Table 3-3 Correspondence between Function Names and Absolute Names

Function name	Absolute name
X	R0
A	R1
C	R2
B	R3
E	R4
D	R5
L	R6
H	R7

Function name	Absolute name
AX	RP0
BC	RP1
DE	RP2
HL	RP3

3.2.5 Special function registers (SFRs)

These registers have special functions, such as mode register and control register functions. They are mapped on a 256-byte memory area, consisting of addresses 0FF00H to 0FFFFH.

Caution: Do not access an address in this area to which no SFR is assigned; otherwise, μPD78234 may be deadlocked. The deadlock is released only by the RESET input.

Table 3-4 lists the SFRs. The meanings of the symbols in this table are as follow:

- Symbol ..... a symbol indicating the SFR address. This is a reserved word for NEC's assembler (RA78K/II) and can be used as sfr variable by #pragma sfr for C compiler (CC78K/II).
- R/W .... indicates whether the SFR is read-only, write-only, or read-write.
  - R/W : read-write
  - R : read-only
  - W : write only
- Bit unit ... indicates bit units in which the SFR can be manipulated. SFRs that can be manipulated in 16 bit units can be described as the operand sfrp for an instruction. If the SFR is to be specified by an address, describe the even address.
  SFRs that can be manipulated bitwise can be described as the operand for a bit manipulation instruction.
- At reset ... indicates the SFR status when the $\overline{\text{RESET}}$ signal has been input.

Table 3-4 List of Special Function Registers (SFRs) (1/3)

Address	Special function register (SFR) name	Symbol	R/W	Bit unit			At reset
				1 bit	8 bits	16 bits	
0FF00H	Port 0	P0	R/W	○	○	—	Undefined
0FF01H	Port 1	P1		○	○	—	
0FF02H	Port 2	P2	R	○	○	—	
0FF03H	Port 3	P3	R/W	○	○	—	
0FF04H	Port 4	P4		○	○	—	
0FF05H	Port 5	P5		○	○	—	
0FF06H	Port 6	P6		○	○	—	xxxx0000B
0FF07H	Port 7	P7	R	○	○	—	Undefined
0FF0AH	Port 0 buffer register	P0L	R/W	○	○	—	
0FF0BH	Port 0 buffer register	P0H		○	○	—	
0FF0CH	Real-time output port control register	RTPC		○	○	—	00H
0FF10H	16-bit compare register (16-bit timer/counter)	CR00		— —	— —	○	Undefined
0FF12H	16-bit compare register (16-bit timer/counter)	CR01		— —	— —	○	
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	— —	— —	○	
0FF1AH	8-bit capture register (8-bit timer/counter 2)	CR22		—	○	—	
0FF1CH	8-bit capture/compare register (8-bit timer/counter 1)	CR11	R/W	—	○	—	
0FF20H	Port 0 mode register	PM0	W	—	○	—	FFH
0FF21H	Port 1 mode register	PM1		—	○	—	
0FF23H	Port 3 mode register	PM3		—	○	—	
0FF25H	Port 5 mode register	PM5		—	○	—	
0FF26H	Port 6 mode register	PM6	R/W	○	○	—	FxH
0FF30H	Capture/compare control register 0	CRC0	W	—	○	—	10H
0FF31H	Timer output control register	TOC		—	○	—	00H
0FF32H	Capture/compare control register 1	CRC1		—	○	—	
0FF34H	Capture/compare control register 2	CRC2		—	○	—	
0FF40H	Pull-up resistor option register	PUO	R/W	○	○	—	
0FF43H	Port 3 mode control register	PMC3		○	○	—	

Table 3-4 List of Special Function Registers (SFRs) (2/3)

Address	Special function register (SFR) name	Symbol	R/W	Bit unit: 1 bit	Bit unit: 8 bits	Bit unit: 16 bits	At reset
0FF50H	16-bit timer register 0	TM0	R	—	—	○	0000H
				—	—		
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	—	○	—	00H
0FF5FH	Timer control register 1	TMC1	R/W	—	○	—	
0FF60H	D/A conversion setting register 0	DACS0	R/W	—	○	—	00H
0FF61H	D/A conversion setting register 1	DACS1	R/W	—	○	—	
0FF68H	A/D converter mode register	ADM	R/W	○	○	—	00H
0FF6AH	A/D conversion result register	ADCR	R	—	○	—	Undefined
0HH70H	PWM control register	PWMC	R/W	—	○	—	05H
0FF72H 0FF73H	PWM modulo register 0	PWM0	W	—	—	○	Undefined
0FF74H 0FF75H	PWM modulo register 1	PWM1	W	—	—	○	
0FF7DH	One-shot pulse output control register	OSPC	R/W	○	○	—	00H
0FF80H	Clock-synchronized serial interface mode register	CSIM	R/W	○	○	—	00H
0FF82H	Serial bus interface control register	SBIC		○	○	—	
0FF86H	Serial shift register	SIO		—	○	—	Undefined
0FF88H	Asynchronous serial interface mode register	ASIM	R/W	○	○	—	80H
0FF8AH	Asynchronous serial interface status register	ASIS	R	○	○	—	00H
0FF8CH	Serial receive buffer : UART	RXB		—	○	—	Undefined
0FF8EH	Serial transmit shift register: UART	TXS	W	—	○	—	
0FF90H	Baud rate generator control register	BRGC		—	○	—	00H
0FFC0H	Standby control register	STBC	R/W	—	○	—	0000x000B
0FFC4H	Memory expansion mode register	MM		○	○	—	20H
0FFC5H	Programmable wait control register	PW		○	○	—	80H
0FFC6H	Refresh mode register	RFM		○	○	—	00H
0FFCFH	Memory size select register*	IMS	W	—	○	—	FFH
0FFD0H | 0FFDFH	(External SFR area)	—	R/W	○	○	—	Undefined
0FFE0H	Interrupt request flag register L	IF0L (IF0)	R/W	○	○	○	0000H
0FFE1H	Interrupt request flag register H	IF0H		○	○		

***:** μPD78P238 only

Table 3-4 List of Special Function Registers (SFRs) (3/3)

Address	Special function register (SFR) name	Symbol		R/W	Bit unit			At reset
					1 bit	8 bits	16 bits	
0FFE4H	Interrupt mask flag register L	MK0L	MK0	R/W	○	○	○	FFFFH
0FFE5H	Interrupt mask flag register H	MK0H			○	○		
0FFE8H	Priority flag register L	PR0L	PR0	R/W	○	○	○	FFFFH
0FFE9H	Priority flag register H	PR0H			○	○		
0FFECH	Interrupt processing format flag register L	ISM0L	PR0		○	○	○	0000H
0FFEDH	Interrupt processing format flag register H	ISM0H			○	○		
0FFF4H	External interrupt mode register 0	INTM0			○	○	—	00H
0FFF5H	External interrupt mode register 1	INTM1			○	○	—	
0FFF8H	Interrupt status register 0	IST			○	○	—	00H

3.3 Notes

(1) μPD78P238 cannot be used with its MODE pin made high (ROM-less operation specified).

(2) The program cannot be fetched from the internal RAM area.

(3) Special function registers (SFRs)
Do not access an address in the 0FF00H through 0FFFFH area, to which an SFR is not assigned. If such an address is accessed by mistake, μPD78234 may be deadlocked. The deadlock is released only by the RESET input.

(4) Stack pointer operations
The entire 64K-byte space can be accessed in the stack addressing mode, but a stack area cannot be secured on the SFR area and internal ROM area.

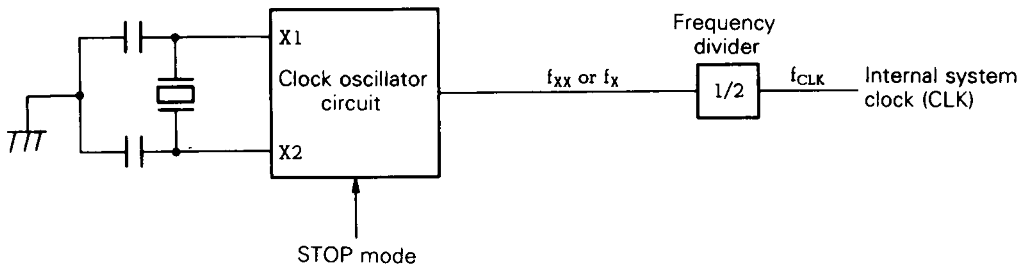
(5) Stack pointer initialization
SP becomes undefined when the $\overline{\text{RESET}}$ signal has been input. The nonmaskable interrupt can be accepted immediately after reset has been released. Therefore, if the nonmaskable interrupt occurs while SP is undefined immediately after reset has been released, an unexpected operation may be performed. To prevent this, initialize SP immediately after reset is released. For details, refer to **14.3.2 Accepting nonmaskable interrupt**.

CHAPTER 4 CLOCK GENERATOR CIRCUIT

4.1 Configuration and Function

The clock generator circuit generates the internal system clock (CLK) that is supplied to the CPU and the internal hardware. It is configured as shown in Fig. 4-1.

Fig. 4-1 Clock Generator Circuit Configuration



Remarks: f_{XX} : crystal/ceramic oscillation frequency
f_X : external clock frequency
f_{CLK} : internal system clock frequency
(= $1/2f_{XX}$ or $1/2f_X$)

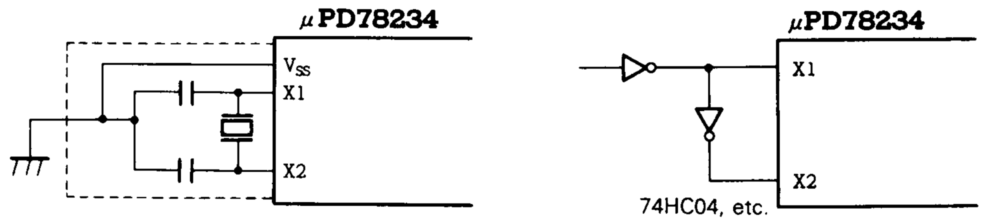
The clock oscillator circuit is oscillated by a crystal or ceramic oscillator connected across the X1 and X2 pins. This circuit stops oscillation, when a standby mode (STOP) is set (refer to **CHAPTER 16 STANDBY FUNCTION**).

An external clock can also be input. In this case, input the clock signal to the X1 pin and input a signal, with a phase reverse to that for the signal input to the X1 pin, to the X2 pin. Note, however, that the STOP mode cannot be used when the external clock is used.

The frequency divider divides the output from the clock oscillator circuit (f_{XX} when a crystal/ceramic oscillator is used and f_X when an external clock is used) by two, to generate the internal system clock (CLK).

Fig. 4-2 External Circuit for Clock Oscillator Circuit

(a) Crystal/ceramic oscillation **(b) External clock**



Caution: When using the oscillation circuit of the clock, wire the portion enclosed in dotted line in Fig. 4-2 as follows to avoid adverse influences on the wiring capacity:

- **Keep the wiring as short as possible.**
- **Do not cross the oscillator circuit signal lines with the signal lines for other circuits.**
- **Do not place the oscillator circuit signal lines in the vicinity of signal lines through which a high current flows.**
- **Grounding the oscillator circuit capacitor must be at the same potential as the V_{SS} pin. Do not ground the capacitor to a ground pattern to which a high current flows.**
- **Do not extract the signal from the oscillator circuit.**

Remarks: In general, the oscillation frequency for the crystal oscillator is extremely stable. Therefore, the crystal oscillator is ideal for time management applications with high accuracy (e.g., watch, frequency measurement, etc.)

The ceramic oscillator frequency stability is slightly inferior to that for the crystal oscillator. However, the ceramic oscillator has three desirable features: a short oscillation start time, compact size, and inexpensive price. Therefore, the ceramic oscillator is considered suitable for ordinary applications, where time management with high accuracy is not required.

In addition, some oscillators contain capacitors and thus can contribute to a reduction in the number of external components and to saving on the mounting area.

4.2 Note

Note the following points, when using the clock generator circuit:

4.2.1 When external clock is input

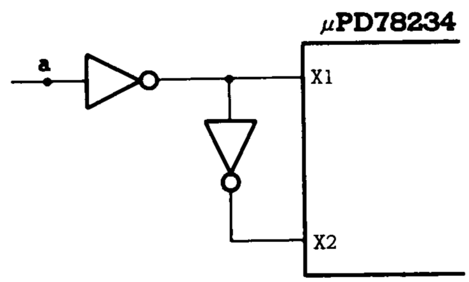
(1) Do not use the STOP mode, when an external clock is input. Otherwise, the microcomputer may be damaged. Even if the microcomputer is not damaged, its reliability is adversely affected.

(2) When inputting an external clock, input the signal in reverse phase to that for the signal input to the X1 pin to the X2 pin; otherwise, the microcomputer may malfunction due to noise.

(3) When inputting an external clock, use an HCMOS or a device having the driving capability equivalent to that for an HCMOS.

(4) Do not extract signals from the X1 and X2 pins. If necessary, extract the signal from point “a” in Fig. 4-3.

Fig. 4-3 Extracting Signal When External Clock Is Input



(5) Keep the wiring between the X1 and X2 pins and inverters as short as possible.

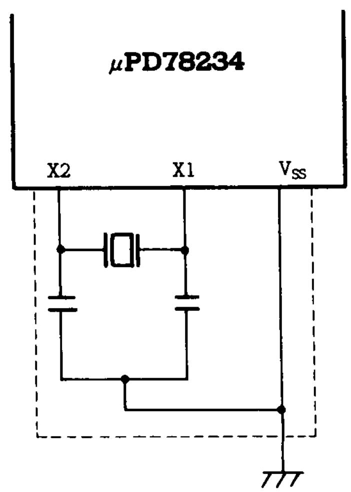
4.2.2 Crystal/ceramic oscillation

(1) The oscillator circuit is a high-frequency analog circuit. Thus, it is necessary to pay attention to the following points:

- Keep the wiring as short as possible.
- Do not cross the oscillator circuit signal lines with the signal lines for other circuits.
- Do not place the oscillator circuit signal lines in the vicinity of signal lines through which a high current flows.
- Grounding the oscillator circuit capacitor must be at the same potential as the V_{SS} pin. Do not ground the capacitor to a ground pattern to which a high current flows.
- Do not extract the signal from the oscillator circuit.

Unless the oscillator circuit operates normally, the microcomputer cannot operate stably. If an oscillation frequency with high accuracy is necessary, consult with the oscillator manufacturer.

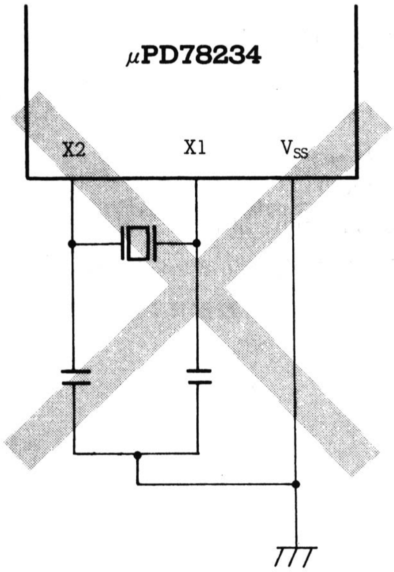
Fig. 4-4 Notes on Oscillator Connections



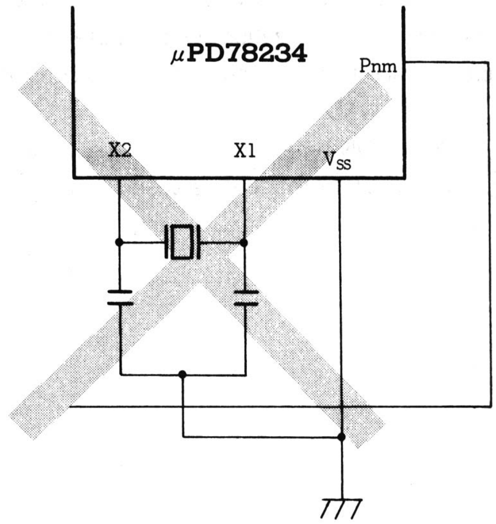
Caution: 1. Connect the oscillator circuit as closely as possible to the X1 and X2 pins.
2. Do not route any other signal lines in the area enclosed by the dotted line in the figure.

Fig. 4-5 Incorrect Oscillator Connection Example

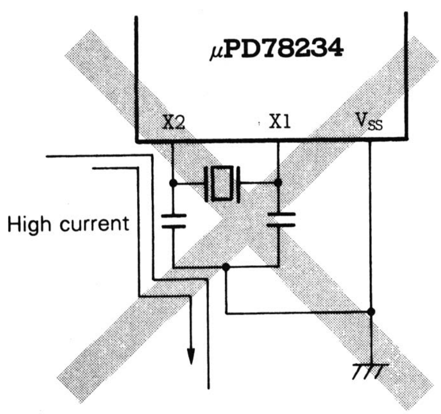
(a) Connected circuit wiring too long



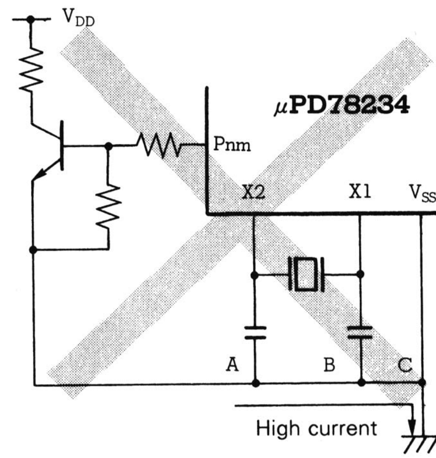
(b) Crossed signal lines



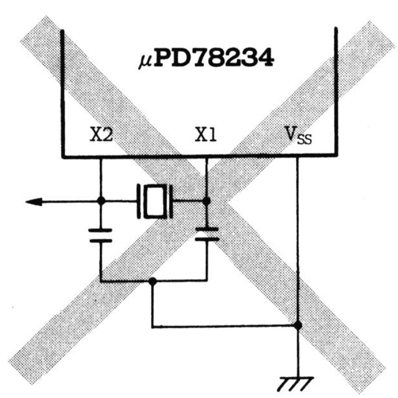
(c) High-current line is closed to signal lines



(d) Current flows through ground line for oscillator circuit (potential at points A, B, and C changes)



(e) Signal is extracted



(2) It is necessary to assure time required for the oscillator circuit to stabilize its operation on power application to the microcomputer or when the operation mode of the microcomputer is changed from the STOP mode. In general, several milliseconds of oscillation stabilization time are necessary when a crystal oscillator is used. When a ceramic oscillator is used, the oscillation stabilization time is several hundred microseconds.
The oscillation stabilization time is determined by the following factors. Make sure that a sufficient time elapses while taking these factors into consideration.

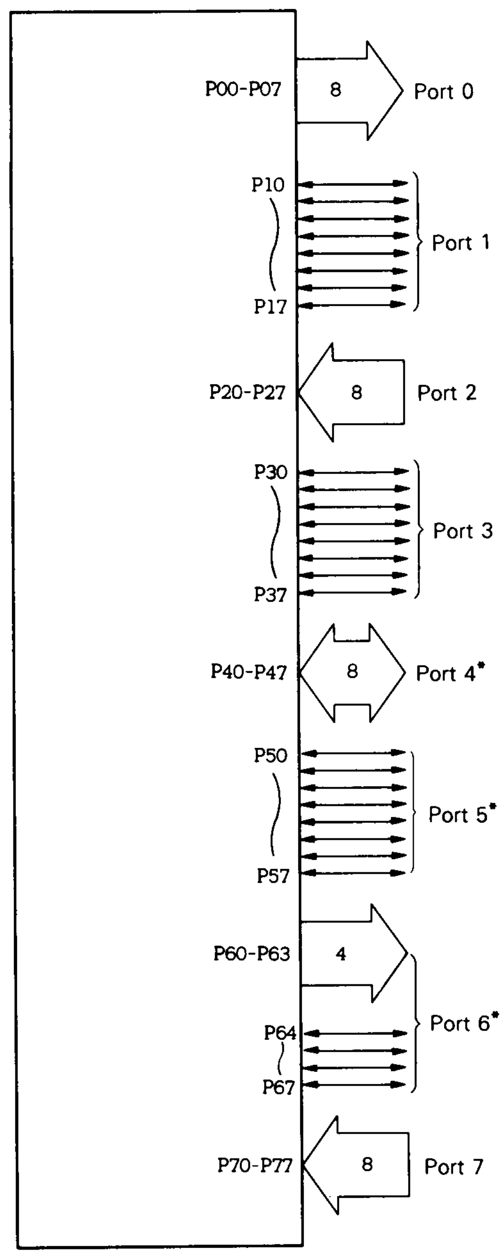
① Power-ON reset : $\overline{\text{RESET}}$ input (reset period)
② When STOP mode is released : (i) $\overline{\text{RESET}}$ input (reset period)
(ii) NMI signal active period + time of timer automatically started
(iii) Time of timer automatically started by valid edge of NMI signal

CHAPTER 5 PORT FUNCTIONS

5.1 Digital I/O Ports

μPD78234 is equipped with the ports shown in Fig. 5-1, with which various control operations can be performed. The functions for each port are listed in Table 5-1. Ports 1 through 6 can be internally connected to pull-up resistors, when so specified by software.

Fig. 5-1 Port Configuration



*: P40 to P47, P50 to P57, P64, and P65 of μPD78233 cannot be used as ports.

Table 5-1 Port Functions

Name	Pin name	Function	Software-specified pull-up resistor
Port 0	P00-P07	Can be specified in output mode or high-impedance state in 8 units of 8 bits each. Can also function as 4-bit real-time output ports (P00-P03 and P04-P07). Can directly drive transistors.	—
Port 1	P10-P17	Can be set in input or output mode in bit units. Can directly drive LEDs	All input pins
Port 2	P20-P27	Input port	In 6 bit units (P22-P27)
Port 3	P30-P37	Can be specified in input or output mode in bit units	All input pins
Port 4	P40-P47	Can be specified in input or output mode in 8 bit units. Can directly drive LEDs	In 8 bit units
Port 5	P50-P57	Can be specified in input or output mode in bit units. Can directly drive LEDs	All input pins
Port 6	P60-P63	Output port	—
	P64-P67	Can be specified in input or output mode in bit units	All input pins
Port 7	P70-P77	Input port	—

Remarks: Ports 4 and 5 and P64 and P65 for μPD78233, respectively, function as the address/data bus, address bus, $\overline{RD}$, and $\overline{WR}$. Therefore, these pins cannot be used to directly drive LEDs or cannot be connected with software pull-up resistors.

Table 5-2 I/O Ports

Port \ I/O	Total	Input	Output	
		Software pull-up resistor	LED direct drive	Transistor direct drive
Input port	16(16)	6(6)	—	—
I/O port	36(18)	36(18)	24(8)	0(0)
Output port	12(12)	—	0(0)	8(8)
Total	64(46)	42(24)	24(8)	8(8)

( ): μPD78233

5.2 Port 0

Port 0 is an 8-bit output port with an output latch and can directly drive transistors. This port can be set in the output mode or high-impedance state by the port 0 mode register (PM0) in 8-bit units.

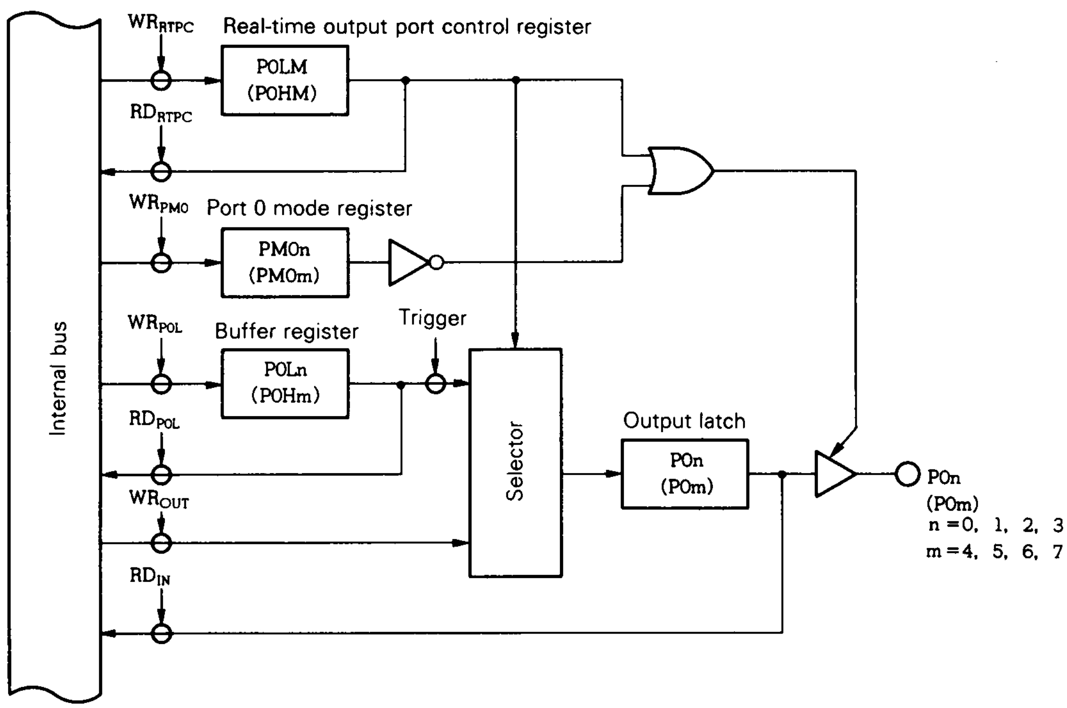
P00 through P03 and P04 through P07 can output the contents of buffer registers (P0L and P0H) at specified time intervals as 4-bit or 8-bit real-time output port. Whether port 0 is used as an ordinary output port or real-time output port is specified by the real-time output port control register (RTPC).

When the $\overline{\text{RESET}}$ signal has been input, the port enters the output high-impedance state, and the output latch contents become undefined.

5.2.1 Hardware configuration

Figure 5-2 shows the port 0 hardware configuration.

Fig. 5-2 Port 0 Configuration



5.2.2 Setting output mode and control mode

Port 0 can be set in output mode by port 0 mode register (PM0), as shown in Fig. 5-3. This register is set by an 8-bit data transfer instruction (the contents of this register cannot be read or written in bit units).

Fig. 5-3 Port 0 Mode Register Format

	7	6	5	4	3	2	1	0	
PM0	PM07	PM06	PM05	PM04	PM03	PM02	PM01	PM00	(FFH when RESET is input)

PM0	Specifies P0n pin mode (n=0-7)
00H	Output mode (output buffer ON)
FFH	High-impedance state (output buffer OFF)
Other than above	Can not be set

When using port 0 as a real-time output port, set the POLM and POHM bits for the real-time output port control register (RTPC) to 1.

When the P0LM and P0HM bits are set, the output buffer for each pin turns ON and the output latch contents are output to the pin, regardless of the PM0 register contents.

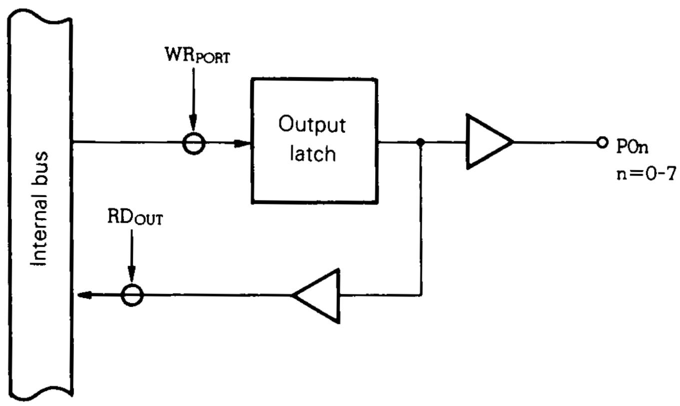
5.2.3 Operation state

Port 0 is an output port.

After this port has been set in the output mode, the output latch becomes valid, and data can be transferred between the output latch and accumulator by a transfer instruction. The output latch contents can be set freely by a logical operation instruction. Once data has been written to the output latch, it is retained until the next data is written to the output latch.

Data cannot be written to the output latch for the port pin specified in the real-time output mode. However, the output latch contents can be read, even in the real-time output port mode.

Fig. 5-4 Port Specified in Output Mode



5.2.4 Pull-up resistor

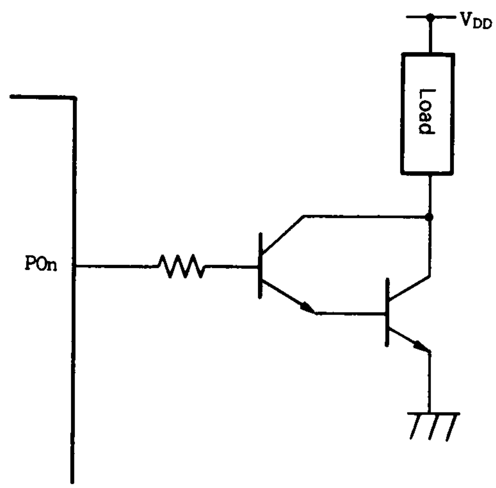
Port 0 is not connected to a pull-up resistor.

5.2.5 Driving transistor

Since the driving capability of port 0 at the high level side of the output buffer is reinforced, the active-high signal for the port can directly drive a transistor.

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

Fig. 5-5 Transistor Driving Example



5.3 Port 1

Port 1 is an 8-bit I/O port with an output latch. This port can be set in the input or output mode by the port 1 mode register (PM1) in bit units. Each pin is connected to a programmable pull-up resistor and can directly drive an LED.

The P10 and P11 pins can be used as PWM output pins, when so specified by the PWM control register (PWMC).

When the $\overline{\text{RESET}}$ signal has been input, the port is set in the input mode (output high-impedance state), and the output latch contents become undefined.

5.3.1 Hardware configuration

Figures 5-6 and 5-7 show the Port 1 hardware configuration.

Fig. 5-6 P10 and P11 (Port 1) Configuration

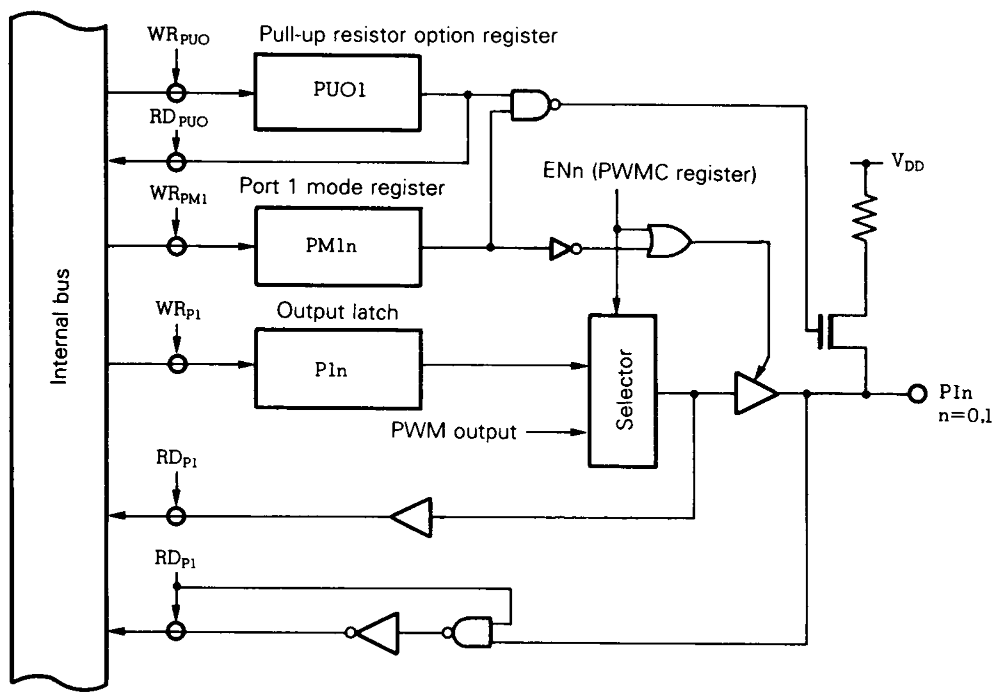
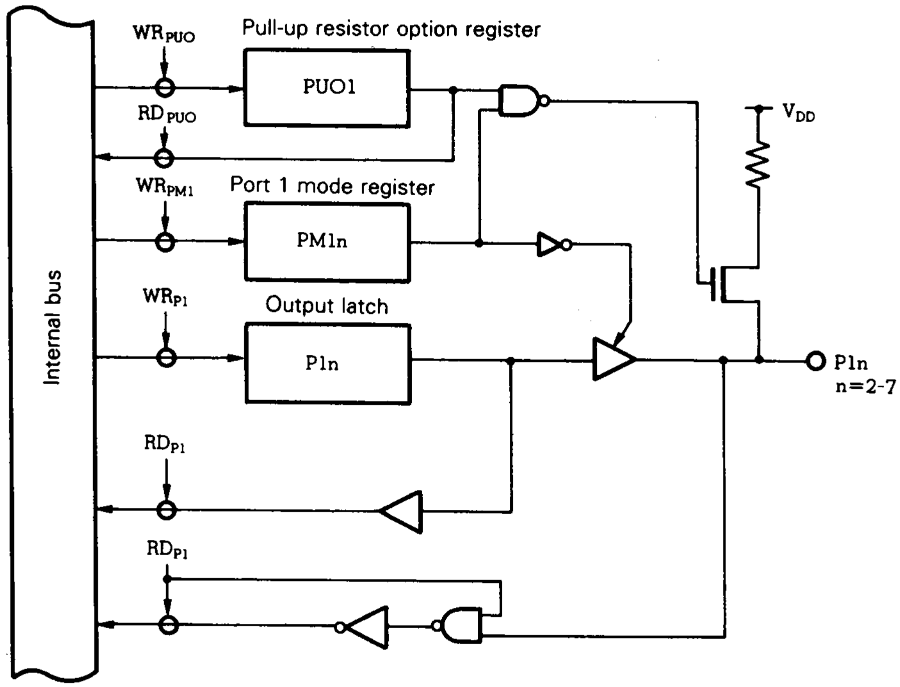


Fig. 5-7 P12 through P17 Configuration (Port 1)



5.3.2 Setting I/O mode and control mode

Port 1 can be set in input or output mode by port 1 mode register (PM1) in bit units, as shown in Fig. 5-8. This register is set by an 8-bit data transfer instruction (the contents of this register cannot be read or written in bit units).

Pins P10 and P11 for this port also function as PWM signal output pins and can be set in the control signal mode by the PWM control register (PWMC), as shown in Table 5-3.

Fig. 5-8 Port 1 Mode Register Format

	7	6	5	4	3	2	1	0	
PM1	PM17	PM16	PM15	PM14	PM13	PM12	PM11	PM10	(FFH when RESET is input)

PM1n	Specifies I/O mode of PM1n pin (n=0-7)
0	Output mode (output buffer ON)
1	Input mode (output buffer OFF)

Table 5-3 Setting PWM Signal Output Function for P10 and P11

Pin	Function	To set PWM signal output function
P10	PWM0	Set PWMC register EN0 bit to 1
P11	PWM1	Set PWMC register EN1 bit to 1

5.3.3 Operation state

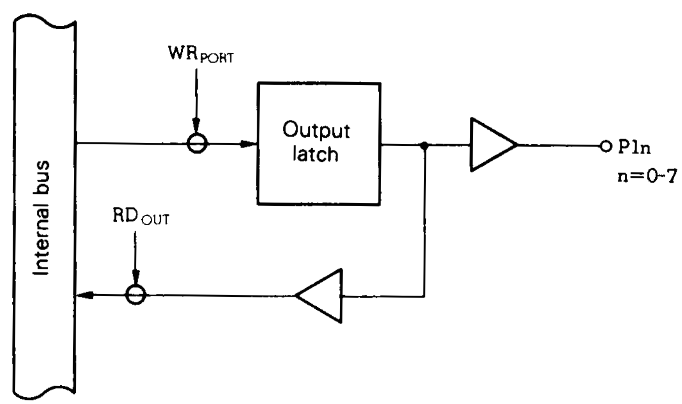
Port 1 is an I/O port. Its P10 and P11 pins can also output PWM signals.

(1) In output mode

The port 1 output latch becomes valid and data can be transferred between the output latch and accumulator by a transfer instruction. The output latch contents can be set freely by a logical operation instruction. Once data has been written to the output latch, it is retained until the next data is written to the output latch*.

***:** This also applies when the other bits of the same port are manipulated by a bit manipulation instruction.

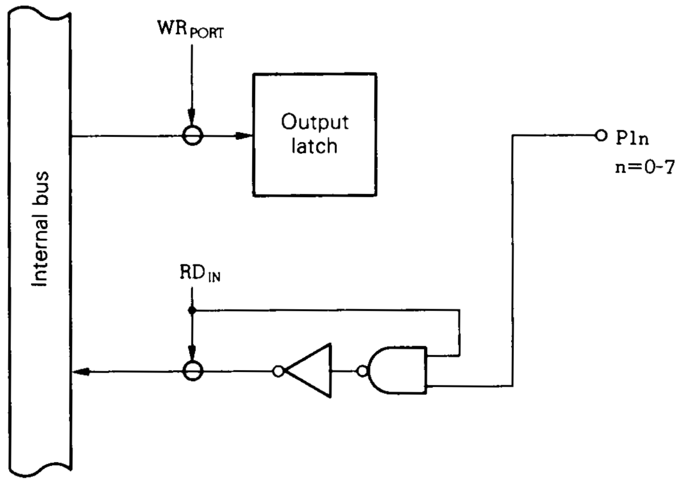
Fig. 5-9 Port Specified in Output Mode



(2) In input mode

The port pins level can be loaded to the accumulator by a transfer instruction. In this case, data can be written to the output latch, and data transferred from the accumulator by a transfer instruction is stored in all the output latches, regardless of whether the port is in the input or output mode. However, the output buffer for the bit specified in the input mode is in the high-impedance state. Therefore, the output buffer contents are not output to the port pin (the output latch contents for the bit currently specified in the input mode are output to the port pin, when the bit mode is changed to the output mode). The contents of the output latch of the bit specified in the input mode cannot be loaded to the accumulator.

Fig. 5-10 Port Specified in Input Mode

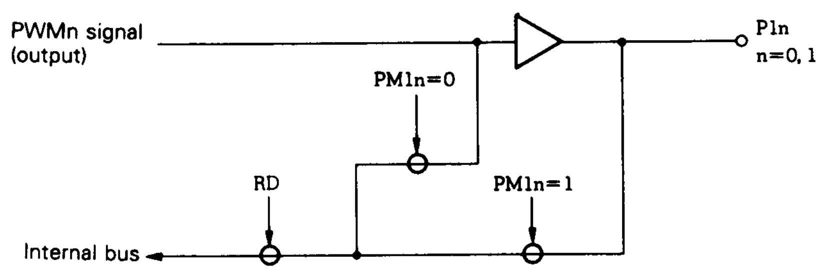


Caution: A bit manipulation instruction, although its ultimate purpose is to manipulate only 1 bit, accesses a port in 8 bit units. If a bit manipulation instruction is executed to manipulate a port which can function as either an input or an output port, the contents of the output latch for the port pins, specified in the input mode or control mode, become undefined (except the bit that has been manipulated by the SET1 or CLR1 instruction, etc.). If a port to be manipulated has a bit, whose mode is to be switched between input and output, this must be taken into consideration. The same applies, when manipulating the port with any other 8-bit arithmetic operation instruction.

(3) In control mode

Port 1 can be used as a PWM signal output port, when the ENn bit (n = 0 or 1) for the PWM control register (PWMC) is set to 1, regardless of the port 1 mode register (PM1) setting. When pins P10 and P11 for port 1 are used to output the PWM signals, the PWM signal states can be checked by executing a port read instruction.

Fig. 5-11 To Output PWM Signal



(a) When port outputs control signals

When a port read instruction is executed, with the port 1 mode register (PM1n) set to 1, the control signal pin level can be read.

When a port read instruction is executed, with PM1n reset to 0, the control signal status in μPD78234 can be read.

5.3.4 Internal pull-up resistor

Port 1 can be connected to internal pull-up resistors. When these internal resistors are used, the number of external components and mounting area to be occupied by the external components can be reduced.

Whether the pull-up resistors are used can be specified in bit units by the PUO1 bit for the pull-up resistor option register (PUO) and port 1 mode register (PM1). When the PM01 is 1, the internal pull-up resistor for the pin set in the input mode by the PM1 (PM1n = 1, n = 0-7) becomes valid.

Specification to connect the pull-up resistor is valid, even for the pin specified in the PWM signal output mode (i.e., the pull-up resistor is connected to the pin that outputs the PWM signal). Therefore, to prevent connecting the pull-up resistor to the PWM signal output pin, clear the corresponding bit content for PM1 to 0 (output mode).

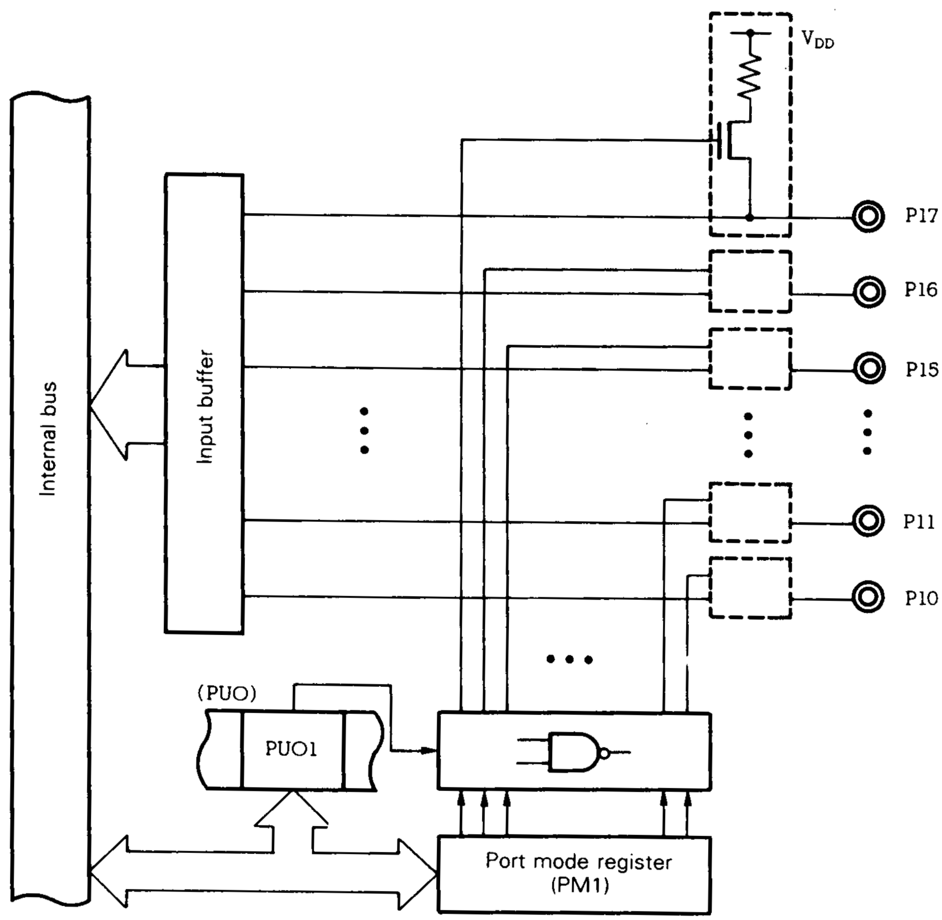
Fig. 5-12 Pull-up Resistor Option Register Format

	7	6	5	4	3	2	1	0	
PUO	0	PUO6	PUO5	PUO4	PUO3	PUO2	PUO1	0	(00H when RESET is input)

PUO1	Specifies pull-up resistor connection of port 1
0	Not connected
1	Connected

Remarks: Set the PUO register to 00H to reduce the power dissipation, before the STOP mode is set.

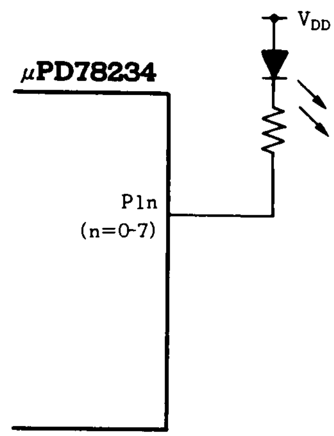
Fig. 5-13 Specifying Pull-up Resistor Connection (Port 1)



5.3.5 Direct drive for LED

The driving capability for the low level side of the port output buffer is reinforced, so that port 1 can directly drive an LED with an active-low signal. Figure 5-14 shows an LED connection to the port.

Fig. 5-14 LED Direct Drive



5.4 Port 2

Port 2 is an 8-bit input port. P22 through P27 for this port are connected to software-programmable pull-up resistors. In addition to functioning as an input port, some port 2 pins also function as an external interrupt signal pin and other control signal pins, as shown in Table 5-4. All the eight pins are Schmitt trigger input pins to prevent malfunctioning due to noise.

Table 5-4 Port 2 Operation Modes

Port	Function
P20	Input port/NMI input*
P21	Input port/INTP0 input/CR11 capture trigger input/output 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/A/D converter external trigger input
P27	Input port/SI input

***:** NMI input is accepted, regardless of whether interrupts are enabled or disabled.

(a) Functions as port pin

The pin level can always be read, or testing is possible, regardless of the shared pin operation.

(b) Functions as control signal input pin

(i) NMI (Non-maskable interrupt)

This is an external nonmaskable interrupt request input pin. Rising edge detection or falling edge detection can be specified by the external interrupt mode register (INTM0).

(ii) INTP0-INTP5 (Interrupt from peripherals)

These are external interrupt request input pins. When the effective edge specified by the external interrupt mode register (INTM0, INTM1) is detected from INTP0-INTP5, an interrupt is generated (refer to **Chapter 13 Edge Detection Function**).

In addition, INTP0-INTP3 and INTP5 can be used as external trigger input pins for various functions as described below.

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

(iii) CI (Clock input)

An 8-bit timer/counter 2 external clock input pin.

(iv) ASCK (Asynchronous serial clock)

An external baud rate clock input pin.

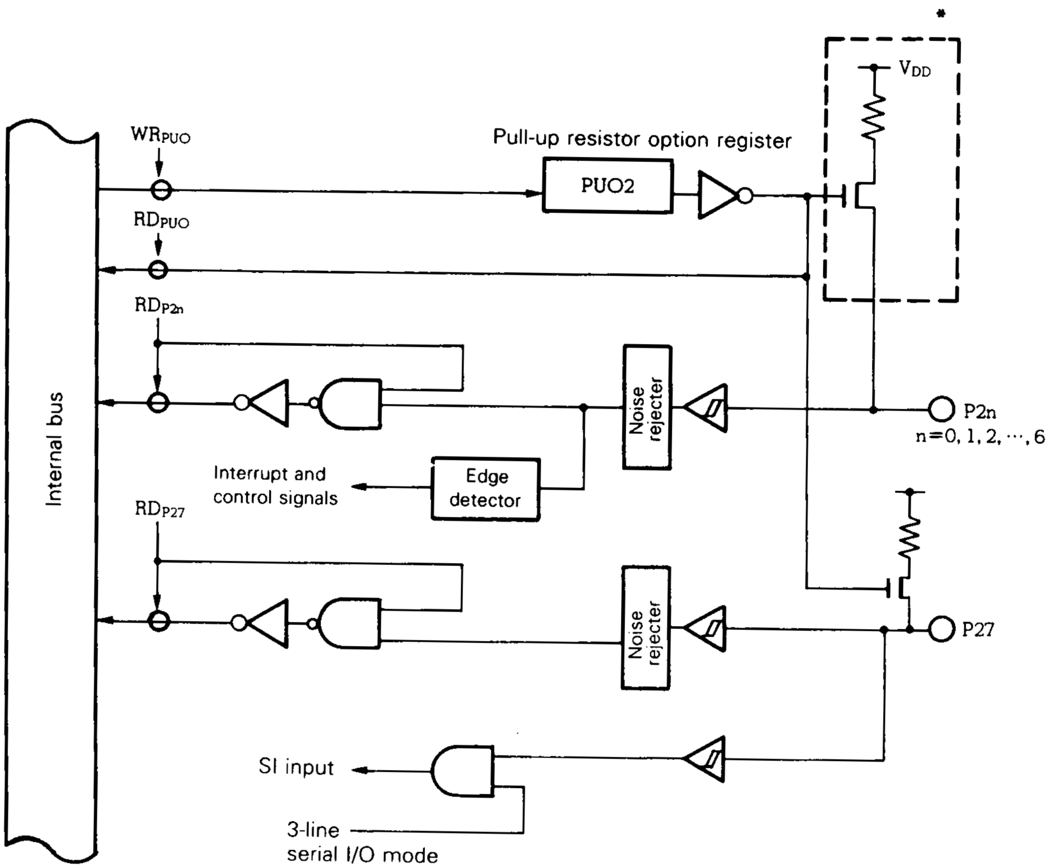
(v) SI (Serial input)

A serial data input (in 3-line serial I/O mode).

5.4.1 Hardware configuration

Figure 5-15 shows the port 2 hardware configuration.

Fig. 5-15 Port 2 Configuration



*: Pins P20 and P21 are not provided with a circuit indicated by the dotted line in the above figure.

5.4.2 Setting I/O mode and control mode

Port 2 is an input port and thus is not provided with a register that sets the input mode.

In addition, the port can always input control signals. Which signal is to be used is determined by the control register for each internal hardware.

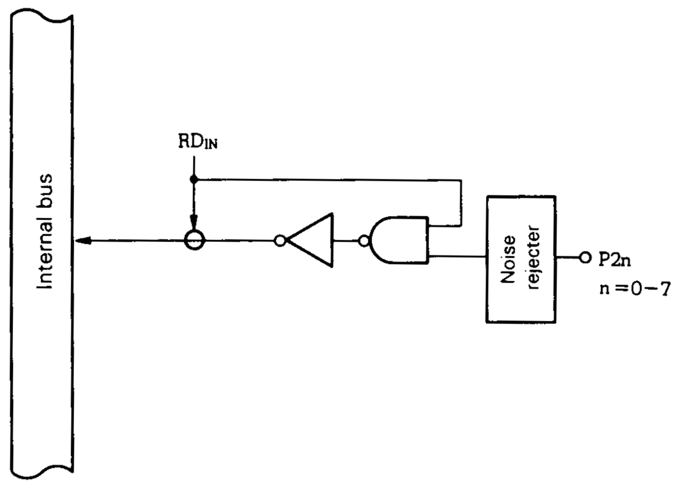
5.4.3 Operation state

Port 2 is an input port. Its pin levels can always be read or tested.

For P20 to P27, the levels after noise elimination will be read out during reading or testing.

Refer to **CHAPTER 13 EDGE DETECTION FUNCTION** for details of noise elimination.

Fig. 5-16 Port Specified in Input Mode



Caution: When reading or testing port 2 in the in-circuit emulator, the pin level before noise elimination will be read out.

5.4.4 Pull-up resistor

Pins P22 through P27 for Port 2 can be internally connected to a pull-up resistor, which can contribute to reduction in the number of external components and of the mounting area, which would otherwise be taken up by the external components.

Whether the internal pull-up resistor is used can be specified by the PUO2 bit of the pull-up resistor option register (PUO) in 6 bit units (but cannot be specified in bit units).

Pins P20 and P21 cannot be connected to a pull-up resistor.

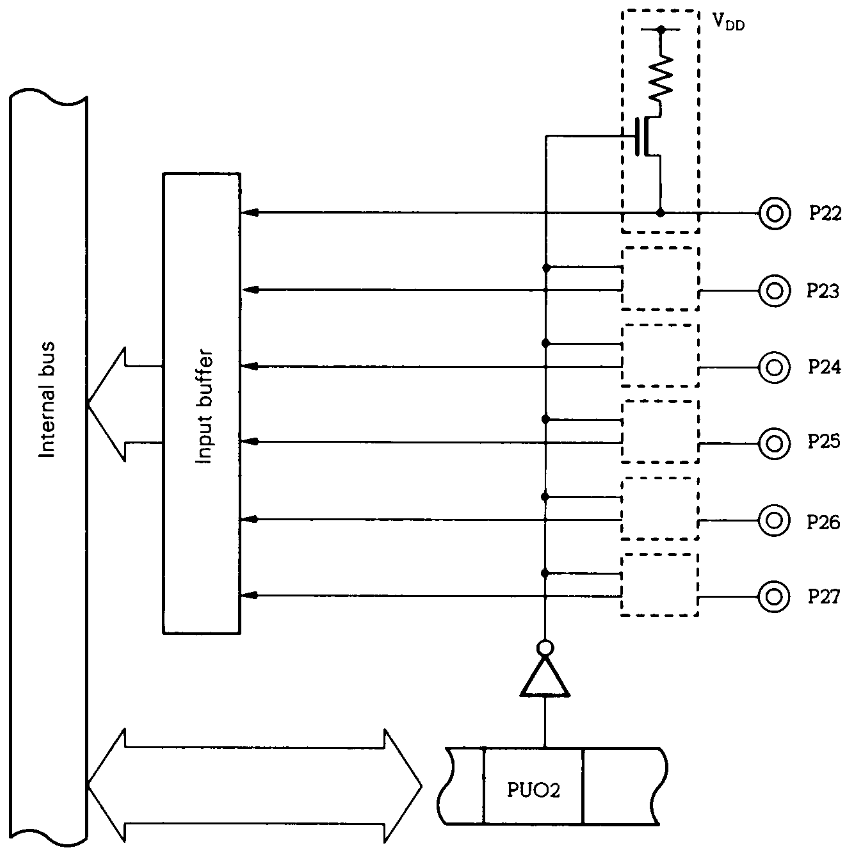
Fig. 5-17 Pull-up Resistor Option Register Format

	7	6	5	4	3	2	1	0	
PUO	0	PUO6	PUO5	PUO4	PUO3	PUO2	PUO1	0	(00H when RESET is input)

PUO2	Specifies pull-up resistor connection to port 2
0	Not connected
1	Connected to P22-P27

Remarks: Set the PUO resistor contents to 00H, before the STOP mode is set, to reduce power dissipation.

Fig. 5-18 Specifying Connection for Pull-up Resister (Port 2)



Caution: Pins P22 through P26 are not pulled up immediately after the RESET signal is input and the functions of the pins multiplexed to these pins (INTP1 through INTP5) may set interrupt flags. Therefore, specify pull-up resistors connection by the initialization routine and then clear the interrupt flags.

5.5 Port 3

Port 3 is an 8-bit I/O port with an output latch. This port can be set in the input or output mode in bit units by the port 3 mode register (PM3). Each pin is connected to a software-programmable pull-up resistor. In addition to functioning as an I/O port, some port 3 pins also function as control signal pins, when so specified by the port 3 mode control register (PMC3), as shown in Table 5-5. The levels for all the pins can be read or tested, regardless of the multiplexed pins functions.

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

Table 5-5 Port 3 Operation Modes (n = 0-7)

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/SB0 I/O
P34		TO0 output
P35		TO1 output
P36		TO2 output
P37		TO3 output

(a) Port mode

Ports specified for port mode by the PMC3 register can be specified for input/output in bit units by the port 3 mode register (PM3).

(b) Control signal input/output mode

These pins can be specified for control pins in bit units by the PMC3 register setting.

(i) RxD (Receive Data)

An asynchronous serial interface serial data input pin.

(ii) TxD (Transmit Data)

An asynchronous serial interface serial data output pin.

(iii) $\overline{\text{SCK}}$ (Serial Clock)

The serial clock input/output pin for clock synchronized serial interface.

(iv) SO (Serial Output)/SB0 (Serial Bus)

SO is the serial data output pin (3-line serial I/O mode). SB0 is the serial bus input/output in the SBI mode.

Remarks: Bit 3 (P33) of port 3 is reserved as "SB0" in the NEC assembler package. In the C compiler this is defined in header file sfrbit.h.

(v) TO0-TO3 (Timer output)

Timer output pins.

5.5.1 Hardware configuration

Figures 5-19 through 5-22 show the Port 3 hardware configurations.

Fig. 5-19 P30 Configuration (Port 3)

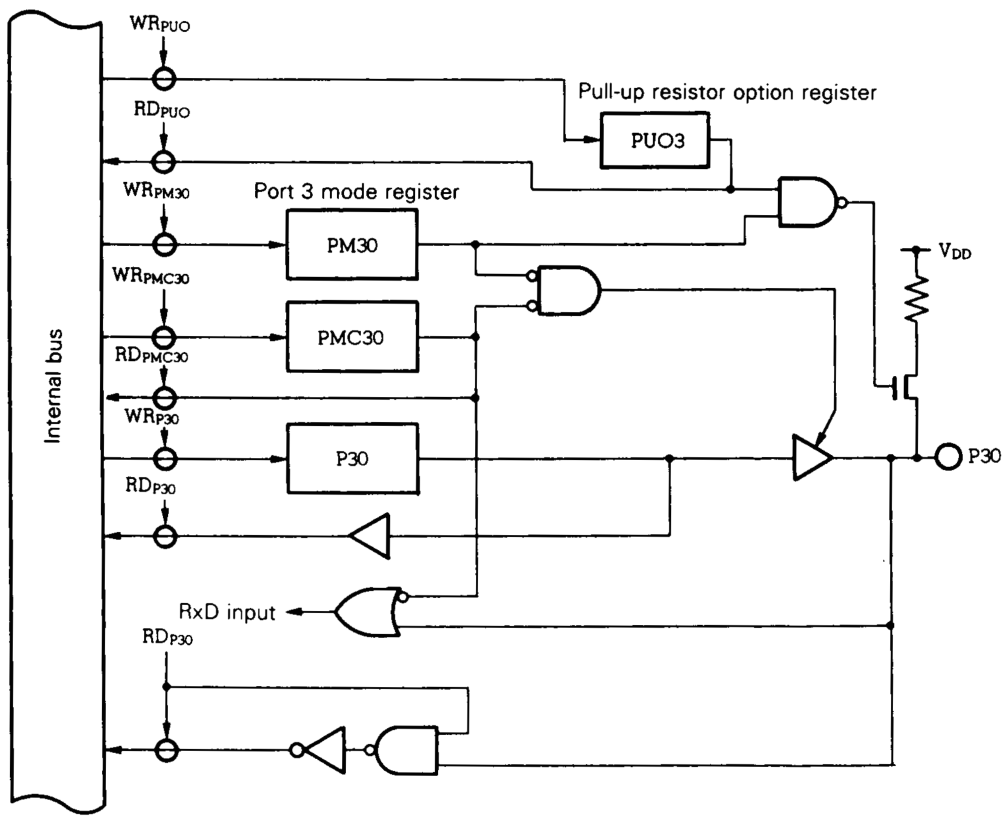


Fig. 5-20 Configurations for P31 and P34 through P37 (Port 3)

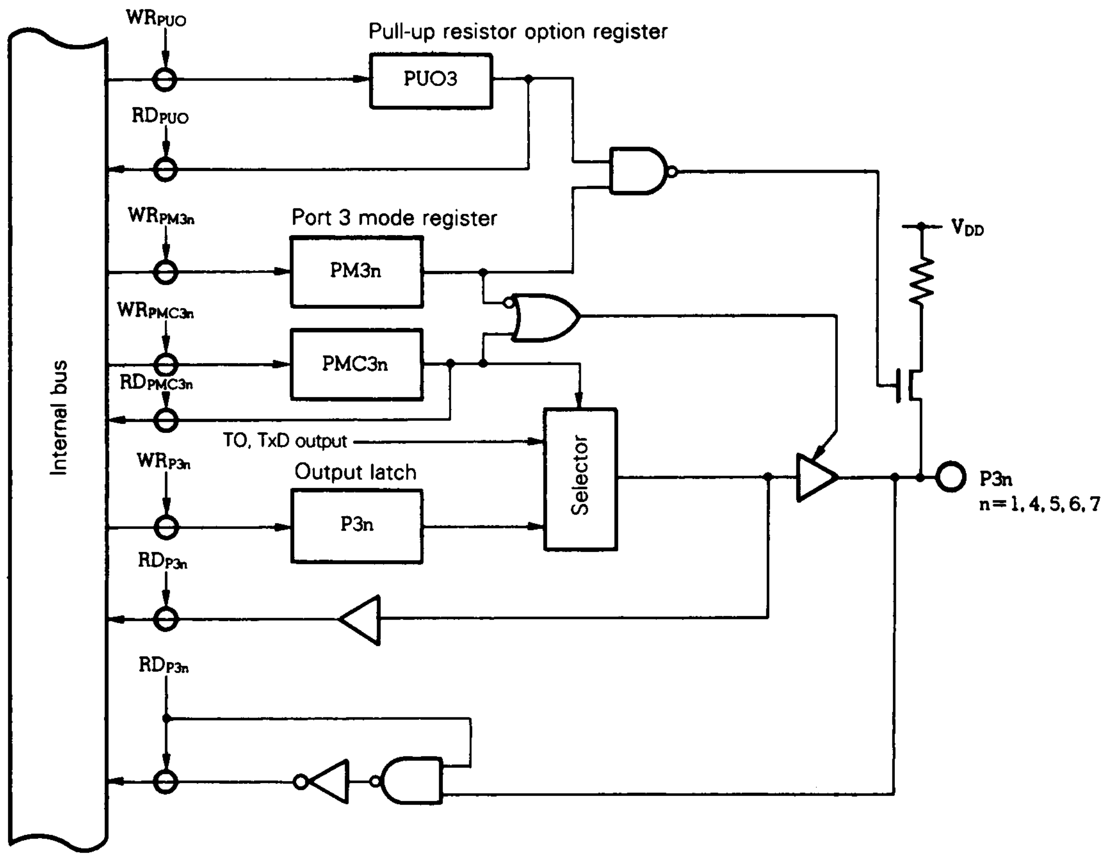


Fig. 5-21 P32 Configuration (Port 3)

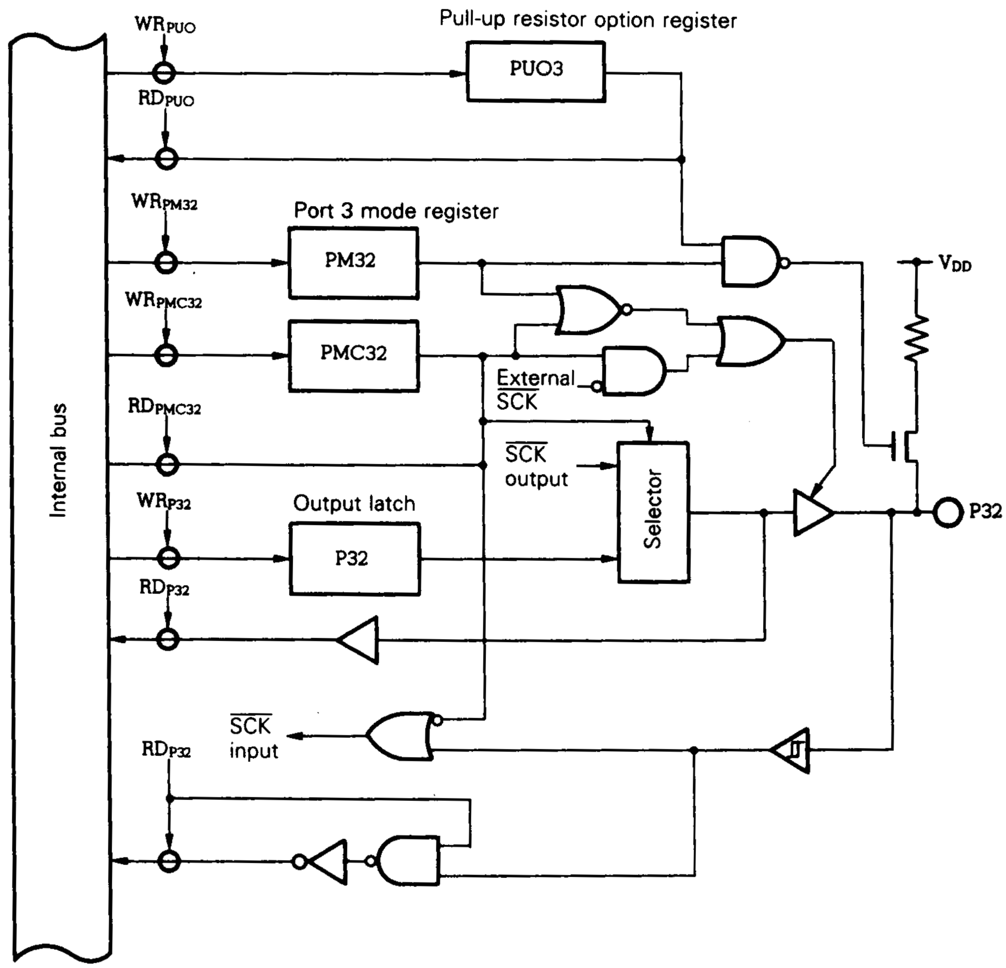
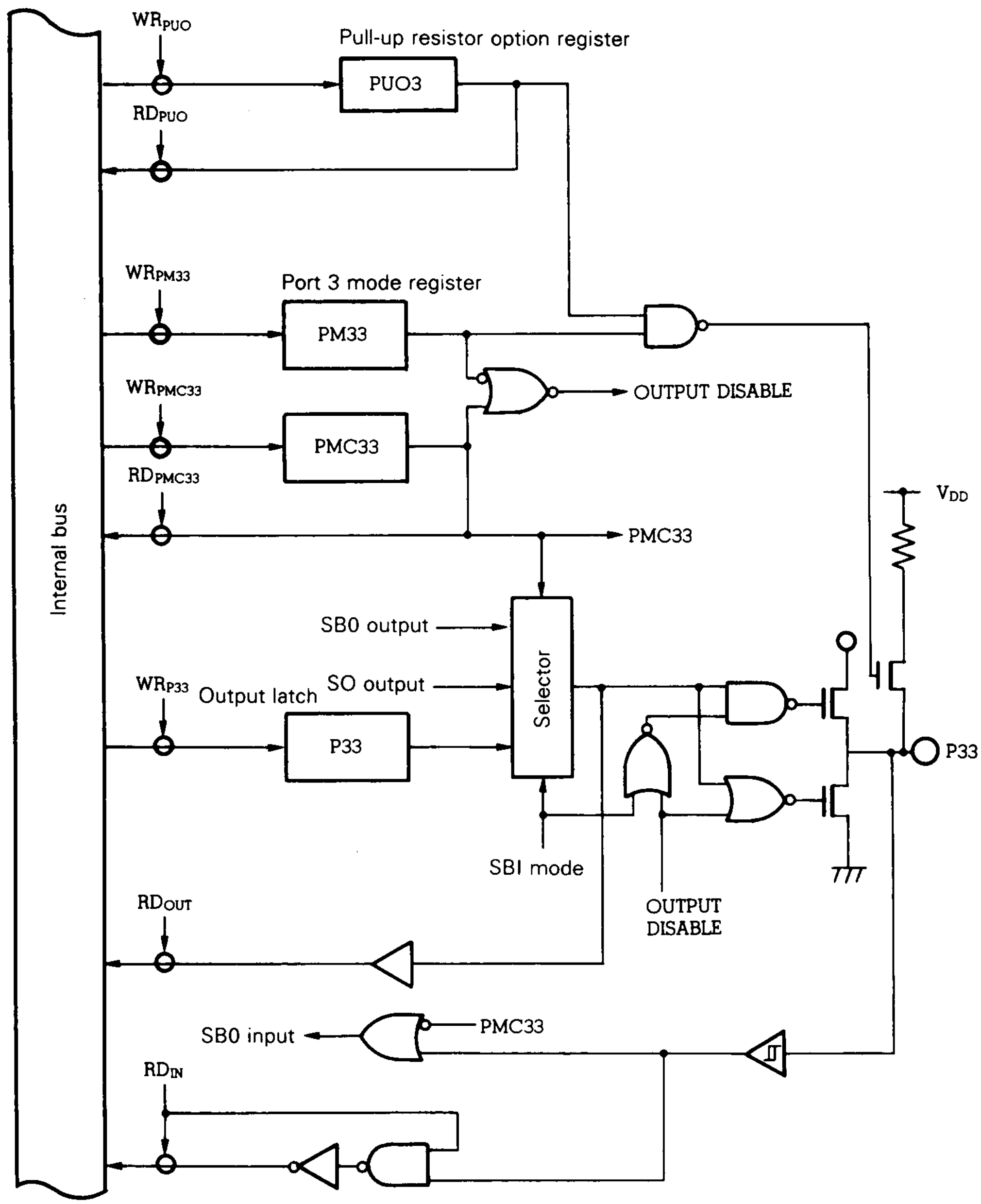


Fig. 5-22 P33 Configuration (Port 3)



5.5.2 Setting of I/O mode and control mode

Port 3 can be set in input or output mode by Port 3 mode register (PM3) in bit units, as shown in Fig. 5-23. This register is set by an 8-bit data transfer instruction (the contents of this register cannot be read or written in bit units).

In addition to the I/O port function, Port 3 also has control signal I/O functions, which can be specified by Port 3 mode control register (PMC3), as shown in Fig. 5-24.

Fig. 5-23 Port 3 Mode Register Format

	7	6	5	4	3	2	1	0	
PM3	PM37	PM36	PM35	PM34	PM33	PM32	PM31	PM30	(FFH when RESET is input)

PM3n	Specifies I/O mode of PM3n pin (n=0-7)
0	Output mode (output buffer ON)
1	Input mode (output buffer OFF)

Fig. 5-24 Port 3 Mode Control Register (PMC3) Format

	7	6	5	4	3	2	1	0	
PMC3	PMC37	PMC36	PMC35	PMC34	PMC33	PMC32	PMC31	PMC30	(00H when RESET is input)

PMC3	Specifies P30 control mode
0	I/O port mode
1	RxD input mode

PMC31	Specifies P31 control mode
0	I/O port mode
1	TxD output mode

PMC32	Specifies P32 control mode
0	I/O port mode
1	$\overline{SCK}$ I/O mode

PMC33	Specifies P33 control mode
0	I/O port mode
1	SO output mode/SB0 I/O mode

PMC3n	Specifies P3n control mode (n=4-7)
0	I/O port mode
1	TOn output mode (n=0-3)

5.5.3 Operation state

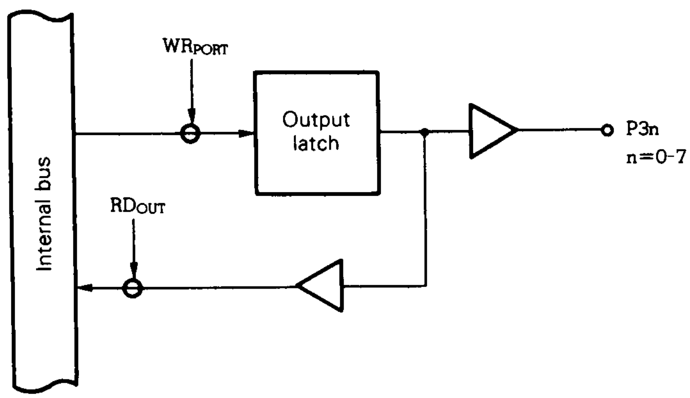
Port 3 is an I/O port multiplexed with control pins.

(1) In output mode

The Port 3 output latch becomes valid and data can be transferred between the output latch and accumulator by a transfer instruction. The output latch contents can be set freely by a logical operation instruction. Once data has been written to the output latch, it is retained until the next data is written to the output latch*.

***:** This also applies, when the other bits of the same port are manipulated by a bit manipulation instruction.

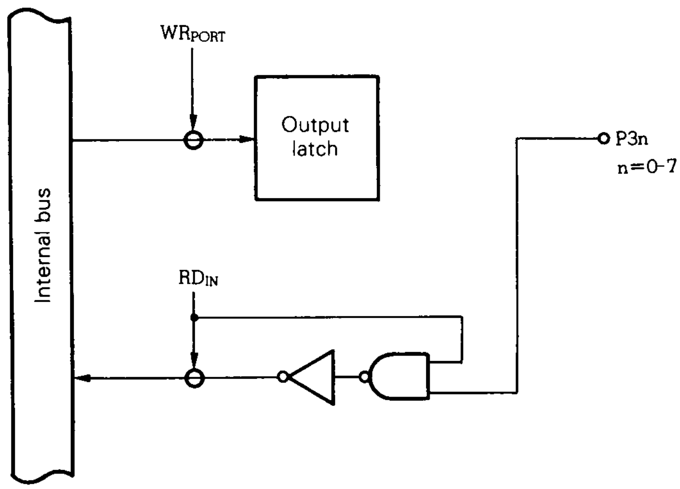
Fig. 5-25 Port Specified in Output Mode



(2) In input mode

The port pins level can be loaded to the accumulator by a transfer instruction. In this case, data can be written to the output latch, and data transferred from the accumulator by a transfer instruction is stored in all the output latches, regardless of whether the port is in the input or output mode. However, the output buffer for the bit specified in the input mode is in the high-impedance state. Therefore, the output buffer contents are not output to the port pin (the output latch contents for the bit currently specified in the input mode are output to the port pin, when the bit mode is changed to the output mode). The output latch contents for the bit specified in the input mode cannot be loaded to the accumulator.

Fig. 5-26 Port Specified in Input Mode

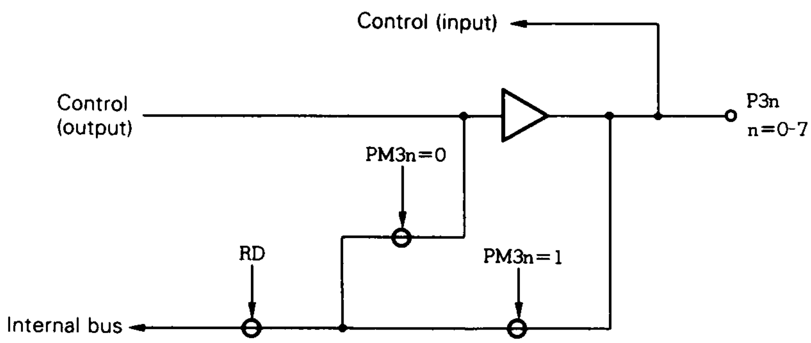


Caution: A bit manipulation instruction, although its ultimate purpose is to manipulate only 1 bit, accesses a port in 8 bit units. If a bit manipulation instruction is executed to manipulate a port, which can function as either an input or an output port, the contents of the ouput latch for the port pins, specified in the input mode or control mode, become undefined (except the bit that has been manipulated by the SET1 or CLR1 instruction). If a port to be manipulated has a bit, whose mode is to be switched between input and output, this must be taken into consideration. The same applies, when manipulating the port with another 8-bit arithmetic operation instruction.

(3) In control mode

Port 3 can be set in the control signal input/output mode in 1-bit units by setting the necessary bit of the port 3 mode control register (PMC3) to 1, regardless of the current setting of the port 3 mode register (PM3). When each pin of the port is used as a control signal pin, the control signal status can be checked by executing a port read instruction.

Fig. 5-27 Port Specified in Control Mode



(a) In control signal output mode

When a bit for the port 3 mode register (PM3) is set to 1, the corresponding control signal pin level can be checked by executing a port read instruction.

The μPD78234 internal control signal status can be checked by executing a port read instruction, when the corresponding bit for the PM3 is cleared to 0.

Remarks: Bit 3 (P33) of port 3 is reserved as “SB0” in the NEC assembler package. In the C compiler, this is defined in header file sfrbit.h.

(b) In control signal input mode

The level for a control signal pin can be checked by executing a port read instruction, only when the corresponding bit for the port 3 mode register (PM3) is set to 1.

5.5.4 Internal pull-up resistor

Port 3 can be connected to internal pull-up resistors. When these internal resistors are used, the number of external components and mounting area to be occupied by the external components can be reduced.

Whether the pull-up resistors are used can be specified in bit units by the PUO3 bit for the pull-up resistor option register (PUO) and port 3 mode register (PM3). When the PUO3 bit is 1, the internal pull-up resistor for the pin set in the input mode by the PM3 register (PM3n = 1, n = 0-7) becomes valid.

The pin specified in the control signal mode can also be connected to a pull-up resistor (i.e., a pull-up resistor can also be connected to a control signal output pin). To prevent connecting a pull-up resistor in the control signal mode, therefore, clear the content of the corresponding bit for PM3 to 0 (output mode).

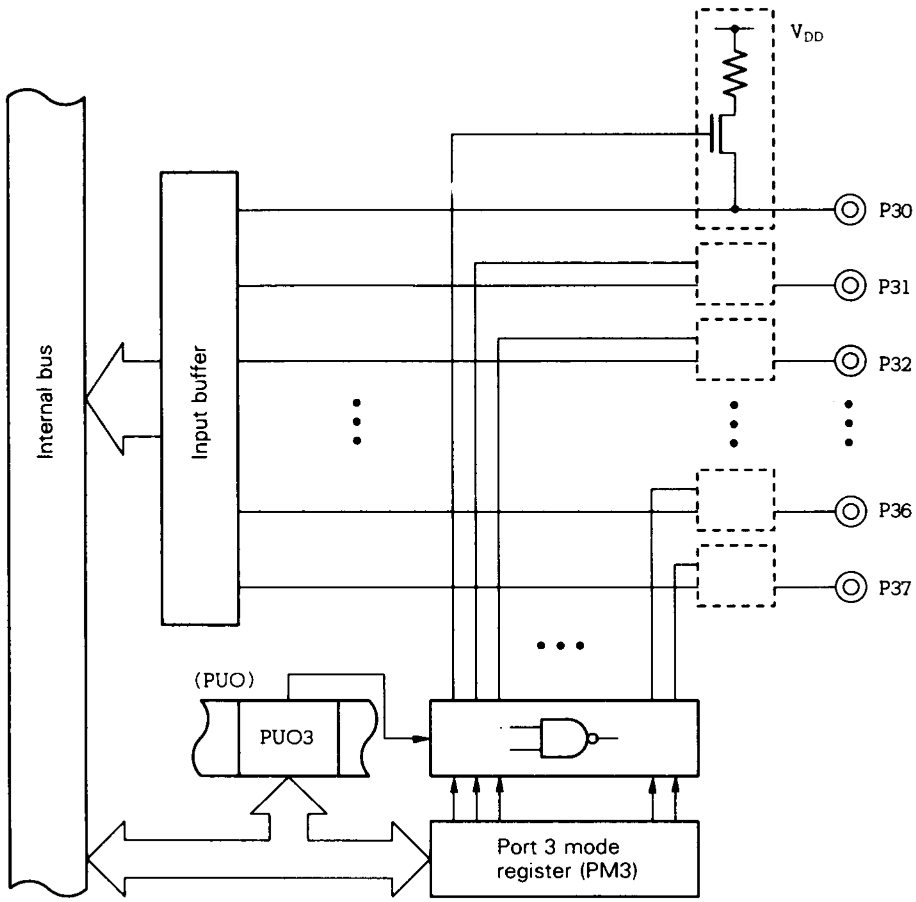
Fig. 5-28 Pull-up Resistor Option Register Format

	7	6	5	4	3	2	1	0	
PUO	0	PUO6	PUO5	PUO4	PUO3	PUO2	PUO1	0	(00H when RESET is input)

PUO3	Specifies pull-up resistor connection of port 3
0	Not connected
1	Connected

Remarks: Set the PUO register to 00H to reduce the power dissipation, before the STOP mode is set.

Fig. 5-29 Pull-up Resistor Connection (Port 3)



5.6 Port 4

Port 4 is an 8-bit I/O port with an output latch. This port can be set in the input or output mode in 8-bit units by the memory expansion mode register (MM). Each pin is connected to a software-programmable pull-up resistor, and can directly drive an LED.

When the external memory or I/O is expanded, this port serves as a time-division address/data bus (AD0 through AD7).

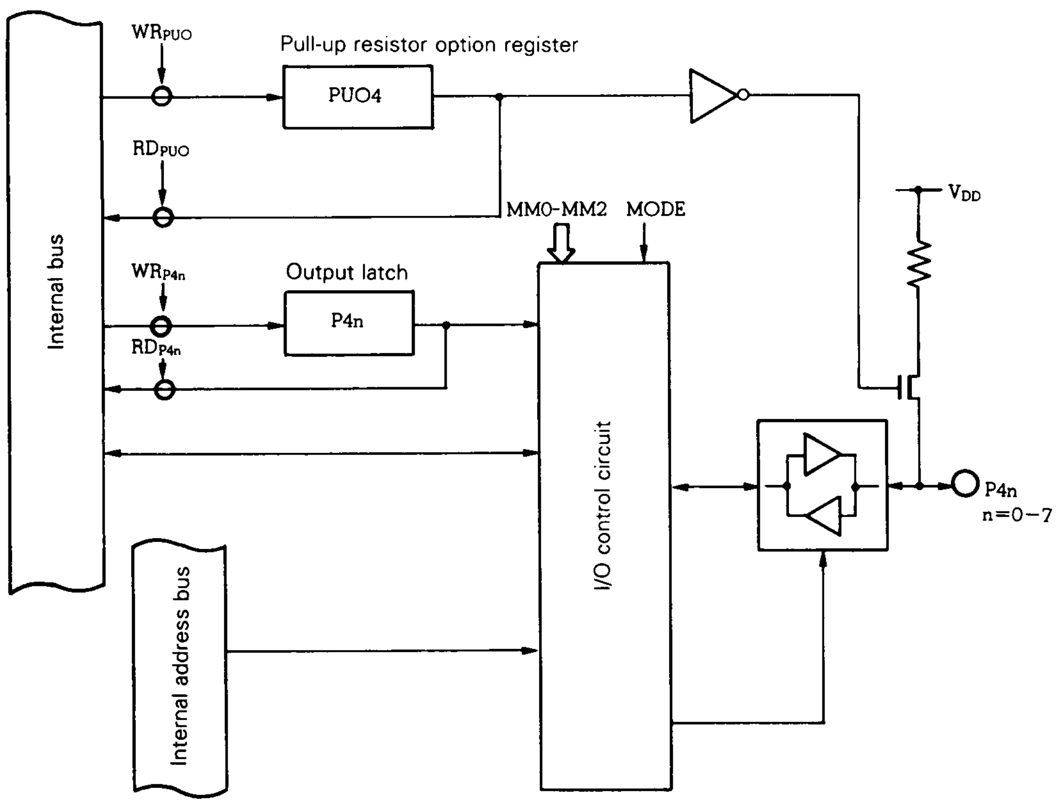
Port 4 for μPD78233 functions as the time-division address/data bus (AD0 through AD7) only.

When the $\overline{\text{RESET}}$ signal has been input, the port is set in the input mode (output high-impedance), and the output latch contents become undefined.

5.6.1 Hardware configuration

Figure 5-30 shows the Port 4 hardware configuration.

Fig. 5-30 Port 4 Configuration



5.6.2 Setting I/O mode and control mode

The Port 4 operation modes are specified by the memory expansion mode register (MM: refer to Figs. 14-4 through 14-6) as shown in Table 5-6.

Table 5-6 Port 4 Operation Modes

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

***:** The MODE pin for μPD78P238 cannot be set to 1.

Note that Port 4 for μPD78233 functions only as an address/data bus (AD0 through AD7).

5.6.3 Operation state

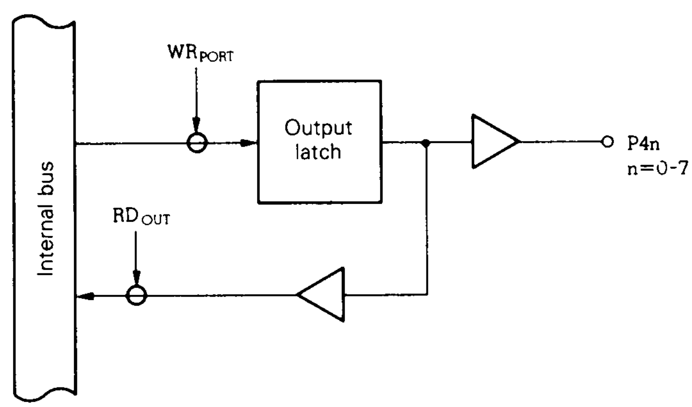
Port 4 is an I/O port multiplexed with an address/data bus (AD0 through AD7).

(1) In output mode

The Port 4 output latch becomes valid and data can be transferred between the output latch and accumulator by a transfer instruction. The output latch contents can be set freely by a logical operation instruction. Once data has been written to the output latch, it is retained until the next data is written to the output latch*.

***:** This also applies, when the other bits of the same port are manipulated by a bit manipulation.

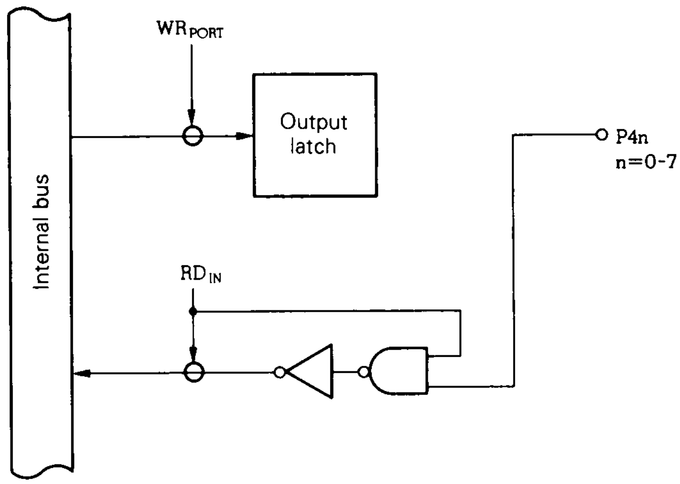
Fig. 5-31 Port Specified in Output Mode



(2) In input mode

The level for the port pins can be loaded to the accumulator by a transfer instruction. In this case, data can be written to the output latch, and data transferred from the accumulator by a transfer instruction, etc. is stored in all the output latches, regardless of whether the port is in the input or output mode. However, the output buffer for the bit specified in the input mode is in the high-impedance state. Therefore, the output buffer contents are not output to the port pin (the output latch contents for the bit currently specified in the input mode are output to the port pin, when the bit mode is changed to the output mode). The output latch contents for the bit specified in the input mode cannot be loaded to the accumulator.

Fig. 5-32 Port Specified in Input Mode



Caution: A bit manipulation instruction, although its ultimate purpose is to manipulate only 1 bit, accesses a port in 8 bit units. If a bit manipulation instruction is executed to manipulate a port, which can function as either an input or an output port, the contents of the output latch for the port pins, specified in the input mode, become undefined (except the bit that has been manipulated by the SET1 or CLR1 instruction). If a port to be manipulated has a bit, whose mode is to be switched between input and output, this must be taken into consideration. The same applies, when manipulating the port with another 8-bit arithmetic operation instruction.

(3) In address/data bus (AD0-AD7) mode

Port 4 is automatically used as an address/data bus, when an external memory is accessed. At this time, do not execute an I/O instruction that manipulates Port 4.

5.6.4 Internal pull-up resistor

Port 4 can be connected to internal pull-up resistors. When these internal resistors are used, the number of external components and mounting area to be occupied by the external components can be reduced.

Whether the pull-up resistors are used can be specified in 8 bit units by the PUO4 bit for the pull-up resistor option register (PUO) (a pull-up resistor connection cannot be specified bitwise).

The pull-up resistors can be connected regardless of whether port 4 is in the input or output mode.

Fig. 5-33 Pull-up Resistor Option Register Format

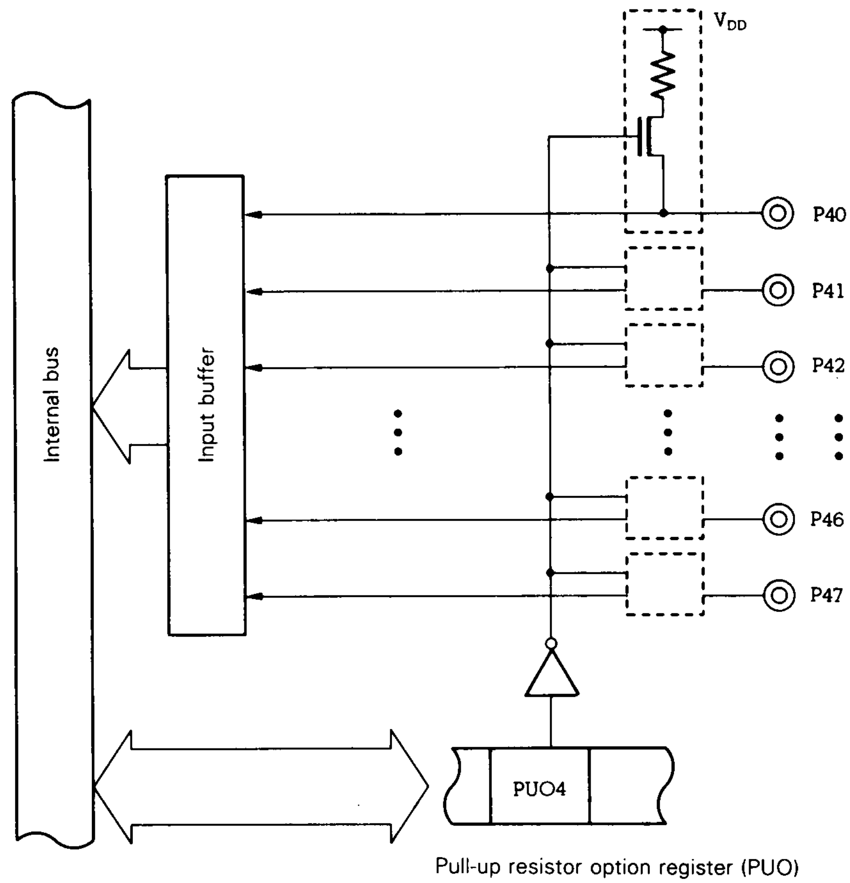
	7	6	5	4	3	2	1	0	
PUO	0	PUO6	PUO5	PUO4	PUO3	PUO2	PUO1	0	(00H when RESET is input)

PUO4	Specifies Port 4 pull-up resistor connection
0	Not connected
1	Connected

Caution: For the μPD78233, port 4 is used as the address/data bus, "0" must always be set to PUO4, to prevent connecting the internal pull-up resistor. Also clear PUO4 for μPD78234, when port 4 for μPD78234 is used as an address/data bus.

Remarks: Set the PUO register to 00H to reduce the power dissipation, before the STOP mode is set.

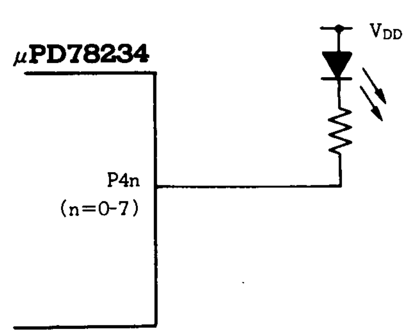
Fig. 5-34 Pull-up Resistor Connection (Port 4)



5.6.5 Direct drive for LED

The driving capability for the low level side of the port output buffer is reinforced, so that Port 4 can directly drive an LED with an active-low signal. Figure 5-35 shows an example of an LED connection to the port.

Fig. 5-35 Direct Drive for LED



5.7 Port 5

Port 5 is an 8-bit I/O port with an output latch. This port can be set in the input or output mode in 1-bit units by the port 5 mode register (PM5). Each pin is connected to a software-programmable pull-up resistor, and can directly drive an LED.

When the external memory or I/O is expanded, P50 through P57 constitute an address bus (A8 through A15).

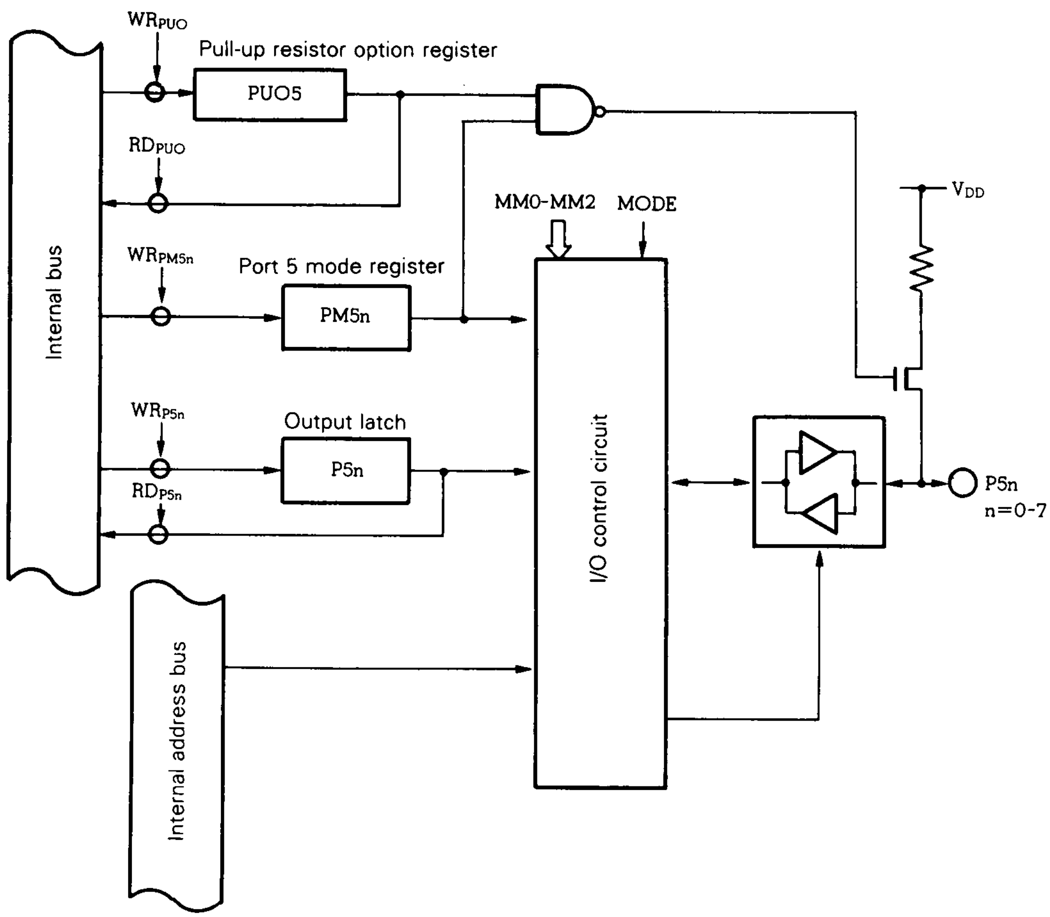
Port 5 for μPD78233 functions as the address bus (A8 through A15) only.

When the $\overline{\text{RESET}}$ signal has been input, the port is set in the input mode (output high-impedance), and the output latch contents become undefined.

5.7.1 Hardware configuration

Figure 5-36 shows the Port 5 hardware configuration.

Fig. 5-36 Port 5 Configuration



5.7.2 Setting I/O mode and control mode

The Port 5 input/output mode is specified in bit units by the Port 5 mode register (PM5), as shown in Fig. 5-37. This register can be set by an 8-bit data transfer instruction (but cannot be manipulated or read in bit units).

Port 5 can also be set in the control signal mode, as shown in Table 5-7, by the memory expansion mode register (MM: refer to **Fig. 15-1**).

Fig. 5-37 Port 5 Mode Register Format

	7	6	5	4	3	2	1	0
PM5	PM57	PM56	PM55	PM54	PM53	PM52	PM51	PM50

(FFH when RESET is input)

PM5n	Specifies mode for PM5n pin (n=0-7)
0	Output mode (output buffer ON)
1	Input mode (output buffer OFF)

Table 5-7 Port 5 Operation Modes

MODE pin	MM register bit			Operation mode
	MM2	MM1	MM0	
0	0	0	x	I/O mode
0	1	1	1	Address bus (A8-A15)
1*	x	x	x	

***:** The μPD78P238 MODE pin cannot be set to 1.

Note that Port 5 for μPD78233 functions only as an address bus (A8 through A15).

5.7.3 Operation state

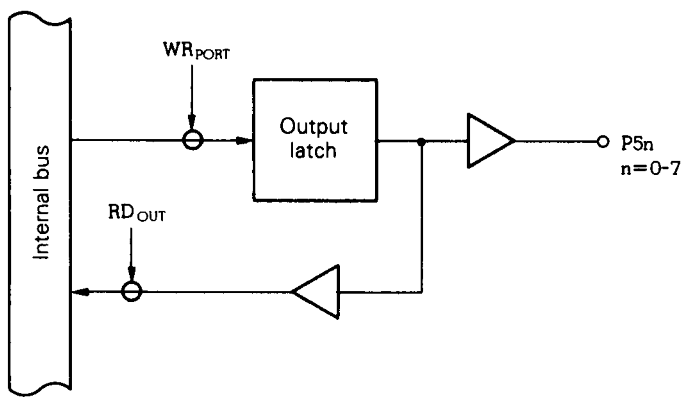
Port 5 is an I/O port multiplexed with an address bus (A8 through A15).

(1) In output mode

The Port 5 output latch becomes valid and data can be transferred between the output latch and accumulator by a transfer instruction. The output latch contents can be set freely by a logical operation instruction. Once data has been written to the output latch, it is retained until the next data is written to the output latch*.

***:** This also applies, when the other bits of the same port are manipulated by a bit manipulation instruction.

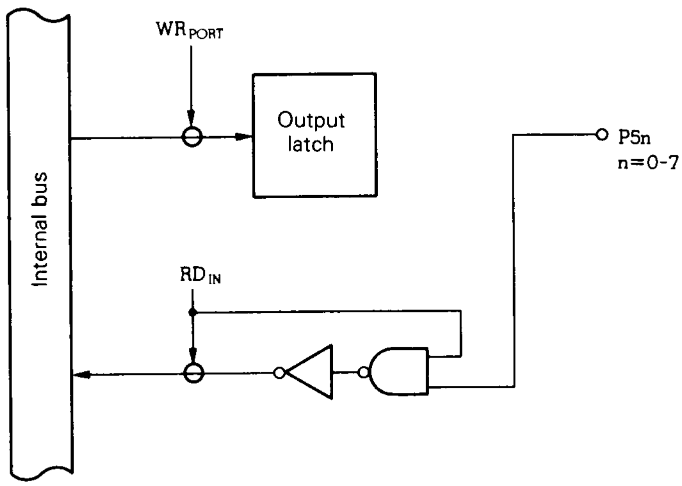
Fig. 5-38 Port Specified in Output Mode



(2) In input mode

The level for the port pins can be loaded to the accumulator by a transfer instruction. In this case, data can be written to the output latch, and data transferred from the accumulator by a transfer instruction, etc. is stored in all the output latches, regardless of whether the port is in the input or output mode. However, the output buffer for the bit specified in the input mode is in the high-impedance state. Therefore, the output buffer contents are not output to the port pin (the output latch contents for the bit currently specified in the input mode are output to the port pin, when the bit mode is changed to the output mode). The output latch contents for the bit specified in the input mode cannot be loaded to the accumulator.

Fig. 5-39 Port Specified in Input Mode



Caution: A bit manipulation instruction, although its ultimate purpose is to manipulate only 1 bit, accesses a port in 8 bit units. If a bit manipulation instruction is executed to manipulate a port, which can function as either an input or an output port, the contents of the output latch for the port pins, specified in the input mode, become undefined (except the bit that has been manipulated by the SET1 or CLR1 instruction, etc.). If a port to be manipulated has a bit, whose mode is to be switched between input and output, this must be taken into consideration. The same applies, when manipulating the port with another 8-bit arithmetic operation instruction.

(3) In address bus (A8-A15) mode

Port 5 is automatically used as an address bus, when an external memory is accessed. At this time, do not execute an I/O instruction that manipulates Port 5.

5.7.4 Internal pull-up resistor

Port 5 can be connected to internal pull-up resistors. When these internal resistors are used, the number of external components and mounting area to be occupied by the external components can be reduced.

Whether the pull-up resistors are used can be specified in bit units by the PUO5 bit for the pull-up resistor option register (PUO) and Port 5 mode register (PM5).

The pull-up resistor for a pin (PM5n = 1, n = 0-7), set in the input mode by the port 5 mode register (PM5), becomes valid when the PUO5 bit is 1.

Fig. 5-40 Pull-up Resistor Option Register Format

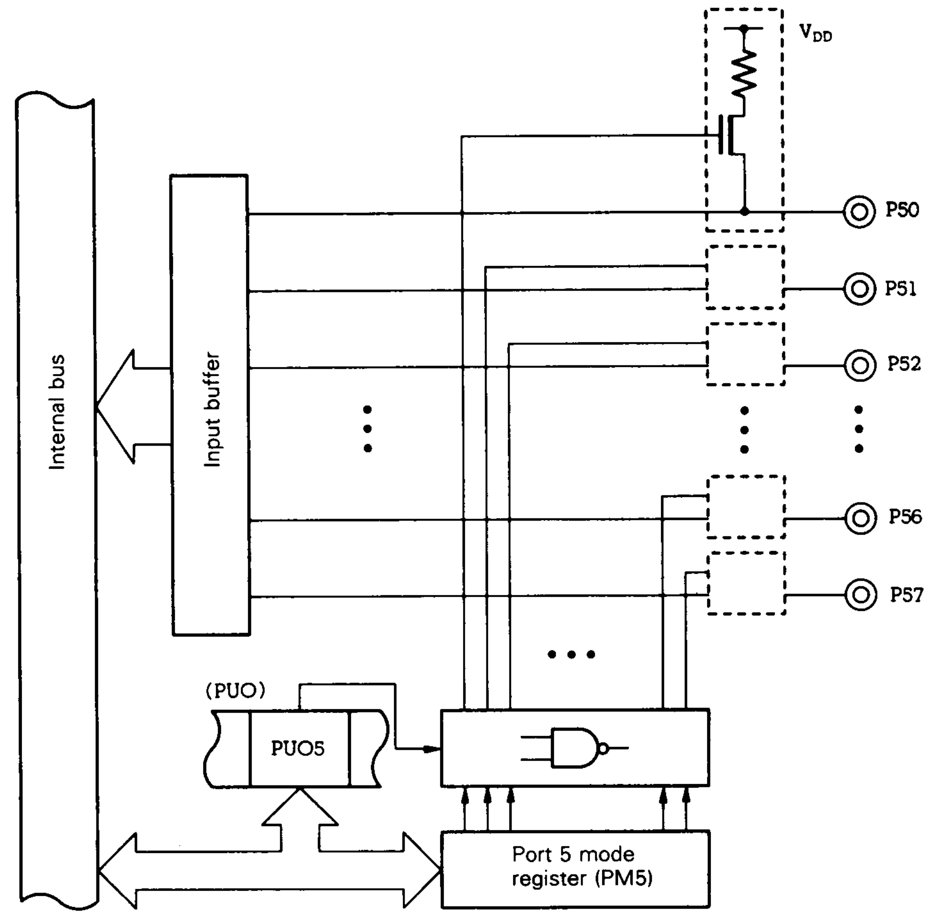
	7	6	5	4	3	2	1	0	
PUO	0	PUO6	PUO5	PUO4	PUO3	PUO2	PUO1	0	(00H when RESET is input)

PUO5	Specifies pull-up resistor connection of port 5
0	Not connected
1	Connected

Caution: For the μPD78233, port 5 is used as the address bus, "0" must always be set to PUO5, to prevent connecting the internal pull-up resistor. Also clear PUO5 for μPD78234, when port 5 for μPD78234 is used as an address/data bus.

Remarks: Set the PUO register to 00H, to reduce power dissipation, before the STOP mode is set.

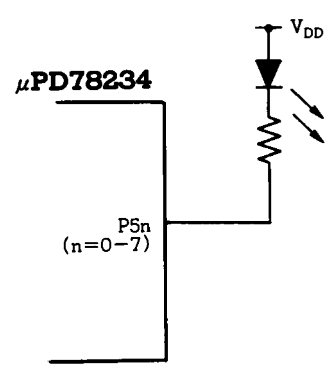
Fig. 5-41 Pull-up Resistor Connection (Port 5)



5.7.5 Direct drive for LED

The driving capability for the port output buffer low level side is reinforced so that Port 5 can directly drive an LED with an active-low signal. Figure 5-42 shows an example of connecting an LED to the port.

Fig. 5-42 Direct Drive for LED



5.8 Port 6

Port 6 is an 8-bit I/O port with an output latch. (P60 through P63 are used as output port pins.) P64 through P67 are connected to software-programmable pull-up resistors.

In addition to functioning as a port pin, each pin in port 6 is also used to input or output a control signal, as shown in Table 5-8. Whether a pin functions as a port pin or control pin is specified by performing the operations shown in the table.

P64 and P65 for μPD78233 function as $\overline{\text{RD}}$ and $\overline{\text{WR}}$ only.

When the $\overline{\text{RESET}}$ signal has been input, P60 through P63 enter the output high-impedance state, and go low when the $\overline{\text{RESET}}$ signal has been removed. P64 through P67 are set in the input mode (output high-impedance state), when the $\overline{\text{RESET}}$ signal has been input. The higher 4 bits for the output latch become undefined and the lower 4 bits are cleared to 0H.

Table 5-8 Port 6 Operation Modes

Pin	Port mode	Control signal I/O mode	Operations to set control signal mode
P60-P63	Output port	A16-A19 output	Set MM6 for MM register to 1
P64	I/O port	$\overline{\text{RD}}$ output	In the case of μPD78233, or when external memory expansion mode is specified by MM2-0 bits for MM register
P65		$\overline{\text{WR}}$ output	
P66		$\overline{\text{WAIT}}$ input	Specified by PW register or PWn1 and PWn0 bits (n = 2,3) for MM register or by setting P66 in input mode
P67		$\overline{\text{REFRQ}}$ output	Set RFEN bit for RFM register to 1

Caution: P60 through P63 enter the output high-impedance state, while the $\overline{\text{RESET}}$ signal is input, but go low after the $\overline{\text{RESET}}$ signal has been removed. Therefore, design an external circuit that allows the low level to be output as the initial status.

Remarks: In detail, refer to Chapter 15 Local Bus Interface Function.

5.8.1 Hardware configuration

Figures 5-43 through 5-46 show the hardware configurations for Port 6.

Fig. 5-43 P60-63 Configuration (Port 6)

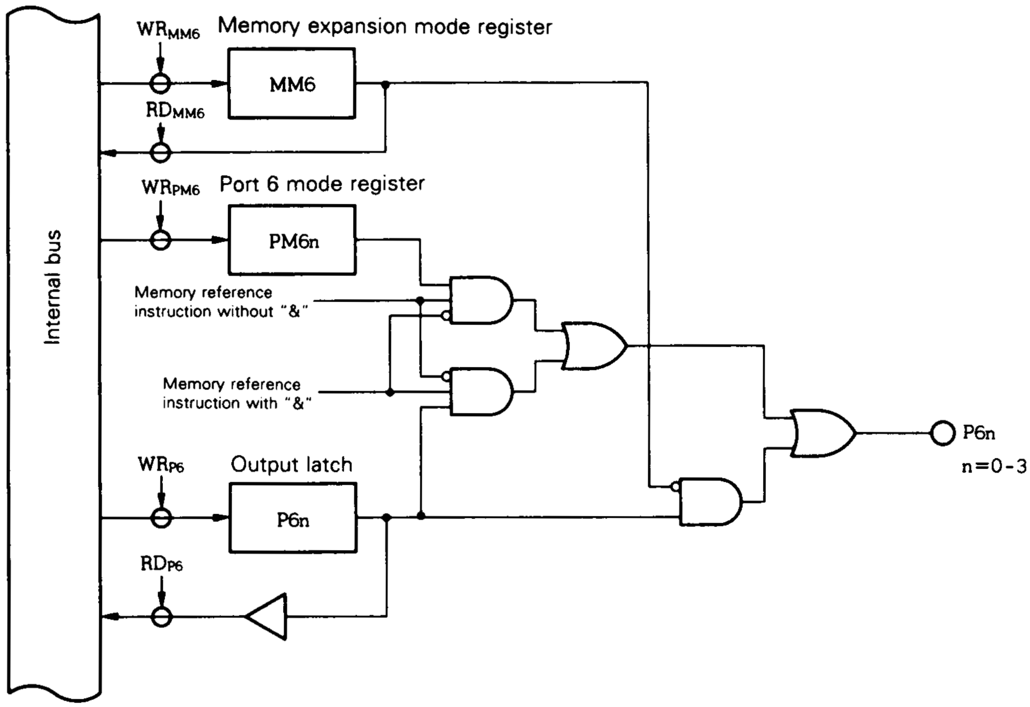


Fig. 5-44 P64 and P65 Configurations (Port 6)

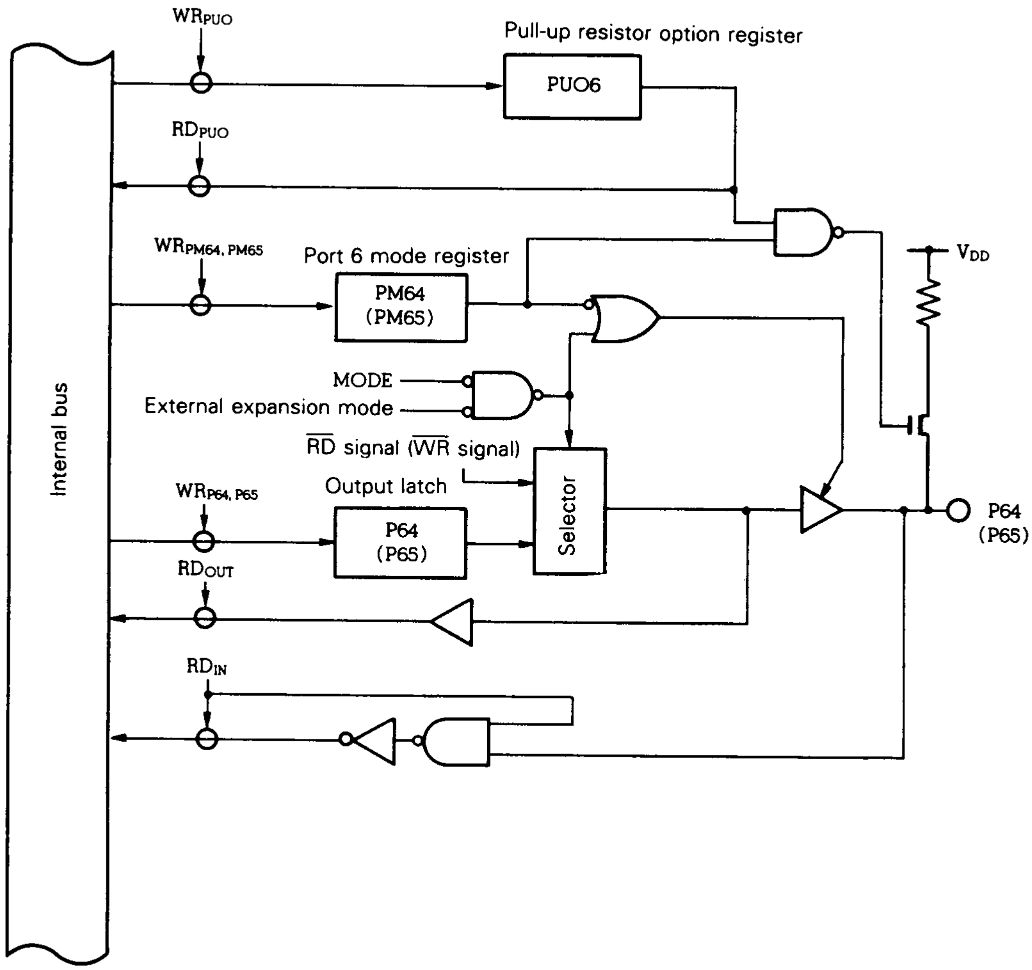


Fig. 5-45 P66 Configuration (Port 6)

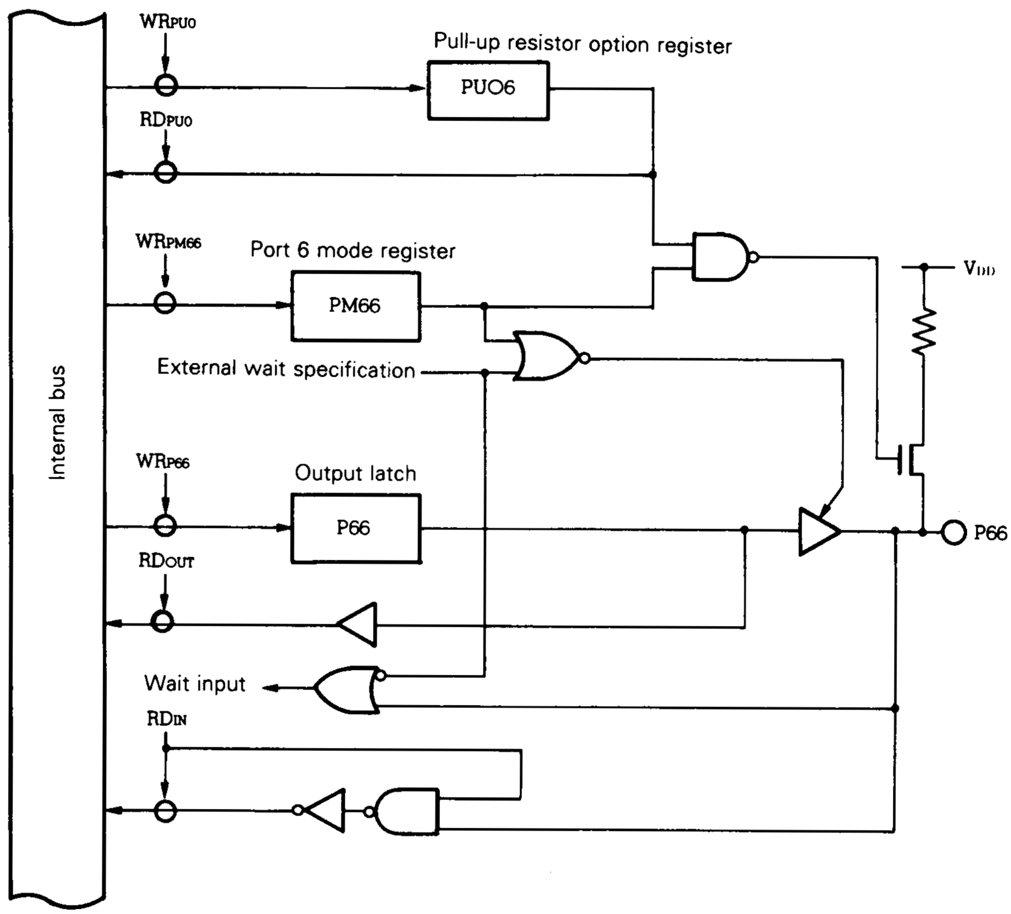
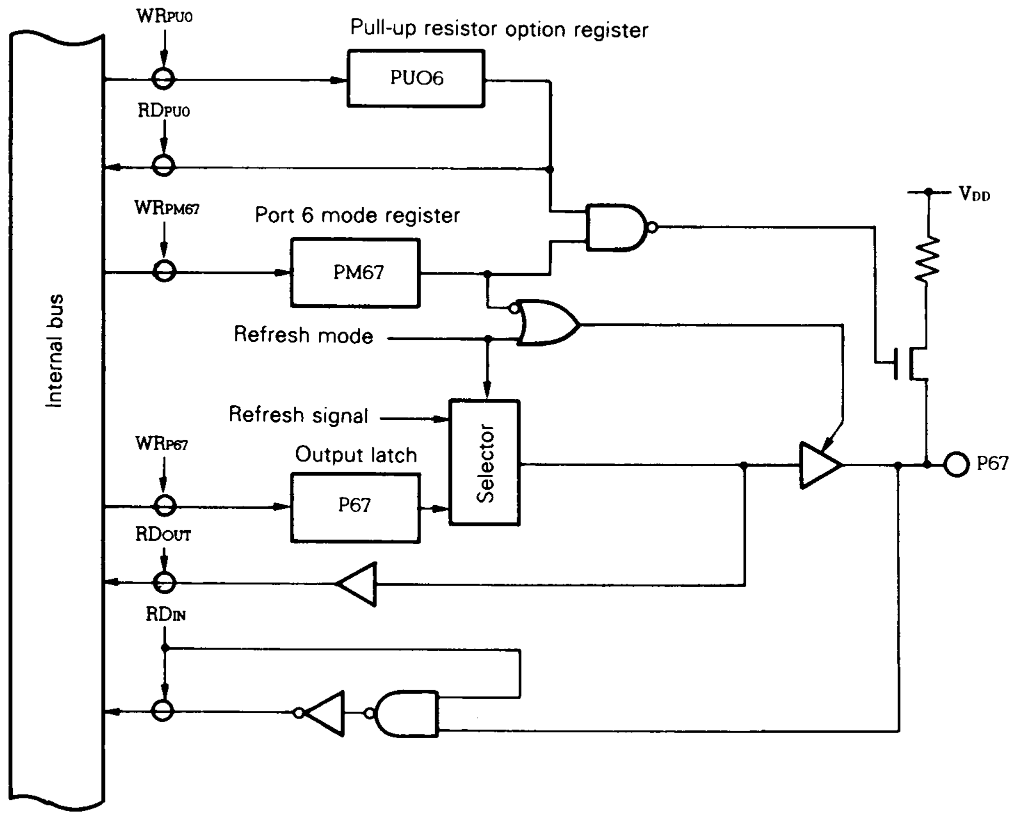


Fig. 5-46 P67 Configuration (Port 6)



5.8.2 Setting I/O mode and control mode

Port 6 can be set in input or output mode by port 6 mode register (PM6) in bit units, as shown in Fig. 5-47.
Table 5-9 shows operations to be performed to set Port 6 in the control signal mode.
Note that pins P64 and P65 for μPD78233 function as the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ signal pins only.

Table 5-9 Port 6 Control Signal Functions

Pin	Function	I/O	To use control signal function
P60	A16	Output	Set MM6 bit for MM register to 1
P61	A17	Output	
P62	A18	Output	
P63	A19	Output	
P64	$\overline{\text{RD}}$	Output	Specify external memory expansion mode by using MM2-MM0 bits for MM register. These μPD78233 pins function as RD and WR pins only
P65	$\overline{\text{WR}}$	Output	
P66	$\overline{\text{WAIT}}$	Input	Specify external wait input by using PWn1 and PWn0 (n = 2 or 3) for PW or MM register or by setting P66 in input mode.
P67	$\overline{\text{REFRQ}}$	Output	Set RFEN bit for RFM register to 1

Cautions: 1. To use pins P60 through P63 as output port pins, be sure to clear the PM60 through PM63 bits to "0"; otherwise, correct emulation cannot be performed by an in-circuit emulator.
2. Set P66/$\overline{\text{WAIT}}$ pin in input mode by PM6 register when using this pin as $\overline{\text{WAIT}}$ pin.

(a) Port mode

Of ports not specified for control mode, P60-P63 becomes output only port, and P64-67 can be specified for input/output in bit units by port mode 6 register (PM6).

(b) Control signal input/output mode

(i) A16-A19 (Address Bus)

The upper address bus output pins when the external memory space is extended (10000H-FFFFFH). These pins are controlled by the memory extension register (MM).

(ii) $\overline{\text{RD}}$ (Read Strobe)

The strobe signal output pin which outputs strobe signal for external memory read operation when the external memory space is extended or in case of the μPD78233.

(iii) $\overline{\text{WR}}$ (Write Strobe)

The strobe signal output pin which outputs strobe signal for external memory write operation when specified by the MM register or in case of μPD78233.

(iv) $\overline{\text{WAIT}}$ (Wait)

A wait signal input pin. Operation of this pin is controlled by the programmable wait control (PW) register or the MM register.

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

This pin outputs refresh pulse to the pseudo-static memory when the pseudo-static memory is externally connected. Operation of this register is controlled by the refresh mode register (RFM).

Fig. 5-47 Port 6 Mode Register Format

	7	6	5	4	3	2	1	0
PM6	PM67	PM66	PM65	PM64	PM63	PM62	PM61	PM60

(F×H when RESET is input)

PM63	PM62	PM61	PM60	Memory bank specification*
0	0	0	0	Bank 0
0	0	0	1	Bank 1
0	0	1	0	Bank 2
0	0	1	1	Bank 3
0	1	0	0	Bank 4
0	1	0	1	Bank 5
0	1	1	0	Bank 6
0	1	1	1	Bank 7
1	0	0	0	Bank 8
1	0	0	1	Bank 9
1	0	1	0	Bank A
1	0	1	1	Bank B
1	1	0	0	Bank C
1	1	0	1	Bank D
1	1	1	0	Bank E
1	1	1	1	Bank F

PM6n	Specifies I/O mode of PM6n pin (n=4-7)
0	Output mode (output buffer ON)
1	Input mode (output buffer OFF)

***:** When the MM6 bit for the memory expansion mode register (MM) is set to 1, the PM6 register specifies a bank, when a memory reference instruction without "&" is executed. When the 1M-byte expansion function is not used, clear the PM63 to PM60 bits to 0.

Remarks: The lower 4 bits (P60 through P63) for Port 6 constitute an output port.

5.8.3 Operation state

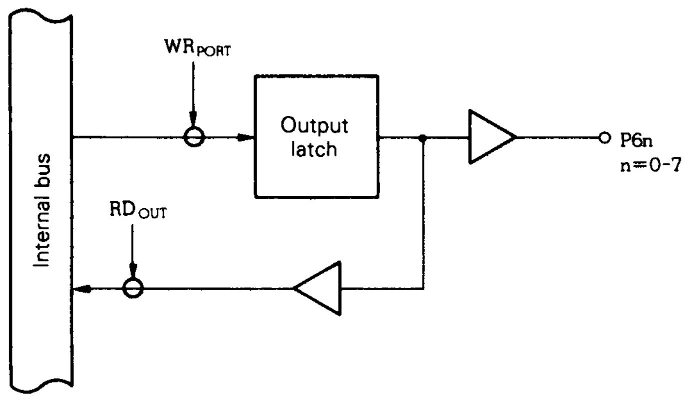
Port 6 is an I/O port multiplexed with control pins.

(1) In output mode

The output latch for Port 3 becomes valid and data can be transferred between the output latch and accumulator by a transfer instruction. The output latch contents can be set freely by a logical operation instruction. Once data has been written to the output latch, it is retained until the next data is written to the output latch*.

***:** This also applies, when the other bits of the same port are manipulated by a bit manipulation instruction.

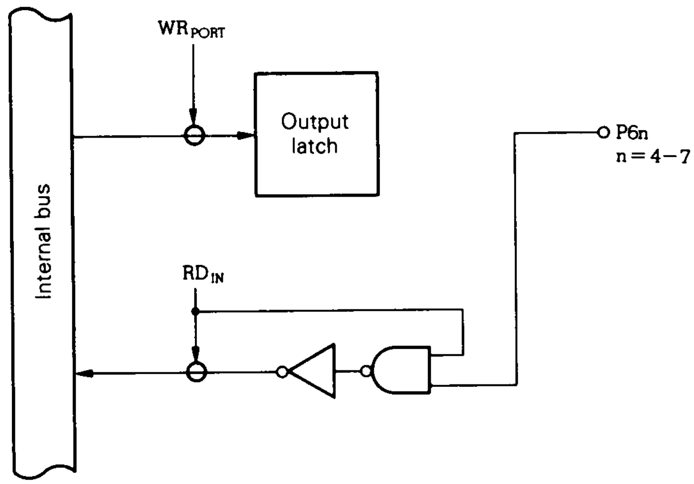
Fig. 5-48 Port Specified in Output Mode



(2) In input mode

The level for the port pins can be loaded to the accumulator by a transfer instruction. In this case, data can be written to the output latch, and data transferred from the accumulator by a transfer instruction is stored in all the output latches, regardless of whether the port is in the input or output mode. However, the output buffer for the bit specified in the input mode is in the high-impedance state. Therefore, the output buffer contents are not output to the port pin (the output latch contents for the bit currently specified in the input mode are output to the port pin, when the bit mode is changed to the output mode). The output latch contents for the bit specified in the input mode cannot be loaded to the accumulator.

Fig. 5-49 Port Specified in Input Mode



Caution: A bit manipulation instruction, although its ultimate purpose is to manipulate only 1 bit, accesses a port in 8 bit units. If a bit manipulation instruction is executed to manipulate a port, which can function as either an input or an output port, the contents of the output latch for the port pins, specified in the input mode or control mode, become undefined (except the bit that has been manipulated by the SET1 or CLR1 instruction etc.). If a port to be manipulated has a bit, whose mode is to be switched between input and output, this must be taken into consideration. The same applies, when manipulating the port with another 8-bit arithmetic operation instruction.

(3) In control mode

The Port 6 pins cannot be manipulated or tested by software, when the port is in the control signal mode.

5.8.4 Internal pull-up resistor

Pins P64 through P67 for Port 6 can be connected to internal pull-up resistors. When these internal resistors are used, the number of external components and mounting area to be occupied by the external components can be reduced.

Whether the pull-up resistors are used can be specified in bit units by the PUO6 bit for the pull-up resistor option register (PUO) and port 6 mode register (PM6). When the PUO6 bit is 1, the internal pull-up resistor for the pin set in the input mode by the PM6 register (PM6n = 1, n = 4-7) becomes valid.

Pins P60 through P63 cannot be connected to a pull-up resistor.

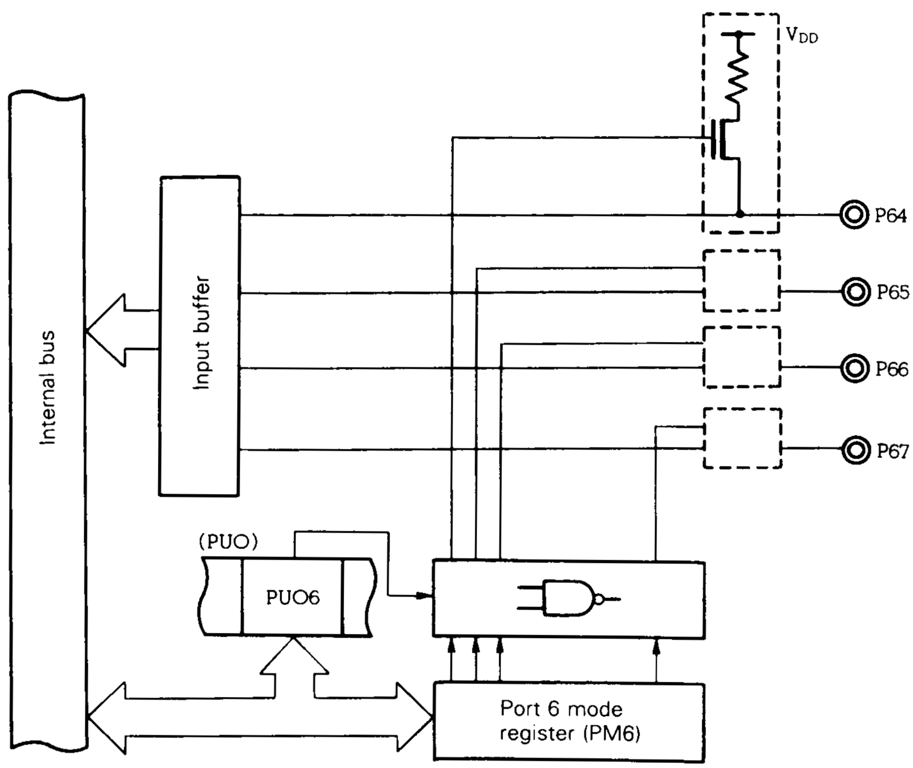
Fig. 5-50 Pull-up Resistor Option Register Format

	7	6	5	4	3	2	1	0	
PUO	0	PUO6	PUO5	PUO4	PUO3	PUO2	PUO1	0	(00H when RESET is input)

PUO6	Specifies pull-up resistor connection of port 6
0	Not connected
1	Connected

Remarks: Set the PUO register to 00H, to reduce the power dissipation, before the STOP mode is set.

Fig. 5-51 Pull-up Resistor Connection (Port 6)



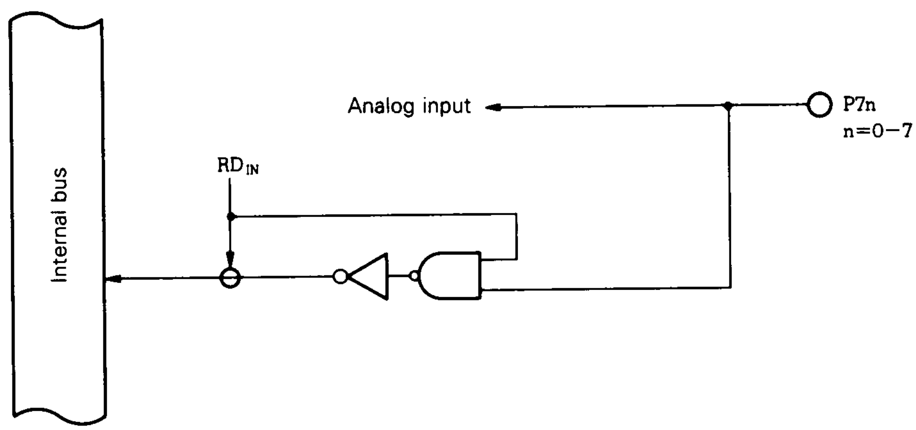
5.9 Port 7

Port 7 is an 8-bit input port. The pins for this port can also function as analog input pins for the A/D converter. The level for each pin can be read or tested, regardless of the shared pin function.

5.9.1 Hardware configuration

Figure 5-52 shows the Port 7 hardware configuration.

Fig. 5-52 Port 7 Configuration



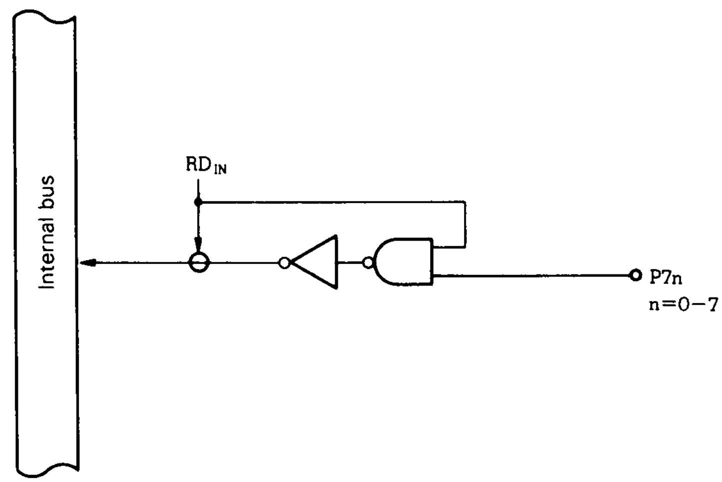
5.9.2 Setting I/O mode and control mode

Port 7 is an input port, through which analog signals can always be input. A mode does not have to be set for this port. Whether the port is used for A/D conversion operation is specified by ADM for the A/D converter (for details, refer to **CHAPTER 9 A/D CONVERTER**).

5.9.3 Operation state

Port 7 is an input port. Its pin levels can always be read or tested.

Fig. 5-53 Port Specified in Input Mode



5.9.4 Internal pull-up resistor

Port 7 is not provided with a pull-up resistor.

5.9.5 Note

When pins P70 through P77 for Port 7 are used as ANI0 through ANI7 pins, do not apply a voltage exceeding or falling below the AV_{SS} to AV_{REF} voltage range to these pins.

For details, refer to **9.5 Notes** in **CHAPTER 9 A/D CONVERTER**.

5.10 Note

(1) All the port pins enter the high-impedance state when the $\overline{RESET}$ signal is input (the internal pull-up resistors are disconnected from the pins).
To prevent the pins from entering the high-impedance state when the $\overline{RESET}$ signal is input, use an appropriate external circuit.

(2) Operations for the pull-up resistor option register (PUO), which connects the internal pull-up resistors, cannot be emulated completely by an in-circuit emulator, due to the following restrictions:

- The correct PUO register value cannot be read, even when the register is read.
- Bit manipulation instructions and arithmetic operation instructions cannot be correctly executed to the PUO register (on some occasions, SFR illegal access break occurs, aborting emulation).
- No pull-up resistor is provided.

Therefore, observe the following two points:

- To the PUO register, only write operation by an 8-bit data transfer instruction (MOV) can be performed.
- Connect pull-up resistors to the target board during debugging.

(3) The output latch contents are not initialized, even by the $\overline{RESET}$ input. To use a port as an output port, be sure to initialize the output latch before turning on the output buffer; otherwise, unexpected data will be output to the output port.
Similarly, to use a port pin as a control signal pin, be sure to initialize the internal hardware and set the pin in the control signal mode.

(4) P22 through P26 are not pulled up immediately after the RESET signal has been input, and the interrupt request flag may be set because of the function of the shared pins (INTP1 through INTP5). Therefore, specify pull-up and clear the interrupt request flag by the initialization routine.

(5) With an in-circuit emulator, the pin levels of port 2 before noise is rejected are read or tested.

(6) To use P60 through P63 in the output port mode, be sure to clear PM60 through PM63 bits to 0; otherwise, emulation cannot be correctly performed by an in-circuit emulator.

(7) Do not apply a voltage outside the AV_{SS} to AV_{REF} range to pins P70 through P77, when they are used as ANI0 through ANI7 pins.
For details, refer to **9.5 Notes** in **CHAPTER 9 A/D CONVERTER**.

(8) For the μPD78234, when using P40-P47 and P50-57 as address/data bus and address bus, respectively, the PUO4 and PUO5 bits of the PUO register must be set to "0", to prevent connecting the internal pull-up resistors.
Also, clear the PUO4 and PUO5 bits for the PUO register for μPD78233 to 0 to prevent connecting the internal pull-up resistors, because P40-P47 and P50-P57 for μPD78233 are used as an address, data bus and address bus.

(9) P60 through P63 enter the output high-impedance state, while the $\overline{\text{RESET}}$ signal is input, but go low after the $\overline{\text{RESET}}$ signal has been removed. Therefore, design an external circuit that allows the low level to be output as the initial status.

(10) A bit manipulation instruction, although its ultimate purpose is to manipulate only 1 bit, accesses a port in 8 bit units. If a bit manipulation instruction is executed to manipulate a port, which can function as either an input or an output port, the contents of the output latch for the port pins, specified in the input mode or control mode, become undefined (except the bit that has been manipulated by the SET1 or CLR1 instruction, etc.). If a port to be manipulated has a bit, whose mode is to be switched between input and output, this must be taken into consideration. The same applies, when manipulating the port with another 8-bit arithmetic operation instruction.

(11) Set the P66/$\overline{\text{WAIT}}$ pin in the input mode by the PM6 register when using the pin as the $\overline{\text{WAIT}}$ pin.

CHAPTER 6 REAL-TIME OUTPUT FUNCTION

6.1 Configuration and Function

The real-time output function is implemented by the hardware shown in Fig. 6-1, including port 0 and buffer registers (P0H and P0L).

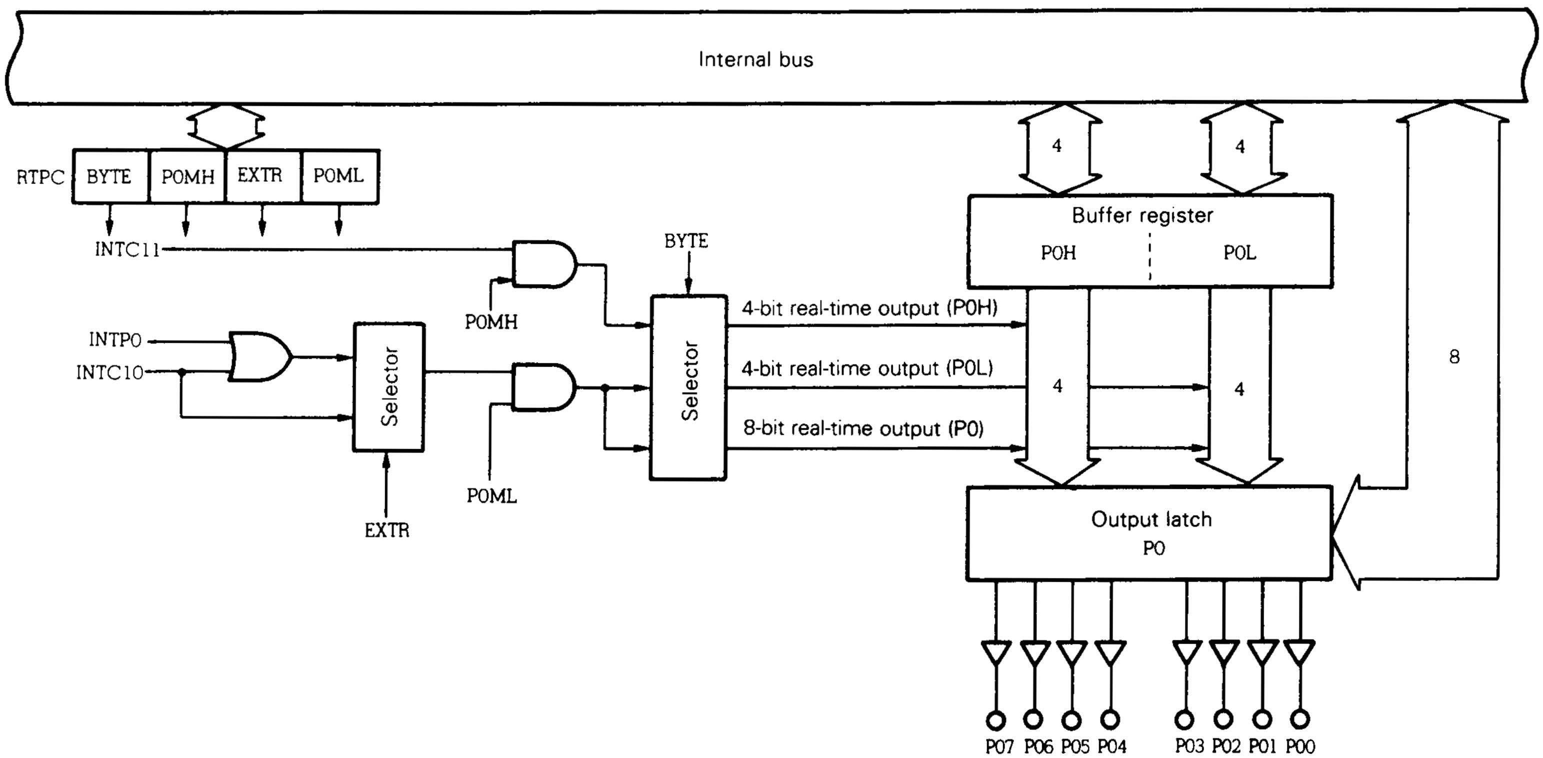
This function is to transfer data, stored in the buffer registers in advance, to the output latches of the port as soon as a timer interrupt or external interrupt occurs, so that the data can be output to an external device. The pin that outputs the data is called a real-time output port.

The following two types of data can be output as "real-time output data":

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

This real-time output function can also be used in combination with the macro service function which is described later to implement, without using software, a pattern generator function whose timing is programmable. This function is suitable for controlling such systems as stepping motors.

Fig. 6-1 Configuration of Real-Time Output Port



6.2 Real-Time Output Port Control Register (RTPC)

This 8-bit register specifies the port 0 functions.

Data can be read from or written to this register by an 8-bit manipulation instruction or bit manipulation instruction. Figure 6-2 shows the format for this register.

When the $\overline{\text{RESET}}$ signal is input, the RTPC register contents are initialized to 00H.

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

	7	6	5	④	3	2	1	⓪
RTPC	BYTE	0	0	POMH	EXTR	0	0	POML

P0ML	Specifies P00-P03 functions
0	Port mode
1	Real-time output port mode

EXTR	Data transfer by INTP0 from buffer register to output latch
0	Disabled
1	Enabled. BYTE=0: Only P0L data is transferred BYTE=1: Both P0L and P0H data are transferred

P0MH	Specifies P04-P07 functions
0	Port mode
1	Real-time output port mode

BYTE	Operation mode for real-time output port
0	4-bit separate real-time output port
1	8-bit real-time output port

Caution: When P0ML and P0MH are set to 1, the output buffer for the corresponding port is turned ON and the output latch contents for port 0 are output, regardless of the port 0 mode register contents. Therefore, initialize the contents of the output latch before using the real-output port.

6.3 Accessing Real-Time Output Port

Buffer registers P0H and P0L are mapped at independent addresses in the SFR area, as shown in Fig. 6-3.

When the 4 bit x 2 channel real-time output function is specified, data can be set in each buffer register independently.

When the 8 bit x 1 channel real-time output function is specified, however, the same data is set in both P0H and P0L, when 8-bit data is written to either of the registers.

Table 6-1 shows the operations to be performed to manipulate port 0 and the buffer registers for port 0.

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

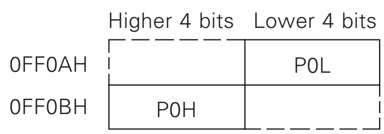


Table 6-1 Operations to Manipulate Port 0 and Buffer Registers

Operation mode	Register	Read operation		Write operation	
		Higher 4 bits	Lower 4 bits	Higher 4 bits	Lower 4 bits
8-bit port mode	P0	Output latch		Output latch	
	P0L	Buffer register*		—	Buffer register
	P0H	Buffer register*		Buffer register	—
8-bit real-time output port mode	P0	Output latch		—	
	P0L	Buffer register		Buffer register	
	P0H	Buffer register		Buffer register	
4-bit separate real-time output port mode	P0	Output latch		—	
	P0L	Buffer register*		—	Buffer register
	P0H	Buffer register*		Buffer register	—
P00-P03: port P04-P07: real-time output port mode	P0	Output latch		—	Output latch
	P0L	Buffer register*		—	Buffer register
	P0H	Buffer register*		Buffer register	—
P00-P03: real-time output port mode P04-P07: port	P0	Output latch		Output latch	—
	P0L	Buffer register*		—	Buffer register
	P0H	Buffer register*		Buffer register	—

***:** The contents of P0H are read to the higher 4 bits, and the contents of P0L are read to the lower 4 bits.

—: The output latch and buffer registers are not affected.

Example showing setting data in buffer registers

- 4 bit x 2 channel data

```
MOV P0L,#05H    ; Sets 0101B in P0L register
MOV P0H,#0C0H  ; Sets 1100B in P0H register
```

- 8 bit x 1 channel data

```
MOV P0L, #0C5H ; Sets 0101B in P0L register and 1100B in P0H register
```

Or,

```
MOV P0H, #0C5H
```

The timing at which the data is output to an output latch can be determined by the following three sources:

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

6.4 Operation

When port 0 is set in the real-time output port mode, the buffer register contents are fetched to the output latch in synchronization with occurrence of the trigger conditions listed in Table 6-2 and output from the port 0 pins.

For example, assume that a signal, indicating the coincidence of the contents of timer 1 (TM1) for the 8-bit timer/ counter 1 with those for compare registers (CR10 and CR11) is selected as the output trigger source. Then, the output data from the port 0 pins can be changed into the values for the buffer registers at time intervals set in advance in the compare registers. The output data from the port 0 output pins can thus be changed sequentially at arbitrary time intervals by using the real-time output port function, along with macro service function (for details of the macro service function, refer to **14.4 Macro Service Function**).

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

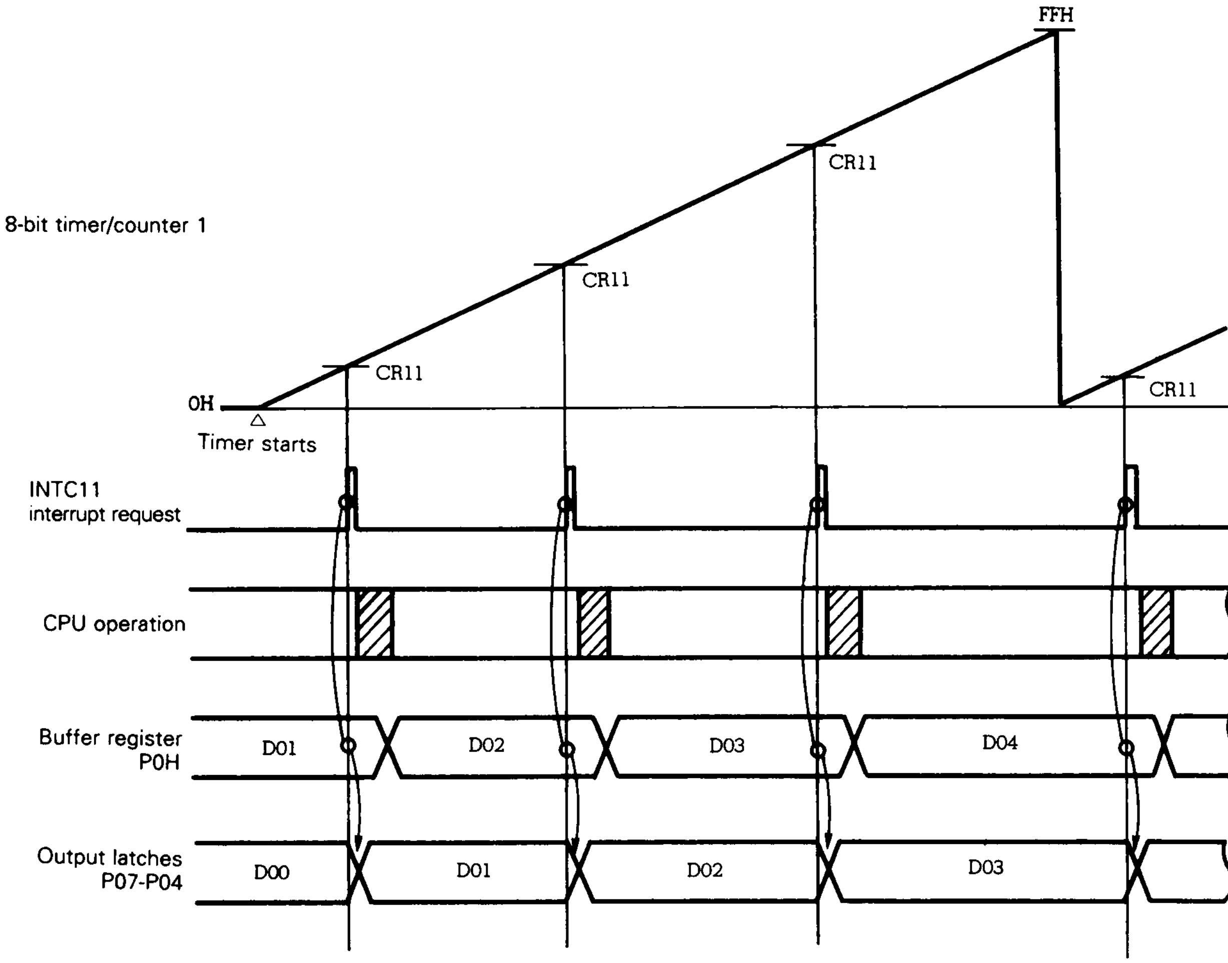
Table 6-2 Output Trigger for Real-Time Output Port (when P0MH = P0ML = 1)

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: An in-circuit emulator cannot reject the digital noise of the INTP0 pin normally. When data transfer from the buffer register to the output latch is enabled by using the signal input to the INTP0 pin, therefore, the operation is performed in accordance with the detected wrong edge. Keep this in mind when using the in-circuit emulator.

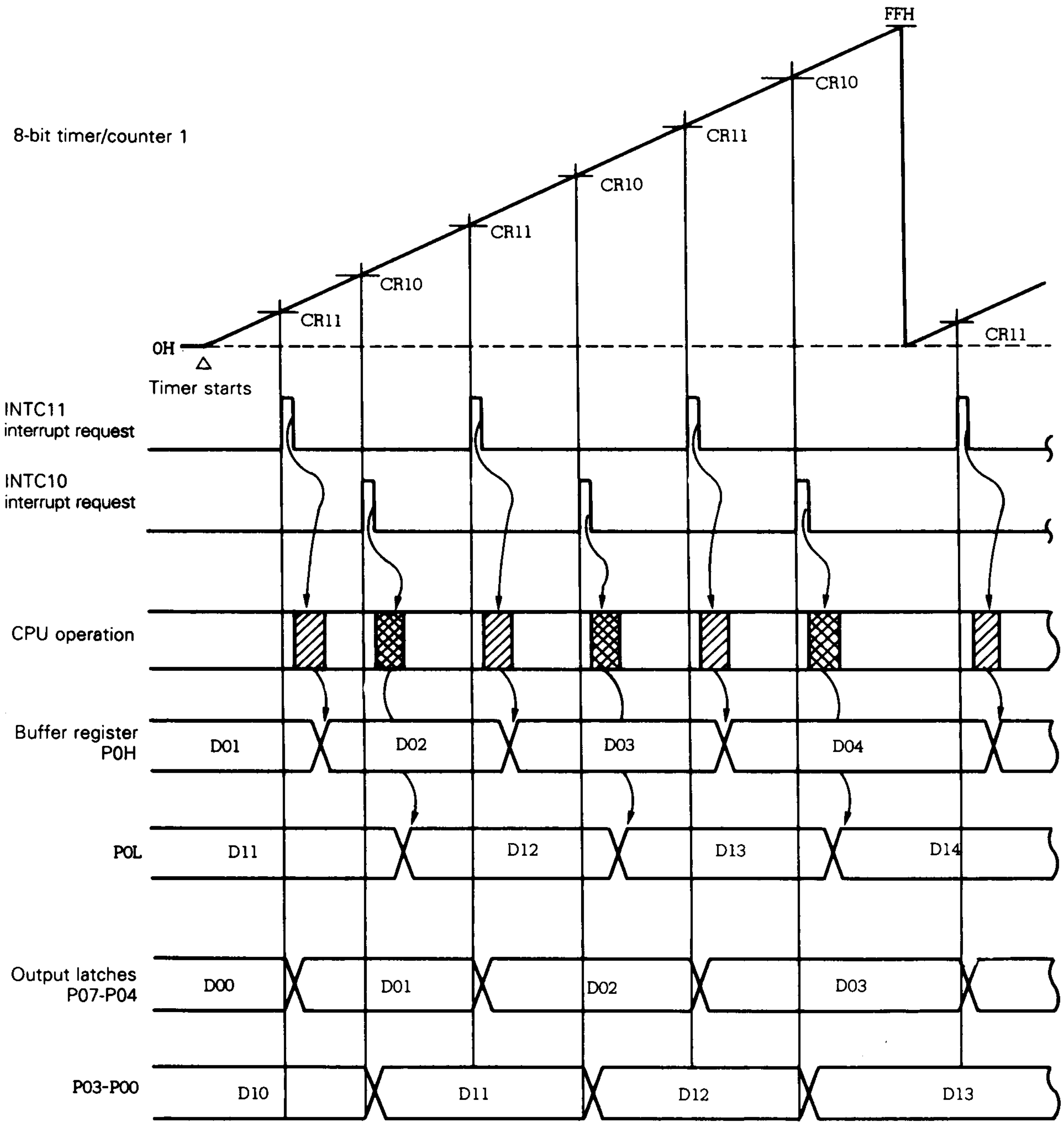
For the detail of detection of the wrong edge, refer to 13.4 Notes in CHAPTER 13 EDGE DETECTION FUNCTION.

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



The shaded portions in the above figure indicate that the buffer and compare register contents are changed by software processing or macro service (refer to **14.4 Macro Service Function**).

Fig. 6-5 Real-Time Output Port Operation Timing (2-ch independent control)



The shaded portions in the above figure indicate that the buffer and compare register contents are changed by software processing or macro service (refer to **14.4 Macro Service Function**).

6.5 Application Example

This section describes an application example for a 4-bit real-time output port, which consists of pins P00 through P03 for port 0.

The buffer register P0L contents are output to P00 through P03 each time the 8-bit timer/counter 1 (TM1) contents coincide with those for compare register CR10. At this time, an interrupt occurs and, in the interrupt routine started by the interrupt, the data to be output next and the next data output timing are set (see **Fig. 6-6**).

For details on how to use timer/counter 1, refer to **7.2 8-bit Timer/Counter 1**.

Fig. 6-7 shows the data to be set in the control register, while Fig. 6-8 illustrates the procedure to set the data in the register. Fig. 6-9 shows the interrupt routine processing.

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

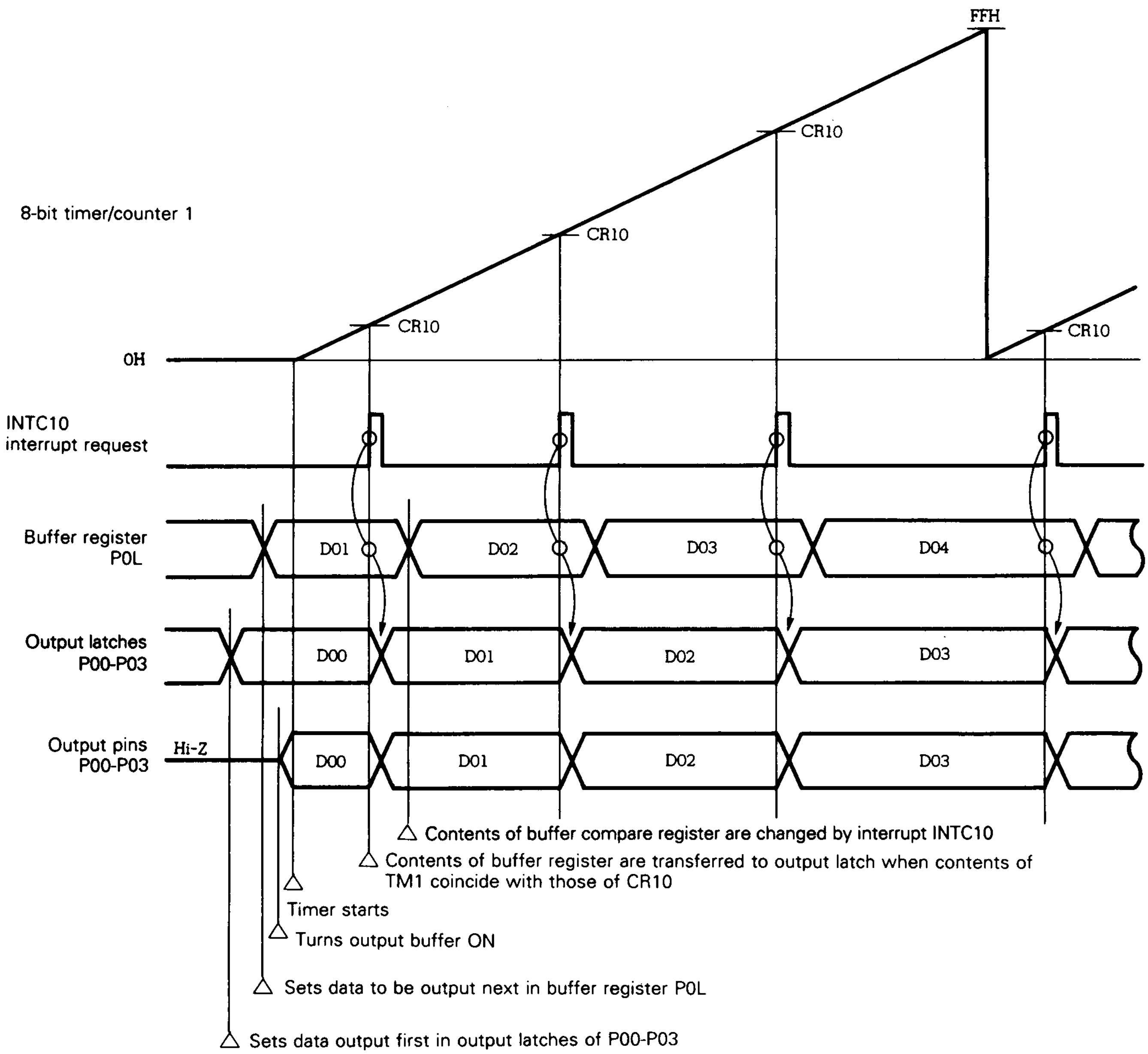


Fig. 6-7 Control Register Contents for Real-Time Output Function

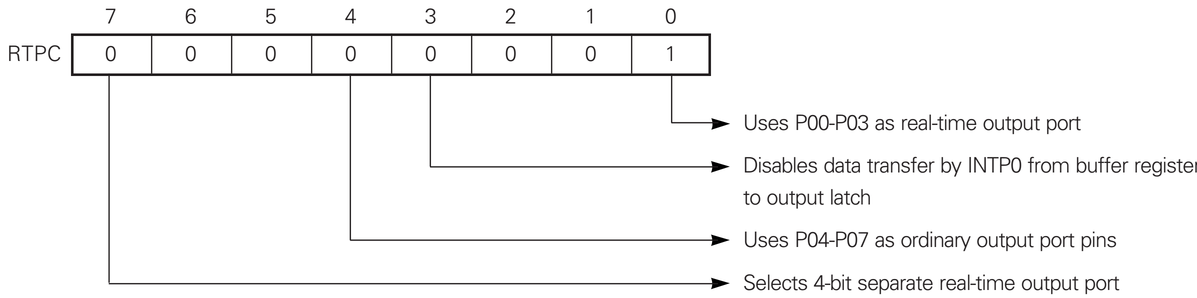


Fig. 6-8 Real-Time Output Function Setting Procedure

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

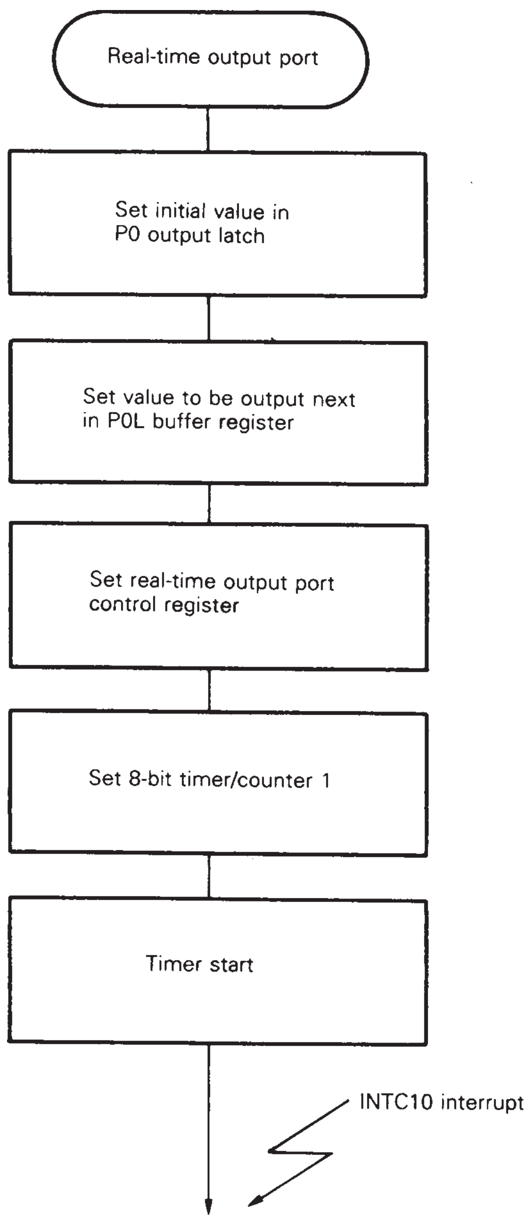
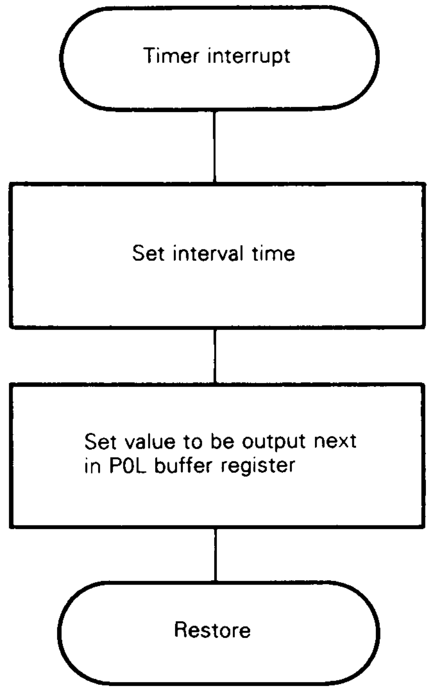


Fig. 6-9 Interrupt Request Processing When Real-Time Output Function Is Used



6.6 Note

(1) When the real-time output port function is specified, the output buffer for port 0 is automatically turned ON, and the P0 output latch contents are output, regardless of the PM0 register contents. Therefore, initialize the output latch contents, before specifying the real-time output port function.

(2) When port 0 is used as a real-time output port, values cannot be directly written to the output latches of the port by software. Therefore, the initial values of the output latches must be set by software before port 0 is specified to function as a real-time output port.
If it is necessary to forcibly change the output data into a constant value, while port 0 is used as a real-time output port, change the mode of the port into the ordinary output mode and then write the values to be output to the output latches by manipulating the RTPC register.

(3) Even when a mode in which data is transferred from the buffer registers to the output latches by using the signal from the INTP0 pin is selected, data is transferred from the buffer registers to the output latches, when the current value for timer/counter 1 (TM1) coincides with the compare register (CR10) contents.
To transfer data from the buffer registers to the output latches by using the signal from the INTP0 pin only, observe any one of the following points. However, the compare register (CR10) for the 8-bit timer/counter 1 cannot be used, no matter which method is used.

(a) Do not use 8-bit timer/counter 1.
(b) To use the capture/compare register (CR11) for the 8-bit timer/counter 1 (TM1) as a compare register, use TM1 as an interval time in a mode in which the CR11 contents are cleared when they coincide with the TM1 value. In this case, however, make sure that the compare register (CR10) contents are greater than the CR11 register contents (so that INTC10 does not occur).
(c) To use the capture/compare register (CR11) for the 8-bit timer/counter 1 as a capture register, use the capture register only when it is guaranteed that the valid edge period for signal input to the INTP0 pin is longer than the time required for the 8-bit timer 1 value to reach FEH from 0.
In this case, the value for compare register (CR10) must be FFH.
In addition, set timer/counter 1 so that its contents are cleared after captured.
(d) If there is no problem, even if data transfer from the buffer registers to the output latches are late by 1 clock of the 8-bit timer 1 (TM1) at maximum rate, and if it is sure that an overflow does not occur in TM1, set the following:

- Set the 8-bit timer/counter 1 so that its value is cleared after captured by the INTP0 signal.
- Do not use the INTP0 signal as the data transfer trigger signal for the real-time output port.
- Set “0H” in compare register (CR10).

(4) If POML and POMH are set to “1”, the output buffer of the respective port is turned ON, regardless of the port 0 mode register contents, and the port 0 output latch contents are output. Therefore, initialize the contents of the output latch before using the real-time output port.

(5) An in-circuit emulator cannot reject the digital noise of the INTP0 pin normally. When data transfer from the buffer register to the output latch is enabled by using the signal input to the INTP0 pin, therefore, the operation is performed in accordance with the detected wrong edge. Keep this in mind when using the in-circuit emulator.
For the detail of detection of the wrong edge, refer to **13.4 Notes** in **CHAPTER 13 EDGE DETECTION FUNCTION**.

CHAPTER 7 TIMER/COUNTER UNITS

μPD78234 is provided with one 16-bit timer/counter unit channel and three 8-bit timer/counter unit channels.

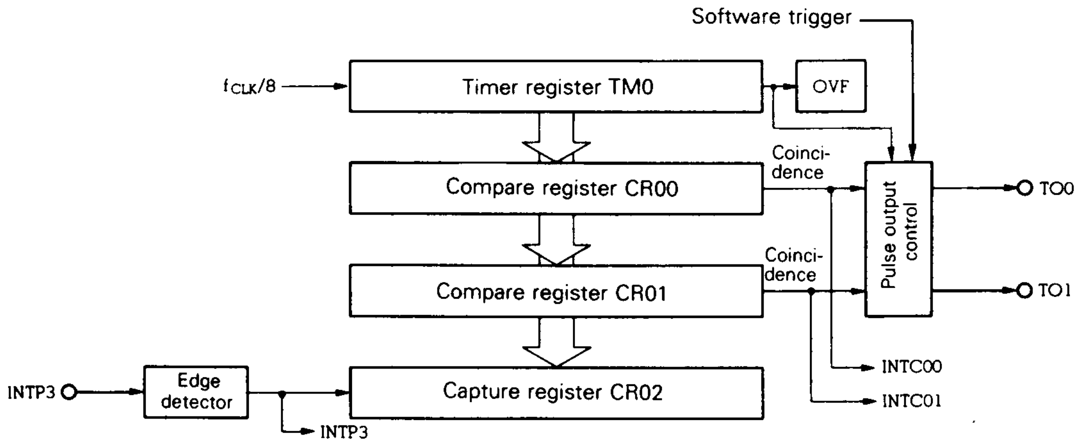
Table 7-1 Timer/Counter Types and Functions

Types and functions \ Unit		16-bit timer/counter	8-bit timer/counter 1	8-bit timer/counter 2	8-bit timer/counter 3
Types	Interval timer	2ch	2ch	2ch	1ch
	External event counter	—	—	○	—
	One-shot timer	—	—	○	—
Function	Timer output	2ch	—	2ch	—
	Toggle output	○	—	○	—
	PWM/PPG output	○	—	○	—
	One-shot pulse output	○	—	—	—
	Real-time output	—	○	—	—
	Pulse width measurement	○	○	○	—
	Interrupt request	2	2	2	1
	Clock source for serial interface	—	—	—	○

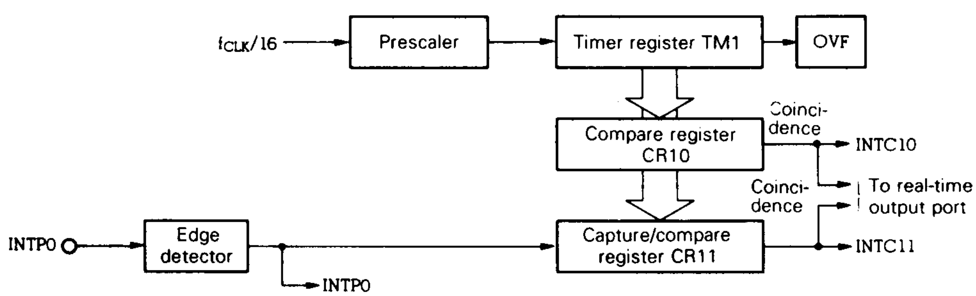
Since these timer/counter units support a total of seven interrupt requests, seven timer channels are available.

Fig. 7-1 Timer/Counter Units Configuration

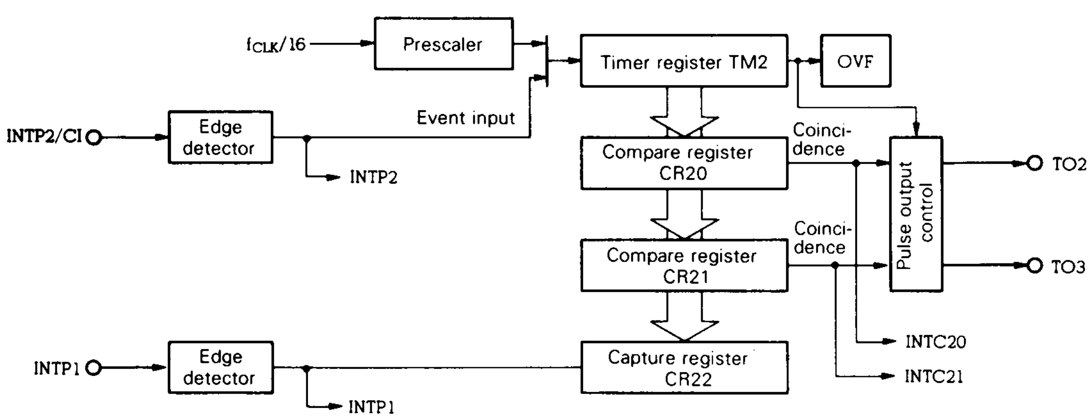
16-bit timer/counter unit



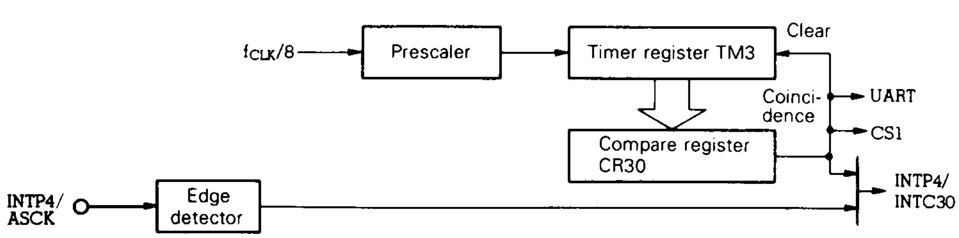
8-bit timer/counter unit 1



8-bit timer/counter unit 2



8-bit timer/counter unit 3



OVF: overflow flag

7.1 16-bit Timer/Counter

7.1.1 Function

The 16-bit timer/counter can be used as an interval timer, to output programmable square waves, and to measure pulse widths. In addition, the 16-bit timer/counter can also be used for the following purposes:

- PWM output
- Cycle measurement
- Software-triggered one-shot pulse output

(1) Interval timer

When the 16-bit timer/counter is used as an interval timer, an internal interrupt occurs at time intervals specified by the interval timer.

Table 7-2 16-bit Timer/Counter Time Interval

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

Figures in ( ) are at f_{CLK} = 6 MHz.

(2) Programmable square wave output

The 16-bit timer/counter can also be used to output programmable square waves through pins TO0 and TO1. These two square waves can be output independently from each other.

Table 7-3 Programmable Square Wave Output Range

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

Figures in ( ) are at f_{CLK} = 6 MHz.

(3) Pulse width measurement

The pulse width for a signal input to external interrupt pin INTP3 can be measured by the 16-bit timer/counter.

Table 7-4 Pulse Width Measurement Range

Measurable pulse width	Resolution
$12/f_{CLK}$ - $2^{16} \times 8/f_{CLK}$ (2.0 μs) (87.4 ms)	$8/f_{CLK}$ (1.3 μs)

Figures in ( ) are at f_{CLK} = 6 MHz.

(4) Software-triggered one-shot pulse output

This function is to make the output pulse level active by software or inactive by hardware (interrupt request signal). Pulses output by pins TO0 and TO1 can be separately controlled by this function.

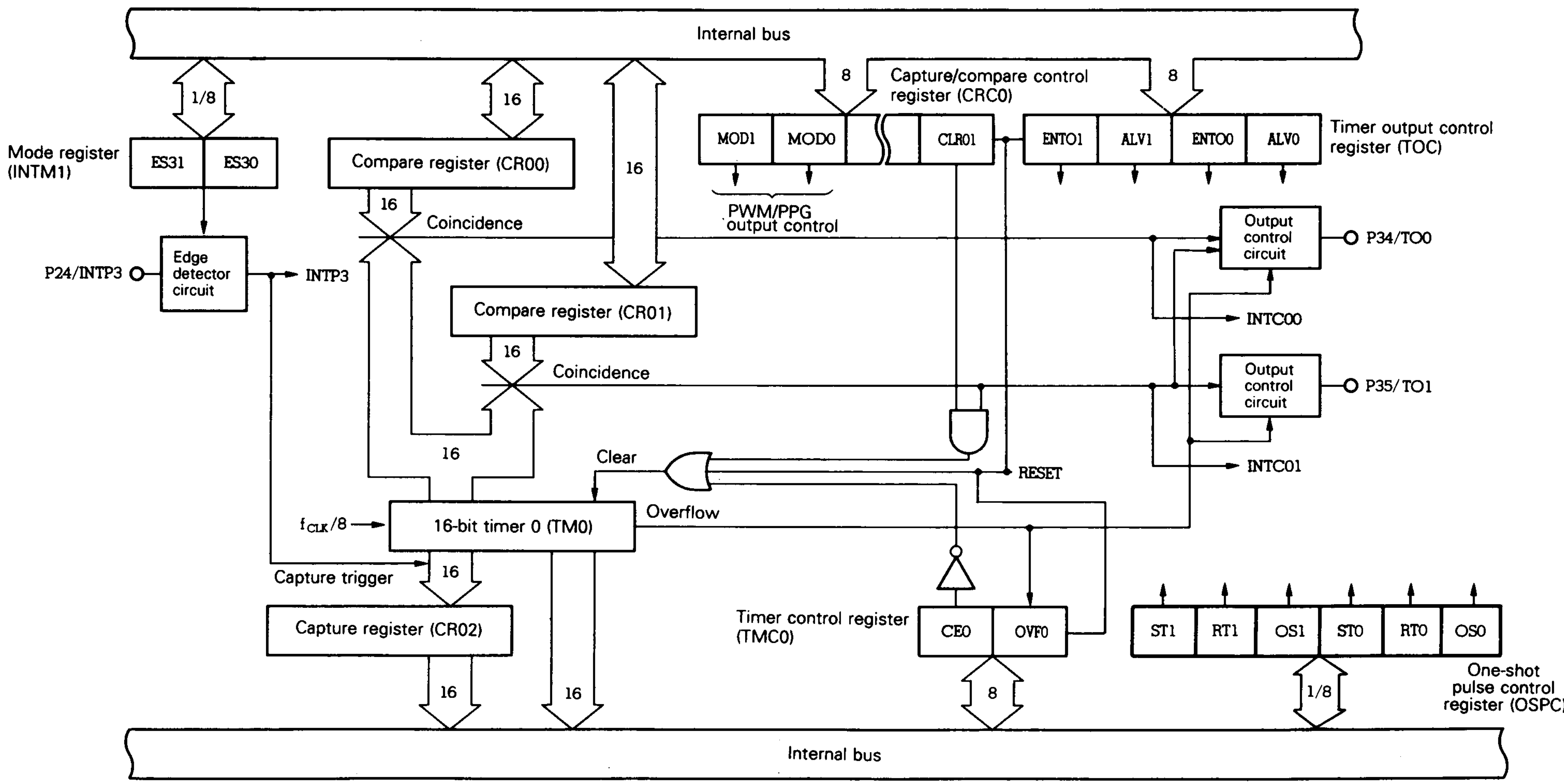
Caution: This software-triggered one-shot pulse output function is different from the one-shot timer function for 8-bit timer/counter 2.

7.1.2 Configuration

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

Figure 7-2 shows the 16-bit timer/counter configuration.

Fig. 7-2 Configuration of 16-bit Timer/Counter



(1) 16-bit timer 0 (TM0)

Timer TM0 counts up count clock $f_{CLK}/8$.

This timer operation can be disabled or enabled by timer control register 0 (TMC0).

The timer contents can only be read by a 16-bit manipulation instruction. When the $\overline{\text{RESET}}$ signal is input, the TM0 contents are cleared to 0000H and TM0 stops counting.

(2) Compare registers (CR00 and CR01)

CR00 and CR01 are 16-bit registers holding values that determine the interval timer cycle.

When the contents of these register coincide with the TM0 value , interrupt requests (INTC00 and INTC01) and a timer output control signal are generated. The TM0 count value can also be cleared, when coincidence takes place.

Data can be read from or written to these registers by a 16-bit manipulation instruction. When the $\overline{\text{RESET}}$ signal is input, the contents of these registers become undefined.

(3) Capture register (CR02)

This 16-bit register captures the TM0 contents.

The CR02 register captures the TM0 contents in synchronization with the valid edge (capture trigger) input to external interrupt request input pin INTP3. The CR02 register contents are retained, until the next capture trigger is generated.

The contents of this register can only be read by a 16-bit manipulation instruction. The contents become undefined, when the $\overline{\text{RESET}}$ signal is input.

(4) Edge detector circuit

This circuit detects the valid edge for a signal input from an external source. It detects the valid edge specified by external interrupt mode register 1 (INTM1), inputs it to the INTP3 input pin and generates interrupt INTP3 and a capture trigger (for INTM1 register details, refer to **Fig. 13-2**).

(5) Output control circuits

These circuits can invert the TM0 output signals when the contents for registers CR00 and CR01 coincide with the timer value. Timer output pins (TO0 and TO1) can output square waves, when so specified by the lower 4 bits of the timer output control register (TOC). At this time, PWM or PPG signals can also be output, when so specified by the capture/compare control register 0 (CRC0).

In addition, software-triggered one-shot pulses can also be output.

The TOC register can also disable or enable the timer output. When the timer output is disabled, pins TO0 and TO1 pins output signals, whose levels are fixed (the output levels are specified by the TOC register).

7.1.3 16-bit timer/counter control registers

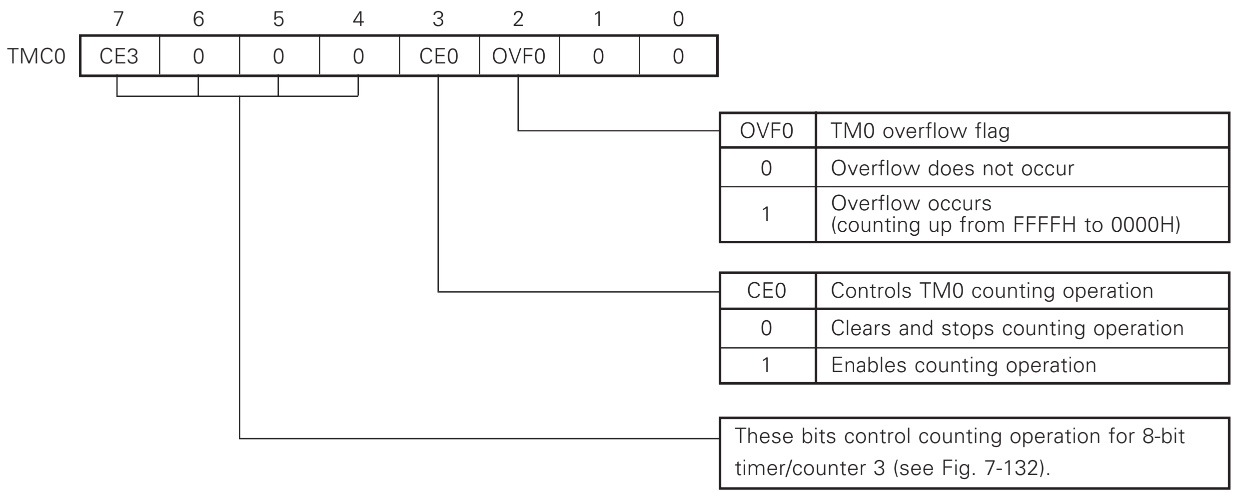
(1) Timer control register 0 (TMC0)

The TMC0 register is an 8-bit register that controls the counting operation for the 16-bit timer 0, TM0. The lower 4 bits of this register are used to control the timer operation. The higher 4 bits are used to control the counting operation for 8-bit timer/counter 3.

Data can be read from or written to this register by an 8-bit manipulation instruction. Figure 7-3 shows the format of this register.

When the $\overline{\text{RESET}}$ signal is input, the TMC0 register contents are cleared to 00H.

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



Remarks: The OVF0 bit can only be cleared by software.

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

The CRC0 register is a register that specifies whether or not the TM0 contents are cleared, when the TM0 count value coincides with the compare register (CR01) contents. This register can also specify timer output modes (TO0 and TO1).

Data can only be written to this register by an 8-bit manipulation instruction. The format for this register is shown in Fig. 7-4.

When the $\overline{\text{RESET}}$ signal is input, the contents of this register are cleared to 10H.

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

	7	6	5	4	3	2	1	0
CRC0	MOD1	MOD0	0	1	CLR01	0	0	0

MOD1	MOD0	CLR01	Specifies timer output mode		Clearing TM0 when TM0=CR01
			TO0	TO1	
0	0	0	Toggle output	Toggle output	Disabled
0	0	1	Toggle output	Toggle output	Enabled
0	1	0	PWM output	Toggle output	Disabled
0	1	1	Inhibited		
1	0	0	PWM output	PWM output	Disabled
1	0	1	Inhibited		
1	1	0	Inhibited		
1	1	1	PPG output	Toggle output	Enabled

(3) Timer output control register (TOC)

This 8-bit register specifies the active level for the timer output pins and disables or enables the timer output. The lower 4 bits of this register control the output operations for the 16-bit timer/counter (i.e., operations for pins TO0 and TO1), while the higher 4 bits control the output operations for 8-bit timer/counter 2 (operations for pins TO2 and TO3).

Data can only be written to this register by an 8-bit manipulation instruction. Figure 7-5 shows the TOC register format.

When the $\overline{\text{RESET}}$ signal is input, the contents of this register are cleared to 00H.

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

	7	6	5	4	3	2	1	0
TOC	ENTO3	ALV3	ENTO2	ALV2	ENTO1	ALV1	ENTO0	ALV0

ALV0	Pin TO0 active level	
	When toggle output is specified and when one-shot pulse is output	When PWM/PPG output is specified
0	Low level	High level
1	High level	Low level

ENTO0	Specifies pin TO0 operation
0	Outputs $\overline{\text{ALV0}}$
1	Enables pulse output

ALV1	Active level for pin TO1	
	When toggle output is specified and when one-shot pulse is specified	When PWM/PPG output is specified
0	Low level	High level
1	High level	Low level

ENTO1	Specifies pin TO1 operation
0	Outputs $\overline{\text{ALV1}}$
1	Enables pulse output

Bits 4 to 7 (ALV2, ENTO2, ALV3, ENTO3): These bits control the output operation for 8-bit timer/counter 2 (pins TO2 and TO3) (see Fig. 7-77).

(4) One-shot pulse output control register (OSPC)

This 8-bit register enables or disables output and output levels for software-triggered one-shot pulses.
Data can be read from or written to this register by an 8-bit manipulation or bit manipulation instruction.
The format for this register is shown in Fig. 7-6.
When the $\overline{\text{RESET}}$ signal is input, the OSPC register contents are cleared to 00H.

Fig. 7-6 One-Shot Pulse Output Control Register (OSPC) Format

	⑦	⑥	5	4	③	②	1	0
OSPC	ST1	RT1	0	OS1	ST0	RT0	0	OS0

OS0	Selects output pulse type
0	Selects toggle output, PWM output, or PPG output
1	Selects software-triggered one-shot pulse

ST0	RT0	Controls TO0 output
0	0	Does not change output
0	1	Outputs inactive level to TO0
1	0	Outputs active level to TO0
1	1	Inhibited

OS1	Selects output pulse type
0	Selects toggle output, PWM output, or PPG output
1	Selects software-triggered one-shot pulse output

ST1	RT1	Controls TO1 output
0	0	Does not change output
0	1	Outputs inactive level to TO1
1	0	Outputs active level to TO1
1	1	Inhibited

Remarks 1: Data can be written to the RT0, ST0, RT1, and ST1 bits for this register. When these bits are read, however, they are always 0s.

2: Disabling or enabling output for pulses from pins TO0 and TO1 and active levels for the output signals are specified by the timer output control register (TOC).

7.1.4 16-bit timer 0 (TM0) operation

(1) Basic operation

The 16-bit timer counts up count clocks which are $f_{CLK}/8$.

When the $\overline{RESET}$ signal is input, the TM0 contents are cleared to 0000H and TM0 stops counting.

The TM0 counting operation is enabled or disabled by bit 3 (CE0) for the timer control register (TMC0). When the CE0 bit is set to 1 by software, the TM0 contents are cleared to 0000H at the first count clock, and then TM0 starts the count-up operation.

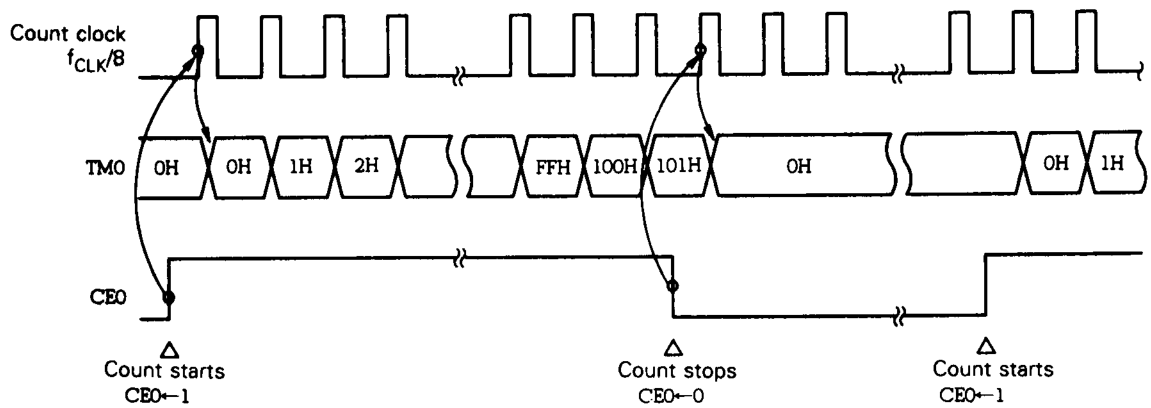
When the CE bit is reset to 0, TM0 is cleared to 0000H at the next count clock, and capture operation and generation for the coincidence signal are stopped.

If an attempt is made to set the CE0 bit, which has already been set to 1, TM0 is not cleared but continues its operation.

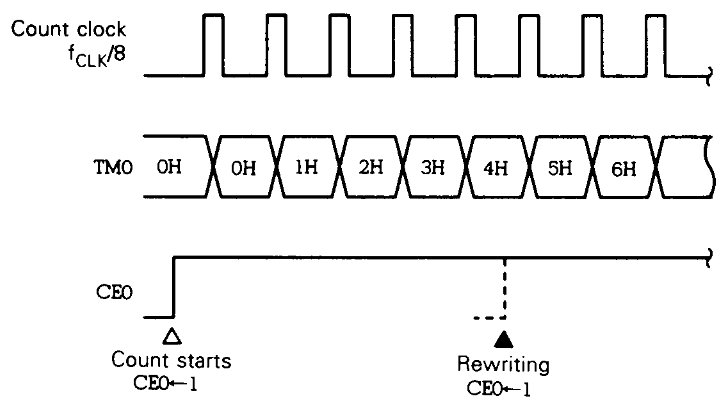
When a count clock is input while the TM0 current value is FFFFH, the timer is cleared to 0000H. At this time, the OVF0 bit is set to 1, sending an overflow signal to the output control circuits. The OVF0 bit can be cleared by software only. At this time, TM0 continues the counting operation.

Fig. 7-7 16-bit Timer 0 (TM0) Basic Operation

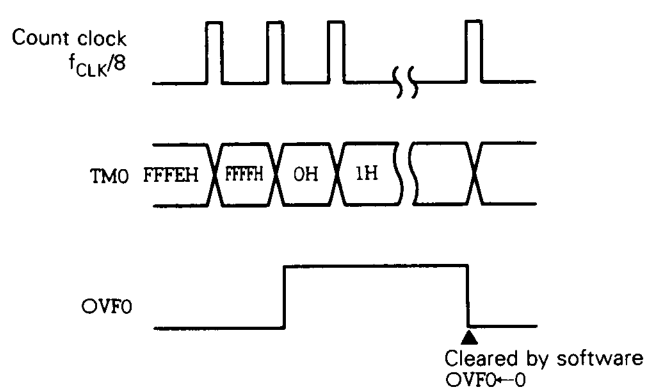
(a) When counting is started, stopped, and then started again



(b) When "1" is written to CE0 bit again after counting is started



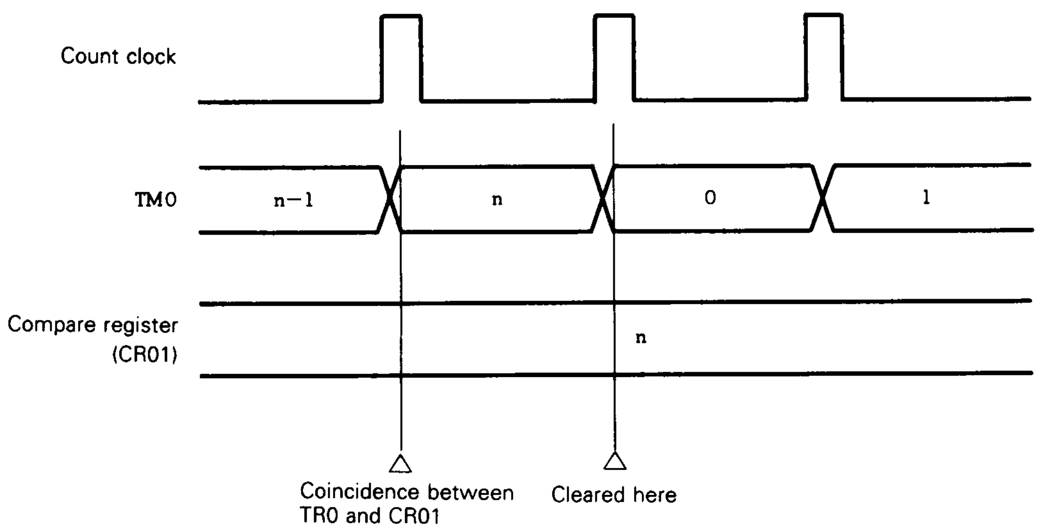
(c) TM0 Operation, when its current value is FFFFH



(2) Clearing operation

The 16-bit timer/counter (TM0) can be automatically cleared, after its value has coincided with the compare register (CR01) contents. When a reason for clearing TM0 has occurred, TM0 is cleared to 0000H at the next count clock. Therefore, even if the reason for clearing has occurred, the current TM0 value is retained, until the next count clock is applied.

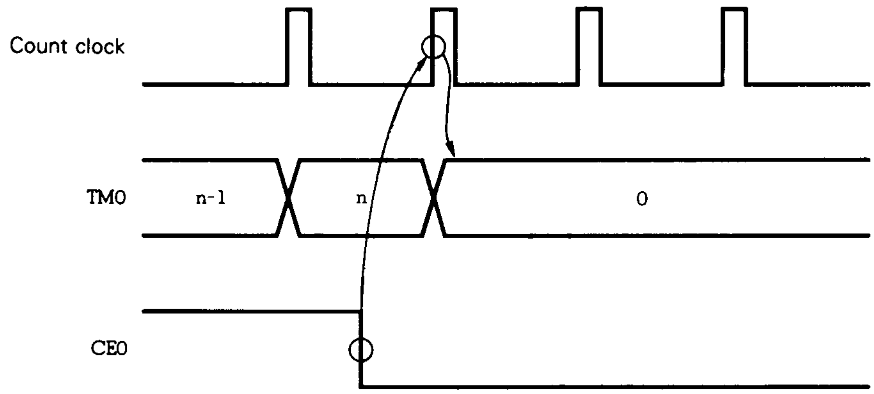
Fig. 7-8 Clearing TM0 by Coincidence with Compare Register (CR01)



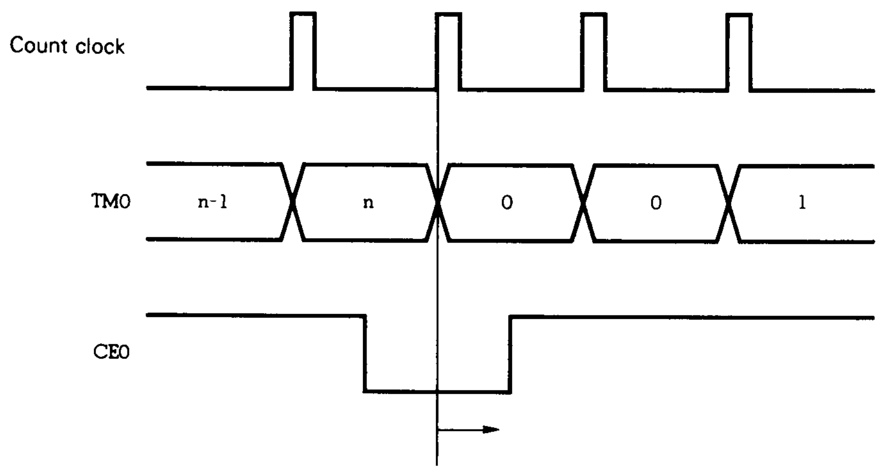
TM0 is also cleared by resetting the CE0 bit of the timer control register (TMC0) to 0 through software. The clear operation is also performed by the count clock after the CE0 bit has been reset to 0. If the CE0 bit is set to 1 after the CE0 bit has been reset to 0 and before TM0 is cleared to 0 (before the first count clock is input after the CE0 bit has been reset to 0), TM0 is cleared to 0 and at the same time, 0 counting operation is performed because counting has been started.

Fig. 7-9 Clear Operation When CE0 Bit Is Reset to 0

(a) Basic operation

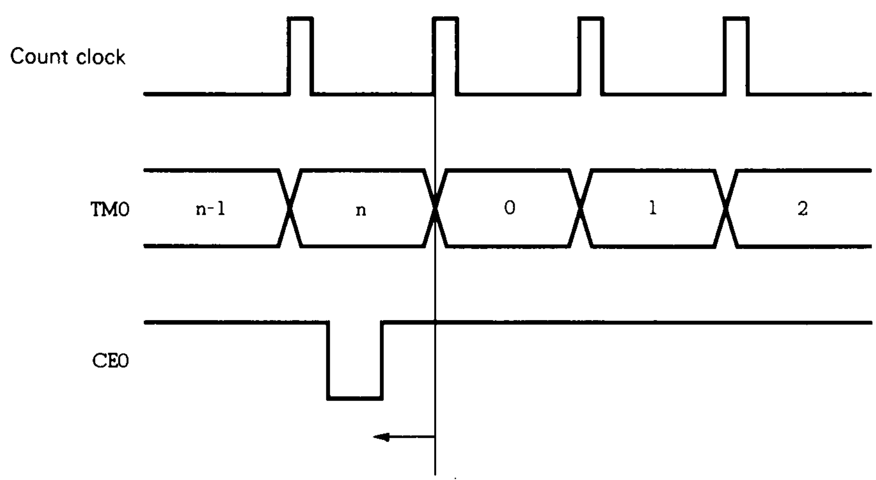


(b) Restart after TM0 has been cleared to 0



If 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 TM0 is cleared to 0



If the CE0 bit is set to 1 before this count clock, TM0 is cleared by CE0 ← 0 and 0 counting is performed by CE0 ← 1 simultaneously.

7.1.5 Operations for compare registers and capture register

(1) Compare operation

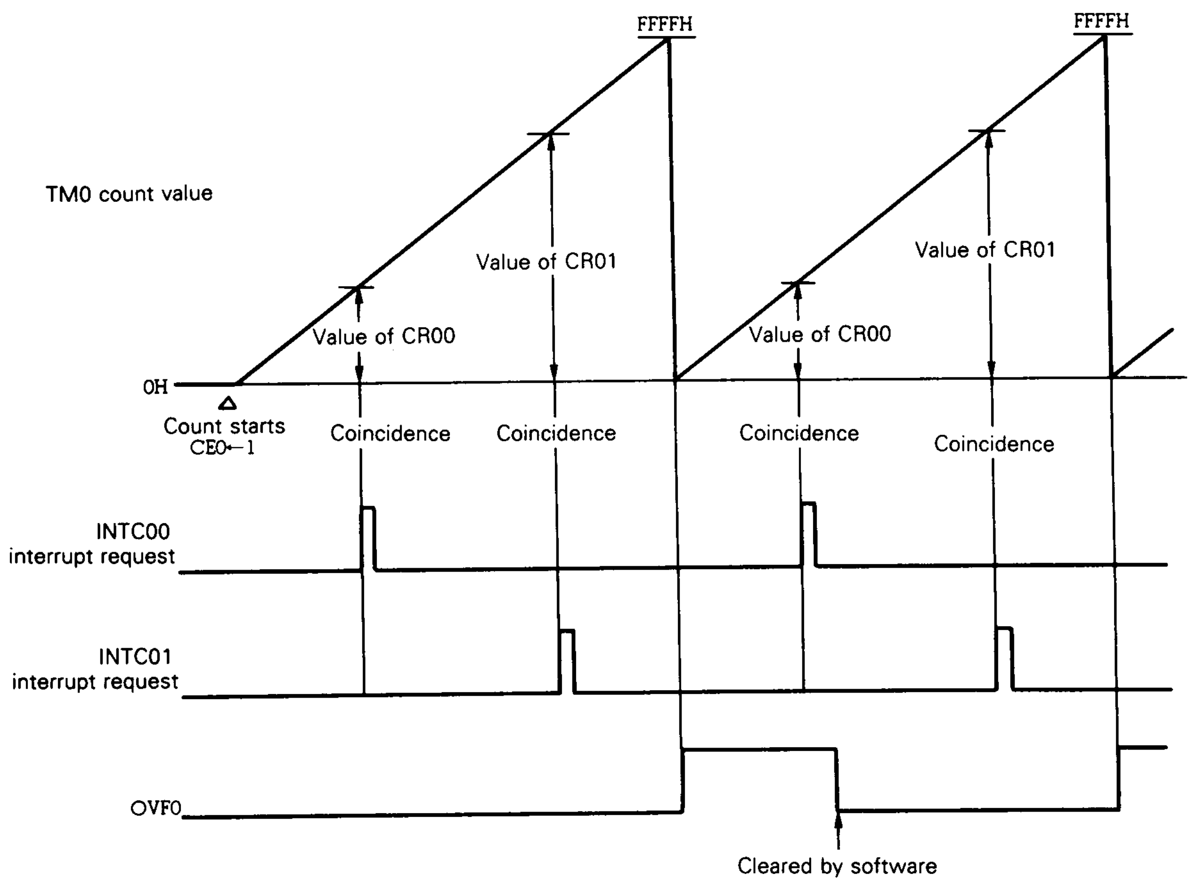
The 16-bit timer/counter can also perform a compare operation, which compares the current count value for the timer with the set values for the compare registers.

When the count value for the 16-bit timer 0 (TM0) coincides with the values set in advance in the compare registers (CR00 and CR01), the timer sends a coincidence signal to the output control circuit. At the same time, interrupt requests (INTC00 and INTC01) are generated.

After the timer value has coincided with the CR01 register value, the TM0 count value can be cleared. In this case, the timer functions as an interval timer that repeatedly counts up to the value set in the CR01 register.

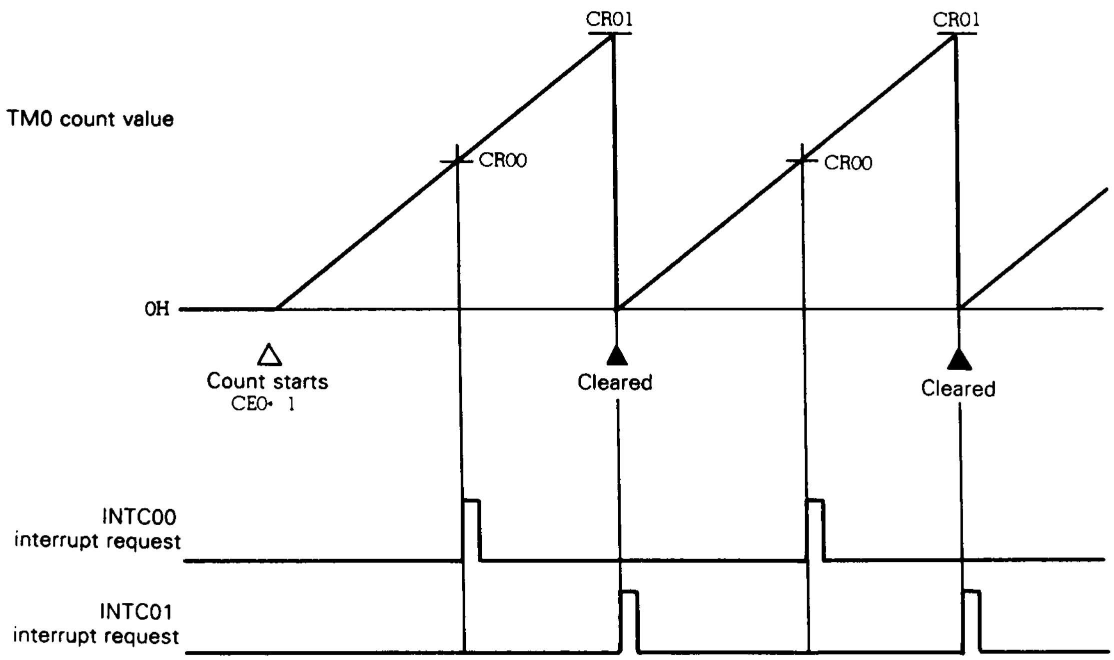
Caution: Before the 16-bit timer/counter performs a compare operation, data must be written to the compare register to be used.

Fig. 7-10 Compare Operation



Remarks: CLR01 = 1

Fig. 7-11 Clearing TM0 after Coincidence Detection



Remarks: CLR01 = 0

(2) Capture operation

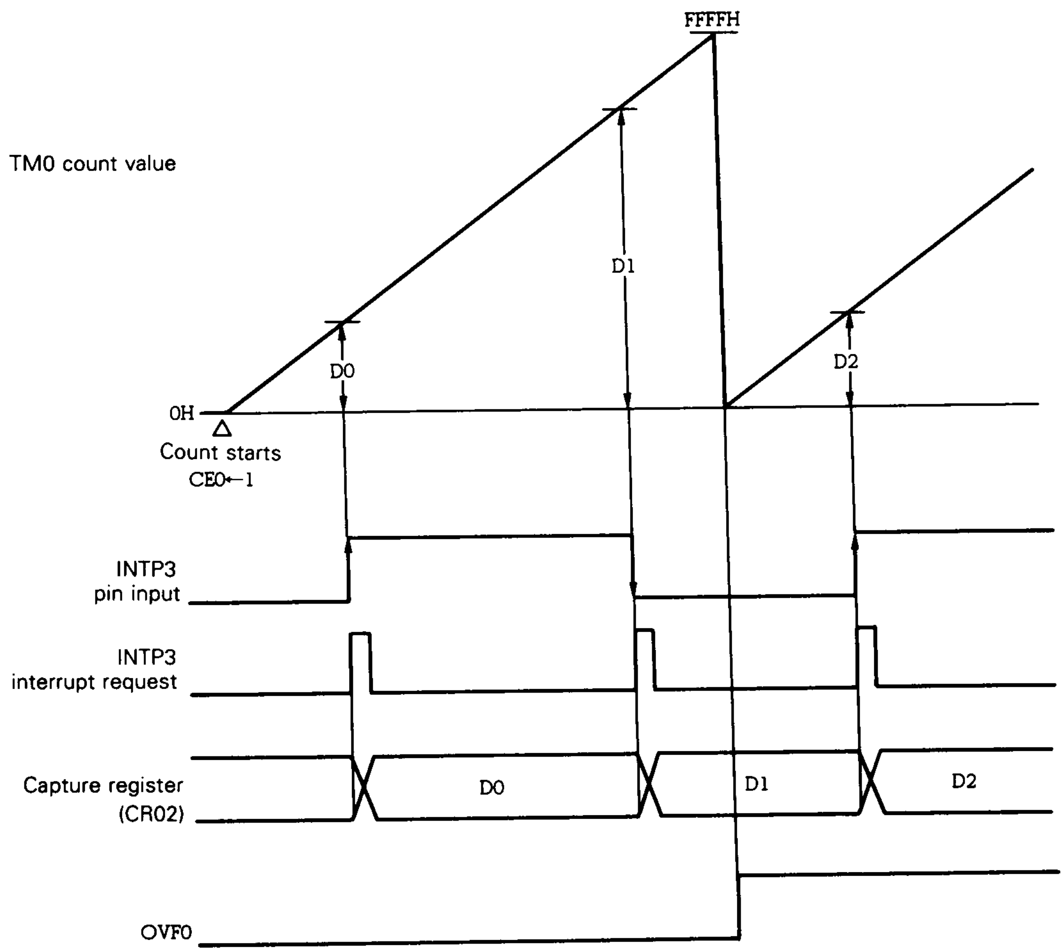
The 16-bit timer/counter can also perform a capture operation by which to capture the count value for the timer to the capture register, which then retains the value, in synchronization with an external trigger.

As the external trigger, the valid edge detected on external interrupt request input pin INTP3 is used (capture trigger). The count value for the 16-bit timer (TM0) is captured to the capture register (CR02) in synchronization with the capture trigger. The CR02 register contents are retained, until the next capture trigger is generated.

The valid edge for the capture trigger is specified by external interrupt mode register 1 (INTM1). If the capture trigger is specified, so that it is detected at both the rising and falling edges, the width of a pulse, input from an external source, can be measured. If the capture trigger is detected at either edge, the input pulse cycle can be measured.

For detailed INTM1 register format, refer to **Fig. 13-2** in **Chapter 13**.

Fig. 7-12 Capture Operation



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

Caution: In the in-circuit emulator, the INTP3 pin cannot normally perform digital noise elimination. When using the capture function, an erroneously detected edge may cause the following:

- **No capture operation will be performed by an erroneously detected edge. However, an interrupt request by edge detection will be generated.**

Therefore, the captured value can be used only after determining if the generated interrupt request is the INTP3 interrupt caused by an erroneously detected edge or a normally generated INTP3 interrupt. Refer to 13.4, "Notes" in CHAPTER 13, "EDGE DETECTION FUNCTION" for details of erroneous edge detection.

7.1.6 Basic operation for output control circuits

The output control circuits control the level for the timer output pins (TO0 and TO1) by using an overflow signal or a coincidence signal for the compare registers. The operation of these circuits is determined by the timer output control register (TOC), capture/compare control register 0 (CRC0), and one-shot pulse output control register (OSPC) (see **Table 7-5**).

To output signals to the timer output pins (TO0 and TO1), the output pins must be set in the control mode in advance by the PMC3 register.

Table 7-5 Timer Output (TO0 and TO1) Operations

TOC				OSPC		CRC0			TO1	TO0
ENTO1	ALV1	ENTO0	ALV0	OS1	OS0	MOD1	MOD0	CLR01		
0	0/1	0	0/1	×	×	×	×	×	Fixed to high/low level	Fixed to high/low level
0	0/1	1	0/1	×	0	0	0	×	Fixed to high/low level	Toggle output (low/high active)
0	0/1	1	0/1	×	0	0	1	0	Fixed to high/low level	PWM output (high/low active)
0	0/1	1	0/1	×	0	1	0	0	Fixed to high/low level	PWM output (high/low active)
0	0/1	1	0/1	×	0	1	1	1	Fixed to high/low level	PPG output (high/low active)
0	0/1	1	0/1	×	1	×	×	×	Fixed to high/low level	One-shot pulse output (low/high active)
1	0/1	0	0/1	0	×	0	×	×	Toggle output (low/high active)	Fixed to high/low level
1	0/1	0	0/1	0	×	1	0	0	PWM output (high/low active)	Fixed to high/low level
1	0/1	0	0/1	0	×	1	1	×	Toggle output (low/high active)	Fixed to high/low level
1	0/1	0	0/1	1	×	×	×	×	One-shot pulse output (low/high active)	Fixed to high/low level
1	0/1	1	0/1	0	0	0	0	×	Toggle output (low/high active)	Toggle output (low/high active)
1	0/1	1	0/1	0	0	0	1	0	Toggle output (low/high active)	PWM output (high/low active)
1	0/1	1	0/1	0	0	1	0	0	PWM output (high/low active)	PWM output (high/low active)
1	0/1	1	0/1	0	0	1	1	1	Toggle output (low/high active)	PPG output (high/low active)
1	0/1	1	0/1	0	1	0	×	×	Toggle output (low/high active)	One-shot pulse output (low/high active)
1	0/1	1	0/1	0	1	1	0	0	PWM output (high/low active)	One-shot pulse output (low/high active)
1	0/1	1	0/1	0	1	1	1	1	Toggle output (low/high active)	One-shot pulse output (low/high active)
1	0/1	1	0/1	1	0	0	0	×	One-shot pulse output (low/high active)	Toggle output (low/high active)
1	0/1	1	0/1	1	0	0	1	0	One-shot pulse output (low/high active)	PWM output (high/low active)
1	0/1	1	0/1	1	0	1	0	0	One-shot pulse output (low/high active)	PWM output (high/low active)
1	0/1	1	0/1	1	0	1	1	1	One-shot pulse output (low/high active)	PPG output (high/low active)
1	0/1	1	0/1	1	1	×	×	×	One-shot pulse output (low/high active)	One-shot pulse output (low/high active)

Remarks:
1. The values on the left and the right of "/" in the ALVn (n=0 or 1) column, respectively, correspond to the statuses on the left and the right of "/" in the TOn (n=0 or 1) column.
2. x's indicate don't care bits and the operation is the same, regardless of whether these bits are 0 or 1. Note, however, that some combinations of these bits are inhibited (See **Fig. 7-4**).
3. Combinations not listed in this table are inhibited.

(1) Basic operation

The output timing for the timer output pins (TO0 and TO1) can be changed as specified by the MOD0, MOD1, and CLR01 bits for the capture/compare control register 0 (CRC0) and one-shot pulse output control register (OSPC), when the ENTOn (n = 0 or 1) bit for the timer output control register (TOC) is set to 1.

By clearing the ENTOn (n = 0 or 1) bit to 0, the levels for the timer output pins (TO0 and TO1) can be fixed. The levels at which the output pins are fixed are determined by the ALVn (n = 0 or 1) bit of the timer output control register (TOC). That is, when the ALVn bit is 0, the output levels are fixed at high level, while the output levels are made low when the ALVn bit is 1.

(2) Toggle output

Toggle output is an operation mode in which the output levels for the timer output pins are inverted, each time the values for the compare registers (CR00 and CR01) coincide with the 16-bit timer 0 (TM0) value. The output level for pin TO0 is inverted when the CR00 value coincides with the TM0 value, and the TO1 output level is inverted, when the CR01 value coincides with the TM0 value.

When the 16-bit timer/counter 0 is stopped by resetting the CE0 bit of the TMC0 register to 0, the output level of the timer/counter is retained.

Fig. 7-13 Toggle Output Operation

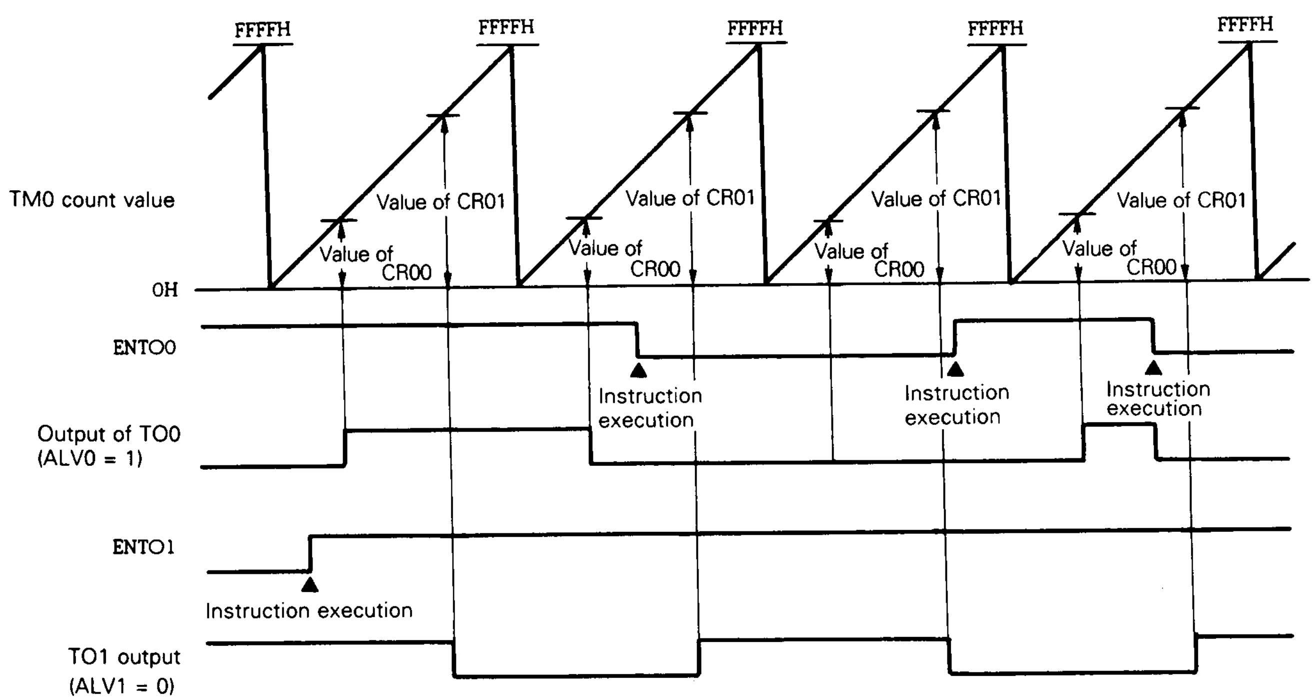


Table 7-6 TO0 and TO1 Toggle Outputs (f_{CLK} = 6 MHz)

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

7.1.7 PWM output

This is an output mode in which the PWM signal is output whose cycle consists of a period during which the 16-bit timer 0 (TM0) counts up to the maximum value. The TO0 pulse width is determined by the CR00 value, and the TO1 pulse width is determined by the CR01 value. To use this function, it is necessary to clear the CLR01 bit for the capture/compare control register 0 (CRC0) to 0.

The pulse cycle and pulse width are related to each other, as follows:

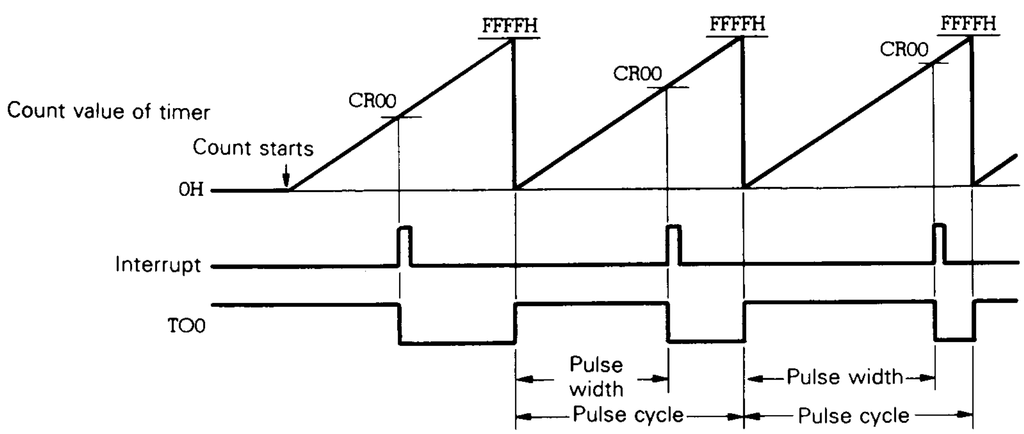
- PWM period = $524288/f_{CLK}$
- PWM pulse width = ((set value for compare register*) x 8 + 2)/f_{CLK} ≒ (set value for compare register) x 8/f_{CLK}

***:** 0 cannot be set in the compare register.

- Duty factor = PWM pulse width/PWM period = (set value for compare register x 8 + 2)/65536 x 8
  ≒ set value for compare register/655336

Caution: The PWM output pulse width is two f_{CLK} clocks longer than the result of the above approximate expression at the active level, and is two f_{CLK} clocks shorter at the inactive level. If high-accuracy output is necessary, take these points into consideration.

Fig. 7-14 PWM Pulse Output



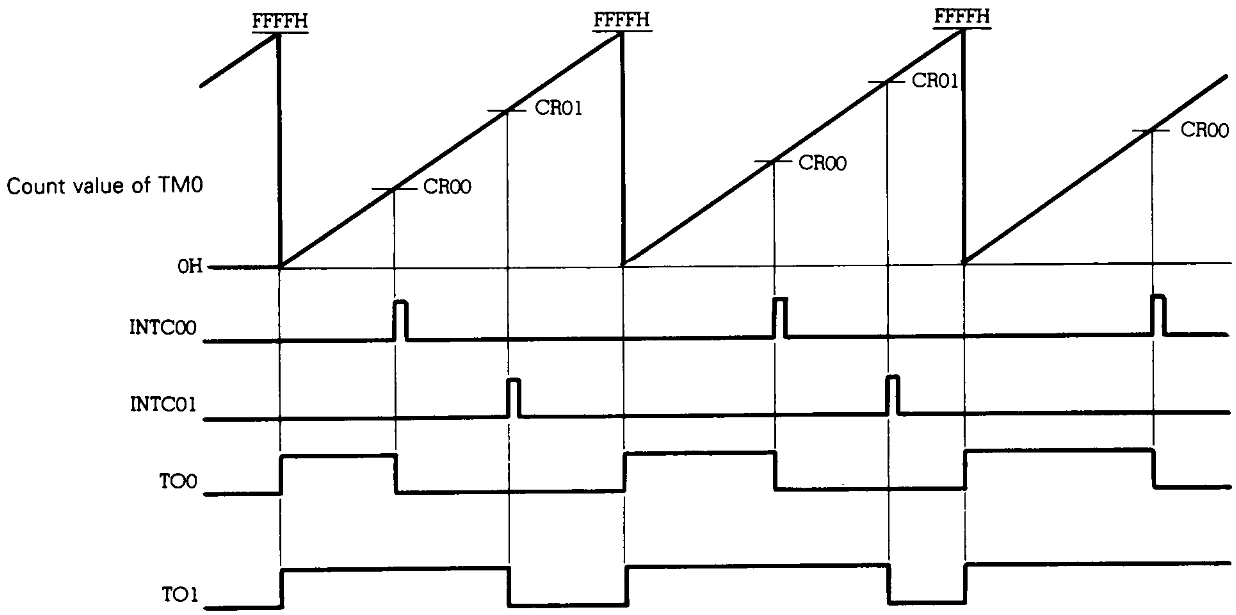
Remarks: ALV0 = 0

Table 7-7 PWM Cycles for TO0 and TO1 (f_{CLK} = 6 MHz)

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

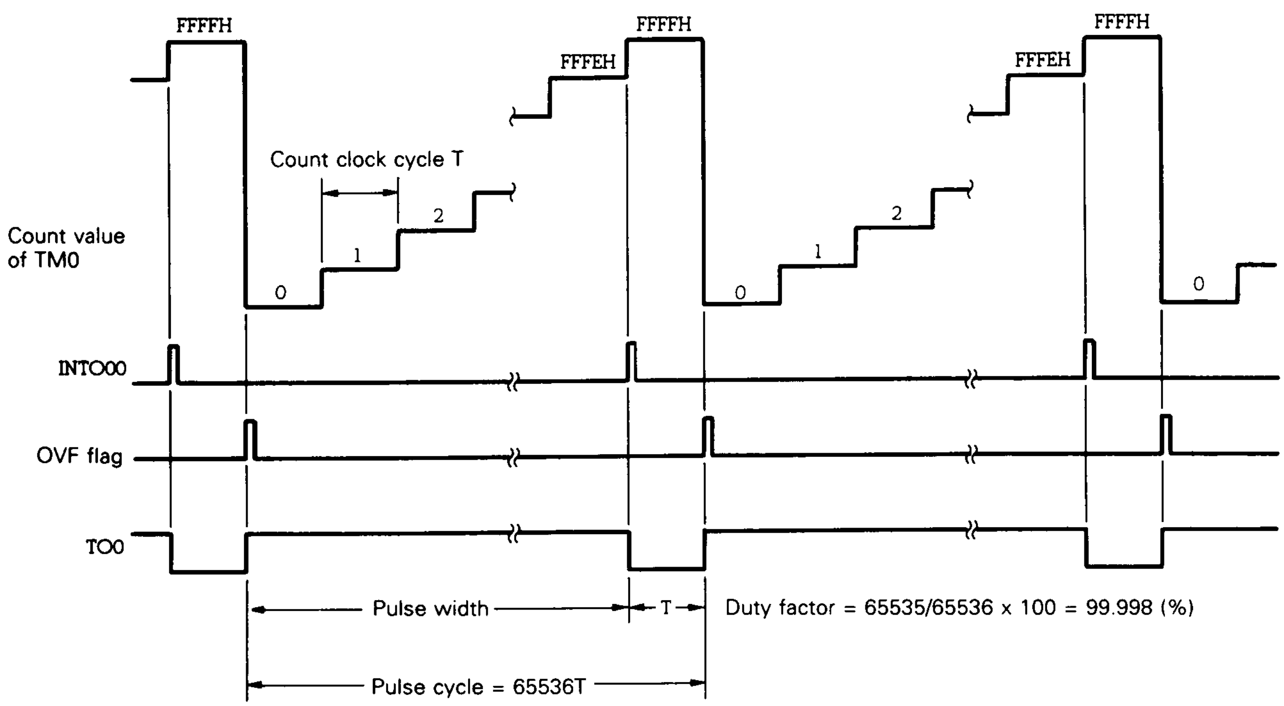
Fig. 7-15 shows a 2-channel PWM output example. Fig. 7-16 shows setting the value FFFFH in the compare register.

Fig. 7-15 PWM Output Example Using TM0



Remarks: ALV0 = 0, ALV1 = 0

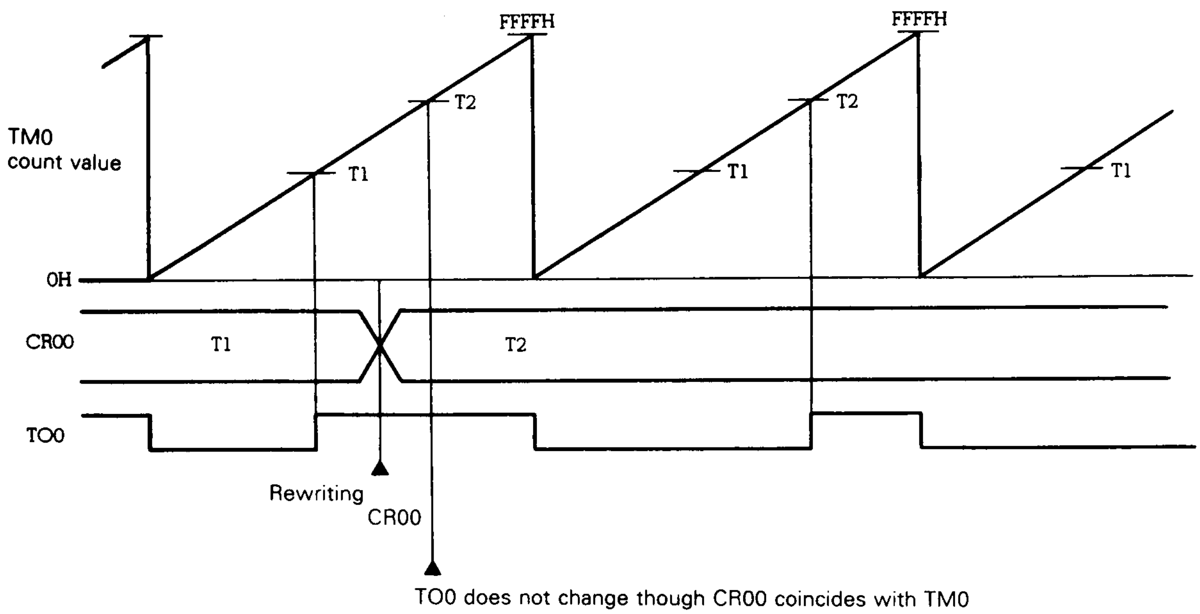
Fig. 7-16 PWM Output Example When CR00 = FFFFH



Remarks: ALV0 = 0

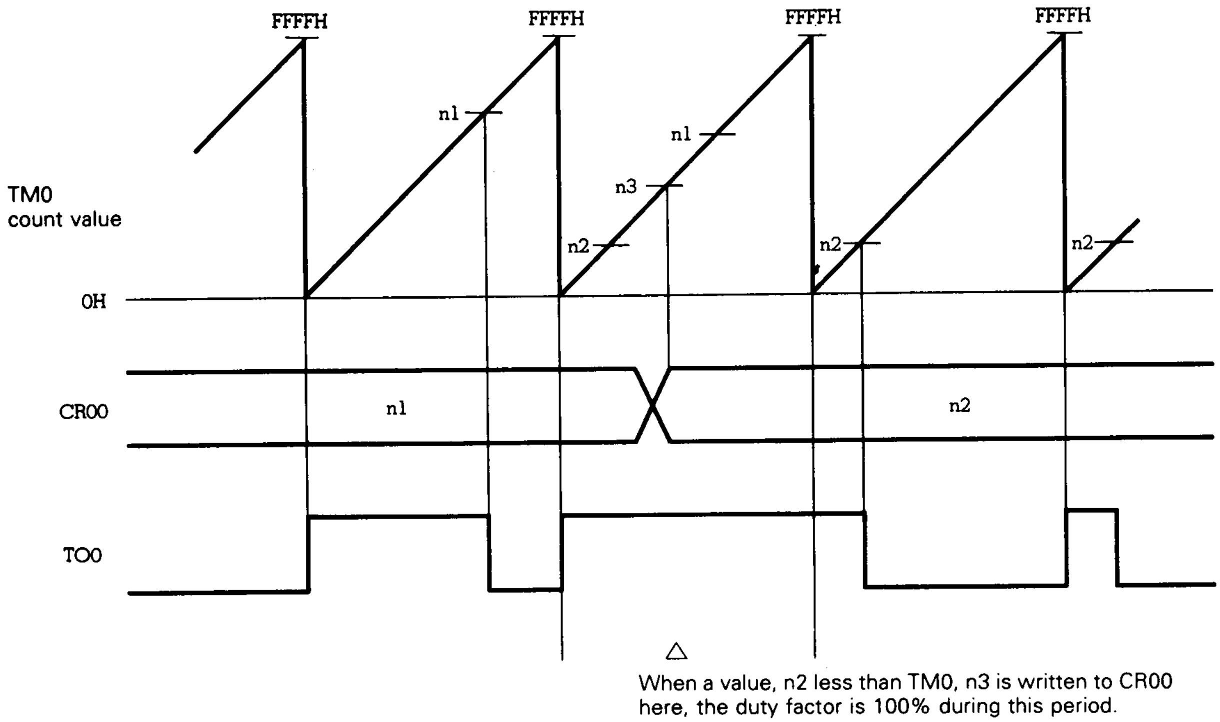
Note that, even if the compare registers (CR00 and CR01) values coincide with the 16-bit timer 0 (TM0) value more than once during one PWM output cycle, the output levels for timer output pins TO0 and TO1 do not change.

Fig. 7-17 Rewriting Compare Registers Contents



Caution: If values less than the value of the 16-bit timer 0 (TM0) are set to the compare registers (CR00, CR01), a PWM signal with a duty factor of 100% is output. Rewrite the values of CR00 and CR01 by using an interrupt that occurs when TM0 coincides with a compare register(CR00, CR01).

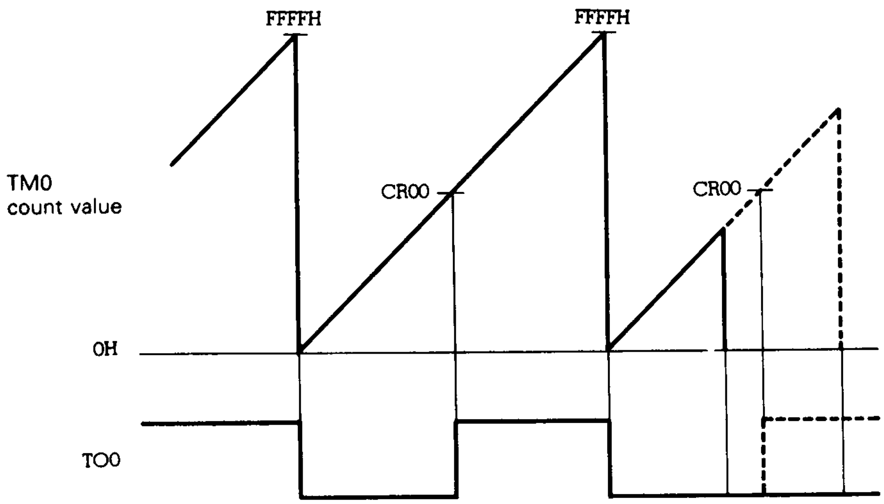
Fig. 7-18 PWM Output Example When Duty Factor Is 100%



Remarks: ALV0 = 0

When the 16-bit timer/counter 0 is stopped by resetting the CE0 bit of TMC0 to 0 while the PWM signal is output, the output level of the timer/counter is retained as is.

Fig. 7-19 When Stopping 16-bit Timer/Counter 0 during PWM Output



Remarks: ALV0 = 1

Caution: When the timer output is disabled (ENTOn = 0, n = 0, 1), the output level of the TOn (n = 0, 1) pin is a complement of the value set to ALVn (n = 0, 1). Note, therefore, that the active level is output, when the timer output is disabled with the PWM output function selected.

7.1.8 PPG output

This function is to output square waves, whose pulse width is determined by compare register CR00 value. The cycle for the pulse is determined by the compare register CR01 value. Therefore, this function is to vary the PWM cycle for the PWM output. The square waves can be output from the pin TO0 only.

To use this function, the CLR01 bit for the capture/compare control register (CRC0) must be set to 1.

The pulse cycle and pulse width are related to each other as follows:

- PPG period = (set value for compare register CR01 + 1) x 8/f_{CLK}
- PPG pulse width = ((set value for compare register CR00) x 8 + 2)/f_{CLK} ≒ set value for CR00 x 8/f_{CLK}

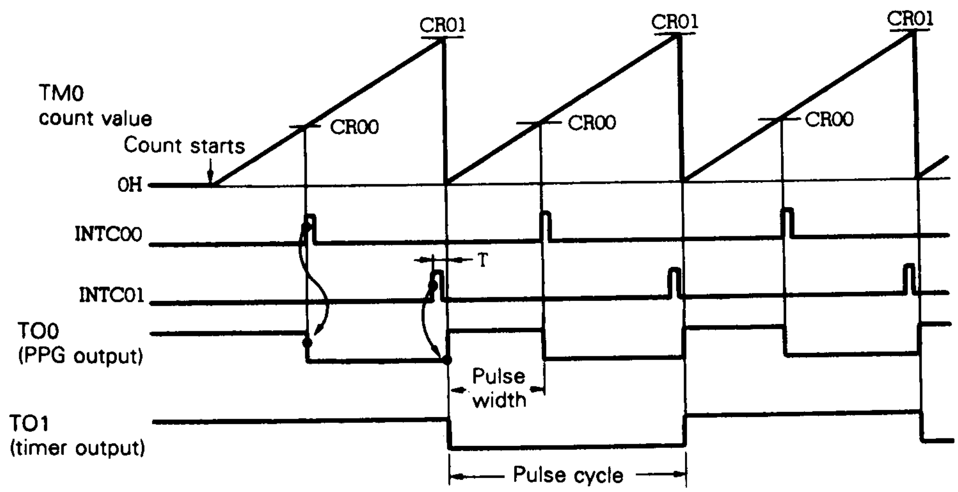
where, CR00 ≦ CR01

- Duty factor = PPG pulse width/PPG period = set value for CR00 x 8 + 2/(set value for CR01 + 1) x 8
  ≒ set value for CR00/set value of CR01 + 1

Caution: The PPG output pulse width is two f_{CLK} clocks longer than the result of the above approximate expression at the active level, and is two f_{CLK} clocks shorter at the inactive level. If high-accuracy output is necessary, or if the PPG pulse period is short, take these points into consideration.

Fig. 7-20 shows an example of the PPG output using the 16-bit timer 0 (TM0). Fig. 7-21 shows an example where CR00 = CR01. In the example shown in Fig. 7-22, CR00 = 0000H.

Fig. 7-20 PPG Output Example, Using TM0



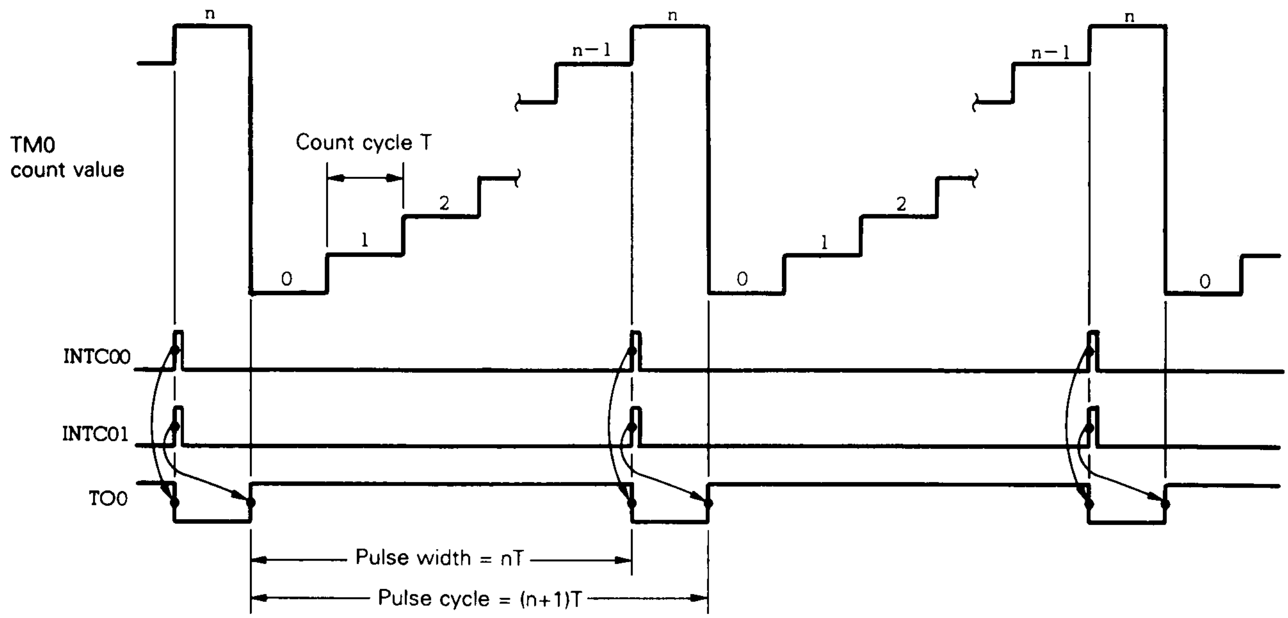
Remarks: ALV0 = 0, ALV1 = 0

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

Count clock	Minimum pulse*	Repeat cycle	PPG frequency
$f_{CLK}/8$	1.3 μs	2.6 μs - 87.4 ms	385 kHz - 11.4 Hz

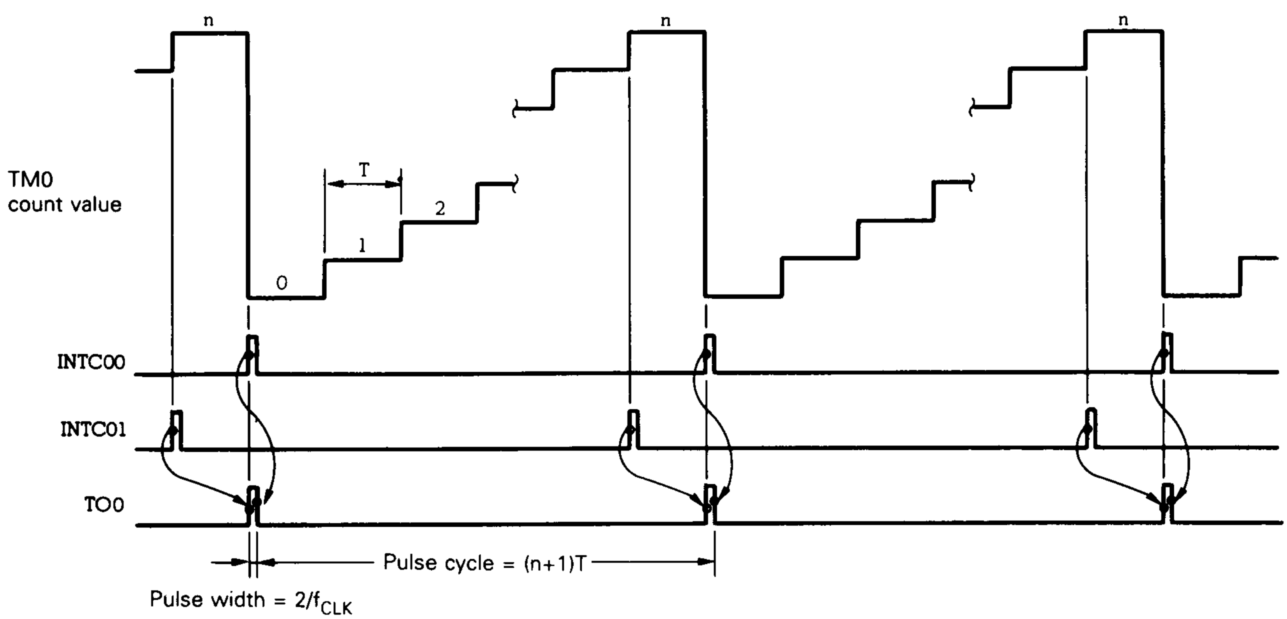
***:** Except when CR00 = 0

Fig. 7-21 PPG Output Example, When CR00 = CR01



Remarks: ALV0 = 0

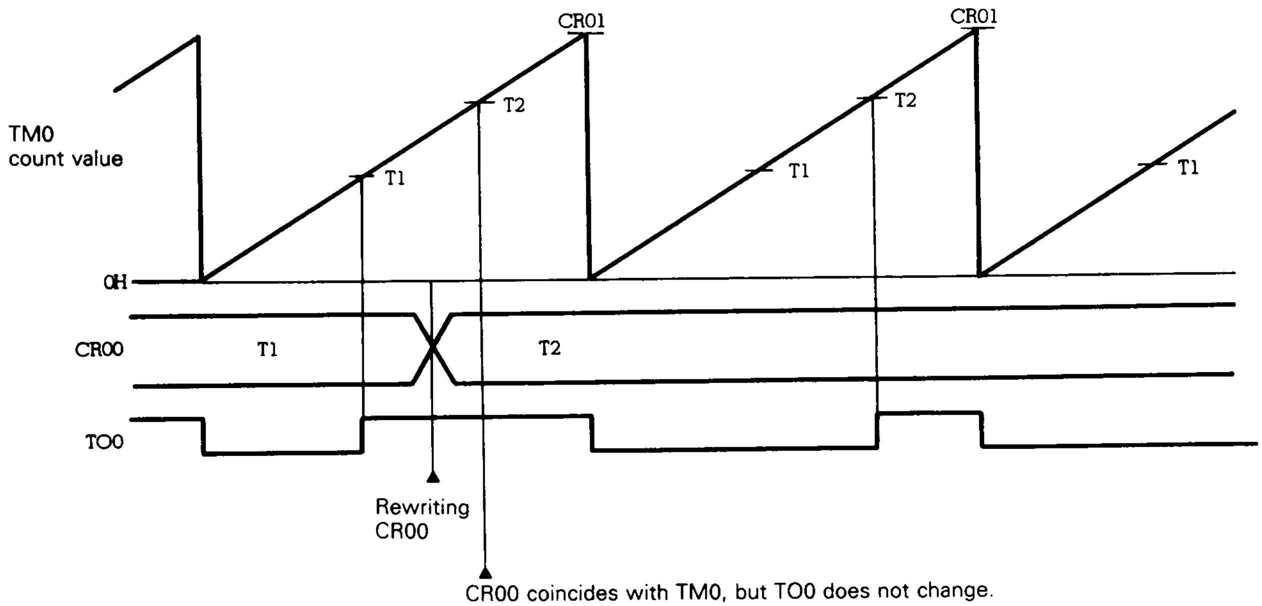
Fig. 7-22 PPG Output Example, When CR00 = 0000H



Remarks: ALV0 = 0

The output levels for timer outputs (TO0 and TO1) do not change, even if the values for the compare registers (CR00 and CR01) coincide with the 16-bit timer 0 (TM0) value more than once during one PPG output period.

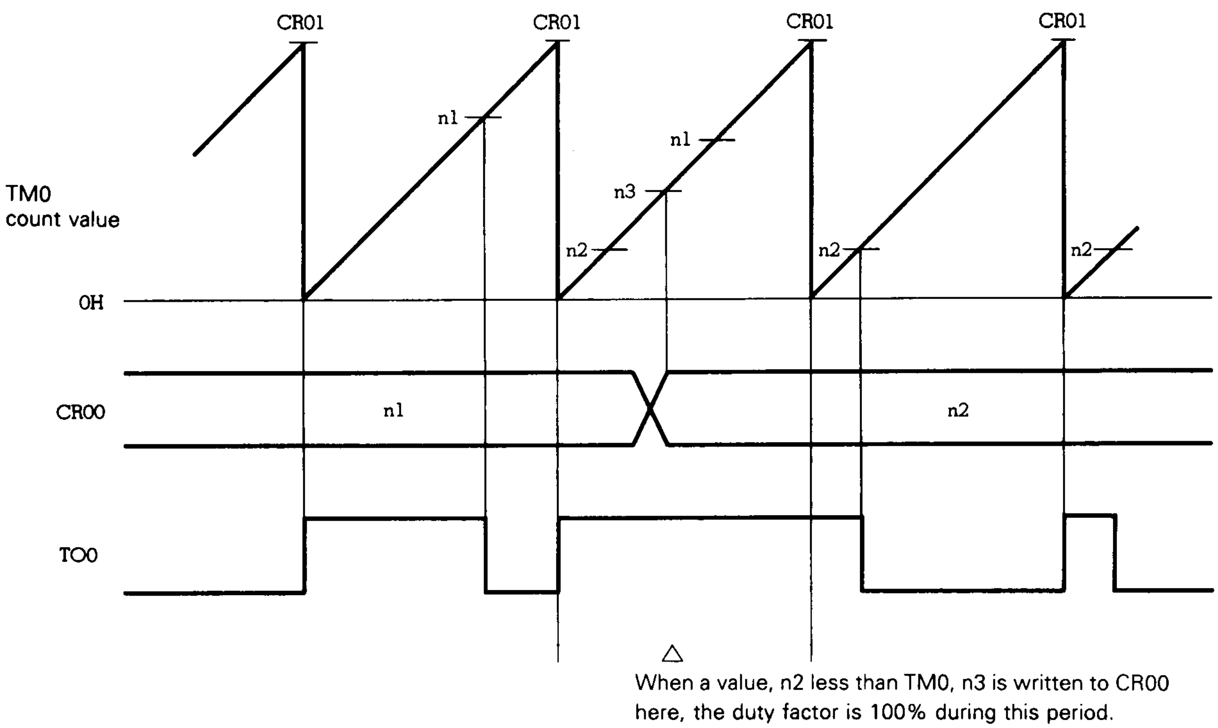
Fig. 7-23 Rewriting Compare Register Example



Remarks: ALV0 = 1

Caution: 1. If a value less than that of 16-bit timer 0 (TM0) is written to compare register CR00 before the value of the CR00 coincides with that of TM0, the duty factor of the PPG cycle becomes 100%. To rewrite the value of CR00, use an interrupt that occurs when the value of CR00 coincides with that of TM0.

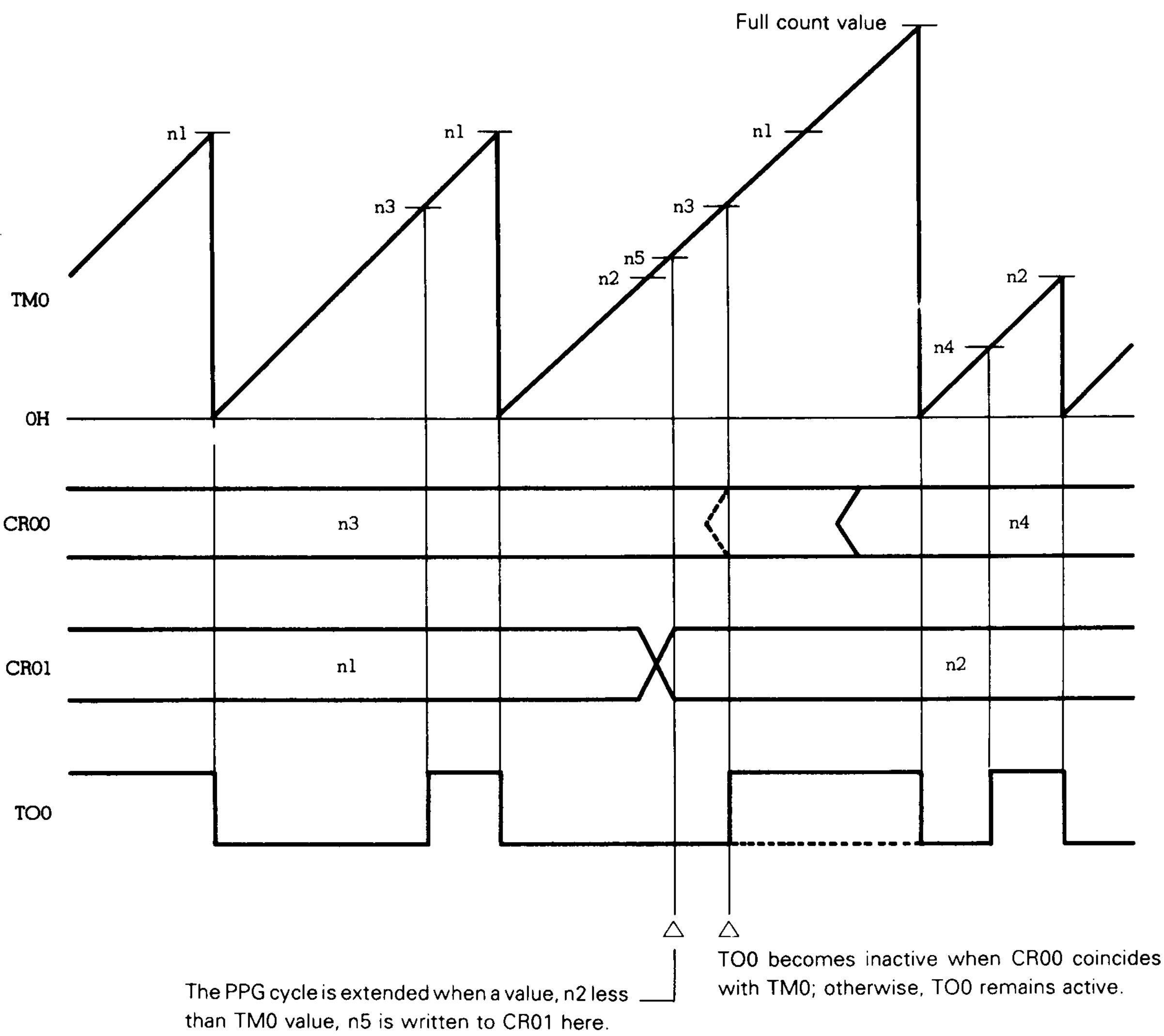
Fig. 7-24 PPG Output Example When Duty Factor Is 100%



Remarks: ALV0 = 0

Caution: 2. When the value of compare register (CR01) is to be changed to a value less than the current value, if a value less than the value of 16-bit timer 0 (TM0) is set to CR01, the PPG cycle is extended to the time required for full count of TM0. At this time, the output level becomes inactive until TM0 overflows and is cleared to 0 if the value of CR01 is rewritten after its value has coincided with TM0, returning to the normal PPG output. If the value of CR01 is changed before CR00 coincides with TM0, the active level is output until CR00 coincides with TM0. If CR00 coincides with TM0 before TM0 overflows and is cleared to 0, the inactive level is output at that point, and the active level is output when TMM0 is cleared to 0, and the normal PPG output is restored.
Rewrite the value of CR01 by using an interrupt that occurs when CR01 coincides with TM0.

Fig. 7-25 PPG Output Example When Output Cycle Is Extended

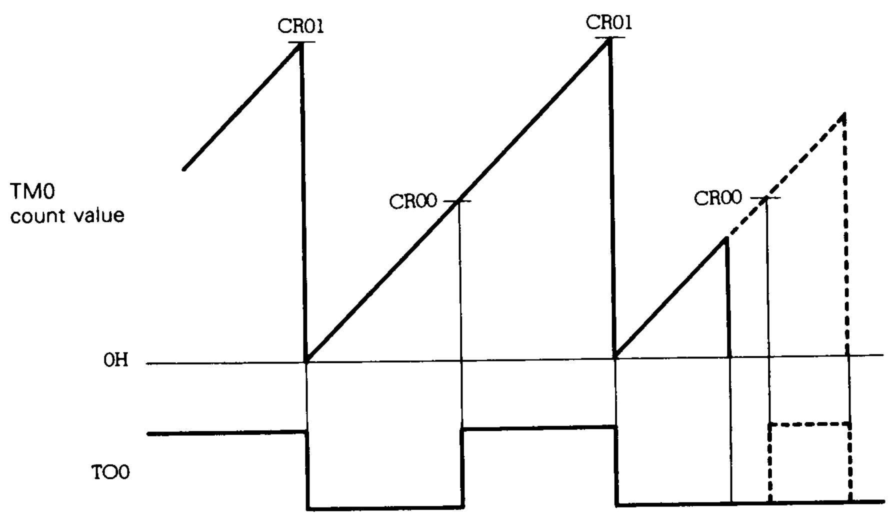


Remarks: ALV0 = 1

Caution: 3. If the PPG cycle is extremely short in respect to the time required for accepting an interrupt, measures described in Notes 1 and 2 above are not effective. Take other measures (such as masking all the interrupts and polling the interrupt flags through software).

When the 16-bit timer/counter 0 is stopped by resetting the CE0 bit of TMC0 to 0 while the PPG signal is output, the active level is output regardless of the output level of the timer/counter.

Fig. 7-26 When Stopping 16-bit Timer/Counter 0 during PPG Output



Caution: When the timer output is disabled (ENTOn = 0, n = 0, 1), the output level for the TOn (n = 0, 1) pin is a complement of the value set to ALVn (n = 0, 1). Note, therefore, that the active level is output, when the timer output is disabled with the PPG output function selected.

7.1.9 Software-triggered one-shot pulse output

The software-triggered one-shot pulse output mode functions to output a one-shot pulse by software.

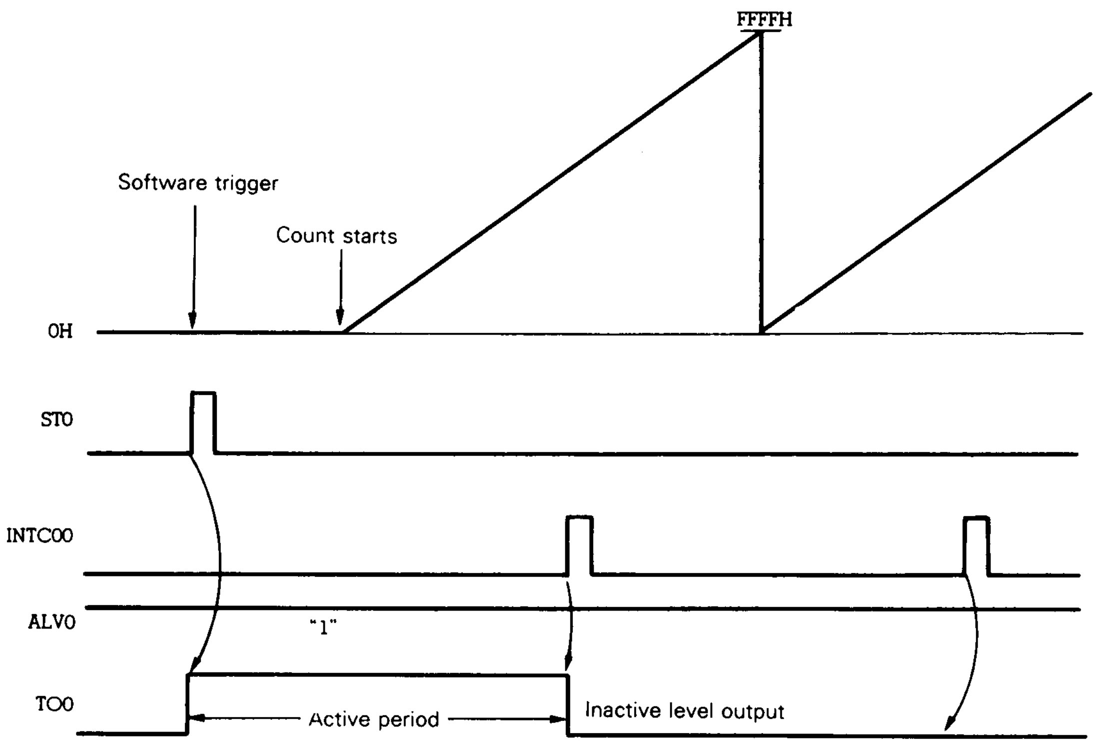
The level for pin TOn (n = 0 or 1) becomes active when the STn bit (n = 0 or 1) for the one-shot pulse output control register (OSPC) is set to 1. After that, pin TOn remains in the active level, until the TM0 value coincides with the CR0n value (n = 0 or 1). When coincidence takes place, pin TOn level becomes inactive, which is maintained until the STn bit is set again. Pin TOn level can also be made inactive by setting the RTn (n = 0 or 1) bit to 1. In this case, pin TOn level remains inactive until the STn bit is set.

Active levels for pins TO0 and TO1 can be controlled independently.

Fig. 7-27 shows an example of software-triggered one-shot pulse output.

When the 16-bit timer/counter 0 is stopped by resetting the CE0 bit of the TMC0 register to 0, the level of the timer/counter is retained as is.

Fig. 7-27 Software-Triggered One-Shot Pulse Output Example



Caution: Do not write "1" to the STn and RTn bits at the same time.

7.1.10 Application examples

(1) Operation as interval timer (1)

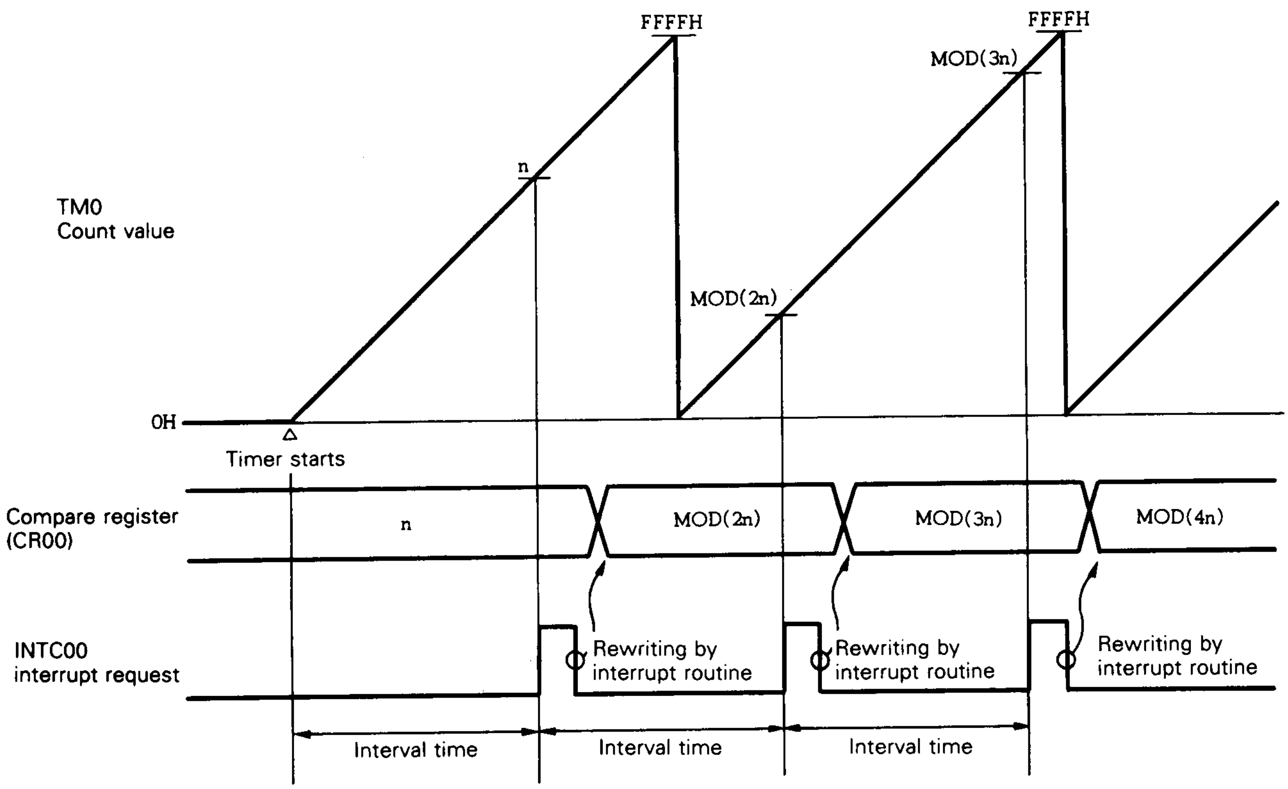
When the 16-bit timer 0 (TM0) is used as a free-running timer and when a fixed value is added to compare registers (CR00 and CR01) contents by an interrupt routine, the timer operates as an interval timer, whose cycle is determined by the value to be added to the compare register contents (see **Fig. 7-28**).

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

Since the 16-bit timer 0 is provided with two compare registers, the timer can operate as an interval timer having two cycles.

Figure 7-29 shows the control register contents. Figure 7-30 shows the procedure used to set the control register contents. Figure 7-31 shows the processing performed by the interrupt routine.

Fig. 7-28 Interval Timer Operation (1) Timing



Remarks: Interval time = n x 8/f_{CLK}, where 1≦ n ≦ FFFFH

Fig. 7-29 Control Register Contents for Interval Timer Operation (1)

(a) Timer control register 0 (TMC0)

	7	6	5	4	3	2	1	0
TMC0	×	0	0	0	1	0	0	0

Bit 2 → Overflow flag
Bit 3 → Count enable of TM0

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

	7	6	5	4	3	2	1	0
CRC0	0	0	0	1	0	0	0	0

Bit 3 → Disables clearing TM0
Bits 7, 6 → Both TO0 and TO1 are used for toggle output

Fig. 7-30 Interval Timer Operation (1) Setting Procedure

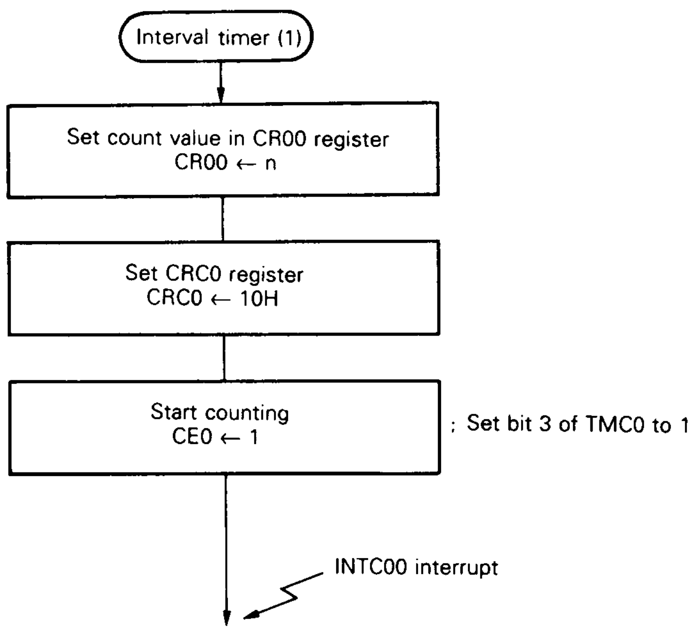
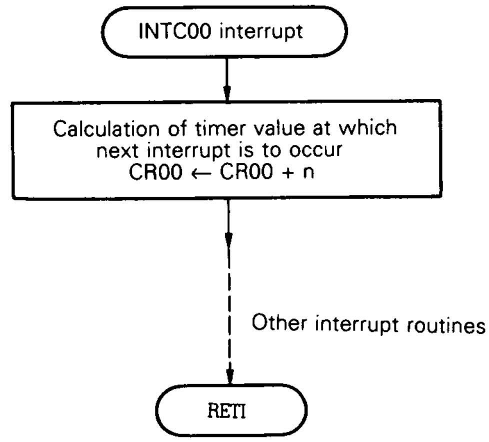


Fig. 7-31 Interrupt Request Processing for Interval Timer Operation (1)



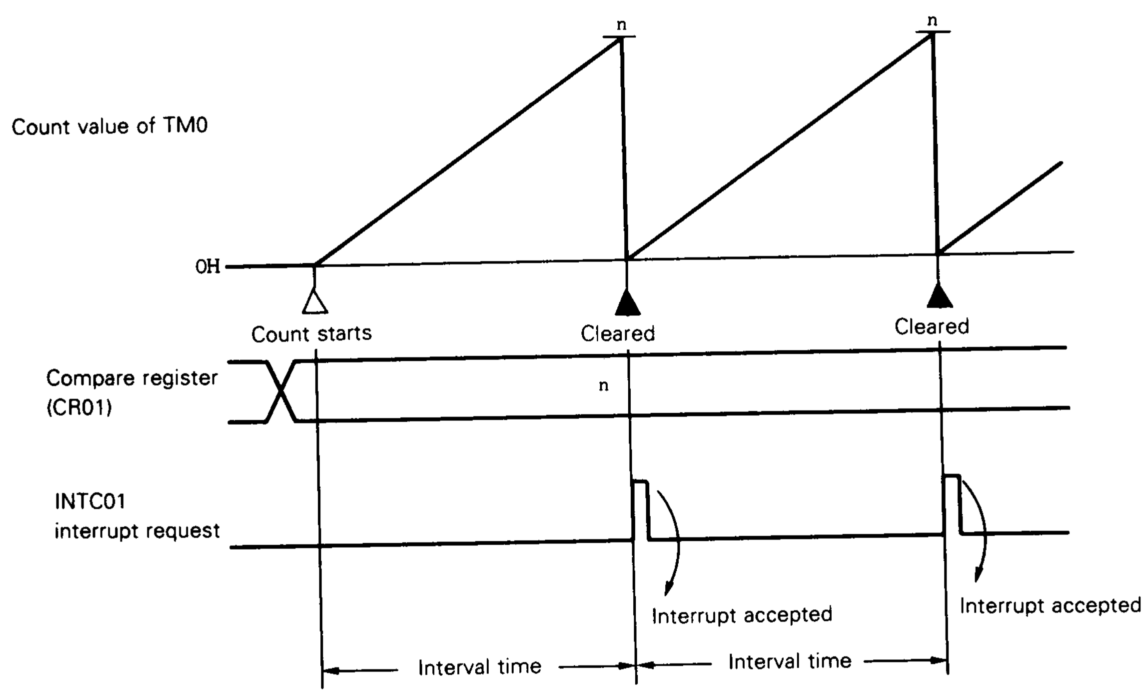
(2) Operation as interval timer (2)

In this example, the 16-bit timer operates as an interval timer that repeatedly generates an interrupt at predetermined time intervals (see **Fig. 7-32**).

In this case, the timer can count from 1.3 μs to 87.4 ms with a 1.3 μs resolution of (at internal system clock f_{CLK} = 6 MHz).

Fig. 7-33 shows the control register contents, while Fig. 7-34 shows the procedure used to set the register contents.

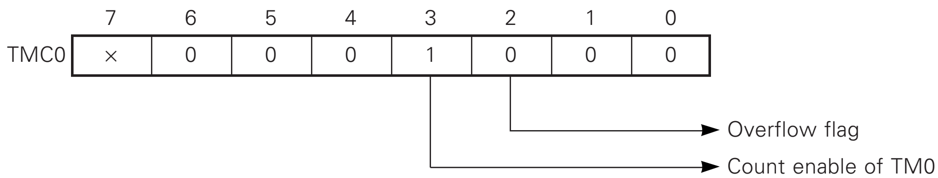
Fig. 7-32 Interval Timer Operation (2) Timing



Remarks: Interval time = (n + 1) x 8/f_{CLK}, where 0 ≦ n ≦ FFFFH

Fig. 7-33 Control Register Contents for Interval Timer Operation (2)

(a) Timer control register 0 (TMC0)



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

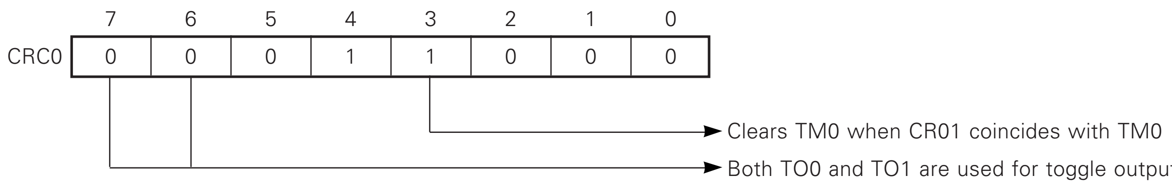
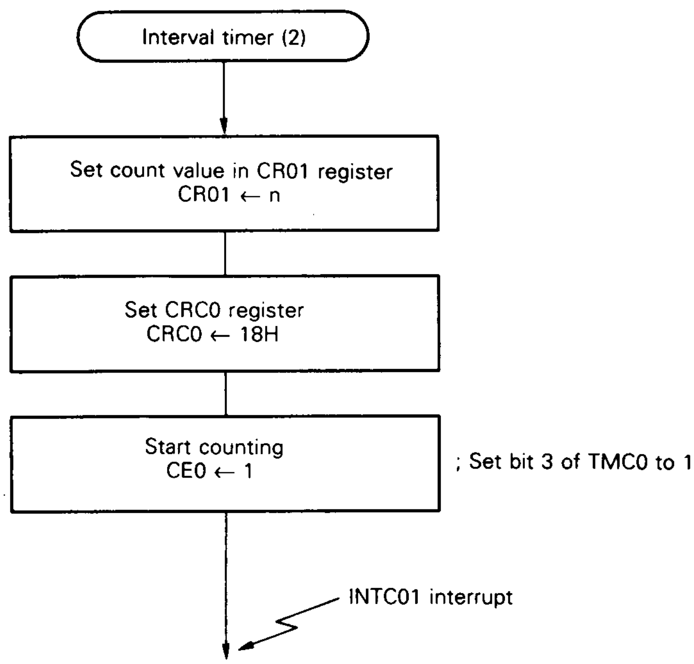


Fig. 7-34 Interval Timer Operation (2) Setting Procedure



(3) Operation to measure pulse width

The high-level or low-level width for an external pulse input to external interrupt request pin INTP3 can be measured by the 16-bit timer.

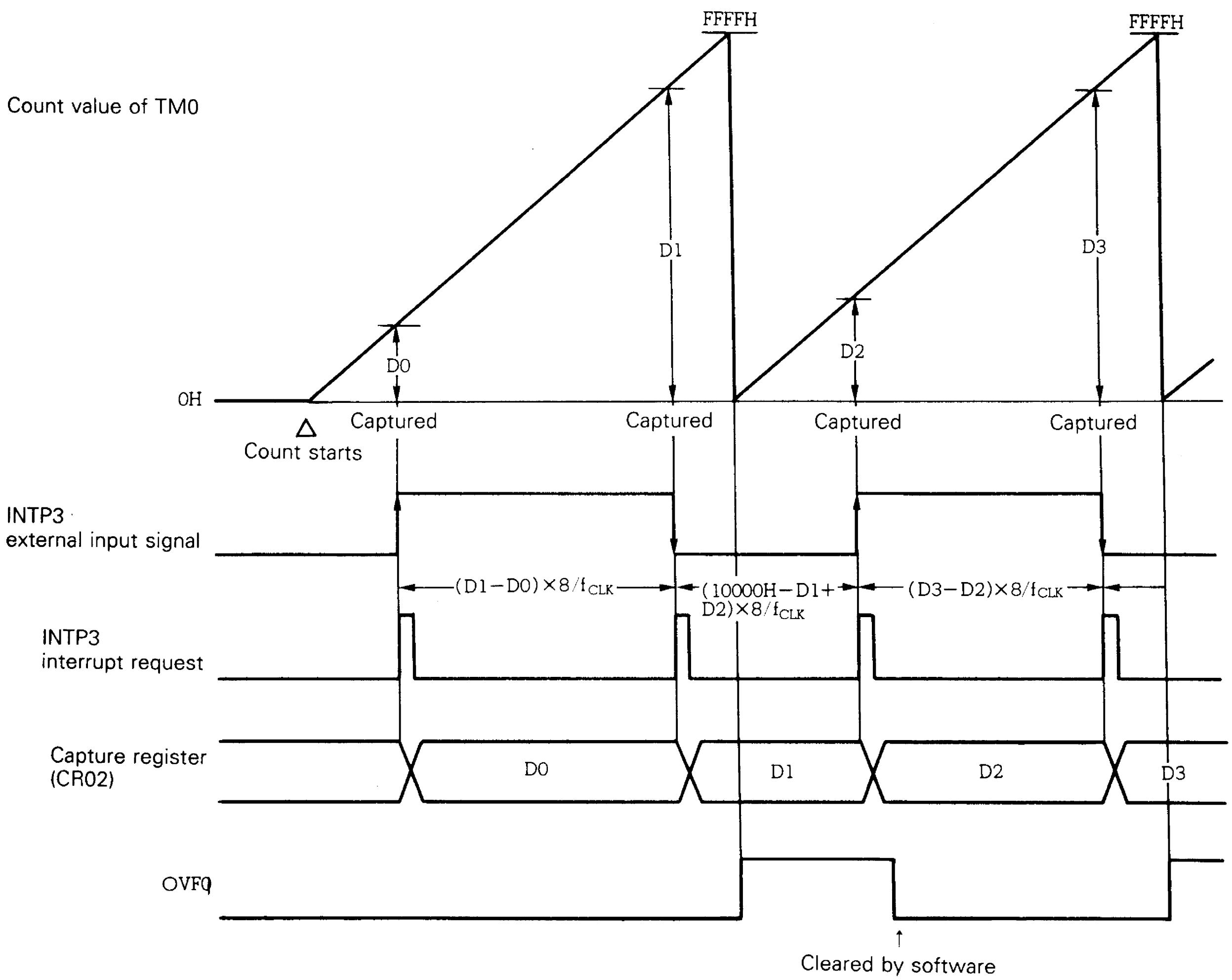
The width of the pulse input to the INTP3 pin must be at least 12 system clocks (2 μs at f_{CLK} = 6 MHz), regardless of whether the level of the pulse is high or low; otherwise, the valid edge of the pulse cannot be detected and captured.

With this pulse width measurement function, a pulse width, ranging from 2.6 μs to 87.4 ms, can be measured with a 1.3 μs resolution (at f_{CLK} = 6 MHz).

As shown in Fig. 7-35, the current count value for the 16-bit timer 0 (TM0) is captured to capture register CR02 in synchronization with the valid edge of the INTP3 pin (the valid edge is specified to be both rising and falling edges). The timer value is then retained in the capture register. To measure the pulse width, the difference between the count value for TM0 (Dn), which has been captured to and retained in the CR02 register when the nth valid edge has been detected, and the count value of TM0 (D_{n-1}) when the n-1th valid edge has been detected, is calculated. This difference is then multiplied by the count clock ($8/f_{CLK}$) to calculate the pulse width.

Figure 7-36 shows the control register contents, while Fig. 7-37 shows the procedure to set the control register contents.

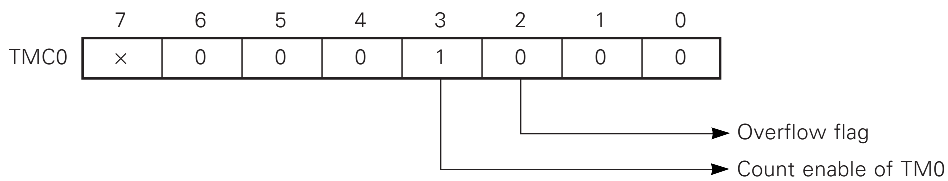
Fig. 7-35 Pulse Width Measurement Timing



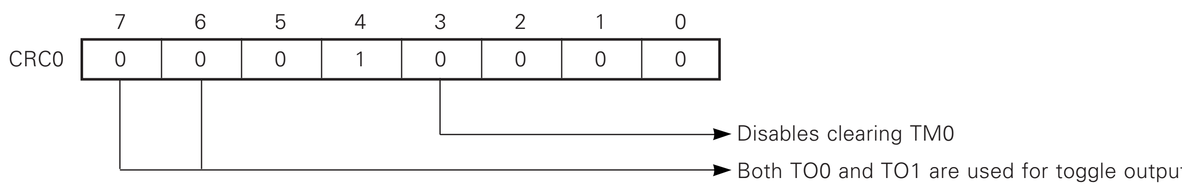
Remarks: Dn: Count value for TM0 (n = 0, 1, 2,...)

Fig. 7-36 Control Register Contents for Pulse Width Measurement

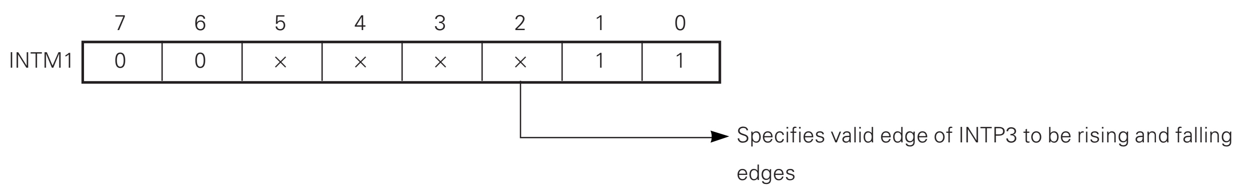
(a) Timer control register 0 (TMC0)



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



(c) External interrupt mode register 1 (INTM1)



×: don't care

Fig. 7-37 Pulse Width Measurement Setting Procedure

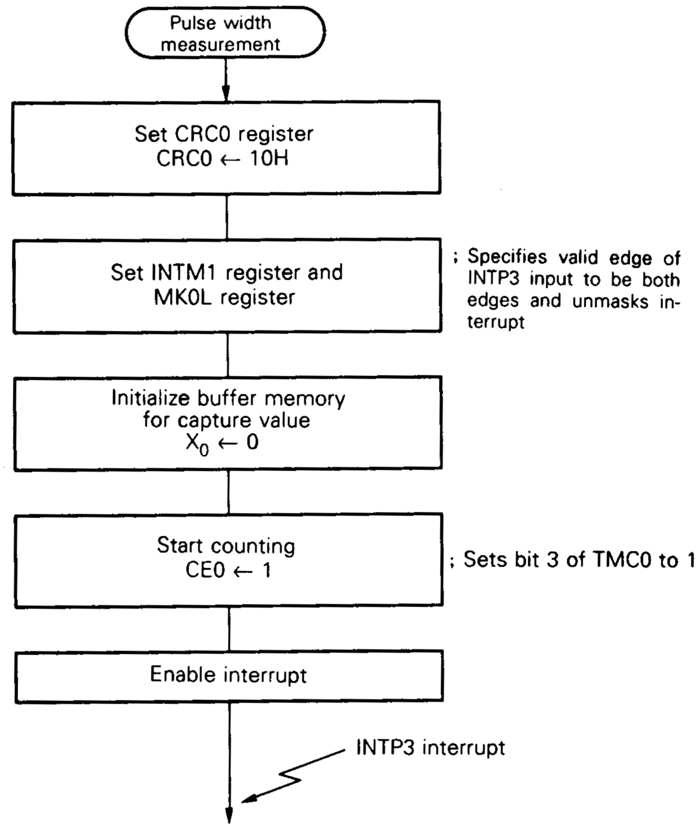
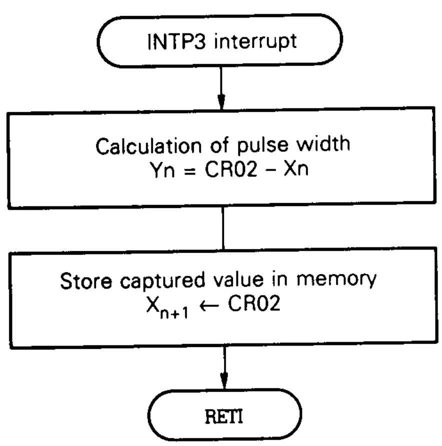


Fig. 7-38 Interrupt Request Processing to Calculate Pulse Width



(4) Operation as PWM output

The PWM output function is to output a pulse with a duty factor determined by the value set in the compare register (see **Fig. 7-39**).

The duty factor is 1/65536-65535/65536 and can be changed in 1/65536 units.

Since one 16-bit timer 0 (TM0) is provided with two compare registers, two types of PWM signals can be output.

Figure 7-40 shows the control register set contents, Fig. 7-41 shows the setting procedure, and Fig. 7-42 shows the procedure to change the duty factor.

Fig. 7-39 PWM Signal Output Example by 16-bit Timer/Counter

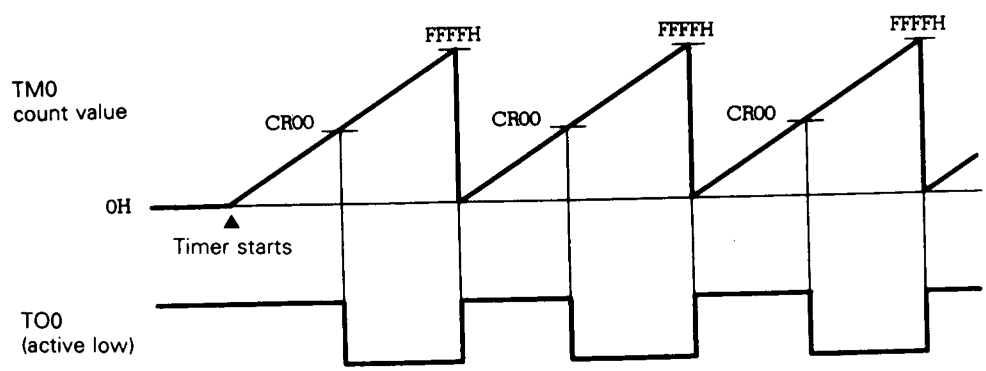
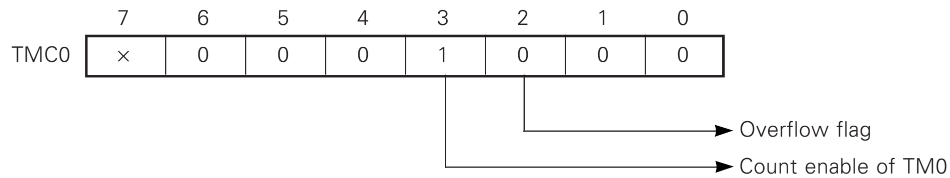
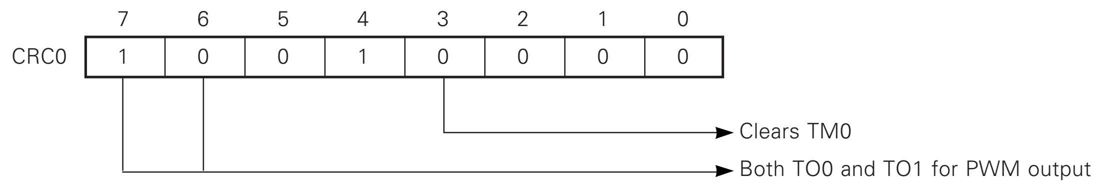


Fig. 7-40 Control Register Contents Set for PWM Output Operation

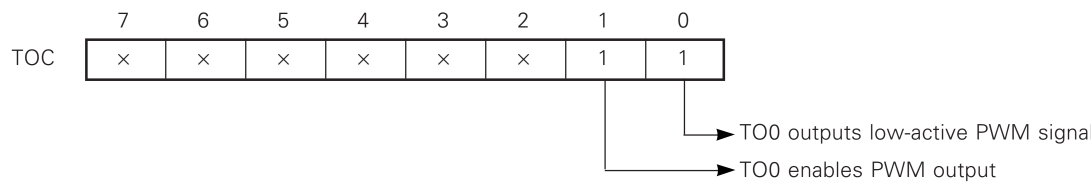
(a) Timer control register 0 (TMC0)



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



(c) Timer output control register (TOC)



(d) Port 3 mode control register (PMC3)

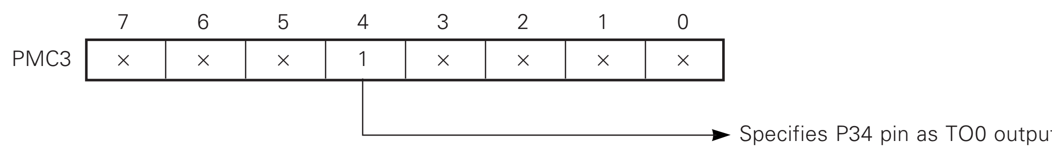


Fig. 7-41 PWM Output Setting Procedure

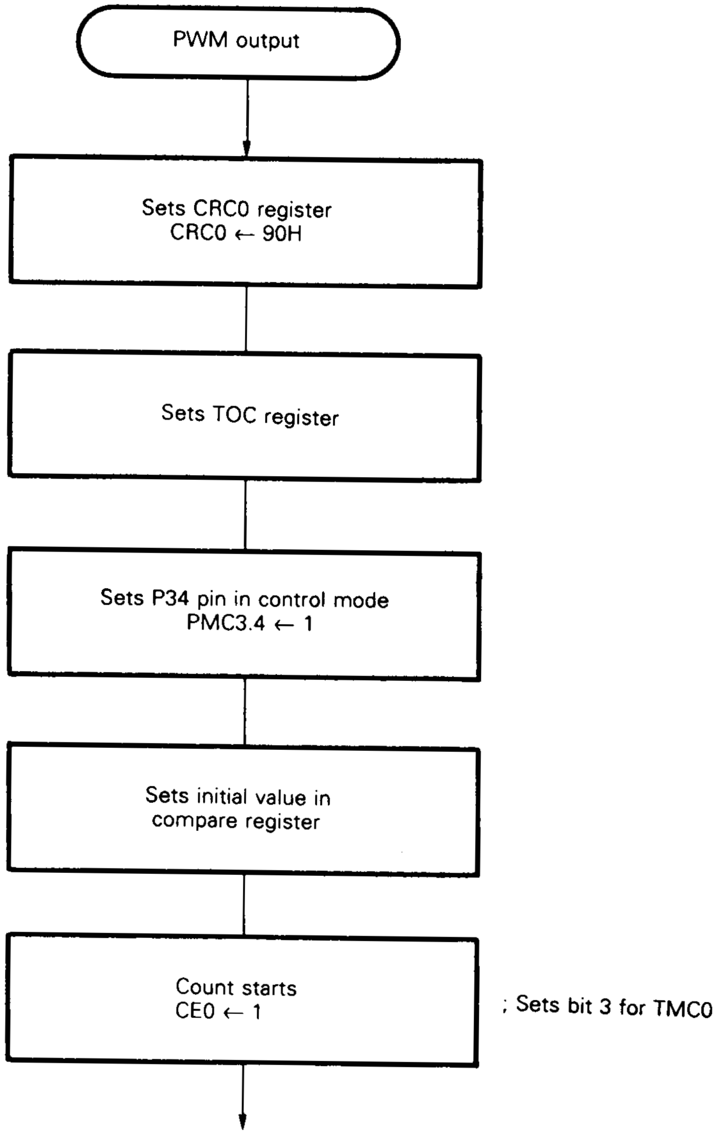
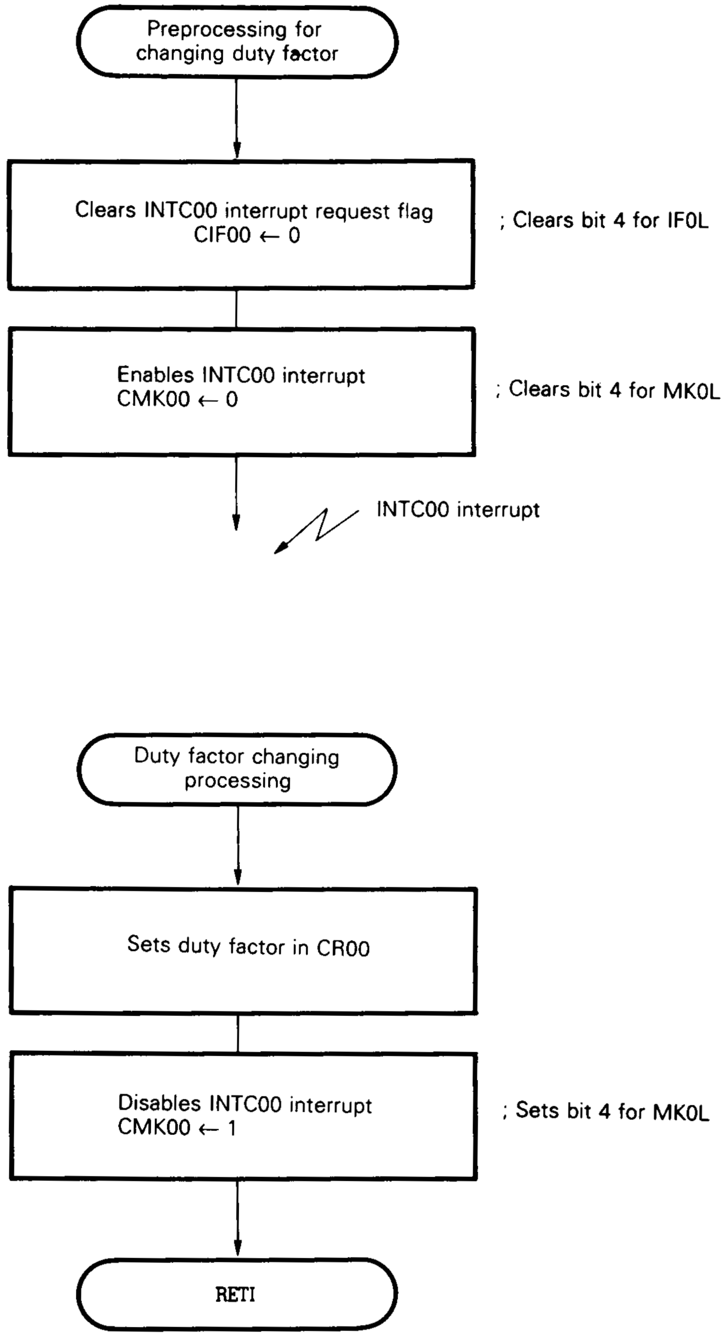


Fig. 7-42 Changing Duty Factor for PWM Output



(5) PPG output operation

The PPG output function is to output a pulse, whose period and duty factor are determined by a value set in the compare register (**Fig. 7-43**).

Figure 7-44 shows the control register set contents, Fig. 7-45 shows the setting procedure, and Fig. 7-46 shows the procedure used to change the duty factor.

Fig. 7-43 16-bit Timer/Counter PPG Signal Output Example

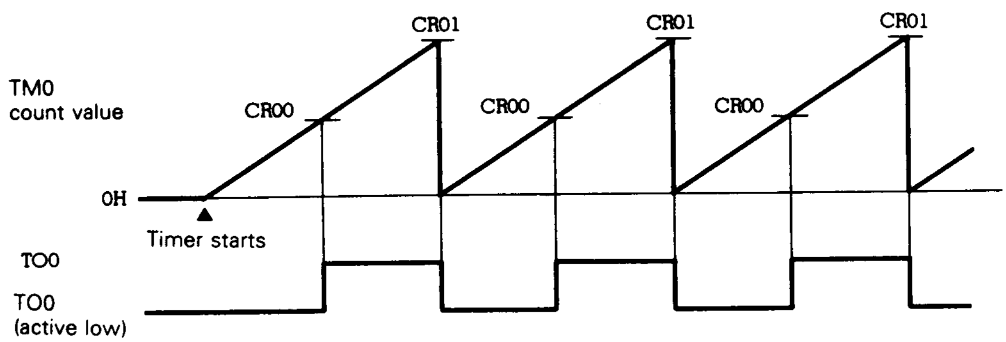
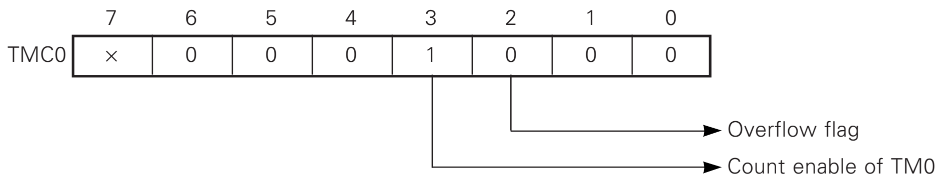
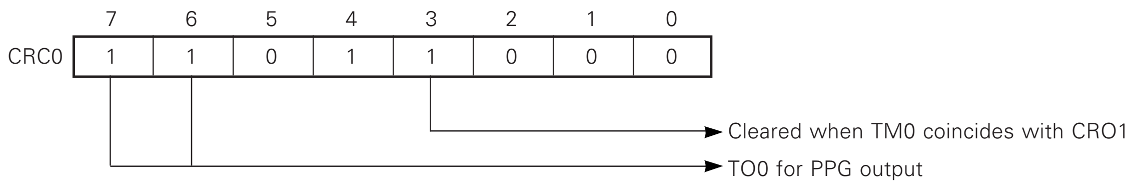


Fig. 7-44 Control Register Contents Set for PPG Output Operation

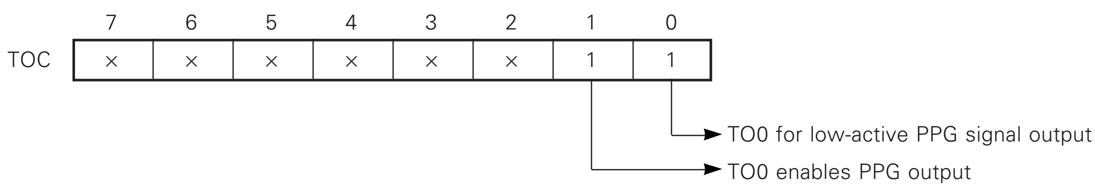
(a) Timer control register 0 (TMC0)



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



(c) Timer output control register (TOC)



(d) Port 3 mode control register (PMC3)

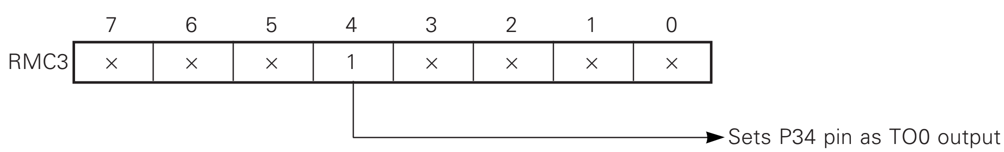


Fig. 7-45 PPG Output Setting Procedure

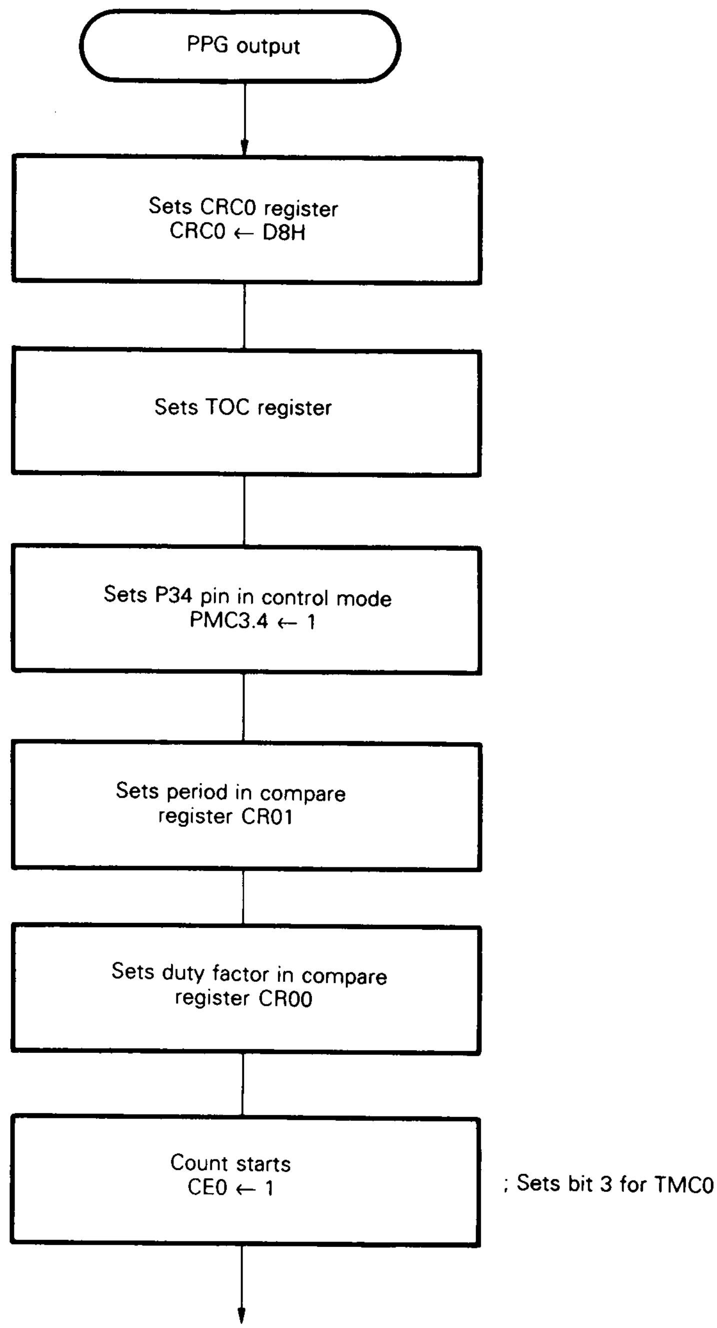
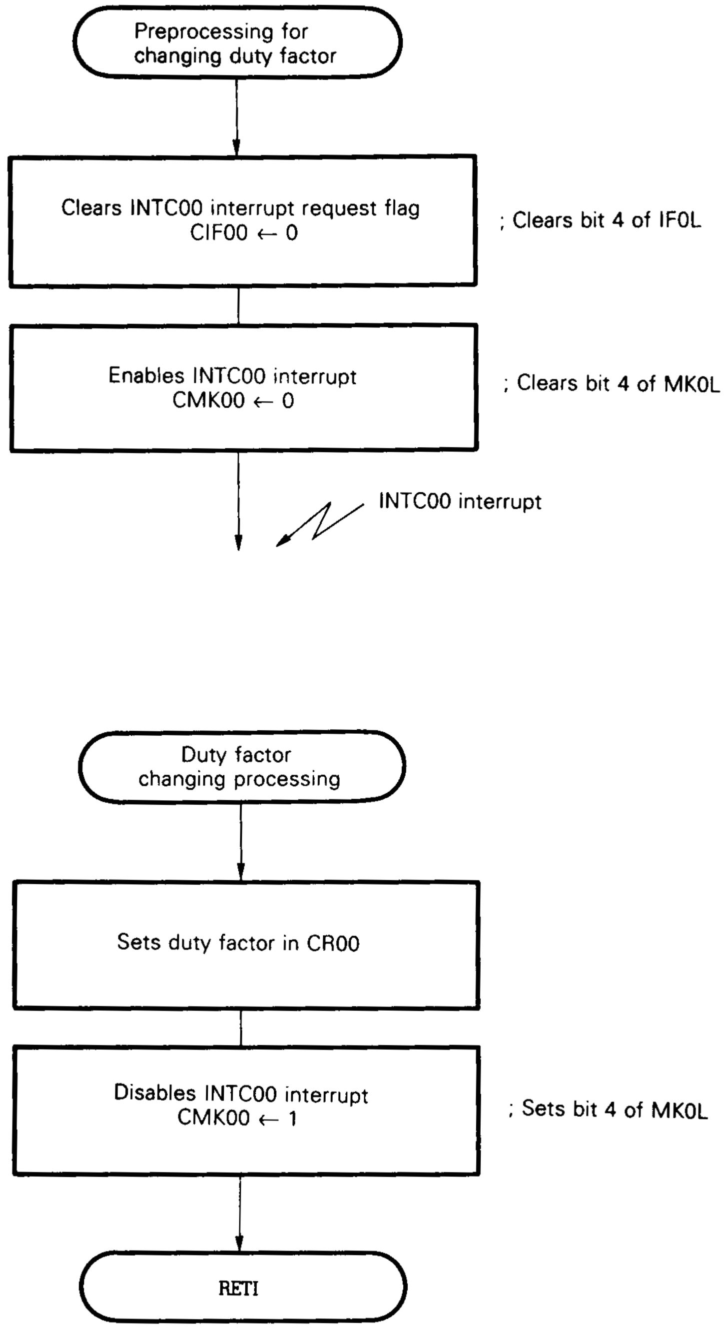


Fig. 7-46 Changing Duty Factor of PPG Output



(6) Software-triggered one-shot pulse output example

The software-triggered one-shot pulse output mode is to output a one-shot pulse triggered by software (**Fig. 7-47**).

Figure 7-48 shows the control registers set contents, and Fig. 7-49 shows the setting procedure.

Fig. 7-47 16-bit Timer/Counter One-Shot Pulse Output

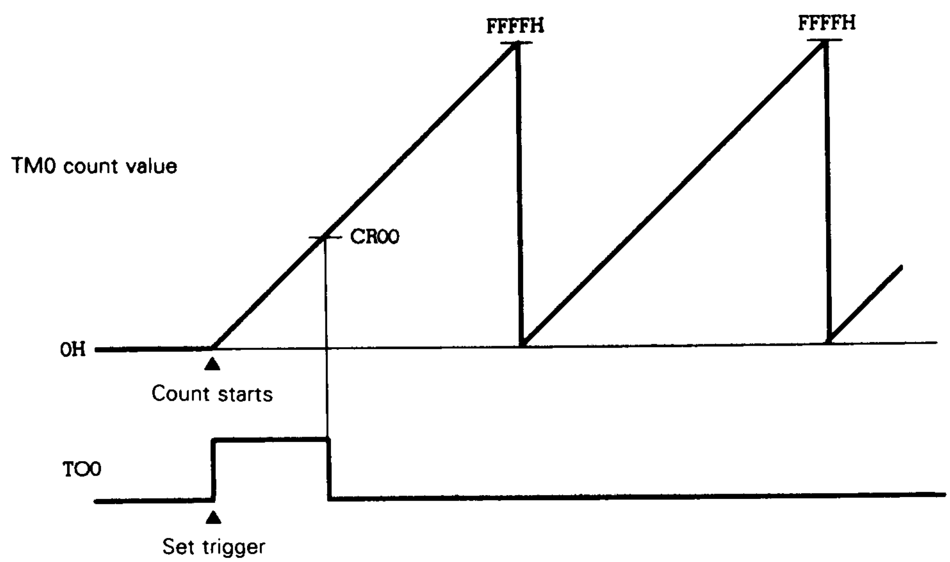
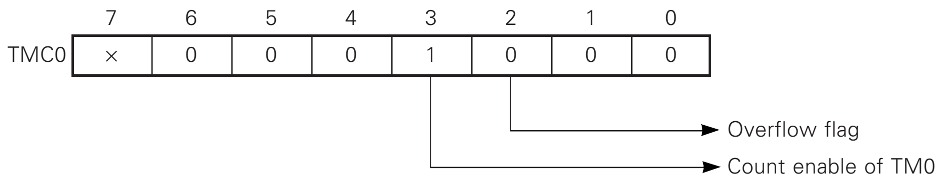
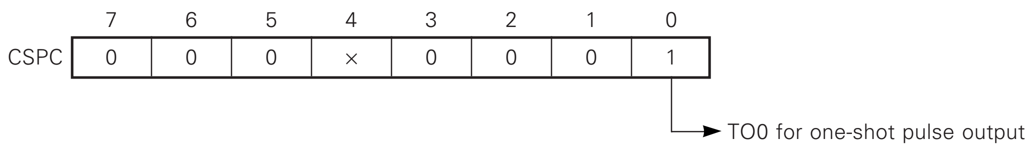


Fig. 7-48 Control Register Contents Set for One-Shot Pulse Output

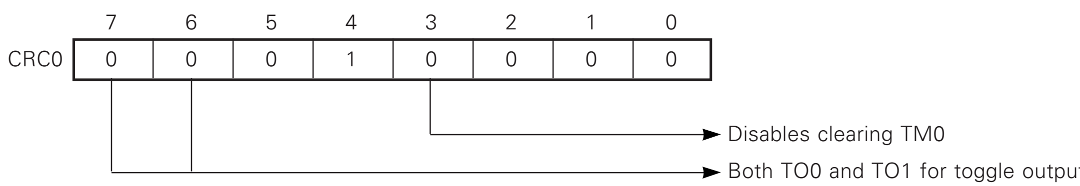
(a) Timer control register 0 (TMC0)



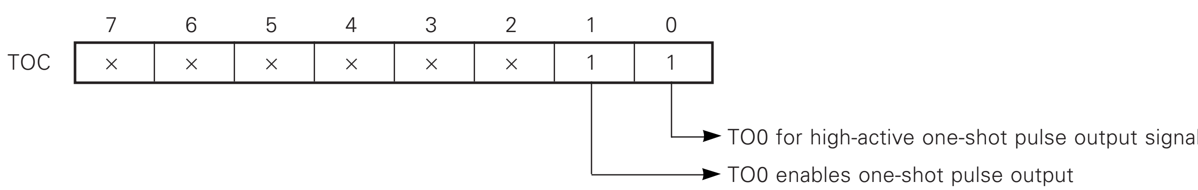
(b) One-shot pulse output control register (OSPC)



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



(d) Timer output control register (TOC)



(e) Port 3 mode control register (PMC3)

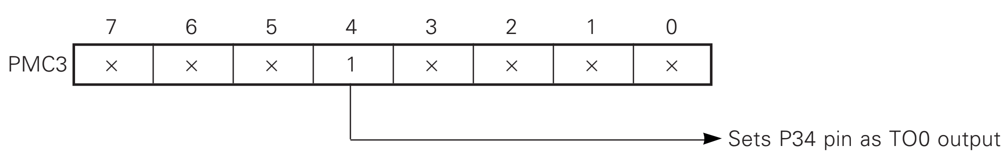
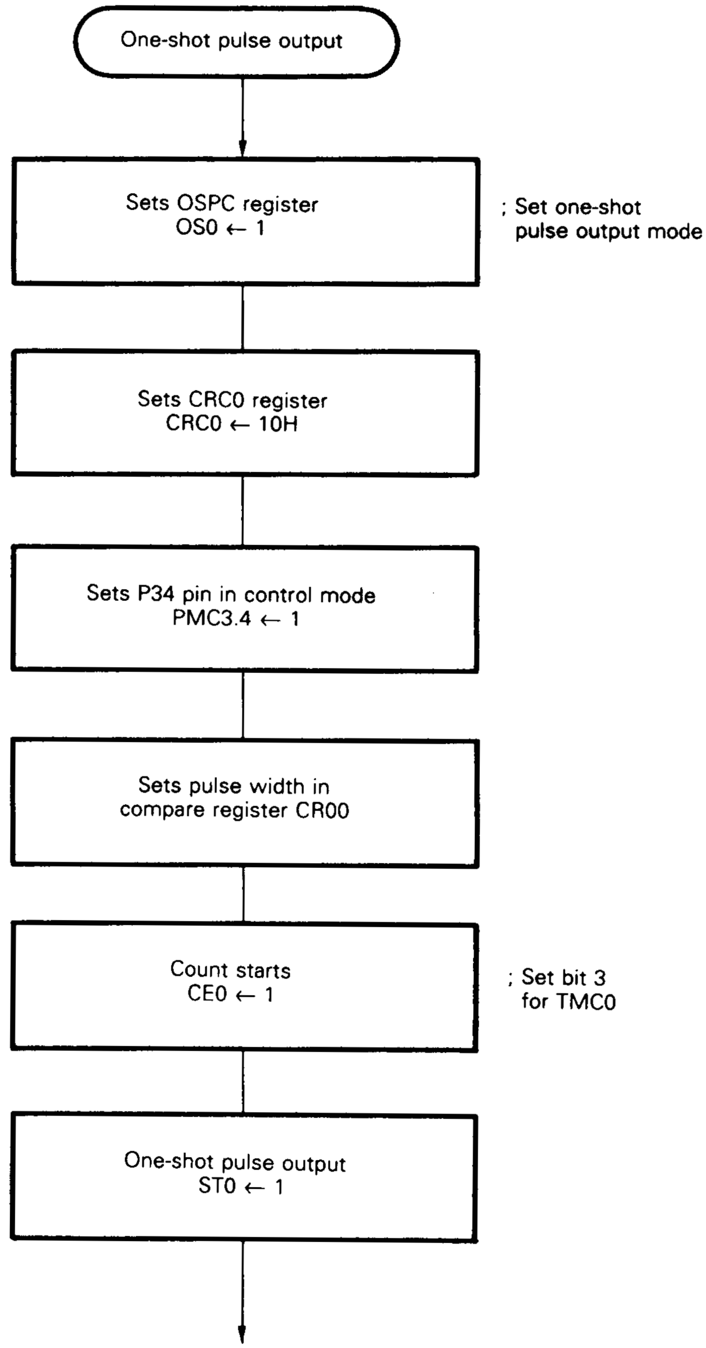


Fig. 7-49 One-Shot Pulse Output Setting Procedure



7.2 8-bit Timer/Counter 1

7.2.1 Function

The 8-bit timer/counter 1 can be used as an interval timer and to measure pulse widths. In addition, it can also be used to generate the output trigger for the real-time output port.

(1) Interval timer

When 8-bit timer/counter 1 is used as an interval timer, an internal interrupt occurs at time intervals specified by the interval timer.

Table 7-9 Time Interval for 8-bit Timer/Counter 1

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

Figures in ( ) are at f_{CLK} = 6 MHz.

(2) Pulse width measurement

The pulse width for a signal input to external interrupt pin INTP0 can be measured by 8-bit timer/counter 1.

Table 7-10 Pulse Width Measurement Range

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

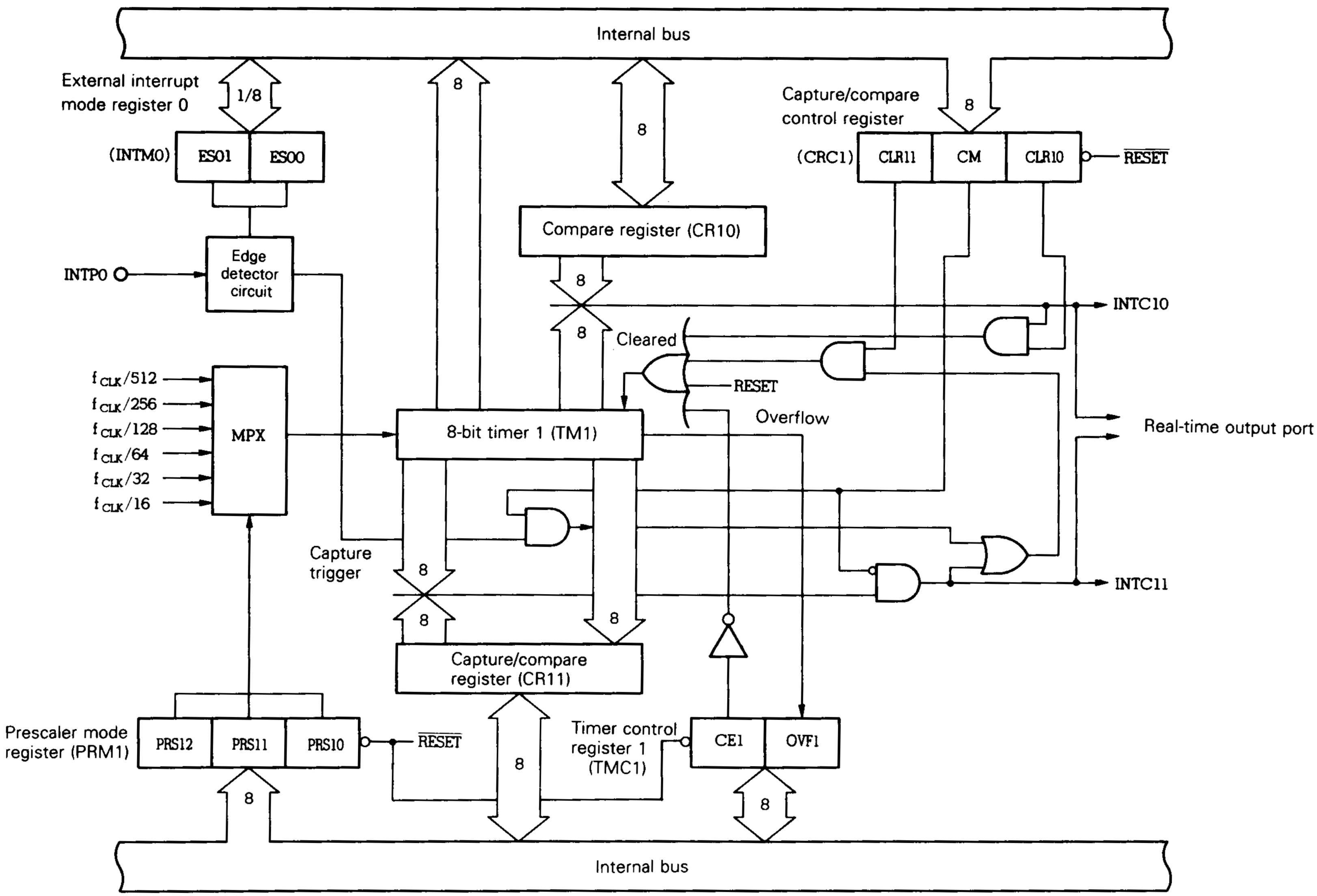
Figures in ( ) are at f_{CLK} = 6 MHz.

7.2.2 Configuration

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

Figure 7-50 shows the 8-bit timer/counter 1 configuration.

Fig. 7-50 Configuration of 8-bit Timer/Counter



(1) 8-bit timer 1 (TM1)

Timer TM1 counts up the count clock specified by the lower 4 bits of the prescaler mode register 1 (PRM1). The operation of this timer can be disabled or enabled by timer control register 1 (TMC1).
The timer contents can only be read by an 8-bit manipulation instruction. When the $\overline{RESET}$ signal is input, the TM1 contents are cleared to 00H and TM1 stops counting.

(2) Compare register (CR10)

CR10 is an 8-bit register holding a value that determines the interval timer cycle.
When the contents of this register coincide with the TM1 value, an interrupt request (INTC10) is generated. This coincidence signal can also be used as a trigger signal for the real-time output port. The TM0 count value can also be cleared, when coincidence takes place.
Data can be read from or written to this register by an 8-bit manipulation instruction. When the $\overline{RESET}$ signal is input, the contents of this register become undefined.

(3) Capture/compare register (CR11)

This 8-bit register can be used as a compare register that detects coincidence between its value and the TM1 count value, when so specified by the capture/compare control register 1 (CRC1), or as a capture register that captures the TM1 count value.

(a) As compare register

When the CR11 register is used as a capture register, it functions as an 8-bit register holding a value that determines the interval timer cycle.

When the TM1 count value coincides with the contents of the CR11 register, an interrupt request (INTC11) is generated.
The TM1 count value can also be cleared when the coincidence takes place. This coincidence signal can also be used as the trigger signal for the real-time output port.

(b) As capture register

When the CR11 register is used as a capture register, it captures the TM1 contents in synchronization with the valid edge (capture trigger) input to external interrupt input pins INTP0.
The CR11 register contents are retained until the next capture trigger is generated. The TM1 contents can be cleared, after they have been captured to the CR11 register.
Data can be read from or written to the capture register by an 8-bit manipulation instruction. When the $\overline{RESET}$ signal is input, the register contents become undefined.

(4) Edge detector circuit

This circuit detects the valid edge for a signal input from an external source. It detects the valid edge, specified by external interrupt mode register 0 (INTM0), inputs it to the INTP0 input pin and generates interrupt INTP0 and a capture trigger (for INTM0 register details, refer to **Fig. 13-1**).

(5) Prescaler

The prescaler generates the count clock from the internal system clock. The clock generated by the prescaler is selected by the selector as the count clock, based on which the timer performs its counting operation.

7.2.3 8-bit timer/counter control registers

(1) Timer control register 1 (TMC1)

The TMC1 register is an 8-bit register that controls the counting operation for 8-bit timers 1 and 2 (TM1 and TM2). The lower 4 bits of this register are used to control the counting operation for 8-bit timer/counter 1 (TM1), while the higher 4 bits are used to control the counting operation for 8-bit timer/counter 2 (TM2). Data can be read from or written to this register by an 8-bit manipulation instruction. Figure 7-51 shows the format for this register.

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

Fig. 7-51 Timer Control Register 1 (TMC1) Format

	7	6	5	4	3	2	1	0
TMC1	CE2	OVF2	CMD2	0	CE1	OVF1	0	0

OVF1	TM1 overflow flag
0	Overflow does not occur
1	Overflow occurs (counting up from FFH to 00H)

CE1	Controls TM1 counting operation
0	Clears and stops counting operation
1	Enables counting operation

Bits 7 to 4: These bits control counting operation for 8-bit timer/counter 2 (see **Fig. 7-74**).

Remarks: The OVF1 bit can only be cleared by software.

(2) Prescaler mode register 1 (PRM1)

This 8-bit register specifies the count clock supplied to 8-bit timers 1 and 2 (TM1 and TM2).

The lower 4 bits for this register are used to specify the count clock supplied to 8-bit timer/counter 1 (TM1), while the higher 4 bits are used to specify the count clock supplied to 8-bit timer/counter 2 (TM2).

Data can only be written to this register by an 8-bit manipulation instruction. Figure 7-52 shows the format for this register.

When the $\overline{\text{RESET}}$ signal is input, the PRM1 contents are cleared to 00H.

Fig. 7-52 Prescaler Mode Register 1 (PRM1) Format

	7	6	5	4	3	2	1	0
PRM1	PRS23	PRS22	PRS21	PRS20	0	PRS12	PRS11	PRS10

f_{CLK}=6MHz

PRS12	PRS11	PRS10	Specifies count clock (Hz)	Resolution
0	0	0	$f_{CLK}/16$	2.6 μs
0	0	1		
0	1	0		
0	1	1	$f_{CLK}/32$	5.3 μs
1	0	0	$f_{CLK}/64$	10.7 μs
1	0	1	$f_{CLK}/128$	21.3 μs
1	1	0	$f_{CLK}/256$	42.7 μs
1	1	1	$f_{CLK}/512$	85.3 μs

PRS23–PRS20: These bits specify the count clock supplied to 8-bit timer/counter 2 (TM2) (see Fig. 7-75).

Remarks: f_{CLK} : system clock frequency

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

Register CRC1 is an 8-bit register that specifies the operation of the capture/compare register (CR11) and a condition under which the 8-bit timer 1 (TM1) contents are cleared.

Data can only be written to this register by an 8-bit manipulation instruction. The format for this register is shown in Fig. 7-53.

When the $\overline{\text{RESET}}$ signal is input, the contents of this register are cleared to 00H.

Fig. 7-53 Capture/Compare Control Register 1 (CRC1) Format

	7	6	5	4	3	2	1	0
CRC1	0	0	0	0	CLR11	CM	CLR10	0

CLR10	Clearing TM1 when TM1 coincides with CR10
0	Disabled
1	Enabled

CLR11	CM	Specifies CR11 register operation	Clearing TM1
0	0	Compare register	Disabled
1	0		Enabled (when TM1 coincides with CR11)
0	1	Capture register	Disabled
1	1		Enabled (CR11 captures M1)

7.2.4 8-bit timer 1 (TM1) operation

(1) Basic operation

The 8-bit timer/counter 1 counts up the count clocks specified by the lower 4 bits for the prescaler mode register 1 (PRM1).

When the $\overline{\text{RESET}}$ signal is input, the TM1 contents are cleared to 00H and TM1 stops counting.

The TM1 counting operation is enabled or disabled by bit 3 (CE1) of timer control register 1 (TMC1) (the 8-bit timer/counter 1 operation is controlled by the lower 4 bits for the TMC1 register). When the CE1 bit is set to 1 by software, the TM1 contents are cleared to 00H at the first count clock, and then TM1 starts the count-up operation.

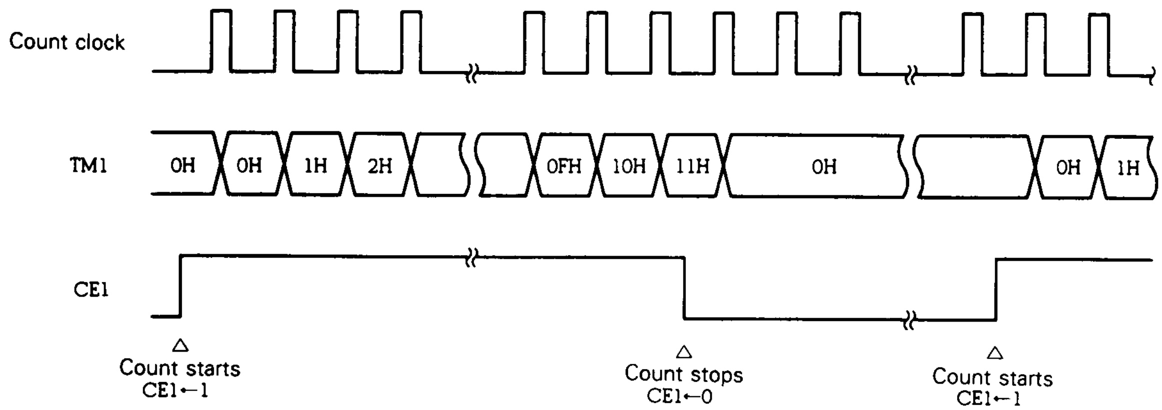
When the CE1 bit is reset to 0, the TM1 contents are cleared to 00H at the next count clock, and its capture operation and generation for the coincidence signal are stopped.

If an attempt is made to set the CE1 bit, which has already been set, to 1, TM1 is not cleared but continues its operation.

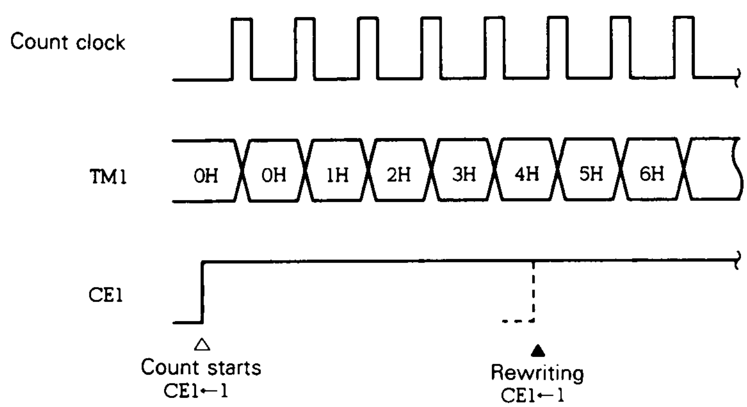
When a count clock is input while the TM1 current value is FFH, the timer is cleared to 00H. At this time, the OVF1 bit is set to 1. The OVF1 bit can only be cleared by software. At this time, TM1 continues the counting operation.

Fig. 7-54 Basic Operation for 8-bit Timer 1 (TM1)

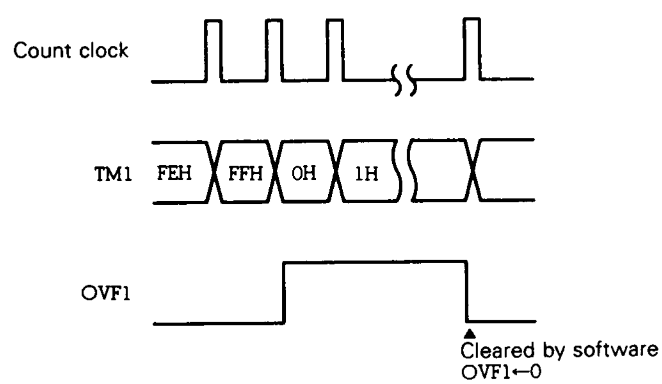
(a) When counting is started, then stopped, and then started again



(b) When "1" is written to CE1 bit again after counting is started



(c) TM1 operation when its current value is FFH



(2) Clearing operation

The 8-bit tiemr 1 (TM1) can be automatically cleared after its value has coincided with the contents of the compare register (CR1m: m = 0, 1) or captured. When the reason for clearing TM1 has occurred, TM1 is cleared to 00H at the next count clock. Therefore, even if the clearing cause has occurred, the current TM1 value is retained, until the next count clock is applied.

Fig. 7-55 Clearing TM1 by Coincidence with Compare Register (CR1m)

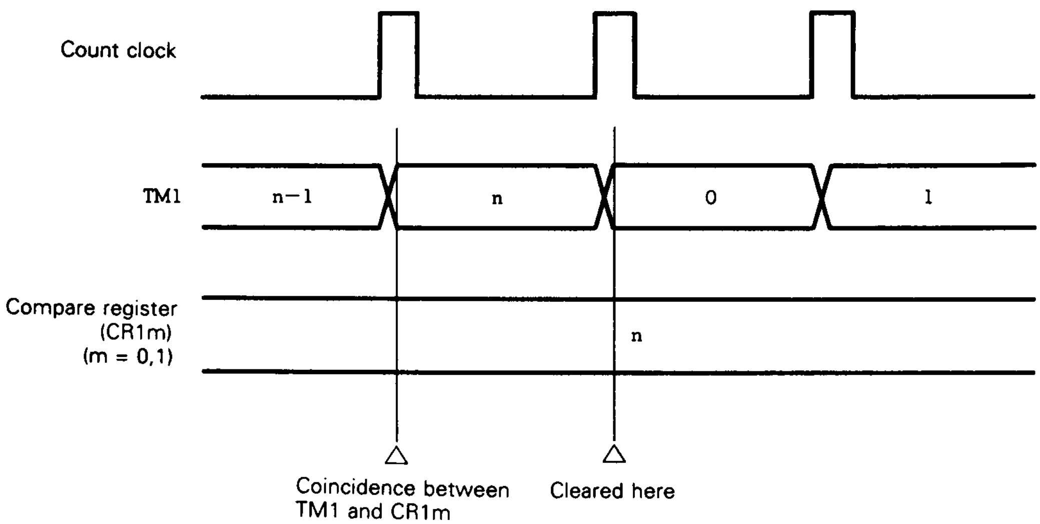
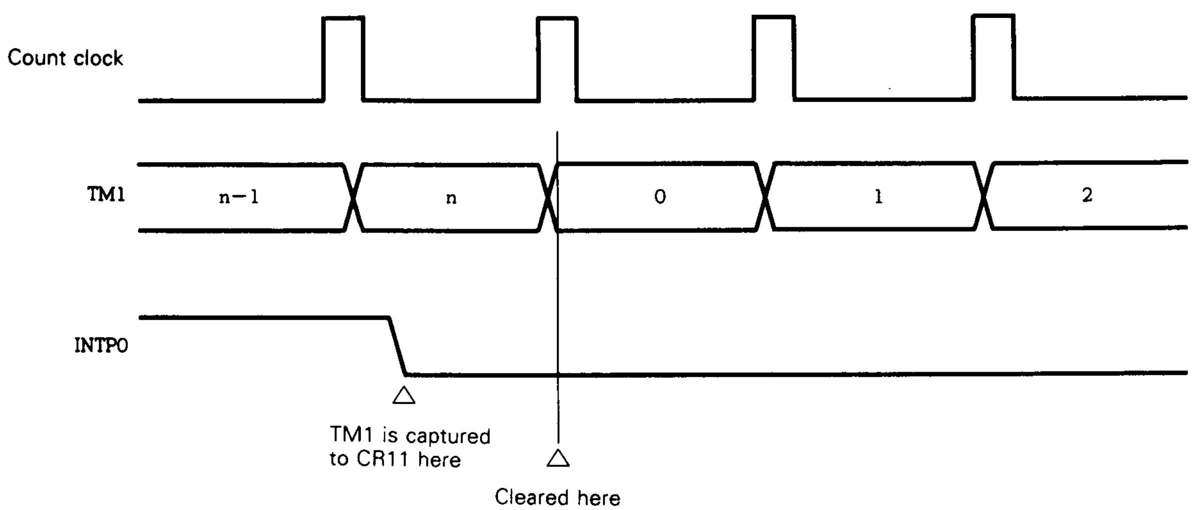


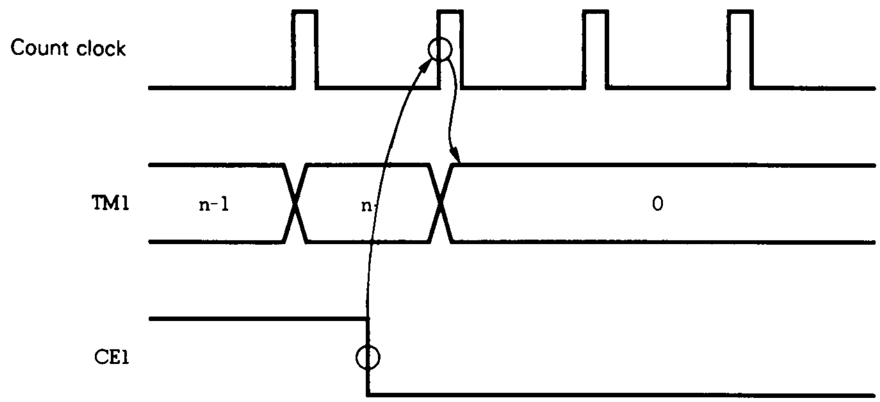
Fig. 7-56 Clearing TM1 after Capturing



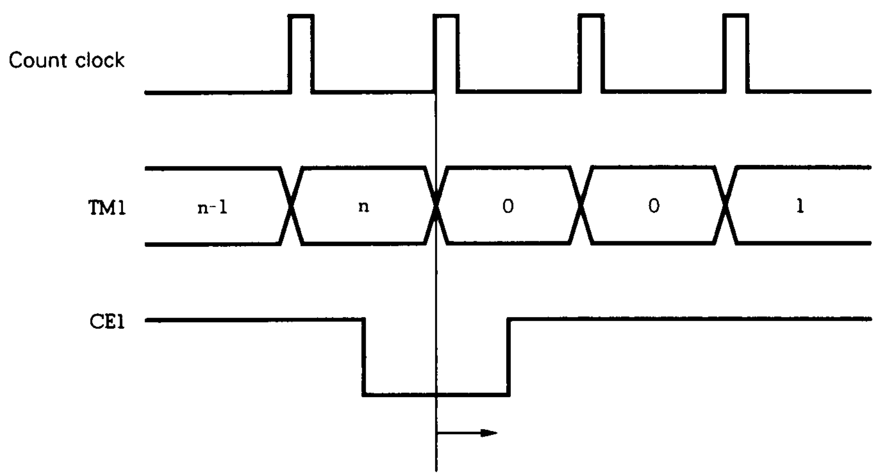
TM1 is also cleared by resetting the CE1 bit of the timer control register (TMC1) to 0 through software. The clear operation is also performed by the count clock after the CE1 bit has been reset to 0. If the CE1 bit is set to 1 after the CE1 bit has been reset to 0 and before TM1 is cleared to 0 (before the first count clock is input after the CE1 bit has been reset to 0), TM1 is cleared to 0 and at the same time, 0 counting operation is performed because counting has been started.

Fig. 7-57 Clear Operation When CE1 Bit Is Reset to 0

(a) Basic operation

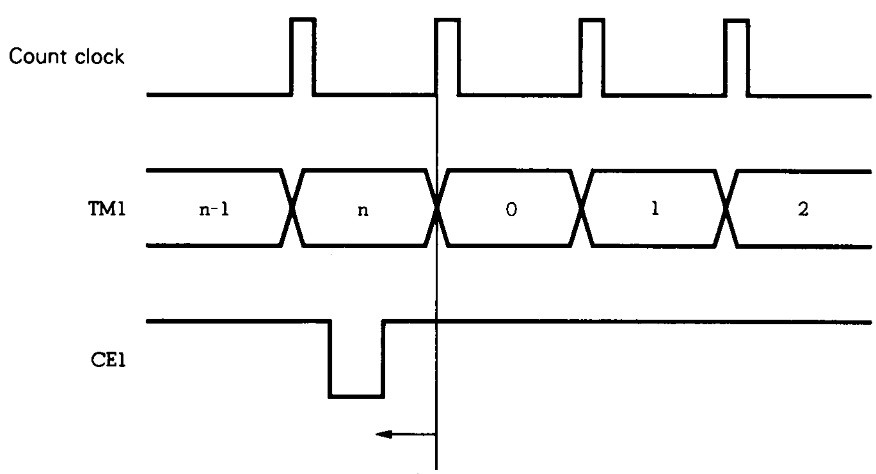


(b) Restart after TM1 has been cleared to 0



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

(c) Restart before TM1 is cleared to 0



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

7.2.5 Compare register and capture/compare register operations

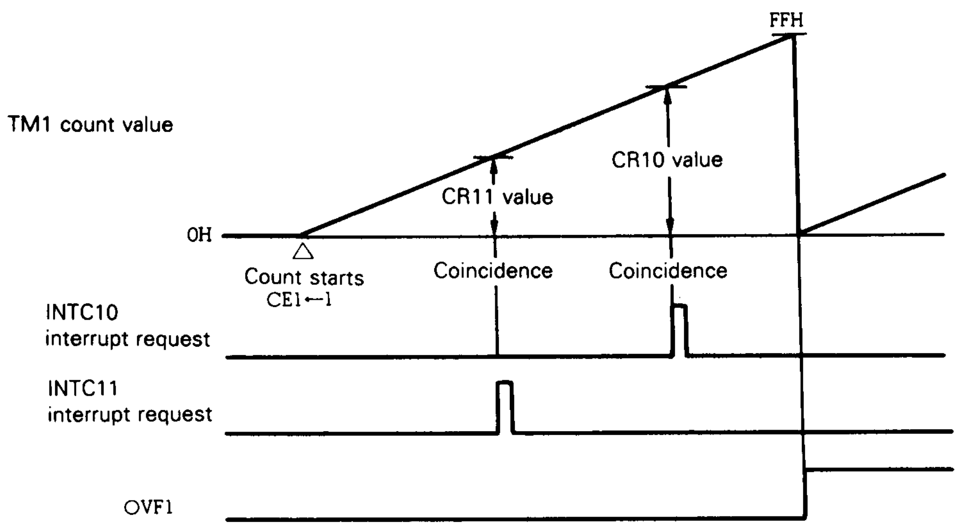
(1) Compare operation

The 8-bit timer/counter 1 can also perform a compare operation, which compares the current count value for the timer with the set value for the compare register.

When the count value for the 8-bit timer 1 (TM1) coincides with the value set in advance in the compare register (CR10) or in the compare/capture register (CR11) set in the compare register mode, an interrupt request signal (INTC10 from the CR10 register or INTC11 from the CR11 register) is generated.

After the timer value has coincided with the compare register value, the TM1 count value can be cleared. In this case, the timer functions as an interval timer that repeatedly counts up to the value set in the CR10 or CR11 register.

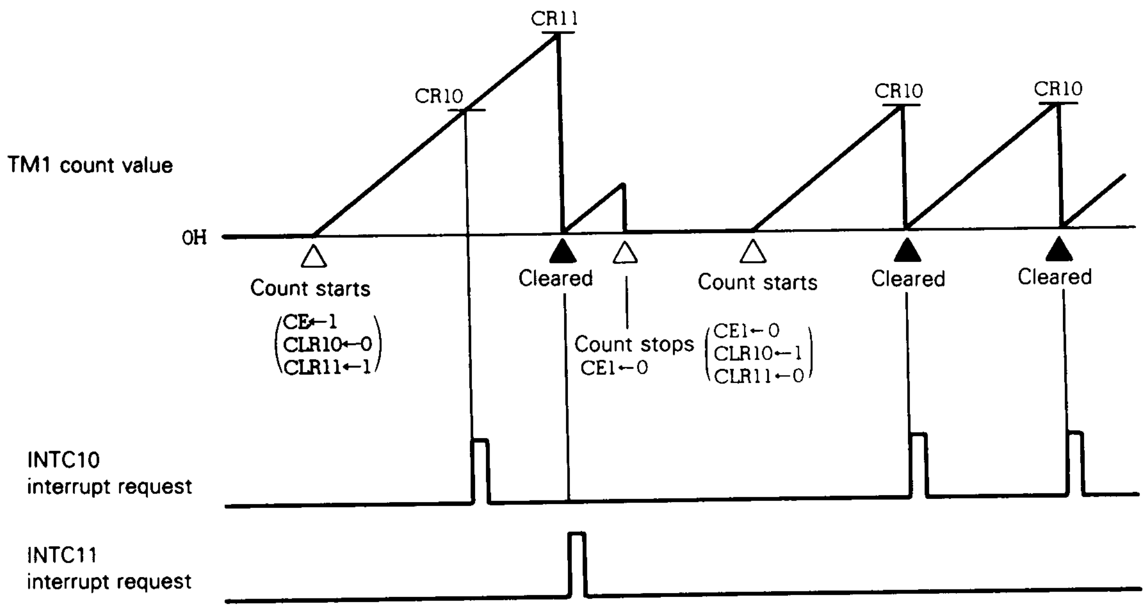
Fig. 7-58 Compare Operation



Remarks: CLR10 = 0, CLR11 = 0, CM = 0

Caution: Refer to 7.5.4 Notes for using in-circuit emulator.

Fig. 7-59 Clearing TM1 after Coincidence Detection



(2) Capture operation

The 8-bit timer/counter 1 can also perform a capture operation by which to capture the count value for the timer to the capture register, which then retains the value, in synchronization with an external trigger.

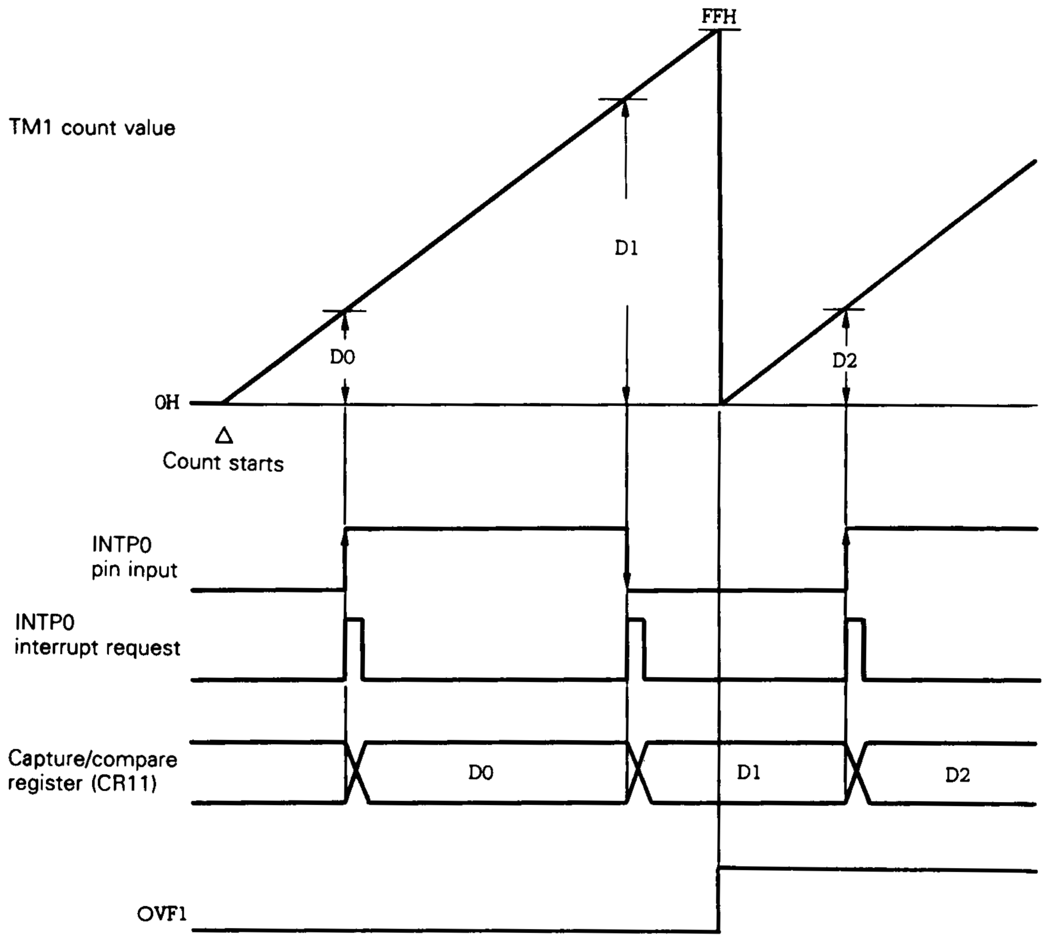
As the external trigger, the valid edge detected on external interrupt request input pin INTP0 is used (capture trigger). The count value for the 8-bit timer 1 (TM1) is captured to the capture/compare register (CR11) in synchronization with the capture trigger. The CR11 register contents are retained until the next capture trigger is generated.

The valid edge of the capture trigger is specified by external interrupt mode register 0 (INTM0). If the capture trigger is specified, so that it is detected at both the rising and falling edges, the width of a pulse input from an external source can be measured. If the capture trigger is detected at either edge, the input pulse cycle can be measured.

For detailed INTM0 register format, refer to **Fig. 13-1** in **Chapter 13 Edge Detection Function**.

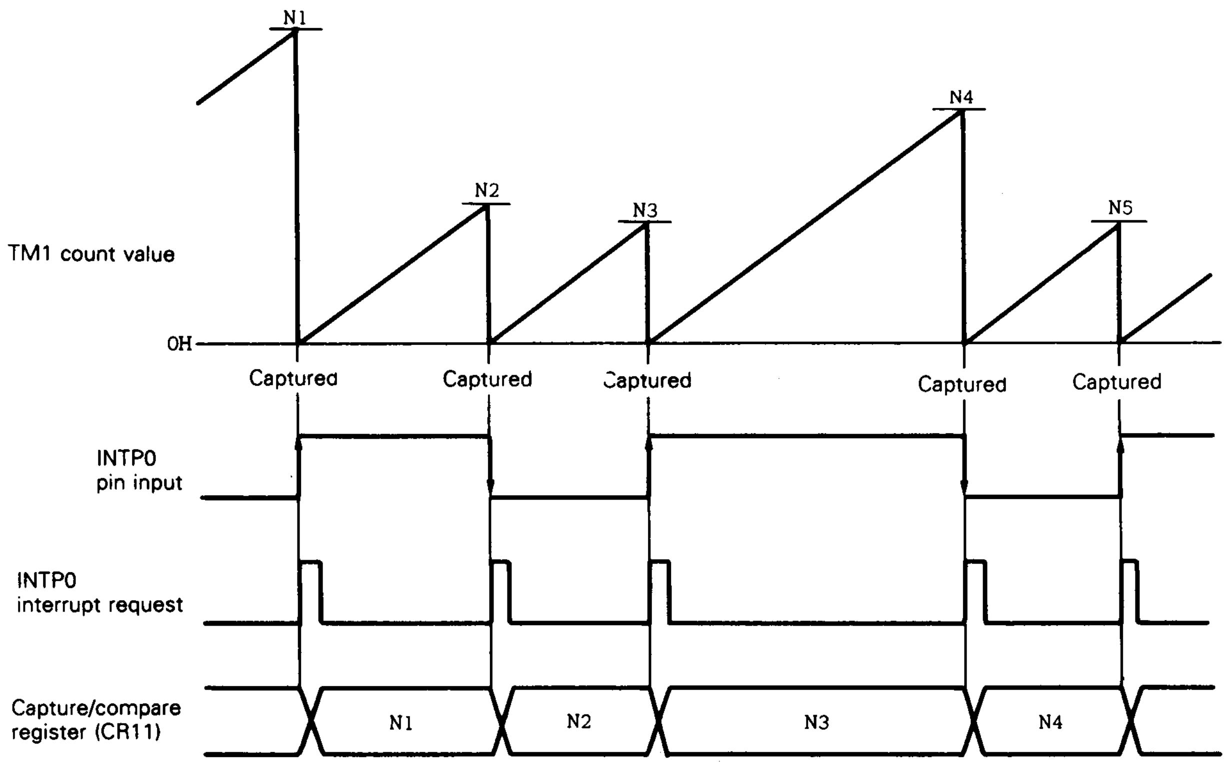
Caution: Refer to 7.5.4 Notes for using in-circuit emulator.

Fig. 7-60 Capture Operation



Remarks: Dn: count value for TM1 (n = 0, 1, 2, ...)
CLR10 = 0, CLR11 = 0, CM = 1

Fig. 7-61 Clearing TM1 after Its Value Has Been Captured



Remarks: NI: Count value of TM1 (n = 0, 1, 2, ...)
CLR10 = 0, CLR11 = 1, CM = 1

7.2.6 Application examples

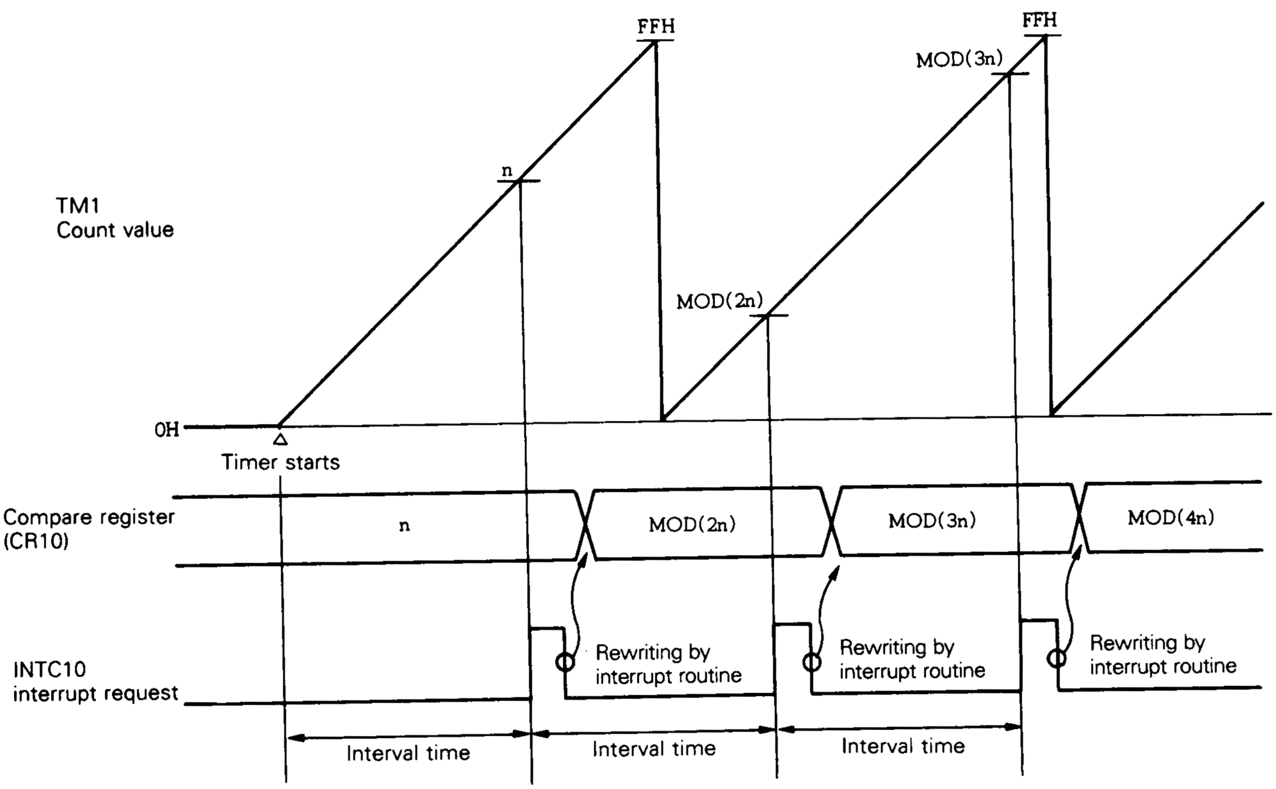
(1) Operation as interval timer (1)

When the 8-bit timer 1 (TM1) is used as a free-running timer and when a fixed value is added to the contents of compare registers (CR10 and CR11) by an interrupt routine, the timer operates as an interval timer, whose cycle is determined by the value to be added to the compare register contents (see **Fig. 7-62**).

Since the 8-bit timer 1 is provided with two compare registers, the timer can operate as an interval timer having two cycles.

Fig. 7-63 shows the control register contents. Fig. 7-64 shows the procedure used to set the control register contents. Fig. 7-65 shows the processing performed by the interrupt routine.

Fig. 7-62 Interval Timer Operation (1) Timing



Remarks: Interval time = n x x/f_{CLK}, where 1 ≦ n ≦ FFH
x = 16, 32, 64, 128, 256, 512

Fig. 7-63 Control Register Contents for Interval Timer Operation (1)

(a) Timer control register 1 (TMC1)

	7	6	5	4	3	2	1	0
TMC1	×	×	×	0	1	0	0	0

- Bit 2 → Overflow flag
- Bit 3 → Count enable of TM1

(b) Prescaler mode register 1 (PRM1)

	7	6	5	4	3	2	1	0
PRM1	×	×	×	×	0	PRS21	PRS11	PRS10

- Bits 2–0 → Specifies count clock (x/f_{CLK}; where x = 16, 32, 64, 128, 256, 512)

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

	7	6	5	4	3	2	1	0
CRC1	0	0	0	0	0	0	0	0

- Bit 1 → Disables clearing TM1, when CR10 coincides with TM1
- Bit 2 → Specifies CR11 register as compare register
- Bit 3 → Disables clearing TM1, when CR11 coincides with TM1

Fig. 7-64 Interval Timer Operation (1) Setting Procedure

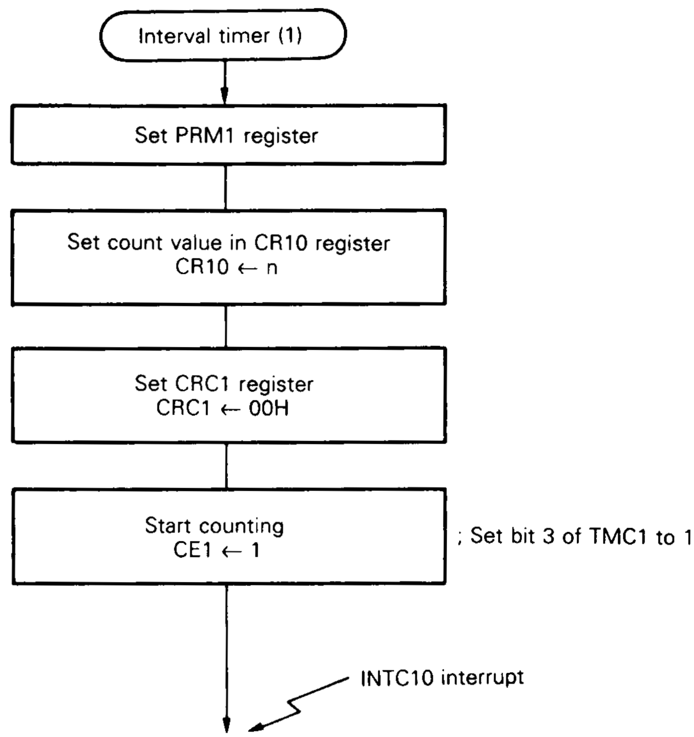
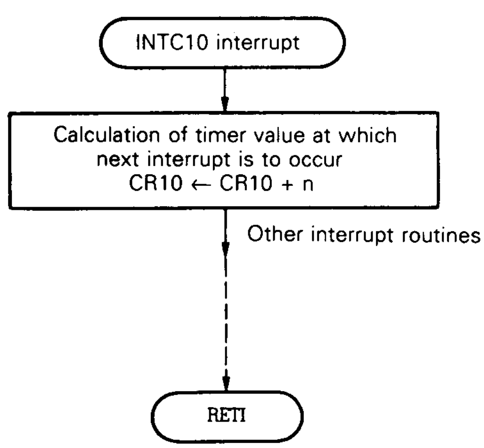


Fig. 7-65 Interrupt Request Processing for Interval Timer Operation (1)

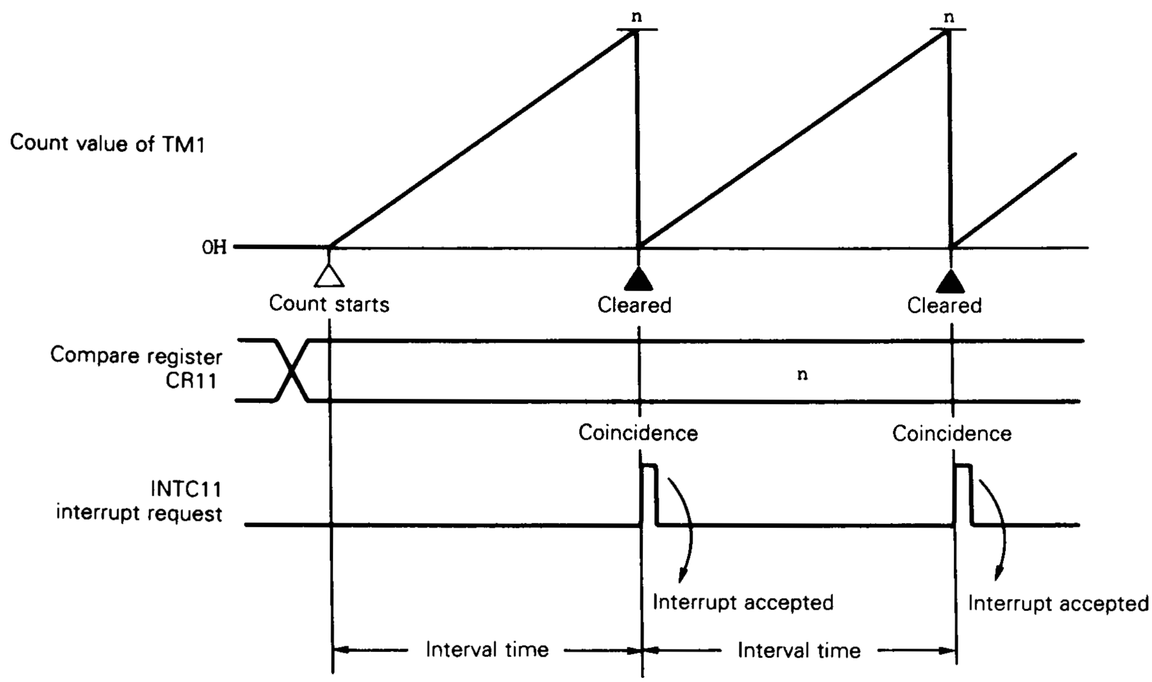


(2) Operation as interval timer (2)

In this example, the 8-bit timer operates as an interval timer that repeatedly generates an interrupt at predetermined time intervals (see **Fig. 7-66**).

Fig. 7-67 shows the control register contents, while Fig. 7-68 shows the procedure used to set the register contents.

Fig. 7-66 Interval Timer Operation (2) Timing(When CR11 is used as the compare register)



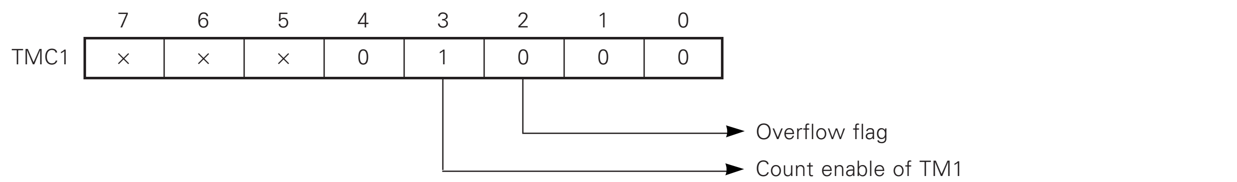
Remarks: Interval time = (n + 1) x x/f_{CLK}

; $0 \leqq n \leqq$ FFH

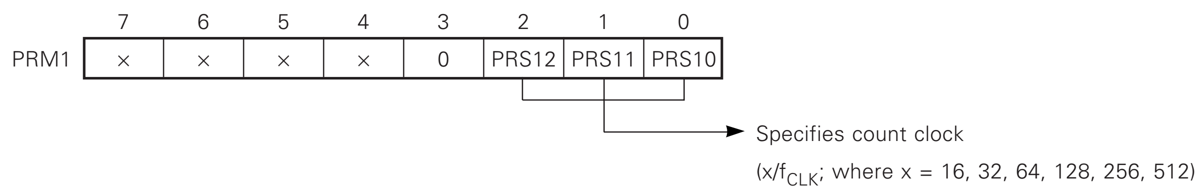
x = 16, 32, 64, 128, 256, 512

Fig. 7-67 Control Register Contents for Interval Timer Operation (2)

(a) Timer control register 1 (TMC1)



(b) Prescaler mode register 1 (PRM1)



(c) Capture/compare control register (CRC1)

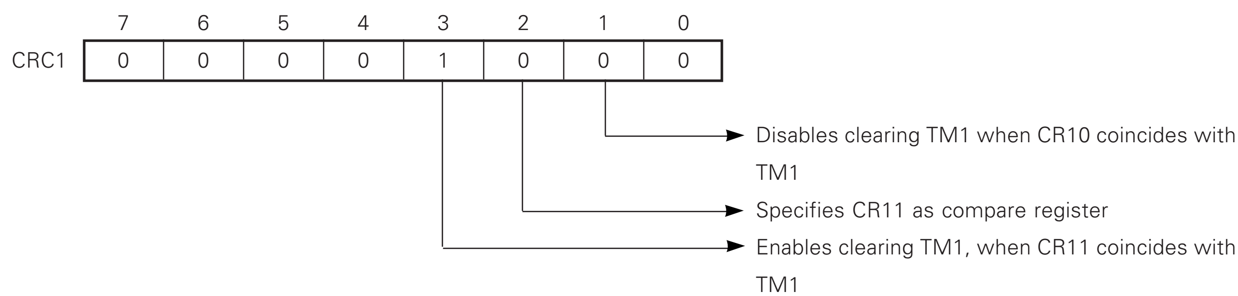
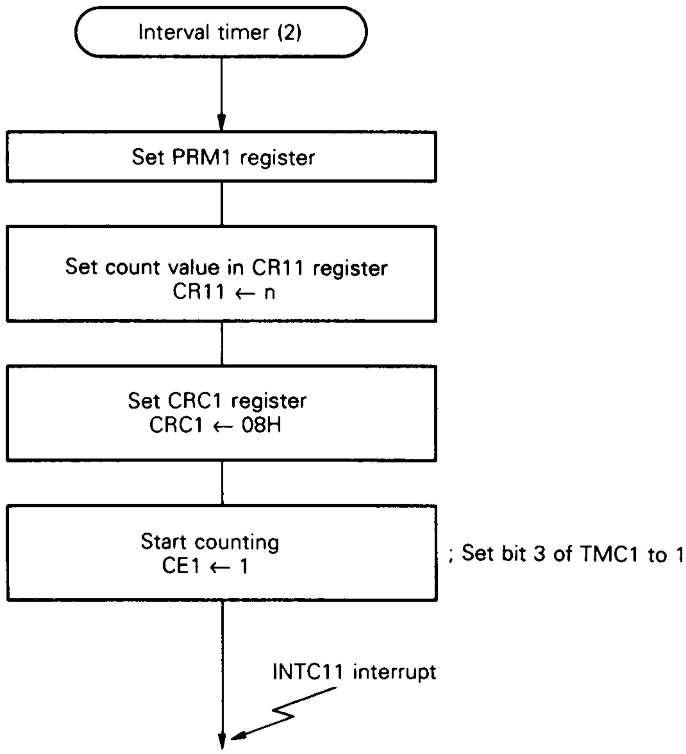


Fig. 7-68 Interval Timer Operation (2) Setting Procedure



(3) Operation to measure pulse width

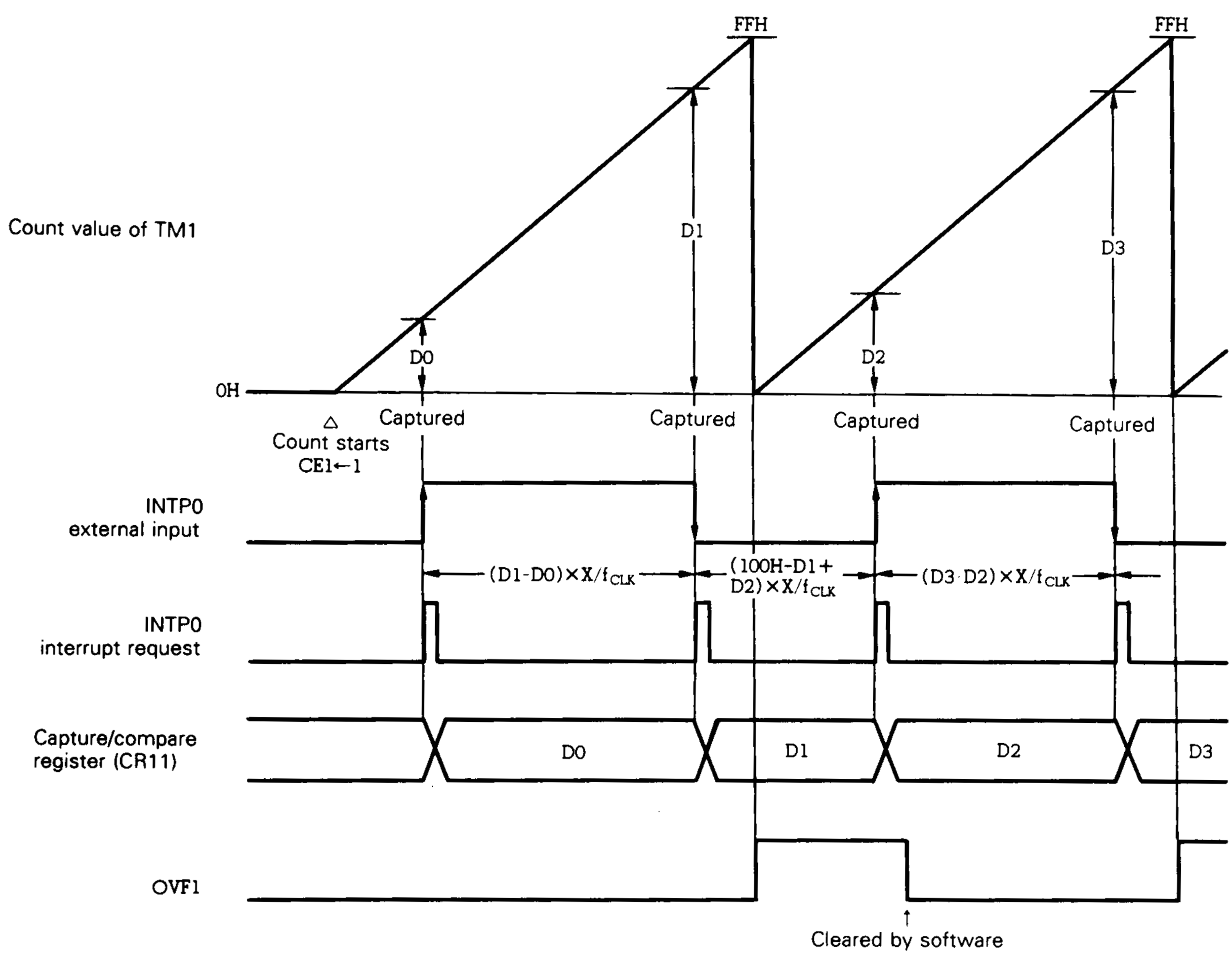
The high-level or low-level width for an external pulse, input to external interrupt request pin INTP0, can be measured by the 8-bit timer.

The width of the pulse, input to the INTP0 pin, must be at least 12 system clocks (2μs at f_{CLK} = 6 MHz), regardless of whether the level of the pulse is high or low. Otherwise, the valid edge for the pulse cannot be detected and captured.

As shown in Fig. 7-69, the current count value for the 8-bit timer (TM1) is captured to capture/compare register CR11, which is specified to function as a capture register, in synchronization with the valid edge for the INTP0 pin (the valid edge is specified to be both the rising and falling edges). The timer value is then retained in the capture register. To measure the pulse width, the difference between the count value for TM1 (Dn), which has been captured to and retained in the CR11 register, when the nth valid edge has been detected, and the count value for TM1 (Dn-1), when the n-1th valid edge has been detected, is calculated. This difference is then multiplied by the count clock (X/f_{CLK}, where X = 16, 32, 64, 128, 256, or 512) to calculate the pulse width.

Fig. 7-70 shows the control register contents, while Fig. 7-71 shows the procedure used to set the control register contents.

Fig. 7-69 Pulse Width Measurement Timing(When CR11 is used as the capture register)



Remarks: Dn: Count value for TM1 (n = 0, 1, 2,...)
x = 16, 32, 64, 128, 256, 512

Fig. 7-70 Control Register Contents for Pulse Width Measurement

(a) Timer control register 1 (TMC1)

	7	6	5	4	3	2	1	0
TMC1	×	×	×	0	1	0	0	0

- Bit 2 → Overflow flag
- Bit 3 → Count enable of TM1

(b) Prescaler mode register 1 (PRM1)

	7	6	5	4	3	2	1	0
PRM1	×	×	×	×	0	PRS12	PRS11	PRS10

- PRS12, PRS11, PRS10 → Specifies count clock (x/f_{CLK}, x = 16, 32, 64, 128, 256, 512)

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

	7	6	5	4	3	2	1	0
CRC1	0	0	0	0	0	1	0	0

- Bit 1 → Clearing TM1, when TM1 coincides with CR10, is disabled
- Bit 2 → Specifies CR11 register as capture register
- Bit 3 → Clearing TM1, when TM1 is captured to CR11, is disabled

(d) External interrupt mode register 0 (INTM0)

	7	6	5	4	3	2	1	0
INTM0	×	×	×	×	1	1	0	×

- Bits 3, 2 → Specifies both rising and falling INTP0 input edges as valid

x: don't care

Fig. 7-71 Pulse Width Measurement Setting Procedure

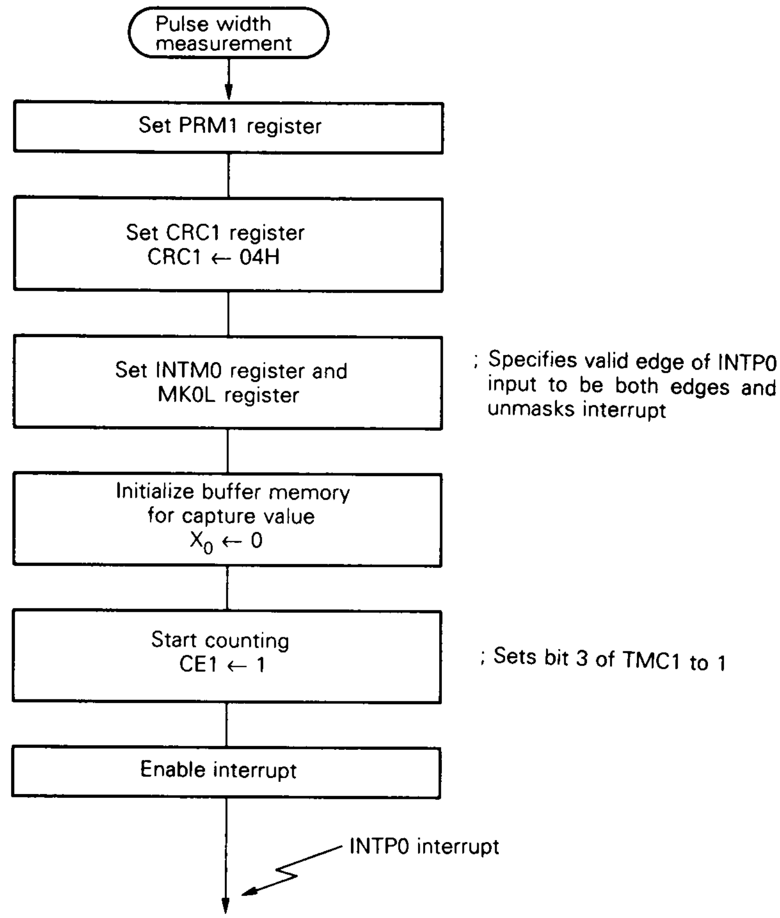
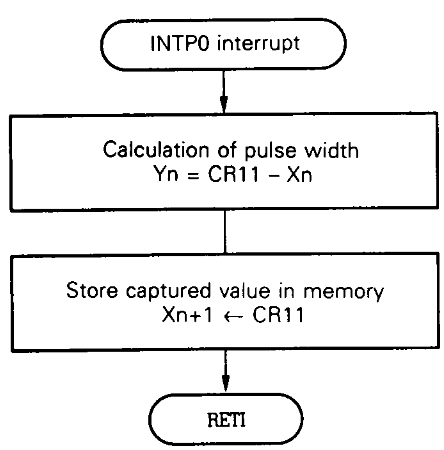


Fig. 7-72 Interrupt Request Processing to Calculate Pulse Width



7.3 8-bit Timer/Counter 2

7.3.1 Function

The 8-bit timer/counter 2 has the following two functions, which are not provided to the other three timers/counters:

- External event counter
- One-shot timer*

This section describes the following four basic functions for 8-bit timer/counter 2:

- Interval timer
- Programmable square wave output
- Pulse width measurement
- External event counter

***:** The one-shot timer function is operation of the timer register (TM2) itself. Therefore, it is different from the one-shot pulse output function for the 16-bit timer/counter.

(1) Interval timer

When 8-bit timer/counter 2 is used as an interval timer, an internal interrupt occurs at time intervals specified by the interval timer.

Table 7-11 8-bit Timer/Counter 2 Time Interval

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

Figures in ( ) are at f_{CLK} = 6 MHz.

(2) Programmable square wave output

This function is to output square waves from two timer output pins, TO2 and TO3. The square waves can be output independently from each other.

Table 7-12 Programmable Square Wave Output Range

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)

Figures in ( ) are at f_{CLK} = 6 MHz

Caution: The data in the above table are for when the internal clock is used.

(3) Pulse width measurement

The pulse width for a signal input to external interrupt pin INTP1 can be measured by 8-bit timer/counter 2.

Table 7-13 Pulse Width Measurement Range

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

Figures in ( ) are at f_{CLK} = 6 MHz.

(4) External event counter

Timer/counter 2 can also be used to count the clock pulses (CI pin input pulse) input from external interrupt input pin INTP2.

Table 7-14 shows the clocks that can be input to 8-bit timer/counter 2.

Table 7-14 Clock That Can Be Input to 8-bit Timer/Counter 2

	To count single edge	To count both edges
Maximum frequency	f_{CLK}/24 (250 kHz)	f_{CLK}/32 (187.5 kHz)
Minimum pulse width* (high and low levels)	12/f_{CLK} (2 μs)	16/f_{CLK} (2.67 μs)

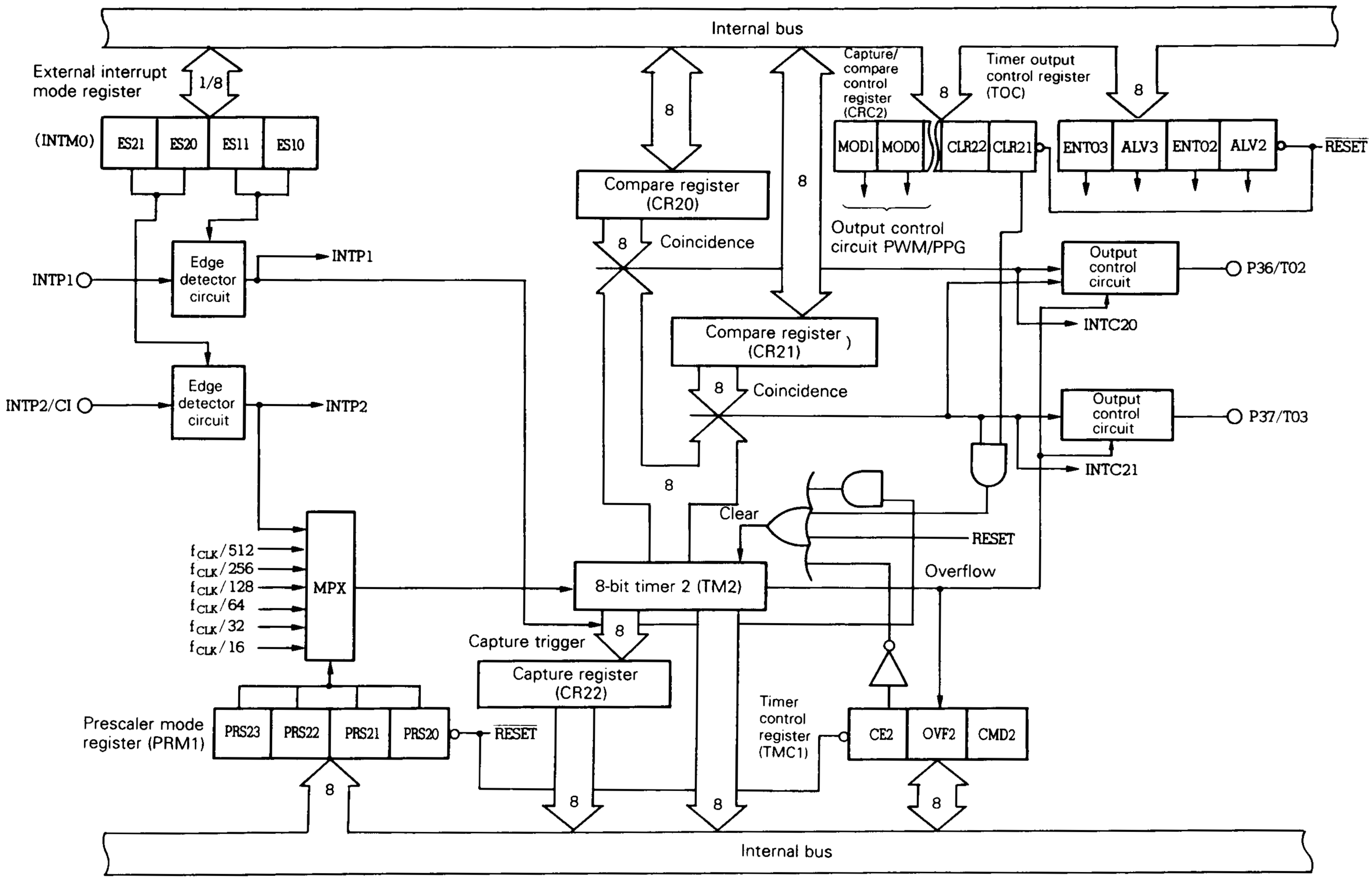
Figures in ( ) are at f_{CLK} = 6 MHz.

7.3.2 Configuration

The 8-bit timer/counter 2 consists of an 8-bit timer (TM2), two 8-bit compare registers (CR20 and CR21), and an 8-bit capture/compare register (CR22).

Figure 7-73 shows the configuration for 8-bit timer/counter 2.

Fig. 7-73 Configuration of 8-bit Timer/Counter 2



(1) 8-bit timer 2 (TM2)

Timer TM2 counts up the count clock specified by the higher 4 bits of the prescaler mode register 1 (PRM1) and external interrupt mode register 0 (INTM0). The internal clock and external clock can be selected as the count clock.

The timer contents can only be read by an 8-bit manipulation instruction. The timer counting operation can be disabled or enabled by timer control register 1 (TMC1).

When the $\overline{\text{RESET}}$ signal is input, the TM2 contents are cleared to 00H and TM2 stops counting.

(2) Compare registers (CR20 and CR21)

CR20 and CR21 are 8-bit registers holding values that determine the interval timer cycle.

When these register contents coincide with the TM2 value, interrupt requests (INTC20 and INTC21) are generated. The TM2 count value can also be cleared when coincidence takes place between the TM2 count value and the CR21 contents.

Data can be read from or written to these registers by an 8-bit manipulation instruction. When the $\overline{\text{RESET}}$ signal is input, the contents of these registers become undefined.

(3) Capture register (CR22)

CR22 is an 8-bit register that captures the TM2 contents.

The TM2 contents are captured to this register in synchronization with the input from the valid edge (capture trigger) to external interrupt input pin INTP1. The CR22 register contents are retained, until the next capture trigger is generated, or the contents of the register are read. After the register has been read, its contents become undefined until the next capture trigger is generated and a new value is set in the register. After the capturing, TM2 can be cleared.

Data can only be read from this register by an 8-bit manipulation instruction. When the $\overline{\text{RESET}}$ signal is input, register contents become undefined.

(4) Edge detector circuit

This circuit detects the valid edge for a signal input from an external source. It detects the valid edge specified by external interrupt mode register 0 (INTM0), input to the INTP1 input pin and generates interrupt INTP1 and a capture trigger. In addition, when the circuit detects the valid edge input from external interrupt request input pin INTP2, it generates the count clock for TM2 and INTP2.

(5) Output control circuits

These circuits can invert the TM2 output signals when the CR20 and CR21 register contents coincide with the timer value. Timer output pins (TO2 and TO3) can output square waves, when so specified by the higher 4 bits of the timer output control register (TOC). At this time, PWM or PPG signals can also be output, when so specified by the capture/compare control register (CRC2).

The TOC register can also disable or enable the timer output. When the timer output is disabled, pins TO2 and TO3 output signals, whose levels are fixed (the output levels are specified by the TOC register).

(6) Prescaler

The prescaler generates the count clock from the internal system clock. The clock generated by the prescaler is selected by the selector as the count clock, based on which timer performs its counting operation.

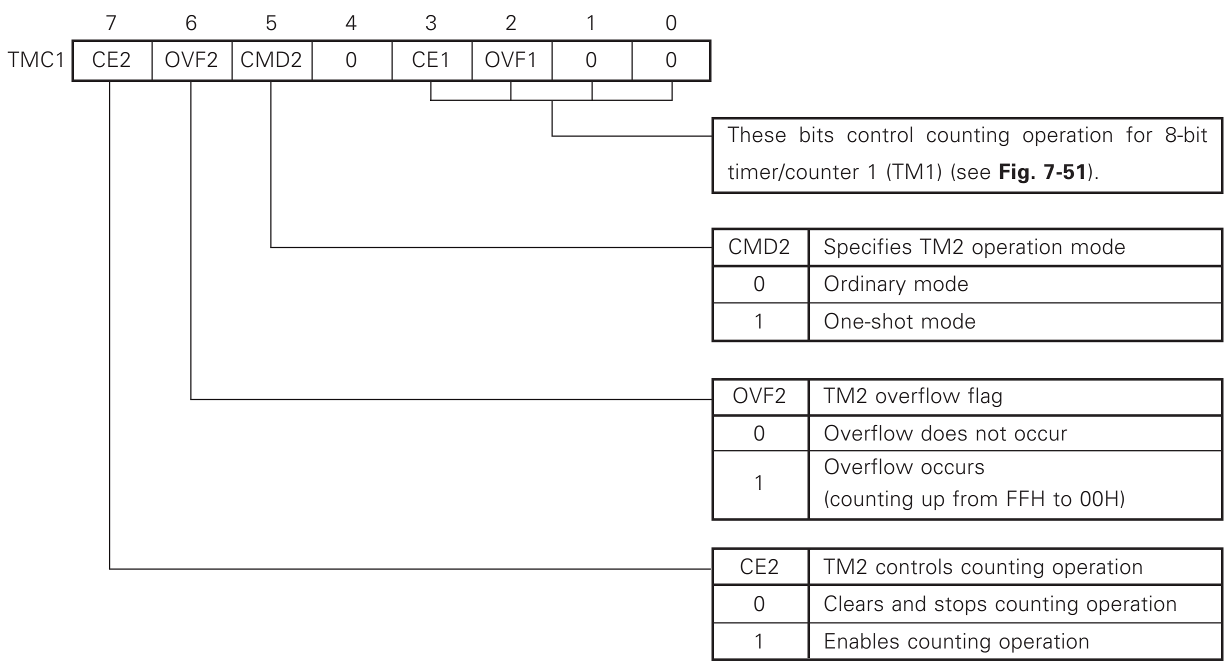
7.3.3 8-bit timer/counter 2 control registers

(1) Timer control register 1 (TMC1)

Register TMC1 is an 8-bit register that controls the counting operation for 8-bit timers 1 and 2 (TM1 and TM2). The lower 4 bits for this register are used to control the counting operation for 8-bit timer/counter 1 (TM1), while the higher 4 bits are used to control the counting operation for 8-bit timer/counter 2 (TM2). Data can be read from or written to this register by an 8-bit manipulation instruction. Fig. 7-74 shows the format for this register.

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

Fig. 7-74 Timer Control Register 1 (TMC1) Format



Caution: The OVF2 bit can be cleared by software only.

(2) Prescaler mode register 1 (PRM1)

This 8-bit register specifies the count clock supplied to 8-bit timers 1 and 2 (TM1 and TM2).

The lower 4 bits for this register are used to specify the count clock supplied to 8-bit timer/counter 1 (TM1), while the higher 4 bits are used to specify the count clock supplied to 8-bit timer/counter 2 (TM2).

Data can only be written to this register by an 8-bit manipulation instruction. Figure 7-75 shows the format of this register.

When the $\overline{\text{RESET}}$ signal is input, the contents of the PRM1 are cleared to 00H.

Fig. 7-75 Prescaler Mode Register 1 (PRM1) Format

	7	6	5	4	3	2	1	0
PRM1	PRS23	PRS22	PRS21	PRS20	0	PRS12	PRS11	PRS10

These bits specify count clock supplied to 8-bit timer/counter 1 (TM1) (see **Fig. 7-52**).

f_{CLK}=6MHz

<table>
<tr><th>PRS23</th><th>PRS22</th><th>PRS21</th><th>PRS20</th><th>Specifies count clock (Hz)</th><th>Resolution</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td rowspan="3">f_{CLK}/16</td><td rowspan="3">2.6 µs</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>f_{CLK}/32</td><td>5.3 µs</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>f_{CLK}/64</td><td>10.7 µs</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>f_{CLK}/128</td><td>21.3 µs</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>f_{CLK}/256</td><td>42.7 µs</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>f_{CLK}/512</td><td>85.3 µs</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td colspan="2">External clock (CI)</td></tr>
</table>

Remarks: f_{CLK}: system clock frequency

(3) Capture/compare control register 2 (CRC2)

The CRC2 register enables or disables clearing the 8-bit timer 2 (TM2) by the capture register (CR22)/compare register (CR21), or specifies the timer output (TO2 and TO3) modes.

Data can only be written to this register by an 8-bit manipulation instruction. The format for this register is shown in Fig. 7-76.

When the $\overline{\text{RESET}}$ signal is input, the contents of this register are cleared to 10H.

Fig. 7-76 Capture/Compare Control Register 2 (CRC2) Format

	7	6	5	4	3	2	1	0
CRC2	MOD1	MOD0	CLR22	1	CLR21	0	0	0

MOD1	MOD0	CLR22	CLR21	Timer output mode		Clearing TM2
				TO2	TO3	
0	0	0	0	Toggle output	Toggle output	Does not clear
0	0	0	1	Toggle output	Toggle output	Clears when TM2 coincides with CR21
0	0	1	0	Toggle output	Toggle output	Clears after TM2 captured to CR22
0	0	1	1	Toggle output	Toggle output	Clears after TM2 coincides with CR21 or TM2 captured to CR22
0	1	0	0	PWM output	Toggle output	Does not clear
1	0	0	0	PWM output	PWM output	Does not clear
1	1	0	1	PPG output	Toggle output	Clears when TM2 coincides with CR21

Caution: Combinations of bits other than above are inhibited.

(4) Timer output control register (TOC)

This 8-bit register specifies the active level for timer output pins and disables or enables timer output.

The lower 4 bits for this register control the output operations for the 16-bit timer/counter (i.e., the operations of pins TO0 and TO1), while the higher 4 bits control the output operations of 8-bit timer/counter 2 (operations for pins TO2 and TO3).

Data can only be written to this register by an 8-bit manipulation instruction. Figure 7-77 shows the TOC register format.

When the $\overline{\text{RESET}}$ signal is input, the contents of this register are cleared to 00H.

Fig. 7-77 Timer Output Control Register (TOC) Format

	7	6	5	4	3	2	1	0
TOC	ENTO3	ALV3	ENTO2	ALV2	ENTO1	ALV1	ENTO0	ALV0

Bits 3 to 0: These bits control the output operation for 16-bit timer/counter (TO0 and TO1 pins) (see **Fig. 7-5**).

ALV2	Pin TO2 active level	
	When toggle output is specified	When PWM/PPG output is specified
0	Low level	High level
1	High level	Low level

ENTO2	Specifies pin TO2 operation
0	Outputs $\overline{\text{ALV2}}$
1	Enables pulse output

ALV3	Active level of TO3 pin	
	When toggle output is specified	When PWM/PPG output is specified
0	Low level	High level
1	High level	Low level

ENTO3	Specifies pin TO3 operation
0	Outputs $\overline{\text{ALV3}}$
1	Enables pulse output

7.3.4 8-bit timer 2 (TM2) operation

(1) Basic operation

The 8-bit timer/counter 2 counts up the count clocks specified by the higher 4 bits of the prescaler mode register 1 (PRM1).

The TM2 counting operation is enabled or disabled by the bit 7 (CE2) for timer control register 1 (TMC1) (operation for 8-bit timer/counter 2 is controlled by the higher 4 bits for the TMC1 register). When the CE2 bit is set to 1 by software, the TM2 contents are cleared to 00H at the first count clock. Then TM2 starts the count-up operation.

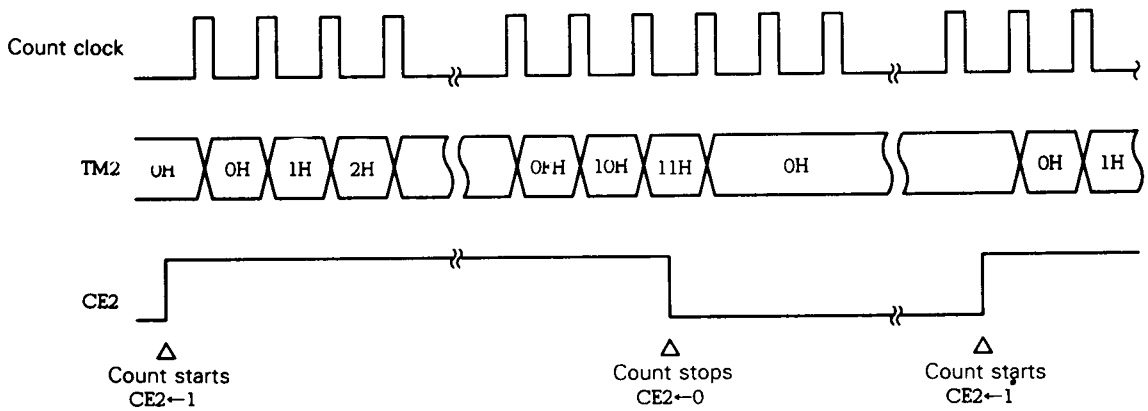
When the CE2 bit is cleared to 0 by software, the TM2 contents are cleared to 00H, at the next count clock, and the capture operation and generation of the coincidence signal are stopped. If an attempt is made to set the CE2 bit, which has already been set to 1, TM2 is not cleared but continues its operation (see **Fig. 7-78(b)**).

When a count clock is input while the current value for TM2 is FFH, the timer is cleared to 00H. At this time, the OVF2 bit is set to 1, sending an overflow signal to the output control circuits. The OVF2 bit can only be cleared by software. At this time, TM2 continues the counting operation.

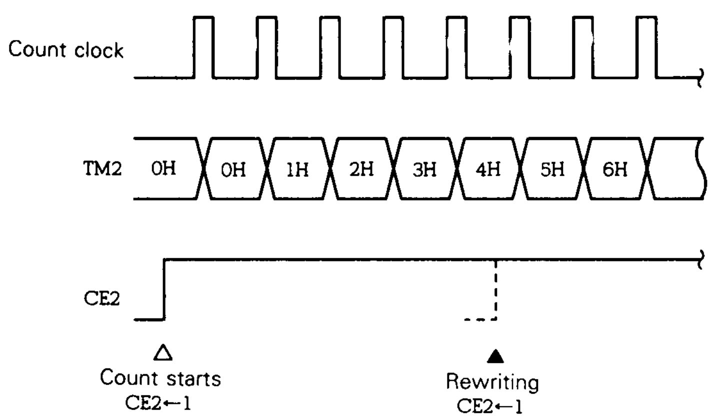
When the $\overline{\text{RESET}}$ signal is input, the TM2 contents are cleared to 00H and TM2 stops counting.

Fig. 7-78 Basic Operation for 8-bit Timer 2 (TM2)

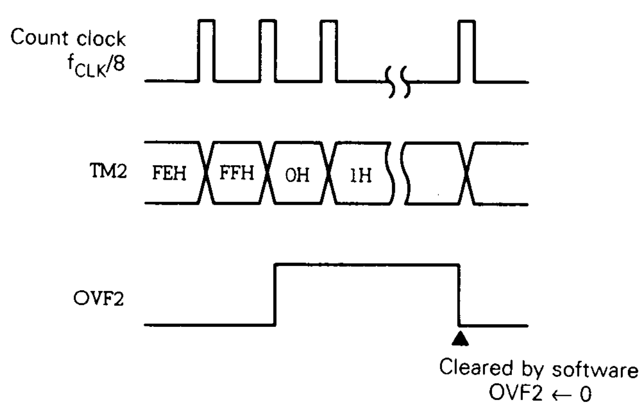
(a) When counting is started, stopped, and then started again



(b) When "1" is written to CE2 bit again after counting is started



(c) TM2 operation, when its current value is FFH



(2) Clearing oepration

The 8-bit timer 2 (TM2) can be automatically cleared after its value has coincided with the contents of the compare register (CR21) or captured. When the reason for clearing TM2 has occurred, TM2 is cleared to 00H at the next count clock. Therefore, even if the clearing cause has occurred, the current TM2 value is retained, until the next count clock is applied.

Fig. 7-79 Clearing TM2 by Coincidence with Compare Register (CR21)

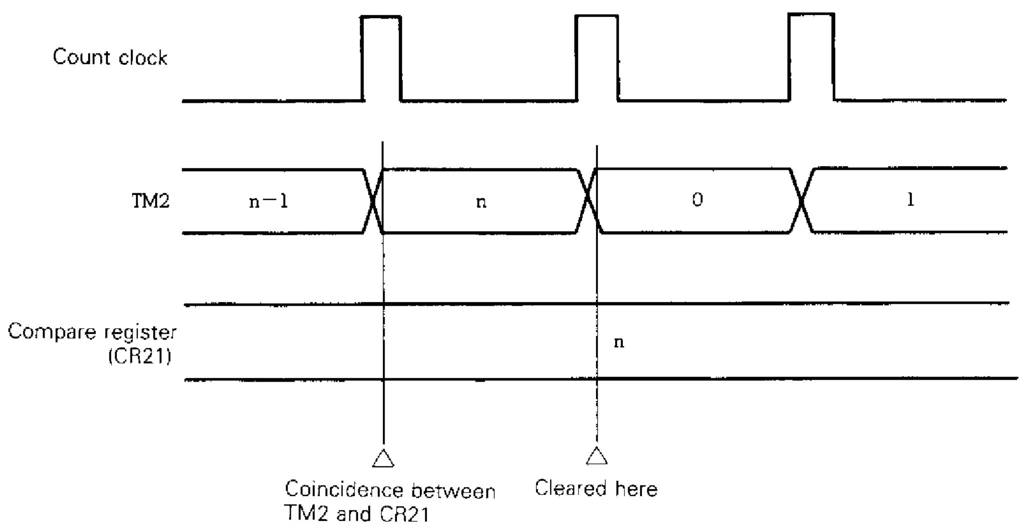
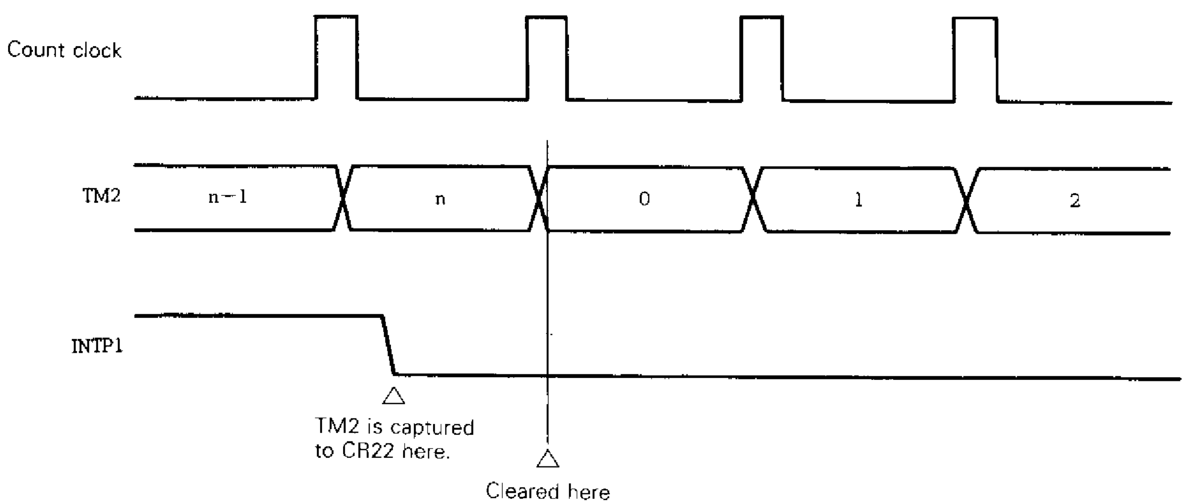


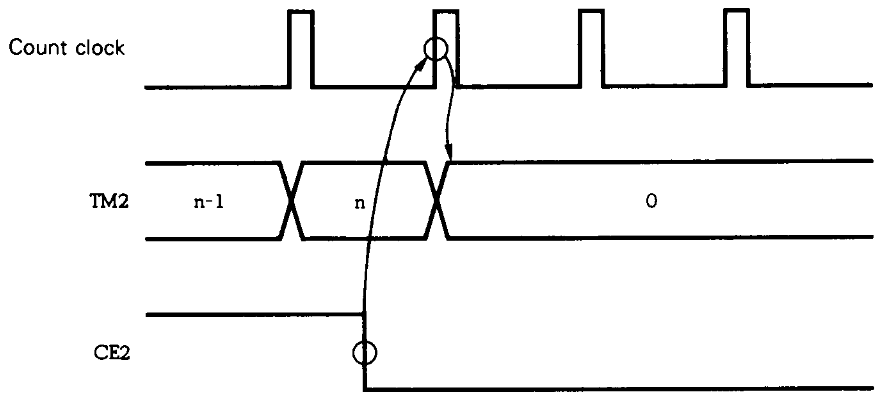
Fig. 7-80 Clearing TM2 after Capturing



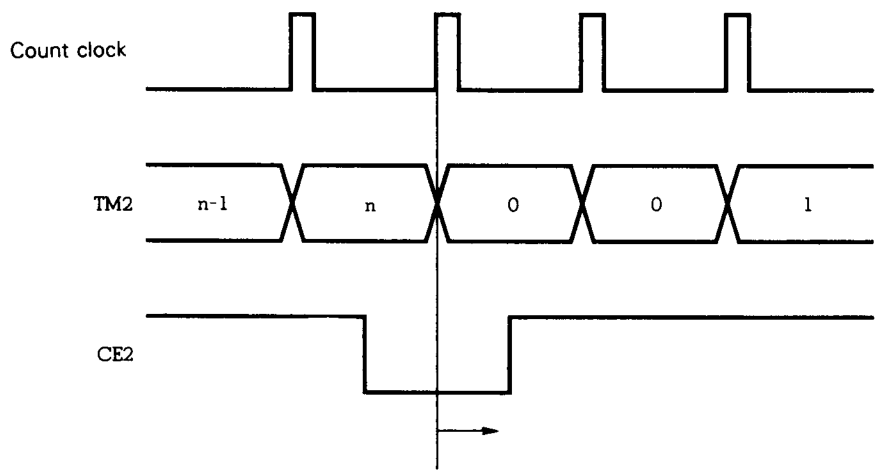
TM2 is also cleared by resetting the CE2 bit of the timer control register (TMC1) to 0 through software. The clear operation is also performed by the count clock after the CE2 bit has been reset to 0. If the CE2 bit is set to 1 after the CE2 bit has been reset to 0 and before TM0 is cleared to 0 (before the first count clock is input after the CE0 bit has been reset to 0), TM2 is cleared to 0 and at the same time, 0 counting operation is performed because counting has been started.

Fig. 7-81 Clear Operation When CE2 Bit Is Reset to 0

(a) Basic operation

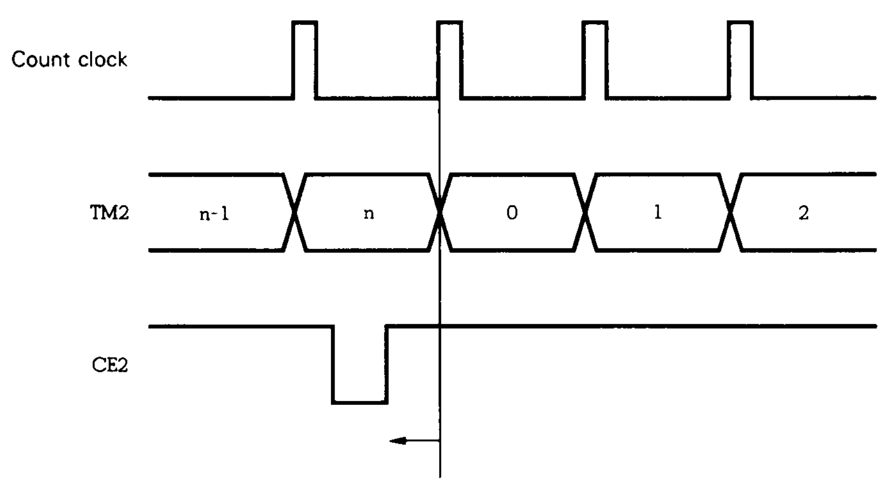


(b) Restart after TM2 has been cleared to 0



If 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 TM2 is cleared to 0



If the CE2 bit is set to 1 before this count clock, TM2 is cleared by CE2 ← 0 and 0 counting is performed by CE2 ← 1 simultaneously.

7.3.5 External event counter function

The 8-bit timer/counter 2 can count the clock pulses input to the CI pin from an external source.

External event counter operation mode does not require any special methods for its selection.

When the count clock for TM2 is specified to be an external clock by the higher 4 bits for prescaler mode register 1 (PRM1), TM2 functions as an external event counter.

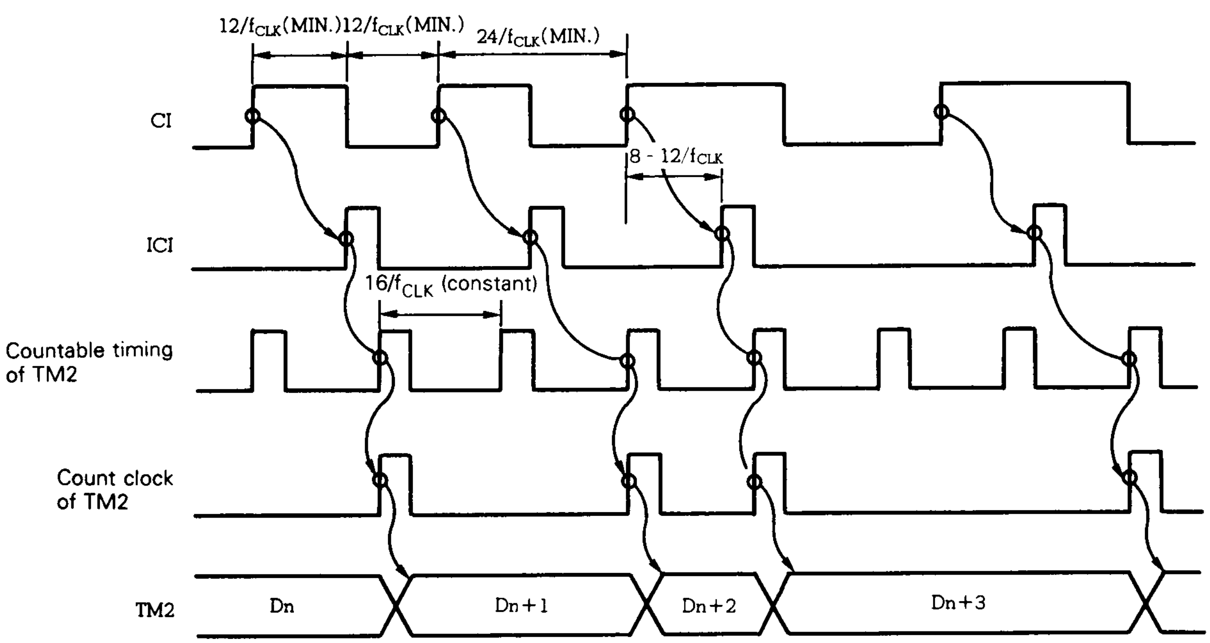
The maximum frequency for an external clock pulse that can be counted as an external event is 187.5 kHz, when both the edges of the CI input signal are to be counted, and is 250 kHz when either of the edges is to be counted (at f_{CLK} = 6 MHz).

To count both the edges, pulse widths at both high and low levels must be at least 16 system clocks (2.67 μs at f_{CLK} = 6 MHz); otherwise, the external events may not be counted. To count either of the edges, the pulse widths at both high and low levels must be at least 12 system clocks (2 μs at f_{CLK} = 6 MHz).

Figure 7-82 shows the operation timing for 8-bit timer/counter 2, when it is used as an external event counter.

Fig. 7-82 External Event Count Timing for 8-bit Timer/Counter 2 (1/2)

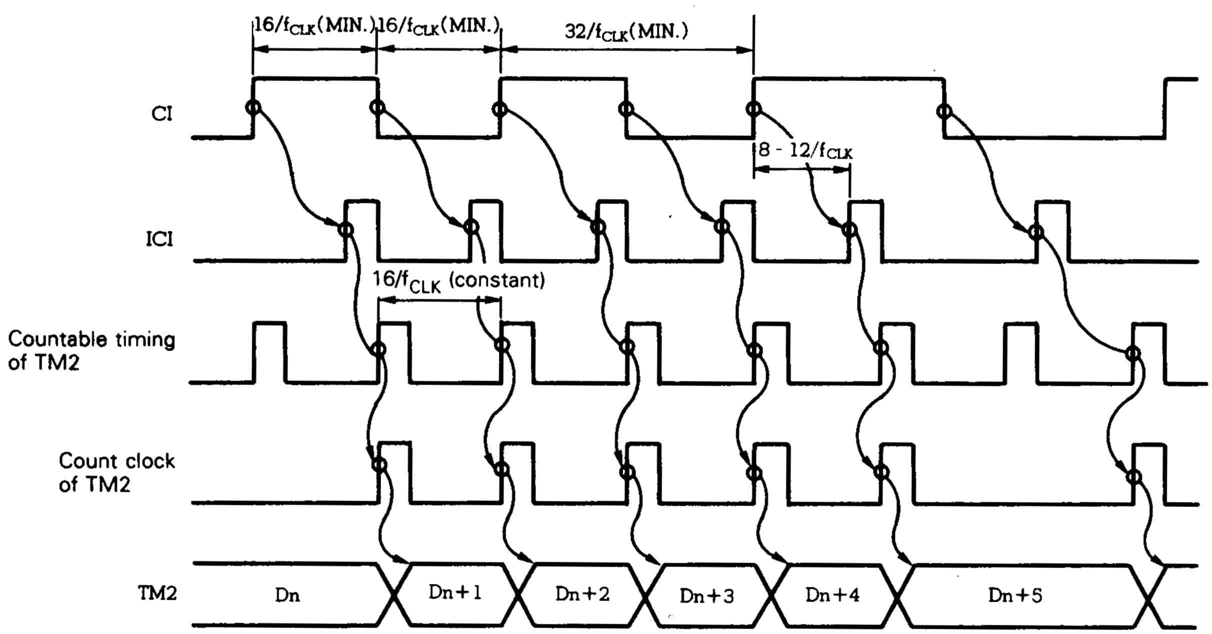
(1) To count single edge (maximum frequency = f_{CLK}/24)



Remarks: ICI: CI input signal after being processed by edge detector circuit

Fig. 7-82 External Event Count Timing for 8-bit Timer/Counter 2 (2/2)

(2) To count both edges (maximum frequency = f_{CLK}/32)



Remarks: ICI: CI input signal after being processed by edge detector circuit

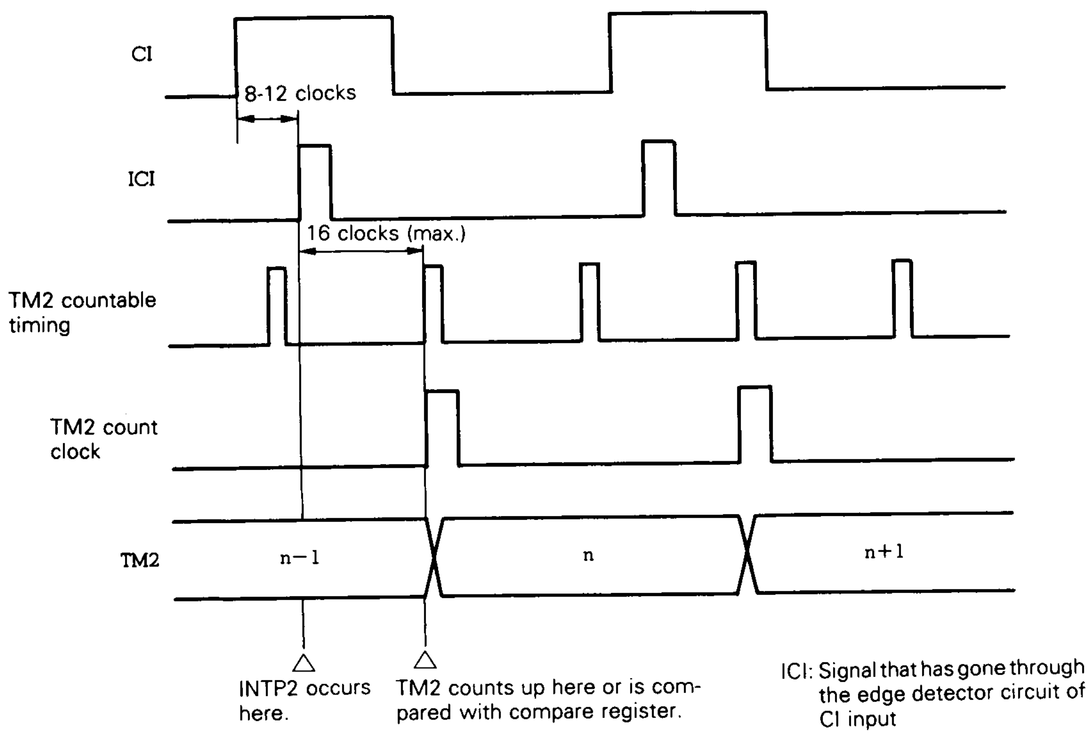
The TM2 counting operation is controlled by the CE2 bit for the TMC1 register, in the same manner as when TM2 performs its basic operation.

When the CE2 bit is set to 1 by software, the TM2 contents are cleared to 00H at the first count clock, and TM2 starts counting up.

When the CE2 bit is cleared to 0 by software, while TM2 is operating, the TM2 contents are cleared to 00H at the next count clock, and TM2 stops. If an attempt is made to set the CE2 bit, which has already been set to 1, the TM2 operation is not affected.

Caution 1: When using the 8-bit timer/counter 2 as an external event counter, up to 28 system clocks (4.67 μs: f_{CLK} = 6 MHz) are required until TM2 is incremented, after the valid edge has been input to the CI pin. Therefore, TM2 may not be incremented after it has been read immediately after the valid edge has been detected. Similarly, an interrupt generated by coincidence between TM2 and compare registers (CR20, CR21) may also be delayed by the edge input. Take this into consideration, when detailed timing control must be implemented after the edge has been input.

Fig. 7-83 Interrupt Request Generation by External Event Counter



Caution 2: When the 8-bit timer/counter 2 is used as an external event counter, the status in which the valid edge is not input at all, and the status in which the valid edge has been input only once, cannot be distinguished by TM2 alone (refer to Fig. 7-84), because the TM2 contents are cleared to 0 in both statuses. If it is necessary to distinguish between the two statuses, use the interrupt request flag of INTP2 (the INTP2 pin is multiplexed with the CI pin and either function can be used). Figure 7-85 shows an example.

Fig. 7-84 If One Valid Edge Input Cannot Be Distinguished from No Valid Edge Input by External Event Counter

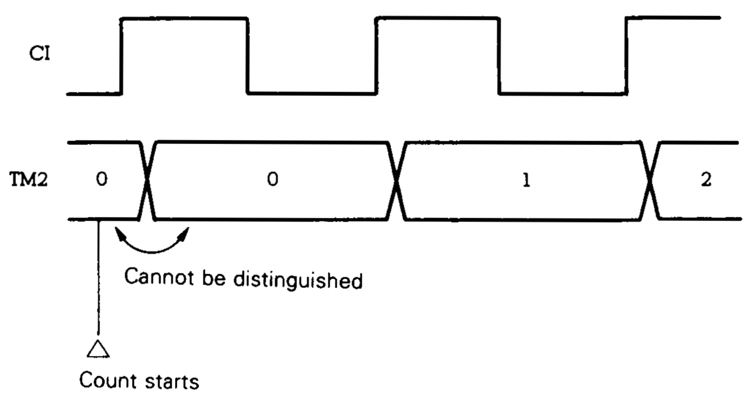
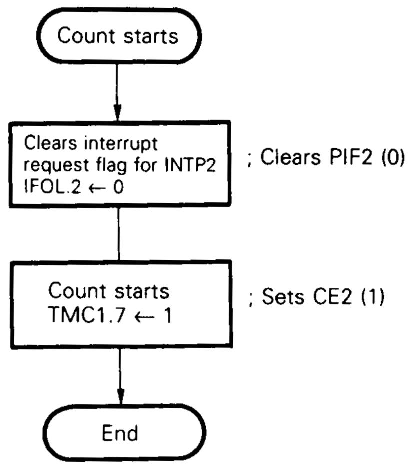
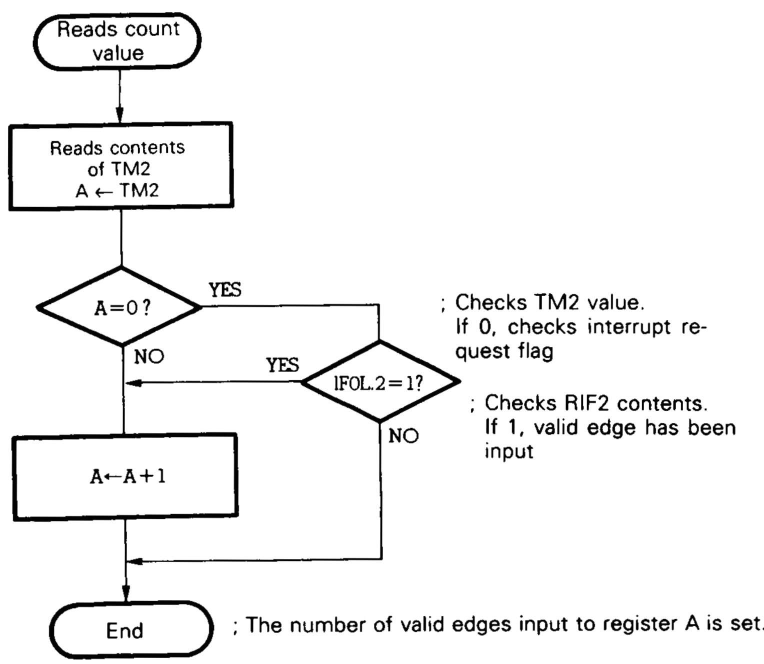


Fig. 7-85 To Distinguish by External Counter

(a) Processing on starting of count



(b) Processing when count value is read

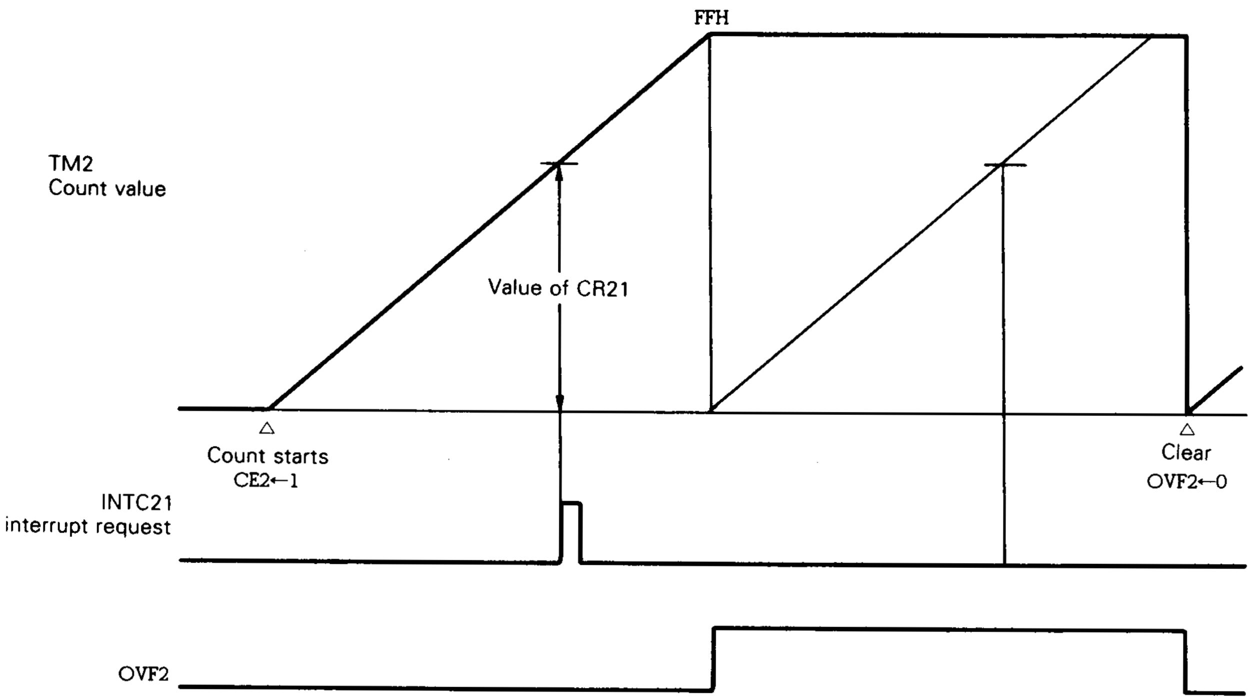


Caution 3: In the in-circuit emulator, the CI/INTP2 pin cannot normally perform digital noise elimination. When using the event counter, refer to 7.5.4 Notes for using the in-circuit emulator.

7.3.6 One-shot timer function

The 8-bit timer/counter 2 is provided with an operation mode in which TM2 automatically stops, when its count value has reached the maximum value (FFH).

Fig. 7-86 One-Shot Timer Operation



As shown in Fig. 7-86, a one-shot interrupt occurs, when the TM2 count value coincides with the values (0H to FFH) set in advance in the CR20 and CR21 registers.

The one-shot timer operation mode can be specified by setting the bit 5 (CMD2) for timer control register 1 (TMC1) to 1 by software.

The TM2 counting operation is controlled by the CE2 bit for the TMC1 register, in the same manner as when TM2 performs the basic operation.

When the CE2 bit is set to 1 by software, the TM2 contents are cleared to 00H at the first count clock, and TM2 starts counting up.

When the current count value for TM2 has reached FFH (full count) as a result, the bit 6 (OVF2) for the TMC1 register is set to 1, and TM2 stops, retaining the last count value, FFH.

To start the one-shot timer operation again, while TM2 stops, clear the OVF2 bit to 0 by software. When this bit is cleared, the TM2 contents are cleared to 00H at the next count clock, and TM2 starts counting up again.

If the CE2 bit is cleared to 0 by software, while TM2 is operating, the TM2 contents are cleared to 00H at the next count clock, and TM2 stops. If an attempt is made to set the CE2 bit, which has already been set to 1, the TM2 counting operation is not affected.

7.3.7 Compare registers and capture register operations

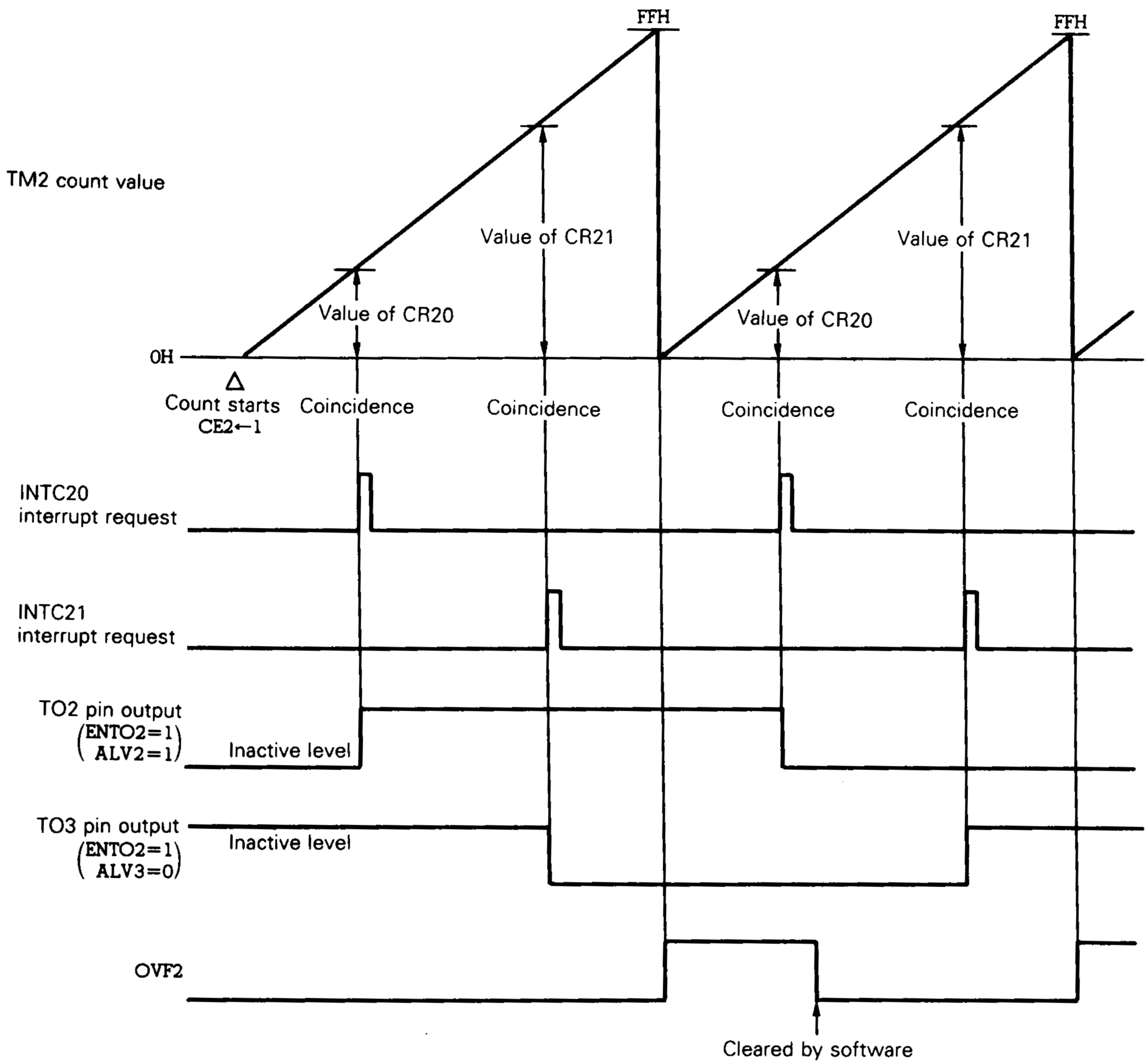
(1) Compare operation

The 8-bit timer/counter 2 can also perform a compare operation, which compares the current count value for the timer with the values set for the compare register.

When the count value for the 8-bit timer 2 (TM2) coincides with the values set in advance in the compare registers (CR20 and CR21), coincidence signals are sent to the output control circuits. At the same time, interrupt request signals (INTC20 and INTC21) are generated.

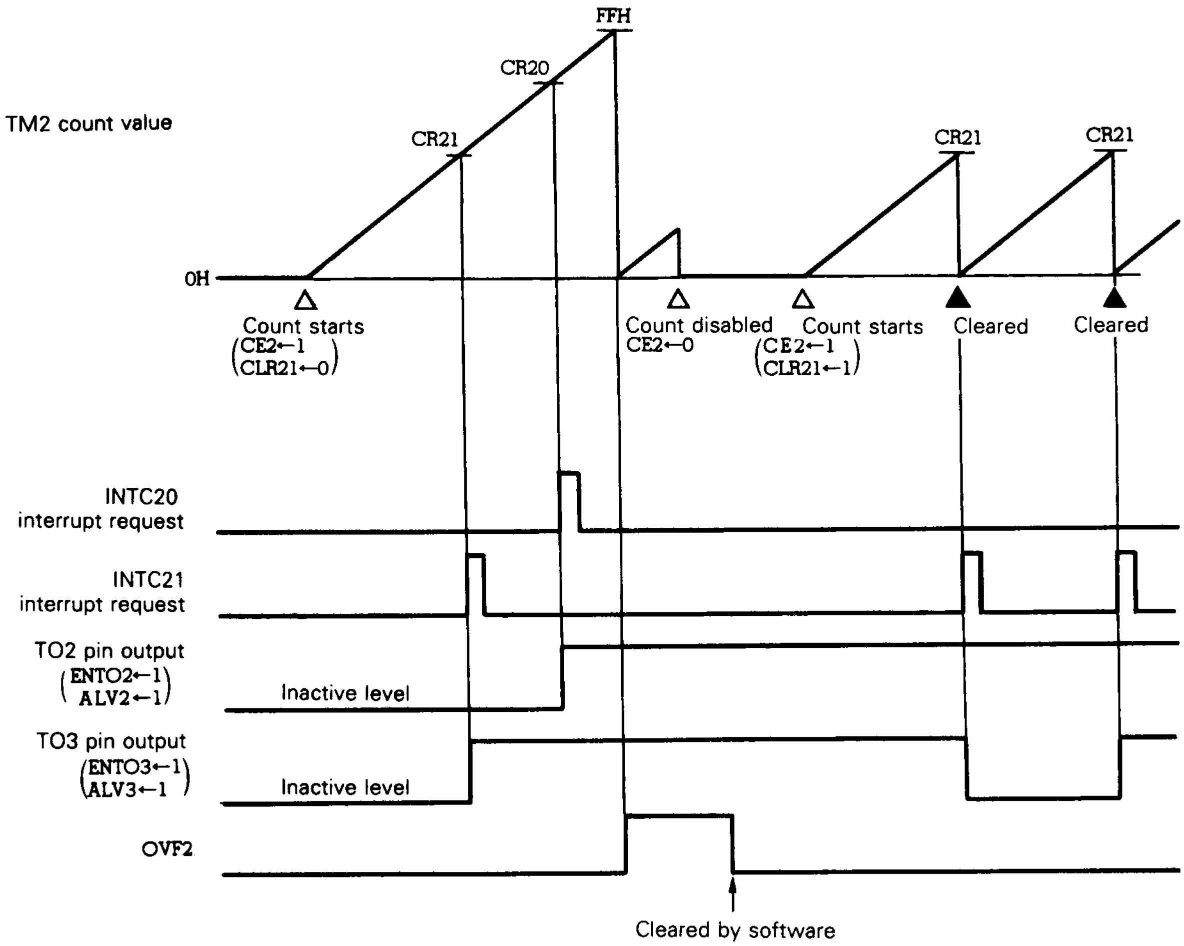
After the timer value has coincided with the CR21 register value, the TM2 count value can be cleared. In this case, the timer functions as an interval timer that repeatedly counts up to the value set in the CR21 register.

Fig. 7-87 Compare Operation



Remarks: CLR21 = 0, CLR22 = 0

Fig. 7-88 Clearing TM2 after Coincidence Detection



Remarks: CLR22 = 0

Caution: Refer to 7.5.4 Notes for using in-circuit emulator.

(2) Capture operation

The 8-bit timer/counter 2 can also perform a capture operation, by which to capture the count value for the timer to the capture register, which then retains the value, in synchronization with an external trigger.

As the external trigger, the valid edge detected on external interrupt request input pin INTP1 is used (capture trigger). The count value for the 8-bit timer 2 (TM2) is captured to the capture/compare register (CR22) in synchronization with the capture trigger. If the CR22 register contents have been read by the program, the register contents will become undefined.

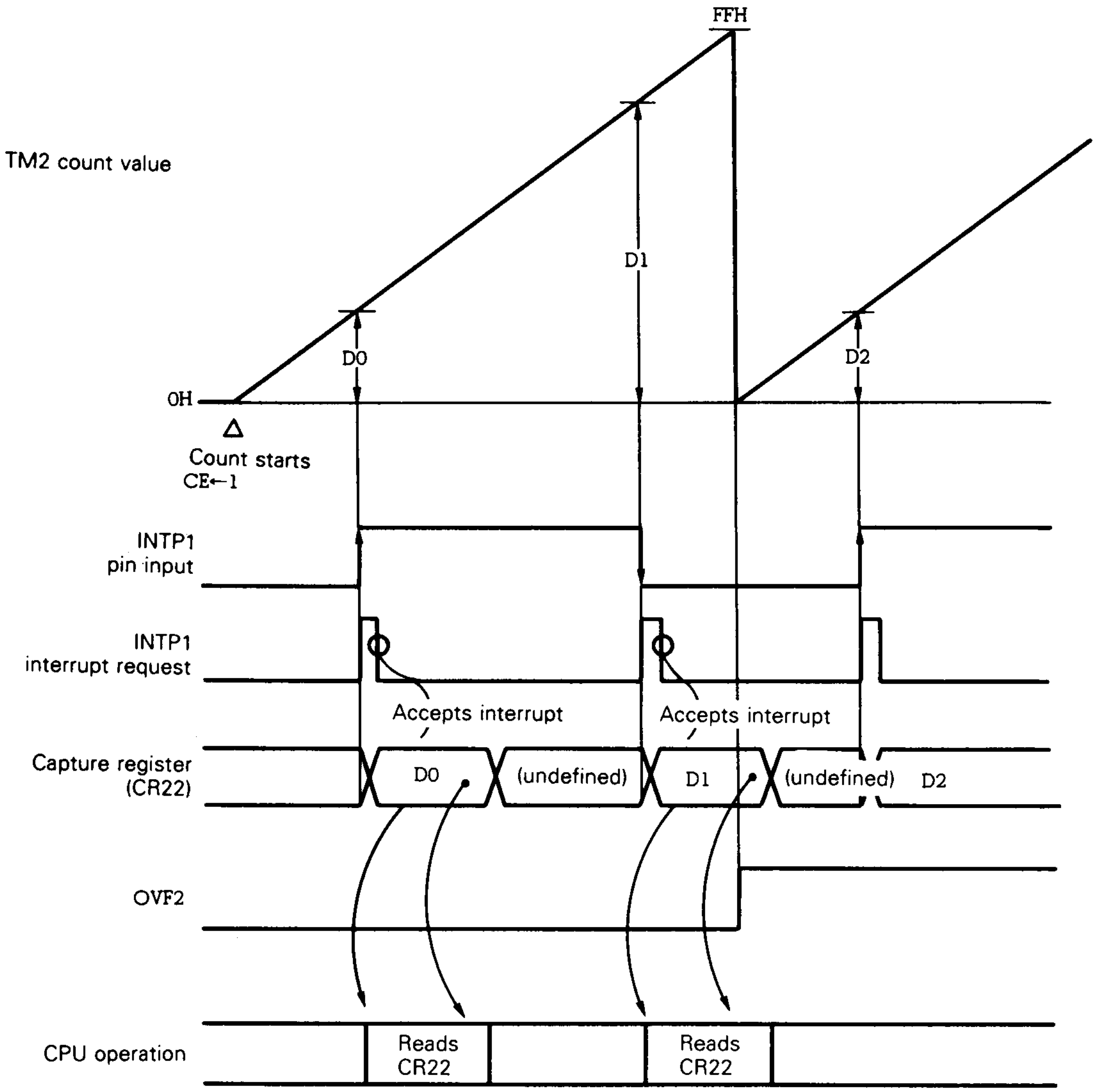
The valid edge for the capture trigger is specified by external interrupt mode register 0 (INTM0). If the capture trigger is specified, so that it is detected at both the rising and falling edges, the width of a pulse input from an external source can be measured. If the capture trigger is detected at either edge, the input pulse cycle can be measured.

For detailed INTM0 register format, refer to Fig. 13-1 in **CHAPTER 13 EDGE DETECTION FUNCTION**.

Cautions: 1. The CR22 register contents become undefined, after they have been read. To use the captured value more than once, save the captured value to a register or memory.

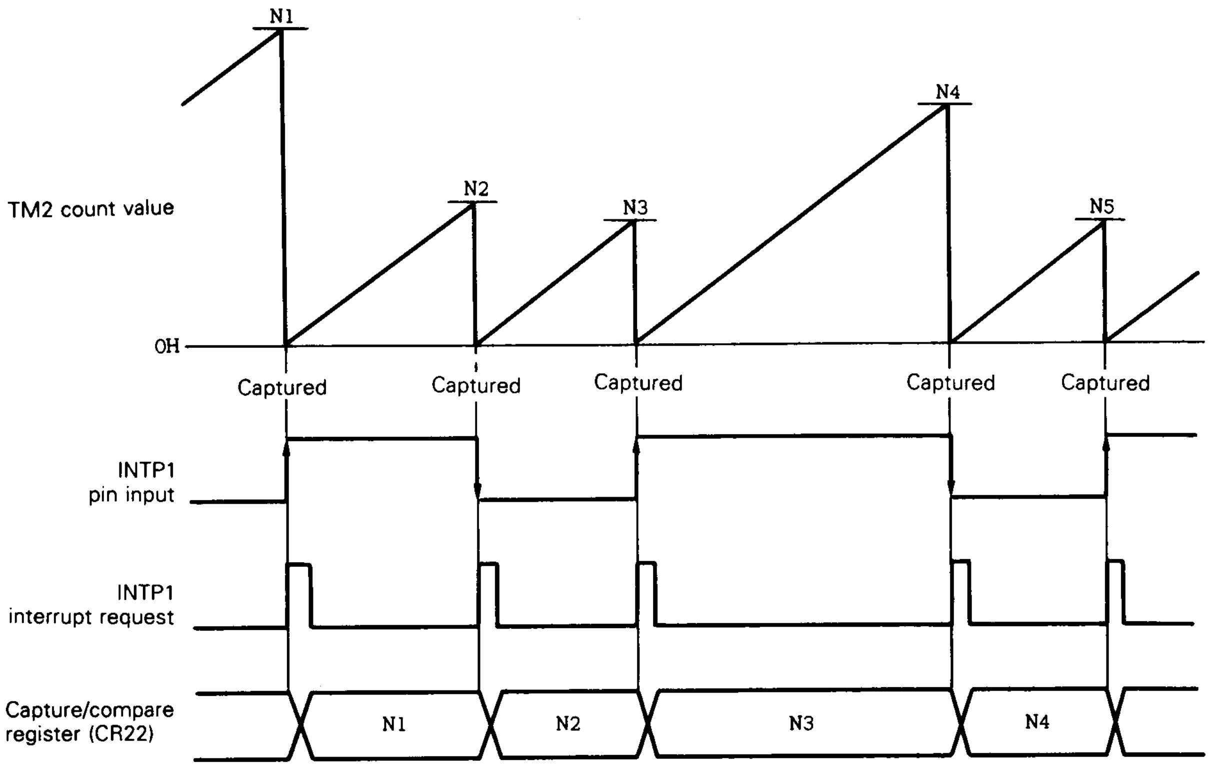
2. Refer to 7.5.4 Notes for using in-circuit emulator.

Fig. 7-89 Capture Operation



Remarks: Dn: count value for TM2 (n = 0, 1, 2, ...)
CLR21 = 0, CLR22 = 0

Fig. 7-90 Clearing TM2 after Its Value Has Been Captured



Remarks: CLR21 = 0, CLR22 = 1

7.3.8 Basic operation for output control circuits

The output control circuits control the levels for timer output pins (TO2 and TO3) by using a coincidence signal of the compare registers. The operations of these circuits are determined by the timer output control register (TOC) and capture/compare control register 2 (CRC2) (see **Table 7-15**). To output signals to the timer output pins (TO2 and TO3), the output pins must be set in the control mode in advance by the PMC3 register.

Table 7-15 Timer output (TO2 and TO3) Operations

TOC				CRC2				TMC1	TO3	TO2
ENTO3	ALV3	ENTO2	ALV2	MOD1	MOD0	CLR22	CLR21	CMD2		
0	0/1	0	0/1	×	×	×	×	×	Fixed to high/low level	Fixed to high/low level
0	0/1	1	0/1	0	0	×*	×	×	Fixed to high/low level	Timer output (low/high active)
1	0/1	0	0/1	0	0	×*	×	×	Toggle output (low/high active)	Fixed to high/low level
1	0/1	1	0/1	0	0	×*	×	×	Toggle output (low/high active)	Toggle output (low/high active)
0	0/1	1	0/1	0	1	0	0	0	Fixed to high/low level	PWM output (high/low active)
1	0/1	0	0/1	0	1	0	0	0	Toggle output (low/high active)	Fixed to high/low level
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	Fixed to high/low level	PWM output (high/low active)
1	0/1	0	0/1	1	0	0	0	0	PWM output (high/low active)	Fixed to high/low level
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	Fixed to high/low level	PPG output (high/low active)
1	0/1	0	0/1	1	1	0	1	0	Toggle output (low/high active)	Fixed to high/low level
1	0/1	1	0/1	1	1	0	1	0	Toggle output (low/high active)	PPG output (high/low active)

***:** Normally, CLR22 is 0 in this case.

Remarks: **1.** The values on the left and the right of "/" in the ALV3 and ALV2 column, respectively, correspond to the statuses on the left and the right of "/" in the TO3 and TO2 column.

2. x's indicate don't care bits and the operation is the same regardless of whether these bits are 0 or 1.

3. Combinations not listed in this table are inhibited.

(1) Basic operation

The output timing for the timer output pins (TO2 and TO3) can be changed as specified by the MOD0, MOD1, and CLR21 bits for the capture/compare control register 2 (CRC2) when the ENTOn (n = 2 or 3) bit for the timer output control register (TOC) is set to 1.

By clearing the ENTOn (n = 2 or 3) bit to 0, the levels for timer output pins (TO2 and TO3) can be fixed. The levels at which the output pins are fixed are determined by the ALVn (n = 2 or 3) bit for the timer output control register (TOC). That is, when the ALVn bit is 0, the output levels are fixed at a high level, while the output levels are made low when the ALVn bit is 1.

(2) Toggle output

Toggle output is an operation mode in which the output levels for the timer output pins are inverted each time the values for the compare registers (CR20 and CR21) coincide with the value for 8-bit timer 2 (TM2). The output level for pin TO2 is inverted when the CR20 value coincides with the TM2 value, and the TO3 output level is inverted when the CR21 value coincides with the TM2 value.

When the 8-bit timer/counter 2 is stopped by resetting the CE2 bit of the TMC1 register to 0, the output level of the timer/counter is retained as is.

Fig. 7-91 Timer Output Operation

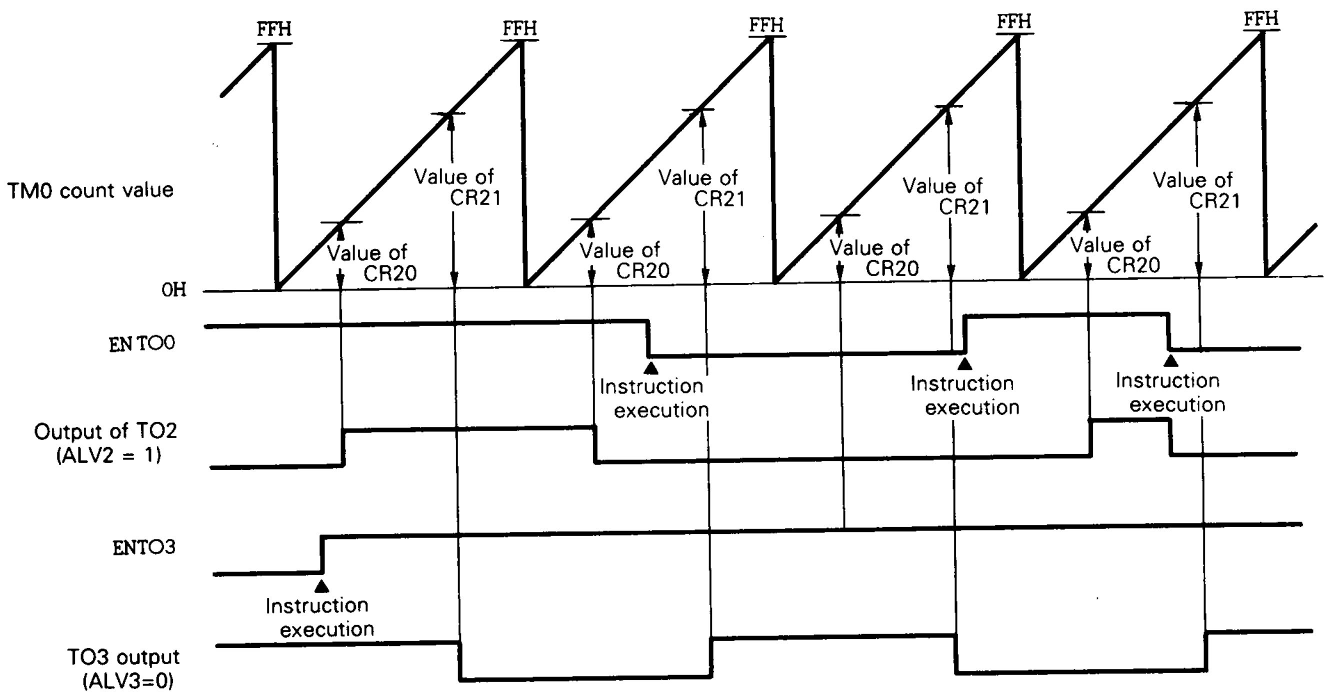


Table 7-16 TO2 and TO3 Toggle Outputs (f_{CLK} = 6 MHz)

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	$2^8 \times 512/f_{CLK}$ (21.8 ms)

Caution: Refer to 7.5.4 Notes for using in-circuit emulator.

7.3.9 PWM output

This is an output mode in which a PWM signal is output, whose cycle consists of a period during which the 8-bit timer (TM2) counts up to the maximum value. The TO2 pulse width is determined by the CR20 value, and the pulse TO3 width is determined by the CR21 value. To use this function, it is necessary to clear the CLR21 bit and the CLR22 bit for the capture/compare control register 2 (CRC2) to 0. Also, clear the CMD2 bit for the timer control register 1 (TMC1) to 0.

The pulse cycle and pulse width are related to each other as follows:

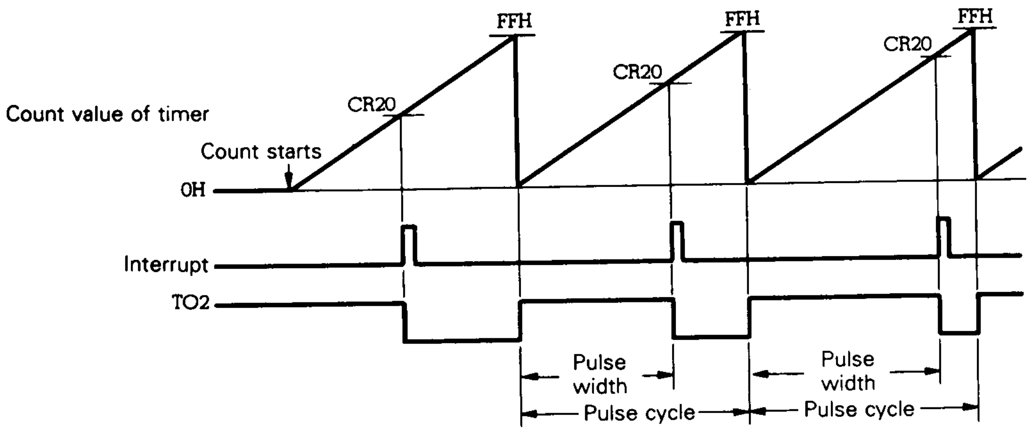
- PWM period = $256/f_{CLK}$
- PWM pulse width = ((set value for compare register*****) × X + 2)/f_{CLK}
  where X = 16, 32, 64, 128, 256, or 512

  ***:** 0 cannot be set in the compare register.

- Duty factor = PWM pulse width/PWM period = [(set value for compare register x X) + 2]/256 × X
  ≒ set value for compare register/256

Caution: The PWM output pulse width is two f_{CLK} clocks longer than the result of the above approximate expression at the active level, and is two f_{CLK} clocks shorter at the inactive level. If high-accuracy output is necessary, or if the count clock is at high speed, take these points into consideration.

Fig. 7-92 PWM Pulse Output



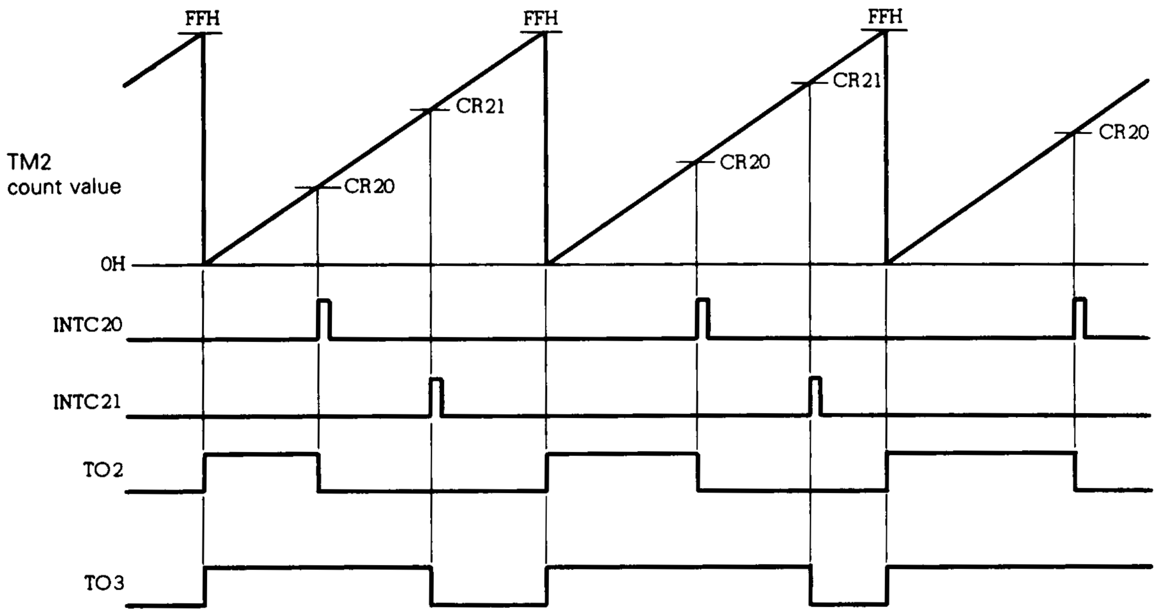
Remarks: ALV2 = 0

Table 7-17 PWM Cycles for TO2 and TO3 (f_{CLK} = 6 MHz)

Count clock	Minimum pulse width [μs]	Cycle [ms]	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

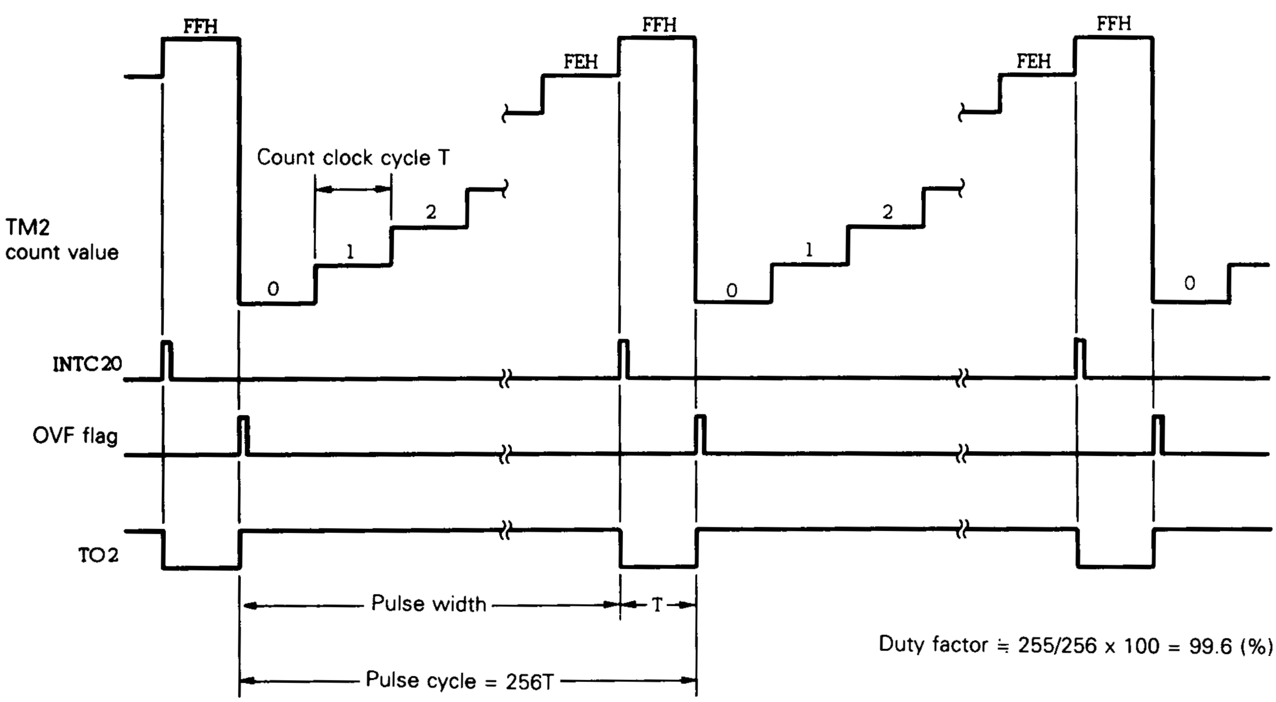
Fig. 7-93 shows a 2-channel PWM output example and Fig. 7-94 shows an example when setting the value FFH in the compare register.

Fig. 7-93 PWM Output Example Using TM2



Remarks: ALV2 = 0, ALV3 = 0

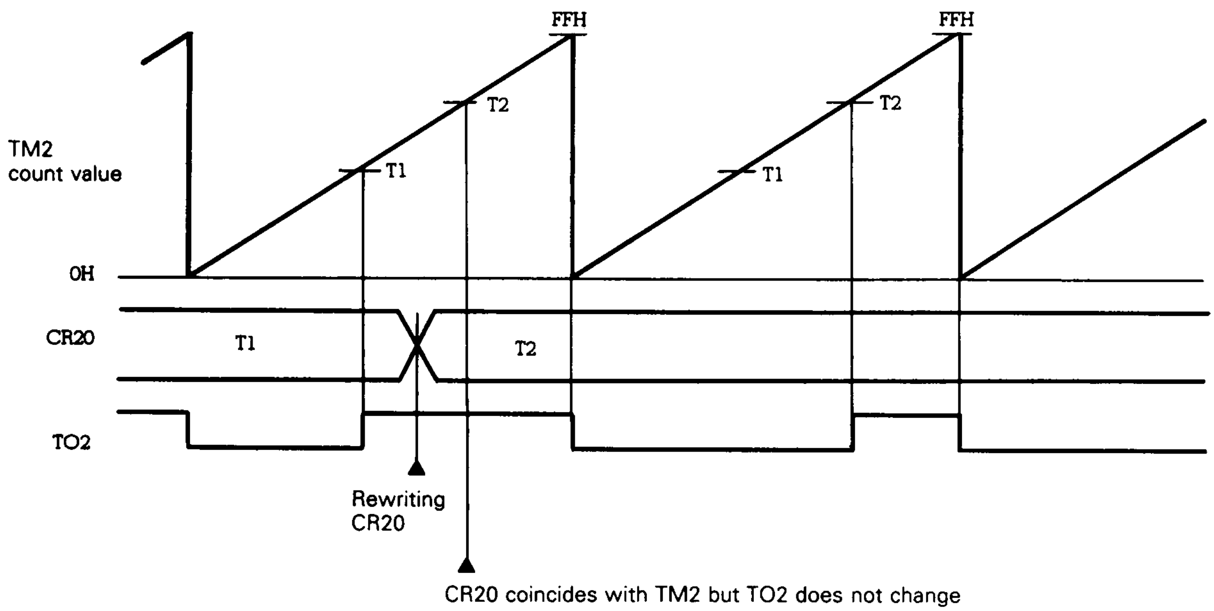
Fig. 7-94 PWM Output Example, When CR20 = FFH



Remarks: ALV2 = 0

Note that, even if the compare registers (CR20 and CR21) values coincide with the 8-bit timer 2 (TM2) value, more than once during one PWM output cycle, the output levels for timer output pins TO2 and TO3 do not change.

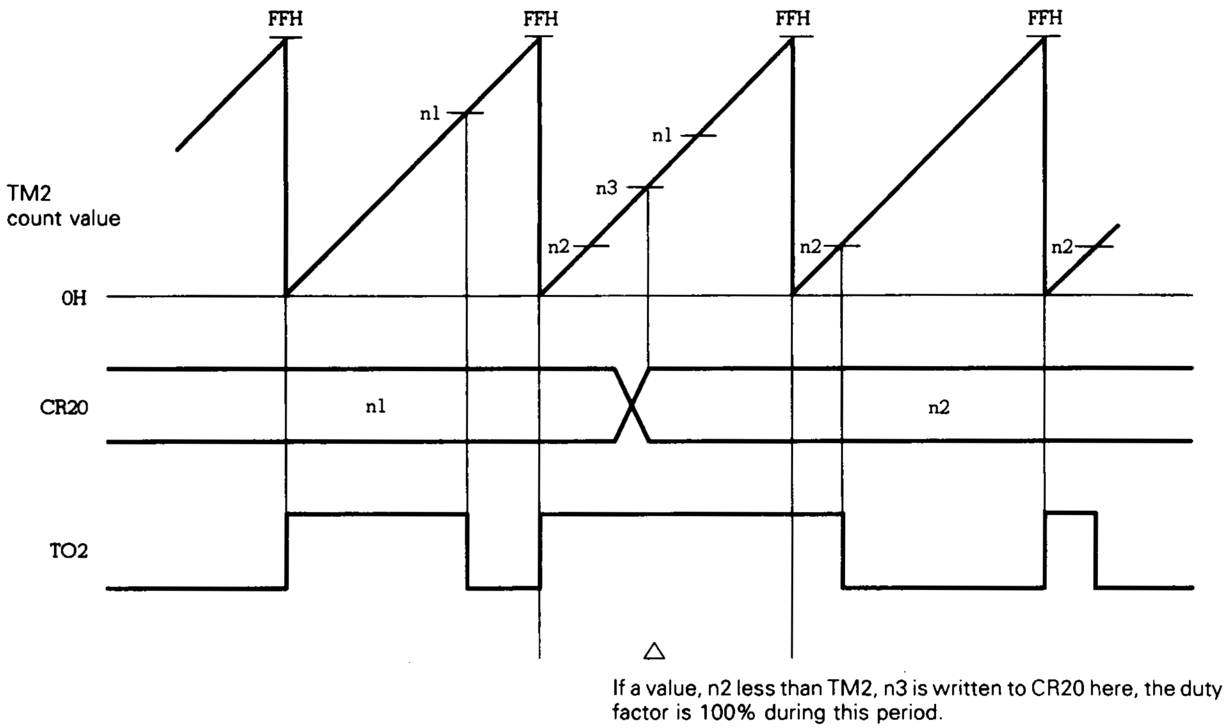
Fig. 7-95 Rewriting Contents for Compare Registers



Remarks: ALV2 = 1

Caution: If values less than the value of 8-bit timer 2 (TM2) are set to compare registers (CR20, CR21), a PWM signal with a duty factor of 100% is output. Rewrite the values of CR20 and CR21 by using an interrupt that occurs when TM2 coincides with a compare register (CR20, CR21).

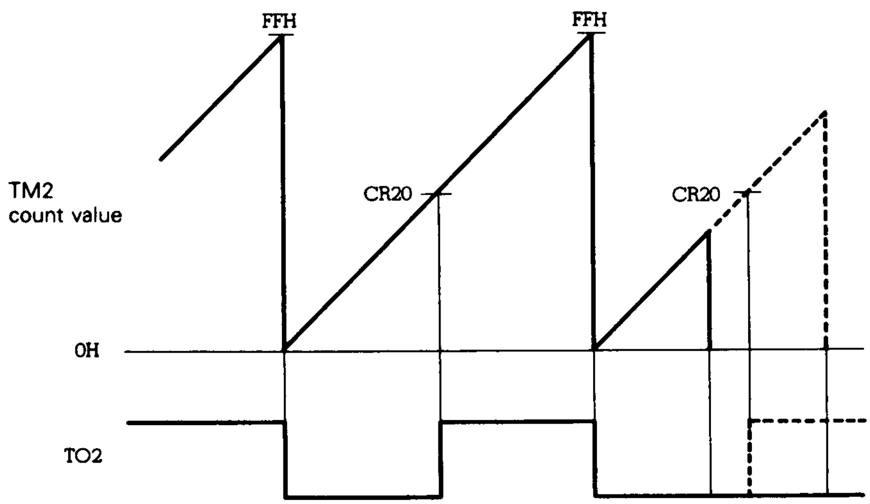
Fig. 7-96 PWM Output Example When Duty Factor Is 100%



Remarks: ALV2 = 0

When 8-bit timer/counter 2 is stopped by resetting the CE2 bit of TMC1 to 0 while the PWM signal is output, the output level is retained as is.

Fig. 7-97 When Stopping 8-bit Timer/Counter 2 during PWM Output



Remarks: ALV 0 = 1

Caution: When the timer output is disabled (ENTOn = 0, n = 2, 3), the output level for the TOn (n = 2, 3) pin is a complement of the value set in ALVn (n = 2, 3). Note, therefore, that the active level is output, when the timer output is disabled with the PWM output function selected.

7.3.10 PPG output

This function is to output square waves whose pulse widths are determined by the compare register CR20 value. The pulse cycle is determined by the compare register CR21 value. Therefore, this function is to vary the PWM cycle of the PWM output. The square waves can be output from the TO2 pin only.

To use this function, set the CLR21 bit for the capture/compare control register (CR22) to 1, and clear the CLR22 bit to 0. Also clear the CMD2 bit for the timer control register 1 (TMC1) to 0.

The pulse cycle and pulse width are related to each other as follows:

- PPG period = (set value for compare register CR21 + 1) × x/f_{CLK} where x = 16, 32, 64, 128, 256, or 512

- PPG pulse width = ((set value for compare register CR20) × x + 2)/f_{CLK} ≒ set value for CR20 × x/f_{CLK}

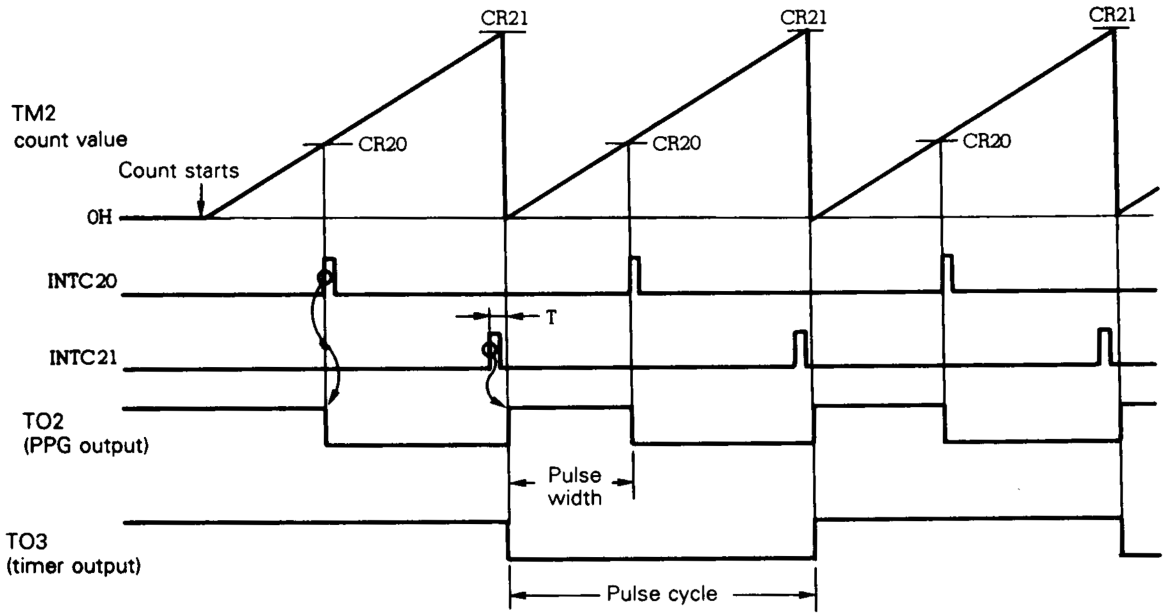
  where CR20 ≦ CR21

- Duty factor = PPG pulse width/PPG period = set value for CR20 × x + 2/(set value of CR21 + 1) × x
  ≒ set value for CR20/set value for CR21 + 1

Caution: The PPG output pulse width is two f_{CLK} clocks longer than the result of the above approximate expression at the active level, and is two f_{CLK} clocks shorter at the inactive level. If high-accuracy output is necessary, or if the count clock is at high speed, take these points into consideration.

Figure 7-98 shows a PPG output example, using the 8-bit timer 2 (TM2). Figure 7-99 shows an example, where CR20 = CR21, and Figure 7-100 is an example, where CR20 = 00H.

Fig. 7-98 PPG Output Example, Using TM2



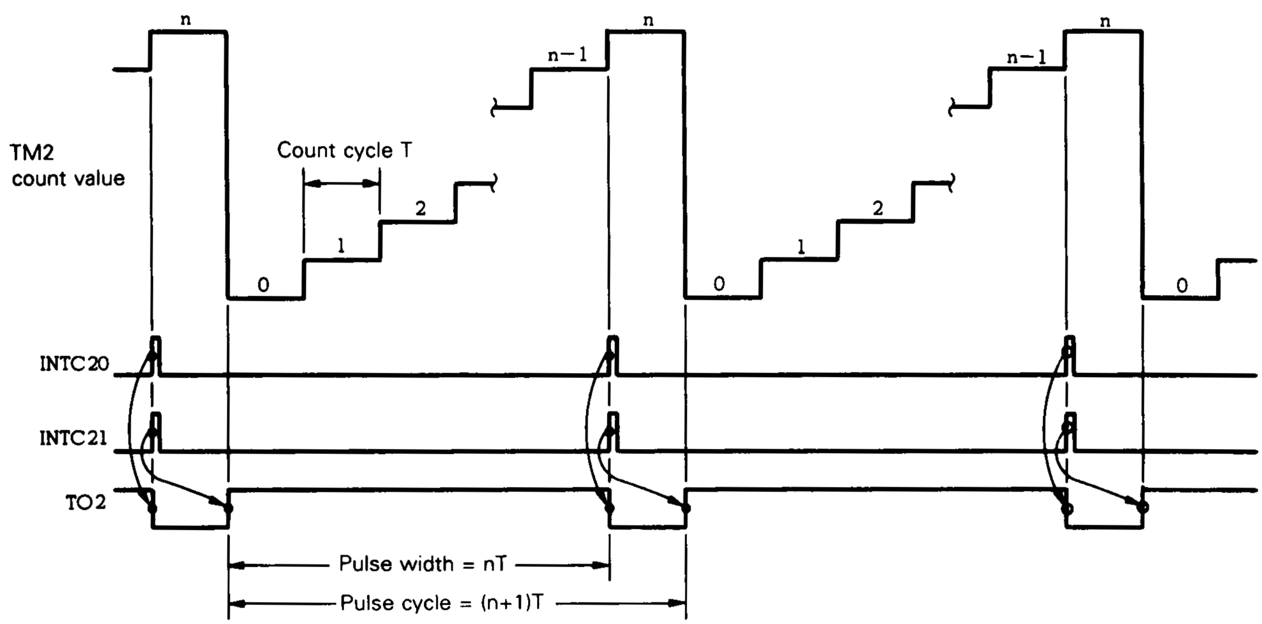
Remarks: ALV2 = 0, ALV3 = 0

Table 7-18 PPG Output for TO2 (f_{CLK} = 6 MHz)

Count clock	Minimum pulse*	PPG cycle	PPG frequency
f_{CLK}/16	2.67 μs	5.33 μs - 683 μs	187.5 kHz - 1.46 kHz
f_{CLK}/32	5.33 μs	10.7 μs - 1.37 ms	93.75 kHz - 732 Hz
f_{CLK}/64	10.7 μs	21.3 μs - 2.73 ms	46.9 kHz - 366 Hz
f_{CLK}/128	21.3 μs	42.7 μs - 5.46 ms	23.4 kHz - 183 Hz
f_{CLK}/256	42.7 μs	85.3 μs - 10.9 ms	11.7 kHz - 91.6 Hz
f_{CLK}/512	85.3 μs	171 μs - 21.8 ms	5.86 kHz - 45.8 Hz

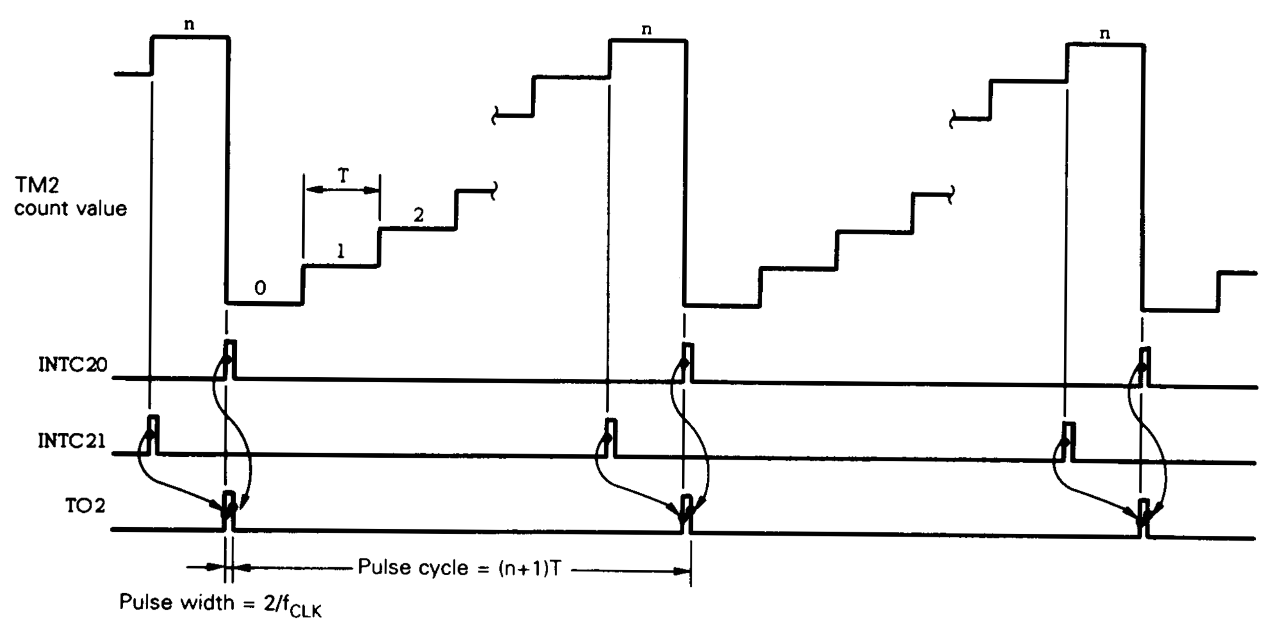
***:** Except when CR20 = 0

Fig. 7-99 PPG Output Example, When CR20 = CR21



Remarks: ALV2 = 0

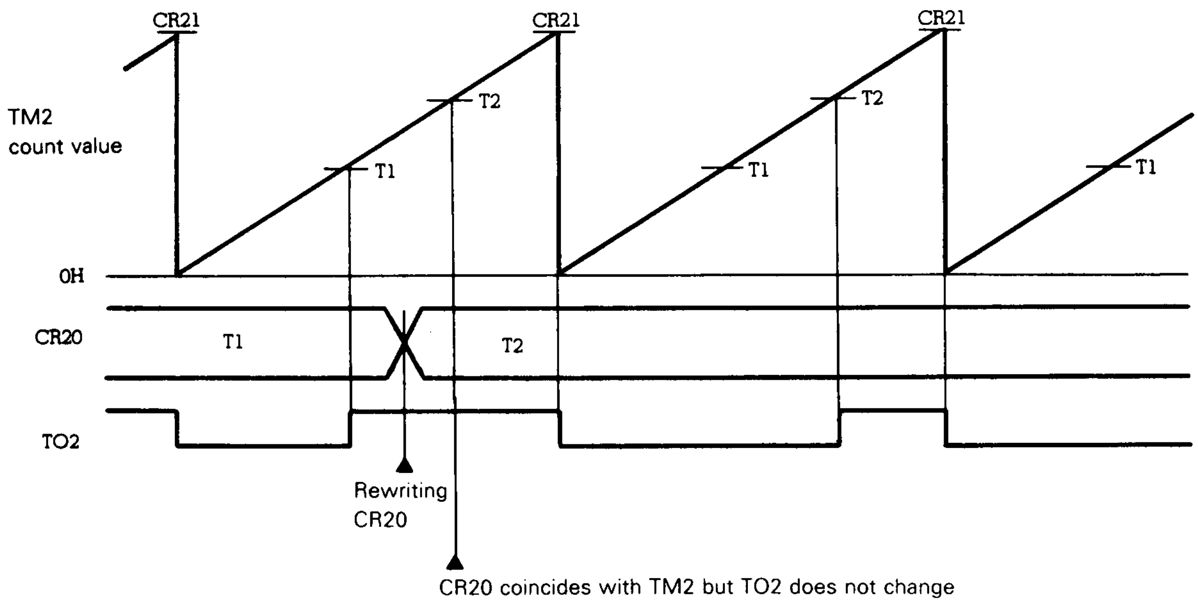
Fig. 7-100 PPG Output Example, When CR20 = 00H



Remarks: ALV2 = 0

Note that, even if the compare register (CR20) values coincide with the 8-bit timer 2 (TM2) value more than once during one PPG output cycle, the output level for timer output pin TO2 does not change.

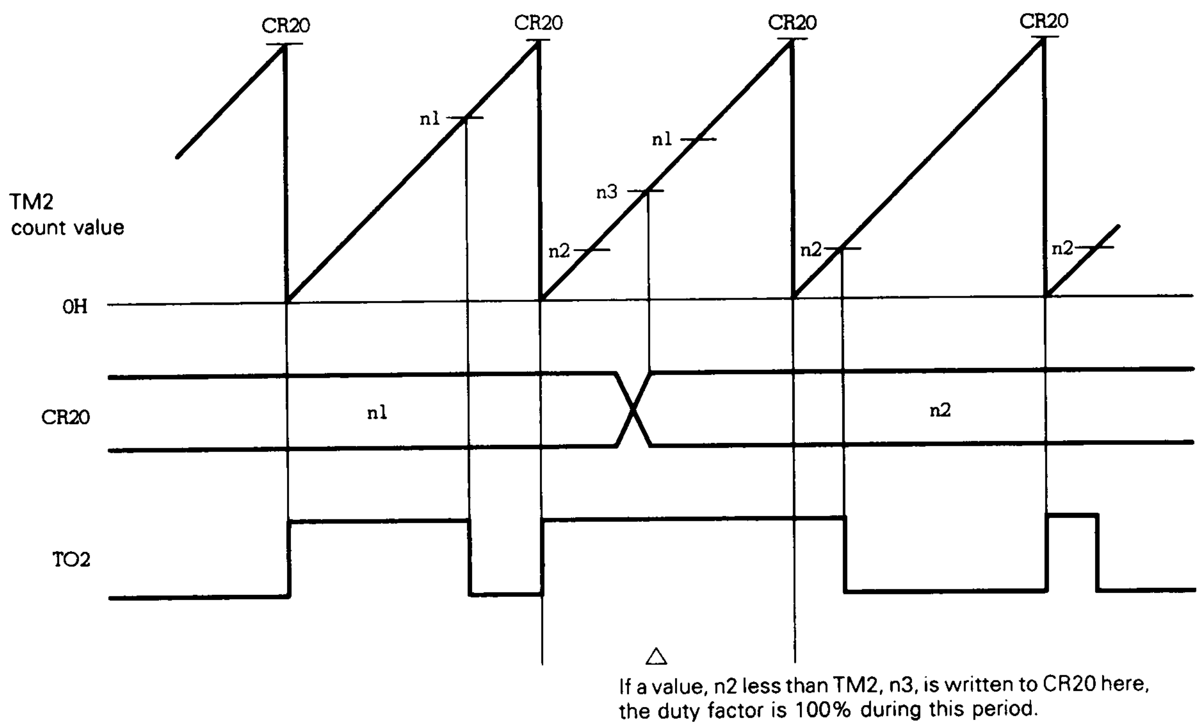
Fig. 7-101 Rewriting Compare Register



Remarks: ALV2 = 1

Caution 1: If a value less than that of 8-bit timer o (TM2) is written to compare register CR20 before the value of the CR20 coincides with that of TM2, the duty factor of the PPG cycle becomes 100%. To rewrite the value of CR20, use an interrupt that occurs when the value of CR20 coincides with that of TM2.

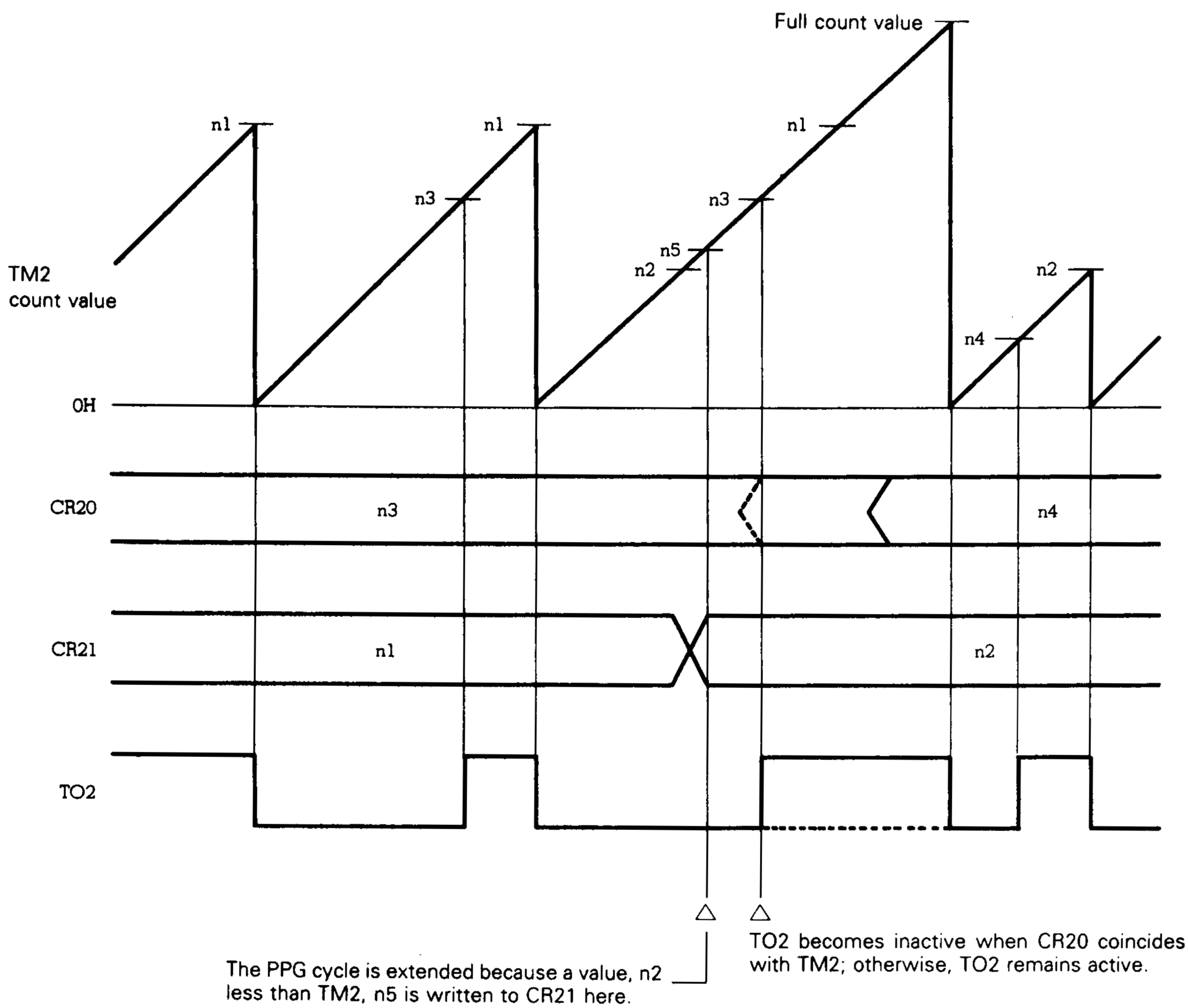
Fig. 7-102 PPG Output Example When Duty Factor is 100%



Remarks: ALV2 = 0

Caution 2: When the value of compare register (CR21) is to be changed to a value less than the current value, if a value less than the value of 8-bit timer 0 (TM2) is set to CR21, the PPG cycle is extended to the time required for full count of TM2. At this time, the output level becomes inactive until TM2 overflows and is cleared to 0 if the value of CR21 is rewritten after its value has coincided with TM2, returning to the normal PPG output. If the value of CR21 is changed before CR20 coincides with TM2, the active level is output until CR20 coincides with TM2. If CR20 coincides with TM0 before TM2 overflows and is cleared to 0, the inactive level is output at that point, and the active level is output when TM20 is cleared to 0, and the normal PPG output is restored.
Rewrite the value of CR21 by using an interrupt that occurs when CR21 coincides with TM2.

Fig. 7-103 PPG Output Example When Output Cycle Is Extended

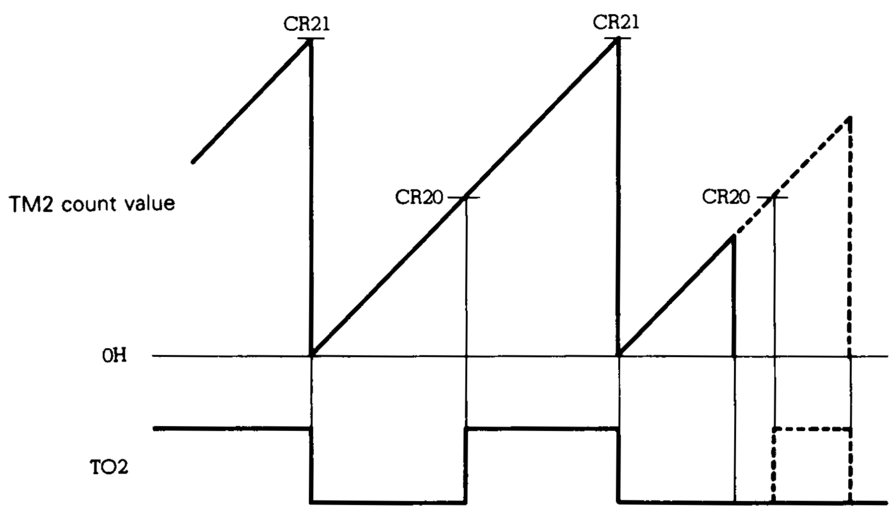


Remarks: ALV2 = 1

Caution 3: If the PPG cycle is extremely short in respect to the time required for accepting an interrupt, measures described in Notes 1 and 2 are not effective.
Take other measures (such as masking all the interrupts and polling the interrupt flags through software).

When 8-bit timer/counter 2 is stopped by resetting the CE2 bit of TMC1 to 0 while the PPG signal is output, the active level is output regardless of the output level of the timer/counter.

Fig. 7-104 When Stopping 8-bit Timer/Counter 2 during PPG Output



Caution: When the timer output is disabled (ENTOn = 0, n = 2, 3), the output level for the TOn (n = 2, 3) pin is a complement of the value set in ALVn (n = 2, 3). Note, therefore, that the active level is output, when the timer output is disabled with the PPG output function selected.

7.3.11 Application examples

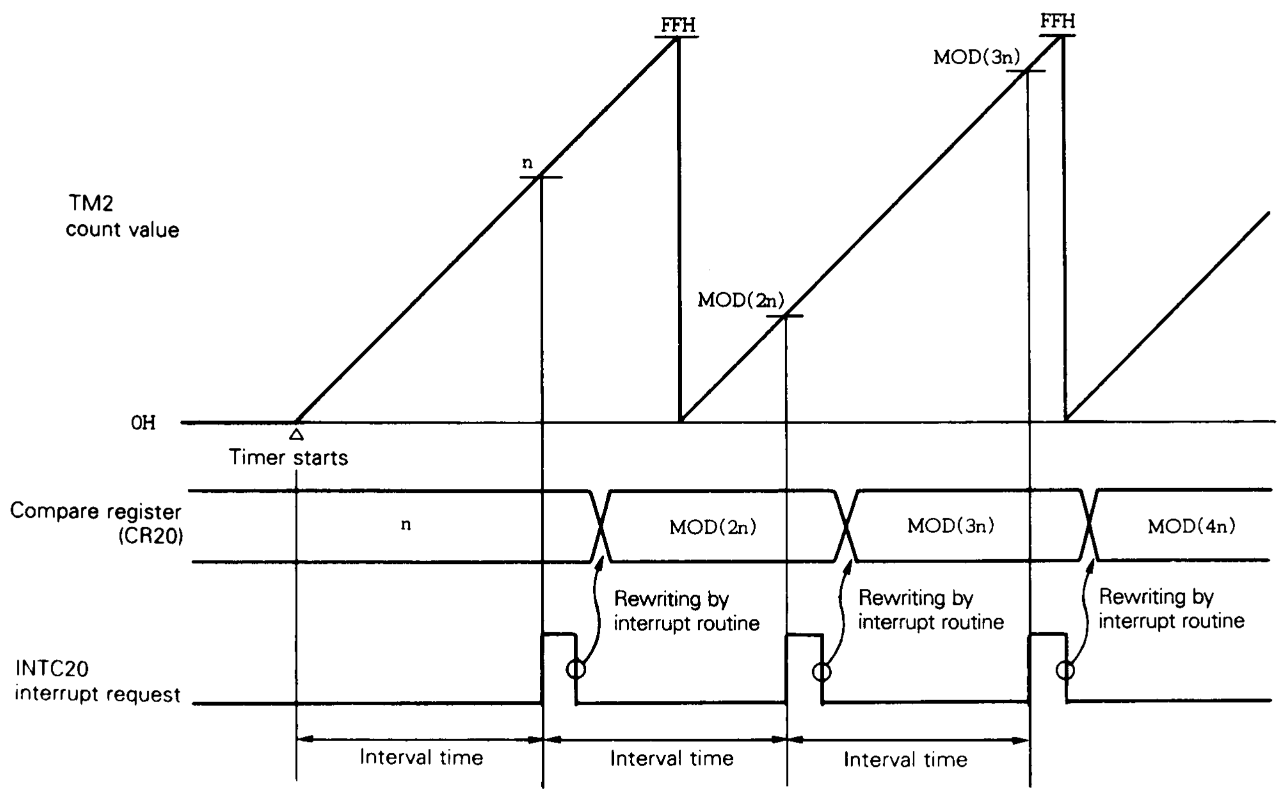
(1) Operation as interval timer (1)

When the 8-bit timer 2 (TM2) is used as a free-running timer and when a fixed value is added to the compare registers (CR20 and CR21) contents by an interrupt routine, the timer operates as an interval timer, whose cycle is determined by the value to be added to the compare register contents (see **Fig. 7-105**).

Since the 8-bit timer 2 (TM2) is provided with two compare registers, the timer can operate as an interval timer having two cycles.

Fig. 7-106 shows the control register contents. Fig. 7-107 shows the procedure used to set the control register contents. Figure 7-108 shows the processing performed by the interrupt routine.

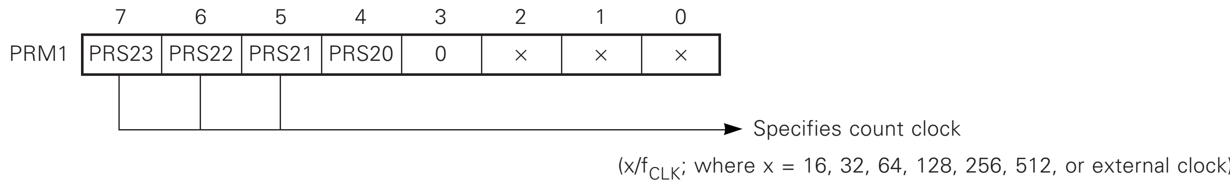
Fig. 7-105 Interval Timer Operation (1) Timing



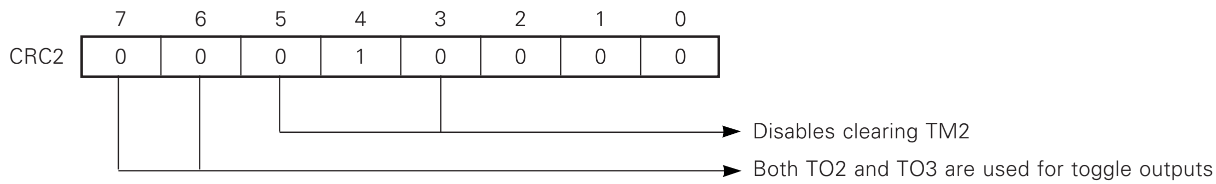
Remarks: Interval time = $n \times x/f_{CLK}$, where $1 \leqq n \leqq$ FFH x = 16, 32, 64, 128, 256, 512

Fig. 7-106 Control Register Contents for Interval Timer Operation (1)

(a) Prescaler mode register 1 (PRM1)



(b) Capture/compare control register 2 (CRC2)



(c) Timer control register 1 (TMC1)

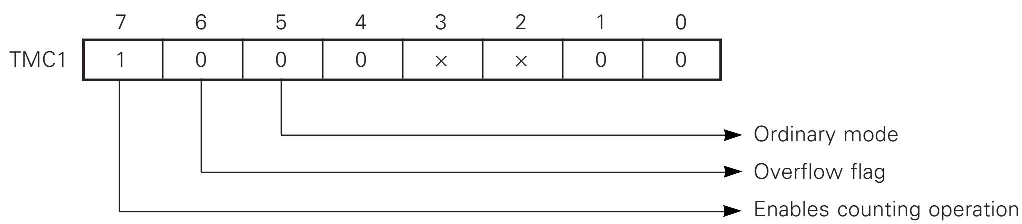


Fig. 7-107 Interval Timer Operation (1) Setting Procedure

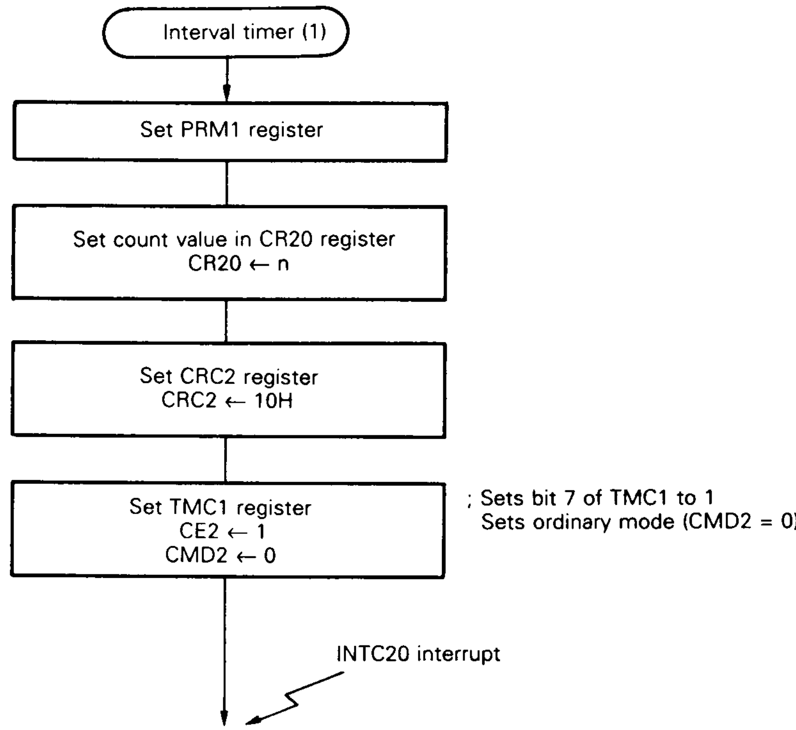
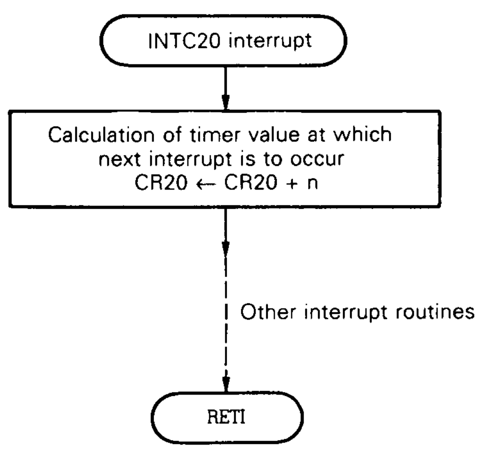


Fig. 7-108 Interrupt Request Processing for Interval Timer Operation (1)

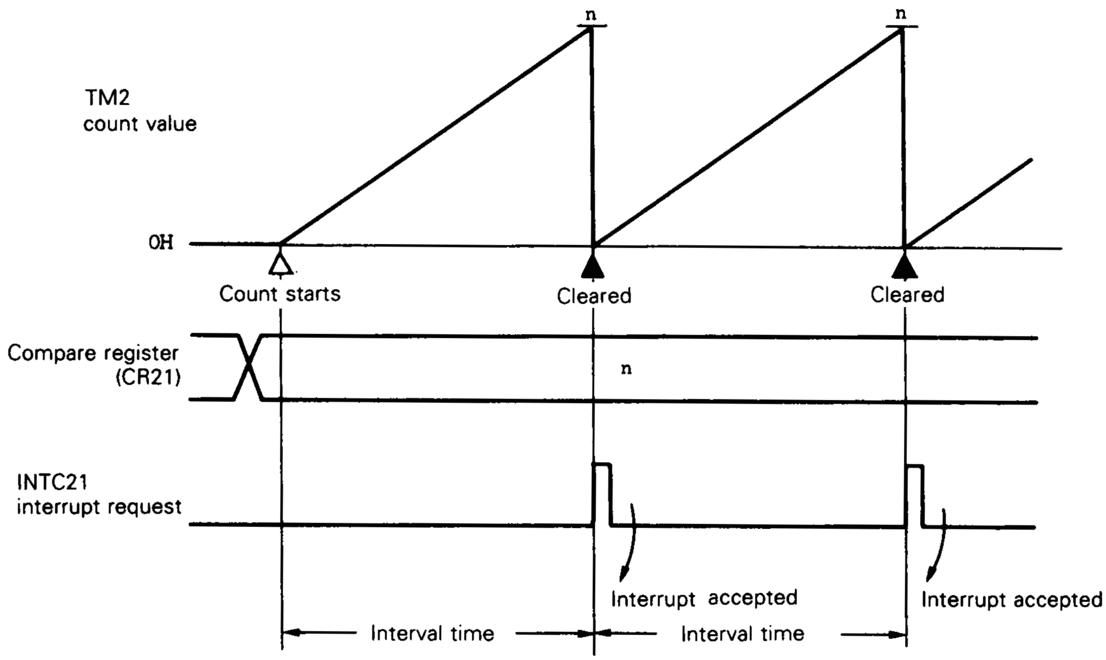


(2) Operation as interval timer (2)

In this example, the 8-bit timer operates as an interval timer that repeatedly generates an interrupt at predetermined time intervals (see **Fig. 7-109**).

Fig. 7-110 shows the control register contents, while Fig. 7-111 shows the procedure used to set the register contents.

Fig. 7-109 Interval Timer Operation (2) Timing



Remarks: Interval time = (n + 1) × x/f_{CLK}; 0 ≦ n ≦ FFH
x = 16, 32, 64, 128, 256, 512

Fig. 7-110 Control Register Contents for Interval Timer Operation (2)

(a) Prescaler mode register 1 (PRM1)

	7	6	5	4	3	2	1	0
PRM1	PRS23	PRS22	PRS21	PRS20	0	×	×	×

PRS23–PRS20 → Specifies count clock
(x/f_{CLK}; where x = 16, 32, 64, 128, 256, 512, or external clock)

(b) Capture/compare control register 2 (CRC2)

	7	6	5	4	3	2	1	0
CRC2	0	0	0	1	1	0	0	0

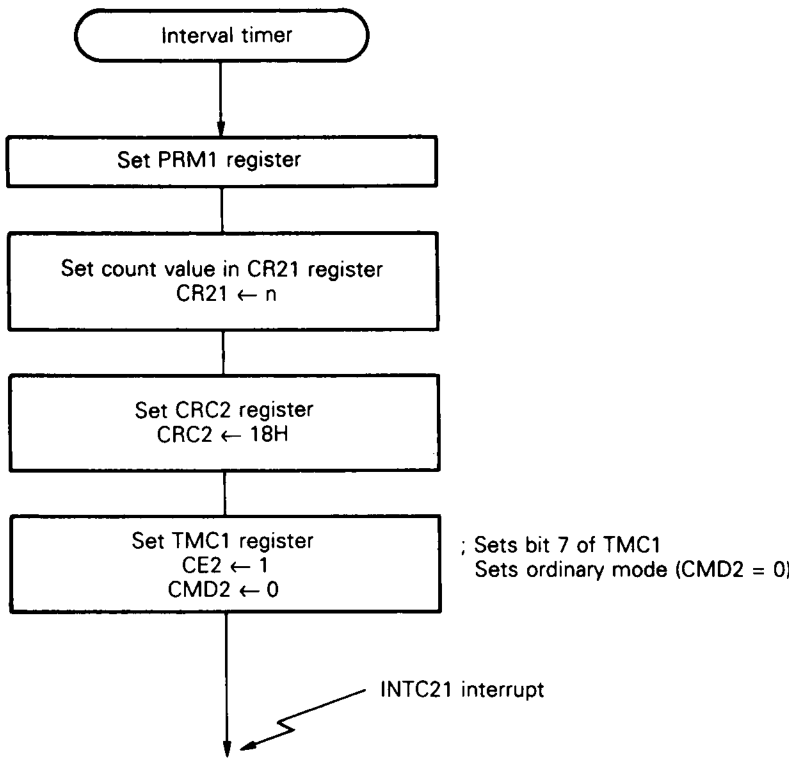
Bit 3 → Enables clearing TM2, when CR21 coincides with TM2
Bit 5 → Disables clearing TM2, when TM2 captured
Bits 7, 6 → Both TO2 and TO3 are used for toggle outputs

(c) Timer control register 1 (TMC1)

	7	6	5	4	3	2	1	0
TMC1	1	0	0	0	×	×	0	0

Bit 5 → Ordinary mode
Bit 6 → Overflow flag
Bit 7 → Enables count operation

Fig. 7-111 Interval Timer Operation (2) Setting Procedure



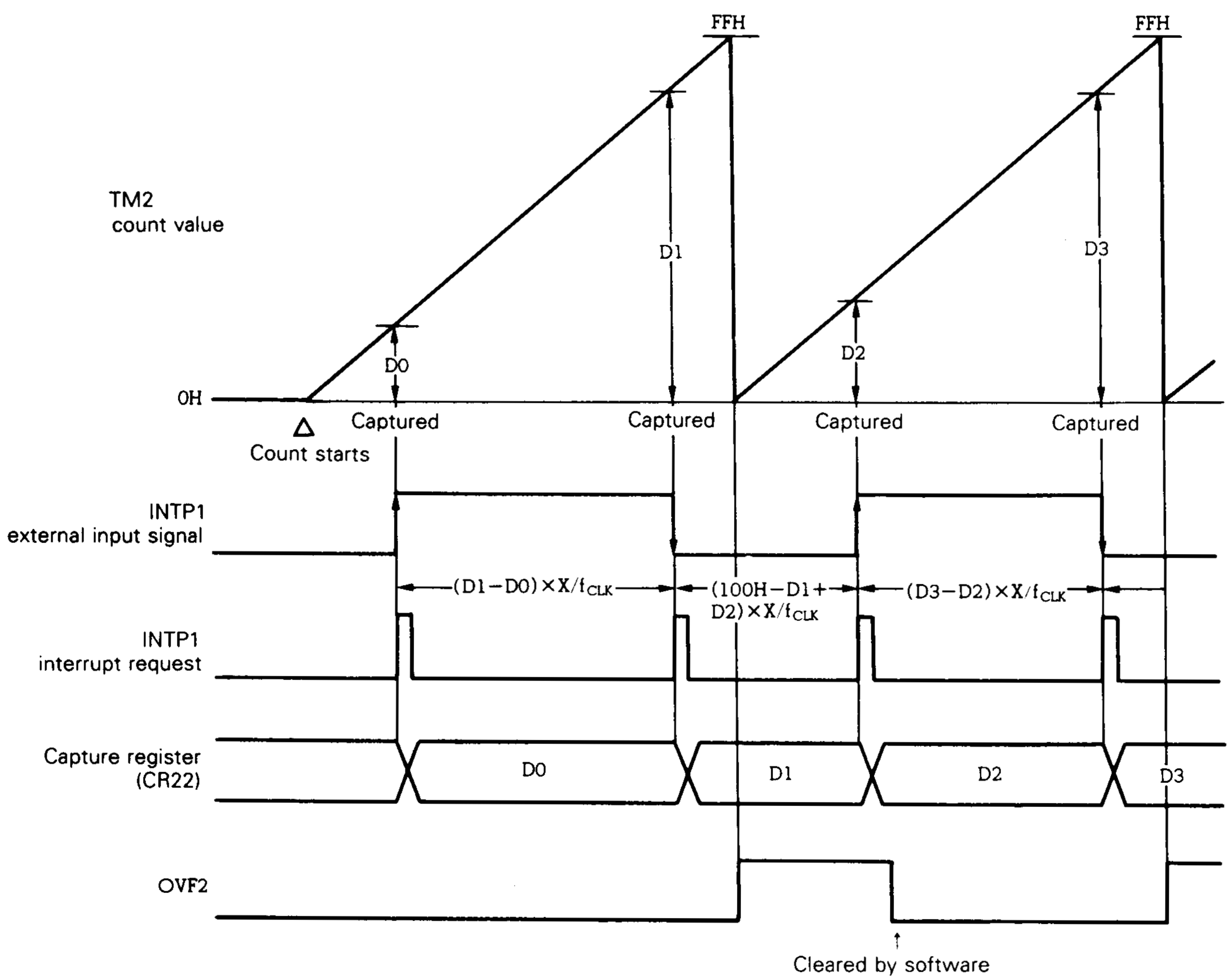
(3) Operation to measure pulse width

The high-level or low-level width for an external pulse input to external interrupt request pin INTP1 can be measured by the 8-bit timer.

The width of the pulse input to the INTP1 pin must be at least 12 system clocks (2 μs at f_{CLK} = 6 MHz), regardless of whether the level of the pulse is high or low; otherwise, the valid edge of the pulse cannot be detected and captured.

As shown in Fig. 7-112, the current count value for the 8-bit timer (TM2) is captured to capture register CR22, in synchronization with the valid edge for the INTP1 pin (the valid edge is specified to be both the rising and falling edges). The timer value is then retained in the capture register. To measure the pulse width, the difference between the count value for TM2 (Dn), which has been captured to and retained in the CR22 register, when the nth valid edge has been detected, and the count value for TM1 (Dn-1), when the n-1th valid edge has been detected, is calculated. This difference is then multiplied by the number of count clocks. Fig. 7-113 shows the control register contents, while Fig. 7-114 shows the procedure used to set the control register contents.

Fig. 7-112 Pulse Width Measurement Timing



Remarks: Dn: Count value for TM2 (n = 0, 1, 2,...)

Fig. 7-113 Control Register Contents for Pulse Width Measurement

(a) Prescaler mode register 1 (PRM1)

	7	6	5	4	3	2	1	0
PRM1	PRS23	PRS22	PRS21	PRS20	0	×	×	×

PRS23–PRS20 → Specifies count clock (x/f_{CLK}, x = 16, 32, 64, 128, 256, 512 or external clock)

(b) Capture/compare control register 2 (CRC2)

	7	6	5	4	3	2	1	0
CRC2	0	0	0	1	0	0	0	0

Bits 5, 3 → Disables clearing TM2

(c) Timer control register 1 (TMC1)

	7	6	5	4	3	2	1	0
TMC1	1	0	0	0	×	×	0	0

Bit 5 → Ordinary mode
Bit 6 → Overflow flag
Bit 7 → Enables count operation

(d) External interrupt mode register 0 (INTM0)

	7	6	5	4	3	2	1	0
INTM0	×	×	1	1	×	×	0	×

Bit 5 → Specifies both rising and falling edges for INTP1 input as valid

Fig. 7-114 Pulse Width Measurement Setting Procedure

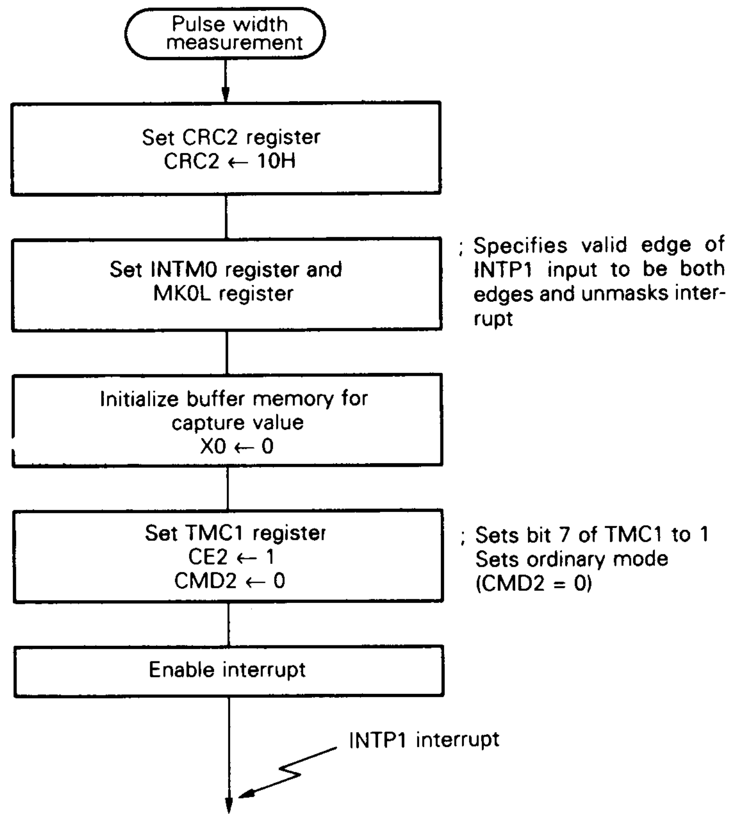
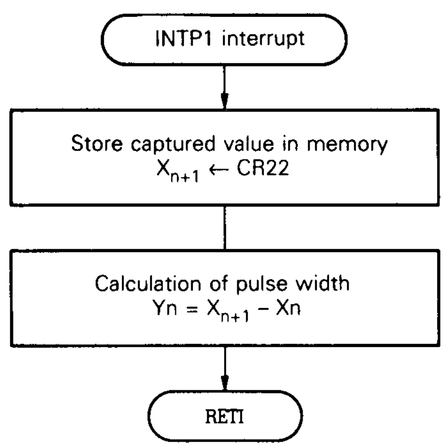


Fig. 7-115 Interrupt Request Processing to Calculate Pulse Width



(4) PWM output operation

The PWM output function is to output a pulse with a duty factor determined by the value set in the compare register (see **Fig. 7-116**).

The duty factor is 1/256 to 255/256 and can be changed in 1/256 units.

Since one 8-bit timer 2 (TM2) is provided with two compare registers, two kinds of PWM signals can be output.

Figure 7-117 shows the set contents for the control register, Fig. 7-118 shows the setting procedure, and Fig. 7-119 shows the procedure used to change the duty factor.

Fig. 7-116 Example for PWM Signal Output by 8-bit Timer/Counter 2

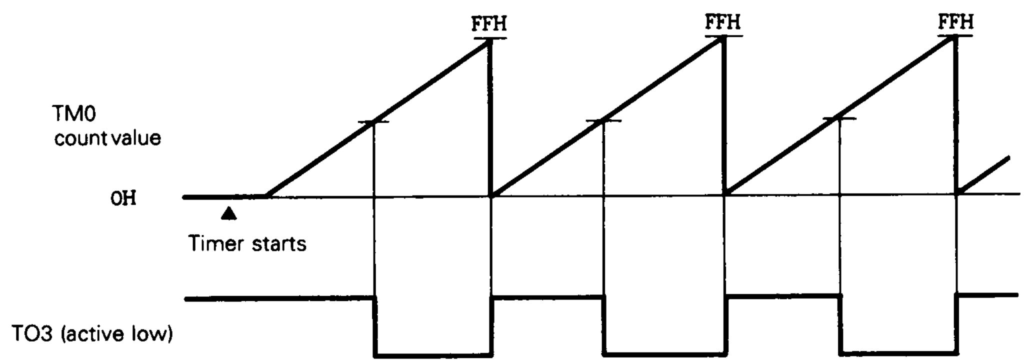
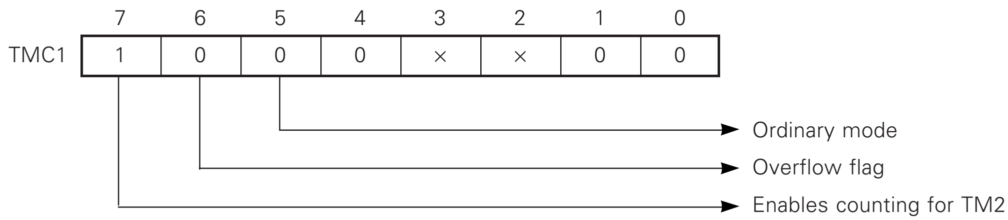
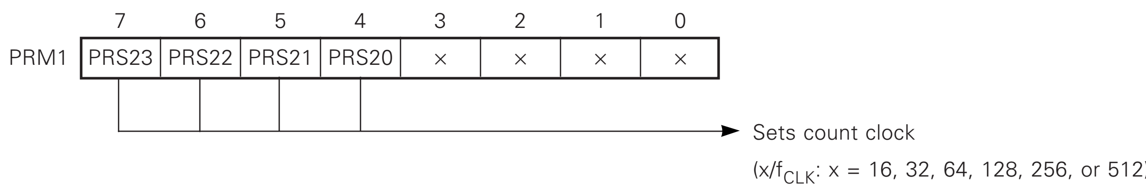


Fig. 7-117 Control Register Contents Set for PWM Output Operation

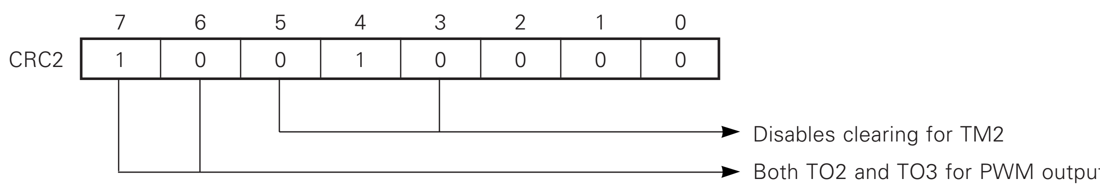
(a) Timer control register 1 (TMC1)



(b) Prescaler mode register 1 (PRM1)



(c) Capture/compare control register 2 (CRC2)



(d) Timer output control register (TOC)

	7	6	5	4	3	2	1	0
TOC	1	0	×	×	×	×	×	×

- Bit 6 (0) → TO3 is high-active PWM signal output
- Bit 7 (1) → Enables PWM output for TO3

(e) Port 3 mode control register (PMC3)

	7	6	5	4	3	2	1	0
PMC3	1	×	×	×	×	×	×	×

- Bit 7 (1) → Sets P37 pin for TO3 output

Fig. 7-118 PWM Output Setting Procedure

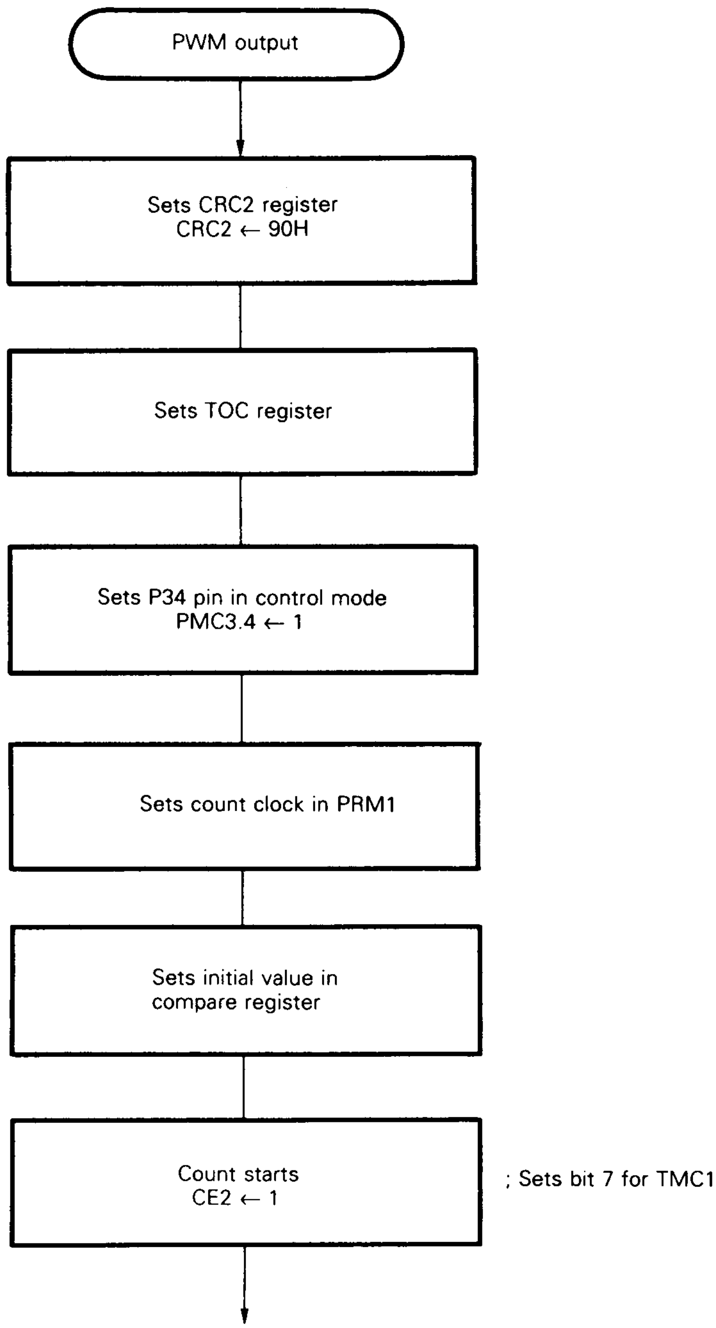
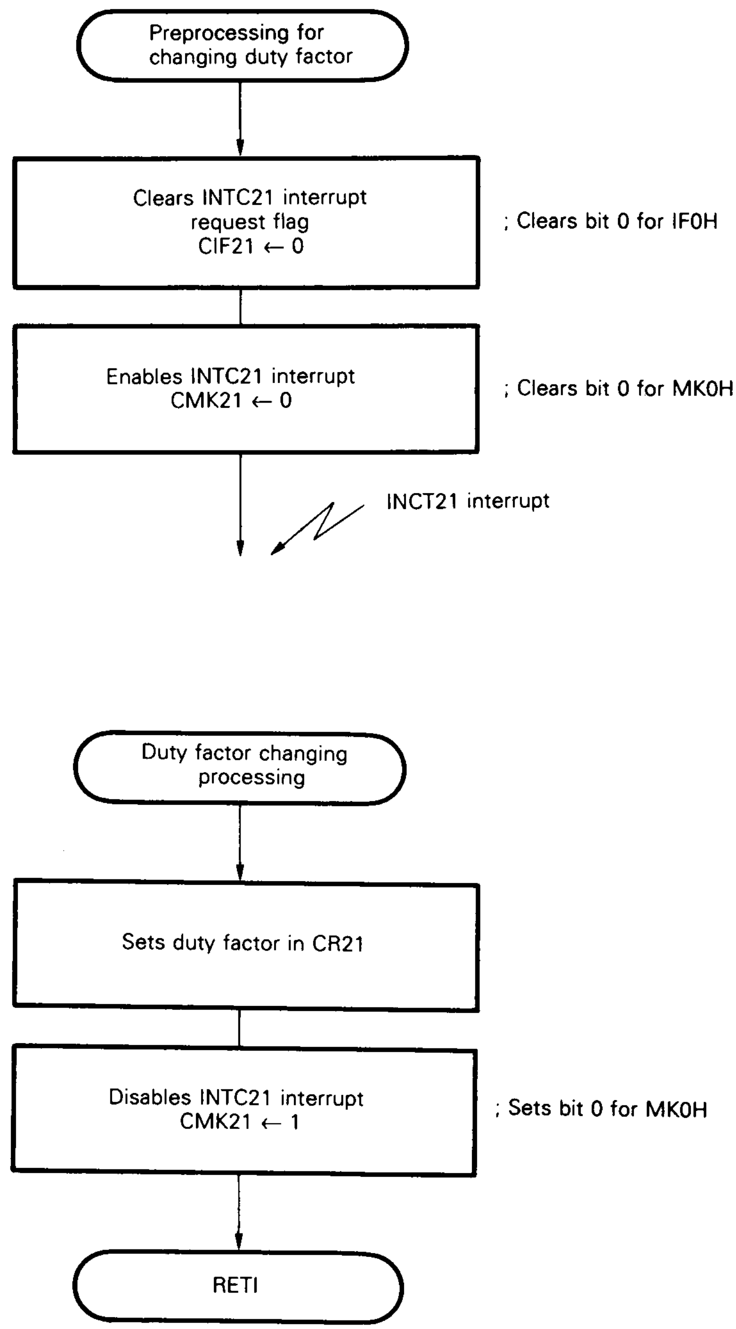


Fig. 7-119 Changing Duty Factor of PWM Output



(5) PPG output operation

The PPG output function is to output a pulse, whose period and duty factor are determined by a value set in the compare register (**Fig. 7-120**).

Figure 7-121 shows the set contents for the control register, Fig. 7-122 shows the setting procedure, and Fig. 7-123 shows the procedure used to change the duty factor.

Fig. 7-120 PPG Signal Output Example of 8-bit Timer/Counter 2

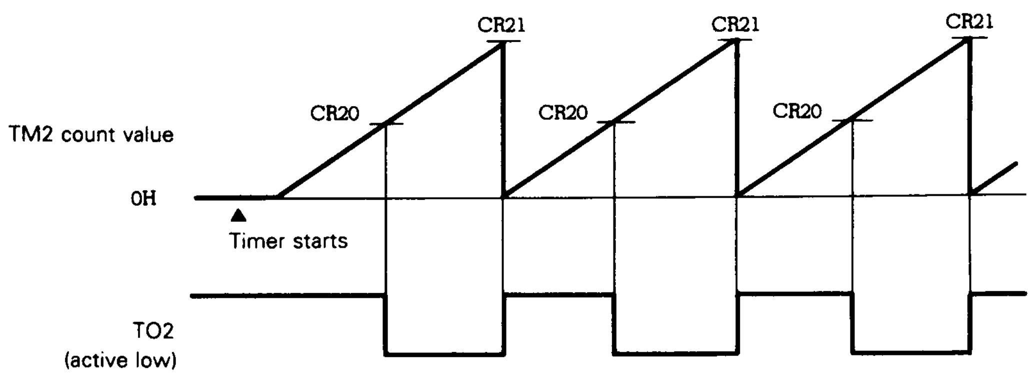


Fig. 7-121 Control Register Contents Set for PPG Output Operation

(a) Timer control register 1 (TMC1)

	7	6	5	4	3	2	1	0
TMC1	1	0	0	0	×	×	0	0

- Bit 5 → Ordinary mode
- Bit 6 → Overflow flag
- Bit 7 → Enables counting for TM2

(b) Prescaler mode register 1 (PRM1)

	7	6	5	4	3	2	1	0
PRM1	PRS23	PRS22	PRS21	PRS20	×	×	×	×

- Bits 7 to 4 → Sets count clock (x/f_{CLK}: x = 16, 32, 64, 128, 256, or 512)

(c) Capture/compare control register 2 (CRC2)

	7	6	5	4	3	2	1	0
CRC2	1	1	0	1	1	0	0	0

- Bit 3 → Clears when TM2 coincides with CR21
- Bit 5 → Disables clearing when TM2 is captured by CR22 register
- Bits 7, 6 → TO2 for PPG output

(d) Timer output control register (TOC)

	7	6	5	4	3	2	1	0
TOC	×	×	1	0	×	×	×	×

- Bit 4 (0) → TO2 for active-high PPG signal output
- Bit 5 (1) → Enables PPG output for TO2

(e) Port 3 mode control register (PMC3)

	7	6	5	4	3	2	1	0
PMC3	×	1	×	×	×	×	×	×

- Bit 6 (1) → Sets P36 pin for TO2 output

Fig. 7-122 PPG Output Setting Procedure

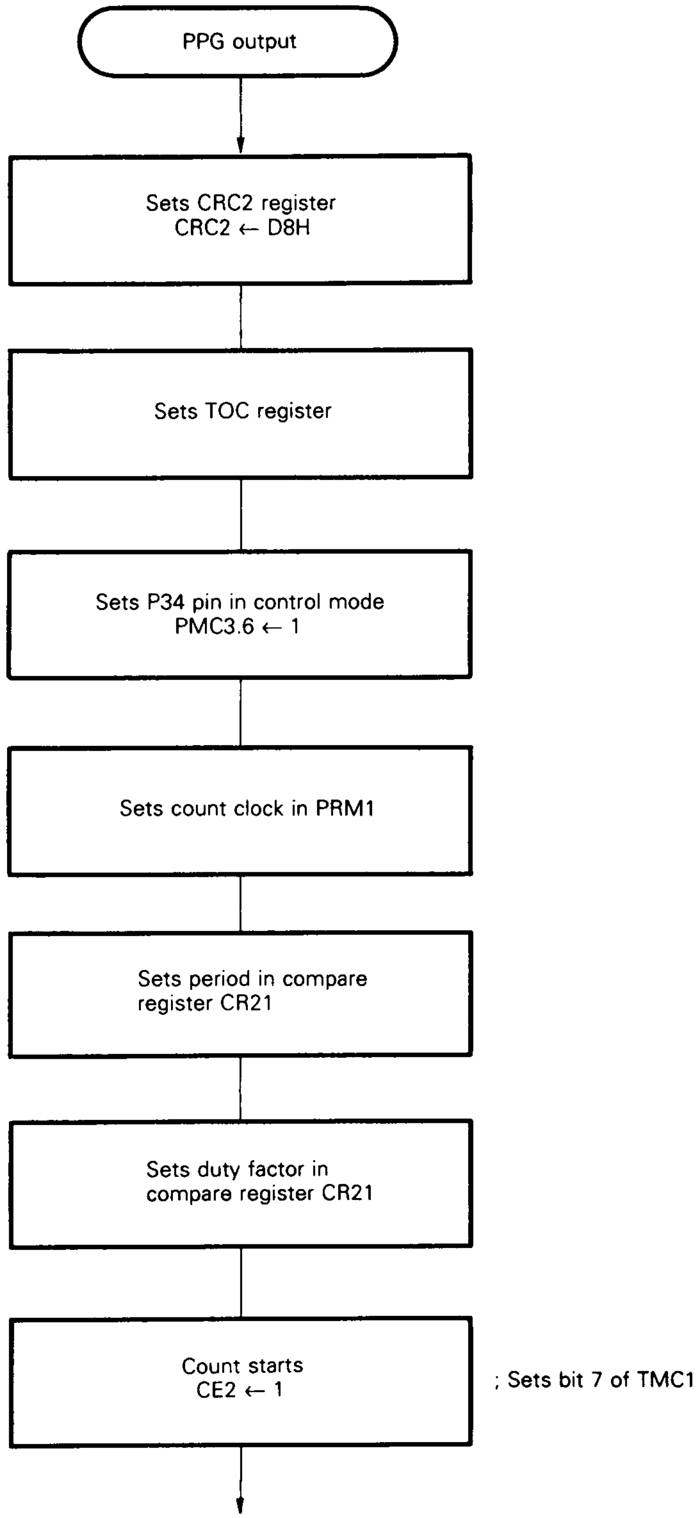
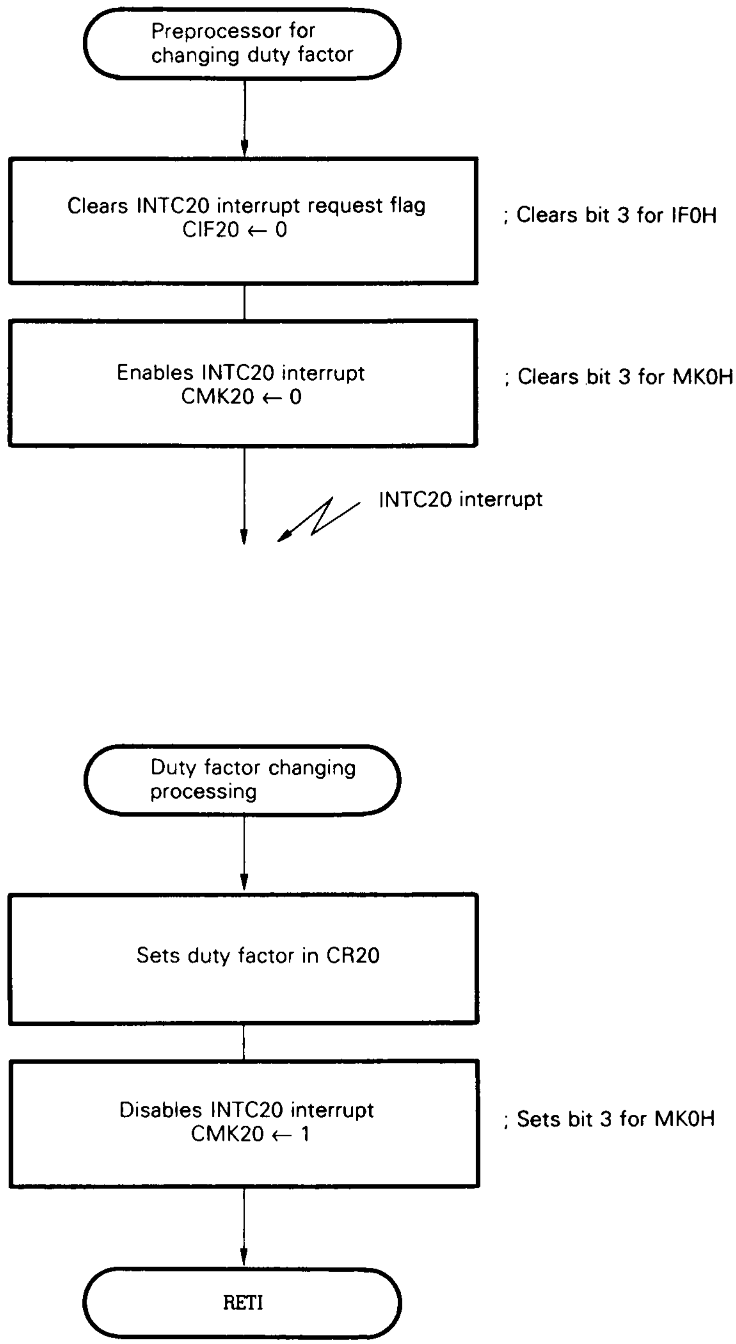


Fig. 7-123 Changing Duty Factor of PPG Output



(6) As external event counter

Timer 2 can be used as an external event counter, that counts the number of clock pulses input to the CI pin, from an external source.

In this case, the TM2 value is incremented in synchronization with the valid edge (rising edge only) for the CI pin, as shown in Fig. 7-124.

Fig. 7-124 External Event Timer Operation (Single Edge Only)

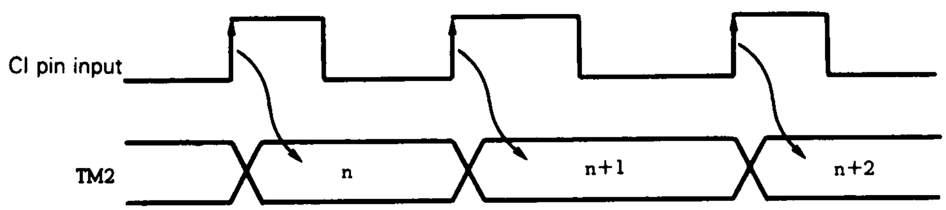
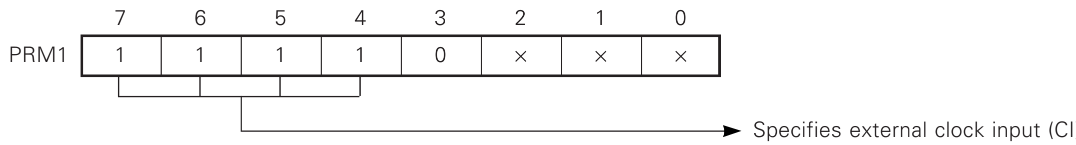


Fig. 7-125 shows the control register contents, when 8-bit timer 2 functions as an external event counter, while Fig. 7-126 shows the procedure used to set the register contents.

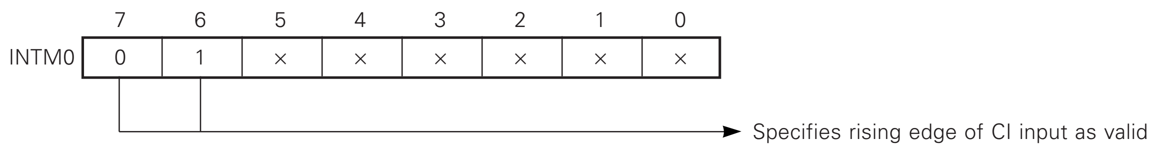
Remarks: The value of TM2 is less than the number of input clock pulses by 1.

Fig. 7-125 Control Register Contents for External Event Counter

(a) Prescaler mode register 1 (PRM1)



(b) External interrupt mode register 0 (INTM0)



(c) Timer control register 1 (TMC1)

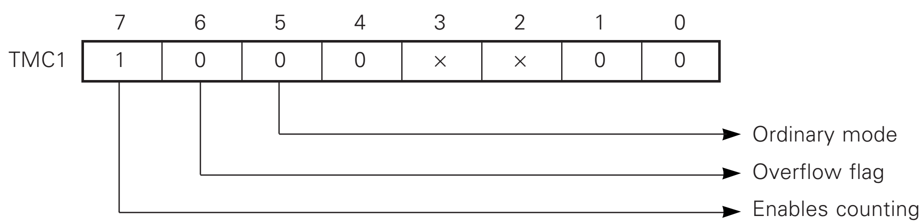
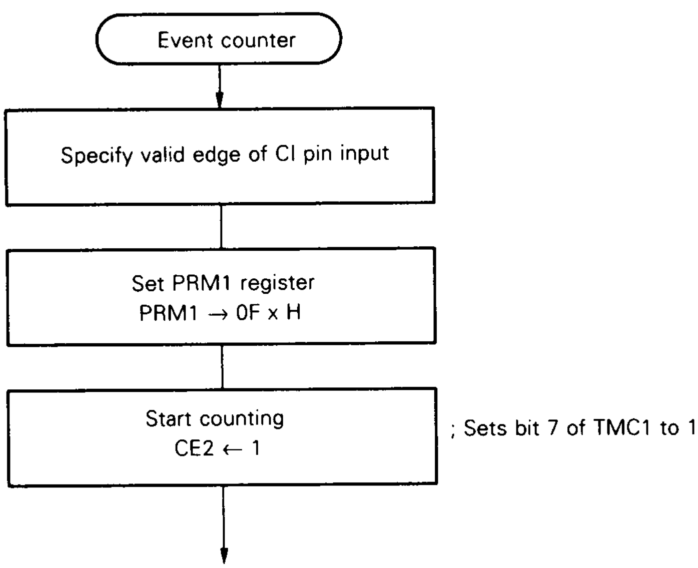


Fig. 7-126 External Event Counter Setting Procedure



(7) As one-shot timer

When 8-bit timer 2 (TM2) is used as a one-shot timer, the timer generates an interrupt only once, after the set time has elapsed (see **Fig. 7-127**).

After the timer has generated an interrupt, the one-shot operation for the timer can be started again, by clearing the OVF2 bit for timer control register 1 (TMC1) to 0.

Fig. 7-128 shows the control register contents for this operation, while Fig. 7-129 shows the procedure used to set the register contents. Fig. 7-130 illustrates how to start the one-shot operation for the second time.

Fig. 7-127 One-Shot Timer Operation

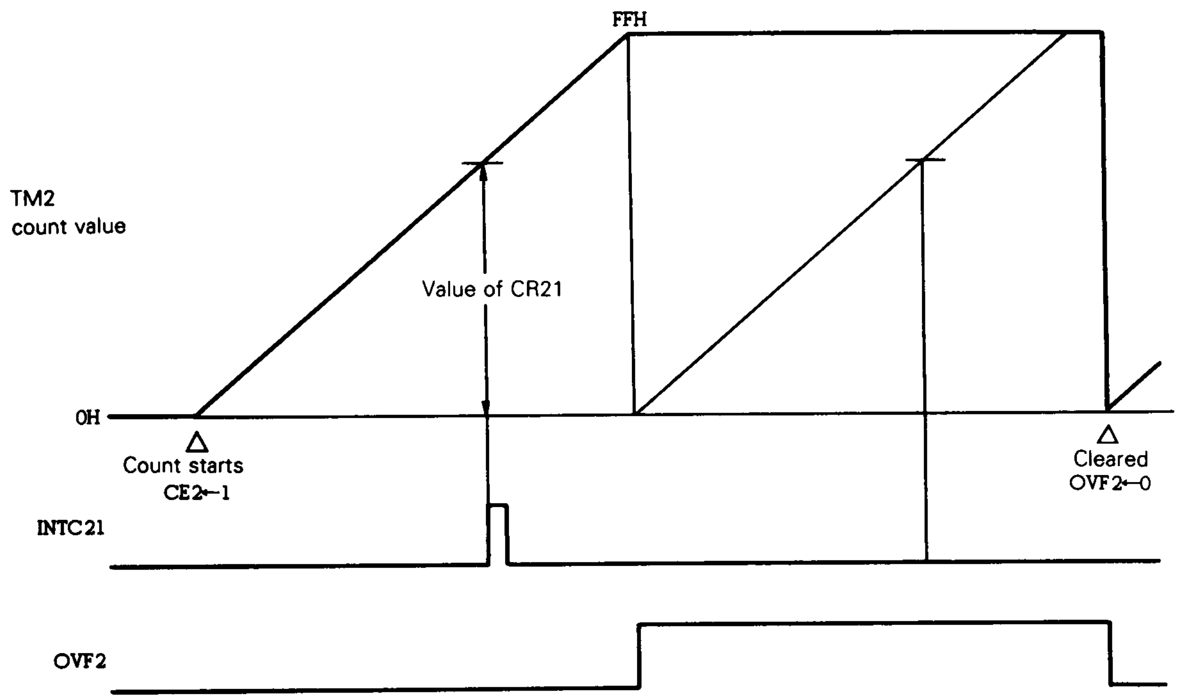
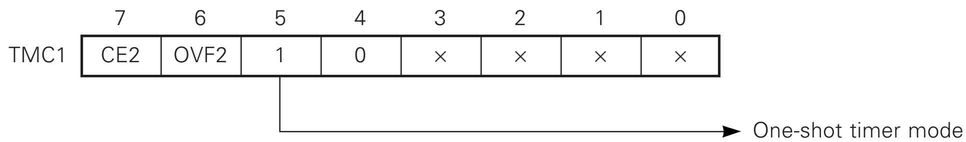
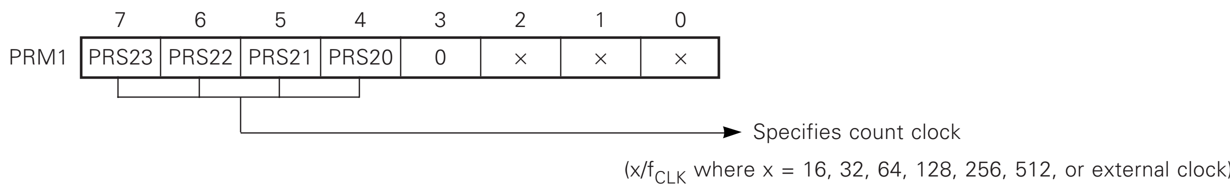


Fig. 7-128 Control Register Contents for One-Shot Timer Operation

(a) Timer control register 1 (TMC1)



(b) Prescaler mode register 1 (PRM1)



(c) Capture/compare control register 2 (CRC2)

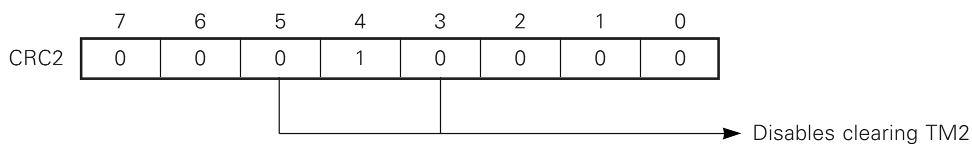


Fig. 7-129 Control Register Setting Procedure

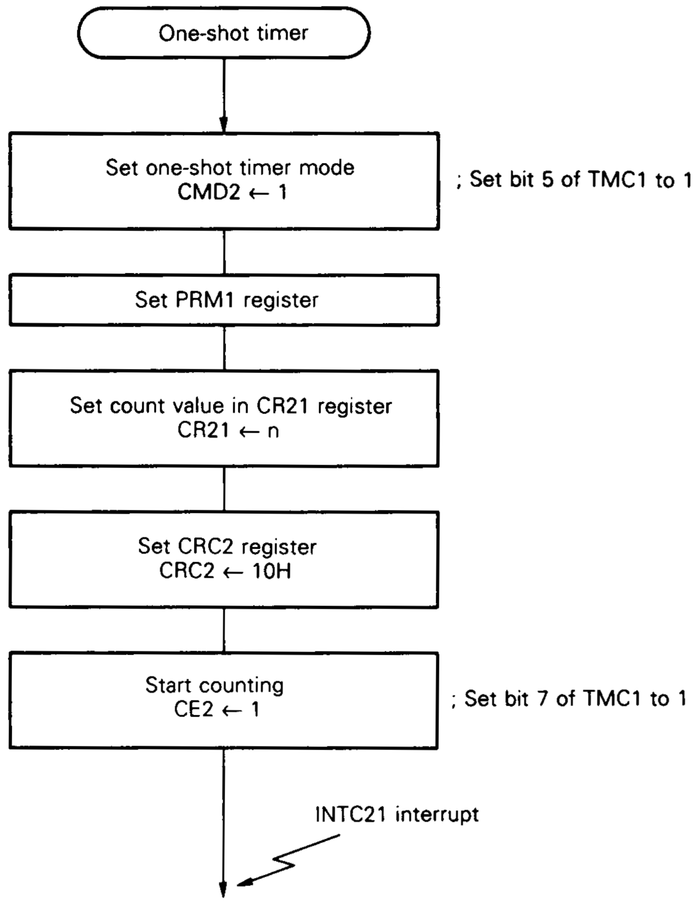
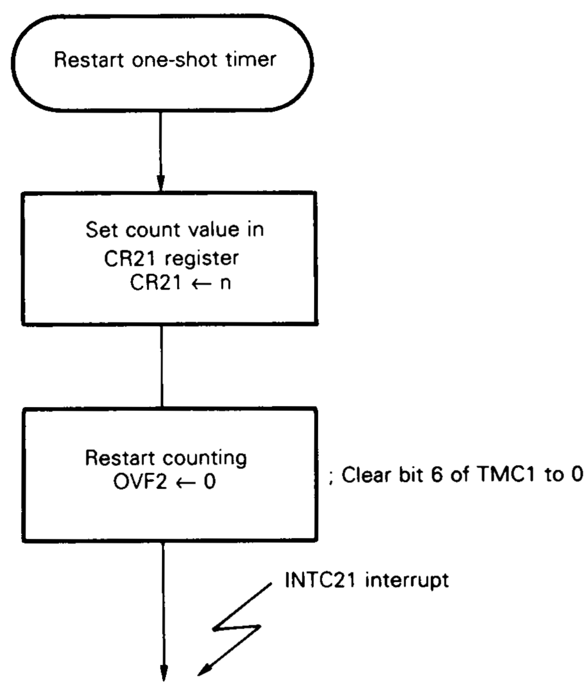


Fig. 7-130 Starting One-Shot Timer for 2nd Time



7.4 8-bit Timer/Counter 3

7.4.1 Function

The 8-bit timer/couner 3 can be used not only as an interval timer but also as a clock source for a baud rate generator.

When the timer/counter is used as an interval timer, it generates an interrupt at predetermined intervals. Table 7-19 shows the time range within which an interval time can be set.

Table 7-19 Time Interval for 8-bit Timer/Counter 3

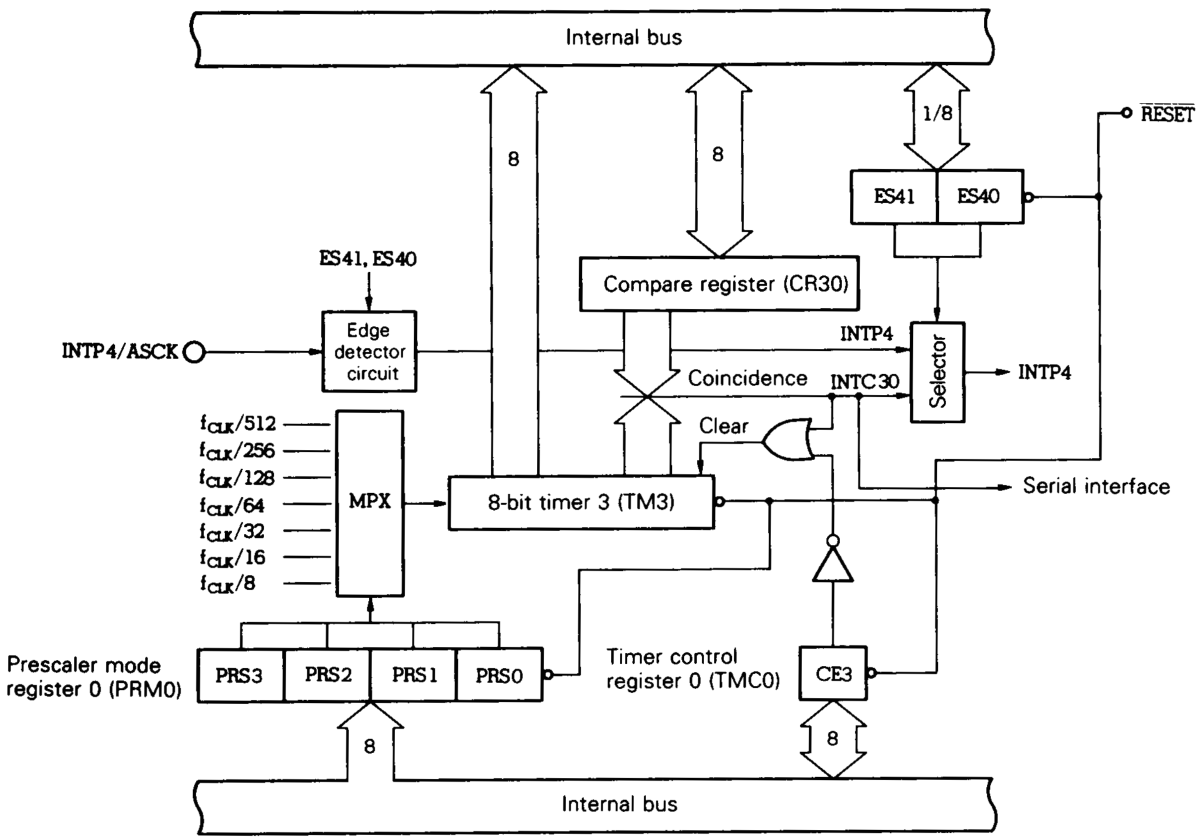
Resolution	Minimum interval	Maximum interval
$8/f_{CLK}$ (1.3 μs)	$8/f_{CLK}$ (1.3 μs)	$2^8 \times 8/f_{CLK}$ (341 μs)
$16/f_{CLK}$ (2.6 μs)	$16/f_{CLK}$ (2.6 μs)	$2^8 \times 16/f_{CLK}$ (683 μs)
$32/f_{CLK}$ (5.3 μs)	$32/f_{CLK}$ (5.3 μs)	$2^8 \times 32/f_{CLK}$ (1.37 ms)
$64/f_{CLK}$ (10.7 μs)	$64/f_{CLK}$ (10.7 μs)	$2^8 \times 64/f_{CLK}$ (2.73 ms)
$128/f_{CLK}$ (21.3 μs)	$128/f_{CLK}$ (21.3 μs)	$2^8 \times 128/f_{CLK}$ (5.46 ms)
$256/f_{CLK}$ (42.7 μs)	$256/f_{CLK}$ (42.7 μs)	$2^8 \times 256/f_{CLK}$ (10.9 ms)
$512/f_{CLK}$ (85.3 μs)	$512/f_{CLK}$ (85.3 μs)	$2^8 \times 512/f_{CLK}$ (21.8 ms)

Figures in ( ) are at f_{CLK} = 6 MHz.

7.4.2 Configuration

The 8-bit timer/counter 3 consists of an 8-bit timer (TM3) and an 8-bit compare register (CR30).
Fig. 7-131 shows the 8-bit timer/counter 3 configuration.

Fig. 7-131 8-bit Timer/Counter 3 Configuration



(1) 8-bit timer 3 (TM3)

Timer TM3 counts up the count clock specified by the higher 4 bits of the prescaler mode register 0 (PRM0). The timer contents can only be read by an 8-bit manipulation instruction. The timer operation can be disabled or enabled by timer control register 0 (TMC0).

When the $\overline{\text{RESET}}$ signal is input, the TM3 contents are cleared to 00H and TM3 stops counting.

(2) Compare register (CR30)

CR30 is an 8-bit register holding a value that determines the cycle for the interval timer.

When the contents of this register coincide with the TM3 value, the TM3 contents are automatically cleared and an interrupt request (INTC30) is generated.

Data can be read from or written to this register by an 8-bit manipulation instruction. When the $\overline{\text{RESET}}$ signal is input, the contents of this register become undefined.

(3) Prescaler

The prescaler generates the count clock from the internal system clock. The clock generated by the prescaler is selected by the selector as the count clock, based on which the timer performs its counting operation.

7.4.3 8-bit timer/counter 3 control registers

(1) Timer control register 0 (TMC0)

Register TMC0 is an 8-bit register that controls the counting operation for 8-bit timer 3 (TM3). The higher 4 bits of this register are used to control the TM3 counting operation, while the lower 4 bits are used to control the operation of the 16-bit timer/counter (TM0). Data can be read from or written to this register by an 8-bit manipulation instruction. Figure 7-132 shows the format for this register.

When the $\overline{\text{RESET}}$ signal is input, the TMC0 register contents are cleared to 00H.

Fig. 7-132 Timer Control Register 0 (TMC0) Format

	7	6	5	4	3	2	1	0
TMC0	CE3	0	0	0	CE0	OVF0	0	0

These bits control count operation for 16-bit timer/counter (TM0) (see **Fig. 7-3**).

CE3	Controls TM3 counting operation
0	Clears and stops counting operation
1	Enables counting operation

(2) Prescaler mode register 0 (PRM0)

This 8-bit register specifies the count clock supplied to 8-bit timer 3 (TM3).

Data can only be written to this register by an 8-bit manipulation instruction. Figure 7-133 shows the format for this register.

When the $\overline{\text{RESET}}$ signal is input, the PRM0 contents are cleared to 00H.

Fig. 7-133 Prescaler Mode Register 0 (PRM0) Format

	7	6	5	4	3	2	1	0
PRM0	PRS3	PRS2	PRS1	PRS0	0	0	0	0

f_{CLK}=6MHz

PRS3	PRS2	PRS1	PRS0	Specifies count clock (Hz)	Resolution
0	0	0	0	$f_{CLK}/8$	1.3 μs
0	0	0	1		
0	0	1	0	$f_{CLK}/16$	2.6 μs
0	0	1	1	$f_{CLK}/32$	5.3 μs
0	1	0	0	$f_{CLK}/64$	10.7 μs
0	1	0	1	$f_{CLK}/128$	21.3 μs
0	1	1	0	$f_{CLK}/256$	42.7 μs
0	1	1	1	$f_{CLK}/512$	85.3 μs
1	×	×	×	Inhibited	

Remarks: f_{CLK}: system clock frequency
x : 1 or 0

7.4.4 8-bit timer 3 (TM3) operation

(1) Basic operation

The 8-bit timer/counter 3 counts up the count clocks, specified by the higher 4 bits of the prescaler mode register 0 (PRM0).

When the $\overline{\text{RESET}}$ signal is input, the TM3 contents are cleared to 00H and TM3 stops counting.

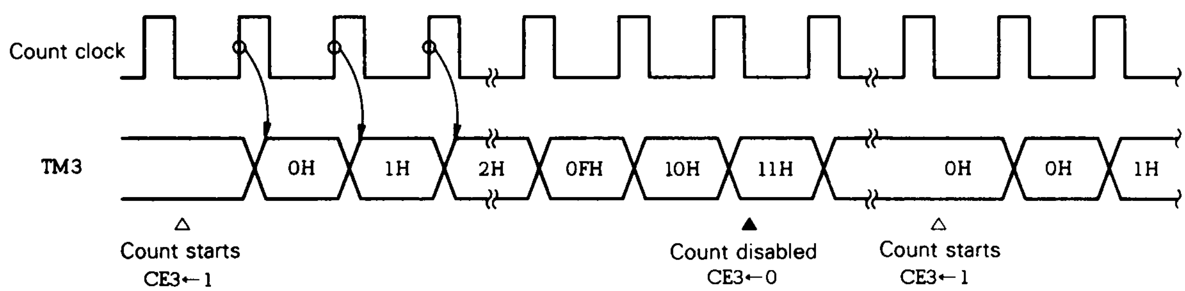
The TM3 counting operation is enabled or disabled by bit 7 (CE3) for timer control register 0 (TMC0) (the 8-bit timer/counter 3 operation is controlled by the higher 4 bits for the TMC0 register).

When the CE3 bit is set to 1 by software, the TM3 contents are cleared to 00H at the next count clock and TM3 starts counting up. By resetting the CE3 bit to 0, the TM3 contents are cleared to 00H at the next count clock, and coincidence signal generation is stopped.

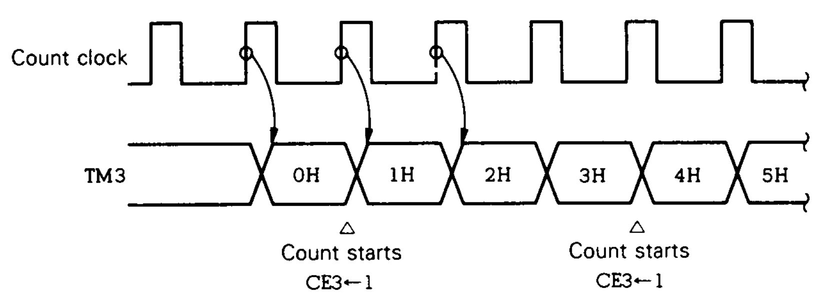
If an attempt is made to set the CE3 bit, which has already been set to 1, TM3 is not cleared but continues its operation.

Fig. 7-134 Basic Operation for 8-bit Timer 3 (TM3)

(a) When counting is started, stopped, and then started again



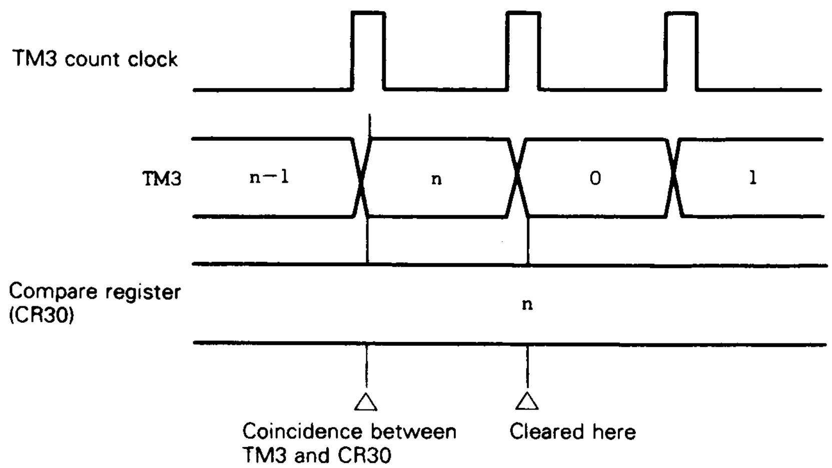
(b) When "1" is written to CE3 bit again after counting is started



(2) Clearing operation

The 8-bit timer 3 (TM3) can be automatically cleared, after its value has coincided with the compare register (CR30) contents. When the reason for clearing TM3 has occurred, TM3 is cleared to 00H at the next count clock. Therefore, even if the clearing cause has occurred, the current TM3 value is retained, until the next count clock is applied.

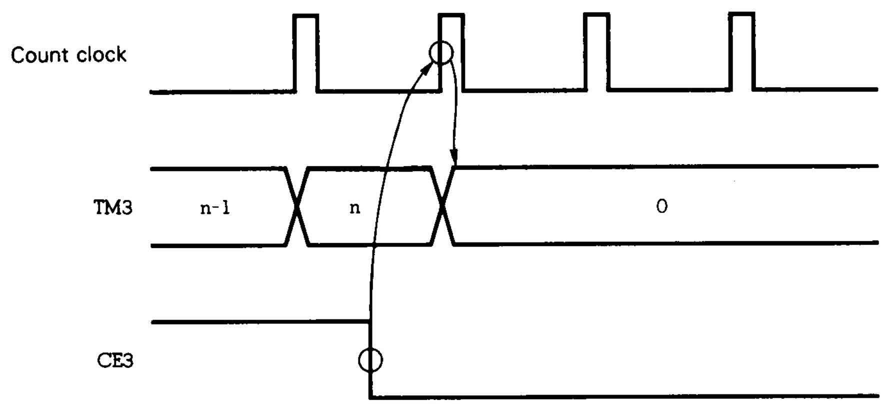
Fig. 7-135 Clearing TM3 by Coincidence with Compare Register



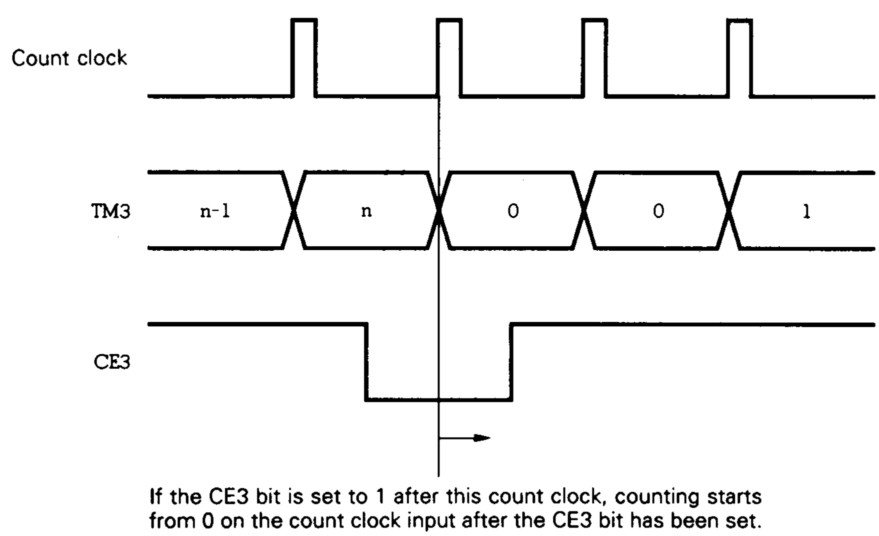
TM3 is also cleared by resetting the CE3 bit of the timer control register (TMC0) to 0 through software. The clear operation is also performed by the count clock after the CE3 bit has been reset to 0. If the CE3 bit is set to 1 after the CE3 bit has been reset to 0 and before TM3 is cleared to 0 (before the first count clock is input after the CE3 bit has been reset to 0), TM3 is cleared to 0 and at the same time, 0 counting operation is performed because counting has been started.

Fig. 7-136 Clear Operation When CE3 Bit Is Reset to 0

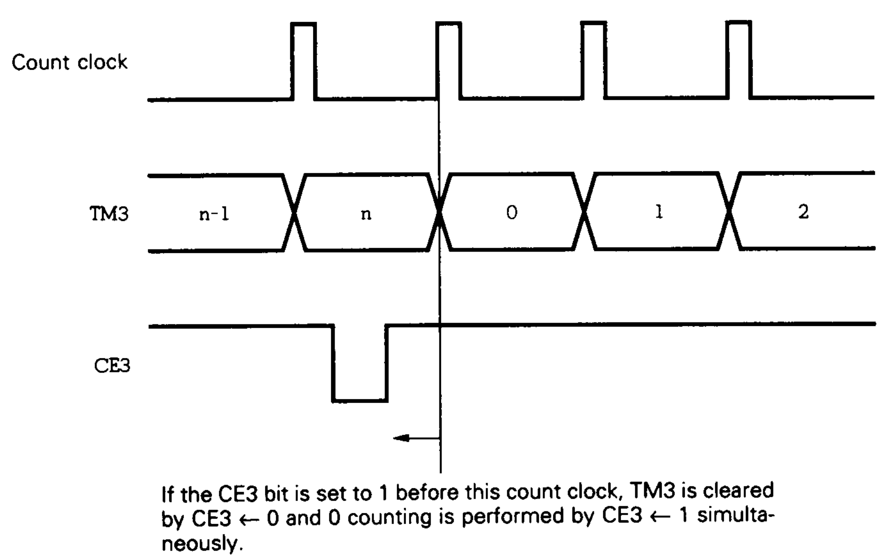
(a) Basic operation



(b) Restart after TM3 has been cleared to 0



(c) Restart before TM3 is cleared to 0



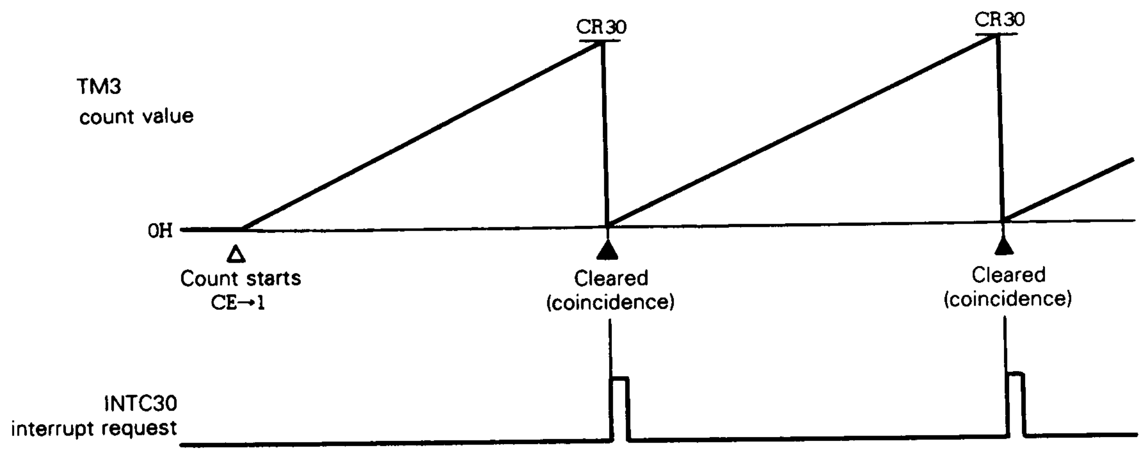
7.4.5 Compare register operation

The 8-bit timer/counter 3 can perform a compare operation which compares the current count value for the timer with the set value for the compare register.

When the count value for the 8-bit timer 3 (TM3) coincides with the value set in advance in the compare register (CR30), an interrupt request signal (INTC30) is generated.

After the timer value has coincided with the compare register value, the TM3 count value is automatically cleared. Therefore, the timer can be used as an interval timer that repeatedly counts up to the value set in the CR30 register.

Fig. 7-137 Compare Operation



7.4.6 Application examples

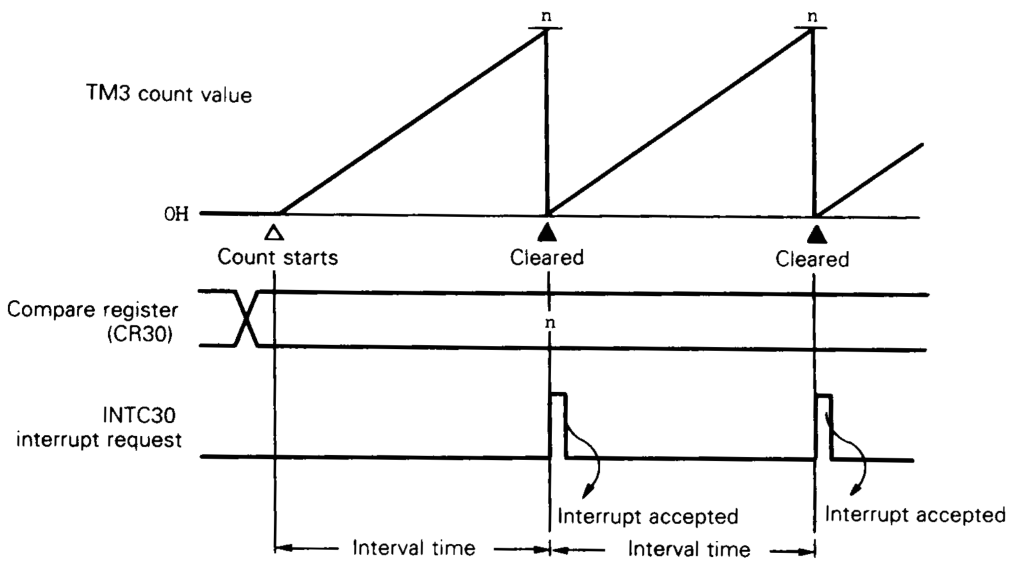
- Operation as interval timer

The 8-bit timer 3 (TM3) can be used as an interval timer that repeatedly generates an interrupt at predetermined intervals. In addition, the timer can also be used as a baud rate generator.

The interval timer can count up to 341 μs with a 1.3 μs resolution, or up to 21.8 ms with an 85.3 μs resolution (at internal system clock f_{CLK} = 6 MHz).

Figure 7-138 illustrates the interval timer operation, while Fig. 7-139 shows the control register contents at this time. Figure 7-140 shows the procedure used to set the control register contents.

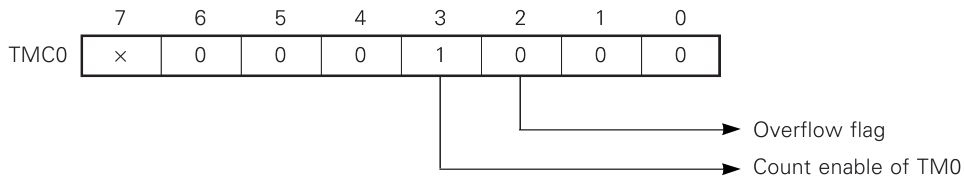
Fig. 7-138 Interval Timer Operation Timing



Remarks: Interval time = (n+1) × x/f_{CLK}, where 0 ≦ n ≦ FFH
x = 8, 16, 32, 64, 128, 256, 512

Fig. 7-139 Control Register Contents for Interval Timer Operation

(a) Timer control register 0 (TMC0)



(b) Prescaler mode register 0 (PRM0)

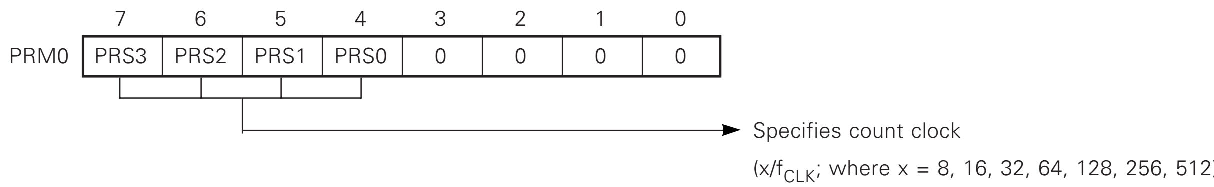
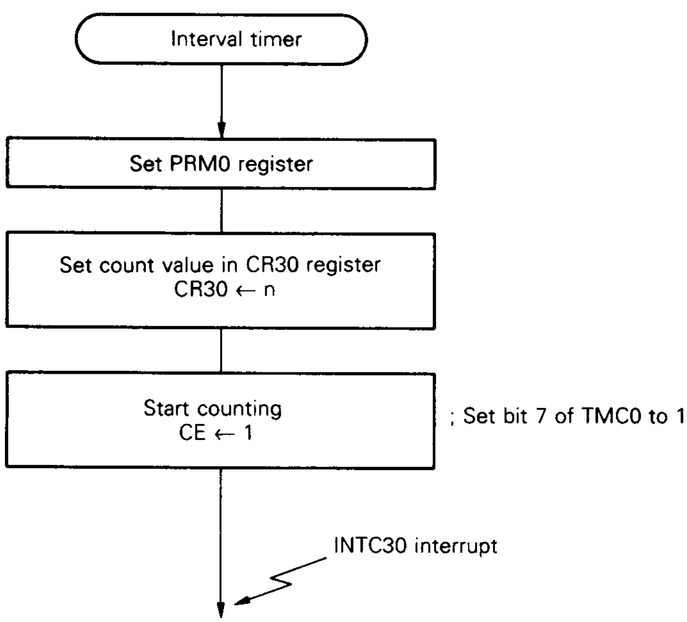


Fig. 7-140 Interval Timer Operation Setting Procedure



7.5 Notes

7.5.1 Notes common to all timers/counters

(1) If the contents of the following registers are changed while a counter is operating (i.e., while the CEm bit for the TMCn register is set), the counter may malfunction. This happens because, if the hardware function is changed because the contents of a register have been changed, the functions for the register, before its contents have been changed, contend with the new register function, causing the register contents to be undefined.
Therefore, be sure to stop the counter before changing the contents of the following registers:

- Prescaler mode register (PRMn)
- Capture/compare control register (CRCn)
- Timer output control register (TOC)
- CMD2 bit for timer control register 1 (TMC1)

(2) The OVFm flag, that retains the overflow signal sent from a timer/counter, is in the TMCn register that controls the timer/counter operation. When a read-modify-write instruction (such as AND TMCn,#7FH) is executed, the OVFm flag may be cleared. (Even when the OVFm flag is 0, when it is read, the flag may be 1, when data is to be written to it, because an overflow may have occurred from the timer. However, since the OVFm flag was 0 when it was read, data 0 is written to the flag, clearing it to 0).
Clearing the OVFm flag for the timer/counter assigned to the same TMCn register can be prevented by the following method:

1. Read the TMm register contents in the timer/counter, using the OVFm flag.
2. Manipulate the timer/counter and read the TMm register contents again.
3. Compare the two read values. If the value read last is less than the value read first, set the OVFm flag.

When this method is used, do not manipulate the OVFm flag by an interrupt routine. Note that the time, from when the TMm flag is read for the first time until the flag is read the second time, must be shorter than the time for the full TMm count.

Example: To not clear the OVF1 flag for timer/counter 1

```
        MOV A,TM1
        MOV TMC1,#xxx01000B; xxx depends on manipulation of timer/counter  2
        CMP A,TM1               ; checks timer value
        BL  $NEXT
        BE  $NEXT
        MOV TMC1,#xxx01100B; sets OVF1
NEXT:
```

(3) If the compare register contents coincide with the timer register contents before an instruction that stops the timer operation is executed, the timer stops its counting operation, but generates an interrupt request. To prevent the interrupt request from being generated, when the timer is stopped, be sure to mask the interrupt by the mask register and then stop the timer.

Example: Program that may generate interrupt request:

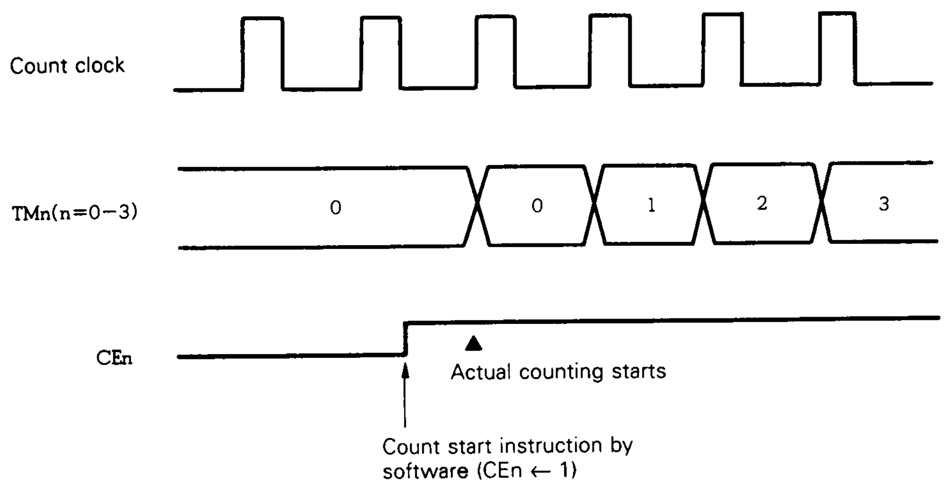
```
        ⋮
MOV TMC1,#6CH
                ← Interrupt request occurs from timer/counter during this period
AND MK0H,#0F6H
        ⋮
```

Program that does not generate interrupt request:

```
        ⋮
AND MK0H,#0F6H  ← Disables interrupt from timer/counter 2
MOV TMC1,#6CH
AND IF0H,#0F6H  ← Clear interrupt request flag for timer/counter 2
        ⋮
```

(4) It takes a time equivalent to up to 1 count clock for the timer/counter to actually start operating after an operation that starts the timer/counter has been performed (CEn ← 1, (n = 0 to 3)). (See **Fig. 7-141**).
For example, when a timer is used as an interval timer, the first interval time is extended by 1 clock, but the second interval time and those that follow are as specified.

Fig. 7-141 Operation When Counting Is Started



(5) Even when an instruction, that stops the timer (CEn ← 0), has been executed, the TMn contents do not immediately become 0, until the count clock is generated after the instruction execution.
If the timer is started (CEn ← 1) after the timer has been stopped and before the timer value reaches 0, timer counts up from 0.

Fig. 7-142 Stopping Count Operation

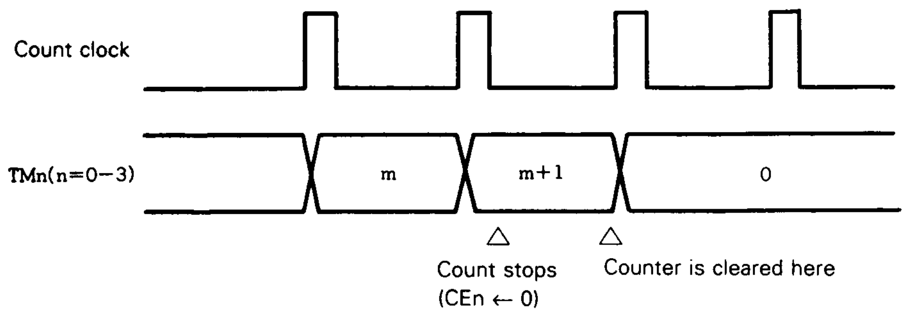
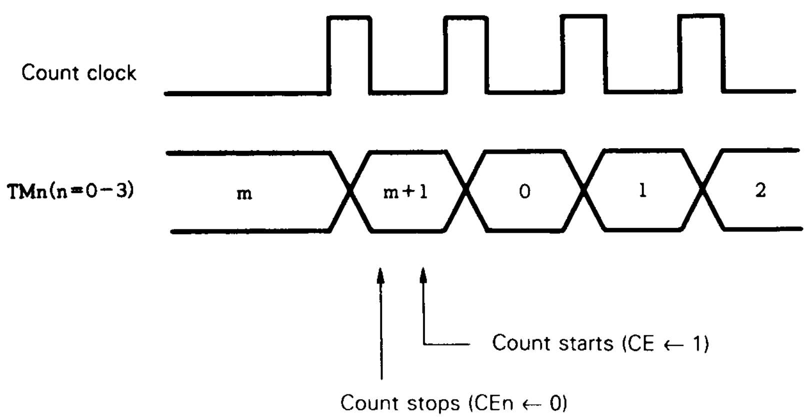


Fig. 7-143 Stopping and Starting Counting Operation



(6) To access the registers for the timers/counters, wait cycles having the following number of cycles are automatically inserted.

Table 7-20 Maximum Number of Wait Cycles Inserted When Registers for Timers/Counters Are Accessed

Register	Maximum number of wait cycles	
	Read	Write
TMCn	0	1
PRMn	—	1
CRCn	—	1
TOC	—	1
OSPC	0	0
CRnm	1	1
TM0	7	—
TM1	15	—
TM2	15	—
TM3	7	—

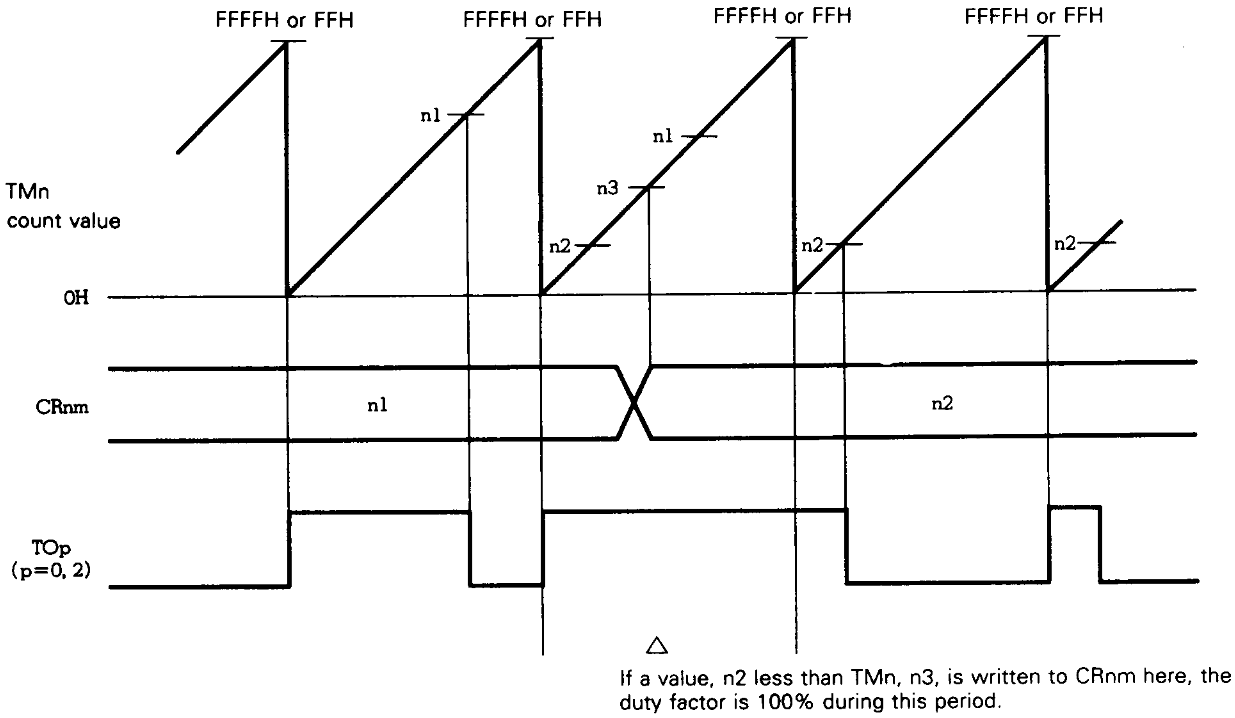
Caution: One clock is $1/f_{CLK}$

(7) While an instruction that writes data to compare register CRnm (n = 0 to 3, m = 0 or 1) is executed, coincidence between the contents in CRnm register, to which data is being written, and the TMn value (n = 0 to 3) is not detected. For example, an interrupt request is not generated, even when the TMn value coincides with the CRnm register value, if the CRnm register contents are not changed before and after new data is written to the register, and the level of the timer output pin (TOn, n = 0 to 3) is not changed.
Write data to the CRnm register, while the timer/counter is operating at the timing at which the TMn value does not coincide with the CRnm register value, before and after the register contents are changed (especially immediately after an interrupt request is generated, because of coincidence between the TMn value and that for CRnm).

(8) Coincidence between the TMn value and CRnm value is detected, only when the TMn value is incremented. Therefore, an interrupt request does not occur, even if the same value as that for TMn is written to CRnm, and the timer output pin level does not change.

(9) When PWM is used, if values less than the value of TMn (n=0, 2) are set to the compare registers CRnm (n=0, 2, m=0, 1), a PWM signal with a duty factor of 100% is output. Rewrite the values of CRnm by using an interrupt that occurs when TMn coincides with a compare CRnm.

Fig. 7-144 PWM Example When Duty Factor Is 100%

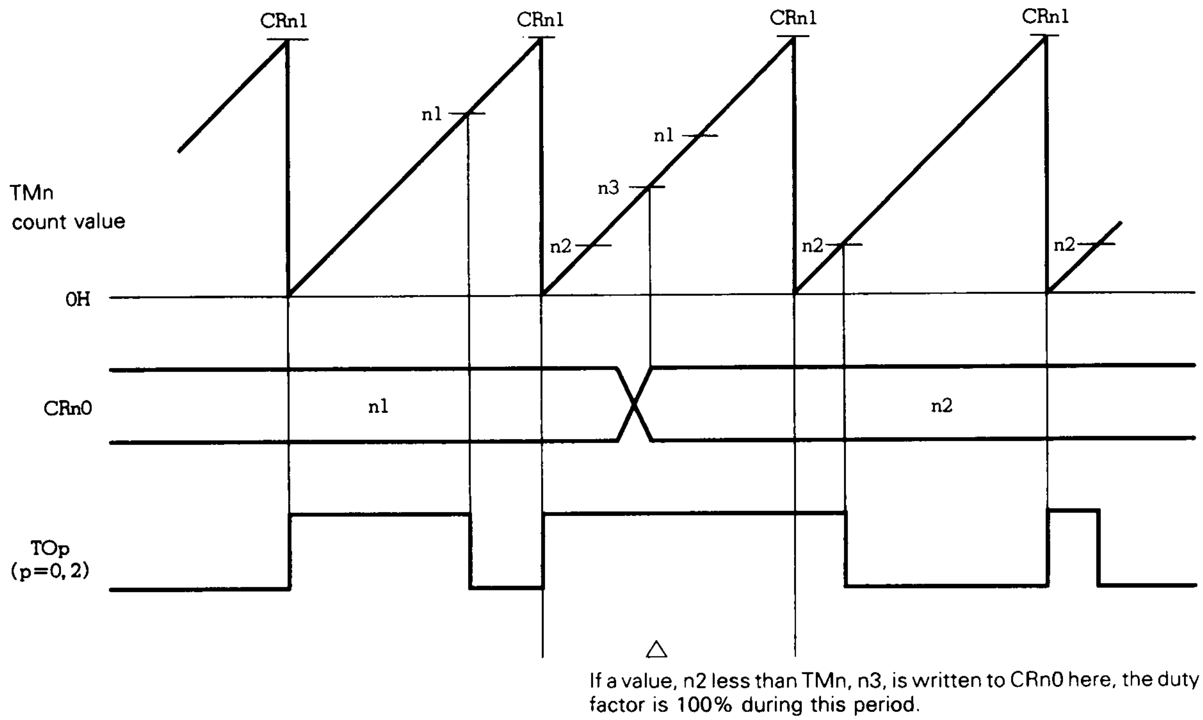


Remarks: ALVp = 0

(10) Notes for rewriting compare register when PPG output is used.

(a) If a value less than that TMn is written to compare register CRn0 before the value of the CRn0 (n=0,2) coincides with that of TMn (n=0, 2) the duty factor of the PPG cycle becomes 100%. To rewrite the value of CRn0, use an interrupt that occurs when the value of CRn0 coincides with that of TMn.

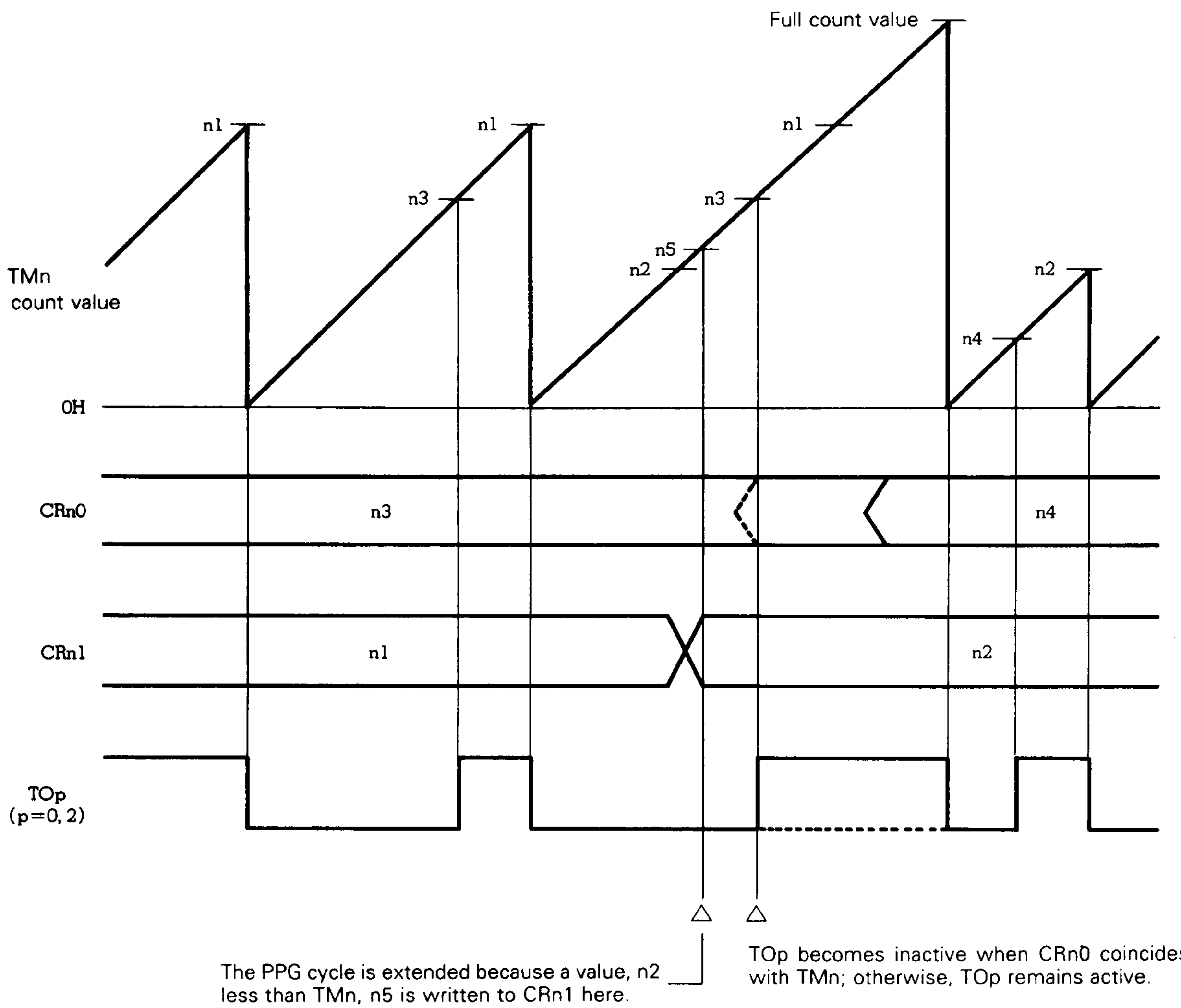
Fig. 7-145 PPG Output Example When Duty Factor Is 100%



Remarks: ALVp = 0

(b) When the value of compare register (CRn1) is to be changed to a value less than the current value, if a value less than the value of TMn is set to CRn1, the PPG cycle is extended to the time required for full count of TMn. At this time, the output level becomes inactive until TMn overflows and is cleared to 0 if the value of CRn1 is rewritten after its value has coincided with TMn, returning to the normal PPG output. If the value of CR01 is changed before CRn0 coincides with TMn, the active level is output until CRn0 coincides with TMn. If CRn0 coincides with TMn before TMn overflows and is cleared to 0, the inactive level is output at that point, and the active level is output when TMn is cleared to 0, and the normal PPG output is restored.
Rewrite the value of CRn1 by using an interrupt that occurs when CRn1 coincides with TMn.

Fig. 7-146 PPG Output Example When Output Cycle Is Extended



Remarks: ALVp = 1

(c) If the PPG cycle is extremely short in respect to the time required for accepting an interrupt, measures described in Notes (a) and (b) are not effective. Take other measures (such as masking all the interrupts and polling the interrupt flags through software).

(11) The PWM output pulse width is two f_{CLK} clocks longer than the result of the approximate expression at the active level, and is two f_{CLK} clocks shorter at the inactive level. If high-accuracy output is necessary, or if the count clock is at high speed, take these points into consideration. For details, refer to (1) PWM output in **7.1.7** and **7.3.9 PWM output**.

(12) When the timer output is disabled (ENTOn = 0, n=0, 1, 2, or 3), the output level for the TOn (n = 0, 1, 2, or 3) pin is a complement of the value set in ALVn (n = 0, 1, 2, or 3). Note, therefore, that the active level is output, when the timer is stopped or timer output is disabled with the PWM or PPG output function selected.

(13) The pulse width for the PPG output is two f_{CLK} clocks longer than the result of the approximate expression at the active level, and is two f_{CLK} clocks shorter at the inactive level. If high-accuracy output is necessary, if the PPG pulse period is short, or if the count clock is at high speed, take these points into consideration. For details, refer to **7.1.8** and **7.3.10 PPG output**.

(14) The in-circuit emulator cannot reject the digital noise normally. Therefore, when using the in-circuit emulator with the edge detection function of the timer/counter, keep in mind the following points:

- Capture/clear operation of timer/counter
  : These operations are not influenced by the erroneously detected edge. Therefore, even if an interrupt is generated by the erroneously detected edge, the capture value is not updated. Note that the value of CR22 becomes undefined after it has been read by the CPU.

- Compare operation of timer/counter
  : If a mode in which the clear operation is to be performed after capture, the coincidence interrupt generation timing is changed due to an influence of the erroneously detected edge. Consequently, the coincidence interrupt occurs even when the value of the timer/counter does not coincide with that of the compare register.
  In this case, the normal coincidence interrupt generation timing is restored if the correct edge is input or if the timer/counter is stopped.
  : The timer output is not influenced by the erroneously detected edge, and operates at the correct timing.

For the detail of erroneous edge detection, refer to **13.4 Notes** in **CHAPTER 13 EDGE DETECTION FUNCTION**.

Caution: Refer to 7.5.4 Notes for using in-circuit emulator.

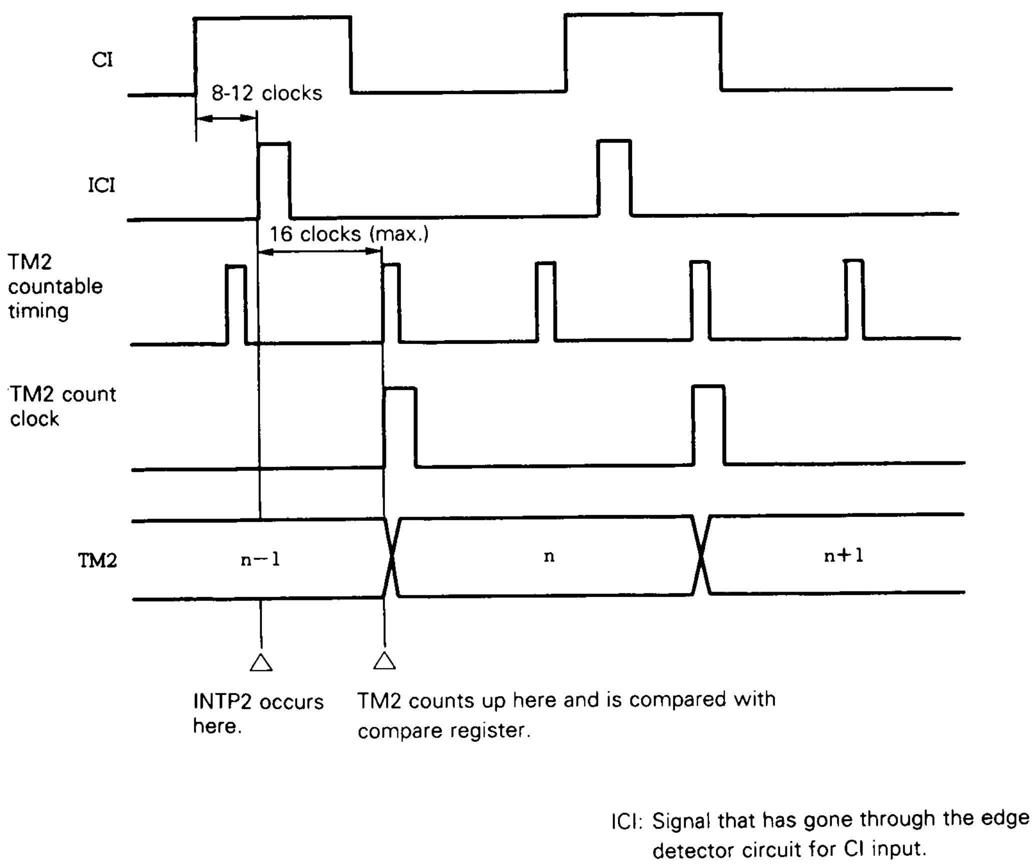
7.5.2 Note on 16-bit timer/counter

Before the 16-bit timer/counter performs a compare operation, data must be written to the 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, up to 28 system clocks (4.67 μs at f_{CLK} = 6 MHz) are required after the valid edge has been input to the CI pin and before the TM2 contents are incremented. Therefore, the TM2 value has not yet been incremented, if it is read immediately after the valid edge has been detected. Moreover, generation of an interrupt, which is generated, when the compare register (CR20 or CR21) contents coincide with the timer value, may take place later than the valid edge input. Take these factors into consideration, when delicate timing control must be performed.

Fig. 7-147 Interrupt Request Generation by External Event Counter



(2) When 8-bit timer/counter 2 is used as an external event counter, TM2 alone cannot make a clear distinction between when the valid edge is not input at all and when the valid edge is input only once (refer to Fig. 7-148), because the TM2 contents are 0 in either case. Use the INTP2 interrupt request flag to make a distinction (the INTP2 pin is multiplexed with the CI pin and both pins functions can be used at the same time).
Fig. 7-149 shows an example.

Fig. 7-148 If One Valid Edge Input Cannot Be Distinguished from No Valid Edge Input by External Event Counter

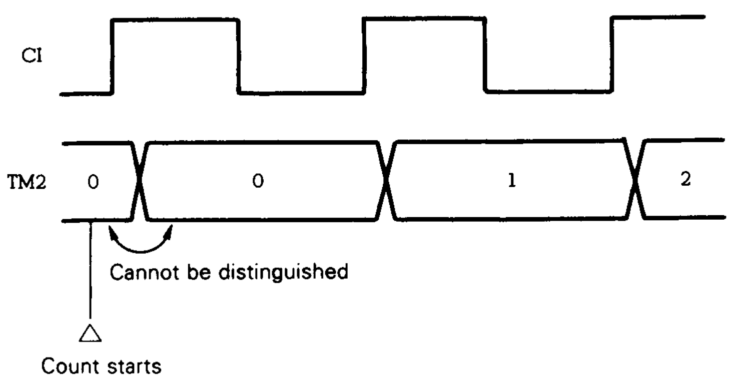
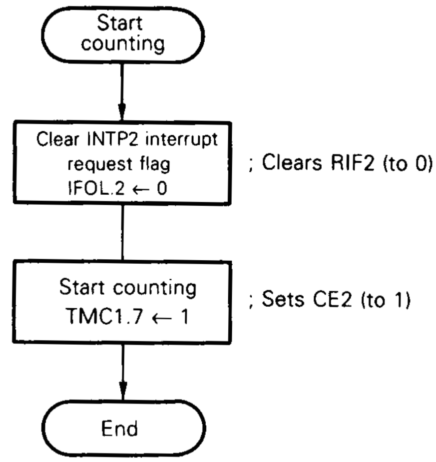
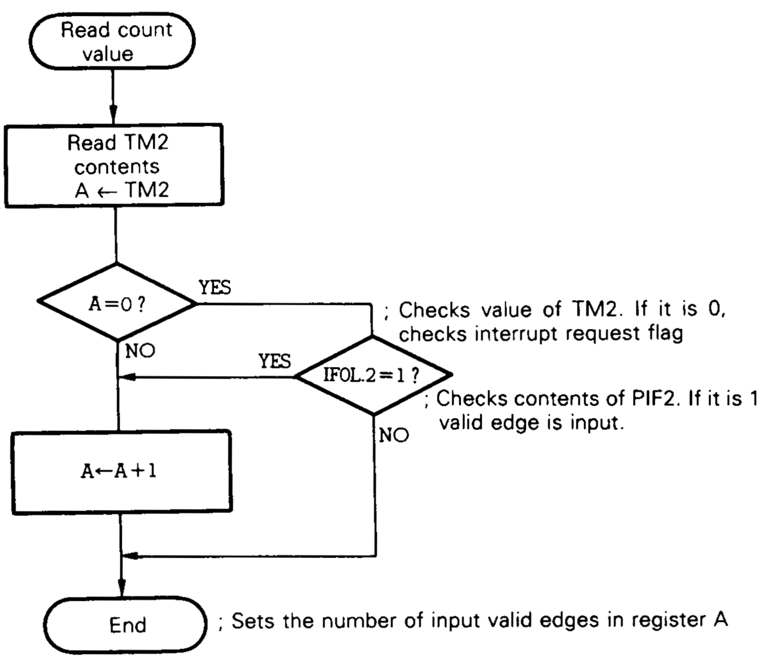


Fig. 7-149 To Detect Input of One Valid Edge by External Event Counter

(a) Processing when counting is started



(b) Processing when count value is read



(3) The in-circuit emulator cannot reject the digital noise normally. Therefore, when using the in-circuit emulator with the edge detection function of the timer/counter, keep in mind the following points, refer to **7.5.4 Notes for using in-circuit emulator**.

(4) The CR22 register contents become undefined, after they have been read. To use the captured value more than once, save the value to a register or memory.

7.5.4 Notes for using the in-circuit emulator

In the in-circuit emulator, the INTP0, INTP1, INTP2/CI, and INTP3 pins cannot normally perform digital noise elimination. Noise may be erroneously detected as an edge. Refer to **13.4 Notes** in **CHAPTER 13 EDGE DETECTION FUNCTION** for details of erroneous edge detection. The following describes how the timer/counter operates when noise is erroneously detected as an edge:

(a) Capture operation

No capture operation is performed by an erroneously detected edge. However, an interupt will be generated. The capture register values detected in the interrupt processing generated by an erroneously detected edge will be as follows:

- CR02, CR11
  Values captured at the previous normal edge
- CR22
  Undefined value

(b) Clear operation after capture (only for 8-bit timer /counters 1 and 2)

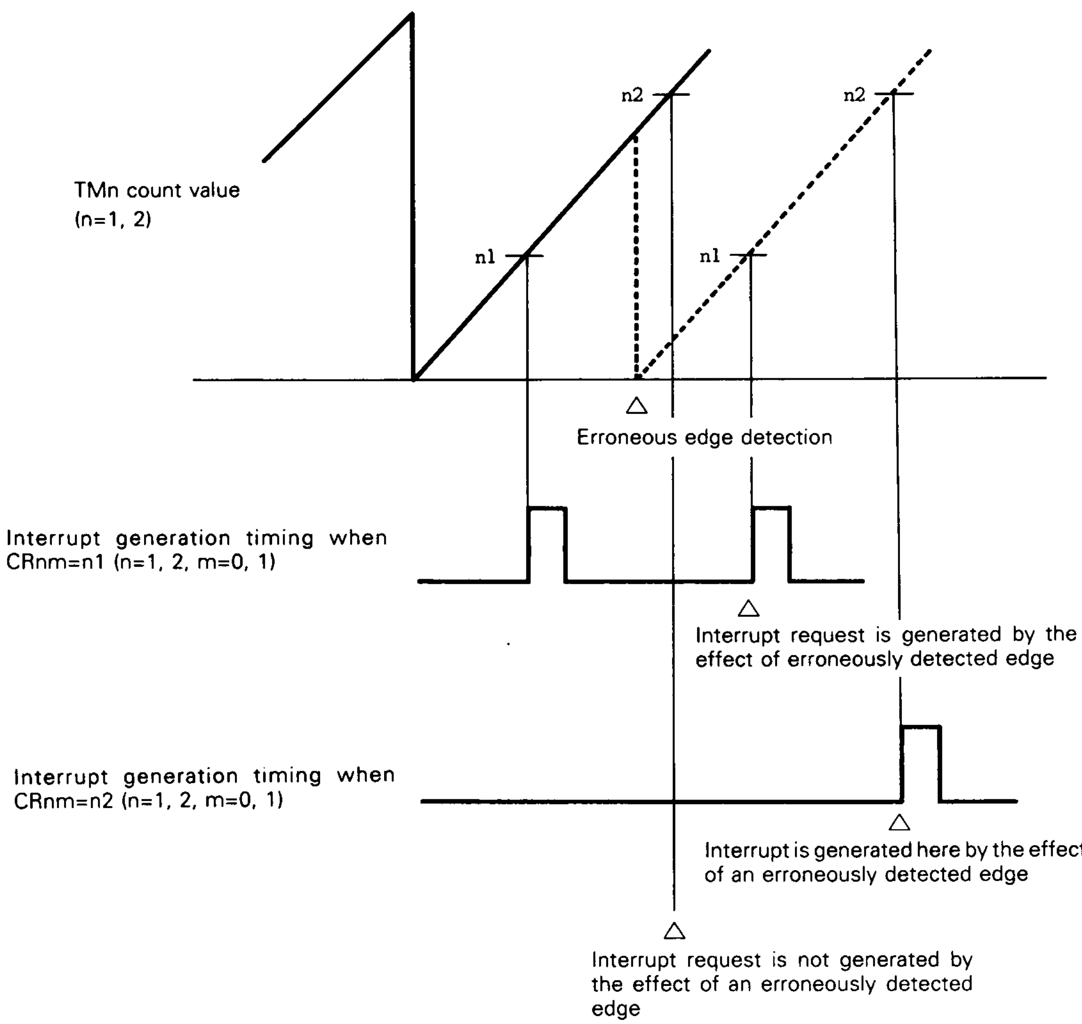
No clear operation is performed by an erroneously detected edge. However, after an edge detection, an interrupt request that should be based on the timer/counter value and the compare register value will be generated at a timing unrelated to the values of the timer/counter and the compare register. The generation of this interrupt will be at a timing that the timer/counter is assumed to be cleared (refer to **Fig. 7-150**).

If coincidence of the the timer/counter value of timer/counter 1 and the value of the compare register is used as the output trigger for the real-time output port, this unrelated timing will turn out to be the output trigger for the real-time output port.

The timer output function of timer/counter 2 operates at normal timing (this is not affected by an erroneously detected edge). The interrupt generation timing error can be corrected by the following operations:

- Clear operation by normal edge
- By clearing (to 0) the CEn (n=1, 2) bit of the respective timer/counter in timer control register 1 (TMC1)

Fig. 7-150 Timing Change of Interrupt Generation by Erroneously Detected Edge



(c) Event counter function (timer/counter 2 only)

The timer/counter value is not affected by an erroneously detected edge. However, the generation timing of the interrupt caused by the coincidence of the timer/counter value and the compare register value is advanced by the number of erroneously detected edges.

The timer output function operates at a normal timing (this is not affected by an erroneously detected edge).

This interrupt generation timing error can be corrected by the following operations:

- When using the clear function after capture, clearing by a normal edge
- By clearing (to 0) the CE2 bit of the timer control register 1 (TMC1)

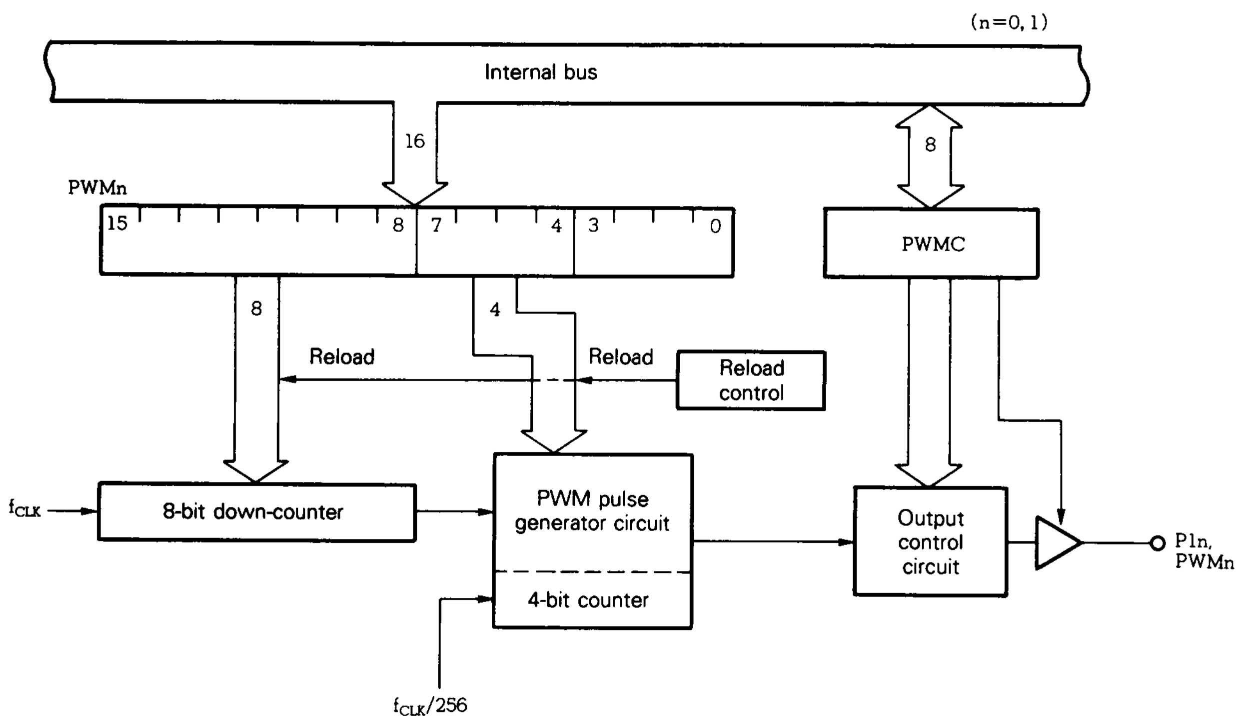
CHAPTER 8 PWM OUTPUT UNIT

μPD78234 is provided with two channels for PWM (pulse width modulation) output circuits, each having a 12 bit resolution. The active level for the PWM output can be specified to be high or low. The PWM output port is multiplexed with pins P10 and P11.

8.1 PWM Output Unit Configuration

Fig. 8-1 shows the the PWM output unit configuration.

Fig. 8-1 PWM Output Unit Configuration



(1) 8-bit down counter

Generates the basic PWM signal timing.

(2) PWM pulse generator circuit (including 4-bit counter)

Controls pulses addition and generates the PWM pulse to be output.

(3) Reload control

Controls reloading the modulo value for the 8-bit down counter and 4-bit counter.

(4) Output control circuit

Controls the PWM signal active level.

8.2 PWM Output Unit Control Registers

8.2.1 PWM control register (PWMC)

Fig. 8-2 PWM Control Register (PWMC) Format

	7	6	5	4	3	2	1	0	
PWMC	SYN1	0	SYN0	0	EN1	ALV1	EN0	ALV0	(05H when RESET is input)

(n=0, 1)

ALVn	Specifies active level for PWM_n pin
0	Low active level
1	High active level

ENn	Controls PWM output for PWM_n pin
0	Disables output. Pin level is determined by PM1, P1, and PUO registers contents (port mode)
1	Enables PWM output

SYNn	Specifies PWM pulse width rewriting cycle
0	Rewrites at every 16 PWM cycles ($2^{12}/f_{CLK}$)
1	Rewrites at every 1 PWM cycle ($2^8/f_{CLK}$)

The PWM control register (PWMC) is an 8-bit register that controls operations for the PWM output pins (PWM0 and PWM1).

Data can be read from or written to this register by a bit manipulation or 8-bit manipulation instruction.

When the $\overline{RESET}$ signal is input, the contents of this register are initialized to 05H, setting the PWM0 and PWM1 pins in the input port mode (output high-impedance state).

8.2.2 PWM modulo registers (PWM0 and PWM1)

PWM0 and PWM1 registers are 16-bit registers that determine the pulse widths for PWM pulses. Data can be set in these registers by a 16-bit data transfer manipulation instruction.

These registers are write-only registers and their contents cannot be read.

Bits 15 through 4 for these registers determine the widths of 12-bit PWM pulses (i.e., PWM pulses have a 12 bit resolution). Bits 3 through 0 are don't care bits and can be 1 or 0.

When the $\overline{\text{RESET}}$ signal is input, the contents for the modulo registers become undefined. Therefore, set data in these registers by an initialization program, and then enable PWM output.

Caution: Do not set the values from 0000H to 00FFH in the PWM modulo register (PWMn; n = 0 or 1). Set the values from 0100H to FFFFH in the PWMn register. The duty factor for the PWM signal that can be output is 17/4096 to 4096/4096.

8.3 PWM Output Unit Operation

8.3.1 Basic PWM output operation

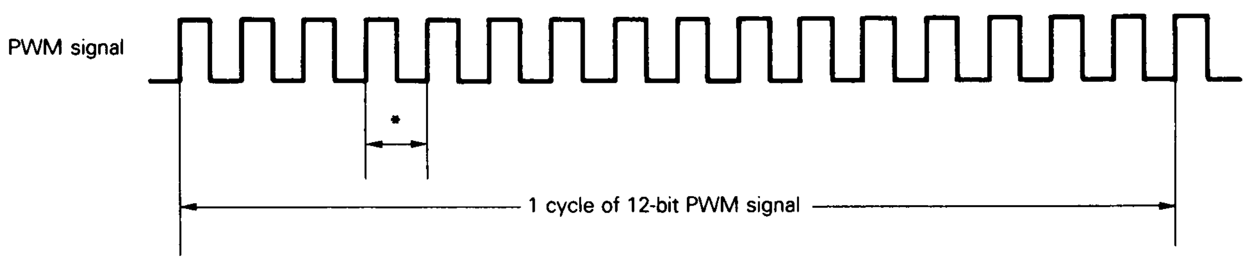
The duty factor for the PWM pulse output is determined by a value set in bits 4 through 15 for the PWM modulo register (PWMn; n = 0 or 1) as follows:

$$\text{PWM pulse output duty factor} = \frac{(\text{PWMn bits 4-15 value})^{*} + 1}{4096}$$

***:** 16 ≦ (Values for bits 4-15 of PWMn) ≦ 4095

The PWM pulse output resolution is 12 bits. This is done by outputting an 8-bit resolution PWM signal, which has a repeat cycle $f_{CLK}/256$ (42.7 μs, 23.4 kHz: f_{CLK} = 6 MHz), 16 times. By adding a pulse ($1/f_{CLK}$) to the 8-bit resolution PWM pulse, which is determined by bits 8 through 15 for the PWMn register,during each cycle, in accordance with the values for bits 4 through 7 of the PWMn register, the PWM pulse signal is generated once in 16 cycles.

Fig. 8-3 Basic PWM Output Operation



***:** PWM pulse 1 cycle, 8-bit resolution.

8.3.2 Enabling/disabling PWM pulse output

To output PWM pulses, set data in the PWM modulo registers, and then set the EN0 and EN1 bits for the PWMC register to 1.

This outputs PWM pulses, whose active levels are specified by the ALV0 and ALV1 bits for the PWMC register, from the PWM output pins.

When the EN0 and EN1 bits for the PWMC register are cleared to 0, the PWM output unit immediately stops its PWM output operation, and the PWM output pins enter the states specified by the PM1, P1, and PUO registers.

That is, when the PM1n bit (n = 0 or 1) for port 1 mode register (PM1) is 0, the corresponding PWM output pin is set in the output mode, and the content of the pin is as specified by the P1n (n = 0 or 1) bit. When the PM1n bit is 1, and when the PUO1 bit for the pull-up resistor option register is 1, the PWM pin level is made high by an internal pull-up resistor. When the PUO1 bit is 0, the PWM pin enters the output high-impedance state.

8.3.3 Specifying active level for PWM pulse

The ALV0 and ALV1 bits for the PWMC register specify the active levels for the PWM pulses output from the PWM output pins.

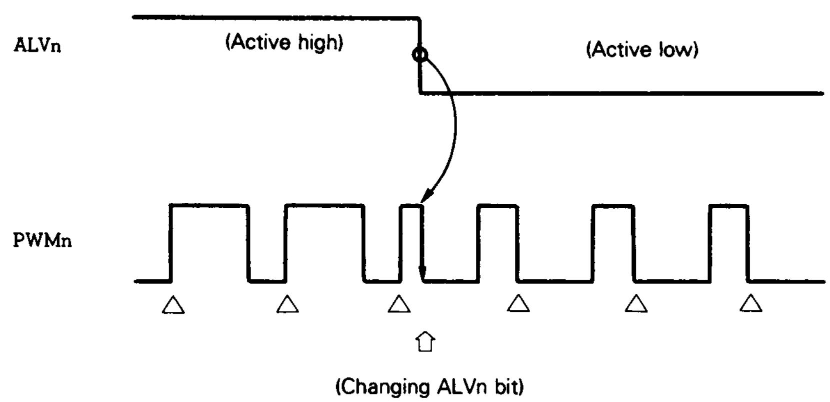
When ALV0 and ALV1 bits are set to 1, the active levels for the PWM pulses are specified to be high levels. When these bits are cleared to 0, low-active PWM pulses are output.

When the contents for ALV0 and ALV1 bits are changed, the active level for the PWM pulses are immediately changed accordingly. Fig. 8-4 shows the relation between the active levels and pin states.

In this figure, the ALVn (n = 0 or 1) bit content is changed, while the ENn (n = 0 or 1) bit for the PWMC register is set to 1, enabling the PWM output.

The pin state does not change, even if the ALVn bit content is changed, while the ENn bit is 0.

Fig. 8-4 Setting Active Level for PWM Output



Remarks: ENn = 1 (n = 0 or 1)

8.3.4 Specifying PWM pulse width changing cycle

The PWM output is started and PWM pulse width is changed every 16 PWM pulse cycles ($2^{12}/f_{CLK}$) or every other PWM pulse cycle ($2^8/f_{CLK}$). The cycle at which the PWM pulse width is changed is specified by the SYNn (n = 0 or 1) bit for the PWMC register.

When the SYNn bit is cleared to 0, the pulse width is changed every 16 PWM pulse cycles ($2^{12}/f_{CLK}$). Therefore, up to 2^{12} clocks (683 μs at f_{CLK} = 6 MHz) are required, until a pulse having the width specified by the data written to the PWM modulo register is output.

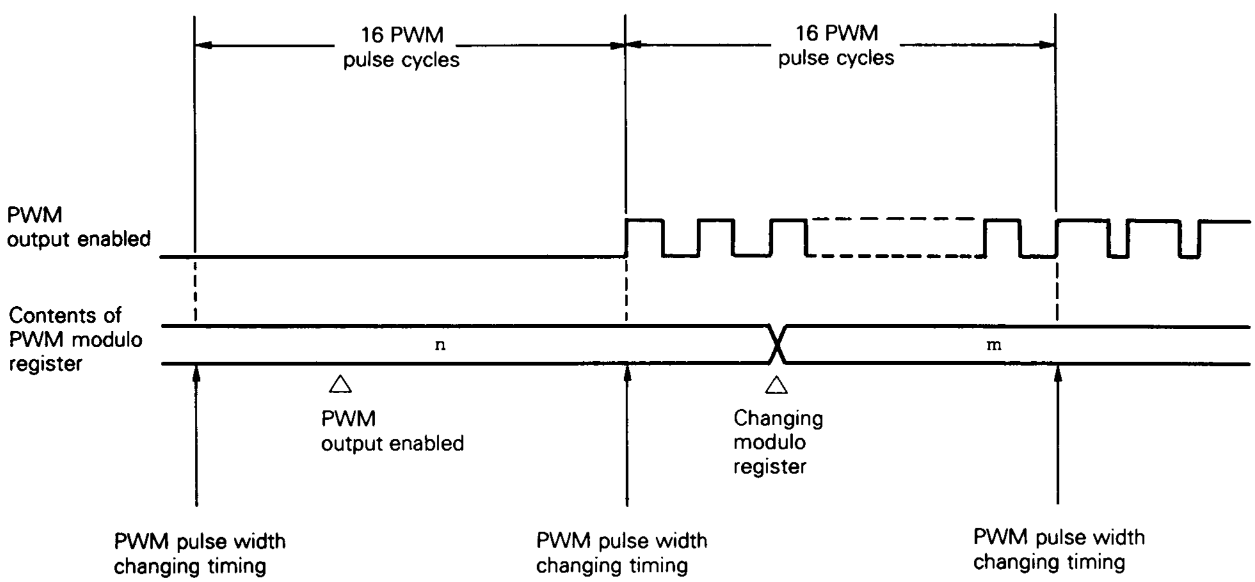
Fig. 8-5 shows a PWM output timing example.

When the SYNn bit is set to 1, the pulse width is changed every other PWM pulse cycle ($2^8/f_{CLK}$). In this case, up to 2^8 clocks (43 μs at f_{CLK} = 6 MHz) are required, until the pulse with the width specified by the data written to the PWM modulo register is output.

If the PWM pulse changing cycle is specified to be $2^8/f_{CLK}$ (i.e., when the SYNn bit is set to 1), the PWM pulse accuracy is more than 8 bits and less than 12 bits, which is lower than the accuracy when the changing cycle is specified to be $2^{12}/f_{CLK}$.

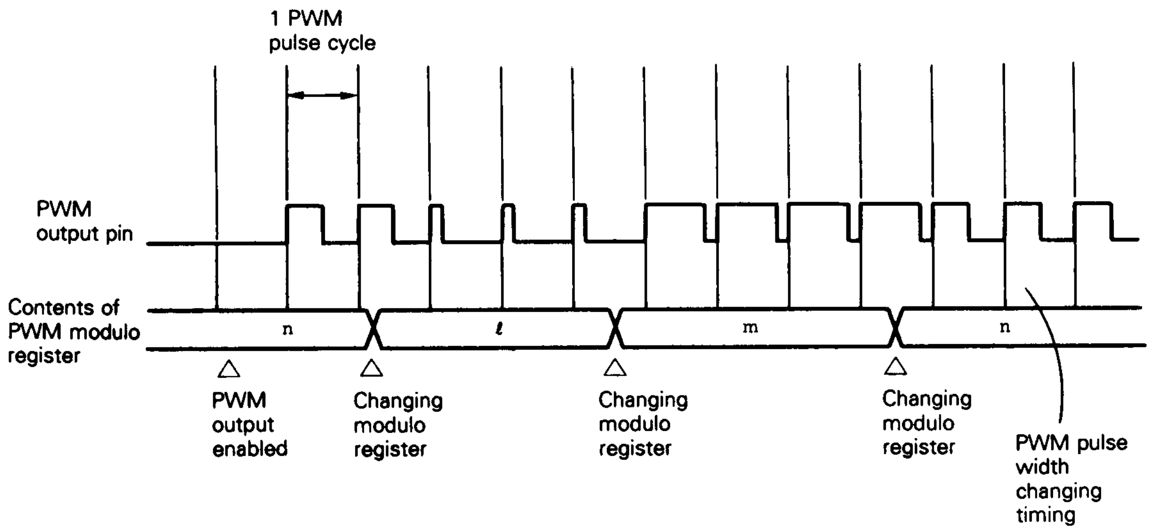
Fig. 8-6 shows a PWM output timing example, with the changing timing specified to be $2^8/f_{CLK}$.

Fig. 8-5 PWM Output Timing Example 1
(PWM pulse width changing cycle: $2^{12}/f_{CLK}$)



Caution: 1. The pulse width is changed every other PWM pulse cycle.
2. The PWM pulse accuracy is 12 bits.

Fig. 8-6 PWM Output Timing Example 2
(PWM pulse width changing cycle: $2^8/f_{CLK}$)



Caution: 1. The pulse width is changed every other PWM pulse cycle.
2. The PWM pulse accuracy is more than 8 bits and less than 12 bits.

Remarks: l, m, and n are the PWM modulo register contents.

8.4 Notes

Do not set the value from 0000H to 00FFH in the PWM modulo register (PWMn; n = 0 or 1). Set the values from 0100H to FFFFH in the PWMn register. The duty factor for the PWM signal, that can be output, is 17/4096 to 4096/4096.

CHAPTER 9 A/D CONVERTER

μPD78234 is equipped with an analog-to-digital (A/D) converter with eight multiplexed input pins (ANI0 through ANI7).

This A/D converter uses successive approximation. The conversion result is stored in 8-bit A/D conversion result register (ADCR). Conversion can be performed at as high a speed as 20 μs (at f_{CLK} = 6 MHz, high-speed conversion) with high accuracy.

The A/D converter can be started in the following two modes:

- Hardware start: Conversion is started by inputting a trigger signal (INTP5).
- Software start: Conversion is started by setting a bit in A/D converter mode register (ADM).

The A/D converter can operate in the following two modes:

- Scan mode: In this mode, several analog signals are sequentially selected from all the input pins.
- Select mode: In this mode, only one input pin is used, from which analog signals are successively input to the converter.

These start and operation modes are specified by the ADM register. In addition, the ADM register is also used to stop the converter operation.

When the conversion result is transferred to the ADCR register, interrupt request INTAD is generated (except in the select mode started by software). Therefore, the conversion result can be successively transferred to the memory by using a macro service.

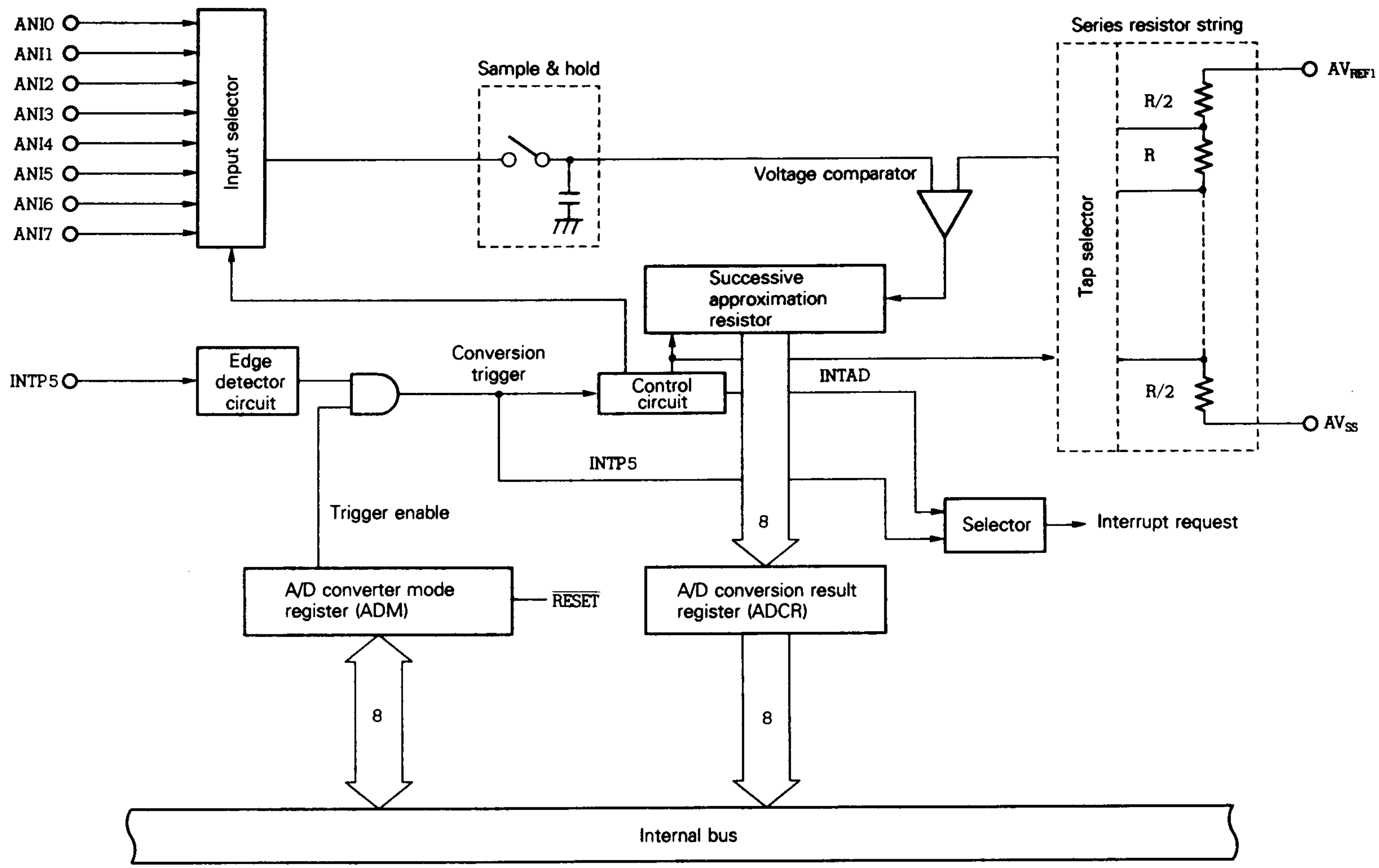
Table 9-1 INTAD Generating Mode

Start mode \ Mode	Scan mode	Select mode
Hardware start	○	○
Software start	○	–

9.1 Configuration

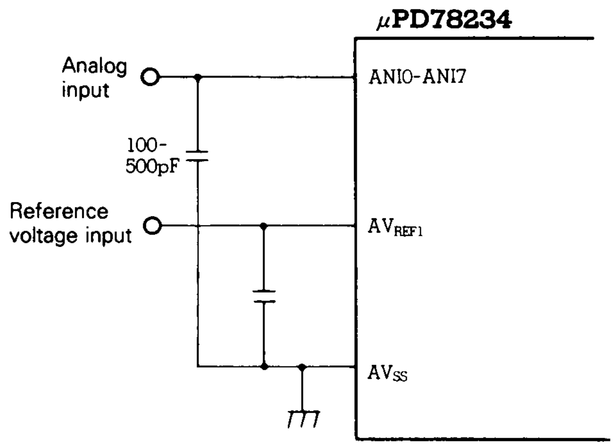
The A/D converter configuration is shown in Fig. 9-1.

Fig. 9-1 A/D Converter Configuration



Caution: 1. To prevent malfunction for analog input pins (ANI0 through ANI7) and reference voltage input pin (AV_{REF1}),insert a capacitor between them and AV_{SS} pin, as shown below.

Fig. 9-2 Connecting Capacitor



2. Make sure that a voltage exceeding or falling below the rated voltage range (AV_{SS} to AV_{REF1}) is not applied to the A/D converter input pins. For detail, refer to 9.5 Notes.

(1) Input circuit

This circuit selects an input analog signal specified by the A/D converter mode register (ADM) and sends the signal to the sample & hold circuit in accordance with a specified operation mode.

(2) Sample & hold

This circuit samples each of the analog signals successively sent from the input circuit and retains the analog signals being converted into digital signals.

(3) Voltage comparator

This circuit compares the potential difference for the input analog signal and the voltage tap for the resistor string.

(4) Series resistor string

This resistor string generates a voltage that matches the input analog signal voltage.

The resistor string is connected between the reference voltage pin (AV_{REF1}) and GND pin (AV_{SS}) for the A/D converter. It consists of 255 resistors, each having equal resistances, as well as two resistors, each having half the resistance of the other 255 resistors, so that the voltage across the AV_{REF} and AV_{SS} pins can be divided into 256 steps.

The voltage tap for the resistor string is selected by a tap selector, which is controlled by the SAR register.

(5) Successive approximation register (SAR)

This 8-bit register sets data, whose resistor string voltage tap value matches an input analog voltage value, on a bit-by-bit basis, starting from the most significant bit (MSB).

After the least significant bit (LSB) for the SAR register has been set (i.e., when A/D conversion has ended), the register contents (conversion result) are retained in the A/D conversion result register (ADCR).

(6) A/D conversion result register (ADCR)

This 8-bit register holds the A/D conversion result. Each time A/D conversion has ended, the conversion result is loaded to this register from the SAR register.

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

(7) Edge detector circuit

This circuit detects the valid edge for a signal input from interrupt request input pin INTP5, to generate an external interrupt request signal (INTP5) and the external trigger for an A/D conversion operation.

The valid edge for the INTP5 input pin is specified by external interrupt mode register 1 (INTM1) (see **Fig. 13-2**). The external trigger is enabled or disabled by the ADM register (see **9.2**).

9.2 A/D Converter Mode Register (ADM)

This 8-bit register controls A/D converter operations.

Data can be read from or written to this register by a bit manipulation or an 8-bit manipulation instruction. Figure 9-3 shows the format for this register.

Bit 0 (MS) for the ADM register controls the A/D converter operation mode.

Bits 1, 2, and 3 (ANI0, 1, and 2) select analog input signals to be converted into digital signals.

Bit 6 (TRG) enables external synchronization for A/D conversion operation. When the TRG bit is set to 1, after the CS bit has been set to 1, the conversion operation is initialized each time the valid edge is input to the INTP5 pin as an external trigger. When the TRG bit is cleared to 0, conversion is carried out, regardless of the INTP5 pin state.

Bit 7 (CS) controls the A/D conversion operation. When this bit is set to 1, the A/D converter starts operating. When the CS bit is cleared to 0, the converter stops, even if it is in the middle of an operation. At this time, however, the ADCR register contents are not updated, nor does INPTAD interrupt occur. Power supply to the voltage comparator is stopped, to reduce the current dissipated by the A/D converter.

When the $\overline{\text{RESET}}$ signal is input, the ADM register contents are initialized to 00H.

Caution: To use the STOP mode, reset the CS bit to 0, before setting the STOP mode, to reduce the current dissipation. If the CS bit remains set to 1, the conversion operation is stopped, when the STOP mode is set, but power supply to the voltage comparator is not stopped. Consequently, the current dissipation by the A/D converter does not decrease.

Fig. 9-3 A/D Converter Mode Register (ADM) Format

	⑦	6	5	4	3	2	1	0
ADM	CS	TRG	0	FR	ANIS2	ANIS1	ANIS0	MS

ANIS2	ANIS1	ANIS0	MS	Specifies A/D conversion mode	
0	0	0	0	Scan mode	Scans ANI0
0	0	1	0		Scans ANI0 and ANI1
0	1	0	0		Scans ANI0 through ANI2
0	1	1	0		Scans ANI0 through ANI3
1	0	0	0		Scans ANI0 through ANI4
1	0	1	0		Scans ANI0 through ANI5
1	1	0	0		Scans ANI0 through ANI6
1	1	1	0		Scans ANI0 through ANI7
0	0	0	1	Select mode	Selects ANI0
0	0	1	1		Selects ANI1
0	1	0	1		Selects ANI2
0	1	1	1		Selects ANI3
1	0	0	1		Selects ANI4
1	0	1	1		Selects ANI5
1	1	0	1		Selects ANI6
1	1	1	1		Selects ANI7

FR	Controls conversion speed	
0	180 clocks*	Low-speed conversion
1	120 clocks*	High-speed conversion

TRG	Controls external trigger
0	Disables external trigger
1	Enables external trigger

CS	Controls A/D conversion operation
0	Stops A/D conversion
1	Starts A/D conversion

***:** 1 clock = $1/f_{CLK} = 2/f_{XX}$

9.3 Operation

9.3.1 Basic A/D converter operation

(1) A/D conversion procedure

The A/D converter operates according to the following procedure:

(a) Analog signal input pins and the operation mode for the A/D converter are specified by the A/D converter mode register (ADM).

(b) Bit 7 (CS) for the ADM register is set to 1, to start the A/D converter.

(c) The most significant bit (bit 7) for the SAR register is automatically set to 1, as soon as conversion is started.

(d) When bit 7 for the SAR register has been set, the tap selector sets the voltage tap on the resistor string to (255/512)AV_{REF1} (≒ 1/2 AV_{REF1})

(e) The resistor string voltage tap value is compared with an input analog signal voltage by the voltage comparator. If the analog input signal voltage is greater than (1/2)AV_{REF1}, the MSB for the SAR register remains set. If it is less than (1/2)AV_{REF1}, the MSB is cleared.

(f) Next, bit 6 for the SAR register is automatically set to 1, and the voltage comparator starts comparing the next analog input signal voltage. The voltage tap for the resistor string is selected as follows, according to the bit 7 value, which has already been set:

- Bit 7 = 1 ... (383/512)AV_{REF1}≒ 3/4AV_{REF1}

- Bit 7 = 0 ... (127/512)AV_{REF1}≒ 1/4AV_{REF1}

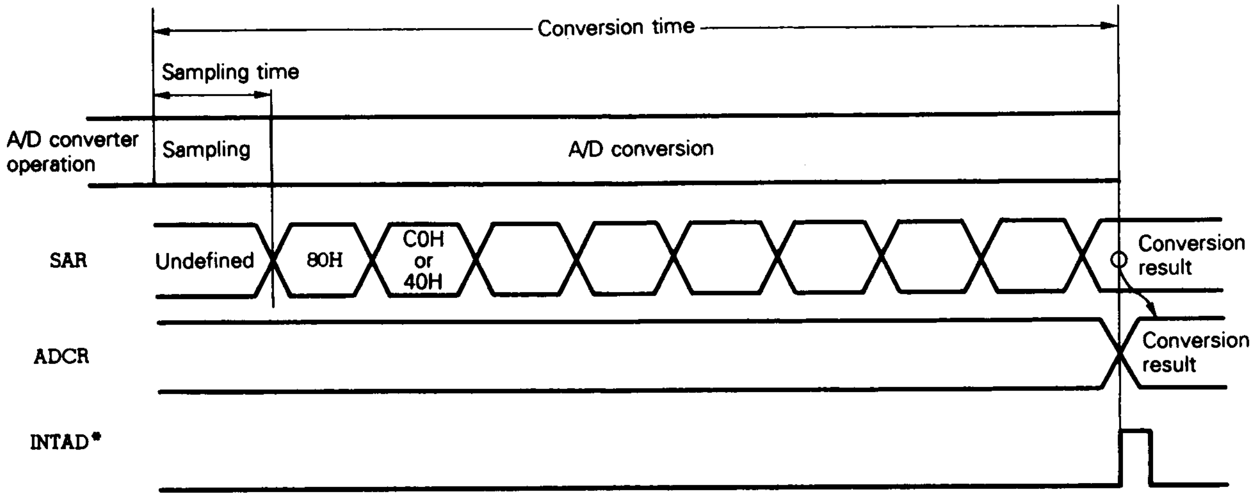
This voltage tap is compared with the input analog voltage. According to the comparison result, bit 6 for the SAR register is manipulated as follows:

- Analog input voltage ≧ voltage tap: bit 6 = 1
- Analog input voltage < voltage tap: bit 6 = 0

(g) These comparisons are successively carried out, until the least significant bit (bit 0) for the SAR register is compared (binary search method).

(h) When all the 8 bits for the SAR register have been compared, valid digital signals are set in the SAR register. These signal values are transferred to and latched in the ADCR register.
At the same time, A/D conversion end interrupt (INTAD) can be generated (except in software-started select mode). Process this INTAD interrupt as either a vector interrupt or macro service (described later).

Fig. 9-4 Basic Operation for A/D converter



*: Except the select mode initiated by software start.

A/D conversions are continuously performed, until the CS bit is cleared to 0 by software.
When data is written to the ADM register, while conversion is in progress, the conversion is initialized. If the CS bit is set to 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 (ANI0 through ANI7) are related to the results of the A/D conversion (value stored in ADCR), as follows:

$$ADCR = INT(\frac{V_{IN}}{AV_{REF1}} \times 256 + 0.5)$$

or,

$$(ADCR - 0.5) \times \frac{AV_{REF1}}{256} \leqq V_{IN} < (ADCR + 0.5) \times \frac{AV_{REF1}}{256}$$

where, INT ( ) : function returning integer of value in ( )

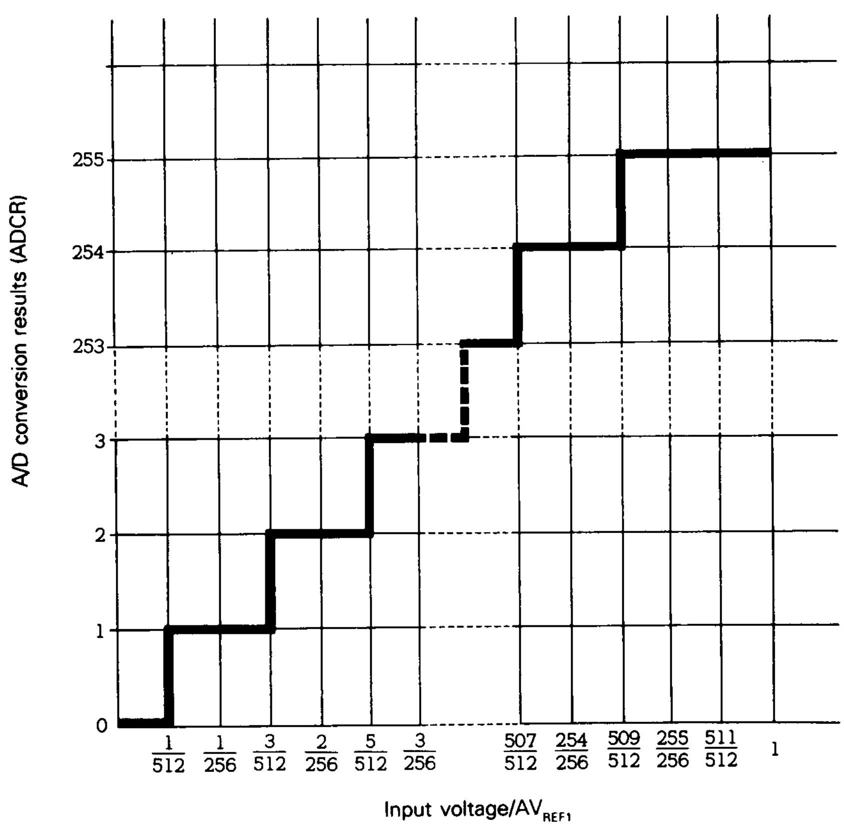
V_{IN} : analog input voltage

AV_{REF1} : AV_{REF1} pin voltage

ADCR : ADCR register value

Figure 9-5 shows the relations between the analog input voltages and the A/D conversion results.

Fig. 9-5 Relations between Analog Input Voltage and A/D Conversion Results



(3) A/D conversion time

The time required for the A/D conversion is determined by the system clock frequency (f_{CLK}) and the FR bit for the ADM register.

This A/D conversion time includes all the time required for one A/D conversion operation and the sampling time.

Table 9-2 shows the conversion time and sampling time values.

Table 9-2 A/D Conversion Time

System clock (f_{CLK}) range	FR bit	Conversion time	Sampling time
2 MHz < f_{CLK} ≦ 6 MHz	0	180/f_{CLK} (30 μs - 90 μs)	36/f_{CLK} (6 μs - 18 μs)
2 MHz ≦ f_{CLK} ≦ 6 MHz	1	120/f_{CLK} (20 μs - 60 μs)	24/f_{CLK} (4 μs -12 μs)

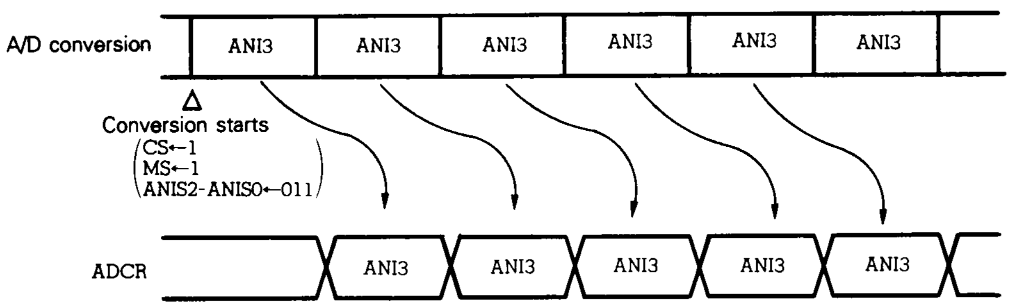
9.3.2 Select mode

Bits 1 through 3 (ANIS0 through ANIS2) for the ADM register specify one analog input pin. The analog voltage input to the specified pin is converted into a digital signal. The resultant digital signal is stored in the A/D conversion result register (ADCR).

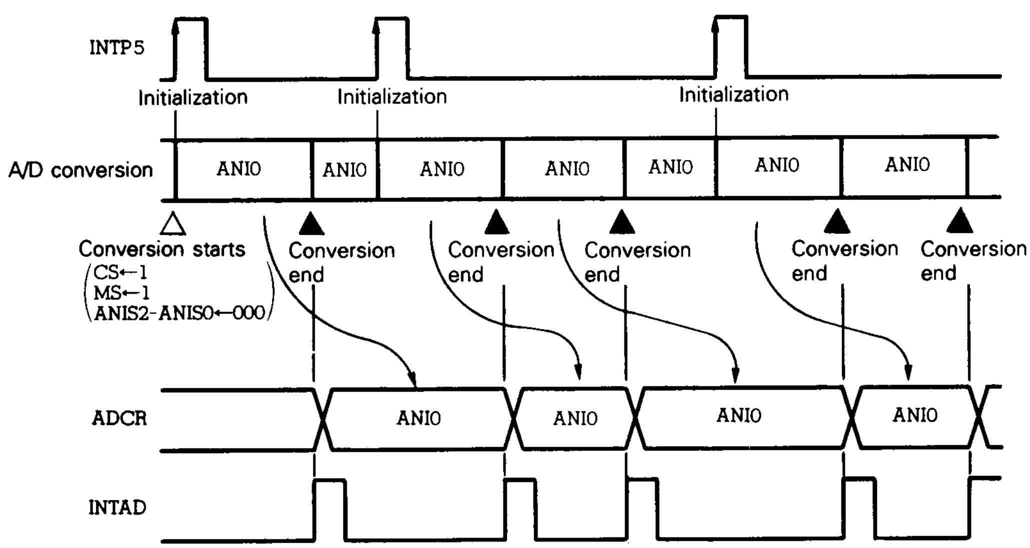
If bit 6 (TRG) for the ADM register is set, thus enabling external trigger at this time, A/D conversion end interrupt INTAD are generated.

Fig. 9-6 Operation Timing in Select Mode

(a) TRG bit←0



(b) TRG bit←1



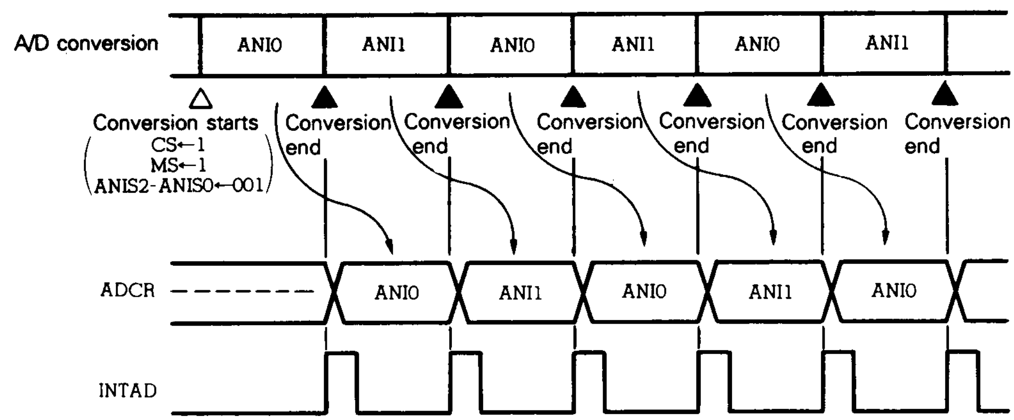
9.3.3 Scan mode

In this mode, signals input from the analog input pins, specified by bits 1 through 3 (ANIS0 through ANIS2) for the A/D converter mode register (ADM), are successively selected and converted into digital signals.

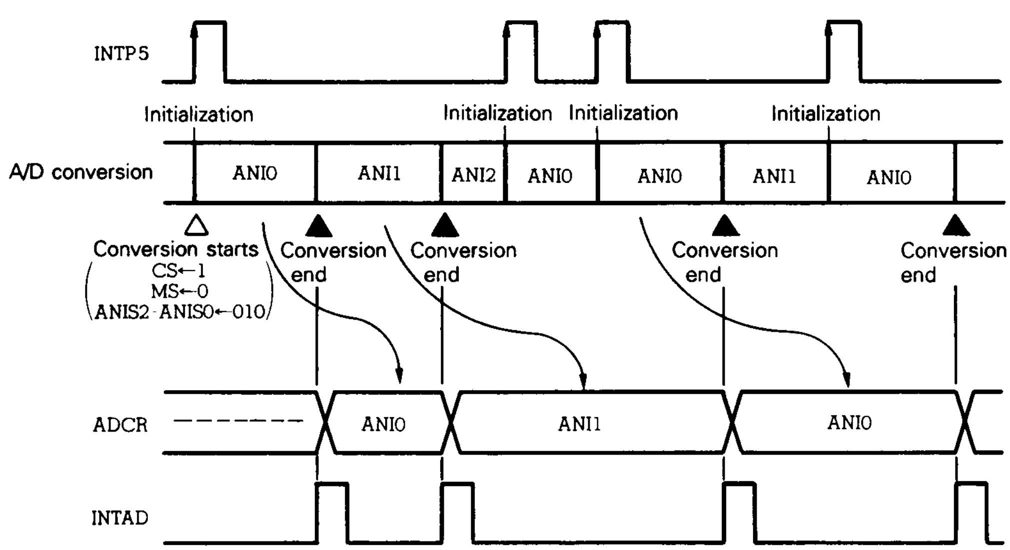
For example, when the ANIS2 to ANIS0 bits for the ADM register are 001, the ANI0 and ANI1 pins are repeatedly scanned, starting from the ANI0 pin, followed by the ANI1 pin, then by ANI0 pin again, and so on. In this mode, each time an input analog signal conversion has ended, the conversion result is stored in the ADCR register, and A/D conversion end interrupt request INTAD are generated.

Fig. 9-7 Operation Timing in Scan Mode

(a) TRG bit←0



(b) TRG bit←1



Caution 1: When the result of the A/D conversion is read by using a vector interrupt with the scan mode of the A/D converter used, if the A/D conversion end interrupt is kept pending for a long time by processing of the other interrupt (180 clocks when the FR bit is 0, and 120 clocks or longer when the FR bit is 1), the conversion result cannot be accurately measured. To measure the conversion result accurately, take the following measures:

- **Keep the processing time of the otehr interrupts shorter than the A/D conversion time.**
- **Use multiplexed interrupt so that the A/D conversion end interrupt can be accepted even while other interrupts are processed.**
- **Use a macro service to generate the A/D conversion end interrupt.**

The A/D conversion end interrupt is also kept pending by the causes described in 14.3.5 Interrupt request and maco service pending, in addition to the other interrupts.
Of the measures described above, use of the macro service is considered to be the simplest.

2: When the ADM register is set after registers related to interrupt have been set in the scan mode, an unwanted interrupt may occur after the ADM register has been set. To prevent this, do as follows:

- **Write to the ADM register.**
- **Reset the interrupt request flag (PIF5) to 0.**
- **Set the interrupt mask flag or interrupt service mode flag.**

9.3.4 Starting A/D conversion by software

The A/D conversion can be started by software by clearing the TRG bit for the ADM register to 0 and by writing 1 to the CS bit for the ADM register.

If a value that resets the TRG bit to 0 and the CS bit to 1 is written to the ADM register, while the A/D conversion is in progress (the CS bit is 1), the ongoing A/D conversion is stopped, and the A/D conversion is started according to the newly written value.

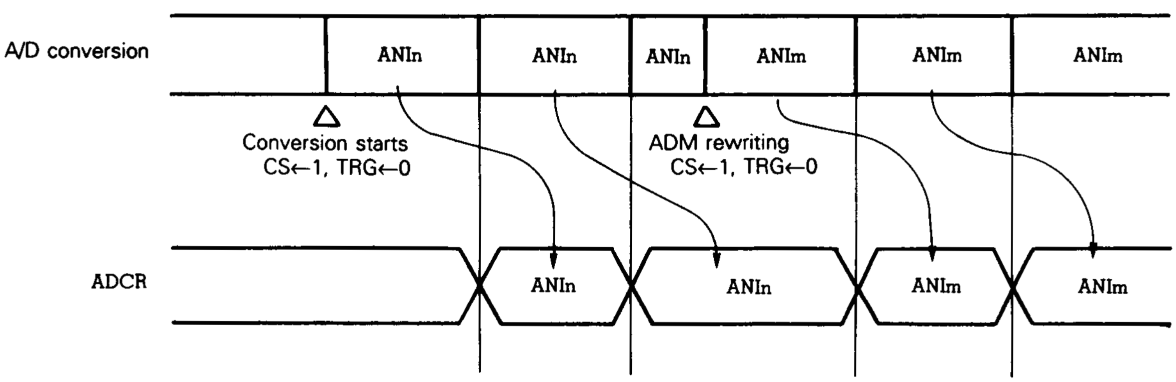
After one A/D conversion operation has finished, the next conversion is immediately started, in accordance with an operation mode set by the ADM register. The conversion is repeated until an instruction that writes the ADM register is executed.

If the A/D conversion is started by software (TRG bit is 0), the INTP5 (P26 pin) input does not affect the conversion operation.

(1) A/D conversion in select mode

In this mode, the analog signal, input to the pin selected by the ADM register, is converted. When the conversion has finished, the analog signal for the same pin is converted again. Interrupt request INTAD does not occur, even when the conversion has finished.

Fig. 9-8 A/D Conversion Started by Software in Select Mode

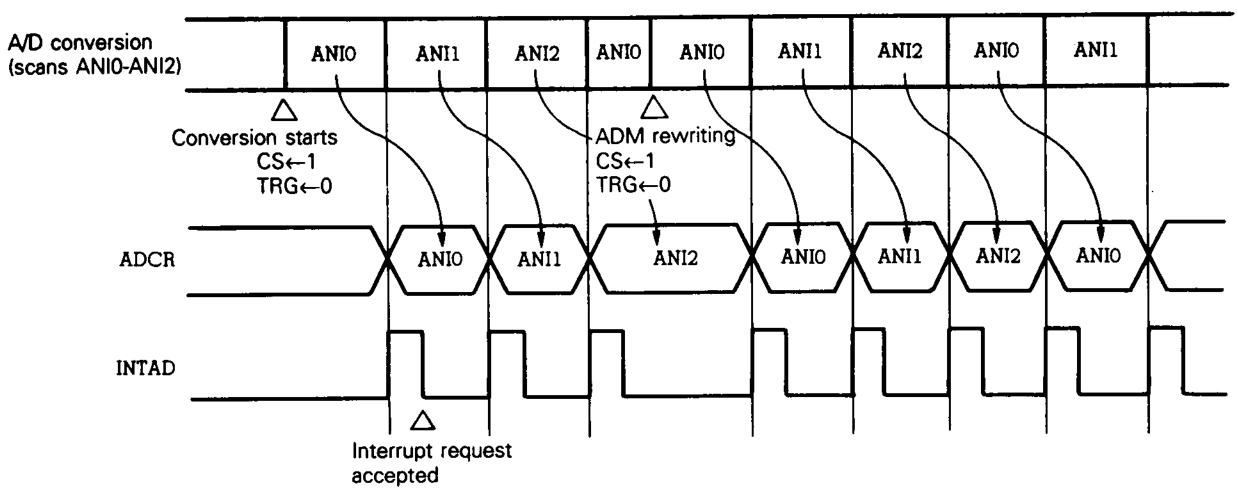


Remarks: n = 0, 1, ..., 7
m = 0, 1, ..., 7

(2) A/D conversion in scan mode

When the conversion is started, the signal input to the ANI0 pin is converted. When the conversion has finished, the signal for the next analog input pin is converted. Each time the conversion has finished, interrupt request INTAD is generated.

Fig. 9-9 A/D Conversion Started by Software in Scan Mode



9.3.5 Starting A/D conversion by hardware

The A/D conversion can be started by hardware by setting both the TRG bit and CS bit for the ADM register to 1. When the TRG bit and CS bit for the ADM register are set to 1, the A/D converter waits for the input of an external signal. The A/D conversion is started when the valid edge is input to the INTP5 pin (P26 pin).

If the valid edge is input to the INTP5 pin again, after the A/D conversion has been started by the valid edge input to the INTP5 pin, the A/D conversion in progress is stopped. The conversion is started from the beginning again, in accordance with the current ADM register contents.

If a value that sets the TRG bit and CS bit to 1 is written to the ADM register again, during the A/D conversion (CS bit is 1), the current conversion operation is stopped (even while the converter is waiting for the input of an external signal). The converter stands by, until the valid edge is input to the INTP5 pin in the A/D conversion operation mode, in accordance with the written value. When the valid edge has been input, the conversion is started.

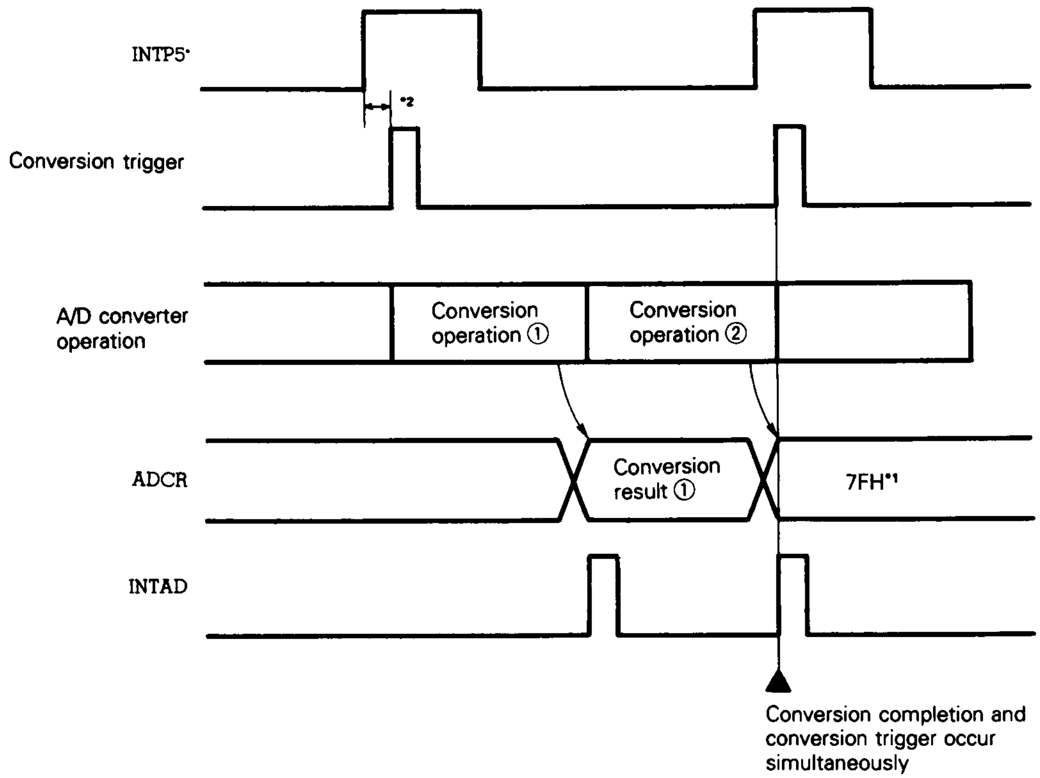
By using this function, the A/D conversion can be carried out in synchronization with an external signal.

When the A/D conversion has finished, the next conversion operation is immediately started in the operation mode set by the ADM register (the converter does not wait for the input from the INTP5 pin). The conversion is repeatedly executed, until an instruction that writes the ADM register is executed or until the valid edge is input to the INTP5 pin.

Caution 1: Eight to twelve system clocks are required, from when the valid edge is input to the INTP5 pin until the A/D conversion is actually started. Take this delay time into consideration when designing the circuit. For details on the edge detection function, refer to Chapter 13 Edge Detection Function.

2: When hardware start is used for starting A/D converter operation, if A/D conversion is initiated by inputting an effective edge to the INTP5 pin, and if an effective edge is again input to the INTP5 pin during the A/D conversion, an erroneous A/D conversion operation may result. Specifically, this holds true after an A/D conversion, if an effective edge is input to the INTP5 pin when storing the conversion result to the A/D conversion result register (ADCR). In this case, the A/D conversion end interrupt (INTAD) is generated.
However, the value stored into the ADCR register is not the conversion result. Instead, it is always 7FH (refer to Fig. 9-10).

Fig. 9-10 An Example of Erroneous A/D Converter Operation Initiated by Hardware Start



***1:** To be more specific, the conversion operation (2) result is stored. However, the value becomes 7FH, due to erroneous operation.

***2:** Time from when INTP5 input is changed to when it is determined as an effective edge. Refer to **CHAPTER 13 EDGE DETECTION FUNCTION**.

In order to avoid this discrepancy, the A/D converter mode register (ADM) must be set again after performing necessary A/D conversion by hardware.
This discrepancy also occurs in the in-circuit emulation in the same way.

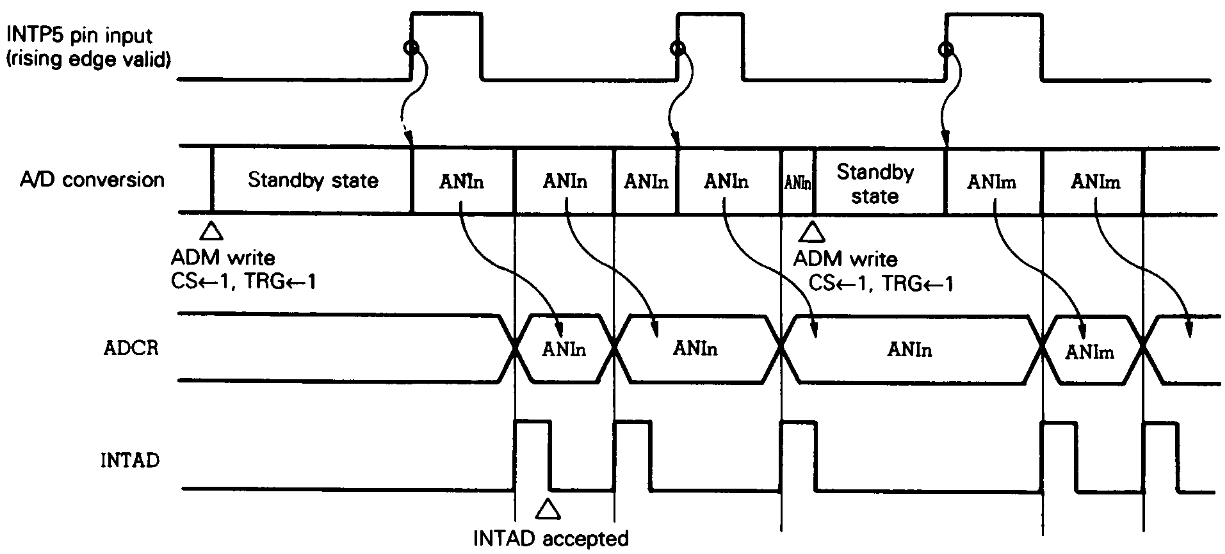
Caution 3: The digital noise of the INTP5 pin cannot be rejected normally with an in-circuit emulator. When A/D conversion is selected to be started by hardware, A/D conversion is started by an erroneously detected edge. For detail on erroneous detection of an edge, refer to 13.4 Notes in CHAPTER 13 EDGE DETECTION FUNCTION.

(1) A/D conversion in select mode

The analog signal for the input pin, selected by the ADM register, is converted into a digital signal. When the A/D conversion has finished, the analog signal for the same input pin is converted again. Each time the conversion has finished, interrupt request INTAD is generated.

When valid edge is input to the INTP5 pin, during the A/D conversion, the conversion in progress is stopped once, and the new conversion operation is started.

Fig. 9-11 A/D Conversion Started by Hardware in Select Mode



Remarks: n = 0, 1, ..., 7

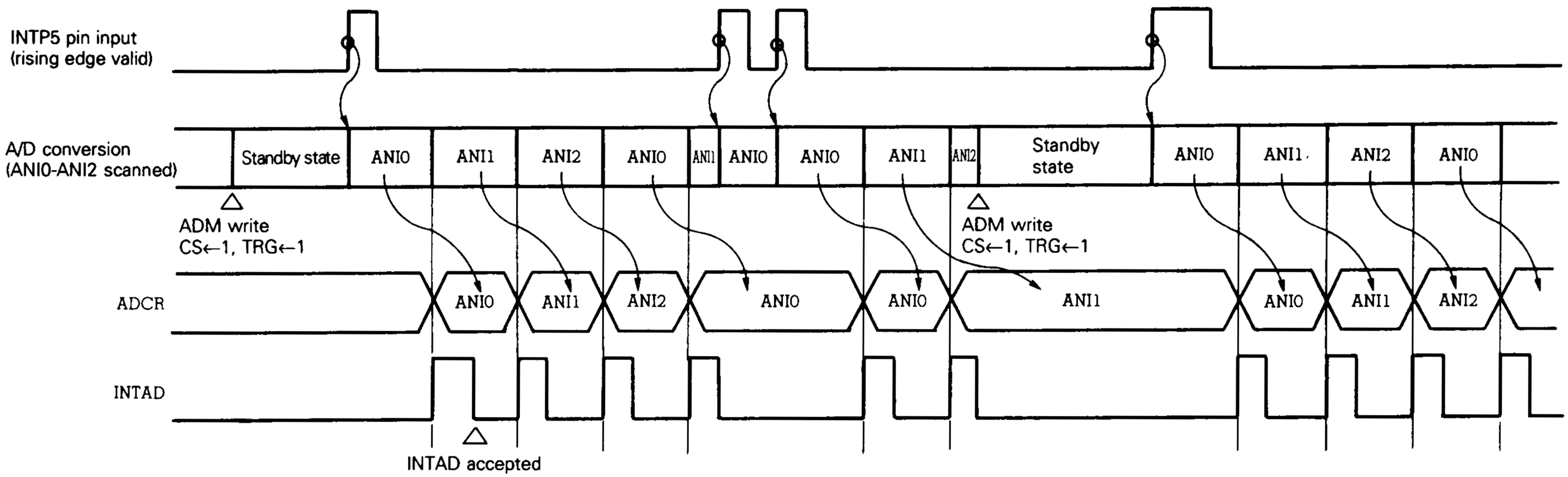
m = 0, 1, ..., 7

(2) A/D conversion in scan mode

When the A/D conversion is started, the analog signal input to the ANI0 pin is converted. When the conversion has finished, the signal from the next analog input pin is converted. Each time the conversion has finished, interrupt request INTAD is generated.

When the valid edge is input to the INTP5 pin, during the A/D conversion, the current conversion operation is stopped, and the ANI0 pin signal conversion is started.

Fig. 9-12 A/D Conversion Started by Hardware in Scan Mode



9.4 External Interrupt for A/D Converter

The A/D converter, except in select mode, generates A/D conversion end interrupt request (INTAD), each time it has finished conversion.

The interrupt control flags for the INTAD interrupt are shared with the interrupt control flags for external interrupt INTP5. Therefore, the interrupt request is generated in the timing shown in Table 9-3 in accordance with the operation state for the A/D converter specified by the ADM register.

The interrupt caused by INTAD is controlled by interrupt control registers in the same manner as the interrupt caused by INTP5. For details, refer to **CHAPTER 14 INTERRUPT FUNCTION**.

Table 9-3 Interrupt Request Generating Conditions in Each A/D Converter Operation Mode

A/D converter operation flag	Interrupt request	Mask flag	Interrupt request	Interrupt generating condition
Stop	PIF5	PMK5	INTP5	Input valid edge to INTP5 pin
Scan mode			INTAD	A/D conversion end
Select mode			INTP5	Input valid edge to INTP5 pin
A/D conversion started by hardware			INTAD	A/D conversion end

9.5 Notes

(1) Range of voltage applied to analog input pins

When using the A/D converter input pins ANI0 through ANI7 (P70 through P77), observe the following point:

- Do not apply a voltage outside the AV_{SS} to AV_{REF1} range to the pin whose signal is converted during the A/D conversion.

Unless this point is abided by, μPD78234 may be destroyed.

(2) A/D conversion started by hardware

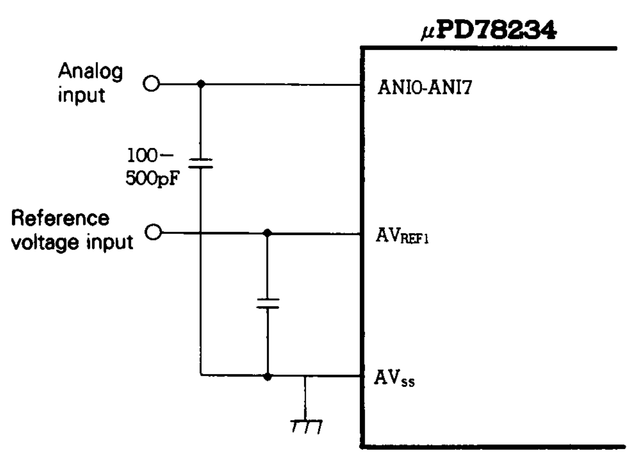
(a) Eight to twelve system clocks are required, from when the valid edge is input to the INTP5 pin until the A/D conversion is actually started. Take this delay time into consideration, when designing the circuit. For details regarding the edge detection function, refer to **CHAPTER 13 EDGE DETECTION FUNCTION**.

(b) In the in-circuit emulator, the INTP5 pin cannot normally perform digital noise elimination. After the hardware start mode is selected to start A/D conversion, A/D conversion may be initiated by an erroneously detected edge. This must be taken into concideration when performing A/D conversion. Refer to **13.4 Notes** in **CHAPTER 13 EDGE DETECTION FUNCTION** for details of erroneous edge detection.

(3) Capacitor connected to analog input pin

Connect a capacitor between the analog input pins (ANI0 through ANI7) and the AV_{SS} pin, and between reference voltage input pin (AV_{REF1}) and AV_{SS}, to prevent malfunctioning due to noise.

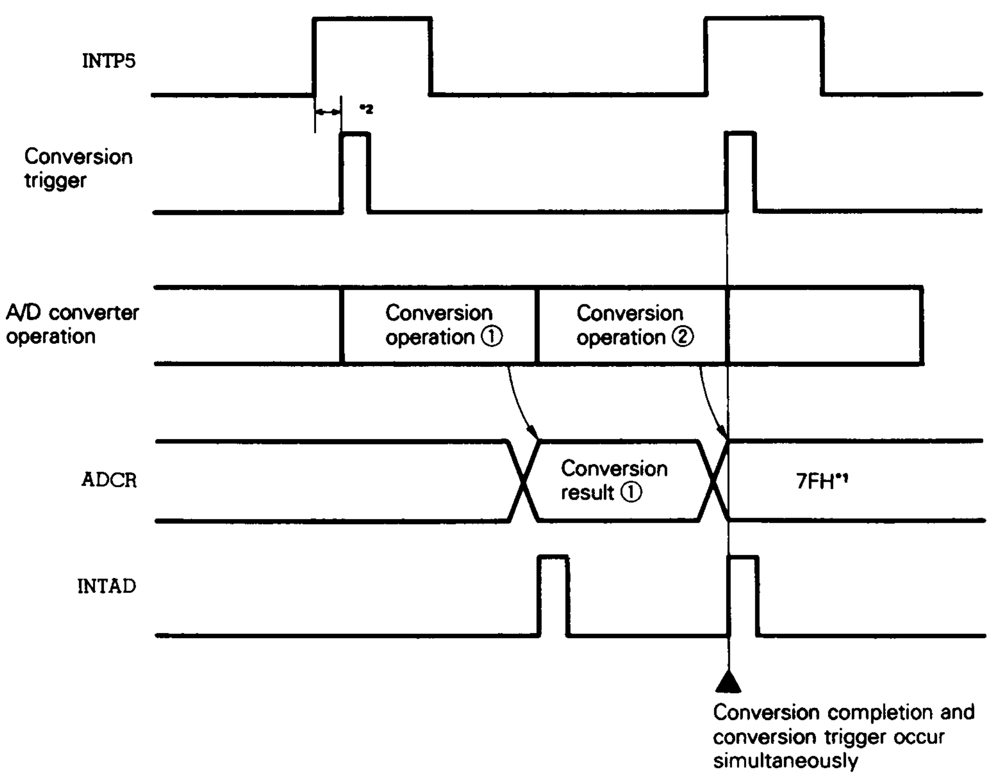
Fig. 9-13 Connecting Capacitor for A/D Converter



(4) To use the STOP mode, reset the CS bit to 0, before setting the STOP mode, to reduce the current dissipation. If the CS bit remains set to 1, the conversion operation is stopped, when the STOP mode is set, but power supply to the voltage comparator is not stopped. Consequently, the current dissipation by the A/D converter does not decrease.

(5) When hardware start is used for starting A/D converter operation, if A/D conversion is initiated by inputting an effective edge to the INTP5 pin, and if an effective edge is again input to the INTP5 pin during the A/D conversion, an erroneous A/D conversion operation may result. Specifically, after an A/D conversion, if an effective edge is input to the INTP5 pin when storing the conversion result to the A/D conversion result register (ADCR). In this case, the A/D conversion end interrupt (INTAD) is generated.
However, the value stored into the ADCR register is not the conversion result, but always 7FH (refer to **Fig. 9-14**).

Fig. 9-14 An Example of Erroneous Operation of A/D Converter Initiated by Hardware Start



***1:** To be more specifically, the conversion result of conversion operation (2) is stored; however, the value becomes 7FH due to erroneous operation.
***2:** Time from when INTP5 input is changed to when it is determined as an effective edge. Refer to **CHAPTER 13 EDGE DETECTION FUNCTION**.

In order to avoid this discrepancy, the A/D converter mode register (ADM) must be set again after performing necessary A/D conversion by hardware.
This discrepancy also occurs in the in-circuit emulation in the same way.

(6) When the result of the A/D conversion is read by using a vector interrupt with the scan mode of the A/D converter used, if the A/D conversion end interrupt is kept pending for a long time by processing of the other interrupt (180 clocks when the FR bit is 0, and 120 clocks or longer when the FR bit is 1), the conversion result cannot be accurately measured. To measure the conversion result accurately, take the following measures:

- Keep theprocessing time of the other interrupts shorter than the A/D conversio time.
- Use multiplexedinterrupt so that the A/D conversion end interrupt can be accepted even while other interrupts are processed.
- Use a macro service to generate the A/D conversion end interrupt.

The A/D conversion end interrupt is also kept pending by the causes described in **14.3.5 Interrupt request and maco service pending**, in addition to the other interrupts.
Of the measures described above, use of the macro service is considered to be the simplest.

(7) When the ADM register is set after registers related to interrupt have been set in the scan mode, an unwanted interrupt may occur after the ADM register has been set. To prevent this, do as follows:

- Write to the ADM register.
- Reset the interrupt request flag (PIF5) TO 0.
- Set the interrupt mask flag or interrupt service mode flag.

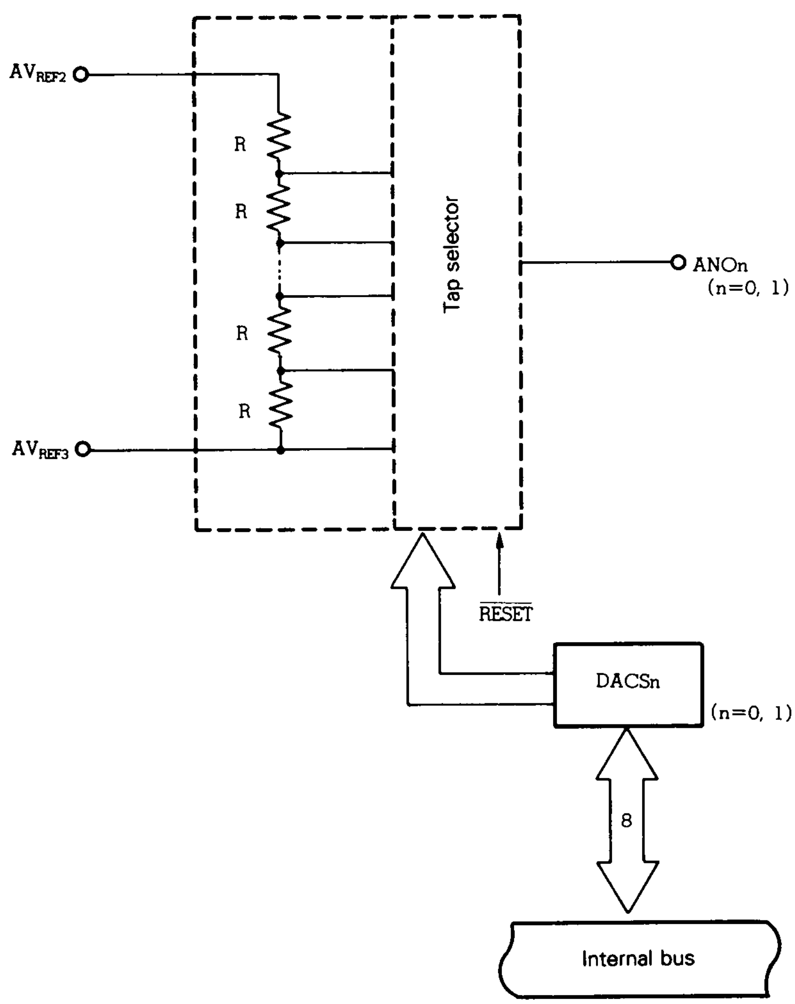
CHAPTER 10 D/A CONVERTER

μPD78234 contains two voltage output digital-to-analog (D/A) converters circuits, each having an 8 bit resolution. These D/A converters have a resistor string format.

10.1 Configuration

D/A converter configuration is as shown in Fig. 10-1.

Fig. 10-1 D/A Converter Configuration (n = 0 or 1)



(1) D/A conversion value setting registers (DACSn, n = 0 or 1)

These registers set voltage values to be output to the ANOn pins (n = 0 or 1). The voltages output to the ANOn pins are determined by the following expression:

$$ANOn = \frac{AV_{REF2} - AV_{REF3}}{256} \times DACSn + AV_{REF3} \text{ [v]}$$

When the $\overline{\text{RESET}}$ signal is input, these register contents are initialized to 00H.

(2) Resistor string

The resistor string consists of 256 resistors, all having equal resistances, connected in series. The resistor string ends are connected to the AV_{REF2} and AV_{REF3} pins. Two independent resistor strings are provided, one for the ANO0 pin and the other for the ANO1 pin.

(3) Tap selector

One of the 256 taps for the resistor string is selected by the DACSn register value and connected to the ANOn pin.

When the $\overline{\text{RESET}}$ signal is at low level, no tap is connected to the ANOn pin, making the pin in the output high-impedance state.

10.2 D/A Converter Operation

By writing a value to be output in the D/A conversion value setting register (DACSn, n = 0 or 1), an analog voltage, equivalent to the value written in the DACSn register, is immediately output from the ANOn pin (n = 0 or 1). The output voltage is maintained, until a new value is written to the DACSn register.

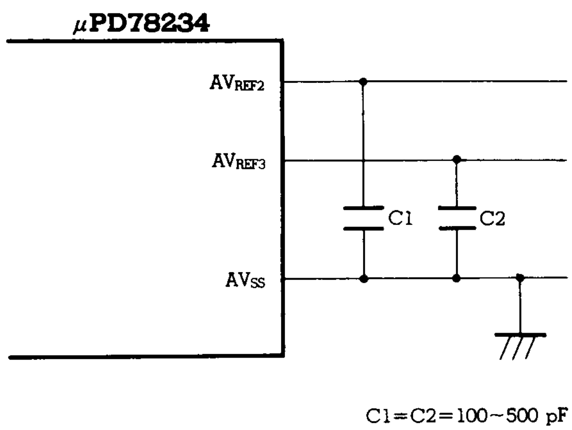
The voltage output from the ANOn pin is determined by the following expression:

$$ANOn = \frac{AV_{REF2} - AV_{REF3}}{256} \times DACSn + AV_{REF3} \text{ [v]}$$

While the $\overline{\text{RESET}}$ signal is at low level, the ANOn pin is in the output high-impedance state, and the DACSn register contents are initialized to 00H. When the $\overline{\text{RESET}}$ signal has gone high, a signal with the same level as that for the AV_{REF3} signal is output from the ANOn pin.

Connect a capacitor across the reference voltage input pin (AV_{REF2}, AV_{REF3}) and AV_{SS} pins to stabilize the operation of the D/A converter.

Fig. 10-2 Example of Connection of Capacitor to Reference Voltage Input Pin of D/A Converter



10.3 Notes

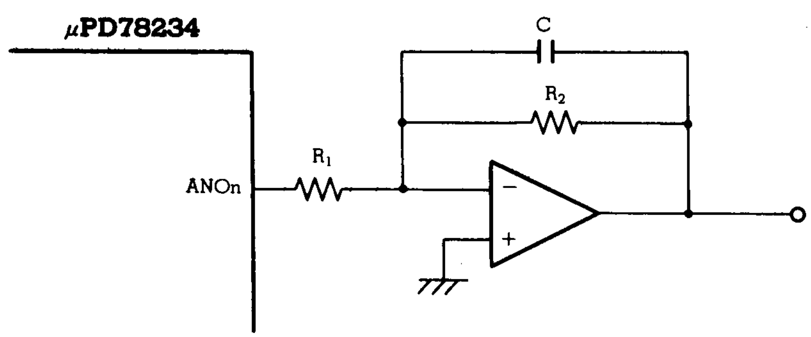
(1) Since the output impedance for the D/A converter is high, the ANOn pin (n = 0 or 1) cannot output a current. If the load input impedance is low, insert a buffer amplifier between the load and the ANOn pin. Keep the wiring of the buffer amplifier and load as short as possible (because the output impedance is high). If long wiring is unavoidable, enclose the wiring in ground patterns.

(2) The D/A converter output voltage changes stepwise. Filter the D/A converter output signal with a lowpass filter.

(3) The μPD78234 D/A converter enters the output high-impedance state while the $\overline{\text{RESET}}$ signal is low. Design the load circuit so that the input impedance can be high.

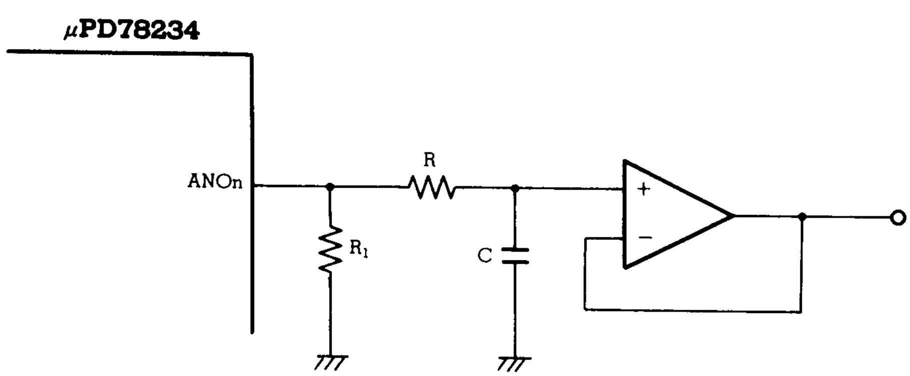
Fig. 10-3 Inserting Buffer Amplifier

(a) Inverting amplifier



- The input impedance for the buffer amplifier is R_1.

(b) Voltage follower



- The buffer amplifier input impedance is R_1.
- Were it not for R_1, the output signal would become undefined, when the $\overline{\text{RESET}}$ signal is low.

(4) The D/A converter outputs the same level as that for the AV_{REF3} pin, when the RESET signal has been removed. Take this into account.

CHAPTER 11 ASYNCHRONOUS SERIAL INTERFACE

As an asynchronous serial interface, UART (Universal Asynchronous Receiver/Transmitter) is provided. This interface transfers 1-byte data following a start bit and is capable of full-duplex transmission.

A UART baud rate generator is also provided, that allows data to be communicated at a wide baud rate range.

In addition, the baud rate can also be defined by dividing the frequency of the clock input to the ASCK pin.

When the UART baud rate generator is used, the MIDI standard baud rate (31.25 kbps) can also be obtained.

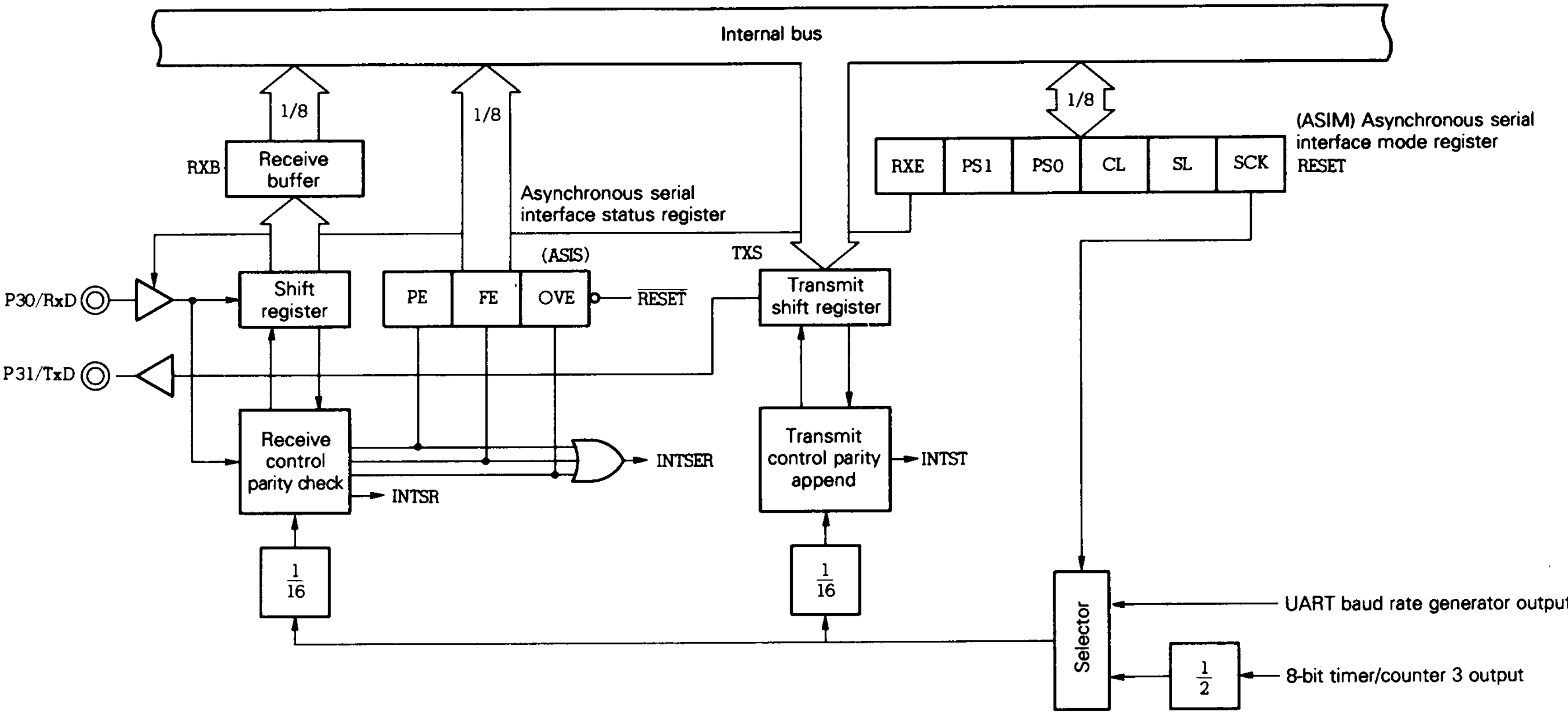
The clock-synchronized serial interface operates independently.

11.1 Configuration

Figure 11-1 shows the asynchronous serial interface configuration.

For the baud rate generator details, refer to **11.4 Baud Rate Generator**.

Fig. 11-1 Asynchronous Serial Interface Configuration



(1) Receive buffer (RXB)

This register holds the receive data. Each time 1 byte of data has been received, the receive data is transferred to this register from the shift register.

When the data length is specified to be 7 bits, the receive data is transferred to bits 0 through 6 for RXB. The MSB for RXB is always "0".

This register's contents can only be read by an 8-bit manipulation instruction. When the $\overline{\text{RESET}}$ signal has been input, the RXB contents become undefined.

(2) Transfer shift register (TXS)

This register sets the data to be transferred. The data written to TXS is transferred as serial data.

When the data length is specified to be 7 bits, bits 0 through 6 in the data written to the TXS register are treated as the transfer data. When data is written to the TXS register, transfer is started. Do not write the TXS register, while the transfer is in progress.

This register can only be written by an 8-bit manipulation instruction. When the $\overline{\text{RESET}}$ signal has been input, the TXS contents become undefined.

(3) Shift register

This register converts the serial data input to the RxD pin into parallel data. When 1 byte of data has been received, the receive data is transferred to the receive buffer.

The shift register cannot be manipulated directly from the CPU.

(4) Receive control parity check

Reception is controlled in accordance with the contents set in the asynchronous serial interface mode register (ASIM). In addition, error check operations, such as parity error check are also performed during reception. If an error has been detected, a value corresponding to the error is set in the asynchronous serial interface status register (ASIS).

(5) Transfer control parity append

Transfer is controlled by appending a start bit, parity bit, and stop bit to the data written to the TXS register, in accordance with the contents set in the ASIM register.

(6) Selector

This selects the clock source for baud rate.

11.2 Asynchronous Serial Interface Control Registers

(1) Asynchronous serial interface mode register (ASIM)

This 8-bit register specifies the asynchronous serial interface operations.

Data can be read from or written to this register by an 8-bit manipulation or bit manipulation instruction. Fig. 11-2 shows the format for this register.

When the $\overline{\text{RESET}}$ signal is input, the contents of this register are initialized to 80H.

Fig. 11-2 Asynchronous Serial Interface Mode Register (ASIM) Format

	7	⑥	5	4	3	2	1	0
ASIM	1	RXE	PS1	PS0	CL	SL	0	SCK

SCK	Specifies serial clock
0	8-bit timer/counter output
1	Baud rate generator output

SL	Specifies stop bit for transmit data
0	1 bit
1	2 bits

CL	Specifies character length for transmit/receive data
0	7 bits
1	8 bits

PS1	PS0	Specifies parity bit for transmit/receive data
0	0	No parity
0	1	Transmit : 0 parity append Receive : No parity error
1	0	Odd parity
1	1	Even parity

RXE	Controls reception
0	Disables reception
1	Enables reception

Caution 1: The asynchronous serial interface mode register (ASIM) must not be modified during transmission operation. If the ASIM register is modified during transmission operation, no transmission can be performed after that (conditions can be returned to normal by $\overline{\text{RESET}}$ input).
Whether or not transmission is being performed can be determined by software using the transmission completion interrupt (INTST) or the interrupt request flag (STIF), which is set by INTST.

2: If the ASIM register is modified during a receive operation, the data being received or the next data to be received may be damaged. Receive should be disabled when changing the mode.

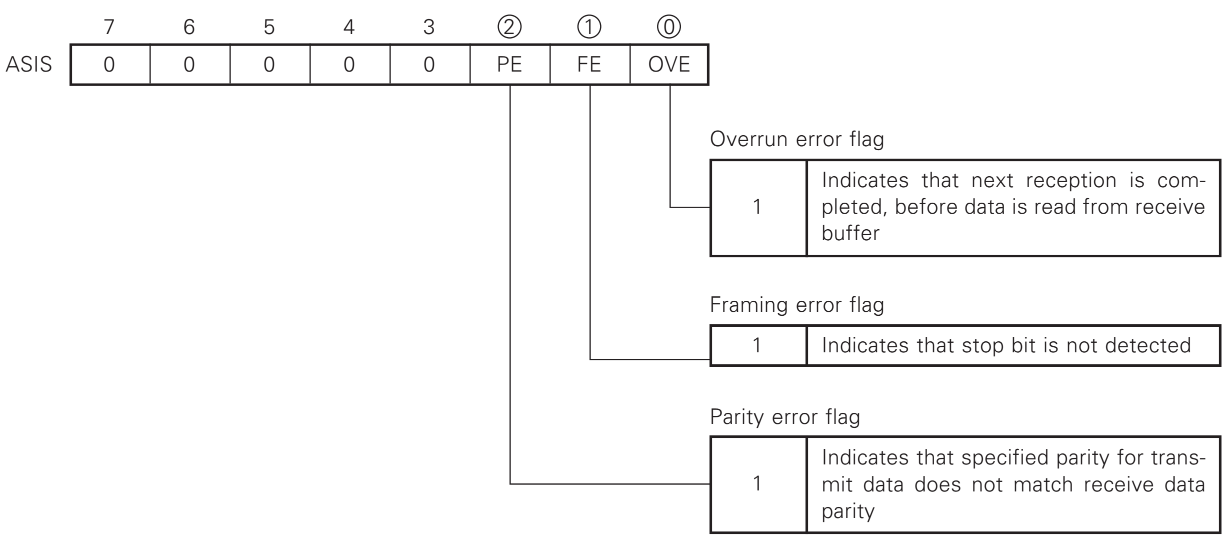
(2) Asynchronous serial interface status register (ASIS)

The ASIS register is a collection of flags that identifies the nature of an error, if an error has occurred during reception. Each flag is set to 1, when a reception error has occurred, and is reset to 0, when data has been read from the receive buffer (RXB). When the new, next data has been received, the overrun error flag (OVE) is set to 1, and the other error flags are reset to 0 (if the next data contains an error, the error flag corresponding to that error is set to 1).

Data can only be read from this register by an 8-bit manipulation or bit manipulation instruction. This data format is shown in Fig. 11-3.

When the $\overline{\text{RESET}}$ signal is input, this register's contents are initialized to 00H.

Fig. 11-3 Asynchronous Serial Interface Status Register (ASIS)



Caution: Be sure to read the receive buffer (RXB) contents, even when a reception error has occurred. Otherwise, an overrun error will occur, when the next data is received, and the error status will persist.

11.3 Asynchronous Serial Interface Operations

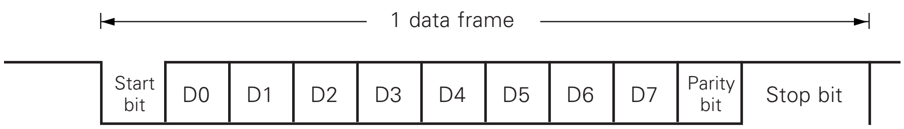
11.3.1 Data format

The asynchronous serial interface transmits/receives serial data in full duplex asynchronous mode.

The transmit/receive data format is shown in Fig. 11-4. One data frame consists of a start bit, character bits, a parity bit, and one or two stop bits.

The number of character bits and stop bits, and whether or not a parity bit is used, are specified by asynchronous serial interface mode register (ASIM).

Fig. 11-4 Asynchronous Serial Interface Format Transmit/Receive Data



- Start bit .... 1 bit
- Character bit ... 7 or 8 bits
- Parity bit ...... Even, odd, 0, or none
- Stop bit .... 1 or 2 bits

The serial transfer rate can be 1.43 bps to 93.75 kbps. It is specified by the asynchronous serial interface mode register and baud rate generator or timer/counter 3.

If an error occurs as a result of receiving serial data, the error can be identified by reading the asynchronous serial interface status register (ASIS) contents.

11.3.2 Parity types and operations

The parity bit is used to detect a bit error in transfer/receive data. Usually, the same parity bit is used at both the transfer and reception sides. When even or odd parity is used, 1-bit (the odd number of) error can be detected. When the 0 parity is used or the parity is not used, no error can be detected.

• Even parity

The parity bit is set to "1", when the transfer data has an odd number of bits, which are "1". The parity bit is reset to "0", when the transfer data has the even number of bits which are "1", so that the number of bits, which are "1" in the transfer data, is even.

When data is received, the number of bits, which are "1" in the receive data, and parity bit is counted, and, if the number of bits which are "1" is odd, a parity error occurs.

• Odd parity

In contrast to the even parity, controls the number of bits in the transfer data and parity bit, which are "1", to be odd.

When data is received, a parity error occurs, if the number of bits in the receive data and parity bit, which are "1", is even.

• 0 parity

When data is transferred, the parity bit is reset to "0", regardless of the transfer data.

During reception, the parity bit is not checked. Therefore, a parity error does not occur, regardless of whether the parity bit is "0" or "1".

• No parity

The parity bit is not appended to the transfer data.

Reception is performed without a parity bit. Because no parity bit is used, no parity error occurs.

11.3.3 Transfer

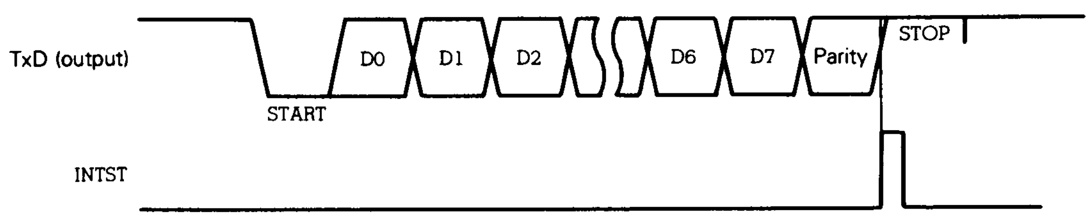
The asynchronous serial interface for μPD78234 is always enabled to transfer. Transfer is started by writing the transfer data to the transfer shift register (TXS). The start bit, parity bit, and stop bit are automatically appended to the data.

When the transfer has been started, the data in the transfer shift register (TXS) is shifted out. When the transfer shift register becomes empty as a result, transfer end interrupt (INTST) is generated.

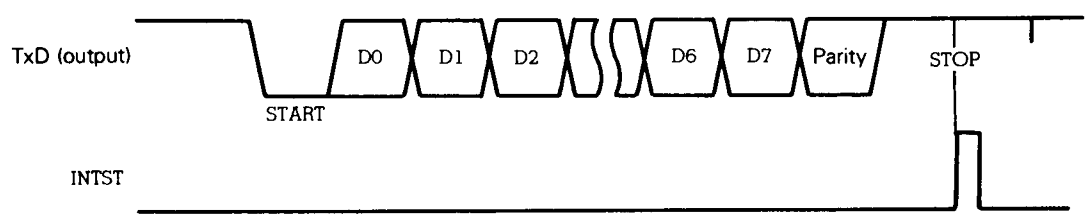
The transfer is stopped, unless the data to be transferred next is written to the transfer shift register.

Fig. 11-5 Asynchronous Serial Interface Transfer End Interrupt Timing

(a) Stop bit length: 1



(b) Stop bit length: 2



Caution 1: The transfer shift register becomes empty, after the $\overline{\text{RESET}}$ signal has been input, but the transfer end interrupt is not generated. Transfer can be started by writing transfer data to the transfer shift register.

2: The asynchronous serial interface mode register (ASIM) must not be modified during transmission operation. If the ASIM register is modified during transmission operation, no transmission can be performed after that (conditions can be returned to normal by $\overline{\text{RESET}}$ input).

Whether or not transmission is being performed can be determined by software using the transmission completion interrupt (INTST) or the interrupt request flag (STIF), which is set by INTST.

11.3.4 Reception

Reception is enabled and the input to the RxD pin is sampled when the RXE bit for the asynchronous serial interface mode register (ASIM) is set to 1.

The RxD pin input is sampled at the serial clock specified by ASIM.

When the RxD pin goes low, the 16-division counter starts counting. When the counter has counted eight times, 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 as a result, it is recognized as a start bit. The 16-division counter will then be initialized, start counting, and sample data. When character data, parity bit, and 1 stop bit* have been detected following the start bit, reception of one frame of data is completed.

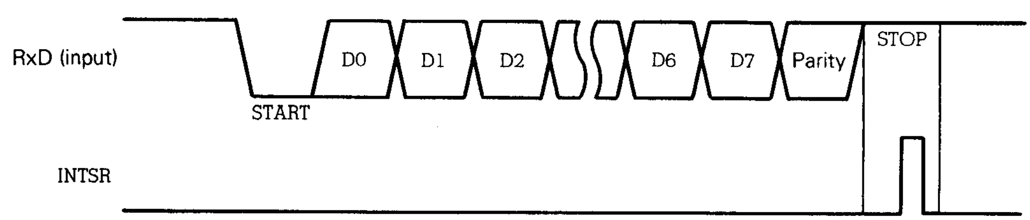
When one frame of data has been received, the receive data in the shift register is transferred to the receive buffer RXB, and reception end interrupt (INTSR) is generated.

If an error occurs, the receive data responsible for the error is transferred to RXB and INTSR is generated.

If the RXE bit is reset to 0 during reception, the reception is immediately stopped. At this time, the RXB and ASIS do not change, do not INTSR and INTSER occur. When the RXE bit is set next time, sampling the start bit is started again.

*: The stop bit operates as 1 bit when data is received, regardless of the specification by the SL bit of the ASIM register.

Fig. 11-6 Asynchronous Serial Interface Reception End Interrupt Timing



Caution 1: Be sure to read the receive buffer (RXB) contents, even when a reception error has occurred. Otherwise, an overrun error will occur, when the next data is received, and the error status will persist.

2: If the ASIM register is modified during a receive operation, the data being received or the next data to be received may be damaged. Receive should be disabled when changing the mode.

11.3.5 Reception error

Three types of errors may occur during reception: parity error, framing error, and overrun error. If any one of these errors has occurred, the corresponding error flag for the asynchronous serial interface register (ASIS) is set and a reception error interrupt (INTSER) is generated. Table 11-1 lists error causes.

By reading the asynchronous serial interface status register (ASIS) contents, using the reception error interrupt routine, which error has occurred can be identified (see **Fig. 11-3** and **11-7**).

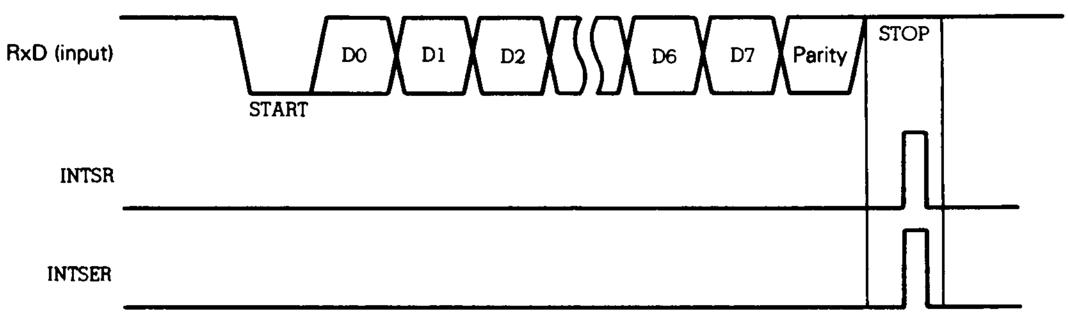
The ASIS contents are reset to 0, when the reception buffer (RXB) contents have been read or when the next data has been received (if the next data contains an error, the corresponding error flag is set).

Table 11-1 Reception Error Causes

Error	Cause
Parity	Parity specified for transfer does not coincide with received data parity
Framing	Stop bit is not detected*
Overrun	Next data has been received before data is read from receive buffer

***:** The stop bit operates as 1 bit when data is received. Therefore, if the stop bit length is specified as 2 bits, and if the second bit of the stop bit is low, an error will not occur. (The second bit of the stop bit is recognized as the start bit of the next data.)

Fig. 11-7 Reception Error Timing



Remarks: With the μPD78234, the break signal cannot be detected by hardware. Since the break signal is a low-level signal of two or more characters, input of the break signal can be judged by detecting consecutive occurrence of two framing errors in which the receive data is 00H through software. Determining when this is caused by two inadvertent framing errors can be performed by checking if the RxD pin level is "0" (possible by setting bit 0 of Port 3 Mode Register (PM3) then reading Port3).

Caution: 1. The contents of ASIS register contents are reset to 0, when the receive buffer (RXB) contents have been read or when the next data has been received. To identify the error, be sure to read ASIS before reading RXB.

When reception is performed by using the macro service, only the occurrence of an error can be learned (by INTSER occurrence or by setting the receive error interrupt request flag (SERIF) to 1). Make sure that this poses no problem.

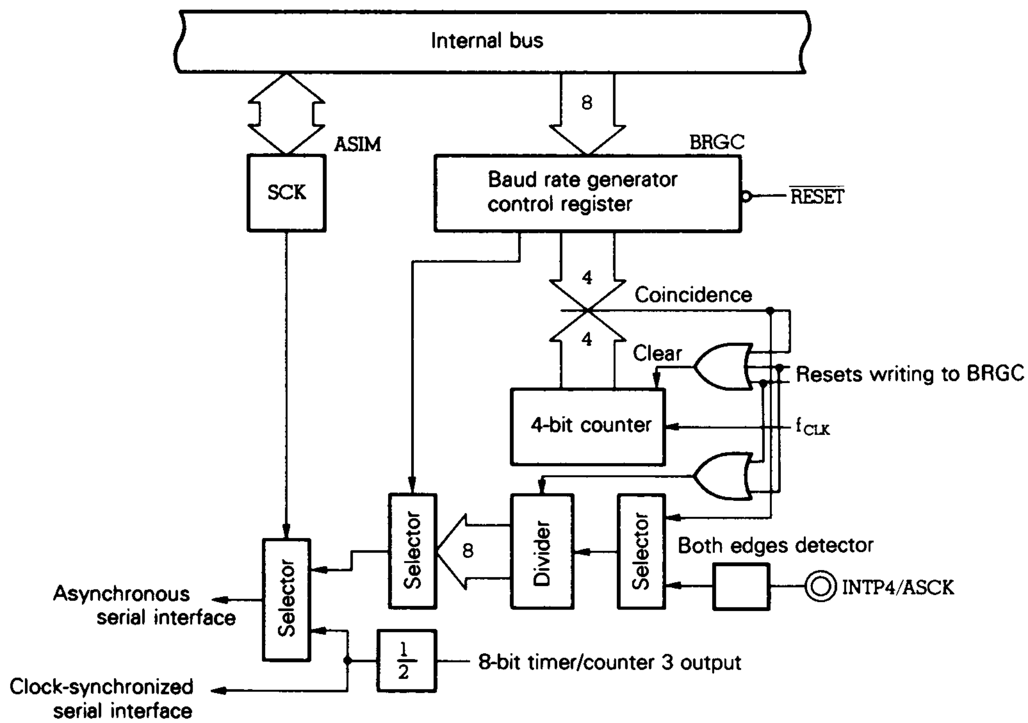
2. Be sure to read the receive buffer (RXB), when a reception error has occurred. Otherwise, an overrun error occurs, when the next data has been received, and the reception error status will persist.

11.4 Baud Rate Generator

11.4.1 Baud rate generator configuration

Fig. 11-8 shows the baud rate generator configuration.

Fig. 11-8 Baud Rate Generator Clock Configuration



(1) 4-bit counter

This counter counts the internal system clock (f_{CLK}) and generates a signal at the frequency selected by the lower 4 bits in the baud rate generator control register (BRGC).

(2) Frequency divider

Divides the signal input from the 4-bit counter or an external baud rate (ASCK) source, and allows the selector at the next stage to select the clock for baud rate.

(3) Both edges detector

Detects both edges of the signal input to the ASCK pin and generates a signal having a frequency two times that of the ASCK input clock. For details on edge detection, refer to **CHAPTER 13 EDGE DETECTION FUNCTION**.

11.4.2 Baud rate generator control register (BRGC)

BRGC is an 8-bit register that sets the clock for baud rate generation under the control of the internal system clock (f_{CLK}).

This register is write-only and can be manipulated by an 8-bit manipulation instruction. The format for this register is shown in Fig. 11-9.

When the $\overline{\text{RESET}}$ signal has been input, this register's contents are initialized 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 written during transfer operation, therefore, baud rate generation is disturbed, making it impossible to perform correct communication. For this reason, do not write the BRGC register during transfer operation.

Fig. 11-9 Baud Rate Generator Control Register (BRGC) Format

	7	6	5	4	3	2	1	0
BRGC	CE	TPS2	TPS1	TPS0	MDL3	MDL2	MDL1	MDL0

MDL3	MDL2	MDL1	MDL0	k	Baud rate generator input clock
0	0	0	0	3	$F_{CLK}/4$
0	0	0	1	1	$F_{CLK}/2$
0	0	1	0	2	$F_{CLK}/3$
0	0	1	1	3	$F_{CLK}/4$
0	1	0	0	4	$F_{CLK}/5$
0	1	0	1	5	$F_{CLK}/6$
0	1	1	0	6	$F_{CLK}/7$
0	1	1	1	7	$F_{CLK}/8$
1	0	0	0	8	$F_{CLK}/9$
1	0	0	1	9	$F_{CLK}/10$
1	0	1	0	10	$F_{CLK}/11$
1	0	1	1	11	$F_{CLK}/12$
1	1	0	0	12	$F_{CLK}/13$
1	1	0	1	13	$F_{CLK}/14$
1	1	1	0	14	$F_{CLK}/15$
1	1	1	1	External clock input(ASCK)	

TPS2	TPS1	TPS0	Frequency divider circuit tap 1/n
0	0	0	1/2
0	0	1	1/4
0	1	0	1/8
0	1	1	1/16
1	0	0	1/32
1	0	1	1/64
1	1	0	1/128
1	1	1	1/256

CE	4-bit counter and frequency divider circuit operation
0	Stop
1	Counting operation

11.4.3 Baud rate generator operations

The baud rate generator starts operating, when the CE bit for the baud rate generator control register (BRGC) is set to 1. The baud rate clock to be generated is either the internal system clock (f_{CLK}) divided, or the clock input from the external baud rate input (ASCK) pin and divided.

To stop the baud rate generator, reset the CE bit to 0.

If the CE bit is set to 1 again, when the bit has already been set, the 4-bit counter for baud rate generation and frequency divider are cleared and the clock for baud rate is generated again.

Caution: If the BRGC register is written during transfer operation, baud rate generation is disturbed, making it impossible to perform correct communication. For this reason, do not write the BRGC register during transfer operation.

(1) Baud rate clock generation from internal system clock (f_{CLK})

The internal system clock (f_{CLK}) is divided by the 4-bit counter.

The resultant signal is divided by the frequency divider, to obtain the baud rate clock.

The baud rate generated from the internal system clock (f_{CLK}) is determined by the following expression:

$$\text{(Baud rate)} = \frac{f_{CLK}}{k+1} \times \frac{1}{n} \times \frac{1}{16}$$

where,

f_{CLK} : internal system clock frequency

k : set value of MDL3-MDL0 bits for BRGC register
(k = 1-14; see **Fig. 11-9**)

1/n : frequency divider tap

16 : serial data sampling rate

(2) Baud rate clock generation from ASCK input

Both ASCK pin input signal edges are detected and the clock for a frequency two times that for the ASCK pin input clock is created. This clock is divided by 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 ASCK pin input, is determined by the following expression:

$$\text{(Baud rate)} = f_{ASCK} \times \frac{2}{n} \times \frac{1}{16}$$

f_{ASCK}: frequency of ASCK input clock

Note that the ASCK input must be less than $f_{CLK}/24$ (250 kHz: f_{CLK} = 6 MHz).

11.5 Baud Rate Setting

The baud rate can be set by three methods shown in Table 11-2, indicating the baud rate range to be generated, the expression used to calculate the baud rate, and the selection method.

Table 11-2 Baud Rate Setting

<table>
<tr><th colspan="2">Baud rate clock source</th><th colspan="2">Selection method</th><th>Calculation expression</th><th>Baud rate range</th></tr>
<tr><td rowspan="2">Baud rate generator</td><td>Internal system clock</td><td rowspan="2">SCK for ASIM register = 1
CE for BRGC register = 1</td><td>MDL0-3 for BRGC register = 0-EH</td><td>$\frac{f_{CLK}}{k+1} \times \frac{1}{n} \times \frac{1}{16}$</td><td>$\frac{f_{CLK}}{61440} - \frac{f_{CLK}}{64}$</td></tr>
<tr><td>ASCK input</td><td>MDL0-3 for BRGC register = FH</td><td>$f_{ASCK} \times \frac{2}{n} \times \frac{1}{16}$</td><td>$\frac{f_{ASCK}}{2048} - \frac{f_{ASCK}}{16}$*</td></tr>
<tr><td colspan="2">8-bit timer/ counter 3</td><td colspan="2">SCK for ASIM register = 0</td><td>$\frac{f_{CLK}}{2^{j+3}} \times \frac{1}{m+1} \times \frac{1}{2} \times \frac{1}{16}$</td><td>$\frac{f_{CLK}}{4194304} - \frac{f_{CLK}}{256}$</td></tr>
</table>

f_{CLK} : internal system clock frequency
k : set value for MDL3-MDL0 bits in BRGC register (k = 1-14: refer to **Fig. 11-9**)
1/n : frequency divider tap (n = 2, 4, 8, 16, 32, 64, 128, 256) f_{ASCK}: frequency of ASCK input clock (0 - f_{CLK}/24)
1/16 : sampling rate of serial data
j : set value for PRS3-PRS0 bit in prescaler mode register (j = 0-6)

PRS3-PRS0	0H	1H	2H	3H	4H	5H	6H	7H
j	0		1	2	3	4	5	6

m : set value (m = 0-255) for 8-bit compare register (CR30)

***:** 0-f_{CLK}/384 including f_{ASCK} input range

11.5.1 Setting examples, when baud rate generator is used

This section shows examples of setting the BRGC register, when the baud rate generator is used.

To use the baud rate generator, set the SCK bit for the asynchronous serial interface mode register (ASIM) to 1.

Table 11-3 Setting BRGC Register Examples When Baud Rate Generator Is Used

Oscillation frequency (fxx) or external clock input (fx)	12 MHz		11.0592 MHz		10.0 MHz		9.8304 MHz		7.3728 MHz	
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

11.5.2 Setting examples, when 8-bit timer/counter 3 is used

Table 11-4 shows setting baud rate examples, when the 8-bit timer/counter 3 is used. When using the 8-bit timer/counter 3, reset the SCK bit for the asynchronous serial interface mode register (ASIM) to 0.

For how to use the 8-bit timer/counter 3, refer to **7.4 8-Bit Timer/Counter 3**.

Table 11-4 Setting Baud Rate Example, 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		
Internal system clock (f_{CLK}) MHz	6 MHz			5.5296 MHz			5.0 MHz			4.9152 MHz			3.6864 MHz		
Baud rate [bps]	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 (%)
75	$f_{CLK}/16$	155	0.16	$f_{CLK}/16$	143	0.00	$f_{CLK}/16$	129	0.16	$f_{CLK}/16$	127	0.00	$f_{CLK}/8$	191	0.00
110	$f_{CLK}/8$	212	0.03	$f_{CLK}/8$	195	0.18	$f_{CLK}/8$	177	0.25	$f_{CLK}/8$	174	0.26	$f_{CLK}/8$	130	0.07
150	$f_{CLK}/8$	155	0.16	$f_{CLK}/8$	143	0.00	$f_{CLK}/8$	129	0.16	$f_{CLK}/8$	127	0.00	$f_{CLK}/8$	95	0.00
300	$f_{CLK}/8$	77	0.16	$f_{CLK}/8$	71	0.00	$f_{CLK}/8$	64	0.16	$f_{CLK}/8$	63	0.00	$f_{CLK}/8$	47	0.00
600	$f_{CLK}/8$	38	0.16	$f_{CLK}/8$	35	0.00	$f_{CLK}/8$	32	1.36	$f_{CLK}/8$	31	0.00	$f_{CLK}/8$	23	0.00
1200	$f_{CLK}/8$	19	2.34	$f_{CLK}/8$	17	0.00	$f_{CLK}/8$	15	1.73	$f_{CLK}/8$	15	0.00	$f_{CLK}/8$	11	0.00
2400	$f_{CLK}/8$	9	2.34	$f_{CLK}/8$	8	0.00	$f_{CLK}/8$	7	1.73	$f_{CLK}/8$	7	0.00	$f_{CLK}/8$	5	0.00
4800	$f_{CLK}/8$	4	2.34	–	–	–	$f_{CLK}/8$	3	1.73	$f_{CLK}/8$	3	0.00	$f_{CLK}/8$	2	0.00
9600	–	–	–	–	–	–	$f_{CLK}/8$	1	1.73	$f_{CLK}/8$	1	0.00	$f_{CLK}/8$	–	–
19200	–	–	–	–	–	–	$f_{CLK}/8$	0	1.73	$f_{CLK}/8$	0	0.00	$f_{CLK}/8$	–	–

11.5.3 Setting examples, when external baud rate input (ASCK) is used

Table 11-5 shows setting examples when the external baud rate input (ASCK) is used. To use the ASCK input, set the SCK bit for the asynchronous serial interface mode register (ASIM) to 1.

Table 11-5 Setting Examples When 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

11.6 Notes

(1) The asynchronous serial interface mode register (ASIM) must not be modified during transmission operation. If the ASIM register is modified during transmission operation, no transmission can be performed after that (conditions can be returned to normal by $\overline{\text{RESET}}$ input).
Whether or not transmission is being performed can be determined by software using the transmission completion interrupt (INTST) or the interrupt request flag (STIF), which is set by INTST.

(2) If the ASIM register is modified during a receive operation, the data being received or the next data to be received may be damaged. Receive should be disabled when changing the mode.

(3) The transfer shift register becomes empty, after the $\overline{\text{RESET}}$ signal has been input, but the transfer end interrupt does not occur. Transfer can be started by writing transfer data to the transfer shift register.

(4) Be sure to read the receive buffer (RXB), even when a reception error has occurred. Otherwise, an overrun error will occur, when the next data is received, and the reception error state will persist.

(5) The ASIS register contents are reset to 0, when the receive buffer (RXB) has been read or when the next data has been received. To identify the nature of an error, be sure to read ASIS before reading RXB.
When reception is performed by using the macro service, only an error occurrence can be learned (by INTSER generation or setting the reception error interrupt request flag (SERIF) to 1). Make sure that this poses no problem.

(6) Do not write the BRGC register during transfer operation. If a write instruction has been executed, the 4-bit counter and frequency divider are reset, and the baud rate clock generated is disturbed, making it impossible to perform correct communication.

CHAPTER 12 CLOCK-SYNCHRONIZED SERIAL INTERFACE

12.1 Function

The clock-synchronized serial interface for μPD78234 is configured as shown in Fig. 12-1.
This serial interface can operate in the following two modes:

(1) 3-line serial I/O mode (MSB first)

In this mode, 8-bit data is transferred by using three lines: serial clock ($\overline{SCK}$), and two serial bus lines (SO and SI). This mode is suitable for connecting peripheral I/Os and display controllers having a conventional clock-synchronized serial interface.

(2) Serial bus interface (SBI) mode (MSB first)

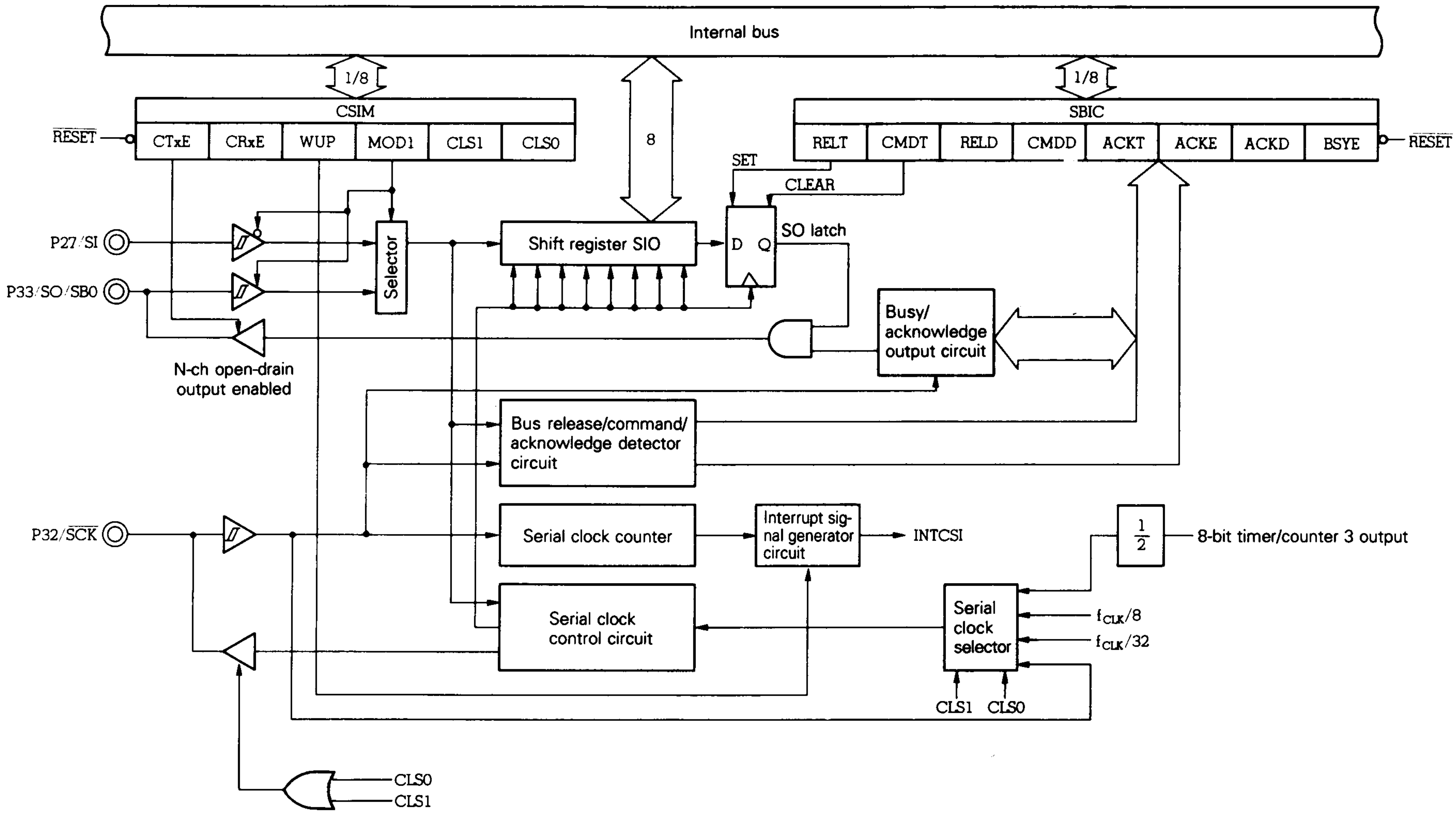
In this mode, two lines, serial clock ($\overline{SCK}$) and serial data bus (SB0), are used to communicate data with several devices. This conforms to NEC's serial bus format.

In the SBI mode, "addresses" which select devices with which the microcomputer is to communicate, "command" which specifies the operations of the selected devices, and actual "data" can be output onto the serial data bus. Therefore, a handshake line, which is required by the conventional clock-synchronized serial interface to connect several devices is not necessary, so that the I/O ports can be effectively used.

12.2 Configuration

Fig. 12-1 shows the clock-synchronized serial interface configuration.

Fig. 12-1 Clock-Synchronized Serial Interface Configuration



(1) Shift register (SIO)

The shift register (SIO) converts 8-bit serial data into 8-bit parallel data, and vice versa. This register is used for both transfer and reception.

Data is shifted into (during reception) or out of (during transfer) the MSB side.

Actual transfer/reception is controlled by writing/reading SIO.

This register can be read or written by an 8-bit manipulation instruction. When the $\overline{\text{RESET}}$ signal has been input, this register's contents become undefined.

(2) SO latch

The SO latch retains the output level for the SO/SB0 pin. This latch can be controlled directly by software in the serial bus interface mode (SBI).

(3) Serial clock selector

Selects the serial clock to be used.

(4) Serial clock counter

Counts the serial clock output or input during transfer/ reception, to check whether 8-bit data has been transferred/ received.

(5) Interrupt signal generator

Controls whether an interrupt request is issued, when the serial clock counter has counted eight serial clocks. In the three-line serial I/O mode, the interrupt request is generated, when the serial clock has counted eight serial clocks. In the SBI mode, the interrupt request is generated, when a specified condition is satisfied.

(6) Serial clock control circuit

Controls serial clock supply to the shift register. When the internal clock is used, the clock to be output to the $\overline{\text{SCK}}$ pin is controlled.

(7) Busy/acknowledge output circuit, bus release/command/acknowledge detector circuit

Outputs and detects various control signals in the SBI mode. Does not operate in the three-line serial mode.

12.3 Control Registers

12.3.1 Clock-synchronized serial interface mode register (CSIM)

The CSIM register is an 8-bit register that specifies the operation mode, serial clock and wake-up functions for the serial interface.

Data can be read from or written to this register by an 8-bit manipulation or bit manipulation instruction. Fig. 12-2 shows the format for this register.

When the $\overline{RESET}$ signal is input, the contents of this register are initialized to 00H.

Fig. 12-2 Clock-Synchronized Serial Interface Mode Register (CSIM) Format

	⑦	⑥	⑤	4	3	2	1	0
CSIM	CTXE	CRXE	WUP	0	MOD1	0	CLS1	CLS0

CLS1	CLS0	Selects serial clock		$\overline{SCK}$ pin	Master/ slave selection in SBI mode
0	0	External clock		Input	Slave
0	1	Internal clock	Timer/ counter 3 output /2	Output	Master
1	0		f_{CLK} /32 *		
1	1		f_{CLK} /8 *		

***:** f_{CLK}: internal system clock

MOD1	WUP	Operation mode	Controls wakeup function
0	0	3-line serial I/O mode	Generates interrupt request at each serial transfer
1	0	SBI mode	
1	1		Generates interrupt request only when address is received

CRXE	Reception operation
0	Disabled
1	Enabled

CTXE	Transmission operation
0	Disabled
1	Enabled

Caution: Do not set CTXE to 1 and clear CRXE to 0, or vice versa, with a single instruction; otherwise, the serial clock counter will malfunction and communication immediately after transfer ends less than 8 bits. Therefore, use two instructions as follows:

Example: To clear CTXE to 0 and set CRXE to 1:
CLR1 CTXE
SET1 CRXE

12.3.2 Serial bus interface control register (SBIC)

The SBIC register is an 8-bit register, consisting of bits that control the serial bus status and bits that indicate the status for various data inputs from the serial bus. This register can be used in the SBI mode only and must not be manipulated in the three-line serial I/O mode.

Use an 8-bit manipulation and bit manipulation instructions to both manipulate this register. Some bits of this register can be read and written, some can only be read, and some can only be written, as shown in Table 12-1. When the write-only bit is read, it is always "0". The SBIC register format is shown in Fig. 12-3.

The contents of this register are initialized to 00H, when the $\overline{\text{RESET}}$ signal has been input.

The ACKD, CMDD, and RELD flags are cleared, when transfer/ reception is disabled (CTxE = CHxE = 0).

Table 12-1 Reading/Writing SBIC Register

	⑦	⑥	⑤	④	③	②	①	⓪
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. 12-3 Serial Bus Interface Control Register (SBIC) Format

	⑦	⑥	⑤	④	③	②	①	⓪
SBIC	BSYE	ACKD	ACKE	ACKT	CMDD	RELD	CMDT	RELT

RELT	Controls trigger output for bus release signal (REL)
0	Does not output
1	Outputs

CMDT	Controls trigger output for command signal (CMD)
0	Does not output
1	Outputs

RELD	Detects bus release signal (REL)
0	Does not output
1	Detects

CMDD	Detects command signal (CMD)
0	Does not detect
1	Detects

ACKT	Controls trigger output for acknowledge signal ($\overline{ACK}$)
0	Does not output
1	Outputs

ACKE	Enables automatic output for acknowledge signal ($\overline{ACK}$)
0	Disables
1	Enables

ACKD	Detects acknowledge signal ($\overline{ACK}$)
0	Does not detect
1	Detects

BSYE	Enables automatic output for synchronous busy signal ($\overline{BUSY}$)
0	Disables
1	Enables

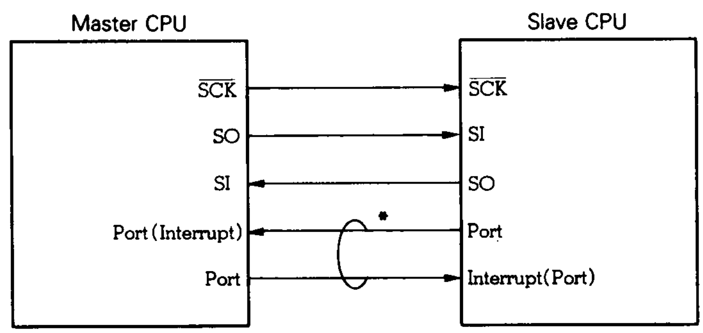
12.4 3-line Serial I/O Mode

In this mode, the microcomputer can communicate data with devices having a conventional clock-synchronized serial interface.

Basically, communication is established by three lines: serial clock ($\overline{SCK}$), serial data output (SO), and serial data input (SI) lines. To connect the microcomputer with several devices, however, a handshake line is necessary.

Fig. 12-4 3-Line Serial I/O System Configuration Example

3-line serial I/O ↔ 3-line serial I/O



***:** Handshake line

12.4.1 Basic operation timing

In the three-line serial I/O mode, data is transferred/received in 8 bits units. The data is transferred/received on a bit-by-bit basis, in synchronization with the serial clock, starting from the MSB.

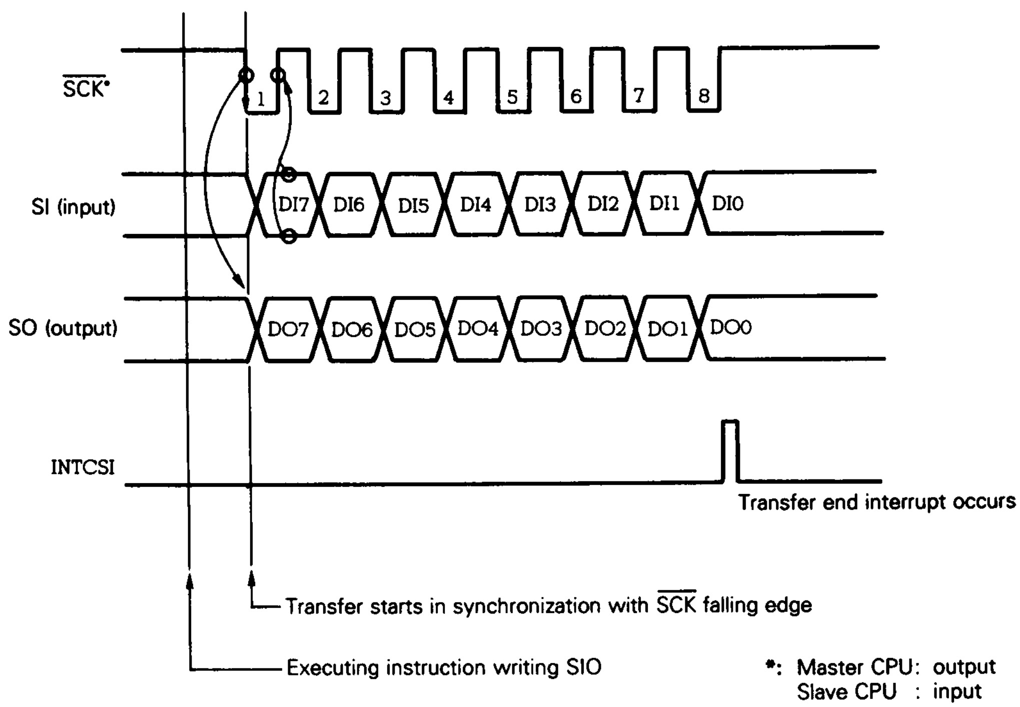
The transfer data is output in synchronization with the $\overline{SCK}$ falling edge. The receive data is sampled at the $\overline{SCK}$ rising edge.

Interrupt request INTCSI is generated at the eighth $\overline{SCK}$ rising edge.

When the internal clock is used as $\overline{SCK}$, $\overline{SCK}$ output is stopped at the eighth $\overline{SCK}$ rising edge, and $\overline{SCK}$ is kept high, until transfer or reception of the next data is started.

Fig. 12-5 shows the timing in the three-line serial I/O mode.

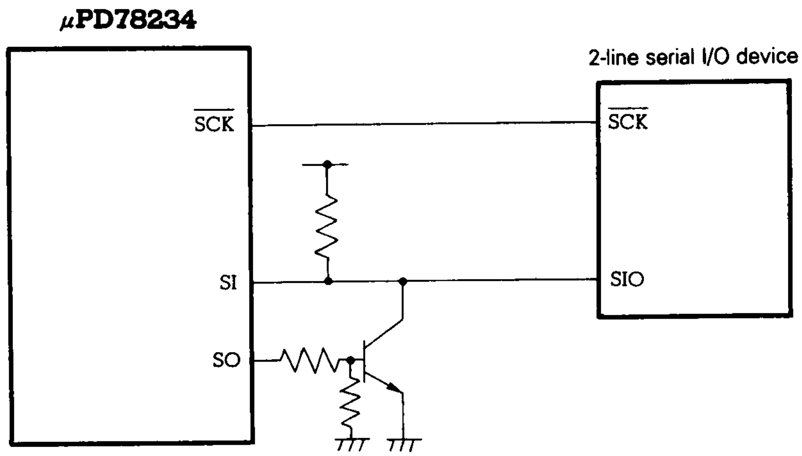
Fig. 12-5 Timing in Three-Line Serial I/O Mode



In the three-line serial I/O Mode, the SO pin functions as a CMOS push-pull output pin.

Remarks: To connect a two-line serial I/O, connect a buffer to the SO pin, as shown in Fig. 12-6. Note that, in this case, the output level is inverted by the buffer. Therefore, write data, opposite to the data to be output, to SIO.
In addition, the internal pull-up resistor must not be connected to the P33/SO pin.

Fig. 12-6 Connecting Two-Line Serial I/O



To execute transmission and reception by means of time division, and when the μPD78234 can control the output level of the SIO pin of a device having a two-line serial I/O, the SI and SO pins can be directly connected. When the μPD78234 transmitts data in this status, be sure to disable any other devices from transmitting.

12.4.2 Operations when only transfer is enabled

Transfer is executed when the CTXE bit for the clock-synchronized serial interface mode register (CSIM) is set to 1. When data is written to the shift register (SIO), with the CTXE bit set to 1, transfer is started.

When the CTXE bit is reset to 0, the SO pin enters the output high-impedance state.

(1) When internal clock is used as serial clock

When transfer is started, the serial clock is output from the $\overline{SCK}$ pin. Data is sequentially output to the SO pin from SIO in synchronization with the serial clock falling edge. The SI pin signal is shifted into SIO, in synchronization with the serial clock rising edge.

It must be noted that it takes one $\overline{SCK}$ clock (max.) to the first falling edge of $\overline{SCK}$ from transmission start up.

When transmission is disabled while in progress (by resetting the CTXE bit to 0), output of the $\overline{SCK}$ clock is stopped at the rising edge of the next $\overline{SCK}$, and transmission is aborted. At this time, the interrupt request (INTCSI) is not generated. The SO pin goes into a high-impedance state.

(2) When external clock is used as serial clock

When transfer is started, data is sequentially output to the SO pin from SIO in synchronization with the falling edge for the serial clock input to the $\overline{SCK}$ pin, after the transfer has been started. The SI pin signal is shifted into SIO in synchronization with the rising edge for the $\overline{SCK}$ pin input. If transfer is not started, shifting is not executed, even when the serial clock is input to the $\overline{SCK}$ pin, and the output level of the SO pin does not change.

When transmission is disabled while in progress (by resetting the CTXE bit to 0), transmission is aborted, and the following $\overline{SCK}$ input is ignored. At this time, the interrupt request (INTCSI) is not generated. The SO pin goes into a high-impedance state.

12.4.3 Operation when only reception is enabled

Reception is performed when the CSIM register CRXE bit is set to 1. When the CRXE bit is changed from 0 to 1, or when data is read from SIO, reception is started.

(1) When internal clock is used as serial clock

When reception is started, the serial clock is output from the $\overline{SCK}$ pin and the SI pin data is sequentially input to SIO, in synchronization with the serial clock rising edge.

It must be noted that it takes one $\overline{SCK}$ clock (max.) to the first falling edge of $\overline{SCK}$ from transmission start up.

Note that one $\overline{SCK}$ clock (max.) is required from transmission start up to the first $\overline{SCK}$ falling edge.

When reception is disabled (by resetting the CRXE bit to 0) while in progress, output of the $\overline{SCK}$ clock is stopped at the rising edge of the next $\overline{SCK}$, and reception is stopped. At this time, the interrupt request (INTCSI) is not generated. The contents of the SIO register become undefined.

(2) When external clock is used as serial clock

When reception is started, the SI pin data is sequentially input to SIO in synchronization with the rising edge for the serial clock input to the $\overline{SCK}$ pin, after reception has been started. Even if the serial clock is input to the $\overline{SCK}$ pin, when reception is not started, the shifting does not take place.

When reception is disabled (by resetting the CRXE bit to 0) while in progress, reception is stopped, and the following $\overline{SCK}$ input is ignored. At this time, the interrupt request (INTCSI) is not generated.

12.4.4 Operations when both transfer and reception are enabled

Both transfer and reception can be carried out at the same time when both the CTXE and CRXE bits for the CSIM register are set to 1. The transfer and reception can be started, when the CRXE bit is changed from 0 to 1, or when data is written to SIO.

When transfer and reception are started for the first time, the CRXE bit always changes from 0 to 1, which immediately starts transfer and reception and increases the possibility that undefined data is output. Therefore, write the first transfer data to SIO, while both transfer and reception are disabled (while both the CTXE and CRXE bits are reset to 0). Then, enable transfer and reception.

When transmission/recepton is disabled (CTXE = CRXE = 0), the SO pin goes into an output high-impedance state.

(1) When internal clock is used as serial clock

When transfer and reception are started, the serial clock is output from the $\overline{\text{SCK}}$ pin. Data is sequentially output from SIO to the SO pin in synchronization with the serial clock falling edge. The SI pin data is sequentially shifted into SIO, in synchronization with the serial clock rising edge.

It must be noted that it takes one $\overline{\text{SCK}}$ clock (max.) to the first falling edge of $\overline{\text{SCK}}$ from transmission start up.

If either transmission or reception is disabled while transmission and reception are in progress, only the disabled operation is stopped. If only transmission is disabled, the SO pin goes into a high-impedance state. If only reception is disabled, the contents of the SIO register become undefined.

If both transmission and reception are disabled, output of the $\overline{\text{SCK}}$ clock is stopped at the rising edge of the next $\overline{\text{SCK}}$, and both transmission and reception are stopped. In this case, the contents of SIO become undefined, and the interrupt request (INTCSI) is not generated. The SO pin goes into a high-impedance state.

(2) When external clock is used as serial clock

When transfer and reception are started, data is sequentially output to the SO pin from SIO, in synchronization with the falling edge for the serial clock input to the $\overline{\text{SCK}}$ pin, after transfer and reception have been started, and the SI pin data is sequentially shifted into SIO in synchronization with the serial clock rising edge. The SIO contents are not shifted, nor is the SO pin output level changed, even when the serial clock is input to the $\overline{\text{SCK}}$ pin, when transfer and reception are not started.

If either transmission or reception is disabled while transmission and reception are in progress, only the disabled operation is stopped. If only transmission is disabled, the SO pin goes into a high-impedance state. If only reception is disabled, the contents of the SIO register become undefined.

If both transmission and reception are disabled, both transmission and reception are stopped, and the following $\overline{\text{SCK}}$ input is ignored. In this case, the contents of SIO become undefined, and the interrupt request (INTCSI) is not generated. The SO pin goes into a high-impedance state.

12.4.5 Corrective action, when serial clock is asynchronous with shifting

When the external clock is selected as the serial clock, the number of serial clocks and shift operation may not match, due to noise. In this case, disable both transfer and reception (by resetting both the CTXE and CRXE bits to 0) to initialize the serial clock counter, so that synchronization between the shift operation and serial clock can be recovered, by causing the serial clock to be input first, when transfer or reception is enabled next time. This is used as the first clock.

12.5 SBI Mode

SBI (serial bus interface) is a high-speed serial interface method conforming to the NEC serial bus format.

SBI is a single-master high-speed serial bus. Its format, by which functions to configure a bus, are added to the clock-synchronized serial I/O, so that communication with more than one device can be established with two signal lines. Therefore, it can reduce the number of ports and wiring lines on a printed circuit board, when a serial bus is configured by two or more microcomputers and peripheral ICs.

For further information on the SBI function, refer to the **Serial Bus Interface (SBI) User's Manual (IEM-5040)**.

12.5.1 Features of SBI

Since the existing serial I/O system has only data transfer functions, it calls for many ports and wiring lines, when more than one device is connected to constitute a serial bus, to identify the chip select signal, command, and data, and to detect the busy status. If these controls are to be performed by software, the software overhead will increase.

The SBI can form a serial bus with two signal lines: serial clock $\overline{\text{SCK}}$ and serial data bus SB0, so that the number of ports for microcomputers and wiring lines on a printed circuit board can be reduced.

The SBI functions are as follow:

(1) Address/command/data identification

The SBI identifies serial data as being an address, command, or data.

(2) Chip select function by address

The master can select a slave chip by transferring an address.

(3) Wakeup function

The slave can easily judge whether it has received an address or not (judgement for chip selection), by using the wakeup function (which can be set or canceled by software).

When the wakeup function is set, a serial reception interrupt (INTCSI) occurs, only when an address has been received.

Therefore, slave CPUs, other than the one selected, can operate independently from the serial communication, even when communication is established among several devices.

(4) Acknowledge signal ($\overline{\text{ACK}}$) control function

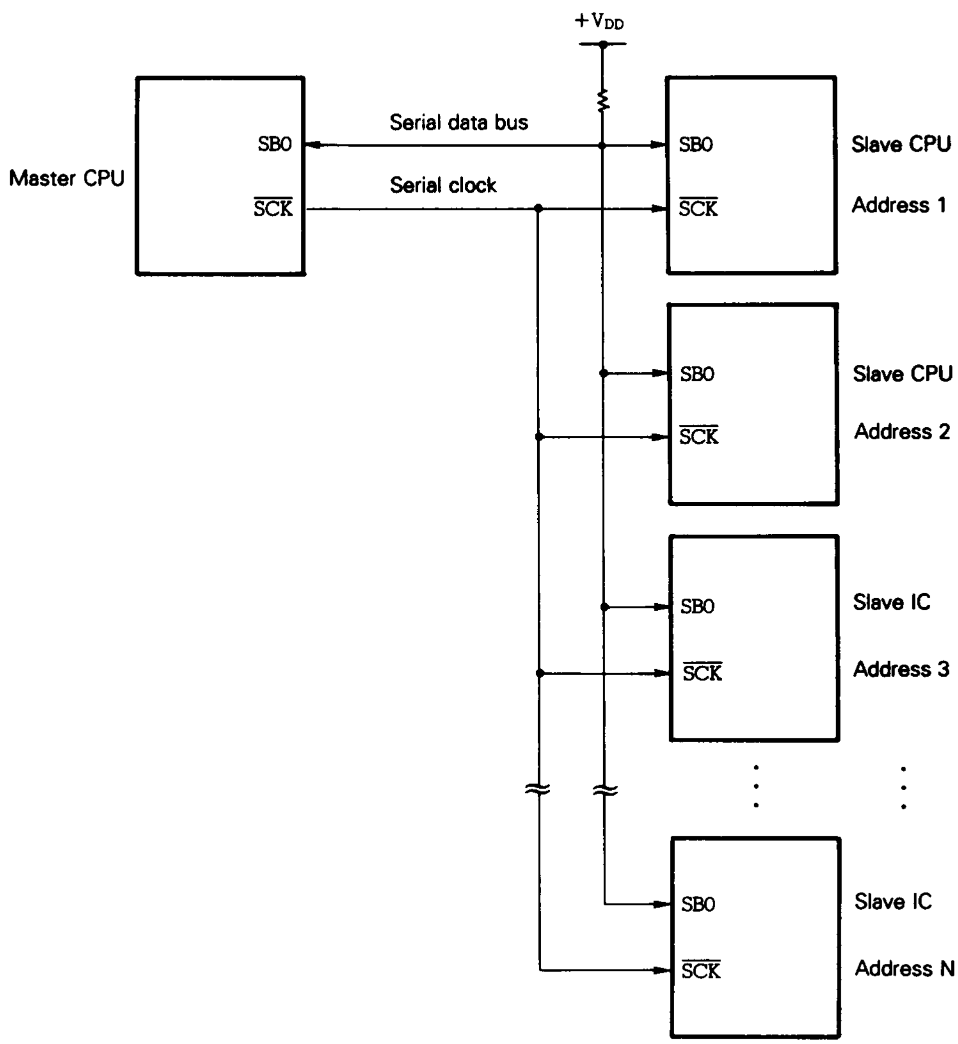
Controls the acknowledge signal to confirm serial data reception.

(5) Busy signal ($\overline{\text{BUSY}}$) control function

Controls the busy signal that informs the slave busy status.

Fig. 12-7 shows a configuration example for a serial bus, that consists of CPUs, having a serial interface conforming to SBI, and peripheral ICs.
The serial data bus pin SB0 for SBI is open-drain output. Therefore, the serial data bus line is wired-ORed. In addition, the serial data bus line must be connected to a pull-up resistor.

Fig. 12-7 Serial Bus Configuration with SBI Example



Caution: To exchange the master with a slave, the master and slave switch over, between the input and output for the serial clock line ($\overline{SCK}$) asynchronously with each other. Therefore, a pull-up resistor must also be connected to the serial clock line ($\overline{SCK}$).

12.5.2 Serial interface configuration

Fig. 12-9 shows the μPD78234 circuit configuration

The serial clock pin ($\overline{SCK}$) and serial data bus pin SB0 are configured as follows:

(1) $\overline{SCK}$ .... Inputs/outputs serial clock
- Master ... CMOS, push-pull output
- Slave ...... Schmitt input

(2) SB0 .... Inputs/outputs serial data.
N-ch open-drain output for both master and slave.
Schmitt input

Since the serial data bus line output is N-ch open-drain, an external pull-up resistor is necessary.

Fig. 12-8 Pin Configuration

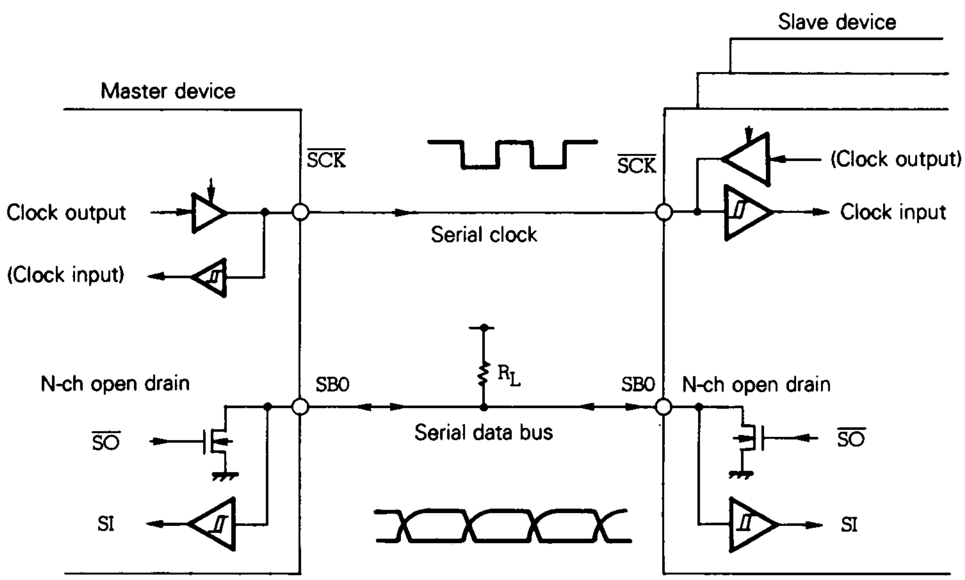
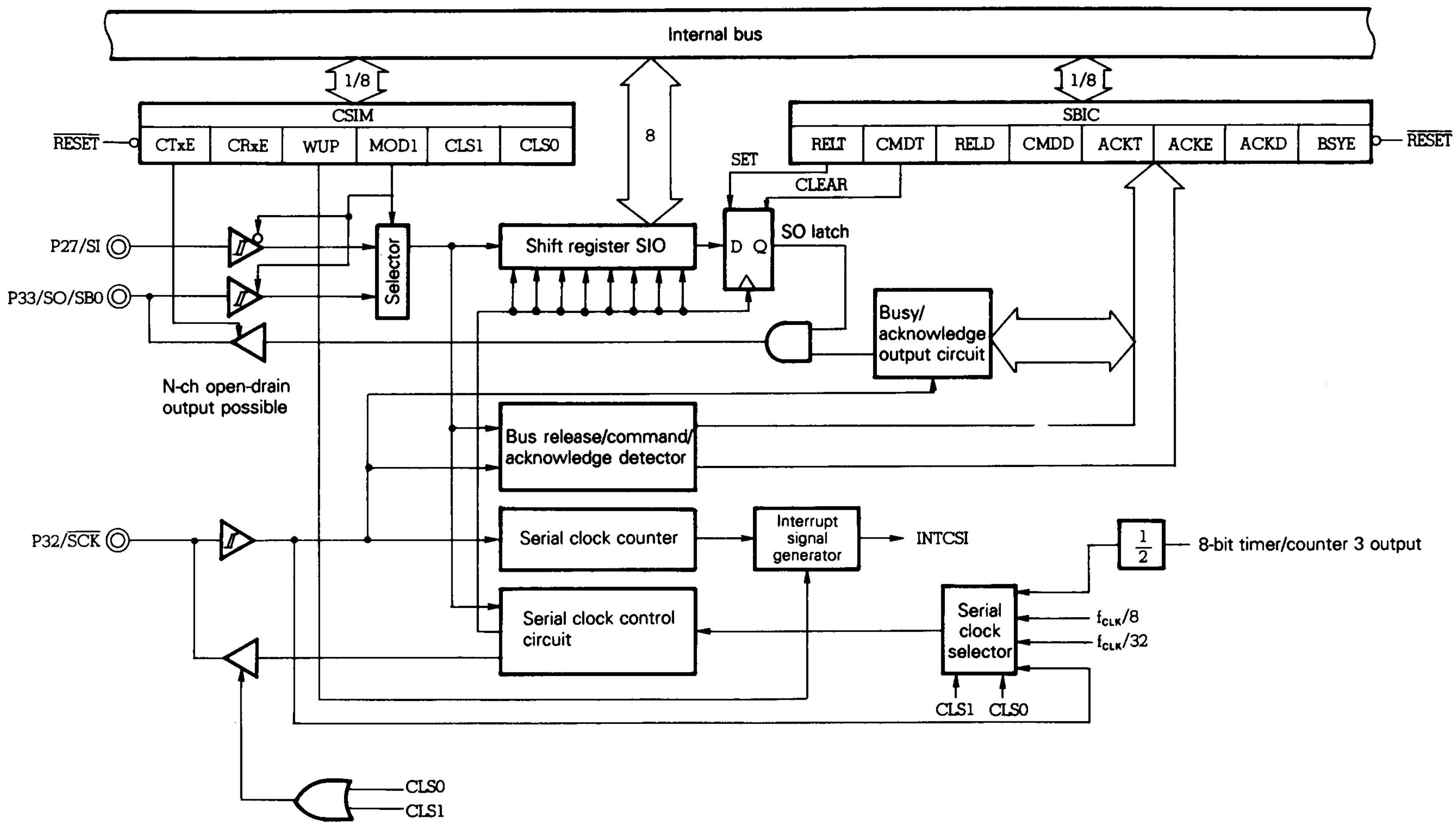


Fig. 12-9 Clock-Synchronized Serial Interface Blockdiagram



12.5.3 Address coincidence detection

Data communication with the SBI is started, when the master transfers an address to select a particular slave device.

The slave device detects whether the address, sent from the master, coincides with the address, assigned to the slave, by software. When the slave is in wakeup status (WUP = 1), it generates a serial transfer end interrupt request, only when it has received an address.

Coincidence address reception processing performed by software cancels the wakeup status (WUP←0), in order to prepare for subsequent command and data reception.

12.5.4 SBI mode control registers

(1) Clock-synchronized serial interface mode register (CSIM)

This is an 8-bit register that specifies the operation mode for the serial interface, serial clock, and wakeup function.

The format for this register is shown in Fig. 12-10.

The CSIM register can be read/written by an 8-bit manipulation or bit manipulation instruction. Note, however, that some bits of this register are read-only and some are write-only.

When the $\overline{\text{RESET}}$ signal has been input, the CSIM register value is initialized to 00H.

Fig. 12-10 Clock-Synchronized Serial Interface Mode Register (CSIM) Format

	⑦	⑥	⑤	4	3	2	1	0
CSIM	CTXE	CRXE	WUP	0	MOD1	0	CLS1	CLS0

CLS1	CLS0	Selects serial clock		$\overline{SCK}$ pin	Selects master/ slave in SBI mode
0	0	External clock		Input	Slave
0	1	Internal clock	Timer counter 3 output /2	Output	Master
1	0		f_{CLK} /32*		
1	1		f_{CLK} /8*		

WUP	MOD1	Operation mode	Controls wakeup function
0	0	3-line serial I/O mode	Generates interrupt request at each serial transfter
0	1	SBI mode	
1	1		Generates interrupt request only when address is received

CRXE	Reception operation
0	Disabled
1	Enabled

CTXE	Transmission operation
0	Disabled
1	Enabled

***:** f_{CLK}: internal system clock

Caution: Do not set CTXE to 1 and clear CRXE to 0, or vice versa, with a single instruction; otherwise, the serial clock counter will malfunction and communication immediately after transfer ends less than 8 bits. Therefore, use two instructions as follows:

Example: To clear CTXE to 0 and set CRXE to 1:
CLR1 CTXE
SET1 CRXE

(2) Serial bus interface control register (SBIC)

This is an 8-bit register, consisting of a bit that controls the serial bus status and flags that indicate various statuses for data input from the serial bus.

This register can be read/written by an 8-bit manipulation or bit manipulation instruction. Note, however, that some bits of this register are read-only and that some are write-only. The register format is shown in Fig. 12-11.

When the $\overline{RESET}$ signal has been input, the SBIC register value is initialized to 00H.

Fig. 12-11 SBIC Register Format (1/2)

	⑦	⑥	⑤	④	③	②	①	⓪	
SBIC	BSYE	ACKD	ACKE	ACKT	CMDD	RELD	CMDT	RELT	(00H when RESET input)

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. The RELT bit is then 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. The CMDT bit is then automatically cleared to 0.

Bus release detection flag (R)

	Clearing condition (RELD = 0)	Setting condition (RELD = 1)
RELD	① When transfer start instruction is executed ② When RESET signal is input ③ CTXE = CRXE = 0	When bus release (REL) signal is detected

Command detection flag (R)

	Clearing condition (CMDD = 0)	Setting condition (CMDD = 1)
CMDD	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. 12-11 SBIC Register Format (2/2)

Acknowledge trigger bit (W)

<table>
<tr><td>ACKT</td><td>When this bit is set after transfer, $\overline{ACK}$ is output in synchronization with the next $\overline{SCK}$. After the $\overline{ACK}$ signal has been output, this bit is automatically cleared to 0.

Note: 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</td></tr>
</table>

Acknowledge enable bit (R/W)

<table>
<tr><td rowspan="3">ACKE</td><td>0</td><td colspan="2">Disables automatic acknowledge signal output</td></tr>
<tr><td rowspan="2">1</td><td>Before transfer</td><td>Outputs $\overline{ACK}$, in synchronization with the 9th $\overline{SCK}$</td></tr>
<tr><td>After transfer</td><td>Outputs $\overline{ACK}$, in synchronization with $\overline{SCK}$ immediately after set instruction execution</td></tr>
</table>

Acknowledge detection flag (R)

<table>
<tr><td rowspan="2">ACKD</td><td>Clearing condition (ACKD = 0)</td><td>Setting condition (ACKD = 1)</td></tr>
<tr><td>① When the $\overline{SCK}$ 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)</td><td>When acknowledge signal ($\overline{ACK}$) is detected</td></tr>
</table>

Busy enable bit (R/W)

<table>
<tr><td rowspan="2">BSYE</td><td>0</td><td>① Disables automatic busy signal output
② Stops busy signal output in synchronization with $\overline{SCK}$ falling immediately after clear instruction execution</td></tr>
<tr><td>1</td><td>Outputs busy signal in synchronization with $\overline{SCK}$ falling after acknowledge signal</td></tr>
</table>

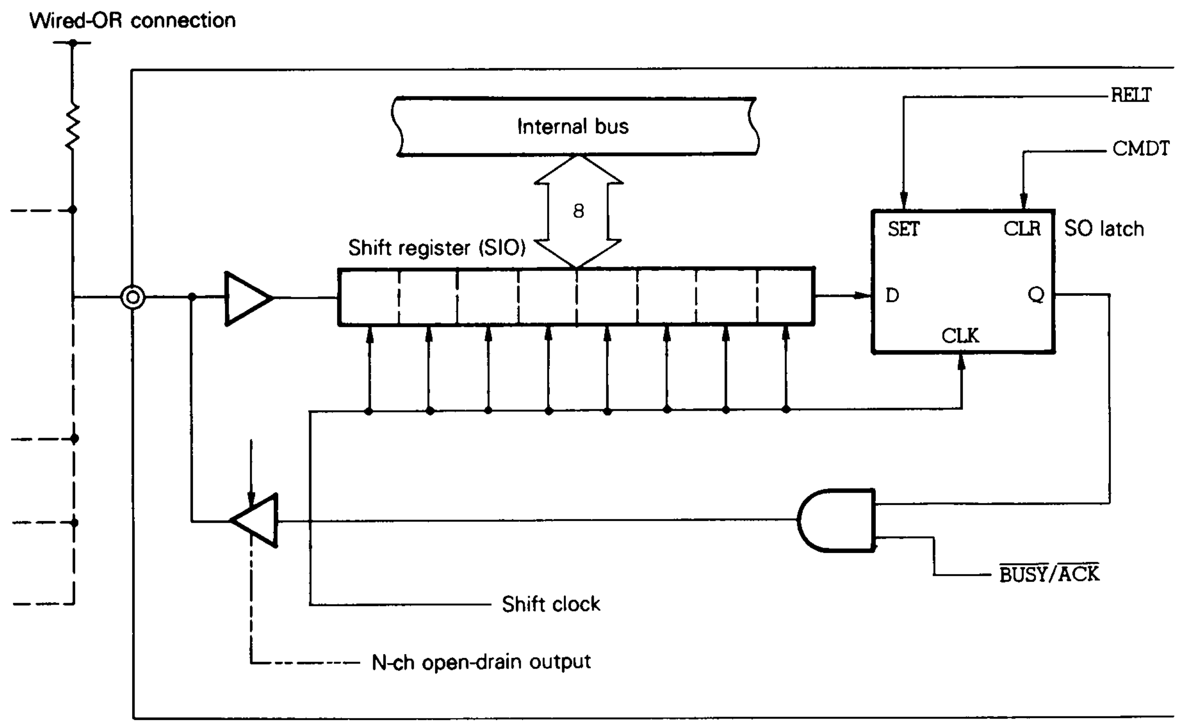
Remarks: (R) : read-only
(W) : write-only
(R/W) : read-write

(3) Shift register (SIO)

This is an 8-bit shift register that converts parallel data into serial data or vice versa.

The data written to this register is output to the serial data bus. In turn, the serial data bus inputs data to this register. Fig. 12-12 shows the shift register peripheral configuration.

Fig. 12-12 Peripheral Configuration of Shift Register



The SBI data bus uses the same pin as the input and output pins. The output pin has N-ch open-drain configuration and is wired-ORed with an external pull-up resistor. Therefore, the device, which is to receive data from this pin, must set FFH in the shift register (SIO), or disable transfer.

12.6 SBI Communication Operation and Signals

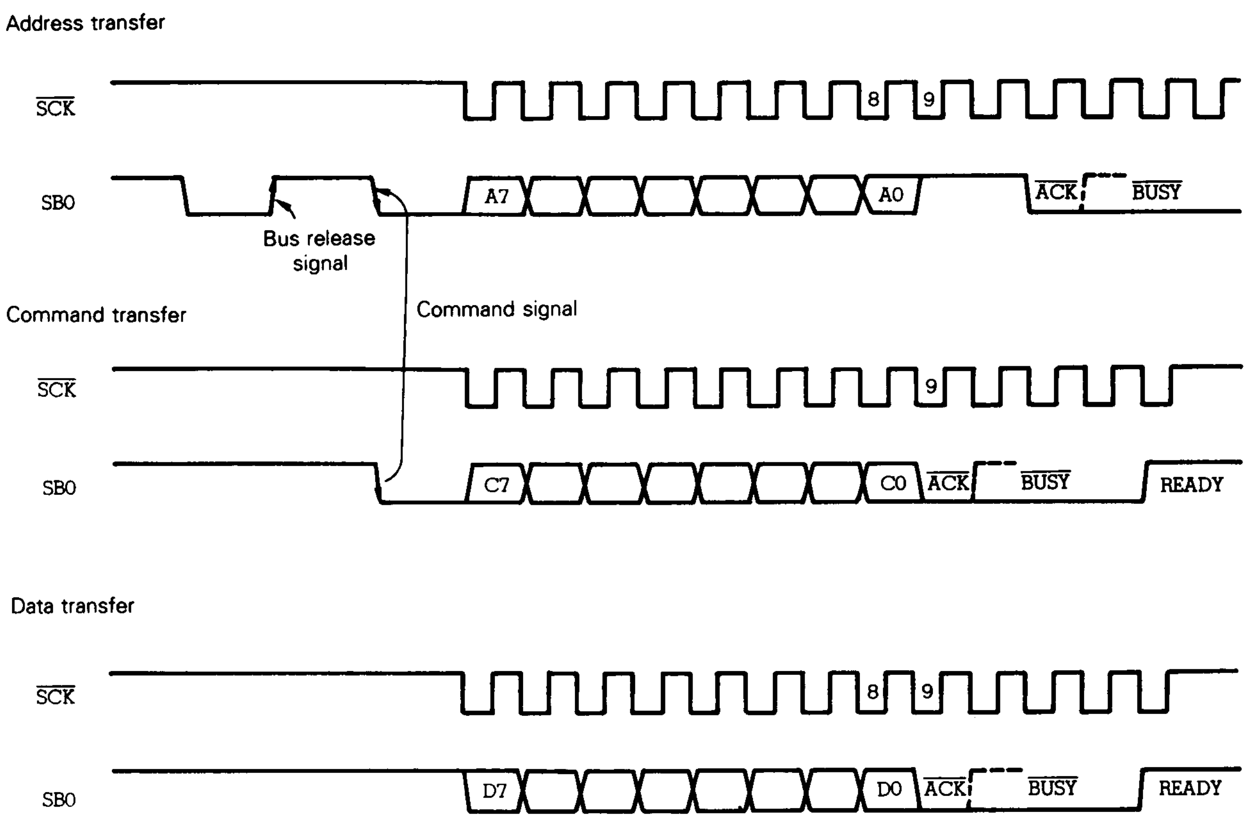
This section describes the SBI serial data format and the meanings of the signals used.

The serial data, transferred by the SBI, are classified into addresses, commands, and data. One serial data frame is made up as follows:

(Bus release signal) + (Command signal) + 8-bit data + $\overline{\text{ACK}}$ + ($\overline{\text{BUSY}}$)

Fig. 12-13 shows the address, command, and data transfer timing.

Fig. 12-13 SBI Transfer Timing



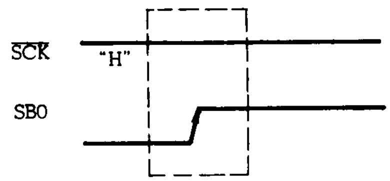
The bus release and command signals are output by the master. The $\overline{\text{BUSY}}$ signal is output by the slave. The $\overline{\text{ACK}}$ signal can be output by both the master and slave (usually, this signal is output by the device which has received 8-bit data).

The serial clock is continuously output by the master, from the start of the 8-bit data transfer until the $\overline{\text{BUSY}}$ signal is released.

12.6.1 Bus release signal (REL)

The bus release signal is the positive transition of the SB0 line (i.e., from the low level to the high level) when the $\overline{\text{SCK}}$ line is high (i.e., when the serial clock is not output). This signal is output by the master.

Fig. 12-14 Bus Release Signal

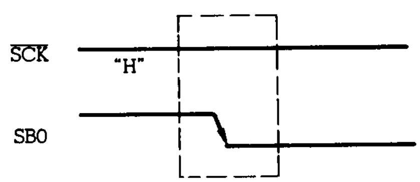


The bus release signal indicates that the master is ready to send an address to a slave, which is provided with hardware that detects the bus release signal.

12.6.2 Command signal (CMD)

The command signal is the negative transition of the SB0 line (i.e., from the high level to the low level) when the $\overline{\text{SCK}}$ line is high (i.e., when the serial clock is not output), and is output by the master.

Fig. 12-15 Command Signal

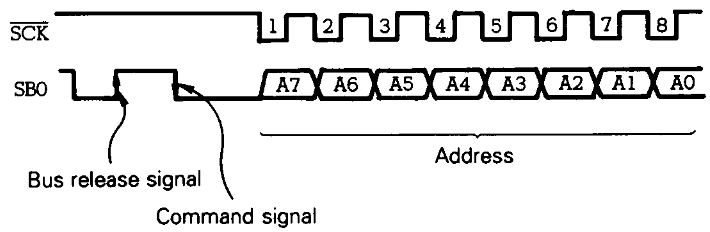


The slave is provided with hardware that detects the command signal.

12.6.3 Address

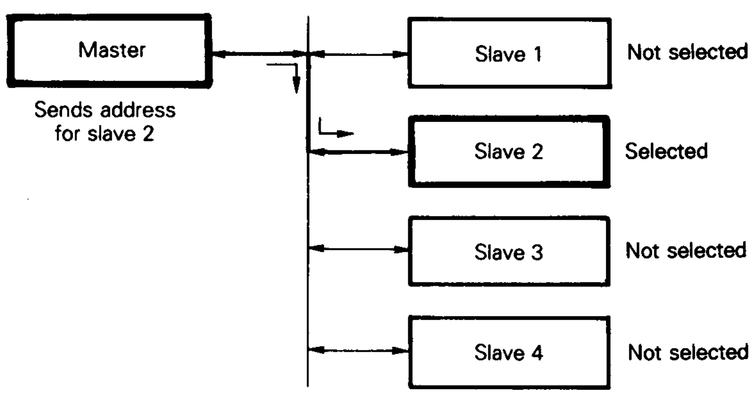
An address is 8-bit data which the master outputs to the slaves connected to the bus line, to select a particular slave.

Fig. 12-16 Address



The 8-bit data, following the bus release and command signals, is defined as an address. The slave detects this condition by hardware and checks whether the 8-bit data coincides with a number assigned to the slave (slave address) by software or hardware. If the 8-bit data coincides with the slave address, the slave is selected. Thereafter, the slave communicates with the master until it is disconnected.

Fig. 12-17 Selecting Slave by Address



12.6.4 Command and data

The master transfers commands, and transfers/receives data with a slave selected by transferring an address.

Fig. 12-18 Command

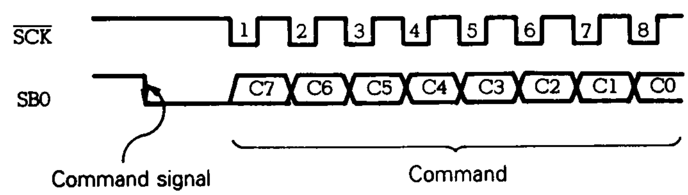
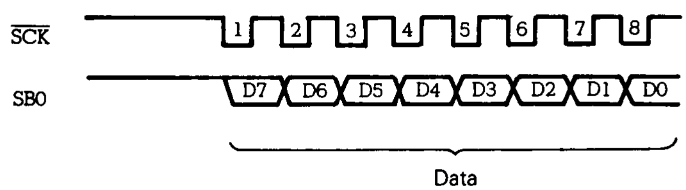


Fig. 12-19 Data



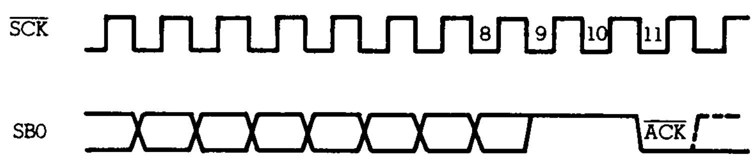
The 8-bit data following the command signal is defined as a command. 8-bit data without a command signal is defined as data. How to use the command and data can be arbitrarily determined by the communication specifications.

12.6.5 Acknowledge signal ($\overline{\text{ACK}}$)

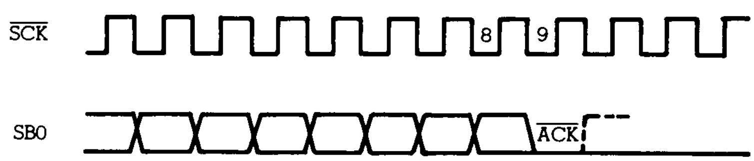
The acknowledge signal is to confirm that data sent from one device has been received by another device.

Fig. 12-20 Acknowledge Signal

[When output in synchronization with 11th $\overline{\text{SCK}}$ clock]



[When output in synchronization with 9th $\overline{\text{SCK}}$ clock]



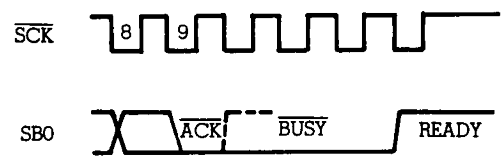
The acknowledge signal is a one-shot pulse in synchronization with the falling edge of the $\overline{\text{SCK}}$ signal, after 8-bit data has been transferred, and can be synchronized with any $\overline{\text{SCK}}$ clock.

The transmitter checks whether or not the receiver has returned the acknowledge signal, after the 8-bit data has been transferred. If the acknowledge signal is not returned within a predetermined time after the data has been transferred, the transmitter judges that the data has not been correctly received.

12.6.6 $\overline{\text{BUSY}}$ and READY signals

The BUSY signal is sent from a slave to the master to indicate that the slave is not ready to transfer/receive data. The ready signal is to indicate that the slave is ready to transfer/receive data.

Fig. 12-21 $\overline{\text{BUSY}}$ and READY Signals



With the SBI, a slave informs the master that the slave is busy by lowering the SB0 line.

The BUSY signal is output following the acknowledge signal, output by the master or a slave. The BUSY signal is set or canceled in synchronization with the falling edge of the $\overline{\text{SCK}}$ signal. When the BUSY signal has been canceled, the master automatically ends the output from the serial clock $\overline{\text{SCK}}$.

When the BUSY signal has been canceled and the slave is ready for transfer/reception, the master can immediately start the next transfer.

12.6.7 Signals

Figs. 12-22 through 12-26 show various SBI signals and flags operations on SBIC. The SBI signals are listed in Table 12-2.

Fig. 12-22 RELT, CMDT, RELD and CMDD Operations

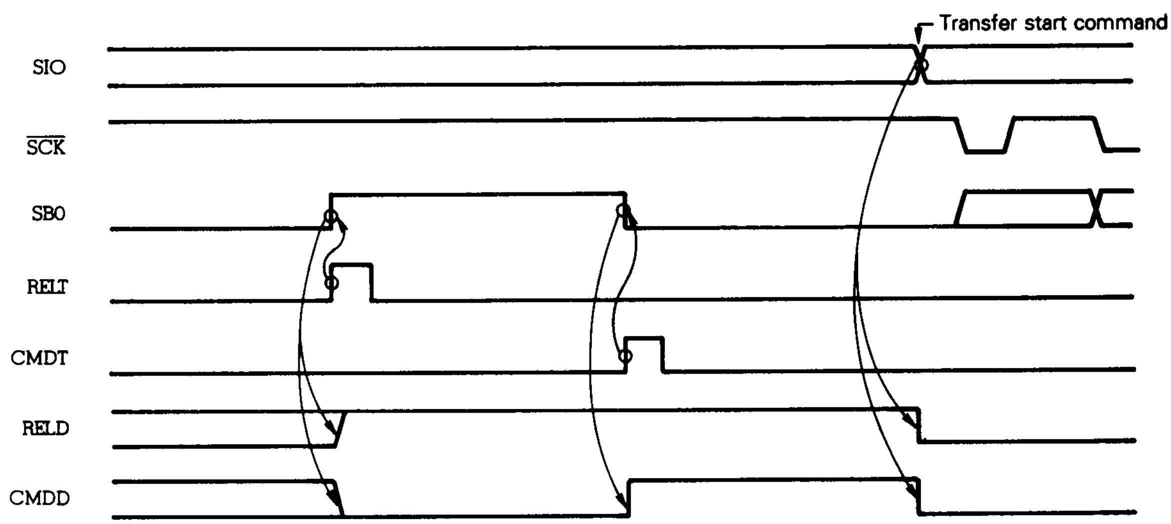
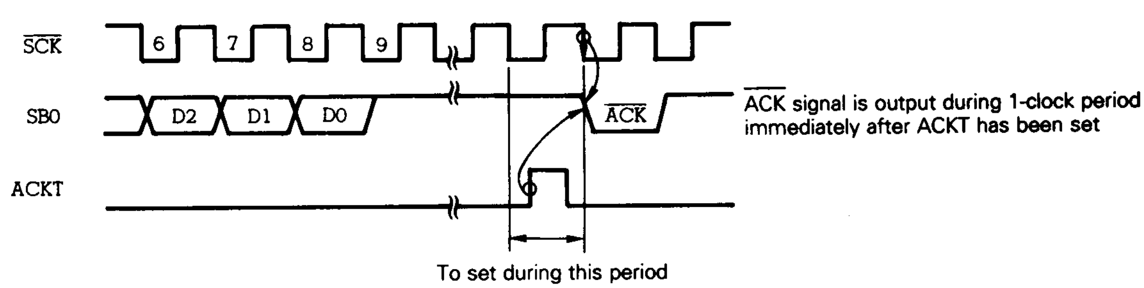


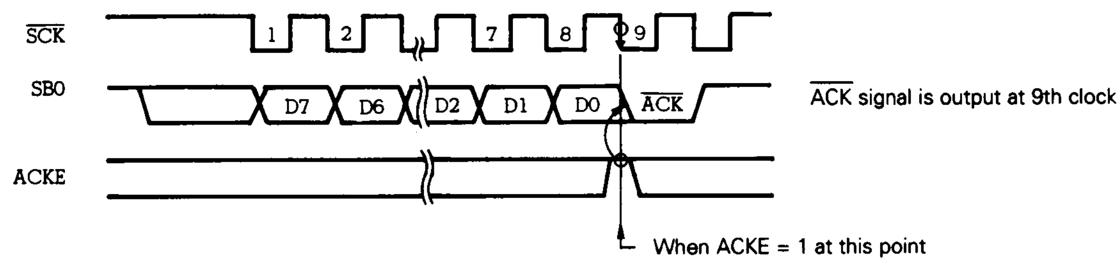
Fig. 12-23 ACKT Operation



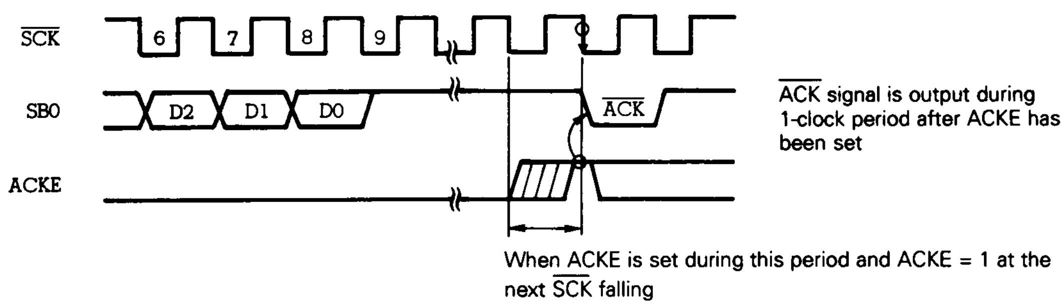
Caution: Do not set ACKT before transfer is completed.

Fig. 12-24 ACKE Operation

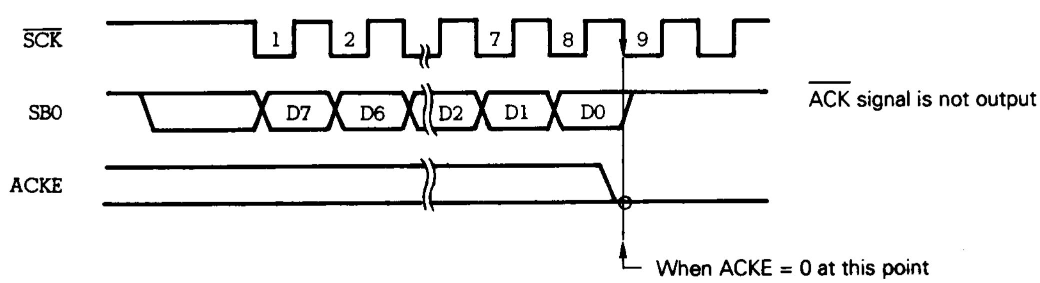
(a) When ACKE = 1 at end of transfer



(b) When set after end of transfer



(c) When ACKE = 0 at end of transfer



(d) When ACKE = 1 period is short

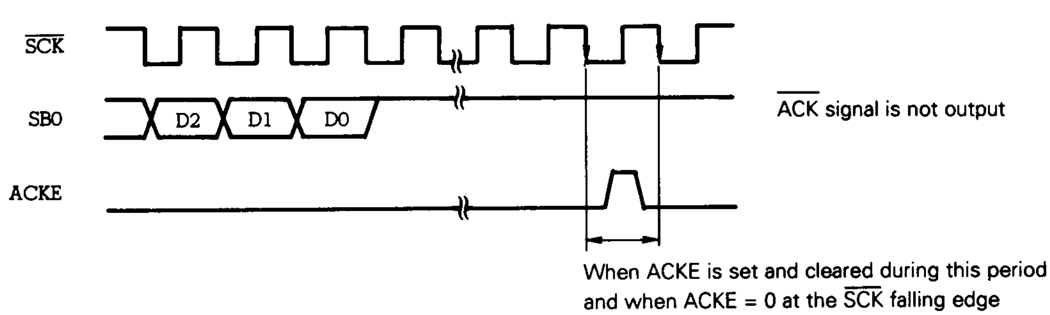
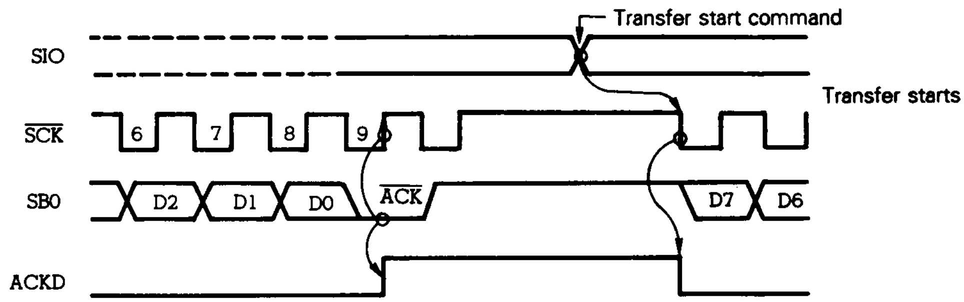
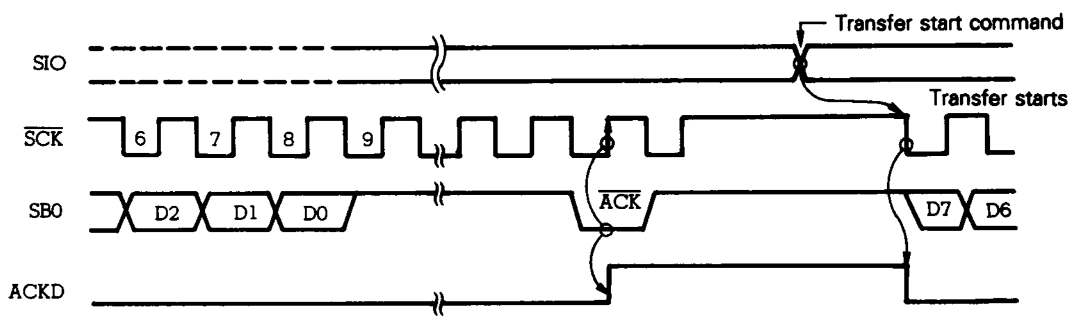


Fig. 12-25 ACKD Operation

(a) When $\overline{ACK}$ signal is output during 9th $\overline{SCK}$ clock



(b) When $\overline{ACK}$ signal is output after 9th $\overline{SCK}$ clock



(c) Clear timing when transfer start command is issued during BUSY

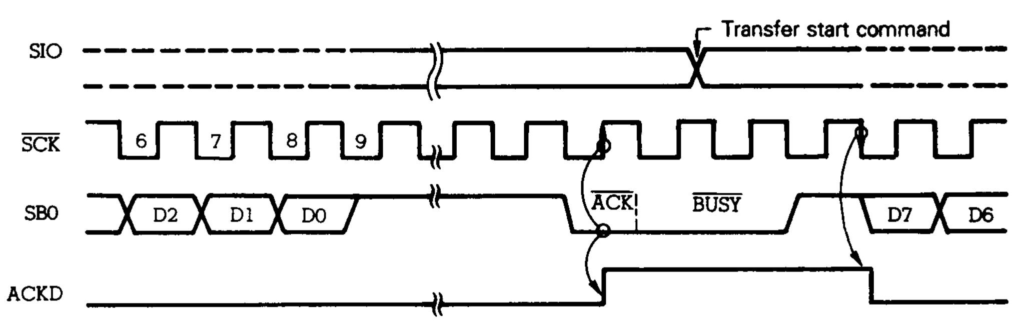


Fig. 12-26 BSYE Operation

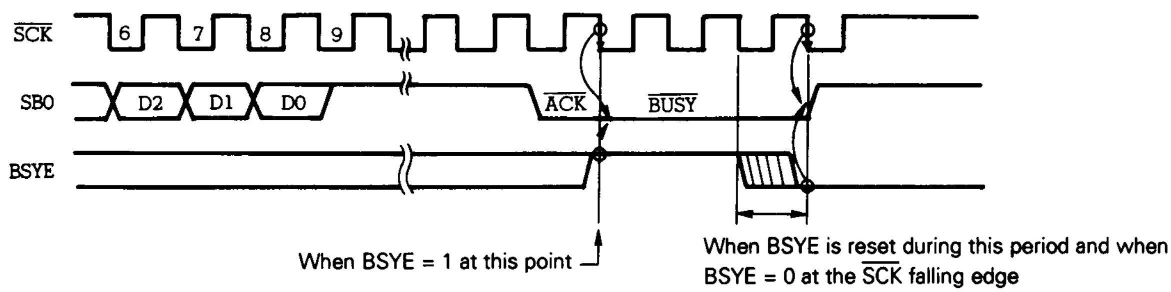


Table 12-2 SBI Signals (1/3)

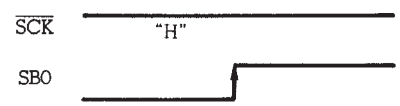
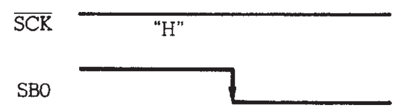
Signal	Output by:	Definition	Timing chart	Output condition	Influence on flag	Meaning
Bus release signal (REL)	Master	SB0 rising edge, when $\overline{SCK}$ = 1		• When RELT is set	• Sets RELD • Clears CMDD	CMD signal is output following this signal to indicate that transferred data is address
Command signal (CMD)	Master	SB0 falling edge, when $\overline{SCK}$ = 1		• When CMDT is set	• Sets CMDD	(1) Transferred data is address, after REL signal is output (2) Transferred data is command, if REL signal is not output

Table 12-2 SBI Signals (2/3)

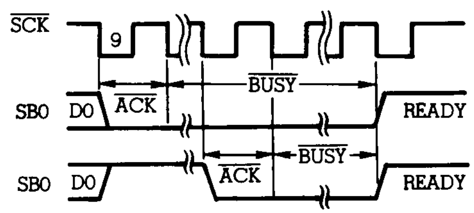
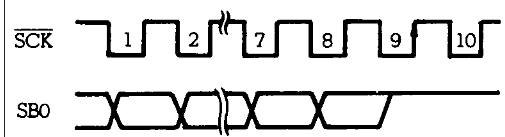
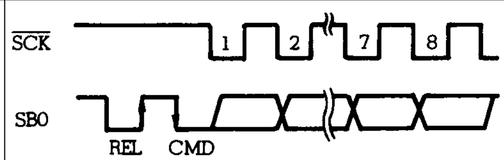
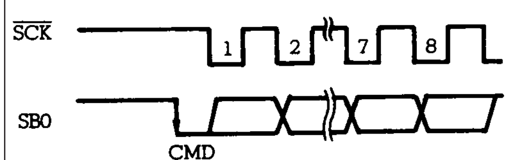
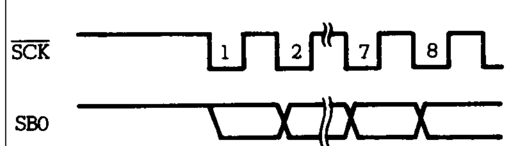
Signal	Output by:	Definition	Timing chart	Output condition	Influence on flag	Meaning
Acknowledge signal ($\overline{ACK}$)	Master/ slave	Low level signal output to SB0 during 1-clock SCK period after serial reception end		① When ACKE = 1 ② When ACKT is set	• Sets ACKD	Reception end
Busy signal ($\overline{BUSY}$)	Slave	Low level signal output to SB0, following acknow-ledge signal		• BSYE = 1	–	Serial transfer/reception is disabled, because processing is in progress
Ready signal (READY)	Slave	High level signal output to SB0, before start of, and after end of serial transfer		① BSYE = 0 ② Data write to SIO, when CTXE = 1 (serial transfer start command)*2 ③ Instruction to read from SIO, when CTXE = 0, CRXE = 1 ④ Changes in CRXE bit from 0 to 1	–	Serial transfer/reception enabled

Table 12-2 SBI Signals (3/3)

Signal	Output by:	Definition	Timing chart	Output condition	Influence on flag	Meaning
Serial clock (SCK)	Master	Synchronization clock for output of address/command/ data, ACK signal and synchronous BUSY signal. Transfers address/command/ data during the first 8 clocks period.		① Execution of instruction to write data to SIO, when CTXE = 1 (serial transfer start command)*2 ② Instruction to read data from SIO, when CTXE = 0, CRXE = 1 ③ Changes in CRXE bit from 0 to 1	Sets CSIIF (at rising edge of 8th clock)*1	Timing for signal output to serial data bus
Address (A7-A0)	Master	8-bit data transferred in synchronization with SCK, after REL and CMD signals output				Address value for slave device on serial bus
Command (C7-C0)	Master	8-bit data transferred in synchronization with SCK, after only CMD signal output. REL signal is not output				Command and message to slave device
Data (D7-D0)	Master/ slave	8-bit data transferred in synchronization with SCK, when both REL and CMD signals are not output				Data processed by slave or master

***1:** CSIIF is always set at the 8th $\overline{SCK}$ clock rising edge, when WUP = 0.
When WUP = 1, CSIIF is set at the 8th $\overline{SCK}$ clock rising edge, only when an address has been received.

***2:** Transfer is not started in the $\overline{BUSY}$ status, until the READY status is set.

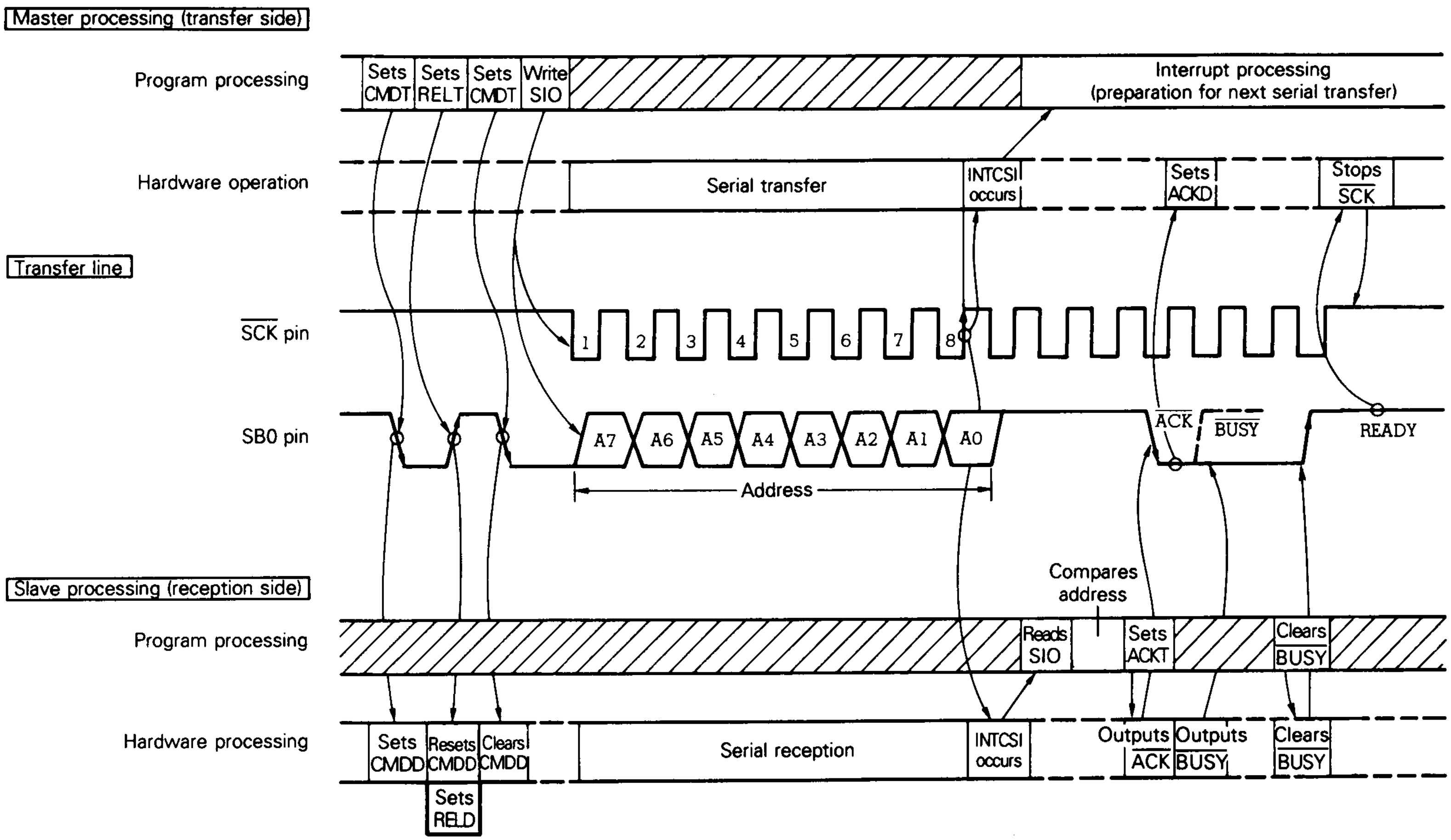
12.6.8 Communication operation

With the SBI, the master usually selects one slave device from several devices, with which the master is to communicate, by outputting an “address” to the serial bus.

After the slave device has been selected, command and data are transferred between the slave device and master to implement serial communication.

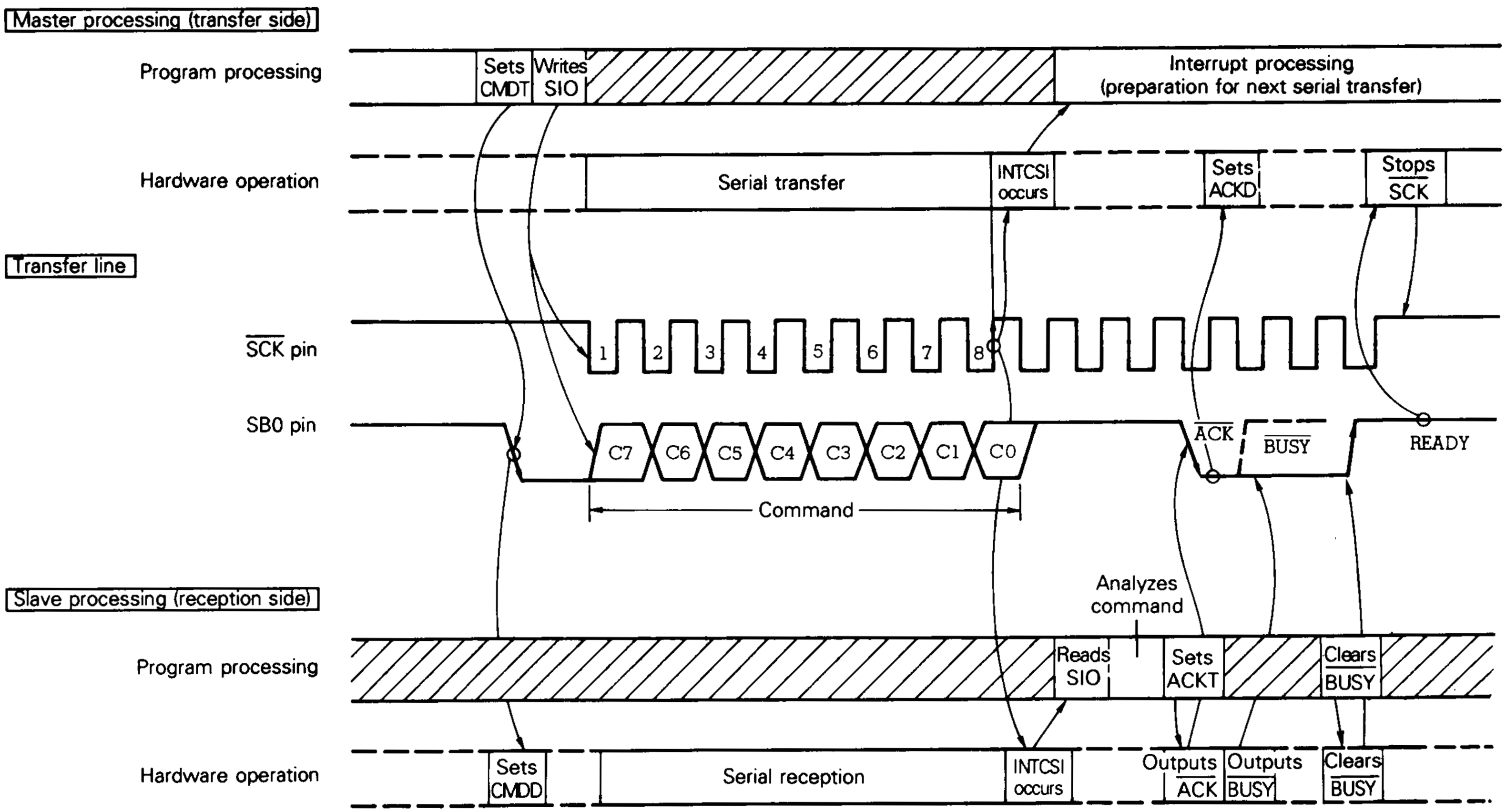
Figs. 12-27 through 12-30 show timing charts for individual data communications.

Fig. 12-27 Address Transfer from Master to Slave



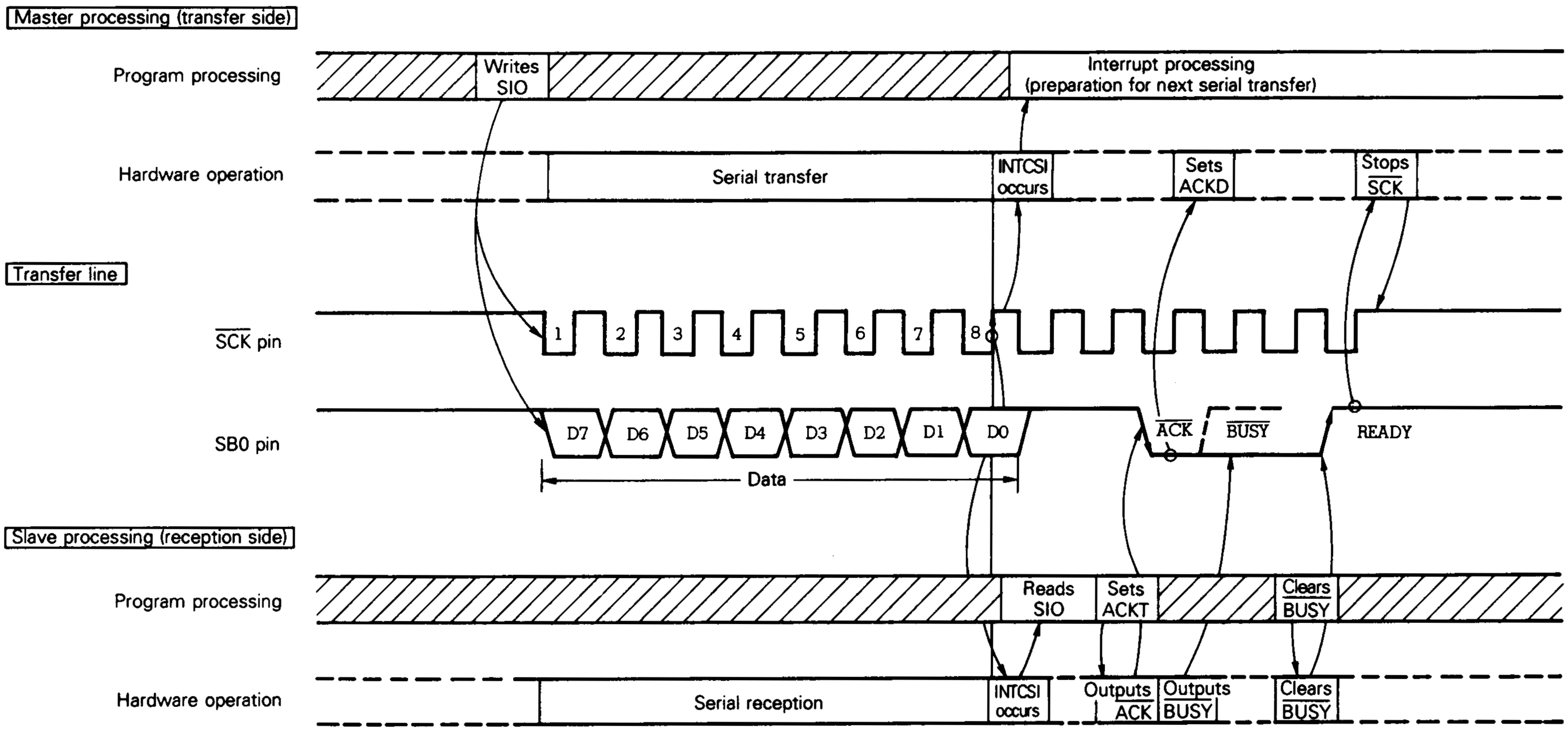
Remarks: This timing is under the following conditions:
- The master enables transfer only
- The slave enables reception only. ACKE = 0, BSYE = 1

Fig. 12-28 Command Transfer from Master to Slave



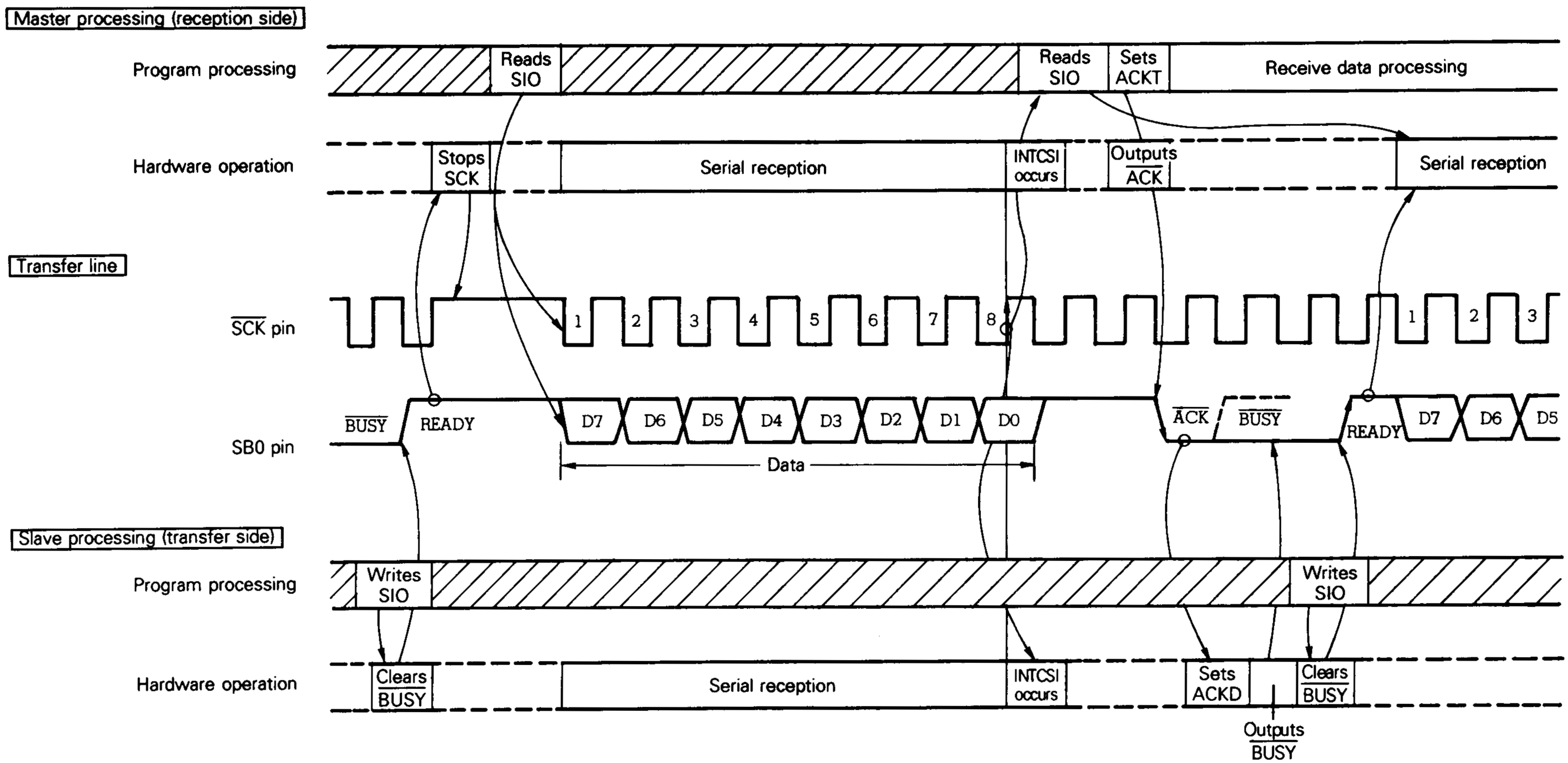
Remarks: This timing is under the following conditions:
- The master enables transfer only
- The slave enables reception only. ACKE = 0, BUSY = 1

Fig. 12-29 Data Transfer from Master to Slave



Remarks: This timing is under the following conditions:
- The master enables transfer only
- The slave enables reception only. ACKE = 0, BSYE = 1

Fig. 12-30 Data Transfer from Slave to Master



Remarks: This timing is under the following conditions:
- The master enables transfer only. ACKE = 0
- The slave enables reception only. BSYE = 1

12.6.9 Clearing BUSY

The condition under which the $\overline{\text{BUSY}}$ signal is cleared differs, depending on whether transfer/reception is enabled, considering high-speed transfer using the SBI macro service function, as shown in the following table:

Table 12-3 $\overline{\text{BUSY}}$ Clearing Conditions

Transfer/reception enable		$\overline{\text{BUSY}}$ clearing conditions
CTxE	CRxE	
0	0	None
0	1	BSYE ← 0 or SIO read access
1	0	BSYE ← 0 or SIO write access*
1	1	

***:** Write FFH to SIO, if the next operation is reception.

12.6.10 Wakeup setting operation

When WUP is set to 1, while the BUSY signal is issued, the wakeup status is set as soon as the READY signal is issued.

In the wakeup status, an interrupt (INTCSI) occurs, only when an address has been received, and the acknowledge signal ($\overline{\text{ACK}}$) is not detected.

12.6.11 Starting transmission/reception

Transmission/reception is started in the same manner as removing the BUSY signal. Transmission/reception is postponed, while a slave is issuing the $\overline{\text{BUSY}}$ signal, even when a transmission/ reception start command has been issued, and is started when the $\overline{\text{BUSY}}$ signal has been removed.

12.7 Notes

(1) Do not set CTXE to 1 and clear CRXE to 0, or vice versa, with a single instruction; otherwise, the serial clock counter will malfunction and communication immediately after transfer ends less than 8 bits. Therefore, use two instructions as follows:

Example: To clear CTXE to 0 and set CRXE to 1:
CLR1 CTXE
SET1 CRXE

(2) When exchanging the master with a slave, the master and slave switch over the input and output for the serial clock line ($\overline{SCK}$) asynchronously with each other. Therefore, a pull-up resistor must also be connected to the serial clock line ($\overline{SCK}$).

(3) Do not set ACKT before the transfer end.

CHAPTER 13 EDGE DETECTION FUNCTION

Pins P20 through P26 are provided with an edge detection function which allows the program to specify whether the rising edge or falling edge should be detected on these pins. With this function, the detected edge is sent to the internal hardware devices. Table 13-1 shows the relations between pins P20 through P26 and the detected edge applications.

Table 13-1 P20 through P26 and Detected Edge Applications

Pin	Application	Register specifying edge to be detected
P20	NMI. Control standby circuit	INTM0
P21	INTP0. Capture signal for 8-bit timer/counter 1. Trigger signal for real-time output port	
P22	INTP1. Capture signal for 8-bit timer/counter 2	
P23	INTP2. CI (count clock for 8-bit timer/counter 2)	
P24	INTP3. Capture signal for 16-bit timer/counter 0	INTM1
P25	INTP4. ASCK (external baud rate input for UART)	
P26	INTP5. Conversion start signal for A/D converter	

The edge detection function can always be effected, except in the STOP mode. (However, the edge detection function for pin P20 can be used, even in the STOP mode.)

13.1 External Interrupt Mode Registers (INTM0 and INTM1)

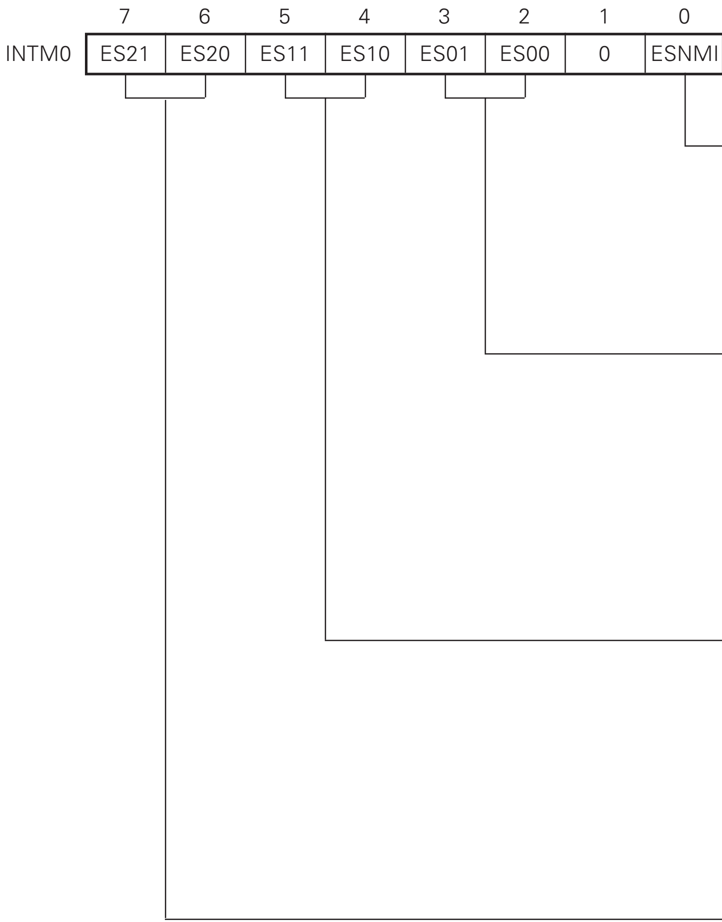
External interrupt mode registers (INTM0 and INTM1) are to specify valid edges to be detected on pins P20 through P26. The INTM0 register specifies valid edges for pins P20 through P23, while the INTM1 register specifies valid edges to be detected on pins P24 through P26.

The INTM1 register is also used to switch over between interrupts INTP4 and INTC30 (for details, refer to **CHAPTER 14 INTERRUPT FUNCTION**).

Data can be read from or written to the INTM0 and INTM1 registers by an 8-bit manipulation instruction and bit manipulation instruction. Figs. 13-1 and 13-2 show the formats for these registers.

When the $\overline{\text{RESET}}$ signal is input, the contents of these registers are initialized to 00H.

Fig. 13-1 External Interrupt Mode Register 0 (INTM0) Format



ESNMI	Specifies edge to be detected on P20 (NMI)
0	Falling edge
1	Rising edge

ES01	ES00	Specifies edge to be detected on P21 (INTP0, CR11 capture trigger, real-time output port output trigger)
0	0	Falling edge
0	1	Rising edge
1	0	Inhibited
1	1	Both falling and rising edges

ES11	ES10	Specifies edge to be detec-ted on P22 (INTP1, CR22 capture trigger)
0	0	Falling edge
0	1	Rising edge
1	0	Inhibited
1	1	Both falling and rising edge

ES21	ES20	Specifies edge to be detec-ted on P23 (INTP2, CI input)
0	0	Falling edge
0	1	Rising edge
1	0	Inhibited
1	1	Both falling and rising edges

Fig. 13-2 External Interrupt Mode Register 1 (INTM1) Format

	7	6	5	4	3	2	1	0
INTM1	0	0	ES51	ES50	ES41	ES40	ES31	ES30

ES31	ES30	Specifies edge detected on P24 (INTP3, CR02 capture trigger)
0	0	Falling edge
0	1	Rising edge
1	0	Inhibited
1	1	Both falling and rising edges

ES41	ES40	Specifies edge to be detected on P25 (INTP4)
0	0	Falling edge
0	1	Rising edge
1	0	Selects INTC30
1	1	Both falling and rising edges

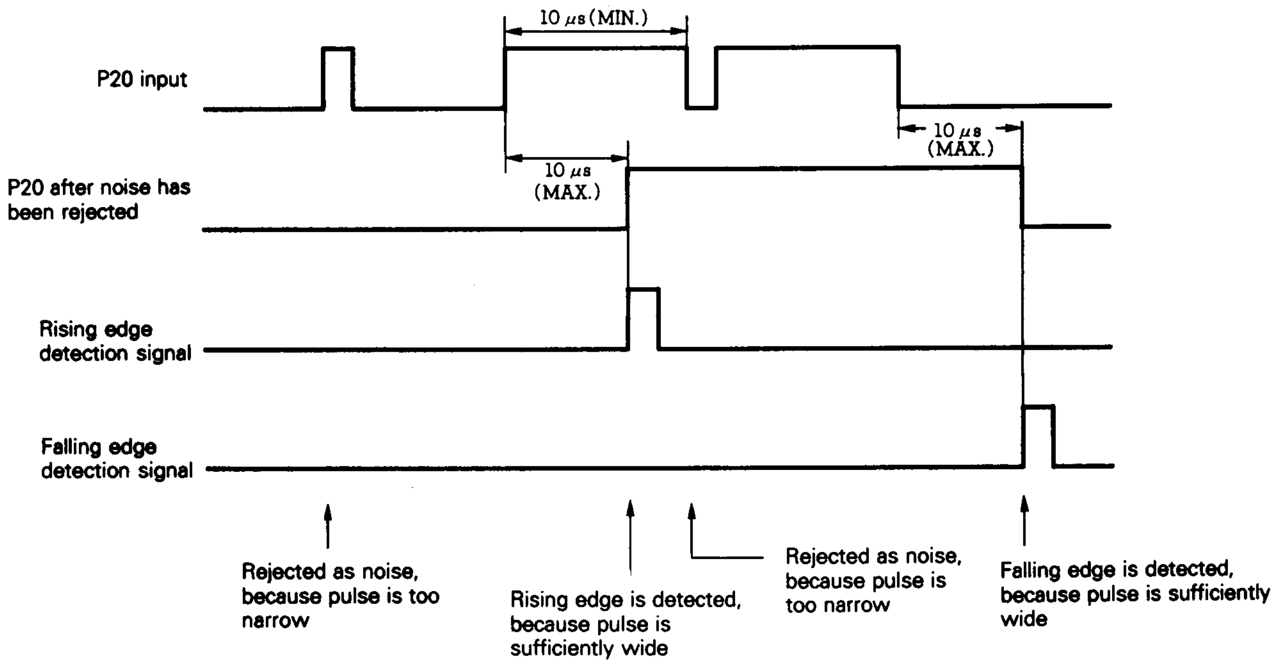
ES51	ES50	Specifies edge to be detected on P26 (INTP5, A/D conversion start signal)
0	0	Falling edge
0	1	Rising edge
1	0	Inhibited
1	1	Both falling and rising edges

Caution: When if the valid edge is changed by writing the INTM0 and INTM1 registers, the valid edge is not detected. If an edge is input while the valid edge is changed, whether or not the input edge is judged to be a valid edge is undefined.

13.2 Edge Detection on Pin P20

Pin P20 detects an edge after rejecting the noise component included in the input signal by means of analog delay. Therefore, this pin cannot detect an edge, unless the pulse for the input signal is sufficiently wide (10 μs).

Fig. 13-3 Edge Detection on Pin P20



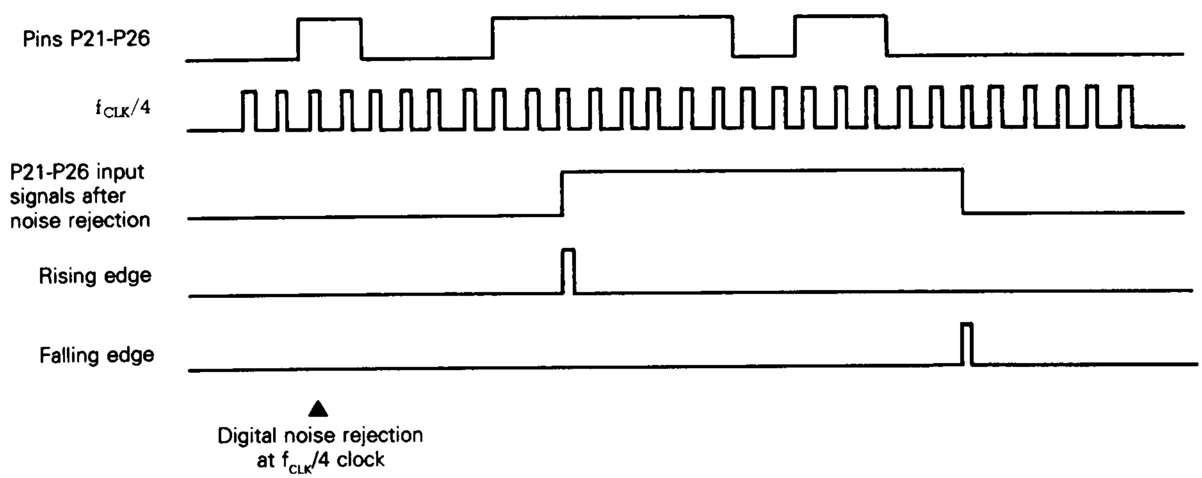
Caution: Pin P20 rejects noise by means of analog delay. Therefore, this pin detects an input edge up to 10 μs after the edge has been input. Unlike the P21 through P26 pins, the delay time, in which the edge is detected, is not a constant value, due to the differences in device characteristics.

13.3 Edge Detection on Pins P21 through P26

Input edges are detected on pins P21 through P26, after the digital noise components, included in the input signals, are rejected by clock sampling.

To reject the digital noise, sampling is performed with $f_{CLK}/4$ clock. Unless three idential signal level are consecutively input, the input signal is rejected as noise. Therefore, the level for an input signal must be held for a period of three cycles or longer (2 μs at f_{CLK} = 6 MHz, $f_{CLK} = 1/2f_{XX}$, f_{XX} = 12 MHz) at a $f_{CLK}/4$ clock, in order for the input signal to be recognized to have a valid edge.

Fig. 13-4 Edge Detection on Pins P21 through P26



Caution: 1. Since the digital noise is rejected by $f_{CLK}/4$ clock, 8 to 12 f_{CLK} clocks are required, until an edge is actually detected after the edge has been input to the pin.

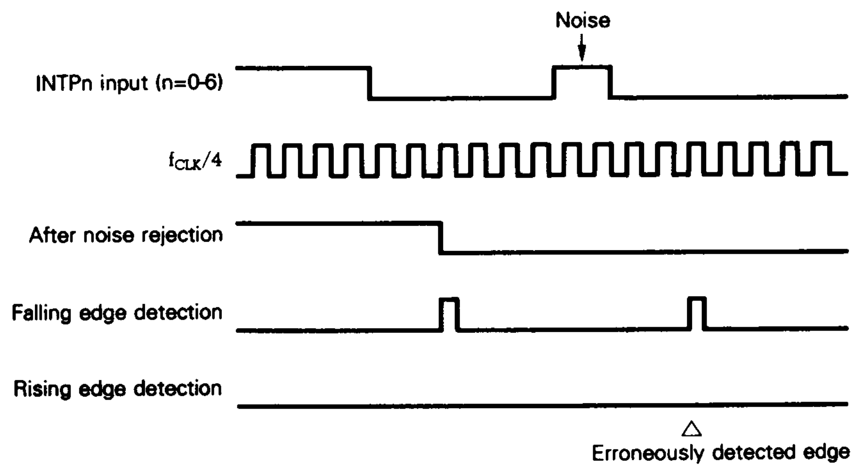
2. If the input pulse width ranges from 8 to 12 f_{CLK} clocks, the valid edge for the input signal may or may not be detected. Extend the pulse width to 12 clocks or more, to assure detecting the valid edge.

3. If the noise component input to the pin is in synchronization with the $f_{CLK}/4$ clock for μPD78234, the noise may not be rejected. In applications where such noise may be input, connect a filter to the input pin to reject the noise.

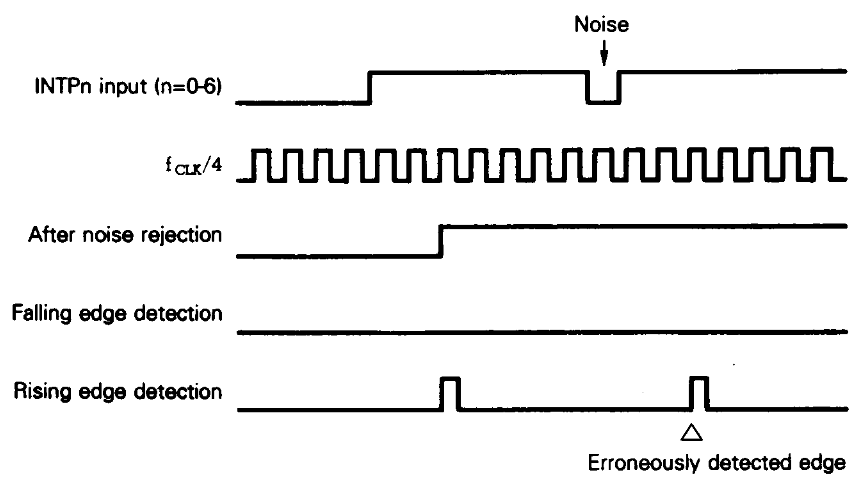
4. The in-circuit emulator cannot reject the digital noise correctly and may erroneously detect a falling edge due to noise while a low level is input, or a rising edge while a high level is input (see Fig. 13-5).
When port 2 is read, the noise is read without rejected.

Fig. 13-5 Erroneous Detection of Edge

(a) Erroneous edge detection during low level input



(b) Erroneous edge detection during high level input



Influences on the real-time output port, A/D converter, and timer/counter due to noise are as follows:

- Real-time output port:
  Operates according to the erroneously detected edge.
- A/D converter:
  Operates according to the erroneously detected edge.
- Capture/clear operation of timer/counter:
  Not influenced by the erroneously detected edge. Therefore, even if an interrupt occurs due to the erroneously detected edge, the capture value is not updated. The value of CR22 becomes undefined after it has been read by the CPU.
- Compare operation of timer/counter:
  When a mode in which the clear operation is performed after capture is set, and when timer/counter 2 is used as an external event counter, the coincidence interrupt generation timing is change due to influence of the erroneously detected edge. Consequently, the coincidence interrupt occurs even when the timer/counter does not coincide with the compare register.
  - When the mode in which the clear operation is performed after capture is set, the correct coincidence interrupt generation timing is restored by the normal edge input or by stopping the timer/counter.
  - When timer/counter 2 is used as an external event counter, the correct coincidence interrupt generation timing is restored by stopping the timer/counter.

  The timer output is not influenced by the erroneously detected edge, and operates at the normal timing.

13.4 Notes

(1) The valid edge is not detected when it has been changed by writing the INTM0 and INTM1 registers. When an edge is input while the valid edge is changed, the edge may or may not be judged as valid.

(2) Since the P20 pin rejects noise by means of analog delay, it detects the edge up to 10 μs after an edge has actually been input. Unlike the P21 through P26 pins, the delay time in which the edge is detected is not a constant value due to the differences in characteristics of devices.

(3) Since digital noise is rejected by the $f_{CLK}/4$ clock, 8 to 12 f_{CLK} clocks are required until the actual edge is detected after an edge has been input to the pin.

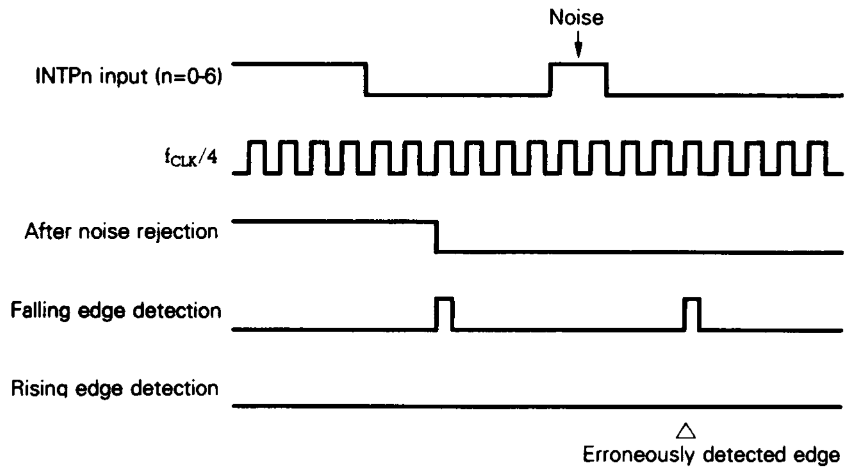
(4) If the width of an input pulse is 8 to 12 f_{CLK} clocks, the valid edge may or may not be detected. For accurate operation, fix the level for a period of 12 clocks or longer.

(5) If the noise input to the pin is in synchronization with the $f_{CLK}/4$ clock in μPD78234, it may not be detected as noise. If this kind of noise input is expected, reject the noise by using an input pin filter.

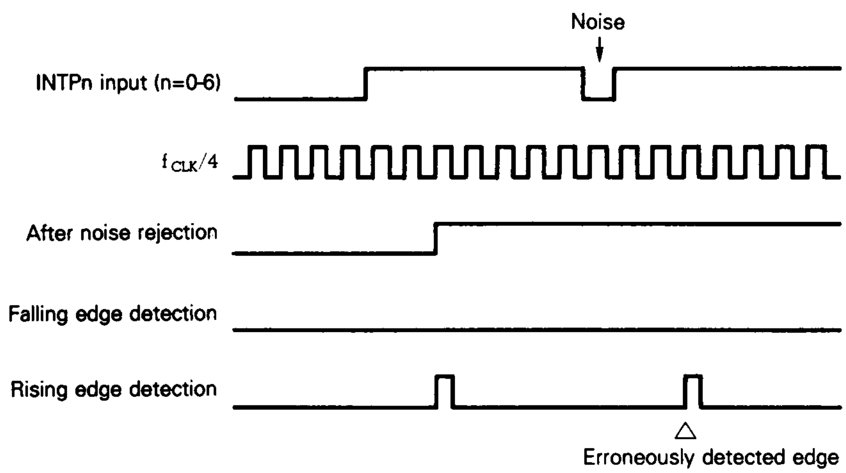
(6) The in-circuit emulator cannot reject the digital noise correctly and may erroneously detect a falling edge due to noise while a low level is input, or a rising edge while a high level is input (see **Fig. 13-6**).
When port 2 is read, the noise is read without rejected.

Fig. 13-6 Erroneous Detection of Edge

(a) Erroneous edge detection during low level input



(b) Erroneous edge detection during high level input



Influences on the real-time output port, A/D converter, and timer/counter due to noise are as follows:

- Real-time output port:
  Operates according to the erroneously detected edge.
- A/D converter:
  Operates according to the erroneously detected edge.
- Capture/clear operation of timer/counter:
  Not influenced by the erroneously detected edge. Therefore, even if an interrupt occurs due to the erroneously detected edge, the capture value is not updated. The value of CR22 becomes undefined after it has been read by the CPU.
- Compare operation of timer/counter:
  When a mode in which the clear operation is performed after capture is set, and when timer/counter 2 is used as an external event counter, the coincidence interrupt generation timing is change due to influence of the erroneously detected edge. Consequently, the coincidence interrupt occurs even when the timer/counter does not coincide with the compare register.
  - When the mode in which the clear operation is performed after capture is set, the correct coincidence interrupt generation timing is restored by the normal edge input or by stopping the timer/counter.
  - When timer/counter 2 is used as an external event counter, the correct coincidence interrupt generation timing is restored by stopping the timer/counter.

  The timer output is not influenced by the erroneously detected edge, and operates at the normal timing.

CHAPTER 14 INTERRUPT FUNCTIONS

μPD78234 is provided with the following two interrupt request processing modes. These modes can be set freely by the program. Interrupt processing by means of a macro service can be selected only for the interrupt request sources shown in Table 14-1, which are provided with macro service processing modes.

Table 14-1 Interrupt Request Processing Modes

Interrupt request processing mode	Processed by:	PC and PSW contents	Processing format
Vector interrupt	Software	Saved and restored	Execution branches to be specified service program
Macro service	Hardware (firmware)	Retained	Predetermined processing, such as data transfer between memory and I/O, is executed

As for the maskable vector interrupt, multiplexed processing control with two levels of priority can be performed.

14.1 Interrupt Request Sources

μPD78234 has 19 interrupt request sources as shown in Table 14-2. Each of these sources is assigned an interrupt vector table.

Table 14-2 Interrupt Request Sources Categories

Interrupt request	Default priority	Interrupt source	Generating unit	Macro service	Vector table address
Software	None	BRK instruction execution		None	003EH
Non-maskable	None	NMI (edge input to pin is detected)	Edge detection	None	0002H
Maskable	0	INTP0 (edge input to pin is detected)		A, B	0006H
	1	INTP1 (edge input to pin is detected)		A, B	0008H
	2	INTP2 (edge input to pin is detected)		A, B	000AH
	3	INTP3 (edge input to 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 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 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 receive 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

Remarks: "Default priority" is fixed by the hardware. If two or more interrupts, each having the same priority occur at the same time, they are processed according to their default priorities.

14.1.1 Software interrupt request

When the BRK instruction that generates a vector interrupt is executed, an interrupt request is generated by software.

The interrupt request, issued by executing the BRK instruction, can be accepted, even when interrupts are disabled, but it is not subject to interrupt priority control.

When the BRK instruction is executed, the vector table contents are unconditionally set in the PC and execution branches.

By executing the BRK instruction in a BRK service routine, the service routine can nest itself.

To exit from the BRK service routine, execute the RETB instruction.

14.1.2 Nonmaskable interrupt request

The nonmaskable interrupt request is input to the NMI pin. When the valid edge, specified by the bit 0 (ESNMI) for the external interrupt mode register 0 (INTM0), is input to the NMI pin, an interrupt request is generated.

Nonmaskable interrupt requests can be accepted unconditionally, even when interrupts are disabled. In addition, the priority for the nonmaskable interrupt is not controlled. Therefore, the nonmaskable interrupt takes precedence over any other interrupts.

14.1.3 Maskable interrupt request

Maskable interrupts can be masked by appropriately setting interrupt mask register (MK0). In addition, all the maskable interrupts can be disabled or enabled by the IE flag of the PSW.

A default priority is assigned to each maskable interrupt request, as shown in Table 14-2, so that, when two or more interrupts, each 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 multiplexed processing can be controlled. However, the macro service is accepted independently from the priority control and the IE flag.

14.1.4 Selecting interrupt source

Interrupts INTP4 and INTC30, and INTP5 and INTAD cannot be used at the same time, because these interrupts share the same vector table and control flags, such as interrupt request flags. Therefore, whether INTP4 or INTC30, or whether INTP5 or INTAD is used must be selected by the program. For the selected interrupt source, the vector table, interrupt request flag (PIFn: n = 4 or 5), interrupt mask flag (PMKn: n = 4 or 5), interrupt service mode flag (PISMn: n = 4 or 5), and priority specification flag (PPRn: n = 4 or 5) are used. The selected interrupt source generates an interrupt or macro service. Interrupt request sources not selected cannot use these flags. Therefore, they cannot generate an interrupt or macro service.

The other interrupt sources do not have to be selected, because they have their own vector tables and control flags.

(1) Selecting INTP4 and INTC30

Interrupt INTP4 or INTC30 is selected by the ES40 and ES41 bits of external mode register 1 (INTM1).

Data can be read from or written to the INTM1 register by an 8-bit manipulation or bit manipulation instruction.

The format for this register is shown in Fig. 14-1.

When the $\overline{\text{RESET}}$ signal is input, the contents of this register are cleared to 00H, so that INTP4 occurs at the rising edge of the INTP4 pin.

Fig. 14-1 INTM1 Register Format

	7	6	5	4	3	2	1	0
INTM1	0	0	ES51	ES50	ES41	ES40	ES31	ES30

ES51, ES50, ES31, ES30: Specifies edge to be detected on P24 and P26 pins (see **Fig. 13-2**)

ES41	ES40	Selects interrupt request source
0	0	INTP4 (which occurs at falling edge for INTP4 pin)
0	1	INTP4 (which occurs at rising edge for INTP4 pin)
1	0	INTC30 (which occurs when TM3 coincides with CR30)
1	1	INTP4 (which occurs at both falling and rising edges for INTP4 pin)

(2) Selecting INTP5 and INTAD

Interrupt INTP5 or INTAD is selected by the A/D converter mode register (ADM). Either of these interrupts is automatically selected, according to the specified operation mode for the A/D converter.

Data can be read from or written to the ADM register by an 8-bit manipulation or bit manipulation instruction. The format for this register is shown in Fig. 14-2. For details regarding for the A/D converter operations, refer to **CHAPTER 9 A/D CONVERTER**.

When the $\overline{\text{RESET}}$ signal is input, the ADM register contents are initialized to 00H.

Fig. 14-2 ADM Register Format

	⑦	6	5	4	3	2	1	0
ADM	CS	TRG	0	FR	ANIS2	ANIS1	ANIS0	MS

FR, ANIS2, ANIS1, ANIS0: These bits do not affect interrupt source selection (for details regarding A/D converter operation, refer to **Fig. 9-3**).

CS, TRG, MS:

CS	TRG	MS	Selected interrupt request source
0	x	x	INTP5 (valid edge input to INTP5)
1	0	0	INTAD (each time A/D conversion ends)
1	0	1	INTP5 (valid edge input to INTP5)
1	1	x	INTAD (each time A/D conversion ends)

14.2 Interrupt Processing Control Registers

The following six registers control interrupt processing:

- 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)

IF0, MK0, ISM0, and PR0 are 16-bit read/write registers. The contents of these registers can be manipulated in 16 or 8 bit units. In addition, each bit for these registers can be set or cleared by a bit manipulation instruction. The IST register and PSW are 8-bit read/write registers, whose contents can be manipulated in 8- or 1-bit unit. Furthermore, the IE flag for the PSW can be manipulated by a dedicated instruction. Figs. 14-3 through 14-8 show the formats for these registers.

Table 14-3 below lists the interrupt request flags, interrupt mask flags, interrupt service mode flags, and priority specification flags corresponding to each interrupt request source.

Table 14-3 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 INTC30	PIF4	PMK4	PISM4	PPR4
INTP5 INTAD	PIF5	PMK5	PISM5	PPR5
INTC20	CIF20	CMK20	CISM20	CPR20
INTSER	SERIF	SERMK	–	SERPR
INTSR	SRIF	SRMK	SRISM	SRPR
INTST	STIF	STMK	STISM	STPR
INTCSI	CSIIF	CSIMK	CSIISM	CSIPR

14.2.1 Interrupt request flag register (IFO)

This 16-bit register consists of interrupt request flags. Each flag is set to 1, when the corresponding interrupt request has been generated, and is cleared to 0 when a vector interrupt is accepted or macro service processing is executed.

When the $\overline{\text{RESET}}$ signal has been input, the contents of this register are cleared to 0000H.

Fig. 14-3 Interrupt Request Flag Register (IFO) Format

	⑦	⑥	⑤	④	③	②	①	⓪
IF0L	CIF11	CIF10	CIF01	CIF00	PIF3	PIF2	PIF1	PIF0
IF0H	CSIIF	STIF	SRIF	SERIF	CIF20	PIF5	PIF4	CIF21

Interrupt request flags

0	Interrupt request is not generated
1	Interrupt request is generated

14.2.2 Interrupt mask register (MK0)

This 16-bit register consists of interrupt mask flags. Each interrupt mask flag enables or disables the corresponding interrupt request.

When the $\overline{\text{RESET}}$ signal is input, the contents of this register are initialized to FFFFH, disabling all maskable interrupts.

When the interrupt mask flag is set to 1, the corresponding interrupt is disabled and is prevented from being accepted.

When the interrupt mask flag is reset to 0, the corresponding interrupt request can be accepted as a vector interrupt or macro service.

Fig. 14-4 Interrupt Mask Register (MK0) Format

	⑦	⑥	⑤	④	③	②	①	⓪
MK0L	CMK11	CMK10	CMK01	CMK00	PMK3	PMK2	PMK1	PMK0
MK0H	CSIMK	STMK	SRMK	SERMK	CMK20	PMK5	PMK4	CMK21

Interrupt mask flags

0	Enables interrupt
1	Disables interrupt

14.2.3 Interrupt service mode register (ISM0)

ISM0 is a 16-bit register consisting of interrupt service mode flags.

When each service mode flag is 0, the corresponding interrupt request is processed in the vector interrupt mode; when the flag is 1, the interrupt request is processed in the macro service mode. When a macro service request is processed the specified number of times, the flag is cleared to 0.

When the $\overline{\text{RESET}}$ signal is input, the contents of this register are initialized to 0000H, specifying the vector interrupt processing.

Fig. 14-5 Interrupt Service Mode Register (ISM0) Format

	⑦	⑥	⑤	④	③	②	①	⓪
ISM0L	CISM11	CISM10	CISM01	CISM00	PISM3	PISM2	PISM1	PISM0
ISM0H	CSIISM	STISM	SRISM	0	CISM20	PISM5	PISM4	CISM21

Interrupt service mode flags

0	Vector interrupt processing
1	Macro service processing

14.2.4 Priority specification flag register (PR0)

This 16-bit register consists of interrupt priority flags that are used to control nesting of interrupts.

This register can specify two interrupt groups, with one having a higher priority and the other having a lower priority. When each priority flag is 0, the corresponding interrupt is specified to belong to the high-priority group; when the flag is 1, the interrupt is specified to be in the low-priority group.

When an interrupt has been accepted, the interrupt specification flag, corresponding to the accepted interrupt, is transferred to the ISP bit for the PSW.

While a low-priority vector interrupt is processed, vector interrupts with lower and higher priorities can be nested, when interrupts are enabled. While a high-priority interrupt is processed, only high-priority vector interrupts can be nested, if interrupts are enabled. Note, however, that a macro service can be accepted, regardless of the priority for the interrupt.

When the $\overline{\text{RESET}}$ signal is input, the contents of this register are initialized to FFFFH, specifying all interrupts to be in the low-priority group.

Fig. 14-6 Priority Specification Flag Register (PR0) Format

	⑦	⑥	⑤	④	③	②	①	⓪
PR0L	CPR11	CPR10	CPR01	CPR00	PPR3	PPR2	PPR1	PPR0
PR0H	CSIPR	STPR	SRPR	SERPR	CPR20	PPR5	PPR4	CPR21

Priority specification flags

0	High priority
1	Low priority

14.2.5 Interrupt status register (IST)

This 8-bit register controls nesting of interrupts accepted by the nonmaskable interrupt request pin (NMI) and, at the same time, indicates the accepting status for the nonmaskable interrupt.

If another nonmaskable interrupt request occurs, while one nonmaskable interrupt processing is performed, the second interrupt request is accepted if the NMIS bit of this register is cleared to 0. If the NMIS bit is set to 1, the second interrupt request is not accepted.

The NMIS bit is set to 1, when a nonmaskable interrupt has been accepted. The NMIS bit is reset to 0, when execution is returned from the processing requested by a nonmaskable interrupt (i.e., when the RETI instruction is executed).

Data can be read from or written to the IST register by an 8-bit manipulation instruction or bit manipulation instruction. The NMIS flag is set to 1, when a nonmaskable interrupt has been accepted, and is reset to 0, when the RETI instruction is executed. The format for this register is shown in Fig. 14-7.

When the $\overline{\text{RESET}}$ signal is input, the contents of this register are initialized to 00H.

Fig. 14-7 Interrupt Status Register (IST) Format

	7	6	5	4	3	2	1	0
IST	0	0	0	0	0	0	0	NMIS

NMIS	Nesting nonmaskable interrupts	Accepting status for for nonmaskable interrupt
0	Enabled	Execution exits from nonmaskable interrupt processing or nonmaskable interrupt request is not accepted
1	Disabled	Nonmaskable interrupt request is accepted or being processed

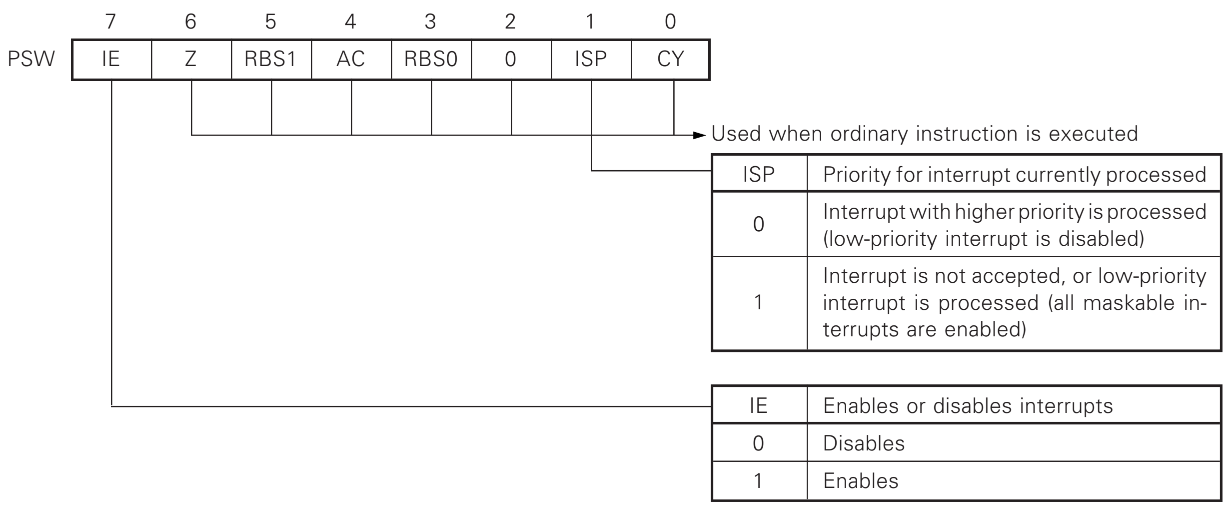
14.2.6 Program status word (PSW)

The PSW is a register that retains and indicates the results of an executed instruction and the current status vis-a-vis interrupt requests. On this register, the IE flag, which enables or disables the maskable interrupt, and ISP flag, which controls multiplexed processing, are mapped.

The PSW can be read or written in 8-bit units. It can also be manipulated by a bit manipulation instruction and sole use instructions (EI and DI). When a vector interrupt has been accepted, or when the BRK instruction has been executed, the PSW contents are saved to stack, and the IE flag is reset to 0. When a maskable interrupt has been accepted, the contents of the priority specification flag, corresponding to the accepted interrupt, are transferred to ISP. ISP is reset (0), when a non-maskable interrupt has been accepted. The PSW contents are also saved to stack by the PUSH PSW instruction, and are restored from stack by the RETI, RETB, and POP PSW instructions.

When the $\overline{\text{RESET}}$ signal has been input, the PSW contents are initialized to 02H.

Fig. 14-8 Program Status Word Format



14.3 Interrupt Processing

14.3.1 Accepting software interrupt

The software interrupt is accepted, when the BRK instruction has been executed. The software interrupt cannot be disabled.

When the software interrupt has been accepted, the PSW and PC contents are saved to stack in that order, the IE flag is reset, the vector table (003EH and 003FH) contents are loaded to the PC, and the program execution branches.

To return from the software interrupt, use the RETB instruction.

Caution: Do not use the RETI instruction to return from the software interrupt.

14.3.2 Accepting nonmaskable interrupt

The nonmaskable interrupt can be accepted, even when interrupts are disabled. This interrupt is not subject to interrupt priority control and therefore takes precedence over any other interrupts.

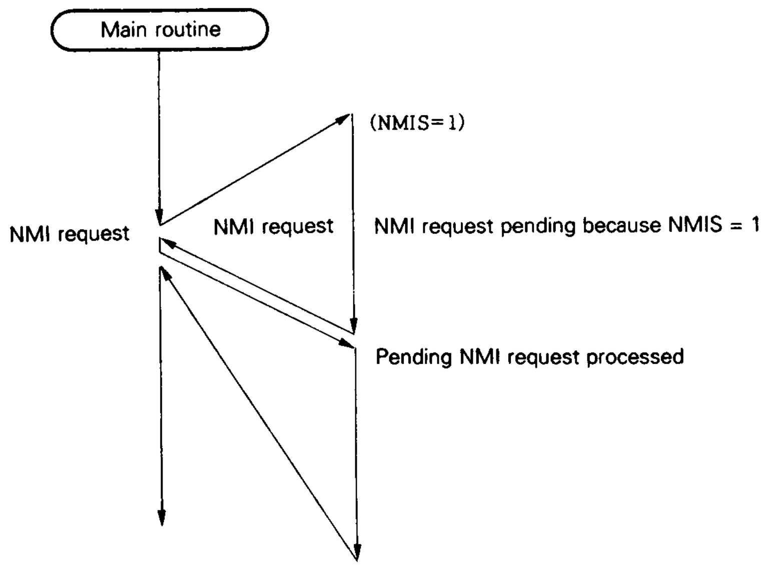
When the nonmaskable interrupt has been accepted, the NMIS bit for the status register (IST) is set to 1, the PSW and PC contents are saved to stack in that order, the IE and ISP flags are reset to 0, the vector table contents are loaded to the PC, and execution branches. The NMIS bit is reset to 0 by the RETI instruction.

When the NMIS bit is set to 1, no new nonmaskable interrupt is accepted and kept pending until the NMIS bit is reset to 0. Usually, therefore, a new nonmaskable interrupt is not accepted while a nonmaskable interrupt service program is executed. A new nonmaskable interrupt that has occurred, while the nonmaskable interrupt service program is executed, is accepted after the current execution of the nonmaskable interrupt service program has been completed (i.e., after the RETI instruction has been executed). However, if two or more new nonmaskable interrupt requests are generated, while a nonmaskable interrupt service program is executed, only one nonmaskable interrupt is accepted, after the nonmaskable interrupt service program has been executed.

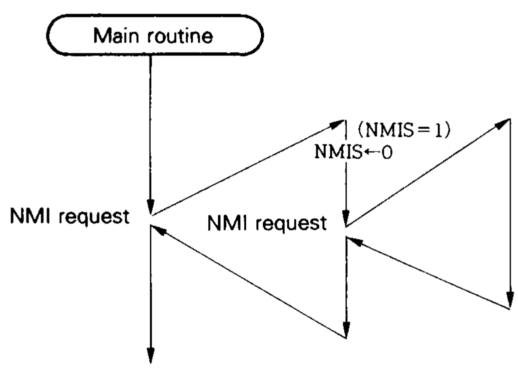
To multiplex nonmaskable interrupt requests, reset the NMIS bit to 0 in a nonmaskable interrupt service program. When NMIS is reset to 0, the nonmaskable interrupt is accepted, even while a nonmaskable interrupt service program is executed.

Fig. 14-9 Accepting NMI Interrupt Request

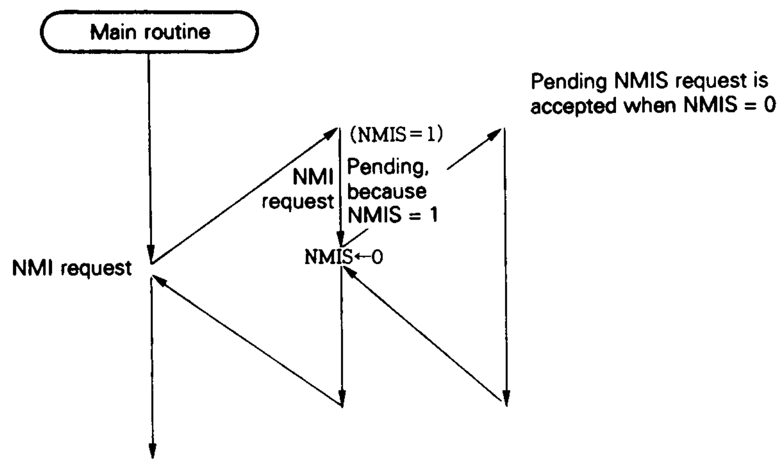
(a) If new NMI request is generated, while NMI service program is executed (when IST register is not manipulated)



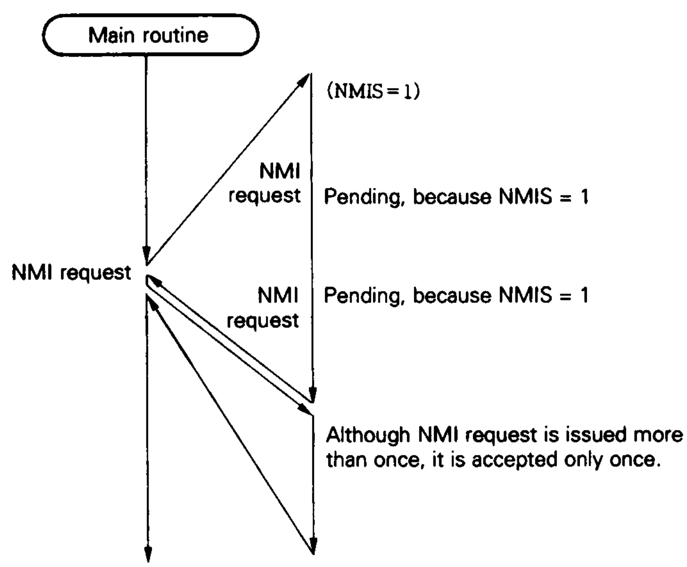
(b) If new NMI request is generated, while NMI service program is executed (when NMIS bit is reset to 0, during NMI service program execution)



(c) If new NMI request is generated, while NMI service program is executed (when NMIS bit is reset to 0 by NMI service program currently executed after NMI request is issued)



(d) If two new NMI requests are generated, while NMI service program is executed (when NMIS bit is not manipulated in NMI service program)



Caution 1: A macro service request is accepted and processed, even while a nonmaskable interrupt service program is executed. To prevent macro service processing, while a nonmaskable interrupt service program is executed, manipulate the interrupt mask register in the nonmaskable service program, so that the macro service request does not generate.

2: If the IE bit for the PSW is set to 1 by executing the EI instruction in a nonmaskable interrupt service program, the maskable interrupt request, assigned higher priority, is accepted. If a maskable interrupt, assigned higher priority, is generated, while a nonmaskable interrupt service program is executed, the service program for the maskable interrupt is executed. When the IE bit and ISP bit of PSW are set to 1, a request for the maskable interrupt assigned lower priority is generated, and the interrupt service program is executed. To exit from the service program for the maskable interrupt, the RETI instruction is used. However, this instruction resets the NMIS bit to 0. Consequently, the nonmaskable interrupt request is enabled to be accepted, even when the nonmaskable interrupt is not to be multiplexed in the nonmaskable interrupt service program to which the execution has exited from the maskable interrupt service program. To prevent multiplexed processing of the nonmaskable interrupt, do not enable interrupts, while the nonmaskable interrupt service program is executed.

3: The non-maskable interrupt is accepted anytime except while a non-maskable interrupt processing program is executed (except when multiplexing of non-maskable interrupts is enabled by clearing the NMIS bit of the IST register to 0 during non-maskable interrupt processing), and since any of the specific instructions indicated in 14.3.5 has been executed until the next instruction is executed. Therefore, the non-maskable interrupt is accepted even when the value of the stack pointer is undefined, for example, immediately after the reset function has been canceled. At this time, the contents of the program counter (PC) and program status word (PSW) are written to addresses to which writing special function register contents is disabled (see Table 3-4 in 3.2.5), depending on the value of the stack pointer, causing the CPU to be deadlocked and unexpected signals to be output from pins. When the contents of the PC and PSW have been written to addresses to which RAM is not mapped, the CPU cannot return from the non-maskable interrupt processing routine to the main routine, and consequently, overruns.

If a falling edge is input to the NMI pin (valid edge of NMI input after reset) almost at the same time as when a rising edge is input to the $\overline{\text{RESET}}$ pin, the execution branches to the non-maskable interrupt processing routine after the reset operation has been completed, without a single instruction executed. If this happens, program hang-up will occur almost certainly. To avoid these phenomena, set an initial value to the stack pointer immediately after the reset operation has ended, and make sure, by means of hardware, that the NMI signal does not go low within 10 μs + 18/f_{CLK} after the $\overline{\text{RESET}}$ signal has risen.

14.3.3 Accepting maskable interrupt

The maskable interrupt is accepted, when the interrupt request flag is set to 1 and when the mask flag corresponding to that interrupt is cleared to 0. When the interrupt is to be processed by a macro service, it is immediately accepted and the processing is performed by the macro service. The vector interrupt is accepted, when interrupts are enabled (i.e., when the IE flag is set to 1). However, while an interrupt with a higher priority is processed (while the ISP flag is reset to 0), an interrupt with a lower priority is not accepted.

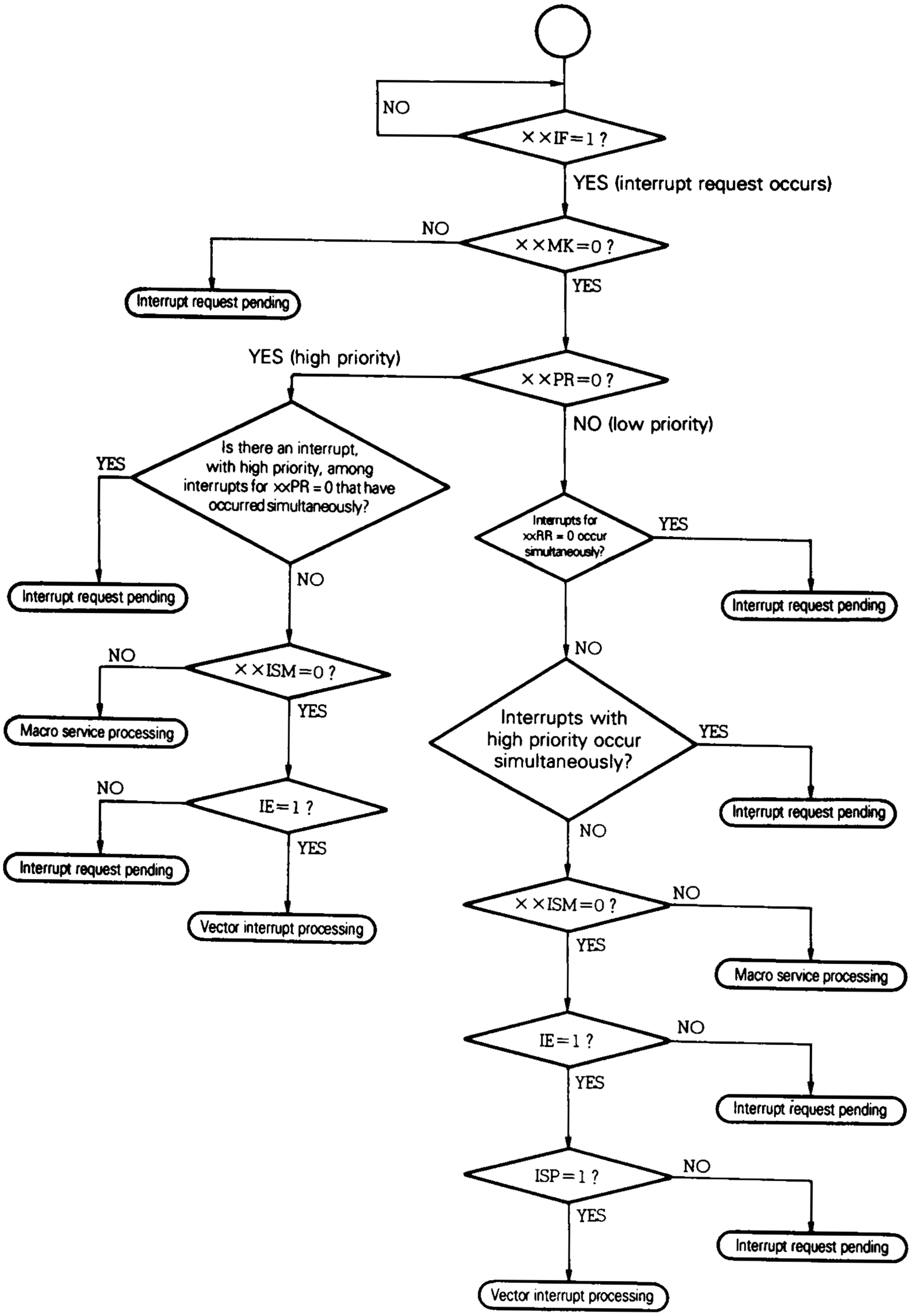
When two or more maskable interrupt requests are generated at the same time, they are accepted, starting from the one specified to have the highest priority by the priority specification flag. If all the interrupt requests have the same priority, they are accepted in accordance with the default priority.

A pending interrupt is accepted, when it is enabled.

Fig. 14-10 shows the algorithm for accepting interrupts.

When the maskable interrupt is accepted, the PSW and PC contents are saved to stack in that order, the IE flag is reset to 0 (disabling interrupts), and the contents of the priority specification flag, corresponding to the accepted interrupt, are transferred to ISP. Then data on a vector table, which is determined for each interrupt request, is loaded to the PC and execution branches. To return from the interrupt routine, use the RETI instruction.

Fig. 14-10 Interrupt Accepting/Processing Algorithm



14.3.4 Nested interrupt processing

μPD78234 can accept another interrupt request, while processing one interrupt request. These interrupts are accepted according to their priorities.

The priorities for interrupts are controlled in two modes: default priority control mode and programmable priority control mode, in which the priorities for interrupts are controlled according to the setting of the priority specification register (PR0). In default priority control mode, interrupts are processed according to the priorities assigned in advance to each interrupt request (default priorities), if two or more interrupts occur at the same time (see **Table 14-2**). In the programmable priority control mode, the PR0 register bit corresponding to an interrupt request is used to divide interrupts into two groups: a high-priority interrupt group and low-priority interrupt group. Table 14-4 shows interrupt requests that can be nested.

Table 14-4 Nested Interrupt Processing

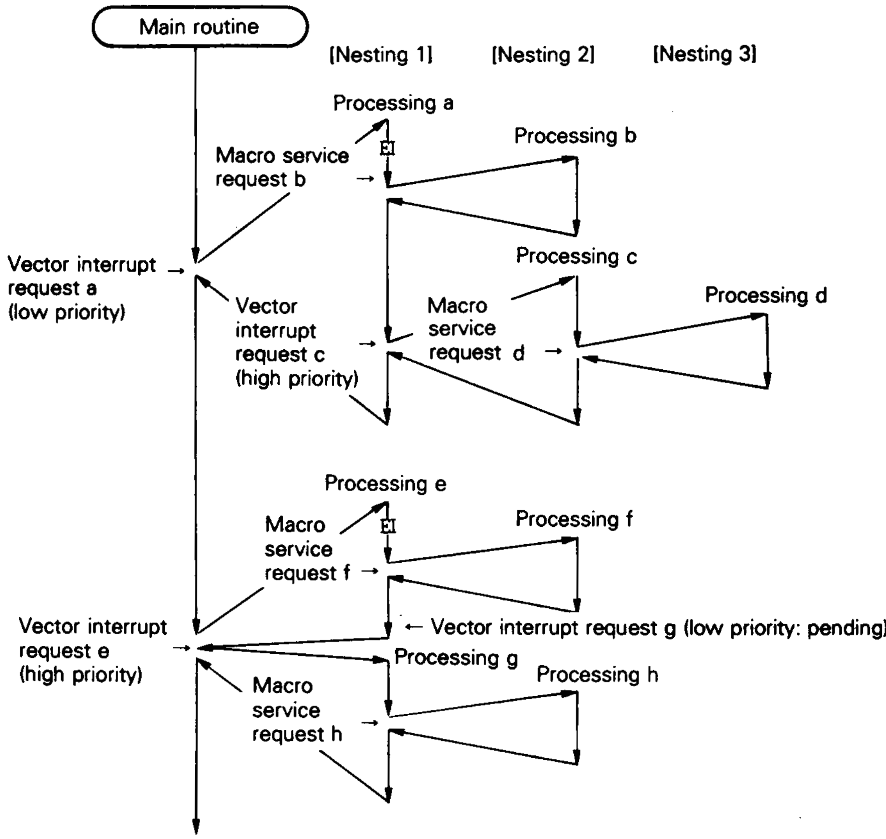
Interrupt request accepted	IE flag	Interrupt request source that can be nested
Interrupts specified low programmable priority	0	● Nonmaskable interrupt ● Maskable interrupt by macro service processing
	1[*1]	● Nonmaskable interrupt ● All maskable interrupts
Interrupts specified high programmable priority	0	● Nonmaskable interrupt ● Maskable interrupt by macro service processing
	1[*1]	● Nonmaskable interrupt ● Maskable interrupt by macro service processing ● Maskable interrupt specified high programmable priority
Nonmaskable interrupt	0	● Nonmaskable interrupt[*2] ● Maskable interrupt by macro service processing
	1[*1]	● Nonmaskable interrupt[*2] ● Maskable interrupt by macro service processing ● Maskable interrupt assigned higher priority[*3]

***1:** Immediately after an interrupt request has been accepted, interrupts are automatically disabled (IE = 0). To enable interrupts (IE = 1), execute the EI instruction.

***2:** While the nonmaskable interrupt is accepted, bit 0 (NMIS) for the interrupt status register (IST) is set to 1. While this bit is set, the nonmaskable interrupt does not occur. To enable nesting nonmaskable interrupts, clear the NMIS flag to 0 by software.

***3:** When the ISP flag is set (to 1), low level interrupt can also be accepted.

Fig. 14-11 Processing When Other Interrupts Occur While One Interrupt Is Being Processed (1/2)



Remarks: Symbols a through h in the figure are to identify interrupt requests and macro service requests.

Fig. 14-11 Processing When Other Interrupts Occur While One Interrupt Is Being Processed (2/2)

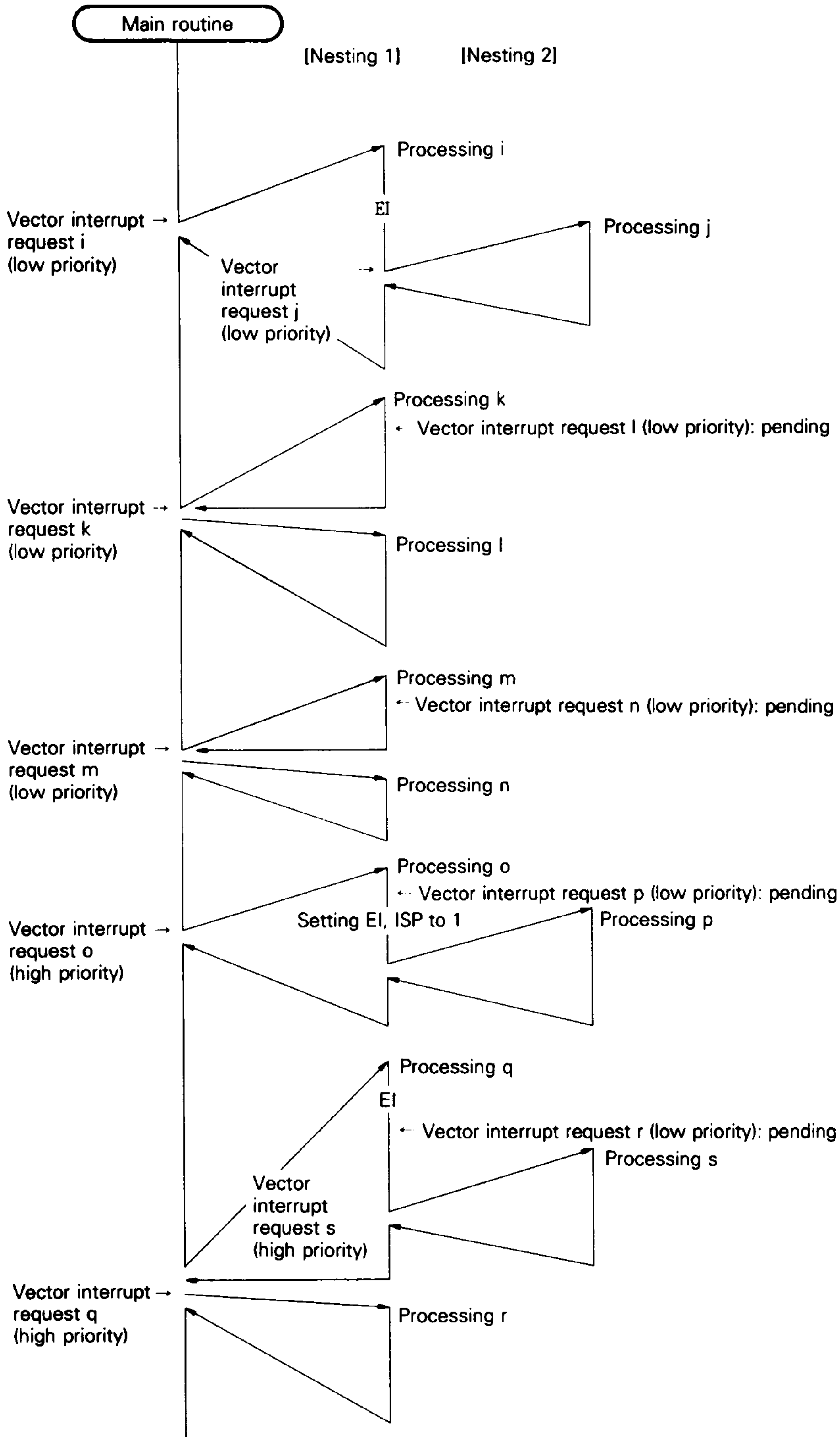
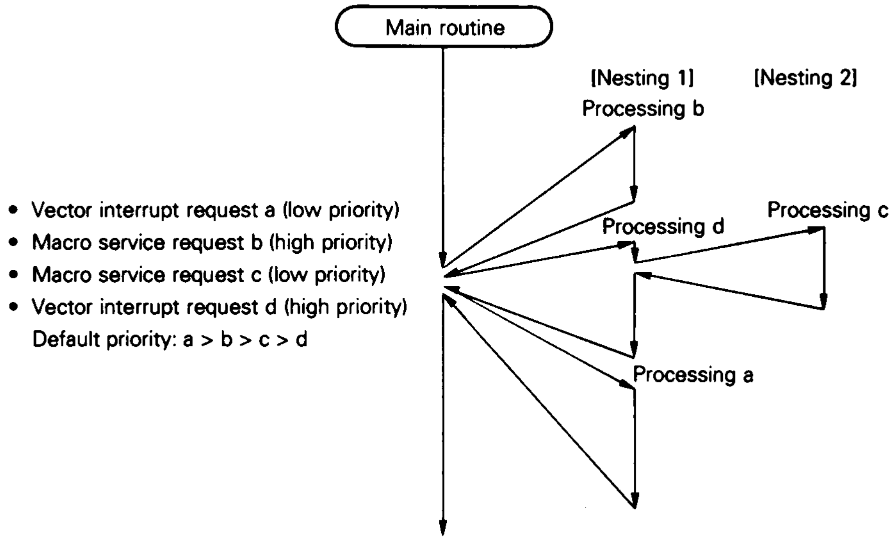


Fig. 14-12 Processing Interrupts Occurring Simultaneously



Remarks: Symbols a through d in the figure are to identify interrupt requests and macro service requests.

14.3.5 Interrupt request and macro service pending

Accepting an interrupt request and processing of a macro service are temporarily kept pending, after one of the following instructions has been executed and before the instruction next to that instruction is executed. However, the software interrupt is not kept pending.

EI
DI
RETI
RETB
POP PSW
MOV PSW, A
MOV PSW, #byte
Instructions manipulating IST, MK0, IF0, PR0, and ISM0 registers
PSW bit manipulation instructions
(except BT PSW.bit, $addr16, BF PSW.bit, $addr16, SET1 CY, NOT1 CY, and CLR1 CY instructions)

Caution 1: When polling the registers related to the interrupt function by using the BF instruction, do not specify the polling instruction as the destination for the BF instruction. If description is made so that program execution branches to the polling instruction, all interrupts and macro services are kept pending, until a condition, under which the polling instruction does not cause branching, is satisfied.

Incorrect example

```
        :
LOOP:   BF IF0H 3, $LOOP
        ×××                 ←
        :
```

All interrupts and macro services are kept pending, until IFOH.3 is set to 1. They are not processed until the instruction next to BF instruction is executed.

Correct example (1)

```
        :
LOOP:   NOP
        BF IF0H.3, $LOOP    ←
        :
```

Interrupts and macro services are not kept pending for a long time, because they are processed after NOP instruction is executed.

Correct example (2)

```
        :
LOOP:   BT IF0H.3, $NEXT
        BR $LOOP            ←
NEXT:   :
```

Use BTCLR instruction, instead of BT instruction, to automatically clear flags.
Interrupts and macro services are not kept pending for a long time, because they are processed after BR instruction is executed.

2: When the above instructions are consecutively executed, and thus interrupts and macro services are kept pending for a long time, use the NOP instruction to create timing, during which the interrupts and macro service can be accepted.

14.3.6 Interrupt and macro service operation timing

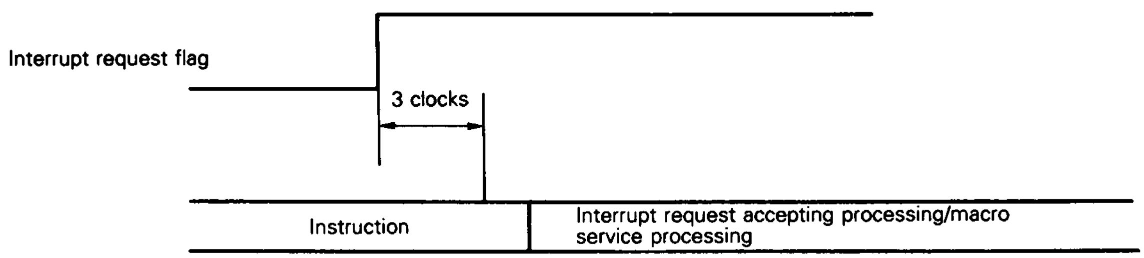
(1) Generation and accepting interrupt request

An interrupt request is generated by each hardware device. 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 for the interrupt.

If accepting the interrupt or macro service is enabled, the interrupt is accepted after the currently executed instruction has been completed. If the instruction currently executed is to keep the interrupt or macro service pending, the interrupt request is not accepted, until the instruction next to that instruction has been executed (for details regarding the instructions that keep the interrupt pending, refer to **14.3.5 Interrupt request and macro service pending**).

Fig. 14-13 Interrupt Generation and Accepting (unit: clock)



(2) Interrupt accepting processing time

To accept an interrupt, the times shown in Table 14-5 are required. Execution of the interrupt processing program is started, after the time shown in Table 14-5 has elapsed.

Table 14-5 Interrupt Acceptance Processing Time*

(unit: clock)

Stack area / Program fetch	Internal RAM	Peripheral RAM	External memory
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)

***:** The time required to execute the current instruction and to identify the priority for an interrupt is not included in the data in the above table.

Remarks: 1. "Internal ROM fetch" means that the program is fetched from the internal ROM. In this case, the IFCH bit, for the memory expansion mode register (MM), is 1. If the IFCH bit is 0, the program is fetched from an external ROM.

2. "Internal RAM" is in the area consisting of addresses 0FE00H through 0FEFFH.

3. "Peripheral RAM" is in the area consisting of addresses 0FC80H through 0FDFFH (0FB00H through 0FDFFH when using μPD78237, μPD78238 and μPD78P238).

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 macro service category, as shown in Tables 14-6 and 14-7.

Table 14-6 Macro Service Processing Time* (MSC = 8 bits)

(unit: clock)

Macro service processing category (Program fetch / Memory)			Internal ROM fetch			External ROM fetch		
			Internal ROM	Internal RAM	External memory	Internal ROM	Internal RAM	External memory
A			–	20	–	–	22	–
B			35	34	36 + w	37	36	38 + w
C	w/o ring control	Data transfer	46	44	48 + w × 2	48	46	50 + w × 2
C	w/o ring control	Automatic addition	49	47	51 + w × 2	51	49	53 + w × 2
C	w/ring control	Data transfer	51/55	49/53	53 + w × 2/ 57 + w × 2	53/57	51/55	55 + w × 2/ 59 + w × 2
C	w/ring control	Automatic addition	54/58	52/56	56 + w × 2/ 60 + w × 2	56/60	54/58	58 + w × 2/ 62 + w × 2

(w = number of wait cycles)

***:** The time required to execute the current instruction and to identify the priority for an interrupt is not included in the data in the above table.

Remarks: 1. The values shown in the internal ROM fetch column are for when the IFCH bit for the memory expansion mode register (MM) is 1. Refer to the external ROM fetch column, when the IFCH bit is 0.

2. The values in the internal RAM column are for when the internal RAMs for 0FE00H through 0FEFFH is used. If some other area is used as the internal RAM, the values shown in the external memory column apply, except w = 0.

3. The values on the right of "/" are for when the ring counter is 0. The values on the left of "/" are for when the ring counter is other than 0.

4. 1 clock = $1/f_{CLK}$ (167 ns: at 12 MHz)

Table 14-7 Macro Service Processing Time* (MSC = 16 bits)

(unit: clock)

Macro service processing category \ Program fetch / Memory			Internal ROM fetch			External ROM fetch		
			Internal ROM	Internal RAM	External memory	Internal ROM	Internal RAM	External memory
B			38	37	39 + w	40	39	41 + w
C	w/o ring control	Data transfer	49	47	51 + w × 2	51	49	53 + w × 2
		Automatic addition	52	50	54 + w × 2	54	52	56 + w × 2
	w/ring control	Data transfer	54/58	52/56	56 + w × 2/ 60 + w × 2	56/60	54/58	58 + w × 2/ 62 + w × 2
		Automatic addition	57/61	55/59	59 + w × 2/ 63 + w × 2	59/63	57/61	61 + w × 2/ 65 + w × 2

(w = number of wait cycles)

***:** The time required to execute the current instruction and to identify the priority for an interrupt is not included in the data in the above table.

Remarks:
1. The values shown in the internal ROM fetch column are for when the IFCH bit for the memory expansion mode register (MM) is 1. Refer to the external ROM fetch column, when the IFCH bit is 0.
2. The values in the internal RAM column are for when the internal RAM for 0FE00H through 0FEFFH is used. If some other area is used as the internal RAM, the values shown in the external memory column apply, except w = 0.
3. The values on the right of "/" are for when the ring counter is 0. The values on the left of "/" are for when the ring counter is other than 0.
4. 1 clock = $1/f_{CLK}$ (167 ns: at 12 MHz)

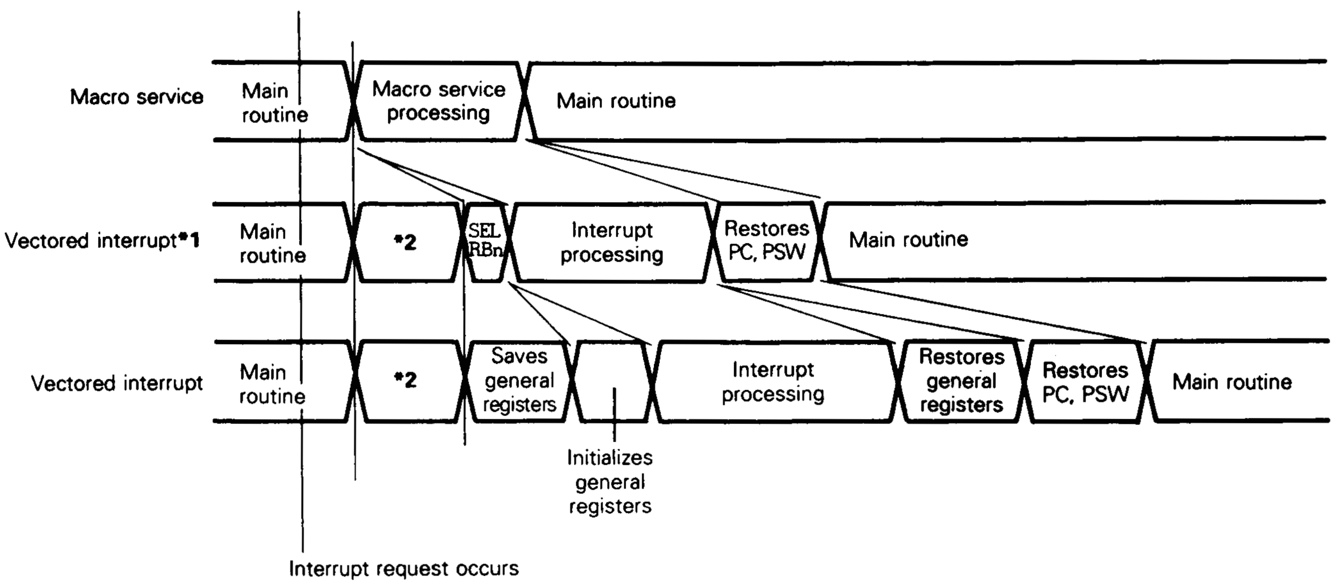
14.4 Macro Service Function

14.4.1 Macro service outline

Macro service is one of the interrupt processing methods. When an ordinary vectored interrupt is processed, the contents of the program counter (PC) and program status word (PSW) are saved to the stack and a vector address is read from the vector table and loaded to PC. With the macro service function, another processing (mainly data transfer) is implemented, instead of these processings. Therefore, a response to an interrupt request can be made faster with the macro service function than by using the vectored interrupt processing technique. In addition, data can be transferred much faster than by the program, so that the processing time can be significantly shortened.

With the macro service function, a vector interrupt can be generated, after processing has been performed the specified number of times, so that the vector interrupt program can be simplified.

Fig. 14-14 Differences between Vectored Interrupt and Macro Service



***1:** When the register bank switching function is used and when the initial values are set in the registers in advance

***2:** Saves PC and PSW to the stack and loads a vector address to PC

14.4.2 Macro service types

The macro service can be used by the 17 types of interrupts shown in Table 14-8 (of which, 15 types can be used simultaneously). In addition, three kinds of operation are available.

Table 14-8 Interrupts That Can Use Macro Service

Interrupt request generating source	Generating unit	Macro service type	Special function register when type A is used
INTP0 (pin input edge detection)	Edge detector	A, B	CR11
INTP1 (pin input edge detection)		A, B	CR22
INTP2 (pin input edge detection)		A, B	TM2
INTP3 (pin input edge detection)		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 (pin input edge detection)	Edge detector	B	–
INTC30 (TM3-CR30 coincidence signal generation)	8-bit timer/ counter 3	A, B	CR30
INTP5 (pin input edge detection)	Edge detector	B	–
INTAD (A/D conversion end)	A/D converter	A, B	ADCR
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 transfer end)		A, B	TXB
INTCSI (clock-synchronized serial interface transfer end)	Clock-synchronization serial interface	A, B	SIO

The following three macro services are available:

(1) Type A

Transfers 1-byte data between a special function register (SFR) and memory, each time an interrupt request is issued. When data has been transferred the specified number of times, a vector interrupt request is generated.

The SFR, with which data is to be transferred, is predetermined for each interrupt request. In addition, the memory is fixed to addresses 0FE00H through 0FEFFH in the internal RAM.

This macro service is easily specified and is suitable for small-capacity high-speed data transfer.

(2) Type B

Transfers 1-byte data between an SFR and memory, each time an interrupt request is issued, like Type A, and generates a vector interrupt request, after data has been transferred the specified number of times.

The SFR and memory, between which data is to be transferred, is specified by a macro service channel (the memory is a 64K-byte area for addresses 0000H through FEFFH).

This macro service is a general-purpose service Type A, and is suitable for high-capacity data transfer.

(3) Type C

Transfers 1-byte data from memory to the real-time output port and the compare register for the 8-bit timer/ counter 1, each time an interrupt request is issued. When data has been transferred the specified number of times, a vector interrupt is generated.

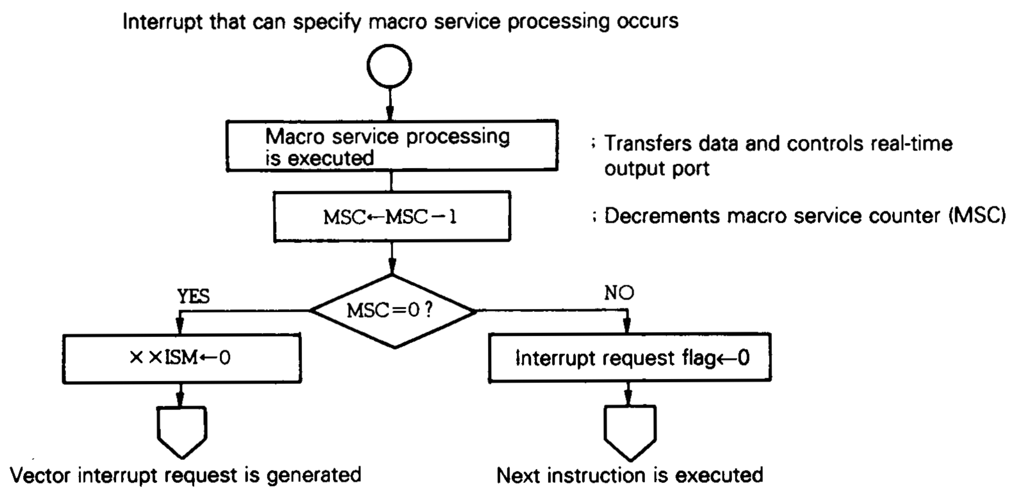
Macro service Type C transfers data to two locations, when one interrupt request is issued. In addition, it can be used for ring control of output data and for automatically adding compare register contents to data. Interrupts that can use Type C are limited to INTC10 and INTC11. In addition, the SFRs, with which data is to be transferred, are also limited. A 64K-byte memory area, addresses 0000H through FEFFH, can be used. Type C is a macro service for the real-time output port and is suitable for controlling a stepping motor.

14.4.3 Macro service basic operation

The interrupt request that can specify the macro service processing, which has been generated by the algorithm shown in Fig. 14-10, is basically processed in the sequence shown in Fig. 14-15.

An interrupt request that can specify a macro service is not affected by the IE flag status, and is disabled only by setting the interrupt mask flag for the interrupt mask register (MK0) to 1. The macro service processing can be executed, even when interrupt is disabled or while an interrupt processing program is executed.

Fig. 14-15 Macro Service Processing Sequence



The macro service and transfer direction categories are determined by the value set in the mode register for the macro service control word. The macro service channel, specified by a channel pointer, is then used in accordance with the type of macro service, and transfer processing is performed.

A macro service channel can be located in a memory, where the macro service counter, which stores the number of times transfer is to be executed, and pointer and data buffer for transfer destination and source, are located, i.e., at any of addresses FE00H through FEFFH in the internal RAM.

14.4.4 Macro service control register

(1) Macro service control word

The macro service function for μPD78234 is controlled by macro service mode registers and macro service channel pointers. The macro service mode registers set macro service processing modes, while the macro service channel pointers indicate the addresses for macro service channels.

The macro service mode registers and macro service channel pointers constitute a macro service control word, which is mapped for each macro service on a part of the internal RAM, as shown in Fig. 14-16.

When processing macro service, it is necessary to set the macro service mode register value and channel pointer corresponding to the interrupt that can specify macro service processing.

Fig. 14-16 Macro Service Control Word Configuration

0FEDFH	Channel pointer	INTSR
0FEDEH	Mode register	
0FEDDH	Channel pointer	INTST
0FEDCH	Mode register	
0FEDBH	Channel pointer	INTCSI
0FEDAH	Mode register	
0FED9H	Channel pointer	INTC10
0FED8H	Mode register	
0EFD7H	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

This 8-bit register specifies the macro service operation. It is located in the internal RAM as a part of the macro service control word (see **Fig. 14-16**).

This register format is shown in Fig. 14-17.

Fig. 14-17 Macro Service Mode Register Format

7	6	5	4	3	2	1	0
CHT2	CHT1	CHT0	B/W	MOD3	MOD2	MOD1	MOD0

	Type A	Type B	Type C: MPT is retained	Type C: MPT is incremented
B/W	0	0: 8-bit macro service counter 1: 16-bit macro service counter		
CHT0	0	1	0	1
CHT1	1	0	1	
CHT2	0	1	1	

MOD3	MOD2	MOD1	MOD0	Type A / Type B	Type C		
0	0	0	0	Data transfer from memory to SFR			
0	0	0	1	Data transfer from SFR to memory			
1	0	0	0		w/o ring control	Data transfer only	P0L
1	0	0	1		w/o ring control	Data transfer only	P0H
1	0	1	0		w/o ring control	w/ automatic addition	P0L
1	0	1	1		w/o ring control	w/ automatic addition	P0H
1	1	0	0		w/ring control	Data transfer only	P0L
1	1	0	1		w/ring control	Data transfer only	P0H
1	1	1	0		w/ring control	w/ automatic addition	P0L
1	1	1	1		w/ring control	w/ automatic addition	P0H

(3) Macro service channel pointer

The macro service channel pointer specifies the address for a macro service channel. The macro service channel can be located in a 256-byte area in the internal RAM, addresses FE00H through FEFFH. The higher 8 bits in the address are fixed; therefore, the lower 8 bits in the highest address for the macro service channel are set in the macro service channel pointer.

14.4.5 Macro service type A

(1) Operation

This macro service transfers data between the buffer memory, in the macro service channel, and a predetermined SFR, each time an interrupt request is generated.

Macro service Type A can transfer data from memory to an SFR or from an SFR to memory.

Data is transferred the number of times set in advance in the macro service counter. When macro service processing is performed once, 8 bits of data are transferred.

This macro service is useful for high-speed data transfer, where only a small quantity of data is to be transferred.

There are 12 interrupt requests that can specify macro service type A. SFRs used as the transfer source and transfer destination are specified by hardware as shown in Table 14-9.

Table 14-9 Interrupt Requests That Can Specify Macro Service Processing and SFRs (Type A)

Interrupt request specifying macro service Type A	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: If macro service type A is used, when the external memory is expanded (always with μPD78233), an illegal write access operation may be generated. The illegal write access is generated when either of following two conditions is satisfied.

(1) When data D0H-DFH is transferred from memory to SFR
(2) When data is transferred from SFR to memory and the transfer destination buffer (memory) address is 0FED0H-0FEDFH, when the macro service is executed

An illegal write access is carried out in the same manner as the ordinary memory access. In addition, a wait state is inserted according to the setting of the PW20 and PW21 bits of the memory expansion mode (MM) register. Table 14-10 shows the conditions under which the illegal write access occurs and the operations.

Table 14-10 Illegal Write Access Occurring Conditions and Operations

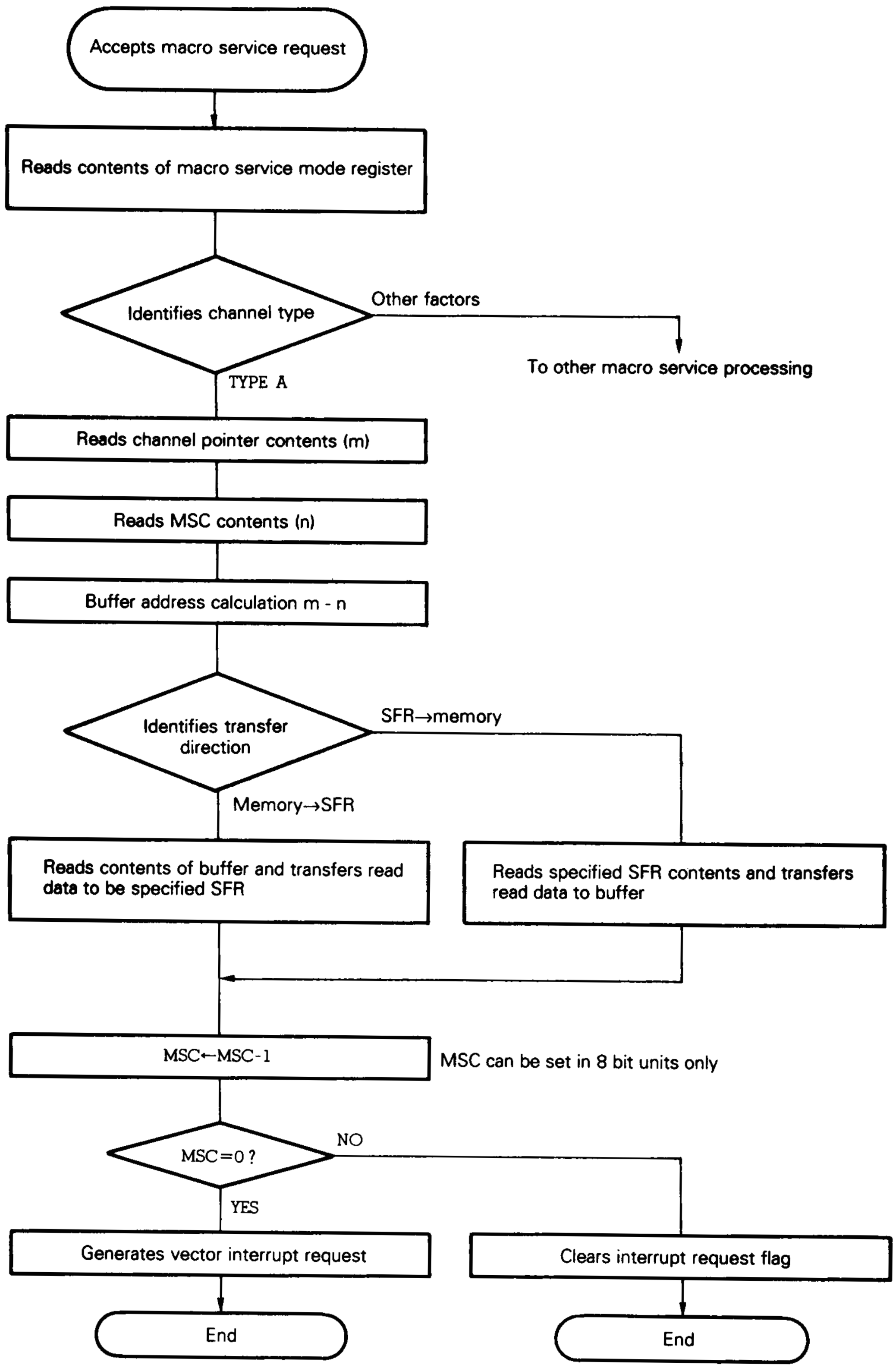
Condition	Illegal write access	
	Address	Data
1	Address of SFR of transfer destination	Data transferred by means of macro service
2	Address of SFR to be transferred	Lower 8 bits of address of transfer destination buffer (memory)

This problem can be corrected by either of these two methods:

(1) It is difficult to solve the problem that occurs under condition 1 through software (because the access is dependent on the transfer data). Therefore, use an external address decoder circuit to keep the image in the area of 0FF00H-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-0FEDFH.

The above problem also occurs with an in-circuit emulator.

Fig. 14-18 Data Transfer Processing by Macro Service (Type A)

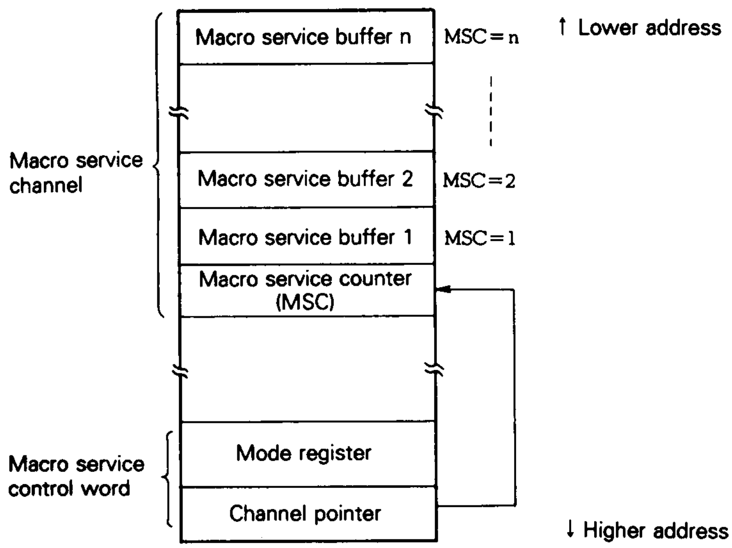


(2) Macro service channel configuration

A channel pointer and a macro service counter (MSC) specify a buffer address on the internal RAM (FE00H through FEFFH), to or from which data is transferred (see **Fig. 14-19**). The lower 8 bits for the address are written in the channel pointer.

The SFR to be accessed is predetermined for each interrupt request (refer to **Table 14-9**).

Fig. 14-19 Type A Macro Service Channel



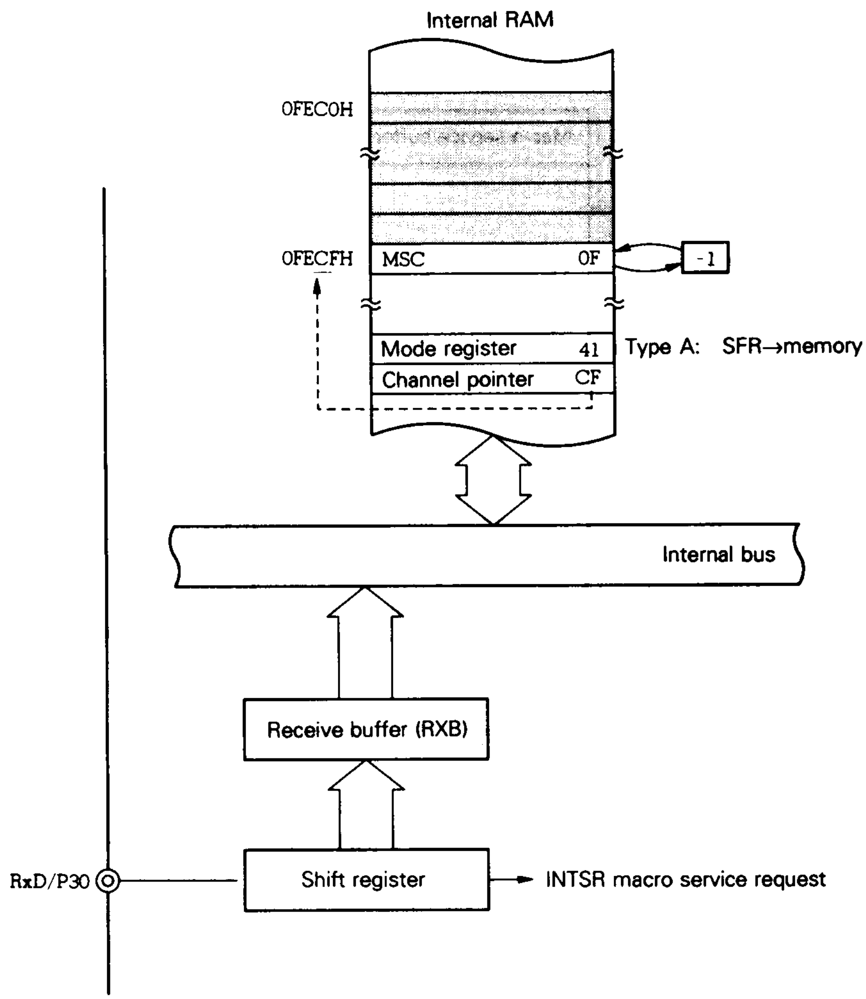
(Macro service buffer address) = (Channel pointer) – (Macro service counter)

Caution: Macro service Type A cannot specify the macro service counter (MSC) to be 16 bits long. Therefore, be sure to write “0” to the bit 4 (B/W) for the macro service mode register.

(3) Using Type A

In the following example, macro service Type A is used to transfer data received by the asynchronous serial interface to a buffer area in the internal RAM.

Fig. 14-20 Asynchronous Serial Reception



14.4.6 Macro service Type B

(1) Operation

Data is transferred between a data area in the memory, specified by the macro service channel, and an SFR. Macro service Type B can transfer data from memory to an SFR or from an SFR to memory.

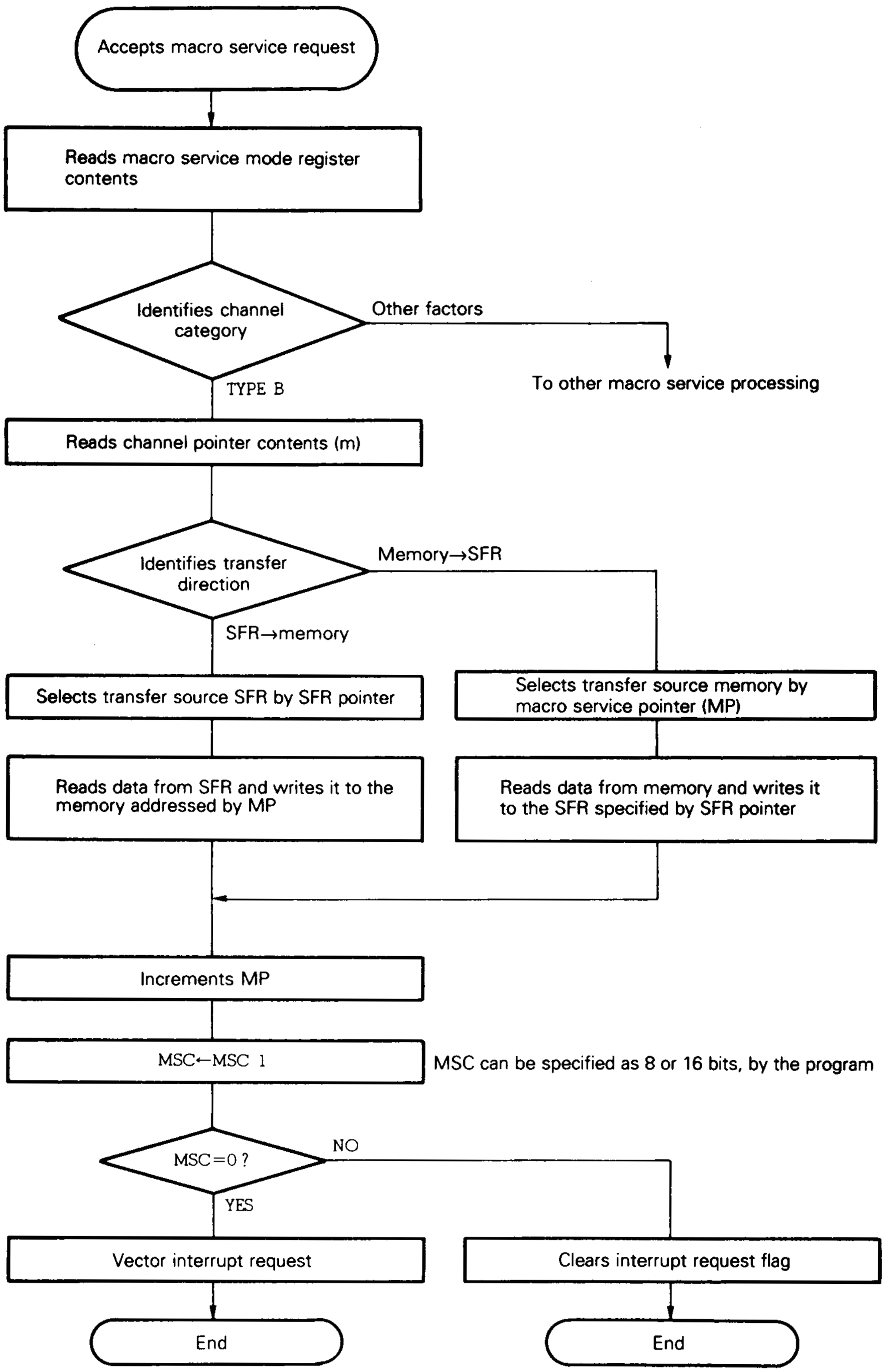
Data is transferred the number of times set in advance in the macro service counter. When macro service processing is performed once, 8 bits of data are transferred.

Macro service Type B can be specified for all the interrupt requests for μPD78234, that can start macro service.

The SFRs that serve as the transfer source and destination for type B can be freely specified by the SFR pointer.

This macro service is a type of general-purpose Type A and has a 64K-byte address area as the data buffer area. Therefore, Type B is suitable for high-capacity data processing.

Fig. 14-21 Data Transfer Processing by Macro Service (Type B)



(2) Macro service channel configuration

The macro service pointer (MP) indicates a data buffer area in the 64K memory area, to or from which data is to be transferred.

The lower 8 bits in the address for an SFR, to or from which data is to be transferred, are written to the SFR pointer (SFRP).

The macro service counter (MSC) specifies the number of times data is to be transferred, and can be specified to be 8 or 16 bits.

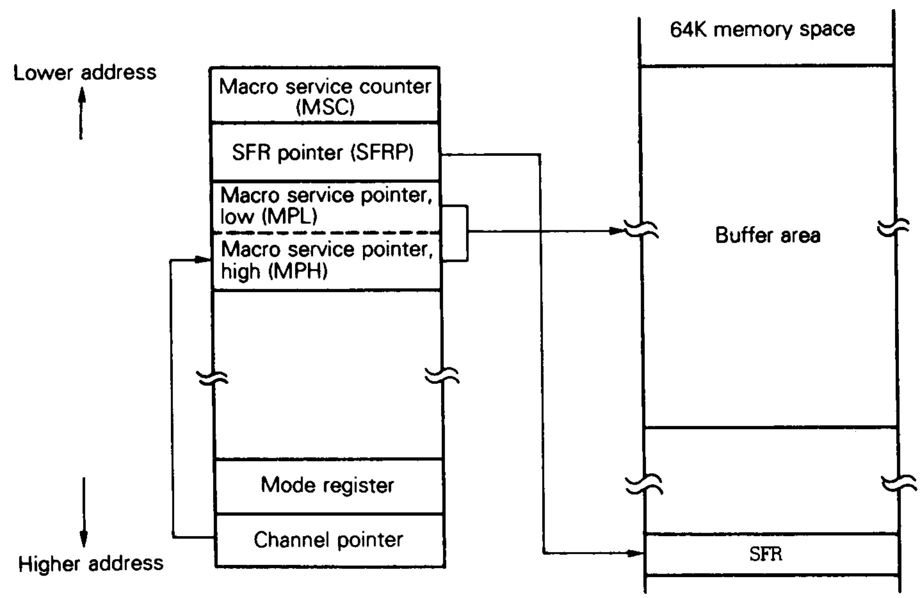
The macro service channel, that stores the macro service pointer, SFR pointer and macro service counter, is located at addresses 0FE00H through 0FEFFH in the internal RAM area.

The macro service channel is indicated by the channel pointer, as shown in Fig. 14-22. The lower 8 bits in the address for the macro service channel are written to the channel pointer.

Caution: The following SFRs cannot be used.
IF0L, IF0H, MK0L, MK0H, PR0L, PR0H, ISM0L, ISM0H, and IST

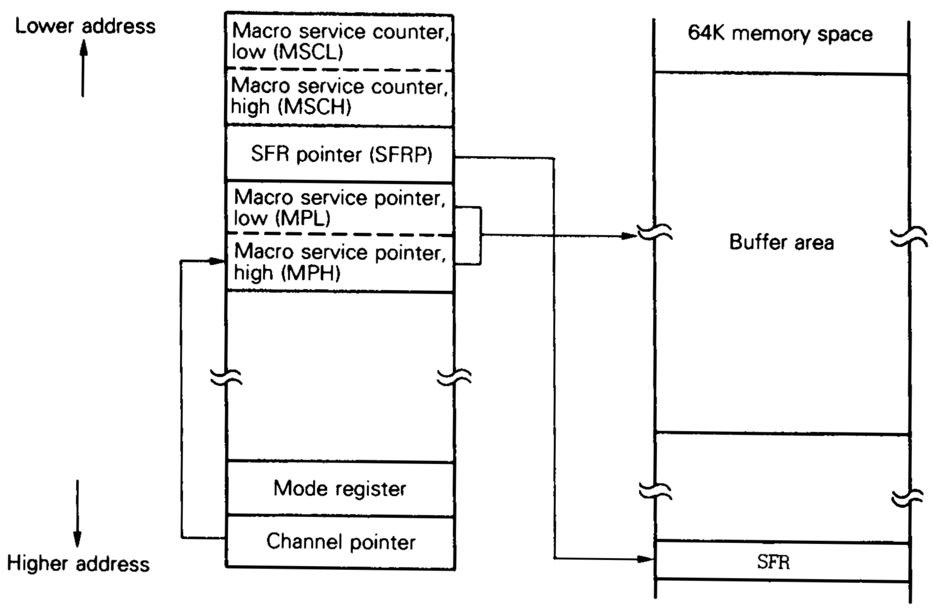
Fig. 14-22 Type B Macro Service Channel

(a) When MSC is 8 bits



(Macro service buffer address) = (macro service pointer)

(b) When MSC is 16 bits



(Macro service buffer address) = (Macro service pointer)

(3) Using type B

In the following example, parallel data is input from port 3, in synchronization with an external signal. Synchronization with the external signal is established, by using external interrupt pin INTP4.

Fig. 14-23 Inputting Parallel Data in Synchronization with External Interrupt

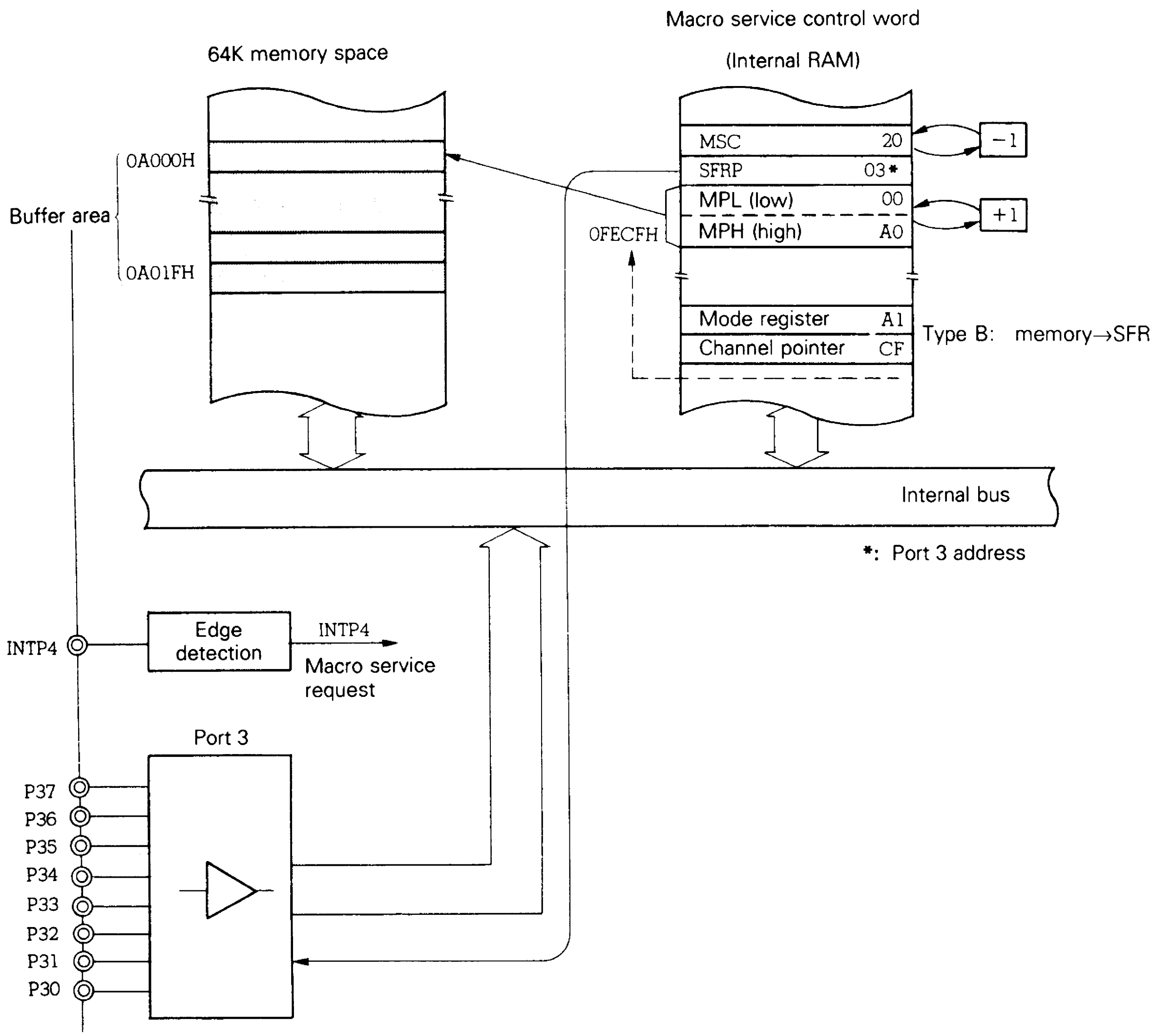
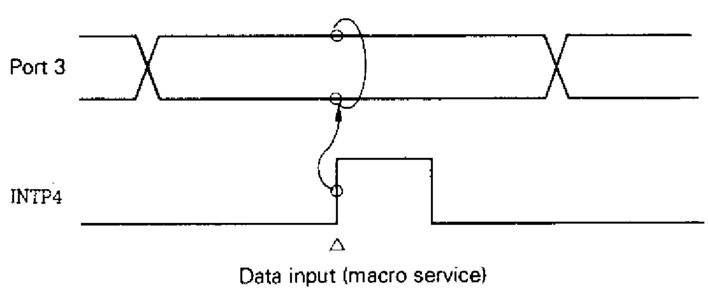


Fig. 14-24 Parallel Data Input Timing



14.4.7 Macro service Type C

(1) Operation

Type C macro service simultaneously controls 8-bit timer/counter 1 and real-time output port. This macro service transfers data to the compare register for 8-bit timer/counter 1 and the buffer register for the real-time output port, in response to one interrupt request.

Only interrupt requests INTC10 and INTC11 can specify Type C macro service. Registers, to which data is to be transferred, are predetermined, as shown in Table 14-11.

Table 14-11 Interrupt Requests That Can Specify Macro Service and SFRs (Type C)

Interrupt request	Data transfer destination addressed by MPT	Data transfer destination addressed by MPD
INTC10	CR10	POL or POH (set by mode register)
INTC11	CR11	POL or POH (set by mode register)

Type C macro service also offers the following ancillary functions, that compress the buffer area and alleviate the workload on the software, in addition to the above data transfer function:

(a) Retention of timer macro service pointer

Whether the timer macro service pointer (MPT) contents are retained or incremented can be specified.

(b) Automatic addition

The data addressed by the timer macro service pointer (MPT) is added to the current contents for the compare register. The result is transferred to the compare register.

If automatic addition is not specified, only the data addressed by MPT is transferred to the compare register.

(c) Ring control

This function is to repeatedly output a data pattern for a predetermined length, automatically.

These ancillary functions can be specified by the mode register for the macro service control word.

Caution 1: MPT and MPD are incremented in 16-bit units, even by macro service Type C. Therefore, exercise care when using the software for μPD78214 and μPD78224 sub-series (because only the lower 8 bits are incremented in these series).

2: If macro service type C is used, when the external memory is expanded (always with μPD78233), an illegal write access operation may be generated, when the following condition is satisfied:

- **When the MPTL address is 0FED0H-0FEDFH**

The illegal write access is implemented in the same manner as an ordinary memory access. In addition, a wait state is inserted according to the PW20 and PW21 bits setting for the memory expansion mode (MM) register. Table 14-12 shows the conditions, under which the illegal write access occurs and the operations.

Table 14-12 Illegal Write Access Occurring Conditions and Operations

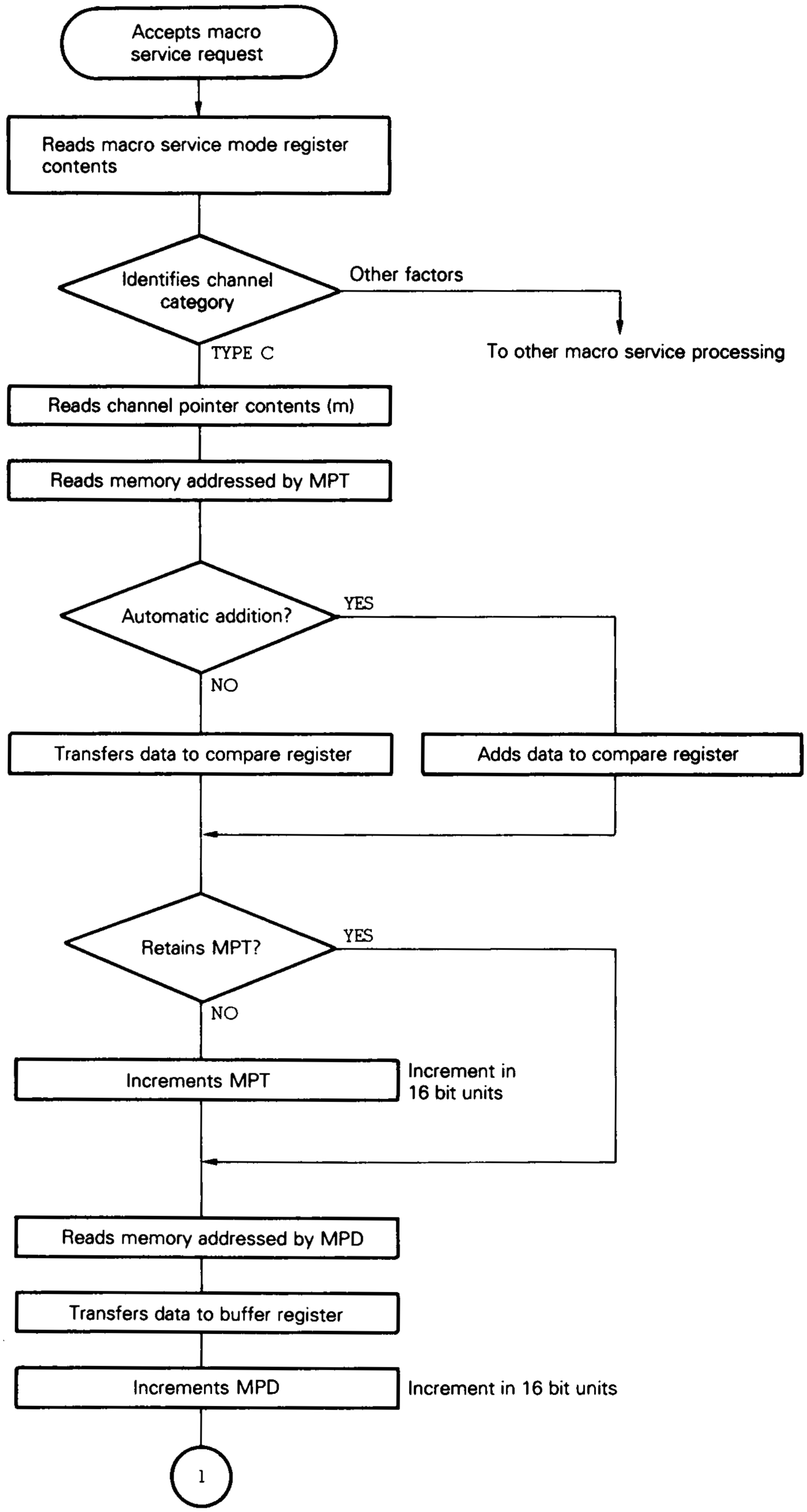
Illegal write access	
Address	Data
Address of SFR of transfer destination (CR10 or CR11)	Lower 8 bits of address of MPTL

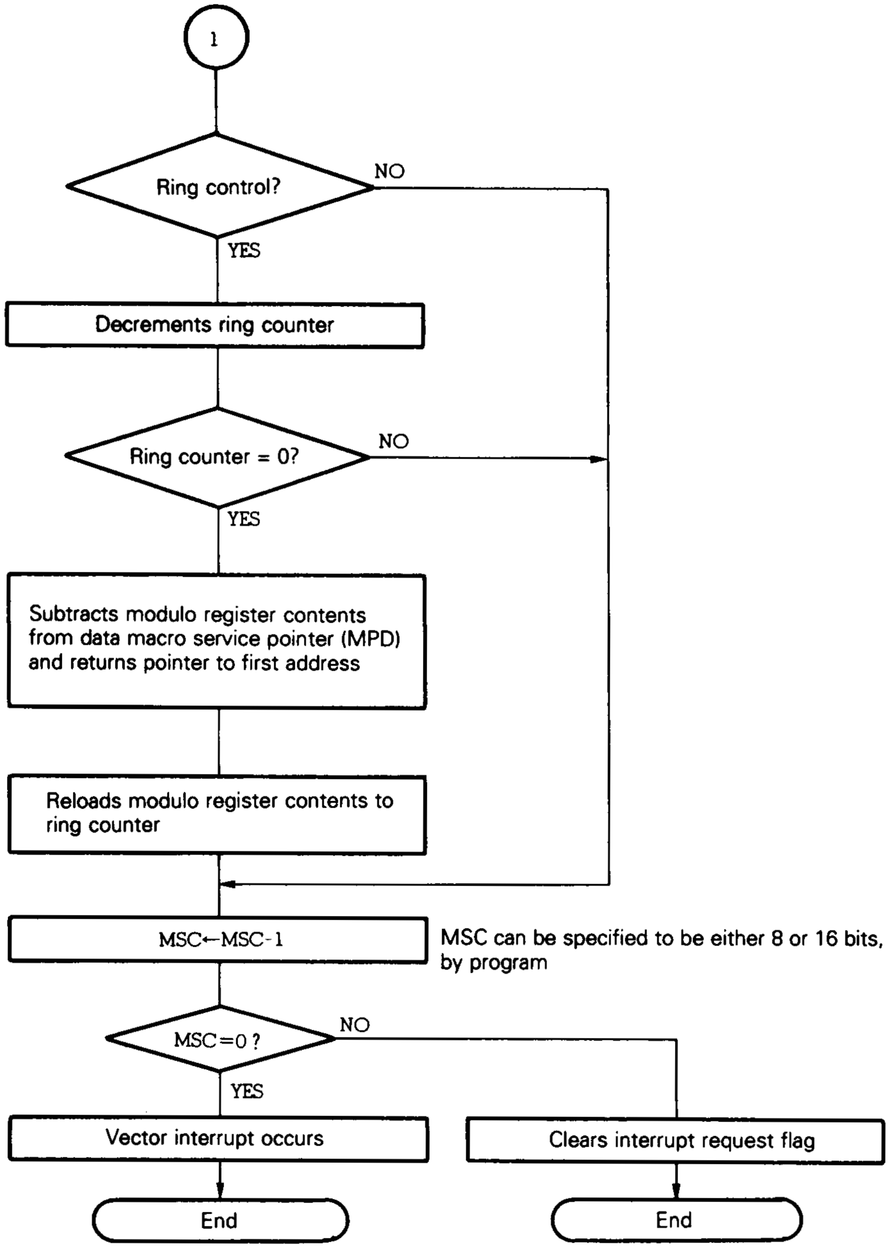
This problem can be corrected by the following method:

- **Locate the address of MPTL so that it is not 0FED0H-0FEDFH.**

The above problem also occurs with an in-circuit emulator.

Fig. 14-25 Data Transfer Processing by Macro Service (Type C)




1
Ring control?
NO
YES
Decrements ring counter
Ring counter = 0?
NO
YES
Subtracts modulo register contents from data macro service pointer (MPD) and returns pointer to first address
Reloads modulo register contents to ring counter
MSC←MSC-1
MSC can be specified to be either 8 or 16 bits, by program
MSC=0 ?
NO
YES
Vector interrupt occurs
Clears interrupt request flag
End
End

(2) Macro service channel configuration

Four macro service channel categories, as shown in Fig. 14-26, are used for Type C macro service.

Timer macro service pointer (MPT) indicates a data buffer area in the 64K memory area, from which data is transferred 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 area, from which data is transferred to the real-time output port.

The modulo register (MR) specifies the number of repetition patterns for ring control.

The ring counter (RC) retains the steps in a pattern, when ring control is used. Usually, the initial value for this counter is set to be the same value as that for the modulo register.

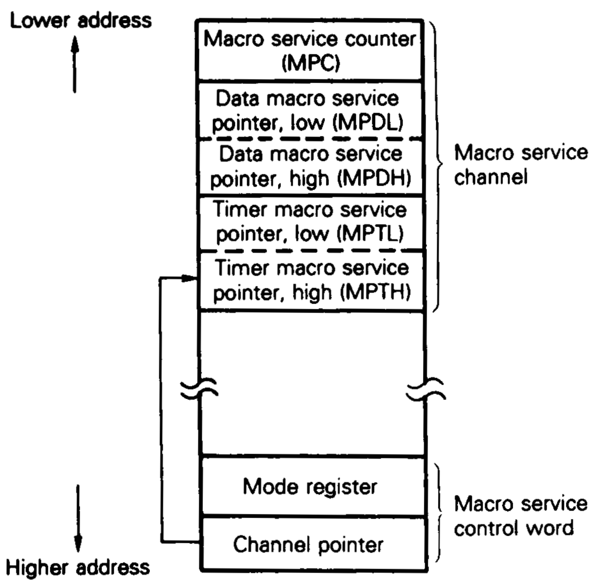
The macro service counter (MSC) specifies the number of times data is to be transferred. MSC can be 8 or 16 bits.

The macro service channel, that stores these pointers and counters, is located at addresses 0FE00H through 0FEFFH for the internal RAM area. The macro service channel is indicated by the channel pointer as shown in Fig. 14-26. The lower 8 bits in the address of the macro service channel are written to the channel pointer.

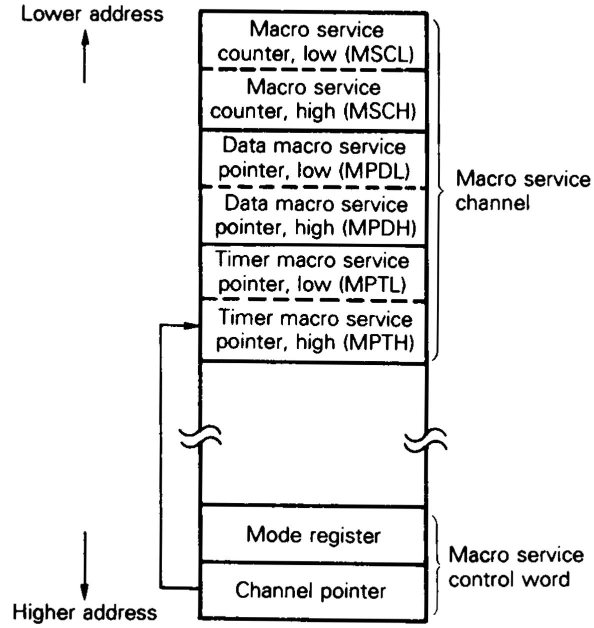
Fig. 14-26 Type C Macro Service Channel

(a) Without ring control

(i) Macro service counter, 8 bits

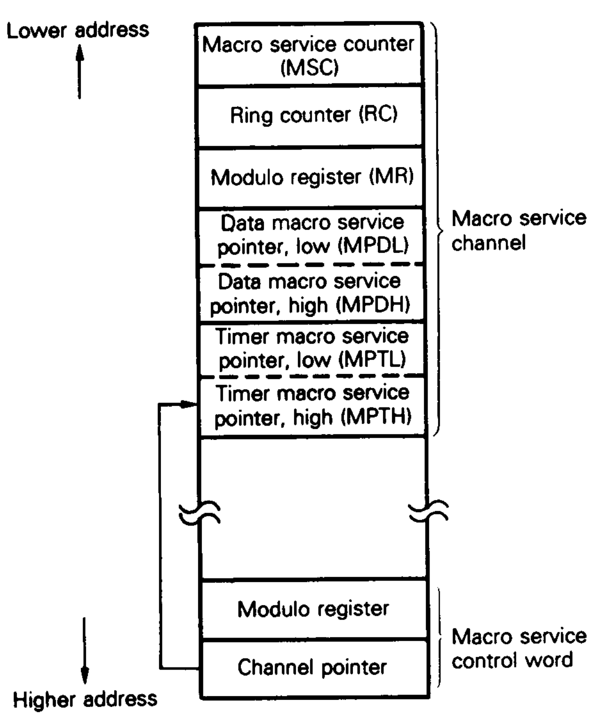


(ii) Macro service counter, 16 bits

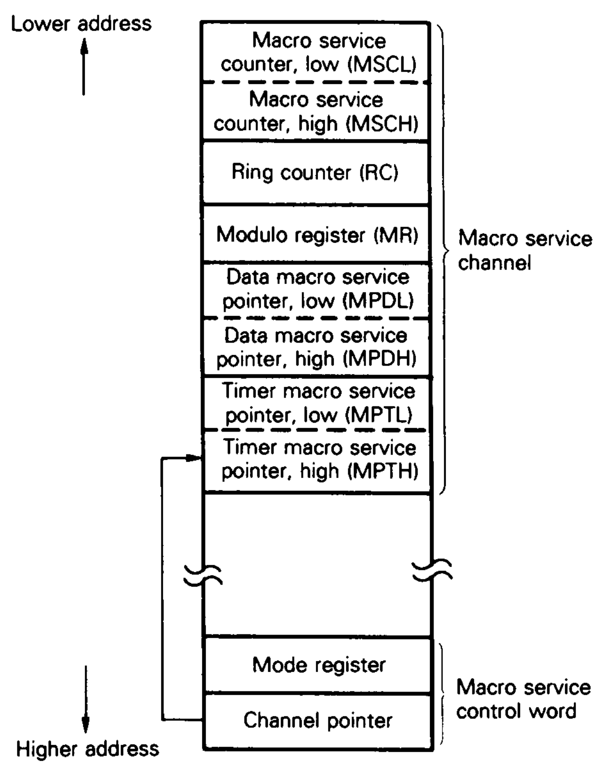


(b) With ring control

(i) Macro service counter, 8 bits



(ii) Macro service counter, 16 bits



(3) Using Type C

(a) Basic operation

In the following example, a data pattern output to the real-time output port and output intervals are directly controlled.

Updating data is transferred from two data areas set in advance 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. 14-27 Stepping Motor Open-Loop Control by Real-Time Output Port

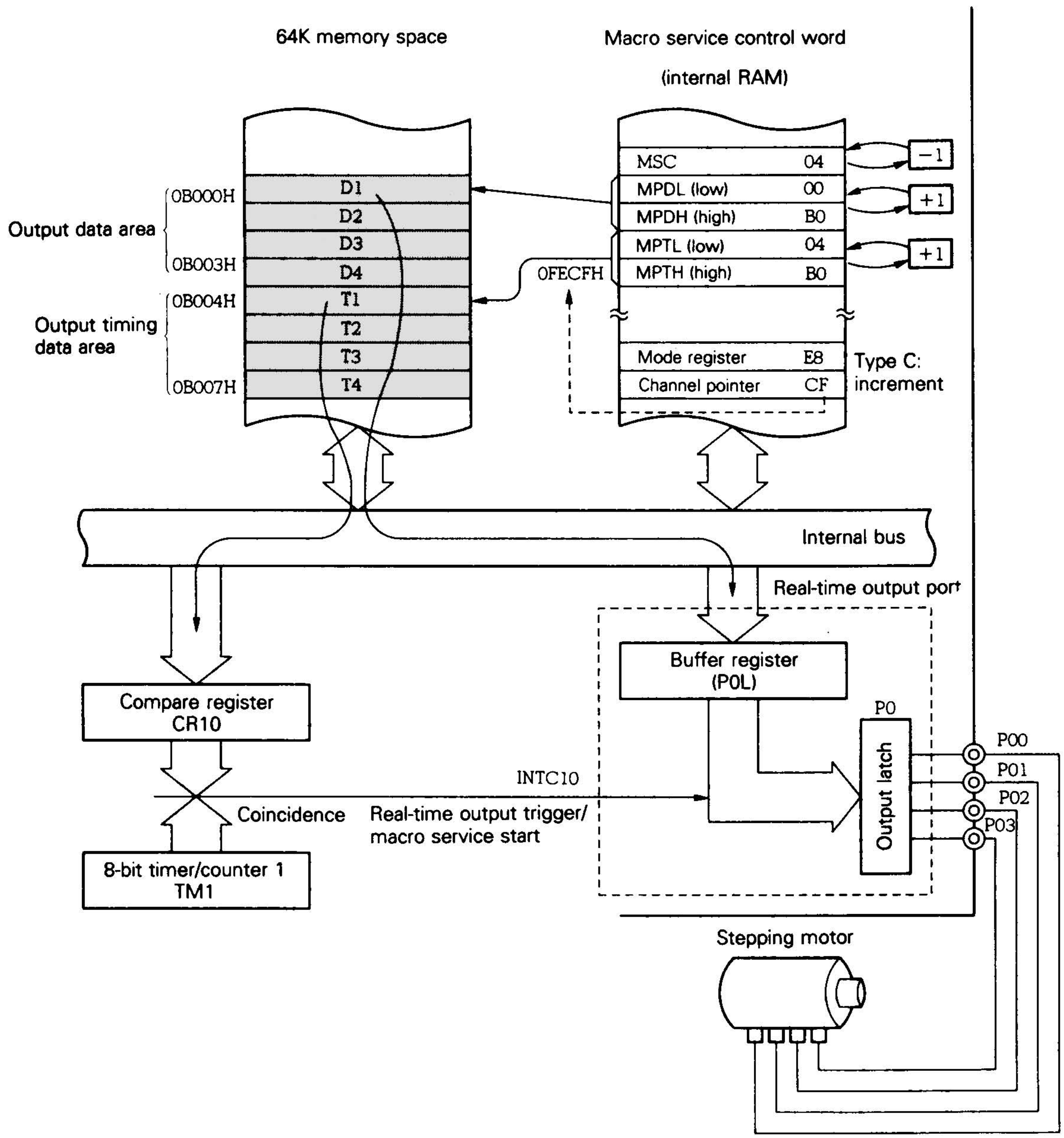
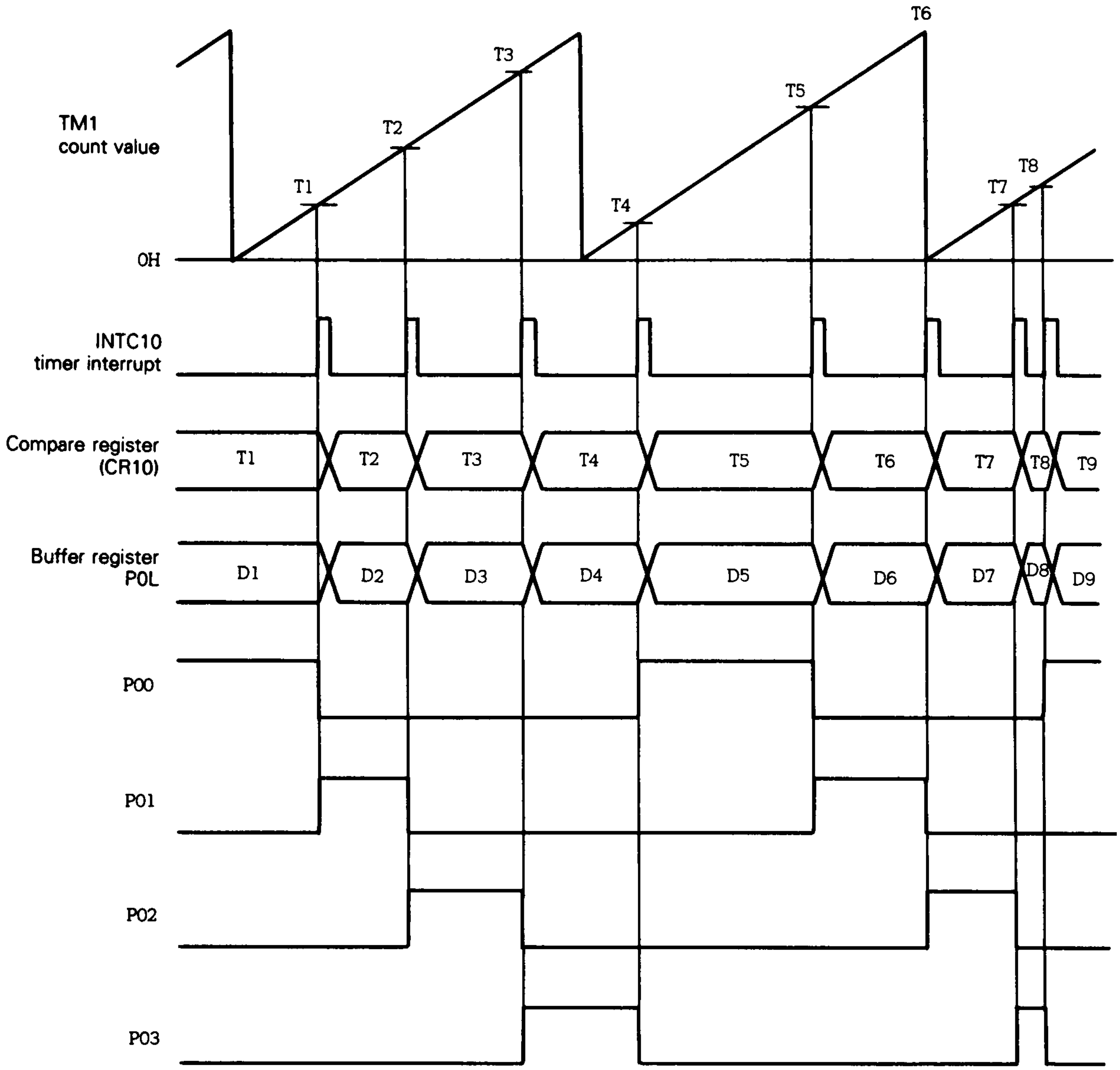


Fig. 14-28 Data Transfer Control Timing



(b) Using automatic addition control and ring control

(i) Automatic addition control

Output timing data (Δt), specified by a macro service pointer (MPT) is added to the contents of a compare register, and the addition result is written to the compare register.

By using this automatic addition control, a set value for the compare register does not have to be calculated by program each time data is set in the compare register.

(ii) Ring control

Ring control is to repeatedly output one cycle of data patterns, which is prepared in advance.

When the ring control is used, only one cycle of output data pattern must be prepared. Therefore, the necessary data ROM area can be kept small.

The macro service counter (MSC) contents are decremented each time the data has been transferred.

An interrupt request is generated, when MSC = 0, even when ring control is performed.

For example, to control a stepping motor, the output data pattern changes, depending on the configuration of the stepping motor to be controlled, and upon the excitation method for the motor, such as single-phase or double-phase excitation. However, the pattern is repeated, regardless of the motor configuration and excitation method. Figs. 14-29 and 14-30, respectively, show examples of controlling a 4-phase stepping motor with single-phase excitation and a stepping motor with 1-2 phase excitation.

Fig. 14-29 Exciting Phase 1 for 4-Phase Stepping Motor

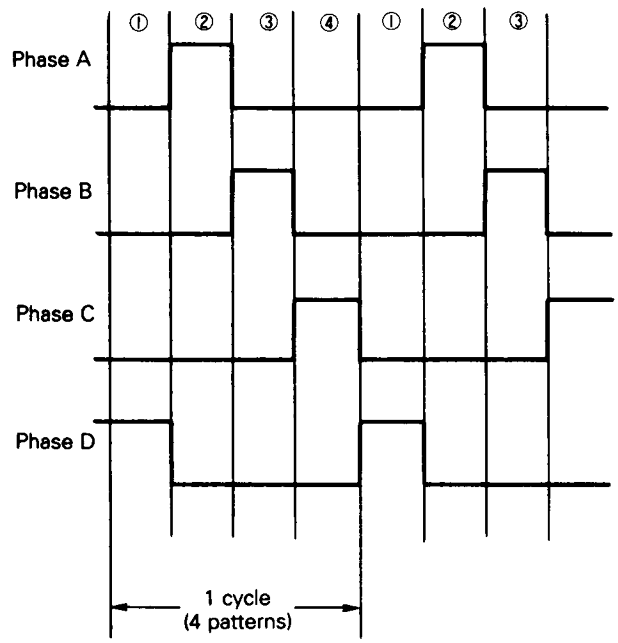


Fig. 14-30 Exciting Phases 1 and 2 for 4-Phase Stepping Motor

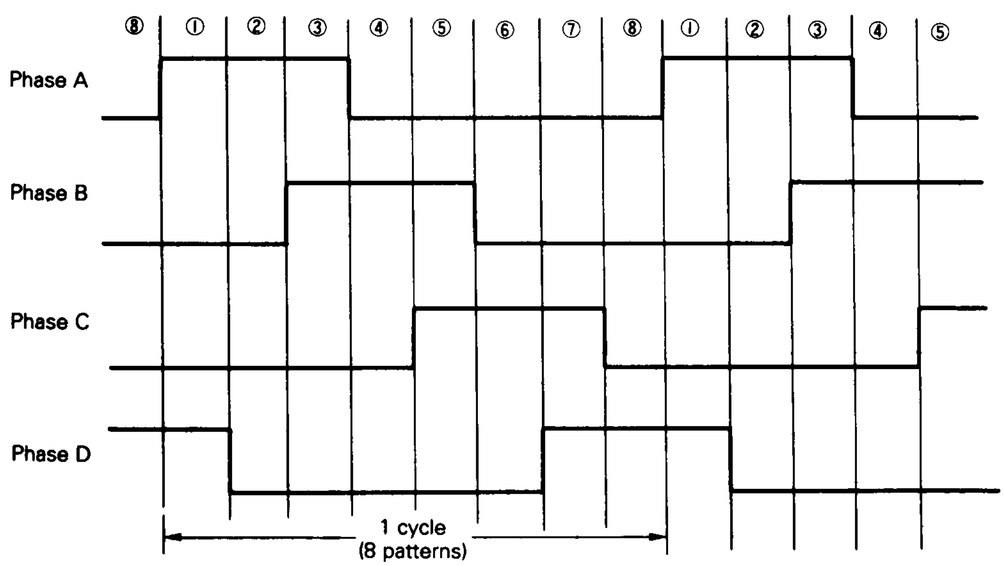


Fig. 14-31 Blockdiagram 1 for Automatic Addition Control + Ring Control (when output timing is changed by exciting phases 1 and 2: MSC = 8 bits)

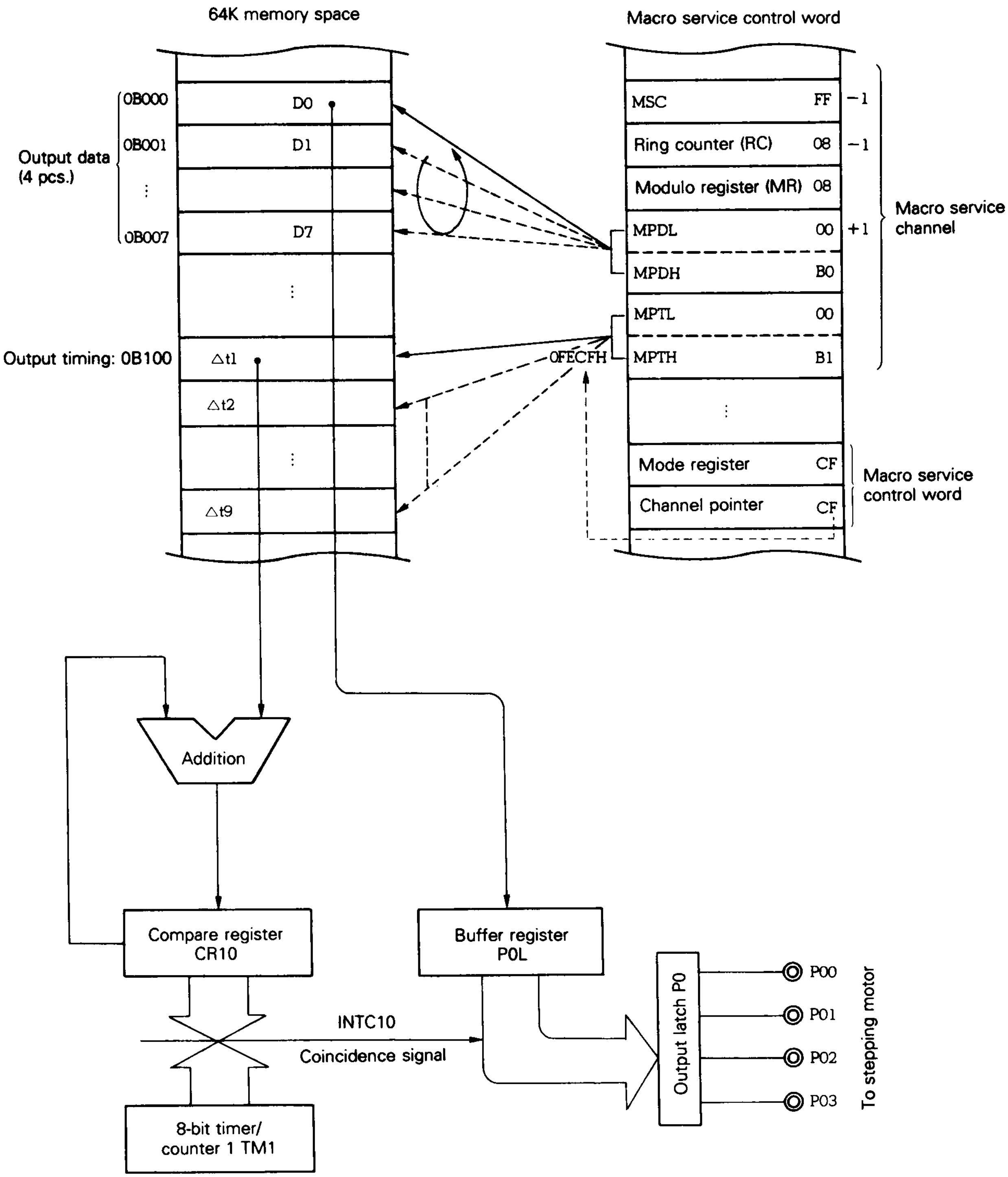
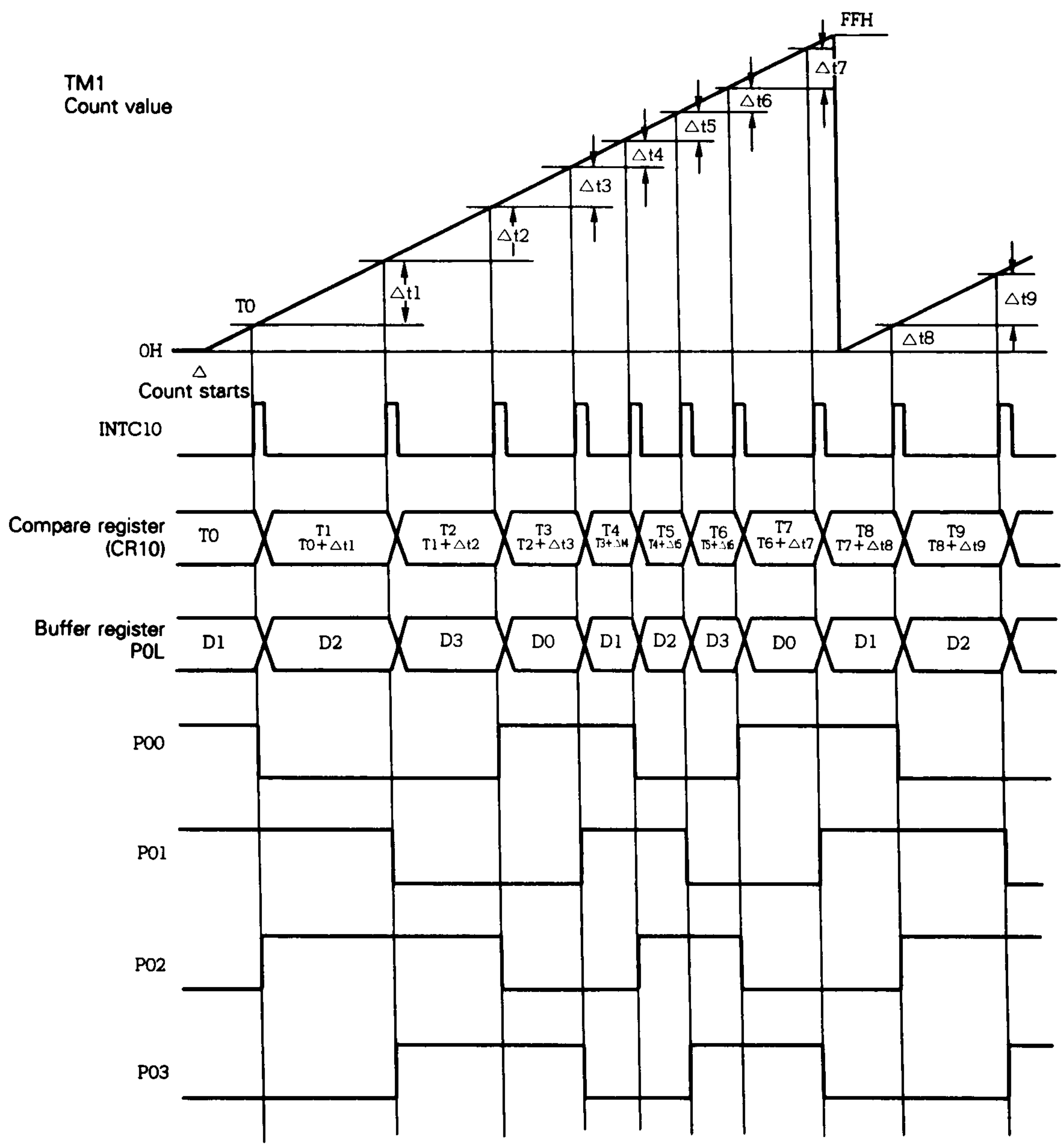


Fig. 14-32 Timing 1 for Automatic Addition Control + Ring Control (when output timing is changed by exciting phases 1 and 2)



Caution: Set a mode in which the MPT contents are incremented.

Fig. 14-33 Blockdiagram 2 for Automatic Addition Control + Ring Control (constant-speed movement, with phases 1 and 2 excited: MSC = 16 bits)

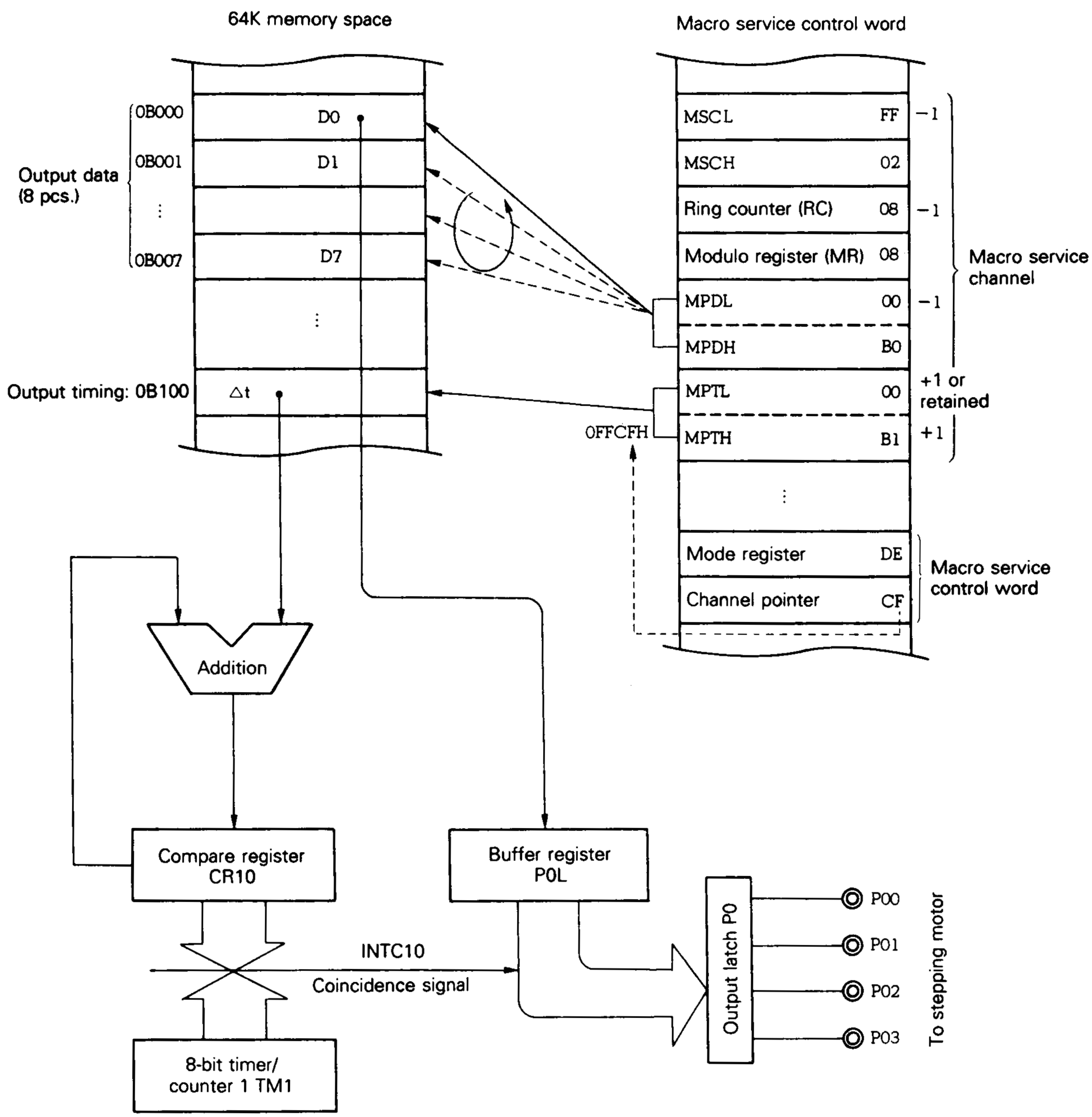
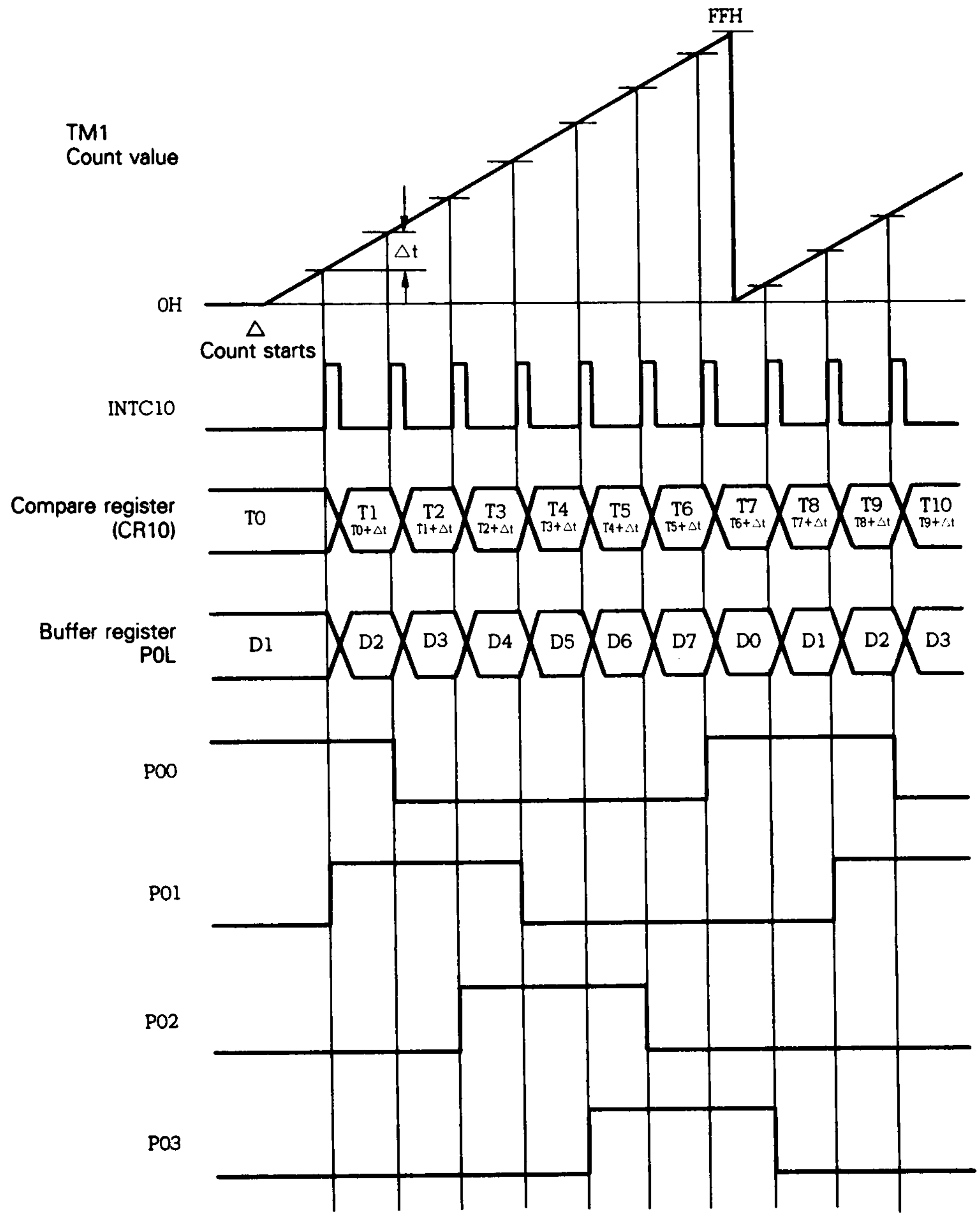


Fig. 14-34 Timing 2 for Automatic Addition Control + Ring Control (constant-speed movement with phases 1 and 2 excited)



Caution: Set a mode in which the MPT contents are retained.

14.5 Notes

(1) Do not use the RETI instruction to return from the software interrupt.

(2) A macro service request is accepted and processed, even while the nonmaskable interrupt service program is executed. To not perform the macro service processing while the nonmaskable interrupt service program is executed, manipulate the interrupt mask register in the nonmaskable interrupt service program, so that the macro service is not generated.

(3) If the IE bit for the PSW is set to 1 by executing the EI instruction in a nonmaskable interrupt service program, the maskable interrupt request, assigned higher priority, is accepted. If a maskable interrupt, assigned higher priority, is generated, while a nonmaskable interrupt service program is executed, the service program for the maskable interrupt is executed. When the IE bit and ISP bit of PSW are set to 1, a request for the maskable interrupt assigned lower priority is generated, and the interrupt service program is executed. To exit from the service program for the maskable interrupt, the RETI instruction is used. However, this instruction resets the NMIS bit to 0. Consequently, the nonmaskable interrupt request is enabled to be accepted, even when the nonmaskable interrupt is not to be multiplexed in the nonmaskable interrupt service program to which the execution has exited from the maskable interrupt service program . To prevent multiplexed processing of the nonmaskable interrupt, do not enable interrupts, while the nonmaskable interrupt service program is executed.

(4) The non-maskable interrupt is accepted anytime except while a non-maskable interrupt processing program is executed (except when multiplexing of non-maskable interrupts is enabled by clearing the NMIS bit of the IST register to 0 during non-maskable interrupt processing), and since any of the specific instructions indicated in **14.3.5** has been executed until the next instruction is executed. Therefore, the non-maskable interrupt is accepted even when the value of the stack pointer is undefined, for example, immediately after the reset function has been canceled. At this time, the contents of the program counter (PC) and program status word (PSW) are written to addresses to which writing special function register contents is disabled (see **Table 3-4** in **3.2.5**), depending on the value of the stack pointer, causing the CPU to be deadlocked and unexpected signals to be output from pins. When the contents of the PC and PSW have been written to addresses to which RAM is not mapped, the CPU cannot return from the non-maskable interrupt processing routine to the main routine, and consequently, overruns.
If a falling edge is input to the NMI pin (valid edge of NMI input after reset) almost at the same time as when a rising edge is input to the $\overline{\text{RESET}}$ pin, the execution branches to the non-maskable interrupt processing routine after the reset operation has been completed, without a single instruction executed. If this happens, program hang-up will occur almost certainly. To avoid these phenomena, set an initial value to the stack pointer immediately after the reset operation has ended, and make sure, by means of hardware, that the NMI signal does not go low within 10 μs + 18/f_{CLK} after the $\overline{\text{RESET}}$ signal has risen.

(5) When polling interrupt-related registers, using the BF instruction, do not specify the BF instruction address as the branch destination. If the program is described, so that the execution branches to the BF instruction address, all interrupts and macro services are kept pending, until a condition under which the branching is not caused by the instruction is satisfied.

Incorrect example

```
        :
LOOP:   BF IF0H.3,$LOOP
        ×××              ←
        :
```

All interrupts and macro services are kept pending, until IF0H.3 is set to 1.
Interrupt and macro service are not executed, until the instruction next to .BF is executed

Correct example (1)

```
        :
LOOP:   NOP
        BF IF0H.3,$LOOP  ←
        :
```

Interrupt and macro service are processed, after NOP instruction has been executed, so that interrupt is not kept pending for a long time

Correct example (2)

```
        :
LOOP:   BT IF0H.3,$NEXT
        BR $LOOP         ←
NEXT:   :
```

Using BTCLR instruction instead of BT instruction automatically clears flags, which is convenient.
Interrupt is not kept pending for a long time, because interrupt and macro service are processed after BR instruction has been executed

(6) For the same reason as in (5), when using the group of instructions corresponding to the **14.3.5 Interrupt request and macro service pending**, and if the interrupt and macro service is kept pending for a long time, insert NOP instructions to create the timing at which the interrupt and macro service are accepted.

(7) MPT and MPD are incremented in 16-bit units, even when macro service Type C is used. (However, only the lower 8 bits are incremented with μPD78214 or μPD78224 series). Therefore, exercise care when the software for μPD78214 or μPD78224 sub-series is used.

(8) With macro service type A, the macro service counter (MSC) cannot be specified to be 16 bits. Therefore, be sure to write 0 to bit 4 (B/W) for the macro service register.

(9) If macro service type A or C is used, when the external memory is expanded (always with μPD78233), an illegal write access operation may be generated, when any of the following three conditions is satisfied:

Condition 1. When data D0H-DFH is transferred from memory to SFR by means of macro service A
2. When data is transferred from SFR to memory and the address for the transfer destination buffer (memory) is 0FED0H-0FEDFH, when macro service type A is executed
3. When the MPTL address is 0FED0H-0FEDFH, when macro service type C is executed

An illegal write access is carried out in the same manner as the ordinary memory access. In addition, a wait state is inserted, according to the PW20 and PW21 bits setting for the memory expansion mode (MM) register. Table 14-13 shows the conditions under which the illegal write access occurs and the operations.

Table 14-13 Illegal Write Access Occurring Conditions and Operations

Condition	Macro service type	Illegal write access	
		Address	Data
1	A	Address of SFR of transfer destination	Data transferred by macro service
2	A	SFR address to be transferred	Lower 8 bits in transfer destination buffer (memory) address
3	C	SFR address (CR10 or CR11) for transfer destination	Lower 8 bits of MPTL address

This problem can be corrected by either of the following two methods:

1. It is difficult to solve a problem that occurs under condition 1 through software (because the access is dependent on the transfer data). Therefore, use an external address decoder circuit to keep the image in the 0FF00H-0FFFFH area from overlapping the external circuit memory addresses.

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 by means of macro service A), and under condition 2, locate the buffer area so that its addresses are not 0FED0H-0FEDFH. Under condition 3, locate the MPTL address, so that is not 0FED0H-0FEDFH.

The above problem also occurs with an in-circuit emulator.

(10) The following SFRs cannot be used with macro service type B:
IF0L, IF0H, MK0L, MK0H, PR0L, PR0H, ISM0L, ISM0H, IST

CHAPTER 15 LOCAL BUS INTERFACE FUNCTION

The local bus interface function is 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 through a multiplexed address/data bus consisting of AD0 through AD7 pins and an address bus consisting of A8 through A19 pins.

Figs. 15-4 and 15-5 shows the basic bus interface timing.

In addition, the wait function, to interface with a low-speed memory, and the refresh signal output function, to refresh the pseudo static RAM, are provided.

15.1 Control Registers

15.1.1 Memory expansion mode register (MM)

This 8-bit register controls an external expansion memory, specifies the number of wait states (address area: 00000H through 0FFFFH), and controls the internal fetch cycle.

This register can be read or written by 8-bit manipulation and bit manipulation instructions. Fig. 15-1 shows the format.

When the $\overline{\text{RESET}}$ signal has been input, the contents of this register are initialized to 20H.

Fig. 15-1 Memory Expansion Mode Register (MM) Format

	7	6	5	4	3	2	1	0
MM	IFCH	MM6	PW21	PW20	0	MM2	MM1	MM0

<table>
<tr><th>MM2</th><th>MM1</th><th>MM0</th><th>Mode</th><th>P50-P57</th><th>P40-P47</th><th>P65</th><th>P64</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td rowspan="2">Single-chip mode</td><td rowspan="2">Port mode</td><td>Input Port</td><td colspan="2" rowspan="2">Port mode</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Output port</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>External memory expansion mode</td><td>A8-A15</td><td>AD0-AD7</td><td>$\overline{\text{WR}}$</td><td>$\overline{\text{RD}}$</td></tr>
</table>

PW21	PW20	Number of wait states (range: 00000H-0FFFFH)
0	0	0
0	1	1
1	0	2
1	1	Number of wait states equivalent to $\overline{\text{WAIT}}$ pin low level period

MM6	Specifies 1M-byte expansion mode
0	P60-P63 as general-purpose output port
1	Output latch for P60-P63 (P60-P63) stores higher address (A16-A19) in external memory expansion mode and P60-P63 function as A16-A19 output pins

IFCH	Controls internal fetch cycle
0	Instruction execution cycle, the same as the external ROM fetch cycle
1	High-speed internal ROM fetch (execution cycle is faster than that for external ROM fetch)

15.1.2 Programmable wait control register (PW)

This 8-bit register specifies the number of wait states in the external expansion data memory area 10000H through FFFFFH. This register can be read or written by 8-bit manipulation and bit manipulation instruction. The format is shown in Fig. 15-2.

When the $\overline{\text{RESET}}$ signal has been input, the contents of this register are initialized to 80H.

Fig. 15-2 Programmable Wait Control Register (PW) Format

	7	6	5	4	3	2	1	0
PW	PW31	PW30	0	0	0	0	0	0

PW31	PW30	Number of wait states (range: 10000H-FFFFFH)
0	0	0
0	1	1
1	0	2
1	1	Number of wait states equivalent to $\overline{\text{WAIT}}$ pin low level period

15.1.3 Memory size select register (IMS)

The IMS register selects the μPD78P238 internal memory, so that the microcomputer operates in the same manner as μPD78234 or μPD78238.

To select the same memory size as that for μPD78234, be sure to write this register immediately after the system has been reset.

The written values must not be changed.

The IMS register can be written only by an 8-bit manipulation instruction. The format is shown in Fig. 15-3.

When the $\overline{\text{RESET}}$ signal has been input, the contents of this register are initialized to FFH, setting the memory mapping the same as that for μPD78238.

Fig. 15-3 Memory Size Select Register

	7	6	5	4	3	2	1	0
IMS	IMS7	IMS6	IMS5	IMS4	IMS3	IMS2	IMS1	IMS0

IMS0-7	Memory size
FFH	Same as μPD78238
DDH	Same as μPD78234

This register is not provided to μPD78234 and μPD78238. However, even if μPD78234 or μPD78238 executes the instruction that writes data to this register, the operations of μPD78234 or μPD78238 are not affected.

15.2 Memory Expansion Function

15.2.1 External memory expansion function

μPD78234 can expand the external memory to 48256 bytes (to 31488 bytes in the case of μPD78238) and external I/Os by the memory expansion mode register (MM).

To expand the external memory space, the P50 through P57 pins constitute an address bus, while the P40 through P47 pins form a multiplexed address/data bus.

μPD78214 can be set in the ROM-less mode by making the MODE pin high. In this case, a 64640-byte (with μPD78238, 64256-byte) external memory and I/Os can be connected to the microcomputer regardless of the setting of the MM register (the MODE pin of μPD78P238 cannot be made high).

μPD78233 can always uses the external memory, because it is a ROM-less product.

Caution: The address data output to P50/A8-P57/A15, P60/A16-P63/A19 is effective only from the rising edge of the ASTB signal to the rising edge of the $\overline{RD}$ signal or the $\overline{WR}$ signal. The output levels of P50/A8-P57/A15, P60/A16-P67/A19 are undefined during other periods. Circuits must be designed in a manner so that the output of an undefined value will cause no problems. Refer to the data sheet of each product for the speicification concering the effective period for address output.

Fig. 15-4 Read Timing

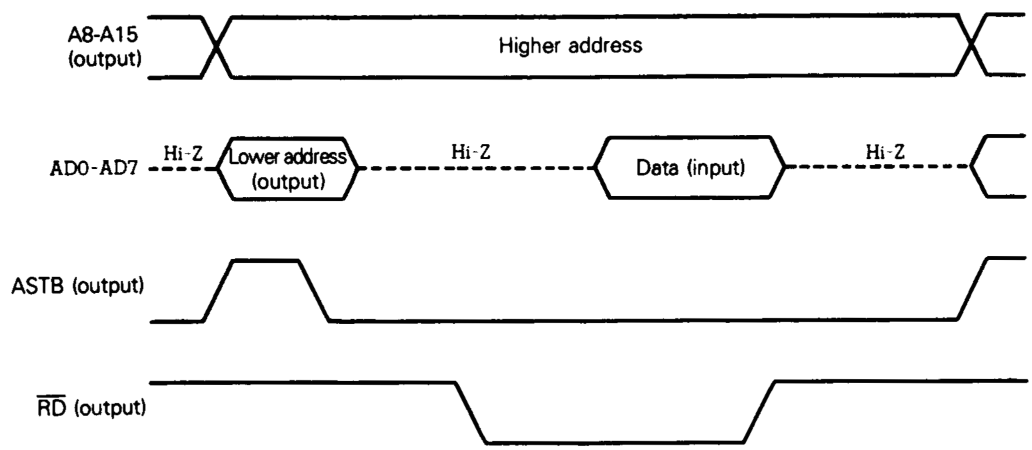
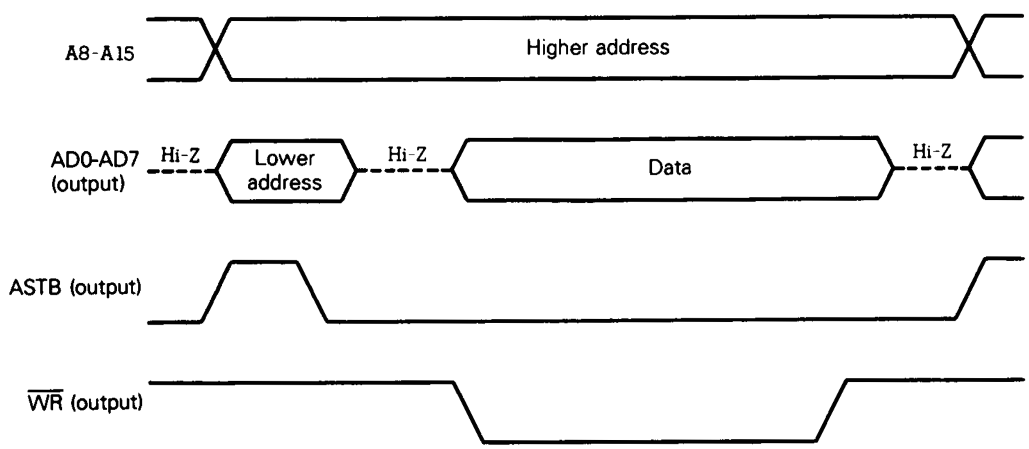


Fig. 15-5 Write Timing



15.2.2 1M-byte expansion function

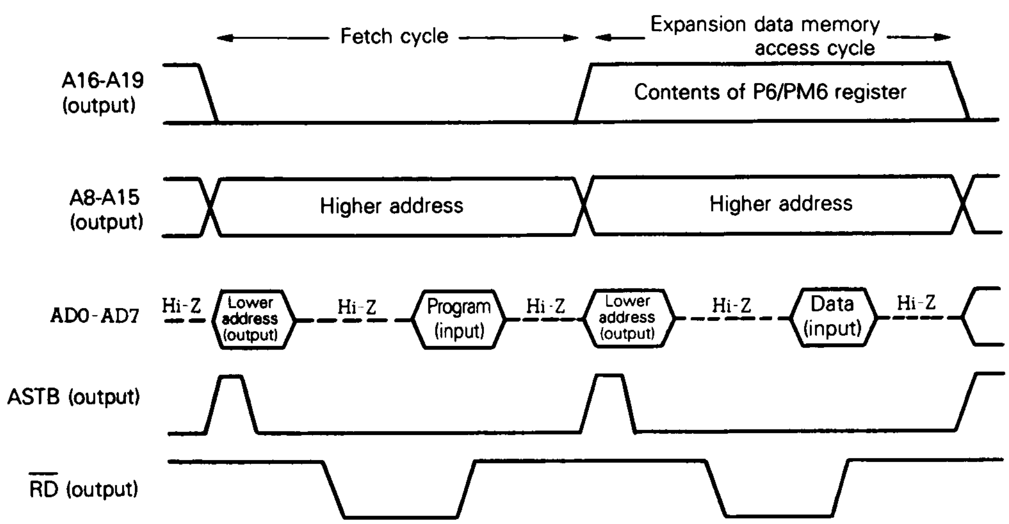
The data memory can be expanded to 960K bytes by setting the MM6 bit of the MM register to 1; therefore, the memory space can be expanded to 1M bytes. In this case, the P60 through P63 pins output the highest address bits (A16 through A19).

Example:

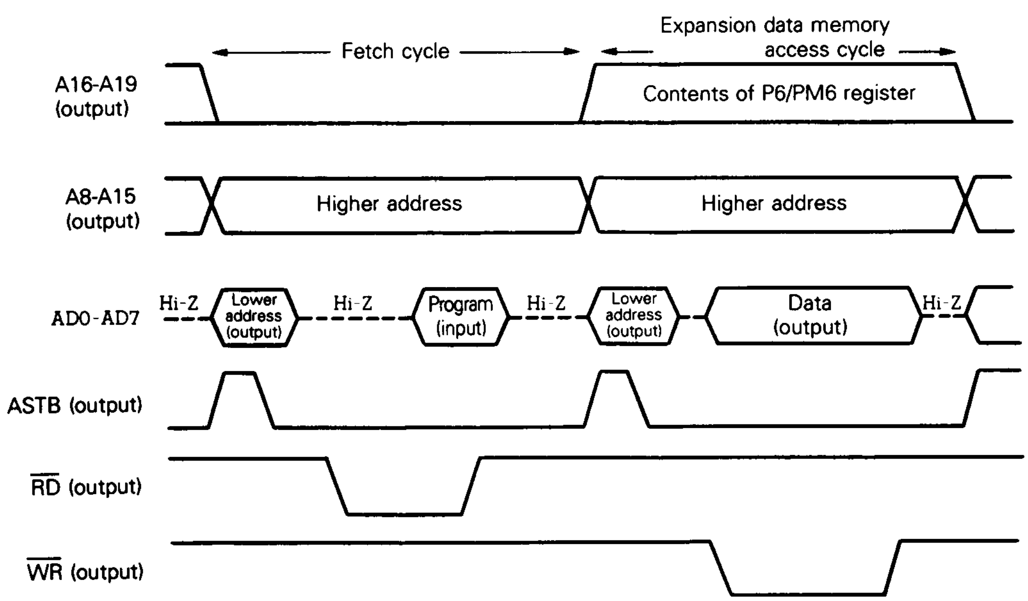
```
MOV MM, #47H    ; Expansion to 1M bytes
MOV P6, #3H     ; Latches highest address information
   :            ; (selects bank 3)
MOV A, &!2000H  ; Stores memory contents at 32000H in
                ; A register
```

Fig. 15-6 Accessing Expansion Data Memory

(a) Read Cycle



(b) Write Cycle



15.2.3 μPD78P238 memory mapping

μPD78P238 is provided with a 32K-byte internal PROM and 1024-byte internal RAM. Therefore, its memory mapping is slightly different from that for μPD78234. To make up for this difference, μPD78P238 is provided with a function making it possible to refrain from using part of the internal memory, when so specified by software (memory size select function).

To select memory size, the memory size select register (IMS) is used. To set the memory mapping the same as that for μPD78234, write this register immediately after μPD78P238 has been reset.

The IMS register is not provided to μPD78234 or 78238. However, an instruction that writes this register can be executed with μPD78234 or 78238, without affecting the microcomputer operations.

15.2.4 Memory map with memory expanded

Figs. 15-7 through 15-10 show the memory maps when the memory is expanded. Even when the memory is expanded, the external devices at the same addresses as those of the internal ROM area, internal RAM area, and SFR area (excluding the external SFR area (0FFD0H through 0FFDFH)) cannot be accessed. When these addresses are accessed, the memory and SFRs in μPD78234 take precedence and are accessed, and the ASTB, $\overline{\text{RD}}$, and $\overline{\text{WR}}$ signals are not output (these signals remain inactive). The output level for the address bus and the address/data bus becomes undefined. However, when the memory is extended, if the internal ROM fetching is specified to the same cycles as the external ROM (this is specified by clearing the IFCH bit of the memory expansion mode register (MM) to "0"), the address and ASTB signal, and $\overline{\text{RD}}$ signal are output when accessing the internal ROM. However, the information on the address/data bus at this time is not fetched, and the CPU reads data from the internal ROM (when the $\overline{\text{RD}}$ signal is active, the μPD78234 address/data bus enters the high-impedance state). The bus cycle in this case will be the same as that of the normal read cycle and wait insertion by the programmable wait function becomes effective.

Caution: When the memory is externally extended (this is always normal for the μPD78233), an illegal write access may occur when macro service Type A or Type C is used.
An illegal write access occurs when one of the following three conditions is satisfied:

Condition 1. When transferring data from the memory to SFR using macro service Type A, and the transfer data is D0H to DFH.
2. When transferring data from SFR to the memory using macro service Type A, and the transfer buffer (memory) address is 0FED0H to 0FEDFH, when the macro service is executed.
3. When MPTL address is 0FED0H to 0FEDFH for macro service Type C.

An illegal write acess is performed in the same way as normal memory access. In addition, a wait is inserted according to the setting of the PW20 and PW21 bits of the memory expansion mode (MM) register. Table 15-1 indicates the conditions and operations for illegal write access.

Table 15-1 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	Lower 8 bits of transfer destination buffer (memory) address
3	C	Transfer destination SFR (CR10 or CR11) address	Lower 8 bits of MPTL address

This discrepancy can be avoided by the following method.

1. **Discrepancy caused by condition 1 is difficult to avoid by means of software (this is because, whether or not an illegal access occurs depends on the transfer data). Therefore, overlapping the image in the area 0FF00H to 0FFFFH over the memory address of the external circuit must be avoided using the external address decoder.**
2. **When the macro service to be used is not relevant to condition 1 (when memory transfer is not used when macro service type A is used), or in case of condition 2, the buffer area must be allocated in a manner so that it will not become address0FED0H to 0FEDFH. In case of condition 3, the MPTL address must be allocated in a manner so that it will not become address 0FED0H to 0FEDFH.**

This discrepancy can also occur in the in-circuit emulator.

Fig. 15-7 Expansion of μPD78233 Data Memory

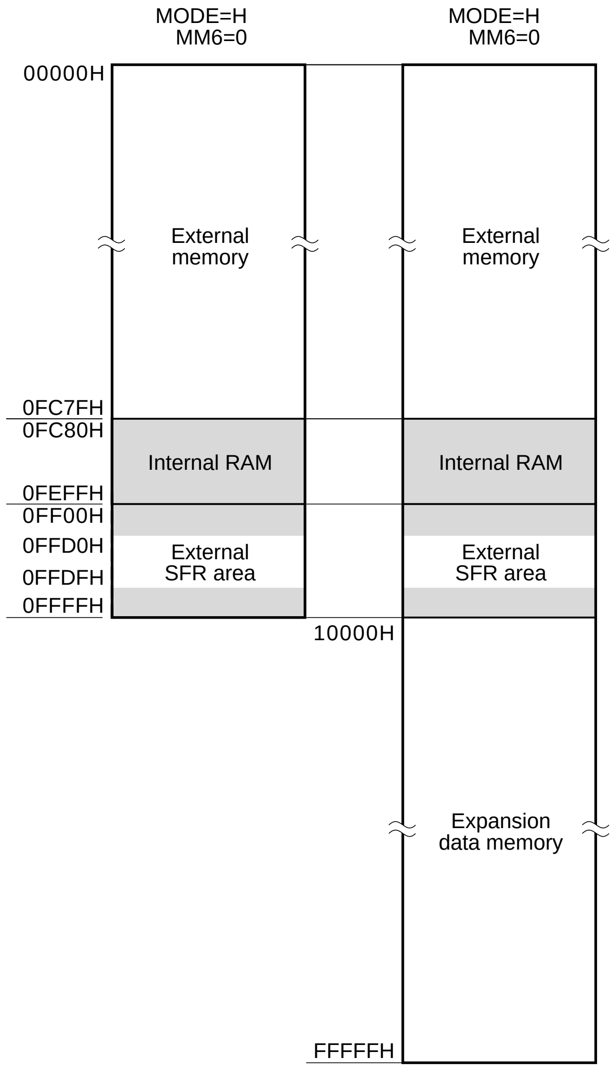


Fig. 15-8 Expansion of Data Memory of μPD78P238 When IMS = DDH and for μPD78234

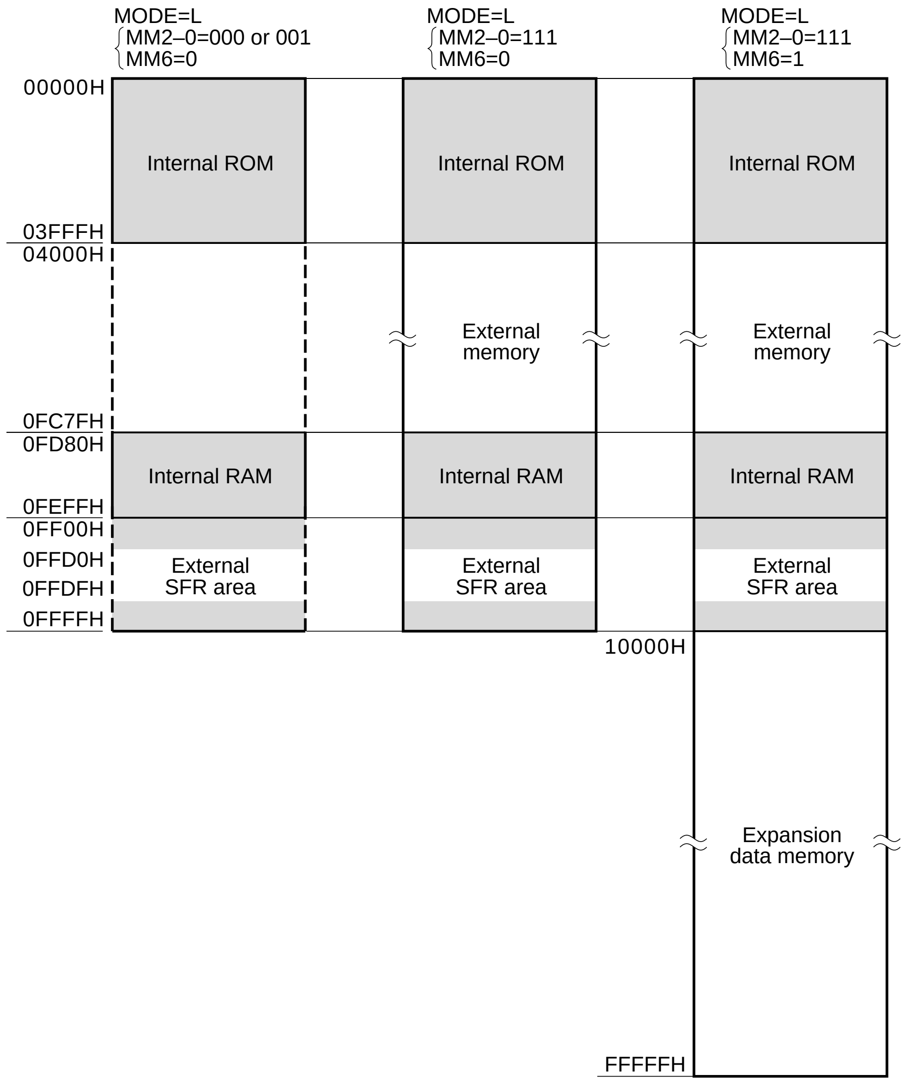


Fig. 15-9 μPD78237 Expanding Data Memory

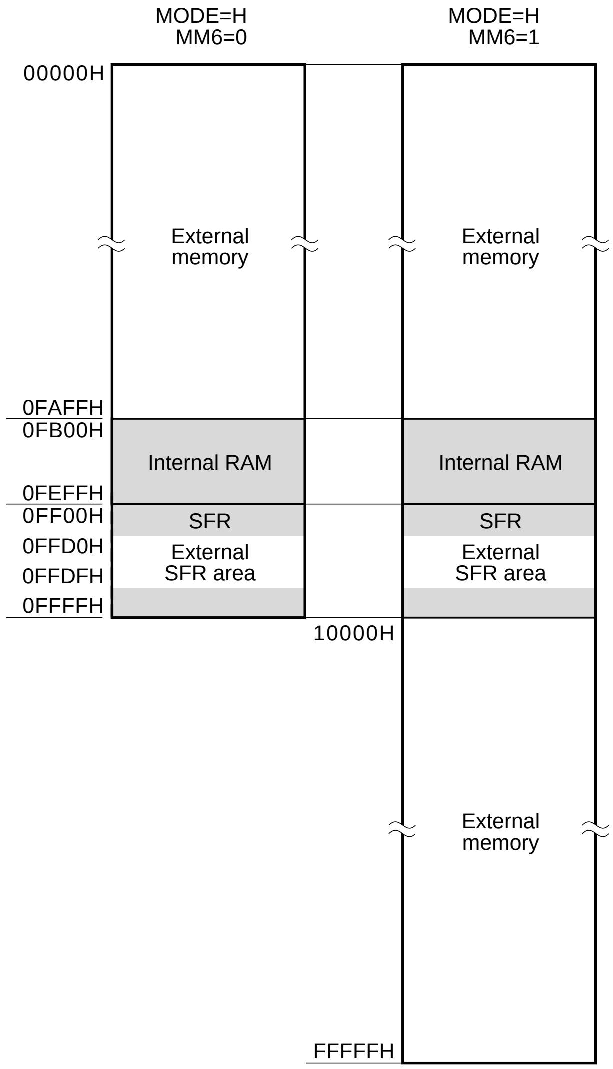
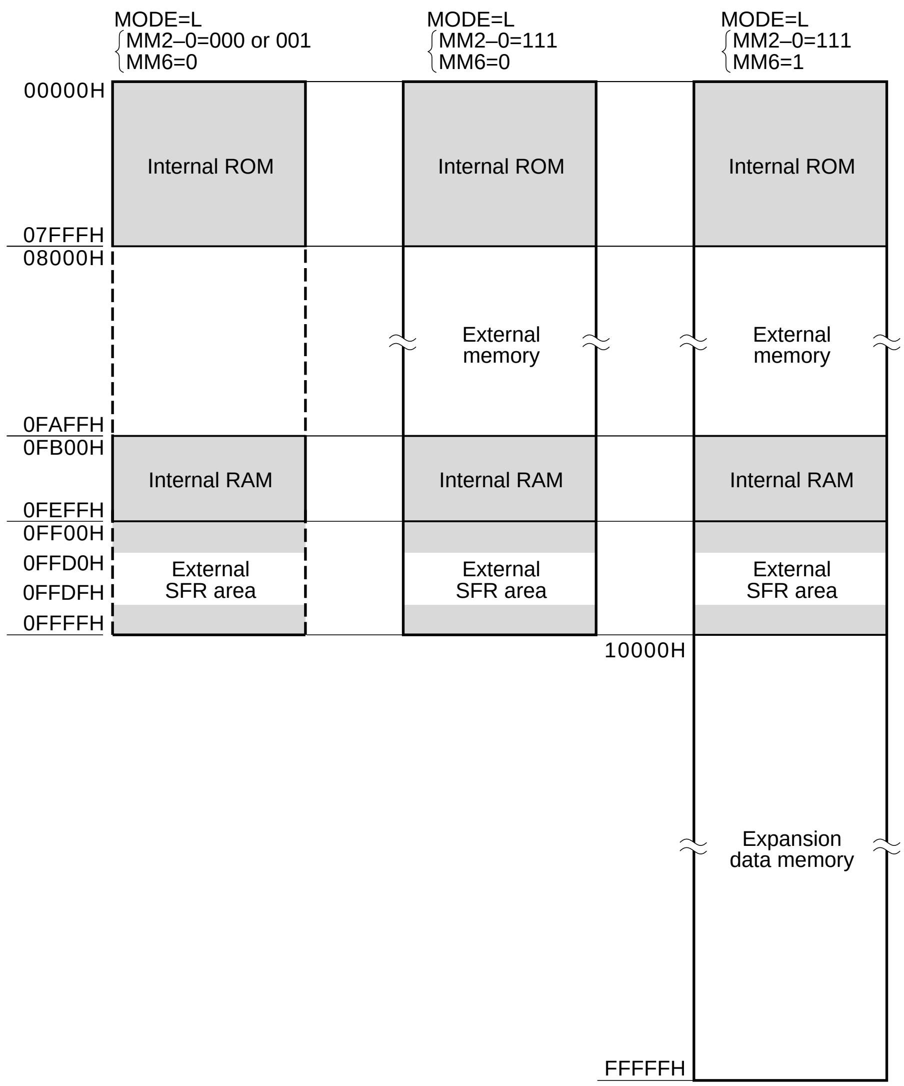


Fig. 15-10 Expanding Data Memory for μPD78238 and μPD78P238, with IMS = FFH



15.2.5 Connecting memories example

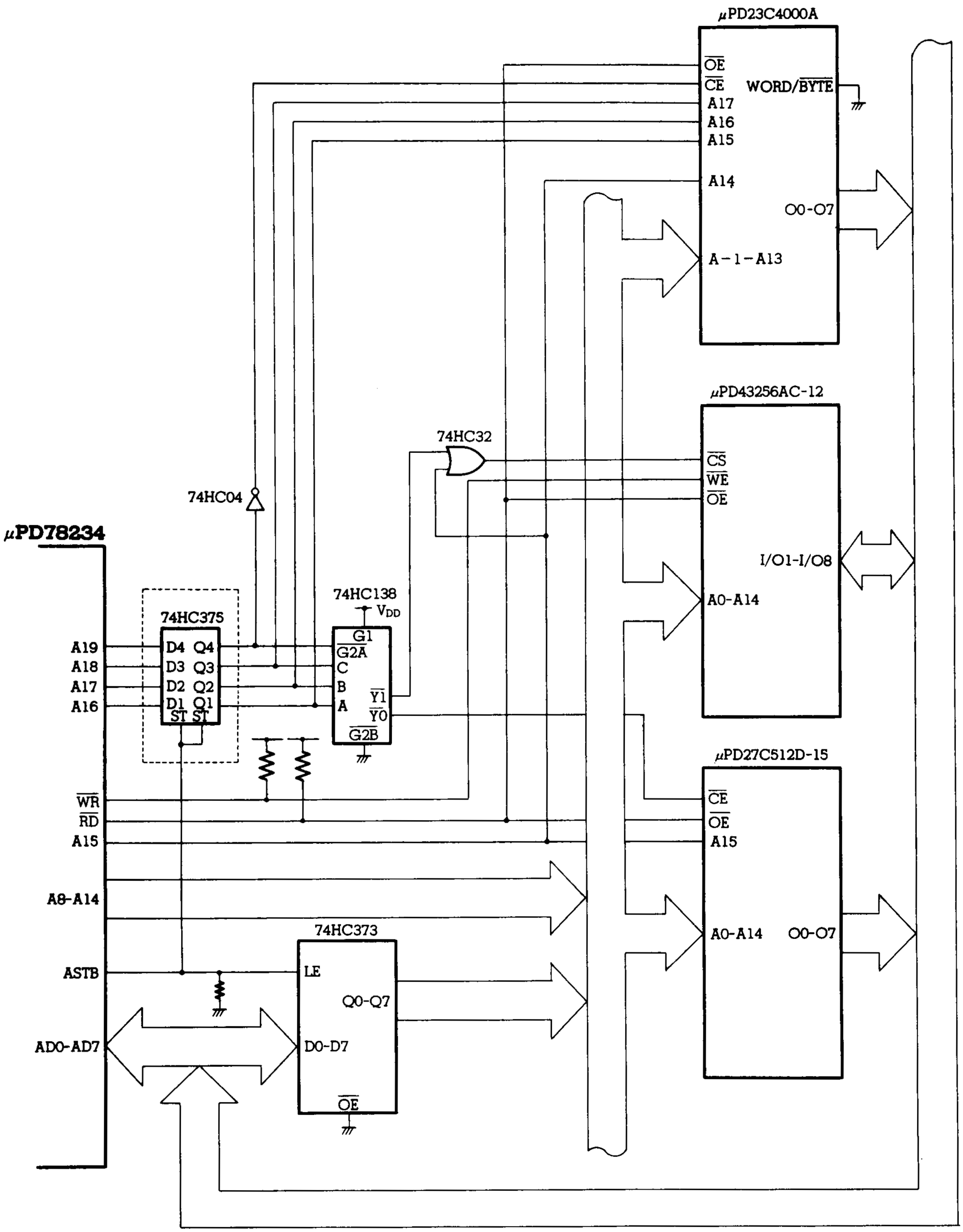
Fig. 15-11 shows an example forconnecting μPD78234 with external memories. In this example, PROM, SRAM, and mask ROM are connected. The addresses are assigned to these memories as follows:

- PROM (μPD27C512D-15) : 0000H-FC7FH (μPD78233)
  4000H-FC7FH (μPD78234)
  0000H-FAFFH (μPD78237)
  8000H-FAFFH (μPD78238)
  and external SFR area FFD0H-FFDFH
- RAM (μPD43256AC-12) : 10000H-17FFFH
- Mask ROM (μPD23C4000A) : 80000H-FFFFFH

Since the mask ROM μPD23C4000A access time is long, insert one wait state, using the programmable wait function (see **15.4 Wait Function**).

The circuit enclosed in a dotted line is necessary, only when an in-circuit emulator is used. Remove 74HC375 and short-circuit Dn with Qn (n = 1 to 4), when the emulator is not used. If this circuit is missing, when the emulator is used, μPD78234 may malfunction (usually, however, the microcomputer operates normally because of the delay time in the address decoder, etc.). When this circuit is not used, a 150-ns model, μPD43256AC-15, can be used as the RAM in the place of μPD43256AC-12.

Fig. 15-11 Example for Connecting External Memories to μPD78234



Remarks: Pull-up resistors need to be connected to the address and address/data bus.

15.3 Internal ROM High-Speed Fetch Function

μPD78234, 78238 and 78P238 are provided with an internal ROM, which can be accessed at high speeds without accessing via a bus control circuit. Usually, the internal ROM is fetched at the same speed as an external ROM. The high-speed fetch function is effected when the IFCH bit for the memory expansion mode register (MM) is set to 1, and the internal ROM is fetched at high speeds.

When the same instruction execution cycle is implemented as that for the external ROM fetching, wait insertion by the wait function is performed. However, no wait is inserted to the internal ROM, when high speed fetching is used.

When the $\overline{RESET}$ signal has been input, the instruction execution cycle becomes equivalent to the external ROM fetch cycle.

15.4 Wait Function

μPD78234 can insert wait cycles in the external memory access cycle when a low-speed external memory or I/O is connected to the microcomputer.

The wait cycles are inserted while the $\overline{RD}$ and $\overline{WR}$ signals are at low level, and each wait cycle extends the low-level periods of these signals by $1/f_{CLK}$ (167 ns at f_{CLK} = 6 MHz).

Wait cycles can be inserted in two modes. In one mode, programmable wait mode, the predetermined number of wait cycles is automatically inserted. In the other mode, insertion of the wait cycles is controlled by an external wait signal.

How the insertion of the wait cycles is controlled is specified by the memory expansion mode register (MM) in respect to the space consisting of 00000H through 0FFFFH, and by the programmable wait control register (PW) in respect to the space of 10000H through FFFFFH. Note that, when the internal ROM and internal RAM are accessed by high-speed fetching, no wait cycle is inserted, and that, when the internal SFRs are accessed, wait cycles are inserted at the necessary timing, regardless of the specification made by the above registers.

When it is specified that accessing is performed in the same number of cycles as that for the external ROM, wait cycles are inserted in accordance with the MM register setting, when the internal ROM is accessed.

The P66 pin functions as the $\overline{WAIT}$ signal input pin, when the input from an external wait signal is specified by either or both of the MM and PW registers. When the $\overline{RESET}$ signal has been input, the P66 pin operates as a general-purpose I/O port pin.

Figs. 15-16 through 15-18 show the bus timing when wait cycles are inserted.

Caution: To use the external wait function, set bit 6 of the PM6 register to 1 to set the P66/$\overline{WAIT}$ pin in the input mode.

Fig. 15-12 Wait Control Space of μPD78233

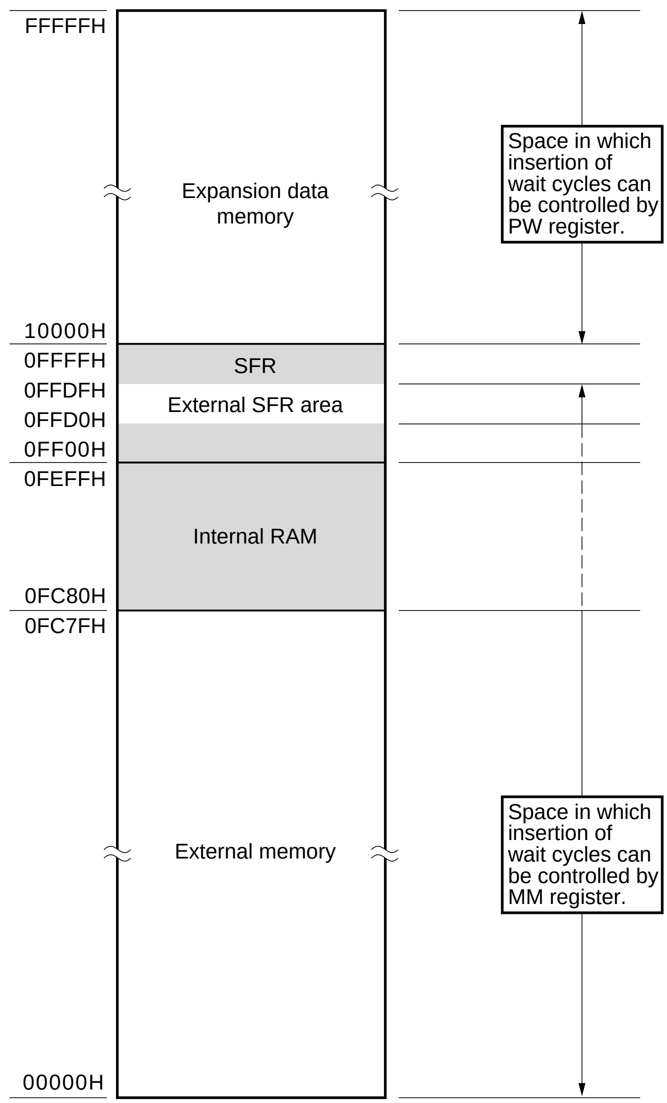


Fig. 15-13 Wait Control Space of μPD78234

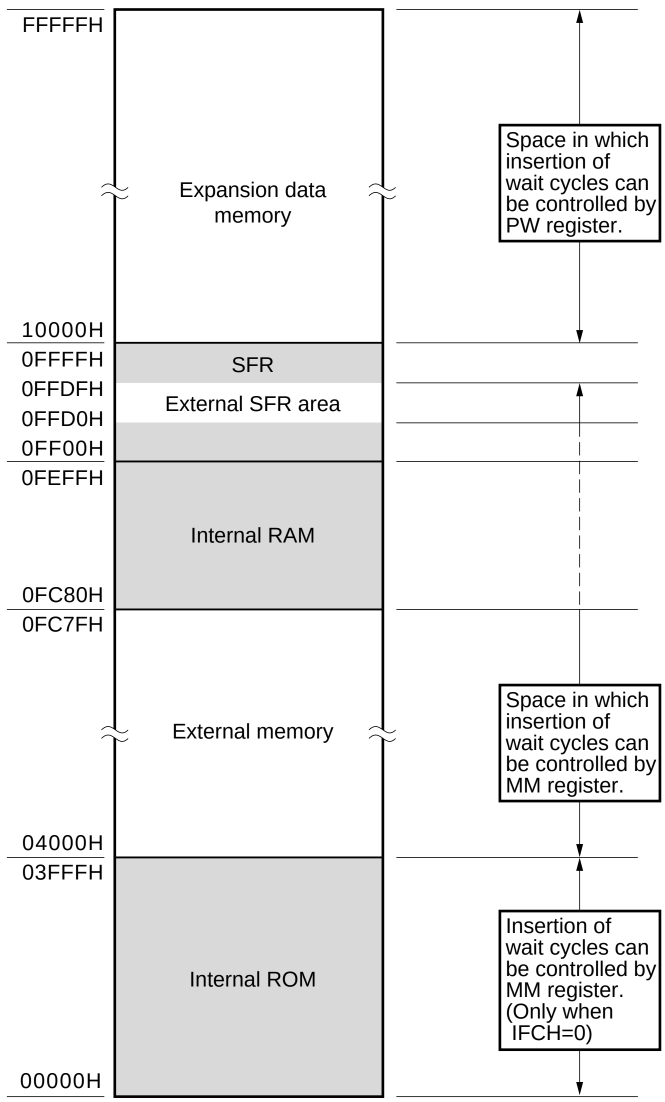


Fig. 15-14 μPD78237 Wait Control Space

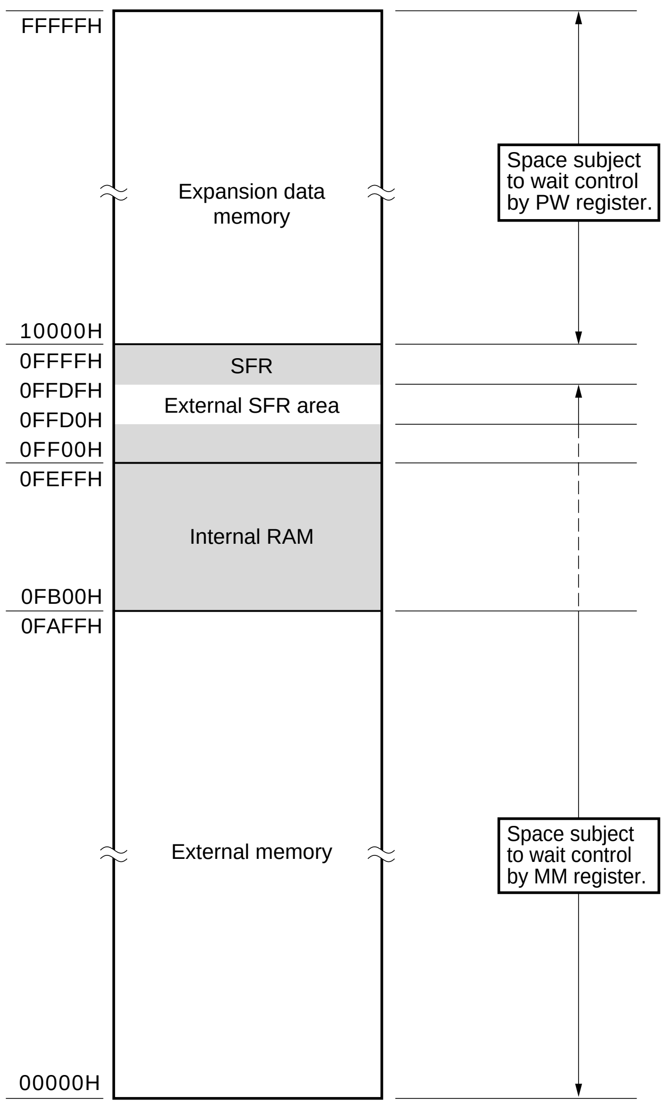


Fig. 15-15 μPD78238 Wait Control Space

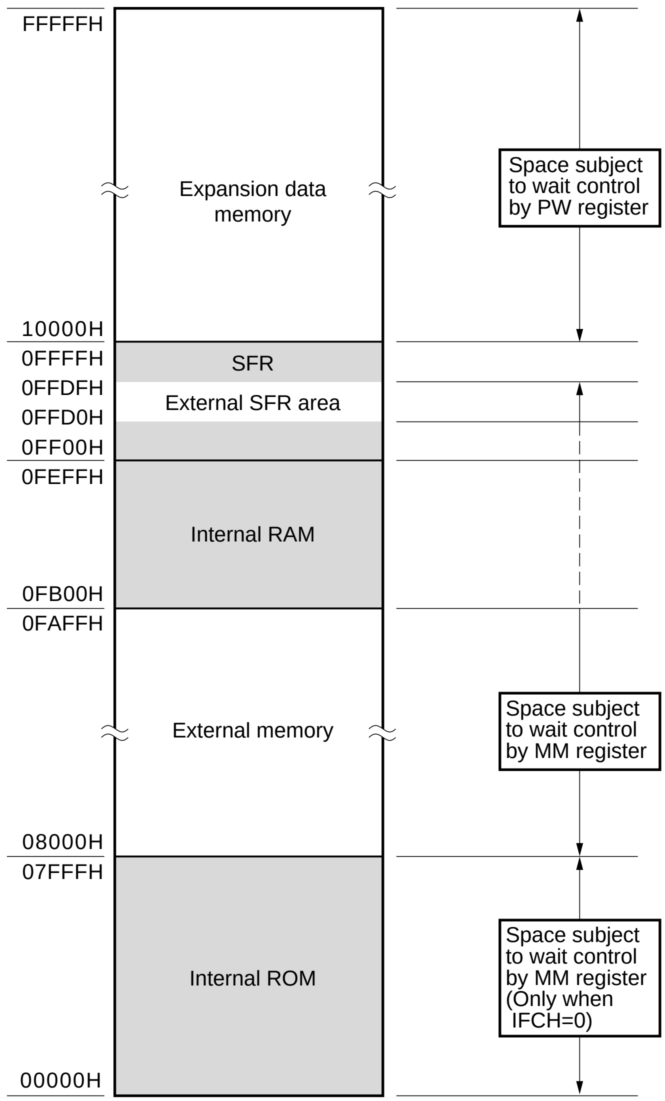
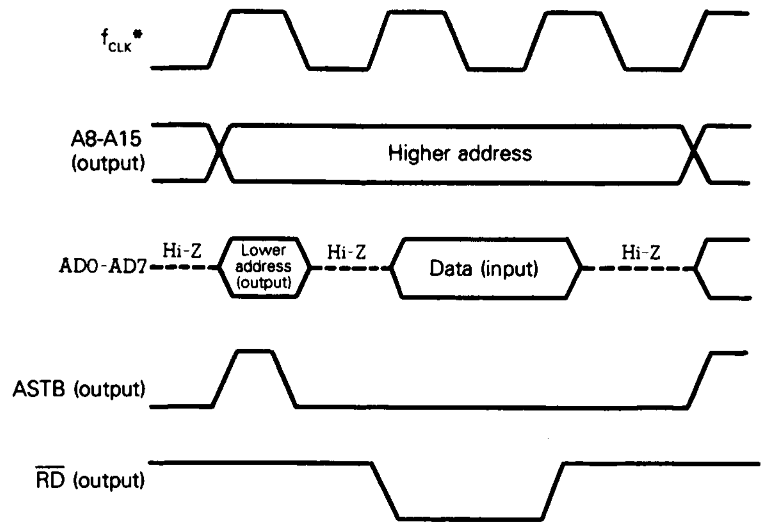


Fig. 15-16 Read Timing of Programmable Wait Function (1/2)

(a) When 0 wait state is set



(b) When 1 wait state is set

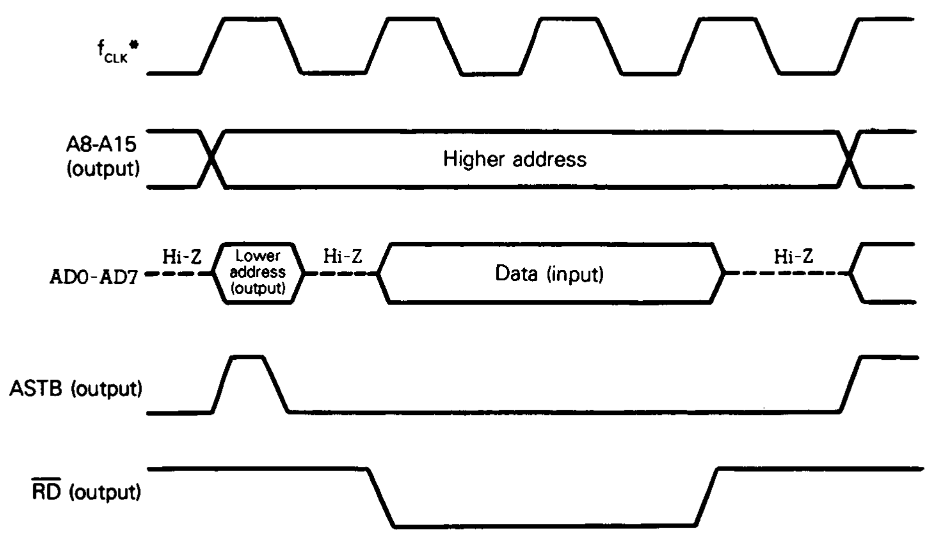
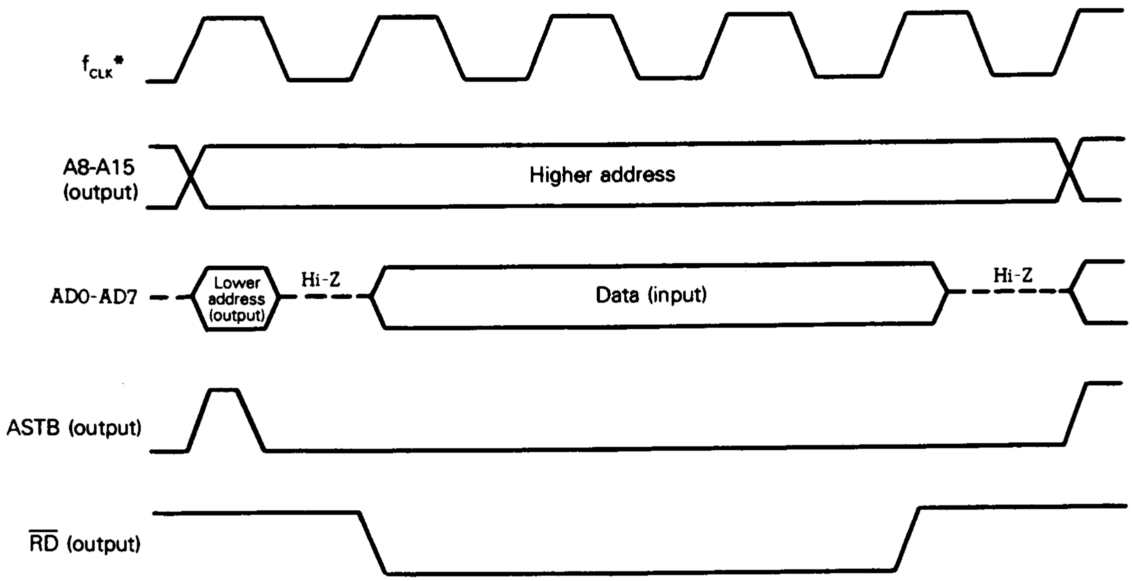


Fig. 15-16 Read Timing of Programmable Wait Function (2/2)

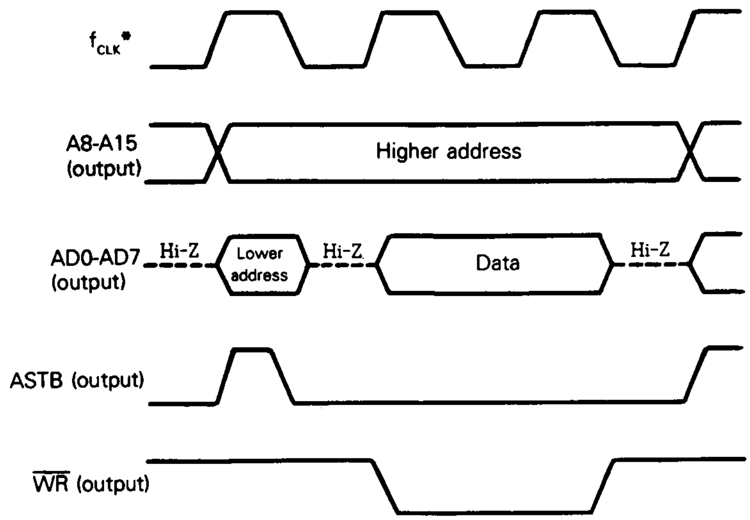
(c) When 2 wait states are set



*: f_{CLK}: system clock frequency (f_{XX}/2)

Fig. 15-17 Write Timing of Programmable Wait Function (1/2)

(a) When 0 wait state is set



(b) When 1 wait state is set

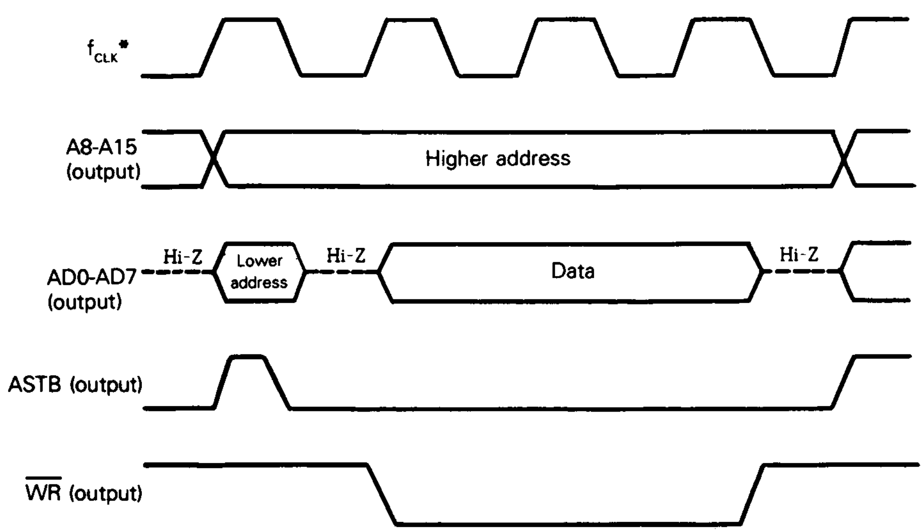
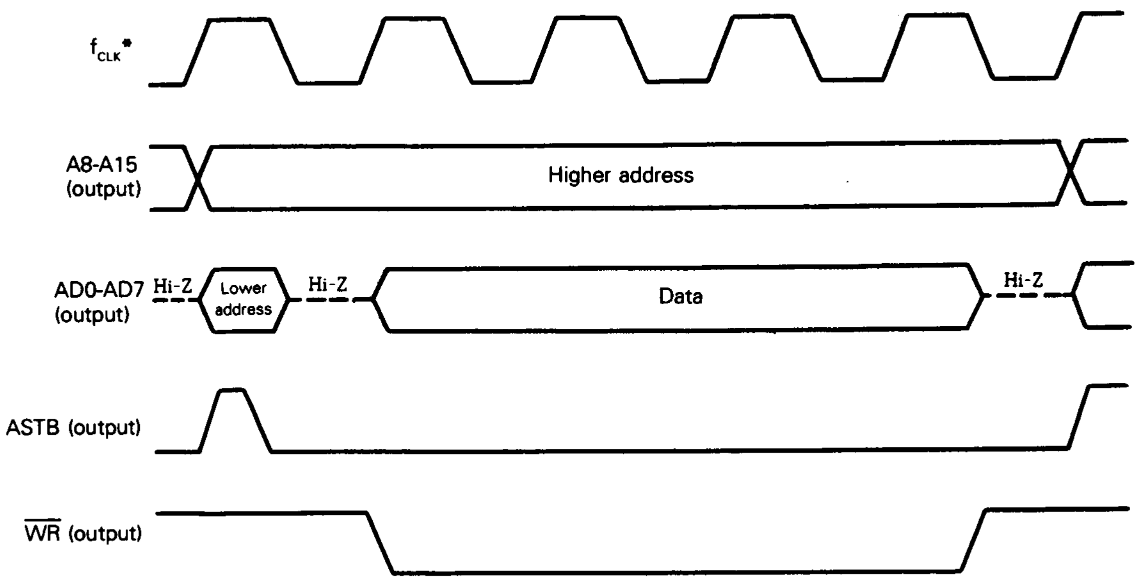


Fig. 15-17 Write Timing of Programmable Wait Function (2/2)

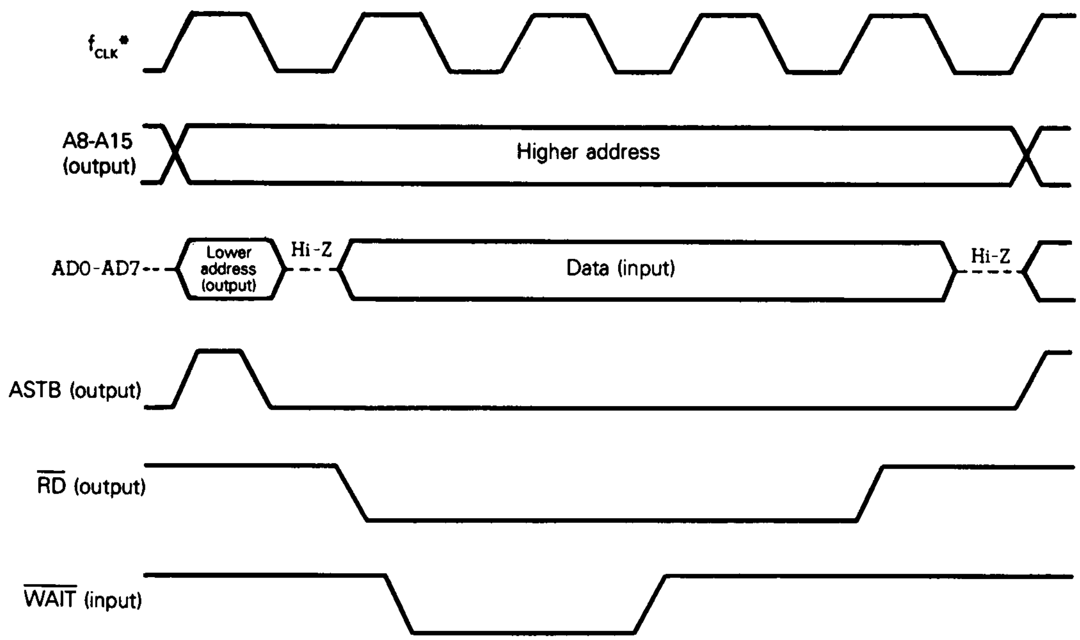
(c) When 2 wait states are set



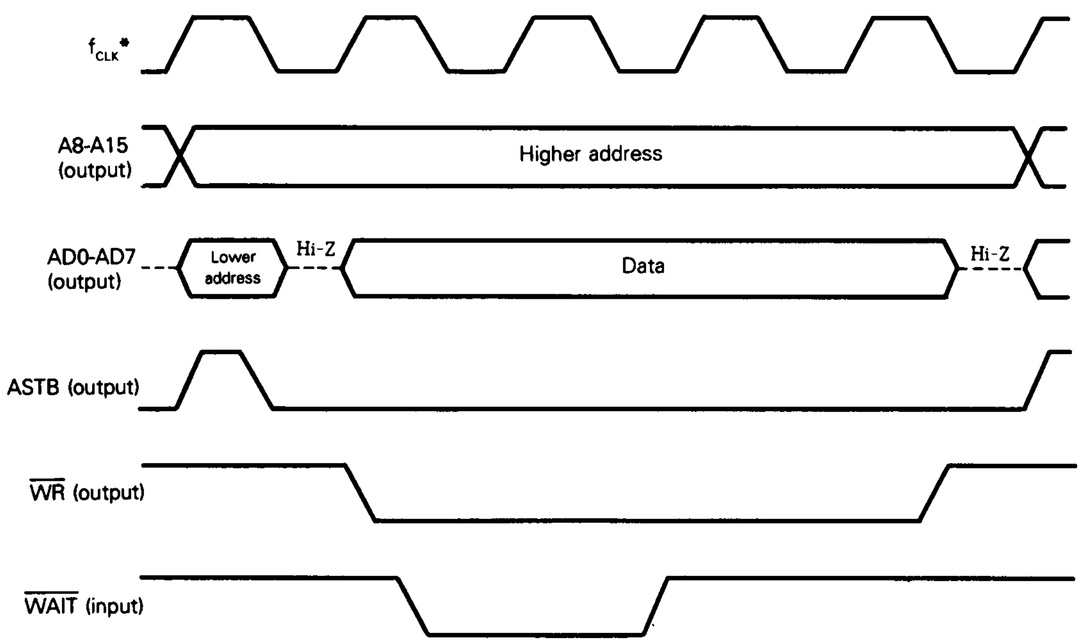
*: f_{CLK}: system clock frequency (f_{XX}/2)

Fig. 15-18 Timing of External Wait Signal

(a) Read timing



(b) Write timing



*: f_{CLK}: system clock frequency (f_{XX}/2)

15.5 Pseudo Static RAM Refresh Function

15.5.1 Function

μPD78234 is provided with pseudo static RAM refresh function that can directly connect pseudo static RAM.

The pseudo static RAM refresh function output refresh pulses at arbitrary intervals. The refresh pulse output intervals are set by the refresh mode register (RFM) and the external access cycle is changed to a refresh bus cycle pseudo static RAM (the write pulse width is smaller by half clock than the pulse width when the pseudo static RAM is not connected).

μPD78234 is also provided with a function that supports self-refresh operation to reduce the power dissipation of the pseudo static RAM application system.

15.5.2 Refresh mode register (RFM)

The RFM register is an 8-bit register that controls the refresh cycle of the pseudo static RAM and switching of self-refresh operation.

Data can be read from or written to this register by an 8-bit manipulation instruction or bit manipulation instruction. The format of this register is shown in Fig. 15-19.

When the $\overline{\text{RESET}}$ signal is input, the contents of this register are initialized to 00H. The $\overline{\text{REFRQ}}$ pin is set in the port mode and functions as the P67 pin.

Fig. 15-19 Refresh Mode Register (RFM) Format

	⑦	6	5	④	3	2	1	0
RFM	RFLV	0	0	RFEN	0	0	RFT1	RFT0

f_{CLK} = 6 MHz

RFT1	RFT0	Refresh pulse output cycle
0	0	$16/f_{CLK}$* (2.6 μs)
0	1	$32/f_{CLK}$ (5.3 μs)
1	0	$64/f_{CLK}$ (10.7 μs)
1	1	$128/f_{CLK}$ (21.3 μs)

RFLV	RFEN	$\overline{\text{REFRQ}}$ pin output control
×	0	Port mode
0	1	Self-refresh operation ($\overline{\text{REFRQ}}$ low level)
1	1	Refresh pulse output enabled

Remarks: ×: 0 or 1

***:** f_{CLK}: system clock frequency ($f_{XX}/2$)

15.5.3 Operation

(1) Pulse refresh operation

To support the pulse refresh cycle of pseudo static RAM, the $\overline{\text{REFRQ}}$ pin outputs a refresh pulse in synchronization with the bus cycle.

Make an adjustment by using the system clock frequency and the bits 1 and 0 (RFT1 and RFT0) in the refresh mode register (RFM), so that the pulse is generated 512 times or more in 8 ms.

Table 15-2 System Clock Frequency and Refresh Pulse Output Cycle When Pseudo Static RAM Is Used

System clock frequency (f_{CLK}) MHz	Refresh pulse output cycle specification	RFT1	RFT0
$4.096 < f_{CLK} \leqq 6$ ($8.192 < f_{XX} \leqq 12$)	$64/f_{CLK}$	1	0
$2.048 < f_{CLK} \leqq 4.096$ ($4.096 < f_{XX} \leqq 8.192$)	$32/f_{CLK}$	0	1
$2 \leqq f_{CLK} \leqq 2.048$ ($4 \leqq f_{XX} \leqq 4.096$)	$16/f_{CLK}$	0	0

This pulse refresh operation should be performed not overlapping the external memory access operation. During the refresh cycle, the external memory access cycle is kept pending (the signals ASTB, $\overline{\text{RD}}$, and $\overline{\text{WR}}$ are inactive) and the refresh cycle is kept pending during the external memory access cycle.

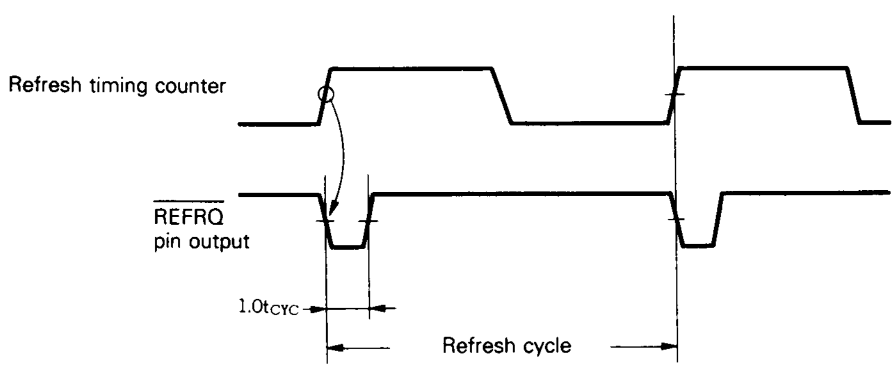
If the pulse refresh operation does not contend with an external memory access operation, the refresh cycle is executed without affecting the instruction execution by the CPU.

(a) When internal memory is accessed

The refresh bus cycle is output at intervals specified by the RFM register even when external pseudo static RAM is not accessed and the internal memory is accessed instead, so that the data stored in pseudo static RAM is retained.

In this case, the instruction execution by the CPU is not affected.

Fig. 15-20 Pulse Refresh Operation When Internal Memory Is Accessed



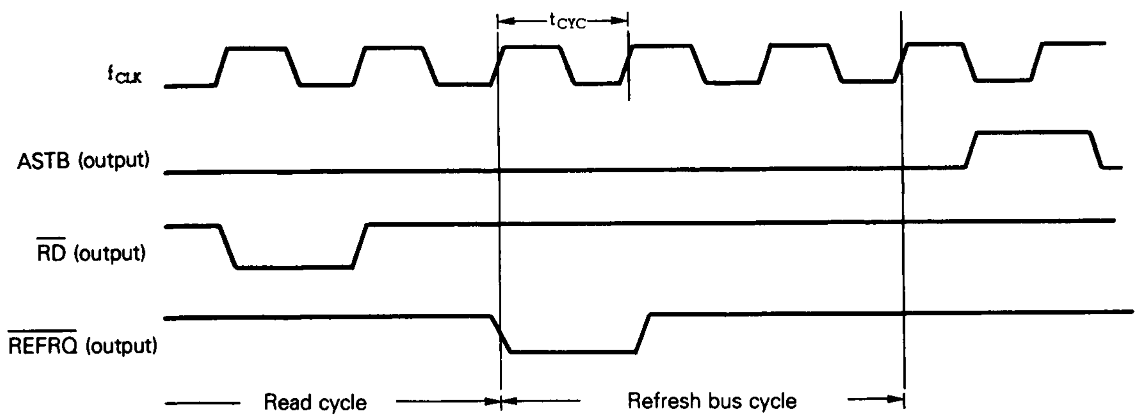
t_{CYC}: system clock cycle time ($1/f_{CLK}$) [ns] (f_{CLK} = 6 MHz)

(b) When external memory is accessed

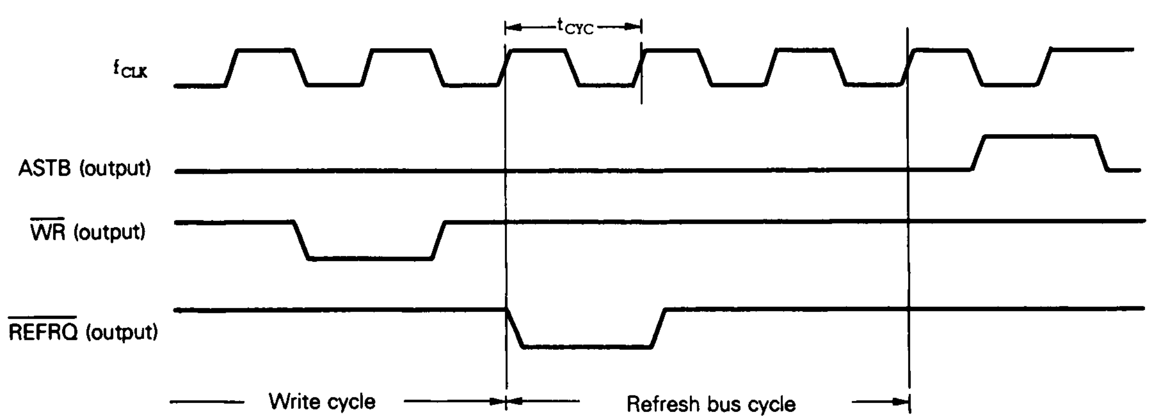
The refresh bus cycle is generated at intervals specified by the refresh mode register (RFM).
pseudo static RAM may malfunction when the access timing overlaps the refresh pulse output timing; therefore, μPD78234 generates a refresh bus cycle equivalent to three clock cycles in synchronization with the bus cycle.

Fig. 15-21 Pulse Refresh Operation When External Memory Is Accessed

(a) When memory is read



(b) When memory is written



(2) Self-refresh operation

This mode is to retain the contents of the pseudo static RAM even in the standby mode.

(a) Setting self-refresh operation mode

When the bit 4 (RFEN) of the RFM register is set to 1 and the bit 7 (RFLV) is cleared to 0, the $\overline{\text{REFRQ}}$ pin outputs a low-level signal, setting pseudo static RAM in the self-refresh operation mode.

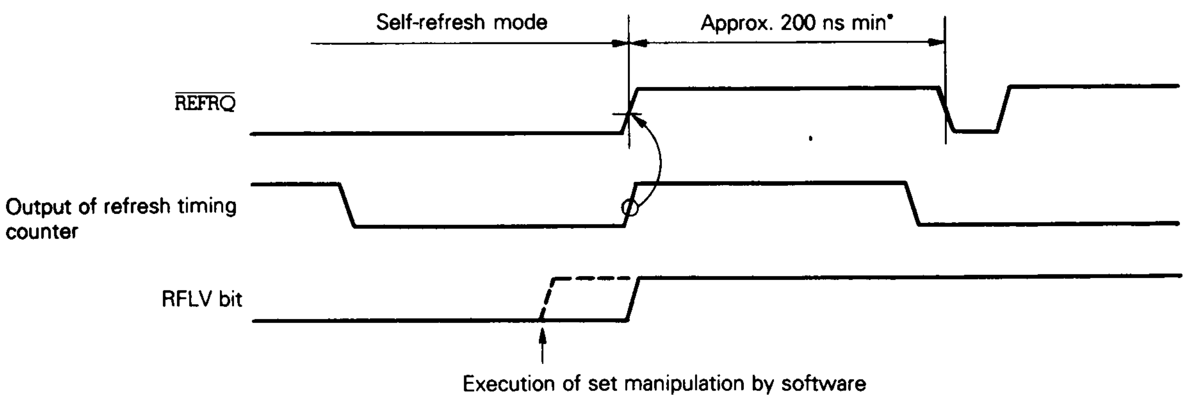
(b) Restoration from self-refresh operation

Refresh pulse output to the pseudo static RAM is disabled, for approximately 200 ns* after the $\overline{\text{REFRQ}}$ pin output level is changed from low level to high level. Therefore, the $\overline{\text{REFRQ}}$ pin output rises in synchronization with the refresh timing counter, so that the refresh pulse is not output during the disable period.

In addition, the read out level for the RFLV bit is set to 1, when the $\overline{\text{REFRQ}}$ pin level changes from low level, to high level so that a $\overline{\text{REFRQ}}$ pin level change from low level to high level can be detected.

***:** Time varies depending on the pseudo static RAM speed ranking.

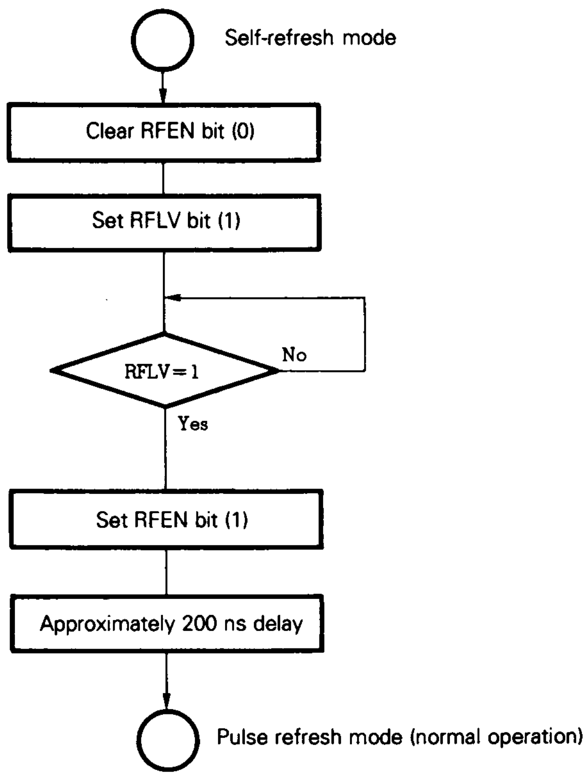
Fig. 15-22 Restoration Timing from Self-Refresh Operation



***:** Referesh disabled period

Caution: When changing the RFLV bit for the refresh mode register (RFM) from 0 to 1, if the RFEN bit has been set to 1 (including when set to 1 simultaneously with the RFLV bit), an approximately 2.6 V glitch may be output from the $\overline{\text{REFRQ}}$ pin for approximately 10 ns.
Therefore, when setting the RFLV bit to 1, set the RFLV bit as indicated in Fig. 15-23. A 200 ns delay after setting the RFEN bit is to assure the access disable time, when the pseudo static RAM returns from the self-refresh mode.

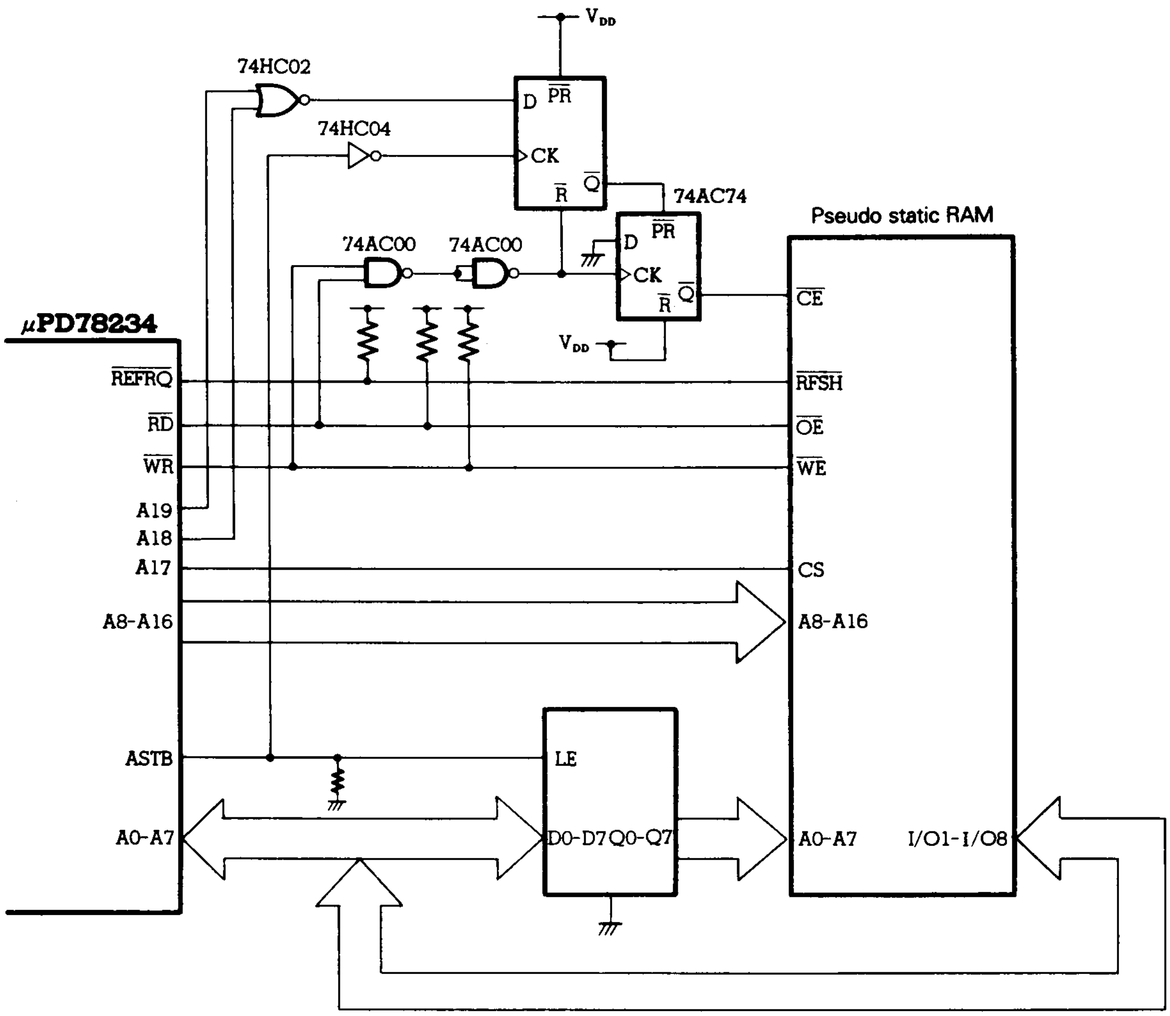
Fig. 15-23 Operation to Return from Self-Refresh Operation



15.5.4 Example for connecting pseudo static RAM

Fig. 15-24 shows an example for connecting a pseudo static RAM. In this example, the pseudo static RAM is assigned to addresses 20000H through 3FFFFH.

Fig. 15-24 Example for Connecting Pseudo Static RAM



Remarks 1: To make sure that the precharge time and access time for the pseudo static RAM elapse, use a device having a high speed. To make sure that the precharge time of the pseudo SRAM elapses, use a high-speed product, such as μPD428128CZ-80 or equivalent.

2: Pull-up resistors need to be connected to the address and address/data bus.

15.6 Note

(1) The address data output to P50/A8-P57/A15, P60/A16-P63/A19 is effective only from the rising edge of the ASTB signal to the rising edge of the $\overline{RD}$ signal or the $\overline{WR}$ signal. The output levels of P50/A8-P57/A15, P60/A16-P67/A19 are undefined during other periods. circuits must be designed in a manner so that the output of an undefined value will cause no problems. Refer to the data sheet of each product for the specifications concerning the effective peiord for address output.

(2) When the memory is externally extended (this is always normal for the μPD78233), an illegal write access may occur when macro service Type A or Type C is used.
An illegal write access occurs when one of the following three conditions is satisfied:

Condition 1: When transferring data from the memory to SFR using macro service Type A, and the transfer data is D0H to DFH.
2: When transferring data from SFR to the memory using macro service Type A, and the transfer buffer (memory) address is 0FED0H to 0FEDFH, when the macro service is executed.
3: When MPTL address is 0FED0H to 0FEDFH for macro service Type C.

An illegal write acess is performed in the same way as normal memory access. In addition, a wait is inserted according to the setting of the PW20 and PW21 bits of the memory expansion mode (MM) register. Table 15-3 indicates the conditions and operations for illegal write access.

Table 15-3 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	Lower 8 bits of transfer destination buffer (memory) address
3	C	Transfer destination SFR (CR10 or CR11) address	Lower 8 bits of MPTL address

This discrepancy can be avoided by the following method.

1. Discrepancy caused by condition 1 is difficult to avoid by means of software (this is because, whether or not an illegal access occurs depends on the transfer data). Therefore, overlapping the image in the area 0FF00H to 0FFFFH over the memory address of the external circuit must be avoided using the external address decoder.

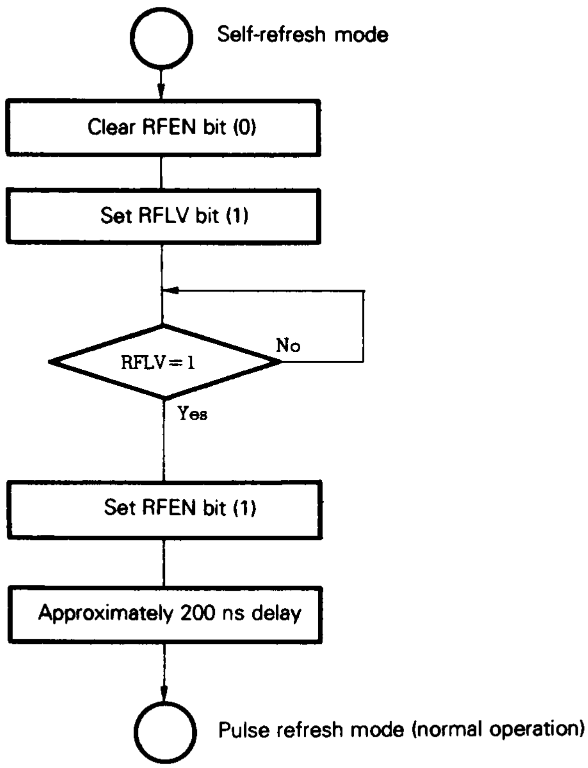
2. When the macro service to be used is not relevant to condition 1 (when memory transfer is not used when macro service type A is used), or in case of condition 2, the buffer area must be allocated in a manner so that it will not become address0FED0H to 0FEDFH. In case of condition 3, the MPTL address must be allocated in a manner so that it will not become address 0FED0H to 0FEDFH.

This discrepancy can also occur in the in circuit emulator.

(3) To use the external wait function, set the bit 6 of the PM register 6 to 1 to set the P66/$\overline{\text{WAIT}}$ pin in the input mode.

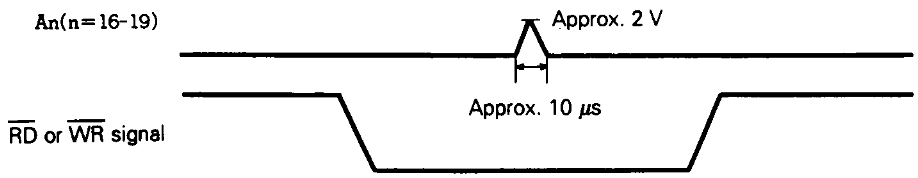
(4) When changing the RFLV bit for the refresh mode register (RFM) from 0 to 1, if the RFEN bit has been set to 1 (including when set to 1 simultaneously with the RFLV bit), an approximately 2.6 V glitch may be output from the $\overline{\text{REFRQ}}$ pin for approximately 10 ns.
Therefore, when setting the RFLV bit to 1, set the RFLV bit as indicated in Fig. 15-25. A 200 ns delay after setting the RFEN bit is to assure the access disable time, when the pseudo static RAM returns from the self-refresh mode.

Fig. 15-25 Operation to Return from Self-Refresh Operation



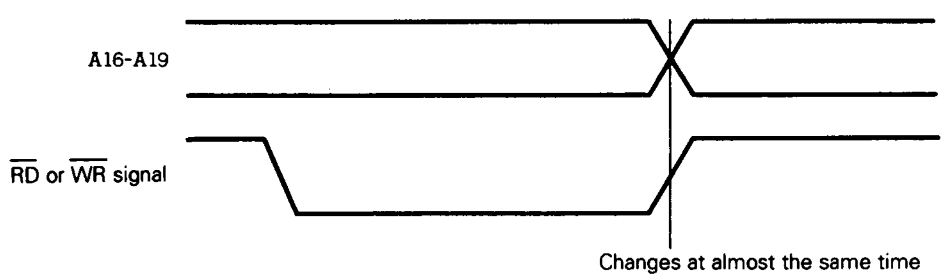
(5) When using the in-circuit emulator, pay attention to the following points:
- A glitch may occur from pin A16 to pin A19, while the $\overline{RD}$ and $\overline{WR}$ signals are active.

Fig. 15-26 Example for A16 to A19 Pin Glitch That May Occur during Emulation



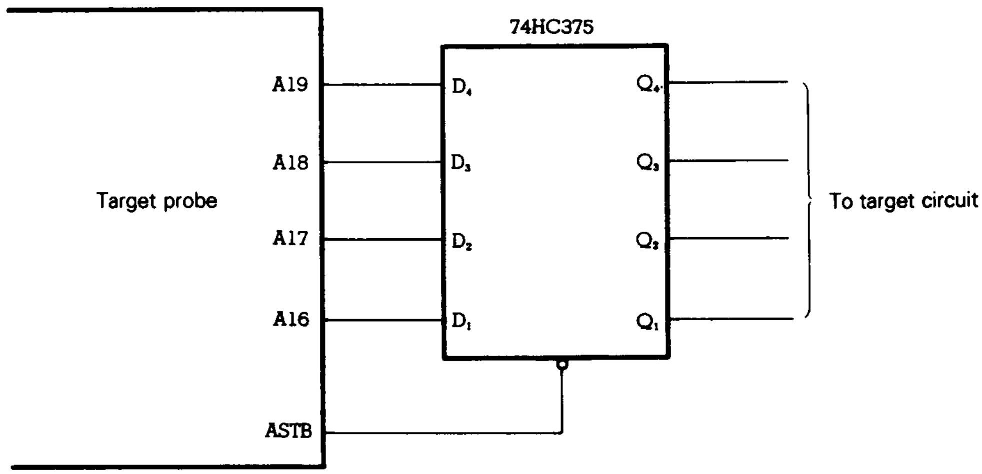
- The hold time for $\overline{RD}$ and $\overline{WR}$ signals in the address signal output to pins A16 to A19 are almost 0 ns.

Fig. 15-27 Address Hold Time Shortage during Emulation



To prevent these problems, it is recommended that a latch be connected to pin A16 to pin A19 during emulation (this measure is not necessary for a device).

Fig. 15-28 Preventing Problems That May Occur during Emulation



CHAPTER 16 STANDBY FUNCTIONS

16.1 Configuration and Functions

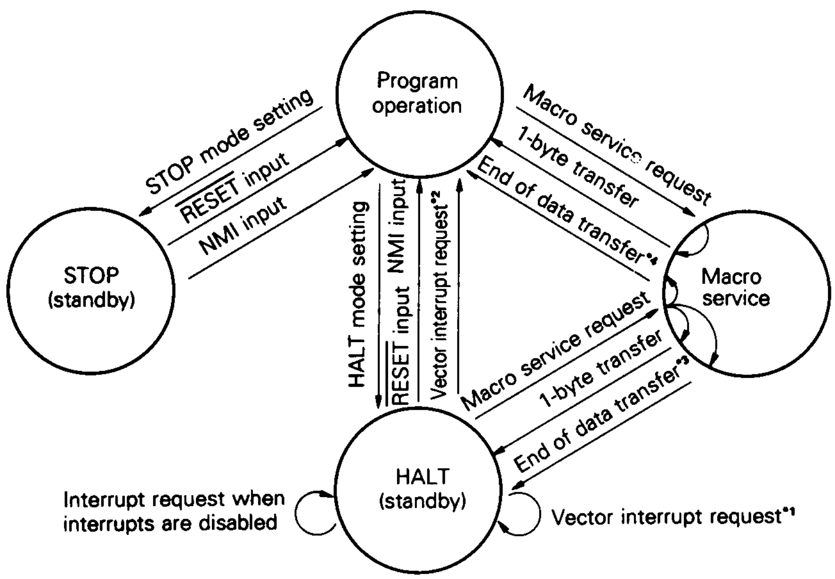
μPD78234 is provided with a standby function that can reduce the power dissipation of the system. This function can be effected in the following two modes:

- HALT mode
  In this mode, the operation clock of the CPU is stopped. By using this mode in combination with an ordinary operation mode so that the CPU intermittently operates, the total power dissipation of the system can be reduced.
- STOP mode
  The oscillator is stopped to stop the system. In this mode, the power dissipation of the system can be minimized, with power due to leakage current dissipated.

These standby modes (STOP/HALT modes) are set by software, as illustrated in Fig. 16-1.

Fig. 16-2 shows the functional block that implements the standby function.

Fig. 16-1 Standby Modes



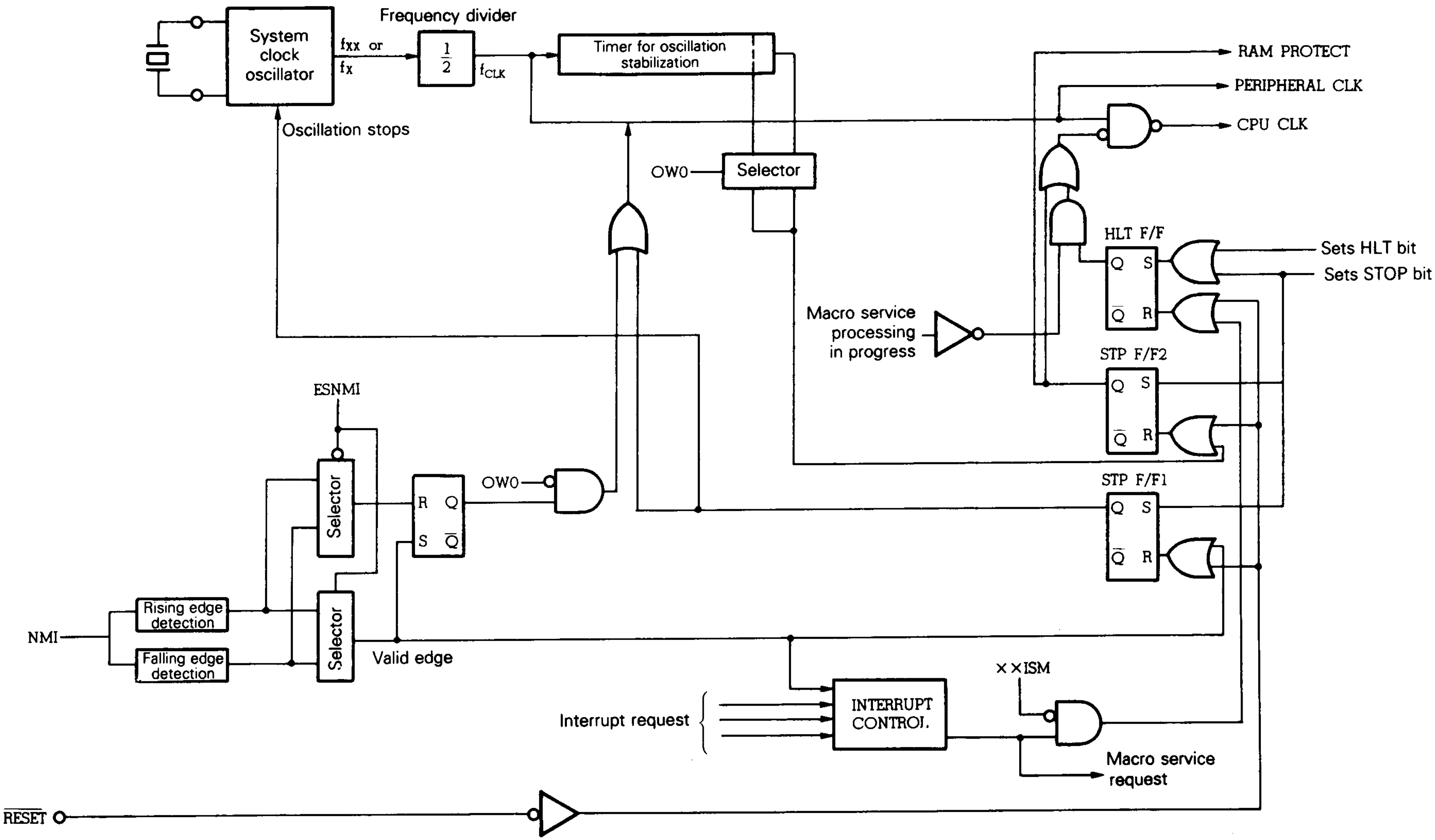
***1:** Vector interrupts with a low priority (Interrupts with a low priority are disabled when the HALT mode is set.)

2: Vector interrupts with a high priority, or when interrupts with a low priority are enabled when the HALT mode is set.

3: Macro services with a low priority (Interrupts with a low priority are disabled when the HALT mode is set.)

4: Macro services with a high priority level, or when interrupts with a low priority are enabled when the HALT mode is set.

Fig. 16-2 Standby Function Block



16.2 Standby Control Register (STBC)

The standby control register is an 8-bit register which controls the standby modes. Although data can be read from and written to this register, data can be written to this register only by a special instruction (MOV STBC,#byte) to prevent the application system from stopping accidentally due to overrunning of the program. Fig. 16-3 shows the format of STBC.

When the $\overline{\text{RESET}}$ signal is input, the contents of STBC are initialized to 0000×000B.

Fig. 16-3 Standby Control Register (STBC) Format

	7	6	5	4	3	2	1	0
STBC	0	OW0	0	0	×	0	STP	HLT

HALT mode select bit

When this bit is set to 1, the HALT mode is set. This bit is automatically cleared to 0 when the HALT mode is released.

STOP mode select bit

When this bit is set to 1, the STOP mode is set. This bit is automatically cleared to 0 when the STOP mode is released.

×: 0 or 1

OW0	Duration of oscillation stabi-lization time when STOP mode is released by NMI
0	High- or low-level time of NMI input + 16-bit count time of os-cillation stabilization counter
1	15-bit count time of oscillation stabilization counter

16.3 HALT Mode

16.3.1 Setting and operations of HALT mode

The HALT mode is set when the HLT bit of the STBC register is set to 1.

Data can be written to the STBC register in units of 8 bits only by a special instruction; therefore, the HALT mode can be set by only the "MOV STBC, #01H" instruction.

Caution: If the HALT mode is set, when the condition, under which the HALT mode is released, is satisfied, the HALT mode is not set, but the next instruction is executed or the execution branches to the vector interrupt service program. To set the HALT mode correctly, clear the interrupt request before setting the HALT mode.

Table 16-1 Operations in HALT Mode

Clock oscillator circuit		Operates
Internal system clock		Operates
CPU		Stops*
I/O lines		Retain status before HALT mode is set
Peripheral functions		Continues
Internal RAM		Retained
Bus line	AD0-AD7	High impedance
	A8-A15	Retained
	A16-A19	Low level
$\overline{RD}$ & $\overline{WR}$ outputs		High level
ASTB output		Low level

***:** Macro service processing is executed.

16.3.2 Releasing HALT mode

The HALT mode can be released by the following three sources:

- Nonmaskable interrupt request (NMI)
- Maskable interrupt request (vector interrupt or macro service)
- $\overline{\text{RESET}}$ input

Table 16-2 lists the conditions under which the HALT mode is released and operations performed after the HALT mode has been released.

Table 16-2 HALT Mode Releasing Conditions and Operations after Release

Releasing source	MK××	PR××	IE	ISP	Operations
$\overline{\text{RESET}}$ input	×	×	×	×	Ordinary reset operation
Nonmaskable interrupt	–	–	×	×	Vector interrupt processing is executed*
Maskable vector interrupt request	0	0	1	×	Vector interrupt processing is executed
	0	×	1	1	
	0	0	0	×	Instruction at the next address is executed (interrupt request is pending)
	0	×	0	1	
	0	1	×	0	HALT mode is retained (interrupt request is pending)
	1	×	×	×	
Macro service	0	0	1	×	Macro service processing is executed. When end condition is satisfied, vector interrupt processing is executed; when not, HALT mode is set again.
	0	×	1	1	
	0	0	0	×	Macro service processing is executed. When end condition is satisfied, instruction at next address is executed; when not, HALT mode is set again.
	0	×	0	1	
	0	1	×	0	HALT mode is set again after macro service processing is executed (interrupt request is pending)
	1	×	×	×	HALT mode is retained.

***:** When the NMIS bit in the interrupt status register (LST) is set to 1, the instruction at the next address is executed. When the NMIS is cleared to 0, the execution branches to the NMI interrupt service program.

Remarks: MK×× : interrupt mask flag
PR×× : priority specification flag
IE : interrupt request enable flag
ISP : interrupt priority status flag

(1) Releasing by nonmaskable interrupt (NMI)

When NMI occurs, the HALT mode is released regardless of whether interrupts are enabled (EI status) or disabled (DI status).

If the NMIS bit in the interrupt status register (IST) is 0, when the HALT mode has been released, the execution branches to the NMI service program. When the NMIS bit is 1 (when the HALT mode is set in the NMI interrupt service program), the program execution is resumed from the instruction next to the one that has set the HALT mode. The execution branches to the NMI interrupt service program, when the NMIS bit is cleared to 0 (by executing the RETI instruction).

(2) Releasing by maskable interrupt request

The HALT mode can only be released by a maskable interrupt, whose interrupt mask flag is 0. If the interrupt priority status flag (ISP) is 0 (enabling the high-priority interrupt only), the HALT mode can only be released by an interrupt, whose priority specification flag is 0 (i.e., by an interrupt with a high priority). However, the macro service is processed, regardless of the priority (an interrupt that is generated at the end of the macro service is subject to priority control).

When the HALT mode has been released and if the interrupt request enable flag (IE) is 1, the execution branches to an interrupt processing program. When the IE flag is 0, the program execution is resumed from the instruction next to the one that has set the HALT mode.

The macro service temporarily releases the HALT mode and is processed once. Then, the HALT mode is set again. When the macro service has been executed the specified number of times, operations are performed in accordance with the IE flag, ISP flag, and priority specification flag.

Table 16-3 Releasing HALT Mode by Maskable Interrupt

Releasing source	MK××	PR××	IE	ISP	Operations
Maskable vector interrupt request	0	0	1	×	Vector interrupt processing is executed
	0	×	1	1	
	0	0	0	×	Instruction at next address is executed (with interrupt request pending)
	0	×	0	1	
	0	1	×	0	HALT mode is retained (interrupt request is pending)
	1	×	×	×	
Macro service	0	0	1	×	Macro service processing is executed. If end condition is satisfied, vector interrupt processing is executed. If not, HALT mode is set again.
	0	×	1	1	
	0	0	0	×	Macro service processing is executed. If end condition is satisfied, instruction at next address is executed. If not, HALT mode is set again.
	0	×	0	1	
	0	1	×	0	Macro service processing is executed and HALT mode is set again. (interrupt request is pending)
	1	×	×	×	HALT mode is retained

Remarks: MK×× : interrupt mask flag
PR×× : priority select flag
IE : interrupt request enable flag
ISP : interrupt priority status flag

(3) Releasing by $\overline{\text{RESET}}$

In the same manner as when the ordinary reset operation is performed, program execution branches to the reset vector address, and the program is executed. Note, however, that the contents of the internal RAM before the HALT mode has been set are retained.

16.4 STOP mode

16.4.1 Setting and operations of STOP mode

The STOP mode is set when the STP bit of the STBC register is set to 1.

Data can be written to the STBC register in units of 8 bits by only a special instruction; therefore, the stop mode can be set by only the "MOV STBC, #02H" instruction.

Table 16-4 Operations in STOP Mode

Clock oscillator circuit		Stops
Internal system clock		Stops
CPU		Stops
I/O lines		Retain status before STOP mode is set
Peripheral functions		Stop all operations*
Internal RAM		Retained
Bus line	AD0-AD7	High impedance
	A8-A15	Retained
	A16-A19	Low level
$\overline{RD}$ & $\overline{WR}$ outputs		High level
ASTB output		Low level

*: Although the A/D converter stops operating, its current dissipation is not reduced, if the CS bit in the A/D converter mode register (ADM) is set.

Caution 1: When the STOP mode is set, the X1 pin is internally short-circuited to V_{SS} (GND potential) to prevent current leakage from the clock oscillator circuit; therefore, use of the STOP mode is inhibited in a system that uses an external clock.

2: Reset the CS bit for the A/D converter.

3: The STOP mode is set, even if the NMI request is kept pending, when an attempt is made to set the STOP mode. To use NMI to release the STOP mode, input the NMI signal again.

16.4.2 Releasing STOP mode

The STOP mode is released by NMI or $\overline{\text{RESET}}$ inputs.

(1) Releasing STOP mode by NMI input

(a) Releasing STOP mode

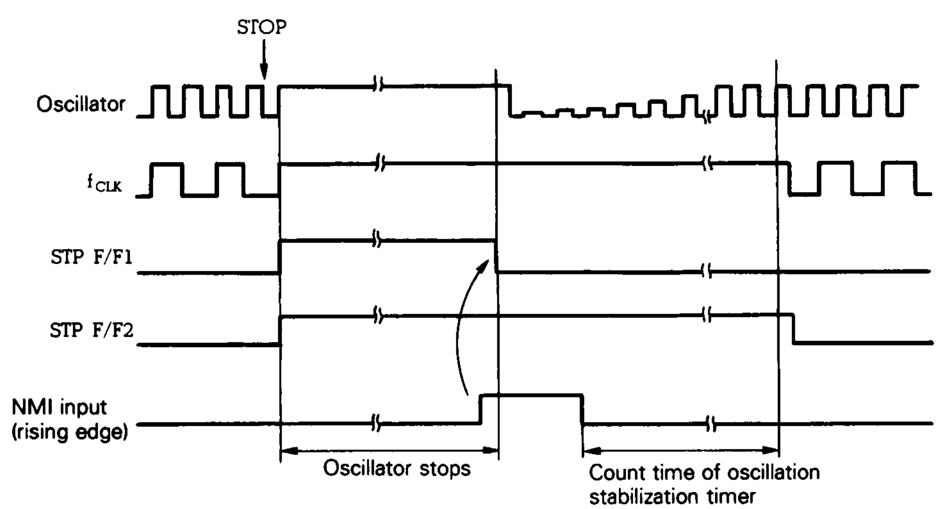
The oscillator resumes oscillation when a valid edge specified by the external interrupt mode register (INTM0) is input to the NMI pin. After the oscillation stabilization time whose duration is specified by the OW0 bit of the standby control register (STBC) has elapsed, the STOP mode is released.

When the STOP mode has been released, execution branches to the NMI interrupt service program if the NMIS bit of the interrupt status register (IST) is 0. If the NMIS bit is 1 (such as when the STOP mode has been set during execution of the NMI interrupt service program), execution is resumed from the instruction next to the instruction that has set the STOP mode, and execution branches to the NMI interrupt service program when NMIS bit is cleared to 0 (by execution of the RETI instruction).

(b) Oscillation stabilization time when OW0 bit = 0

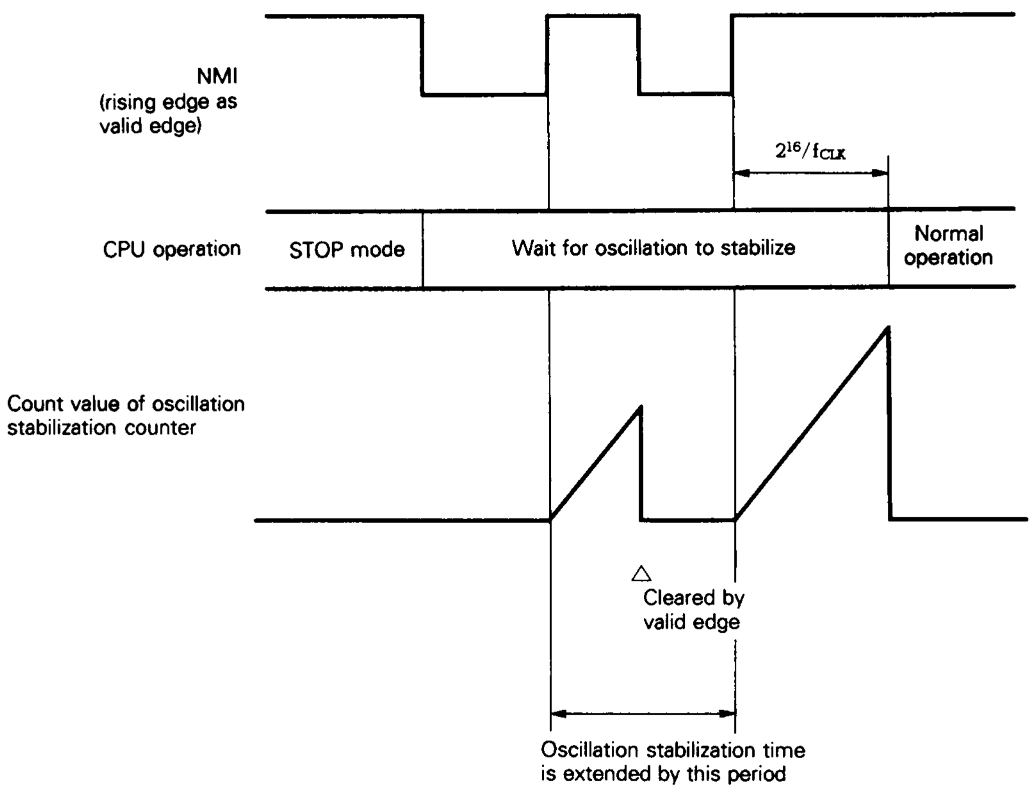
When a valid edge which is specified by the external interrupt mode register (INTM0) is input to the NMI input pin, the oscillator resumes its operation. When the input level of the NMI pin subsequently returns to the original level, the counter for oscillation stabilization starts operating. When this counter has counted 16 bits (approx. 11 ms at f_{XX} = 12 MHz), supply of the internal system clock is started. Therefore, a wait time, which is the sum of the high- or low-level width after a valid edge has been detected on the NMI input pin, and the count time required for the oscillation stabilization counter to count 16 bits, elapses, during which the oscillator stabilizes its operation.

Fig. 16-4 Releasing STOP Mode by NMI Input (when OW0 bit = 0)



Caution: If a valid edge is input to the NMI pin during the oscillation stabilization time, the counter that counts the oscillation stabilization time is cleared and the oscillation stabilization time is extended as follows:
Time until valid edge is input + active level period of NMI
To make sure that the microcomputer is released from the STOP mode, fix the level of the NMI pin to the inactive level during the oscillation stabilization time.

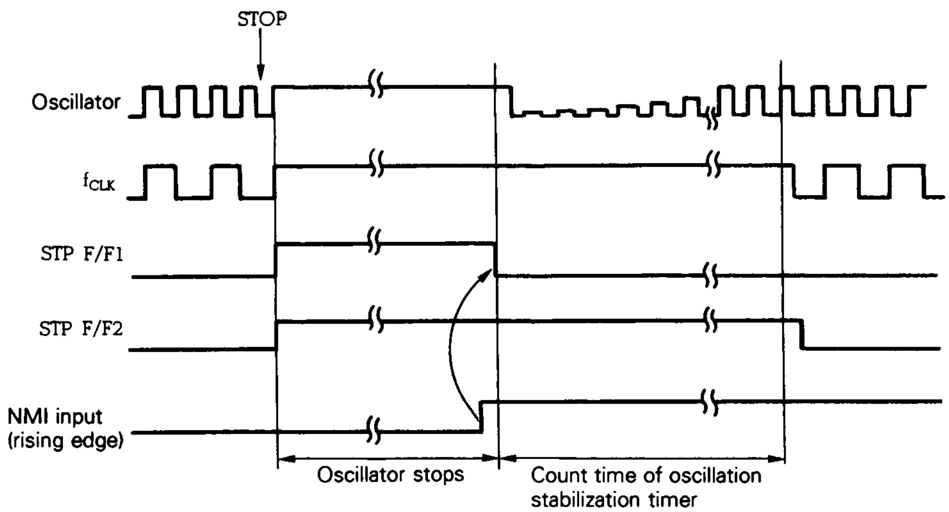
Fig. 16-5 When Oscillation Stabilization Time Is Extended



(c) Oscillation stabilazation time when OW0 bit = 1

The oscillator resumes its operation when a valid edge specified by the external interrupt mode register is input to the NMI pin. At the same time, the oscillation stabilization counter starts operating. When this counter has counted 15 bits (approx. 5.5 ms at f_{XX} = 12 MHz), supply of the internal system clock is started. Therefore, a wait time, which is the time required for the counter to count 15 bits, elapses after a valid edge has been detected on the NMI pin.

Fig. 16-6 Releasing STOP Mode by NMI Input (when OW0 bit = 1)



Caution: If the STOP mode is released by NMI input with the OW0 bit set to 1, emulation cannot be performed by in-circuit emulator. Use μPD78P238 for evaluation when this function is used.

(2) Releasing STOP mode by $\overline{\text{RESET}}$ input

The oscillator starts operating when the level of the $\overline{\text{RESET}}$ input pin has been made low. Make sure that the oscillation stabilization time elapses during the active period of the $\overline{\text{RESET}}$ signal. When the $\overline{\text{RESET}}$ signal is subsequently made high, the ordinary operation is started.

Unlike when the ordinary $\overline{\text{RESET}}$ operation is performed, the data memory retains the contents set before the STOP mode was set.

16.4.3 Notes on using STOP mode

The following items must be checked to reduce the current dissipation in the STOP mode.

(1) Is output level of each output pin appropriate?

The appropriate output level of each pin differs depending on the circuit on the next stage. Select an output level that minimizes the current dissipation.

- If a high level is output when the input impedance of the circuit on the next stage is low, a current flows from the power supply to the port, increasing the current dissipation. This happens especially when the circuit on the next stage is a CMOS IC. The input impedance of a CMOS IC is low when the power is turned off. Output a low level to decrease the current dissipation and not to affect the reliability of the CMOS IC adversely. If a high level is output, a latch up may be caused when the power is applied again.

- Depending on the circuit on the next stage, inputting a low level increase the current dissipation. In this case, output either a high level or high impedance to reduce the current dissipation.

Setting the output level differs depending on the port.

- If the port is in the control mode, the output level is determined according to the status of the internal hardware. Therefore, it is necessary to set the output level taking the status of the internal hardware into consideration.

- In the port mode, the output level can be set by writing to the output latch of the port and port mode register through software.

When the port is in the control mode, the output level can be set easily by changing the mode to the port mode.

(2) Is the input level of each pin appropriate?

Keep the voltage level input to each pin to within V_{SS} to V_{DD}. If a voltage exceeding this range is applied, not only the current dissipation increases, but also the reliability of the μPD78234 is affected.
Also do not apply the intermediate voltage.

(3) Is the internal pull-up resistor necessary?

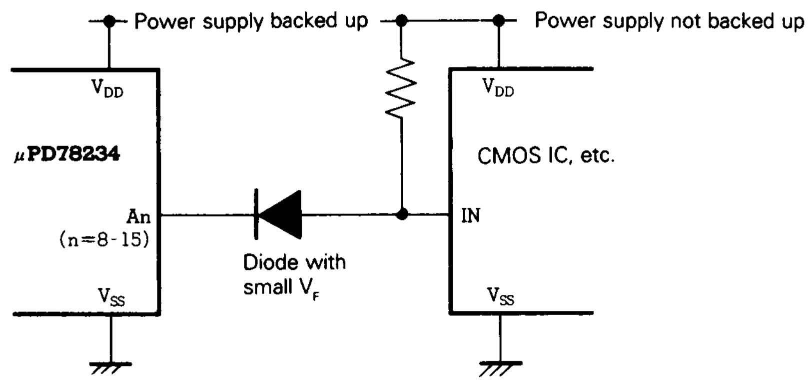
An unnecessary pull-up resistor increases the current dissipation and causes a latch up of the other devices. Specify a mode in which only the necessary pull-up resistors are used.
If necessary and unnecessary pull-up resistors exist in mix, connect external pull-up resistors to the necessary portion and specify a mode in which the internal pull-up resistors are not used.

(4) Is the processing of the address bus and address/data bus appropriate?

The address bus outputs an undfined value (high or low level) in the STOP mode. If the input impedance of the external circuit is low, a current flows out from the address bus, increasing the current dissipation. Therefore, increase the input impedance of the circuit connected to the address bus even in the STOP mode. Especially, the input impedance of a CMOS IC decreases when the power is turned off, causing an abrupt increase in the current dissipation or adverse influences on the reliability of the IC. Therefore, process the address bus as follows:

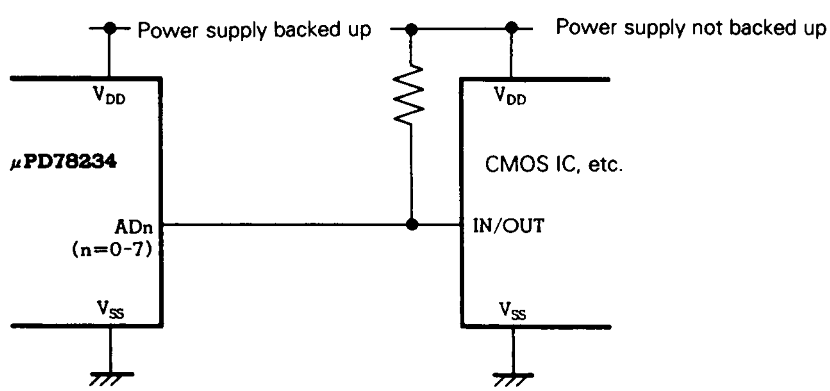
- Do not turn off the power to the CMOS IC (supply power even in the STOP mode).
- Connect a diode with a small V_F as shown in Fig. 16-7 to prevent the current from flowing.

Fig. 16-7 Example of Processing of Address Bus



The address/data bus goes into a high-impedance state in the STOP mode. Usually, the address/data bus is pulled up by pull-up resistors. When these pull-up resistors are connected to a power supply backed up, a current flows through the pull-up resistors, increasing the current dissipation, if the input impedance of the circuit connected to a power supply not acked up. therefore, connect the pull-up resistors to a power supply not backed up, as shown in Fig. 16-8.

Fig. 16-8 Example of Processing of Address/Data Bus



Process the $\overline{RD}$, $\overline{WR}$, ASTB, and $\overline{REFRQ}$ pins in the same manner as the address bus because these pins output fixed levels in the STOP mode. The ASTB pin outputs a low level; therefore, no external component is necessary.

The voltage level input to the $\overline{WAIT}$ pin must be in the range of V_{SS} to V_{DD}. If a voltage exceeding this range is applied, not only the current dissipation increases, but also the reliability of the μPD78234 is adversely affected.

The above measures are necessary for the μPD78233. With the μPD78234, however, the measures are simpler when the address bus and address/data bus are set in the port mode.

★ **(5) A/D converter**

The current flowing into the AV_{REF1} pin can be decreased by resetting the CS bit (bit 7) of the A/D converter mode register (ADM). To reduce the current further, cut off the current supply to the AV_{REF1} pin with an external circuit.

Make sure that the AV_{DD} pin is at the same potential as V_{DD}. If the power is not supplied to the AD_{DD} pin in the STOP mode, not only will the power dissipation increase, but also the reliability will be adversely affected.

(6) D/A converter

The operating current of the D/A converter can be cut-off by setting the input level of the AV_{REF2} pin to the same level as that of the AV_{REF3} pin. Then the output level of ANOn (n = 0, 1) becomes the same level as that of the AV_{REF3}. Therefore, the AV_{REF3} pin level should be adjusted in such a manner that the current flow of the next stage will be a minimum.

No external voltage should be applied to the ANOn pin.

Application of external voltage may not only increase the operating current but may also damage or impair the reliability of the μPD78234.

16.5 Notes

(1) If an attempt is made to set the HALT mode, when the condition, under which the HALT mode is released, is satisfied, the HALT mode is not set and the next instruction is executed or the execution branches to a vector interrupt service program. To accurately set the HALT mode, clear the interrupt request before setting the HALT mode.

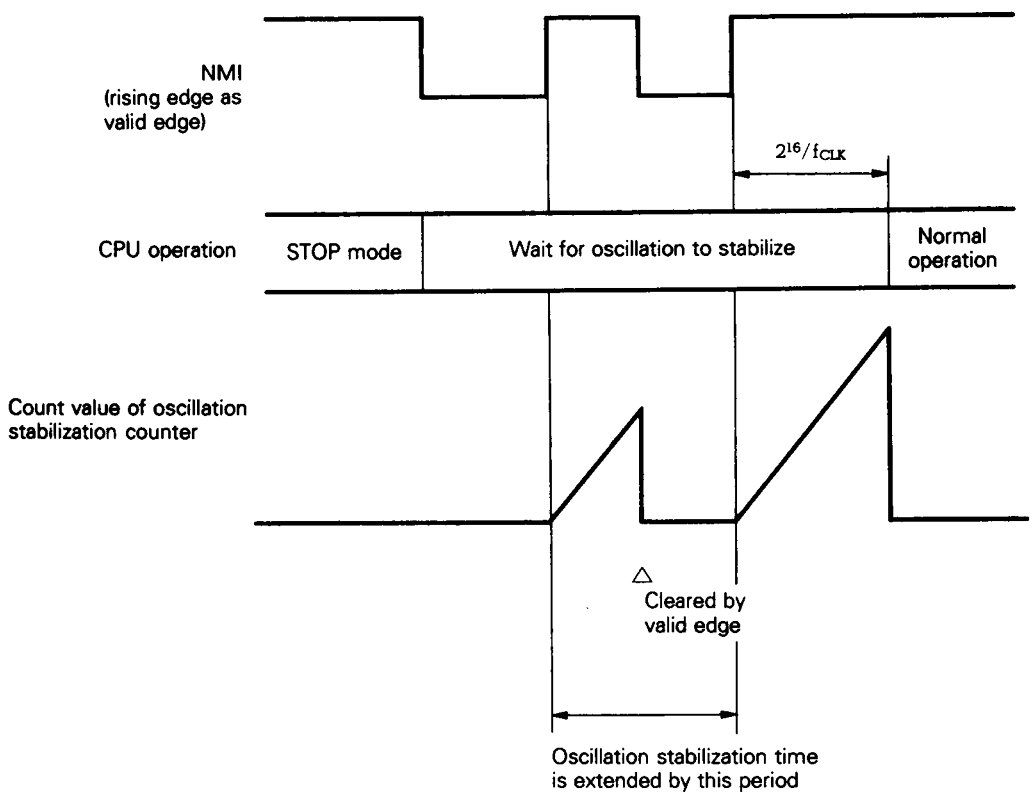
(2) When the STOP mode is set, the X1 pin is internally short-circuited to V_{SS} (GND potential) to prevent current leakage from the clock oscillator circuit; therefore, use of the STOP mode is inhibited in a system that uses an external clock.

(3) When the STOP mode is set, reset the CS bit for the A/D converter.

(4) The STOP mode is set, even when the NMI request is kept pending, when an attempt is made to set the STOP mode. To use NMI to release the STOP mode, input the NMI signal again.

(5) If a valid edge is input to the NMI pin during the oscillation stabilization time when the OW0 bit of STBC = 0, the counter that counts the oscillation stabilization time is cleared and the oscillation stabilization time is extended as follows:
Time until valid edge is input + active level period of NMI
To make sure that the microcomputer is released from the STOP mode, fix the level of the NMI pin to the inactive level during the oscillation stabilization time.

Fig. 16-9 When Oscillation Stabilization Time Is Extended



(6) Operations with the OW0 bit set to 1 cannot be emulated by the in-circuit emulator, when the STOP mode is released by NMI. Evaluate this function using μPD78P238.

CHAPTER 17 RESET FUNCTION

17.1 Reset Function

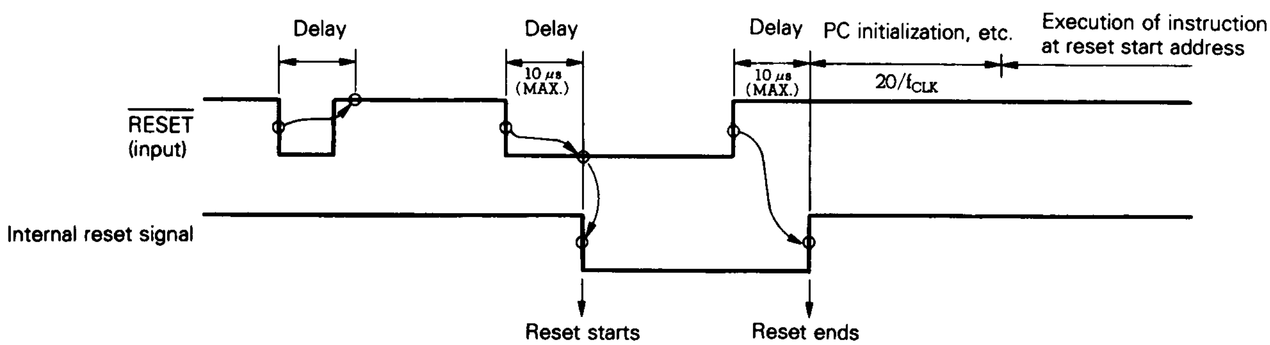
When a low-level signal is input to the $\overline{\text{RESET}}$ pin, the system is reset, and each hardware peripheral enters the status shown in Table 17-2. All the pins, except power pins, enter the high-impedance status. Table 17-1 shows the status of each pin while the RESET signal is applied and after the RESET signal has been removed.

When the $\overline{\text{RESET}}$ signal goes high, the effect of the signal is canceled, and the contents of the reset vector table at address 00000H are set in the bits 7 through 0 of the program counter (PC). The contents of the table at address 00001H are set in the bits 15 through 8 of the PC. Program execution starts from an address indicated by these contents of the PC. Therefore, execution can be started from any address.

Initialize the contents of each register by program as necessary.

The $\overline{\text{RESET}}$ input pin is provided with an analog delay noise rejecter circuit to prevent the system from malfunctioning due to noise (see **Fig. 17-1**).

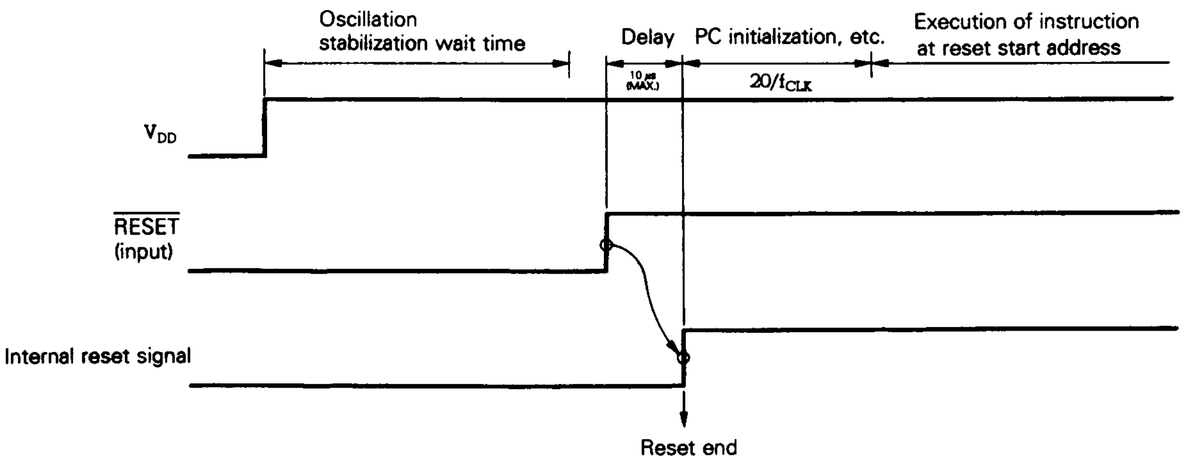
Fig. 17-1 Accepting RESET Signal ★



Remarks: f_{CLK}: system clock frequency (f_{XX}/2)

★ Hold the $\overline{RESET}$ signal in the active level when the system is to be reset on power application, until the oscillation stabilization wait time (about 30 ms; which varies depending on the oscillator used) elapses.

★ **Fig. 17-2 Reset Operation on Power Application**



Remarks: f_{CLK}: system clock frequency (f_{XX}/2)

Table 17-1 Pin Status during and after Reset

Pin name	I/O	During reset	After reset
P00-P07	Output	Hi-Z	Hi-Z
P10/PWM0-P17	I/O	Hi-Z	Hi-Z (input port mode)
P20/NMI-P27/SI	Input	Hi-Z	Hi-Z (input port)
P30/RxD-P37/TO3	I/O	Hi-Z	Hi-Z (input port mode)
P40/AD0-P47/AD7	I/O	Hi-Z	Hi-Z (input port mode)*
P50/A8-P57/A15	I/O	Hi-Z	Hi-Z (input port mode)*
P60/A16-P63/A19	Output	Hi-Z	0
P64/$\overline{RD}$, P65/$\overline{WR}$	I/O	Hi-Z	Hi-Z (input port mode)*
P66/$\overline{WAIT}$, P67/$\overline{REFRQ}$	I/O	Hi-Z	Hi-Z (input port mode)
P70/ANI0-P77/ANI7	Input	Hi-Z	Hi-Z (input port)
ASTB	Output	Hi-Z	0
ANO0, ANO1	Output	Hi-Z	Outputs input voltage of AV_{REF3} pin

***:** When the ROM-less mode is set because the MODE pin is 1, these pins constitute the address/data bus and output signals to fetch a reset vector address from address 0000H (see **Fig. 17-3(a)**).

Table 17-2 Hardware Status after Reset (1/2)

Hardware				Status after reset
Program counter (PC)				Contents of reset vector table (0000H, 0001H) are set
Stack pointer (SP)				Undefined*
Program status word (PSW)				02H
Internal RAM	Data memory			02H
	General-purpose registers (X, A, C, B, E, D, L, H)			Undefined*
Port	Ports 0, 1, 3, 4, 5			Undefined (high impedance)
	Port 6			×0H
Port mode register	PM0, PM1, PM3, PM5			FFH
	PM6			F×H
Port 3 mode control register (PMC3)				00H
Pull-up resistor option register (PUO)				00H
Real-time output port control register (RTPC)				00H
Timer/counter unit	16-bit timer/counter	Timer (TM0)		0000H
		Compare registers (CR00, CR01)		Undefined
		Capture register (CR02)		
	8-bit timer/counter	Timers (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
	One-shot pulse output control register (OSPC)			00H
PWM	PWM control register (PWMC)			05H
	PWM modulo registers (PWM0, PWM1)			Undefined
A/D converter	Mode register (ADM)			00H
	A/D conversion result register (ADCR)			Undefined
D/A converter	D/A conversion value setting registers (DACS0, DACS1)			00H

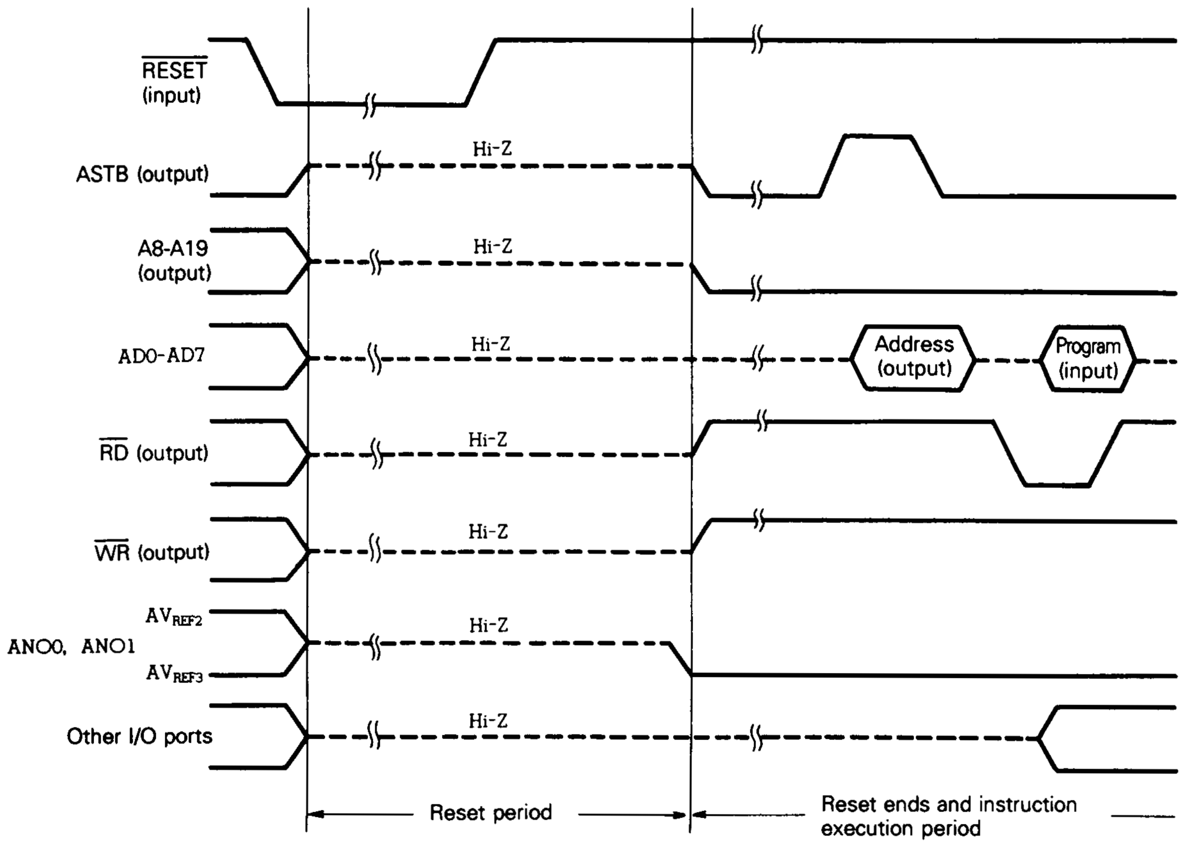
***:** When the STOP mode is released by the $\overline{\text{RESET}}$ signal, the values before the STOP mode has been set are retained.

Table 17-2 Hardware Status after Reset (2/2)

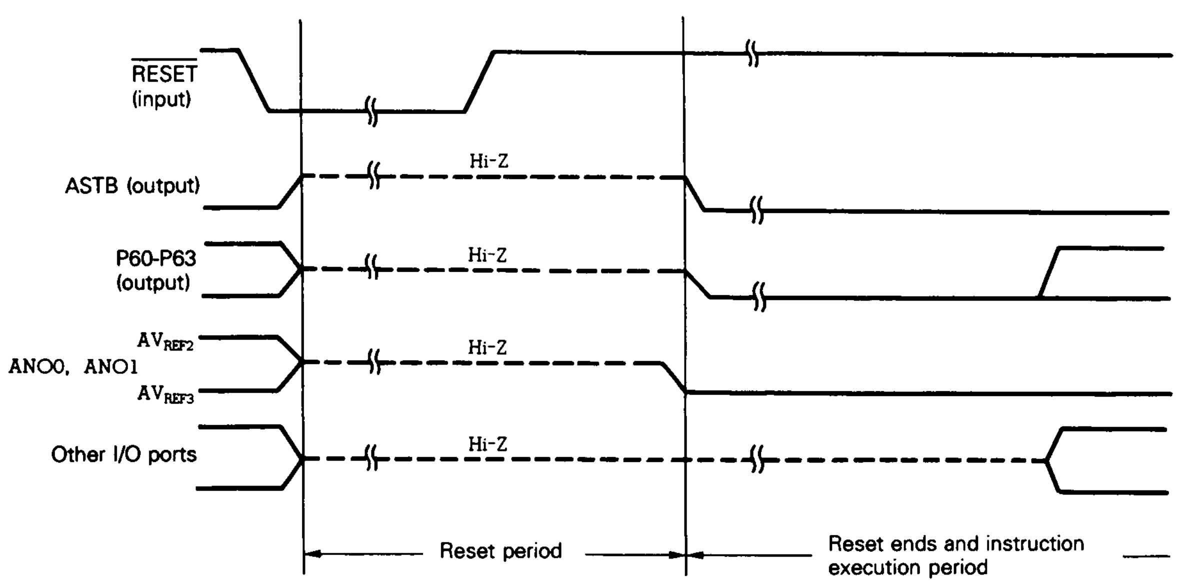
Hardware		Status after reset
Serial interface	Mode register (CSIM)	00H
	Shift register (SIO)	Undefined
	Asynchronous mode register (ASIM)	80H
	Asynchronous status register (ASIS)	00H
	Serial bus control register (SBIC)	00H
	Serial receive buffer (RXB)	Undefined
	Serial transfer shift register (TXS)	Undefined
	Baud rate generator control register (BRGC)	00H
Memory expansion mode register (MM)		20H
Programmable wait control register (PW)		80H
Refresh mode register (RFM)		00H
Interrupt	Interrupt request flag register (IF0)	0000H
	Interrupt mask register (MK0)	FFFFH
	Priority select flag register (PR0)	FFFFH
	Interrupt service mode register (ISM0)	0000H
	Interrupt status register (IST)	00H
External interrupt mode register (INTM0, INTM1)		00H
Standby control register (STBC)		0000 × 000B
Internal memory size select register (IMS)		FFH

Fig. 17-3 Timing When Reset Signal Is Input

(a) μPD78233



(b) μPD78234



17.2 Note

Keep the $\overline{\text{RESET}}$ signal low on power application, even after the supply voltage has risen to the specified level, until the oscillator stabilizes.

CHAPTER 18 APPLICATION EXAMPLES

18.1 Open-Loop Control of Stepping Motor

This section shows an example of controlling stepping motors with the real-time output function, 8-bit timer/counter 1, and macro service function of μPD78234.

Fig. 18-1 shows the functional blocks to control two stepping motors.

An interrupt signal is generated when the current value of 8-bit timer/counter 1 (TM1) coincides with the value of two compare registers (CR10 and CR11). This interrupt signal triggers the macro service function, which allows the CPU to automatically transfer table data stored in memory in advance to the compare registers and buffer registers of the real-time output port. At this time, the compare registers transfer data equivalent to interval time at which the next interrupt is to be generated.

Data to be output next from port 0 pin is transferred to the buffer registers.

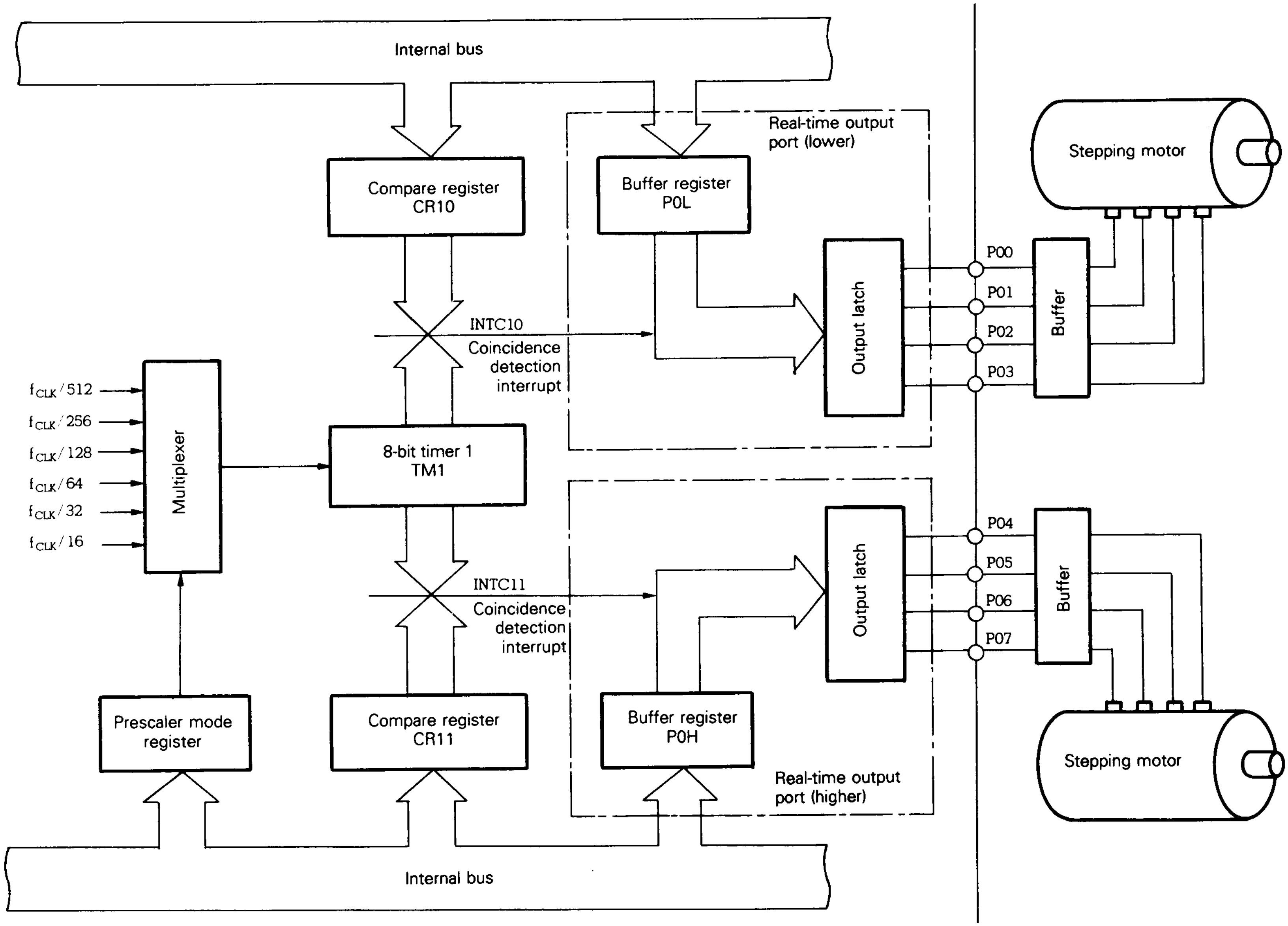
The open-loop control of stepping motors using μPD78234 brings about the following merits:

(1) The real-time output function provides accurate control (output) signals at specified time intervals.

(2) Two stepping motors can be controlled independently of each other by using one 8-bit timer/counter and two compare registers in combination, enhancing the efficiency of hardware.

(3) Complicated control operations such as acceleration and deceleration of stepping motors can be implemented without intervention by software thanks to the macro service function.

Fig. 18-1 Example of Controlling Two Stepping Motors



18.2 Serial Communication with Several Devices

Fig. 18-2 shows an example of system configured with the serial bus interface. The serial bus interface can transfer addresses (that select devices), commands, data, as well as acknowledge and busy signals by using only two lines: serial clock and serial bus lines.

To establish serial communication between μPD78234, which serves as the master device in the example below, and several devices, the master device outputs an address to the serial bus line to select a slave device with which the master device is to communicate data. The slave device, in turn, compares the address it has received against the address assigned to the slave device in advance. This comparison is carried out by software. If the received address matches the address assigned to the slave device, the slave device sends an acknowledge signal to the master device, in response to which the master device starts transferring commands or data with the slave device. Fig. 18-3 shows an example of communication by using the serial bus interface (SBI).

Fig. 18-2 Example of System Configuration with Serial Bus Interface

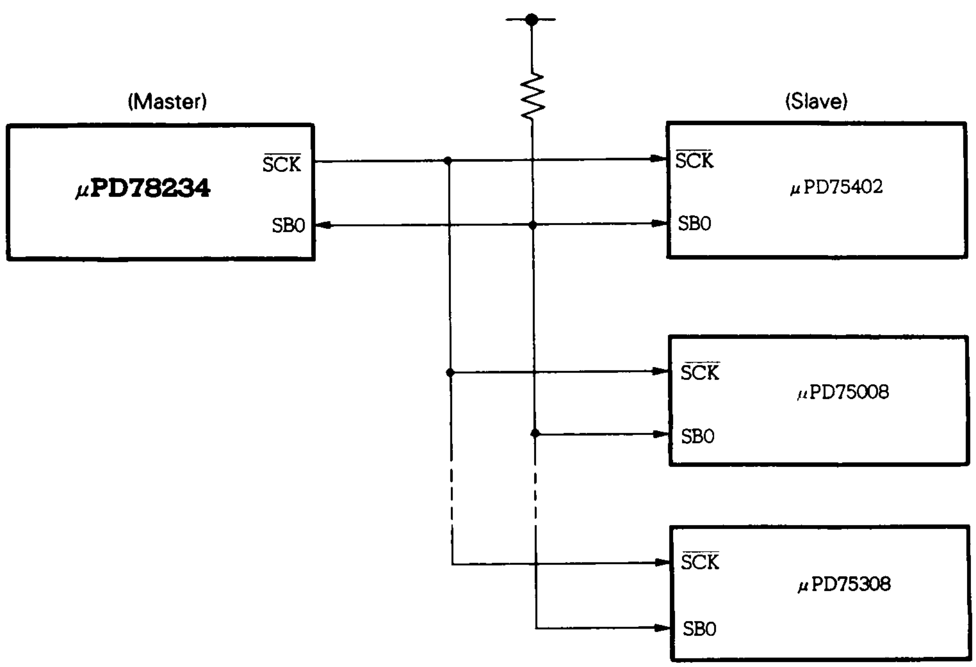


Fig. 18-3 Example of Communication with SBI

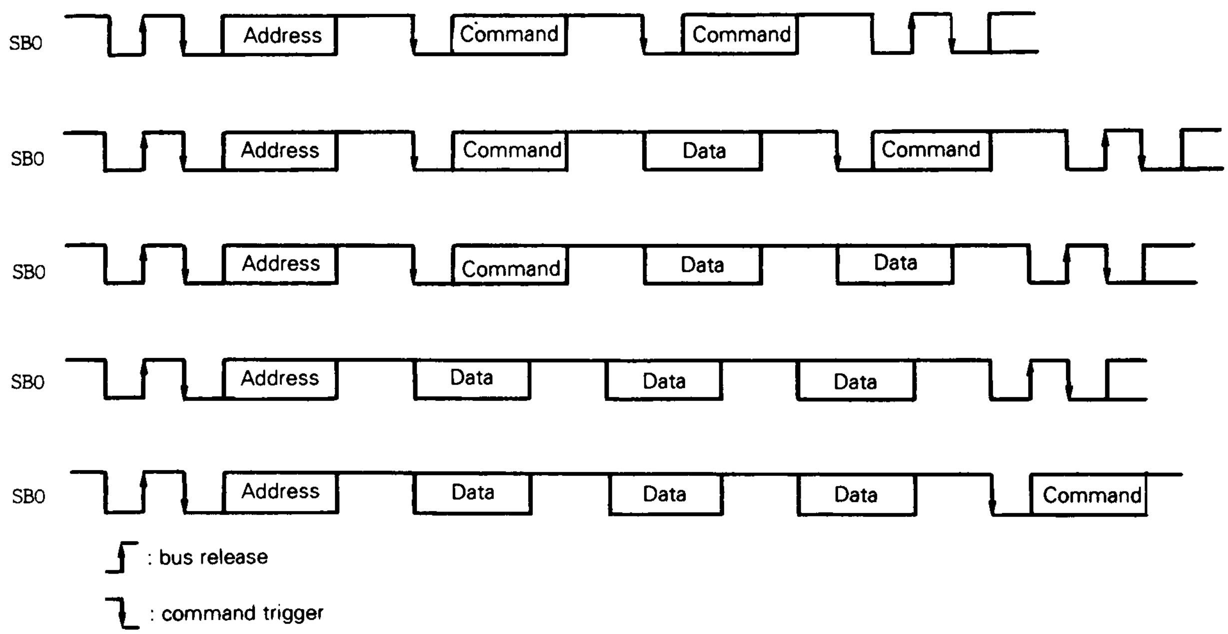
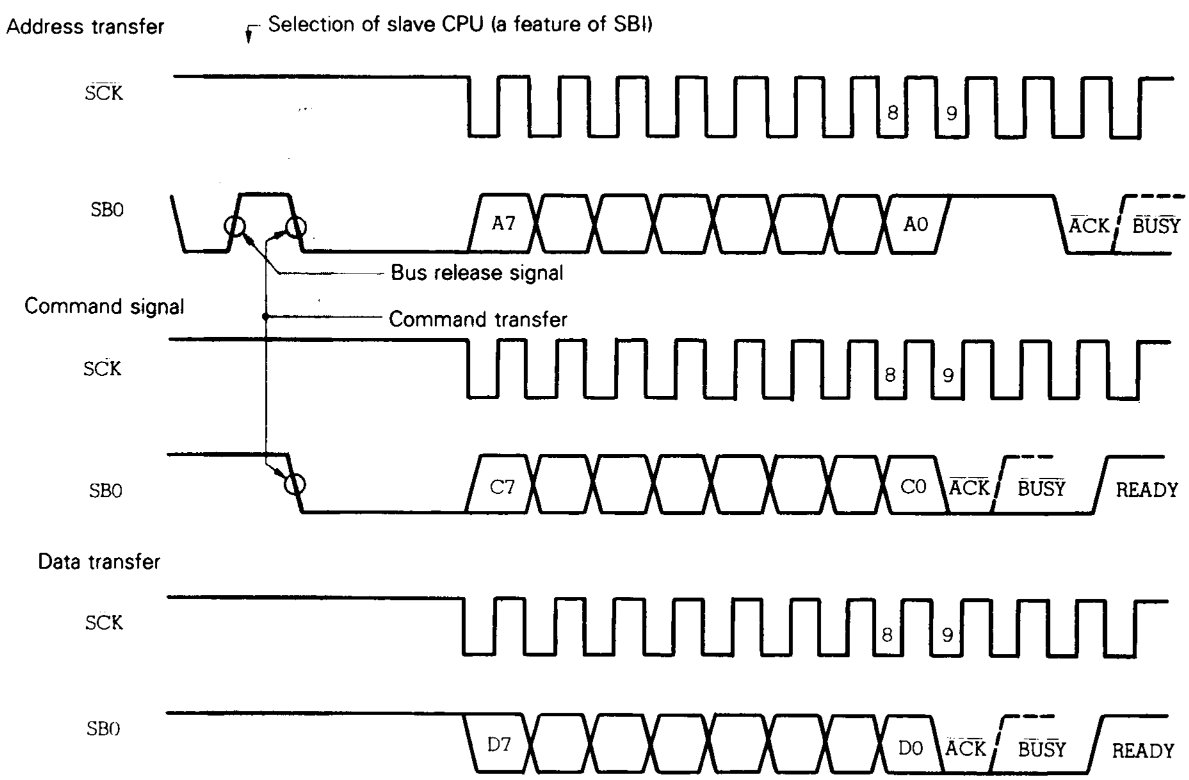


Fig. 18-4 Serial Bus Communication Timing



The internal program memory of μPD78P238 is a 32768 × 8 bit PROM to which data can be electrically written. To program this PROM, the PROM programming mode must be set by the MODE/V_{PP} and the $\overline{RESET}$ pins.

The programming characteristics are compatible with those of uPD27C256A*.

***:** The mode in which the program pulse is 100 us is not supported.

19.1 Operation Mode

μPD78P238 is set in the PROM programming mode when 5 or 12.5 V is applied to its MODE/V_{PP} pin and the $\overline{RESET}$ pin is made low. In this mode, the operation modes shown in Table 19-1 can be set by setting the $\overline{CE}$ and $\overline{OE}$ pins.

The contents of the PROM can be read when μPD78P238 is set in the read mode.

Table 19-1 PROM Programming Operation Modes

<table>
<tr><th>Pin
Mode</th><th>$\overline{RESET}$</th><th>MODE/V_{PP}</th><th>V_{DD}</th><th>$\overline{CE}$</th><th>$\overline{OE}$</th><th>D0-D7</th></tr>
<tr><td>Program write</td><td rowspan="6">L</td><td rowspan="3">+12.5 V</td><td rowspan="3">+6 V</td><td>L</td><td>H</td><td>Data input</td></tr>
<tr><td>Program verify</td><td>H</td><td>L</td><td>Data output</td></tr>
<tr><td>Program inhibit</td><td>H</td><td>H</td><td>High impedance</td></tr>
<tr><td>Read</td><td rowspan="3">+5 V</td><td rowspan="3">+5 V</td><td>L</td><td>L</td><td>Data output</td></tr>
<tr><td>Output disable</td><td>L</td><td>H</td><td>High impedance</td></tr>
<tr><td>Standby</td><td>H</td><td>L/H</td><td>High impedance</td></tr>
</table>

Caution: Making both the $\overline{CE}$ and $\overline{OE}$ pins low is inhibited when +12.5 V is applied to the MODE/V_{PP} pin and +6 V is applied to the V_{DD} pin.

19.2 Writing PROM

Data can be written to the PROM at high speeds in the following procedure:

(1) Make the $\overline{\text{RESET}}$, P17, and P60 through P63 pins low. Apply +5 V to the MODE/V_{PP} pin. Process pins not used as described in 2.3.

(2) Apply +6 V to the V_{DD} pin, and +12.5 V to the MODE/V_{PP} pin.

(3) Supply the initial address.

(4) Supply the write data.

(5) Supply a 1-ms program pulse (active low) to the $\overline{\text{CE}}$ pin.

(6) The verify mode is set. If the data has been written, go to step (8). If not, repeat steps (4) through (6). If the data cannot be written after these steps have been repeated 25 times, go to step (7).

(7) The device is defective. Stop writing.

(8) Supply the write data and a program pulse whose duration is (the number of times steps (4) to (6) have been repeated) × 3 ms (additional writing).

(9) Increment the address.

(10) Repeat steps (4) to (9) until the last address has been programmed.

Fig. 19-1 shows the timing of (2) through (8).

Fig. 19-1 PROM Write/Verify Timing

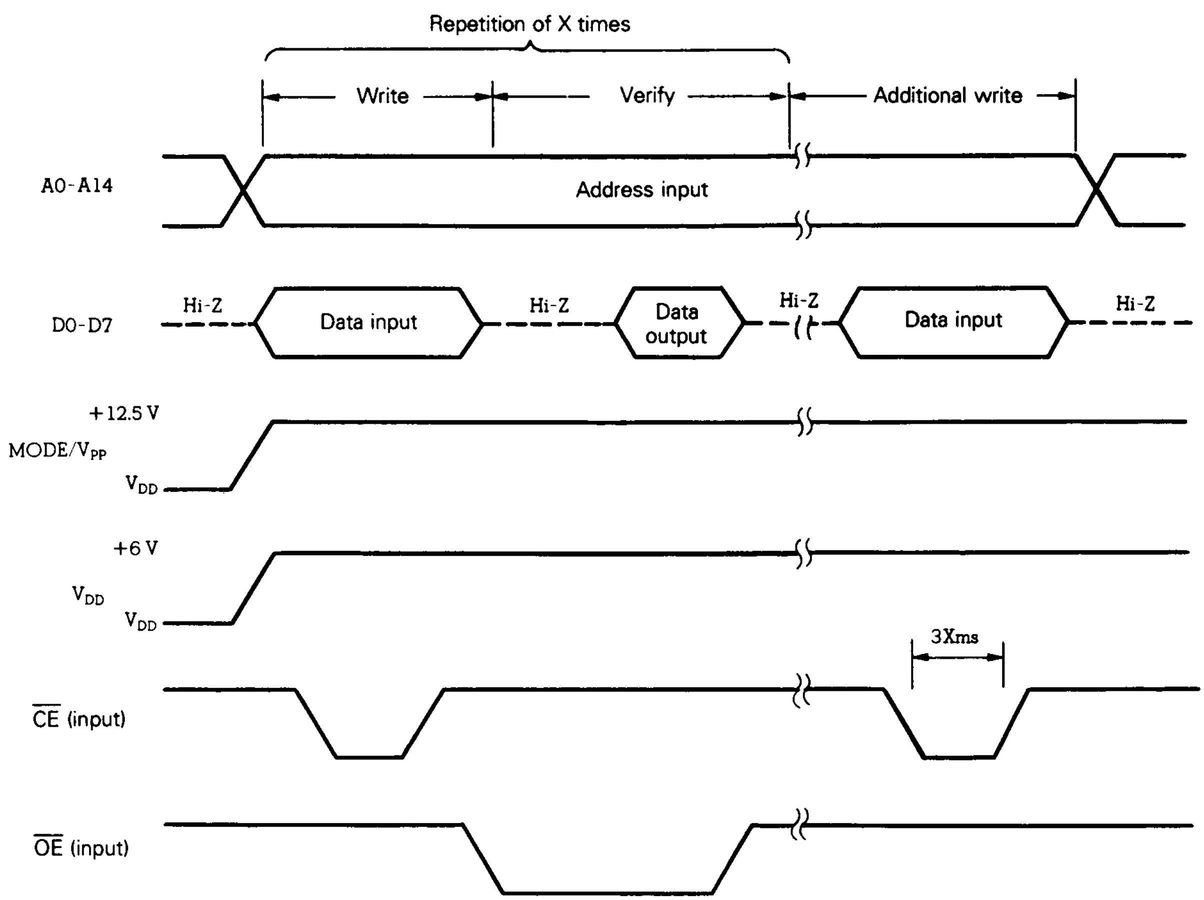
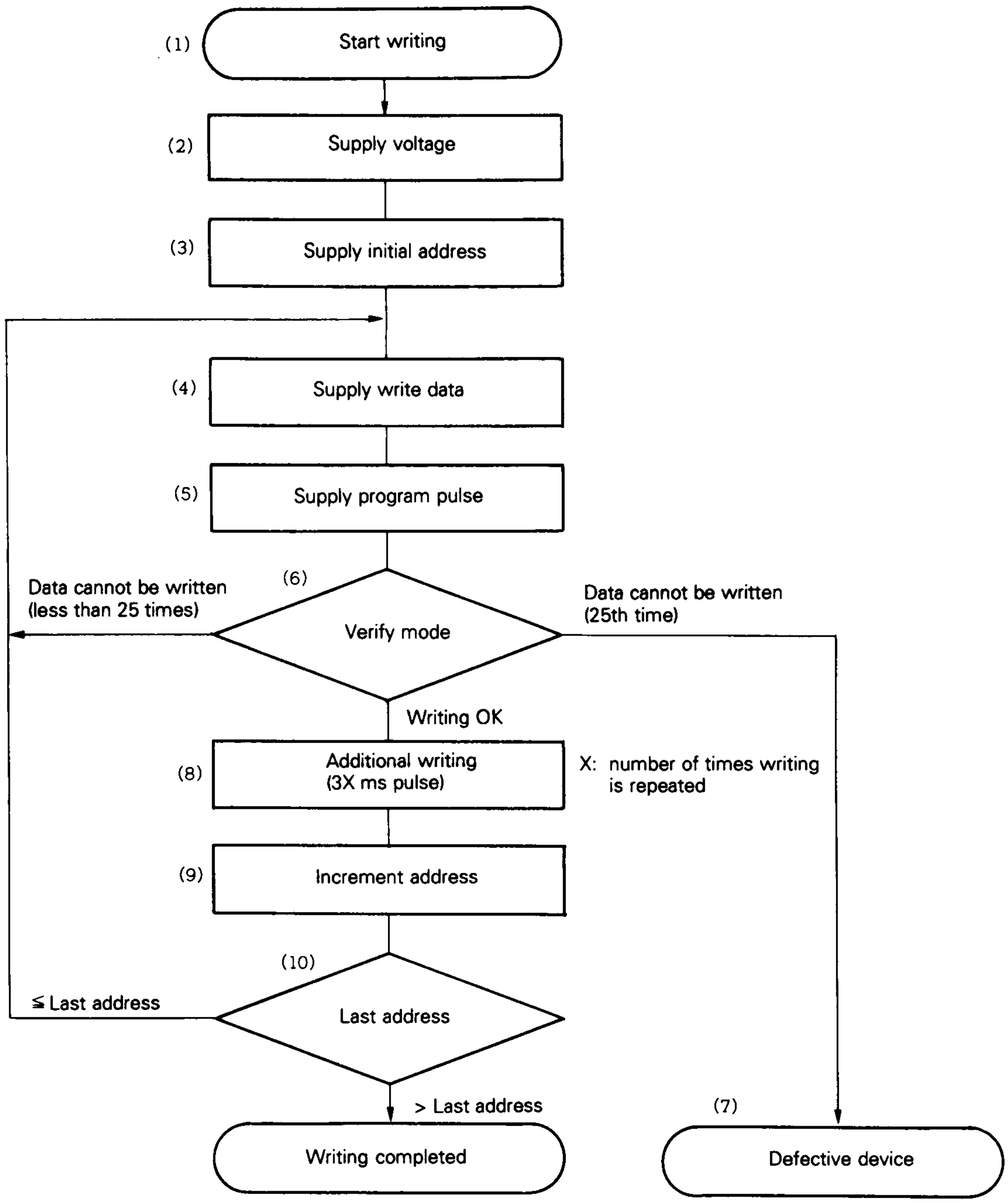


Fig. 19-2 Writing Procedure



19.3 PROM Reading Procedure

The contents of the PROM can be read out to the external data bus (D0-D7) in the following procedure:

(1) Make the $\overline{\text{RESET}}$ pin low. Supply +5 V to the MODE/V_{PP} pin. Process pins not used as described in **1.3.2**.

(2) Supply 5 V to the V_{DD} and V_{PP} pins.

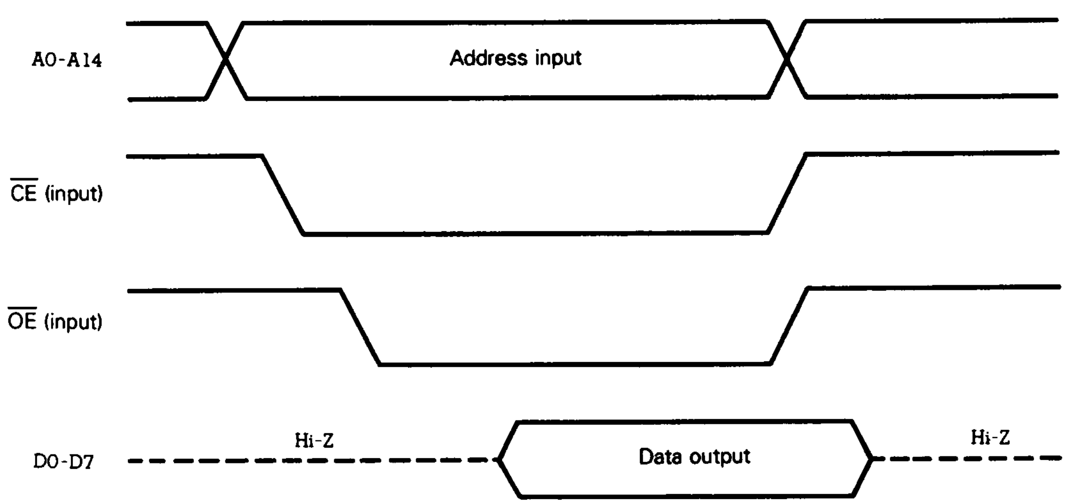
(3) Input the data of the address to be read to the A0 through A14 pins.

(4) Read mode

(5) Output data to the D0 through D7 pins.

Fig. 19-3 shows the timing of steps (2) through (5) above.

Fig. 19-3 PROM Read Timing



19.4 Note

Lowering both $\overline{CE}$ and $\overline{OE}$ is inhibited, when MODE/V_{PP} is set to +12.5 V and V_{DD} is set to +6 V.

CHAPTER 20 INSTRUCTION OPERATION

The following describes the operation of each instruction of the μPD78234 sub-series. Refer to the **INSTRUCTION PART** of the **78K/II SERIES USERS MANUAL** (IEU-1311) for details of operation, machine language (instruction code) and the number of clock states of each instruction.

20.1 Legend

20.1.1 Operand field

Describe one or more operand in the operand field of an instruction in the specified operand representation format (for details, refer to the Assembler Specifications). In the explanation in this chapter, two or more operands may be shown for an instruction, in which case select and describe one of the operands. The uppercase characters and symbols +, –, #, !, $, /, [ ], and & are keywords that must be described as they are.

Describe an appropriate numeric value or label as immediate data. When describing immediate data as a label, be sure to describe +, –, #, !, $, /, [ ], and & as they are. r and rp can be described in both functional name and absolute name.

+	: autoincrement
–	: autodecrement
#	: immediate data
!	: absolute address
$	: relative address
/	: bit inversion
[ ]	: indirect addressing
&	: subbank
r,r'	: register; function name : X, A, C, B, E, D, L, H absolute name : R0 through R7
rl	: register group 1; B, C
rp,rp'	: register pair; function name : AX, BC, DE, HL absolute name : RP0 through RP3
sfr	: special function register; P0, P1, P2, P3, P4, P5, P6, P7, P0H, P0L, RTPC, CR10, CR11, CR20, CR21, CR22, CR30, PM0, PM1, PM3, PM5, PM6, PMC3, PUO, CRC0, CRC1, CRC2, TOC, TM1, TM2, TM3, TMC0, TMC1, PRM0, PRM1, OSPC, PWMC, ADM, ADCR, DACS0, DACS1, CSIM, SBIC, SIO, ASIM, ASIS, RXB, TXS, BRGC, STBC (for special instruction only), MM, PW, RFM, IF0L, IF0H, MK0L, MK0H, PR0L, PR0H, ISM0L, ISM0H, INTM0, INTM1, IST, IMS
sfrp	: special function register pair; CR00, CR01, CR02, TM0, PWM0, PWM1, IF0, MK0, PR0, ISM0
mem	: memory addressed 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 addressed in indirect addressing group 1 mode ; [DE], [HL]
saddr, saddr'	:	memory addressed in short direct addressing saddr' mode; immediate data FE20H to FF1FH or label
saddrp	:	memory addressed in short direct addressing pair mode; immediate data FE20H to FF1EH or label
addr16	:	16-bit address; immediate data 0000H to FFFFH or label
addr11	:	11-bit address; immediate data 800H to FFFH or label
addr5	:	5-bit address; immediate data 40H to 7EH 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 bits shifted; 3-bit immediate data (0 to 7)
RBn	:	register bank; RB0 to RB3

20.1.2 Operation field

A	: A register; 8-bit accumulator
X	: X register
B	: B register
C	: C register
D	: D register
E	: E register
H	: H register
L	: L register
R0 to R7	: registers 0 to 7 (absolute name)
AX	: register pair (AX); 16-bit accumulator
BC	: register pair (BC)
DE	: register pair (DE)
HL	: register pair (HL)
RP0 to RP3	: register pairs 0 to 3 (absolute name)
PC	: program counter
SP	: stack pointer
PSW	: program status word
CY	: carry flag
AC	: auxiliary carry flag
Z	: zero flag
RBS1 to RBS0	: register bank select flags
IE	: interrupt request enable flag
STBC	: standby control register
jdisp8	: signed 8-bit data (displacement: –128 to +127)
( )	: memory contents indicated by address or register contents in ( )
××H	: hexadecimal number
$\times_H$, $\times_L$	: higher 8 bits and lower 8 bits of 16-bit register pair

20.1.3 Flag field

(Blank)	: No change
0	: Cleared to 0
1	: Set to 1
×	: Set or cleared depending on the result
R	: Value previously saved is restored

20.2 Operation Lists

(1) 8-bit data transfer operation: MOV, XCH

Mnemonic	Operand	Bytes	Operation	Flag Z	Flag AC	Flag CY
MOV	r, #byte	2	r ← byte			
	saddr, #byte	3	(saddr) ← byte			
	sfr, #byte	3	sfr ← byte			
	r, r'	2	r ← r'			
	A, r	1	A ← r			
	A, saddr	2	A ← (saddr)			
	saddr, A	2	(saddr) ← A			
	saddr, saddr'	3	(saddr) ← (saddr')			
	A, sfr	2	A ← sfr			
	sfr, A	2	sfr ← A			
	A, mem	1-4	A ← (mem)			
	A, &mem	2-5	A ← (&mem)			
	mem, A	1-4	(mem) ← A			
	&mem, A	2-5	(&mem) ← A			
	A, !addr16	4	A ← (!addr16)			
	A, & !addr16	5	A ← (& !addr16)			
	!addr16, A	4	(!addr16) ← A			
	& !addr16, A	5	(& !addr16) ← A			
	PSW, #byte	3	PSW ← byte	×	×	×
	PSW, A	2	PSW ← A	×	×	×
	A, PSW	2	A ← PSW			
XCH	A, r	1	A ↔ r			
	r, r'	2	r ↔ r'			
	A, mem	2-4	A ↔ (mem)			
	A, &mem	3-5	A ↔ (&mem)			
	A, saddr	2	A ↔ (saddr)			
	A, sfr	3	A ↔ sfr			
	saddr, saddr'	3	(saddr) ↔ (saddr')			

(2) 16-bit data transfer instructions: MOVW

Mnemonic	Operand	Bytes	Operation	Flag 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 operation instructions: ADD, ADDC, SUB, SUBC, AND, OR, XOR, CMP

Mnemonic	Operand	Bytes	Operation	Flag		
				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	×	×	×
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	×	×	×

Mnemonic	Operand	Bytes	Operation	Flag		
				Z	AC	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)	×		
XOR	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)	×		
CMP	A, #byte	2	A–byte	×	×	×
	saddr, #byte	3	(saddr)–byte	×	×	×
	sfr, #byte	4	sfr–byte	×	×	×
	r, r′	2	r–r′	×	×	×
	A, saddr	2	A–(saddr)	×	×	×
	A, sfr	3	A–sfr	×	×	×
	saddr, saddr′	3	(saddr)–(saddr′)	×	×	×
	A, mem	2-4	A–(mem)	×	×	×
	A, &mem	3-5	A–(&mem)	×	×	×

(4) 16-bit arithmetic operation instructions: ADDW, SUBW, CMPW

Mnemonic	Operand	Bytes	Operation	Flag Z	Flag AC	Flag CY
ADDW	AX, #word	3	AX, CY ← AX+word	×	×	×
	AX, rp	2	AX, CY ← AX+rp	×	×	×
	AX, saddrp	2	AX, CY ← AX+(saddrp)	×	×	×
	AX, sfrp	3	AX, CY ← AX+sfrp	×	×	×
SUBW	AX, #word	3	AX, CY ← AX–word	×	×	×
	AX, rp	2	AX, CY ← AX–rp	×	×	×
	AX, saddrp	2	AX, CY ← AX–(saddrp)	×	×	×
	AX, sfrp	3	AX, CY ← AX–sfrp	×	×	×
CMPW	AX, #word	3	AX–word	×	×	×
	AX, rp	2	AX–rp	×	×	×
	AX, saddrp	2	AX–(saddrp)	×	×	×
	AX, sfrp	3	AX–sfrp	×	×	×

(5) Multiplication/division instructions: MULU, DIVUW

Mnemonic	Operand	Bytes	Operation	Flag Z	Flag AC	Flag CY
MULU	r	2	AX ← A×r			
DIVUW	r	2	AX(quotient), r(remainder) ← AX÷r When r=0, r ← X, AX ← 0FFFFH			

(6) Increment/decrement instructions: INC, DEC, INCW, DECW

Mnemonic	Operand	Bytes	Operation	Flag Z	Flag AC	Flag CY
INC	r	1	r ← r+1	×	×	
	saddr	2	(saddr) ← (saddr)+1	×	×	
DEC	r	1	r ← r–1	×	×	
	saddr	2	(saddr) ← (saddr)–1	×	×	
INCW	rp	1	rp ← rp+1			
DECW	rp	1	rp ← rp–1			

(7) Shift/rotate instructions: ROR, ROL, RORC, ROLC, SHR, SHL, SHRW, SHLW, ROR4, ROL4

Mnemonic	Operand	Bytes	Operation	Flag Z	Flag AC	Flag CY
ROR	r, n	2	(CY, $r_7 \leftarrow r_0$, $r_{m-1} \leftarrow r_m$) × n times, n=0-7			×
ROL	r, n	2	(CY, $r_0 \leftarrow r_7$, $r_{m+1} \leftarrow r_m$) × n times, n=0-7			×
RORC	r, n	2	(CY $\leftarrow r_0$, $r_7 \leftarrow$ CY, $r_{m-1} \leftarrow r_m$) × n times, n=0-7			×
ROLC	r, n	2	(CY $\leftarrow r_7$, $r_0 \leftarrow$ CY, $r_{m+1} \leftarrow r_m$) × n times, n=0-7			×
SHR	r, n	2	(CY $\leftarrow r_0$, $r_7 \leftarrow 0$, $r_{m-1} \leftarrow r_m$) × n times, n=0-7	×	0	×
SHL	r, n	2	(CY $\leftarrow r_7$, $r_0 \leftarrow 0$, $r_{m+1} \leftarrow r_m$) × n times, n=0-7	×	0	×
SHRW	rp, n	2	(CY $\leftarrow rp_0$, $rp_{15} \leftarrow 0$, $rp_{m-1} \leftarrow rp_m$) × n times, n=0-7	×	0	×
SHLW	rp, n	2	(CY $\leftarrow rp_{15}$, $rp_0 \leftarrow 0$, $rp_{m+1} \leftarrow rp_m$) × n times, n=0-7	×	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	$A_{3-0} \leftarrow (\&mem1)_{7-4}$, $(\&mem1)_{3-2} \leftarrow A_{3-0}$, $(\&mem1)_{7-4} \leftarrow (\&mem1)_{3-0}$			

(8) BCD adjustment instructions: ADJBA, ADJBS

Mnemonic	Operand	Bytes	Operation	Flag Z	Flag AC	Flag CY
ADJBA		1	Decimal Adjust Accumulator after Addition	×	×	×
ADJBS		1	Decimal Adjust Accumulator after Subtract	×	×	×

(9) Bit manipulation instructions: MOV1, AND1, OR1, XOR1, SET1, CLR1, NOT1

Mnemonic	Operand	Bytes	Operation	Flag		
				Z	AC	CY
MOV1	CY, saddr.bit	3	CY ← (saddr.bit)			×
	CY, sfr.bit	3	CY ← sfr.bit			×
	CY, A.bit	2	CY ← A.bit			×
	CY, X.bit	2	CY ← X.bit			×
	CY, PSW.bit	2	CY ← PSW.bit			×
	saddr.bit, CY	3	(saddr.bit) ← CY			
	sfr.bit, CY	3	sfr.bit ← CY			
	A.bit, CY	2	A.bit ← CY			
	X.bit, CY	2	X.bit ← CY			
	PSW.bit, CY	2	PSW.bit ← CY	×	×	
AND1	CY, saddr.bit	3	CY ← CY∧(saddr.bit)			×
	CY, /saddr.bit	3	CY ← CY∧$\overline{\text{(saddr.bit)}}$			×
	CY, sfr.bit	3	CY ← CY∧sfr.bit			×
	CY, /sfr.bit	3	CY ← CY∧$\overline{\text{sfr.bit}}$			×
	CY, A.bit	2	CY ← CY∧A.bit			×
	CY, /A.bit	2	CY ← CY∧$\overline{\text{A.bit}}$			×
	CY, X.bit	2	CY ← CY∧X.bit			×
	CY, /X.bit	2	CY ← CY∧$\overline{\text{X.bit}}$			×
	CY, /PSW.bit	2	CY ← CY∧PSW.bit			×
OR1	CY, saddr.bit	3	CY ← CY∨(saddr.bit)			×
	CY, /saddr.bit	3	CY ← CY∨$\overline{\text{(saddr.bit)}}$			×
	CY, sfr.bit	3	CY ← CY∨sfr.bit			×
	CY, /sfr.bit	3	CY ← CY∨$\overline{\text{sfr.bit}}$			×
	CY, A.bit	2	CY ← CY∨A.bit			×
	CY, /A.bit	2	CY ← CY∨$\overline{\text{A.bit}}$			×
	CY, X.bit	2	CY ← CY∨X.bit			×
	CY, /X.bit	2	CY ← CY∨$\overline{\text{X.bit}}$			×
	CY, PSW.bit	2	CY ← CY∨PSW.bit			×
	CY, /PSW.bit	2	CY ← CY∨$\overline{\text{PSW.bit}}$			×
XOR1	CY, saddr.bit	3	CY ← CY⊻(saddr.bit)			×
	CY, sfr.bit	3	CY ← CY⊻sfr.bit			×
	CY, A.bit	2	CY ← CY⊻A.bit			×
	CY, X.bit	2	CY ← CY⊻X.bit			×
	CY, PSW.bit	2	CY ← CY⊻PSW.bit			×

Mnemonic	Operand	Bytes	Operation	Flag		
				Z	AC	CY
SET1	saddr.bit	2	(saddr.bit) ← 1			
	sfr.bit	3	sfr.bit ← 1			
	A.bit	2	A.bit ← 1			
	X.bit	2	X.bit ← 1			
	PSW.bit	2	PSW.bit ← 1	×	×	×
	CY	1	CY ← 1			1
CLR1	saddr.bit	2	(saddr.bit) ← 0			
	sfr.bit	3	sfr.bit ← 0			
	A.bit	2	A.bit ← 0			
	X.bit	2	X.bit ← 0			
	PSW.bit	2	PSW.bit ← 0	×	×	×
	CY	1	CY ← 0			0
NOT1	saddr.bit	3	(saddr.bit) ← $\overline{\text{(saddr.bit)}}$			
	sfr.bit	3	sfr.bit ← $\overline{\text{sfr.bit}}$			
	A.bit	2	A.bit ← $\overline{\text{A.bit}}$			
	X.bit	2	X.bit ← $\overline{\text{X.bit}}$			
	PSW.bit	2	PSW.bit ← $\overline{\text{PSW.bit}}$	×	×	×
	CY	1	CY ← $\overline{\text{CY}}$			×

(10) Call/return instructions: CALL, CALLF, CALLT, BRK, RET, RETI, RETB

Mnemonic	Operand	Bytes	Operation	Flag		
				Z	AC	CY
CALL	!addr16	3	(SP–1) ← $(PC+3)_H$, (SP–2) ← $(PC+3)_L$, PC ← addr16, SP ← SP–2			
	rp	2	(SP–1) ← $(PC+2)_H$, (SP–2) ← $(PC+2)_L$, PC_H, ← rp_H, PC_L, ← rp_L, SP ← SP–2			
CALLF	!addr11	2	(SP–1) ← $(PC+2)_H$, (SP–2) ← $(PC+2)_L$, PC_{15-11} ← 00001, PC_{10-0} ← addr11, SP ← SP–2			
CALLT	[addr5]	1	(SP–1) ← $(PC+1)_H$, (SP–2) ← $(PC+1)_L$, PC_H ← (0000000001, addr5+1), PC_L ← (0000000001, addr5), SP ← SP–2			
BRK		1	(SP–1) ← PSW, (SP–2) ← $(PC+1)_H$ (SP–3) ← $(PC+1)_L$, PC_L ← (003EH), PC_H ← (003FH), SP ← SP–3, IE ← 0			
RET		1	PC_L, ← (SP), PC_H ← (SP+1), SP ← SP+2			
RETI		1	PC_L ← (SP), PC_H ← (SP+1), PSW ← (SP+2), SP ← SP+3, NMIS ← 0	R	R	R
RETB		1	PC_L ← (SP), PC_H ← (SP+1), PSW ← (SP+2), SP ← SP+3	R	R	R

(11) Stack manipulation instructions: PUSH, POP, MOVW, INCW, DECW

Mnemonic	Operand	Bytes	Operation	Flag		
				Z	AC	CY
PUSH	PSW	1	(SP–1) ← PSW, SP ← SP–1			
	sfr	2	(SP–1) ← sfr, SP ← SP–1			
	rp	1	(SP–1) ← rp_H, (SP–2) ← rp_L, SP ← SP–2			
POP	PSW	1	PSW ← (SP), SP ← SP+1	R	R	R
	sfr	2	sfr ← (SP), SP ← SP+1			
	rp	1	rp_L ← (SP), rp_H ← (SP+1), SP ← SP+2			
MOVW	SP, #word	4	SP ← word			
	SP, AX	2	SP ← AX			
	AX, SP	2	AX ← SP			
INCW	SP	2	SP ← SP+1			
DECW	SP	2	SP ← SP–1			

(12) Unconditional branch instructions: BR

Mnemonic	Operand	Bytes	Operation	Flag		
				Z	AC	CY
BR	!addr16	3	PC ← addr16			
	rp	2	PC_H ← rp_H, PC_L ← rp_L			
	$addr16	2	PC ← PC+2+jdisp8			

(13) Conditional branch instructions: BC, BL, BNC, BNL, BZ, BE, BNZ, BNE, BT, BF, BTCLR, DBNZ

Mnemonic	Operand	Bytes	Operation	Flag		
				Z	AC	CY
BC **BL**	$addr16	2	PC ← PC+2+jdisp8 if CY=1			
BNC **BNL**	$addr16	2	PC ← PC+2+jdisp8 if CY=0			
BZ **BE**	$addr16	2	PC ← PC+2+jdisp8 if Z=1			
BNZ **BNE**	$addr16	2	PC ← PC+2+jdisp8 if Z=0			
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 PSW.bit	×	×	×
DBNZ	rl, $addr16	2	rl ← rl–1, then PC PC+2+jdisp8 if rl≠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	Bytes	Operation	Flag		
				Z	AC	CY
MOV	STBC, #byte	4	STBC ← byte			
SEL	RBn	2	RBS1–0 ← n, n=0–3			
NOP		1	No Operation			
EI		1	IE ← 1(Enable Interrupt)			
DI		1	IE ← 0(Disable Interrupt)			

20.3 Classification of Instructions by Addressing Mode

(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, DBNZ, PUSH, POP

Table 20-1 8-bit Instructions

Second operand / First operand	#byte	A	r r'	saddr saddr'	sfr	mem	&mem	!addr16	&!addr16	PSW	n	None*2
A	ADD*1		MOV XCH	MOV XCH ADD*1	MOV XCH ADD*1	MOV XCH ADD*1	MOV XCH ADD*1	MOV	MOV	MOV		
r	MOV		MOV XCH ADD*1								ROR RORC ROL ROLC SHR SHL	MULU DIVUW DEC INC
rl												DBNZ
saddr	MOV ADD*1	MOV		MOV XCH ADD*1								DEC INC DBNZ
sfr	MOV ADD*1	MOV										POP PUSH
mem &mem		MOV										
mem1 &mem1												ROR4 ROL4
!addr16 &!addr16		MOV										
PSW	MOV	MOV										POP PUSH
STBC	MOV											

***1:** ADDC, SUB, SUBC, AND, OR , XOR, and CMP are the same as ADD.

***2:** Either the second operand is not provided, or the second operand is not an operand address.

(2) 16-bit instructions

MOVW, ADDW, SUBW, CMPW, INCW, DECW, SHRW, SHLW, PUSH, POP

Table 20-2 16-bit Instructions

First operand \ Second operand	#word	AX	rp rp'	saddrp	sfrp	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, BTCLR

Table 20-3 16-bit Instructions

First operand \ Second operand	CY	A.bit	/A.bit	X.bit	/X.bit	saddr.bit	/saddr.bit	sfr.bit	/sfr.bit	PSW.bit	/PSW.bit	None*
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

***:** Either the seond operand is not provided, or the second operand is not an operand address.

(4) Call and branch instructions

CALL, CALLF, CALLT, BR, BC, BT, BF, BTCLR, DBNZ, BL, BNC, BNL, BZ, BE, BNZ, BNE

Table 20-4 Call and Branch Instructions

Operand of instruction address	$addr16	!addr16	rp	!addr11	[addr5]
Basic instruction	BR BC*	CALL BR	CALL BR	CALLF	CALLT
Compound instruction	BT BF BTCLR DBNZ				

***:** 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, SEL

APPENDIX A 78K/II SERIES PRODUCT LIST

A list of the 78K/II series products is presented on the following pages.

Series name		μPD78214 Sub-Series			μPD78218A Sub-Series		μPD78224 Sub-Series	
Item	Product name	μPD78212	μPD78213	μPD78214 (μPD78P214)	μPD78217A	μPD78218A (μPD78P218A)	μPD78220	μPD78224 (μPD78P224)
Basic instructions		65 (common instructions for 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 (no. of clocks)		When stack area is in internal dual port RAM: 5 or 7 Any other case: 7 or 9			When stack area is in internal dual port RAM: 6 Any other case: 8		When stack area is in internal dual port port RAM: 5 or 7 Any other case: 7 or 9	
Operating temperature range and operating voltage range		Other than μPD78P218A: –40 to +85°C, V_{DD}=+5 V±10% μPD78P218A: –40°C to +85°C, V_{DD}=+5 V±0.3 V					–40 to +85°C, V_{DD}=+5 V±5% –10 to +70°C, V_{DD}=+5 V±10%	
General-purpose registers		8 bits × 8 × 4 banks						
Bank register		P6 and PM6					P6 only	
Internal memory	ROM	8K bytes	None	16K bytes	None	32K bytes	None	16K bytes
Internal memory	EEPROM	None						
Internal memory	RAM	384 bytes	512 bytes		1024 bytes		640 bytes	
Memory space		Program memory space: 64K bytes, data memory space: 1M bytes						
I/O pins	Input	14					8	
I/O pins	Output	12					12	20
I/O pins	I/O	28	10	28	10	28	25	35
I/O pins	Total	54	36	54	36	54	45	63
Ancillary function pins*	w/pull-up resistor	34	16	34	16	34	None	
Ancillary function pins*	LED direct drive output	16	0	16	0	16	8	
Ancillary function pins*	Transistor direct drive	8					None	
I/O pins	P0	8-bit output port						
I/O pins	P1	–					8-bit I/O port	
I/O pins	P2	8-bit input port						
I/O pins	P3	8-bit I/O port						
I/O pins	P4	8-bit I/O port	–	8-bit I/O port	–	8-bit I/O port	–	8-bit I/O port
I/O pins	P5	8-bit I/O port	–	8-bit I/O port	–	8-bit I/O port	–	8-bit output port
I/O pins	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
I/O pins	P7	6-bit input port					7-bit I/O port	

***:** 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 (common instructions for all 78K/II series products)					
500 ns	333 ns	500 ns	333 ns	500 ns	333 ns
When stack area is in internal dual port RAM: 6 other case: 8					
–40°C 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 bytes	None	32K bytes (32K/16K bytes*)	None	16K bytes
None				512 bytes	
640 bytes		1024 bytes	1024 bytes (1024/640 bytes*)	512 bytes	
Program memory space: 64K bytes, data memory space: 1M bytes					
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 + 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
8-bit input port				6-bit input port	

***:** Set through software

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 input voltage	Only pins selected by ANI0-ANI2 of ADM register. Always 0V to AV_{REF} pin voltage.			Pins subject to A/D conversion. 0 V to AV_{REF} pin voltage 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 from 0FFD0H-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)

<table>
<tr><td colspan="4">μPD78234 Sub-Series</td><td colspan="2">μPD78244 Sub-Series</td></tr>
<tr><td>μPD78233</td><td>μPD78234</td><td>μPD78237</td><td>μPD78238
(μPD78P238)</td><td>μPD78243</td><td>μPD78244</td></tr>
<tr><td colspan="4">12 bits × 2</td><td colspan="2">None</td></tr>
<tr><td colspan="6">None</td></tr>
<tr><td colspan="6">8 bits × 8</td></tr>
<tr><td colspan="4">Selected freely</td><td colspan="2">Selected according to operating frequency</td></tr>
<tr><td colspan="4">3.4 V to V_{DD}</td><td colspan="2">3.6 V to V_{DD}</td></tr>
<tr><td colspan="6">Only pins subject to A/D conversion.
0 V to AV_{REF} pin voltage during A/D conversion</td></tr>
<tr><td colspan="4">8 bits × 2</td><td colspan="2">None</td></tr>
<tr><td colspan="6">1</td></tr>
<tr><td colspan="6">3</td></tr>
<tr><td colspan="6">4</td></tr>
<tr><td colspan="6">Provided</td></tr>
<tr><td colspan="6">Provided</td></tr>
<tr><td colspan="6">7</td></tr>
<tr><td colspan="6">16 bytes from 0FFD0H-0FFDFH</td></tr>
<tr><td colspan="6">1 channel</td></tr>
<tr><td colspan="6">1 channel (SBI)</td></tr>
<tr><td colspan="6">Provided (shared with timer/counter 3)</td></tr>
<tr><td colspan="6">Provided</td></tr>
<tr><td colspan="6">Provided</td></tr>
<tr><td colspan="6">4 bits × 2 or 8 bits × 1</td></tr>
</table>

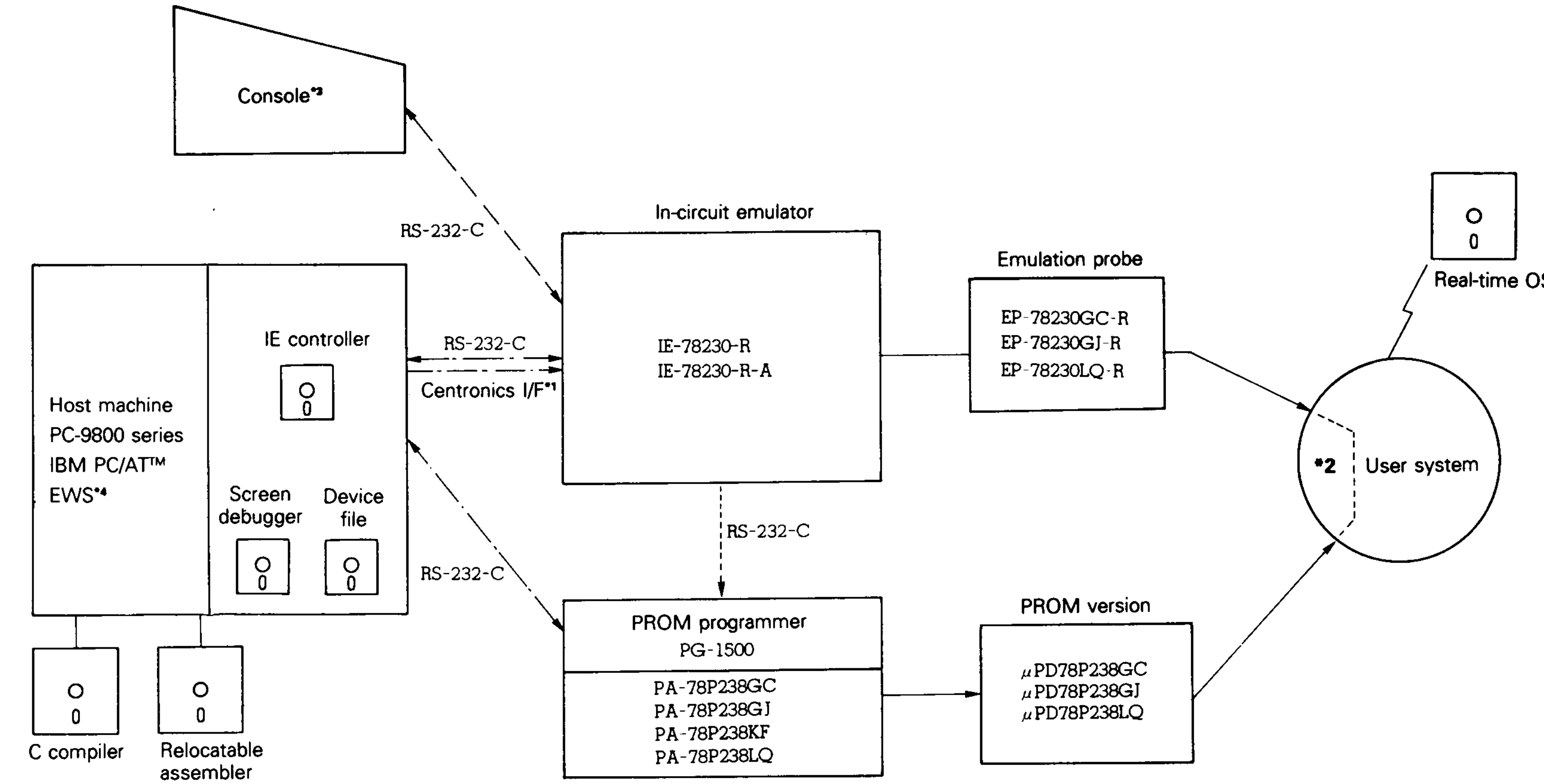
Series name		μPD78214 Sub-Series			μPD78218A Sub-Series		μPD78224 Sub-Series	
Item	Product name	μPD78212	μPD78213	μPD78214 (μPD78P214)	μPD78217A	μPD78218A (μPD78P218A)	μPD78220	μPD78224 (μPD78P224)
Interrupt		2 levels (programmable), vector/macro service						
	External	7					8	
	Internal	12					9	
	Interrupts that can use macro service	15					6	
	Bits of macro service counter	8 bits only			8/16 bits selectable (except tyep A)		8 bits only	
	Incrementing MPD, MPT of macro service Type C	Increments lower 8 bits only (higher bits not affected)			Increments 16 bits		Increments lower 8 bits only (higher bits not affected)	
	Macro service	μPD78214 sub-series and μPD78224 sub-series are identical. μPD78218A, μPD78234, and μPD78244 sub-series are identical. Execution time of these execution time differs depending on mode. Refer to User's Manual of each model.						
	Constraints on memory-to-SFR data transfer by is D0H to DFH macro service type A	Occurs when transfer data is D0H to DFH			Occurs when transfer source buffer (memory) address is 0FED0H to 0FEDFH		Occurs when transfer data is D0H toDFH	
Standby function		HALT/STOP mode						
	Oscillation stanbilization time on releas-ing STOP mode	Fixed			Two times selectable		Fixed	
Pseudo SRAM refresh function		Provided (refresh pulse width: $1/f_{CLK}$)					Provided (refresh pulse width: $1.5/f_{CLK}$)	
Restrictions on memory access		None					FC80H-FDFFH cannot be accessed, while refresh function	
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		• 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 w/window: μPD78P214 only			• 64-pin plastic shrink DIP (750 mil) • 64-pin plastic QFP (14 × 14 mm) • 64-pin ceramic shrink DIP w/window: μPD78P218A only		• 84-pin plastic QFJ • 94-pin plastic QFP (20 × 20 mm)	

(3/3)

<table>
<tr><td colspan="4">μPD78234 Sub-Series</td><td colspan="2">μPD78244 Sub-Series</td></tr>
<tr><td>μPD78233</td><td>μPD78234</td><td>μPD78237</td><td>μPD78238
(μPD78P238)</td><td>μPD78243</td><td>μPD78244</td></tr>
<tr><td colspan="6">2 levels (programmable), vector/macro service</td></tr>
<tr><td colspan="6">7</td></tr>
<tr><td colspan="4">12</td><td colspan="2">14</td></tr>
<tr><td colspan="6">15</td></tr>
<tr><td colspan="6">8/16 bits selectable (except type A)</td></tr>
<tr><td colspan="6">Increments 16 bits</td></tr>
<tr><td colspan="6">μPD78214 sub-series and μPD78224 sub-series are identical.
μPD78218A, μPD78234, and μPD78244 sub-series are identical.
Execution time of these execution time differs depending on mode. Refer to User’s Manual of each model.</td></tr>
<tr><td colspan="4">Occurs when transfer data is D0H to DFH</td><td colspan="2">Occurs when transfer source buffer (memory) address is 0FED0H to 0FEDFH</td></tr>
<tr><td colspan="6">HALT/STOP mode</td></tr>
<tr><td colspan="6">Two times selectable</td></tr>
<tr><td colspan="6">Provided (refresh pulse width: $1/f_{CLK}$)</td></tr>
<tr><td colspan="6">None</td></tr>
<tr><td>–</td><td>MODE pin = high level</td><td>–</td><td>MODE pin = high level (can not set)</td><td>–</td><td>$\overline{EA}$ pin = low level</td></tr>
<tr><td colspan="4">• 84-pin plastic QFJ
• 80-pin plastic QFP (14 × 14 mm)
• 94-pin plastic QFP (20 × 20 mm)
• 94-pin ceramic WQFN: μPD78P238 only</td><td colspan="2">• 64-pin plastic shrink DIP (750 mil)
• 64-pin plastic QFP (14 × 14 mm)</td></tr>
</table>

APPENDIX B DEVELOPMENT TOOLS

The development tools in the following pages are readily available for development of systems using μPD78234 sub-series.



—·—·— : When connected to the host machine

— — — — : When connected to console with IE used in stand-alone mode

***1:** Used when high-speed file transfer (down load)

***2:** EV-9200GC-80, EV-9200G-94

***3:** Only when IE-78230-R is used

***4:** EWS can be either the HP9000/300 series, SUN4/3900, or EWS-4800/200

B.1 Hardware (1/2)

IE-78230-R-A	This in-circuit emulator is a reinforced model of IE-78230-R and can be used with any models in the μPD78234 sub-series*1. It can be used when a PC-9800 series computer or a IBM PC/AT computer is used as the host machine. Optional screen debugger and device file are necessary, and in combination with these, the in-circuit emulator can perform debugging at level of source program in C language or structured assembly language. Features such as simultaneous tracing of data access and program fetch and C0 coverage enhance debugging and program test efficiencies. If you already have IE-78230-R, it can be upgraded to IE-78230-R-A by optional board (IE-78200-R-BK).
IE-78230-R*2	This is an in-circuit emulator for development the μPD78234 sub-series application system and debugging. Debugging is carried out by connecting the in-circuit emulator to a host machine or a console. When the emulator is connected to a host machine, symbolic debugging and object file transfer with the host machine can be performed, substantially enhancing the debugging efficiency. The in-circuit emulator is provided with two channels of RS-232-C serial interfaces and can also be connected to a PROM writer such as PG-1500.
IE-78230-R-EM IE-78200-R-EM*2 IE-78200-R-BK	This board is for upgrading the in-circuit emulator for the 75X and 78K series to the IE-78230-R or IE-78230-R-A. Refer to **B.3** for details.
EP-78230LQ-R	Emulation probe for μPD78233LQ, μPD78234LQ-xxx, μPD78237LQ, μPD78238LQ-xxx, and μPD78P238LQ.
EP-78230GJ-R	Emulation probe for μPD78233GJ-5BG, μPD78234GJ-xxx-5BG, μPD78237GJ-5BG, μPD78238GJ-xxx-5BG, μPD78P238GJ-5BG, μPD78234GJ(A)-xxx-5BG, and μPD78238GJ(A)-xxx-5BG.
EP-78230GC-R	Emulation probe for μPD78233GC-3B9 μPD78234GC-xxx-3B9, μPD78237GC-3B9, μPD78238GC-xxx-3B9, μPD78P238GC-3B9, μPD78234GC(A)-xxx-3B9, and μPD78238GC(A)-xxx-3B9.

***1:** μPD78233, 78234, 78237, 78238, 78P238, 78234(A), 78238(A)

***2:** IE-78230-R and IE-78200-R-EM are not produced any more.
Use IE-78230-R-A and IE-78200-R-BK instead.

B.1 Hardware (2/2)

EV-9200G-94*	Socket to be mounted on a user system development for μPD78233GJ-5BG, μPD78234GJ-xxx-5BG, μPD78237GJ-5BG, μPD78238GJ-xxx-5BG or μPD78P238GJ-5BG. Used with EP-78230GJ.
EV-9200GC-80*	Socket to be mounted on a user system developed for μPD78233GC-3B9, μPD78234GC-xxx-3B9, μPD78237GC-3B9, μPD78238-xxx-3B9 or μPD78P238GC-3B9. Used with EP-78230GC.
EV-9900	Jig used for removing μPD78P238KF instead from EV-9200G-94. You can use a pair of tweezers instead of the jig. However, using this jig makes the removal easier. If you use two jigs, the removal is even easier.
PG-1500	This is a PROM programmer that can program single-chip microcomputers with PROM in stand-alone mode or under control of a host machine, when connected with an accessory board and an optional programmer adapter. This PROM programmer can program representative PROMs from 256K-bit models to 4M-bit models.
PA-78P238LQ	PROM programmer adapter for μPD78P238LQ and is used in combination with PG-1500.
PA-78P238GJ	PROM programmer adapter for μPD78P238GJ-5BG and is used in combination with PG-1500.
PA-78P238GC	PROM programmer adapter for μPD78P238GC-3B9 and is used in combination with PG-1500.
PA-78P238KF	PROM programmer adapter for μPD78P238KF, used in combination with PG-1500.

***:** Place your order for EV-9200G-94 and EV-9200G-80 in units of 5.
EP-78230GJ-R and EP-78230GC-R are provided with one EV-9200G-94 or EV-9200GC-80.

B.2 Software

B.2.1 Language processing software (1/3)

<table>
<tr>
<td>78K/II series relocatable assembler (RA78K/II)</td>
<td>A relocatable assembler, that can be used commonly for the 78K/II series products. Since this assembler is provided with macro functions, it enhances the development efficiency. A structured assembler, that can explicitly describe the program control structure, is also supplied, so that the program productivity and maintainability can be improved. The relocatable assembler consists of the following programs:
<table>
<tr><td>Structured assembler preprocessor
(program name: ST78K2)</td><td>Converts the format of a source program, described in the assembly language of the structured assembler, into that in which the source can be input to the relocatable assembler.</td></tr>
<tr><td>Relocatable assembler
(program name: RA78K2)</td><td>A program thatconverts the source program, described in an assembly language, into machine language codes and that creates relocatable object module files.</td></tr>
<tr><td>Linker
(program name: LK78K2)</td><td>Links the object module file, created by the relocatable assembler, with the library file, to determine the absolute addresses for the program and create the load module file.</td></tr>
<tr><td>Object converter
(program name: OC78K2)</td><td>Converts the format for the load format file, created by the linker, into that in which the file can be down-loaded to an in-circuit emulator or PROM programmer.</td></tr>
<tr><td>Librarian
(program name: LB78K2)</td><td>Links the object module files, output by the relocatable assembler, to create one library file. Also updates the library file.</td></tr>
<tr><td>List converter
(program name: LCNV78K2)</td><td>Creates an assemble list, with absolute values, from the assemble list file output by the relocatable assembler by using the object module file and load module file.</td></tr>
</table>
</td>
</tr>
</table>

Language processing software (2/3)

	Host machine	OS	Supply media	Part number
78K/II series relocatable assembler (RA78K/II)				
	PC-9800 series	MS-DOS™ (Ver. 3.30 to Ver. 5.00A*3)	8"2D*1	μS5A1RA78K2
			5"2HD	μS5A10RA78K2
			3.5"2HD	μS5A13RA78K2
★	IBM PC/AT or compatible equipment	Refer to **B.2.4**.	5"2D*2	μS7B11RA78K2
			5"2HC	μS7B10RA78K2
			3.5"2HC	μS7B13RA78K2
	HP9000 Series 300™	HP-UX™ (rel.7.05B)	Cartridge tape (QIC-24)	μS3H15RA78K2
	SPARCstation™	Sun OS™ (rel.4.1.1)		μS3K15RA78K2
	EWS-4800 series™ (RISC)	EWS-UX/V™ (rel.4.0)		μS3M15RA78K2

78K/II series C compiler (CC78K/II)	This is a C compiler that can be commonly used for 78K/II series. The language specifications conform to ANSI, and the program can be contained in ROM. The special function register manipulation function, bit manipulation function, variables which use short direct addressing, and the interrupt control function, are provided so that effective programming is possible, and higher object efficiency can be expected. In addition, a start up routine sample program and the standard function object library are provided. When using this compiler, the 78K/II series relocatable assembler (RA78K/II) is necessary.			
	Host machine	OS	Supply media	Part number
	PC-9800 series	MS-DOS™ (Ver. 3.30 to Ver. 5.00A*3)	5"2HD	μS5A10CC78K2
			3.5"2HD	μS5A13CC78K2
★	IBM PC/AT or compatible equipment	Refer to **B.2.4**.	5"2D*2	μS7B11CC78K2
			5"2HC	μS7B10CC78K2
			3.5"2HC	μS7B13CC78K2
	HP9000 Series 300	HP-UX (rel.7.05B)	Cartridge tape (QIC-24)	μS3H15CC78K2
	SPARCstation	Sun OS (rel.4.1.1)		μS3K15CC78K2
	EWS-4800 series (RISC)	EWS-UX/V (rel.4.0)		μS3M15CC78K2

***1:** The 8-inch 2D model has been superseded by the 5-inch 2HD or 3.5-inch 2HD models. The 5-inch 2HD model will be supplied for those who have already purchased an 8-inch 2D model, when the product is upgraded in the future.

***2:** The 5-inch 2D model has been superseded by the 5-inch 2HC models. The 5-inch 2HC model will be supplied for those who have already purchased a 5-inch 2D model, when the product is upgraded in the future.

***3:** Ver. 5.00/5.00A has a task swap function, but this function cannot be used with this software.

Language processing software (3/3)

78K/II series C compiler library source file (CC78K/II-L)	Source program of library supplied with CC78K/II. Necessary for remodeling library (in accordance with user specifications)			
	Host machine	OS	Supply media	Part number
	PC-9800 series	MS-DOS (Ver. 3.30 to Ver. 5.00A*)	5"2HD	μS5A10CC78K2-L
			3.5"2HD	μS5A13CC78K2-L
	IBM PC/AT or compatible equipment	Refer to **B.2.4**.	5"2HC	μS7B10CC78K2-L
			3.5"2HC	μS7B13CC78K2-L ★
	HP9000 Series 300	HP-UX (rel.7.05B)	Cartridge tape (QIC-24)	μS3H15RA78K2-L
	SPARCstation	Sun OS (rel.4.1.1)		μS3K15RA78K2-L
	EWS-4800 series (RISC)	EWS-UX/V (rel.4.0)		μS3M15RA78K2-L

***:** Ver. 5.00/5.00A has a task swap function, but this function cannot be used with this software.

B.2.2 Software for in-circuit emulator

Screen debugger (SD78K/II)	This program controls the in-circuit emulator for the 78K/II series, when used in combination with the device file (DF78230). This program can be used with an in-circuit emulator upgraded to IE-78230-R-A or equivalent and with a PC-9800 series computer or IBM PC/AT computer as the host machine. This program debugs source programs written in C, structured assembly, or assembly language, and divides the screen of the host machine for simultaneous display of various information, to enhance debugging efficiency.

Host machine	OS	Supply media	Part number
PC-9800 series	MS-DOS (Ver. 3.30 to Ver. 5.00A*1)	5"2HD	μS5A10SD78K2
		3.5"2HD	μS5A13SD78K2
★ IBM PC/AT or compatible equipment	Refer to **B.2.4**.	5"2HC	μS7B10SD78K2
		3.5"2HC	μS7B13SD78K2

Device file (DF78230)	Used in combination with the screen debugger (SD78K/II) to debug the μPD78234 sub-series

Host machine	OS	Supply media	Part number
PC-9800 series	MS-DOS (Ver. 3.30 to Ver. 5.00A*1)	5"2HD	μS5A10DF78230
		3.5"2HD	μS5A13DF78230
★ IBM PC/AT or compatible equipment	Refer to **B.2.4**.	5"2HC	μS7B10DF78230
		3.5"2HC	μS7B13DF78230

In-circuit emulator control program*4 (IE78230)	A program that controls IE-78230-R from a host machine. This program can automatically execute commands to enhance the debugging efficiency. The following control programs are available for IE-78230-R:

Emulator	Host machine	OS	Supply media	Ordering code
IE-78230-R IE-78230-R-EM	PC-9800 series	MS-DOS (Ver. 3.10 to Ver. 5.00A*1)	8"2D*2	μS5A1IE78230
			5"2HD	μS5A10IE78230
			3.5"2HD	μS5A13IE78230
	★ IBM PC/AT or compatible equipment	Refer to **B.2.4**.	5"2D*3	μS7B11IE78230
			5"2HC	μS7B10IE78230
			3.5"2HC	μS7B13IE78230

***1:** Ver. 5.00/5.00A has a task swap function, but this function cannot be used with this software.

2: The 8-inch 2D model has been superseded by the 5-inch 2HD or 3.5-inch 2HD models. The 5-inch 2HD model will be supplied for those who have already purchased an 8-inch 2D model, when the product is upgraded in the future.

3: The 5-inch 2D model has been superseded by the 5-inch 2HC model. The 5-inch 2HC model will be supplied for those who have already purchased an 5-inch 2D model, when the product is upgraded in the future.

4: This in-circuit emulator control program cannot be used with the IE-78230-R-A. Also, an in-circuit emulator cannot be upgraded to be equivalent to the IE-78230-R-A. Purchase this control program only if you possess the IE-78230-R or equivalent (the IE-78230-R cannot now be purchased).

B.2.3 Software for PROM programmer

PG-1500 controller	This program connects PG-1500 to a host machine with a serial and parallel interfaces to control PG-1500 from the host machine.			
	Host machine	OS	Supply media	Part number
	PC-9800 series	MS-DOS (Ver. 3.10 to Ver. 5.00A*1)	5"2HD	μS5A10PG1500
			3.5"2HD	μS5A13PG1500
	IBM PC/AT or compatible equipment	Refer to **B.2.4**.	5"2D*2	μS7B11PG1500
			5"2HC	μS7B10PG1500 ★
			3.5"2HC	μS7B13PG1500

***1:** Ver. 5.00/5.00A has a task swap function, but this function cannot be used with this software.

***2:** The 5-inch 2D model has been superseded by the 5-inch 2HC model. The 5-inch 2HC model will be supplied for those who have already purchased an 5-inch 2D model, when the product is upgraded in the future.

B.2.4 OS for IBM PC ★

The following are supported as IBM PC-use OS

OS	Version
PC DOS™	Ver. 3.1 to Ver. 6.1
Windows™*1	Ver. 3.1
MS-DOS	Ver. 5.0 to Ver. 6.0 5.0/V*2
IBM DOS™	J5.02/V*2

***1:** In the case of fuzzy logic creation tools, PC DOS and Windows are used in combination.

2: Only supports English-language mode.

Caution Ver. 500/5.00A has a task swap function, but this function cannot be used with this software.

B.3 Upgrading Other In-Circuit Emulators to IE-78230-R

If you already have an in-circuit emulator for the 78K series or 75X series, it can be used as the in-circuit emulator for the 78K/II series if an internal board is exchanged with an optional board.

Note, however, that the control program corresponding to the upgraded version of the in-circuit emulator is necessary.

B.3.1 Upgrading to IE-78230-R-A

Your Emulator	IE Group No.	Necessary Board	Remarks
IE-78240-R-A IE-78140-R	1	IE-78230-R-EM	———
IE-78230-R*	2	IE-78200-R-BK	———
IE-78112-R* IE-78210-R* IE-78220-R* IE-78310-R* IE-78310A-R	3	IE-78200-R-BK IE-78230-R-EM	The high-speed down-loading function cannot be used. If you have an in-circuit emulator in Group 1, 2, or 4, upgrading system based on that in-circuit emulator is recommended. If you have an in-circuit emulator in group 1, IE-78200-R-BK is not necessary. (IE-78200-R-BK is provided in in-circuit emulator in Group 1 IE).
IE-75000-R IE-75001-R IE-78000-R IE-78130-R IE-78240-R IE-78320-R* IE-78327-R IE-78330-R IE-78350-R IE-78600-R*	4	IE-78200-R-BK IE-78230-R-EM	If you have Group 1 IE, IE-78200-R-BK is not necessary (IE-78200-R-BK is provided in Group 1 IE).

***:** Production was terminated, and no available.

B.3.2 Upgrading system to IE-78230-R

Your Emulator	IE Group No.	Necessary Board	Remarks
IE-78112-R* IE-78210-R* IE-78220-R*	1	IE-78230-R-EM	The high-speed down-loading function cannot be used. If you have Group 4 IE, upgrading system based on that in-circuit emulator is recommended.
IE-78130-R IE-78240-R*	2	IE-78230-R-EM	———
IE-78310-R* IE-78310A-R	3	IE-78200-R-EM* IE-78230-R-EM	The high-speed down-loading function cannot be used. If you have Group 1 IE, IE-78200-R-EM is not necessary (IE-78200-R-EM is provided in Group 1 IE).
IE-75000-R IE-75001-R IE-78000-R IE-78320-R* IE-78327-R IE-78330-R IE-78350-R IE-78600-R*	4	IE-78200-R-EM* IE-78230-R-EM	If you have Group 1 IE, IE-78200-R-EM is not necessary (IE-78200-R-EM is provided in Group 1 IE).
IE-78140-R IE-78240-R-A	5	IE-78200-R-EM* IE-78230-R-EM	Upgrading system to IE-78230-R-A is recommended.

***:** Production was terminated, and no available.

C.1 Real-Time OS

<table>
<tr><td rowspan="10">Real-time OS (RX78K/II)</td><td colspan="4">The RX78K/II is designed to provide a multi-task environment in the feild of control appliction where real-time operation is required. With the RX78K/II, idle CPU time can be used for other processing so as to increase the overall system throughout.
In the RX78K/II, 31 system calls conforming to the uITRON specifications are provided.
The RX78K/II package is provided with a tool (configurator) to create the RX78K/II nucleus and two or more information tables.
When this is used, RAM of 1 Kbytes or more is required*1.</td></tr>
<tr><td>Host machine</td><td>OS</td><td>Supply media</td><td>Part number</td></tr>
<tr><td rowspan="2">PC-9800 series</td><td rowspan="2">MS-DOS™ (Ver. 3.30 to Ver. 5.00A*2)</td><td>5"2HD</td><td>μS5A10RX78217</td></tr>
<tr><td>3.5"2HD</td><td>μS5A13RX78217</td></tr>
<tr><td rowspan="2">IBM PC/AT or compatible equipment</td><td rowspan="2">Refer to B.2.4.</td><td>5"2HC</td><td>μS7B10RX78217</td></tr>
<tr><td>3.5"2HC</td><td>μS7B13RA78217</td></tr>
<tr><td>HP9000 Series 300</td><td>HP-UX (rel.7.05B)</td><td rowspan="3">Cartridge tape (QIC-24)</td><td>μS3H15RX78217</td></tr>
<tr><td>SPARCstation</td><td>Sun OS (rel.4.1.1)</td><td>μS3K15RX78217</td></tr>
<tr><td>EWS-4800 series (RISC)</td><td>EWS-UX/V (rel.4.0)</td><td>μS3M15RX78217</td></tr>
</table>

***1:** Target devices: μPD78237, 78238, 78P238

***2:** Ver. 5.00/5.00A has a task swap function, but this function cannot be used with this software.

Caution: If you want to buy the RX78K/II, you must fill in the application form in advance and then conclude the Assent to Use contract.

Remarks: To use the RX78K/II real-time OS, the RA78K/II assembler package (optional) is necessary.

C.2 Fuzzy Inference Development Support System

Fuzzy knowledge data creation tool (FE9000)	This program supports input/edition (edit) and evaluation (simulation) of fuzzy knowledge data (fuzzy rule and membership function).			
	Host machine	OS	Supply media	Part number
	PC-9800 series	MS-DOS™ (Ver. 3.30 to Ver. 5.00A*)	5"2HD	μS5A10FE9000
			3.5"2HD	μS5A13FE9000
★	IBM PC/AT or compatible equipment	Refer to **B.2.4**.	5"2HC	μS7B10FE9200
			3.5"2HC	μS7B13FE9200
Translator (FT9080)	This program translates the fuzzy knowledge data obtained by using the fuzzy knowledge data creation tool into an assembler source program for the RA78K/II.			
	Host machine	OS	Supply media	Part number
	PC-9800 series	MS-DOS™ (Ver. 3.30 to Ver. 5.00A*)	5"2HD	μS5A10FT9080
			3.5"2HD	μS5A13FT9080
★	IBM PC/AT or compatible equipment	Refer to **B.2.4**.	5"2HC	μS7B10FT9085
			3.5"2HC	μS7B13FT9085
Fuzzy inference module (FI78K/II)	This program executes the fuzzy inference. Link this program to the fuzzy knowledge data translated by the translator in order to execute fuzzy inference.			
	Host machine	OS	Supply media	Part number
	PC-9800 series	MS-DOS™ (Ver. 3.30 to Ver. 5.00A*)	5"2HD	μS5A10FI78K2
			3.5"2HD	μS5A13FI78K2
★	IBM PC/AT or compatible equipment	Refer to **B.2.4**.	5"2HC	μS7B10FI78K2
			3.5"2HC	μS7B13FI78K2
Fuzzy inference debugger (FD78K/II)	This is support software to evaluate and adjust fuzzy inference data at the hardware level by using an in-circuit emulator.			
	Host machine	OS	Supply media	Part number
	PC-9800 series	MS-DOS™ (Ver. 3.30 to Ver. 5.00A*)	5"2HD	μS5A10FD78K2
			3.5"2HD	μS5A13FD78K2
★	IBM PC/AT or compatible equipment	Refer to **B.2.4**.	5"2HC	μS7B10FD78K2
			3.5"2HC	μS7B13FD78K2

***:** Ver. 5.00/5.00A has a task swap function, but this function cannot be used with this software.

APPENDIX D REGISTER INDEX

D.1 Register Index (Alphabetical order)

APPENDIX E INDEX

E.1 Alphabetical Glossary

[C]

[D]

[R]

[S]

[T]

E.2 Symbols

[A]

[F]

[H]

[I]

[L]

[M]

[N]

[O]

[T]

[U]

[V]

[W]

[X]

[Z]