NEC

Preliminary User's Manual

μPD784225, 784225Y SUBSERIES

16-/8-Bit Single Chip Microcontrollers

Hardware

μPD784224 μPD784224Y
μPD784225 μPD784225Y
μPD78F4225 μPD78F4225Y

Document No. U12697EJ1V0UM00 (1st Edition)
Date Published November 1997 N

[MEMO]

NOTES FOR CMOS DEVICES

① PRECAUTION AGAINST ESD FOR SEMICONDUCTORS

Note:

Strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it once, when it has occurred. Environmental control must be adequate. When it is dry, humidifier should be used. It is recommended to avoid using insulators that easily build static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work bench and floor should be grounded. The operator should be grounded using wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with semiconductor devices on it.

② HANDLING OF UNUSED INPUT PINS FOR CMOS

Note:

No connection for CMOS device inputs can be cause of malfunction. If no connection is provided to the input pins, it is possible that an internal input level may be generated due to noise, etc., hence causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using a pull-up or pull-down circuitry. Each unused pin should be connected to V_{DD} or GND with a resistor, if it is considered to have a possibility of being an output pin. All handling related to the unused pins must be judged device by device and related specifications governing the devices.

③ STATUS BEFORE INITIALIZATION OF MOS DEVICES

Note:

Power-on does not necessarily define initial status of MOS device. Production process of MOS does not define the initial operation status of the device. Immediately after the power source is turned ON, the devices with reset function have not yet been initialized. Hence, power-on does not guarantee out-pin levels, I/O settings or contents of registers. Device is not initialized until the reset signal is received. Reset operation must be executed immediately after power-on for devices having reset function.

The export of these products from Japan is regulated by the Japanese government. The export of some or all of these products may be prohibited without governmental license. To export or re-export some or all of these products from a country other than Japan may also be prohibited without a license from that country. Please call an NEC sales representative.

Licence not needed	: μPD78F4225, 78F4225Y
The customer must judge the need for licence	: μPD784224, 784225, 784224Y, 784225Y

The application circuits and their parameters are for reference only and are not intended for use in actual design-ins.

Purchase of NEC I^2C components conveys a license under the Philips I^2C Patent Rights to use these components in an I^2C system, provided that the system conforms to the I^2C Standard Specification as defined by Philips.

The information contained in this document is being issued in advance of the production cycle for the device. The parameters for the device may change before final production or NEC Corporation, at its own discretion, may withdraw the device prior to its production.

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

While NEC Corporation has been making continuous effort to enhance the reliability of its semiconductor devices, the possibility of defects cannot be eliminated entirely. To minimize risks of damage or injury to persons or property arising from a defect in an NEC semiconductor device, customers must incorporate sufficient safety measures in its design, such as redundancy, fire-containment, and anti-failure features.

NEC devices are classified into the following three quality grades:

"Standard", "Special", and "Specific". The Specific quality grade applies only to devices developed based on a customer designated "quality assurance program" for a specific application. The recommended applications of a device depend on its quality grade, as indicated below. Customers must check the quality grade of each device before using it in a particular application.

- Standard: Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots
- Special: Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support)
- Specific: Aircrafts, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems or medical equipment for life support, etc.

The quality grade of NEC devices is "Standard" unless otherwise specified in NEC's Data Sheets or Data Books. If customers intend to use NEC devices for applications other than those specified for Standard quality grade, they should contact an NEC sales representative in advance.

Anti-radioactive design is not implemented in this product.

M5 96.5

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics Inc. (U.S.)
Santa Clara, California
Tel: 800-366-9782
Fax: 800-729-9288

NEC Electronics (Germany) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 02
Fax: 0211-65 03 490

NEC Electronics (UK) Ltd.
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Italiana s.r.1.
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

NEC Electronics (Germany) GmbH
Benelux Office
Eindhoven, The Netherlands
Tel: 040-2445845
Fax: 040-2444580

NEC Electronics (France) S.A.
Velizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

NEC Electronics (France) S.A.
Spain Office
Madrid, Spain
Tel: 01-504-2787
Fax: 01-504-2860

NEC Electronics (Germany) GmbH
Scandinavia Office
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Singapore Pte. Ltd.
United Square, Singapore 1130
Tel: 253-8311
Fax: 250-3583

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-719-2377
Fax: 02-719-5951

NEC do Brasil S.A.
Sao Paulo-SP, Brasil
Tel: 011-889-1680
Fax: 011-889-1689

J96. 8

[MEMO]

INTRODUCTION

Target Users

This manual explains the functions of the μPD784225 and μ784225Y Subseries to engineers who will design application systems.

Objective

This manual describes the hardware functions of the μPD784225 and 784225Y Subseries.

Organization

The μPD784225 and 784225Y Subseries user's manual is divided into two parts, the hardware edition (this manual) and the instruction edition.

Hardware Edition	Instruction Edition
Pin functions	CPU functions
Internal block functions	Addressing
Interrupts	Instruction set
Other on-chip peripheral functions	

There are Cautions associated with using this product.
Be sure to read the Cautions in the text of each chapter and summarized at the end of each chapter.

How to read this manual

Reading this manual requires general knowledge about electronics, logic circuits, and microcontrollers.

- **If there are no particular differences in the function**
  μPD784225 in the μPD784225 Subseries is described as the representative mask ROM product, and μPD78F4225 is described as the representative flash memory product.

- **If there are differences in the function**
  Each product name is presented and described separately.
  Since representative μPD784225 Subseries products are described even this case, for information on the operation of μPD784225Y Subseries products, read the sections on the μPD784224Y, 784225Y, and 78F4225Y instead of the μPD784224, 784225, and 78F4225.

- **To understand the overall function**
  → Read following the table of contents.

- **To debug when the operation is unusual**
  → Since the cautions are summarized at the end of each chapter, see the cautions associated with the function.

- **Since the cautions are summarized at the end of each chapter, see the cautions associated with the function.**
  → See **Appendix D Register Index.**

- **For detailed explanations of the instruction functions**
  → Refer to the other manual **78K/IV Series User's Manual, Instruction (U10905E).**

- **For explanations of the application examples of the functions**
  → Refer to the application notes.

Differences between the μPD784225 Subseries and the μPD784225Y Subseries

The only functional difference between the μPD784225 Subseries and μPD784225Y Subseries is the clock synchronized serial interface. The two subseries share the other functions.

Caution

The clock-synchronized serial interface is described in the following two chapters.

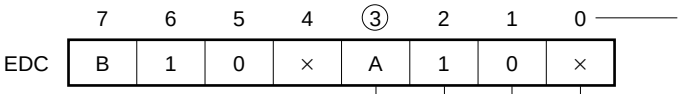
- **Chapter 17 3-wire Serial I/O Mode**
- **Chapter 18 I2C Bus Mode (Only the μPD784225Y Subseries)**

Read both chapters for an overview of the serial interface in chapter 15.

Explanatory notes

Precedence in data notation	:	The left side is the most significant digit. The right side is the least significant digit.
Active low notation	:	$\overline{\times\times\times}$ (overbar on pin or signal name)
Note	:	Note about the text
Caution	:	Contents that demand particular attention
Remarks	:	Supplemental description of the text
Numerical notations	:	Binary numbers = ××××B or ××××
	:	Decimal numbers - ××××
	:	Hexadecimal numbers - ××××H

Register notation



For the bit with a bitnumber circled, its bitname is a reserved word for NEC's assembler or is defined as an sfr variable by the # pragm sfr directive for C compiler.

When writing	When reading
Write 0 or 1. The operation is not affected by either value	Read 0 or 1.
Must write 0.	
Must write 1.	
Write the value for the function you want to use.	Read the value that conforms to the operating state.

Never write a combination of codes that have "Setting Disabled" written in the register description in this manual.

Characters that are confused : 0 (zero), O (capital o)
: 1 (one), l (letter l), I (capital i)

Related documents Be aware that “Preliminary” is not indicated on the documents although some of the related documents are preliminary editions.

Documents related to device

Document Name	Document No.	
	Japanese	English
μPD784224, 784225 Preliminary Product Information	U12498J	To be prepared
μPD78F4225 Preliminary Product Information	U12499J	U12499E
μPD784225 Subseries Special Function Register Table	U12698J	—
μPD784224Y, 784225Y Preliminary Product Information	U12376J	U12376E
μPD78F4225Y Preliminary Product Information	U12377J	U12377E
μPD784225Y Subseries Special Function Register Table	U12699J	—
μPD784225, 784225Y Subseries User’s Manual, Hardware	U12697J	This document
78K/IV Series Application Note, Software Basics	U10095J	—
78K/IV Series User’s Manual, Instruction	U10905J	U10905E
78K/IV Series Instruction Table	U10594J	—
78K/IV Series Instruction Set	U10595J	—

Documents related to development tools (User’s Manuals)

Document Name		Document No.	
		Japanese	English
RA78K4 Assembler Package	Operation	U11334J	U11334E
	Language	U11162J	U11162E
RA78K4 Structured Assembler Preprocessor		U11743J	U11743E
CC78K4 Series	Operation	EEU-960	U11572E
	Language	EEU-961	U11571E
CC78K Series Library Source File		U12322J	—
IE-784000-R		EEU-5004	EEU-1534
IE-784218-R-EM1		U12155J	U12155E
SM78K4 System Simulator, Windows™ Based	Reference	U10093J	U10093E
SM78K Series System Simulator	External parts user open interface specification	U10092J	U10092E
ID78K4 Integrated Debugger, Windows Based	Reference	U10440J	U10440E
ID78K4 Integrated Debugger HP9000 Series 700 (HP-UX Based)	Reference	U11960J	—

Caution The contents of the above related documents are subject to change without notice. Be sure to use the latest edition of a document for designing.

Documents related to embedded software (User's Manual)

Document Name		Document No.	
		Japanese	English
78K/IV Series Real-Time OS	Basics	U10603J	U10603E
	Installation	U10604J	U10604E
	Debugger	U10364J	–
78K/IV Series OS MX78K4	Basics	U11779J	–

Other documents

Document Name	Document No.	
	Japanese	English
IC Semiconductor Device Package Manual	C10943X	
Semiconductor Device Mounting Technology Manual	C10535J	C10535E
Quality Grades on NEC Semiconductor Devices	C11531J	C11531E
NEC Semiconductor Device Reliability/Quality Control System	C10983J	C10983E
Electrostatic Discharge (ESD) Test	MEM-539	–
Guide to Quality Assurance for Semiconductor Devices	C11893J	MEI-1202
Guide to Microcomputer-Related Products by Third Parties	U11416J	–

Caution The contents of the above related documents are subject to change without notice. Be sure to use the latest edition of a document for designing.

[MEMO]

CONTENTS

LIST OF FIGURES (1/7)

LIST OF FIGURES (2/7)

LIST OF FIGURES (3/7)

LIST OF FIGURES (4/7)

LIST OF FIGURES (5/7)

LIST OF FIGURES (6/7)

LIST OF FIGURES (7/7)

LIST OF TABLES (13)

LIST OF TABLES (2/3)

LIST OF TABLES (3/3)

CHAPTER 1 OVERVIEW

The μPD784225 Subseries is a member of the 78K/IV Series, and is an 80-pin general-purpose microcontroller restricted to the μPD784216 Subseries functions to which a ROM correction function is added. The 78K/IV Series has 8-/16-bit single-chip microcontrollers and provides a high-performance CPU with an access function for a 1-Mbyte memory space.

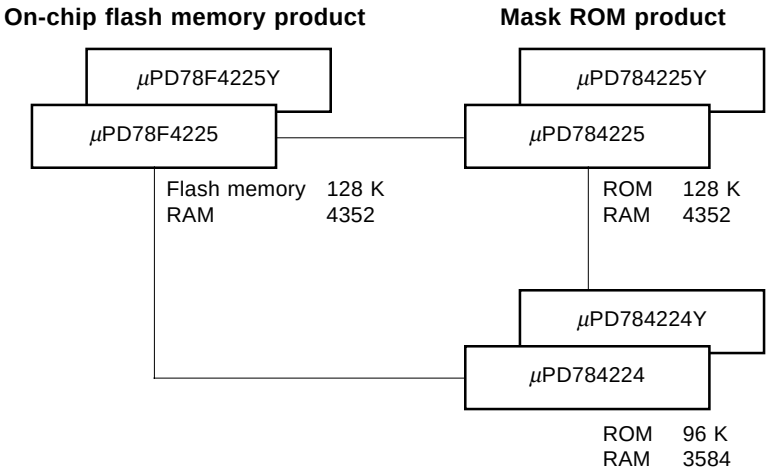
The μPD784225 has a 128-Kbyte ROM and 4,352-byte RAM on chip. In addition, it has a high-performance timer/counter, an 8-bit A/D converter, an 8-bit D/A converter, and an independent 2-channel serial interface.

The μPD84224 is the μPD784225 with a 96-Kbyte mask ROM and a 3,584-byte RAM.

The μPD78F4225 is the μPD784225 with the mask ROM replaced by a flash memory.

The μPD784225Y Subseries is the μPD784225 Subseries with an added I^2C bus control function.

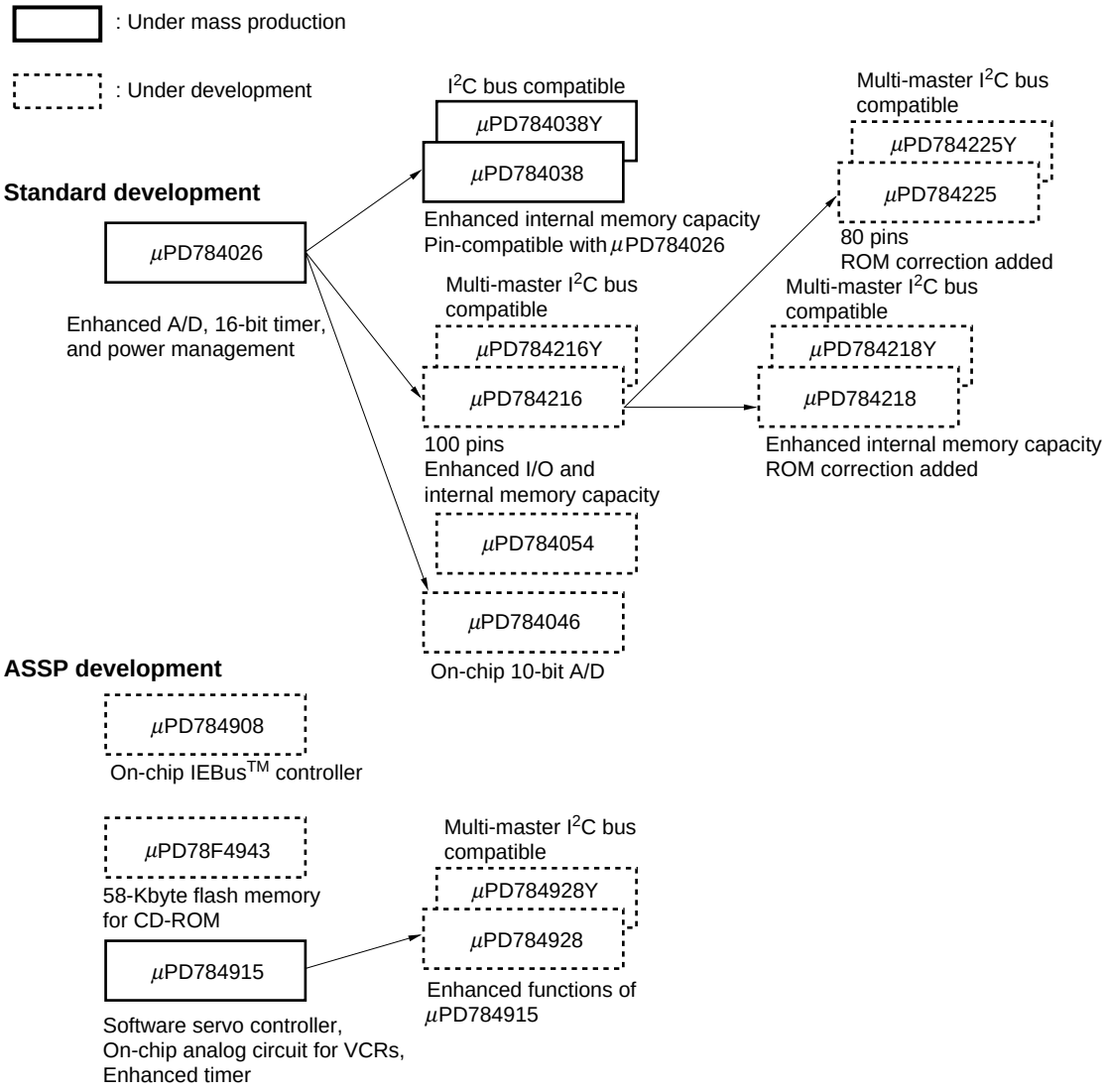
The relationships among the products are shown below.



These products can be applied in the following areas.

- Car audio, portable audio, telephones, etc.

78K/IV Series Product Development Diagram



1.1 Features

- Inherits the peripheral functions of the μPD780058 Subseries
- Minimum instruction execution time
  - 160 ns (main system clock: f_{XX} = 12.5-MHz operation)
  - 61 μs (subsystem clock: f_{XT} = 32.768-kHz operation)
- Instruction set suited for control applications
- Interrupt controller (4-level priority)
  - Vectored interrupt servicing, macro service, context switching
- Standby function
  - HALT, STOP, IDLE modes
  - In the low power consumption mode: HALT, IDLE modes (subsystem clock operation)
- On-chip memory:

Mask ROM	128 Kbytes (μPD784225)
	96 Kbytes (μPD784224)
Flash memory	128 Kbytes (μPD78F4225)
RAM	4,352 bytes (μPD784225, 78F4225)
	3,584 bytes (μPD784224)

- I/O pins: 67
  - Software programmable pullup resistors: 57 inputs
  - LED direct drive possible: 16 outputs
- Timer/counter: 16-bit timer/counter × 1 unit
  8-bit timer/counter × 4 units
- Watch timer: 1 channel
- Watchdog timer: 1 channel
- Serial interfaces
  - UART/IOE (3-wire serial I/O): 2 channels (on-chip baud rate generator)
  - CSI (3-wire serial I/O, multimaster compatible I^2C bus**Note**): 1 channel
- A/D converter: 8-bit resolution × 8 channels
- D/A converter: 8-bit resolution × 2 channels
- Real-time output port (by combining with the timer/counter, two stepping motors can be independently controlled.)
- Clock frequency function
- Clock output function: Select from f_{XX}, $f_{XX}/2$, $f_{XX}/2^2$, $f_{XX}/2^3$, $f_{XX}/2^4$, $f_{XX}/2^5$, $f_{XX}/2^6$, $f_{XX}/2^7$, f_{XT}
- Buzzer output function: Select from $f_{XX}/2^{10}$, $f_{XX}/2^{11}$, $f_{XX}/2^{12}$, $f_{XX}/2^{13}$
- Power supply voltage: V_{DD} = 1.8 to 5.5 V

Note Only in the μPD784225Y Subseries

1.2 Ordering Information

(1) μPD784225 Subseries

Part Number	Package	On-chip ROM
μPD784224GC-×××-8BT	80-pin plastic QFP (14 × 14 mm)	Mask ROM
μPD784224GK-×××-BE9	80-pin plastic TQFP (fine pitch) (12 × 12 mm)	Mask ROM
μPD784225GC-×××-8BT	80-pin plastic QFP (14 × 14 mm)	Mask ROM
μPD784225GK-×××-BE9	80-pin plastic TQFP (fine pitch) (12 × 12 mm)	Mask ROM
μPD78F4225GC-8BT	80-pin plastic QFP (14 × 14 mm)	Flash memory
μPD78F4225GK-BE9	80-pin plastic TQFP (fine pitch) (12 × 12 mm)	Flash memory

Remark ××× indicates the ROM code number.

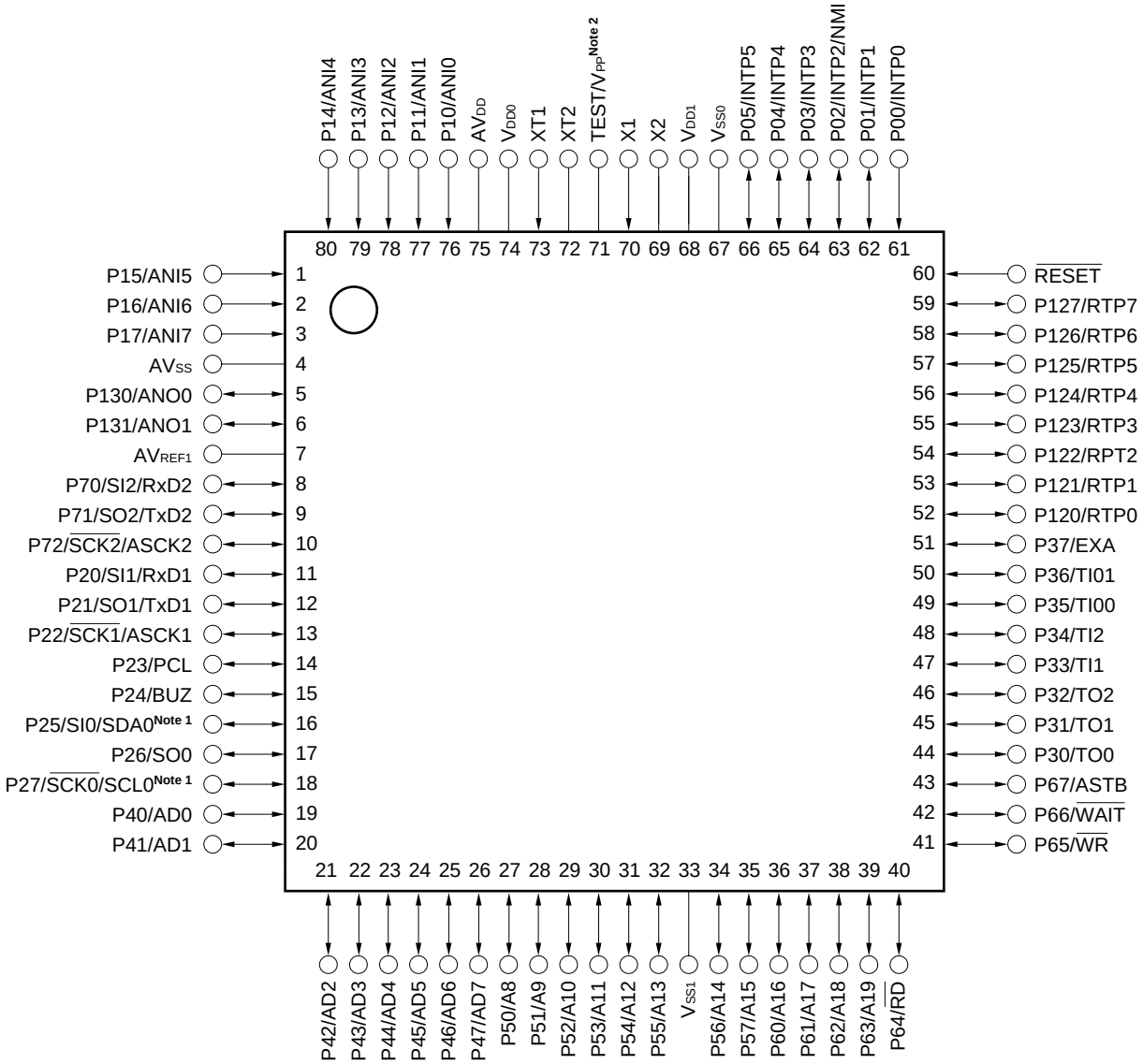
(2) μPD784225Y Subseries

Part Number	Package	On-chip ROM
μPD784224YGC-×××-8BT	80-pin plastic QFP (14 × 14 mm)	Mask ROM
μPD784224YGK-×××-BE9	80-pin plastic TQFP (fine pitch) (12 × 12 mm)	Mask ROM
μPD784225YGC-×××-8BT	80-pin plastic QFP (14 × 14 mm)	Mask ROM
μPD784225YGK-×××-BE9	80-pin plastic TQFP (fine pitch) (12 × 12 mm)	Mask ROM
μPD78F4225YGC-8BT	80-pin plastic QFP (14 × 14 mm)	Flash memory
μPD78F4225YGK-BE9	80-pin plastic TQFP (fine pitch) (12 × 12 mm)	Flash memory

Remark ××× indicates the ROM code number.

1.3 Pin Configuration (Top View)

- **80-pin plastic QFP (14 × 14 mm)**
  μPD784224GC-×××-8BT, 784225GC-×××-8BT, 784224YGC-×××-8BT,
  μPD784225YGC-×××-8BT, 78F4225GC-8BT, 78F4225YGC-8BT
- **80-pin plastic TQFP (fine pitch) (12 × 12 mm)**
  μPD784224GK-×××-BE9, 784225GK-×××-BE9, 784224YGK-×××-BE9,
  μPD784225YGK-×××-BE9, 78F4225GK-BE9, 78F4225YGK-BE9



Notes 1. The SDA0 and SCL0 pins are provided only for the μPD784225Y Subseries.

2. The V_{PP} pin is povided only for the μPD78F4225 and 78F4225Y.

Cautions 1. Connect the TEST/V_{PP} pin directly to V_{SS0} when in the normal operating mode.

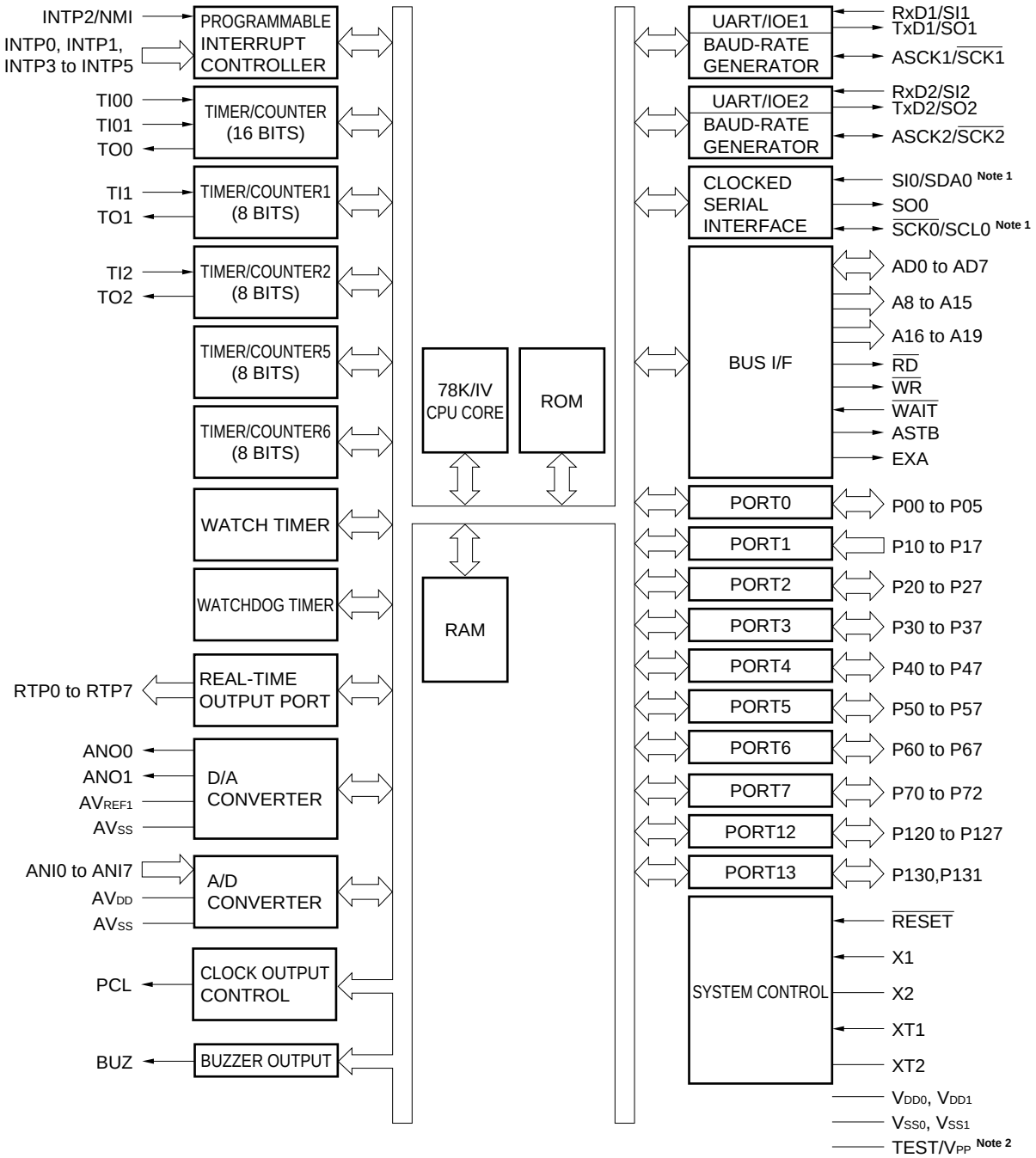
2. Directly connect the AV_{SS} pin to V_{SS0}.

Remark When used in applications that require the noise generated from the inside of the micro-computer to be reduced, it is recommended that certain noise reductions measures are observed, such as supplying electrical power separately to V_{DD0} and V_{DD1}, and connecting separate ground lines for V_{SS0} and V_{SS1}.

A8 to A19 : Address Bus
AD0 to AD7 : Address/Data Bus
ANI0 to ANI7 : Analog Input
ANO0, ANO1 : Analog Output
ASCK1, ASCK2: Asynchronous Serial Clock
ASTB : Address Strobe
AV_{DD} : Analog Power Supply
AV_{REF1} : Analog Reference Voltage
AV_{SS} : Analog Ground
BUZ : Buzzer Clock
EXA : External Access Status Output
INTP0 to INTP5: Interrupt from Peripherals
NMI : Non-maskable Interrupt
P00 to P05 : Port 0
P10 to P17 : Port 1
P20 to P27 : Port 2
P30 to P37 : Port 3
P40 to P47 : Port 4
P50 to P57 : Port 5
P60 to P67 : Port 6
P70 to P72 : Port 7
P120 to P127 : Port 12
P130, P131 : Port 13
PCL : Programmable Clock
$\overline{RD}$: Read Strobe
$\overline{RESET}$: Reset
RTP0 to RTP7 : Real-time Output Port
RxD1, RxD2 : Receive Data
$\overline{SCK0}$ to $\overline{SCK2}$: Serial Clock
SCL0**Note 1** : Serial Clock
SDA0**Note 1** : Serial Data
SI0 to SI2 : Serial Input
SO0 to SO2 : Serial Output
TEST : Test
TI00, TI01, TI1, TI2: Timer Input
TO0 to TO2 : Timer Output
TxD1, TxD2 : Transmit Data
V_{DD0}, V_{DD1} : Power Supply
V_{PP}**Note 2** : Programming Power Supply
V_{SS0}, V_{SS1} : Ground
$\overline{WAIT}$: Wait
$\overline{WR}$: Write Strobe
X1, X2 : Crystal (Main System Clock)
XT1, T2 : Crystal (Subsystem Clock)

Notes 1. The SDA0 and SCL0 pins are provided only for the μPD784225Y Subseries.
2. The V_{PP} pin is provided only for the μPD78F4225 and 78F4225Y.

1.4 Block Diagram



Notes 1. The SDA0 and SCL0 pins are provided only for the μPD784225Y Subseries.

2. The V_{PP} pin is provided only for the μPD78F4225 and 78F4225Y.

Remark The capacities of the on-chip ROM and RAM differ with the product.

1.5 Function List

(1/2)

Item \ Product Name		μPD784224 μPD784224Y	μPD784225 μPD784225Y	μPD78F4225 μPD78F4225Y
No. of basic instructions (mnemonics)		113		
General-purpose registers		8 bits × 16 registers × 8 banks or 16 bits × 8 registers × 8 banks (memory mapping)		
Minimum instruction execution time		• 160 ns, 320 ns, 640 ns, 1,280 ns, 2,560 ns (main system clock running at 12.5 MHz) • 61 μs (subsystem clock running at 32.768 kHz)		
Internal memory	ROM	96 Kbytes (mask ROM)	128 Kbytes (mask ROM)	128 Kbytes (flash memory)
	RAM	3,584 bytes	4,352 bytes	
Memory space		1 Mbyte of combined program and data		
I/O ports	Total	67		
	CMOS inputs	8		
	CMOS I/O	59		
Pins with added functions[Note]	Pins with pull-up resistors	57		
	LED direct drive outputs	16		
Real-time output ports		4 bits × 2 or 8 bits × 1		
Timer/counters		Timer/counter: (16 bits) Timer register × 1 Capture/compare register × 2 Pulse output possible • PWM/PPG output • Square wave output • One-shot pulse output		
		Timer/counter 1: (8 bits) Timer register × 1 Compare register × 1 Pulse output possible • PWM output • Square wave output		
		Timer/counter 2: (8 bits) Timer register × 1 Compare register × 1 Pulse output possible • PWM output • Square wave output		
		Timer/counter 5: (8 bits) Timer register × 1 Compare register × 1 Pulse output possible • PWM output • Square wave output		
		Timer/counter 6: (8 bits) Timer register × 1 Compare register × 1 Pulse output possible • PWM output • Square wave output		

Note The pins with added functions include the I/O pins.

(2/2)

Item \ Product Name		μPD784224 μPD784224Y	μPD784225 μPD784225Y	μPD78F4225 μPD78F4225Y
Serial interfaces		• UART/IOE (3-wire serial I/O : 2 channels (on-chip baud rate generator) • CSI (3-wire serial I/O, multimaster compatible I^2C bus**Note**): 1 channel		
A/D converter		8-bit resolution × 8 channels		
D/A converter		8-bit resolution × 2 channels		
Clock output:		Select from f_{XX}, $f_{XX}/2$, $f_{XX}/2^2$, $f_{XX}/2^3$, $f_{XX}/2^4$, $f_{XX}/2^5$, $f_{XX}/2^6$, $f_{XX}/2^7$, f_{XT}		
Buzzer output :		Select from $f_{XX}/2^{10}$, $f_{XX}/2^{11}$, $f_{XX}/2^{12}$, $f_{XX}/2^{13}$		
Watch timer		1 channel		
Watchdog timer		1 channel		
Standby		• HALT, STOP, IDLE modes • In the low power consumption mode (CPU operation by subsystem clock): HALT/IDLE mode		
Interrupts	Hardware sources	25 (internal: 18, external: 7)		
	Software sources	BRK instruction, BRKCS instruction, operand error		
	Non-maskable	Internal: 1, external: 1		
	Maskable	Internal: 17, external: 6		
		• 4-level programmable priority • Three processing formats: Vectored interrupt, macro service, context switching		
Power supply voltage		V_{DD} = 1.8 to 5.5 V		
Package		• 80-pin plastic TQFP (fine pitch) (12 × 12 mm) • 80-pin plastic QFP (14 × 14 mm)		

Note Only in the μPD784225Y Subseries

1.6 Differences Between μPD784225 Subseries Products and μPD784225Y Subseries Products

Item \ Product Name	μPD784224 μPD784224Y	μPD784225 μPD784225Y	μPD78F4225 μPD78F4225Y
Internal ROM	96 Kbytes (mask ROM)	128 Kbytes (mask ROM)	128 Kbytes (flash memory)
Internal RAM	3,584 bytes	4,352 bytes	
Internal memory size switching register (IMS)	No		Yes
V_{PP} pin	No		Yes

CHAPTER 2 PIN FUNCTIONS

2.1 Pin Function List

(1) Port pins (1/2)

Pin Symbol	I/O	Multi - use Pins	Function
P00	I/O	INTP0	Port 0 (P0) : • 6-bit I/O port • Can specify input or output in 1-bit units • Regardless of whether the input or output mode is specified, on-chip pull-up resistor can be specified by software in 1-bit units.
P01		INTP1	
P02		INTP2/NMI	
P03		INTP3	
P04		INTP4	
P05		INTP5	
P10 to P17	Input	ANI0 to ANI7	Port 1 (P1) : • 8-bit dedicated input port
P20	I/O	RxD1/SI1	Port 2 (P2) : • 8-bit I/O port • Can specify input or output in 1-bit units • Regardless of whether the input or output mode is specified, on-chip pull-up resistor can be specified by software in 1-bit units.
P21		TxD1/SO1	
P22		ASCK1/$\overline{\text{SCK1}}$	
P23		PCL	
P24		BUZ	
P25		SI0/SDA0**Note**	
P26		SO0	
P27		$\overline{\text{SCK0}}$/SCL0**Note**	
P30	I/O	TO0	Port 3 (P3) : • 8-bit I/O port • Can specify input or output in 1-bit units • Regardless of whether the input or output mode is specified, on-chip pull-up resistor can be specified by software in 1-bit units.
P31		TO1	
P32		TO2	
P33		TI1	
P34		TI2	
P35		TI00	
P36		TI01	
P37		EXA	
P40 to P47	I/O	AD0 to AD7	Port 4 (P4) : • 8-bit I/O port • Can specify input or output in 1-bit units • For input mode pins, on-chip pull-up resistors can be specified at once by software. • Can directly drive LEDs

Note The SDA0 and SCL0 pins are provided only for the μPD784225Y Subseries.

(1) Port Pins (2/2)

Pin Symbol	I/O	Multi - use Pins	Function
P50 to P57	I/O	A8 to A15	Port 5 (P5) : • 8-bit I/O port • Can specify input or output in 1-bit units • For input mode pins, on-chip pull-up resistors can be specified at once by software. • Can directly drive LEDs
P60	I/O	A16	Port 6 (P6) : • 8-bit I/O port • Can specify input or output in 1-bit units • For input mode pins, on-chip pull-up resistors can be specified at once by software.
P61		A17	
P62		A18	
P63		A19	
P64		$\overline{RD}$	
P65		$\overline{WR}$	
P66		$\overline{WAIT}$	
P67		ASTB	
P70	I/O	RxD2/SI2	Port 7 (P7) : • 3-bit I/O port • Can specify input or output in 1-bit units • Regardless of whether the input or output mode is specified, on-chip pull-up resistor can be specified by software in 1-bit units.
P71		TxD2/SO2	
P72		ASCK2/$\overline{SCK2}$	
P120 to P127	I/O	RTP0 to RTP7	Port 12 (P12) : • 8-bit I/O port • Can specify input or output in 1-bit units • Regardless of whether the input or output mode is specified, on-chip pull-up resistor can be specified by software in 1-bit units.
P130, P131	I/O	ANO0, ANO1	PORT 13 (P13): • 2-bit I/O port • Can specify input or output in 1-bit units

(2) Non-port pins (1/2)

Pin Symbol	I/O	Multi-use Pins	Function
TI00	Input	P35	External count clock input to 16-bit timer register
TI01		P36	Capture trigger signal input to capture/compare register 00
TI1		P33	External count clock input to 8-bit timer register 1
TI2		P34	External count clock input to 8-bit timer register 2
TO0	Output	P30	16-bit timer output (also for 14-bit PWM output)
TO1		P31	8-bit timer output (also for 8-bit PWM output)
TO2		P32	
RxD1	Input	P20/SI1	Serial data input (UART1)
RxD2		P70/SI2	Serial data input (UART2)
TxD1	Output	P21/SO1	Serial data output (UART1)
TxD2		P71/SO2	Serial data output (UART2)
ASCK1	Input	P22/$\overline{\text{SCK1}}$	Baud rate clock input (UART1)
ASCK2		P72/$\overline{\text{SCK2}}$	Baud rate clock input (UART2)
SI0	Input	P25/SDA0**Note**	Serial data input (3-wire serial I/O0)
SI1		P20/RxD1	Serial data input (3-wire serial I/O1)
SI2		P70/RxD2	Serial data input (3-wire serial I/O2)
SO0	Output	P26	Serial data output (3-wire serial I/O0)
SO1		P21/TxD1	Serial data output (3-wire serial I/O1)
SO2		P71/TxD2	Serial data output (3-wire serial I/O2)
SDA0**Note**	I/O	P25/SI0	Serial Data I/O (I^2C bus)
$\overline{\text{SCK0}}$	I/O	P27/SCL0**Note**	Serial clock I/O (3-wire serial I/O0)
$\overline{\text{SCK1}}$		P22/ASCK1	Serial clock I/O (3-wire serial I/O1)
$\overline{\text{SCK2}}$		P72/ASCK2	Serial clock I/O (3-wire serial I/O2)
SCL0**Note**		P27/$\overline{\text{SCK0}}$	Serial clock I/O (I^2C bus)

Note The SDA0 and SCL0 pins are provided only for the μPD784225Y Subseries.

(2) Non-port pins (2/2)

Pin Symbol	I/O	Multi-use Pins	Function
NMI	Input	P02/INTP2	Non-maskable interrupt request input
INTP0		P00	External interrupt request input
INTP1		P01	
INTP2		P02/NMI	
INTP3		P03	
INTP4		P04	
INTP5		P05	
PCL	Output	P23	Clock output (for main system clock, subsystem clock trimming)
BUZ	Output	P24	Buzzer output
RTP0 to RTP7	Output	P120 to P127	Real-time output port that outputs data synchronized to the trigger
AD0 to AD7	I/O	P40 to P47	Low-order address/data bus when the memory is externally expanded
A8 to A15	Output	P50 to P57	Middle-order address bus when the memory is externally expanded
A16 to A19		P60 to P63	High-order address bus when the memory is externally expanded
$\overline{RD}$	Output	P64	Strobe signal output for external memory read
$\overline{WR}$		P65	Strobe signal output for external memory write
$\overline{WAIT}$	Input	P66	Wait insertion during external memory access
ASTB	Output	P67	Strobe output that externally latches the address information that is output to ports 4 to 6 in order to access external memory.
EXA	Output	P37	External access status output
$\overline{RESET}$	Input	—	System reset input
X1	Input	—	Crystal connection for main system clock oscillation
X2	—		
XT1	Input	—	Crystal connection for subsystem clock oscillation
XT2	—		
ANI0 to ANI7	Input	P10 to P17	Analog voltage input to A/D converter
ANO0, ANO1	Output	P130, P131	Analog voltage output to D/A converter
AV_{REF1}	—	—	Reference voltage applied to D/A converter
AV_{DD}			Positive power supply to A/D converter. Connect to V_{DD0}.
AV_{SS}			Ground for A/D converter and D/A converter. Connect to V_{SS0}.
V_{DD0}			Port area's positive power source
V_{SS0}			Port area's ground electrical potential
V_{DD1}			Positive power source (excluding the port area)
V_{SS1}			Ground electrical potential (excluding the port area)
TEST		V_{PP}**Note**	Directly connect to V_{SS} (IC test pin).
V_{PP}**Note**		TEST	Flash memory programming mode setting High voltage application pin during program write/verify

Note The V_{PP} pin is provided only for the μPD78F4225 and 78F4225Y.

2.2 Pin Function Description

(1) P00 to P05 (Port0)

This port is a 6-bit I/O port. In addition to being an I/O port, it has an external interrupt request input function. The following operating modes are bit selectable.

(a) Port mode

This port functions as a 6-bit I/O port. The port 0 mode register can specify an input port or output port for each bit. Regardless of whether the input mode/output mode is specified, pull-up resistors can be connected in 1-bit units by pull-up resistor option register 0.

(b) Control mode

The port functions as an external interrupt request input.

(i) INTP0 to INTP5

INTP0 to INTP5 are external interrupt request input pins that can select the valid edge (rising edge, falling edge, both rising and falling edges). The valid edge can be specified by the external interrupt rising edge enable register and the external interrupt falling edge enable register. INTP2 also becomes the external trigger signal input pin of the real-time output port by the valid edge input.

(ii) NMI

This is the external nonmaskable interrupt request input pin. The valid edge can be specified by the external interrupt rising edge enable register and the external interrupt falling edge enable register.

(2) P10 to P17 (Port1)

This port is an 8-bit dedicated input port. In addition to being a general-purpose input port, this port functions as the analog input for the A/D converter.

It does not have on-chip pull-up resistors.

(a) Port mode

The port functions as an 8-bit dedicated input port.

(b) Control mode

The port functions as the analog input pins (ANI0 to ANI7) of the A/D converter. The values are undefined when the pins specified for analog input are read.

(3) P20 to P27 (Port2)

This port is an 8-bit I/O port. In addition to being an I/O port, this port has the data I/O function, clock I/O function, clock output function, and output buzzer function of the serial interface.

The following operating modes are bit selectable.

(a) Port mode

This mode functions as an 8-bit I/O port. The port 2 mode register can specify input ports or output ports in 1-bit units. Regardless of whether the input mode/output mode is specified, pull-up resistor can be specified by the pull-up resistance option register 2 in 1-bit units.

(b) Control mode

This port functions as the data I/O pins, clock I/O pins, clock output pins, and buzzer output pins of the serial interface.

Pins P25 to P27 can be specified in the N-channel open drain by the port function control register (PF2) (only in the μPD784225Y Subseries).

(i) SI0, SI1, SO0, SO1, SDA0[Note]

These pins are the I/O pins for serial data in the serial interface.

(ii) $\overline{\text{SCK0}}$, $\overline{\text{SCK1}}$, SCL0[Note]

These pins are the I/O pins for the serial clock of the serial interface.

(iii) RxD1, TxD1

These pins are the I/O pins for serial data in the asynchronous serial interface.

Note Only in the μPD784225Y Subseries

(iv) ASCK1

This is the I/O pin for the baud rate clock of the asynchronous serial interface.

(v) PCL

This is the clock output pin.

(vi) BUZ

This is the buzzer output pin.

(4) P30 to P37 (Port3)

This port is an 8-bit I/O port. In addition to being an I/O port, this port has the timer I/O function.
The following operating modes are bit selectable.

(a) Port mode

The port functions as an 8-bit I/O port. The port 3 mode register can specify input ports or output ports in 1-bit units. Regardless of whether the input mode/output mode is specified, pull-up resistor can be specified by the pull-up resistance option register 3 in 1-bit units.

(b) Control mode

The port functions as timer I/O.

(i) TI00

This is the external clock input pin to the 16-bit timer/counter.

(ii) TI01

This is the capture trigger signal input pin to capture/compare register 0.

(iii) TI1, TI2

These are external clock input pins to the 8-bit timer/counter.

(iv) TO0 to TO2

These are timer output pins.

(5) P40 to P47 (Port4)

This is an 8-bit I/O port. In addition to being an I/O port, this port has an address/data bus function. LEDs can be directly driven.
The following operating modes can be specified in 1-bit units.

(a) Port mode

This port functions as an 8-bit I/O port. The port 4 mode register can specify input ports or output ports in 1-bit units. When used as an input port, pull-up resistors can be connected in 8-bit units with bit 4 (PUO4) of the pull-up resistor option register.

(b) Control mode

The port functions as the low-order address/data bus pins (AD0 to AD7) when in the external memory expansion mode. If PUO4 = 1, pull-up resistors can be connected.

(6) P50 to P57 (Port5)

This port is an 8-bit I/O port. In addition to being an I/O port, it has an address bus function. LEDs can be directly driven.

The following operating modes can be specified in 1-bit units.

(a) Port mode

The port functions as an 8-bit I/O port. The port 5 mode register can specify input ports or output ports in 1-bit units. When used as an input port, bit 5 (PUO5) of the pull-up resistor option register can connect the pull-up resistors in 8-bit units.

(b) Control mode

The port functions as the middle address bus pins (A8 to A15) in the external memory expansion mode. If PU05 = 1, pull-up resistors can be connected.

(7) P60 to P67 (Port6)

This port is an 8-bit I/O port. In addition to being an I/O port, this port has the address bus function and control function in the external memory expansion mode.

The following operating modes can be specified in 1-bit units.

(a) Port mode

The port functions as an 8-bit I/O port. The port 6 mode register can specify input ports or output ports in 1-bit units. When used as an input port, bit 6 (PUO6) of the pull-up resistor option register can connect the pull-up resistors in 8-bit unit.

(b) Control mode

P60 to P63 function as the high-order address bus pins (A16 to A19) in the external memory expansion mode. P64 to P67 function as the control signal output pins ($\overline{RD}$, $\overline{WR}$, $\overline{WAIT}$, ASTB) in the external memory expansion mode. If PUO6 = 1 in the external memory expansion mode, pull-up resistors can be connected.

Caution When external waits are not used in the external memory expansion mode, P66 can be used as an I/O port.

(8) P70 to P72 (Port7)

This is a 3-bit I/O port. In addition to being an I/O port, this port also has the data I/O and clock I/O functions of the serial interface.

The following operating modes can be specified in 1-bit units.

(a) Port mode

The port functions as a 3-bit I/O port. The port 7 mode register can specify input ports or output ports in 1-bit units. Regardless of whether the input mode/output mode is specified, pull-up resistor can be specified by the pull-up resistance option register 7 in 1-bit units.

(b) Control mode

The port functions as data I/O and clock I/O for the serial interface.

(i) SI2, SO2

These are the I/O pins for serial data in the serial interface.

(ii) $\overline{SCK2}$

This is the I/O pin of the serial clock in the serial interface.

(iii) RxD2, TxD2

These are the serial data I/O pins in the asynchronous serial interface.

(iv) ASCK2

This is baud rate clock input pin in the asynchronous serial interface.

(9) P120 to P127 (Port12)

This port is an 8-bit I/O port. In addition to being an I/O port, it has a real-time output port function.
The following operating modes are bit selectable.

(a) Port mode

The port functions as an 8-bit I/O port. The port 12 mode register can specify input ports or output ports in 1-bit units. Regardless of whether the input mode/output mode is specified, pull-up resistor can be specified by the pull-up resistance option register 12 in 1-bit units.

(b) Control mode

The port functions as the real-time output port (RTP0 to RTP7) that outputs data synchronized to a trigger. When the pins specified as the real-time output port are read, 0 is read.

(10) P130, P131 (Port13)

This port is a 2-bit I/O port. In addition to being an I/O port, it has the analog output function of the D/A converter.
The following operating modes can be specified in 2-bit units.

(a) Port mode

The port functions as a 2-bit I/O port. The port 13 mode register can specify input ports or output ports in 1-bit units. There are no internal pull-up resistors.

(b) Control mode

The port functions as the analog outputs (ANO0, ANO1) of the D/A converter. The values are undefined when the pins specified as analog output are read.

Caution If the D/A converter uses only one channel when $AV_{REF1} < V_{DD}$, use either of the following processes at pins that are not used for analog output.

- **Set 1 (input mode) in the port mode register (PM13X) and connect to V_{SS0}.**
- **(output mode) in the port mode register (PM13X). Set the output latch to 0 and output a low level.**

(11) AV_{REF1}

This is the standard voltage input pin of the D/A converter.
If the D/A converter is not used, connect to pin V_{DD0}.

(12) AV_{DD}

This is the analog voltage supply pin of the A/D converter. Even if the A/D converter is not used, always use this pin at the same potential as pin V_{DD0}. AV_{DD} is an alternative pin with the standard voltage input pin of the A/D converter.

(13) AV_{SS}

This is the ground electrical potential pin of the A/D converter. Even if the A/D converter is not used, always use this pin at the same potential as pin V_{SS0}.

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

This is the active low system reset input pin.

(15) X1, X2

These are the crystal oscillator connection pins for main system clock oscillation.

When an external clock is supplied, input this clock signal at X1, and its inverted signal at X2.

(16) XT1, XT2

These are the crystal oscillator connection pins for subsystem clock oscillation.

When an external clock is supplied, input this clock signal at XT1, and its inverted signal at XT2.

(17) V_{DD0}, V_{DD1}

V_{DD0} is the port area's positive power supply pin.

V_{DD1} is the positive power supply pin for areas other than the port area and analog area.

(18) V_{SS0}, V_{SS1}

V_{SS0} is the port area's ground electrical potential pin.

V_{SS1} is the ground electrical potential pin for areas other than the port area and analog area.

(19) V_{PP} (μPD78F4225, 78F4225Y)

This is the high-voltage application pin when setting the flash memory programming mode and writing or verifying the program.

In the normal operating mode, connect directly to V_{SS0}.

(20) TEST

This is the pin used in the IC test. Always connect directly to V_{SS0}.

2.3 Pin I/O Circuit and Handling of Unused Pins

Table 2-1 shows the I/O circuit type for the pins and how to handle unused pins.
See Figure 2-1 for each type of I/O circuit.

Table 2-1. I/O Circuit Type for Each Pin and Handling Unused Pins (1/2)

Pin Symbol	I/O Circuit Type	I/O	Recommended Connection When Unused
P00/INTP0	8-C	I/O	During input : Connect to V_{SS0} through each resistor. During output: Leave open
P01/INTP1			
P02/INTP2/NMI			
P03/INTP3 to P05/INTP5			
P10/ANI0 to P17/ANI7	9	Input	Connect to V_{SS0} or V_{DD0}.
P20/RxD1/SI1	10-B	I/O	During input : Connect to V_{SS0} through each resistor. During output: Leave open
P21/TxD1/SO1			
P22/ASCK1/$\overline{SCK1}$			
P23/PCL			
P24/BUZ			
P25/SDA0[Note]/SI0			
P26/SO0			
P27/SCL0[Note]/$\overline{SCK0}$			
P30/TO0 to P32/TO2	8-C		
P33/TI1, P34/TI2			
P35/TI00, P36/TI01			
P37/EXA			
P40/AD0 to P47/AD7	5-H		
P50/A8 to P57/A15			
P60/A16 to P63/A19			
P64/$\overline{RD}$			
P65/$\overline{WR}$			
P66/$\overline{WAIT}$			
P67/ASTB			
P70/RxD2/SI2	8-C		
P71/TxD2/SO2			
P72/ASCK2/$\overline{SCK2}$			

Note The SDA0 and SCL0 pins are provided only for the μPD784225Y Subseries.

Table 2-1. I/O Circuit Type for Each Pin and Handling Unused Pins (2/2)

Pin Symbol	I/O Circuit Type	I/O	Recommended Connection When Unused
P120/RTP0 to P127/RTP7	8-C	I/O	During input : Connect to V_{SS0} through each resistor. During output: Leave open
P130/ANO0, P131/ANO1	12-C		
$\overline{\text{RESET}}$	2	Input	—
XT1	16		Connect to V_{SS0}.
XT2		—	Leave open
AV_{REF1}	—		Connect to V_{DD0}.
AV_{DD}			
AV_{SS}			Connect to V_{SS0}.
TEST/V_{PP} **Note**			Connect directly to V_{SS0}.

Note The V_{PP} pin is provided only for the μPD78F4225, 78F4225Y.

Remark Since the type number is unified with the 78K Series, it is not always to the serial numbers for each product (circuits that are not on-chip).

Figure 2-1. Pin I/O Circuit (1/2)

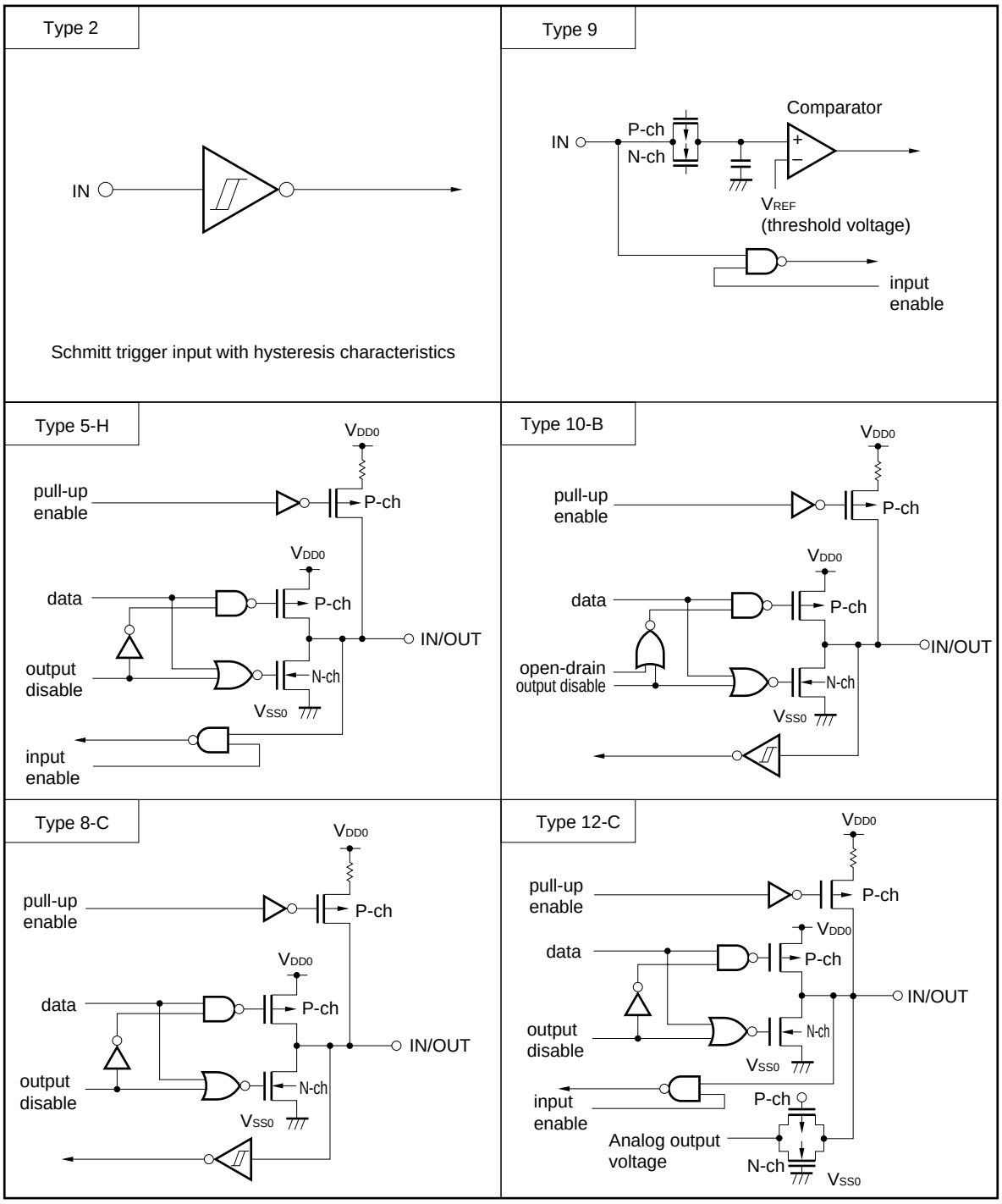
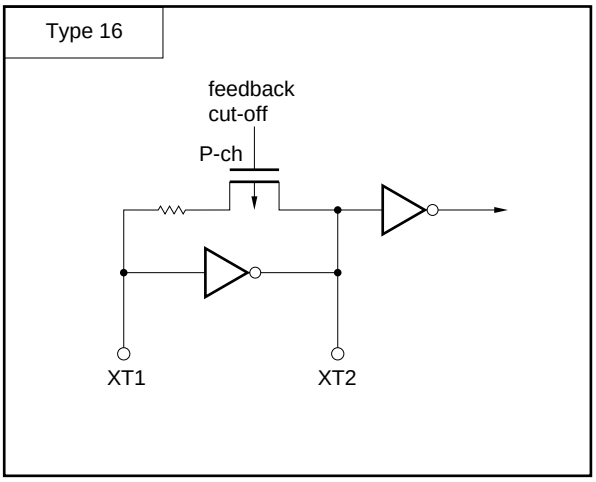


Figure 2-1. Pin I/O Circuit (2/2)



[MEMO]

CHAPTER 3 CPU ARCHITECTURE

3.1 Memory Space

The μPD784225 can access a 1-Mbyte space. The mapping of the internal data space differs with the LOCATION instruction (special function register and internal RAM). The LOCATION instruction must always be executed after clearing a reset and cannot be used more than once.

The program after clearing a reset must be as follows.

```
RSTVCT    CSEG    AT 0
          DW      RSTSTRT
            to
INITSEG   CSEG    BASE
RSTSTRT:  LOCATION 0H; or LOCATION 0FH
          MOVG    SP, #STKBGN
```

(1) When the LOCATION 0 instruction is executed

- **On-chip memory**

  The internal data area and internal ROM space are as follows.

Part Name	Internal Data Area	Internal ROM Space
μPD784224	0F100H to 0FFFFH	00000H to 0F0FFH 10000H to 17FFFH
μPD784225	0EE00H to 0FFFFH	00000H to 0EDFFH 10000H to 1FFFFH

Caution The following space that is overlapped from the internal data area in the on-chip ROM cannot be used while the LOCATION 0 instruction is executing.

Part Name	Unused Area
μPD784224	0F100H to 0FFFFH (3,840 bytes)
μPD784225	0EE00H to 0FFFFH (4,608 bytes)

- **External memory**

  External memory is accessed in the external memory expansion mode.

(2) When the LOCATION 0FH instruction is executed

- **Internal memory**

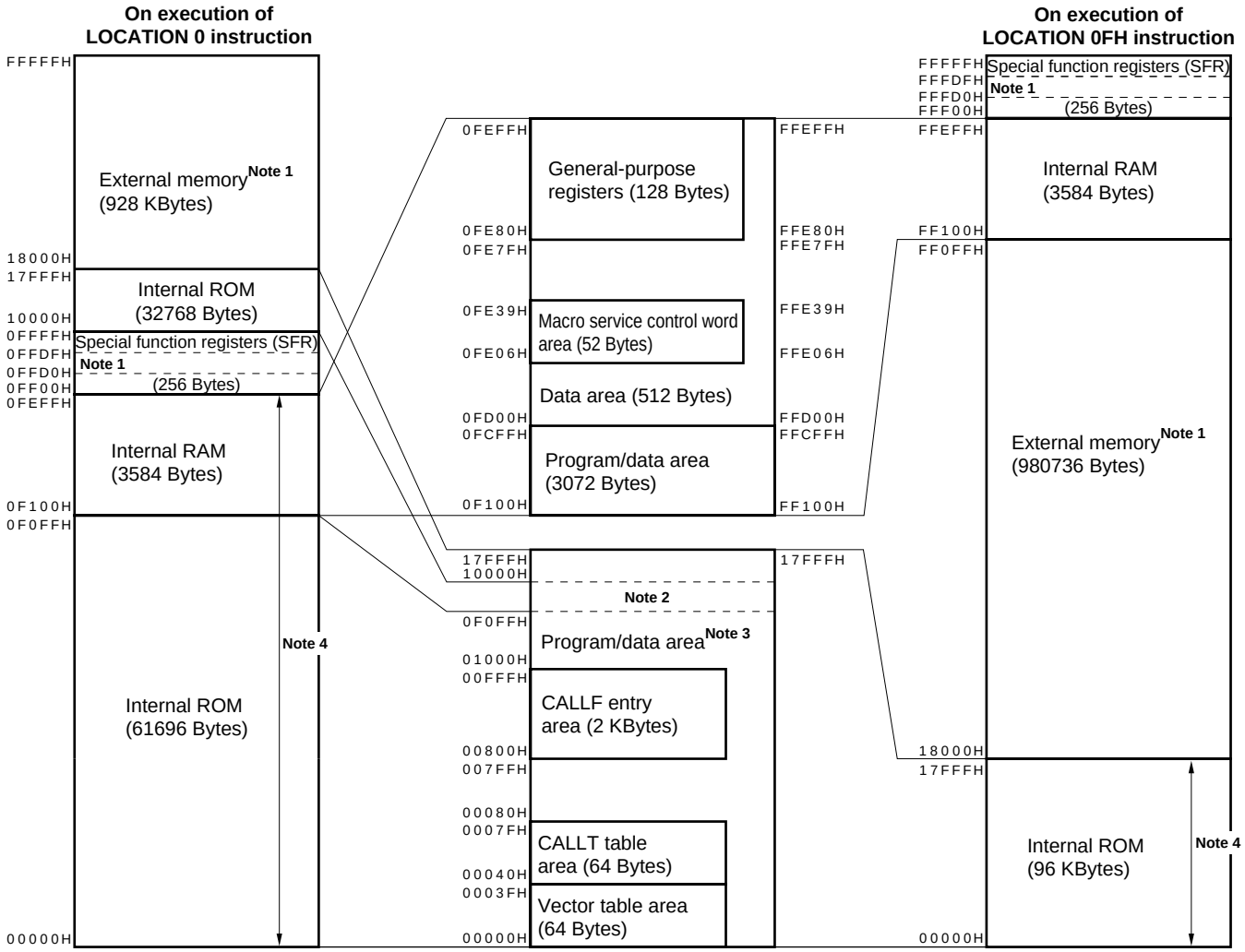
  The internal data area and internal ROM space are as follows.

Part Name	Internal Data Area	Internal ROM Space
μPD784224	FF100H to FFFFFH	00000H to 17FFFH
μPD784225	FEE00H to FFFFFH	00000H to 1FFFFH

- **External Memory**

  External memory is accessed in the external memory expansion mode.

Figure 3-1. μPD784224 Memory Map



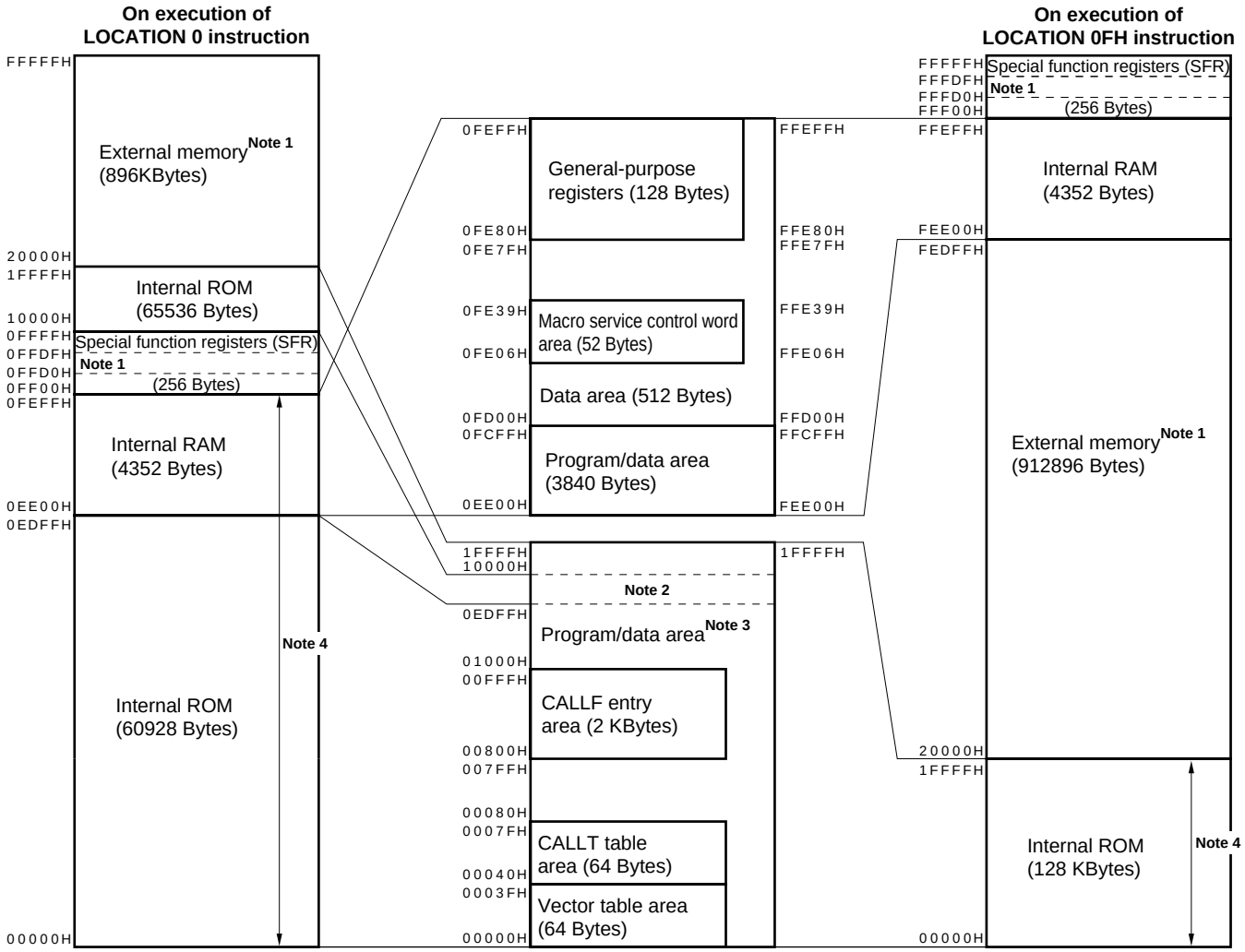
Notes 1. Access in the external memory expansion mode.

2. The 3,840 bytes in this area can be used as the internal ROM only when the LOCATION 0FH instruction is executing.

3. LOCATION 0 instruction execution: 94,464 bytes; LOCATION 0FH instruction execution: 98,304 bytes

4. This is the base area and the entry area based on resets or interrupts. However, the internal RAM is excluded in a reset.

Figure 3-2. μPD784225 Memory Map



Notes 1. Access in the external memory expansion mode.

2. The 4,608 bytes in this area can be used as the internal ROM only when the LOCATION 0FH instruction is executing.

3. LOCATION 0 instruction execution: 126,464 bytes; LOCATION 0FH instruction execution: 131,072 bytes

4. This is the base area and the entry area based on resets or interrupts. However, the internal RAM is excluded in a reset.

3.2 Internal ROM Space

The following products in the μPD784225 subseries have on-chip ROMs that can store the programs and table data.

If the internal ROM area or internal data area overlap when the LOCATION 0 instruction is executing, the internal data area becomes the access target. The internal ROM area in the overlapping part cannot be accessed.

Part Name	Internal ROM	Access Space	
		LOCATION 0 Instruction	LOCATION 0FH Instruction
μPD784224	96 k × 8 bits	00000H to 0F0FFH 10000H to 17FFFH	00000H to 17FFFH
μPD784225 μPD78F4225	128 k × 8 bits	00000H to 0EDFFH 10000H to 1FFFFH	00000H to 1FFFFH

The internal ROM can be accessed at high speed. Usually, a fetch is at the same speed as an external ROM fetch. By setting (to 1) the IFCH bit of the memory expansion mode register (MM), the high-speed fetch function is used. An internal ROM fetch is a high-speed fetch (fetch in two system clocks in 2-byte units).

If the instruction execution cycle similar to the external ROM fetch is selected, waits are inserted by the wait function. However, when a high-speed fetch is used, waits cannot be inserted for the internal ROM. However, do not set external waits for the internal ROM area. If an external wait is set for the internal ROM area, the CPU enters the deadlock state. The deadlock state is only released by a reset input.

$\overline{\text{RESET}}$ input causes an instruction execution cycle similar to the external ROM fetch cycle.

3.3 Base Area

The area from 0 to FFFFH becomes the base area. The base area is the target in the following uses.

- Reset entry address
- Interrupt entry address
- Entry address for CALLT instruction
- 16-bit immediate addressing mode (instruction address addressing)
- 16-bit direct addressing mode
- 16-bit register addressing mode (instruction address addressing)
- 16-bit register indirect addressing mode
- Short direct 16-bit memory indirect addressing mode

This base area is allocated in the vector table area, CALLT instruction table area, and CALLF instruction entry area.

When the LOCATION 0 instruction is executing, the internal data area is placed in the base area. Be aware that the program is not fetched from the internal high-speed RAM area and special function register (SFR) area in the internal data area. Also, use the data in the internal RAM area after initialization.

3.3.1 Vector table area

The 64-byte area from 00000H to 0003FH is reserved as the vector table area. The program start addresses for branching by interrupt requests and $\overline{\text{RESET}}$ input are stored in the vector table area. If context switching is used by each interrupt, the register bank number of the switch destination is stored.

The portion that is not used as the vector table can be used as program memory or data memory.

The values written in the vector table are a 16-bit values. Therefore, branching can only be to the base area.

Table 3-1. Vector Table Address

Interrupt Source	Vector Table Address	Interrupt Source	Vector Table Address
BRK instruction	003EH	INTST1	001CH
Operand error	003CH	INTSER2	001EH
NMI	0002H	INSR2	0020H
INTWDT (non-maskable)	0004H	INTCSI2	
INTWDT (maskable)	0006H	INTST2	0022H
INTP0	0008H	INTTM3	0024H
INTP1	000AH	INTTM00	0026H
INTP2	000CH	INTTM01	0028H
INTP3	000EH	INTTM1	002AH
INTP4	0010H	INTTM2	002CH
INTP5	0012H	INTAD	002EH
INTIIC0[Note]	0016H	INTTM5	0030H
INTCSI0		INTTM6	0032H
INTSER1	0018H	INTWT	0038H
INTSR1	001AH		
INTCSI1			

Note Only in μPD784225Y Subseries

3.3.2 CALLT instruction table area

The 64-Kbyte area from 00040H to 0007FH can store the subroutine entry addresses for the 1-byte call instruction (CALLT).

In a CALLT instruction, this table is referenced and the base area address written in the table is branched to as the subroutine. Since a CALLT instruction is one byte, many subroutine call descriptions in the program can be CALLT instructions, so the object size of the program can be reduced. Since a maximum of 32 subroutine entry addresses can be described in the table, they should be registered in order from the most frequently described.

When not used as the CALLT instruction table, the area can be used as normal program memory or data memory.

3.3.3 CALLF instruction entry area

The area from 00800H to 00FFFH can be for direct subroutine calls in the 2-byte call instruction (CALLF).

Since a CALLF instruction is a 2-byte call instruction, compared to when using the CALL instruction (3 bytes or 4 bytes) of a direct subroutine call, the object size can be reduced.

When you want to achieve high speed, describing direct subroutines in this area is effective.

If you want to decrease the object size, an unconditional branch (BR) is described in this area, and the actual subroutine is placed outside of this area. When a subroutine is called from five or more locations, reducing the object size is attempted. In this case, since only a 4-byte location for the BR instruction is used in the CALLF entry area, the object size of many subroutines can be reduced.

3.4 Internal Data Space

The internal data space consists of the internal RAM space and the special function register area (see **Figures 3-1** and **3-2**).

The final address in the internal data area can be set to 0FFFFH (when executing the LOCATION 0 instruction) or FFFFFH (when executing the LOCATION 0FH instruction) by the LOCATION instruction. The address selection of the internal data area by this LOCATION 0 must be executed once immediately after a reset is cleared. After one selection, updating is not possible. The program following a reset clear must be as shown in the example. If the internal data area and another area are allocated to the same address, the internal data area becomes the access target, and the other area cannot be accessed.

Example

```
RSTVCT      CSEG AT 0
            DW   RSTSTRT

             to

INITSEG     CSEG  BASE
RSTSTRT:    LOCATION 0H ; or LOCATION 0FH
            MOVG SP, #STKBGN
```

Caution When the LOCATION 0 instruction is executing, the program after clearing the reset must not overlap the internal data area. In addition, make sure the entry address of the servicing routine for a nonmaskable interrupt such as NMI does not overlap the internal data area. The entry area for a maskable interrupt must be initialized before referencing the internal data area.

3.4.1 Internal RAM space

The μPD784225 has an on-chip general-purpose static RAM.

This space has the following configuration.

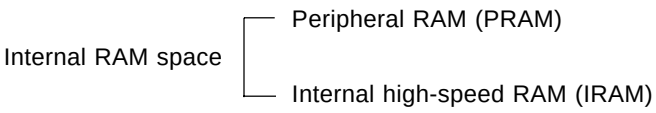


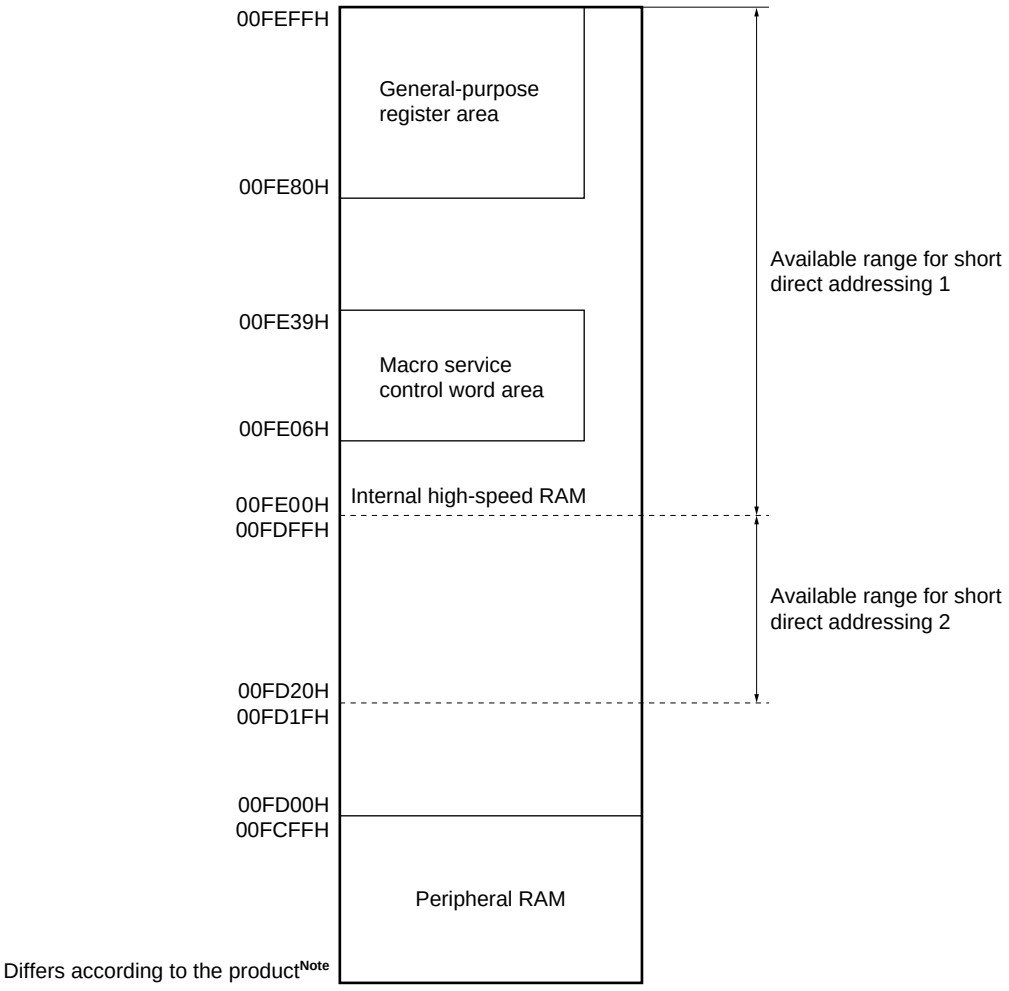
Table 3-2. Internal RAM Space List

Internal RAM / Product Name	Internal RAM Space		
		Peripheral RAM: PRAM	Internal High-speed RAM: IRAM
μPD784224	3,584 bytes (0F100H to 0FEFFH)	3,073 bytes (0F100H to 0FCFFH)	512 bytes (0FD00H to 0FEFFH)
μPD784225 μPD78F4225	4,352 bytes (0EE00H to 0FEFFH)	3,840 bytes (0EE00H to 0FCFFH)	

Remark The addresses in the table are the values when the LOCATION 0 instruction is executing. When the LOCATION 0FH instruction is executing, 0F0000H is added to the above values.

Figure 3-3 is the internal RAM memory map.

Figure 3-3. Internal RAM Memory Map



Note μPD784224 : 00F100H
μPD784225, 78F4225 : 00EE00H

Remark The addresses in the figure are the values when the LOCATION 0 instruction is executing. When the LOCATION 0FH instruction is executing, 0F0000H is added to the above values.

(1) Internal high-speed RAM (IRAM)

The internal high-speed RAM can be accessed at high speed. FD20H to FEFFH can use the short direct addressing mode for high-speed access. The two short direct addressing modes are short direct addressing 1 and short direct addressing 2 that are based on the address of the target. Both addressing modes have the same function. In a portion of the instructions, short direct addressing 2 has a shorter word length than short direct addressing 1. For details, see **78K/IV Series User's Manual, Instruction (U10905E)**.

A program cannot be fetched from IRAM. If a program is fetched from an address that is mapped by IRAM, the CPU runs wild.

The following areas are reserved in IRAM.

- General - purpose register area : FE80H - FEFFH
- Macro service control word area : FE06H - FE39H
- Macro service channel area : FE00H - FEFFH (The address is set by a macro service control word.)

When reserved functions are not used in these areas, they can be used as normal data memory.

Remark The addresses in this text are the addresses when the LOCATION 0 instruction is executing. When the LOCATION 0FH instruction is executing, 0F0000H is added to the values in this text.

(2) Peripheral RAM (PRAM)

The peripheral RAM (PRAM) is used as normal program memory or data memory. When used as the program memory, the program must be written beforehand in the peripheral RAM by a program.

A program fetch from the peripheral RAM is high speed and can occur in two clocks in 2-byte units.

3.4.2 Special function register (SFR) area

The special function register (SFR) of the on-chip peripheral hardware is mapped to the area from 0FF00H to 0FFFFH (Refer to **Figures 3-1** to **3-2**).

The area from 0FFD0H to 0FFDFH is mapped as the external SFR area. Peripheral I/O externally connected in the external memory expansion mode (set by the memory expansion mode register (MM)) can be accessed.

Caution In this area, do not access an address that is not mapped in SFR. If mistakenly accessed, the CPU enters the deadlock state. The deadlock state is released only by reset input.

Remark The addresses in this text are the addresses only when the LOCATION 0 instruction is executing. If the LOCATION 0FH instruction is executing, 0F0000H is added to the values in the text.

3.4.3 External SFR area

In the products of the μPD784225 subseries, the 16-byte area of the 0FFD0H to 0FFDFH area (during LOCATION 0 instruction execution, or 0FFFD0H to 0FFFDFH area during LOCATION 0FH instruction execution) in the SFR area is mapped as the external SFR area. In the external memory expansion mode, the address bus and address/data bus are used and the externally attached peripheral I/O can be accessed.

Since the external SFR area can be accessed by SFR addressing, the features are that peripheral I/O operations can be simplified; the object size can be reduced; and macro service can be used.

The bus operation when accessing an external SFR area is the same as a normal memory access.

3.5 External Memory Space

The external memory space is the memory space that can be accessed based on the setting of the memory expansion mode register (MM). The program and table data can be stored and peripheral I/O devices can be assigned.

3.6 μPD78F4225 Memory Mapping

The μPD78F4225 has a 128-Kbyte flash memory and 4,352-byte internal RAM.

The μPD78F4225 has a function (memory size switching function) so that a part of the internal memory is not used by the software.

The memory size is switched by the internal memory size switching register (IMS).

Based on the IMS setting, the memory mapping can be the same memory mapping as the mask ROM products with different internal memories (ROM, RAM).

IMS can only be written by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets the register to FFH.

Figure 3-4. Internal Memory Size Switching Register (IMS) Format

Address: 0FFFCH After Reset: FFH W

Symbol	7	6	5	4	3	2	1	0
IMS	1	1	ROM1	ROM0	1	1	RAM1	RAM0

ROM1	ROM0	Internal ROM Capacity Selections
0	0	48 Kbytes
0	1	64 Kbytes
1	0	96 Kbytes
1	1	128 Kbytes

RAM1	RAM0	Internal RAM Capacity Selections
0	0	1536 bytes
0	1	2304 bytes
1	0	3072 bytes
1	1	3840 bytes

Caution Mask ROM products (μPD784224, 784225) do not have an IMS.

Table 3-3 shows the IMS settings that have the same memory map as the mask ROM products.

Table 3-3. Settings of the Internal Memory Size Switching Register (IMS)

Target Mask ROM Products	IMS Settings
μPD784224	EEH
μPD784225	FFH

3.7 Control Registers

The control registers are the program counter (PC), program status word (PSW), and stack pointer (SP).

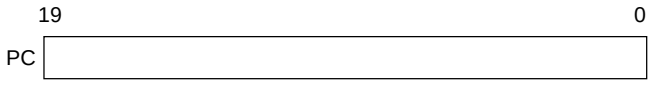
3.7.1 Program counter (PC)

This is a 20-bit binary counter that saves address information about the program to be executed next (see **Figure 3-5**).

Usually, this counter is automatically incremented based on the number of bytes in the instruction to be fetched. When the instruction that is branched is executed, the immediate data or register contents are set.

$\overline{\text{RESET}}$ input sets the low-order 16 bits of the PC to the 16-bit data at addresses 0 and 1, and 0000 in the high-order four bits of the PC.

Figure 3-5. Program Counter (PC) Format



3.7.2 Program status word (PSW)

The program status word (PSW) is a 16-bit register that consists of various flags that are set and reset based on the result of the instruction execution.

A read or write access is performed in high-order 8 bit (PSWH) and the low-order 8 bits (PSWL) units. In addition, bit manipulation instructions can manipulate each flag.

The contents of the PSW are automatically saved on the stack when a vectored interrupt request is accepted and when a BRK instruction is executed, and are automatically restored when a RETI or RETB instruction is executed. When context switching is used, the contents are automatically saved to PR3, and automatically restored when a RETCS or RETCSB instruction is executed.

$\overline{\text{RESET}}$ input resets all of the bits to 0.

Always write 0 in the bits indicated by “0” in Figure 3-6. The contents of bits indicated by “-” are undefined when read.

Figure 3-6. Program Status Word (PSW) Format

Symbol	7	6	5	4	3	2	1	0
PSWH	UF	RBS2	RBS1	RBS0	—	—	—	—

	7	6	5	4	3	2	1	0
PSWL	S	Z	RSS	AC	IE	P/V	0	CY

Each flag is described below.

(1) Carry flag (CY)

This is the flag that stores the carry or borrow of an operation result.

When a shift rotate instruction is executed, the shifted out value is stored. When a bit manipulation instruction is executed, this flag functions as the bit accumulator.

The CY flag state can be tested by a conditional branch instruction.

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

The P/V flag has the following two actions in accordance with the execution of the operation instruction.

The state of the P/V flag can be tested by a conditional branch instruction.

- **Parity flag action**

  The results of executing the logical instructions, shift rotate instructions, and CHKL and CHKLA instructions are set to 1 when an even number of bits is set to 1. If the number of bits is odd, the result is reset to 0. However, for 16-bit shift instructions, the parity flag from only the low-order 8 bits of the operation result is valid.

- **Overflow flag action**

  The result of executing an arithmetic operation instruction is set to 1 only when the numerical range expressed in two's complement is exceeded. Otherwise, the result is reset to 0. Specifically, the result is the exclusive or of the carry from the MSB and the carry to the MSB and becomes the flag contents. For example, in 8-bit arithmetic operations, the two's complement range is 80H (–128) to 7FH (+127). If the operation result is outside this range, the flag is set to 1. If inside the range, it is reset to 0.

Example The action of the overflow flag when an 8-bit addition instruction is executed is described next.

When 78H (+120) and 69H (+105) are added, the operation result becomes E1H (+225). Since the upper limit of two's complement is exceeded, the P/V flag is set to 1. In a two's complement expression, E1H becomes –31.

```
   78H (+120) =     0111  1000
+) 69H (+105) = +) 0110  1001
                    0  1110  0001 = –31 P/V = 1
                    ↑
                   CY
```

Next, since the operation result of the addition of the following two negative numbers falls within the two's complement range, the P/V flag is reset to 0.

```
   FBH (–5)   =     1111  1011
+) F0H (–16)  = +) 1111  0000
                    1  1110  1011 = –21 P/V = 0
                    ↑
                   CY
```

(3) Interrupt request enable flag (IE)

This flag controls the CPU interrupt request acceptance.

If IE is 0, interrupts are disabled, and only nonmaskable interrupts and unmasked macro services can be accepted. Otherwise, everything is disabled.

If IE is 1, the interrupt enable state is entered. Enabling the acceptance of interrupt requests is controlled by the interrupt mask flags that correspond to each interrupt request and the priority of each interrupt.

This flag is set to 1 by executing the EI instruction and is reset to 0 by executing the DI instruction or accepting an interrupt.

(4) Auxiliary carry flag (AC)

If the operation result has a carry from bit 3 or a borrow to bit 3, this flag is set to 1. Otherwise, the flag is reset to 0.

This flag is used when the ADJBA and ADJBS instructions are executing.

(5) Register set selection flag (RSS)

This flag sets the general-purpose registers that function as X, A, C, and B and the general-purpose register pairs (16 bits) that function as AX and BC.

This flag is used to maintain compatibility with the 78K/III series. Always set this flag to 0 except when using a 78K/III series program.

(6) 0 flag (Z)

This flag indicates that the operation result is 0.

If the operation result is 0, this flag is set to 1. Otherwise, it is reset to 0. The state of the Z flag can be tested by conditional branch instructions.

(7) Sign flag (S)

This flag indicates that the MSB in the operation result is 1.

The flag is set to 1 when the MSB of the operation result is 1. If 0, the flag is reset to 0. The S flag state can be tested by the conditional branch instructions.

(8) Register bank selection flags (RBS0 - RBS2)

This is the 3-bit flag that selects one of the eight register banks (register banks 0 to 7). (refer to **Table 3-4**.)

Three bit information that indicates the register bank selected by executing the SEL RBn instruction is stored.

Table 3-4. Register Bank Selection

RBS2	RBS1	RBS0	Set Register Bank
0	0	0	Register bank 0
0	0	1	Register bank 1
0	1	0	Register bank 2
0	1	1	Register bank 3
1	0	0	Register bank 4
1	0	1	Register bank 5
1	1	0	Register bank 6
1	1	1	Register bank 7

(9) User flag (UF)

This flag is set and reset by a user program and can be used in program control.

3.7.3 Using the RSS bit

Basically, always use with the RSS bit fixed at 0.

The following descriptions discuss using a 78K/III series program and a program that sets the RSS bit to 1. Reading is not necessary if the RSS bit is fixed at 0.

The RSS bit enables the functions in A (R1), X (R0), B (R3), C (R2), AX (RP0), and BC (RP1) to also be used in registers R4 to R7 (RP2, RP3). When this bit is effectively used, efficient programs in terms of program size and program execution can be written.

Sometimes, however, unexpected problems arise if used carelessly. Consequently, always set the RSS bit to 0. Use with the RSS bit set to 1 only when 78K/III series programs will be used.

By setting the RSS bit to 0 in all programs, writing and debugging programs become more efficient.

Even if a program where the RSS bit is set to 1 is used, when possible, it is recommended to use the program after modifying the program so that the RSS bit is not set to 1.

(1) Using the RSS bit

- Registers used in instructions where the A, X, B, C, and AX registers are directly described in the operand column of the operation list (see **28.2**)

- Registers that are implicitly specified in instructions that use the A, AX, B, and C registers by implied addressing

- Registers that are used in addressing in instructions that use the A, B, and C registers in indexed addressing and based indexed addressing

The registers used in these cases are switched in the following ways by the RSS bit.

- When RSS = 0
  A→R1, X→R0, B→R3, C→R2, AX→RP0, BC→RP1

- When RSS = 1
  A→R5, X→R4, B→R7, C→R6, AX→RP2, BC→RP3

The registers used in other cases always become the same registers regardless of the contents of the RSS bit. For registers A, X, B, C, AX, and BC in NEC assembler RA78K4, instruction code is generated for any register described by name or for registers set by an RSS pseudo instruction in the assembler.
When the RSS bit is set or reset, always specify an RSS pseudo instruction immediately before (or immediately after) that instruction (see the following examples).

<Program examples>

- When RSS = 0

```
RSS 0          ; RSS pseudo instruction
CLR1 PSWL. 5
MOV B, A       ; This description corresponds to “MOV R3, R1”.
```

- When RSS = 1

```
RSS 1          ; RSS pseudo instruction
SET1 PSWL. 5
MOV B, A       ; This description corresponds to “MOV R7, R5”.
```

(2) Generation of instruction code in the RA78K4

- In the RA78K4, when an instruction with the same function as an instruction that directly specifies A or AX in the operand column in the operation list of the instruction is used, the instruction code that directly describes A or AX in the operand column is given priority and generated.

Example The MOV A,r instruction where r is B has the same function as the MOV r, r’ instruction where r is A and r’ is B. In addition, they have the same (MOV A,B) description in the assembler source program. In this case, RA78K4 generates code that corresponds to the MOV A, r instruction.

- If A, X, B, C, AX, or BC is described in an instruction that specifies r, r', rp, or rp' in the operand column, the A, X, B, C, AX, or BC instruction code generates the instruction code that specifies the following registers based on the operand of the RSS pseudo instruction in RA78K4.

Register	RSS = 0	RSS = 1
A	R1	R5
X	R0	R4
B	R3	R7
C	R2	R6
AX	RP0	RP2
BC	RP1	RP3

- If R0 to R7 and RP0 to RP4 are specified in r, r', rp, and rp' in the operand column, an instruction code that conforms to the specification is output. (Instruction code that directly describes A or AX in the operand column is not output.)

- The A, B, and C registers that are used in indexed addressing and based indexed addressing cannot be described as R1, R3, R2, or R5, R7, R6.

(3) Usage Warnings

Switching the RSS bit obtains the same effect as holding two register sets. However, be careful and write the program so that implicit descriptions in the program and dynamically changing the RSS bit during program execution always agree.

Also, since a program with RSS = 1 cannot be used in a program that uses context switching, the portability of the program becomes poor. Furthermore, since different registers having the same name are used, the readability of the program worsens, and debugging becomes difficult. Therefore, when RSS = 1 must be used, write the program while taking these problems into consideration.

A register that does not have the RSS bit set can be accessed by specifying the absolute name.

3.7.4 Stack pointer (SP)

The 24-bit register saves the starting address of the stack (LIFO: 00000H to FFFFFFH) (refer to **Figure 3-7**). The stack is used for addressing during subroutine processing or interrupt servicing. Always set the most-significant four bits to zero.

The contents of the SP are decremented before writing to the stack area and incremented after reading from the stack (refer to **Figures 3-8** and **3-9**).

SP is accessed by special instructions.

Since the SP contents become undefined when $\overline{\text{RESET}}$ is input, always initialize the SP from the initialization program immediately after clearing the reset (before accepting a subroutine call or interrupt).

Example Initializing SP

```
MOVG SP,#0FEE0H   ; SP ← 0FEE0H (when used from FEDFH)
```

Figure 3-7. Stack Pointer (SP) Format

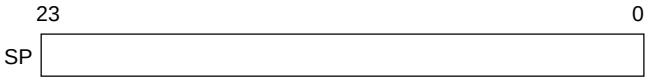


Figure 3-8. Data Saved to the Stack

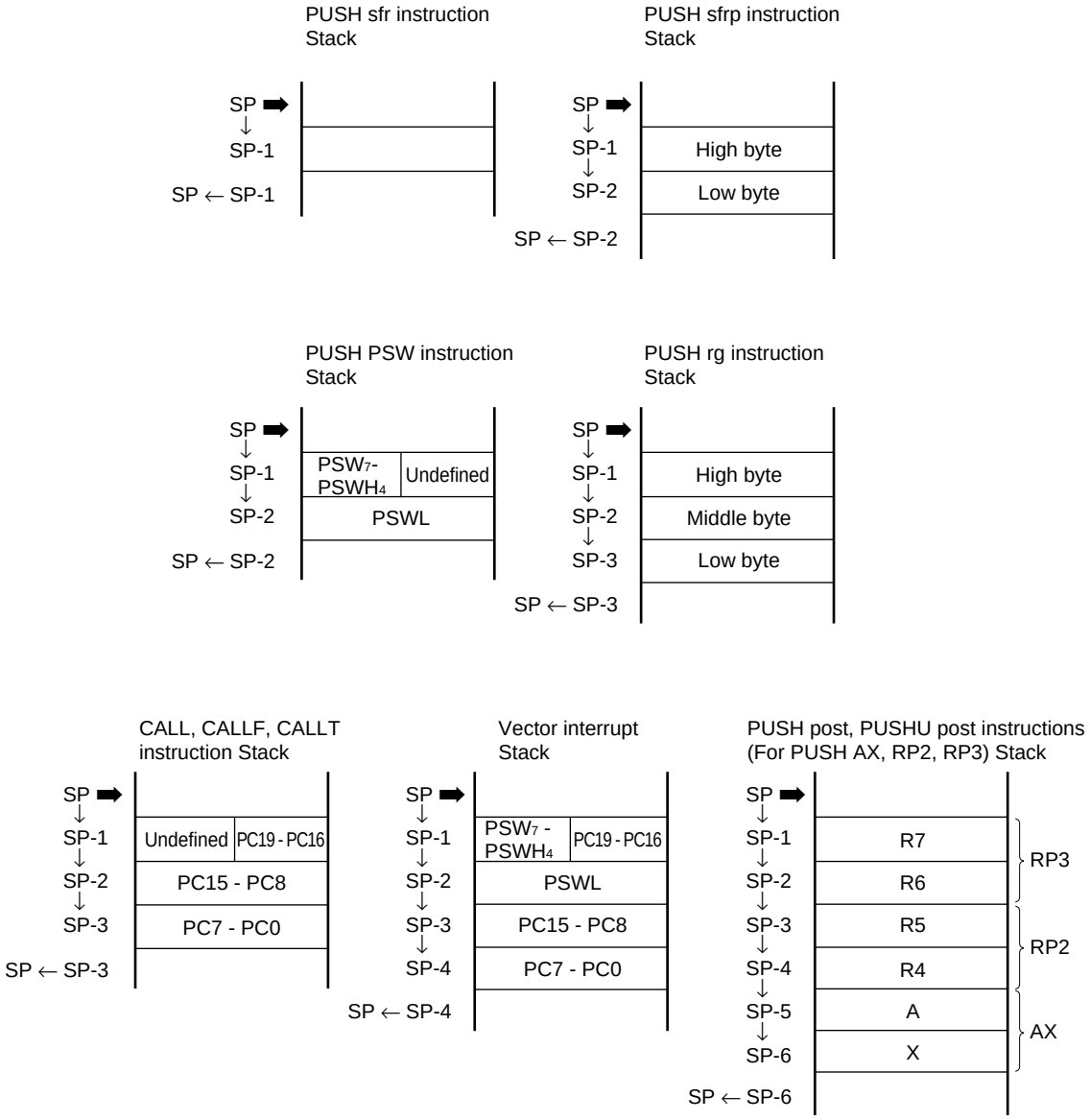
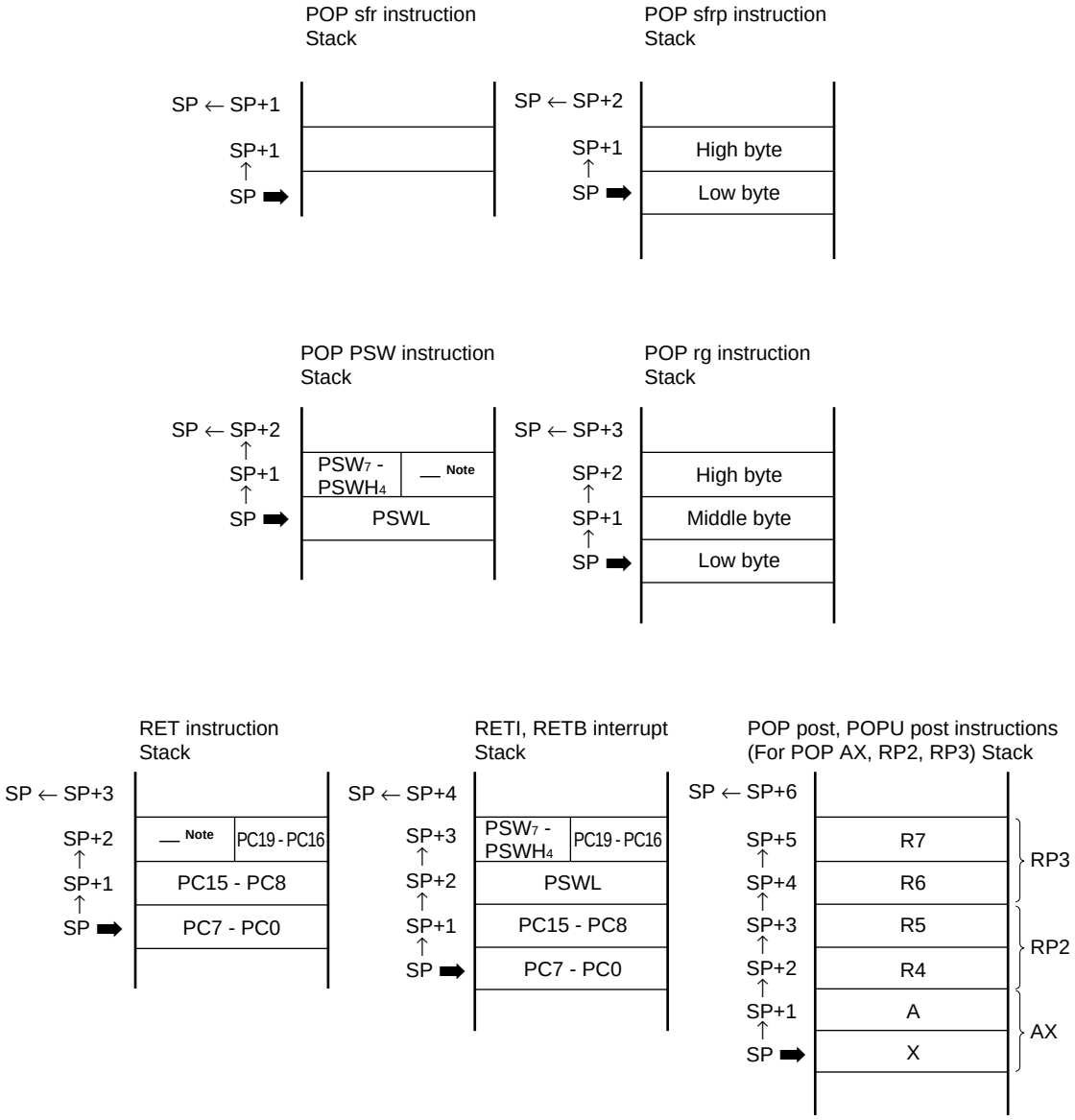


Figure 3-9. Data Restored from the Stack



Note This 4-bit data is ignored.

Cautions 1. In stack addressing, the entire 1-Mbyte space can be accessed, but the stack cannot be guaranteed in the SFR area and internal ROM area.

2. The stack pointer (SP) becomes undefined when $\overline{\text{RESET}}$ is input. In addition, even when SP is in the undefined state, nonmaskable interrupts can be accepted. Therefore, when the SP is in the undefined state immediately after the reset is cleared and a request for a nonmaskable interrupt is generated, unexpected actions sometimes occur. To avoid this danger, always specify the following in the program after clearing a reset.

```
RSTVCT    CSEG AT 0
          DW   RSTSTRT

            to

INITSEG   CSEG BASE
RSTSTRT:  LOCATION 0H; or LOCATION 0FH
          MOVG SP, #STKBGN
```

3.8 General-Purpose Registers

3.8.1 Structure

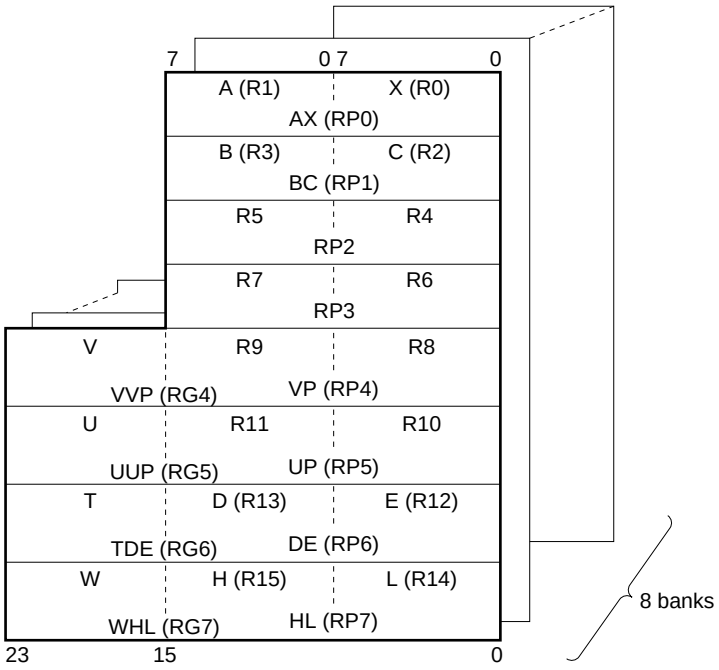
There are sixteen 8-bit general-purpose registers. In addition, two 8-bit general-purpose registers can be combined and used as a 16-bit general-purpose register. Furthermore, four of the 16-bit general-purpose registers are combined with an 8-bit register for address expansion and used as a 24-bit address specification register.

The general-purpose registers except for the V, U, T, and W registers for address expansion are mapped to the internal RAM.

These register sets provide eight banks and can be switched by the software or context switching.

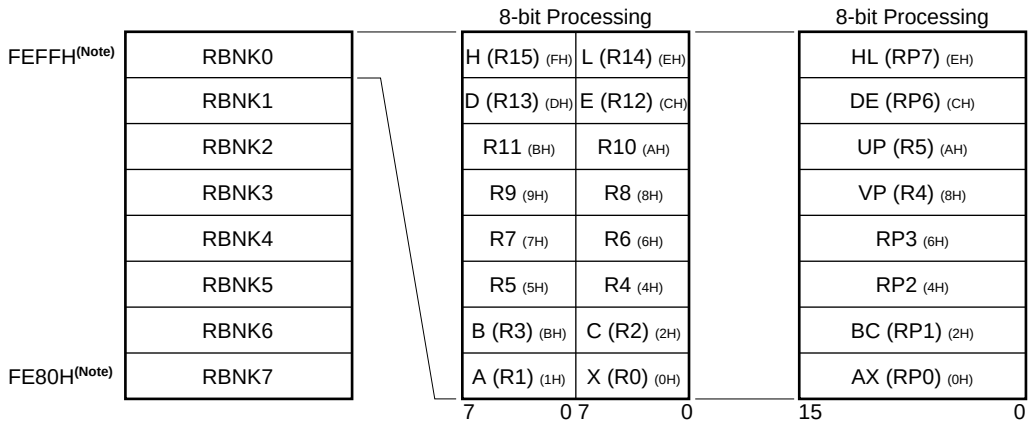
$\overline{\text{RESET}}$ input selects register bank 0. In addition, the register banks that are used in an executing program can be verified by reading the register bank selection flags (RBS0, RBS1, RBS2) in the PSW.

Figure 3-10. General-purpose Register Format



Remark The parentheses enclose the absolute names.

Figure 3-11. General-purpose Register Addresses

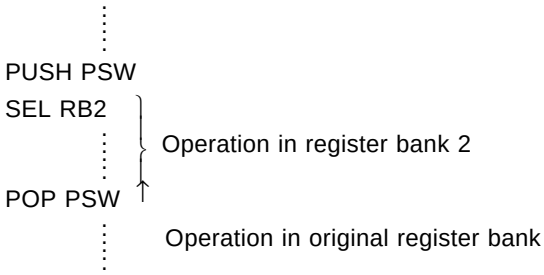


Note These are the addresses when the LOCATION 0 instruction is executing. The addresses when the LOCATION 0FH instruction is executed are the sum of the above values and 0F0000H.

Caution R4, R5, R6, R7, RP2, and RP3 can be used as the X, A, C, B, AX, and BC registers when the RSS bit in the PSW is set to 1. However, use this function only when using a 78K/III series program.

Remark When changing the register bank and when returning to the original register bank is necessary, execute the SEL RBn instruction after using the PUSH PSW instruction to save the PSW to the stack. If the stack position is not changed when returning to the original state, the POP PSW instruction is used to return. When the register banks in the vectored interrupt processing program are updated, PSW is automatically saved on the stack when an interrupt is accepted and returned by the RETI and RETB instructions. Therefore, when one register bank is used in an interrupt servicing routine, only the SEL RBn instruction is executed, and the PUSH PSW or POP PSW instruction does not have to be executed.

Example When register bank 2 is specified



3.8.2 Functions

In addition to being manipulatable in 8-bit units, general-purpose registers can be a pair of two 8-bit registers and be manipulated in 16-bit units. Also four of the 16-bit registers can be combined with the 8-bit register for address expansion and manipulated in 24-bit units.

Each register can generally be used as the temporary storage for the operation result or the operand of the operation instruction between registers.

The area from 0FE80H to 0FEFFH (during LOCATION 0 instruction execution, or the 0FFE80H to 0FFEFFH during LOCATION 0FH instruction execution) can be accessed by specifying an address as normal data memory whether or not it is used as the general-purpose register area.

Since there are eight register banks in the 78K/IV series, efficient programs can be written by suitably using the register banks in normal processing or interrupt processing.

Each register has the unique functions shown below.

A (R1):

- This register is primarily for 8-bit data transfers and operation processing. It can be combined with all of the addressing modes for 8-bit data.
- This register can be used to store bit data.
- This register can be used as a register that stores the offset value during indexed addressing or based indexed addressing.

X (R0):

- This register can store bit data.

AX (RP0):

- This register is primarily for 16-bit data transfers and operation results. It can be combined with all of the addressing modes for 16-bit data.

AXDE:

- When a DIVUX, MACW, or MACSW instruction is executing, this register can be used to store 32-bit data.

B (R3):

- This register functions as a loop counter and can be used by the DBNZ instruction.
- This register can store the offset in indexed addressing and based indexed addressing.
- This register is used as the data pointer in a MACW or MACSW instruction.

C (R2):

- This register functions as a loop counter and can be used by the DBNZ instruction.
- This register can store the offset in based indexed addressing.
- This register is used as the counter in string and SACW instructions.
- This register is used as the data pointer in a MACW or MACSW instruction.

RP2:

- When context switching is used, this register saves the low-order 16 bits of the program counter (PC).

RP3:

- When context switching is used, this register saves the most significant 4 bits of the program counter (PC) and the program status word (PSW) (except bits 0 to 3 in PSWH).

VVP (RG4):

- This register functions as a pointer and specifies the base address in register indirect addressing, based addressing, and based indirect addressing.

UUP (RG5):

- This register functions as a user stack pointer and implements another stack separate from the system stack by the PUSHU and POPU instructions.
- This register functions as a pointer and acts as the register that specifies the base address during register indirect addressing and based addressing.

DE (RP6), HL (RP7):

- This register stores the offset during indexed addressing and based indexed addressing.

TDE (RG6):

- This register functions as a pointer and sets the base address in register indirect addressing and based addressing.
- This register functions as a pointer in string and SACW instructions.

WHL (RG7):

- This register primarily performs 24-bit data transfers and operation processing.
- This register functions as a pointer and specifies the base address during register indirect addressing or based addressing.
- This functions as a pointer in string and SACW instructions.

In addition to its function name (X, A, C, B, E, D, L, H, AX, BC, VP, UP, DE, HL, VVP, UUP, TDE, WHL) that emphasizes its unique function, each register can be described by its absolute name (R0 to R15, RP0 to RP7, RG4 to RG7). For the correspondence, refer to **Table 3-5**.

Table 3-5. Correspondence between Function Names and Absolute Names

(a) 8-bit registers

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

(b) 16-bit registers

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

(c) 24-bit registers

Absolute Name	Function Name
RG4	VVP
RG5	UUP
RG6	TDE
RG7	WHL

Note Use RSS = 1 only when a 78K/III series program is used.

Remark R8 to R11 do not have function names.

3.9 Special Function Registers (SFRs)

These registers are assigned special functions such as the mode register and control register of the on-chip peripheral hardware and are mapped to the 256-byte area from 0FF00H to 0FFFFH[Note].

Note These are the addresses when the LOCATION 0 instruction is executing. They are FFF00H to FFFFFH when the LOCATION 0FH instruction is executing.

Caution In this area, do not access an address that is not allocated by an SFR. If erroneously accessed, the μPD784225 enters the deadlock state. The deadlock state is released only by reset input.

Table 3-6 shows the list of special function registers (SFRs). The meanings of the items are described next.

- Symbol ··· This symbol indicates the on-chip SFR. In NEC assembler RA78K4, this is a reserved word. In C compiler CC78K4, it can be used as an sfr variable by a #pragma sfr directive.
- R/W ··· Indicates whether the corresponding SFR can be read or written.
  R/W : Can read/write
  R : Read only
  W : Write only
- Bit manipulation unit ··· When the corresponding SFR is manipulated, the appropriate bit manipulation unit is indicated. An SFR that can manipulate 16 bits can be described in the sfrp operand. If specified by an address, an even address is described.
  An SFR that can manipulate one bit can be described in bit manipulation instructions.
- After Reset ··· Indicates the state of each register when $\overline{\text{RESET}}$ is input.

Table 3-6. Special Function Register (SFR) List (1/4)

Address Note 1	Name of Special Function Register (SFR)	Symbol	R/W	Bit Manipulation Unit			After Reset
				1 bit	8 bits	16 bits	
0FF00H	Port 0	P0	R/W	○	○	—	00H Note 2
0FF01H	Port 1	P1	R	○	○	—	
0FF02H	Port 2	P2	R/W	○	○	—	
0FF03H	Port 3	P3		○	○	—	
0FF04H	Port 4	P4		○	○	—	
0FF05H	Port 5	P5		○	○	—	
0FF06H	Port 6	P6		○	○	—	
0FF07H	Port 7	P7		○	○	—	
0FF0CH	Port 12	P12		○	○	—	
0FF0DH	Port 13	P13		○	○	—	
0FF10H	16-bit timer register	TM0	R	—	—	○	0000H
0FF11H							
0FF12H	Capture/compare register 00 (16-bit timer/counter)	CR00	R/W	—	—	○	
0FF13H							
0FF14H	Capture/compare register 00 (16-bit timer/counter)	CR01		—	—	○	
0FF15H							
0FF16H	Capture/compare control register 0	CRC0		○	○	—	00H
0FF18H	16-bit timer mode control register	TMC0		○	○	—	
0FF1AH	16-bit timer output control register	TOC0		○	○	—	
0FF1CH	Prescaler mode register 0	PRM0		○	○	—	
0FF20H	Port 0 mode register	PM0		○	○	—	FFH
0FF22H	Port 2 mode register	PM2		○	○	—	
0FF23H	Port 3 mode register	PM3		○	○	—	
0FF24H	Port 4 mode register	PM4		○	○	—	
0FF25H	Port 5 mode register	PM5		○	○	—	
0FF26H	Port 6 mode register	PM6		○	○	—	
0FF27H	Port 7 mode register	PM7		○	○	—	
0FF2CH	Port 12 mode register	PM12		○	○	—	
0FF2DH	Port 13 mode register	PM13		○	○	—	
0FF30H	Pullup resistor option register 0	PU0		○	○	—	00H
0FF32H	Pullup resistor option register 2	PU2		○	○	—	
0FF33H	Pullup resistor option register 3	PU3		○	○	—	
0FF37H	Pullup resistor option register 7	PU7		○	○	—	

Notes 1. These values are when the LOCATION 0 instruction is executing. When the LOCATION 0FH instruction is executing, F0000H is added to these values.

2. Since each port is initialized in the input mode by a reset, in fact, 00H is not read out. The output latch is initialized to 0.

Table 3-6. Special Function Register (SFR) List (2/4)

Address Note 1	Name of Special Function Register (SFR)	Symbol		R/W	Bit Manipulation Unit			After Reset
					1 bit	8 bits	16 bits	
0FF3CH	Pullup resistor option register 12	PU12		R/W	○	○	—	00H
0FF40H	Clock output control register	CKS			○	○	—	
0FF42H	Port function control register Note 2	PF2			○	○	—	
0FF4EH	Pullup resistor option register	PUO			○	○	—	
0FF50H	8-bit timer register 1	TM1	TW1W	R	—	○	○	0000H
0FF51H	8-bit timer register 2	TM2			—	○		
0FF52H	Compare register 10 (8-bit timer/counter 1)	CR10	CR1W	R/W	—	○	○	
0FF53H	Compare register 20 (8-bit timer/counter 2)	CR20			—	○		
0FF54H	8-bit timer mode control register 1	TMC1	TMC1W		○	○	○	
0FF55H	8-bit timer mode control register 2	TMC2			○	○		
0FF56H	Prescaler mode register 1	PRM1	PRM1W		—	○	○	
0FF57H	Prescaler mode register 2	PRM2			—	○		
0FF60H	8-bit timer register 5	TM5	TM5W	R	—	○	○	
0FF61H	8-bit timer register 6	TM6			—	○		
0FF64H	Compare register 50 (8-bit timer/counter 5)	CR50	CR5W	R/W	—	○	○	
0FF65H	Compare register 60 (8-bit timer/counter 6)	CR60			—	○		
0FF68H	8-bit timer mode control register 5	TMC5	TMC5W		○	○	○	
0FF69H	8-bit timer mode control register 6	TMC6			○	○		
0FF6CH	Prescaler mode register 5	PRM5	PRM5W		—	○	○	
0FF6DH	Prescaler mode register 6	PRM6			—	○		
0FF70H	Asynchronous serial interface mode register 1	ASIM1			○	○	—	00H
0FF71H	Asynchronous serial interface mode register 2	ASIM2			○	○	—	
0FF72H	Asynchronous serial interface status register 1	ASIS1		R	○	○	—	
0FF73H	Asynchronous serial interface status register 2	ASIS2			○	○	—	
0FF74H	Transmission shift register 1	TXS1		W	—	○	—	FFH
	Reception buffer register 1	RXB1		R	—	○	—	
0FF75H	Transmission shift register 2	TXS2		W	—	○	—	
	Reception buffer register 2	RXB2		R	—	○	—	
0FF76H	Baud rate generator control register 1	BRGC1		R/W	○	○	—	00H
0FF77H	Baud rate generator control register 2	BRGC2			○	○	—	
0FF7AH	Oscillation mode selection register	CC			○	○	—	
0FF80H	A/D converter mode register	ADM			○	○	—	
0FF81H	A/D converter input selection register	ADIS			○	○	—	
0FF83H	A/D conversion result register	ADCR		R	—	○	—	Undefined

Notes 1. These are the values when the LOCATION 0 instruction is executing. When the LOCATION 0FH instruction is executing, F0000H is added to this value.

2. Only for the μPD784225Y Subseries

Table 3-6. Special Function Register (SFR) List (3/4)

Address Note 1	Name of Special Function Register (SFR)	Symbol		R/W	Bit Manipulation Unit			After Reset
					1 bit	8 bits	16 bits	
0FF84H	D/A conversion value setting register 0	DACS0		R/W	○	○	—	00H
0FF85H	D/A conversion value setting register 1	DACS1			○	○	—	
0FF86H	D/A converter mode register 0	DAM0			○	○	—	
0FF87H	D/A converter mode register 1	DAM1			○	○	—	
0FF88H	ROM correction control register	CORC			○	○	—	
0FF89H	ROM correction address register H	CORAH			—	○	—	
0FF8AH	ROM correction address register L	CORAL			—	—	○	0000H
0FF8BH								
0FF8DH	External access status enable register	EXAE			○	○	—	00H
0FF90H	Serial operating mode register 0	CSIM0			○	○	—	
0FF91H	Serial operating mode register 1	CSIM1			○	○	—	
0FF92H	Serial operating mode register 2	CSIM2			○	○	—	
0FF94H	Serial I/O shift register 0	SIO0			—	○	—	
0FF95H	Serial I/O shift register 1	SIO1			—	○	—	
0FF96H	Serial I/O shift register 2	SIO2			—	○	—	
0FF98H	Real-time output buffer register L	RTBL			—	○	—	
0FF99H	Real-time output buffer register H	RTBH			—	○	—	
0FF9AH	Real-time output port mode register	RTPM			○	○	—	
0FF9BH	Real-time output port control register	RTPC			○	○	—	
0FF9CH	Watch timer mode control register	WTM			○	○	—	
0FFA0H	External interrupt rising edge enable register	EGP0			○	○	—	
0FFA2H	External interrupt falling edge enable register	EGN0			○	○	—	
0FFA8H	In-service priority register	ISPR		R	○	○	—	00H
0FFA9H	Interrupt selection control register	SNMI		R/W	○	○	—	
0FFAAH	Interrupt mode control register	IMC			○	○	—	80H
0FFACH	Interrupt mask flag register 0L	MK0L	MK0		○	○	○	FFFFH
0FFADH	Interrupt mask flag register 0H	MK0H			○	○		
0FFAEH	Interrupt mask flag register 1L	MK1L	MK1		○	○	○	
0FFAFH	Interrupt mask flag register 1H	MK1H			○	○		
0FFB0H	I^2C bus control register Note 2	IICC0			○	○	—	00H
0FFB2H	Serial clock prescaler mode register Note 2	SPRM0			○	○	—	
0FFB4H	Slave address register Note 2	SVA0			—	○	—	
0FFB6H	I^2C bus status register Note 2	IICS0		R	○	○	—	
0FFB8H	Serial shift register Note 2	IIC0		R/W	○	○	—	

Notes 1. These are the values when the LOCATION 0 instruction is executing. When the LOCATION 0FH instruction is executing, F0000H is added to this value.

2. Only in the μPD784225Y Subseries.

Table 3-6. Special Function Register (SFR) List (4/4)

Address Note 1	Name of Special Function Register (SFR)	Symbol	R/W	Bit Manipulation Unit			After Reset
				1 bit	8 bits	16 bits	
0FFC0H	Standby control register	STBC	R/W	—	○	—	30H
0FFC2H	Watchdog timer mode register	WDM		—	○	—	00H
0FFC4H	Memory expansion mode register	MM		○	○	—	20H
0FFC7H	Programmable wait control register 1	PWC1		○	○	—	AAH
0FFCEH	Clock status register	PCS	R	○	○	—	32H
0FFCFH	Oscillation stabilizing time selection register	OSTS	R/W	○	○	—	00H
0FFD0H to 0FFDFH	External SFR area	—		○	○	—	—
0FFE0H	Interrupt control register (INTWDTM)	WDTIC		○	○	—	43H
0FFE1H	Interrupt control register (INTP0)	PIC0		○	○	—	
0FFE2H	Interrupt control register (INTP1)	PIC1		○	○	—	
0FFE3H	Interrupt control register (INTP2)	PIC2		○	○	—	
0FFE4H	Interrupt control register (INTP3)	PIC3		○	○	—	
0FFE5H	Interrupt control register (INTP4)	PIC4		○	○	—	
0FFE6H	Interrupt control register (INTP5)	PIC5		○	○	—	
0FFE8H	Interrupt control register (INTIIC0Note 2/INTCSI0)	CSIIC0		○	○	—	
0FFE9H	Interrupt control register (INTSER1)	SERIC1		○	○	—	
0FFEAH	Interrupt control register (INTSR1/INTCSI1)	SRIC1		○	○	—	
0FFEBH	Interrupt control register (INTST1)	STIC1		○	○	—	
0FFECH	Interrupt control register (INTSER2)	SERIC2		○	○	—	
0FFEDH	Interrupt control register (INTSR2/INTCSI2)	SRIC2		○	○	—	
0FFEEH	Interrupt control register (INTST2)	STIC2		○	○	—	
0FFEFH	Interrupt control register (INTTM3)	TMIC3		○	○	—	
0FFF0H	Interrupt control register (INTTM00)	TMIC00		○	○	—	
0FFF1H	Interrupt control register (INTTM01)	TMIC01		○	○	—	
0FFF2H	Interrupt control register (INTTM1)	TMIC1		○	○	—	
0FFF3H	Interrupt control register (INTTM2)	TMIC2		○	○	—	
0FFF4H	Interrupt control register (INTAD)	ADIC		○	○	—	
0FFF5H	Interrupt control register (INTTM5)	TMIC5		○	○	—	
0FFF6H	Interrupt control register (INTTM6)	TMIC6		○	○	—	
0FFF9H	Interrupt control register (INTWT)	WTIC		○	○	—	
0FFFCH	Internal memory size switching registerNote 3	IMS	W	○	○	—	FFH

Notes 1. These values are when the LOCATION 0 instruction is executing. When the LOCATION 0FH instruction is executing, F0000H is added to these values.

2. Only in the μPD784225Y Subseries

3. Only in the μPD78F4225 and 78F4225Y

3.10 Cautions

(1) Program fetches are not possible from the internal high-speed RAM space (when executing the LOCATION 0 instruction: 0FD00H - 0FEFFH, when executing the LOCATION 0FH instruction: FFD00H - FFEFFH)

(2) Special function register (SFR)
Do not access an address that is allocated to an SFR in the area from 0FF00H to 0FFFFH[Note]. If mistakenly accessed, the μPD784225 enters the deadlock state. The deadlock state is released only by $\overline{\text{RESET}}$ input.

Note These addresses are when the LOCATION 0 instruction is executing. They are FFF00H to FFFFFH when the LOCATION 0FH instruction is executing.

(3) Stack pointer (SP) operation
Although the entire 1-Mbyte space can be accessed by stack addressing, the stack cannot be guaranteed in the SFR area and the internal ROM space.

(4) Stack pointer (SP) initialization
The SP becomes undefined when $\overline{\text{RESET}}$ is input. Even after a reset is cleared, nonmaskable interrupts can be accepted. Therefore, the SP enters an undefined state immediately after clearing the reset. When a nonmaskable interrupt request is generated, unexpected operations sometimes occur. To minimize these dangers, always describe the following in the program immediately after clearing a reset.

```
RSTVCT    CSEG AT 0
          DW   RSTSTRT

           to

INITSEG   CSEG BASE
RSTSTRT:  LOCATION 0H; or LOCATION 0FH
          MOVG SP, #STKBGN
```

CHAPTER 4 CLOCK GENERATOR

4.1 Functions

The clock generator generates the clock to be supplied to the CPU and peripheral hardware. The following two types of system clock oscillators are available.

(1) Main system clock oscillator

This circuit oscillates at frequencies of 2 to 12.5 MHz. Oscillation can be stopped by executing the STOP instruction or setting the standby control register (STBC).

(2) Subsystem clock oscillator

This circuit oscillates at the frequency of 32.768 KHz. Oscillation cannot be stopped. If the subsystem clock oscillator is not used, not using the internal feedback resistance can be set by STBC. This enables the power consumption to be decreased in the STOP mode.

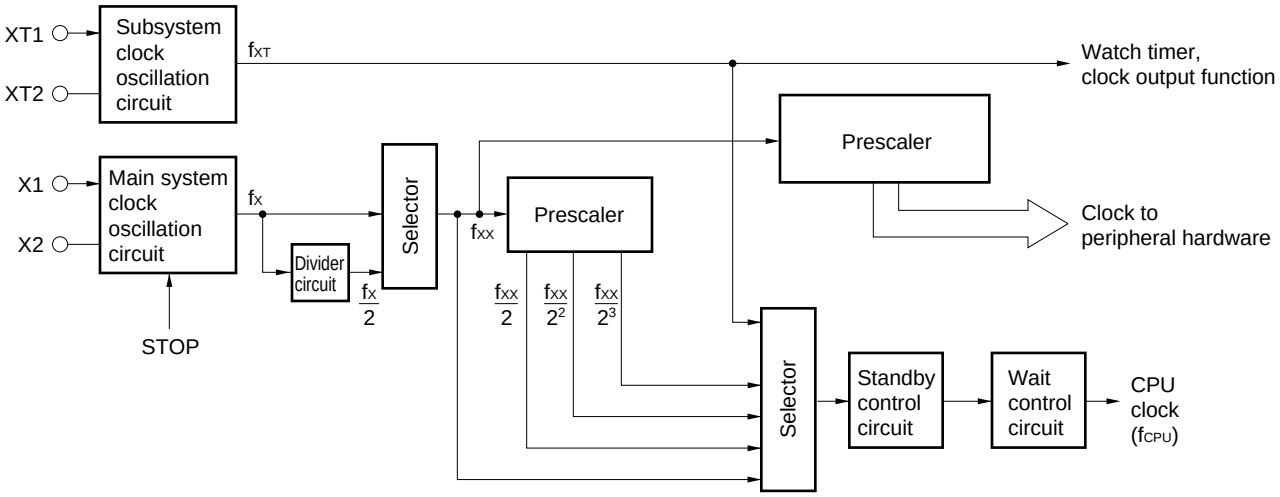
4.2 Configuration

The clock generator consists of the following hardware.

Table 4-1. Clock Generator Configuration

Item	Configuration
Control register	Standby control register (STBC) Oscillation mode selection register (CC) Clock status register (PCS) Oscillation stabilization time specification register (OSTS)
Oscillator	Main system clock oscillator Subsystem clock oscillator

Figure 4-1. Block Diagram of Clock Generator



4.3 Control Register

(1) Standby control register (STBC)

This register is used to set the standby mode and select internal system clock. For the details of the standby mode, refer to **CHAPTER 24 STANDBY FUNCTION**.

The write operation can be performed only using dedicated instructions to avoid entering into the standby mode due to an inadvertent program loop. These dedicated instructions, MOV STBC and #byte, have a special code structure (4 bytes). The write operation is performed only when the OP code of the 3rd byte and 4th byte are complements of each other. When the 3rd byte and 4th byte are not complements of each other, the write operation is not performed and an operand error interrupt is generated. In this case, the return address saved in the stack area indicates the address of the instruction that caused an error. Therefore, the address that caused an error can be determined from the return address that is saved in the stack area.

If a return from an operand error is performed simply with the RETB instruction, an infinite loop will be caused. Because the operand error interrupt occurs only in the case of an inadvertent program loop (if MOV STBC or #byte is described, only the correct dedicated instruction is generated in NEC's RA78K4 assembler), initialize the system for the program that processes an operand error interrupt.

Other write instructions such as MOV STBC, A, AND STBC, #byte, and SET1 STBC.7 are ignored and no operation is performed. In other words, neither is a write operation to STBC performed nor is an interrupt such as an operand error interrupt generated. STBC can be read out any time by means of a data transfer instruction.

$\overline{\text{RESET}}$ input sets STBC to 30H.

Figure 4-2 shows the format of STBC.

Figure 4-2. Standby Control Register (STBC) Format

Address: 0FFC0H After Reset: 30H R/W

Symbol	7	6	5	4	3	2	1	0
STBC	SBK	CK2	CK1	CK0	0	MCK	STP	HLT

SBK	Subsystem Clock Oscillation Control
0	Use oscillator (Internal feedback resistor is used.)
1	Stop oscillator (Internal feedback resistor is not used.)

CK2	CK1	CK0	CPU Clock Selection
0	0	0	f_{XX}
0	0	1	$f_{XX}/2$
0	1	0	$f_{XX}/4$
0	1	1	$f_{XX}/8$
1	×	×	f_{XT}

MCK	Main System Clock Oscillation Control
0	Use oscillator (Internal feedback resistor is used.)
1	Stop oscillator (Internal feedback resistor is not used.)

STP	HLT	Operation Specification Flag
0	0	Normal operation mode
0	1	HALT mode (Automatically cleared upon cancellation of HALT mode)
1	0	STOP mode (Automatically cleared upon cancellation of STOP mode)
1	1	IDLE mode (Automatically cleared upon cancellation of IDLE mode)

Cautions 1. When using the STOP mode during external clock input, make sure to set to 1 the EXTC bit of the oscillation stabilization time specification register (OSTS) before setting the STOP mode. If the STOP mode is used during external clock input when the EXTC bit of OSTS has been cleared (0), the μPD784225 may be damaged or its reliability may be impaired.

When setting to 1 the EXTC bit of OSTS, the clock with the opposite phase of the clock input to the X1 pin must be input to the X2 pin.

2. Perform the NOP instruction three times after a standby instruction (after standby release). Otherwise if contention arises between a standby instruction execution and an interrupt request, the standby instruction is not performed and the interrupt erquest is accepted after the execution of several instructions. The instructions executed before the interrupt request is accepted are instructions whose execution is started within 6 clocks maximum following execution of the standby instruction.

Example **MOV STBC #byte**
NOP
NOP
NOP
•
•
•

3. When CK2 = 0, the oscillation of the main system clock does not stop even if MCK is set to 1 (Refer to 4.5.1 Main system clock operation).

Remarks 1. f_{XX}: Main system frequency (f_X or $f_X/2$)
f_X: Main system clock oscillation frequency
f_{XT}: Subsystem clock oscillation frequency
2. ×: don't care

(2) Oscillation mode selection register (CC)

This register specifies whether clock output from the main system clock oscillator with the same frequency as the external clock, or clock output that is half of the original frequency is used to operate the internal circuit.
CC is set by a 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input sets CC to 00H.

Figure 4-3. Oscillation Mode Selection Register (CC) Format

Address: 0FF7AH After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
CC	ENMP	0	0	0	0	0	0	0

ENMP	Main System Clock Selection
0	Half of original oscillation frequency
1	Through rate clock mode

Cautions 1. If the subsystem clock is selected via the standby control mode register (STBC), the ENMP bit specification becomes invalid.
2. The ENMP bit cannot be reset by software. This bit is reset performing the system reset.

(3) Clock status register (PCS)

This register is a read-only 8-bit register that indicates the CPU clock operation status. By reading bit 2 and bits 4 to 7 of PCS, the relevant bit of the standby control register (STBC) can be read.

PCS is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PCS to 32H.

Figure 4-4. Clock Status Register (PCS) Format

Address: 0FFCEH After Reset: 32H R

Symbol	7	6	5	4	3	2	1	0
PCS	SBK	CK2	CK1	CK0	0	MCK	1	HLT

SBK	Feedback Resistor Status of Subsystem Clock
0	Internal feedback resistor is used.
1	Internal feedback resistor is not used.

CK2	CK1	CK0	CPU Clock Operating Frequency
0	0	0	f_{XX}
0	0	1	$f_{XX}/2$
0	1	0	$f_{XX}/4$
0	1	1	$f_{XX}/8$
1	×	×	f_{XT}

MCK	Oscillation Status of Main System Clock
0	Use oscillator
1	Stop oscillator

CST	CPU Clock Status
0	Main system clock operation
1	Subsystem clock operation

(4) Oscillation stabilization specification register (OSTS)

This register specifies the operation of the oscillator. Either a crystal/ceramic resonator or external clock is set to the EXTC bit in OSTS as the clock used. The STOP mode can be set even during external clock input only when the EXTC bit is set 1.

OSTS is set by a 1-bit or 8-bit transfer instruction.

$\overline{\text{RESET}}$ input sets OSTS to 00H.

Figure 4-5. Oscillation Stabilization Specification Register (OSTS) Format

Address: 0FFCFH After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
OSTS	EXTC	0	0	0	0	OSTS2	OSTS1	OSTS0

EXTC	External Clock Selection
0	Crystal/ceramic resonator is used
1	External clock is used

EXTC	OSTS2	OSTS1	OSTS0	Oscillation Stabilization Time
0	0	0	0	$2^{19}/f_{XX}$ (41.9 ms)
0	0	0	1	$2^{18}/f_{XX}$ (21.0 ms)
0	0	1	0	$2^{17}/f_{XX}$ (10.5 ms)
0	0	1	1	$2^{16}/f_{XX}$ (5.2 ms)
0	1	0	0	$2^{15}/f_{XX}$ (2.6 ms)
0	1	0	1	$2^{14}/f_{XX}$ (1.3 ms)
0	1	1	0	$2^{13}/f_{XX}$ (655 μs)
0	1	1	1	$2^{12}/f_{XX}$ (328 μs)
1	×	×	×	$512/f_{XX}$ (41.0 μs)

Cautions

1. **When a crystal/ceramic resonator is used, make sure to clear the EXTC bit to 0. If the EXTC bit is set to 1, oscillation stops.**
2. **When using the STOP mode during external clock input, make sure to set the EXTC bit to 1 before setting the STOP mode. If the STOP mode is used during external clock input when the EXTC bit of OSTS has been cleared, the μPD784225 may be damaged or its reliability may be impaired.**
3. **If the EXTC bit is set to 1 during external clock input, the opposite phase of the clock input to the X1 pin must be input to the X2 pin. If the EXTC bit is set to 1, the μPD784225 operates only with the clock input to the X2 pin.**

Remark

1. Figures in parentheses apply to operation with f_{XX} = 12.5 MHz.
2. ×: don't care

4.4 System Clock Oscillator

4.4.1 Main system clock oscillator

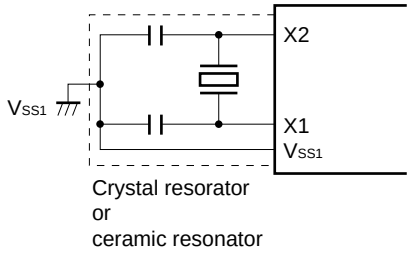
The main system clock oscillator oscillates with a crystal resonator or a ceramic resonator (standard: 12.5 MHz) connected to the X1 and X2 pins.

External clocks can be input to the main system clock oscillator. In this case, input a clock signal to the X1 pin and an antiphase clock signal to the X2 pin.

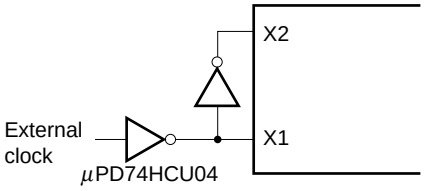
Figure 4-6 shows an external circuit of the main system clock oscillator.

Figure 4-6. External Circuit of Main System Clock Oscillator

(a) Crystal or ceramic oscillation



(b) External clock



4.4.2 Subsystem clock oscillator

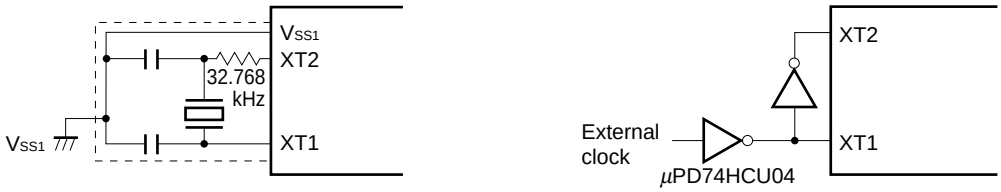
The subsystem clock oscillator oscillates with a crystal resonator (standard: 32.768 kHz) connected to the XT1 and XT2 pins.

External clocks can be input to the main system clock oscillator. In this case, input a clock signal to the XT1 pin and an antiphase clock signal to the XT2 pin.

Figure 4-7 shows an external circuit of the subsystem clock oscillator.

Figure 4-7. External Circuit of Subsystem Clock Oscillator

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



Cautions 1. When using a main system clock oscillator and a subsystem clock oscillator, carry out wiring in the broken line area in Figures 4-6 and 4-7 to prevent any effects from wiring capacities.

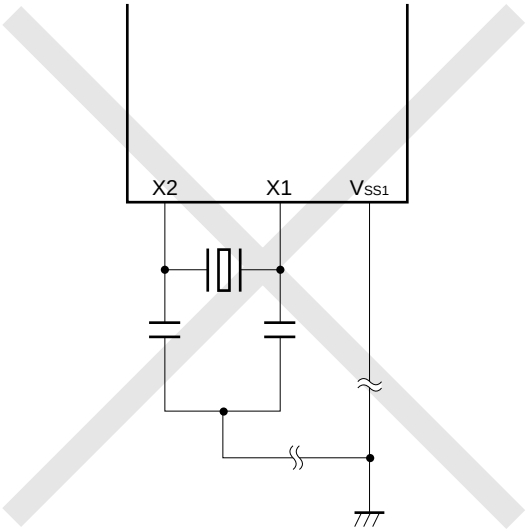
- **Minimize the wiring length.**
- **Do not allow wiring to intersect with other signal conductors. Do not allow wiring to come near changing high current.**
- **Set the potential of the grounding position of the oscillator capacitor to that of V_{SS1}. Do not ground to any ground pattern where high current is present.**
- **Do not fetch signals from the oscillator.**

Take special note of the fact that the subsystem clock oscillator is a circuit with low-level amplification so that current consumption is maintained at low levels.

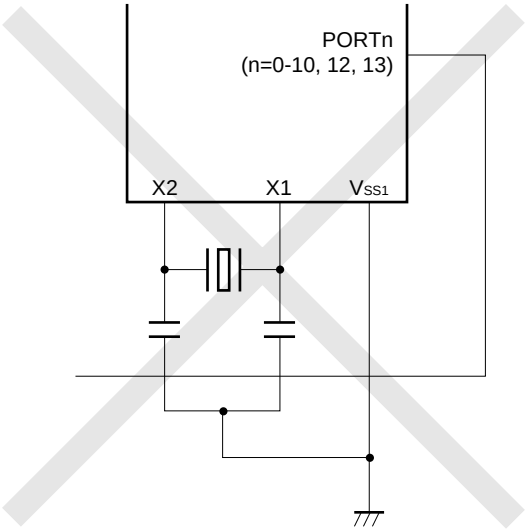
Figure 4-8 shows examples of oscillators that are connected incorrectly.

Figure 4-8. Examples of Oscillator Connected Incorrectly (1/2)

(a) Wiring of connection circuits is too long



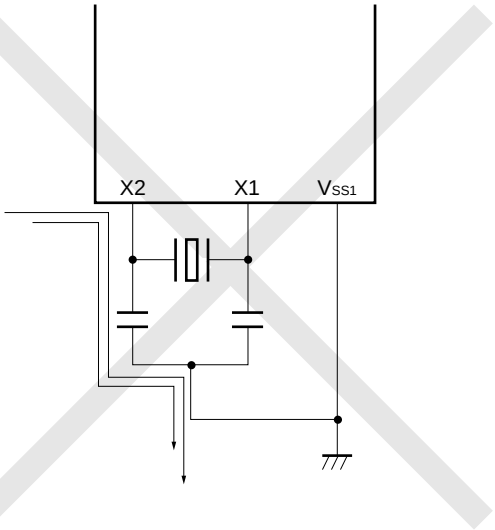
(b) Signal conductors intersect each other



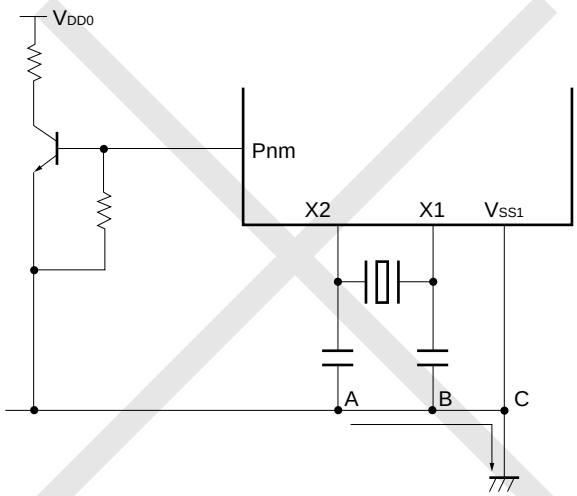
Remark When using a subsystem clock, replace X1 and X2 with XT1 and XT2, respectively. Further, insert resistors in series on the side of XT2.

Figure 4-8. Examples of Oscillator Connected Incorrectly (2/2)

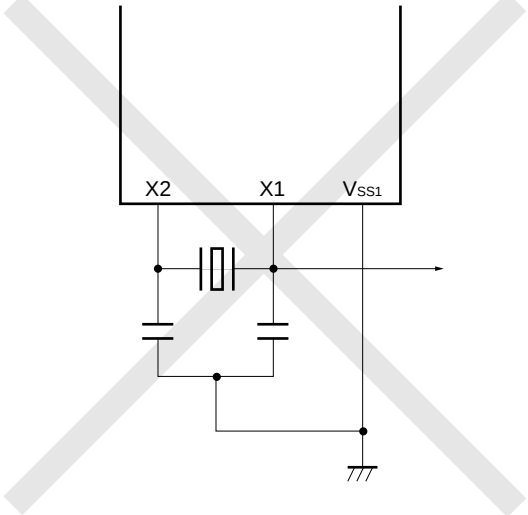
(c) Changing high current is too near a signal conductor



(d) Current flows through the groundin line of the oscillator (potential at points A, B, and C fluctuate)



(e) Signals are fetched



Remark When using a subsystem clock, replace X1 and X2 with XT1 and XT2, respectively. Also, insert resistors in series on the XT2 side.

Cautions 2. When XT2 and X1 are wired in parallel, the cross-talk noise of X1 may increase with XT2, resulting in malfunctioning. To prevent this from occurring, it is recommended not to wire XT1 and XT2 in parallel.

4.4.3 Frequency divider

The frequency divider divides the main system clock oscillator output (f_{XX}) and generates various clocks.

4.4.4 When no subsystem clocks are used

If it is not necessary to use subsystem clocks for low power consumption operations and clock operations, connect the XT1 and XT2 pins as follows.

XT1: Connect to V_{SS1}
XT2: Leave open

In this state, however, some current may leak via the internal feedback resistor of the subsystem clock oscillator when the main system clock stops. To minimize leakage current, set 1 bit 7 (SBK) of the standby control register (STBC). In this case also, connect the XT1 and XT2 pins as described above.

4.5 Clock Generator Operations

The clock generator generates the following types of clocks and controls the CPU operating mode including the standby mode.

- Main system clock (f_{XX})
- Subsystem clock (f_{XT})
- CPU clock (f_{CPU})
- Clock to peripheral hardware

The following clock generator functions and operations are determined with the standby control register (STBC) and the oscillation mode selection register (CC).

(a) Upon generation of the $\overline{\text{RESET}}$ signal, the lowest speed mode of the main system clock (160 ns when operated at 12.5 MHz) is selected (STBC = 04H, CC = 00H). Main system clock oscillation stops while low level is applied to the $\overline{\text{RESET}}$ pin.

(b) With the main system clock selected, one of the six CPU clock types (160 ns, 320 ns, 640 ns, 1280 ns, 2560 ns: 12.5 MHz) can be selected by setting the STBC and CC.

(c) With the main system clock selected, two standby modes, the STOP mode and the HALT mode, are available. To decrease current consumption in the STOP mode, the subsystem clock feedback resistor can be disconnected to stop the subsystem clock with bit 7 (SBK) of STBC, when the system does not use a subsystem clock.

(d) STBC can be used to select the subsystem clock and to operate the system with low current consumption (61 μs when operated at 32.768 kHz).

(e) With the subsystem clock selected, main system clock oscillation can be stopped with STBC. The HALT mode can be used. However, the STOP mode cannot be used. (Subsystem clock oscillation cannot be stopped.)

(f) The main system clock is divided and supplied to the peripheral hardware. The subsystem clock is supplied to the 16-bit timer/counter, the watch timer, and clock output functions only. Thus, the 16-bit timer/counter (when watch timer output is selected for count clock during operation with a subsystem clock), the watch function, and the clock output function can also be continued in the standby state. However, since all other peripheral hardware operate with the main system clock, the peripheral hardware (except external input clock operation) also stops if the main system clock is stopped.

4.5.1 Main system clock operations

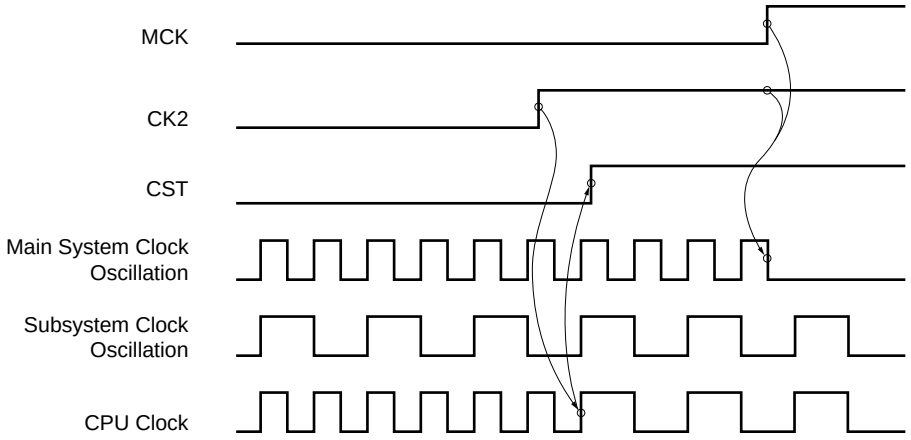
During operation with the main system clock (with bit 6 (CK2) of the standby control register (STBC) set to 0), the following operations are carried out.

(a) Because the operation guarantee instruction execution speed depends on the power supply voltage, the instruction execution time can be changed by setting bits 4 to 6 (CK0 to CK2) of the STBC.

(b) If bit 2 (MCK) of the STBC is set to 1 when operated with the main system clock, the main system clock oscillation does not stop. When bit 6 (CK2) of the STBC is set to 1 and the operation is switched to subsystem clock operation (CST = 1) after that, the main system clock oscillation stops (refer to **Figure 4-9**).

Figure 4-9. Main System Clock Stop Function (1/2)

(a) Operation when MCK is set after setting CK2 during main system clock operation



(b) Operation when MCK is set during main system clock operation

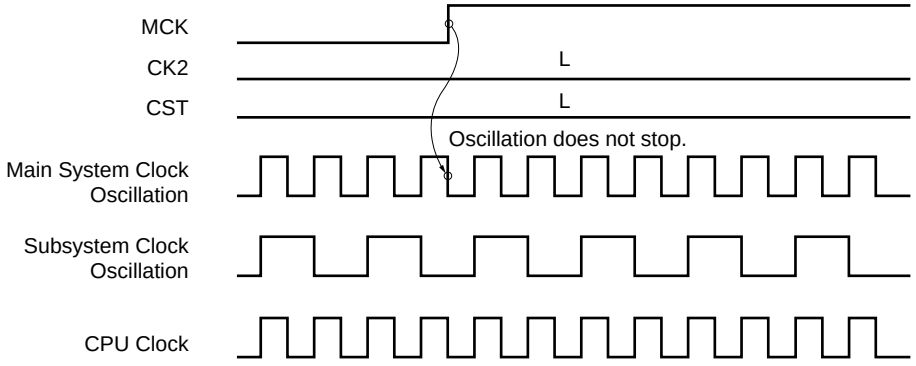
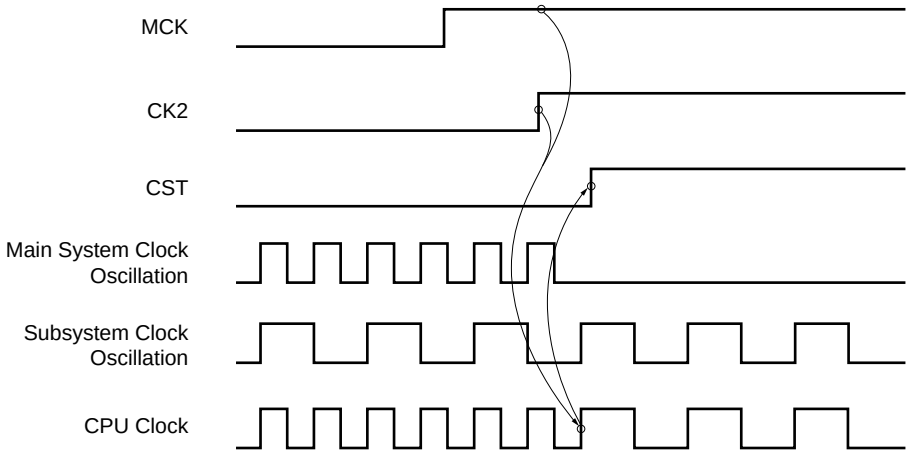


Figure 4-9. Main System Clock Stop Function (2/2)

(c) Operation when CK2 is set after setting MCK during main system clock operation



4.5.2 Subsystem clock operations

When operated with the subsystem clock (with bit 6 (CK2) of the standby control register (STBC) set to 1), the following operations are carried out.

(a) The instruction execution time remains constant (61 μs when operated at 32.768 kHz) irrespective of setting bits 4 and 5 (CK0 and CK1) of the STBC.

(b) Watchdog timer counting stops.

Caution Do not set the STOP mode while the subsystem clock is operating.

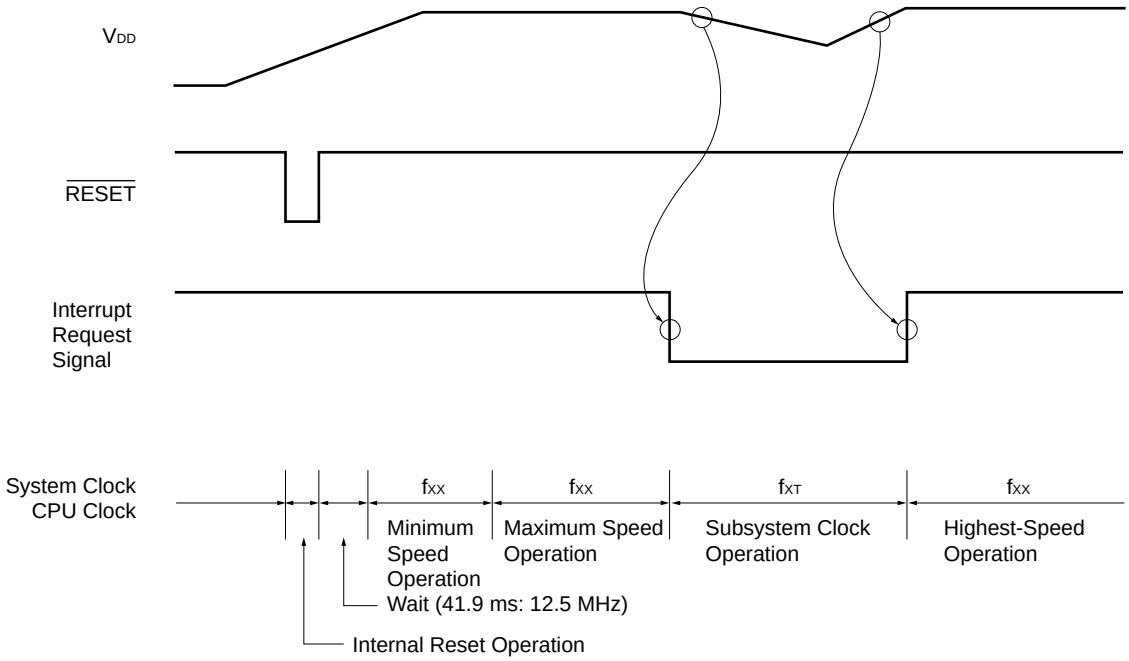
4.6 Changing System Clock and CPU Clock Settings

The system clock and CPU clock can be switched by means of bits 4 to 6 (CK0 to CK2) of the standby control register (STBC).

Whether the system is operating on the main system clock or the subsystem clock can be determined by the value of bit 0 (CST) of the clock status register (PCS).

This section describes the switching procedure between the system clock and the CPU clock.

Figure 4-10. System Clock and CPU Clock Switching



(1) The CPU is reset by setting the $\overline{\text{RESET}}$ signal to low level after power-on. After that, when reset is released by setting the $\overline{\text{RESET}}$ signal to high level, the main system clock starts oscillating. At this time, the oscillation stabilization time ($2^{19}/f_X$) is secured automatically.
After that, the CPU starts executing the instruction at the minimum speed of the main system clock (2560 ns when operated at 12.5 MHz).

(2) After the lapse of a sufficient time for the V_{DD} voltage to increase to enable operation at maximum speed, the STBC and CC are rewritten and maximum-speed operation is carried out.

(3) Upon detection of a decrease in the V_{DD} voltage due to an interrupt, the main system clock is switched to the subsystem clock (which must be in a stable oscillation state).

(4) Upon detection of V_{DD} voltage reset due to an interrupt, 0 is set to STBC bit 2 (MCK) and oscillation of the main system clock is started. After the lapse of time required for stabilization of oscillation, STBC is rewritten and maximum-speed operation is resumed.

Caution When a subsystem clock is being operated while the main system clock is stopped, if switching back to the main system clock, be sure to switch after securing the oscillation stabilization time by software.

CHAPTER 5 PORT FUNCTIONS

5.1 Port Functions

The ports shown in Figure 5-1, which enable a variety of controls, are provided. The function of each port is described in Table 5-1. On-chip pull-up registers can be specified for ports 0, 2 to 7, and 12 by software during input.

Figure 5-1. Port Configuration

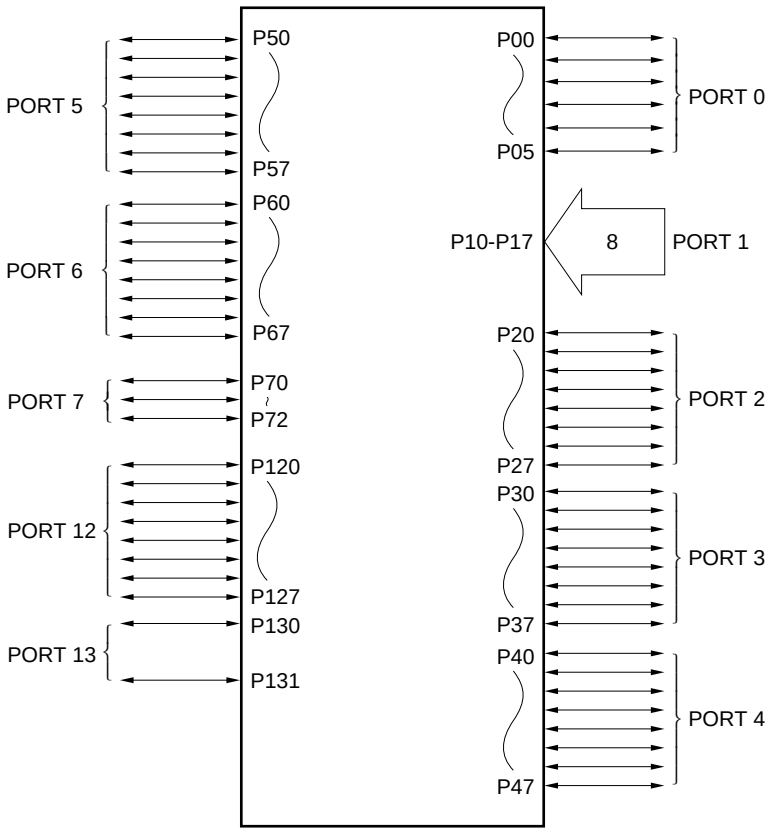


Table 5-1. Port Functions

Port	Pin Name	Function	Specification of Software Pull-Up Resistor
Port 0	P00 to P05	• Can be specified for input or output in 1-bit units	Specifiable in 1-bit units
Port 1	P10 to P17	• Input port	—
Port 2	P20 to P27	• Can be specified for input or output in 1-bit units	Specifiable in 1-bit units
Port 3	P30 to P37	• Can be specified for input or output in 1-bit units	Specifiable in 1-bit units
Port 4	P40 to P47	• Can be specified for input or output in 1-bit units • Can drive LED directly	Specifiable individually for each port
Port 5	P50 to P57	• Can be specified for input or output in 1-bit units • Can drive LED directly	Specifiable individually for each port
Port 6	P60 to P67	• Can be specified for input or output in 1-bit units	Specifiable individually for each port
Port 7	P70 to P72	• Can be specified for input or output in 1-bit units	Specifiable in 1-bit units
Port 12	P120 to P127	• Can be specified for input or output in 1-bit units	Specifiable in 1-bit units
Port 13	P130, P131	• Can be specified for input or output in 1-bit units	—

5.2 Port Configuration

Ports consist of the following hardware:

Table 5-2. Port Configuration

Item	Configuration
Control register	Port mode register (PMm: m = 0, 2 to 7, 12, 13) Pull-up resistor option register (PUO, PUm: m = 0, 2, 3, 7, 12)
Port	Total: 67 ports (8 inputs, 59 inputs/outputs)
Pull-up resistor	Total: 57 (software control)

5.2.1 Port 0

Port 0 is a 6-bit input/output port with output latch. The P00 to P05 pins can specify the input mode/output mode in 1-bit units with the port 0 mode register. A pull-up resistor can be connected to the P00 to P05 pins via the pull-up resistor option register 0, regardless of whether the input mode or output mode is specified.

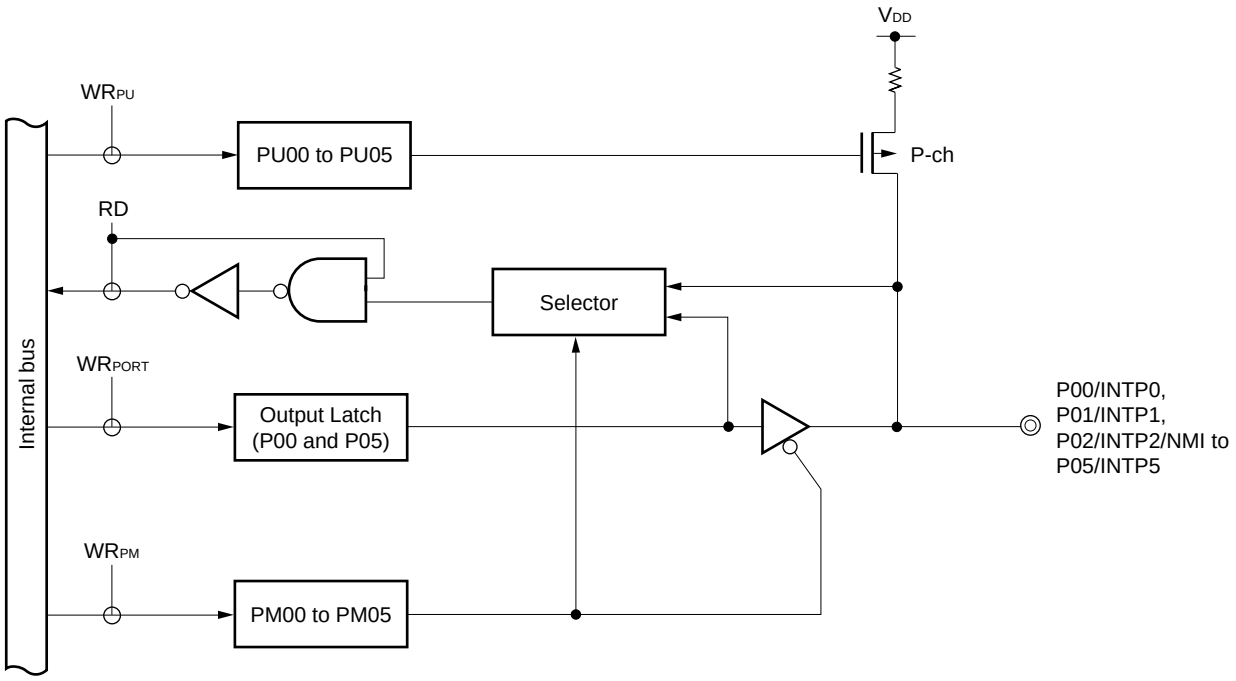
Port 0 also supports external interrupt request input as an alternate function.

$\overline{\text{RESET}}$ input sets port 0 to the input mode.

Figure 5-2 shows the block diagram of port 0.

Caution Because port 0 also serves for external interrupt request input, specifying the port function output mode and changing the output level sets the interrupt request flag. Thus, when the output mode is used, set the interrupt mask flag to 1.

Figure 5-2. Block Diagram of P00 to P05



PU : Pull-up resistor option register
PM : Port mode register
RD : Port 0 read signal
WR : Port 0 write signal

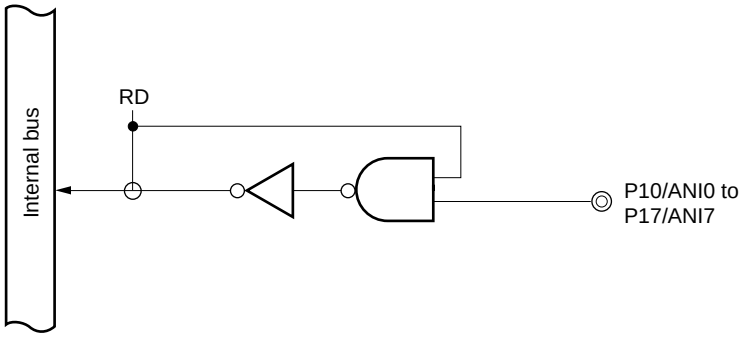
5.2.2 Port 1

This is an 8-bit input-only port with no on-chip pull-up resistor.

Port 1 supports A/D converter analog input as an alternate function.

Figure 5-3 shows a block diagram of port 1.

Figure 5-3. Block Diagram of P10 to P17



RD : Port 1 read signal

5.2.3 Port 2

Port 2 is an 8-bit input/output port with output latch. P20 to P27 pins can specify the input mode/output mode in 1-bit units with the port 2 mode register. A pull-up resistor can be connected to the P20 to P27 pins via the pull-up resistor option register 2, regardless of whether the input mode or output mode is specified.

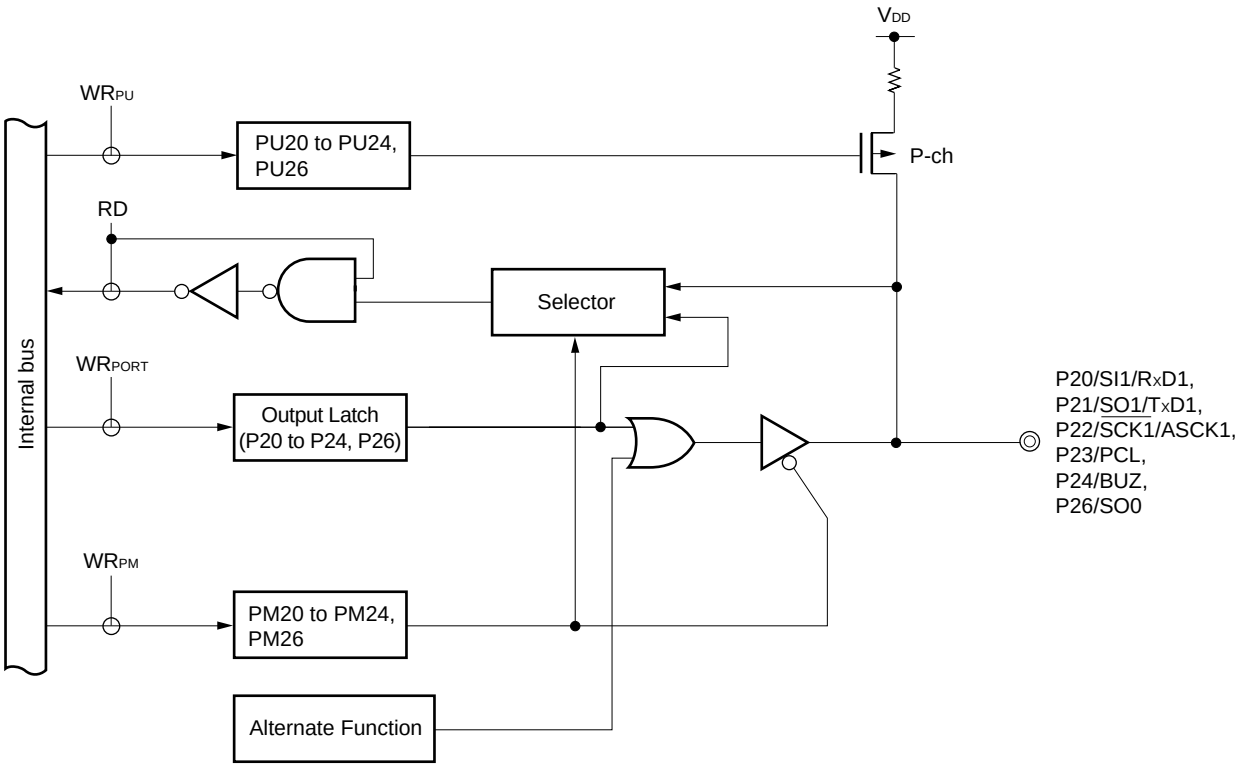
The P25 and P27 pins can be specified as N-ch open-drain with a port function control register (only μPD784225Y Subseries).

Port 2 supports serial interface data input/output, clock input/output, clock output, and buzzer output as alternate functions.

$\overline{\text{RESET}}$ input sets port 2 to the input mode.

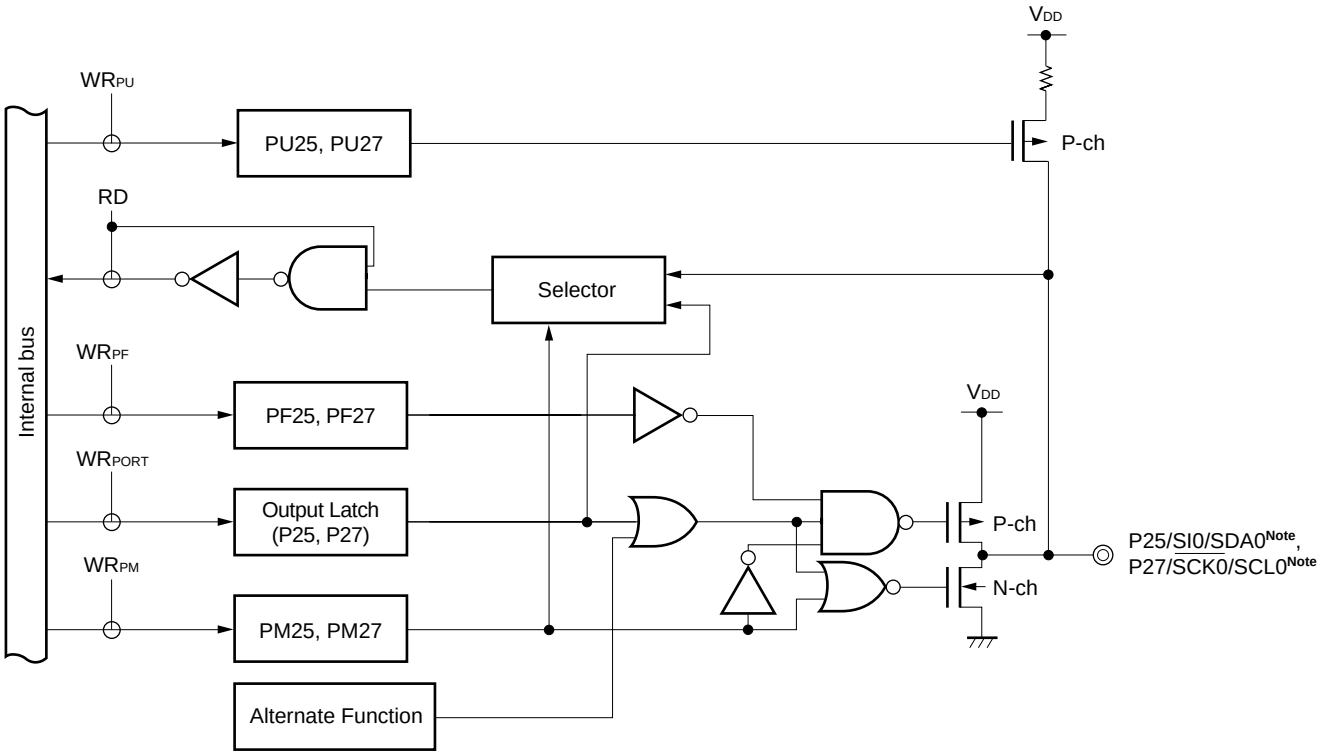
Figures 5-4 and 5-5 show a block diagram of port 2.

Figure 5-4. Block Diagram of P20 to P24 and P26



PU : Pull-up resistor option register
PM : Port mode register
RD : Port 2 read signal
WR : Port 2 write signal

Figure 5-5. Block Diagram of P25 and P27



Note The SDA0, SCL0 pins apply only to the μPD784225Y Subseries.

PU : Pull-up resistor option register
PF : Port function control register
PM : Port mode register
RD : Port 2 read signal
WR : Port 2 write signal

5.2.4 Port 3

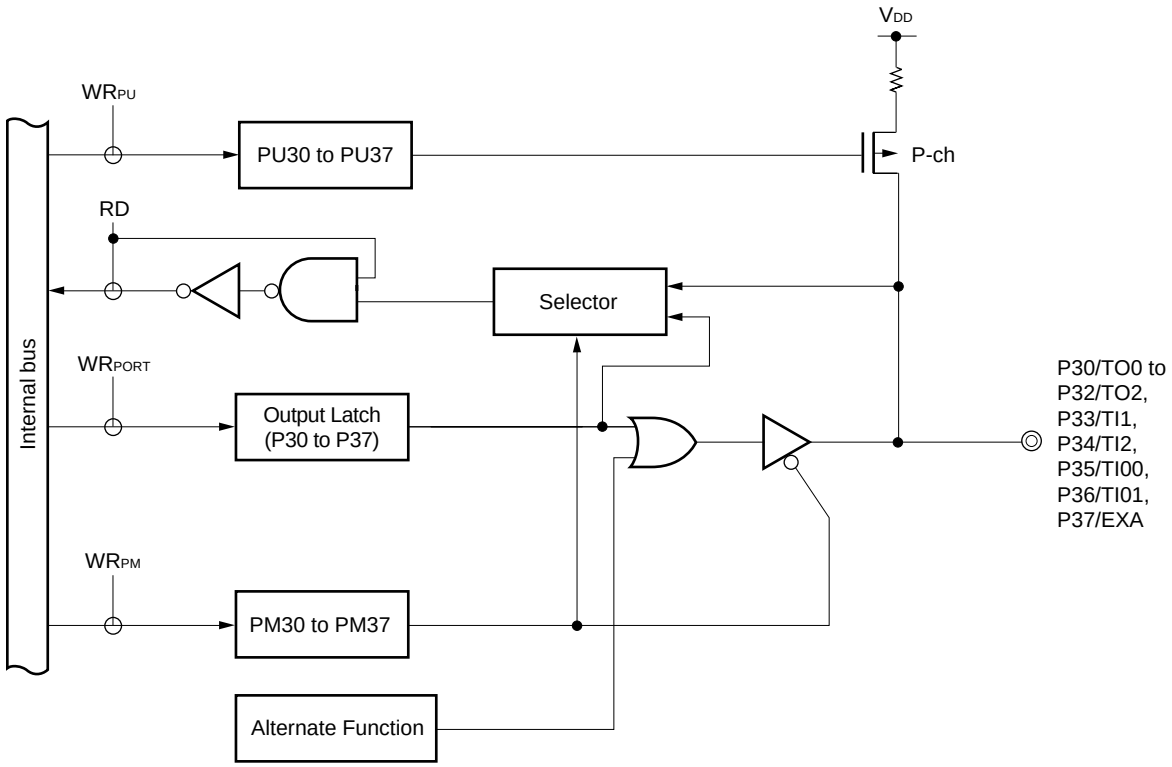
Port 3 is an 8-bit input/output port with output latch. The P30 to P37 pins can specify the input mode/output mode in 1-bit units with the port 3 mode register. A pull-up resistor can be connected to the P30 to P37 pins via the pull-up resistor option register 3, regardless of whether the input mode or output mode is specified.

Port 3 supports timer input/output as an alternate function.

$\overline{\text{RESET}}$ input sets port 3 to the input mode.

Figure 5-6 shows a block diagram of port 3.

Figure 5-6. Block Diagram of P30 to P37



PU : Pull-up resistor option register
PM : Port mode register
RD : Port 3 read signal
WR : Port 3 write signal

5.2.5 Port 4

Port 4 is an 8-bit input/output port with output latch. The P40 to P47 pins can specify the input mode/output mode in 1-bit units with the port 4 mode register. When the P40 to P47 pins are used as input ports, a pull-up resistor can be connected to them in 8-bit units with bit 4 (PUO4) of the pull-up resistor option register.

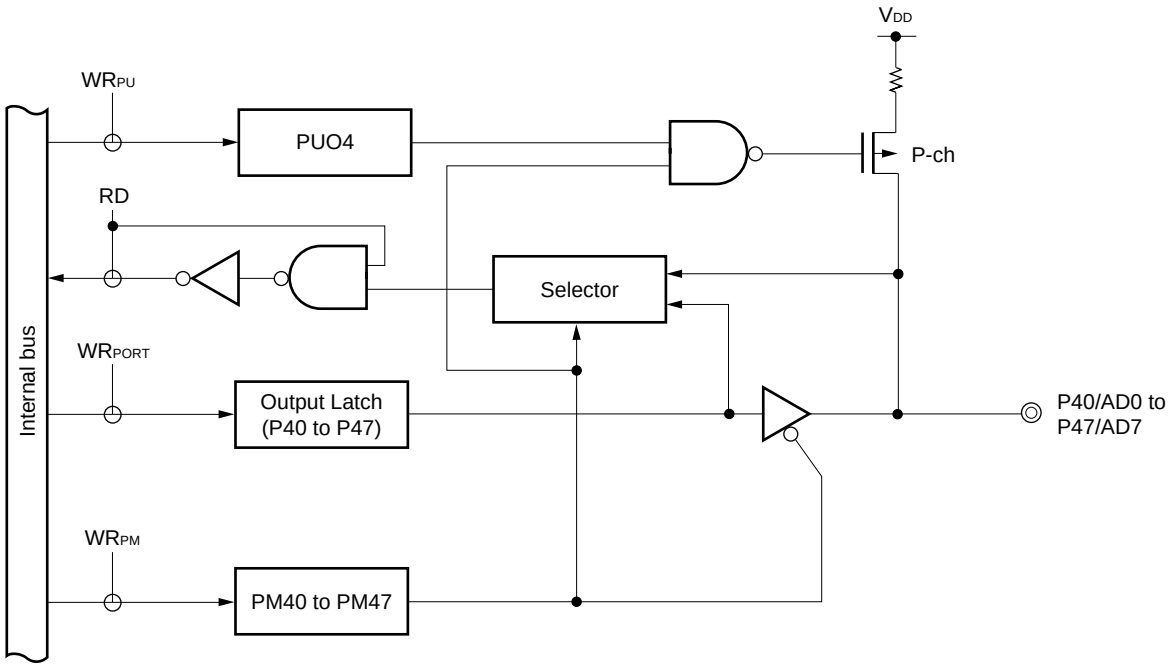
Port 4 can drive LED directly.

Port 4 supports the address/data bus function in the external memory expansion mode as an alternate function.

$\overline{\text{RESET}}$ input sets port 4 to the input mode.

Figure 5-7 shows a block diagram of port 4.

Figure 5-7. Block Diagram of P40 to P47



PUO : Pull-up resistor option register
PM : Port mode register
RD : Port 4 read signal
WR : Port 4 write signal

5.2.6 Port 5

Port 5 is an 8-bit input/output port with output latch. The P50 to P57 pins can specify the input mode/output mode in 1-bit units with the port 5 mode register. When the P50 to P57 pins are used as input ports, a pull-up resistor can be connected to them in 8-bit units with bit 5 (PUO5) of the pull-up resistor option register.

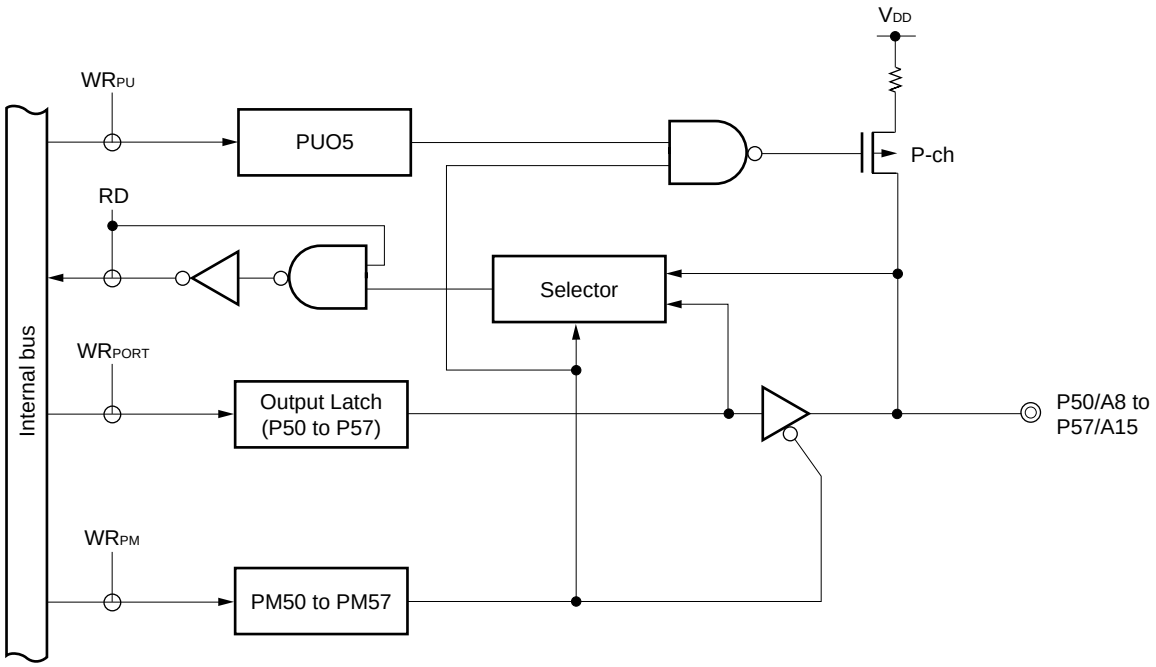
Port 5 can drive LEDs directly.

Port 5 supports the address bus function in the external memory expansion mode as an alternative function.

$\overline{\text{RESET}}$ input sets port 5 to the input mode.

Figure 5-8 shows a block diagram of port 5.

Figure 5-8. Block Diagram of P50 to P57



PUO : Pull-up resistor option register
PM : Port mode register
RD : Port 5 read signal
WR : Port 5 write signal

5.2.7 Port 6

Port 6 is an 8-bit input/output port with output latch. The P60 to P67 pins can specify the input mode/output mode in 1-bit units with the port 6 mode register.

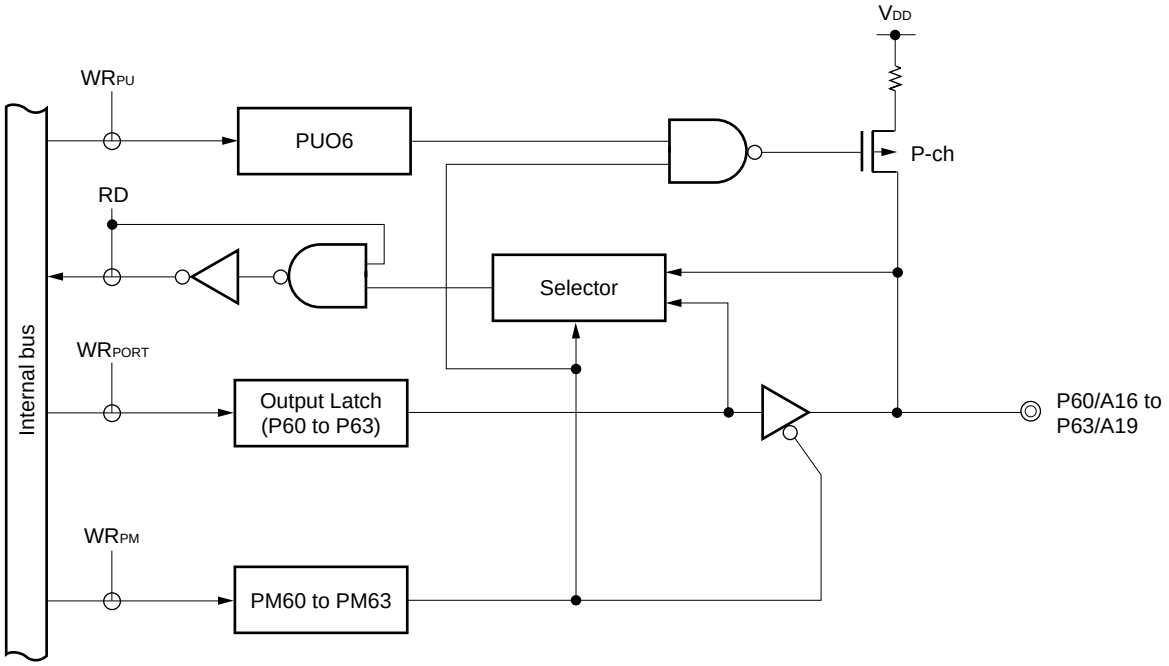
When pins P60 to P67 are used as input ports, a pull-up resistor can be connected to them in 8-bit units with bit 6 (PUO6) of the pull-up resistor option register.

Port 6 supports the address bus function and the control signal output function in external memory expansion mode as alternate functions.

$\overline{\text{RESET}}$ input sets port 6 to the input mode.

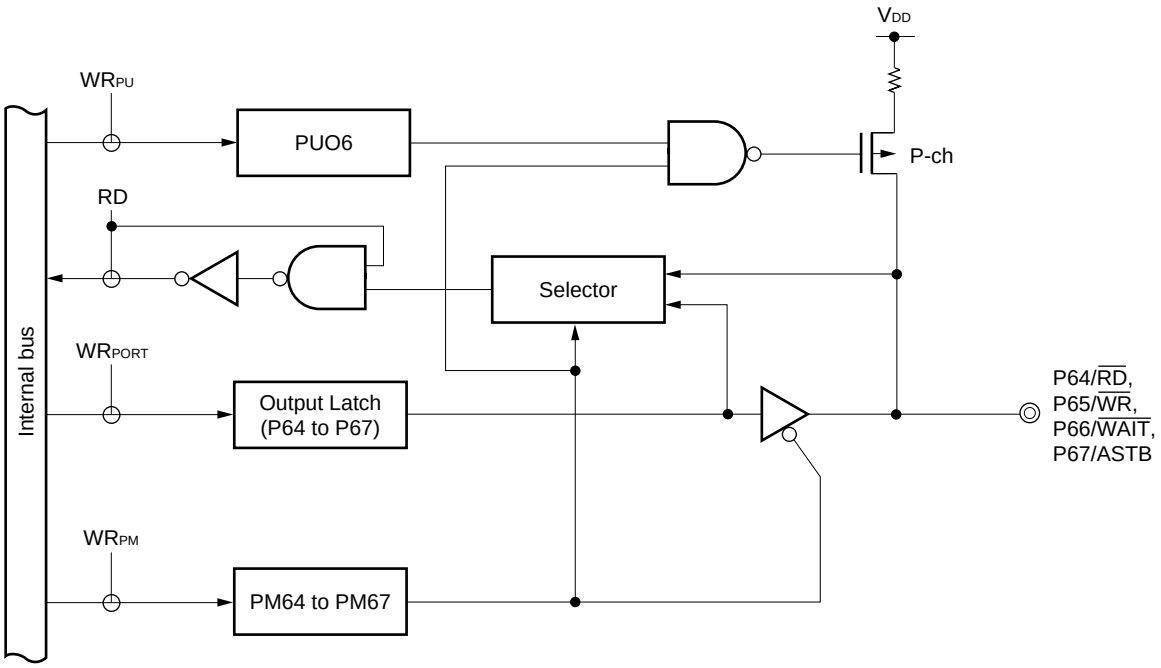
Figures 5-9 and 5-10 show block diagrams of port 6.

Figure 5-9. Block Diagram of P60 to P63



PUO : Pull-up resistor option register
PM : Port mode register
RD : Port 6 read signal
WR : Port 6 write signal

Figure 5-10. Block Diagram of P64 to P67



PUO : Pull-up resistor option register
PM : Port mode register
RD : Port 6 read signal
WR : Port 6 write signal

5.2.8 Port 7

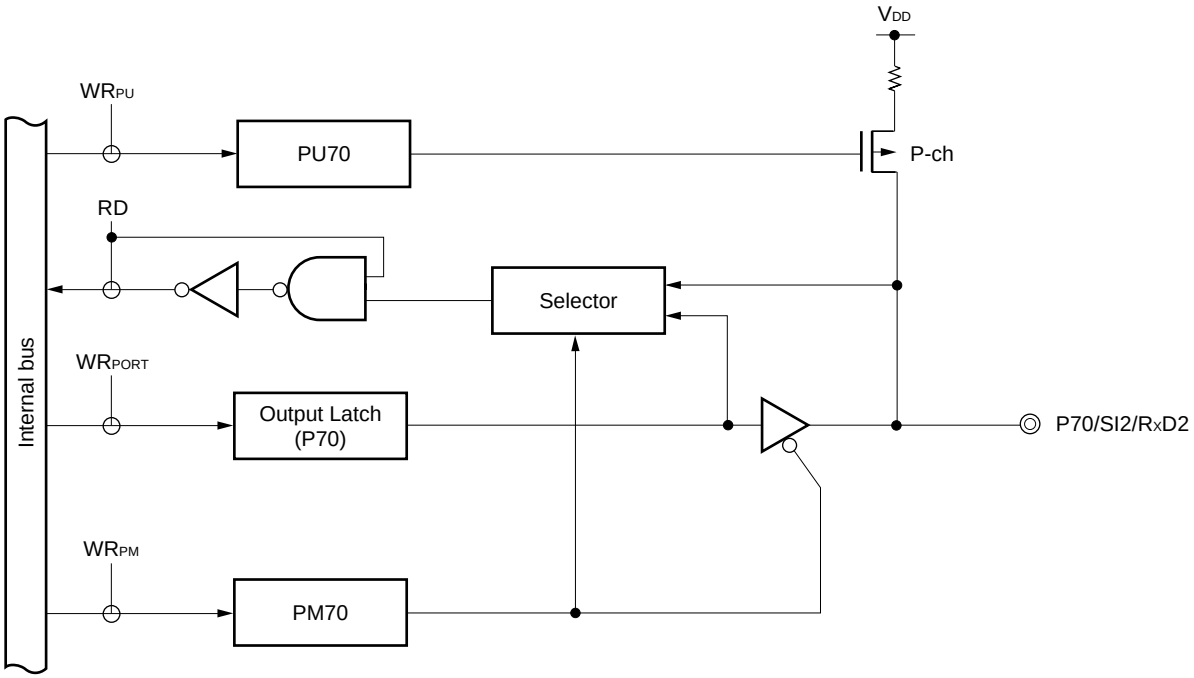
This is a 3-bit input/output port with output latch. Input mode/output mode can be specified in 1-bit units with the port 7 mode register. A pull-up resistor can be connected to the P70 to P72 pins via the pull-up resistor option register 7, regardless of whether the input mode or output mode is specified.

Port 7 supports serial interface data input/output and clock input/output as alternate functions.

$\overline{\text{RESET}}$ input sets port 7 to the input mode.

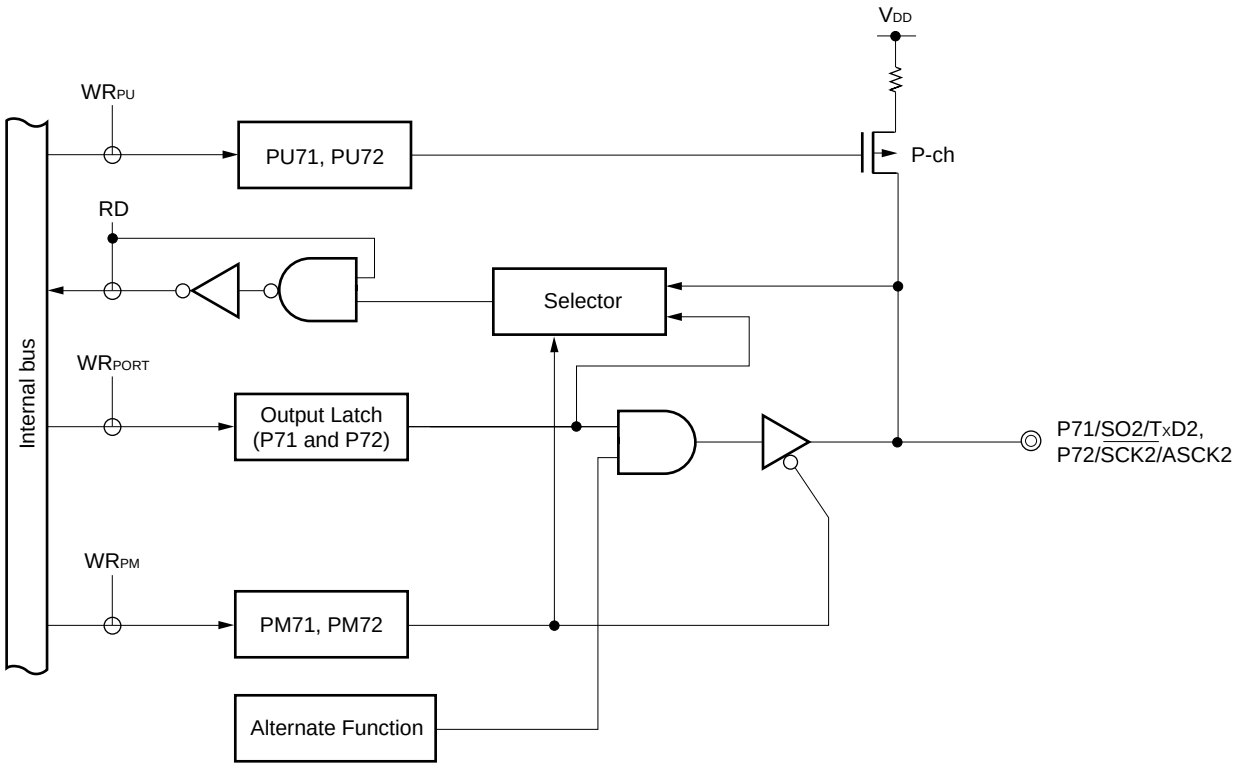
Figures 5-11 and 5-12 show block diagrams of port 7.

Figure 5-11. Block Diagram of P70



PU : Pull-up resistor option register
PM : Port mode register
RD : Port 7 read signal
WR : Port 7 write signal

Figure 5-12. Block Diagram of P71 and P72



PU : Pull-up resistor option register
PM : Port mode register
RD : Port 7 read signal
WR : Port 7 write signal

5.2.9 Port 12

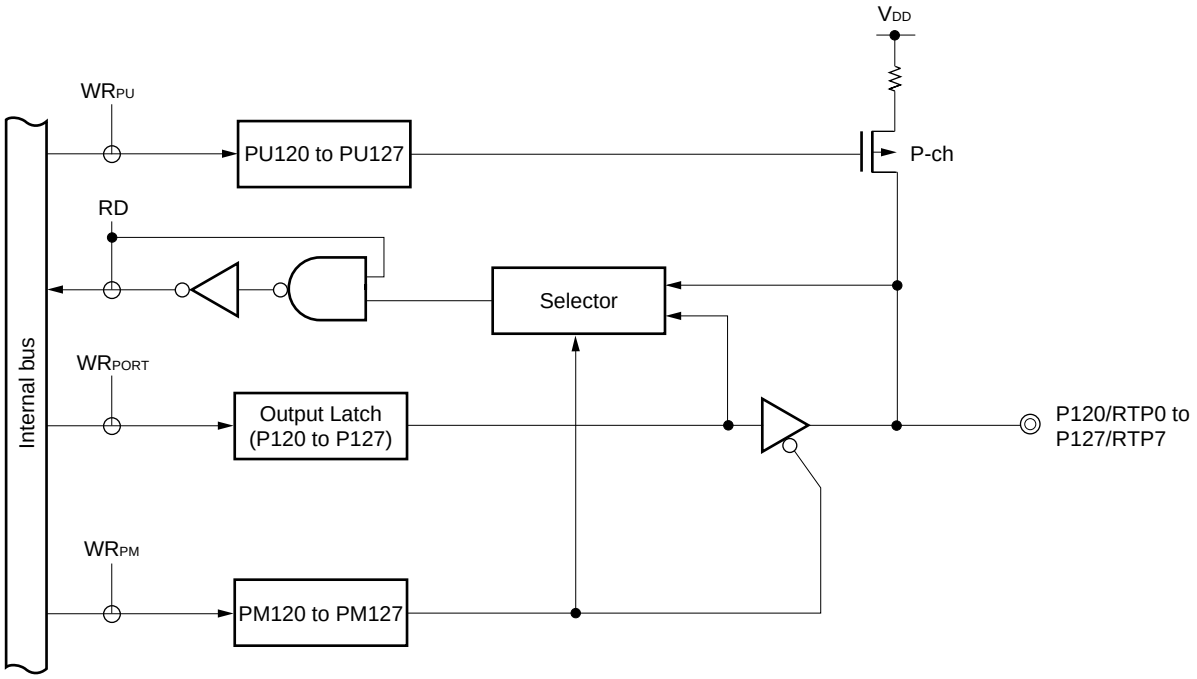
This is an 8-bit input/output port with output latch. Input mode/output mode can be specified in 1-bit units with the port 12 mode register. A pull-up resistor can be connected to the P120 to P127 pins via the pull-up resistor option register 12, regardless of whether the input mode or output mode is specified.

Port 12 supports the real-time output function as an alternative funcion.

$\overline{\text{RESET}}$ input sets port 12 to the input mode.

Figure 5-13 shows a block diagram of port 12.

Figure 5-13. Block Diagram of P120 to P127



PU : Pull-up resistor option register
PM : Port mode register
RD : Port 12 read signal
WR : Port 12 write signal

5.2.10 Port 13

This is a 2-bit input/output port with output latch. The input mode/output mode can be specified in 1-bit units with the port 13 mode register. Port 13 does not include a pull-up resistor.

Port 13 supports D/A converter analog output as an alternate function.

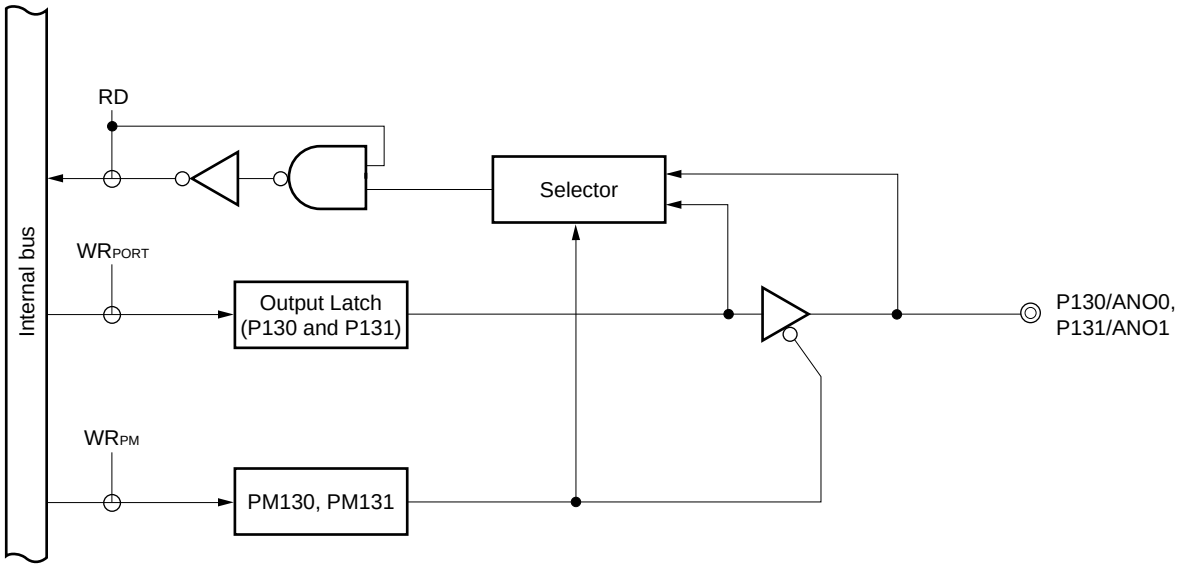
$\overline{\text{RESET}}$ input sets port 13 to the input mode.

Figure 5-14 shows a block diagram of port 13.

Caution When only either one of the D/A converter channels is used with $AV_{REF1} < V_{DD}$, the other pins that are not used as analog outputs must be set as follows:

- **Set the port mode register (PM13x) to 1 (input mode) and connect the pin to V_{SS0}.**
- **Set the port mode register (PM13x) to 0 (output mode) and the output latch to 0 to output low level from the pin.**

Figure 5-14. Block Diagram of P130 and P131



PM : Port mode register
RD : Port 13 read signal
WR : Port 13 write signal

5.3 Control Registers

The following three types of registers control the ports.

- Port mode registers (PM0, PM2 to PM7, PM12, PM13)
- Pull-up resistor option registers (PU0, PU2, PU3, PU7, PU12, PUO)
- Port function control register (PF2)[Note]

Note Applies only to the μPD784225Y Subseries.

(1) Port mode registers (PM0, PM2 to PM7, PM12, PM13)

These registers are used to set port input/output in 1-bit units.

PM0, PM2 to PM7, PM12, and PM13 are set with a 1-bit or 8-bit memory manipulation instruction, respectively.

$\overline{\text{RESET}}$ input sets port mode registers to FFH.

When port pins are used as alternate function pins, set the port mode registers and output latches according to Table 5-3.

Caution As port 0 has an alternative function as external interrupt request input, specifying the port function output mode and changing the output level sets the interrupt request flag. When the output mode is used, therefore, the interrupt mask flag should be set to 1 beforehand.

Table 5-3. Port Mode Register and Output Latch Settings when Using Alternate Functions

Pin Name	Alternate Function		PM××	P××	Pin Name	Alternate Function		PM××	P××
	Name	I/O				Name	I/O		
P00, P01	INTP0, INTP1	Input	1	×	P35, P36	TI00, TI01	Input	1	×
P02	INTP2/NMI	Input	1	×	P37	EXA	Output	0	0
P03 to P05	INTP3 to INTP5	Input	1	×	P40 to P47	AD0 to AD7	I/O	×Note2	
P10 to P17Note1	ANI0 to ANI7	Input	—		P50 to P57	A8 to A15	Output	×Note2	
P20	RxD1/SI1	Input	1	×	P60 to P63	A16 to A19	Output	×Note2	
P21	TxD1/SO1	Output	0	0	P64	$\overline{RD}$	Output	×Note2	
P22	ASK1	Input	1	×	P65	$\overline{WR}$	Output	×Note2	
	$\overline{SCK1}$	Input	1	×	P66	$\overline{WAIT}$	Input	×Note2	
		Output	0	0	P67	ASTB	Output	×Note2	
P23	PCL	Output	0	0	P70	RxD2/SI2	Input	1	×
P24	BUZ	Output	0	0	P71	TxD2/SO2	Output	0	0
P25	SI0	Input	1	×	P72	ASK2	Input	1	×
	SDA0Note3	I/O	0	0		$\overline{SCK2}$	Input	1	×
P26	SO0	Output	0	0			Output	0	0
P27	$\overline{SCK0}$	Input	1	×	P120 to P127	RTP0 to RTP7	Output	0	desired value
		Output	0	0	P130, P131Note1	ANO0, ANO1	Output	1	×
	SCL0Note3	I/O	0	0					
P30 to P32	TO0 to TO2	Output	0	0					
P33, P34	TI1, TI2	Input	1	×					

Notes 1. If the read command is executed for these ports when they are being used as alternate function pins, the data read will be undefined.

2. The function is set with the memory expansion mode register (MM) when the P40 to P47, P50 to P57 and P60 to P67 pins are used as alternate function pins.

3. The SDA0 and SCL0 pins are only available on the μPD784225Y Subseries.

Cautions 1. When not using external wait in the external memory extension mode, the P66 pin can be used as an I/O port.

2. Specify the SCL0/P27 and SDA0/P25 pins in the N-ch open-drain by setting the port function control register (PF2) when the I^2C bus mode is to be used.

Remark × : don't care (setting is not required)

— : Port mode register and output latch do not exist

PM×× : Port mode register

P×× : Port output latch

Figure 5-15. Port Mode Register Format

Address: 0FF20H, 0FF22H to 0FF27H, 0FF2CH, 0FF2DH After Reset: FFH R/W

Symbol	7	6	5	4	3	2	1	0
PM0	1	1	PM05	PM04	PM03	PM02	PM01	PM00
PM2	PM27	PM26	PM25	PM24	PM23	PM22	PM21	PM20
PM3	PM37	PM36	PM35	PM34	PM33	PM32	PM31	PM30
PM4	PM47	PM46	PM45	PM44	PM43	PM42	PM41	PM40
PM5	PM57	PM56	PM55	PM54	PM53	PM52	PM51	PM50
PM6	PM67	PM66	PM65	PM64	PM63	PM62	PM61	PM60
PM7	1	1	1	1	1	PM72	PM71	PM70
PM12	PM127	PM126	PM125	PM124	PM123	PM122	PM121	PM120
PM13	1	1	1	1	1	1	PM131	PM130

PMxn	Pxn Pin I/O Mode Specification (x = 0: n = 0 to 6 x = 2 to 6, 12: n = 0 to 7 x = 7: n = 0 to 2 x = 13: n = 0, 1)
0	Output mode (output buffer ON)
1	Input mode (output buffer OFF)

(2) Pull-up resistor option registers (PU0, PU2, PU3, PU7, PU12, PUO)

These registers are used to set whether to use an internal pull-up resistor at each port or not in 1-bit or 8-bit units. PUn (n = 0, 2, 3, 7, 12) can specify the pull-up resistor connection of each port pin. PUO can specify the pull-up resistor connection of ports 4, 5, and 6. Pull-up resistors are connected irrespective of whether an alternate function is used.

These registers are set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets these registers to 00H.

Cautions 1. Ports 1, and 13 do not incorporate a pull-up resistor.

2. Ports 4, 5, and 6 can connect a pull-up resistor during external memory expansion mode.

Figure 5-16. Pull-Up Resistor Option Register Format

Address: 0FF30H, 0FF32H, 0FF33H, 0FF37H, 0FF3CH After Reset: 00H R/W

Symbol	⑦	⑥	⑤	④	③	②	①	⓪
PU0	0	0	PU05	PU04	PU03	PU02	PU01	PU00
PU2	PU27	PU26	PU25	PU24	PU23	PU22	PU21	PU20
PU3	PU37	PU36	PU35	PU34	PU33	PU32	PU31	PU30
PU7	0	0	0	0	0	PU72	PU71	PU70
PU12	PU127	PU126	PU125	PU124	PU123	PU122	PU121	PU120

PUxn	Pxn Pin Pull-Up Resistor Specification (x = 0: n = 0 to 6 x = 2, 3, 12: n = 0 to 7 x = 7: n = 0 to 2)
0	No pull-up resistor connection
1	Pull-up resistor connection

Address: 0FF4EH After reset: 00H R/W

Symbol	7	⑥	⑤	④	3	2	1	0
PUO	0	PUO6	PUO5	PUO4	0	0	0	0

PUOn	Port n Pull-Up Resistor Specification (n = 4 to 6)
0	No pull-up resistor connection
1	Pull-up resistor connection

(3) Port function control register (PF2)

This register specifies N-ch open drain for pins P25 and P27.

PF2 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PF2 to 00H.

Caution Only the μPD784225Y Subseries incorporates PF2. When using the I^2C bus mode (serial interface), make sure to specify N-ch open drain for the P25 and P27 pins.

Figure 5-17. Port Function Control Register (PF2) Format

Address: 0FF42H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
PF2	PF27	0	PF25	0	0	0	0	0

PF2n	P2n Pin N-ch Open Drain Specification (n = 5, 7)
0	Don't set N-ch open-drain
1	Set N-ch open-drain

5.4 Operations

Port operations differ depending on whether the input or output mode is set, as shown below.

5.4.1 Writing to input/output port

(1) Output mode

A value is written to the output latch by a transfer instruction, and the output latch contents are output from the pin.

Once data is written to the output latch, it is retained until data is written to the output latch again.

(2) Input mode

A value is written to the output latch by a transfer instruction, but since the output buffer is OFF, the pin status does not change.

Once data is written to the output latch, it is retained until data is written to the output latch again.

Caution In the case of 1-bit memory manipulation instructions, although a single bit is manipulated, the port is accessed in 8-bit units. Therefore, on a port with a mixture of input and output pins, the output latch contents for pins specified as input are undefined except for the manipulated bit.

5.4.2 Reading from input/output port

(1) Output mode

The output latch contents are read by a transfer instruction. The output latch contents do not change.

(2) Input mode

The pin status is read by a transfer instruction. The output latch contents do not change.

5.4.3 Operations on input/output port

(1) Output mode

An operation is performed on the output latch contents, and the result is written to the output latch. The output latch contents are output from the pins.

Once data is written to the output latch, it is retained until data is written to the output latch again.

(2) Input mode

The output latch contents are undefined, but since the output buffer is OFF, the pin status does not change.

Caution In the case of 1-bit memory manipulation instructions, although a single bit is manipulated, the port is accessed in 8-bit units. Therefore, on a port with a mixture of input and output pins, the output latch contents for pins specified as input are undefined, except for the manipulated bit.

CHAPTER 6 REAL-TIME OUTPUT FUNCTIONS

6.1 Functions

The real-time output function transfers preset data in the real-time output buffer register to the output latch by hardware synchronized to the generation of a timer interrupt or an external interrupt and outputs it off the chip. Also, the pins for output off the chip are called the real-time output port.

Since jitter-free signals can be output by using the real-time output port, the operation is optimized for the control of stepping motors, for example.

The port mode or real-time output mode is bit selectable.

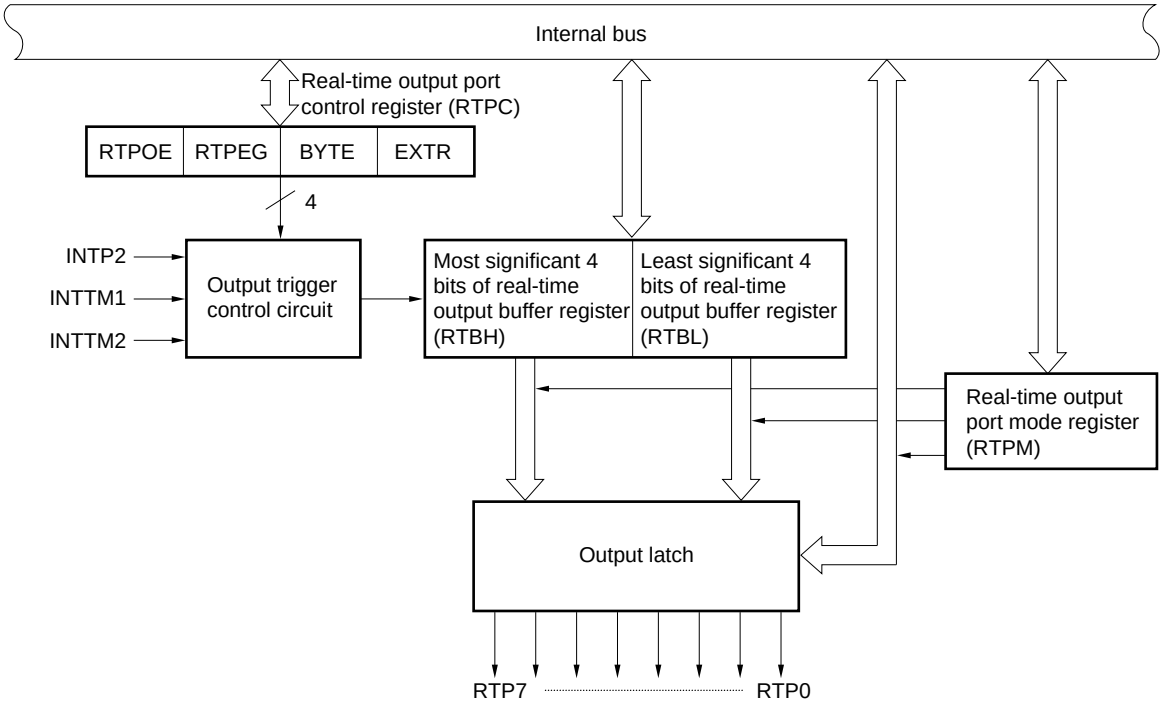
6.2 Structure

The real-time output port consists of the following hardware.

Table 6-1. Real-Time Output Configuration

Item	Structure
Registers	Real-time output buffer registers (RTBL, RTBH)
Control registers	Real-time output port mode register (RTPM) Real-time output port control register (RTPC)

Figure 6-1. Block Diagram of Real-Time Output Port



- **Real-time output buffer registers (RTBL, RTBH)**

  These 4-bit registers save the output data beforehand. RTBL and RTBH are mapped to independent addresses in the special function register (SFR) as shown in Figure 6-2.

  When the 4 bits × 2 channels operating mode is specified, RTBL and RTBH can be independently set with data. In addition, if the addresses of both RTBL and RTBH are specified, the data in both registers can be read in a batch.

  When the 8 bits × 1 channel operating mode is specified, writing 8-bit data to either RTBL or RTBH can set data in either register. In addition, if the addresses of either RTBL and RTBH are specified, the data in both can be read in a batch.

  Table 6-2 lists the operations for manipulating RTBL and RTBH.

Figure 6-2. Real-Time Output Buffer Register Configuration

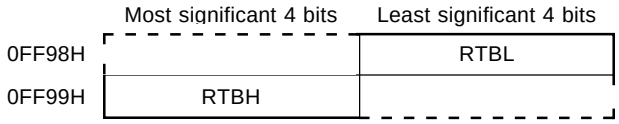


Table 6-2. Operation for Manipulating Real-Time Output Buffer Registers

Operating Mode	Manipulated Register	Reading Note 1		Writing Note 2	
		Most significant 4 bits	Least significant 4 bits	Most significant 4 bits	Least significant 4 bits
4 bits × 2 channels	RTBL	RTBH	RTBL	Invalid	RTBL
	RTBH	RTBH	RTBL	RTBH	Invalid
8 bits × 1 channels	RTBL	RTBH	RTBL	RTBH	RTBL
	RTBH	RTBH	RTBL	RTBH	RTBL

Notes 1. Only the bits specified in the real-time output port mode can be read. When the bits set in the bits set in the port mode are read, zeros are read.

2. After setting the real-time output port, set the output data in RTBL and RTBH until the real-time output trigger is generated.

6.3 Control Registers

The real-time output port is controlled by the following two registers.

- Real-time output port mode register (RTPM)
- Real-time output port control register (RTPC)

(1) Real-time output port mode register (RTPM)

This register sets the real-time output port mode and port mode selections in 1-bit units.

RTPM is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets RTPM to 00H.

Figure 6-3. Real-Time Output Port Mode Register (RTPM) Format

Address: 0FF9AH After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
RTPM	RTPM7	RTPM6	RTPM5	RTPM4	RTPM3	RTPM2	RTPM1	RTPM0

RTPMm	Real-time Output Port Selection (m = 0 to 7)
0	Port mode
1	Real-time output mode

Cautions 1. When used as a real-time output port, set the port for real-time output in the output mode.

2. The port specified as a real-time output port cannot set data in the output latch. Therefore, when the initial values are set, set data in the output latch before setting the rea-time output port mode.

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

This register sets the operating mode and output trigger of the real-time output port.

Table 6-3 shows the relationships between the operating modes and output triggers of the real-time output port.

RTPC is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets RTPC to 00H.

Figure 6-4. Real-Time Output Port Control Register (RTPC) Format

Address: 0FF9BH After Reset: 00H R/W

Symbol	⑦	⑥	⑤	④	3	2	1	0
RTPC	RTPOE	RTPEG	BYTE	EXTR	0	0	0	0

RTPOE	Real-time Output Port Operation Control
0	Operation disabled **Note**
1	Operation enabled

RTPEG	INTP2 Valid Edge Setting
0	Falling edge
1	Rising edge

BYTE	Real-time Output Port Operation Mode
0	4 bits × 2 channels
1	8 bits × 1 channels

EXTR	Real-time Output Control by INTP2
0	Main system clock operation
1	Subsystem clock operation

Note When real-time output operation is disabled (RTPOE = 0), RTP0 to RTP7 output 0.

Table 6-3. Operating Modes and Output Triggers of Real-Time Output Port

BYTE	EXTR	Operating Mode	RTBH → Port Output	RTBL → Port Output
0	0	4 bits × 2 channels	INTTM2	INTTM1
0	1		INTTM1	INTP2
1	0	8 bits × 1 channel	INTTM1	
1	1		INTP2	

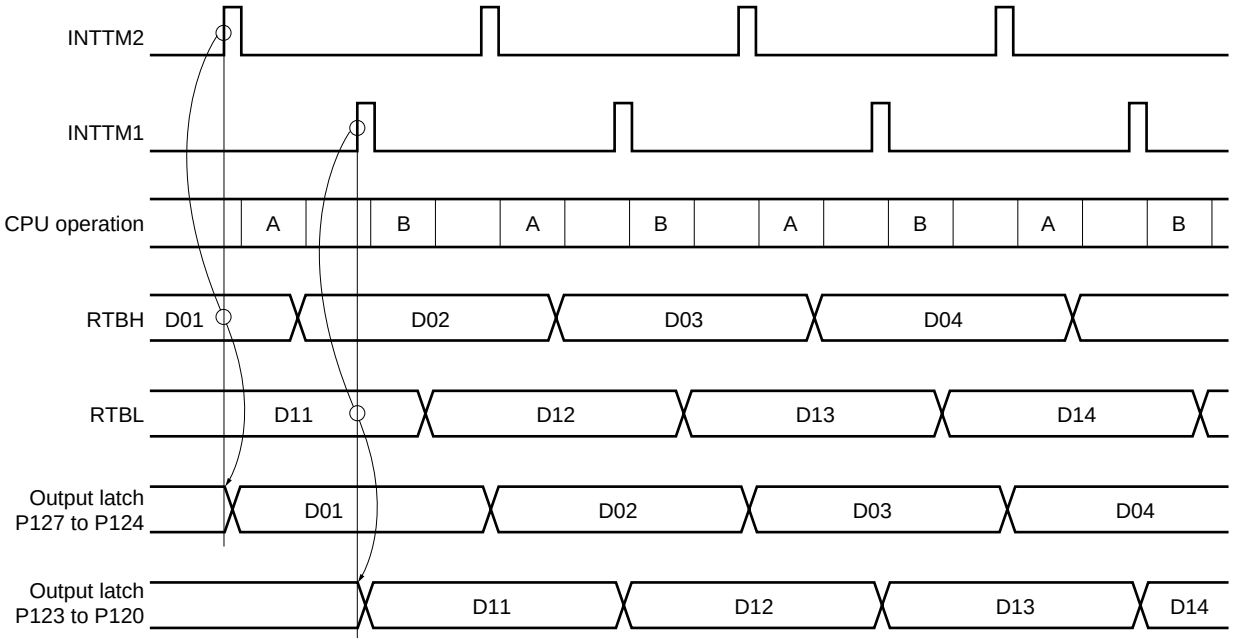
6.4 Operation

When real-time output is enabled by bit 7 (RTPOE) = 1 in the real-time output port control register, data in the real-time output buffer register (RTBH, RTBL) are transferred to the output latch synchronized to the generation of the selected transfer trigger (set by EXTR and BYTE Note). Based on the setting of the real-time output port mode register (RTPM), only the transferred data for the bits specified in the real-time output port are output from bits RTP0 to RTP7. A port set in the port mode by RTPM can be used as a general-purpose I/O port.

When the real-time output operation is disabled by RTPOE = 0, RTP0 to RTP7 output zero regardless of the RTPM setting.

Note EXTR: Bit 4 of the real-time output port control register (RTPC)
BYTE: Bit 5 of the real-time output port control register (RTPC)

Figure 6-5. Example of the Operation Timing of Real-Time Output Port (EXTR = 0, BYTE = 0)



A: Software processing by INTTM2 (RTBH write)
B: Software processing by INTTM1 (RTBL write)

6.5 Using this Function

(1) Disabling the real-time output operation
Set bit 7 (RTPOE) = 0 in the real-time output port control register (RTPC).

(2) Initial settings
- Set the initial value in the output latch.
- Set the real-time output port mode or port mode for each bit.
  Set the real-time output port mode register (RTPM).
- Select the trigger and the valid edge.
  Set bits 4, 5, and 6 (EXTR, BYTE, RTPEDG) of RTPC.
- Set the same initial value as the output latch in the real-time output buffer registers (RTBH, RTBL).

(3) Enable real-time output operation.
RTPOE = 1

(4) Set the next output in RTBL and RTBL until the selected transfer trigger is generated.

(5) The next real-time output values are sequentially set in the RTBH and RTBL by the interrupt processing for the selected trigger.

6.6 Cautions

(1) For the initial setting, set bit 7 (RTPOE) in the real-time output port control register (RTPC) to 0 to disable the real-time output operation.

(2) When the real-time output operation is disabled (RTPOE = 0) once, always set the same initial value as in the output latch in the real-time output buffer registers (RTBH, RTBL) before enabling real-time output (RTPOE = 0 $\rightarrow$ 1).

[MEMO]

CHAPTER 7 TIMER/COUNTER OVERVIEW

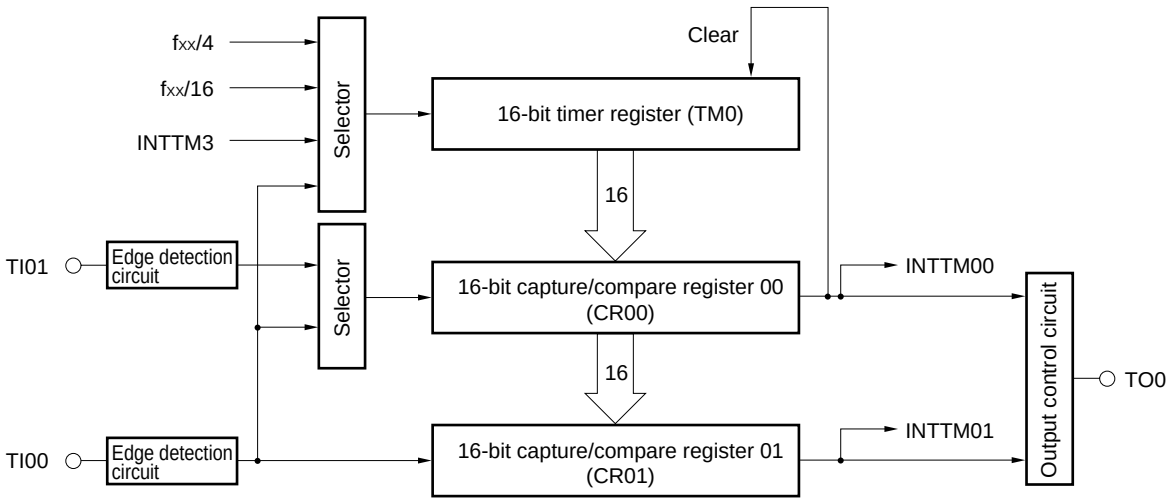
There are one on-chip 16-bit timer/counter and four on-chip 8-bit timer/counters.
Since a total of six interrupt requests is supported, these timer/counters can function as six units of timer/counters.

Table 7-1. Timer/Counter Operation

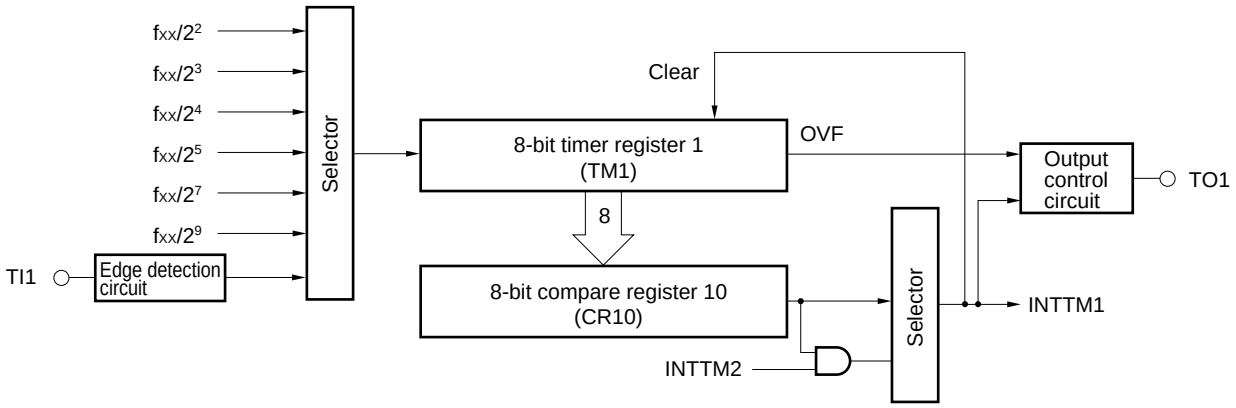
Item \ Name		16-bit Timer/ counter	8-bit Timer/ counter 1	8-bit Timer/ counter 2	8-bit Timer/ counter 5	8-bit Timer/ counter 6
Count width	8 bits	—	○	○	○	○
	16 bits	○	○		○	
Operating mode	Interval timer	1ch	1ch	1ch	1ch	1ch
	External event counter	○	○	○	—	—
Function	Timer output	1ch	1ch	1ch	—	—
	PPG output	○	—	—	—	—
	PWM output	○	○	○	—	—
	Square wave output	○	○	○	—	—
	One-shot pulse output	○	—	—	—	—
	Pulse width measurement	2 inputs	—	—	—	—
	No. of interrupt requests	2	1	1	1	1

Figure 7-1. Timer/Counter Block Diagram (1/2)

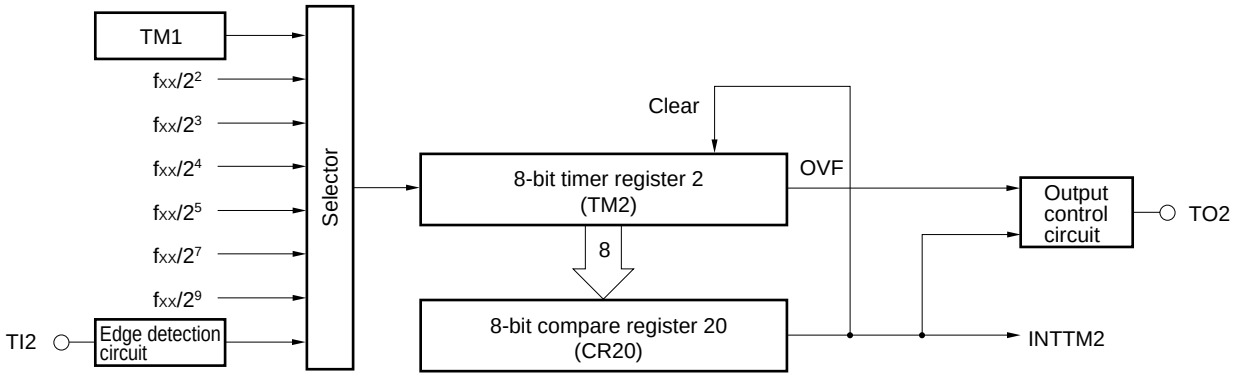
16-bit Timer/Counter



8-bit Timer/Counter 1



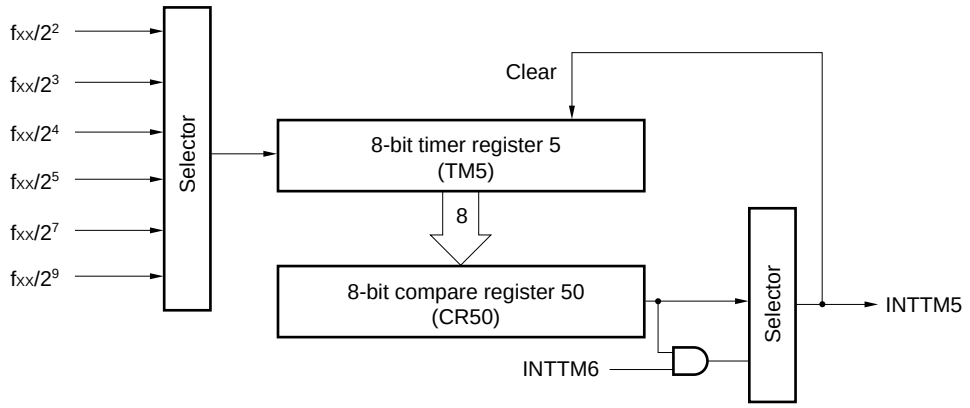
8-bit Timer/Counter 2



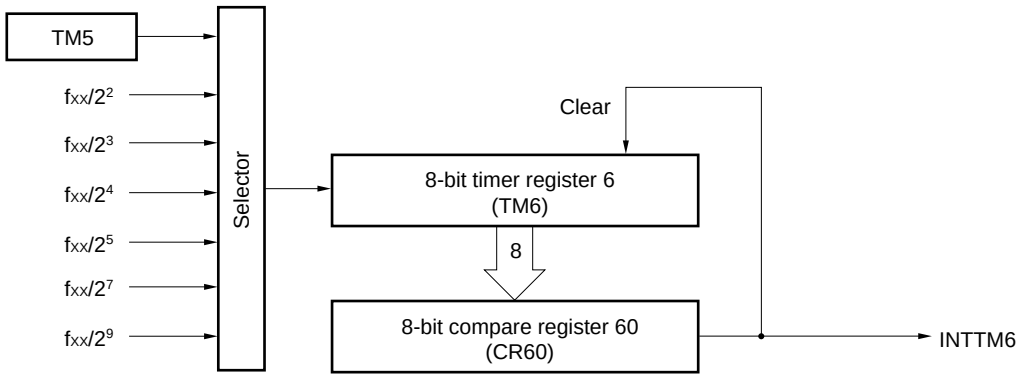
Remark OVF: Overflow flag

Figure 7-1. Timer/Counter Block Diagram (2/2)

8-bit Timer/Counters 5



8-bit Timer/Counters 6



[MEMO]

CHAPTER 8 16-BIT TIMER/COUNTER

8.1 Function

16-bit timer/counter (TM0) has the following functions:

- Interval timer
- PPG output
- Pulse width measurement
- External event counter
- Square wave output
- One-shot pulse output

(1) Interval timer

When 16-bit timer/counter is used as an interval timer, it generates an interrupt request at predetermined time intervals.

(2) PPG output

16-bit timer/counter can output a square wave whose frequency and output pulse width can be freely set.

(3) Pulse width measurement

16-bit timer/counter can be used to measure the pulse width of a signal input from an external source.

(4) External event counter

16-bit timer/counter can be used to measure the number of pulses of a signal input from an external source.

(5) Square wave output

16-bit timer/counter can output a square wave any frequency.

(6) One-shot pulse output

16-bit timer/counter can output a one-shot pulse with any output pulse width.

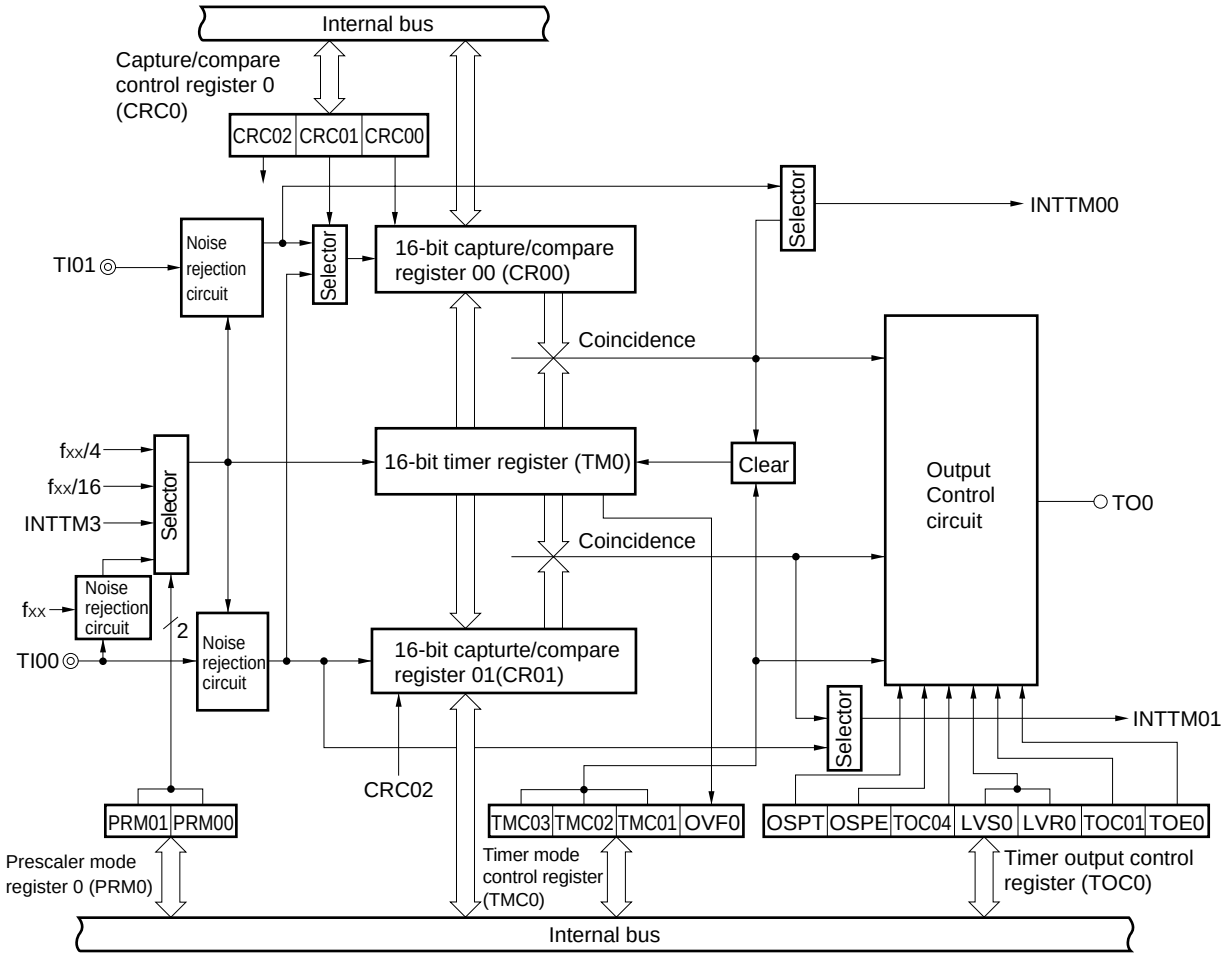
8.2 Configuration

16-bit timer/counter (TM0) consists of the following hardware:

Table 8-1. Configuration of 16-Bit Timer/Counter (TM0)

Item	Configuration
Timer register	16 bits × 1 (TM0)
Register	Capture/compare register: 16 bits × 2 (CR00, CR01)
Timer output	1 (TO0)
Control register	16-bit timer mode control register (TMC0) Capture/compare control register 0 (CRC0) 16-bit timer output control register (TOC0) Prescaler mode register 0 (PRM0)

Figure 8-1. Block Diagram of 16-Bit Timer/Counter (TM0)



(1) 16-bit timer register (TM0)

TM0 is a 16-bit read-only register that counts count pulses.

The counter is incremented in synchronization with the rising edge of an input clock. If the count value is read during operation, input of the count clock is temporarily stopped, and the count value at that point is read. The count value is reset to 0000H in the following cases:

<1> $\overline{\text{RESET}}$ is input .
<2> TMC03 and TMC02 are cleared.
<3> Valid edge of TI00 is input in the clear & start mode by inputting valid edge of TI00.
<4> TM0 and CR00 coincide with each other in the clear & start mode on coincidence between TM0 and CR00.
<5> Bit 6 of TOC0 (OSPT) is set or if the valid edge of TI00 is input in the one-shot pulse output mode.

(2) Capture/compare register 00 (CR00)

CR00 is a 16-bit register that functions as a capture register and as a compare register. Whether this register functions as a capture or compare register is specified by using bit 0 (CRC00) of the capture/compare control register 0.

- **When using CR00 as compare register**

  The value set to CR00 is always compared with the count value of the 16-bit timer register (TM0). When the values of the two coincide, an interrupt request (INTTM00) is generated. When TM00 is used as an interval timer, CR00 can also be used as a register that holds the interval time.

- **When using CR00 as capture register**

  The valid edge of the TI00 or TI01 pin can be selected as a capture trigger. The valid edges for TI00 and TI01 are set with the prescaler mode register 0 (PRM0).

  Tables 8-2 and 8-3 show the conditions that apply when the capture trigger is specified as the valid edge of the TI00 pin and the valid edge of the TI01 pin respectively.

Table 8-2. Valid Edge of TI00 Pin and Valid Edge of Capture Trigger of Capture/Compare Register

ES01	ES00	Valid Edge of TI01 Pin	Capture Trigger of CR00
0	0	Falling edge	Rising edge
0	1	Rising edge	Falling edge
1	0	Setting prohibited	Setting prohibited
1	1	Both rising and falling edges	No capture operation

Table 8-3. Valid Edge of TI01 Pin and Valid Edge of Capture Trigger of Capture/Compare Register

ES01	ES00	Valid Edge of TI01 Pin	Capture Trigger of CR00
0	0	Falling edge	Rising edge
0	1	Rising edge	Rising edge
1	0	Setting prohibited	Setting prohibited
1	1	Both rising and falling edges	Both rising and falling edges

CR00 is set by a 16-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input makes CR00 undefined.

Caution Set any value other than 0000H in CR00. When using the register as an event counter, a count for one-pulse can not be operated.

(3) Capture/compare register 01 (CR01)

This is a 16-bit register that can be used as a capture register and a compare register. Whether it is used as a capture register or compare register is specified by bit 2 (CRC02) of the capture/compare control register 0.

- **When using CR01 as compare register**

  The value set to CR01 is always compared with the count value of the 16-bit timer register (TM0). When the values of the two coincide, an interrupt request (INTTM01) is generated.

- **When using CR01 as capture register**

  The valid edge of the TI00 pin can be selected as a capture trigger. The valid edge for TI00 is set with the prescaler mode register 0 (PRM0).

  CR01 is set by a 16-bit memory manipulation instruction.

  $\overline{\text{RESET}}$ input makes CR01 undefined.

  Caution Set any value other than 0000H in CR01. When using an event counter, a count for one-pulse can not be operated.

8.3 16-Bit Timer/Counter Control Register

The following four types of registers control 16-bit timer/counter (TM0).

- 16-bit timer mode control register (TMC0)
- Capture/compare control register (CRC0)
- 16-bit timer output control register (TOC0)
- Prescaler mode register 0 (PRM0)

(1) 16-bit timer mode control register (TMC0)

This register specifies the operation mode of the 16-bit timer; and the clear mode, output timing, and overflow detection of the 16-bit timer register.

TMC0 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets TMC0 to 00H.

Caution The 16-bit timer register starts operating when a value other than 0, 0 (operation stop mode) is set to TMC02 and TMC03. To stop the operation, set 0, 0 to TMC02 and TMC03.

Figure 8-2. Format of 16-Bit Timer Mode Control Register (TMC0)

Address: 0FF18H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	⓪
TMC0	0	0	0	0	TMC03	TMC02	TMC01	OVF0

TMC03	TMC02	TMC01	Operating Mode Clear Mode and Clear Mode	Selection of TO0 Output Timing	Generation of Interrupt
0	0	0	Operation stop (TM0 is cleared to 0).	Not affected	Does not generate.
0	0	1			
0	1	0	Free running mode	Coincidence between TM0 and CR00 or coincidence between TM0 and CR01	Generates on coincidence between TM0 and CR00 and coincidence between TM0 and CR01.
0	1	1		Coincidence between TM0 and CR00, coincidence between TM0 and CR01, or valid edge of TI00	
1	0	0	Clears and starts at valid edge of TI00.	Coincidence between TM0 and CR00 or coincidence between TM0 and CR01	
1	0	1		Coincidence between TM0 and CR00, coincidence between TM0 and CR01, or valid edge of TI00	
1	1	0	Clears and starts on coincidence between TM0 and CR00.	Coincidence between TM0 and CR00 or coincidence between TM0 and CR01	
1	1	1		Coincidence between TM0 and CR00, coincidence between TM0 and CR01, or valid edge of TI00	

0VF0	Detection of Overflow of 16-Bit Timer Register
0	Overflows.
1	Does not overflow.

Cautions 1. Before changing the clear mode and TO0 output timing, be sure to stop the timer operation (reset TMC02 and TMC03 to 0, 0).

The valid edge of the TI00 pin is selected by using the prescaler mode register 0 (PRM0).

2. When a mode in which the timer is cleared and started on coincidence between TM0 and CR00, the OVF0 flag is set to 1 when the count value of TM0 changes from FFFFH to 0000H with CR00 set to FFFFH.

Remark TO0 : output pin of 16-bit timer/counter (TM0)
TI00 : input pin of 16-bit timer/counter (TM0)
TM0 : 16-bit timer register
CR00 : compare register 00
CR01 : compare register 01

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

This register controls the operation of the capture/compare registers (CR00 and CR01).

CRC0 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets CRC0 to 00H.

Figure 8-3. Format of Capture/Compare Control Register 0 (CRC0)

Address: FF16H After Reset: 04H R/W

Symbol	7	6	5	4	3	2	1	0
CRC0	0	0	0	0	0	CRC02	CRC01	CRC00

CRC02	Selection of Operation Mode of CR01
0	Operates as compare register.
1	Operates as capture register.

CRC01	Selection of Capture Trigger of CR00
0	Captured at valid edge of TI01.
1	Captured in reverse phase of valid edge of TI00.

CRC00	Selection of Operation Mode of CR00
0	Operates as compare register.
1	Operates as capture register.

Cautions 1. Before setting CRC0, be sure to stop the timer operation.

2. When the mode in which the timer is cleared and started on coincidence between TM0 and CR00 is selected by the 16-bit timer mode control register (TMC0), do not specify CR00 as a capture register.

(3) 16-bit timer output control register (TOC0)

This register controls the operation of the 16-bit timer/counter (TM0) output control circuit by setting or resetting the R-S flip-flop (LV0), enabling or disabling reverse output, enabling or disabling output of 16-bit timer/counter (TM0), enabling or disabling one-shot pulse output operation, and selecting an output trigger for a one-shot pulse by software.

TOC0 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets TOC0 to 00H.

Figure 8-4 shows the format of TOC0.

Figure 8-4. Format of 16-Bit Timer Output Control Register (TOC0)

Address: 0FF1AH After Reset: 00H R/W

Symbol	7	⑥	⑤	4	③	②	1	⓪
TOC0	0	OSPT	OSPE	TOC04	LVS0	LVR0	TOC01	TOE0

OSPT	Output Trigger Control of One-Shot Pulse by Software
0	No one-shot pulse trigger
1	Uses one-shot pulse trigger.

OSPE	Controls of One-Shot Pulse Output Operation
0	Successive pulse output
1	One-shot pulse output

TOC04	Timer Output F/F Control on Coincidence between CR01 and TM0
0	Disables reverse timer output F/F.
1	Enables reverse timer output F/F.

LVS0	LVR0	Set Status of Timer Output F/F of 16-Bit Timer/Counter (TM0)
0	0	Not affected
0	1	Resets timer output F/F (0).
1	0	Sets timer output F/F (1).
1	1	Setting prohibited

TOC01	Timer Output F/F Control on Coincidence between CR00 and TM0
0	Disables reverse timer output F/F.
1	Enables reverse timer output F/F.

TOE0	Output Control of 16-Bit Timer/Counter (TM0)
0	Disables output (output is fixed to 0 level).
1	Enables output.

Cautions 1. Before setting TOC0, be sure to stop the timer operation.

2. LVS0 and LVR0 are 0 when read after data have been set to them.

3. OSPT is 0 when read because it is automatically cleared after data has been set.

(4) Prescaler mode register 0 (PRM0)

This register selects a count clock of the 16-bit timer/counter (TM0) and the valid edge of TI00, TI01 input.

PRM0 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PRM0 to 00H.

Figure 8-5. Format of Prescaler Mode Register 0 (PRM0)

Address: 0FF1CH After Reset : 00H R/W

Symbol	7	6	5	4	3	2	1	0
PRM0	ES11	ES10	ES01	ES00	0	0	PRM01	PRM00

ES11	ES10	Selection of Valid Edge of TI01
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both falling and rising edges

ES01	ES00	Selection of Valid Edge of TI00
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both falling and rising edges

PRM01	PRM00	Selection of Count Clock
0	0	f_{XX}/4 (3.13 MHz)
0	1	f_{XX}/16 (781 kHz)
1	0	INTTM3 (Timer output for clock)
1	1	Valid edge of TI00

Caution When selecting the valid edge of TI00 as the count clock, do not specify the valid edge of TI00 to clear and start the timer and as a capture trigger.

Remark Figures in parentheses apply to operation with f_{XX} = 12.5 MHz

8.4 Operation

8.4.1 Operation as interval timer (16 bits)

16-bit timer/counter operates as an interval timer when the 16-bit timer mode control register (TMC0) and capture/compare control register 0 (CRC0) are set as shown in Figure 8-6.

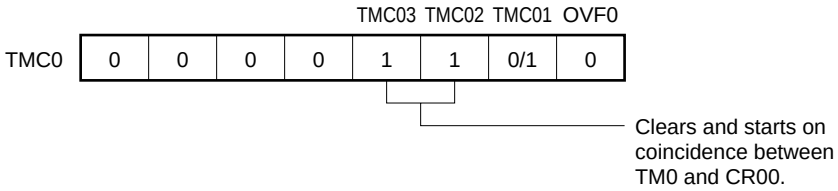
In this case, 16-bit timer/counter repeatedly generates an interrupt at the time interval specified by the count value set in advance to the 16-bit capture/compare register 00 (CR00).

When the count value of the 16-bit timer register (TM0) coincides with the set value of CR00, the value of TM0 is cleared to 0, and the timer continues counting. At the same time, an interrupt request signal (INTTM00) is generated.

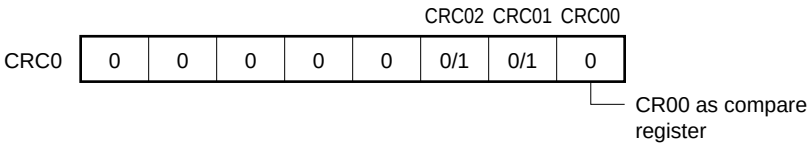
The count clock of the 16-bit timer/event counter can be selected by bits 0 and 1 (PRM00 and PRM01) of the prescaler mode register 0 (PRM0).

Figure 8-6. Control Register Settings when Timer 0 Operates as Interval Timer

(a) 16-bit timer mode control register (TMC0)



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



Remark 0/1 : When these bits are reset to 0 or set to 1, the other functions can be used along with the interval timer function. For details, refer to **Figures 8-2** and **8-3**.

Figure 8-7. Configuration of Interval Timer

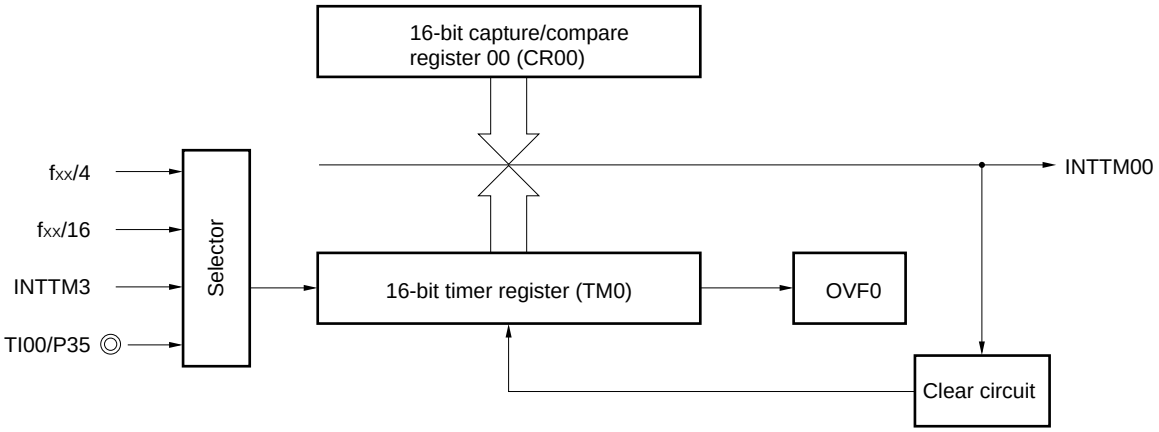
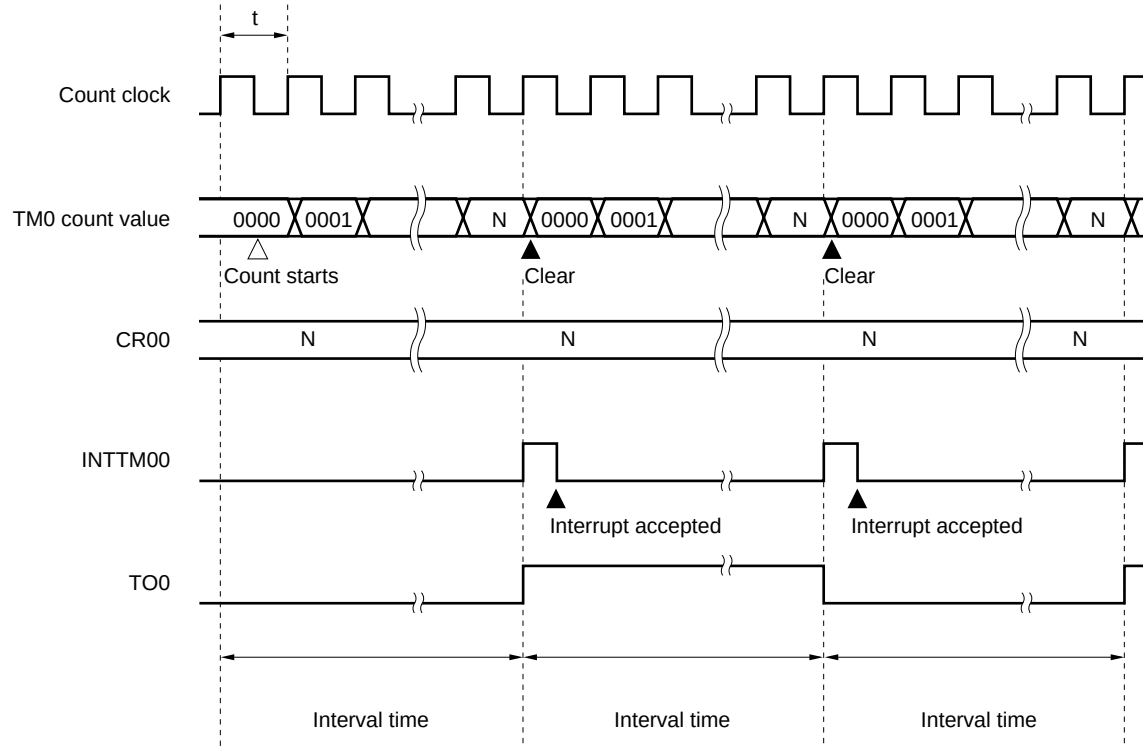


Figure 8-8. Timing of Interval Timer Operation



Remark Interval time = (N+1) × t: N = 00H to FFH

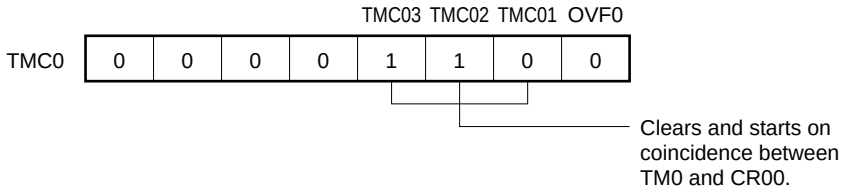
8.4.2 PPG output operation

16-bit timer/counter can be used for PPG (Programmable Pulse Generator) output by setting the 16-bit timer mode control register (TMC0) and capture/compare control register 0 (CRC0) as shown in Figure 8-9.

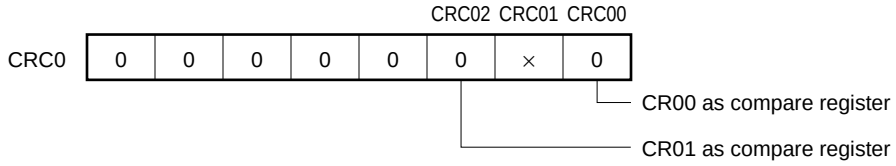
The PPG output function outputs a rectangular wave with a cycle specified by the count value set in advance to the 16-bit capture/compare register 00 (CR00) and a pulse width specified by the count value set in advance to the 16-bit capture/compare register 01 (CR01).

Figure 8-9. Control Register Settings in PPG Output Operation

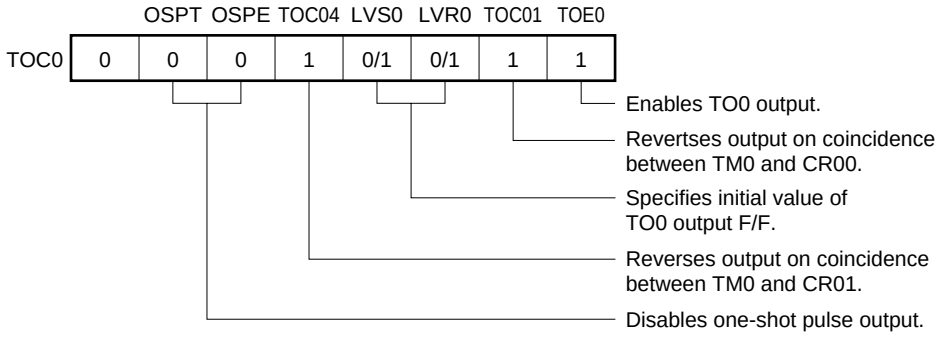
(a) 16-bit timer mode control register (TMC0)



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



(c) 16-bit timer output control register (TOC0)



Renark × : don't care

Caution Make sure that 0000H ≤ CR01 < CR00 ≤ FFFFH is set to CR00 and CR01.

8.4.3 Pulse width measurement

The 16-bit timer register (TM0) can be used to measure the pulse widths of the signals input to the TI00/P35 and TI01/P36 pins.

Measurement can be carried out with TM0 used as a free running counter or by restarting the timer in synchronization with the edge of the signal input to the TI00/P35 pin.

(1) Pulse width measurement with free running counter and one capture register

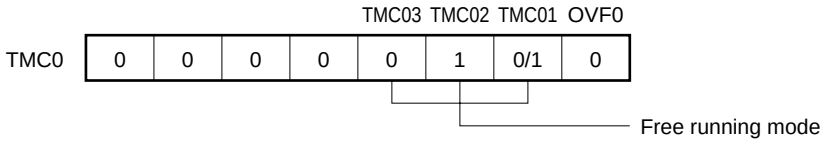
If the edge specified by the prescaler mode register 0 (PRM0) is input to the TI00/P35 pin when the 16-bit timer register (TM0) is used as a free running counter (refer to **Figure 8-10**), the value of TM0 is loaded to the 16-bit capture/compare register 01 (CR01), and an external interrupt request signal (INTTM00) is set.

The edge is specified by using bits 6 and 7 (ES10 and ES11) of the prescaler mode register 0 (PRM0). The rising edge, falling edge, or both the rising and falling edges can be selected.

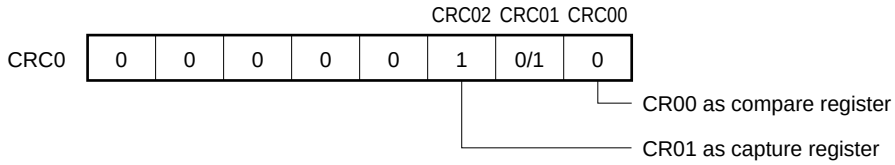
The valid edge is detected through sampling at a count clock cycle selected by the prescaler mode register 0n (PRM0), and the capture operation is not performed until the valid level is detected two times. Therefore, noise with a short pulse width can be rejected.

Figure 8-10. Control Register Settings for Pulse Width Measurement with Free Running Counter and One Capture Register

(a) 16-bit timer mode control register (TMC0)



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



Remark 0/1 : When these bits are reset to 0 or set to 1, the other functions can be used along with the pulse width measurement function. For details, refer to **Figures 8-2** and **8-3**.

Figure 8-11. Configuration for Pulse Width Measurement with Free Running Counter

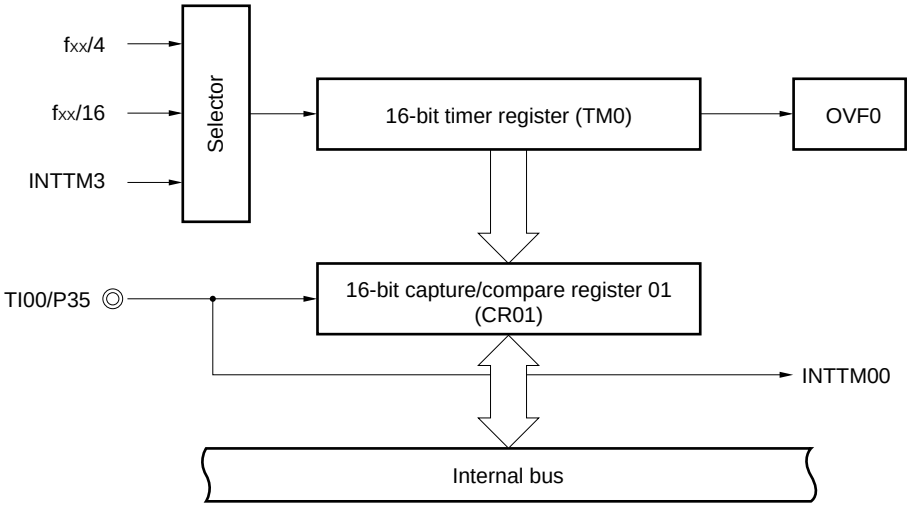
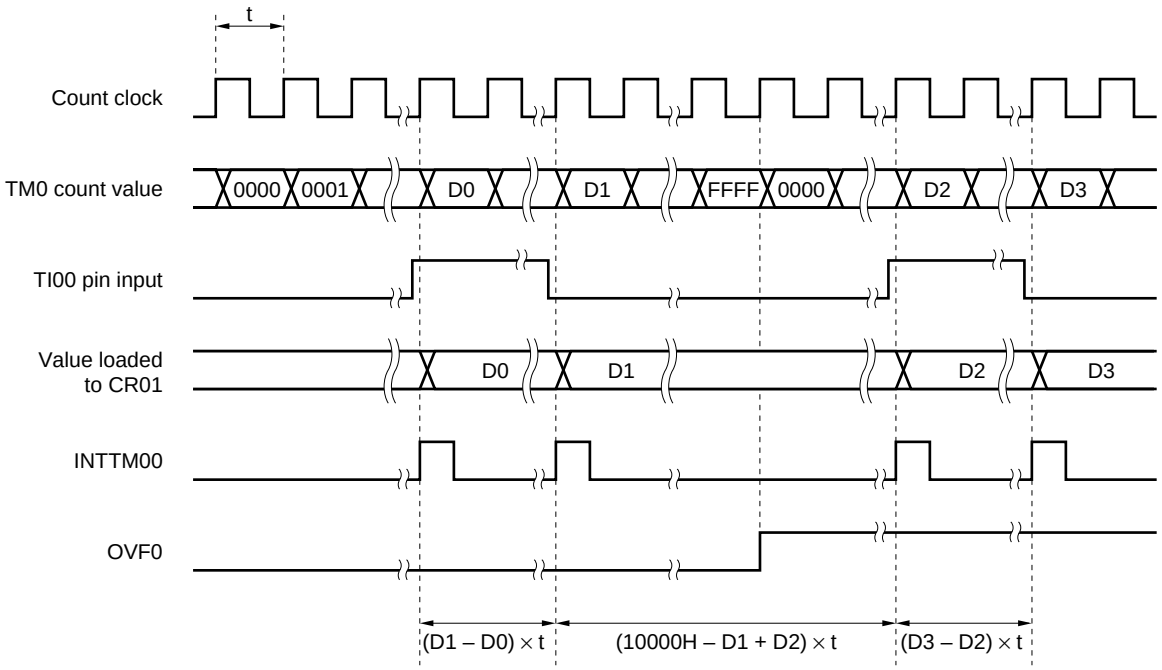


Figure 8-12. Timing of Pulse Width Measurement with Free Running Counter and One Capture Register (with both edges specified)



(2) Measurement of two pulse widths with free running counter

The pulse widths of the two signals respectively input to the TI00/P35 and TI01/P36 pins can be measured when the 16-bit timer register (TM0) is used as a free running counter (refer to **Figure 8-13**).

When the edge specified by bits 4 and 5 (ES10 and ES11) of the prescaler mode register 0 (PRM0) is input to the TI00/P35 pin, the value of the TM0 is loaded to the 16-bit capture/compare register 01 (CR01) and an external interrupt request signal (INTTM01) is set.

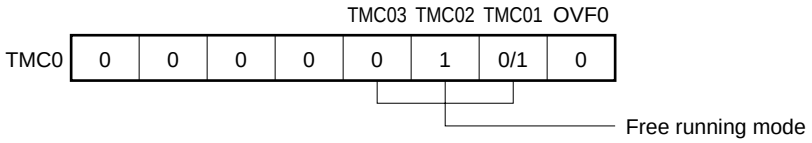
When the edge specified by bits 6 and 7 (ES20 and ES21) is input to the TI01/P36 pin, the value of TM0 is loaded to the 16-bit capture/compare register 00 (CR00), and an external interrupt request signal (INTTM00) is set.

The edges of the TI00/P35 and TI01/P36 pins are specified by bits 4 and 5 (ES00 and ES01) and bits 6 and 7 (ES10 and ES11) of PRM0, respectively. The rising, falling, or both rising and falling edges can be specified.

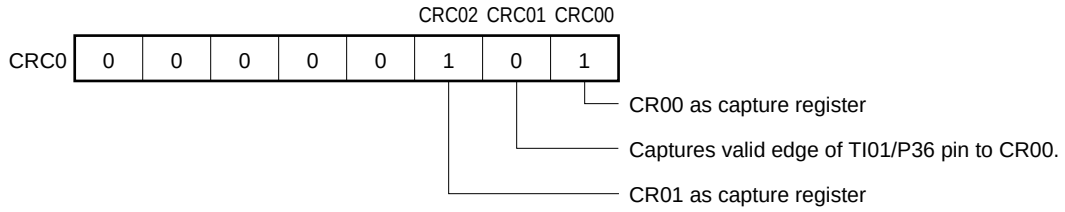
The valid edge of TI00/P35 pin and TI01/P36 pin is detected through sampling at a count clock cycle selected by the prescaler mode register 0 (PRM0), and the capture operation is not performed until the valid level is detected two times. Therefore, noise with a short pulse width can be rejected.

Figure 8-13. Control Register Settings for Measurement of Two Pulse Widths with Free Running Counter

(a) 16-bit timer mode control register (TMC0)



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



Remark 0/1 : When these bits are reset to 0 or set to 1, the other functions can be used along with the pulse width measurement function. For details, refer to **Figures 8-2** and **8-3**.

- **Capture operation (free running mode)**

  The following figure illustrates the operation of the capture register when the capture trigger is input.

Figure 8-14. CR01 Capture Operation with Rising Edge Specified

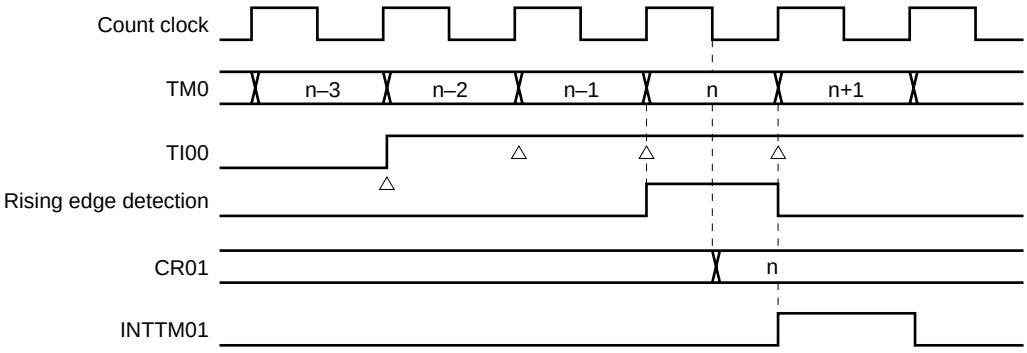
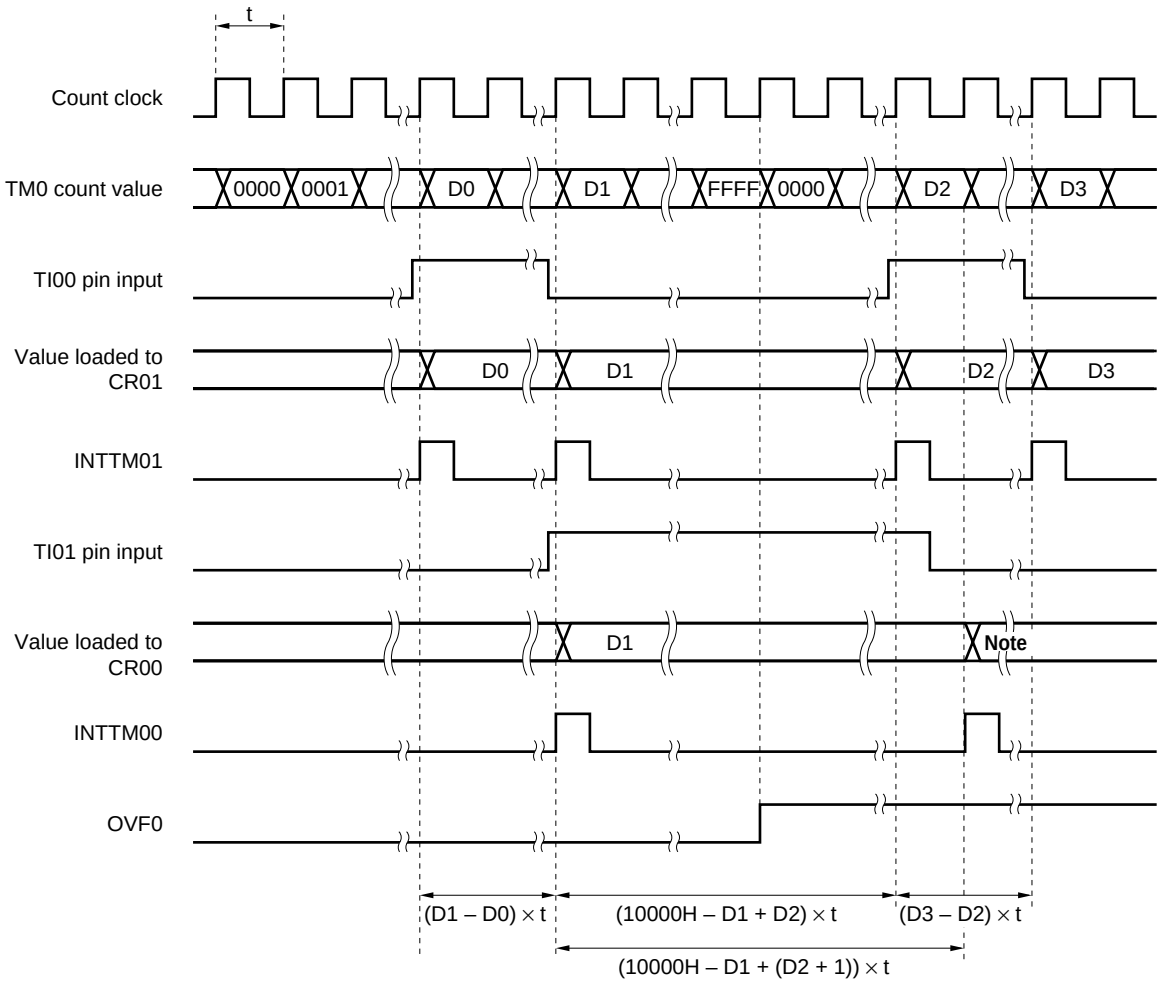


Figure 8-15. Timing of Pulse Width Measurement with Free Running Counter (with both edges specified)



Note D2 + 1

(3) Pulse width measurement with free running counter and two capture registers

When the 16-bit timer register (TM0) is used as a free running counter (refer to **Figure 8-16**), the pulse width of the signal input to the TI00/P35 pin can be measured.

When the edge specified by bits 4 and 5 (ES00 and ES01) of the prescaler mode register 0 (PRM0) is input to the TI00/P35 pin, the value of TM0 is loaded to the 16-bit capture/compare register 01 (CR01), and an external interrupt request signal (INTTM01) is set.

The value of TM0 is also loaded to the 16-bit capture/compare register 00 (CR00) when an edge reverse to the one that triggers capturing to CR01 is input.

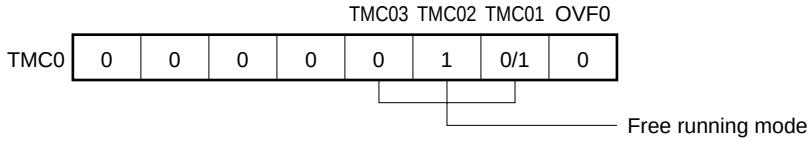
The edge of the TI00/P35 pin is specified by bits 4 and 5 (ES00 and ES01). The rising or falling edge can be specified.

The valid edge of TI00/P35 pin is detected through sampling at a count clock cycle selected by the prescaler mode register 0 (PRM0), and the capture operation is not performed until the valid level is detected two times. Therefore, noise with a short pulse width can be rejected.

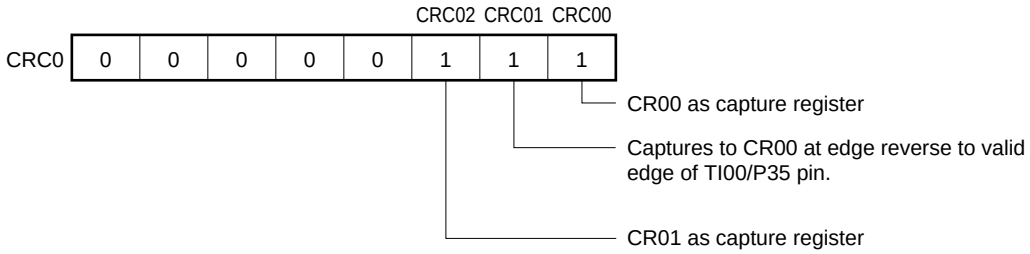
Caution If the valid edge of the TI00/P35 pin is specified to be both the rising and falling edges, the capture/compare register 00 (CR00) cannot perform its capture operation.

Figure 8-16. Control Register Settings for Pulse Width Measurement with Free Running Counter and Two Capture Registers

(a) 16-bit timer mode control register (TMC0)

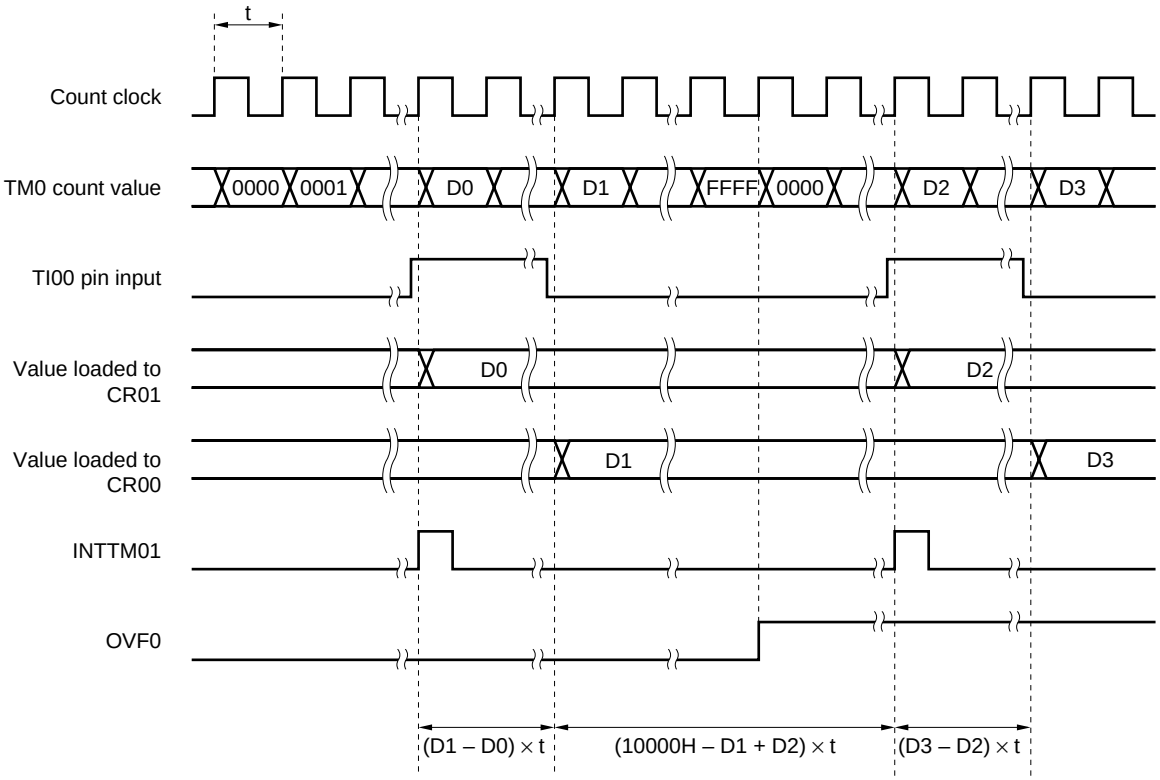


(b) Capture/compare control register 0 (CRC0)



Remark 0/1 : When these bits are reset to 0 or set to 1, the other functions can be used along with the pulse width measurement function. For details, refer to **Figures 8-2** and **8-3**.

Figure 8-17. Timing of Pulse Width Measurement with Free Running Counter and Two Capture Registers (with rising edge specified)



(4) Pulse width measurement by restarting

When the valid edge of the TI00/P35 pin is detected, the pulse width of the signal input to the TI00/P35 pin can be measured by clearing the 16-bit timer register (TM0) once and then resuming counting after loading the count value of TM0 to the 16-bit capture/compare register 01 (CR01) (Refer to **Figure 8-18**).

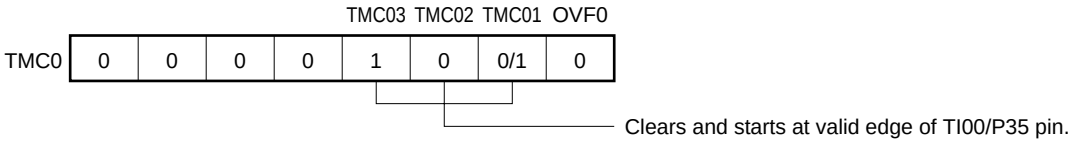
The edge of the TI00/P35 pin is specified by bits 4 and 5 (ES00 and ES01) of the prescaler mode register 0 (PRM0). The rising or falling edge can be specified.

The valid edge is detected through sampling at a count clock cycle selected by the prescaler mode register 0 (PRM0), and the capture operation is not performed until the valid level is detected two times. Therefore, noise with a short pulse width can be rejected.

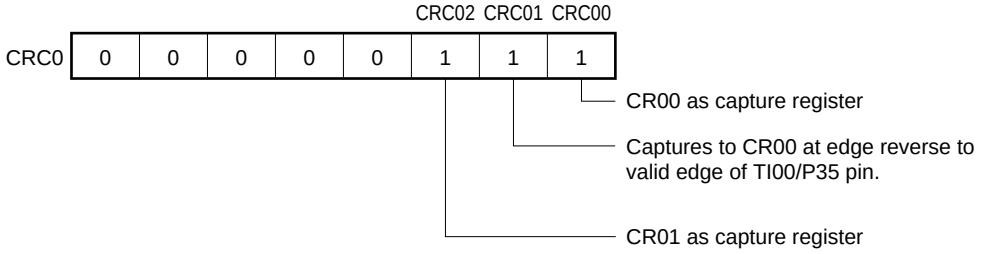
Caution If the valid edge of the TI00/P35 pin is specified to be both the rising and falling edges, the capture/compare register 00 (CR00) cannot perform its capture operation.

Figure 8-18. Control Register Settings for Pulse Width Measurement by Restarting

(a) 16-bit timer mode control register (TMC0)

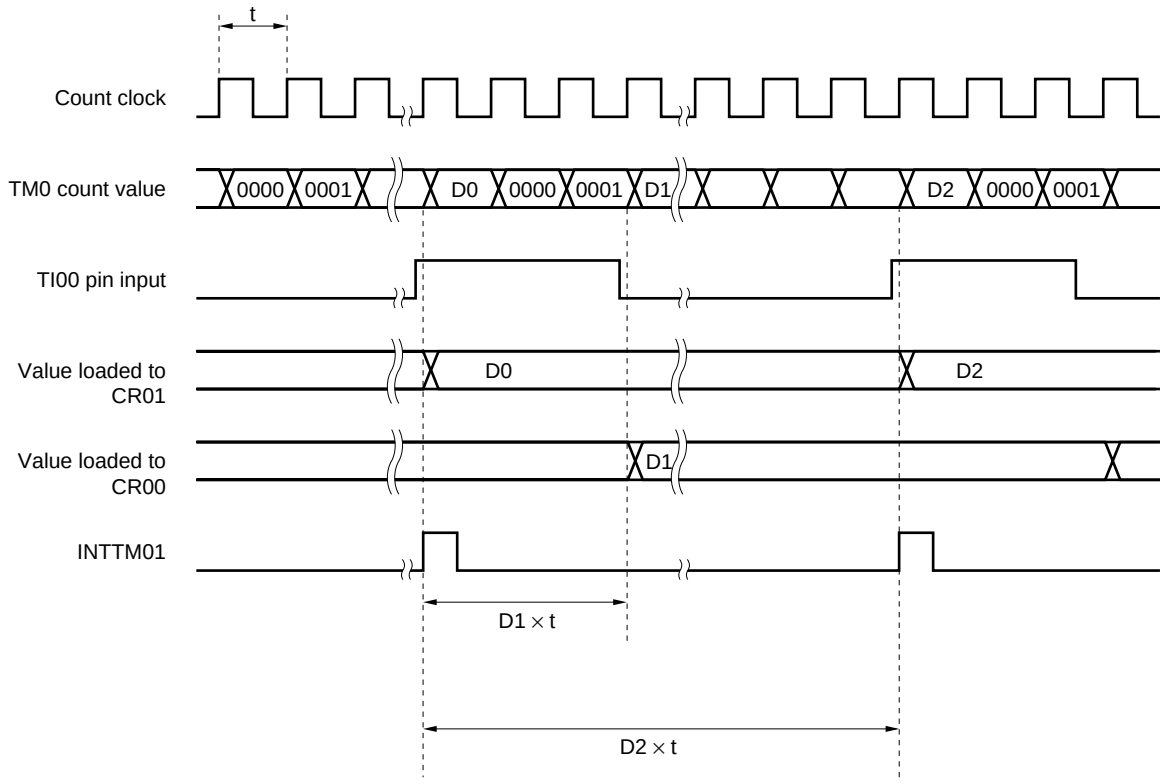


(b) Capture/compare control register 0 (CRC0)



Remark 0/1 : When these bits are reset to 0 or set to 1, the other functions can be used along with the pulse measurement function. For details, refer to **Figures 8-2** and **8-3**.

Figure 8-19. Timing of Pulse Width Measurement by Restarting (with rising edge specified)



8.4.4 Operation as external event counter

16-bit timer /counter can be used as an external event counter which counts the number of clock pulses input to the TI00/P35 pin from an external source by using the 16-bit timer register (TM0).

Each time the valid edge specified by the prescaler mode register 0 (PRM0) has been input to the TI00/P35 pin, TM0 is incremented.

When the count value of TM0 coincides with the value of the 16-bit capture/compare register 00 (CR00), TM0 is cleared to 0, and an interrupt request signal (INTTM00) is generated.

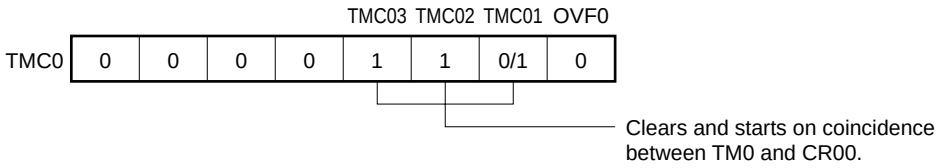
Set any value other than 0000H in CR00 (single pulse count can not be operated.)

The edge of the TI00/P35 pin is specified by bits 4 and 5 (ES00 and ES01) of the prescaler mode register 0 (PRM0). The rising, falling, or both the rising and falling edges can be specified.

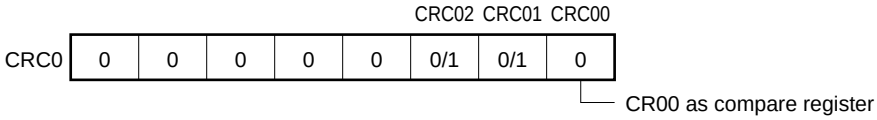
The valid edge is detected through sampling at a count clock cycle selected by the prescaler mode register 0 (PRM0), and the capture operation is not performed until the valid level is detected two times. Therefore, noise with a short pulse width can be rejected.

Figure 8-20. Control Register Settings in External Event Counter Mode

(a) 16-bit timer mode control register (TMC0)



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



Remark 0/1 : When these bits are reset to 0 or set to 1, the other functions can be used along with the external event counter function. For details, refer to **Figures 8-2** and **8-3.**

Figure 8-21. Configuration of External Event Counter

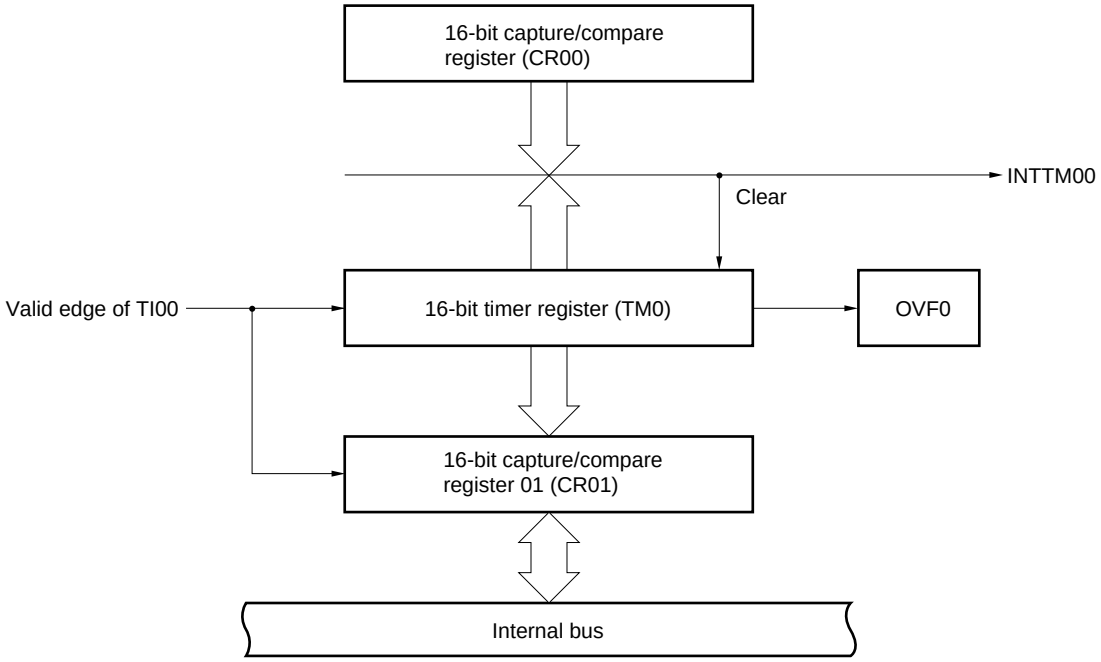
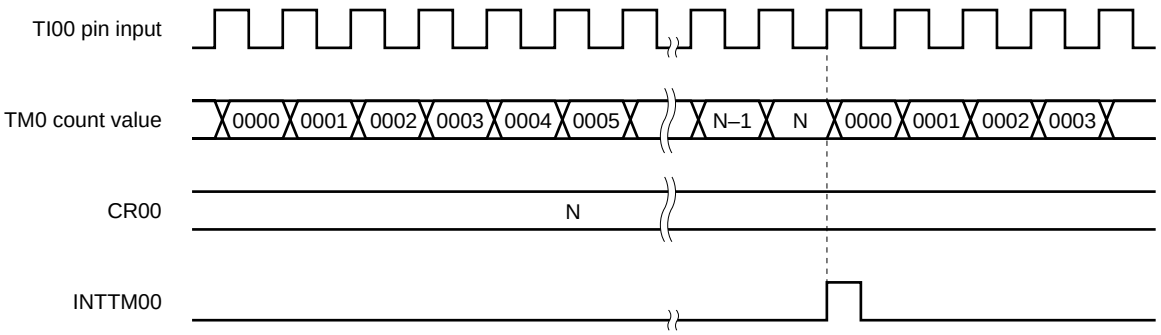


Figure 8-22. Timing of External Event Counter Operation (with rising edge specified)



Caution Read TM0 when reading the count value of the external event counter.

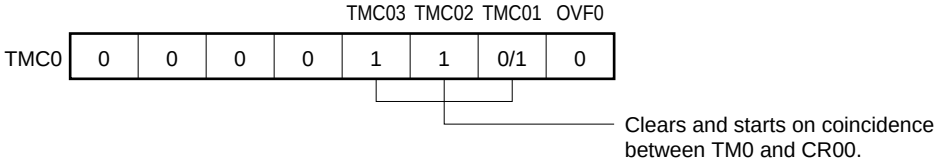
8.4.5 Operation to output square wave

Operates as the square wave output for the user-defined frequency that is used as the interval for the count value previously set in the 16-bit capture/compare register 00 (CR00).

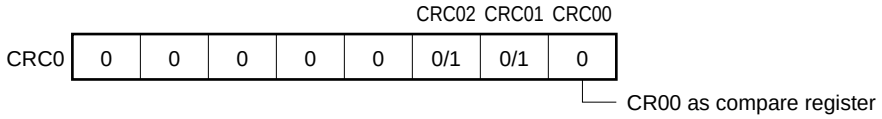
By setting bits 0 (TOE0) and 1 (TOC01) of the 16-bit timer output control register (TOC0) to 1, the output status of the TO0/P30 pin is reversed at an interval specified by the count value set in advance to CR00. In this way, a square wave of any frequency can be output.

Figure 8-23. Set Contents of Control Registers in Square Wave Output Mode

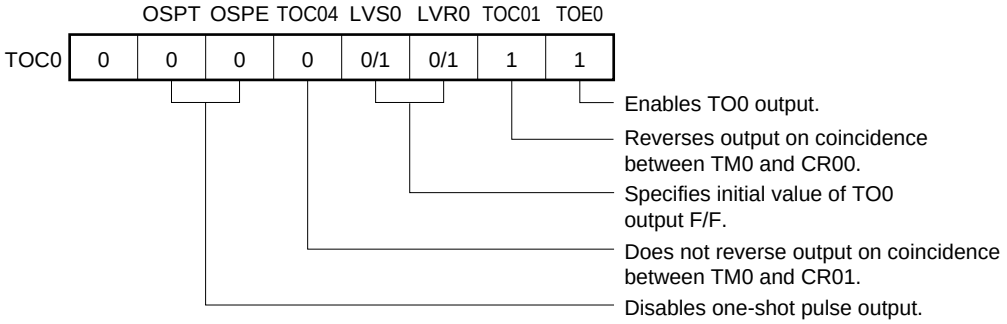
(a) 16-bit timer mode control register (TMC0)



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

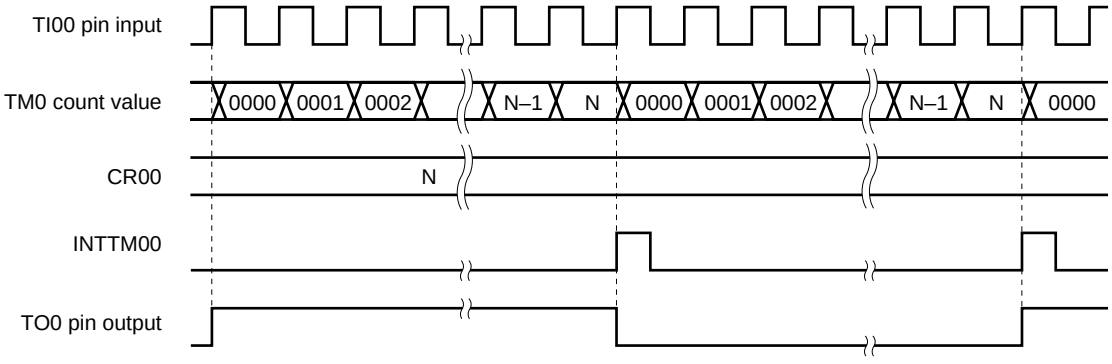


(c) 16-bit timer output control register (TOC0)



Remark 0/1 : When these bits are reset to 0 or set to 1, the other functions can be used along with the square wave output function. For details, refer to **Figures 8-2**, **8-3**, and **8-4**.

Figure 8-24. Timing of Square Wave Output Operation



8.4.6 Operation to output one-shot pulse

16-bit timer/counter can output a one-shot pulse in synchronization with a software trigger and an external trigger (TI00/P35 pin input).

(1) One-shot pulse output with software trigger

A one-shot pulse can be output from the TO0/P30 pin by setting the 16-bit timer mode control register (TMC0), capture/compare control register 0 (CRC0), and 16-bit timer output control register (TOC0) as shown in Figure 8-25, and by setting bit 6 (OSPT) of TOC0 by software.

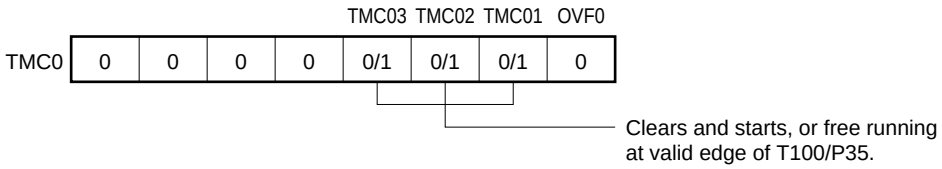
By setting OSPT to 1, the 16-bit timer/event counter is cleared and started, and its output is asserted active at the count value set in advance to the 16-bit capture/compare register 01 (CR01). After that, the output is deasserted inactive at the count value set in advance to the 16-bit capture/compare register 00 (CR00).

Even after the one-shot pulse has been output, TM0 continues its operation. To stop TM0, TMC0 must be reset to 00H.

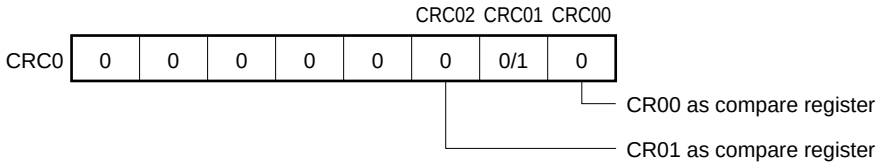
Caution Do not set OSPT to 1 while the one-shot pulse is being output. To output the one-shot pulse again, wait until INTTM00, which occurs on coincidence between TM0 and CR00, occurs.

Figure 8-25. Control Register Settings for One-Shot Pulse Output with Software Trigger

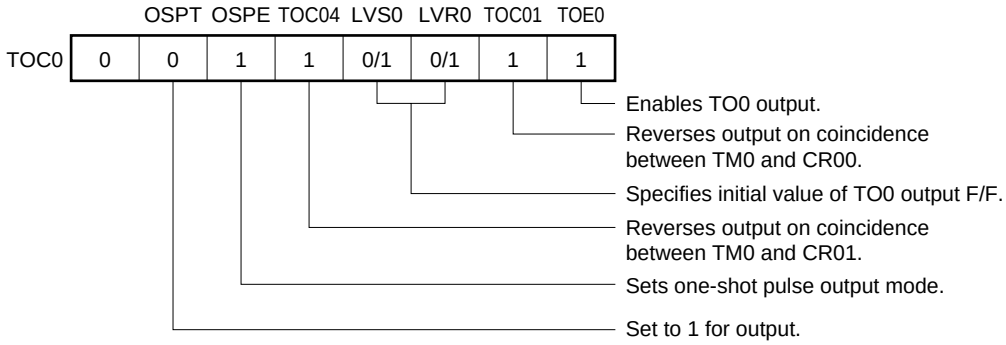
(a) 16-bit timer mode control register (TMC0)



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



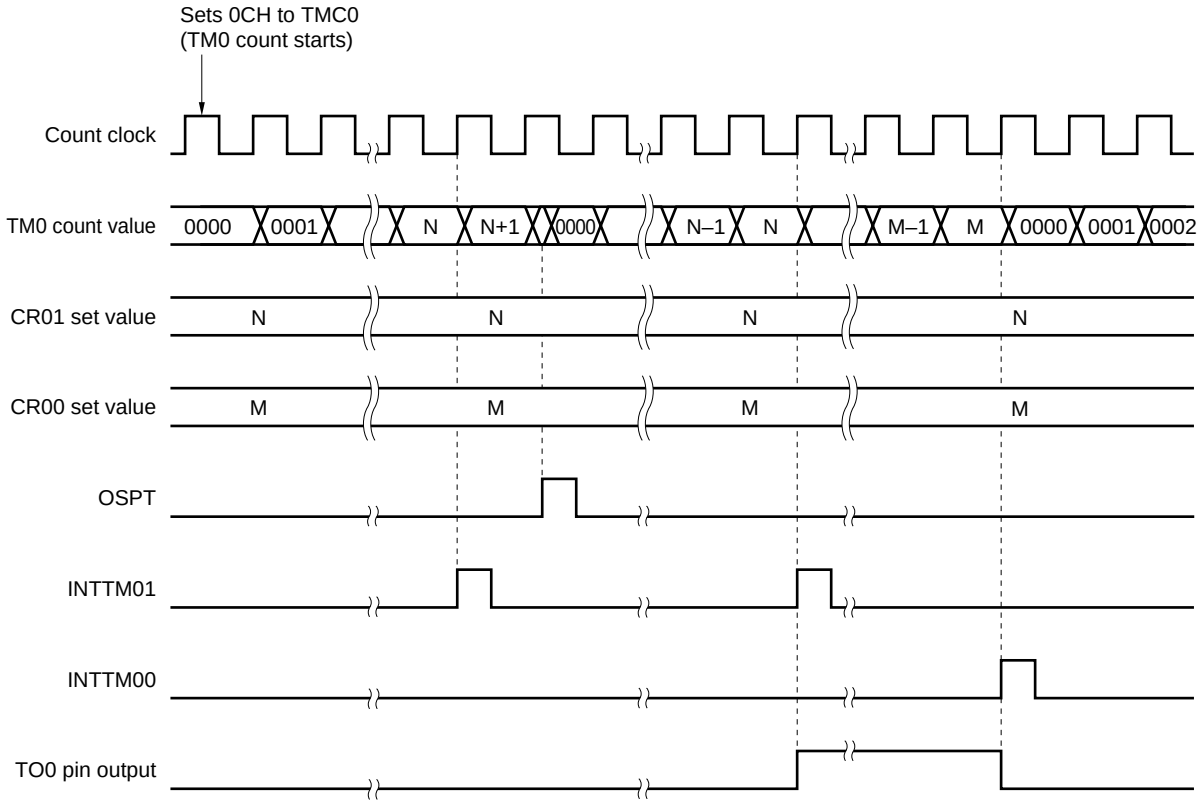
(c) 16-bit timer output control register (TOC0)



Caution Set a value in the following range to CR00 and CR01.
0000H < CR01 < CR00 - FFFFH

Remark 0/1 : When these bits are reset to 0 or set to 1, the other functions can be used along with the one-shot pulse output function. For details, refer to **Figures 8-2**, **8-3**, and **8-4**.

Figure 8-26. Timing of One-Shot Pulse Output Operation with Software Trigger



Caution The 16-bit timer register starts operating as soon as a value other than 0, 0 (operation stop mode) has been set to TMC02 and TMC03.

(2) One-shot pulse output with external trigger

A one-shot pulse can be output from the TO0/P30 pin by setting the 16-bit timer mode control register (TMC0), capture/compare control register 0 (CRC0), and 16-bit timer output control register (TOC0) as shown in Figure 8-27, and by using the valid edge of the TI00/P35 pin as an external trigger.

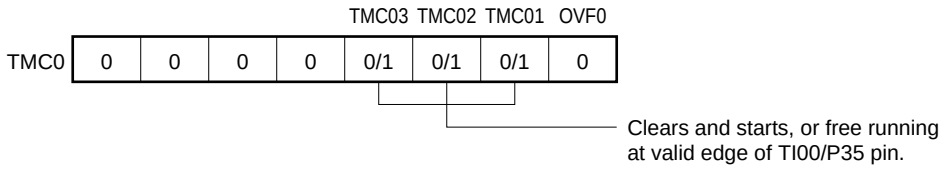
The valid edge of the TI00/P35 pin is specified by bits 4 and 5 (ES00 and ES01) of the prescaler mode register 0 (PRM0). The rising, falling, or both the rising and falling edges can be specified.

When the valid edge of the TI00/P35 pin is detected, the 16-bit timer/event counter is cleared and started, and the output is asserted active at the count value set in advance to the 16-bit capture/compare register 01 (CR01). After that, the output is deasserted inactive at the count value set in advance to the 16-bit capture/compare register 00 (CR00).

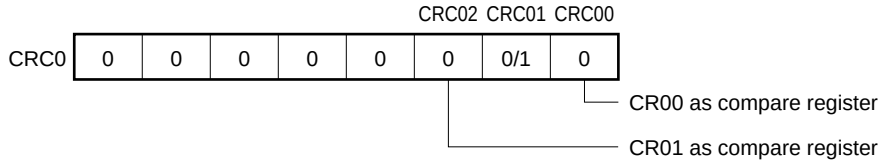
Caution Even if the external trigger is generated again while the one-shot pulse is output, it is ignored.

Figure 8-27. Control Register Settings for One-Shot Pulse Output with External Trigger

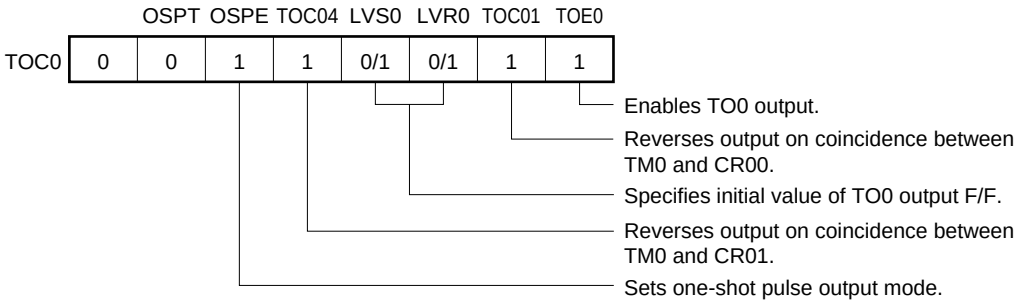
(a) 16-bit timer mode control register (TMC0)



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



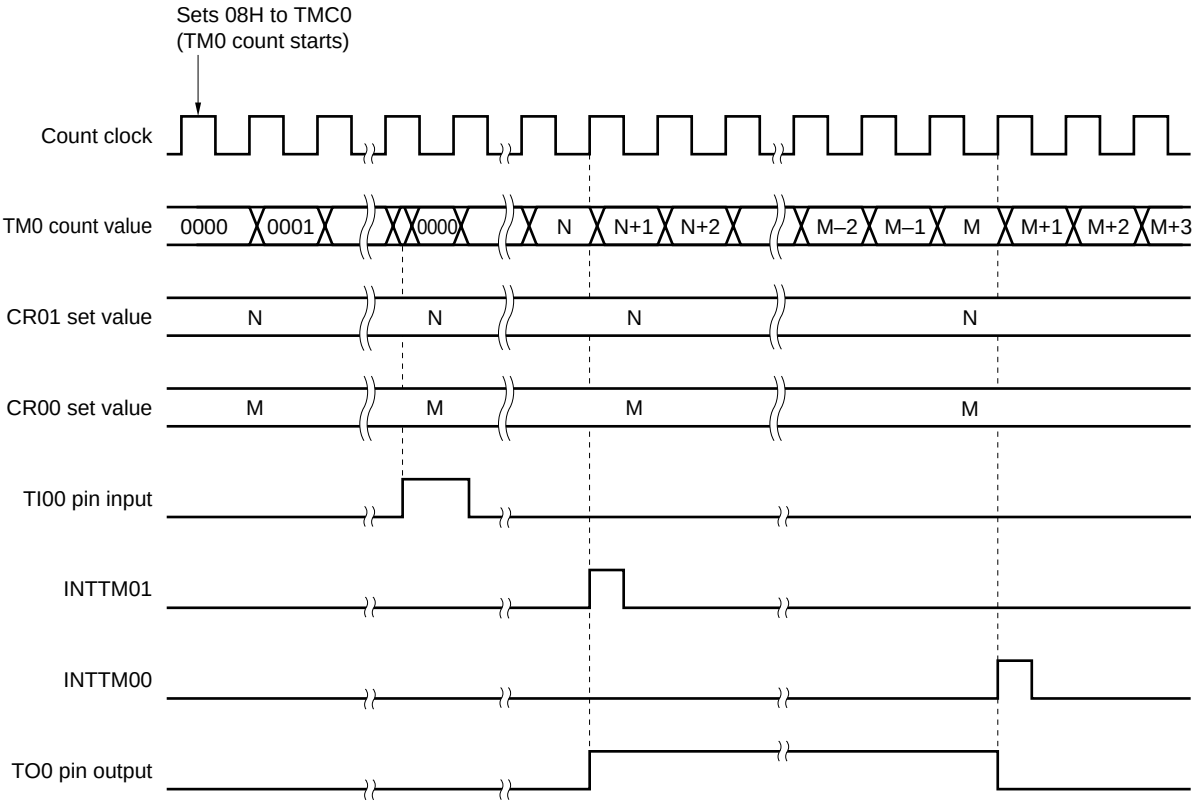
(c) 16-bit timer output control register (TOC0)



Caution Set a value in the following range to CR00 and CR01.
0000H < CR01 < CR00 ≤ FFFFH

Remark 0/1 : When these bits are reset to 0 or set to 1, the other functions can be used along with the one-shot pulse output function. For details, refer to **Figures 8-2**, **8-3,** and **8-4**.

Figure 8-28. Timing of One-Shot Pulse Output Operation with External Trigger (with rising edge specified)



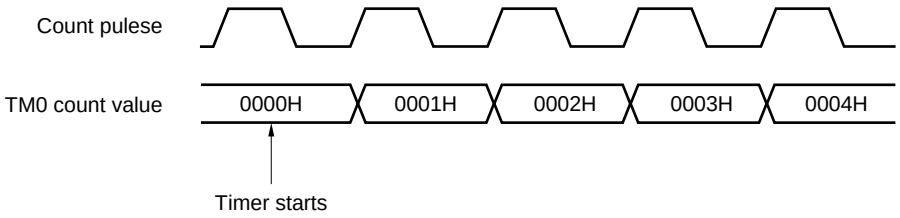
Caution The 16-bit timer register starts operating as soon as a value other than 0, 0 (operation stop mode) has been set to TMC02 and TMC03.

8.5 Cautions

(1) Error on starting timer

An error of up to 1 clock occurs before the coincidence signal is generated after the timer has been started. This is because the 16-bit timer register (TM0) is started asynchronously in respect to the count pulse.

Figure 8-29. Start Timing of 16-Bit Timer Register



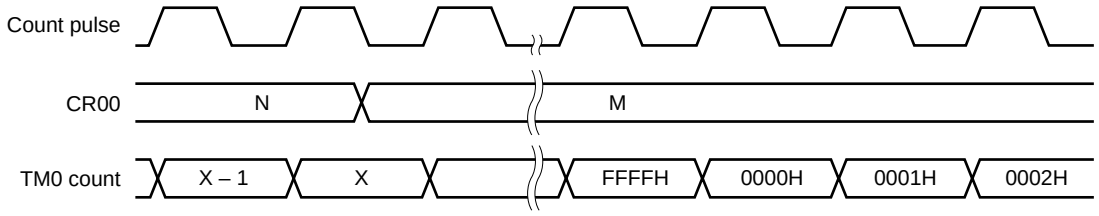
(2) 16-bit compare register settings

Set any value other than 0000H in the 16-bit capture/compare registers 00 and 01 (CR00, 01).
Single pulse counts are consequently not possible for the usage time, used as an event counter.

(3) Setting compare register during timer count operation

If the value to which the current value of the 16-bit capture/compare register 00 (CR00) has been changed is less than the value of the 16-bit timer register (TM0), TM0 continues counting, overflows, and starts counting again from 0. If the new value of CR00 (M) is less than the old value (N), the timer must be restarted after the value of CR00 has been changed.

Figure 8-30. Timing after Changing Compare Register during Timer Count Operation

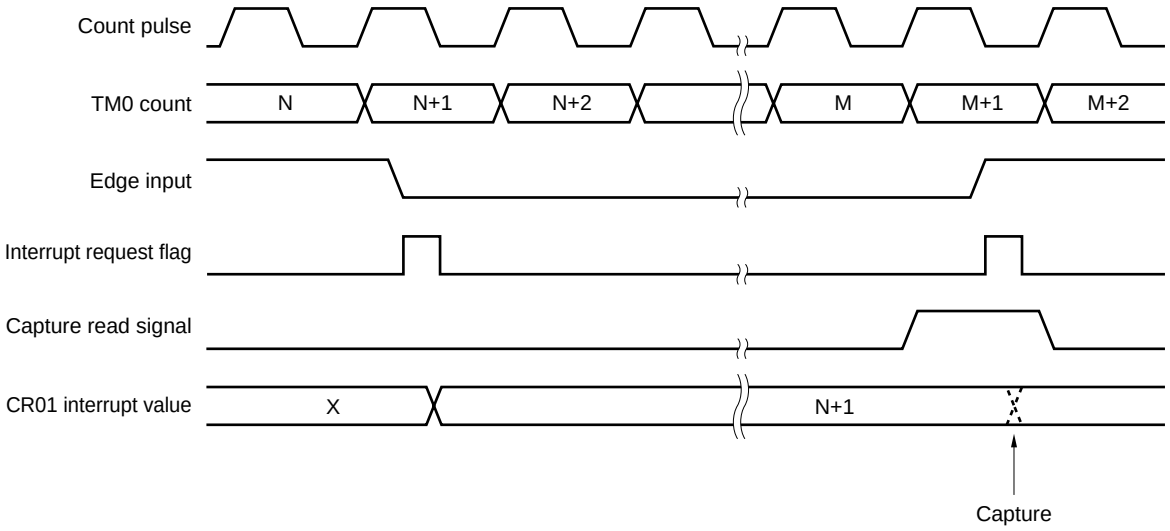


Remark N > X > M

(4) Data hold timing of capture register

If the valid edge is input to the TI00/P35 pin while the 16-bit capture/compare register 01 (CR01) is read, CR01 performs the capture operation, but this capture value is not guaranteed. However, the interrupt request flag (INTTM01) is set as a result of detection of the valid edge

Figure 8-31. Data Hold Timing of Capture Register



(5) Setting valid edge

The valid edge of the TI00/P35 terminal sets 0,0 in bits 2 and 3 of the 16-bit timer mode control register (TMC0), and this setting should be made the moment timer operations have been halted. The valid edge is set with bits 4 and 5 of the prescaler mode register 0 (ES00, ES01).

(6) Re-triggering one-shot pulse

(a) One-shot pulse output by software

When a one-shot pulse is output, do not set OSPT to 1. Do not output the one-shot pulse again until INTTM00, which occurs on coincidence between TM0 and CR00, occurs.

(b) One-shot pulse output with external trigger

If the external trigger occurs while a one-shot pulse is output, it is ignored.

(7) Operation of OVF0 flag

The OVF0 flag is set to 1 in the following case:

Select mode in which 16-bit timer/counter is cleared and started on coincidence between TM0 and CR00

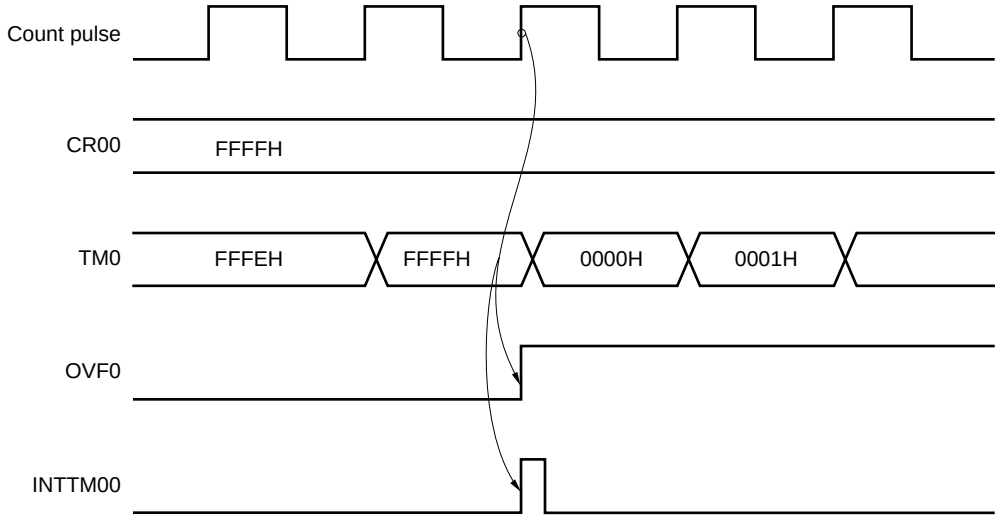
↓

Set CR00 to FFFFH.

↓

When TM0 counts up from FFFFH to 0000H

Figure 8-32. Operation Timing of OVF0 Flag



(8) Contention operation

1. Contention between the read period of 16-bit capture/compare registers (CR00 and CR01) and the capture trigger input (CR00 and CR01 are used as capture registers.)
   The capture trigger input is preceded. The read data of CR00 and CR01 is undefined.

2. Match timing contention between the write period of 16-bit capture/compare registers (CR00 and CR01) and 16-bit timer register (TM0). (CR00 and CR01 are used as compare registers.)
   A match discrimination is not normally performed. Do not perform the write operation of CR00 and CR01 around the match timing.

[MEMO]

CHAPTER 9 8-BIT TIMER/COUNTER 1, 2

9.1 Functions

8-bit timer/counter 1, 2 (TM1, TM2) have the following two modes.

- Mode using 8-bit timer/counter 1, 2 (TM1, TM2) alone (individual mode)
- Mode using the cascade connection (16-bit resolution: cascade connection mode)

These two modes are described next.

(1) Mode using 8-bit timer/counter 1, 2 alone (individual mode)

The timer operates as an 8-bit timer/event counter.
It can have the following functions.

- Interval timer
- External event counter
- Square wave output
- PWM output

(2) Mode using the cascade connection (16-bit resolution: cascade connection mode)

The timer operates as a 16-bit timer/event counter by connecting in cascade.
It can have the following functions.

- Interval timer with 16-bit resolution
- External event counter with 16-bit resolution
- Square wave output with 16-bit resolution

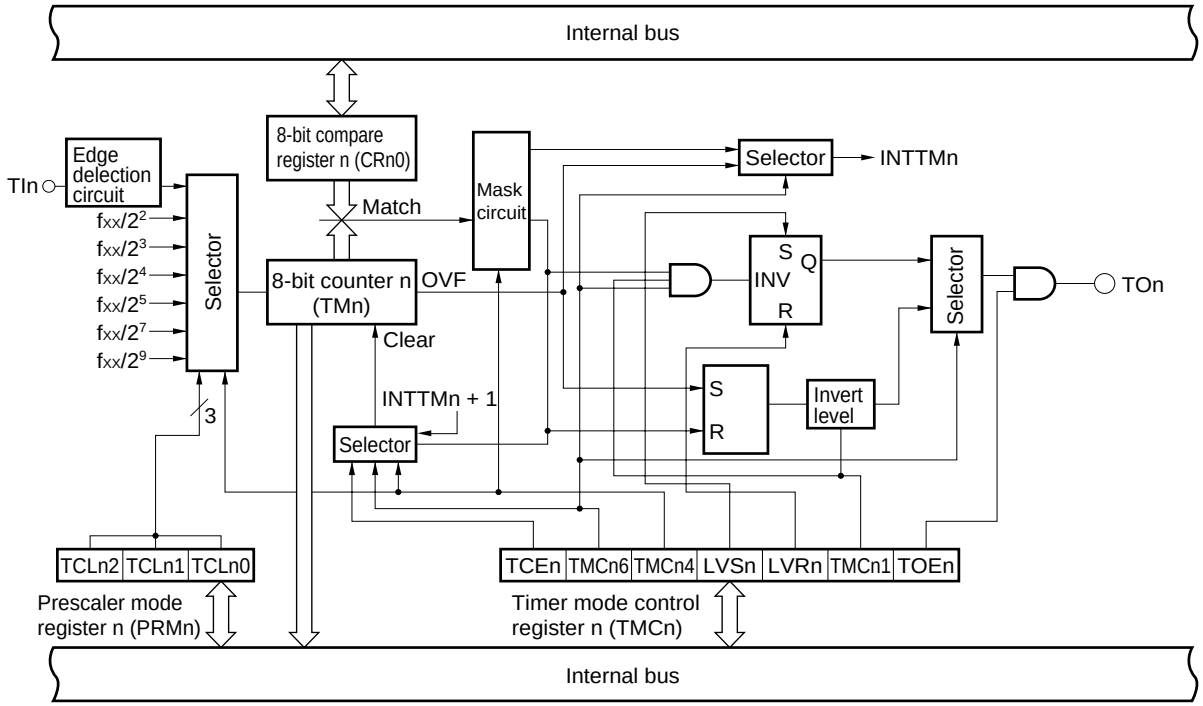
9.2 Configuration

8-bit timer/counter 1, 2 are constructed from the following hardware.

Table 9-1. 8-Bit Timer/Counter 1, 2 Configuration

Item	Configuration
Timer register	8-bit × 2 (TM1, TM2)
Register	8-bit × 2 (CR10, CR20)
Timer output	2 (TO1, TO2)
Control register	8-bit timer mode control register 1 (TMC1) 8-bit timer mode control register 2 (TMC2) Prescaler mode register 1 (PRM1) Prescaler mode register 2 (PRM2)

Figure 9-1. Block Diagram of 8-Bit Timer/Counter 1, 2



Remark n = 1, 2

(1) 8-bit timer register 1, 2 (TM1, TM2)

TM1and TM2 are an 8-bit read-only registers that counts the count pulses.

The counter is incremented synchronous to the rising edge of the count clock. When the count is read out during operation, the count clock input temporarily stops and the count is read at that time. In the following cases, the count becomes 00H.

<1> $\overline{\text{RESET}}$ is input.
<2> TCEn is cleared.
<3> TMn and CRn0 match in the clear and start mode.

Caution During the cascade connection the count becomes 00H even when TCE1 in TM1 is cleared.

Remark n = 1, 2

(2) 8-bit compare register (CR10, CR20)

The value set in CR10 and CR20 are compared to the count in the 8-bit timer register 1 (TM1) and 8-bit timer register 2 (TM2), respectively. If the two values match, interrupt requests (INTTM1, INTTM2) is generated (except in the PWM mode).

The values of CR10 and CR20 can be set in the range of 00H to FFH, and can be written during counting.

Caution If data is set in a cascade connection, always set after stopping the timer.

9.3 Control Registers

The following four registers control 8-bit timer/counter 1, 2.

- 8-bit timer mode control register 1, 2 (TMC1, TMC2)
- Prescaler mode register 1, 2 (PRM1, PRM2)

(1) 8-bit timer mode control register 1, 2 (TMC1, TMC2)

The TMC1, TMC2 registers make the following six settings.

<1> Controls the counting for the 8-bit timer register 1, 2 (TM1, TM2).
<2> Selects the operating mode of the 8-bit timer register 1, 2 (TM1, TM2).
<3> Selects the individual mode or cascade mode.
<4> Sets the state of the timer output flip-flop.
<5> Controls the timer flip-flop or selects the active level during the PWM (free running) mode.
<6> Controls timer output.

TMC1 and TMC2 are set by a 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input sets TMC1 and TMC2 to 00H.
Figures 9-2 and 9-3 show the TMC1 format and TMC2 format respectively.

Figure 9-2. Format of the 8-Bit Timer Mode Control Register 1 (TMC1)

Address: 0FF54H After Reset: 00H R/W

Symbol	⑦	6	5	4	③	②	1	⓪
TMC1	TCE1	TMC16	0	0	LVS1	LVR1	TMC11	TOE1

TCE1	TM1 Count Control
0	Counting is deisabled (prescaler disabled) after the counter is cleared to 0.
1	Start counting

TMC16	TM1 Operating Mode Selection
0	Clear and start mode whebn TM1 and CR10 match.
1	PWM (free running) mode

LVS1	LVR1	Setting the State of the Timer Output Flip-Flop
0	0	No change
0	1	Reset the timer output filp-flop (to 0).
1	0	Set the timer output flip-flop (to 1).
1	1	Setting prohibit

TMC11	Other than PWM Mode (TMC16 = 0)	PWM Mode (TMC16 = 1)
	Timer flip-flop control	Active level selection
0	Disable inversion operation	Active high
1	Enable inversion operation	Active low

TOE1	Timer Output Control
0	Disable ouput (port mode)
1	Enable ouput

Remarks 1. In the PWM mode, the PWM output is set to the inactive level by TCE1 = 0.

2. If LVS1 and LVR1 are read after setting data, 0 is read.

Figure 9-3. Format of the 8-Bit Timer Mode Control Register 2 (TMC2)

Address: 0FF55H After Reset: 00H R/W

Symbol	⑦	6	5	4	③	②	1	⓪
TMC2	TCE2	TMC26	0	TMC24	LVS2	LVR2	TMC21	TOE2

TCE2	TM2 Count Control
0	Counting is deisabled (prescaler disabled) after the counter is cleared to 0.
1	Start counting

TMC26	TM2 Operating Mode Selection
0	Clear and start mode whebn TM2 and CR20 match
1	PWM (free running) mode

TMC24	Individual Mode or Cascade Connection Mode Selection
0	Individual mode
1	Cascade connection mode (connection with TM1)

LVS2	LVR2	Setting the State of the Timer Output Flip-Flop
0	0	No change
0	1	Reset the timer output filp-flop (to 0).
1	0	Set the timer output flip-flop (to 1).
1	1	Setting prohibit

TMC21	Other than PWM Mode (TMC26 = 0)	PWM Mode (TMC26 = 1)
	Timer flip-flop control	Active level selection
0	Disable inversion operation	Active high
1	Enable inversion operation	Active low

TOE2	Timer Output Control
0	Disable ouput (port mode)
1	Enable ouput

Remarks 1. In the PWM mode, the PWM output is set to the inactive level by TCE2 = 0.

2. If LVS2 and LVR2 are read after setting data, 0 is read.

(2) Prescaler mode register 1, 2 (PRM1, PRM2)

This register sets the count clock of 8-bit timer register 1, 2 (TM1, TM2) and the effective edge of TI1, TI2 inputs.

PRM1and PRM2 are set by an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PRM1 and PRM2 to 00H.

Figure 9-4. Format of the Prescaler Mode Register 1 (PRM1)

Address: 0FF56H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
PRM1	0	0	0	0	0	TCL12	TCL11	TCL10

TCL12	TCL11	TCL10	Count Clock Selection
0	0	0	Falling edge of TI1
0	0	1	Rising edge of TI1
0	1	0	$f_{XX}/4$ (3.13 MHz)
0	1	1	$f_{XX}/8$ (1.56 MHz)
1	0	0	$f_{XX}/16$ (781 kHz)
1	0	1	$f_{XX}/32$ (391 kHz)
1	1	0	$f_{XX}/128$ (97.6 kHz)
1	1	1	$f_{XX}/512$ (24.4 kHz)

Cautions 1. If writing data different than that of PRM1 is written, stop the timer beforehand.

2. Be sure to set bits 3 to 7 of PRM1 to 0.

Remark Figures in parentheses apply to operation with f_{XX} = 12.5 MHz.

Figure 9-5. Format of the Prescaler Mode Register 2 (PRM2)

Address: 0FF57H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
PRM2	0	0	0	0	0	TCL22	TCL21	TCL20

TCL22	TCL21	TCL20	Count Clock Selection
0	0	0	Falling edge of TI2
0	0	1	Rising edge of TI2
0	1	0	$f_{XX}/4$ (3.13 MHz)
0	1	1	$f_{XX}/8$ (1.56 MHz)
1	0	0	$f_{XX}/16$ (781 kHz)
1	0	1	$f_{XX}/32$ (391 kHz)
1	1	0	$f_{XX}/128$ (97.6 kHz)
1	1	1	$f_{XX}/512$ (24.4 kHz)

Cautions 1. If writing data different than that of PRM2 is written, stop the timer beforehand.

2. Be sure to set 0 to bits 3 to 7 of PRM2.

Remark Figures in parentheses apply to operation with f_{XX} = 12.5 MHz.

9.4 Operation

9.4.1 Operating as an interval timer (8-bit operation)

The timer operates as an interval timer that repeatedly generates interrupt requests at the interval of the preset count in the 8-bit compare register 10, 20 (CR10, CR20).

If the count in the 8-bit timer register 1, 2 (TM1, TM2) matches the value set in CR10, CR20, simultaneous to clearing the value of TM1, TM2 to 0 and continuing the count, the interrupt request signal (INTTM1, INTTM2) is generated.

The TM1 and TM2 count clocks can be selected with bit 0 to 2 (TCLn0 to TCLn2) in the prescaler mode register 1, 2 (PRM1, PRM2).

<Setting method>

<1> Set each register.
- PRMn : Selects the count clock.
- CRn0 : Compare value
- TMCn : Selects the clear and start mode when TMn and CRn0 match.
  (TMCn = 0000××××0B, × is don't care)

<2> When TCEn = 1 is set, counting starts.

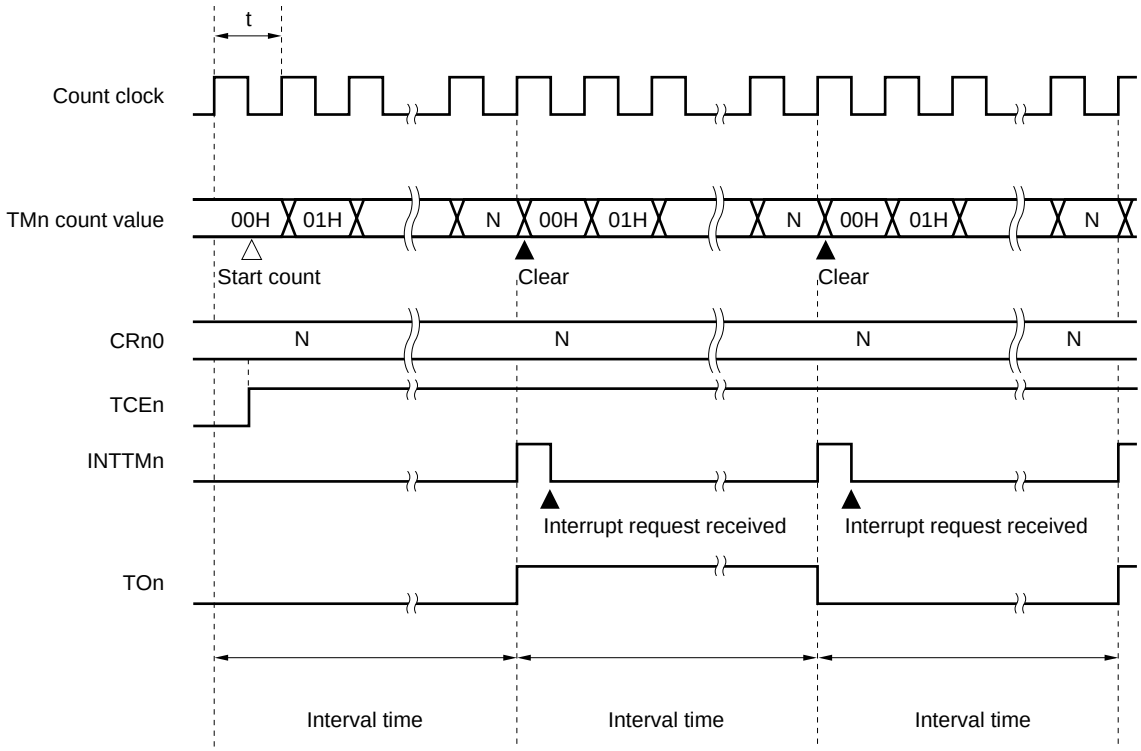
<3> When the values of TMn and CRn0 match, INTTMn is generated (TMn is cleared to 00H).

<4> Then, INTTMn is repeatedly generated during the same interval. When counting stops, set TCEn = 0.

Remark n = 1, 2

Figure 9-6. Timing of Interval Timer Operation (1/3)

(a) Basic operation

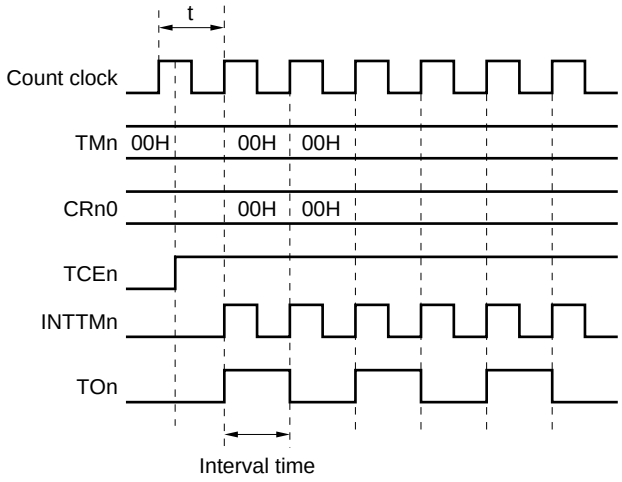


Remarks 1. Interval time = (n+1) × t: N = 00H to FFH

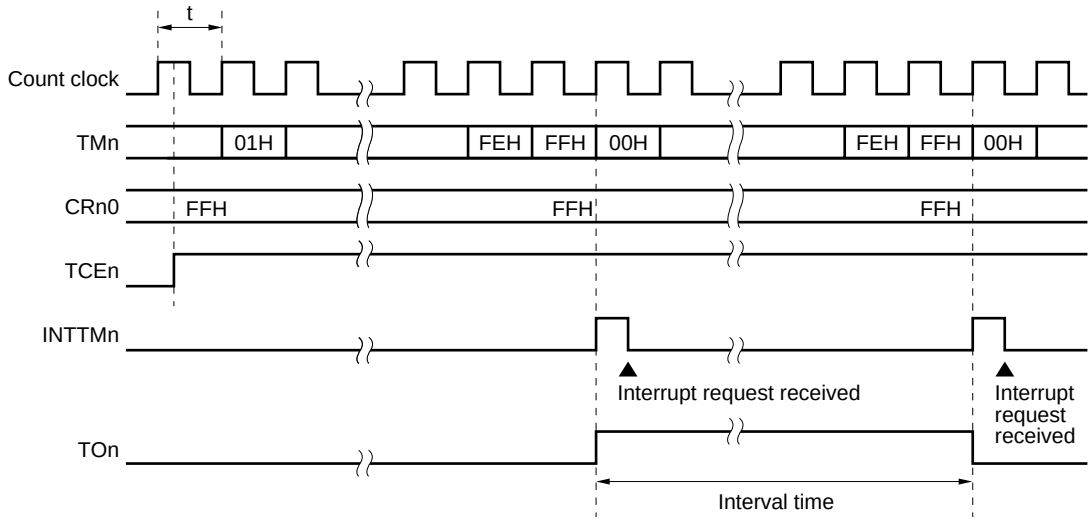
2. n = 1, 2

Figure 9-6. Timing of Interval Timer Operation (2/3)

(b) When CRn0 = 00H



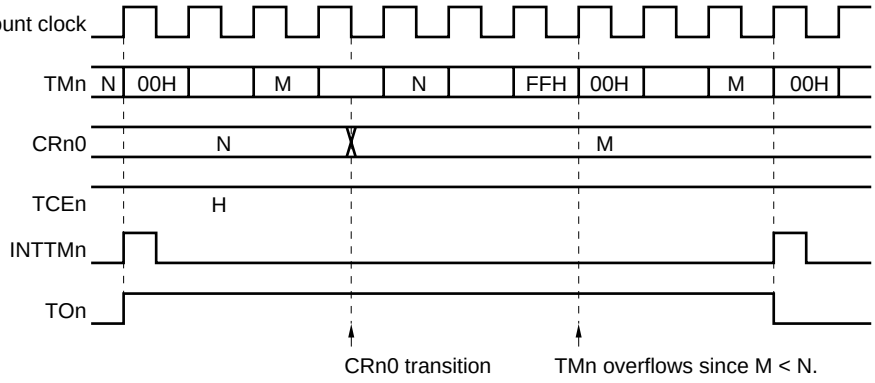
(c) When CRn0 = FFH



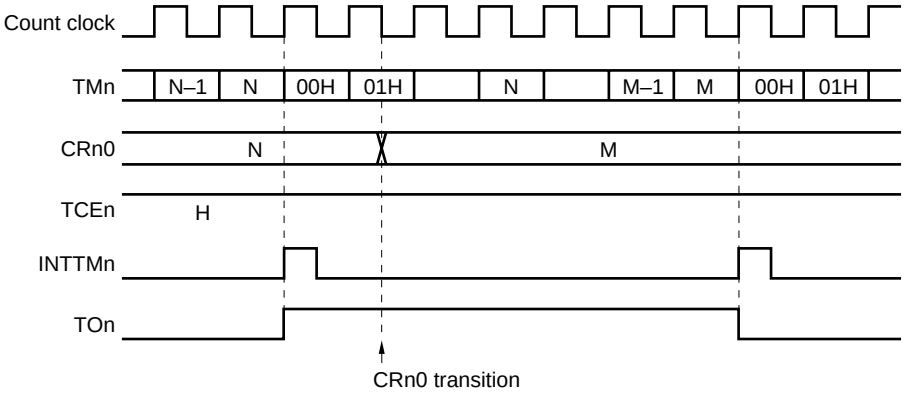
Remark n = 1, 2

Figure 9-6. Timing of Interval Timer Operation (3/3)

(d) Operated by CRn0 transition (M < N)



(e) Operated by CRn0 transition (M > N)



Remark n = 1, 2

9.4.2 Operating as an external event counter

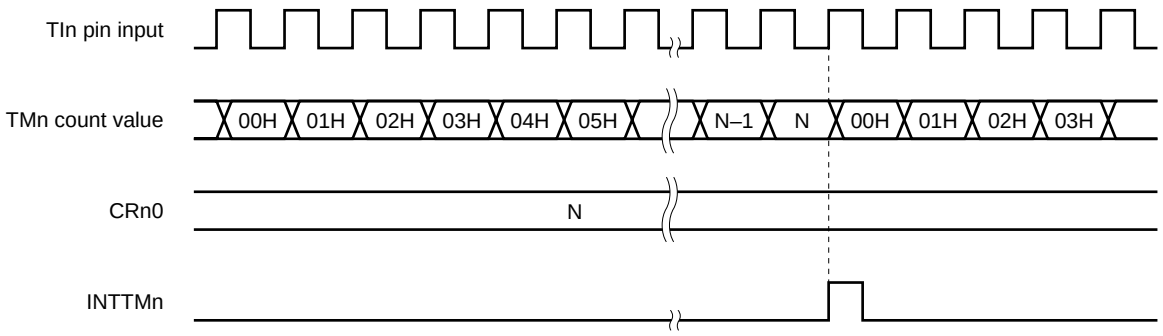
The external event counter counts the number of external clock pulses that are input to TI1/P33, TI1/P34 pins with 8-bit timer register 1, 2 (TM1, TM2).

Each time a valid edge specified in the prescaler mode register 1, 2 (PRM1, PRM2) is input, TM1and TM2 are incremented. The edge setting is selected to be either a rising edge falling edge.

If the counting of TM1and TM2 matches with the values of 8-bit compare register 10, 20 (CR10, CR20), the TM1and TM2 are cleared to 0 and the interrupt request signal (INTTM1, INTTM2) is generated.

INTTM1 and INTTM2 are generated each time when the value of the TM1 and TM2 matches with the value of CR10 and CR20.

Figure 9-7. Timing of the External Event Counter Operation (when rising edge is set)



Remark N = 00H to FFH
n = 1, 2

9.4.3 Operating as an square wave output (8-bit resolution)

A square wave having any frequency is output at the interval preset in the 8-bit compare register 10, 20 (CR10, CR20).

By setting bit 0 of the 8-bit timer mode control register 1, 2 (TMC1, TMC2) to 1, the output state of TO1, TO2 is inverted with the count preset in CR510, CR20 as the interval. Therefore, a square wave output having any frequency (duty cycle = 50 %) is possible.

<Setting method>

<1> Set the registers.

- Set the port latch and port n mode register to 0.
- PRMn : Select the count clock.
- CRn0 : Compare value
- TMCn : Clear and start mode when TMn and CRn0 match.

LVSn	LVRn	Setting State of Timer Ouput Flip-Flop
1	0	High level output
0	1	Low level output

Inversion of timer ouput flip-flop enabled
Timer output permit → TOEn = 1

<2> When TCEn = 1 is set, the counter starts operating.

<3> If the values of TMn and CRn0 match, the timer output flip-flop inverts. Also, INTTMn is generated and TMn is cleared to 00H.

<4> Then, the timer output flip-flop is inverted for the same interval to output a square wave from TOn.

Remark n = 1, 2

9.4.4 Operating as an 8-bit PWM output

By setting bit 6 (TMC16, TMC26) of the 8-bit timer mode control register 1, 2 (TMC1, TMC2) to 1, the timer operates as a PWM output.

Pulses with the duty cycle determined by the value set in the 8-bit compare register 10, 20 (CR10, CR20) is output from TO1, TO2.

Set the width of the active level of the PWM pulse in CR10, CR20. The active level can be selected by bit 1 (TMC11, TMC12) in TMC1, TMC2.

The count clock can be selected by bits 0 to 2 (TCLn0 to TCLn2) of the prescaler mode register 1, 2 (PRM1, PRM2).

The PWM output can be enabled and disabled by bit 0 (TOE1, TOE2) of TMC1, TMC2.

(1) Basic operation of the PWM output

<Setting method>

<1> Set the port latch and port mode register n to 0.
<2> Set the active level width in the 8-bit compare register n (CRn0).
<3> Select the count clock in the prescaler mode register n (PRMn).
<4> Set the active level in bit 1 (TMCn1) of TMCn.
<5> If bit 7 (TCEn) of TMCn is set to 1, counting starts. When counting stops, set TCEn to 0.

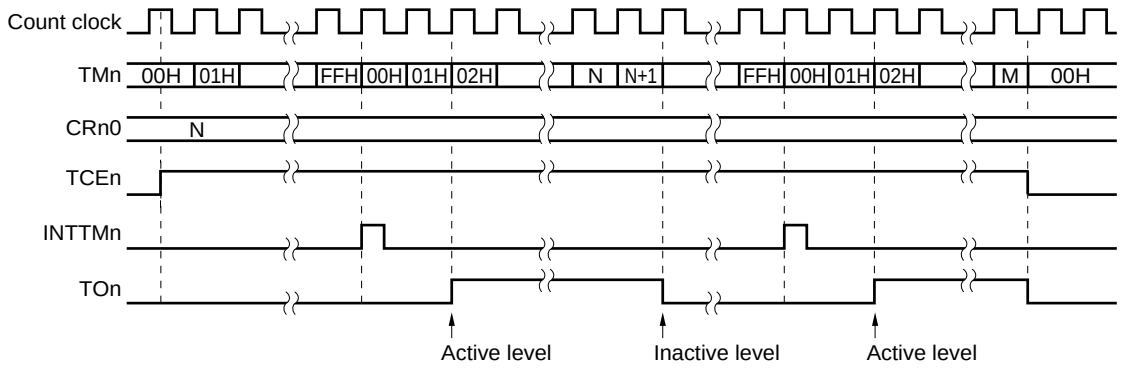
<PWM output operation>

<1> When counting starts, the PWM output (output from TOn) outputs the inactive level until an overflow occurs.
<2> When the overflow occurs, the active level specified in step <1> in the setting method is output. The active level is output until CRn0 and the count of the 8-bit counter n (TMn) match.
<3> The PWM output after CRn and the count match is the inactive level until an overflow occurs again.
<4> Steps <2> and <3> repeat until counting stops.
<5> If counting is stopped by TCEn = 0, the PWM output goes to the inactive level.

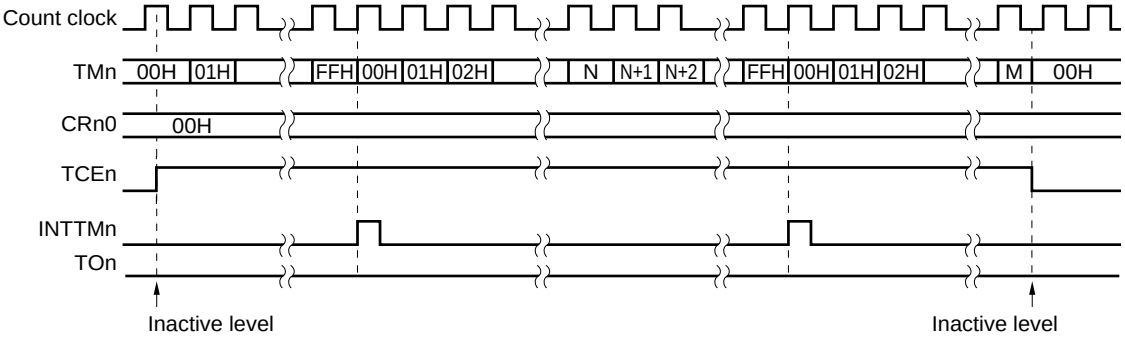
Remark n = 1, 2

Figure 9-8. Timing of the PWM Output

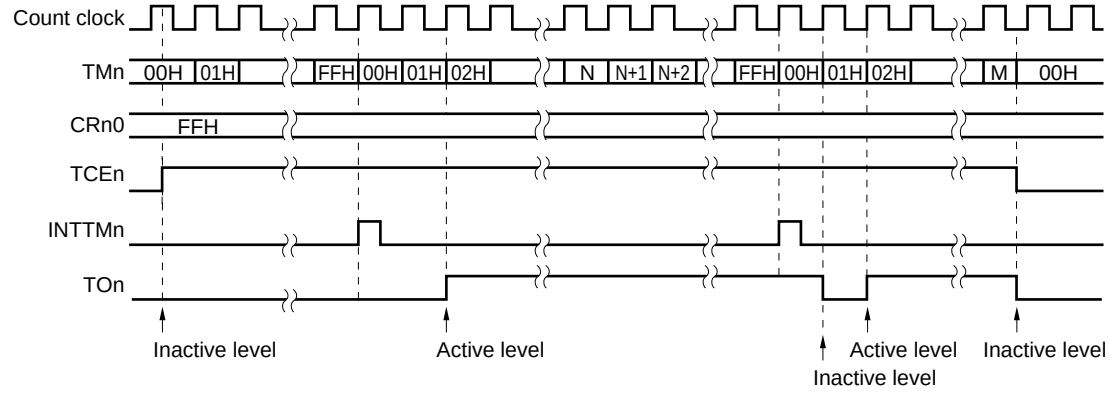
(a) Basic operation (active level = H)



(b) When CRn0 = 0



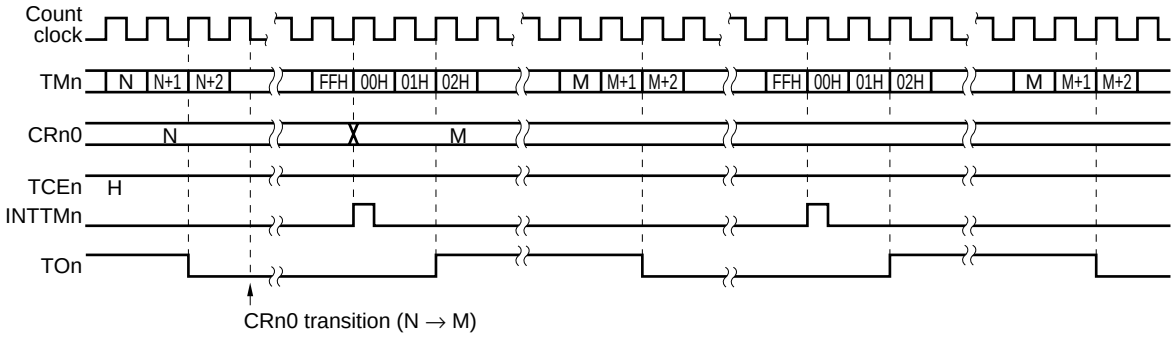
(c) When CRn0 = FFH



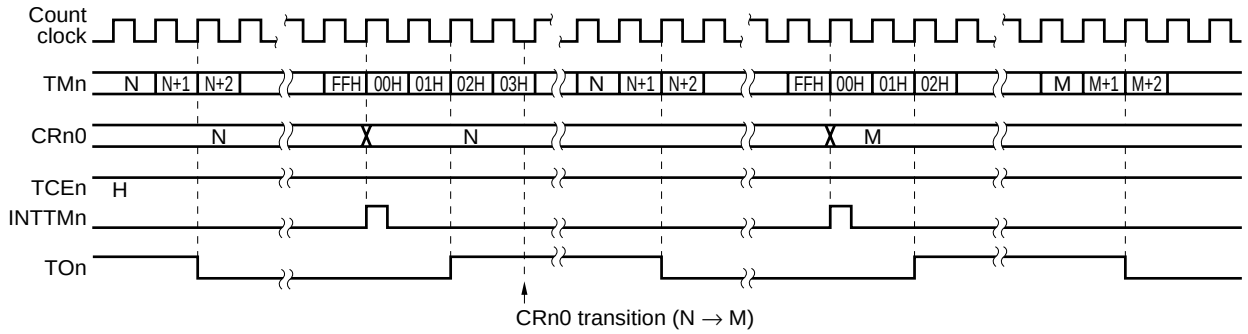
Remark n = 1, 2

Figure 9-9. Timing of Operation Based on CRn0 Transitions

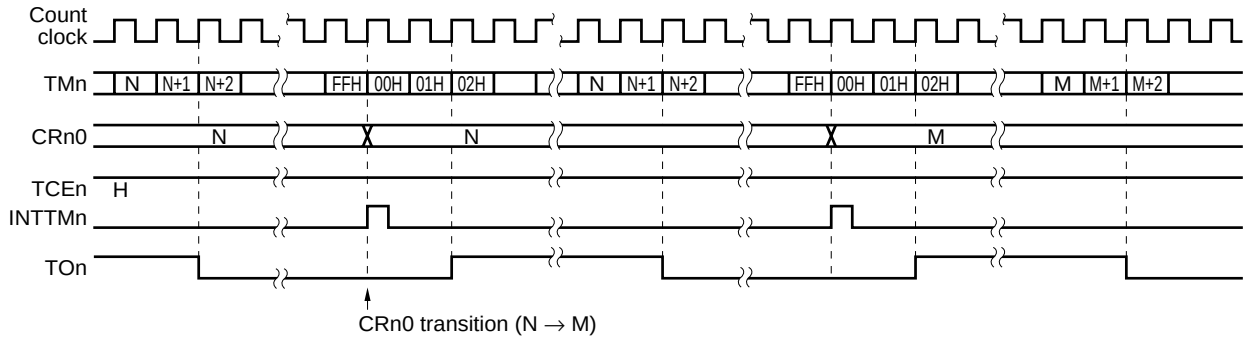
(a) When the CRn0 value from N to M before TMn overflows



(b) When the CRn0 value changes from N to M after TMn overflows



(c) When the CRn0 value changes from N to M during two clocks (00H, 01H) immediately after TMn overflows



Remark n = 1, 2

9.4.5 Operating as an interval timer (16-bit operation)

- **Cascade connection (16-bit timer) mode**
  By setting bit 4 (TMC24) of the 8-bit timer mode control register 2 (TMC2) to 1, the timer enters the timer/counter mode with 16-bit resolution.
  With the count preset in the 8-bit compare register 10, 20 (CR10, CR20) as the interval, the timer operates as an interval timer by repeatedly generating interrupt requests.

<Setting method>

<1> Set each register.
- PRM1 : TM1 selects the count clock. TM2 connected in cascade are not used in setting.
- CRn0 : Compare values (Each compare value can be set from 00H to FFH.)
- TMCn : Select the clear and start mode when TMn and CRn0 match.

  TM1 → TMC1 = 0000×××0B, ×: don't care
  TM2 → TMC2 = 0001×××0B, ×: don't care

<2> Setting TCE2 = 1 for TMC2 and finally setting TCE1 = 1 in TMC1 starts the count operation.

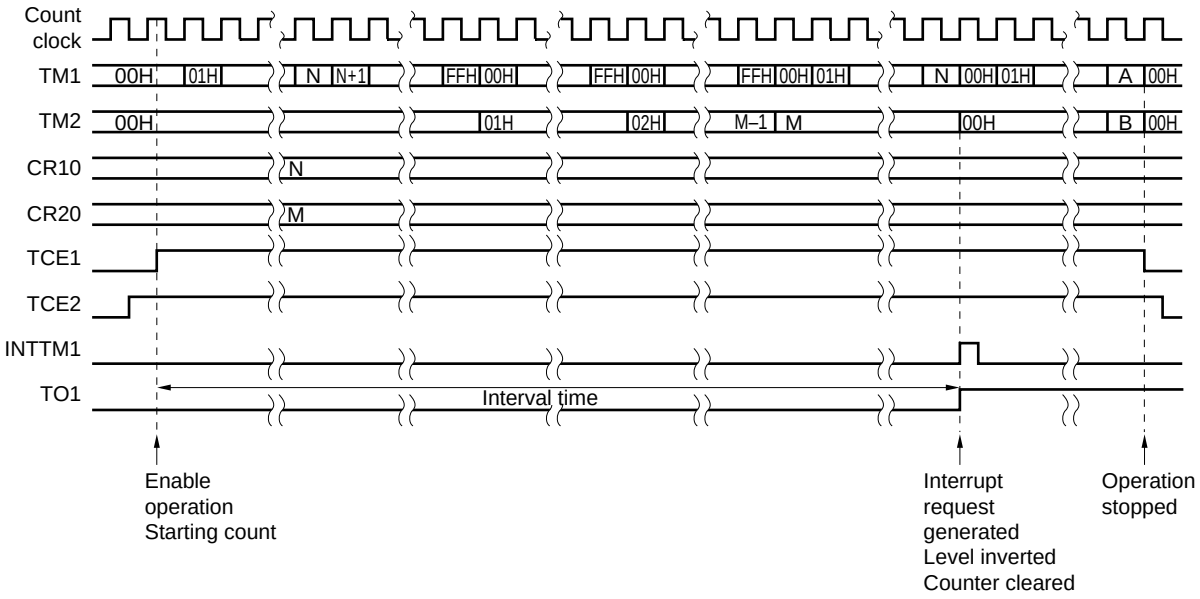
<3> If the values of TMn of all timers connected in cascade and CRn0 match, the INTTM1 of TM1 is generated. (TM1 and TM2 are cleared to 00H.)

<4> INTTM1 are repeatedly generated at the same interval.

Cautions 1. Always set the compare register (CR10, CR20) after stopping timer operation.
2. If TM2 count matches CR20 even when used in a cascade connection, INTTM2 of TM2 is generated. Always mask the high-order timer in order to disable interrupts.
3. The TCE1, TCE2 setting begins at TM2. Set theTM1 last.
4. Restarting and stopping the count is possible by setting only 1 or 0 in TCE1 of TM1 to start and stop it.

Figure 9-10 shows a timing example of the cascade connection mode with 16-bit resolution.

Figure 9-10. Cascade Connection Mode with 16-Bit Resolution

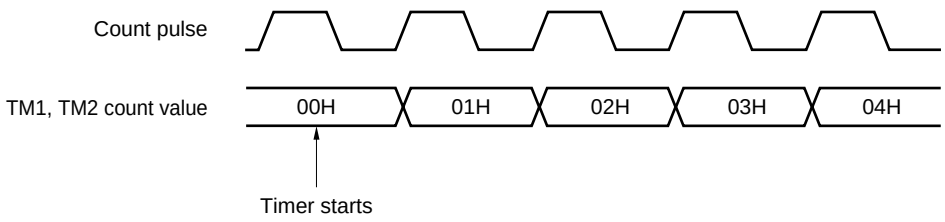


9.5. Cautions

(1) Error when the timer starts

The time until the coincidence signal is generated after the timer starts has a maximum error of one clock. The reason is the starting of the 8-bit timer register 1, 2 (TM1, TM2) is asynchronous with respect to the count pulse.

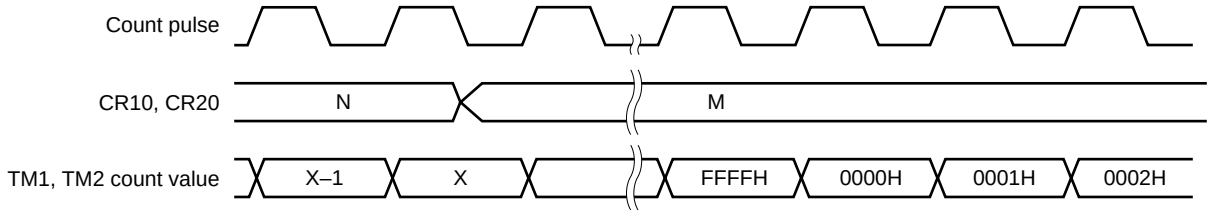
Figure 9-11. Start Timing of 8-Bit Timer Register



(2) Operation after the compare register is changed while the timer is counting

If the value after the 8-bit compare register 10, 20 (CR10, CR20) changes is less than the value of the 8-bit timer register (TM1, TM2), counting continues, overflows, and counting starts again from 0. Consequently, when the value (M) after CR10, CR20 changes is less than the value (N) before the change, the timer must restart after CR10, CR20 changes.

Figure 9-12. Timing After the Compare Register Changes During Timer Counting



Caution Except when the TI1, TI2 input is selected, always set TCE1 = 0, TCE2 = 0 before setting the STOP mode.

Remark N > X > M

(3) TM1, TM2 read out during timer operation

Since the count clock stops temporarily when TM1 and TM2 are read during operation, select for the count clock a waveform with a high and low level that exceed 2 cycles of the CPU clock.

CHAPTER 10 8-BIT TIMER/COUNTER 5, 6

10.1 Functions

8-bit timer/counter 5, 6 (TM5, TM6) have the following two modes.

- Mode using 8-bit timer/counter 5, 6 (TM5, TM6) alone (individual mode)
- Mode using the cascade connection (16-bit resolution: cascade connection mode)

These two modes are described next.

(1) Mode using 8-bit timer/counter 5, 6 alone (individual mode)

The timer operates as an 8-bit timer/event counter.
It can have the following functions.

- Interval timer
- External event counter
- Square wave output
- PWM output

(2) Mode using the cascade connection (16-bit resolution: cascade connection mode)

The timer operates as a 16-bit timer/event counter by connecting in cascade.
It can have the following functions.

- Interval timer with 16-bit resolution
- External event counter with 16-bit resolution
- Square wave output with 16-bit resolution

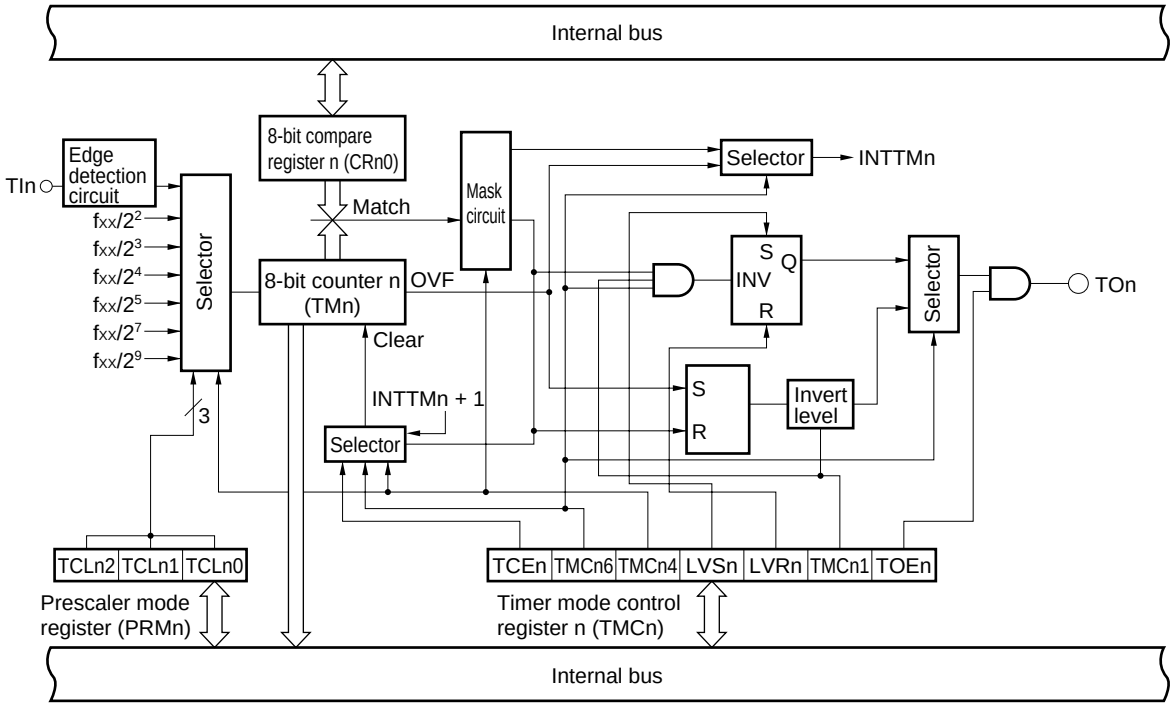
10.2 Configuration

8-bit timer/counter 5, 6 are constructed from the following hardware.

Table 10-1. 8-Bit Timer/Counter 5, 6 Configuration

Item	Configuration
Timer register	8-bit × 2 (TM5, TM6)
Register	8-bit × 2 (CR50, CR60)
Timer output	2 (TO5, TO6)
Control register	8-bit timer mode control register 5 (TMC5) 8-bit timer mode control register 6 (TMC6) Prescaler mode register 5 (PRM5) Prescaler mode register 6 (PRM6)

Figure 10-1. Block Diagram of 8-Bit Timer/Counter 5, 6



Remark n = 5, 6

(1) 8-bit timer register 5, 6 (TM5, TM6)

TM5 and TM6 are 8-bit read-only registers that counts the count pulses.

The counter is incremented synchronous to the rising edge of the count clock. When the count is read out during operation, the count clock input temporarily stops and the count is read at that time. In the following cases, the count becomes 00H.

<1> $\overline{\text{RESET}}$ is input.
<2> TCEn is cleared.
<3> TMn and CRn0 match in the clear and start mode.

Caution During the cascade connection, the count becomes 00H even when TCE5 in TM5 is cleared.

Remark n = 5, 6

(2) 8-bit compare register (CR50, CR60)

The value set in CR50 and CR60 are compared to the count in the 8-bit timer register 5 (TM5) and 8-bit timer register 6 (TM6), respectively. If the two values match, interrupt requests (INTTM5, INTTM6) is generated (except in the PWM mode).

The values of CR50 and CR60 can be set in the range of 00H to FFH, and can be written during counting.

Caution If data is set in a cascade connection, always set after stopping the timer.

10.3 Control Registers

The following four registers control 8-bit timer/counter 5, 6.

- 8-bit timer mode control register 5, 6 (TMC5, TMC6)
- Prescaler mode register 5, 6 (PRM5, PRM6)

(1) 8-bit timer mode control register 5, 6 (TMC5, TMC6)

The TMC5, TMC6 registers make the following six settings.

<1> Controls the counting for the 8-bit timer register 5, 6 (TM5, TM6).
<2> Selects the operating mode of the 8-bit timer register 5, 6 (TM5, TM6).
<3> Selects the individual mode or cascade mode.
<4> Sets the state of the timer output flip-flop.
<5> Controls the timer flip-flop or selects the active level during the PWM (free running) mode.
<6> Controls timer output.

TMC5 and TMC6 are set by a 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input sets TMC5 and TMC6 to 00H.
Figures 10-2 and 10-3 show the TMC5 format and TMC6 format respectively.

Figure 10-2. Format of the 8-Bit Timer Mode Control Register 5 (TMC5)

Address: 0FF68H After Reset: 00H R/W

Symbol	⑦	6	5	4	③	②	1	⓪
TMC5	TCE5	TMC56	0	0	LVS5	LVR5	TMC51	TOE5

TCE5	TM5 Count Control
0	Counting is disabled (prescaler disabled) after the counter is cleared to 0.
1	Start counting

TMC56	TM1 Operating Mode Selection
0	Clear and start mode whebn TM5 and CR50 match.
1	PWM (free running) mode

LVS5	LVR5	Setting the State of the Timer Output Flip-Flop
0	0	No change
0	1	Reset the timer output filp-flop (to 0).
1	0	Set the timer output flip-flop (to 1).
1	1	Setting prohibit

TMC51	Other than PWM Mode (TMC56 = 0)	PWM Mode (TMC56 = 1)
	Timer flip-flop control	Active level selection
0	Disable inversion operation	Active high
1	Enable inversion operation	Active low

TOE5	Timer Output Control
0	Disable ouput (port mode)
1	Enable ouput

Remarks 1. In the PWM mode, the PWM output is set to the inactive level by TCE5 = 0.

2. If LVS5 and LVR5 are read after setting data, 0 is read.

Figure 10-3. Format of the 8-Bit Timer Mode Control Register 6 (TMC6)

Address: 0FF69H After Reset: 00H R/W

Symbol	⑦	6	5	4	③	②	1	⓪
TMC6	TCE6	TMC66	0	TMC64	LVS6	LVR6	TMC61	TOE6

TCE6	TM6 Count Control
0	Counting is deisabled (prescaler disabled) after the counter is cleared to 0.
1	Start counting

TMC66	TM6 Operating Mode Selection
0	Clear and start mode whebn TM6 and CR60 match
1	PWM (free running) mode

TMC64	Individual Mode or Cascade Connection Mode Selection
0	Individual mode
1	Cascade connection mode (connection with TM5)

LVS6	LVR6	Setting the State of the Timer Output Flip-Flop
0	0	No change
0	1	Reset the timer output filp-flop (to 0).
1	0	Set the timer output flip-flop (to 1).
1	1	Setting prohibit

TMC61	Other than PWM Mode (TMC66 = 0)	PWM Mode (TMC66 = 1)
	Timer flip-flop control	Active level selection
0	Disable inversion operation	Active high
1	Enable inversion operation	Active low

TOE6	Timer Output Control
0	Disable ouput (port mode)
1	Enable ouput

Remarks 1. In the PWM mode, the PWM output is set to the inactive level by TCE6 = 0.

2. If LVS6 and LVR6 are read after setting data, 0 is read.

(2) Prescaler mode register 5, 6 (PRM5, PRM6)

This register sets the count clock of 8-bit timer register 5, 6 (TM5, TM6) and the effective edge of TI5, TI6 inputs.

PRM5 and PRM6 are set by an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PRM5 and PRM6 to 00H.

Figure 10-4. Format of the Prescaler Mode Register 5 (PRM5)

Address: 0FF6CH After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
PRM5	0	0	0	0	0	TCL52	TCL51	TCL50

TCL52	TCL51	TCL50	Count Clock Selection
0	0	0	Falling edge of TI5
0	0	1	Rising edge of TI5
0	1	0	$f_{XX}/4$ (3.13 MHz)
0	1	1	$f_{XX}/8$ (1.56 MHz)
1	0	0	$f_{XX}/16$ (781 kHz)
1	0	1	$f_{XX}/32$ (391 kHz)
1	1	0	$f_{XX}/128$ (97.6 kHz)
1	1	1	$f_{XX}/512$ (24.4 kHz)

Cautions 1. If writing data different than that of PRM5 is written, stop the timer beforehand.

2. Be sure to set bits 3 to 7 of PRM5.

Remark Figures in parentheses apply to operation with f_{XX} = 12.5 MHz.

Figure 10-5. Format of the Prescaler Mode Register 6 (PRM6)

Address: 0FF6DH After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
PRM6	0	0	0	0	0	TCL62	TCL61	TCL60

TCL62	TCL61	TCL60	Count Clock Selection
0	0	0	Falling edge of TI6
0	0	1	Rising edge of TI6
0	1	0	$f_{XX}/4$ (3.13 MHz)
0	1	1	$f_{XX}/8$ (1.56 MHz)
1	0	0	$f_{XX}/16$ (781 kHz)
1	0	1	$f_{XX}/32$ (391 kHz)
1	1	0	$f_{XX}/128$ (97.6 kHz)
1	1	1	$f_{XX}/512$ (24.4 kHz)

Cautions 1. If writing data different than that of PRM6 is written, stop the timer beforehand.

2. Be sure to set bits 3 to 7 of PRM6 to 0.

Remark Figures in parentheses apply to operation with f_{XX} = 12.5 MHz.

10.4 Operation

10.4.1 Operating as an interval timer (8-bit operation)

The timer operates as an interval timer that repeatedly generates interrupt requests at the interval of the preset count in the 8-bit compare register 50, 60 (CR50, CR60).

If the count in the 8-bit timer register 5, 6 (TM5, TM6) matches the value set in CR50, CR60, simultaneous to clearing the value of TM5, TM6 to 0 and continuing the count, the interrupt request signal (INTTM5, INTTM6) is generated.

The TM5 and TM6 count clocks can be selected with bit 0 to 2 (TCLn0 to TCLn2) in the prescaler mode register 5, 6 (PRM5, PRM6).

<Setting method>

<1> Set each register.
- PRMn : Selects the count clock.
- CRn0 : Compare value
- TMCn : Selects the clear and start mode when TMn and CRn0 match.
  (TMCn = 0000××××0B, × is don't care)

<2> When TCEn = 1 is set, counting starts.

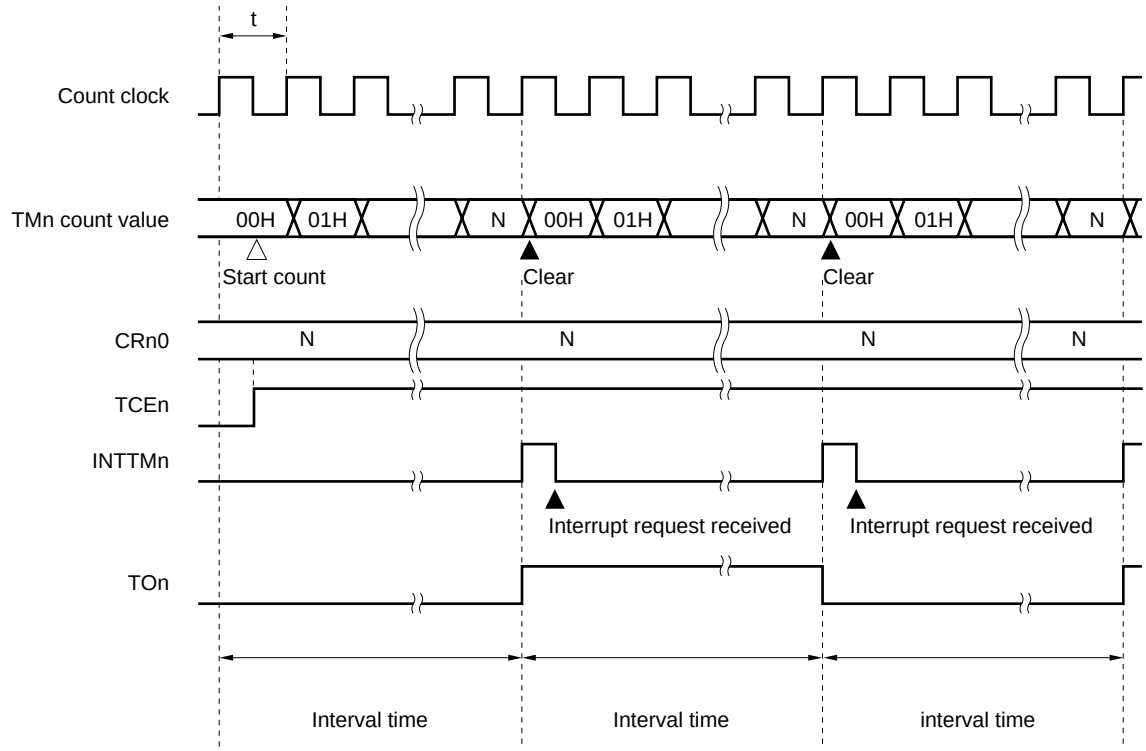
<3> When the values of TMn and CRn0 match, INTTMn is generated (TMn is cleared to 00H).

<4> Then, INTTMn is repeatedly generated during the same interval. When counting stops, set TCEn = 0.

Remark n = 5, 6

Figure 10-6. Timing of Interval Timer Operation (1/3)

(a) Basic operation

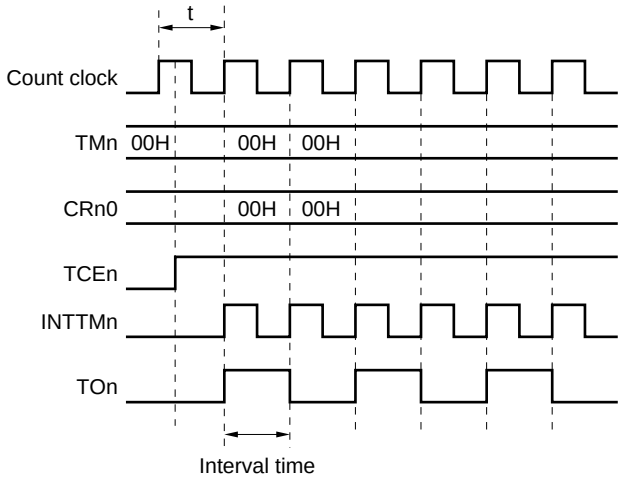


Remarks 1. Interval time = (N+1) × t; N = 00H to FFH

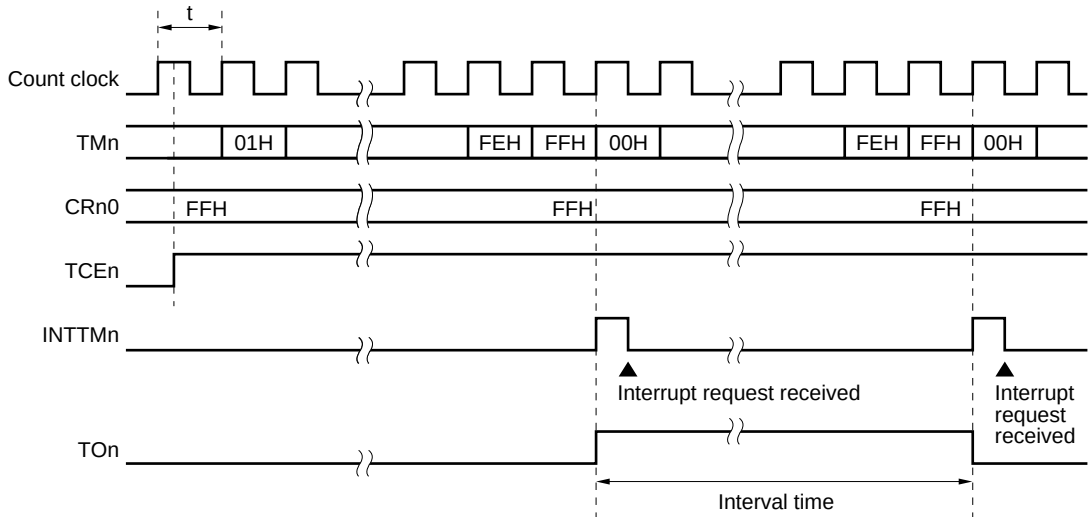
2. n = 5, 6

Figure 10-6. Timing of Interval Timer Operation (2/3)

(b) When CRn0 = 00H



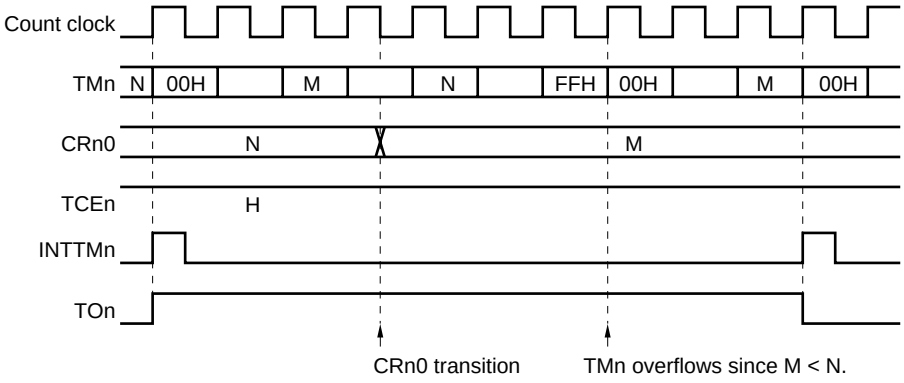
(c) When CRn0 = FFH



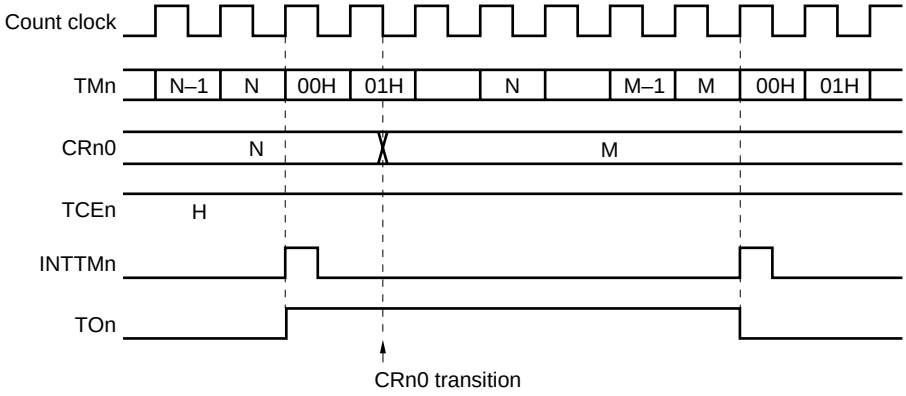
Remark n = 5, 6

Figure 10-6. Timing of Interval Timer Operation (3/3)

(d) Operated by CRn0 transition (M < N)



(e) Operated by CRn0 transition (M > N)



Remark n = 5, 6

10.4.2 Operating as an external event counter

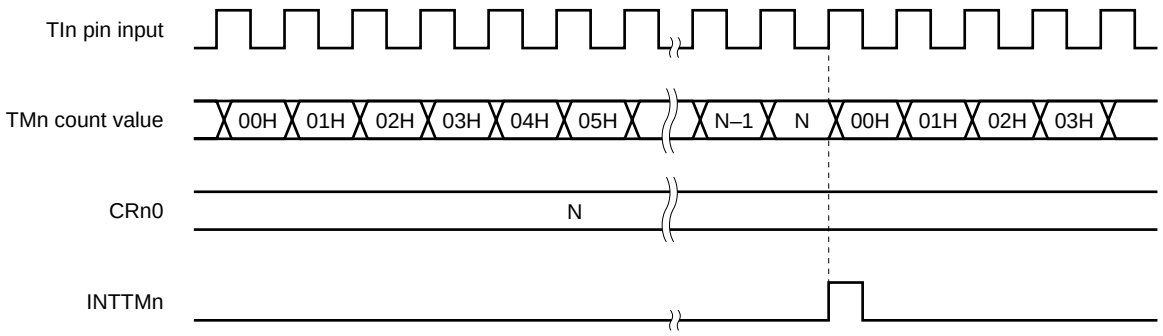
The external event counter counts the number of external clock pulses that are input to TI5/P100, TI5/P101 pins with 8-bit timer register 5, 6 (TM5, TM6).

Each time a valid edge specified in the prescaler mode register 5, 6 (PRM5, PRM6) is input, TM5 and TM6 are incremented. The edge setting is selected to be either a rising edge falling edge.

If the counting of TM5 and TM6 matches with the values of 8-bit compare register 50, 60 (CR50, CR60), the TM5 and TM6 are cleared to 0 and the interrupt request signal (INTTM5, INTTM6) is generated.

INTTM5 and INTTM6 are generated each time when the value of the TM5 and TM6 matches with the value of CR50 and CR60.

Figure 10-7. Timing of the External Event Counter Operation (when rising edge is set)



Remark N = 00H to FFH
n = 5, 6

10.4.3 Operating as an square wave output (8-bit resolution)

A square wave having any frequency is output at the interval preset in the 8-bit compare register 50, 60 (CR50, CR60).

By setting bit 0 (TOE5, TOE6) of the 8-bit timer mode control register 5, 6 (TMC5, TMC6) to 1, the output state of TO5, TO6 is inverted with the count preset in CR50, CR60 as the interval. Therefore, a square wave output having any frequency (duty cycle = 50 %) is possible.

<Setting method>

<1> Set the registers.

- Set the port latch and port n mode register to 0.
- PRMn : Select the count clock.
- CRn0 : Compare value
- TMCn : Clear and start mode when TMn and CRn0 match.

LVSn	LVRn	Setting State of Timer Ouput Flip-Flop
1	0	High level output
0	1	Low level output

Inversion of timer ouput flip-flop enabled
Timer output enabled → TOEn = 1

<2> When TCEn = 1 is set, the counter starts operating.

<3> If the values of TMn and CRn0 match, the timer output flip-flop inverts. Also, INTTMn is generated and TMn is cleared to 00H.

<4> Then, the timer output flip-flop is inverted for the same interval to output a square wave from TOn.

Remark n = 5, 6

10.4.4 Operating as an 8-bit PWM output

By setting bit 6 (TMC56, TMC66) of the 8-bit timer mode control register 5, 6 (TMC5, TMC6) to 1, the timer operates as a PWM output.

Pulses with the duty cycle determined by the value set in the 8-bit compare register 50, 60 (CR50, CR60) is output from TO5, TO6.

Set the width of the active level of the PWM pulse in CR50, CR60. The active level can be selected by bit 1 (TMC51, TMC61) in TMC5, TMC6.

The count clock can be selected by bits 0 to 2 (TCLn0 to TCLn2) of the prescaler mode register 5, 6 (PRM5, PRM6).

The PWM output can be enabled and disabled by bit 0 (TOE5, TOE6) of TMC5, TMC6.

(1) Basic operation of the PWM output

<Setting method>

<1> Set the port latch and port mode register n to 0.
<2> Set the active level width in the 8-bit compare register n (CRn0).
<3> Select the count clock in the prescaler mode register n (PRMn).
<4> Set the active level in bit 1 (TMCn1) of TMCn.
<5> If bit 7 (TCEn) of TMCn is set to 1, counting starts. When counting stops, set TCEn to 0.

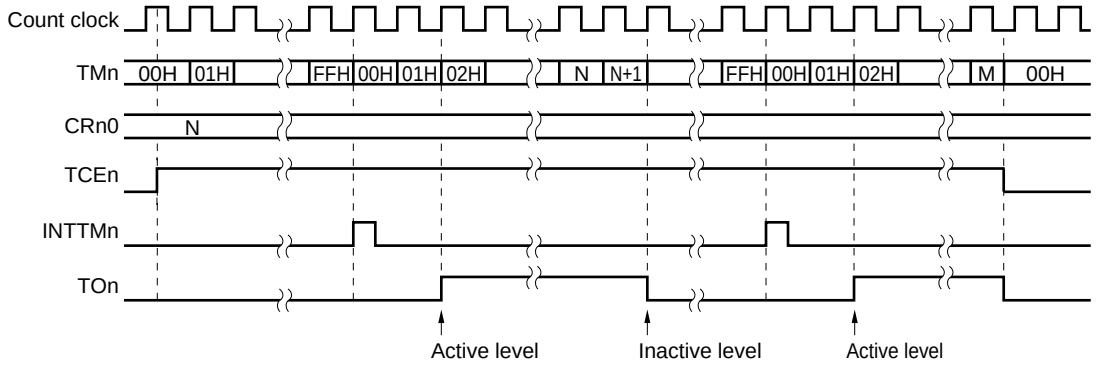
<PWM output operation>

<1> When counting starts, the PWM output (output from TOn) outputs the inactive level until an overflow occurs.
<2> When the overflow occurs, the active level specified in step <1> in the setting method is output. The active level is output until CRn0 and the count of the 8-bit counter n (TMn) match.
<3> The PWM output after CRn and the count match is the inactive level until an overflow occurs again.
<4> Steps <2> and <3> repeat until counting stops.
<5> If counting is stopped by TCEn = 0, the PWM output goes to the inactive level.

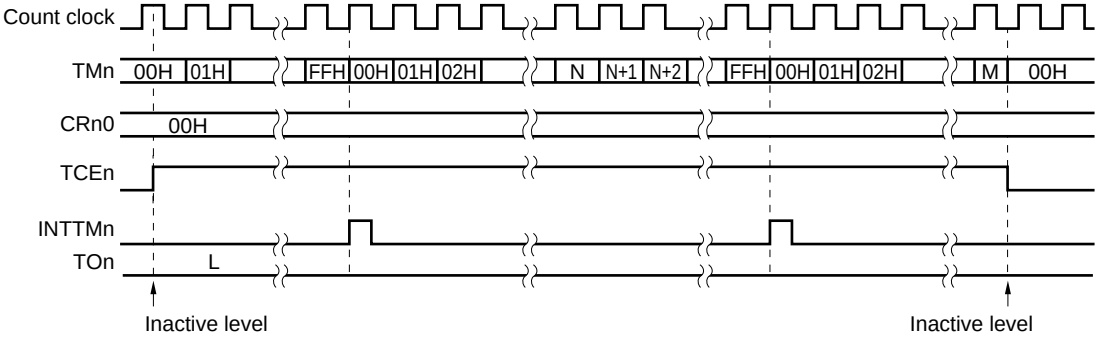
Remark n = 5, 6

Figure 10-8. Timing of the PWM Output

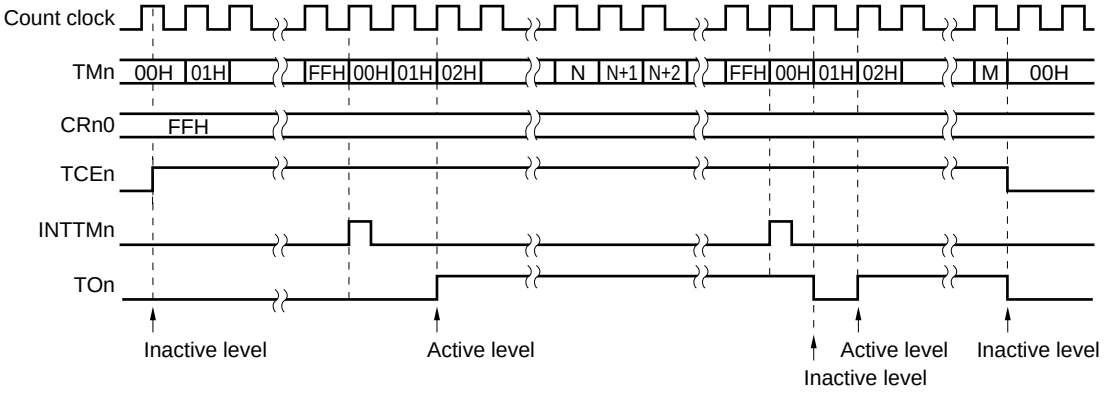
(a) Basic operation (active level = H)



(b) When CRn0 = 0



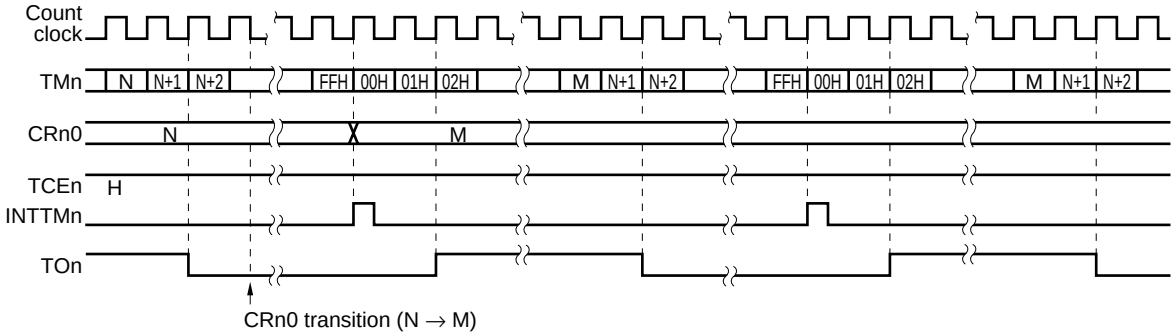
(c) When CRn0 = FFH



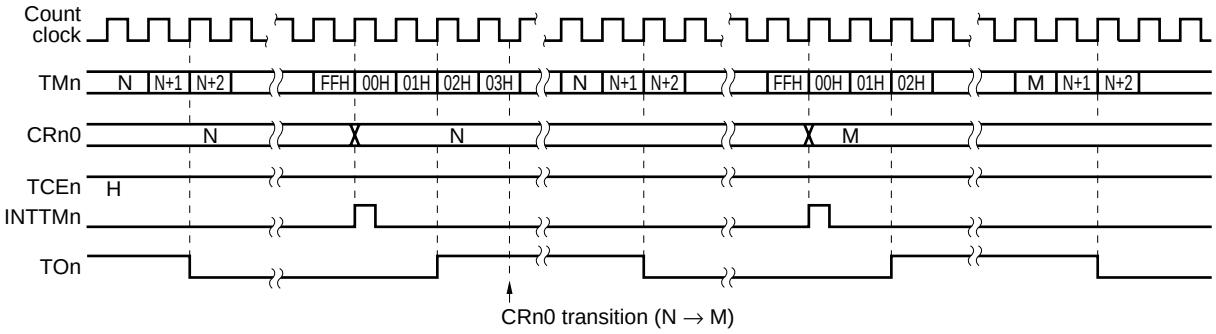
Remark n = 5, 6

Figure 10-9. Timing of Operation Based on CRn0 Transitions

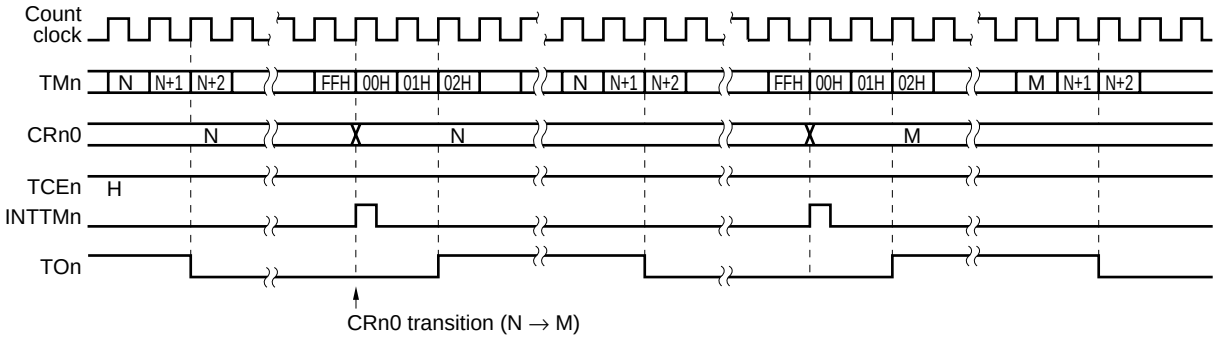
(a) When the CRn0 value from N to M before TMn overflows



(b) When the CRn0 value changes from N to M after TMn overflows



(c) When the CRn0 value changes from N to M during two clocks (00H, 01H) immediately after TMn overflows



Remark n = 5, 6

10.4.5 Operating as an interval timer (16-bit operation)

- **Cascade connection (16-bit timer) mode**
  By setting bit 4 (TMC64) of the 8-bit timer mode control register 6 (TMC6) to 1, the timer enters the timer/counter mode with 16-bit resolution.
  With the count preset in the 8-bit compare register 50, 60 (CR50, CR60) as the interval, the timer operates as an interval timer by repeatedly generating interrupt requests.

<Setting method>

<1> Set each register.
- PRM5 : TM5 selects the count clock. TM6 connected in cascade are not used in setting.
- CRn0 : Compare values (Each compare value can be set from 00H to FFH.)
- TMCn : Select the clear and start mode when TMn and CRn0 match.

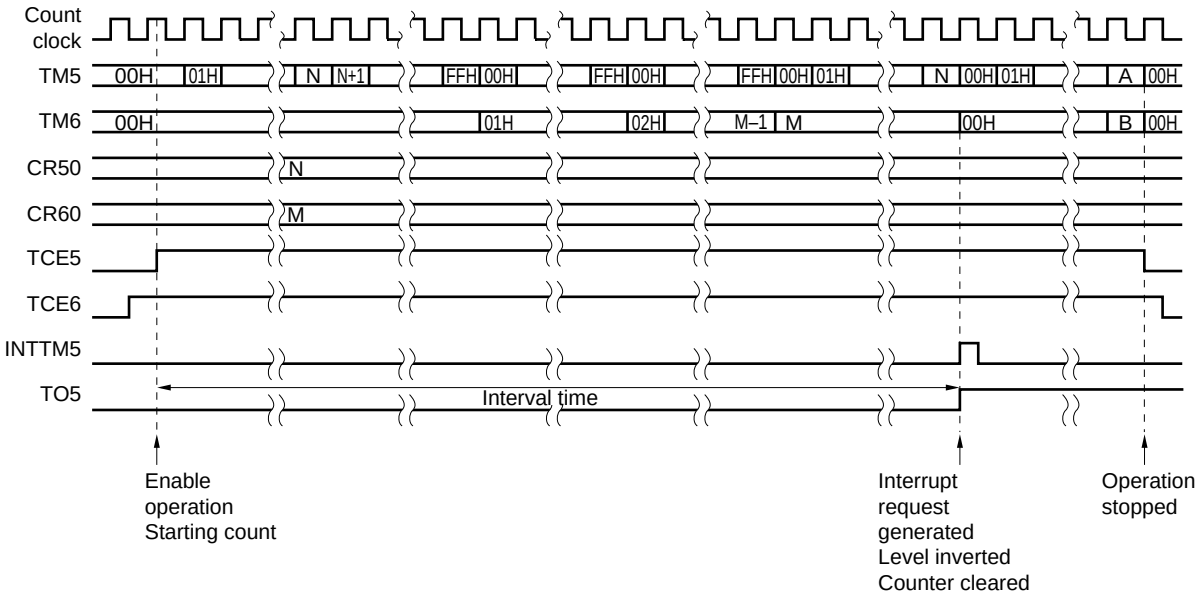
  TM5 → TMC5 = 0000×××0B, ×: don't care
  TM6 → TMC6 = 0001×××0B, ×: don't care

<2> Setting TCE6 = 1 for TMC6 and finally setting TCE5 = 1 in TMC5 starts the count operation.
<3> If the values of TMn of all timers connected in cascade and CRn0 match, the INTTM5 of TM5 is generated. (TM5 and TM6 are cleared to 00H.)
<4> INTTM5 are repeatedly generated at the same interval.

Cautions 1. Always set the compare register (CR50, CR60) after stopping timer operation.
2. If TM6 count matches CR60 even when used in a cascade connection, INTTM6 of TM6 is generated. Always mask TM6 in order to disable interrupts.
3. The TCE5, TCE6 setting begins at TM6. Set theTM5 last.
4. Restarting and stopping the count is possible by setting only 1 or 0 in TCE5 of TM5 to start and stop it.

Figure 10-10 shows a timing example of the cascade connection mode with 16-bit resolution.

Figure 10-10. Cascade Connection Mode with 16-Bit Resolution

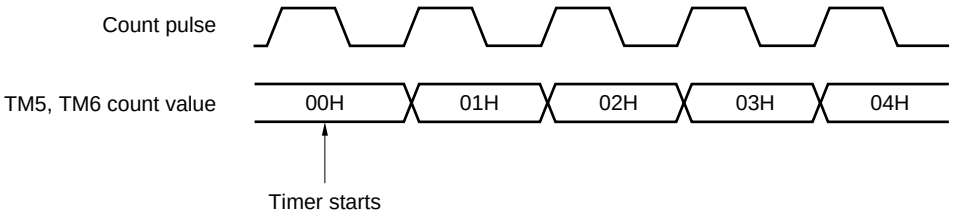


10.5 Cautions

(1) Error when the timer starts

The time until the coincidence signal is generated after the timer starts has a maximum error of one clock. The reason is the starting of the 8-bit timer register 5, 6 (TM5, TM6) is asynchronous with respect to the count pulse.

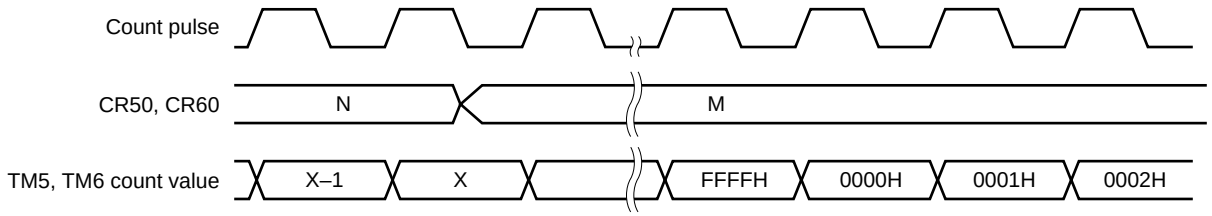
Figure 10-11. Start Timing of 8-Bit Timer Register



(2) Operation after the compare register is changed while the timer is counting

If the value after the 8-bit compare register 50, 60 (CR50, CR60) changes is less than the value of the 8-bit timer register (TM5, TM6), counting continues, overflows, and counting starts again from 0. Consequently, when the value (M) after CR50, CR60 changes is less than the value (N) before the change, the timer must restart after CR50, CR60 changes.

Figure 10-12. Timing After the Compare Register Changes During Timer Counting



Remark N > X > M

Caution Except when the TI5, TI6 input is selected, always set TCE5 = 0, TCE6 = 0 before setting the STOP mode.

(3) TM5, TM6 read out during timer operation

Since reading out TM5, TM6 during operation occurs while the selected clock is temporarily stopped, select some high or low level waveform that is longer than the selected clock.

CHAPTER 11 WATCH TIMER

11.1 Function

The watch timer has the following functions:

- Watch timer
- Interval timer

The watch timer and interval timer functions can be used at the same time.

(1) Watch timer

The watch timer generates an interrupt request (INTWT) at time intervals of 0.5 seconds by using the main system clock of 4.19 MHz or subsystem clock of 32.768 kHz.

Caution The time interval of 0.5 seconds cannot be created with the 12.5-MHz main system clock. Use the 32.768-kHz subsystem clock to create the 0.5-second time interval.

(2) Interval timer

The watch timer generates an interrupt request (INTTM3) at time intervals specified in advance.

Table 11-1. Interval Time of Interval Timer

Interval Time	f_X = 12.5 MHz	f_{XT} = 4.19 MHz	f_{XT} = 32.768 kHz
$2^{11} \times 1/f_X$	164 μs	488 μs	488 μs
$2^{12} \times 1/f_X$	328 μs	977 μs	977 μs
$2^{13} \times 1/f_X$	655 μs	1.95 ms	1.95 ms
$2^{14} \times 1/f_X$	1.31 ms	3.91 ms	3.91 ms
$2^{15} \times 1/f_X$	2.62ms	7.81 ms	7.81 ms
$2^{16} \times 1/f_X$	5.24 ms	15.6 ms	15.6 ms

Remark f_X : Main system clock oscillation frequency
f_{XT} : Subsystem clock oscillation frequency

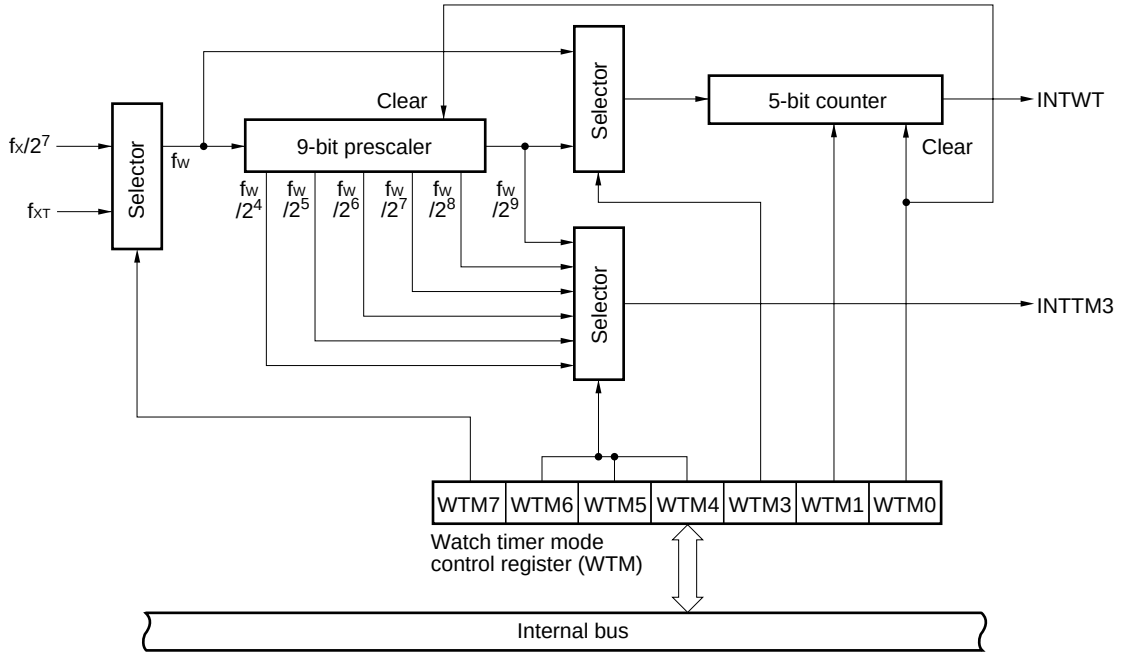
11.2 Configuration

The watch timer consists of the following hardware.

Table 11-2. Configuration of Watch Timer

Item	Configuration
Counter	5 bits × 1
Prescaler	9 bits × 1
Control register	Watch timer mode control register (WTM)

Figure 11-1. Block Diagram of Watch Timer



Remark f_X : Main system clock oscillation frequency
f_{XT} : Subsystem clock oscillation frequency

11.3 Watch Timer Control Register

- **Watch timer mode control register (WTM)**
  This register enables or disables the count clock and operation of the watch timer, sets the interval time of the prescaler, controls the operation of the 5-bit counter, and sets the set time of the watch flag.
  WTM is set by a 1-bit or 8-bit memory manipulation instruction.
  $\overline{\text{RESET}}$ input sets WTM to 00H.

Figure 11-2. Format of Watch Timer Mode Control Register (WTM)

Address: 0FF9CH After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
WTM	WTM7	WTM6	WTM5	WTM4	WTM3	0	WTM1	WTM0

WTM7	Selects Count Clock of Watch Timer
0	Main system clock ($f_X/2^7$)
1	Subsystem clock (f_{XT})

WTM6	WTM5	WTM4	Selects Interval Time of Prescaler
0	0	0	$2^4/f_W$ (488 μs)
0	0	1	$2^5/f_W$ (977 μs)
0	1	0	$2^6/f_W$ (1.95 ms)
0	1	1	$2^7/f_W$ (3.91 ms)
1	0	0	$2^8/f_W$ (7.81 ms)
1	0	1	$2^9/f_W$ (15.6 ms)
Others			Setting prohibited

WTM3	Selects Set Time of Watch Flag
0	$2^{14}/f_W$ (0.5 s)
1	$2^5/f_W$ (977 μs)

WTM1	Controls Operation of 5-Bit Counter
0	Clear after operation stop
1	Start

WTM0	Controls Operation of 5-Bit Counter
0	Operation stop (clear both prescaler and timer)
1	Operation enable

Remarks 1. f_W : Watch timer clock frequency ($f_X/2^7$ or f_{XT})

f_X : Main system clock oscillation frequency

f_{XT} : Subsystem clock oscillation frequency

2. Figures in parentheses apply to operation with f_W = 32.768 kHz.

11.4 Operation

11.4.1 Operation as watch timer

The watch timer operates with time intervals of 0.5 seconds with the main system clock (4.19 MHz) or subsystem clock (32.768 kHz).

The watch timer generates an interrupt request (INTWT) at fixed time intervals.

The count operation of the watch timer is started when bits 0 (WTM0) and 1 (WTM1) of the watch timer mode control register (WTM) are set to 1. When these bits are cleared to 0, the 5-bit counter is cleared, and the watch timer stops the count operation.

When the interval timer function is started at the same time, the watch timer can be started from 0 second by resetting WTM1 to 0. However, an error of up to 0.5 seconds may occur when the watch timer overflows (INTWT).

11.4.2 Operation as interval timer

The watch timer can also be used as an interval timer that repeatedly generates an interrupt request (INTTM3) at intervals specified by a count value set in advance.

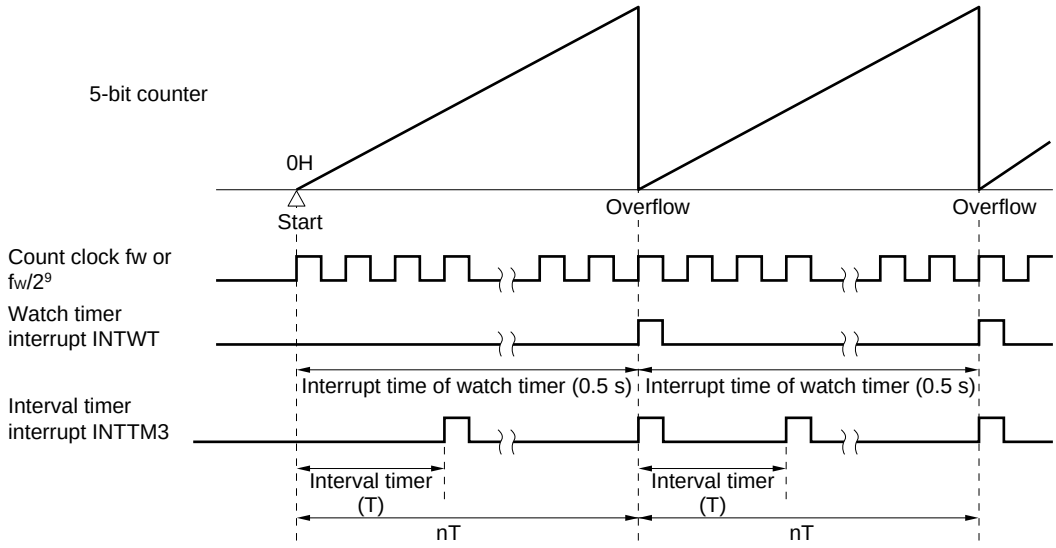
The interval time can be selected by bits 4 through 6 (WTM4 through WTM6) of the watch timer mode control register (WTM).

Table 11-3. Interval Time of Interval Timer

WTM6	WTM5	WTM4	Interval Time	f_X = 12.5 MHz	f_X = 4.19 MHz	f_{XT} = 32.768 kHz
0	0	0	$2^4 \times 1/f_W$	164 μs	488 μs	488 μs
0	0	1	$2^5 \times 1/f_W$	328 μs	977 μs	977 μs
0	1	0	$2^6 \times 1/f_W$	655 μs	1.95 ms	1.95 ms
0	1	1	$2^7 \times 1/f_W$	1.31 ms	3.91 ms	3.91 ms
1	0	0	$2^8 \times 1/f_W$	2.62 ms	7.81 ms	7.81 ms
1	0	1	$2^9 \times 1/f_W$	5.24 ms	15.6 ms	15.6 ms
Others			Setting prohibited			

Remark f_W : Watch timer clock frequency ($f_{XX}/2^7$ or f_{XT})
f_X : Main system clock oscillation frequency
f_{XT} : Subsystem clock oscillation frequency

Figure 11-3. Operation Timing of Watch Timer/Interval Timer



Remarks **1.** f_W : Watch timer clock frequency

2. ( ) : f_W = 32.768 kHz

3. n : Number of interval timer operations

CHAPTER 12 WATCHDOG TIMER

The watchdog timer detects runaway programs.

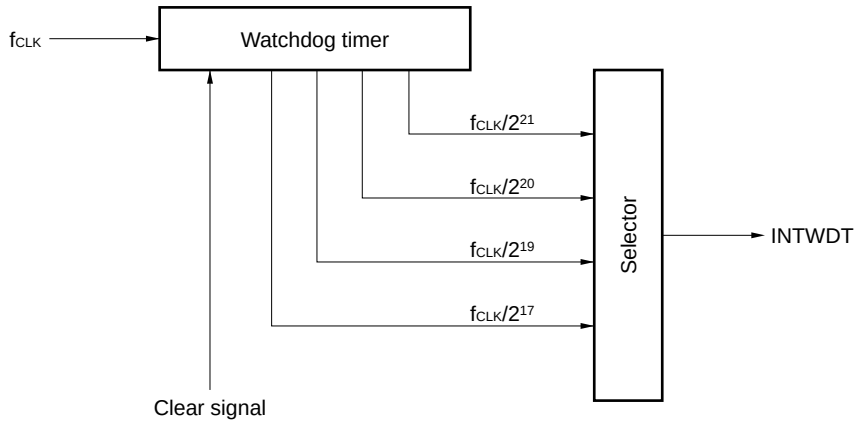
Program or system errors are detected by the generation of watchdog timer interrupts. Therefore, at each location in the program, the instruction that clears the watchdog timer (starts the count) within a constant time is input.

If the watchdog timer overflows without executing the instruction that clears the watchdog timer within the set period, a watchdog timer interrupt (INTWDT) is generated to signal a program error.

12.1 Structure

Figure 12-1 is a block diagram of the watchdog timer.

Figure 12-1. Watchdog Timer Block Diagram



Caution When the subsystem clock is selected as the internal system clock (f_{CLK}), the watchdog timer stops operating.

Remark f_{CLK}: Internal system clock (f_{XX} to $f_{XX}/8$)

12.2 Control Register

- **Watchdog timer mode register (WDM)**

  The WDM is the 8-bit register that controls watchdog timer operation.

  To prevent the watchdog timer from erroneously clearing this register due to a runaway program, this register is only written by a special instruction. This special instruction has a special code format (4 bytes) in the MOV WDM #byte instruction. Writing takes place only when the third and fourth op codes are mutual 1's complements. If the third and fourth op codes are not mutual 1's complements and not written, the operand error interrupt is generated. In this case, the return address saved in the stack is the address of the instruction that caused the error. Therefore, the address that caused the error can be identified from the return address saved in the stack. If returning by simply using the RETB instruction from the operand error, an infinite loop results.

  Since an operand error interrupt is generated only when the program is running wild (the correct special instruction is only generated when MOV WDM #byte is described in the RA78K4 NEC assembler), make the program initialize the system.

  Other write instructions (MOV WDM, A; AND WDM, #byte; SET1 WDM, etc.) are ignored and nothing happens. In other words, WDM is not written, and interrupts, such as operand error interrupts, are not generated.

  After a system reset ($\overline{\text{RESET}}$ input), when the watchdog timer starts (when the RUN bit is set to one), the WDM contents cannot change. Only a reset can stop the watchdog timer. The watchdog timer can be cleared by a special instruction.

  The WDM can be read by 8-bit data transfer instructions.

  $\overline{\text{RESET}}$ input sets WDM to 00H.

  Figure 12-2 shows the WDM format.

Figure 12-2. Watchdog Timer Mode Register (WDM) Format

Address: 0FFC2H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
WDM	RUN	0	0	WDT4	0	WDT2	WDT1	0

RUN	Watchdog Timer Operation Setting
0	Stops the watchdog timer.
1	Clears the watchdog timer and starts counting.

WDT4	Watchdog Timer Interrupt Request Priority
0	Watchdog timer interrupt request <NMI pin input interrupt request
1	Watchdog timer interrupt request >NMI pin input interrupt request

WDT2	WDT1	Count Clock	Overflow Time [ms] (f_{CLK} = 12.5 MHz)
0	0	$f_{CLK}/2^{17}$	10.5
0	1	$f_{CLK}/2^{19}$	41.9
1	0	$f_{CLK}/2^{20}$	83.9
1	1	$f_{CLK}/2^{21}$	167.8

Cautions 1. Only the special instruction (MOV WDM, #byte) can write to the watchdog timer mode register (WDM).

2. When writing to WDM to set the RUN bit to 1, write the same value every time. Even if different values are written, the contents written the first time cannot be updated.

3. When the RUN bit is set to 1, it cannot be reset to 0 by the software.

Remark f_{CLK}: Internal system clock (f_{XX} to $f_{XX}/8$)

f_{XX} : Main system clock frequency

12.3 Operations

12.3.1 Count operation

The watchdog timer is cleared by setting the RUN bit of the watchdog timer mode register (WDM) to 1 to start counting. After the RUN bit is set to 1, when the overflow time set by bits WDT2 and WDT1 in WDM has elapsed, a nonmaskable interrupt (INTWDT) is generated.

If the RUN bit is reset to 1 before the overflow time elapses, the watchdog timer is cleared, and counting restarts.

12.3.2 Interrupt priority order

The watchdog timer interrupt (INTWDT) is a nonmaskable interrupt. In addition to the INTWDT, the nonmaskable interrupts include the interrupt (NMI) from the NMI pin. By setting bit 4 of the watchdog timer mode register (WDM), the acceptance order when INTWDT and NMI are simultaneously generated can be set.

If accepting NMI is given priority, even if INTWDT is generated in an NMI processing program that is executing, INTWDT is not accepted, but is accepted after the NMI processing program ends.

12.4 Cautions

12.4.1 General cautions when using the watchdog timer

(1) The watchdog timer is one way to detect runaway operation, but all runaway operations cannot be detected. Therefore, in a device that particularly demands reliability, the runaway operation must be detected early not only by the on-chip watchdog timer but by an externally attached circuit; and when returning to the normal state or while in the stable state, processing like stopping the operation must be possible.

(2) The watchdog timer cannot detect runaway operation in the following cases.

<1> When the watchdog timer is cleared in a timer interrupt servicing program
<2> When there are successive temporary stores of interrupt requests and macro services (see **22.9 When Interrupt Requests and Macro Service are Temporarily Held Pending**)
<3> When runaway operation is caused by logical errors in the program (when each module in the program operates normally, but the entire system does not operate properly), and when the watchdog timer is periodically cleared
<4> When the watchdog timer is periodically cleared by an instruction group that is executed during runaway operation
<5> When the STOP mode and HALT mode or IDLE mode is the result of runaway operation
<6> When the watchdog timer also runs wild when the CPU runs wild because of introduced noise

In cases <1>, <2>, and <3>, detection becomes possible by correcting the program.
In case <4>, the watchdog timer can be cleared only by the 4-byte special instruction. Similarly in <5>, if there is no 4-byte special instruction, the STOP mode and HALT mode or IDLE mode cannot be set. Since the result of the runaway operation is to enter state <2>, three or more bytes of consecutive data must be a specific pattern (example, BT PSWL.bit, $$). Therefore, the results of <4>, <5>, and the runaway operation are believed to very rarely enter state <2>.

12.4.2 Cautions about the μPD784225 Subseries watchdog timer

(1) Only the special instruction (MOV WDM, #byte) can write to watchdog timer mode register (WDM).

(2) If the RUN bit is set to 1 by writing to the watchdog timer mode register (WDM), write the same value every time. Even when different values are written, the contents written the first time cannot be changed.

(3) If the RUN bit is set to 1, it cannot be reset to 0 by the software.

CHAPTER 13 A/D CONVERTER

13.1 Functions

The A/D converter converts analog inputs to digital values, and is configured by eight 8-bit resolution channels (ANI0 to ANI7).

Successive approximation is used as the conversion method, and conversion results are saved in the 8-bit A/D conversion result register (ADCR).

A/D conversion can be begun by the following two methods.

(1) Hardware start

Conversion is started by trigger input (P03) (rising edge, falling edge, or both rising and falling edges can be specified).

(2) Software start

Conversion is started by setting the A/D converter mode register (ADM).

Select one channel for analog input from ANI0 to ANI7, and perform A/D conversion. If hardware start is used, A/D conversion stops at the end of the A/D conversion operation. If software start is used, the A/D conversion operation is repeated. Each time one A/D conversion is completed, an interrupt request (INTAD) is issued.

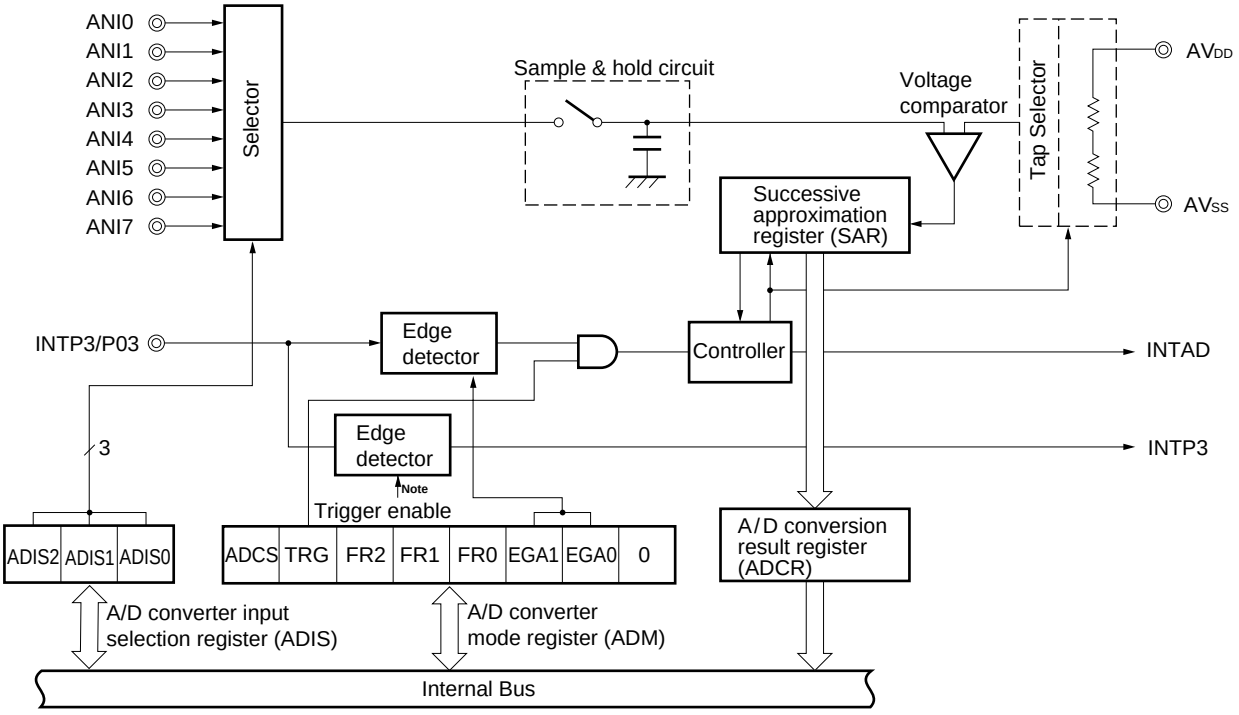
13.2 Configuration

The A/D converter has the following hardware configuration.

Table 13-1. A/D Converter Configuration

Item	Configuration
Analog input	8 channels (ANI0 to ANI7)
Control registers	A/D converter mode register (ADM) A/D converter input selection register (ADIS)
Registers	Successive approximation register (SAR) A/D conversion result register (ADCR)

Figure 13-1. A/D Converter Block Diagram



Note Valid edge specified with bit 3 (EGP3, EGN3) of the external interrupt Rising edge/Falling edge enable registers (EGP0, EGN0). (Refer to **Figure 21-1 Format of External Interrupt Rising Edge Enable Register (EGP0) and External Interrupt Falling Edge Enable Register (EGN0)**.)

(1) Successive approximation register (SAR)

Compares the voltage of the analog input with the voltage tap (comparison voltage) from the series resistor string, and saves the result from the most significant bit (MSB).

The contents of SAR will be transmitted across to the A/D conversion result register everything down to the last bit (LSB) is retained (A/D conversion finished).

(2) A/D conversion result register (ADCR)

Holds A/D conversion results. At the end of each A/D conversion operation, the conversion result from the successive approximation register is loaded.

ADCR is read with an 8-bit memory manipulation operation.

$\overline{\text{RESET}}$ input makes its contents undefined.

(3) Sample & hold circuit

Samples analog inputs one by one as they are sent from the input circuit, and sends them to the voltage comparator. The sampled analog input voltages are saved during A/D conversion.

(4) Voltage comparator

Compares the analog input voltage with the output voltage of the series resistor string.

(5) Series resistor string

Placed between AV_{DD} and AV_{SS}, generates the voltage that is compared with that of analog input signal.

(6) ANI0 to ANI7 pins

Eight analog input channels used for inputting analog data to the A/D converter for A/D conversion. Pins not selected for analog input with the A/D convertor input selection register (ADIS) can be used as input ports.

Cautions 1. Use ANI0 to ANI7 input voltages within the rated voltage range. Inputting a voltage equal to or greater than AV_{DD}, or equal to or smaller than AV_{SS} (even if within the absolute maximum rated range) will cause the channel's conversion values to become undefined, or may affect the conversion values of other channels.

2. Analog input (AN10 to AN17) pins alternate with input port (P10 to P17) pins. When performing an A/D conversion with the selection of any one of inputs from AN10 to AN17, do not execute input instructions to port 1 during conversion. Conversion resolution may decrease. When a digital pulse is applied to the pin which adjoins a pin in the A/D conversion, an expected A/D conversion value may not be acquired due to the coupling noise. Therefore do not apply a pulse to the pin which adjoins the pin in the A/D conversion.

(7) AV_{SS} pin

Ground pin of the A/D converter. Always use this pin at the same electric potential as the V_{SS} pin, even when not using the A/D converter.

(8) AV_{DD} pin

Analog power supply pin and reference voltage input pin of the A/D converter. Always keep this pin at the same electric potential as the V_{DD} pin, even when not using the A/D converter.

13.3 Registers

The A/D converter controls the following two registers.

- A/D converter mode register (ADM)
- A/D converter input selection register (ADIS)

(1) A/D converter mode register (ADM)

Used to set the A/D conversion time of analog input to be converted, start/stop of conversion operation, and external triggers.

ADM is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets ADM to 00H.

Figure 13-2. A/D Converter Mode Register (ADM) Format

Address: 0FF80H After Reset: 00H R/W

Symbol	⑦	⑥	⑤	④	③	②	①	0
ADM	ADCS	TRG	FR2	FR1	FR0	EGA1	EGA0	0

ADCS	A/D Conversion Control
0	Conversion stop
1	Conversion enable

TRG	Software Start/Hardware Start Selection
0	Software start
1	Hardware start

FR2	FR1	FR0	A/D Conversion Time Selection		
			Number of clocks	@f_{XX} = 12.5 MHz	@f_{XX} = 6.25 MHz
0	0	0	144/f_{XX}	Setting prohibited	23.0 μs
0	0	1	120/f_{XX}	Setting prohibited	19.2 μs
0	1	0	96/f_{XX}	Setting prohibited	15.4 μs
1	0	0	288/f_{XX}	23.0 μs	46.1 μs
1	0	1	240/f_{XX}	19.2 μs	38.4 μs
1	1	0	192/f_{XX}	15.4 μs	30.7 μs
Other than above			–	Setting prohibited	

EGA1	EGA0	External Trigger Signal Valid Edge Selection
0	0	No edge detectiont
0	1	Detection of falling edge
1	0	Detection of rising edge
1	1	Detection of both falling and rising edges

Cautions 1. Do not set the A/D conversion time less than 14 μs.

2. When overwritting FR0 to FR2 to unidentical data, temporarily halt the A/D conversion operations before continuing.

Remark f_{XX}: Main system clock frequency (f_X or f_X/2)

f_X : Main system clock oscillation frequency

(2) A/D converter input selection register (ADIS)

Used to specify the input ports for analog signals to be A/D converted.

ADIS can be set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets ADIS to 00H.

Figure 13-3. A/D Converter Input Selection Register (ADIS) Format

Address: 0FF81H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
ADIS	0	0	0	0	0	ADIS2	ADIS1	ADIS0

ADIS2	ADIS1	ADIS0	Analog Input Channel Setting
0	0	0	ANI0
0	0	1	ANI1
0	1	0	ANI2
0	1	1	ANI3
1	0	0	ANI4
1	0	1	ANI5
1	1	0	ANI6
1	1	1	ANI7

13.4 Operations

13.4.1 Basic Operations of A/D Converter

<1> Select one channel for A/D conversion with the A/D converter input selection register (ADIS).

<2> The voltage input to the selected analog input channel is sampled by the sample & hold circuit.

<3> After sampling has been performed for a certain time, the sample & hold circuit enters the hold status, and the input analog voltage is held until A/D conversion ends.

<4> Bit 7 of the successive approximation register (SAR) is set. The tab selector sets the voltage tab for the series resistance string at (1/2)AV_{DD}.

<5> The difference in voltage between the series resistance string's voltage table and analog input is compared with the voltage comparator. If the analog input is greater than (1/2)AV_{DD}, the setting for the SAR MSB will remain the same. If it is smaller than (1/2)AV_{DD}, the MSB will be reset.

<6> Next, bit 6 of SAR is automatically set, and the next comparison is started. The series resistor string voltage tap is selected as shown below according to bit 7 to which a result has already been set.

- Bit 7 = 1 : (3/4) AV_{DD}
- Bit 7 = 0 : (1/4) AV_{DD}

The voltage tap and analog input voltage are compared, and bit 6 of SAR is manipulated according to the result, as follows.

- Analog input voltage • Voltage tap : Bit 6 = 1
- Analog input voltage < Voltage tap : Bit 6 = 0

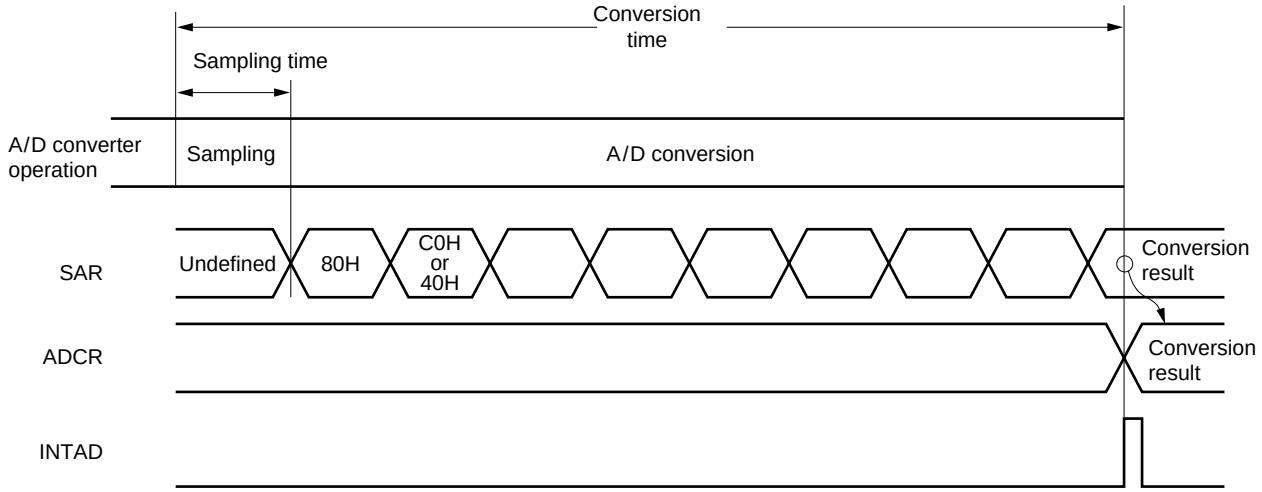
<7> Comparisons of this kind are repeated until bit 0 of SAR.

<8> When comparison of all eight bits is completed, the valid digital result remains in SAR, and this value is transferred to the A/D conversion result and latched.

At the same time, it is possible to have an A/D conversion end interrupt request (INTAD) issued.

Caution The value of the first A/D conversion is undefined immediately after the A/D conversion operation starts.

Figure 13-4. Basic Operations of A/D Converter



A/D conversion is performed continuously until bit 7 (ADCS) of A/D converter mode register (ADM) is reset 0 by software.

If a write operation to ADM or A/D converter input selected register (ADIS) is performed during A/D conversion, the conversion operation is initialized and conversion starts from the beginning if the ADCS bit is set 1.

$\overline{\text{RESET}}$ input makes A/D conversion result register undefined.

13.4.2 Input voltage and conversion result

The relationship between the analog input voltage input to the analog input pins (ANI0 to ANI7) and the A/D conversion result (value saved in A/D conversion result register (ADCR)) is expressed by the following equation.

$$ADCR = INT\left(\frac{V_{IN}}{AV_{DD}} \times 256 + 0.5\right)$$

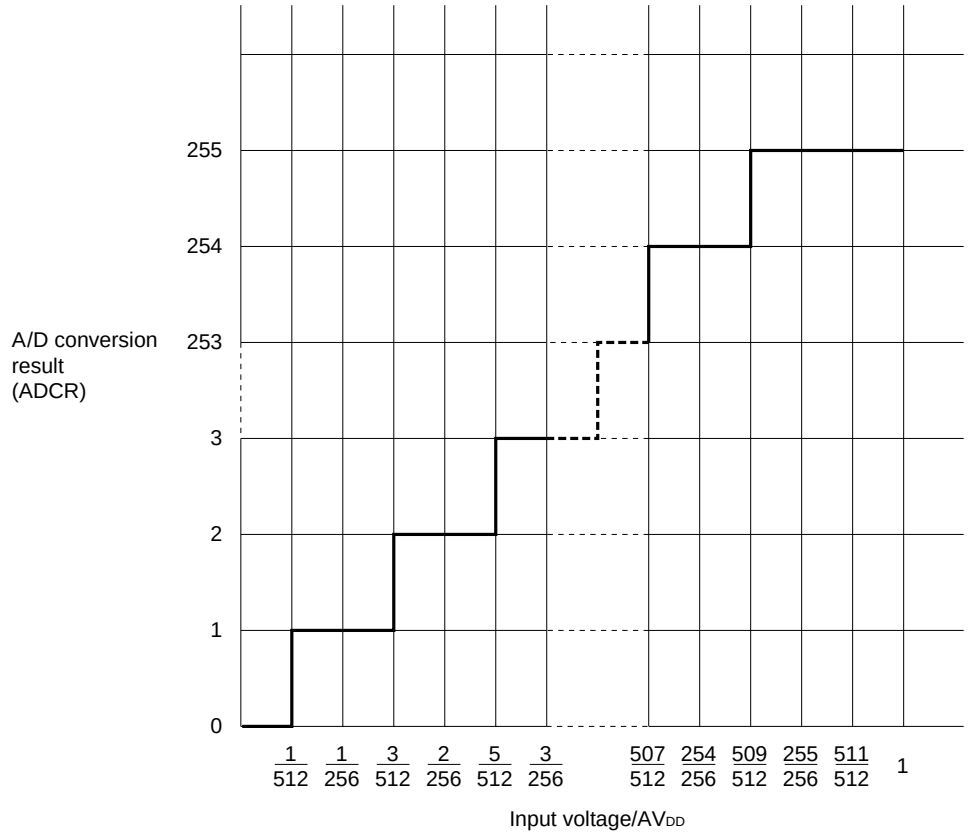
or

$$(ADCR - 0.5) \times \frac{AV_{DD}}{256} - V_{IN} < (ADCR + 0.5) \times \frac{AV_{DD}}{256}$$

Remark INT( ) : Function returning the integer portion of the value in parentheses
V_{IN} : Analog input voltage
AV_{DD} : AV_{DD} pin voltage
ADCR : A/D conversion result register (ADCR) value

Figure 13-5 shows the relationship between analog input voltage and the A/D conversion result.

Figure 13-5. Relationship between Analog Input Voltage and A/D Conversion Result



13.4.3 Operations mode of A/D converter

Select one channel for analog input from between ANI0 to ANI7 with the A/D converter input selection register (ADIS) and commence A/D conversion.

A/D conversion can be started in the following two ways.

- Hardware start : Conversion start by trigger input (P03)
- Software start : Conversion start by setting A/D converter mode register (ADM)

In addition to this, the result of A/D conversion will be stored in the A/D conversion result register (ADCR), and at the same time, an interrupt request signal (INTAD) will be issued.

(1) A/D conversion operation by hardware start

The A/D conversion operation can be made to enter the standby status by setting “1” to bit 6 (TRG) and bit 7 (ADCS) of the A/D converter mode register (ADM). When an external trigger signal (P03) is input, conversion of the voltage applied to the analog input pin set with ADIS begins.

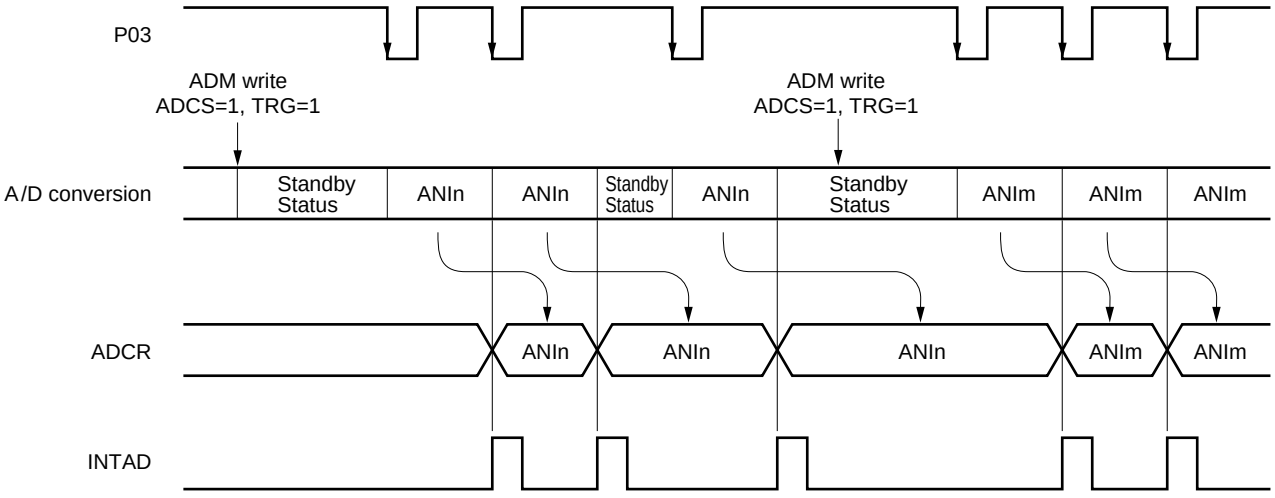
The result of conversion will be stored in the A/D conversion result register (ADCR) when A/D conversion operation have finished, and an interrupt request signal (INTAD) will be issued. When the A/D conversion operation that was started completes the first A/D conversion, no other A/D conversion operation is started unless an external trigger signal is input.

The A/D conversion process will be suspended if ADCS is overwritten during A/D conversion operations, and it will enter a stand-by mode until a new external trigger signal is input. The A/D conversion process will be restarted from the beginning when an external trigger signal is input once again. The A/D conversion process will be started when the next external trigger signal is received when ADCS is overwritten with A/D conversion in the stand-by mode.

If, during A/D conversion, data whose ADCS is 0 is written to ADM, A/D conversion is immediately stopped.

Caution When P03/INTP3 is used as the external trigger input (P03), specify a valid edge with bits 1 and 2 (EGA0 and EGA1) of the A/D converter mode register (ADM) and set 1 to the interrupt mask flag (PMK3).

Figure 13-6. A/D Conversion Operation by Hardware Start (When Falling Edge is Specified)



Remark n = 0, 1, ... , 7
m = 0, 1, ... , 7

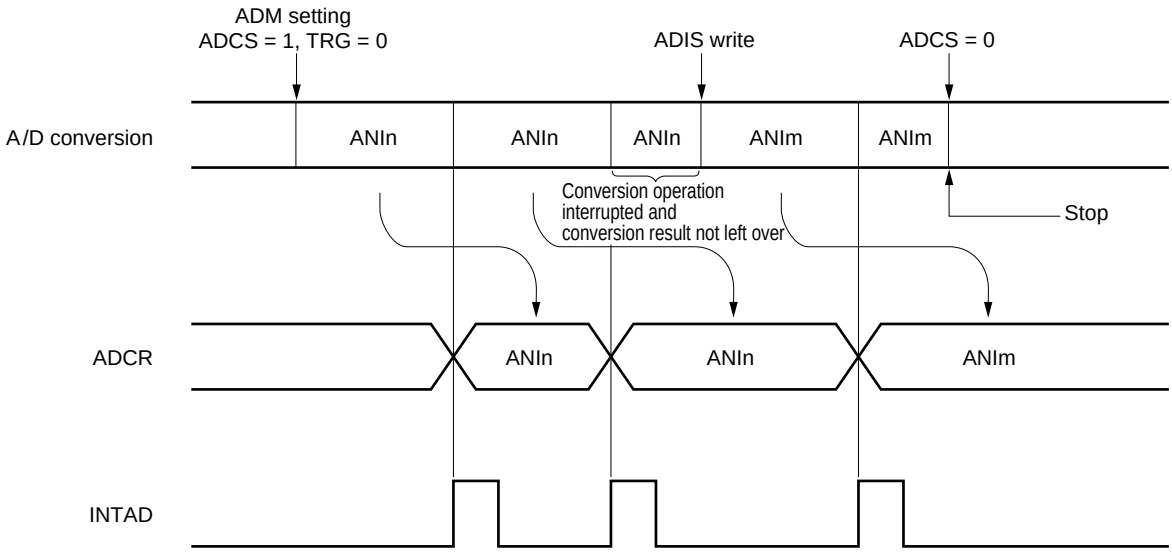
(2) A/D conversion operation by software start

A/D conversion of the voltage applied to the analog input pin specified with A/D converter input selected register (ADIS) is started by setting “0” to bit 6 (TRG) and “1” to bit 7 (ADCS) of the A/D converter mode register (ADM). When A/D conversion ends, the conversion result is saved in the A/D conversion result register (ADCR), and an interrupt request signal (INTAD) is issued. When an A/D conversion operation that was started completes the first A/D conversion, the next A/D conversion starts immediately. A/D conversion operations are performed continuously until new data is written to ADM.

The A/D conversion process will be suspended if ADCS is overwritten during A/D conversion operations, and A/D conversion operations for the newly selected analog input channel will be started.

If, during A/D conversion, data where ADCS is 0 is written to ADM, the A/D conversion operation is immediately stopped.

Figure 13-7. A/D Conversion Operation by Software Start



Remark n = 0, 1, ... , 7
m = 0, 1, ... , 7

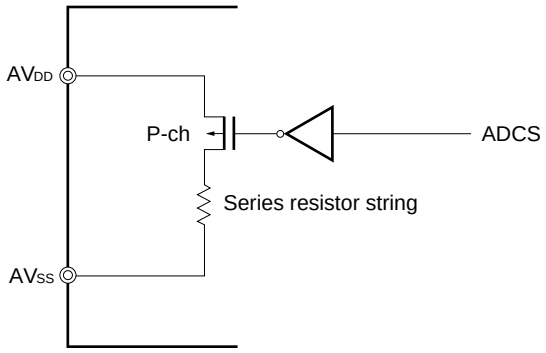
13.5 Cautions

(1) Current dissipation in standby mode

The A/D converter operation is stopped during the standby mode. At this time, the current dissipation can be reduced by setting bit 7 (ADCS) of the A/D converter mode register (ADM) to “0”.

The method to reduce the current dissipation in the standby mode is shown in Figure 13-8.

Figure 13-8. Method to Reduce Current Dissipation in Standby Mode



(2) ANI0 to ANI7 input range

Use ANI0 to ANI7 input voltages within the rated voltage range. Inputting a voltage equal to or greater than AV_{DD}, or equal to or smaller than AV_{SS} (even if within the absolute maximum rated range) will cause the channel’s conversion values to become undefined, or may affect the conversion values of other channels.

(3) Contention operation

<1> Contention with ADCR read due to contention between A/D conversion result register (ADCR) write and instruction at conversion end

The read operation to ADCR is prioritized. After the read operation, a new conversion result is written to ADCR.

<2> Contention between ADCR write and external trigger signal input at conversion end

External trigger signals cannot be received during A/D conversion. Therefore, external trigger signals during ADCR write operation are not received.

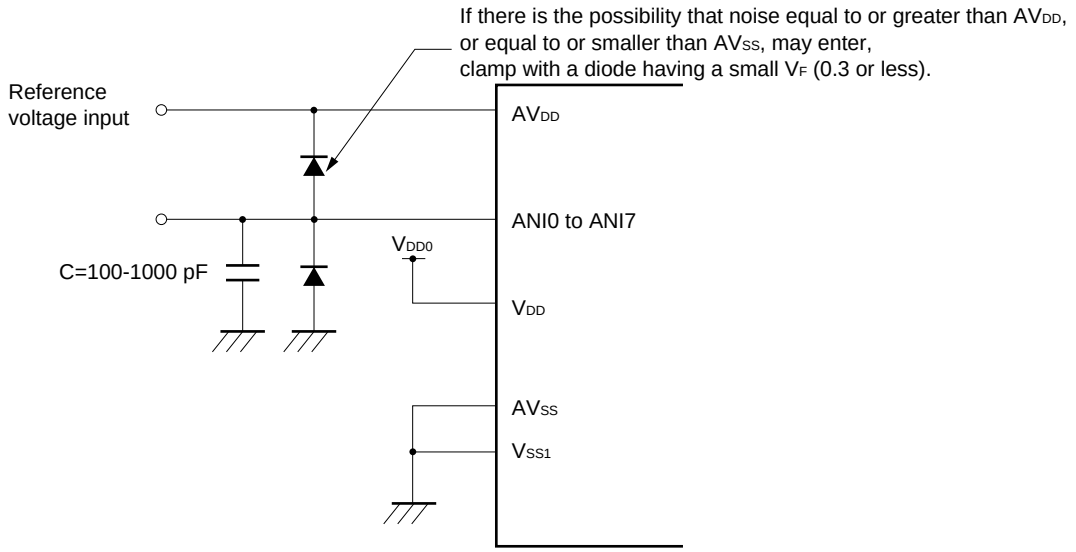
<3> Contention between ADCR write and A/D converter mode register (ADM) write, or between A/D converter input selection register (ADIS) write at conversion end

The write operation to ADM or ADIS is prioritized. Write to ADCR is not performed. Moreover, no interrupt signal (INTAD) is issued at conversion end.

(4) Anti-noise measures

Attention must be paid to noise fed to AV_{DD} and ANI0 to ANI7 to preserve the 8-bit resolution. The influence of noise grows proportionally to the output impedance of the analog input source. Therefore, it is recommended to connect C externally, as shown in Figure 13-9.

Figure 13-9. Handling of Analog Input Pin



(5) ANI0/P10 to ANI7/P17

The analog input pins (ANI0 to ANI7) can also be used as an input port pin (P10 to P17).

Do not execute the input command that corresponds with PORT 1 during conversion if any pin from between ANI0 to ANI7 has been selected and A/D conversion run. This would result in a lowered resolution.

Moreover, if a digital pulse is applied to pins adjacent to the pin for which A/D conversion is being performed, the A/D conversion value will not be obtained as expected because of coupling noise. Therefore, do not apply a pulse to pins adjacent to the pin for which A/D conversion is being performed.

(6) Input impedance of AV_{DD} pin

A series resistor string of approximately 46 ký is connected between the AV_{DD} and AV_{SS} pins.

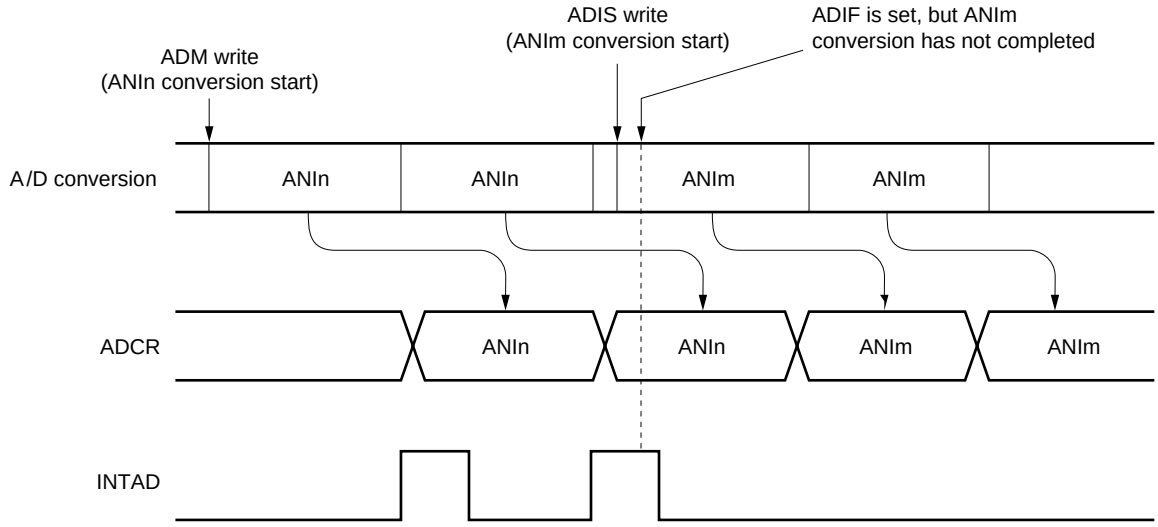
Therefore, if the output impedance of the reference voltage source is high, connecting in parallel a series resistor string between the AV_{DD} and AV_{SS} pins will result in a large reference voltage error.

(7) Interrupt request flag (ADIF)

The interrupt request flag (ADIF) is not cleared even if the A/D converter input selected register (ADIS) is changed. Owing to this, there will be cases when the A/D conversion result and ADIF that correspond with the pre-amended analog input immediately prior to ADM overwriting will be set if the analog input pin is amended during A/D conversion. It must therefore be noted that the ADIF will be set regardless of whether the A/D conversion for the amended analog input has finished or not when ADIF is read immediately after ADIS has been overwritten. These facts should be kept in mind.

Moreover, if A/D conversion is stopped once and then resumed, clear ADIF before resuming conversion.

Figure 13-10. A/D Conversion End Interrupt Request Generation Timing



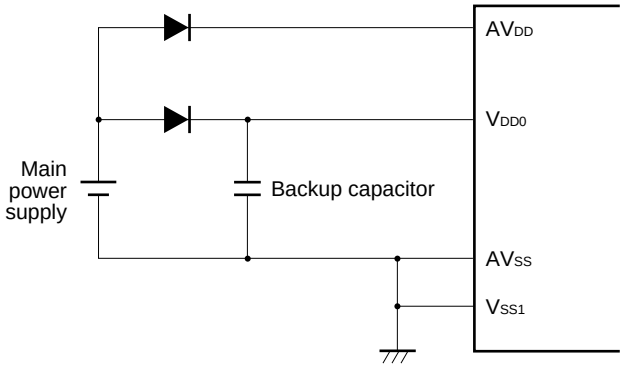
Remark n = 0, 1, ⋯, 7
m = 0, 1, ⋯, 7

(8) AV_{DD} pin

The AV_{DD} pin is the analog circuit's power pin and A/D converter's standard voltage input pin and also supplies power to the ANI0/P10 to ANI7/P17 input circuits.

Therefore, be sure to apply the same electric potential level as V_{DD} as shown in Figure 13-11, even in applications that can be switched to a backup power supply.

Figure 13-11. Handling of AV_{DD} Pin



(9) Conversion results immediately after A/D conversion is started

The value of the first A/D conversion is undefined immediately after the A/D conversion operation starts. Poll the A/D conversion terminate interrupt reuest (INTAD) and take measures such as discarding the first conversion result.

[MEMO]

CHAPTER 14 D/A CONVERTER

14.1 Function

The D/A converter converts the digital input into analog values and consists of two channels of voltage output D/A converters with 8-bit resolution.

The conversion method is a R-2R resistor ladder.

Set DACE0 of D/A converter mode register 0 (DAM0) and DACE1 of D/A converter mode register 1 (DAM1) to start the D/A conversion.

The D/A converter has the following two modes.

(1) Normal mode

After D/A conversion, the analog voltage is immediately output.

(2) Real-time output mode

After D/A conversion, the analog voltage is output synchronized to the output trigger.

Since a sine wave is created when this mode is used, MSK modems can be easily incorporated into cordless phones.

Caution If only one channel of the D/A converter is used when $AV_{REF1} < V_{DD}$, make either of the following setting at pins that are not used for analog output.

- **Set the port mode register (PM13X) to one (input mode) and connect to V_{SS}.**
- **Set the port mode register (PM13X) to zero (output mode) and the output latch to zero, and output a low level.**

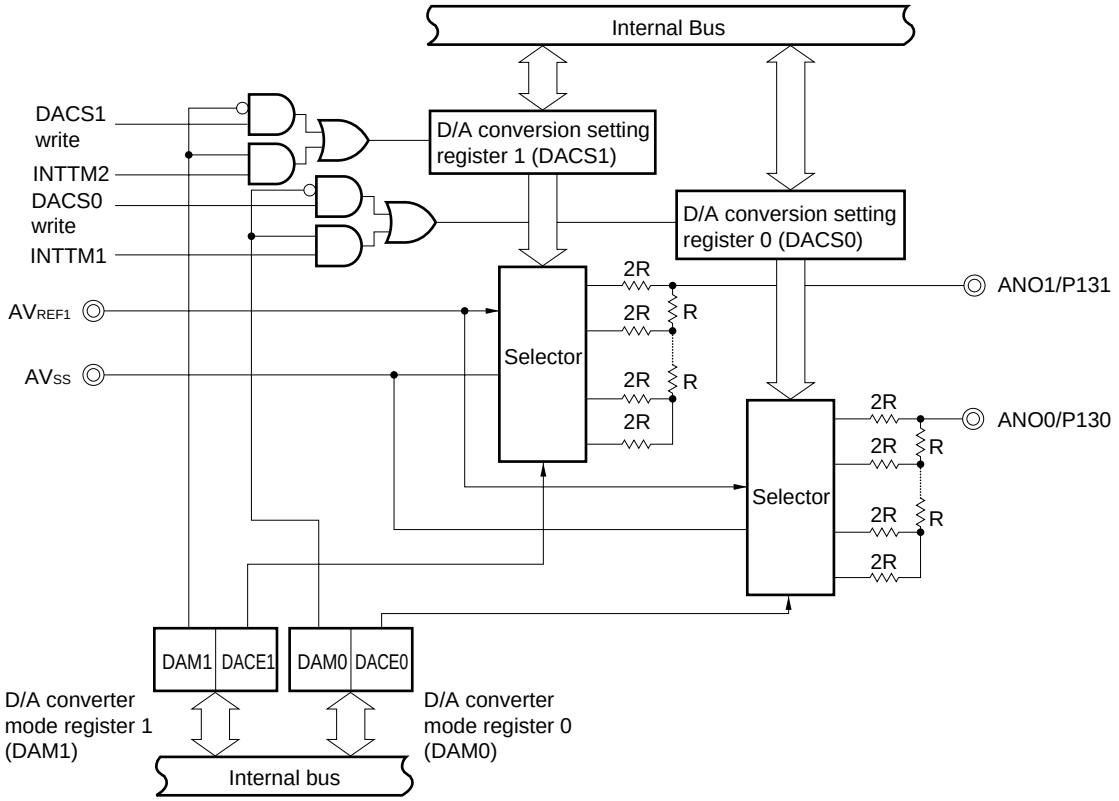
14.2 Structure

The D/A converter has the following hardware.

Table 14-1. D/A Converter Structure

Item	Structure
Registers	D/A conversion setting register 0 (DACS0) D/A conversion setting register 1 (DACS1)
Control registers	D/A converter mode register 0 (DAM0) D/A converter mode register 1 (DAM1)

Figure 14-1. D/A Converter Block Diagram



(1) D/A conversion setting registers 0, 1 (DACS0, DACS1)

The DACS0 and DACS1 registers set the analog voltages that are output to the ANO0 and ANO1 pins, respectively.

DACS0 and DACS1 are set by 8-bit memory manipulation instructions.

$\overline{\text{RESET}}$ input sets DACS0 and DACS1 to 00H.

The analog voltages output by the ANO0 and ANO1 pins are determined by the following equation.

$$\text{ANOn output voltage} = AV_{REF1} \times \frac{\text{DACSn}}{256}$$

n = 0, 1

Cautions 1. In the real-time output mode, when the data set in DACS0 and DACS1 are read before the output trigger is generated, the set data is not read and the previous data is read.

2. In the real-time output mode, set the data of DACS0 and DACS1 until the next output trigger is generated after the output trigger is generated.

14.3 Control Registers

- **D/A converter mode registers 0, 1 (DAM0, DAM1)**

  D/A converters are controlled by D/A converter mode registers 0, 1 (DAM0, DAM1). These registers enable or stop the operation of the D/A converters.

  DAM0 and DAM1 are set by a 1-bit and 8-bit memory manipulation instruction.

  $\overline{\text{RESET}}$ input sets DAM0 and DAM1 to 00H.

Figure 14-2. D/A Converter Mode Registers 0, 1 (DAM0, DAM1) Formats

Address: 0FF86H, 0FF87H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	⓪
DAMn	0	0	0	0	0	0	DAMn	DACEn

DAMn	D/A Converter Channel n Operating Mode
0	Normal mode
1	Real-time output mode

DACEn	D/A Converter Channel n Control
0	Stop conversion
1	Enable conversion

Cautions 1. When the D/A converters are used, set the shared port pins to the input mode and disconnect the pullup resistors.

2. Always set bits 2 to 7 to 0.

3. The output when the D/A converter operation has stopped enters high impedance state.

4. The output triggers in the real-time output mode are INTTM1 in channel 0 and INTTM2 in channel 1.

Remark n = 0, 1

14.4 Operation

<1> Select the operating mode in channel 0 in DAM0 of D/A converter mode register 0 (DAM0) and the operating mode of the channel 1 in DAM1 of D/A converter mode register 1 (DAM1).

<2> Set the data that corresponds to the analog voltages that are output to pins ANO0/P130 and ANO1/P131 of D/A conversion setting registers 0 and 1 (DACS0, DACS1).

<3> Set DACE0 of DAM0 and DACE1 of DAM1 to start D/A conversion in channels 0 and 1.

<4> After D/A conversion in the normal mode, the analog voltages at pins ANO0/P130 and ANO1/P131 are immediately output. In the real-time output mode, the analog voltage is output synchronized to the output trigger.

<5> In the normal mode, the output analog voltages are maintained until new data are set in DACS0 and DACS1. In the real-time output mode, after new data are set in DACS0 and DACS1, they are held until the next output trigger is generated.

Caution Set DACE0 and DACE1 after data are set in DACS0 and DACS1.

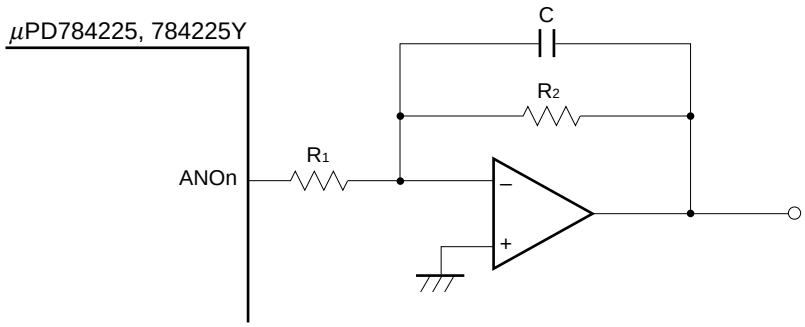
14.5 Cautions

(1) Output impedances of the D/A converters

Since the output impedances of the D/A converters are high, the current cannot be taken from the ANOn pin (n = 0,1). If the input impedance of the load is low, insert a buffer amp between the load and the ANOn pin. In addition, use the shortest possible wire from the buffer amp or load (to increase the output impedance). If the wire is long, surround it with a ground pattern.

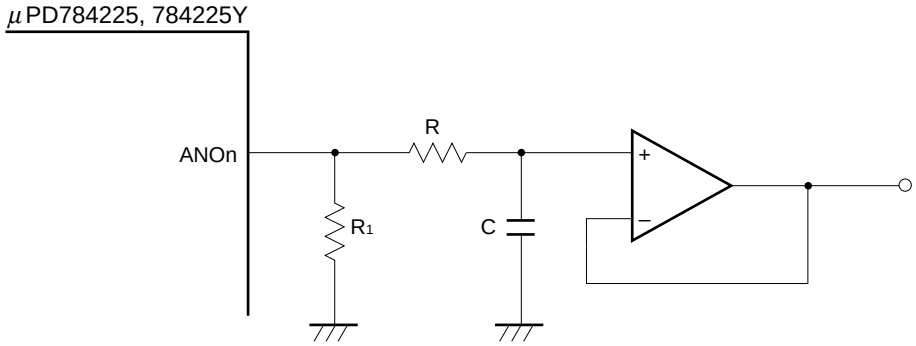
Figure 14-3. Buffer Amp Insertion Example

(a) Inverting Amp



- The input impedance of the buffer amp is R_1.

(b) Voltage follower



- The input impedance of the buffer amp is R_1.
- If there is no R_1 and $\overline{\text{RESET}}$ is low, the output is undefined.

(2) Output voltages of the D/A converters

Since the output voltages of the D/A converters change in stages, use the signals output from the D/A converters after passing them through low-pass filters.

(3) AV_{REF1} pin

Deal with pins not being used for analog output in either of the following ways when the D/A converter is only using one channel with $AV_{REF1} < V_{DD}$.

- Set the port mode register (PM13X) to 1 (input mode) and connect to V_{SS0}.
- Set the port mode register (PM13X) to 0 (output mode), set the output latch to zero, and output a low level.

[MEMO]

CHAPTER 15 SERIAL INTERFACE OVERVIEW

The μPD784225 Subseries has a serial interface with three independent channels. Therefore, communication outside and within the system can be simultaneous on the three channels.

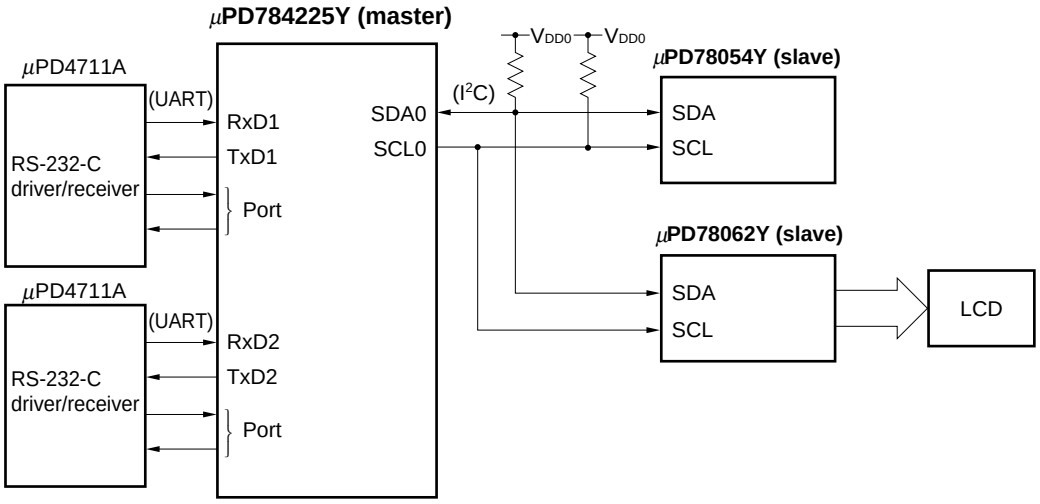
- Asynchronous serial interface (UART)/3-wire serial I/O (IOE) × 2 channels
  → See **CHAPER 16**.

- Clock-synchronized serial interface (CSI) × 1 channel
  - 3-wire serial I/O mode (MSB or LSB first)
    → See **CHAPTER 17**.

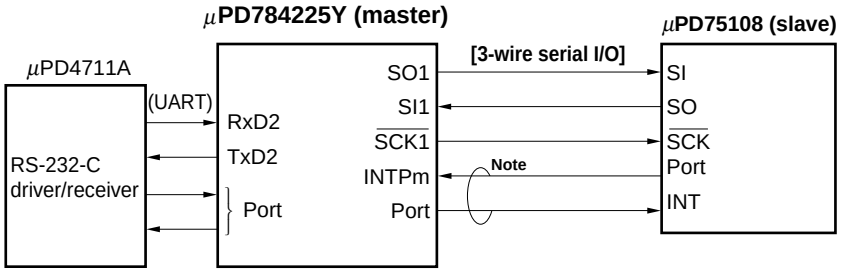
  - I^2C bus mode (multi-master compatible) (only in the μPD784225Y Subseries)
    → See **CHAPTER 18**.

Figure 15-1. Serial Interface Example

(a) UART + I^2C



(b) UART + 3-wired serial I/O



Note Handshake lines

CHAPTER 16 ASYNCHRONOUS SERIAL INTERFACE/3-WIRE SERIAL I/O

μPD784225 provides on chip two serial interface channels for which the asynchronous serial interface (UART) mode and the 3-wire serial I/O (IOE) mode can be selected.

These two serial interface channels have exactly the same functions.

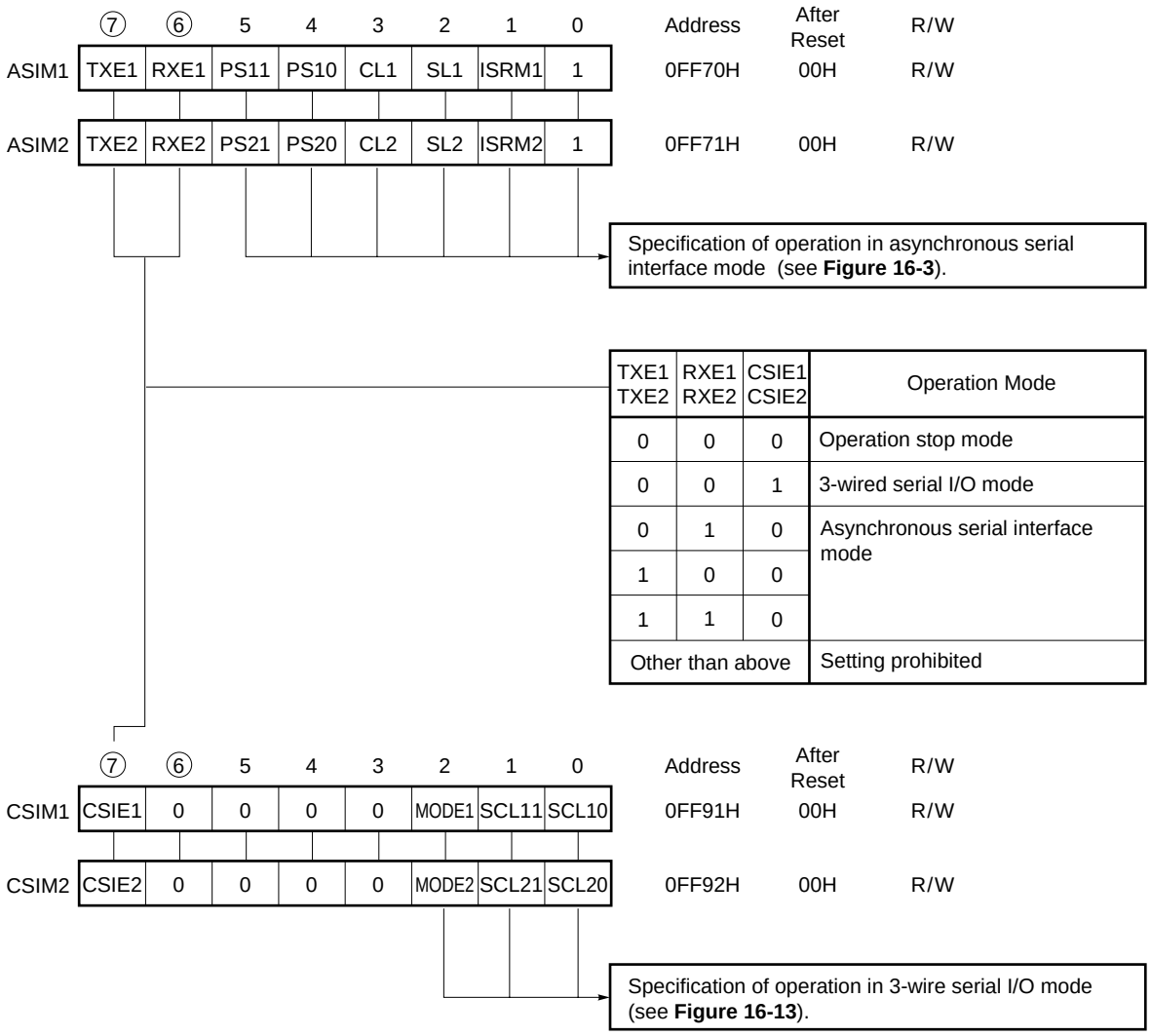
Table 16-1. Designation Differences between UART1/IOE1 and UART2/IOE2

Item	UART1/IOE1	UART1/IOE2
Pin name	P22/ASCK1/$\overline{\text{SCK1}}$, P20/RxD1/SI1, P21/TxD1/SO1	P72/ASCK2/$\overline{\text{SCK2}}$, P70/RxD2/SI2, P71/TxD2/SO2
Asynchronous serial interface mode register	ASIM1	ASIM2
Name of bits inside asynchronous serial interface mode register	TXE1, RXE1, PS11, PS10, CL1, SL1, ISRM1	TXE2, RXE2, PS21, PS20, CL2, SL2, ISRM2
Asynchronous serial interface status register	ASIS1	ASIS2
Name of bits inside asynchronous serial interface status register	PE1, FE1, OVE1	PE2, FE2, OVE2
Serial operation mode register	CSIM1	CSIM2
Name of bits inside serial operation mode register	CSIE1, MODE1, SCL11, SCL10	CSIE2, MODE2, SCL21, SCL20
Baud rate generator control register	BRGC1	BRGC2
Name of bits inside baud rate generator control register	TPS10 to TPS12, MDL10 to MDL13	TPS20 to TPS22, MDL20 to MDL23
Interrupt request name	INTSR1/INTCSI1, INTSER1, INTST1	INTSR2/INTCSI2, INTSER2, INTST2
Interrupt control register and name of bits used in this chapter	SRIC1, SERIC1, STIC1, SRIF1, SERIF1, STIF1	SRIC2, SERIC2, STIC2, SRIF2, SERIF2, STIF2

16.1 Switching Asynchronous Serial Interface Mode and 3-Wire Serial I/O Mode

The asynchronous serial interface mode and the 3-wire serial I/O mode cannot be used at the same time. Both these modes can be switched by setting the asynchronous serial interface mode registers (ASIM1, ASIM2) and the serial operation mode registers (CSIM1, CSIM2), as shown in Figure 16-1 below.

Figure 16-1. Switching Asynchronous Serial Interface Mode and 3-Wire Serial I/O Mode



16.2 Asynchronous Serial Interface Mode

The asynchronous serial interface (UART: Universal Asynchronous Receiver Transmitter) offers the following two modes.

(1) Operation stop mode

This mode is used when serial transfer is not performed to reduce the power consumption.

(2) Asynchronous serial interface (UART) mode

This mode is used to send and receive 1-byte data that follows the start bit, and supports full-duplex transmission.

A UART-dedicated baud rate generator is provided on-chip, enabling transmission at any baud rate within a broad range. The baud rate can also be defined by dividing the input clock to the ASCK pin.

The MIDI specification baud rate (31.25 kbps) can be used by utilizing the UART-dedicated baud rate generator.

16.2.1 Configuration

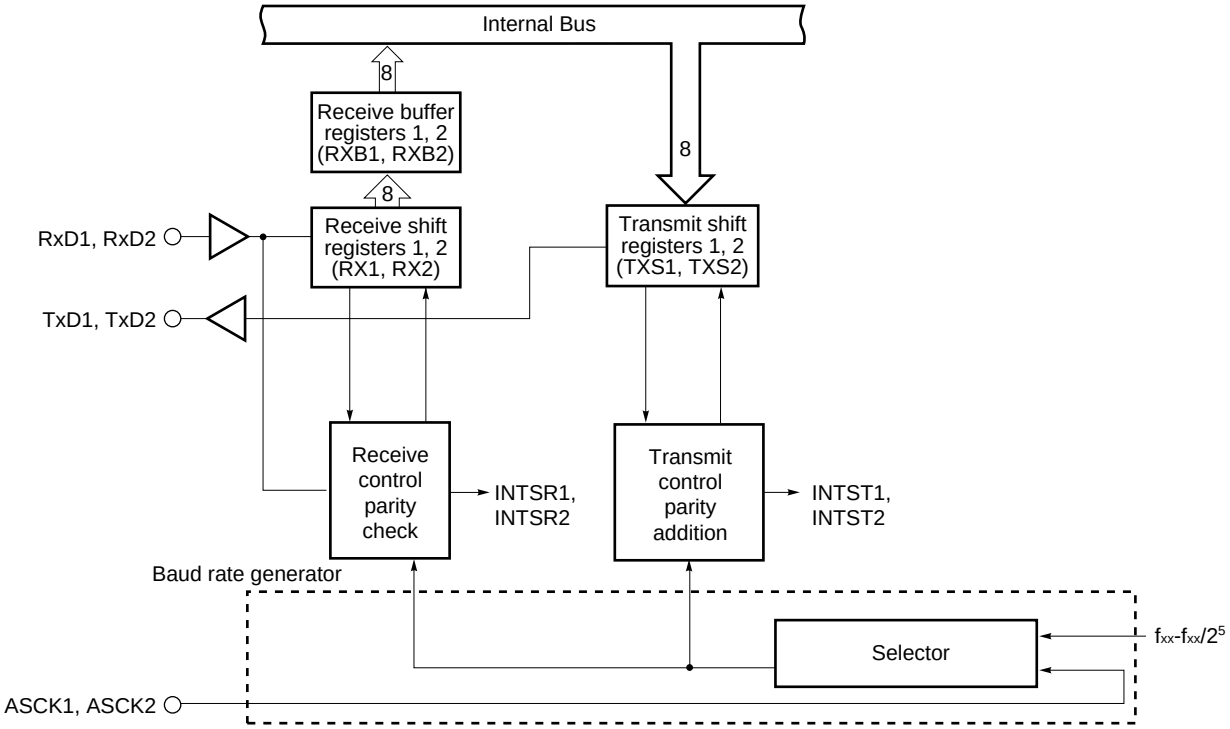
The asynchronous serial interface has the following hardware configuration.

Figure 16-2 gives the block diagram of the asynchronous serial interface.

Table 16-2. Asynchronous Serial Interface Configuration

Item	Configuration
Registers	Transmit shift registers (TXS1, TXS2) Receive shift registers (RX1, RX2) Receive buffer registers (RXB1, RXB2)
Control registers	Asynchronous serial interface mode registers (ASIM1, ASIM2) Asynchronous serial interface status registers (ASIS1, ASIS2) Baud rate generator control registers (BRGC1, BRGC2)

Figure 16-2. Block Diagram in Asynchronous Serial Interface Mode



(1) Transmit shift registers (TXS1, TXS2)

These registers are used to set transmit data. Data written to TXS1 and TXS2 is sent as serial data.

If a data length of 7 bits is specified, bits 0 to 6 of the data written to TXS1 and TXS2 are transferred as transmit data. Transmission is started by writing data to TXS1 and TXS2.

TX1 and TX2 can be written with an 8-bit memory manipulation instruction, but cannot be read.

$\overline{\text{RESET}}$ input sets TXS1 and TXS2 to FFH.

Caution Do not write to TXS1 and TXS2 during transmission.

TXS1, TXS2 and receive buffer registers RXB1, RXB2 are allocated to the same address. Therefore, attempting to read TXS1 and TXS2 will result in reading the values of RXB1 and RXB2.

(2) Receive shift registers (RX1, RX2)

These registers are used to convert serial data input to the RxD1 and RxD2 pins to parallel data. Receive data is transferred to the receive buffer register (RXB1, RSB2) one byte at a time as it is received.

RX1 and RX2 cannot be directly manipulated by program.

(3) Receive buffer registers (RXB1, RXB2)

These registers are used to hold receive data. Each time one byte of data is received, new receive data is transferred from the receive shift registers (RX1, RX2)

If a data length of 7 bits is specified, receive data is transferred to bits 0 to 6 of RXB1 and RXB2, and the MSB of RXB1 and RXB2 always becomes 0.

RXB1 and RXB2 can be read by an 8-bit memory manipulation instruction, but cannot be written.

$\overline{\text{RESET}}$ input sets RXB1 and RXB2 to FFH.

Caution RXB1, RXB2 and transmit shift registers TXB1, TXB2 are allocated to the same address. Therefore, attempting to read RXB1 and RXB2 will result in reading the values of TXB1 and TXB2.

(4) Transmission control circuit

This circuit controls transmit operations such as the addition of a start bit, parity bit, and stop bit(s) to data written to transmit shift registers (TXS1, TXS2), according to the contents set to the asynchronous serial interface mode registers (ASIM1, ASIM2).

(5) Reception control circuit

This circuit controls reception according to the contents set to the asynchronous serial interface mode registers (ASIM1, ASIM2). It also performs error check for parity errors, etc., during reception and transmission. If it detects an error, it sets a value corresponding to the nature of the error in the asynchronous serial interface status registers (ASIS1, ASIS2).

16.2.2 Control registers

The asynchronous serial interface controls the following six types of registers.

- Asynchronous serial interface mode registers 1, 2 (ASIM1, ASIM2)
- Asynchronous serial interface status registers 1, 2 (ASIS1, ASIS2)
- Baud rate generator control registers 1, 2 (BRGC1, BRGC2)

(1) Asynchronous serial interface mode registers 1, 2 (ASIM1, ASIM2)

ASIM1 and ASIM2 are 8-bit registers that control serial transfer using the asynchronous serial interface.

ASIM1 and ASIM2 are set by a 1-bit or 8-bit memory manipulation operation.

$\overline{\text{RESET}}$ input sets ASIM1 and ASIM2 to 00H.

Figure 16-3. Asynchronous Serial Interface Mode Registers 1, 2 (ASIM1, ASIM2) Format

Address: 0FF70H, 0FF71H After Reset: 00H R/W

Symbol	⑦	⑥	5	4	3	2	1	0
ASIMn	TXEn	RXEn	PSn1	PSn0	CLn	SLn	ISRMn	0 **Note**

TXEn	RXEn	Operation Mode	RxD1/P20, RxD2/P70 Pin Function	TxD1/P21, TxD2/P71 Pin Function
0	0	Operation stop	Port function	Port function
0	1	UART mode (Receive only)	Serial function	Port function
1	0	UART mode (Transmit only)	Port function	Serial function
1	1	UART mode (Transmit/Receive)	Serial function	Serial function

PSn1	PSn0	Parity Bit Specification
0	0	No parity
0	1	Always add 0 parity during transmission Do not perform parity check during reception (parity error not generated)
1	0	Odd parity
1	1	Even parity

CLn	Character Length Specification
0	7 bits
1	8 bits

SLn	Transmit Data Stop Bit Length Specification
0	1 bit
1	2 bits

ISRMn	Receive Completion Interrupt Control at Error Occurrence
0	Generate receive completion interrupt request when error occurs
1	Do not generate receive completion interrupt request when error occurs

Note Ensure that “0” is written in bit-0.

Caution Switch across to the operational mode after stopping serial sending and receiving operations.

Remark n = 1, 2

(2) Asynchronous serial interface status registers 1, 2 (ASIS1, ASIS2)

ASIS1 and ASIS2 are registers used display the type of error when a receive error occurs.

ASIS1 and ASIS2 can be read with 1-bit and 8-bit memory manipulation instructions.

$\overline{\text{RESET}}$ input sets ASIS1 and ASIS2 to 00H.

Figure 16-4. Asynchronous Serial Interface Status Registers 1, 2 (ASIS1, ASIS2) Format

Address: 0FF72H, 0FF73H After Reset: 00H R/W

Symbol	7	6	5	4	3	②	①	⓪
ASISn	0	0	0	0	0	PEn	FEn	OVEn

PEn	Parity Error Flag
0	Parity error not generated
1	Parity error generated (when parity of transmit data does not match)

FEn	Framing Error Flag
0	Framing error not generated
1	Framing error generated**Note 1** (when stop bit(s) is not detected)

OVEn	Overrun Error Flag
0	Overrun error not generated
1	Overrun error generated**Note2** (When next receive operation is completed before data from receive buffer register is read)

Notes 1. Even if the stop bit length has been set to 2 bits with bit 2 (SLn) of the asynchronous serial interface mode register (ASIMn), stop bit detection during reception is only 1 bit.

2. Be sure to read the receive buffer register (RXBn) when an overrun error occurs. An overrun error is generated each time data is received until RXBn is read.

Remark n = 1, 2

(3) Baud rate generator control registers 1, 2 (BRGC1, BRGC2)

BRGC1 and BRGC2 are registers used to set the serial clock of the asynchronous serial interface.

BRGC1 and BRGC2 are set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets BRGC1 and BRGC2 to 00H.

Figure 16-5. Baud Rate Generator Control Registers 1, 2 (BRGC1, BRGC2) Format

Address: 0FF76H, 0FF77H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
BRGCn	0	TPSn2	TPSn1	TPSn0	MDLn3	MDLn2	MDLn1	MDLn0

TPSn2	TPSn1	TPSn0	5-Bit Counter Source Clock Selection
0	0	0	External clock (ASCKn)
0	0	1	f_{XX} (12.5 MHz)
0	1	0	$f_{XX}/2$ (6.5 MHz)
0	1	1	$f_{XX}/4$ (3.13 MHz)
1	0	0	$f_{XX}/8$ (1.56 MHz)
1	0	1	$f_{XX}/16$ (781 kHz)
1	1	0	$f_{XX}/32$ (391 kHz)
1	1	1	TO1 (TM1 output)

MDLn3	MDLn2	MDLn1	TPSn0	Baud Rate Generator Input Clock Selection	k
0	0	0	0	$f_{SCK}/16$	0
0	0	0	1	$f_{SCK}/17$	1
0	0	1	0	$f_{SCK}/18$	2
0	0	1	1	$f_{SCK}/19$	3
0	1	0	0	$f_{SCK}/20$	4
0	1	0	1	$f_{SCK}/21$	5
0	1	1	0	$f_{SCK}/22$	6
0	1	1	1	$f_{SCK}/23$	7
1	0	0	0	$f_{SCK}/24$	8
1	0	0	1	$f_{SCK}/25$	9
1	0	1	0	$f_{SCK}/26$	10
1	0	1	1	$f_{SCK}/27$	11
1	1	0	0	$f_{SCK}/28$	12
1	1	0	1	$f_{SCK}/29$	13
1	1	1	0	$f_{SCK}/30$	14
1	1	1	1	Setting prohibited	–

Caution If a write operation to BRGCn is performed during communication, the baud rate generator output will become garbled and normal communication will not be achieved. Consequently, do not write in BRGCn during communications.

Remarks 1. n = 1, 2

2. Data in parentheses is for when f_{XX} = 12.5 MHz

3. f_{SCK} : Source clock of 5-bit counter

4. k : Value set in MDLn0 to MDLn3 (0 - k - 14)

16.3 Operation

The asynchronous serial interface has the following two types of operation modes.

- Operation stop mode
- Asynchronous serial interface (UART) mode

16.3.1 Operation stop mode

Serial transfer cannot be performed in the operation stop mode, resulting in reduced power consumption. Moreover, in the operation stop mode, pins can be used as regular ports.

(1) Register setting

Setting of the operation stop mode is done with asynchronous serial interface mode registers 1 and 2 (ASIM1, ASIM2).

ASIM1 and ASIM2 are set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets ASIM 1 and ASIM2 to 00H.

Address: 0FF70H, 0FF71H After Reset: 00H R/W

Symbol	⑦	⑥	5	4	3	2	1	0
ASIMn	TXEn	RXEn	PSn1	PSn0	CLn	SLn	ISRMn	0 **Note**

TXEn	RXEn	Operation Mode	RxD1/P20, RxD2/P70 Pin Function	TxD1/P21, TxD2/P71 Pin Function
0	0	Operation stop	Port function	Port function
0	1	UART mode (Receive only)	Serial function	Port function
1	0	UART mode (Transmit only)	Port function	Serial function
1	1	UART mode (Transmit/Receive)	Serial function	Serial function

Note Ensure that “0” is written in bit-0.

Caution Switch across to the operational mode after stopping serial sending and receiving operations.

Remark n = 1, 2

16.3.2 Asynchronous serial interface (UART) mode

This mode is used to transmit and receive the 1-byte data following the start bit. It supports full-duplex operation.

A UART-dedicated baud rate generator is incorporated enabling communication using any baud rate within a large range.

The MIDI standard's baud rate (31.25 kbps) can be used utilizing the UART-dedicated baud rate generator.

(1) Register setting

The UART mode is set with asynchronous serial interface mode registers 1 and 2 (ASIM1, ASIM2), asynchronous serial interface status registers 1 and 2 (ASIS1, ASIS2), and baud rate generator control registers 1 and 2 (BRGC1, BRGC2).

(a) Asynchronous serial interface mode registers 1, 2 (ASIM1, ASIM2)

ASIM1 and ASIM2 can be set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets ASIM1 and ASIM2 to 00H.

Address: 0FF70H, 0FF71H After Reset: 00H R/W

Symbol	⑦	⑥	5	4	3	2	1	0
ASIMn	TXEn	RXEn	PSn1	PSn0	CLn	SLn	ISRMn	0 **Note**

TXEn	RXEn	Operation Mode	RxD1/P20, RxD2/P70 Pin Function	TxD1/P21, TxD2/P71 Pin Function
0	0	Operation stop	Port function	Port function
0	1	UART mode (Receive only)	Serial function	Port function
1	0	UART mode (Transmit only)	Port function	Serial function
1	1	UART mode (Transmit/Receive)	Serial function	Serial function

PSn1	PSn0	Parity Bit Specification
0	0	No parity
0	1	Always add 0 parity during transmission Do not perform parity check during reception (parity error not generated)
1	0	Odd parity
1	1	Even parity

CLn	Character Length Specification
0	7 bits
1	8 bits

SLn	Transmit Data Stop Bit Length Specification
0	1 bit
1	2 bits

ISRMn	Receive Completion Interrupt Control at Error Occurrence
0	Generate receive completion interrupt when error occurs
1	Do not generate receive completion interrupt when error occurs

Notes Ensure that “0” is written in bit-0.

Caution Switch across to the operational mode after stopping serial sending and receiving operations.

Remark n = 1, 2

(b) Asynchronous serial interface status registers 1, 2 (ASIS1, ASIS2)

ASIS1 and ASIS2 can be read by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets ASIS1 and ASIS 2 to 00H.

Address: 0FF72H, 0FF73H After Reset: 00H R

Symbol	7	6	5	4	3	②	①	⓪
ASISn	0	0	0	0	0	PEn	FEn	OVEn

PEn	Parity Error Flag
0	Parity error not generated
1	Parity error generated (when parity of transmit data does not match)

FEn	Framing Error Flag
0	Framing error not generated
1	Framing error generated[Note 1] (when stop bit(s) is not detected)

OVEn	Overrun Error Flag
0	Overrun error not generated
1	Overrun error generated[Note2] (When next receive operation is completed before data from receive buffer register is read)

Notes 1. Even if the stop bit length has been set to 2 bits with bit 2 (SLn) of the asynchronous serial interface mode register (ASIMn), stop bit detection during reception is only 1 bit.

2. Be sure to read the receive buffer register (RXBn) when an overrun error occurs. An overrun error is generated each time data is received until RXBn is read.

Remark n = 1, 2

(c) Baud rate generator control registers 1, 2 (BRGC1, BRGC2)

BRGC1 and BRGC2 are set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets BRGC1 and BRGC2 to 00H.

Address: 0FF76H, 0FF77H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
BRGCn	0	TPSn2	TPSn1	TPSn0	MDLn3	MDLn2	MDLn1	MDLn0

TPSn2	TPSn1	TPSn0	5-Bit Counter Source Clock Selection
0	0	0	External clock (ASCKn)
0	0	1	f_{XX} (12.5 MHz)
0	1	0	$f_{XX}/2$ (6.25 MHz)
0	1	1	$f_{XX}/4$ (3.13 MHz)
1	0	0	$f_{XX}/8$ (1.56 MHz)
1	0	1	$f_{XX}/16$ (781 kHz)
1	1	0	$f_{XX}/32$ (391 kHz)
1	1	1	TO1 (TM1 output)

MDLn3	MDLn2	MDLn1	MDLn0	Baud Rate Generator Input Clock Selection	k
0	0	0	0	$f_{SCK}/16$	0
0	0	0	1	$f_{SCK}/17$	1
0	0	1	0	$f_{SCK}/18$	2
0	0	1	1	$f_{SCK}/19$	3
0	1	0	0	$f_{SCK}/20$	4
0	1	0	1	$f_{SCK}/21$	5
0	1	1	0	$f_{SCK}/22$	6
0	1	1	1	$f_{SCK}/23$	7
1	0	0	0	$f_{SCK}/24$	8
1	0	0	1	$f_{SCK}/25$	9
1	0	1	0	$f_{SCK}/26$	10
1	0	1	1	$f_{SCK}/27$	11
1	1	0	0	$f_{SCK}/28$	12
1	1	0	1	$f_{SCK}/29$	13
1	1	1	0	$f_{SCK}/30$	14
1	1	1	1	Setting prohibited	–

Caution If a write operation to BRGC1 and BRGC2 is performed during communication, the baud rate generator output will become garbled and normal communication will not be achieved. Consequently, do not write in BRGC1 or BRGC2 during communications.

Remarks 1. n = 1, 2

2. Figures in parentheses apply to operation with f_{XX} = 12.5 MHz.

3. f_{SCK} : Source clock of 5-bit counter

4. k : Value set in MDLn0 to MDLn3 (0 - k - 14)

The transmission clock for the baud rate to be generated is the signal obtained by dividing the main system clock.

- **Generation of transmit/receive clock for baud rate by using main system clock**
  Generate the transmit/receive clock by dividing the main system clock. The baud rate generated from the main system clock is obtained from the following equation.

$$[\text{Baud rate}] = \frac{f_X}{2^{m+1} \times (k + 16)} \text{ [Hz]}$$

f_X : Main system clock oscillation frequency
m : Value set in TPSn0 to TPSn2 (0 - m - 5)
k : Value set in MDLn0 to MDLn3 (0 - k - 14)

The relation between the source clock of the 5-bit counter and the m value is shown in Figure 16-3.

Table 16-3. Relation between 5-Bit Counter Source Clock and m Value

TPSn2	TPSn1	TPSn0	5-Bit Counter Source Clock Selection	m
0	0	0	External clock I (ASCKn)	—
0	0	1	f_{XX} (12.5 MHz)	0
0	1	0	$f_{XX}/2$ (6.25 MHz)	1
0	1	1	$f_{XX}/4$ (3.13 MHz)	2
1	0	0	$f_{XX}/8$ (1.56 MHz)	3
1	0	1	$f_{XX}/16$ (781 kHz)	4
1	1	0	$f_{XX}/32$ (391 kHz)	5
1	1	1	TO1 (TM1 output)	—

Remarks 1. n = 1, 2
2. Figures in parentheses apply to operation with f_{XX} = 12.5 MHz.

- **Baud rate capacity error range**

  The baud rate capacity range depends on the number of bits per frame and the counter division ratio [1/(16+k)].

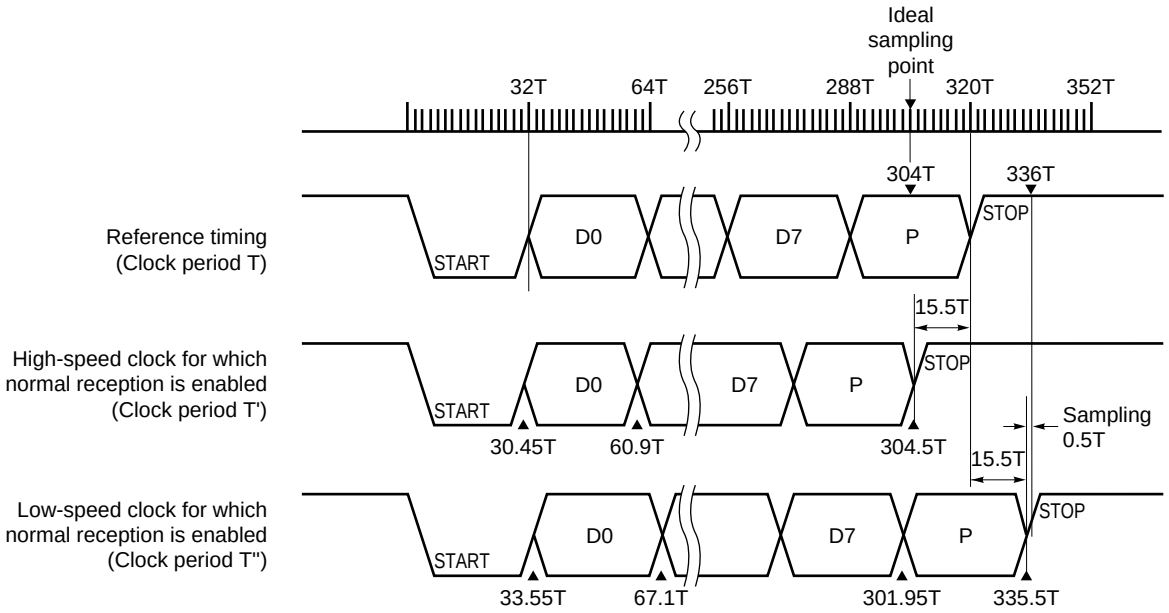
  Table 16-4 shows the relation between the main system clock and the baud rate, Table 16-6 shows a baud rate capacity error example.

Table 16-4. Relation between Main System Clock and Baud Rate

Baud Rate (bps)	f_{XX} = 12.5 MHz		f_{XX} = 6.25 MHz		f_{XX} = 3.00 MHz	
	BRGC value	Error (%)	BRGC value	Error (%)	BRGC value	Error (%)
2400	—	—	—	—	64H	2.80
4800	—	—	64H	1.73	54H	2.80
9600	64H	1.73	54H	1.73	44H	2.80
19200	54H	1.73	44H	1.73	34H	2.80
31250	49H	0.00	39H	0.00	29H	2.80
38400	44H	1.73	34H	1.73	24H	2.80
76800	34H	1.73	24H	1.73	14H	2.80
150K	24H	1.73	14H	1.73	—	—
300K	14H	1.73	—	—	—	—

Remark When TM1 output is used, 150 to 38400 bps is supported (during operation with f_{XX} = 12.5 MHz)

Figure 16-6. Baud Rate Capacity Error Considering Sampling Errors (When k = 0)



Remark T : 5-bit counter source clock period

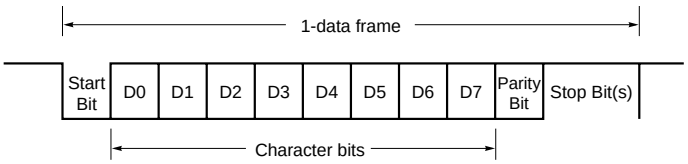
$$\text{Baud rate capacity error (k = 0)} \quad \frac{\pm 15.5}{320} \times 100 = 4.8438\ (\%)$$

(2) Communication operation

(a) Data format

The format for sending and receiving data is shown in Figure 16-7.

Figure 16-7. Asynchronous Serial Interface Transmit/Receive Data Format



Each data frame is composed for the bits outlined below.

- Start bit ..... 1 bit
- Character bits ..... 7 bits/8 bits
- Parity bit ..... Even parity/Odd parity/0 parity/No parity
- Stop bit(s) ..... 1 bit/2 bits

Specification of the character bit length inside data frames, selection of the parity, and selection of the stop bit length, are performed with the asynchronous serial interface mode register n (ASIMn).

f 7 bits has been selected as the number of character bits, only the low-order 7 bits (bits 0 to 6) are valid. In the case of transmission, the highest-order bit (bit 7) is ignored. In the case of reception, the highest-order bit (bit 7) always becomes “0”.

The setting of the serial transfer rate is performed with the ASIMn and the baud rate generator control register n (BRGCn).

If a serial data reception error occurs, it is possible to determine the contents of the reception error by reading the status of the asynchronous serial interface status register n (ASISn).

Remark n = 1, 2

(b) Parity types and operations

Parity bits serve to detect bit errors in transmit data. Normally, the parity bit used on the transmit side and the receive side are of the same type. In the case of even parity and odd parity, it is possible to detect “1” bit (odd number) errors. In the case of 0 parity and no parity, errors cannot be detected.

(i) Even parity

- **During transmission**

  Makes the number of “1”s in transmit data that includes the parity bit even. The value of the parity bit changes as follows.

  If the number of “1” bits in transmit data is odd : 1
  if the number of “1” bits in transmit data is even: 0

- **During reception**

  The number of “1” bits in receive data that includes the parity bit is counted, and if it is odd, a parity error occurs.

(ii) Odd parity

- **During transmission**

  Odd parity is the reverse of even parity. It makes the number of “1”s in transmit data that includes the parity bit even. The value of the parity bit changes as follows.

  If the number of “1” bits in transmit data is odd : 1
  if the number of “1” bits in transmit data is even: 0

- **During reception**

  The number of “1” bits in receive data is counted, and if it is even, a parity error occurs.

(iii) 0 Parity

During transmission, makes the parity bit “0”, regardless of the transmit data.

Parity bit check is not performed during reception. Therefore, no parity error occurs, regardless of whether the parity bit value is “0” or “1”.

(iv) No parity

No parity is appended to transmit data.

Transmit data is received assuming that it has no parity bit. No parity error can occur because there is no parity bit.

(c) Transmission

Transmission is begun by writing transmit data to the transmission shift register n (TXSn). The start bit, parity bit, and stop bit(s) are automatically added.

The contents of the transmit shift register n (TXSn) are shifted out upon transmission start, and when the transmit shift register n (TXSn) becomes empty, a transmit interrupt (INTSTn) is generated.

Cautions In the case of UART transmission, follow the procedure below when performing transmission for the first time.

<1> Set the port to the input mode (PM21 = 1 or PM71 = 1), and write 0 to the port latch.

<2> Write transmit data to TXS, and start transmission.

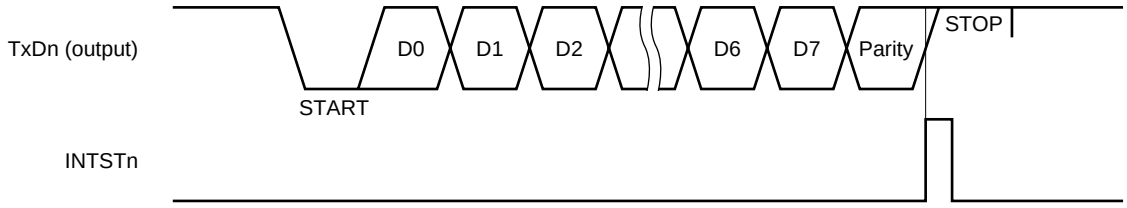
<3> Set the port to the output mode (PM21 = 0 or PM71 = 0).

If the port is set to the output mode first, 0 will be output from the pins, which may cause malfunction.

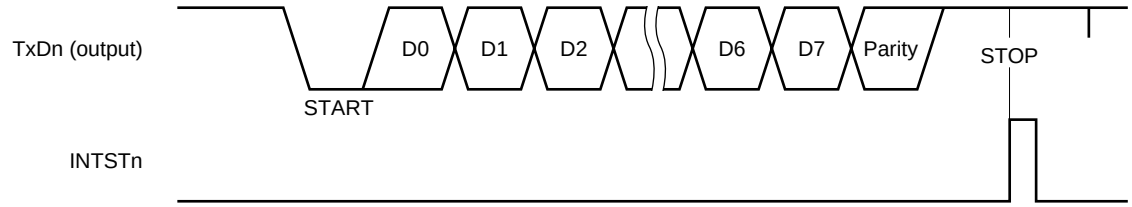
Remark n = 1, 2

Figure 16-8. Asynchronous Serial Interface Transmit Completion Interrupt Request Timing

(a) Stop bit length: 1



(b) Stop bit length: 2



Caution Do not write to the asynchronous serial interface mode register n (ASIMn) during transmission. If you write to the ASIMn register during transmission, further transmission operations may become impossible (in this case, input $\overline{\text{RESET}}$ to return to normal). Whether transmission is in progress or not can be judged by software, using the transmit completion interrupt (INTSTn) or the interrupt request flag (STIFn) set by INTSTn.

Remark n = 1, 2

(d) Reception

When the RXEn bit of the asynchronous serial interface mode register n (ASIMn) is set to 1, reception is enabled and sampling of the RxDn pin input is performed.

Sampling of the RxDn pin input is performed by the serial clock set in ASIMn.

The 5-bit counter for the baud rate generator will begin counting when the RxDn pin input reaches low level, and the 'start timing' signal for data sampling will be output when half of the time set for the baud rate has passed. If the result of re-sampling the RxDn pin input with this start timing signal is low level, the RxDn pin input is perceived as the start bit, the 5-bit counter is initialized and begins counting, and data sampling is performed. When, following the start bit, character data, the parity bit, and one stop bit are detected, reception of one frame of data is completed.

When reception of one frame of data is completed, the receive data in the shift register is transferred to the receive shift register (RXBn),and a receive completion interrupt (INTSRn) is generated.

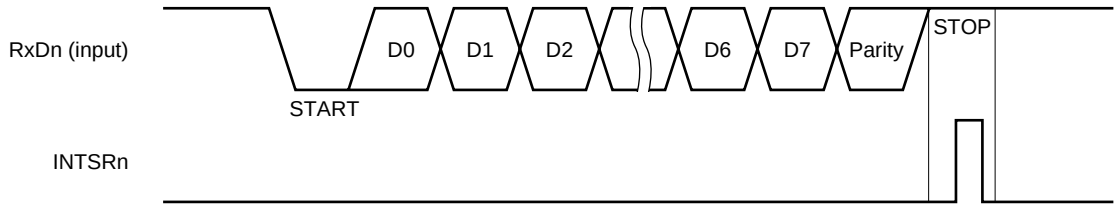
Also, even if an error occurs, the receiving data for which the error occurred is transferred to RXBn. If an error occurs, when bit 1 (ISRMn) of ASIMn is cleared (0), INTSRn is generated. (refer to **Figure 16-10**).

When bit ISRMn is set (1), INTSRn is not generated.

When bit RXEn is reset to 0 during a receive operation, the receive operation is immediately stopped. At this time, the contents of RXBn and ASISn remain unchanged, and INTSRn and INTSERn are not generated.

Remark n = 1, 2

Figure 16-9. Asynchronous Serial Interface Receive Completion Interrupt Request Timing



Caution Even when a receive error occurs, be sure to read the receive buffer register (RXBn). If RXBn is not read, an overrun error will occur during reception of the next data, and the reception error status will continue indefinitely.

Remark n = 1, 2

(e) Receive error

Errors that occur during reception are of three types: parity errors, framing errors, and overrun errors. As the data reception result error flag is set inside the asynchronous serial interface status register n (ASISn), the receive error interrupt request (INTSERn) is generated. A receive error interruption is generated before a receive end interrupt request (INTSRn). Receive error causes are shown in Table 16-5.

What type of error has occurred during reception can be detected by reading the contents of the asynchronous serial interface status register n (ASISn) during processing of the receive error interrupt (INTSERn) (rerer to **Table 16-5** and **Figure 16-10**).

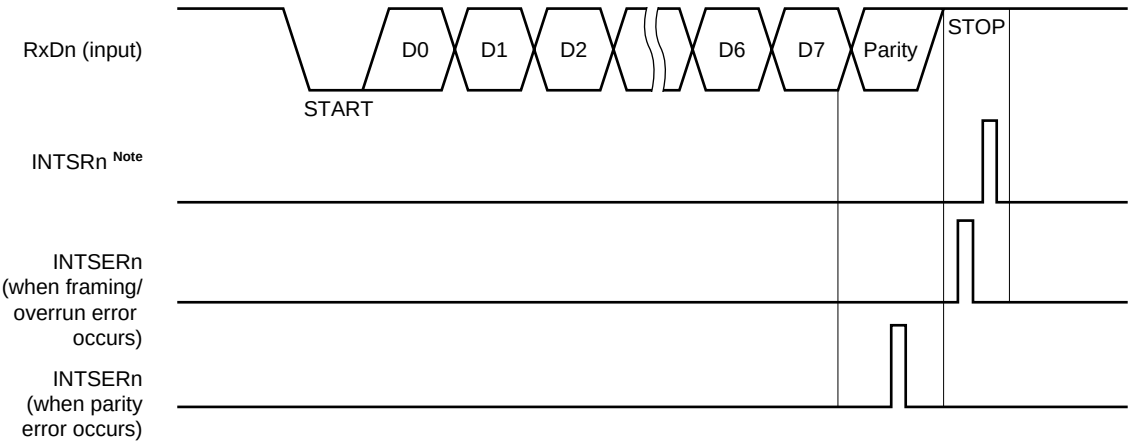
The contents of ASISn are reset to 0 either when the receive buffer register n (RXBn) is read or when the next data is received (If the next data has an error, this error flag is set).

Remark n = 1, 2

Table 16-5. Receive Error Causes

Receive Error	Cause	ASISn
Parity error	Parity specified for transmission and parity of receive data don't match	04H
Framing error	Stop bit was not detected	02H
Overrun error	Next data reception was completed before data was read from the receive buffer register	01H

Figure 16-10. Receive Error Timing



Note INTSRn will not be triggered if an error occurs when the ISRMn bit has been set (1).

Cautions 1. The contents of the ASISn register are reset to 0 either when the receive buffer register n (RXBn) is read or when the next data is received. To find out the contents of the error, be sure to read ASIS before reading RXBn.

2. Be sure to read the receive buffer register n (RXBn) even when a receive error occurs. If RXBn is not read, an overrun error will occur at reception of the next data, and the receive error status will continue indefinitely.

Remark n = 1, 2

16.3.3 Standby mode operation

(1) HALT mode operation

Serial transfer operation is normally performed.

(2) STOP mode or IDLE mode operation

(a) When internal clock is selected as serial clock

The asynchronous serial interface mode register n (ASIMn), transmit shift register n (TXSn), receive shift register n (RXn), and receive buffer register n (RXBn) stop operation holding the value immediately before the clock stops.

If the clock stops (STOP mode) during transmission, the TxDn pin output data immediately before the clock stopped is held. If the clock stops during reception, receive data up to immediately before the clock stopped is stored, and subsequent operation is stopped. When the clock is restarted, reception is resumed.

Remark n = 1, 2

(b) When external clock is selected as serial clock

Serial transmission is performed normally. However, interrupt requests are pended without being acknowledged. Interrupt requests are acknowledged after the STOP mode or IDLE mode has been released through NMI input, INTP0 to INTP5 input, and INTWT.

16.4 3-Wire Serial I/O Mode

This mode is used to perform 8-bit data transfer with the serial clock ($\overline{SCK1}$, $\overline{SCK2}$), serial output (SO1, SO2), and serial input (SI1, SI2) lines.

The 3-wire serial I/O mode supports simultaneous transmit/receive operation, thereby reducing the data transfer processing time.

The start bit of 8-bit data for serial transfer is fixed as the MSB.

The 3-wire serial I/O mode is effective when connecting a peripheral I/O with an on-chip clock synchronization serial interface, a display controller, etc.

16.4.1 Configuration

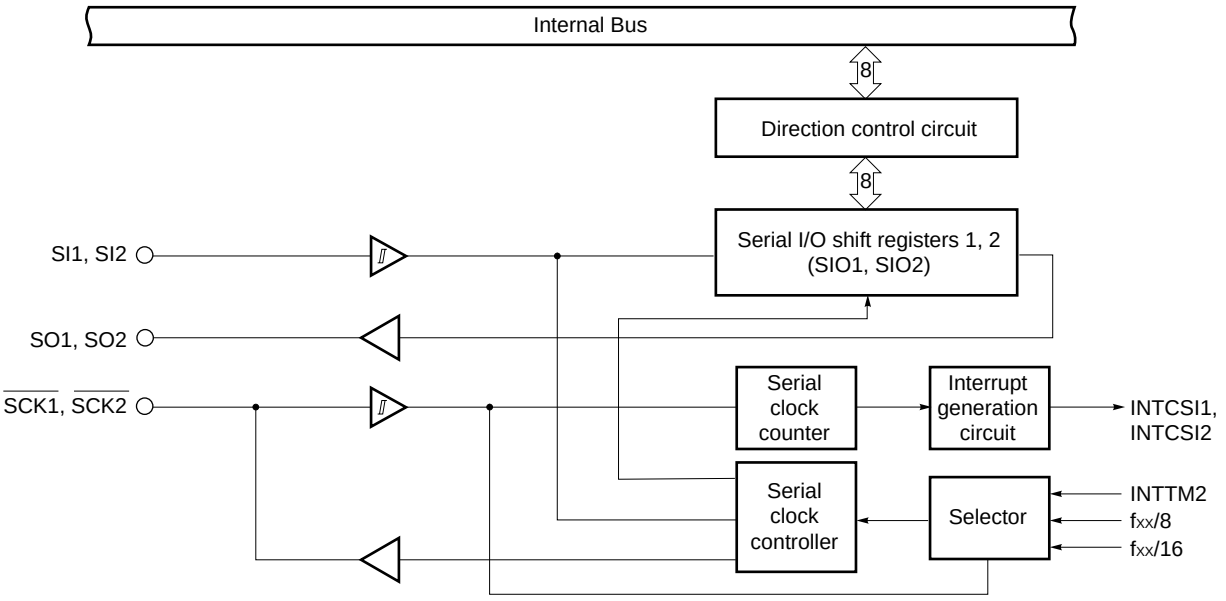
The 3-wire serial I/O mode has the following hardware configuration.

Figure 16-11 shows the block diagram for the 3-wire serial I/O mode.

Table 16-6. 3-Wire Serial I/O Configuration

Item	Configuration
Register	Serial I/O shift registers 1, 2 (SIO1, SIO2)
Control register	Serial operation mode registers 1, 2 (CSIM1, CSIM2)

Figure 16-11. Block Diagram in 3-Wire Serial I/O Mode



- **Serial I/O shift registers 1, 2 (SIO1, SIO2)**

  These are 8-bit registers that perform parallel-serial conversion, and serial transmission/reception (shift operation) in synchronization with the serial clock.

  SIOn is set with an 8-bit memory manipulation instruction.

  When bit 7 (CSIEn) of the serial operation mode register (CSIMn) is 1, serial operation can be started by writing/reading data to/from SIOn.

  During transmission, data written to SIOn is output to the serial output pin (SOn).

  During reception, data is read into SIOn from the serial input pin (SIn).

  $\overline{\text{RESET}}$ input sets SIO1 and SIO2 to 00H.

Caution During transfer operation, do not perform access to SIOn other than access acting as a transfer start trigger (read and write are prohibited when MODEn = 0 and MODEn = 1, respectively).

Remark n = 1, 2

16.4.2 Control registers

- **Serial operation mode registers 1, 2 (CSIM1, CSIM 2)**

  CSIM1 and CSIM2 are used to set the serial clock, operation mode, and operation enable/disable during the 3-wire serial I/O mode.

  CSIM1 and SCIM2 can be set by a 1-bit or 8-bit memory manipulation instruction.

  $\overline{\text{RESET}}$ input sets CSIM1 and CSIM2 to 00H.

Figure 16-12. Serial Operation Mode Registers 1, 2 (CSIM1, CSIM2) Format

Address: 0FF91H, 0FF92H After Reset: 00H R/W

Symbol	⑦	6	5	4	3	2	1	0
CSIMn	CSIEn	0	0	0	0	MODEn	SCLn1	SCLn0

CSIEn	SIOn Operation Enable/Disable Setting		
	Shift Register Operation	Serial Counter	Port
0	Operation disable	Clear	Port function[Note]
1	Operation enable	Counter operation enable	Serial function + port function

MODEn	Transfer Operation Mode Flag		
	Operation Mode	Transfer Start Trigger	SOn Output
0	Transmit/receive mode	SIOn write	Normal output
1	Receive only mode	SIOn read	Fix to low level

SCLn1	SCLn0	Clock Selection
0	0	External clock to $\overline{\text{SCKn}}$
0	1	8-bit timer/counter 2 (TM2) output
1	0	$f_{XX}/8$ (1.56 MHz)
1	1	$f_{XX}/16$ (781 kHz)

Note When CSIEn = 0 (SIOn operation stop status), pins connected to SIn, SOn and $\overline{\text{SCKn}}$ can be used as ports.

Remarks 1. n = 1, 2

2. Figures in parentheses apply to operation with f_{XX} = 12.5 MHz.

16.4.3 Operation

The following two types of 3-wire serial I/O operation mode are available.

- Operation stop mode
- 3-wire serial I/O mode

(1) Operation stop mode

Serial transfer is not possible in the operation stop mode, which reduces power consumption.
Moreover, in operation stop mode, pins can normally be used a I/O ports.

(a) Register setting

The operation stop mode is set by serial operation registers 1 and 2 (CSIM1, CSIM2).
CSIM1 and CSIM2 can be set by a 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input sets CSIM1 and CSIM2 to 00H.

Figure 16-13. Serial Operation Mode Registers 1, 2 (CSIM1, CSIM2) Format

Address: 0FF91H, 0FF92H After Reset: 00H R/W

Symbol	⑦	6	5	4	3	2	1	0
CSIMn	CSIEn	0	0	0	0	MODEn	SCLn1	SCLn0

CSIEn	SIOn Operation Enable/Disable Setting		
	Shift Register Operation	Serial Counter	Port
0	Operation disable	Clear	Port function[Note]
1	Operation enable	Counter operation enable	Serial function + port function

Note When CSIEn = 0 (SIOn operation stop status), pins connected to SIn, SOn and $\overline{\text{SCKn}}$ can be used as ports.

Remark n = 1, 2

(2) 3-wire serial I/O mode

The 3-wire serial I/O mode is effective when connecting a peripheral I/O with an on-chip clock synchronization serial interface, a display controller, etc.

This mode is used to perform communication with the serial clock ($\overline{SCK1}$, $\overline{SCK2}$), serial output (SO1, SO2), and serial input (SI1, SI2) lines.

(a) Register setting

The 3-wire serial I/O mode is set by serial operation mode registers 1 and 2 (CSIM1, CSIM2).

CSIM1 and CSIM2 can be set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{RESET}$ input sets CSIM1 and CSIM2 to 00H.

Figure 16-14. Serial Operation Mode Registers 1, 2 (CSIM1, CSIM2) Format

Address: 0FF91H, 0FF92H After Reset: 00H R/W

Symbol	⑦	6	5	4	3	2	1	0
CSIMn	CSIEn	0	0	0	0	MODEn	SCLn1	SCLn0

CSIEn	SIOn Operation Enable/Disable Setting		
	Shift Register Operation	Serial Counter	Port
0	Operation disable	Clear	Port function[Note]
1	Operation enable	Counter operation enable	Serial function + port function

MODEn	Transfer Operation Mode Flag		
	Operation Mode	Transfer Start Trigger	SOn Output
0	Transmit/receive mode	SIOn write	Normal output
1	Receive only mode	SIOn read	Fix to low level

SCLn1	SCLn0	Clock Selection
0	0	External clock to $\overline{SCKn}$
0	1	8-bit timer/counter 2 (TM2) output
1	0	f_{XX}/8 (1.56 MHz)
1	1	f_{XX}/16 (781 kHz)

Note When CSIEn = 0 (SIOn operation stop status), pins connected to SIn, SOn and $\overline{SCKn}$ can be used as ports.

Remarks 1. n = 1, 2

2. Figures in parentheses apply to operation with f_{XX} = 12.5 MHz.

(b) Normal operation

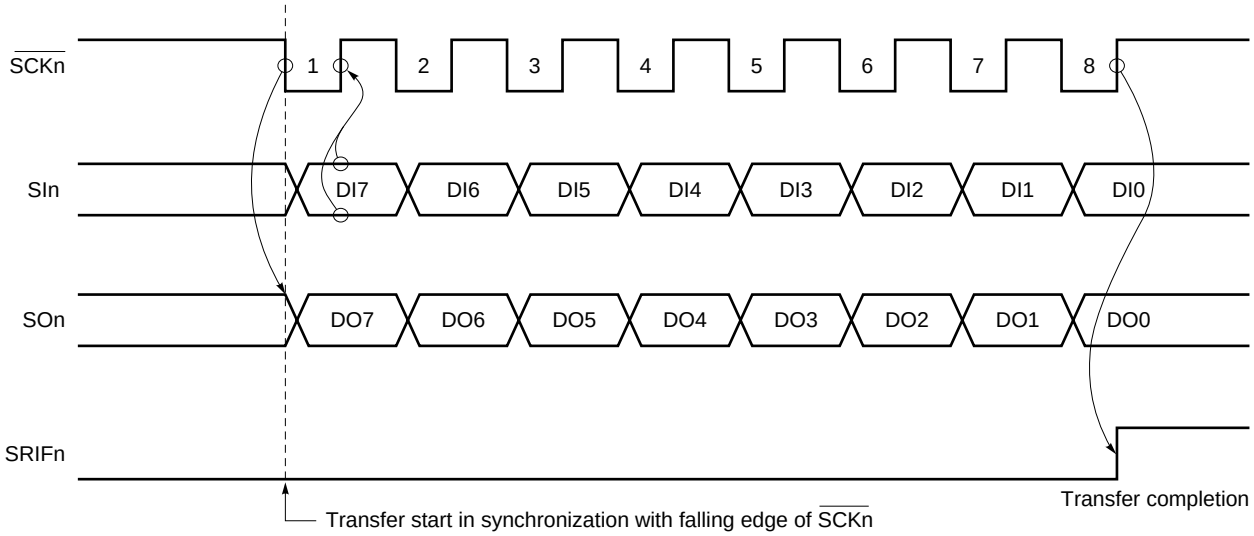
The 3-wire serial I/O mode performs data transfer in 8-byte units. Data is transmitted and received one byte at a time in synchronization with the serial clock.

The shift operation of the serial I/O shift register n (SIOn) is performed in synchronization with the falling edge of the serial clock ($\overline{SCKn}$). Transmit data is held in the SOn latch, and is output from the SOn pin. Receive data input to the SIn pin is latched to SIOn at the rising edge of the $\overline{SCKn}$ signal.

SIOn operation is automatically stopped when 8-bit transfer ends, and an interrupt request flag (SRIFn) is set.

Remark n = 1, 2

Figure 16-15. 3-Wire Serial I/O Mode Timing



Remark n = 1, 2

(c) Transfer start

Serial transfer is started by setting transmit data (or reading) to the serial I/O shift register n (SIOn) when the following two conditions are satisfied.

- SIOn operation control bit (CSIEn) = 1
- Following 8-bit serial transfer, the internal serial clock is stopped, or $\overline{SCKn}$ is high level
- Transmit/receive mode
  When CSIEn = 1 and MODEn = 0, and transfer is started with SIOn write
- Receive-only mode
  When CSIEn = 1 and MODEn = 1, and transfer is started with SIOn read.

Caution After data is written to SIOn, transfer will not start even if CSIEn is set to "1".

Serial transfer automatically stops at the end of 8-bit transfer, and the interrupt request flag (SRIFn) is set.

Remark n = 1, 2

CHAPTER 17 3-WIRE SERIAL I/O MODE

17.1 Function

This mode transfers 8-bit data by using the three lines of the serial clock ($\overline{SCK0}$), the serial output (SO0), and the serial input (SI0).

Since the 3-wire serial I/O mode can perform simultaneous transmission and reception, the data transfer processing time becomes shorter.

The starting bit of the 8-bit data to be serially transferred is fixed at the MSB.

The 3-wire serial I/O mode is valid when the peripheral I/O or display controller equipped with a clock-synchronized serial interface is connected.

17.2 Structure

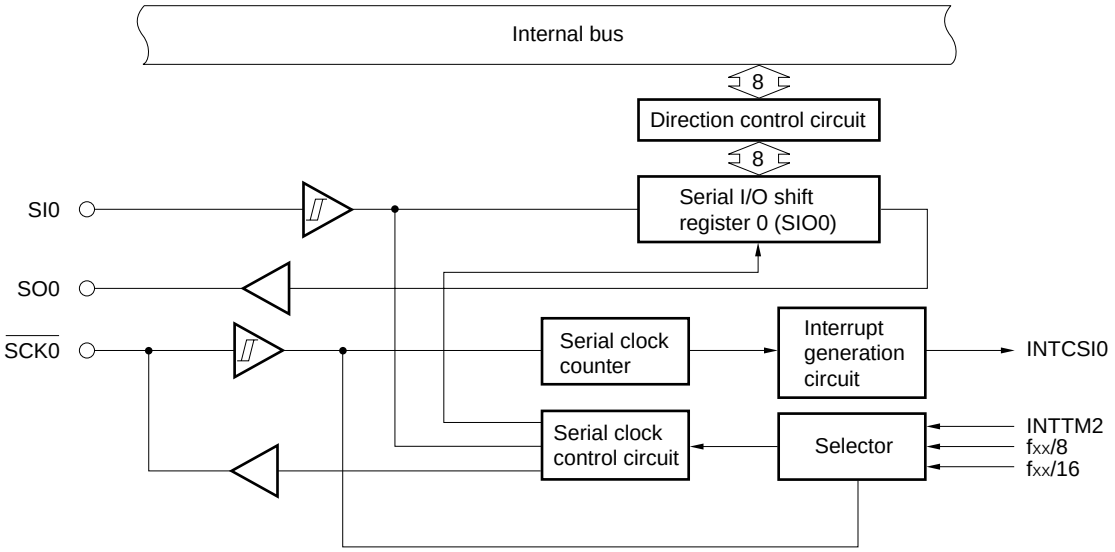
The 3-wire serial I/O mode is installed in the following hardware.

Figure 17-1 is a block diagram of the clock-synchronized serial interface (CSI) in the 3-wire serial I/O mode.

Table 17-1. 3-Wire Serial I/O Structure

Item	Structure
Register	Serial I/O shift register 0 (SIO0)
Control register	Serial operating mode register 0 (CSIM0)

Figure 17-1. Block Diagram of the Clock-Synchronized Serial Interface (in the 3-wire Serial I/O Mode)



- **Serial I/O shift register 0 (SIO0)**

  This 8-bit shift register performs parallel to serial conversion and serially communication (shift operation) synchronized to the serial clock.

  SIO0 is set by an 8-bit memory manipulation instruction.

  When bit 7 (CSIE0) in serial operating mode register 0 (CSIM0) is one, serial operation starts by writing data to or reading it from SIO0.

  When transmitting, the data written to SIO0 is output to the serial output (SO0).

  When receiving, data is read from the serial input (SI0) to SIO0.

  $\overline{\text{RESET}}$ input sets SIO0 to 00H.

Caution Do not access SIO0 during a transfer except for an access that becomes a transfer start trigger. (When MODE0 = 0, reading is disabled; and when MODE0 = 1, writing is disabled.)

17.3 Control Registers

- **Serial operating mode register 0 (CSIM0)**

  The CSIM0 register sets the serial clock and operating mode to the 3-wire serial I/O mode, and enables or stops operation.

  CSIM0 is set by a 1-bit or 8-bit memory manipulation instruction.

  $\overline{\text{RESET}}$ input sets CSIM0 to 00H.

Figure 17-2. Serial Operating Mode Register 0 (CSIM0) Format

Address: 0FF90H After Reset: 00H R/W

Symbol	(7)	6	5	4	3	2	1	0
CSIM0	CSIE0	0	0	0	0	MODE0	SCL01	SCL00

CSIE0	SIO0 Operation Enable/Disable Setting		
	Shift Register Operation	Serial Counter	Port
0	Disable operation	Clear	Port function[Note]
1	Enable operation	Enable count	Serial function + Port function

MODE0	Transfer Operation Mode Flag		
	Operation Mode	Transfer Start Trigger	SO0 Output
0	Transmit/receive communication mode	SIO0 write	Normal output
1	Receive only mode	SIO0 read	Fixed low

SCL01	SCL00	Clock Selection
0	0	External clock to $\overline{\text{SCK0}}$
0	1	8-bit timer/counter 2 (TM2) output
1	0	f_{XX}/8 (1.56 MHz)
1	1	f_{XX}/16 (781 KHz)

Note If CSIE0 = 0 (SIO0 operation stopped state), the pins connected to SI0, SO0 and $\overline{\text{SCK0}}$ can function as ports.

Remark Figures in parentheses apply to operation with f_{XX} = 12.5 MHz.

17.4 Operation

3-wire serial I/O has the following two operating modes.

- Operation stopped mode
- 3-wire serial I/O mode

(1) Operation stopped mode

Since serial transfers are not performed in the operation stopped mode, power consumption can be decreased. In the operation stopped mode, the pin can be used as an ordinary I/O port.

(a) Register settings

The operation stopped mode is set in serial operating mode register 0 (CSIM0).

CSIM0 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets CSIM0 to 00H.

Figure 17-3. Serial Operating Mode Register 0 (CSIM0) Format

Address: 0FF90H After Reset: 00H R/W

Symbol	⑦	6	5	4	3	2	1	0
CSIM0	CSIE0	0	0	0	0	MODE0	SCL01	SCL00

CSIE0	SIO0 Operating Enable/Disable Setting		
	Shift Register Operation	Serial Counter	Port
0	Disable operation	Clear	Port function[Note]
1	Enable operation	Enable operation count	Serial function + port function

Note If CSIE0 = 0 (SIO0 operation stopped state), the pins connected to SI0, SO0 and $\overline{\text{SCK0}}$ can function as ports.

(2) 3-wire serial I/O mode

The 3-wire serial I/O mode is valid when connected to peripheral I/O or a display controller equipped with the clock-synchronized serial interface.

Communication is over three lines, the serial clock ($\overline{SCK0}$), serial output (SO0), and serial input (SI0).

(a) Register setting

The 3-wire serial I/O mode is set in serial operating mode register 0 (CSIM0).

CSIM0 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{RESET}$ input sets CSIM0 to 00H.

Figure 17-4. Serial Operating Mode Register 0 (CSIM0) Format

Address: 0FF90H After Reset: 00H R/W

Symbol	⑦	6	5	4	3	2	1	0
CSIM0	CSIE0	0	0	0	0	MODE0	SCL01	SCL00

CSIE0	SIO0 Operation Enable/Disable Setting		
	Shift Register Operation	Serial Counter	Port
0	Disable operation	Clear	Port function**Note**
1	Enable operation	Enable operation count	Serial function + port function

MODE0	Transfer Operation Mode Flag		
	Operation Mode	Transfer Start Trigger	SO0 Output
0	Transmit/receive communication mode	SIO0 write	Normal output
1	Receive only mode	SIO0 read	Low level fixed

SCL01	SCL00	Clock Selection
0	0	External clock to $\overline{SCK0}$
0	1	8-bit timer/counter 2 (TM2) output
1	0	f_{XX}/8 (1.56 MHz)
1	1	f_{XX}/16 (781 KHz)

Note If CSIE0 = 0 (SIO0 operation stopped state), the pins connected to SI0, SO0 and $\overline{SCK0}$ can function as ports.

Remark Figures in parentheses apply to operation with f_{XX} = 12.5 MHz.

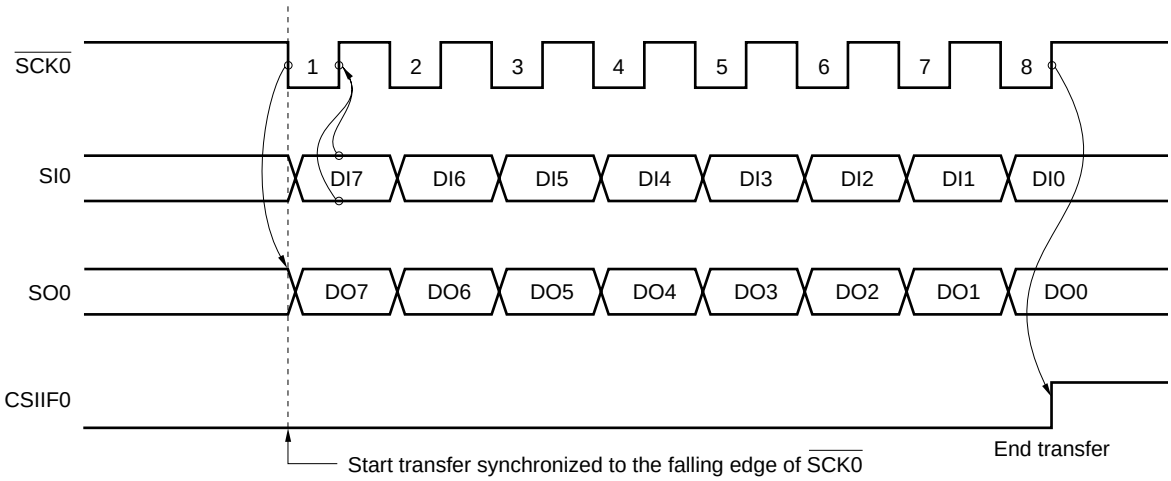
(b) Communication

The 3-wire serial I/O mode transmits and receives in 8-bit units. Data is transmitted and received with each bit synchronized to the serial clock.

The shifting of the serial I/O shift register 0 (SIO0) is synchronized to the falling edge of the serial clock ($\overline{SCK0}$). The transmitted data are held in the latch and output from the SO0 pin. At the rising edge of $\overline{SCK0}$, the received data that was input at the SI0 pin is latched to SIO0.

The end of the 8-bit transfer automatically stops SIO0 operation and sets the interrupt request flag (CSIIF0).

Figure 17-5. 3-Wire Serial I/O Mode Timing



(c) Start transfer

If the following two conditions are satisfied, the serial transfer starts when the transfer data is set in the serial I/O shift register (SIO0).

- Control bit (CSIE0) = 1 during SIO0 operation
- After an 8-bit serial transfer, the internal serial clock enters the stopped state or $\overline{SCK0}$ is high.
- Transmit/receive communication mode
  When CSIE0 = 1 and MODE0 = 0, the transfer starts with an SIO0 write.
- Receive only mode
  When CSIE0 = 1 and MODE0 = 1, the transfer starts with an SIO0 read.

Caution Even if CSIE0 becomes one after the data is written to SIO0, transfer does not start.

Serial transfer is automatically stopped by the end of the 8-bit transfer, and the interrupt request flag (CSIIF0) is set.

CHAPTER 18 I^2C BUS MODE (ONLY μPD784225Y SUBSERIES)

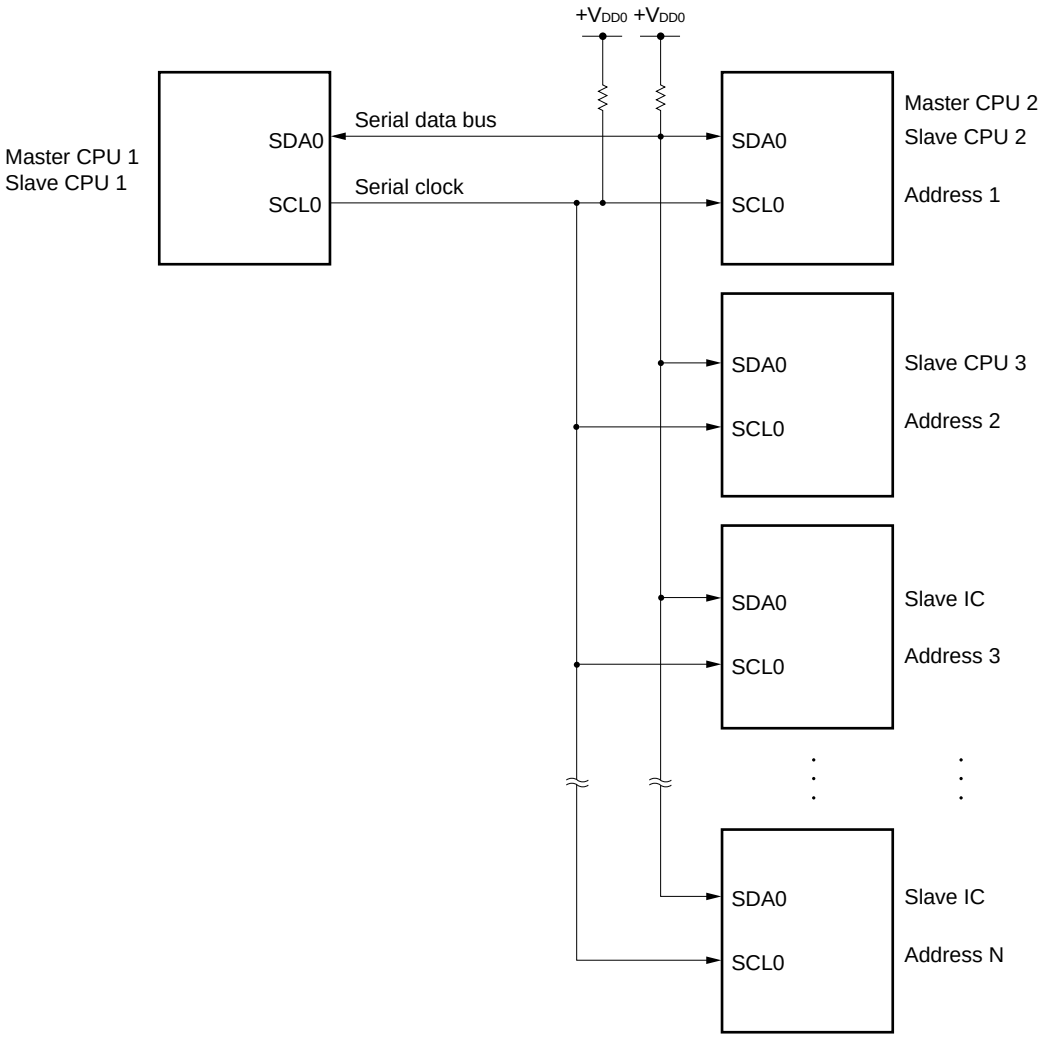
18.1 Function Overview

- **I^2C (Inter IC) bus mode (multi-master compatible)**
  This interface communicates with devices that conform to the I^2C bus format.
  Eight bit data transfers with multiple devices are performed by the two lines of the serial clock (SCL0) and the serial data bus (SDA0).
  In the I^2C bus mode, the master can output the start condition, data, and stop condition on the serial data bus to the slaves.
  The slaves automatically detect the received data by hardware. The I^2C bus control portion of the application program can be simplified by using this function.
  Since SCL0 and SDA0 become open drain outputs in the I^2C bus mode, pullup resistors are required on the serial clock line and serial data bus line.

Cautions 1. If the power to the μPD784225Y is disconnected while μPD784225Y functions are not used, the problem is I^2C communication will no longer be possible. Even when not used, do not disconnect the power to the μPD784225Y.

2. If the I^2C bus mode is used, set the SCL0/P27 and SDA0/P25 pins to N-channel open-drains by setting the port function control register (PF2).

Figure 18-1. Serial Bus Configuration Example in I²C Bus Mode



18.2 Structure

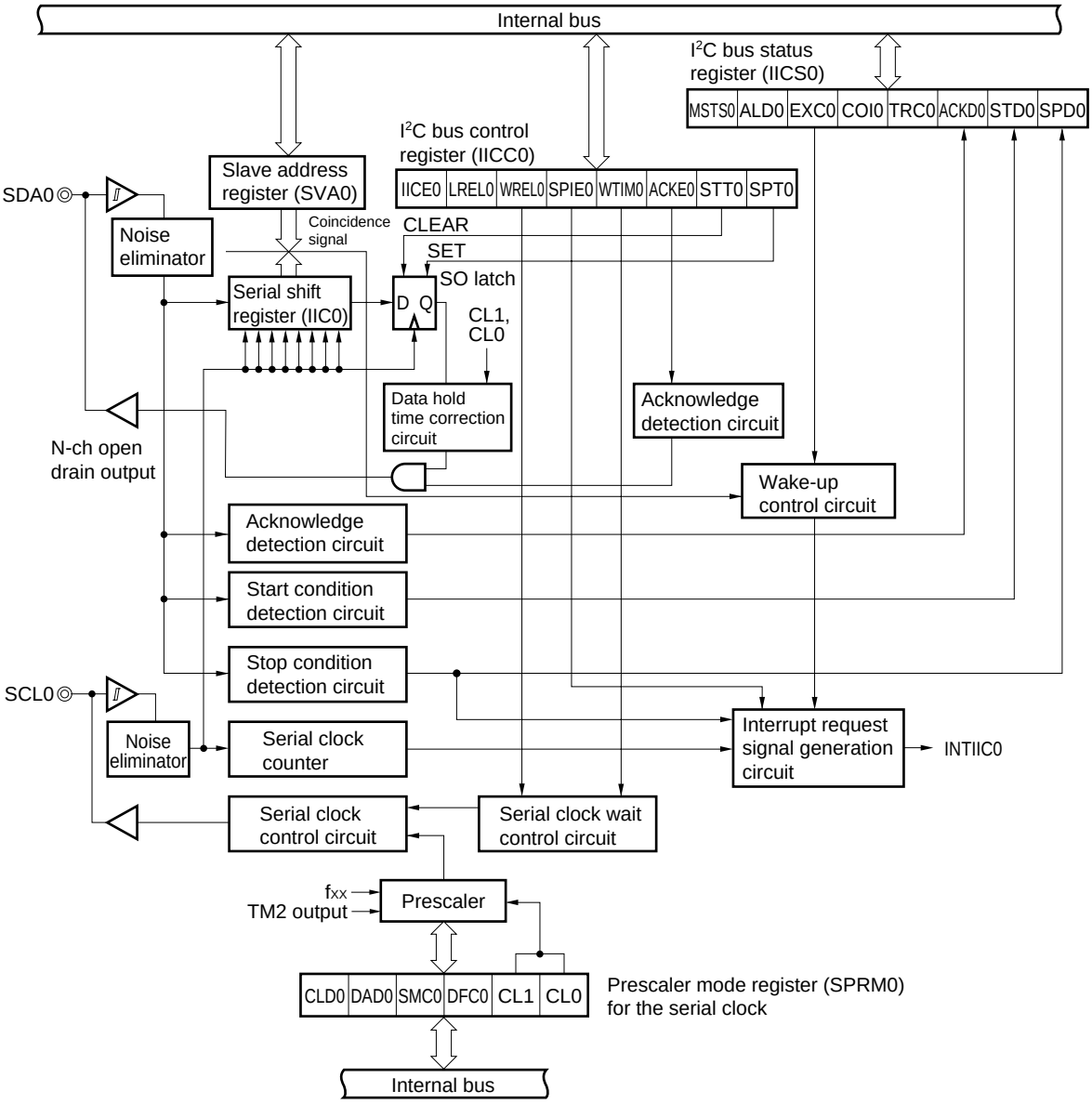
The clock-synchronized serial interface in the I²C bus mode consists of the following hardware.

Figure 18-2 is a block diagram of clock-synchronized serial interface (CSI) in the I²C bus mode.

Table 18-1. I²C Bus Mode Structure

Item	Structure
Registers	Serial shift register (IIC0) Slave address register (SVA0)
Control registers	I²C bus control register (IICC0) I²C bus status register (IICS0) Prescaler mode register (SPRM0) for the serial clock

Figure 18-2. Block Diagram of Clock-synchronized Serial Interface (I²C Bus Mode)



(1) Serial shift register (IIC0)

The IIC0 register converts 8-bit serial data into 8-bit parallel data and 8-bit parallel data into 8-bit serial data. IIC0 is used in both transmission and reception.

The actual transmission and reception are controlled by writing and reading IIC0.

IIC0 is set by an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets IIC0 to 00H.

(2) Slave address register (SVA0)

When used as a slave, this register sets a slave address.

SVA0 is set by an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets SVA0 to 00H.

(3) SO latch

The SO latch holds the output level of the SDA0 pin.

(4) Wake-up control circuit

This circuit generates an interrupt request when the address set in the slave address register (SVA0) and the reception address matched, or when an extended code was received.

(5) Clock selector

This selects the sampling clock that is used.

(6) Serial clock counter

The serial clock that is output or input during transmission or reception is counted to check 8-bit data communication.

(7) Interrupt request signal generation circuit

This circuit controls the generation of the interrupt request signal (INTIIC0).

The I²C interrupt request generates the following two triggers.

- Eighth or ninth clock of the serial clock (set by the WTIM0 bit**Note**)
- Interrupt request generated by detecting the stop condition (set by the SPIE0 bit**Note**)

Note WTIM0 bit: Bit 3 in the I²C bus control register (IICC0)
SPIE0 bit : Bit 4 in the I²C bus control register (IICC0)

(8) Serial clock control circuit

In the master mode, the clock output to pin SCL0 is generated by the sampling clock.

(9) Serial clock wait control circuit

This circuit controls the wait timing.

(10) Acknowledge output circuit, stop condition detection circuit, start condition detection circuit, acknowledge detection circuit

These circuits output and detect the control signals.

(11) Data hold time correction circuit.

This circuit generates the hold time of the data to the falling edge of the serial clock.

18.3 Control Registers

The I²C bus mode is controlled by the following three registers.

- I²C bus control register (IICC0)
- I²C bus status register (IICS0)
- Prescaler mode register (SPRM0) for the serial clock

The following registers are also used.

- Serial shift register (IIC0)
- Slave address register (SVA0)

(1) I²C bus control register (IICC0)

The IICC0 register enables and disables the I²C bus mode, sets the wait timing, and sets other I²C bus mode operations.

IICC0 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets IICC0 to 00H.

Figure 18-3. I²C Bus Control Register (IICC0) Format (1/4)

Address: 0FFB0H After Reset: 00H R/W

Symbol	⑦	⑥	⑤	④	③	②	①	⓪
IICC0	IICE0	LREL0	WREL0	SPIE0	WTIM0	ACKE0	STT0	SPT0

IICE0	I²C Operation Enabled	
0	Disables operation. Preset the expansion register (IICS0). Stops internal operation.	
1	Enables operation.	
• Clear condition (IICE0 = 0)		• Set condition (IICE0 = 1)
• Cleared by an instruction • When $\overline{\text{RESET}}$ is input		• Set by an instruction

LREL0	Release Communication	
0	Normal operation	
1	Releases microcontroller from the current communication and sets it in the wait state. Automatically clears after execution. The extended code that is unrelated to the base is used during reception. The SCL0 and SDA0 lines are put in the high impedance state. The following flags are cleared. • STD0 • ACKD0 • TRC0 • COI0 • EXC0 • MSTS0	
Until the following communication participation conditions are satisfied, the wait state that was released from communication is entered. • Start as the master after detecting the stop condition. • Address match or extended code reception after the start condition		
Clear condition (LREL0 = 0)[Note]		Set condition (LREL0 = 1)
• Automatically cleared after execution. • When $\overline{\text{RESET}}$ is input		• Set by an instruction

WREL0	Wait Release	
0	The wait is not released.	
1	The wait is released. After the wait is released, it is automatically cleared.	
Clear condition (WREL0 = 0)[Note]		Set condition (WREL0 = 1)
• Automatically cleared after execution. • When $\overline{\text{RESET}}$ is input		• Set by an instruction

SPIE0	Enable/Disable the Generation of Interrupt Requests by Stop Condition Detection	
0	Disable	
1	Enable	
Clear condition (SPIE0 = 0)[Note]		Set condition (SPIE0 = 1)
• Cleared by an instruction • When $\overline{\text{RESET}}$ is input		• Set by an instruction

Note By setting IICE0 = 0, this flag signal becomes invalid.

Figure 18-3. I²C Bus Control Register (IICC0) Format (2/4)

WTIM0	Control of Wait and Interrupt Request Generation
0	Interrupt request generated at the falling edge of the eighth clock For the master: After the eighth clock is output, wait with the clock output low. For the slave : After the eighth clock is input, the master waits with the clock low.
1	Interrupt request generated at the falling edge of the ninth clock For the master: After the ninth clock is output, wait with the clock output low. For the slave : After the ninth clock is input, the master waits with the clock low.
This bit setting becomes invalid during an address transfer, and becomes valid after the transfer ends. In the master, a wait is inserted at the falling edge of the ninth clock in an address transfer. The slave that received the base address inserts a wait at the falling edge of the ninth clock after the acknowledge is generated. The slave that received the extended code inserts the waits at the falling edge of the eighth clock.	
Clear condition (WTIM0 = 0)**Note**	Set condition (WTIM0 = 1)
• Cleared by an instruction • When $\overline{\text{RESET}}$ is input	• Set by an instruction

ACKE0	Acknowledge Control
0	Acknowledge is disabled.
1	Acknowledge is enabled. The SDA0 line during the ninth clock period goes low. However, the control is invalid during an address transfer. When EXC0 = 1, the control is valid.
Clear condition (ACKE0 = 0)**Note**	Set condition (ACKE0 = 1)
• Cleared by an instruction • When $\overline{\text{RESET}}$ is input	Set by an instruction

Note IICE0 = 0 makes this flag signal invalid.

Figure 18-3. I²C Bus Control Register (IICC0) Format (3/4)

STT0	Start Condition Trigger
0	The start condition is not generated.
1	When the bus is released (slave condition): The start condition is generated (started as the master). The SDA0 line is changed from high to low, and the start condition is generated. Then, the standard time is guaranteed, and SCL0 goes low. When not participating with the bus: The trigger functions as the start condition reserved flag. When set, the start condition is automatically generated after the bus is released.

Cautions on set timing

- Master reception : Setting is prohibited during transfer. STT0 can be set only during the wait period after ACKE0 = 0 is set and the fact that reception is completed is passed to the slave.
- Master transmission : During the ACK0 acknowledge period, the start condition may not be normally generated. Set STT0 during the wait period.
- Setting synchronized to SPT0 is prohibited.

Clear condition (STT0 = 0)[Note]	Set condition (STT0 = 1)
• Cleared by an instruction • When arbitration failed • Clear after generating the start condition in the master. • When $\overline{\text{RESET}}$ is input	• Set by an instruction

Note IICE0 = 0 makes this flag signal invalid.

Figure 18-3. I²C Bus Control Register (IICC0) Format (4/4)

SPT0	Stop Condition Trigger
0	The stop condition is not generated.
1	The stop condition is generated (ends the transfer as the master). After the SDA0 line goes low, the SCL0 line goes high, or wait until SCL0 goes high. Then, the standard time is guaranteed; the SDA0 line is changed from low to high; and the stop condition is generated.

Cautions on set timing

- Master reception : Setting is prohibited during transfer.
  SPT0 can be set only during the wait period after ACK0=0 is set and the fact that reception is completed is passed to the slave.
- Master transmission : During the ACK0 acknowledge period, the start condition may not be normally generated. Set SPT0 during the wait period.
- Setting synchronized to STT0 is prohibited.
- Set SPT0 only by the master.[Note 1]
- When WTIM0 = 0 is set, be aware that if SPT0 is set during the wait period after the eighth clock is output, the stop condition is generated during the high level of the ninth clock after the wait is released.
  When the ninth clock must be output, set WTIM0 = 0 → 1 during the wait period after the eighth clock is output, and set SPT0 during the wait period after the ninth clock is output.

Clear condition (SPT0 = 0)[Note 2]	Set condition (SPT0 = 1)
• Cleared by an instruction • When arbitration failed • Automatically clear after the stop condition is detected • When $\overline{\text{RESET}}$ is input	• Set by an instruction

Notes 1. Set SPT0 only by the master. However, SPT0 must be set once, and the stop condition generated while the master is operating until the first stop condition is detected after operation is enabled. For details, refer to **18.5.15 Additional warnings**.

2. IICE0 = 0 makes this flag signal invalid.

Caution When bit 3 (TRC0) = 1 in the I²C bus status register (IICS0), after WREL0 is set at the ninth clock and the wait is released, TRC0 is cleared, and the SDA0 line has a high impedance.

Remark STD0 : Bit 1 in I²C bus status register (IICS0)
ACKD0 : Bit 2 in I²C bus status register (IICS0)
TRC0 : Bit 3 in I²C bus status register (IICS0)
COI0 : Bit 4 in I²C bus status register (IICS0)
EXC0 : Bit 5 in I²C bus status register (IICS0)
MSTS0 : Bit 7 in I²C bus status register (IICS0)

(2) I²C bus status register (IICS0)

The IICS0 register displays the status of the I²C bus.

IICS0 is set by a 1-bit or 8-bit memory manipulation instruction. IICS0 can only be read.

$\overline{\text{RESET}}$ input sets IICS0 to 00H.

Figure 18-4. I²C Bus Status Register (IICS0) Format (1/3)

Address: 0FFB6H After Reset: 00H R

Symbol	⑦	⑥	⑤	④	③	②	①	⓪
IICS0	MSTS0	ALD0	EXC0	COI0	TRC0	ACKD0	STD0	SPD0

MSTS0	Master State
0	Slave state or communication wait state
1	Master communication state

Clear condition (MSTS0 = 0)	Set condition (MSTS0 = 1)
• When the stop condition is detected • When ALD0 = 1 • Cleared by LREL0 = 1 • When $\overline{\text{IICE0}}$ = 1 → 0 • When $\overline{\text{RESET}}$ is input	• When start condition is generated

ALD0	Arbitration Failed Detection
0	No arbitration state or arbitration win state
1	Arbitration failed state. MSTS0 is cleared.

Clear condition (ALD0 = 0)	Set condition (ALD0 = 1)
• Automatically cleared after IICS0 is read[Note] • When $\overline{\text{IICE0}}$ = 1 → 0 • When $\overline{\text{RESET}}$ is input	• When arbitration failed

EXC0	Extended Code Reception Detection
0	The extended code is not received.
1	The extended code is received.

Clear condition (EXC0 = 0)	Set condition (EXC0 = 1)
• When the start condition is detected • When the stop condition is detected • Cleared by LREL0 = 1 • When $\overline{\text{IICE0}}$ = 1 → 0 • When $\overline{\text{RESET}}$ is input	• When the most significant four bits of the received address data are 0000 or 1111 (set by the rising edge of the eighth clock)

Note This is cleared when a bit manipulation instruction is executed for a bit not in IICS0.

Figure 18-4. I²C Bus Status Register (IICS0) Format (2/3)

COI0	Address Coincidence Detection
0	The address does not match.
1	The address matches.

Clear condition (COI0 = 0)	Set condition (COI0 = 1)
• During start condition detection • During stop condition detection • Cleared by LREL0 = 1 • When IICE0 = 1 → 0 • When $\overline{\text{RESET}}$ is input	• When the received address matches the base address (SVA0) (set at the rising edge of the eighth clock)

TRC0	Transmission/Reception State Detection
0	Reception state (not the transmission state). The SDA0 line has high impedance.
1	Transmission state. The value in the SO latch can be output to the SDA0 line (valid after the falling edge of the ninth clock of the first byte)

Clear condition (TRC0 = 0)	Set condition (TRC0 = 1)
• When the stop condition is detected • Cleared by LREL0 = 1 • When IICE0 = 1 → 0 • Cleared by WREL0 = 1 • When ALD0 = 0 → 1 • When $\overline{\text{RESET}}$ is input In the master • When one is output to the first byte LSB (transfer direction specification bit) In the slave • When the start condition is detected When not participating in the communication	In the master • When the start condition is generated In the slave • When one is input to the LSB of the first byte (transfer direction specification bit)

ACKD0	Acknowledge Detection
0	The acknowledge is not detected.
1	The acknowledge is detected.

Clear condition (ACKD0 = 0)	Set condition (ACKD0 = 1)
• When the stop condition is detected • At the rising edge of the first clock in the next byte • Cleared by LREL0 = 1 • When IICE0 = 1 → 0 • When $\overline{\text{RESET}}$ is input	When the SDA0 line is low at the rising edge of the ninth clock of SCL0

Figure 18-4. I²C Bus Status Register (IICS0) Format (3/3)

STD0	Start Condition Detection
0	The start condition is not detected.
1	The start condition is detected. This indicates the address transfer period.

Clear condition (STD0 = 0)	Set condition (STD0 = 1)
• When the stop condition is detected • At the rising edge of the first clock of the next byte after transferring the address • Cleared by LREL0 = 1 • When $\overline{\text{IICE0}}$ = 1 → 0 • When $\overline{\text{RESET}}$ is input	• When the start condition is detected

SPD0	Stop Condition Detection
0	The stop condition is not detected.
1	The stop condition is detected. Communication is ended by the master, and the bus is released.

Clear condition (SPD0 = 0)	Set condition (SPD0 = 1)
• After the bit is set, at the rising edge of the first clock in the address transfer byte after detecting the start condition • When $\overline{\text{IICE0}}$ = 1 → 0 • When $\overline{\text{RESET}}$ is input	• When the stop condition is detected

Remark LREL0: Bit 6 of I²C bus control register (IICC0)
IICE0 : Bit 7 of I²C bus control register (IICC0)

(3) Prescaler mode register (SPRM0) for the serial clock

The SPRM0 register sets the transfer clock of the I²C bus.

SPRM0 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets SPRM0 to 00H.

Figure 18-5. Format of the Prescaler Mode Register (SPRM0) for Serial Clock

Address: 0FFB2H After Reset: 00H R/W[Note]

Symbol	7	6	⑤	④	3	2	1	0
SPRM0	0	0	CLD	DAD	SMC	DFC	CL1	CL0

CLD	SCL0 Line Level Detection (valid only when IICE0 = 1)
0	Detects a low SCL0 line.
1	Detects a high SCL0 line.

Clear condition (CLD = 0)	Set condition (CLD = 1)
• When the SCL0 line is low • When $\overline{\text{IICE0}}$ = 0 • When $\overline{\text{RESET}}$ is input	• When the SCL0 line is high

DAD	SDA0 Line Level Detection (valid only when IICE0 = 1)
0	Detects a low SDA0 line.
1	Detects a high SDA0 line.

Clear condition (DAD = 0)	Set condition (DAD = 1)
• When the SDA0 line is low • When $\overline{\text{IICE0}}$ = 0 • When $\overline{\text{RESET}}$ is input	• When the SDA0 line is high

SMC	DFC	CL1	CL0	Transfer Clock
0	0	0	0	fxx/44
0	0	0	1	fxx/86
0	0	1	0	fxx/172
0	0	1	1	TM2 output/66
0	1	0	0	fxx/46
0	1	0	1	fxx/88
0	1	1	0	fxx/176
0	1	1	1	TM2 output/68
1	×	0	×	fxx/24
1	×	1	0	fxx/48
1	×	1	1	TM2 output/18

Note Bits 4 and 5 are read only.

Remarks 1. IICE0: Bit 7 of the I²C bus control register (IICC0)

2. X: don't care

(4) Serial shift register (IIC0)

This register performs serial communication (shift operation) synchronized to the serial clock.

Although this register can be read and written in 1-bit and 8-bit units, do not write data to IIC0 during a data transfer.

Address: 0FFB8H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
IIC0								

(5) Slave address register (SVA0)

This register stores the slave address of the I^2C bus.

It can be read and written in 8-bit units, but bit 0 is fixed at zero.

Address: 0FFB4H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
SVA0								0

18.4 I²C Bus Mode Function

18.4.1 Pin configuration

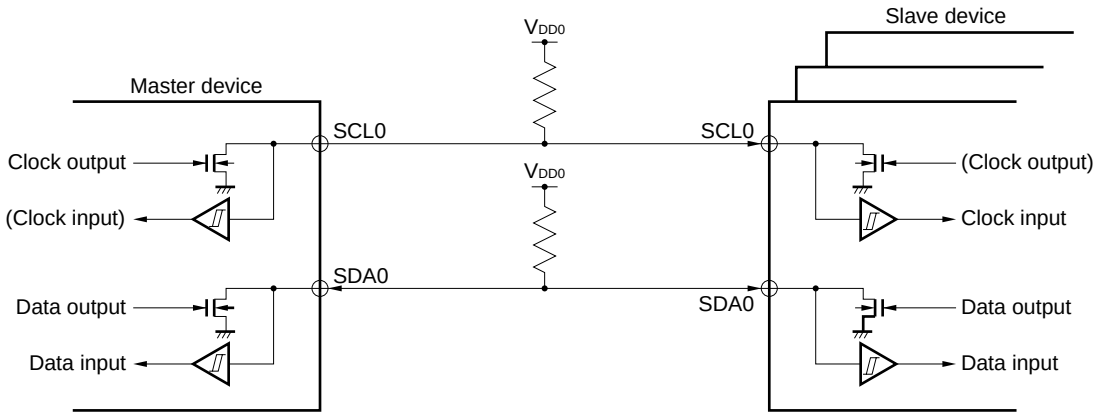
The serial clock pin (SCL0) and the serial data bus pin (SDA0) have the following configurations.

(1) SCL0 ·····I/O pin for the serial clock
The outputs to both the master and slave are N-channel open drains. The input is a Schmitt input.

(2) SDA0 ·····Shared I/O pin for serial data
The outputs to both the master and slave are N-channel open drains. The input is a Schmitt input.

Since the outputs of the serial clock line and serial data bus line are N-channel open drains, external pullup resistor are required.

Figure 18-6. Pin Configuration

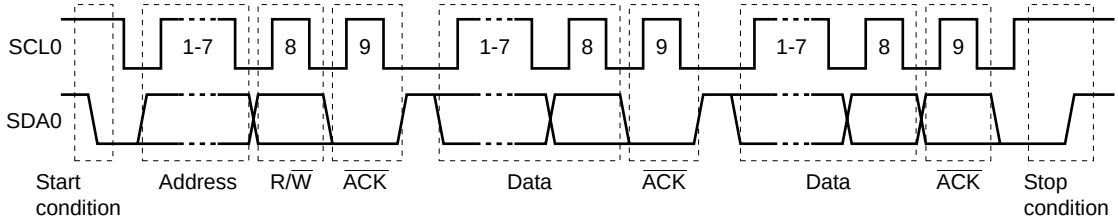


18.5 I²C Bus Definitions and Control Method

Next, the serial data communication formats of the I²C bus and the meanings of the signals used are described.

Figure 18-7 shows the transfer timing of the start condition, data, and stop condition that are output on the serial data bus of the I²C bus.

Figure 18-7. Serial Data Transfer Timing of I²C Bus



The master outputs the start condition, slave address, and stop condition.

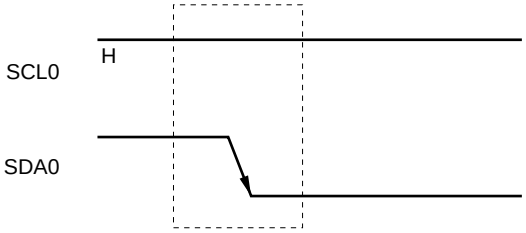
The acknowledge signal ($\overline{ACK}$) can be output by either the master or slave. (Normally, this is output on side receiving 8-bit data.)

The serial clock (SCL0) continues to the master output. However, the slave can extend the SCL0 low-level period and insert waits.

18.5.1 Start condition

When the SCL0 pin is high, the start condition is the SDA0 pin changing from high to low. The start conditions for the SCL0 and SDA0 pins are the signals output when the master starts the serial transfer to the slave. The slave has hardware that detects the start condition.

Figure 18-8. Start Condition



The start condition is output when bit 1 (STT0) of the I²C bus control register (IICC0) is set to one in the stop condition detection state (SPD0: when bit 0 = 1 in the I²C bus status register (IICS0)). In addition, when the start condition is detected, bit 1 (STD0) in IICS0 is set to one.

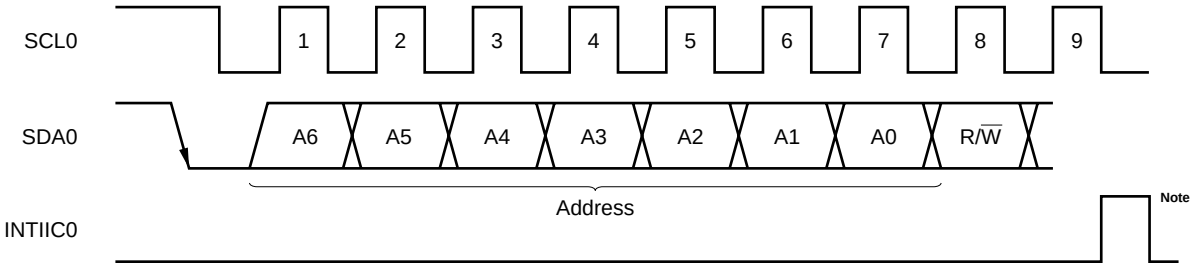
18.5.2 Address

The 7-bit data following the start condition defines the address.

The address is 7-bit data that is output so that the master selects a specific slave from the multiple slaves connected to the bus line. Therefore, the slaves on the bus line must have different addresses.

The slave detects this condition by hardware, and determines whether the 7-bit data matches the slave address register (SVA0). After the slave was selected when the 7-bit data matched the SVA0 value, communication with the master continues until the master sends a start or stop condition.

Figure 18-9. Address



Note When the base address or extended code was received during slave operation, INTIIC0 is not generated.

The address is output by matching the slave address and matching the transfer direction described in **18.5.3 Transfer direction specification** to the serial shift register (IIC0) in 8 bits. In addition, the received address is written to IIC0.

The slave address is allocated to the most significant seven bits of IIC0.

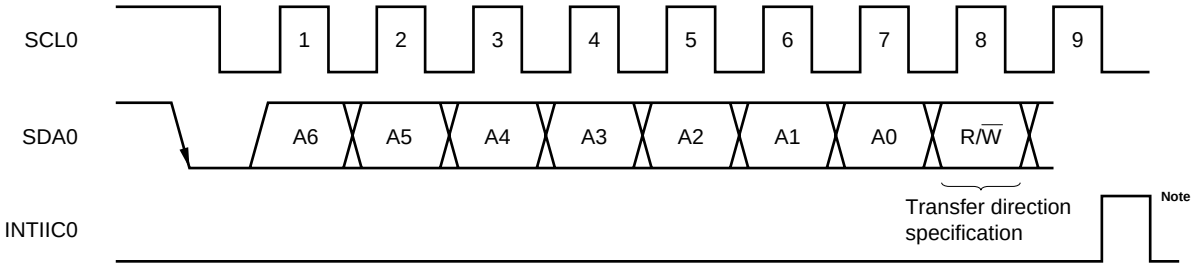
18.5.3 Transfer direction specification

Since the master specifies the transfer direction after the 7-bit address, 1-bit data is transmitted.

A transfer direction specification bit of 0 indicates that the master transmits the data to the slave.

A transfer direction specification bit of 1 indicates that the master receives the data from the slave.

Figure 18-10. Transfer Direction Specification



Note When the base address or extended code is received during slave operation, INTIIC0 is not generated.

18.5.4 Acknowledge signal ($\overline{\text{ACK}}$)

The acknowledge signal verifies the reception of the serial data on the transmitting and receiving sides.

The receiving side returns the acknowledge signal each time 8-bit data is received. Usually, after transmitting 8-bit data, the transmitting side receives an acknowledge signal. However, if the master receives, the acknowledge signal is not output when the last data is received. After an 8-bit transmission, the transmitting side detects whether the receiving side returned an acknowledge signal. If an acknowledge signal is returned, the following processing is performed assuming that reception was correctly performed. Since reception has not been performed correctly if the acknowledge signal is not returned from the slave, the master outputs the stop condition to stop transmission.

If an acknowledge signal is not returned, the following two causes are considered.

<1> The reception is not correct.

<2> The last data has been received.

When the receiving side sets the SDA0 line low at the ninth clock, the acknowledge signal becomes active (normal reception response).

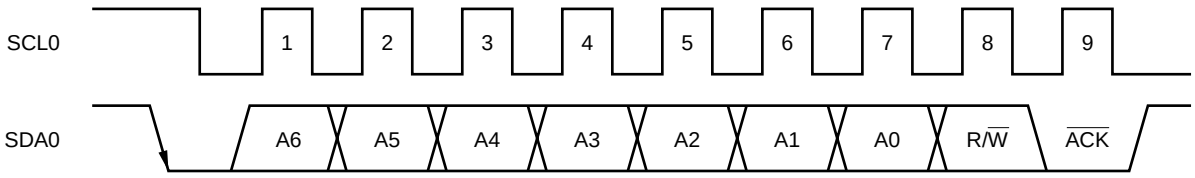
If bit 2 (ACKE0) = 1 in the I²C bus control register (IICC0), the enable automatic generation of the acknowledge signal state is entered.

Bit 3 (TRC0) in the I²C bus status register (IICS0) is set by the data in the eighth bit following the 7-bit address information. However, set ACKE0 = 1 in the reception state when TRC0 bit is 0.

When the slave is receiving (TRC0 = 0), the slave side receives multiple bytes and the next data is not required, when ACKE0 = 0, the master side cannot start the next transfer.

Similarly, the next data is not needed when the master is receiving (TRC0 = 0), set ACKE0 = 0 so that the $\overline{\text{ACK}}$ signal is not generated when you want to output a restart or stop condition. This prevents the output of MSB data in the data on the SDA0 line when the slave is transmitting (transmission stopped).

Figure 18-11. Acknowledge Signal



When receiving the base address, the automatic output of the acknowledge is synchronized to the falling edge of the eighth clock of SCL0 regardless of the ACKE0 value. When receiving at an address other than the base address, the acknowledge signal is not output.

The output method of the acknowledge signal when receiving data is as follows based on the wait timing.

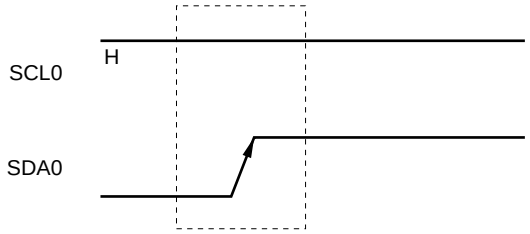
- When 8 clock waits are selected: The acknowledge signal is synchronized to the falling edge of the eighth clock of SCL output by setting ACKE0 = 1 before the wait is released.
- When 9 clock waits are selected: By setting ACKE0 = 1 beforehand, the acknowledge signal is synchronized to the falling edge of the eighth clock of SCL0 and is automatically output.

18.5.5 Stop condition

When the SCL0 pin is high and the SDA0 pin changes from low to high, the stop condition results.

The stop condition is the signal output by the master to the slave when serial transfer ends. The slave has hardware that detects the stop condition.

Figure 18-12. Stop Condition



The stop condition is generated when bit 0 (SPT0) of the I²C bus control register (IICC0) is set to one. And when the stop condition is detected, if bit 0 (SPD0) in the I²C bus status register (IICS0) is set to 1 and bit 4 (SPIE0) of IICC0 is also set to 1, INTIIC0 is generated.

18.5.6 Wait signal ($\overline{\text{WAIT}}$)

The wait signal notifies the other side that the master or slave is being prepared (wait state) for data communication.

The wait state is notified to the other side by setting the SCL0 pin low. When both the master and the slave are released from the wait state, the next transfer can start.

Figure 18-13. Wait Signal (1/2)

(1) The master has a 9 clock wait, and the slave has an 8 clock wait
(Master: transmission, Slave: receiving, ACKE0 = 1)

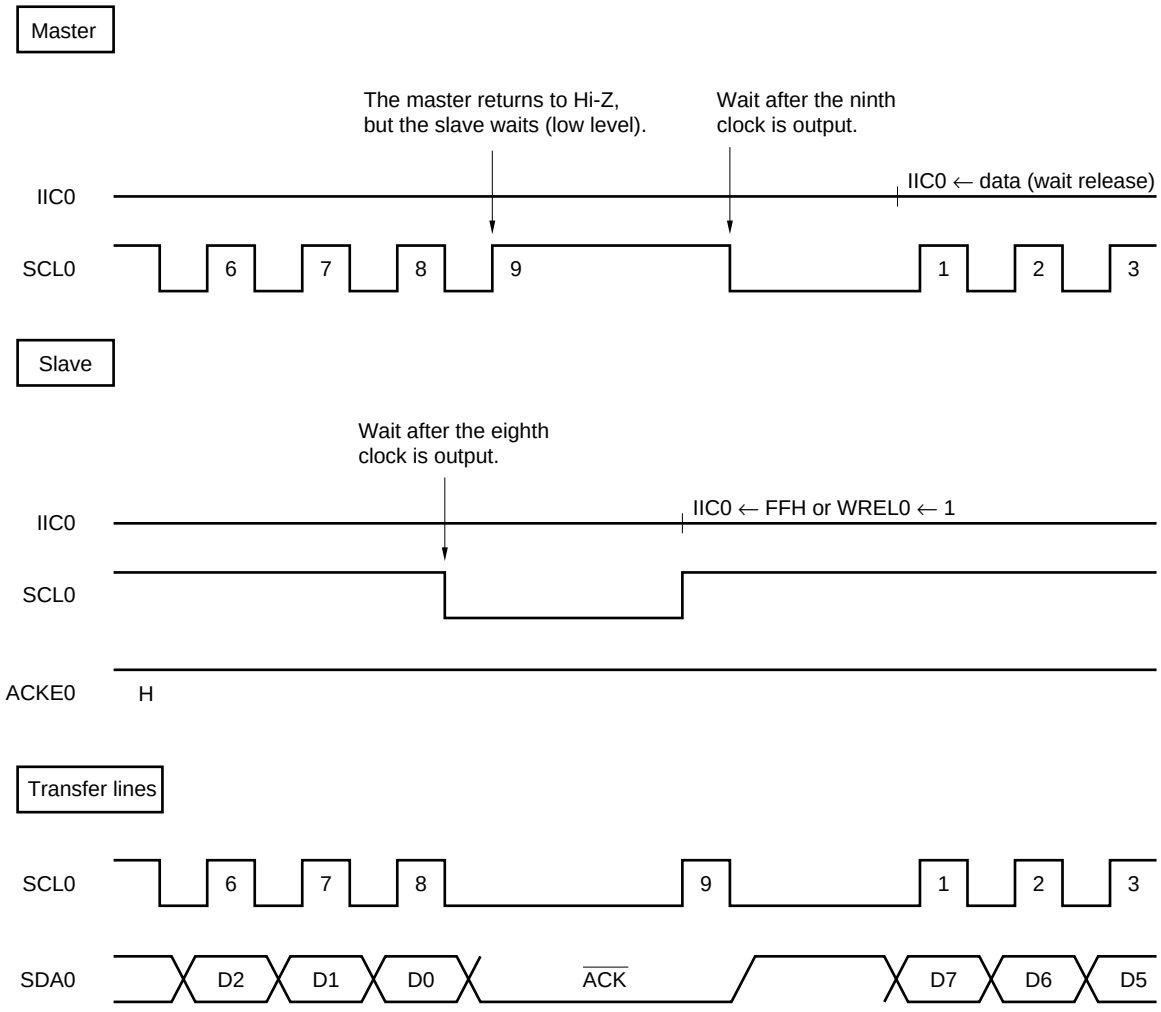
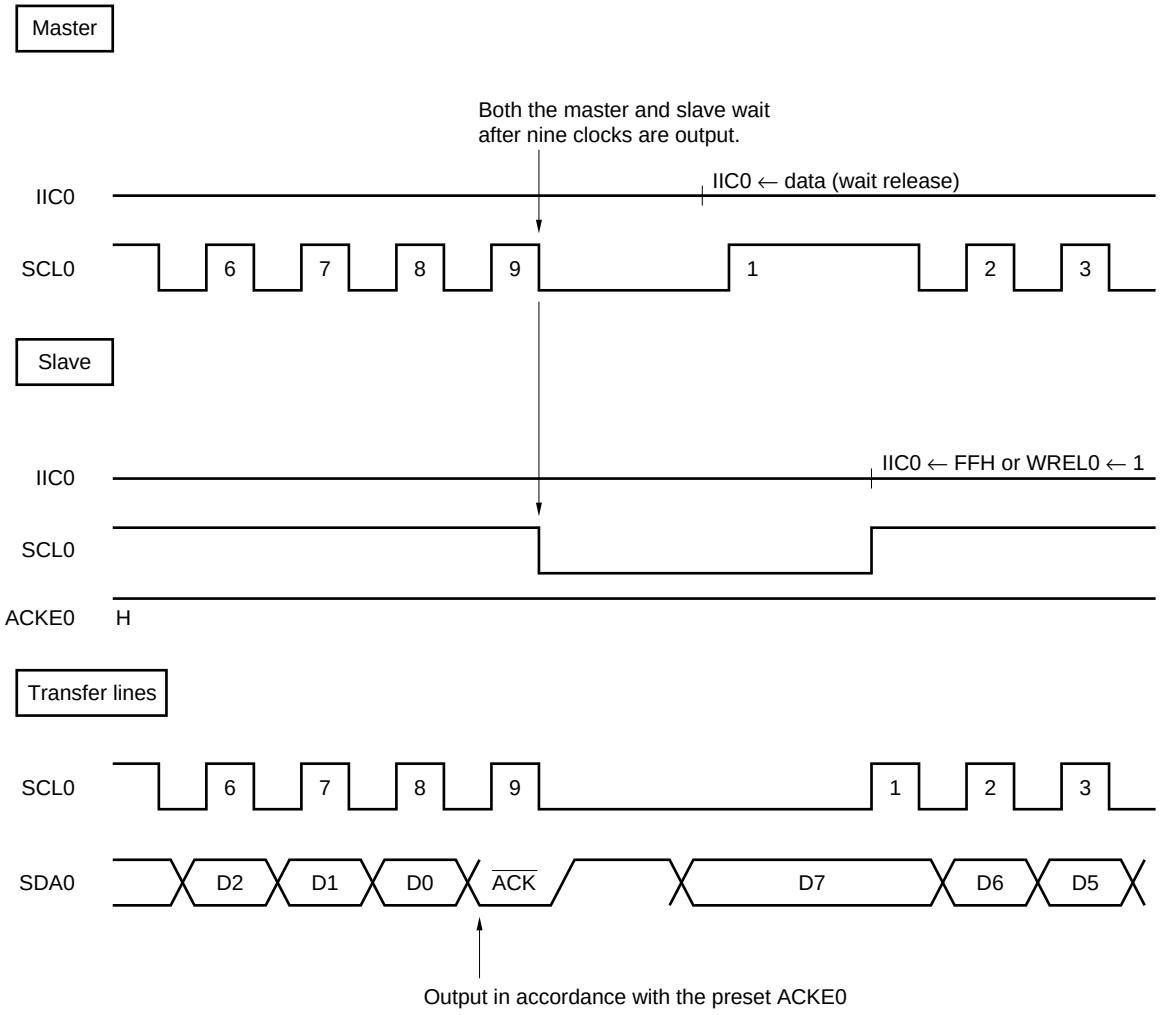


Figure 18-13. Wait Signal (2/2)

(2) Both the master and slave have 9 clock waits
(Master: transmitting, Slave: receiving, ACKE0 = 1)



Remark ACKE0 : Bit 2 in I²C bus control register (IICC0)
WREL0 : Bit 5 in I²C bus control register (IICC0)

A wait is automatically generated by setting bit 3 (WTIM0) of the I²C bus control register (IICC0).

Normally, when bit 5 (WREL0) = 1 in IICC0 or FFH is written to the serial shift register (IIC0), the receiving side releases the wait; when data is written to IIC0, the transmitting side releases the wait.

In the master, the wait can be released by the following methods.

- IICC0 bit 1 (STT0) = 1
- IICC0 bit 0 (SPT0) = 1

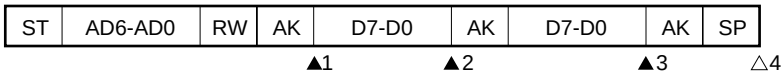
18.5.7 I²C interrupt request (INTIIC0)

This section describes the values of the I²C bus status register (IICS0) at the INTIIC0 interrupt request generation timing and the INTIIC0 interrupt request timing.

(1) Master operation

(a) Start - Address - Data - Data - Stop (normal communication)

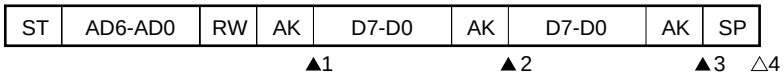
<1> When WTIM0 = 0



▲1 : IICS0 = 10×××110B
▲2 : IICS0 = 10×××000B
▲3 : IICS0 = 10×××000B
△4 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

<2> When WTIM0 = 1



▲1 : IICS0 = 10×××110B
▲2 : IICS0 = 10×××100B
▲3 : IICS0 = 10××××00B
△4 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

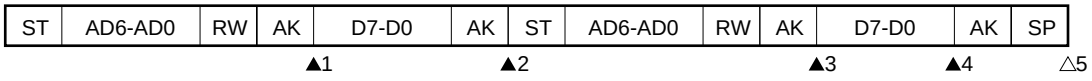
(b) Start - Address - Data - Start - Address - Data - Stop (Restart)

<1> When WTIM0 = 0

▲1 : IICS0 = 10×××110B
▲2 : IICS0 = 10×××000B
▲3 : IICS0 = 10×××110B
▲4 : IICS0 = 10×××000B
△5 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

<2> When WTIM0 = 1

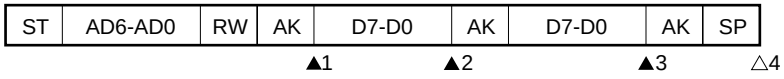


▲1 : IICS0 = 10×××110B
▲2 : IICS0 = 10××××00B
▲3 : IICS0 = 10×××110B
▲4 : IICS0 = 10××××00B
△5 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

(c) Start - Code - Data - Data - Stop (Extended code transmission)

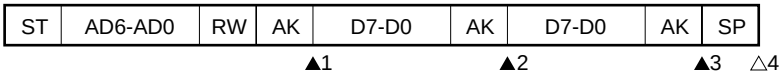
<1> When WTIM0 = 0



▲1 : IICS0 = 1010×110B
▲2 : IICS0 = 1010×000B
▲3 : IICS0 = 1010×000B
△4 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

<2> When WTIM0 = 1



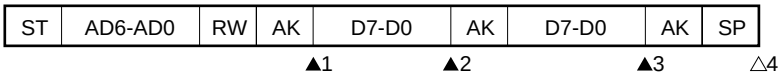
▲1 : IICS0 = 1010×110B
▲2 : IICS0 = 1010×100B
▲3 : IICS0 = 1010××00B
△4 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

(2) Slave operation (when receiving slave address data (SVA0 match))

(a) Start - Address - Data - Data - Stop

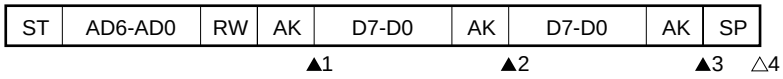
<1> When WTIM0 = 0



▲1 : IICS0 = 0001×110B
▲2 : IICS0 = 0001×000B
▲3 : IICS0 = 0001×000B
△4 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

<2> When WTIM0 = 1

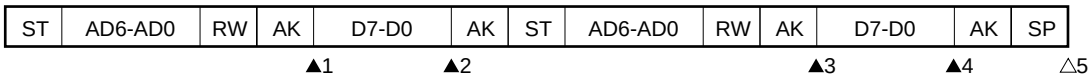


▲1 : IICS0 = 0001×110B
▲2 : IICS0 = 0001×100B
▲3 : IICS0 = 0001××00B
△4 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

(b) Start - Address - Data - Start - Address - Data - Stop

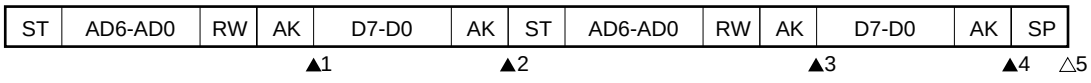
<1> When WTIM0 = 0 (SVA0 match after restart)



▲1 : IICS0 = 0001×110B
▲2 : IICS0 = 0001×000B
▲3 : IICS0 = 0001×110B
▲4 : IICS0 = 0001×000B
△5 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

<2> When WTIM0 = 1 (SVA0 match after restart)

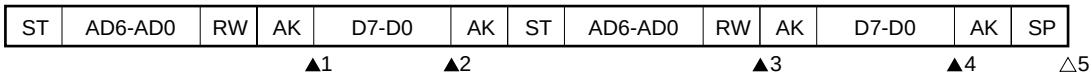


▲1 : IICS0 = 0001×110B
▲2 : IICS0 = 0001××00B
▲3 : IICS0 = 0001×110B
▲4 : IICS0 = 0001××00B
△5 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

(c) Start - Address - Data - Start - Code - Data - Stop

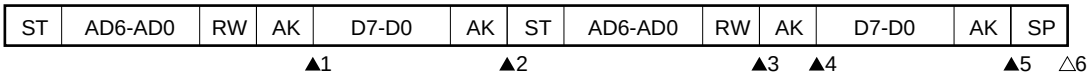
<1> When WTIM0 = 0 (extended code received after restart)



▲1 : IICS0 = 0001×110B
▲2 : IICS0 = 0001×000B
▲3 : IICS0 = 0010×010B
▲4 : IICS0 = 0010×000B
△5 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

<2> When WTIM0 = 1 (extended code received after restart)

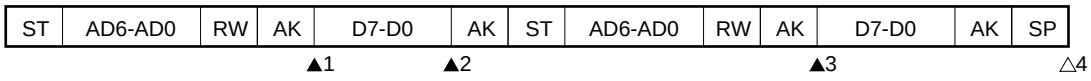


▲1 : IICS0 = 0001×110B
▲2 : IICS0 = 0001××00B
▲3 : IICS0 = 0010×010B
▲4 : IICS0 = 0010×110B
▲5 : IICS0 = 0010××00B
△6 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

(d) Start - Address - Data - Start - Address - Data - Stop

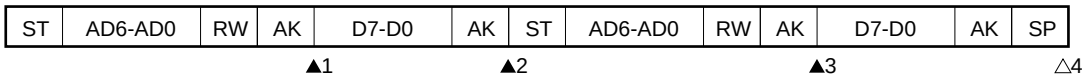
<1> When WTIM0 = 0 (no address match after restart (not extended code))



▲1 : IICS0 = 0001×110B
▲2 : IICS0 = 0001×000B
▲3 : IICS0 = 0000××10B
△4 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

<2> When WTIM0 = 1 (no address match after restart (not extended code))



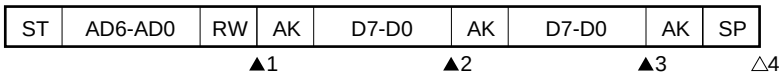
▲1 : IICS0 = 0001×110B
▲2 : IICS0 = 0001××00B
▲3 : IICS0 = 0000××10B
△4 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

(3) Slave operation (when receiving the extended code)

(a) Start - Code - Data - Data - Stop

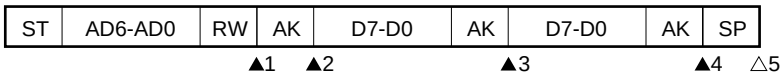
<1> When WTIM0 = 0



▲1 : IICS0 = 0010×010B
▲2 : IICS0 = 0010×000B
▲3 : IICS0 = 0010×000B
△4 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

<2> When WTIM0 = 1

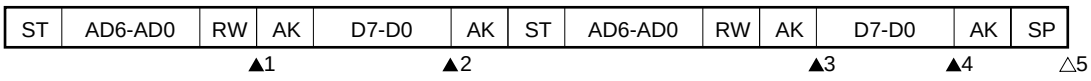


▲1 : IICS0 = 0010×010B
▲2 : IICS0 = 0010×110B
▲3 : IICS0 = 0010×100B
▲4 : IICS0 = 0010××00B
△5 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

(b) Start - Code - Data - Start - Address - Data - Stop

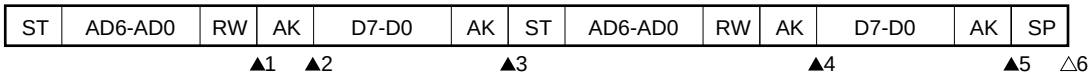
<1> When WTIM0 = 0 (SVA0 match after restart)



▲1 : IICS0 = 0010×010B
▲2 : IICS0 = 0010×000B
▲3 : IICS0 = 0001×110B
▲4 : IICS0 = 0001×000B
△5 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

<2> When WTIM0 = 1 (SVA0 match after restart)

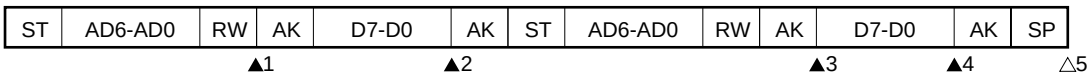


▲1 : IICS0 = 0010×010B
▲2 : IICS0 = 0010×110B
▲3 : IICS0 = 0010××00B
▲4 : IICS0 = 0001×110B
▲5 : IICS0 = 0001××00B
△6 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

(c) Start - Code - Data - Start - Code - Data - Stop

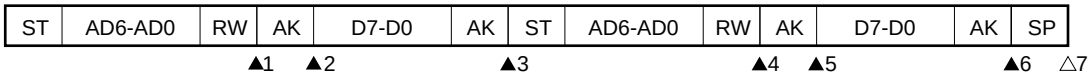
<1> When WTIM0 = 0 (extended code received after restart)



▲1 : IICS0 = 0010×010B
▲2 : IICS0 = 0010×000B
▲3 : IICS0 = 0010×010B
▲4 : IICS0 = 0010×000B
△5 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

<2> When WTIM0 = 1 (extended code received after restart)

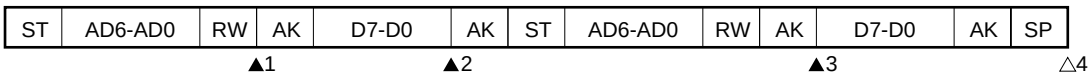


▲1 : IICS0 = 0010×010B
▲2 : IICS0 = 0010×110B
▲3 : IICS0 = 0010××00B
▲4 : IICS0 = 0010×010B
▲5 : IICS0 = 0010×110B
▲6 : IICS0 = 0010××00B
△7 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

(d) Start - Code - Data - Start - Address - Data - Stop

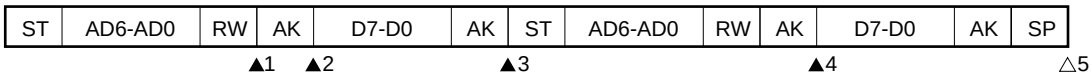
<1> When WTIM0 = 0 (no address match after restart (not an extended code))



▲1 : IICS0 = 0010×010B
▲2 : IICS0 = 0010×000B
▲3 : IICS0 = 00000×10B
△4 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

<2> When WTIM0 = 1 (no address match after restart (not an extended code))

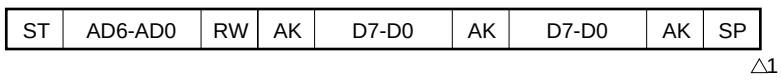


▲1 : IICS0 = 0010×010B
▲2 : IICS0 = 0010×110B
▲3 : IICS0 = 0010××00B
▲4 : IICS0 = 00000×10B
△5 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

(4) Not participating in communication

(a) Start - Code - Data - Data - Stop



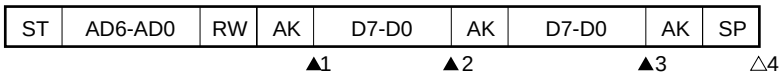
△1 : IICS0 = 00000001B

Remarks △: Generated only when SPIE0 = 1

(5) Arbitration failed operation (operates as the slave after arbitration fails)

(a) When arbitration failed during the transfer of slave address data

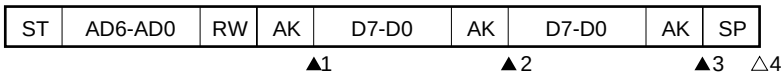
<1> When WTIM0 = 0



▲1 : IICS0 = 0101×110B (Example: Read ALD0 during interrupt servicing.)
▲2 : IICS0 = 0001×000B
▲3 : IICS0 = 0001×000B
△4 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

<2> When WTIM0 = 1

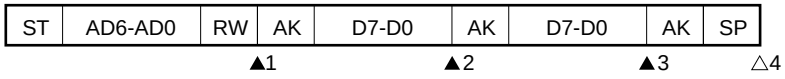


▲1 : IICS0 = 0101×110B (Example: Read ALD0 during interrupt servicing.)
▲2 : IICS0 = 0001×100B
▲3 : IICS0 = 0001××00B
△4 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

(b) When arbitration failed while transmitting an extended code

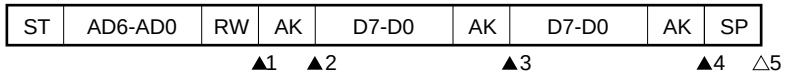
<1> When WTIM0 = 0



▲1 : IICS0 = 0110×010B (Example: Read ALD0 during interrupt servicing.)
▲2 : IICS0 = 0010×000B
▲3 : IICS0 = 0010×000B
△4 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

<2> When WTIM0 = 1

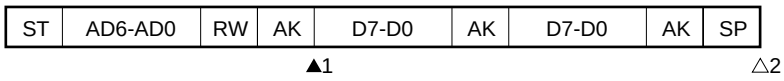


▲1 : IICS0 = 0110×010B (Example: Read ALD0 during interrupt servicing.)
▲2 : IICS0 = 0010×110B
▲3 : IICS0 = 0010×100B
▲4 : IICS0 = 0010××00B
△5 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

(6) Arbitration failed operation (no participation after arbitration failed)

(a) When arbitration failed while transmitting slave address data



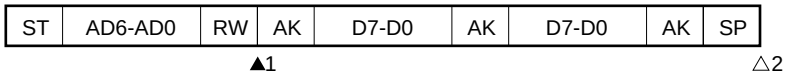
▲1 : IICS0 = 01000110B (Example: Read ALD0 during interrupt servicing.)

△2 : IICS0 = 00000001B

Remarks ▲: Always generated.

△: Generated only when SPIE0 = 1

(b) When arbitration failed while transmitting an extended code



▲1 : IICS0 = 0110×010B (Example: Read ALD0 during interrupt servicing.)

Set LREL0 = 1 from the software.

△2 : IICS0 = 00000001B

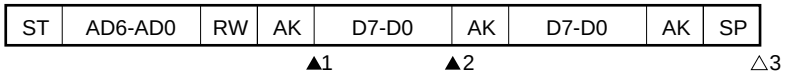
Remarks ▲: Always generated.

△: Generated only when SPIE0 = 1

×: Don't care

(c) When arbitration failed during a data transfer

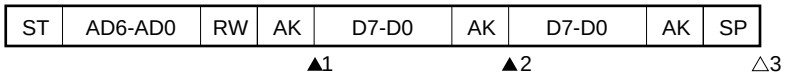
<1> When WTIM0 = 0



▲1 : IICS0 = 10001110B
▲2 : IICS0 = 01000000B (Example: Read ALD0 during interrupt servicing.)
△3 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1

<2> When WTIM0 = 1

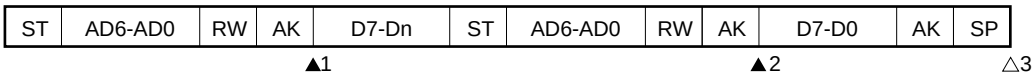


▲1 : IICS0 = 10001110B
▲2 : IICS0 = 01000100B (Example: Read ALD0 during interrupt servicing.)
△3 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1

(d) When failed in the restart condition during a data transfer

<1> Not an extended code (Example: SVA0 match)



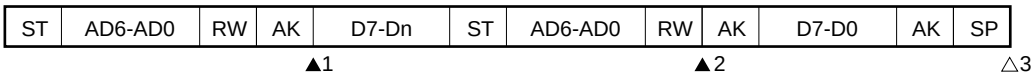
▲1 : IICS0 = 1000×110B

▲2 : IICS0 = 01000110B (Example: Read ALD0 during interrupt servicing.)

△3 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care
Dn = D6-D0

<2> Extended code



▲1 : IICS0 = 1000×110B

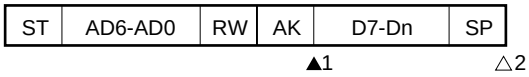
▲2 : IICS0 = 0110×010B (Example: Read ALD0 during interrupt servicing.)

IICC0:LREL0 = 1 set by software.

△3 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care
Dn = D6-D0

(e) When failed in the stop condition during a data transfer



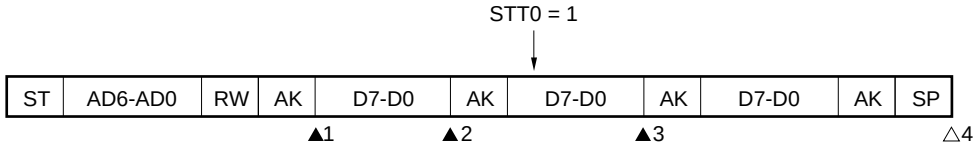
▲1 : IICS0 = 1000×110B

△2 : IICS0 = 01000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care
Dn = D6-D0

(f) When arbitration failed at a low data level and the restart condition was about to be generated

<1> When WTIM0 = 0



▲1 : IICS0 = 1000×110B

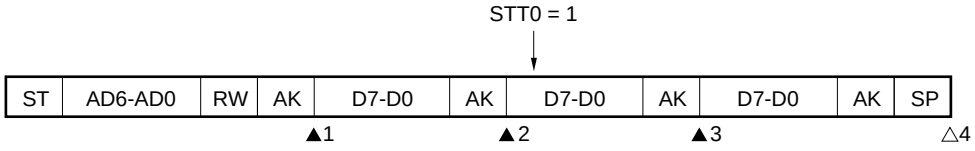
▲2 : IICS0 = 1000×000B

▲3 : IICS0 = 01000000B (Example: Read ALD0 during interrupt servicing.)

△4 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

<2> When WTIM0 = 1



▲1 : IICS0 = 1000×110B

▲2 : IICS0 = 1000××00B

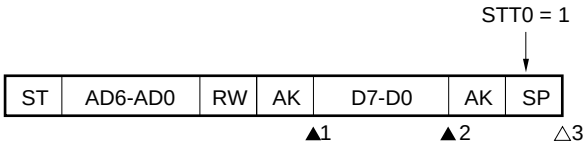
▲3 : IICS0 = 01000100B (Example: Read ALD0 during interrupt servicing.)

△4 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

(g) When arbitration failed in a stop condition and the restart condition was about to be generated

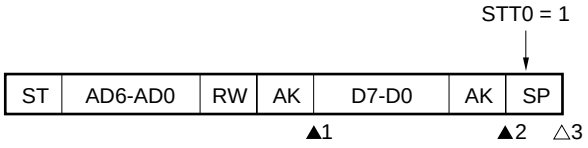
<1> When WTIM0 = 0



▲1 : IICS0 = 1000×110B
▲2 : IICS0 = 1000×000B
△3 : IICS0 = 01000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

<2> When WTIM0 = 1

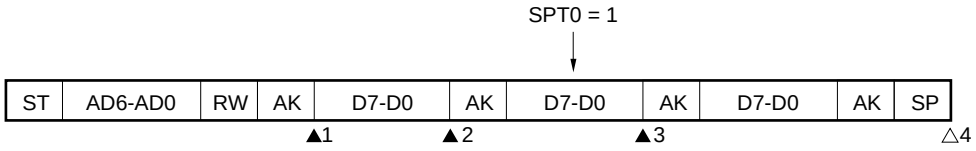


▲1 : IICS0 = 1000×110B
▲2 : IICS0 = 1000××00B
△3 : IICS0 = 01000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

(h) When arbitration failed in the low data level and the stop condition was about to be generated

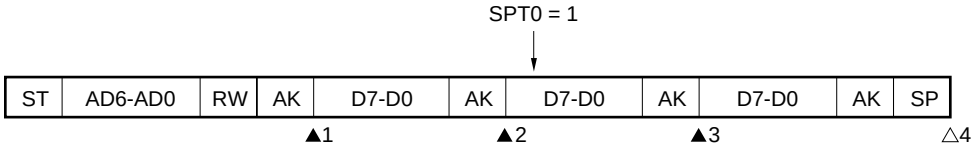
<1> When WTIM0 = 0



▲1 : IICS0 = 1000×110B
▲2 : IICS0 = 1000×000B
▲3 : IICS0 = 01000000B (Example: Read ALD0 during interrupt servicing.)
△4 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

<2> When WTIM0 = 1



▲1 : IICS0 = 1000×110B
▲2 : IICS0 = 1000××00B
▲3 : IICS0 = 01000000B (Example: Read ALD0 during interrupt servicing.)
△4 : IICS0 = 00000001B

Remarks ▲: Always generated.
△: Generated only when SPIE0 = 1
× : Don't care

18.5.8 Interrupt request (INTIIC0) generation timing and wait control

By setting the WTIM0 bit in the I²C bus control register (IICC0), INTIIC0 is generated at the timing shown in Table 18-2 and wait control is performed.

Table 18-2. INTIIC0 Generation Timing and Wait Control

WTIM0	During Slave Operation			During Master Operation		
	Address	Data Reception	Data Transmission	Address	Data Reception	Data Transmission
0	9 **(Notes 1, 2)**	8 **(Note 2)**	9 **(Note 2)**	9	8	9
1	9 **(Notes 1, 2)**	9 **(Note 2)**	9 **(Note 2)**	9	9	9

Notes 1. The INTIIC0 signal and wait of the slave are generated on the falling edge of the ninth clock only when the address set in the slave address register matches (SVA0).
In this case, $\overline{ACK}$ is output regardless of the ACKE0 setting. The slave that received the extended code generates INTIIC0 at the falling edge of the eighth clock.

2. When the address that received SVA0 does not match, INTIIC0 and wait are not generated.

Remark The numbers in the table indicate the number of clocks in the serial clock. In addition, the interrupt request and wait control are both synchronized to the falling edge of the serial clock.

(1) When transmitting and receiving an address

- When the slave is operating : The interrupt and wait timing are determined regardless of the WTIM0 bit.
- When the master is operating: The interrupt and wait timing are generated by the falling edge of the ninth clock regardless of the WTIM0 bit.

(2) When receiving data

- When the master and slave are operating: The interrupt and wait timing are set by the WTIM0 bit.

(3) When transmitting data

- When the master and slave are operating: The interrupt and wait timing are set by the WTIM0 bit.

(4) Releasing a wait

The following four methods release a wait.

- WREL0 = 1 in the I²C bus control register (IICC0)
- Writing to the serial shift register (IIC0)
- Setting the start condition (STT0 = 1 in IICC0)
- Setting the stop condition (SPT0 = 1 in IICC0)

When eight clock waits are selected (WTIM0 = 0), the output level of $\overline{\text{ACK}}$ must be determined before releasing the wait.

(5) Stop condition detection

INTIIC0 is generated when the stop condition is detected.

18.5.9 Address match detection

In the I²C bus mode, the master can select a specific slave device by transmitting the slave address.

Address matching can be detected automatically by the hardware. When the base address is set in the slave address register (SVA0), if the slave address transmitted from the master matches the address set in SVA0, or if the extended code is received, an INTIIC0 interrupt request occurs.

18.5.10 Error detection

In the I²C bus mode, since the state of the serial bus (SDA0) during transmission is introduced into the serial shift register (IIC0) of the transmitting device, transmission errors can be detected by comparing the IIC0 data before and after the transmission. In this case, if two data differ, the decision is that a transmission error was generated.

18.5.11 Extended codes

(1) If the most significant four bits of the receiving address are “0000” or “1111”, an extended code is received and the extended code received flag (EXC0) is set. The interrupt request (INTIIC0) is generated at the falling edge of the eighth clock. The base address stored in the slave address register (SVA0) is not affected.

(2) In 10-bit address transfers, the following occurs when “111110XX” is set in SVA0 and “111110XX0” is transferred from the master. However, INTIIC0 is generated at the falling edge of the eighth clock.

- Most significant 4 bits of data match: EXC0 = 1**Note**
- 7-bit data match : COI0 = 1**Note**

Note EXC0: Bit 5 of the I²C bus status register (IICS0)
COI0 : Bit 4 of the I²C bus status register (IICS0)

(3) Since the processing after an interrupt request is generated differs depending on the data that follows the extended code, the software performs this processing.
For example, when operation as a slave is not desired after an extended code is received, set bit 6 (LREL0) = 1 in the I²C bus control register (IICC0) and enter the next communication wait state.

Table 18-3. Definitions of the Extended Code Bits

Slave Address	R/W Bit	Description
0000 000	0	General call address
0000 000	1	Start byte
0000 001	×	CBUS address
0000 010	×	Address reserved in the different bus format
1111 0××	×	10-bit slave address setting

18.5.12 Arbitration

When multiple masters simultaneously output start conditions (when STT0 = 1 occurs before STD0 = 1**Note**), the master communicates while the clock is adjusted until the data differ. This operation is called arbitration.

A master that failed arbitration sets the arbitration failed flag (ALD0) of the I²C bus status register (IICS0) at the timing of the failed arbitration. The SCL0 and SDA0 lines enter the high impedance state, and the bus is released.

Failed arbitration is detected when ALD0 = 1 by software at the timing of the interrupt request generated next (eighth or ninth clock, stop condition detection, etc.).

At the timing for generating the interrupt request, refer to **18.5.7 I²C interrupt request (INTIIC0)**.

Note STD0: Bit 1 in the I²C bus status register (IICS0)
STT0: Bit 1 in the I²C bus control register (IICC0)

Figure 18-14. Example of Arbitration Timing

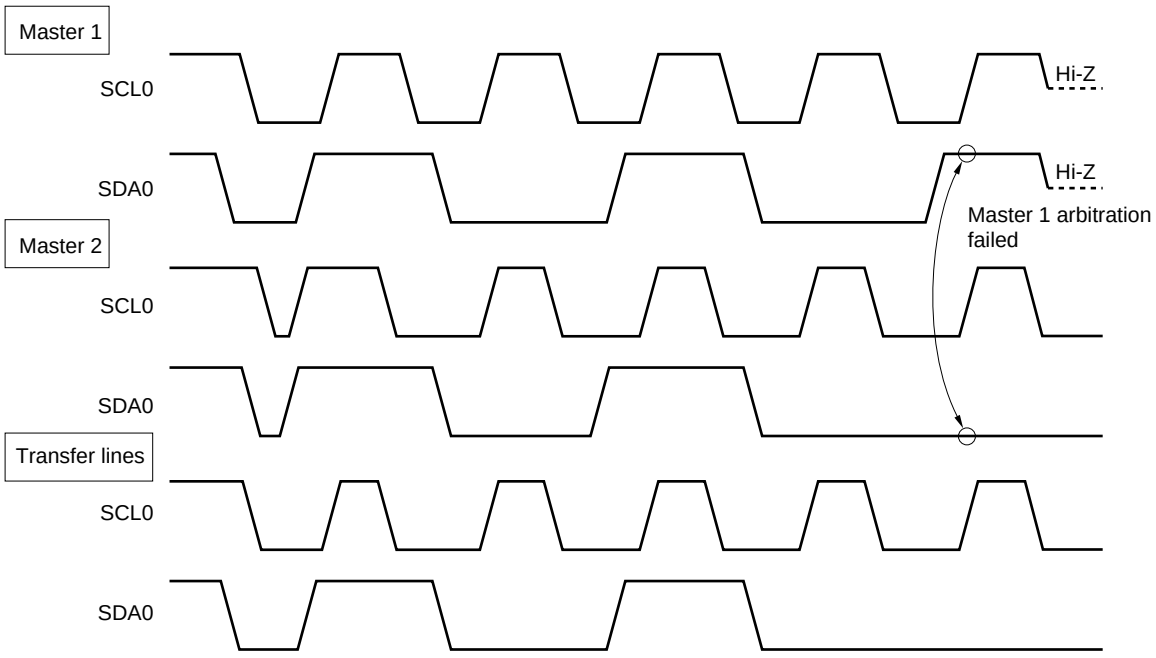


Table 18-4. Arbitration Generation States and Interrupt Request Generation Timing

Arbitration Generation State	Interrupt Request Generation Timing
During address transmission	Falling edge of clock 8 or 9 after byte transfer[Note 1]
Read/write information after address transmission	
During extended code transmission	
Read/write information after extended code transmission	
During data transmission	
During $\overline{\text{ACK}}$ transfer period after data transmission	
Restart condition detection during data transfer	
Stop condition detection during data transfer	When stop condition is output (SPIE0 = 1)[Note 2]
Data is low when the restart condition is about to be output.	Falling edge of clock 8 or 9 after byte transfer[Note 1]
The restart condition should be output, but the stop condition is detected.	Stop condition is output (SPIE0 = 1)[Note 2]
Data is low when the stop condition is about to be output.	Falling edge of clock 8 or 9 after byte transfer[Note 1]
SCL0 is low when the restart condition is about to be output.	

Notes 1. If WTIM0 = 1 (bit 3 = 1 in the I²C bus control register (IICC0)), an interrupt request is generated at the timing of the falling edge of the ninth clock. If WTIM0 = 0 and the slave address of the extended code is received, an interrupt request is generated at the timing of the falling edge of the eighth clock.

2. When arbitration is possible, use the master to set SPIE0 = 1.

Remark SPIE0 : Bit 5 in the I²C bus control register (IICC0)

18.5.13 Wake-up function

This is a slave function of the I²C bus and generates the interrupt request (INTIIC0) when the base address and extended code were received.

When the address does not match, an unused interrupt request is not generated, and efficient processing is possible.

When the start condition is detected, the wake-up standby function is entered. Since the master can become a slave in an arbitration failure (when a start condition was output), the wake-up standby function is entered while the address is transmitted.

However, when the stop condition is detected, the generation of interrupt requests is enabled or disabled based on the setting of bit 5 (SPIE0) in the I²C bus register (IICC0) unrelated to the wake-up function.

18.5.14 Communication reservation

When you want the master to communicate after being in the not participating state in the bus, the start condition can be transmitted when a bus is released by reserving communication. The following two states are included when the bus does not participate.

- When there was no arbitration in the master and the slave
- When the extended code is received and operation is not as a slave (bus released when $\overline{ACK}$ is not returned, and bit 6 (LREL0) = 1 in the I²C bus control register (IICC0))

When bit 1 (STT0) of IICC0 is set in the not participating state in the bus, after the bus is released (after stop condition detection), the start condition is automatically generated, and the wait state is entered. When the bus release is detected (stop condition detection), the address transfer starts as the master by the write operation of the serial shift register (IIC0). In this case, set bit 4 (SPIE0) in IICC0.

When STT0 is set, whether it operates as a start condition or for communication reservation is determined by the bus state.

- When the bus is released ..... Start condition generation
- When the bus is not released (standby state) ..... Communication reservation

The method that detects the operation of STT0 sets STT0 and verifies the STT0 bit again after the wait time elapses.

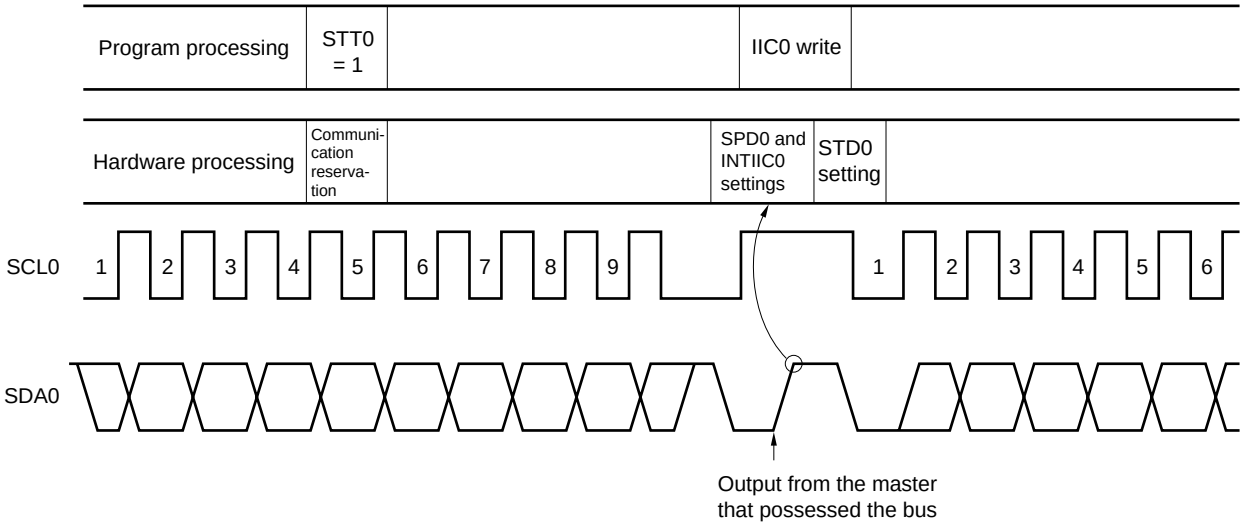
Use the software to save the wait time which is a time listed in Table 18-5. The wait time can be set by bits 3, 1, and 0 (SMC, CL1, CL0) in the prescaler mode register (SPRM0) for the serial clock.

Table 18-5. Wait Times

SMC	CL1	CL0	Wait Time
0	0	0	26 clocks
0	0	1	46 clocks
0	1	0	
0	1	1	37 clocks
1	0	0	16 clocks
1	0	1	
1	1	0	
1	1	1	13 clocks

Figure 18-15 shows the timing of communication reservation.

Figure 18-15. Timing of Communication Reservation



IIC0 : Serial shift register
STT0: Bit 1 in the I²C bus control register (IICC0)
STD0: Bit 1 in the I²C bus status register (IICS0)
SPD0: Bit 0 in the I²C bus status register (IICS0)

The communication reservation is accepted at the following timing. After bit 1 (STD0) = 1 in the I²C bus status register (IICS0), the communication is reserved by bit 1 (STT0) = 1 in the I²C bus control register (IICC0) until the stop condition is detected.

Figure 18-16. Communication Reservation Acceptance Timing

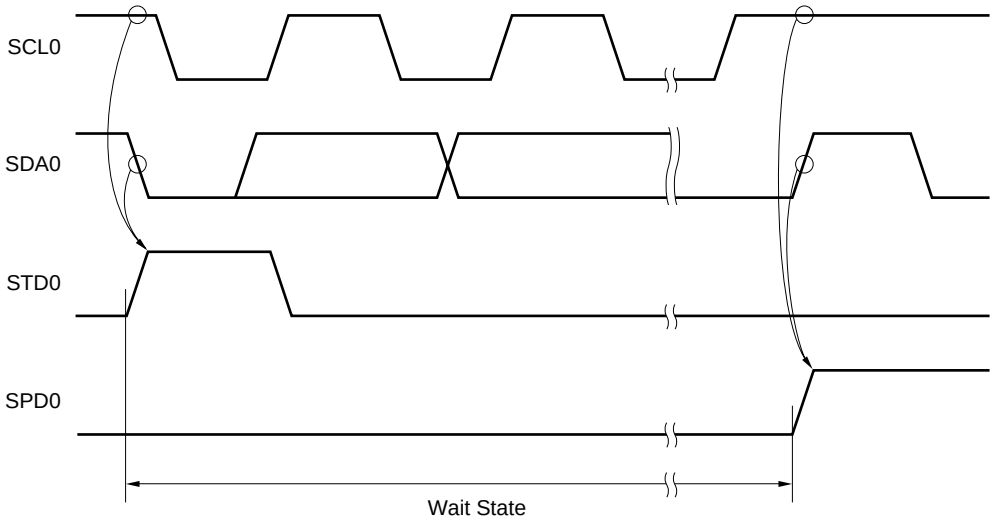
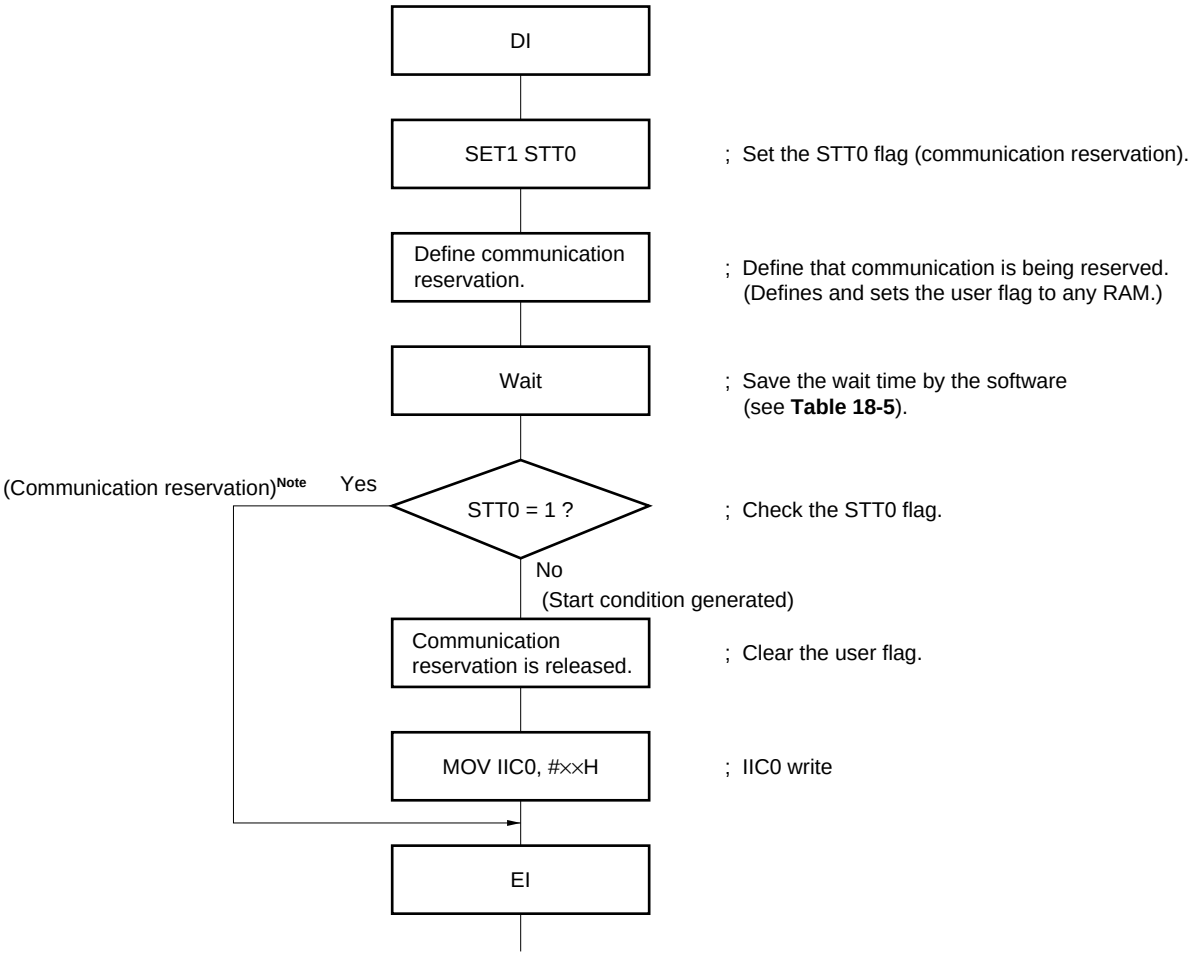


Figure 18-17 shows the communication reservation procedure.

Figure 18-17. Communication Reservation Procedure



Note When the communication is being reserved, the serial shift register (IIC0) is written by the stop condition interrupt.

Remark STT0 : Bit 1 in the I²C bus control register (IICC0)
IIC0 : Serial shift register

18.5.15 Additional warnings

After a reset, when the master is communicating from the state where the stop condition is not detected (bus is not released), perform master communication after the stop condition is first generated and the bus is released.

The master cannot communicate in the state where the bus is not released (the stop condition is not detected) in the multi-master.

The following procedure generates the stop condition.

<1> Set the prescaler mode register (SPRM0) for the serial clock.

<2> Set bit 7 (IICE0) in the I²C bus control register (IICC0).

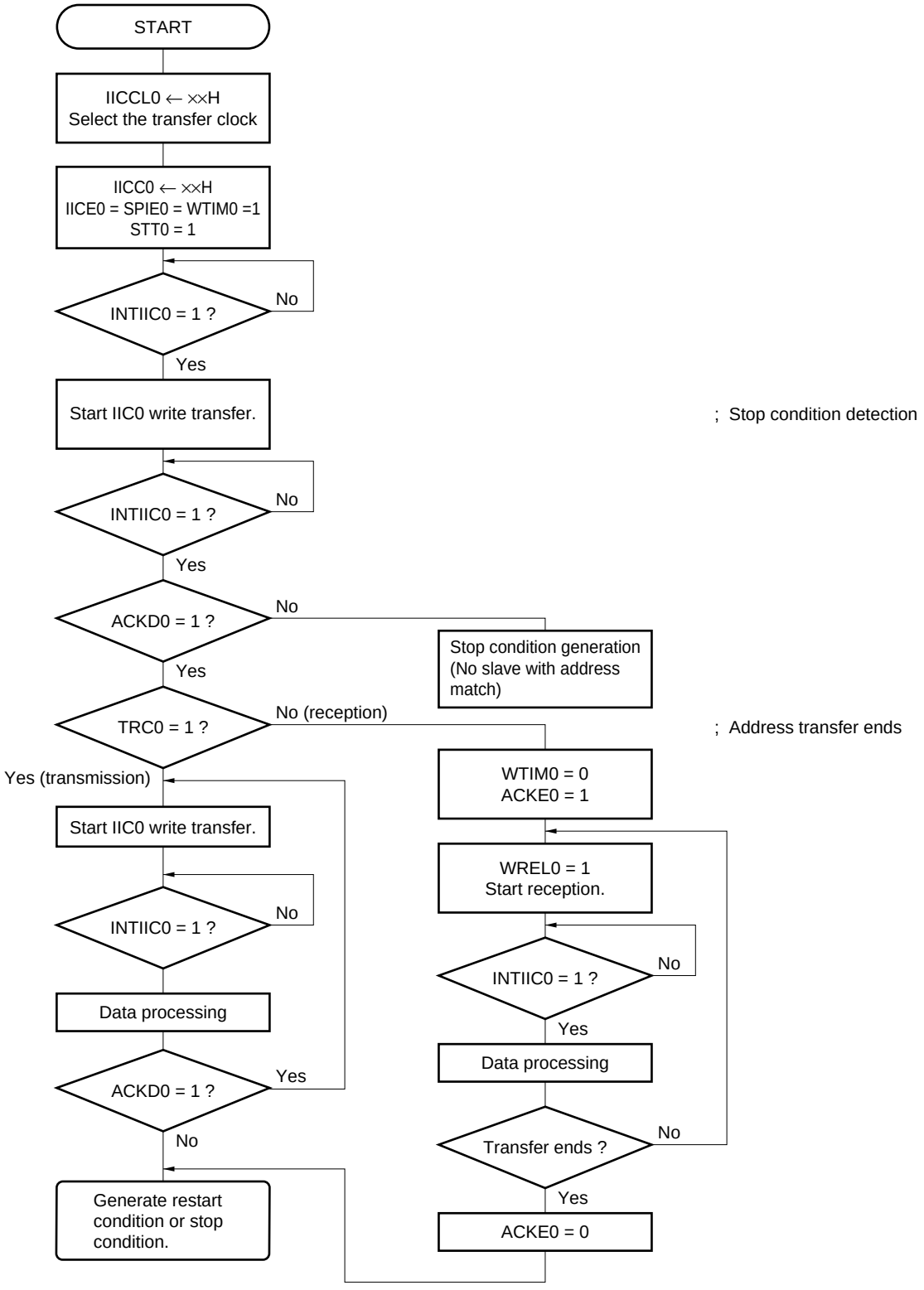
<3> Set bit 0 of IICC0.

18.5.16 Communication operation

(1) Master operation

The following example shows the master operating procedure.

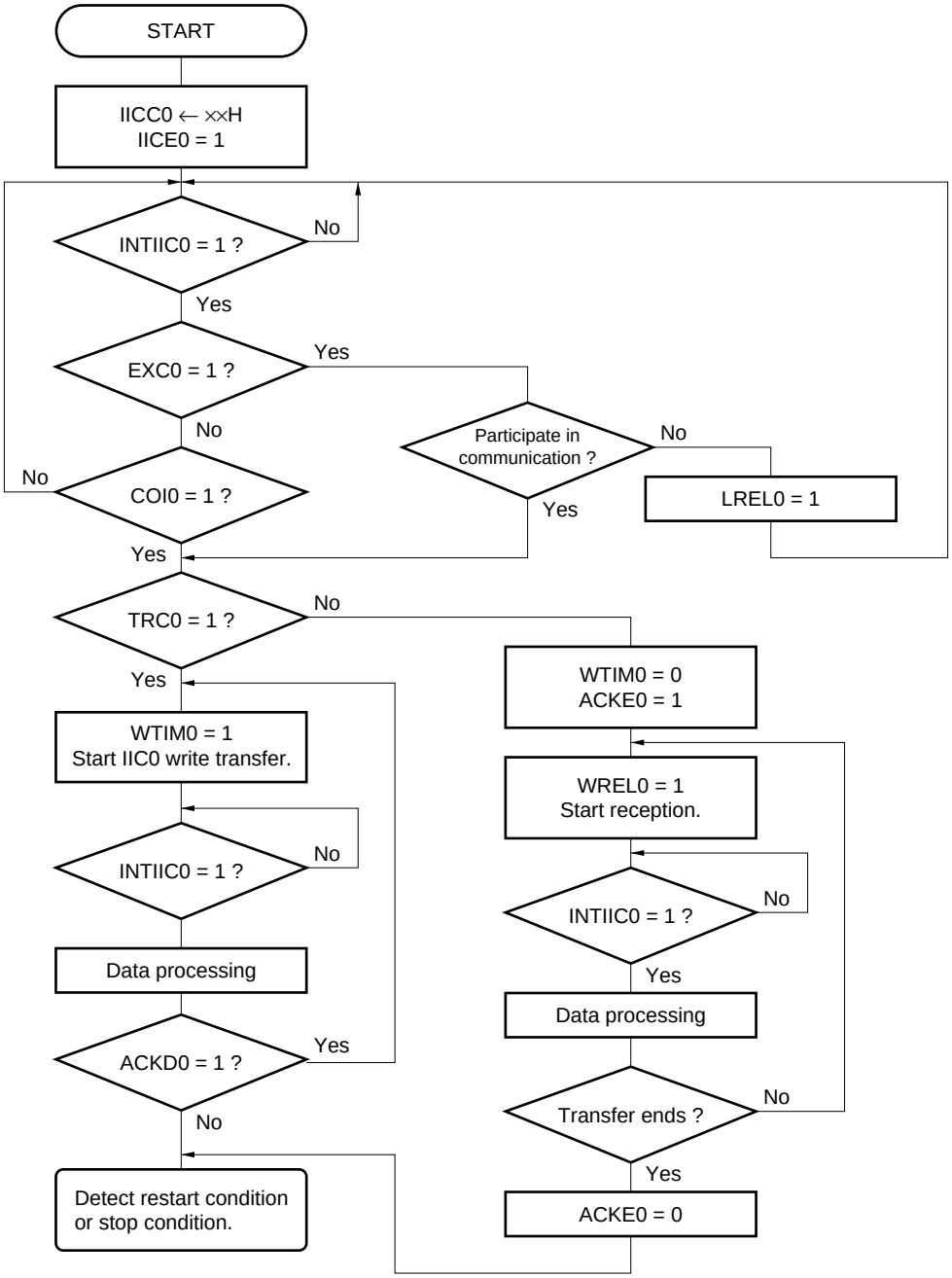
Figure 18-18. Master Operating Procedure



(2) Slave operation

The following example is the slave operating procedure.

Figure 18-19. Slave Operating Procedure



18.6 Timing Charts

In the I²C bus mode, the master outputs an address on the serial bus and selects one of the slave devices from multiple slave devices as the communication target.

The master transmits the TRC0 bit, bit 3 of the I²C bus status register (IICS0), that indicates the transfer direction of the data after the slave address and starts serial communication with the slave.

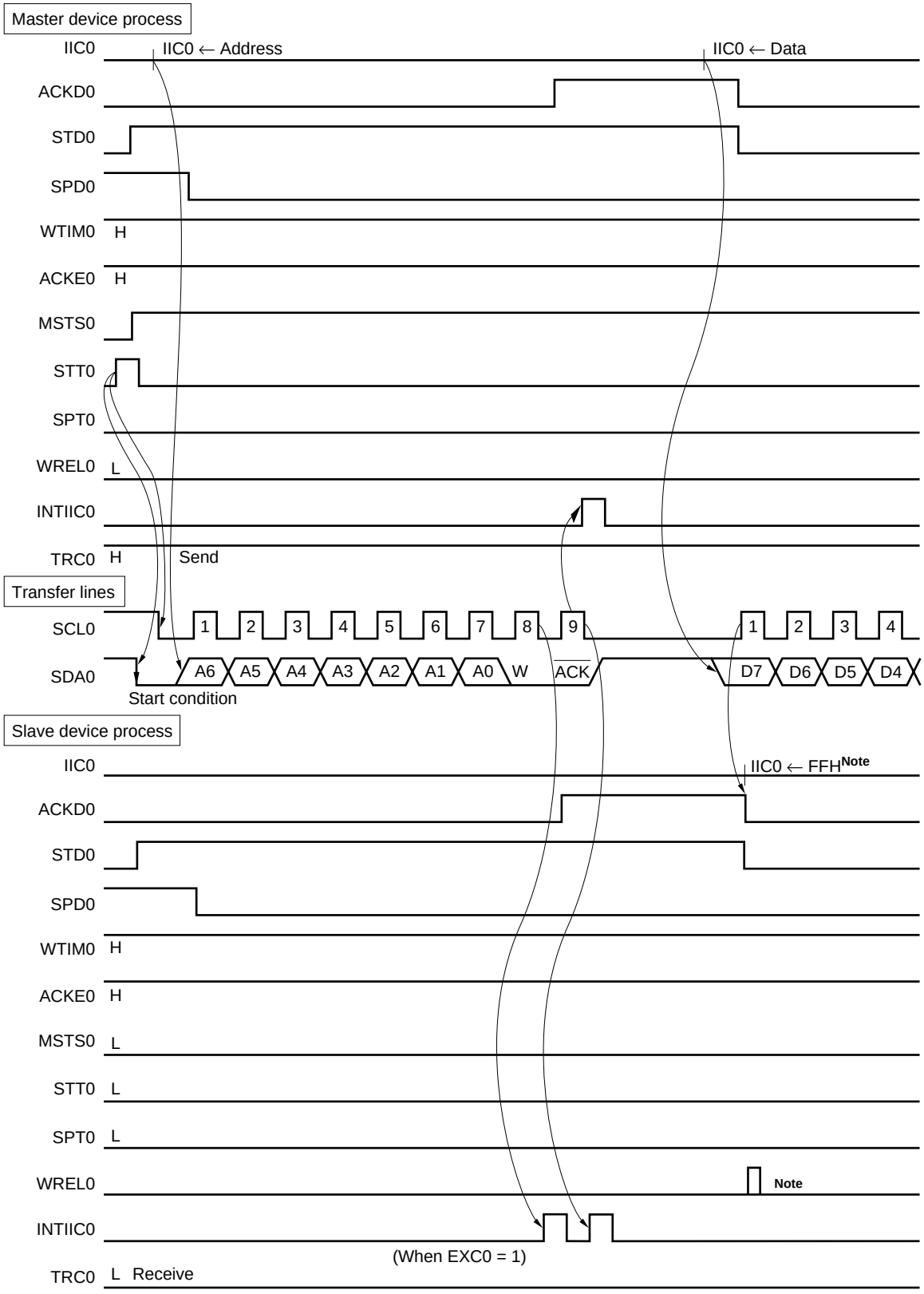
Figures 18-20 and 18-21 are the timing charts for data communication.

Shifting of the shift register (IIC0) is synchronized to the falling edge of the serial clock (SCL0). The transmission data is transferred to the SO0 latch and output from the SDA0 pin with the MSB first.

The data input at the SDA0 pin is received by IIC0 at the rising edge of SCL0.

Figure 18-20. Master → Slave Communication Example (when master and slave select 9 clock waits) (1/3)

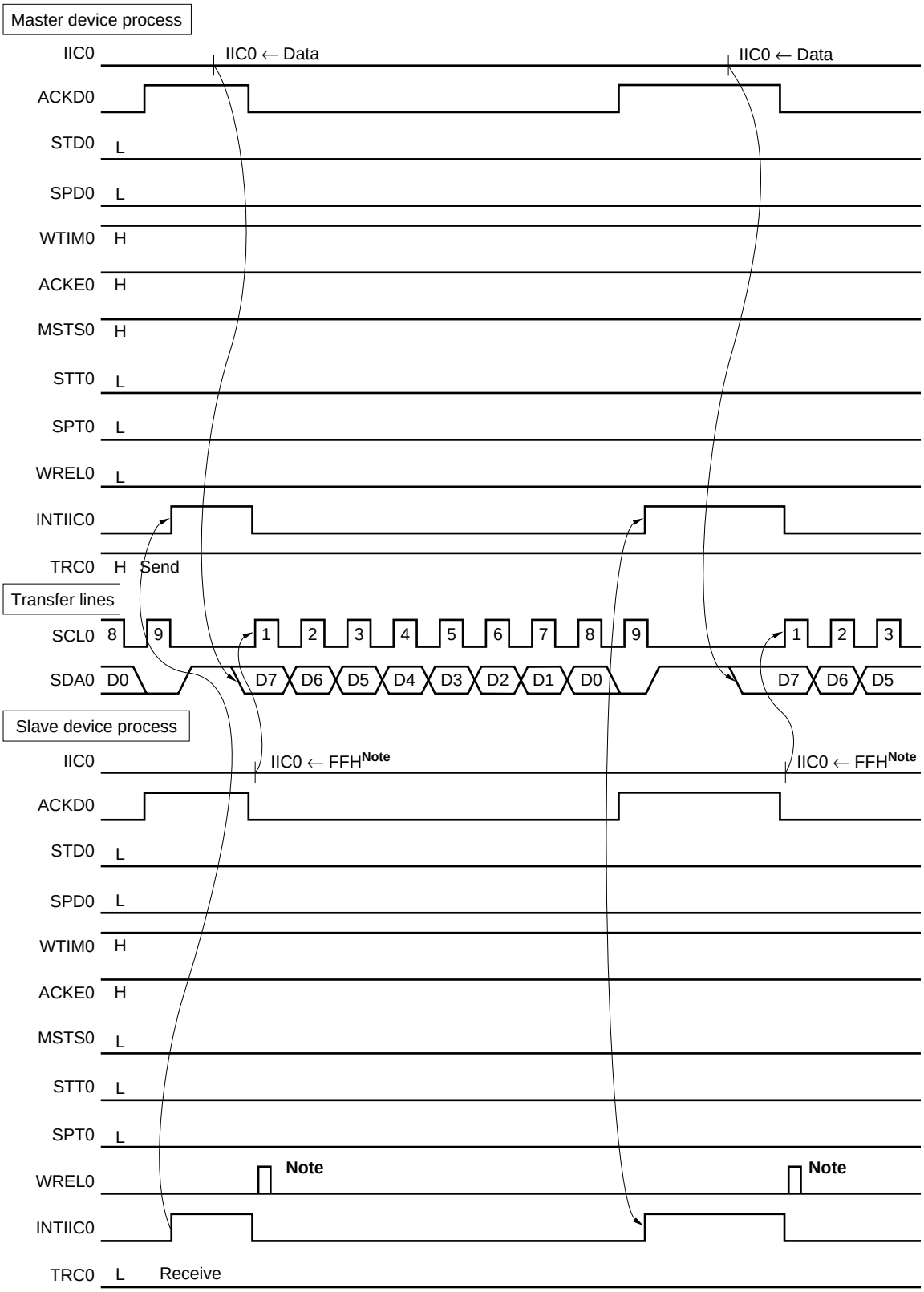
(1) Start Condition - Address



Note Release the slave wait by either IIC0 ← FFH or setting WREL0.

Figure 18-20. Master → Slave Communication Example (when master and slave select 9 clock waits) (2/3)

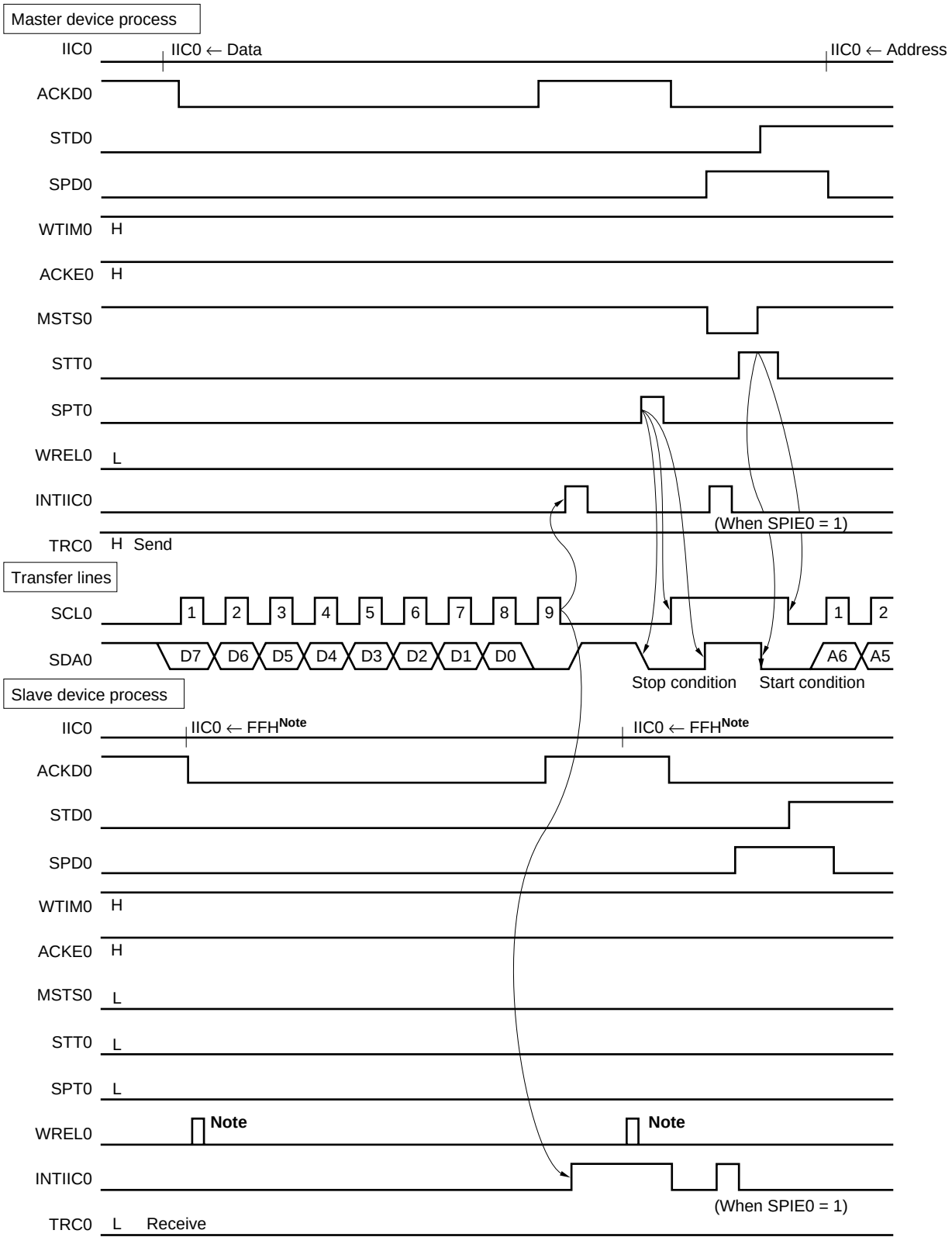
(2) Data



Note Release the slave wait by either IIC0 ← FFH or setting WREL0.

Figure 18-20. Master → Slave Communication Example (when master and slave select 9 clock waits) (3/3)

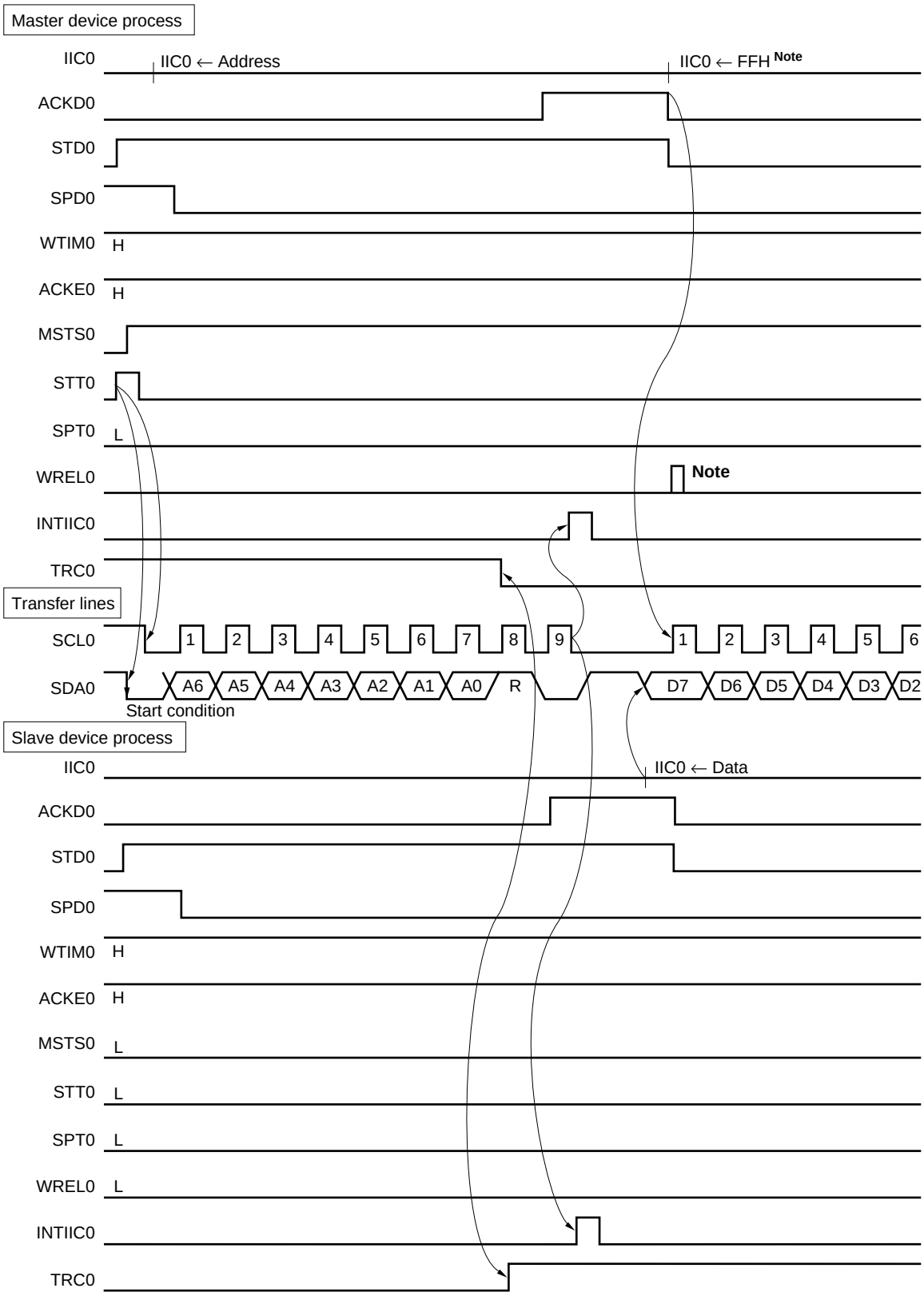
(3) Stop Condition



Note Release the slave wait by either IIC0 ← FFH or setting WREL0.

Figure 18-21. Slave → Master Communication Example (when master and slave select 9 clock waits) (1/3)

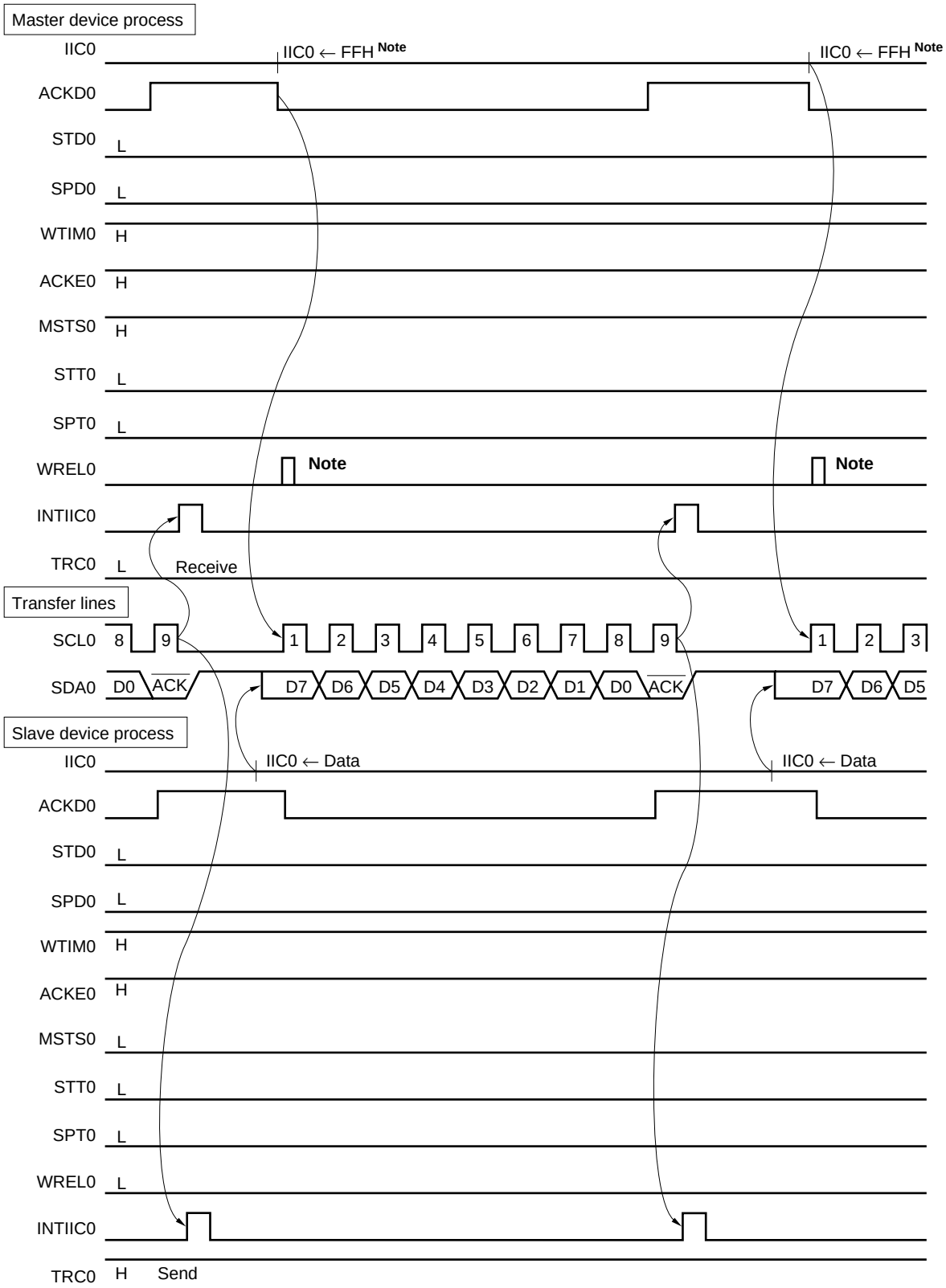
(1) Start Condition - Address



Note Release the slave wait by either IIC0 ← FFH or setting WREL0.

Figure 18-21. Slave → Master Communication Example (when master and slave select 9 clock waits) (2/3)

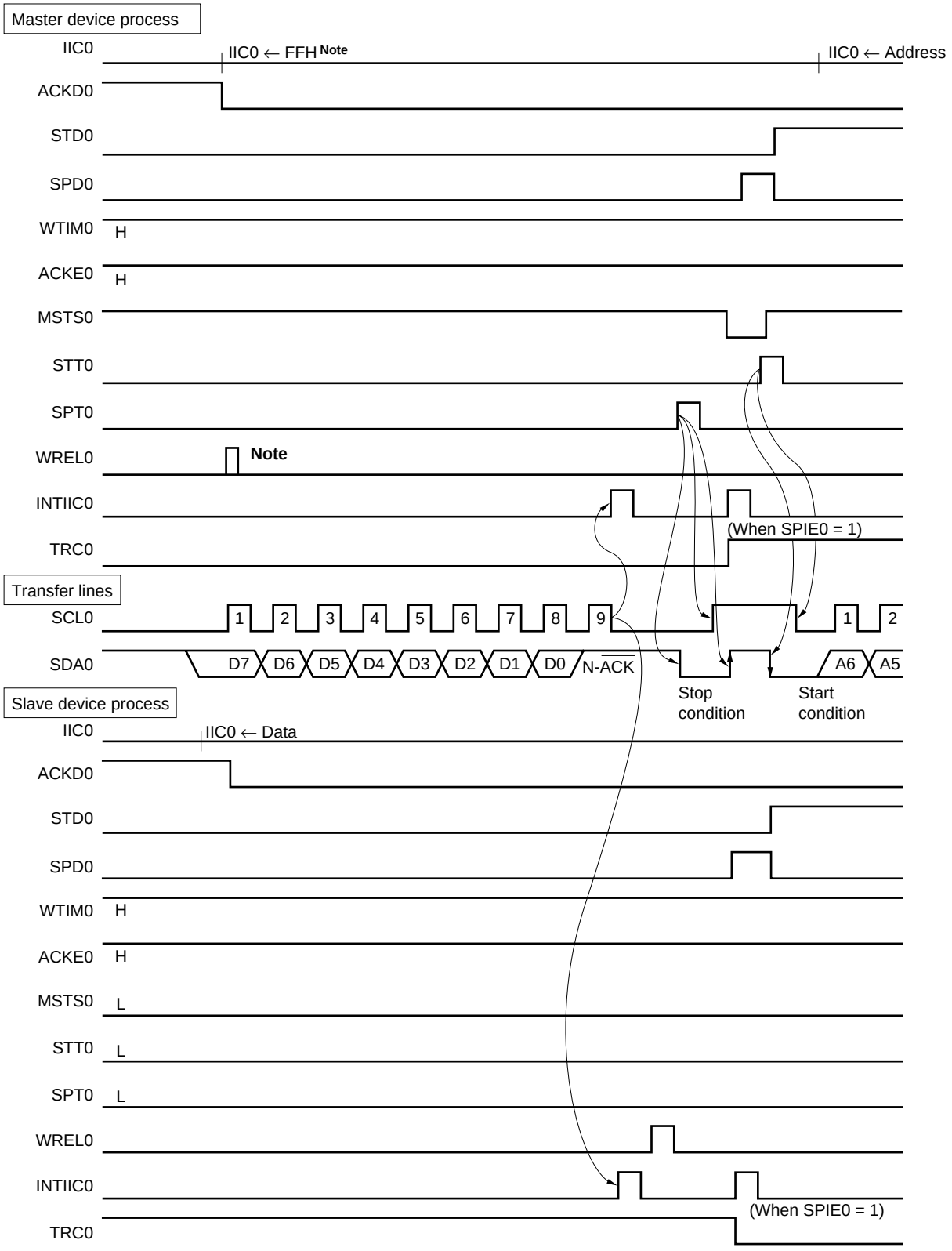
(2) Data



Note Release the slave wait by either IIC0 ← FFH or setting WREL0.

Figure 18-21. Slave → Master Communication Example (when master and slave select 9 clock waits) (3/3)

(3) Stop Condition



Note Release the slave wait by either IIC0 ← FFH or setting WREL0.

CHAPTER 19 CLOCK OUTPUT FUNCTION

19.1 Functions

The clock output function is used to output the clock supplied to a peripheral LSI and carrier output during remote transmission. The clock selected by means of the clock output control register (CKS) is output from the PCL/P23 pin.

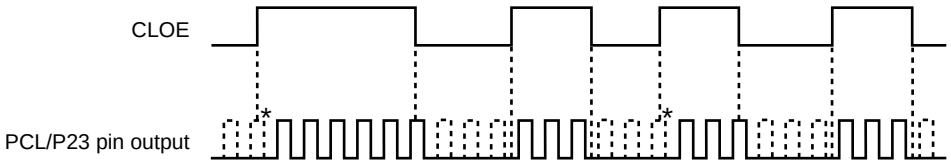
To output the clock pulse, follow the procedure described below.

<1> Select the output frequency of the clock pulse (while clock pulse output is disabled) with bits 0 to 3 (CCS0 to CCS3).
<2> Set 0 in output latch P23.
<3> Set bit 3 (PM23) of the port mode register (PM2) to 0 (to set the output mode).
<4> Set bit 4 (CLOE) of CKS to 1.

Caution If the output latch of P23 is set to 1, clock output cannot be used.

Remark The clock output function is designed so that pulses with a narrow width are not output when clock output enable/disabled is switched (See "*" in **Figure 19-1**).

Figure 19-1. Remote Control Output Application Example



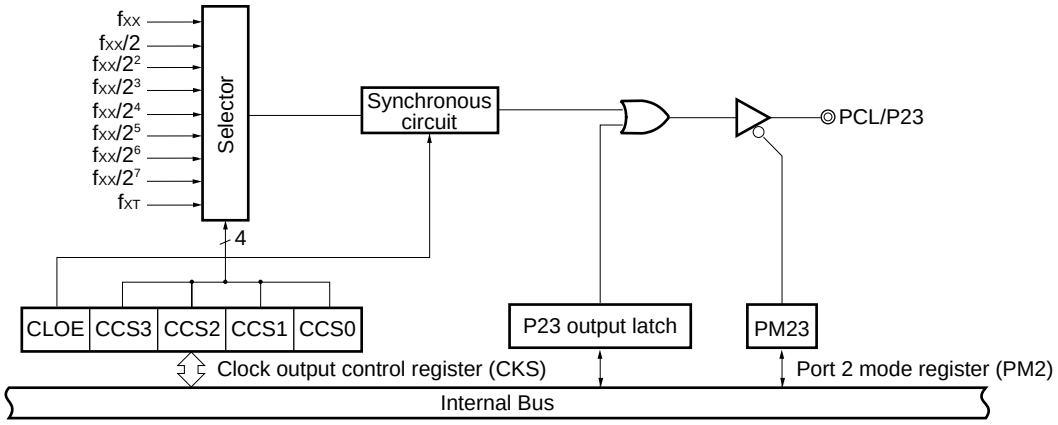
19.2 Configuration

The clock output function consists of the following hardware.

Table 19-1. Clock Output Function Configuration

Item	Configuration
Control register	Clock output control register (CKS) Port 2 mode register (PM2)

Figure 19-2. Clock Output Function Block Diagram



19.3 Control Registers

The following two types of registers are used to control the clock output function.

- Clock output control register (CKS)
- Port 2 mode register (PM2)

(1) Clock output control register (CKS)

This register sets the PCL output clock.

CKS is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets CKS to 00H.

Remark CKS provides a function for setting the buzzer output clock besides setting the PCL output clock.

Figure 19-3. Clock Output Control Register (CKS) Format

Address: 0FF40H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
CKS	BZOE	BCS1	BCS0	CLOE	CCS3	CCS2	CCS1	CCS0

BZOE	Buzzer Output Control (See **Figure 20-2**)

BCS1	BCS0	Buzzer Output Frequency Selection (See **Figure 20-2**)

CLOE	Clock Output Control
0	Clock Output Stop
1	Clock Output Start

CCS3	CCS2	CCS1	CCS0	Clock Output Frequency Selection
0	0	0	0	f_{XX} (12.5 MHz)
0	0	0	1	$f_{XX}/2$ (6.25 MHz)
0	0	1	0	$f_{XX}/4$ (3.13 MHz)
0	0	1	1	$f_{XX}/8$ (1.56 MHz)
0	1	0	0	$f_{XX}/16$ (781 kHz)
0	1	0	1	$f_{XX}/32$ (391 kHz)
0	1	1	0	$f_{XX}/64$ (195 kHz)
0	1	1	1	$f_{XX}/128$ (97.7 kHz)
1	0	0	0	f_{XT} (32.768 kHz)
Other than above				Setting prohibited

Remarks 1. f_{XX}: Main system clock frequency (f_X or $f_X/2$)

2. f_X: Main system clock oscillation frequency

3. f_{XT}: Subsystem clock oscillation frequency

4. Figures in parentheses apply to operation with f_{XX} = 12.5 MHz or f_{XT} = 32.768 kHz.

(2) Port 2 mode register (PM2)

This register sets input/output for port 2 in 1-bit units.

When using the P23/PCL pin for clock output, set the output latch of PM23 and P23 to 0.

PM2 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PM2 to FFH.

Figure 19-4. Port 2 Mode Register (PM2) Format

Address: 0FF22H After Reset: FFH R/W

Symbol	7	6	5	4	3	2	1	0
PM2	PM27	PM26	PM25	PM24	PM23	PM22	PM21	PM20

PM2n	P2n Pin Input/Output Mode Selection (n = 0 to 7)
0	Output mode (output buffer ON)
1	Input mode (output buffer OFF)

CHAPTER 20 BUZZER OUTPUT FUNCTIONS

20.1 Function

This function outputs a square wave at the frequencies of 1.5 kHz, 3.1 kHz, 6.1 kHz, and 12.2 kHz. The buzzer frequency selected by the clock output control register (CKS) is output from the BUZ/P24 pin.

The following procedure outputs the buzzer frequency.

<1> Select the buzzer output frequency by using bits 5 to 7 (BCS0, BCS1, BZOE) of CKS. (In a state where the buzzer is prohibited from sounding.)
<2> Set the P24 output latch to 0.
<3> Set bit 4 (PM24) of the port 2 mode register (PM2) to 0 (set the output mode).

Caution When the output latch of P24 is set to one, the buzzer output cannot be used.

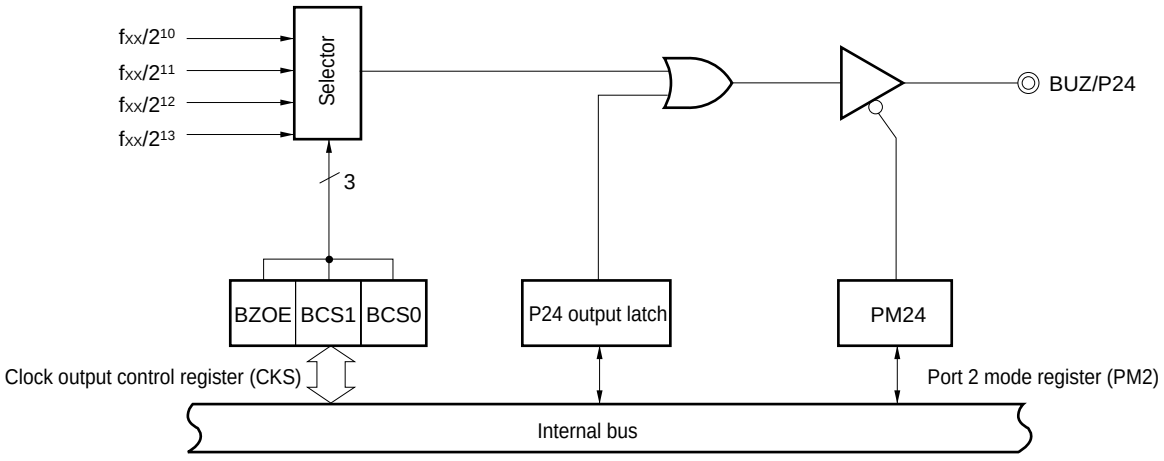
20.2 Structure

The buzzer output function consists of the following hardware.

Table 20-1. Buzzer Output Function Structure

Item	Structure
Control register	Clock output control register (CKS) Port 2 mode register (PM2)

Figure 20-1. Buzzer Output Function Block Diagram



20.3 Control Registers

The buzzer output function is controlled by the following two registers.

- Clock output control register (CKS)
- Port 2 mode register (PM2)

(1) Clock output control register (CKS)

This register sets the frequency of the buzzer output.

CKS is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets CKS to 00H.

Remark CKS has the function of setting the clock for PCL output except for the buzzer output frequency setting.

Figure 20-2. Clock Output Control Register (CKS) Format

Address: 0FF40H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
CKS	BZOE	BCS1	BCS0	CLOE	CCS3	CCS2	CCS1	CCS0

BZOE	Buzzer Output Buzzer
0	Stop buzzer output
1	Start buzzer output

BCS1	BCS0	Buzzer Output Frequency Selection
0	0	$f_{XX}/2^{10}$ (12.2 kHz)
0	1	$f_{XX}/2^{11}$ (6.1 kHz)
1	0	$f_{XX}/2^{12}$ (3.1 kHz)
1	1	$f_{XX}/2^{13}$ (1.5 kHz)

CLOE	Clock Output Control (refer to **Figure 19-3**)

CCS3	CCS2	CCS1	CCS0	Clock Output Frequency Selection (refer to **Figure 19-3**)

Remarks 1. f_{XX}: Main system clock frequency (f_X or $f_X/2$)

2. f_X : Main system clock oscillation frequency

3. Figures in parentheses apply to operation with f_{XX} = 12.5 MHz.

(2) Port 2 mode register (PM2)

This register sets port 2 I/O in 1-bit units.

When the P24/BUZ pin is used as the buzzer output function, set the output latches of PM24 and P24 to 0.

PM2 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PM2 to FFH.

Figure 20-3. Port 2 Mode Register (PM2) Format

Address: 0FF22H After Reset: FFH R/W

Symbol	7	6	5	4	3	2	1	0
PM2	PM27	PM26	PM25	PM24	PM23	PM22	PM21	PM20

PM2n	P2n Pin I/O Mode Selection (n = 0 to 7)
0	Output mode (output buffer ON)
1	Input mode (output buffer OFF)

[MEMO]

CHAPTER 21 EDGE DETECTION FUNCTION

The P00 to P05 pins have an edge detection function that can be programmed to detect the rising edge or falling edge and sends the detected edge to on-chip hardware components.

The edge detection function is always functioning, even in the STOP mode and IDLE mode.

21.1 Control Registers

- **External Interrupt Rising Edge Enable Register (EGP0), External Interrupt Falling Edge Enable Register (EGN0)**
  The EGP0 and EGN0 registers specify the effective edge to be detected by the P00 to P05 pins. They can read/write with an 8-bit manipulation instruction or a bit manipulation instruction.
  $\overline{\text{RESET}}$ input sets the EGP0 and EGN0 to 00H.

Figure 21-1. Format of External Interrupt Rising Edge Enable Register (EGP0) and External Interrupt Falling Edge Enable Register (EGN0)

Address: 0FFA0H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
EGP0	0	0	EGP5	EGP4	EGP3	EGP2	EGP1	EGP0

Address: 0FFA2H After Reset: 00H R/W

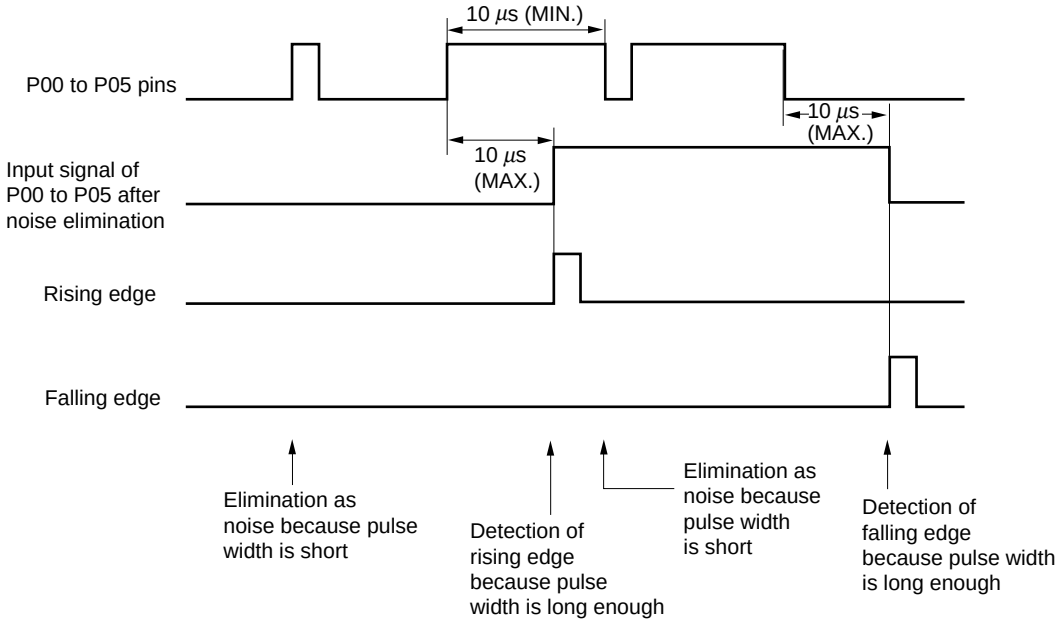
Symbol	7	6	5	4	3	2	1	0
EGN0	0	0	EGN5	EGN4	EGN3	EGN2	EGN1	EGN0

EGPn	EGNn	INTPn Pin Effective Edge (n = 0 to 5)
0	0	Interrupt disable
0	1	Falling edge
1	0	Rising edge
1	1	Both rising and falling edges

21.2 Edge Detection of P00 to P05 Pins

The P00 to P05 pins detect the edges after noise elimination by analog delay. Therefore, edge detection requires that a pulse width be maintaned for at least 10 μs.

Figure 21-2. Edge Detection of P00 to P05 Pins



Caution Since noise at the P00 to P05 pins is eliminated by analog delay, the edge is detected at most 10 μs after an actual edge is input. The delay time until edge detection is not a constant value because of differences in device characteristics.

CHAPTER 22 INTERRUPT FUNCTIONS

The μPD784225 is provided with three interrupt request service modes (refer to **Table 22-1**). These three service modes can be set as required in the program. However interrupt service by macro service can only be selected for interrupt request sources provided with the macro service processing mode shown in Table 22-2. Context switching cannot be selected for non-maskable interrupts or operand error interrupts.

Multiple-interrupt control using 4 priority levels can easily be performed for maskable vectored interrupts.

Table 22-1. Interrupt Request Service Modes

Interrupt Request Service Mode	Servicing Performed	PC & PSW Contents	Service
Vectored interrupts	Software	Saving to & restoration from stack	Executed by branching to service program at address[Note] specified by vector table
Context switching		Saving to & restoration from fixed area in register bank	Executed by automatic switching to register bank specified by vector table and branching to service program at address[Note] specified by fixed area in register bank
Macro service	Hardware (firmware)	Retained	Execution of pre-set service such as data transfers between memory and I/O

Note The start addresses of all interrupt service programs must be in the base area. If the body of a service program cannot be located in the base area, a branch instruction to the service program should be written in the base area.

22.1 Interrupt Request Sources

The μPD784225 has the 28 interrupt request sources shown in Table 22-2, with a vector table allocated to each.

Table 22-2. Interrupt Request Sources (1/2)

Type of Interrupt Request	Default Priority	Interrupt Request Generating Source	Generating Unit	Interrupt Control Register Name	Context Switching	Macro Service	Macro Service Control Word Address	Vector Table Address
Software	None	BRK instruction execution	—	—	Not possible	Not possible	—	003EH
		BRKCS instruction execution	—	—	Possible	Not possible	—	—
Operand error	None	Invalid operand in MOV STBC, #byte instruction or MOV WDM, #byte instruction, and LOCATION instruction	—	—	Not possible	Not possible	—	003CH
Non-maskable	None	NMI (pin input edge detection)	Edge detection	—	Not possible	Not possible	—	0002H
		INTWDT (watchdog timer overflow)	Watchdog timer	—	Not possible	Not possible	—	0004H

Table 22-2. Interrupt Request Sources (2/2)

Type of Interrupt Request	Default Priority	Interrupt Request Generating Source	Generating Unit	Interrupt Control Register Name	Context Switching	Macro Service	Macro Service Control Word Address	Vector Table Address
Maskable	0	INTWDTM (Watchdog timer overflow)	Watchdog timer	WDTIC	Possible	Possible	0FE06H	0006H
	1	INTP0 (Pin input edge detection)	Edge detection	PIC0			0FE08H	0008H
	2	INTP1 (Pin input edge detection)		PIC1			0FE0AH	000AH
	3	INTP2 (Pin input edge detection)		PIC2			0FE0CH	000CH
	4	INTP3 (Pin input edge detection)		PIC3			0FE0EH	000EH
	5	INTP4 (Pin input edge detection)		PIC4			0FE10H	0010H
	6	INTP5 (Pin input edge detection)		PIC5			0FE12H	0012H
	7	INTIIC0 (CSI0 I^2C bus transfer end)[Note] INTCSI0 (CSI0 3-wire transfer end)	Clock synchronous serial interface	CSIIC0			0FE16H	0016H
	8	INTSER1 (ASI1 UART reception error)	Asynchronous serial interface/ clock synchronous serial interface 1	SERIC1			0FE18H	0018H
	9	INTSR1 (ASI1 UART reception end) INTCSI1 (CSI1 3-wire transfer end)		SRIC1			0FE1AH	001AH
	10	INTST1 (ASI1 UART transmission end)		STIC1			0FE1CH	001CH
	11	INTSER2 (ASI2 UART reception error)	Asynchronous serial interface/ clock synchronous serial interface 2	SERIC2			0FE1EH	001EH
	12	INTSR2 (ASI2 UART reception end) INTCSI2 (CSI2 3-wire transfer end)		SRIC2			0FE20H	0020H
	13	INTST2 (ASI2 UART transmission end)		STIC2			0FE22H	0022H
	14	INTTM3 (Reference time interval signal from watch timer)	Watch timer	TMIC3			0FE24H	0024H
	15	INTTM00 (Match signal generation of 16-bit timer register and capture/ compare register(CR00))	Timer/ Counter	TMIC00			0FE26H	0026H
	16	INTTM01 (Match signal generation of 16-bit timer register and capture/ compare register(CR01))		TMIC01			0FE28H	0028H
	17	INTTM1 (Match signal generation of 8-bit timer/counter 1)	Timer/ counter 1	TMIC1			0FE2AH	002AH
	18	INTTM2 (Match signal generation of 8-bit timer/counter 2)	Timer/ counter 2	TMIC2			0FE2CH	002CH
	19	INTAD (A/D converter conversion end)	A/D converter	ADIC			0FE2EH	002EH
	20	INTTM5 (Match signal generation of 8-bit timer/counter 5)	Timer/ counter 5	TMIC5			0FE30H	0030H
	21	INTTM6 (Match signal generation of 8-bit timer/counter 6)	Timer/ counter 6	TMIC6			0FE32H	0032H
	22	INTWT (Watch timer overflow)	Watch timer	WTIC			0FE38H	0038H

Note μPD784225Y Subseries only

Remarks 1. The default priority is a fixed number. This indicates the order of priority when interrupt requests specified as having the same priority are generated simultaneously.

2. ASI: Asynchronous serial interface
CSI: Clock synchronous serial interface

22.1.1 Software interrupts

Interrupts by software consist of the BRK instruction which generates a vectored interrupt and the BRKCS instruction which performs context switching.

Software interrupts are acknowledged even in the interrupt disabled state, and are not subject to priority control.

22.1.2 Operand error interrupts

These interrupts are generated if there is an illegal operand in an MOV STBC, #byte instruction or MOV WDMC, #byte instruction, and LOCATION instruction.

Operand error interrupts are acknowledged even in the interrupt disabled state, and are not subject to priority control.

22.1.3 Non-maskable interrupts

A non-maskable interrupt is generated by NMI pin input or the watchdog timer.

Non-maskable interrupts are acknowledged unconditionally**Note**, even in the interrupt disabled state. They are not subject to interrupt priority control, and are of higher priority that any other interrupt.

Note Except during execution of the service program for the same non-maskable interrupt, and during execution of the service program for a higher-priority non-maskable interrupt

22.1.4 Maskable interrupts

A maskable interrupt is one subject to masking control according to the setting of an interrupt mask flag. In addition, acknowledgment enabling/disabling can be specified for all maskable interrupts by means of the IE flag in the program status word (PSW).

In addition to normal vectored interruption, maskable interrupts can be acknowledged by context switching and macro service (though some interrupts cannot use macro service: refer to **Table 22-2**).

The priority order for maskable interrupt requests when interrupt requests of the same priority are generated simultaneously is predetermined (default priority) as shown in Table 22-2. Also, multiprocessing control can be performed with interrupt priorities divided into 4 levels. However, macro service requests are acknowledged without regard to priority control or the IE flag.

22.2 Interrupt Service Modes

There are three μPD784225 interrupt service modes, as follows:

- Vectored interrupt service
- Macro service
- Context switching

22.2.1 Vectored interrupt service

When an interrupt is acknowledged, the program counter (PC) and program status word (PSW) are automatically saved to the stack, a branch is made to the address indicated by the data stored in the vector table, and the interrupt service routine is executed.

22.2.2 Macro service

When an interrupt is acknowledged, CPU execution is temporarily suspended and a data transfer is performed by hardware. Since macro service is performed without the intermediation of the CPU, it is not necessary to save or restore CPU statuses such as the program counter (PC) and program status word (PSW) contents. This is therefore very effective in improving the CPU service time (refer to **22.8 Macro Service Function**).

22.2.3 Context switching

When an interrupt is acknowledged, the prescribed register bank is selected by hardware, a branch is made to a pre-set vector address in the register bank, and at the same time the current program counter (PC) and program status word (PSW) are saved in the register bank (refer to **22.4.2 BRKCS instruction software interrupt (software context switching) acknowledgment operation** and **22.7.2 Context switching**).

Remark "Context" refers to the CPU registers that can be accessed by a program while that program is being executed. These registers include general registers, the program counter (PC), program status word (PSW), and stack pointer (SP).

22.3 Interrupt Processing Control Registers

μPD784225 interrupt processing is controlled for each interrupt request by various control registers that perform interrupt processing specification. The interrupt control registers are listed in Table 22-3.

Table 22-3. Control Registers

Register Name	Symbol	Function
Interrupt control registers	WDTIC, PIC0, PIC1, PIC2, PIC3, PIC4, PIC5, CSIIC0, SERIC1, SRIC1, STIC1, SERIC2, SRIC2, STIC2, TMIC3, TMIC00, TMIC01, TMIC1, TMIC2, ADIC, TMIC5, TMIC6, WTIC	Registers to record generation of interrupt request, control masking, specify vectored interrupt processing or macro service processing, enable or disable context switching function, and specify priority.
Interrupt mask registers	MK0 (MK0L, MK0H) MK1 (MK1L, MK1H)	Control masking of maskable interrupt request. Associated with mask control flag in interrupt control register. Can be accessed in word or byte units.
In-service priority register	ISPR	Records priority of interrupt request currently accepted.
Interrupt mode control register	IMC	Controls nesting of maskable interrupt with priority specified to lowest level (level 3).
Interrupt selection control register	SNMI	Selects whether to use input signal from P02 pin and interrupt signal from watchdog timer as maskable interrupt or NMI.
Watchdog timer mode register	WDM	Specifies priorities of interrupt by NMI pin input and overflow of watchdog timer.
Program status word	PSW	Enables or disables accepting maskable interrupt.

An interrupt control register is allocated to each interrupt source. The flags of each register perform control of the contents corresponding to the relevant bit position in the register. The interrupt control register flag names corresponding to each interrupt request signal are shown in Table 22-4.

Table 23-4. Flag List of Interrupt Control Registers for Interrupt Requests

Default Priority	Interrupt Request Signal	Interrupt Control Register					
			Interrupt Request Flag	Interrupt Mask Flag	Macro Service Enable Flag	Priority Specification Flag	Context Switching Enable Flag
0	INTWDTM	WDTIC	WDTIF	WDTMK	WDTISM	WDTPR0 WDTPR1	WDTCSE
1	INTP0	PIC0	PIF0	PMK0	PISM0	PPR00 PPR01	PCSE0
2	INTP1	PIC1	PIF1	PMK1	PISM1	PPR10 PPR11	PCSE1
3	INTP2	PIC2	PIF2	PMK2	PISM2	PPR20 PPR21	PCSE2
4	INTP3	PIC3	PIF3	PMK3	PISM3	PPR30 PPR31	PCSE3
5	INTP4	PIC4	PIF4	PMK4	PISM4	PPR40 PPR41	PCSE4
6	INTP5	PIC5	PIF5	PMK5	PISM5	PPR50 PPR51	PCSE5
7	INTIIC0 INTCSI0	CSIIC0	CSIIF0	CSIMK0	CSIISM0	CSIPR00 CSIPR01	CSICSE0
8	INTSER1	SERIC1	SERIF1	SERMK1	SERISM1	SERPR10 SERPR11	SERCSE1
9	INTSR1 INTCSI1	SRIC1	SRIF1	SRMK1	SRISM1	SRPR10 SRPR11	SRCSE1
10	INTST1	STIC1	STIF1	STMK1	STISM1	STPR10 STPR11	STCSE1
11	INTSER2	SERIC2	SERIF2	SERMK2	SERISM2	SERPR20 SERPR21	SERCSE2
12	INTSR2 INTCSI2	SRIC2	SRIF2	SRMK2	SRISM2	SRPR20 SRPR21	SRCSE2
13	INTST2	STIC2	STIF2	STMK2	STISM2	STPR20 STPR21	STCSE2
14	INTTM3	TMIC3	TMIF3	TMMK3	TMISM3	TMPR30 TMPR31	TMCSE3
15	INTTM00	TMIC00	TMIF00	TMMK00	TMISM00	TMPR000 TMPR001	TMCSE00
16	INTTM01	TMIC01	TMIF01	TMMK01	TMISM01	TMPR010 TMPR011	TMCSE01
17	INTTM1	TMIC1	TMIF1	TMMK1	TMISM1	TMPR10 TMPR11	TMCSE1
18	INTTM2	TMIC2	TMIF2	TMMK2	TMISM2	TMPR20 TMPR21	TMCSE2
19	INTAD	ADIC	ADIF	ADMK	ADISM	ADPR00 ADPR01	ADCSE
20	INTTM5	TMIC5	TMIF5	TMMK5	TMISM5	TMPR50 TMPR51	TMCSE5
21	INTTM6	TMIC6	TMIF6	TMMK6	TMISM6	TMPR60 TMPR61	TMCSE6
22	INTWT	WTIC	WTIF	WTMK	WTISM	WTPR0 WTPR1	WTCSE

22.3.1 Interrupt control registers

An interrupt control register is allocated to each interrupt source, and performs priority control, mask control, etc., for the corresponding interrupt request. The interrupt control register format is shown in Figure 22-1.

(1) Priority specification flags (××PR1/××PR0)

The priority specification flags specify the priority on an individual interrupt source basis for the 23 maskable interrupts.

Up to 4 priority levels can be specified, and a number of interrupt sources can be specified at the same level. Among maskable interrupt sources, level 0 is the highest priority.

If multiple interrupt requests are generated simultaneously among interrupt source of the same priority level, they are acknowledged in default priority order.

These flags can be manipulated bit-wise by software.

$\overline{\text{RESET}}$ input sets all bits to 1.

(2) Context switching enable flag (××CSE)

The context switching enable flag specifies that a maskable interrupt request is to be serviced by context switching.

In context switching, the register bank specified beforehand is selected by hardware, a branch is made to a vector address stored beforehand in the register bank, and at the same time the current contents of the program counter (PC) and program status word (PSW) are saved in the register bank.

Context switching is suitable for real-time processing, since execution of interrupt servicing can be started faster than with normal vectored interrupt servicing.

This flag can be manipulated bit-wise by software.

$\overline{\text{RESET}}$ input sets all bits to 0.

(3) Macro service enable flag (××ISM)

The macro service enable flag specifies whether an interrupt request corresponding to that flag is to be handled by vectored interruption or context switching, or by macro service.

When macro service processing is selected, at the end of the macro service (when the macro service counter reaches 0) the macro service enable flag is automatically cleared (0) by hardware (vectored interrupt service/ context switching service).

This flag can be manipulated bit-wise by software.

$\overline{\text{RESET}}$ input sets all bits to 0.

(4) Interrupt mask flag (××MK)

An interrupt mask flag specifies enabling/disabling of vectored interrupt servicing and macro service processing for the interrupt request corresponding to that flag.

The interrupt mask contents are not changed by the start of interrupt service, etc., and are the same as the interrupt mask register contents (refer to **22.3.2 Interrupt mask registers (MK0/MK1)**).

Macro service processing requests are also subject to mask control, and macro service requests can also be masked with this flag.

This flag can be manipulated by software.

$\overline{\text{RESET}}$ input sets all bits to 1.

(5) Interrupt request flag (××IF)

An interrupt request flag is set (1) by generation of the interrupt request that corresponds to that flag. When the interrupt is acknowledged, the flag is automatically cleared (0) by hardware.

This flag can be manipulated by software.

$\overline{\text{RESET}}$ input sets all bits to 0.

Figure 22-1. Interrupt Control Register (xxICn) (1/3)

Address : 0FFE0H to 0FFE6H, 0FFE8H After Reset : 43H R/W

Symbol	7	6	5	4	3	2	1	0
WDTIC	WDTIF	WDTMK	WDTISM	WDCSE	0	0	WDTPR1	WDTPR0
PIC0	PIF0	PMK0	PISM0	PCSE0	0	0	PPR01	PPR00
PIC1	PIF1	PMK1	PISM1	PCSE1	0	0	PPR11	PPR10
PIC2	PIF2	PMK2	PISM2	PCSE2	0	0	PPR21	PPR20
PIC3	PIF3	PMK3	PISM3	PCSE3	0	0	PPR31	PPR30
PIC4	PIF4	PMK4	PISM4	PCSE4	0	0	PPR41	PPR40
PIC5	PIF5	PMK5	PISM5	PCSE5	0	0	PPR51	PPR50
CSIIC0	CSIIF0	CSIMK0	CSIISM0	CSICSE0	0	0	CSIPR01	CSIPR00

xxIFn	Interrupt Request Generation
0	No interrupt request (Interrupt signal is not generated.)
1	Interrupt request (Interrupt signal is generated)

xxMKn	Interrupt Processing Enable/Disable
0	Interrupt processing enable
1	Interrupt processing disable

xxISMn	Interrupt Processing Mode Specification
0	Vectored interrupt processing/Context switching processing
1	Macro service processing

xxCSEn	Context Switching Processing Specification
0	Processing with vectored interrupt
1	Processing with context switching

xxPRn1	xxPRn0	Interrupt Request Priority Specification
0	0	Priority 0 (Highest priority)
0	1	Priority 1
1	0	Priority 2
1	1	Priority 3

Figure 22-1. Interrupt Control Register (xxICn) (2/3)

Address : 0FFE9H to 0FFF1H After Reset : 43H R/W

Symbol	7	6	5	4	3	2	1	0
SERIC1	SERIF1	SERMK1	SERISM1	SERCSE1	0	0	SERPR11	SERPR10
SRIC1	SRIF1	SRMK1	SRISM1	SRCSE1	0	0	SRPR11	SRPR10
STIC1	STIF1	STMK1	STISM1	STCSE1	0	0	STPR11	STPR10
SERIC2	SERIF2	SERMK2	SERISM2	SERCSE2	0	0	SERPR21	SERPR20
SRIC2	SRIF2	SRMK2	SRISM2	SRCSE2	0	0	SRPR21	SRPR20
STIC2	STIF2	STMK2	STISM2	STCSE2	0	0	STPR21	STPR20
TMIC3	TMIF3	TMMK3	TMISM3	TMCSE3	0	0	TMPR31	TMPR30
TMIC00	TMIF00	TMMK0	TMISM0	TMCSE0	0	0	TMPR001	TMPR000
TMIC01	TMIF01	TMMK01	TMISM01	TMCSE01	0	0	TMPR011	TMPR010

xxIFn	Interrupt Request Generation
0	No interrupt request (Interrupt signal is not generated.)
1	Interrupt request (Interrupt signal is generated)

xxMKn	Interrupt Processing Enable/Disable
0	Interrupt processing enable
1	Interrupt processing disable

xxISMn	Interrupt Processing Mode Specification
0	Vectored interrupt processing/Context switching processing
1	Macro service processing

xxCSEn	Context Switching Processing Specification
0	Processing with vectored interrupt
1	Processing with context switching

xxPRn1	xxPRn0	Interrupt Request Priority Specification
0	0	Priority 0 (Highest priority)
0	1	Priority 1
1	0	Priority 2
1	1	Priority 3

Figure 22-1. Interrupt Control Register (xxICn) (3/3)

Address : 0FFF2H to 0FFF6H, 0FFF9H After Reset : 43H R/W

Symbol	7	6	5	4	3	2	1	0
TMIC1	TMIF1	TMMK1	TMISM1	TMCSE1	0	0	TMPR11	TMPR10
TMIC2	TMIF2	TMMK2	TMISM2	TMCSE2	0	0	TMPR21	TMPR20
ADIC	ADIF	ADMK	ADISM	ADCSE	0	0	ADPR01	ADPR00
TMIC5	TMIF5	TMMK5	TMISM5	TMCSE5	0	0	TMPR51	TMPR50
TMIC6	TMIF6	TMMK6	TMISM6	TMCSE6	0	0	TMPR61	TMPR60
WTIC	WTIF	WTMK	WTISM	WTCSE	0	0	WTPR1	WTPR0

xxIFn	Interrupt Request Generation
0	No interrupt request (Interrupt signal is not generated.)
1	Interrupt request (Interrupt signal is generated)

xxMKn	Interrupt Processing Enable/Disable
0	Interrupt processing enable
1	Interrupt processing disable

xxISMn	Interrupt Processing Mode Specification
0	Vectored interrupt processing/Context switching processing
1	Macro service processing

xxCSEn	Context Switching Processing Specification
0	Processing with vectored interrupt
1	Processing with context switching

xxPRn1	xxPRn0	Interrupt Request Priority Specification
0	0	Priority 0 (Highest priority)
0	1	Priority 1
1	0	Priority 2
1	1	Priority 3

22.3.2 Interrupt mask registers (MK0, MK1)

The MK0 and MK1 are composed of interrupt mask flags. MK0 and MK1 are 16-bit registers which can be manipulated as a 16-bit unit. MK0 can be manipulated in 8 bit units using MK0L and MK0H, and similarly MK1 can be manipulated using MK1L and MK1H.

In addition, each bit of the MK0 and MK1 can be manipulated individually with a bit manipulation instruction. Each interrupt mask flag controls enabling/disabling of the corresponding interrupt request.

When an interrupt mask flag is set (1), acknowledgment of the corresponding interrupt request is disabled.

When an interrupt mask flag is cleared (0), the corresponding interrupt request can be acknowledged as a vectored interrupt or macro service request.

Each interrupt mask flag in the MK0 and MK1 is the same flag as the interrupt mask flag in the interrupt control register. The MK0 and MK1 are provided for en bloc control of interrupt masking.

After $\overline{\text{RESET}}$ input, the MK0 and MK1 are set to FFFFH, and all maskable interrupts are disabled.

Figure 22-2. Format of Interrupt Mask Registers (MK0, MK1)

<Byte access>

Address : 0FFACH to 0FFAFH After Reset : FFH R/W

Symbol	7	6	5	4	3	2	1	0
MK0L	1	PMK5	PMK4	PMK3	PMK2	PMK1	PMK0	WDTMK
MK0H	TMMK3	STMK2	SRMK2	SERMK2	STMK1	SRMK1	SERMK1	CSIMK0
MK1L	1	TMMK6	TMMK5	ADMK	TMMK2	TMMK1	TMMK01	TMMK00
MK1H	1	1	1	1	1	1	WTMK	1

xxMKn	Interrupt Request Enable/Disable
0	Interrupt processing enable
1	Interrupt processing disable

<Word access>

Address : 0FFACH, 0FFAEH After Reset : FFFFH R/W

Symbol	15	14	13	12	11	10	9	8
MK0	TMMK3	STMK2	SRMK2	SERMK2	STMK1	SRMK1	SERMK1	CSIMK0
	7	6	5	4	3	2	1	0
	1	PMK5	PMK4	PMK3	PMK2	PMK1	PMK0	WDTMK
	15	14	13	12	11	10	9	8
MK1	1	1	1	1	1	1	WTMK	1
	7	6	5	4	3	2	1	0
	1	TMMK6	TMMK5	ADMK	TMMK2	TMMK1	TMMK01	TMMK00

xxMKn	Interrupt Request Enable/Disable
0	Interrupt processing enable
1	Interrupt processing disable

22.3.3 In-service priority register (ISPR)

The ISPR shows the priority level of the maskable interrupt currently being serviced and the non-maskable interrupt being processed. When a maskable interrupt request is acknowledged, the bit corresponding to the priority of that interrupt request is set (1), and remains set until the service program ends. When a non-maskable interrupt is acknowledged, the bit corresponding to the priority of that non-maskable interrupt is set (1), and remains set until the service program ends.

When an RETI instruction or RETCS instruction is executed, the bit, among those set (1) in the ISPR, that corresponds to the highest-priority interrupt request is automatically cleared (0) by hardware.

The contents of the ISPR are not changed by execution of an RETB or RETCSB instruction.

$\overline{\text{RESET}}$ input clears the ISPR register to 00H.

Figure 22-3. Format of In-Service Priority Register (ISPR)

Address : 0FFA8H After Reset : 00H R

Symbol	7	6	5	4	3	2	1	0
ISPR	NMIS	WDTS	0	0	ISPR3	ISPR2	ISPR1	ISPR0

NMIS	NMI Processing Status
0	NMI interrupt is not accepted.
1	NMI interrupt is accepted.

WDTS	Watchdog Timer Interrupt Processing Status
0	Watchdog timer interrupt is not accepted.
1	Watchdog timer interrupt is accepted.

ISPRn	Priority Level (n = 0 to 3)
0	Interrupt of priority level n is not accepted.
1	Interrupt of priority level n is accepted.

Caution The in-service priority register (ISPR) is a read-only register. The microcontroller may malfunction if this register is written.

22.3.4 Interrupt mode control register (IMC)

The IMC contains the PRSL flag. The PRSL flag specifies enabling/disabling of nesting of maskable interrupts for which the lowest priority level (level 3) is specified.

When the IMC is manipulated, the interrupt disabled state (DI state) should be set first to prevent misoperation.

The IMC can be read or written to with an 8-bit manipulation instruction or bit manipulation instruction.

$\overline{\text{RESET}}$ input sets the IMC register to 80H.

Figure 22-4. Format of Interrupt Mode Control Register (IMC)

Address : 0FFAAH After Reset : 80H R

Symbol	7	6	5	4	3	2	1	0
IMC	PRSL	0	0	0	0	0	0	0

PRSL	Nesting Control of Maskable Interrupt (lowest level)
0	Interrupts with level 3 (lowest level) can be nested.
1	Nesting of interrupts with level 3 (lowest level) is disabled.

22.3.5 Watchdog timer mode register (WDM)

The WDT4 bit of the WDM specifies the priority of NMI pin input non-maskable interrupts and watchdog timer overflow non-maskable interrupts.

The WDM can be written to only by a dedicated instruction. This dedicated instruction, MOV WDM, #byte, has a special code configuration (4 bytes), and a write is not performed unless the 3rd and 4th bytes of the operation code are mutual complements.

If the 3rd and 4th bytes of the operation code are not mutual 1's complements, a write is not performed and an operand error interrupt is generated. In this case, the return address saved in the stack area is the address of the instruction that was the source of the error, and thus the address that was the source of the error can be identified from the return address saved in the stack area.

If recovery from an operand error is simply performed by means of an RETB instruction, an endless loop will result.

As an operand error interrupt is only generated in the event of an inadvertent program loop (with the NEC assembler, RA78K4, only the correct dedicated instruction is generated when MOV WDM, #byte is written), system initialization should be performed by the program.

Other write instructions (MOV WDM, A; AND WDM, #byte; SET1 WDM.7, etc.) are ignored and do not perform any operation. That is, a write is not performed to the WDM, and an interrupt such as an operand error interrupt is not generated.

The WDM can be read at any time by a data transfer instruction.

$\overline{\text{RESET}}$ input clears the WDM register to 00H.

Figure 22-5. Format of Watchdog Timer Mode Register (WDM)

Address : 0FFC2H After Reset : 00H R/W

Symbol	7	6	5	4	3	2	1	0
WDM	RUN	0	0	WDT4	0	WDT2	WDT1	0

RUN	Specifies Operation of Watchdog Timer (refer to **Figure 12-2**).

WDT4	Priority of Watchdog Timer Interrupt Request
0	Watchdog timer interrupt request < NMI pin input interrupt request
1	Watchdog timer interrupt request > NMI pin input interrupt request

WDT2	WDT1	Specifies Count Clock of Watchdog Timer (refer to **Figure 12-2**).

Caution The watchdog timer mode register (WDM) can be written only by using a dedicated instruction (MOV WDM, #byte).

22.3.6 Interrupt selection control register (SNMI)

The SNMI selects whether to use and interrupt request signals from the watchdog timer inputs from the P02 pin as maskable interrupt signals or non-maskable interrupts.

Since the bit of this register can be set (1) only once after reset, the bit should be cleared (0) by reset.

The SNMI is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets the SNMI to 00H.

Figure 22-6. Format of Interrupt Selection Control Register (SNMI)

Address : 0FFA9H After Reset : 00H R/W

Symbol	7	6	5	4	3	2	1	0
SNMI	0	0	0	0	0	0	SWDT	SNMI

SWDT	Watchdog Timer Interrupt Selection
0	Use as non-maskable interrupt. Interrupt processing cannot be disabled with interrupt mask register.
1	Use as maskable interrupt. Vectored interrupts and macro service can be used. Interrupt processing can be disabled with interrupt mask register.

SNMI	P02 Pin Function Selection
0	Use as INTP2. Vectored interrupts and macro service can be used. Interrupt processing can be disabled with interrupt mask register. At this time, the standby mode with the P02 pin is accomplished with a maskable interrupt.
1	Use as MNI. Interrupt processing cannot be disabled with interrupt mask register. At this time, release of the standby mode with the P02 pin is accomplished with NMI.

22.3.7 Program status word (PSW)

The PSW is a register that holds the current status regarding instruction execution results and interrupt requests. The IE flag that sets enabling/disabling of maskable interrupts is mapped in the low-order 8 bits of the PSW (PSWL).

PSWL can be read or written to with an 8-bit manipulation instruction, and can also be manipulated with a bit manipulation instruction or dedicated instruction (EI/DI).

When a vectored interrupt is acknowledged or a BRK instruction is executed, PSWL is saved to the stack and the IE flag is cleared (0). PSWL is also saved to the stack by the PUSH PSW instruction, and is restored from the stack by the RETI, RETB and POP PSW instructions.

When context switching or a BRKCS instruction is executed, PSWL is saved to a fixed area in the register bank, and the IE flag is cleared (0). PSWL is restored from the fixed area in the register bank by an RETCSI or RETCSB instruction.

$\overline{\text{RESET}}$ input sets PSWL to 00H.

Figure 22-7. Format of Program Status Word (PSWL)

After Reset : 00H

Symbol	7	6	5	4	3	2	1	0
PSWL	S	Z	RSS	AC	IE	P/V	0	CY

S	Used for normal instruction execution
Z	
RSS	
AC	

IE	Enable or Disable Accepting Interrupt
0	Disable
1	Enable

P/V	Used for normal instruction execution
CY	

22.4 Software Interrupt Acknowledgment Operations

A software interrupt is acknowledged in response to execution of a BRK or BRKCS instruction. Software interrupts cannot be disabled.

22.4.1 BRK instruction software interrupt acknowledgment operation

When a BRK instruction is executed, the program status word (PSW) and program counter (PC) are saved in that order to the stack, the IE flag is cleared (0), the vector table (003EH/003FH) contents are loaded into the low-order 16 bits of the PC, and 0000B into the high-order 4 bits, and a branch is performed (the start of the service program must be in the base area).

The RETB instruction must be used to return from a BRK instruction software interrupt.

Caution The RETI instruction must not be used to return from a BRK instruction software interrupt.

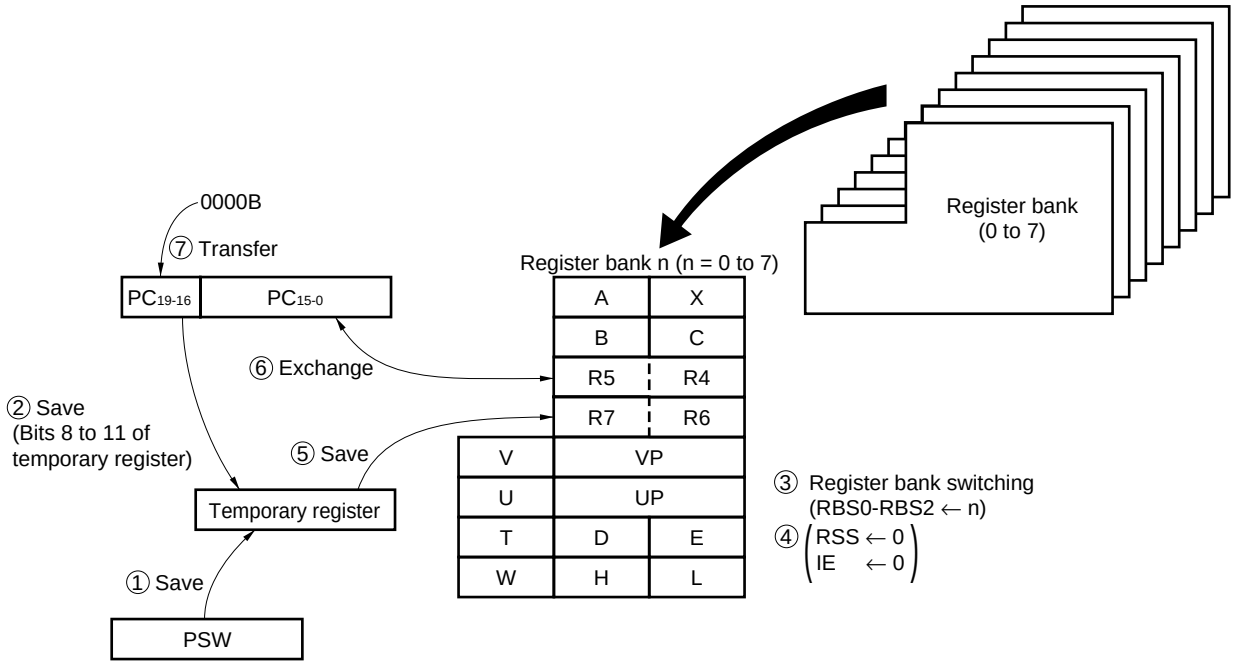
22.4.2 BRKCS instruction software interrupt (software context switching) acknowledgment operation

The context switching function can be initiated by executing a BRKCS instruction.

The register bank to be used after context switching is specified by the BRKCS instruction operand.

When a BRKCS instruction is executed, the program branches to the start address of the interrupt service program (which must be in the base area) stored beforehand in the specified register bank, and the contents of the program status word (PSW) and program counter (PC) are saved in the register bank.

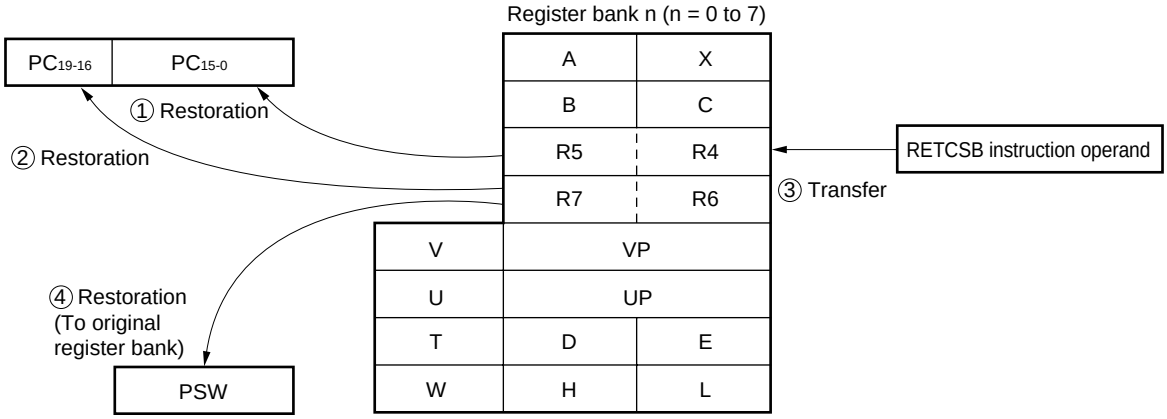
Figure 22-8. Context Switching Operation by Execution of a BRKCS Instruction



The RETCSB instruction is used to return from a software interrupt due to a BRKCS instruction. The RETCSB instruction must specify the start address of the interrupt service program for the next time context switching is performed by a BRKCS instruction. This interrupt service program start address must be in the base area.

Caution The RETCS instruction must not be used to return from a BRKCS instruction software interrupt.

Figure 22-9. Return from BRKCS Instruction Software Interrupt (RETCSB Instruction Operation)



22.5 Operand Error Interrupt Acknovwedgment Operation

An operand error interrupt is generated when the data obtained by inverting all the bits of the 3rd byte of the operand of an MOV STBC, #byte instruction or LOCATION instruction or an MOV WDM,#byte instruction does not match the 4th byte of the operand. Operand error interrupts cannot be disabled.

When an operand error interrupt is generated, the program status word (PSW) and the start address of the instruction that caused the error are saved to the stack, the IE flag is cleared (0), the vector table value is loaded into the program counter (PC), and a branch is performed (within the base area only).

As the address saved to the stack is the start address of the instruction in which the error occurred, simply writing an RETB instruction at the end of the operand error interrupt service program will result in generation of another operand error interrupt. You should therefore either process the address in the stack or inifialize the program by referring to **22.12 Restoring Interrupt Function to Initial State**.

22.6 Non-maskable Interrupt Acknowledgment Operation

Non-maskable interrupts are acknowledged even in the interrupt disabled state. Non-maskable interrupts can be acknowledged at all times except during execution of the service program for an identical non-maskable interrupt or a non-maskable interrupt of higher priority.

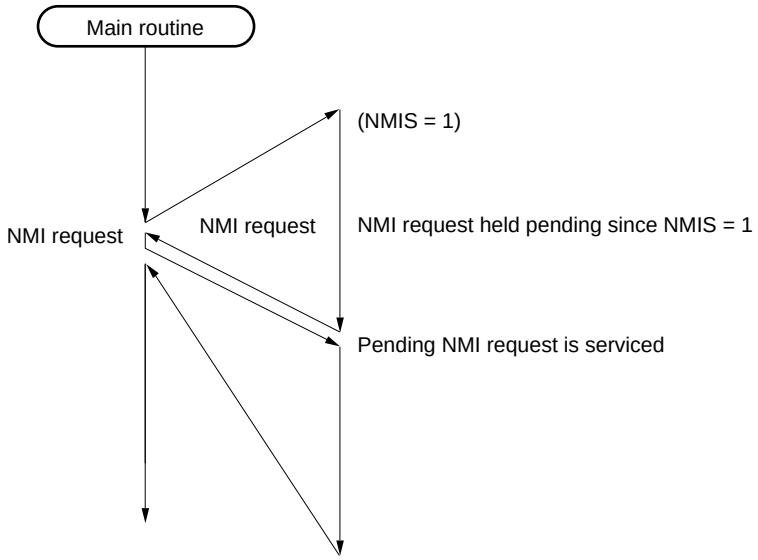
The relative priorities of non-maskable interrupts are set by the WDT4 bit of the watchdog timer mode register (WDM) (see **22.3.5 Watchdog timer mode register (WDM)**).

Except in the cases described in **22.9 When Interrupt Requests and Macro Service are Temporarily Held Pending**, a non-maskable interrupt request is acknowledged immediately. When a non-maskable interrupt request is acknowledged, the program status word (PSW) and program counter (PC) are saved in that order to the stack, the IE flag is cleared (0), the in-service priority register (ISPR) bit corresponding to the acknowledged non-maskable interrupt is set (1), the vector table contents are loaded into the PC, and a branch is performed. The ISPR bit that is set (1) is the NMIS bit in the case of a non-maskable interrupt due to edge input to the NMI pin, and the WDTS bit in the case of watchdog timer overflow.

When the non-maskable interrupt service program is executed, non-maskable interrupt requests of the same priority as the non-maskable interrupt currently being executed and non-maskable interrupts of lower priority than the non-maskable interrupt currently being executed are held pending. A pending non-maskable interrupt is acknowledge after completion of the non-maskable interrupt service program currently being executed (after execution of the RETI instruction). However, even if the same non-maskable interrupt request is generated more than once during execution of the non-maskable interrupt service program, only one non-maskable interrupt is acknowledged after completion of the non-maskable interrupt service program.

Figure 22-10. Non-Maskable Interrupt Request Acknowledgment Operations (1/2)

(a) When a new NMI request is generated during NMI service program execution



(b) When a watchdog timer interrupt request is generated during NMI service program execution (when the watchdog timer interrupt priority is higher (when WDT4 in the WDM = 1))

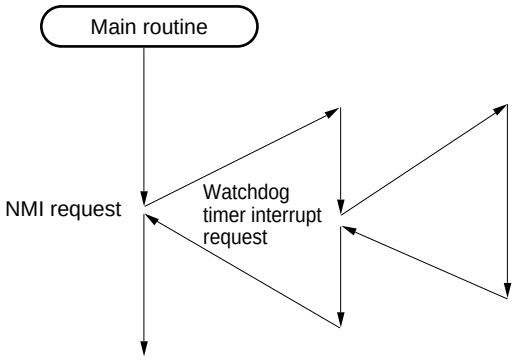
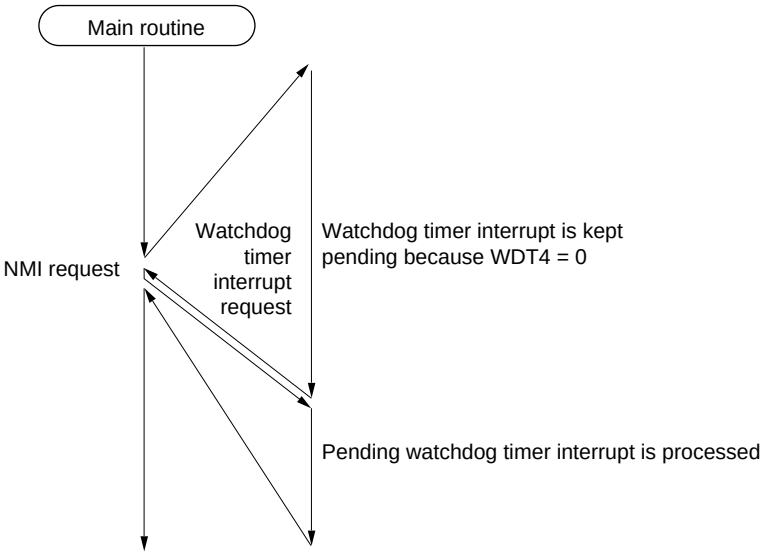
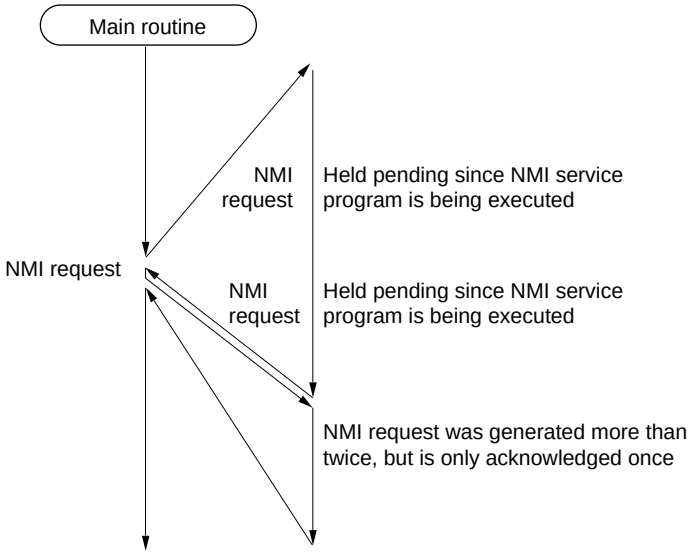


Figure 22-10. Non-Maskable Interrupt Request Acknowledgment Operations (2/2)

(c) When a watchdog timer interrupt request is generated during NMI service program execution (when the NMI interrupt priority is higher (when WDT4 in the WDM = 0))



(d) When an NMI request is generated twice during NMI service program execution



Cautions 1. **Macro service requests are acknowledged and serviced even during execution of a non-maskable interrupt service program. If you do not want macro service processing to be performed during a non-maskable interrupt service program, you should manipulate the interrupt mask register in the non-maskable interrupt service program to prevent macro service generation.**

2. **The RETI instruction must be used to return from a non-maskable interrupt. Subsequent interrupt acknowledgment will not be performed normally if a different instruction is used. Refer to Section 22.12 Restoring Interrupt Function to Initial State when a program is to be restarted from the initial status after a non-maskable interrupt acknowledgement.**

3. **Non-maskable interrupts are always acknowledged, except during non-maskable interrupt service program execution (except when a high non-maskable interrupt request is generated during execution of a low-priority non-maskable interrupt service program) and for a certain period after execution of the special instructions shown in 22.9. Therefore, a non-maskable interrupt will be acknowledged even when the stack pointer (SP) value is undefined, in particular after reset release, etc. In this case, depending on the value of the SP, it may happen that the program counter (PC) and program status word (PSW) are written to the address of a write-inhibited special function register (SFRs) (see Table 3.6 in 3.9 Special Function Registers (SFRs)), and the CPU becomes deadlocked, or an unexpected signal is output from a pin, or the PC and PSW are written to an address in which RAM is not mounted, with the result that the return from the non-maskable interrupt service program is not performed normally and a software upsets occurs.**

Therefore, the program following $\overline{\text{RESET}}$ release must be as shown below.

```
        CSEG AT 0
        DW   STRT
        CSEG BASE
STRT:
        LOCATION   0FH; or LOCATION 0
        MOVG SP, #imm24
```

22.7 Maskable Interrupt Acknowledgment Operation

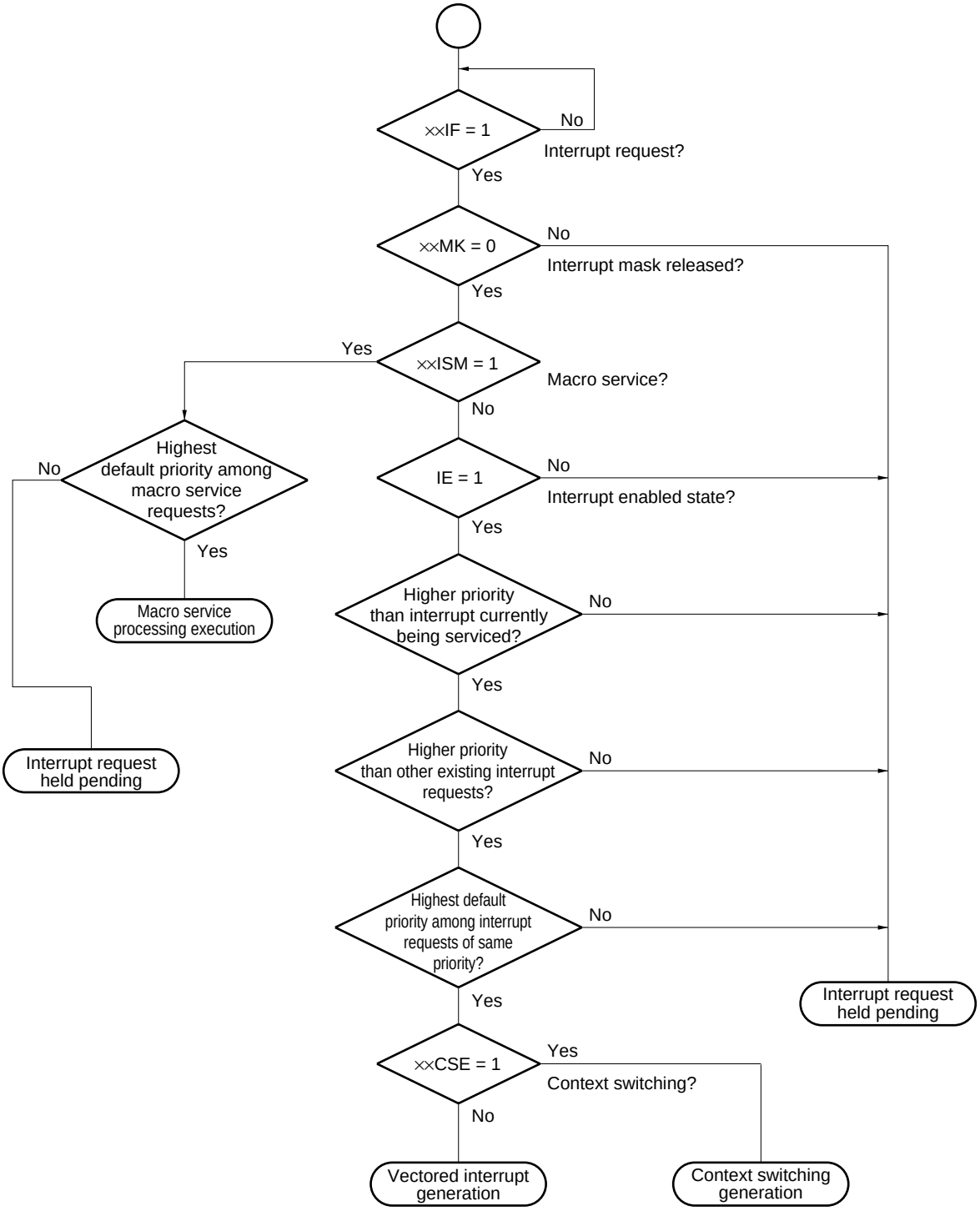
A maskable interrupt can be acknowledged when the interrupt request flag is set (1) and the mask flag for that interrupt is cleared (0). When servicing is performed by macro service, the interrupt is acknowledged and serviced by macro service immediately. In the case of vectored interruption and context switching, an interrupt is acknowledged in the interrupt enabled state (when the IE flag is set (1)) if the priority of that interrupt is one for which acknowledgment is permitted.

If maskable interrupt requests are generated simultaneously, the interrupt for which the highest priority is specified by the priority specification flag is acknowledged. If the interrupts have the same priority specified, they are acknowledged in accordance with their default priorities.

A pending interrupt is acknowledged when a state in which it can be acknowledged is established.

The interrupt acknowledgment algorithm is shown in Figure 22-11.

Figure 22-11. Interrupt Acknowledgment Processing Algorithm



22.7.1 Vectored interruption

When a vectored interruption maskable interrupt request is acknowledged, the program status word (PSW) and program counter (PC) are saved in that order to the stack, the IE flag is cleared (0) (the interrupt disabled state is set), and the in-service priority register (ISPR) bit corresponding to the priority of the acknowledged interrupt is set (1). Also, data in the vector table predetermined for each interrupt request is loaded into the PC, and a branch is performed. The return from a vectored interrupt is performed by means of the RETI instruction.

Caution When a maskable interrupt is acknowledged by vectored interruption, the RETI instruction must be used to return from the interrupt. Subsequent interrupt acknowledgment will not be performed normally if a different instruction is used.

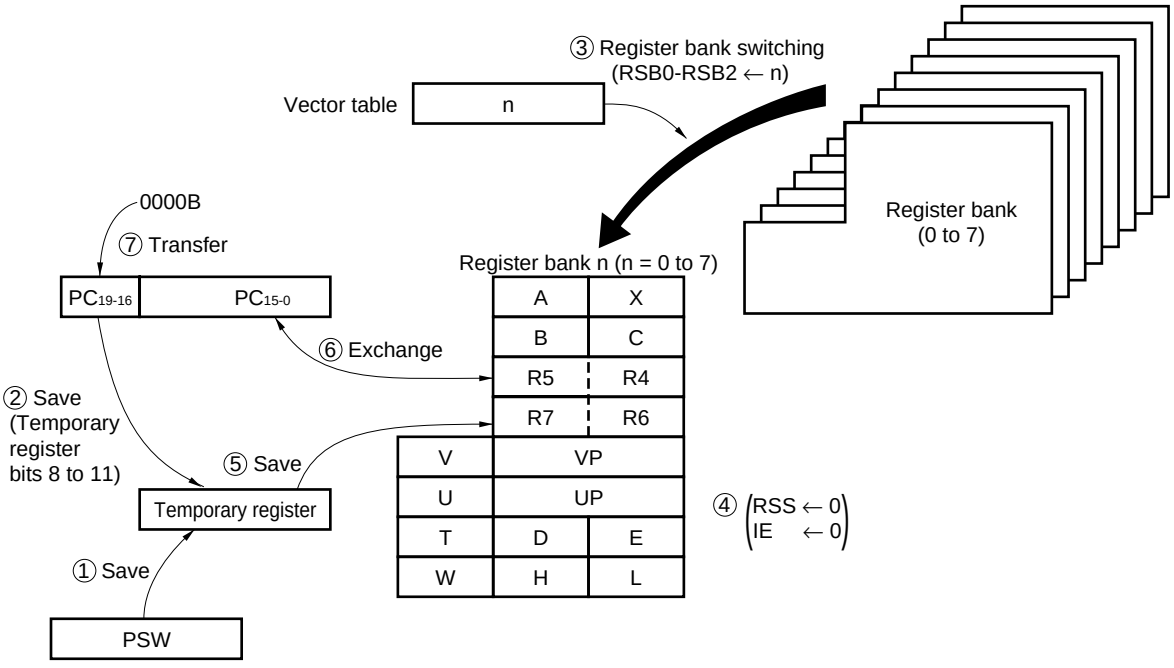
22.7.2 Context switching

Initiation of the context switching function is enabled by setting (1) the context switching enable flag of the interrupt control register.

When an interrupt request for which the context switching function is enabled is acknowledged, the register bank specified by 3 bits of the lower address (even address) of the corresponding vector table address is selected.

The vector address stored beforehand in the selected register bank is transferred to the program counter (PC), and at the same time the contents of the PC and program status word (PSW) up to that time are saved in the register bank and branching is performed to the interrupt service program.

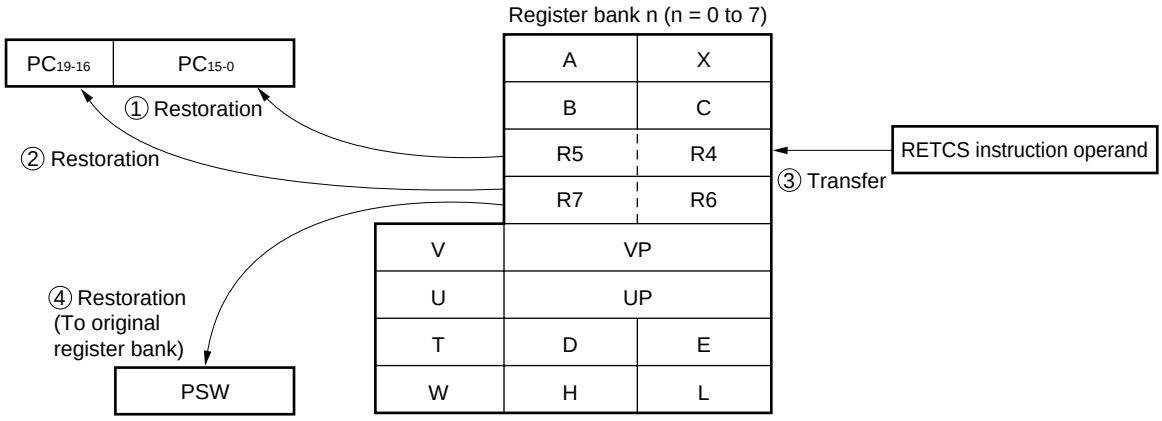
Figure 22-12. Context Switching Operation by Generation of an Interrupt Request



The RETCS instruction is used to return from an interrupt that uses the context switching function. The RETCS instruction must specify the start address of the interrupt service program to be executed when that interrupt is acknowledged next. This interrupt service program start address must be in the base area.

Caution The RETCS instruction must be used to return from an interrupt serviced by context switching. Subsequent interrupt acknowledgment will not be performed normally if a different instruction is used.

Figure 22-13. Return from Interrupt that Uses Context Switching by Means of RETCS Instruction



22.7.3 Maskable interrupt priority levels

The μPD784225 performs multiple interrupt servicing in which an interrupt is acknowledged during servicing of another interrupt. Multiple interrupts can be controlled by priority levels.

There are two kinds of priority control, control by default priority and programmable priority control in accordance with the setting of the priority specification flag. In priority control by means of default priority, interrupt service is performed in accordance with the priority preassigned to each interrupt request (default priority) (refer to **Table 22-2**). In programmable priority control, interrupt requests are divided into four levels according to the setting of the priority specification flag. Interrupt requests for which multiple interruption is permitted are shown in Table 22-5.

Since the IE flag is cleared (0) automatically when an interrupt is acknowledged, when multiple interruption is used, the IE flag should be set (1) to enable interrupts by executing an IE instruction in the interrupt service program, etc.

Table 22-5. Multiple Interrupt Servicing

Priority of Interrupt Currently Being Acknowledged	ISPR Value	IE Flag in PSW	PRSL in IMC Register	Acknowledgeable Maskable Interrupts
No interrupt being acknowledged	00000000	0	×	• All macro service only
		1	×	• All maskable interrupts
3	00001000	0	×	• All macro service only
		1	0	• All maskable interrupts
		1	1	• All macro service • Maskable interrupts specified as priority 0/1/2
2	0000×100	0	×	• All macro service only
		1	×	• All macro service • Maskable interrupts specified as priority 0/1
1	0000××10	0	×	• All macro service only
		1	×	• All macro service • Maskable interrupts specified as priority 0
0	0000×××1	×	×	• All macro service only
Non-maskable interrupts	1000×××× 0100×××× 1100××××	×	×	• All macro service only

Figure 22-14. Examples of Servicing When Another Interrupt Request is Generated During Interrupt Service (1/3)

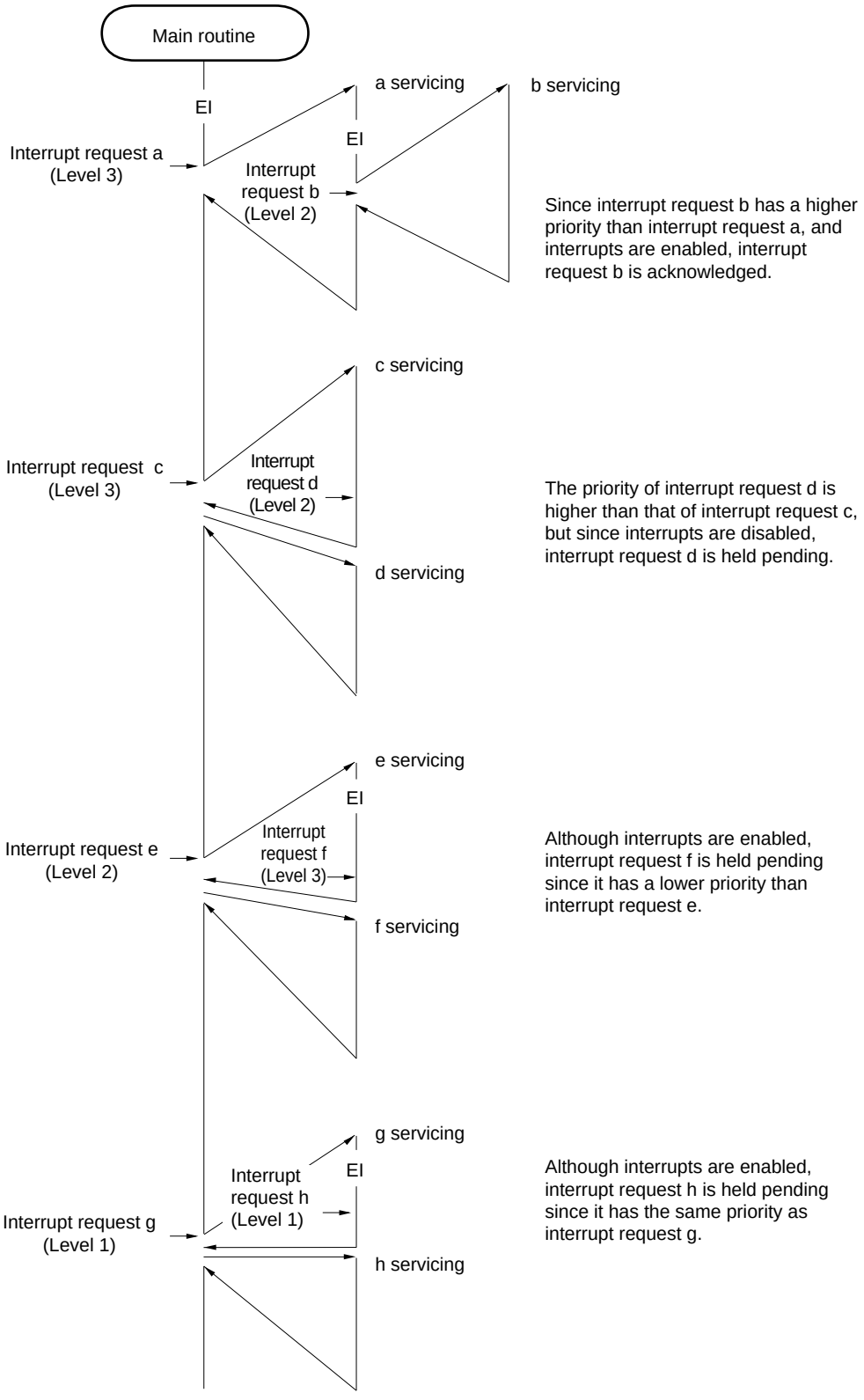
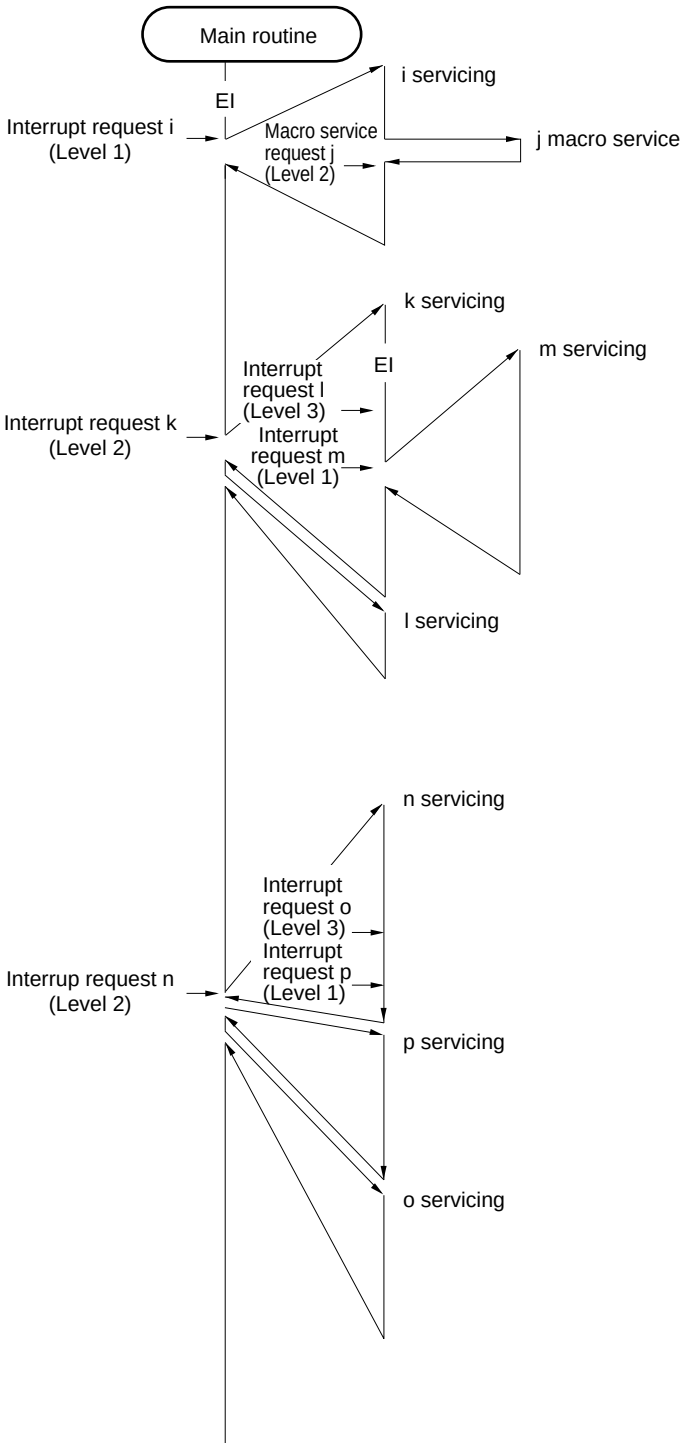


Figure 22-14. Examples of Servicing When Another Interrupt Request is Generated During Interrupt Service (2/3)

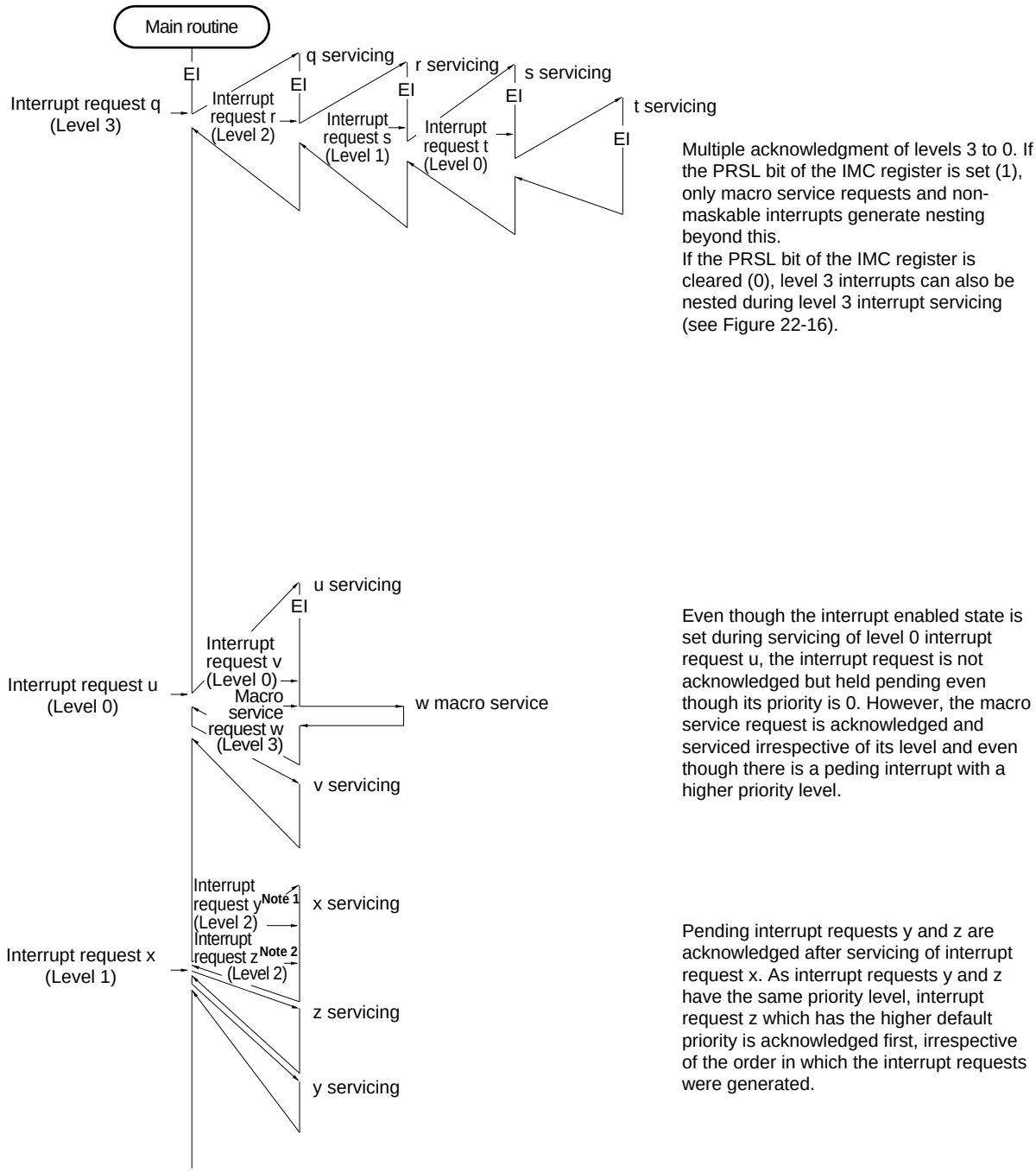


The macro service request is serviced irrespective of interrupt enabling/disabling and priority.

The interrupt request is held peding since it has a lower priority than interrupt request k. Interrupt request m generated after interrupt request l has a higher priority, and is therefore acknowledged first.

Since servicing of interrupt request n performed in the interrupt disabled state, interrupt requests o and p are held pending.
After interrupt request n servicing, the pending interrupt requests are acknowledged. Although interrupt request o was generated first, interrupt request p has a higher priority and is therefore acknowledged first.

Figure 22-14. Examples of Servicing When Another Interrupt Request is Generated During Interrupt Service (3/3)



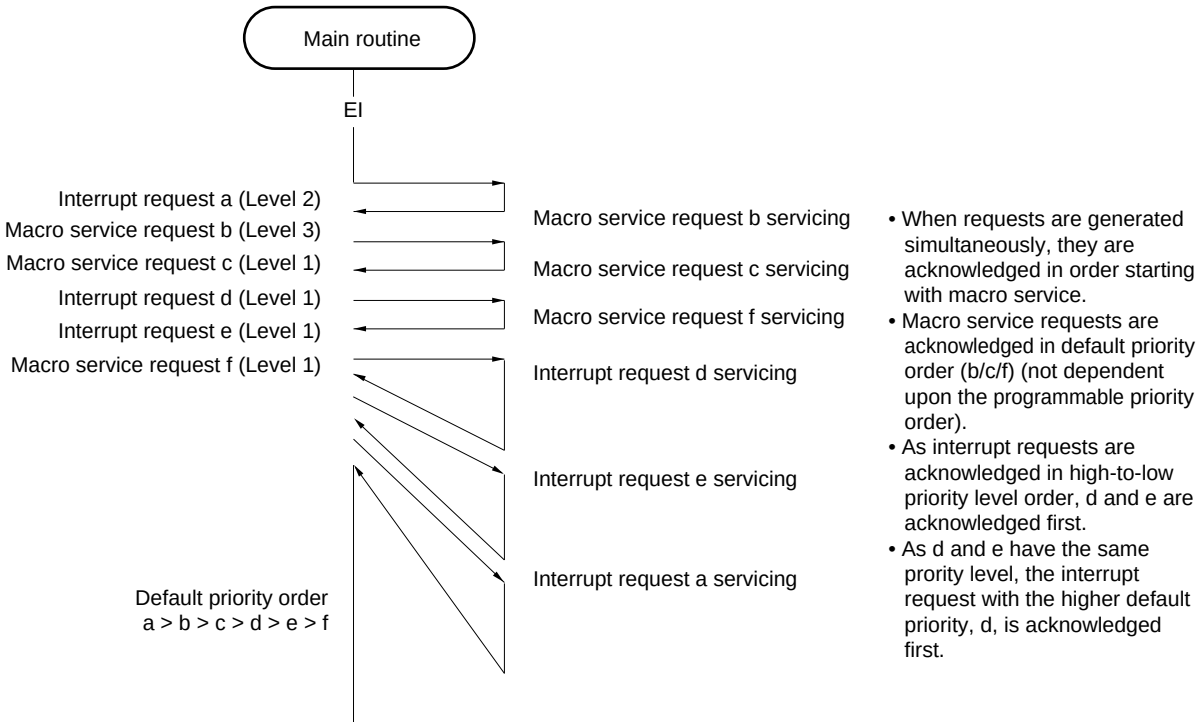
Notes 1. Low default priority

2. High default priority

Remarks 1. “a” to “z” in the figure above are arbitrary names used to differentiate between the interrupt requests and macro service requests.

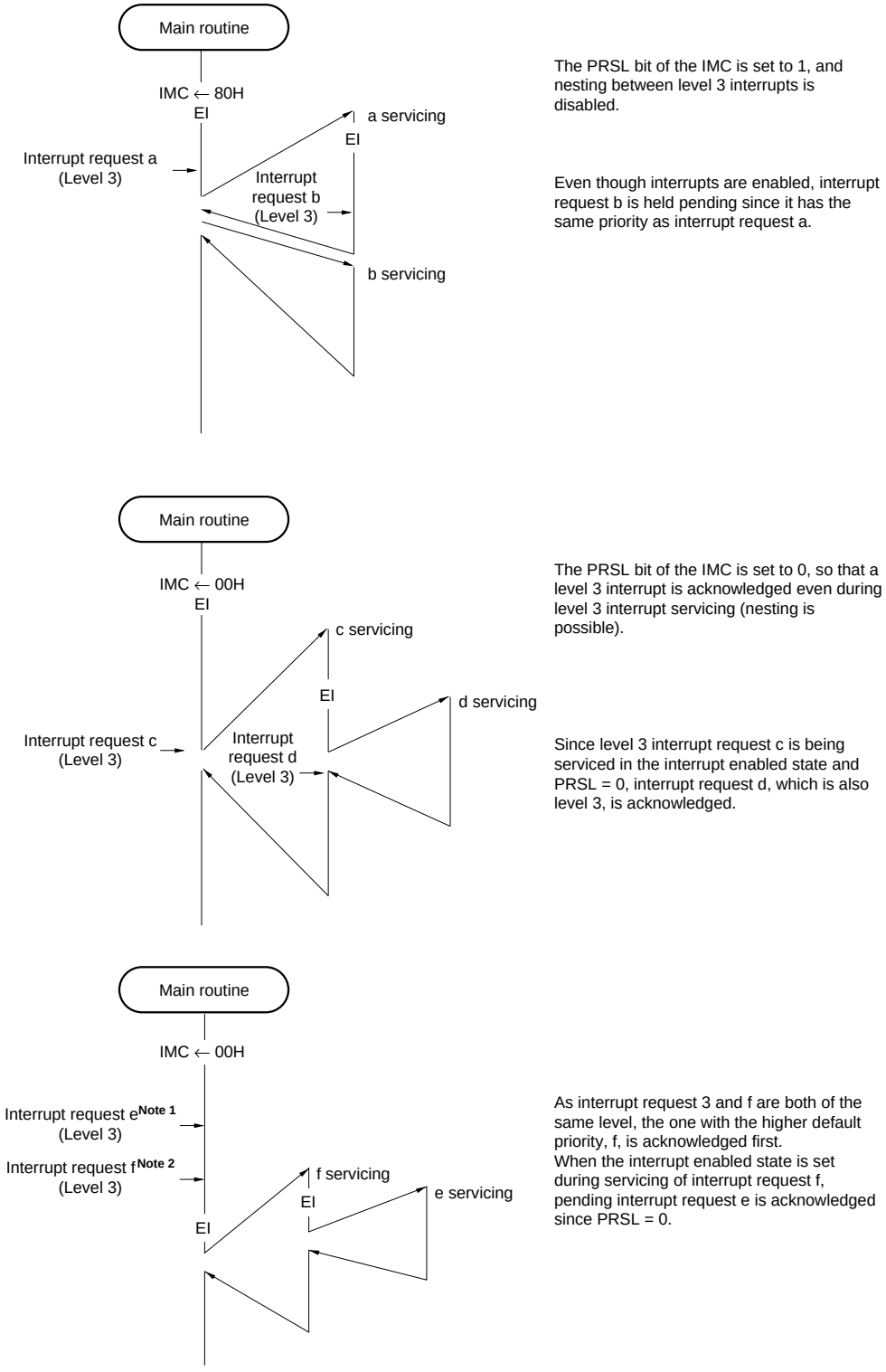
2. High/low default priorities in the figure indicate the relative priority levels of the two interrupt requests.

Figure 22-15. Examples of Servicing of Simultaneously Generated Interrupts Request



Remark “a” to “f” in the figure above are arbitrary names used to differentiate between the interrupt requests and macro service requests.

Figure 22-16. Differences in Level 3 Interrupt Acknowledgment According to IMC Register Setting



Notes 1. Low default priority

2. High default priority

Remarks 1. “a” to “f” in the figure above are arbitrary names used to differentiate between the interrupt requests and macro service requests.

2. High/low default priorities in the figure indicate the relative priority levels of the two interrupt requests.

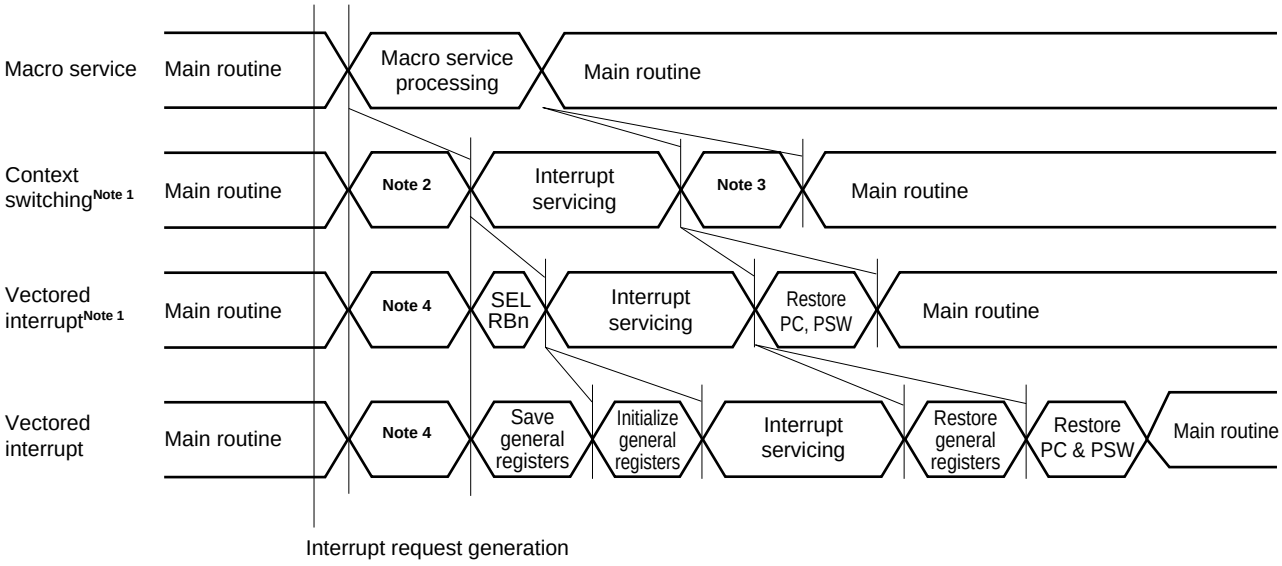
22.8 Macro Service Function

22.8.1 Outline of macro service function

Macro service is one method of servicing interrupts. With a normal interrupt, the program counter (PC) and program status word (PSW) are saved, and the start address of the interrupt service program is loaded into the PC, but with macro service, different processing (mainly data transfers) is performed instead of this processing. This enables interrupt requests to be responded to quickly, and moreover, since transfer processing is faster than processing by a program, the processing time can also be reduced.

Also, since a vectored interrupt is generated after processing has been performed the specified number of times, another advantage is that vectored interrupt programs can be simplified.

Figure 22-17. Differences between Vectored Interrupt and Macro Service Processing



Notes 1. When register bank switching is used, and an initial value has been set in the register beforehand

2. Register bank switching by context switching, saving of PC and PSW

3. Register bank, PC and PSW restoration by context switching

4. PC and PSW saved to the stack, vector address loaded into PC

22.8.2 Types of macro service

Macro service can be used with the 23 kinds of interrupts shown in Table 22-6. There are four kinds of operation, which can be used to suit the application.

Table 22-6. Interrupts for Which Macro Service Can be Used

Default Priority	Interrupt Request Generation Source	Generating Unit	Macro Service Control Word Address
0	INTWDTM (Watchdog timer overflow)	Watchdog timer	0FE06H
1	INTP0 (Pin input edge detection)	Edge detection	0FE08H
2	INTP1 (Pin input edge detection)		0FE0AH
3	INTP2 (Pin input edge detection)		0FE0CH
4	INTP3 (Pin input edge detection)		0FE0EH
5	INTP4 (Pin input edge detection)		0FE10H
6	INTP5 (Pin input edge detection)		0FE12H
7	INTIIC0 (CSI0 I^2C bus transfer end)**Note**	Clock synchronous serial interface	0FE16H
	INTCSI0 (CSI0 3-wire transfer end)		
8	INTSER1 (ASI1 UART reception error)	Asynchronous serial interface/clock synchronous serial interface 1	0FE18H
9	INTSR1 (ASI1 UART reception end)		0FE1AH
	INTCSII (CSI1 3-wire transfer end)		
10	INTST1 (ASI1 UART transmission end)		0FE1CH
11	INTSER2 (ASI2 UART reception error)	Asynchronous serial interface/clock synchronous serial interface 2	0FE1EH
12	INTSR2 (ASI2 UART reception end)		0FE20H
	INTCSI2 (CSI2 3-wire transfer end)		
13	INTST2 (ASI2 UART transmission end)		0FE22H
14	INTTM3 (Reference time interval signal from watch timer)	Watch timer	0FE24H
15	INTTM00 (Match signal generation of 16-bit timer register and capture/compare register(CR00))	Timer/Counter	0FE26H
16	INTTM01 (Match signal generation of 16-bit timer register and capture/compare register(CR01))		0FE28H
17	INTTM1 (Match signal generation of 8-bit timer/counter 1)	Timer/counter 1	0FE2AH
18	INTTM2 (Match signal generation of 8-bit timer/counter 2)	Timer/counter 2	0FE2CH
19	INTAD (A/D converter conversion end)	A/D converter	0FE2EH
20	INTTM5 (Match signal generation of 8-bit timer/counter 5)	Timer/counter 5	0FE30H
21	INTTM6 (Match signal generation of 8-bit timer/counter 6)	Timer/counter 6	0FE32H
22	INTWT (Watch timer overflow)	Watch timer	0FE38H

Note μPD784225Y Subseries only

Remarks 1. The default priority is a fixed number. This indicates the order of priority when macro service requests are generated simultaneously,

2. ASI: Asynchronous serial interface
CSI: Clock synchronous serial interface

There are four kinds of macro service, as shown below.

(1) Type A

One byte or one word of data is transferred between a special function register (SFR) and memory each time an interrupt request is generated, and a vectored interrupt request is generated when the specified number of transfers have been performed.

Memory that can be used in the transfers is limited to internal RAM addresses 0FE00H to 0FEFFH when the LOCATION 0 instruction is executed, and addresses 0FFE00H to 0FFEFFH when the LOCATION 0FH instruction is executed.

The specification method is simple and is suitable for low-volume, high-speed data transfers.

(2) Type B

As with type A, one byte or one word of data is transferred between a special function register (SFR) and memory each time an interrupt request is generated, and a vectored interrupt request is generated when the specified number of transfers have been performed.

The SFR and memory to be used in the transfers is specified by the macro service channel (the entire 1M-byte memory space can be used).

This is a general version of type A, suitable for large volumes of transfer data.

(3) Type C

Data is transferred from memory to two special function registers (SFR) each time an interrupt request is generated, and a vectored interrupt request is generated when the specified number of transfers have been performed.

With type C macro service, not only are data transfers performed to two locations in response to a single interrupt request, but it is also possible to add output data ring control and a function that automatically adds data to a compare register. The entire 1-Mbyte memory space can be used.

Type C is mainly used with the INTTM1 and INTTM2 interrupts, and is used for stepping motor control, etc., by macro service, with RTBL or RTBH and CR10, CR1W used as the SFRs to which data is transferred.

(4) Counter mode

This mode is to decrement the macro service counter (MSC) when an interrupt occurs and is used to count the division operation of an interrupt and interrupt generation circuit.

When MSC is 0, a vector interrupt can be generated.

To restart the macro service, MSC must be set again.

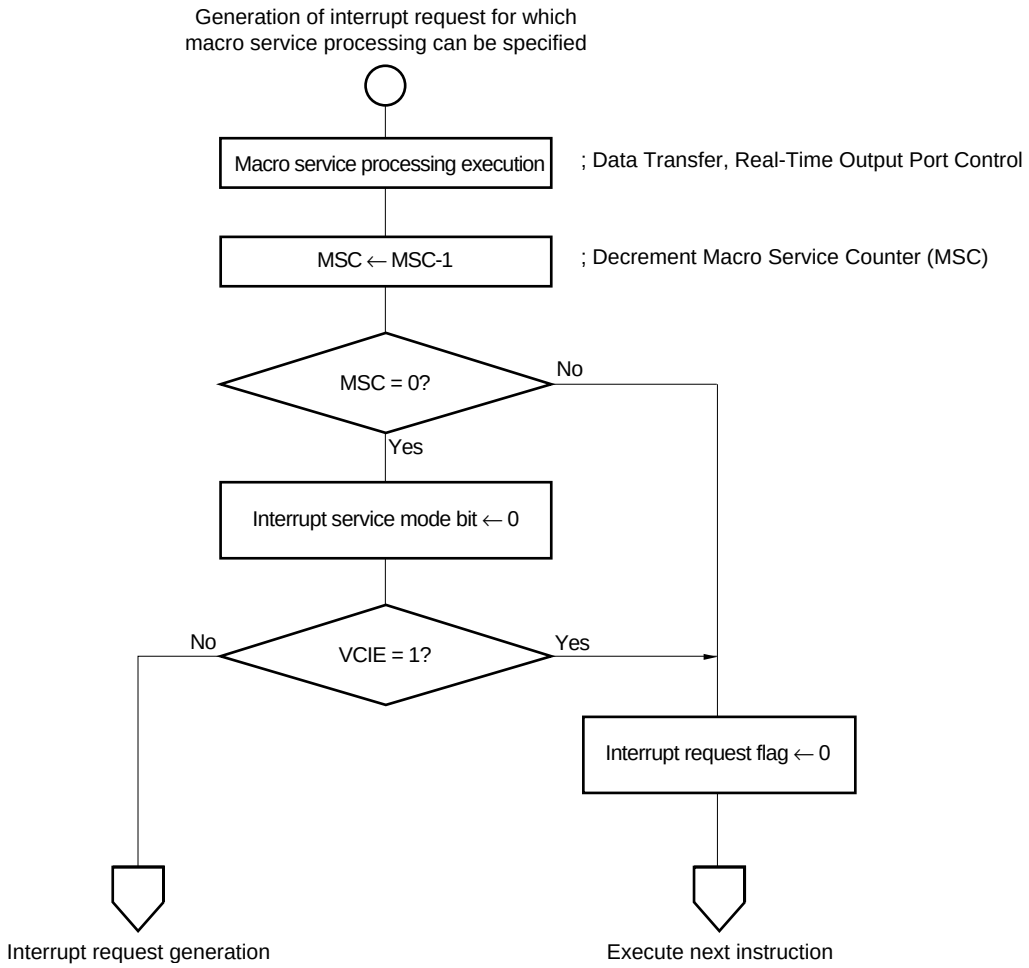
MSC is fixed to 16 bits and cannot be used as an 8-bit counter.

22.8.3 Basic macro service operation

Interrupt requests for which the macro service processing generated by the algorithm shown in Figure 22-11 can be specified are basically serviced in the sequence shown in Figure 22-18.

Interrupt requests for which macro service processing can be specified are not affected by the status of the IE flag, but are disabled by setting (1) an interrupt mask flag in the interrupt mask register (MK0). Macro service processing can be executed in the interrupt disabled state and during execution of an interrupt service program.

Figure 22-18. Macro Service Processing Sequence



The macro service type and transfer direction are determined by the value set in the macro service control word mode register. Transfer processing is then performed using the macro service channel specified by the channel pointer according to the macro service type.

The macro service channel is memory which contains the macro service counter which records the number of transfers, the transfer destination and transfer source pointers, and data buffers, and can be located at any address in the range FE00H to FEFFH when the LOCATION 0 instruction is executed, or FFE00H to FFEFFH when the LOCATION 0FH instruction is executed.

22.8.4 Operation at end of macro service

In macro service, processing is performed the number of times specified during execution of another program. Macro service ends when the processing has been performed the specified number of times (when the macro service counter (MSC) reaches 0). Either of two operations may be performed at this point, as specified by the VCIE bit (bit 7) of the macro service mode register for each macro service.

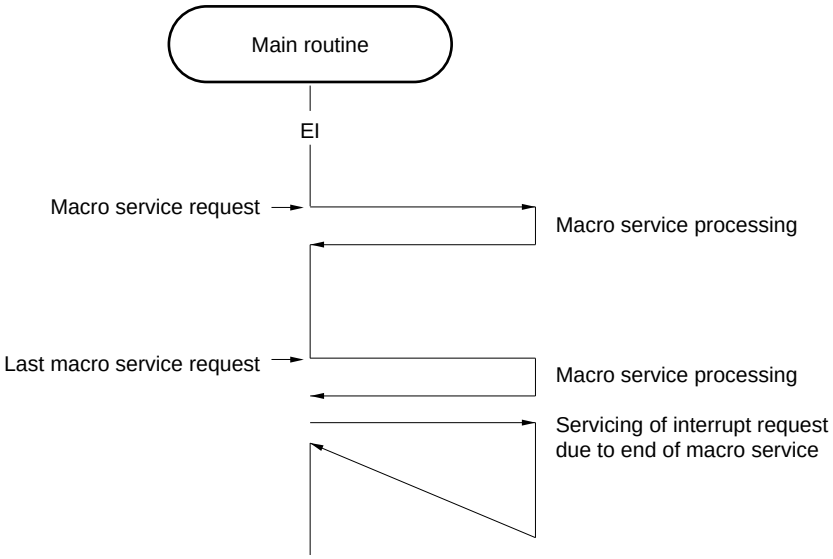
(1) When VCIE bit is 0

In this mode, an interrupt is generated as soon as the macro service ends. Figure 22-18 shows an example of macro service and interrupt acknowledgment operations when the VCIE bit is 0.

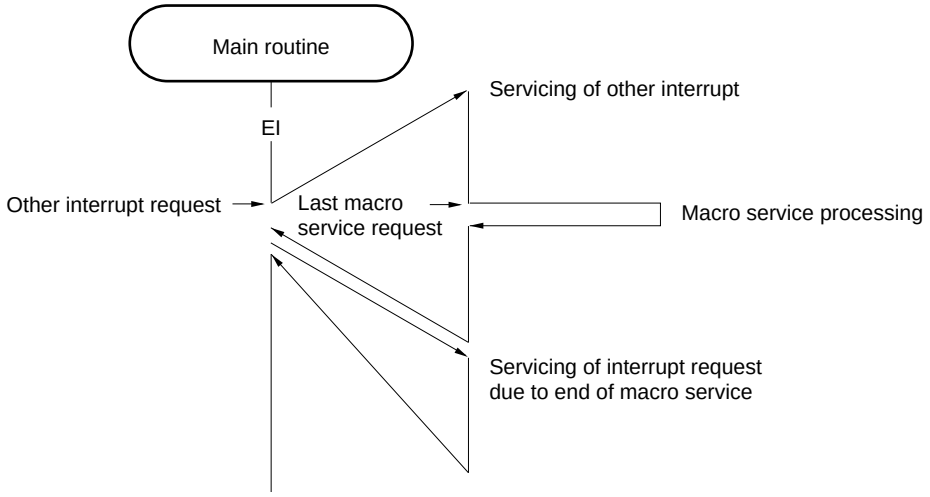
This mode is used when a series of operations end with the last macro service processing performed, for instance. It is mainly used in the following cases:

- Asynchronous serial interface receive data buffering (INTSR1/INTSR2)
- A/D conversion result fetch (INTAD)
- Compare register update as the result of a match between a timer register and the compare register (INTTM00, INTTM01, INTTM1, INTTM2, INTTM5, and INTTM6)

Figure 22-19. Operation at End of Macro Service When VCIE = 0



At the end of macro service (MSC = 0), an interrupt request is generated and acknowledged.



If the last macro service is performed when the interrupt due to the end of macro service cannot be acknowledged while other interrupt servicing is being executed, etc., that interrupt is held pending until it can be acknowledged.

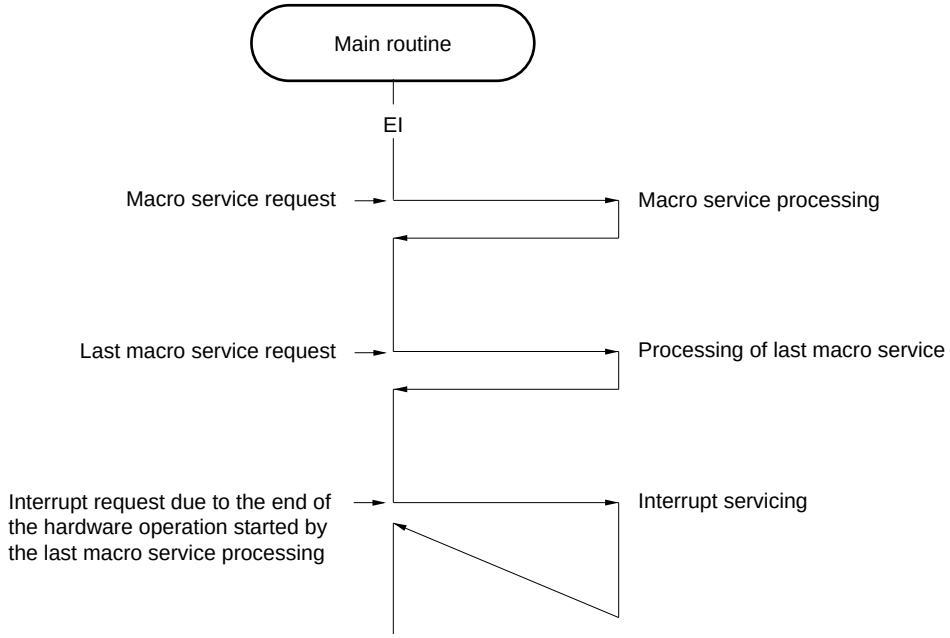
(2) When VCIE bit is 1

In this mode, an interrupt is not generated after macro service ends. Figure 22-20 shows an example of macro service and interrupt acknowledgment operations when the VCIE bit is 1.

This mode is used when the final operation is to be started by the last macro service processing performed, for instance. It is mainly used in the following cases:

- Clocked serial interface receive data transfers (INTCSI0, INTCSI1/INTCSI2)
- Asynchronous serial interface data transfers (INTST1, INTST2)
- To stop a stepping motor in the case (INTTM1, INTTM2) of stepping motor control by means of macro service type C using the real-time output port and timer/counter.

Figure 22-20. Operation at End of Macro Service When VCIE = 1



22.8.5 Macro service control registers

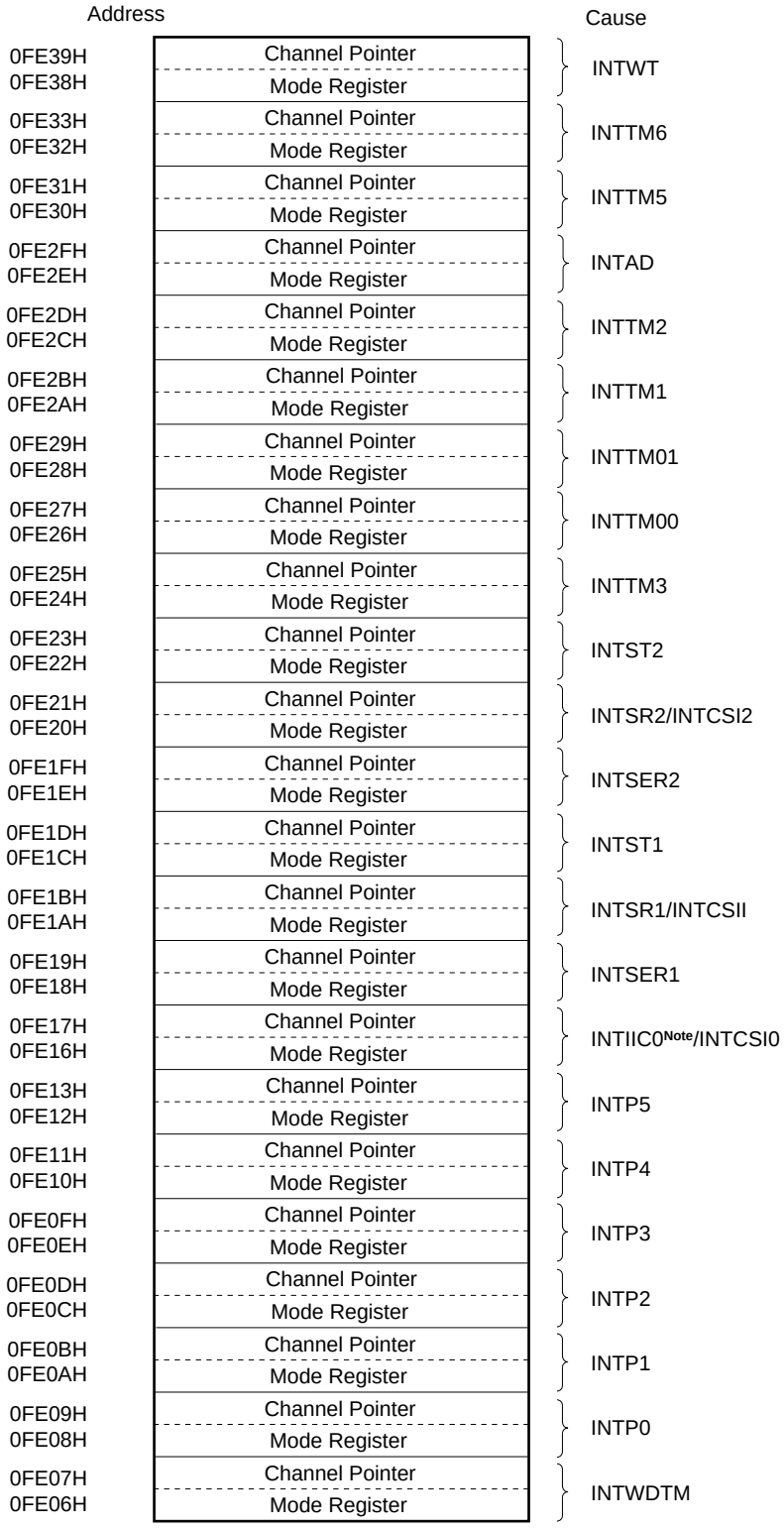
(1) Macro service control word

The μPD784225's macro service function is controlled by the macro service control mode register and macro service channel pointer. The macro service processing mode is set by means of the macro service mode register, and the macro service channel address is indicated by the macro service channel pointer.

The macro service mode register and macro service channel pointer are mapped onto the part of the internal RAM shown in Figure 22-21 for each macro service as the macro service control word.

When macro service processing is performed, the macro service mode register and channel pointer values corresponding to the interrupt requests for which macro service processing can be specified must be set beforehand.

Figure 22-21. Macro Service Control Word Format



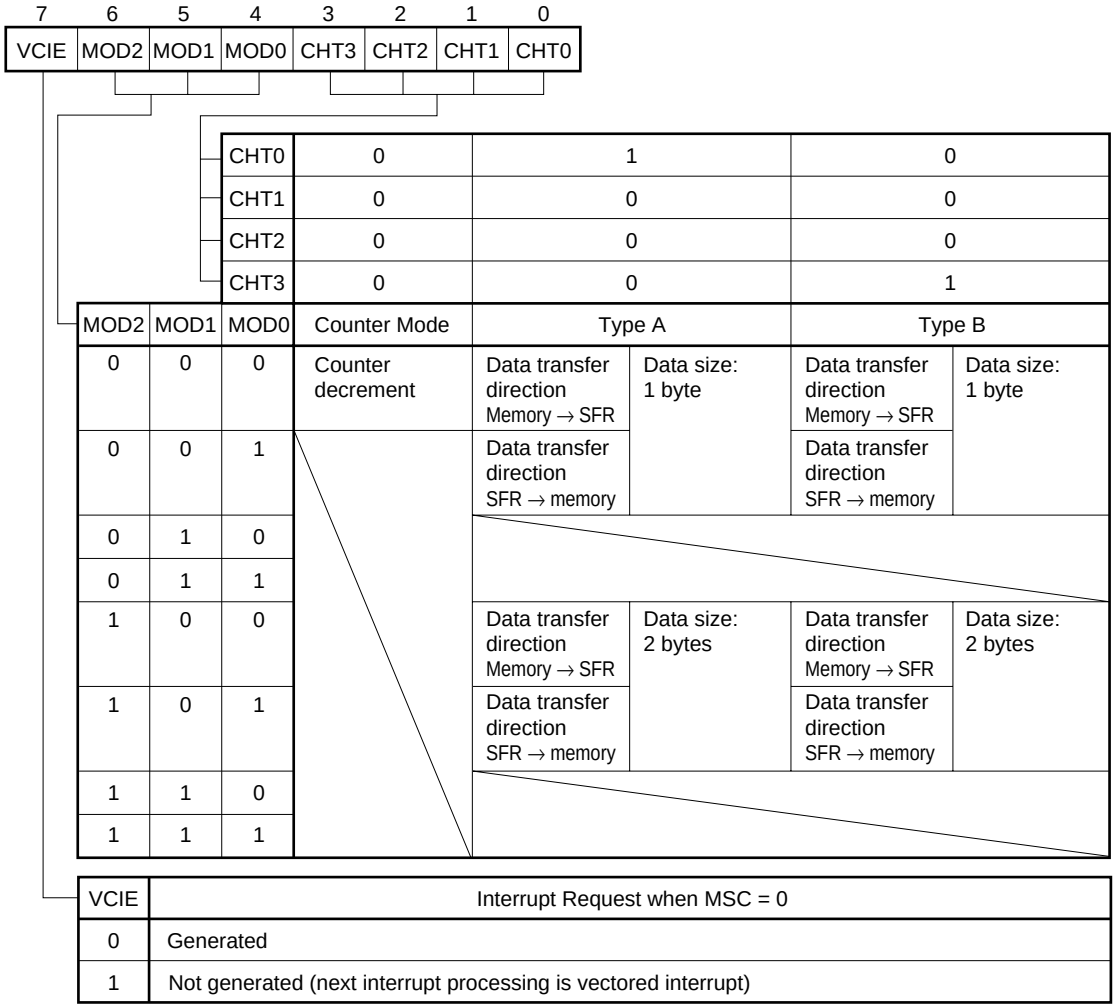
Note μPD784225Y Subseries only

(2) Macro service mode register

The macro service mode register is an 8-bit register that specifies the macro service operation. This register is written in internal RAM as part of the macro service control word (refer to **Figure 22-21**).

The format of the macro service mode register is shown in Figure 22-22.

Figure 22-22. Macro Service Mode Register Format (1/2)

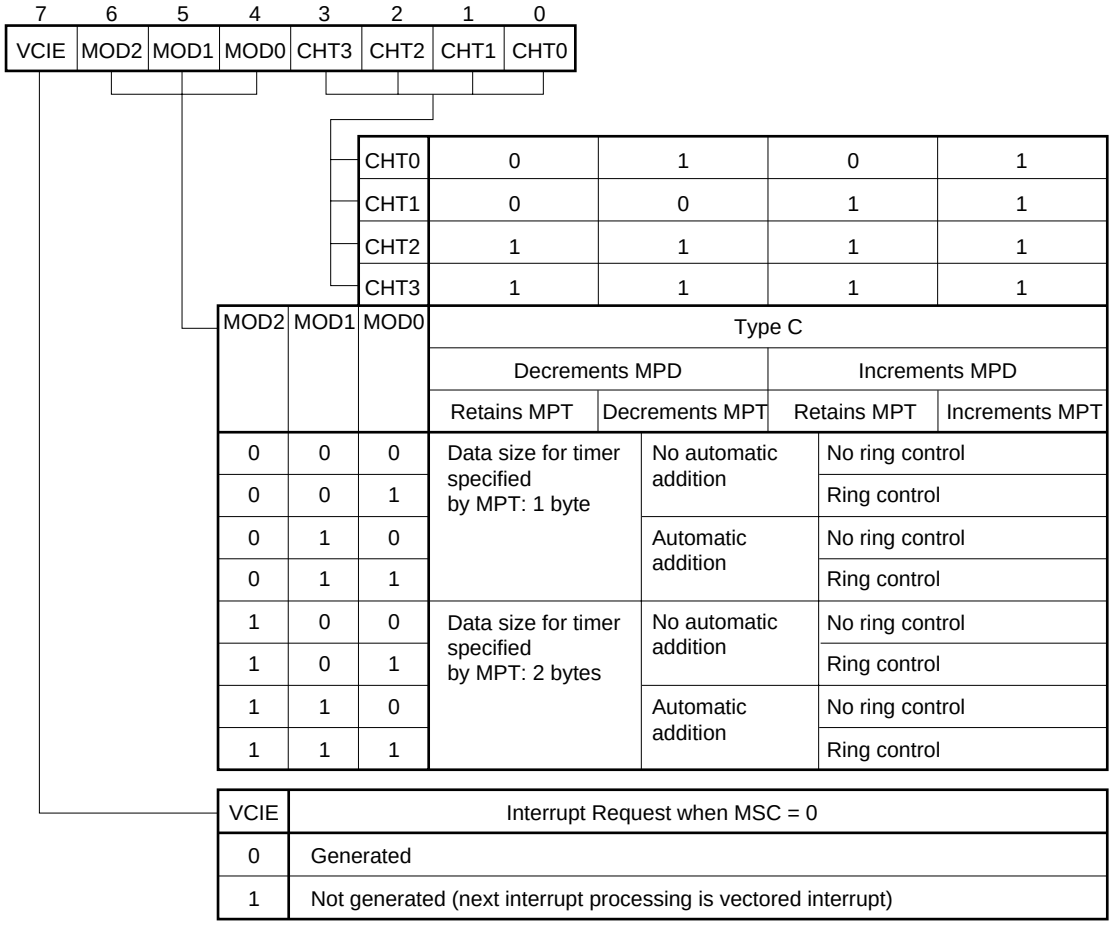


CHT0	0	1	0				
CHT1	0	0	0				
CHT2	0	0	0				
CHT3	0	0	1				

MOD2	MOD1	MOD0	Counter Mode	Type A		Type B	
0	0	0	Counter decrement	Data transfer direction Memory → SFR	Data size: 1 byte	Data transfer direction Memory → SFR	Data size: 1 byte
0	0	1		Data transfer direction SFR → memory		Data transfer direction SFR → memory	
0	1	0					
0	1	1					
1	0	0		Data transfer direction Memory → SFR	Data size: 2 bytes	Data transfer direction Memory → SFR	Data size: 2 bytes
1	0	1		Data transfer direction SFR → memory		Data transfer direction SFR → memory	
1	1	0					
1	1	1					

VCIE	Interrupt Request when MSC = 0
0	Generated
1	Not generated (next interrupt processing is vectored interrupt)

Figure 22-22. Macro Service Mode Register Format (2/2)



7	6	5	4	3	2	1	0
VCIE	MOD2	MOD1	MOD0	CHT3	CHT2	CHT1	CHT0

CHT0	0	1	0	1
CHT1	0	0	1	1
CHT2	1	1	1	1
CHT3	1	1	1	1

MOD2	MOD1	MOD0	Type C			
			Decrements MPD		Increments MPD	
			Retains MPT	Decrements MPT	Retains MPT	Increments MPT
0	0	0	Data size for timer specified by MPT: 1 byte	No automatic addition	No ring control	
0	0	1			Ring control	
0	1	0		Automatic addition	No ring control	
0	1	1			Ring control	
1	0	0	Data size for timer specified by MPT: 2 bytes	No automatic addition	No ring control	
1	0	1			Ring control	
1	1	0		Automatic addition	No ring control	
1	1	1			Ring control	

VCIE	Interrupt Request when MSC = 0
0	Generated
1	Not generated (next interrupt processing is vectored interrupt)

(3) Macro service channel pointer

The macro service channel pointer specifies the macro service channel address. The macro service channel can be located in the 256-byte space from FE00H to FEFFH when the LOCATION 0 instruction is executed, or FFE00H to FFEFFH when the LOCATION 0FH instruction is executed, and the high-order 16 bits of the address are fixed. Therefore, the low-order 8 bits of the data stored to the highest address of the macro service channel are set in the macro service channel pointer.

22.8.6 Macro service type A

(1) Operation

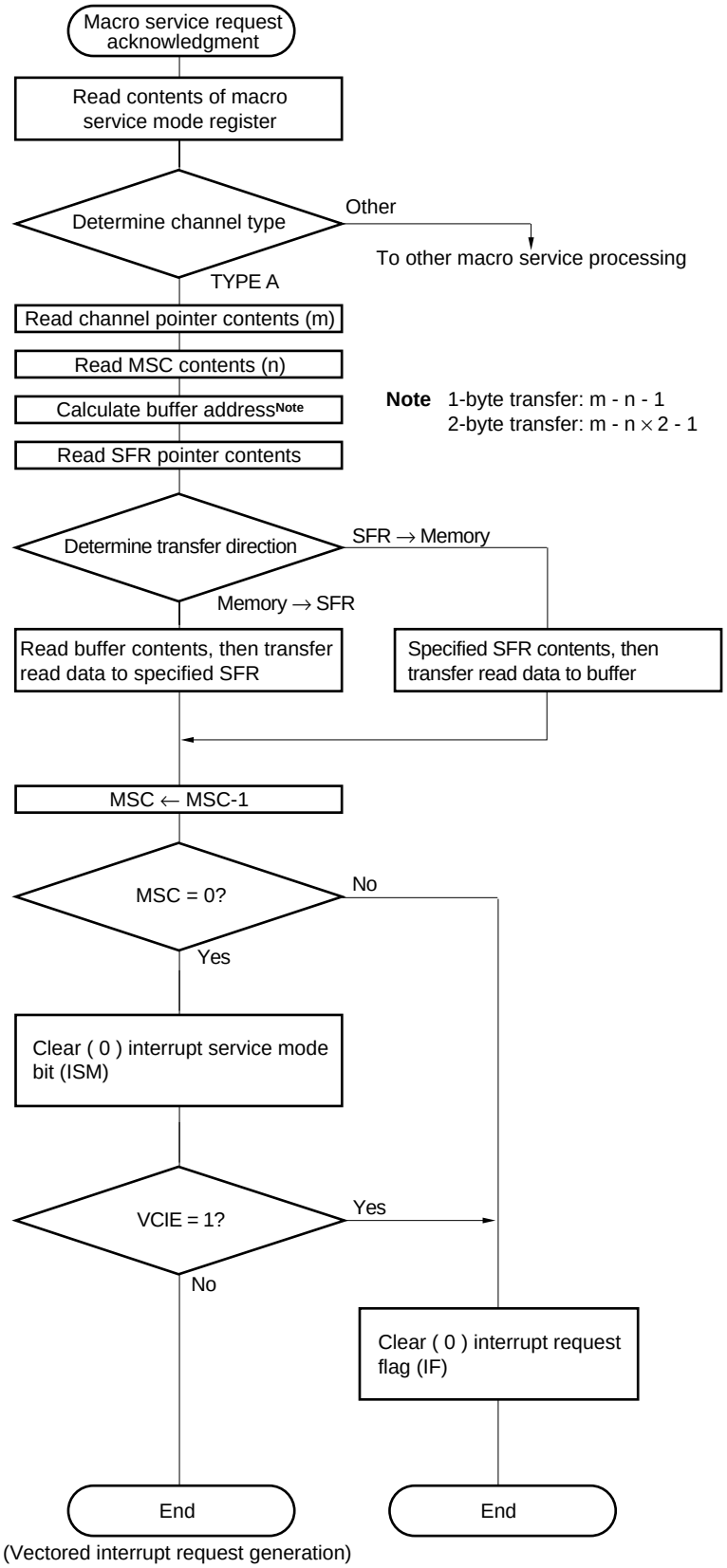
Data transfers are performed between buffer memory in the macro service channel and an SFR specified in the macro service channel.

With type A, the data transfer direction can be selected as memory-to-SFR or SFR-to-memory.

Data transfers are performed the number of times set beforehand in the macro service counter. One macro service processing transfers 8-bit or 16-bit data.

Type A macro service is useful when the amount of data to be transferred is small, as transfers can be performed at high speed.

Figure 22-23. Macro Service Data Transfer Processing Flow (Type A)

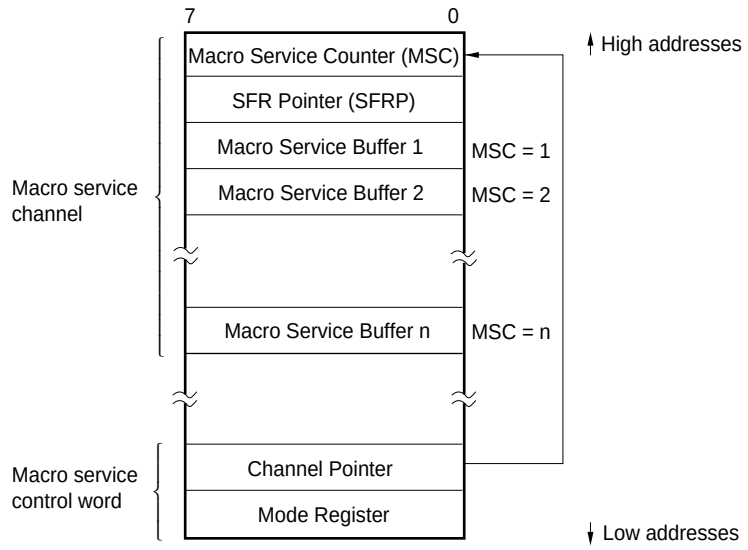


(2) Macro service channel configuration

The channel pointer and 8-bit macro service counter (MSC) indicate the buffer address in internal RAM (FE00H to FEFFH when the LOCATION 0 instruction is executed, or FFE00H to FFEFFH when the LOCATION 0FH instruction is executed) which is the transfer source or transfer destination (refer to **Figure 22-24**). In the channel pointer, the low-order 8 bits of the address are written to the macro service counter in the macro service channel. The SFR involved with the access is specified by the SFR pointer (SFRP). The low-order 8 bits of the SFR address are written to the SFRP.

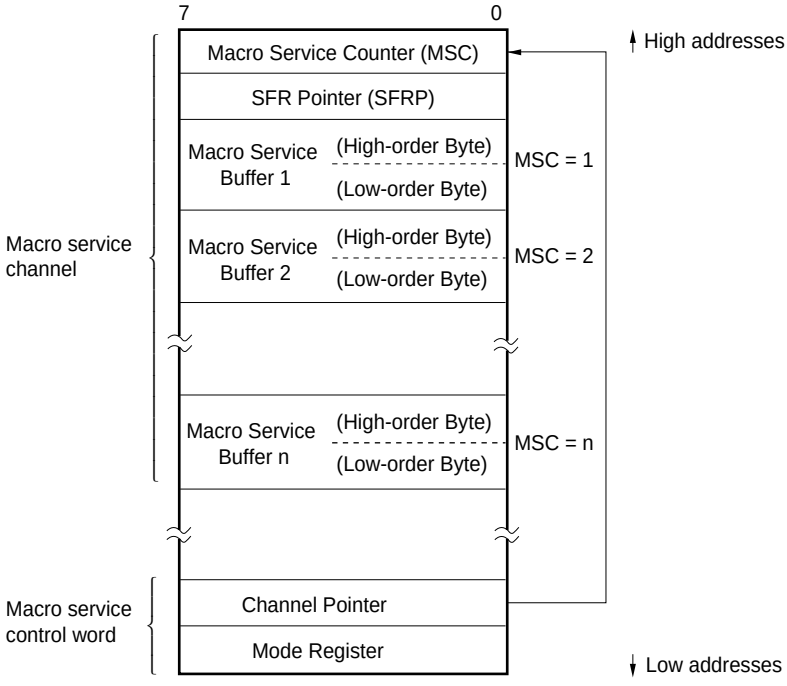
Figure 22-24. Type A Macro Service Channel

(a) 1-byte transfers



Macro service buffer address = (channel pointer) - (macro service counter) - 1

(b) 2-byte transfers

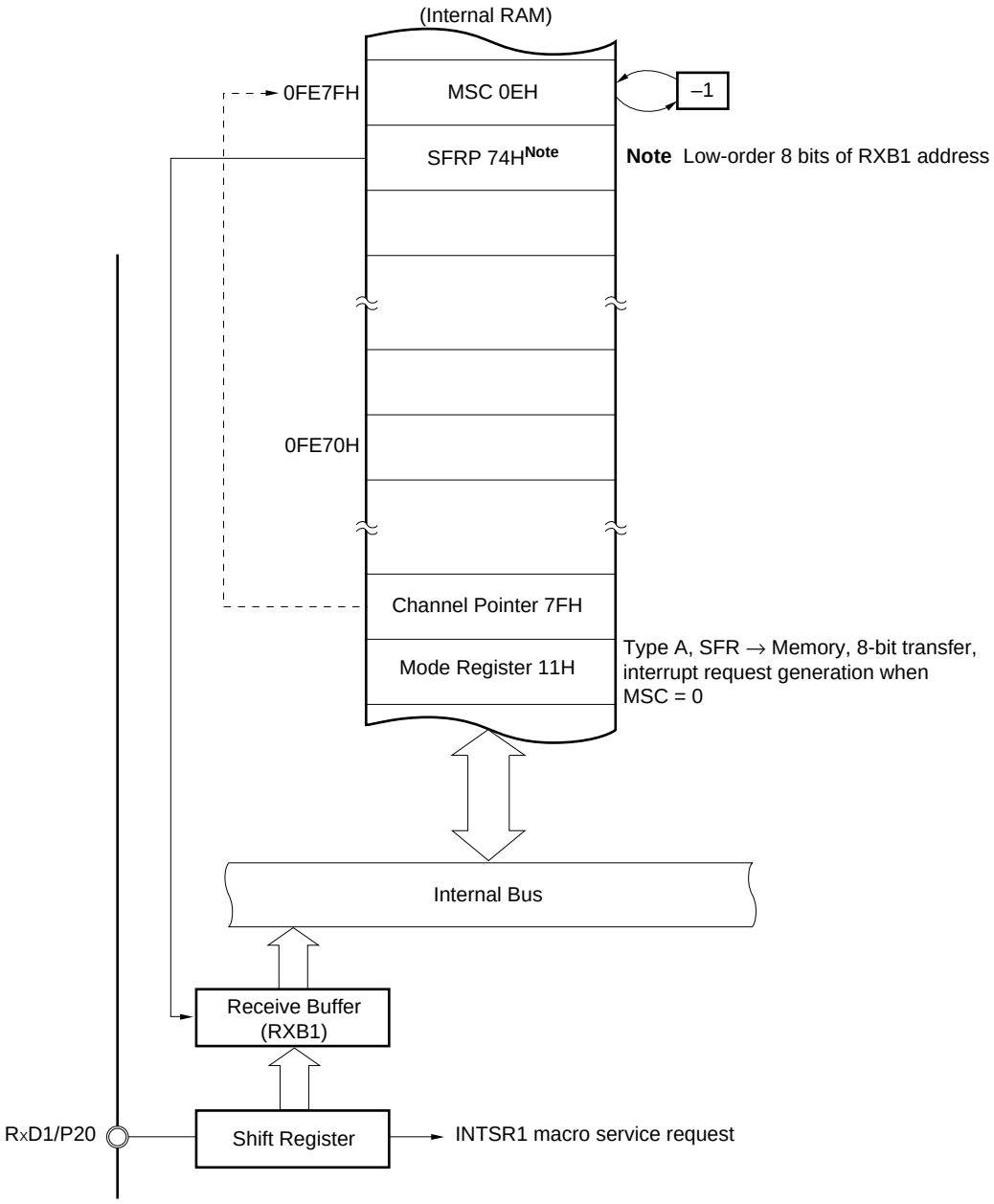


Macro service buffer address = (channel pointer) - (macro service counter) × 2 - 1

(3) Example of use of type A

An example is shown below in which data received via the asynchronous serial interface is transferred to a buffer area in on-chip RAM.

Figure 22-25. Asynchronous Serial Reception



Remark Addresses in the figure are the values when the LOCATION 0 instruction is executed.
When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values in the figure.

22.8.7 Macro service type B

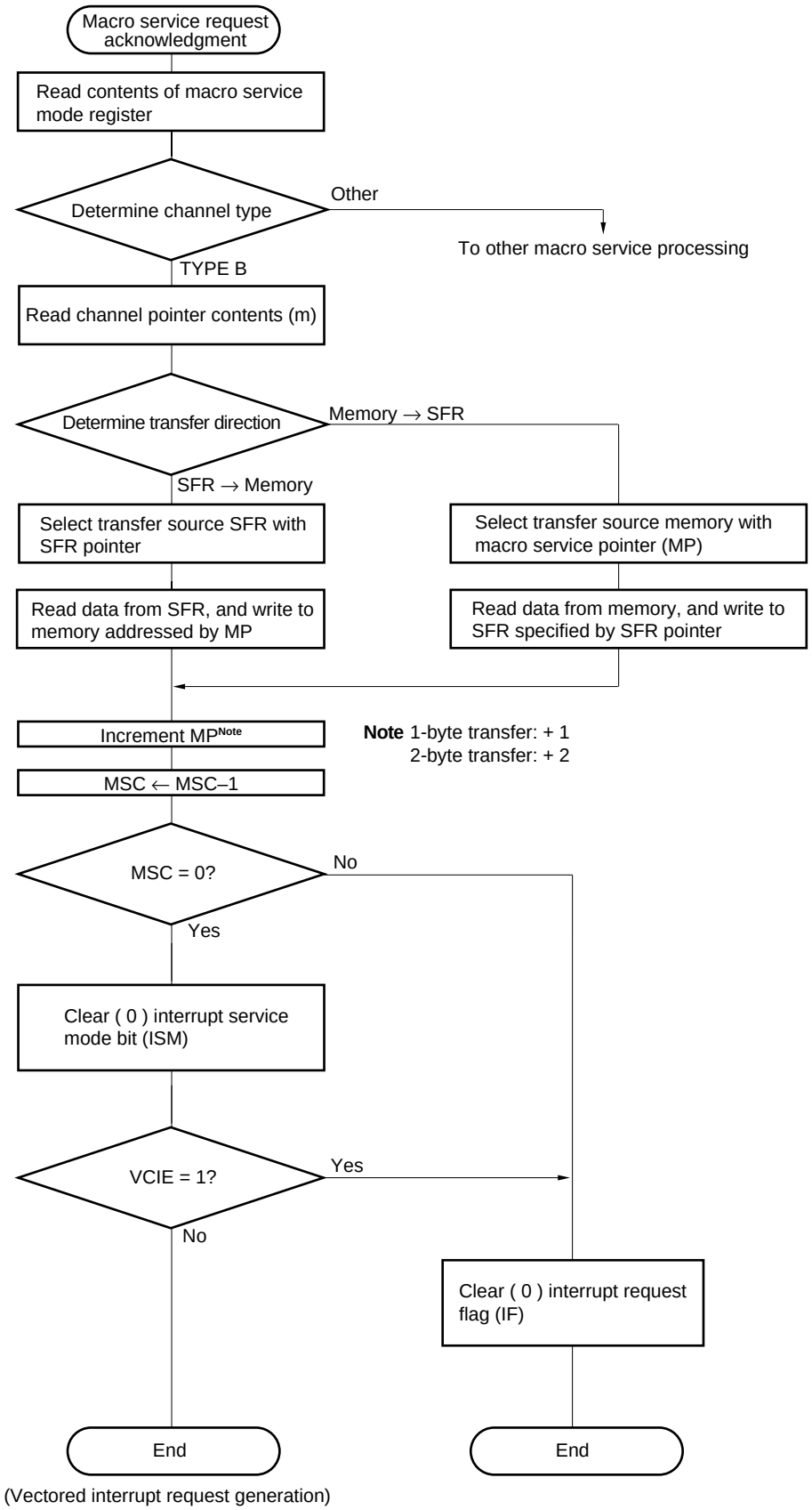
(1) Operation

Data transfers are performed between a data area in memory and an SFR specified by the macro service channel. With type B, the data transfer direction can be selected as memory-to-SFR or SFR-to-memory.

Data transfers are performed the number of times set beforehand in the macro service counter. One macro service processing transfers 8-bit or 16-bit data.

This type of macro service is macro service type A for general purposes and is ideal for processing a large amount of data because up to 64 Kbytes of data buffer area when 8-bit data is transferred or 1 Mbyte of data buffer area when 16-bit data is transferred can be set in any address space.

Figure 22-26. Macro Service Data Transfer Processing Flow (Type B)



(2) Macro service channel configuration

The macro service pointer (MP) indicates the data buffer area in the 1-Mbyte memory space that is the transfer destination or transfer source.

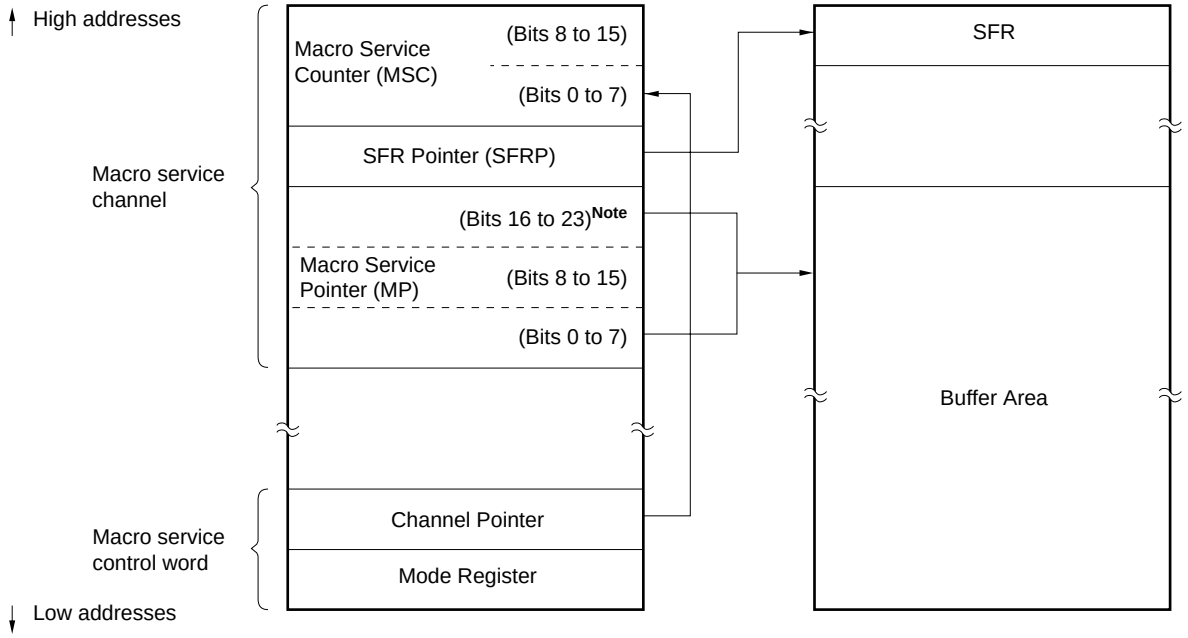
The low-order 8 bits of the SFR that is the transfer destination or transfer source is written to the SFR pointer (SFRP).

The macro service counter (MSC) is a 16-bit counter that specifies the number of data transfers.

The macro service channel that stores the MP, SFRP and MSC is located in internal RAM space addresses 0FE00H to 0FEFFH when the LOCATION 0 instruction is executed, or 0FFE00H to 0FFEFFH when the LOCATION 0FH instruction is executed.

The macro service channel is indicated by the channel pointer as shown in Figure 22-27. In the channel pointer, the low-order 8 bits of the address are written to the macro service counter in the macro service channel.

Figure 22-27. Type B Macro Service Channel



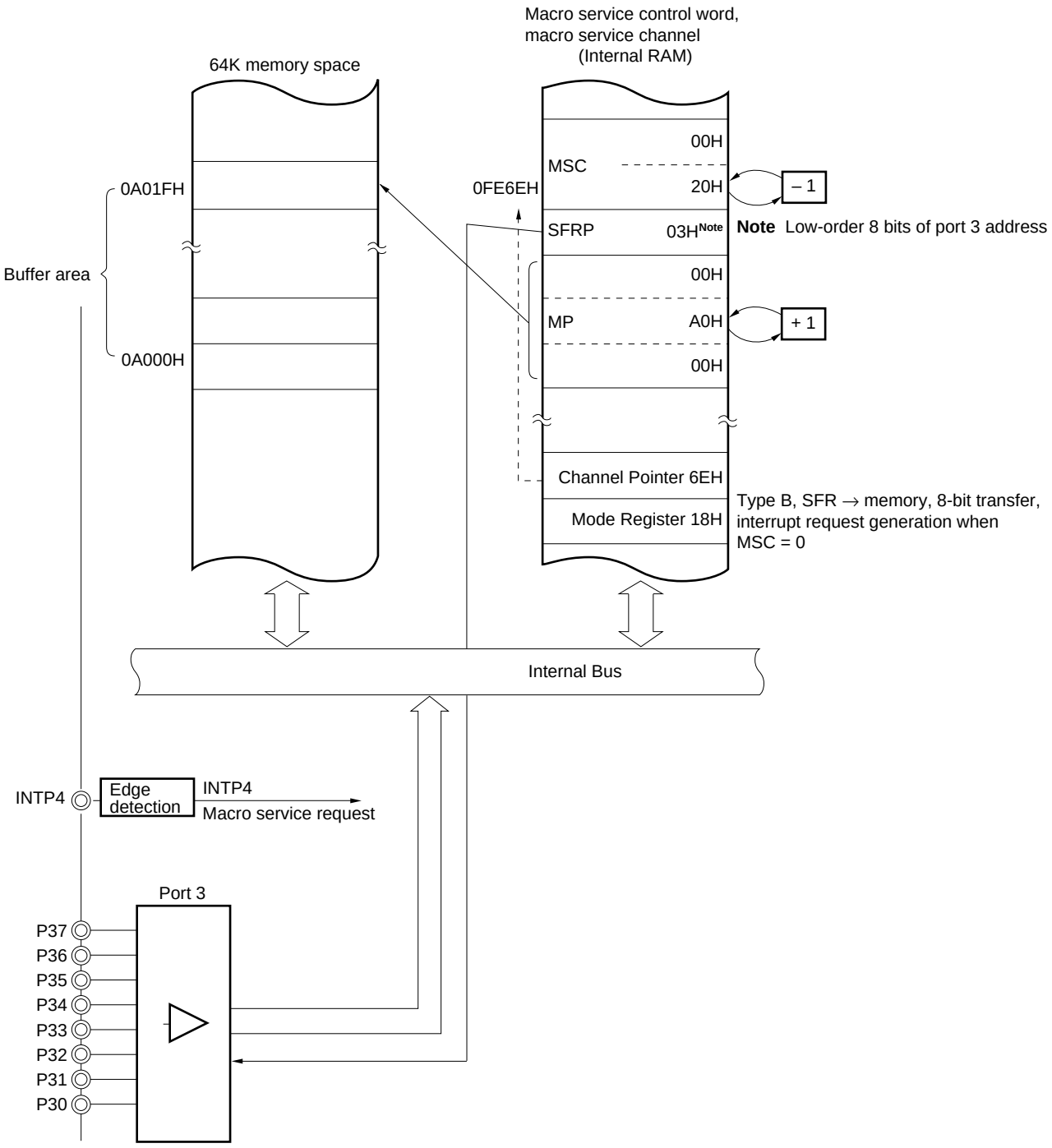
Macro service buffer address = macro service pointer

Note Bits 20 to 23 must be set to 0.

(3) Example of use of type B

An example is shown below in which parallel data is input from port 3 in synchronization with an external signal. The INTP4 external interrupt pin is used for synchronization with the external signal.

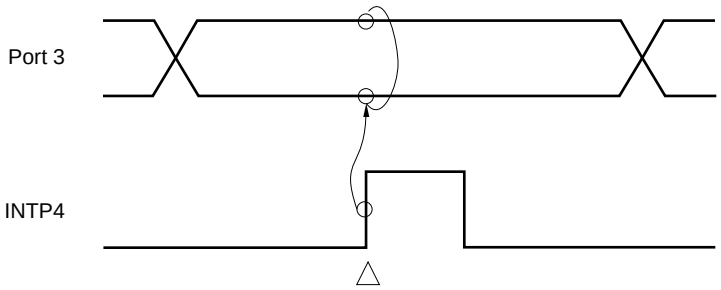
Figure 22-28. Parallel Data Input Synchronized with External Interrupts



Remark Macro service channel addresses in the figure are the values when the LOCATION 0 instruction is executed.

When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values in the figure.

Figure 22-29. Parallel Data Input Timing



22.8.8 Macro service type C

(1) Operation

In type C macro service, data in the memory specified by the macro service channel is transferred to two SFRs, for timer use and data use, specified by the macro service channel in response to a single interrupt request (the SFRs can be freely selected). An 8-bit or 16-bit timer SFR can be selected.

In addition to the basic data transfers described above, type C macro service, the following functions can be added to type C macro service to reduce the size of the buffer area and alleviate the burden on software.

These specifications are made by using the mode register of the macro service control word.

(a) Updating of timer macro service pointer

It is possible to choose whether the timer macro service pointer (MPT) is to be kept as it is or incremented/decremented. The MPT is incremented or decremented in the same direction as the macro service pointer (MPD) for data.

(b) Updating of data macro service pointer

It is possible to choose whether the data macro service pointer (MPD) is to be incremented or decremented.

(c) Automatic addition

The current compare register value is added to the data addressed by the timer macro service pointer (MPT), and the result is transferred to the compare register. If automatic addition is not specified, the data addressed by the MPT is simply transferred to the compare register.

(d) Ring control

An output data pattern of the length specified beforehand is automatically output repeatedly.

Figure 22-30. Macro Service Data Transfer Processing Flow (Type C) (1/2)

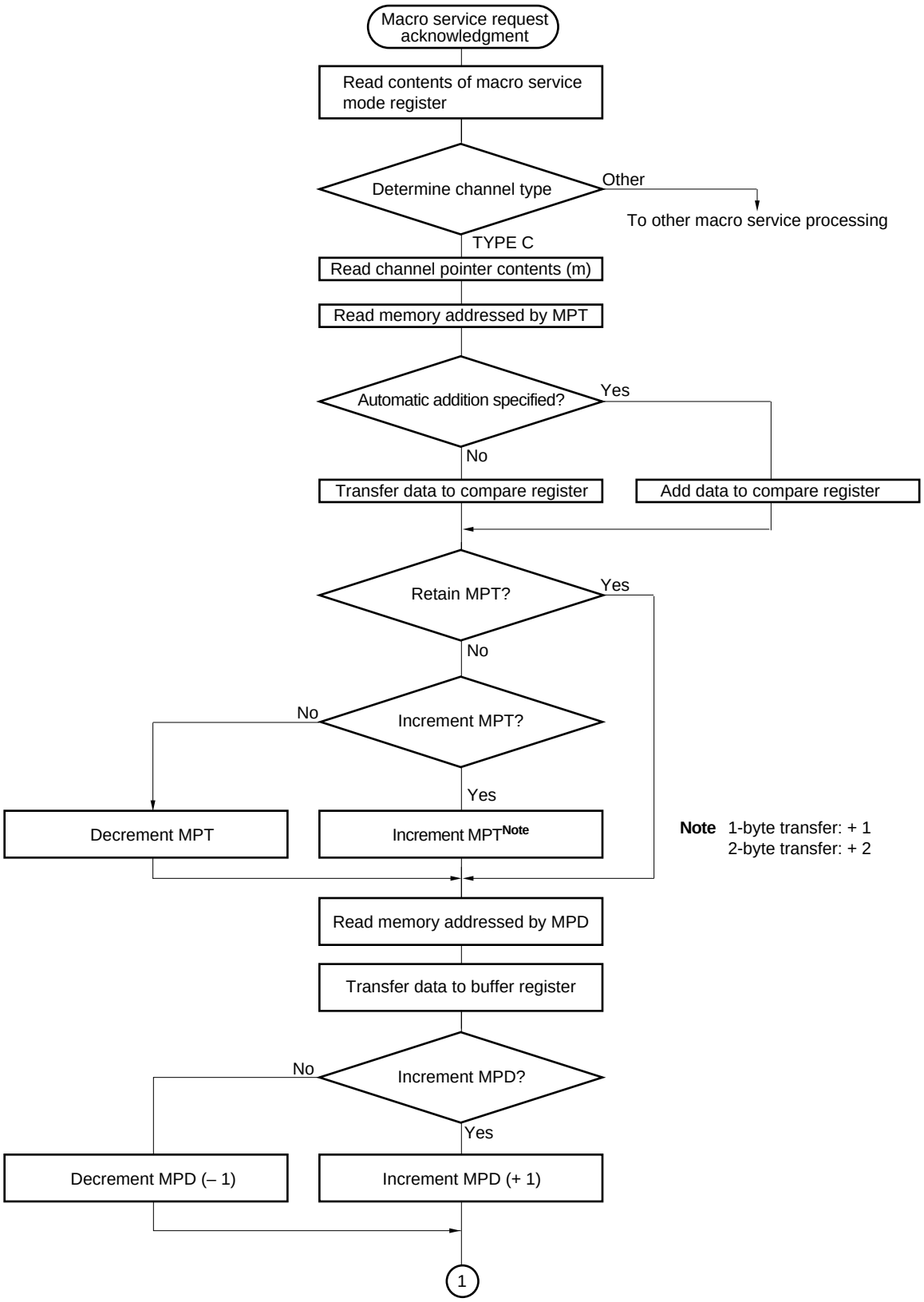
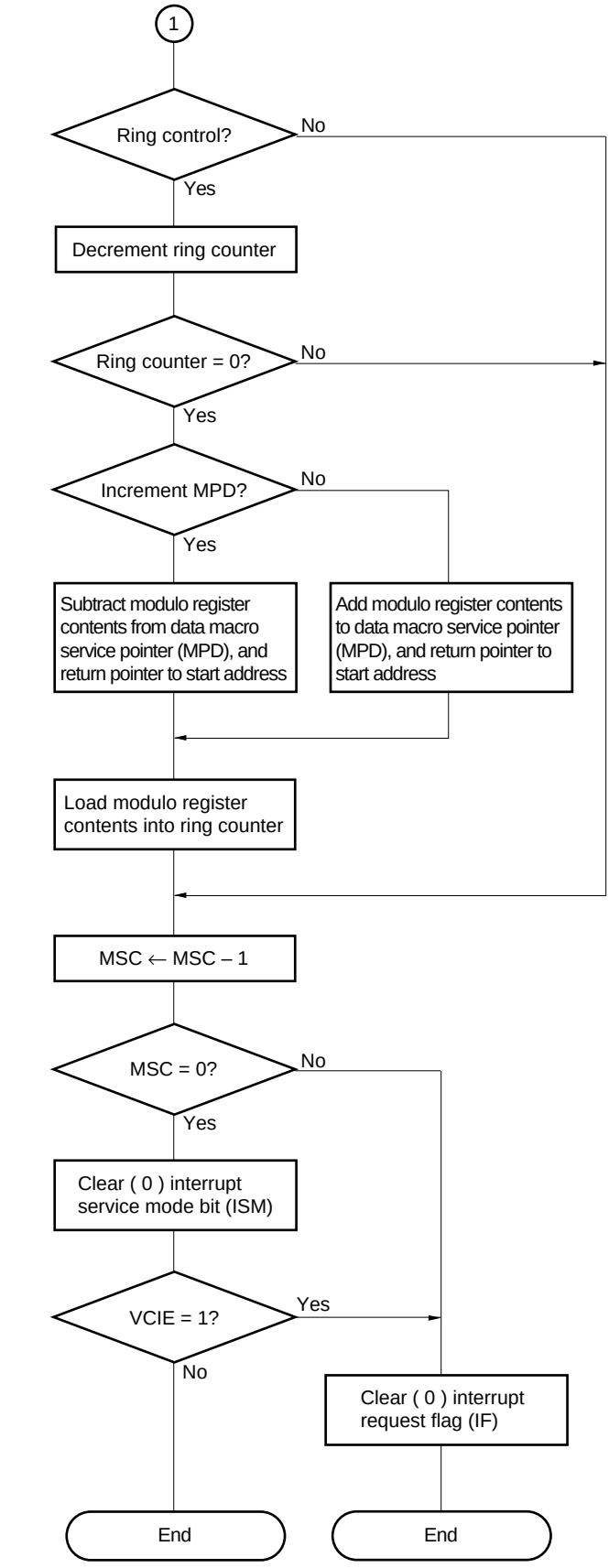


Figure 22-30. Macro Service Data Transfer Processing Flow (Type C) (2/2)



(Vectored interrupt request generation)

(2) Macro service channel configuration

There are two kinds of type C macro service channel, as shown in Figure 22-31.

The timer macro service pointer (MPT) mainly indicates the data buffer area in the 1-Mbyte memory space to be transferred or added to the timer/counter compare register.

The data macro service pointer (MPD) indicates the data buffer area in the 1-Mbyte memory space to be transferred to the real-time output port.

The modulo register (MR) specifies the number of repeat patterns when ring control is used.

The ring counter (RC) holds the step in the pattern when ring control is used. When initialization is performed, the same value as in the MR is normally set in this counter.

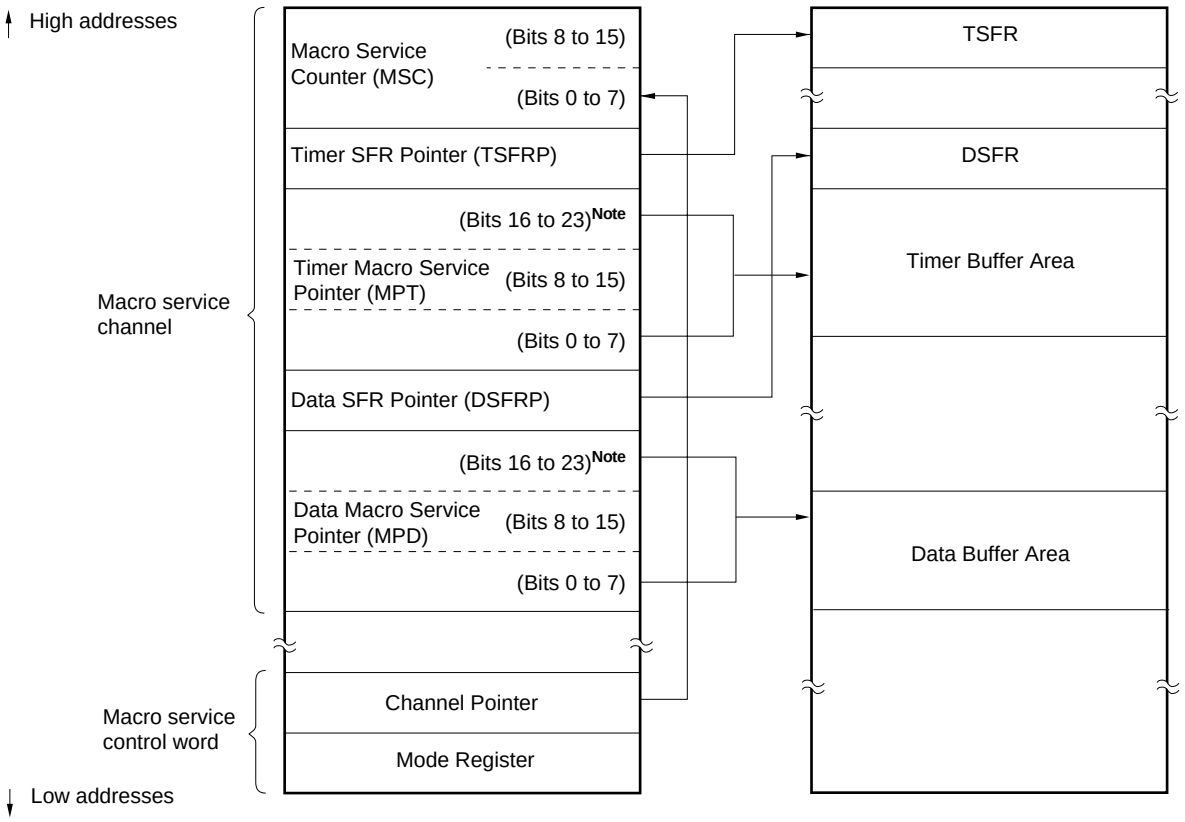
The macro service counter (MSC) is a 16-bit counter that specifies the number of data transfers.

The low-order 8 bits of the SFR that is the transfer destination is written to the timer SFR pointer (TSFRP) and data SFR pointer (DSFRP).

The macro service channel that stores these pointers and counters is located in internal RAM space addresses 0FE00H to 0FEFFH when the LOCATION 0 instruction is executed, or 0FFE00H to 0FFEFFH when the LOCATION 0FH instruction is executed. The macro service channel is indicated by the channel pointer as shown in Figure 22-31. In the channel pointer, the low-order 8 bits of the address are written to the macro service counter in the macro service channel.

Figure 22-31. Type C Macro Service Channel (1/2)

(a) No ring control

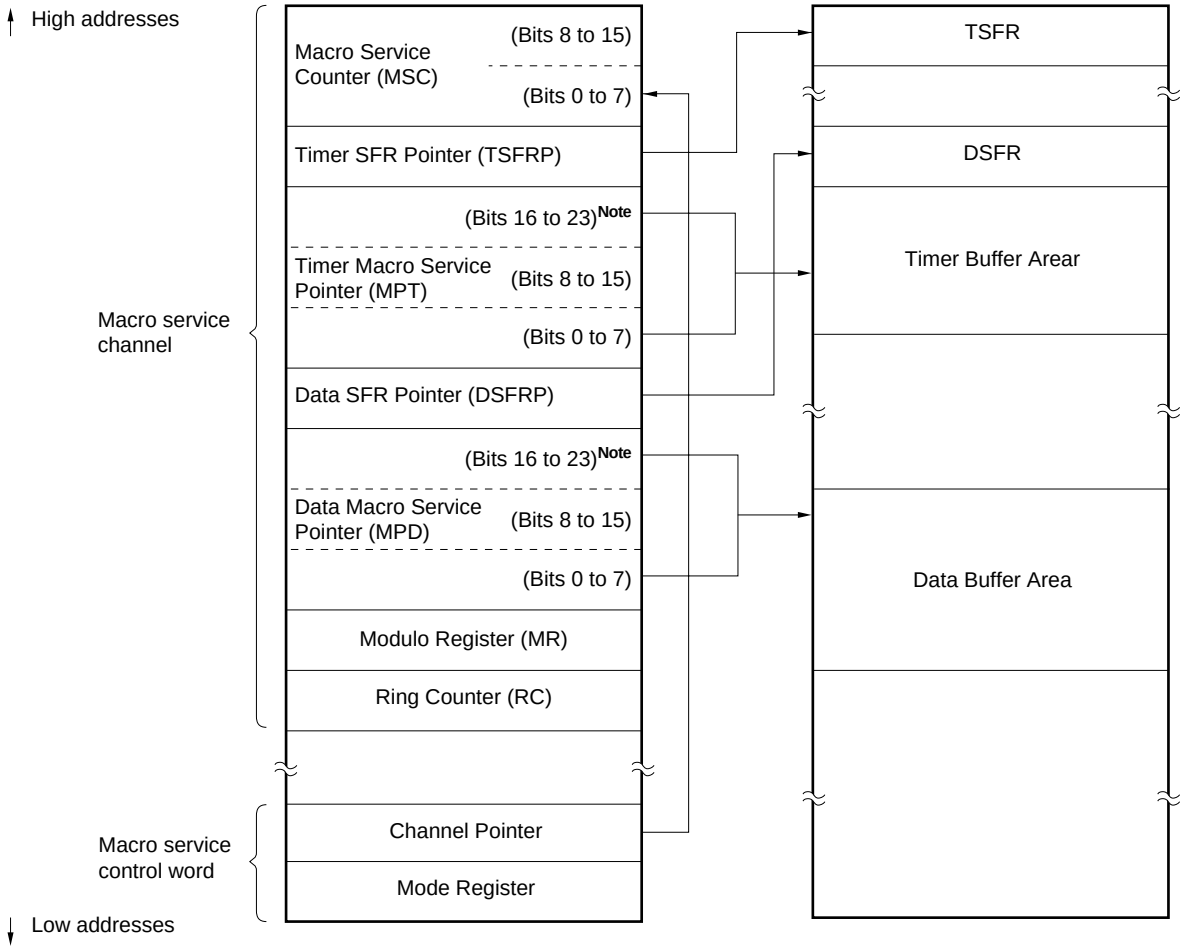


Macro service buffer address = macro service pointer

Note Bits 20 to 23 must be set to 0.

Figure 22-31. Type C Macro Service Channel (2/2)

(b) With ring control



Macro service buffer address = macro service pointer

Note Bits 20 to 23 must be set to 0.

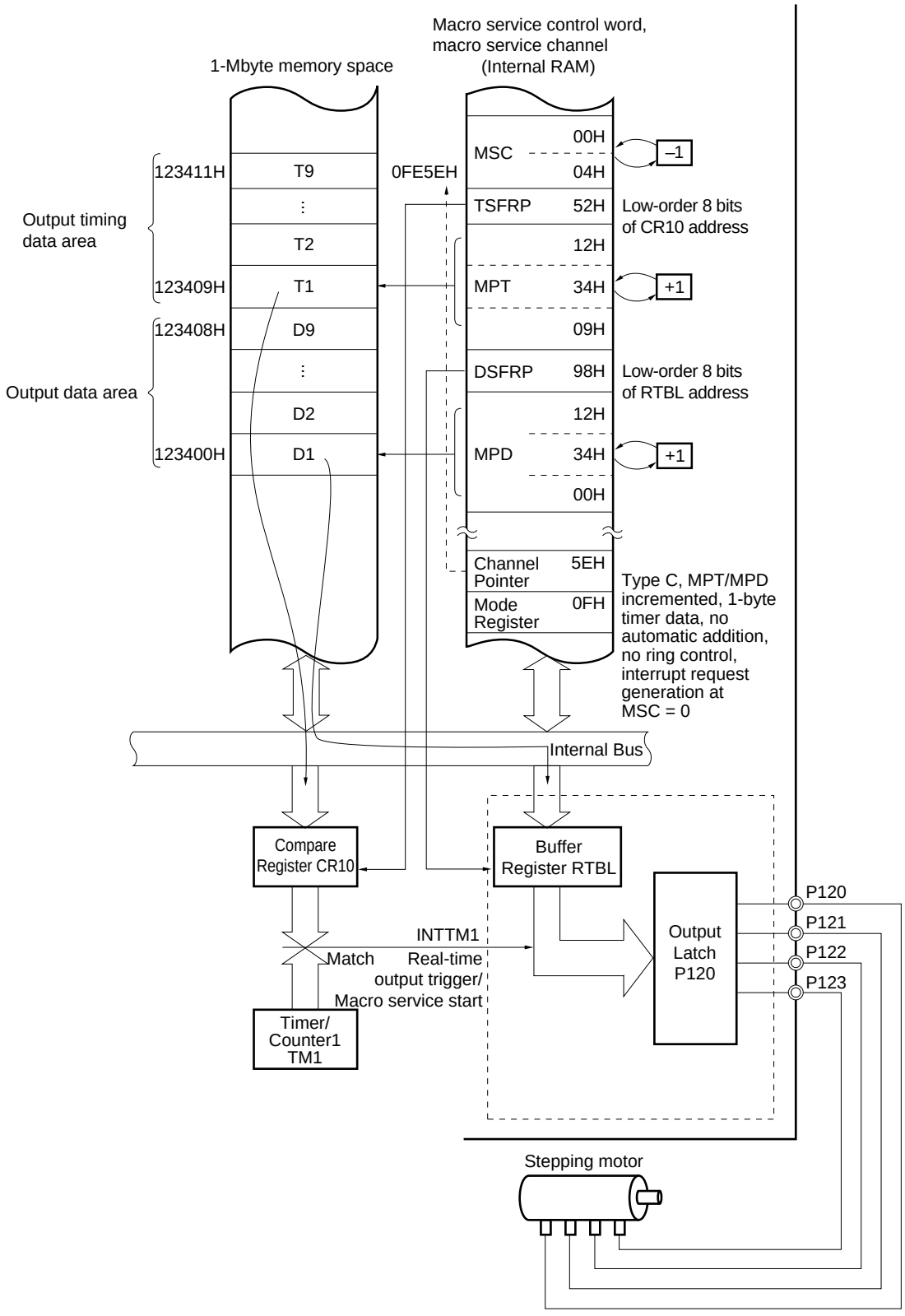
(3) Examples of use of type C

(a) Basic operation

An example is shown below in which the output pattern to the real-time output port and the output interval are directly controlled.

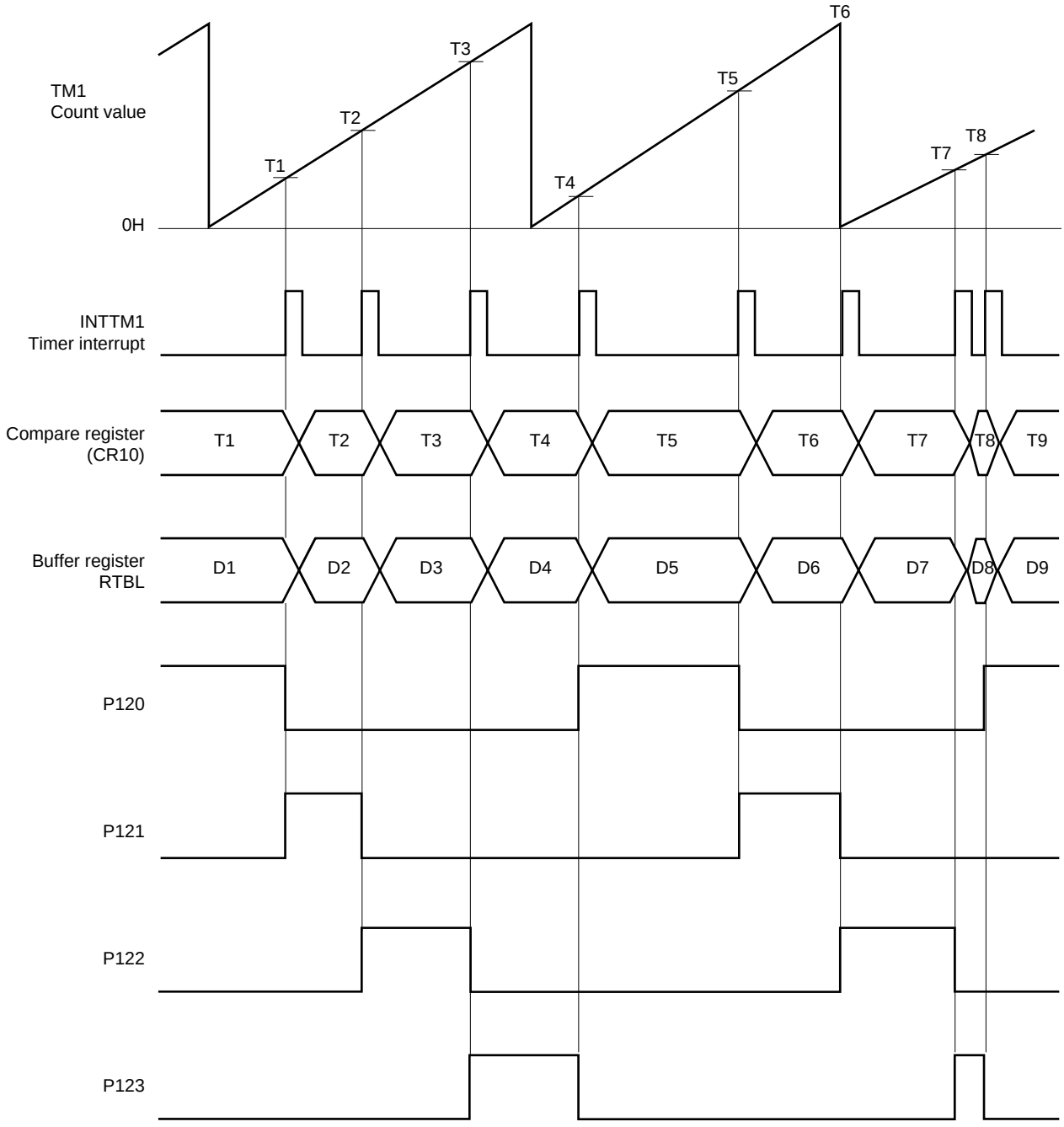
Update data is transferred from the two data storage areas set in the 1-Mbyte space beforehand to the real-time output function buffer register (RTBL) and the compare register (CR10).

Figure 22-32. Stepping Motor Open Loop Control by Real-Time Output Port



Remark Internal RAM addresses in the figure are the values when the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values in the figure.

Figure 22-33. Data Transfer Control Timing



(b) Examples of use of automatic addition control and ring control

(i) Automatic addition control

The output timing data (Đt) specified by the macro service pointer (MPT) is added to the contents of the compare register, and the result is written back to the compare register.

Use of this automatic addition control eliminates the need to calculate the compare register setting value in the program each time.

(ii) Ring control

With ring control, the predetermined output patterns is prepared for one cycle only, and the one-cycle data patterns are output repeatedly in order in ring form.

When ring control is used, only the output patterns for one cycle need be prepared, allowing the size of the data ROM area to be reduced.

The macro service counter (MSC) is decremented each time a data transfer is performed.

With ring control, too, an interrupt request is generated when MSC = 0.

When controlling a stepping motor, for example, the output patterns will vary depending on the configuration of the stepping motor concerned, and the phase excitation method (single-phase excitation, two-phase excitation, etc.), but repeat patterns are used in all cases. Examples of single-phase excitation and 1-2-phase excitation of a 4-phase stepping motor are shown in Figures 22-34 and 22-35.

Figure 22-34. Single-Phase Excitation of 4-Phase Stepping Motor

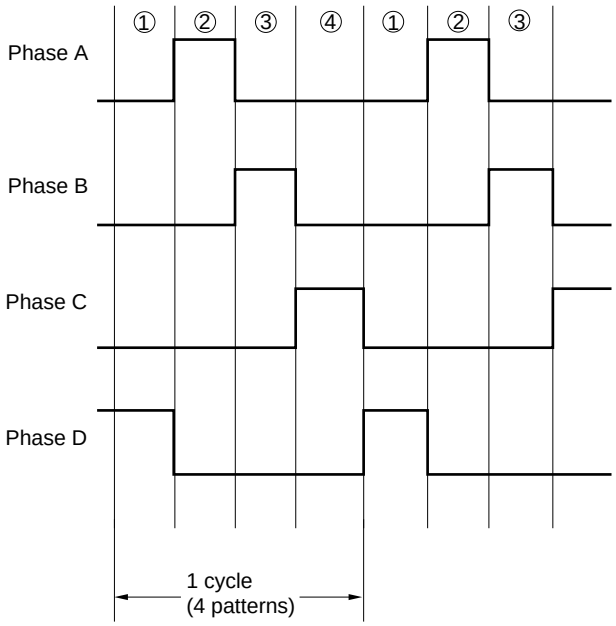


Figure 22-35. 1-2-Phase Excitation of 4-Phase Stepping Motor

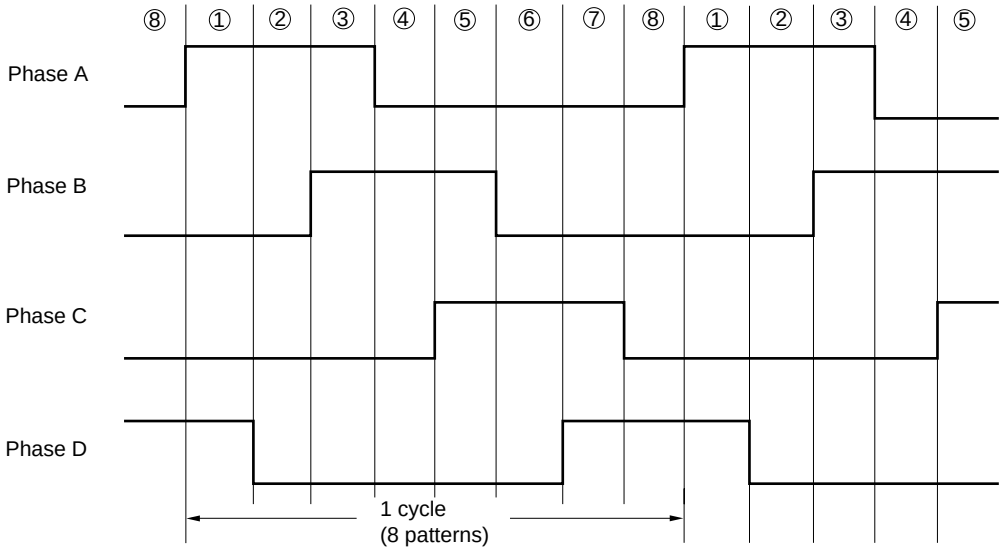
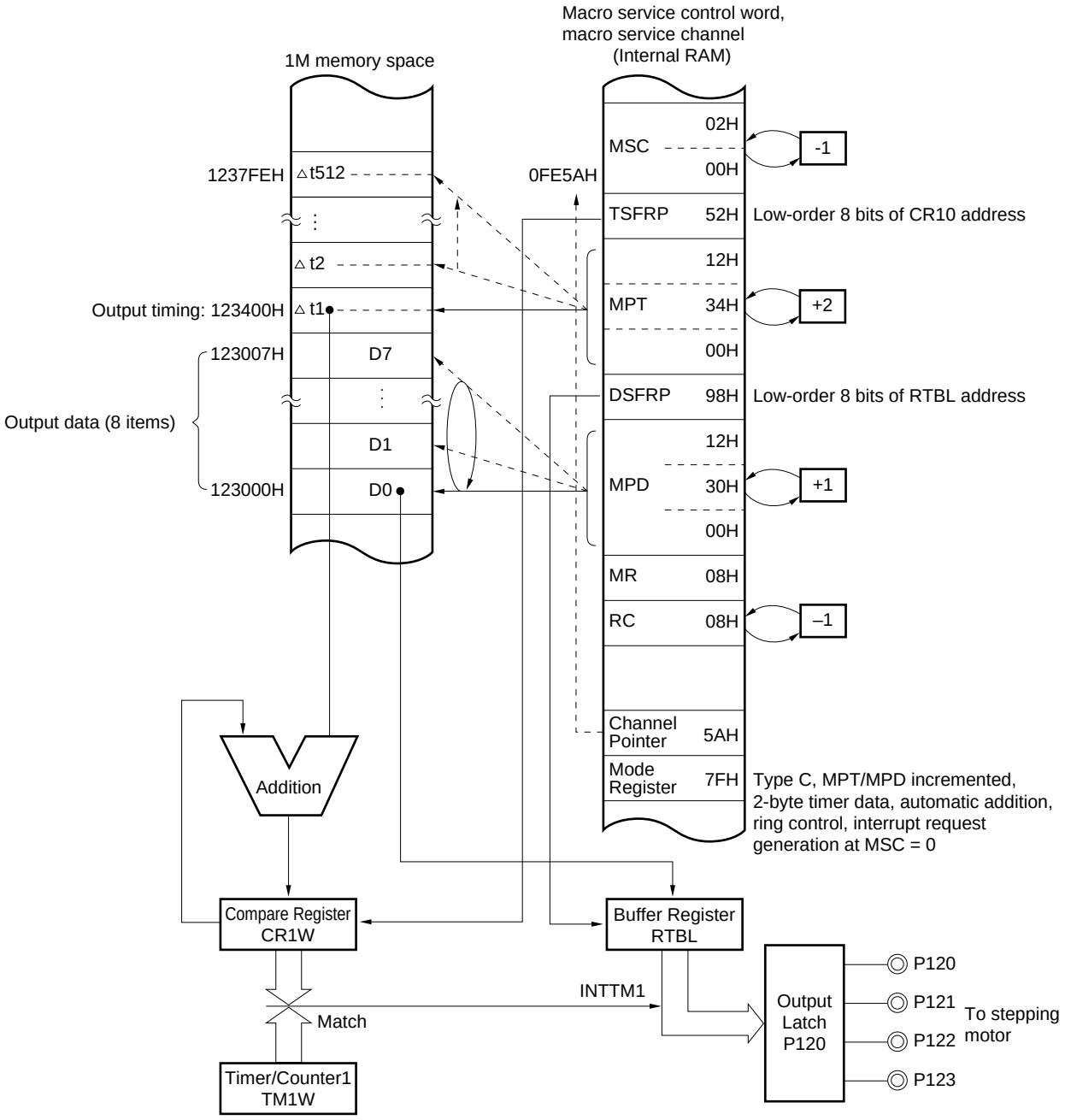


Figure 22-36. Automatic Addition Control + Ring Control Block Diagram 1 (When Output Timing Varies with 1-2-Phase Excitation)



Remark Internal RAM addresses in the figure are the values when the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values in the figure.

Figure 22-37. Automatic Addition Control + Ring Control Timing Diagram 1 (When Output Timing Varies with 1-2-Phase Excitation)

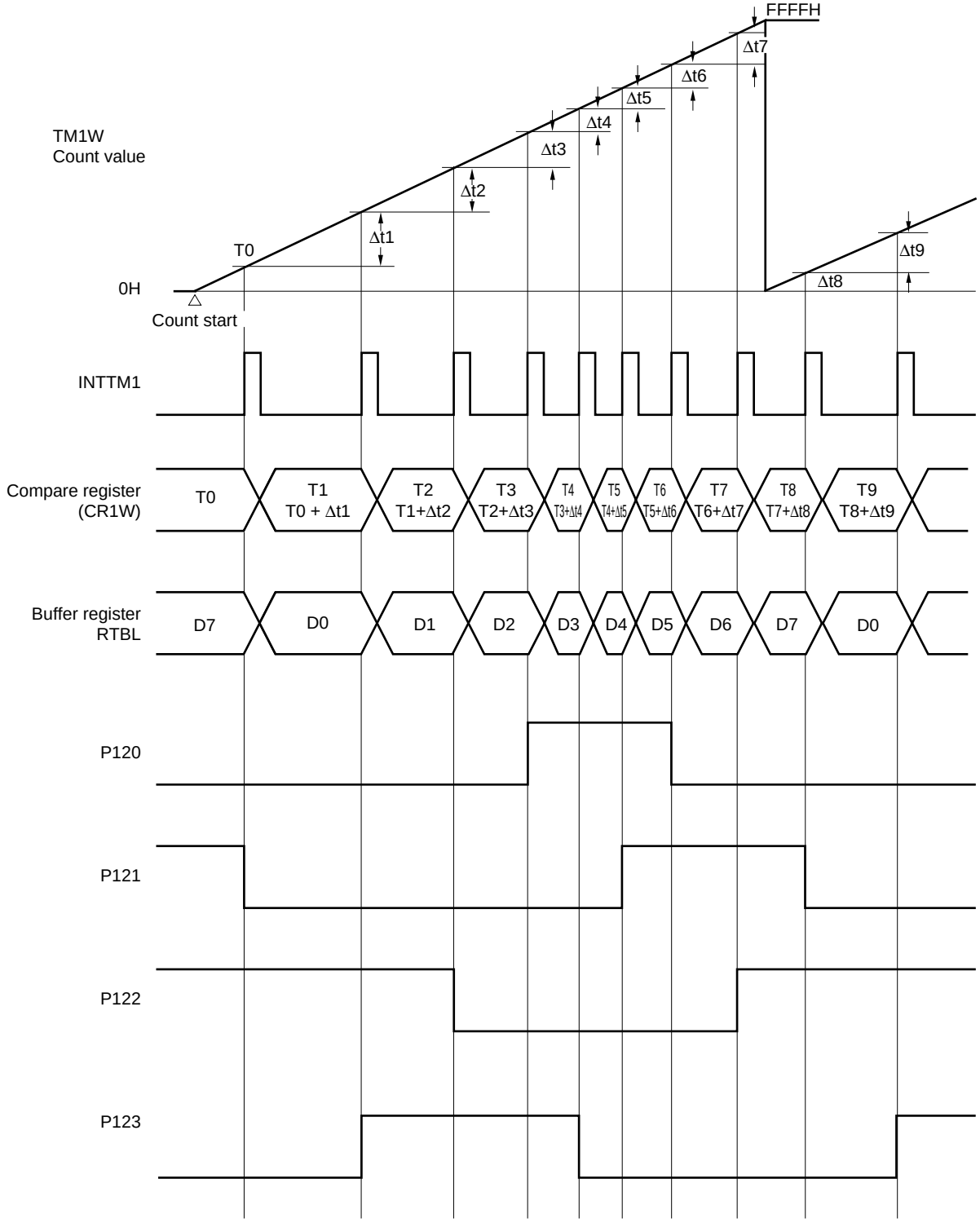
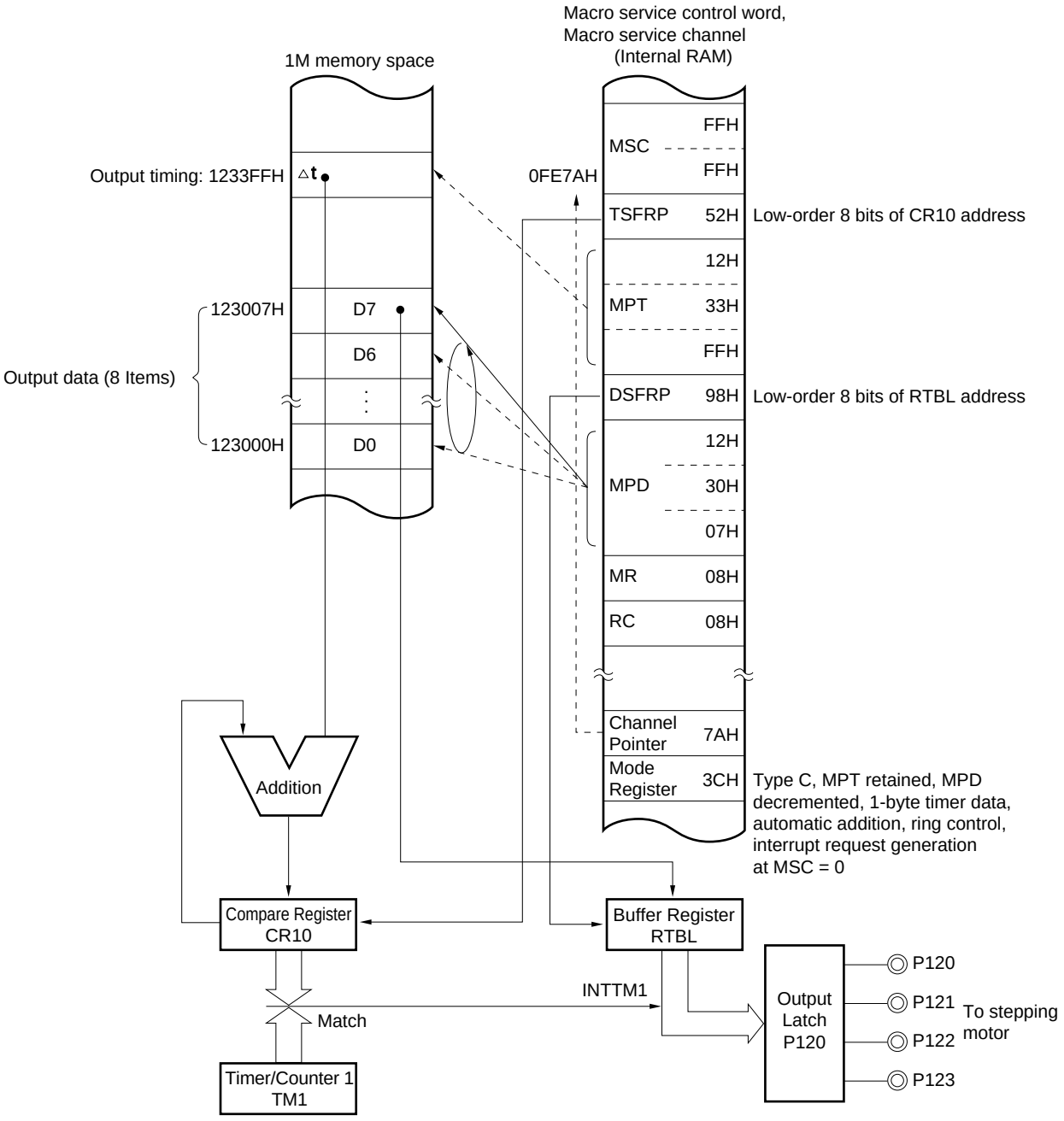
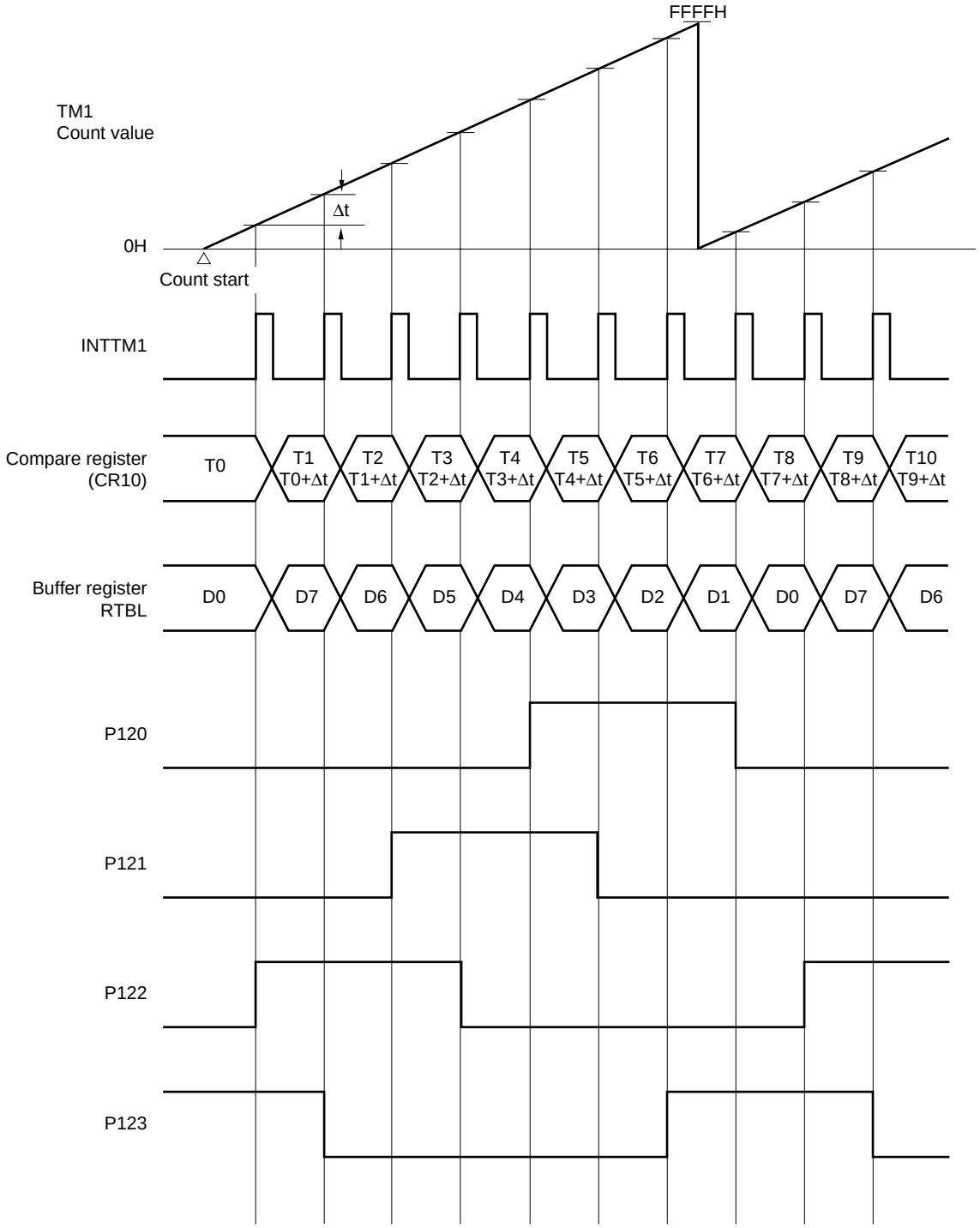


Figure 22-38. Automatic Addition Control + Ring Control Block Diagram 2 (1-2-Phase Excitation Constant-Velocity Operation)



Remark Internal RAM addresses in the figure are the values when the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values in the figure.

Figure 22-39. Automatic Addition Control + Ring Control Timing Diagram 2 (1-2-Phase Excitation Constant-Velocity Operation)



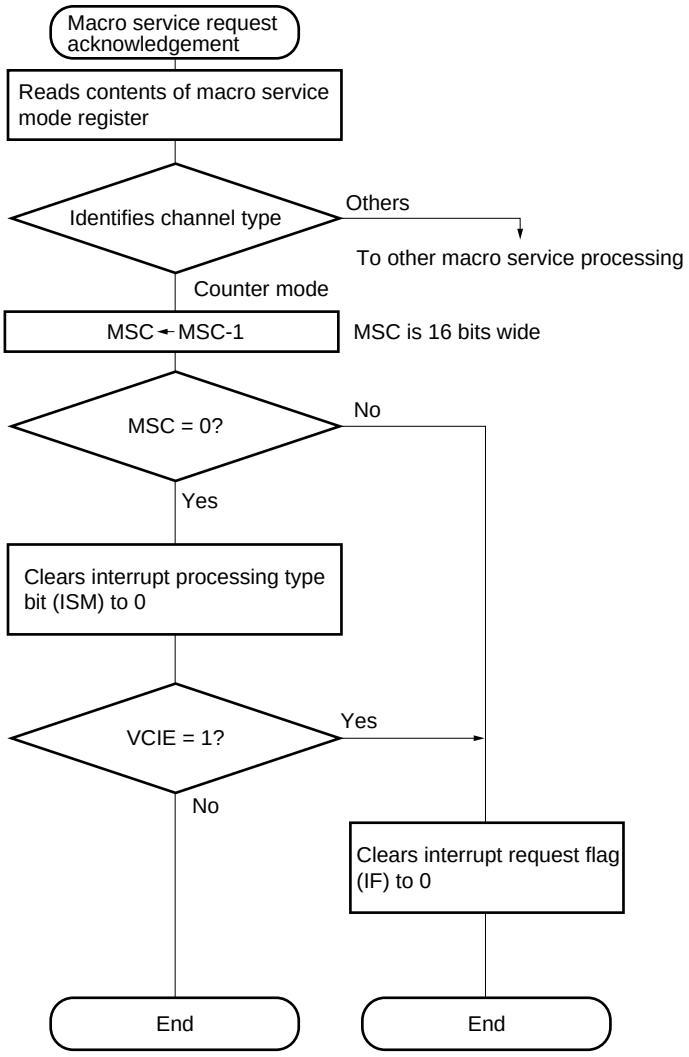
22.8.9 Counter mode

(1) Operation

MSC is decremented the number of times set in advance to the macro service counter (MSC).

Because the number of times an interrupt occurs can be counted, this function can be used as an event counter where the interrupt generation cycle is long.

Figure 22-40. Macro Service Data Transfer Processing Flow (Counter Mode)

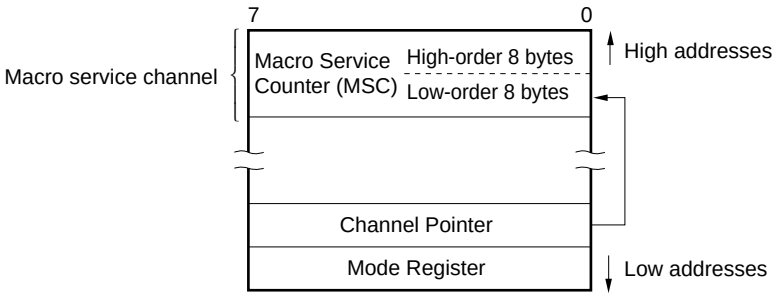


(Vectored interrupt request is generated)

(2) Configuration of macro service channel

The macro service channel consists of only a 16-bit macro service counter (MSC). The low-order 8 bits of the address of the MSC are written to the channel pointer.

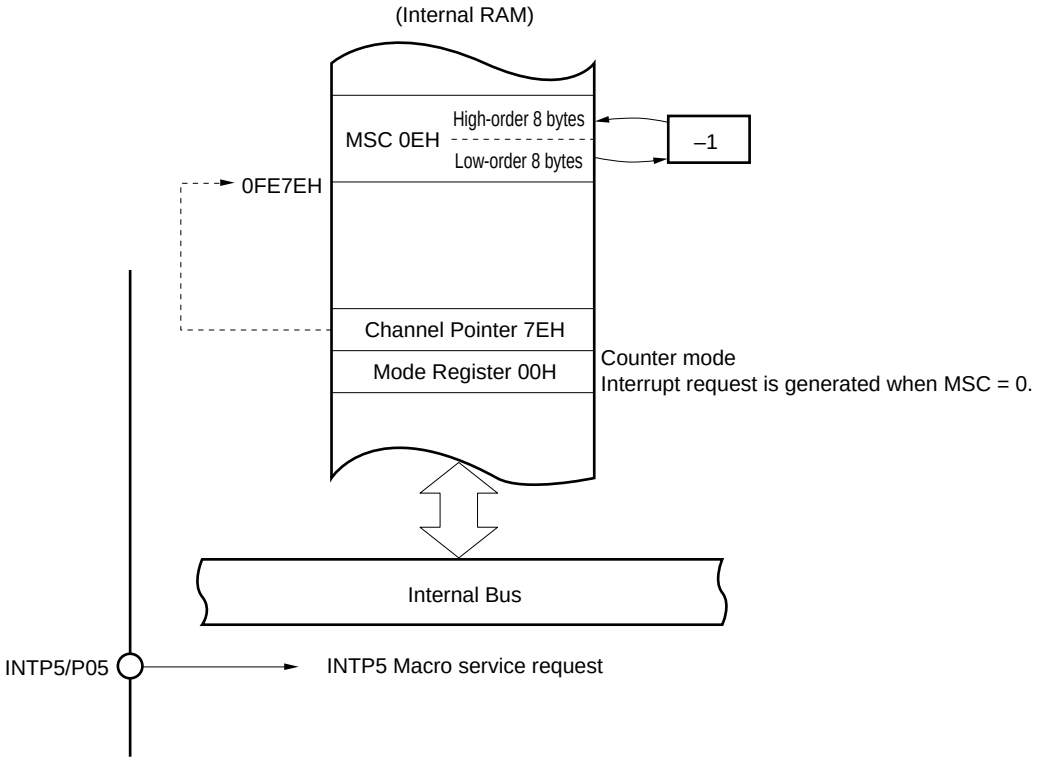
Figure 22-41. Counter Mode



(3) Example of using counter mode

Here is an example of counting the number of edges input to external interrupt pin INTP5.

Figure 22-42. Counting Number of Edges



Remark The internal RAM address in the figure above is the value when the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, add 0F0000H to this value.

22.9 When Interrupt Requests and Macro Service are Temporarily Held Pending

When the following instructions are executed, interrupt acknowledgment and macro service processing is deferred for 8 system clock cycles. However, software interrupts are not deferred.

EI
DI
BRK
BRKCS
RETCS
RETCSB !addr16
RETI
RETB
LOCATION 0H or LOCATION 0FH
POP PSW
POPU post
MOV PSWL, A
MOV PSWL, #byte
MOVG SP, # imm 24
Write instruction and bit manipulation instruction (excluding BT and BF) to interrupt control registers[Note] MK0, MK1 IMC, ISPR, and SNMI.
PSW bit manipulation instruction
(Excluding the BT PSWL.bit, $addr20, BF PSWL.bit, $addr20, BT PSWH.bit, $addr20, BF PSWH.bit, $addr20, SET1 CY, NOT1 CY, and CLR1 CY instructions)

Note Interrupt control registers: WDTIC, PIC0, PIC1, PIC2, PIC3, PIC4, PIC5, CSIIC0, SERIC1, SRIC1, STIC1, SERIC2, SRIC2, STIC2, TMIC3, TMIC00, TMIC01, TMIC1, TMIC2, ADIC, TMIC5, TMIC6, WTIC

Cautions 1. When an interrupt related register is polled using a BF instruction, etc., the branch destination of that BR instruction, etc., should not be that instruction. If a program is written in which a branch is made to that instruction itself, all interrupts and macro service requests will be held pending until a condition whereby a branch is not made by that instruction arises.

Bad Example

```
        ⋮
LOOP :  BF   PIC0.7,  $LOOP     All interrupts and macro service requests are held pending until PIC0.7 is 1.
        ×××                     ← Interrupts and macro service requests are not serviced
        ⋮                         until after execution of the instruction following the BF
                                  instruction.
```

Good Example (1)

```
        ⋮
LOOP :  NOP
        BF   PIC0.7,  $LOOP     ← Interrupts and macro service requests are serviced after
        ⋮                         execution of the NOP instruction, so that interrupts are
                                  never held pending for a long period.
```

Good Example (2)

```
        ⋮
LOOP :  BT   PIC0.7,  $NEXT     Using a BTCLR instruction instead of a BT instruction has
                                the advantage that the flag is cleared (0) automatically.
        BR   $LOOP              ← Interrupts and macro service requests are serviced after
NEXT :  ⋮                         execution of the BR instruction, so that interrupts are
                                  never held pending for a long period.
```

2. For a similar reason, if problems are caused by a long pending period for interrupts and macro service when instructions to which the above applies are used in succession, a time at which interrupts and macro service requests can be acknowledged should be provided by inserting an NOP instruction, etc., in the series of instructions.

22.10 Instructions Whose Execution is Temporarily Subspended by an Interrupt or Macro Service

Execution of the following instructions is temporarily suspended by an acknowledgeable interrupt request or macro service request, and the interrupt or macro service request is acknowledged. The suspended instruction is resumed after completion of the interrupt service program or macro service processing.

Temporarily suspended instructions:

MOVM, XCHM, MOVBK, XCHBK
CMPME, CMPMNE, CMPMC, CMPMNC
CMPBKE, CMPBKNE, CMPBKC, CMPBKNC
SACW

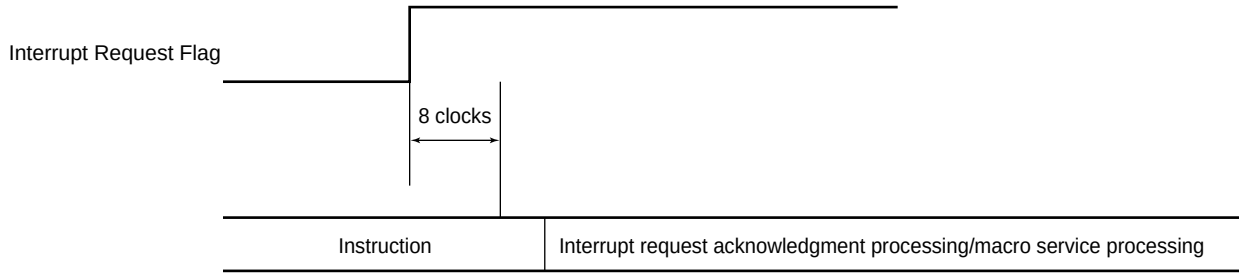
22.11 Interrupt and Macro Service Operation Timing

Interrupt requests are generated by hardware. The generated interrupt request sets (1) an interrupt request flag.

When the interrupt request flag is set (1), a time of 8 clocks (0.64 μs: f_{XX} = 12.5 MHz) is taken to determine the priority, etc.

Following this, if acknowledgment of that interrupt or macro service is enabled, interrupt request acknowledgment processing is performed when the instruction being executed ends. If the instruction being executed is one which temporarily defers interrupts and macro service, the interrupt request is acknowledged after the following instruction (refer to **22.9 When Interrupt Requests and Macro Service are Temporarily Held Pending** for deferred instructions).

Figure 22-43. Interrupt Request Generation and Acknowledgment (Unit: Clock = 1/f_{CLK})



22.11.1 Interrupt acknowledge processing time

The time shown in Table 22-7 is required to acknowledge an interrupt request. After the time shown in this table has elapsed, execution of the interrupt processing program is started.

Table 22-7. Interrupt Acknowledge Processing Time

(Unit: Clock = 1/f_{CLK})

Vector Table	IROM						EMEM					
Branch destination	IROM, PRAM			EMEM			PRAM			EMEM		
Stack	IRAM	PRAM	EMEM	IRAM	PRAM	EMEM	IRAM	PRAM	EMEM	IRAM	PRAM	EMEM
Vectored interrupts	26	29	37 + 4n	27	30	38 + 4n	30	33	41 + 4n	31	34	42 + 4n
Context switching	22	–	–	23	–	–	22	–	–	23	–	–

Remarks 1. IROM : internal ROM (with high-speed fetch specified)
PRAM : peripheral RAM of internal RAM (only when LOCATION 0 instruction is executed in the case of branch destination)
IRAM : internal high-speed RAM
EMEM : internal ROM when external memory and high-speed fetch are not specified

2. n is the number of wait states per byte necessary for writing data to the stack (the number of wait states is the sum of the number of address wait states and the number of access wait states).

3. It the vector table is EMEM, and if wait states are inserted in reading the vector table, add 2 m to the value of the vectored interrupt in the above table, and add m to the value of context switching, where m is the number of wait states per byte necessary for reading the vector table.

4. It the branch destination is EMEM and if wait states are inserted in reading the instruction at the branch destination, add that number of wait states.

5. If the stack is occupied by PRAM and if the value of the stack pointer (SP) is odd, add 4 to the value in the above table.

6. The number of wait states is the sum of the number of address wait states and the number of access wait states.

22.11.2 Processing time of macro service

Macro service processing time differs depending on the type of the macro service, as shown in Table 22-8.

Table 22-8. Macro Service Processing Time

(Units: Clock = 1/f_{CLK})

Processing Type of Macro Service			Data Area	
			IRAM	Others
Type A	SFR → memory	1 byte	24	–
		2 bytes	25	–
	Memory → SFR	1 byte	24	–
		2 bytes	26	–
Type B	SFR → memory		33	35
	Memory → SFR		34	36
Type C			49	53
Counter mode	MSC • 0		17	–
	USC = 0		25	–

Remarks 1. IRAM: internal high-speed RAM

2. In the following cases in the other data areas, add the number of clocks specified below.

- If the data size is 2 bytes with IROM or PRAM, and the data is located at an odd address: 4 clocks
- If the data size is 1 byte with EMEM: number of wait states for data access
- If the data size is 2 bytes with EMEM: 4 + 2n (where n is the number of wait states per byte)

3. If MSC = 0 with type A, B, or C, add 1 clock.

4. With type C, add the following value depending on the function to be used and the status at that time.

- Ring control: 4 clocks. Adds 7 more clocks if the ring counter is 0 during ring control.

22.12 Restoring Interrupt Function to Initial State

If an inadvertent program loop or system error is detected by means of an operand error interrupt, the watchdog timer, NMI pin input, etc., the entire system must be restored to its initial state. In the μPD784225, interrupt acknowledgment related priority control is performed by hardware. This interrupt acknowledgment related hardware must also be restored to its initial state, otherwise subsequent interrupt acknowledgment control may not be performed normally.

A method of initializing interrupt acknowledgment related hardware in the program is shown below. The only way of performing initialization by hardware is by $\overline{\text{RESET}}$ input.

Example

```
        MOVW    MK0, #0FFFFH    ;  Mask all maskable interrupts
        MOV     MK1L, #0FFH
IRESL   :
        CMP     ISPR, #0        ;  No interrupt service programs running?
        BZ      $NEXT
        MOVG    SP, #RETVAL     ;  Forcibly change SP location
        RETI                    ;  Forcibly terminate running interrupt service program, return
                                   address = IRESL
RETVAL  :
        DW      LOWW (IRESL)    ;  Stack data to return to IRESL with RETI instruction
        DB      0
        DB      HIGHW (IRESL)   ;  LOWW & HIGHW are assembler operators for calculating low-
                                   order 16 bits & high-order 16 bits respectively of symbol NEXT
NEXT    :
```

- It is necessary to ensure that a non-maskable interrupt request is not generated via the NMI pin during execution of this program.
- After this, on-chip peripheral hardware initialization and interrupt control register initialization are performed.
- When interrupt control register initialization is performed, the interrupt request flags must be cleared (0).

22.13 Cautions

(1) The in-service priority register (ISPR) is read-only. Writing to this register may result in misoperation.

(2) The watchdog timer mode register (WDM) can only be written to with a dedicated instruction (MOV WDM/#byte).

(3) The RETI instruction must not be used to return from a software interrupt caused by a BRK instruction. Use the RETB instruction.

(4) The RETCS instruction must not be used to return from a software interrupt caused by a BRKCS instruction. Use the RETCSB instruction.

(5) When a maskable interrupt is acknowledged by vectored interruption, the RETI instruction must be used to return from the interrupt. Subsequent interrupt related operations will not be performed normally if a different instruction is used.

(6) The RETCS instruction must be used to return from a context switching interrupt. Subsequent interrupt related operations will not be performed normally if a different instruction is used.

(7) Macro service requests are acknowledged and serviced even during execution of a non-maskable interrupt service program. If you do not want macro service processing to be performed during a non-maskable interrupt service program, you should manipulate the interrupt mask register in the non-maskable interrupt service program to prevent macro service generation.

(8) The RETI instruction must be used to return from a non-maskable interrupt. Subsequent interrupt acknowledgment will not be performed normally if a different instruction is used. Refer to **Section 22.12 Restoring Interrupt Function to Initial State** when a program is to be restarted from the initial status after a non-maskable interrupt acknowledgement.

(9) Non-maskable interrupts are always acknowledged, except during non-maskable interrupt service program execution (except when a high non-maskable interrupt request is generated during execution of a low-priority non-maskable interrupt service program) and for a certain period after execution of the special instructions shown in **22.9**. Therefore, a non-maskable interrupt will be acknowledged even when the stack pointer (SP) value is undefined, in particular after reset release, etc. In this case, depending on the value of the SP, it may happen that the program counter (PC) and program status word (PSW) are written to the address of a write-inhibited special function register (SFR) (refer to **Table 3-6 in 3.9 Special Function Registers (SFRs)**), and the CPU becomes deadlocked, or an unexpected signal output from a pin, or PC and PSW are written to an address is which RAM is not mounted, with the result that the return from the non-maskable interrupt service program is not performed normally and a software upset occurs.
Therefore, the program following $\overline{\text{RESET}}$ release must be as follows.

```
        CSEG  AT 0
        DW    STRT
        CSEG  BASE
STRT:
        LOCATION 0FH; or LOCATION 0
        MOVG  SP, #imm24
```

(10) When an interrupt related register is polled using a BF instruction, etc., the branch destination of that BR instruction, etc., should not be that instruction. If a program is written in which a branch is made to that instruction itself, all interrupts and macro service requests will be held pending until a condition whereby a branch is not made by that instruction arises.

Bad Example

```
        ⋮
LOOP:   BF   PIC0.7, $LOOP      All interrupts and macro service requests are held pending until
                                PIC0.7 is 1.
        ×××                     ← Interrupts and macro service requests are not serviced until after
        ⋮                       execution of the instruction following the BF instruction.
```

Good Example (1)

```
LOOP:   NOP
        BF   PIC0.7, $LOOP      ← Interrupts and macro service requests are serviced after execution of
                                the NOP instruction, so that interrupts are never held pending for a
        ⋮                       long period.
```

Good Example (2)

```
        ⋮
LOOP:   BT   PIC0.7, $NEXT      Using a BTCLR instruction instead of a BT instruction has the
                                advantage that the flag is cleared (0) automatically.
        BR   $LOOP              ← Interrupts and macro service requests are serviced after execution of
                                the BR instruction, so that interrupts are never held pending for a long
                                period.
NEXT:   ⋮
```

(11) For a similar reason to that given in (10), if problems are caused by a long pending period for interrupts and macro service when instructions to which the above applies are used in succession, a time at which interrupts and macro service requests can be acknowledged should be provided by inserting an NOP instruction, etc., in the series of instructions.

[MEMO]

CHAPTER 23 LOCAL BUS INTERFACE FUNCTIONS

23.1 External Memory Expansion Function

The external memory expansion function connects external memory to the areas other than the internal ROM, RAM, and SFR.

A time-divided address/data bus is used to connect external memory. When external memory is connected, the number of ports used can be reduced.

When external memory is connected, ports 4 to 6 are used. Ports 4 and 6 control address/data and read/write strobes, and wait and address strobes, etc.

Table 23-1. Pin Functions in External Memory Expansion Mode

Pin Functions in External Device Connect		Alternate Functions
Name	Function	
AD0 to AD7	Multiplexed address/data bus	P40 to P47
A8 to A15	Middle address bus	P50 to P57
A16 to A19	High address bus	P60 to P63
$\overline{RD}$	Read strobe	P64
$\overline{WR}$	Write strobe	P65
$\overline{WAIT}$	Wait signal	P66
ASTB	Address strobe	P67

Table 23-2. Pin States in Ports 4 to 6 in External Memory Expansion Mode

Port / External Expansion Mode	Port 4	Port 5								Port 6							
	0 to 7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Single-chip mode	Port	Port								Port							
256-kbyte expansion mode	Address/data	Address								Address				Port			$\overline{RD}$, $\overline{WR}$, $\overline{WAIT}$, ASTB
1-Mbyte expansion	Address/data	Address								Address							$\overline{RD}$, $\overline{WR}$, $\overline{WAIT}$, ASTB

Caution When the external wait function is not used, the $\overline{WAIT}$ pin can be used as the port in all of the modes.

23.2 Control Registers

(1) Memory expansion mode register (MM)

MM is an 8-bit register that controls the external expanded memory, sets the number of address waits, and controls the internal fetch cycle.

MM can be read or written by a 1-bit or 8-bit memory manipulation instruction. Figure 23-1 shows the MM format. $\overline{\text{RESET}}$ input sets MM to 20H.

Figure 23-1. Memory Expansion Mode Register (MM) Format

Address: 0FFC4H After Reset: 20H R/W

Symbol	7	6	5	4	3	2	1	0
MM	IFCH	0	AW	0	MM3	MM2	MM1	MM0

IFCH	Internal ROM Fetch
0	Fetch at the same speed as from external memory. All of the wait control settings are valid.
1	High-speed fetch The wait control settings are invalid.

AW	Address Wait Setting
0	An address wait is not inserted.
1	A one-clock address wait is inserted in the address output timing.

MM3	MM2	MM1	MM0	Mode	Port 4 (P40 to P47)	Port 5 (P50 to P57)	P60 to P63		P64	P65	P66
0	0	0	0	Single-chip mode	Port						
1	0	0	0	256-kbyte expansion mode	AD0 to AD7	A8 to A15	A16, A17	Port	$\overline{\text{RD}}$	$\overline{\text{WR}}$	ASTB
1	0	0	1	1-Mbyte expansion mode			A16 to A19				
Other than above				Setting prohibited							

(2) Programmable wait control register (PWC1)

PWC1 is an 8-bit register that sets the number of waits.

The insertion of wait cycles is controlled by PWC1 over the entire space.

PWC1 can be read and written by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PWC1 to AAH.

Figure 23-2. Programmable Wait Control Register (PWC1) Format

Address: 0FFC7H After Reset: AAH R/W

Symbol	7	6	5	4	3	2	1	0
PWC1	×	×	×	×	×	×	PW01	PW00

PW01	PW00	Insertion Wait Cycles	Data Access Cycles, Fetch Cycles
0	0	0	3
0	1	1	4
1	0	2	5
1	1	Low level period that is input at the $\overline{\text{WAIT}}$ pin	–

Remarks 1. The insertion of wait cycles is controlled by the entire address space (except for the peripheral RAM area).

2. ×: don't care

23.3 Memory Map for External Memory Expansion

Figures 23-4 and 23-5 show the memory map during memory expansion. Even during memory expansion, an external device at the same address as the internal ROM area, internal RAM area, or SFR area (except for the external SFR area (0FFD0H-0FFDFH)) cannot be accessed. If these areas are accessed, the memory and SFR in μPD784225 are accessed with priority, and the ASTB, $\overline{RD}$, and $\overline{WD}$ signals are not output (remaining at the inactive level). The output level of the address bus remains at the previous output level. The output of the address/data bus has a high impedance.

Except in the 1-Mbyte expansion mode, an address for external output is output in the state that masked the high order side of the address set by the program.

<Example 1>

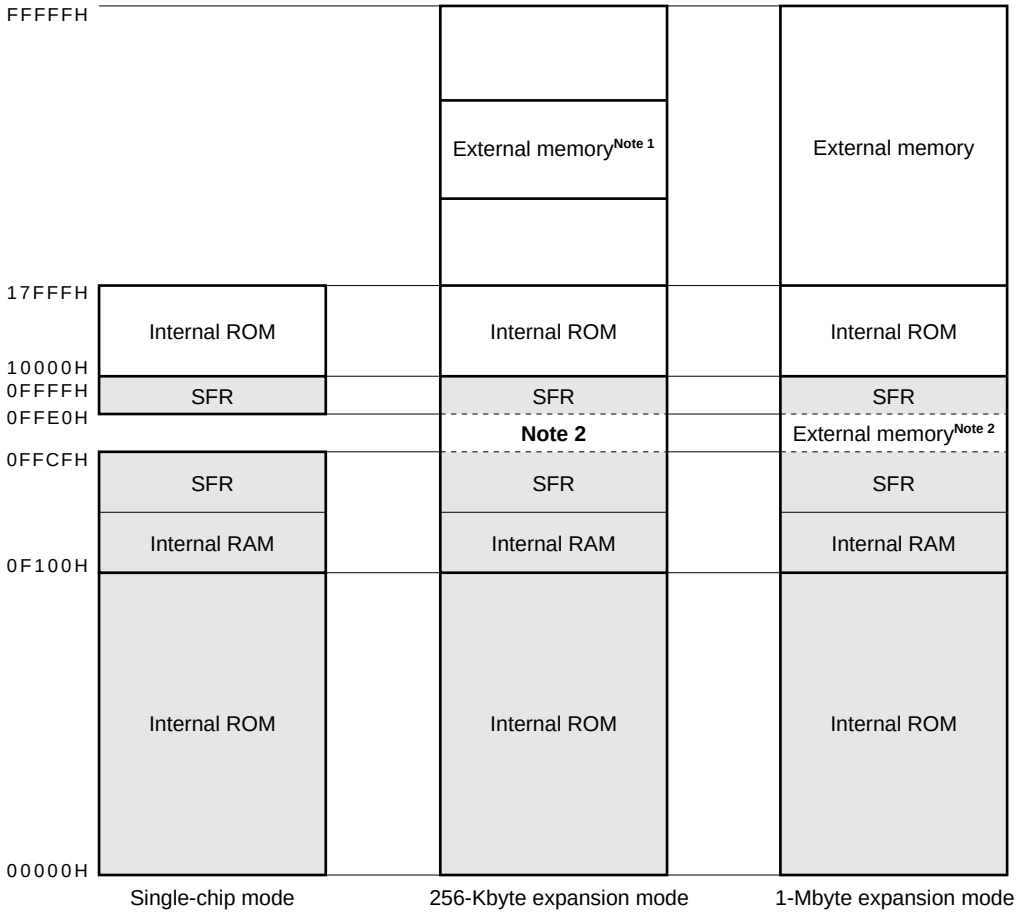
When address 54321H is accessed in the program in the 256-Kbyte expansion mode, the address that is output becomes 14321H.

<Example 2>

When address 67821H is accessed in the program in the 256-Kbyte expansion mode, the address that is output becomes 27821H.

Figure 23-3. μPD784224 Memory Map (1/2)

(a) When executing the LOCATION 0 instruction

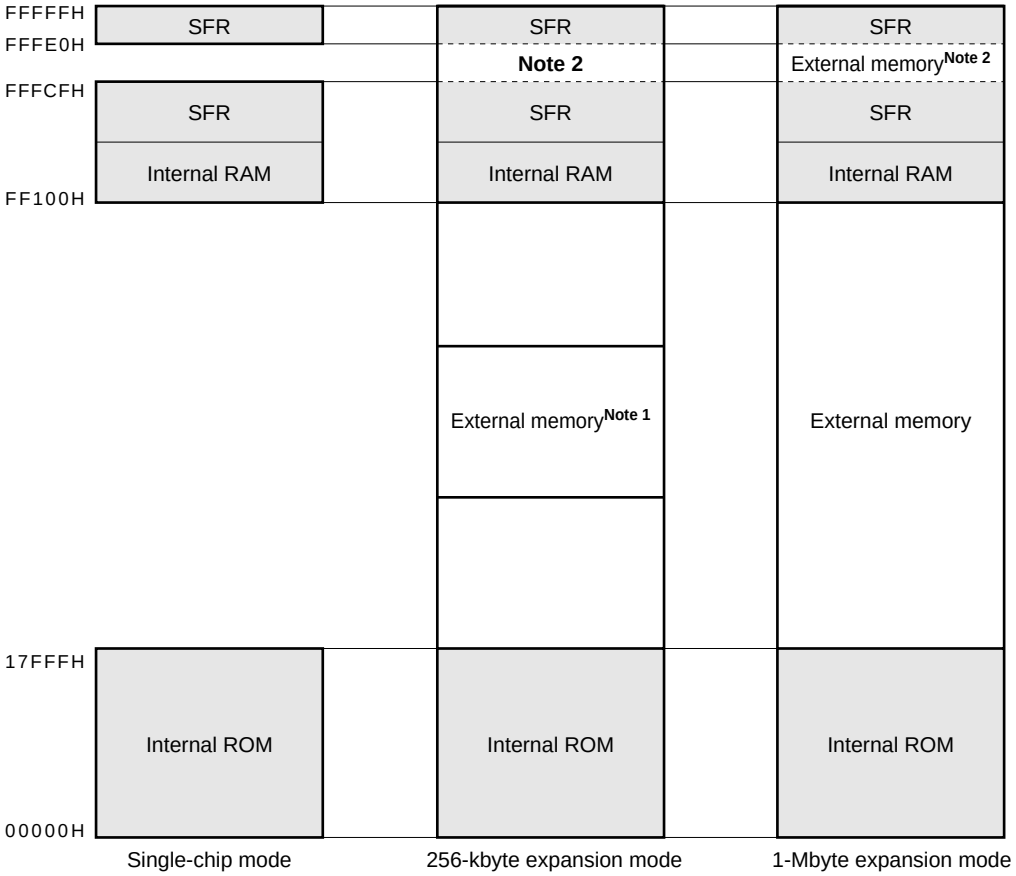


Notes 1. Area having any expansion size in the unshaded parts

2. External SFR area

Figure 23-3. μPD784224 Memory Map (2/2)

(b) When executing the LOCATION 0FH instruction

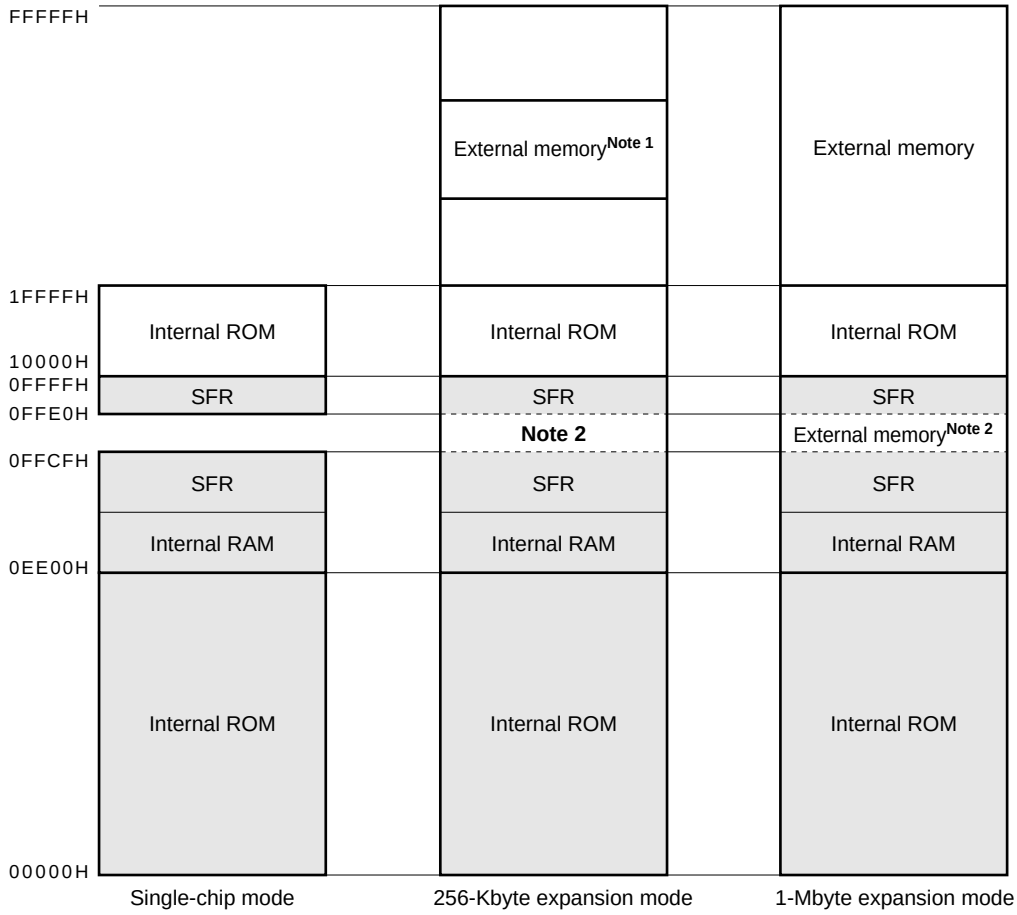


Notes 1. Area having any expansion size in the unshaded parts

2. External SFR area

Figure 23-4. μPD784225 Memory Map (1/2)

(a) When executing the LOCATION 0 instruction

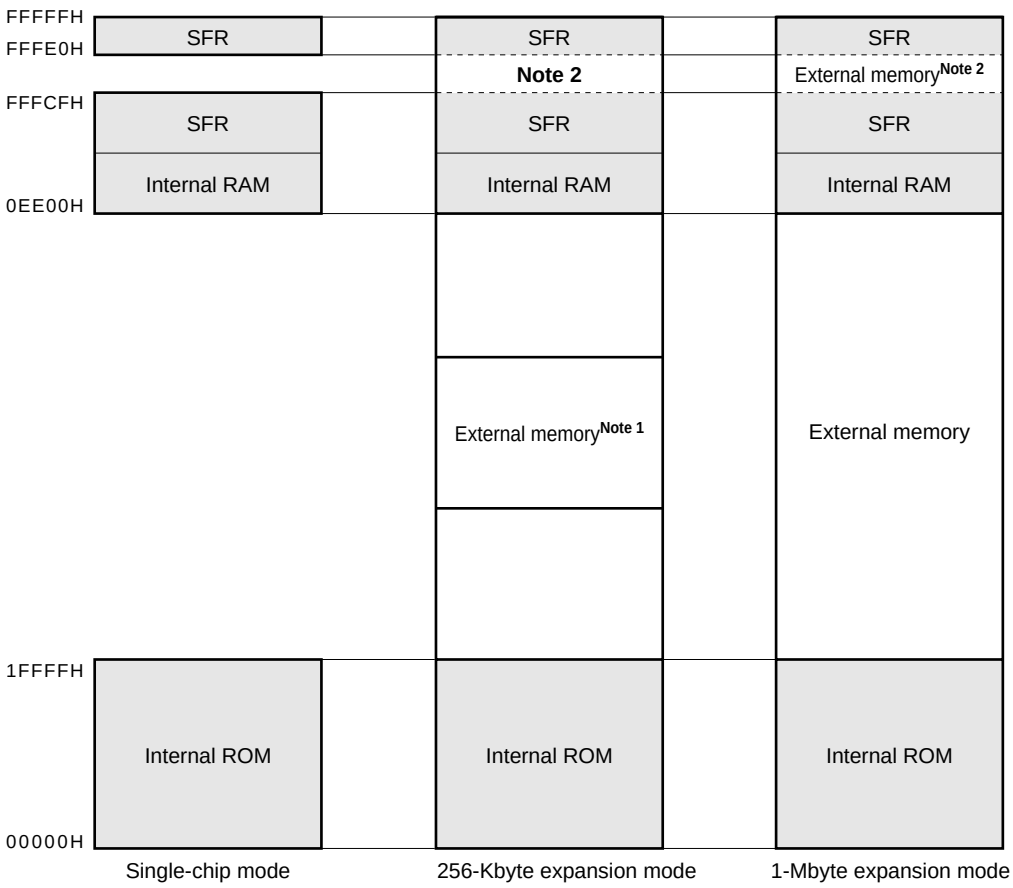


Notes 1. Area having any expansion size in the unshaded parts

2. External SFR area

Figure 23-4. μPD784225 Memory Map (2/2)

(b) When executing the LOCATION 0FH instruction



Notes 1. Area having any expansion size in the unshaded parts

2. External SFR area

23.4 Timing of External Memory Expansion Functions

The timing control signal output pins in the external memory expansion mode are described below

(1) $\overline{\text{RD}}$ pin (shared by : P64)

This pin outputs the read strobe during an instruction fetch or a data access from external memory.
During an internal memory access, the read strobe is not output (held at the high level)

(2) $\overline{\text{WR}}$ pin (shared by : P65)

This pin outputs the write strobe during a data access to external memory.
During an internal memory access, the write strobe is not output (held at the high level).

(3) $\overline{\text{WAIT}}$ pin (shared by : P66)

This pin inputs the external wait signal.
When the external wait is not used, $\overline{\text{WAIT}}$ pin can be used as an I/O port.
During an internal memory access, the external wait signal is ignored.

(4) ASTB pin (shared by : P67)

This pin always outputs the address strobe in any instruction fetch or data access from external memory.
During an internal memory access, the address strobe is not output (keeps low level).

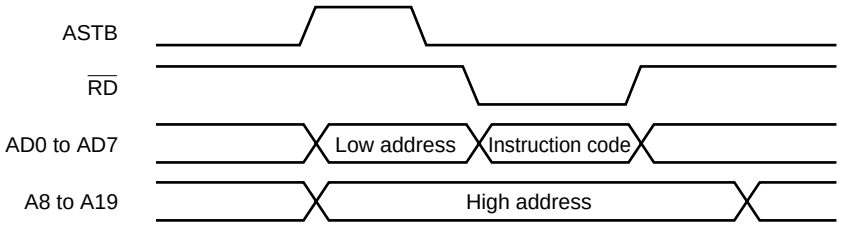
(5) AD0 to AD7, A8 to A15, A16 to A19 pins (shared by : P40 to P47, P50 to P57, P60 to P63)

These pins output the address and data signals. When an instruction is fetched or data is accessed from external memory, valid signals are output or input.
During an internal memory access, the signals do not change.

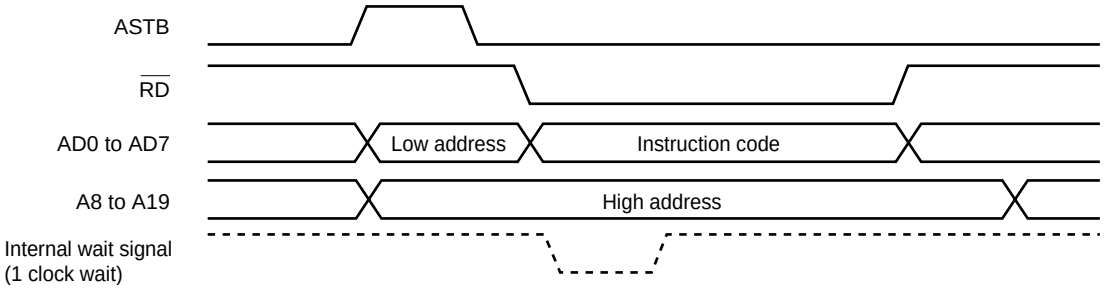
Figures 23-5 to 23-8 are the timing charts.

Figure 23-5. Instruction Fetch from External Memory in External Memory Expansion Mode

(a) Setting 0 wait cycles (PW01 = 0, PW00 = 0, 0)



(b) Setting 1 wait cycle (PW01 = 0, PW00 = 0, 1)



(c) Setting an external wait (PW01 = 1, PW00 = 1, 1)

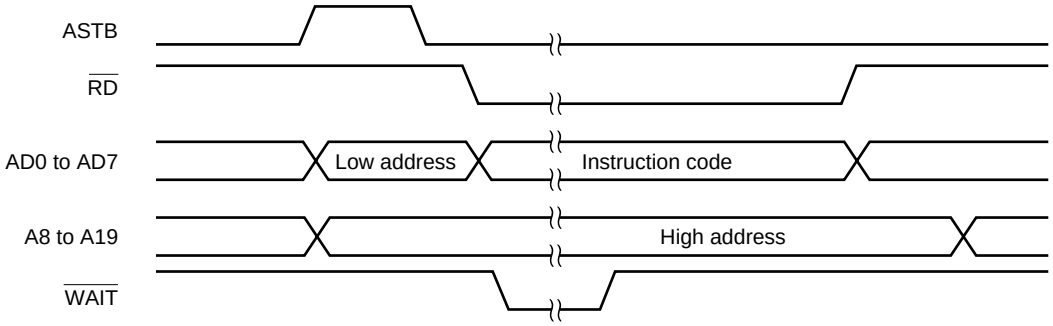
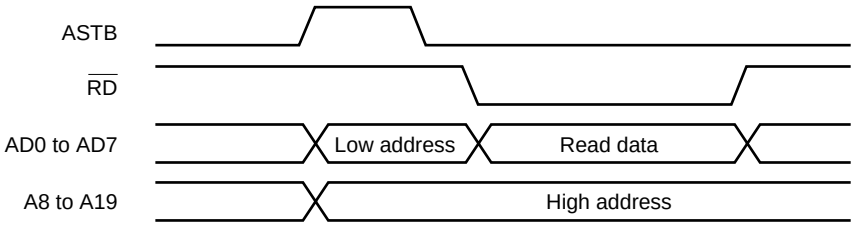
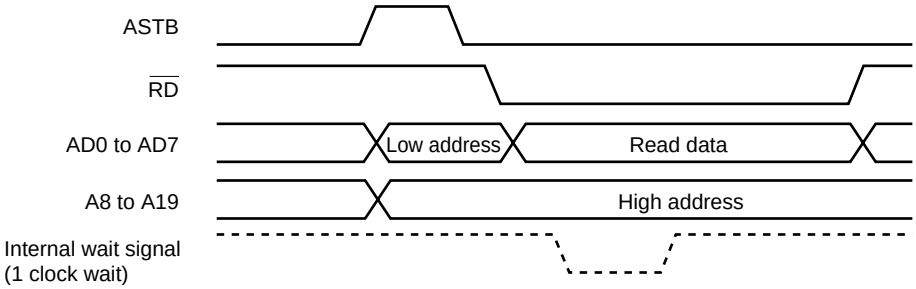


Figure 23-6. Read Timing for External Memory in External Memory Expansion Mode

(a) Setting 0 wait cycles (PW01 = 0, PW00 = 0, 0)



(b) Setting 1 wait cycle (PW01 = 0, PW00 = 0, 1)



(c) Setting an external wait (PW01 = 1, PW00 = 1, 1)

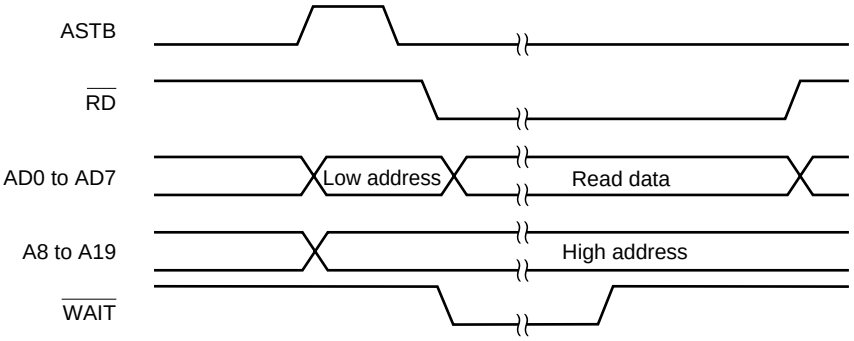
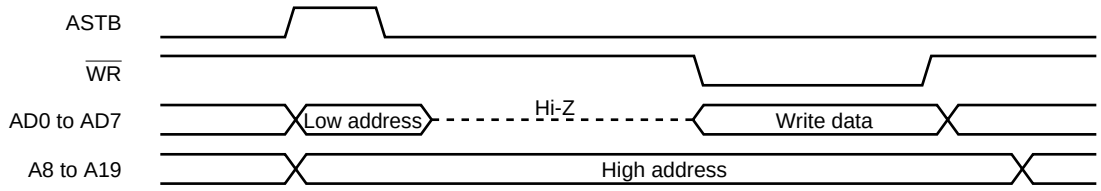
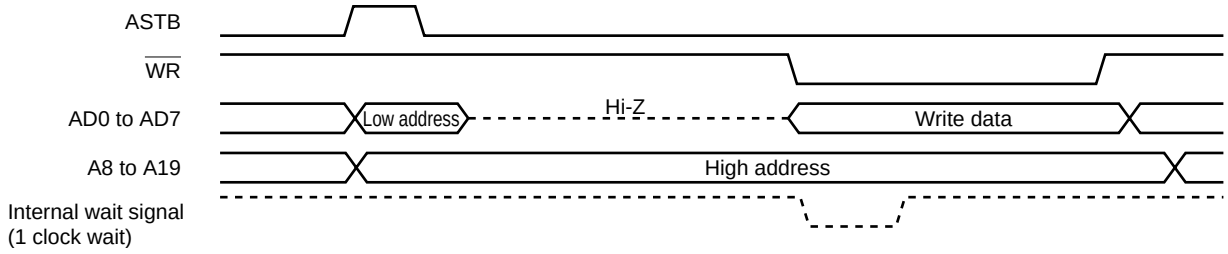


Figure 23-7. External Write Timing for External Memory in External Memory Expansion Mode

(a) Setting 0 wait cycles (PW01 = 0, PW00 = 0, 0)



(b) Setting 1 wait cycle (PW01 = 0, PW00 = 0, 1)



(c) Setting an external wait (PW01 = 1, PW00 = 1, 1)

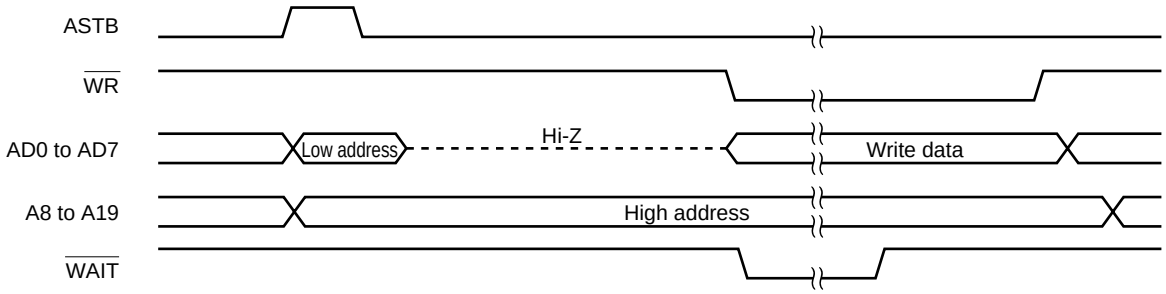
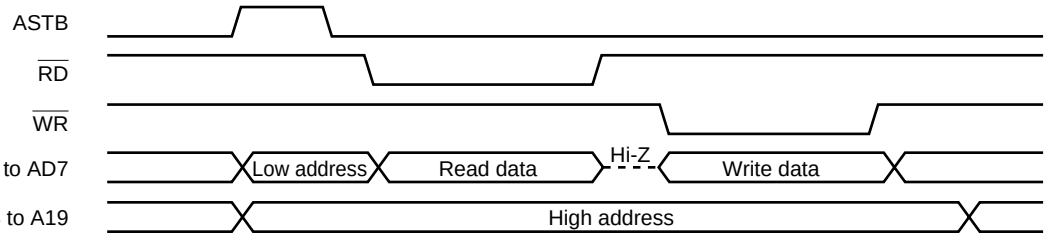
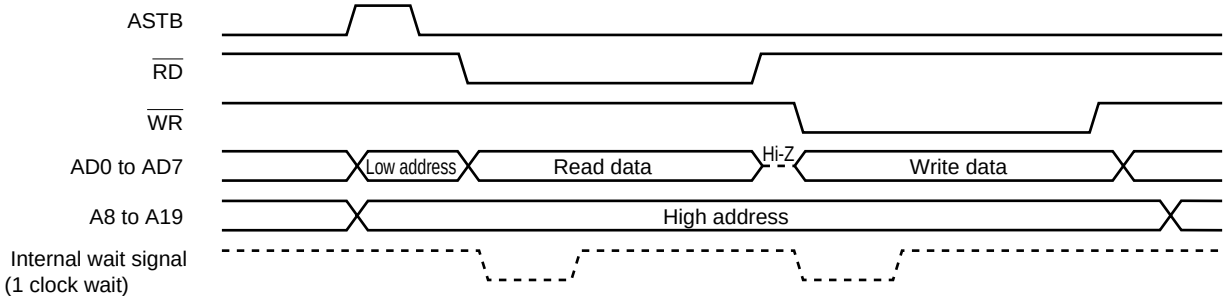


Figure 23-8. Read Modify Write Timing for External Memory in External Memory Expansion Mode

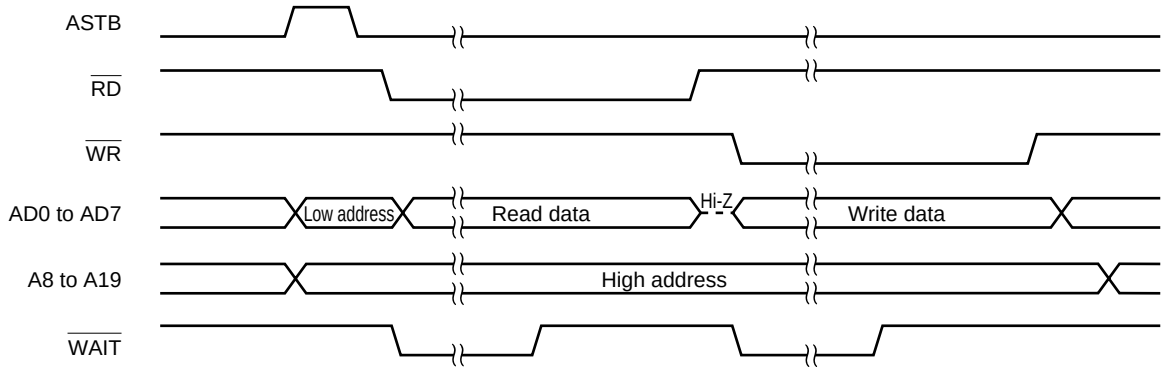
(a) Setting 0 wait cycles (PW01 = 0, PW00 = 0, 0)



(b) Setting 1 wait cycle (PW01 = 0, PW00 = 0, 1)



(c) Setting an external wait (PW01 = 1, PW00 = 1, 1)



23.5 Wait Functions

If slow memory and I/O are connected externally to the μPD784225, waits can be inserted in the external memory access cycle.

During the wait cycle, there is an address wait to guarantee the address decoding time and an access wait to guarantee the access time.

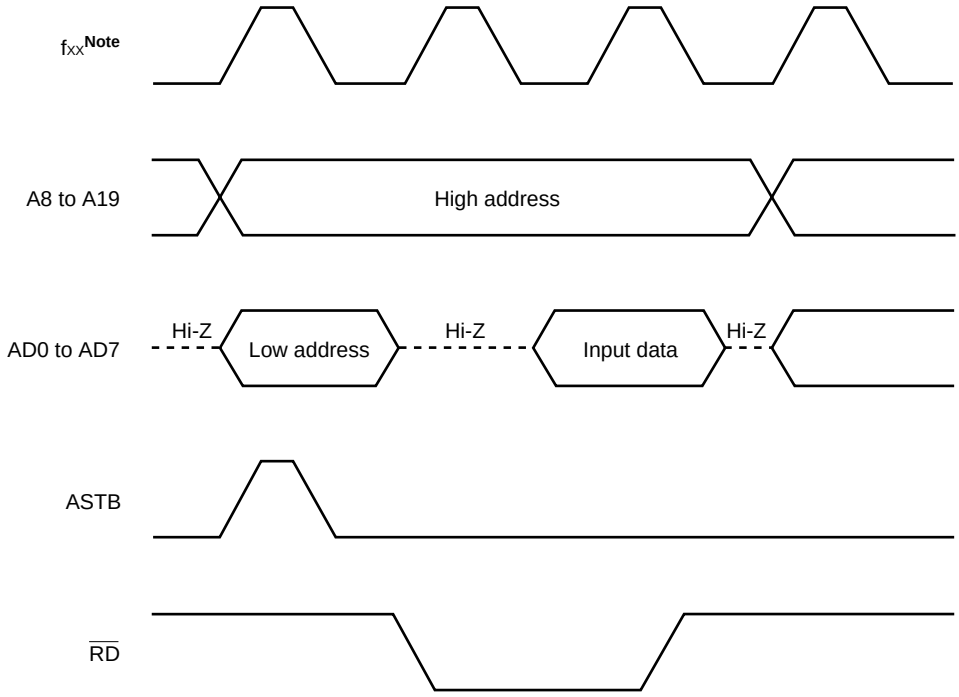
23.5.1 Address wait

An address wait guarantees the address decoding time. By setting the AW bit in the memory expansion mode register (MM) to one, an address wait is inserted into the entire memory access time**Note**. When the address wait is inserted, the high level period of the ASTB signal is lengthened by one system clock (when 80 ns, f_{XX} = 12.5 MHz).

Note This excludes the internal RAM, internal SFR, and internal ROM during a high-speed fetch. When the internal ROM access is set to have the same cycle as an external ROM access, an address wait is inserted during an internal ROM access.

Figure 23-9. Read/Write Timing by Address Wait Function (1/3)

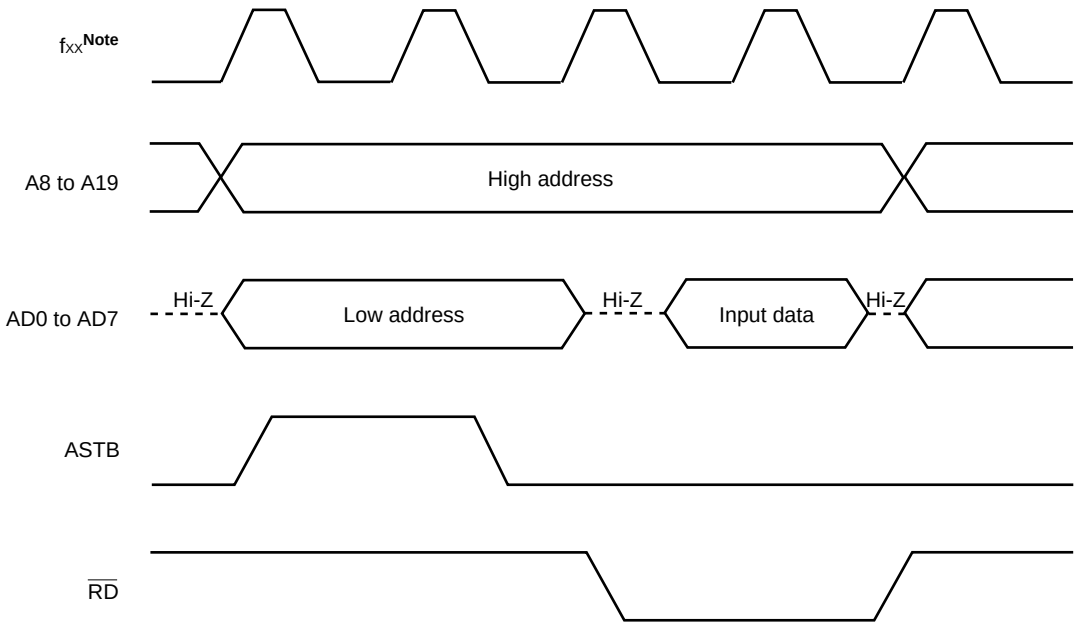
(a) Read timing when an address wait is not inserted



Note f_{XX}: Main system clock frequency. This signal is only in the μPD784225.

Figure 23-9. Read/Write Timing by Address Wait Function (2/3)

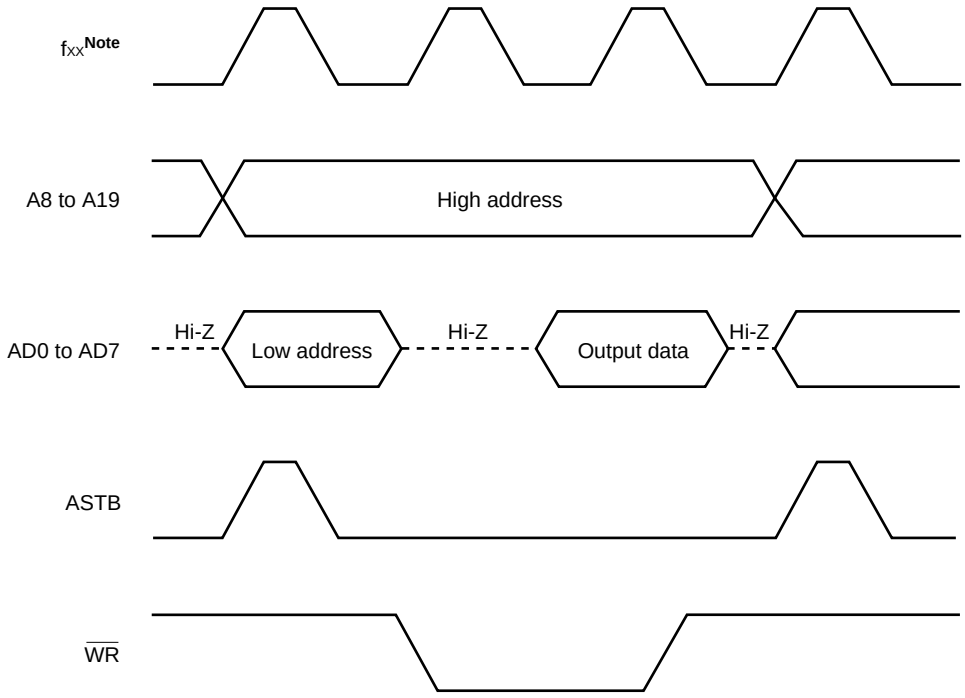
(b) Read timing when an address wait is inserted



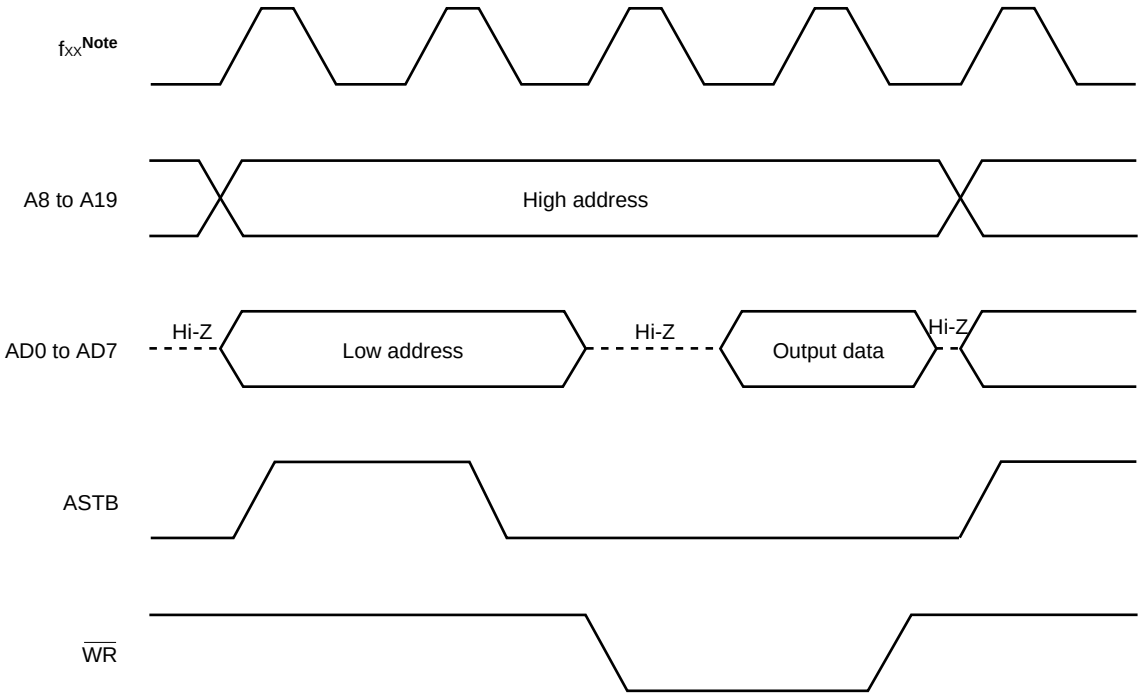
Note fXX: Main system clock frequency. This signal is only in the μPD784225.

Figure 23-9. Read/Write Timing by Address Wait Function (3/3)

(c) Write timing when an address wait is not inserted



(d) Write timing when an address wait is inserted



Note fXX: Main system clock frequency. This signal is only in the μPD784225.

23.5.2 Access wait

An access wait is inserted during low $\overline{RD}$ and $\overline{WR}$ signals. The low level is lengthened by 1/f_{XX} (80 ns, f_{XX} = 12.5 MHz) per cycle.

The wait insertion methods are the programmable wait function that automatically inserts a preset number of cycles and the external wait function that is controlled from the outside by the wait signal.

Wait cycle insertion control is set by the programmable wait control register (PWC1) for the 1-Mbyte memory space. If an internal ROM or internal RAM is accessed during a high-speed fetch, a wait is not inserted. If accessing an internal SFR, a wait is inserted based on required timing unrelated to this setting.

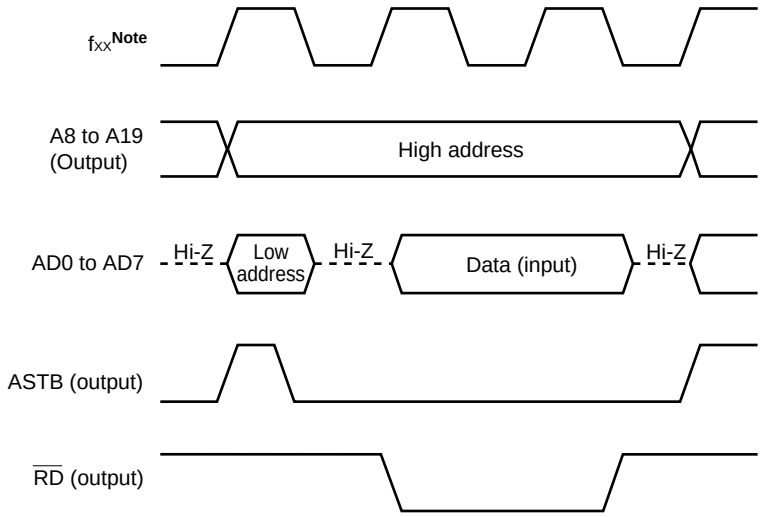
If set so that an access has the same number of cycles as for an external ROM, a wait is also inserted in an internal ROM access in accordance with the PWC1 setting.

If there is space that was externally selected to be controlled by the wait signal by PWC1, pin P66 acts as the $\overline{WAIT}$ signal input pin. $\overline{RESET}$ input makes pin P66 act as an ordinary I/O port.

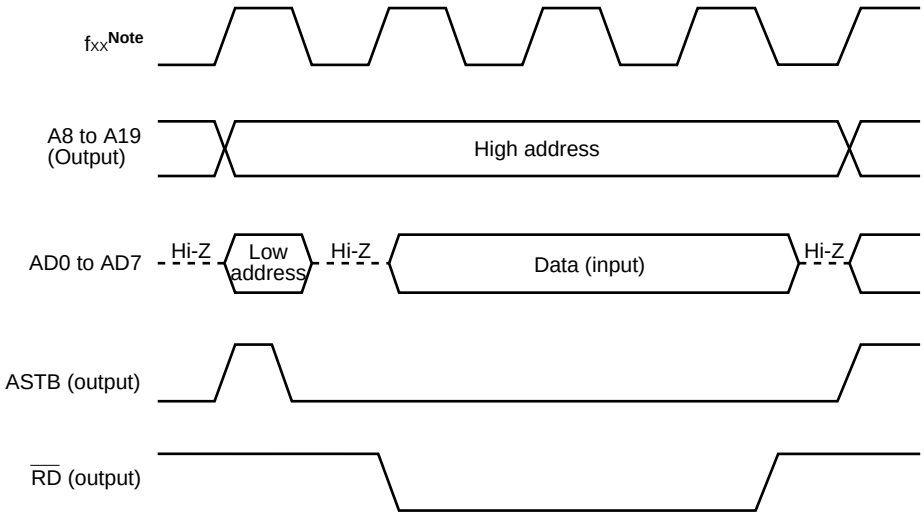
Figures 23-10 to 23-12 show the bus timing when an access wait is inserted.

Figure 23-10. Read Timing by Access Wait Function (1/2)

(a) Setting 0 wait cycles (PW01 = 0, PW00 = 0, 0)



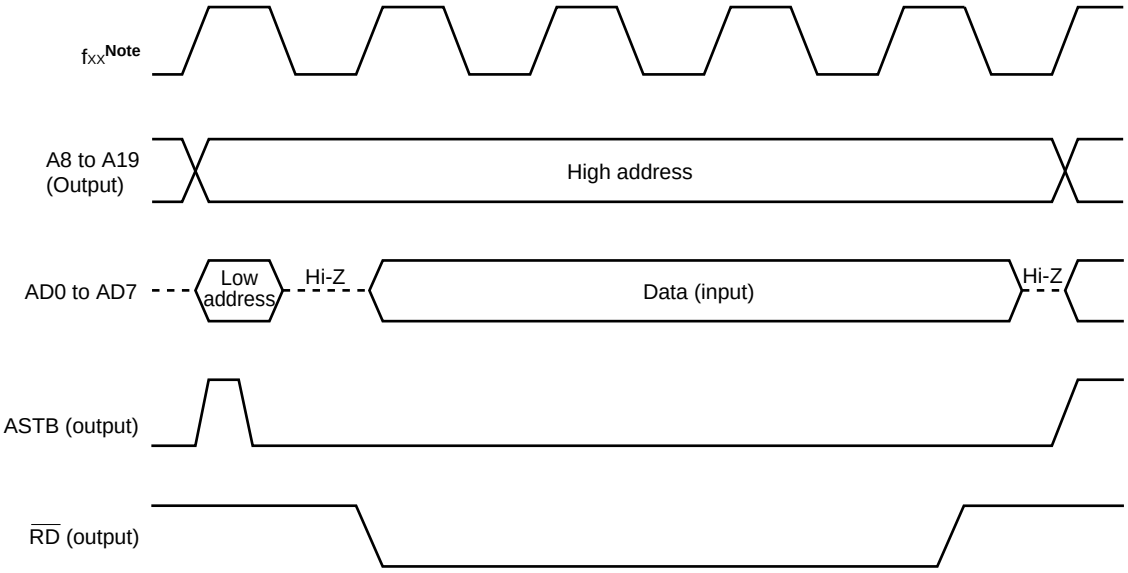
(b) Setting 1 wait cycle (PW01 = 0, PW00 = 0, 1)



Note fXX: Main system clock frequency. This signal is only in the μPD784225.

Figure 23-10. Read Timing by Access Wait Function (2/2)

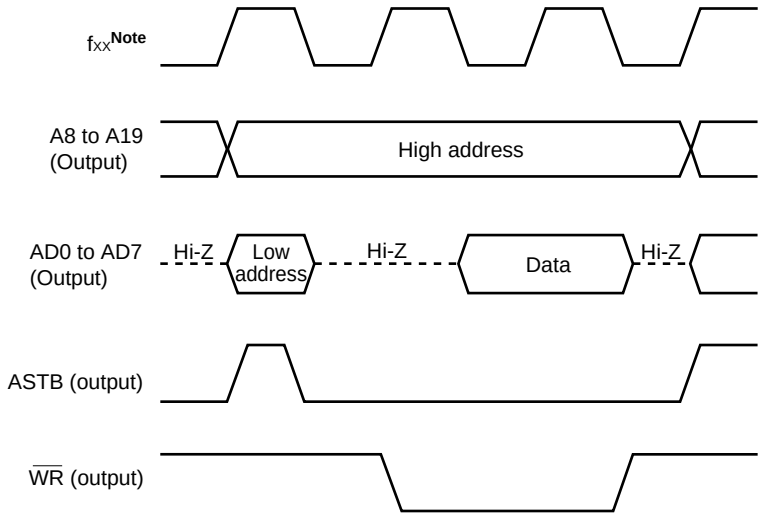
(c) Setting 2 wait cycles (PW01 = 1, PW00 = 1, 0)



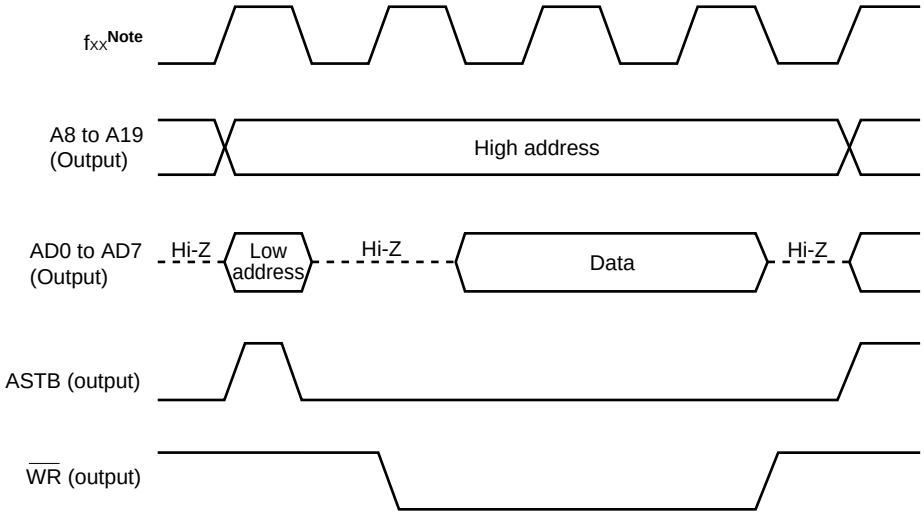
Note fXX: Main system clock frequency. This signal is only in the μPD784225.

Figure 23-11. Write Timing by Access Wait Function (1/2)

(a) Setting 0 wait cycles (PW01 = 0, PW00 = 0, 0)



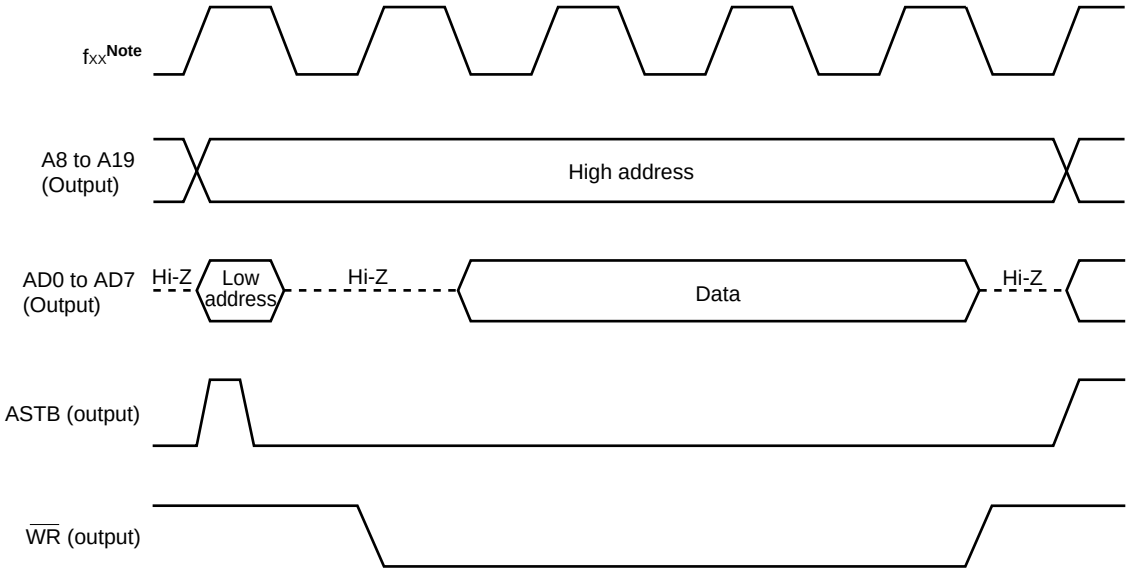
(b) Setting 1 wait cycle (PW01 = 0, PW00 = 0, 1)



Note fXX: Main system clock frequency. This signal is only in the μPD784225.

Figure 23-11. Write Timing by Access Wait Function (2/2)

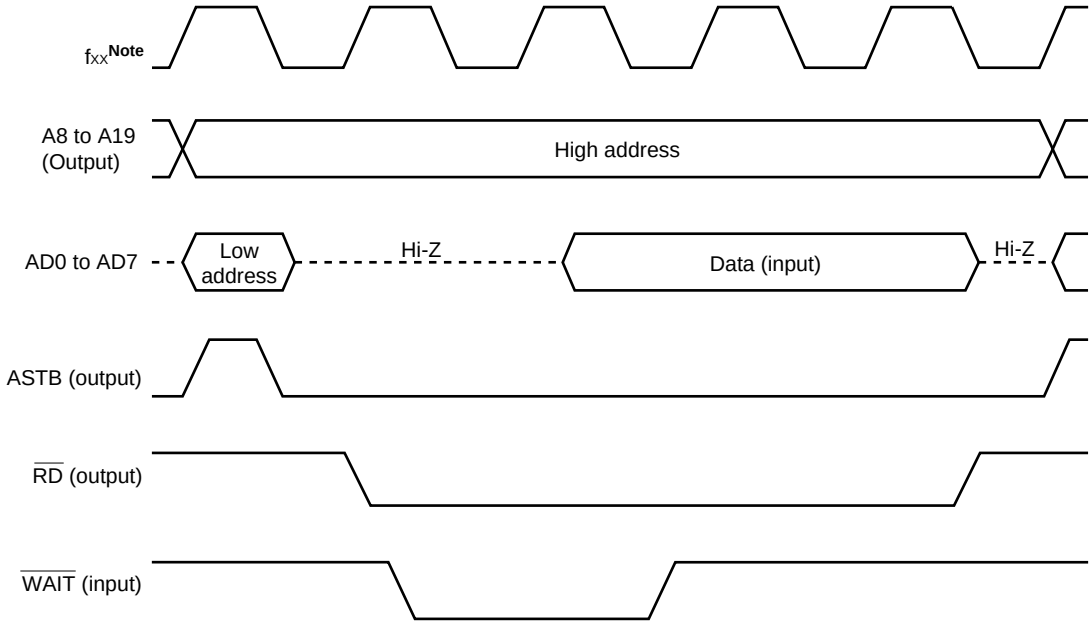
(c) Setting 2 wait cycles (PW01 = 1, PW00 = 1, 0)



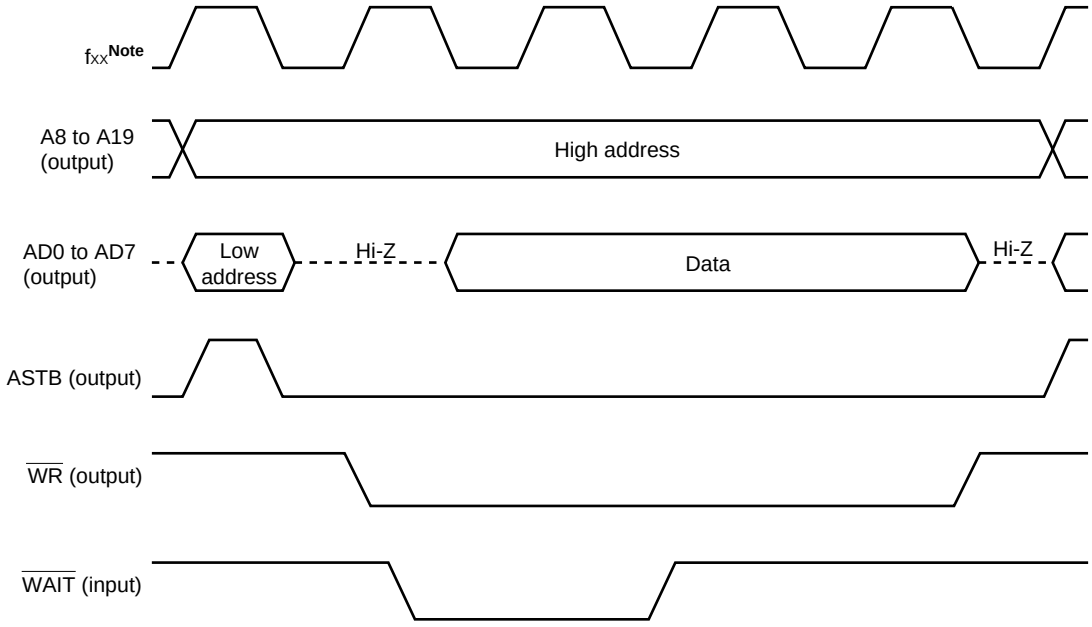
Note fXX: Main system clock frequency. This signal is only in the μPD784225.

Figure 23-12. Timing by External Wait Signal

(a) Read Timing (PW01 = 1, PW00 = 1, 1)



(b) Write timing (PW01 = 1, PW00 = 1, 1)



Note fXX: Main system clock frequency. This signal is only in the μPD784225.

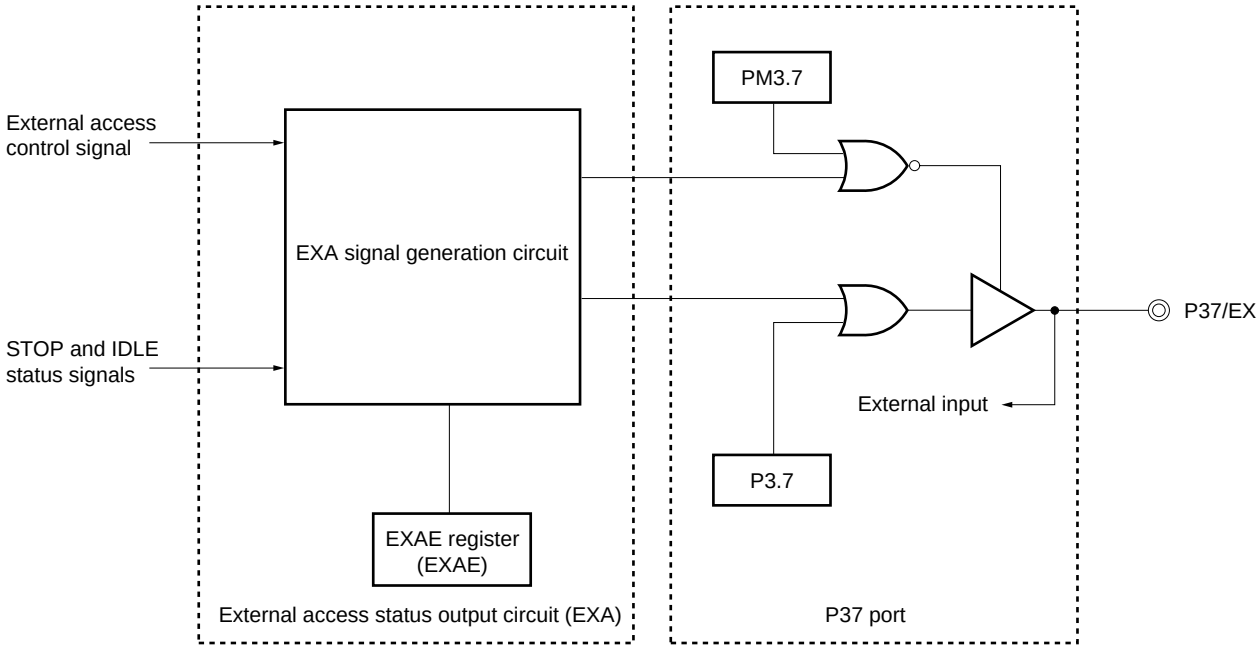
23.6 External Access Status Output Function

23.6.1 Summary

The external access status signal is output from the P37/EXA pin. This signal is output at the moment of external access when use of the external bus interface function has been enabled. This signal detected the external access status of other devices connected to the external bus, prohibits other devices from outputting data to the external bus, and enables receipt.

23.6.2 Configuration of the external access status output function

Figure 23-13. Configuration of the External Access Status Output Function



23.6.3 External access status enable register

The external access status enable register (EXAE) controls the EXA signal output indicated during external access.

EXAE are set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets to 00H.

Figure 23-14. External Access Status Enable Register (EXAE) Format

Address: 0FF8DH After Reset: 00H

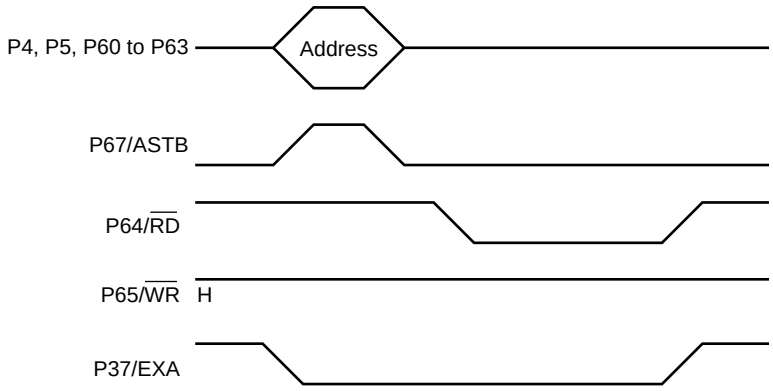
Symbol	7	6	5	4	3	2	1	0
EXAE	0	0	0	0	0	0	0	EXAE0

EXAE	P37 Function
0	Port function
1	External access status function

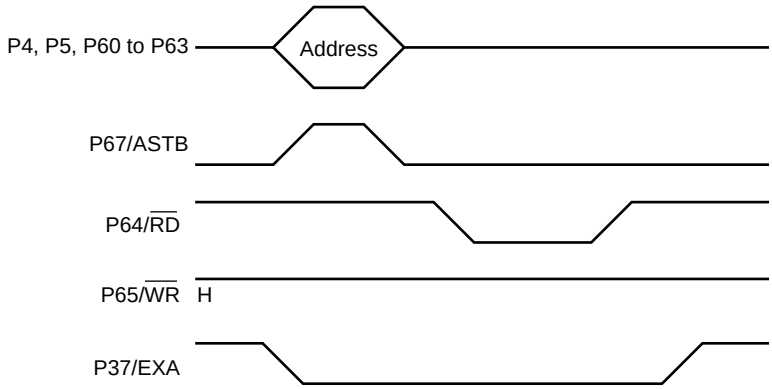
23.6.4 External access status signal timing

A timing chart for the P37/EXA and external bus interface pin is shown below. The EXA signal is set at low active, and indicates the external access status when at “0”.

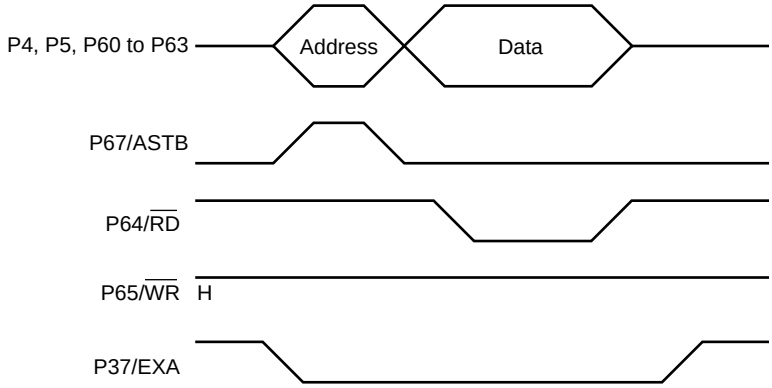
(a) Data fetch timing



(b) Data read timing



(c) Data write timing



23.6.5 EXA pin status during each mode

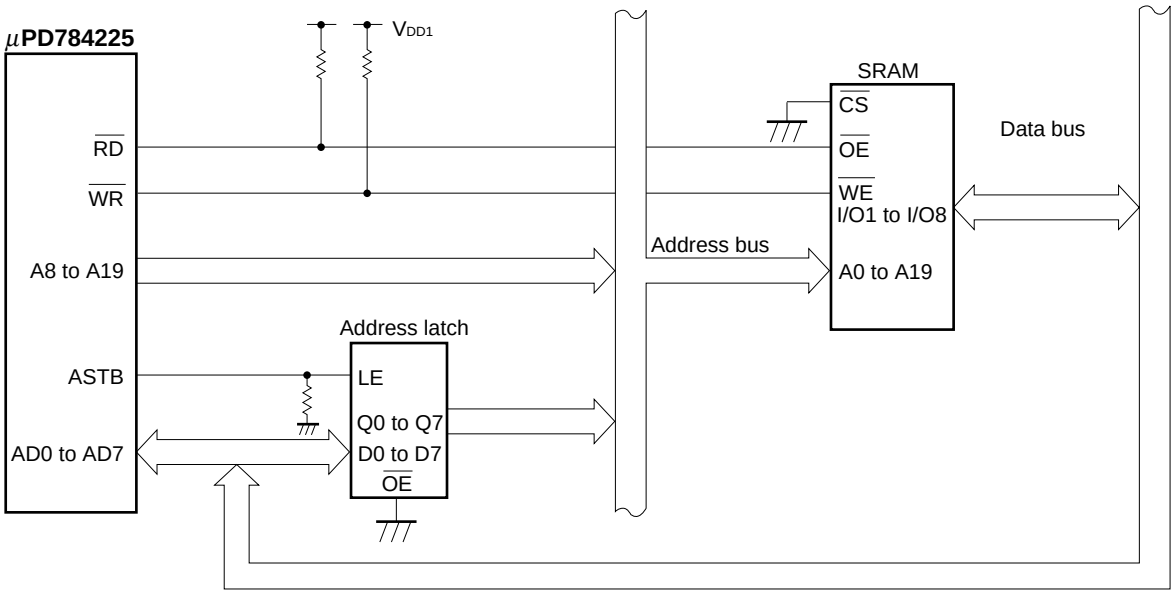
P37/EXA pin status during each mode is shown in Table 23-3.

Table 23-3. P37/EXA Pin Status During Each Mode

Mode	P37/EXA Functions
When reset	Hi-Z
When reset	Hi-Z immediate after the reset is canceled (input mode, PM3.7 = 1) Port operations when EXAE = 00H with PM3.7 and P3.7 = 0 EXA signal output enabled when EXAE = 01H with PM3.7 and P3.7 = 0
When in the HALT mode	Stand-by
When in the IDLE mode	Hi-Z
When in the STOP mode	Hi-Z

23.7 External Memory Connection Example

Figure 23-15. Example of Local Bus Interface (Multiplexed Bus)



CHAPTER 24 STANDBY FUNCTION

24.1 Structure and Function

The μPD784225 has a standby function that can decrease the system's power consumption. The standby function has the following six modes.

Table 24-1. Standby Function Modes

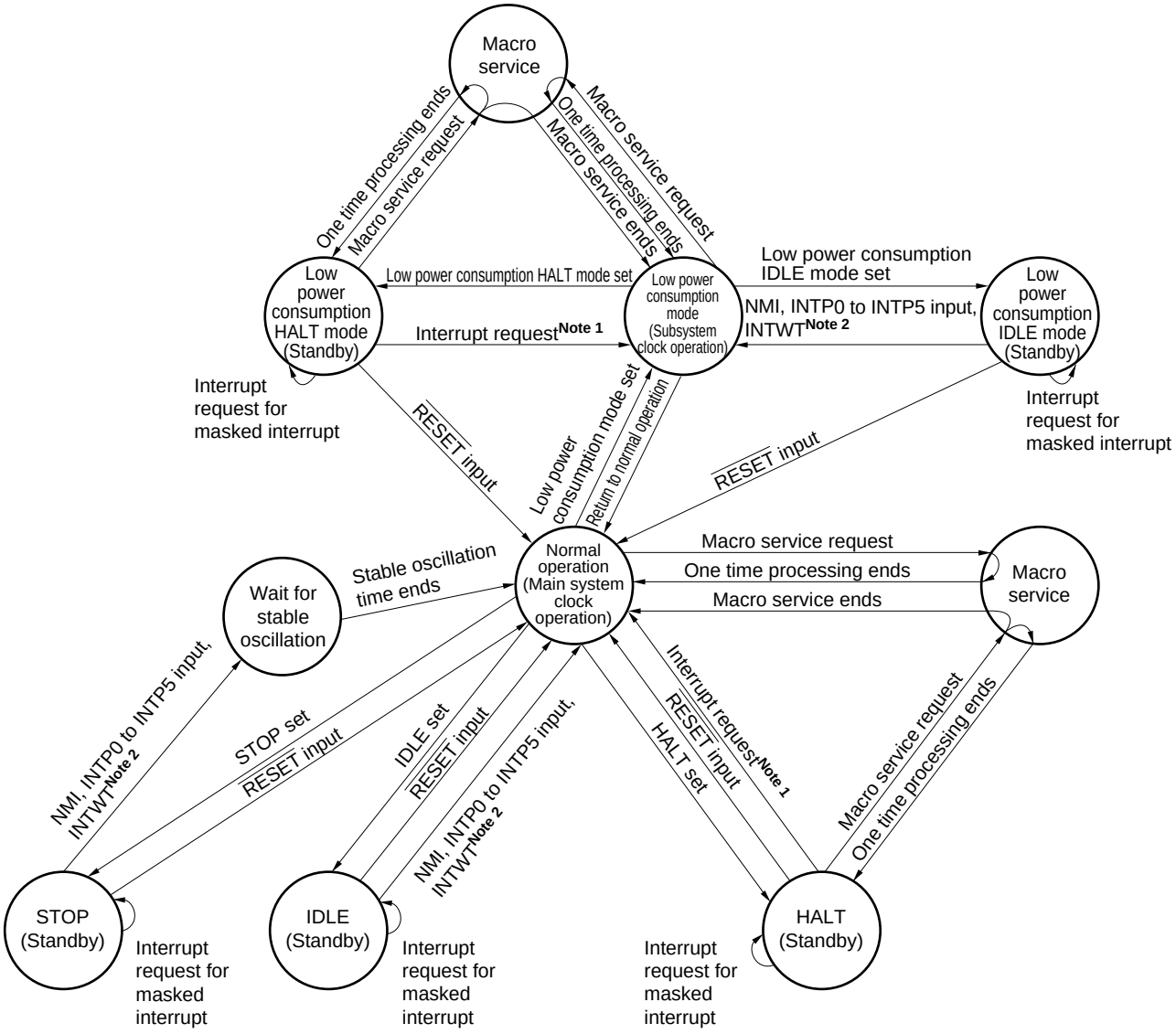
HALT mode	Stops the CPU operating clock. The average power consumption can be reduced by intermittent operation during normal operation.
STOP mode	Stops the main system clock. All of the operations in the chip are stopped, and the extremely low power consumption state of only a leakage current is entered.
IDLE mode	In this mode, the oscillation circuit continues operating while the rest of the system stops. Normal program operation can return to power consumption near that of the STOP mode and for the same time as the HALT mode.
Low power consumption mode	The subsystem clock is used as the system clock, and the main system clock is stopped. Since reduced power consumption is designed, the CPU can operate with the subsystem clock.
Low power consumption HALT mode	The CPU operating clock is stopped by the standby function in the low power consumption mode. Power consumption for the entire system is decreased.
Low power consumption IDLE mode	The oscillation circuit continues operating while the rest of the system is stopped by the standby function in the low power consumption mode. Power consumption for the entire system is decreased.

These modes are programmable.

Macro serivce can be started from the HALT mode and the low power consumption HALT mode. After macro service execution, the device is returned to the HALT mode.

Figure 24-1 shows the standby function state transitions.

Figure 24-1. Standby Function State Transitions



Notes 1. Only unmasked interrupt requests

2. Only unmasked INTP0 to INTP5, watch timer interrupt (INTWT)

Remark NMI is only valid with external input. The watchdog timer cannot be used for the release of Standby (HALT mode/STOP mode/IDLE mode.)

24.2 Control Registers

(1) Standby control register (STBC)

The STBC register sets the STOP mode and selects the internal system clock.

To prevent the standby mode from accidentally being entered due to a runaway program, this register can only be written by a special instruction. This special instruction is MOV STBC, #byte which has a special code structure (4 bytes). This register can only be written when the third and fourth byte op codes are mutual 1's complements. If the third and fourth byte op codes are not mutual 1's complements, the register is not written and an operand error interrupt is generated. In this case, the return address that is saved on the stack is the address of the instruction that caused the error. Therefore, the address that caused the error can be determined from the return address saved on the stack.

If the RETB instruction is used to simply return from an operand error, an infinite loop occurs.

Since an operand error interrupt is generated only when the program runs wild (only the correct instruction is generated when MOV STBC, #byte is specified in RA78K4 NEC assembler), make the program initialize the system.

Other write instructions (i.e., MOV STBC, A; STBC, #byte; SET1 STBC.7) are ignored and nothing happens. In other words, STBC is not written, and an interrupt, such as an operand error interrupt, is not generated.

STBC can always be read by a data transfer instruction.

$\overline{\text{RESET}}$ input sets STBC to 30H.

Figure 24-2 shows the STBC format.

Figure 24-2. Standby Control Register (STBC) Format

Address: 0FFC0H After Reset: 30H R/W

Symbol	7	6	5	4	3	2	1	0
STBC	SBK	CK2	CK1	CK0	0	MCK	STP	HLT

SBK	Subsystem Clock Oscillation Control
0	Oscillation circuit operation. (Use internal feedback resistors.)
1	Oscillation circuit stop. (Do not use internal feedback resistors.)

CK2	CK1	CK0	CPU Clock Selection
0	0	0	f_{XX}
0	0	1	$f_{XX}/2$
0	1	0	$f_{XX}/4$
0	1	1	$f_{XX}/8$
1	×	×	f_{XT}

MCK	Main System Clock Oscillation Control
0	Oscillation circuit operation. (Use internal feedback resistors.)
1	Oscillation circuit stop. (Do not use internal feedback resistors.)

STP	HLT	Operation Setting Flag
0	0	Normal operating mode
0	1	HALT mode (automatically cleared when the HALT mode is released)
1	0	STOP mode (automatically cleared when the STOP mode is released)
1	1	IDLE mode (automatically cleared when the IDLE mode is released)

Cautions 1. If the STOP mode is used when an external clock is input, set the STOP mode after setting bit EXTC in the oscillation stable time setting register (OSTS) to one. Using the STOP mode in the state where bit EXTC of OSTS is cleared (0) while the external clock is input may destroy the μPD784225 or reduce reliability. When the EXTC bit of OSTS is set to one, always input at pin X2 the clock that has the inverse phase of the clock input at pin X1.

2. Execute three NOP instructions after the standby instruction (after releasing the standby). If this is not done, when the execution of a standby instruction competes with an interrupt request, the standby instruction is not executed, and interrupts are accepted after executing multiple instructions that follow a standby instruction. The instruction that is executed before accepting the interrupt starts executing within a maximum of six clocks after the standby instruction is executed.

Example **MOV STBC, #byte**
NOP
NOP
NOP
:

3. When CK2 = 0, even if MCK = 1, the oscillation of the main system clock does not stop (refer to 4.5.1 Main system clock operations).

Remarks 1. f_{XX}: Main system clock frequency (f_X or $f_X/2$)
f_X : Main system clock oscillation frequency
f_{XT}: Subsystem clock oscillation frequency

2. × : don't care

(2) Clock status register (PCS)

PCS is a read-only 8-bit register that shows the operating state of the CPU clock. When bits 2, and 4 to 7 in PCS are read, the corresponding bits in the standby control register (STBC) can be read.

PCS is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PCS to 32H.

Figure 24-3. Clock Status Register (PCS) Format

Address: 0FFCEH After Reset: 32H R

Symbol	7	6	5	4	3	2	1	0
PCS	SBK	CK2	CK1	CK0	0	MCK	1	CST

SBK	Feedback Resistor State for Subsystem Clock
0	Use internal feedback resistors.
1	Use internal feedback resistors.

CK2	CK1	CK0	CPU Clock Operating Frequency
0	0	0	f_{XX}
0	0	1	$f_{XX}/2$
0	1	0	$f_{XX}/4$
0	1	1	$f_{XX}/8$
1	×	×	f_{XT}

MCK	Main System Clock Oscillation Control
0	Oscillation circuit operation.
1	Oscillation circuit stop.

CST	CPU Clock State
0	Main system clock operation
1	Subsystem clock operation

Remark ×: don't care

(3) Oscillation stable time setting register (OSTS)

The OSTS register sets the oscillation circuit operation and the oscillation stabilization time when the STOP mode is released. Whether a crystal/ceramic oscillator or an external clock will be used is set in the EXTC bit of OSTS. If only the EXTC bit is set to one, the STOP mode can also be set when the external clock is input.

Bits OSTS0 to OSTS2 in OSTS select the oscillation stabilization time when the STOP mode is released. Generally, select an oscillation stabilization time of at least 40 ms when using a crystal oscillator and at least 4 ms when using a ceramic oscillator.

The time until the stablilization oscillation is affected by the crystal/ceramic oscillator that is used and the capacitance of the connected capacitor. Therefore, if you want a short oscillation stabilization time, consult the manufacturer of the crystal/ceramic oscillator.

OSTS can be set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets OSTS to 00H.

Figure 24-4 shows the OSTS format.

Figure 24-4. Oscillation Stabilization Time Setting Register (OSTS) Format

Address: 0FFCFH After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
OSTS	EXTC	0	0	0	0	OSTS2	OSTS1	OSTS0

EXTC	External Clock Selection
0	Use crystal/ceramic oscillation
1	Use an external clock

EXTC	OSTS2	OSTS1	OSTS0	Oscillation Stabilization Time Selection
0	0	0	0	$2^{19}/f_{XX}$ (41.9 ms)
0	0	0	1	$2^{18}/f_{XX}$ (21.0 ms)
0	0	1	0	$2^{17}/f_{XX}$ (10.5 ms)
0	0	1	1	$2^{16}/f_{XX}$ (5.2 ms)
0	1	0	0	$2^{15}/f_{XX}$ (2.6 ms)
0	1	0	1	$2^{14}/f_{XX}$ (1.3 ms)
0	1	1	0	$2^{13}/f_{XX}$ (655 μs)
0	1	1	1	$2^{12}/f_{XX}$ (328 μs)
1	×	×	×	$512/f_{XX}$ (41.0 μs)

Cautions 1. When using crystal/ceramic oscillation, always clear the EXTC bit to zero. When the EXTC bit is set to one, oscillation stops.

2. If the STOP mode is used when an external clock is input, always set the EXTC bit to one and then set the STOP mode. Using the STOP mode in the state where the EXTC bit is cleared (0) while the external clock is input may destroy μPD784225 or reduce reliability.

3. When the EXTC bit is set to one when an external clock is input, input to pin X2 a clock that has the inverse phase of the clock input to pin X1. If the EXTC bit is set to one, μPD784225 only operates with the clock that is input to the X2 pin.

Remarks 1. Figures in parentheses apply to operation with f_{XX} = 12.5 MHz.

2. ×: don't care

24.3 HALT Mode

24.3.1 Settings and operating states of HALT mode

The HALT mode is set by setting the HLT bit in standby control register (STBC) to 1.

STBC can be written in with 8-bit data by a special instruction. Therefore, the HALT mode is specified by the MOV STBC, #byte instruction.

When enable interrupts is set (IE flag in PSW is set to one), specify three NOP instructions after the HALT mode setting instruction (after the HALT mode is released). If this is not done, after the HALT mode is released, multiple instructions may execute before interrupts are accepted. Unfortunately, the order relationship between the interrupt process and instruction execution changes. Since problems caused by the changes in the execution order are prevented, the measures described earlier are required.

The system clock when setting can be set to either the main system clock or the subsystem clock.

The operating states in the HALT mode are described next.

Table 24-2. Operating States in the HALT Mode

HALT Mode Setting / Item		HALT Instruction Mode Setting During Main System Clock Operation		HALT Instruction Mode Setting During Subsystem Clock Operation	
		No subsystem clock **Note 1**	Subsystem clock **Note 2**	When the main system clock continues oscillating	When the main system clock stops oscillating
Clock oscillation circuit		Both the main system clock and subsystem clock can oscillate. The clock supply to the CPU stops.			
CPU		Operation disabled			
Port (output latch)		Holds the state before setting the HALT mode.			
16-bit timer/counter		Operation enabled			Operational when the watch timer output is selected as the count clock (Select f_{XT} as the count clock of the watch timer.)
8-bit timer/counters 1, 2		Operation enabled			Operational when TI1 and TI2 are selected as the count clocks
8-bit timer/counters 5, 6		Operation enabled			Operational when TI5 and TI6 are selected as the count clocks
Watch timer		Operational when $f_{XX}/2^8$ is selected as the count clock	Operation enabled		Operational when f_{XT} is selected as the count clock
Watchdog timer		Operation enabled		Operation disabled	
A/D converter		Operation enabled			Operation disabled
D/A converter		Operation enabled			
Real-time output port		Operation enabled			
Serial interface		Operation enabled			Operational during an external SCK.
External interrupt	INTP0 to INTP5	Operation enabled			
Bus lines during external expansion	AD0 to AD7	High impedance			
	A0 to A19	Holds the state before the HALT mode is set			
	ASTB	Low level			
	$\overline{WR}$, $\overline{RD}$	High level			
	$\overline{WAIT}$	High impedance			

Notes 1. This includes not supplying the external clock.

2. This includes supplying the external clock.

24.3.2 Releasing HALT mode

The HALT mode can be released by the following three sources.

- Non-maskable interrupt request. (Only possible for NMI pin input.)
- Maskable interrupt request (vectored interrupt, context switching, macro service)
- $\overline{\text{RESET}}$ input

Table 24-3 lists the release source and describes the operation after release. Operations following the canceling of the HALT mode are also shown in Figure 24-5.

Table 24-3. HALT Mode Release and Operate After Release

Release Source	MK[Note 1]	IE[Note 2]	State During Release	Operation After Release
$\overline{\text{RESET}}$ input	×	×	–	Normal reset operation
NMI pin input	×	×	• None while executing a non-maskable interrupt service program • Executing a low-priority non-maskable interrupt service program	Accepts interrupt requests
			• Executing the service program for the same request • Executing a high-priority non-maskable interrupt service program	The instruction following the MOV STBC, #byte instruction is executed. (The interrupt request that released the HALT mode is saved[Note 3].)
Maskable interrupt request (except for a macro service request)	0	1	• None while executing an interrupt service program • Executing a low-priority maskable interrupt service program • The PRSL bit[Note 4] is cleared to zero while executing an interrupt service program at priority level 3.	Accepts interrupt requests
			• Executing a maskable interrupt service program with the same priority (This excludes executing an interrupt service program in priority level 3 when the PRSL bit[Note 4] is cleared to zero.) • Executing a high-priority interrupt service program	The instruction following MOV STBC, #byte is executed. (The interrupt request that released the HALT mode is saved[Note 3].)
	0	0	–	
	1	×	–	Holds the HALT mode
Macro service request	0	×	–	Macro service process execution End condition is not satisfied → End HALT mode condition is satisfied again → When VCIE[Note 5] = 1: HALT mode again When VCIE[Note 5] = 0: Same as a release by the maskable interrupt request
	1	×	–	Holds the HALT mode

Notes 1. Interrupt mask bit in each interrupt request source

2. Interrupt enable flag in the program status word (PSW)

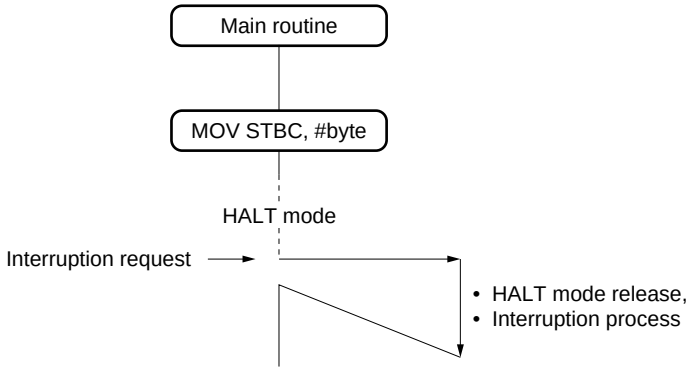
3. The held interrupt request is accepted when acceptance is enabled.

4. Bit in the interrupt mode control register (IMC)

5. Bit in the macro service mode register of the macro service control word that is in each macro service request source

Figure 24-5. Operations After HALT Mode Release (1/4)

(1) Interrupt after HALT mode.



(2) Reset after HALT mode.

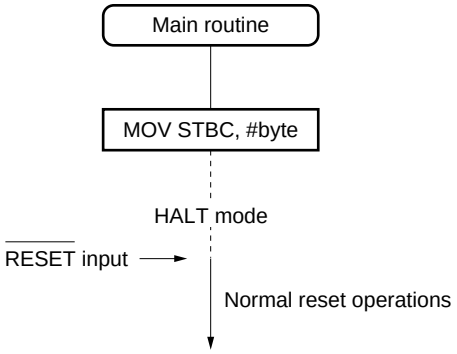
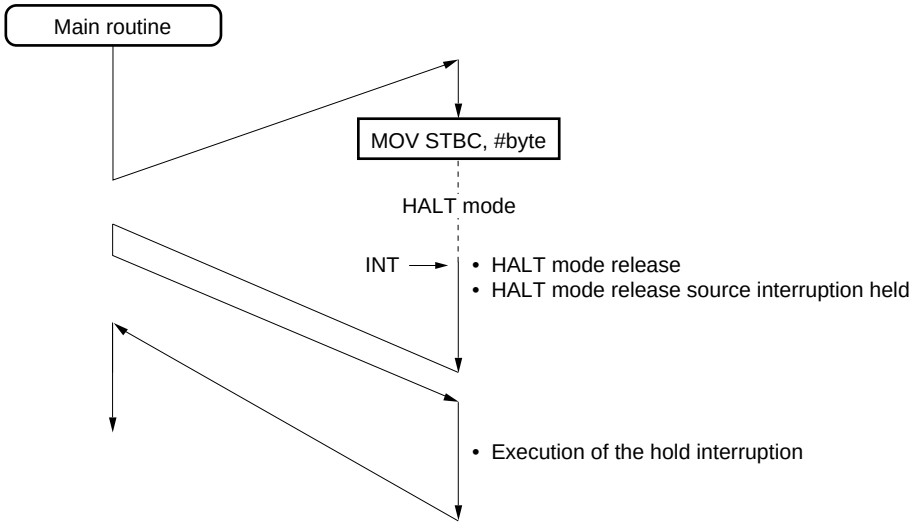


Figure 24-5. Operations After HALT Mode Release (2/4)

(3) HALT mode during interrupt processing routine whose priority is higher than or equal to release source interrupt.



(4) HALT mode during interrupt processing routine whose priority is lower than release source interrupt.

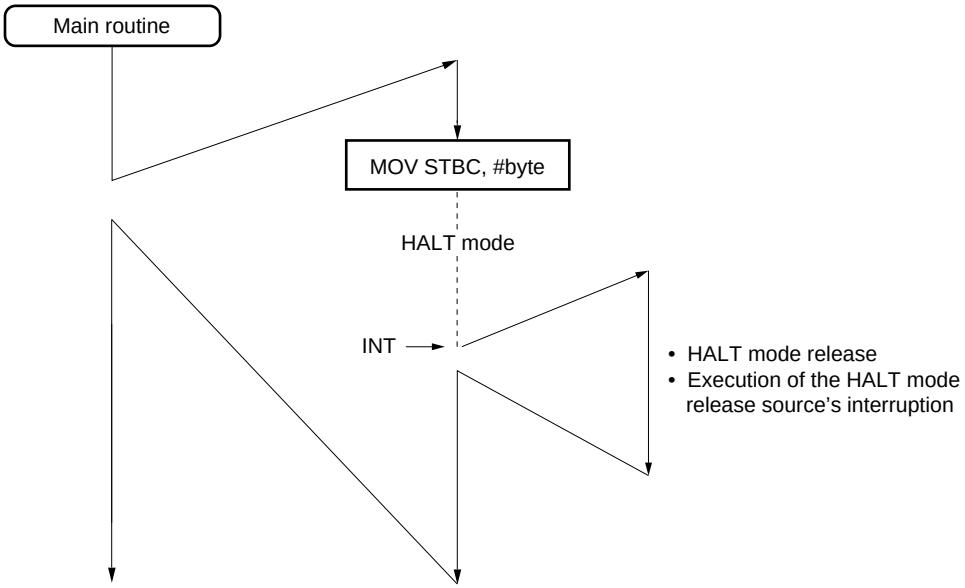
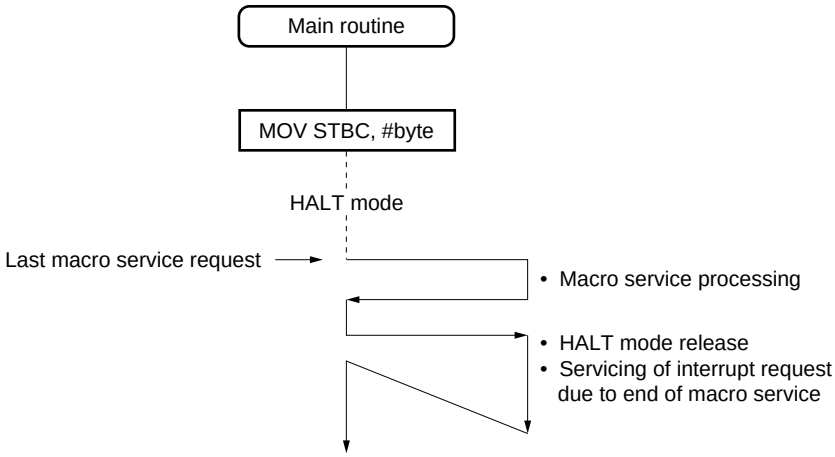


Figure 24-5. Operations After HALT Mode Release (3/4)

(5) Macro service request during HALT mode.

(a) Immediately after macro service end condition is satisfied, interrupt request is issued, (VCIE=0).



(b) Macro service end condition is not satisfied, or after macro service end condition is satisfied, interrupt request is not issued.

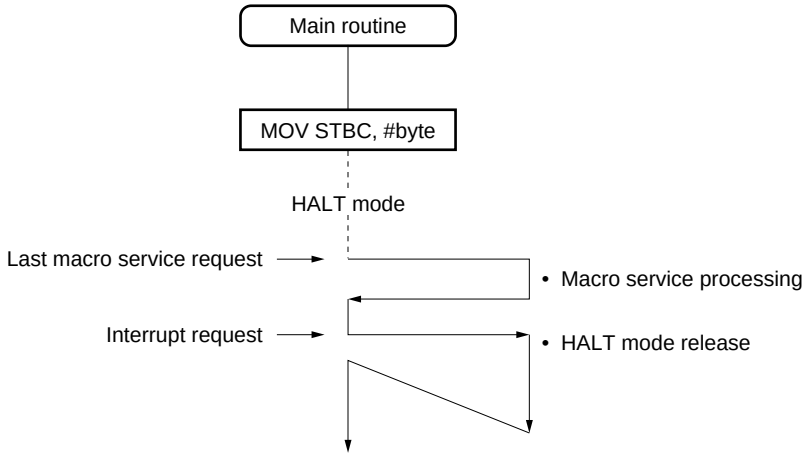
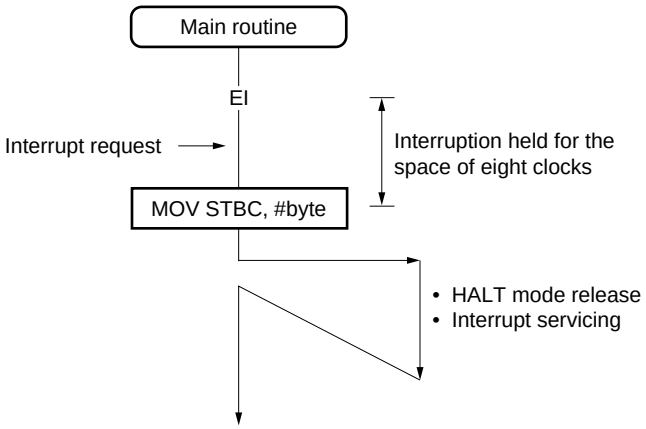
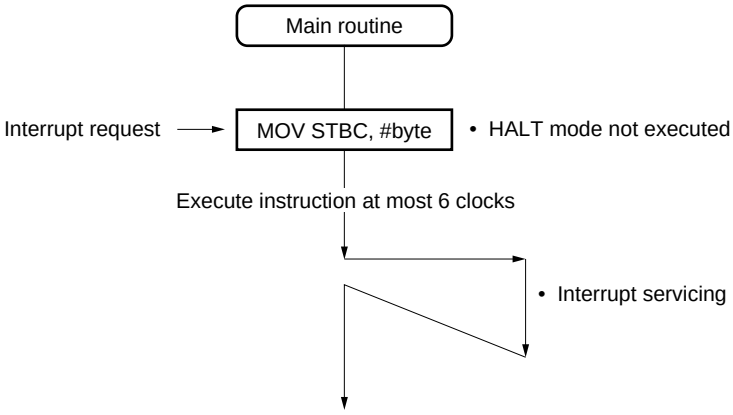


Figure 24-5. Operations After HALT Mode Release (4/4)

(6) HALT mode which the interrupt is held, which is enabled in an instruction that interrupt requests are temporarily held.



(7) Contention between HALT instruction and interrupt.



(1) Released by a nonmaskable interrupt

When a nonmaskable interrupt is generated, the halt mode is released regardless of the enable state (EI) and disable state (DI) for interrupt acceptance.

If the nonmaskable interrupt that released the HALT mode can be accepted when released from the HALT mode, that nonmaskable interrupt is accepted, and execution branches to the service program. If it cannot be accepted, the instruction following the instruction that set the HALT mode (MOV STBC, #byte instruction) is executed. The nonmaskable interrupt that released the HALT mode is accepted when acceptance is possible. For details about accepting nonmaskable interrupts, refer to **22.6 Non-maskable Interrupt Acknowledgment Operation**.

★ **Caution The HALT mode cannot be released with the watchdog timer.**

(2) Released by a maskable interrupt request

The HALT mode released by a maskable interrupt request can only be released by an interrupt where the interrupt mask flag is 0.

If an interrupt can be accepted when the halt mode is released and the interrupt request enable flag (IE) is set to one, execution branches to the interrupt service program. If the IE flag is cleared to zero when acceptance is not possible, execution restarts from the next instruction that sets the HALT mode. For details about interrupt acceptance, refer to **22.7 Maskable Interrupt Acknowledgment Operation**.

A macro service temporarily releases the HALT mode, performs the one-time processing, and returns again to the HALT mode. If the macro service is only specified several times, the HALT mode is released when the VCIE bit in the macro service mode register in the macro service control word is cleared to 0.

The operation after this release is identical to the release by the maskable interrupt described earlier. Also when the VCIE bit is set to 1, the HALT mode is entered again, and the HALT mode is released by the next interrupt request.

Table 24-4. Releasing HALT Mode by Maskable Interrupt Request

Release Source	MK[Note 1]	IE[Note 2]	State During Release	Operation After Release
Maskable interrupt request (except for a macro service request)	0	1	• None while executing an interrupt service program • Executing a low-priority maskable interrupt service program • The PRSL bit[Note 4] is cleared to zero while executing an interrupt service program at priority level 3.	Accepts interrupt requests
			• Executing a maskable interrupt service program with the same priority (This excludes executing an interrupt service program in priority level 3 when the PRSL bit[Note 4] is cleared to zero.) • Executing a high-priority interrupt service program	The instruction following the MOV STBC, #byte instruction is executed. (The interrupt request that released the HALT mode is saved[Note 3].)
	0	0	–	
	1	×	–	Holds the HALT mode
Macro service request	0	×	–	Macro service process execution End condition is not satisfied → End HALT mode condition is satisfied again → When VCIE[Note 5] = 1: HALT mode again When VCIE[Note 5] = 0: Same as a release by a maskable interrupt request
	1	×	–	Holds the HALT mode

Notes 1. Interrupt mask bit in each interrupt request source

2. Interrupt enable flag in the program status word (PSW)

3. The held interrupt request is accepted when acceptance is possible.

4. Bit in the interrupt mode control register (IMC)

5. Bit in the macro service mode register of the macro service control word that is in each macro service request source

(3) Released by $\overline{\text{RESET}}$ input

After branching to the reset vector address as in a normal reset, the program executes. However, the contents of the internal RAM hold the value before the HALT mode was set.

24.4 STOP Mode

24.4.1 Settings and operating states of STOP mode

The STOP mode is set by setting the STP bit in the standby control register (STBC) to one.

STBC can be written with 8-bit data by a special instruction. Therefore, the STOP mode is set by the MOV STBC, #byte instruction.

When enable interrupts is set (IE flag in PSW is set to one), specify three NOP instructions after the STOP mode setting instruction (after the STOP mode is released). If this is not done, after the STOP mode is released, multiple instructions can be executed before interrupts are accepted. Unfortunately, the order relationship between the interrupt process and instruction execution changes. Since the problems caused by changes in the execution order are prevented, the measures described earlier are required.

The system clock during setting can only be set to the main system clock.

Caution Since an interrupt request signal is used when releasing the standby mode, when there is an interrupt source that sets the interrupt request flag or resets the interrupt mask flag, even though the stanby mode is entered, it is immediately released. Therefore, in the STOP mode, the HALT mode is entered immediately after the HALT instruction is executed, and the operating mode returns after waiting only the time set in the oscillation stable time selection register (OSTS).

Next, the operating states during the STOP mode are described.

Table 24-5. Operating States in STOP Mode

STOP Mode Setting / Item		When there is a main system clock	When there is no subsystem clock
Clock generation circuit		Only main system clock stops oscillating.	
CPU		Operation disabled	
Port (output latch)		Holds the state before the STOP mode was set.	
16-bit timer/counter		Operational when the watch timer output is selected as the count clock (Select f_{XT} as the count clock of the watch timer)	Operation disabled
8-bit timer/counters 1, 2		Operational only when TI1 and TI2 are selected as the count clocks	
8-bit timer/counters 5, 6		Operational only when TI5 and TI6 are selected as the count clocks	
Watch timer		Operational only when f_{XT} is selected as the count clock	Operation enabled
Watchdog timer		Operation disabled	
A/D converter		Operation disabled	
D/A converter		Operation enabled	
Real-time output port		Operational when an external trigger is used or TI1 and TI2 are selected as the count clocks of the 8-bit timer/counters 1 and 2	
Serial interface	Except I^2C bus mode	Operational only when an external input clock is selected as the serial clock	
	I^2C bus mode	Operation disabled	
External interrupt	INTP0 to INTP5	Operation enabled	
Bus lines during external expansion	AD0 to AD7	High impedance	
	A0 to A19	Holds the state before the STOP mode is set	
	ASTB	Low level	
	$\overline{WR}$, $\overline{RD}$	High level	
	$\overline{WAIT}$	High impedance	

Caution In the STOP mode, only external interrupts (INTP0 to INTP5) and watch timer interrupts (INTWT) can release the STOP mode and be acknowledged are pended, and acknowledged after the STOP mode has been released through NMI input, INTP0 to INTP5 input or INTWT.

24.4.2 Releasing STOP mode

The STOP mode is released by NMI input, INTP0 to INTP5 input, watch timer interrupt (INTWT), or $\overline{\text{RESET}}$ input.

Outlines of the release sources and operations following release are shown in Table 24-6. Operations following release of the STOP mode are also shown in Figure 24-6.

Table 24-6. Releasing STOP Mode and Operation After Release

Release Source	MK[Note 1]	ISM[Note 2]	IE[Note 3]	State During Release	Operation After Release
$\overline{\text{RESET}}$ input	×	×	×	–	Normal reset operation
NMI pin input	×	×	×	• None while executing a non-maskable interrupt service program • Executing a low-priority non-maskable interrupt service program	Accepts interrupt requests
				• Executing the service program for the NMI pin input • Executing a high-priority non-maskable interrupt service program	The instruction following the MOV STBC, #byte instruction is executed. (The interrupt request that released the STOP mode is saved[Note 4].)
INTP0 to INTP5 pin input, watch timer interrupt	0	0	1	• None while executing an interrupt service program • Executing a low-priority maskable interrupt service program • The PRSL bit[Note 5] is cleared to zero while an interrupt service program at priority level 3 is executing.	Accepts interrupt requests
				• Executing a maskable interrupt service program with the same priority (This excludes executing an interrupt service program in priority level 3 when the PRSL bit[Note 5] is cleared to zero.) • Executing a high-priority interrupt service program	The instruction following MOV STBC, #byte instruction is executed. (The interrupt request that released the STOP mode is saved[Note 4].)
	0	0	0	–	
	1	0	×	–	Holds the STOP mode
	×	1	×		

Notes 1. Interrupt mask bit in each interrupt request source

2. Macro service enable flag that is in each interrupt request source

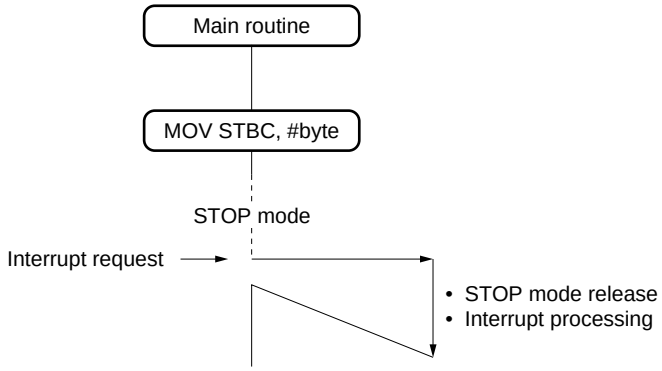
3. Interrupt enable flag in the program status word (PSW)

4. The saved interrupt request is accepted when acceptance is possible.

5. Bit in the interrupt mode control register (IMC)

Figure 24-6. Operations After the STOP Mode has been Release (1/2)

(1) Interrupt after STOP mode.



(2) Reset after STOP mode.

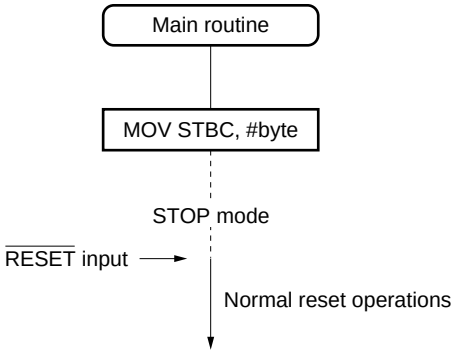
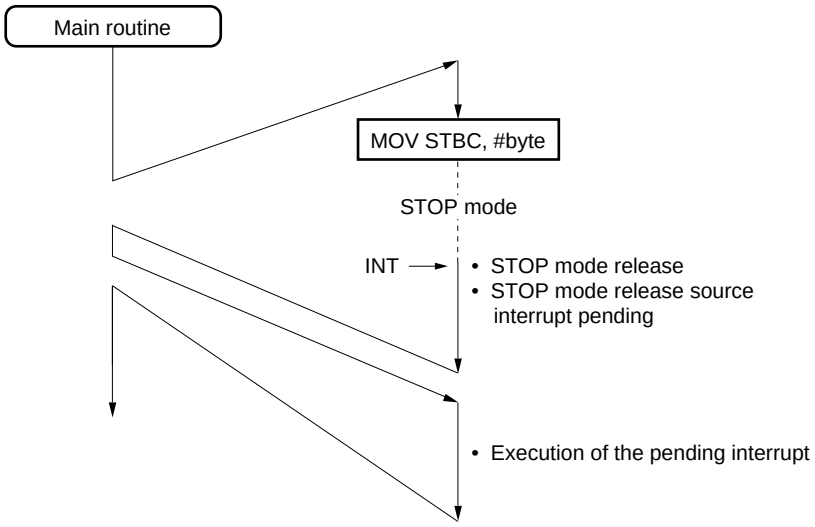
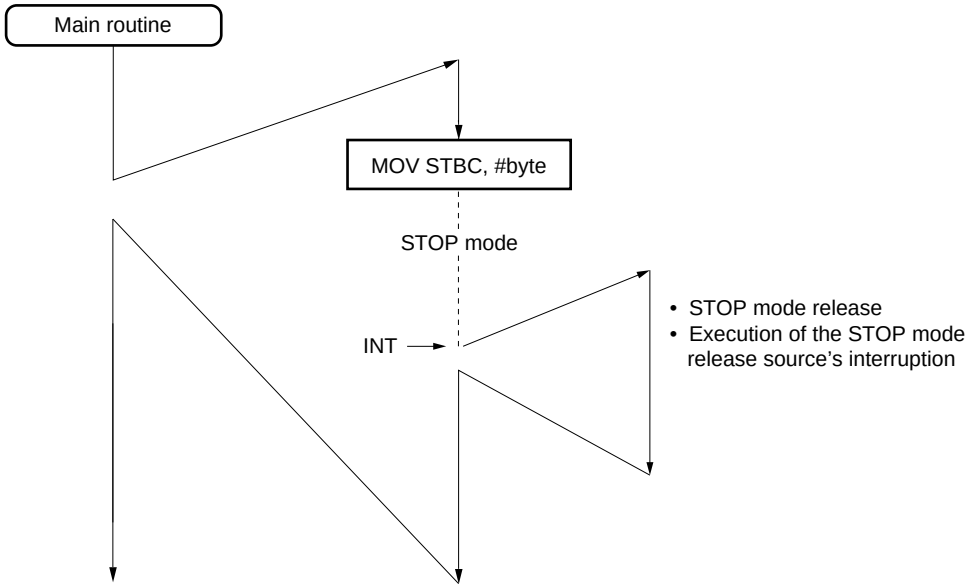


Figure 24-6. Operations After the STOP Mode has been Release (2/2)

(3) STOP mode during interrupt processing routine whose priority is lower than release source interrupt.



(4) STOP mode during interrupt processing routine whose priority is lower than release source interrupt.



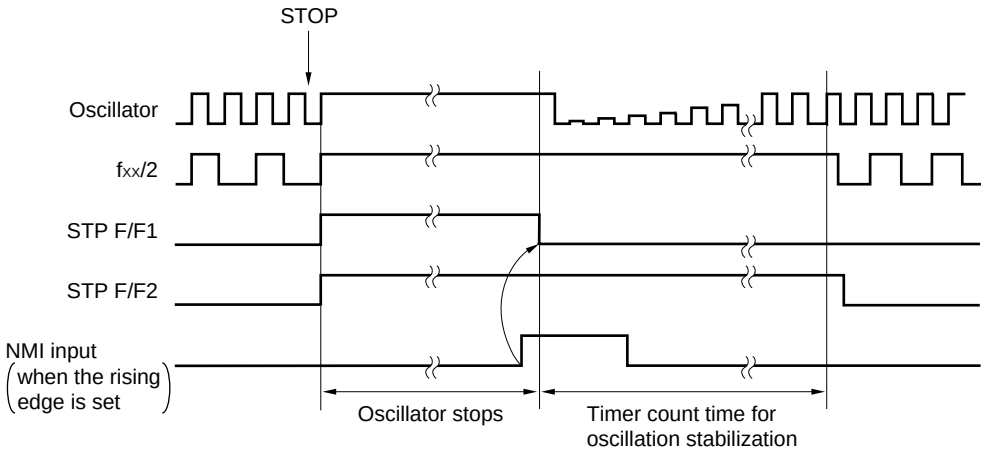
(1) Releasing the STOP mode by NMI input

When the valid edge specified in the external interrupt edge enable registers (EGP0, EGN0) is input by the NMI input, the oscillator starts oscillating again. Then the STOP mode is released after the oscillation stabilization time set in the oscillation stabilization time setting register (OSTS) elapses.

When the STOP mode is released and non-maskable interrupts from the NMI pin input can be accepted, execution branches to the NMI interrupt service program. If acceptance is not possible (such as when set in the STOP mode in the NMI interrupt service program), execution starts again from the instruction following the instruction that set the STOP mode. When acceptance is enabled, execution branches to the NMI interrupt service program (by executing the RETI instruction).

For details about NMI interrupt acceptance, refer to **22.6 Non-maskable Interrupt Acknowledgment Operation**.

Figure 24-7. Releasing STOP Mode by NMI Input



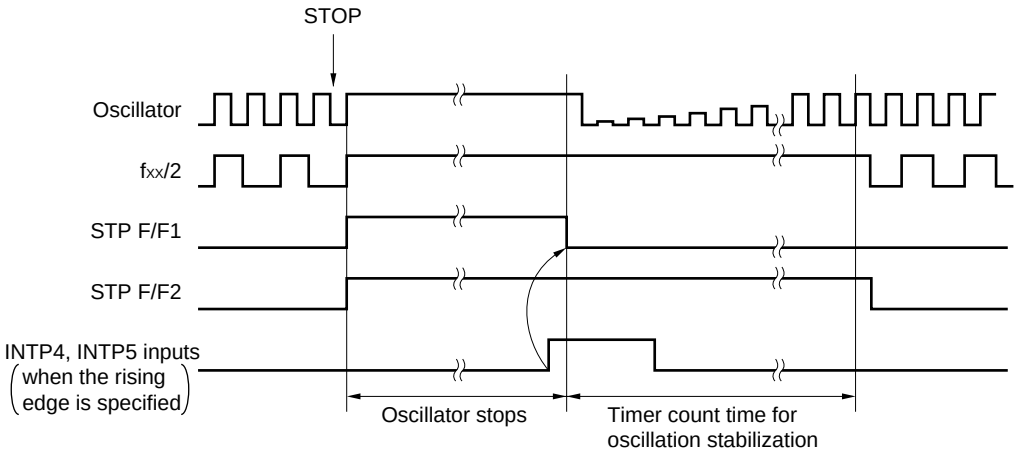
(2) Releasing the STOP mode by INTP0 to INTP5 input and watch timer interrupt

If interrupt masking is released through INTP0 to INTP5 input and macro service is disabled, the oscillator restarts oscillating when a valid edge specified in the external interrupt edge enable registers (EGP0, EGN0) is input to INTP0 to INTP5. At the same time, an overflow will occur with the watch timer and the IDLE mode will be canceled when the watch timer interrupt mask is released and macro services are prohibited. Then the STOP mode is released after the oscillation stabilization time specified in the oscillation stabilization time setting register (OSTS) elapses.

If interrupts can be accepted when released from the STOP mode and the interrupt enable flag (IE) is set to 1, execution branches to the interrupt service program. If the IE flag is cleared to zero when acceptance is not possible, execution starts again from the instruction following the instruction that set the STOP mode.

For details on interrupt acceptance, refer to **22.7 Maskable Interrupt Acknowledgment Operation**.

Figure 24-8. Example of Releasing STOP Mode by INTP4 and INTP5 Inputs



(3) Releasing the STOP mode by $\overline{\text{RESET}}$ input

When $\overline{\text{RESET}}$ input falls from high to low and the reset condition is entered, the oscillator starts oscillating. Maintain the oscillation stabilization time for the $\overline{\text{RESET}}$ active period. Then, when the $\overline{\text{RESET}}$ rises, normal operation starts.

The difference from the normal reset operation is the data memory saves the contents before setting the STOP mode.

24.5 IDLE Mode

24.5.1 Settings and operating states of IDLE mode

The IDLE mode is set by setting both bits STP and HLT in the standby control register (STBC) to one.

STBC can only be written with 8-bit data by using a special instruction. Therefore, the IDLE mode is set by the MOV STBC, #byte instruction.

When enable interrupts is set (the IE flag in PSW is set to one), specify three NOP instructions after the IDLE mode setting instruction (after the IDLE mode is released). If this is not done, after the IDLE mode is released, multiple instructions can be executed before interrupts are accepted. Unfortunately, the order relationship between the interrupt processing and the instruction execution changes. To prevent the problems caused by the change in the execution order, the measures described earlier are required.

The system clock when setting can be set to either the main system clock or the subsystem clock.

The operating states in the IDLE mode are described next.

Table 24-7. Operating States in IDLE Mode

IDLE Mode Setting / Item		When there is a subsystem clock	When there is not a subsystem clock
Clock generation circuit		The oscillation circuits in both the main system clock and subsystem clock continue operating. The clock supply to both the CPU and peripherals is stopped.	
CPU		Operation disabled	
Port (output latch)		Saves the state before the IDLE mode was set	
16-bit timer/counter		Operational when the watch timer output is selected as the count clock (Select f_{XT} as the count clock of the watch timer.)	Operation disabled
8-bit timer/counters 1, 2		Operational only when TI1 and TI2 are selected as the count clocks	
8-bit timer/counters 5, 6		Operational only when TI5 and TI6 are selected as the count clocks	
Watch timer		Operational only when f_{XT} is selected as the count clock	Operation enabled
Watchdog timer		Operation disabled	
A/D converter		Operation disabled	
D/A converter		Operation enabled	
Real-time output port		Operational when an external trigger is used or TI1 and TI2 are selected as the count clocks of the 8-bit timer/counters 1 and 2	
Serial interface	Except I^2C bus mode	Operational only when an external input clock is selected as the serial clock	
	I^2C bus mode	Operation disabled	
External interrupt	INTP0 to INTP5	Operation enabled	
Bus lines during external expansion	AD0 to AD7	High impedance	
	A0 to A19	Holds the state before the IDLE mode is set	
	ASTB	Low level	
	$\overline{WR}$, $\overline{RD}$	High level	
	$\overline{WAIT}$	High impedance	

Caution In the IDLE mode, only external interrupts (INTP0 to INTP5) and watch timer interrupts (INTWT) can release the IDLE mode and be acknowledged as interrupt requests. All other interrupt requests are pended, and acknowledged after the IDLE mode has been released through NMI input, INTP0 to INTP5 input or INTWT.

24.5.2 Releasing IDLE mode

The IDLE mode is released by NMI input, INTP0 to INTP5 input, watch timer interrupt (INTWT), or $\overline{\text{RESET}}$ input.

Outlines of the release sources and operations following release are shown in Table 24-8. Operations following release of the IDLE mode are also shown in Figure 24-9.

Table 24-8. Releasing IDLE Mode and Operation After Release

Release Source	MK[Note 1]	ISM[Note 2]	IE[Note 3]	State During Release	Operation After Release
$\overline{\text{RESET}}$ input	×	×	×	–	Normal reset operation
NMI pin input	×	×	×	• None while executing a non-maskable interrupt service program • Executing a low-priority non-maskable interrupt service program	Accepts interrupt requests
				• Executing the service program for the NMI pin input • Executing a high-priority non-maskable interrupt service program	Executes the instruction following the MOV STBC, #byte instruction (The interrupt request that released the IDLE mode is saved[Note 4].)
INTP0 to INTP5 pin input, watch timer interrupt	0	0	1	• None while executing an interrupt service program • Executing a low-priority maskable interrupt service program • The PRSL bit[Note 5] is cleared to zero while executing an interrupt service program at priority level 3.	Accepts interrupt requests
				• Executing the maskable interrupt service program with the same priority (This excludes executing an interrupt service program in priority level 3 when the PRSL bit[Note 5] is cleared to zero.) • Executing a high-priority interrupt service program	Execute the instruction following the MOV STBC, #byte instruction. (The interrupt request that released the IDLE mode is saved[Note 4].)
	0	0	0	–	
	1	0	×	–	Holds the IDLE mode
	×	1	×		

Notes 1. Interrupt mask bit in each interrupt request source

2. Macro service enable flag that is in each interrupt request source

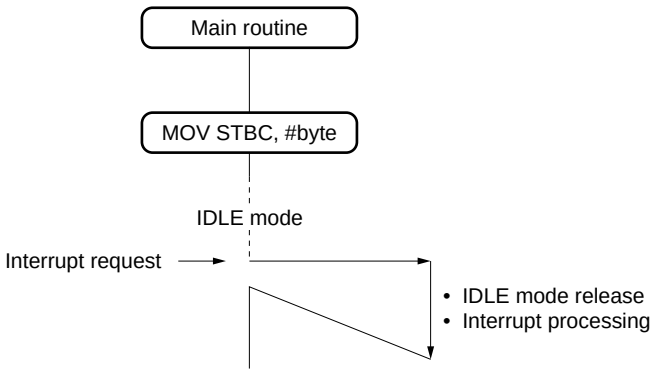
3. Interrupt enable flag in the program status word (PSW)

4. The saved interrupt request is accepted when acceptance is possible.

5. Bit in the interrupt mode control register (IMC)

Figure 24-9. Operations After IDLE Mode Release (1/2)

(1) Interrupt after IDLE mode.



(2) Reset after IDLE mode.

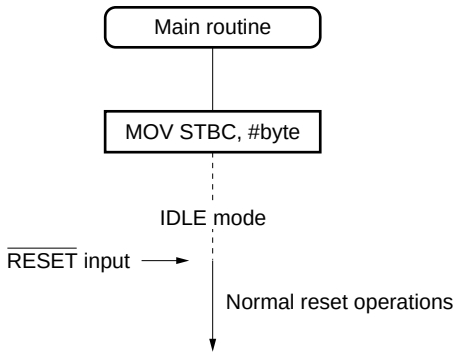
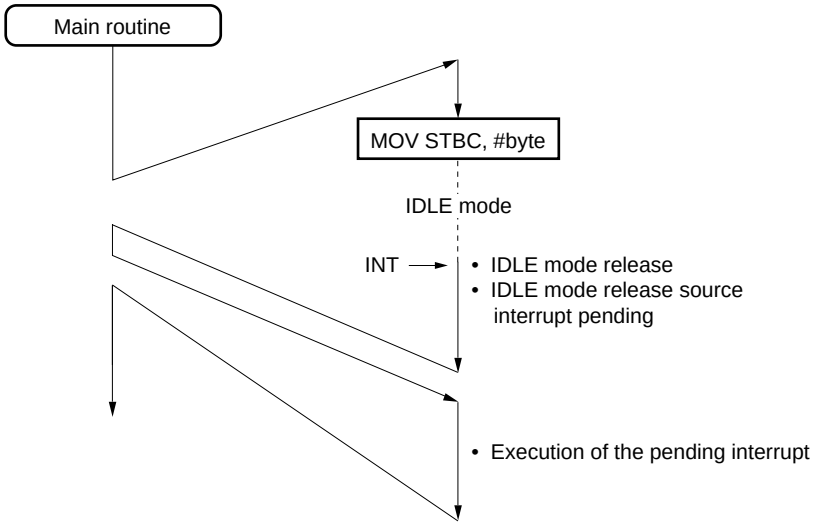
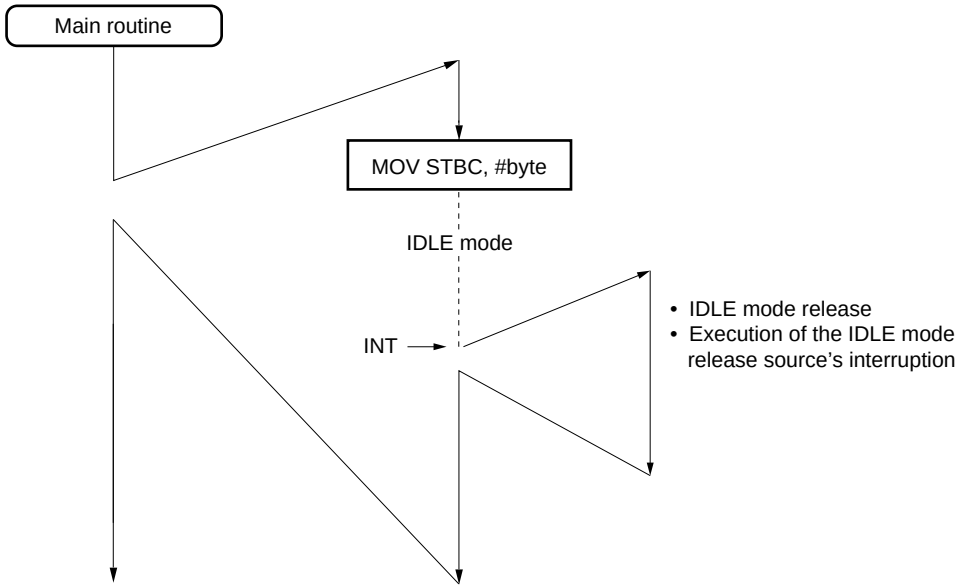


Figure 24-9. Operations After IDLE Mode Release (2/2)

(3) IDLE mode during interrupt processing routine whose priority is higher than or equal to release source interrupt.



(4) IDLE mode during interrupt processing routine whose priority is lower than release source interrupt.



(1) Releasing the IDLE mode by NMI input

When the valid edge specified in the external interrupt edge enable registers (EGP0, EGN0) is input by the NMI input, the IDLE mode is released.

When the IDLE mode is released and the nonmaskable interrupt from the NMI pin input can be accepted, execution branches to the NMI interrupt service program. If acceptance is not possible (such as when set in the IDLE mode in the NMI interrupt service program), execution starts again from the instruction following the instruction that set the IDLE mode. When acceptance is enabled, execution branches to the NMI interrupt service program (by executing the RETI instruction).

For details about NMI interrupt acceptance, refer to **22.6 Non-maskable Interrupt Acknowledgment Operation**.

(2) Releasing the IDLE mode by INTP0 to INTP5 input and watch timer interrupt

If interrupt masking by INTP0 to INTP5 input is canceled and macro service is prohibited and the valid edge specified with the external interrupt edge enable register (EGP0, EGN0) is input to INTP0 to INTP5, the IDLE mode is canceled. At the same time, an overflow will occur with the watch timer and the IDLE mode will be released when the watch timer interrupt mask is released and macro services are prohibited.

If interrupts can be accepted when released from the IDLE mode and the interrupt enable flag (IE) is set to one, execution branches to the interrupt service program. If the IE flag is cleared to zero when acceptance is not possible, execution starts again from the instruction following the instruction that set the IDLE mode.

For details on interrupt acceptance, refer to **22.7 Maskable Interrupt Acknowledgment Operation**.

(3) Releasing the IDLE mode by $\overline{\text{RESET}}$ input

When $\overline{\text{RESET}}$ input falls from high to low and the reset condition is entered, the oscillator starts oscillating. Maintain the oscillation stabilization time for the $\overline{\text{RESET}}$ active period. Then, when the $\overline{\text{RESET}}$ rises, normal operation starts.

The difference from the normal reset operation is the data memory saves the contents before setting the IDLE mode.

24.6 Chack Items When Using STOP or IDLE Mode

The checks required to decrease the consumption current when using the STOP mode or IDLE mode are described below.

(1) Is the output level of each output pin appropriate?

The appropriate output level of each pin differs with the circuit in the next stage. Select the output level so that the consumption current is minimized.

- If a high level is output when the input impedance of the circuit in the next stage is low, current flows from the power source to the port, and the consumption current increases. This occurs when the circuit in the next stage is, for example, a CMOS IC. When the power supply is turned off, the input impedance of a CMOS IC becomes low. To suppress the consumption current and not negatively affect the reliability of the CMOS IC, output a low level. If a high level is output, latch-up results when the power supply is applied again.

- Depending on the circuit in the next stage, the consumption current sometimes increases when a low level is input. In this case, output a high level or high impedance to eliminate the consumption current.

- When the circuit in the next stage is a CMOS IC, if the output is high impedance when power is supplied to the CMOS IC, the consumption current of the CMOS IC sometimes increases (in this case, the CMOS IC overheats and is sometimes destroyed). In this case, output a suitable level or pullup or pulldown resistors.

The setting method for the output level differs with the port mode.

- Since the output level is determined by the state of the internal hardware when the port is in the control mode, the output level must be set while considering the state of the internal hardware.

- The output level can be set by writing to the output latch of the port and the port mode register by the software when in the port mode.

When the port enters the control mode, the port mode is changed by simply setting the output level.

(2) Is the input level to each input pin appropriate?

Set the voltage level input to each pin within the range from the V_{SS} voltage to the V_{DD} voltage. If a voltage outside of this range is applied, not only does the consumption current increase, but the reliability of the μPD784225 is negatively affected.

In addition, do not increase the middle voltage.

(3) Are internal pullup resistors needed?

Unnecessary pull-up resistors increase the consumption current and are another cause of device latch-up. Set the pull-up resistors to the mode in which they are used only in the required parts.

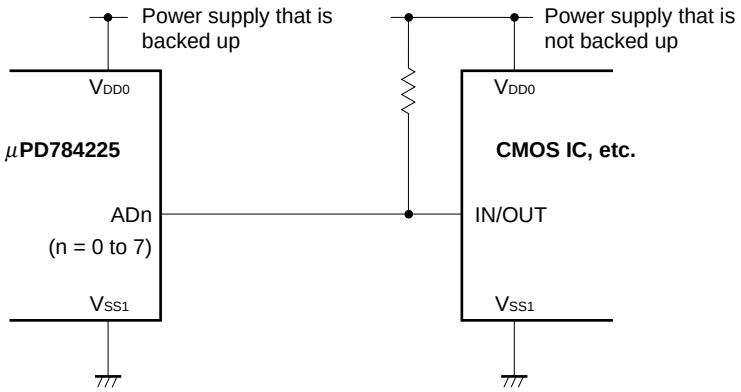
When the parts needing pull-up resistors and the parts not needing them are mixed together, externally connect the pull-up resistors where they are needed and set the mode in which the internal pull-up resistors are not used.

(4) Are the address bus, the address/data bus, etc. handled appropriately?

The address bus, address/data bus, and $\overline{RD}$ and $\overline{WR}$ pins have high impedances in the STOP and IDLE modes. Normally, these pins are pulled up by pull-up resistors. If the pull-up resistors are connected to the power supply that is backed up, the current flows through the pull-up resistors when the low input impedance of the circuit connected to the power supply that is not backed up, and the consumption current increases. Therefore, connect the pull-up resistors on the power supply side that is not backed up as shown in Figure 24-10.

The ASTB pin has a high impedance in both the STOP and IDLE modes. Handle in the manner described in (1) above.

Figure 24-10. Example of Handling Address/Data Bus



Set the input voltage level applied to the $\overline{WAIT}$ pin in the range from the V_{SS1} voltage to the V_{DD0} voltage. If a voltage outside of this range is applied, not only does the consumption current increase, but the reliability of the μPD784225 is negatively affected.

(5) A/D converter

The current flowing through pin AV_{DD} can be reduced by clearing the ADCS bit, that is bit 7 in the A/D converter mode register (ADM), to zero. Furthermore, if you want to decrease the current, disconnect the current supplied to AV_{DD} by an externally attached circuit.

The AV_{DD} pin must always have the same voltage as the V_{DD} pin. If current is not supplied to the AV_{DD} pin in the STOP mode, not only does the consumption current increase, but the reliability of the μPD784225 is negatively affected.

(6) D/A converter

The D/A converter consumes a constant current in the STOP and IDLE modes. By clearing the DACEn (n = 0, 1) bits in the D/A converter mode registers (DAM0, DAM1) to zero, the output of ANOn (n = 0, 1) has high impedance, and the consumption current can be decreased.

Do not apply an external voltage to the ANOn pins. In an external voltage is applied, not only is the consumption current increased, but the μPD784225 may be destroyed or the reliability decreased.

24.7 Low Power Consumption Mode

24.7.1 Setting low power consumption mode[Note]

When the low power consumption mode is entered, set 40H in the standby control register (STBC). This setting switches the system clock from the main system clock to the subsystem clock.

Whether the system clock switched to the subsystem clock can be verified from the data read from bit CST in the clock status register (PCS) (refer to **Figure 24-3**).

To check whether switching has ended, set 44H in STBC to stop the oscillation of the main system clock. Then switch to the backup power supply from the main power supply.

Note The low power consumption mode is the state where the subsystem clock is used as the system clock, and the main system clock is stopped.

Figure 24-11 shows the flow for setting subsystem clock operation. Figure 24-12 shows the setting timing diagram.

Figure 24-11. Flow for Setting Subsystem Clock Operation

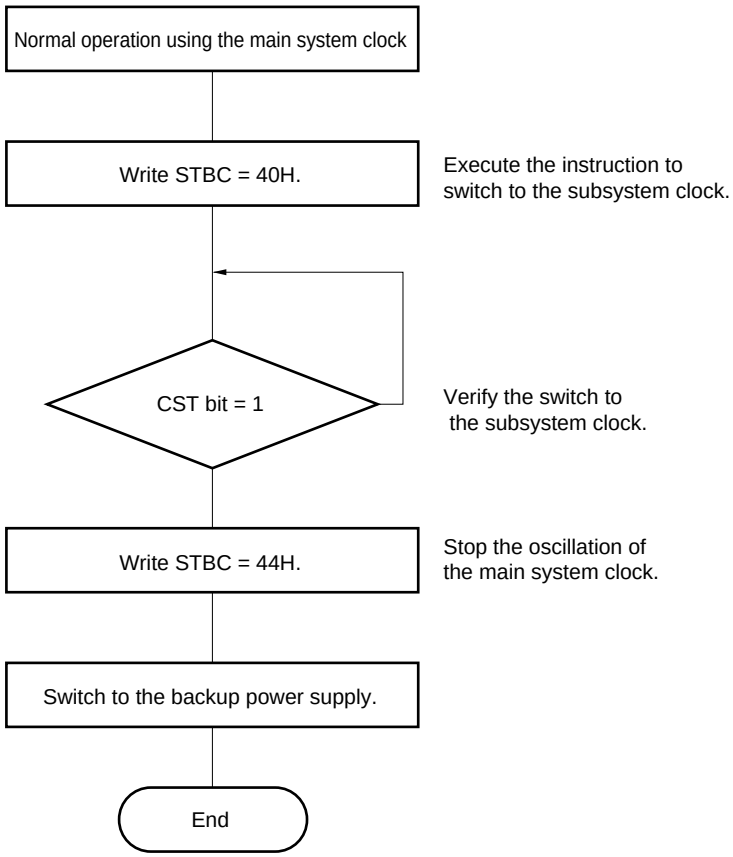
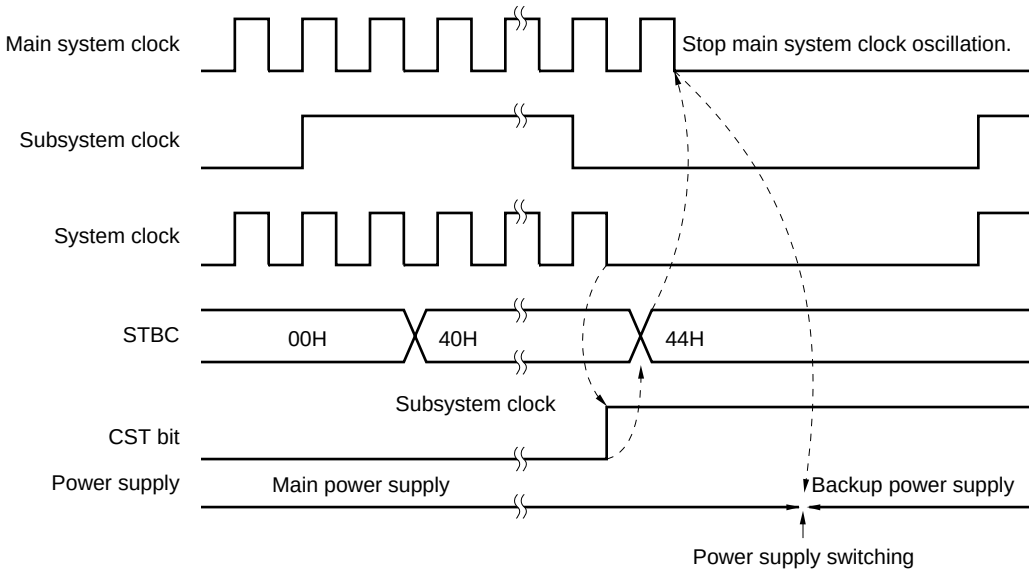


Figure 24-12. Setting Timing for Subsystem Clock Operation



24.7.2 Returning to main system clock operation

When returning to main system clock operation from subsystem clock operation, the system power supply first switches to the main power supply and enables the oscillation of the main system clock (set STBC = 40H). Then the software waits the oscillation stabilization time of the main system clock, and the system clock switches to the main system clock (set STBC to 00H).

Cautions 1. When returning from subsystem clock operation (stopped oscillation of the main system clock) to main system clock operation, do not simultaneously specify bit MCK = 0 and bit CK2 = 0 by write instructions to STBC.

2. The oscillation stabilization time setting register (OSTS) specifies the oscillation stabilization time after the STOP mode is released, except when released by $\overline{\text{RESET}}$, when the system clock is the main system clock. This cannot be used when the system clock is restored from the subsystem clock to the main system clock.

Figure 24-13 is the flow for restoring main system clock operation, and Figure 24-14 is the restore timing diagram.

Figure 24-13. Flow to Restore Main System Clock Operation

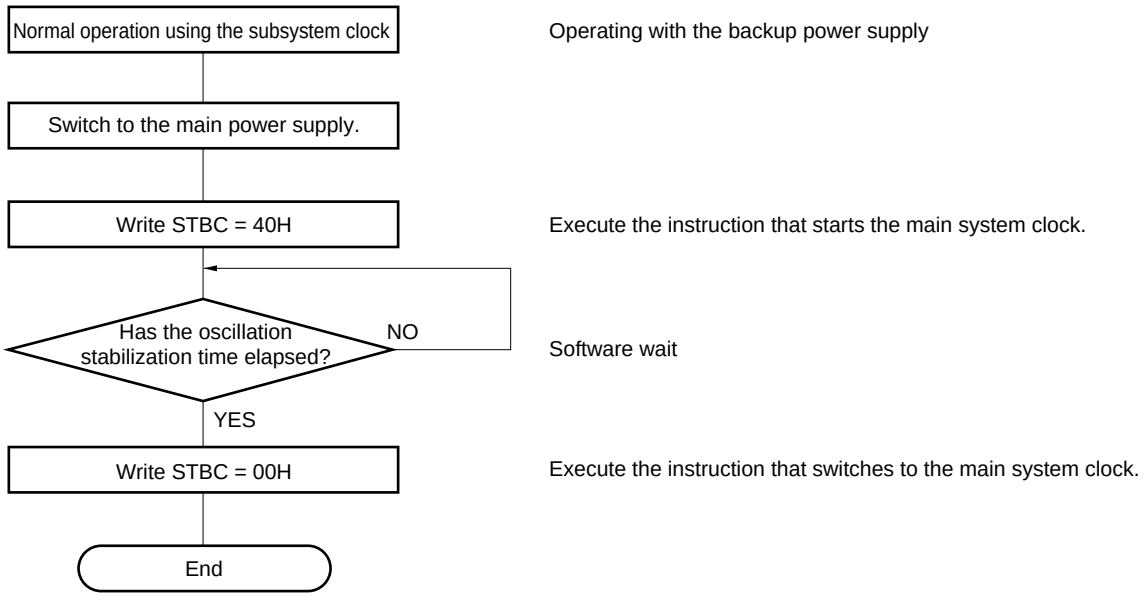
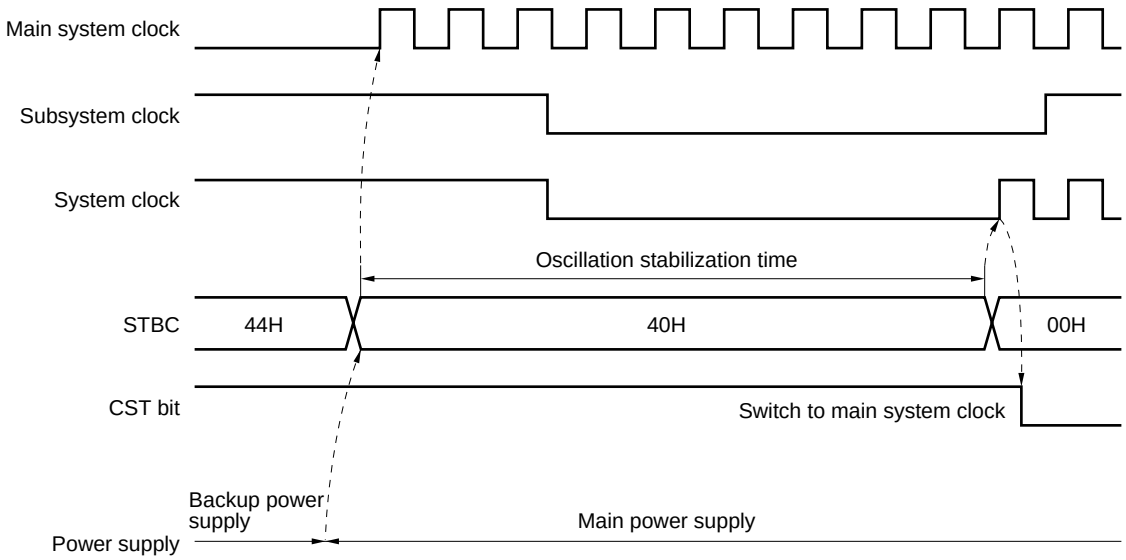


Figure 24-14. Timing for Restoring Main System Clock Operation



24.7.3 Standby function in low power consumption mode

The standby function in the low power consumption mode has a HALT mode and an IDLE mode.

(1) HALT mode

(a) HALT mode settings and the operating states

When set in the HALT mode in the low power consumption mode, set 45H in STBC. Table 24-9 shows the operating states in the HALT mode.

Table 24-9. Operating States in HALT Mode

Item		Operating State
Clock generation circuit		The clock supplied to the CPU stops, and only the main system clock stops oscillating.
CPU		Operation disabled
Port (output latch)		Saves the state before setting the HALT mode.
16-bit timer/counter		Operational when the watch timer output is selected as the count clock (Select f_{XT} as the count clock of the watch timer)
8-bit timer/counters 1, 2		Operational when TI1 and TI2 are selected as the count clocks
8-bit timer/counters 5, 6		Operational when TI5 and TI6 are selected as the count clocks
Watch timer		Operational only when f_{XT} is selected as the count clock
Watchdog timer		Operation disabled
A/D converter		Operation disabled
D/A converter		Operation enabled
Real-time output port		Operational when an external trigger is used or TI1 and TI2 are selected as the count clocks of the 8-bit timer/counters 1 and 2
Serial interface	Except I^2C bus mode	Operational only when an external input clock is selected as the serial clock
	I^2C bus mode	Operation disabled
External interrupt	INTP0 to INTP5	Operation enabled
Bus lines during external expansion	AD0 to AD7	High impedance
	A0 to A19	Saves the state before the HALT mode is set
	ASTM	Low level
	$\overline{WR}$, $\overline{RD}$	High level
	$\overline{WAIT}$	High impedance

(b) Releasing the HALT mode

(i) Releasing the HALT mode by NMI input

When the valid edge specified by the external interrupt edge enable registers (EGP0, EGN0) is input to the NMI input, the IDLE mode is released.

When released from the HALT mode, if non-maskable interrupts by the NMI pin input can be accepted, execution branches to the NMI interrupt service program. If interrupts cannot be accepted (when set in the HALT mode by the NMI interrupt service program), execution starts again from the instruction following the instruction that set the HALT mode. When interrupts can be accepted (by executing the RETI instruction), execution branches to the NMI interrupt service program.

For details about accepting NMI interrupts, refer to **22.6 Non-maskable Interrupt Acknowledgment Operation**.

(ii) Releasing the HALT mode by a maskable interrupt request

An unmasked maskable interrupt request is generated to release the HALT mode.

When the HALT mode is released and the interrupt enable flag (IE) is set to one, if the interrupt can be accepted, execution branches to interrupt service program. When interrupts cannot be accepted and when the IE flag is cleared to zero, execution restarts from the instruction following the instruction that set the HALT mode.

For details about interrupt acceptance, refer to **22.7 Maskable Interrupt Acknowledgment Operation**.

(iii) Releasing the HALT mode by $\overline{\text{RESET}}$ input

When $\overline{\text{RESET}}$ input falls from high to low and the reset condition is entered, the oscillator starts oscillating. Maintain the oscillation stable time for the $\overline{\text{RESET}}$ active period. Then when $\overline{\text{RESET}}$ rises, normal operation starts.

The difference from the normal reset operation is the data memory saves the contents before setting the HALT mode.

(2) IDLE mode

(a) Setting the IDLE mode and the operating states

When the low power consumption mode is set in the IDLE mode, set 47H in STBC.

Table 24-10 shows the operating states in the IDLE mode.

Table 24-10. Operating States in IDLE Mode

Item		Operating State
Clock generation circuit		The main system clock stops oscillating. The oscillation circuit of the subsystem clock continues operating. The clock supplied to the CPU and the peripherals stops.
CPU		Operation disabled
Port (output latch)		Saves the state before setting the IDLE mode
16-bit timer/counter		Operational when the watch timer output is selected as the count clock (Select f_{XT} as the count clock of the watch timer.)
8-bit timer/counters 1, 2		Operational when TI1 and TI2 are selected as the count clocks
8-bit timer/counters 5, 6		Operational when TI5 and TI6 are selected as the count clocks
Watch timer		Operational only when f_{XT} is selected as the count clock
Watchdog timer		Operation disabled
A/D converter		Operation disabled
D/A converter		Operation enabled
Real-time output port		Operational when an external trigger is used or TI1 and TI2 are selected as the count clocks of the 8-bit timer/counters 1 and 2
Serial interface	Except I^2C bus mode	Operational only when an external input clock is selected as the serial clock
	I^2C bus mode	Operation disabled
External interrupt	INTP0 to INTP5	Operation enabled
Bus lines during external expansion	AD0 to AD7	High impedance
	A0 to A19	Holds the state before the IDLE mode is set
	ASTB	Low level
	$\overline{WR}$, $\overline{RD}$	High level
	$\overline{WAIT}$	High impedance

Caution In the IDLE mode, only external interrupts (INTP0 to INTP5) and watch timer interrupts (INTWT) can release the IDLE mode and be acknowledged as interrupt requests. All other interrupt requests are pended, and acknowledged after the IDLE mode has been released through NMI input, INTP0 to INTP5 input, and INTWT.

(b) Releasing the IDLE mode

(i) Releasing the IDLE mode by NMI input

When the valid edge set in the external interrupt edge enable registers (EGP0, EGN0) is input to the NMI input, the IDLE mode is released.

When the IDLE mode is released and non-maskable interrupts by the NMI pin input can be accepted, execution branches to the NMI interrupt service program. When interrupts cannot be accepted (when set to the IDLE mode in the NMI interrupt service program), execution restarts from the instruction following the instruction that set the IDLE mode. When interrupts can be accepted (by executing the RETI instruction), execution branches to the NMI interrupt service program.

For details about accepting NMI interrupts, refer to **22.6 Non-maskable Interrupt Acknowledgment Operation**.

(ii) Releasing IDLE mode by INTP0 to INTP5 inputs and watch timer interrupt

If interrupt masking is released through INTP0 to INTP5 input and macro service is disabled, the oscillator restarts oscillating when a valid edge specified in the external interrupt edge enable registers (EGP0, EGN0) is input to INTP0 to INTP5. At the same time, an overflow will occur with the watch timer and the IDLE mode will be released when the watch timer interrupt mask is released and macro services are prohibited.

When the IDLE mode is released and the interrupt enable flag (IE) is set to one, if interrupts can be accepted, execution branches to interrupt service program. When interrupts cannot be accepted and when the IE flag is cleared to zero, execution restarts from the instruction following the instruction that set the IDLE mode.

For details about interrupt acceptance, refer to **22.7 Maskable Interrupt Acknowledgment Operation**.

(iii) Releasing the IDLE mode by $\overline{\text{RESET}}$ input

When $\overline{\text{RESET}}$ input falls from high to low and the reset condition is entered, the oscillator starts oscillating. Maintain the oscillation stable time for the $\overline{\text{RESET}}$ active period. Then when $\overline{\text{RESET}}$ rises, normal operation starts.

The difference from the normal reset operation is the data memory saves the contents before setting the IDLE mode.

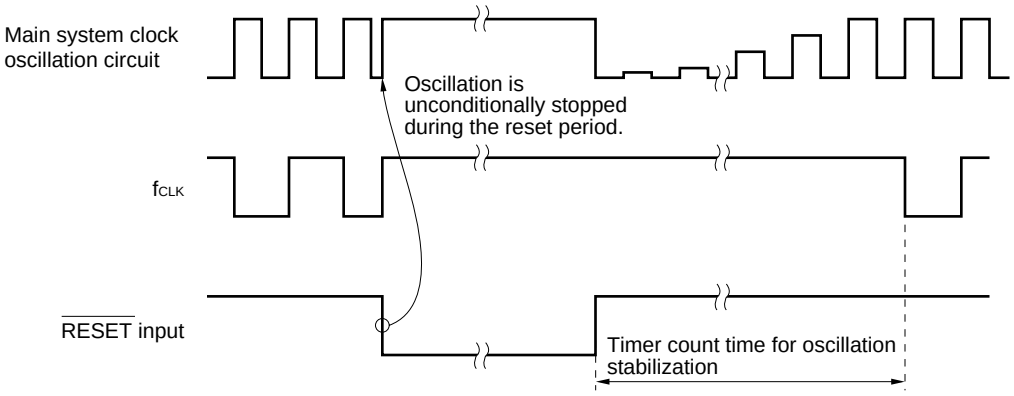
[MEMO]

CHAPTER 25 RESET FUNCTION

When a low level is input to $\overline{\text{RESET}}$ input pin, system reset is performed. The hardware enters the states listed in Figure 25-1. Since the oscillation of the main system clock unconditionally stops during the reset period, the consumption current of the entire system can be reduced.

When $\overline{\text{RESET}}$ input goes from low to high, the reset state is released. After the count time of the timer for oscillation stabilization (41.9 ms: in 12.5-MHz operation), the content of the reset vector table is set in the program counter (PC). Execution branches to the address set in the PC, and program execution starts from the branch destination address. Therefore, the reset can start from any address.

Figure 25-1. Oscillation of Main System Clock in Reset Period



To prevent error operation caused by noise, a noise elimination circuit based on an analog delay is installed at $\overline{\text{RESET}}$ input pin.

Figure 25-2. Accepting Reset Signal

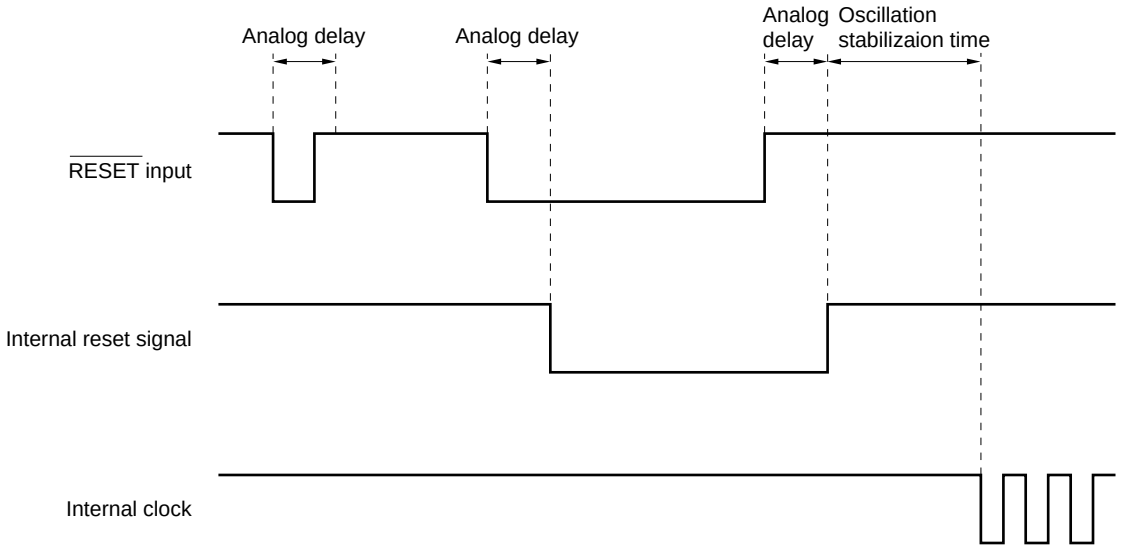


Table 25-1. State After Reset for All Hardware Resets

Hardware	State During Reset ($\overline{\text{RESET}}$ = L)	State After Reset ($\overline{\text{RESET}}$ = H)
Main system clock oscillation circuit	Oscillation stops	Oscillation starts
Subsystem clock oscillation circuit	Not affected by the reset	
Program counter (PC)	Undefined	Set a value in the reset vectored table.
Stack pointer (SP)	Undefined	
Program status word (PSW)	Initialize to 0000H.	
Internal RAM	This is undefined. However, when the standby state is released by a reset, the value is saved before setting standby.	
I/O lines	The input and output buffers turn off.	High impedance
Other hardware	Initialize to the fixed state**Note**.	

Note See **Table 3-6, Special Function Register (SFR) List** when resetting.

CHAPTER 26. ROM CORRECTION

26.1 ROM Correction Functions

μPD784224, 784225, 784224Y and 784225Y convert part of the program within the mask ROM into the program within the internal expansion ROM.

The use of ROM correction enables command bugs discovered in the mask ROM to be repaired, and change the flow of the program.

ROM correction can be used in a maximum of four locations within the internal ROM (program).

Caution Note that ROM correction cannot perform emulation in the in-circuit emulator (IE-784000-R, IE-784000-R-EM).

In more detail, the command addresses that require repair from the inactive memory connected to an external micro-computer by a user program and the repair command codes are loaded into the peripheral RAM.

The above addresses and the internal ROM access addresses are compared by the comparator built into the micro computer during execution of internal ROM programs (during command fetch), and internal ROM's output data is then converted to call command (CALLT) codes and output when a match is determined.

When the CALLT command codes are changed to valid commands by the CPU and executed, the CALLT table is referenced, and the process routine and other peripheral RAM are branched. At this point, a CALLT table is prepared for each repair address for referencing purposes. Four repair address can be set for the μPD784225.

Match-ups with address pointer 0	: CALLT table (0078H) Conversion command code: FCH
Match-ups with address pointer 1	: CALLT table (007AH) Conversion command code: FDH
Match-ups with address pointer 2	: CALLT table (007BH) Conversion command code: FEH
Match-ups with address pointer 3	: CALLT table (007CH) Conversion command code: FFH

Caution As it is necessary to reserve four locations for the CALLT tables when the ROM correction function is used (0078H, 007AH, 007CH, 007EH), ensure that these are not used for other applications. However, the CALLT tables can be used if the ROM correction function is not being used.

The differences between 78K/IV ROM correction and 78K/0 ROM correction are shown in Table 26-1.

Table 26-1. Differences between 78K/IV ROM Correction and 78K/0 ROM Correction

Difference	78K/IV	78K/0
Generated command codes	CALLT instruction (1-byte instruction: FCH, FDH, FEH, FFH)	Peripheral RAM
Address comparison conditions	Instruction fetch only	Instruction fetch only
Correction status flag	None As there is a possibility that the addresses match owing to an invalid fetch, the status is not necessary	Yes
Jump destination address during correction	CALLT Table 0078H, 007AH, 007CH, 007EH	Fixed address on the peripheral RAM

26.2 ROM Correction Configuration

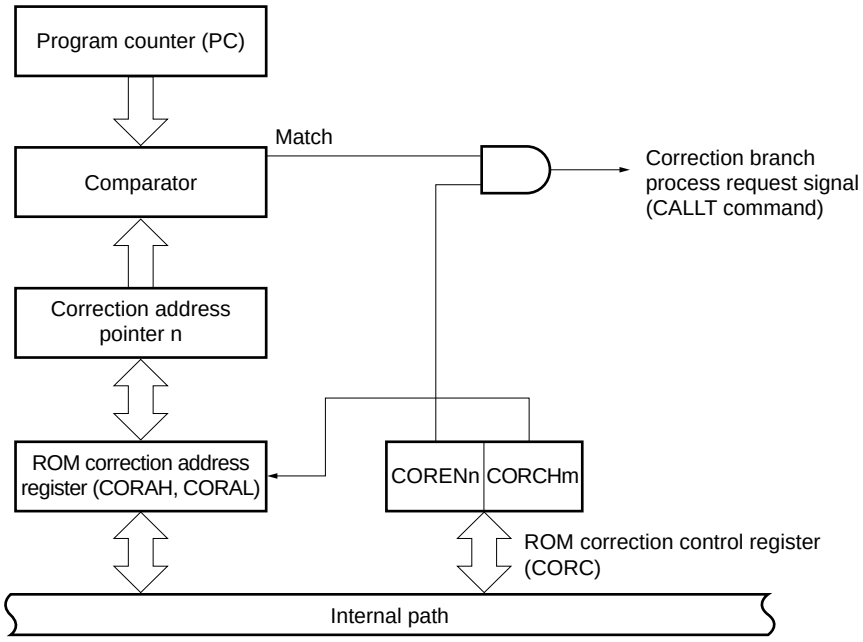
ROM correction is composed of the following hardware.

Table 26-2. ROM Correction Configuration

Item	Configuration
Register	ROM correction address register H, L (CORAH, CORAL)
Control register	ROM correction control register (CORN)

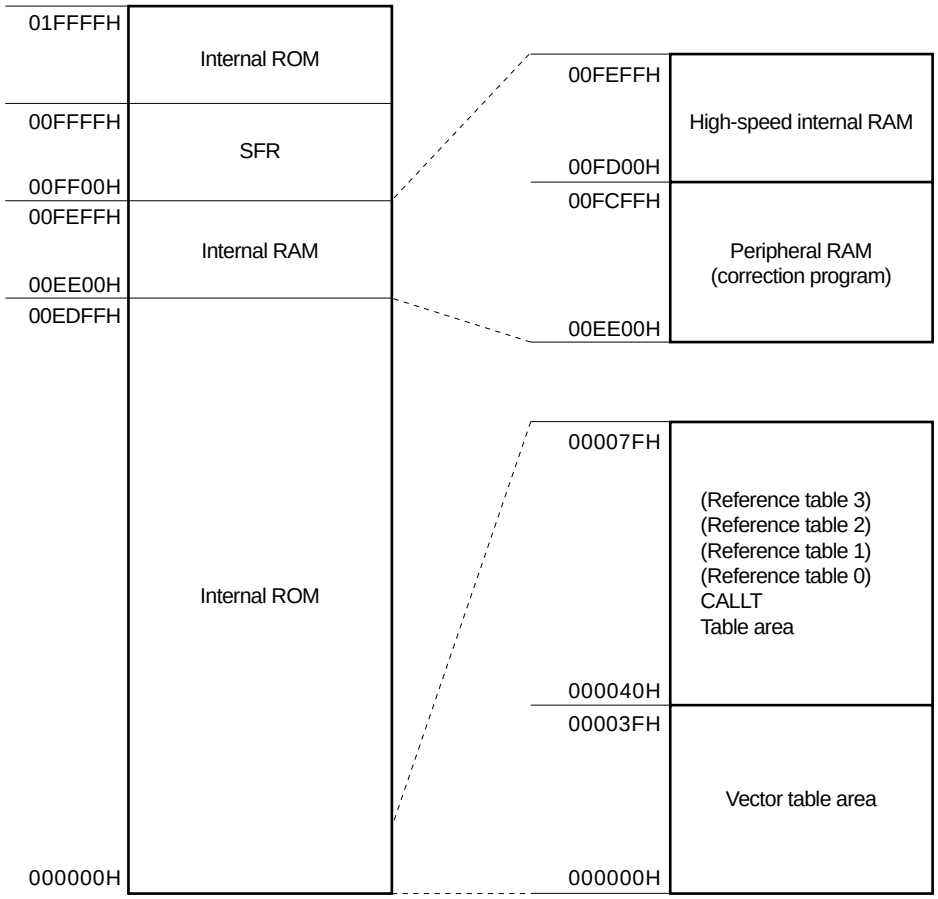
A ROM correction block diagram is shown in Figure 26-1, and Figure 26-2 shows an example of memory mapping.

Figure 26-1. ROM Correction Block Diagram



Remark n = 0 to 3, m = 0, 1

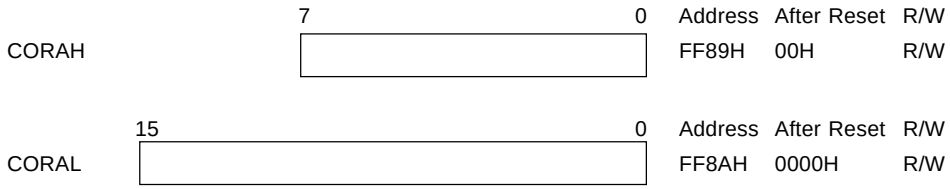
Figure 26-2. Memory Mapping Example (μPD784225)



(1) ROM correction address register (CORAH, CORAL)

The register that sets the header address (correction address) of the command within the mask ROM that needs to be repaired. A maximum of four program locations can be repaired with ROM correction. First of all, the channel is selected with bit 0 (CORCH0) and bit 1 (CORCH1) of the ROM correction control register (CORC), and the address is then set in the specified channel's address pointer when the address is written in CORAH and CORAL.

Figure 26-3. ROM Correction Address Register (CORAH, CORAL) Format



(2) Comparator

The ROM correction address registers H and L (CORAH, CORAL) normally compare the corrected address value with the fetch register value. If any of the ROM correction control register (CORC) bits between bit 4 to bit 7 (COREN0 to 3) are 1 and the correct address matches the fetch address value, a table reference instruction (CALLT) is issued from the ROM correction circuit.

26.3. Control Register for ROM Correction

ROM correction is controlled by the ROM correction control register (CORC).

(1) ROM correction control register (CORC)

The register that controls the issuance of the table reference instruction (CALLT) when the correct address set in ROM correction address registers H and L (CORAH, CORAL) match the value of the fetch address.

This is composed of a correction enable flag (COREN0 to 3) that enbales or disables match detection with the comparator, and four channel correction pointers.

CORC is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets 00H to CORC.

Figure 26-4. ROM Correction Control Register (CORC) Format

Address : 0FF88H After Reset : 00H R/W

Symbol	7	6	5	4	3	2	1	0
CORC	COREN3	COREN2	COREN1	COREN0	0	0	CORCH1	CORCH0

CORENn	Controls the match detection for the ROM correction address register and the fetch address.
0	Disabled
1	Enabled

CORCH1	CORCH0	Channel selection
0	0	Address pointer channel 0
0	1	Address pointer channel 1
1	0	Address pointer channel 2
1	1	Address pointer channel 3

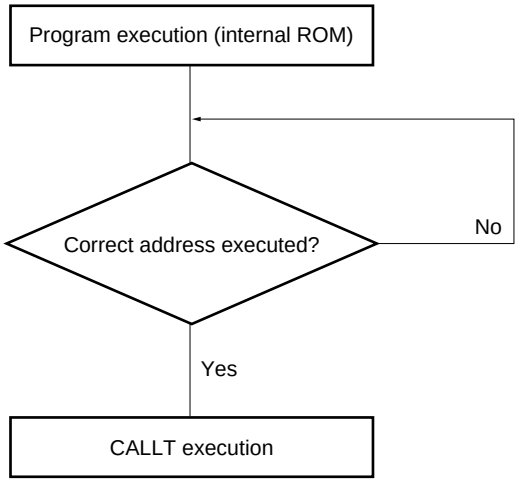
Remark n = 0 to 3

26.4 Usage of ROM Correction

<1> The correct address and post-correction instruction (correction program) are stored in the microcontroller external inactive memory (EEPROM™).

<2> A substitute instruction is read from the inactive memory with the use of a serial interface when the initialization program is running after being reset, and this is stored in the peripheral RAM and external memory. The correction channel is then selected, the address for the command that requires correction is read and set in the ROM correction address registers (CORAH, CORAL), and the correction enable flag (COREN0 to 3) is set at 1. A maximum of four locations can be set.

<3> Execute the CALLT instruction during execution of the corrected address.



<4> CALLT routine branch
When matched with address pointer 0: CALLT table (0078H)
When matched with address pointer 1: CALLT table (007AH)
When matched with address pointer 2: CALLT table (007CH)
When matched with address pointer 3: CALLT table (007EH)

<5> Execute substitute instruction

<6> Add +3 to the stack pointer (SP)

<7> Restore to any addresses with the branch instruction (BR)

26.5 Conditions for Executing ROM Correction

In order to use the ROM correction function, it is necessary for the external environment and program to satisfy the following conditions.

(1) External environment

Must be connected externally to an inactive memory, and be configured to read that data.

(2) Target program

The data setting instruction for CORC, CORAH and CORAL will be previously annotated in the target program (program stored in the ROM).

The set-up data (the items written in lower-case in the set-up example below) must be read from the external inactive memory, and the correct number of required correction pointers must be set.

Example of four pointer settings

```
                MOV     CORC,   #00H        ; Specified channel 0
                MOVW    CORAL,  #ch0 datal  ; Sets the channel 0 matching address
                MOV     CORAH, #ch0 datah   ; Sets the channel 0 matching address
                MOV     CORC,   #01H        ; Specified channel 1
                MOVW    CORAL,  #ch1 datal  ; Sets the channel 1 matching address
                MOV     CORAH, #ch1 datah   ; Sets the channel 1 matching address
                MOV     CORC,   #02H        ; Specified channel 2
                MOVW    CORAL,  #ch2 datal  ; Sets the channel 2 matching address
                MOV     CORAH, #ch2 datah   ; Sets the channel 2 matching address
                MOV     CORC,   #chH        ; Specified channel 3
                MOV     CORAL,  #ch3 datah  ; Sets the channel 3 matching address
                MOV     CORAH, #ch3 datal   ; Sets the channel 3 matching address
                MOV     CORC,   #romcor en
                                            ; Sets 00H when correction is disabled
                                            ; Sets F0H when correction is operated
                BR      $NORMAL
                BR      ! ! COR ADDR        ; Specifies the address of the correction program
                ;
NOMAL instruction; next instruction
```

(3) Setting the branch instruction in the CALLT table.

In the case of the above program, the header address for the BR!!COR_ADDR instruction is specified. (COR ADDR indicates the address where the correction program is located.)

The reason for this being branched into the CALLT instruction and BR instruction is owing to the fact that only the base area can be branched with CALLT. There is no necessity to branch into two levels when it is to be attached to the RAM base area with the LOCATION instruction.

CHAPTER 27 μPD78F4225 AND μPD78F4225Y PROGRAMMING

Flash memories in the μPD784225 and 784225Y Subseries are available for μPD78F4225 and 78F4225Y. This chapter provides an explanation of using the μPD78F4225 and 78F4225Y as substitutes for the μPD78F4225.

μPD78F4225 and 78F4225Y are versions with built-in flash memories that enable programs to be written, deleted and overwritten while mounted onto substrates. The differences between flash memory versions (μPD78F4225 and 78F4225Y) and mask ROM versions (μPD784224, 784225, 784224Y and 784225Y) are shown in Table 27-1.

Table 27-1. Differences between the μPD78F4225 and 78F4225Y Mask ROM Versions

Item	μPD78F4225, 78F4225Y	Mask ROM versions
Internal ROM structure	Flash memory	Mask ROM
Internal ROM capacity	128 Kbytes	μPD784224, 784224Y: 96 Kbytes μPD784225, 784225Y: 128 Kbytes
Internal RAM capacity	4,352 bytes	μPD784224, 784224Y: 3,584 bytes μPD784225, 784225Y: 4,352 bytes
Amending internal ROM capacity with the internal flash memory size switching register (IMS)	Possible Note	Not possible
V_{PP} pin	Availabe	Not available

Note The capacity of the flash memory will become 128 Kbytes and the internal RAM capacity will become 4,352 bytes with $\overline{\text{RESET}}$ input.

Caution Noise resistance and noise immunity differ with flash memory versions and mask ROM versions. When examinations are to be performed on overwriting from flash memory versions to mask ROM versions during the transition from trials to mass production, ensure that the mask ROM's CS versions (not ES products) have been sufficiently evaluated.

27.1 Internal Memory Size Switching Register (IMS)

The IMS is a register to prevent a certain part of the internal memory from being used by software. By setting the IMS, it is possible to establish a memory map that is the same as the mask ROM product's memory map for the internal memory (ROM, RAM) which has a different capacity.

IMS is set by a 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets IMS to FFH.

Figure 27-1. Internal Memory Size Switching Register (IMS) Format

Address : 0FFFCH After Reset : FFH W

Symbol	7	6	5	4	3	2	1	0
IMS	1	1	ROM1	ROM0	1	1	RAM1	RAM0

ROM1	ROM0	Internal ROM Capacity Selection
0	0	48 Kbyte
0	1	64 Kbyte
1	0	96 Kbyte
1	1	128 Kbyte

RAM1	RAM0	Internal High-Speed RAM Capacity Selection
0	0	1,536 bytes
0	1	2,304 bytes
1	0	3,072 bytes
1	1	3,840 bytes

Caution The IMS is not available for mask ROM versions (μPD784224, 784225, 784224Y and 784225Y).

The IMS settings to create the same memory map as mask ROM versions is shown in table 27-2.

Table 27-2. Internal Memory Size Switching Register (IMS) Settings

Relevant Mask ROM Product	IMS Setting
μPD784224, 784224Y	EEH
μPD784225, 784225Y	FFH

27.2 Flash Memory Overwriting

The μPD78F4225 is equipped with an on-chip 128-Kbyte flash memory. There are two methods available for overwriting in the built-in flash memory.

- On-board overwrite mode : Overwriting performed with the use of the flash writer.
- Self overwrite mode : Overwriting performed by a program loaded into the μPD78F4225 flash memory.

Flash memory overwriting can be performed 100 times. Program amendments can also be performed with the self overwrite mode without having to prepare a special writing tool when upgraded.

+10V of electrical power is required for deleting and writing in the flash memory.

27.3 On-board Overwrite Mode

The on-board overwrite mode is used with the target system mounted (on-board). Overwriting is performed by connecting a special flash writer (Flashpro II) to the host machine or target system. Overwriting is controlled via the serial interface.

Remark Flashpro II is a product of Naitou Densei Machidaseisakusho Co., Ltd.

On-board overwrite mode settings are performed by controlling the TEST/V_{PP} pin and $\overline{\text{RESET}}$ pin. Serial interface selected is performed with the number of pulses applied to the TEST/V_{PP} pin.

27.3.1 Selecting communication protocol

Flashpro II writes to flash memory by serial communication. The communication protocol is selected from Table 27-3 then writing is performed. The selection of the communication protocol has the format shown in Figure 27-2. Each communication protocol is selected by the number of V_{PP} pulses shown in Table 27-3.

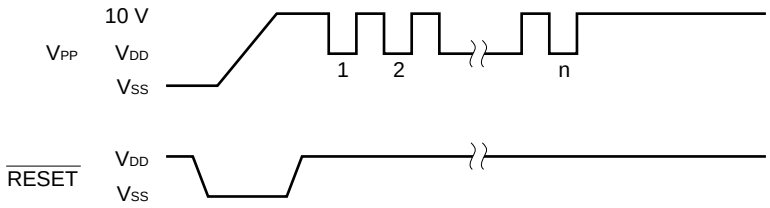
Table 27-3. Communication Protocols

Communication Protocol	No. of Channels	Pins Used	No. of V_{PP} Pulses
3-wire serial I/O	3	$\overline{SCK0}$/SCL0[Note]/P27 SO0/P26 SI0/SDA0[Note]/P25	0
		$\overline{SCK1}$/ASCK1/P22 SO1/TxD1/P21 SI1/RxD1/P20	1
		$\overline{SCK2}$/ASCK2/P72 SO2/TxD2/P71 SI2/RxD2/P70	2
UART	2	TxD1/SO1/P21 RxD1/SI1/P20	8
		TxD2/SO2/P71 RxD2/SI2/P70	9

Note Only in the μPD784225Y Subseries

Caution Select the communication protocol by using number of V_{PP} pulses given in Table 27-3.

Figure 27-2. Communication Protocol Selection Format



27.3.2 On-board overwrite mode functions

By transmitting and receiving various commands and data by the selected communication protocol, operations such as writing to the flash memory are performed. Table 27-4 shows the major functions.

Table 27-4. On-board Overwrite Mode Major Functions

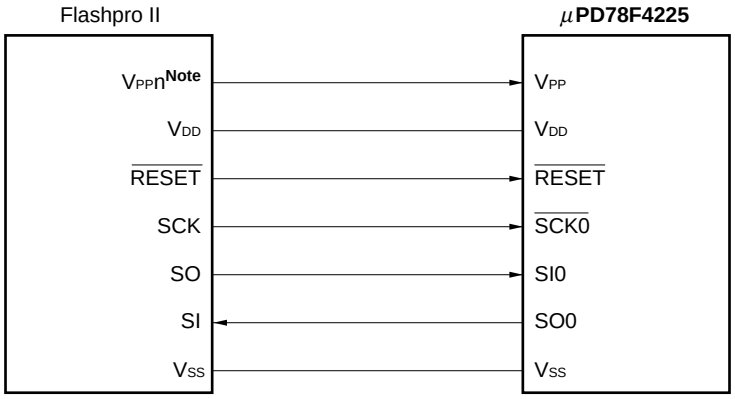
Function	Description
Batch erase	Erase the entire memory contents.
Block erase	Erase the contents of the specified memory block where one memory block is 16 Kbytes.
Batch blank check	Checks the erase state of the entire memory.
Block blank check	Checks the erase state of the specified block.
Data write	Writes to the flash memory based on the start write address and the number of data written (number of bytes).
Batch verify	Compares the data input to the contents of the entire memory.
Block verify	Compares the data input to the contents of the specified memory block.

Verification for the flash memory entails supplying the data to be verified from an external source via a serial interface, and then outputting the existence of unmatched data to the external source after referencing the blocks or all of the data. Consequently, the flash memory is not equipped with a read function, and it is not possible for third parties to read the contents of the flash memory with the use of the verification function.

27.3.3 Connecting Flashpro II

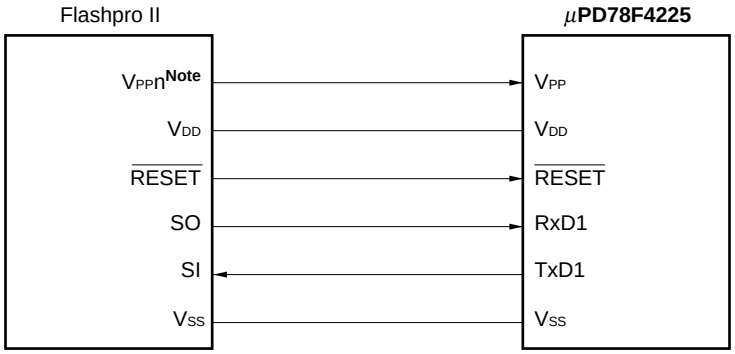
The connection between the Flashpro II and the μPD78F4225 differs with the communication protocol (3-wire serial I/O or UART). Figures 27-3 and 27-4 are the connection diagrams in each case.

Figure 27-3. Flashpro II Connection in 3-wire Serial I/O Method (when using the 3-wire serial I/O0)



Note n = 1, 2

Figure 27-4. Flashpro II Connection in UART Method (when using UART1)



Note n = 1, 2

27.4 Self Overwrite Mode

The self overwrite mode is a mode that enables the μPD78F4225 itself to overwrite in the on-chip flash memory. Contrary to normal operations and memory maps, the self overwrite mode can only use a part of the flash memory, internal data areas, and external memory. When in the self overwrite mode, a specific register value is set in the program located in this area, and deletion and writing is controlled by calling a sub-routine to this specific address.

In order to prevent not being able to write in the flash memory owing to malfunctions during the writing process (power cuts, etc.), an area known as the boot area in which deletion is not possible has been prepared. Flash memory outside of the boot area cannot be used when in the self overwrite mode.

Remark 10V ± 0.3V of voltage is to be applied to the V_{PP} pin only when in the self overwrite mode.

A configuration chart using four pin regulators is shown in Figure 27-5, and a chart for self overwrite mode operation timing is shown in Figure 27-6.

Figure 27-5. Configuration Chart Using Four Pin Regulators

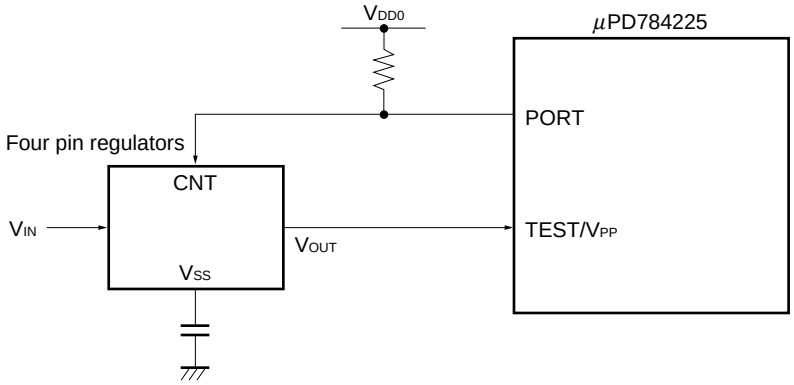
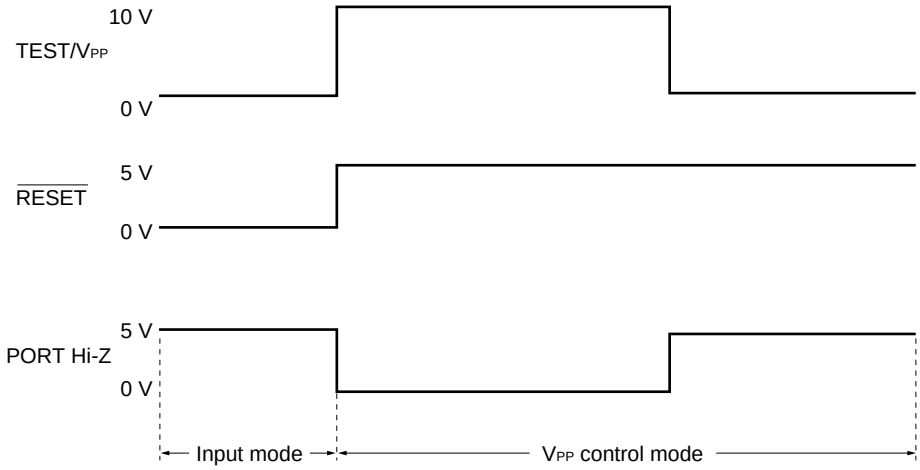


Figure 27-6. Chart for Self Overwrite Mode Operation Timing



28.1 Examples

(1) Operand expression format and description (1/2)

Expression Format	Description
r, r'Note 1	X(R0), A(R1), C(R2), B(R3), R4, R5, R6, R7, R8, R9, R10, R11, E(R12), D(R13), L(R14), H(R15)
r1Note 1	X(R0), A(R1), C(R2), B(R3), R4, R5, R6, R7
r2	R8, R9, R10, R11, E(R12), D(R13), L(R14), H(R15)
r3	V, U, T, W
rp, rp'Note 2	AX(RP0), BC(RP1), RP2, RP3, VP(RP4),UP(RP5), DE(RP6), HL(RP7)
rp1Note 2	AX(RP0), BC(RP1), RP2, RP3
rp2	VP(RP4), UP(RP5), DE(RP6), HL(RP7)
rg, rg'	VVP(RG4), UUP(RG5), TDE(RG6), WHL(RG7)
sfr	Special function register symbol (see the special function register table)
sfrp	Special function register symbol (16-bit manipulation register: see the special function register table)
postNote 2	AX(RP0), BC(RP1), RP2, RP3, VP(RP4), UP(RP5)/PSW, DE(RP6), HL(RP7) Multiple descriptions are possible. However, UP is restricted to the PUSH/POP instruction, and PSW is restricted to the PUSHU/POPU instruction.
mem	[TDE], [WHL], [TDE+], [WHL+], [TDE -], [WHL -], [VVP], [UUP]: register indirect addressing [TDE+byte], [WHL+byte], [SP+byte], [UUP+byte], [VVP+byte]: based addressing imm24[A], imm24[B], imm24[DE], imm24[HL]: indexed addressing [TDE+A], [TDE+B], [TDE+C], [WHL+A], [WHL+B], [WHL+C], [VVP+DE], [VVP+HL]: based indexed addressing
mem1	Everything under mem except [WHL+] and [WHL–]
mem2	[TDE], [WHL]
mem3	[AX], [BC], [RP2], [RP3], [VVP], [UUP], [TDE], [WHL]

Notes 1. By setting the RSS bit to one, R4 to R7 can be used as X, A, C, and B. Use this function only when 78K/III Series programs are also used.

2. By setting the RSS bit to one, RP2 and RP3 can be used as AX and BC. Use this function only when 78K/III Series programs are also used.

(1) Operand expression format and description (2/2)

Expression Format	Description
Note	
saddr, saddr'	FD20H - FF1FH Immediate data or label
saddr1	FE00H - FEFFH Immediate data or label
saddr2	FD20H - FDFFH, FF00H - FF1FH Immediate data or label
saddrp	FD20H - FF1EH Immediate data or label (when manipulating 16 bits)
saddrp1	FE00H - FEFFH Immediate data or label (when manipulating 16 bits)
saddrp2	FD20H - FDFFH, FF00H - FF1EH Immediate data or label (when manipulating 16 bits)
saddrg	FD20H - FEFDH Immediate data or label (when manipulating 24 bits)
saddrg1	FE00H - FEFDH Immediate data or label (when manipulating 24 bits)
saddrg2	FD20H - FDFFH Immediate data or label (when manipulating 24 bits)
addr24	0H - FFFFFFH Immediate data or label
addr20	0H - FFFFFH Immediate data or label
addr16	0H - FFFFH Immediate data or label
addr11	800H - FFFH Immediate data or label
addr8	0FE00H - 0FEFFH[Note] Immediate data or label
addr5	40H - 7EH Immediate data or label
imm24	24-bit immediate data or label
word	16-bit immediate data or label
byte	8-bit immediate data or label
bit	3-bit immediate data or label
n	3-bit immediate data
locaddr	00H or 0FH

Note When 00H is set by the LOCATION instruction, these addresses become the addresses shown here. When 0FH is set by the LOCATION instruction, the values of the addresses shown here added to F0000H become the addresses.

(2) Operand column symbols

Symbol	Description
+	Auto increment
–	Auto decrement
#	Immediate data
!	16-bit absolute address
!!	24-bit/20-bit absolute address
$	8-bit relative address
$!	16-bit relative address
/	Bit reversal
[ ]	Indirect addressing
[%]	24-bit indirect addressing

(3) Flag column symbols

Symbol	Description
(Blank)	Not changed
0	Clear to zero.
1	Set to one.
×	Set or clear based on the result.
P	Operate with the P/V flag as the parity flag.
V	Operate with the P/V flag as the overflow flag.
R	Restore the previously saved value.

(4) Operation column symbols

Symbol	Description
jdisp8	Two's complement data (8 bits) of the relative address distance between the head address of the next instruction and the branch address
jdisp16	Two's complement data (16 bits) of the relative address distance between the head address of the next instruction and the branch address
PC_{HW}	PC bits 16 to 19
PC_{LW}	PC bits 0 to 15

(5) Number of bytes in instruction that has mem in operand

mem Mode	Register Indirect Addressing		Based Addressing	Indexed Addressing	Based Indexed Addressing
No. of bytes	1	2[Note]	3	5	2

Note This becomes a 1-byte instruction only when [TDE], [WHL], [TDE+], [TDE–], [WHL+], or [WHL–] is described in mem in the MOV instruction.

(6) Number of bytes in instruction that has saddr, saddrp, r, or rp in operand

The number of bytes in an instruction that has saddr, saddrp, r, or rp in the operand is described in two parts divided by a slash (/). The following table shows the number of bytes in each one.

Description	No. of Bytes on Left Side	No. of Bytes on Right Side
saddr	saddr2	saddr1
saddrp	saddrp2	saddrp1
r	r1	r2
rp	rp1	rp2

(7) Descriptions of instructions with mem in operand and string instructions

The TDE, WHL, VVP, and UUP (24-bit registers) operands can be described by DE, HL, VP, and UP. However, when DE, HL, VP, and UP are described, they are handled as TDE, WHL, VVP, and UUP (24-bit registers).

28.2 List of Operations

(1) 8-bit data transfer instruction: MOV

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOV	r, #byte	2/3	r ← byte					
	saddr, #byte	3/4	(saddr) ← byte					
	sfr, #byte	3	sfr ← byte					
	!addr16,, #byte	5	(saddr16) ← byte					
	!!addr24, #byte	6	(addr24) ← byte					
	r, r'	2/3	r ← r'					
	A, r	1/2	A ← r					
	A, saddr2	2	A ← (saddr2)					
	r, saddr	3	r ← (saddr)					
	saddr2, A	2	(saddr2) ← A					
	saddr, r	3	(saddr) ← r					
	A, sfr	2	A ← sfr					
	r, sfr	3	r ← sfr					
	sfr, A	2	sfr ← A					
	sfr, r	3	sfr ← r					
	saddr, saddr'	4	(saddr) ← (saddr')					
	r, !addr16	4	r ← (addr16)					
	!addr16, r	4	(addr16) ← r					
	r, !!addr24	5	r ← (addr24)					
	!!addr24, r	5	(addr24) ← r					
	A, [saddrp]	2/3	A ← ((saddrp))					
	A, [%saddrg]	3/4	A ← ((saddrg))					
	A, mem	1-5	A ← (mem)					
	[saddrp], A	2/3	((saddrp)) ← A					
	[%saddrg], A	3/4	((saddrg)) ← A					
	mem, A	1-5	(mem) ← A					
	PSWL #byte	3	PSW_L ← byte	×	×	×	×	×
	PSWH #byte	3	PSW_H ← byte					
	PSWL, A	2	PSW_L ← A	×	×	×	×	×
	PSWH, A	2	PSW_H ← A					
	A, PSWL	2	A ← PSW_L					
	A, PSWH	2	A ← PSW_H					
	r3, #byte	3	r3 ← byte					
	A, r3	2	A ← r3					
	r3, A	2	r3 ← A					

(2) 16-bit data transfer instruction: MOVW

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOVW	rp, #word	3	rp ← word					
	saddrp, #word	4/5	(saddrp) ← word					
	sfrp, #word	4	sfrp ← word					
	!addr16, #word	6	(addr16) ← word					
	!!addr24, #word	7	(addr24) ← word					
	rp, rp'	2	rp ← rp'					
	AX, saddrp2	2	AX ← (saddrp2)					
	rp, saddrp	3	rp ← (saddrp)					
	saddrp2, AX	2	(saddrp2) ← AX					
	saddrp, rp	3	(saddrp) ← rp					
	AX, sfrp	2	AX ← sfrp					
	rp, sfrp	3	rp ← sfrp					
	sfrp, AX	2	sfrp ← AX					
	sfrp, rp	3	sfrp ← rp					
	saddrp, saddrp'	4	(saddrp) ← (saddrp')					
	rp, !addr16	4	rp ← (addr16)					
	!addr16, rp	4	(addr16) ← rp					
	rp, !!addr24	5	rp ← (addr24)					
	!!addr24, rp	5	(addr24) ← rp					
	AX, [saddrp]	3/4	AX ← ((saddrp))					
	AX, [%saddrg]	3/4	AX ← ((saddrg))					
	AX, mem	2-5	AX ← (mem)					
	[saddrp], AX	3/4	((saddrp)) ← AX					
	[%saddrg], AX	3/4	((saddrg)) ← AX					
	mem, AX	2-5	(mem) ← AX					

(3) 24-bit data transfer instruction: MOVG

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOVG	rg, #imm24	5	rg ← imm24					
	rg, rg'	2	rg ← rg'					
	rg, !!addr24	5	rg ← (addr24)					
	!!addr24, rg	5	(addr24) ← rg					
	rg, saddrg	3	rg ← (saddrg)					
	saddrg, rg	3	(saddrg) ← rg					
	WHL, [%saddrg]	3/4	WHL ← ((saddrg))					
	[%saddrg], WHL	3/4	((saddrg)) ← WHL					
	WHL, mem1	2-5	WHL ← (mem1)					
	mem1, WHL	2-5	(mem1) ← WHL					

(4) 8-bit data exchange instruction: XCH

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
XCH	r, r'	2/3	r ↔ r'					
	A, r	1/2	A ↔ r´					
	A, saddr2	2	A ↔ (saddr2)					
	r, saddr	3	r ↔ (saddr)					
	r, sfr	3	r ↔ sfr					
	saddr, saddr'	4	(saddr) ↔ (saddr')					
	r, !addr16	4	r ↔ (addr16)					
	r, !!addr24	5	r ↔ (addr24)					
	A, [saddrp]	2/3	A↔ ((saddrp))					
	A, [%saddrg]	3/4	A ↔ ((saddrg))					
	A, mem	2-5	A ↔ (mem)					

(5) 16-bit data exchange instruction: XCHW

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
XCHW	rp, rp'	2	rp ↔ rp'					
	AX, saddrp2	2	AX ↔ (saddrp2)					
	rp, saddrp	3	rp ↔ (saddrp)					
	rp, sfrp	3	rp ↔ sfrp					
	AX, [saddrp]	3/4	AX ↔ ((saddrp))					
	AX, [%saddrg]	3/4	AX ↔ ((saddrg))					
	AX, !addr16	4	AX ↔ (addr16)					
	AX, !!addr24	5	AX ↔ (addr24)					
	saddrp, saddrp'	4	(saddrp) ↔ (saddrp')					
	AX, mem	2-5	AX ↔ (mem)					

(6) 8-bit arithmetic instructions: ADD, ADDC, SUB, SUBC, CMP, AND, OR, XOR

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADD	A, #byte	2	A, CY ← A + byte	×	×	×	V	×
	r, #byte	3	r, CY ← r + byte	×	×	×	V	×
	saddr, #byte	3/4	(saddr), CY ← (saddr) + byte	×	×	×	V	×
	sfr, #byte	4	sfr, CY ← sfr + byte	×	×	×	V	×
	r, r'	2/3	r, CY ← r + r'	×	×	×	V	×
	A, saddr2	2	A, CY ← A + (saddr2)	×	×	×	V	×
	r, saddr	3	r, CY ← r + (saddr)	×	×	×	V	×
	saddr, r	3	(saddr), CY ← (saddr) + r	×	×	×	V	×
	r, sfr	3	r, CY ← r + sfr	×	×	×	V	×
	sfr, r	3	sfr, CY ← sfr + r	×	×	×	V	×
	saddr, saddr'	4	(saddr), CY ← (saddr) + (saddr')	×	×	×	V	×
	A, [saddrp]	3/4	A, CY ← A + ((saddrp))	×	×	×	V	×
	A, [%saddrg]	3/4	A, CY ← A + ((saddrg))	×	×	×	V	×
	[saddrp], A	3/4	((saddrp)), CY ← ((saddrp)) + A	×	×	×	V	×
	[%saddrg], A	3/4	((saddrg)), CY ← ((saddrg)) + A	×	×	×	V	×
	A, !addr16	4	A, CY ← A + (addr16)	×	×	×	V	×
	A, !!addr24	5	A, CY ← A + (addr24)	×	×	×	V	×
	!addr16, A	4	(addr16), CY ← (addr16) + A	×	×	×	V	×
	!!addr24, A	5	(addr24), CY ← (addr24) + A	×	×	×	V	×
	A, mem	2-5	A, CY ← A + (mem)	×	×	×	V	×
	mem, A	2-5	(mem), CY ← (mem) + A	×	×	×	V	×

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADDC	A, #byte	2	A, CY ← A + byte + CY	×	×	×	V	×
	r, #byte	3	r, CY ← r + byte + CY	×	×	×	V	×
	saddr, #byte	3/4	(saddr), CY ← (saddr) + byte + CY	×	×	×	V	×
	sfr, #byte	4	sfr, CY ← sfr + byte + CY	×	×	×	V	×
	r, r'	2/3	r, CY ← r + r' + CY	×	×	×	V	×
	A, saddr2	2	A, CY ← A + (saddr2) + CY	×	×	×	V	×
	r, saddr	3	r, CY ← r + (saddr) + CY	×	×	×	V	×
	saddr, r	3	(saddr), CY ← (saddr) + r + CY	×	×	×	V	×
	r, sfr	3	r, CY ← r + sfr + CY	×	×	×	V	×
	sfr, r	3	sfr, CY ← sfr + r + CY	×	×	×	V	×
	saddr, saddr'	4	(saddr), CY ← (saddr) + (saddr') + CY	×	×	×	V	×
	A, [saddrp]	3/4	A, CY ← A + ((saddrp)) + CY	×	×	×	V	×
	A, [%saddrg]	3/4	A, CY ← A + ((saddrg)) + CY	×	×	×	V	×
	[saddrp], A	3/4	((saddrp)), CY ← ((saddrp)) + A + CY	×	×	×	V	×
	[%saddrg], A	3/4	((saddrg)), CY ← ((saddrp)) + A + CY	×	×	×	V	×
	A, !addr16	4	A, CY ← A + (addr16) + CY	×	×	×	V	×
	A, !!addr24	5	A, CY ← A + (addr24) +CY	×	×	×	V	×
	!addr16, A	4	(addr16), CY ← (addr16) + A + CY	×	×	×	V	×
	!!addr24, A	5	(addr24), CY ← (addr24) + A + CY	×	×	×	V	×
	A, mem	2-5	A, CY ← A + (mem) + CY	×	×	×	V	×
	mem, A	2-5	(mem), CY ← (mem) + A + CY	×	×	×	V	×

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
SUB	A, #byte	2	A, CY ← A – byte	×	×	×	V	×
	r, #byte	3	r, CY ← r – byte	×	×	×	V	×
	saddr, #byte	3/4	(saddr), CY ← (saddr) – byte	×	×	×	V	×
	sfr, #byte	4	sfr, CY ← sfr – byte	×	×	×	V	×
	r, r'	2/3	r, CY ← r – r'	×	×	×	V	×
	A, saddr2	2	A, CY ← A – (saddr2)	×	×	×	V	×
	r, saddr	3	r, CY ← r – (saddr)	×	×	×	V	×
	saddr, r	3	(saddr), CY ← (saddr) – r	×	×	×	V	×
	r, sfr	3	r, CY ← r – sfr	×	×	×	V	×
	sfr, r	3	sfr, CY ← sfr – r	×	×	×	V	×
	saddr, saddr'	4	(saddr), CY ← (saddr) – (saddr')	×	×	×	V	×
	A, [saddrp]	3/4	A, CY ← A – ((saddrp))	×	×	×	V	×
	A, [%saddrg]	3/4	A, CY ← A – ((saddrg))	×	×	×	V	×
	[saddrp], A	3/4	((saddrp)), CY ← ((saddrp)) – A	×	×	×	V	×
	[%saddrg], A	3/4	((saddrg)), CY ← ((saddrg)) – A	×	×	×	V	×
	A, !addr16	4	A, CY ← A – (addr16)	×	×	×	V	×
	A, !!addr24	5	A, CY ← A – (addr24)	×	×	×	V	×
	!addr16, A	4	(addr16), CY ← (addr16) – A	×	×	×	V	×
	!!addr24, A	5	(addr24), CY ← (addr24) – A	×	×	×	V	×
	A, mem	2-5	A, CY ← A – (mem)	×	×	×	V	×
	mem, A	2-5	(mem), CY ← (mem) – A	×	×	×	V	×

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
SUBC	A, #byte	2	A, CY ← A – byte – CY	×	×	×	V	×
	r, #byte	3	r, CY ← r – byte – CY	×	×	×	V	×
	saddr, #byte	3/4	(saddr), CY ← (saddr) – byte – CY	×	×	×	V	×
	sfr, #byte	4	sfr, CY ← sfr – byte – CY	×	×	×	V	×
	r, r'	2/3	r, CY ← r – r' – CY	×	×	×	V	×
	A, saddr2	2	A, CY ← A – (saddr2) – CY	×	×	×	V	×
	r, saddr	3	r, CY ← r – (saddr) – CY	×	×	×	V	×
	saddr, r	3	(saddr), CY ← (saddr) – r – CY	×	×	×	V	×
	r, sfr	3	r, CY ← r – sfr – CY	×	×	×	V	×
	sfr, r	3	sfr, CY ← sfr – r – CY	×	×	×	V	×
	saddr, saddr'	4	(saddr), CY ← (saddr) – (saddr') – CY	×	×	×	V	×
	A, [saddrp]	3/4	A, CY ← A – ((saddrp)) – CY	×	×	×	V	×
	A, [%saddrg]	3/4	A, CY ← A – ((saddrg)) – CY	×	×	×	V	×
	[saddrp], A	3/4	((saddrp)), CY ← ((saddrp)) – A – CY	×	×	×	V	×
	[%saddrg], A	3/4	((saddrg)), CY ← ((saddrg)) – A – CY	×	×	×	V	×
	A, !addr16	4	A, CY ← A – (addr16) – CY	×	×	×	V	×
	A, !!addr24	5	A, CY ← A – (addr24) – CY	×	×	×	V	×
	!addr16, A	4	(addr16), CY ← (addr16) – A – CY	×	×	×	V	×
	!!addr24, A	5	(addr24), CY ← (addr24) – A – CY	×	×	×	V	×
	A, mem	2-5	A, CY ← A – (mem) – CY	×	×	×	V	×
	mem, A	2-5	(mem), CY ← (mem) – A – CY	×	×	×	V	×

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
CMP	A, #byte	2	A – byte	×	×	×	V	×
	r, #byte	3	r – byte	×	×	×	V	×
	saddr, #byte	3/4	(saddr) – byte	×	×	×	V	×
	sfr, #byte	4	sfr – byte	×	×	×	V	×
	r, r'	2/3	r – r'	×	×	×	V	×
	A, saddr2	2	A – (saddr2)	×	×	×	V	×
	r, saddr	3	r – (saddr)	×	×	×	V	×
	saddr, r	3	(saddr) – r	×	×	×	V	×
	r, sfr	3	r – sfr	×	×	×	V	×
	sfr, r	3	sfr – r	×	×	×	V	×
	saddr, saddr'	4	(saddr) – (saddr')	×	×	×	V	×
	A, [saddrp]	3/4	A – ((saddrp))	×	×	×	V	×
	A, [%saddrg]	3/4	A – ((saddrg))	×	×	×	V	×
	[saddrp], A	3/4	((saddrp)) – A	×	×	×	V	×
	[%saddrg], A	3/4	((saddrg)) – A	×	×	×	V	×
	A, !addr16	4	A – (addr16)	×	×	×	V	×
	A, !!addr24	5	A – (addr24)	×	×	×	V	×
	!addr16, A	4	(addr16) – A	×	×	×	V	×
	!!addr24, A	5	(addr24) – A	×	×	×	V	×
	A, mem	2-5	A – (mem)	×	×	×	V	×
	mem, A	2-5	(mem) – A	×	×	×	V	×

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
AND	A, #byte	2	A ← A∧byte	×	×		P	
	r, #byte	3	r ← r∧byte	×	×		P	
	saddr, #byte	3/4	(saddr) ← (saddr) ∧byte	×	×		P	
	sfr, #byte	4	sfr ← sfr∧byte	×	×		P	
	r, r'	2/3	r ← r∧r'	×	×		P	
	A, saddr2	2	A ← A∧ (saddr2)	×	×		P	
	r, saddr	3	r ← r∧ (saddr)	×	×		P	
	saddr, r	3	(saddr) ← (saddr) ∧r	×	×		P	
	r, sfr	3	r ← r∧sfr	×	×		P	
	sfr, r	3	sfr ← sfr∧r	×	×		P	
	saddr, saddr'	4	(saddr) ← (saddr) ∧ (saddr')	×	×		P	
	A, [saddrp]	3/4	A ← A∧ ((saddrp))	×	×		P	
	A, [%saddrg]	3/4	A ← A∧ ((saddrg))	×	×		P	
	[saddrp], A	3/4	((saddrp)) ← ((saddrp)) ∧A	×	×		P	
	[%saddrg], A	3/4	((saddrg)) ← ((saddrg)) ∧A	×	×		P	
	A, !addr16	4	A ← A∧ (addr16)	×	×		P	
	A, !!addr24	5	A ← A∧ (addr24)	×	×		P	
	!addr16, A	4	(addr16) ← (addr16) ∧A	×	×		P	
	!!addr24, A	5	(addr24) ← (addr24) ∧A	×	×		P	
	A, mem	2-5	A ← A∧(mem)	×	×		P	
	mem, A	2-5	(mem) ← (mem) ∧A	×	×		P	

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
OR	A, #byte	2	A ← A∨byte	×	×		P	
	r, #byte	3	r ← r∨byte	×	×		P	
	saddr, #byte	3/4	(saddr) ← (saddr) ∨byte	×	×		P	
	sfr, #byte	4	sfr ← sfr∨byte	×	×		P	
	r, r'	2/3	r ← r∨r'	×	×		P	
	A, saddr2	2	A ← A∨ (saddr2)	×	×		P	
	r, saddr	3	r ← r∨ (saddr)	×	×		P	
	saddr, r	3	(saddr) ← (saddr) ∨r	×	×		P	
	r, sfr	3	r ← r∨sfr	×	×		P	
	sfr, r	3	sfr ← sfr∨r	×	×		P	
	saddr, saddr'	4	(saddr) ← (saddr) ∨ (saddr')	×	×		P	
	A, [saddrp]	3/4	A ← A∨ ((saddrp))	×	×		P	
	A, [%saddrg]	3/4	A ← A∨ ((saddrg))	×	×		P	
	[saddrp], A	3/4	((saddrp)) ← ((saddrp)) ∨A	×	×		P	
	[%saddrg], A	3/4	((saddrg)) ← ((saddrg)) ∨A	×	×		P	
	A, !addr16	4	A ← A∨ (saddr16)	×	×		P	
	A, !!addr24	5	A ← A∨ (saddr24)	×	×		P	
	!addr16, A	4	(addr16) ← (addr16) ∨A	×	×		P	
	!!addr24, A	5	(addr24) ← (addr24) ∨A	×	×		P	
	A, mem	2-5	A ← A∨ (mem)	×	×		P	
	mem, A	2-5	(mem) ← (mem) ∨A	×	×		P	

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
XOR	A, #byte	2	A ← A ∀ byte	×	×		P	
	r, #byte	3	r ← r ∀ byte	×	×		P	
	saddr, #byte	3/4	(saddr) ← (saddr) ∀ byte	×	×		P	
	sfr, #byte	4	sfr ← sfr ∀ byte	×	×		P	
	r, r'	2/3	r ← r ∀ r'	×	×		P	
	A, saddr2	2	A ← A ∀ (saddr2)	×	×		P	
	r, saddr	3	r ← r ∀ (saddr)	×	×		P	
	saddr, r	3	(saddr) ← (saddr) ∀ r	×	×		P	
	r, sfr	3	r ← r ∀ sfr	×	×		P	
	sfr, r	3	sfr ← sfr ∀ r	×	×		P	
	saddr, saddr'	4	(saddr) ← (saddr) ∀ (saddr')	×	×		P	
	A, [saddrp]	3/4	A ← A ∀ ((saddrp))	×	×		P	
	A, [%saddrg]	3/4	A ← A ∀ ((saddrg))	×	×		P	
	[saddrp], A	3/4	((saddrp)) ← ((saddrp)) ∀ A	×	×		P	
	[%saddrg], A	3/4	((saddrg)) ← ((saddrg)) ∀ A	×	×		P	
	A, !addr16	4	A ← A ∀ (addr16)	×	×		P	
	A, !!addr24	5	A ← A ∀ (addr24)	×	×		P	
	!addr16, A	4	(addr16) ← (addr16) ∀ A	×	×		P	
	!!addr24, A	5	(addr24) ← (addr24) ∀ A	×	×		P	
	A, mem	2-5	A ← A ∀ (mem)	×	×		P	
	mem, A	2-5	(mem) ← (mem) ∀ A	×	×		P	

(7) 16-bit arithmetic instructions: ADDW, SUBW, CMPW

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADDW	AX, #word	3	AX, CY ← AX + word	×	×	×	V	×
	rp, #word	4	rp, CY ← rp + word	×	×	×	V	×
	rp, rp'	2	rp, CY ← rp + rp'	×	×	×	V	×
	AX, saddrp2	2	AX, CY ← AX + (saddrp2)	×	×	×	V	×
	rp, saddrp	3	rp, CY ← rp + (saddrp)	×	×	×	V	×
	saddrp, rp	3	(saddrp), CY ← (saddrp) + rp	×	×	×	V	×
	rp, sfrp	3	rp, CY ← rp + sfrp	×	×	×	V	×
	sfrp, rp	3	sfrp, CY ← sfrp + rp	×	×	×	V	×
	saddrp, #word	4/5	(saddrp), CY ← (saddrp) + word	×	×	×	V	×
	sfrp, #word	5	sfrp, CY ← sfrp + word	×	×	×	V	×
	saddrp, saddrp'	4	(saddrp), CY ← (saddrp) + (saddrp')	×	×	×	V	×
SUBW	AX, #word	3	AX, CY ← AX – word	×	×	×	V	×
	rp, #word	4	rp, CY ← rp – word	×	×	×	V	×
	rp, rp'	2	rp, CY ← rp – rp'	×	×	×	V	×
	AX, saddrp2	2	AX, CY ← AX – (saddrp2)	×	×	×	V	×
	rp, saddrp	3	rp, CY ← rp – (saddrp)	×	×	×	V	×
	saddrp, rp	3	(saddrp), CY ← (saddrp) – rp	×	×	×	V	×
	rp, sfrp	3	rp, CY ← rp – sfrp	×	×	×	V	×
	sfrp, rp	3	sfrp, CY ← sfrp – rp	×	×	×	V	×
	saddrp, #word	4/5	(saddrp), CY ← (saddrp) – word	×	×	×	V	×
	sfrp, #word	5	sfrp, CY ← sfrp – word	×	×	×	V	×
	saddrp, saddrp'	4	(saddrp), CY ← (saddrp) – (saddrp')	×	×	×	V	×
CMPW	AX, #word	3	AX – word	×	×	×	V	×
	rp, #word	4	rp – word	×	×	×	V	×
	rp, rp'	2	rp – rp'	×	×	×	V	×
	AX, saddrp2	2	AX – (saddrp2)	×	×	×	V	×
	rp, saddrp	3	rp – (saddrp)	×	×	×	V	×
	saddrp, rp	3	(saddrp) – rp	×	×	×	V	×
	rp, sfrp	3	rp – sfrp	×	×	×	V	×
	sfrp, rp	3	sfrp – rp	×	×	×	V	×
	saddrp, #word	4/5	(saddrp) – word	×	×	×	V	×
	sfrp, #word	5	sfrp – word	×	×	×	V	×
	saddrp, saddrp'	4	(saddrp) – (saddrp')	×	×	×	V	×

(8) 24-bit arithmetic instructions: ADDG, SUBG

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADDG	rg, rg'	2	rg, CY ← rg + rg'	×	×	×	V	×
	rg, #imm24	5	rg, CY ← rg + imm24	×	×	×	V	×
	WHL, saddrg	3	WHL, CY ← WHL + (saddrg)	×	×	×	V	×
SUBG	rg, rg'	2	rg, CY ← rg – rg'	×	×	×	V	×
	rg, #imm24	5	rg, CY ← rg – imm24	×	×	×	V	×
	WHL, saddrg	3	WHL, CY ← WHL – (saddrg)	×	×	×	V	×

(9) Multiplicative instructions: MULU, MULUW, MULW, DIVUW, DIVUX

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MULU	r	2/3	AX ← AXr					
MULUW	rp	2	AX (high order), rp (low order) ← AXXrp					
MULW	rp	2	AX (high order), rp (low order) ← AXXrp					
DIVUW	r	2/3	AX (quotient), r (remainder) ← AX ÷ r**Note 1**					
DIVUX	rp	2	AXDE (quotient), rp (remainder) ← AXDE ÷ rp**Note 2**					

Notes 1. When r = 0, r ← X, AX ← FFFFH

2. When rp = 0, rp ← DE, AXDE ← FFFFFFFFH

(10) Special arithmetic instructions: MACW, MACSW, SACW

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MACW	byte	3	AXDE ← (B) X (C) + AXDE, B ← B + 2, C ← C + 2, byte ← byte – 1 End if (byte = 0 or P/V = 1)	×	×	×	V	×
MACSW	byte	3	AXDE ← (B) X (C) + AXDE, B ← B + 2, C ← C + 2, byte ← byte – 1 if byte = 0 then End if P/V = 1 then if overflow AXDE ← 7FFFFFFFH, End if underflow AXDE ← 80000000H, End	×	×	×	V	×
SACW	[TDE+], [WHL+]	4	AX ← | (TDE) – (WHL) | + AX, TDE ← TDE + 2, WHL ← WHL + 2 C ← C – 1 End if (C = 0 or CY = 1)	×	×	×	V	×

(11) Increment and decrement instructions: INC, DEC, INCW, DECW, INCG, DECG

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
INC	r	1/2	r ← r + 1	×	×	×	V	
	saddr	2/3	(saddr) ← (saddr) + 1	×	×	×	V	
DEC	r	1/2	r ← r – 1	×	×	×	V	
	saddr	2/3	(saddr) ← (saddr) – 1	×	×	×	V	
INCW	rp	2/1	rp ← rp + 1					
	saddrp	3/4	(saddrp) ← (saddrp) + 1					
DECW	rp	2/1	rp ← rp – 1					
	saddrp	3/4	(saddrp) ← (saddrp) – 1					
INCG	rg	2	rg ← rg + 1					
DECG	rg	2	rg ← rg –1					

(12) Decimal adjust instructions: ADJBA, ADJBS, CVTBW

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADJBA		2	Decimal Adjust Accumulator after Addition	×	×	×	P	×
ADJBS		2	Decimal Adjust Accumulator after Subtract	×	×	×	P	×
CVTBW		1	X ← A, A ← 00H if A_7 = 0 X ← A, A ← FFH if A_7 = 1					

(13) Shift and rotate instructions: ROR, ROL, RORC, ROLC, SHR, SHL, SHRW, SHLW, ROR4, ROL4

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ROR	r, n	2/3	(CY, $r_7 \leftarrow r_0$, $r_{m-1} \leftarrow r_m$) × n n = 0 to 7				P	×
ROL	r, n	2/3	(CY, $r_0 \leftarrow r_7$, $r_{m+1} \leftarrow r_m$) × n n = 0 to 7				P	×
RORC	r, n	2/3	(CY $\leftarrow r_0$, $r_7 \leftarrow$ CY, $r_{m-1} \leftarrow r_m$) × n n = 0 to 7				P	×
ROLC	r, n	2/3	(CY $\leftarrow r_7$, $r_0 \leftarrow$ CY, $r_{m+1} \leftarrow r_m$) × n n = 0 to 7				P	×
SHR	r, n	2/3	(CY $\leftarrow r_0$, $r_7 \leftarrow 0$, $r_{m-1} \leftarrow r_m$) × n n = 0 to 7	×	×	0	P	×
SHL	r, n	2/3	(CY $\leftarrow r_7$, $r_0 \leftarrow 0$, $r_{m+1} \leftarrow r_m$) × n n = 0 to 7	×	×	0	P	×
SHRW	rp, n	2	(CY $\leftarrow rp_0$, $rp_{15} \leftarrow 0$, $rp_{m-1} \leftarrow rp_m$) × n n = 0 to 7	×	×	0	P	×
SHLW	rp, n	2	(CY $\leftarrow rp_{15}$, $rp_0 \leftarrow 0$, $rp_{m+1} \leftarrow rp_m$) × n n = 0 to 7	×	×	0	P	×
ROR4	mem3	2	$A_{3-0} \leftarrow (mem3)_{3-0}$, $(mem3)_{7-4} \leftarrow A_{3-0}$, $(mem3)_{3-0} \leftarrow (mem3)_{7-4}$					
ROL4	mem3	2	$A_{3-0} \leftarrow (mem3)_{7-4}$, $(mem3)_{3-0} \leftarrow A_{3-0}$, $(mem3)_{7-4} \leftarrow (mem3)_{3-0}$					

(14) Bit manipulation instructions: MOV1, AND1, OR1, XOR1, NOT1, SET1, CLR1

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOV1	CY, saddr.bit	3/4	CY ← (saddr.bit)					×
	CY, sfr.bit	3	CY ← sfr.bit					×
	CY, X.bit	2	CY ← X.bit					×
	CY, A.bit	2	CY ← A.bit					×
	CY, PSWL.bit	2	CY ← PSW_L.bit					×
	CY, PSWH.bit	2	CY ← PSW_H.bit					×
	CY, !addr16.bit	5	CY ← !addr16.bit					×
	CY, !!addr24.bit	2	CY ← !!addr24.bit					×
	CY, mem2.bit	2	CY ← mem2.bit					×
	saddr.bit, CY	3/4	(saddr.bit) ← CY					
	sfr.bit, CY	3	sfr.bit ← CY					
	X.bit, CY	2	X.bit ← CY					
	A.bit, CY	2	A.bit ← CY					
	PSWL.bit, CY	2	PSW_L.bit ← CY	×	×	×	×	×
	PSWH.bit, CY	2	PSW_H.bit ← CY					
	!addr16.bit, CY	5	!addr16.bit ← CY					
	!!addr24.bit, CY	6	!!addr24.bit ← CY					
	mem2.bit, CY	2	mem2.bit ← CY					

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
AND1	CY, saddr.bit	3/4	CY ← CY ∧ (saddr.bit)					×
	CY, /saddr.bit	3/4	CY ← CY ∧ $\overline{(saddr.bit)}$					×
	CY, sfr.bit	3	CY ← CY ∧ sfr.bit					×
	CY, /sfr.bit	3	CY ← CY ∧ $\overline{sfr.bit}$					×
	CY, X.bit	2	CY ← CY ∧ X.bit					×
	CY, /X.bit	2	CY ← CY ∧ $\overline{X.bit}$					×
	CY, A.bit	2	CY ← CY ∧ A.bit					×
	CY, /A.bit	2	CY ← CY ∧ $\overline{A.bit}$					×
	CY, PSWL.bit	2	CY ← CY ∧ PSW_L.bit					×
	CY, /PSWL.bit	2	CY ← CY ∧ $\overline{PSW_L.bit}$					×
	CY, PSWH.bit	2	CY ← CY ∧ PSW_H.bit					×
	CY, /PSWH.bit	2	CY ← CY ∧ $\overline{PSW_H.bit}$					×
	CY, !addr16.bit	5	CY ← CY ∧ !addr16.bit					×
	CY, /!addr16.bit	5	CY ← CY ∧ $\overline{!addr16.bit}$					×
	CY, !!addr24.bit	2	CY ← CY ∧ !!addr24.bit					×
	CY, /!!addr24.bit	6	CY ← CY ∧ $\overline{!!addr24.bit}$					×
	CY, mem2.bit	2	CY ← CY ∧ mem2.bit					×
	CY, /mem2.bit	2	CY ← CY ∧ $\overline{mem2.bit}$					×
OR1	CY, saddr.bit	3/4	CY ← CY ∨ (saddr.bit)					×
	CY, /saddr.bit	3/4	CY ← CY ∨ $\overline{(saddr.bit)}$					×
	CY, sfr.bit	3	CY ← CY ∨ sfr.bit					×
	CY, /sfr.bit	3	CY ← CY ∨ $\overline{sfr.bit}$					×
	CY, X.bit	2	CY ← CY ∨ X.bit					×
	CY, /X.bit	2	CY ← CY ∨ $\overline{X.bit}$					×
	CY, A.bit	2	CY ← CY ∨ A.bit					×
	CY, /A.bit	2	CY ← CY ∨ $\overline{A.bit}$					×
	CY, PSWL.bit	2	CY ← CY ∨ PSW_L.bit					×
	CY, /PSWL.bit	2	CY ← CY ∨ $\overline{PSW_L.bit}$					×
	CY, PSWH.bit	2	CY ← CY ∨ PSW_H.bit					×
	CY, /PSWH.bit	2	CY ← CY ∨ $\overline{PSW_H.bit}$					×
	CY, !addr16.bit	5	CY ← CY ∨ !addr16.bit					×
	CY, /!addr16.bit	5	CY ← CY ∨ $\overline{!addr16.bit}$					×
	CY, !!addr24.bit	2	CY ← CY ∨ !!addr24.bit					×
	CY, /!!addr24.bit	6	CY ← CY ∨ $\overline{!!addr24.bit}$					×
	CY, mem2.bit	2	CY ← CY ∨ mem2.bit					×
	CY, /mem2.bit	2	CY ← CY ∨ $\overline{mem2.bit}$					×

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
XOR1	CY, saddr.bit	3/4	CY ← CY ∀ (saddr.bit)					×
	CY, sfr.bit	3	CY ← CY ∀ sfr.bit					×
	CY, X.bit	2	CY ← CY ∀ X.bit					×
	CY, A.bit	2	CY ← CY ∀ A.bit					×
	CY, PSWL.bit	2	CY ← CY ∀ PSW_L.bit					×
	CY, PSWH.bit	2	CY ← CY ∀ PSW_H.bit					×
	CY, !addr16.bit	5	CY ← CY ∀ !addr16.bit					×
	CY, !!addr24.bit	2	CY ← CY ∀ !!addr24.bit					×
	CY, mem2.bit	2	CY ← CY ∀ mem2.bit					×
NOT1	saddr.bit	3/4	(saddr.bit) ← $\overline{(saddr.bit)}$					
	sfr.bit	3	sfr.bit ← $\overline{sfr.bit}$					
	X.bit	2	X.bit ← $\overline{X.bit}$					
	A.bit	2	A.bit ← $\overline{A.bit}$					
	PSWL.bit	2	PSWL.bit ← $\overline{PSW_L.bit}$	×	×	×	×	×
	PSWH.bit	2	PSWH.bit ← $\overline{PSW_H.bit}$					
	!addr16.bit	5	!addr16.bit ← $\overline{!addr16.bit}$					
	!!addr24.bit	2	!!addr24.bit ← $\overline{!!addr24.bit}$					
	mem2.bit	2	mem2.bit ← $\overline{mem2.bit}$					
	CY	1	CY ← $\overline{CY}$					×
SET1	saddr.bit	2/3	(saddr.bit) ← 1					
	sfr.bit	3	sfr.bit ← 1					
	X.bit	2	X.bit ← 1					
	A.bit	2	A.bit ← 1					
	PSWL.bit	2	PSW_L.bit ← 1	×	×	×	×	×
	PSWH.bit	2	PSW_H.bit ← 1					
	!addr16.bit	5	!addr16.bit ← 1					
	!!addr24.bit	2	!!addr24.bit ← 1					
	mem2.bit	2	mem2.bit ←1					
	CY	1	CY ← 1					1
CLR1	saddr.bit	2/3	(saddr.bit) ← 0					
	sfr.bit	3	sfr.bit ← 0					
	X.bit	2	X.bit ← 0					
	A.bit	2	A.bit ← 0					
	PSWL.bit	2	PSW_L.bit ← 0	×	×	×	×	×
	PSWH.bit	2	PSW_H.bit ← 0					
	!addr16.bit	5	!addr16.bit ← 0					
	!!addr24.bit	2	!!addr24.bit ← 0					
	mem2.bit	2	mem2.bit ←0					
	CY	1	CY ← 0					0

(15) Stack manipulation instructions: PUSH, PUSHU, POP, POPU, MOVG, ADDWG, SUBWG, INCG, DECG

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
PUSH	PSW	1	(SP – 2) ← PSW, SP ← SP – 2					
	sfrp	3	(SP – 2) ← sfrp, SP ← SP – 2					
	sfr	3	(SP – 1) ← sfr, SP ← SP – 1					
	post	2	{(SP – 2) ← post, SP ← SP – 2} × m**Note**					
	rg	2	(SP – 3) ← rg, SP ← SP – 3					
PUSHU	post	2	{(UUP – 2) ← post, UUP ← UUP – 2} × m**Note**					
POP	PSW	1	PSW ← (SP), SP ← SP + 2	R	R	R	R	R
	sfrp	3	sfrp ← (SP), SP ← SP + 2					
	sfr	3	sfr ← (SP), SP ← SP + 1					
	post	2	{post ← (SP), SP ← SP + 2} × m**Note**					
	rg	2	rg ← (SP), SP ← SP + 3					
POPU	post	2	{post ← (UUP), UUP ← UUP + 2} × m**Note**					
MOVG	SP, #imm24	5	SP ← imm24					
	SP, WHL	2	SP ← WHL					
	WHL, SP	2	WHL ← SP					
ADDWG	SP, #word	4	SP ← SP + word					
SUBWG	SP, #word	4	SP ← SP – word					
INCG	SP	2	SP ← SP + 1					
DECG	SP	2	SP ← SP – 1					

Note m is the number of registers specified by post.

(16) Call return instructions: CALL, CALLF, CALLT, BRK, BRKCS, RET, RETI, RETB, RETCS, RETCSB

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
CALL	!addr16	3	(SP – 3) ← (PC + 3), SP ← SP – 3, PC_{HW} ← 0, PC_{LW} ← addr16					
	!!addr20	4	(SP – 3) ← (PC + 4), SP ← SP – 3, PC ← addr20					
	rp	2	(SP – 3) ← (PC + 2), SP ← SP – 3, PC_{HW} ← 0, PC_{LW} ← rp					
	rg	2	(SP – 3) ← (PC + 2), SP ← SP – 3, PC ← rg					
	[rp]	2	(SP – 3) ← (PC + 2), SP ← SP – 3, PC_{HW} ← 0, PC_{LW} ← (rp)					
	[rg]	2	(SP – 3) ← (PC + 2), SP ← SP – 3, PC ← (rg)					
	$!addr20	3	(SP – 3) ← (PC + 3), SP ← SP – 3, PC ← PC + 3 + jdisp16					
CALLF	!addr11	2	(SP – 3) ← (PC + 2), SP ← SP – 3 $PC_{19–12}$ ← 0, PC_{11} ← 1, $PC_{10–0}$ ← addr11					
CALLT	[addr5]	1	(SP – 3) ← (PC + 1), SP ← SP – 3 PC_{HW} ← 0, PC_{LW} ← (addr5)					
BRK		1	(SP – 2) ← PSW, $(SP – 1)_{0–3}$ ←, $(PC + 1)_{HW}$, $(SP – 4)$ ← $(PC + 1)_{LW}$, SP ← SP – 4 PC_{HW} ← 0, PC_{LW} ← (003EH)					
BRKCS	RBn	2	PC_{LW} ← RP2, RP3 ← PSW, RBS2 – 0 ← n, RSS ← 0, IE ← 0, $RP3_{8–11}$ ← PC_{HW}, PC_{HW} ← 0					
RET		1	PC ← (SP), SP ← SP + 3					
RETI		1	PC_{LW} ← (SP), PC_{HW} ← $(SP + 3)_{0–3}$, PSW ← (SP + 2), SP ← SP + 4 The flag with the highest priority that is set to one in the ISPR is cleared to 0.	R	R	R	R	R
RETB		1	PC_{LW} ← (SP), PC_{HW} ← $(SP + 3)_{0–3}$, PSW ← (SP + 2), SP ← SP + 4	R	R	R	R	R
RETCS	!addr16	3	PSW ← RP3, PC_{LW} ← RP2, RP2 ← addr16, PC_{HW} ← $RP3_{8–11}$ The flag with the highest priority that is set to one in the ISPR is cleared to 0.	R	R	R	R	R
RETCSB	!addr16	4	PSW ← RP3, PC_{LW} ← RP2, RP2 ← addr16, PC_{HW} ← $RP3_{8–11}$	R	R	R	R	R

(17) Unconditional branch instruction: BR

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
BR	!addr16	3	$PC_{HW} \leftarrow 0$, $PC_{LW} \leftarrow$ addr16					
	!!addr20	4	PC ← addr20					
	rp	2	$PC_{HW} \leftarrow 0$, $PC_{LW} \leftarrow$ rp					
	rg	2	PC ← rg					
	[rp]	2	$PC_{HW} \leftarrow 0$, $PC_{LW} \leftarrow$ (rp)					
	[rg]	2	PC ← (rg)					
	$addr20	2	PC ← PC + 2 + jdisp8					
	$!addr20	3	PC ← PC + 3 + jdisp16					

(18) Conditional branch instructions: BNZ, BNE, BZ, BE, BNC, BNL, BC, BL, BNV, BPO, BV, BPE, BP, BN, BLT, BGE, BLE, BGT, BNH, BH, BF, BT, BTCLR, BFSET, DBNZ

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
BNZ	$addr20	2	PC ← PC + 2 + jdisp8 if Z = 0					
BNE								
BZ	$addr20	2	PC ← PC + 2 + jdisp8 if Z = 1					
BE								
BNC	$addr20	2	PC ← PC + 2 + jdisp8 if CY = 0					
BNL								
BC	$addr20	2	PC ← PC + 2 + jdisp8 if CY = 1					
BL								
BNV	$addr20	2	PC ← PC + 2 + jdisp8 if P/V = 0					
BPO								
BV	$addr20	2	PC ← PC + 2 + jdisp8 if P/V = 1					
BPE								
BP	$addr20	2	PC ← PC + 2 + jdisp8 if S = 0					
BN	$addr20	2	PC ← PC + 2 + jdisp8 if S = 1					
BLT	$addr20	3	PC ← PC + 3 + jdisp8 if P/V ⊻ S = 1					
BGE	$addr20	3	PC ← PC + 3 + jidsp8 if P/V ⊻ S = 0					
BLE	$addr20	3	PC ← PC + 3 + jdisp8 if (P/V ⊻ S) ∨ Z = 1					
BGT	$addr20	3	PC ← PC + 3 + jidsp8 if (P/V ⊻ S) ∨ Z = 0					
BNH	$addr20	3	PC ← PC + 3 + jdisp8 if Z∨CY = 1					
BH	$addr20	3	PC ← PC + 3 + jidsp8 if Z∨CY = 0					
BF	saddr.bit, $addr20	4/5	PC ← PC + 4[Note] + jdisp8 if (saddr.bit) = 0					
	sfr.bit, $addr20	4	PC ← PC + 4 + jdisp8 if sfr.bit = 0					
	X.bit, $addr20	3	PC ← PC + 3 + jdisp8 if X.bit = 0					
	A.bit, $addr20	3	PC ← PC + 3 + jdisp8 if A.bit = 0					
	PSWL.bit, $addr20	3	PC ← PC + 3 + jdisp8 if PSW_L.bit = 0					
	PSWH.bit, $addr20	3	PC ← PC + 3 + jdisp8 if PSW_H.bit = 0					
	!addr16.bit, $addr20	6	PC ← PC + 3 + jdisp8 if !addr16.bit = 0					
	!!addr24.bit, $addr20	3	PC ← PC + 3 + jdisp8 if !!addr24.bit = 0					
	mem2.bit, $addr20	3	PC ← PC + 3 + jdisp8 if mem2.bit = 0					

Note This is used when the number of bytes is four. When five, it becomes PC ← PC + 5 + jdisp8.

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
BT	saddr.bit, $addr20	3/4	PC ← PC + 3[Note 1] + jdisp8 if (saddr.bit) = 1					
	sfr.bit, $addr20	4	PC ← PC + 4 + jdisp8 if sfr.bit = 1					
	X.bit, $addr20	3	PC ← PC + 3 + jdisp8 if X.bit = 1					
	A.bit, $addr20	3	PC ← PC + 3 + jdisp8 if A.bit = 1					
	PSWL.bit, $addr20	3	PC ← PC + 3 + jdisp8 if PSW_L.bit = 1					
	PSWH.bit, $addr20	3	PC ← PC + 3 + jdisp8 if PSW_H.bit = 1					
	!addr16.bit, $addr20	6	PC ← PC + 3 + jdisp8 if !addr16.bit = 1					
	!!addr24.bit, $addr20	3	PC ← PC + 3 + jdisp8 if !!addr24.bit = 1					
	mem2.bit, $addr20	3	PC ← PC + 3 + jdisp8 if mem2.bit = 1					
BTCLR	saddr.bit, $addr20	4/5	{PC ← PC + 4[Note 2] + jdisp8, (saddr.bit) ← 0} if (saddr.bit = 1)					
	sfr.bit, $addr20	4	{PC ← PC + 4 + jdisp8, sfr.bit ← 0} if sfr. bit = 1					
	X.bit, $addr20	3	{PC ← PC + 3 + jdisp8, X.bit ← 0} if X.bit = 1					
	A.bit, $addr20	3	{PC ← PC + 3 + jdisp8, A.bit ← 0} if A.bit = 1					
	PSWL.bit, $addr20	3	{PC ← PC + 3 + jdisp8, PSW_L.bit ← 0} if PSW_L.bit = 1	×	×	×	×	×
	PSWH.bit, $addr20	3	{PC ← PC + 3 + jdisp8, PSW_H.bit ← 0} if PSW_H.bit = 1					
	!addr16.bit, $addr20	6	{PC ← PC + 3 + jdisp8, !addr16.bit ← 0} if !addr16.bit = 1					
	!!addr24.bit, $addr20	3	{PC ← PC + 3 + jdisp8, !!addr24.bit ← 0} if !!addr24.bit = 1					
	mem2.bit, $addr20	3	{PC ← PC + 3 + jdisp8, mem2.bit ← 0} if mem2.bit = 1					

Notes 1. This is used when the number of bytes is three. When four, it becomes PC ← PC + 4 + jdisp8.

2. This is used when the number of bytes is four. When five, it becomes PC ← PC + 5 + jdisp8.

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
BFSET	saddr.bit, $addr20	4/5	{PC ← PC + 4[Note 2] + jdisp8, (saddr.bit) ← 1} if (saddr.bit = 0)					
	sfr.bit, $addr20	4	{PC ← PC + 4 + jdisp8, sfr.bit ← 1} if sfr. bit = 0					
	X.bit, $addr20	3	{PC ← PC + 3 + jdisp8, X.bit ← 1} if X.bit = 0					
	A.bit, $addr20	3	{PC ← PC + 3 + jdisp8, A.bit ← 1} if A.bit = 0					
	PSWL.bit, $addr20	3	{PC ← PC + 3 + jdisp8, PSW_L.bit ← 1} if PSW_L.bit = 0	×	×	×	×	×
	PSWH.bit, $addr20	3	{PC ← PC + 3 + jdisp8, PSW_H.bit ← 1} if PSW_H.bit = 0					
	!addr16.bit, $addr20	6	{PC ← PC + 3 + jdisp8, !addr16.bit ← 1} if !addr16.bit = 0					
	!!addr24.bit, $addr20	3	{PC ← PC + 3 + jdisp8, !!addr24.bit ← 1} if !!addr24.bit = 0					
	mem2.bit, $addr20	3	{PC ← PC + 3 + jdisp8, mem2.bit ← 1} if mem2.bit = 0					
DBNZ	B, $addr20	2	B ← B – 1, PC ← PC + 2 + jdisp8 if B ≠ 0					
	C. $addr20	2	C ← C – 1, PC ← PC + 2 + jdisp8 if C ≠ 0					
	saddr, $addr20	3/4	(saddr) ← (saddr) – 1, PC ← PC + 3[Note 1] + jdisp8 if (saddr) ≠ 0					

Notes 1. This is used when the number of bytes is three. When four, it becomes PC ← PC + 4 + jdisp8.

2. This is used when the number of bytes is four. When five, it becomes PC ← PC + 5 + jdisp8.

(19) CPU control instructions: MOV, LOCATION, SEL, SWRS, NOP, EI, DI

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOV	STBC, #byte	4	STBC ← byte					
	WDM, #byte	4	WDM ← byte					
LOCATION	locaddr	4	Specification of the high-order word of the location address of the SFR and internal data area					
SEL	RBn	2	RSS ← 0, RBS2 – 0 ← n					
	RBn, ALT	2	RSS ← 1, RBS2 – 0 ← n					
SWRS		2	RSS ← $\overline{RSS}$					
NOP		1	No operation					
EI		1	IE ← 1 (Enable interrupt)					
DI		1	IE ← 0 (Disable interrupt)					

(20) String instructions: MOVTBLW, MOVM, XCHM, MOVBK, XCHBK, CMPME, CMPMNE, CMPMC, CMPMNC, CMPBKE, CMPBKNE, CMPBKC, CMPBKNC

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOVTBLW	!addr8, byte	4	(addr8 + 2) ← (addr8), byte ← byte – 1, addr8 ← addr8 – 2 End if byte = 0					
MOVM	[TDE+], A	2	(TDE) ← A, TDE ← TDE + 1, C ← C – 1 End if C = 0					
	[TDE–], A	2	(TDE) ← A, TDE ← TDE – 1, C ← C – 1 End if C = 0					
XCHM	[TDE+], A	2	(TDE) ↔ A, TDE ← TDE + 1, C ← C – 1 End if C = 0					
	[TDE–], A	2	(TDE) ↔ A, TDE ← TDE – 1, C ← C – 1 End if C = 0					
MOVBK	[TDE+], [WHL+]	2	(TDE) ← (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C –1 End if C = 0					
	[TDE–], [WHL–]	2	(TDE) ← (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C –1 End if C = 0					
XCHBK	[TDE+], [WHL+]	2	(TDE) ↔ (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C –1 End if C = 0					
	[TDE–], [WHL–]	2	(TDE) ↔ (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C –1 End if C = 0					
CMPME	[TDE+], A	2	(TDE) – A, TDE ← TDE + 1, C ← C – 1 End if C = 0 or Z = 0	×	×	×	V	×
	[TDE–], A	2	(TDE) – A, TDE ← TDE – 1, C ← C – 1 End if C = 0 or Z = 0	×	×	×	V	×
CMPMNE	[TDE+], A	2	(TDE) – A, TDE ← TDE + 1, C ← C – 1 End if C = 0 or Z = 1	×	×	×	V	×
	[TDE–], A	2	(TDE) – A, TDE ← TDE – 1, C ← C – 1 End if C = 0 or Z = 1	×	×	×	V	×
CMPMC	[TDE+], A	2	(TDE) – A, TDE ← TDE + 1, C ← C – 1 End if C = 0 or CY = 0	×	×	×	V	×
	[TDE–], A	2	(TDE) – A, TDE ← TDE – 1, C ← C – 1 End if C = 0 or CY = 0	×	×	×	V	×
CMPMNC	[TDE+], A	2	(TDE) – A, TDE ← TDE + 1, C ← C – 1 End if C = 0 or CY = 1	×	×	×	V	×
	[TDE–], A	2	(TDE) – A, TDE ← TDE – 1, C ← C – 1 End if C = 0 or CY = 1	×	×	×	V	×
CMPBKE	[TDE+], [WHL+]	2	(TDE) – (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C –1 End if C = 0 or Z = 0	×	×	×	V	×
	[TDE–], [WHL–]	2	(TDE) – (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C –1 End if C = 0 or Z = 0	×	×	×	V	×
CMPBKNE	[TDE+], [WHL+]	2	(TDE) – (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C –1 End if C = 0 or Z = 1	×	×	×	V	×
	[TDE–], [WHL–]	2	(TDE) – (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C –1 End if C = 0 or Z = 1	×	×	×	V	×
CMPBKC	[TDE+], [WHL+]	2	(TDE) – (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C –1 End if C = 0 or CY = 0	×	×	×	V	×
	[TDE–], [WHL–]	2	(TDE) – (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C –1 End if C = 0 or CY = 0	×	×	×	V	×
CMPBKNC	[TDE+], [WHL+]	2	(TDE) – (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C –1 End if C = 0 or CY = 1	×	×	×	V	×
	[TDE–], [WHL–]	2	(TDE) – (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C –1 End if C = 0 or CY = 1	×	×	×	V	×

28.3 Lists of Addressing Instructions

(1) 8-bit instructions (The values enclosed by parentheses are combined to express the A description as r.)

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, MOVM, XCHM, CMPME, CMPMNE, CMPMNC, CMPMC, MOVBK, XCHBK, CMPBKE, CMPBKNE, CMPBKNC, CMPBKC

Table 28-1. 8-Bit Addressing Instructions

First operand \ Second operand	#byte	A	r r'	saddr saddr'	sfr	!addr16 !!addr24	mem [saddrp] [%saddrg]	r3 PSWL PSWH	[WHL+] [WHL–]	n	None[Note 2]
A	(MOV) ADD[Note 1]	(MOV) (XCH) (ADD)[Note 1]	MOV XCH (ADD)[Note 1]	(MOV)[Note 6] (XCH)[Note 6] (ADD)[Notes 1, 6]	MOV (XCH) (ADD)[Note 1]	(MOV) (XCH) ADD[Note 1]	MOV XCH ADD[Note 1]	MOV	(MOV) (XCH) (ADD)[Note 1]		
r	MOV ADD[Note 1]	(MOV) (XCH) (ADD)[Note 1]	MOV XCH ADD[Note 1]	MOV XCH ADD[Note 1]	MOV XCH ADD[Note 1]	MOV XCH				ROR[Note 3]	MULU DIVUW INC DEC
saddr	MOV ADD[Note 1]	(MOV)[Note 6] (ADD)[Note 1]	MOV ADD[Note 1]	MOV XCH ADD[Note 1]							INC DEC DBNZ
sfr	MOV ADD[Note 1]	MOV (ADD)[Note 1]	MOV ADD[Note 1]								PUSH POP
!addr16 !!addr24	MOV	MOV ADD[Note 1]	MOV								
mem [saddrp] [%saddrg]		MOV ADD[Note 1]									
mem3											ROR4 ROL4
r3 PSWL PSWH	MOV	MOV									
B, C											DBNZ
STBC, WDM	MOV										
[TDE+] [TDE–]		(MOV) (ADD)[Note 1] MOVM[Note 4]							MOVBK[Note 5]		

Notes 1. ADDC, SUB, SUBC, AND, OR, XOR, and CMP are identical to ADD.

2. There is no second operand, or the second operand is not an operand address.

3. ROL, RORC, ROLC, SHR, and SHL are identical to ROR.

4. XCHM, CMPME, CMPMNE, CMPMNC, and CMPMC are identical to MOVM.

5. XCHBK, CMPBKE, CMPBKNE, CMPBKNC, and CMPBKC are identical to MOVBK.

6. When saddr is saddr2 in this combination, the instruction has a short code length.

(2) 16-bit instructions (The values enclosed by parentheses are combined to express AX description as rp.)

MOVM, XCHW, ADDW, SUBW, CMPW, MULUW, MULW, DIVUX, INCW, DECW, SHRW, SHLW, PUSH, POP, ADDWG, SUBWG, PUSHU, POPU, MOVTBLW, MACW, MACSW, SACW

Table 28-2. 16-Bit Addressing Instructions

First operand \ Second operand	#word	AX	rp rp'	saddrp saddrp'	sfrp	!addr16 !!addr24	mem [saddrp] [%saddrg]	[WHL+]	byte	n	None[Note 2]
AX	(MOVW) ADDW[Note 1]	(MOVW) (XCHW) (ADD)[Note 1]	(MOVW) (XCHW) (ADDW)[Note 1]	(MOVW)[Note 3] (XCHW)[Note 3] (ADDW)[Notes 1, 3]	MOVW (XCHW) (ADDW)[Note 1]	(MOVW) XCHW	MOVW XCHW	(MOVW) (XCHW)			
rp	MOVW ADDW[Note 1]	(MOVW) (XCHW) (ADDW)[Note 1]	MOVW XCHW ADDW[Note 1]	MOVW XCHW ADDW[Note 1]	MOVW XCHW ADDW[Note 1]	MOVW				SHRW SHLW	MULW[Note 4] INCW DECW
saddrp	MOVW ADDW[Note 1]	(MOVW)[Note 3] (ADDW)[Note 1]	MOVW ADDW[Note 1]	MOVW XCHW ADDW[Note 1]							INCW DECW
sfrp	MOVW ADDW[Note 1]	MOVW (ADDW)[Note 1]	MOVW (ADDW)[Note 1]								PUSH POP
!addr16 !!addr24	MOVW	(MOVW)	MOVW						MOVTBLW		
mem [saddrp] [%saddrg]		MOVW									
PSW											PUSH POP
SP	ADDWG SUBWG										
post											PUSH POP PUSHU POPU
[TDE+]		(MOVW)						SACW			
byte											MACW MACSW

Notes 1. SUBW and CMPW are identical to ADDW.

2. There is no second operand, or the second operand is not an operand address.

3. When saddrp is saddrp2 in this combination, this is a short code length instruction.

4. MULUW and DIVUX are identical to MULW.

(3) 24-bit instructions (The values enclosed by parentheses are combined to express WHL description as rg.)

MOVG, ADDG, SUBG, INCG, DECG, PUSH, POP

Table 28-3. 24-Bit Addressing Instructions

First operand \ Second operand	#imm24	WHL	rg rg'	saddrg	!!addr24	mem1	[%saddrg]	SP	None[Note]
WHL	(MOVG) (ADDG) (SUBG)	(MOVG) (ADDG) (SUBG)	(MOVG) (ADDG) (SUBG)	(MOVG) ADDG SUBG	(MOVG)	MOVG	MOVG	MOVG	
rg	MOVG ADDG SUBG	(MOVG) (ADDG) (SUBG)	MOVG ADDG SUBG	MOVG	MOVG				INCG DECG PUSH POP
saddrg		(MOVG)	MOVG						
!!addr24		(MOVG)	MOVG						
mem1		MOVG							
[%saddrg]		MOVG							
SP	MOVG	MOVG							INCG DECG

Note There is no second operand, or the second operand is not an operand address.

(4) Bit manipulation instructions

MOV1, AND1, OR1, XOR1, SET1, CLR1, NOT1, BT, BF, BTCLR, BFSET

Table 28-4. Bit Manipulation Instruction Addressing Instructions

First operand \ Second operand	CY	saddr.bit sfr.bit A.bit X.bit PSWL.bit PSWH.bit mem2.bit !addr16.bit !!addr24.bit	/saddr.bit /sfr.bit /A.bit /X.bit /PSWL.bit /PSWH.bit /mem2.bit /!addr16.bit /!!addr24.bit	None[Note]
CY		MOV1 AND1 OR1 XOR1	AND1 OR1	NOT1 SET1 CLR1
saddr.bit sfr.bit A.bit X.bit PSWL.bit PSWH.bit mem2.bit !addr16.bit !!addr24.bit	MOV1			NOT1 SET1 CLR1 BF BT BTCLR BFSET

Note There is no second operand, or the second operand is not an operand address.

(5) Call return instructions and branch instructions

CALL, CALLF, CALLT, BRK, RET, RETI, RETB, RETCS, RETCSB, BRKCS, BR, BNZ, BNE, BZ, BE, BNC, BNL, BC, BL, BNV, BPO, BV, BPE, BP, BN, BLT, BGE, BLE, BGT, BNH, BH, BF, BT, BTCLR, BFSET, DBNZ

Table 28-5. Call Return Instructions and Branch Instruction Addressing Instructions

Instruction Address Operand	$addr20	$!addr20	!addr16	!!addr20	rp	rg	[rp]	[rg]	!addr11	[addr5]	RBn	None
Basic instructions	BC[Note] BR	CALL BR	CALL BR RETCS RETCSB	CALL BR	CALL BR	CALL BR	CALL BR	CALL BR	CALLF	CALLT	BRKCS	BRK RET RETI RETB
Composite instructions	BF BT BTCLR BFSET DBNZ											

Note BNZ, BNE, BZ, BE, BNC, BNL, BL, BNV, BPO, BV, BPE, BP, BN, BLT, BGE, BLE, BGT, BNH, and BH are identical to BC.

(6) Other instructions

ADJBA, ADJBS, CVTBW, LOCATION, SEL, NOT, EI, DI, SWRS

APPENDIX A MAJOR DIFFERENCES BETWEEN THE μPD784216 SUBSERIES AND THE μPD780058 SUBSERIES

Item \ Series Name		μPD784225 Subseries	μPD784216 Subseries	μPD780058 Subseries
CPU		16-bit CPU		8-bit CPU
Minimum instruction execution time	When the main system clock is selected	160 ns (at 12.5-MHz operation)		400 ns (at 5.0-MHz operation)
	When the subsystem clock is selected	61 μs (at 32.768-kHz operation		122 μs (at 32.768-kHz operation)
Memory space		1 Mbytes		64 Kbytes
I/O port	Total	67	86	68
	CMOS inputs	8	8	2
	CMOS I/O	59	72	62
	N-channel open drain I/O	—	6	4
Pins with added functions[Note]	Pins with pullup resistors	57	70	66 (62 for flash memory versions)
	LED direct drive outputs	16	22	12
	Medium voltage pins	—	6	4
Timer/counters		• 16-bit timer/counter × 1 unit • 8-bit timer/counter × 4 units	• 16-bit timer/counter × 1 unit • 8-bit timer/counter × 6 units	• 16-bit timer/counter × 1 unit • 8-bit timer/counter × 2 units
Serial interface		• UART/IOE (3-wire serial I/O) × 2 channels • CSI (3-wire serial I/O) × 1 channel		• UART (time division transmission function)/ IOE (3-wire serial I/O) × 2 channels • CSI (3-wire serial I/ O, 2-wire serial I/O, SBI) × 1 channel • CSI (3-wire serial I/O with automatic communication function) × 1 channel
Interrupts	NMI pin	Yes		No
	Macro service	Yes		No
	Context switching	Yes		No
	Programmable priority	4 levels		2 levels
Standby function		• HALT/STOP/IDLE mode • In low power consumption mode: HALT or IDLE mode		HALT/STOP mode
ROM correction		Yes	No	Yes
Package		• 80-pin plastic QFP (14 × 14 mm) • 80-pin plastic TQFP (fine pitch) (12 × 12 mm)	• 100-pin plastic QFP (fine pitch) (14 × 14 mm) • 100-pin plastic QFP (14 × 20 mm)	• 80-pin plastic QFP (14 × 14 mm) • 80-pin plastic TQFP (fine pitch) (12 × 12 mm)

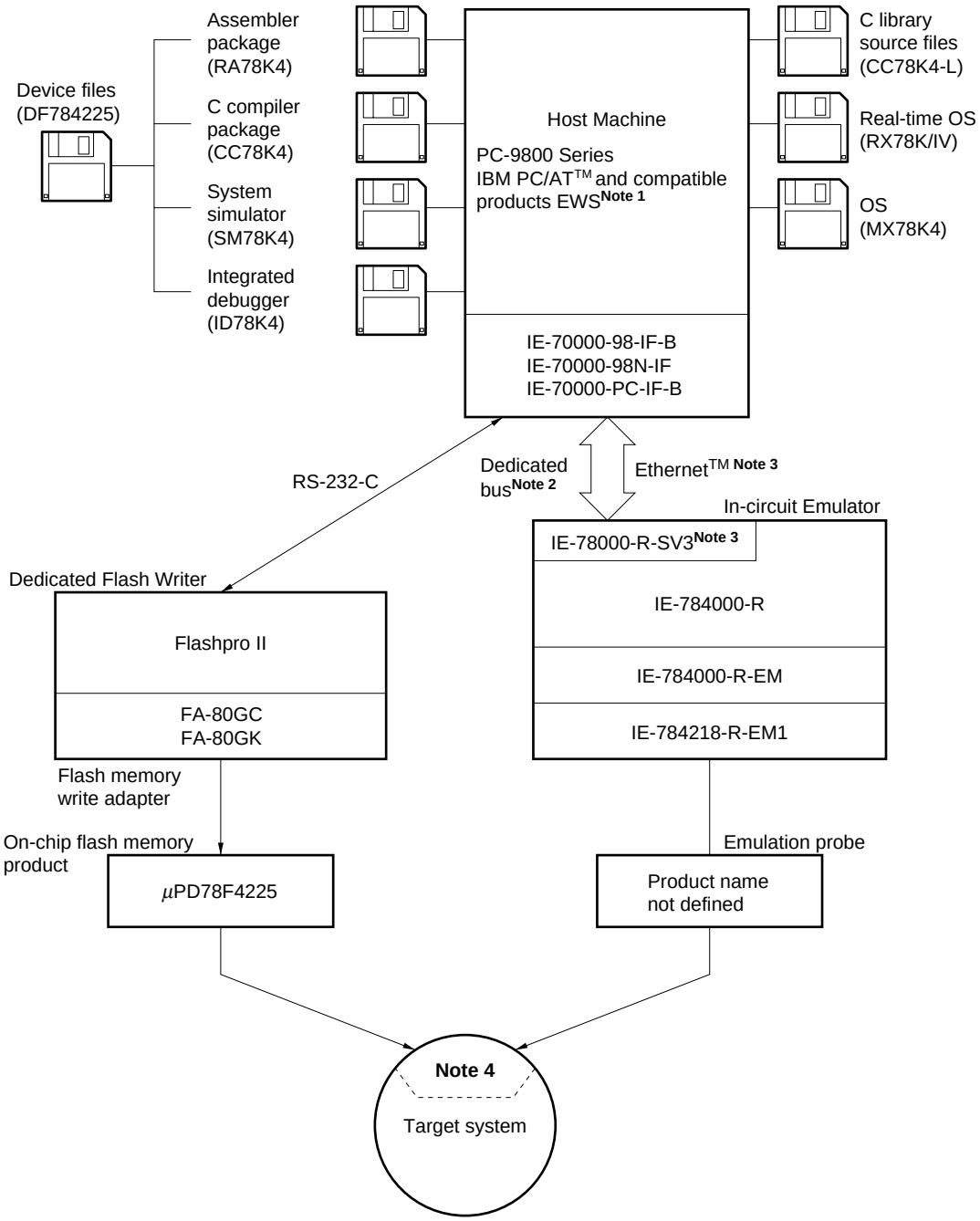
Note The pins with added functions are included in the I/O pins.

[MEMO]

APPENDIX B DEVELOPMENT TOOLS

The tools needed to develop systems that use μPD784225 Subseries products are described.

Figure B-1. Development Tool Structure



Notes 1. SM78K4, ID78K4, and MX78K4 are not supported.

2. A PC-9800 series or IBM PC/AT PC is used as the host machine.

3. An engineering workstation (EWS) is used as the host machine.

4. Socket adapter (EV-9200GC-80) or conversion adapter (TGK-080)

Remark In this figure, the software supply media are 3.5-inch floppy disks.

B.1 Language Processing Software (1/2)

<table>
<tr><td>RA78K4 assembler package</td><td>This relocatable assembler can translate device files and be used with all products commonly in the 78K/IV series. Since this relocatable assembler has a macro function, improved development efficiency is designed. In addition, a structured assembler that can explicitly describe the program control structure is also added, and program productivity and maintainability can be increased. This assembler can be used with the device file (DF784225), that is sold separately.
<table>
<tr><td>Structured assembler preprocessor
(Program name: ST78K4)</td><td>Source program described in assembly language in the structured assembler is translated into a format that can be input to the relocatable assembler.</td></tr>
<tr><td>Relocatable assembler
(Program name RA78K4)</td><td>This program translates a source program written in assembly language into machine language and generates relocatable object module files.</td></tr>
<tr><td>Linker
(Program name: LK78K4)</td><td>The object module files created by the relocatable assembler and the library files are linked. The absolute address of the program is set, and the load module file is created.</td></tr>
<tr><td>Object converter
(Program name: OC78K4)</td><td>The load module file created by the linker is converted into a format that can be downloaded to the in-circuit emulator or dedicated flash writer.</td></tr>
<tr><td>Librarian
(Program name: LB78K4)</td><td>The object module files output by the relocatable assembler are linked to create one library file. In addition, the library file is changed.</td></tr>
<tr><td>List converter
(Program name: LCNV78K4)</td><td>The assembler list files output by the relocatable assembler, the object module files, and load module files are used to create the assemble list embedded with absolute values.</td></tr>
</table>
Part number: μS××××RA78K4</td></tr>
</table>

B.1 Language Processing Software (2/2)

CC78K4 C compiler package	This C compiler can translate device files and be used with all 78K/IV series products. The language specification conforms to ANSI, and the programs can be stored on a ROM. The functions provide special function register operations, bit manipulation, variables that use short direct addressing, and interrupt control functions. By using these functions, the programs can be described more efficiently, and more efficient objects can be implemented. In addition, sample programs of the start-up routine and object libraries of standard functions are provided. It can be used with the assembler package (RA78K4) and device file (DF784225) that are sold separately.
	Part number: μS××××CC78K4
CC78K4-L C library source file	These are the source programs in the library in the C compiler package (CC78K4). They are required when the library is updated (better match to the user specifications).
	Part number: μS××××CC78K4-L
DF784225**Note** Device file	This file contains device-specific information. It can be used with the RA78K4, CC78K4, SM78K4, and ID78K4, that are sold separately.
	Part number: μS××××DF784225

Note DF784225 can be used with all of the products of RA78K4, CC78K4, SM78K4, and ID78K4.

Remark The ×××× part number differs with the host machine and operating system that are used.

μS××××RA78K4
μS××××CC78K4
μS××××CC78K4-L
μS××××DF784225

××××	Host Machine	OS	Delivery Media
5A13	PC-9800 series	MS-DOS	3.5-inch 2HD
5A10		(Ver. 3.30 to Ver. 6.2**Note**)	5-inch 2HD
7B13	IBM PC/AT and compatible machines	See **B.4.**	3.5-inch 2HC
7B10			5-inch 2HC
3P16	HP9000 series 700	HP-UX (rel.9.01)	Digital audio tape (DAT)
3K15	SPARCstation™	SunOS™ (rel.4.1.1)	Cartridge tape (QIC-24)
3R15	NEWS™	NEWS-OS™	Cartridge tape (QIC-24)

Note The task swap function is available in MS-DOS Ver. 5.0 or later versions but cannot be used by the software described above.

B.2 Debugging Tools

B.2.1 Hardware

IE-784000-R	IE-784000-R is an in-circuit emulator that can be used with the 78K/IV series. IE-784000-R can be used with the optional IE-784000-R-EM and IE-784218-R-EM1 emulation boards, that are sold separately. The host machine is connected in order to debug. The integrated debugger and the device files that are sold separately are required. By using this in-circuit emulator in combination with them, debugging is possible at the source program levels of the C language and the structured assembly language. Debugging and program inspection have better efficiency than the C0 coverage function. IE-784000-R can be connected to the host machine by Ethernet or a dedicated bus. A separately sold interface adapter is required.
IE-784000-R-EM	This emulation board is used with the 78K/IV Series.
IE-784218-R-EM1**Note**	This board emulates the peripheral functions of the μPD784225 Subseries. It can be used with the IE-784000-R.
IE-70000-98-IF-B	This is the interface adapter when a PC-9800 series PC is used as the host machine.
IE-70000-98N-IF	This is the interface adapter and cable when a PC-9800 series notebook PC is used as the host machine.
IE-70000-PC-IF-B	This is the interface adapter when an IBM PC/AT PC is the host machine.
IE-78000-R-SV3	This is the interface board when an EWS is used as the host machine and to connect to a board in the IE-784000-R. 10BASE-5 is supported for Ethernet. If another method is used, a commercial adapter is required.
IE-784000-R-BK	This board upgrades the system of the in-circuit emulator used in the 75X/XL Series and 78K Series to the IE-784000-R.
Product name not defined	This is the emulation probe for the μPD784225. It is used for 80-pin plastic TQFP (fine pitch) (GK-BE9 type). One 80-pin conversion adapter TKG-080SDW that facilitates target system development is attached.
TGK-080SDW	This conversion adapter connects the target system board manufactured to mount the 80-pin plastic TQFP (fine pitch) (GK-BE9 type) to the EP-78054GK-R. TGK-080SDW is a product of TOKYO ELETECH CORPORATION. Customers should consult an NEC representative.
Product name not fixed	This is the emulation probe for the μPD784225 Subseries. It is used for 80-pin plastic QFP (GC-8BT type). One 80-pin conversion socket EV-9200GC-80 that facilitates target system development is attached.
EV-9200GC-80	The target system substrate created to enable the 80-pin plastic QFP (GC-8BT type) to be mounted, and the conversion adapter for connecting the μPD784225 Subseries emulation probe.

Note Under development

Remark TGK-080SDW is sold individually.
The EV-9200GC-80 is sold in groups of five.

B.2.2 Software (1/2)

SM78K4 system simulator	When the target system operation is being simulated on the host machine, debugging can be performed on the C source level or assembler level. SM78K4 runs on Windows. Even if the IE-784000-R is not used, by using the SM78K4, verification of the logic and performance of the application can be performed independent of the hardware. Development efficiency and software quality are improved. This can be used with the device file (DF784225), that is sold separately.
	Part number: μS××××SM78K4
ID78K4 Integrated debugger	This is the control program for debugging the 78K/IV Series. The graphical user interface (GUI) is Windows on a personal computer and OSF/Motif™ on an EWS. The appearances and operations that conform to these GUIs are provided. In addition, the debugging functions for the C language are improved. By using the window integrated functions that link the source program display, disassemble display, and memory display to the trace result, the trace result can be displayed at the C language level. By incorporating function expansion modules, such as the task debugger or the system performance analyzer, the debugging efficiency of the program run on a real-time operating system can be improved. This can be used with the device file (DF784225), that is sold separately.
	Part number: μS××××ID78K4
DF784225 **(Note)** Device file	This file contains device-specific information. It can be used with the SM78K4, ID78K4, RA78K4, and CC78K4, that are sold separately.
	Part number: μS××××DF784225

Note DF784225 can be used with all of the products of RA78K4, CC78K4, SM78K4, and ID78K4. This is being developed.

Remark The part number ×××× differs with the host machine and operating system used.

μS××××SM78K4
μS××××ID78K4

××××	Host Machine	OS	Delivery Media
AA13	PC-9800 series	MS-DOS (Ver. 3.30 to Ver. 6.2**Note**) + Windows (Ver. 3.1)	3.5-inch 2HD
AB13	IBM PC/AT and compatible machines (Windows Japanese version)	See **B.4.**	3.5-inch 2HC
BB13	IBM PC/AT and compatible machines (Windows English version)		3.5-inch 2HC

Note The task swap function is available in MS-DOS Ver. 5.0 or later versions but cannot be used by the software described above.

B.2.2 Software (2/2)

μSxxxxDF784225

xxxx	Host Machine	OS	Delivery Media
5A13	PC-9800 series	MS-DOS (Ver. 3.30 to Ver. 6.2[Note])	3.5-inch 2HD
5A10			5-inch 2HD
7B13	IBM PC/AT and compatible machines	See **B.4**.	3.5-inch 2HC
7B10			5-inch 2HC
3P16	HP9000 series 700	HP-UX (rel.9.01)	Digital audio tape (DAT)
3K15	SPARCstation	SunOS (rel.4.1.1)	Cartridge tape (QIC-24)

Note The task swap function is available in MS-DOS Ver. 5.0 or later versions but cannot be used by the software described above.

B.3 Flash Memory Write Tools

Flashpro II	This is a flash writer dedicated for the on-chip flash memory microcontrollers. This is a product of Naitou Densei Machidaseisakusho Co., Ltd.
FA-80GC[Note]	This is a flash memory write adapter for the (μPD784225 and 784225Y Subseries to connect to Flashpro II. This has a 80-pin plastic QFP (GC-8BT type). This is a product of Naitou Densei Machidaseisakusho Co., Ltd.
FA-80GK[Note]	This is the flash memory write adapter for the (μPD784225 and 784225Y Subseries to connect to the Flashpro II. This has a 80-pin plastic TQFP (fine pitch) (GK-BE9 type). This is a product of Naitou Densei Machidaseisakusho Co., Ltd.

Note Under development

B.4 IBM PC Operating Systems

The following IBM PC operating systems are supported.

OS	Version
PC DOS	Ver. 5.02 to Ver. 6.3 J6.1/V[Note 2] to J6.3/V[Note 2]
Windows[Note 1]	Ver.3.1 to Ver.3.11
MS-DOS	Ver. 5.0 to Ver 6.22 5.0/V[Note 2] to 6.2/V[Note 2]
IBM DOS™	J5.02/V[Note 2]

Notes 1. For an integrated debugger, this is used with PC DOS and Windows.

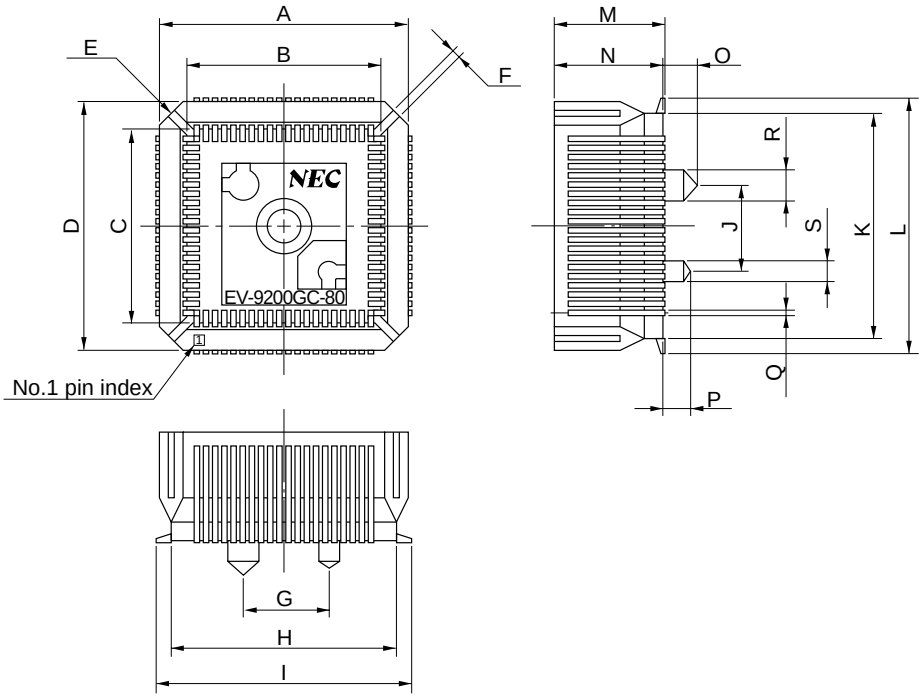
2. This is supported only in the English mode.

Caution The task swap function is available in MS-DOS Ver. 5.0 or later versions but cannot be used by the software described above.

B.5 Socket Adapter (EV-9200GC-80) and Conversion Adapter (TGK-080SDW)

(1) The package drawing of the socket adapter (EV-9200GC-80) and recommended board installation pattern.

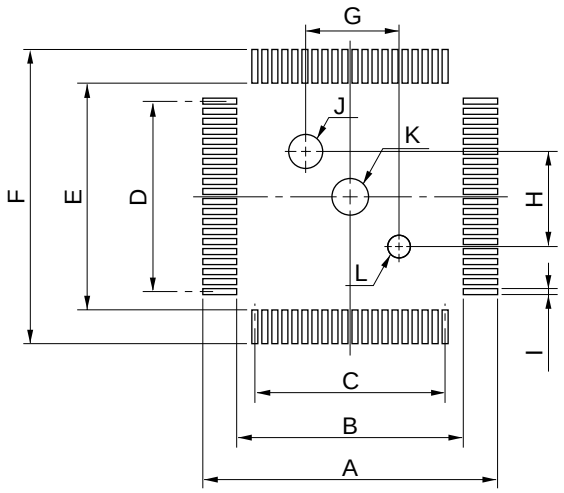
Figure B-2. Package Drawing of EV-9200GC-80 (Reference) (Units: mm)



EV-9200GC-80-G1E

ITEM	MILLIMETERS	INCHES
A	18.0	0.709
B	14.4	0.567
C	14.4	0.567
D	18.0	0.709
E	4-C 2.0	4-C 0.079
F	0.8	0.031
G	6.0	0.236
H	16.0	0.63
I	18.7	0.736
J	6.0	0.236
K	16.0	0.63
L	18.7	0.736
M	8.2	0.323
N	8.0	0.315
O	2.5	0.098
P	2.0	0.079
Q	0.35	0.014
R	ϕ2.3	ϕ0.091
S	ϕ1.5	ϕ0.059

Figure B-3. Recommended Board Installation Pattern of EV-9200GC-80 (Reference) (Units: mm)



EV-9200GC-80-P1E

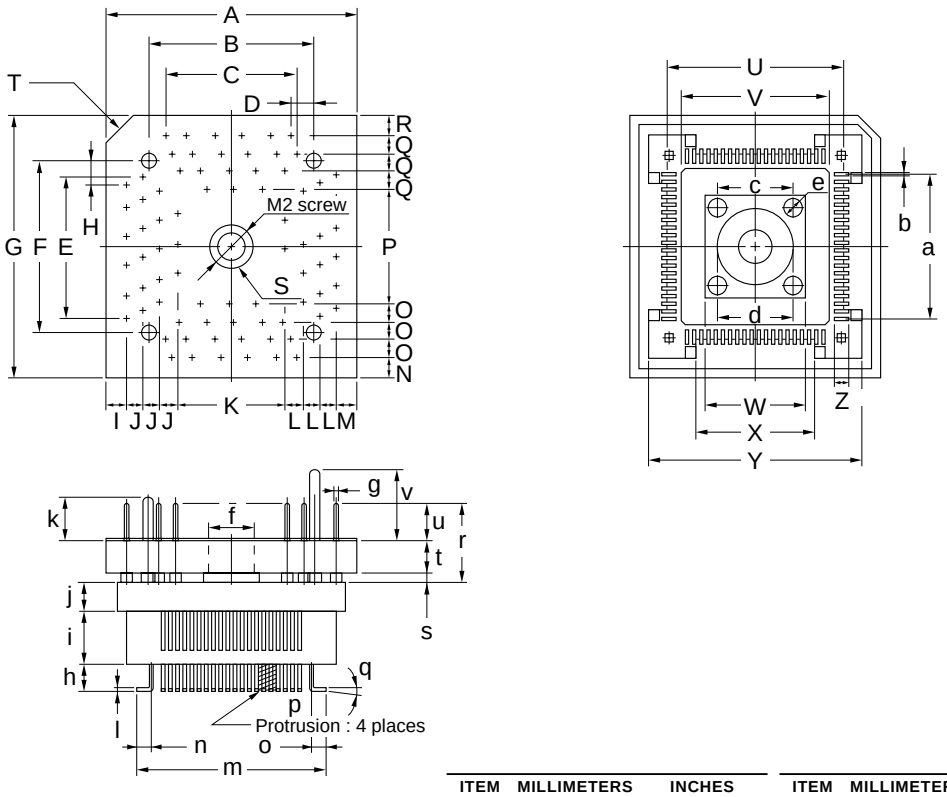
ITEM	MILLIMETERS	INCHES
A	19.7	0.776
B	15.0	0.591
C	$0.65\pm0.02\times19=12.35\pm0.05$	$0.026^{+0.001}_{-0.002}\times0.748=0.486^{+0.003}_{-0.002}$
D	$0.65\pm0.02\times19=12.35\pm0.05$	$0.026^{+0.001}_{-0.002}\times0.748=0.486^{+0.003}_{-0.002}$
E	15.0	0.591
F	19.7	0.776
G	6.0 ± 0.05	$0.236^{+0.003}_{-0.002}$
H	6.0 ± 0.05	$0.236^{+0.003}_{-0.002}$
I	0.35 ± 0.02	$0.014^{+0.001}_{-0.001}$
J	$\phi2.36\pm0.03$	$\phi0.093^{+0.001}_{-0.002}$
K	$\phi2.3$	$\phi0.091$
L	$\phi1.57\pm0.03$	$\phi0.062^{+0.001}_{-0.002}$

Caution Dimensions of mount pad for EV-9200 and that for target device (QFP) may be different in some parts. For the recommended mount pad dimensions for QFP, refer to "SEMICONDUCTOR DEVICE MOUNTING TECHNOLOGY MANUAL" (C10535E).

(2) Package drawing of the adapter (TGK-080SDW)

Combined with the emulation probe and mounted on the substrate.

Figure B-4. TGK-080SDW Package Drawing (Reference) (Units: mm)



ITEM	MILLIMETERS	INCHES
A	18.0	0.709
B	11.77	0.463
C	0.5x19=9.5	0.020x0.748=0.374
D	0.5	0.020
E	0.5x19=9.5	0.020x0.748=0.374
F	11.77	0.463
G	18.0	0.709
H	0.5	0.020
I	1.58	0.062
J	1.2	0.047
K	7.64	0.301
L	1.2	0.047
M	1.58	0.062
N	1.58	0.062
O	1.2	0.047
P	7.64	0.301
Q	1.2	0.047
R	1.58	0.062
S	ϕ3.55	ϕ0.140
T	C 2.0	C 0.079
U	12.31	0.485
V	10.17	0.400
W	6.8	0.268
X	8.24	0.324
Y	14.8	0.583
Z	1.4±0.2	0.055±0.008

ITEM	MILLIMETERS	INCHES
a	0.5x19=9.5±0.10	0.020x0.748=0.374±0.004
b	0.25	0.010
c	ϕ5.3	ϕ0.209
d	ϕ5.3	ϕ0.209
e	ϕ1.3	ϕ0.051
f	ϕ3.55	ϕ0.140
g	ϕ0.3	ϕ0.012
h	1.85±0.2	0.073±0.008
i	3.5	0.138
j	2.0	0.079
k	3.0	0.118
l	0.25	0.010
m	14.0	0.551
n	1.4±0.2	0.055±0.008
o	1.4±0.2	0.055±0.008
p	h=1.8 ϕ1.3	h=0.071 ϕ0.051
q	0~5°	0.000~0.197°
r	5.9	0.232
s	0.8	0.031
t	2.4	0.094
u	2.7	0.106
v	3.9	0.154

TGK-080SDW-G1E

Note Made by TOKYO ELETECH Corp.

[MEMO]

APPENDIX C BUILTIN SOFTWARE

C.1 Real-time Operating System (1/2)

RX78K/IV real-time OS	RX78K/IV targets the control field that demands real-time performance and has the objective of implementing a multitasking environment. The CPU idle time is allocated to other processing and can improve the overall system performance. In RX78K/IV, system calls (31) that conform to μITRON specifications are provided. In addition, it provides a tool (configurator) that creates the RX78K/IV nucleus and multiple information tables. This is used with the optional assembler package (RA78K4) and device file (DF784225).
	Part number: μS××××RX78K4-ΔΔΔΔ

Caution When purchasing the RX78K/IV, fill out the purchase application and sign a license.

Remark The ×××× and ΔΔΔΔ part numbers differ with the host machine and operating system used.

μS××××RX78K4-ΔΔΔΔ

ΔΔΔΔ	Product Overview	Maximum Number Used During Production
001	Evaluation object	Do not use in manufactured products.
100K	Production object	100,000
001M		1,000,000
010M		10,000,000
S01	Source program	Source program for the production object

××××	Host Machine	OS	Delivery Media
5A13	PC-9800 series	MS-DOS (Ver. 3.30 - Ver. 6.2[Note])	3.5-inch 2HD
5A10			5-inch 2HD
7B13	IBM PC/AT and compatible	See **B.4.**	3.5-inch 2HC
7B10			5-inch 2HC
3P16	HP9000 series 700	HP-UX (rel.9.01)	Digital audio tape (DAT)
3K15	SPARCstation	SunOS (rel.4.1.1)	Cartridge tape (QIC-24)

Note The task swap function is available in MS-DOS Ver. 5.0 or later versions but cannot be used by the software described above.

C.1 Real-time Operating System (2/2)

MX78K/4 OS	This is the operating system of μITRON specification subset. The MX78K4 nucleus is added. Task management, event management, and time management are performed. Task management controls the task execution order and switches to the task executed next.
	Part number: μS××××MX78K4-ΔΔΔ

Remark The ×××× and ΔΔΔ part numbers differ with the host machine and operating system used.

μS××××MX78K4-ΔΔΔ

ΔΔΔ	Product Overview	Cautions
001	Evaluation object	Use when prototyping.
XX	Manufactured object	Use during production
S01	Source program	Can purchase only when purchasing a manufactured object.

××××	Host Machine	OS	Delivery Media
5A13	PC-9800 series	MS-DOS (Ver. 3.30 to Ver. 6.2[Note])	3.5-inch 2HD
5A10			5-inch 2HD
7B13	IBM PC/AT and compatible	See **B.4**.	3.5-inch 2HC
7B10			5-inch 2HC

Note The task swap function is available in MS-DOS Ver. 5.0 or later versions but cannot be used by the software described above.

APPENDIX D REGISTER INDEX

D.1 Register Index

D.2 Register Index (alphabetical order)

[W]

[MEMO]

NEC

Facsimile Message

From:

Name

Company

Tel. FAX

Address

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

Thank you for your kind support.

North America
NEC Electronics Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288
1-408-588-6130

Europe
NEC Electronics (Europe) GmbH
Technical Documentation Dept.
Fax: +49-211-6503-274

South America
NEC do Brasil S.A.
Fax: +55-11-889-1689

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Korea
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: 02-528-4411

Taiwan
NEC Electronics Taiwan Ltd.
Fax: 02-719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Japan
NEC Corporation
Semiconductor Solution Engineering Division
Technical Information Support Dept.
Fax: 044-548-7900

I would like to report the following error/make the following suggestion:

Document title: _____

Document number: _____ Page number: _____

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	❑	❑	❑	❑
Technical Accuracy	❑	❑	❑	❑
Organization	❑	❑	❑	❑