FUJITSU SEMICONDUCTOR
CONTROLLER MANUAL

CM44-10107-4E

F^2MC-16LX

16-BIT MICROCONTROLLER

MB90560/565 Series

HARDWARE MANUAL

FUJITSU

F^2MC-16LX

16-BIT MICROCONTROLLER

MB90560/565 Series

HARDWARE MANUAL

FUJITSU LIMITED

PREFACE

■ Objectives and Intended Reader

Thank you for purchasing Fujitsu semiconductor products.

The MB90560/565 series was developed as a group of general-purpose models in the F²MC-16 LX series, which is a family of original 16-bit single-chip microcontrollers that can be used for application specific IC (ASIC).

This manual is intended for engineers who design products using the MB90560/565 series of microcontrollers. The manual describes the functions and operation of the MB90560/565 series.

■ Trademarks

F^2MC is the abbreviation of FUJITSU Flexible Microcontroller.

Other system and product names mentioned herein are trademarks of their respective owners.

The symbols ™ and ® are sometimes omitted in the text.

■ Organization of This Manual

This manual consists of the following 20 chapters:

CHAPTER 1 "OVERVIEW"

This chapter describes the features of the MB90560/565 series.

CHAPTER 2 "CPU"

This chapter describes the functions and operations of the CPU of the MB90560/565 series.

CHAPTER 3 "RESETS"

This chapter describes resets for the MB90560/565 series.

CHAPTER 4 "CLOCKS"

This chapter describes the clocks used by MB90560/565-series.

CHAPTER 5 "LOW POWER CONSUMPTION MODE"

This chapter describes the low power consumption mode of MB90560/565 series.

CHAPTER 6 "INTERRUPTS"

This chapter explains the interrupts and extended intelligent I/O service (EI^2OS) in the MB905605/565 series.

CHAPTER 7 "SETTING A MODE"

This chapter describes the operating modes of the MB90560/565 series.

CHAPTER 8 "I/O PORTS"

This chapter describes the functions and operations of the I/O ports.

CHAPTER 9 "TIMEBASE TIMER"

This chapter describes the functions and operation of the timebase timer.

CHAPTER 10 "WATCHDOG TIMER"

This chapter describes the functions and operation of the watchdog timer.

CHAPTER 11 "16-BIT RELOAD TIMER"

The chapter describes the functions and operation of the 16-bit reload timer.

CHAPTER 12 "MULTIFUNCTIONAL TIMERS"

This chapter describes the operations of the multifunctional timers of MB90560/565 series.

CHAPTER 13 "UART"

This chapter explains the functions and operation of UART.

CHAPTER 14 "DTP/EXTERNAL INTERRUPT CIRCUIT"

This chapter describes the functions and operation of the DTP/external interrupt circuit.

CHAPTER 15 "DELAYED INTERRUPT GENERATOR MODULE"

This chapter describes the functions and operation of the MB90560/565 series delayed interrupt generator module.

CHAPTER 16 "8/10-BIT A/D CONVERTER"

This chapter describes the functions and operation of the 8/10-bit A/D converter.

CHAPTER 17 "ADDRESS MATCH DETECTION FUNCTION"

This chapter describes the address match detection function and operation of the MB90560/565 series.

CHAPTER 18 "ROM MIRRORING FUNCTION SELECTION MODULE"

This chapter describes the function and operation of the MB90560/565 series ROM mirroring function selection module.

CHAPTER 19 "512K BIT FLASH MEMORY"

This chapter describes the functions and operations of the 512K-bit (64KB) flash memory of the MB90560/565 series.

CHAPTER 20 "1M-BIT (128KB) FLASH MEMORY"

This chapter describes the functions and operations of the 1M-bit (128KB) flash memory of the MB90F568.

CHAPTER 21 "EXAMPLE OF MB90F562/F562B/F568 SERIAL PROGRAMMING CONNECTION"

This chapter provides examples of serial programming connection with the AF220/AF210/AF120/AF110 flash microcomputer programmer manufactured by YDC Corporation.

APPENDIX

The appendix provides I/O maps and outlines instructions.

- The contents of this document are subject to change without notice. Customers are advised to consult with FUJITSU sales representatives before ordering.
- The information and circuit diagrams in this document are presented as examples of semiconductor device applications, and are not intended to be incorporated in devices for actual use. Also, FUJITSU is unable to assume responsibility for infringement of any patent rights or other rights of third parties arising from the use of this information or circuit diagrams.
- The products described in this document are designed, and manufactured as contemplated for general use, including without limitation, ordinary industrial use, general office use, personal use, and household use, but are not designed, developed and manufactured as contemplated (1) for use accompanying fatal risks or dangers that, unless extremely high safety is secured, could have a serious effect to the public, and could lead directly to death, personal injury, severe physical damage or other loss (i.e., nuclear reaction control in nuclear facility, aircraft flight control, air traffic control, mass transport control, medical life support system, missile launch control in weapon system), or (2) for use requiring extremely high reliability (i.e., submersible repeater and artificial satellite).
  Please note that Fujitsu will not be liable against you and/or any third party for any claims or damages arising in connection with above-mentioned uses of the products.
- Any semiconductor devices have an inherent chance of failure. You must protect against injury, damage or loss from such failures by incorporating safety design measures into your facility and equipment such as redundancy, fire protection, and prevention of over-current levels and other abnormal operating conditions.
- If any products described in this document represent goods or technologies subject to certain restrictions on export under the Foreign Exchange and Foreign Trade Law of Japan, the prior authorization by Japanese government will be required for export of those products from Japan.

CONTENTS

CHAPTER 1 OVERVIEW

This chapter describes the features of the MB90560/565 series.

1.1 Features

The MB90560/565 series of general-purpose 16-bit microcontrollers was developed for applications that require high-speed real-time processing in a variety of industrial systems, OA equipment, and process control systems. A specific feature of the series is a built-in multifunctional timer that can easily output desired waveforms.

The instruction set inherits the AT architecture of the original Fujitsu F^2MC-8L and F^2MC-16L, and has additional instructions supporting high-level languages. In addition, it has an extended addressing mode, enhanced multiply/divide instructions (signed), and reinforced bit manipulation instructions. The chip also has a 32-bit accumulator that enables long-word data processing.

■ Features of the MB90560/565 Series

❍ Clock

- Built-in oscillator circuit and PLL clock multiplier circuit
- Source oscillation; Main clock that divides the source oscillation by two (0.5 to 8 MHz when the source oscillation is 1 to 16 MHz); and PLL clock that multiplies the source oscillation by 1 to 4 (4 to 16 MHz when the source oscillation is 4 MHz), which can be configured from the machine clock
- Minimum instruction execution time: 62.5 ns (when the source oscillation is 4 MHz, PLL clock is multiplied by 4, and Vcc is 5 V)

❍ Maximum memory address space: 16M bytes

- Internal 24-bit addressing
- Bank addressing

❍ Optimum instruction set for controller applications

- Many data types (bit, byte, word, and long word)
- As many as 23 addressing modes
- Enhanced high-precision arithmetic operation by a 32-bit accumulator
- Enhanced signed multiply/divide instructions and RETI instruction function

❍ Instruction set supporting high-level language (C) and multi-tasking

- System stack pointer
- Instruction set symmetry and barrel shift instructions.

❍ Program patch function (Two-address pointer)

- **4-byte instruction queue**

- **Interrupt function**
  - Programmable priority level
  - Interrupt function based on 32 interrupt causes
- **Data transfer function**
  - Extended intelligent I/O service function using up to 16 channels
- **Low-power consumption mode (standby mode)**
  - Sleep mode (in which the CPU operating clock stops)
  - Time-base timer mode (in which only the source oscillation and time-base timer are active)
  - Stop mode (in which the source oscillation stops)
  - CPU intermittent operation mode (in which the CPU operates at every specified cycle)
- **Packages**
  - QFP-64 (FPT-64P-M09: 0.65 mm pin pitch, FPT-64P-M06: 1.00 mm pin pitch)
  - SH-DIP (DIP-64-M01: 1.778 mm pin pitch)
- **Process : CMOS technology**

■ Internal Peripheral Functions (resources)

- **I/O ports: Up to 51 posts**

- **Time-base timer: 1 channel**

- **Watchdog timer: 1 channel**

- **16-bit reload timer: 2 channels**

❍ **Advanced timer: 1 channel**

- 16-bit free running timer: 1 channel
- Output compare: 6 channels
  - When the counter value of the 16-bit free-running timer matches the value set in the compare register, an interrupt request can be output.
- Input capture: 4 channels
  - When the effective edge of a signal that is output from an external input terminal is detected, the counter value of the 16-bit free-running timer can be read into the input capture data register. Furthermore, an interrupt request can be output.
- 8/16-bit PPG timer: 8 bits x 6 channels or 16 bits x 3 channels
  - Capable of changing the duty or period of the output pulses as desired.
- Waveform generation circuit (8-bit timer: 3 channels)

❍ **UART: 2 channels**

- With full duplex double buffer (8-bit length)
- Capable of asynchronous or clock synchronous serial transfer (I/O extended serial)

❍ **DTP/external interrupt (8 channels)**

- Activating the extended intelligent I/O service when an external interrupt is input
- Outputting an interrupt when an external interrupt is input

❍ **Delayed interrupt generator module**

Generates an interrupt request for task switching.

❍ **8/10-bit A/D converter (8 channels)**

- Selectable resolution of 8 or 10 bits

1.2 Product Lineup

Table 1.2-1 "Product Lineup of the MB90560/565 Series" shows the product lineup of the MB90560 series. Table 1.2-2 "Product Lineup of the MB90565 Series" shows the product lineup of the MB90565 series.

■ **Product Lineup**

Table 1.2-1 Product Lineup of the MB90560/565 Series

Model	MB90V560	MB90F562/B	MB90562/A	MB90561/A
Type	Evaluation device	Flash Type ROM	Mask ROM Type	
ROM size	Not installed	64K bytes		32K bytes
Power supply for emulators only (*1)	-	-		-
RAM size	4K bytes	2K bytes		1K bytes
CPU function	Number of basic instructions: 351 Minimum instruction execution time: 62.5 ns/4 MHz (when PLL clock is multiplied by 4) Number of addressing modes: 23 Program patch function: 2-address pointer Maximum memory address space: 16M bytes			
Port	I/O ports (CMOS): 51			
UART	With a full duplex double buffer. Capable of synchronous or asynchronous clock transfer. Also can be used for serial I/O. Dedicated built-in baud rate generator. Two built-in channels.			
16-bit reload timer	16-bit reload timer operation. Two built-in channels.			
Advanced timer	16-bit free running timer x 1 channel Output compare x 6 channels Input capture x 4 channels 8/16-bit PPG timer (8-bit mode x 6 channels, 16-bit mode x 3 channels) Waveform generation circuit: 8-bit timer x 3 channels Three-phase waveform output with dead time period provided.			
8/10-bit A/D converter	8 channels (input multiplex) Capable of 8-bit or 10-bit resolution Conversion time:6.13µs or less (operation at internal 16 MHz clock)			
DTP/External interrupt	8 channels (8 channels can be used for this and A/D input.) Interrupt cause: L-to-H edge, H-to-L edge, L-level, or H-level selectable			
Low-power consumption mode	Sleep mode, time-base timer mode, stop mode, and CPU intermittent mode			

Table 1.2-1 Product Lineup of the MB90560/565 Series (Continued)

Model	MB90V560	MB90F562/B	MB90562/A	MB90561/A
Process	CMOS			
Package	PGA256	QFP-64 (FPT-64P-M09: 0.65 mm pin pitch) QFP-64 (FPT-64P-M06: 1.00 mm pin pitch) SH-DIP (DIP-64P-M01: 1.778 mm pin pitch)		
Operating voltage	5V+10% to 5V-10% (When the maximum machine clock is 16 MHz)			

*1 This is the setting of the DIP switch S2 when the emulation pod MB2145-507 is used. For details, see Section 2.7 "Dedicated Registers", of the "MB2145-507 Hardware Manual."

Table 1.2-2 Product Lineup of the MB90565 Series

Model	MB90F568	MB90568	MB90567
Type	Built-in flash memory type	Built-in mask ROM type	
ROM size	128K bytes		96K bytes
RAM size	4K bytes		4K bytes
Power supply for emulators only (*1)	-		-
CPU function	Number of basic instructions: 351 Minimum instruction execution time: 62.5 ns/4 MHz (when the PLL clock multiplies source oscillation by 4) Number of addressing modes: 23 Program patch function: 2-address pointer Maximum memory address space: 16M bytes		
Port	I/O ports (CMOS): 51		
UART	With a full duplex double buffer. Capable of synchronous or asynchronous clock transfer. Can also be used for serial I/O. Dedicated built-in baud rate generator. Two built-in channels.		
16-bit reload timer	16-bit reload timer operation. Two built-in channels.		
Advanced timer	16-bit free-running timer x 1 channel Output compare x 6 channels Input capture x 4 channels 8/16-bit PPG timer (8-bit mode x 6 channels, 16-bit mode x 3 channels) Output of three-phase waveform and dead time period for waveform generation circuit (8-bit timer x 3 channels)		
8/10-bit A/D converter	8 channels (input multiplex) Capable of 8-bit or 10-bit resolution Conversion time: 6.13 µs (on maximum machine clock of 16 MHz)		
DTP/External interrupt	8 channels (8 channels can be used for this and A/D input.) Interrupt cause: L-to-H edge, H-to-L edge, L-level, or H-level selectable		

Table 1.2-2 Product Lineup of the MB90565 Series (Continued)

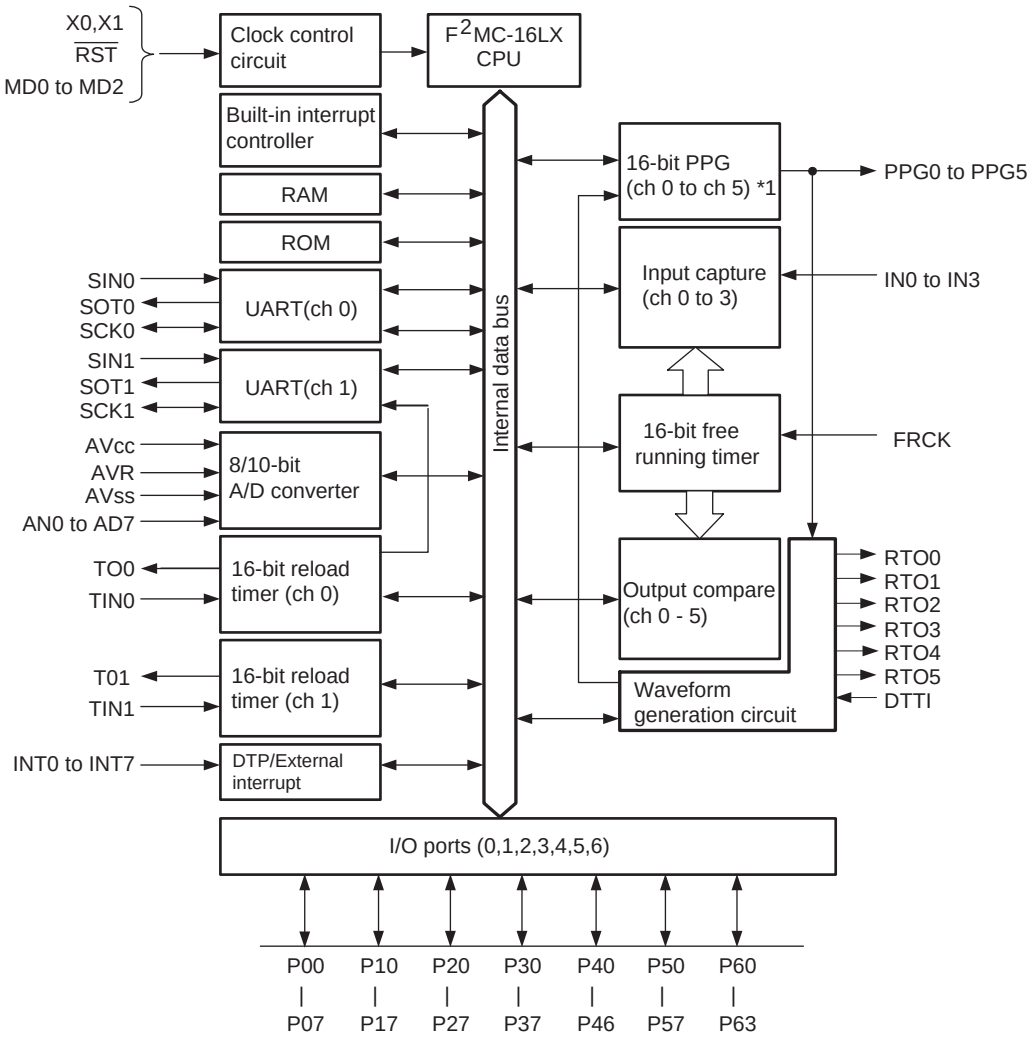
Model	MB90F568	MB90568	MB90567
Low-power consumption mode	Sleep mode, time-base timer mode, stop mode, and CPU intermittent mode		
Process	CMOS		
Package	QFP-64 (FPT-64P-M09: 0.65 mm pin pitch) QFP-64 (FPT-64P-M06: 1.00 mm pin pitch)		
Operating voltage	3V+10% to 3V-10% (on maximum machine clock of 8 MHz)		

1.3 Block Diagram

Figure 1.3-1 "Block Diagram" shows the block diagram of the MB90560/565 series.

■ Block Diagram

Figure 1.3-1 Block Diagram



* : These channels are used for 8-bit PPG. Three output channels (ch 1/3/5) are used for 16-bit PPG.

Note:
The I/O ports and the peripheral functions (resources) share the pins.
For more information, see Sections 1.4, "Pin Assignments" and 1.6, "Pin Functions."
The pins cannot be used as I/O ports when used as those of the peripheral functions (resources).

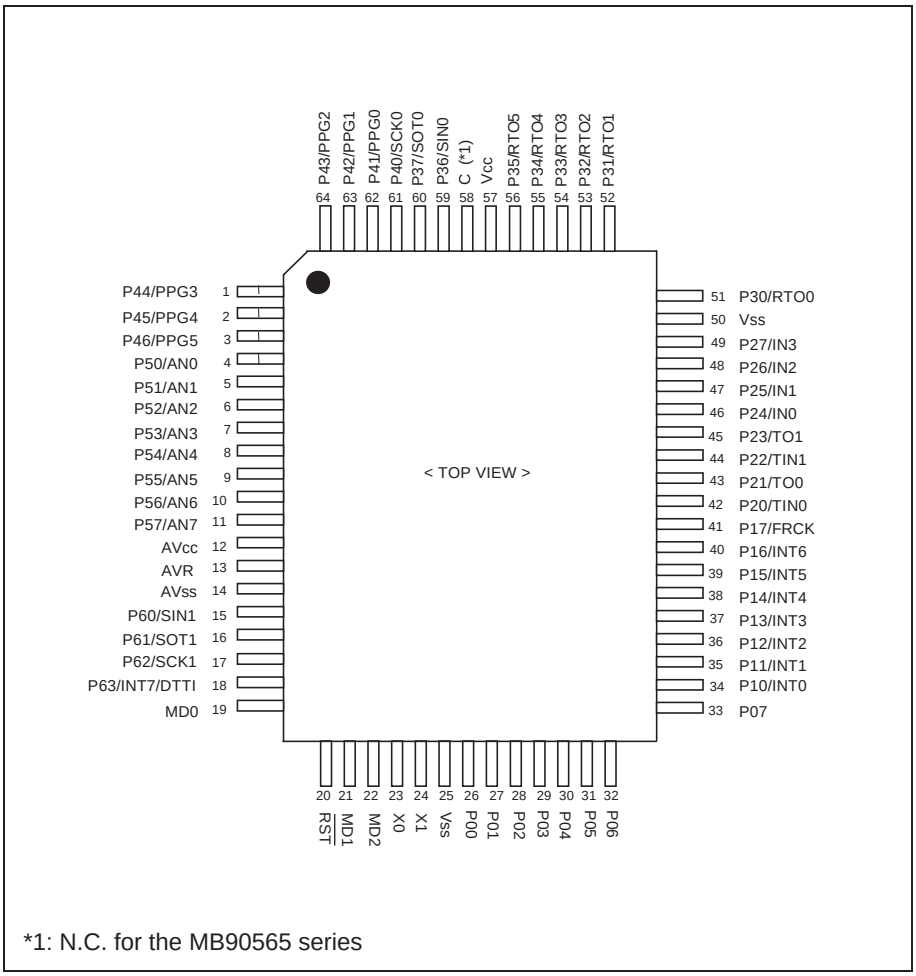
1.4 Pin Assignments

This section provides the pin assignments for the following three packages:

- **FPT-64P-M06**
- **FPT-64P-M09**
- **DIP-64P-M01 (Neither MB90568, MB90567 nor MB90F568 are supported.)**

■ Pin Assignment of FPT-64P-M06

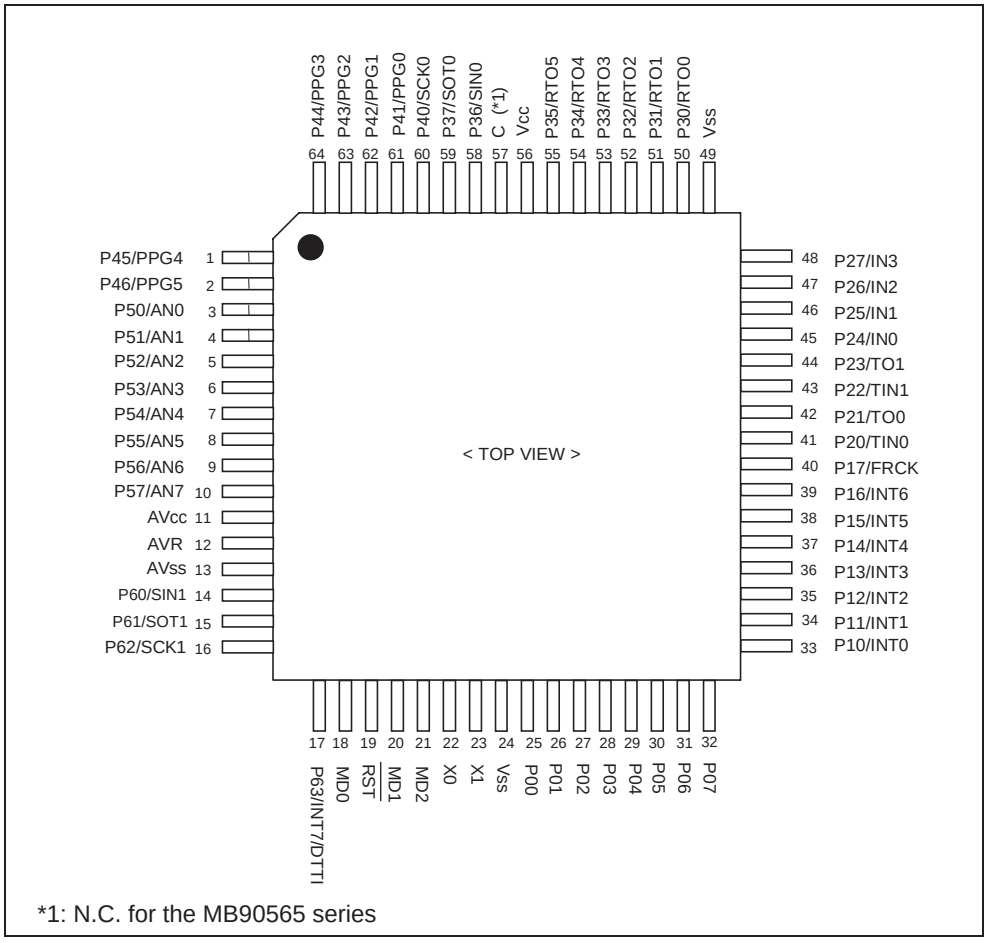
Figure 1.4-1 Pin Assignment of FPT-64P-M06



*1: N.C. for the MB90565 series

■ Pin Assignment of FPT-64P-M09

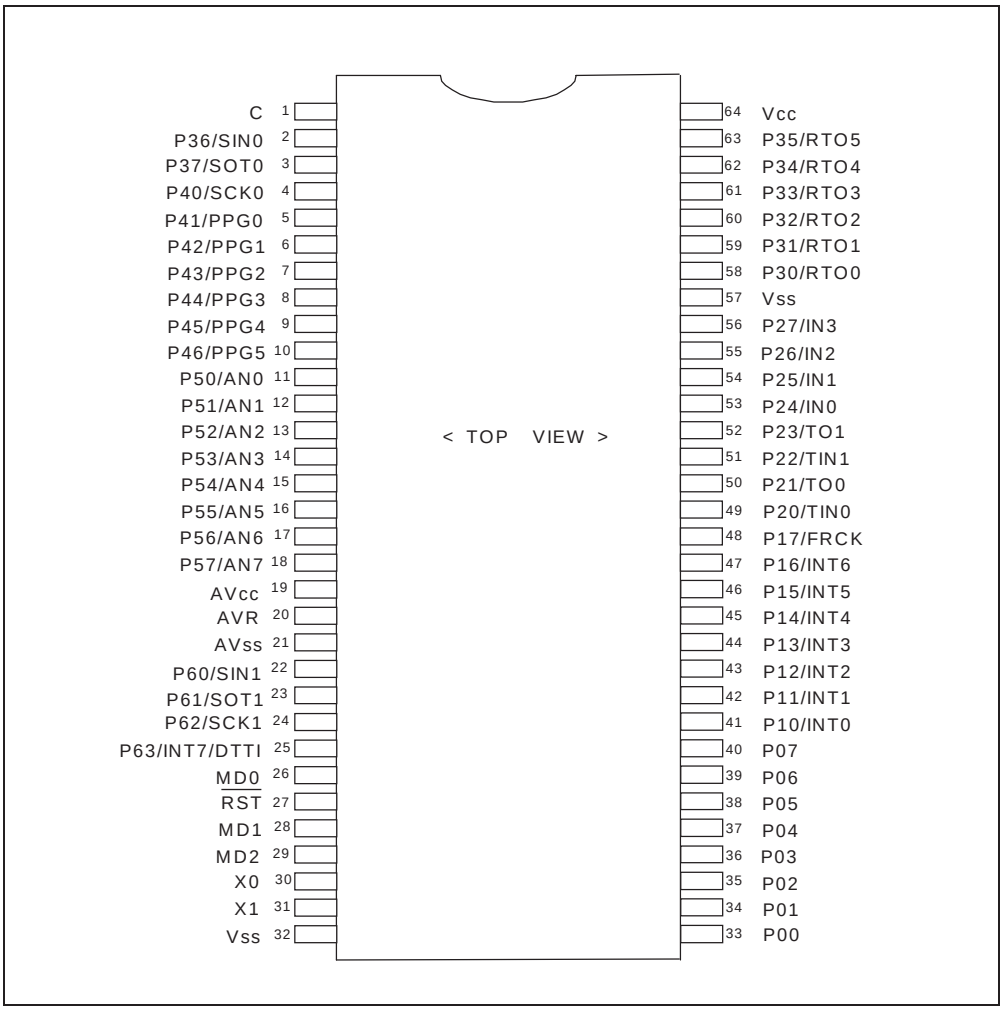
Figure 1.4-2 Pin Assignment of FPT-64P-M09



*1: N.C. for the MB90565 series

■ Pin Assignment of DIP-64P-M01

Figure 1.4-3 Pin Assignment of DIP-64P-M01



1.5 Package Dimensions

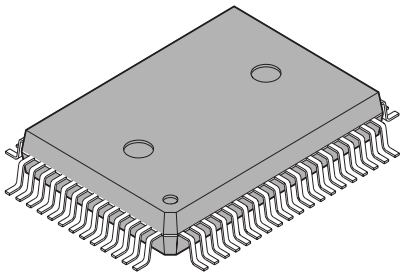
This section provides the dimensions of the following three packages:

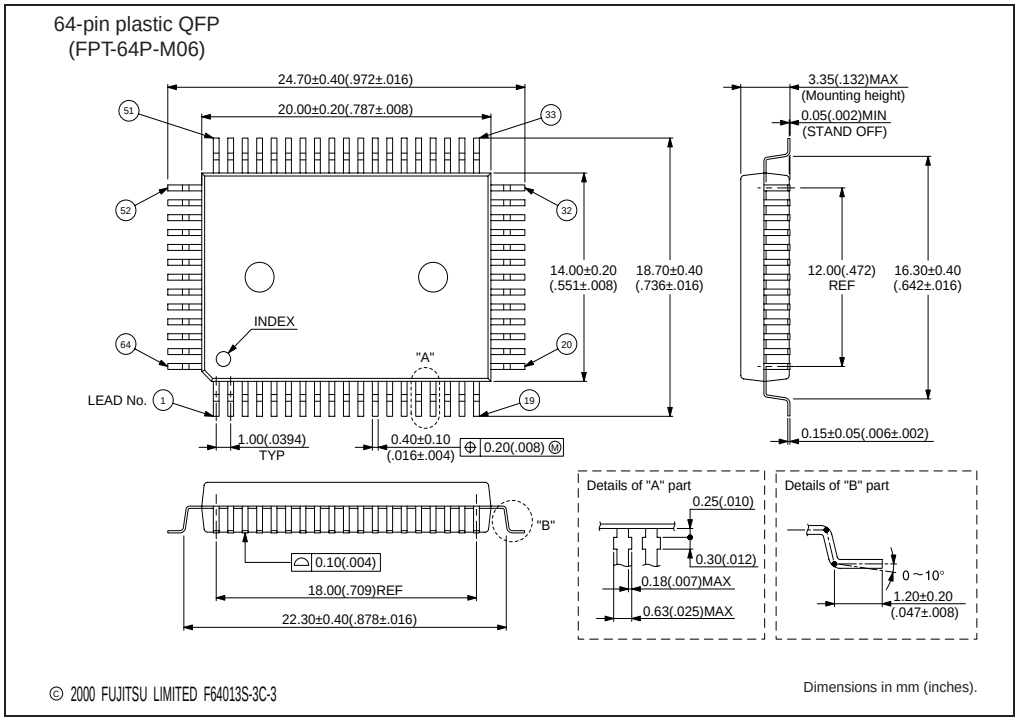
- **FPT-64P-M06**
- **FPT-64P-M09**
- **DIP-64P-M01 (Neither MB90568, MB90567 nor MB90F568 are supported.)**

These package dimensions are for reference only. For formal package dimensions, consult a Fujitsu representative.

■ Dimensions of FPT-64P-M06 Package

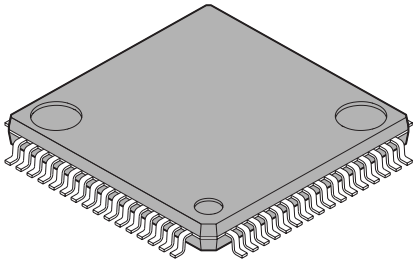
Figure 1.5-1 Dimensions of FPT-64P-M06 Package

64-pin plastic QFP  (FPT-64P-M06)	Lead pitch	1.00 mm
	Package width × package length	14 × 20 mm
	Lead shape	Gullwing
	Sealing method	Plastic mold
	Length of flat portion of pins	1.20 mm

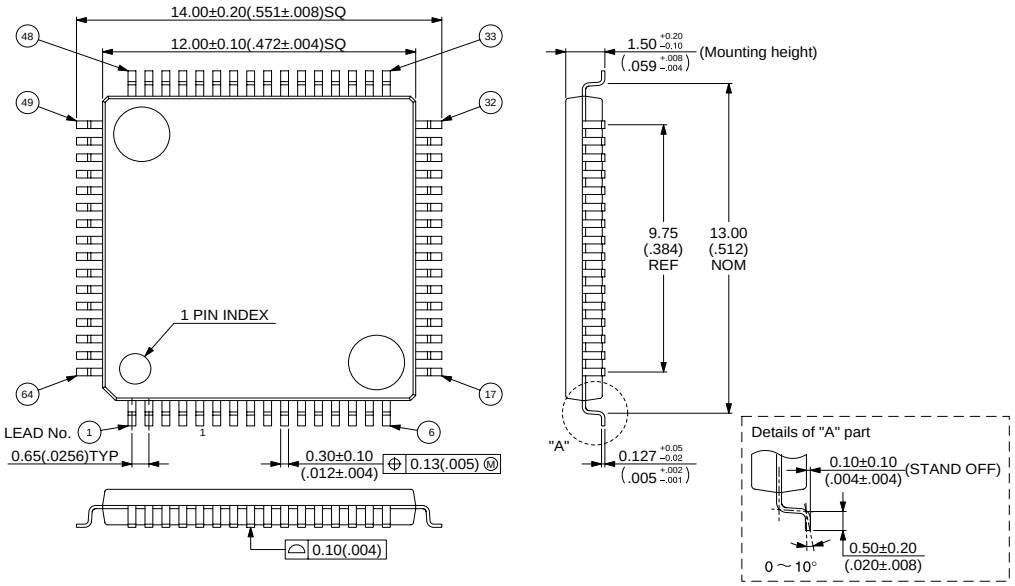


■ Dimensions of FPT-64P-M09 Package

Figure 1.5-2 Dimensions of FPT-64P-M09 Package

64-pin plastic LQFP  (FPT-64P-M09)	Lead pitch	0.65 mm
	Package width × package length	12 × 12 mm
	Lead shape	Gullwing
	Sealing method	Plastic mold

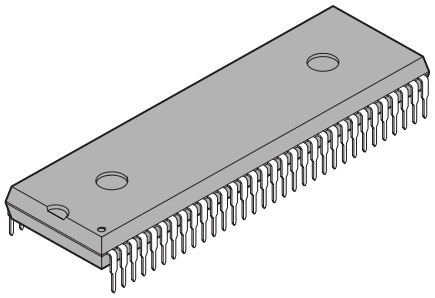
64-pin plastic LQFP
(FPT-64P-M09)



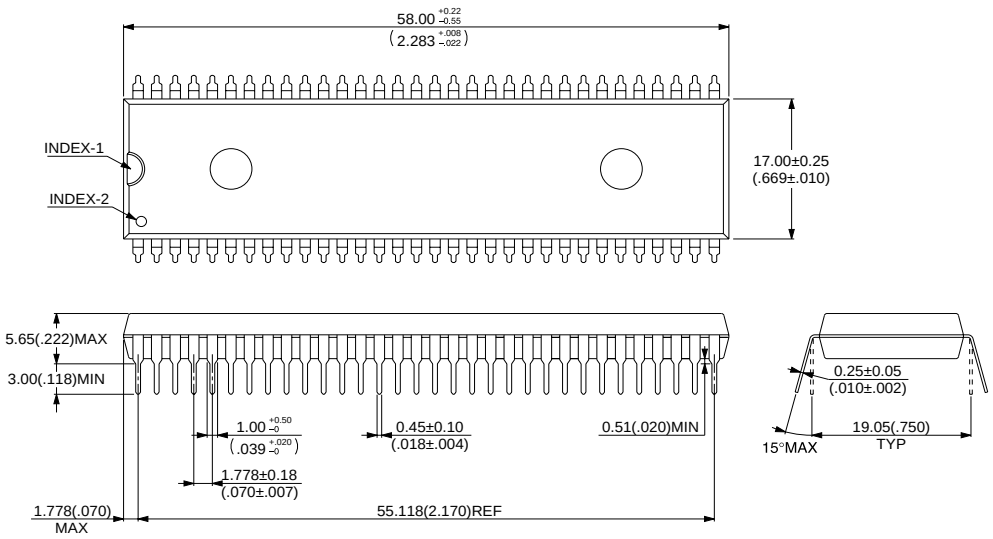
Dimensions in mm (inches).

■ Dimensions of DIP-64P-M01 Package

Figure 1.5-3 Dimensions of DIP-64P-M01 Package

64-pin plastic SH-DIP  (DIP-64P-M01)	Lead pitch	1.778mm(70mil)
	Row spacing	19.05mm(750mil)
	Sealing method	Plastic mold

64-pin plastic SH-DIP
(DIP-64P-M01)



Dimensions in mm (inches).

1.6 Pin Functions

Table 1.6-1 "Pin Functions" summarizes the pin names, circuit types, states at reset time, and functions.

■ **Pin Functions**

Table 1.6-1 Pin Functions

Pin No.			Pin name	Circuit type (*1)	State/ function at reset time	Function
QFPM06	QFPM09	SDIP				
23,24	22,23	30,31	X0·X1	A	Oscillating	Connect an oscillator to these pins. When an external clock is connected, leave the X1 pin open.
20	19	27	$\overline{\text{RST}}$	B	Reset input	External reset input pin
26 to 33	25 to 32	33 to 40	P00 to P07	C	Port input (Hi-z output)	General-purpose I/O ports
34 to 40	33 to 39	41 to 47	P10 to P16	C		General-purpose I/O ports
			INT0 to 6			Can be used as interrupt request input channels 0 to 6. Input is enabled when 1 is set in corresponding bits of EN0 to EN6 during standby mode to determine an input port. To use the pins, set a corresponding bit of the analog input enable register (ADER) to the port setting.
41	40	48	P17	C		General-purpose I/O port
			FRCK			External clock input pin for free-running timer. Input is enabled when the clock input of free-running timer is set to determine an input port. To use the pins, set a corresponding bit of the analog input enable register (ADER) to the port setting.

Table 1.6-1 Pin Functions (Continued)

Pin No.			Pin name	Circuit type (*1)	State/ function at reset time	Function
QFPM06	QFPM09	SDIP				
42	41	49	P20	D	Port input (Hi-z)	General-purpose I/O port
			TIN0			External clock input pin for reload timer channel 0. Input is enabled when the external clock input is set to determine an input port.
43	42	50	P21	D		General-purpose I/O port
			TO0			Event output pin for reload timer channel 0. Output is enabled when the event output enable is set.
44	43	51	P22	D		General-purpose I/O port
			TIN1			External clock input pin for reload timer channel 1. Input is enabled when the external clock input is set to determine an input port.
45	44	52	P23	D		General-purpose I/O port
			TO1			Event output pin for reload timer channel 1. Output is enabled when the event output enable is set.
46 to 49	45 to 48	53 to 56	P24 to 27	D		General-purpose I/O ports
			IN0 to 3			Trigger input pins for input capture channels 0 to 3. Input is enabled when the input capture trigger is set to determine an input port.
51 to 56	50 to 55	58 to 63	P30 to P35	E		General-purpose I/O ports
			RTO0 to 5			Output compare event output pins or waveform generator output pins. A waveform specified in the waveform generator is output. If no waveform is generated, set the output compare event output enable to output the output compare. To use the pins, set a corresponding bit of the ADER to the port setting.

Table 1.6-1 Pin Functions (Continued)

Pin No.			Pin name	Circuit type (*1)	State/ function at reset time	Function
QFPM06	QFPM09	SDIP				
59	58	2	P36	D	Port input (Hi-z)	General-purpose I/O port
			SIN0			Serial data input pin for UART channel 0. Do not use this pin as another input pin because it is used whenever UART channel 0 accepts input.
60	59	3	P37	D		General-purpose I/O port
			SOT0			Serial data output pin for UART channel 0. This function is enabled when UART channel 0 enables data output.
61	60	4	P40	D		General-purpose I/O port
			SCK0			Serial clock I/O pin for UART channel 0. This function is enabled when UART channel 0 enables clock output.
62 to 64, 1 to 3	61 to 64, 1,2	5 to 10	P41 to P46	D		General-purpose I/O ports
			PPG0 to 5			Output pins for PPG channels 0 to 5. This function is enabled when PPG channels 0 to 5 enable output.
4 to 11	3 to 10	11 to 18	P50 to P57	F	Analog input	General-purpose I/O ports
			AN0 to 7			A/D converter analog input pins. This function is enabled when the analog input specification is enabled. (ADER:bit0 to 7)
12	11	19	AVcc	-	Power input	Vcc power input pin for the A/D converter.
13	12	20	AVR	G	Referenc voltage input pin	Reference voltage input pin for the A/D converter. This voltage must not exceed Vcc. Vref- is fixed to AVss.
14	13	21	AVss	-	Power input	Vss power input pin for the A/D converter.

Table 1.6-1 Pin Functions (Continued)

Pin No.			Pin name	Circuit type (*1)	State/ function at reset time	Function
QFPM06	QFPM09	SDIP				
15	14	22	P60	D	Port input (Hi-z)	General-purpose I/O port
			SIN1			Serial data input pin for UART channel 1. Do not use this pin as another input pin because it is used whenever UART channel 1 accepts input.
16	15	23	P61	D		General-purpose I/O port
			SOT1			Serial data output pin for UART channel 1. This function is enabled when UART channel 1 enables data output.
17	16	24	P62	D		General-purpose I/O port
			SCK1			Serial clock I/O pin for UART channel 1. This function is enabled when UART channel 1 enables clock output.
18	17	25	P63	D		General-purpose I/O port
			INT7			Can be used as interrupt request input channel 7. Input is enabled when 1 is set in corresponding bits of EN7 during standby mode to determine an input port.
			DTT1			Pin level fixed input pin used when RT00 to RT05 pins are used. Input is enabled when the input enable is set in the waveform generator.
58	57	1	C (*2)	-	Capacitance pin power input	Capacitance pin for power stabilization. Connect a ceramic capacitor of about 0.1 μF to the outside.

Table 1.6-1 Pin Functions (Continued)

Pin No.			Pin name	Circuit type (*1)	State/ function at reset time	Function
QFPM06	QFPM09	SDIP				
19	18	26	MD0	B	Mode input pin	Input pin for operation mode specification. Connect this pin directly to Vcc or Vss.
21	20	28	MD1	B		Input pin for operation mode specification. Connect this pin directly to Vcc or Vss.
22	21	29	MD2	B		Input pin for operation mode specification. Connect this pin directly to Vcc or Vss.
25,50	24,49	32,57	Vss	-	Power input pin	Power (GND) input pin
57	56	64	Vcc	-		Power (5 V) input pin

*1: For information on the circuit types, see Section 1.7, "I/O Circuit Types."
*2: N.C. for the MB90565 series

1.7 I/O Circuit Types

Table 1.7-1 "I/O Circuit Types" summarizes the I/O circuit types of the MB90560/565 series.

■ **I/O Circuit Types9**

Table 1.7-1 I/O Circuit Types

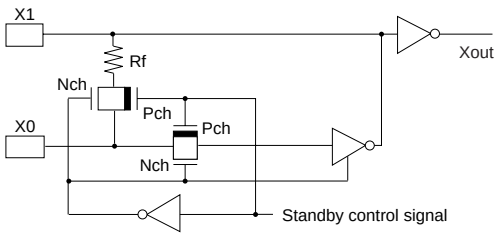
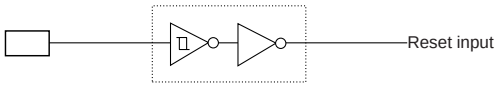
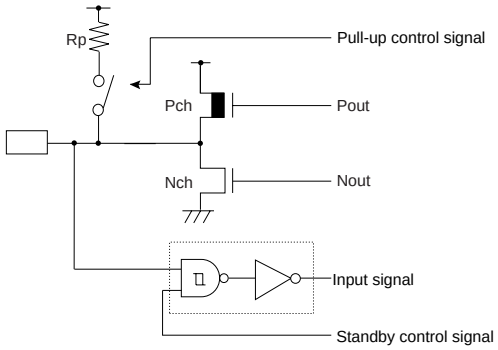
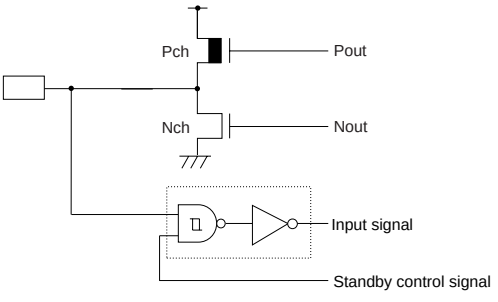
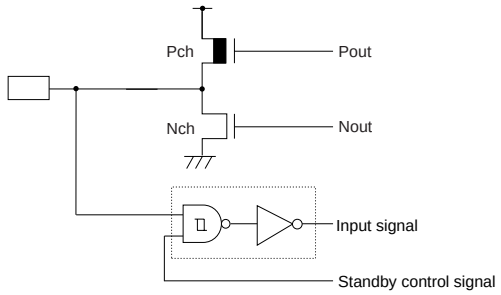
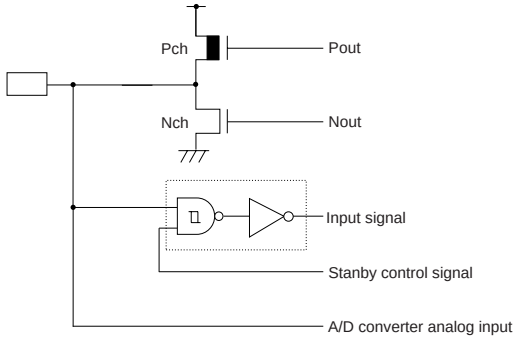
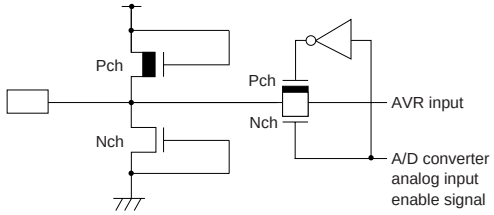
Classification	Circuit type	Remarks
A		• Oscillation circuit Built-in oscillation feedback resistance (Rf)
B		• Hysteresis reset input pin
C		• CMOS hysteresis I/O pin with pull-up control • CMOS output • CMOS hysteresis input (with standby control for input rejection) • Built-in pull-up resistance (Rp) **Note:** The pull-up resistance is enabled while the port is in the input status.
D		• CMOS hysteresis I/O pin • CMOS output • CMOS hysteresis input (with standby control for input rejection) **Note:** • Outputs of the I/O port and the built-in resources share one output buffer. • Inputs of the I/O port and the built-in resources share one input buffer.

Table 1.7-1 I/O Circuit Types (Continued)

Classification	Circuit type	Remarks
E		• CMOS I/O pin • CMOS output • CMOS input (with standby control for input rejection)

F		• Analog/CMOS hysteresis I/O pin
		• CMOS output
		• CMOS hysteresis input (with standby control for input rejection)
		• Analog input (The analog input of A/D converter is enabled when "1" is set in a corresponding bit of the analog input enable register (ADER).
		  • Outputs of the I/O port and the built-in resources share one output buffer.
		  • Inputs of the I/O port and the built-in resources share one input buffer.

G		• A/D converter (AVR) voltage input pin

1.8 Notes on Handling Devices

When handling devices, pay special attention to the following nine items:

- **Strict observation of maximum rated voltage (latchup prevention)**
- **Stabilization of supply voltage**
- **Power-on**
- **Treatment of unused input pins**
- **Treatment of A/D converter power pin**
- **External clock**
- **Power supply pin**
- **Analog power-on sequence of A/D converter**
- **Notes on using the "DIV A, Ri" or "DIVW A, RWi" instruction**

■ Notes on Handling Devices

❍ Be sure that the maximum rated voltage is not exceeded (latchup prevention).

Do not apply a voltage higher than Vcc or lower than Vss to the input and output terminals of the MB90560 or 565 series. Do not apply a voltage higher than the rating between Vcc and Vss. If a voltage higher than the rating is applied, a latchup may occur. A latchup may result in thermal damage to an element.

When turning the power to analog circuits, on or off be sure that the analog supply voltages (AVcc, AVRH, and AVcc) and analog input voltage do not exceed the digital supply voltage (Vcc).

❍ Stabilize the supply voltages.

Even within the operation guarantee range of the Vcc supply voltage, a malfunction can be caused if the supply voltage undergoes a rapid change. For voltage stabilization guidelines, the Vcc ripple fluctuations (P-P value) at commercial frequencies (50 to 60 Hz) should be suppressed to "10%" or less of the reference Vcc value. During a momentary change such as when switching a supply voltage, voltage fluctuations should also be suppressed so that the "transient fluctuation rate" is 0.1 V/ms or less.

❍ Power-on

To prevent a malfunction in the built-in voltage drop circuit, secure "50µS (between 0.2 V and 2.7 V)" or more for the voltage rise time during power-on.

❍ Treatment of unused input pins

An unused input pin, if left open, may cause a malfunction or a permanent damage due to a latchup. Every unused input pin must be pulled up or down using resistance of 2 kΩ or more.

An unused I/O pin must be opened by setting it to output mode or handled in the same way as an input pin after it is set to input mode.

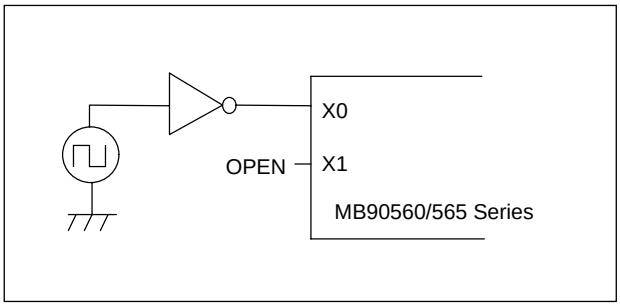
❍ **Treatment of A/D converter power pin**

When the A/D converter is not used, connect the pins as follows: AVcc =Vcc, AVss = AVRH = AVRL = Vss.

❍ **Notes on external clock**

When an external clock is used, the oscillation stabilization wait time is required at power-on reset or at cancellation of subclock mode or stop mode. When an external clock is used, connect only the X0 pin and leave the X1 pin open.

Figure 1.8-1 Sample Application of External Clock



❍ **Power supply pins**

When a device has two or more Vcc or Vss pins, the pins that should have equal potential are connected within the device in order to prevent a latchup or other malfunction. To reduce extraneous emission, to prevent a malfunction of the strobe signal due to an increase in the group level, and to maintain the local output current rating, connect all these power supply pins to an external power supply and to the same ground.

The current source should be connected to the Vcc and Vss pins of the device with minimum impedance. It is recommended that a bypass capacitor of about 0.1μF be connected near the terminals between Vcc and Vss.

❍ **Power-on and power-off sequences**

The power to the A/D converter (AVcc, AVR, AVcc) and analog inputs (AN0 to AN7) must be turned on after the power to the digital circuits (Vcc) is turned on. When turning off the power, turn off the power to the digital circuits (Vcc) after turning off the power to the A/D converter and analog inputs. When the power is turned on or off, AVR should not exceed AVcc.

Also, when a pin that is used for analog input is also used as an input port, the input voltage should not exceed AVcc. (The power to the analog circuits and the power to the digital circuits can be simultaneously turned on or off.)

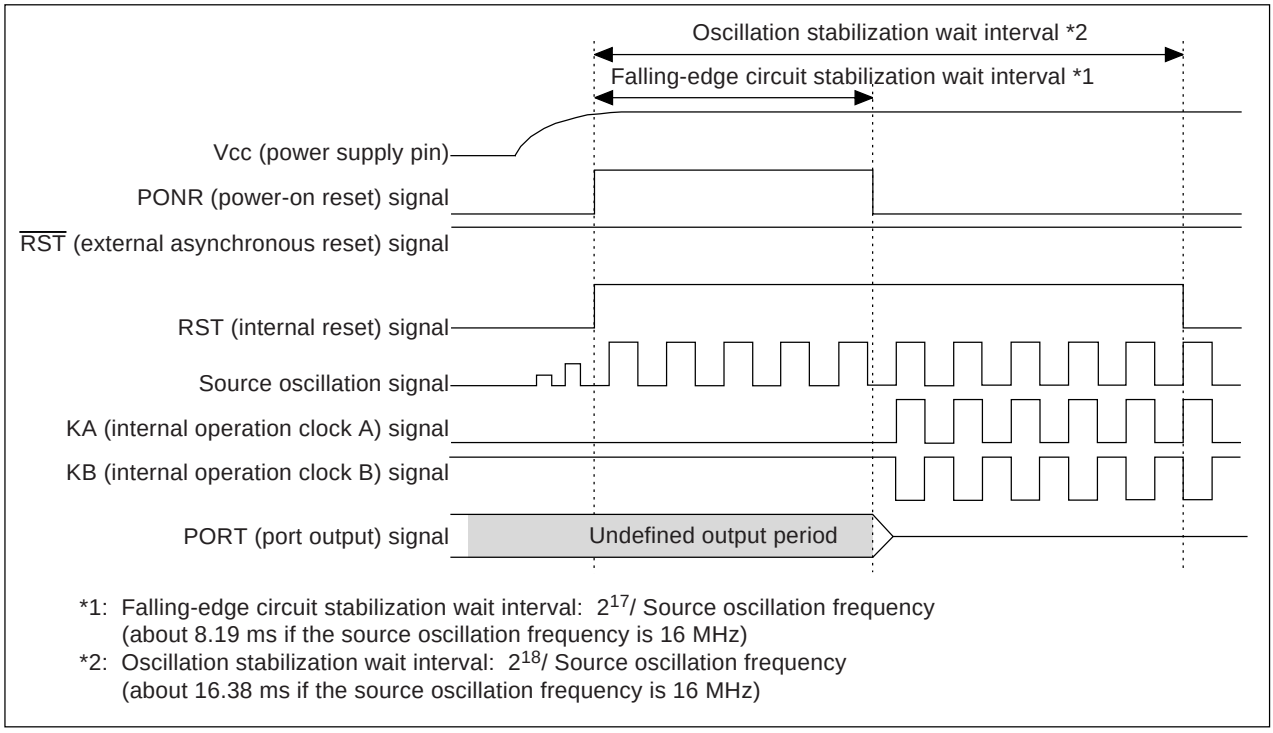
❍ **Undefined outputs from Ports 0 and 1**

Ports 0 and 1 output undefined signals if the $\overline{\text{RST}}$ pin is "H" during the oscillation stabilization wait interval of the falling-edge circuit (during power-on rest) after power-on. If the $\overline{\text{RST}}$ pin is "L", Ports 0 and 1 enter the high-impedance state.

Note that the timing is as shown in Figure 1.8-2 "Timing Chart for Undefined Outputs from Ports 0 and 1 (if the RST Terminal is "H")" and Figure 1.8-3 "Timing Chart for High Impedance State of Ports 0 and 1 (if the RST Terminal is "L")" (applicable models: MB90F562/B and MB90V560).

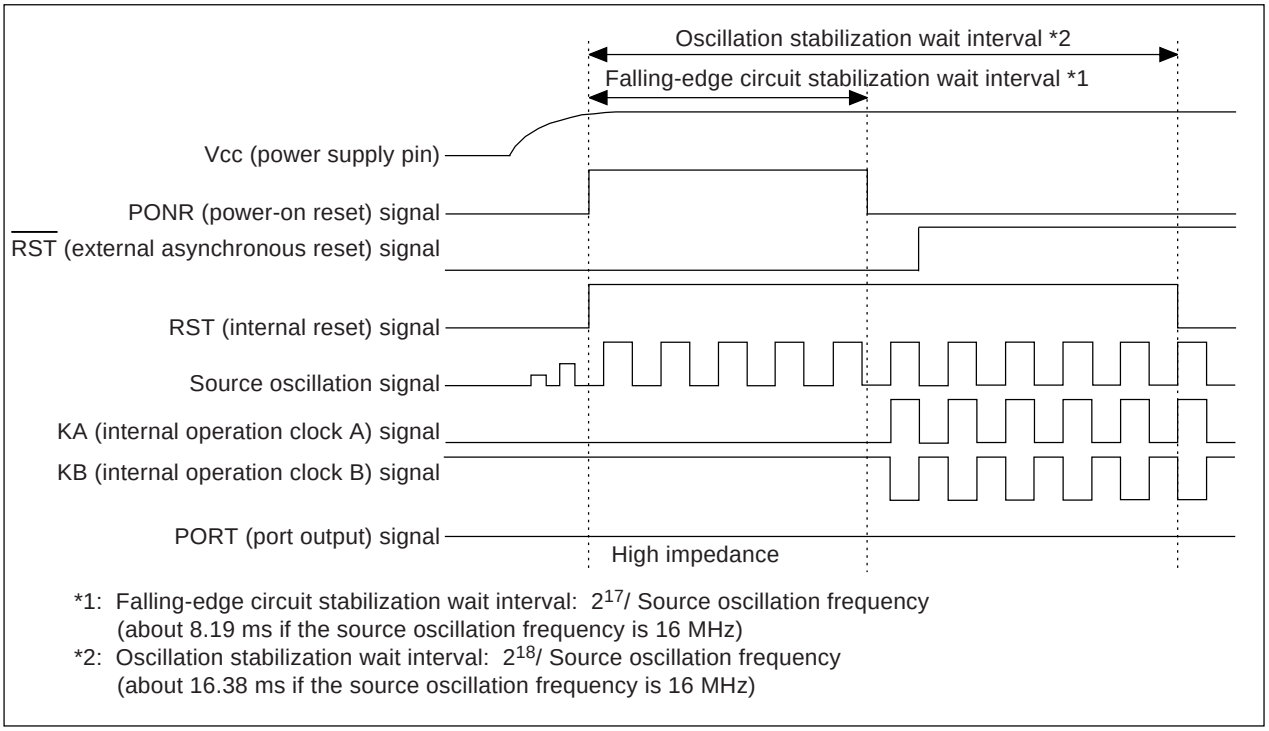
On a model without a falling-edge circuit, Ports 0 and 1 do not output undefined signals because there is no oscillation stabilization wait interval (applicable models: MB90561/A, MB90562/A, MB90567/8, and MB90F568).

Figure 1.8-2 Timing Chart for Undefined Outputs from Ports 0 and 1 (if the RST Terminal is "H")



*1: Falling-edge circuit stabilization wait interval: 2^{17}/ Source oscillation frequency
(about 8.19 ms if the source oscillation frequency is 16 MHz)
*2: Oscillation stabilization wait interval: 2^{18}/ Source oscillation frequency
(about 16.38 ms if the source oscillation frequency is 16 MHz)

Figure 1.8-3 Timing Chart for High Impedance State of Ports 0 and 1 (if the RST Terminal is "L")



*1: Falling-edge circuit stabilization wait interval: 2^{17}/ Source oscillation frequency
(about 8.19 ms if the source oscillation frequency is 16 MHz)
*2: Oscillation stabilization wait interval: 2^{18}/ Source oscillation frequency
(about 16.38 ms if the source oscillation frequency is 16 MHz)

❍ **Notes on using the "DIV A, Ri" or "DIVW A, RWi" instruction**

The remainder obtained by executing the signed multiplication and division instruction, "DIV A, Ri" or "DIVW A, RWi" is affected by the bank register and stored at an address of the memory bank specified in the bank register.

Therefore, the "DIV A, Ri" or "DIVW A, RWi" instruction should be used after setting the

corresponding bank register value to '00_H'.

Reference:

For more information, see Sections 2.9.5, "Notes on Using the "DIV A, Ri" or "DIVW A, RWi" Instruction."

For more information on the bank register, see Section 2.7.9, "Bank Registers (PCB, DTB, USB, SSB, ADB)."

❍ **Using REALOS**

The extended intelligent I/O Service (EI^2OS) is disabled if REALOS is used.

CHAPTER 2 CPU

This chapter describes the functions and operations of the CPU of the MB90560/565 series.

2.1 CPU

The F^{2}MC-16LX CPU core is a 16-bit CPU designed for use in applications, such as welfare and mobile equipment, which require high-speed real-time processing. The instruction set of the F^{2}MC-16LX was designed for controllers so that it can perform various types of control at high speeds and efficiencies.

The F^{2}MC-16LX CPU core processes 32-bit data using a built-in 32-bit accumulator. Memory space, which can be extended to up to 16M bytes, can be accessed in either linear or bank access mode. The instruction set inherits the AT architecture of F^{2}MC-8L and F^{2}MC-16L and has additional instructions supporting C language. In addition, it has an extended addressing mode, enhanced multiply/divide instructions, and reinforced bit manipulation instructions. The features of the F^{2}MC-16XL CPU are shown below:

■ CPU

- ❍ **Minimum instruction execution time: 62.5 ns (source oscillation at 4 MHz and clock multiplication by 4)**

- ❍ **Maximum memory address space: 16M bytes. Access in linear or bank mode**

- ❍ **Instruction set optimum for controller applications**
  - Many data types (bit, byte, word, and long word)
  - As many as 23 addressing modes
  - Enhanced high-precision arithmetic operation by a 32-bit accumulator
  - Enhanced signed multiply/divide instructions and RETI instruction function

- ❍ **Interrupt function**

  Eight programmable priority levels

- ❍ **Automatic transfer function independent to CPU**

  Extended intelligent I/O service using up to 16 channels

- ❍ **Instruction set supporting high-level language (C) and multi-tasking**

  System stack pointer, instruction set symmetry, and barrel shift instructions

- **Increased execution speed: 4-byte instruction queue**

Note:

The MB90560/565 series runs only in single-chip mode so only internal ROM and RAM and internal peripheral memory space can be accessed.

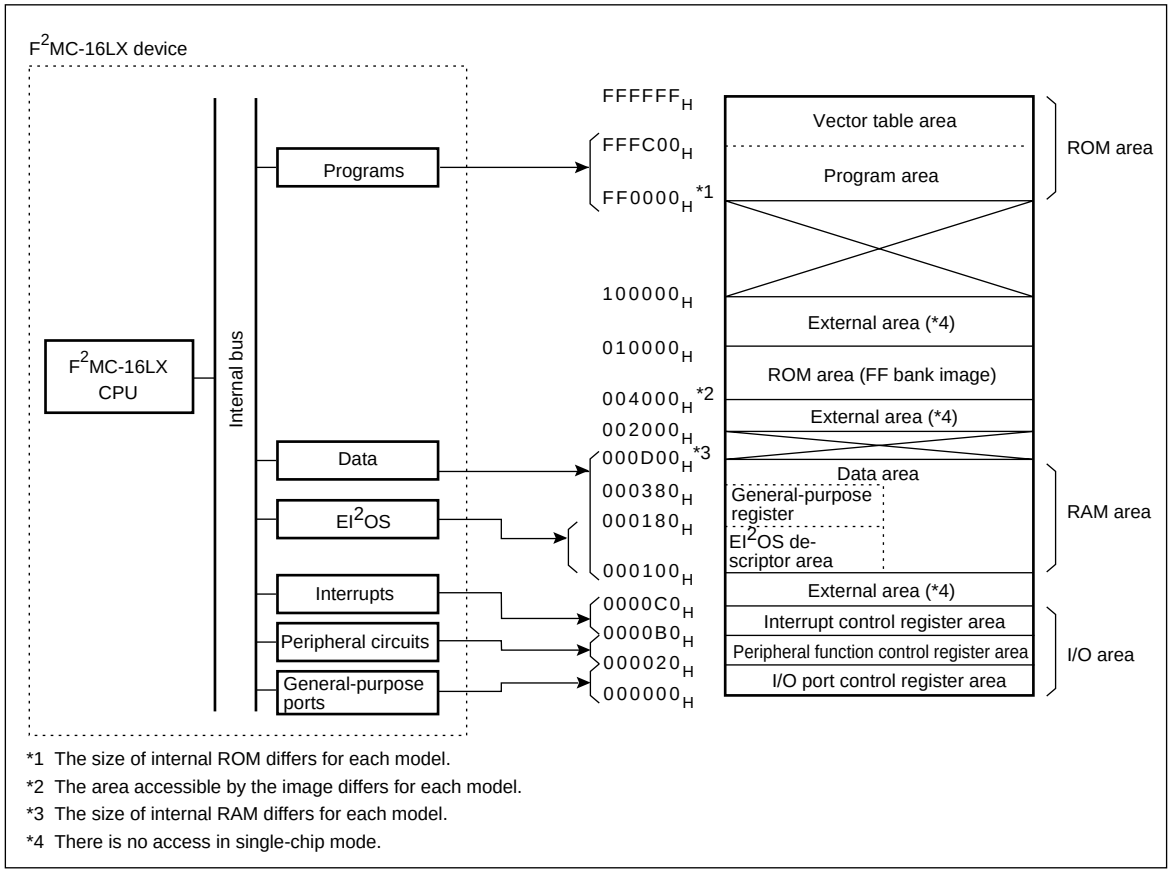
2.2 Memory Space

All I/O, programs, and data are located in the 16M-byte memory space of the F^{2}MC-16LX. The RAM space is used for extended intelligent I/O service (EI^{2}OS) descriptors, general-purpose registers, and vector tables.

■ Memory Space

All I/O, programs, and data are located in the 16M-byte memory space of the F^{2}MC-16LX CPU. The CPU is able to access each built-in peripheral function (resource) through a memory space address indicated by the 24-bit address bus.

Figure 2.2-1 Sample Relationship between the MB90560/565 Series System



*1 The size of internal ROM differs for each model.
*2 The area accessible by the image differs for each model.
*3 The size of internal RAM differs for each model.
*4 There is no access in single-chip mode.

■ ROM Area

❍ Vector table area (address: "FFFC00$_H$ to FFFFFF$_H$")

- This area is used as a vector table for vector call instructions, interrupt vectors, and reset vectors.
- This area is allocated at the highest addresses of the ROM area. The start address of the corresponding processing routine is stored in each vector table address.

❍ Program area (address: Up to "FFFBFF$_H$")

- Mask ROM (or flash memory) is built in as an internal program area.
- The size of the internal ROM (or flash memory) differs for each model.

■ RAM Area

❍ Data area (address: From "000100$_H$")

- The static RAM is built in as an internal data area.
- The size of internal RAM differs for each model.

❍ General-purpose register area (address: "000180$_H$ to 00037F$_H$")

- Auxiliary registers used for 8-bit, 16-bit, and 32-bit arithmetic operations and transfer are allocated in this area.
- When this area is not used as a general purpose register, it can be used as ordinary RAM.
- When this area is used as a general-purpose register, general-purpose register addressing enables high-speed access within a few instruction cycles.

❍ Extended intelligent I/O service (EI^{2}OS) descriptor area (address: "000100$_H$ to 00017F$_H$")

- This area retains the transfer modes, I/O addresses, transfer count, and buffer addresses of the Extended Intelligent I/O Service (EI^{2}OS).
- If the Extended Intelligent I/O Service (EI^{2}OS) is not used, this area can be used as ordinary RAM.

■ I/O Area

❍ Interrupt control register area (address: "0000B0$_H$ to 0000BF$_H$")

The interrupt control registers (ICR00 to ICR15) correspond to the built-in peripheral functions that have an interrupt function. These registers set interrupt levels and control the extended intelligent I/O service (EI^{2}OS).

❍ Peripheral function control register area (address: "000020$_H$ to 0000AF$_H$")

This register controls the built-in peripheral functions (resources).

❍ I/O port control register area (address: "000000$_H$ to 00001F$_H$")

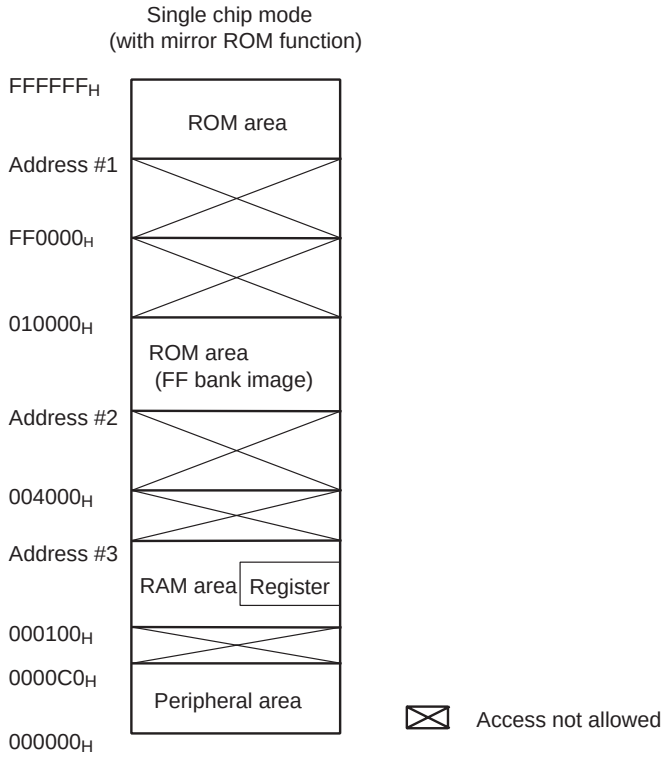
This register controls the I/O port.

2.3 Memory Maps

This section shows the memory map for each model of MB90560/565 series.

■ Memory Maps

Figure 2.3-1 Memory Maps



Model	Address #1	Address #2	Address #3
MB90561/A	FF8000$_H$	008000$_H$	000500$_H$
MB90562/A	FF0000$_H$	004000$_H$	000900$_H$
MB90F562/B	FF0000$_H$	004000$_H$	000900$_H$
MB90567	FE8000$_H$	004000$_H$	001100$_H$
MB90568	FE0000$_H$	004000$_H$	001100$_H$
MB90F568	FE0000$_H$	004000$_H$	001100$_H$
MB90V560	FF0000$_H$ *	004000$_H$ *	001100$_H$

*1: The MB90V560 does not contain ROM. Assume these areas as its ROM areas for development purposes.

Note:

- If the ROM mirroring function register is set, the data in the upper side of FF bank ("$FF4000_H$ to $FFFFFF_H$") can be seen as a mirror image in the upper side of 00 bank ("004000_H to $00FFFF_H$").
- For information on setting the ROM mirroring function, see Chapter 18 "ROM MIRRORING FUNCTION SELECTION MODULE".

Reference:

The ROM mirroring function allows the small model of the C compiler to be used.

The low-order 16-bit address of FF bank becomes the same as the low-order 16-bit address of 00 bank. However, not all the data in the ROM area can be seen as a mirror image in 00 bank because the ROM area of FF bank exceeds 48K bytes.

To use the small model of the C compiler, store the data table in "$FF4000_H$ to $FFFFFF_H$" to show the data table as a mirror image in "004000_H to $00FFFF_H$". Thus, the data table in the ROM area can be referenced without the "far" specification declared in the pointer.

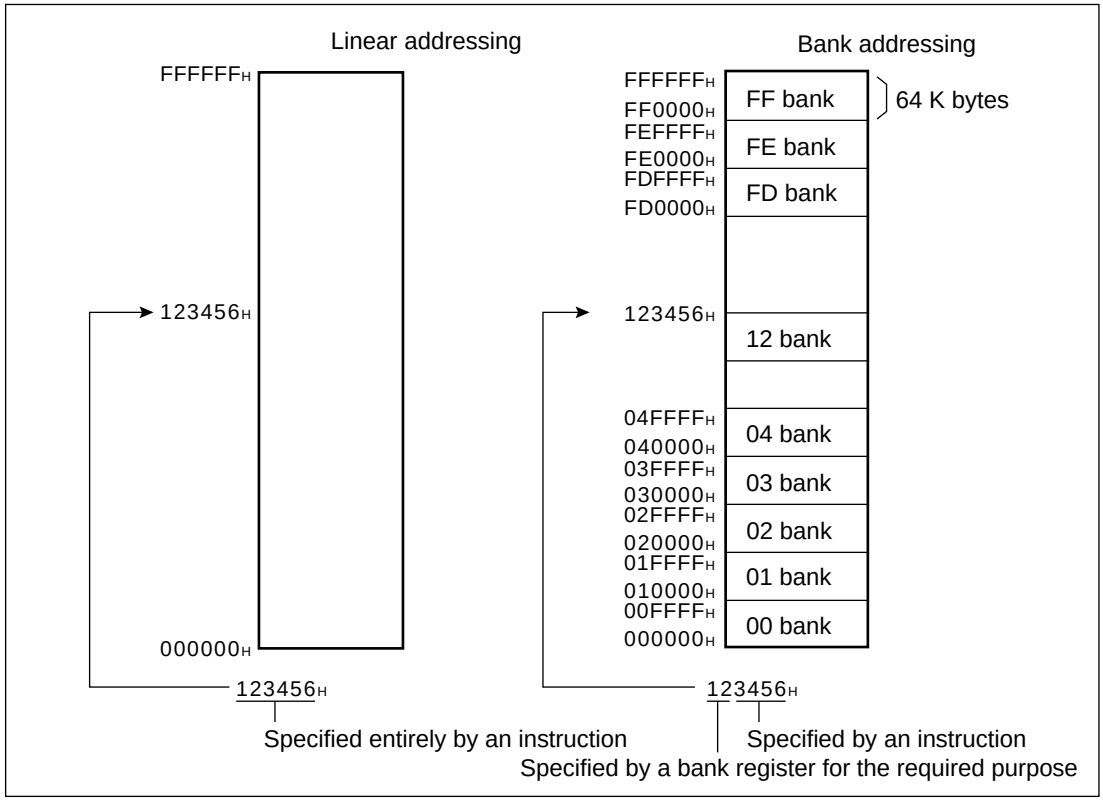
2.4 Addressing

Addresses can be generated using linear addressing and bank addressing.
In linear addressing, the 16M-byte space is directly specified using a continuous 24-bit address.
In bank addressing, the 16M-byte space is divided into 256 64K-byte banks. The upper 8 bits of the address are specified by a bank register and the lower 16 bits of the address are directly specified by an instruction.

The F^{2}MC-16LX series uses different address generation methods depending on the instruction.

■ Linear Addressing and Bank Addressing

Figure 2.4-1 Linear Addressing and Bank Addressing Memory Management



2.4.1 Address Specification by Linear Addressing

The linear addressing has two types of address specified:

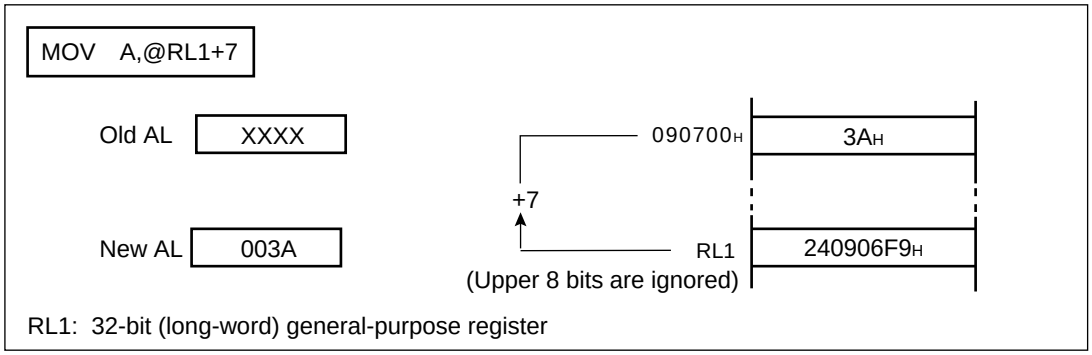
- **Specify 24-bit address directly in the instruction as operand**
- **Specify 24-bit address in a 32-bit general-purpose registers, using the lower 24 bits**

■ **Linear Addressing by 24-Bit Operand Specification**

Figure 2.4-2 Example of Direct Specification of a 24-bit Physical Address in Linear Addressing

■ **Addressing by Indirect Specification with a 32-Bit Register**

Figure 2.4-3 Example of Indirect Specification with a 32-bit General-Purpose Register in Linear Addressing



2.4.2 Address Specification by Bank Addressing

In address specification by bank addressing, the 16M-byte space is divided into 256 64K-byte banks. The upper 8 bits of the address are specified by a bank register and the lower 16 bits of the address are directly specified by an instruction.
The five types of bank registers classified by function are as follows:

- **Program bank register (PCB)**
- **Data bank register (DTB)**
- **User stack bank register (USB)**
- **System stack bank register (SSB)**
- **Additional bank register (ADB)**

■ Bank Registers and Access Space

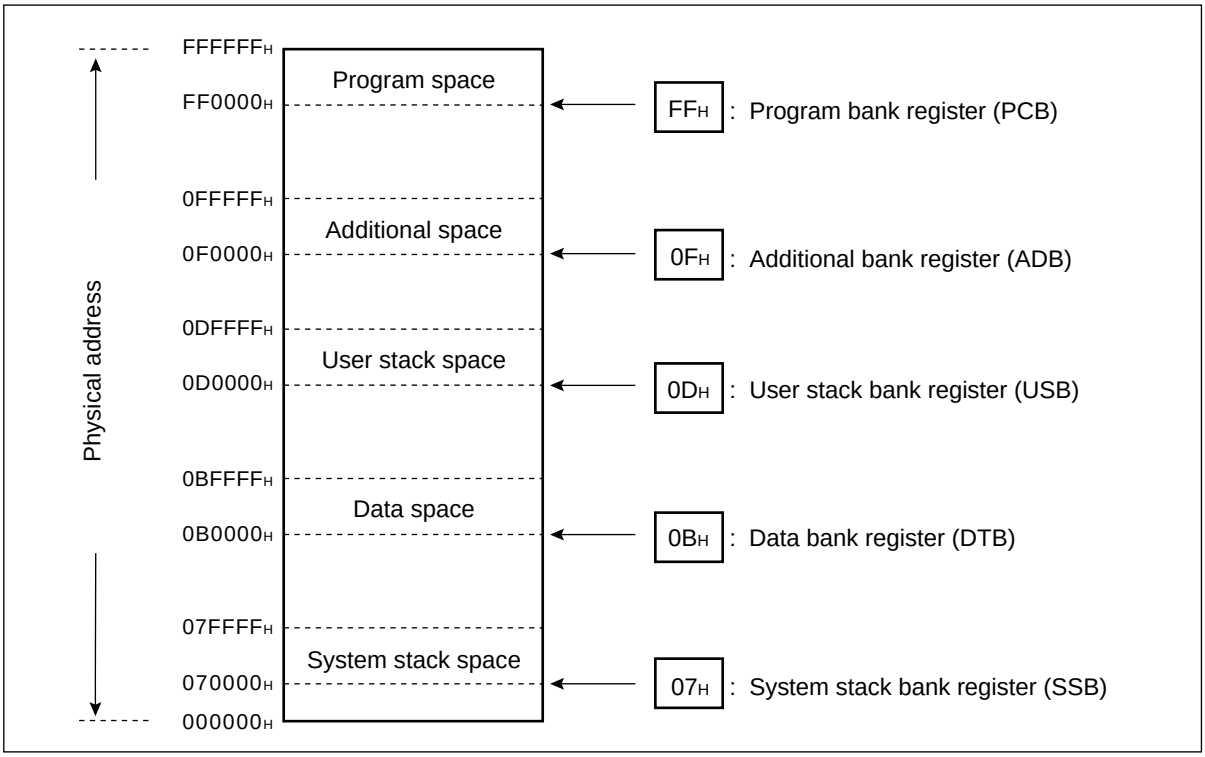
Table 2.4-1 Access Space and Main Function of Each Bank Register

Bank register name	Access space	Main function	Initial value after a reset
Program bank register (PCB)	Program (PC) space	Instruction codes, vector tables, and immediate-value data are stored.	FF_H
Data bank register (DTB)	Data (DT) space	Read/write data is stored. Internal or external peripheral control registers and data registers are accessed.	00_H
User stack bank register (USB)	Stack (SP) space	This area is used for stack accesses such as when PUSH/POP instructions and interrupt registers are saved. The SSB is used when the stack flag in the condition register (CCR:S) is 1. The USB is used when the stack flag in the condition register (CCR:S) is 0. (*1)	00_H
System stack bank register (SSB) (*1)			00_H
Additional bank register (ADB)	Additional (AD) space	Data that overflows from the data (DT) space is stored.	00_H

*1 The SSB is always used as an interrupt stack.

See Section 2.7.9 "Bank Registers (PCB, DTB, USB, SSB, ADB)" for details.

Figure 2.4-4 Sample Bank Addressing



■ Bank Addressing and Default Space

To improve instruction coding efficiency, each instruction has a defined default space for each addressing method, as shown in Table 2.4-2 "Addressing and Default Spaces". To use a space other than the default space, specify a prefix code for a bank before the instruction. This enables the bank space that corresponds to the specified prefix code to be accessed.See Section 2.9 "Prefix Codes" for details about prefix codes.

Table 2.4-2 Addressing and Default Spaces

Default space	Addressing
Program space	PC indirect, program access, branching
Data space	Addressing using @RW0, @RW1, @RW4, and @RW5, @A, addr16, dir
Stack space	Addressing using PUSHW, POPW, @RW3, and @RW7
Additional space	Addressing using @RW2 and @RW6

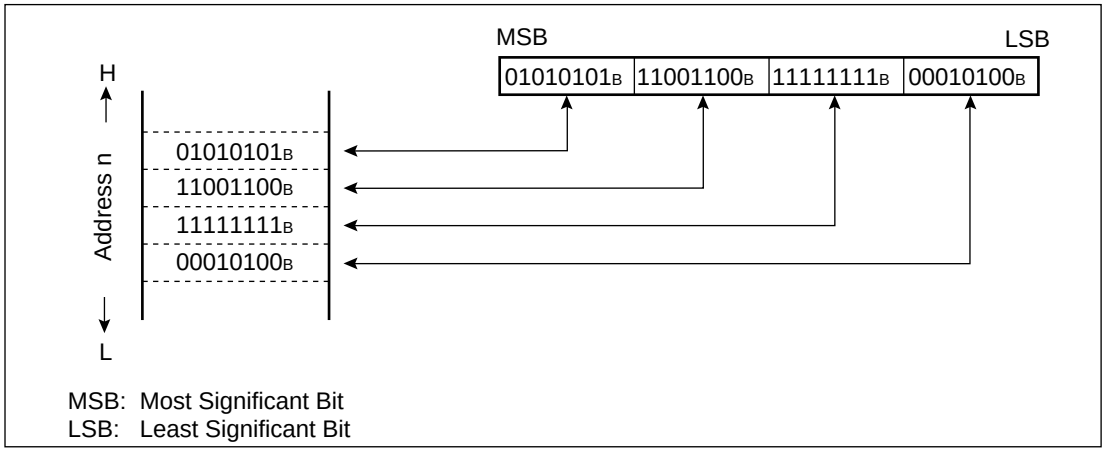
2.5 Memory Location of Multibyte Data

Multibyte data is written to memory sequentially from the lower address. If multibyte data is 32-bit data, the lower 16 bits are transferred followed by the upper 16 bits. If a reset signal is input immediately after the low-order data is written, the high-order data may not be written.

■ Storage of Multibyte Data in RAM

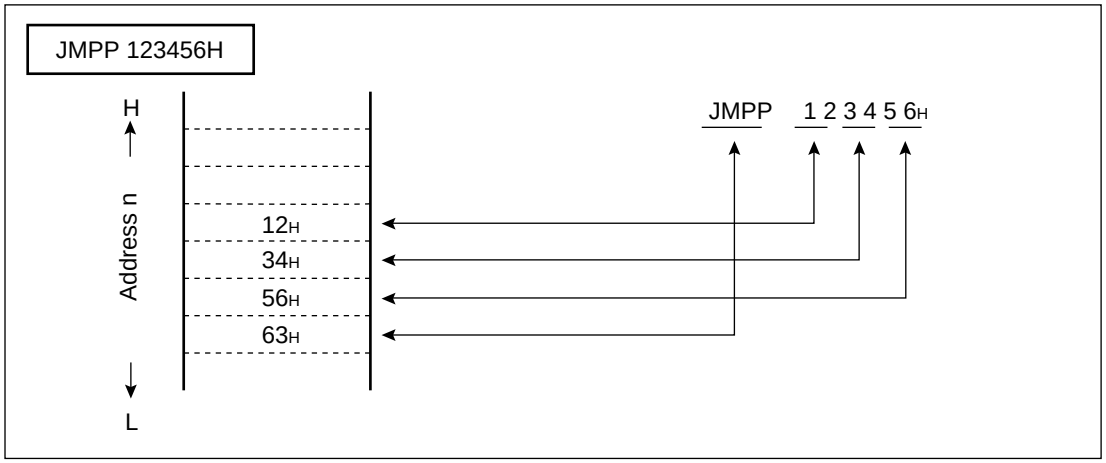
The lower 8 bits of the data is located at address n, and subsequent data is located at address n + 1, address n + 2, address n + 3, and so on, in this sequence.

Figure 2.5-1 Storage of Multibyte Data in RAM



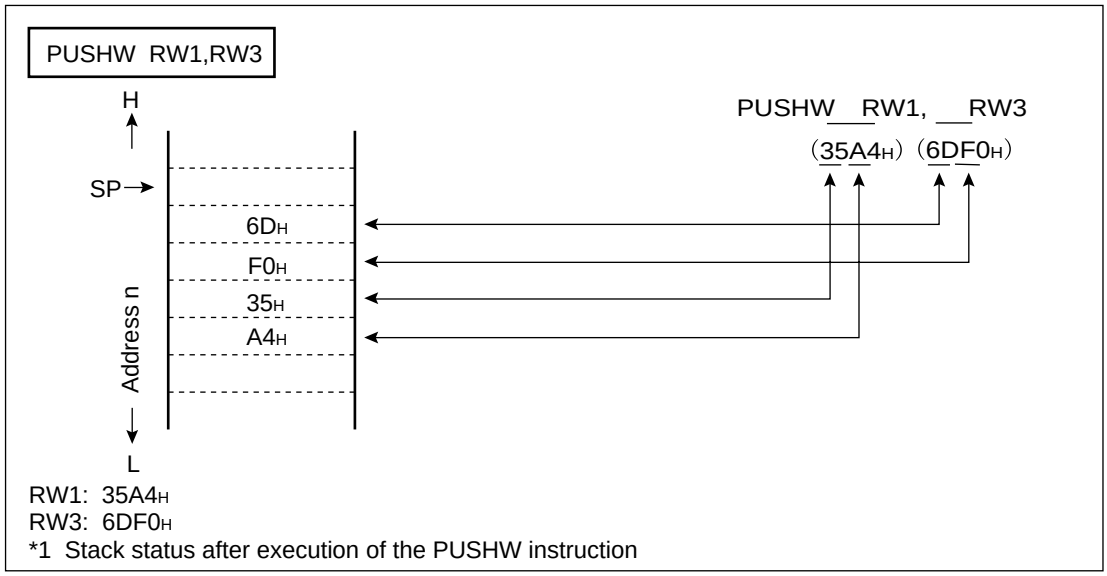
■ Storage of Multibyte Operand

Figure 2.5-2 Storage of a Multibyte Operand in RAM



■ Storage of Multibyte Data in a Stack

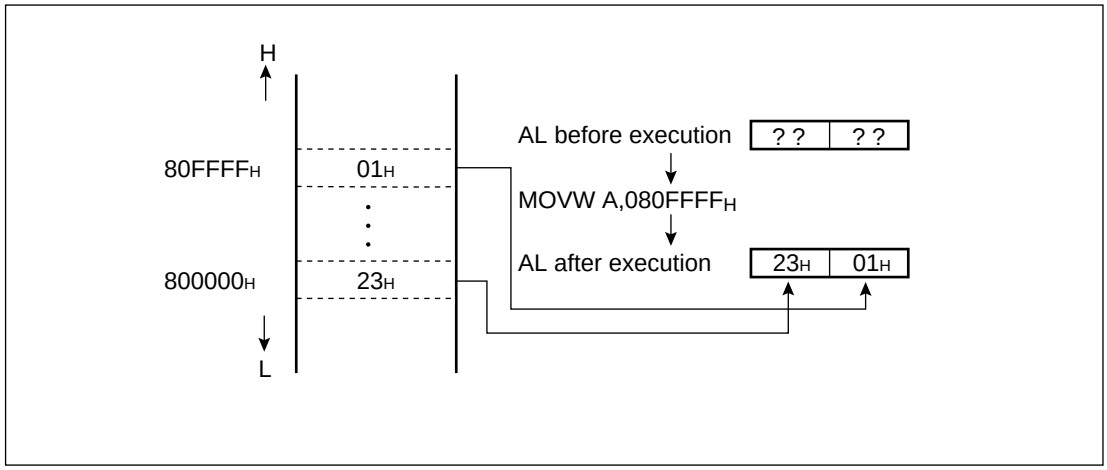
Figure 2.5-3 Storage of Multibyte Data in a Stack



■ Multibyte Data Access

Multibyte data is generally accessed within a bank. For an instruction that accesses multibyte data, the address "$FFFF_H$" is followed by "0000_H" in the same bank.

Figure 2.5-4 Multibyte Data Access on a Bank Boundary



2.6 Registers

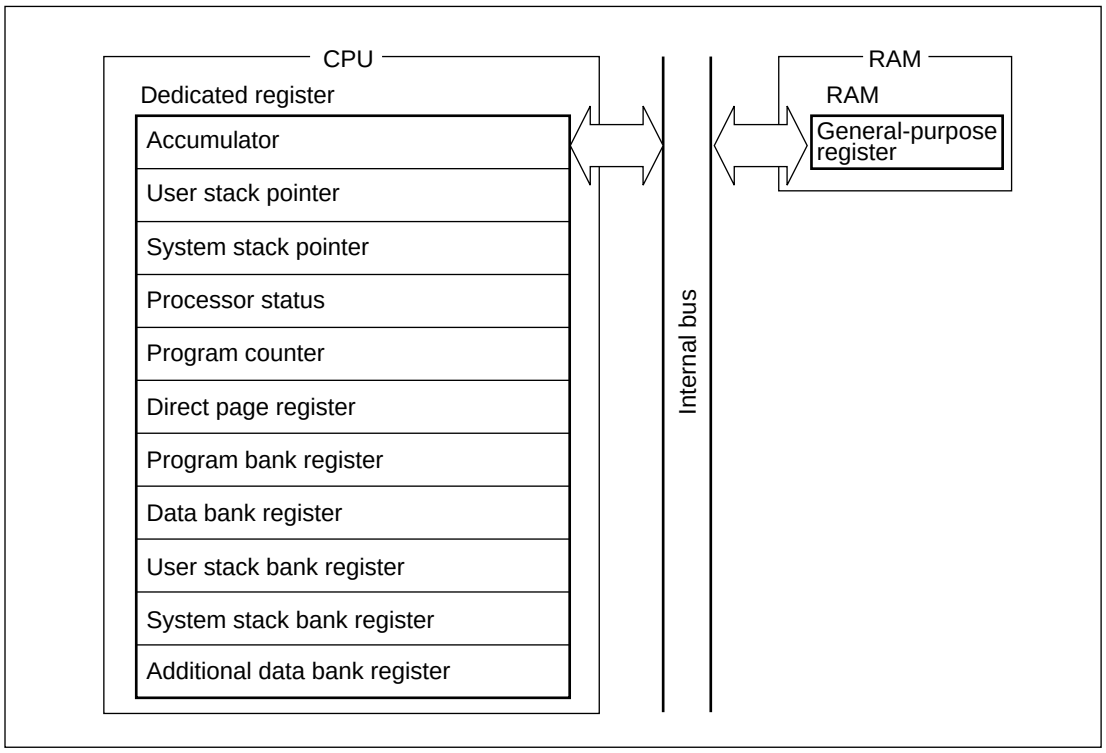
F^{2}MC-16LX registers are classified into internal dedicated CPU registers and built-in RAM general-purpose registers.

■ Dedicated Registers and General-Purpose Registers

Dedicated registers are built-in hardware in the CPU.

General-purpose registers coexist with RAM in the CPU address space.

Figure 2.6-1 Dedicated Registers and General-Purpose Registers



2.7 Dedicated Registers

The following 11 registers are dedicated registers in the CPU.

- **Accumulator (A)**
- **User stack pointer (USP)**
- **System stack pointer (SSP)**
- **Processor status (PS)**
- **Program counter (PC)**
- **Direct page register (DPR)**
- **Program bank register (PCB)**
- **Data bank register (DTB)**
- **User stack bank register (USB)**
- **System stack bank register (SSB)**
- **Additional data bank register (ADB)**

■ Configuration of Dedicated Registers

Figure 2.7-1 Configuration of Dedicated Registers

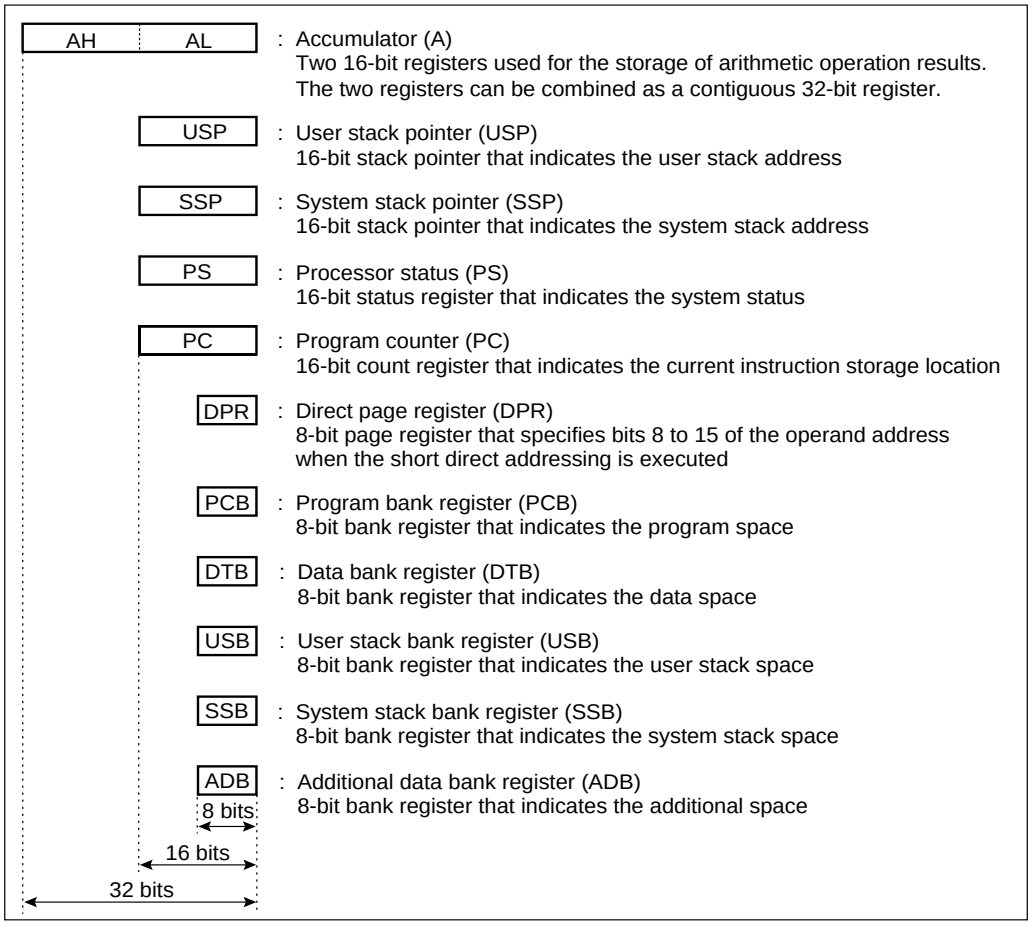
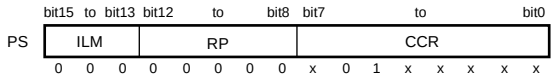


Table 2.7-1 Initial Values of the Dedicated Registers

Dedicated register	Initial value
Accumulator (A)	Undefined
User stack pointer (USP)	Undefined
System stack pointer (SSP)	Undefined
Processor status (PS)	
Program counter (PC)	Value in reset vector (contents of $FFFFDC_H$, $FFFFDD_H$)
Direct page register (DPR)	01_H
Program bank register (PCB)	Value in reset vector (contents of $FFFFDE_H$)
Data bank register (DTB)	00_H
User stack bank register (USB)	00_H
System stack bank register (SSB)	00_H
Additional data bank register (ADB)	00_H

2.7.1 Accumulator (A)

The accumulator (A) consists of two 16-bit arithmetic operation registers (AH and AL). The accumulator is used to temporarily store the results of an arithmetic operation and data.
The accumulator (A) can be used as a 32-bit, 16-bit, or 8-bit register. Arithmetic operations can be performed between memory and other registers or between the higher 16-bit arithmetic operation register (AH) and the lower 6-bit arithmetic operation register (AL). The A register has a data retention function: When data not longer than a word is transferred to the AL register, data stored in the AL register before the transfer is transferred to the AH register. (Data is not retained for some instructions.)

■ Accumulator (A)

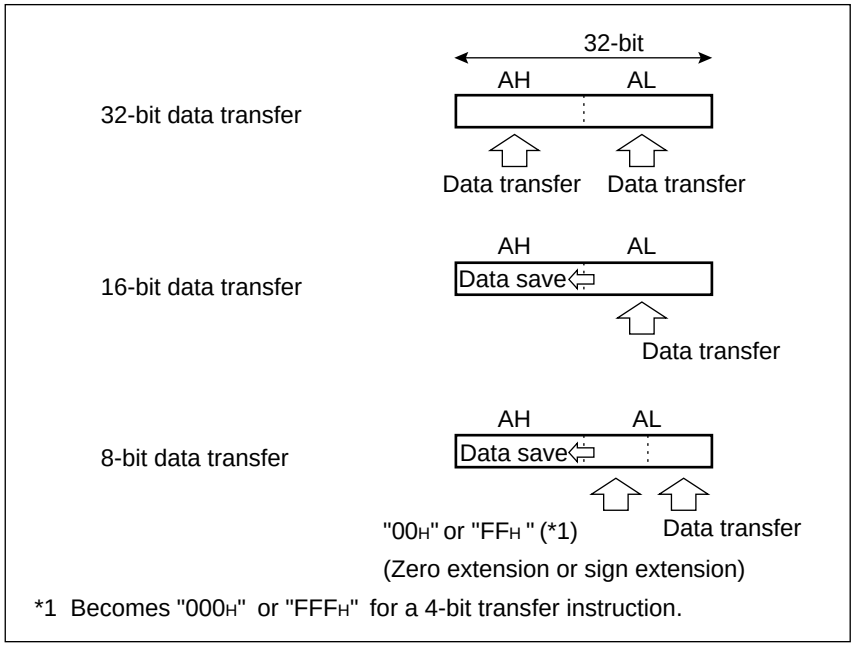
❍ Data transfer to the accumulator

The accumulator (A) can handle 32-bit (long word), 16-bit (word), and 8-bit (byte) data. The four-bit data transfer instruction (MOVN) is exceptionally provided but the data is processed in the same way as that for 8-bit data.

- For 32-bit data processing, the AH and AL registers are used in combination.
- For 16-bit and 8-bit data, the AL register is used while the AH register retains data in the AL register.
- Data not longer than a byte, when transferred to the AL register, becomes 16 bits long through sign or zero extension and is stored in the AL register. Data stored in the AL register can be handled as 16-bit or 8-bit data.

Figure 2.7-3 "Example of AL-AH Transfer in the Accumulator (A) (8-bit Immediate Value, Zero Extension)" to Figure 2.7-6 "Example of AL-AH Transfer in the Accumulator (A) (16 bits, Register Indirect)" show specific examples of transfer.

Figure 2.7-2 Data Transfer to the Accumulator



❍ **Accumulator byte-processing arithmetic operation**

When a byte-processing arithmetic operation instruction is executed for the AL register, the upper 8 bits of the AL register before the arithmetic operation is executed are ignored. The upper 8 bits of the arithmetic operation results are all zeros.

❍ **Initial value of the accumulator**

The initial value after a reset is undefined.

Figure 2.7-3 Example of AL-AH Transfer in the Accumulator (A) (8-bit Immediate Value, Zero Extension)

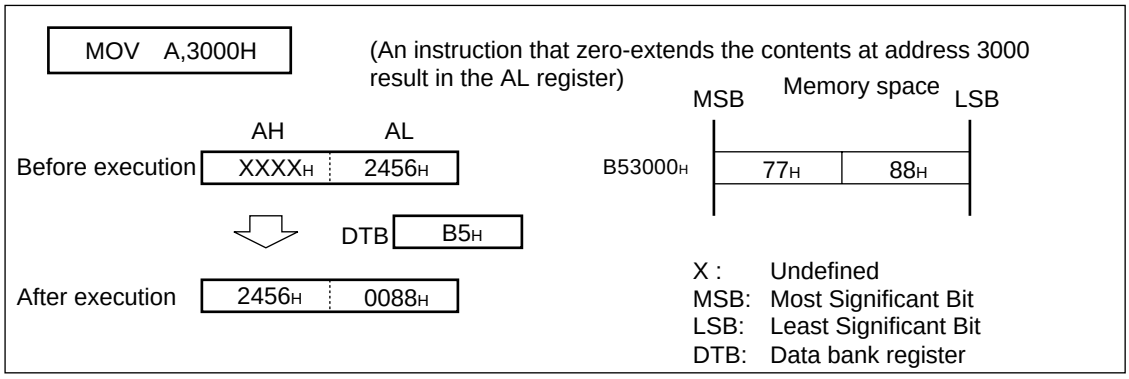


Figure 2.7-4 Example of AL-AH Transfer in the Accumulator (A) (8-bit Immediate Value, Sign Extension)

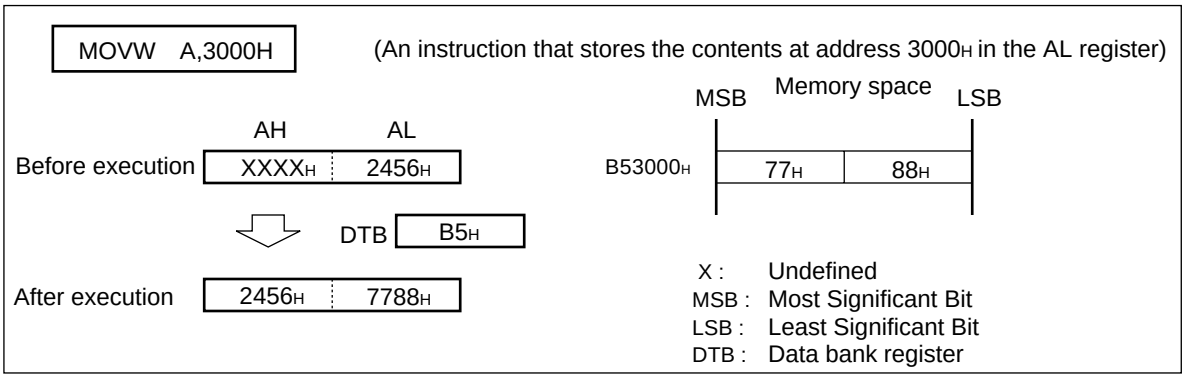


Figure 2.7-5 Example of 32-bit Data Transfer to the Accumulator (A) (Register Indirect)

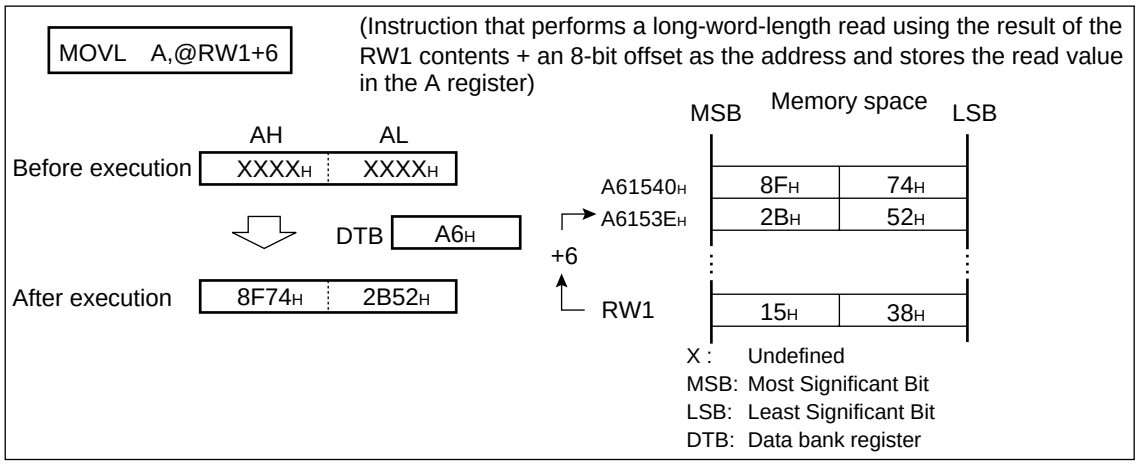
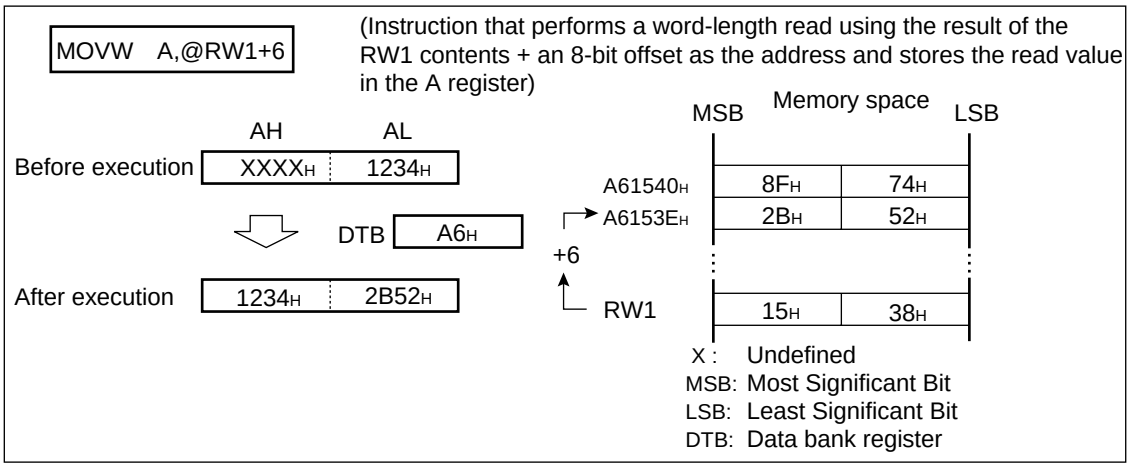


Figure 2.7-6 Example of AL-AH Transfer in the Accumulator (A) (16 bits, Register Indirect)



2.7.2 Stack Pointers (USP, SSP)

There are two types of stack pointers: a user stack pointer (USP) and a system stack pointer (SSP). Each stack pointer is a 24-bit register that indicates the memory address of the location of the destination for saved data or a return address when PUSH instructions, POP instructions, and subroutines are executed. The upper 8 bits of the stack address are specified by the user stack bank register (USB) or system stack bank register (SSB).
When the S flag of the condition code register (CCR) is 0, the USP and USB registers are valid. When the S flag is 1, the SSP and SSB registers are valid.

■ Stack Selection

The F^{2}MC-16LX uses two types of stack: a system stack and a user stack.

The stack address is determined, as shown in Table 2.7-2 "Stack Address Specification", by the S flag in the processor status (PS:CCR).

Table 2.7-2 Stack Address Specification

S flag	Stack address (24-bit)	
	Upper 8 bits	Lower 16 bits
0	User stack bank register (USB)	User stack pointer (USP)
1 (*)	System stack bank register (SSB)	System stack pointer (SSP)

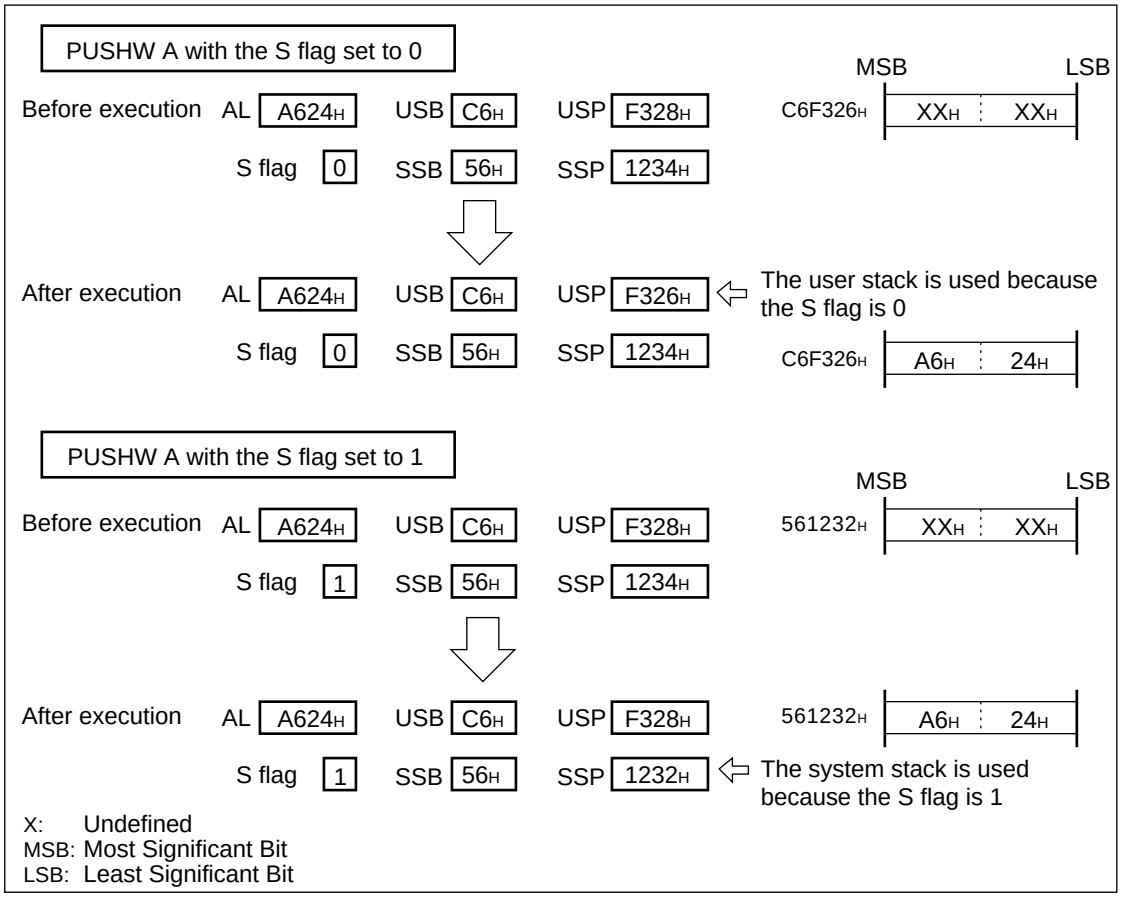
(*) : Initial value

Since a rest initializes the S flag to "1", the system stack is used by default. The system stack is used for interrupt routine stack operations and the user stack is used for all other types of stack operation. The system stack should be used unless the stack space is not divided.

Note:

When an interrupt is accepted, the S flag is set to "1" and thus the system stack is used.

Figure 2.7-7 Stack Operation Instruction and Stack Pointer



Note:

- To set a stack address in the stack pointer, use an even-numbered address. If an odd-numbered address is used, a word is accessed in two separate times.
- The USP and SSP registers have undefined initial values.

■ System Stack Pointer (SSP)

To use the system stack pointer (SSP), set the S flag in the condition code register (CCR) to "1". If the S flag is set to "1", the upper 8 bits of the address to be used for the stack operation are indicated by the system stack bank register (SSB).

For more information on the condition code register (CCR), see Section 2.7.4 "Condition Code Register (PS: CCR)". For more information on the system stack bank register (SSB), see Section 2.7.9 "Bank Registers (PCB, DTB, USB, SSB, ADB)".

■ User Stack Pointer (USP)

To use the user stack pointer (USP), set the S flag in the condition code register (CCR) to "0". If the S flag is set to "0", the upper 8 bits of the address to be used for the stack operation are indicated by the user stack bank register (USB).

For more information on the condition code register (CCR), see Section 2.7.4 "Condition Code Register (PS: CCR)". For more information on the system stack bank register (SSB), see Section 2.7.9 "Bank Registers (PCB, DTB, USB, SSB, ADB)".

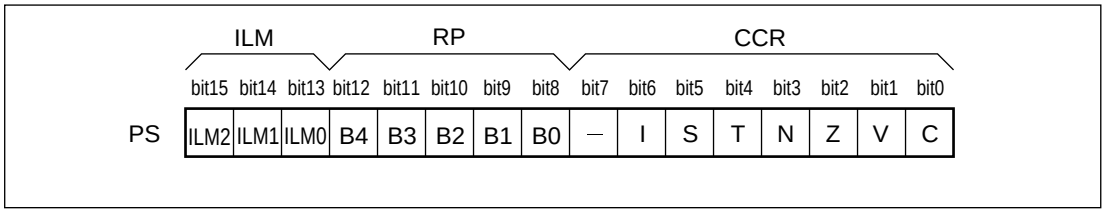
2.7.3 Processor Status (PS)

The processor status (PS) consists of CPU control bits and bits that indicate the CPU status. The Processor status (PS) consists of the following three registers:

- **Condition code register (CCR)**
- **Register bank pointer (RP)**
- **Interrupt level mask register (ILM)**

■ Processor Status (PS) Configuration

Figure 2.7-8 Processor Status (PS) Configuration



❍ Condition Code Register (CCR)

This register consists of flags that are set to "1" or reset to "0" in accordance with instruction execution results or interrupts.

For more information on the flags, see Section 2.7.4 "Condition Code Register (PS: CCR)".

❍ Register Bank Pointer (RP)

This pointer points to the first address of the memory block (register bank) used as the general-purpose register in the RAM area.

There are 32 banks for general-purpose registers. Set values "00_H to $1F_H$" in the RP to specify a bank.

For the setting method and more information on this pointer, see Section 2.7.5 "Register Bank Pointer (PS: RP)".

❍ Interrupt Level Mask Register (ILM)

This register indicates the level of an interrupt currently accepted by the CPU. The value is compared with that of the interrupt level setting bits (ICR: IL0 to IL2) in the interrupt control register (ICR00 to ICR15) set so that an interrupt request corresponds to each peripheral function (resource).

For the setting method and more information on this register, see Section 2.7.6 "Interrupt Level Mask Register (PS: ILM)".

2.7.4 Condition Code Register (PS: CCR)

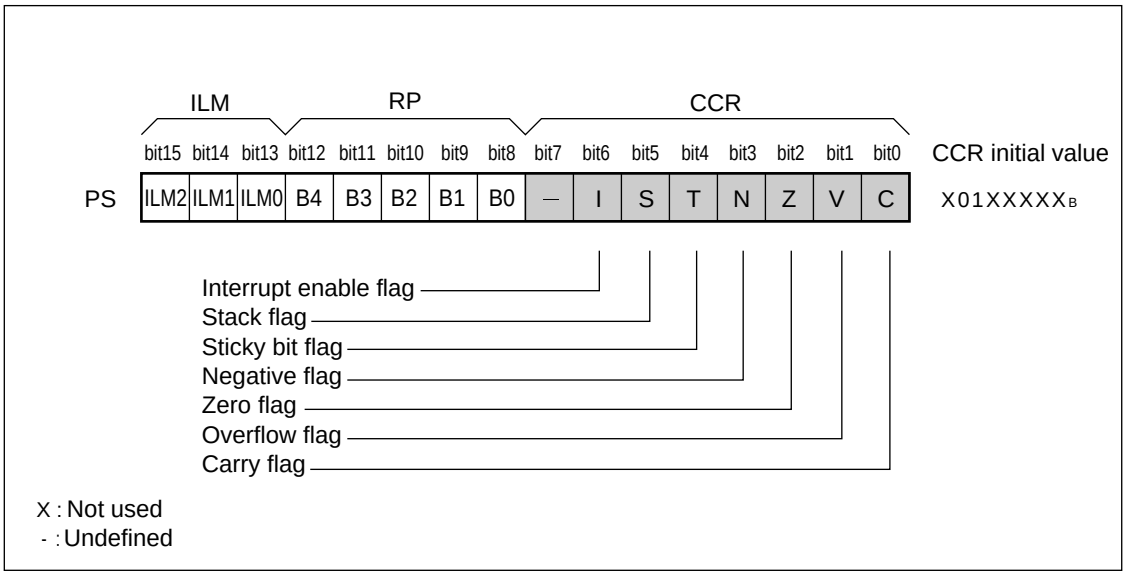
The condition code register (CCR) is an 8-bit register that consists of the following bits:

- **Bits that indicate the result of an arithmetic operation and the contents of transfer data**
- **Bits that control the acceptance of an interrupt request**

■ Condition Code Register (CCR) Configuration

Refer to the programming manual for details about the status of the condition code register (CCR) during instruction execution.

Figure 2.7-9 Condition Code Register (CCR) Configuration



❍ Interrupt enable flag (I)

Interrupts are enabled when the I flag is set to "1", or disabled when the I flag is reset to "0", in response to any interrupt request other than software interrupts. The I flag is reset to "0" by an external or software reset.

❍ Stack flag (S)

This flag indicates the pointer used for a stack operation. The user stack pointer (USP) is valid if the S flag is reset to "0". The system stack pointer (SSP) is valid if the S flag is set to "1". The S flag is set to "1" when an interrupt is accepted or when an external or software reset is asserted.

For more information on the stack pointers, see Section 2.7.2 "Stack Pointers (USP, SSP)"

❍ Sticky bit flag (T)

The T flag is set to "1" if the data shifted out of by the carry contains "1" during execution of a logical or arithmetic right shift instruction. Otherwise, the T flag is reset to "0". The T flag is also

reset to "0" if the shift amount is zero.

- **Negative flag (N)**

  The N flag is set to "1" if the most significant bit (MSB) of the general-purpose registers (RL0 to RL3) that store the operation result is "1". Otherwise, the N flag is reset to "0".

  For more information on general-purpose registers, see Section 2.8 "General-Purpose Registers".

- **Zero flag (Z)**

  The Z flag is set to "1" if the general-purpose registers (RL0 to RL3) that store the operation result are "0000_H". Otherwise, the Z flag is reset to "0".

  For more information on general-purpose registers, see Section 2.8 "General-Purpose Registers".

- **Overflow flag (V)**

  The V flag is set to "1" if an overflow occurs in a signed numeric value. Otherwise, the V flag is reset to "0".

- **Carry flag (C)**

  The C flag is set to "1" if a carry from the most significant bit or a borrow to the least significant bit occurs during an arithmetic operation. Otherwise, the C flag is reset to "0".

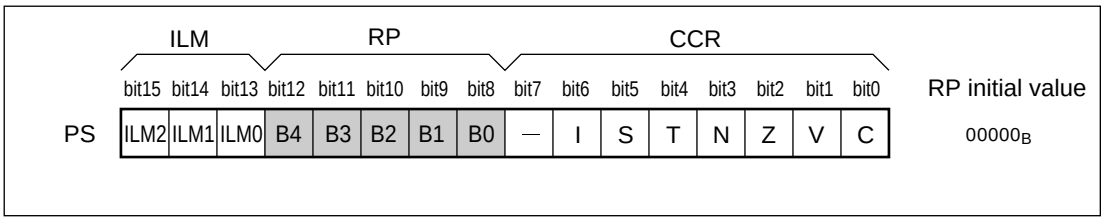
2.7.5 Register Bank Pointer (PS: RP)

The register bank pointer (RP) is a five-bit register that points to the first address of the general-purpose register bank currently used.

■ Register Bank Pointer (RP)

Figure 2.7-10 "Configuration of the Register Bank Pointer (RP)" shows the configuration of the register bank pointer (RP) register.

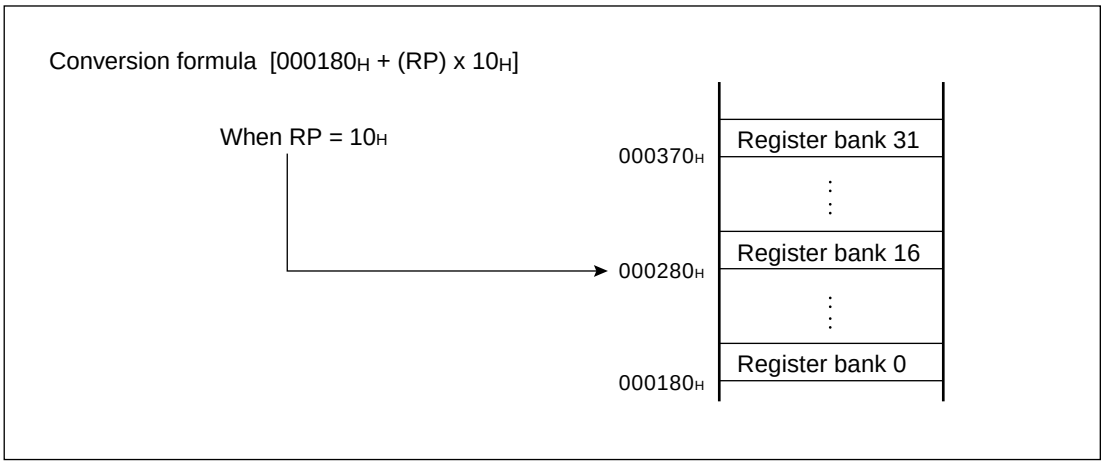
Figure 2.7-10 Configuration of the Register Bank Pointer (RP)



■ General-Purpose Register Area and Register Bank Pointer

The register bank pointer (RP) points to the relationship between the general-purpose register of the F^{2}MC-16LX and the address in internal RAM. The relationship between the contents of the RP register and the address follows the conversion rules shown in Figure 2.7-11 "Conversion Rules for Physical Address of General-Purpose Register Area".

Figure 2.7-11 Conversion Rules for Physical Address of General-Purpose Register Area



- Although an assembler instruction can use an 8-bit immediate value transfer instruction for transfer to the register bank pointer (RP). However, only the lower 5 bits of the data are valid.

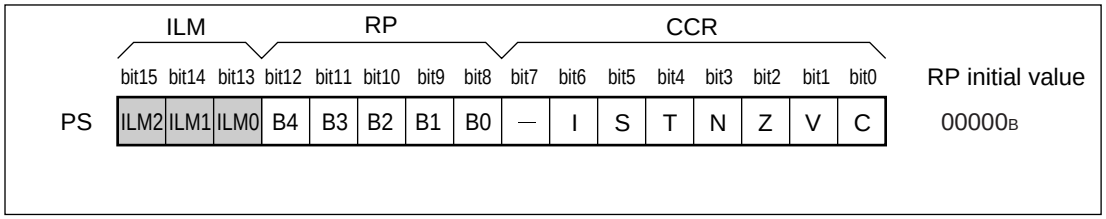
2.7.6 Interrupt Level Mask Register (PS: ILM)

The interrupt level mask register (ILM) is a 3-bit register that indicates the level of the interrupt currently accepted by the CPU.

■ Interrupt Level Mask Register (ILM)

See CHAPTER 6 "INTERRUPTS" for details about interrupts.

Figure 2.7-12 Configuration of the Interrupt Level Mask Register (ILM)



The interrupt level mask register (ILM) indicates the level of the interrupt currently accepted by the CPU.

A level of the interrupt currently accepted by the CPU can be set in the ILM register. The CPU does not accept any interrupt with an interrupt level lower than that set in the ILM register.

- A reset sets the highest interrupt level in the ILM register, causing no interrupt to be accepted.
- Although an assembler instruction can use an 8-bit immediate value transfer instruction for transfer to the interrupt level mask register (ILM). However, only the lower 3 bits of the data are valid.

Table 2.7-3 Interrupt Level Mask Register (ILM) and Interrupt Level Priority

ILM2	ILM1	ILM0	Interrupt level	Interrupt level priority
0	0	0	0	Highest (interrupts disabled)
0	0	1	1	
0	1	0	2	
0	1	1	3	
1	0	0	4	
1	0	1	5	
1	1	0	6	
1	1	1	7	Lowest

2.7.7 Program Counter (PC)

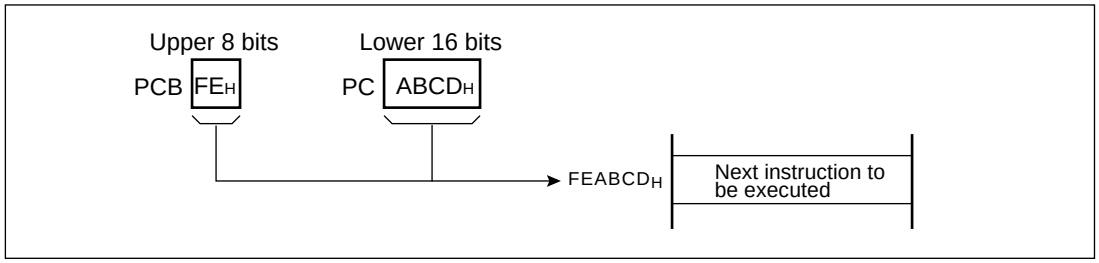
The program counter (PC) is a 16-bit counter that indicates the lower 16 bits of the address of the next instruction to be executed by the CPU.

■ Program Counter (PC)

The program bank register (PCB) specifies the upper 8 bits, and the program counter (PC) specifies the lower 16 bits, of the address of the next instruction to be executed by the CPU. The address of the next instruction to be executed is as shown in Figure 2.7-13 "Program Counter (PC)". The contents of the PC are updated by conditional branch instructions, subroutine call instructions, interrupts, and resets. The PC can also be used as a base pointer for reading operands.

For more information on the PCB, see Section 2.7.9 "Bank Registers (PCB, DTB, USB, SSB, ADB)".

Figure 2.7-13 Program Counter (PC)



Note:

The PC and PCB cannot be rewritten directly by a program (instructions such as MOV PC and #0FF$_H$).

2.7.8 Direct Page Register (DPR)

The direct page register (DPR) is an 8-bit register that specifies bits 8 to 15 (addr8 to addr15) of the operand address when a short direct addressing instruction is executed. A reset initializes the DPR to "01_H".

■ Direct Page Register (DPR)

For information on the abbreviated direct address specification method, see Appendix B.3 "Direct Addressing".

Figure 2.7-14 Physical Address Generation by the Direct Page Register (DPR)

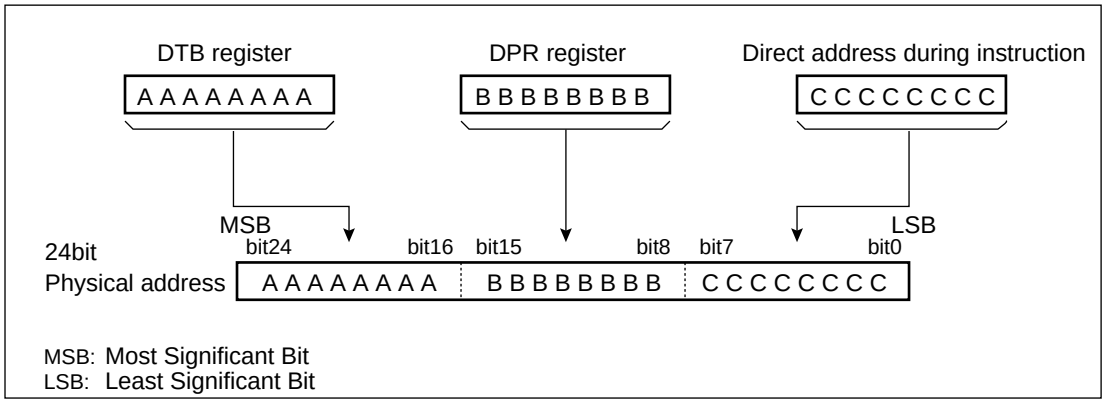
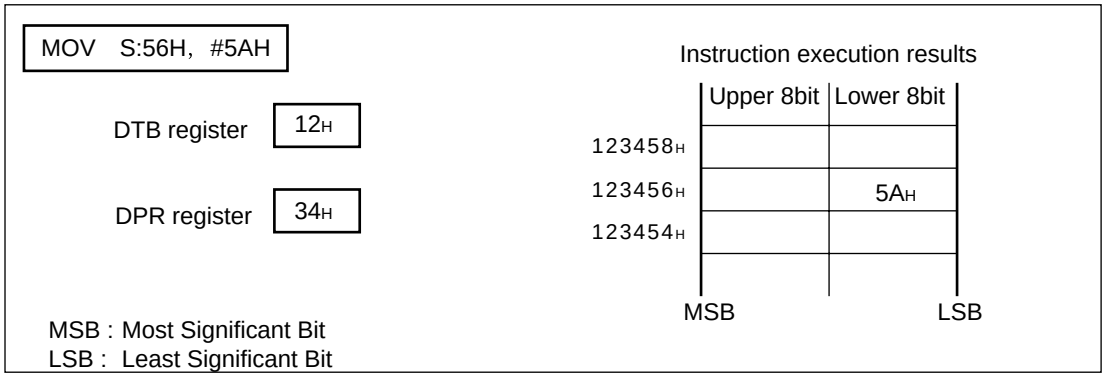


Figure 2.7-15 Example of Direct Page Register (DPR) Setting and Data Access



2.7.9 Bank Registers (PCB, DTB, USB, SSB, ADB)

Bank registers specify the highest 8-bit address by bank addressing. The five bank registers are as follows:

- **Program bank register (PCB)**
- **Data bank register (DTB)**
- **User stack bank register (DTB)**
- **System stack bank register (SSB)**
- **Additional bank register (ADB)**

The PCB, DTB, USB, SSB, and ADB registers indicate the individual memory banks where the program space, data space, user stack space, system stack space, and additional space are located.

■ Bank Registers (PCB, DTB, USB, SSB, ADB)

❍ Program bank register (PCB)

The PCB is a bank register that specifies the program (PC) space.

❍ Data bank register (DTB)

The DTB is a bank register that specifies the data (DT) space.

❍ User stack bank register (USB), system stack bank register (SSB)

The USB and SSB are bank registers that specify the stack (SP) space.

❍ Additional bank register (ADB)

The ADB is a bank register that specifies the additional (AD) space.

❍ Bank settings and data access

All bank registers are 8 bits in length. A reset initializes the PCB to "FF_H" and the DTB, USB, SSB, and ADB to "00_H". The PCB can be read but cannot be written to. Bank registers other than the PCB can be read and written to.

Note:

The MB90560/565 series supports up to the memory space contained in the device.

See Section 2.4.2 "Address Specification by Bank Addressing" for the operation of each register.

2.8 General-Purpose Registers

The general-purpose registers are a memory block allocated in RAM at 000180_H to $00037F_H$ as register banks, each of which consists of eight 16-bit segments.
The general-purpose registers can be used as general-purpose 8-bit registers (byte registers R0 to R7), 16-bit registers (word registers RW0 to RW7), or 32-bit registers (long-word registers RL0 to RL7).
General-purpose registers can access RAM with a short instruction at high speed. Since general-purpose registers are blocked into register banks, protection of register contents and division into function units can readily be performed. When a general-purpose register is used as a long-word register, it can be used as a linear pointer that directly accesses the entire space.

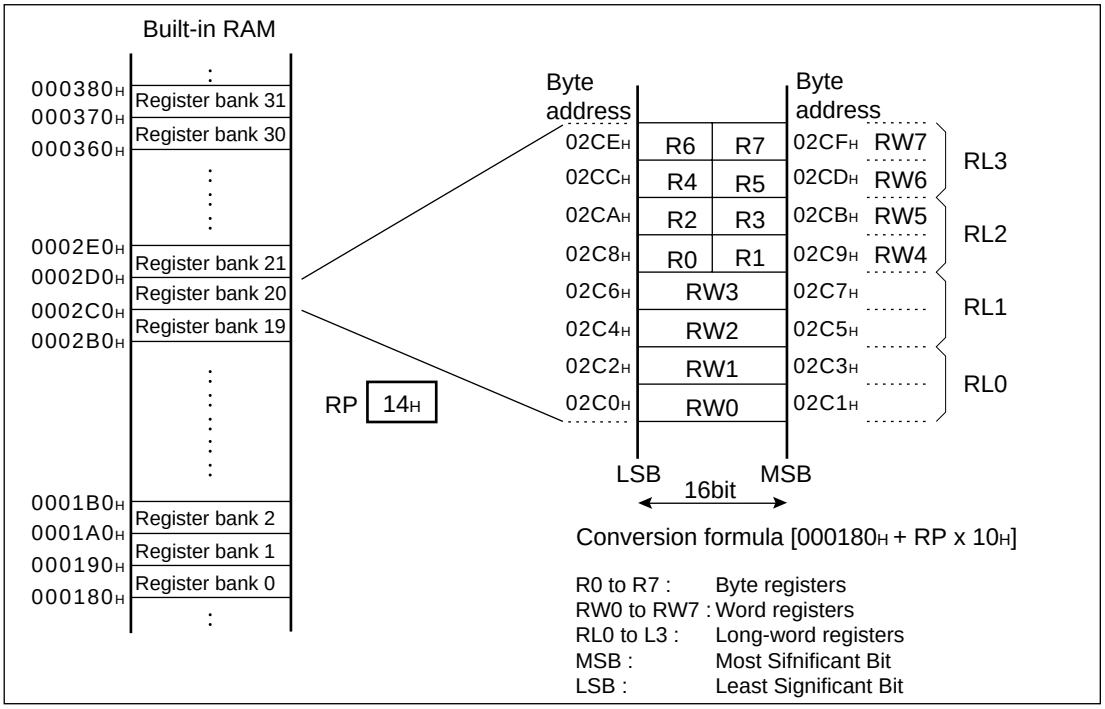
■ Configuration of a General-Purpose Register

General-purpose registers exist in RAM at "000180_H" to "$00037F_H$" and are configured as 32 banks. The register bank pointer (RP) specifies the bank. The RP determines the first address of each bank as shown in the following equation. 16 bits multiplied by 8 are defined as one register bank.

First address of general-purpose register = $000180_H + RP \times 10_H$

For more information on the PR, see Section 2.7.5 "Register Bank Pointer (PS: RP)".

Figure 2.8-1 Location and Configuration of the General-Purpose Register Banks in the Memory Space



Note:

The register bank pointer (RP) is initialized to 00_H after a reset.

■ Register Bank

A reset does not initialize the contents of the register bank as with RAM but the status before reset is retained. A power-on reset, however, makes the contents undefined.

Table 2.8-1 Typical Functions of General-Purpose Registers

Register name	Function
R0 to R7	Used as an operand in various instructions **Note:** R0 is also used as a barrel shift counter and an instruction normalization counter
RW0 to RW7	Used as a pointer Used as an operand in various instructions **Note:** RW0 is used also as a string instruction counter
RL0 to RL3	Used as a long pointer Used as an operand in various instructions

2.9 Prefix Codes

Prefix codes are placed before an instruction to partially change the operation of the instruction. The three types of prefix codes are as follows:

- **Bank select prefix (PCB, DTB, ADB, SPB)**
- **Common register bank prefix (CMR)**
- **Flag change suppression prefix (NCC)**

■ Prefix Codes

❍ Bank select prefix (PCB, DTB, ADB, SPB)

A bank select prefix is placed before an instruction to select the memory space to be accessed by the instruction regardless of the addressing method.

For more information, see Section 2.9.1 "Bank Select Prefix (PCB, DTB, ADB, SPB)".

❍ Common register bank prefix (CMR)

The common register bank prefix is placed before an instruction that accesses a register bank to change the register accessed by the instruction to the common bank (register bank selected when RP = 0) at 000180_H to $00018F_H$ regardless of the current register bank pointer (RP) value.

For more information, see Section 2.9.2 "Common Register Bank Prefix (CMR)".

❍ Flag change suppression prefix (NCC)

The flag change suppression prefix code is placed before an instruction to suppress a flag change accompanying the execution of the instruction.

For more information, see Section 2.9.3 "Flag Change Suppression Prefix (NCC)".

2.9.1 Bank Select Prefix (PCB, DTB, ADB, SPB)

Memory space used for data access is determined for each addressing method. However, placing a bank select prefix before an instruction selects the memory space to be accessed by the instruction regardless of the addressing method.

■ **Bank Select Prefixes (PCB, DTB, ADB, SPB)**

Table 2.9-1 Bank Select Prefix Codes

Bank select prefix	Selected space
PCB	Program space
DTB	Data space
ADB	Additional space
SPB	When the value of the S flag in the condition code register (CCR) is 0 and the user stack space is 1, the system stack space is used.

If a bank select prefix is used, some instructions perform an unexpected operation.

Table 2.9-2 Instructions Not Affected by Bank Select Prefix Codes

Instruction type	Instruction				Effect of bank select prefix
String instruction	MOVS SCEQ FILS		MOVSW SCWEQ FILSW		The bank register specified by the operand is used irrespective of whether a prefix is used.
Stack operation instruction	PUSHW		POPW		When the S flag is 0, the user stack bank (USB) is used whether or not there is a prefix. When the S flag is 1, the system stack bank (SSB) is used regardless of whether a prefix is used.
I/O access instruction	MOV MOVW MOV MOV MOVB SETB BBC WBTC	A,io A,io io, A io,#imm8 A,io:bp io:bp io:bp, rel io,bp	MOVX MOVW MOVW MOVB CLRB BBS WBTS	A,io io, A io,#imm16 io:bp,A io:bp io:bp, rel io:bp	The I/O space (000000_H to $0000FF_H$) is accessed whether or not there is a prefix.
Interrupt return instruction	RETI				The system stack bank (SSB) is used whether or not a prefix is used.

Table 2.9-3 Instructions Whose Use Requires Caution When Bank Select Prefix

Instruction type	Instruction		Explanation
Flag change instruction	AND OR	CCR, #imm8 CCR, #imm8	The effect of the prefix extends to the next instruction.
ILM setting instruction	MOV	ILM, #imm8	The effect of the prefix extends to the next instruction.
PS return instruction	POPW	PS	Do not place a bank select prefix before the PS return instruction.

2.9.2 Common Register Bank Prefix (CMR)

Placing the common register bank prefix (CMR) before an instruction that accesses a register bank changes the register accessed by it to the common bank at "000180_H" to "$00018F_H$" (register bank selected when RP = "00_H") regardless of the current register bank pointer (RP) value.

■ Common Register Bank Prefix (CMR)

To facilitate data exchange between multiple tasks, the F^2MC-16LX provides a common bank that can be commonly used by these tasks. The common bank is located at addresses "000180_H" to "$00018F_H$".

However, be careful when you use this prefix with the instructions listed in Table 2.9-4 "Instructions Whose Use Requires Caution When the Common Register Bank Prefix (CMR) Is Used ".

Table 2.9-4 Instructions Whose Use Requires Caution When the Common Register Bank Prefix (CMR) Is Used

Instruction type	Instruction			Explanation
String instruction	MOVS SCEQ FILS		MOVSW SCWEQ FILSW	Do not place the CMR prefix before the string instruction.
Flag change instruction	AND	CCR,#imm8	OR CCR,#imm8	The effect of the prefix extends to the next instruction.
PS return instruction	POPW	PS		The effect of the prefix extends to the next instruction.
ILM setting instruction	MOV	ILM,#imm8		The effect of the prefix extends to the next instruction.

2.9.3 Flag Change Suppression Prefix (NCC)

The flag change suppression prefix (NCC) code is set before an instruction to suppress a flag change accompanying the execution of the instruction.

■ Flag Change Suppression Prefix (NCC)

Use the flag change suppression prefix (NCC) to suppress unnecessary flag changes. Changes in the T, N, Z, V, and C flags can be suppressed.

Be careful when you use this prefix with the instructions listed in Table 2.9-5 "Instructions Whose Use Requires Caution When the Flag Change Suppression Prefix (NCC) Is Used ".

For more information on the T, N, Z, V, and C flags, see Section 2.7.4 "Condition Code Register (PS: CCR)".

Table 2.9-5 Instructions Whose Use Requires Caution When the Flag Change Suppression Prefix (NCC) Is Used

Instruction type	Instruction	Explanation
String instruction	MOVS MOVSW SCEQ SCWEQ FILS FILSW	Do not place the NCC prefix before the string instruction.
Flag change instruction	AND CCR,#imm8 OR CCR,#imm8	The condition code register (CCR) changes as defined in the instruction specification whether or not a prefix is used. The effect of prefix extends to the next instruction.
PS return instruction	POPW PS	The condition code register (CCR) changes as defined in the instruction specification whether or not a prefix is used. The effect of prefix extends to the next instruction.
ILM setting instruction	MOV ILM,#imm8	The effect of prefix extends to the next instruction.
Interrupt instruction Interrupt return instruction	INT #vct8 INT9 INT adder16 INTP addr24 RETI	The condition code register (CCR) changes as defined in the instruction specification whether or not a prefix is used.
Context switch instruction	JCTX @A	The condition code register (CCR) changes as defined in the instruction specification whether or not a prefix is used.

2.9.4 Restrictions on Prefix Codes

The following restrictions are imposed on the use of prefix codes:

- **Interrupt requests are not accepted during the execution of prefix codes and interrupt suppression instructions.**
- **If a prefix code is placed before an interrupt instruction, the effect of the prefix code is delayed.**
- **If consecutively placed prefix codes conflict, the last prefix code is valid.**

■ Prefix Codes and Interrupt Suppression Instructions

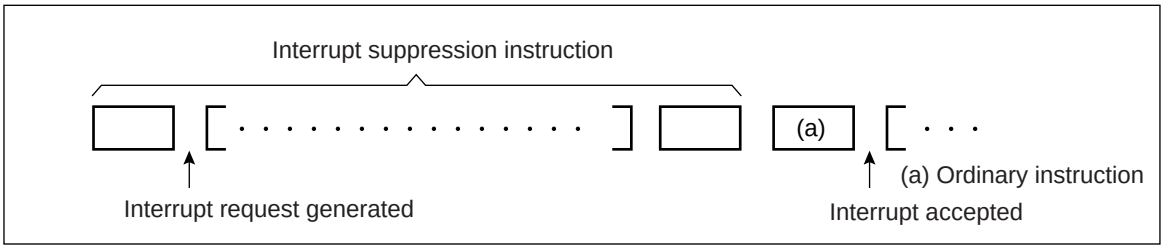
Table 2.9-6 Prefix Codes and Interrupt Suppression Instructions

	Prefix codes	Interrupt suppression instructions (instructions that delay the effect of prefix codes)	
Instructions that do not accept interrupt requests	PCB DTB ADB SPB CMR NCC	MOV OR AND POPW	ILM, #imm8 CCR, #imm8 CCR, #imm8 PS

❍ Interrupt Suppression

As shown in Figure 2.9-1 "Interrupt Suppression", an interrupt request generated during the execution of prefix codes and interrupt instructions is not accepted. The interrupt is not processed until the first instruction that is not governed by a prefix code or that is not an interrupt suppression instruction is executed.

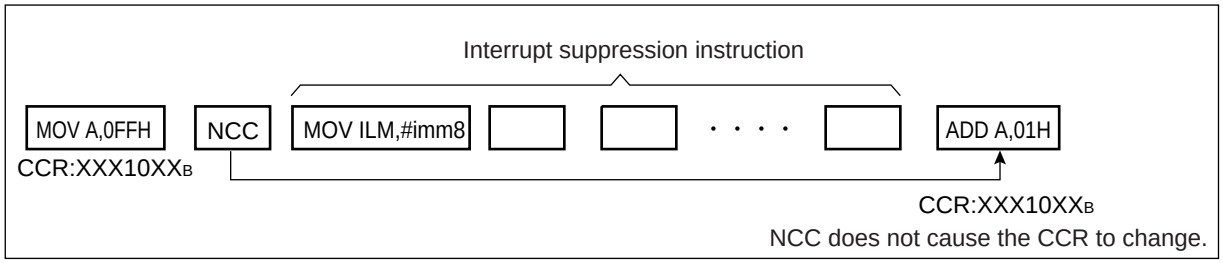
Figure 2.9-1 Interrupt Suppression



❍ **Delay of the effect of prefix codes**

If a prefix code is placed before an interrupt suppression instruction, the prefix code takes effect with the first instruction executed after the interrupt suppression instruction.

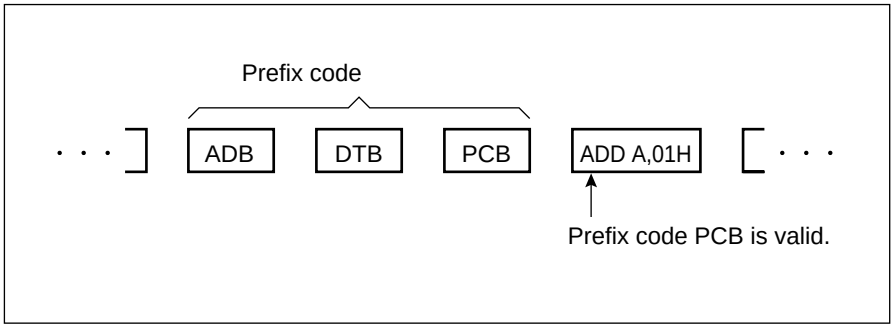
Figure 2.9-2 Interrupt Suppression Instructions and Prefix Codes



■ Consecutive Prefix Codes

When consecutive conflicting prefix codes (PCB, ADB, DTB, and SPB) are specified, the last prefix code is valid.

Figure 2.9-3 Consecutive Prefix Codes



2.9.5 Notes on Using the "DIV A, Ri" or "DIVW A, RWi" Instruction

To use the "DIV A, Ri" or "DIVW A, RWi" instruction, set the bank register to "00_H".

■ Notes on Using the "DIV A, Ri" or "DIVW A, RWi" Instruction

Table 2.9-7 Notes on Using the "DIV A, Ri" or "DIVW A, RWi" Instruction (i = 0 to 7)

Instruction	Bank register name to be affected by executing the instruction described on the left	Address in which the remainder is stored
DIV A,R0	DTB	(DTB:8 upper bits)+(0180_H+RPx10_H+8_H:16 lower bits)
DIV A,R1		(DTB:8 upper bits)+(0180_H+RPx10_H+9_H:16 lower bits)
DIV A,R4		(DTB:8 upper bits)+(0180_H+RPx10_H+C_H:16 lower bits)
DIV A,R5		(DTB:8 upper bits)+(0180_H+RPx10_H+D_H:16 lower bits)
DIVW A,RW0		(DTB:8 upper bits)+(0180_H+RPx10_H+0_H:16 lower bits)
DIVW A,RW1		(DTB:8 upper bits)+(0180_H+RPx10_H+2_H:16 lower bits)
DIVW A,RW4		(DTB:8 upper bits)+(0180_H+RPx10_H+8_H:16 lower bits)
DIVW A,RW5		(DTB:8 upper bits)+(0180_H+RPx10_H+A_H:16 lower bits)
DIV A,R2	ADB	(ADB:8 upper bits)+(0180_H+RPx10_H+A_H:16 lower bits)
DIV A,R6		(ADB:8 upper bits)+(0180_H+RPx10_H+E_H:16 lower bits)
DIVW A,RW2		(ADB:8 upper bits)+(0180_H+RPx10_H+4_H:16 lower bits)
DIVW A,RW6		(ADB:8 upper bits)+(0180_H+RPx10_H+E_H:16 lower bits)
DIV A,R3	USB SSB (*1)	(USB (*2) 8 upper bits)+(0180_H+RPx10_H+B_H:16 lower bits)
DIV A,R7		(USB (*2) 8 upper bits)+(0180_H+RPx10_H+F_HF6 lower bits)
DIVW A,RW3		(USB (*2) 8 upper bits)+(0180_H+RPx10_H+6_H:16 lower bits)
DIVW A,RW7		(USB (*2) 8 upper bits)+(0180_H+RPx10_H+E_H:16 lower bits)

*1 : Depending on the S bit in the CCR register
*2 : When the S bit in the CCR register is 0.

If the bank register (DTB, ADB, USB, SSB) value is "00_H", the remainder obtained through division is stored in the instruction operand register. Otherwise, the upper-8-bit address is specified in the bank register corresponding to the instruction operand register, and the lower-16-bit address is the same as the address in the instruction operand register. The remainder is stored in the bank register specified by the upper 8 bits.

Example:

If "DIV A, R0" is executed when DTB = "053_H" and RP = "03_H", the R0 address is "0180_H" + RP ("03_H") x "10_H" + "08_H" (R0 equivalent address) = "$0001B8_H$".

Here, the bank register specified by "DIV A, R0" is a data bank register (DTB), the remainder is stored at the address to which the bank address "053_H" is added, i.e., "$05301B8_H$". (For more information on the Ri and RWi registers, see Section 2.8 "General-Purpose Registers".)

■ Careful Development Based on the Above Notes

To allow you to develop a program while working around the precautions on using the "DIV A, Ri" or "DIVW A, RWi" instruction, we provide a special compiler and a special assembler. The compiler has been modified so that it does not generate the instructions shown in Table 2.9-7 "Notes on Using the "DIV A, Ri" or "DIVW A, RWi" Instruction (i = 0 to 7)". The assembler has an additional function that replaces the above instructions with equivalent instruction sequences. Use the following compiler and assembler:

- Compiler:
  - cc907 V02L06 or later versions, and fcc907s V30L02 or later versions.
- Assembler:
  - asm907a V03L04 or later versions, and fasm907s V30L04 (Rev. 300004) or later versions.

CHAPTER 3 RESETS

This chapter describes resets for the MB90560/565 series.

3.1 Resets

If a reset cause is generated, the CPU stops the current execution process and waits for the reset to be cleared. When the reset is cleared, the CPU begins processing at the address indicated by the reset vector.
There are four causes of a reset:

- **Power-on reset (at power-on)**
- **Watchdog timer overflow (during the use of a watchdog timer)**
- **External reset input via the $\overline{\text{RST}}$ pin**
- **Setting "0" in the internal reset signal generation bit (RST) of the low power consumption mode control register (software reset)**

■ Reset Causes

Table 3.1-1 Reset Causes

Type of reset	Cause	Machine clock	Watchdog timer	Oscillation stabilization wait
External pin	"L" level input to $\overline{\text{RST}}$ pin	Main clock frequency (MCLK)	Stop	No
Software	"0" written to the internal reset signal generation bit (RST) of the low power consumption mode control register (LPMCR)	Main clock frequency (MCLK)	Stop	No
Watchdog timer	Watchdog timer overflow if the watchdog function is enabled	Main clock frequency (MCLK)	Stop	No
Power-on	Power-on	Main clock frequency (MCLK)	Stop	Yes

MCLK: Main clock frequency (oscillation clock frequency divided by 2: 2/HCLK)

❍ External reset

An external reset is generated if the external reset terminal ($\overline{RST}$ terminal) is set to the "L" level. The "L" level must be input at least for 16 machine cycles (16/ϕ). While the machine clock is used, no oscillation stabilization wait interval is placed even if a reset occurs due to the "L" level input to the external reset pin.

Reference:

If this pin is asserted while an instruction is executed via the external reset pin (while a transfer instruction such as MOV is executed), the reset input becomes valid after the instruction being executed is completed.

For a string-processing instruction (such as MOVS), however, the reset input may become valid before the transfer due to the specified counter value is completed.

If the external reset pin is asserted, the port pin enters the reset status regardless of the instruction execution cycle (asynchronous operation if asserted).

❍ Software reset

A software reset is a reset for three machine cycles (3/ϕ) generated by writing "0" to the internal reset signal generation bit (RST) of the low power consumption mode control register (LPMCR). The oscillation stabilization wait interval is not required for software resets.

❍ Watchdog timer reset

A watchdog timer reset is generated unless "0" is written to the watchdog timer control bit (WTE) of the watchdog timer control register (WDTC) within the time specified in the interval time setting bits (WT1, WT0) of the WDTC after the watchdog timer is activated.

❍ Power-on reset

A power-on reset is generated when the power is turned on. The oscillation stabilization wait interval for the MB90V560 and MB90F562/B is fixed at 2^{18}/HCLK (about 65.54 ms if the source oscillation is 4 MHz). The oscillation stabilization wait interval for the MB90561/A, MB90562/A, MB90F568, MB90567, and MB90568 is fixed at 2^{17}/HCLK (about 32.77 ms if the source oscillation is 4 MHz). A reset occurs after the oscillation stabilization wait interval has elapsed.

Information: Definition of clocks

HCLK: Oscillation clock frequency (Clock supplied from the oscillation pin)

MCLK: Main clock frequency (Clock obtained by dividing the source oscillation by two)

ϕ : Machine clock (CPU operating clock)

1/ϕ: Machine cycle (CPU operating clock cycle)

See Section 4.1 "Clocks" for details about clocks.

3.2 Reset Causes and Oscillation Stabilization Wait Intervals

The F^{2}MC-16LX has four reset causes. The oscillation stabilization wait interval for a reset depends on the reset cause.

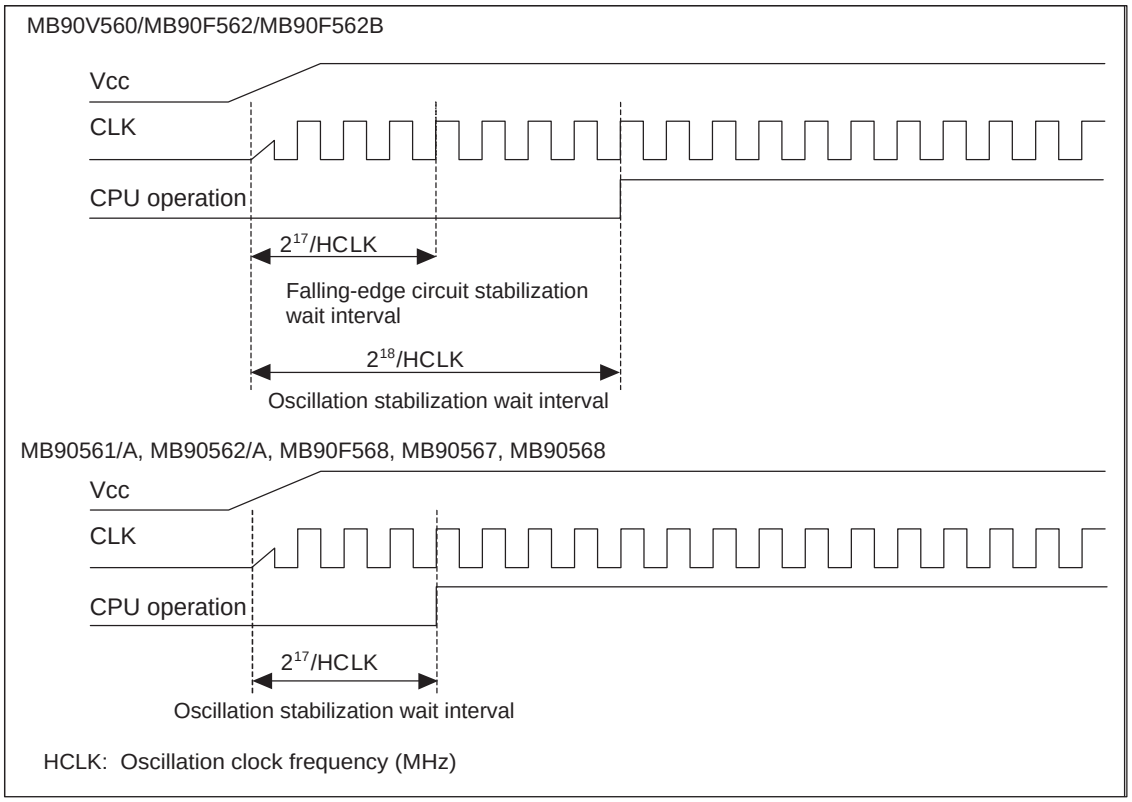
■ Reset Causes and Oscillation Stabilization Wait Intervals

Table 3.2-1 Reset Causes and Oscillation Stabilization Wait Intervals

Reset cause	Oscillation stabilization wait interval The corresponding time interval for an oscillation clock frequency of 4 MHz is given in parentheses.
Power-on reset	MB90V560, MB90F562/B: 2^{18}/HCLK (approximately 65.54 ms) MB90561/A, MB90562/A, MB90F568, MB90567, MB90568: 2^{17}/HCLK (approximately 32.77 ms)
Watchdog timer	None. (The WS1 and WS0 bits are initialized to "11_B".)
External reset	None. (The WS1 and WS0 bits are initialized to "11_B".)
Software reset	None. (The WS1 and WS0 bits are initialized to "11_B".)

HCLK: Oscillation clock frequency (MHz)

Figure 3.2-1 Oscillation Stabilization Wait Interval for the MB90560 and 565 Series during a Power-on Reset



Note:

Oscillation clock oscillators generally require an oscillation stabilization wait interval from the start of oscillation until they stabilize at their natural frequency. Be sure to set a proper oscillation stabilization wait interval for the oscillator to be used.

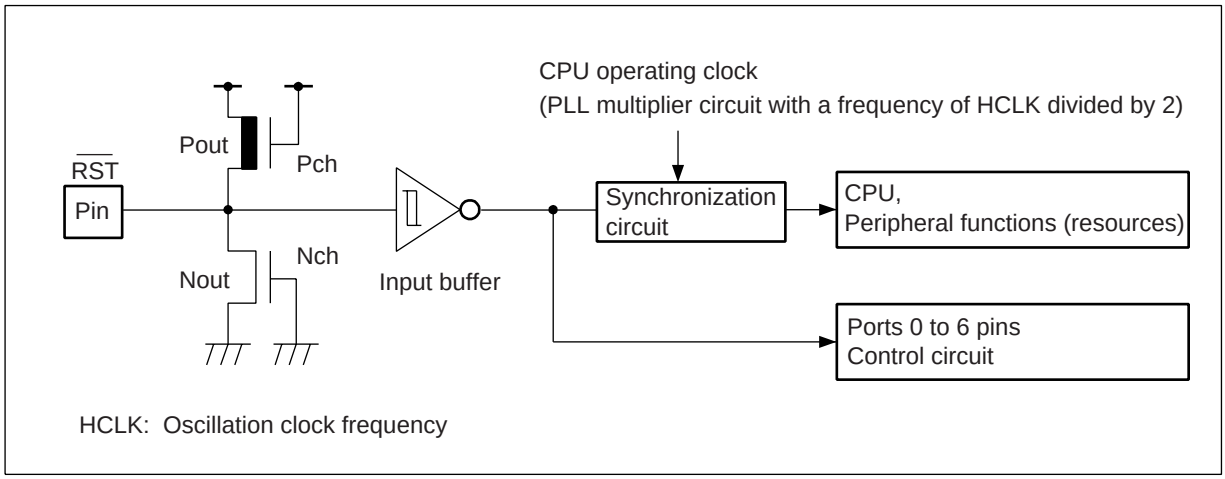
See Section 4.5 "Oscillation Stabilization Wait Interval" for details about Oscillation Stabilization Wait Interval.

3.3 External Reset Pin

A reset occurs if an "L" level signal is input to the external reset pin ($\overline{\text{RST}}$ pin).

■ Block Diagrams of the External Reset Pin

Figure 3.3-1 Block Diagram of Reset Occurrence



Note:

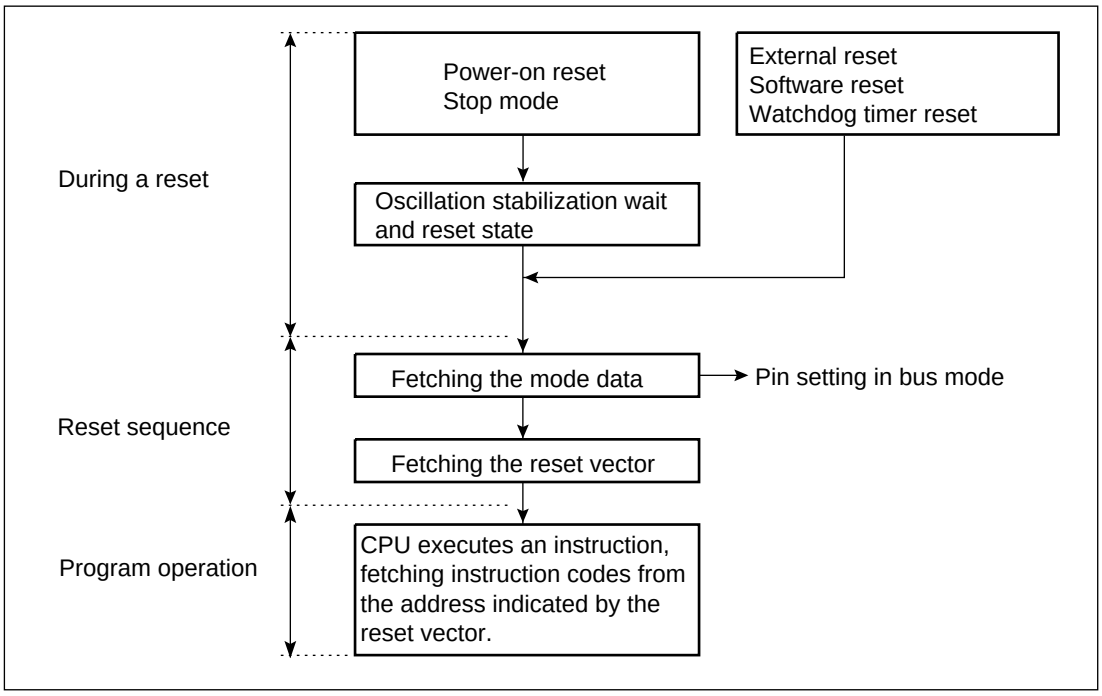
The machine clock is required to initialize the internal circuit. When a reset signal is input, the clock must be supplied from the oscillation pin.

3.4 Reset Operation

When a reset is cleared, the mode data and the reset vector stored in the internal or external memory are fetched. The mode data register determines the CPU operating mode. The reset vector determines the execution start address used after a reset sequence ends.

■ Overview of Reset Operation

Figure 3.4-1 Reset Operation Flow



■ Mode Pins

Setting the mode pins (MD0 to MD2) specifies how to fetch the mode data and the reset vector. Fetching the mode data and the reset vector is performed in the reset sequence. See Section 7.2 "Mode Pins (MD2 to MD0)" for details about mode pins.

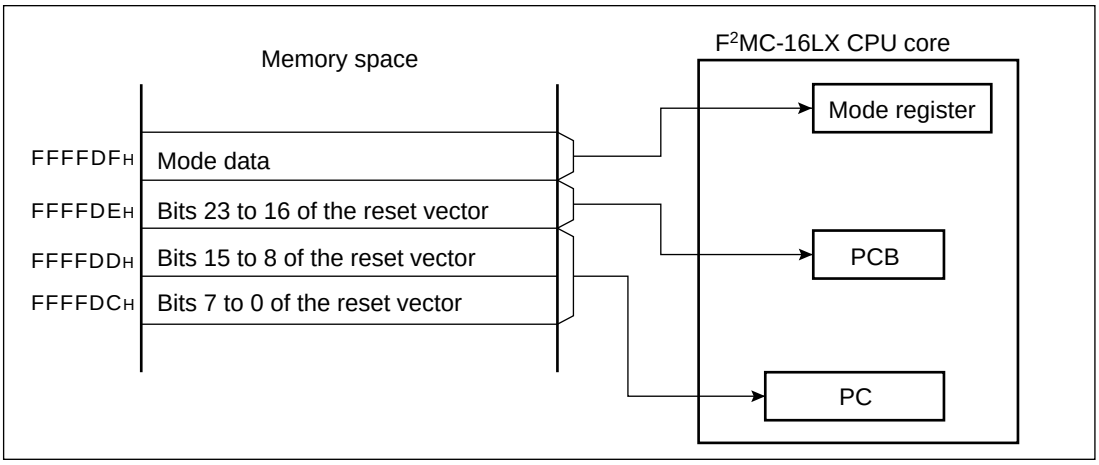
■ Mode Data Fetch

When a reset is cleared, the CPU transfers mode data to the mode data register. After the mode data is transferred, the reset vector is transferred to the program counter (PC) and the program counter bank register (PCB).

The mode data register can determine the bus mode and the bus width. The reset vector can determine the program start address.

For more information, see CHAPTER 7 "SETTING A MODE".

Figure 3.4-2 Transfer of Reset Vector and Mode Data



Reference:

Set the mode pins to specify whether the mode data and the reset vector should be read from internal ROM (or flash memory) or from external memory. If the mode pins are set to external memory, the mode data and the reset vector are read from the external memory. If the mode pins are set to internal ROM (or flash memory), the mode data and the reset vector are read from the internal ROM (or flash memory). If the single-chip mode is used, set the mode pins to internal ROM (or flash memory).

For more information, see Section 7.2 "Mode Pins (MD2 to MD0)".

❍ Mode data register (address: $FFFFDF_H$)

The mode data register setting can be changed while a reset sequence is executed. The mode data register setting is valid after a reset vector is fetched. No new contents can be written to the mode data register even if an instruction is used to specify mode data at "$FFFFDF_H$".

For more information, see Section 7.3 "Mode Data Register".

❍ Reset vector (address: "$FFFFDC_H$" to "$FFFFDE_H$")

The reset vector determines the program start address used after a reset is cleared. A program is executed from the address specified in the reset vector.

3.5 Reset Cause Bits

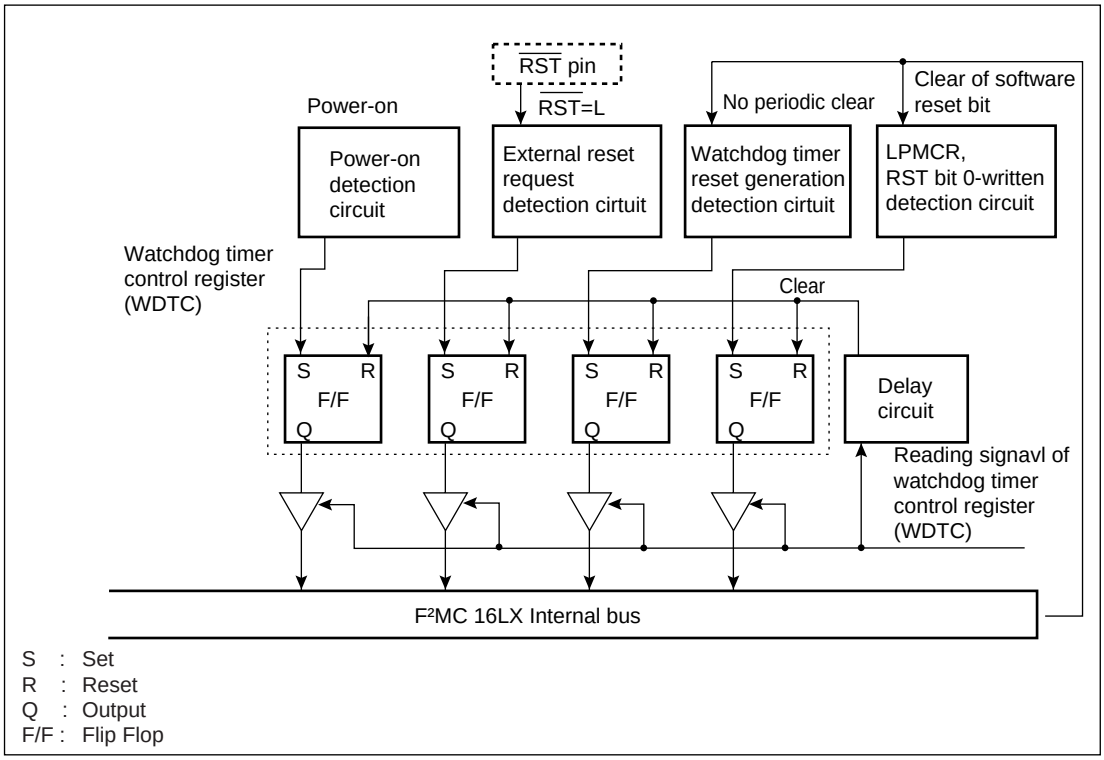
Read the watchdog timer control register (WDTC) to identify a reset cause.

■ Reset Cause Bits

Read the reset cause flag bits PONR, WRST, ERST, and SRST of the watchdog timer control register (WDTC) to identify a reset cause. If a reset cause needs to be identified after a reset is cleared, read the reset cause flag bits PONR, WRST, ERST, and SRST of the watchdog timer control register (WDTC).

The PONR, WRST, ERST, and SRST bits are cleared to "0" if the watchdog timer control register (WDTC) is read.

Figure 3.5-1 Block Diagram of Reset Cause Bits



■ Correspondence between Reset Cause FLAG Bits and Reset Causes

Figure 3.5-2 Configuration of Reset Cause Bits (Watchdog Timer Control Register)

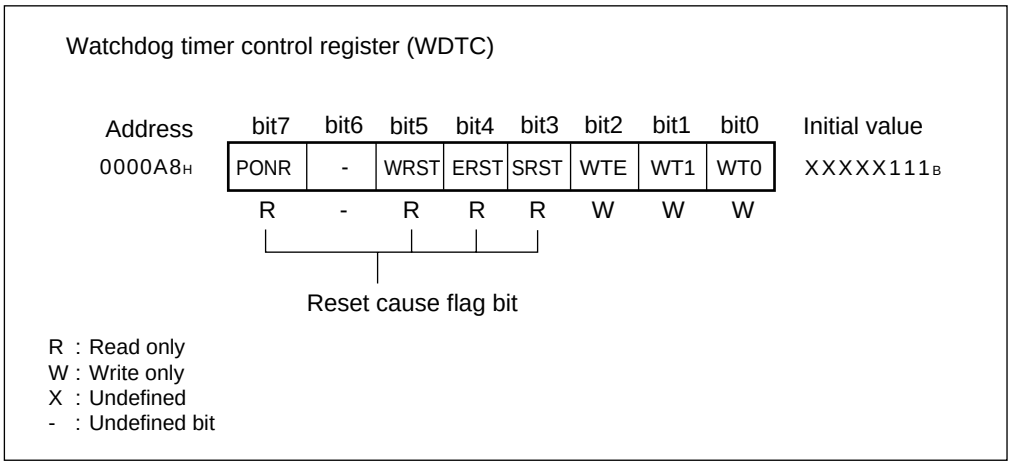


Table 3.5-1 Correspondence between Reset Cause Bits and Reset Causes

Reset cause	PONR	WRST	ERST	SRST
Power-on reset	1	X	X	X
Watchdog timer overflow	*	1	*	*
External reset request via RSTX pin	*	*	1	*
Software reset request	*	*	*	1

*: Previous state retained
X: Undefined

■ Notes about Reset Cause Bits

❍ Multiple reset causes generated at the same time

When multiple reset causes are detected, the PONR, WRST, ERST, and SRST bits of the watchdog timer control register (WDTC) are set to "1".

[Example]

If an external reset and a watchdog timer reset occur at the same time, the ERST and WRST bits of the watchdog timer control register (WDTC) are set to "1".

❍ Power-on reset

If a power-on reset occurs, the PONR bit of the watchdog timer control register (WDTC) is set to "1" and the WRST, ERST, and SRST bits are undefined.

If the PONR bit is set to "1", the contents of the WRST, ERST, and SRST bits should be ignored.

❍ Clearing the reset cause flag bits

The PONR, WRST, ERST, and SRST bits are cleared to "0" if the watchdog timer control register (WDTC) is read. In other words, these reset cause flag bits are not cleared to "0" unless the watchdog timer control register (WDTC) is read even if a reset occurs.

3.6 Status of Pins in a Reset

This section describes the status of pins when a reset occurs.

■ Status of Pins during a Reset

Ports 0 to 6 becomes a high impedance output by reset, and the mode data is read from internal ROM (or flash memory).

■ Status of Pins after Mode Data is Read

After mode data is read, Ports 0 to 6 becomes a high impedance output, and the mode data is read from internal ROM (or flash memory).

Note:

Specify an external pin level that disables external circuits.

See Table 5.7-1 "State of Pins in Single-Chip Mode" for information about the state of pins during a reset.

CHAPTER 4 CLOCKS

This chapter describes the clocks used by MB90560/565-series.

4.1 Clocks

The clock generation block controls the operating clock of the CPU and peripheral functions (resources). The following four clocks are available:

- **Oscillation clock**
- **Main clock**
- **PLL clock**
- **Machine clock**

■ Clocks

The clock generation block contains the oscillation circuit and the PLL clock multiplier circuit. The clock generation block controls the oscillation stabilization wait interval and PLL clock multiplication as well as controls the operation of switching the clock with a clock selector.

❍ Oscillation clock frequency (HCLK)

The oscillation clock is generated either from an oscillator connected to the X0 and X1 pins or by input of an external clock.

❍ Main clock

The main clock, which is the oscillation clock divided by 2, supplies the clock input to the timebase timer and the clock selector.

❍ PLL clock (PCLK)

The PLL clock is obtained by multiplying the oscillation clock in the internal PLL clock multiplier circuit. Four different clocks (multiplied by 1 through 4) can be generated.

❍ Machine clock

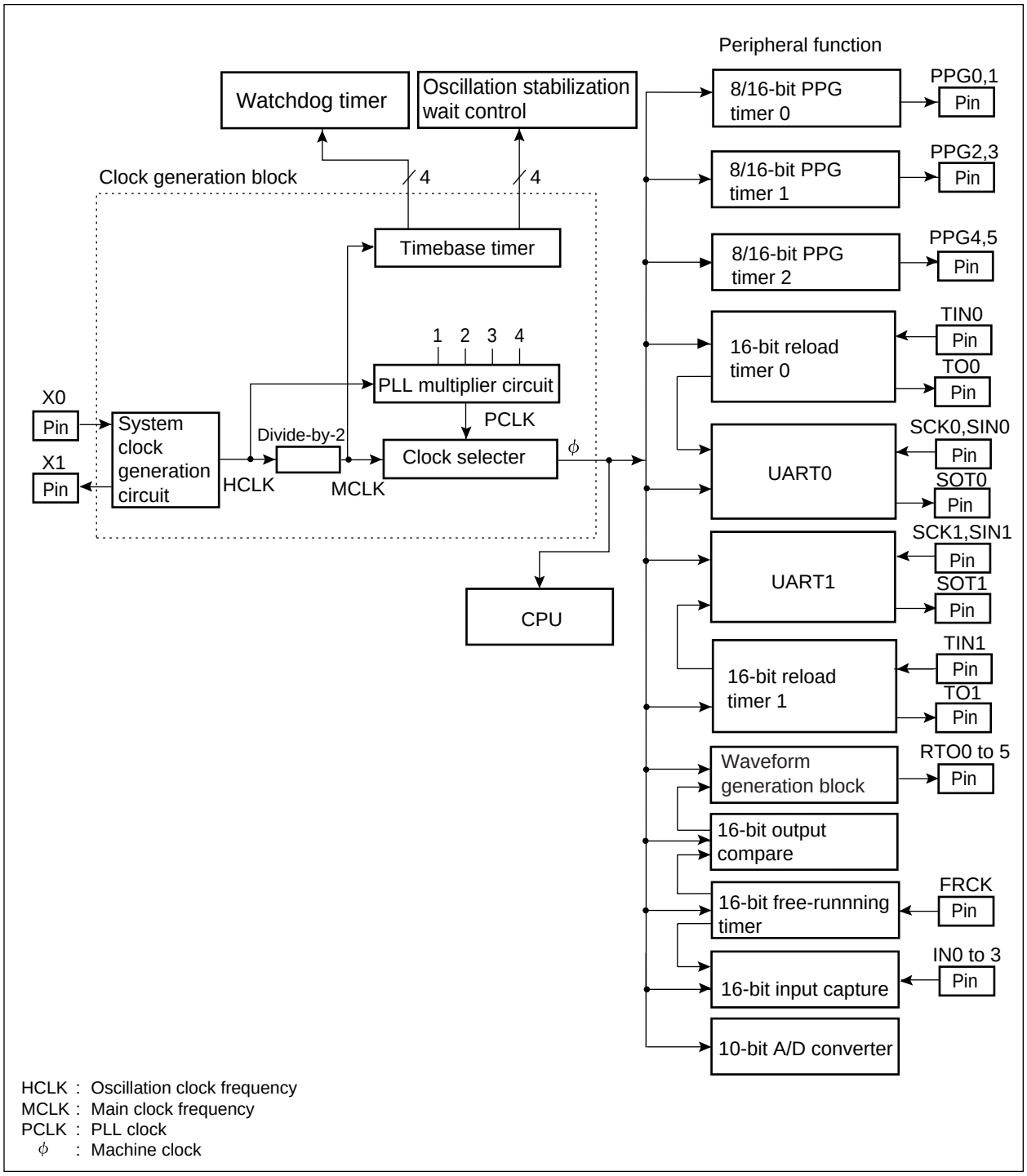
The machine clock is the operating clock of the CPU and peripheral functions (resources). One machine clock cycle is called a machine cycle. Either the main clock or a PLL clock can be selected.

Note:

If the operating voltage is 5 V, an oscillation clock frequency from 3 MHz to 16 MHz can be used. The maximum operating frequency of the CPU and peripheral functions (resources) is 16 MHz. If a frequency multiplier higher than the operating frequency is specified, devices will not operate normally. If the oscillation clock of 16MHz is generated, only a multiplier of 1 can be specified.

■ Clock Supply Map

Figure 4.1-1 Clock Supply Map



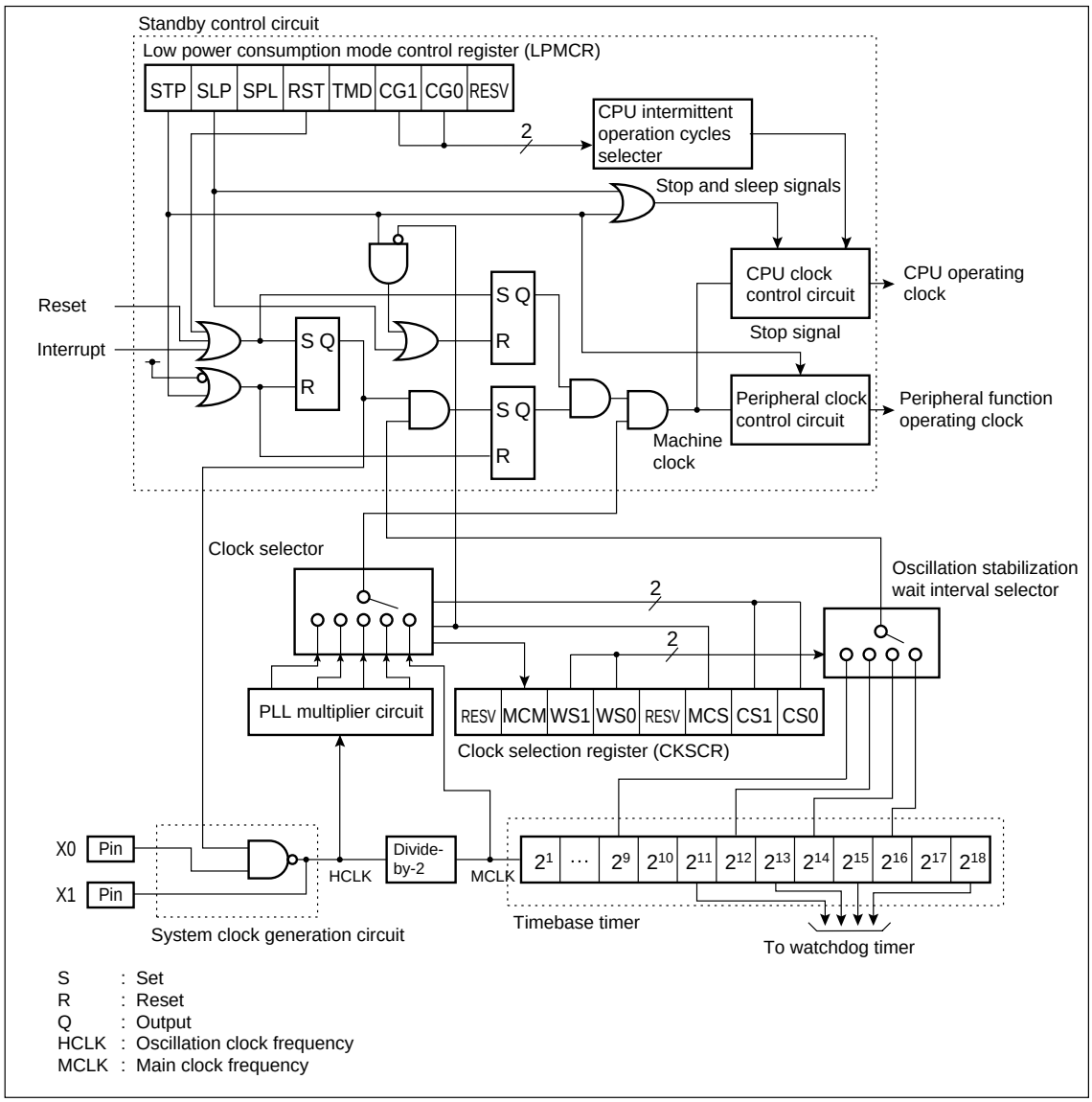
4.2 Block Diagram of the Clock Generation Block

The clock generation block consists of the following five blocks:

- **System clock generation circuit**
- **PLL multiplier circuitS**
- **Clock selector**
- **Clock selection register (CKSCR)**
- **Oscillation stabilization wait interval selector**

■ Block Diagram of the Clock Generation Block

Figure 4.2-1 Block Diagram of the Clock Generation Block



❍ **System clock generation circuit**

The system clock generation circuit generates an oscillation clock from an oscillator connected to the X0 and X1 pins or by input of an external clock.

❍ **PLL multiplier circuit**

The PLL multiplier circuit multiplies the oscillation clock and supplies the resultant clock to the clock selector.

❍ **Clock selector**

The clock selector selects the clock to be supplied to the CPU and peripheral clock control circuits from among the main clock and the PLL clock.

❍ **Clock selection register (CKSCR)**

The clock selection register selects the machine clock and determines the oscillation stabilization wait interval and the PLL clock multiplier, etc.

❍ **Oscillation stabilization wait interval selector**

The oscillation stabilization wait interval selector selects the oscillation stabilization wait interval of the oscillation clock when the stop mode is cleared. One of the four time-base timer outputs is selected to determine the oscillation stabilization wait interval.

4.3 Clock Selection Register (CKSCR)

The clock selection register (CKSCR) switches the machine clock and sets the oscillation stabilization wait interval and the PLL clock multiplier, etc.

■ Configuration of the Clock Selection Register (CKSCR)

Figure 4.3-1 Configuration of the Clock Selection Register (CKSCR)

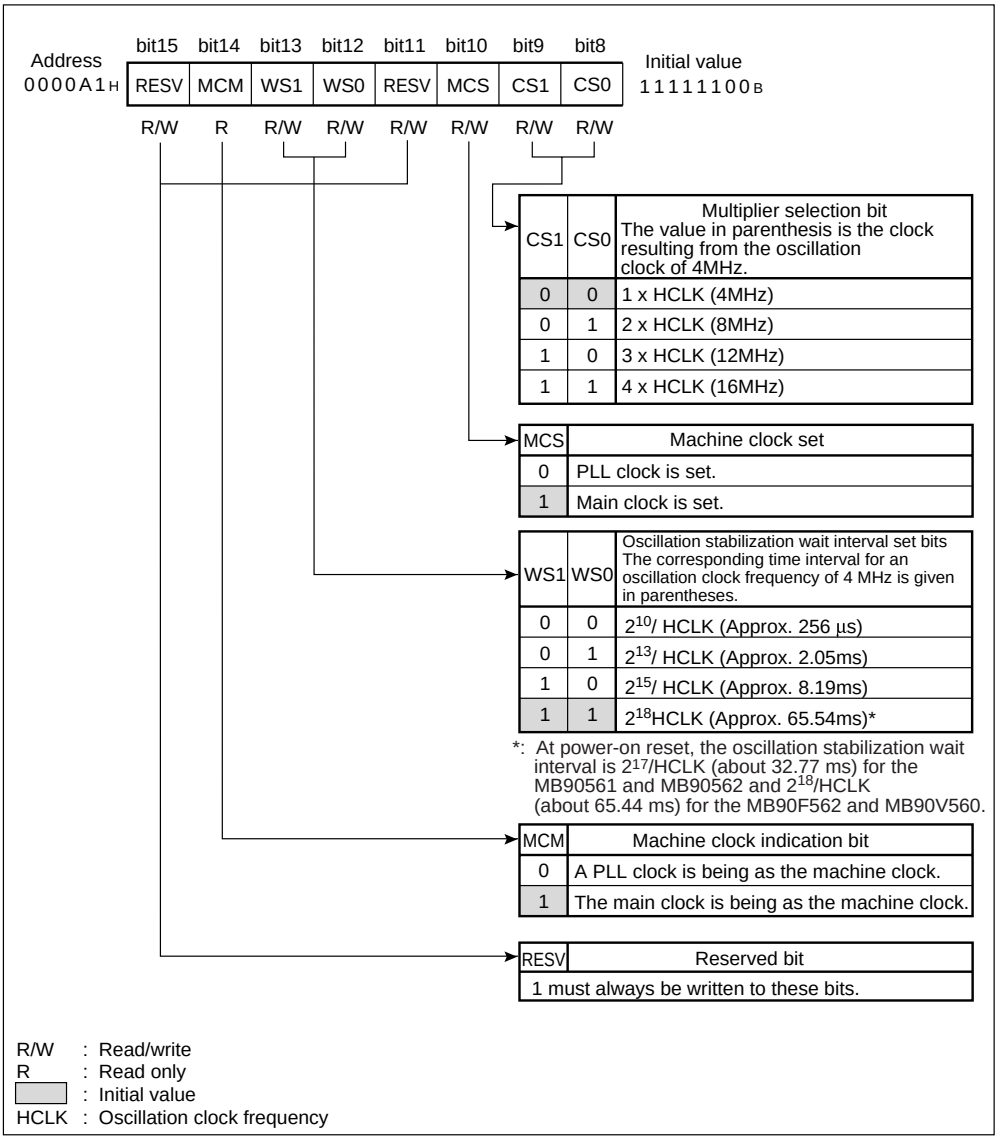


Table 4.3-1 Function Description of Each Bit of the Clock Selection Register (CKSCR)

Bit name		Function
bit 15 bit 11	RESV: Reserved bit	• Always set "1".
bit 14	MCM: Machine clock indication bit	• This bit indicates whether the main clock or a PLL clock has been selected as the machine clock. • When this bit is set to "0", a PLL clock has been selected. • When this bit is set to "1", the main clock has been selected. • If the machine clock selection bit (MCS) is set to "0" and MCM is set to "1", the PLL clock oscillation stabilization wait interval is in effect.
bit 13 bit 12	WS1, WS0: WS1, WS0: Oscillation stabilization wait interval selection bits	• These bits select the oscillation stabilization wait interval for the oscillation clock after the stop mode has been cleared due to an external interrupt. • A reset cause initializes these bits to "11_B". • Specify an oscillation stabilization wait interval appropriate for the oscillator used.
bit 10	MCS: Machine clock selection bit	• This bit specifies whether the main clock or a PLL clock is selected as the machine clock. • When this bit is set to "0", a PLL clock is selected. • When this bit is set to "1", the main clock is selected. • If this bit has been set to "1" and is reset to "0", the oscillation stabilization wait interval for the PLL clock starts. As a result, the time-base timer counter and the interrupt request flag bit (TBOF) of the time-base timer counter control register (TBTC) are cleared to "0". • For PLL clocks, the oscillation stabilization wait interval is fixed to 2^{14}/HCLK. The oscillation stabilization wait interval is about 4.1 ms if the oscillation clock frequency is 4 MHz.) • When the main clock has been selected, the oscillation clock divided by 2 is used as the machine clock. The machine clock frequency is 2 MHz if the oscillation clock frequency is 4 MHz. • A reset initializes this bit to 1. **Note:** The MCS bit set to "1" can be reset to "0" while the interrupt request enable bit (TBIE) of the time-base timer counter control register (TBTC) or the interrupt level mask register (ILM) are set to disable timer-base timer interrupt requests.
bit 9 bit 8	CS1, CS0: Multiplier selection bits	• These bits select a PLL clock multiplier. • One of the four multipliers can be selected. • A reset initializes these bits to "00_B". **Note:** These bits cannot be set while the machine clock selection bit (MCS) or the machine clock indication bit (MCM) is set to "0". Set these bits only after setting the MCS bit to "1".

HCLK: Oscillation clock frequency

4.4 Clock Mode

Two clock modes are provided: main clock mode and PLL clock mode.

■ Main Clock Mode and PLL Clock Mode

❍ Main clock mode

In main clock mode, the main clock is used as the machine clock of the CPU and peripheral functions (resources) while the PLL clocks are disabled.

❍ PLL clock mode

In PLL clock mode, a PLL clock is used as the machine clock of the CPU and peripheral functions (resources). Specify a PLL clock multiplier in the multiplier selection bits (CS1 and CS0) of the clock selection register (CKSCR).

■ Clock Mode Transition

Setting the machine clock selection bit (MCS) of the clock selection register (CKSCR) causes switching between main clock mode and PLL clock mode.

❍ Switching from main clock mode to PLL clock mode

When the MCS bit of the CKSCR that is set to "1" is reset to "0", switching from the main clock to a PLL clock occurs after the PLL clock oscillation stabilization wait interval (2^{14}/HCLK) has elapsed.

❍ Switching from PLL clock mode to main clock mode

When the MCS bit of the CKSCR that is set to "0" is reset to "1", switching from a PLL clock to the main clock occurs when the edges of the PLL clock and the main clock coincide (after 1 to 8 PLL clocks).

Note:

Before setting the peripheral functions (resources) after the machine clock switching, make sure that the machine clock has been switched by referring to the MCM bit of the CKSCR.

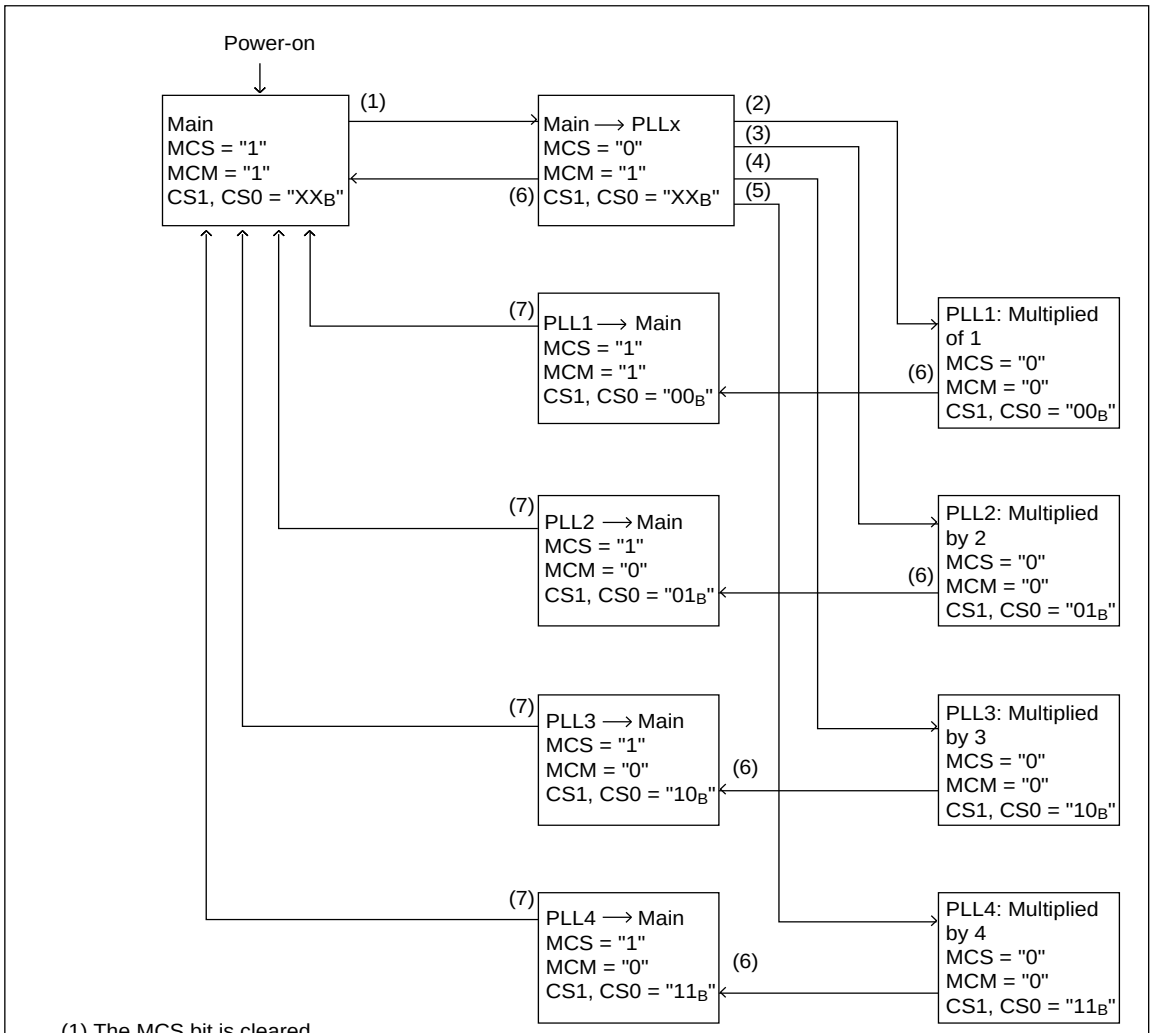
■ Selection of a PLL Clock Multiplier

Set the multiplier selection bits (CS1 and CS0) of the CKSCR to "00_B" and "11_B" to set one of the four PLL clock multipliers (1 through 4).

■ Machine Clock

Either the main clock or a PLL clock is used as the machine clock. The machine clock is an operating clock of the CPU and peripheral functions (resources). Set either the main clock or a PLL clock in the MCS bit of the CKSCR.

Figure 4.4-1 Status Change Diagram for Machine Clock Selection



(1) The MCS bit is cleared.
(2) The PLL clock oscillation stabilization wait ends with CS1 and CS0 = 00.
(3) The PLL clock oscillation stabilization wait ends with CS1 and CS0 = 01.
(4) The PLL clock oscillation stabilization wait ends with CS1 and CS0 = 10.
(5) The PLL clock oscillation stabilization wait ends with CS1 and CS0 = 11.
(6) The MCS bit is set (including watchdog timer resets).
(7) PLL clock and main clock frequency synchronization timing
MCS: Machine clock set bit of CKSCR
MCM: Machine clock indication bit of CKSCR
CS1, CS0: Multiplier set bits of CKSCR

Note:

The initial value for the machine clock setting is main clock (CKSCR:MCS = 1).

4.5 Oscillation Stabilization Wait Interval

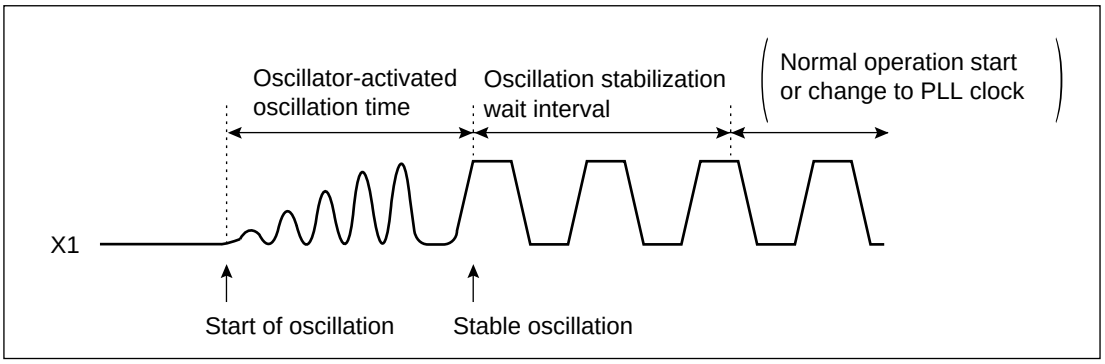
Whenever the power is turned on, or whenever stop mode is cleared, oscillation begins following a state in which there was no oscillation. Accordingly, an oscillation stabilization wait interval is required. Also, whenever the switching from the main clock to a PLL clock occurs, an oscillation stabilization wait interval is required after the oscillation of the PLL clock starts.

■ Oscillation Stabilization Wait Interval

Specify an oscillation stabilization wait interval appropriate for the oscillator used because the oscillation stabilizes in different lengths of time depending on the oscillator type. Specify an appropriate oscillation stabilization wait interval in the oscillation stabilization wait interval selection bits (WS1 and WS0) of the clock selection register (CKSCR).

When switching from the main clock to a PLL clock occurs, the CPU operates on the main clock during an oscillation stabilization wait interval and starts to operate on a PLL clock.

Figure 4.5-1 Operation When Oscillation Starts



4.6 Connection of an Oscillator or an External Clock to the Microcontroller

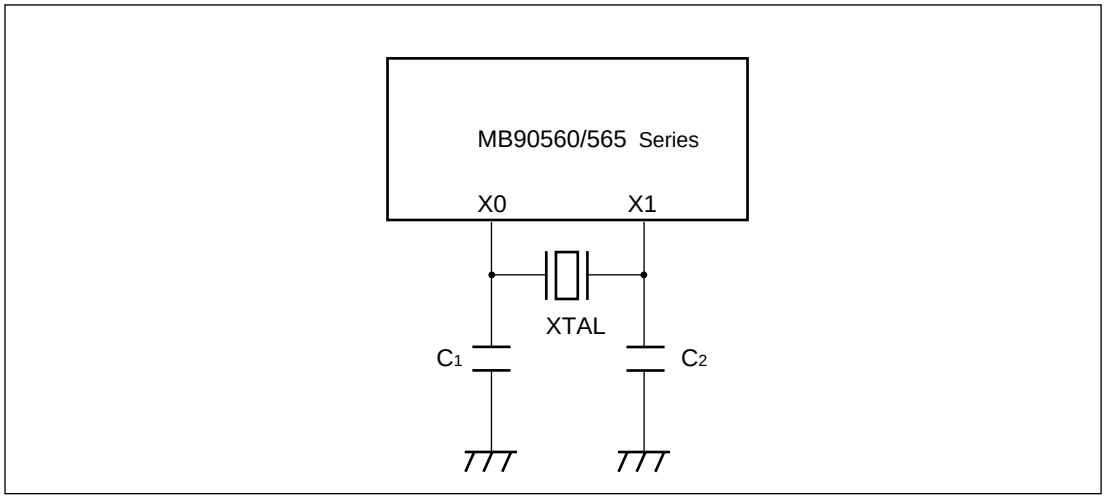
The MB90560 and 565 series contains a system clock generation circuit. An oscillator can be connected to the X0 and X1 pins. Alternatively, an external clock may be input.

■ Connection of an Osillator or an External Clock to the Microcontroller

❍ Example of connecting a crystal or ceramic oscillator to the microcontroller

Connect a crystal or ceramic oscillator as shown in the example in Figure 4.6-1 "Example of Connecting a Crystal or Ceramic Oscillator to the Microcontroller".

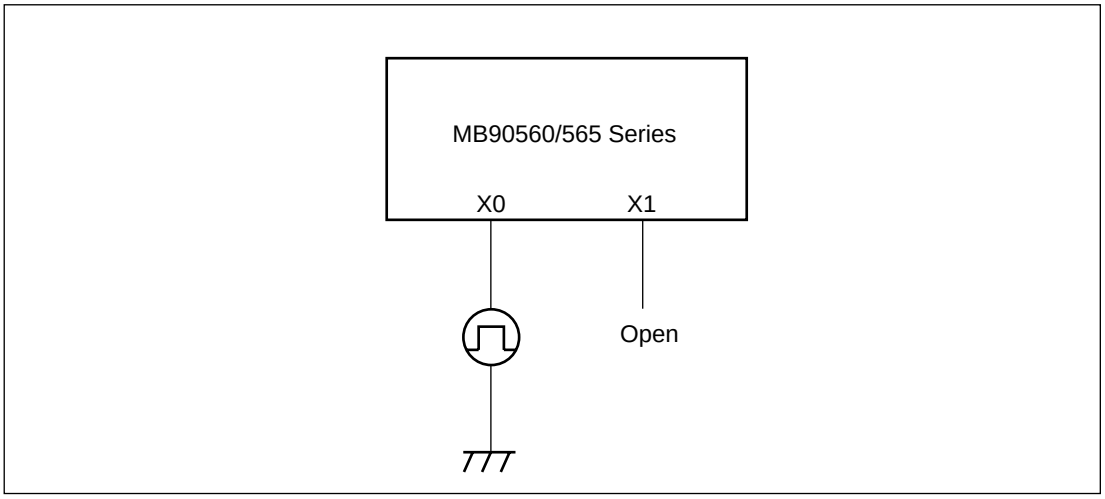
Figure 4.6-1 Example of Connecting a Crystal or Ceramic Oscillator to the Microcontroller



❍ Example of connecting an external clock to the microcontroller

As shown in Figure 4.6-2 "Example of Connecting an External Clock to the Microcontroller", connect an external clock to pin X0. Pin X1 must be open.

Figure 4.6-2 Example of Connecting an External Clock to the Microcontroller



CHAPTER 5 LOW POWER CONSUMPTION MODE

This chapter describes the low power consumption mode of MB90560/565 series.

5.1 Low Power Consumption Mode

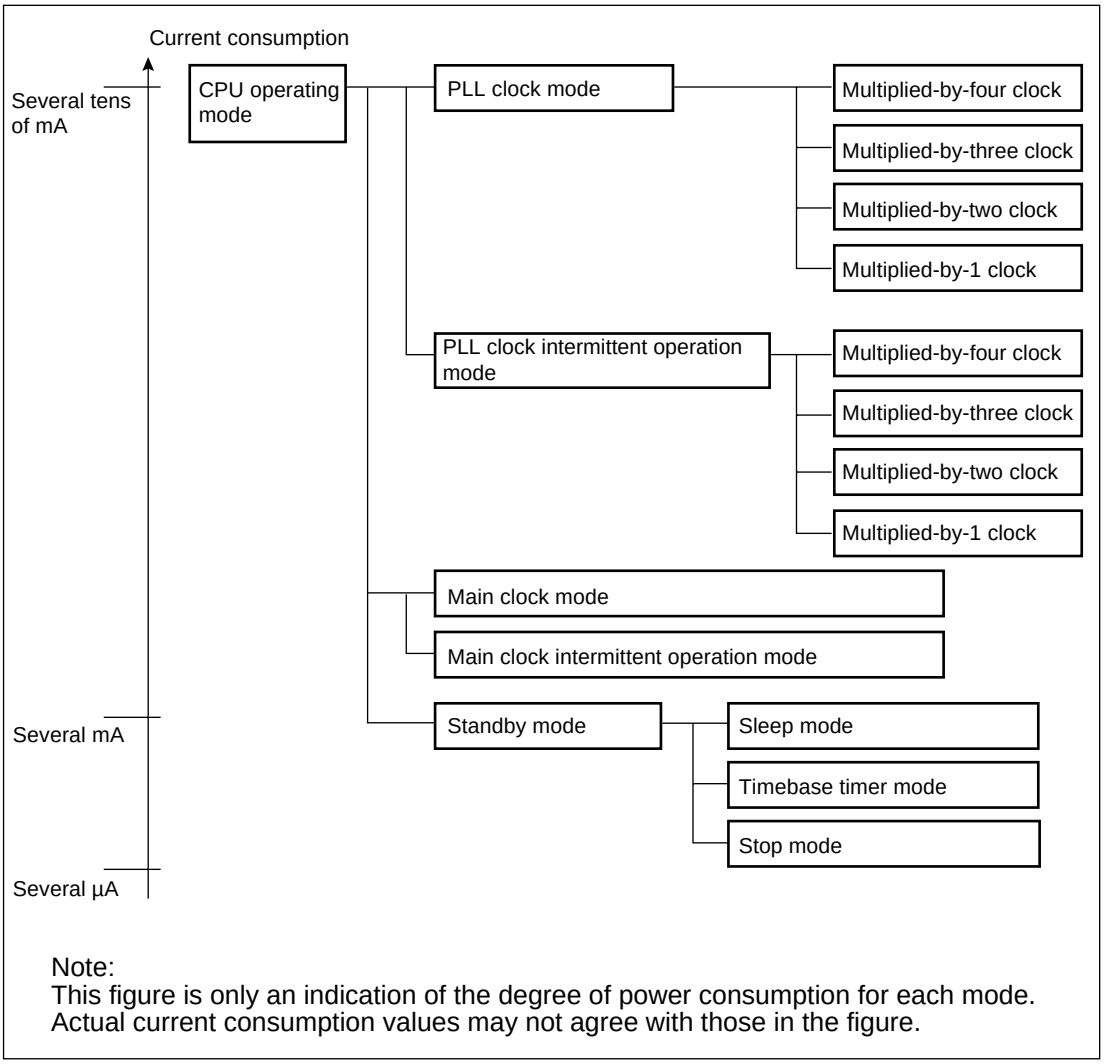
The MB90560 and 565 series have the following low power consumption modes, one of which can be selected depending on the operating clock setting and the clock operation control.

- **CPU intermittent operation mode (PLL clock intermittent operation mode and main clock intermittent operation mode)**
- **Standby mode (sleep mode, timebase timer mode)**

All modes other than PLL clock mode are low power consumption modes.

■ CPU Operating Modes and Current Consumption

Figure 5.1-1 CPU Operating Modes and Current Consumption



Note:
This figure is only an indication of the degree of power consumption for each mode.
Actual current consumption values may not agree with those in the figure.

■ Clock Mode

❍ PLL clock mode

The CPU and peripheral functions (resources) operate on a PLL clock.

❍ Main clock mode

The CPU and peripheral functions (resources) operate on the main clock. In the main clock mode, the PLL multiplier circuit is disabled.

Reference:

See Section 4.4 "Clock Mode", for details about clock mode.

■ CPU Intermittent Operation Mode

The CPU operates intermittently while the machine clock is supplied to the peripheral functions (resources).

■ Standby Mode

❍ PLL sleep mode

The CPU operating clock is stopped. Other components continue to operate on a PLL clock.

❍ Main sleep mode

The CPU operating clock is stopped. Other components continue to operate on the main clock.

❍ Timebase timer mode

All the components but the oscillation clock and the timebase timer are stopped.

❍ Stop mode

The oscillation clock is stopped. All the functions are stopped.

Note:

Because stop mode and hardware standby mode turn the oscillation clock off, these modes save the most power while data is being retained. Because the MB90560/565 series does not have a pin for the hardware standby function, the stop mode cannot be used.

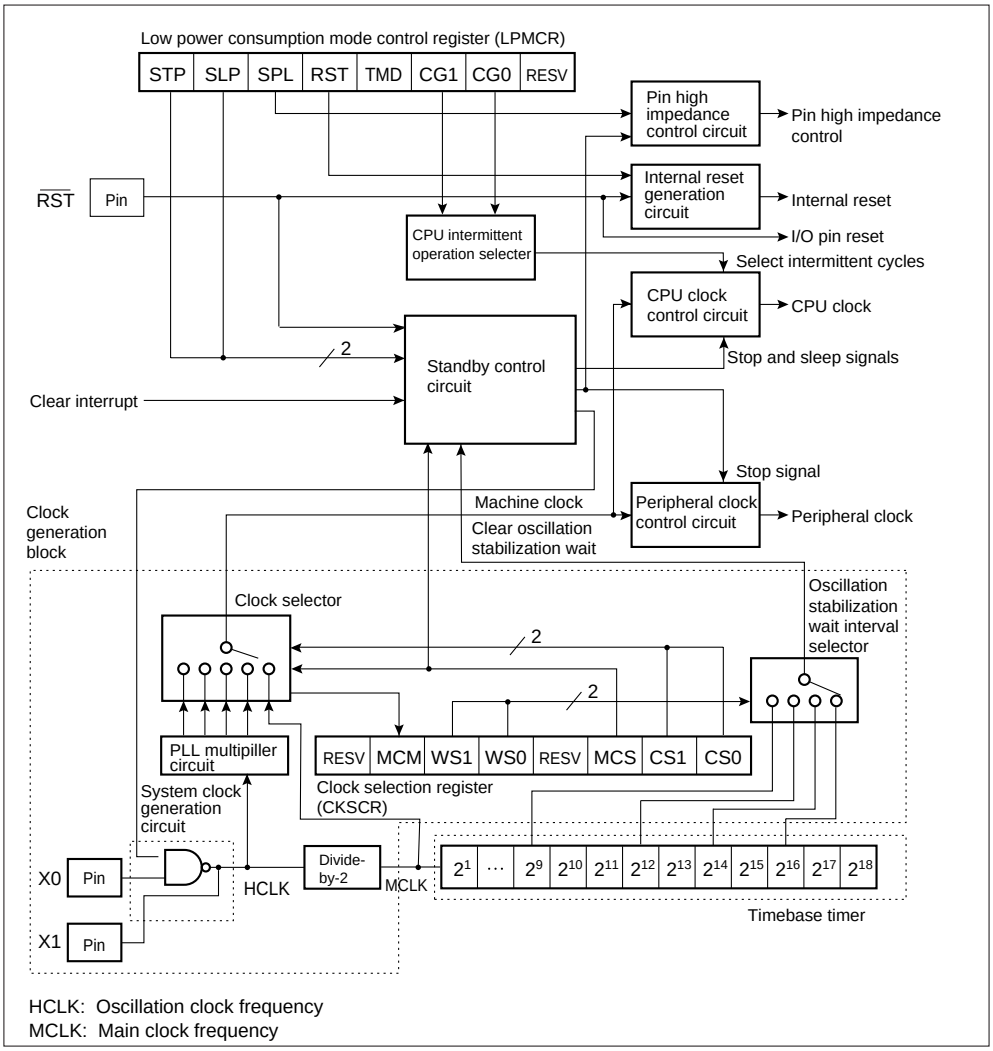
5.2 Block Diagram of the Low Power Consumption Control Circuit

The low power consumption control circuit consists of the following seven blocks:

- **CPU intermittent operation selector**
- **Standby clock control circuit**
- **CPU clock control circuit**
- **Peripheral clock control circuit**
- **Pin high-impedance control circuit**
- **Internal reset generation circuit**
- **Low power consumption mode control register (LPMCR)**

■ Block Diagram of the Low Power Consumption Control Circuit

Figure 5.2-1 Block Diagram of the Low Power Consumption Control Circuit



❍ **CPU intermittent operation selector**

This selector selects the number of clock pulses during which the CPU is halted in the CPU intermittent operation mode.

❍ **Standby control circuit**

This circuit controls the CPU clock control circuit, the peripheral clock control circuit, and the pin high-impedance control circuit to switch to, or release low power consumption mode.

❍ **CPU clock control circuit**

This circuit control the clocks supplied to the CPU.

❍ **Peripheral clock control circuit**

This circuit control the clocks supplied to the peripheral functions (resources).

❍ **Pin high-impedance control circuit**

A setting of this circuit causes the I/O pins to have high impedance in timebase timer mode or stop mode. For the I/O pins configured to accept the connection of a pull-up resistor, this circuit disconnects the pull-up resistor in stop mode.

❍ **Internal reset generation circuit**

This circuit generates an internal reset signal.

❍ **Low power consumption mode control register (LPMCR)**

This register selects the switching to, or release of low power consumption mode and the number of clock pulses during which the CPU is halted in CPU intermittent operation mode.

5.3 Low Power Consumption Mode Control Register (LPMCR)

The low power consumption mode control register (LPMCR) selects the switching to, or release of low power consumption mode and the number of clock pulses during which the CPU is halted in CPU intermittent operation mode.

■ Low Power Consumption Mode Control Register (LPMCR)

Figure 5.3-1 Configuration of the Low Power Consumption Mode Control Register (LPMCR)

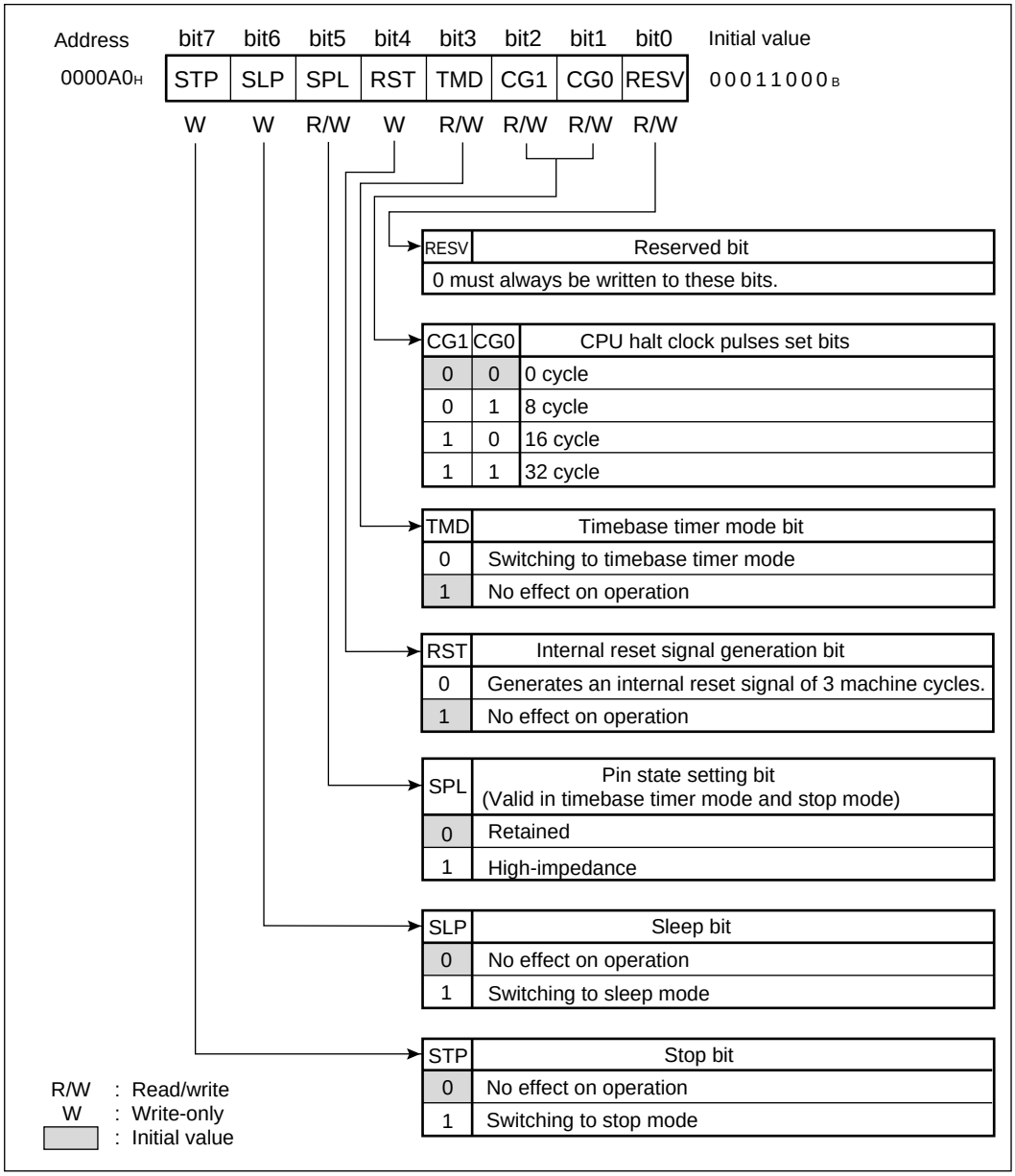


Table 5.3-1 Function Description of Each Bit of the Low Power Consumption Mode Control Register (LPMCR)

Bit name		Function
bit 7	STP: Stop bit	• This bit selects the stop mode. • When this bit is set to "1", the microcontroller enters stop mode. • When this bit is set to "0", there is no effect on operation. • An external reset or the output of a hardware interrupt resets this bit to "0". • The read value of this bit is "0".
bit 6	SLP: Sleep bit	• This bit selects sleep mode. • When this bit is set to "1", the microcontroller enters sleep mode. • When this bit is set to "0", there is no effect on operation. • An external reset or the output of a hardware interrupt resets this bit to "0". • The read value of this bit is "0".
bit 5	SPL: Pin state setting bit valid in timebase timer mode or stop mode)	• This bit selects a terminal status in timebase timer mode or stop mode. • When this bit is set to "0", an I/O pin has a retained level. • When this bit is set to "1", an I/O pin has high impedance. • An external reset resets this bit to "0".
bit 4	RST: Internal reset signal generation bit	• This bit selects an internal reset. • When this bit is set to "0", an internal reset signal of three machine cycles is generated. • When this bit is set to "1", there is no effect on operation. • The read value of this bit is "1".
bit 3	TMD: Timebase timer mode bit	• This bit selects the switching to timebase timer mode. • When this bit is set to "0", the microcontroller enters timebase timer mode. • When this bit is set to "1", there is no effect on operation. • An external reset or the output of a timebase timer interrupt initializes this bit to "1". • The read value of this bit is "1".
bit 2 bit 1	CG1, CG0: CPU halt clock pulses selection bits	• These bits set the number of CPU halt clock pulses in CPU intermittent operation mode. • The clock supplied to the CPU is stopped for the specified number of clock cycles every time after an instruction is executed. • These bits select one of the four clock pulses. • A reset initializes these bits to "00_B".
bit 0	RESV: Reserved bit	• This bit must be set to "0".

■ Access to the Low Power Consumption Mode Control Registor Rister

Use one of the instructions listed in Table 5.3-2 "Instructions to Be Used for Switching to Low Power Consumption Mode" to set low power consumption mode control register. The operation is not assured if any instruction other than that listed in Table 5.3-2 "Instructions to Be Used for Switching to Low Power Consumption Mode" is used to enter low power consumption mode.

When a word length is used to set the low power consumption mode control register, use word access to the low power consumption control circuit ($0000A0_H$, $0000A1_H$).

Table 5.3-2 Instructions to Be Used for Switching to Low Power Consumption Mode

MOV io,#imm8	MOV dir,#imm8	MOV eam,#imm8	MOV eam,Ri
MOV io,A	MOV dir,A	MOV addr,A	MOV eam,A
MOV @RLi+disp8,A	MOVP addr24,A		
MOVW io,#imm16	MOVW dir,#imm16	MOVW eam,#imm16	MOVW eam,RWi
MOVW io,A	MOVW dir,A	MOVW addr16	MOVW eam,A
MOVW @RLi+disp8,A	MOVPW addr24,A		

■ Priority of standby modes

If stop mode (LPMCR: STP), sleep mode (LPMCR: SLP), and timebase timer mode (LPMCR: TMD) are simultaneously specified, the microcontroller enters the following order of priority:

stop mode, timebase timer mode, or sleep mode

5.4 CPU Intermittent Operation Mode

In CPU intermittent operation mode, the CPU operates intermittently while the peripheral functions (resources) operate on the machine clock.

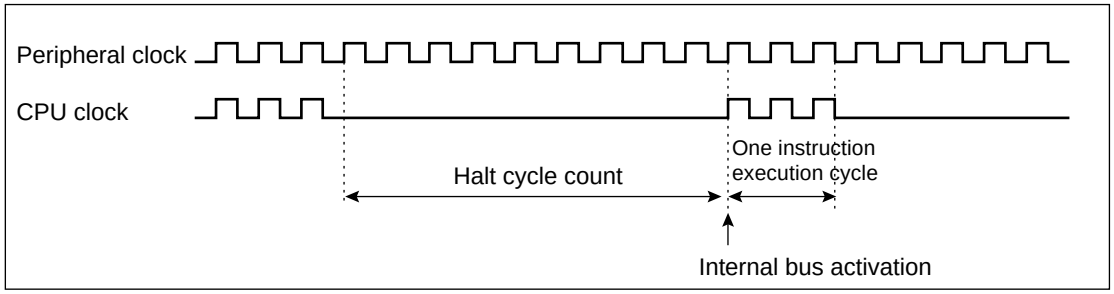
■ CPU Intermittent Operation Mode

In CPU intermittent operation mode, the machine clock to be supplied to the CPU is halted for a certain period every time after an instruction is executed, so that the activation of an internal bus cycle is delayed.

The "halt cycle count" shown in Figure 5.4-1 "Clock Pulses during CPU Intermittent Operation" can be specified in the CPU operating clock halt cycle count setting bits (CG1 and CG0) of the low power consumption mode control register (LPMCR).

For more information on the halt cycle count, see Section 5.3 "Low Power Consumption Mode Control Register (LPMCR)".

Figure 5.4-1 Clock Pulses during CPU Intermittent Operation



5.5 Standby Mode

Standby mode includes the sleep mode (PLL sleep mode and main sleep mode), timebase timer mode, and stop mode.

■ Operating Status during Standby Mode

Table 5.5-1 Operation Statuses in Standby Mode

Standby mode		Condition for switch	Oscil-lation	Clock	CPU	Timebase timer	Peripheral	Pin	Release event
Sleep mode	PLL sleep mode	MCS = 0 SLP = 1	Active	Active	In-active	Active	Active	Active	External Rese t or Interrupt
	Main sleep mode	MCS = 1 SLP = 1							
Time-base timer mode	Timebase timer mode (SPL = 0)	TMD = 0		In-active	In-active		Inactive	Hold	External reset or Interrupt (*1)
	Timebase timer mode (SPL = 1)	TMD = 0						Hi-z	
Stop mode	Stop mode (SPL = 0)	STP = 1	In-active			Inactive	Inactive	Hold	External reset or Interrupt (*2)
	Stop mode (SPL = 1)	STP = 1						Hi-z	

*1: Timebase timer
*2: External interrupt
SPL: Pin state setting bit of low power consumption mode control register (LPMCR)
SLP: Sleep bit of low power consumption mode control register (LPMCR)
STP: Timebase timer or stop bit of low power consumption mode control register (LPMCR)
TMD: Timebase timer mode bit of low power consumption mode control register (LPMCR)
MCS: Machine clock selection bit of clock set register (CKSCR)
Hi-z: High-impedance

5.5.1 Sleep Mode

In sleep mode, the CPU operating clock is halted while components other than the CPU continue to operate. When switching to sleep mode is specified, the microcontroller enters PLL sleep mode if PLL clock mode has been specified or the main sleep mode if the main clock mode has been specified.

■ Switching to Sleep Mode

Set the sleep mode bit (SLP) of the low power consumption mode control register (LPMCR) to "1", the timebase timer mode bit (TMD) to "1", and the stop mode bit (STP) to "0" to enter sleep mode. When switching to sleep mode is specified, the microcontroller enters PLL sleep mode if the machine clock setting bit (MCS) of the clock selection register (CKSCR) is set to "0". Alternatively, the microcontroller enters main sleep mode if the MCS is set to "1".

❍ Data retention function

In sleep mode, the contents of both the dedicated registers and internal RAM are retained.

For more information on dedicated registers, see Section 2.7 "Dedicated Registers".

❍ Hold function

In sleep mode, the hold function is enabled. A hold request sets the hold status.

❍ Operation during output of an interrupt request

While an interrupt request is output, the microcontroller does not enter sleep mode but executes the next instruction even if the sleep mode bit (SLP) of the low power consumption mode control register (LPMCR) is set to "1".

■ Release of Sleep Mode

An external reset or the output of a hardware interrupt releases sleep mode.

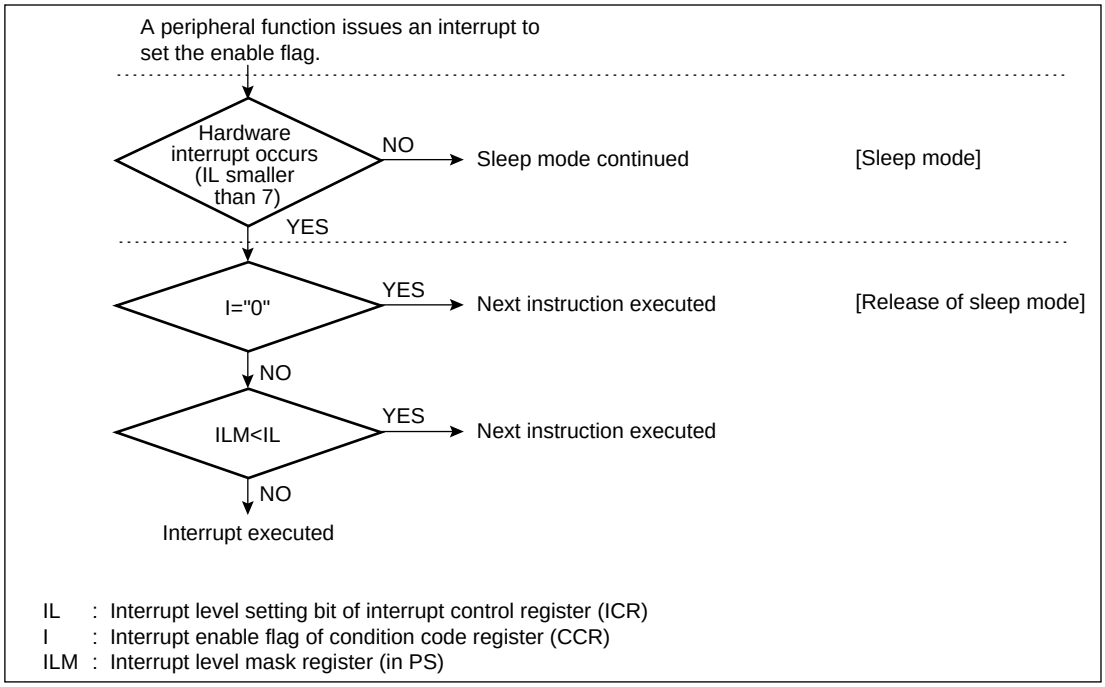
❍ Release by an external reset

For more information, see Section 3.4 "Reset Operation".

❍ Release by a hardware interrupt

An interrupt request with an interrupt level higher than 7 (Interrupt control register ICR: IL2, IL1, IL0 = "000_B" to "110_B") is required to cause a hardware interrupt to release sleep mode.

Figure 5.5-1 Release of Sleep Mode by the Output of a Hardware Interrupt



5.5.2 Timebase Timer Mode

In timebase timer mode, all of the operations other than the source oscillation and the timebase timer are stopped.

■ Switching to Timebase Timer Mode

If the timebase timer mode bit (TMD) of the low power consumption mode control register (LPMCR) is set to "0", the microcontroller enters timebase timer mode.

❍ Data retention function

In timebase timer mode, the contents of both the dedicated registers and internal RAM are retained.

For more information on dedicated registers, see Section 2.7 "Dedicated Registers".

❍ Hold function

In timebase timer mode, the hold function is disabled. A hold request that is input in timebase timer mode causes the external bus pins to have high impedance and sets the HAKX pin to "H".

❍ Operation during output of an interrupt request

While an interrupt request is output, the microcontroller does not enter timebase timer mode even if the timebase timer mode bit (TMD) of the low power consumption mode control register (LPMCR) is set to "0".

❍ Status of pins

The I/O pins in timebase timer mode can be set to retain a previous level or have high impedance in the pin status setting bit (SPL) of the low power consumption mode control register (LPMCR).

■ Release of Timebase Timer Mode

An external reset or the output of a hardware interrupt releases timebase timer mode.

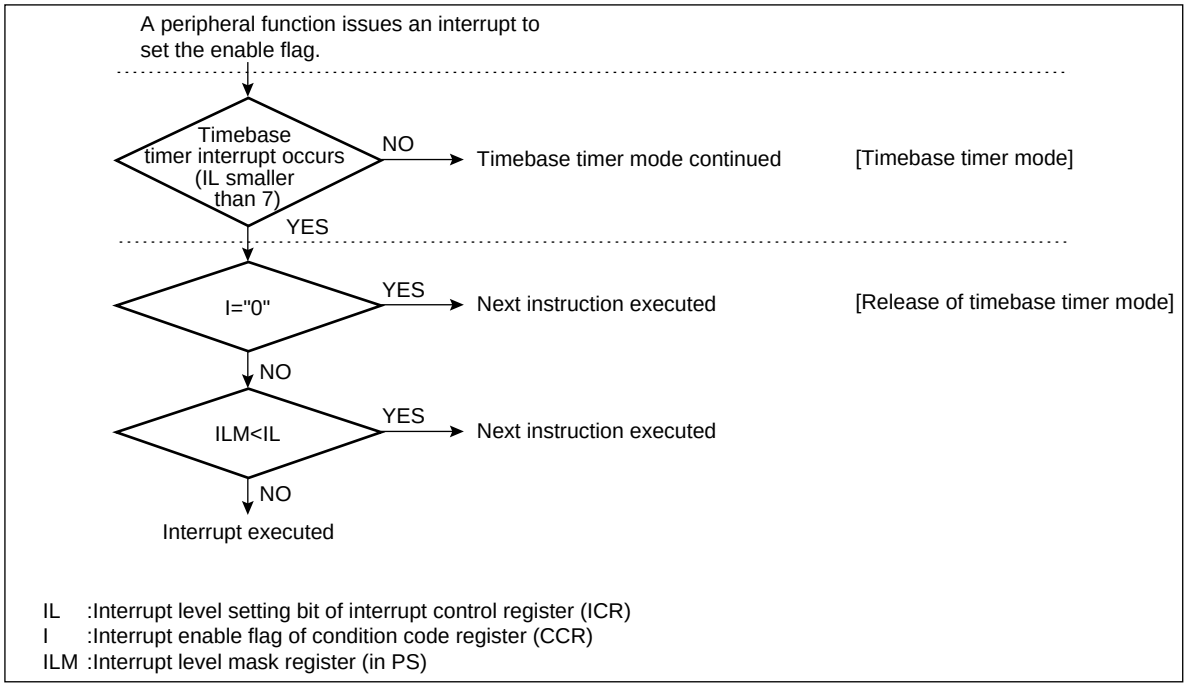
❍ Release by an external reset

For more information, see Section 3.4 "Reset Operation".

❍ **Release by a timebase timer interrupt**

If timebase timer mode is released by a timebase timer interrupt, a timebase timer with an interrupt level higher than 7 interrupt request (interrupt control register ICR: IL2, IL1, IL0 = "000_B" to "110_B") are required.

Figure 5.5-2 Release of Timebase timer Mode by an External Interrupt Output



5.5.3 Stop Mode

In stop mode, the source oscillation is stopped. Since all the functions are stopped, data can be retained while minimum power is consumed.

■ Switching to Stop Mode

If the stop mode bit (STP) of the low power consumption mode control register (LPMCR) is set to "1", the microcontroller enters stop mode.

❍ Data retention function

In stop mode, the contents of both the dedicated registers and RAM are retained.

For more information on dedicated registers, see Section 2.7 "Dedicated Registers".

❍ Hold function

In stop mode, the hold function is disabled. A hold request that is input in stop mode causes the external bus pins to have high impedance and sets the HAKX pin to "H".

❍ Operation during output of an interrupt request

While an interrupt request is output, the microcontroller does not enter stop mode even if the stop mode bit (STP) of the low power consumption mode control register (LPMCR) is set to "1".

❍ Status of pins

The I/O pins in stop mode can be set to retain a previous level or have high impedance in the pin status setting bit (SPL) of the low power consumption mode control register (LPMCR).

■ Release of Stop Mode

An external reset or the output of a hardware interrupt releases stop mode.

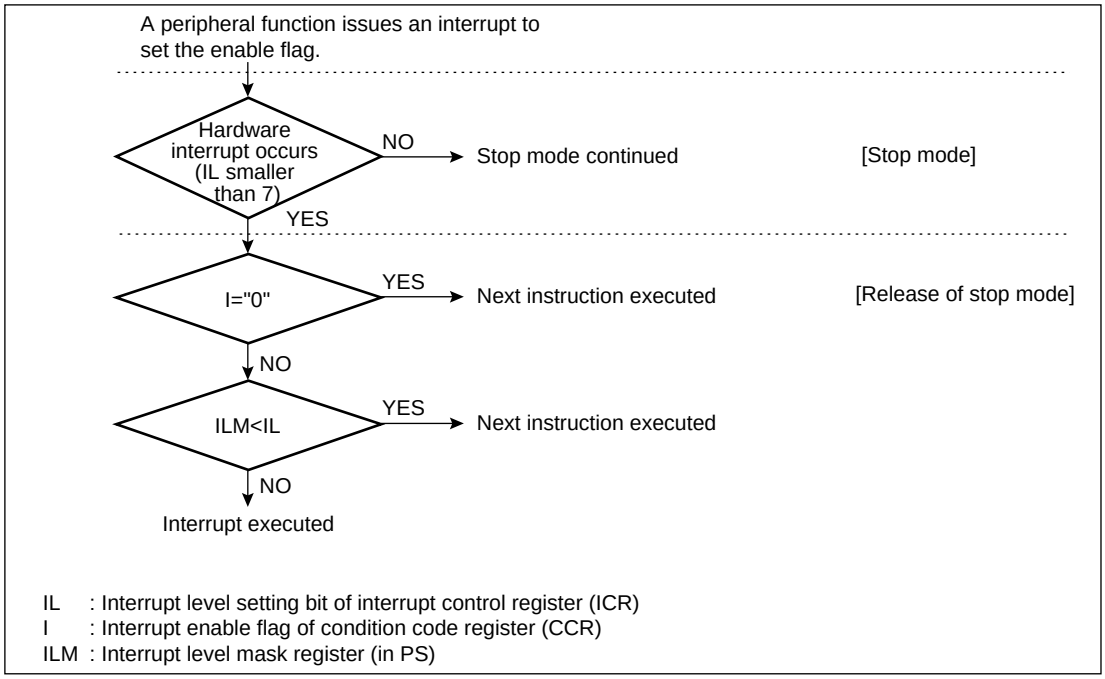
❍ Release by an external reset

For more information, see Section 3.4 "Reset Operation".

❍ Release by a stop interrupt

A stop or external interrupt request with an interrupt level higher than 7 (Interrupt control register ICR: IL2, IL1, IL0 = "000_B" to "110_B") is required to cause an external interrupt to release stop mode.

Figure 5.5-3 Release of Stop Mode by an External Interrupt Output



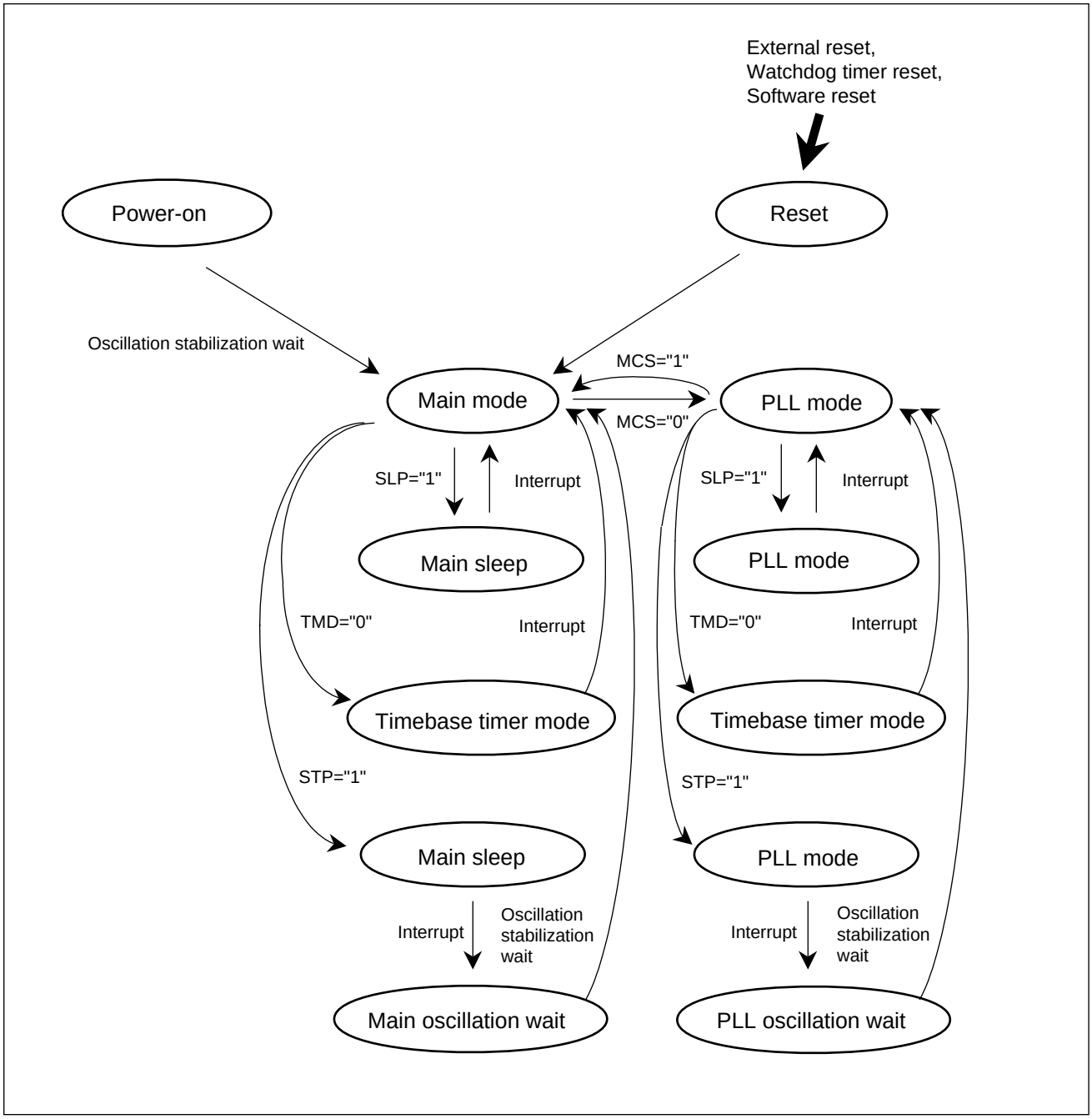
IL : Interrupt level setting bit of interrupt control register (ICR)
I : Interrupt enable flag of condition code register (CCR)
ILM : Interrupt level mask register (in PS)

5.6 Status Change Diagram

This section shows the status change diagram of the CPU operation modes of the MB90560 and 565 series.

■ Status Change Diagram

Figure 5.6-1 Status Change Diagram



5.7 Status of Pins in Standby Mode and during Hold and Reset

This section summarizes the statuses of pins in standby mode and during hold and reset.

■ Software Pull-Up Resistor

For I/O pins configured in software to accept the connection of a pull-up resistor, making an output setting disconnects the pull-up resistor.

■ Status of Pins in Single-Chip Mode

Table 5.7-1 State of Pins in Single-Chip Mode

Pin name	Standby mode			Hold	Reset
	Sleep mode	Stop mode			
		SPL=0	SPL=1		
P00 to P07 P17 P20 to P27 P30 to P37 P40 P41 to P46 P50 to P57 P60 to P63	The preceding status is retained. (*2)	The preceding status is retained. (*2)	Input shut off (*3) /output Hi-z (**)	Output Hi-z (*4)	Output Hi-z (*4)
P10 to P16		Input enabled (*1)			

*1 "Input enabled" means that the input function is enabled. However, the input function is enabled only if an external interrupt is enabled. These pins, when used as output ports, conform to the setting of the pin status setting bit (SPL) of the low power consumption mode control register (LPMCR).
*2 "The preceding status is retained" means that the output status of the pin immediately before switching to standby mode is retained (*5). However, note that the input is disabled (*6) if the pin was in the input status.
*3 "Input shut off" means that the input to the pin is inhibited.
*4 "Output Hi-Z" means that the pin-driving transistor is disabled and that the pin is made to have high impedance.
*5 "the output status is retained as is" means that the output value of a peripheral function (resource) or a port is retained.
*6 "the input is disabled" means that a value input to the pin cannot be accepted internally because an internal circuit is not running.

5.8 Usage Notes on Low Power Consumption Mode

Note the following items when using low power consumption mode:

- **Switching to standby mode and interrupts**
- **Release of standby mode by an interrupt**
- **Release of stop mode by an external interrupt**
- **Oscillation stabilization wait interval**

■ Switching to Standby Mode and Interrupts

While an interrupt request is output, the microcontroller does not enter standby mode even if the stop mode bit (STP) of the low power consumption mode control register (LPMCR) is set to "1", the sleep mode bit (SLP) is set to "1", or the timebase timer mode bit (TMD) is set to "0".

■ Release of Standby Mode by an Interrupt

Standby mode is released if an interrupt request with an interrupt level higher than 7 (Interrupt control register ICR: IL2, IL1, IL0 = "000_B" to "110_B") is output in sleep mode, timebase timer mode, or stop mode.

If the interrupt level setting bit (ICR: IL2, IL1, IL0) corresponding to an interrupt request has a priority higher than the interrupt level mask register (ILM) and the interrupt enable flag of the condition code register is enabled (CCR:I = 1), the interrupt is accepted and the interrupt processing routine is executed. Unless the interrupt is accepted, the processing starts again from the next instruction to the one that set standby mode.

Note:

Interrupt disable setting is required before the setting of standby mode unless an interrupt processing routine is executed immediately after standby mode is released.

■ Release of Stop Mode by an External interrupt

To release stop mode by an external interrupt, set the DTP/interrupt enable register (ENIR) and the request level setting register (ELVR) before the microcontroller enters stop mode.

Select one of the "H" level, "L" level, rising edge, and falling edge as an input cause.

■ Oscillation Stabilization Wait Interval

❍ Source clock oscillation stabilization wait interval

An oscillation stabilization wait interval is required after stop mode is released because the source oscillation has been halted in stop mode. The oscillation stabilization wait interval can be set in the oscillation stabilization wait interval setting bits (WS1, WS0) of the clock selection register (CKSCR).

❍ PLL clock oscillation stabilization wait interval

A PLL clock oscillation stabilization wait interval is required after the operating clock is changed from the main clock to the PLL clock because the PLL clock is halted while the CPU operates on the main clock. While waiting for PLL clock oscillation stabilization, the CPU operates on the main clock.

The PLL clock oscillation stabilization wait interval is fixed at 2^{14}/HCLK (HCLK: clock oscillation frequency).

CHAPTER 6 INTERRUPTS

This chapter explains the interrupts and extended intelligent I/O service (EI^{2}OS) in the MB905605/565 series.

6.1 Interrupts

The MB90560 and 565 series have three interrupt functions and an exception processing function:

- **Hardware interrupts**
- **Software interrupts**
- **Interrupts from extended intelligent I/O service (EI^{2}OS)**
- **Exception processing**

■ Interrupt Types and Functions

❍ Hardware interrupt

A hardware interrupt transfers control to an interrupt processing program in response to an interrupt request from a peripheral function (resource). For more information, see Section 6.4 "Hardware Interrupts".

❍ Software interrupt

A software interrupt transfers control to an interrupt processing program if a software interrupt instruction (INT instruction) is executed on a program. For more information, see Section 6.5 "Software Interrupts".

❍ Interrupt from extended intelligent I/O service (EI^{2}OS)

The extended intelligent I/O service (EI^{2}OS) can transfer data between a register contained in a peripheral function (resource) and internal memory if settings are made in the interrupt control registers (ICR00 to ICR15) and the extended intelligent I/O service descriptor (ISD).

When the data transfers have been terminated, the interrupt processing program is executed. For more information, see Section 6.6 "Interrupt of Extended Intelligent I/O Service (EI^{2}OS)".

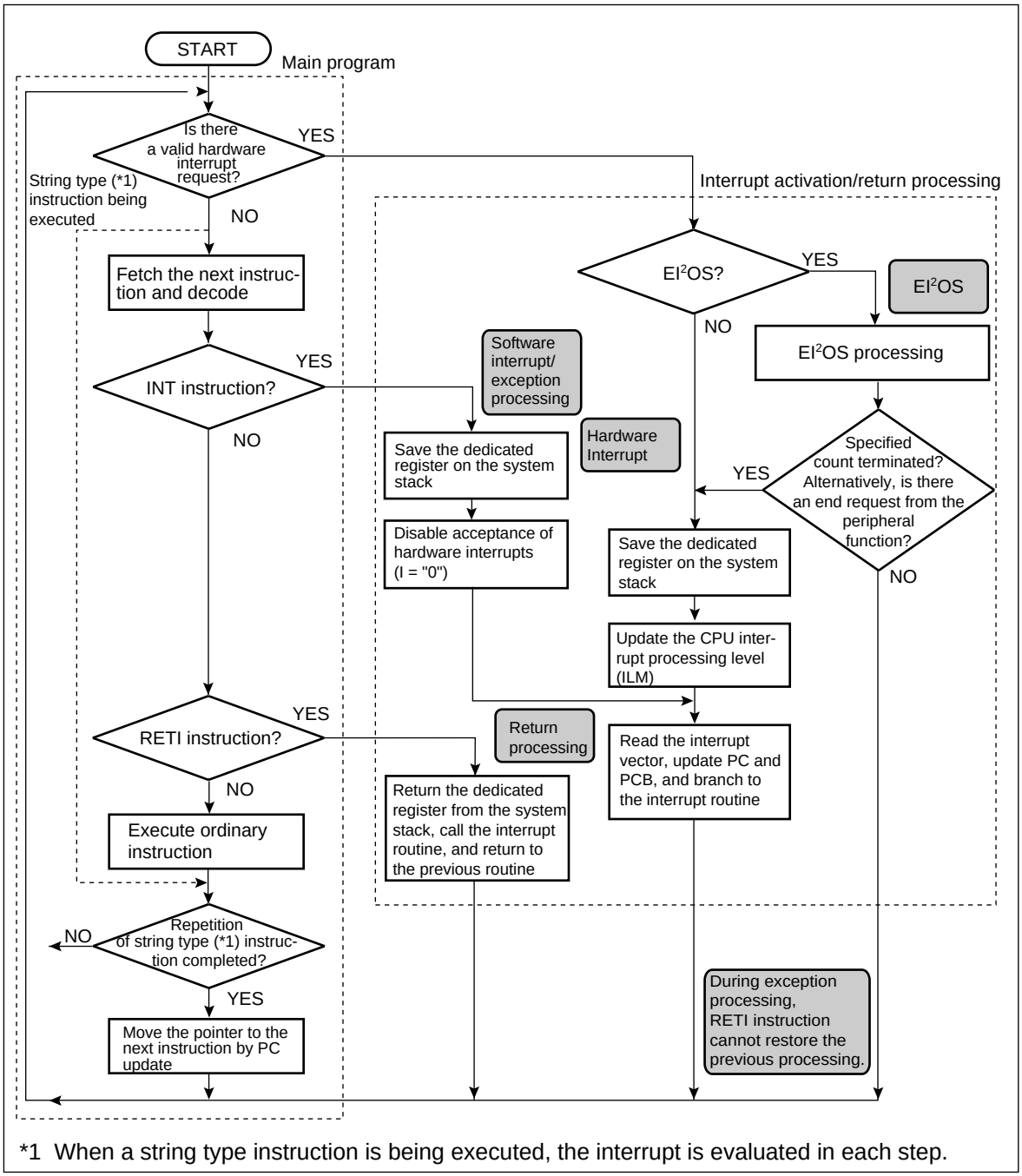
❍ Exception processing

Exception processing is performed if an undefined instruction code is executed.

If exception processing is performed, the register value currently processed is saved to the system stack and the processing branches to the exception processing routine. For more information, see Section 6.7 "Exception Processing Interrupt".

■ Interrupt Operation

Figure 6.1-1 Overall Flow of Interrupt Operation



*1 When a string type instruction is being executed, the interrupt is evaluated in each step.

6.2 Interrupt Causes and Interrupt Vectors

TThe MB90560 and 565 series have functions for handling 256 types of interrupt cause. The 256 interrupt vector tables are allocated to the memory at the highest addresses.
Software interrupts can use 256 interrupt instructions (INT0 to INT255). Note that INT8 is shared with a reset vector interrupt and that INT10 is shared with exception processing. INT11 to INT 42 are shared with an interrupt from a peripheral function (resource).

■ Interrupt Vectors

Interrupt vector tables referenced during interrupt processing are allocated to the highest addresses in the memory area ($FFFC00_H$ to $FFFFFF_H$). Interrupt vectors share the same area with EI^2OS, exception processing, hardware, and software interrupts.

Table 6.2-1 "Interrupt Vectors" shows the assignment of software interrupt instructions, interrupt numbers, and interrupt vectors.

Table 6.2-1 Interrupt Vectors

Software interrupt instruction	Vector address L	Vector address M	Vector address H	Mode data	Interrupt No.	Hardware interrupt
INT0	$FFFFFC_H$	$FFFFFD_H$	$FFFFFE_H$	Not used	#0	None
:	:	:	:	:	:	:
INT7	$FFFFE0_H$	$FFFFE1_H$	$FFFFE2_H$	Not used	#7	None
INT8	$FFFFDC_H$	$FFFFDD_H$	$FFFFDE_H$	$FFFFDF_H$	#8	(RESET Vector)
INT9	$FFFFD8_H$	$FFFFD9_H$	$FFFFDA_H$	Not used	#9	None
INT10	$FFFFD4_H$	$FFFFD5_H$	$FFFFD6_H$	Not used	#10	<Exception processing>
INT11	$FFFFD0_H$	$FFFFD1_H$	$FFFFD2_H$	Not used	#11	Hardware interrupt #0
INT12	$FFFFCC_H$	$FFFFCD_H$	$FFFFCE_H$	Not used	#12	Hardware interrupt #1
INT13	$FFFFC8_H$	$FFFFC9_H$	$FFFFCA_H$	Not used	#13	Hardware interrupt #2
INT14	$FFFFC4_H$	$FFFFC5_H$	$FFFFC6_H$	Not used	#14	Hardware interrupt #3
:	:	:	:	:	:	:
INT254	$FFFC04_H$	$FFFC05_H$	$FFFC06_H$	Not used	#254	None
INT255	$FFFC00_H$	$FFFC01_H$	$FFFC02_H$	Not used	#255	None

Reference:

Interrupt vectors not defined during software design should be set at the exception processing address.

■ Interrupt Causes and Interrupt Vectors/Interrupt Control Registers

Table 6.2-2 Hardware Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers

Interrupt cause	EI^{2}OS support	Interrupt vector			Interrupt control register		Priority
		Number *1		Address	ICR	Address	
Reset	X	#08	08_H	$FFFFDC_H$	-	-	High ↓ Low
INT9 instruction	X	#09	09_H	$FFFFD8_H$	-	-	
Exception processing	X	#10	$0A_H$	$FFFFD4_H$	-	-	
A/D converter conversion termination	O	#11	$0B_H$	$FFFFD0_H$	ICR00	$0000B0_H$	
Output compare channel 0 match	△	#13	$0D_H$	$FFFFC8_H$	ICR01	$0000B1_H$	
8/16-bit PPG timer 0 counter borrow	△	#14	$0E_H$	$FFFFC4_H$			
Output compare channel 1 match	△	#15	$0F_H$	$FFFFC0_H$	ICR02	$0000B2_H$	
8/16-bit PPG timer 1 counter borrow	△	#16	10_H	$FFFFBC_H$			
Output compare channel 2 match	△	#17	11_H	$FFFFB8_H$	ICR03	$0000B3_H$	
8/16-bit PPG timer 2 counter borrow	△	#18	12_H	$FFFFB4_H$			
Output compare channel 3 match	△	#19	13_H	$FFFFB0_H$	ICR04	$0000B4_H$	
8/16-bit PPG timer 3 counter borrow	△	#20	14_H	$FFFFAC_H$			
Output compare channel 4 match	△	#21	15_H	$FFFFA8_H$	ICR05	$0000B5_H$	
8/16-bit PPG timer 4 counter borrow	△	#22	16_H	$FFFFA4_H$			
Output compare channel 5 match	△	#23	17_H	$FFFFA0_H$	ICR06	$0000B6_H$	
8/16-bit PPG timer 5 counter borrow	△	#24	18_H	$FFFF9C_H$			
DTP/external interrupt channels 0/1 detection	△	#25	19_H	$FFFF98_H$	ICR07	$0000B7_H$	
DTP/external interrupt channels 2/3 detection	△	#26	$1A_H$	$FFFF94_H$			
DTP/external interrupt channels 4/5 detection	△	#27	$1B_H$	$FFFF90_H$	ICR08	$0000B8_H$	
DTP/external interrupt channels 6/7 detection	△	#28	$1C_H$	$FFFF8C_H$			
8-bit timer 0/1/2 counter borrow	X	#29	$1D_H$	$FFFF88_H$	ICR09	$0000B9_H$	
16-bit reload timer 0 underflow	O	#30	$1E_H$	$FFFF84_H$			
16-bit free-running timer overflow	X	#31	$1F_H$	$FFFF80_H$	ICR10	$0000BA_H$	
16-bit reload timer 1 underflow	O	#32	20_H	$FFFF7C_H$			
Input capture channels 0/1	O	#33	21_H	$FFFF78_H$	ICR11	$0000BB_H$	
16-bit free-running timer clear	X	#34	22_H	$FFFF74_H$			
Input capture channels 2/3	O	#35	23_H	$FFFF70_H$	ICR12	$0000BC_H$	
Timebase timer	X	#36	24_H	$FFFF6C_H$			

Table 6.2-2 Hardware Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers (Continued)

Interrupt cause	EI^{2}OS support	Interrupt vector			Interrupt control register		Priority
		Number *1		Address	ICR	Address	
UART1 receive	◎	#37	25_H	$FFFF68_H$	ICR13	$0000BD_H$	High ↕ Low
UART1 send	△	#38	26_H	$FFFF64_H$			
UART0 receive	◎	#39	27_H	$FFFF60_H$	ICR14	$0000BE_H$	
UART0 send	△	#40	28_H	$FFFF5C_H$			
Flash memory status	X	#41	29_H	$FFFF58_H$	ICR15	$0000BF_H$	
Delayed interrupt generator module	X	#42	$2A_H$	$FFFF54_H$			

O: Can be used.
X: Cannot be used.
◎ : Usable if used with the EI^{2}OS stop function.
△ : Usable when an interrupt cause that shares the ICR is not used.
*1: If multiple interrupts of the same level are output simultaneously, an interrupt cause with a smaller interrupt vector number takes precedence.

■ Interrupt Control Register (ICR) and Interrupt Cause

Figure 6.2-1 Relationship between Interrupt Control Register (ICR) and Interrupt Cause

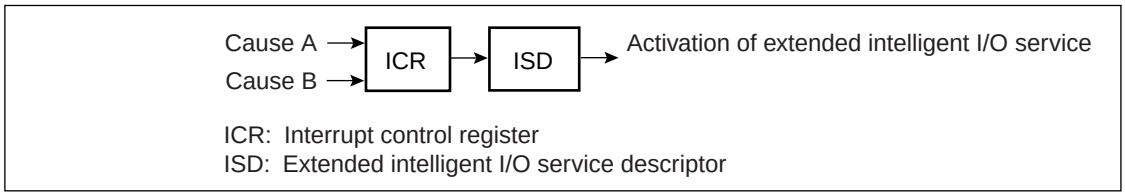


Figure 6.2-1 "Relationship between Interrupt Control Register (ICR) and Interrupt Cause" shows that, since an interrupt level is set in each interrupt control register (ICR), Causes A and B connected to the same ICR have the same interrupt level.

If multiple interrupt causes of the same interrupt level are output, the one with a smaller vector number takes precedence.

To activate the extended intelligent I/O service due to Cause A, disable the interrupt request output due to Cause B in a peripheral function (resource).

When the interrupt request output due to Cause B is enabled in a peripheral function (resource), the extended intelligent I/O service is activated even if an interrupt request due to Cause B is output.

6.3 Interrupt Control Registers (ICR) and Peripheral Functions (Resource)

The interrupt control registers ICR00 to ICR15 correspond to all peripheral functions that have the interrupt function. These registers control interrupts and the extended intelligent I/O service (EI^{2}OS).

■ Interrupt Control Registers

Table 6.3-1 Interrupt Control Registers

Address	Register	Abbreviation	Corresponding peripheral function (Resource)
$0000B0_H$	Interrupt control register 00	ICR00	A/D converter
$0000B1_H$	Interrupt control register 01	ICR01	Output compare 0, 8/16-bit PPG timer 0
$0000B2_H$	Interrupt control register 02	ICR02	Output compare 1, 8/16-bit PPG timer 1
$0000B3_H$	Interrupt control register 03	ICR03	Output compare 2, 8/16-bit PPG timer 2
$0000B4_H$	Interrupt control register 04	ICR04	Output compare 3, 8/16-bit PPG timer 3
$0000B5_H$	Interrupt control register 05	ICR05	Output compare 4, 8/16-bit PPG timer 4
$0000B6_H$	Interrupt control register 06	ICR06	Output compare 5, 8/16-bit PPG timer 5
$0000B7_H$	Interrupt control register 07	ICR07	DTP/external interrupts 0, 1, 2, 3
$0000B8_H$	Interrupt control register 08	ICR08	DTP/external interrupts 4, 5, 6, 7
$0000B9_H$	Interrupt control register 09	ICR09	8-bit timers 0, 1, 2, 16-bit reload timer 0
$0000BA_H$	Interrupt control register 10	ICR10	16-bit free-running timer overflow, 16-bit reload timer 1
$0000BB_H$	Interrupt control register 11	ICR11	Input capture 0, 1, 16-bit free-running timer clear
$0000BC_H$	Interrupt control register 12	ICR12	Input capture 2, 3, timebase timer
$0000BD_H$	Interrupt control register 13	ICR13	UART1
$0000BE_H$	Interrupt control register 14	ICR14	UART0
$0000BF_H$	Interrupt control register 15	ICR15	Flash memory, delayed interrupt generator module

■ Interrupt Control Register Functions

The following four settings can be made in an interrupt control register (ICR).

- Set the interrupt level of the corresponding peripheral function (resource).
- Set an interrupt of the peripheral function (resource) either as an interrupt or as the extended intelligent I/O service.
- Set the descriptor address of the extended intelligent I/O service (EI^{2}OS).

- Display the status of the extended intelligent I/O service (EI^2OS)

Interrupt control registers (ICRs) have different functions during the writing and reading of data.

Note:

When interrupt control registers (ICRs) are set, a read-modify-write instruction of SETB and CLRB cannot be used to access it.

6.3.1 Interrupt Control Registers (ICR00 to ICR15)

Interrupt control registers can determine the interrupt processing or the extended intelligent I/O service processing when an interrupt request is output. Interrupt control registers (ICRs) have different functions during the writing and reading of data.

■ Interrupt Control Registers (ICR00 to ICR15)

Figure 6.3-1 Interrupt Control Registers (ICR00 to ICR15) during Writing

Writing (setting the interrupt control registers)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000B0H to 0000BFH	ICS3	ICS2	ICS1	ICS0	ISE	IL2	IL1	IL0	XXXX0111B
	W	W	W	W	R/W	R/W	R/W	R/W	

IL2	IL1	IL0	Interrupt level setting bit
0	0	0	Interrupt level 0 (highest)
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	Interrupt level 7 (no interrupt)

ISE	EI^{2}OS enable bit
0	Activates the interrupt sequence when an interrupt occurs
1	Activates EI^{2}OS when an interrupt occurs

ICS3	ICS2	ICS1	ICS0	EI^{2}OS descriptor address setting bit
0	0	0	0	000100H
0	0	0	1	000108H
0	0	1	0	000110H
0	0	1	1	000118H
0	1	0	0	000120H
0	1	0	1	000128H
0	1	1	0	000130H
0	1	1	1	000138H
1	0	0	0	000140H
1	0	0	1	000148H
1	0	1	0	000150H
1	0	1	1	000158H
1	1	0	0	000160H
1	1	0	1	000168H
1	1	1	0	000170H
1	1	1	1	000178H

R/W : Read/write
W : Write only
* : No interrupt processing on interrupt level 7
: Initial value

Figure 6.3-2 Interrupt Control Registers (ICR00 to ICR15) during Reading

Reading

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000B0H to 0000BFH	—	—	S1	S0	ISE	IL2	IL1	IL0	XX000111B
	-	-	R	R	R/W	R/W	R/W	R/W	

IL2	IL1	IL0	Interrupt level setting bit
0	0	0	Interrupt level 0 (highest)
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	Interrupt level 7 (no interrupt)

ISE	EI^{2}OS enable bit
0	Activates the interrupt sequence when an interrupt occurs
1	Activates EI^{2}OS when an interrupt occurs

S1	S0	EI^{2}OS status
0	0	EI^{2}OS operation in progress or EI2OS not activated
0	1	Stopped status due to count termination
1	0	Reserved
1	1	Stopped status due to a request from the peripheral function

R/W : Read/write
R : Read only
X : Undefined
- : Undefined bit
* : No interrupt processing on interrupt level 7
: Initial value

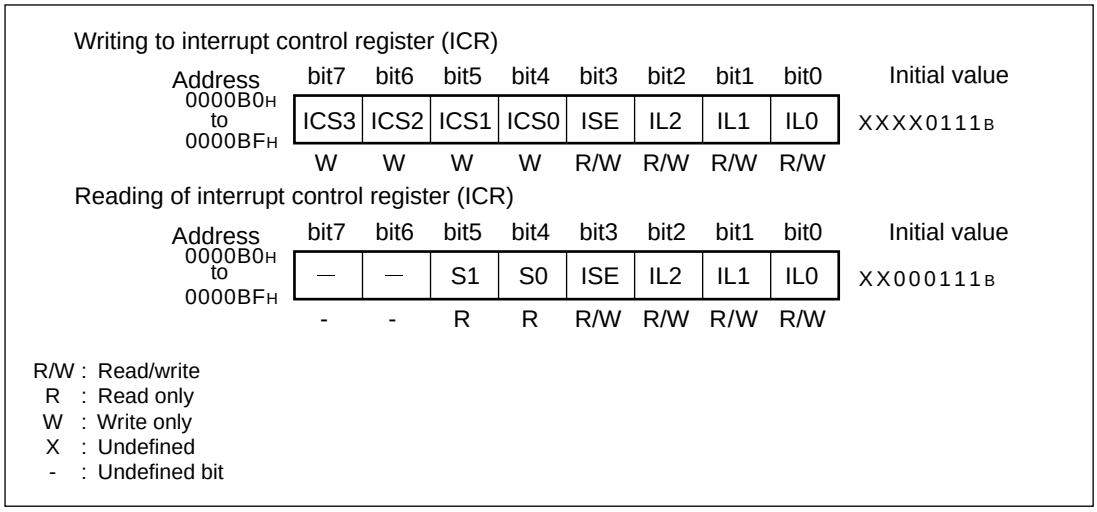
6.3.2 Interrupt Control Register (ICR) Functions

The interrupt control registers (ICR00 to ICR15) can specify the following settings:

- **Interrupt level setting**
- **Extended intelligent I/O service (EI^{2}OS) enable setting**
- **Extended intelligent I/O service (EI^{2}OS) descriptor address setting**
- **Extended intelligent I/O service (EI^{2}OS) operation status display**

■ Configuration of Interrupt Control Registers (ICR)

Figure 6.3-3 Configuration of Interrupt Control Registers (ICR)



Reference:

The settings in the EI^{2}OS descriptor address setting bits (ICS3 to ICS0) are valid when the extended intelligent I/O service (EI^{2}OS) is activated. Set the EI^{2}OS enable bit (ISE) to "1" to activate it. Alternatively, set the ISE to "0" to refrain from activating it. If EI^{2}OS is not activated, the ICS3 to ICS0 bits need not be set.

■ Interrupt Control Register Functions

❍ Interrupt level setting

These bits can set the interrupt level of the corresponding peripheral function (resource). A reset initializes these bits to level 7.

Table 6.3-2 Correspondence between the Interrupt Level Setting Bits and Interrupt Levels

IL2	IL1	IL0	Interrupt level
0	0	0	0 (highest priority) ↕ 6 (lowest priority)
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	7 (no interrupts)

❍ Extended intelligent I/O service (EI^{2}OS) enable setting

When an interrupt request is output, EI^{2}OS is activated if the EI^{2}OS enable bit (ISE) is set to "1". Alternatively, an interrupt sequence is activated if the ISE bit is set to "0". When the EI^{2}OS processing is completed, the ISE bit is reset to "0". If a peripheral function (resource) has no EI^{2}OS function, set the ISE bit to "0" using software. A reset initializes the ISE bit to "0".

❍ Extended intelligent I/O service (EI^{2}OS) descriptor address setting

The EI^{2}OS descriptor address setting bits (ICS3 to ICS0) are valid during writing. Set the EI^{2}OS descriptor address in these bits. Set values in the ICS3 to ICS0 bits to set the EI^{2}OS descriptor address. A reset initializes the ICS3 to ICS0 bits to "0000_B".

Table 6.3-3 Correspondence between the EI^{2}OS Descriptor Address Setting Bits and Descriptor Addresses

ICS3	ICS2	ICS1	ICS0	Descriptor address
0	0	0	0	000100_H
0	0	0	1	000108_H
0	0	1	0	000110_H
0	0	1	1	000118_H
0	1	0	0	000120_H
0	1	0	1	000128_H
0	1	1	0	000130_H
0	1	1	1	000138_H
1	0	0	0	000140_H
1	0	0	1	000148^H
1	0	1	0	000150_H
1	0	1	1	000158_H
1	1	0	0	000160_H
1	1	0	1	000168_H
1	1	1	0	000170_H
1	1	1	1	000178_H

❍ **Extended intelligent I/O service (EI^{2}OS) operation status display**

The EI^{2}OS status bits (S1 and S0) are valid during reading. Read the S1 and S0 bits while EI^{2}OS is activated to determine whether EI^{2}OS is running or terminated. A reset initializes the S1 and S0 bits to "00_B".

Table 6.3-4 Relationship between EI^{2}OS Status Bits and the EI^{2}OS Status

S1	S0	EI^{2}OS status
0	0	EI^{2}OS operation in progress or EI^{2}OS not activated
0	1	Stopped status due to count termination
1	0	Reserved
1	1	Stopped status due to a request from the peripheral function (resource)

6.4 Hardware Interrupts

A hardware interrupt operates as follows: an interrupt request that is output by a peripheral function (resource) temporarily interrupts a program being executed by the CPU and transfers control to a user-defined interrupt processing program. The extended intelligent I/O service (EI^{2}OS) is also handled as a hardware interrupt.

■ Hardware Interrupts

❍ Hardware interrupt function

The hardware interrupt function determines whether an interrupt can be accepted. To do so, it compares the interrupt level of an interrupt request that is output by a peripheral function (resource) with the interrupt level mask register (PS: ILM) while referring to the contents of the I flag (PS: I).

If a hardware interrupt is accepted, the contents of the direct page register (DPR), accumulator (A), program counter (PC), processor status register (PS), and bank registers (ADB, DTB, and PCB) are saved to the system stack. An interrupt level stored in the ICR register is then stored in the interrupt level mask register (ILM). Finally, the processing branches to the interrupt vector and the interrupt processing program is executed.

❍ Multiple interrupts

A hardware interrupt can be activated while the interrupt processing program is being executed.

❍ Extended intelligent I/O service (EI^{2}OS)

EI^{2}OS is a data transfer function between memory and I/O registers. When the transfer of data to the extended intelligent I/O service descriptor is completed, a hardware interrupt is activated. EI^{2}OS cannot be activated in duplicate. While EI^{2}OS is being processed, no interrupt request or EI^{2}OS request is accepted. When the processing of EI^{2}OS is completed, interrupt request or EI^{2}OS request is accepted.

❍ External interrupt

An external interrupt is accepted as a hardware interrupt if a circuit that can output an interrupt request from an external terminal (DTP/external interrupt circuit) detects an interrupt request.

❍ Interrupt vector

Interrupt vector tables referenced during interrupt processing are allocated to memory at $FFFC00_H$ to $FFFFFF_H$.

See Section 6.2 "Interrupt causes and Interrupt Vectors", for more information about the allocation of interrupt numbers and interrupt vectors.

■ Hardware Interrupt Structure

The four mechanisms (seven locations) shown in Table 6.4-1 "Mechanisms Used for Hardware Interrupts" are used for hardware interrupts. These four mechanisms (seven locations) must be configured in a user program before hardware interrupts can be used.

Table 6.4-1 Mechanisms Used for Hardware Interrupts

	Mechanism	Function
Peripheral function	Interrupt enable bit, interrupt request bit	Controls interrupt requests from a peripheral function (resource)
Interrupt controller	Interrupt control register (ICR)	Sets the interrupt level and controls EI^2OS
CPU	Interrupt enable flag (I)	Identifies the interrupt enable status
	Interrupt level mask register (ILM)	Compares the request interrupt level and current interrupt level
	Microcode	Executes the interrupt processing routine
$FFFC00_H$ to $FFFFFF_H$ in memory	Interrupt vector table	Stores the branch destination address for interrupt processing

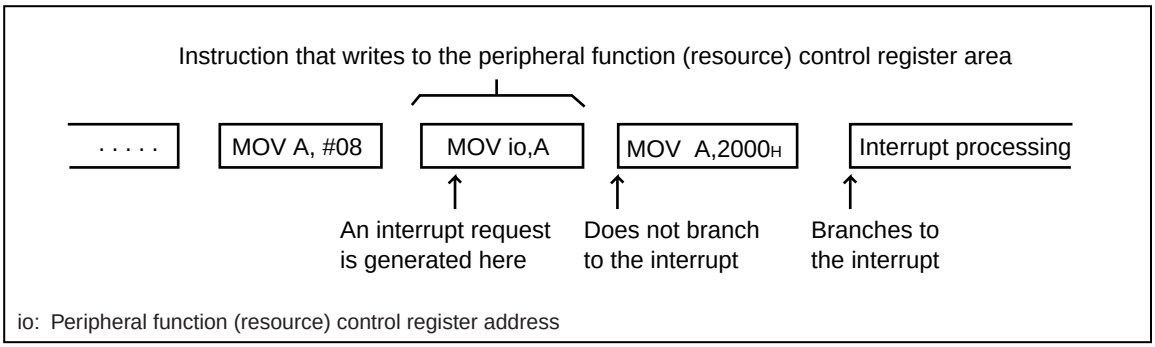
■ Hardware Interrupt Disable

Acceptance of a hardware interrupt request is disabled under the following conditions:

❍ **Hardware interrupt acceptance disable during writing to the peripheral function (resource) control register**

No hardware interrupt request is accepted while data is written to the peripheral function (resource) control register.

Figure 6.4-1 Hardware Interrupt Request While Writing to the Peripheral Function (Resource) Control Register Area



❍ **Hardware interrupt acceptance disable by interrupt suppression instructions**

The ten types of hardware interrupt suppression instructions listed in Table 6.4-2 "Hardware Interrupt Suppression Instruction" ignore interrupt requests without detecting whether a hardware interrupt request exists.

If a hardware interrupt request is output while a hardware interrupt suppression instruction is being executed, the interrupt is accepted and processed after completion of the hardware interrupt suppression instruction and the subsequent execution of an instruction other than the hardware interrupt suppression instruction.

Table 6.4-2 Hardware Interrupt Suppression Instruction

	Prefix code	Interrupts/hold suppression instructions (instructions that delay the effect of the prefix code)	
Instructions that do not accept interrupts and hold requests	PCB DTB ADB SPB CMR NCC	MOV OR AND POPW	ILM, #imm8 CCR, #imm8 CCR, #imm8 PS

❍ **Hardware interrupt acceptance disable during execution of a software interrupt**

When a software interrupt is activated, the I flag is cleared to 0. In this state, other interrupt requests cannot be accepted.

6.4.1 Operation of Hardware Interrupts

This section explains hardware interrupt operation from generation of a interrupt request to the completion of interrupt processing.

■ Hardware Interrupt Activation

❍ Peripheral function (resource) operation (generation of an interrupt request)

A peripheral function (resource) that has a hardware interrupt request function has an "interrupt request flag bit" and an "interrupt enable flag" in the corresponding peripheral function (resource) control registers. The interrupt request flag bit indicates the presence of an interrupt request. The interrupt enable flag determines whether a CPU interrupt request is enabled or disabled. When an interrupt cause defined in a peripheral function is detected, an interrupt request is output to an interrupt controller as long as the interrupt request flag bit is set to "1" and the interrupt enable bit is set to enable an interrupt request to the CPU.

❍ Interrupt controller operation (interrupt request control)

The interrupt controller compares the interrupt level (IL) to set a request having the highest level. If multiple interrupts of the same level are output simultaneously, an interrupt request with the smallest number takes precedence (see Table 6.2-1 "Interrupt Vectors").

❍ CPU operation (interrupt request acceptance and interrupt processing)

The CPU compares the received interrupt level (ICR: IL2 to IL0) and the interrupt level mask register (ILM). If IL2 to IL0 are greater than ILM and interrupts are enabled (PS: CCR: I = "1"), the CPU terminates the instruction being executed and performs the interrupt processing. If the EI^2OS enable bit (ISE) of the interrupt control register (ICR) is set to "0", the CPU performs the interrupt processing. If the ISE is set to "1", the CPU activates EI^2OS.

Interrupt processing saves the contents of the dedicated registers (12 bytes including A, DPR, ADB, DTB, PCB, PC, and PS) on the system stack (the system stack space indicated by the SSB and SSP).

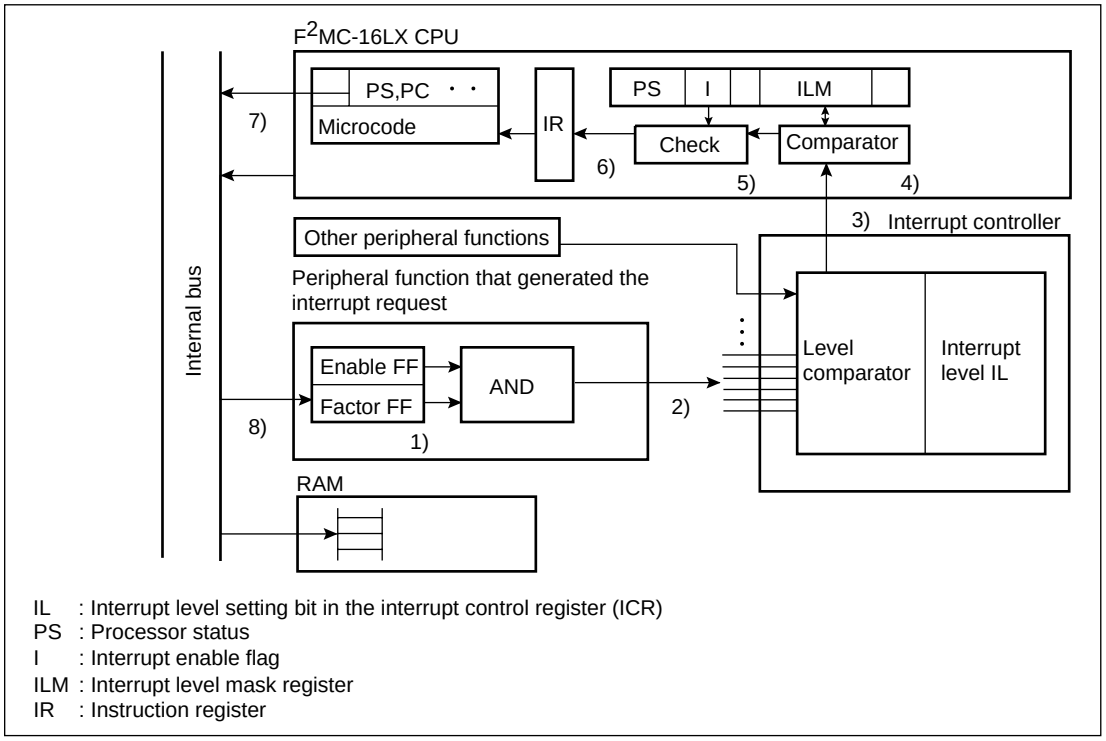
The CPU then loads data into the interrupt vector program counters (PCB and PC), updates the ILM, and sets the stack flag (S) (sets CCR: S = 1 and activates the system stack).

■ Returning from a Hardware Interrupt

If an interrupt processing program writes "0" to the interrupt request flag bit of a peripheral function (resource) that output an interrupt cause and the RETI instruction is executed, data saved on the system stack is restored to the dedicated registers and the program processing that was executed before branching due to an interrupt is resumed.

■ Hardware Interrupt Operation

Figure 6.4-2 Hardware Interrupt Operation



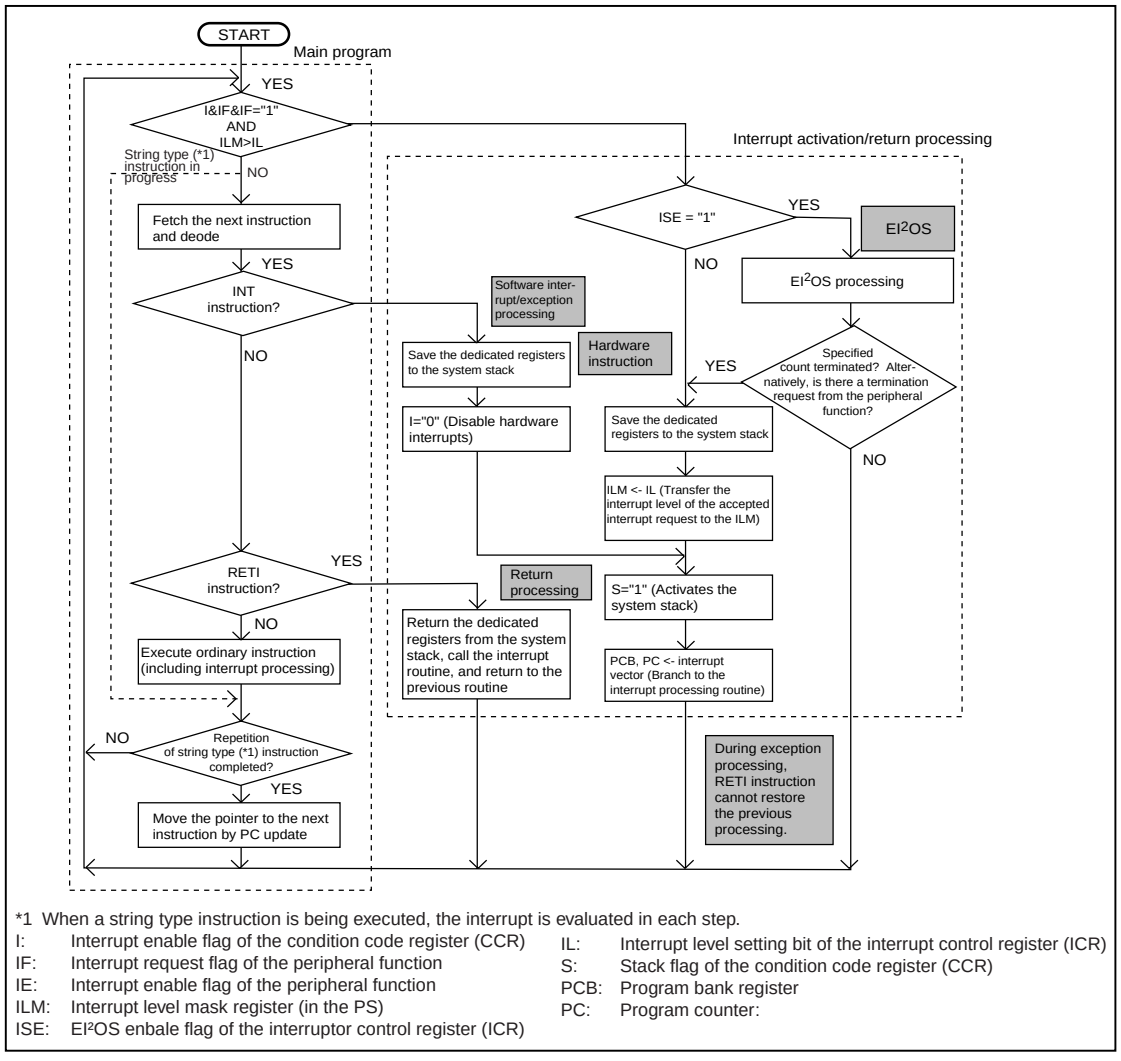
1. An interrupt cause is output within the peripheral functions (resources).
2. If the interrupt enable bit in the peripheral functions (resources) is set to enable interrupts, interrupt requests are output from the peripheral functions (resources) to the interrupt controller.
3. The interrupt controller that receives interrupt requests from the peripheral functions (resources) checks the priorities of interrupt requests simultaneously received and transfers the interrupt level (IL) of an interrupt request with the highest priority to the CPU.
4. The CPU compares the interrupt level (IL) requested by the interrupt controller with the interrupt level mask register (ILM).
5. If the comparison indicates a higher priority than the current interrupt processing level, the CPU checks the contents of the I flag in the condition code register (CCR).
6. If, in the check, the I flag in the CCR is found to be set to Enabled (CCR: I = "1"), the CPU waits until the execution of an instruction being executed is terminated. When it is terminated, the CPU sets the requested level (IL2 to IL0) in the ILM.
7. The values in the dedicated registers are saved to the system stack. The processing branches to the interrupt processing routine.
8. If a program in the interrupt processing routine sets the interrupt request flag bit of a peripheral function (resource) to "0" and the RETI instruction is executed, data saved on the system stack is restored to the dedicated registers and the interrupt processing is terminated.

6.4.2 Processing for Interrupt Operation

When a peripheral function (resource) outputs an interrupt request and the CPU accepts it, the interrupt processing is performed after the instruction currently being executed is terminated. If the EI^{2}OS enable bit (ISE) of the interrupt control register (ICR) is set to "0", the CPU performs the interrupt processing. If the ISE is set to "1", the CPU activates the extended intelligent I/O service (EI^{2}OS). If a software interrupt is output by the INT instruction, the instruction currently being executed is suspended, the interrupt processing routine is performed, and hardware interrupts are disabled.

■ Processing for Interrupt Operation

Figure 6.4-3 Flow of Interrupt Processing



*1 When a string type instruction is being executed, the interrupt is evaluated in each step.

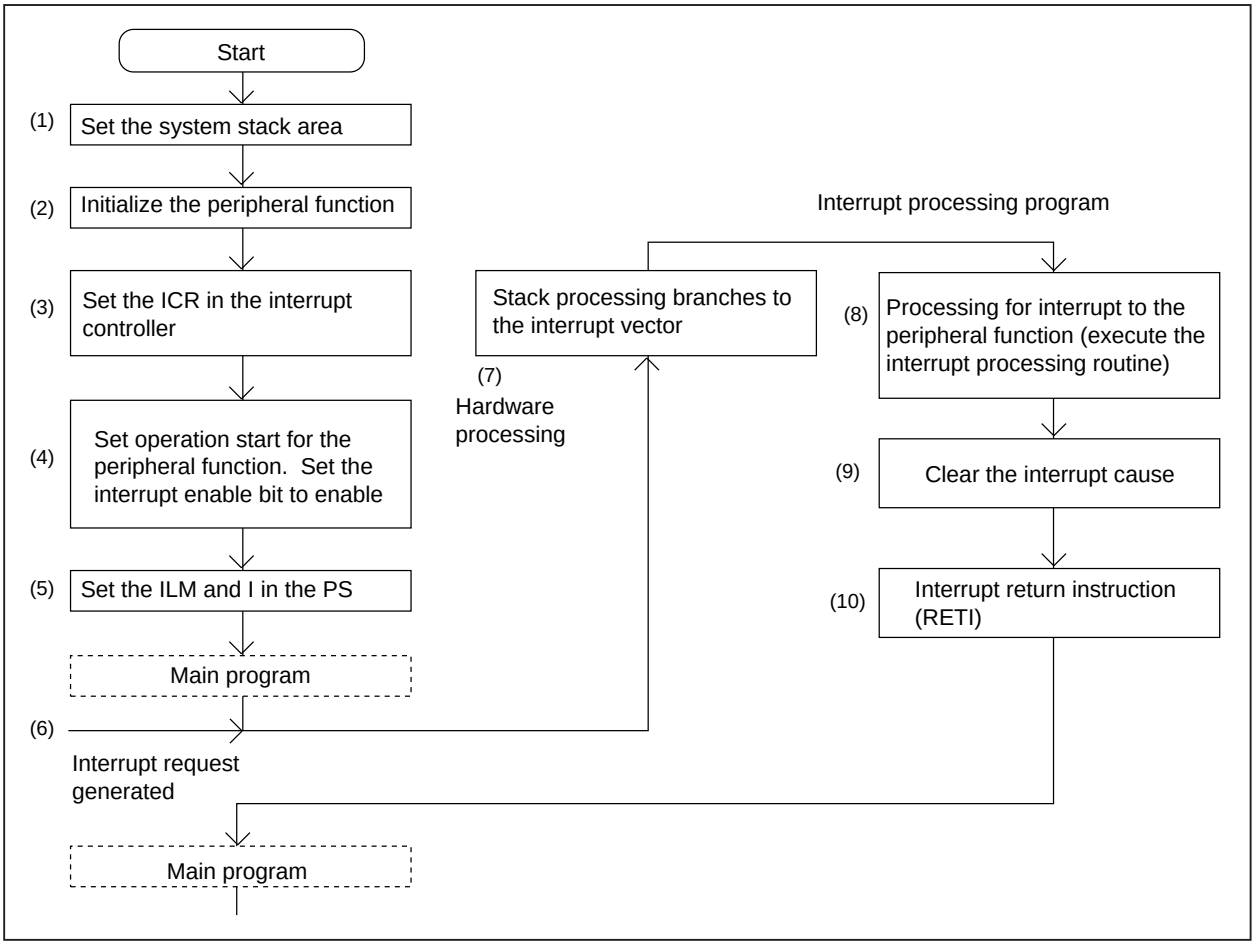
I: Interrupt enable flag of the condition code register (CCR)
IF: Interrupt request flag of the peripheral function
IE: Interrupt enable flag of the peripheral function
ILM: Interrupt level mask register (in the PS)
ISE: EI²OS enbale flag of the interruptor control register (ICR)
IL: Interrupt level setting bit of the interrupt control register (ICR)
S: Stack flag of the condition code register (CCR)
PCB: Program bank register
PC: Program counter:

6.4.3 Procedure for Using Hardware Interrupts

Before hardware interrupts can be used, the system stack area, peripheral function (resource), and interrupt control register (ICR) must be set.

■ Procedure for Using Hardware Interrupts

Figure 6.4-4 Procedure for Using Hardware Interrupts



1. Set the system stack area.
2. Set the operation of a peripheral function (resource).
3. Set the interrupt control register (ICR).
4. Set the interrupt enable bit of the peripheral function (resource) to enable the output of interrupt requests.
5. Set the interrupt level mask register (ILM) and interrupt enable flag (I) to interrupt acceptable.
6. If an interrupt request of the peripheral function (resource) is detected, a hardware interrupt request is output.
7. The interrupt processing hardware saves the dedicated register values to the system stack. The processing then branches to the interrupt processing program.

8. The interrupt processing program processes the peripheral function (resource) in response to the generated interrupt.
9. Clear the peripheral function (resource) interrupt request.
10. Execute the interrupt return instruction (RETI instruction), and return to the program before branching.

6.4.4 Multiple Interrupts

Multiple hardware interrupts can be implemented in response to multiple interrupt requests from peripheral functions (resources). However, multiple extended intelligent I/O services cannot be activated.

■ Multiple Interrupts

❍ Operation of multiple interrupts

If an interrupt request with an interrupt level that is higher than the one being executed is output, the current interrupt processing is suspended and the higher-priority interrupt request is executed. When the processing of the higher-priority interrupt ends, the previous interrupt processing resumes.

If, during execution of interrupt processing, an interrupt request with a level equal to or lower than the current interrupt processing is output, the new interrupt request is suspended until the current interrupt processing ends, unless the I flag of the condition code register (ICR) or the interrupt level mask register (ILM) is changed. When the current interrupt processing ends, the suspended interrupt request is executed.

Other multiple interrupts to be activated during an interrupt can be temporarily disabled by setting the I flag in the condition code register (CCR) in the interrupt processing routine to interrupts not allowed (CCR: I = 0) or the interrupt level mask register (ILM) to interrupts not allowed (ILM = 000_B).

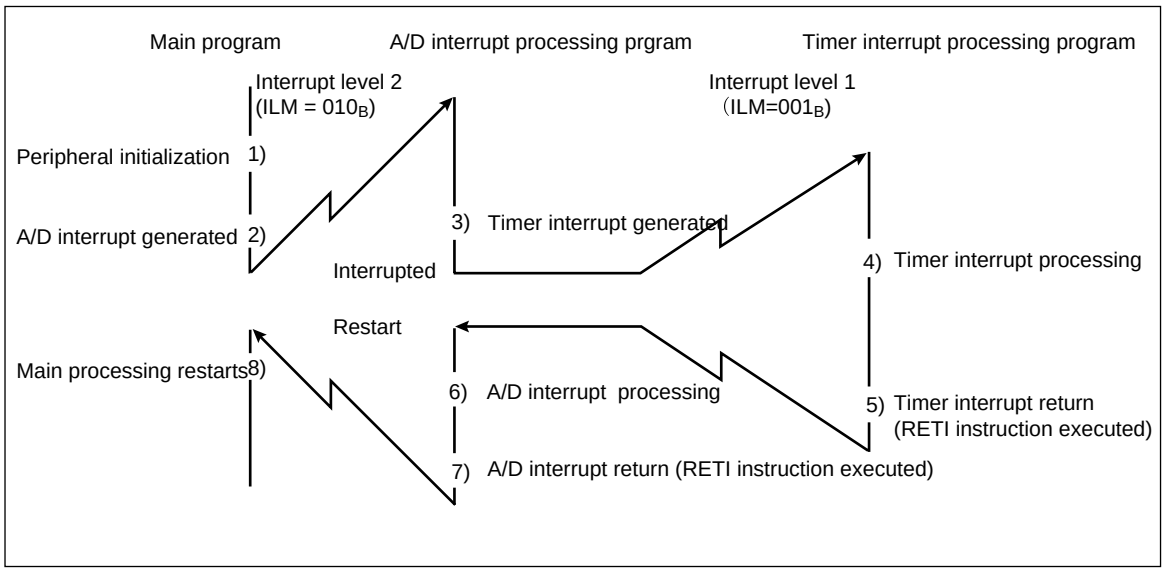
Note:

- 0 to 7 can be set as the interrupt level. If level 7 is set, the CPU does not accept interrupt requests.
- The extended intelligent I/O service (EI^2OS) cannot be used for the activation of multiple interrupts. During processing of the extended intelligent I/O service (EI^2OS), all other interrupt requests and extended intelligent I/O service requests are held.

❍ **Example of multiple interrupts**

This example of multiple interrupt processing assumes that a timer interrupt is given a priority that is higher than an A/D converter interrupt. In this example, the A/D converter interrupt level is set to 2, and the timer interrupt level is set to 1. If a timer interrupt is generated during processing of the A/D converter interrupt, the processing shown in Figure 6.4-5 "Example of Multiple Interrupts" is performed.

Figure 6.4-5 Example of Multiple Interrupt Processing



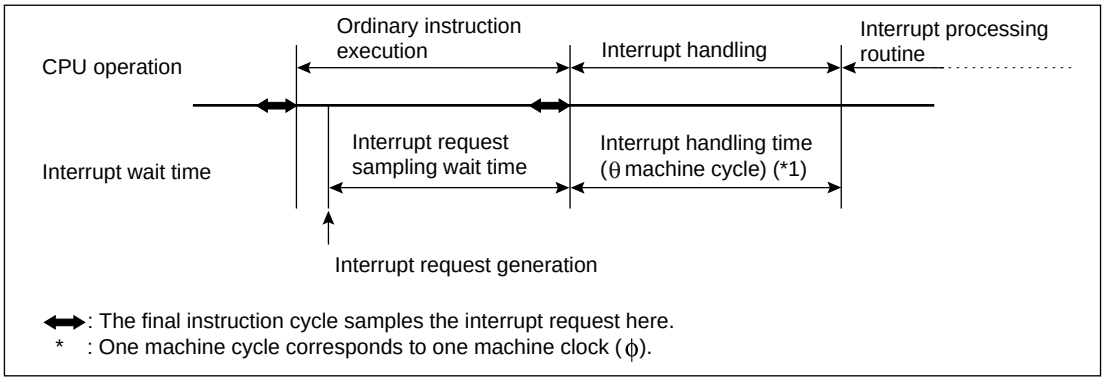
6.4.5 Hardware Interrupt Processing Time

This section describes the processing time required from the output of a hardware interrupt request until the interrupt processing routine is executed.

■ Hardware Interrupt Processing Time

The interrupt request sampling wait time and the interrupt handling time (time required to prepare for interrupt processing) are required from the output of a hardware interrupt request until the interrupt processing routine is executed.

Figure 6.4-6 Interrupt Processing Time



❍ Interrupt request sampling wait time

The interrupt request sampling wait time refers to the time required from the output of an interrupt request by a peripheral function (resource) until the instruction being executed terminates. The CPU checks through sampling whether an interrupt request is output in the final cycle of an instruction being executed. The interrupt request sampling wait time is required because no interrupt request can be recognized while an instruction is being executed.

❍ **Interrupt handling time (ϕ machine cycle)**

After accepting an interrupt request, the CPU saves the values of dedicated registers to the system stack and fetches the interrupt vector. The interrupt handling time is therefore required. Obtain the interrupt handling time by using the following equations:

When an interrupt starts to be processed: ϕ =24 + 6 x Z machine cycles
When control is returned from an interrupt: ϕ =11 + 6 x Z machine cycles (RETI instruction)

The interrupt handling time varies depending on the address of the stack pointer.

Table 6.4-3 Interpolation Values (Z) for the Interrupt Handling Time

Address pointed to by the stack pointer	Interpolation value (Z)
External 8-bit	+4
External even-numbered address	+1
External odd-numbered address	+4
Internal even-numbered address	0
Internal odd-numbered address	+2

6.5 Software Interrupts

When the software interrupt instruction is executed, control is transferred from the main program to the interrupt processing program. Hardware interrupts are disabled while the software interrupt instruction is being executed.

■ Software Interrupt Activation

❍ Software interrupt activation

The INT instruction is used to activate a software interrupt. A software interrupt does not have an interrupt request flag bit or enable flag like that of a hardware interrupt. Thus, an interrupt request is output whenever the INT instruction is executed.

❍ Hardware interrupt suppression

Since the INT instruction does not have interrupt levels, the interrupt level mask register (ILM) is not updated. During the execution of the INT instruction, the I flag of the condition code register (CCR) is set to 0, and hardware interrupts are masked.

To enable hardware interrupts during software interrupt processing, set the I flag of the condition code register (CCR) to "1" in the software interrupt processing routine.

❍ Software interrupt operation

When the CPU fetches the INT instruction, the software interrupt processing microcode is activated. This microcode saves the internal CPU registers on the system stack, masks hardware interrupts (CCR: I = 0), and branches to the corresponding interrupt vector.

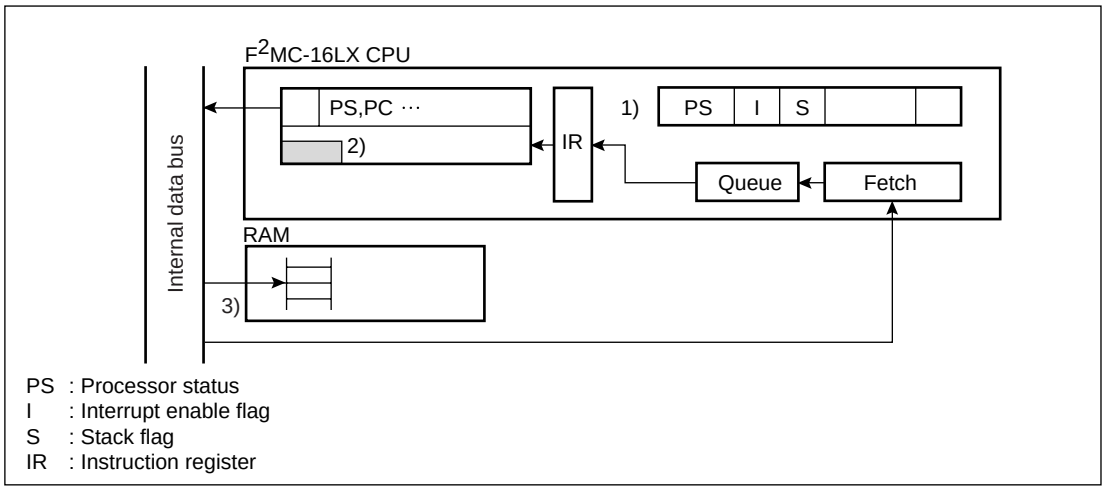
See Section 6.2 "Interrupt Causes and Interrupt Vectors", in CHAPTER 6 "INTERRUPTS" for more information about the allocation of interrupt numbers and interrupt vectors.

■ Returning from a Software Interrupt

In the interrupt processing program, when the interrupt return instruction (RETI instruction) is executed, the data saved to the system stack is restored to the dedicated registers and the processing that was being executed before branching for the interrupt is resumed.

■ Software Interrupt Operation

Figure 6.5-1 Software Interrupt Operation



1. A software interrupt instruction is (INT instruction) executed.
2. The values of the dedicated registers are saved to the system stack. Hardware interrupts are masked. The processing branches to the interrupt vector.
3. The RETI instruction in the user interrupt processing routine terminates the interrupt processing.

Note:

When the program bank register (PCB) is "FF_H", the vector area of the CALLV instruction overlaps the INT #vct8 instruction table. When you create software, watch for duplicated addresses of the CALLV and INT #vct8 instructions.

6.6 Interrupt of Extended Intelligent I/O Service (EI^{2}OS)

The extended intelligent I/O service (EI^{2}OS) automatically transfers data between a peripheral function (resource) and memory. When the data transfer terminates, a hardware interrupt is generated.

■ Extended Intelligent I/O Service (EI^{2}OS)

The extended intelligent I/O service (EI^{2}OS) is a type of hardware interrupt. EI^{2}OS transfers data between a peripheral function (resource) and memory. The user should create program to activate and terminate EI^{2}OS but need not create a data transfer program.

❍ Advantages of extended intelligent I/O service (EI^{2}OS)

Compared to data transfer performed by the interrupt processing routine, EI^{2}OS has the following advantages.

- Coding a transfer program is not necessary, reducing program size.
- Since transfer can be activated due to an interrupt cause of a peripheral function (resource), there is no need of polling for a data transfer cause.
- Incrementing of a transfer address can be selected.
- Incrementing or no update can be selected for the I/O register address.

❍ Extended intelligent I/O service (EI^{2}OS) termination interrupt

The processing branches to the interrupt processing routine when data transfer by EI^{2}OS terminates.

An interrupt processing program can determine the EI^{2}OS termination cause by checking the EI^{2}OS status bits (S1 and S0) of the interrupt control register (ICR).

Interrupt numbers and interrupt vectors are permanently set for each peripheral. See Section 6.2 "Interrupt Causes and Interrupt Vectors", in CHAPTER 6 "INTERRUPTS" for more information.

❍ Interrupt control register (ICR)

This register, which is located in the interrupt controller, activates EI^{2}OS, specifies the EI^{2}OS channel, and displays the EI^{2}OS termination status.

❍ **Extended intelligent I/O service (EI^{2}OS) descriptor (ISD)**

This descriptor, which is located in RAM at 000100_H to $00017F_H$, is an eight-byte data that retains the transfer mode, resource address, transfer count, and buffer address. The descriptor handles 16 channels. The channel is specified by the interrupt control register (ICR).

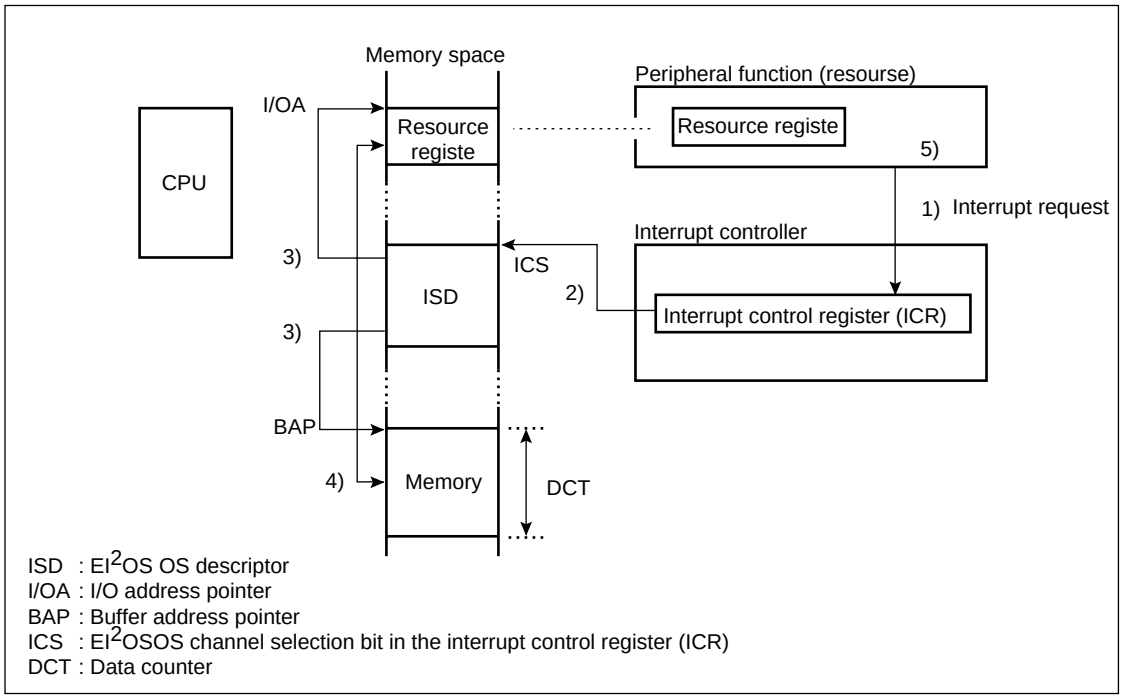
Note:

When the extended intelligent I/O service (EI^{2}OS) is operating, execution of the CPU program stops.

When REALOS is used, EI^{2}OS is disabled.

■ Operation of the Extended Intelligent I/O Service (EI^{2}OS)

Figure 6.6-1 Extended Intelligent I/O Service (EI^{2}OS) Operation



1. A peripheral function (resource) outputs an interrupt request.
2. The interrupt controller selects the EI^{2}OS descriptor in accordance with the setting in the interrupt control register (ICR).
3. The transfer source and transfer destination are read from the descriptor.
4. Transfer is performed between resource and memory.
5. After data transfer is completed, the interrupt request flag bit of the peripheral function (resource) is cleared to "0".

6.6.1 Extended Intelligent I/O Service (EI^{2}OS) Descriptor (ISD)

The extended intelligent I/O service (EI^{2}OS) descriptor (ISD) resides in internal RAM at 000100_H to $00017F_H$. The ISD consists of 8 bytes x 16 channels.

■ Configuration of the Extended Intelligent I/O Service (EI^{2}OS) Descriptor (ISD)

The ISD consists of 8 bytes x 16 channels.

Figure 6.6-2 Configuration of EI^{2}OS Descriptor (ISD)

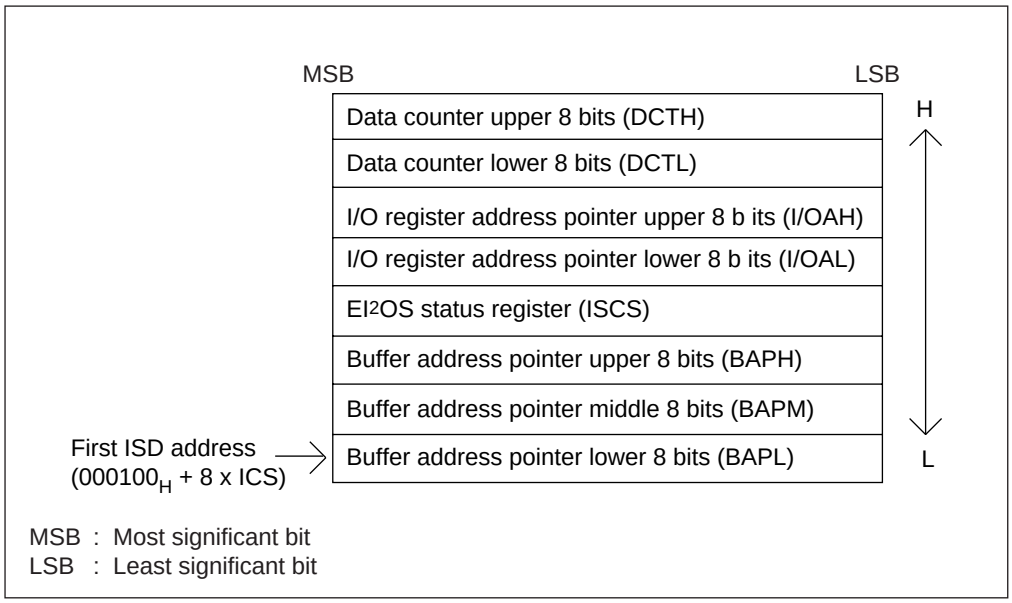


Table 6.6-1 Correspondence between Channel Numbers and Descriptor Addresses

Channel	Descriptor address (*1)
0	000100_H
1	000108_H
2	000110_H
3	000118_H
4	000120_H
5	000128_H
6	000130_H
7	000138_H
8	000140_H

Table 6.6-1 Correspondence between Channel Numbers and Descriptor Addresses

Channel	Descriptor address (*1)
9	000148_H
10	000150_H
11	000158_H
12	000160_H
13	000168_H
14	000170_H
15	000178_H

* The ISD address indicates the first address of 8 bytes.

6.6.2 Registers of the Extended Intelligent I/O Service (EI^{2}OS) Descriptor (ISD)

The extended intelligent I/O service (EI^{2}OS) descriptor (ISD) consists of the following four registers that account for eight bytes in total:

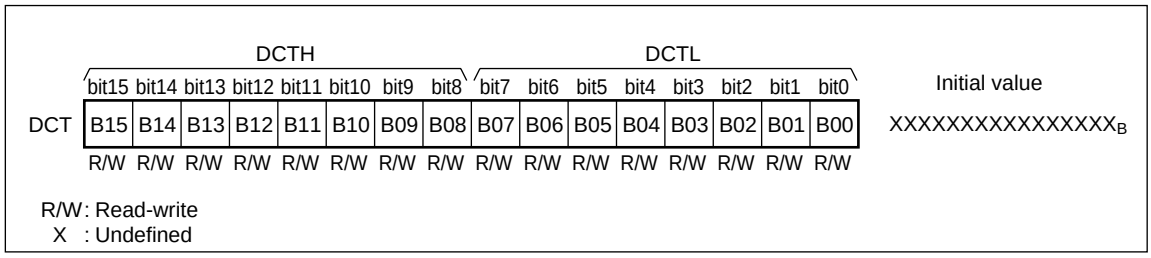
- **Data counter (DCT: 2 bytes)**
- **I/O register address pointer (I/OA: 2 bytes)**
- **EI^{2}OS status register (ISCS: 1 byte)**
- **Buffer address pointer (BAP: 3 bytes)**

The initial values of these registers are undefined.

■ Data Counter (DCT)

The DCT is a 16-bit register in which a transfer data byte count can be set. Every time one byte of data is transferred, the counter is decremented by one. EI^{2}OS terminates when the data counter reaches "0000_H".

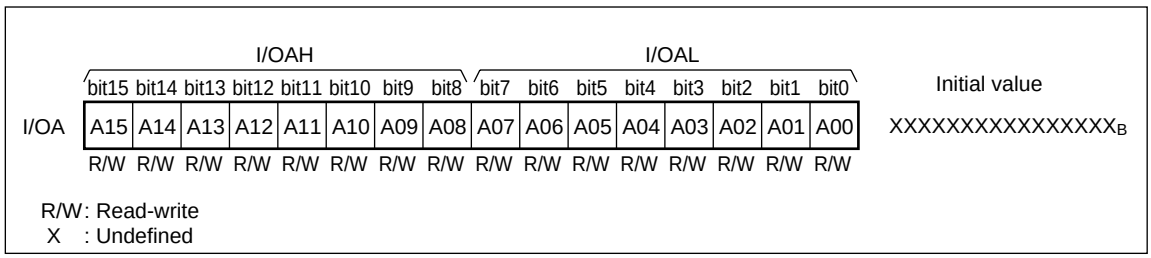
Figure 6.6-3 Configuration of DCT



■ I/O Register Address Pointer (IOA)

The IOA is a 16-bit register that indicates the lower address (A15 to A0) of the I/O register used to transfer data to and from the buffer. The upper address (A23 to A16) is 00_H. Any I/O from 0000_H to $FFFF_H$ can be specified by address.

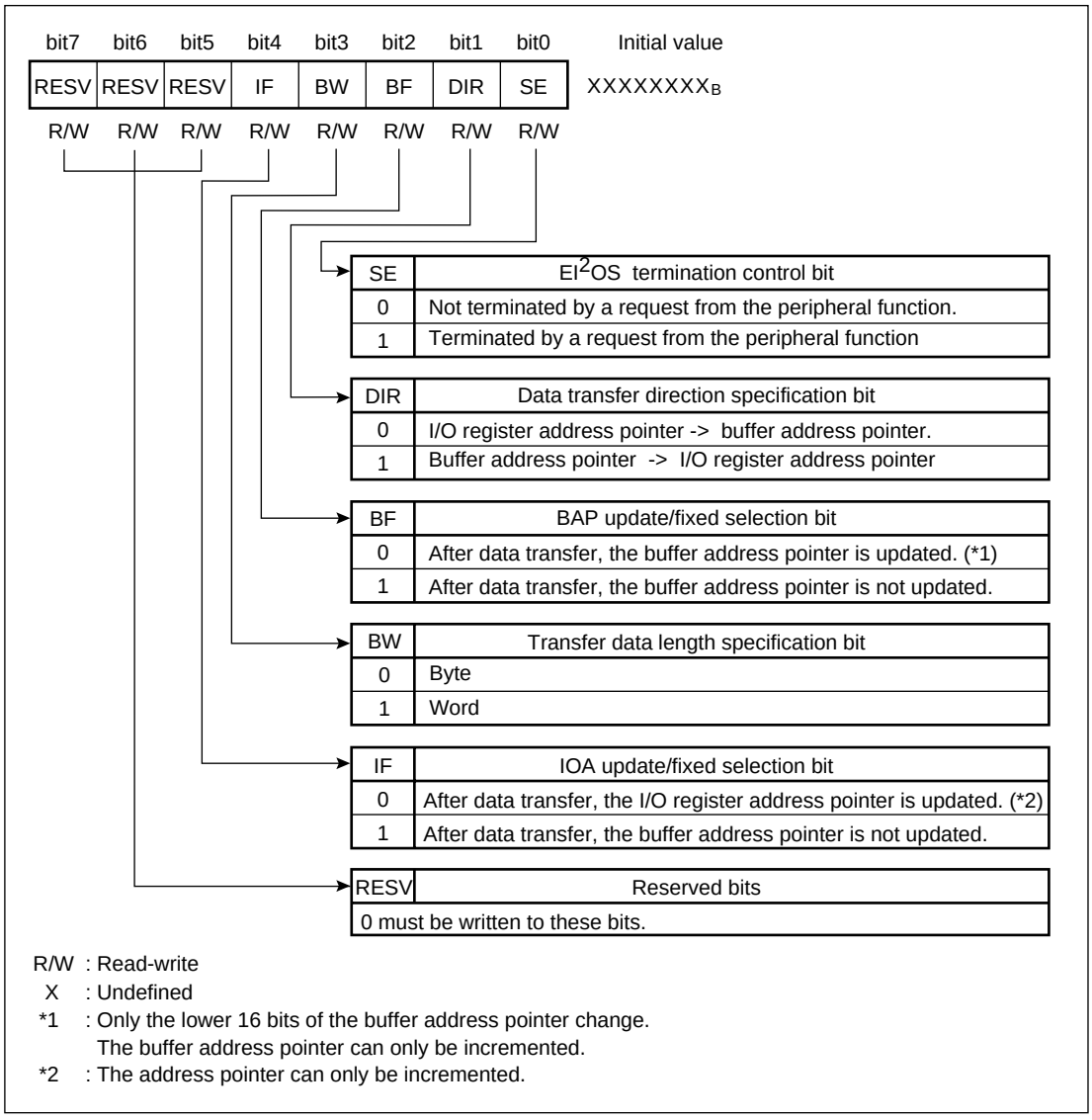
Figure 6.6-4 Configuration of I/O Register Address Pointer (IOA)



■ Extended Intelligent I/O Service (EI^{2}OS) Status Register (ISCS)

The ISCS is an 8-bit register. The ISCS indicates the update/fixed for the buffer address pointer and I/O register address pointer, transfer data format (byte or word), and transfer direction.

Figure 6.6-5 Configuration of EI^{2}OS Status Register (ISCS)

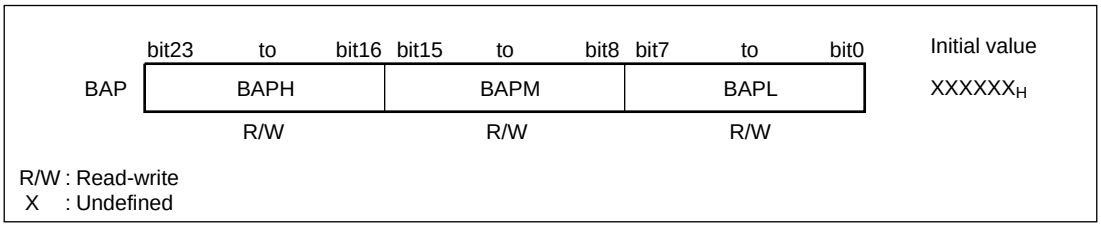


R/W : Read-write
X : Undefined
*1 : Only the lower 16 bits of the buffer address pointer change.
The buffer address pointer can only be incremented.
*2 : The address pointer can only be incremented.

■ Buffer Address Pointer (BAP)

The buffer address pointer (BAP) is a 24-bit register in which a memory address of the data transfer source can be set in the EI^{2}OS operation. Data can be transferred between a 16M-byte memory address and a peripheral function (resource) address because a BAP exists in every channel of EI^{2}OS. If the BAP update/fixed selection bit (BF) of the EI^{2}OS status register (ISCS) is set to "0", only the lower 16 bits (BAPM and BAPL) are incremented and the upper 8 bits (BAPH) are not incremented.

Figure 6.6-6 Configuration of Buffer Address Pointer (BAP)



Reference:

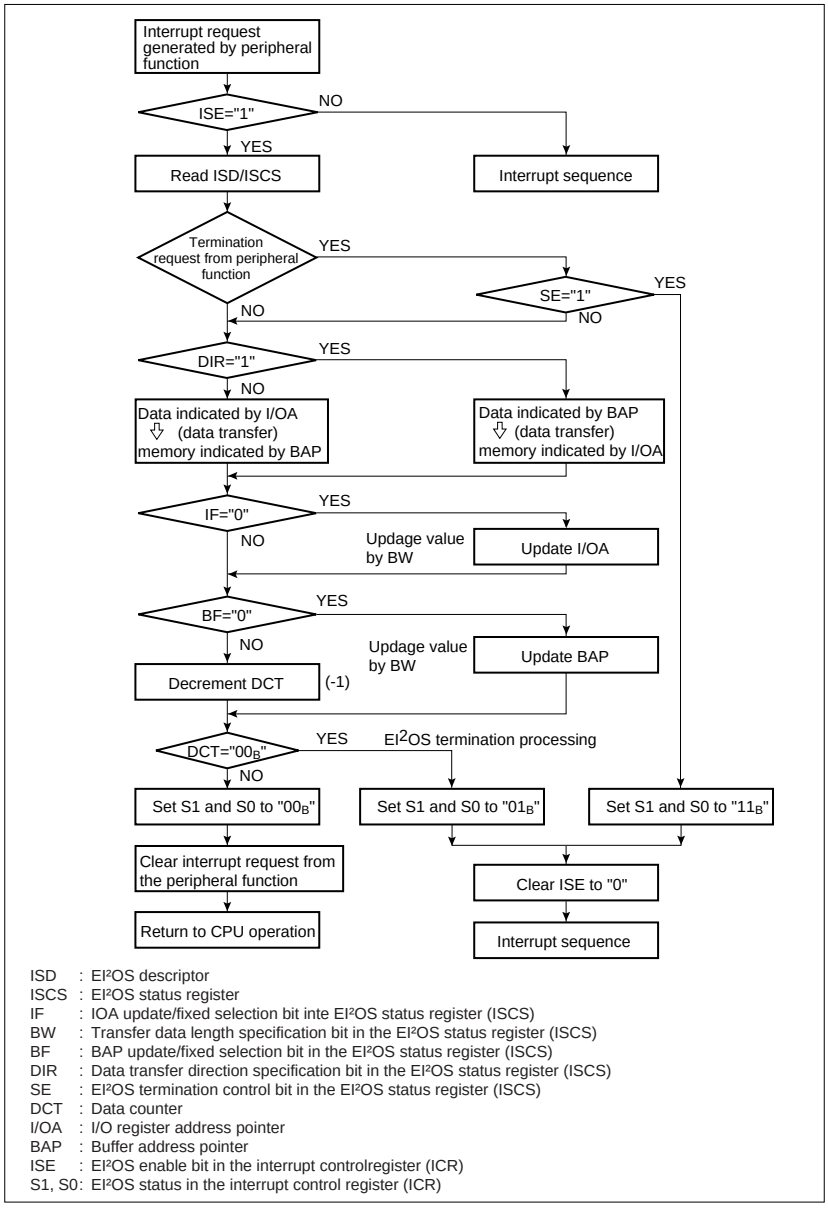
- The area that can be specified by the I/O address pointer (IOA) extends from 000000_H to $00FFFF_H$.
- The area that can be specified with the buffer address pointer (BAP) extends from 000000_H to $FFFFFF_H$.
- The maximum transfer count that can be specified by the data counter (DCT) is 65,536 (64 K bytes).

6.6.3 Operation of the Extended Intelligent I/O Service (EI^{2}OS)

The CPU uses EI^{2}OS to transfer data if a peripheral function (resource) outputs an interrupt request while the corresponding interrupt control register (ICR) is set to enable the activation of EI^{2}OS. When the EI^{2}OS processing terminates, the hardware interrupt processing is performed.

■ Operation Flow of the Extended Intelligent I/O Service (EI^{2}OS)

Figure 6.6-7 Flow of Extended Intelligent I/O Service (EI^{2}OS) Operation



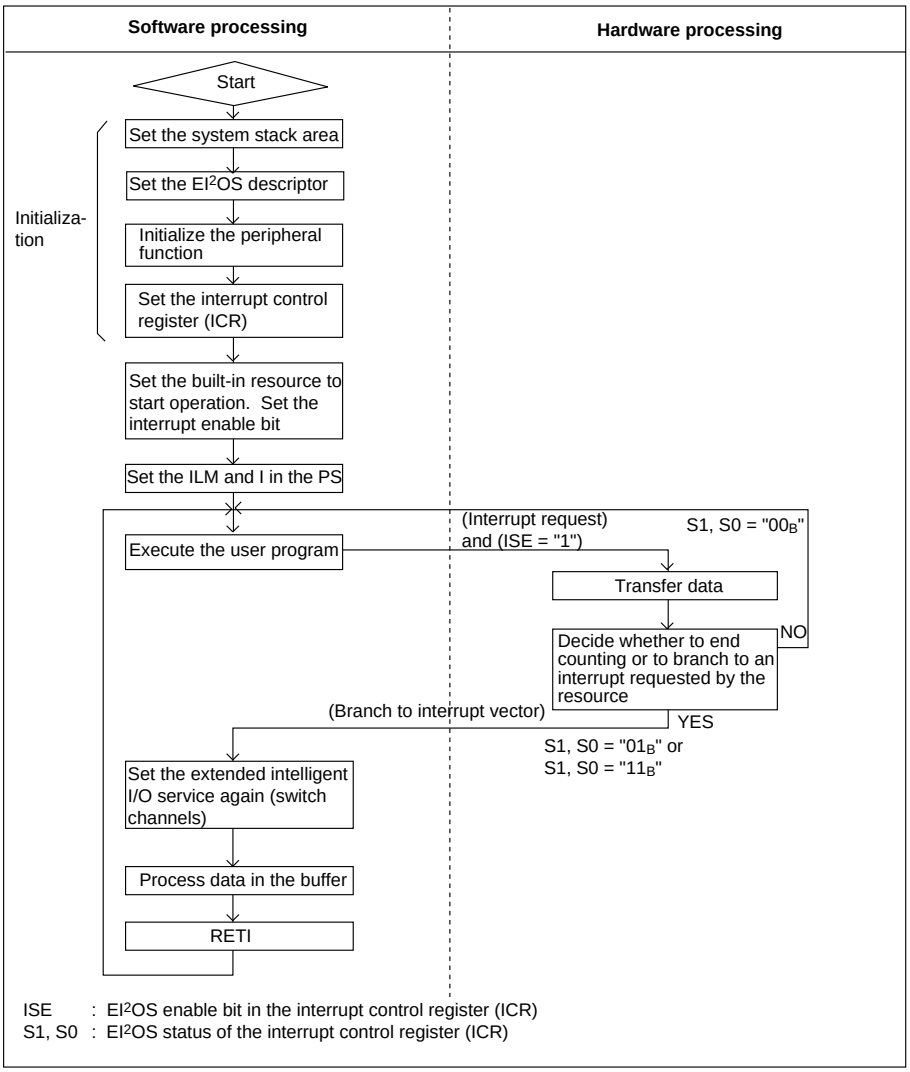
ISD : EI²OS descriptor
ISCS : EI²OS status register
IF : IOA update/fixed selection bit inte EI²OS status register (ISCS)
BW : Transfer data length specification bit in the EI²OS status register (ISCS)
BF : BAP update/fixed selection bit in the EI²OS status register (ISCS)
DIR : Data transfer direction specification bit in the EI²OS status register (ISCS)
SE : EI²OS termination control bit in the EI²OS status register (ISCS)
DCT : Data counter
I/OA : I/O register address pointer
BAP : Buffer address pointer
ISE : EI²OS enable bit in the interrupt controlregister (ICR)
S1, S0: EI²OS status in the interrupt control register (ICR)

6.6.4 Procedure for Using the Extended Intelligent I/O Service (EI^{2}OS)

Before the extended intelligent I/O service (EI^{2}OS) can be used, the system stack area, extended intelligent I/O service (EI^{2}OS) descriptor, perpheral function (resource), and interrupt control register (ICR) must be set.

■ Procedure for Using the Extended Intelligent I/O Service (EI^{2}OS)

Figure 6.6-8 Procedure for Using the Extended Intelligent I/O Service (EI^{2}OS)



6.6.5 Processing Time of the Extended Intelligent I/O Service (EI^{2}OS)

The time required to process the extended intelligent I/O service (EI^{2}OS) varies with the settings for the EI^{2}OS descriptor (ISD):

- **Setting in the EI^{2}OS status register (ISCS)**
- **Address pointed to by the I/O register address pointer (I/OA)**
- **Address pointed to by the buffer address pointer (BAP)**
- **External data bus length for external access**
- **Transfer data length**

Because the hardware interrupt is activated when data transfer by EI^{2}OS terminates, the interrupt handling time is added.

■ Processing Time (one transfer time) of the Extended Intelligent I/O Service (EI^{2}OS)

❍ When data transfer continues

The EI^{2}OS processing time for data transfer continuation is shown in Table 6.6-2 "Extended Intelligent I/O Service Execution Time" based on the EI^{2}OS status register (ISCS) setting.

Table 6.6-2 Extended Intelligent I/O Service Execution Time

EI^{2}OS termination control bit (SE) setting		Terminates due to termination request from the peripherall		Ignores termination request from the peripheral	
I/OA update/fixed selection bit (IF) setting		Fixed	Update	Fixed	Update
BAP address update/fixed selection bit (BF) setting	Fixed	32	34	33	35
	Update	34	36	35	37

Unit: Machine cycle (One machine cycle corresponds to one clock cycle of the machine clock (ϕ).

As shown in Table 6.3-3 "Correspondence between the EI^{2}OS Channel Selection Bits and Descriptor Addresses", interpolation is necessary depending on the EI^{2}OS execution condition.

Table 6.6-3 Data Transfer Interpolation Value for EI^{2}OS Execution Time

I/O register address pointer			Internal access		External access	
			B/Even	Odd	B/Even	8/Odd
Buffer address pointer	Internal access	B/Even	0	+2	+1	+4
		Odd	+2	+4	+3	+6
	External access	B/Even	+1	+3	+2	+5
		8/Odd	+4	+6	+5	+8

B: Byte data transfer
8: External bus using the 8-bit word transfer
Even: Even-numbered address word transfer
Odd: Odd-numbered address word transfer

❍ **When the data counter (DCT) count terminates (final data transfer)**

Interrupt handling time is added because the hardware interrupt is activated when data transfer by EI^{2}OS terminates. The EI^{2}OS processing time when counting terminates is calculated by using the equation below. Z in this equation is a correction value for the interrupt handling time.

EI^{2}OS processing time when counting terminates =
EI^{2}OS processing time when data is transferred + $\underbrace{(21 + 6 \times Z)}$ machine cycles
Interrupt handling time

The interrupt handling time is different for each address pointed to by the stack pointer.

Table 6.6-4 Interpolation Value (Z) for the Interrupt Handling Time

Address pointed to by the stack pointer	Interpolation value (Z)
External 8-bit	+4
External even-numbered address	+1
External odd-numbered address	+4
Internal even-numbered address	0
Internal odd-numbered address	+2

❍ **For termination by a termination request from the peripheral function (resource)**

When data transfer by EI^{2}OS is terminated before completion due to a termination request from the peripheral function (resource) (ICR: S1, S0 = 11_B), the data transfer is not performed and a hardware interrupt is activated. The EI^{2}OS processing time is calculated with the following formula. Z in the formula indicates the interpolation value for the interrupt handling time (Table 6.6-4 "Interpolation Value (Z) for the Interrupt Handling Time").

EI^{2}OS processing time for termination before completion = 36 + 6 x Z) machine cycles

One machine cycle corresponds to one clock cycle of the machine clock (ϕ).

6.7 Exception Processing Interrupt

In the MB90560 and 565 series, exception processing occurs if an undefined instruction is executed. Exception processing is basically the same as an interrupt. When an exception occurs between instructions, program processing is interrupted and the processing branches to the exception processing routine.
Exception processing, occurring due to an unexpected operation, can be used to execute an undefined instruction and detect a CPU runaway status during debugging.

■ Exception Processing

❍ Exception processing operation

In the MB90560 and 565 series, the processing branches to the exception processing routine if an instruction undefined in the instruction map is executed.

The following processing is executed before exception processing branches to the interrupt routine:

- The A, DPR, ADB, DTB, PCB, PC, and PS registers are saved to the system stack.
- The I flag of the condition code register (CCR) is cleared to 0, and hardware interrupts are masked.
- The S flag of the condition code register (CCR) is set to 1, and the system stack is activated

The program counter (PC) value saved to the stack is the exact address where the undefined instruction is stored. For 2-byte or longer instruction codes, the code identified as undefined is stored at this address. When the exception factor type must be determined within the exception processing routine, use this PC value.

❍ Return from exception processing

If the RETI instruction is used to return control from exception processing, a branch to the exception processing routine occurs again because the PC is pointing to an undefined instruction. Use a software reset or input the "L" level from the $\overline{\text{RST}}$ pin (an external reset).

6.8 Stack Operations for Interrupt Processing

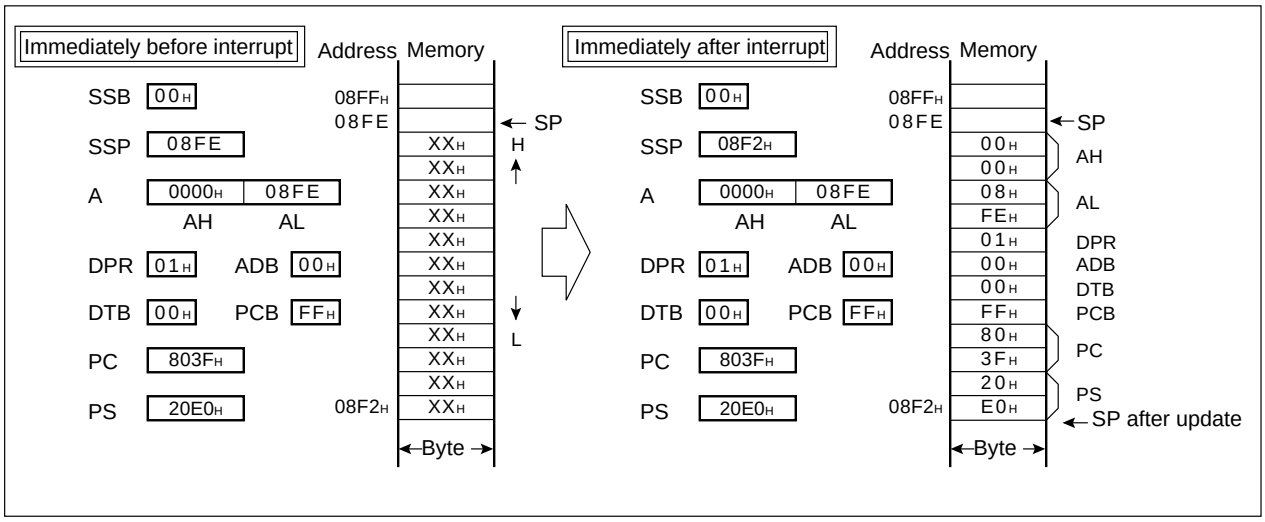
Once an interrupt is accepted, the contents of the dedicated registers are saved to the system stack before a branch to interrupt processing. Execute the interrupt return instruction after the interrupt processing terminates to restore the values saved on the system stack to the dedicated registers.

■ Stack Operations at the Start of Interrupt Processing

Once an interrupt is accepted, the CPU saves the contents of the current dedicated registers to the system stack in the order given below:

- Accumulator (A)
- Direct page register (DPR)
- Additional data bank register (ADB)
- Data bank register (DTB)
- Program bank register (PCB)
- Program counter (PC)
- Processor status (PS)

Figure 6.8-1 Stack Operations at the Start of Interrupt Processing



■ Stack Operations on Return from Interrupt Processing

When the interrupt return instruction (RETI) is executed at the termination of interrupt processing, the PS, PC, PCB, DTB, ADB, DPR, and A values are returned from the stack in reverse order from the order they were placed on the stack. The dedicated registers are restored to the status they had immediately before the start of interrupt processing.

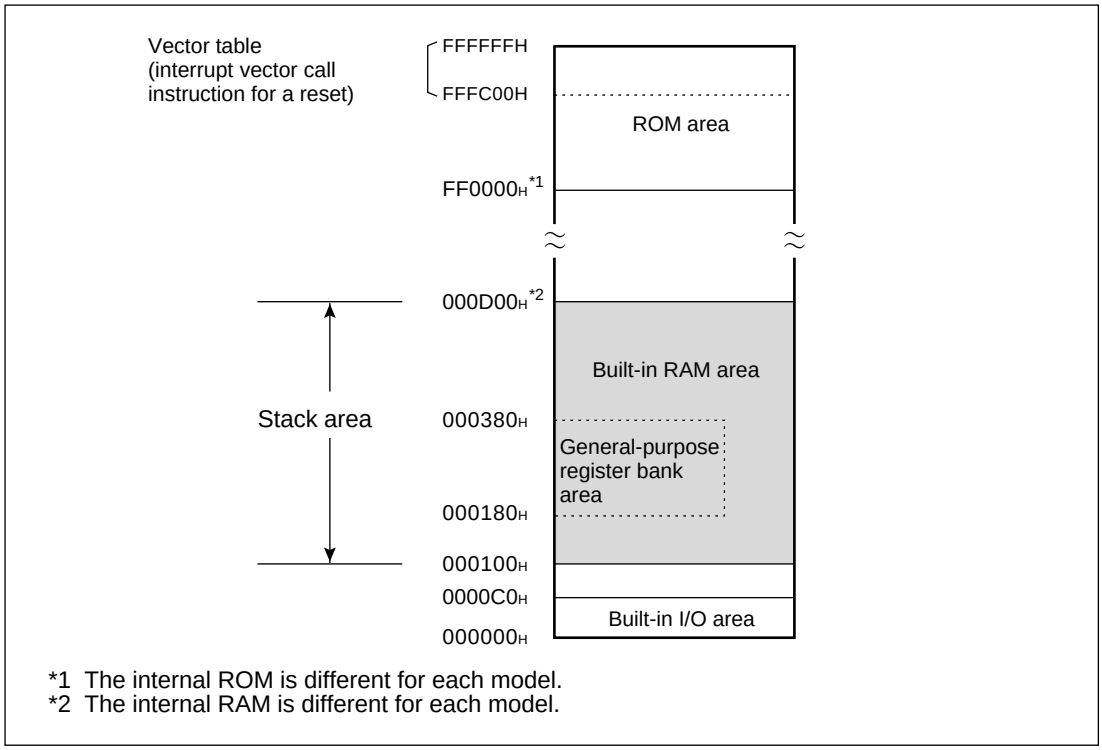
■ Stack Area

❍ Stack area allocation

The stack area is used for saving and restoring the program counter (PC) when the subroutine call instruction (CALL) and vector call instruction (CALLV) are executed in addition to interrupt processing. The stack area is used for temporary saving and restoring of registers by the PUSHW and POPW instructions.

The stack area is allocated together with the data area in RAM.

Figure 6.8-2 Stack Area



*1 The internal ROM is different for each model.
*2 The internal RAM is different for each model.

Note:

- Generally set an even-numbered address in the stack pointers (SSP and USP). If an odd-numbered address is set, one extra cycle is required to save data to, and restore data from the stack.
- Allocate the system stack area, user stack area, and data area so that they do not overlap.

❍ System stack and user stack

The system stack area is used for interrupt processing. When an interrupt is output, the user area being used is switched to the system stack. Use only the system stack unless the stack space needs to be divided in particular.

6.9 Sample Programs for Interrupt Processing

This section contains sample programs for interrupt processing.

■ Sample Programs for Interrupt Processing

❍ Processing specifications

The following is a sample program for an interrupt that uses external interrupt 0 (INT0).

❍ Sample coding

```
DDR1     EQU  000011H           ;Port 1 direction register
ENIR     EQU  000030H           ;DTP/interrupt permission register
EIRR     EQU  000031H           ;DTP/interrupt cause register
ELVR     EQU  000032H           ;Request level setting register
ICR00    EQU  0000B0H           ;Interrupt control register OO
STACK    SSEG                   ;Stack
         RW   100
STACK_T  RW   1
STACK    ENDS
;--------Main program ----------------------------------------------
CODE     CSEG
START:
         MOV  RP,#0             ;General-purpose registers
                                 use the first bank.
         MOV  ILM, #07H         ;Sets ILM in PS to level 7.
         MOV  A, #!STACK_T      ;Sets system stack.
         MOV  SSB, A
         MOVW A, #STACK_T       ;Sets stack pointer, then
         MOVW SP, A             ;Sets SSP because S flag = 1.
         MOV  DDR1, #00000000B  ;Sets P10/INT0 pin to input.
         OR   CCR, #040H        ;Sets I flag of CCR in PS,
                                 enables interrupts.
         MOV  I:ICR00, #00H     ;Sets interrupt level to 0
                                  (highest priority).
         MOV  I:ELVR, #00000001B;Requests that INT0 be made level H.
         MOV  I:EIRR, #00H      ;Clears INT0 interrupt cause.
         MOV  I:ENIR, #01H      ;Enables INT0 input.
          :
LOOP:    NOP                    ;Dummy loop
         NOP
         NOP
         NOP
         BRA  LOOP              ;Unconditional jump
---------Interrupt program ----------------------------------------
ED_INT1:
         MOV  I:EIRR, #00H      ;Acceptance of new INT0 not allowed
         NOP
         NOP
         NOP
         NOP
```

```
        NOP
        NOP
        RETI                ;Return from interrupt
CODE    ENDS
;--------Vector setting----------------------------------------------
VECT    CSEGABS=0FFH
        ORG  0FF98H         ;Sets vector for interrupt #25 (19H).
        DSL  ED_INT1
        ORG  0FFDCH         ;Sets reset vector.
        DSL  START
        DB   00H            ;Sets single-chip mode.
VECT    ENDS
        END  START
```

■ Processing Specifications of Sample Program for Extended Intelligent I/O Service (EI^{2}OS)

❍ Processing Specifications

1. This program detects the H level signal input to the INT0 pin and activates the extended intelligent I/O service (EI^{2}OS).
2. When the H level is input to the INT0 pin, EI^{2}OS is activated. Data is transferred from port 0 to the memory at the 3000_H address.
3. The number of transfer data bytes is 100 bytes. After 100 bytes are transferred, an interrupt is generated because EI^{2}OS transfer has terminated.

❍ Sample coding

```
DDR1    EQU   000011H           ;Port 1-direction register
ENIR    EQU   000030H           ;DTP/interrupt permission register
EIRR    EQU   000031H           ;DTP/interrupt cause register
ELVR    EQU   000032H           ;Request level setting register
ICR00   EQU   0000B0H           ;Interrupt control register 00
BAPL    EQU   000100H           ;Lower buffer address pointer
BAPM    EQU   000101H           ;Middle buffer address pointer
BAPH    EQU   000102H           ;Upper buffer address pointer
ISCS    EQU   000103H           ;EI2OS status
I/OAL   EQU   000104H           ;Lower I/O address pointer
I/OAH   EQU   000105H           ;Upper I/O address pointer
DCTL    EQU   000106H           ;Low-order data counter
DCTH    EQU   000107H           ;High-order data counter
ER0     EQU   EIRR:0            ;Definition of external interrupt
                                 request flag bit
STACK   SSEG                    ;Stack
        RW    100
STACK_T RW    1
STACK   ENDS
;-------------------Main program-------------------------------------
CODE    CSEG
START:
        AND   CCR, #0BFH       ;Clears the I flag of the CCR in the
                                 PS and prohibits interrupts.
        MOV   RP, #00           ;Sets the register bank pointer.
        MOV   A, #!STACK_T      ;Sets the system stack.
        MOV   SSB, A
```

```
          MOVW  A, #STACK_T       ;Sets the stack pointer, then
          MOVW  SP, A             ;Sets SSP because the S flag = 1.
          MOV   I:DDR1, #00000000B;Sets the P10/INT0 pin to input.
          MOV   BAPL, #00H        ;Sets the buffer address (003000H).
          MOV   BAPM, #30H
          MOV   BAPH, #00H
          MOV   ISCS, #00010001B  ;No I/O address update, byte
                                   transfer,buffer address updated
                                  ;I/O -> buffer transfer, terminated
                                   by the peripheral function.
          MOV   I/OAL, #00H       ;Sets the transfer source address
                                   (port 0:000000H).
          MOV   I/OAH, #00H
          MOV   DCTL, #64H        ;Sets the number of transfer bytes
                                   (100 bytes).
          MOV   DCTH, #00H
          MOV   I:ICR00,#00001000B;EI2OS channel 0, EI2OS enable,
                                   interrupt level 0
                                   (highest priority)
          MOV   I:ELVR, #00000001B;Requests that INT0 be made H level.
          MOV   I:EIRR, #00H      ;Clears the INT0 interrupt cause.
          MOV   I:ENIR, #01H      ;Enables INT0 interrupts.
          MOV   ILM, #07H         ;Sets the ILM in the PS to level 7.
          OR    CCR, #040H        ;Sets the I flag of the CCR in the
                                   PS and enables interrupts.
           :
LOOP      BRA   LOOP              ;Infinite loop
;---------------Interrupt program-----------------------------------
WARI      CLRB  ER0               ;Clears interrupt/DTP request flag.
           :
          User processing         ;Checks EI2OS termination factor,
                    :             ;processes data in buffer,
                                   sets EI2OS again.
          RETI
CODE      ENDS
;---------------Vectort processing----------------------------------
VECT      CSEG  ABS=0FFH
          ORG   0FFD0H            ;Sets vector for interrupt #11
(0BH).
          DSL   WARI
          ORG   0FFDCH            ;Sets reset vector.
          DSL   START
          DB    00H               ;Sets single-chip mode.
VECT      ENDS
          END   START
```

CHAPTER 7 SETTING A MODE

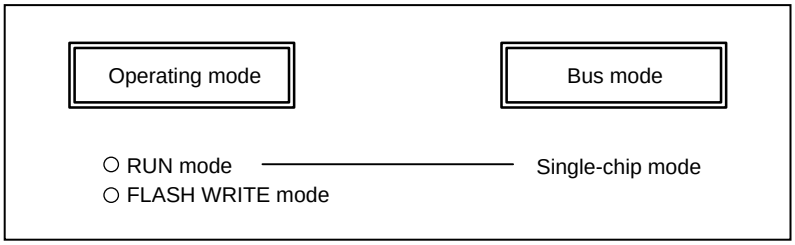
This chapter describes the operating modes of the MB90560/565 series.

7.1 Setting a Mode

Set the mode pin level used after a reset and the mode data in the mode register to determine the operating mode.

■ Mode Setting

Figure 7.1-1 Mode Classification



■ Operating Modes

An operating mode can be specified in the mode pins (MD2 to MD0) and the bus mode setting bits (M1 and M0) of the mode data register. The microcontroller operates in the specified operating mode.

Note:

Set single-chip mode for the MB90560 and 565 series.

To set single-chip mode, set the MD2 to MD0 pins to "011_B" and the M1 and M0 bits of the mode data register to "00_B", respectively.

■ Bus Mode

The bus mode varies depending on whether the memory into which the reset vector should be read is internal or external. Set the MD2 to MD0 pins and the M1 and M0 bits of the mode data register to specify a bus mode. Set the MD2 to MD0 pins to determine a bus mode in which a reset vector and mode data should be read. Additionally, set the M1 and M0 bits of the mode data register to specify a bus mode.

For more information, see Sections 7.2 "Mode Pins (MD2 to MD0)" and 7.3 "Mode Data Register".

■ RUN Mode

The RUN mode means CPU operating mode. The RUN mode includes main clock mode, PLL clock mode, and various low power consumption modes. See CHAPTER 5 "LOW POWER CONSUMPTION MODE" for details.

7.2 Mode Pins (MD2 to MD0)

Three external pins, MD2 to MD0, are supported as the mode pins. These are used to specify how the reset vector and mode data are fetched.

■ Mode Pins (MD2 to MD0)

Use the mode pins to specify whether a reset vector should be read from external or internal memory. Use the mode data register to specify the external data bus width if a reset vector should be read from external memory.

For a built-in flash memory type, use the mode pins to specify the flash memory write mode in which a program should be written to the built-in flash memory.

Table 7.2-1 Mode Pin Settings

MD2	MD2	MD0	Mode name	Reset vector access area	External data bus width	Remarks
0	0	0	Setting not allowed			
0	0	1				
0	1	0				
0	1	1	Internal vector mode	Internal memory	Specified in the mode data register	The reset sequence and subsequent sequences are controlled by mode data.
1	0	0	Setting not allowed			
1	0	1				
1	1	0	Flash serial write mode	-	-	-
1	1	1	Flash memory mode	-	-	Mode when the parallel writer is used

MD2 to MD0: Connect the pins to Vss for 0 and to Vcc for 1.
*: The flash memory serial write cannot be executed just by setting the mode terminal.

Another terminal must be also set. For details, see CHAPTER 21 "EXAMPLE OF MB90F562/F562B/F568 SERIAL PROGRAMMING CONNECTION".

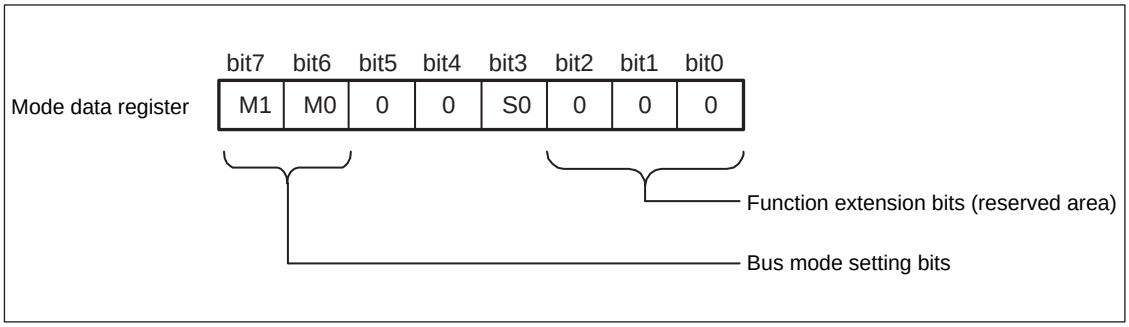
7.3 Mode Data Register

The mode data register is at memory location $FFFFDF_H$, and is used to specify the operation after a reset sequence.

■ Mode Data Register

The mode data at address "$FFFFDF_H$" can be fetched to the mode data register while a reset sequence is being executed. The contents of the mode data register can be changed while a reset sequence is being executed. They cannot be changed using an instruction. The mode data setting takes effect after a reset sequence.

Figure 7.3-1 Mode Data Register Configuration



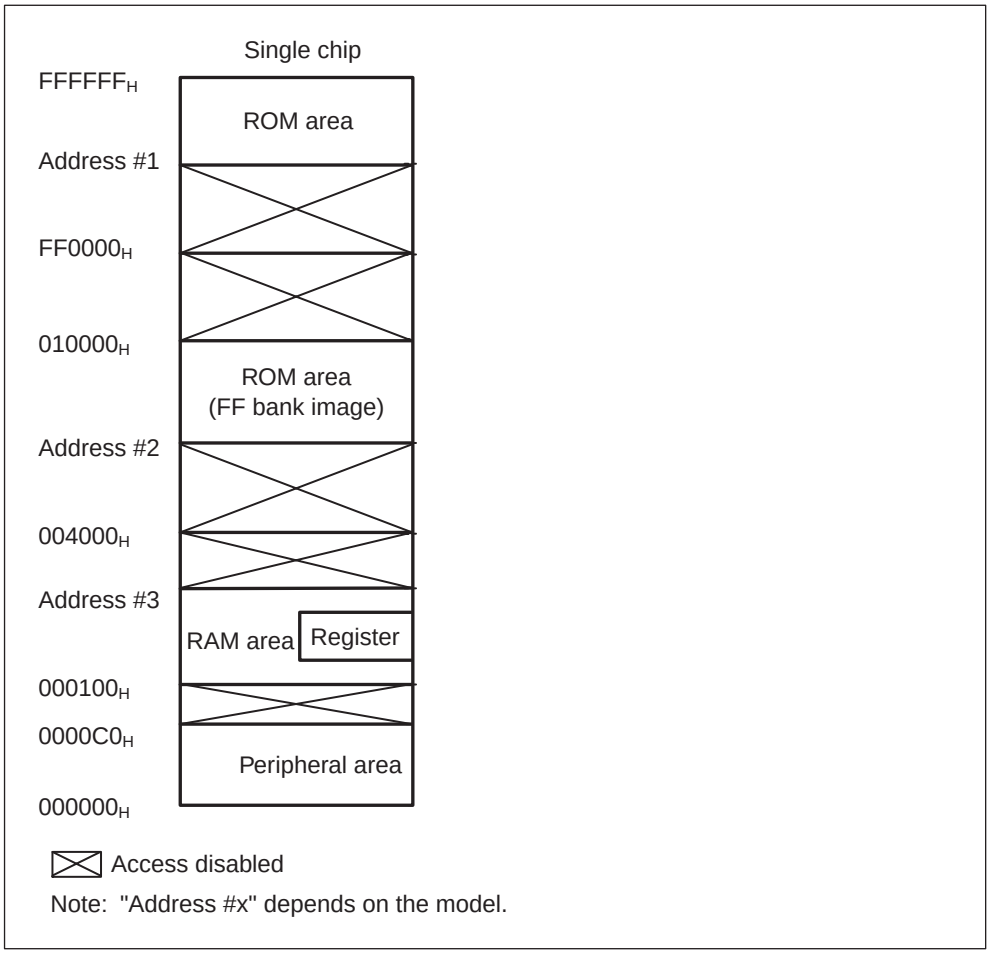
■ Bus Mode Setting Bits

The bus mode setting bits specify operating mode after a reset sequence.

Table 7.3-1 Bus Mode Setting Bits and Functions

M1	M0	Function	Remarks
0	0	Single-chip mode	Described in this manual
0	1	(Setting not allowed)	-
1	0		
1	1		

Figure 7.3-2 Memory Map in Single-Chip Mode



■ Relationship between Mode Pins and Mode Data

Table 7.3-2 Relationship between Mode Pins and Mode Data

Bus mode	Mode pin settings			Mode data settings	
	MD2	MD1	MD0	M1	M0
Single-chip mode	0	1	1	0	0

CHAPTER 8 I/O PORTS

This chapter describes the functions and operations of the I/O ports.

8.1 Overview of I/O Ports

Up to 51 I/O ports (parallel I/O ports) are provided. The ports are also used as resource I/O pins (I/O pins of peripheral functions).

■ I/O Port Functions

I/O ports include port direction registers (DDRs) and port data registers (PDRs). Each bit in a DDR specifies input or output of a port pin. A PDR register specifies output data to a port pin. If a DDR register sets an I/O port pin to input, read a PDR register to read the level value of the port pin. If a DDR register sets an I/O port pin to output, the value of a PDR register is output to the port pin. The following lists resources to be used also as I/O port functions:

- Port 0: I/O port
- Port 1: I/O port/ (external interrupt input pins, free-running timer external clock input pin) resource used in combination
- Port 2: I/O port/ resource (16-bit reload timer, input capture) used in combination
- Port 3: I/O port/ resource (output compare, UART0) used in combination
- Port 4: I/O port/ resource (UART0, 8-bit/16-bit PPG timer) used in combination
- Port 5: I/O port/ resource (analog input pins) used in combination
- Port 6: I/O port/ resource (UART1) used in combination

Table 8.1-1 Functions of Individual Ports

Port	Pin	Input form	Output form	Function								
Port0	P00 to P07	CMOS (hysteresis)	CMOS pull-up resistor selectable	General I/O port	P07	P06	P05	P04	P03	P02	P01	P00
Port1	P10/INT0 to P17/FRCK			General I/O port	P17	P16	P15	P14	P13	P12	P11	P10
				Resource	FRCK	INT6	INT5	INT4	INT3	INT2	INT1	INT0
Port2	P20/TIN0 to P27/IN3		CMOS	General I/O port	P27	P26	P25	P24	P23	P22	P21	P20
				Resource	IN3	IN2	IN1	IN0	T01	TIN1	TO0	TIN0
Port3	P30/RTO0 to P37/SOT0			General I/O port	P37	P36	P35	P34	P33	P32	P31	P30
				Resource	SOT0	SIN0	RTO5	RTO4	RTO3	RTO2	RTO1	RTO0
Port4	P40/SCK0 to P46/PPG5	CMOS (hysteresis)		General I/O port	-	P46	P45	P44	P43	P42	P41	P40
				Resource	-	PPG5	PPG4	PPG3	PPG2	PPG1	PPG0	SCK0
Port5	P50/AN0 to P57/AN7	Analog/ CMOS (hysteresis)		General I/O port	P57	P56	P55	P54	P53	P52	P51	P50
				Analog input	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
Port6	P60/SIN1 to P63/INT7	CMOS (hysteresis)		General I/O port	-	-	-	-	P63	P62	P61	P60
				Resource	-	-	-	-	INT1 / DTTI	SCK1	SOT1	SIN1

Note:

Port 5 is also used for analog input pins. To use the port as a general-purpose port, set the Port 5 direction register (DDR5) and Port 5 data register (PDR5) as well as the analog input enable register (ADER) to "00_H". A reset initializes the ADER to "FF_H". To specify analog input and I/O ports for each pin, set the corresponding bit of the ADER to "0".

8.2 Registers for I/O ports

This section provides a list of registers related to I/O port settings.

■ Registers for I/O Ports

Table 8.2-1 Registers for Ports 0 to 6

Register	Read/Write	Address	Initial value
Port 0 data register (PDR0)	R/W	000000_H	$XXXXXXXX_B$
Port 1 data register (PDR1)	R/W	000001_H	$XXXXXXXX_B$
Port 2 data register (PDR2)	R/W	000002_H	$XXXXXXXX_B$
Port 3 data register (PDR3)	R/W	000003_H	$XXXXXXXX_B$
Port 4 data register (PDR4)	R/W	000004_H	$XXXXXXXX_B$
Port 5 data register (PDR5)	R/W	000005_H	$XXXXXXXX_B$
Port 6 data register (PDR6)	R/W	000006_H	$XXXXXXXX_B$
Port 0 data direction register (DDR0)	R/W	000010_H	00000000_B
Port 1 data direction register (DDR1)	R/W	000011_H	00000000_B
Port 2 data direction register (DDR2)	R/W	000012_H	00000000_B
Port 3 data direction register (DDR3)	R/W	000013_H	00000000_B
Port 4 data direction register (DDR4)	R/W	000014_H	$X0000000_B$
Port 5 data direction register (DDR5)	R/W	000015_H	00000000_B
Port 6 data direction register (DDR6)	R/W	000016_H	$XXXX0000_B$
Analog data input enable register (ADER)	R/W	000017_H	11111111_B
Port 0 pull-up resistor setting register (RDR0)	R/W	$00008C_H$	00000000_B
Port 1 pull-up resistor setting register (RDR1)	R/W	$00008D_H$	00000000_B

R/W: Read/write enabled
X: Undefined

8.3 Port 0

Port 0 is an I/O port. This section describes the configuration of Port 0, a block diagram of pins, and registers.

■ Port 0 Configuration

Port 0 consists of the following elements:

- I/O pins
- Port 0 data register (PDR0)
- Port 0 data direction register (DDR0)
- Port 0 pull-up resistor setting register (RDR0)

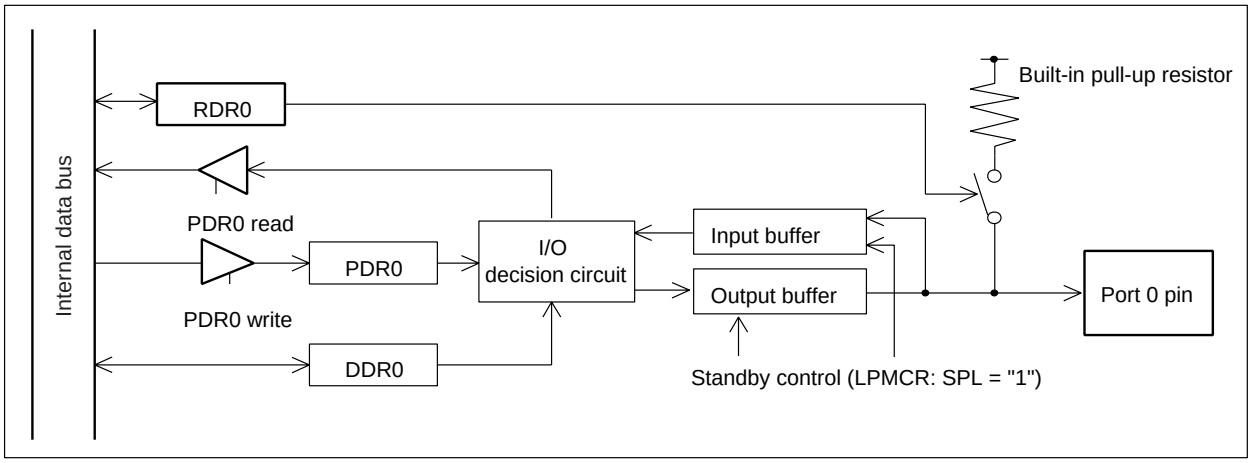
■ Port 0 Pins

Table 8.3-1 Pin Settings of Port 0

I/O port name	Pin name	I/O port function	I/O form	
			Input	Output
Port 0	P00	P00	CMOS (hysteresis)	CMOS
	P01	P01		
	P02	P02		
	P03	P03		
	P04	P04		
	P05	P05		
	P06	P06		
	P07	P07		

■ Block Diagram of Port 0 Pins

Figure 8.3-1 Block Diagram of Port 0 Pins



■ Port 0 Registers

Port 0 registers are PDR0, DDR0, and RDR0. The bits making up each register correspond to the port 0 pins on a one-to-one basis.

Table 8.3-2 Port 0 Pins and Their Corresponding Register Bits

I/O Port	Register bits and corresponding port pins								
Port 0	PDR0,DDR0,RDR0	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
	Corresponding pin	P07	P06	P05	P04	P03	P02	P01	P00

8.3.1 Port 0 Registers (PDR0, DDR0, and RDR0)

This section describes the port 0 registers

■ Functions of Port 0 Registers

❍ **Port 0 data register (PDR0)**

The PDR0 register specifies the output values of pins of Port 0.

❍ **Port 0 data direction register (DDR0)**

The DDR0 register specifies the I/O directions of pins of Port 0. A pin serves as an output port when a bit corresponding to the pin is set to "1". A pin serves as an input port when a bit corresponding to the pin is set to "0".

❍ **Port 0 pull-up resistor setting register (RDR0)**

Each bit of the RDR0 register corresponding to a pin specifies whether a pull-up resistor should be connected to, or disconnected from the pin. A pull-up resistor is connected to a pin of Port 0 when a bit corresponding to the pin is set to "1". A pull-up resistor is disconnected from a pin of Port 0 when a bit corresponding to the pin is set to "0".

Table 8.3-3 Port 0 Register Functions

Register name	Bit value	During reading		During writing		Address	Initial value
		Input port	Output port	Input port	Output port		
Port 0 data register (PDR0)	0	The pin is at the "L" level.	The corresponding bit of PDR0 register is set to "0".	"0" is written to the corresponding bit of PDR0 register.	The "L" level is output from the pin.	000000_H	$XXXXXXXX_B$
	1	The pin is at the "H" level.	The corresponding bit of PDR0 register is set to "1".	"1" is written to the corresponding bit of PDR0 register.	The "H" level is output from the pin.		
Port 0 direction register (DDR0)	0	The corresponding bit of DDR0 register is set to "0".		The pin serves as an input port.		000010_H	00000000_B
	1	The corresponding bit of DDR0 register is set to "1".		The pin serves as an output port.			
Port 0 pull-up resistor setting register (RDR0)	0	The corresponding bit of RDR0 register is set to "0".		The pull-up resistor is disconnected and the pin has high impedance.	The pull-up resistor is disconnected.	$00008C_H$	00000000_B
	1	The corresponding bit of RDR0 register is set to "1".		The pull-up resistor is connected and the pin retains the "H" level.	The pull-up resistor is disconnected.		

X: Undefined

8.3.2 Operation of Port 0

This section describes the operation of port 0.

■ Operation of Port 0

❍ **Setting Port 0 as an output port in the Port 0 direction register (DDR0)**

- The value stored in the Port 0 data register (PDR0) is output to the Port 0 pin.
- If the PDR0 register is read, the value stored in the PDR0 register is output.

❍ **Setting Port 0 as an input port in the Port 0 direction register (DDR0)**

- The Port 0 pin has high impedance. However, the Port 0 pin retains the "H" level if the bit in the Port 0 pull-up resistor register (RDR0) is set to "1" and thus the pull-up resistor is connected.
- If the Port 0 data register (PDR0) is set to a value, the value stored in the PDR0 register is retained but not output to the pin.
- If the PDR0 register is read, the pin input level ("0" for "L" or "1" for "H") is output.

Note:

If a read-modify-write instruction (such as the bit set instruction) is used to access the PDR0 register, no bit specified for output in the DDR0 register is affected. For a bit specified for input in the DDR0 register, however, the pin input level is written to the PDR0 register. Therefore, to change a bit specified for input to output, first write an output value to the PDR0 register and then specify the DDR0 register as an output port.

❍ **Port operation after a reset**

- When the CPU is reset, the DDR0 and RDR0 registers are initialized to "00_H" and the Port 0 pin has high impedance.
- The PDR0 register is not initialized when the CPU is reset. To use the PDR0 as an output port, first write an output value to the PDR0 register and then specify the DDR0 register as an output port.

❍ **Port operation in stop or time-base timer mode**

If the port switches to stop mode or timebase timer mode while the pin status setting bit (SPL) of the low power consumption mode control register (LPMCR) is set to "1", the pins come to have high impedance regardless of the value in the Port 0 direction register (DDR0). Note that the input buffer is forcibly shut off to prevent leakage due to an open circuit.

Note:

If a pull-up resistor is connected, setting the pin status setting bit (SPL) of the low power consumption mode control register (LPMCR) to "1" does not disconnect the pull-up resistor but holds the pin level at "H".

Table 8.3-4 States of Port 0 Pins

Pin	Normal operation	Sleep mode	Stop mode or time-base timer mode (SPL = 0)	Stop mode or time-base timer mode (SPL = 1, RDR = 0)	Stop mode or time-base timer mode (SPL = 1, RDR = 1)
P00 to P07	I/O port	I/O port	Input shut off/ output held	Input shut down/ output in Hi-z	Input shut down/held at H level

SPL: Pin state specification bit of low-power consumption mode control register (LPMCR:SPL)
Hi-z: High impedance

8.4 Port 1

Port 1 is an I/O port that can also be used for resource input. Each of the port pins can be switched between a resource and an I/O port in accordance with the corresponding bit. This section describes the configuration of Port 1, a block diagram of pins, and registers mainly in terms of I/O ports.

■ Port 1 Configuration

Port 1 consists of the following elements:

- I/O port pins/resource input pins (P10/INT0 to P17/FRCK)
- Port 1 data register (PDR1)
- Port 1 data direction register (DDR1)
- Port 1 pull-up resistor setting register (RDR1)

■ Port 1 Pins

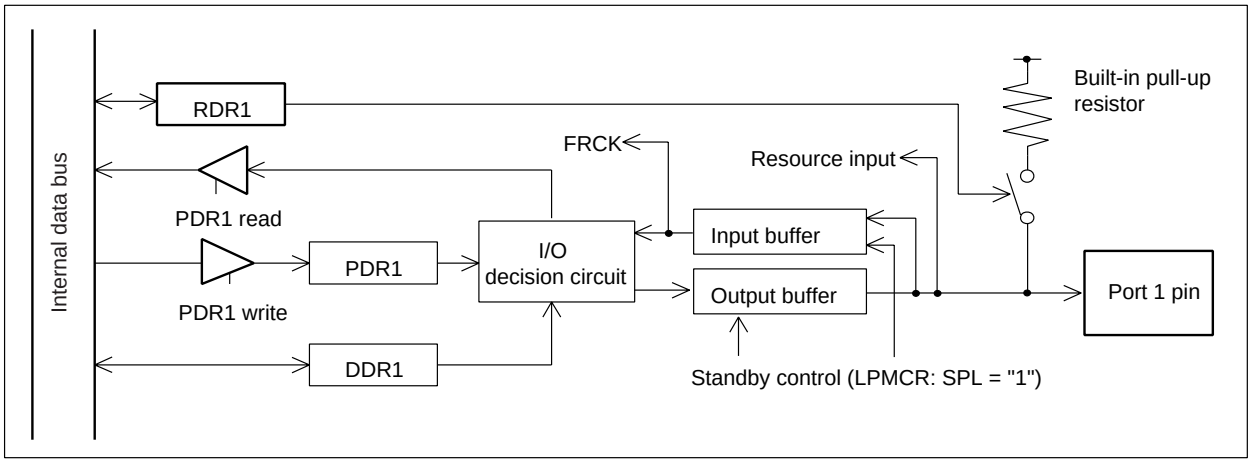
The I/O pins of Port 1 also serve as resource input. The I/O pins, when used as resource input pins, cannot be used as I/O ports.

Table 8.4-1 Port 1 Pins

I/O port name	Pin	I/O Port function	Resource function		I/O form	
					Input	Output
Port 1	P10/INT0	P10	INT0	External interrupt input	CMOS (hysteresis)	CMOS
	P11/INT1	P11	INT1			
	P12/INT2	P12	INT2			
	P13/INT3	P13	INT3			
	P14/INT4	P14	INT4			
	P15/INT5	P15	INT5			
	P16/INT6	P16	INT6			
	P17/FRCK	P17	FRCK	External clock input for free-running timer		

■ Block Diagram of Port 1 Pins

Figure 8.4-1 Block Diagram of Port 1 Pins



■ Port 1 Registers

Port 1 registers are PDR1, DDR1, and RDR1. The bits making up each register correspond to the port 1 pins on a one-to-one basis.

Table 8.4-2 Port 1 Pins and Their Corresponding Register Bits

I/O port name	Register bits and corresponding port pins								
Port 1	PDR1,DDR1,RDR1	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
	Corresponding pin	P17	P16	P15	P14	P13	P12	P11	P10

8.4.1 Port 1 Registers (PDR1, DDR1, and RDR1)

This section describes the port 1 registers.

■ Functions of Port 1 Registers

❍ **Port 1 data register (PDR1)**

The PDR1 register specifies the output value of each pin of Port 1.

❍ **Port 1 direction register (DDR1)**

The DDR1 register specifies the I/O directions of pins of Port 1. A pin serves as an output port when a bit corresponding to the pin is set to "1". A pin serves as an input port when a bit corresponding to the pin is set to "0".

❍ **Port 1 pull-up resistor setting register (RDR1)**

Each bit of the RDR1 register corresponding to a pin specifies whether a pull-up resistor should be connected to, or disconnected from the pin. A pull-up resistor is connected to a pin of Port 1 when a bit corresponding to the pin is set to "1". A pull-up resistor is disconnected from a pin of Port 1 when a bit corresponding to the pin is set to "0".

Table 8.4-3 Port 1 Register Functions

Register	Bit value	During reading		During writing		Address	Initial value
		Input port	Output port	Input port	Output port		
Port 1 data register (PDR1)	0	The pin is at the "L" level.	The corresponding bit of PDR1 register is set to "0".	"0" is written to the corresponding bit of PDR1 register.	The "L" level is output from the pin.	000001_H	$XXXXXXXX_B$
	1	The pin is at the "H" level.	The corresponding bit of PDR1 register is set to "1".	"1" is written to the corresponding bit of PDR1 register.	The "H" level is output from the pin.		
Port 1 data register (DDR1)	0	The corresponding bit of DDR1 register is set to "0".		The pin serves as an input port.		000011_H	00000000_B
	1	The corresponding bit of DDR1 register is set to "1".		The pin serves as an output port.			
Port 1 pull-up resistor setting register (RDR1)	0	The corresponding bit of RDR1 register is set to "0".		The pull-up resistor is disconnected and the pin has high impedance.	The pull-up resistor is disconnected.	$00008D_H$	00000000_B
	1	The corresponding bit of RDR1 register is set to "1".		The pull-up resistor is connected and the pin retains the "H" level.	The pull-up resistor is disconnected.		

X: Undefined

8.4.2 Operation of Port 1

This section describes the operation of port 1.

■ Operation of Port 1

❍ Setting Port 1 as an output port in the Port 1 direction register (DDR1)

- Setting Port 1 as an output port in the Port 1 direction register (DDR1)
- The value stored in the Port 1 data register (PDR1) is output to the Port 1 pin. If the PDR1 register is read, the value stored in PDR1 register is output.

❍ Setting Port 1 as an input port in the Port 1 direction register (DDR1)

- The Port 1 pin has high impedance. However, the Port 1 pin retains the "H" level if the bit in the Port 1 pull-up resistor register (RDR1) is set to "1" and thus the pull-up resistor is connected.
- If the Port 1 data register (PDR1) is set to a value, the value stored in the PDR1 register is retained but not output to the pin.
- If the PDR1 register is read, the pin input level ("0" for "L" or "1" for "H") is output.

Note:

If a read-modify-write instruction (such as the bit set instruction) is used to access the PDR1 register, no bit specified for output in the DDR1 register is affected. For a bit specified for input in the DDR1 register, however, the pin input level is written to the PDR1 register. Therefore, to change a bit specified for input to output, first write an output value to the PDR1 register and then specify the DDR1 register as an output port.

❍ Port operation after a reset

- When the CPU is reset, the DDR1 and RDR1 registers are initialized to "00_H" and the Port 1 pin has high impedance.
- The PDR1 register is not initialized when the CPU is reset. To use the port in output mode, therefore, output mode must be specified in the DDR1 register after the output data is set in the PDR1 register.

❍ Port operation in stop or time-base timer mode

If the port switches to stop mode or timebase timer mode while the pin status setting bit (SPL) of the low power consumption mode control register (LPMCR) is set to "1", the pins come to have high impedance regardless of the value in the Port 1 direction register (DDR1). Note that the input buffer is forcibly shut off to prevent leakage due to an open circuit.

Note:

If a pull-up resistor is connected, setting the pin status setting bit (SPL) of the low power consumption mode control register (LPMCR) to "1" does not disconnect the pull-up resistor but holds the pin level at "H".

Table 8.4-4 States of Port 1 Pins

Pin	Normal operation	Sleep mode	Stop mode or time-base timer mode (SPL = 0)	Stop mode or time-base timer mode (SPL = 1, RDR = 0)	Stop mode or time-base timer mode (SPL = 1, RDR = 1)
P10 to P17	I/O port	I/O port	Input shut off/ output held	Input enabled/ output in Hi-z	Input shut down/ held at H level
INT0 to INT6	External interrupt input 0 to 6	External interrupt input 0 to 6			
FRCK	External clock input for free-running timer	External clock input for free-running timer			

SPL: Pin state specification bit of low-power consumption mode control register (LPMCR:SPL)
Hi-z: High impedance
Note: These pins serve as INT0 to INT6 and FRCK if they are used as resource functions.

8.5 Port 2

Port 2 is an I/O port that can also be used for resource input and output. Each of the port pins can be switched between a resource and an I/O port in accordance with the corresponding bit. This section describes the configuration of Port 2, a block diagram of pins, and registers mainly in terms of I/O ports.

■ Port 2 Configuration

Port 2 consists of the following elements:

- I/O pins/resource I/O pins (P20/TIN0 to P27/IN3)
- Port 2 data register (PDR2)
- Port 2 data direction register (DDR2)

■ Port 2 Pins

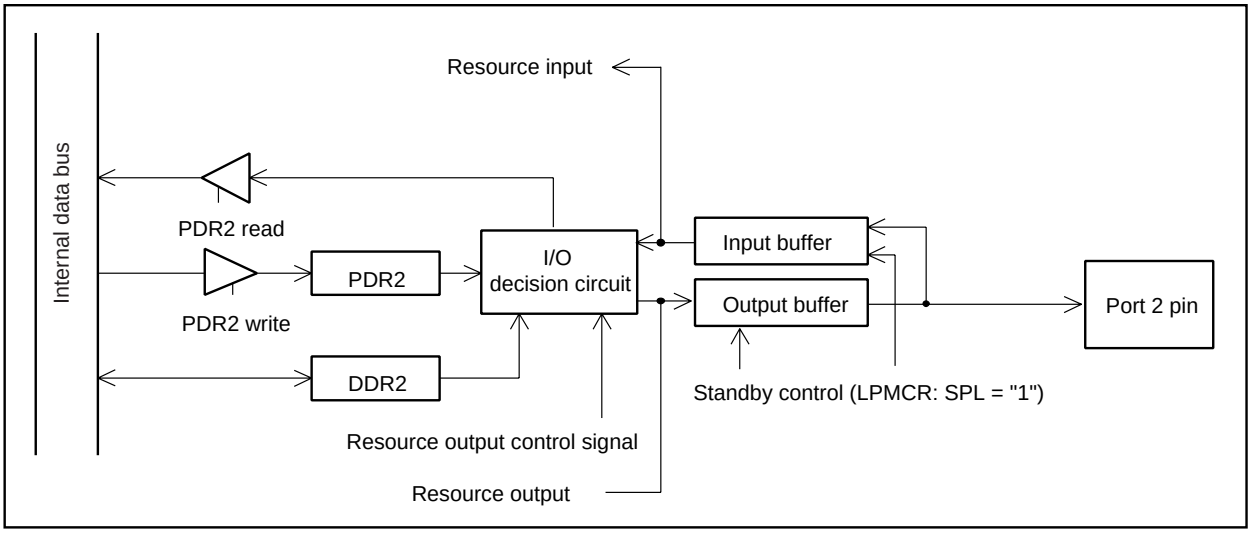
The port 2 I/O pins are also used as resource I/O pins. The pins cannot be used as I/O port pins when they are used as resource I/O pins.

Table 8.5-1 Port 2 Pins

I/O port name	Pin	Port function	Resource function		I/O form	
					Input	Output
Port 2	P20/TIN0	P20	TIN0	External clock input for reload timer 0	CMOS (hysteresis)	CMOS
	P21/TO0	P21	TO0	Event output for reload timer 0		
	P22/TIN1	P22	TIN1	External clock input for reload timer 1		
	P23/TO1	P23	TO1	Event output for reload timer 1		
	P24/IN0	P24	IN0	Input capture channel 0 input		
	P25/IN1	P25	IN1	Input capture channel 1 input		
	P26/IN2	P26	IN2	Input capture channel 2 input		
	P27/IN3	P27	IN3	Input capture channel 3 input		

■ Block Diagram of Port 2 Pins

Figure 8.5-1 Block Diagram of Port 2 Pins



■ Port 2 Registers

Port 2 registers are PDR2 and DDR2. The bits making up each register correspond to the port 2 pins on a one-to-one basis.

Table 8.5-2 Port 2 Pins and Their Corresponding Register Bits

I/O port name	Register bits and corresponding port pins								
Port 2	PDR2,DDR2	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
	Corresponding pin	P27	P26	P25	P24	P23	P22	P21	P20

8.5.1 Port 2 Registers (PDR2 and DDR2)

This section describes the port 2 registers.

■ Functions of Port 2 Registers

❍ **Port 2 data register (PDR2)**

The PDR2 register specifies the output value of each pin of Port 2.

❍ **Port 2 data direction register (DDR2)**

The DDR2 register specifies the I/O directions of pins of Port 2. A pin serves as an output port when a bit corresponding to the pin is set to "1". A pin serves as an input port when a bit corresponding to the pin is set to "0".

Table 8.5-3 Port 2 Register Functions

Register	Bit value	During reading		During writing		Address	Initial value
		Input port	Output port	Input port	Output port		
Port 2 data register (PDR2)	0	The pin is at the "L" level.	The corresponding bit of PDR2 register is set to "0".	"0" is written to the corresponding bit of PDR2 register.	The "L" level is output from the pin.	000002_H	$XXXXXXXX_B$
	1	The pin is at the "H" level.	The corresponding bit of PDR2 register is set to "1".	"1" is written to the corresponding bit of PDR2 register.	The "H" level is output from the pin.		
Port 2 data direction register (DDR2))	0	The corresponding bit of DDR2 register is set to "0".		The pin serves as an input port.		000012_H	00000000_B
	1	The corresponding bit of DDR2 register is set to "1".		The pin serves as an output port.			

X: Undefined

8.5.2 Operation of Port 2

This section describes the operation of port 2.

■ Operation of Port 2

❍ Setting Port 2 as an output port in the Port 2 direction register (DDR2)

- The value stored in the Port 2 data register (PDR2) is output to the Port 2 pin.
- If the PDR2 register is read, the value stored in PDR2 register is output.

❍ Setting Port 2 as an input port in the Port 2 direction register (DDR2)

- The Port 2 pin has high impedance.
- If the Port 2 data register (PDR2) is set to a value, the value stored in the PDR2 register is retained but not output to the pin.
- If the PDR2 register is read, the pin input level ("0" for "L" or "1" for "H") is output.

Note:

If a read-modify-write instruction (such as the bit set instruction) is used to access the PDR2 register, no bit specified for output in the DDR2 register is affected. For a bit specified for input in the DDR2 register, however, the pin input level is written to the PDR2 register. Therefore, to change a bit specified for input to output, first set an output value to the PDR2 register and then specify the DDR2 register as an output port.

❍ Port operation after a reset

- When the CPU is reset, the DDR2 register is initialized to "00_H" and the Port 2 pin has high impedance.
- The PDR2 register is not initialized when the CPU is reset. To use the PDR2 as an output port, first write an output value to the PDR2 register and then specify the DDR2 register as an output port.

❍ Port operation in stop or time-base timer mode

If the port switches to stop mode or timebase timer mode while the pin status setting bit (SPL) of the low power consumption mode control register (LPMCR) is set to "1", the pins come to have high impedance regardless of the value in the Port 2 direction register (DDR2). Note that the input buffer is forcibly shut off to prevent leakage due to an open circuit.

Table 8.5-4 States of Port 2 Pins

Pin	Normal operation	Sleep mode	Stop mode or time-base timer mode (SPL = 0)	Stop mode or time-base timer mode (SPL = 1)
P20 to P27	I/O port	I/O port	Input shut off/ output held	Input shut down/ output in Hi-z
TIN0, TIN1	External clock input for reload timers 0, 1	External clock input for reload timers 0, 1		
TO0, TO1	Event output for reload timers 0, 1	Event output for reload timers 0, 1		
IN0 to IN3	Trigger input for input captures 0 to 3	Trigger input for input captures 0 to 3		

SPL: Pin state specification bit of low-power consumption mode control register (LPMCR)
Hi-z: High impedance
Note: These pins serve as TIN0, TIN1, T00, T01, and IN0 to IN3 if they are used as resource functions.

8.6 Port 3

Port 3 is an I/O port that can also be used for resource input and output. Each of the port pins can be switched between a resource and an I/O port in accordance with the corresponding bit. This section describes the configuration of Port 3, a block diagram of pins, and registers mainly in terms of I/O ports.

■ Port 3 Configuration

Port 3 consists of the following:

- I/O pins/resource I/O pins (P30/RTO0 to P37/SOT0)
- Port 3 data register (PDR3)
- Port 3 data direction register (DDR3)

■ Port 3 Pins

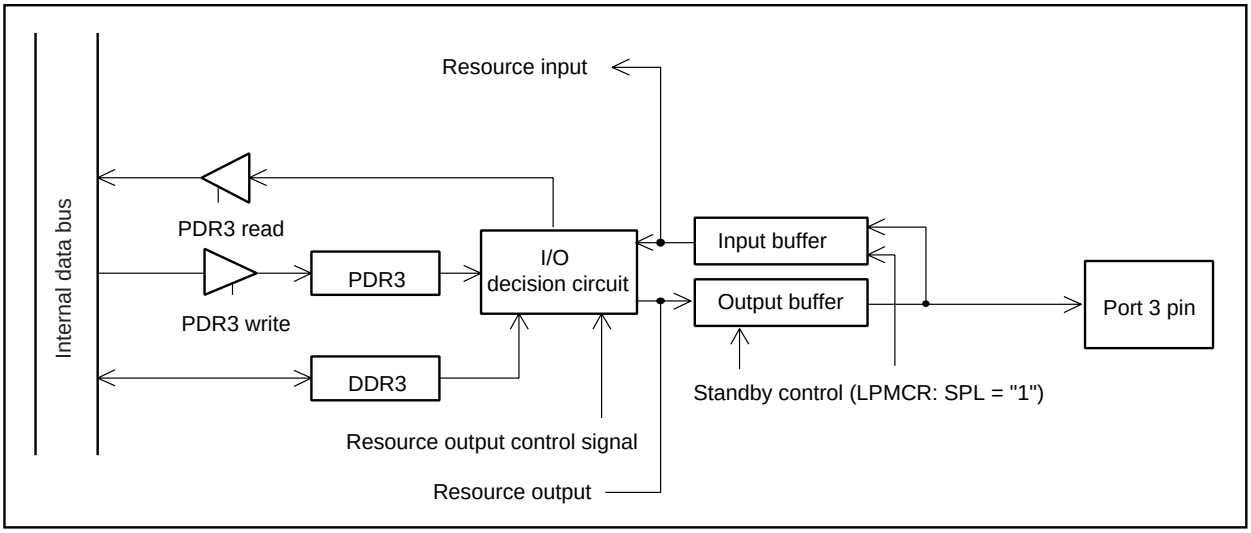
The port 3 I/O pins are also used as resource I/O pins. Therefore, the pins cannot be used as I/O port pins when they are used as resource I/O pins.

Table 8.6-1 Port 3 Pins

I/O port name	Pin	I/O port function	Resource function		I/O form	
					Input	Output
Port 3	P30/RTO0	P30	RTO0	External interrupt request input 0	CMOS (hysteresis)	CMOS
	P31/RTO1	P31	RTO1	External interrupt request input 1		
	P32/RTO2	P32	RTO2	External interrupt request input 2		
	P33/RTO3	P33	RTO3	External interrupt request input 3		
	P34/RTO4	P34	RTO4	External interrupt request input 4		
	P35/RTO5	P35	RTO5	External interrupt request input 5		
	P36/SIN0	P36	SIN0	UART0 serial data input		
	P37/SOT0	P37	SOT0	UART0 serial data output		

■ Block Diagram of Port 3 Pins

Figure 8.6-1 Block Diagram of Port 3 Pins



■ Port 3 Registers

Port 3 registers are PDR3 and DDR3. The bits making up each register correspond to the port 3 pins on a one-to-one basis.

Table 8.6-2 Port 3 Pins and Their Corresponding Register Bits

I/O port name	Register bits and corresponding port pins								
Port 3	PDR3, DDR3	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
	Corresponding pin	P37	P36	P35	P34	P33	P32	P31	P30

8.6.1 Port 3 Registers (PDR3 and DDR3)

This section describes the port 3 registers.

■ Functions of Port 3 Registers

❍ Port 3 data register (PDR3)

The PDR3 register specifies the output value of each pin of Port 3.

❍ Port 3 data direction register (DDR3)

The DDR3 register specifies the direction of a data flow (input or output) at each pin (bit) of port 3. When a DDR3 register bit is 1, the corresponding port (pin) is set as an output port. When the bit is 0, the port (pin) is set as an input port.

Table 8.6-3 Port 3 Register Functions

Register	Bit value	During reading		During writing		Address	Initial value
		Input port	Output port	Input port	Output port		
Port 3 data register (PDR3)	0	The pin is at the "L" level.	The corresponding bit of PDR3 register is set to "0".	"0" is written to the corresponding bit of PDR3 register.	The "L" level is output from the pin.	000003_H	$XXXXXXXX_B$
	1	The pin is at the "H" level.	The corresponding bit of PDR3 register is set to "1".	"1" is written to the corresponding bit of PDR3 register.	The "H" level is output from the pin.		
Port 3 data direction register (DDR3)	0	The corresponding bit of DDR3 register is set to "0".		The pin serves as an input port.		000013_H	00000000_B
	1	The corresponding bit of DDR3 register is set to "1".		The pin serves as an output port.			

X: Undefined

8.6.2 Operation of Port 3

This section describes the operation of port 3.

■ Operation of Port 3

❍ Setting Port 3 as an output port in the Port 3 direction register (DDR3)

- The value stored in the Port 3 data register (PDR3) is output to the Port 3 pin.
- If the PDR3 register is read, the value stored in PDR3 register is output.

❍ Setting Port 3 as an input port in the Port 3 direction register (DDR3)

- The Port 3 pin has high impedance.
- If the Port 3 data register (PDR3) is set to a value, the value stored in the PDR3 register is retained but not output to the pin.
- If the PDR3 register is read, the pin input level ("0" for "L" or "1" for "H") is output.

Note:

If a read-modify-write instruction (such as the bit set instruction) is used to access the PDR3 register, no bit specified for output in the DDR3 register is affected. For a bit specified for input in the DDR3 register, however, the pin input level is written to the PDR3 register. Therefore, to change a bit specified for input to output, first write an output value to the PDR3 register and then specify the DDR3 register as an output port.

❍ Port operation after a reset

- When the CPU is reset, the DDR3 register is initialized to "00_H" and the Port 3 pin has high impedance.
- The PDR3 register is not initialized when the CPU is reset. To use the PDR3 as an output port, first write an output value to the PDR3 register and then specify the DDR3 register as an output port.

❍ Port operation in stop or time-base timer mode

If the port switches to stop mode or timebase timer mode while the pin status setting bit (SPL) of the low power consumption mode control register (LPMCR) is set to "1", the pins come to have high impedance regardless of the value in the Port 3 direction register (DDR3). Note that the input buffer is forcibly shut off to prevent leakage due to an open circuit.

Table 8.6-4 States of Port 3 Pins

Pin	Normal operation	Sleep mode	Stop mode or time-base timer mode (SPL = 0)	Stop mode or time-base timer mode (SPL = 1)
P30 to P37	I/O port	I/O port	Input shut off/ output held	Input shut down/ output in Hi-z
RT00 to RT05	Event output 0 to 5 and waveform generation block output 0 to 5 for output compare	Event output 0 to 5 and waveform generation block output 0 to 5 for output compare		
SIN0	UART0 serial data input	UART0 serial data input		
SOT0	UART0 serial data output	UART0 serial data output		

SPL: Pin state specification bit of low-power consumption mode control register (LPMCR:SPL)
Hi-z: High impedance
Note: These pins serve as RT00 to RT05, SIN0, and S0T0 if they are used as resource functions.

8.7 Port 4

Port 4 is an I/O port that can also be used for resource input and output. Each of the port pins can be switched between a resource and an I/O port in accordance with the corresponding bit. This section describes the configuration of Port 4, a block diagram of pins, and registers mainly in terms of I/O ports.

■ Port 4 Configuration

Port 4 consists of the following:

- I/O pins/resource I/O pins (P40/SCK0 to P46/PPG5)
- Port 4 data register (PDR4)
- Port 4 data direction register (DDR4)

■ Port 4 Pins

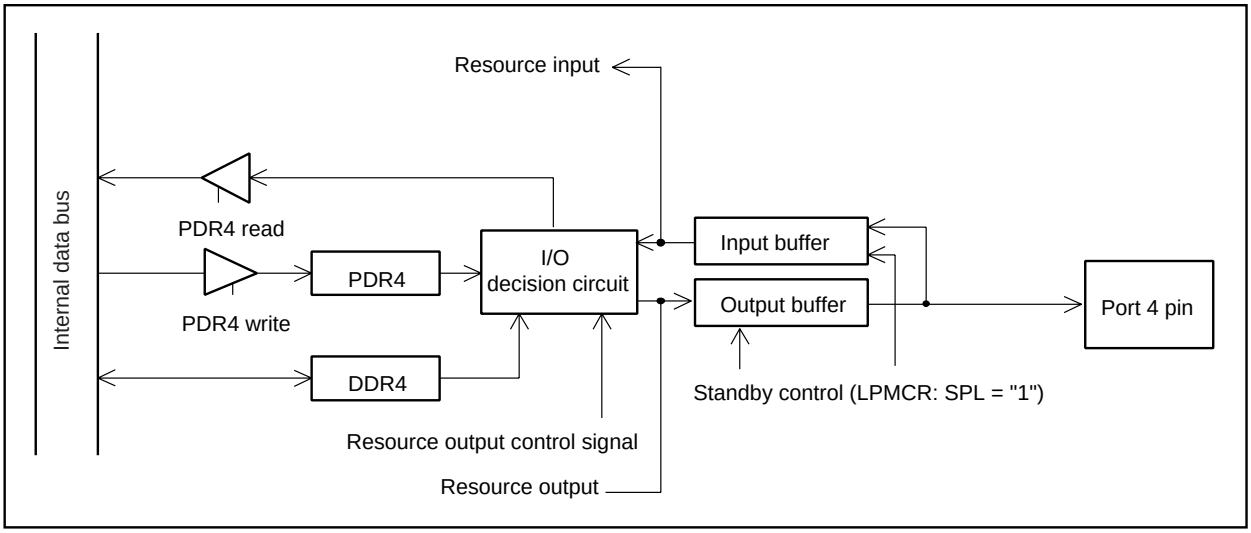
The port 4 I/O pins are also used as resource I/O pins. The pins cannot be used as I/O port pins when they are used as resource I/O pins.

Table 8.7-1 Port 4 Pins

I/O port name	Pin	I/O port function	Resource function		I/O form	
					Input	Output
Port 4	P40/SCK0	P40	SCK0	UART0 serial clock I/O	CMOS (hysteresis)	CMOS
	P41/PPG0	P41	PPG0	PPG0 output		
	P42/PPG1	P42	PPG1	PPG1 output		
	P43/PPG2	P43	PPG2	PPG2 output		
	P44/PPG3	P44	PPG3	PPG3 output		
	P45/PPG4	P45	PPG4	PPG4 output		
	P46/PPG5	P46	PPG5	PPG5 output		

■ Block Diagram of Port 4 Pins

Figure 8.7-1 Block Diagram of Port 4 Pins



Note:

When the resource output enable bit is set, the port is forcibly caused to function as resource output pins regardless of the value in the DDR4 register.

■ Port 4 Registers

Port 4 registers are PDR4 and DDR4. The bits making up each register correspond to the port 4 pins on a one-to-one basis.

Table 8.7-2 Port 4 Pins and Their Corresponding Register Bits

I/O port name	Register bits and corresponding port pins								
Port 4	PDR4,DDR4	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
	Corresponding pin	-	P46	P45	P44	P43	P42	P41	P40

8.7.1 Port 4 Registers (PDR4 and DDR4)

This section describes the port 4 registers.

■ Functions of Port 4 Registers

❍ **Port 4 data register (PDR4)**

The PDR4 register specifies the output value of each pin of Port 4.

❍ **Port 4 data direction register (DDR4)**

The DDR4 register specifies the I/O directions of pins of Port 4. A pin serves as an output port when a bit corresponding to the pin is set to "1". A pin serves as an input port when a bit corresponding to the pin is set to "0".

Reference:

- To use a port pin as an output pin for a resource function, set the resource output enable bit corresponding to the output pin to Enabled. The pin can be used as an output pin for a resource function regardless of the setting value of the Port 4 direction register (DDR4).
- To use a port pin as an input pin for a resource function, set the DDR4 bit corresponding to the input pin "0" to set it as an input port.

Table 8.7-3 Port 4 Register Functions

Register	Bit value	During reading		During writing		Address	Initial value
		Input port	Output port	Input port	Output port		
Port 4 data register (PDR4)	0	The pin is at the "L" level.	The corresponding bit of PDR4 register is set to "0".	"0" is written to the corresponding bit of PDR4 register.	The "L" level is output from the pin.	000004_H	$XXXXXXXX_B$
	1	The pin is at the "H" level.	The corresponding bit of PDR4 register is set to "1".	"1" is written to the corresponding bit of PDR4 register.	The "H" level is output from the pin.		
Port 4 data direction register (DDR4)	0	The corresponding bit of DDR4 register is set to "0".		The pin serves as an input port.		000014_H	$X0000000_B$
	1	The corresponding bit of DDR4 register is set to "1".		The pin serves as an output port.			

X: Undefined

8.7.2 Operation of Port 4

This section describes the operation of port 4.

■ Operation of Port 4

❍ **Setting Port 4 as an output port in the Port 4 direction register (DDR4)**

- The value stored in the Port 4 data register (PDR4) is output to the Port 4 pin.
- If the PDR4 register is read, the value stored in PDR4 register is output.

❍ **Setting Port 4 as an input port in the Port 4 direction register (DDR4)**

- The Port 4 pin has high impedance.
- If the Port 4 data register (PDR4) is set to a value, the value stored in the PDR4 register is retained but not output to the pin.
- If the PDR4 register is read, the pin input level ("0" for "L" or "1" for "H") is output.

Note:

If a read-modify-write instruction (such as the bit set instruction) is used to access the PDR4 register, no bit specified for output in the DDR4 register is affected. For a bit specified for input in the DDR4 register, however, the pin input level is written to the PDR4 register. Therefore, to change a bit specified for input to output, first write an output value to the PDR4 register and then specify the DDR4 register as an output port.

❍ **Port operation for resource output**

The resource output enable bit is set to enable the port to be used for resource output. The state of the resource enable bit takes precedence when specifying a switch between input and output. Even if a DDR4 register bit is 0, the corresponding port pin is used for resource output if the resource has been enabled for output. If the DDR4 bit corresponding to the pin that can also be used for resource output is set to "0", set the resource output to Enabled to read the out put value of the resource.

❍ **Port operation for resource input**

To use for resource input the I/O port that can also be used for resource input, set the DDR4 bit corresponding to the pin to "0".

❍ **Port operation after a reset**

- When the CPU is reset, the DDR4 register is initialized to "00_H" and the Port 4 pin has high impedance.
- The PDR4 register is not initialized when the CPU is reset. To use the PDR4 as an output port, first write an output value to the PDR4 register and then specify the DDR4 register as an output port.

❍ **Port operation in stop or time-base timer mode**

If the port switches to stop mode or timebase timer mode while the pin status setting bit (SPL) of the low power consumption mode control register (LPMCR) is set to "1", the pins come to have high impedance regardless of the value in the Port 4 direction register (DDR4). Note that the input buffer is forcibly shut off to prevent leakage due to an open circuit.

Table 8.7-4 States of Port 4 Pins

Pin	Normal operation	Sleep mode	Stop mode or time-base timer mode (SPL = 0)	Stop mode or time-base timer mode (SPL = 1)
P40 to P46	I/O port	I/O port	Input shut off/ output held	Input shut down/ output in Hi-z
SCK0	UART0 serial clock I/O	UART0 serial clock I/O		
PPG0 to 5	PPG output	PPG output		

SPL: Pin state specification bit of low-power consumption mode control register (LPMCR:SPL)
Hi-z: High impedance
Note: These pins serve as SCK0 and PPG0 to PPG5 if they are used as resource functions.

8.8 Port 5

Port 5 is an I/O port that can also be used for A/D converter analog input. Each of the port pins can be switched between analog input and an I/O port in accordance with the corresponding bit. This section describes the configuration of Port 5, a block diagram of pins, and registers mainly in terms of I/O ports.

■ Port 5 Configuration

Port 5 consists of the following:

- Analog input pin of the I/O port pin or A/D converter (P50/AN0 to P57/AN7)
- Port 5 data register (PDR5)
- Port 5 data direction register (DDR5)
- Analog input enable register (ADER)

■ Port 5 Pins

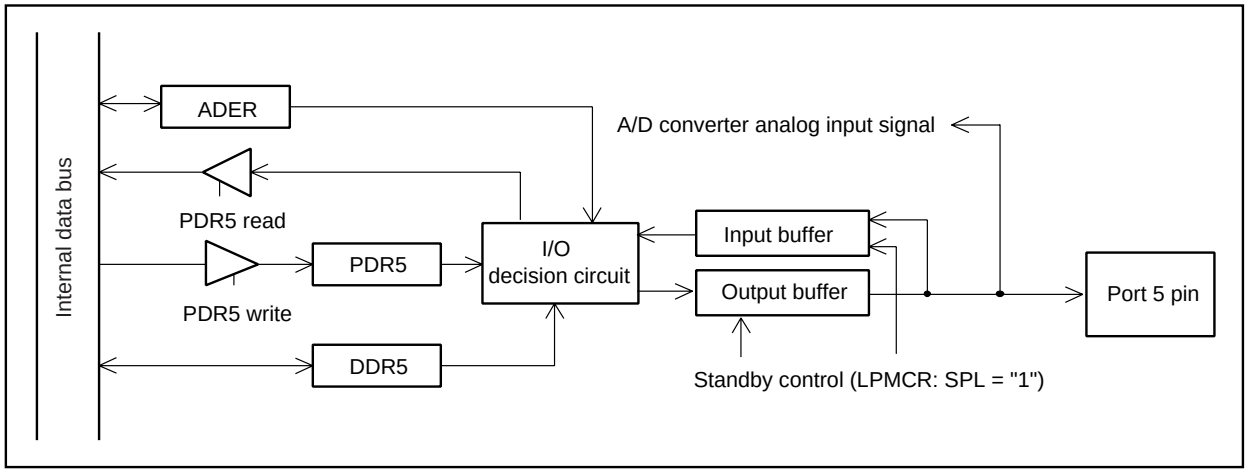
The I/O pins of Port 5 also serve as analog input of the A/D converter. The I/O pins, when used as analog input pins of the A/D converter, cannot be used as I/O ports. Conversely, the I/O pins, when used as I/O ports, cannot be used as analog input pins of the A/D converter.

Table 8.8-1 Port 5 Pins

I/O port name	Pin	Port function	Resource function		I/O form	
					Input	Output
Port 5	P50/AN0	P50	AN0	A/D converter analog input 0	CMOS (hysteresis)	CMOS
	P51/AN1	P51	AN1	A/D converter analog input 1		
	P52/AN2	P52	AN2	A/D converter analog input 2		
	P53/AN3	P53	AN3	A/D converter analog input 3		
	P54/AN4	P54	AN4	A/D converter analog input 4		
	P55/AN5	P55	AN5	A/D converter analog input 5		
	P56/AN6	P56	AN6	A/D converter analog input 6		
	P57/AN7	P57	AN7	A/D converter analog input 7		

■ Block Diagram of Port 5 Pins

Figure 8.8-1 Block Diagram of Port 5 Pins



Note:

To use a port pin as an input port, set the corresponding bit of the Port 5 direction register (DDR5) to "0" and the corresponding bit of the analog input enable register (ADER) to "0".

To use a port pin as an analog input pin, set the corresponding bit of the DDR5 register to "0" and the corresponding bit of the ADER register to "1".

■ Port 5 Registers

Port 5 registers are PDR5, DDR5, and ADER. The bits making up each register correspond to the port 5 pins on a one-to-one basis.

Table 8.8-2 Port 5 Pins and Their Corresponding Register Bits

I/O port name	Register bits and corresponding port pins								
Port 5	PDR5,DDR5,ADER	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
	Corresponding pin	P57	P56	P55	P54	P53	P52	P51	P50

8.8.1 Port 5 Registers (PDR5, DDR5, and ADER)

This section describes the port 5 registers.

■ Functions of Port 5 Registers

❍ **Port 5 data register (PDR5)**

The PDR5 register specifies the output value of each pin of Port 5.

❍ **Port 5 data direction register (DDR5)**

The DDR5 register specifies the direction of a data flow (input or output) at each pin (bit) of port 5. When a DDR5 register bit is 1, the corresponding port (pin) is set as an output port. When the bit is 0, the port (pin) is set as an input port.

❍ **Analog input enable register (ADER)**

Each bit of the ADER register can be set as an I/O port or analog input of the A/D converter. A pin serves as analog input if the bit corresponding to the port (pin) is set to "1", or an I/O port if the bit is set to "0".

Note:

If a pin is set as an I/O port, the input of a signal at an intermediate level causes an input leakage current. To use a pin for analog input, be sure to set the corresponding bit of the ADER register as analog input of the A/D converter.

Reference:

A reset initializes the DDR5 register to "00_H" and the ADER register to "FF_H" and sets analog input of the A/D converter.

Table 8.8-3 "Port 5 Register Functions" lists the functions of the port 5 registers.

Table 8.8-3 Port 5 Register Functions

Register	Bit value	During reading		During writing		Address	Initial value
		Input port	Output port	Input port	Output port		
Port 5 data register (PDR5)	0	The pin is at the "L" level.	The corresponding bit of PDR5 register is set to "0".	"0" is written to the corresponding bit of PDR5 register.	The "L" level is output from the pin.	000005_H	$XXXXXXX_B$
	1	The pin is at the "H" level.	The corresponding bit of PDR5 register is set to "1".	"1" is written to the corresponding bit of PDR5 register.	The "H" level is output from the pin.		
Port 5 data direction register (DDR5)	0	The corresponding bit of DDR5 register is set to "0".		The pin serves as an input port.		000015_H	00000000_B
	1	The corresponding bit of DDR5 register is set to "1".		The pin serves as an output port.			
Analog input enable register (ADER)	0	The corresponding bit of ADER register is set to "0".		I/O port		000017_H	11111111_B
	1	The corresponding bit of ADER register is set to "1".		Analog input of A/D converter			

X: Undefined

8.8.2 Operation of Port 5

This section describes the operation of port 5.

■ Operation of Port 5

❍ **Setting Port 5 as an output port in the Port 5 direction register (DDR5) and the analog input enable register (ADER)**

- The value stored in the Port 5 data register (PDR5) is output to the Port 5 pin.
- If the PDR5 register is read, the value stored in PDR5 register is output.

❍ **Setting Port 5 as an input port in the Port 5 direction register (DDR5) and the analog input enable register (ADER)**

- The Port 5 pin has high impedance.
- If the Port 5 data register (PDR5) is set to a value, the value stored in the PDR5 register is retained but not output to the pin.
- If the PDR5 register is read, the pin input level ("0" for "L" or "1" for "H") is output.

Note:

If a read-modify-write instruction (such as the bit set instruction) is used to access the PDR5 register, no bit specified for output in the DDR5 register is affected. For a bit specified for input in the DDR5 register, however, the pin input level is written to the PDR5 register. Therefore, to change a bit specified for input to output, first write an output value to the PDR5 register and then specify the DDR5 register as an output port.

❍ **Setting Port 5 as analog input of the A/D converter**

To use a port pin as analog input of the A/D converter, set the bit of the ADER register corresponding to the analog input pin of the A/D converter to "1". If a pin is set for analog input of the A/D converter, read a corresponding bit of the PDR5 register to obtain the read value of "0".

❍ **Port operation as resource output**

To use a port pin as resource output, set the bit of the ADER register corresponding to the resource output pin to "0" and the output enable bit of the resource to be used to Enabled. Even if a pin that also serves as the DDR5 resource pin is set to "0", the resource can be output as long as the resource output is enabled because resource output gets higher priority than port output. If a pin that also serves as the DDR5 resource pin is set to "0", enable the resource output to read the output value of the resource.

❍ **Port operation as resource input**

To use for resource input the I/O port pin that can also be used for resource input, set the corresponding bit of the ADER register to "0" and the corresponding bit of the DDR5 register to "0".

❍ **Port operation after a reset**

- When the CPU is reset, the DDR5 register is initialized to "00_H", the ADER register is initialized to "FF_H", and the register pin is set for analog input of the A/D converter. To use the register pin as an I/O port, set the ADER register to "00_H" to enable port I/O mode.
- The PDR5 register is not initialized when the CPU is reset. To use the PDR5 as an output port, first write an output value to the PDR5 register and then specify the DDR5 register as an output port.

❍ **Port operation in stop or time-base timer mode**

If the port switches to stop mode or timebase timer mode while the pin status setting bit (SPL) of the low power consumption mode control register (LPMCR) is set to "1", the pins come to have high impedance regardless of the value in the Port 5 direction register (DDR5). Note that the input buffer is forcibly shut off to prevent leakage due to an open circuit.

Table 8.8-4 States of Port 5 Pins

Pin	Normal operation	Sleep mode	Stop mode or time-base timer mode (SPL = 0)	Stop mode or time-base timer mode (SPL = 1)
P50 to P57	I/O port	I/O port	Input shut off/ output held	IInput shut down/ output in Hi-z
AN0 to AN7	Analog input of A/D converter 0 to 7	Analog input of A/D converter 0 to 7		

SPL:Pin state specification bit of low-power consumption mode control register (LPMCR:SPL)
Hi-z: High impedance
Note: These pins serve as AN0 to AN7 if they are used as resource functions.

8.9 Port 6

Port 6 is an I/O port that can also be used for resource I/O. Each of the port pins can be switched between a resource and an I/O port in accordance with the corresponding bit. This section describes the configuration of Port 6, a block diagram of pins, and registers mainly in terms of I/O ports.

■ Port 6 Configuration

Port 6 consists of the following:

- I/O pins/resource I/O pins (P60/SIN1 to P63/INT7/DTTI)
- Port 6 data register (PDR6)
- Port 6 data direction register (DDR6)

■ Port 6 Pins

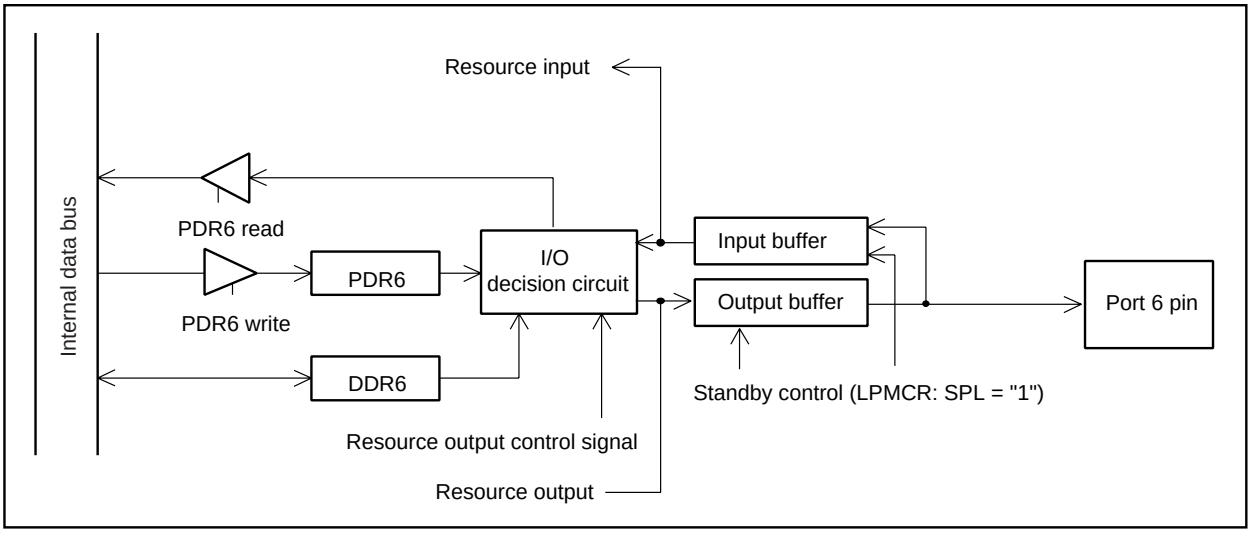
The port 6 I/O pins are also used as resource I/O pins. The pins cannot be used as general-purpose I/O port pins when they are used as resource I/O pins.

Table 8.9-1 Port 6 Pins

I/O port name	Pin	Port function	Resource function		I/O form	
					Input	Output
Port 6	P60/SIN1	P60	SIN1	UART1 serial data input	CMOS (hysteresis)	CMOS
	P61/SOT1	P61	SOT1	UART1 serial data output		
	P62/SCK1	P62	SCK1	UART1 serial clock I/O		
	P63/INT7	P63	INT7	External interrupt input 7		

■ Block Diagram of Port 6 Pins

Figure 8.9-1 Block Diagram of Port 6 Pins



Note:

If the resource output enable bit is set to Enabled, the register pin forcibly serves as a resource output pin regardless of the value in the Port 6 direction register (DDR6).

■ Port 6 Registers

Port 6 registers are PDR6 and DDR6. The bits making up each register correspond to the port 6 pins on a one-to-one basis.

Table 8.9-2 Port 6 Pins and Their Corresponding Register Bits

I/O port name	Register bits and corresponding port pins								
Port 6	PDR6,DDR6	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
	Corresponding pin	-	-	-	-	P63	P62	P61	P60

8.9.1 Port 6 Registers (PDR6 and DDR6)

This section describes the port 6 registers.

■ Functions of Port 6 Registers

❍ **Port 6 data register (PDR6)**

The PDR6 register specifies the output value of each pin of Port 6.

❍ **Port 6 data direction register (DDR6)**

The DDR6 register specifies the direction of a data flow (input or output) at each pin (bit) of port 6. When a DDR6 register bit is 1, the corresponding port (pin) is set as an output port. When the bit is 0, the port (pin) is set as an input port.

Reference:

- To use a port pin as an output pin of the resource function, set the resource output enable bit corresponding to the output pin to Enabled. Then, the pin can be used as an output pin of the resource function regardless of the setting value in the DDR6 register.
- To use a port pin as an input pin of the resource function, set the DDR6 bit corresponding to the input pin to "0" to set it as an input port.

Table 8.9-3 Port 6 Register Functions

Register	Bit value	During reading		During writing		Address	Initial value
		Input port	Output port	Input port	Output port		
Port 6 data register (PDR6)	0	The pin is at the "L" level.	The corresponding bit of PDR6 register is set to "0".	"0" is written to the corresponding bit of PDR6 register.	The "L" level is output from the pin.	000006_H	$XXXXXXXX_B$
	1	The pin is at the "H" level.	The corresponding bit of PDR6 register is set to "1".	"1" is written to the corresponding bit of PDR6 register.	The "H" level is output from the pin.		
Port 6 data direction register (DDR6)	0	The corresponding bit of DDR6 register is set to "0".		The pin serves as an input port.		000016_H	$XXXX0000_B$
	1	The corresponding bit of DDR6 register is set to "1".		The pin serves as an output port.			

X: Undefined

8.9.2 Operation of Port 6

This section describes the operation of port 6.

■ Operation of Port 6

❍ **Setting Port 6 as an output port in the Port 6 direction register (DDR6)**

- The value stored in the Port 6 data register (PDR6) is output to the Port 6 pin.
- If the PDR6 register is read, the value stored in PDR6 register is output.

❍ **Setting Port 6 as an input port in the Port 6 direction register (DDR6)**

- The Port 6 pin has high impedance.
- If the Port 6 data register (PDR6) is set to a value, the value stored in the PDR6 register is retained but not output to the pin.
- If the PDR6 register is read, the pin input level ("0" for "L" or "1" for "H") is output.

Note:

If a read-modify-write instruction (such as the bit set instruction) is used to access the PDR6 register, no bit specified for output in the DDR6 register is affected. For a bit specified for input in the DDR6 register, however, the pin input level is written to the PDR6 register. Therefore, to change a bit specified for input to output, first write an output value to the PDR6 register and then specify the DDR6 register as an output port.

❍ **Port operation for resource output**

To use a port pin as resource output, set the output enable bit of the resource to be used to Enabled. Even if a pin that also serves as the DDR6 resource pin is set to "0", the resource can be output as long as the resource output is enabled because resource output gets higher priority than port output. If a pin that also serves as the DDR6 resource pin is set to "0", set the resource output to Enabled to read the output value of the resource

❍ **Port operation for resource input**

To use for resource input the I/O port pin that can also be used for resource input, set the corresponding bit of the DDR6 register to "0".

❍ **Port operation after a reset**

- When the CPU is reset, the DDR6 register is initialized to "00_H" and the Port 6 pin has high impedance.
- The PDR6 register is not initialized when the CPU is reset. To use the port in output mode, therefore, output mode must be specified in the DDR6 register after output data is set in the PDR6 register.

❍ **Port operation in stop or time-base timer mode**

If the port switches to stop mode or timebase timer mode while the pin status setting bit (SPL) of the low power consumption mode control register (LPMCR) is set to "1", the pins come to have high impedance regardless of the value in the Port 6 direction register (DDR6). Note that the input buffer is forcibly shut off to prevent leakage due to an open circuit.

Table 8.9-4 States of Port 6 Pins

Pin	Normal operation	Sleep mode	Stop mode or time-base timer mode (SPL = 0)	Stop mode or time-base timer mode (SPL = 1)
P60 to P63	I/O port	I/O port	Input shut off/output held	Input shut off/output in Hi-z
SIN1	UART1 serial data input	UART1 serial data input		
SOT1	UART1 serial data output	UART1 serial data output		
SCK1	UART1 serial clock I/O	UART1 serial clock I/O		
INT7	External interrupt request 7	External interrupt request 7	Input shut off/output held (Input enabled if external interrupt enabled)	Input shut off/output in Hi-z (Input enabled if external interrupt enabled)

SPL:Pin state specification bit of low-power consumption mode control register (LPMCR)
Hi-z:High impedance
Note: These pins serve as SIN1, S0T1, SCK1, and INT7 if they are used as resource functions.

8.10 Sample I/O Port Program

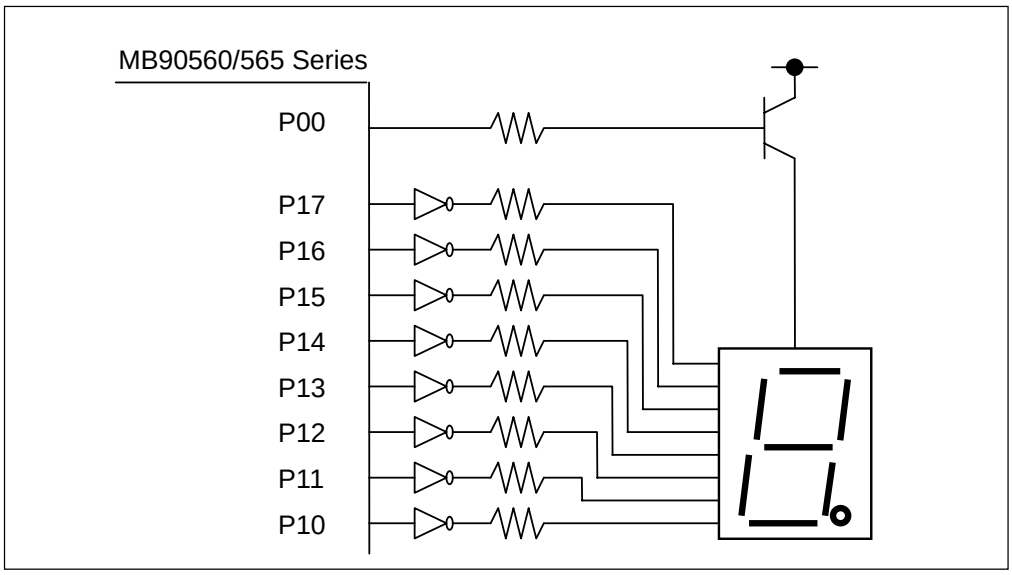
This section provides a sample program using I/O port pins.

■ Sample I/O Port Program

❍ Processing specifications

- Ports 0 and 1 are used to turn on all segments of a seven-segment (eight-segment if the decimal point is included) LED.
- Pin P00 corresponds to the anode common pin of the LED, and pins P10 to P17 correspond to the segment pins.

Figure 8.10-1 Example of Eight-segment LED Connection



❍ Coding example

```
PDR0  EQU   000000H
PDR1  EQU   000001H
DDR0  EQU   000010H
DDR1  EQU   000011H
;------Main program-------------------------------------------------
CODE         CSEG
START:
                              ;Initialization
      MOV   I:PDR0,#00000000B ;Puts P00 at a low level (#xxxxxxx0B).
      MOV   I:DDR0,#11111111B ;Puts all port 0 bits in output mode.
      MOV   I:PDR1,#11111111B ;Sets all port 1 bits to 1.
      MOV   I:DDR1,#11111111B ;Puts all port 1 bits in output mode.
CODE  ENDS
;---------------------------------------------------------------------
      END   START
```

CHAPTER 9 TIMEBASE TIMER

This chapter describes the functions and operation of the timebase timer.

9.1 Overview of the Timebase Timer

The timebase timer is an 18-bit free-running counter that counts up in synchronization with the main clock. The timer has two functions: An interval timer function that can select one of four intervals and a function for supplying clocks to the oscillation stabilization interval timer and the watchdog timer.

■ Interval Timer Function

The interval timer function repeatedly generates an interrupt request at a given interval.

- An interrupt request is generated when the interval timer bit for the timebase counter overflows.
- The interval timer bit (interval) can be selected from four types.

Table 9.1-1 Intervals for the Timebase Timer

Main clock cycle	Interval cycle
2/HCLK (0.5 µs)	2^{12}/HCLK (Approx. 1.0 ms)
	2^{14}HCLK (Approx. 4.1 ms)
	2^{16}/HCLK (Approx. 16.4 ms)
	2^{19}/HCLK (Approx. 131.1 ms)

HCLK: Oscillation clock frequency
Values in parentheses are for a 4 MHz oscillation clock.

■ Clock Supply Function

The clock supply function supplies clocks to the oscillation settling time timer and to some peripheral functions.

Table 9.1-2 Clock Cycle Time Supplied from the Timebase Timer

Clock destination	Clock cycle time	Remarks
Oscillation setting time	2^{13}/HCLK (Approx. 2.0 ms)	Oscillation settling time for ceramic vibrator
	2^{15}/HCLK (Approx. 8.2 ms)	Oscillation settling time for crystal vibrator
	2^{17}/HCLK (Approx. 32.8 ms)	
Watchdog timer	2^{12}/HCLK (Approx. 1.0 ms)	Count-up clock for watchdog timer
	2^{14}/HCLK (Approx. 4.1 ms)	
	2^{16}/HCLK (Approx. 16.4 ms)	
	2^{19}/HCLK (Approx. 131.1 ms)	

HCLK: Oscillation clock frequency
Values in parentheses occurs during operation of the 4 MHz oscillation clock.

Reference:

The oscillation settling time is the yardstick because the oscillation cycle time is unstable as soon as oscillation starts.

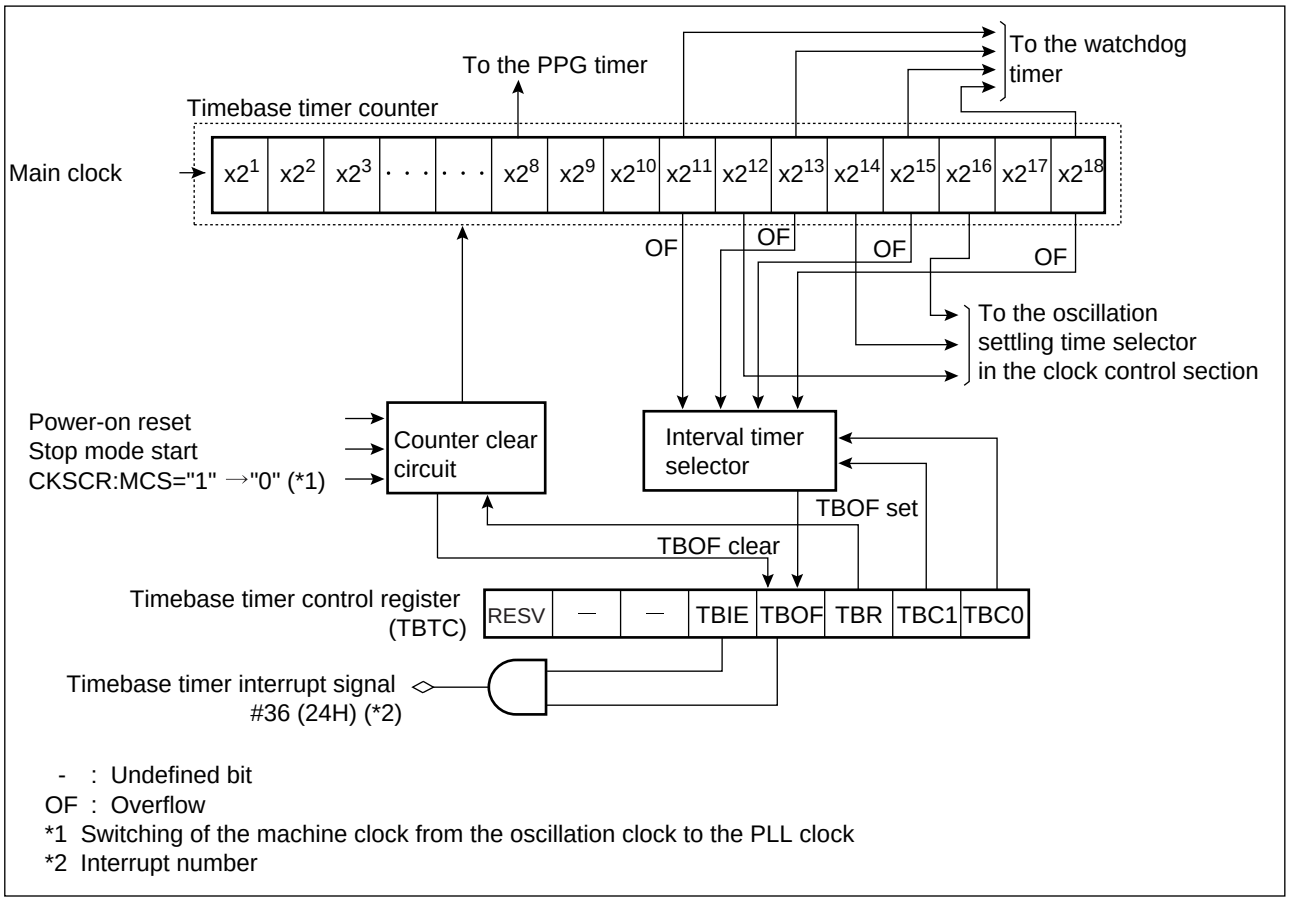
9.2 Configuration of the Timebase Timer

The timebase timer consists of the following four blocks:

- **Timebase timer counter**
- **Counter clear circuit**
- **Interval timer selector**
- **Timebase timer control register (TBTC)**

■ Block Diagram of the Timebase Timer

Figure 9.2-1 Block Diagram of the Timebase Timer



❍ Timebase timer counter

An 18-bit up counter that uses the main clock as the count clock

❍ Counter clear circuit

Clears the timebase timer counter by setting the timebase timer initialization bit (TBR) of the timebase timer control register (TBTC) to "0", performing a power-on reset, sending the CPU into stop mode (LPMCR: STP = "1"), and changing the machine clock from the main clock to the PLL clock (CKSCR: MCS = "1" -> "0").

- **Interval timer selector**

  Selects one of four outputs of the timebase timer counter. An overflow of the selected bit becomes an interrupt cause.

- **Timebase timer control register (TBTC)**

  Selects the interval, clears the timebase timer counter, controls an interrupt request, and checks the status.

9.3 Timebase Timer Control Register (TBTC)

The timebase timer control register (TBTC) selects the interval, clears the timebase timer counter, controls an interrupt request, and checks the status.

■ Timebase Timer Control Register (TBTC)

Figure 9.3-1 Timebase Timer Control Register (TBTC)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0000A9$_H$	RESV	—	—	TBIE	TBOF	TBR	TBC1	TBC0	1XX00100$_B$
	R/W	-	-	R/W	R/W	W	R/W	R/W	

TBC1	TBC0	Interval selection bit
0	0	2^{12}/HCLK (Approx. 1.0 ms)
0	1	2^{14}/HCLK (Approx. 4.1 ms)
1	0	2^{16}/HCLK (Approx. 16.4 ms)
1	1	2^{19}/HCLK (Approx. 131 ms)

Values in parentheses are for a 4 MHz oscillation clock.

TBR	Timebase timer initialization bit
0	Clearing of the timebase timer counter and an interrupt request
1	No effect on operation

TBOF	Interrupt request flag bit	
	During reading	During writing
0	No interrupt request made	Clearing of an interrupt request
1	Interrupt request made	No effect on operation

TBIE	Interrupt request enable bit
0	Interrupt request output disabled
1	Interrupt request output enabled

RESV	Reserved bit
Be sure to write 1 to this bit.	

R/W : Read/write
W : Write only
- : Not used
x : Undefined
□ : Initial value
HCLK : Oscillation clock frequency

Table 9.3-1 Function Description of Each Bit in the Timebase Timer Control Register (TBTC)

Bit name		Function
bit15	RESV: Reserved bit	• Be sure to write 1 to this bit.
bit14 bit13	Not used	• When read, the value is undefined. • Writing has no effect on operation.
bit12	TBIE: Interrupt request enable bit	• This bit enables interrupt requests. • When this bit and the interrupt request flag bit (TBOF) are 1, an interrupt request is output.
bit11	TBOF: Interrupt request flag bit	• TBOF is a flag bit for an interrupt request. • This bit is set to "1" when the interval timer bit selected for the timebase timer counter overflows. • When the interrupt request enable bit (TBIE) is set to "1", an interrupt request is output. • Set this bit to "0" to clear an interrupt request. • When this bit is set to "1", there is no effect on operation. **Note:Note:** • To set this bit to "0", set the interrupt request enable bit (TBIE) or the interrupt level mask register (ILM) of the processor status (PS) to Disabled. • This bit is cleared to "0" when a transition to stop mode occurs, the timebase timer is cleared due to the timebase timer initialization bit (TBR), or a reset occurs.
bit10	TBR: Timebase timer initialization bit	• Used to clear the timebase timer counter. • When 0 is written to this bit, the counter is cleared and the TBOF bit is cleared. If 1 is written, the bit does not change and there is no effect. **[Reference]** The read value is always 1.
bit9 bit8	TBC1, TBC0: Interval selection bit	• Used to select an interval timer cycle. • The bit for the interval timer of the timebase timer counter is specified. • Four types of interval can be selected.

9.4 Timebase Timer Interrupts

The timebase timer can output an interrupt request when the bit specifying the interval timer overflows (Interval timer function).

■ Timebase Timer Interrupts

The interrupt request flag bit (TBOF) of the timebase timer control register (TBTC) is set to "1" when the timebase timer counter counts up on the main clock and the specified interval timer overflows. If the TBOF bit is set to "1" while the interrupt request enable bit is set to Enabled (TBTC: TBIE = "1"), an interrupt request (interrupt number #36) is output and the interrupt processing routine is executed. In the interrupt processing routine, set the TBOF bit to "0" to clear the interrupt request. When the specified interval timer bit overflows, the TBOF bit is set to "1" regardless of the value in the TBIE bit.

Note:

Clear the TBOF bit of the TBTC register to "0" while the TBIE bit or the ILM register of the processor status (PS) is set to Disabled.

Reference:

- The timebase timer cannot use the extended intelligent I/O service (EI^2OS).

■ Timebase Timer Interrupts and EI^2OS

Table 9.4-1 Timebase Interrupts and EI^2OS

Interrupt number	Interrupt level setting register		Vector table address			EI^2OS
	Register name	Address	Lower	Upper	Bank	
#36 (24_H)	ICR12	$0000BC_H$	$FFFF6C_H$	$FFFF6D_H$	$FFFF6E_H$	x

x: Not available

Note:

ICR12 is used both for timebase timer interrupts and input capture channel 2 and channel 3 interrupts. Both of these interrupts can be used but have the same interrupt level.

9.5 Operation of the Timebase Timer

This section describes the operation of the interval timer function, the oscillation stabilization interval timer function, and the clock supply function.

■ Operation of the Interval Timer Function (Timebase Timer)

The interval timer function generates an interrupt request for each interval

The stabilization in Figure 9.5-1 "Stabilization of the Timebase Timer" is required to all the timer to operate as an interval timer.

Figure 9.5-1 Setting of the Timebase Timer

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
0000A9H	RESV	—	—	TBIE	TBOF	TBR	TBC1	TBC0
	1	-	-	◎	0	0	◎	◎

◎ : Used
0 : Set 0.
1 : Set 0.
- : Undefined bit

- The timebase timer continues counting up as long as the clock oscillates.
- If the timebase timer counter is cleared (TBTC: TBR = "0"), the counter counts up from "0000000000000000_B". When the specified interval timer bit overflows, the interrupt request flag bit (TBOF) of the timebase timer control register (TBTC) is set to "1". If the overflow occurs while the interrupt request enable bit is set to Enabled (TBIE = "1"), interrupt requests are output at every specified interval based on the time when the counter was cleared.
- The interval may become longer than the time set because of timebase timer clearing.

■ Oscillation Stabilization Wait Interval Timer Function

The timebase timer is also used as the oscillation time timer for oscillation and the PLL clocks.

The timebase timer is also used for the oscillation stabilization interval of the oscillation clock and the PLL clock. The timebase timer counts up the oscillation stabilization interval since it has the value "0000000000000000_B" (counter cleared) and until it detects the oscillation stabilization interval. When the timebase timer returns from timebase timer mode to PLL clock mode, the oscillation stabilization interval indicates only the portion from the middle of counting because the timebase timer counter has not been cleared.

Table 9.5-1 Timebase Timer Counter Clearing and Oscillation Stabilization Wait Intervals

Operation	Counter clear	TBOF clear	Oscillation Stabilization Wait Interval
TBTC: Writing of 0 to TBR	O	O	-
Power-on reset	O	O	Oscillation clock oscillation stabilization wait Interval
Watchdog reset	O	O	
Releasing of stop mode	O	O	Oscillation clock oscillation stabilization wait Interval (at return to main clock mode)
Transition from oscillation clock mode to PLL clock mode (MCS = 1 to 0)	O	O	PLL clock oscillation stabilization wait Interval
Releasing of timebase timer mode	X	X	PLL clock oscillation stabilization wait Interval (at return to PLL clock mode)
Releasing of sleep mode	X	X	-

O: Available
X: Not available

■ Clock Supply Function

The timebase timer supplies clocks to the watchdog timer. Clearing of the timebase counter affects operation of the watchdog timer. For more information, see Section 9.6 "Usage Notes on the Timebase Timer ".

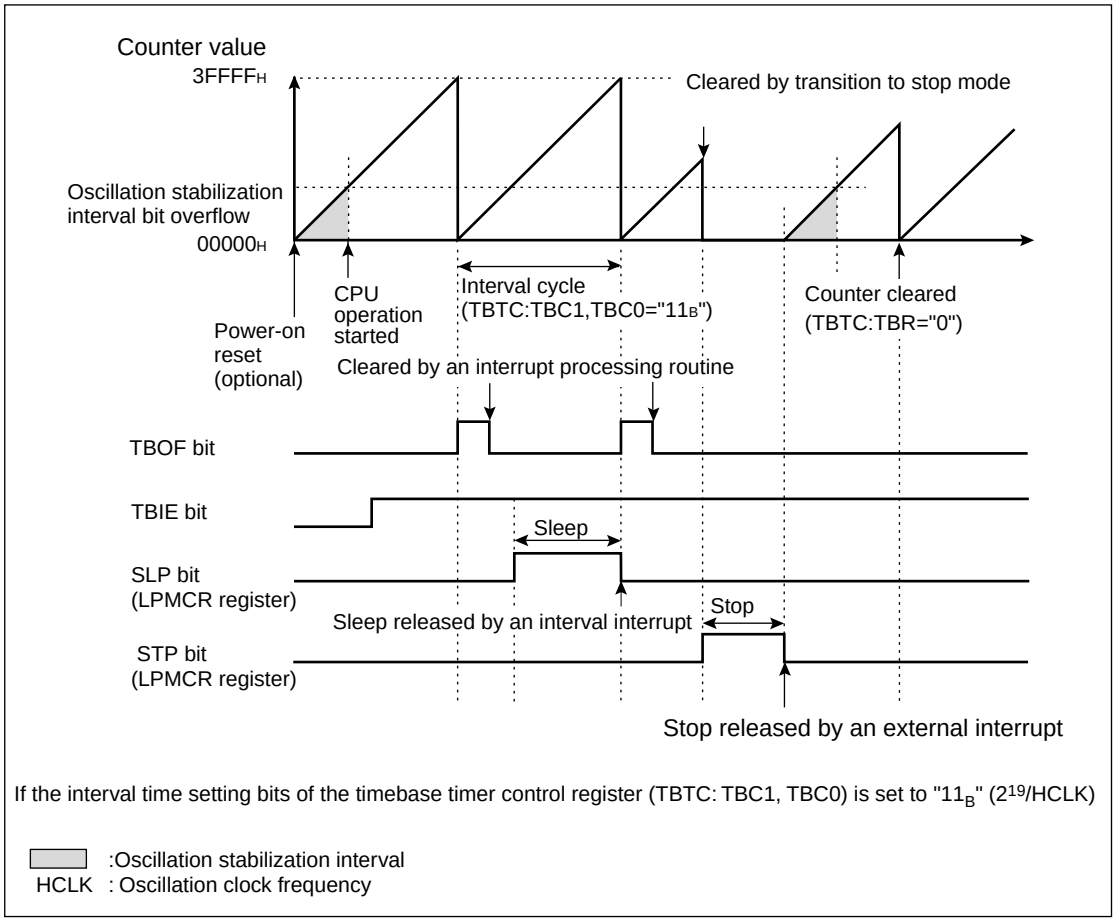
■ Timebase Timer Operation Statuses

Figure 9.5-2 "Timebase Timer Operations" shows the following operation statuses.

- When a power-on reset occurs
- When the CPU enters sleep mode while the interval timer function is operating
- When the CPU enters stop mode
- When the timebase timer counter clear request is issued

When the CPU enters stop mode, the timebase timer counter is cleared and the timebase timer counter stops. To release stop mode, use the timebase timer counter to count the oscillation stabilization interval.

Figure 9.5-2 Timebase Timer Operations



If the interval time setting bits of the timebase timer control register (TBTC: TBC1, TBC0) is set to "11_B" (2^{19}/HCLK)

:Oscillation stabilization interval
HCLK : Oscillation clock frequency

9.6 Usage Notes on the Timebase Timer

This section provides notes on how clearing of an interrupt request or clearing of the timebase timer counter affects the functions.

■ Timebase Timer Usage Notes

❍ Clearing interrupt requests

Clear the interrupt request flag bit (TBOF) of the timebase timer control register (TBTC) to "0" while the interrupt request enable bit (TBIE) or the interrupt level mask register (ILM) of the processor status (PS) is set to disabled.

❍ Functions affected by clearing of the timebase timer counter

- Interval timer function (interval interrupt)
- When the watchdog timer is being used

❍ Use of the timebase timer as the oscillation settling time timer

In stop mode in which the operating clock stops, the timebase timer counter is cleared and stopped. When the timebase timer counter is cleared, the clock supplied from it starts to be supplied again from the initial state. As a result, the H level may be shortened or the L level may be prolonged by half a cycle at the maximum. Although the clock for the watchdog timer also starts to be supplied again from the initial state, the watchdog timer operates in normal cycles because the watchdog timer counter is cleared at the same time.

❍ Notes on peripheral functions to which clocks are supplied from the timebase timer

At power-on or in stop mode, the source oscillation is stopped. Thus, the timebase timer counter places the oscillation stabilization interval for the operating clock using the clock supplied from the oscillator. Depending on the oscillator type, an appropriate oscillation stabilization interval must be specified.

For more information, see Section 4.5 "Oscillation Stabilization Wait Interval".

9.7 Sample Program for the Timebase Timer Program

This section contains a sample program for the timebase timer.

■ Sample Program for the Timebase Timer

❍ Processing

An interval interrupt of 2^{12}/HCLK (HCLK: oscillation clock) is repeatedly generated. The interval becomes approx. 1.0 ms (during 4 MHz operation).

❍ Coding example

```
ICR12 EQU  0000BCH          ;Timebase timer interrupt control
                             register
TBTC  EQU  0000A9H          ;Timebase timer control register
TBOF  EQU  TBTC:3           ;Interrupt request flag bit
;-----Main program-----------------------------------------------
CODECSEG
START:
  ;      :                  ;Assumes that stack pointer (SP) has
                             already been initialized.
      AND  CCR,#0BFH        ;Disables interrupts.
      MOV  I:ICR12,#00H     ;Interrupt level 0 (highest)
      MOV  I:TBTC,#10010000B;Fixes upper 3 bits.
                            ;Enables interrupts and clears
                             TBOF to "0".
                            ;Clears counter.
                            ;Selects interval 212/HCLK
      MOV  ILM,#07H         ;Sets PS ILM to level 7.
      OR   CCR,#40H         ;Enables interrupts.
LOOP: MOV  A,#00H           ;Endless loop
      MOV  A,#01H
      BRA  LOOP
;-----Interrupt program------------------------------------------
WARI:
      CLRB I:TBOF           ;Clears interrupt request flag to "0"
  ;    :
  ;   User handling
  ;    :
      RETI                  ;Returns from interrupt.
CODE  ENDS
;-------Vector setting-------------------------------------------
VECT  CSEG ABS=0FFH
      ORG  0FF6CH           ;Sets vector for interrupt #36 (24H).
      DSL  WARI
      ORG  0FFDCH           ;Sets reset vector.
      DSL  START
      DB   00H              ;Sets single-chip mode.
VECT  ENDS
      END  START
```

CHAPTER 10 WATCHDOG TIMER

This chapter describes the functions and operation of the watchdog timer.

10.1 Overview of the Watchdog Timer

The watchdog timer is a 2-bit counter that uses the output of the timebase timer counter as the count clock. After the watchdog timer is activated, the CPU is reset within a specified interval unless the watchdog timer is cleared.

■ Watchdog Timer Function

The watchdog timer is provided to handle program runaways. The watchdog timer, once activated, must continue to be cleared within every specified interval. If the program results in an endless loop and the watchdog timer is not cleared within the minimum time shown in Table 10.1-1 "Intervals for the Watchdog Timer" a watchdog reset is issued to the CPU, sending it into the reset status. Specify the watchdog timer interval in the interval setting bits (WT1, WT0) of the watchdog timer control register (WDTC).

Table 10.1-1 Intervals for the Watchdog Timer

WT1	WT0	Interval		
		Minimum (*1)	Maximum (*1)	Oscillation clock cycle count
0	0	Approx. 3.58 ms	Approx. 4.61 ms	2^{14}±2^{11} cycle
0	1	Approx. 14.33 ms	Approx. 18.3 ms	2^{16}±2^{13} cycle
1	0	Approx. 57.23 ms	Approx. 73.73 ms	2^{18}±2^{15} cycle
1	1	Approx. 458.75 ms	Approx. 589.82 ms	2^{21}±2^{18} cycle

*1 Value during operation of the 4 MHz oscillation clock

For more information on a watchdog timer interval, see Section 10.4 "Operation of the Watchdog Timer".

Note:

The watchdog counter consists of a 2-bit counter that uses the carry signals of the timebase timer as count clocks. Therefore, if the timebase timer is cleared, the watchdog reset generation time may become longer than the time set.

Reference:

The watchdog timer, after being activated, can be stopped using a power-on reset or a reset from the watchdog timer. The watchdog timer can be cleared with an external reset, an internal reset, writing to the watchdog control bit (WTE) of the watchdog timer control register (WDTC), or transition to sleep or stop mode. However, the watchdog function is enabled and not stopped.

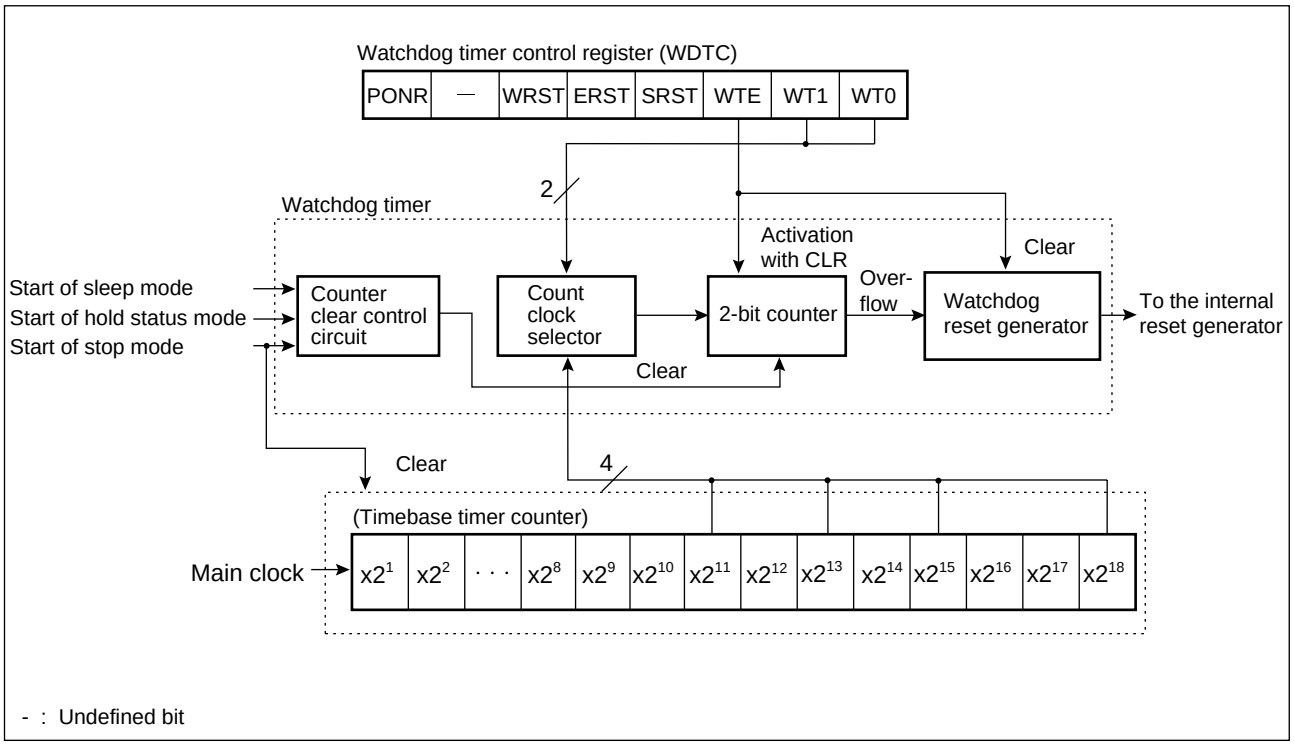
10.2 Configuration of the Watchdog Timer

The watchdog timer consists of the following five blocks:

- **Count clock selector**
- **Watchdog timer (2-bit counter)**
- **Watchdog reset generator**
- **Counter clear control circuit**
- **Watchdog timer control register (WDTC)**

■ Block Diagram of the Watchdog Timer

Figure 10.2-1 Block Diagram of the Watchdog Timer



❍ Count clock selector

Used to select one of the four timebase timer output clocks as the count clock of the watchdog timer. Setting the count clock of the watchdog timer determines a watchdog reset interval.

❍ Watchdog counter (2-bit counter)

The watchdog timer is a 2-bit timer that counts the clock specified by the count clock selector.

❍ Watchdog reset generator

Used to generate the reset signal by an overflow of the watchdog counter.

- **Counter clear circuit**

  Used to clear the watchdog counter and to control the operation or stopping of the counter.

- **Watchdog timer control register (WDTC)**

  Used to set the interval, activate and clear the watchdog timer, and hold the reset generation cause.

10.3 Watchdog Timer Control Register (WDTC)

This section describes the watchdog timer control register (WDTC).

■ Watchdog Timer Control Register (WDTC)

Figure 10.3-1 Watchdog Timer Control Register (WDTC)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000A8H	PONR	—	WRST	ERST	SRST	WTE	WT1	WT0	1XXXX111B
	R	-	R	R	R	W	W	W	

WT1	WT0	Interval selection bit (for 4 MHz HCLK)		
		Interval		Oscillation clock cycle count
		Minimum	Maximum	
0	0	Approx. 3.58ms	Approx. 4.61ms	$2^{14} \pm 2^{11}$ cycle
0	1	Approx. 14.33ms	Approx. 18.3ms	$2^{16} \pm 2^{13}$ cycle
1	0	Approx. 57.23ms	Approx. 73.73ms	$2^{18} \pm 2^{15}$ cycle
1	1	Approx. 458.75ms	Approx. 589.82ms	$2^{21} \pm 2^{18}$ cycle

WTE	Watchdog control bit
0	- Activation of the watchdog timer (At first write after reset) - Clearing of the watchdog timer (At second or subsequent write after reset)
1	No operation

Reset cause bit				Reset cause
PONR	WRST	ERST	SRST	
1	X	X	X	Power-on
*	1	*	*	Watchdog timer
*	*	1	*	External reset ($\overline{\text{RST}}$ pin input)
*	*	*	1	Software reset (LPMCR: RST)

R : Read only
W : Write only
X : Undefined
- : Undefined bit
* : Retains the previous status.
□ : Initial value
HCLK : Oscillation clock frequency

Table 10.3-1 Function Description of Each Bit of the Watchdog Timer Control Register (WDTC)

Bit name		Function
bit7 bit5 bit4 bit3	PONR, WRST, ERST, SRST: Reset cause bits	• Read-only bits for indicating the reset cause. If more than one reset cause occurs, the bit for each reset cause occurring is set to 1. • These bits are all cleared to 0 after the watchdog timer control register (WDTC) is read. • A power-on reset sets the reset cause flag bit PONR to "1" and sets the reset cause flag bits WRST, ERST, and SRST to an undefined value. • If the PONR bit is set to "1", the contents of the WRST, ERST, and SRST bits should be ignored.
bit6	-: Undefined bit	• The value of this bit is not defined if it is read. • The set value does not affect the operaton.
bit2	WTE: Watchdog timer control bit	• When "0" is written to this bit, the watchdog timer is activated. However, this occurs when the register is accessed for the first time after a power-on reset or a reset from the watchdog timer. • When "0" is written to this bit after the watchdog timer is activated, the watchdog timer is cleared. • Writing 1 does not affect operation.
bit1 bit0	WT1, WT0: Interval selection bit	• Used to select the watchdog timer interval. • Data in these bits is valid when the watchdog timer is activated. Data can be written to this bit and the watchdog control bit (WTE) at the same time. Data written to these bits after the watchdog timer is activated is invalid. • These bits are write-only.

10.4 Operation of the Watchdog Timer

The watchdog timer generates a watchdog reset by an overflow of the watchdog counter.

■ Watchdog Timer Operation

Operation of the watchdog timer requires the setting in Figure 10.4-1 "Setting of the Watchdog Timer".

Figure 10.4-1 Setting of the Watchdog Timer

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0000A8H	PONR	—	WRST	ERST	SRST	WTE	WT1	WT0
						0	◎	◎

◎ : Used
0 : Set 0.

❍ Activating the watchdog timer

- To activate the watchdog timer, set the watchdog control bit (WTE) of the watchdog timer control register (WDTC) to "0" for the first time after a reset from the watchdog timer is generated after a power-on reset. The interval can be set in the interval setting bits (WT1, WT0) of the watchdog timer control register (WDTC).
- The watchdog timer, after being activated, can be stopped using a power-on reset or a reset from the watchdog timer. The watchdog timer cannot be stopped by an external reset, a software reset, writing to the watchdog control bit (WTE) of the watchdog timer control register (WDTC), or transition to sleep or timebase timer mode.

❍ Clearing the watchdog timer

- To clear the watchdog timer, set the watchdog control bit (WTE) of the watchdog timer control register (WDTC) to "0".
- The watchdog timer can be cleared also by input of an external reset or an internal reset or transition to sleep mode.
- Transition to timebase timer mode clears and stops the watchdog timer.

❍ Intervals for the watchdog timer

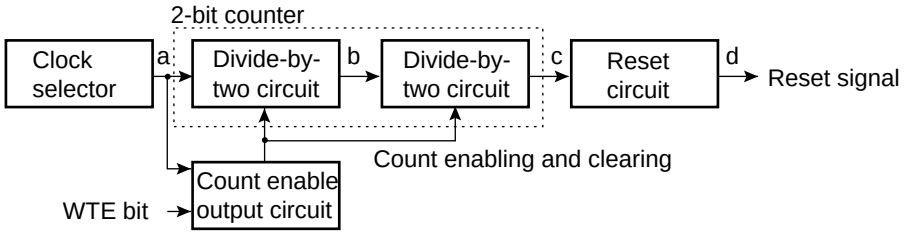
Figure 10.4-2 "Clear Timing and Watchdog Timer Intervals" shows the relationship between the clear timing of the watchdog timer and intervals.

❍ Checking a reset cause

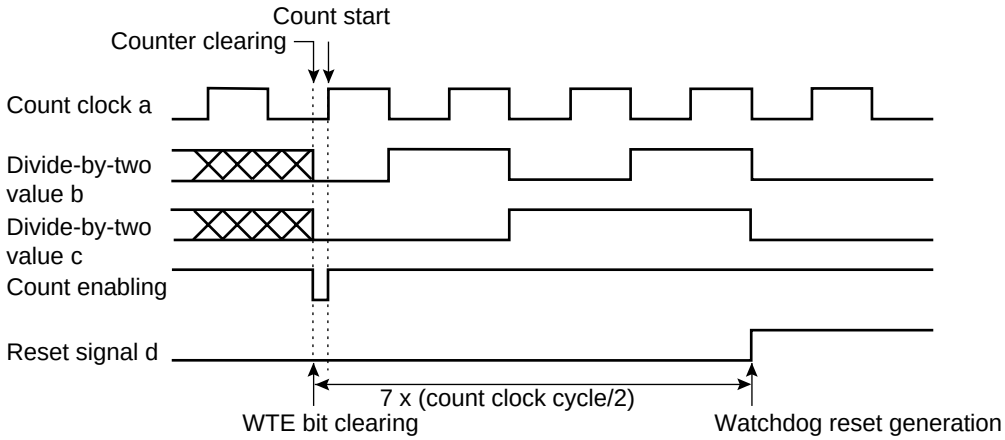
A reset cause can be determined by checking the PONR, WRST, ERST, and SRST bits of the watchdog timer control register (WDTC) after a reset.

Figure 10.4-2 Clear Timing and Watchdog Timer Intervals

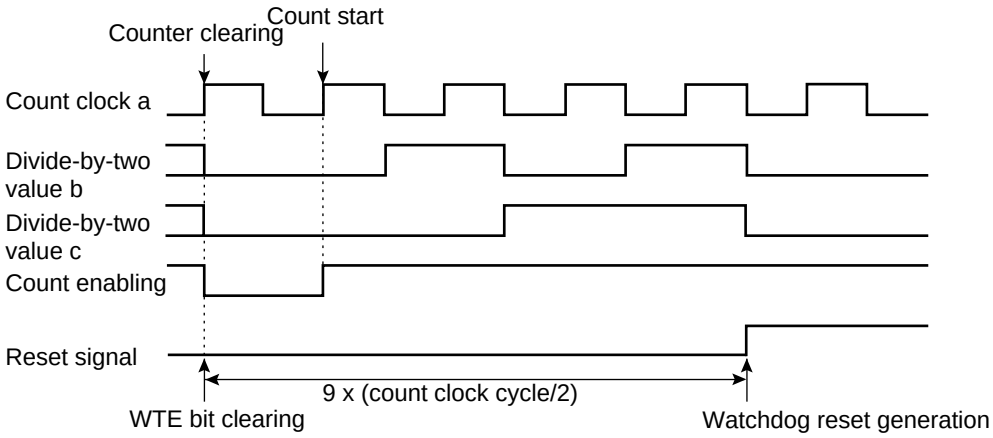
[WDG timer block diagram]



[Minimum interval] When the WTE bit is cleared immediately before the count clock rises:



[Maximum interval] When the WTE bit is cleared immediately after the count clock rises:



10.5 Usage Notes on the Watchdog Timer

Notes on using the watchdog timer are given below.

■ Usage Notes on the Watchdog Timer

❍ **Stopping the watchdog timer**

Once the watchdog timer is activated, it cannot stop until a power-on or watchdog reset occurs.

❍ **Interval setting**

The interval setting becomes valid when the watchdog timer is activated. The interval setting is ignored unless the watchdog timer is activated.

❍ **Notes on program creation**

If the watchdog timer is repetitiously cleared in a program, the processing time of the program including the interrupt processing must be equal to or less than the minimum interval.

❍ **Watchdog timer operation in timebase timer mode**

While the CPU is stopped, the watchdog timer is cleared and stopped. The watchdog timer is restarted when the CPU is switched from timebase timer mode to main clock mode or PLL clock mode.

10.6 Sample Program for the Watchdog Timer

This section contains a sample program for the watchdog timer.

■ Sample Program for the Watchdog Timer

❍ Processing

- The watchdog timer is cleared every time in the main program loop.
- The main loop must make one iteration within the minimum watchdog timer interval.

❍ Coding example

```
WDTC      EQU   000A8H          ;Watchdog timer control register
WTE       EQU   WDTC:2          ;Watchdog control bit
;-------Main program-----------------------------------------------
CODE      CSEG
START:
;          :                    ;Assumes that stack pointer (SP) has
                                 already been

WDG_START:
          MOV   WDTC,#00000011B;Activates watchdog timer.
                                ;Selects the interval 221 ±218 cycle.
;--------Main loop-------------------------------------------------
MAIN:     CLRB  I:WTE           ;Clears watchdog timer.
;          :                    Clears two bits regularly.
;         User processing
;          :
          JMP   MAIN            ;Loops in less time than the
                                 watchdog timer interval.
CODE      ENDS
;--------Vector setting--------------------------------------------
VECT      CSEG  ABS=0FFH
          ORG   0FFDCH          ;Sets reset vector.
          DSL   START
          DB    00H             ;Sets single-chip mode.
VECT      ENDS
          END   START
```

CHAPTER 11 16-BIT RELOAD TIMER

The chapter describes the functions and operation of the 16-bit reload timer.

11.1 Overview of the 16-Bit Reload Timer

The MB90560 and 565 series have two channels of 16-bit reload timer. The following two clock modes and the following two counter operation modes can be selected.

Clock modes

- **Internal clock mode: The timer counts down in synchronization with the internal clock.**
- **Event count mode: The timer counts down in synchronization with the external input pulse.**

Counter operation modes

- **Reload mode: The timer repeats counting by reloading the count setting value.**
- **One-shot mode: The timer stops counting when an underflow occurs.**

■ 16-bit reload timer operating modes

Table 11.1-1 16-Bit Reload Timer Operating Modes

Clock mode	Counter operation	Operation of 16-bit reload timer
Internal clock mode	Reload mode	Software trigger operation External trigger operation External gate input operation
	One-shot mode	
Event count mode (external clock mode)	Reload mode	Software trigger operation
	One-shot mode	

■ Internal Clock Mode

The 16-bit reload timer is in internal clock mode if the count clock setting bits (CSL1, CSL0) of the timer control status register (TMCSR) are set to "00_B", "01_B", or "10_B".

For internal clock mode, select one of the following three operations:

❍ Software trigger operation

Counting starts when the software trigger bit (TRG) is set to "1" while the count enable bit (CNTE) of the TMCSR register is set to "1".

❍ External trigger operation

Counting starts when a valid edge (rising, falling, or both edges) of trigger input specified in the operation mode setting bits (MOD2, MOD1, and MOD0) is input to the TIN pins while the CNTE bit of the TMCSR register is set to "1".

❍ External gate input operation

Counting continues while a valid level of gate input ("L" or "H") specified in the operation mode setting bits (MOD2, MOD1, and MOD0) is input to the TIN pins while the CNTE bit of the TMCSR register is set to "1".

■ Event Count Mode (External Clock Mode)

The 16-bit reload timer is in event count mode (external clock) if the count clock setting bits (CSL1, SCL0) of the timer control status register (TMCSR) are set to "11_B". Counting starts when a valid edge (rising, falling, or both edges) of trigger input specified in the operation mode setting bits (MOD2, MOD1, and MOD0) is input to the TIN pins while the CNTE bit is set to "1". The 16-bit reload timer can also be used as an interval timer if external clock is input periodically.

■ Counter Operation

❍ Reload mode

Counting starts when an underflow of the 16-bit down counter (change from "0000_H" to "$FFFF_H$") loads the values of the 16-bit reload register (TMRLR0/TMRHR0, TMRLR1/TMRHR1) into the 16-bit down counter. Since the 16-bit reload timer causes an interrupt request to occur for an underflow condition, it can be used as an interval timer. Every time an underflow occurs, a reversed toggle waveform can be output from the T0 pin.

Table 11.1-2 Intervals for the 16-Bit Reload Timer

Count clock	Count clock period	Interval
Internal clock	$2^1/\phi$ (0.125 µs)	0.125 µs to 8.192 ms
	$2^3/\phi$ (0.5 µs)	0.5 µs to 32.768 ms
	$2^5/\phi$ (2.0 µs)	2.0 µs to 131.1 ms
External clock	$2^3/\phi$ or more (0.5 µs)	0.5 µs or more

ϕ : Mchine clock
Values in parentheses are for a 16 MHz machine clock.

❍ Single-shot mode

An underflow of the 16-bit down counter (change from "0000_H" to "$FFFF_H$") stops counting.

Reference:

- 16-bit reload timer 0 can be used to create the baud rate of UART0.
- 16-bit reload timer 1 can be used to create the baud rate of UART1 and provide a start trigger of the A/D converter.

■ 16-Bit Reload Timer Interrupts and EI^{2}OS

An underflow of the 16-bit down counter (change from "0000_H" to "$FFFF_H$") outputs an interrupt request.

Table 11.1-3 16-Bit Reload Timer Interrupts and EI^{2}OS

Channel	Interrupt number	Interrupt control register		Vector table address			EI^{2}OS
		Register name	Address	Lower	Upper	Bank	
16-bit reload timer 0 (*1)	#30 ($1E_H$)	ICR09	$0000B9_H$	$FFFF84_H$	$FFFF85_H$	$FFFF86_H$	O
16-bit reload timer 1 (*2)	#32 (20_H)	ICR10	$0000BA_H$	$FFFF7C_H$	$FFFF7D_H$	$FFFF7E_H$	

O: Enabled
*1 16-bit reload timer 0 is assigned to the same interrupt control register (ICR09) as 8-bit timer 0, 1, and 2 counter borrow.
*2 16-bit reload timer 1 is assigned to the same interrupt control register (ICR10) as 16-bit free-running timer overflow.

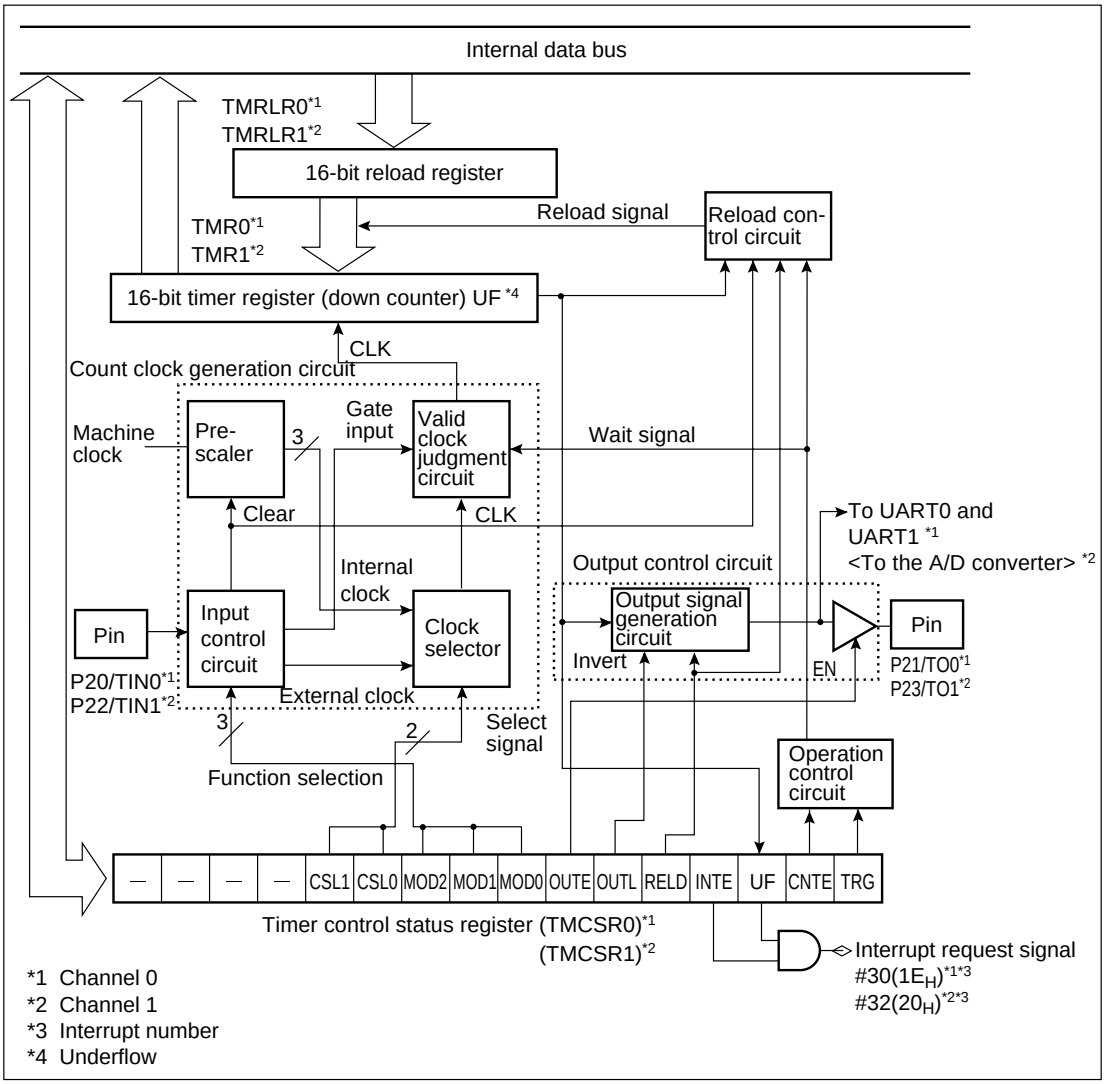
11.2 Configuration of the 16-Bit Reload Timer

16-bit reload timers 0 and 1 each consist of the following seven blocks:

- **Count clock generation circuit**
- **Reload control circuit**
- **Output control circuit**
- **Operation control circuit**
- **16-bit timer register (TMR0/TMR1)**
- **16-bit reload register (TMRLR0/TMRLR1, TMRHR0/TMRHR1)**
- **Timer control status register (TMCSR0/TMCSR1: H, TMCSR0/TMCSR1: L)**

■ Block Diagram of the 16-Bit Reload Timer

Figure 11.2-1 Block Diagram of the 16-Bit Reload Timer



- **Count clock generation circuit**

  This circuit generates the count clock for the 16-bit reload timer from the machine clock or external input clock.

- **Reload control circuit**

  This circuit controls the reload operation through activation of the 16-bit down counter and an underflow (change from "0000_H" to "$FFFF_H$").

- **Output control circuit**

  This circuit controls the inversion of the TO pin output through an underflow of the 16-bit down counter (change from "0000_H" to "$FFFF_H$") and enabling and disabling of TO pin output.

- **Operation control circuit**

  This circuit controls activation and stop of the 16-bit down counter.

- **16-bit timer register (TMR0/TMR1)**

  This register is a 16-bit down counter. The current counter value is read from this register during a read operation.

- **16-bit reload register (TMRLR0/TMRHR0, TMRLR1/TMRHR1)**

  This register stores a value to be loaded to the 16-bit down counter. The setting value of this register is loaded to the 16-bit down counter, which then starts counting down.

- **Timer control status register (TMCSR0/TMCSR1: H,TMCSR0/TMCSR1: L)**

  This register selects the operation mode of the 16-bit reload timer, the count clock, and the operating conditions, enables and disables counting, controls interrupts, and checks the statuses of interrupt requests.

11.3 16-Bit Reload Timer Pins

This section describes the pins of the 16-bit reload timer and provides a pin block diagram.

■ 16-Bit Reload Timer Pins

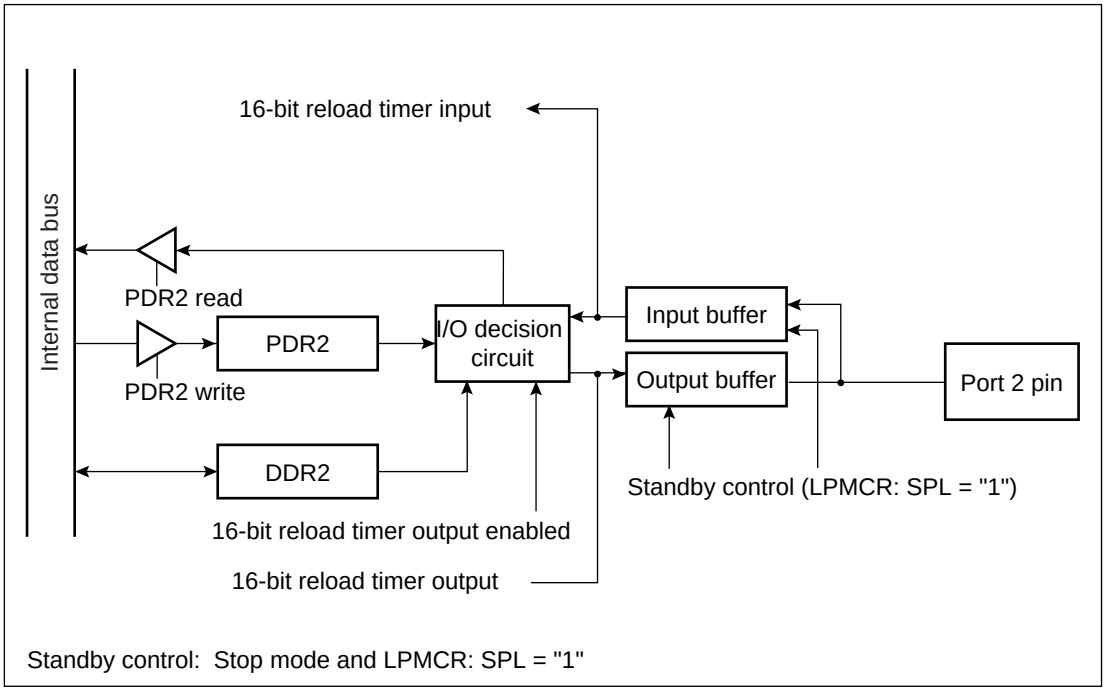
The pins of the 16-bit reload timer are shared with the general-purpose ports.

Table 11.3-1 16-Bit Reload Timer Pins

Pin name	Pin function	I/O format	Pull-up option	Standby control	Settings required for pins
P20/TIN0	I/O port: timer 0 input	CMOS output/ CMOS hysteresis input	Selection allowed	Available	Setting for the input port (DDR2: bit 0 = 0)
P21/TO0	I/O port: timer 0 output				Setting for timer 0 output enable (TMCSR0: OUTE = 1)
P22/TIN1	I/O port: timer 1 input				Setting for the input port (DDR2: bit 2 = 0)
P23/TO1	I/O port: timer 1 output				Setting for timer 1 output enable (TMCSR1: OUTE = 1)

■ Block Diagram of the 16-Bit Reload Timer Pins

Figure 11.3-1 Block Diagram of the 16-Bit Reload Timer Pins



11.4 16-Bit Reload Timer Registers

The 16-bit reload timer registers are as follows.

■ 16-Bit Reload Timer Registers

Figure 11.4-1 16-Bit Reload Timer Registers

Address	bit15 ····· bit8 bit7 ····· bit0
ch0:000083H, 000082H	Timer control status register channel 0 (TMCSR0)
ch0:000085H, 000084H	16-bit timer register channel 0/16-bit reload register channel 0 (*1) (TMR0, TMRHR0/TMRLR0)
ch1:000087H, 000086H	Timer control status register channel 1 (TMCSR1)
ch1:000089H, 000088H	16-bit timer register channel 1/16-bit reload register channel 1 (*1) (TMR1, TMRHR1/TMRLR1)

*1 This register functions as a 16-bit timer register (TMR) during reading, and functions as a 16-bit reload register (TMRHR/TMRLR) during writing.

11.4.1 Timer Control Status Register, High Part (TMCSR0/ TMCSR1: H)

Bits 11 to 7 of the timer control status registers (TMCSR0 and TMCSR1) are used to select the operating mode and the count clock of the 16-bit reload timer.

■ Timer Control Status Register, High Part (TMCSR0/TMCSR1: H)

Figure 11.4-2 Timer Control Status Register, High Part (TMCSR0/TMCSR1: H)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7	Initial value
ch0:000083H ch1:000087H	—	—	—	—	CSL1	CSL0	MOD2	MOD1	MOD0	XXXX00000B
	-	-	-	-	R/W	R/W	R/W	R/W	R/W	

MOD2	MOD1	MOD0	Operating mode selection bit (in the internal clock mode)	
			Input terminal function	Valid edge, level
0	0	0	Trigger inhibit	-
0	0	1	Trigger input	Rising edge
0	1	0		Falling edge
0	1	1		Both edges
1	X	0	Gate input	"L" level
1	X	1		"H" level

MOD2	MOD1	MOD0	Operating mode selection bit (in the event count mode)	
			Input terminal function	Valid edge
X	0	0	-	-
X	0	1	Trigger input	Rising edge
X	1	0		Falling edge
X	1	1		Both edges

CSL1	CSL0	Count clock selection bit	
		Function	Count clock
0	0	Internal clock mode	$2^1/\phi$ (0.125 μs)
0	1		$2^3/\phi$ (0.5 μs)
1	0		$2^5/\phi$ (2.0 μs)
1	1	Event count mode	External event input

R/W : Readable and writable
X : Indefinite
- : Undefined bit
□ : Initial value
ϕ : Machine clock, the value in the parentheses ( ) indicates the value when the machine clock is operated at 16MHz.

Table 11.4-1 Function Description of Each Bit of the High Part of the Timer Control Status Register (TMCSR0/TMCSR1: H)

Bit name		Function
bit15 bit14 bit13 bit12	Not used Undefined bit	• When these bits are read, their values are undefined. • Writing to these bits has no effect on operation.
bit11 bit10	CSL1, CSL0: Count clock selection bits	• These bits select the count clock of the 16-bit reload timer. • When these bits are set to "00_B", "01_B", and "10_B", internal clock mode is selected. • When these bits are set to 11_B, event count mode is set.
bit9 bit8 bit7	MOD2, MOD1, MOD0: Operating mode selection bits	• These bits select operation mode. **In internal clock mode:** • The MOD2 bit is used to select input pin functions. • When the MOD2 bit is set to "0", the input pin is used as a trigger input pin. Whenever a valid edge is input, the value in the 16-bit reload register (TMRLR0/TMRHR0, TMRLR1/TMRHR1) is loaded into the counter and counting starts. The MOD1 and MOD0 bits select the type of valid edge. • When the MOD2 bit is set to "1", the input pin becomes a gate. Counting continues as long as a valid level specified in the MOD0 bit is input. • The value specified in the MOD1 bit has no effect on operation. **In event count mode:** • The MOD2 bit is not affected. • In event count mode, the input pin becomes a trigger input pin. Counting starts when a valid edge specified in the MOD1 and MOD0 bits is input.

11.4.2 Timer Control Ctatus Register, Low Part (TMCSR0/ TMCSR1: L)

The lower seven bits of the timer control status registers (TMCSR0 and TMCSR1) set the operating conditions for the 16-bit reload timer, enable and disable counting, control interrupts, and check the statuses of interrupt requests.

■ Timer Control Status Register, Low Part (TMCSR0/TMCSR1: L)

Figure 11.4-3 Timer Control Status Register, Low Part (TMCSR0/TMCSR1: L)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ch0:000082H ch1:000086H	* MOD0	OUTE	OUTL	RELD	INTE	UF	CNTE	TRG	00000000B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

TRG	Software trigger bit
0	No change, no effect on other bits
1	Counting starts after data loading

CNTE	Count enable bit
0	Counting stopped
1	Counting enabled (wait for the start trigger))

UF	Underflow interrupt request flag bit	
	During reading	During writing
0	No interrupt request issued	Clearing the interrupt request
1	Interrupt request issued	No effect on operation

INTE	Underflow interrupt request enable bit
0	Interrupt request output disabled
1	Interrupt request output enabled

RELD	Reload selection bit
0	One-shot mode (reload disabled)
1	Reload mode (reload enabled)

OUTL	Pin output level selection bit	
	In single-shot mode (RELD="0")	In reload mode (RELD="1")
0	Square wave of H during counting	Toggle output of L when counting is started.
1	Square wave of L during counting	Toggle output of H when counting is started.

OUTE	Timer output enable bit		
	Pin functions	Registers and pins corresponding to each channel	
		TMCSR0	TMCSR1
0	I/O port	P21	P23
1	Timer output	TO0	TO1

R/W : Read/write
: Initial value
* : See Section 11.4.1, "Timer control status register, high part (TMCSR0, TMCSR1: H)," for MOD0 (bit 7)

Table 11.4-2 Function Description of Each Bit of the Low Part of the Timer Control StatusRegister (TMCSR0/TMCSR1: L)

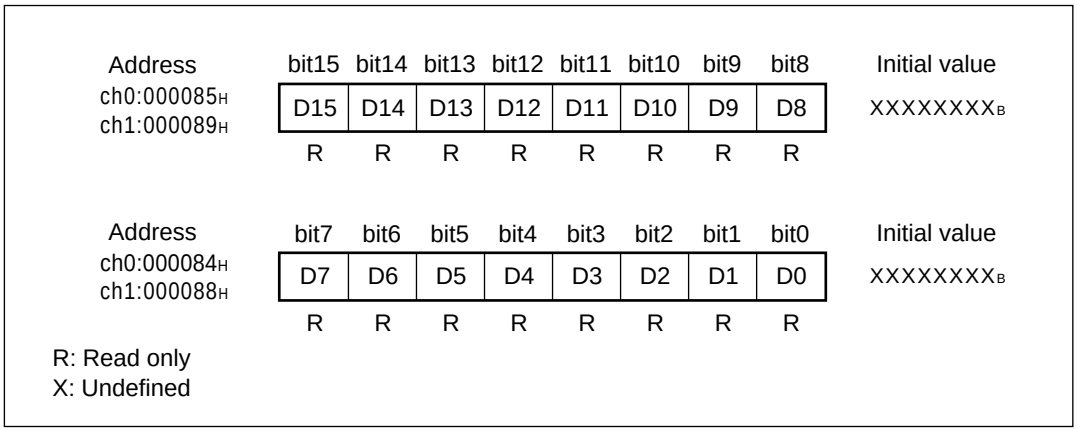
Bit name		Function
bit6	OUTE: Timer output enable bit	• This bit enables or disables output to the timer output pin. • When this bit is set to "0", the pin serves as an I/O port. When this bit is set to "1", the pin serves as a timer output pin.
bit5	OUTL: Pin output level setting bit	• This bit selects the output level of the timer output pin. • The timer output pin outputs a toggle waveform in reload mode or a square wave indicating that counting is in progress in one-shot mode. • Opposite output levels are output from the pin depending on whether this bit is set to "0" or "1".
bit4	RELD: Reload selection bit	• This bit enables reloading. • When this bit is set to "1", the timer is in reload mode. When an underflow of the 16-bit down counter occurs, the value stored in the 16-bit reload register is loaded into the 16-bit down counter and counting continues. • When this bit is set to "0", the timer is in one-shot mode. When an underflow of the 16-bit down counter occurs, counting stops.
bit3	INTE: Underflow interrupt request enable bit	• This bit enables interrupt requests. • When this bit and the interrupt request flag (UF) bit are 1, the timer outputs an interrupt request.
bit2	UF: Underflow interrupt request flag bit	• This is a flag bit for an interrupt request. • This bit is set to "1" when an underflow of the 16-bit down counter occurs. • An interrupt request is output when this bit is set to "1" while the underflow interrupt request enable bit (INTE) is set to "1". • Setting this bit to "0" clears an interrupt request. • Setting this bit to "1" has no effect on operation. • This bit is also cleared when EI^2OS is activated.
bit1	CNTE: Count enable bit	• This bit enables or disables counting. • When this bit is set to "1", the counter is placed in trigger standby mode. Counting starts when the software trigger bit (TRG) is set to "1" or a valid edge (rising, falling, or both edges) of trigger input specified in the operation mode setting bits (MOD2, MOD1, and MOD0) is input to the TIN pins. • When this bit is set to "0", counting stops.
bit0	TRG: Software trigger bit	• This bit starts the interval timer function or counter function with software. • When this bit is set to "1" while the count enable bit (CNTE) is set to "1", the value stored in the 16-bit reload register is loaded into the 16-bit down counter and counting starts. • When this bit is set to "0", there is no effect on operation. • The read value is "0".

11.4.3 16-bit Timer Register (TMR0/TMR1)

The 16-bit timer register (TMR0/TMR1) is always able to read the count value from the 16-bit down counter.

■ 16-bi t Timer Register (TMR0/TMR1)

Figure 11.4-4 16-Bit Timer Register (TMR0/TMR1)



The 16-bit timer register is a 16-bit down counter.

When the software trigger bit (TRG) is set to "1" while the count enable bit (CNTE) of the timer control status register (TMCSR) is set to "1" or a valid edge (rising, falling, or both edges) of trigger input specified in the operation mode setting bits (MOD2, MOD1, and MOD0) is input to the TIN pins, the values stored in the 16-bit reload register (TMRLR0/TMRHR0, TMRLR1/TMRHR1) are loaded into the 16-bit down counter and counting starts. This register holds the value of the 16-bit timer register (TMR0/TMR1) while counting is stopped (TMCSR: CNTE = "0").

Note:

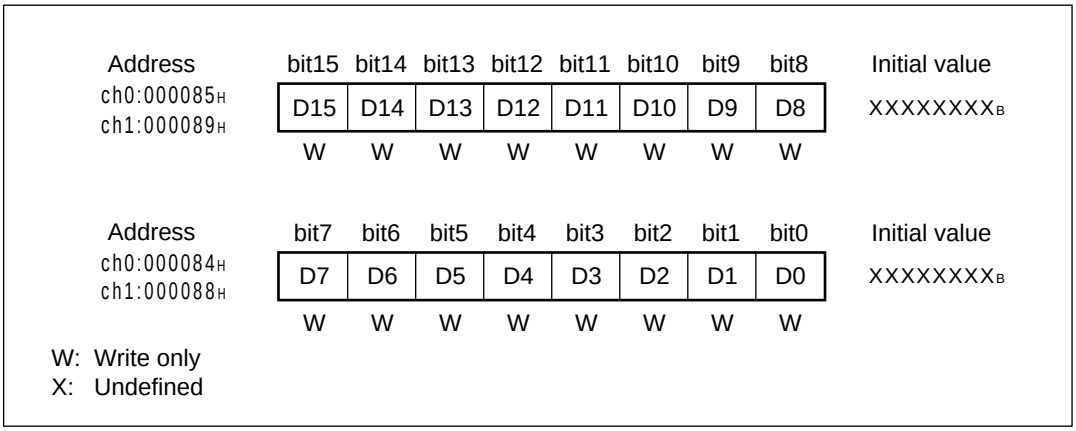
- Be sure to use a word transfer instruction (MOVW A, $003A_H$) to read data from the 16-bit timer register (TMR0/TMR1).
- Although 16-bit timer register (TMR0/TMR1) is read-only and the 16-bit reload register (TMRLR0/TMRHR0, TMRLR1/TMRHR1) is write-only, they are placed at the same address. Thus, writing a value to the 16-bit timer register has no effect on this register because the value is written to the 16-bit reload register.

11.4.4 16-bit Reload Register (TMRLR0/TMRHR0, TMRLR1/TMRHR1)

The 16-bit reload register (TMRLR0/TMRHR0, TMRLR1/TMRHR1) sets a reload value in the 16-bit down counter. The value written to this register is loaded into the down counter, and the value is counted down.

■ 16-Bit Reload Register (TMRLR0/TMRHR0, TMRLR1/TMRHR1)

Figure 11.4-5 16-Bit Reload Register (TMRLR0/TMRHR0, TMRLR1/TMRHR1)



Counting must be stopped (TMCSR: CNTE = "0") regardless of the operating mode of the 16-bit reload register when a value is written to the 16-bit reload register (TMRLR0/TMRHR0, TMRLR1/TMRHR1). When the software trigger bit (TRG) is set to "1" or a valid edge (rising, falling, or both edges) of trigger input specified in the operation mode setting bits (MOD2, MOD1, and MOD0) is input to the TIN pins, a value stored in the 16-bit reload register is loaded into the 16-bit down counter and countdown starts.

In reload mode, when an underflow of the 16-bit down counter occurs (change from "0000_H" to "$FFFF_H$"), a value stored in the 16-bit reload register (TMRLR0/TMRHR0, TMRLR1/TMRHR1) is loaded into the 16-bit down counter and countdown continues. In one-shot mode, when an underflow of the 16-bit down counter occurs, the 16-bit down counter stops at the value "$FFFF_H$".

Note:

- Counting must be stopped (TMCSR: CNTE = "0") when a value is written to the 16-bit reload register (TMRLR0/TMRHR0, TMRLR1/TMRHR1).
- Use a word transfer instruction (MOVW $003A_H$, A) to write a value to the 16-bit reload register (TMRLR0/TMRHR0, TMRLR1/TMRHR1).
- Although the 16-bit reload register (TMRLR0/TMRHR0, TMRLR1/TMRHR1) is write-only and the 16-bit timer register (TMR0/TMR1) is read-only, they are placed at the same address. Since different values are written to, and read from these registers, none of the INC/DEC and other instructions that result in read-modify-write (RMW) operations can be used.

11.5 16-Bit Reload Timer Interrupts

The 16-bit reload timer outputs an interrupt request when an underflow of the 16-bit down counter occurs. The 16-bit reload timer also supports the extended intelligent I/O service (EI^{2}OS).

■ 16-Bit Reload Timer Interrupts

Table 11.5-1 Interrupt Control Bits and Interrupt Causes of the 16-Bit Reload Timer

	16-bit reload timer 0	16-bit reload timer 1
Underflow interrupt request flag bit	TMCSR0: UF	TMCSR1: UF
Underflow interrupt request enable bit	TMCSR0: INTE	TMCSR1: INTE
Interrupt cause	Underflow of the 16-bit down counter (TMR0)	Underflow of the 16-bit down counter (TMR1)

In the 16-bit reload timer, the underflow interrupt request flag bit (UF) of the timer control status register (TMCSR) is set to "1" when an underflow of the 16-bit down counter occurs (change from "0000_H" to "$FFFF_H$"). If, at this time, the underflow interrupt request enable bit is set to Enabled (TMSCR: INTE = "1"), an interrupt request is output.

■ EI^{2}OS Function of the 16-Bit Reload Timer

The 16-bit reload timer allows the use of the extended intelligent I/O service (EI^{2}OS) when an underflow of the 16-bit down counter occurs (change from "0000_H" to "$FFFF_H$").

11.6 Operation of the 16-Bit Reload Timer

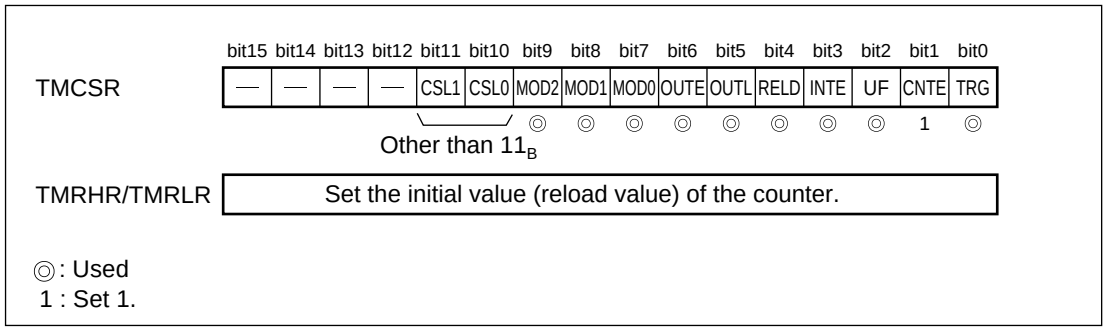
This section describes the 16-bit reload timer settings and counter operating status.

■ 16-Bit Reload Timer Settings

❍ Internal clock mode setting

The setting shown in Figure 11.6-1 "Internal Clock Mode Setting" is required to operate this timer as an interval timer.

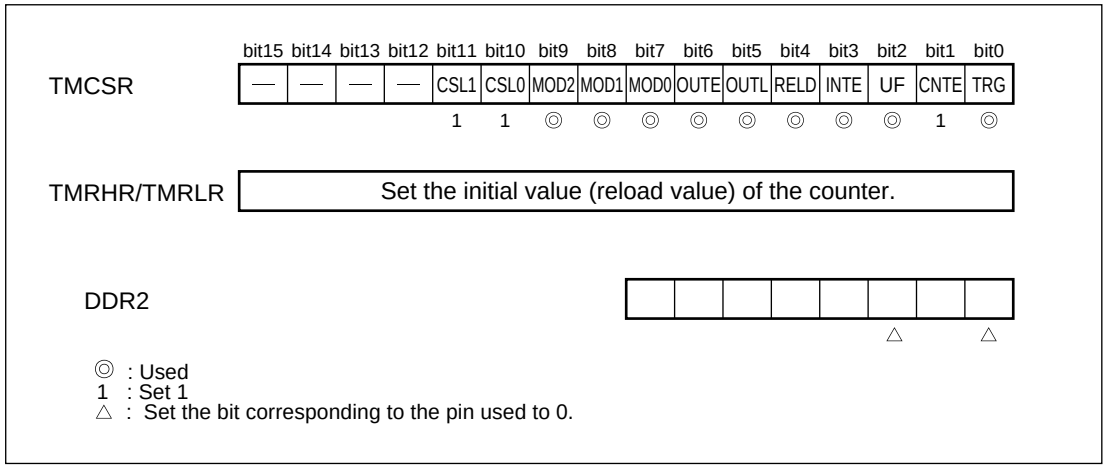
Figure 11.6-1 Internal Clock Mode Setting



❍ Event counter mode setting

he setting shown in Figure 11.6-2 "Event Counter Mode Setting" is required to operate this timer as an event counter.

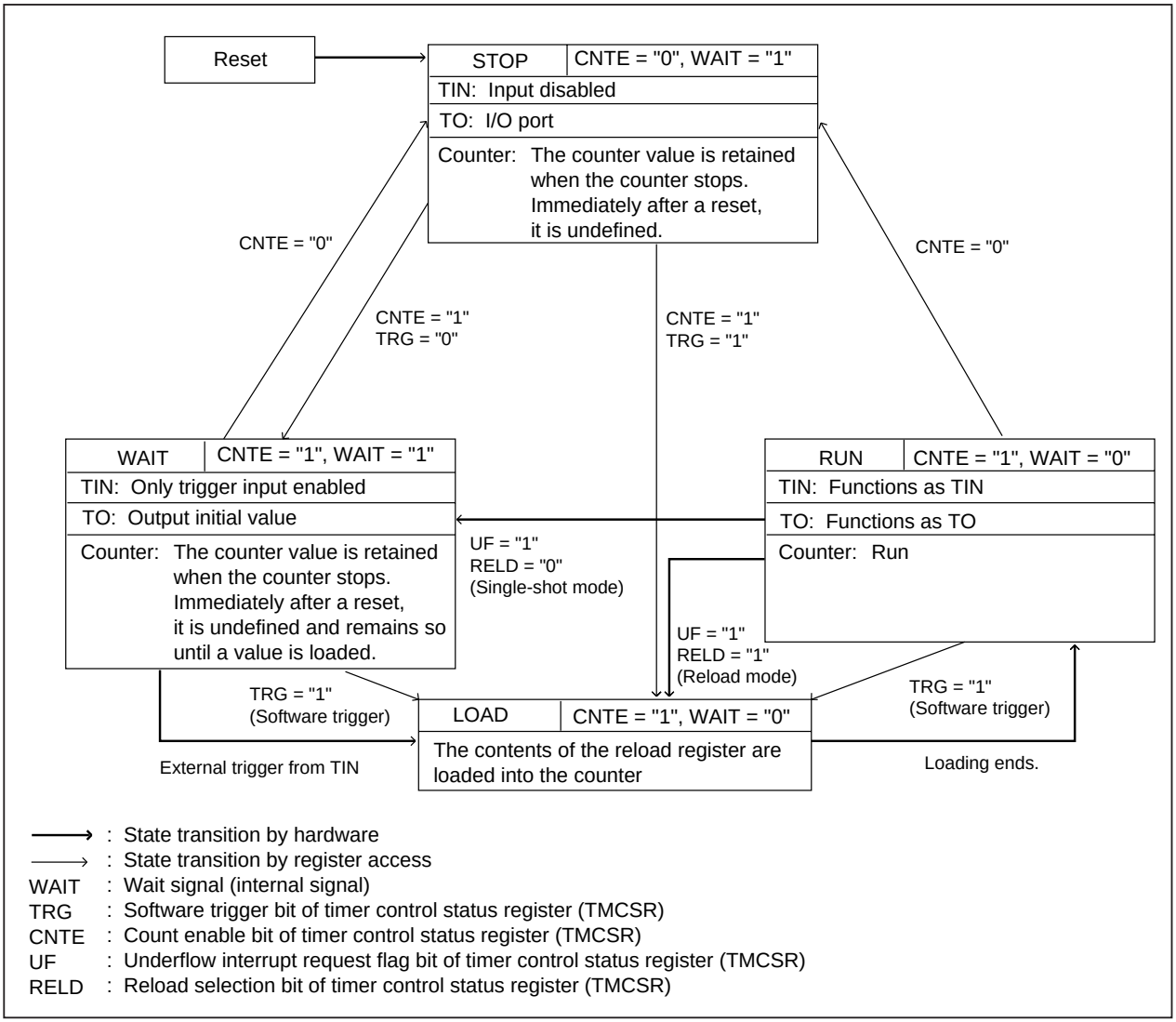
Figure 11.6-2 Event Counter Mode Setting



■ Counter Operating Status

The 16-bit down counter status is determined by the count enable bit (CNTE) values of the timer control status register (TMCSR) and the internal trigger wait signal value (WAIT). Figure 11.6-3 "Counter Status Transition" shows the relationship between the count enable bit (CNTE) values and the internal trigger wait signal (WAIT) values in the stop status (STOP status), trigger wait status (WAIT status), and running status (RUN status)

Figure 11.6-3 Counter Status Transition



11.6.1 Internal Clock Mode (Reload Mode)

The 16-bit reload timer count downs the 16-bit down counter in synchronization with the internal count clock and outputs an interrupt request when an underflow occurs (change from "0000_H" to "$FFFF_H$"). It can also output a toggle waveform from the timer output pin.

■ Operation in Internal Clock Mode (Reload Mode)

While the count enable bit (CNTE) of the timer control status register (TMCSR) is set to "1", a value stored in the 16-bit reload register (TMRLR0/TMRHR0, TMRLR1/TMRHR1) is loaded into the 16-bit down counter, and then countdown starts in one of the following cases: the software trigger bit (TRG) is set to "1", or a valid edge (rising, falling, or both edges) of the trigger input specified in the operation mode setting bits (MOD2, MOD1, and MOD0) is input to the TIN pins. When the CNTE bit and the software trigger bit are set to "1" simultaneously, countdown starts as soon as counting is enabled.

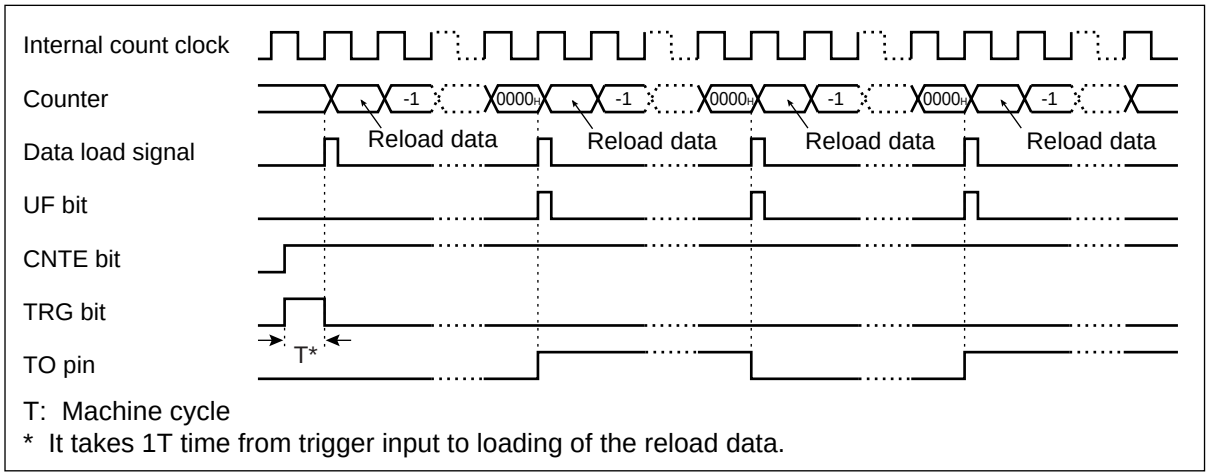
When an underflow of the 16-bit down counter occurs (change from "0000_H" to "$FFFF_H$"), a value stored in the 16-bit reload register (TMRLR0/TMRHR0, TMRLR1/TMRHR1) is loaded into the 16-bit down counter and countdown continues. An interrupt request is output when an underflow of the 16-bit down counter occurs while the underflow interrupt request flag bit (UF) of the timer control status register (TMCSR) is set to "1" and the underflow interrupt request enable bit (INTE) is set to "1".

The timer can also output from the TO pin a toggle waveform, which is inverted for each underflow.

❍ Software trigger operation

Counting starts when the software trigger bit (TRG) is set to "1" while the count enable bit (CNTE) of the timer control status register (TMCSR) is set to "1".

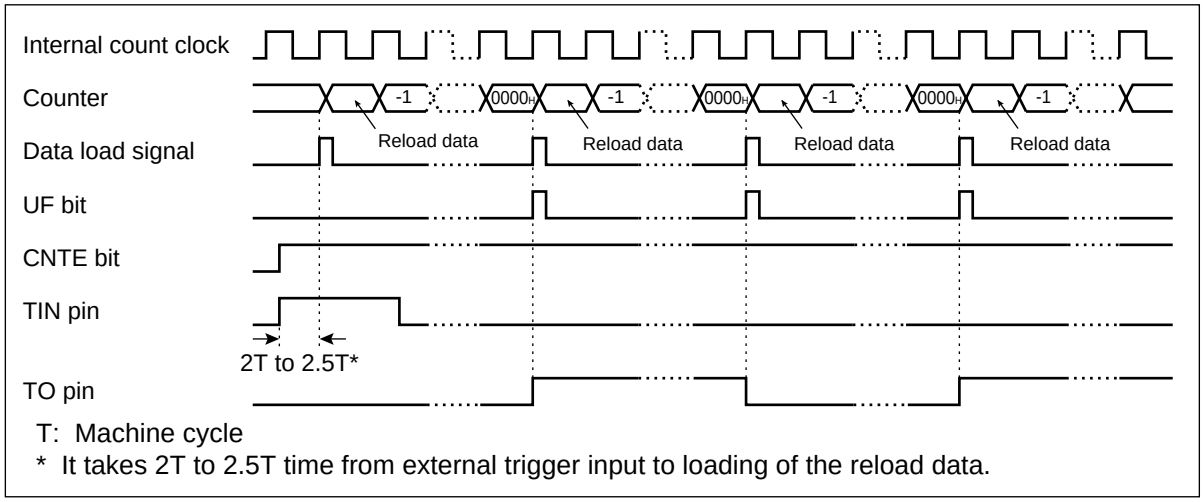
Figure 11.6-4 Count Operation in Reload Mode (Software Trigger Operation)



❍ **External trigger operation**

Counting starts when a valid edge (rising, falling, or both edges) of trigger input specified in the operation mode setting bits (MOD2, MOD1, and MOD0) is input to the TIN pins while the count enable bit (CNTE) of the timer control status register (TMCSR) is set to "1".

Figure 11.6-5 Count Operation in Reload Mode (External Trigger Operation)



T: Machine cycle

* It takes 2T to 2.5T time from external trigger input to loading of the reload data.

Note:

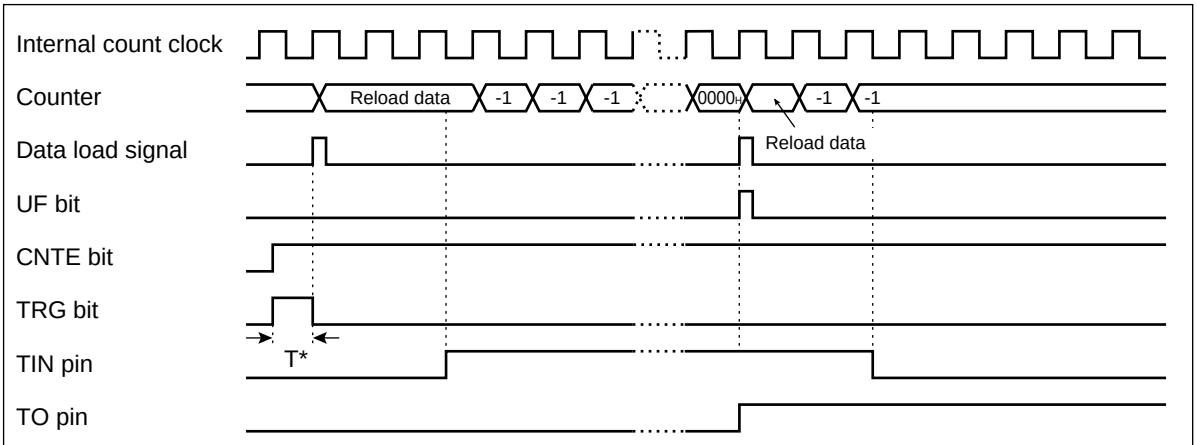
Specify 2/ϕ (ϕ : machine clock frequency) or more for the width of a trigger pulse to be input to the TIN pin.

❍ **Gate input operation**

Counting starts when the software trigger bit (TRG) is set to "1" while the count enable bit (CNTE) of the timer control status register (TMCSR) is set to "1".

Counting continues while a valid level of gate input ("L" or "H") specified in the operation mode setting bits (MOD2, MOD1, and MOD0) is input to the TIN pin.

Figure 11.6-6 Count Operation in Reload Mode (External Input Operation)



T: Machine cycle

* It takes 1T time from trigger input to loading of the reload data.

Note:

Specify 2/ϕ (ϕ : machine clock frequency) or more for the width of a gate input pulse to be input to the TIN pin.

11.6.2 Internal Clock Mode (Single-shot Mode)

The 16-bit reload timer count downs the 16-bit down counter in synchronization with the internal count clock and outputs an interrupt request when an underflow occurs (change from "0000_H" to "$FFFF_H$"). It can also output a square waveform from the T0 pin.

■ Internal Clock Mode (Single-Shot Mode)

When the software trigger bit (TRG) is set to "1" while the count enable bit (CNTE) of the timer control status register (TMCSR) is set to "1" or a valid edge (rising, falling, or both edges) of trigger input specified in the operation mode setting bits (MOD2, MOD1, and MOD0) is input to the TIN pins, a value stored in the 16-bit reload register (TMRLR0/TMRHR0, TMRLR1/TMRHR1) is loaded into the 16-bit down counter and countdown starts. When both the CNTE bit and the software trigger bit (TMCSR: TRG) are set to "1", countdown starts as soon as counting is enabled.

When an underflow of the 16-bit down counter occurs (change from "0000_H" to "$FFFF_H$"), the 16-bit down counter stops counting at the value "$FFFF_H$".

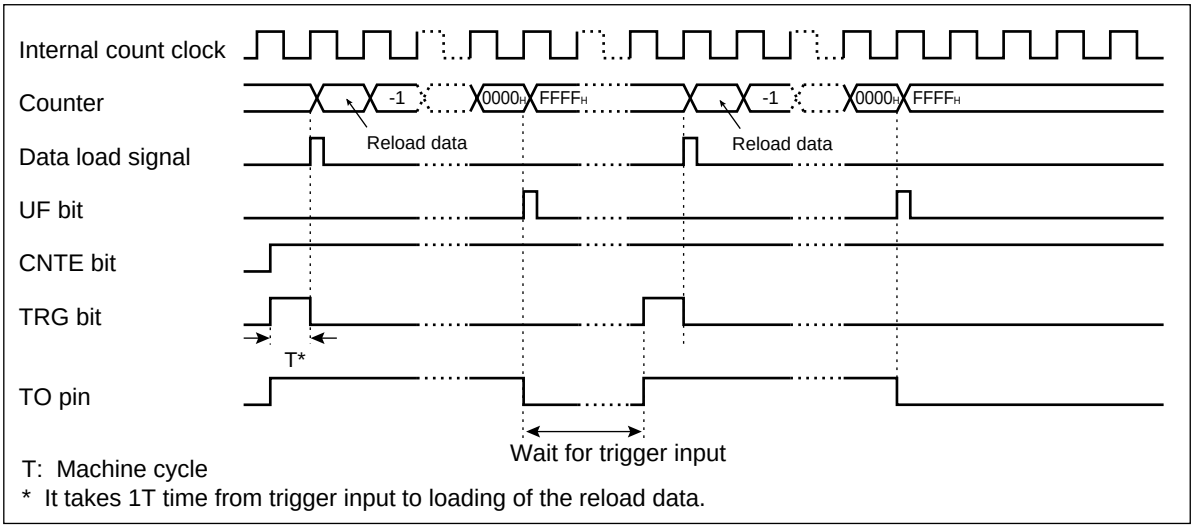
An interrupt request is output when an underflow of the 16-bit down counter occurs while the underflow interrupt request flag bit (UF) of the timer control status register (TMCSR) is set to "1" and the underflow interrupt request enable bit (INTE) is set to "1".

The timer can also output from the TO pin a rectangular waveform indicating that counting is in progress.

❍ Software trigger operation

Counting starts when the software trigger bit (TRG) is set to "1" while the count enable bit (CNTE) of the timer control status register (TMCSR) is set to "1".

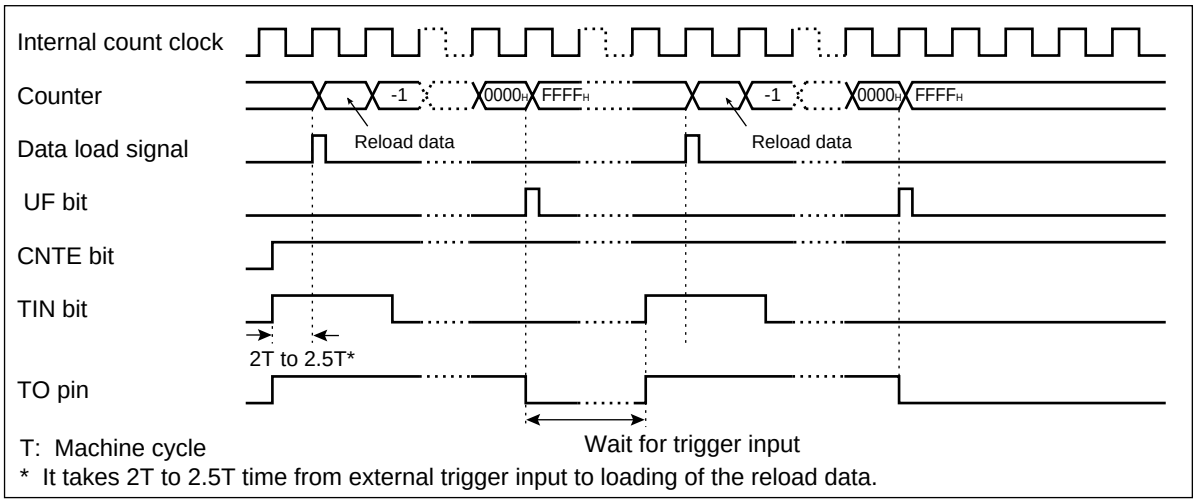
Figure 11.6-7 Count Operation in Single-Shot Mode (Software Trigger Operation)



❍ **External trigger input operation**

Counting starts when a valid edge (rising, falling, or both edges) of trigger input specified in the operation mode setting bits (MOD2, MOD1, and MOD0) is input to the TIN pins while the count enable bit (CNTE) of the timer control status register (TMCSR) is set to "1".

Figure 11.6-8 Count Operation in Single-Shot Mode (External Trigger Input Operation)



Note:

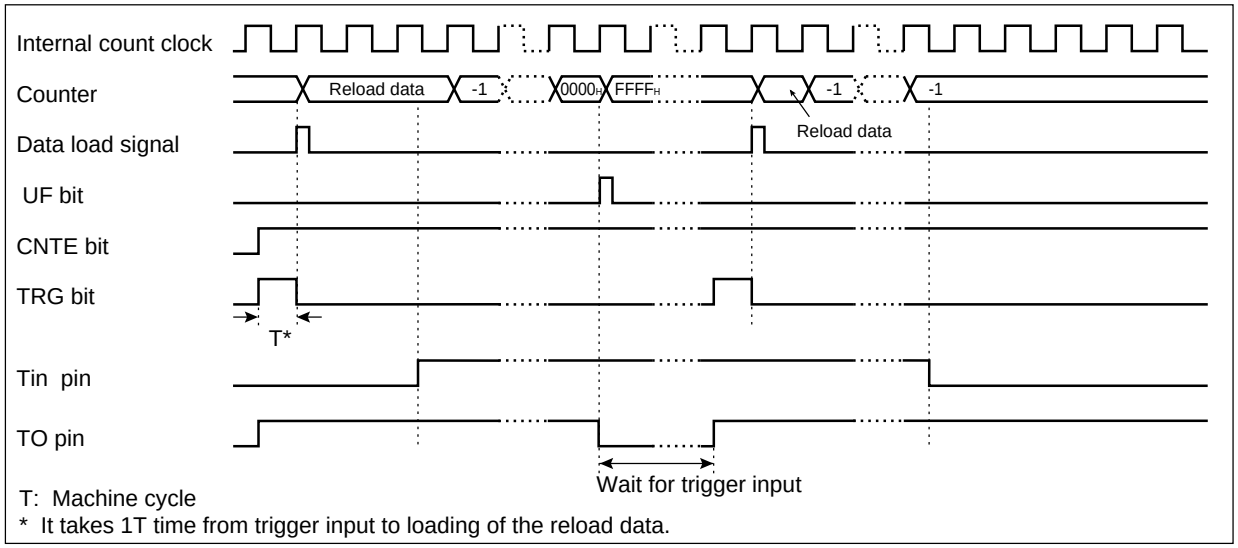
Specify 2/ϕ or more for the width of the trigger pulse input to the TIN pin.

❍ **External Gate input operation**

Counting starts when the software trigger bit (TRG) is set to "1" while the count enable bit (CNTE) of the timer control status register (TMCSR) is set to "1".

Counting continues while a valid level of gate input ("L" or "H") specified in the operation mode setting bits (MOD2, MOD1, and MOD0) is input to the TIN pin.

Figure 11.6-9 Count Operation in Single-Shot Mode (External Gate Input Operation)



Note:

Specify 2/ϕ (ϕ : machine clock frequency) or more for the width of a gate input pulse to be input to the TIN pin.

11.6.3 Event Count Mode

The 16-bit reload timer count downs the 16-bit down counter every time it detects a valid edge of pulse input to the TIN pin and outputs an interrupt request when an underflow occurs (change from "0000_H" to "$FFFF_H$"). It can also output a toggle or square waveform from the T0 pin.

■ Event Count Mode

When the software trigger bit (TRG) is set to "1" while the count enable bit (CNTE) of the timer control status register (TMCSR) is set to "1", a value stored in the 16-bit reload register (TMRLR0/TMRHR0, TMRLR1/TMRHR1) is loaded into the 16-bit down counter and countdown occurs every time a valid edge (rising, falling, or both edges) of pulse input to the TIN pin (external count clock) is detected. When both the CNTE bit and the software trigger bit (TRG) are set to "1", countdown starts as soon as counting is enabled.

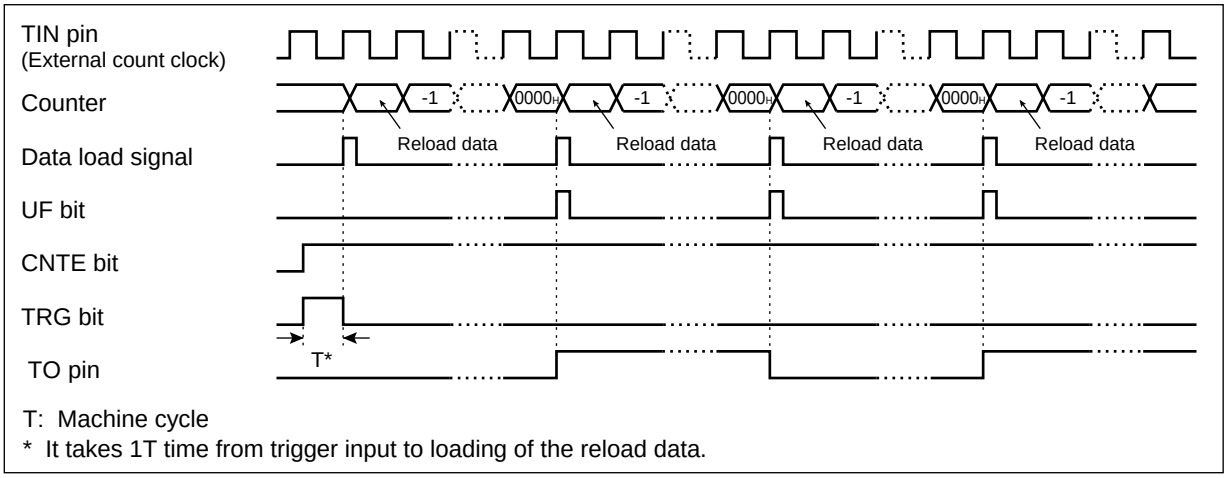
❍ Operation in reload mode

When an underflow of the 16-bit down counter occurs (change from "0000_H" to "$FFFF_H$"), a value stored in the 16-bit reload register (TMRLR0/TMRHR0, TMRLR1/TMRHR1) is loaded into the 16-bit down counter and countdown continues.

An interrupt request is output when an underflow of the 16-bit down counter occurs (change from "0000_H" to "$FFFF_H$") while the underflow interrupt request flag bit (UF) of the timer control status register (TMCSR) is set to "1" and the underflow interrupt request enable bit (INTE) is set to "1".

The timer can also output from the TO pin a toggle waveform, which is inverted for each underflow.

Figure 11.6-10 Count Operation in Reload Mode (Event Count Mode)



Note:

Specify $2^2/\phi$ or more for the H and L widths of the pulse input to the TIN pin.

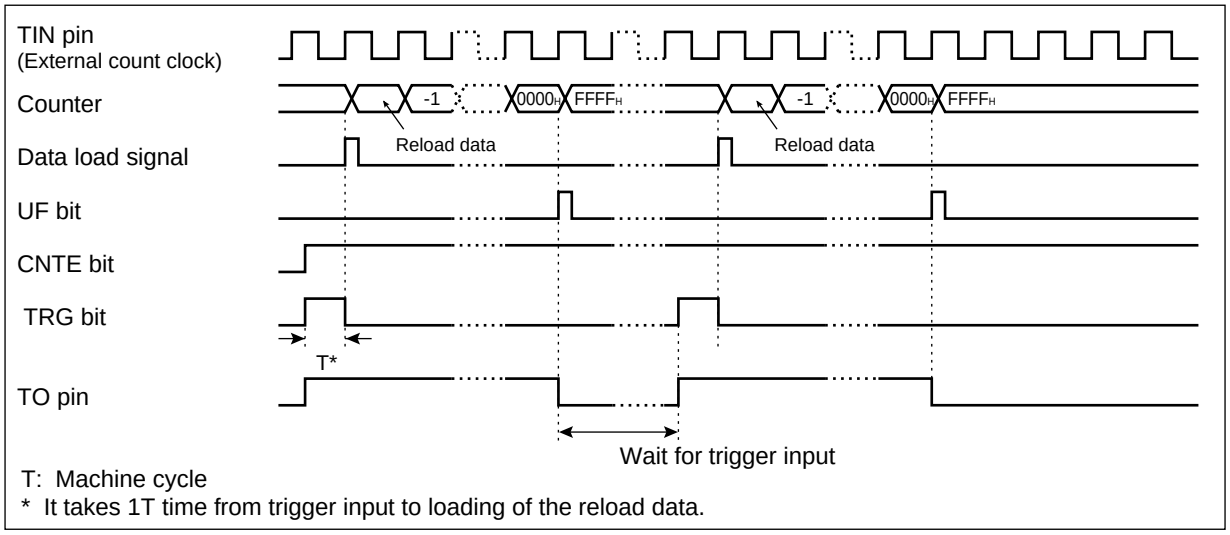
❍ Operation in single-shot mode

When an underflow of the 16-bit down counter occurs (change from "0000_H" to "$FFFF_H$"), the 16-bit down counter stops counting at the value "$FFFF_H$".

An interrupt request is output when an underflow of the 16-bit down counter occurs (change from "0000_H" to "$FFFF_H$") while the underflow interrupt request flag bit (UF) of the timer control status register (TMCSR) is set to "1" and the underflow interrupt request enable bit (INTE) is set to "1".

The timer can also output from the TO pin a rectangular waveform indicating that count

Figure 11.6-11 Counter Operation in Single-Shot Mode (Event Count Mode)



Note:

Specify $2^2/\phi$ or more for the H and L widths of the pluse input to the TIN pin.

11.7 Usage Notes on the 16-Bit Reload Timer

Notes on using the 16-bit reload timer are given below.

■ Usage Notes on the 16-Bit Reload Timer

❍ Notes on using a program for setting

- Write a value to the 16-bit reload register (TMRLR0, TMRHR0, TMRLR1, TMRHR1) when counting stops (TMCSR: CNTE = 0). Also, a value can be read from the 16-bit timer register (TMR0/TMR1) even during counting, but always be sure to use a word transfer instruction (MOVW A, dir, etc.).
- Counting must be stopped (TMCSR: CNTE = "0") when the count clock setting bits (CSL1 and CSL0) of the timer control register (TMCSR) are changed.

❍ Notes about interrupts

- When the UF bit of the timer control status register (TMCSR0: L, H, TMCSR1: L, H) is set to 1 and an interrupt request is enabled (TMCSR: INTE = 1), control cannot be returned from interrupt processing. Always clear the UF bit.
- 16-bit reload timer 0 shares the interrupt control register with 8-bit timer 0, 1, and 2 counter borrow. 16-bit reload timer 1 shares the interrupt control register with 16-bit free-running timer overflow. Thus, an interrupt with a smaller vector number takes precedence when multiple interrupts of the same level are output.

11.8 Sample Programs for the 16-Bit Reload Timer

This section contains sample programs for the 16-bit reload timer in internal clock mode and event count mode.

■ Sample Program in Internal Clock Mode

❍ Processing

- A 25-ms interval timer interrupt is generated with 16-bit reload timer 0.
- The timer is used in reload mode to repeatedly generate an interrupt.
- The 16-bit reload timer is activated using a software trigger, not an externally input trigger.
- EI^{2}OS is not used.
- 16 MHz is used for the machine clock, and 2µs is used for the count clock.

❍ Coding example

```
ICR09  EQU   0000B9H          ;Interrupt control register for
                               the 16-bit reload timer
TMCSR  EQU   000082H          ;Timer control status register
TMR    EQU   000084H          ;16-bit timer register
TMRLR  EQU   000084H          ;16-bit reload register
UF     EQU   TMCSR:2          ;Underflow interrupt request flag bit
CNTE   EQU   TMCSR:1          ;Counter operation enable bit
TRG    EQU   TMCSR:0          ;Software trigger bit
;-------Main program-----------------------------------------------
CODE         CSEG
START:
;       :                     ;Assumes that stack pointer (SP) has
                               already been initialized.
       AND   CCR,#0BFH        ;Interrupt disable
       MOV   I:ICR09,#00H     ;Interrupt level 0 (strongest)
       CLRB  I:CNTE           ;Temporary stopping of counter
       MOVW  I:TMRLR,#30D4H   ;Sets data for 25-ms timer.
       MOVW  I:TMCSR,#0000100000011011B
                              ;Interval timer operation, 2 µs clock
                              ;Disables external trigger and
                               external output.
                              ;Selects reload mode, and enables
                               interrupts.
                              ;Clears interrupt flag, and
                               starts counter.
       MOV   ILM,#07H         ;Sets ILM in PS to level 7
       OR    CCR,#40H         ;Interrupt enable
LOOP:  MOV   A,#00H           ;Endless loop
       MOV   A,#01H           ;
       BRA   LOOP             ;
;-------Interrupt program-------------------------------------------
WARI:
```

```
        CLRB I:UF               ;Clears underflow interrupt
                                 request flag.
;         :
;       User processing
;         :
        RETI                    ;Returns from interrupt

CODE    ENDS
;-------Vector setting----------------------------------------------
VECT     CSEG ABS=0FFH
         ORG  0FF88H            ;Sets vector for interrupt #29 (1DH).
         DSL  WARI
         ORG  0FFDCH            ;Sets reset vector
         DSL  START
         DB   00H               ;Sets single-chip mode.
VECT     ENDS
         END  START
```

■ Sample Program in Event Count Mode

❍ Processing

- When the rising edge of the pulse input to the external event input pin is counted 10,000 times with 16-bit reload timer/counter 0, an interrupt is generated.
- The timer operates in single-shot mode.
- External trigger input selects the rising edge.
- EI^{2}OS is not used.

❍ Coding example

```
ICR09 EQU  0000B9H              ;Interrupt control register for
                                 the 16-bit reload timer
TMCSR EQU   000082H             ;Timer control status register
TMR   EQU   000084H             ;16-bit timer register
TMRLR EQU   000084H             ;16-bit reload register
DDR2  EQU   000012H             ;Port direction register
UF    EQU   TMCSR:2             ;Underflow interrupt request flag bit
CNTE  EQU   TMCSR:1             ;Counter operation enable bit
TRG   EQU   TMCSR:0             ;Software trigger bit
;-------Main program-----------------------------------------------
CODE  CSEG
START:
;      :                        ;Assumes that stack pointer (SP) has
                                 already been initialized.
      AND   CCR,#0BFH           ;Interrupt disable
      MOV   I:ICR09,#00H        ;Interrupt level 0 (strongest)
      MOV   I:DDR2,#00H         ;Sets P20/TIN0 pin to input.
      CLRB  I:CNTE              ;Temporary stopping of counter
      MOVW  I:TMRLR,#2710H      ;Sets reload value to 10,000.
      MOVW  I:TMCSR,#0000110010001011B
                               ;Counter operation, external trigger,
                                rising disables edge and external
                                output.
                               ;Selects single-shot mode and enables
```

```
                                interrupts.
                               ;Clears interrupt flag, starts
                                counter.
       MOV  ILM,#07H           ;Sets ILM in PS to level 7.
       OR   CCR,#40H           ;Interrupt enable
LOOP:  MOV  A,#00H             ;Endless loop
       MOV  A,#01H             ;
       BRA  LOOP               ;
;-------Interrupt program-----------------------------------------
WARI:
       CLRB I:UF               ;Cears underflow interrupt request
                                flag.
;       :
;      User processing
;       :
       RETI                    ;Returns from interrupt.
CODE   ENDS
;-------Vector setting--------------------------------------------
VECT   CSEG ABS=0FFH
       ORG  0FF84H             ;Sets vector for interrupt #30 (1EH).
       DSL  WARI
       ORG  0FFDCH             ;Sets reset vector
       DSL  START
       DB   00H                ;Sets single-chip mode.
       VECT ENDS
       END  START
```

CHAPTER 12 MULTIFUNCTIONAL TIMERS

This chapter describes the operations of the multifunctional timers of MB90560/565 series.

12.1 Overview of Multifunctional Timers

The multifunctional timer unit enables 12 channels of independent waveform outputs based on the 16-bit free-running timer, and also enables the measurements of input pulse width and external clock periods.

■ Multifunctional Timer Configuration

❍ 16-bit free-running timer (one channel)

The 16-bit free-running timer consists of a 16-bit up-counter (timer data register (TCDT)), compare clear register (CPCLR), timer control status register (TCCS), and prescaler.

The counter output values of the 16-bit free-running timer are used as the base timer of the output compare and input capture.

- The following eight counter operating clocks are available to choose from:
  - $1/\phi$, $2/\phi$, $2^2/\phi$, $2^3/\phi$, $2^4/\phi$, $2^5/\phi$, $2^6/\phi$, $2^7/\phi$
  - ϕ: Machine clock frequency
- An interrupt can be output when an overflow of the counter value of the 16-bit free-running timer occurs, or when the counter value and the compare clear register (CPCLR) value of the 16-bit free-running timer match (TCCS: ICRE="1", MODE="1") and then the counter value of the 16-bit free-running timer is cleared to "0000_H".
- The counter value of the 16-bit free-running timer can be cleared to "0000_H" when the counter value is reset, the clear bit (SCLR) of the timer control status register (TCCS) is set to "1", the counter value and the compare clear register (CPCLR) value of the 16-bit free-running timer match (TCCS: MODE="1"), or the timer data register (TCDT) is set to "0000_H".

❍ Output compare (six channels)

The output compare unit consists of compare registers (OCCP0 to OCCP5), compare control registers (OCS0 to OCS5), and a compare output latch.

When the counter value of the 16-bit free-running timer matches that of a compare register (OCCP0 to OCCP5) value, the output level can be inverted and, at the same time, an interrupt can be output.

- Compare registers (OCCP0 to OCCP5) can be operated independently in six channels. Compare registers (OCCP0 to OCCP5) of each channel have a corresponding output pin and the lower compare control registers (OCS0, OCS2, OCS4) of each channel have an interrupt request flag.
- Two channels of compare registers (OCCP0 to OCCP5) can be used to invert the pin output.
- When the counter value and a compare register (OCCP0 to OCCP5) value of the 16-bit free-running timer match (OCS0, OCS2, OCS4: IOP0="1", IOP1="1"), an interrupt can be output (OCS0, OCS2, OCS4: IOE0="1", IOE1="1").
- An initial value can be set to the pin output of each channel.

❍ **Input capture (four channels)**

The input capture unit consists of input capture data registers (IPCP0 to IPCP3) corresponding to the external input pins (IN0 to IN3) and input capture control registers (ICS01 and ICS23).

When a valid edge of the signal that is input through an external input pin is detected, the counter value of the 16-bit free-running timer can be stored in an input capture data register (IPCP0 to IPCP3) and, at the same time, an interrupt can be output.

- Each channel of the input capture unit can be operated independently.
- A valid edge (rising edge, falling edge, and both edges) of the external signal can be set.
- An interrupt can be output by detecting a valid edge of an external signal (ICS01, ICS23: ICE0="1", ICE1="1", ICE2="1", ICE3="1").

❍ **8/16-bit PPG timer (8-bit: six channels, 16-bit: three channels)**

The 8/16-bit PPG timer consists of an 8-bit down-counter (PCNT), PPG control registers (PPGC0 to PPGC5), PPG clock control registers (PCS01, PCS23, PCS45), and PPG reload registers (PRLL0 to PRLL5, PRLH0 to PRLH5).

The 8/16-bit PPG timer works as an event timer when it is used as an 8/16-bit reload timer. It is also possible to output pulses of any frequency and duty ratio.

- 8-bit PPG mode
  - Each channel operates independently as an 8-bit PPG.
- 8-bit prescaler + 8-bit PPG mode
  - 8-bit PPG operation at any frequency is enabled by operating channel 0 (channel 2, channel 4) as an 8-bit prescaler and counting channel 1 (channel 3, channel 5) using the underflow output of channel 0 (channel 2, channel 4).
- 16-bit PPG mode
  - 16-bit PPG operation is enabled by linking channel 0 (channel 2, channel 4) and channel 1 (channel 3, channel 5).
- PPG operation
  - Pulse waveforms can be output at any frequency and duty ratio (ratio of the "H" level period to the "L" level period of pulse waveforms). The timer can also be used as a D/A converter by using an external circuit.

❍ **Waveform generator**

The waveform generation block consists of an 8-bit timer, 8-bit timer control registers (DTCR0 to DTCR2), 8-bit reload registers (TMRR0 to TMRR2), and a waveform control register (SIGCR).

This circuit can generate DC chopper output and the non-overlap 3-phase waveform output used for converter control by using the real-time output (RT0 to RT5) and the 8/16-bit PPG timer.

- By using the 8-bit timer as a dead-time timer, non-overlap waveforms can be generated by adding non-overlap time delays to the pulse output of the PPG timer (dead-time timer function).
- By using the 8-bit timer as a dead-time timer, non-overlap waveforms can be generated by adding non-overlap time delays to the real-time output (RT1, RT3, RT5) of the PPG timer (dead-time timer function).
- The GATE signal can be generated and the PPG timer operation can be controlled by matching (rising edge of the real-time output (RT)) of the counter value of 16-bit free-running timer and a compare register (OCCP0 to OCCP5) value of the output compare (GATE function).

- The GATE signal can be generated and the PPG timer operation can be controlled by activating the 8-bit timer when the counter value of 16-bit free-running timer and a compare register (OCCP0 to OCCP5) value of the output compare match (rising edge of the real-time output (RT)). (GATE function)
- RT00 to RT05 pin output can be controlled by the DTTI pin input. Pin control can be performed from an external device even if the oscillation is stopped by making the DTTI pin input "clockless" (The pin level can be set for each pin by using a program). However, it is necessary to set beforehand the I/O ports (P30 to P35) to output and an output value to the port3 data register (PDR3).

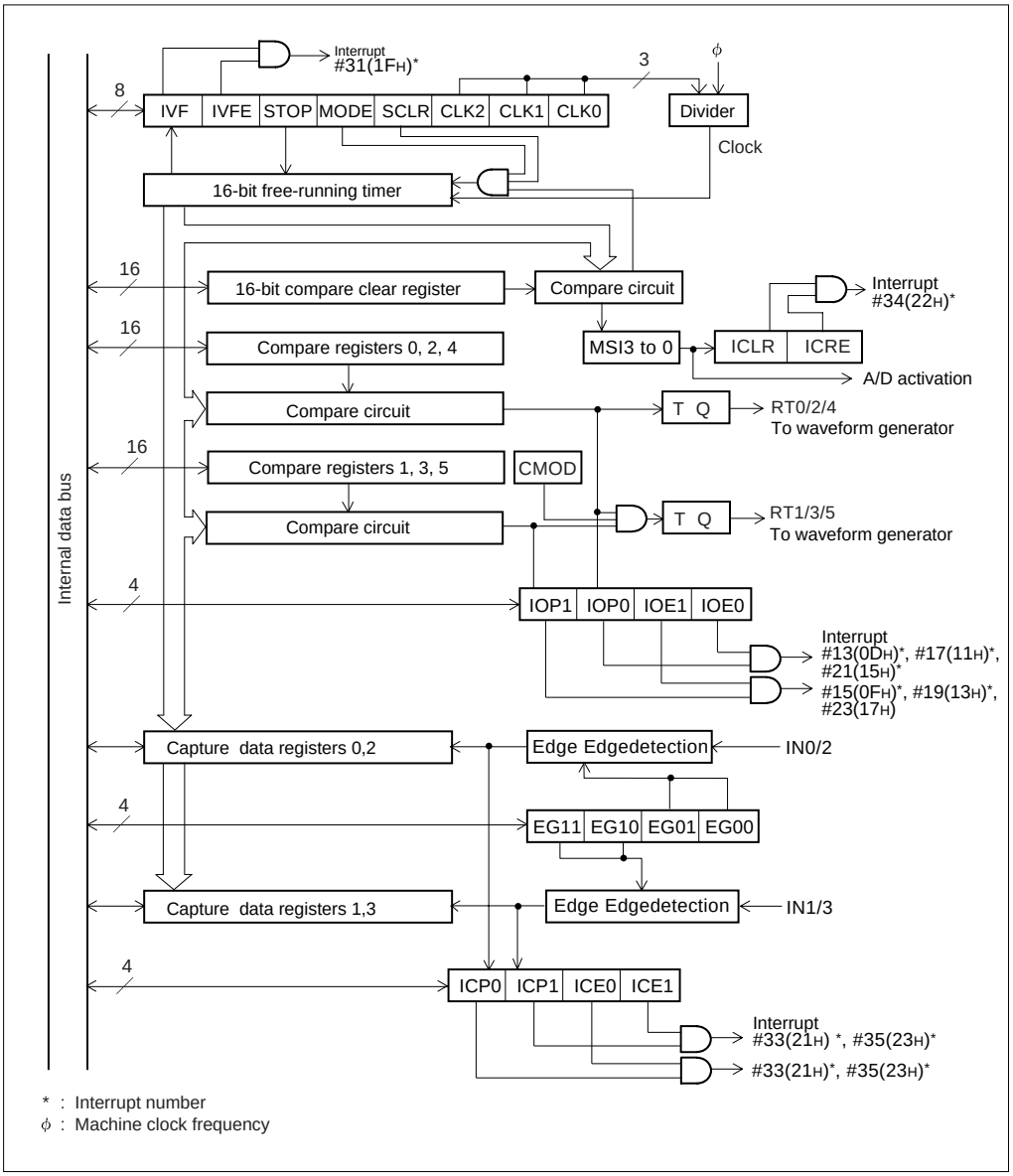
12.2 Configuration of Multifunctional Timers

The multifunctional timer unit consists of the following five blocks:

- **16-bit free-running timer (one channel)**
- **Output compare (six channels)**
- **Input capture (four channels)**
- **8/16-bit PPG timer (8-bit: six channels, 16-bit: three channels)**
- **Waveform generation block**

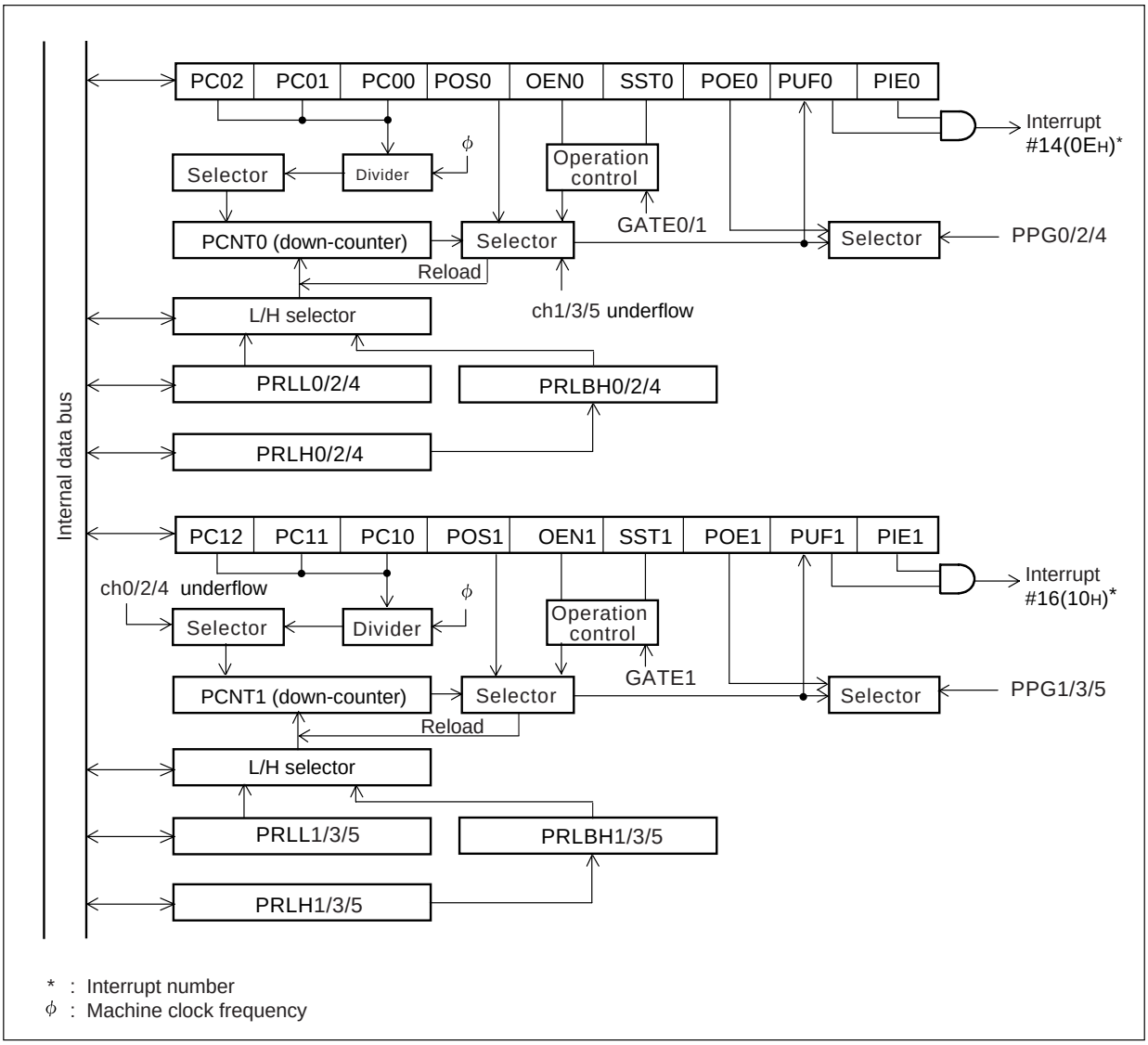
■ Real-Time I/O Unit Block Diagram

Figure 12.2-1 Real-Time I/O Unit Block Diagram



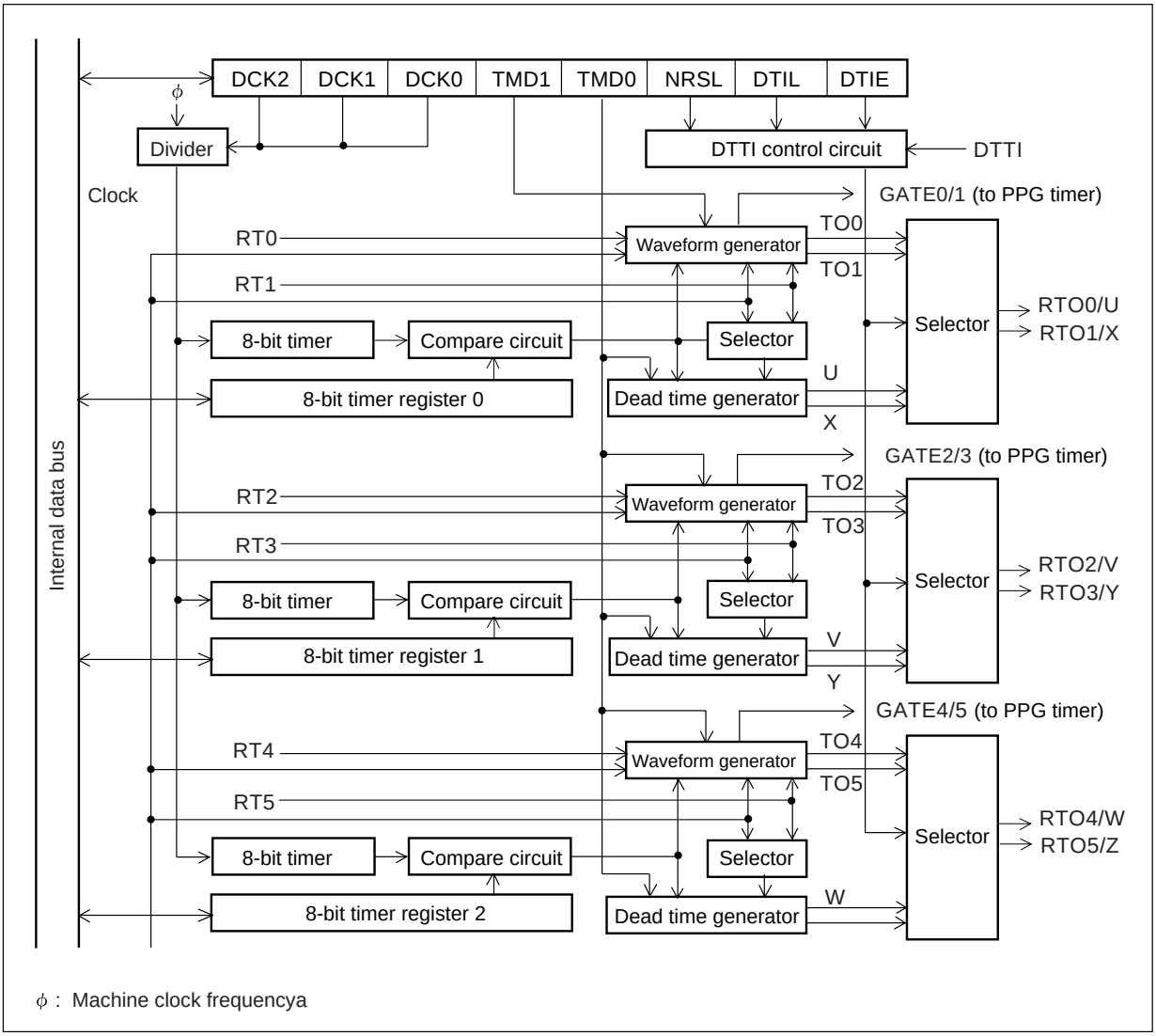
■ 8/16-Bit PPG Timer Block Diagram

Figure 12.2-2 8/16-Bit PPG Timer Block Diagram



■ Waveform Generator Block Diagram

Figure 12.2-3 Waveform Generator Block Diagram

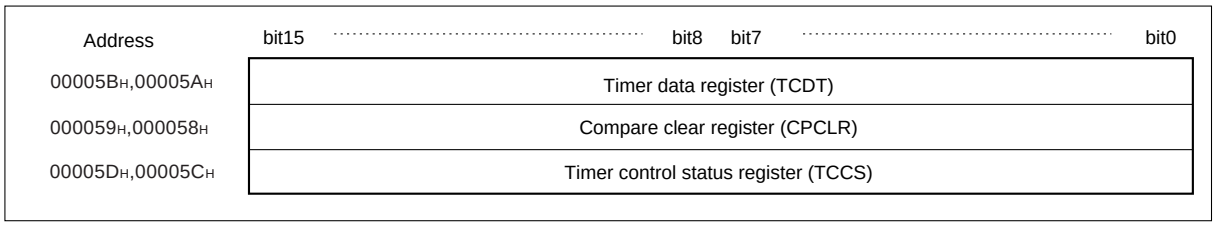


12.3 Multifunctional Timer Registers

This section presents the description of the multifunctional timer unit registers.

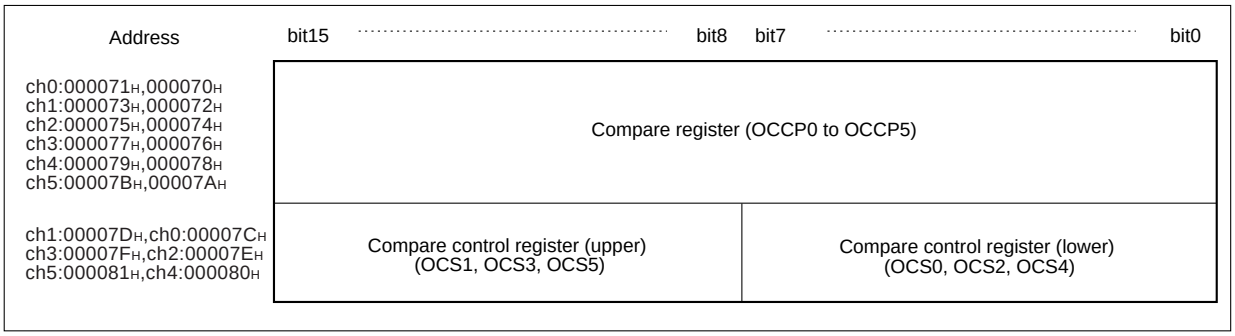
■ 16-Bit Free-Running Timer Registers

Figure 12.3-1 16-Bit Free-Running Timer Registers



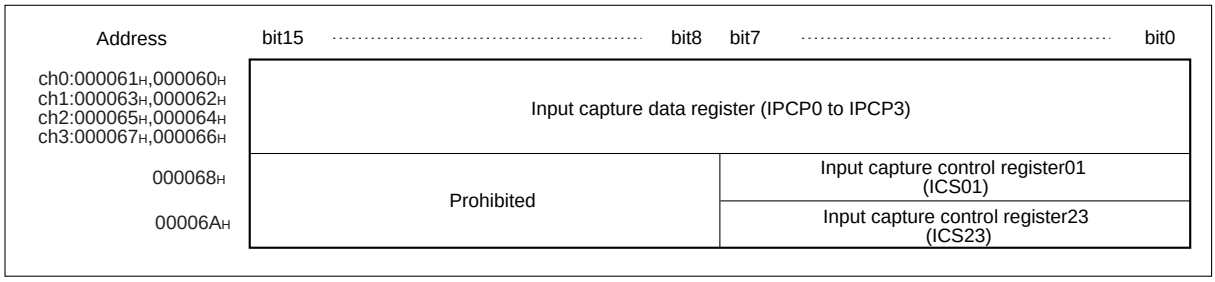
■ Output Compare Registers

Figure 12.3-2 Output Compare Registers



■ Input Capture Registers

Figure 12.3-3 Input Capture Registers



■ 8/16-Bit PPG Timer Registers

Figure 12.3-4 8/16-Bit PPG Timer Registers

Address	bit15 ····· bit8	bit7 ····· bit0
ch1:00003DH,ch0:00003CH ch3:000045H,ch2:000044H ch5:00004DH,ch4:00004CH	PPG control register (upper) (PPGC1, PPGC3, PPGC5)	PPG control register (lower) (PPGC0, PPGC2, PPGC4)
ch0,ch1:00003EH ch2,ch3:000046H ch4,ch5:00004EH	Prohibited	PPG clock control register (PCS01, PCS23, PCS45)
ch0:000039H,000038H ch1:00003BH,00003AH ch2:000041H,000040H ch3:000043H,000042H ch4:000049H,000048H ch5:00004BH,00004AH	PPG reload register (upper) (PRLH0 to PRLH5)	PPG reload register (lower) (PRLL0 to PRLL5)

■ Waveform Generator Registers

Figure 12.3-5 Waveform Generation Block Registers

Address	bit15 ····· bit8	bit7 ····· bit0
ch0:000051H,000050H ch1:000053H,000052H ch3:000055H,000054H	8-bit timer control register (DTCR0, DTCR1, DTCR2)	8-bit reload register (TMRR0, MTRR1, TMRR2)
000056H	Prohibited	Waveform control register (SIGCR)

12.3.1 16-bit Free-Running Timer Register

The 16-bit free-running timer uses the following three types of registers:

- **Timer data register (TCDT)**
- **Compare clear register (CPCLR)**
- **Timer control status register (TCCS)**

■ Timer Data Register (TCDT)

Figure 12.3-6 Timer Data Register (TCDT)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
00005B$_H$	T15	T14	T13	T12	T11	T10	T09	T08	00000000$_B$
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00005A$_H$	T07	T06	T05	T04	T03	T02	T01	T00	00000000$_B$
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Read/write

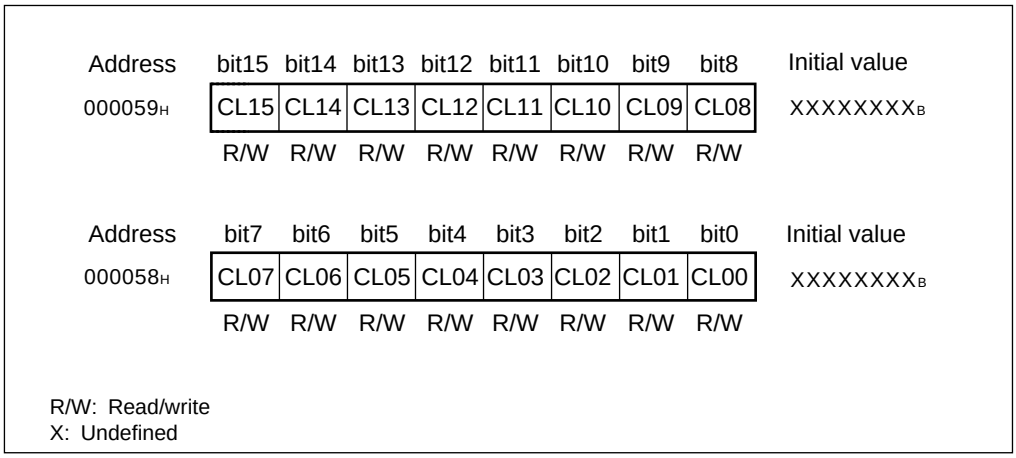
This is the counter of the 16-bit free-running timer. The counter value is cleared to "0000_H" by a reset. When setting a counter value to the timer data register (TCDT), stop the counter. (TCCS: STOP="1") For this register, set access in word units.

The counter value of the 16-bit free-running timer is cleared to "0000_H" when:

- Reset operation
- Setting "1" to the clear bit (SCLR) of the timer control status register (TCCS)
- Overflow of the counter value of the 16-bit free-running timer
- Match of the counter value and the compare clear register (CPCLR) value of the 16-bit free-running timer (TCCS: MODE="1")
- Setting "0000_H" to the timer data register (TCDT)

■ Compare Clear Register (CPCLR)

Figure 12.3-7 Compare Clear Register (CPCLR)



This is a 16-bit register to which the value to be compared with the counter value of the 16-bit free-running timer is set. The initial value of this register is undefined. Therefore, it is necessary to set a value to the compare clear register (CPCLR) before enabling an interrupt operation or clearing the counter value to "0000_H". For this register, set access in word units.

■ Timer Control Status Register (TCCS)

Figure 12.3-8 Timer Control Status Register (Upper) (TCCS)

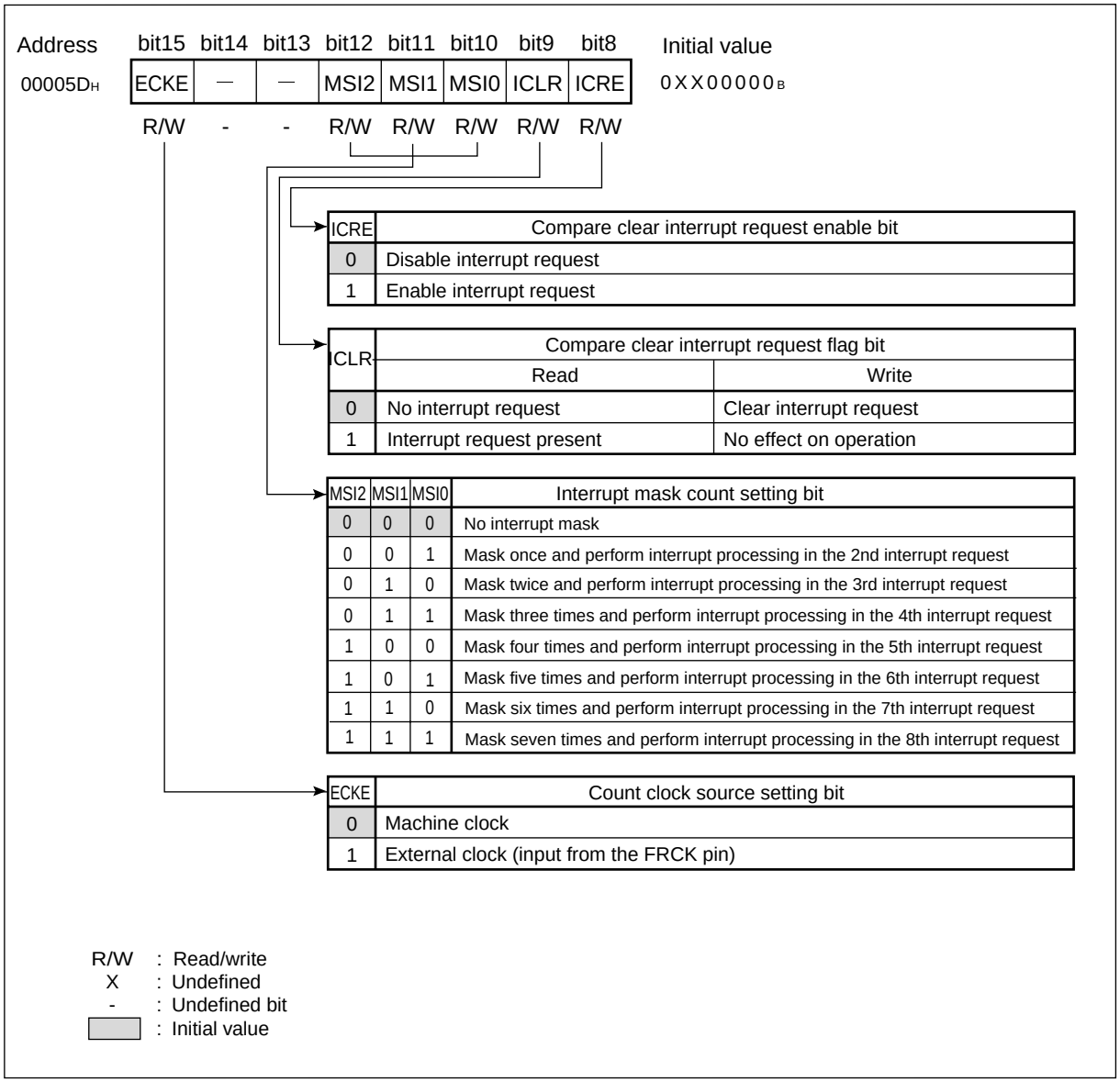


Table 12.3-1 Functions of Bits for Timer Control Status Register (Upper) (TCCS)

Bit name		Function
bit15	ECKE: Count clock source setting bit	This bit is used to specify whether the machine clock or an external clock is used as the count clock of the 16-bit free-running timer. The count clock source is changed immediately after the bit is set. Therefore, change the clock source while the output compare and input capture units are stopped. **Note:** To set the machine clock, set the count clock setting bits (CLK2 to CLK0). To set an external clock, set the FRCK pin as an input port (DDR1: bit15="0").
bit14 bit13	-: Undefined bit	If these bits are read, the values are undefined. Values set to these bits do not affect operation.
bit12 bit11 bit10	MSI2, MSI1, MSI0: Interrupt mask count setting bit	These bits set the number of times a compare clear interrupt is to be masked. The set value becomes the mask count directly. **Example:** • If "010_B" is set, interrupt processing is performed at the 3rd request after masking twice. • If "000_B" is set, no interrupt cause is masked.
bit9	ICLR: Compare clear interrupt request flag bit	This bit is a flag that requests an interrupt. If the counter value and the compare clear register (CPCLR) value of the 16-bit free-running timer match and the counter value is then cleared (bit4: MODE="1") to "0000_H", this bit is set to "1". When this bit is set to "1" while the compare clear interrupt enable bit (bit8: ICRE) is "1", an interrupt request is output. If this bit is "0", an interrupt request is cleared. If this bit is "1", operation is not affected. A read-modify-write instruction always reads "1" from this bit.
bit8	ICRE: Compare clear interrupt request enable bit	This bit enables an interrupt request. When the compare clear interrupt request flag bit (bit9: ICLR) is set to "1" while this bit is "1", an interrupt request is output.

■ Timer Control Status Register (Lower) (TCCS)

Figure 12.3-9 Timer Control Status Register (Lower) (TCCS)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00005C$_H$	IVF	IVFE	STOP	MODE	SCLR	CLK2	CLK1	CLK0	00000000$_B$
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

CLK2	CLK1	CLK0	Count clock setting bit				
			Count clock	ϕ=16MHz	ϕ = 8MHz	ϕ= 4MHz	ϕ= 1MHz
0	0	0	1 / ϕ	62.5ns	125 ns	250 ns	1 μs
0	0	1	2 / ϕ	125 ns	250 ns	500 ns	2 μs
0	1	0	2^2/ϕ	250 ns	500 ns	1 μs	4 μs
0	1	1	2^3/ϕ	500 ns	1 μs	2 μs	8 μs
1	0	0	2^4/ϕ	1 μs	2 μs	4 μs	16 μs
1	0	1	2^5/ϕ	2 μs	4 μs	8 μs	32 μs
1	1	0	2^6/ϕ	4 μs	8 μs	16 μs	64 μs
1	1	1	2^7/ϕ	8 μs	16 μs	32 μs	128 μs

SCLR	Clear bit
0	No effect on operation
1	Initialize the counter value to "0000$_H$"

MODE	Free-running timer initialization condition setting bit
0	Initialize by reset and clear bit
1	Initialize by reset, clear bit, and compare register value

STOP	Count stop bit
0	Enable the count (counter start)
1	Disable the count (counter stop)

IVFE	Free-running timer interrupt request enable bit
0	Disable an interrupt
1	Enable an interrupt

IVF	Free-running timer interrupt request flag bit	
	Read	Write
0	No interrupt request	Clear interrupt request
1	Interrupt request present	No effect on operation

R/W : No effect on operation
□ : Initial value
ϕ : Machine clock frequency

Table 12.3-2 Functions of Bits for Timer Control Status Register (Lower) (TCCS)

Bit name		Function
bit7	IVF: Free-running timer interrupt request flag bit	This bit is a flag that requests an interrupt. When an overflow of the counter value of the 16-bit free-running timer occurs, "1" is set. When this bit is set to "1" while the free-running timer interrupt enable bit (IVFE) is "1", an interrupt request is output. If this bit is "0", an interrupt request is cleared. If this bit is "1", operation is not affected. A read-modify-write instruction always reads "1" from this bit.
bit6	IVFE: Free-running timer interrupt request enable bit	This bit enables an interrupt request. When the free-running timer interrupt request flag bit (IVF) is set to "1" while this bit is "1", an interrupt request is output.
bit5	STOP: Count stop bit	This bit is used to stop the 16-bit free-running timer from counting. If this bit is "0", the counter of the 16-bit free-running timer is started. If this bit is "1", the counter of the 16-bit free-running timer is stopped. **Note:** When the counter of the 16-bit free-running timer stops, the output compare operation also stops.
bit4	MODE: Free-running timer initialization condition setting bit	This bit sets the initialization condition for the counter value of the 16-bit free-running timer. If "0" is set, the counter value is cleared to "0000_H" by reset and clear bit (SCLR="1"). If "1" is set, the counter value is cleared to "0000_H" by a match of the counter value and the compare clear register (CPCLR) value of the 16-bit free-running timer, in addition to reset and clear bit (SCLR="1"). **Note:** The counter value after detecting the initialization condition set to the MODE bit is cleared.
bit3	SCLR: Clear bit	This bit is used to clear the counter value to "0000_H" during operation of the counter of the 16-bit free-running timer. If this bit is "0", operation is not affected. If this bit is "1", the counter value is cleared to "0000_H". The read value is always "0". **Note:** To clear the counter value to "0000_H" while the counter of the 16-bit free-running timer is stopped (STOP="1"), set "0000_H" to the timer data register (TCDT).
bit2 bit1 bit0	CLK2, CLK1, CLK0: Count clock setting bit	These bits are used to set the count clock of the 16-bit free-running timer. The count clock is changed immediately after these bits are set. Therefore, set these bits while the output compare and input capture units are stopped.

12.3.2 Output Compare Registers

The output compare unit uses the following two types of registers:

- **Compare registers (OCCP0 to OCCP5)**
- **Compare control registers (OCS0 to OCS5)**

■ Compare Registers (OCCP0 to OCCP5)

Figure 12.3-10 Compare Registers (OCCP0 to OCCP5)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
ch0:000071$_H$ ch1:000073$_H$ ch2:000075$_H$ ch3:000077$_H$ ch4:000079$_H$ ch5:00007B$_H$	OP15	OP14	OP13	OP12	OP11	OP10	OP09	OP08	XXXXXXXX$_B$
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ch0:000070$_H$ ch1:000072$_H$ ch2:000074$_H$ ch3:000076$_H$ ch4:000078$_H$ ch5:00007A$_H$	OP07	OP06	OP05	OP04	OP03	OP02	OP01	OP00	XXXXXXXX$_B$
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W : Read/write
X : Undefined

This is a 16-bit register to which the value to be compared with the counter value of the 16-bit free-running timer is set. The initial value of this register is undefined. Therefore, it is necessary to set a value to the compare registers (OCCP0 to OCCP5) before enabling an interrupt operation or enabling the pin output of the output compare. For this register, set access in word units.

■ Compare Control Register (Upper) (OCS1, OCS3, OCS5)

Figure 12.3-11 Compare Control Register (Upper) (OCS1, OCS3, OCS5)

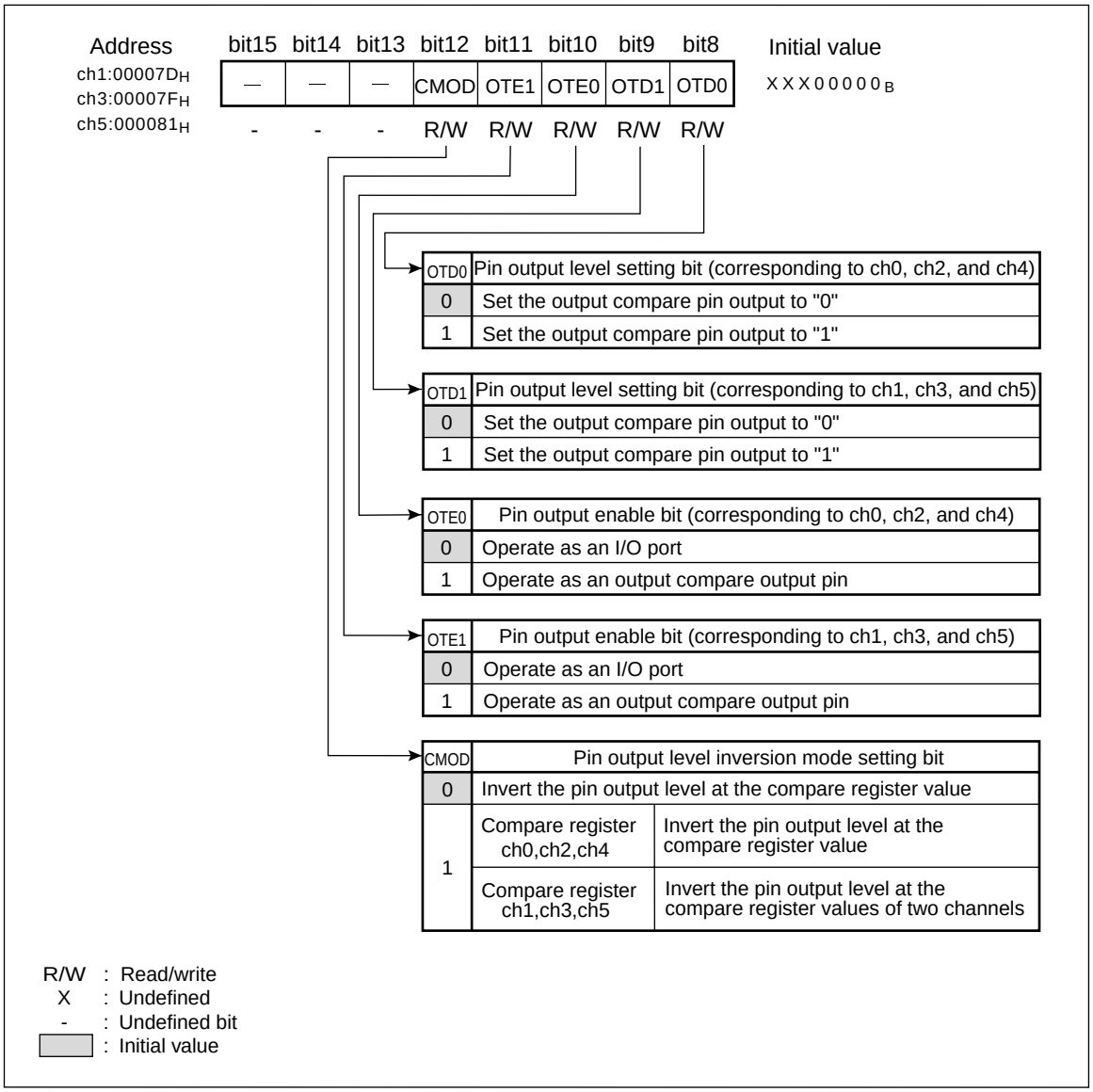


Table 12.3-3 "Functions of Bits for Compare Control Register (Upper) (OCS1, OCS3, and OCS5)" lists the functions of each bit of the compare control register (upper) (OCS1, OCS3, and OCS5). In Table 12.3-3 "Functions of Bits for Compare Control Register (Upper) (OCS1, OCS3, and OCS5)", the explanation refers to the compare control register (upper) (OCS1). For the compare control register (upper) (OCS3), replace ch0 with ch2 and ch1 with ch3. For the compare control register (upper) (OCS5), replace ch0 with ch4 and ch1 with ch5.

Table 12.3-3 Functions of Bits for Compare Control Register (Upper) (OCS1, OCS3, and OCS5)

Bit name		Function
bit15 bit14 bit13	-: Undefined bit	When these bits are read, the values read from them are undefined. The values set here do not affect operation.
bit12	CMOD: Pin output level inversion mode setting bit	This bit is used to set the inversion mode of the pin output level upon a match between the ch0/ch1 value of the compare register (OCCP) and the counter value of the 16-bit free-running timer. If this bit is "0", the RT0 pin corresponding to the compare register (OCCP) ch0 and compare register (OCCP) ch1 inverts the output level at the compare register (OCCP) value of each channel. If this bit is "1", the RT00 pin corresponding to the compare register (OCCP) ch0 inverts the output level at the compare register (OCCP) value of ch0, the RT01 pin corresponding to the compare register (OCCP) ch1 inverts the output level at the compare register (OCCP) value of ch0/ch1. If the compare register (OCCP) ch0 value and the compare register (OCCP) ch1 value are the same, operation will be the same as when only one compare register is used.
bit11 bit10	OTE1 (corresponding to ch1), OTE0 (corresponding to ch0): Pin output enable bit	These bits are used to enable output compare pin output. If this bit is "0", the pin corresponding to each channel operates as an I/O port. If this bit is "1", the pin corresponding to each channel operates as an output compare output pin. **Note:** To use a dead-time timer in the waveform generation block, the operation mode setting bit (TMD2) of the 8-bit timer control register (DTCR) needs to be set to "1".
bit9 bit8	OTD1 (corresponding to ch1), OTD0 (corresponding to ch0): Pin output level setting bit	These bits are used to set the pin output level of output compare. If this bit is "0", the output compare pin output is set to "0". If this bit is "1", the output compare pin output is set to "1". Before making a setting, stop the output compare operation. When these bits are read, the pin output values of output compare are read.

■ Compare Control Register (Lower) (OCS0, OCS2, OCS4)

Figure 12.3-12 Compare Control Register (Lower) (OCS0, OCS2, OCS4)

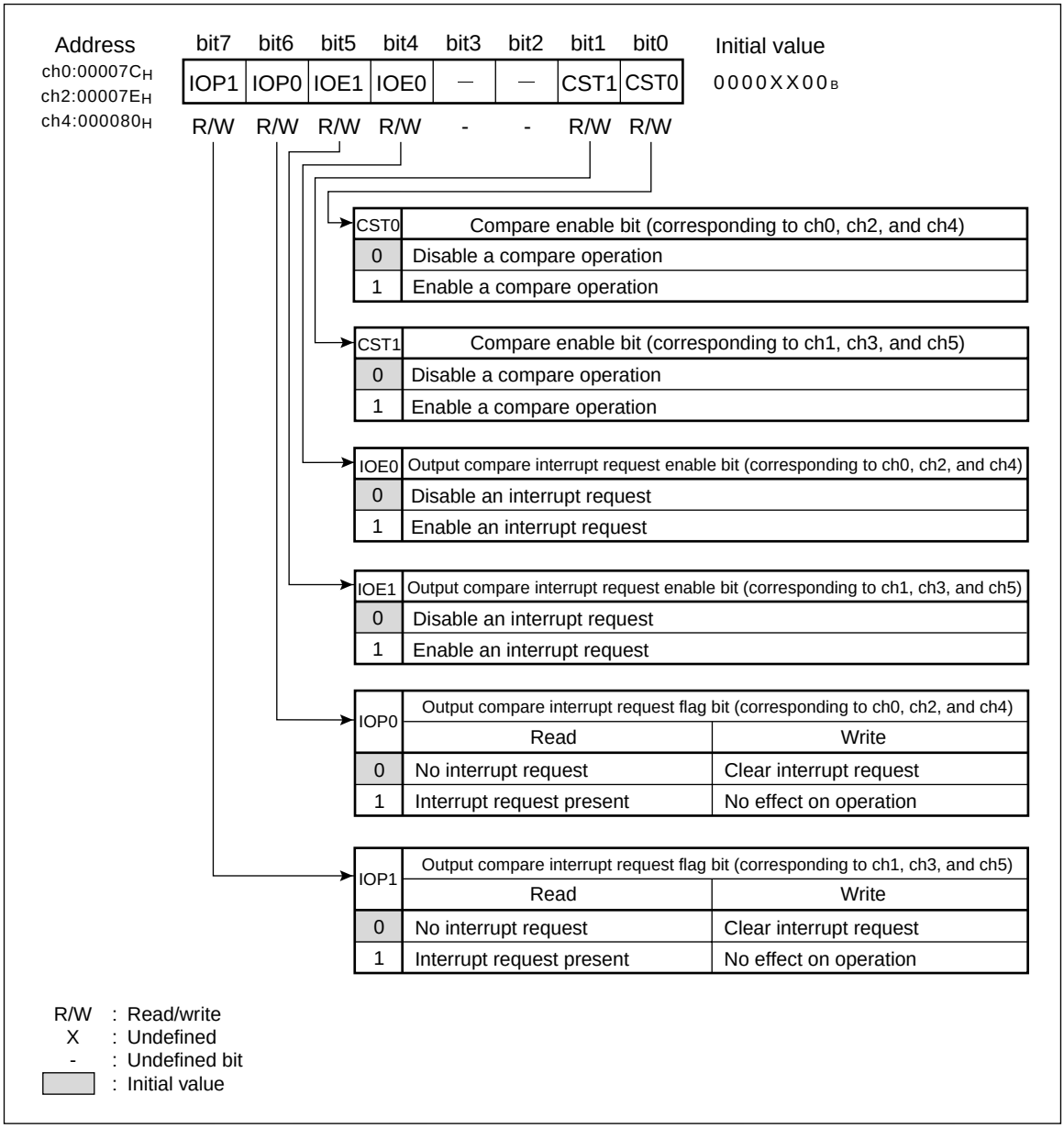


Table 12.3-4 "Functions of Bits for Compare Control Register (Lower) (OCS0, OCS2, and OCS4)" lists the functions of each bit of the compare control register (lower) (OCS0, OCS2, and OCS4). In Table 12.3-4 "Functions of Bits for Compare Control Register (Lower) (OCS0, OCS2, and OCS4)", the explanation refers to the compare control register (lower) (OCS0). For the compare control register (lower) (OCS2), replace ch0 with ch2 and ch1 with ch3. For the compare control register (lower) (OCS4), replace ch0 with ch4 and ch1 with ch5.

Table 12.3-4 Functions of Bits for Compare Control Register (Lower) (OCS0, OCS2, and OCS4)

Bit name		Function
bit7 bit6	IOP1 (corresponding to ch1), IOP0 (corresponding to ch0): Output compare interrupt request flag bit	This bit is a flag that requests an interrupt. If the compare register (OCCP) value and the counter value of the 16-bit free-running timer match, this bit is set to "1". When this bit is set to "1" while the output compare interrupt enable bit (IOE1, IOE0) is "1", an interrupt request is output. If this bit is "0", an interrupt request is cleared. If this bit is "1", operation is not affected. A read-modify-write instruction always reads "1" from this bit.
bit5 bit4	IOE1 (corresponding to ch1), IOE0 (corresponding to ch0): Output compare interrupt request enable bit	This bit enables an interrupt request. When the output compare interrupt request flag bit (IOP1, IOP0) is set to "1" while this bit is "1", an interrupt request is output.
bit3 bit2	-: Undefined bit	When these bits are read, the values read from them are undefined. The values set here do not affect operation.
bit1 bit0	CST1 (corresponding to ch1), CST0 (corresponding to ch0): Compare enable bit	This bit enables a compare operation between the compare register (OCCP) value and the counter value of the 16-bit free-running timer. Set a value to the compare register (OCCP) before enabling a compare operation. If this bit is "1", a compare operation is performed. **Note:** Because the output compare is synchronized with the 16-bit free-running timer, a compare operation also is stopped when the counter of the 16-bit free-running timer is stopped (TCCS: STOP=1).

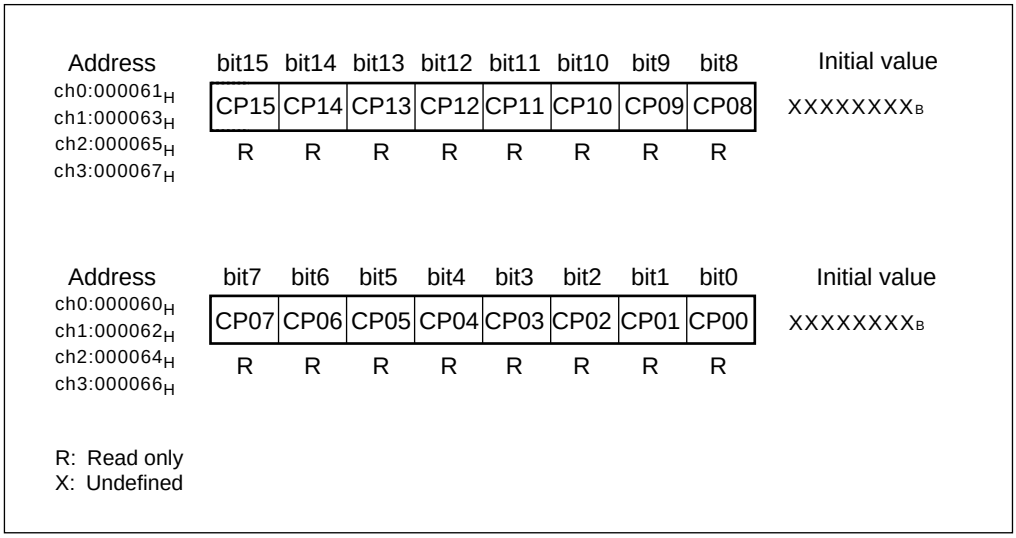
12.3.3 Input Capture Registers

The input capture unit uses the following two types of registers:

- **Input capture data registers (IPCP0 to IPCP3)**
- **Input capture control register (ICS01/23)**

■ Input Capture Registers (IPCP0 to IPCP3)

Figure 12.3-13 Input Capture Registers (IPCP0 to IPCP3)



An input capture register retains the value of the 16-bit free-running timer when a valid edge of the input waveforms from an external input pin (IN0 to IN3) is detected. For these registers, set access in word units. No setting can be made to the registers.

■ Input Capture Control Register01 (ICS01)

Figure 12.3-14 Input Capture Control Register01 (ICS01)

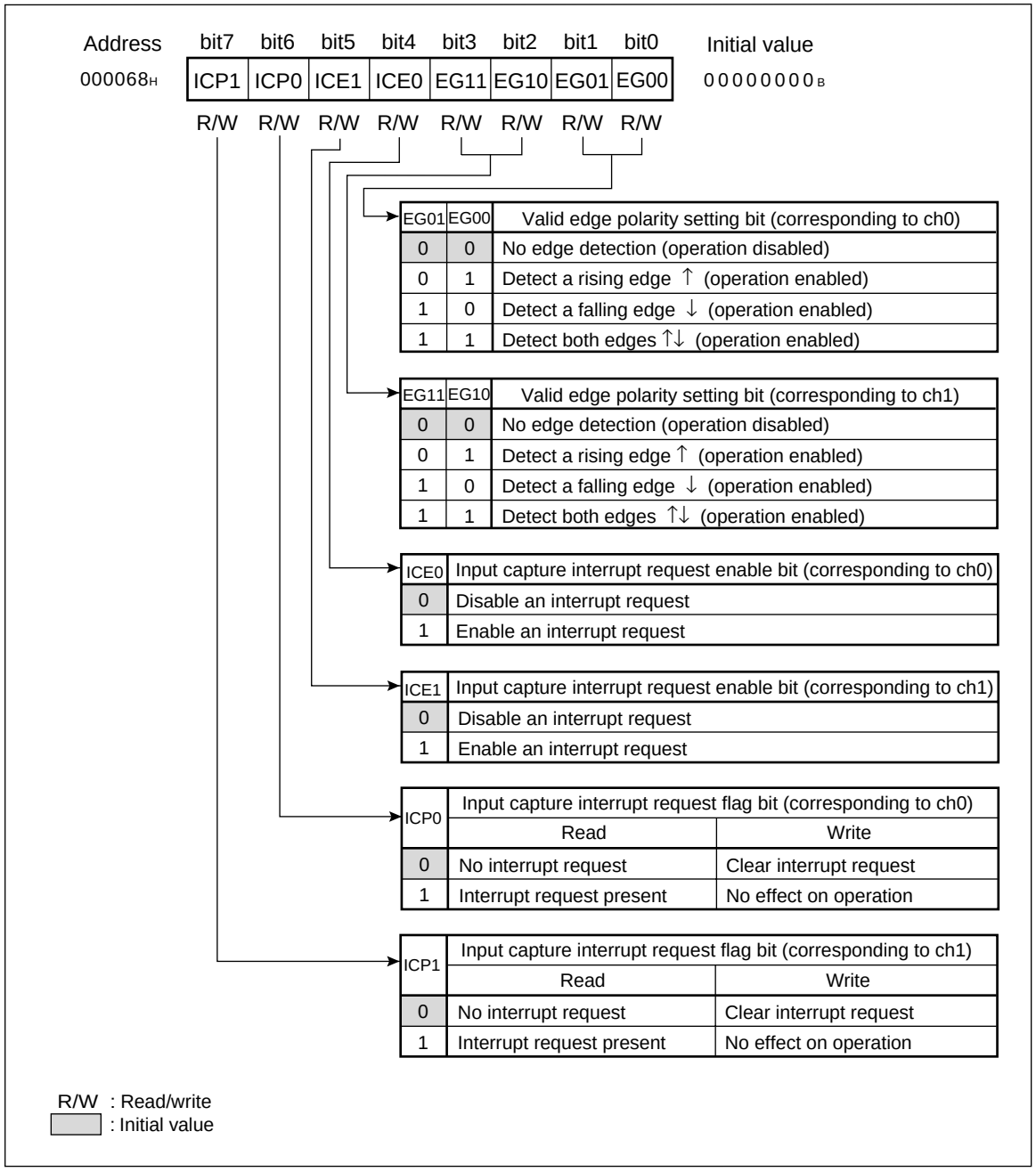


Table 12.3-5 Functions of Bits for Input Capture Control Register01 (ICS01)

Bit name		Function
bit7 bit6	ICP1 (corresponding to ch1), ICP0 (corresponding to ch0): Input capture interrupt request flag bit	This bit is a flag that requests an interrupt. If the valid edge polarity of an external input pin (IN0 to IN3) is detected, this bit is set to "1". When this bit is set to "1" while the input capture interrupt enable bit (ICE1, ICE0) is "1", an interrupt request is output. If this bit is "0", an interrupt request is cleared. If this bit is "1", operation is not affected. A read-modify-write instruction always reads "1" from this bit.
bit5 bit4	ICE1 (corresponding to ch1), ICE0 (corresponding to ch0): Input capture interrupt request enable bit	This bit enables an interrupt request. When the input capture interrupt request flag bit (ICP1, ICP0) is set to "1" while this bit is "1", an interrupt request is output.
bit3 bit2 bit1 bit0	EG11, EG10: (corresponding to ch1), EG01, EG00: (corresponding to ch0) Valid edge polarity setting bit	These bits are used to set the valid edge polarity of input waveforms and the input capture operation permission. If any of "01_B" to "11_B" is set, an input capture operation is enabled. The valid edge polarity of input waveforms can be selected from the rising-edge detection, falling-edge detection and detection of both edges. If "00_B" is set, an input capture operation is disabled and edge detection is disabled.

■ Input Capture Control Register23 (ICS23)

Figure 12.3-15 Input Capture Control Register23 (ICS23)

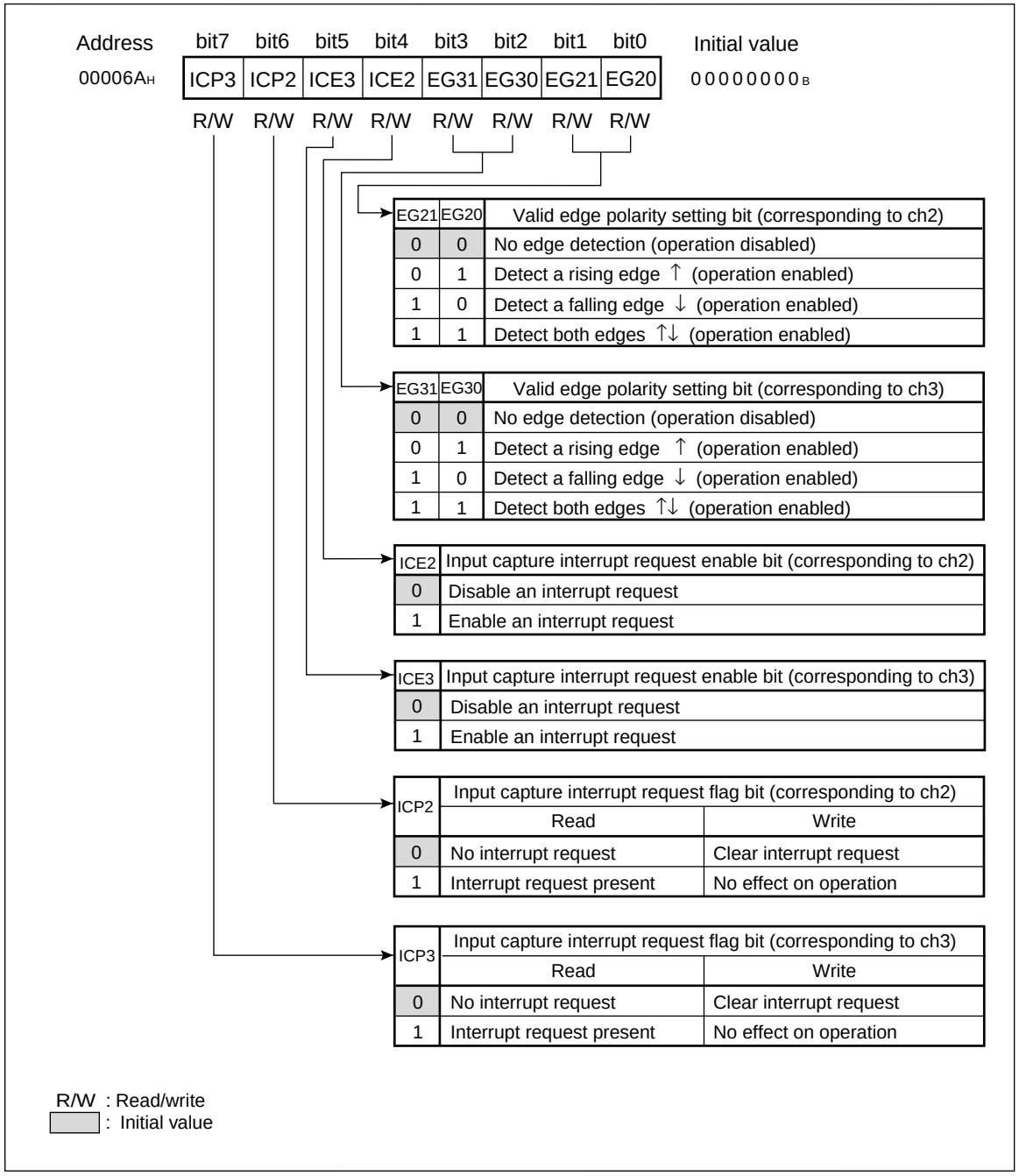


Table 12.3-6 Functions of Bits for Input Capture Control register23 (ICS23)

Bit name		Function
bit7 bit6	ICP3 (corresponding to ch3), ICP2 (corresponding to ch2): Input capture interrupt request flag bit	This bit is a flag that requests an interrupt. If the valid edge polarity of an external input pin (IN0 to IN3) is detected, this bit is set to "1". When this bit is set to "1" while the input capture interrupt enable bit (ICE3, ICE2) is "1", an interrupt request is output. If this bit is "0", an interrupt request is cleared. If this bit is "1", operation is not affected. A read-modify-write instruction always reads "1" from this bit.
bit5 bit4	ICE3 (corresponding to ch3), ICE2 (corresponding to ch2): Input capture interrupt request enable bit	This bit enables an interrupt request. When the input capture interrupt request flag bit (ICP3, ICP2) is set to "1" while this bit is "1", an interrupt request is output.
bit3 bit2 bit1 bit0	EG31, EG30: (corresponding to ch3), EG21, EG20: (corresponding to ch2) Valid edge polarity setting bit	These bits are used to set the valid edge polarity of input waveforms and the input capture operation permission. If any of "01_B" to "11_B" is set, an input capture operation is enabled. The valid edge polarity of input waveforms can be selected from the rising-edge detection, falling-edge detection and detection of both edges. If "00_B" is set, an input capture operation is disabled and edge detection is disabled.

12.3.4 8/16-bit PPG Timer Registers

The 8/16-bit PPG timer uses the following three types of registers:

- **PPG control registers (PPGC0 to PPGC5)**
- **PPG clock control registers (PCS01, PCS23, PCS45)**
- **PPG reload registers (PRLL0 to PRLL5, PRLH0 to PRLH5)**

■ PPG Control Registers (Upper) (PPGC1, PPGC3, PPGC5)

Figure 12.3-16 PPG Control Registers (Upper) (PPGC1, PPGC3, PPGC5)

Address
ch1:00003D$_H$
ch3:000045$_H$
ch5:00004D$_H$

bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
PEN1	SST1	POE1	PIE1	PUF1	MD01	MD00	RESV
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Initial value
00000001$_B$

RESV	Reservation bit
Always set "1"	

MD01	MD00	PPG timer operation mode setting bit (corresponding to ch0/ch1, ch2/ch3, and ch4/ch5)
0	0	8-bit PPG mode
0	1	8-bit prescaler + 8-bit PPG mode
1	0	Setting prohibited
1	1	16-bit PPG mode

PUF1	PPG timer interrupt request flag bit (corresponding to ch1, ch3, and ch5)	
	Read	Write
0	No interrupt request	Clear interrupt request
1	Interrupt request present	No effect on operation

PIE1	PPG timer interrupt request enable bit (corresponding to ch1, ch3, and ch5)
0	Disable an interrupt request
1	Enable an interrupt request

POE1	Pin output enable bit (corresponding to ch1, ch3, and ch5)
0	Operate as an I/O port
1	Operate as a PPG output pin

SST1	Activation condition setting bit (corresponding to ch1, ch3, and ch5)
0	Software activation
1	Hardware activation

PEN1	Operation enable bit (corresponding to ch1, ch3, and ch5)
0	Disable an operation
1	Enable an operation

R/W : Read/write
[ ] : Initial value

Table 12.3-7 Functions of Bits for PPG Control Register (Upper) (PPGC1, PPGC3, and PPGC5)

Bit name		Function
bit15	PEN1: Operation enable bit	This bit enables an operation when software activation is set. When this bit is set to "1" while the activation condition setting bit (SST1) is "0", the PPG timer starts to count. If the activation condition setting bit (SST1) is set to "1", operation is not affected.
bit14	SST1: Activation condition setting bit	This bit is used to set the activation conditions for the PPG timer. When the operation enable bit (PEN1) is set to "1" while this bit is "0", the PPG timer starts to count. If "1" is set, the operation enable bit (PEN1) does not affect operation. PPG operates during the "H" period of the GATE signal input from the waveform generation block.
bit13	POE1: Pin output enable bit	This bit enables pulse output of the PPG timer to the PPG pin. If this bit is "1", operation is not affected. A read-modify-write instruction always reads "1" from this bit.
bit12	PIE:1 PPG timer interrupt request enable bit	This bit enables an interrupt request. When the PPG timer interrupt request flag bit (PUF1) is set to "1" while this bit is "1", an interrupt request is output.
bit11	PIE1: PPG timer interrupt request flag bit	This bit is a flag that requests an interrupt. If an underflow of the counter value of the PPG timer occurs, this bit is set to "1". When this bit is set to "1" while the PPG timer interrupt enable bit (PIE1) is "1", an interrupt request is output. If this bit is "0", an interrupt request is cleared. If this bit is "1", operation is not affected. A read-modify-write instruction always reads "1" from this bit. **Note:** For 8-bit PPG mode and 8-bit prescaler + 8-bit PPG mode, "1" is set when an underflow ("00_H" --> "FF_H") of the counter value occurs. For 16-bit PPG mode, "1" is set when an underflow ("0000_H" --> "$FFFF_H$") of the counter value occurs.
bit10 bit9	MDO1, MDO0: PPG timer operation mode setting bit	These bits are used to set the operation mode of the PPG timer. **Note:** To set the software activation (SST1="0"), note the following points: • To set "01_B", do not set "0" to the operation enable bit (PEN0) and "1" to the operation enable bit (PEN1) • To set "11_B", set "1" or "0" to the operation enable bits (PEN0, PEN1) simultaneously in word transfer. To set the hardware activation (SST1="1"), note the following points: • To set "01_B", set "1" to the operation enable bit (PEN0) and then "1" to the activation condition setting bit (SST1). • Before setting "0" to the operation enable bit (PEN0), set "0" to the activation condition setting bit (SST1). • To set "11_B", set "1" or "0" to the activation condition setting bits (SST0, SST1) simultaneously in word transfer.
bit8	RESV: Reservation bit	Always specify "1".

■ PPG Control Registers (Lower) (PPGC0, PPGC2, PPGC4)

Figure 12.3-17 PPG Control Registers (Lower) (PPGC0, PPGC2, PPGC4)

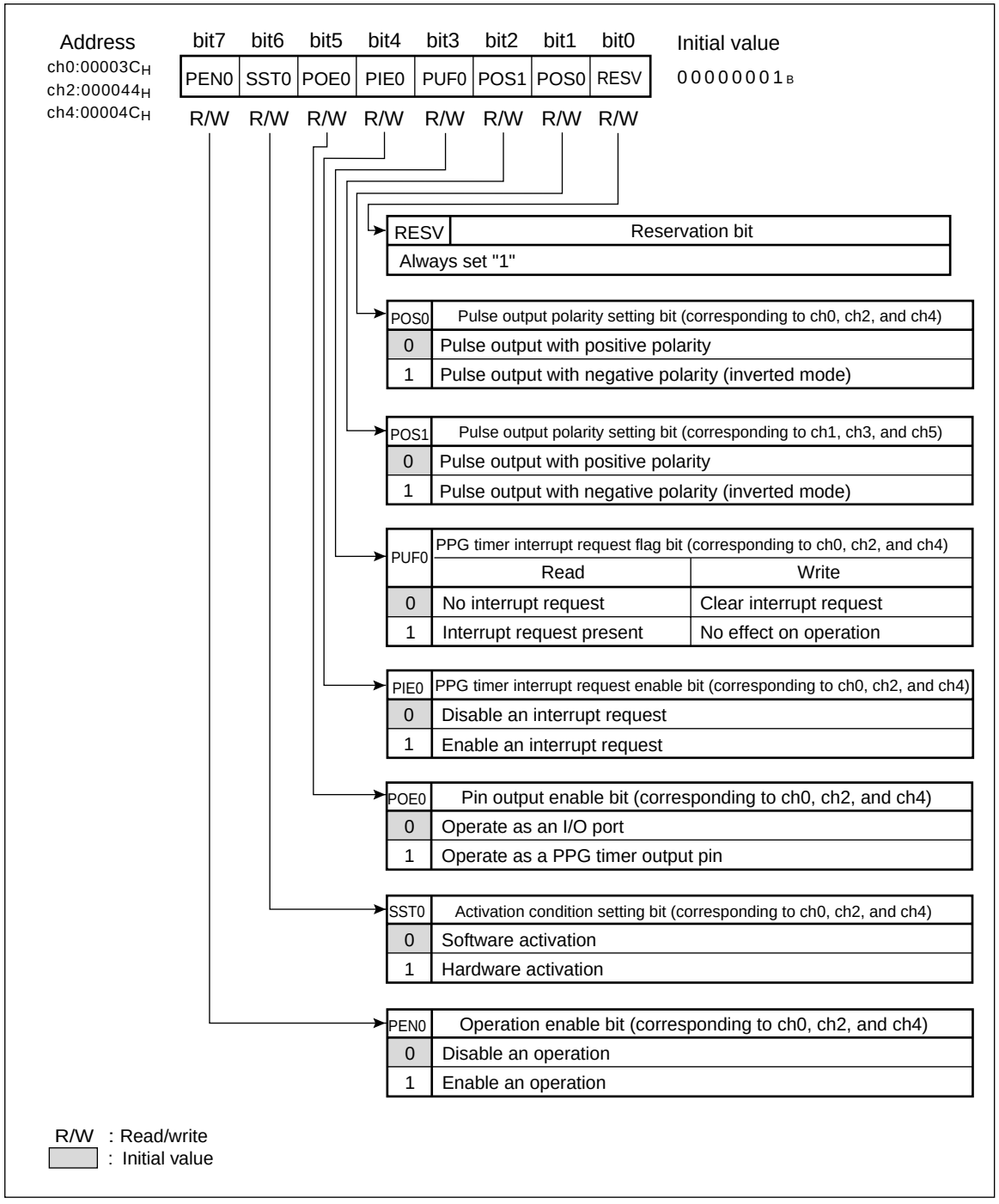


Table 12.3-8 Functions of Bits for PPG Control Register (Lower) (PPGC0, PPGC2, and PPGC4) [The PPG control registers (lower) correspond to ch0, ch2, and ch4]

Bit name		Function
bit7	PEN0: Operation enable bit	This bit enables an operation when software activation is set. When this bit is set to "1" while the activation condition setting bit (SST0) is "0", the PPG timer starts to count. If the activation condition setting bit (SST0) is set to "1", operation is not affected.
bit6	SST0: Activation condition setting bit	This bit is used to set the activation conditions for the PPG timer. When the operation enable bit (PEN0) is set to "1" while this bit is "0", the PPG timer starts to count. If "1" is set, the operation enable bit (PEN0) setting does not affect operation. PPG operates during the "H" period of the GATE signal input from the waveform generation block.
bit5	POE0: Pin output enable bit	This bit enables pulse output of the PPG timer to the PPG pin. If this bit is "0", the pin corresponding to each channel operates as an I/O port. If this bit is "1", the pin corresponding to each channel operates as an output pin of PPG.
bit4	PIE0: PPG timer interrupt request enable bit	This bit enables an interrupt request. When the PPG timer interrupt request flag bit (PUF0) is set to "1" while this bit is "1", an interrupt request is output.
bit3	PUF0: PPG timer interrupt request flag bit	This bit is a flag that requests an interrupt. If an underflow of the counter value of the PPG timer occurs, this bit is set to "1". When this bit is set to "1" while the PPG timer interrupt enable bit (PIE0) is "1", an interrupt request is output. If this bit is "0", an interrupt request is cleared. If this bit is "1", operation is not affected. A read-modify-write instruction always reads "1" from this bit. **Note:** For 8-bit PPG mode and 8-bit prescaler + 8-bit PPG mode, "1" is set when an underflow ("00_H" --> "FF_H") of the counter value occurs. For 16-bit PPG mode, "1" is set when an underflow ("0000_H" --> "$FFFF_H$") of the counter value occurs.
bit2	POS1: (corresponding to ch1, ch3, and ch5) Pulse output polarity setting bit	This bit is used to set the pulse output polarity of the PPG timer to an external pin.
bit1	POS0: (corresponding to ch0, ch2, and ch4) Pulse output polarity setting bit	This bit is used to set the pulse output polarity of the PPG timer to an external pin.
bit0	RESV: Reservation bit	Always specify "1".

■ PPG Clock Control Registers (PCS01, PCS23, PCS45)

Figure 12.3-18 PPG Clock Control Registers (PCS01, PCS23, PCS45)

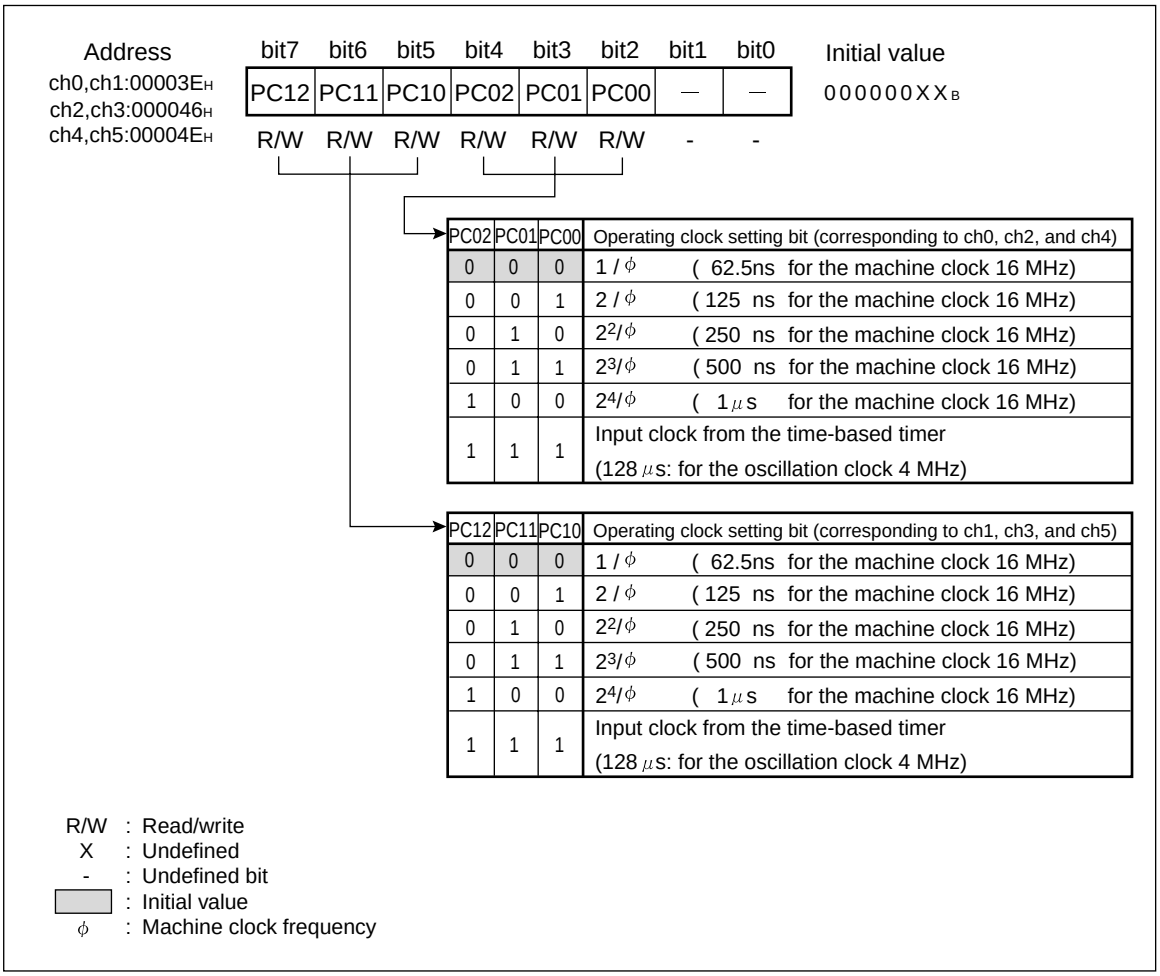
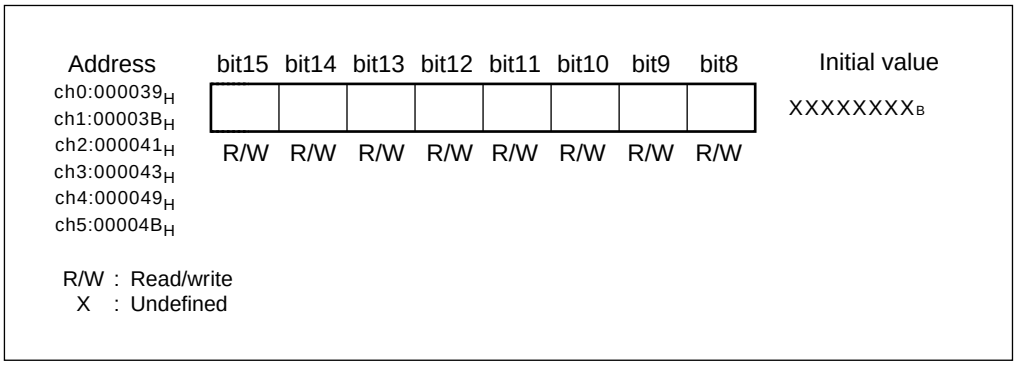


Table 12.3-9 Functions of Bits for PPG Clock Control Register (PCS01, PCS23, PCS45)

Bit name		Function
bit7 bit6 bit5	PC12, PC11, PC10: (corresponding to ch1, ch3, and ch5) Operating clock setting bit	These bits are used to set the operating clock of the 8-bit down-counter (PCNT). **Note:** In 8-bit prescaler + 8-bit PPG mode and 16-bit PPG mode, the PC12/ PC11/PC10 settings do not affect operation because PPG ch1 (ch3, ch5) operates by receiving the operating clock from PPG ch0 (ch2, ch4).
bit4 bit3 bit2	PC02, PC01, PC00: (corresponding to ch0, ch2, and ch4) Operating clock setting bit	These bits are used to set the operating clock of the 8-bit down-counter (PCNT).
bit1 bit0	-: Undefined bit	If these bits are read, the values are undefined. Values set to these bits do not affect operation.

■ PPG Reload Registers (Upper) (PRLH0 to PRLH5)

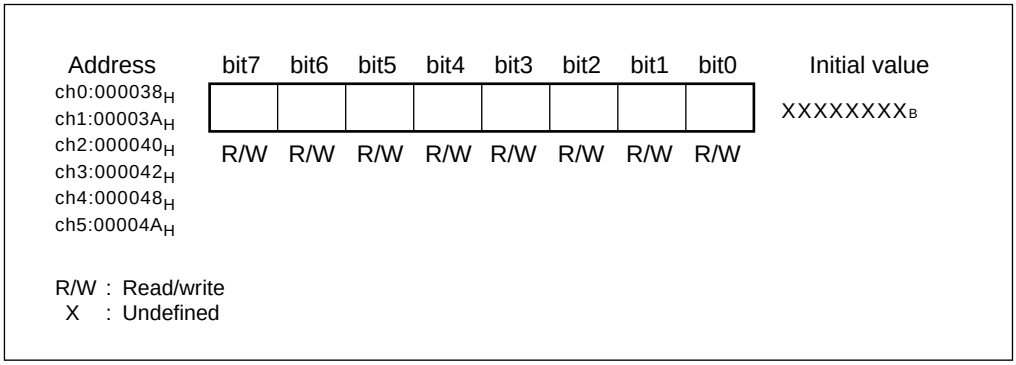
Figure 12.3-19 PPG Reload Registers (Upper) (PRLH0 to PRLH5)



These registers are used to set the upper side reload values of the 8-bit down-counter (PCNT).

■ PPG Reload Registers (Lower) (PRLL0 to PRLL5)

Figure 12.3-20 PPG Reload Registers (Lower) (PRLL0 to PRLL5)



These registers are used to set the lower side reload values of the 8-bit down-counter (PCNT).

Note:

If, in 8-bit prescaler + 8-bit PPG mode, different values are set to the PPG reload register (upper) (PRLH) and the PPG reload register (lower) (PRLL) of ch0 (ch2, ch4), PPG waveforms of ch1 (ch3, ch5) will be different in each cycle. Therefore, set the same value to the PPG reload register (upper) (PRLH) and the PPG reload register (lower) (PRLL) of ch0 (ch2, ch4).

12.3.5 Waveform Generator Registers

The waveform generator uses the following three types of registers:

- **8-bit timer control registers (DTCR0 to DTCR2)**
- **8-bit reload registers (TMRR0 to TMRR2)**
- **Waveform control register (SIGCR)**

■ 8-Bit Timer Control Registers (DTCR0 to DTCR2)

Figure 12.3-21 8-Bit Timer Control Registers (DTCR0 to DTCR2)

Address
ch0:000051H
ch1:000053H
ch2:000055H

bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
DMOD	GTEN	PGEN	TMIF	TMIE	TMD2	TMD1	TMD0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Initial value
00000000B

TMD2	TMD1	TMD0	Operation mode setting bit
0	0	0	Stop an operation of the waveform generation block. The 8-bit timer can be used as a reload timer.
0	0	1	Output pulses of the PPG timer during the "H" period of each RT. Make settings for the PPG timer to be output in the PPG timer setting bits (PGS1, PGS0) of the waveform control register (SIGCR).
0	1	0	Start the 8-bit timer on the rising edge of each RT to run the PPG timer until the timer stops (timer mode).
1	0	0	Generate non-overlap waveforms from the RT1 waveforms (dead-time timer mode).
1	1	1	Start and stop the PPG timer using the RT1 (RT3, RT5) output as the GATE signal to generate non-overlap waveforms from the PPG output waveforms (dead-time timer mode).

TMIE	8-bit timer interrupt request enable bit
0	Disable an interrupt request
1	Enable an interrupt request

TMIF	8-bit timer interrupt request flag bit (corresponding to ch0, ch2, and ch4)	
	Read	Write
0	No interrupt request	Clear interrupt request
1	Interrupt request present	No effect on operation

PGEN	Pulse output enable bit
0	Disable pulse output of the PPG timer
1	Enable pulse output of the PPG timer

GTEN	GATE signal output setting bit
0	No GATE signal output (asynchronous mode)
1	GATE signal output present (synchronous mode)

DMOD	Waveform output polarity setting bit
0	Waveform output with positive polarity
1	Waveform output with negative polarity

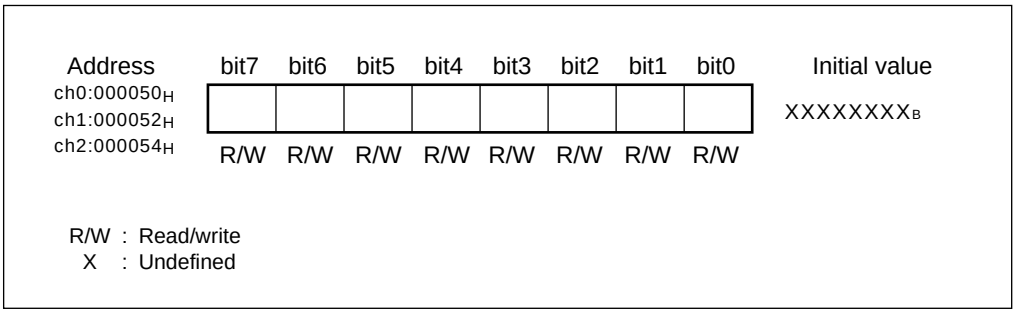
R/W : Read/write
□ : Initial value

Table 12.3-10 Functions of Bits for 8-Bit Timer Control Registers (DTCR0 to DTCR2)

Bit name		Function
bit15	DMOD: Waveform output polarity setting bit	This bit is used to set the waveform output polarity when the 8-bit timer is run as a dead-time timer. When the 8-bit timer is not run as a dead-time timer (TMD2="0"), the DMOD bit setting does not affect operation.
bit14	GTEN: GATE signal output setting bit	This bit is used to set the GATE signal output for PPG timer operation.
bit13	PGEN: Pulse output enable bit	This bit enables pulse output of the PPG timer to the RT0 pin.
bit12	TMIF: 8-bit timer interrupt request flag bit	This bit is a flag that requests an interrupt. If an underflow of the counter value of the 8-bit timer occurs, this bit is set to "1". When this bit is set to "1" while the 8-bit timer interrupt enable bit (TMIE) is "1", an interrupt request is output. If this bit is "0", an interrupt request is cleared. If this bit is "1", operation is not affected. A read-modify-write instruction always reads "0" from this bit.
bit11	TMIE: 8-bit timer interrupt enable bit	This bit is used to start the 8-bit timer and to enable an interrupt request. If "1" is set, the operation of the 8-bit timer is started. When the 8-bit timer interrupt request flag bit (TMIF) is set to "1", an interrupt request is output. **Note:** The 8-bit timer can be used as a reload timer only if "000_B" is set to bit10 to bit8: TMD2 to TMD0.
bit10 bit9 bit8	TMD2, TMD1, TMD0: Operation mode setting bit	These bits are used to set the operation mode of the waveform generation block. **Note:** Settings other than those described here may lead to malfunctioning. Never make any other settings. To set "111_B", set 8-bit PPG mode as the operation mode of the PPG timer (PPGC: MD01, MD00="00_B").

■ 8-Bit Reload Registers (TMRR0 to TMRR2)

Figure 12.3-22 8-Bit Reload Registers (TMRR0 to TMRR2)



Each of these registers stores a comparison value of the 8-bit timer. The value in the register is reloaded when the 8-bit timer is started. Therefore, if the register value is rewritten during timer operation, the value is validated when the timer is next started. When the waveform generator is used in dead-time timer mode, these registers are used to set the non-overlap time (dead time).

- Use the following formula to calculate the non-overlap time:

Non-overlap time = (TMRR register set value + 1) x operating clock

"00_H" cannot be set to the TMRR register. When using the generator in timer mode, these registers are used to set the GATE time for PPG timer operation.

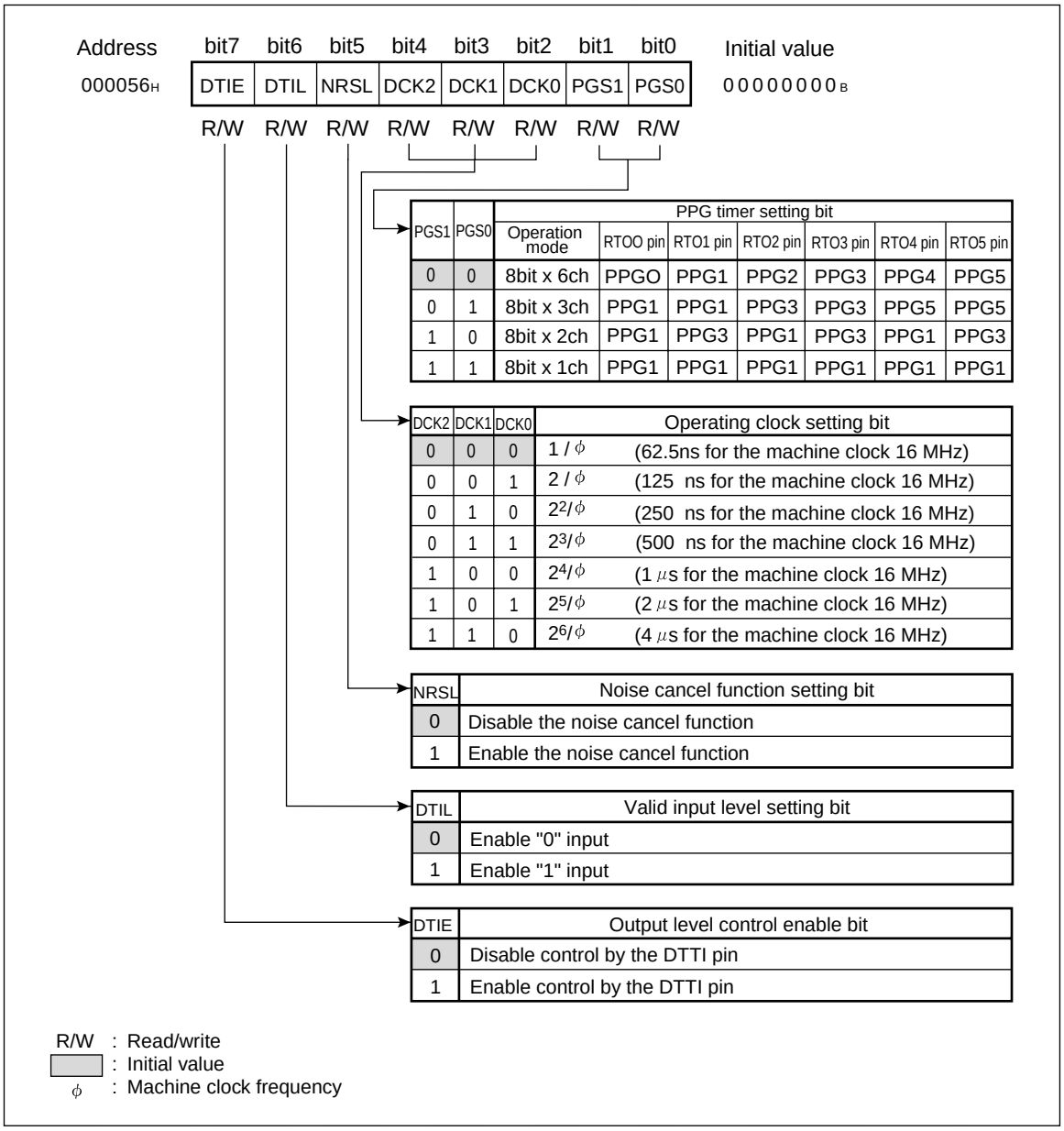
- Use the following formula to calculate the GATE time:

GATE time = (TMRR register set value + 1) x operating clock

"00_H" cannot be set to the TMRR register.

■ Waveform Control Register (SIGCR)

Figure 12.3-23 Waveform Control Register (SIGCR)



Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
000056H	DTIE	DTIL	NRSL	DCK2	DCK1	DCK0	PGS1	PGS0	00000000B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

PGS1	PGS0	PPG timer setting bit						
		Operation mode	RTOO pin	RTO1 pin	RTO2 pin	RTO3 pin	RTO4 pin	RTO5 pin
0	0	8bit x 6ch	PPGO	PPG1	PPG2	PPG3	PPG4	PPG5
0	1	8bit x 3ch	PPG1	PPG1	PPG3	PPG3	PPG5	PPG5
1	0	8bit x 2ch	PPG1	PPG3	PPG1	PPG3	PPG1	PPG3
1	1	8bit x 1ch	PPG1	PPG1	PPG1	PPG1	PPG1	PPG1

DCK2	DCK1	DCK0	Operating clock setting bit
0	0	0	1 / ϕ (62.5ns for the machine clock 16 MHz)
0	0	1	2 / ϕ (125 ns for the machine clock 16 MHz)
0	1	0	$2^2/\phi$ (250 ns for the machine clock 16 MHz)
0	1	1	$2^3/\phi$ (500 ns for the machine clock 16 MHz)
1	0	0	$2^4/\phi$ (1 μs for the machine clock 16 MHz)
1	0	1	$2^5/\phi$ (2 μs for the machine clock 16 MHz)
1	1	0	$2^6/\phi$ (4 μs for the machine clock 16 MHz)

NRSL	Noise cancel function setting bit
0	Disable the noise cancel function
1	Enable the noise cancel function

DTIL	Valid input level setting bit
0	Enable "0" input
1	Enable "1" input

DTIE	Output level control enable bit
0	Disable control by the DTTI pin
1	Enable control by the DTTI pin

R/W : Read/write
□ : Initial value
ϕ : Machine clock frequency

Table 12.3-11 Functions of Bits for Waveform Control Register (SIGCR)

Bit name		Function
bit7	DTIE: Output level control enable bit	This bit enables control of the output level of the RT0 pin through DTTI pin input.
bit6	DTIL: Valid input level setting bit	This bit is used to set the valid input level of the DTTI pin.
bit5	NRSL: Noise cancel function setting bit	This bit is used to set the noise cancel function of the DTTI pin input. The noise cancel function operates the 2-bit internal counter depending on the valid input level. If the valid level is retained until an overflow of the counter value occurs, the DTTI pin input is accepted. The pulse width of noise that can be canceled is about 4 machine cycles. **Note:** If the noise cancel function is enabled, the DTTI pin input is disabled when the oscillation clock is stopped in stop mode.
bit4 bit3 bit2	DCK2, DCK1, DCK0: Operating clock setting bit	These bits are used to set the operating clock of the 8-bit timer.
bit1 bit0	PGS1, PGS0: PPG timer setting bit	These bits are used to set the PPG timer that is used to output PPG timer pulses to the RT0 pin. When the GATE signal output setting bit (GTEN) and pulse enable bit (PGEN) of the 8-bit timer control register (DTCR) are set to "1", the corresponding GATE signal (synchronous mode) for PPG timer operation is output. **Note:** Even if 16-bit PPG mode is set in the PPG timer settings, the corresponding PPG timer can output pulses and can be controlled to start/stop it. PPG timer pulses can be output after software activation (set by the PPG control register (PPGC)) of the PPG timer without using the GATE signal for PPG timer operation (asynchronous mode).

12.4 Operation of Multifunctional Timers

This section describes the operation of the multifunctional timer unit.

■ Operation of Multifunctional Timers

❍ 16-bit free-running timer

The counter value of the 16-bit free-running timer starts counting up from "0000_H" after a reset. The counter output value is used as the reference time (base timer) for output compare and input capture.

❍ 16-bit output compare

The output compare unit compares the compare register (OCCP) value with the counter value of the 16-bit free-running timer. If the unit detects a match, it inverts the output level and then outputs an interrupt.

❍ 16-bit input capture

The input capture unit places the counter value of the 16-bit free-running timer into the input capture data register (IPCP) and then outputs an interrupt when the unit detects a valid edge of input signal from an external input pin (IN0 to IN3).

❍ 8/16-bit PPG timer

The 8/16-bit PPG timer has three operation modes; 8-bit PPG mode, 8-bit prescaler + 8-bit PPG mode, and 16-bit PPG mode.

❍ Waveform generator

The waveform generation block can generate various timing waveforms using the real-time output (RT), 8/16-bit PPG timer, and 8-bit timer.

12.4.1 16-Bit Free-Running Timer

The counter value of the 16-bit free-running timer starts counting up from "0000_H" after a reset. The counter output value is used as the reference time (base timer) for output compare and input capture.

■ Operation of 16-Bit Free-Running Timer

The counter value of the 16-bit free-running timer is cleared to "0000_H" when:

- Reset operation
- Setting of "1" to the clear bit (SCLR) of the timer control status register (TCCS)
- Overflow of the counter value of the 16-bit free-running timer
- Match of the counter value and the compare clear register (CPCLR) value of the 16-bit free-running timer (TCCS: MODE="1")
- Setting of "0000_H" to the timer data register (TCDT)

An interrupt is output when an overflow of the counter value of the 16-bit free-running timer occurs, or when the counter value and the compare clear register (CPCLR) value of the 16-bit free-running timer match (TCCS: ICRE="1", MODE="1") and then the counter value of the 16-bit free-running timer is cleared to "0000_H".

Figure 12.4-1 Counter Cleared by an Overflow

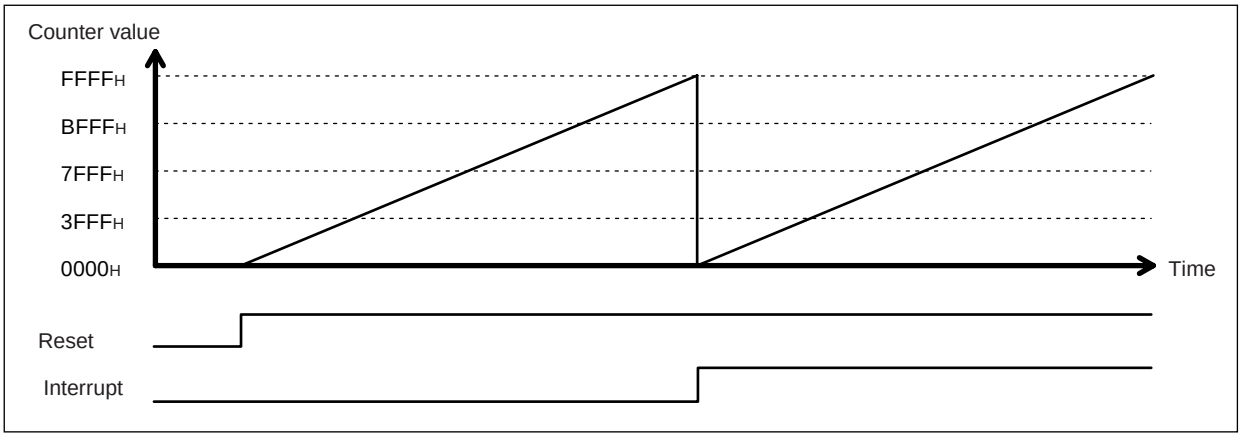
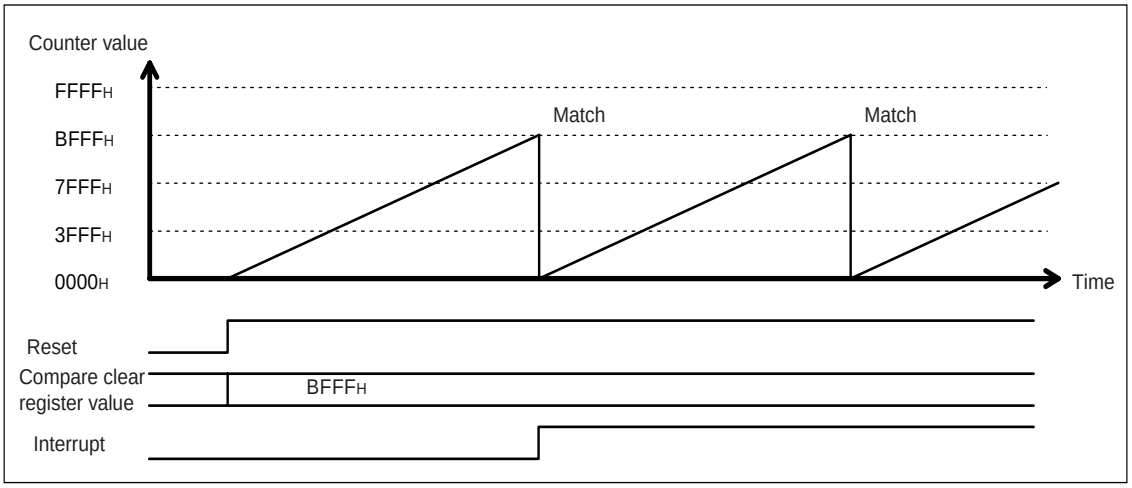


Figure 12.4-2 Interrupt by a Match between the Counter Value and the Compare Clear Register Value

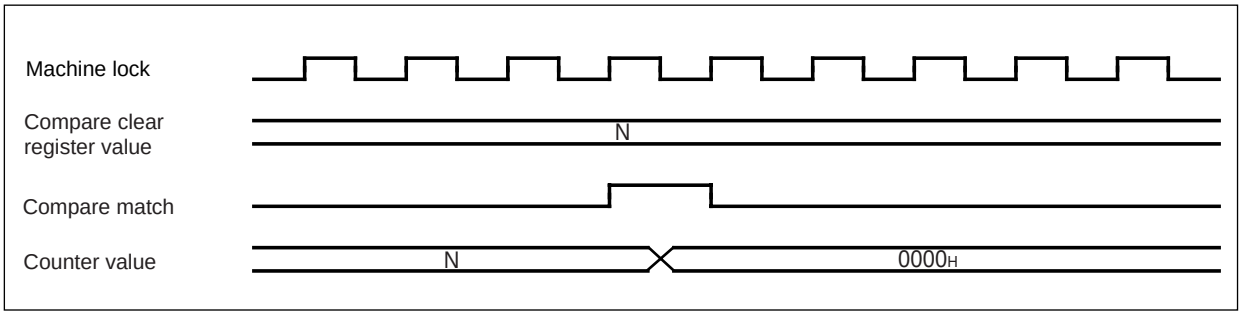


■ Clear Timing of Counter Value of 16-Bit Free-Running Timer

The counter value of the 16-bit free-running timer can be cleared to "0000_H" by a reset operation, setting of "1" to the clear bit (SCLR) of the timer control status register (TCCS), a match of the counter value and the compare clear register (CPCLR) value of the 16-bit free-running timer (TCCS: MODE="1"), or setting of "0000_H" to the timer data register (TCDT).

Clearing of the counter value by a reset operation or setting of the clear bit of the timer control status register occurs asynchronously when it is cleared. However, clearing of the counter value by a match of the counter value and the compare clear register (CPCLR) value of the 16-bit free-running timer occurs in synchronization with the count timing.

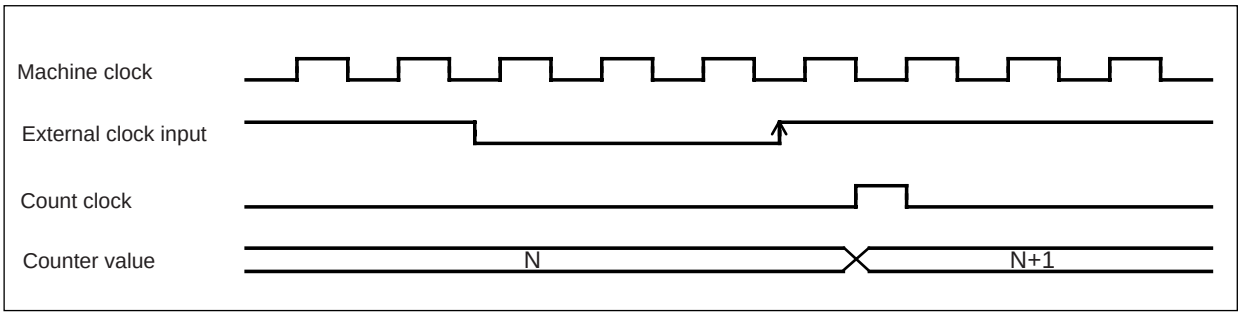
Figure 12.4-3 16-Bit Free-Running Timer Clear Timing



■ Count Timing of Counter Value of 16-Bit Free-Running Timer

The counter value of the 16-bit free-running timer is counted up based on the input machine clock or an external clock. If an external clock is selected, the timer counts up at a rising edge.

Figure 12.4-4 Count Timing of 16-Bit Free-Running Timer



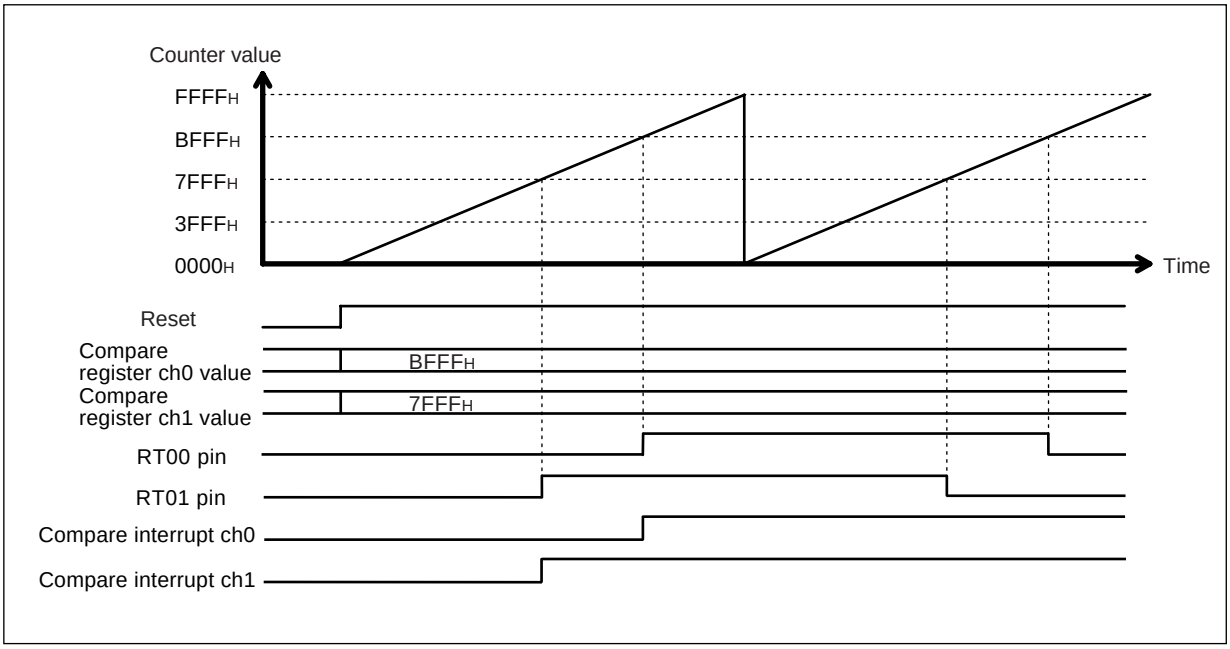
12.4.2 Output Compare

The output compare unit compares a compare register (OCCP0 to OCCP5) value with the counter value of the 16-bit free-running timer. If the unit detects a match, it can invert the output level and output an interrupt.

■ Operation of Output Compare

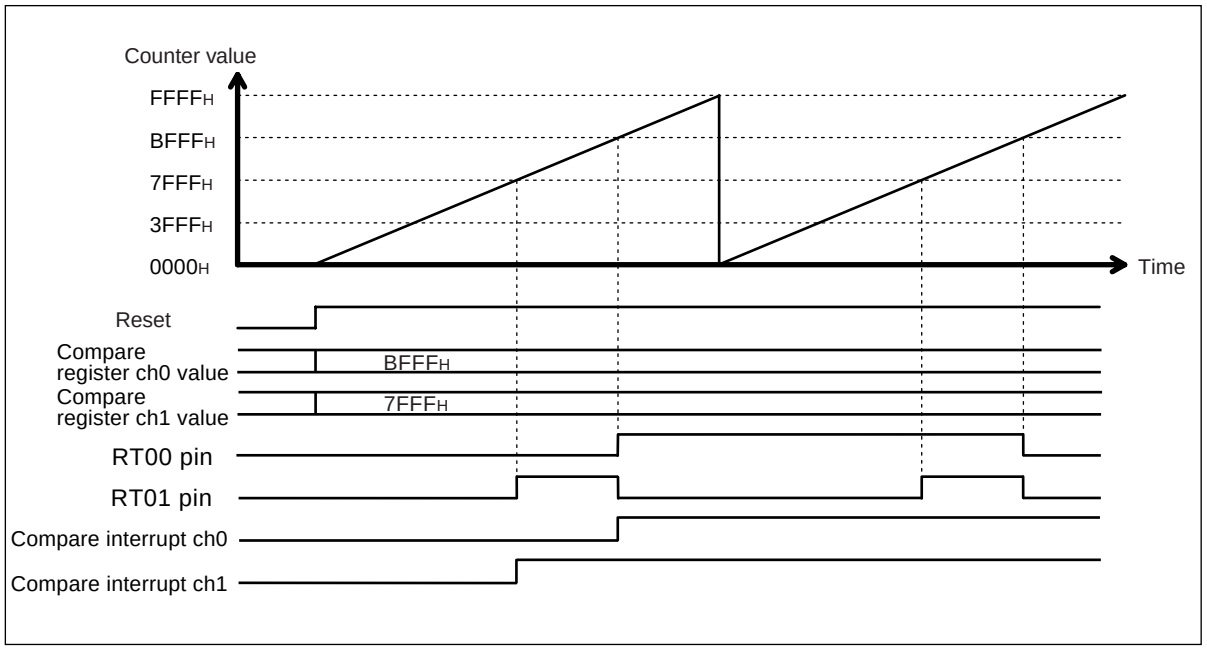
❍ When CMOD of the compare control register (OCS) is "0"

Figure 12.4-5 Example of Output Waveforms when Compare Register (OCCP) Channel 0 and Channel 1 Values Are Used



❍ **When bit12: CMOD of the compare control register (OCS) is "1"**

Figure 12.4-6 Example of Output Waveforms when Compare Register (OCCP) Channel 0 and Channel 1 Values Are Used



■ Output Compare Timing

When the compare register (OCCP) value matches the counter value of the 16-bit free-running timer, the output compare unit generates a compare match signal to invert the output level and output an interrupt. When a compare match occurs, the output level is inverted in synchronization with the count timing of the counter value of the 16-bit free-running timer. No compare operation with the counter value is performed when setting the compare register.

Figure 12.4-7 Compare Operation during Replacement of Compare Registers

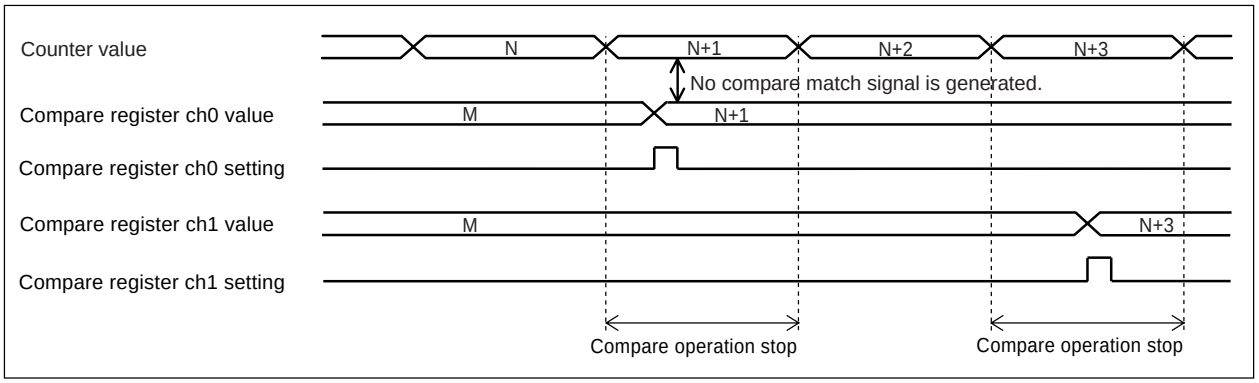


Figure 12.4-8 Compare Interrupt Timing

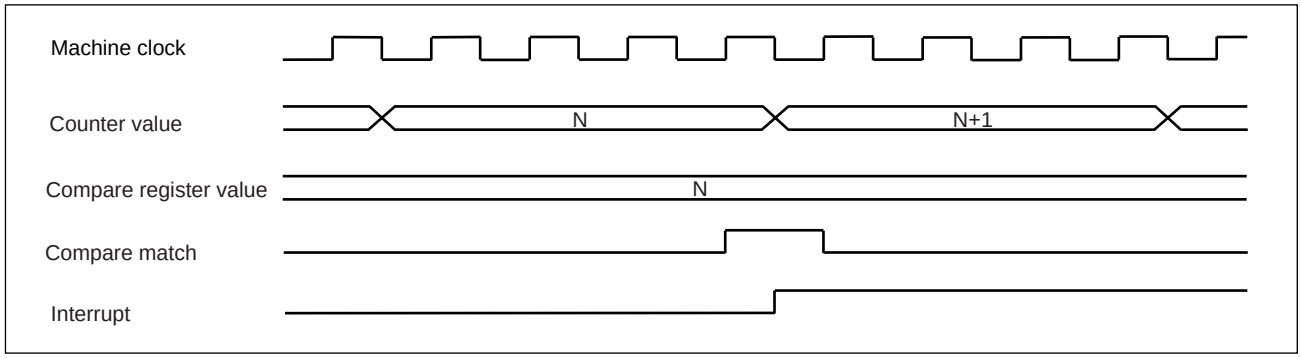
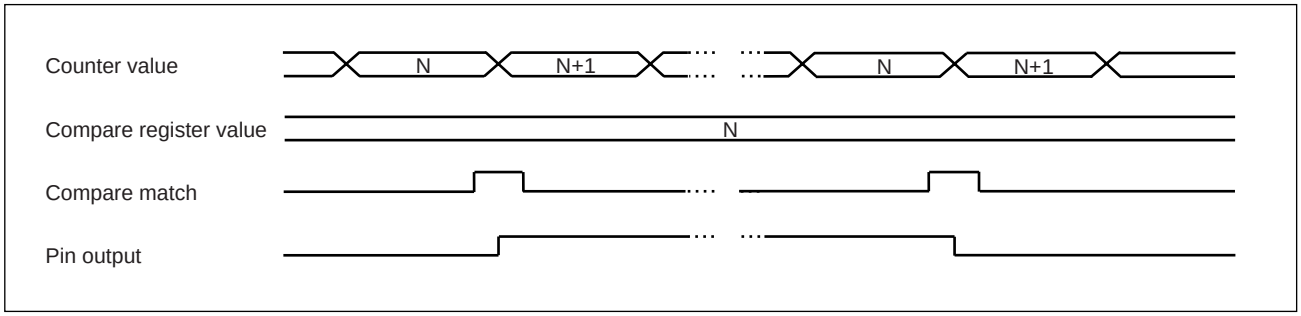


Figure 12.4-9 Pin Output Transition Timing

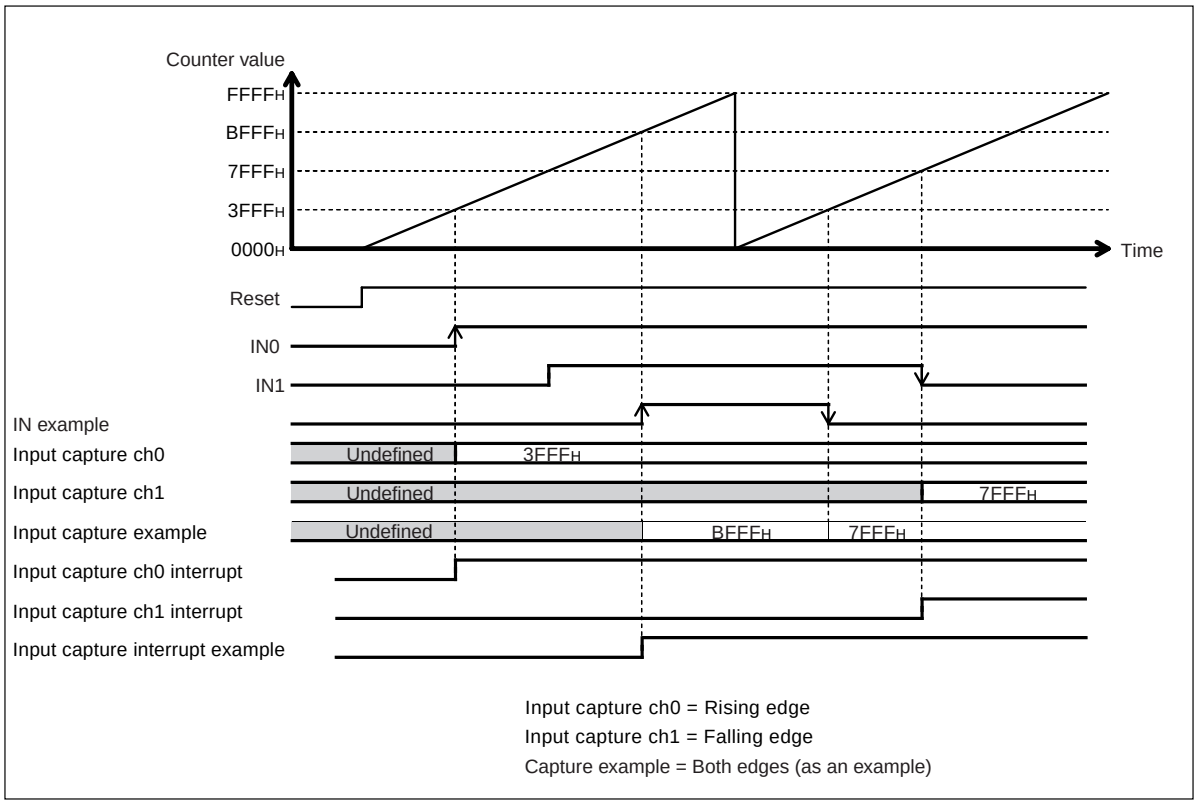


12.4.3 Input Capture

The input capture unit places the counter value of the 16-bit free-running timer into the input capture data register (IPCP) and then outputs an interrupt when the unit detects a valid edge of input signal from an external input pin (IN0 to IN3).

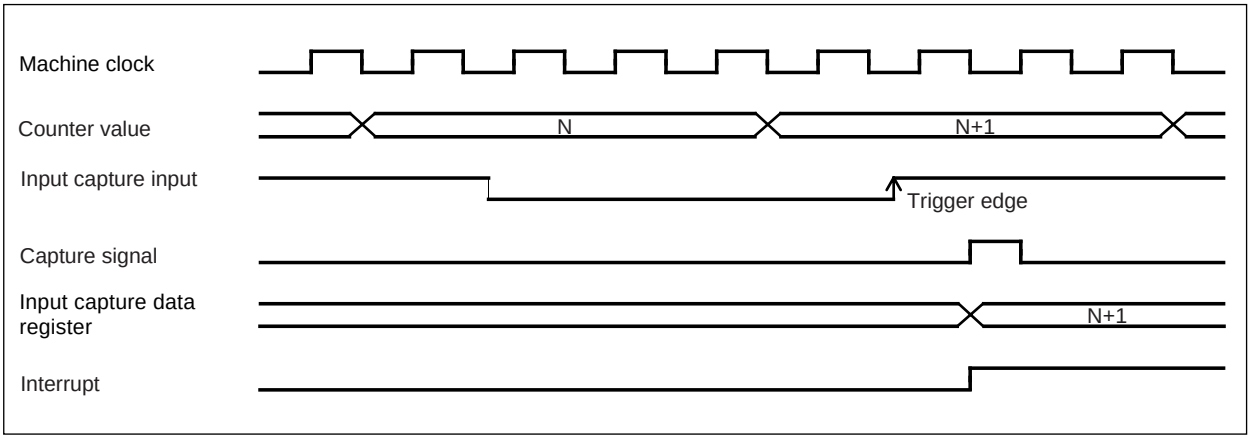
■ Operation of Input Capture

Figure 12.4-10 Example of Input by Input Capture



■ Input Capture Input Timing

Figure 12.4-11 Input Timing for Input Signal



12.4.4 8/16-Bit PPG Timer

The 8/16-bit PPG timer has three operation modes; 8-bit PPG mode, 8-bit prescaler + 8-bit PPG mode, and 16-bit PPG mode.
This section explains channels 0 and 1 of the 8/16-bit PPG timer. For channels 2 and 3 of the 8/16-bit PPG timer, replace channel 0 with channel 2 and channel 1 with channel 3. For channels 4 and 5 of the 8/16-bit PPG timer, replace channel 0 with channel 4 and channel 1 with channel 5.

■ Operation of 8/16-Bit PPG Timer

Each channel of the 8-bit PPG unit has an 8-bit PPG reload register (upper) (PRLH) and an 8-bit PPG reload register (lower) (PRLL). Values written into these registers are alternately reloaded from the upper side/lower side reload register into the 8-bit down-counter (PCNT) and counted down in synchronization with the count clock. The polarity of the pin output (PPG0 to PPG5 pin) is inverted during reloading by a counter underflow. This operation enables the pin output (PPG pin) to become the pulses with the "H"/"L" width associated with the PPG reload register (PRLH, PRLL) value.

Since the output polarity can be set by software, pin output can easily be inverted. Operation is started by setting the PPG control register or entering "H" of the GATE signal.

- When PPGC:SST1 and SST0 are set to "0", setting PPGC1:PEN1 and PPGC0:PEN0 to "1" starts operation.
- When PPGC:SST1 and SST0 are set to "1", operation starts when the GATE signals turn to "H" and continues for the period during which the signal is at the "H" level.

Table 12.4-1 Relationship between Reload Operation and Pulse Output

Reload operation	Pin output transition							
	Positive polarity mode				Negative polarity mode			
PRLH --> PCNT	PPG0/1	[0-->1]		Rise	PPG0/1	[1-->0]		Fall
PRLL --> PCNT	PPG0/1	[1-->0]		Fall	PPG0/1	[0-->1]		Rise

When PPGC:PIE0 and PIE1 are set to "1", an interrupt request is output by an underflow ("00_H" -> "FF_H": 8-bit PPG mode, "0000_H" -> "$FFFF_H$": 16-bit PPG mode) of the counter value of each channel.

■ Operation Mode

❍ 8-bit PPG mode

In this mode, the timer operates as an 8-bit PPG. The PPG0 pin corresponds to the channel 0 PPG output, and the PPG1 pin corresponds to the channel 1 PPG output.

❍ 8-bit prescaler + 8-bit PPG mode

In this mode, channel 0 operates as an 8-bit prescaler, and channel 1 counts by the channel-0 underflow output so that 8-bit PPG waveforms at any period can be output. The PPG0 pin corresponds to the channel-0 prescaler output, and the PPG1 pin corresponds to the channel-1 PPG output.

❍ 16-bit PPG mode

In this mode, channels 0 and 1 are linked for operation as a 16-bit PPG timer. Both PPG0 and PPG1 pins correspond to the 16-bit PPG output.

■ PPG Operation

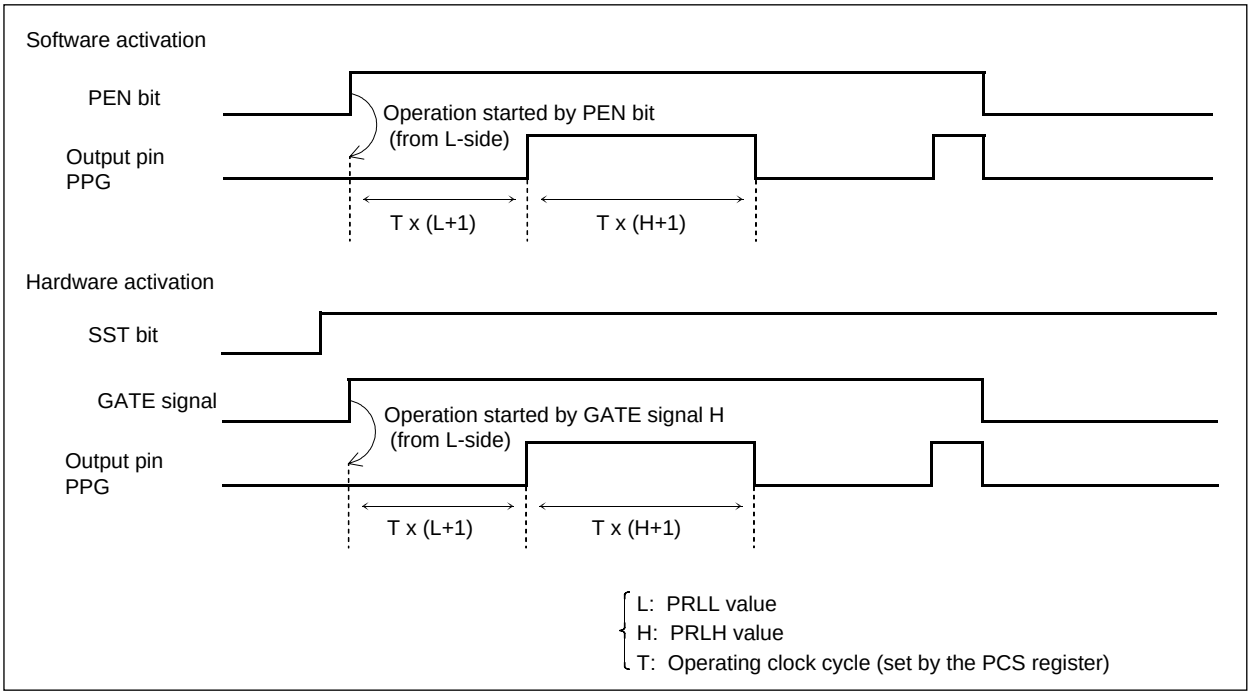
The channel-0 PPG timer starts counting when PPGC:PEN0 is set to "1" while PPGC:SST0 is "0" (software activation). When the GATE signal becomes "H" even if PPGC: SST0 is set to "1", the PPG timer starts counting (hardware activation).

The channel-1 PPG timer starts counting when PPGC:PEN1 is set to "1" while PPGC:SST1 is "0" (software activation). When the GATE signal becomes "H" even if PPGC: SST1 is set to "1", the PPG timer starts counting (hardware activation).

The channel-0 PPG timer stops counting when PPGC:PEN0 is set to "0" and the channel-1 PPG timer stops counting when PPGC:PEN1 is set to "0". Alternatively, the PPG timer stops counting when the GATE signal becomes "L". After the timer is stopped, the pulse output is held at the "L" level (This applies to the positive-polarity output. The pulse output is held at the "H" level for negative-polarity output).

In 8-bit prescaler + 8-bit PPG mode, if activated by software, make settings to not enable operation of channel 1 (PPGC: PEN0="0"), and to disable operation of channel 1 (PPGC: PEN1="1"). In 16-bit PPG mode, set "0" or "1" simultaneously to PPGC: PEN0 and PEN1.

Figure 12.4-12 PPG operating output pulse



■ Relationships between Reload Value and Pulse Width

The value obtained by multiplying the "value set to the PPG reload register (PRLH, PRLL) + 1" by the operating clock cycle becomes the pulse width to be output. When the reload register (PRLH, PRLL) value in 8-bit PPG mode is "00_H" or the reload register (PRLH, PRLL) value in 16-bit PPG operation mode is "0000_H", the pulse width will be set for one cycle of the operating clock. Similarly, when the reload register (PRLH, PRLL) value in 8-bit PPG operation mode is "FF_H", the pulse width will be set for 256 cycles of the operating clock. When the reload register (PRLH, PRLL) value in 16-bit PPG operation mode is "$FFFF_H$", the pulse width will be set for 65,536 cycles of the operating clock.

The pulse with formula is shown below:

PH=Tx(H+1)
PL=Tx(L=1)

PH : H pulse width
PL: L pulse width
H: PRLH value
L: PRLL value
T: Input clock cycle

Note:

The above formula is applicable when the output polarity is positive. When the output polarity is negative, P_L and P_H are reversed.

■ Selection of Count Clock

The operating clock used for the 8-bit down-counter (PCNT) of the PPG timer can be selected from six operating clocks of 1/16 to 1 times the machine clock and the input clock from the time-base timer. Use bit4 to bit2 (PC02 to PC00) of the PPG clock control register (PCS) to select the operating clock for PPG channel 0 and bit7 to bit5 (PC12 to PC10) to select the operating clock for PPG channel 1.

In 8-bit prescaler + 8-bit PPG mode or 16-bit PPG mode, however, PPG channel 1 receives the operating clock from PPF channel 0, and therefore the bit7 to bit5 (PC12 to PC10) settings of the PPG clock control register (PCS) do not affect operation.

Note:

When the time-base timer input is used as the operating clock, the cycle drifts in the following counts:

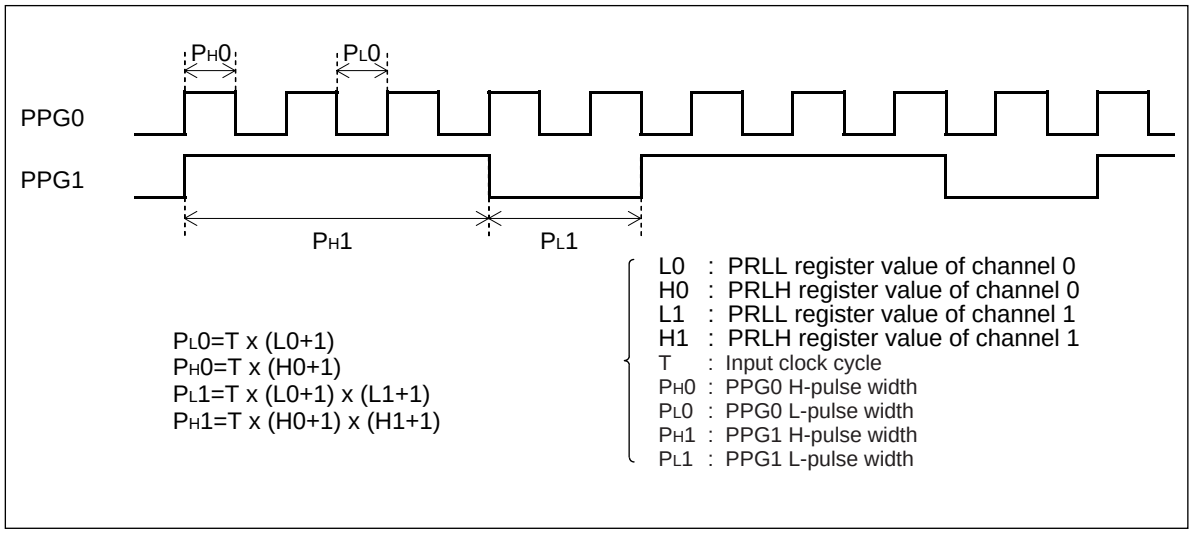
- First count activated by a trigger signal
- First count after canceling stop mode
- First count when PPG channel 1 is activated while PPG channel 0 in 8-bit prescaler + 8-bit PPG mode is operating and PPG channel 1 is stopped
- Count when the time-base timer is cleared (TBTC: TBR="1") during operation

■ Control of Pulse Pin Output

The pulse output generated by the operation of the 8/16-bit PPG timer can be output to external pins (PPG pins). Output to the external pins is enabled by setting the PPG control register (PPGC). When PPGC0: POE0 is set to "1", the PPG channel-0 pulse output is enabled to the PPG0 pin. When PPGC1: POE1 is set to "1", the PPG channel-1 pulse output is enabled to the PPG1 pin. When PPGC0: POE0 and PPGC1: POE1 are set to "0", pulse output to the external pins (PPG0 and PPG1 pins) is not enabled and both the PPG0 and PPG1 pins operate as an I/O port.

In 8-bit prescaler + 8-bit PPG mode, the PPG0 pin outputs the toggle waveforms of 8-bit prescaler and the PPG1 pin outputs the 8-bit PPG pulses. In 16-bit PPG mode, the same waveform is output through both PPG0 and PPG1 pins. Therefore, the same output can be obtained regardless of which external pin is enabled.

Figure 12.4-13 Output Waveform in 8-Bit Prescaler + 8-Bit PPG Mode



Note:

In 8-bit prescaler + 8-bit PPG mode, set the same values to the PRLH register and PRLL register of channel 0, 2 and 4.

■ Interrupt

The 8/16-bit PPG timer outputs an interrupt request when an underflow of the counter value of the PPG timer occurs.

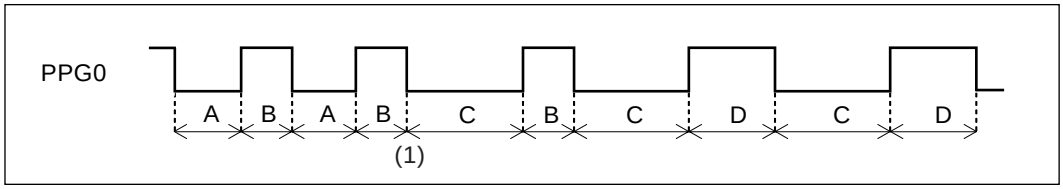
In 8-bit PPG mode and 8-bit prescaler + 8-bit PPG mode, an interrupt request is output when an underflow ("00_H" -> "FF_H") of the 8-bit counter value occurs.

In 16-bit PPG mode, an interrupt request is output when an underflow ("0000_H" -> "$FFFF_H$") of the 16-bit counter value occurs. When an interrupt request is output, "1" is simultaneously set to the PPG timer interrupt request flag bit (PPGC1: PUF1, PPGC0: PUF0). To clear the interrupt request, set "0" simultaneously.

■ Reload Register Write Timing

In 8-bit PPG mode and 8-bit prescaler + 8-bit PPG mode, word transfer instructions should be used to set the PPG reload register (PRLH, PRLL). If two byte transfer instructions are used for settings, the pulse width is delayed depending on the timing.

Figure 12.4-14 Write Timing Chart

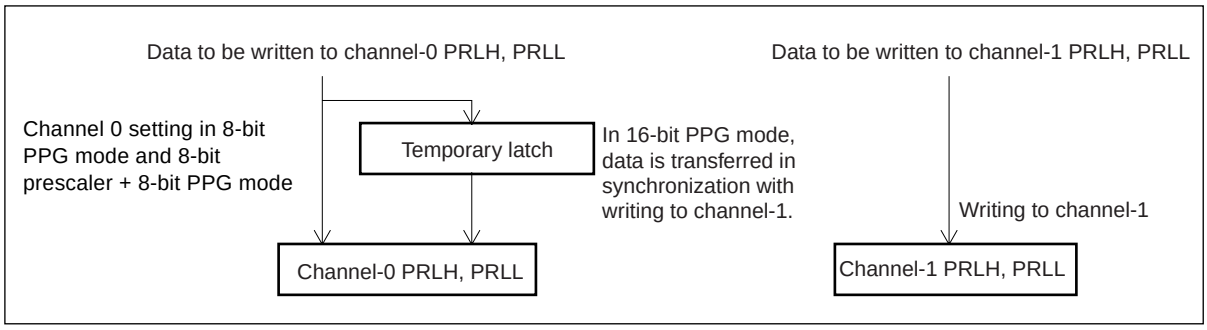


If, in the timing chart shown in Figure 12.4-14 "Write Timing Chart", the PPG reload register (lower) (PRLL) value is set from A to C before the timing of (1) and the PPG reload register (upper) (PRLH) value is set from B to D after the timing of (1), the count-C pulses for the L side and count-B pulses for the H side are generated just after making these settings, rising to the delay of pulse with change. This occurs because the PPG reload register (PRLH, PRLL) values at the timing of (1) are PRLL register value = C and PRLH register value = B.

When using 16-bit PPG mode, set the PPG reload register (PRLH, PRLL) channels 0 and 1 using the long word transfer, or the word transfer in order of channel 0 -> channel 1. In 16-bit PPG mode, the value set to the PPG reload register (PRLH, PRLL) channel 0 is temporarily stored in the temporary latch and then set to the PPG reload register (PRLH, PRLL) channel 0 in synchronization with the setting of the PPG reload register (PRLH, PRLL) channel 1.

In 8-bit PPG mode and 8-bit prescaler + 8-bit PPG mode, as shown in Figure 12.4-15 "Block Diagram for PPG Reload Register (PRLH, PRLL) Setting", channel 0 and channel 1 of the PPG reload register (PRLH, PRLL) can be set independently.

Figure 12.4-15 Block Diagram for PPG Reload Register (PRLH, PRLL) Setting



12.4.5 Waveform Generator

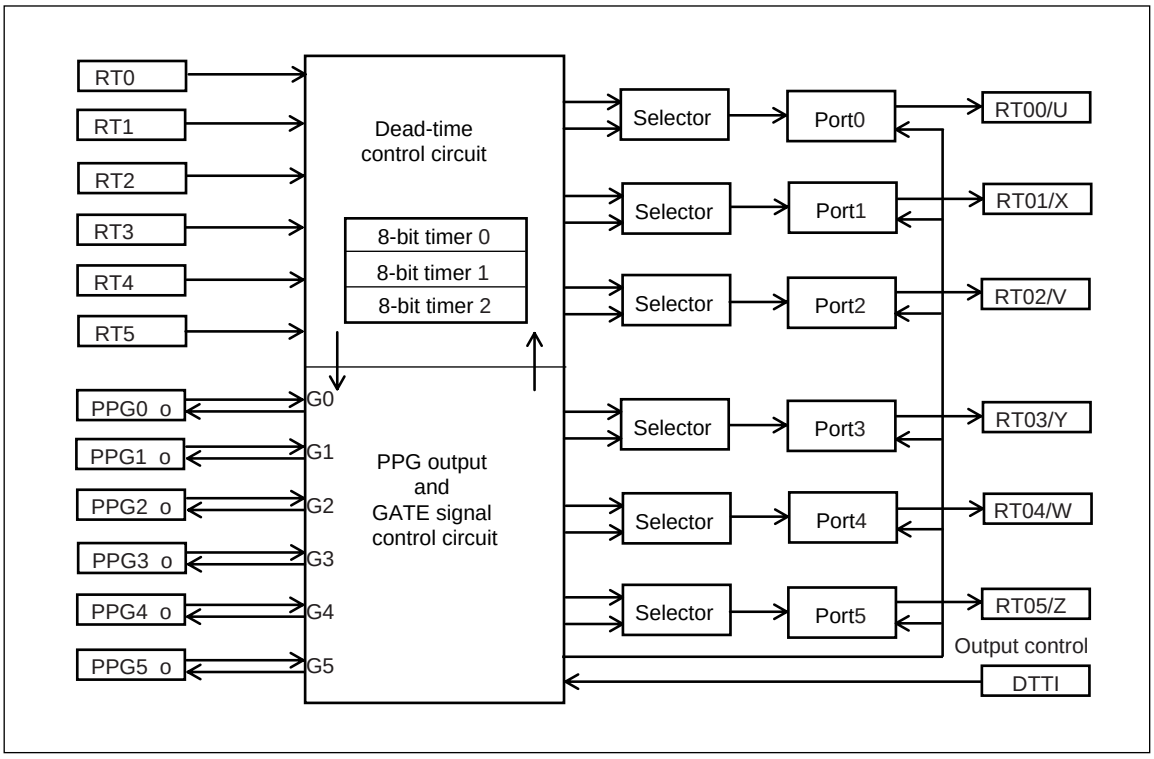
The waveform generator can generate various timing waveforms using real-time outputs (RT) , the 8/16-bit PPG timer, and the 8-bit timer.

■ Waveform Generator

The waveform generator can generates timing waveforms as follows:

- By using the 8-bit timer as a dead-time timer, the waveform generation block can generate non-overlap waveforms by adding the non-overlap time delay to the real-time output (RT1, RT3, RT5) and PPG timer (PPG1, PPG3, PPG5) pulse output (dead-time function).
- An 8-bit timer is started at a rising edge of the real-time output (RT), and the selected (SIGCR: PGS1, PGS0) PPG timer outputs the clock until the counter value of the 8-bit timer and the 8-bit reload register (TMRR) value match (GATE function).

Figure 12.4-16 Waveform Generation Block Operation



12.4.6 Operation of Dead-Time Control Circuit

The dead-time control circuit generates non-overlap waveforms with a delay of the non-overlap time applied to the real-time output (RT1, RT3, RT5) and PPG timer (PPG1, PPG3, PPG5) pulse output.

■ Generating non-overlap signals of RT1/RT3/RT5

Generating Non-Overlap Waveforms of Real-time Output (RT1, RT3, RT5)

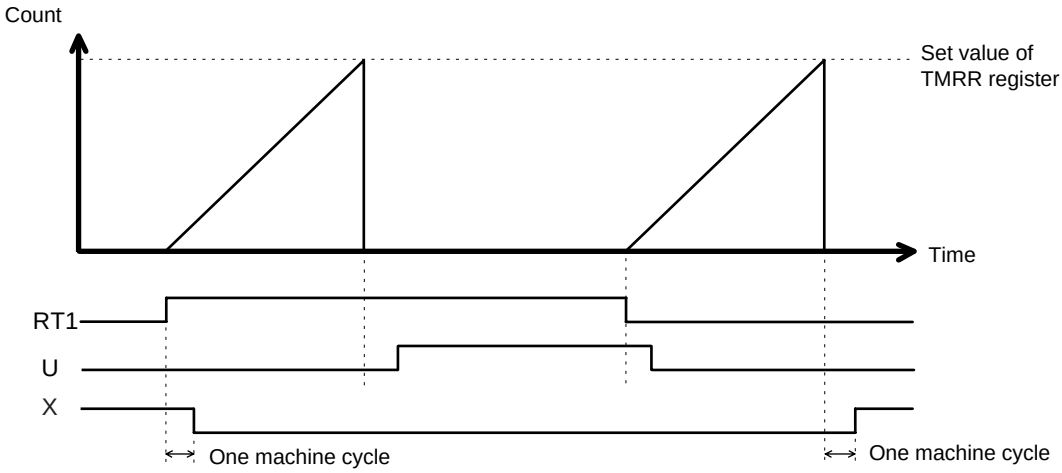
- Non-overlap waveforms with positive-polarity waveform output (DTCR: DMOD="0") are generated by applying a delay of the non-overlap time specified in the 8-bit reload register (TMRR) at a rising edge of the real-time output (RT1, RT3, RT5) and real-time output (RT1, RT3, RT5) inverted pulses. If the pulse width of real-time output (RT1, RT3, RT5) is smaller than the specified non-overlap time, the 8-bit timer restarts counting from "00_H" at the next real-time output edge.

[Register settings]

TCDT: 0000000000000000_B
CPCLR: $XXXXXXXXXXXXXXXX_B$ (Cycle)
TCCS: $X\text{--}XXXXXXX0X0XXX_B$
OCS: $\text{---}1XXXXXXXX\text{--}11_B$
OCCP: $XXXXXXXXXXXXXXXX_B$ (Comparison value)
DTCR: 00000100_B
TMRR: $XXXXXXXX_B$ (Non-overlap time)
SIGCR: $XXXXXXXX_B$ (DTTI input and 8-bit timer count clock)

Set X according to user's operation.
-: Undefined bit

Figure 12.4-17 Generation of Positive-Polarity Non-Overlap Waveforms of Real-time Output



Pin name	Output signal
U	Waveform with a delay applied at a rising edge of RT1 ↑
V	Waveform with a delay applied at a rising edge of RT3 ↑
W	Waveform with a delay applied at a rising edge of RT5 ↑
X	Waveform with a delay applied at a rising edge of RT1 inverted pulses ↑
Y	Waveform with a delay applied at a rising edge of RT3 inverted pulses ↑
Z	Waveform with a delay applied at a rising edge of RT5 inverted pulses ↑

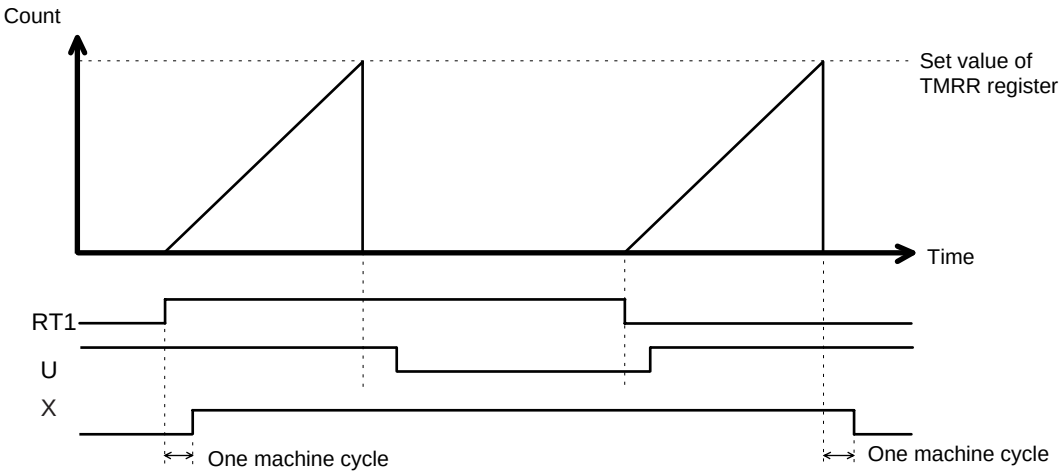
- Non-overlap waveforms with negative-polarity waveform output (DTCR: DMOD="1") are generated by applying a delay of the non-overlap time specified in the 8-bit reload register (TMRR) at a falling edge of the real-time output (RT1, RT3, RT5) inverted pulses and real-time output (RT1, RT3, RT5). If the pulse width of real-time output (RT1, RT3, RT5) is smaller than the specified non-overlap time, the 8-bit timer restarts counting from "00_H" at the next real-time output edge.

[Register settings]

TCDT: 0000000000000000_B
CPCLR: $XXXXXXXXXXXXXXXX_B$ (Cycle)
TCCS: $X\text{--}XXXXXXX0X0XXX_B$
OCS: $\text{----}1XXXXXXXX\text{--}11_B$
OCCP: $XXXXXXXXXXXXXXXX_B$ (Comparison value)
DTCR: 10000100_B
TMRR: $XXXXXXXX_B$ (Non-overlap time)
SIGCR: $XXXXXXXX_B$ (DTTI input and 8-bit timer count clock)

Set X according to user's operation.
-: Undefined bit

Figure 12.4-18 Generation of Negative-Polarity Non-Overlap Waveforms of Real-time Output



Pin name	Output sinal
U	Waveform with a delay applied at a falling edge of RT1 inverted pulses ↓
V	Waveform with a delay applied at a falling edge of RT3 inverted pulses ↓
W	Waveform with a delay applied at a falling edge of RT5 inverted pulses ↓
X	Waveform with a delay applied at a falling edge of RT1 ↓
Y	Waveform with a delay applied at a falling edge of RT3 ↓
Z	Waveform with a delay applied at a falling edge of RT5 ↓

■ Generating Non-Overlap Waveforms of PPG Timer (PPG1, PPG3, PPG5)

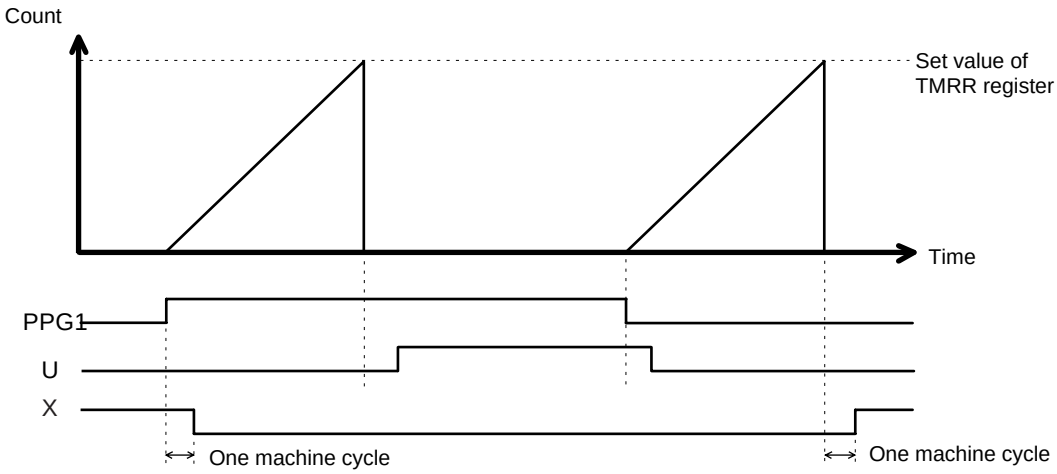
- Non-overlap waveforms with positive-polarity waveform output (DTCR: DMOD="0") are generated by applying a delay of the non-overlap time specified in the 8-bit reload register (TMRR) at a rising edge of the PPG timer (PPG1, PPG3, PPG5) pulses and PPG timer (PPG1, PPG3, PPG5) inverted pulses. If the pulse width of the PPG timer (PPG1, PPG3, PPG5) is smaller than the specified non-overlap time, the 8-bit timer restarts counting from "00_H" at the next PPG timer pulse edge.

[Register settings]

TCDT: 0000000000000000_B
CPCLR: $XXXXXXXXXXXXXXXX_B$ (Cycle setting)
TCCS: $X\text{--}XXXXXXX0X0XXX_B$
OCS: $\text{----}1XXXXXXXX\text{--}11_B$
OCCP: $XXXXXXXXXXXXXXXX_B$ (Compare value setting)
DTCR: 01000111_B
TMRR: $XXXXXXXX_B$ (Non-overlap time setting)
SIGCR: $XXXXXXXX_B$ (DTTI input and 8-bit timer operating clock settings)

Set X according to user's operation.
-: Undefined bit

Figure 12.4-19 Generation of Positive-Polarity Non-Overlap Waveforms of PPG Timer



Pin name	Output signal
U	Waveform with a delay applied at a rising edge of PPG1 ↑
V	Waveform with a delay applied at a rising edge of PPG3 ↑
W	Waveform with a delay applied at a rising edge of PPG5 ↑
X	Waveform with a delay applied at a rising edge of PPG1 inverted pulses ↑
Y	Waveform with a delay applied at a rising edge of PPG3 inverted pulses ↑
Z	Waveform with a delay applied at a rising edge of PPG5 inverted pulses ↑

- Non-overlap waveforms with negative-polarity waveform output (DTCR: DMOD="1") are generated by applying a delay of the non-overlap time specified in the 8-bit reload register (TMRR) at a falling edge of the PPG timer (PPG1, PPG3, PPG5) inverted pulses and PPG timer (PPG1, PPG3, PPG5) pulses. If the pulse width of the PPG timer (PPG1, PPG3, PPG5) is smaller than the specified non-overlap time, the 8-bit timer restarts counting from

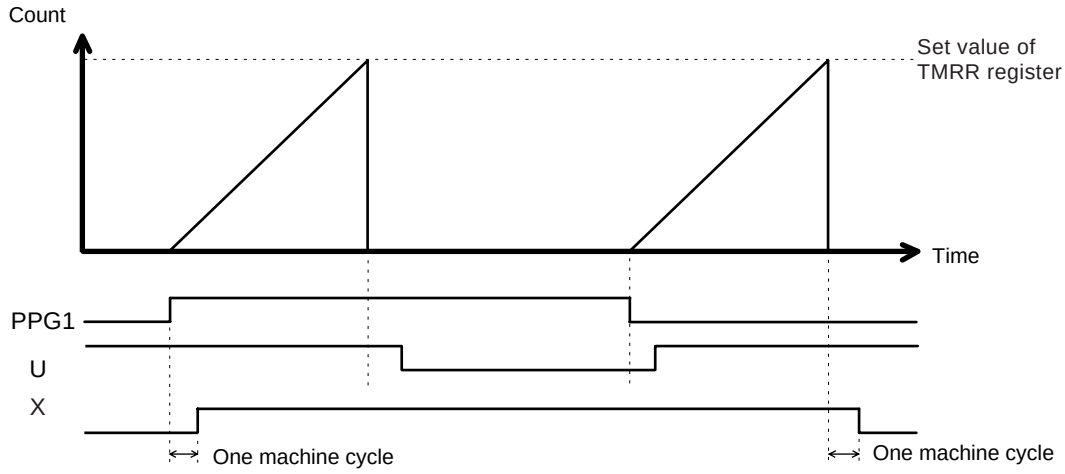
"00_H" at the next PPG timer pulse edge.

[Register settings]

TCDT: 0000000000000000_B
CPCLR: $XXXXXXXXXXXXXXXX_B$ (Cycle setting)
TCCS: $X\text{--}XXXXXXX0X0XXX_B$
OCS: $\text{----}1XXXXXXXX\text{--}11_B$
OCCP: $XXXXXXXXXXXXXXXX_B$ (Compare value setting)
DTCR: 11000111_B
TMRR: $XXXXXXXX_B$ (Non-overlap time setting)
SIGCR: $XXXXXXXX_B$ (DTTI input and 8-bit timer operating clock settings)

Set X according to user's operation.
-: Undefined bit

Figure 12.4-20 Generation of Negative-Polarity Non-Overlap Waveforms of PPG Timer



Pin name	Output signal
U	Waveform with a delay applied at a falling edge of PPG1 inverted pulses ↓
V	Waveform with a delay applied at a falling edge of PPG3 inverted pulses ↓
W	Waveform with a delay applied at a falling edge of PPG5 inverted pulses ↓
X	Waveform with a delay applied at a falling edge of PPG1 ↓
Y	Waveform with a delay applied at a falling edge of PPG3 ↓
Z	Waveform with a delay applied at a falling edge of PPG5 ↓

12.4.7 Operation of PPG Output and GATE Signal Control Circuit

When "1" is set to the GATE signal output setting bit (GTEN) and pulse output enable bit (PGEN) of the 8-bit timer control register (DTCR), the PPG output and GATE signal control circuit output the pulses of the PPG timer set in the PPG timer setting bit (PGS1, PGS0) of the waveform control register (SIGCR) to the RT0 pin. The GATE signal to control the PPG timer is generated by the real-time output (RT) and 8-bit timer.

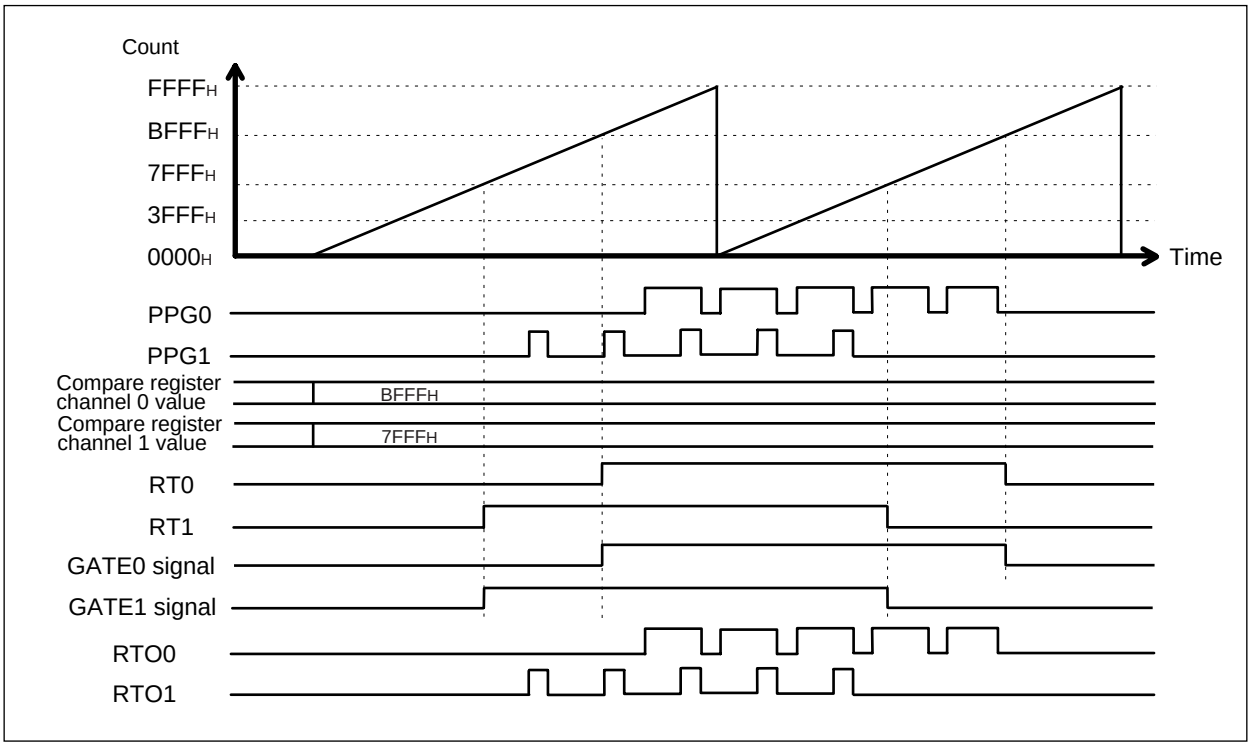
■ **Generating PPG Output and the GATE Signal by the Real-time Output (RT)**

[Register settings]

TCDT: 0000000000000000_B
CPCLR: $XXXXXXXXXXXXXXXX_B$ (Cycle setting)
TCCS: $X\text{--}XXXXXXX0X0XXX_B$
OCS: $\text{----}1XXXXXXXX\text{--}11_B$
OCCP: $XXXXXXXXXXXXXXXX_B$ (Compare value setting)
DTCR: 01100001_B
TMRR: $XXXXXXXX_B$
SIGCR: $XXXXXXXX_B$ (DTTI input and 8-bit timer operating clock settings)

Set X according to user's operation.
-: Undefined bit

Figure 12.4-21 Generation of PPG Output and GATE Signal by Real-time Output (RT)



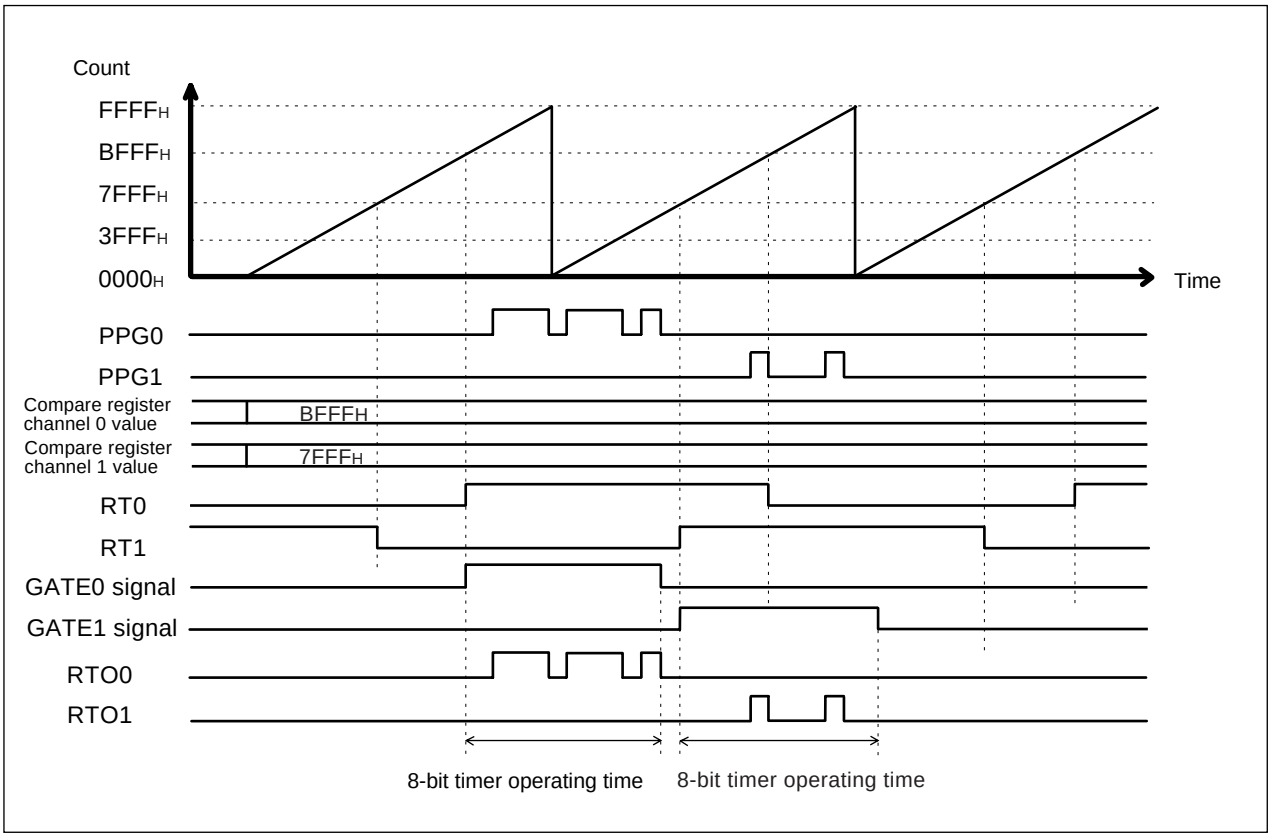
■ Generating PPG Output and the GATE Signal by the 8-Bit Timer

[Register settings]

TCDT: 0000000000000000_B
CPCLR: $XXXXXXXXXXXXXXXX_B$ (Cycle setting)
TCCS: $X\text{--}XXXXXXX0X0XXX_B$
OCS: $\text{----}1XXXXXXXX\text{--}11_B$
OCCP: $XXXXXXXXXXXXXXXX_B$ (Compare value setting)
DTCR: 01100001_B
TMRR: $XXXXXXXX_B$
SIGCR: $XXXXXXXX_B$ (DTTI input and 8-bit timer operating clock settings)

Set X according to user's operation.
-: Undefined bit

Figure 12.4-22 Generation of PPG Output and GATE Signal by 8-Bit Timer



Note:

8-bit timer 0 is used for RT0 and RT1, 8-bit timer 1 is used for RT2 and RT3, and 8-bit timer 2 is used for RT4 and RT5. Be careful not to use an RT and attempt to start the corresponding timer that is already operating. Doing so prolongs the GATE signal and causes a malfunction.

12.4.8 Control of DTTI Pin Input

By setting the output level control enable bit (DTIE) of the waveform control register (SIGCR) to "1", the control of the RT0 pin by DTTI pin input is enabled. If the valid input level set in the valid input level setting bit (DTIL) of the waveform control register (SIGCR) is input to the DTT1 pin, the RT0 pin output is fixed to the level specified in the port 3 data register (PDR3).

■ Control of DTTI Pin Input

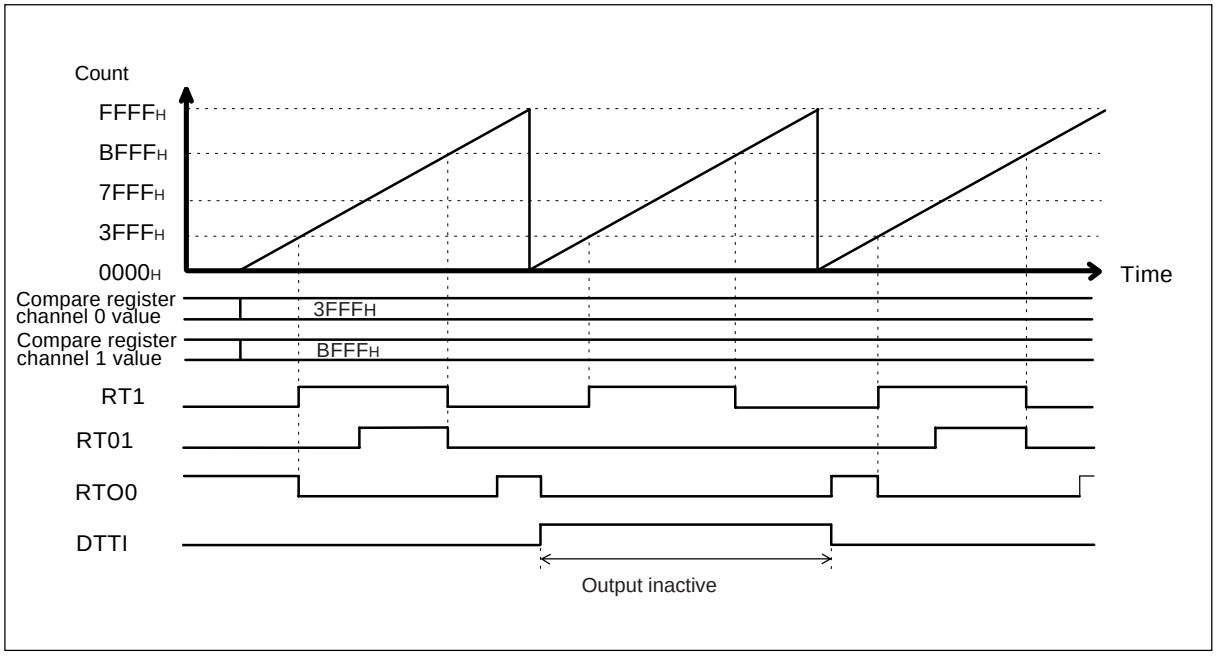
While the output is fixed at the level set in the port 3 data register (PDR3) by DTTI pin input, the 8-bit timer does not stop and continues to generate waveforms, but does not output any waveforms to the RT0 pin.

[Register settings]

TCDT: 0000000000000000_B
CPCLR: $XXXXXXXXXXXXXXXX_B$ (Cycle setting)
TCCS: $X\text{--}XXXXXXX0X0XXX_B$
OCS: $\text{----}0XXXXXXXX\text{--}11_B$
OCCP: $XXXXXXXXXXXXXXXX_B$ (Compare value setting)
DTCR: 0000100_B
TMRR: $XXXXXXXX_B$ (Non-overlap time setting)
SIGCR: $110XXXXX_B$ (DTTI input and 8-bit timer operating clock settings)
PDR3: $00XXXX00_B$ (Set the inactive level for each terminal.)

Set X according to user's operation.
-: Undefined bit

Figure 12.4-23 Operation Performed when DTTI Pin Input is Enabled



■ Noise Cancel Function for DTTI Pin

When bit 5 (NRSL) of the waveform control register is set to 1, the noise cancel function for DTTI pin input can be used. When the noise cancel function is selected, the time for fixing an output pin at the inactive level is delayed for about four machine clocks by the noise cancel circuit. Since the noise cancel circuit uses a peripheral clock, input is invalidated even if the DTTI input is enabled in a mode such as STOP mode in which the oscillator stops.

CHAPTER 13 UART

This chapter explains the functions and operation of UART.

13.1 Overview of UART

UART is a general-purpose serial data communication interface for performing synchronous or asynchronous (start-stop synchronization) communication with external devices. The UART has a bidirectional communication function (normal mode), additionally the master-slave communication function (multiprocessor mode) is only available for the master system.

■ UART Functions

❍ UART Functions

UART is a general-purpose serial data communication interface for transmitting serial data to and receiving data from another CPU and peripheral devices. It has the functions listed in Table 13.1-1 "UART Functions".

Table 13.1-1 UART Functions

	Function
Data buffer	Full-duplex, double buffering
Transfer mode	• Clock synchronous (using start and stop bits) • Clock asynchronous (start-stop synchronization)
Baud rate	• Up to 2MHz (when the machine clock is operated at 16MHz) • A dedicated baud rate generator is provided. • Baud rate by an external clock (clock input through the SCK0/SCK1 pins) • Internal clock (internal clocks supplied from 16-bit reload timer 0 can be used.) • The baud rate can be selected from a total of eight types
Data length	• 7 bits (in asynchronous normal mode only) • 8 bits
Signal mode	Non-return to zero (NRZ)
Reception error detection	• Framing error • Overrun error • Parity error (cannot be detected in multiprocessor mode.)
Interrupt request	• Reception interrupt (reception completion and reception error detection) • Transmission interrupt (transmission completion) • Extended intelligent I/O service (EI^20S) is available for both transmission and reception interrupts.
Master-slave communication function (multiprocessor mode	One-to-n communication (one master to n slaves) can be performed. (This function is supported only for the master system.)

Note:

During clock synchronous transfer, start and stop bits are not added so only data is transferred in UART.

Table 13.1-2 UART Operation Mode

Operation mode		Data length		Synchronization on mode	Stop bit length
		When parity is disabled	When parity is enabled		
0	Normal mode	7 or 8 bits		Asynchronous	1 or 2 bits *2
1	Multiprocessor	8+1*1	-	Asynchronous	
2	Normal mode	8	-	Synchronous	None

-: Setting not possible
*1: "+1" indicates the address/data selection bit (A/D) for communication control.
*2: During reception, only one stop bit can be detected.

■ UART Interrupt and EI^{2}OS

Table 13.1-3 UART Interrupt and EI^{2}OS

Interrupt cause	Interrupt number	Interrupt control register		Vector table address			EI^{2}OS
		Register name	Address	Lower	Upper	Bank	
UART1 reception interrupt	#37(25_H)	ICR13	$0000BD_H$	$FFFF68_H$	$FFFF69_H$	$FFFF6A_H$	O
UART1 transmission interrupt	#38(26_H)	ICR13	$0000BD_H$	$FFFF64_H$	$FFFF65_H$	$FFFF66_H$	X
UART0 reception interrupt	#39(27_H)	ICR14	$0000BE_H$	$FFFF60_H$	$FFFF61_H$	$FFFF62_H$	O
UART0 transmission interrupt	#40(28_H)	ICR14	$0000BE_H$	$FFFF5C_H$	$FFFF5D_H$	$FFFF5E_H$	X

O: Provided with a function that detects a UART reception error and stops EI^{2}OS
X: Usable when ICR13 and ICR14 or interrupt causes that share an interrupt vector are not used

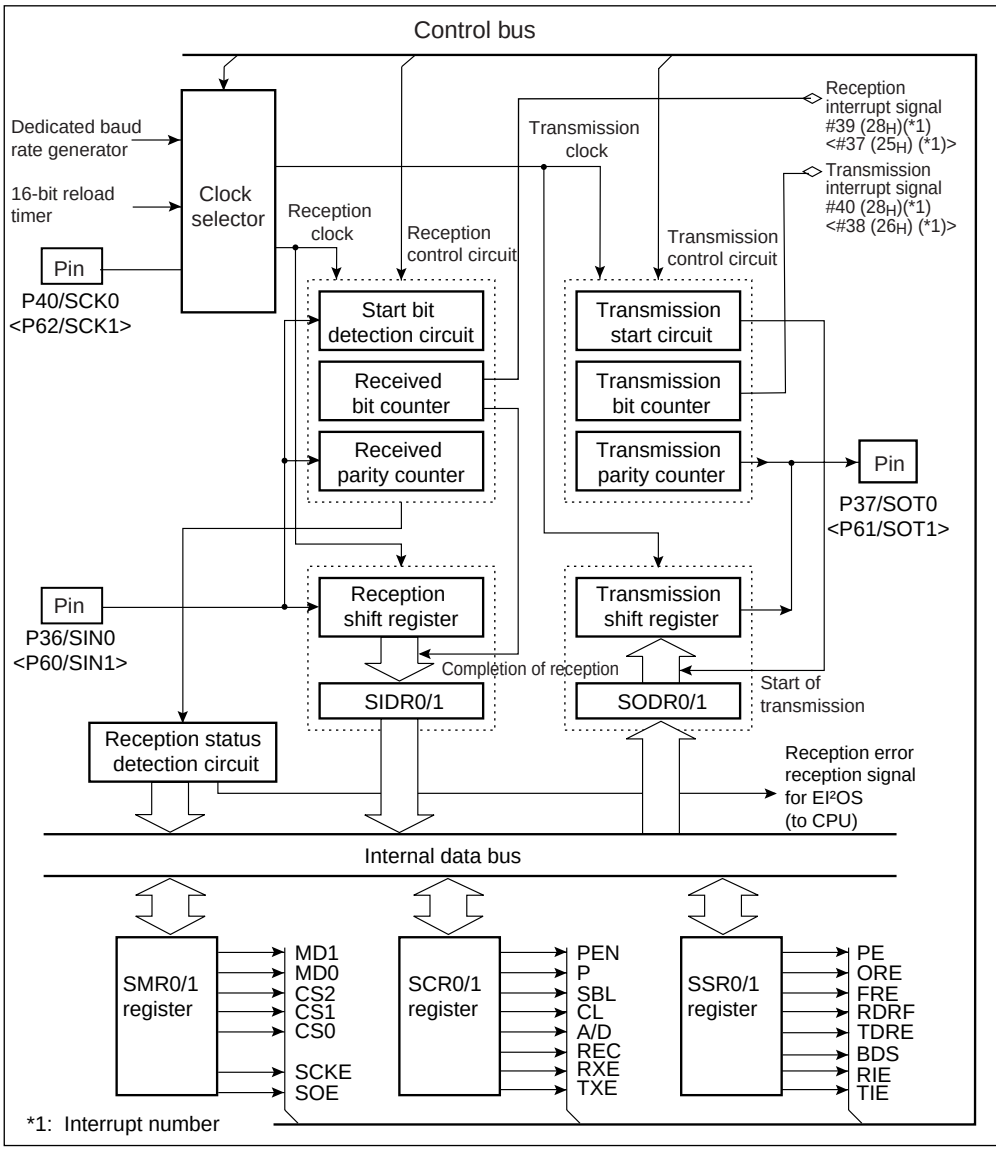
13.2 Configuration of UART

UART consists of the following 11 blocks:

- **Clock Selector**
- **Reception Control Circuit**
- **Transmission Control Circuit**
- **Reception Status Detection Circuit**
- **Reception Shift Register**
- **Transmission Shift Register**
- **Mode Control Register (SMR0/1)**
- **Control Register (SCR0/1)**
- **Status Register (SSR0/1)**
- **Input Data Register (SIDR0/1)**
- **Output Data Register (SODR0/1)**

■ Block Diagram of UART

Figure 13.2-1 Block Diagram of UART



❍ Clock Selector

The clock selector selects the sending/receiving clock from the dedicated baud rate generator, external input clock (clock input through the SCK0/SCK1 pins), and internal clock (clock supplied from the 16-bit reload timer).

❍ Reception Control Circuit

The reception control circuit consists of a received bit counter, start bit detection circuit, and received parity counter.The received bit counter counts receive data bits. When reception of one data item for the specified data length is complete, the received bit counter generates a reception interrupt request. The start bit detection circuit detects start bits from the serial input signal. When the circuit detects a start bit, it writes data in the SIDR0/1 register by shifting at the specified transfer rate. The received parity counter calculates the parity of the receive data.

❍ **Transmission Control Circuit**

The transmission control circuit consists of a transmission bit counter, transmission start circuit, and transmission parity counter.The transmission bit counter counts transmission data bits. When transmission of one data item of the specified data length is complete, the transmission bit counter generates a transmission interrupt request. The transmission start circuit starts transmission when send data is written to the output data register (SODR0/SODR1). The transmission parity counter generates the parity bits for data when transmitting data with parity.

❍ **Reception Shift Register**

The reception shift register fetches receive data input from the SIN0/1 pin, shifting the data bit by bit. When reception is complete, the reception shift register transfers receive data to the SIDR0/1 register.

❍ **Transmission Shift Register**

The transmission shift register transfers data written to the SODR0/1 register to itself and outputs the data to the SOT0/1 pin, shifting the data bit by bit.

❍ **Mode Control Register (SMR0/1)**

The mode register performs the operations of the operation mode setting, baud rate clock setting, serial clock I/O control, and output enable setting of serial data to pins.

❍ **Control Register (SCR0/1)**

The control register performs the operations of the parity presence/absence setting, parity setting, stop bit length/data length settings, frame data format setting in operation mode 1, clearing of received error flag bits, and enable/disable setting of send/receive operations.

❍ **Status Register (SSR0/1)**

The status register performs the operations of status check for transmission/reception and errors, transfer direction setting of serial data, and enable/disable setting of send/receive interrupt requests.

❍ **Input Data Register (SIDR0/1)**

This register retains receive data.

❍ **Output Data Register (SODR0/1)**

This register sets transmission data. Data written to this register is converted to serial data and output.

13.3 UART Pins

This section describes the UART pins and provides a pin block diagram.

■ UART Pins

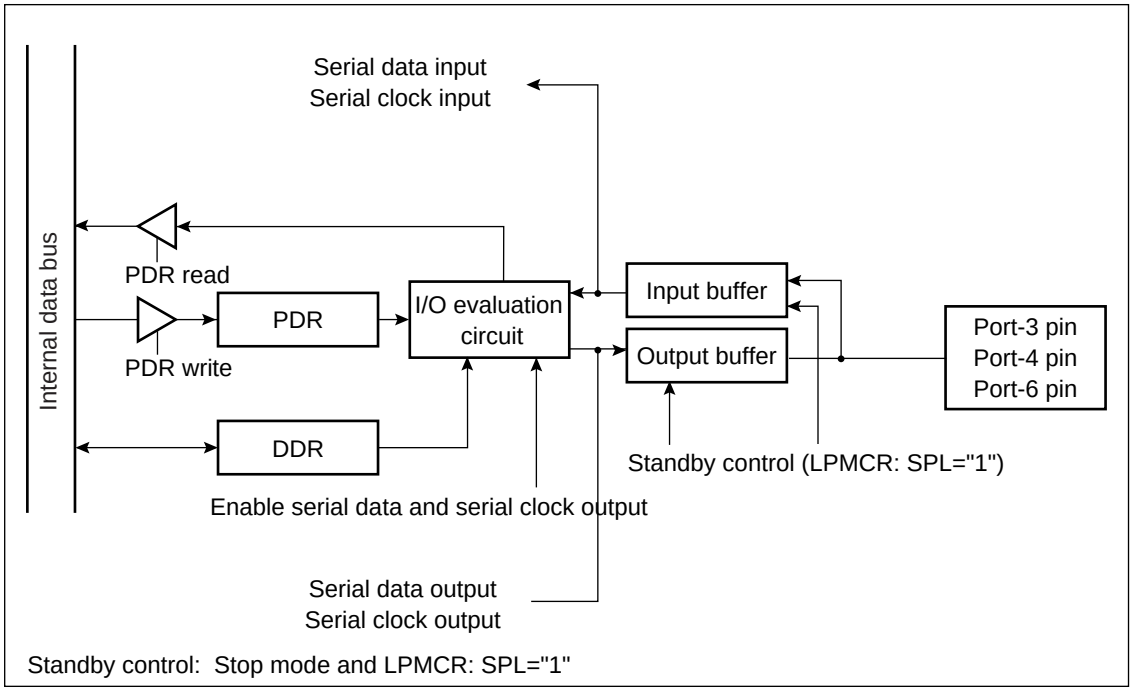
The UART pins also serve as I/O ports.

Table 13.3-1 UART Pins

Pin name	Pin function	I/O format	Pull-up	Standby control	Setting required to use pin
P36/SIN0	Port I/O or serial data input	CMOS output and CMOS hysteresis input	Nothing	Provided	Set as an input port (DDR3: bit0 = 0)
P37/SOT0	Port I/O or serial data output				Set to serial data output enable mode (SMR0: SOE = 1)
P40/SCK0	Port I/O or serial clock input/output				Set as an input port (DDR4: bit 0=0)
					Set to serial clock output enable mode (SMR0:SCKE = 1)
P60/SIN1	Port I/O or serial data input				Set as an input port (DDR6: bit0 = 0)
P61/SOT1	Port I/O or serial data output				Set to serial data output enable mode (SMR1: SOE = 1)
P62/SCK1	Port I/O or serial clock input/output				Set as an input port (DDR6: bit2 = 0)
					Set to serial clock output enable mode (SMR1:SCKE = 1)

■ Block Diagram of UART Pins

Figure 13.3-1 Block Diagram of UART Pins



13.4 UART Registers

The following figure shows the UART registers.

■ UART Registers

Figure 13.4-1 UART Registers

Address	bit15 ····· bit8	bit7 ····· bit0
ch0:000021 H ,20 H ch1:000025 H ,24 H	SCR (control register)	SMR (mode control register)
ch0:000023 H, 22 H ch1:000027 H ,26 H	SSR (status register)	SIDR/SODR (input/output data register)
ch0:000029 H ch1:00002B H	CDCR (communication prescaler control register)	Reserved

13.4.1 Control Register (SCR0/SCR1)

The control register (SCR0/SCR1) is a register for performing the operations of the parity presence/absence setting, parity setting, stop bit length/data length settings, frame data format setting in operation mode 1, clearing of received error flag bits, and enable/disable setting of send/receive operations.

■ Control Register (SCR0/SCR1)

Figure 13.4-2 Control Register (SCR0/SCR1)

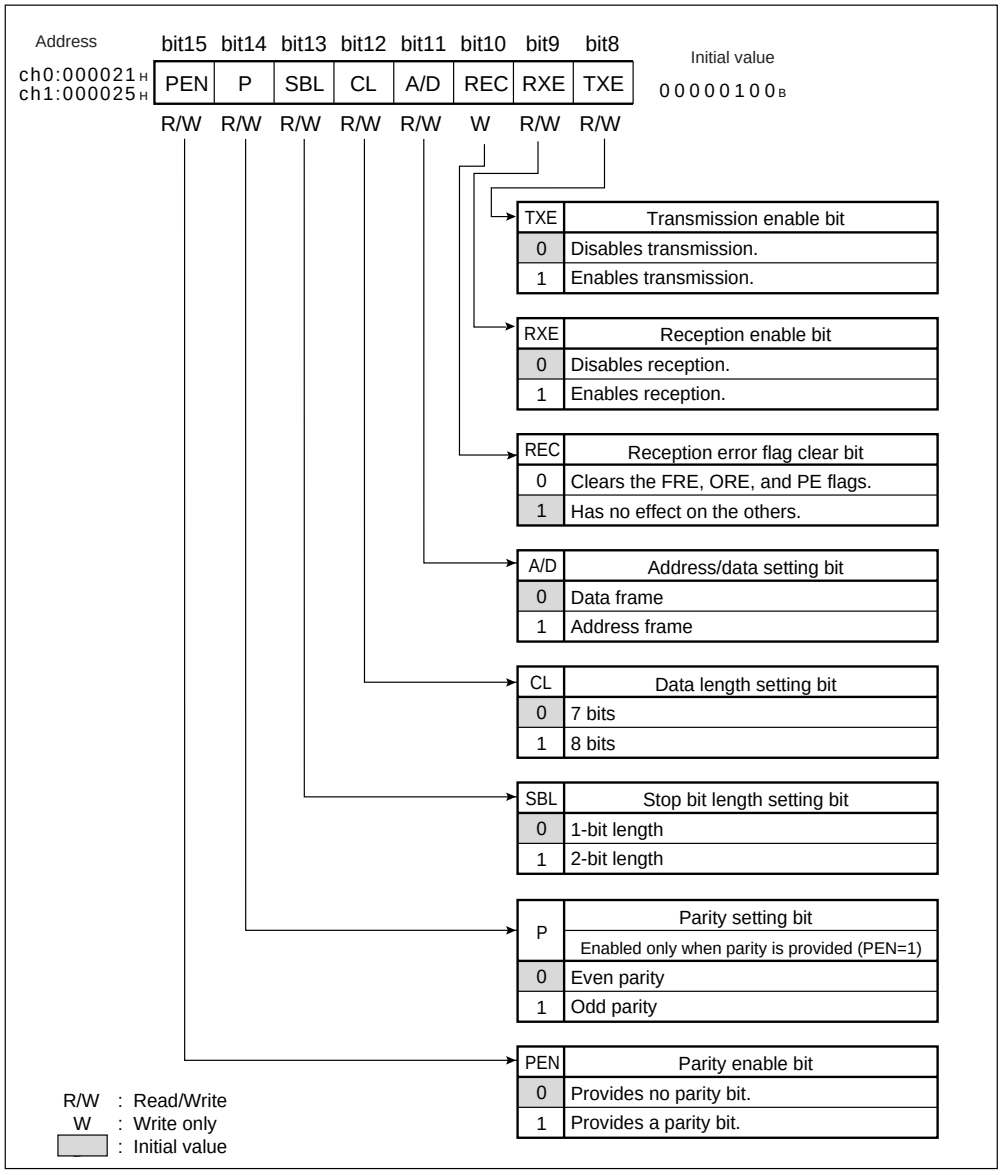


Table 13.4-1 Functions of Bits for Control Register (SCR0/SCR1)

Bit name		Function
bit15	PEN: Parity enable bit	This bit selects whether to add a parity bit during transmission in serial data input-output mode or to detect it during reception. **Note:** No parity can be used in operation modes 1 and 2. Therefore, fix this bit to 0.
bit14	P: Parity selection bit	This bit specifies the odd parity/even parity. **Note:** Valid only when parity presence (PEN="1") is selected.
bit13	SBL: Stop bit length selection bit	This bit selects the length of the stop bits or the frame end mark of send data in asynchronous transfer mode. **Note:** During reception, only the first bit of the stop bits is detected.
bit12	CL: Data length selection bit	This bit specifies the length of send and receive data. **Note:** Seven bits can be selected in operation mode 0 (asynchronous) only. Be sure to select eight bits (CL=1) in operation mode 1 (multiprocessor mode) and operation mode 2 (synchronous).
bit11	A/D: Address/ data selection bit	• Specify the data format of a frame to be sent or received in multiprocessor mode (mode 1). • Select usual data when this bit is 0, and select address data when the bit is 1.
bit10	REC: Reception error flag clear bit	• This bit clears the FRE, ORE, and PE flags of the status register (SSR0/1). • Write 0 to this bit to clear the FRE, ORE, and PE flag. Writing 1 to this bit has no effect on the others. **Note:** If UART is active and a reception interrupt is enabled, clear the REC bit only when the FRE, DRE, or PE flag indicates 1.
bit9	RXE: Reception enable bit	• This bit controls UART reception. • When this bit is 0, reception is disabled. When it is 1, reception is enabled. **Note:** If reception operation is disabled during reception, reception of data currently being received is finished and the received data is stored in the input data register (SIDR0/SIDR1), and then the reception operation is stopped.

Table 13.4-1 Functions of Bits for Control Register (SCR0/SCR1) (Continued)

Bit name		Function
bit8	TXE: Transmission enable bit	• This bit controls UART transmission. • When this bit is 0, transmission is disabled. When the bit is 1, transmission is enabled. **Note:** When transmission operation is disabled during transmission, wait until there is no data in the send data buffers (SODR0/1) before stopping the transmission operation. If transmission operation is disabled during transmission, transmission operation is stopped after all data in the output data register (SODR0/SODR1) is sent out. When setting "0", write data to the output data register (SODR0/SODR1) and then wait for at least 1/16 period of the baud rate for the clock asynchronous transfer mode and the same period as the baud rate for the clock synchronous transfer mode.

13.4.2 Mode Register (SMR0/SMR1)

The mode register (SMR0/SMR1) is a register for performing the operations of the operation mode setting, baud rate clock setting, serial clock I/O control, and output enable setting of serial data to pins.

■ Mode Control Register (SMR0/SMR1)

Figure 13.4-3 Mode Control Register (SMR0/SMR1)

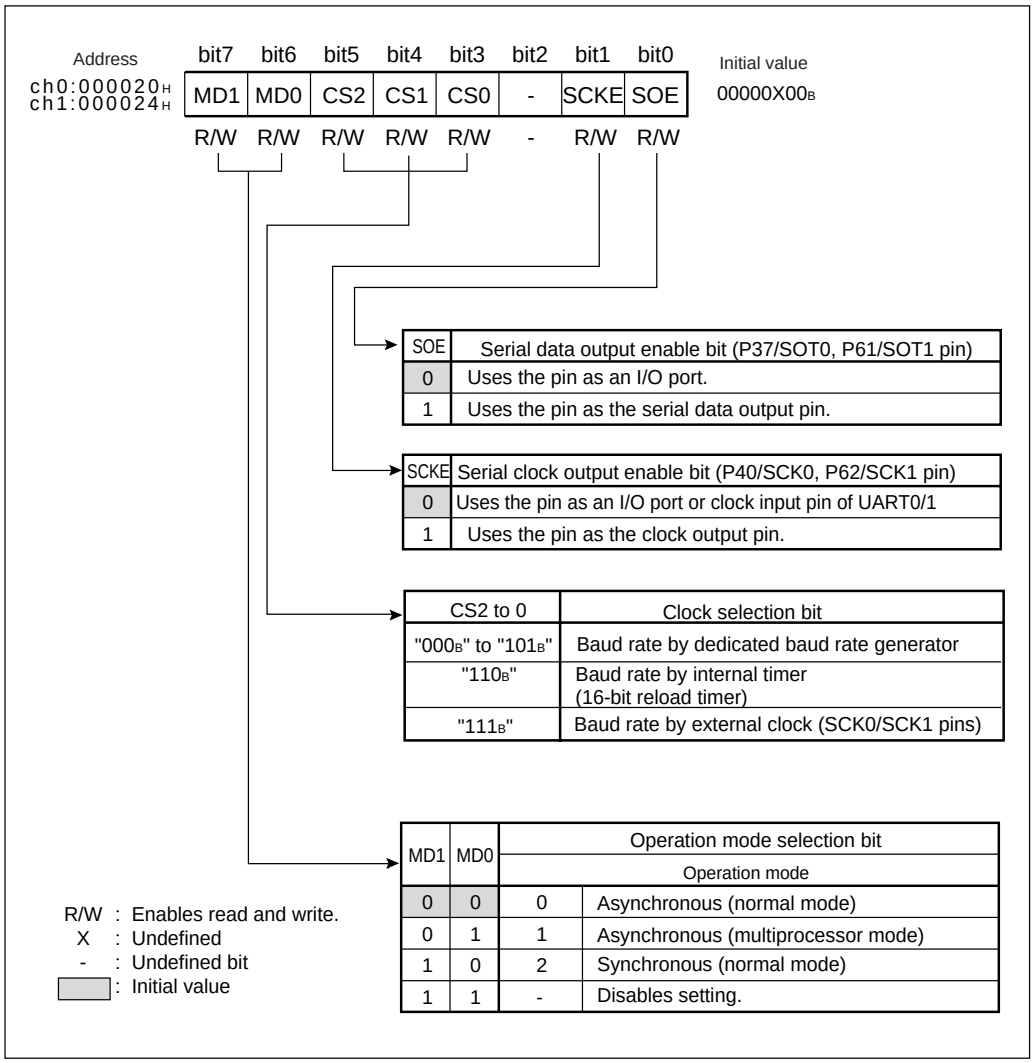


Table 13.4-2 Functions of Bits for Mode Register (SMR0/SMR1)

Bit name		Function
bit7 bit6	MD1 and MD0: Operation mode selection bits	These bits select an operation mode. **Note:** Operation mode 1 (multiprocessor mode) can be used only from the master system during master-slave communication. UART cannot be used from the slave system because it has no address/data detection function during reception.
bit5 bit4 bit3	CS2 to CS0: Clock selection bits	• This bit selects a baud rate clock source. When the dedicated baud rate generator is selected, the baud rate is determined at the same time. • When the dedicated baud rate generator is selected, eight baud rates can be selected: 6 types for the dedicated baud rate generator selected, 1 type for the internal timer selected, and 1 type for the external clock selected. • Clock input can be selected from external clocks (SCK0/SCK1 pin input), the internal clock (16-bit reload timer), and the dedicated baud rate generator.
bit2	-: Undefined bit	• When this bit is read, the value is undefined. • The value set to this bit does not affect operation.
bit1	SCKE: Serial clock output enable bit	• This bit controls the serial clock input-output ports. • When this bit is 0, the P40/SCK0 and P62/SCK1 pins operate as input-output ports or serial clock input pins. When this bit is 1, the pins operate as serial clock output pins. **Note:** - To use the P40/SCK0 and P62/SCK1 pins as serial clock input (SCKE="0") pins, set them as input pins. Also, select an external clock (SMR0/SMR1: CS2 to CS0="111_B") using the clock setting bits. - To use the P40/SCO0 and P62/SCO1 pins as serial clock output (SCKE="1") pins, select the dedicated baud rate generator (SMR0/SMR1: CS2 to CS0="000_B" to "101_B") or internal clock (SMR0/SMR1: CS2 to CS0="110_B"). **Reference:** When the SCK0/1 pin is assigned to serial clock output (SCKE=1), it functions as the serial clock output pin regardless of the status of the input-output ports.
bit0	SOE: Serial data output enable bit	• This bit enables the output of serial data. • When this bit "0", the P37/SOT0, P61/SOT1 pins operate as an I/O port. • When this bit "1", the P37/SOT0, P61/SOT1 pins operate as a serial data output pin. **Reference:** When the serial data output (SOE="1") is selected, the pins operate as a serial data output pin regardless of the state of the I/O port.

13.4.3 Status Register (SSR0/SSR1)

The status register (SSR0/SSR1) is a register for performing the operations of status check for transmission/reception and errors, transfer direction setting of serial data, and enable/disable setting of send/receive interrupts.

■ Status Register (SSR0/SSR1)

Figure 13.4-4 Status Register (SSR0/SSR1)

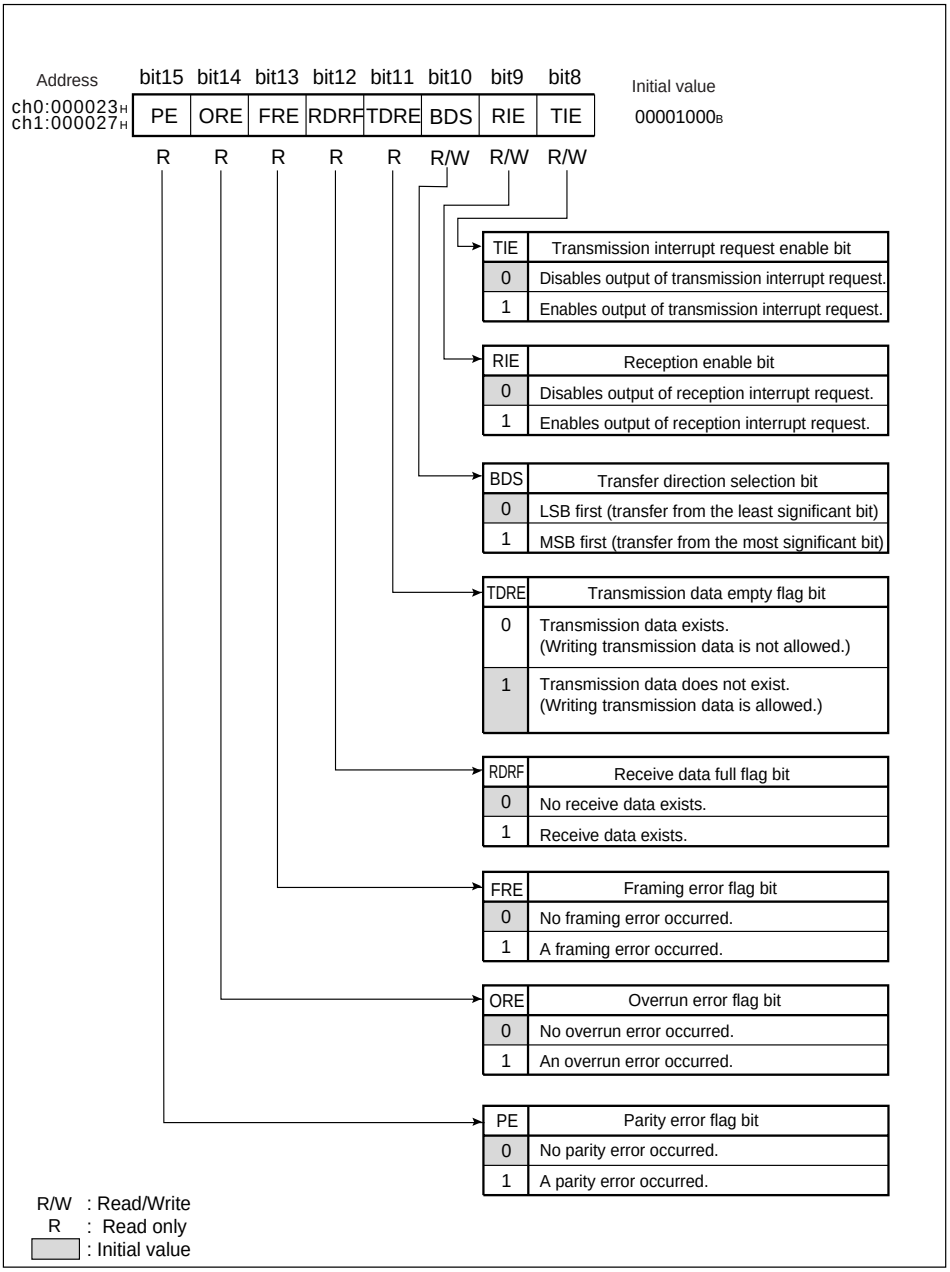


Table 13.4-3 Functions of Bits for Status Register (SSR0/SSR1)

No.	Bit name	Function
bit15	PE: Parity error flag bit	• This bit is set to "1" when a parity error occurs during reception. • This bit is cleared to "0" when "0" is written to the reception error flag clear bit (REC) of the control register (SCR0/SCR1). • When this bit is set to "1" while the reception interrupt request enable bit (RIE) is 1, a reception interrupt request is output. • When this bit is set to "1", data in the input data register (SIDR0/SIDR1) will be invalid.
bit14	ORE: Overrun error flag bit	• This bit is set to "1" when an overrun error occurs during reception. • This bit is cleared to "0" when "0" is written to the reception error flag clear bit (REC) of the control register (SCR0/SCR1). • When this bit is set to "1" while the reception interrupt request enable bit (RIE) is 1, a reception interrupt request is output. • When this bit is set to "1", data in the input data register (SIDR0/SIDR1) will be invalid.
bit13	FRE: Framing error flag bit	• This bit is set to "1" when a framing error occurs during reception. • This bit is cleared to "0" when "0" is written to the reception error flag clear bit (REC) of the control register (SCR0/SCR1). • When this bit is set to "1" while the reception interrupt request enable bit (RIE) is 1, a reception interrupt request is output. • When this bit is set to "1", data in the input data register (SIDR0/SIDR1) will be invalid.
bit12	RDRF: Receive data full flag bit	• This bit indicates the status of the input data register (SIDR0/SIDR1). • This bit is set to "1" when the receive data is stored in the input data register (SIDR0/SIDR1). • This bit is cleared to "0" when the input data register (SIDR0/SIDR1) is read. • When the reception interrupt request enable bit (RIE) is set to 1 while this bit is "1", a reception interrupt request is output.
bit11	TDRE: Transmission data empty flag bit	• This bit indicates the status of the output data register (SODR0/SODR1). • This bit is cleared to "0" when transmission data is written to the output data register (SODR0/SODR1). • This bit is set to "1" when data is sent after loading it into the transmission shift register. • When the transmission interrupt request enable bit (TIE) is set to 1 while this bit is "1", a transmission interrupt request is output. **Note:** "1" is set in the initial state.
bit10	BDS: Transfer direction selection bit	• This bit specifies the transfer direction of serial data. • When this bit is set to "0", transfer starts with the lowest-order bit (LSB first). • When this bit is set to "1", transfer starts with the highest-order bit (MSB first). **Note:** The high-order and low-order sides of data are interchanged with each other when reading from, or writing to the serial data register. Therefore, if data is written to the output data register (SODR0/SODR1) and then this bit is rewritten, the written data becomes invalid.

Table 13.4-3 Functions of Bits for Status Register (SSR0/SSR1) (Continued)

No.	Bit name	Function
bit9	RIE: Reception interrupt request enable bit	• This bit enables a reception interrupt request. • When the reception data full flag enable bit (RDRF) is set to "1" or one of the reception error flag bits (PE, ORE, and FRE) is set to "1" while this bit is "1", a reception interrupt request is output.
bit8	TIE: Transmission interrupt request enable bit	• This bit enables a transmission interrupt request. • When the transmission data empty flag bit (TDRE) is set to "1" while this bit is "1", a transmission interrupt request is output.

13.4.4 Input Data Register (SIDR0/SIDR1) and Output Data Register (SODR0/SODR1)

The input data register (SIDR0/SIDR1) is a serial data reception register. The output data register (SODR0/SODR1) is a serial data transmission register.

■ Input Data Register (SIDR0/SIDR1)

Figure 13.4-5 "Input Data Register (SIDR0/SIDR1)" shows the bit configuration of input data register .

Figure 13.4-5 Input Data Register (SIDR0/SIDR1)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ch0:000022H ch1:000026H	D7	D6	D5	D4	D3	D2	D1	D0	XXXXXXXXB
	R	R	R	R	R	R	R	R	

R : Read only
X : Indefinite

The input data register (SIDR0/SIDR1) is a register to store receive data. The serial data signal transmitted to the SIN0/SIN1 pin is converted in the shift register and then stored in the input data register (SIDR0/SIDR1). When the data length is 7 bits long in operation mode 0, bit7 (D7) becomes invalid. When receive data is stored in this register, the reception data full flag bit (RDRF) of the status register (SSR0/SSR1) is set to "1". If the reception interrupt request output is enabled (SSR0/SSR1: RIE="1"), a reception interrupt is output.

Read the input data register (SIDR0/SIDR1) when the reception data full flag bit (RDRF) of the status register (SSR0/SSR1) is set to "1". The reception data full flag bit (RDRF) is cleared to "0" when the input data register (SIDR0/SIDR1) is read. When a reception error occurs (one of SSR0/SSR1: PE, ORE, FRE is "1"), data in the input data register (SIDR0/SIDR1) becomes invalid.

■ Output Data Register (SODR0/SODR1)

Figure 13.4-6 Output Data Register (SODR0/SODR1)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ch0:000022H ch1:000026H	D7	D6	D5	D4	D3	D2	D1	D0	XXXXXXXXB
	W	W	W	W	W	W	W	W	

W : Write only
X : Indefinite

When data to be transmitted is written to the output data register (SODR0/SODR1), if transmission is enabled, the send data is transferred to the transmission shift register, converted into serial data, and then transmitted from the serial data output pin (SOT0/SOT1 pin). When the data length is 7 bits long in operation mode 0, bit7 (D7) becomes invalid.

When the transmission data is written to the output data register, the transmission data empty flag bit (TDRE) of the status register (SSR0/SSR1) is cleared to "0". When transfer to the transmission shift register is completed, the status register is set to "1". If the transmission data empty flag bit (TDRE) is "1", the next send data can be written. When the transmission data empty flag bit (TDRE) is set to "1" while the transmission interrupt request output is enabled (SSR0/SSR1: TIE="1"), a transmission interrupt is output. When a transmission interrupt is output, write the next transmission data after the transmission data empty flag bit (TDRE) is set to "1".

Note:

The output data register (SODR0/SODR1) is a write-only register and the input data register (SIDR0/SIDR1) is a read-only register. These registers are located at the same address, and so the read value is different from the write value. Therefore, instructions that perform a read-modify-write (RMW) operation such as the INC/DEC instruction cannot be used.

13.4.5 Communication Prescaler Control Register (CDCR0/ CDCR1)

The communication prescaler control register (CDCR0/CDCR1) is a register that controls the division of machine clocks.

■ Communication Prescaler Control Register (CDCR0/CDCR1)

The operation clocks of UART can be obtained by dividing machine clocks. UART is designed to obtain certain baud rates for various machine cycles. Output from the communication prescaler is used for the operation clocks of extended I-O serial interfaces.

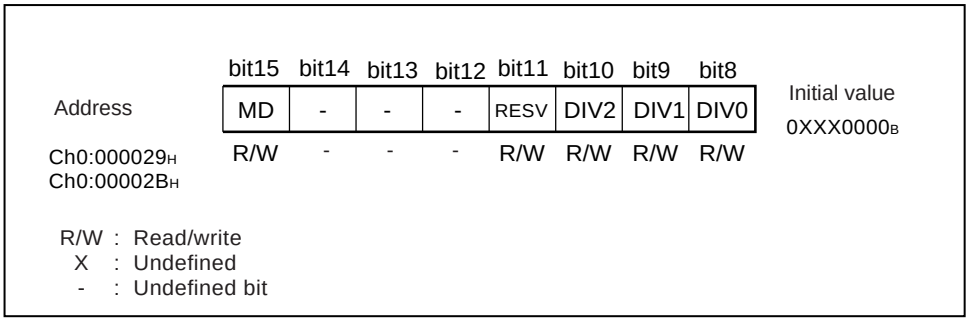


Table 13.4-4 Functions of Bits for Communication Prescaler Control Register (CDCR0/CDCR1)

No.	Bit name	Function
bit15	MD: Communication prescaler operation enable bit	• This bit enables a communication prescaler operation. • If this bit is "1", the communication prescaler operates. • If this bit is "0", the communication prescaler stops.
bit14 bit13 bit12	-: Undefined bit	• When these bits are read, values are undefined. • Values set to these bits do not affect operation.
bit11	RESV: Reservation bit	• Always set "0".
bit10 bit9 bit8	DIV2 to DIV0: Frequency division ratio setting bit	• These bits are used to set the frequency division ratio of the machine clock. • For the set values, see Table 13.4-5 "Machine Clock Division Ratio".

Table 13.4-5 Machine Clock Division Ratio

MD	DIV2	DIV1	DIV0	Div
0	-	-	-	Setting not allowed
1	0	0	0	1
1	0	0	1	2
1	0	1	0	3
1	0	1	1	4
1	1	0	0	5
1	1	0	1	6
1	1	1	0	7
1	1	1	1	8

Div: Machine clock division ratio

Note:

If a division ratio is changed, wait two time cycles as clock stabilization time before starting communication.

13.5 UART Interrupts

UART uses both reception and transmission interrupts and outputs an interrupt request for either of the following causes:

- **A reception interrupt is output when receive data is set to the input data register (SIDR0/SIDR1) or a reception error occurs.**
- **A transmission interrupt is output when send data is transferred from the output data register (SODR0/SODR1) to the transmission shift register.**

The extended intelligent I/O service (EI^{2}OS) is available for these interrupts.

■ UART Interrupts

Table 13.5-1 Interrupt Control Bits and Interrupt Causes of UART

Reception/ transmission	Interrupt request flag bit	Operation mode			Interrupt cause	Interrupt cause enable bit	When interrupt request flag is cleared
		0	1	2			
Reception	RDRF	O	O	O	Loading receive data into buffers (SIDR0/1)	SSR0/1:RIE	Receive data is read.
	ORE	O	O	O	Overrun error		0 is written to the reception error flag clear bit (SCR0/ 1:REC).
	FRE	O	O	X	Framing error		
	PE	O	X	X	Parity error		
Transmission	TDRE	O	O	O	Send data transfer from the output data register (SODR0/SODR1) completed	SSR0/1:TIE	Transmission data is written

O: Used
X: Not used

❍ Reception Interrupt

In reception mode, "1" is set to the reception data full flag bit (RDRF) or one of the reception error flag bits (ORE, FRE, PE) in the status register (SSR0/SSR1) when data reception is completed, or an overrun error, framing error, or parity error occurs. If the reception interrupt is enabled (SSR0/SSR1: RIE="1"), a reception interrupt request is output.

The reception data full flag bit (RDRF) of the status register (SSR0/SSR1) is cleared to "0" when the input data register (SIDR0/SIDR1) is read. All reception error flag bits (PE, ORE, FRE) of the status register (SSR0/SSR1) are cleared to "0" when the reception error flag clear bit (REC) of the status register (SSR0/SSR1) is set to "0".

❍ Transmission Interrupt

The transmission data empty flag bit (TDRE) of the status register (SSR0/SSR1) is set to "1" when send data is transferred from the output data register (SODR0/SODR1) to the transfer shift register. If the transmission interrupt is enabled (SSR0/SSR1: TIE="1"), a transmission interrupt request is output.

■ UART Interrupts and EI^{2}OS

Table 13.5-2 UART Interrupts and EI^{2}OS

Interrupt cause	Interrupt number	Interrupt control register		Vector table address			EI^{2}OS
		Register name	Address	Lower	Upper	Bank	
UART1 reception interrupt	#37(25_H)	ICR13	$0000BD_H$	$FFFF68_H$	$FFFF69_H$	$FFFF6A_H$	O
UART1 transmission interrupt	#38(26_H)	ICR13	$0000BD_H$	$FFFF64_H$	$FFFF65_H$	$FFFF66_H$	△
UART0 reception interrupt	#39(27_H)	ICR14	$0000BE_H$	$FFFF60_H$	$FFFF61_H$	$FFFF62_H$	O
UART0 transmission interrupt	#40(28_H)	ICR14	$0000BE_H$	$FFFF5C_H$	$FFFF5D_H$	$FFFF5E_H$	△

O: Provided with a function that detects a UART reception error and stops EI^{2}OS
△ : Usable when interrupt causes that share the ICR13 and ICR14 or the interrupt vectors are not used

■ UART EI^{2}OS Functions

UART has a circuit for operating EI^{2}OS, which can be started up for either reception or transmission interrupts.

❍ For Reception

EI^{2}OS can be used regardless of the status of other resources.

❍ For Transmission

UART shares the interrupt registers (ICR13 and ICR14) with the UART reception interrupts. Therefore, EI^{2}OS can be started up only when no UART reception interrupts are used.

13.5.1 Reception Interrupt Generation and Flag Set Timing

The following are reception interrupt causes: completion of reception (SSR0/SSR1: RDRF="1") and occurrence of a reception error (one of SSR0/SSR1: PE, ORE, and FRE is "1").

■ Reception Interrupt Generation and Flag Set Timing

Receive data is stored in the input data register (SIDR0/SIDR1) and the reception data full flag bit (RDRF) of the status register (SSR0/SSR1) is set to "1" when a stop bit is detected (in operation mode 0 or 1) or the last bit of data (D7) is detected (in operation mode 2). When a reception error occurs, one of the reception error flags (PE, ORE, FRE) is set to "1". If any of the reception error flag bits in each operation mode is set to "1", the value contained in the input data register (SIDR0/SIDR1) becomes invalid.

❍ Operation Mode 0 (Asynchronous, Normal Mode)

The reception data full flag bit (RDRF) of the status register (SSR0/SSR1) is set to "1" when a stop bit is detected. If a reception error is detected, "1" is set to one of the reception error flag bits (PE, ORE, FRE).

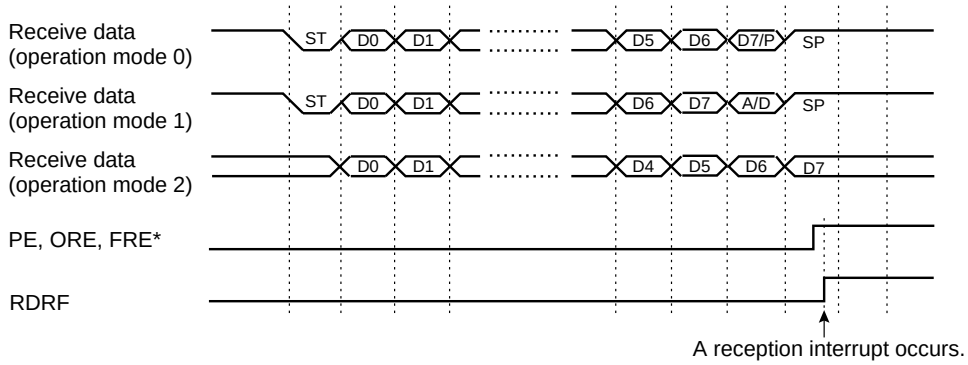
❍ Operation Mode 1 (Asynchronous, Multiprocessor Mode)

The reception data full flag bit (RDRF) of the status register (SSR0/SSR1) is set to "1" when a stop bit is detected. If a reception error is detected, "1" is set to either of the reception error flag bits (ORE, FRE). Parity errors cannot be detected.

❍ Operation Mode 2 (Synchronous, Normal Mode)

The reception data full flag bit (RDRF) of the status register (SSR0/SSR1) is set to "1" when the last bit (D7) of receive data is detected. If a reception error is detected, "1" is set to the reception error flag bit (ORE). Parity and framing errors cannot be detected.

Figure 13.5-1 Reception Operation and Flag Set Timing



* : The PE flag cannot be used in mode 1.
The PE and PRE flags cannot be used in mode 2.
ST : Start bit
SP : Stop bit
A/D : Mode 2 (multiprocessor mode) address/data selection bit

❍ Reception Interrupt Output Timing

When the reception data full flag bit (RDRF) of the status register (SSR0/SSR1) or the one of the reception error flag bits (PE, ORE, FRE) is set to "1" while the reception interrupt is enabled (SSR0/SSR1: RIE="1"), a reception interrupt request is output.

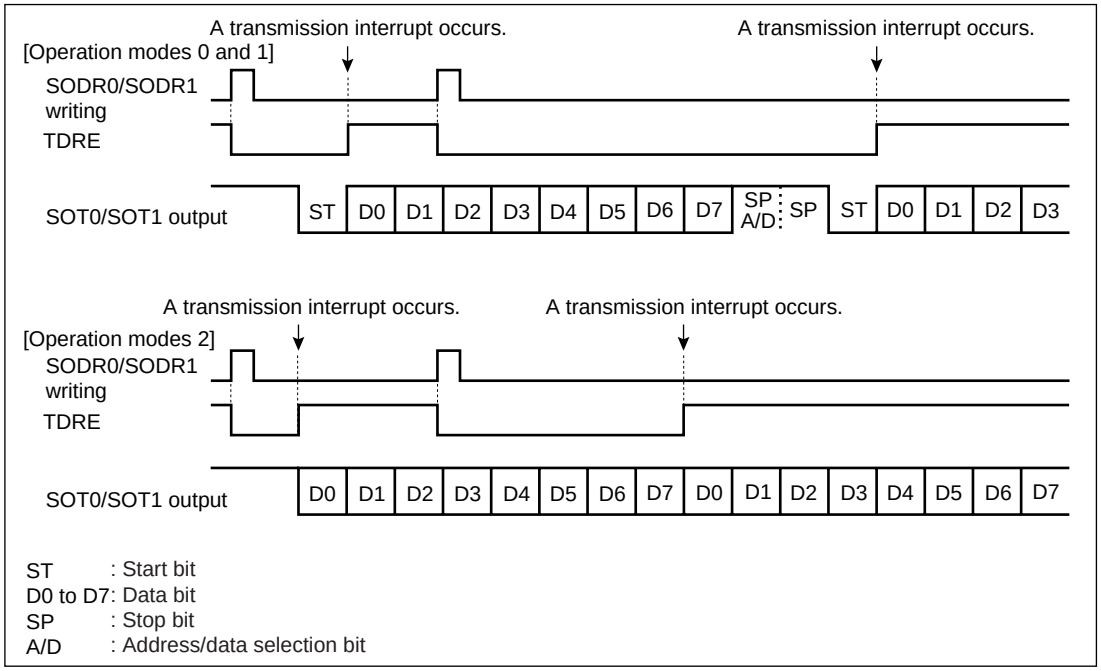
13.5.2 Transmission Interrupt Generation and Flag Set Timing

A transmission interrupt is output when send data is transferred from the output data register (SODR0/SODR1) to the transmission shift register and the next piece of data is ready to be written.

■ Transmission Interrupt Output and Flag Set Timing

The transmission data empty flag bit (TDRE) of the status register (SSR0/SSR1) is set to "1" when data written to the output data register (SODR0/SODR1) is transferred to the transmission shift register and the next piece of data is ready to be written. The transmission data empty flag bit (TDRE) is cleared to "0" when send data is written to the output data register (SODR0/SODR1).

Figure 13.5-2 Transmission Operation and Flag Set Timing



❍ Transmission Interrupt Request Output Timing

When the transmission data empty flag bit (TDRF) of the status register (SSR0/SSR1) is set to "1" while the transmission interrupt is enabled (SSR0/SSR1: TIE="1"), a transmission interrupt request is output.

Note:

Because the transmission data empty flag bit (TDRF) of the status register (SSR0/SSR1) is set to "1" in the initial state, a transmission interrupt request is output when the transmission interrupt is enabled (SSR0/SSR1: TIE="1"). The transmission data empty flag bit (TDRE) is a read-only bit. Accordingly, the only way to clear it is to write new data to the output data register (SODR0/SODR1). Specify carefully the timing for enabling a transmission interrupt.

13.6 UART Baud Rates

One of the following can be selected as the UART transmitting/receiving block:

- **Dedicated baud rate generator**
- **Internal clock (16-bit reload timer 0)**
- **External clock (clock input to the SCK0/SCK1 pin)**

■ UART Baud Rate Selection

The baud rate selection circuit is designed as shown below. One of the following three types of baud rates can be selected:

❍ Baud Rates Determined Using the Dedicated Baud Rate Generator

UART has an internal dedicated baud rate generator. One of six baud rates can be selected using the mode control register (SMR0/1).

An asynchronous or clock synchronous baud rate is selected using the machine clock and CS2 to CS0 bits of the mode control register (SMR0/1) .

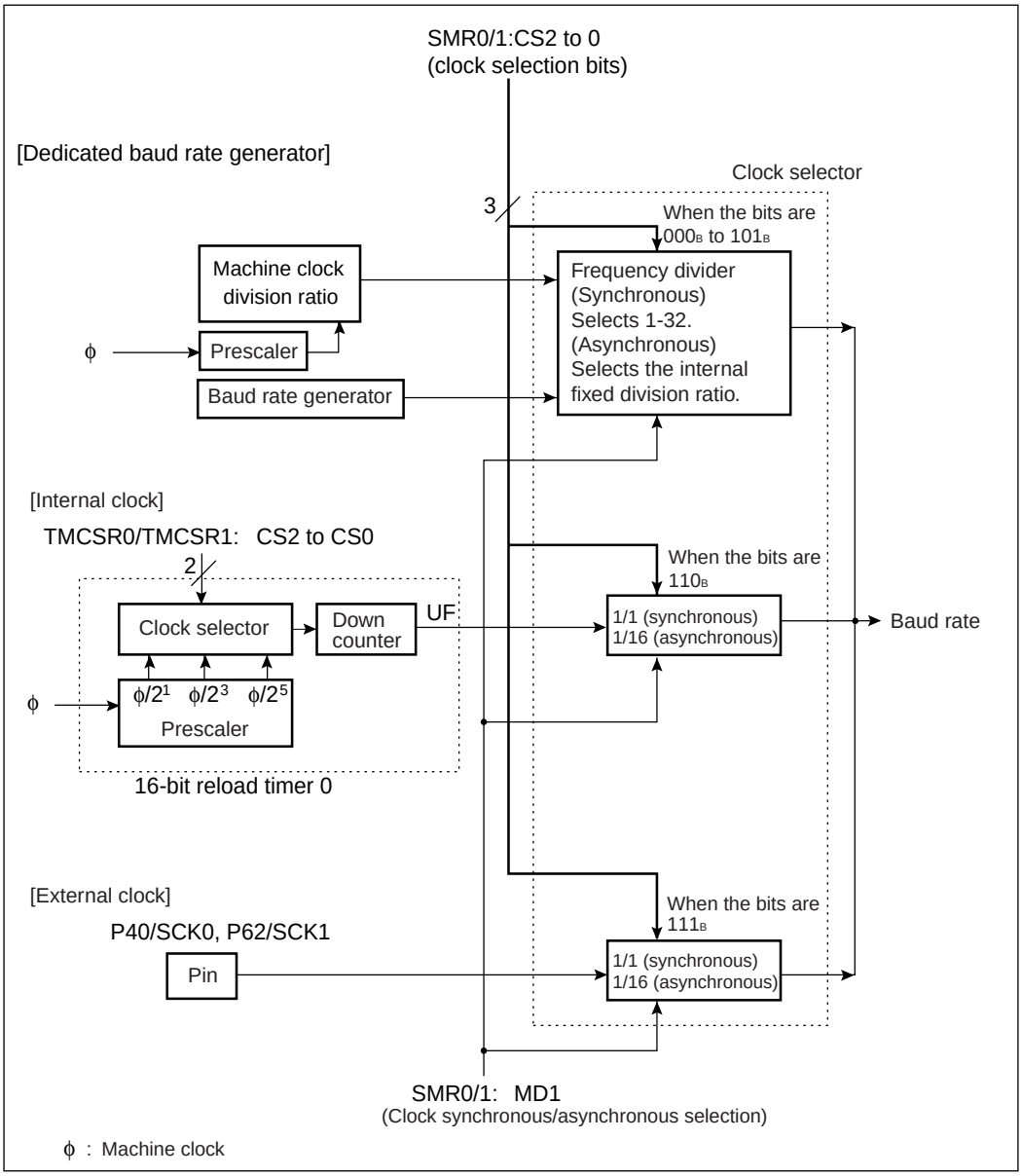
❍ Baud Rates Determined Using the Internal Timer

The internal clock supplied from 16-bit reload timer is used as is (synchronous) or by dividing it by 16 (asynchronous) for the baud rate. Any baud rate can be set by setting the reload value.

❍ Baud Rates Determined Using the External Clock

The clock input from the UART clock pulse input pins (P40/SCK0 and P62/SCK1) is used as is (synchronous) or by dividing it by 16 (asynchronous) for the baud rate. Any baud rate can be set externally.

Figure 13.6-1 Baud Rate Selection Circuit



13.6.1 Baud Rates Determined Using the Dedicated Baud Rate Generator

This section describes the baud rates that can be set when the clock from the dedicated baud rate generator is selected as the UART transfer clock.

■ Baud Rates Determined Using the Dedicated Baud Rate Generator

When the clock setting bits of the mode register (SMR0/SMR1) are set to "000_B-101_B", the baud rate is set by the dedicated baud rate generator.

When the transfer clock is generated by the dedicated baud rate generator, the machine clock is divided by the machine clock prescaler and then divided by using the transfer clock division ratio set by the clock selector. The machine clock division ratios are common to the asynchronous and synchronous baud rates, but the transfer clock division ratio is set by the clock setting bits (CS2 to CS0) of the mode register (SMR0/SMR1) separately for the asynchronous and synchronous baud rates.

The actual transfer ratio can be calculated by using the following formulas:

asynchronous baud rate= ϕ / (prescaler division ratio)/(asynchronous transfer clock division ratio)

synchronous baud rate= ϕ / (prescaler division ratio)/(synchronous transfer clock division ratio)

ϕ : Machine clock frequency

❍ Division Ratios for the Prescaler (Common to Asynchronous and Synchronous Baud Rates)

As listed in Table 13.6-1 "Setting of Each Division Ratio by the Machine Clock Prescaler", the machine clock division ratio is set by the division ratio setting bits (DIV2 to DIV0) of the communication prescaler control register (CDCR0/CDCR1).

Table 13.6-1 Setting of Each Division Ratio by the Machine Clock Prescaler

MD	DIV2	DIV1	DIV0	Div
0	-	-	-	Setting not allowed
1	0	0	0	1
1	0	0	1	2
1	0	1	0	3
1	0	1	1	4
1	1	0	0	5
1	1	0	1	6
1	1	1	0	7
1	1	1	1	8

Div: Mchine clock division ratio

❍ Synchronous Transfer Clock

The synchronous baud rate is set, as listed in Table 13.6-2 "Synchronous Baud Rate Setting", by the clock setting bits (CS2 to CS0) of the mode register (SMR0/SMR1).

Table 13.6-2 Synchronous Baud Rate Setting

CS2	CS1	CS0	Division ratio for CLK synchronization	Calculation formula
0	0	0	2 Mbps	(ϕ / div)/1
0	0	1	1 Mbps	(ϕ / div)/2
0	1	0	500 kbps	(ϕ / div)/4
0	1	1	250 kbps	(ϕ / div)/8
1	0	0	125 kbps	(ϕ / div)/16
1	0	1	62.5 kbps	(ϕ / div)/32

However, the baud rate is calculated based on the values of ϕ (machine clock) = 16MHz and div (machine clock division ratio) = 8.

❍ Asynchronous Transfer Clock Division Ratios

Asynchronous baud rates is selected using the CS2 to CS0 bits of the mode control register (SMR0/1) as listed in Table 13.6-3 "Selection of Synchronous Baud Rate Division Ratios".

Table 13.6-3 Selection of Synchronous Baud Rate

CS2	CS1	CS0	Asynchronus (start-stop synchronization)	Calculation formula
0	0	0	76,923 bps	(ϕ / div)/(8x13x2)
0	0	1	38,461 bps	(ϕ / div)/(8x13x4)
0	1	0	19,230 bps	(ϕ / div)/(8x13x8)
0	1	1	9,615 bps	(ϕ / div)/(8x13x16)
1	0	0	500 kbps	(ϕ / div)/(8x2x2)
1	0	1	250 kbps	(ϕ / div)/(8x2x4)

However, the baud rate is calculated based on the values of ϕ (machine clock) = 16MHz and div (machine clock division ratio) = 1.

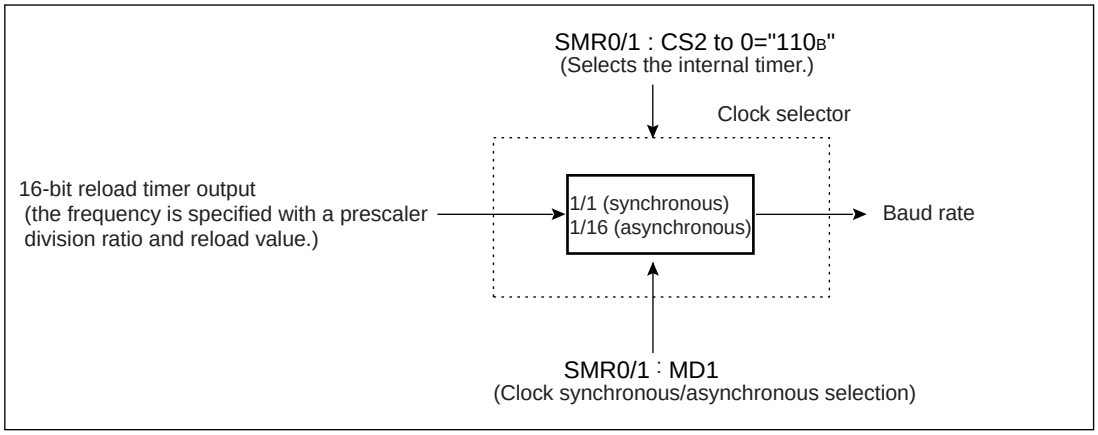
13.6.2 Baud Rates Determined Using the Internal Timer (16-bit Reload Timer)

This section describes the settings used when the internal clock supplied from 16-bit reload timer is selected as the UART transfer clock. It also shows the baud rate calculation formulas.

■ Baud Rates Determined Using the Internal Timer (16-bit Reload Timer)

If the clock setting bits (CS2 to CS0) of the mode register (SMR0/SMR1) are set to "110B", the baud rate is set by the internal clock. The baud rate can be set by specifying the prescaler division ratio and reload value of the 16-bit reload timer.

Figure 13.6-2 Baud Rate Selection Circuit for the Internal Timer (16-Bit Reload Timer)



❍ Baud Rate Calculation Formulas

Asynchronous baud rate = (ϕ / N)/(16x2x (n+1)) bps

Synchronous baud rate = (ϕ / N)/(2x (n+1)) bps

ϕ: Machine clock

N: Division ratio for the prescaler of 16-bit reload timer 0 (2^1 , 2^3, or 2^5)

n: Reload value for 16-bit reload timer 0 (0 to 65535)

❍ Examples of Setting Reload Values (Machine Clock: 7.3728 MHz)

Table 13.6-4 Baud Rates and Reload Values

Baud rate	Reload value (n)			
	Clock asynchronous (start-stop synchronization)		Clock synchronous	
	$N=2^1$ (machine cycle divided by 2)	$N=2^3$ (machine cycle divided by 8)	$N=2^1$ (machine cycle divided by 2)	$N=2^3$ (machine cycle divided by 8)
38400	2	-	47	11
19200	5	-	95	23
9600	11	2	191	47
4800	23	5	383	95
2400	47	11	767	191
1200	95	23	1535	383
600	191	47	3071	767
300	383	95	6143	1535

N: Division ratio for the prescaler of 16-bit reload timer 0
-: Setting not allowed

Note:

Specify 1/2 of the system clock as the transfer rate in operation mode 2 (CLK synchronous mode).

13.6.3 Baud Rates Determined Using the External Clock

This section describes the settings used when the external clock is selected as the UART transfer clock. It also shows the baud rate calculation formulas.

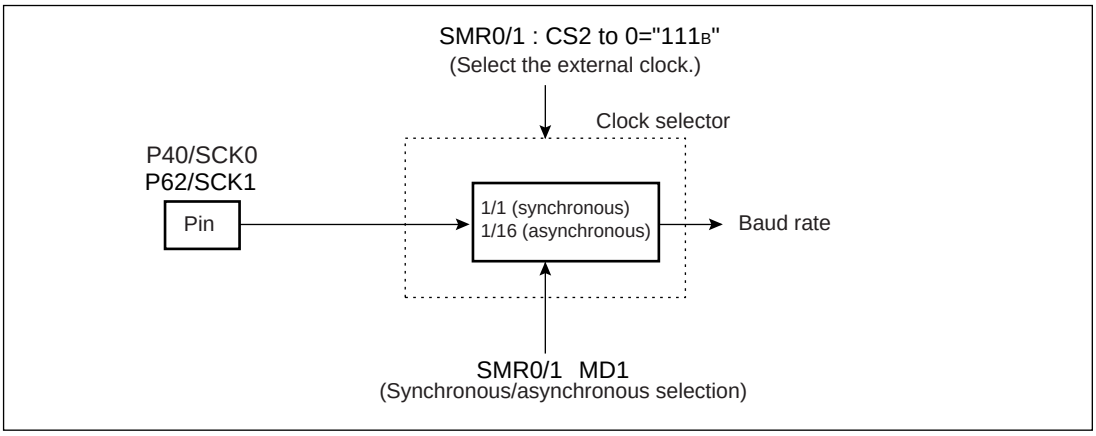
■ Baud Rates Determined Using the External Clock

The following three settings are required to select the baud rate determined by using the external clock:

- Write 111_B to the CS2 to CS0 bits of the mode control register (SMR0/1) to select the baud rate determined by using the external clock input.
- Set the P40/SCK0 and P62/SCK1 pins as input ports (DDR4: bit 0 = 0 and DDR6: bit 2 = 0).
- Write 0 to the SCKE bit of the mode control register (SMR0/1) to set the pin as an external clock input pin.

As shown in Figure 13.6-3 "Baud Rate Selection Circuit for the External Clock", a baud rate is selected using the external clock input from the SCK0/1 pin. To change the baud rate, the external input clock cycle must be changed because the internal division ratio is fixed.

Figure 13.6-3 Baud Rate Selection Circuit for the External Clock



❍ Baud Rate Calculation Formulas

asynchronous baud rate = f/16 bps

synchronous baud rate = f bps

f: External clock (up to 2 MHz)

13.7 Operation of UART

UART operates in operation modes 0 and 2 for normal bidirectional serial communication and in operation mode 1 for master-slave communication.

■ Operation of UART

❍ Operation modes

There are three UART operation modes: modes 0 to 2. As listed in Table 13.7-1 "UART Operation Mode", an operation mode can be selected according to the inter-CPU connection method and data transfer mode.

Table 13.7-1 UART Operation Mode

Operation mode		Data length		Synchroni-zation mode	Stop bit length
		When parity is disabled	When parity is enabled		
0	Normal mode	7 or 8 bits		Asynchronous	1 or 2 bits*2
1	Multiprocessor mode	8+1*1	-	Asynchronous	
2	Normal mode	8	-	Synchronous	None

-: Setting not possible
*1: "+1" indicates the address/data selection bit (A/D) for communication control.
*2: During reception, only one stop bit can be detected.

Note:

Operation mode 1 of UART is used only from the master system during master-slave connection.

❍ Inter-CPU Connection Method

One-to-one connection (normal mode) and master-slave connection (multiprocessor mode) can be selected. For either connection method, the data length, whether to enable parity, and the synchronization method must be common to all CPU. Select an operation mode as follows:

- In the one-to-one connection method, operation mode 0 or 2 must be used in the two CPUs. Select operation mode 0 for asynchronous transfer mode and operation mode 2 for synchronous transfer mode.
- Select operation mode 1 for the master-slave connection method and use it from the master system. Select "When parity is disabled" for this connection method.

❍ Synchronization Method

Asynchronous mode (start-stop synchronization) or clock synchronous mode can be selected in any operation mode.

❍ **Signal Method**

UART can treat data only in non-return to zero (NRZ) format.

❍ **Operation Enable**

UART controls both transmission and reception using the control register (SCR0/SCR1) with its transmission enable bit (TXE) and reception enable bit (RXE). If any of the operations is disabled during operation, the following will occur:

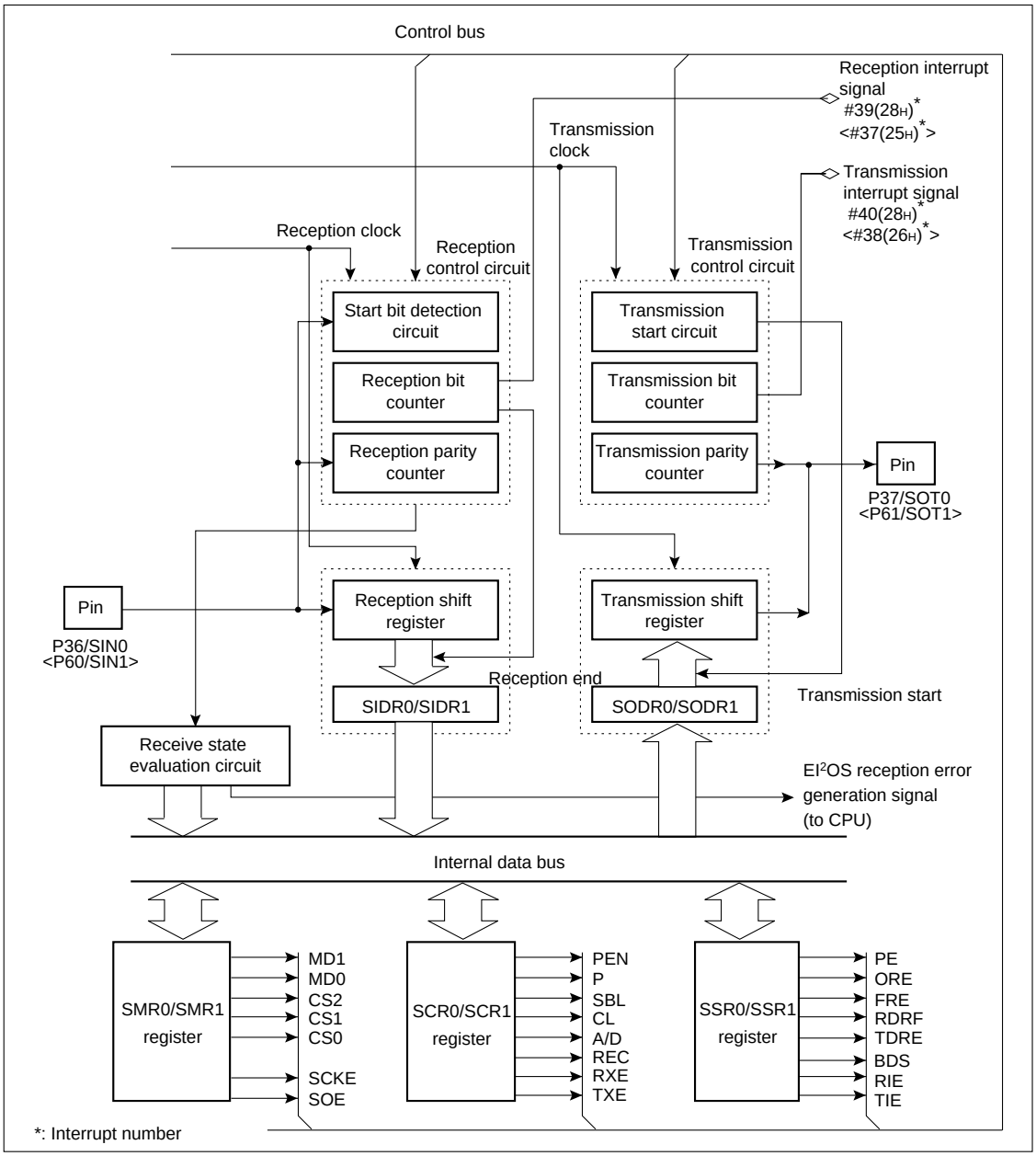
- If reception operation is disabled during reception (data is input to the reception shift register), finish frame reception and store the received data in the input data register (SIDR0/1). Then stop the reception operation.
- If the transmission operation is disabled during transmission (data is output from the transmission shift register), wait until there is no data in the output data register (SIDR0/1) before stopping the transmission operation.
- When mode 1 is selected for URAT, the received data bit 9 is ignored.

13.7.1 Operation in Asynchronous Mode (Operation Modes 0 and 1)

When UART is used in operation mode 0 (normal mode) or operation mode 1 (multiprocessor mode), asynchronous transfer mode is selected.

■ Operation in Asynchronous Mode

Figure 13.7-1 Block Diagram in Asynchronous Mode (Operation Mode 0/1)

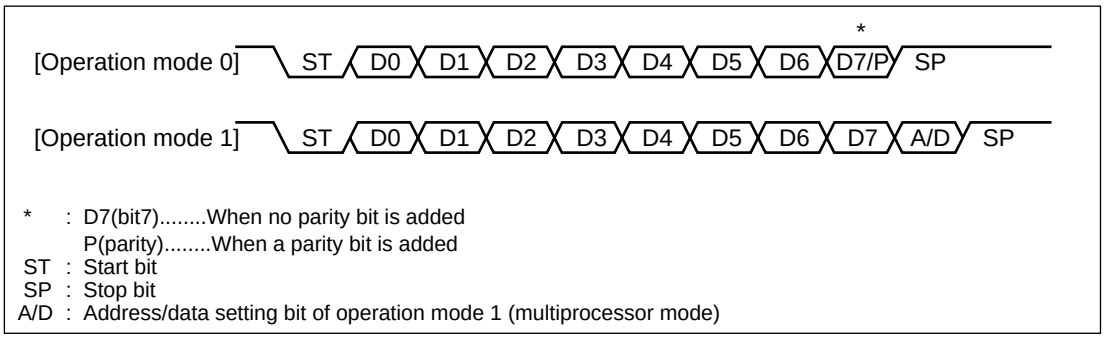


❍ Transfer Data Format

Transfer data always starts with the start bit ("L" level) and ends with the stop bit ("H" level). The data of the specified data bit length is transferred in "LSB first."

- In operation mode 1, the data is fixed to eight bits with an address/data setting bit (A/D) bit instead of a parity bit.

Figure 13.7-2 Transfer Data Format (Operation Modes 0 and 1)



* : D7(bit7).....When no parity bit is added
P(parity).....When a parity bit is added
ST : Start bit
SP : Stop bit
A/D : Address/data setting bit of operation mode 1 (multiprocessor mode)

❍ Transmission Operation

When the transmission data empty flag bit (TDRE) of the status register (SSR0/SSR1) is set to "1", send data is written to the output data register (SODR0/SODR1). If the transmission is enabled (SCR0/SCR1: TXE="1"), data is sent.

When the send data is transferred to the transmission shift register and transmission is started, the transmission data empty flag bit (TDRE) of the status register (SSR0/SSR1) is set to "1" again so that the next piece of send data can be set. If the transmission interrupt request output is enabled (SSR0/SSR1: TIE="1"), a transmission interrupt request is output so that send data is set to the output data register (SODR0/SODR1). The transmission data empty flag bit (TDRE) is cleared to "0" after writing the send data to the output data register (SODR0/SODR1).

❍ Reception Operation

If the reception is enabled (SCR0/SCR1: RXE="1"), reception operation is always performed. When a start bit is detected, one frame of data is received in accordance with the data format specified in the control register (SCR0/SCR1). When reception of one frame is completed, if a reception error occurs, one of the reception error flag bits (PE, ORE, FRE) of the status register (SSR0/SSR1) is set to "1" and then the reception data full flag bit (RDRF) is set to "1". If the reception interrupt request output is enabled (SSR0/SSR1: RIE="1"), a reception interrupt request is output. Check each of the reception error flag bits (PE, ORE, FRE) of the status register (SSR0/SSR1). If no error occurred in reception, read the input data register (SIDR0/SIDR1). If an error has occurred, take the appropriate measures to deal with the error. The reception data full flag bit (RDRF) is cleared to "0" after reading receive data from the input data register (SIDR0/SIDR1).

❍ Stop Bit

One or two stop bits can be set for transmission. The receiving side always evaluates the first one bit.

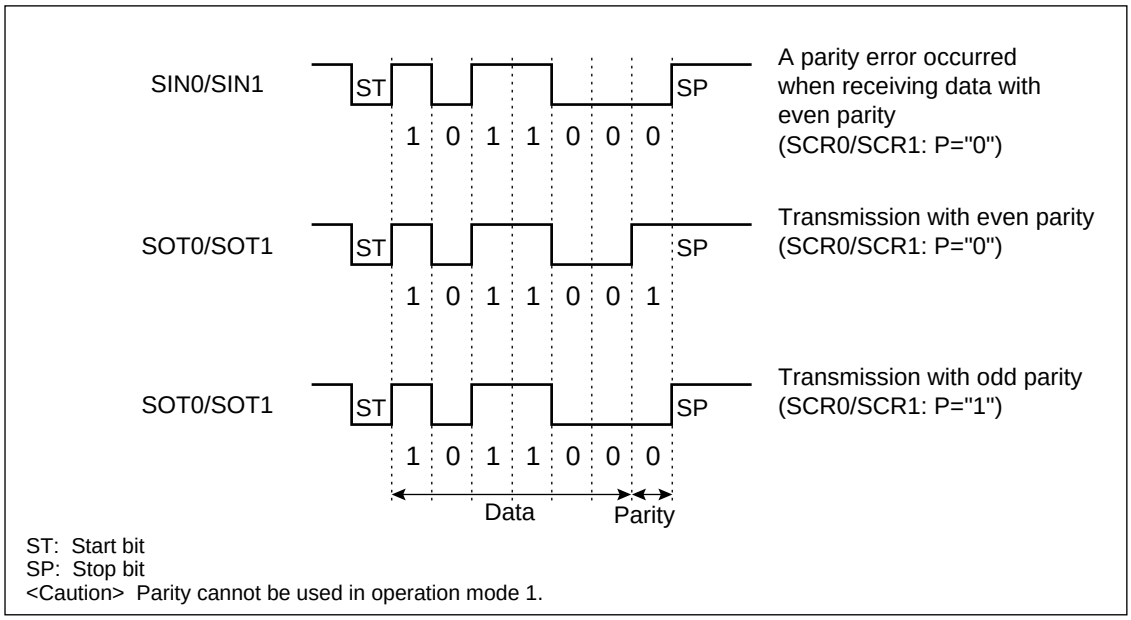
❍ Error Detection

- In operation mode 0, parity, overrun, and framing errors can be detected.
- In operation mode 1, overrun and framing errors can be detected, but parity errors cannot be detected.

❍ **Parity**

Parity can be used only in operation mode 0. The parity presence/absence can be set in the parity enable bit (PEB) of the control register (SCR0/SCR1) and even parity/odd parity can be set in the parity setting bit (P). Parity cannot be used in operation mode 1.

Figure 13.7-3 Transmission Data when Parity is Enabled



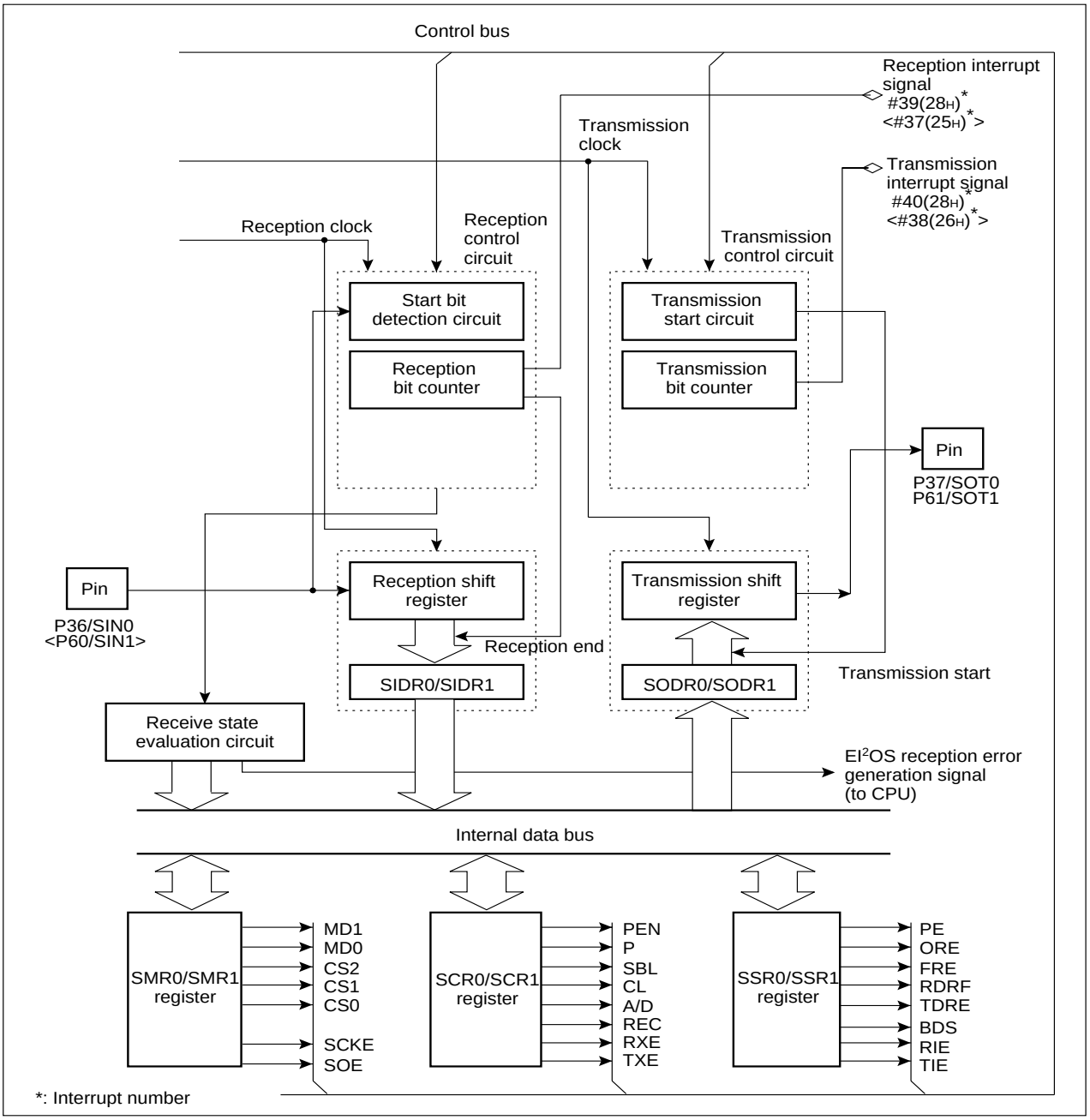
ST: Start bit
SP: Stop bit
<Caution> Parity cannot be used in operation mode 1.

13.7.2 Operation in Synchronous Mode (Operation Mode 2)

When UART is used in operation mode 2 (normal mode), the synchronous transfer mode is selected.

■ Operation in Synchronous Mode (Operation Mode 2)

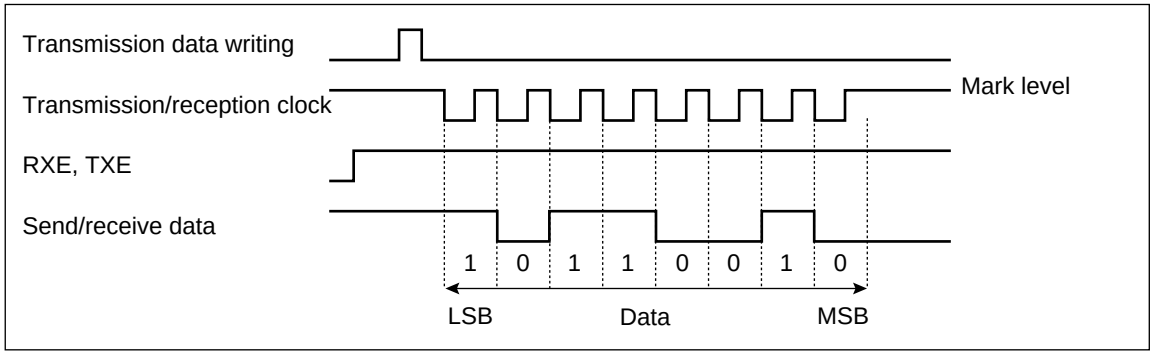
Figure 13.7-4 Block Diagram in Synchronous Mode (Operation Mode 2)



❍ **Transfer Data Format**

In synchronous mode, 8-bit data is transferred in LSB first mode.

Figure 13.7-5 Transfer Data Format (Operation Mode 2)



❍ **Clock Supply**

In clock synchronous (I/O extended serial) mode, as many clocks as the number of transmission/reception bits need to be supplied.

- If the internal clock is selected, a data reception synchronous clock is generated when data is received.
- If an external clock is selected, a clock for exactly one byte needs to be supplied from an external source after making sure that the output data register (SODR0/SODR1) on the sending side UART contains data (SSR0/SSR1: TDRE="0"). The mark level "H" must be retained before transmission starts and after it is completed.

❍ **Error Detection**

Only overrun errors can be detected, and parity and framing errors cannot be detected.

❍ **Initialization**

The following shows the values to be set to use synchronous mode:

[Mode register (SMR0/SMR1)]

MD1, MD0: Set "10_B".

CS2, CS1, CS0: Specify the clock input of the clock selector.

SCKE: Set "1" for the dedicated baud rate generator or the internal clock. Set "0" for the clock output or an external clock.

SOE: Set "1" for transmission. Set "0" for reception.

[Control register (SCR0/SCR1)]

PEN: Set "0".

P, SBL, A/D: These bits make no sense.

CL: Set "1" (8-bit data).

REC: Set "0" (Clear all error flags for initialization).

RXE, TXE: Set "1" to either of the bits.

[Status register (SSR0/SSR1)]

RIE: Set "1" to use interrupts and "0" to not use interrupts.

TIE: Set "1" to use interrupts and "0" to not use interrupts.

❍ **Communication Start**

Write data to the output data register (SODR0/SODR1) start communication. Note that the receiving side must also write send data to the output data register (SODR0/SODR1) to start communication.

❍ **Communication End**

The reception data full flag bit (RDRF) of the status register (SSR0/SSR1) is set to "1" when transmission or reception of one frame of data is completed. When receiving data, check the overrun error flag bit (ORE) to see if the communication has been normally executed.

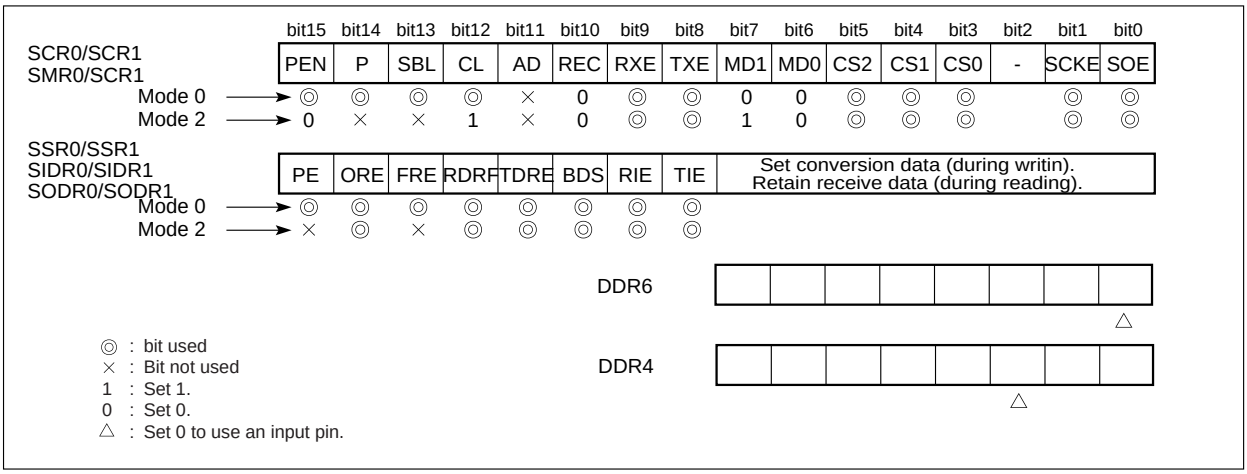
13.7.3 Bidirectional Communication Function (Normal Mode)

In operation mode 0 or 2, serial bidirectional communication (one-to-one connection) is available. Select operation mode 0 for asynchronous communication and operation mode 2 for synchronous communication.

■ Bidirectional Communication Function

The settings shown in Figure 13.7-6 "Settings for UART1 Operation Mode 0" are required to operate UART1 in normal mode (operation mode 0 or 2).

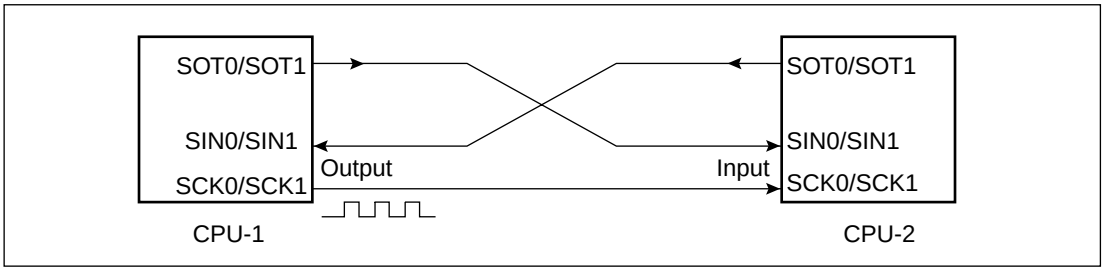
Figure 13.7-6 Settings for UART1 Operation Mode 0



❍ Inter-CPU Connection

As shown in Figure 13.7-7 "Connection Example of UART1 Bidirectional Communication", interconnect two CPU.

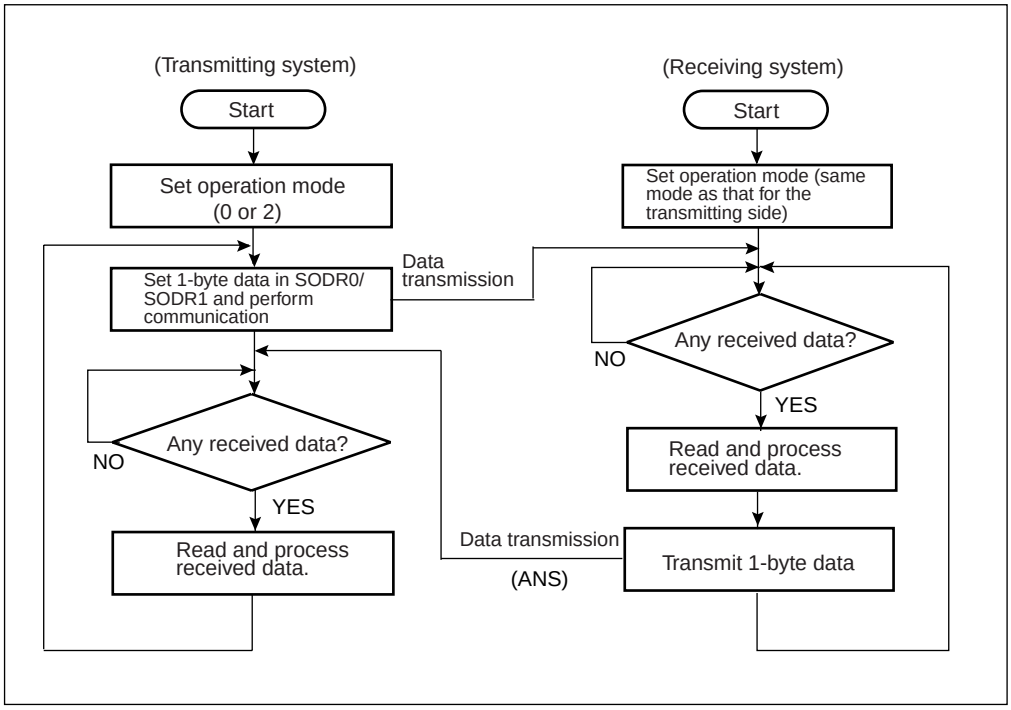
Figure 13.7-7 Connection Example of UART1 Bidirectional Communication



❍ Communication Procedure

Communication starts from the transmitting system when transmission data has been prepared. An ANS is returned periodically (byte by byte in this example) when the receiving system receives transmission data.

Figure 13.7-8 Example of Bidirectional Communication Flowchart



13.7.4 Master-slave Communication Function (Multiprocessor Mode)

With UART, communication with multiple CPU connected in master-slave mode is available. However, UART can be used only from the master system.

■ Master-slave Communication Function

The settings shown in Figure 13.7-9 "Settings for UART1 Operation Mode 1" are required to operate UART1 in multiprocessor mode (operation mode 1).

Figure 13.7-9 Settings for UART1 Operation Mode 1

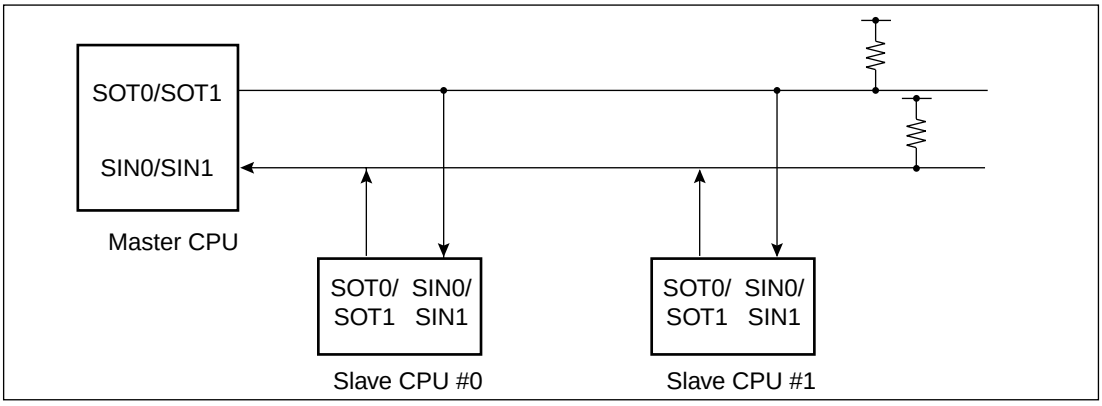
	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
SCR0/SCR1 SMR0/SMR1	PEN	P	SBL	CL	AD	REC	RXE	TXE	MD1	MD0	CS2	CS1	CS0	-	SCKE	SOE
	0	×	◎	1	◎	0	◎	◎	0	1	◎	◎	◎		0	◎
SSR0/SSR1 SIDR0/SIDR1 SODR0/SODR1	PE	ORE	FRE	RDRF	TDRE	BDS	RIE	TIE	Set transmission data (during writing). Retain receive data (during reading).							
	×	◎	◎	◎	◎	◎	◎	◎								
DDR6																
																△

◎ : Bit used
× : Bit not used
1 : Set 1.
0 : Set 0.
△ : Set 0 to use an input pin.

❍ Inter-CPU Connection

As shown in Figure 13.7-10 "Connection Example of UART Master-Slave Communication", a communication system consists of one master CPU and multiple slave CPU connected to two communication lines. UART1 can be used only from the master CPU.

Figure 13.7-10 Connection Example of UART Master-Slave Communication



❍ **Function Selection**

Select the operation mode and data transfer mode for master-slave communication as shown in Table 13.7-2 "Selection of the Master-Slave Communication Function".

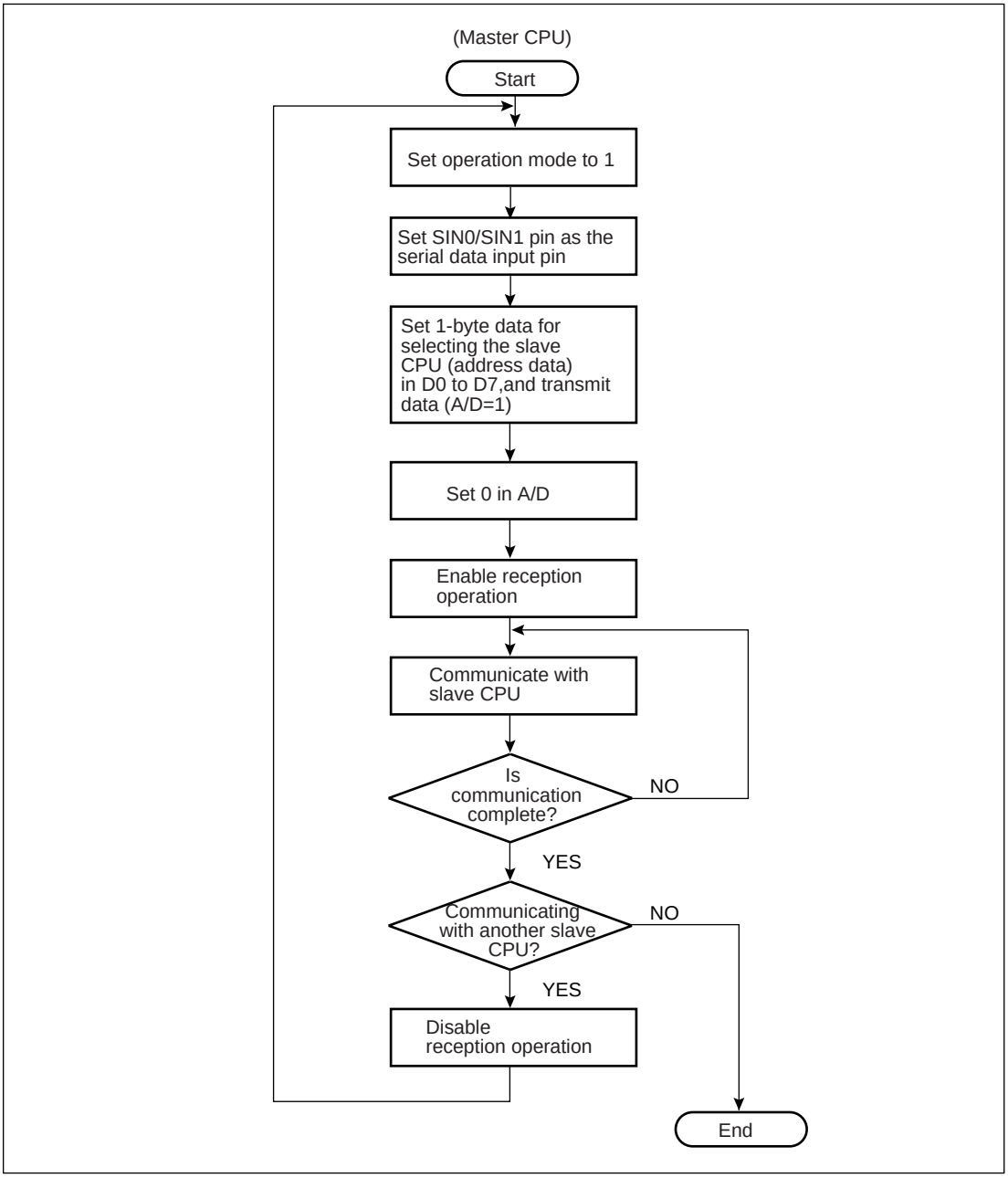
Table 13.7-2 Selection of the Master-Slave Communication Function

	Operation mode		Data	Parity	Synchroni-zation method	Stop bit
	Master CPU	Slave CPU				
Address transmission and reception	Operation mode 1	-	A/D="1" + 8-bit address	None	Asynchronous	1 or 2 bits
Data transmission and reception			A/D="0" + 8-bit address			

❍ **Communication Procedure**

When the master CPU transmits address data, communication starts. The A/D bit in the address data is set to 1, and the communication destination slave CPU is selected. Each slave CPU checks the address data using a program. When the address data indicates the address assigned to a slave CPU, the slave CPU communicates with the master CPU (ordinary data).

Figure 13.7-11 Master-Slave Communication Flowchart



13.8 Notes on Using UART

Notes on using UART are given below.

■ Notes on Using UART

❍ Enabling Operations

UART has the transmission enable bit (TXE) and reception enable bit (RXE) in the control register (SCR0/SCR1). Both transmission and reception operations need to be enabled before starting transfer because the transmission/reception enable bits (TXE/RXE) are set to "0" in the initial state. The transfer can also be canceled by disabling the transmission/reception as required.

❍ Communication Mode Setting

Set the communication mode while the system is not operating. If the mode is set during transmission or reception, the transmission or reception data is not guaranteed.

❍ Synchronous Mode

UART clock synchronous mode (operation mode 2) uses clock control (I/O extended serial) mode, in which start and stop bits are not added to the data.

❍ Transmission Interrupt Enabling Timing

The default (initial value) of the transmission data empty flag bit (SSR0/1: TDRE) is 1 (no transmission data and transmission data write enable state). A transmission interrupt request is generated as soon as the transmission interrupt requests are enabled (SSR0/1: TIE=1). Be sure to set the TIE flag to 1 after setting the transmission data.

❍ Reception in Operation Mode 1 (Multiprocessor Mode)

In operation mode 1 (multiprocessor mode) of UART, the 9-bit receive operation cannot be performed.

CHAPTER 14 DTP/EXTERNAL INTERRUPT CIRCUIT

This chapter describes the functions and operation of the DTP/external interrupt circuit.

14.1 Overview of the DTP/External Interrupt Circuit

The data transfer peripheral (DTP)/external interrupt circuit detects interrupt request input from the external interrupt input pins (INT7 to INT0) to output interrupt requests.

■ DTP/External Interrupt Circuit Functions

The DTP/external interrupt circuit function outputs an interrupt request when it detects an edge or level signal input to the external interrupt input pins (INT7 to INT0).

When an interrupt request is accepted by the CPU and the extended intelligent I/O service (EI^2OP) is enabled, the automatic data transfer (DTP function) is performed by EI^2OP before branching to the interrupt processing routine. If EI^2OP is disabled, the automatic data transfer (DTP function) by EI^2OP is not activated; instead, direct branching to the interrupt processing routine takes place.

Table 14.1-1 Overview of the DTP/External Interrupt Circuit

	External interrupt function	DTP function
Input pins	Eight (P10/INT0 to P16/INT6 and P63/INT7/DTTI)	
Interrupt cause	By using the request level setting register (ELVR), the detection level or edge type can be set for each pin.	
	Input of the "L" level/"H" level/rising edge/falling edge	
Interrupt number	#25 (19_H) - #28 ($1C_H$)	
Interrupt control	The output of interrupt requests is enabled and disabled using the DTP/interrupt enable register (ENIR).	
Interrupt flag	Interrupt causes are stored in the DTP/interrupt cause register (EIRR).	
Processing selection	EI^2OS is disabled (ICR: ISE = 0).	EI^2OS is enabled (ICR: ISE = 1).
Processing	The circuit branches to an external interrupt processing routine.	Branching to the interrupt processing routine after automatic data transfer by EI^2OS

ICR: Interrupt control register

■ Interrupt of the DTP/external interrupt circuit and EI^{2}OS

Table 14.1-2 Interrupt of the DTP/External Interrupt Circuit and EI^{2}OS

Channel	Interrupt number	Interrupt control register		Vector table address			EI^{2}OS
		Register name	Address	Lower	Upper	Bank	
INT0/INT1	#25 (19_H)	ICR07	$0000B7_H$	$FFFF98_H$	$FFFF99_H$	$FFFF9A_H$	△
INT2/INT3	#26 ($1A_H$)			$FFFF94_H$	$FFFF95_H$	$FFFF96_H$	
INT4/INT5	#27 ($1B_H$)	ICR08	$0000B8_H$	$FFFF90_H$	$FFFF91_H$	$FFFF92_H$	
INT6/INT7	#28 ($1C_H$)			$FFFF8C_H$	$FFFF8D_H$	$FFFF8E_H$	

△ : Available when not using an interrupt request that shares the ICR07 and ICR08 registers.

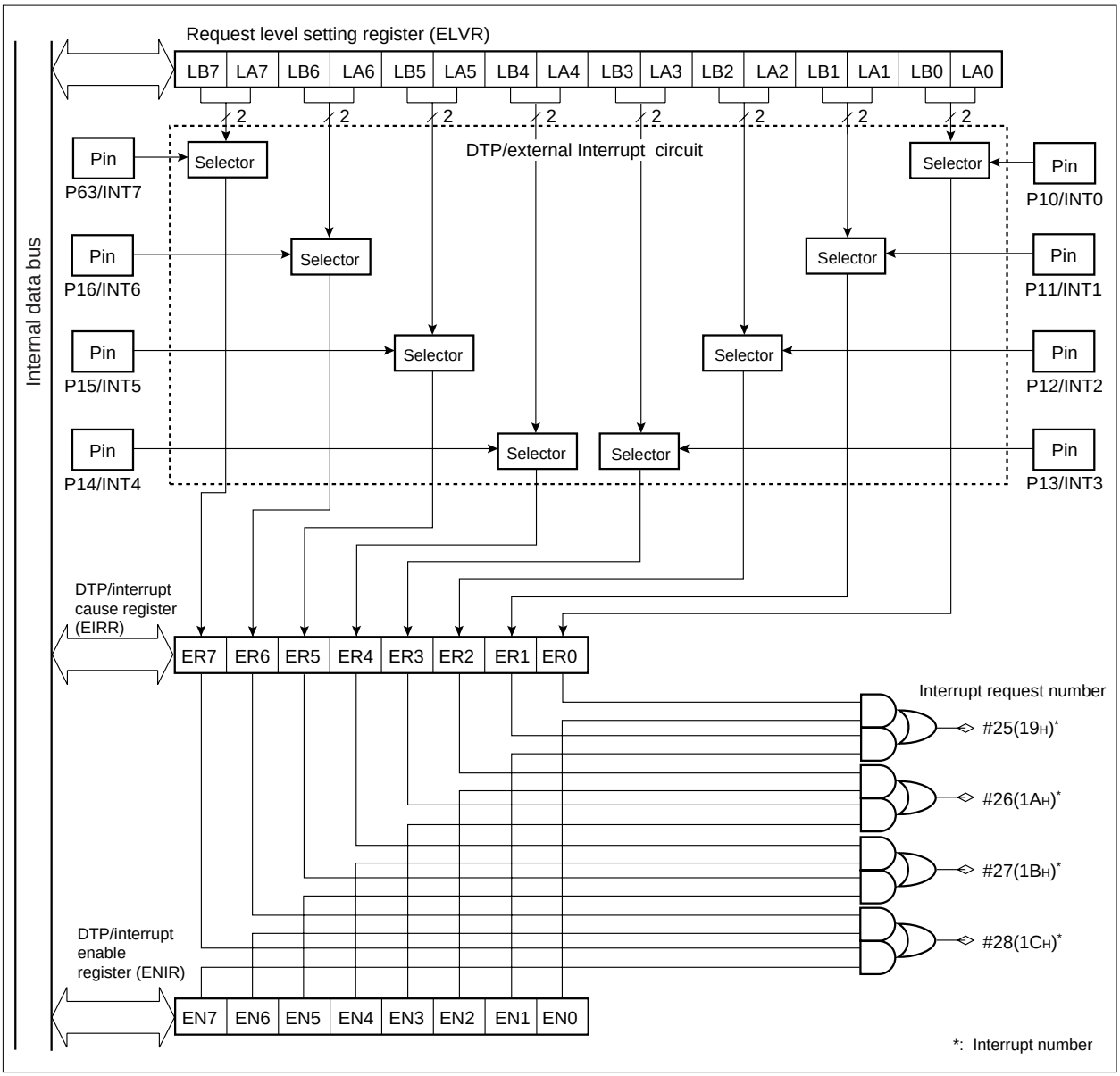
14.2 Configuration of the DTP/External Interrupt Circuit

The DTP/external interrupt circuit consists of four blocks:

- **DTP/interrupt input detection circuit**
- **Request level setting register (ELVR)**
- **DTP/interrupt cause register (EIRR)**
- **DTP/interrupt enable register (ENIR)**

■ Block Diagram of the DTP/External Interrupt Circuit

Figure 14.2-1 Block Diagram of the DTP/External Interrupt Circuit



- **DTP/external interrupt input detection circuit**

  Upon detecting a match of the signal input to an external interrupt input pin (INT7 to INT0) and the level or edge specified in the request level setting register (ELVR), the DTP/external interrupt cause flag bit (EIRR: ER7 to ER0) corresponding to the external interrupt input pin (INT7 to INT0) is set to "1".

- **Request level setting register (ELVR)**

  This register sets the detection condition (level or edge) of interrupt requests for each external interrupt input pin (INT7 to INT0).

- **DTP/interrupt cause register (EIRR)**

  This register retains and clears interrupt causes.

- **DTP/interrupt enable register (ENIR)**

  This register enables/disables interrupt requests for each external interrupt input pin (INT7 to INT0).

14.3 DTP/External Interrupt Circuit Pins

This section describes the DTP/external interrupt circuit pins and provides a pin block diagram.

■ DTP/External Interrupt Circuit Pins

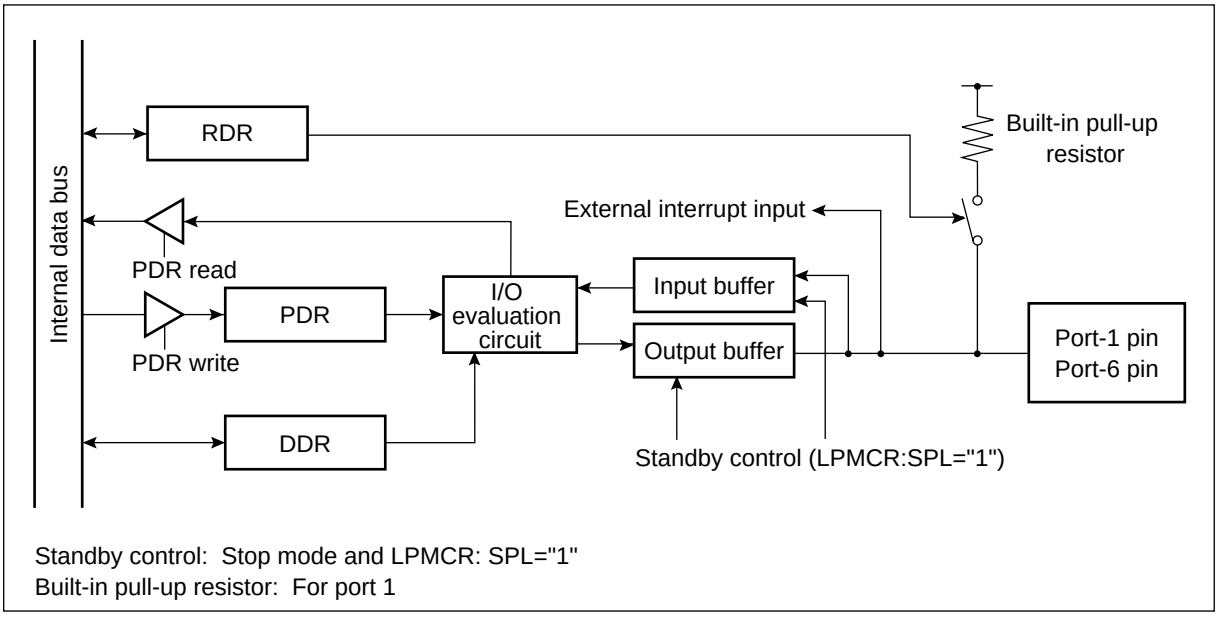
The DTP/external interrupt circuit pins are also used as I/O ports.

Table 14.3-1 DTP/External Interrupt Circuit Pins

Pin name	Function	I/O format	Pull-up resistor	Standby control	Setting required to use pins
P10/INT0	Port 1 input-output/external interrupt input	CMOS output/ CMOS hysteresis input	Not provided	Not provided	Set the pin as an input port (DDR1: bit8 = 0)
P11/INT1					Set the pin as an input port (DDR1: bit9 = 0)
P12/INT2					Set the pin as an input port (DDR1: bit10 = 0)
P13/INT3					Set the pin as an input port (DDR1: bit11 = 0)
P14/INT4					Set the pin as an input port (DDR1: bit12 = 0)
P15/INT5					Set the pin as an input port (DDR1: bit13 = 0)
P16/INT6					Set the pin as an input port (DDR1: bit14 = 0)
P63/INT7	Port 6 input-output/external interrupt input/ Dead Time Timer Interrupt Input				Set the pin as an input port (DDR6: bit3 = 0)

■ Block Diagram of the DTP/External Interrupt Circuit Pins

Figure 14.3-1 Block Diagram of the DTP/External Interrupt Circuit Pins



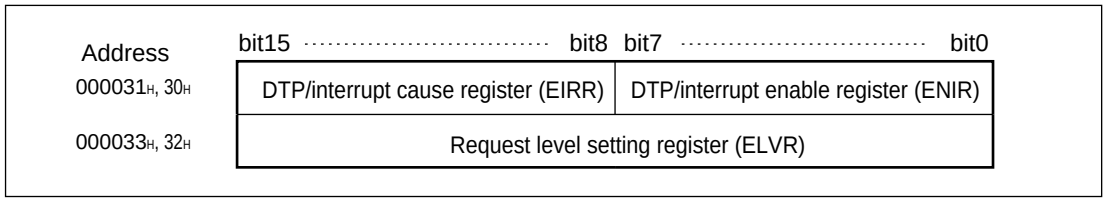
Standby control: Stop mode and LPMCR: SPL="1"
Built-in pull-up resistor: For port 1

14.4 DTP/External Interrupt Circuit Registers

This section describes IDTP/external interrupt circuit registers.

■ DTP/External Interrupt Circuit register

Figure 14.4-1 DTP/External Interrupt Circuit Registers

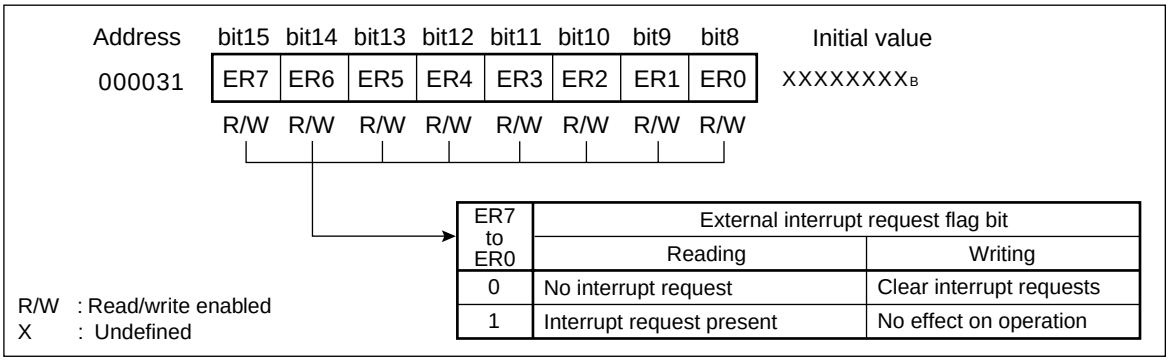


14.4.1 DTP/Interrupt Cause Register (EIRR)

The DTP/interrupt cause register (EIRR) stores and clears interrupt causes.

■ **DTP/Interrupt Cause Register (EIRR)**

Figure 14.4-2 DTP/Interrupt Cause Register (EIRR)



ER7 to ER0	External interrupt request flag bit	
	Reading	Writing
0	No interrupt request	Clear interrupt requests
1	Interrupt request present	No effect on operation

Table 14.4-1 Function Description of Each Bit of the DTP/Interrupt Cause Register (EIRR)

Bit name			Function
bit15	ER7:	External interrupt request flag bit	• This bit is a flag that requests an interrupt. • This bit is set to "1" when the level or edge signal set in the external interrupt request detection condition setting bit (LB7, LA7) of the request level setting register (ELVR) is detected in the external interrupt input pin (INT7). • When this bit is set to "1" while the external interrupt request enable bit (EN7) of the DTP/external interrupt enable register (ENIR) is set to "1", an interrupt request is output. • When this bit is "0", the interrupt request is cleared. • When this bit is "1", operation is not affected.
bit14	ER6:		• This bit is a flag that requests an interrupt. • This bit is set to "1" when the level or edge signal set in the external interrupt request detection condition setting bit (LB6, LA6) of the request level setting register (ELVR) is detected in the external interrupt input pin (INT6). • When this bit is set to "1" while the external interrupt request enable bit (EN6) of the DTP/external interrupt enable register (ENIR) is set to "1", an interrupt request is output. • When this bit is "0", the interrupt request is cleared. • When this bit is "1", operation is not affected.
bit13	ER5:		• This bit is a flag that requests an interrupt. • This bit is set to "1" when the level or edge signal set in the external interrupt request detection condition setting bit (LB5, LA5) of the request level setting register (ELVR) is detected in the external interrupt input pin (INT5). • When this bit is set to "1" while the external interrupt request enable bit (EN5) of the DTP/external interrupt enable register (ENIR) is set to "1", an interrupt request is output. • When this bit is "0", the interrupt request is cleared. • When this bit is "1", operation is not affected.
bit12	ER4:		• This bit is a flag that requests an interrupt. • This bit is set to "1" when the level or edge signal set in the external interrupt request detection condition setting bit (LB4, LA4) of the request level setting register (ELVR) is detected in the external interrupt input pin (INT4). • When this bit is set to "1" while the external interrupt request enable bit (EN4) of the DTP/external interrupt enable register (ENIR) is set to "1", an interrupt request is output. • When this bit is "0", the interrupt request is cleared. • When this bit is "1", operation is not affected.

Table 14.4-1 Function Description of Each Bit of the DTP/Interrupt Cause Register (EIRR) (Continued)

Bit name			Function
bit11	ER3:	External interrupt request flag bit	• This bit is a flag that requests an interrupt. • This bit is set to "1" when the level or edge signal set in the external interrupt request detection condition setting bit (LB3, LA3) of the request level setting register (ELVR) is detected in the external interrupt input pin (INT3). • When this bit is set to "1" while the external interrupt request enable bit (EN3) of the DTP/external interrupt enable register (ENIR) is set to "1", an interrupt request is output. • When this bit is "0", the interrupt request is cleared. • When this bit is "1", operation is not affected.
bit10	ER2:		• This bit is a flag that requests an interrupt. • This bit is set to "1" when the level or edge signal set in the external interrupt request detection condition setting bit (LB2, LA2) of the request level setting register (ELVR) is detected in the external interrupt input pin (INT2). • When this bit is set to "1" while the external interrupt request enable bit (EN2) of the DTP/external interrupt enable register (ENIR) is set to "1", an interrupt request is output. • When this bit is "0", the interrupt request is cleared. • When this bit is "1", operation is not affected.
bit9	ER1:		• This bit is a flag that requests an interrupt. • This bit is set to "1" when the level or edge signal set in the external interrupt request detection condition setting bit (LB1, LA1) of the request level setting register (ELVR) is detected in the external interrupt input pin (INT1). • When this bit is set to "1" while the external interrupt request enable bit (EN1) of the DTP/external interrupt enable register (ENIR) is set to "1", an interrupt request is output. • When this bit is "0", the interrupt request is cleared. • When this bit is "1", operation is not affected.
bit8	ER0:		• This bit is a flag that requests an interrupt. • This bit is set to "1" when the level or edge signal set in the external interrupt request detection condition setting bit (LB0, LA0) of the request level setting register (ELVR) is detected in the external interrupt input pin (INT0). • When this bit is set to "1" while the external interrupt request enable bit (EN0) of the DTP/external interrupt enable register (ENIR) is set to "1", an interrupt request is output. • When this bit is "0", the interrupt request is cleared. • When this bit is "1", operation is not affected.

Reference:

When the extended intelligent I/O service (EI^2OS) is activated as a DTP function, the corresponding external interrupt request flag bit (ER7 to ER0) is cleared to "0" when the transfer of one piece of data is completed.

Note:

When setting an external interrupt request flag bit (ER7 to ER0) to "0", be sure to set the bit from which "1" is read by software to "0". If an external interrupt request flag bit (ER7 to ER0) is set to "1" by hardware when setting it to "0", it will be cleared to "0".

14.4.2 DTP/Interrupt Enable Register (ENIR)

The DTP/interrupt enable register (ENIR) enables/disables an external interrupt request for each external interrupt pin (INT7 to INT0).

■ **DTP/Interrupt Enable Register (ENIR)**

Figure 14.4-3 DTP/Interrupt Enable Register (ENIR)

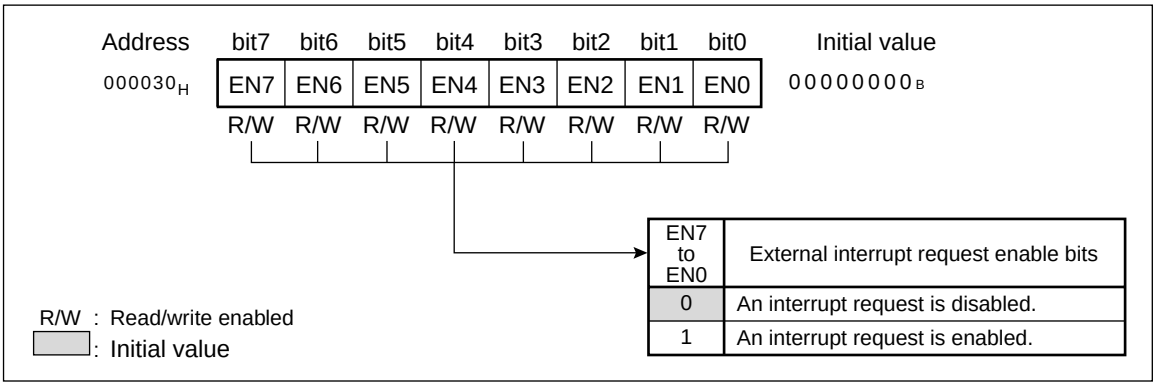


Table 14.4-2 Function Description of Each Bit of the DTP/Interrupt Enable Register (ENIR)

Bit name			Function
bit7	EN7:	External Interrupt request enable bit	• This bit enables an interrupt request. • When the external interrupt request flag bit (ER7) of the DTP/interrupt cause register (EIRR) is set to "1" while this bit is set to "1", an interrupt request is output.
bit6	EN6:		• This bit enables an interrupt request. • When the external interrupt request flag bit (ER6) of the DTP/interrupt cause register (EIRR) is set to "1" while this bit is set to "1", an interrupt request is output.
bit5	EN5:		• This bit enables an interrupt request. • When the external interrupt request flag bit (ER5) of the DTP/interrupt cause register (EIRR) is set to "1" while this bit is set to "1", an interrupt request is output.
bit4	EN4:		• This bit enables an interrupt request. • When the external interrupt request flag bit (ER4) of the DTP/interrupt cause register (EIRR) is set to "1" while this bit is set to "1", an interrupt request is output.
bit3	EN3:		• This bit enables an interrupt request. • When the external interrupt request flag bit (ER3) of the DTP/interrupt cause register (EIRR) is set to "1" while this bit is set to "1", an interrupt request is output.
bit2	EN2:		• This bit enables an interrupt request. • When the external interrupt request flag bit (ER2) of the DTP/interrupt cause register (EIRR) is set to "1" while this bit is set to "1", an interrupt request is output.
bit1	EN1:		• This bit enables an interrupt request. • When the external interrupt request flag bit (ER1) of the DTP/interrupt cause register (EIRR) is set to "1" while this bit is set to "1", an interrupt request is output.
bit0	EN0:		• This bit enables an interrupt request. • When the external interrupt request flag bit (ER0) of the DTP/interrupt cause register (EIRR) is set to "1" while this bit is set to "1", an interrupt request is output.

Reference:

To use an external interrupt input pin (INT7 to INT0) that also serves as an I/O port, set the bit that also serves the corresponding I/O port of the port direction register (DDR) to "0" to use the pin as an input port.

The states of the external interrupt input pins (INT7 to INT0) can be read directly using the port data register (PDR) regardless of the states of the external interrupt request enable bits (ENIR: EN7 to EN0).

External interrupt request flag bits (ER7 to ER0) of the DTP/interrupt cause register (EIRR) are set to "1" regardless of the values of the external interrupt request enable bits (ENIR: EN7 to EN0) when a DTP/external interrupt request signal is detected.

Table 14.4-3 Correspondence between the DTP/Interrupt Control Registers (EIRR and ENIR) and Each Channel

DTP/external interrupt pin	Interrupt number	External interrupt request flag bit (EIRR)	External interrupt request enable bit (ENIR)
P63/INT7	#28 ($1C_H$)	ER7	EN7
P16/INT6		ER6	EN6
P15/INT5	#27 ($1B_H$)	ER5	EN5
P14/INT4		ER4	EN4
P13/INT3	#26 ($1A_H$)	ER3	EN3
P12/INT2		ER2	EN2
P11/INT1	#25 (19_H)	ER1	EN1
P10/INT0		ER0	EN0

14.4.3 Request Level Setting Rregister (ELVR)

The request level setting register (ELVR) sets the detection condition (level or edge) for interrupt requests for each external interrupt input pin (INT7 to INT0).

■ Request Level Setting Register (ELVR)

Figure 14.4-4 Request Level Setting Register (ELVR)

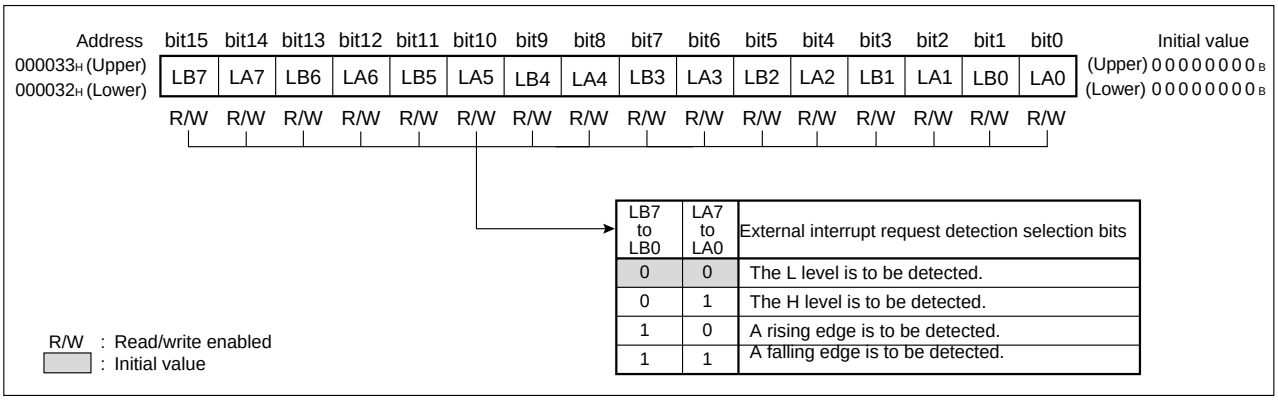


Table 14.4-4 Function Description of Each Bit of the Request Level Setting Register (ELVR)

Bit name			Function
Bit15	LB7:	External interrupt request detection condition setting bit	• This bit is used to set the detection condition (level or edge) for interrupt requests from the signal input to the external interrupt input pin (INT7).
bit14	LA7:		
bit13	LB6:		• This bit is used to set the detection condition (level or edge) for interrupt requests from the signal input to the external interrupt input pin (INT6).
bit12	LA6:		
bit11	LB5:		• This bit is used to set the detection condition (level or edge) for interrupt requests from the signal input to the external interrupt input pin (INT5).
bit10	LA5:		
bit9	LB4:		• This bit is used to set the detection condition (level or edge) for interrupt requests from the signal input to the external interrupt input pin (INT4).
bit8	LA4:		
bit7	LB3:		• This bit is used to set the detection condition (level or edge) for interrupt requests from the signal input to the external interrupt input pin (INT3).
bit6	LA3:		
bit5	LB2:		• This bit is used to set the detection condition (level or edge) for interrupt requests from the signal input to the external interrupt input pin (INT2).
bit4	LA2:		
bit3	LB1:		• This bit is used to set the detection condition (level or edge) for interrupt requests from the signal input to the external interrupt input pin (INT1).
bit2	LA1:		
bit1	LB0:		• This bit is used to set the detection condition (level or edge) for interrupt requests from the signal input to the external interrupt input pin (INT0).
bit0	LA0:		

Reference:

When the detection signal set in the request level setting register (ELVR) is input to an external interrupt input pin (INT7 to INT0), the external interrupt request flag bit (EIRR: ER7 to ER0) of the corresponding pin is set to "1" regardless of the setting in the DTP/interrupt enable register (ENIR).

Table 14.4-5 Correspondence between Request Level Setting Register (ELVR) and Each Channel

External interrupt input pin	Interrupt number	Bit name
P63/INT7	#28 ($1C_H$)	LB7, LA7
P16/INT6		LB6, LA6
P15/INT5	#27 ($1B_H$)	LB5, LA5
P14/INT4		LB4, LA4
P13/INT3	#26 ($1A_H$)	LB3, LA3
P12/INT2		LB2, LA2
P11/INT1	#25 (19_H)	LB1, LA1
P10/INT0		LB0, LA0

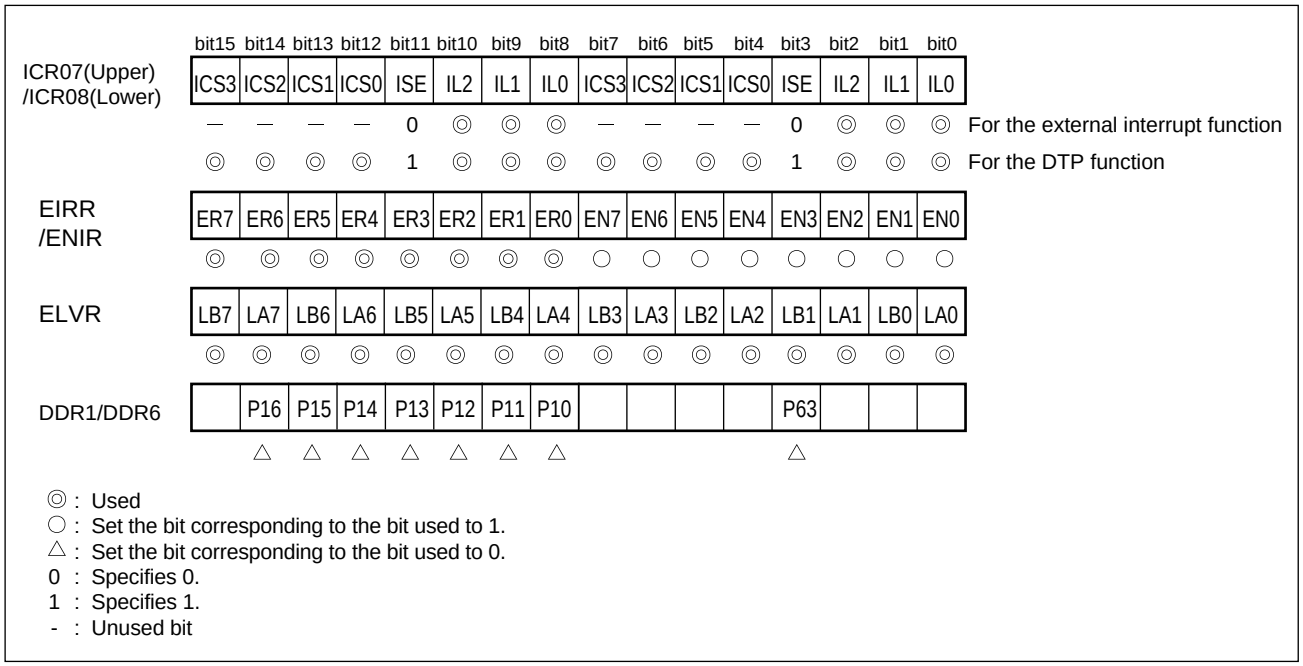
14.5 Operation of the DTP/External Interrupt Circuit

The DTP/external interrupt circuit provides the external interrupt function and the DTP function. This section describes the settings required for each function and the operation of the circuit.

■ Setting the DTP/External Interrupt Circuit

Figure 14.5-1 "DTP/External Interrupt Circuit" shows the settings required to operate the DTP/external interrupt circuit.

Figure 14.5-1 DTP/External Interrupt Circuit



Set the DTP/external interrupt circuit registers in accordance with the following procedure because an interrupt request may be generated erroneously when setting them.

1. Set the DTP/interrupt enable register (ENIR) to "00_H" to disable interrupt requests.
2. Set the interrupt detection condition to the external interrupt request detection condition setting bit (LB7 to LB0, LA7 to LA0) corresponding to the external interrupt input pin (INT7 to INT0) of the request level setting register (ELVR).
3. Set the external interrupt request flag bit (ER7 to ER0) corresponding to the external interrupt input pin (INT7 to INT0) of the DTP/interrupt cause register (EIRR) to "0" to clear the interrupt request.
4. To enable an interrupt request, set the external interrupt request enable bit (EN7 to EN0) corresponding to the external interrupt input pin (INT7 to INT0) of the DTP/interrupt enable register (ENIR) to "1".

❍ Switching between the external interrupt function and the DTP function

Switching between the external interrupt function and the DTP function is set by the EI^{2}OS enable bit (ISE) of the interrupt control register (ICR) corresponding to the interrupt cause to be used. When the EI^{2}OS enable bit (ISE) is set to "1", the extended intelligent I/O service (EI^{2}OS) is enabled, operating as the DTP function. When the EI^{2}OS enable bit (ISE) is set to "0", the extended intelligent I/O service (EI^{2}OS) is disabled, and the register operates as the external interrupt function.

■ Operation of the DTP/External Interrupt Circuit

Table 14.5-1 Control Bit and Interrupt Cause of the DTP/External Interrupt Circuit

	DTP/external interrupt circuit
External interrupt request flag bit	EIRR: ER7 to ER0
External interrupt request enable bit	ENIR: EN7 to EN0
Interrupt cause	Input of an effective edge or level to pin INT7 to INT0

The DTP/external interrupt circuit outputs an interrupt request to the interrupt controller when, after operations are set to the request level setting register (ELVR), DTP/interrupt cause register (EIRR), and DTP/interrupt enable register (ENIR), the detection condition set in the request level setting register (ELVR) is input to the corresponding external interrupt input pin (INT7 to INT0). When the EI^{2}OS enable bit (ICR: ISE) of the interrupt control register is "0", interrupt processing is performed. When the EI^{2}OS enable bit (ICR: ISE) of the interrupt control register is "1", interrupt processing is performed after the extended intelligent I/O service processing (DTP processing) is executed.

Figure 14.5-2 DTP/External Interrupt Circuit

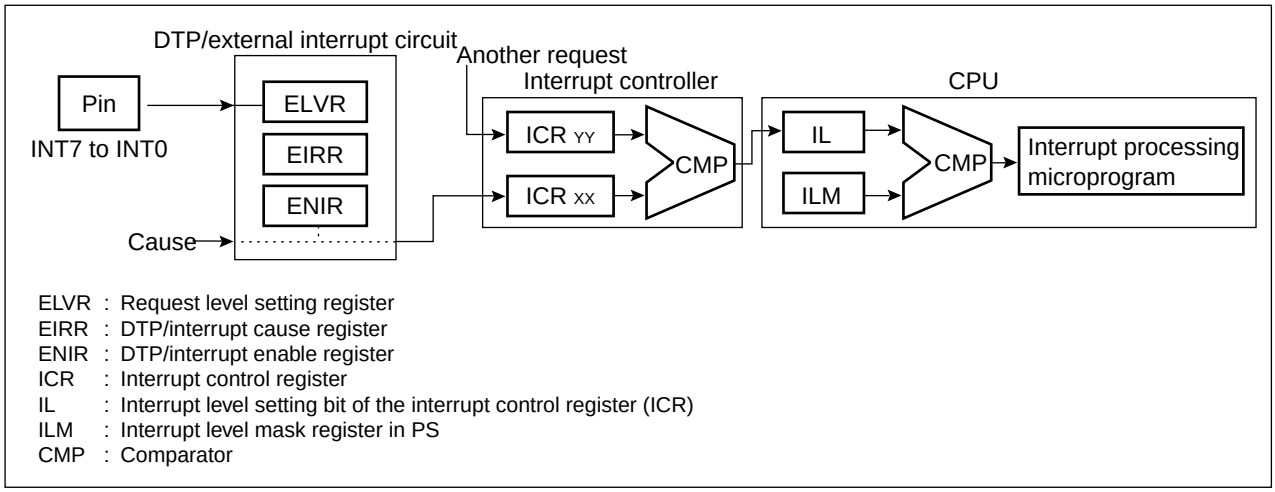
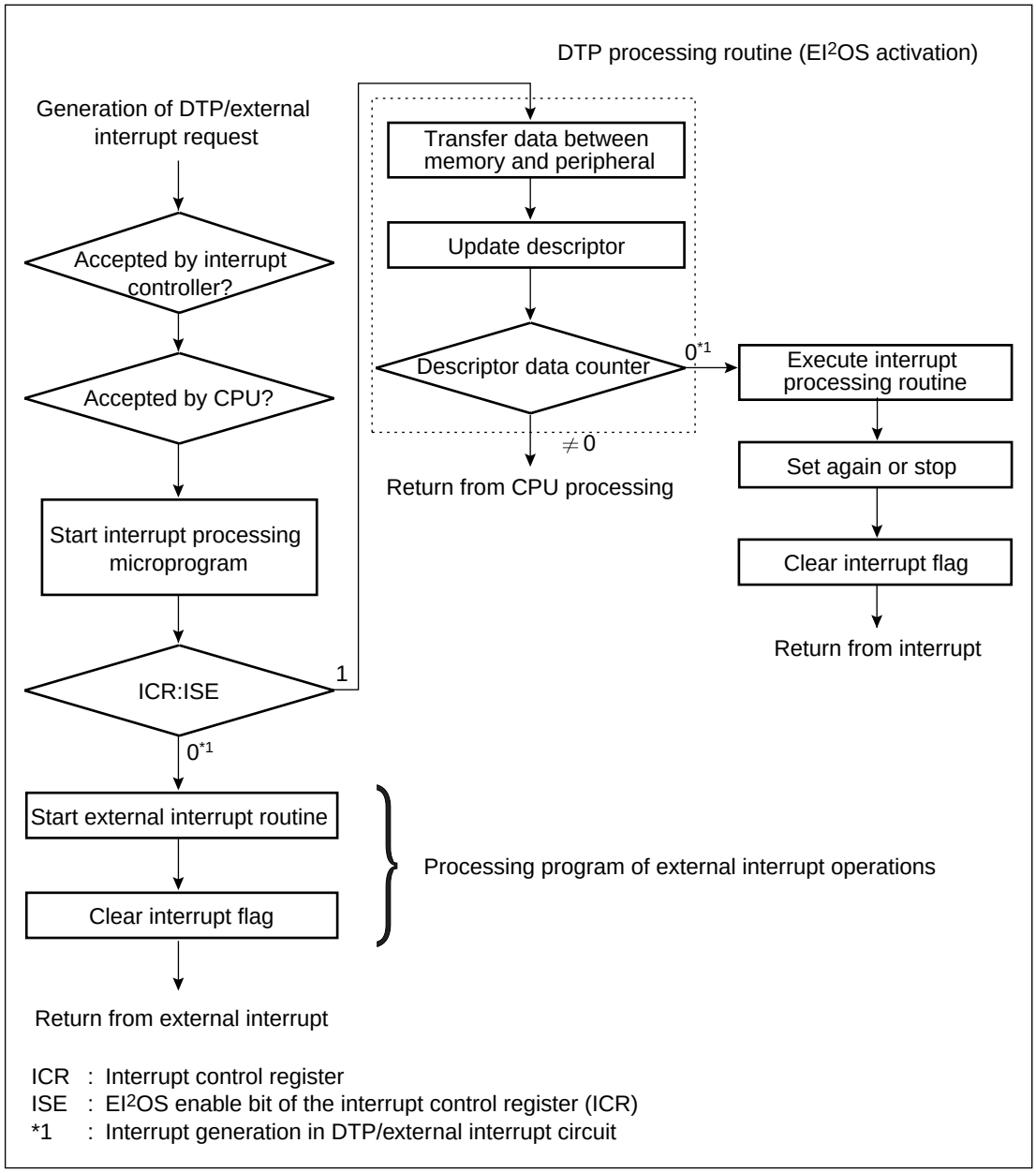


Figure 14.5-3 Flowchart of DTP/External Interrupt Circuit



14.5.1 External Interrupt Function

The DTP/external interrupt circuit has an external interrupt function that outputs an interrupt request when the input signal is input to an external interrupt input pin (INT7 to INT0).

■ External Interrupt Function

When the detection condition (level or edge) set in the request level setting register (ELVR) is input to an external interrupt input pin (INT7 to INT0), the external interrupt request flag bit (ER7 to ER0) corresponding to the pin of the DTP/external interrupt cause register (EIRR) is set to "1". If the external interrupt request enable bit (EN7 to EN0) corresponding to the external interrupt input pin (INT7 to INT0) of the DTP/external enable register (ENIR) is set to "1", an interrupt request is output to the interrupt controller. The interrupt controller determines the interrupt level (ICR: IL2 to IL0) of interrupt requests from peripheral functions (resources) and priorities when interrupt requests are output simultaneously. The CPU determines whether to accept an interrupt request based on the interrupt level mask register (PS: ILM) and interrupt enable flag (PS: CCR: I). When the CPU accepts an interrupt request, it performs interrupt processing before branching to the interrupt processing routine. In the interrupt processing program, set the corresponding external interrupt request flag bit (ER7 to ER0) to "0" and clear the interrupt request before returning from the interrupt using an interrupt return instruction.

Note:

When the interrupt processing program is activated, be sure to set the external interrupt request flag bit (EIRR: EN7 to EN0) that caused the activation to "0". It is not possible to return from an interrupt while the external interrupt request flag bit (EIRR: EN7 to EN0) is set to "1".

14.5.2 DTP Function

The DTP/external interrupt circuit has a DTP (Data Transfer Peripheral) function that detects the data transfer request signal input to an external interrupt input pin (INT7 to INT0) from external peripheral devices and activates the extended intelligent I/O service.

■ Operation of the DTP Function

The DTP function is a function for detecting the data transfer request signal input to an external interrupt input pin (INT7 to INT0) from external peripheral devices to automatically transfer data between memory and the peripheral devices.

The extended intelligent I/O service is activated by the external interrupt function. The operation of the DTP function is the same as that of the external interrupt function until an interrupt request is accepted by the CPU. If the EI^{2}OS operation is enabled (ICR: ISE="1"), EI^{2}OS is activated when an interrupt request is accepted to start data transfer. When the data transfer is completed, the descriptor is updated and the external interrupt request flag bit (EIRR: ER7 to ER0) is cleared to "0" to operate as the external interrupt function again. When the transfer by EI^{2}OS is completed, branching to the interrupt processing routine pointed to by the vector address of the external interrupt occurs. Peripheral devices that are externally connected should remove the cause input of the data transfer request signal (DTP cause) within three machine cycles after the first transfer started.

Figure 14.5-4 Example of Interfacing with External Peripheral Devices (Timing)

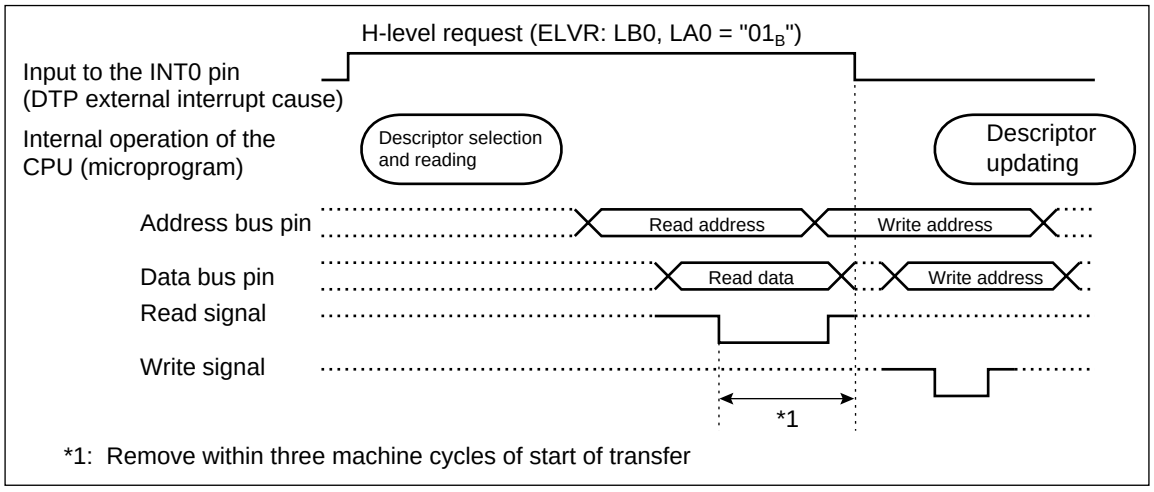
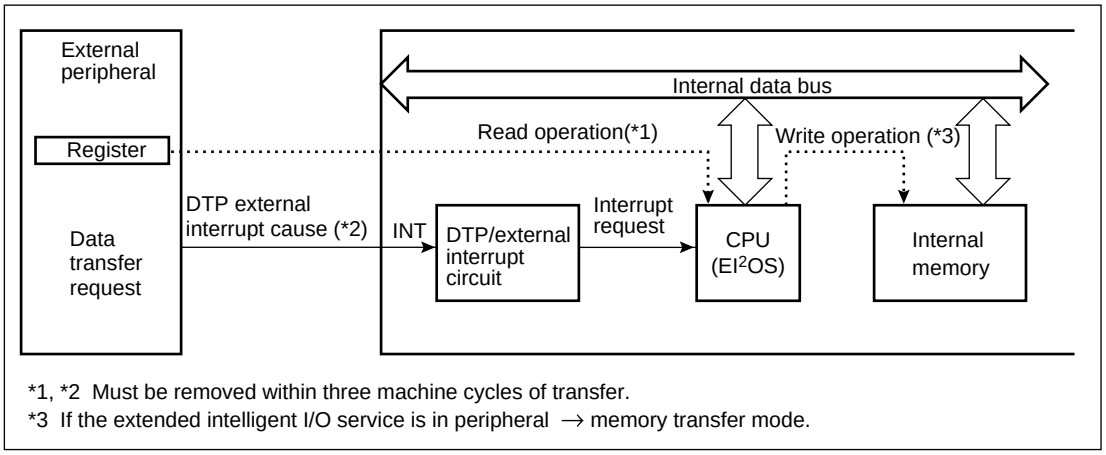


Figure 14.5-5 Example of Interfacing with External Peripheral Devices (Block Diagram)



*1, *2 Must be removed within three machine cycles of transfer.
*3 If the extended intelligent I/O service is in peripheral → memory transfer mode.

14.6 Usage Notes on the DTP/External Interrupt Circuit

Notes on the signal to be input to the DTP/external interrupt circuit, release from standby mode, and interrupts are given below.

■ Usage Notes on the DTP/External Interrupt Circuit

❍ Conditions for external peripherals using the DTP function

To support the DTP function, peripheral devices that are externally connected must be able to automatically clear data transfer requests after transfer is carried out. If an externally connected peripheral device continues to output a transfer request longer than three machine cycles after the CPU started the transfer operation, the DTP/external interrupt circuit interprets the request as another transfer request and performs the data transfer operation again.

❍ Input polarities of external interrupts

- If the request level setting register (ELVR) is set for edge detection, the pulse width of at least three machine cycles is required from the point of change of the input level to detect the input of an edge that is to become an interrupt request.
- If the request level setting register (ELVR) is set for level detection, and the level for interrupt request is input, the cause flip-flop in the DTP/interrupt cause register (EIRR) is set to "1" and retains the cause, as shown in Figure 14.6-1 "Clearing the Cause Retention Circuit When a Level is Specified". Thus, even if the interrupt cause is removed, the request to the interrupt controller remains active. To cancel the request to the interrupt controller, set the external interrupt request flag bits (EIRR: ER7 to ER0) to "0" to clear the cause flip-flop to "0", as shown in Figure 14.6-1 "Clearing the Cause Retention Circuit When a Level is Specified".

Figure 14.6-1 Clearing the Cause Retention Circuit When a Level is Specified

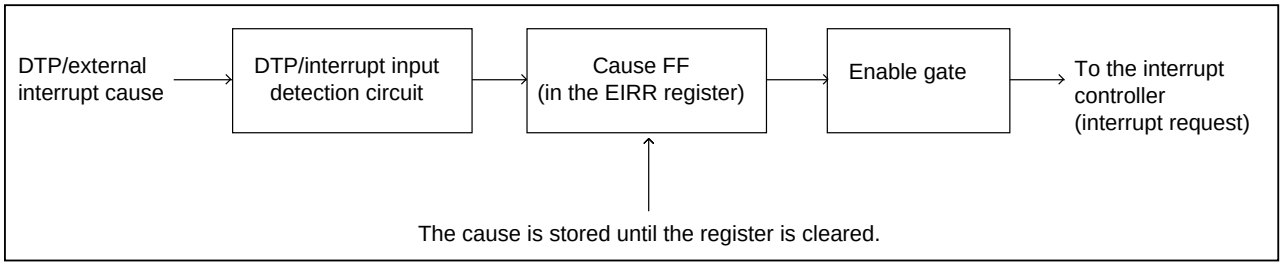
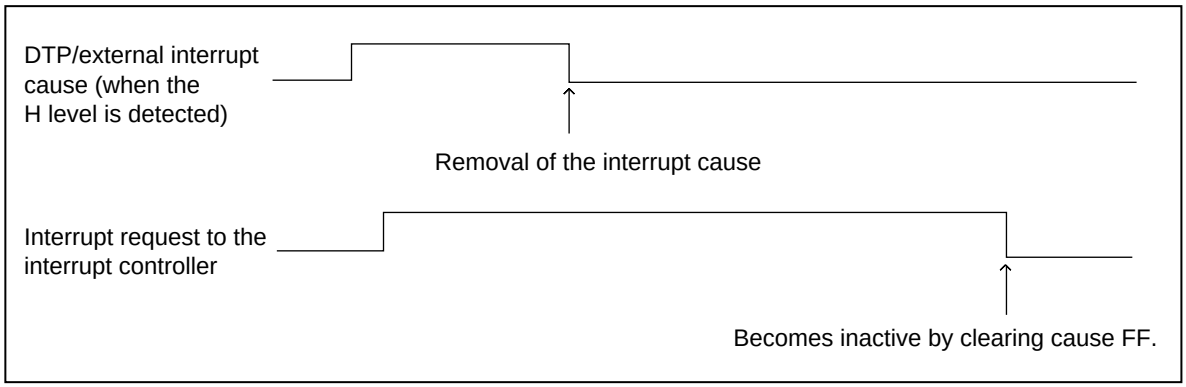


Figure 14.6-2 DTP/External Interrupt Cause and Interrupt Request When the Output of Interrupt Requests is Enbaled



❍ **Notes about interrupts**

When the external interrupt function is used to branch to an interrupt processing routine, it is not possible to return from the interrupt processing program if an external interrupt request flag bit (EIRR: ER7 to ER0) is "1" and an external interrupt request enable bit (ENIR: EN7 to EN0) is "1". Be sure to clear the external interrupt request flag bits (EIRR: ER7 to ER0) to "0" in the processing program. (When using the DTP function, the external interrupt request flag bits (EIRR: ER7 to ER0) are cleared to "0" by EI^2OS.)

If the register is set for level detection, it is not possible to return from the interrupt processing program when the level signal of an interrupt request is input to an external interrupt input pin (INT7 to INT0) because, even if an external interrupt request flag bit (EIRR: ER7 to ER0) is set to "0", it is set to "1" again. Thus, disable an interrupt request or remove the level signal for an interrupt request.

14.7 Sample Programs for the DTP/External Interrupt Circuit

This section contains sample programs for the external interrupt function and the DTP function.

■ Sample Program for the External Interrupt Function

❍ Processing

The rising edge of the pulse input to the INT0 pin is detected, and an external interrupt is generate.

❍ Coding example

```
ICR07   EQU    0000B7H      ;Interrupt control register for the DTP/
                             external interrupt circuit
DDR1  EQU     000011H       ;Port 1 direction register
ENIR  EQU     000030H       ;DTP/interrupt enable register
EIRR  EQU     000031H       ;DTP/interrupt cause register
ELVRL EQU     000032H       ;Request level setting register
ELVRH EQU     000033H       ;Request level setting register
ER0   EQU     EIRR:0        ;INT0 interrupt flag bit
EN0   EQU     ENIR:0        ;INT0 interrupt enable bit
;-------Main program-----------------------------------------------
DE    CSEG
START:
;      :                    ;Assumes that stack pointer (SP) has
                             already been initialized.
      MOV  I:DDR1,#00000000B;Sets DDR1 as an input port.
      AND  CCR,#0BFH        ;Disables interrupts.
      MOV  I:ICR07,#00H     ;Interrupt level:0 (highest).
                             Disables EI2OS.
      CLRB EN0              ;Disables INT0, using ENIR.
      MOV  I:ELVR,#00000010B;Selects the rising edge for INT0.
      CLRB I:ER0            ;Clears the cause for INT0 using EIRR.
      SETB I:EN0            ;Enables INT0 using ENIR.
      MOV  ILM,#07H         ;Sets ILM in PS to level 7.
      OR   CCR,#40H         ;Enables interrupts.
LOOP: MOV  A,#00H           ;Endless loop
      MOV  A,#01H
      BRA  LOOP
;-------Interrupt program------------------------------------------
WARI
      CLRB ER0              ;Clears the interrupt request flag.
;      :
;     User processing
;      :
      RETI                  ;Returns from interrupt.
CODE  ENDS
;-------Vector setting---------------------------------------------
```

```
VECT  CSEG ABS=0FFH
      ORG  0FF98H           ;Sets vector for interrupt #25 (19H).
      DSL  WARI
      ORG  0FFDCH           ;Sets reset vector.
      DSL  START
      DB   00H              ;Sets single-chip mode.
VECT  ENDS
      END  START
```

■ Sample Program for the DTP Function

❍ Processing

- The H level of the signal input to the INT0 pin is detected, and channel 0 of the extended intelligent I/O service (EI^{2}OS) is activated.
- Data is output from RAM to port 0 by DTP processing (EI^{2}OS).

❍ Coding example

```
ICR07 EQU  0000B7H          ;Interrupt control register for the DTP/
                             external interrupt circuit
DDR0  EQU  000010H          ;Port 0 direction register
DDR1  EQU  000011H          ;Port 1 direction register
ENIR  EQU  000030H          ;DTP/interrupt enable register
EIRR  EQU  000031H          ;DTP/interrupt cause register
ELVRL EQU  000032H          ;Request level setting register
ELVRH EQU  000033H          ;Request level setting register
ER0   EQU  EIRR:0           ;INT0 interrupt flag bit
EN0   EQU  ENIR:0           ;INT0 interrupt enable bit
BAPL  EQU  000100H          ;Buffer address pointer, lower
BAPM  EQU  000101H          ;Buffer address pointer, middle
BAPH  EQU  000102H          ;Buffer address pointer, upper
ISCS  EQU  000103H          ;EI  OS status register
IOAL  EQU  000104H          ;I/O address register, lower
IOAH  EQU  000105H          ;I/O address register, upper
DCTL  EQU  000106H          ;Data counter, lower
DCTH  EQU  000107H          ;Data counter, upper
;-------Main program-----------------------------------------------
CODE  CSEG
START:
;      :                    ;Assumes that stack pointer (SP) has
                             already been initialized.
      MOV  I:DDR0,#11111111B;Sets DDR0 as an output port.
      MOV  I:DDR5,#00000000B;Sets DDR1 as an input port.
      AND  CCR,#0BFH        ;Disables interrupts.
      MOV  I:ICR07,#08H     ;Interrupt level:0 (highest)
                            ;Enables EI2OS.  Channel 0
      MOV  BAPL,#00H        ;Sets the address of the output data
      MOV  BAPM,#06H        ;
      MOV  BAPH,#00H        ;
      MOV  ISCS,#12H        ;Byte transfer. I/O address fixed.
                             Buffer address + 1.
                             Transfer from memory to I/O
      MOV  IOAL,#00H       ;Address pointer specifies port0(PDR0)as
      MOV  IOAH,#00H         the transfer destination.
      MOV  DCTL,#0AH        ;Number of transfers:  10
      MOV  DCTH,#00H        ;
      CLRB I:EN0            ;Disables INT0 using ENIR.
      MOV  I:ELVR,#00000001B;Selects H level for INT0.
      CLRB I:ER0            ;Clears the cause of INT0 using EIRR.
      SETB I:EN0            ;Enables INT0 using ENIR.
      MOV  ILM,#07H         ;Sets ILM in PS to level 7.
      OR   CCR,#40H         ;Enables interrupts.
LOOP: MOV  A,#00H           ;Endless loop
```

```
      MOV  A,#01H           ;
      BRA  LOOP             ;
;-------Interrupt program-----------------------------------------
WARI:
      CLRB I:ER0            ;Clears the interrupt request flag.
;      :                    ;Switches the channel and changes the
                             transfer address, if required.
;     User processing       ;Specifies processing again, such as the
                             termination of EI2OS.  To terminate the
                             processing, interrupts must be
                             disabled.
;      :
      RETI                  ;Returns from the interrupt.
CODE  ENDS
;-------Vector setting--------------------------------------------
VECT  CSEG ABS=0FFH
      ORG  0FF98H           ;Sets vector for interrupt #25 (19H).
      DSL  WARI
      ORG  0FFDCH           ;Sets reset vector.
      DSL  START
      DB   00H              ;Sets single-chip mode.
VECT  ENDS
      END  START
```

CHAPTER 15 DELAYED INTERRUPT GEMERATOR MODULE

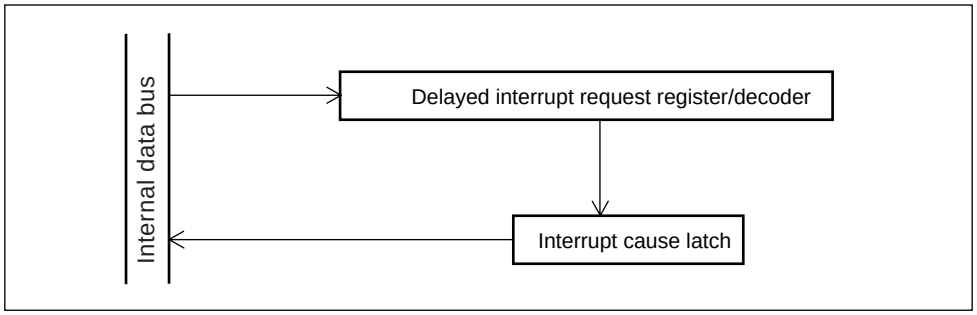
This chapter describes the functions and operation of the MB90560/565 series delayed interrupt generator module.

15.1 Overview of the Delayed Interrupt Generator Module

The delayed interrupt generator module outputs interrupt requests for task switching. By using this module, interrupt requests for task switching can be output and canceled for the MB90560/565 series CPU via software.

■ Block Diagram of the Delayed Interrupt Generator Module

Figure 15.1-1 Block Diagram of the Delayed Interrupt Generator Module



15.2 Delayed Interrupt Cause/Cancel Register (DIRR)

This section explains the delayed interrupt cause/cancel register (DIRR).

■ Delayed Interrupt Cause/Cancel Register (DIRR)

Figure 15.2-1 Delayed Interrupt Cause/Cancel Register (DIRR)

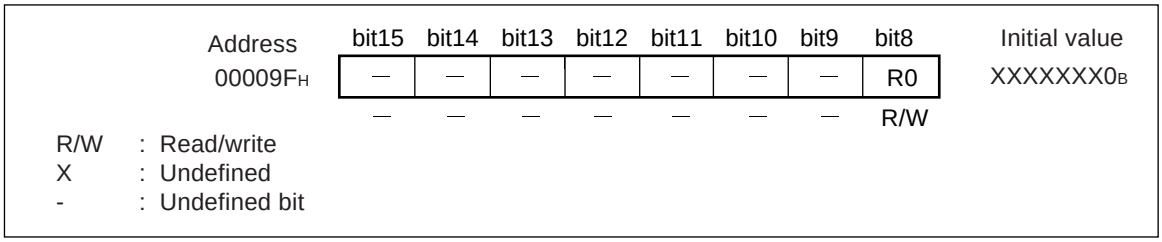


Table 15.2-1 Functional Explanation of Each Bit of the Delayed Interrupt Cause/Cancel Register (DIRR)

Bit name		Function
bit15 to bit9	-: Undefined bit	• When these bits are read, the values are undefined. • Writing to these bits does not affect operation.
bit8	R0: Delayed interrupt request output bit	• This bit sets the generation/cancel of a delayed interrupt request. • When this bit is "1", a delayed interrupt request is output. • When this bit is "0", the delayed interrupt request is cleared. • When a reset is specified, interrupt causes are canceled (cleared to "0").

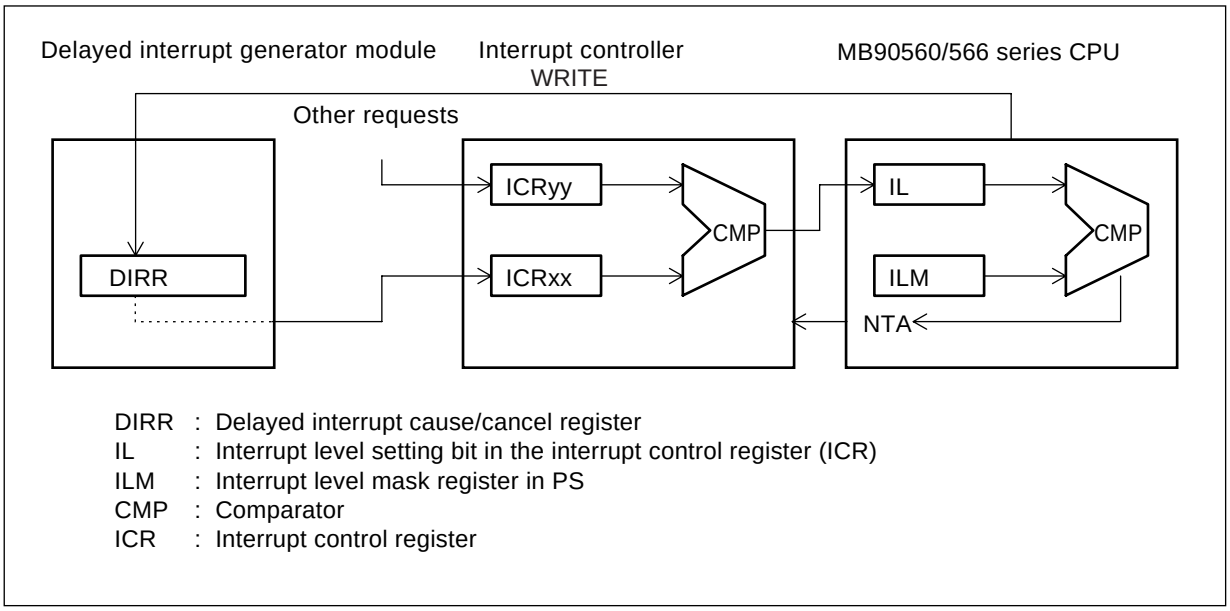
15.3 Operation of the Delayed Interrupt Generator Module

When the delayed interrupt request output bit (R0) of the delayed interrupt cause/ cancel register (DIRR) is set to "1" using software, a delayed interrupt request is output to the interrupt controller.

■ Operation of the Delayed Interrupt Generator Module

When the delayed interrupt request output bit (R0) of the delayed interrupt cause/cancel register (DIRR) is set to "1" using software, an interrupt request is output to the interrupt controller. If interrupt requests other than the delayed interrupt have lower priorities or there is no interrupt request other than the delayed interrupt request, the interrupt controller outputs an interrupt request to the CPU. The CPU compares the interrupt request level with the interrupt level mask register (ILM) in the processor status register (PS). If the interrupt request level is higher than the interrupt level mask register (ILM), the hardware imbedded processing microprogram is activated after the instruction currently being executed is completed, to execute the delayed interrupt processing routine. If the delayed interrupt request output bit (R0) of the delayed interrupt cause/cancel register (DIRR) is set to "0" in the interrupt processing routine, the delayed interrupt cause is cleared and the task is switched.

Figure 15.3-1 Operation of the Delayed Interrupt Generator Module



DIRR : Delayed interrupt cause/cancel register
IL : Interrupt level setting bit in the interrupt control register (ICR)
ILM : Interrupt level mask register in PS
CMP : Comparator
ICR : Interrupt control register

15.4 Precautions to Follow when Using the Delayed Interrupt Generator Module

This section explains the precautions to follow when using the delayed interrupt generator module.

■ Precautions to Follow when Using the Delayed Interrupt Generator Module

❍ Delayed interrupt request

If the delayed interrupt request output bit (R0) of the delayed interrupt cause/cancel register (DIRR) is not set to "0" after interrupt processing is completed using an interrupt processing routine or while an interrupt processing routine is being executed, it is not possible to return from the interrupt processing.

CHAPTER 16 8/10-BIT A/D CONVERTER

This chapter describes the functions and operation of the 8/10-bit A/D converter.

16.1 Overview of the 8/10-Bit A/D Converter

The 8/10-bit A/D converter has a function for converting the analog input voltage into a 10-bit or 8-bit value using the RC-type successive approximation conversion method.

■ Functions of the 8/10-Bit A/D Converter

The following shows the functions of the 8/10-bit A/D converter:

- The minimum conversion time is 6.13 μs (for a machine clock of 16 MHz; includes the sampling time).
- The minimum sampling time is 2.0 μs (for a machine clock of 16 MHz).
- The converter uses the RC-type successive approximation conversion method with a sample hold circuit.
- A resolution of 10 bits or 8 bits can be selected.
- The input signal can be set using a program from the 8-channel analog input pins.
- At the end of A/D conversion, an interrupt request can be generated and EI^{2}OS can be activated.
- If A/D conversion is performed while an interrupt is enabled, the conversion data protection function is activated.
- The conversion can be activated by software, 16-bit reload timer 1 output (rising edge), and 16-bit free-running timer zero detection edge.

Table 16.1-1 "8/10-bit A/D Converter Conversion Modes" lists four types of conversion modes.

Table 16.1-1 8/10-bit A/D Converter Conversion Modes

Conversion mode	Single conversion	Scan conversion
Single conversion mode 1 Single conversion mode 2	Converts the input of a specified channel (single channel) just once.	Converts the inputs of two or more consecutive channels (up to eight channels) just once.
Continuous conversion mode	Converts the input of a specified channel (single channel) repeatedly.	Converts the inputs of two or more consecutive channels (up to eight channels) repeatedly.
Stop conversion mode	Converts the input of a specified channel (single channel), after which it is on standby for the next activation.	Converts the inputs of two or more consecutive channels (up to eight channels) just once, after which it is on standby for the next activation.

■ 8/10-Bit A/D Converter Interrupts and EI^{2}OS

Table 16.1-2 8/10-Bit A/D Converter Interrupts and EI^{2}OS

Interrupt No.	Interrupt control register		Vector table address			EI^{2}OS
	Register name	Address	Lower	Upper	Bank	
#11 ($0B_H$)	ICR00	$0000B0_H$	$FFFFD0_H$	$FFFFD1_H$	$FFFFD2_H$	O

O: Available

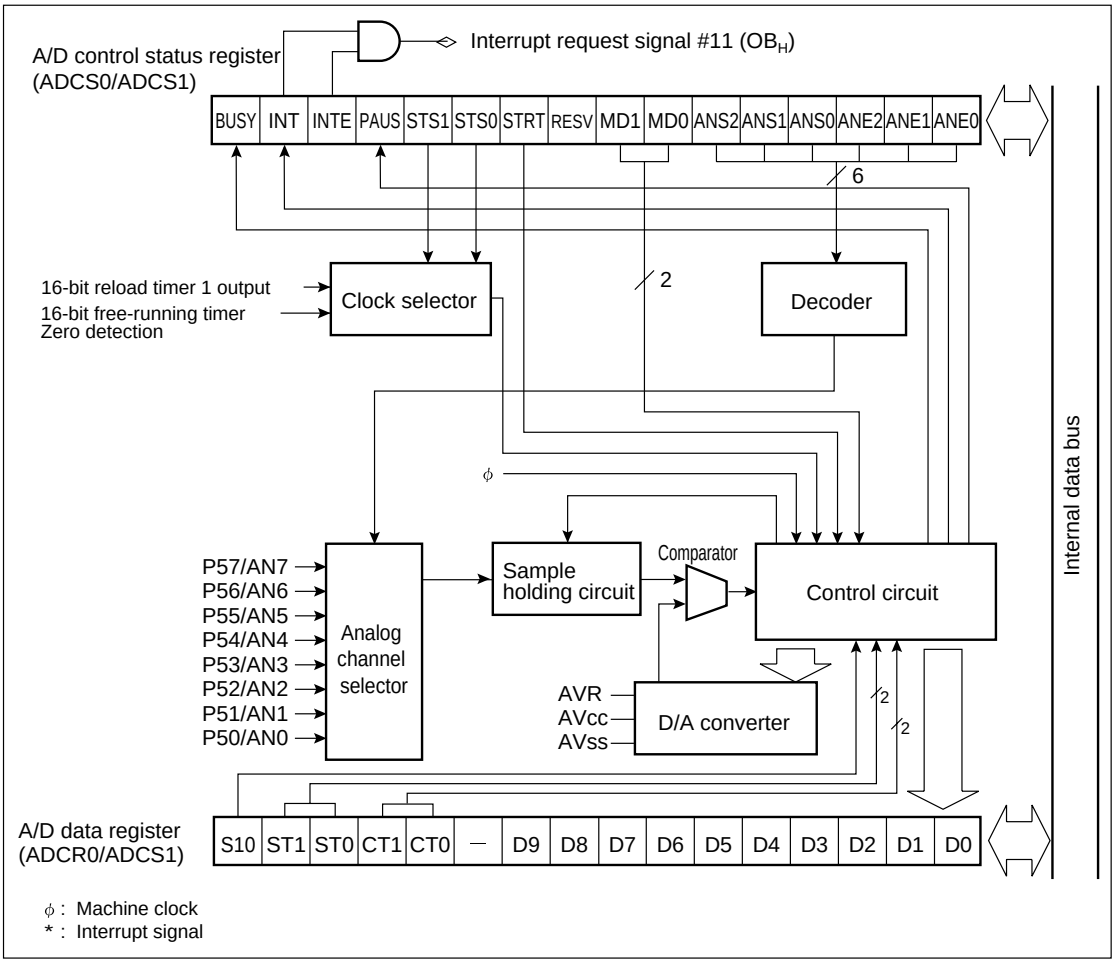
16.2 Configuration of the 8/10-Bit A/D Converter

The 8/10-bit A/D converter has nine blocks:

- **A/D control status register (ADCS0/ADCS1)**
- **A/D data register (ADCR0/ADCR1)**
- **Clock selector (Input clock selector for activating A/D conversion)**
- **Decoder**
- **Analog channel selector**
- **Sample hold circuit**
- **D/A converter**
- **Comparator**
- **Control circuit**

■ Block Diagram of the 8/10-Bit A/D Converter

Figure 16.2-1 Block Diagram of the 8/10-Bit A/D Converter



- **A/D control status register (ADCS0/ADCS1)**

  The A/D control status register (ADCS0) sets A/D conversion mode and the A/D conversion start/end channel.

  The A/D control status register (ADCS1) sets the A/D conversion activation trigger and enable/disable of interrupt requests and checks the interrupt request status and whether the A/D conversion has halted/is in progress.

- **A/D data register (ADCR0/ADCR1)**

  This register stores the results of A/D conversion. It also sets the resolution for A/D conversion, sampling time during A/D conversion, and compare time during A/D conversion.

- **Clock selector**

  The clock selector selects the clock for activating A/D conversion. Either 16-bit reload timer channel 1 output or 16-bit free-running timer zero detection can be used as the activation clock.

- **Decoder**

  This circuit sets the analog input pin to be used based on the A/D conversion end channel setting bits (ANE0 to ANE2) and A/D conversion start channel setting bits (ANS0 to ANS2) of the A/D control status register (ADCS0).

- **Analog channel selector**

  This circuit selects the pin to be used from eight analog input pins.

- **Sample hold circuit**

  This circuit maintains the input voltage from the pin set by the analog channel selector. By maintaining the input voltage just after starting A/D conversion, it is not affected by input voltage variations during A/D conversion (during comparison).

- **D/A converter**

  This circuit generates a reference voltage for comparison with the input voltage maintained by the sample hold circuit.

- **Comparator**

  This circuit compares the input voltage maintained by the sample hold circuit with the output voltage of the D/A converter to determine which is greater.

- **Control circuit**

  This circuit determines the A/D conversion value based on the decision signal generated by the comparator. When the A/D conversion has been completed, the circuit sets the conversion result in the A/D data register (ADCR0/ADCR1) and generates an interrupt request.

16.3 8/10-Bit A/D Converter Pins

This section describes the 8/10-bit A/D converter pins and provides pin block diagrams.

■ 8/10-Bit A/D Converter Pins

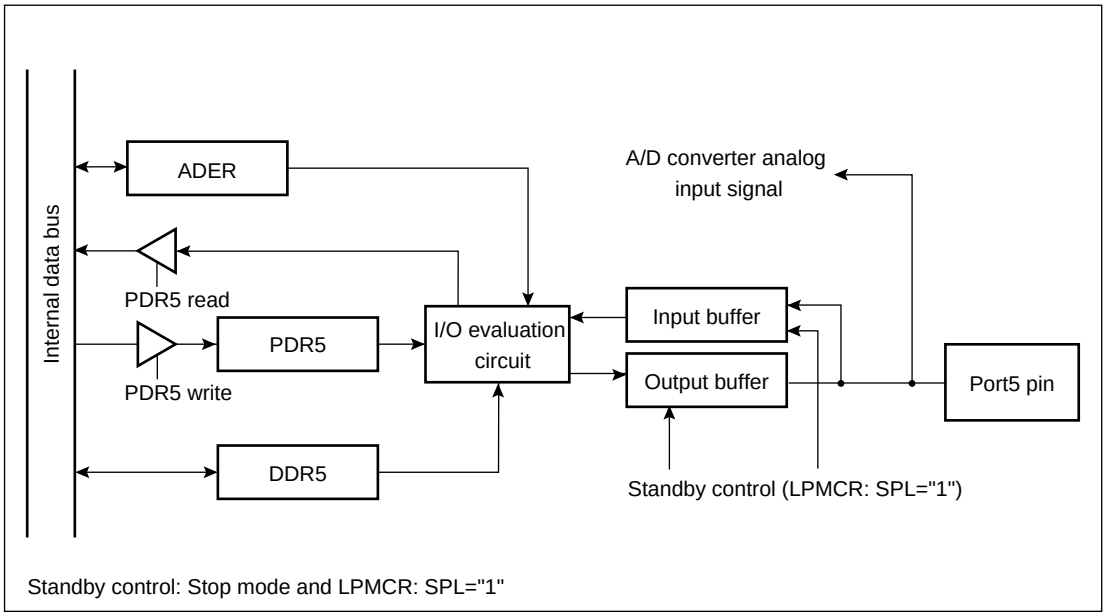
The A/D converter pins are also used as I/O ports.

Table 16.3-1 8/10-bit A/D Converter Pins

Function	Pin name	Pin function	Input-output signal type	Pull-up option	Standby control	Setting for using the pin
Channel 0	P50/AN0	Port 5 input-output or analog input	CMOS output/ CMOS hysteresis input or analog input	Not selectable	Not selectable	Set as an input port (DDR5: bit15 to bit8="0") Set as an analog port (ADER: bit15 to bit8="1")
Channel 1	P51/AN1					
Channel 2	P52/AN2					
Channel 3	P53/AN3					
Channel 4	P54/AN4					
Channel 5	P55/AN5					
Channel 6	P56/AN6					
Channel 7	P57/AN7					

■ Block Diagrams of the 8/10-Bit A/D Converter Pins

Figure 16.3-1 Block Diagram of the P50/AN0 to P57/AN7 Pins



Note:

- To use a pin as an input port, set the corresponding bit (bit15 to bit8) of the port direction register (DDR5) to "0" and the corresponding bit (bit15 to bit8) of the analog input enable register (ADER) to "0".
- To use a pin as an analog input pin, set the corresponding bit (bit15 to bit8) of the analog input enable register (ADER) to "1". The value read from the port data register (PDR5) is "00_H".

16.4 8/10-Bit A/D Converter Registers

This section lists the 8/10-bit A/D converter registers.

■ 8/10-Bit A/D Converter Registers

Figure 16.4-1 List of Registers of the 8/10-Bit A/D Converter

Address	bit15 ····· bit8	bit7 ····· bit0
000017_H	Analog input enable register (ADER)	
000035_H, 000034_H	A/D control status register(ADCS1)	A/D control status register (ADCS0)
000037_H, 000036_H	A/D data register(ADCR1)	A/D data register (ADCR0)

16.4.1 A/D Control Status Register 1 (ADCS1)

The A/D control status register (ADCS1) sets the A/D conversion activation trigger and enable/disable of interrupt requests and checks the interrupt request status and whether the A/D conversion has halted/is in progress.

■ Upper Bits of the A/D Control Status Register 1 (ADCS1)

Figure 16.4-2 A/D Control Status Register 1 (ADCS1)

Address: 000035H

bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
BUSY	INT	INTE	PAUS	STS1	STS0	STRT	RESV
R/W	R/W	R/W	R/W	R/W	R/W	W	R/W

Initial value: 00000000B

RESV	Reserved bit
Always write 0 to this bit.	

STRT	A/D conversion activation bit (valid only when activated by software (ADC2: EXT= 0))
0	Does not activate the A/D conversion.
1	Activate the A/D conversion function.

STS1	STS0	A/D activation select bit
0	0	Activation by software.
0	1	Activation by Zero detection of 16-bit free-running timer or activation of software
1	0	Activation by 16-bit reload timer1 or activation of software
1	1	Activation by Zero detection of 16-bit free-running timer, 16-bit reload timer1, or activation of software

PAUS	Halt flag bit (valid only when EI^{2}OS is used)
0	A/D conversion is active
1	A/D conversion is halted.

INTE	Interrupt request enable bit
0	Disables interrupt request output.
1	A/D conversion is halted.

INT	Interrupt request flag bit	
	Reading	Writing
0	A/D conversion has not been completed.	Clears interrupt request
1	A/D conversion has been completed.	No effect on operation

BUSY	Busy bit	
	Reading	Writing
0	A/D conversion is halted.	Stops the A/D conversion.
1	A/D conversion is in progress.	No change,no effect on other bits.

R/W : Read/write
W : Write only
□ : Initial value

Table 16.4-1 Function Description of Each Bit of A/D Control Status Register 1 (ADCS1)

Bit name		Function
bit15	BUSY: Busy bit	• This bit indicates the operating status of the A/D converter • When this bit is "0", the A/D conversion has halted. • When this bit is "1", the A/D conversion is in progress. • When this bit is set to "0", the A/D conversion is forced to halt. • When this bit is set to "1", operation is not affected. **Note:** Do not set the forced A/D conversion stop and the activation (BUSY="0", STRT="1") simultaneously.
bit14	INT: Interrupt request flag bit	• This bit is a flag that requests an interrupt. • This bit is set to "1" when A/D conversion results are stored in the A/D data register (ADCR0/ADCR1). • When this bit is set to "1" while the interrupt request enable bit (INTE) is "1", an interrupt request is output. • When this bit is set to "0", the interrupt request is cleared. • When this bit is set to "1", operation is not affected. • When EI^{2}OS is used, this bit is cleared to "0". **Note:** To clear the interrupt requests, stop the A/D conversion.
bit13	INTE: Interrupt request enable bit	• This bit enables an interrupt request. • When the interrupt request flag bit (INT) is set to "1" while this bit is set to "1", an interrupt request is output. • To use EI^{2}OS, set this bit to "1".
bit12	PAUS: Halt flag bit	• This bit is set to "1" when the A/D conversion stops temporarily. • When EI^{2}OS is used in continuous conversion mode, this bit is set to "1" if transfer of the last piece of data to memory has not been completed even though A/D conversion is completed. Thus, the A/D conversion is stopped temporarily and conversion data is not stored in the A/D data register (ADCR0/ ADCR1). • When data transfer of the last piece of data to memory is completed, this bit is cleared to "0" and the A/D conversion is then restarted. **Note:** This bit is valid when EI^{2}OS is used.
bit11 bit10	STS1, STS0: A/D activation select bit	• These bits select how A/D conversion is to be activated. • When two or more activation causes are shared, activation is the result of the cause that occurs first. **Note:** Change the setting during A/D conversion only while there is no corresponding activation cause, since the change becomes effective immediately.
bit9	STRT: A/D conversion activation bit	• This bit allows software to start A/D conversion. • Writing 1 to this bit activates A/D conversion. • In stop conversion mode, conversion cannot be reactivated with this bit. • The value read from this bit is "1". **Note:** Never perform the forced stop and activation (BUSY="0", STRT="1") of the A/D conversion simultaneously.
bit8	RESV: Reserved bit	**Note:** Always write 0 to this bit.

16.4.2 A/D Control Status Register 0 (ADCS0)

The A/D control status register (ADCS0) sets A/D conversion mode and the A/D conversion start/end channel.

■ A/D Control Status Register 0 (ADCS0)

Figure 16.4-3 A/D Control Status Register 0 (ADCS0)

Address: 000034H

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
MD1	MD0	ANS2	ANS1	ANS0	ANE2	ANE1	ANE0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Initial value: 00000000B

ANE2	ANE1	ANE0	A/D conversion end channel setting bit
0	0	0	AN0 pin
0	0	1	AN1 pin
0	1	0	AN2 pin
0	1	1	AN3 pin
1	0	0	AN4 pin
1	0	1	AN5 pin
1	1	0	AN6 pin
1	1	1	AN7 pin

ANS2	ANS1	ANS0	A/D conversion start channel setting bit		
			Halt	Read during conversion	Read during a pause in stop conversion mode
0	0	0	AN0 pin	Number of the current conversion channel	Number of the last conversion channel
0	0	1	AN1 pin		
0	1	0	AN2 pin		
0	1	1	AN3 pin		
1	0	0	AN4 pin		
1	0	1	AN5 pin		
1	1	0	AN6 pin		
1	1	1	AN7 pin		

MD1	MD0	A/D conversion mode setting bit
0	0	Single conversion mode 1 (reactivation allowed during operation)
0	1	Single conversion mode 2 (reactivation not allowed during operation)
1	0	Continuous conversion mode (reactivation not allowed during operation)
1	1	Stop conversion mode (reactivation not allowed during operation)

R/W : Read/write
□ : Initial value

Table 16.4-2 Function Description of Each Bit of A/D Control Status Register 0 (ADCS0)

Bit name		Function
bit7 bit6	MD1, MD0: A/D conversion mode setting bit	• This bit is used to set the A/D conversion mode. • Single conversion mode 1, single conversion mode 2, continuous conversion mode , and stop conversion mode can be set. **Single conversion mode 1:** A/D conversion is performed from the channel specified by the A/D conversion start channel setting bits (ANS2 to ANS0) to that specified by the A/D conversion end channel setting bits (ANE2 to ANE0) before terminating. Reactivation during operation is allowed. **Single conversion mode 2:** A/D conversion is performed from the channel specified by the A/D conversion start channel setting bits (ANS2 to ANS0) to that specified by the A/D conversion end channel setting bits (ANE2 to ANE0) before terminating. Reactivation during operation is not allowed. **Continuous conversion mode:** A/D conversion is performed from the channel specified by the A/D conversion start channel setting bits (ANS2 to ANS0) to that specified by the A/D conversion end channel setting bits (ANE2 to ANE0) is repeated until the conversion stop is forcibly implemented by the Converting bit (BUSY). Reactivation during operation is not allowed. **Stop conversion mode:** A/D conversion is performed from the channel specified by the A/D conversion start channel setting bits (ANS2 to ANS0) to that specified by the A/D conversion end channel setting bits (ANE2 to ANE0) is repeated by making a pause for each channel until the conversion stop is forcibly implemented by the Converting bit (BUSY). Reactivation during operation is not allowed. Reactivation during the pause depends on the activation trigger set in the A/D activation trigger setting bits (STS1, STS0). **Note:** The impossibility of reactivation of each conversion mode (simple, continuous, and stop) is applied to the 16-bit free-running timer 0 detection, 16-bit reload timer1, and activation of all software.
bit5 bit4 bit3	ANS2, ANS1, ANS0: A/D conversion start channel setting bit	• These bits are used to set the A/D conversion start channel and check the channel number being converted. • When the A/D conversion is activated, it is started from the channel specified by the A/D conversion start channel setting bits (ANS2 to ANS0). • During A/D conversion, the channel number being converted is read out. During a pause in stop conversion mode, the channel number converted just before is read out.

Table 16.4-2 Function Description of Each Bit of A/D Control Status Register 0 (ADCS0) (Continued)

Bit name		Function
bit2 bit1 bit0	ANE2, ANE1, ANE0: A/D conversion end channel setting bit	• This bit is used to set the A/D conversion end channel. • When the A/D conversion is activated, it is performed up to the channel specified by the A/D conversion end channel setting bits (ANE2 to ANE0). • If the same channel as that specified by the A/D conversion start channel setting bits (ANS2 to ANS0) is specified, the specified channel is A/D converted. • When continuous conversion mode or stop conversion mode is set and A/D conversion up to the channel specified by the A/D conversion end channel setting bits (ANE2 to ANE0) is completed, the A/D conversion is repeated by returning to the channel specified by the A/D conversion start channel setting bits (ANS2 to ANS0). If the value specified as the start channel is larger than that specified as the end channel, A/D conversion is performed from the start channel to AN7 and then from AN0 to the end channel to complete the 1st conversion.

16.4.3 A/D Data Register (ADCR0/ADCR1)

The A/D data register (ADCR0/ADCR1) stores the results of A/D conversion. It also sets the resolution for A/D conversion, sampling time during A/D conversion, and compare time during A/D conversion.

■ A/D Data Register (ADCR0/ADCR1)

Figure 16.4-4 A/D Data Register (ADCR0/ADCR1)

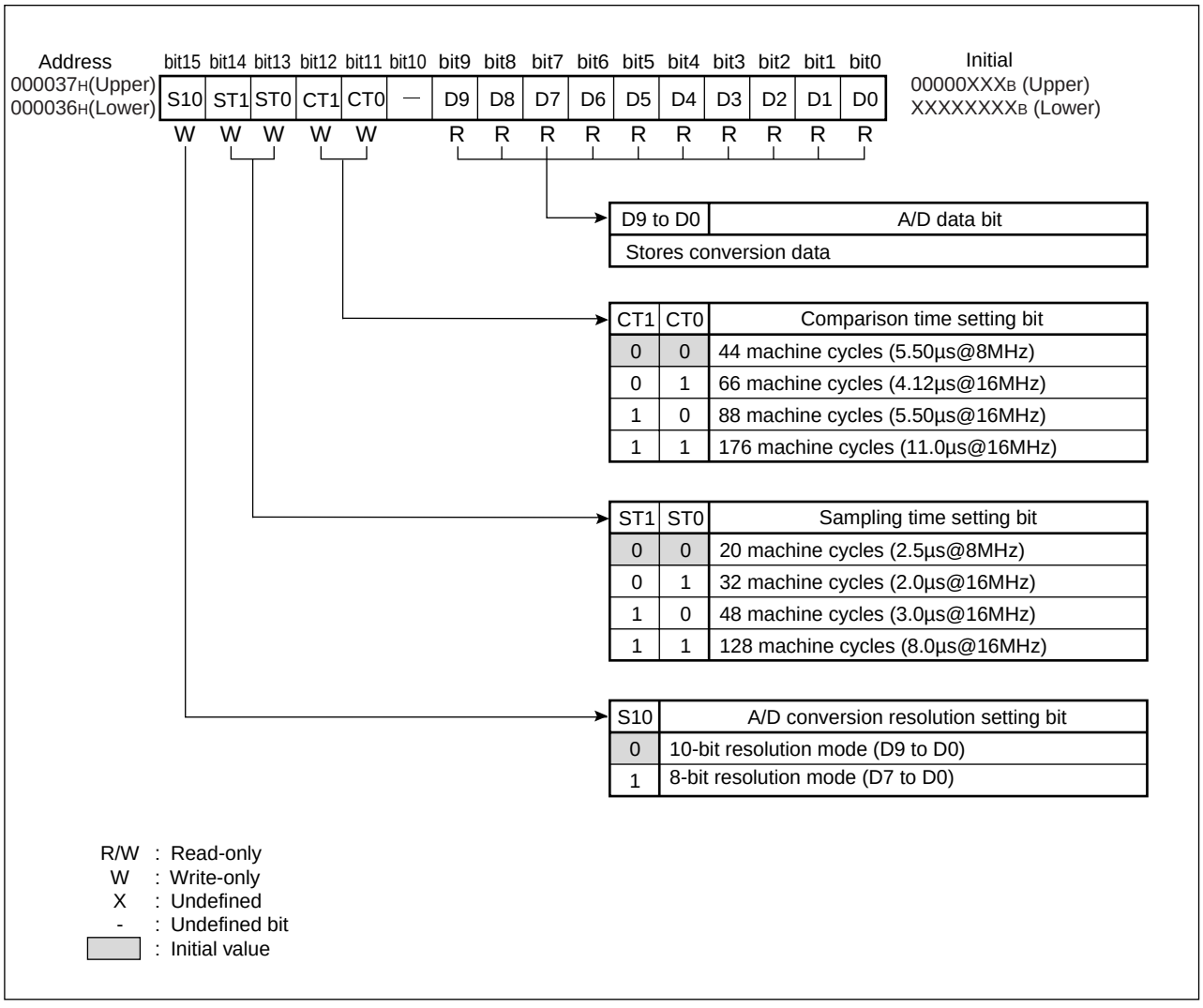


Table 16.4-3 Function Description of Each Bit of A/D Control Status Register (ADCR0/ADCR1)

Bit name		Function
bit15	S10: A/D conversion resolution setting bit	• This bit is used to set the resolution for A/D conversion. • When this bit is "0", the 10-bit resolution is set. • When this bit is "1", the 8-bit resolution is set. **Note:** A/D data bits to be used depend on the resolution. In 10-bit resolution mode, the D9 to D0 bits are used. In 8-bit resolution mode, the D7 to D0 bits are used.
bit14 bit13	ST1, ST0: Sampling time setting bit	• This bit is used to set the sampling time for A/D conversion. • When A/D conversion is activated, the analog input is captured for the time interval specified by the sampling time setting bits (ST1, ST0). **Note:** • When "00_B" is set, the machine clock frequency should be equal to or less than 8 MHz. • If "00_B" is set when the machine clock frequency is 16 MHz, normal analog conversion values may not be obtained.
bit12 bit11	CT1, CT0: Comparison time setting bit	• This bit is used to set the compare time for A/D conversion. • The A/D conversion result is determined after the analog input is captured (sampling time passed) and the compare time interval specified by the compare time setting bits (CT1, CT0). In 10-bit resolution mode, the result is stored in the A/D data bits (D9 to D0). In 8-bit resolution mode, the result is stored in the A/D data bits (D7 to D0). **Note:** • When "00_B" is set, the machine clock frequency should be equal to or less than 8 MHz. • If "00_B" is set when the machine clock frequency is 16 MHz, normal analog conversion values may not be obtained.
vit10	-: Undefined bit	• The value read from this bit is undefined. • The value set to this bit does not affect operation.
bit9 - bit0	D9 - D0: A/D data bit	• These bits are used to store A/D conversion results. They are rewritten after each A/D conversion is completed. • Normally, the final conversion value is stored. • The initial value is undefined. **Note:** The A/D conversion data protection function is available (For details, see Section 16.6, "Operation of the 8/10-Bit A/D Converter"). Do not write data to the A/D data bits during A/D conversion.

Note:

- Be sure to stop the A/D conversion before rewriting the A/D conversion resolution setting bit (S10). If the bit is rewritten after the A/D conversion started, the A/D data register (ADCR0/ ADCR1) contents are undefined.
- To read the A/D data register (ADCR0/ADCR1), be sure to use the word transfer instructions (such as MOVW A and $002E_H$) when 10-bit resolution mode is specified.

16.5 8/10-Bit A/D Converter Interrupts

The 8/10-bit A/D converter can generate an interrupt request when the data for the A/D conversion is set in the A/D data register. This function supports the extended intelligent I/O service (EI^{2}OS).

■ 8/10-bit A/D Converter Interrupts

Table 16.5-1 Interrupt Control Bits of the 8/10-Bit A/D Converter and the Interrupt Cause

	8/10-bit A/D converter
Interrupt request flag bit	ADCS: INT="1"
Interrupt request enable bit	ADCS: INTE="1"
Interrupt cause	Writing the A/D conversion result to the A/D data register

When A/D conversion is activated and the A/D conversion result is stored in the A/D data register (ADCR0/ADCR1), the interrupt request flag bit (INT) of the A/D control status register (ADCS1) is set to "1". If the interrupt request enable bit (INTE) is "1", an interrupt request is output.

■ 8/10-Bit A/D Converter Interrupts and EI^{2}OS

Table 16.5-2 8/10-Bit A/D Converter Interrupts and EI^{2}OS

Interrupt No.	Interrupt control register		Vector table address			EI^{2}OS
	Register name	Address	Lower	Upper	Bank	
#11 ($0B_H$)	ICR00	$0000B0_H$	$FFFFD0_H$	$FFFFD1_H$	$FFFFD2_H$	o

o: Available

■ EI^{2}OS Function of the 8/10-Bit A/D Converter

Using the EI^{2}OS function, the 10-bit A/D converter can transfer A/D conversion results to memory. When using the EI^{2}OS function, the conversion data protection function works so that A/D conversion is temporarily stopped until A/D conversion data is transferred to memory and the A/D control status register (ADCS1) is cleared to "0" to prevent data omission.

16.6 Operation of the 8/10-Bit A/D Converter

The 8/10-bit A/D converter has four conversion modes: single conversion mode 1, single conversion mode 2, continuous conversion mode, and stop conversion mode. This section explains each of these modes.

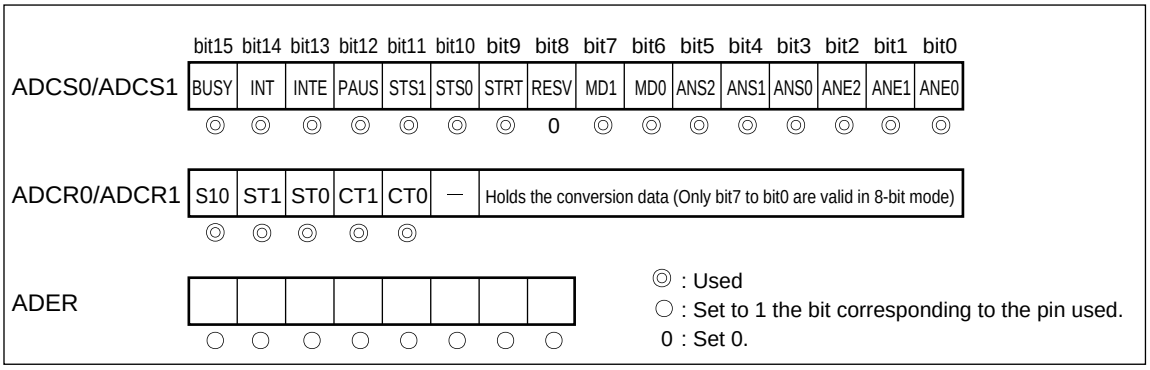
■ Operation in Single Conversion Mode 1/Operation in Single Conversion Mode 2

In single operation mode 1 and single operation mode 2, analog input from the start channel specified by the A/D conversion start channel setting bits (ANS2 to ANS0) of the A/D control status register (ADCS0) to the end channel specified by the A/D conversion end channel setting bits (ANE2 to ANE0) is A/D converted before terminating.

If the start channel and the end channel are the same, only one channel specified by the A/D conversion start channel setting bits (ANS2 to ANS0) is A/D converted.

Reactivation during operation is possible in single conversion mode 1, but is not possible in single conversion mode 2. For operation in single conversion mode 1 or 2, settings shown in Figure 16.6-1 "Settings for Single Conversion Mode" are required.

Figure 16.6-1 Settings for Single Conversion Mode



Reference:

The following are sample conversion sequences in single conversion mode:

ANS = 000_B, ANE = 011_B:AN0 --> AN1 --> AN2 --> AN3 --> End

ANS = 110_B, ANE = 010_B:AN6 --> AN7 --> AN0 --> AN1 --> AN2 --> End

ANS = 011_B, ANE = 011_B:AN3 --> END

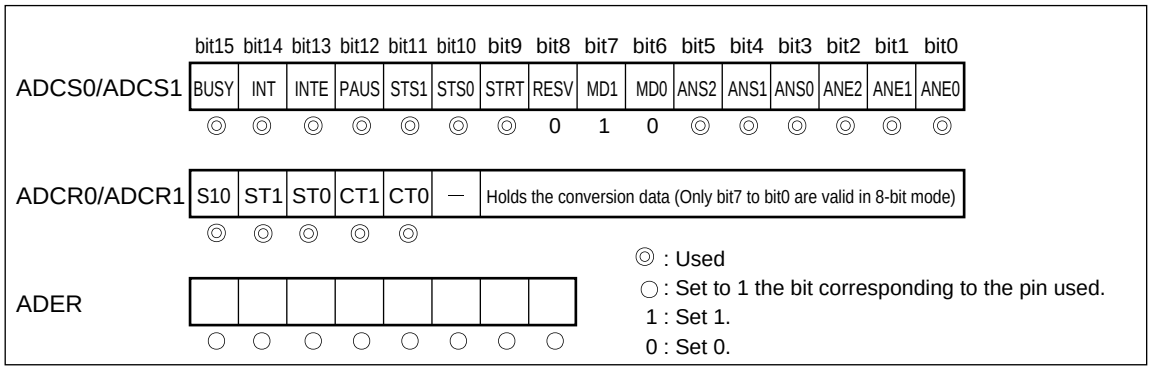
■ Operation in Continuous Conversion Mode

In continuous conversion mode, analog input from the start channel specified by the A/D conversion start channel setting bits (ANS2 to ANS0) of the A/D control status register (ADCS0) to the end channel specified by the A/D conversion end channel setting bits (ANE2 to ANE0) is A/D converted to return to the analog input specified by the A/D conversion start channel setting bits (ANS2 to ANS0) to repeat the A/D conversion.

If the start channel and the end channel are the same, the A/D conversion of the channel specified by the A/D conversion start channel setting bits (ANS2 to ANS0) is repeated.

The A/D conversion does not stop until the Converting bit (BUSY) of the A/D control status register (ADCS1) is set to "0". Reactivation during operation is not possible. For operation in continuous conversion mode, the settings shown in Figure 16.6-2 "Settings for Continuous Conversion Mode" are required.

Figure 16.6-2 Settings for Continuous Conversion Mode



Reference:

The following are sample conversion sequences in continuous conversion mode:

ANS = 000_B, ANE = 011_B:AN0 --> AN1 --> AN2--> AN3--> AN0--> Repeat

ANS = 110_B, ANE = 010_B:AN6 --> AN7 --> AN0--> AN1--> AN2 --> AN6 --> Repeat

ANS = 011_B, ANE = 011_B:AN3 --> AN3 --> Repeat

■ Operation in Stop Conversion Mode

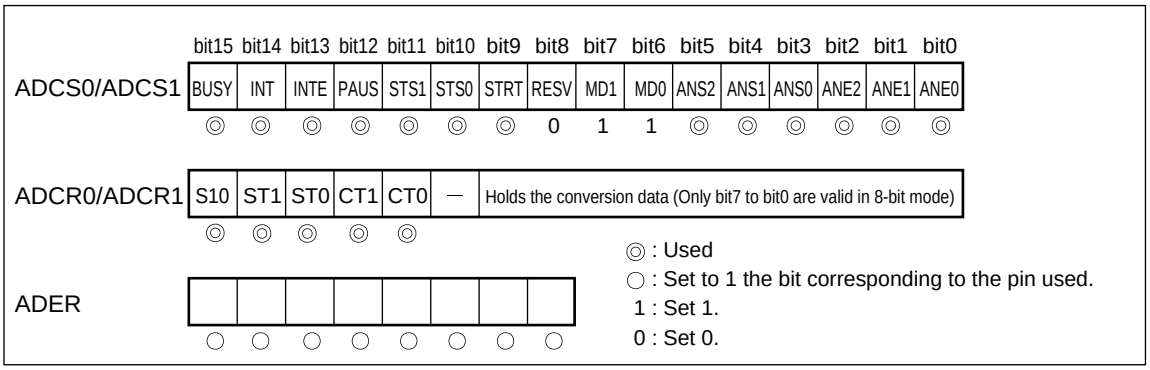
In stop conversion mode, analog input from the start channel specified by the A/D conversion start channel setting bits (ANS2 to ANS0) of the A/D control status register (ADCS0) to the end channel specified by the A/D conversion end channel setting bits (ANE2 to ANE0) is A/D converted by making a pause for each channel before returning to the analog input specified by the A/D conversion start channel setting bits (ANS2 to ANS0) to repeat the A/D conversion and pause.

If the start channel and the end channel are the same, the A/D conversion of the channel specified by the A/D conversion start channel setting bits (ANS2 to ANS0) is repeated.

In the pause state, reactivation method of the A/D conversion depends on the activation cause specified in the activation cause set bit (STS1, STS0) in the A/D control status register (ADCS1).

The A/D conversion does not stop until the Converting bit (BUSY) of the A/D control status register (ADCS1) is set to "0". Reactivation during operation is not possible. For operation in stop conversion mode, the settings shown in Figure 16.6-3 "Settings for Stop Conversion Mode" are required.

Figure 16.6-3 Settings for Stop Conversion Mode



Reference:

The following are sample conversion sequences in stop conversion mode:

ANS = 000_B, ANE = 011_B:

AN0 --> Pause --> AN1 --> Pause --> AN2 --> Pause --> AN3 --> Pause --> AN0 --> Repeat

ANS = 110_B, ANE = 001_B:

AN6 --> Pause --> AN7 --> Pause --> AN0 --> Pause --> AN1 --> Pause --> AN6 --> Repeat

ANS = 011_B, ANE = 011_B:

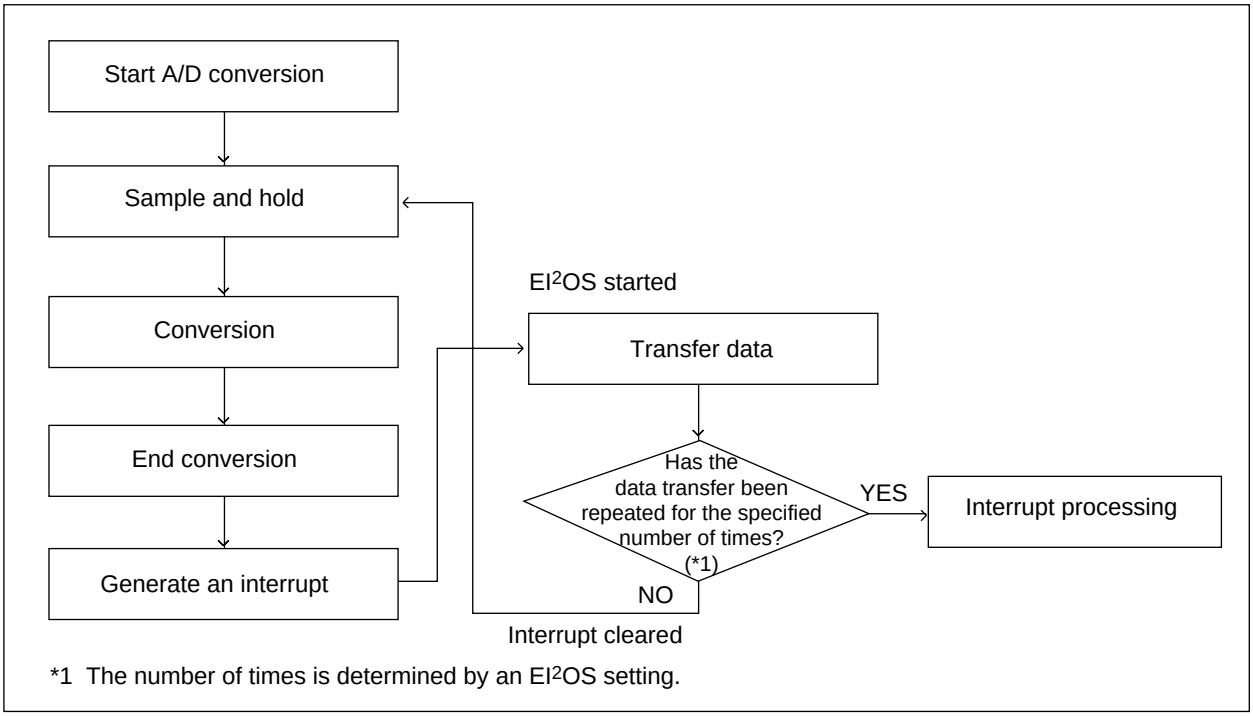
AN3 --> Pause --> AN3 --> Pause --> Repeat

16.6.1 Conversion Using EI^{2}OS

The 8/10-bit A/D converter can use EI^{2}OS transfer the A/D conversion result to memory.

■ Conversion Using EI^{2}OS

Figure 16.6-4 Sample Operation Flowchart When EI^{2}OS is Used



When EI^{2}OS is used, the conversion data protection function prevents any part of the data from being lost even in continuous conversion. Multiple data items can be safely transferred to memory.

16.6.2 A/D Conversion Data Protection Function

When A/D conversion is performed in the interrupt enabled state, the conversion data protection function operates.

■ A/D Conversion Data Protection Function

The 8/10-bit A/D converter has only one data register for conversion data storage. Thus, when A/D conversion is performed, the data stored in the data register is rewritten when the conversion is completed. In continuous conversion mode, if conversion data is transferred to memory too late, part of the stored data will be missing.

As measures against such data omission, the data protection function works as shown below if an interrupt request is enabled (ADCS1: INTE="1").

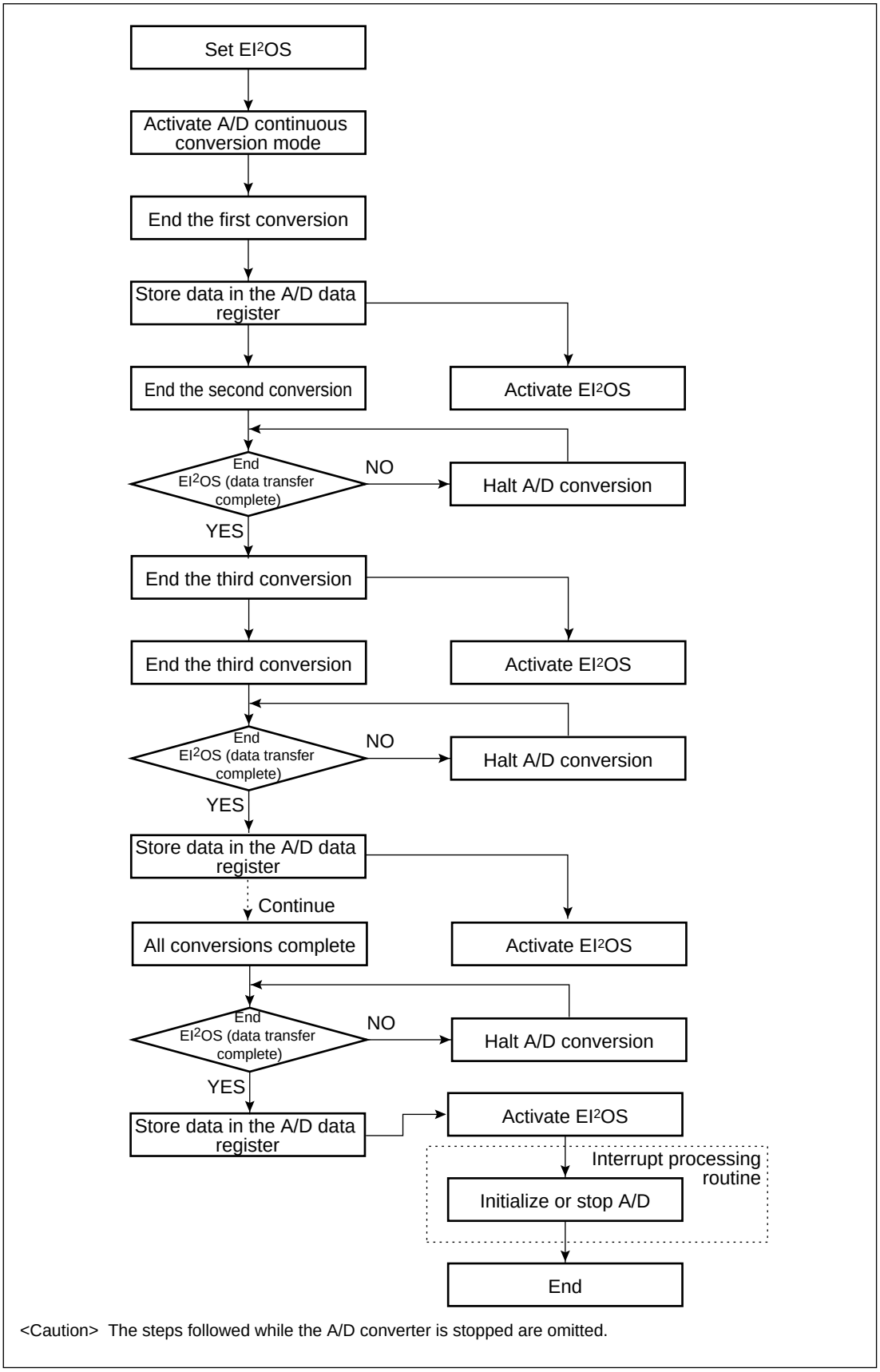
❍ Data protection function when EI^{2}OS is not used

When conversion data is stored in the A/D data register (ADCR0/ADCR1), the interrupt request flag bit (INT) of the A/D control status register 1 (ADCS1) is set to "1" and the A/D conversion is halted. The A/D conversion is restarted after transferring the A/D data register (ADCR0/ ADCR1) values to memory in the interrupt routine and the interrupt request flag bit (INT) is cleared to "0".

❍ Data protection function when EI^{2}OS is used

In continuous conversion mode, the pause flag bit (PAUS) of the A/D control status register 1 (ADCS1) is set to "1" if EI^{2}OS is used and A/D conversion is completed, but the transfer of the last piece of data to memory is not completed so that the A/D conversion is halted and conversion data is not stored in the A/D data register (ADCR0/ADCR1). When the transfer of the last piece of data to memory is completed, the pause flag bit (PAUS) is cleared to "0" and the A/D conversion is restarted.

Figure 16.6-5 Flow of the Data Protection Function when EI^{2}OS is Used



<Caution> The steps followed while the A/D converter is stopped are omitted.

Note:

- The conversion data protection function operates only in the interrupt enabled state (ADCS1: INTE = 1).
- If interrupts are disabled during a pause of A/D conversion while EI^{2}OS is operating, the A/D conversion may be reactivated. This will cause new data to be written before the old data is transferred.
- Reactivation attempted during a pause will destroy the standby data.

16.7 Usage Notes on the 8/10-Bit A/D Converter

Notes on using the 8/10-bit A/D converter.

■ Usage Notes on the 8/10-Bit A/D Converter

❍ **Analog input pin**

The analog input pins of the A/D converter are also used as the I/O pins of port 5. The pins are used by switching the port 5 direction register (DDR5) and the analog input enable register (ADER). To use a pin for analog input of the A/D converter, set the corresponding bit (bit15 to bit8) of the port 5 direction register (DDR5) to "0" (Set as an input port) and then set the corresponding bit (bit7 to bit0) of the analog input enable register (ADER) to "1" (Set the analog input mode) to determine the input gate on the port side. If an intermediate-level signal is input in port I/O mode (ADER: bit7 to bit0="0"), an input leakage current flows through the gate.

For details of the port 5 direction register (DDR5) and the analog input enable register (ADER), see Section 8.8.1, "Port 5 Registers (PDR5, DDR5, and ADER)".

❍ **Note on using an internal timer**

To activate the A/D converter with the internal timer, set the A/D activation trigger setting bits (STS1, STS0) of the A/D control status register (ADCS1). Set the internal timer to the inactive level ("L" for the internal clock). If the internal clock remains at the active level and data is written to the A/D control status register (ADCS0/ADCS1), the A/D converter may be activated.

❍ **Sequence of turning on the A/D converter and analog input**

Be sure to apply the voltage to the power supply pins (AVcc, AVR, and AVss) of the A/D converter and the analog input pins (AN0 to AN7) after turning on the digital power supply (Vcc). Turn off the digital power supply (Vcc) after turning off the A/D converter and the analog input power supply. Turn on and turn off the voltage so that AVR does not exceed AVcc.

❍ **Supply voltage to the A/D converter**

The supply voltage to the A/D converter (AVcc) must not exceed the digital power supply (Vcc); otherwise, latchup may occur.

16.8 Sample Program 1 for the 8/10-Bit A/D Converter (Single Conversion Mode Using EI^{2}OS)

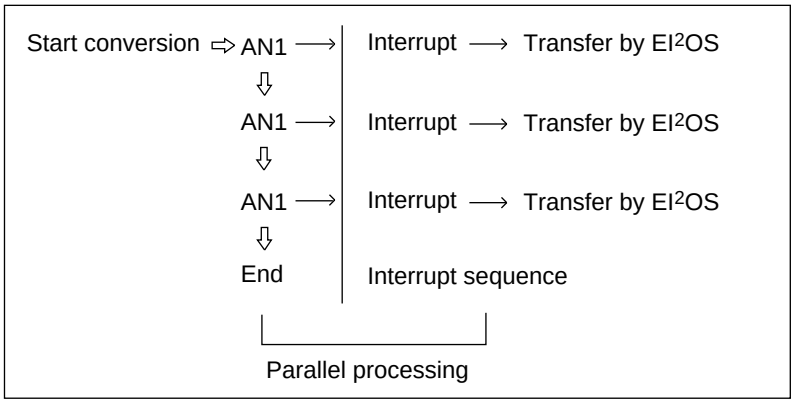
This section contains a sample program for A/D conversion in single conversion mode using EI^{2}OS.

■ Sample Program for Single Conversion Mode Using EI^{2}OS

❍ Processing

- Analog inputs AN1 to AN3 are converted once.
- The conversion data is sequentially transferred to addresses 200_H to 205_H.
- A resolution of 10 bits is selected.
- The conversion is activated by software.

Figure 16.8-1 Flowchart of an Example of the EI^{2}OS Activation Program in Single Conversion Mode



❍ **Coding example**

```
BAPL  EQU 000100H        ;Lower buffer address pointer
BAPM  EQU 000101H        ;Middle buffer address pointer
BAPH  EQU 000102H        ;Upper buffer address pointer
ISCS  EQU 000103H        ;EI  OS status register
IOAL  EQU 000104H        ;Lower I/O address register
IOAH  EQU 000105H        ;Upper I/O address register
DCTL  EQU 000106H        ;Lower data counter
DCTH  EQU 000107H        ;Upper data counter
DDR5  EQU 000015H        ;Port 5 direction register
ADER  EQU 000017H        ;Analog input enable register
ICR00 EQU 0000B0H        ;Interrupt control register for A/DC
ADCS0 EQU 000034H        ;A/D control status register
ADCS1 EQU 000035H        ;
ADCR0 EQU 000036H        ;A/D data register
ADCR1 EQU 000037H        ;
;-------Main program-----------------------------------------------
CODECSEG
START:                   ;Assumes that the stack pointer (SP) has
                          already been initialized.
      AND CCR,#0BFH      ;Disables interrupts.
      MOV ICR00,#00H     ;Interrupt level:  0 (highest priority)
      MOV BAPL,#00H      ;Sets the address to which the conversion
                          data is transferred and stored.
      MOV BAPM,#02H      ;(Uses 200H to 205H.)
      MOV BAPH,#00H      ;
      MOV ISCS,#18H      ;Transfers word data, adds 1 to the
                          address, then transfers the data from
                          I/O to memory.
      MOV IOAL,#36H      ;Sets the address of the analog data
register as the
      MOV IOAH,#00H      ;transfer source address pointer.
      MOV DCTL,#03H      ;Sets the EI  OS transfer count to three,
                          which is the same value as the conversion
                          count.
      MOV DDR5,#11110001B;Sets P51 to P53 as input.
      MOV ADER,#00001110B;Sets P51/AN1 to P53/AN3 as analog inputs.
      MOV DCTH,#00H      ;
      MOV ADCS0,#0BH     ;Single activation.  Converts AN1 to AN3.
      MOV ADCS1,#0A2H    ;Software activation.  Begins A/D
                          conversion. Enables interrupts.
      MOV ILM,#07H       ;Sets ILM in PS to level 7.
      OR  CCR,#40H       ;Enables interrupts.
LOOP: MOV A,#00H         ;Endless loop
      MOV A,#01H
      BRA LOOP
;-------Interrupt program------------------------------------------
 ED_INT1:
      MOV I:ADCS1,#00H   ;Stops A/D conversion.  Clears and disables
                          the interrupt flag.
      RETI               ;Returns from interrupt.
 CODEENDS
;-------Vector setting---------------------------------------------
VECT CSEG ABS=0FFH
     ORG  0FFD0H         ;Sets vector for interrupt #11 (0BH)
     DSL  ED_INT1
```

```
        ORG  0FFDCH          ;Sets reset vector.
        DSL  START
        DB   00H             ;Sets single-chip mode.
VECT    ENDS
        END  START
```

16.9 Sample Program 2 for the 8/10-Bit A/D Converter (Continuous Conversion Mode Using EI^{2}OS)

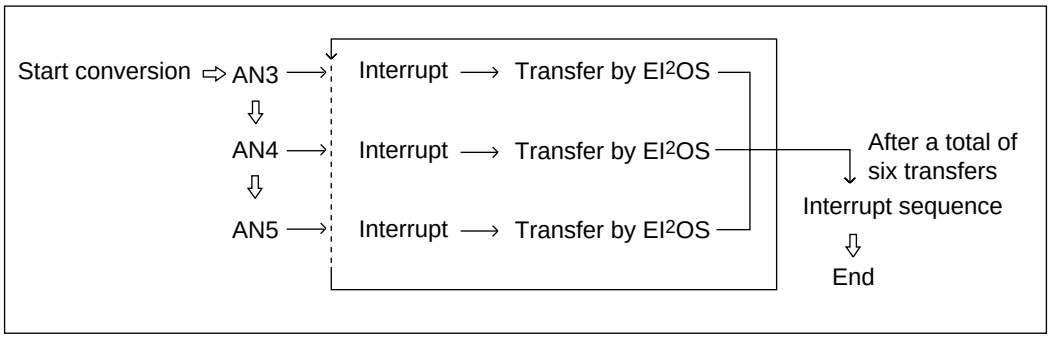
This section contains a sample program for A/D conversion in continuous conversion mode using EI^{2}OS.

■ Sample Program for Continuous Conversion Mode Using EI^{2}OS

❍ Processing

- Analog inputs AN3 to AN5 are converted twice. Two conversion data items are obtained for each channel.
- The conversion data is sequentially transferred to addresses 600_H to $60B_H$.
- A resolution of 10 bits is selected.
- The conversion is activated by 16-bit reload timer 1.

Figure 16.9-1 Flowchart of an Example of the EI^{2}OS Activation Program in Continuous Conversion Mode



❍ **Coding example**

```
BAPL     EQU  000100H        ;Lower buffer address pointer
BAPM     EQU  000101H        ;Middle buffer address pointer
BAPH     EQU  000102H        ;Upper buffer address pointer
ISCS     EQU  000103H        ;EIðOS status registerr
IOAL     EQU  000104H        ;Lower I/O address register
IOAH     EQU  000105H        ;Upper I/O address register
DCTL     EQU  000106H        ;Lower data counter
DCTH     EQU  000107H        ;Upper data counter
DDR5     EQU  000015H        ;Port 5 direction register
ADER     EQU  000017H        ;Analog input enable register
ICR00    EQU  0000B0H        ;Interrupt control register for A/DC
ADCS0    EQU  000034H        ;A/D control status register
ADCS1    EQU  000035H        ;
ADCR0    EQU  000036H        ;A/D data register
ADCR1    EQU  000037H        ;
TMCSR1:L EQU  000086H        ;Control status register 1
TMCSR1:H EQU  000087H        ;
TMRLR1   EQU  000088H        ;16-bit reload register
TMRHR1   EQU  000089H        ;
;-------Main program-----------------------------------------------
CODE
START:                       ;Assumes that the stack pointer (SP)
                              has already been initialized.
         AND  CCR,#0BFH      ;Disables interrupts.
         MOV  ICR00,#08H     ;Interrupt level;  0 (highest
                              priority).  Enables interruupts.
         MOV  BAPL,#00H      ;Sets the address to which conversion
                              data is stored.
         MOV  BAPM,#06H      ;Uses 600H to 60BH.)
         MOV  BAPH,#00H      ;
         MOV  ISCS,#18H      ;Transfers word data, adds 1 to
                              the address, then
                             ;transfers from I/O to memory.
         MOV  IOAL,#36H      ;Sets the address of the analog
                              data register as the
         MOV  IOAH,#00H      ;transfer source address pointer.
         MOV  DCTL,#06H      ;Six transfers by EI2OS (two transfers
                              each for three channels)
         MOV  DDR5,#00000000B;Sets P50 to P57 as input.
         MOV  ADER,#00111000B;Sets P53/AN3 to P55/AN5 as analog
                              input.
         MOV  DCTH,#00H      ;
         MOV  ADCS0,#9DH     ;Continuous conversion mode.  Converts
                              AN3 to AN5 CH.
         MOV  ADCS1,#0A8H    ;Activates the 16-bit timer, starts A/D
                              conversion, and enables interrupts.
         MOVW TMRLR1,#0320H  ;Sets the timer value to 800 (320H),
                              100 µs.
         MOV  TMCRH1,#00H    ;Sets the clock source to 125 ns and
                              disables external trigger.
         MOV  TMCRL1,#12H    ;Disables timer output, disables
                              interrupts, and enables reload.
         MOV  TMCRL1,#13H    ;Activates the 16-bit timer.
         MOV  ILM,#07H       ;Sets ILM in PS to level 7.
         OR   CCR,#40H       ;Enables interrupts.
```

```
LOOP:    MOV  A,#00H         ;Endless loop
         MOV  A,#01H
         BRA  LOOP
;-------Interrupt program-----------------------------------------
ED_INT1:
         MOV  I:ADCS1,#80H   ;Does not stop A/D conversion.  Clears
                              and disables the interrupt flag.
         RETI                ;Returns from interrupt.
CODE     ENDS
;-------Vector setting--------------------------------------------
VECT     CSEG ABS=0FFH
         ORG  0FFD0H         ;Sets vector for interrupt #11 (0BH).
         DSL  ED_INT1
         ORG  0FFDCH         ;Sets reset vector.
         DSL  START
         DB   00H            ;Sets single-chip mode.
VECT     ENDS
         END  START
```

16.10 Sample Program 3 for the 8/10-Bit A/D Converter (Stop Conversion Mode Using EI^{2}OS)

This section contains a sample program for A/D conversion in stop conversion mode using EI^{2}OS.

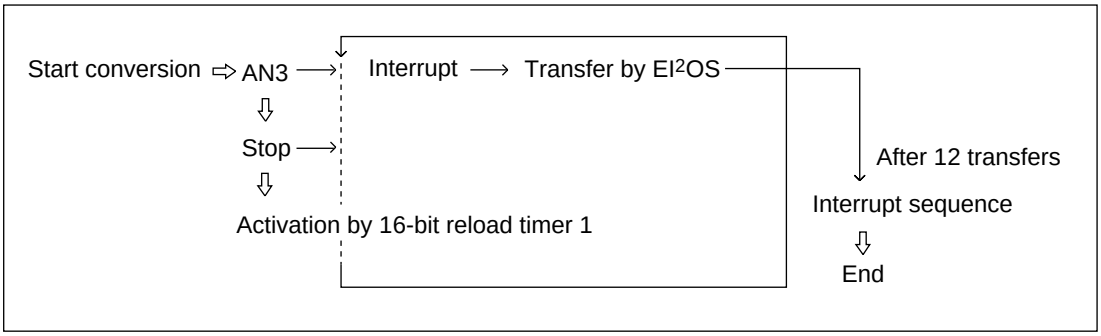
■ Sample Program for Stop Conversion Mode Using EI^{2}OS

❍ Processing

- Analog input AN3 is converted 12 times at regular intervals.
- The conversion data is sequentially transferred to addresses 600_H to 617_H.
- A resolution of 10 bits is selected.
- The conversion is activated by 16-bit reload timer.

Figure 16.10-1 "Flowchart of Program Using EI^{2}OS (Stop Conversion Mode)" shows a flowchart of the program using EI^{2}OS (stop conversion mode).

Figure 16.10-1 Flowchart of an Example of the EI^{2}OS Activation Program in Stop Conversion Mode



❍ **Coding example**

```
BAPL     EQU  000100H        ;Lower buffer address pointer
BAPM     EQU  000101H        ;Middle buffer address pointer
BAPH     EQU  000102H        ;Upper buffer address pointer
ISCS     EQU  000103H        ;EI2OS status registerr
IOAL     EQU  000104H        ;Lower I/O address register
IOAH     EQU  000105H        ;Upper I/O address register
DCTL     EQU  000106H        ;Lower data counter
DCTH     EQU  000107H        ;Upper data counter
DDR5     EQU  000015H        ;Port 5 direction register
ADER     EQU  000017H        ;Analog input enable register
ICR00    EQU  0000B0H        ;Interrupt control register for A/DC
ADCS0    EQU  000034H        ;A/D control status register
ADCS1    EQU  000035H        ;
ADCR0    EQU  000036H        ;A/D data register
ADCR1    EQU  000037H        ;
TMCSR1:L EQU  000086H        ;Control status register 1
TMCSR1:H EQU  000087H        ;
TMRLR1   EQU  000088H        ;16-bit reload register
TMRHR1   EQU  000089H        ;
;-------Main program-----------------------------------------------
CODE     CSEG
START:                       ;Assumes that the stack pointer (SP)
                              has already been initialized.
         AND  CCR,#0BFH      ;Disables interrupts.
         MOV  ICR00,#08H     ;Interrupt level:  0 (highest
                              priority).
         MOV  BAPL,#00H      ;Sets the address to which conversion
                              data is stored.
         MOV  BAPM,#06H      ;(Uses 600H to 617H.)
         MOV  BAPH,#00H      ;
         MOV  ISCS,#19H      ;Transfers word data, adds 1 to
                              the address,
                             ;transfers from I/O to memory, then
                             ;ends by aresource request.
         MOV  IOAL,#36H      ;Sets the address of the analog
                              data register as the
         MOV  IOAH,#00H      ;transfer source address pointer.
         MOV  DCTL,#0CH      ;Transfers only channel 3 twelve
                              times by  EI2OS
         MOV  DDR5,#00000000B;Sets P50 to P57 as input.
         MOV  ADER,#00001000B;Sets P53/AN3 as analog input.
         MOV  ADCS0,#0DBH    ;Stop conversion mode.  Converts
                              AN3 CH.
         MOV  ADCS1,#0A8H    ;Activates the 16-bit timer, starts A/D
                              conversion, and enables interrupts.
         MOVW TMRLR1,#0320H  ;Sets the timer value to 800 (320H),
                              100 µs.
         MOV  TMCRH1,#00H    ;Sets the clock source to 125 ns and
                              disables external trigger.
         MOV  TMCRL1,#12H    ;Disables timer output, disables
                              interrupts, and enables reload.
         MOV  TMCRL1,#13H    ;Activates the 16-bit timer.
         MOV  ILM,#07H       ;Sets ILM in PS to level 7.
         OR   CCR,#40H       ;Enables interrupts.
LOOP:    MOV  A,#00H         ;Endless loop
```

```
         MOV  A,#01H
         BRA  LOOP
;-------Interrupt program-----------------------------------------
ED_INT1:
         MOV  I:ADCS1,#80H   ;Does not stop A/D conversion.  Clears
                              and disables the interrupt flag.
         RETI                ;Returns from interrupt.
CODE     ENDS
;-------Vector setting--------------------------------------------
VECT     CSEG ABS=0FFH
         ORG  0FFD0H         ;Sets vector for interrupt #11 (0BH).
         DSL  ED_INT1
         ORG  0FFDCH         ;Sets reset vector.
         DSL  START
         DB   00H            ;Sets single-chip mode.
VECT     ENDS
         END  START
```

CHAPTER 17 ADDRESS MATCH DETECTION FUNCTION

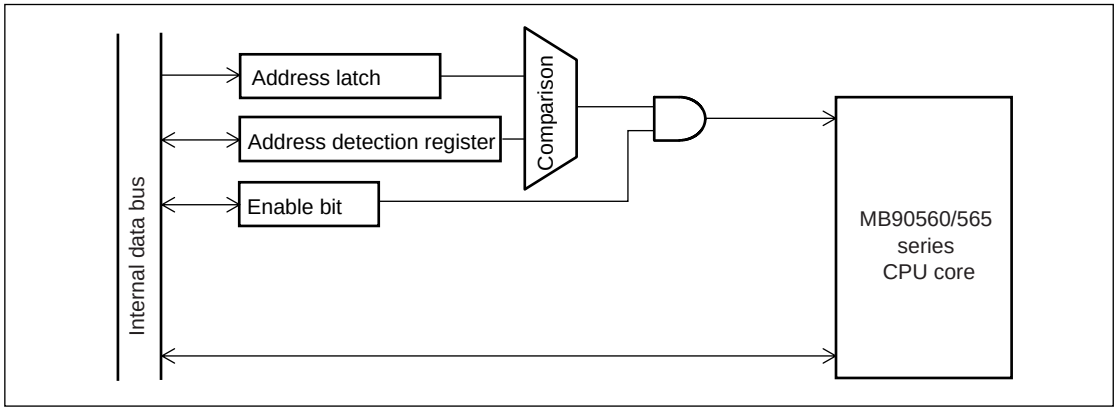
This chapter describes the address match detection function and operation of the MB90560/565 series.

17.1 Overview of the Address Match Detection Function

If the program address matches the value set in the address match detection register, the instruction code to be read by the CPU is replaced with an INT9 instruction code. By performing processing using an INT #9 interrupt routine, a program patch application function can be implemented.

■ Block Diagram of the Address Match Detection Function

Figure 17.1-1 Block Diagram of the Address Match Detection Function

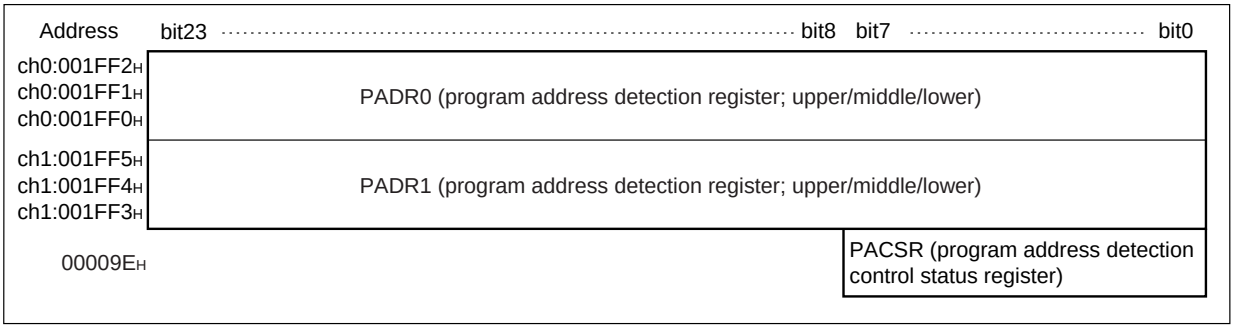


17.2 Registers of the Address Match Detection Function

This section shows a list of registers of the address match detection function.

■ List of Registers of the Address Match Detection Function

Figure 17.2-1 List of Registers of the Address Match Detection Function

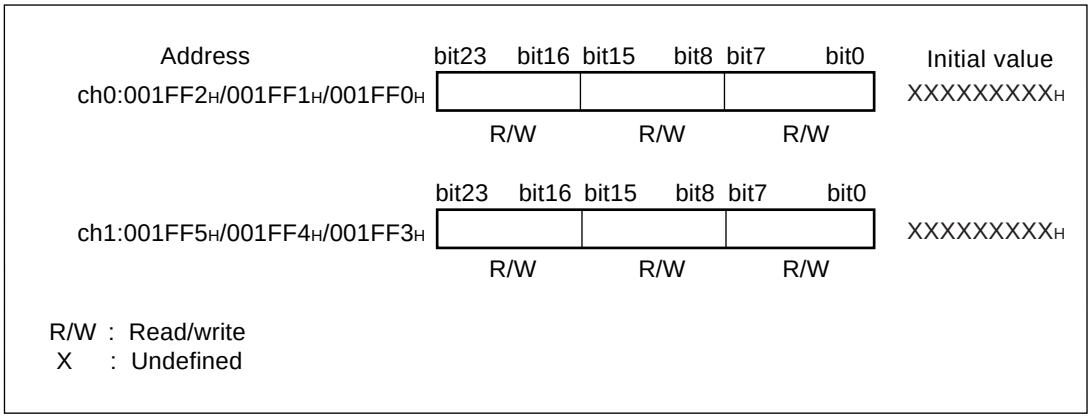


17.2.1 Program Address Detection Register (PADR0/PADR1)

The program address detection register (PADR0/PADR1) is used to set the address for comparison.

■ Program Address Detection Register (PADR0/PADR1)

Figure 17.2-2 Program Address Detection Register (PADR0/PADR1)



If the corresponding interrupt enable bit of the program address detection control status register (PACSR) is "1", the program address is compared with the value set in the program address detection register (PADR0/PADR1). If both match, an INT9 instruction is output. If the interrupt enable bit is "0", the INT9 instruction is not output.

The following table lists the correspondence between the program address detection control status register (PACSR) and the interrupt enable bit.

Address detection register	Interrupt enable bit
PADR0	AD0E
PADR1	AD1E

17.2.2 Program Address Detection Control Status Register (PACSR)

The program address detection control status register (PACSR) is used to perform the interrupt control of the address match detection function.

■ **Program Address Detection Control Status Register (PACSR)**

Figure 17.2-3 Program Address Detection Control Status Register (PACSR)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00009E$_H$	RESV	RESV	RESV	RESV	AD1E	RESV	AD0E	RESV	00000000$_B$
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W : Read/write
X : Undefined

Table 17.2-1 Functional Explanation of Each Bit of the Program Address Detection Control Status Register (PACSR)

Bit name		Function
bit7 bit6 bit5 bit4	RESV: Reserved bit	• Always set "0".
bit3	AD1E: PADR1 interrupt request enable bit	• This bit enables an interrupt of PADR0. • When this bit is "1", the program address detection register (PADR1) value and the program address are compared. If both match, an INT9 instruction is output. •
bit2	RESV: Reserved bit	• Always set "0".
bit1	AD0E: PADR0 interrupt request enable bit	• This bit enables an interrupt of PADR0. • When this bit is "1", the program address detection register (PADR0) value and the program address are compared. If both match, the interrupt flag bit (AD0D) of the PADR0 is set to "1" and an INT9 instruction is output.
bit0	RESV: Reserved bit	• Always set "0".

17.3 Operation of the Address Match Detection Function

This section explains the operation of the address match detection function.

■ Operation of the Address Match Detection Function

If the program address matches the value set in the address match detection register, the instruction code to be read by the CPU is replaced with an INT9 instruction code (01_H). When the CPU executes the instruction at the specified program address, the INT9 instruction is executed. By performing processing using an INT #9 interrupt routine, a program patch application function can be implemented.

Two program address detection registers (PADR0/PADR1) are available, each of which has an interrupt enable bit (AD1E, AD0E). If the interrupt enable bit (AD1E, AD0E) is "1", the value set in the address detection register and the program address are compared. If both match, the instruction code to be read by the CPU is replaced with an INT9 instruction code.

Note:

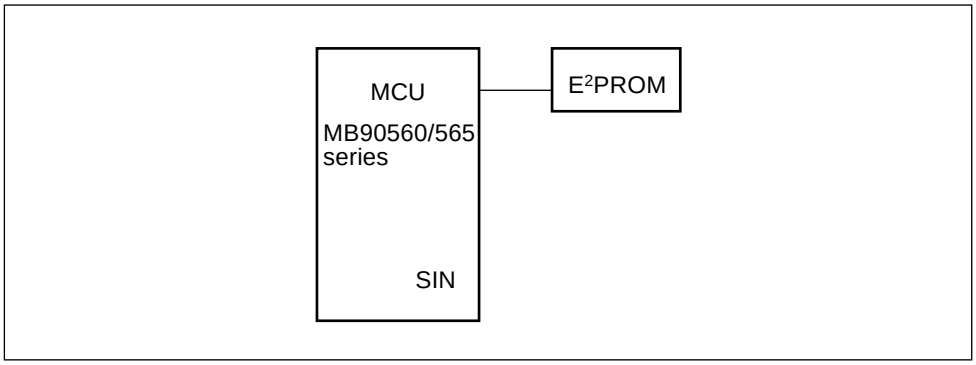
The address detection function does not work correctly if a program address after the 1st byte of the instruction is set in the address detection register. Change the address detection register after setting the interrupt enable bit to "0". If the setting of the address detection register is changed while the interrupt enable bit is set to "1", an address detection may be mistakenly performed while making settings.

17.4 Example of Using the Address Match Detection Function

This section contains example of Using the Address Match Detection Function.

■ System Configuration

Figure 17.4-1 System Configuration Example



■ E²PROM Memory Map

Address	Meaning
0000_H	Number of bytes of patch program No. 0 (0 for no program error)
0001_H	Bit 7 to bit 0 of program address No. 0
0002_H	Bit 15 to bit 8 of program address No. 0
0003_H	Bit 24 to bit 16 of program address No. 0
0004_H	Number of bytes of patch program No. 1 (0 for no program error)
0005_H	Bit 7 to bit 0 of program address No. 1
0006_H	Bit 15 to bit 8 of program address No. 1
0007_H	Bit 24 to bit 16 of program address No. 1
0010_H+ Number of bytes of patch program No. 0	Original of patch program No. 0

■ Initial State

The contents of E²PROM are all 0s.

■ INT9 Interrupt

An interrupt routine determines which address cause triggered an interrupt request based on the value of the program counter (PC) saved in the stack before branching to the program from which the interrupt request was output.

Figure 17.4-2 System Configuration Example

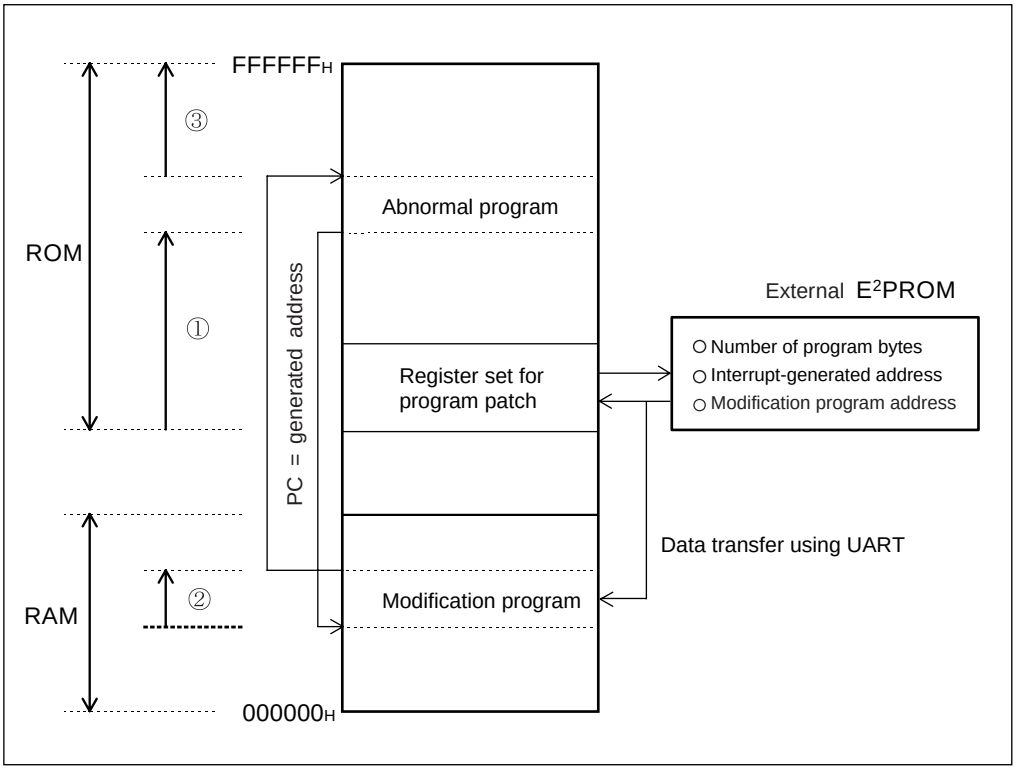
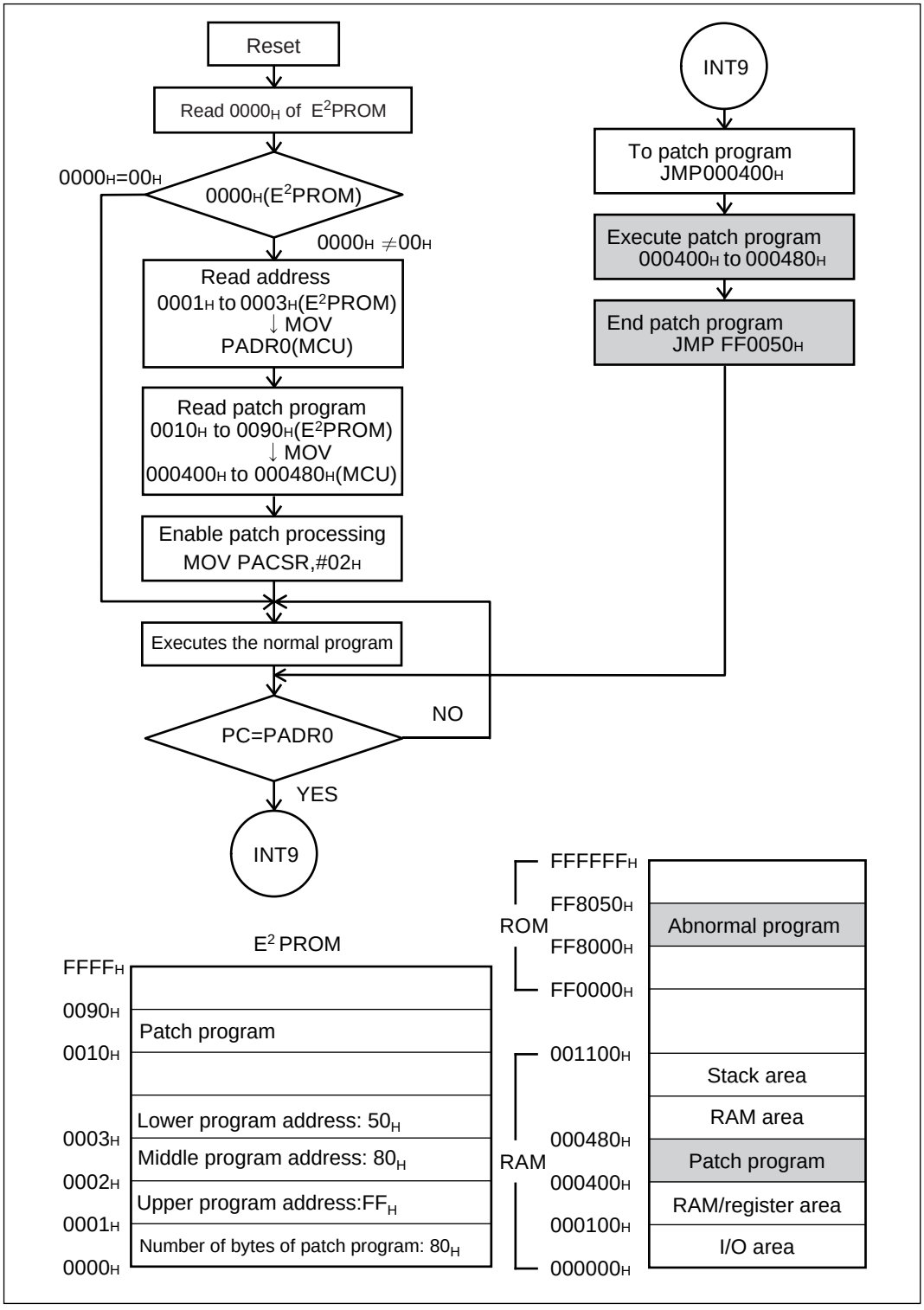


Figure 17.4-3 Flowchart of Program Patch Processing



CHAPTER 18 ROM MIRRORING FUNCTION SELECTION MODULE

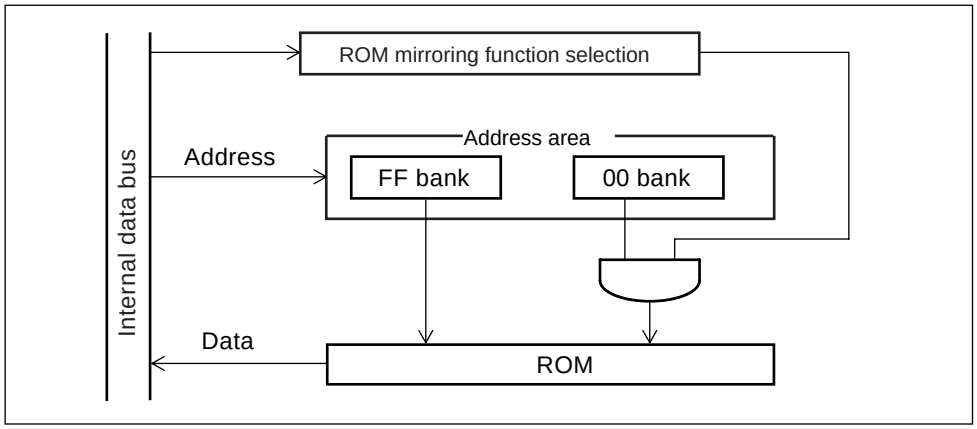
This chapter describes the function and operation of the MB90560/565 series ROM mirroring function selection module.

18.1 Overview of the ROM Mirroring Function Selection Module

The ROM mirroring function selection module can access ROM data in bank FF from bank 00 by setting its register.

■ Block diagram of the ROM mirroring function selection module

Figure 18.1-1 Block Diagram



18.2 ROM Mirroring Function Selection Register (ROMM)

This section explains the register in the ROM mirroring function selection module.

■ ROM Mirroring Function Selection Register (ROMM)

Figure 18.2-1 ROM Mirroring Function Selection Register (ROMM)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
00006F$_H$	-	-	-	-	-	-	-	MI	XXXXXXX1$_B$
	-	-	-	-	-	-	-	W	

W : Write only
X : Undefined
- : Undefined bit

Note:

Do not set the ROM mirroring function selection register while accessing the addresses "004000$_H$ to 00FFFF$_H$".

Table 18.2-1 Functional Explanation of the ROM Mirroring Function Selection Register (ROMM)

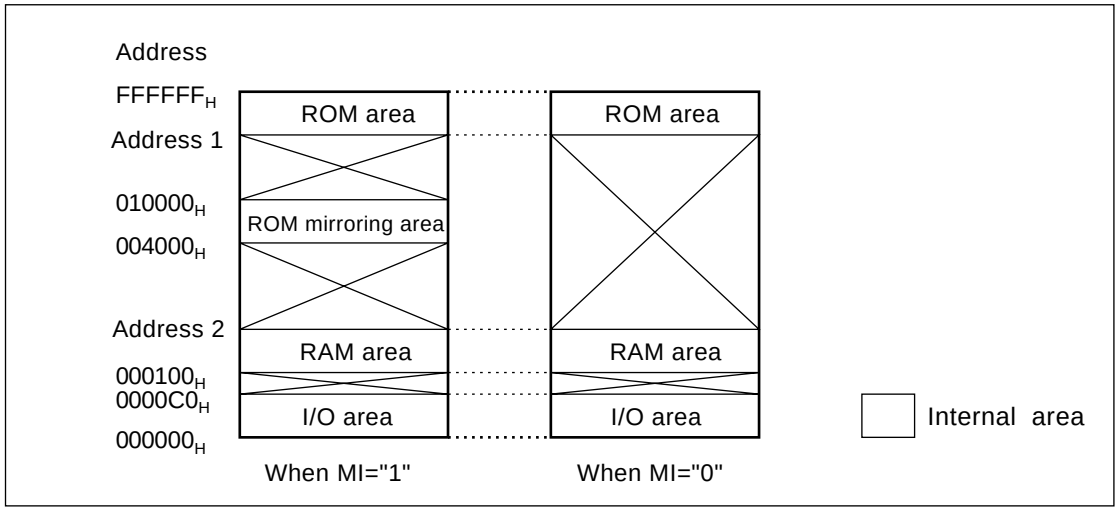
Bit name		Function
bit15 to bit9	-: Undefined bit	• The values read from these bits are undefined.
bit8	MI: ROM mirroring function setting bit	This bit is used to set the ROM mirroring function. When this bit is "1", ROM data in bank FF can be read from bank 00. When this bit is "0", ROM data in bank FF cannot be read from bank 00.

Note:

Bank 00 accesses the addresses "FF4000$_H$ to FFFFFF$_H$" from the addresses "004000$_H$ to 00FFFF$_H$". Therefore, the addresses less than "FF3FFF$_H$" cannot be accessed even if the ROM mirroring function is set.

	MB90561/A	MB90562/A	MB90F562/B	MB90567	MB90568	MB90F568	MB90V560
Address 1	FF8000$_H$	FF0000$_H$	FF0000$_H$	FE8000$_H$	FE0000$_H$	FE0000$_H$	FE0000$_H$
Address 2	000500$_H$	000900$_H$	000900$_H$	001100$_H$	001100$_H$	001100$_H$	001100$_H$

Figure 18.2-2 Memory Space



CHAPTER 19 512K-BIT (64 KB) FLASH MEMORY

This chapter describes the functions and operations of the 512K-bit (64KB) flash memory of the MB90560/565 series.

- **Parallel programmer**
- **Serial dedicated programmer**
- **Write/erase operation by program execution**

This chapter provides an explanation for the above item "Write/erase operation by program execution."

19.1 Overview of the 512K-Bit Flash Memory

The 512K-bit (64K bytes) flash memory is allocated in the FF bank on the CPU memory map, and the function of the flash memory interface circuit enables the read/access or program access from the CPU to the flash memory, same as the mask ROM. The write/erase operation to the flash memory can be executed through the flash memory interface circuit by executing an instruction issued from the CPU. Therefore, the flash memory mounted can be rewritten under the control of the internal CPU, so that the program or data can be upgraded or updated more efficiently.
However, no selector operation such as the enable sector protect can be used.

■ Characteristics of the 512K-Bit Flash Memory

- 64K words x 8 bits/32K words x 16 bits (16K+8K+8K+32K) sector configuration
- Use of automatic program algorithm (Embedded Algorithm: Equivalent to MBM29F400TA)
- Erase pause/restart function provided
- Detection of completion of writing/erasing using data polling or toggle bit functions
- Detection of completion of writing/erasing using CPU interrupts
- Compatibility with the JEDEC standard-type command
- Sector erase function (any combination of sectors)
- Number of write/erase operations 10,000 times guaranteed.

"Embedded Algorithm" is the trademark of Advanced Micro Device.

■ Writing to/Erasing Flash Memory

The write/erase operation of the flash memory cannot be executed simultaneously. In executing the data write/erase operation in the flash memory, only the write operation can be executed without a program access from the flash memory, by copying a program on the flash memory to RAM and executing the program.

For details, see Section 19.6.2, "Writing the Data".

■ Register on the Flash Memory

❍ Flash memory control status register (FMCS)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000AE$_H$	INTE	RDYINT	WE	RDY	Reserved	LPM1	Reserved	LPM0	00000000$_B$
	R/W	R/W	R/W	R	W	W	W	R/W	

R/W : Read/write
R : Read only
W : Write only

19.2 512K-Bit Flash Memory Sector Configuration

Figure 19.2-1 "512K-Bit (64KB) Flash Memory Sector Configuration" shows the sector configuration in the 512K-bit flash memory. The address indicated in Figure 19.2-1 "512K-Bit Flash Memory Sector Configuration" is classified into the upper address and lower address of each sector.

■ Sector Configuration

For access from the CPU, SA0 to SA3 are allocated to the FF bank register.

Figure 19.2-1 512K-Bit (64KB) Flash Memory Sector Configuration

Flash memory		CPU address	Programmer address *
SA3(16K bytes)	Upper	FFFFFF$_H$	7FFFF$_H$
	Lower	FFC000$_H$	7C000$_H$
SA2(8K bytes)	Upper	FFBFFF$_H$	7BFFF$_H$
	Lower	FFA000$_H$	7A000$_H$
SA1(8K bytes)	Upper	FF9FFF$_H$	79FFF$_H$
	Lower	FF8000$_H$	78000$_H$
SA0(32K bytes)	Upper	FF7FFF$_H$	77FFF$_H$
	Lower	FF0000$_H$	70000$_H$

* Programmer address
The programmer address is equivalent to the CPU address when writing the data to the flash memory using the parallel programmer.
If the write/erase operation is executed using the general-purpose programmer, the write/erase operation is executed using this address.

19.3 Flash Memory Control Status Register (FMCS)

The flash memory control status register (FMCS) is used to control writing/erasing of flash memory.

■ Flash Memory Control Status Register (FMCS)

Figure 19.3-1 Flash Memory Control Status Register (FMCS)

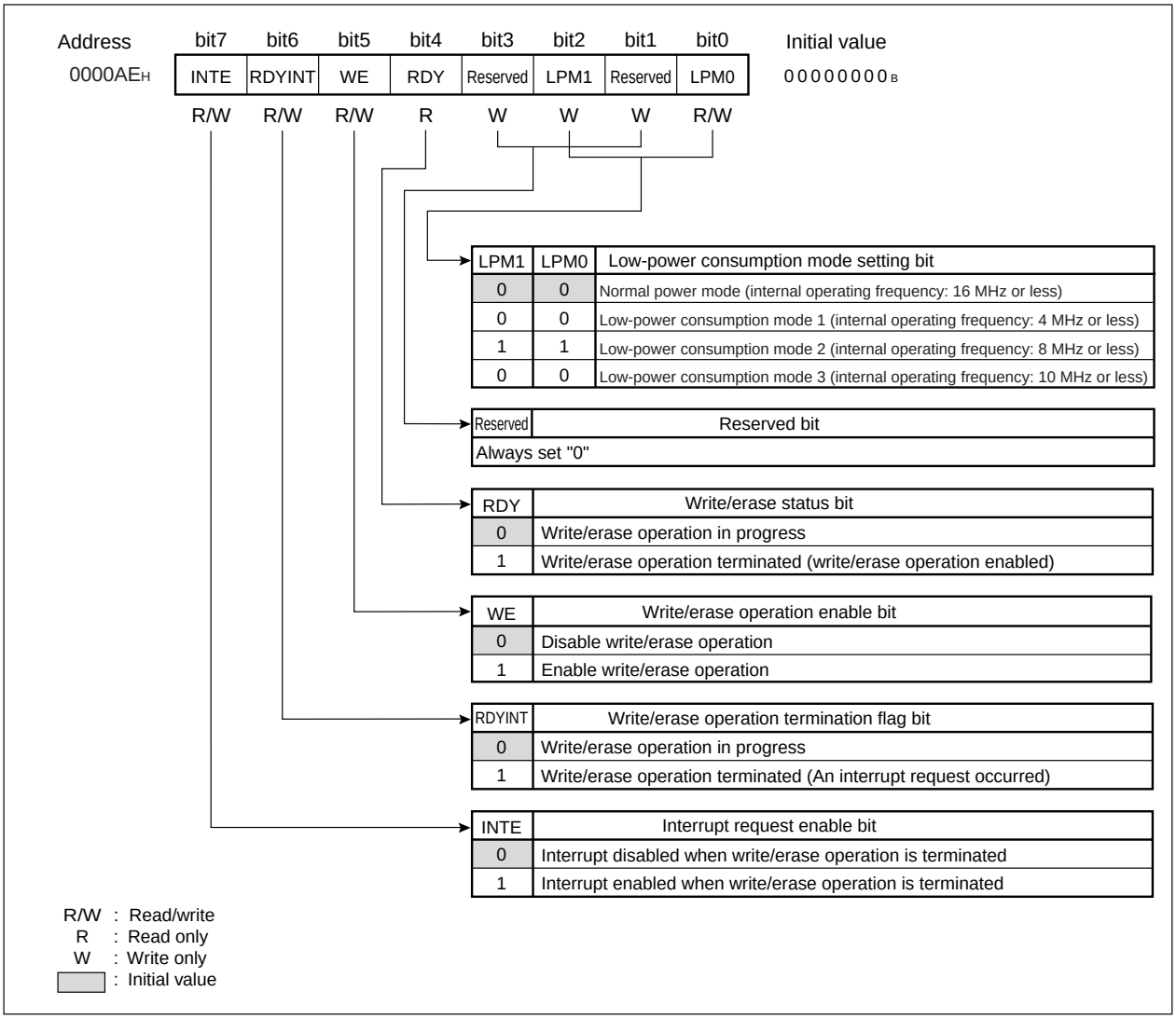
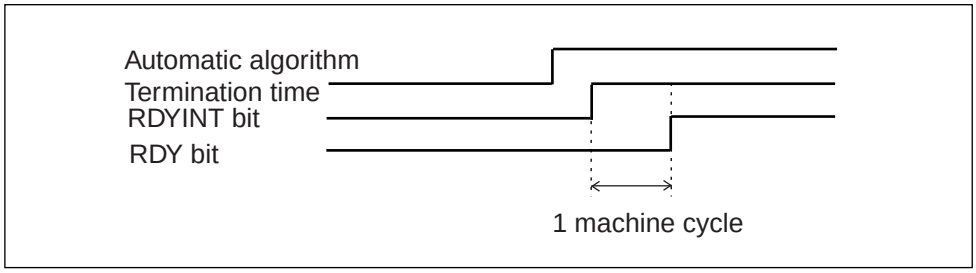


Table 19.3-1 Functional Explanation of Each Bits of the Flash Memory Control Status Register (FMCS)

Bit name		Function
bit7	INTE: Interrupt request enable bit	• This bit enables output of an interrupt request to the CPU when write/erase operation of flash memory terminates. • When the RDYINT bit is set to "1" while this bit is set to "1", an interrupt request is output. • Even if the RDYINT bit is set to "1", no interrupt request is output while this bit is "0".
bit6	RDYINT: write/erase operation termination flag bit	• When write/erase operation of flash memory terminates, this bit is set to "1", enabling write/erase operation to the flash memory. • When this bit is set to "0", it is cleared to "0", disabling write/erase operation to the flash memory. • When this bit is set to "1", operation is not affected. • This bit is also set to "1" when the automatic algorithm in flash memory terminates. (For details, see Section 19.4, "Method of Starting the Automatic Algorithm in Flash Memory.") • When a read-modify-write (RMW) instruction is used, "1" is always read.
bit5	WE: write/erase operation enable flag bit	• When this bit is "1", write/erase operation to the flash memory can be performed after executing a write/erase command sequence to bank FF. (For details, see Section 19.4, "Method of Starting the Automatic Algorithm in Flash Memory.") • When this bit is "0", no write/erase signal is generated even after executing a write/erase command sequence to bank FF. • This bit is set to "0" as its initial value, disabling operation. Before activating a write/erase command to the flash memory, be sure to set this bit to "1" to enable operation. **Note:** Set this bit to "0" if you do not want to perform write/erase operation.
bit4	RDY: Write/erase status bit	• While this bit is cleared to "0", it is not possible to perform write/erase operation to the flash memory. • Even if this bit is cleared to "0", the read/reset commands and suspend commands such as the selector erase/halt commands can be accepted. This bit is set to "1" when write/erase operation terminates.
bit3 bit1	Reserved: Reserved bit	• Always set this bit to "1".
bit2 bit0	LPM1,LPM0: Low-power consumption mode setting bit	• This bit is used to set the access time from the CPU to the flash memory. • The values that can be set vary depending on the internal operating frequency. • With decreasing internal operating frequency, power consumption by the flash memory unit can be reduced.

Note:

The operation termination flag bit (RDYINT) and the write/erase status bit (RDY) are not changed simultaneously. Write programs in such a way that whether the write/erase operation has terminated is determined by either of the two bits.


Automatic algorithm
Termination time
RDYINT bit
RDY bit
1 machine cycle

19.4 Starting the Flash Memory Automatic Algorithm

Four types of commands are available for starting the flash memory automatic algorithm: Read/Reset, Write, and Chip Erase. Control of suspend and restart is enabled for sector erase.

■ Command Sequence Table

All the data is written to the command register in units of bytes, though it should be accessed and written in units of words.

In this case, the data in the upper bytes is ignored.

Table 19.4-1 Command Sequence Table

Command sequence	Bus write cycle	1st bus write cycle		2 nd bus write cycle		3rd bus write cycle		4th bus write cycle		5th bus write cycle		6th bus write cycle	
		Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
Read/reset	1	$FFXXXX_H$	$XXF0_H$	-	-	-	-	-	-	-	-	-	-
Read/reset	4	$FFAAAA_H$	$XXAA_H$	$FF5554_H$	$XX55_H$	$FFAAAA_H$	$XXF0_H$	RA	RD	-	-	-	-
Write program	4	$FFAAAA_H$	$XXAA_H$	$FF5554_H$	$XX55_H$	$FFAAAA_H$	$XXA0_H$	PA (even)	PD (word)	-	-	-	-
Chip erase	6	$FFAAAA_H$	$XXAA_H$	$FF5554_H$	$XX55_H$	$FFAAAA_H$	$XX80_H$	$FFAAAA_H$	$XXAA_H$	$FF5554_H$	$XX55_H$	$FFAAAA_H$	$XX10_H$
Sector erase	6	$FFAAAA_H$	$XXAA_H$	$FF5554_H$	$XX55_H$	$FFAAAA_H$	$XX80_H$	$FFAAAA_H$	$XXAA_H$	$FF5554_H$	$XX55_H$	SA (even)	$XX30_H$
Sector erase suspend		Entering address "$FFXXXX_H$" data ($xxB0_H$) suspends erasing during sector erase.											
Sector erase restart		Entering address "$FFXXXX_H$" data ($xx30_H$) .restarts erasing after erasing is suspended during sector erase.											

* : Two types of read/reset commands can reset the flash memory to the read mode.

Note:

The addresses in the above table are the values on the CPU memory map. "X" is an optional value.

RA: Read address

PA: Write address. Only the even address can be specified.

SA: Sector address. For details, see Section 19.2 "512K-Bit Flash Memory Sector Configuration". The even address can be specified.

RD: Read data

PD: Write data. Only the word data can be specified.

19.5 Confirming the Automatic Algorithm Execution State

Flash memory contains a hardware sequence for checking the internal flash memory operating state because the automatic algorithm controls the write/erase flow.

■ Hardware Sequence Flag

The hardware sequence flag consists of the four flag bits, DQ7, DQ6, DQ5, and DQ3. These flag bits have the data polling flag (DQ7) function, toggle bit flag (DQ6) function, time limit exceeded flag (DQ5) function, and sector erase timer flag (DQ3) function, respectively. These functions can verify whether the write/chip sector erase is terminated or whether the erase code write is valid.

The hardware sequence flag can be referenced by accessing/reading the address of the target sector in the flash memory, after setting the command sequence (see Table 19.5-1 "Hardware Sequence Flag Bit Allocation") indicates the hardware sequence flag bit allocation.

Table 19.5-1 Hardware Sequence Flag Bit Allocation

Bit No.	7	6	5	4	3	2	1	0
Hardware sequence flag	DQ7	DQ6	DQ5	-	DQ3	-	-	-

Whether the automatic write algorithm or chip/sector erase algorithm is being performed or has terminated can be determined by checking the hardware sequence flags or the write/erase status bit (RDY) of the flash memory control register (FMCS). After the write/erase operation is terminated, the flash memory is returned to the read/reset status. When creating write/erase programs, perform processing such as data readout after verifying the completion of write/erase operation by using the hardware sequence flags (DQ7, DQ6, DQ5, and DQ3). Whether the second sector erase code write or a later one is valid can also be checked by using the hardware sequence flags.

Table 19.5-2 Hardware Sequence Flag Function List

State		DQ7	DQ6	DQ5	DQ3
Status transition during normal operation	Write operation--> Write completion (when the write address is specified)	$\overline{DQ7}$ --> DATA:7	Toggle --> DATA:6	0 --> DATA:5	0 --> DATA:3
	Chip/sector erase operation --> Erase completion	0 --> 1	Toggle --> Stop	0 --> 1	1
	Sector erase wait --> Erase start	0	Toggle	0	0 --> 1
	Erase processing --> Sector erase suspend (Sector being erased)	0 --> 1	Toggle --> 1	0	1 --> 0
	Sector erase suspend --> Erase restart (Sector being erased)	1 --> 0	1 --> Toggle	0	0 --> 1
	While the sector erase is being suspended --> Sector not being erased)	DATA:7	DATA:6	DATA	DATA:3
Abnormal operation	Write operation	$\overline{DQ7}$	Toggle	1	0
	Chip/sector erase operation	0	Toggle	1	1

19.5.1 Data Polling Flag (DQ7)

The data polling flag (DQ7) indicates whether the automatic algorithm is being executed or has been terminated, using the data polling function. Table 19.5-3 "Data Polling Flag Statous Transition" and Table 19.5-4 "Data Poling Flag Status Transition " shows the data polling flag status transition.

■ **When the Write Operation is Executed.**

If read access is made during execution of the automatic write algorithm, the data polling flag (DQ7) outputs the value that inverts data written last. If read access is made after the automatic write algorithm has terminated, DQ7 outputs the address to which the read access was made.

■ **When the Chip/Sector Erase Operation is Executed.**

If read access is made to the sector currently being erased during execution of the sector erase algorithm, the data polling flag (DQ7) outputs "0". When the sector erase is completed, the data polling flag (DQ7) outputs "1". If read access is made during execution of the chip erase algorithm, the data polling flag (DQ7) outputs "0". When the chip erase is completed, the data polling flag (DQ7) outputs "1".

■ **When the Sector Erase Suspend is Executed.**

If read access is made while sector erase is halted, the data polling flag (DQ7) outputs "1" if the address to which the read access was made is the sector currently being erased, and otherwise the value read during access (DATA: 7). By referring to this flag together with the toggle bit flag (DQ6), whether the sector erase is halted or which sector is currently being erased can be determined.

Note:

When the automatic algorithm is started, the read/access to the specified address is ignored. As for the data reading, the end of data polling flag (DQ7) is posted, and then other data bit can be output. Therefore, the data read operation after the end of the automatic algorithm should be executed next to the read/access after verifying the end of data polling flag.

Table 19.5-3 Data Polling Flag Status Transition - Status transition during normal operation

Operating status	Write operation -->Completion	Chip/sector erase -->Completion	Sector erase wait --> Start	Sector erase -->Erase suspend (Sector being erased)	Sector erase suspend -->Restart (Sector being erased)	During the sector erase suspend (Sector not being erased)
DQ7	$\overline{DQ7}$-->DATA:7	0-->1	0	0-->1	1-->0	DATA:7

Table 19.5-4 Data Poling Flag Status Transition - Status transition during abnormal operation

Operating status	Write operation	Chip/sector erase operation
DQ7	$\overline{DQ7}$	0

Table 19.5-3 "Data Polling Flag Status Transition" and Table 19.5-4 "Data Poling Flag Status Transition" are extracted from Table 19.5-2 "Hardware Sequence Flag Function List ".

19.5.2 Toggle Bit Flag (DQ6)

The toggle bit flag specifies whether the automatic algorithm is being executed or has been terminated, using the toggle bit function, the same as the data polling flag. Table 19.5-5 "Toggle Bit Flag Status Transition" and Table 19.5-6 "Toggle Bit Flag Status Transition" shows the toggle bit flag status transition.

■ **When the Write Operation or Chip/Sector Erase Operation is Executed.**

If continuous read access is made during execution of the automatic write algorithm or chip/sector erase algorithm is being performed, the toggle bit flag (DQ6) outputs "1" and "0" alternately (toggle output) each time read access is made. If continuous read access is made after the automatic write algorithm or chip/sector erase algorithm is completed, DQ6 outputs the value read each time read access is made.

■ **When the Sector Erase Suspend is Executed.**

If read access is made while sector erase is halted, the toggle bit flag (DQ6) outputs "1" if the read address is the sector currently being erased. Otherwise, the value read by the read access is output.

Reference:

If the sector to which data should be written is write-protected (sector protection), the toggle output from the toggle bit flag (DQ6) terminates after about 2 µs without rewriting data. If all sectors to be erased are write-protected (sector protection), DQ6 returns to the read/reset status after toggle output from DQ6 for about 100 µs without rewriting data.

Table 19.5-5 Toggle Bit Flag Status Transition - Status transition during normal operation

Operating status	Write operation -->Completion	Chip/sector erase -->Completion	Sector erase wait --> Start	Sector erase -->Erase suspend (Sector being erased)	Sector erase suspend -->Restart (Sector being erased)	During the sector erase suspend (Sector not being erased)
DQ6	Toggle -->DATA:6	Toggle-->Stop	Toggle	Toggle-->1	1-->Toggle	DATA:6

Table 19.5-6 Toggle Bit Flag Status Transition - Status transition during abnormal operation

Operating status	Write operation	Chip/sector erase operation
DQ6	Toggle	Toggle

Table 19.5-5 "Toggle Bit Flag Status Transition" and Table 19.5-6 "Toggle Bit Flag Status Transition" are extracted from Table 19.5-2 "Hardware Sequence Flag Function List ".

19.5.3 Time Limit Exceeded Flag (DQ5)

The time limit exceeded flag (DQ5) indicates that the automatic algorithm execution time has exceeded the time defined within the flash memory (i.e., internal pulse count). Table 19.5-7 "Transition of the Time Limit Exceeded Flag Status" and Table 19.5-8 "Transition of the Time Limit Exceeded Flag Status " shows the transition of the time limit exceeded flag status.

■ **When the Write Operation or Chip/Sector Erase Operation is Executed.**

If read access is made after the automatic write algorithm or chip/sector erase algorithm is activated, the timing limit exceeded flag (DQ5) outputs "0" if the execution time is within the defined time (time required for write/erase operation), and "1" if the defined time has passed. DQ5 can determine whether the write/erase operation is successful irrespective of whether the automatic algorithm is being executed or has been completed. When DQ5 outputs "1", it can be assumed that the write operation has failed if the automatic algorithm is being performed by the data polling function or toggle bit function.

For example, if an attempt is made to write "1" to the flash memory bit that contains "0", the flash memory is locked, the automatic algorithm does not terminate, and thus the data polling flag (DQ7) cannot output valid data. The toggle bit flag (DQ6) does not stop its toggle operation, exceeding the time limit and DQ5 outputs "1". "1" output from DQ5 does not indicate that the flash memory is defective. Rather, it indicates that the flash memory was not used correctly, so you need to execute the reset command.

Table 19.5-7 Transition of the Time Limit Exceeded Flag Status - Status transition during normal operation

Operating status	Write operation -->Completion	Chip/sector erase -->Completion	Sector erase wait --> Start	Sector erase -->Erase suspend (Sector being erased)	Sector erase suspend -->Restart (Sector being erased)	During the sector erase suspend (Sector not being erased)
DQ5	0-->DATA:5	0-->1	0	0	0	DATA:6

Table 19.5-8 Transition of the Time Limit Exceeded Flag Status - Status transition during abnormal operation

Operating status	Write operation	Chip/sector erase operation
DQ5	1	1

Table 19.5-7 "Transition of the Time Limit Exceeded Flag Status " and Table 19.5-8 "Transition of the Time Limit Exceeded Flag Status " are extracted from Table 19.5-2 "Hardware Sequence Flag Function List ".

19.5.4 Sector Erase Timer Flag (DQ3)

After starting the sector erase command, the sector erase timer flag (DQ3) indicates whether it is "during the sector erase waiting period." Table 19.5-9 "Sector Erase Timer Flag Status Transition" and Table 19.5-10 "Sector Erase Timer Flag Status Transition" shows the sector erase timer flag status transition.

■ **When the Sector Erase Operation is Executed.**

If read access is made after the sector erase command is activated, the sector erase timer flag (DQ3) outputs "0" if the execution time is within the sector erase wait period, and "1" if the sector erase wait period has passed.

When the erase algorithm is being executed by the data polling function or toggle bit function (DQ7="0", DQ6=toggle output), the sector erase is being performed if DQ3 outputs "1". Then, if a command other than the erase halt command is set, the command is ignored until the erase is terminated.

When DQ3 outputs "0", flash memory accepts write operation of sector erase code. Be sure to perform the write operation of the sector erase code after making sure that the output from DQ3 is "0". When DQ3 outputs "1", no sector erase command is accepted.

■ **When the Sector Erase Operation is Executed.**

If read access is made while the sector erase is halted, DQ3 outputs "1" if the read address is the sector being erased. Otherwise, the value read during access is output.

Table 19.5-9 Sector Erase Timer Flag Status Transition - Status transition during normal operation

Operating status	Write operation -->Completion	Chip/sector erase -->Completion	Sector erase wait --> Start	Sector erase -->Erase suspend (Sector being erased)	Sector erase suspend -->Restart (Sector being erased)	During the sector erase suspend (Sector not being erased)
DQ3	0-->DATA:3	1	0-->1	1-->0	0-->1	DATA:3

Table 19.5-10 Sector Erase Timer Flag Status Transition - Status transition during abnormal operation

Operating status	Write operation	Chip/sector erase operation
DQ3	0	1

Table 19.5-9 "Sector Erase Timer Flag Status Transition" and Table 19.5-10 "Sector Erase Timer Flag Status Transition" are extracted from Table 19.5-2 "Hardware Sequence Flag Function List ".

19.6 Detailed Explanation on the Flash Memory Write/Erase

This section explains the procedures for issuing the command to start the automatic algorithm, reading/resetting the flash memory, writing the data to the flash memory, erasing the chip, erasing the sector, suspending the sector erase, and restarting the sector erase.

■ Detailed Explanation on the Flash Memory Write/Erase

The read/reset, write, chip erase, sector erase, sector erase suspend, or erase start operation can be performed by the automatic algorithm which can be started by setting the command sequence (see Section 19.4 "Starting the Flash Memory Automatic Algorithm") to each bus write cycle. The write cycles to the respective buses must be executed continuously. The ending time of the automatic algorithm can be notified by the data polling function and so on. After the normal end of the automatic algorithm, the flash memory returns to the read/reset status.

Details of the operations of read/reset, write, chip erase, sector erase, sector erase suspend, and sector erase restart are explained from Section 19.6.1, "Setting the Write/Reset Status" to Section 19.6.6, "Restarting the Sector Erase".

19.6.1 Setting the Write/Reset Status

This section explains the procedure of issuing the read/reset command and setting the flash memory to the read/reset status.

■ Setting the Write/Reset Status

To set the flash memory to the read/reset status, send the read/reset command in the command sequence table (See Table 19.4-1 "Command Sequence Table") from the CPU to the flash memory continuously.

There are two types of read/reset command sequences. However, these command sequences have no essential difference.

The read/reset status is the initial status of the flash memory. When the power supply is turned on, or when the command is normally terminated, the flash memory is always set to the read/reset status. The read/reset status means the status of the flash memory that is waiting for another command to be input.

In the read/reset status, data can be read from the flash memory by executing a usual read/access command. The data can be program-accessed from CPU same as the mask ROM. This command is not required to make read access to the flash memory.

If the command has not terminated normally, use the read/reset command to initialize the automatic algorithm.

19.6.2 Writing the Data

This section explains the procedure of issuing the write command and writing the data to the flash memory. Figure 19.6-1 "Example of Procedure of Writing the Data to the Flash Memory" shows an example of procedure of writing data to the flash memory.

■ Writing the Data

To activate the automatic data write algorithm of the flash memory, send the write command in the command sequence table (See Table 19.4-1 "Command Sequence Table") from the CPU to the flash memory continuously. When the data write operation to the target address in the 4th cycle has been terminated, the automatic algorithm is started for automatic writing.

■ How to Specify the Address

Only even addresses can be specified as the write address in the write data cycle. If an odd address is specified, data cannot be correctly written. That is, it is necessary to write the data in units of words to even addresses.

Data can be written to the flash memory, and any address sequence may be specified, even if data has been written across the sector boundary. However, only one-word data can be written by executing the write command once.

■ Notes on Writing the Data

It is not possible to convert the bit data from "0" back into "1" by reading data. If the bit data "1" is written to the bit data "0", the data polling algorithm or toggle operation does not terminate and the flash memory elements are determined to be defective. Then, the time limit exceeded flag (DQ5) is set to "1" after the defined time for write operation has passed or the bit data "1" appears to have been written but is not actually done. If data is read in the read/reset status, the bit data read from the flash memory is "0". The erase operation is required to convert the bit data from "0" back to "1".

All commands are ignored while automatic writing is being performed. Note that the data at the address for writing is not assurred if the hardware is reset during automatic writing.

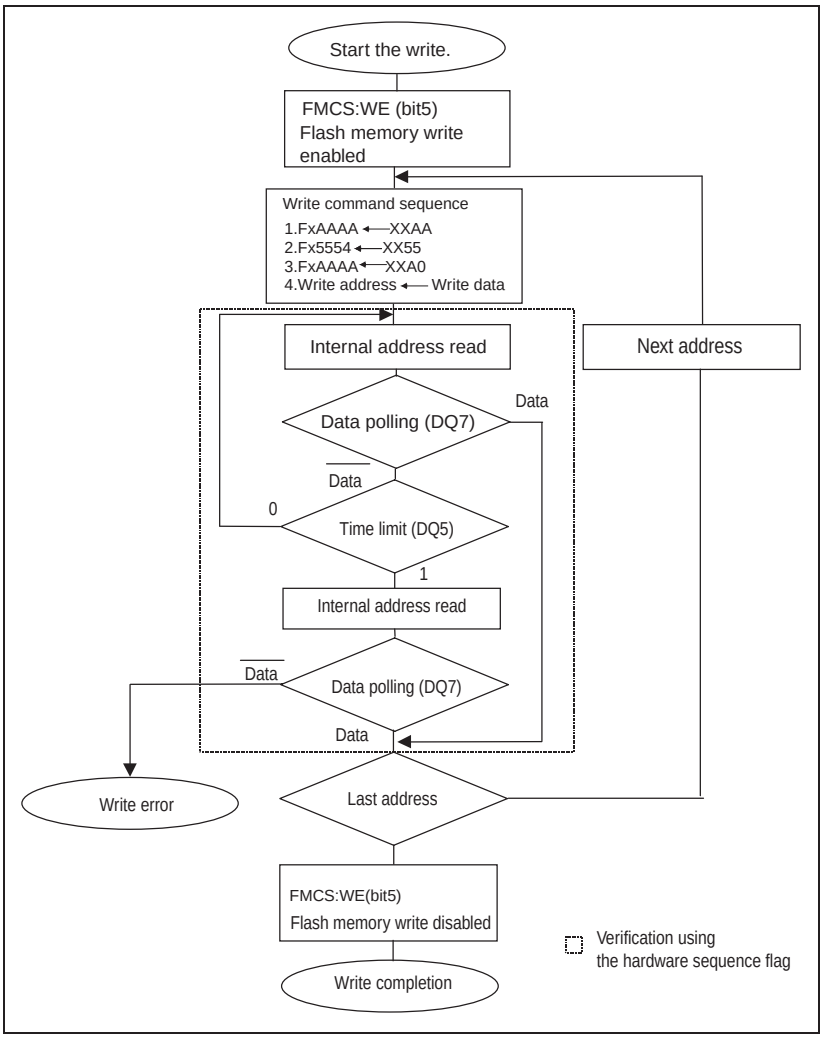
■ Procedure of Writing the Data to the Flash Memory

Using the hardware sequence flag (see Section 19.5 "Confirming the Automatic Algorithm Execution State"), the status of the automatic algorithm within the flash memory can be determined. Here, the data polling flag (DQ7) is used for verifying the end of writing.

The data reading for checking the flag is started from the last-written address. The data polling flag (DQ7) is changed at the same time when the time limit exceeded flag (DQ5) is changed, so the data polling flag (DQ7) must be rechecked even if the time limit exceeded flag (DQ5) is "1."

Similarly, the toggle bit flag (DQ6) stops the toggle operation at the same time when the time limit exceeded flag (DQ5) is changed to "1", so the toggle bit flag (DQ6) must be rechecked.

Figure 19.6-1 Example of Procedure of Writing the Data to the Flash Memory



19.6.3 Erasing the Data (Chip Erase)

This section explains the procedure of issuing the chip erase command and erasing all the data in the flash memory.

■ Erasing the data (Chip erase)

To erase all of the data from the flash memory, send the chip erase command in the command sequence table (See Table 19.4-1 "Command Sequence Table") from the CPU to the flash memory.

The chip erase command is executed by executing the bus operation six times. When the 6th-cycle write operation has been completed, the chip erase operation is started. The user need not write the data to the flash memory before chip erase operation. During the automatic erase algorithm execution, the flash memory writes data "0" to all the cells and verifies them before they are automatically erased.

19.6.4 Erasing the Data (Sector Erase)

This section explains the procedure of issuing the sector erase command and erasing any sector from the flash memory. This command enables each sector to be erased, and two or more sectors to be specified at the same time.

■ Erasing the data (Sector erase)

To erase an arbitrary sector from the flash memory, send the sector erase command in the command sequence table (See Table 19.4-1 "Command Sequence Table") from the CPU to the flash memory.

■ Method of Specifying a Sector

The sector erase command is executed by executing the bus operation six times. The sector erase wait of 50µs is started by writing the sector erase code (30_H) to any accessible even address in the target sector in the 6th-cycle bus operation. To erase two or more sectors, the erase code (30_H) should be written to the address in the target sector just after the above operation.

■ Notes on Specifying Two or More Sectors

The erase operation is started after the last sector erase code is written and the sector erase wait period of 50µs is terminated. Thus, the next erase sector address and erase code (i.e. in the 6th cycle of the command sequence) must be input each within 50µs to erase two or more sectors simultaneously, which may not be accepted later than 50µs. It can be checked whether the succeeding sector erase code write operation is valid, using the sector erase timer flag (DQ3). In this case, the address from which the sector erase timer flag (DQ3) is read must indicate the sector to be erased.

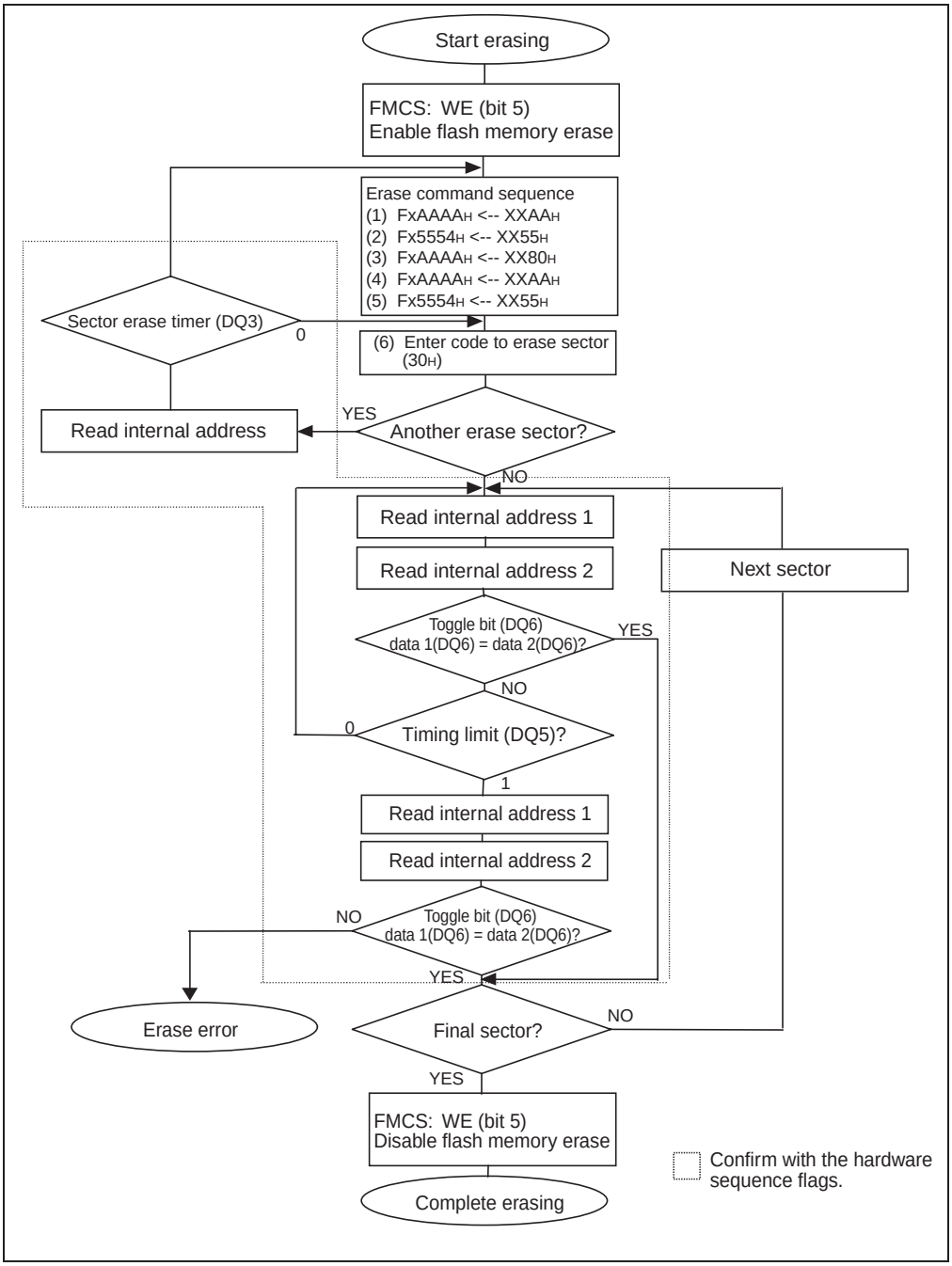
■ Procedure of Erasing a Sector

Using the hardware sequence flag (see Section 19.5 "Confirming the Automatic Algorithm Execution State"), the status of the automatic algorithm within the flash memory can be determined. Figure 19.6-2 "Example of Procedure of Erasing the Sector from the Flash Memory" shows an example of procedure of erasing the sector from the flash memory. Here, the toggle bit flag (DQ6) is used for verifying the end of writing. Note that data to be used for checking the flag is read from the sector to be erased.

The toggle bit flag (DQ6) stops the toggle operation at the same time when the time limit exceeded flag (DQ5) is changed to "1", so the toggle bit flag (DQ6) must be rechecked even if the time limit exceeded flag (DQ5) is "1."

Similarly, the data polling flag (DQ7) is changed at the same time when the time limit exceeded flag (DQ5) is changed, so the data polling flag (DQ7) must be rechecked.

Figure 19.6-2 Example of Procedure of Erasing the Sector from the Flash Memory



19.6.5 Suspending the Sector Erase

This section explains the procedure of issuing the sector erase suspend command and suspending the erase of a sector from the flash memory. This command can read the data from the sector not being erased.

■ Suspending the Sector Erase

To suspend the sector erase from the flash memory, send the sector erase suspend command in the command sequence table (See Table 19.4-1 "Command Sequence Table") from the CPU to the flash memory.

The sector erase suspend command suspends the sector erase operation, and enables the data reading from the sector not being erased. In this case, only the read operation can be executed, but the write operation cannot be done. This command is valid only during the sector erase operation including the erase waiting time, but it is ignored during the chip erase operation or the write operation.

This command is executed by writing the erase suspend code ($B0_H$). In this case, the address must indicate an address within the flash memory. During the erase suspend operation, the reissued erase suspend command is ignored.

If the sector erase suspend command is input during the sector erase waiting period, the sector erase wait is immediately terminated to stop the erase operation, and the flash memory enters the erase stop status. If the sector erase suspend command is input during the sector erase operation after the sector erase waiting period, the flash memory enters the erase suspend status after a lapse of up to 15µs.

19.6.6 Restarting the Sector Erase

This section explains the procedure of issuing the sector erase restart command and restarting the operation of erasing a sector from the flash memory, which has been suspended.

■ Restarting the Sector Erase

To restart the suspended sector erase, send the sector erase restart command in the command sequence table (See Table 19.4-1 "Command Sequence Table") from the CPU to the flash memory.

The sector erase restart command is used to restart the sector erase operation when the flash memory is in the sector erase suspend status caused by the sector erase suspend command. This command is executed by writing the erase restart code (30_H). In this case, the address must indicate an address within the flash memory area.

However, the sector erase restart command issued during the sector erase operation is ignored.

19.7 Programming Example of 512K-Bit Flash Memory

This section presents a programming example for the use of 512K-bit (64K bytes) flash memory.

■ Programming Example Using 512K-Bit Flash Memory

```
                NAME FLASHWE
                TITLE FLASHWE
;-------------------------------------------------------------------
;512k-FLASH  Sample program for 512-Kb FLASH
;
;1: Transfer the program from the flash memory (address:
    FFBC00H  sector: SA2) to RAM (address: 000700H).
;2: Run the transferred program in RAM.
;3: Write the value of PDR1 to the flash memory (address:
    FF0000H  sector: SA0).
;4: Read the written value (address: FF0000H  sector: SA0),
    and output the value to PDR2.
;5: Erase the sector (SA0) to which the value was written.
;6: Output a confirmation that the data was erased.
;  Requirements:
;               - Number of bytes transferred to RAM:  100 H (256 B)
;               - Determination that writing or erasing has ended
;                 Determined via DQ5 (timing limit exceeded flag)
;                 Determined via DQ6 (toggle bit flag)
;                 Determined via RDY (FMCS)
;               - Error handling
;                 Output Hi to P00 to P07.
;                 Issue a reset command.
;-------------------------------------------------------------------
;
RESOUS  IOSEG ABS=00            ;"RESOUS" I/O segment definition
        ORG   0000H
PDR0    RB    1
PDR1    RB    1
PDR2    RB    1
PDR3    RB    1
        ORG   0010H
DDR0    RB    1
DDR1    RB    1
DDR2    RB    1
DDR3    RB    1
        ORG   00A1H
CKSCR   RB    1
        ORG   00AEH
FMCS    RB    1
        ORG   006FH
ROMM    RB    1
RESOUS  ENDS
;
SSTA    SSEG
```

```
        RW    0127H
STA_T   RW    1
SSTA    ENDS
;
DATA    DSEG  ABS=0FFH          ;Flash command address
        ORG   5554H
COMADR2 RW    1
        ORG   0AAAAH
COMADR1 RW    1
DATA    ENDS
;////////////////////////////////////////////////////////////
;Main program (SA1)
;////////////////////////////////////////////////////////////
CODE    CSEG
START:
        ;////////////////////////////////////////////////////
        ;Initialization
        ;////////////////////////////////////////////////////
        MOV   CKSCR,#0BAH       ;Set a frequency multiplier of
three.
        MOV   RP,#0
        MOV   A,#!STA_T
        MOV   SSB,A
        MOVW  A,#STA_T
        MOVW  SP,A
        MOV   ROMM,#00H         ;Mirror OFF
        MOV   PDR0,#00H         ;For error check
        MOV   DDR0,#0FFH
        MOV   PDR1,#00H         ;Data input port
        MOV   DDR1,#00H
        MOV   PDR2,#00H         ;Data output port
        MOV   DDR2,#0FFH
        ;////////////////////////////////////////////////////////
        ;The flash memory write/erase program (FFBC00H) is
         transferred to RAM (1500H address).
        ;////////////////////////////////////////////////////////
        MOVW  A,#0700H          ;Destination RAM area
        MOVW  A,#0BC00H         ;Source address (location of
                                 program)
        MOVW  RW0,#100H         ;Number of bytes of data
                                 transferred to RAM
        MOVS  ADB,PCB           ;Transfer 100H from FFBC00H
                                 to 000700H.
        CALLP 000700H           ;Jump to the address where the
                                 transferred program exists.
        ;////////////////////////////////////////////////////
        ;Data output
        ;////////////////////////////////////////////////////
OUT     MOV   A,#0FEH
        MOV   ADB,A
        MOVW  RW2,#0000H
        MOVW  A,@RW2+00
        MOV   PDR2,A
END     JMP   *
CODE    ENDS
;////////////////////////////////////////////////////////////////
; Flash program write/erase program (SA2)
```

```
;////////////////////////////////////////////////////////////
RAMPRG  CSEG  ABS=0FFH
        ORG   0BC00H
;       ////////////////////////////////////////
;       Initialization
;       ////////////////////////////////////////
        MOVW  RW0,#0500H        ;RW0:Reserve RAM space for input
                                 area 00:0500 to
        MOVW  RW2,#0000H        ;RW2:Flash memory write address
                                 FF:0000 to
        MOV   A,#00H            ;DTB modification
        MOV   DTB,A             ;Bank specification for @RW0
        MOV   A,#0FFH           ;ADB modification 1
        MOV   ADB,A             ;Bank specification for write mode
                                 specification address
        MOV   PDR3,#00H         ;Switch initialization
        MOV   DDR3,#00H
;
WAIT1   BBC   PDR3:0,WAIT1      ;Writing starts if PDR3:0 Hi.
;
;////////////////////////////////////////////
; Write (SA0)
;////////////////////////////////////////////
        MOV   A,PDR1
        MOVW  @RW0+00,A         ;PDR1 data is stored in RAM.
        MOV   FMCS,#20H         ;Write mode setting
        MOVW  ADB:COMADR1,#00AAH;Flash write command 1
        MOVW  ADB:COMADR2,#0055H;Flash write command 2
        MOVW  ADB:COMADR1,#00A0H;Flash write command 3
;
        MOVW  A,@RW0+00         ;Input data (RW0) is written
                                 to flash memory (RW2).
        MOVW  @RW2+00,A
WRITE   ;Wait time check
;       ////////////////////////////////////////////////////////
;       If the "time-limit-exceeded" check flag is set while
        toggling is on, branches to ERROR.
;       ////////////////////////////////////////////////////////
        MOVW  A,@RW2+00
        AND   A,#20H            ;DQ5 time limit check
        BZ    NTOW              ;Time limit exceeded
        MOVW  A,@RW2+00         ;AH
        MOVW  A,@RW2+00         ;AL
        XORW  A                ;XOR of AH and AL (1 when the values
                                 are different.)
        AND   A,#40H            ;Checks whether the DQ6 toggle bit
                                 is different.
        BNZ   ERROR             ;If it is different, branches to
ERROR
;       ////////////////////////////////////////////
;       Write-end check (FMCS-RDY)
;       ////////////////////////////////////////////
NTOW    MOVW  A,FMCS
        AND   A,#10H            ;FMCS RDY bit (4 bit) is extracted.
        BZ    WRITE             ;Verifies that writing has ended.
        MOV   FMCS,#00H         ;Write mode is released.
        ;////////////////////////////////////////////////////
```

```
        ; Write data output
        ;////////////////////////////////////////////////
        MOVW  RW2,#0000H        ;Write data output
        MOVW  A,@RW2+00
        MOV   PDR2,A
;
WAIT2   BBC   PDR3:1,WAIT2      ;If PDR3:1 Hi, erasing of the sector
starts.
;
;//////////////////////////////////////////
; Erasing sectors (SA0)
;//////////////////////////////////////////
        MOVW  @RW2+00,#0000H    ;Address initialization
        MOV   FMCS,#20H         ;Erase mode setting
        MOVW  ADB:COMADR1,#00AAH;Flash memory erase command 1
        MOVW  ADB:COMADR2,#0055H;Flash memory erase command 2
        MOVW  ADB:COMADR1,#0080H;Flash memory erase command 3
        MOVW  ADB:COMADR1,#00AAH;Flash memory erase command 4
        MOVW  ADB:COMADR2,#0055H;Flash memory erase command 5
        MOVW  @RW2+00,#0030H    ;Issuing the erase command
                                 to the sector to be erased. 6
ELS     ;Wait-time check
;       ////////////////////////////////////////////////////////
;       When the "time-limit-exceeded" check flag is set and
        toggling is on, branches to ERROR.
;       When the "time-limit-exceeded" check flag is set and
        toggling is on, branches to ERROR.
;       ////////////////////////////////////////////////////////
        MOVW  A,@RW2+00
        AND   A,#20H            ;DQ5 time limit check
        BZ    NTOE              ;Time limit exceeded
        MOVW  A,@RW2+00         ;AH From DQ6 during writing
        MOVW  A,@RW2+00         ;AL Hi and Low are output
                                 alternately by each read.
        XORW  A                 ;XOR of AH and AL (If the
                                 DQ6 value is toggled,
                                 X OR is 1, indicating that
                                 writing is in progress.)
        AND   A,#40H            ;Checks whether the DQ6
                                 toggle bit is Hi.
        BNZ   ERROR             ;If the bit is Hi, branches to ERROR
; ////////////////////////////////////////////
; Erase-end check (FMCS-RDY)
; ////////////////////////////////////////////
NTOE    MOVW  A,FMCS             ;
        AND   A,#10H             ;FMCS RDY bit (4 bit) is extracted.
        BZ    ELS                ;Verifies that erasing of
                                  the sector has ended.
        MOV   FMCS,#00H          ;Flash memory erase mode is
                                  released.
        RETP                     ;Returns to the main program.
;//////////////////////////////////////////
; Error
;//////////////////////////////////////////
ERROR
        MOV   ADB:COMADR1,#0F0H  ;Reset command (enabling
                                  data reading)
```

```
        MOV   FMCS,#00H          ;Flash command mode is released.
        MOV   PDR0,#0FFH         ;Confirmation of error processing.
        RETP                     ;Returns to the main program.
RAMPRG  ENDS
;////////////////////////////////////////
VECT    CSEG  ABS=0FFH
        ORG   0FFDCH
        DSL   START
        DB    00H
VECT    ENDS
;
        END   START
```

CHAPTER 20 1M-BIT (128KB) FLASH MEMORY

This chapter describes the functions and operations of the 1M-bit (128KB) flash memory of the MB90F568.

- **Parallel programmer**
- **Serial dedicated programmer**
- **Write/erase operation by program execution**

This chapter provides an explanation for the above item "Write/erase operation by program execution."

20.1 Overview of the 1M-Bit Flash Memory

he 1M-bit flash memory is mapped to the FC to FF banks in the CPU memory map. The functions of the flash memory interface circuit enable read-access and program-access from the CPU in the same way as mask ROM. Instructions from the CPU can be used via the flash memory interface circuit to write data to and erase data from the flash memory. Internal CPU control therefore enables rewriting of the flash memory while it is mounted. As a result, improvements in programs and data can be performed efficiently.
However, no selector operation such as the enable sector protect can be used.

■ Characteristics of the 1M-Bit Flash Memory

- Sectors of 128K words x 8/32K words x 16 bits (16K+8K+8K+32K+64K)
- Use of automatic program algorithm (Embedded Algorithm: Equivalent to MBM29F400TA)
- Erase pause/restart function provided
- Detection of completion of writing/erasing using data polling or toggle bit functions
- Detection of completion of writing/erasing using CPU interrupts
- Compatibility with the JEDEC standard-type command
- Sector erase function (any combination of sectors)
- Minimum of 10,000 write/erase operations

"Embedded Algorithm" is the trademark of Advanced Micro Device.

■ Writing to/Erasing Flash Memory

The flash memory cannot be written to and read at the same time. That is, when data is written to or erased from the flash memory, the program in the flash memory must first be copied to RAM. The entire process is then executed in RAM so that data is simply written to the flash memory. This eliminates the need for the program to access the flash memory from the flash memory itself.

■ Register on the Flash Memory

❍ Flash memory control status register (FMCS)

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: $0000AE_H$	INTE	RDYINT	WE	RDY	Reserved	-	-	LPM	FMCS
Read/write ⇒	(R/W)	(R/W)	(R/W)	(R)	(W)	(W)	(W)	(R/W)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(-)	(-)	(0)	

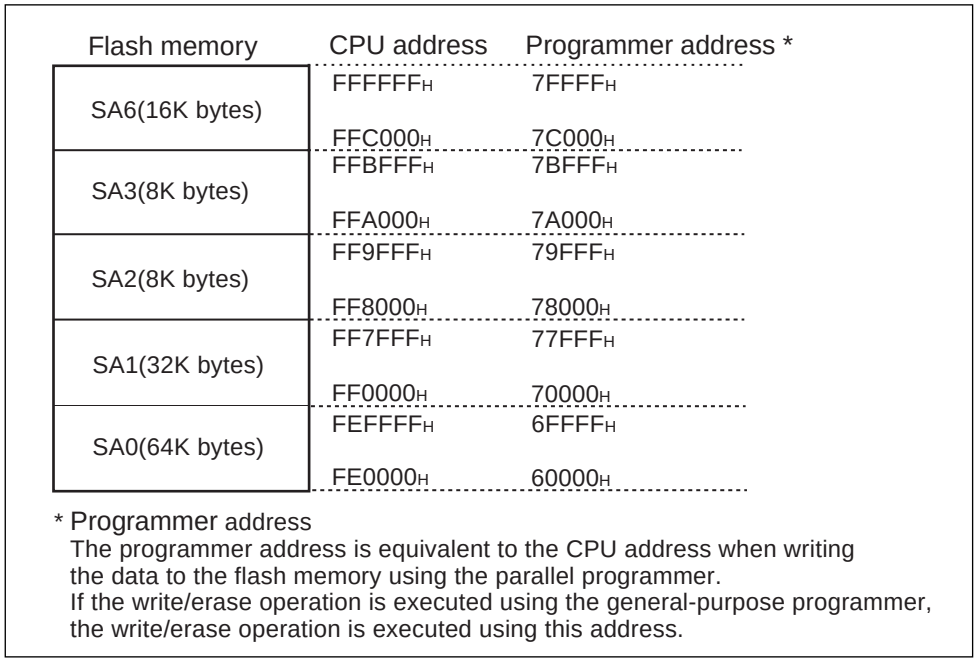
20.2 1M-Bit Flash Memory Sector Configuration

Figure 20.2-1 "1M-Bit (128KB) Flash Memory Sector Configuration" shows the sector configuration in the 1M-bit flash memory. The address indicated in Figure 20.2-1 "1M-Bit Flash Memory Sector Configuration" is classified into the upper address and lower address of each sector.

■ Sector Configuration

For access from the CPU, SA0 is allocated to the FE bank register and SA1 to SA6 are allocated to the FF bank register.

Figure 20.2-1 1M-Bit (128KB) Flash Memory Sector Configuration



* Programmer address
The programmer address is equivalent to the CPU address when writing the data to the flash memory using the parallel programmer.
If the write/erase operation is executed using the general-purpose programmer, the write/erase operation is executed using this address.

20.3 Flash Memory Control Status Register (FMCS)

The flash memory control status register (FMCS), together with the flash memory interface circuit, is used to write data to and erase data from the flash memory.

■ Flash Memory Control Status Register (FMCS)

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: $0000AE_H$	INTE	RDYINT	WE	RDY	Reserved	-	-	LPM	FMCS
Read/write ⇒	(R/W)	(R/W)	(R/W)	(R)	(W)	(W)	(W)	(R/W)	
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(-)	(-)	(0)	

❍ Explanation of Bits

[Bit 7] INTE (interrupt enable)

This bit generates an interrupt to the CPU when flash memory write/erase terminates.

An interrupt to the CPU is generated when the INTE and RDYINT bits are 1. No interrupt is generated when the INTE bit is 0.

0: Disables interrupts when write/erase terminates.

1: Enables interrupts when write/erase terminates.

[Bit 6] RDYINT (ready interrupt)

This bit indicates the operating state of the flash memory.

This bit is set to 1 when flash memory write/erase terminates. Data cannot be written to or erased from the flash memory while this bit is 0 after a flash memory write/erase. Flash memory write/erase is enabled when write/erase terminates and this bit is set to 1.

Writing 0 clears this bit to 0. Writing 1 is ignored. This bit is set to 1 at the termination timing of the flash memory automatic algorithm (see Section 20.4 "Starting the Flash Memory Automatic Algorithm"). When the read-modify-write (RMW) instruction is used, 1 is always read.

0: Write/erase is being executed.

1: Write/erase has terminated (interrupt request generated).

[Bit 5] WE (write enable)

This bit enables writing to the flash memory area.

When this bit is 1, writing after the command sequence (see Section 20.4 "Starting the Flash Memory Automatic Algorithm") is issued to the FC to FF bank writes to the flash memory area. When this bit is 0, the write/erase signal is not generated. This bit is used when the flash memory write/erase command is started.

If write/erase is not performed, it is recommended that this bit be set to 0 to prevent data from being mistakenly written to the flash memory.

0: Disables flash memory write/erase.

1: Enables flash memory write/erase.

[Bit 4] RDY (ready)

This bit enables flash memory write/erase.

Flash memory write/erase is disabled while this bit is 0. However, Suspend commands, such as the Read/Reset command and Sector Erase Suspend command, can be accepted even if this bit is 0.

0: Write/erase is being executed.

1: Write/erase has terminated (next data write/erase enabled).

[Bit 3] Reserved bit

These bits are reserved for testing. During regular use, they should always be set to 0.

[Bits 2 and 1] Free bits

During regular use, they should always be set to 0.

[Bit 0] LPM (low power mode)

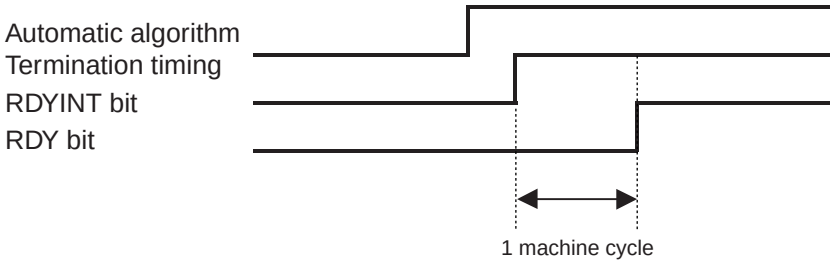
Setting LPM bit to "1" minimizes the select signal to flash memory when flash memory is accessed, thereby suppressing the power consumption of the main unit of flash memory. However, because the access time when this bit is 1 is considerably longer than the access time when this bit is 0, flash memory access is impossible when the CPU operates at high speed. Operate the CPU at a frequency of 4 MHz or lower when using this mode.

0: Normal power consumption mode

1: Low-power consumption mode (The CPU operates at an internal operating frequency of 4 MHz or lower.)

Note:

The RDYINT and RDY bits cannot be changed at the same time. Create a program so that decisions are made using one or the other of these bits.



20.4 Starting the Flash Memory Automatic Algorithm

Four types of commands are available for starting the flash memory automatic algorithm: Read/Reset, Write, and Chip Erase. Control of suspend and restart is enabled for sector erase.

■ Command Sequence Table

Table 20.4-1 "Command Sequence Table" lists the commands used for flash memory write/erase. All of the data written to the command register is in bytes, but use word access to write. The data in the high-order bytes at this time is ignored.

Table 20.4-1 Command Sequence Table

Command sequence	Bus write access	1st bus write cycle		2nd bus write cycle		3rd bus write cycle		4th bus write cycle		5th bus write cycle		6th bus write cycle	
		Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
Read/Reset (*1)	1	FxXXXX	XXF0	-	-	-	-	-	-	-	-	-	-
Read/Reset (*1)	4	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XXF0	RA	RD	-	-	-	-
Write program	4	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XXA0	PA (even)	PD (word)	-	-	-	-
Chip Erase	6	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX80	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX10
Sector Erase	6	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX80	FxAAAA	XXAA	Fx5554	XX55	SA (even)	XX30
Sector Erase Suspend		Entering address FxXXXX data (xxB0H) suspends erasing during sector erase.											
Sector Erase Restart		Entering address FxXXXX data (xx30H) restarts erasing after erasing is suspended during sector erase.											

Note:

- The addresses Fx in the table mean FF and FE. Use these addresses as the access target bank values for operations.
- The addresses in the table are the values in the CPU memory map. All addresses and data are represented using hexadecimal notation. However, the letter X is an optional value.
- RA: Read address
- PA: Write address. Only even addresses can be specified.
- SA: Sector address. See Section 20.2 "1M-Bit Flash Memory Sector Configuration".
- RD: Read data

PD: Write data. Only word data can be specified.

*1 Both of the two types of Read/Reset commands can reset the flash memory to read mode.

20.5 Confirming the Automatic Algorithm Execution State

Because the write/erase flow of the flash memory is controlled using the automatic algorithm, the flash memory has hardware for posting its internal operating state and completion of operation. This automatic algorithm enables confirmation of the operating state of the built-in flash memory using the following hardware sequences.

■ Hardware Sequence Flags

The hardware sequence flags are configured from the five-bit output of DQ7, DQ6, DQ5, and DQ3. The functions of these bits are those of the data polling flag (DQ7), toggle bit flag (DQ6), timing limit exceeded flag (DQ5), and sector erase timer flag (DQ3). The hardware sequence flags can therefore be used to confirm that writing or chip sector erase has been completed or that erase code write is valid.

The hardware sequence flags can be accessed by read-accessing the addresses of the target sectors in the flash memory after setting of the command sequence (see Table 20.4-1 "Command Sequence Table" in Section 20.4 "Starting the Flash Memory Automatic Algorithm"). Table 20.5-1 "Bit Assignments of Hardware Sequence Flags" lists the bit assignments of the hardware sequence flags.

Table 20.5-1 Bit Assignments of Hardware Sequence Flags

Bit No.	7	6	5	4	3	2	1	0
Hardware sequence flag	DQ7	DQ6	DQ5	-	DQ3	-	-	-

To determine whether automatic writing or chip sector erase is being executed, the hardware sequence flags can be checked or the status can be determined from the RDY bit of the flash memory control register (FMCS) that indicates whether writing has been completed. After writing/erasing has terminated, the state returns to the read/reset state. When creating a program, use one of the flags to confirm that automatic writing/erasing has terminated. Then, perform the next processing operation, such as data read. In addition, the hardware sequence flags can be used to confirm whether a second or subsequent sector erase code write is valid. The following sections describe each hardware sequence flag separately. Table 20.5-2 "Hardware Sequence Flag Functions" lists the functions of the hardware sequence flags.

Table 20.5-2 Hardware Sequence Flag Functions

State		DQ7	DQ6	DQ5	DQ3
State change for normal operation	Write --> Write completed (write address specified)	$\overline{DQ7}$ --> DATA:7	Toggle --> DATA:6	0 --> DATA:5	0 --> DATA:3
	Chip/sector erase --> Erase completed	0 --> 1	Toggle --> Stop	0 --> 1	1
	Sector erase wait --> Erase started	0	Toggle	0	0 --> 1
	Erase --> Sector erase suspended (sector being erased)	0 --> 1	Toggle --> 1	0	1 --> 0
	Sector erase suspend --> Erase restarted (sector being erased)	1 --> 0	1 --> Toggle	0	0 --> 1
	Sector erase suspended (sector not being erased)	DATA:7	DATA:6	DATA:5	DATA:3
Abnormal operation	Write	$\overline{DQ7}$	Toggle	1	0
	Chip/sector erase	0	Toggle	1	1

20.5.1 Data Polling Flag (DQ7)

The data polling flag (DQ7) uses the data polling function to post that the automatic algorithm is being executed or has terminated.

■ Data Polling Flag (DQ7)

Table 20.5-3 "Data Polling Flag State Transitions (state change for normal operation)" and Table 20.5-4 "Data Polling Flag State Transitions (state change for abnormal operation)" list the state transitions of the data polling flag.

Table 20.5-3 Data Polling Flag State Transitions (state change for normal operation)

Operating state	Write --> Completed	Chip/sector erase --> Completed	Sector erase wait --> Started	Sector erase --> Erase suspend (sector being erased)	Sector erase suspend --> Restarted (sector being erased)	Sector erase suspended (sector not being erased)
DQ7	$\overline{DQ7}$ --> DATA:7	0 --> 1	0	0 --> 1	1 --> 0	DATA:7

Table 20.5-4 Data Polling Flag State Transitions (state change for abnormal operation)

Operating state	Write	Chip/sector erase
DQ7	$\overline{DQ7}$	0

❍ Write

Read-access during execution of the automatic write algorithm causes the flash memory to output the opposite data of bit 7 last written, regardless of the value at the address specified by the address signal. Read-access at the end of the automatic write algorithm causes the flash memory to output bit 7 of the read value of the address specified by the address signal.

❍ Chip/sector erase

For a sector erase, read-access during execution of the chip erase/sector erase algorithm causes the flash memory to output 0 from the sector currently being erased. For a chip erase, read-access causes the flash memory to output 0 regardless of the value at the address specified by the address signal. Read-access at the end of the automatic write algorithm causes the flash memory to output 1 in the same way.

❍ **Sector erase suspend**

Read-access for an access from sector erase suspend causes the flash memory to output 1 if the address specified by the address signal belongs to the sector being erased. The flash memory outputs bit 7 (DATA: 7) of the read value at the address specified by the address signal if the address specified by the address signal does not belong to the sector being erased. Referencing this flag together with the toggle bit flag (DQ6) enables a decision to be made on whether the flash memory is in the erase suspended state and which sector is being erased.

Note:

When the automatic algorithm is being started, read-access to the specified address is ignored. Since termination of the data polling flag (DQ7) can be accepted for a data read and other bits output, data read after the automatic algorithm has terminated should be performed after read-access has confirmed that data polling has terminated.

20.5.2 Toggle Bit Flag (DQ6)

Like the data polling flag, the toggle bit flag (DQ6) uses the toggle bit function to post that the automatic algorithm is being executed or has terminated.

■ Toggle Bit Flag (DQ6)

Table 20.5-5 "Toggle Bit Flag State Transitions (state change for normal operation)" and Table 20.5-6 "Toggle Bit Flag State Transitions (state change for abnormal operation)" list the state transitions of the toggle bit flag.

Table 20.5-5 Toggle Bit Flag State Transitions (state change for normal operation)

Operating state	Write --> Completed	Chip/sector erase --> Completed	Sector erase wait --> Started	Sector erase --> Erase suspend (sector being erased)	Sector erase suspend --> Restarted (sector being erased)	Sector erase suspended (sector not being erased)
DQ6	Toggle --> DATA:6	Toggle --> Stop	Toggle	Toggle --> 1	1 --> Toggle	DATA:6

Table 20.5-6 Toggle Bit Flag State Transitions (state change for abnormal operation)

Operating state	Write	Chip/sector erase
DQ6	Toggle	Toggle

❍ Write/chip sector erase

Continuous read-access during execution of the automatic write algorithm and chip/sector erase algorithm causes the flash memory to toggle the 1 or 0 state for every read cycle, regardless of the value at the address specified by the address signal. Continuous read-access at the end of the automatic write algorithm and chip/sector erase algorithm causes the flash memory to stop toggling bit 6 and output bit 6 (DATA: 6) of the read value of the address specified by the address signal.

❍ Sector erase suspend

Read-access during a sector erase suspend causes the flash memory to output 1 if the address specified by the address signal belongs to the sector being erased. The flash memory outputs bit 6 (DATA: 6) of the read value at the address specified by the address signal if the address specified by the address signal does not belong to the sector being erased.

Note:

For a write, if the sector where data is to be written is rewrite-protected, the toggle bit terminates the toggle operation after approximately 2 μs and without any data being rewritten.

For an erase, if all of the selected sectors are write-protected, the toggle bit performs toggling for approximately 100 μs and then returns to the read/reset state without any data being rewritten.

20.5.3 Timing Limit Exceeded Flag (DQ5)

The timing limit exceeded flag (DQ5) is used to post that execution of the automatic algorithm has exceeded the time (internal pulse count) prescribed in the flash memory.

■ Timing Limit Exceeded Flag (DQ5)

Table 20.5-7 "Timing Limit Exceeded Flag State Transitions (state change for normal operation)" and Table 20.5-8 "Timing Limit Exceeded Bit Flag State Transitions (state change for abnormal operation)" list the state transitions of the timing limit exceeded flag.

Table 20.5-7 Timing Limit Exceeded Flag State Transitions (state change for normal operation)

Operating state	Write --> Completed	Chip/sector erase --> Completed	Sector erase wait --> Started	Sector erase --> Erase suspend (sector being erased)	Sector erase suspend --> Restarted (sector being erased)	Sector erase suspended (sector not being erased)
DQ5	0 --> DATA:5	0 --> 1	0	0	0	DATA:5

Table 20.5-8 Timing Limit Exceeded Bit Flag State Transitions (state change for abnormal operation)

Operating state	Write	Chip/sector erase
DQ5	1	1

❍ Write/chip sector erase

Read-access after write or chip/sector erase automatic algorithm activation causes the flash memory to output 0 if the time is within the prescribed time (time required for write/erase) or to output 1 if the prescribed time has been exceeded. Because this is done regardless of whether the automatic algorithm is being executed or has terminated, it is possible to determine whether write/erase was successful or unsuccessful. That is, when this flag outputs 1, writing can be determined to have been unsuccessful if the automatic algorithm is still being executed by the data polling function or toggle bit function.

For example, writing 1 to a flash memory address where 0 has been written will cause the fail state to occur. In this case, the flash memory will lock and execution of the automatic algorithm will not terminate. As a result, valid data will not be output from the data polling flag (DQ7). In addition, the toggle bit flag (DQ6) will exceed the time limit without stopping the toggle operation and the timing limit exceeded flag (DQ5) will output 1. Note that this state indicates that the flash memory is not faulty, but has been used correctly. When this state occurs, execute the Reset command.

20.5.4 Sector Erase Timer Flag (DQ3)

The sector erase timer flag (DQ3) is used to post whether the automatic algorithm is being executed during the sector erase wait period after the Sector Erase command has been started.

■ **Sector Erase Timer Flag (DQ3)**

Table 20.5-9 "Sector Erase Timer Flag State Transitions (state change for normal operation)" and Table 20.5-10 "Sector Erase Timer Flag State Transitions (state change for abnormal operation)" list the state transitions of the sector erase timer flag.

Table 20.5-9 Sector Erase Timer Flag State Transitions (state change for normal operation)

Operating state	Write --> Completed	Chip/sector erase --> Completed	Sector erase wait --> Started	Sector erase --> Erase suspend (sector being erased)	Sector erase suspend --> Restarted (sector being erased)	Sector erase suspended (sector not being erased)
DQ3	0 --> DATA:3	1	0 --> 1	1 --> 0	0 --> 1	DATA:3

Table 20.5-10 Sector Erase Timer Flag State Transitions (state change for abnormal operation)

Operating state	Write	Chip/sector erase
DQ3	0	1

❍ **Sector erase**

Read-access after the Sector Erase command has been started causes the flash memory to output 0 if the automatic algorithm is being executed during the sector erase wait period, regardless of the value at the address specified by the address signal of the sector that issued the command. The flash memory outputs 1 if the sector erase wait period has been exceeded.

When the data polling function or toggle bit function indicates that the erase algorithm is being executed, internally controlled erase has already started if this flag is 1. Continuous write of the sector erase codes or commands other than the Sector Erase Suspend command will be ignored until erase is terminated.

If this flag is 0, the flash memory will accept write of additional sector erase codes. To confirm this, it is recommended that the state of this flag be checked before continuing to write sector erase codes. If this flag is 1 after the second state check, it is possible that additional sector erase codes may not be accepted.

❍ **Sector erase**

Read-access during execution of sector erase suspend causes the flash memory to output 1 if the address specified by the address signal belongs to the sector being erased. The flash memory outputs bit 3 (DATA: 3) of the read value of the address specified by the address signal if the address specified by the address signal does not belong to the sector being erased.

20.6 Detailed Explanation of Writing to and Erasing Flash Memory

This section describes each operation procedure of flash memory Read/Reset, Write, Chip Erase, Sector Erase, Sector Erase Suspend, and Sector Erase Restart when a command that starts the automatic algorithm is issued.

■ Detailed Explanation of Flash Memory Write/erase

The flash memory executes the automatic algorithm by issuing a command sequence (see Table 20.4-1 "Command Sequence Table" in Section 20.4 "Starting the Flash Memory Automatic Algorithm") for a write cycle to the bus to perform Read/Reset, Write, Chip Erase, Sector Erase, Sector Erase Suspend, or Sector Erase Restart operations. Each bus write cycle must be performed continuously. In addition, whether the automatic algorithm has terminated can be determined using the data polling or other function. At normal termination, the flash memory is returned to the read/reset state.

Each operation of the flash memory is described in the following order:

- Setting the read/reset state
- Writing data
- Erasing all data (erasing chips)
- Erasing optional data (erasing sectors)
- Suspending sector erase
- Restarting sector erase

20.6.1 Setting Flash Memory to the Read/reset State

This section describes the procedure for issuing the Read/Reset command to set the flash memory to the read/reset state.

■ Setting the Flash Memory to the Read/reset State

The flash memory can be set to the read/reset state by sending the Read/Reset command in the command sequence table (see Table 20.4-1 "Command Sequence Table" in Section 20.4 "Starting the Flash Memory Automatic Algorithm") continuously to the target sector in the flash memory.

The Read/Reset command has two types of command sequences that execute the first and third bus operations. However, there are no essential differences between these command sequences.

The read/reset state is the initial state of the flash memory. When power is supplied on and when a command terminates normally, the flash memory is set to the read/reset state. In the read/reset state, other commands wait for input.

In the read/reset state, data is read by regular read-access. As with the mask ROM, program access from the CPU is enabled. The Read/Reset command is not required to read data by a regular read. The Read/Reset command is mainly used to initialize the automatic algorithm in such cases as when a command does not terminate normally.

20.6.2 Writing Data to Flash Memory

This section describes the procedure for issuing the Write command to write data to the flash memory.

■ Writing Data to the Flash Memory

The data write automatic algorithm of the flash memory can be started by sending the Write command in the command sequence table (see Table 20.4-1 "Command Sequence Table" in Section 20.4 "Starting the Flash Memory Automatic Algorithm") continuously to the target sector in the flash memory. When data write to the target address is completed in the fourth cycle, the automatic algorithm and automatic write are started.

❍ Specifying addresses

Only even addresses can be specified as the write addresses specified in a write data cycle. Odd addresses cannot be written correctly. That is, writing to even addresses must be done in units of word data.

Writing can be done in any order of addresses or even if the sector boundary is exceeded. However, the Write command writes only data of one word for each execution.

❍ Notes on writing data

Writing cannot return data 0 to data 1. When data 1 is written to data 0, the data polling algorithm (DQ7) or toggle operation (DQ6) does not terminate and the flash memory elements are determined to be faulty. If the time prescribed for writing is thus exceeded, the timing limit exceeded flag (DQ5) is determined to be an error. Otherwise, the data is viewed as if dummy data 1 had been written. However, when data is read in the read/reset state, the data remains 0. Data 0 can be set to data 1 only by erase operations.

All commands are ignored during execution of the automatic write algorithm. If a hardware reset is started during writing, the data of the written addresses will be unpredictable.

■ Writing to the Flash Memory

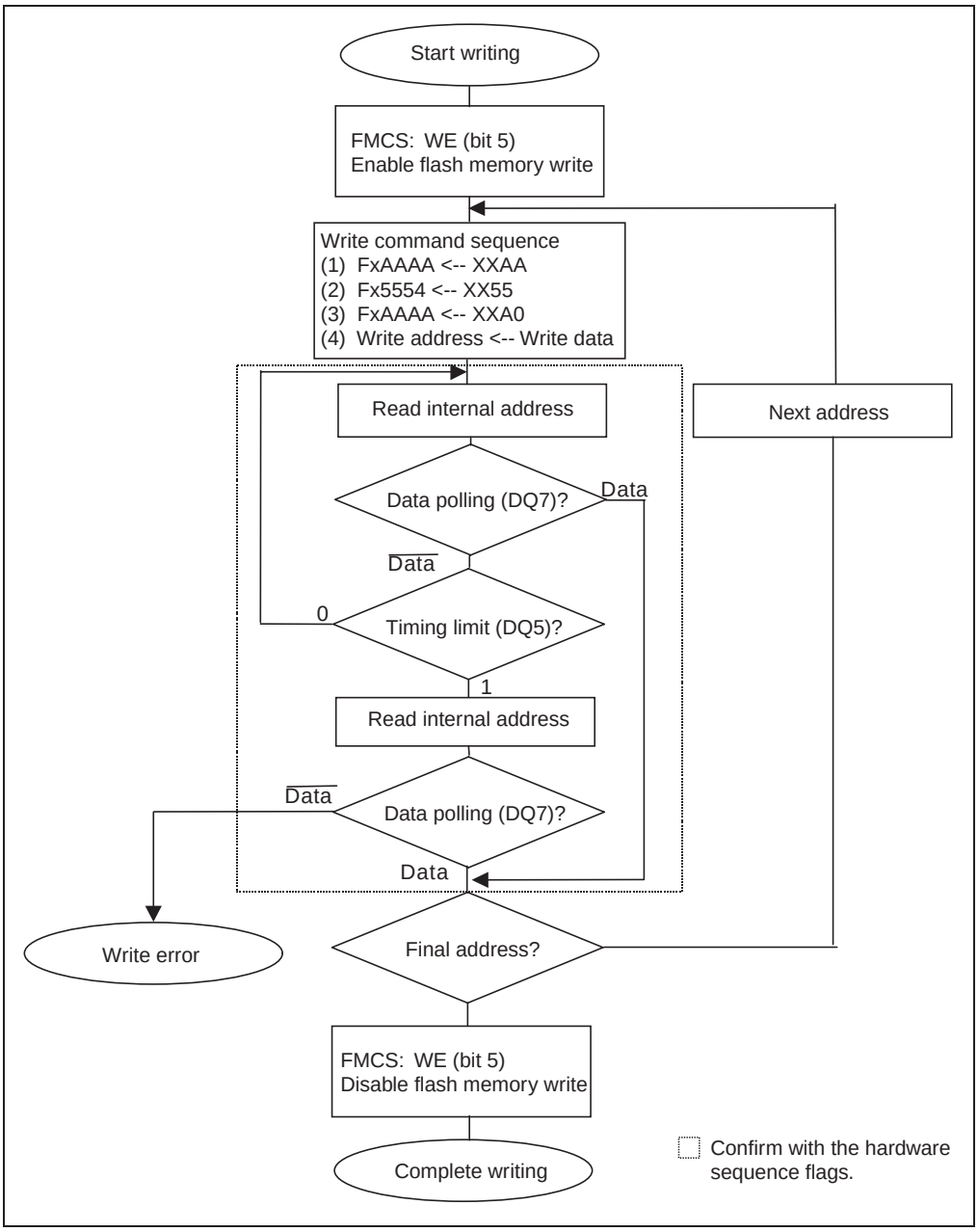
Figure 20.6-1 "Example of the Flash Memory Write Procedure" is an example of the procedure for writing to the flash memory. The hardware sequence flags (see Section 20.5 "Confirming the Automatic Algorithm Execution State") can be used to determine the state of the automatic algorithm in the flash memory. Here, the data polling flag (DQ7) is used to confirm that writing has terminated.

The data read to check the flag is read from the address written to last.

The data polling flag (DQ7) changes at the same time that the timing limit exceeded flag (DQ5) changes. For example, even if the timing limit exceeded flag (DQ5) is 1, the data polling flag bit (DQ7) must be rechecked.

Also for the toggle bit flag (DQ6), the toggle operation stops at the same time that the timing limit exceeded flag bit (DQ5) changes to 1. The toggle bit flag (DQ6) must therefore be rechecked.

Figure 20.6-1 Example of the Flash Memory Write Procedure



20.6.3 Erasing all data (erasing chips) of flash memory

This section describes the procedure for issuing the Chip Erase command to erase all data in the flash memory.

■ Erasing All Data in the Flash Memory (erasing chips)

All data can be erased from the flash memory by sending the Chip Erase command in the command sequence table (see Table 20.4-1 "Command Sequence Table" in Section 20.4 "Starting the Flash Memory Automatic Algorithm") continuously to the target sector in the flash memory.

The Chip Erase command is executed in six bus operations. When writing of the sixth cycle is completed, the chip erase operation is started. For chip erase, the user need not write to the flash memory before erasing. During execution of the automatic erase algorithm, the flash memory writes 0 for verification before all of the cells are erased automatically.

20.6.4 Erasing Optional Data (erasing sectors) in Flash Memory

This section describes the procedure for issuing the Sector Erase command to erase optional data (erase sector) in the flash memory. Individual sectors can be erased. Multiple sectors can also be specified at one time.

■ Erasing Optional Data (erasing sectors) in the Flash Memory

Optional sectors in the flash memory can be erased by sending the Sector Erase command in the command sequence table (see Table 20.4-1 "Command Sequence Table" in Section 20.4 "Starting the Flash Memory Automatic Algorithm") continuously to the target sector in the flash memory.

❍ Specifying sectors

The Sector Erase command is executed in six bus operations. Sector erase wait of 50 μs is started by writing the sector erase code (30H) to an accessible even-numbered address in the target sector in the sixth cycle. To erase multiple sectors, write the erase code (30H) to the addresses in the target sectors after the above processing operation.

❍ Notes on specifying multiple sectors

Erase is started when the sector erase wait period of 50 μs terminates after the final sector erase code has been written. That is, to erase multiple sectors at one time, an erase code (sixth cycle of the command sequence) must be written within 50 μs of writing of the address of a sector and the address of the next sector must be written within 50 μs of writing of the previous erase code. Otherwise, the address and erase code may not be accepted. The sector erase timer (hardware sequence flag DQ3) can be used to check whether writing of the subsequent sector erase code is valid. At this time, specify so that the address used for reading the sector erase timer indicates the sector to be erased.

■ Erasing Sectors in the Flash Memory

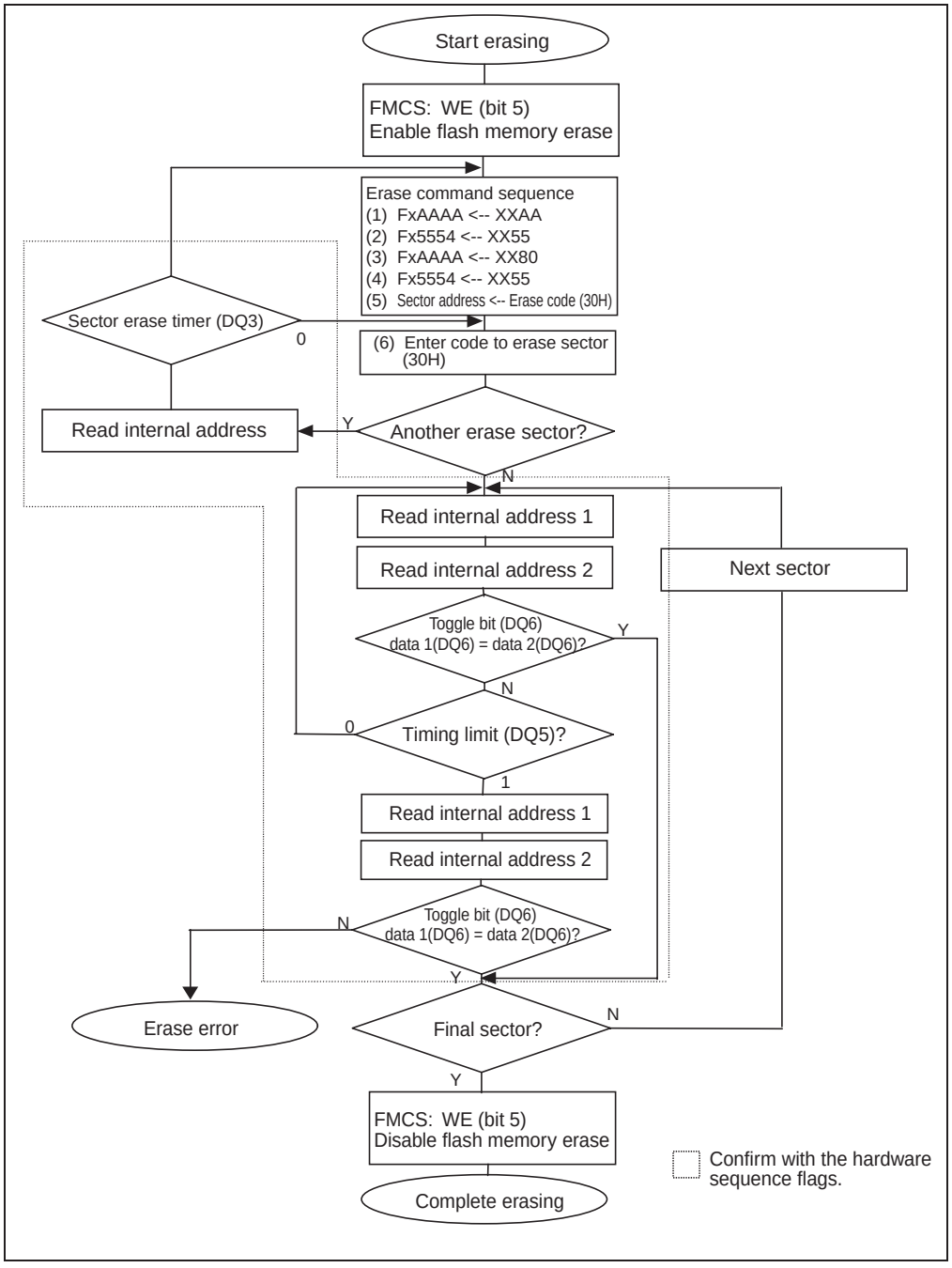
The hardware sequence flags (see Section 20.5 "Confirming the Automatic Algorithm Execution State") can be used to determine the state of the automatic algorithm in the flash memory. Figure 20.6-2 "Example of the Flash Memory Sector Erase Procedure" an example of the procedure for erasing sectors in the flash memory. Here, the toggle bit flag (DQ6) is used to confirm that erasing has terminated.

The data that is read to check the flag is read from the sector to be erased.

The toggle bit flag (DQ6) stops the toggle operation at the same time that the timing limit exceeded flag (DQ5) is changed to 1. For example, even if the timing limit exceeded flag (DQ5) is 1, the toggle bit flag (DQ6) must be rechecked.

The data polling flag (DQ7) also changes at the same time that the timing limit exceeded flag bit (DQ5) changes. As a result, the data polling flag (DQ7) must be rechecked.

Figure 20.6-2 Example of the Flash Memory Sector Erase Procedure



20.6.5 Suspending Sector Erase of Flash Memory

This section describes the procedure for issuing the Sector Erase Suspend command to suspend erasing of flash memory sectors. Data can be read from sectors that are not being erased.

■ Suspending Erasing of Flash Memory Sectors

Erasing of flash memory sectors can be suspended by sending the Sector Erase Suspend command in the command sequence table (see Table 20.4-1 "Command Sequence Table" in Section 20.4 "Starting the Flash Memory Automatic Algorithm") continuously to the target sector in the flash memory.

The Sector Erase Suspend command suspends the sector erase operation being executed and enables data to be read from sectors that are not being erased. In this state, only reading is enabled; data cannot be written. This command is valid only during sector erase operations that include an erase wait time. The command will be ignored during chip erase or write operations.

This command is implemented by writing the erase suspend code (B0H). At this time, specify an optional address in the flash memory for the address. An Erase Suspend command issued again during erasing of sectors will be ignored.

Entering the Sector Erase Suspend command during the sector erase wait period will immediately terminate sector erase wait, cancel the erase operation, and set the erase stop state. Entering the Erase Suspend command during the erase operation after the sector erase wait period has terminated will set the erase suspend state after a maximum period of 15 μs has elapsed.

20.6.6 Restarting Sector Erase of Flash Memory

This section describes the procedure for issuing the Sector Erase Restart command to restart suspended erasing of flash memory sectors.

■ Restarting Erasing of Flash Memory Sectors

Suspended erasing of flash memory sectors can be restarted by sending the Sector Erase Restart command in the command sequence table (see Table 20.4-1 "Command Sequence Table" in Section 20.4 "Starting the Flash Memory Automatic Algorithm") continuously to the target sector in the flash memory.

The Sector Erase Restart command is used to restart erasing of sectors from the sector erase suspend state set using the Sector Erase Suspend command. The Sector Erase Restart command is implemented by writing the erase restart code (30H). At this time, specify an optional address in the flash memory area for the address.

If a Sector Erase Restart command is issued during sector erase, the command will be ignored.

CHAPTER 21 EXAMPLE OF MB90F562/F562B/F568 SERIAL PROGRAMMING CONNECTION

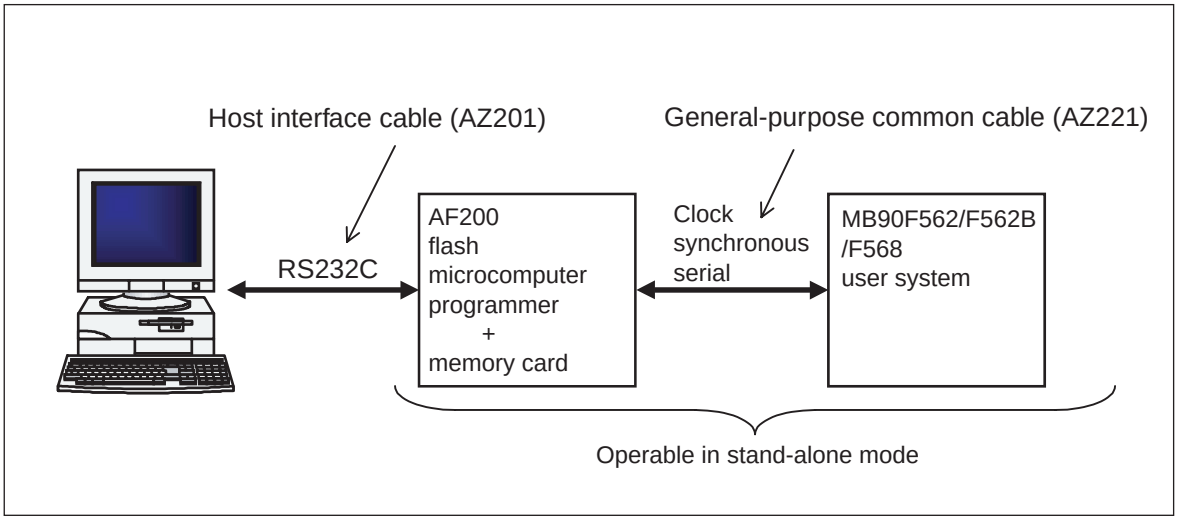
This chapter provides examples of serial programming connection with the AF220/ AF210/AF120/AF110 flash microcomputer programmer manufactured by YDC Corporation.

21.1 Standard Configuration for Serial On-board Programming (Fujitsu Standard)

MB90F562/F562B/F568 supports serial on-board writing (Fujitsu standard) to flash ROM. This describes the specifications for serial on-board writing.

■ Standard Configuration for Fujitsu Standard Serial on-board Programming

The flash microcomputer programmer (AF220/AF210/AF120/AF110) manufactured by YDC Corporation is used for Fujitsu standard serial on-board writing.



Note:

Contact YDC Corporation for information about the functionality and operation of the flash microcomputer programmer (AF220/AF210/AF120/AF110) and information on the general-purpose common cable for connection (AZ221) and connectors.

Table 21.1-1 Pins Used for Fujitsu Standard Serial on-board Programming

Pin	Function	Description
MD2,MD1, MD0	Mode pin	Serial programming mode is entered by setting MD2=1, MD1=1, and MD0=0.
X0, X1	Oscillator pin	In serial programming mode, since the internal operating clock of the CPU is one times the PLL clock, the oscillation clock frequency is the internal operation clock. Therefore, when performing serial programming operation, the range of frequencies that can be written to the high-speed oscillation input pin is from 1 MHz to 16 MHz.
P00, P01	Programming start pin	Input the "L" level to P00 and the "H" level to P01.

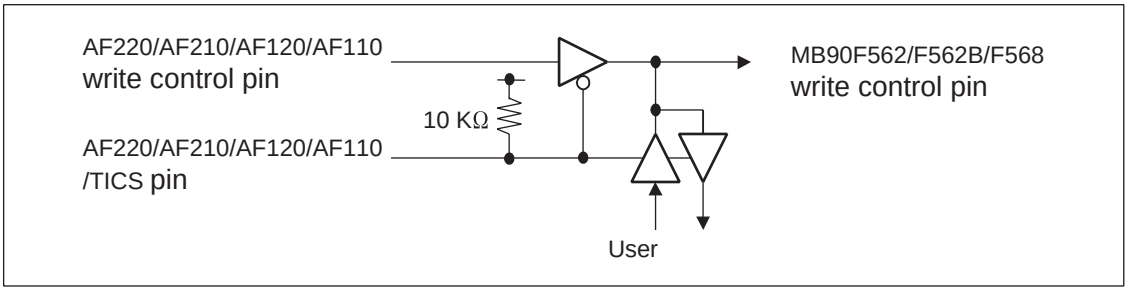
Table 21.1-1 Pins Used for Fujitsu Standard Serial on-board Programming (Continued)

Pin	Function	Description
RSTX	Reset pin	-
SIN1	Serial data input pin	UART0 and UART1 are used in clock synchronous mode. The pins used in clock synchronous mode of UART0 are SIN1, SOT1, and SCK0 in programming mode.
SOT1	Serial data output pin	
SCK0	Serial clock input pin	
C	C terminal	Capacity terminal for stabilizing the power supply. Connect an external ceramic capacitor of about 0.1µ.
V_{CC}	Power supply pin	To supply the programming voltage (MB90F562/F562B: 5V - 10% to 5V + 10%, MB90F568: 3V - 10% to 3V + 10%) from the user system, the flash microcomputer programmer need not be connected. Make sure that no short-circuit occurs with the user-side power supply when making connections.
VSS	Ground pin	Used also as the ground pin for the flash microcomputer programmer.

Note:

When the P00, P01,SIN1,SOT1,and SCK0 pins are also used by the user system, the control circuit shown in Figure 21.1-1 "Control Circuit" is required. (The /TICS signal of the flash microcomputer programmer can separate the user circuit during serial writing. See the connection example shown later.)

Figure 21.1-1 Control Circuit



Refer to the following four serial writing examples in Sections 20.2 to 20.5.

- Example of serial programming connection
  - When power supplied by user
- Example of serial programming connection
  - When power supplied from flash microcomputer programmer
- Example of minimum connection to flash microcomputer programmer
  - When power supplied from user
- Example of minimum connection to flash microcomputer programmer
  - When power supplied from flash microcomputer programmer

The serial clock frequency of MB90F562/F562B/F568 that can be input is determined by the formula below. Thus, it is necessary to change the setting of the serial clock input frequency in the flash microcomputer programmer depending on the oscillation clock frequency currently being used.

Serial clock frequency that can be input = 0.125 x oscillation clock frequency

Table 21.1-2 Maximum serial clock frequency

Oscillation clock frequency	Maximum serial clock frequency of a microcomputer that can be input	Maximum serial clock frequency of AF220/ AF210/AF120/AF110 that can be set	Maximum serial clock frequency of AF200 that can be set
For 4 MHz	500 KHz	500 KHz	500 KHz
For 8 MHz	1 MHz	850 KHz	500 KHz
For 16 MHz	2 MHz	1.25 MHz	500 KHz

Table 21.1-3 System Configuration of the Flash Microcomputer Programmer (AF220/AF210/AF120/ AF110) (YDC Corporation)

Model		Function
Main unit	AF200/AC4P	Ethernet interface built-in model: /100V - 220V power adapter
	AF210/AC4P	Standard model: /100V - 220V power adapter
	AF120/AC4P	Single key/Ethernet interface built-in model: /100V - 220V power adapter
	AF110/AC4P	Single key model: /100V - 220V power adapter
AZ221		PC/AT RS232C cable only for Programmer
AZ210		Standard target probe (a) Length: 1 m
FF201		Fujitsu F^2MC-16LX flash microcomputer control module
AZ290		Remote controller
AZ264		Power regulator (MB90F568: Required when power is supplied from the flash microcomputer programmer to the 3V products)
/P2		2MB PC Card (option) Flash memory capacity: up to 128 MB supported
/P4		4MB PC Card (option) Flash memory capacity: up to 512 MB supported

Inquiries: YDC Corporation
Telephone number: (81)-42-333-6224

Note:

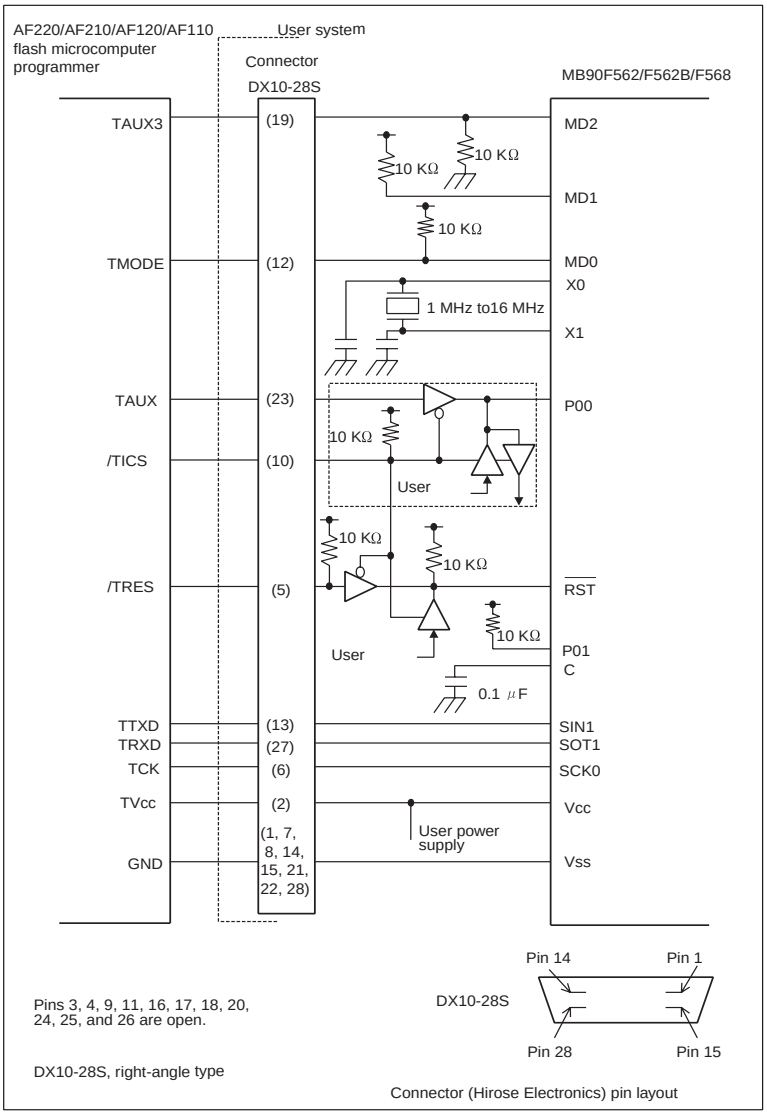
The AF200 flash microcomputer programmer is no longer manufactured, but can be supported by using the control module FF201. Also, for an example of the serial programming connection, see the example of the connection in the next section.

21.2 Example of Connection for Serial Programming (Power Supplied by User)

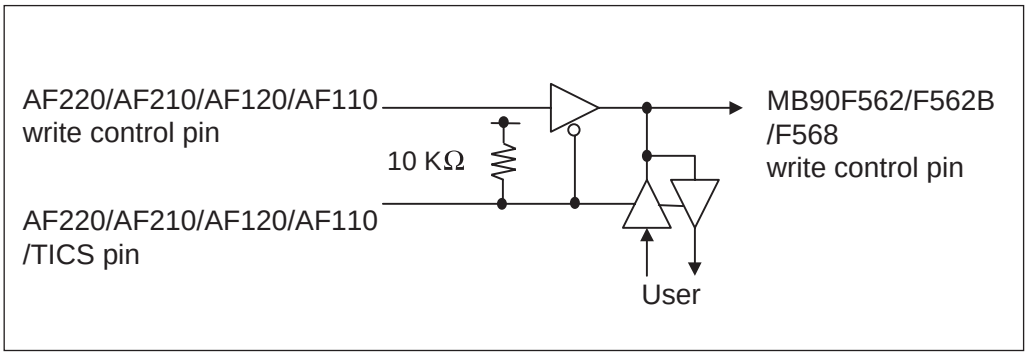
Figure 21.2-1 "Example of Connection for Serial Programming in MB90F562/F562B/F568 Internal Vector Mode (When Power Supplied by User)" shows an example of connection for serial writing when the power supply voltage of the microcomputer is supplied from the user power. MD2=1 and MD0=0 are each input to the mode pins MD2 and MD0 from TAUX3 and TMODE of AF220/AF210/AF120/AF110.
Serial reprogramming mode: MD2, MD1, MD0=110_B

■ Example of Connection for Serial Programming (When Power Supplied by User)

Figure 21.2-1 Example of Connection for Serial Programming in MB90F562/F562B/F568 Internal Vector Mode (When Power Supplied by User)



- When the user system also uses pins SIN1, SOT1, and SCK0, the control circuit shown below is necessary, just as it is for P00. (During serial writing, the user circuit can be disconnected by the flash microcomputer programmer /TICS signal.)
- Before connecting to AF220/AF210/AF120/AF110, turn off the user power.



21.3 Example of Connection for Serial Programming (When Power Supplied from Programmer)

Figure 21.3-1 "Example of Connection for Serial Programming in MB90F562/F562B Internal Vector Mode (When Power Supplied from Programmer)" shows an example of connection for serial writing when the power supply voltage of the microcomputer is supplied from the programmer power. MD2=1 and MD0=0 are each input to the mode pins MD2 and MD0 from TAUX3 and TMODE of AF220/AF210/AF120/AF110.
Serial reprogramming mode: MD2, MD1, MD0=110_B

■ Example of Connection for Serial Programming (When Power Supplied from Programmer)

Figure 21.3-1 Example of Connection for Serial Programming in MB90F562/F562B Internal Vector Mode (When Power Supplied from Programmer)

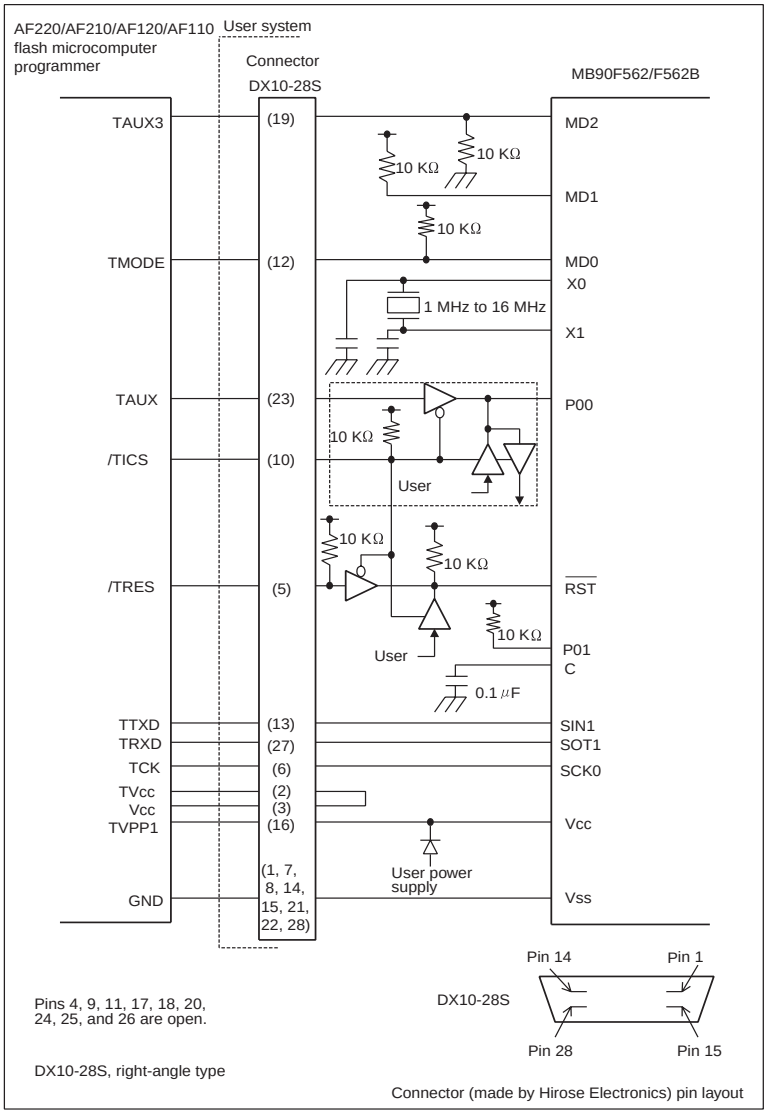
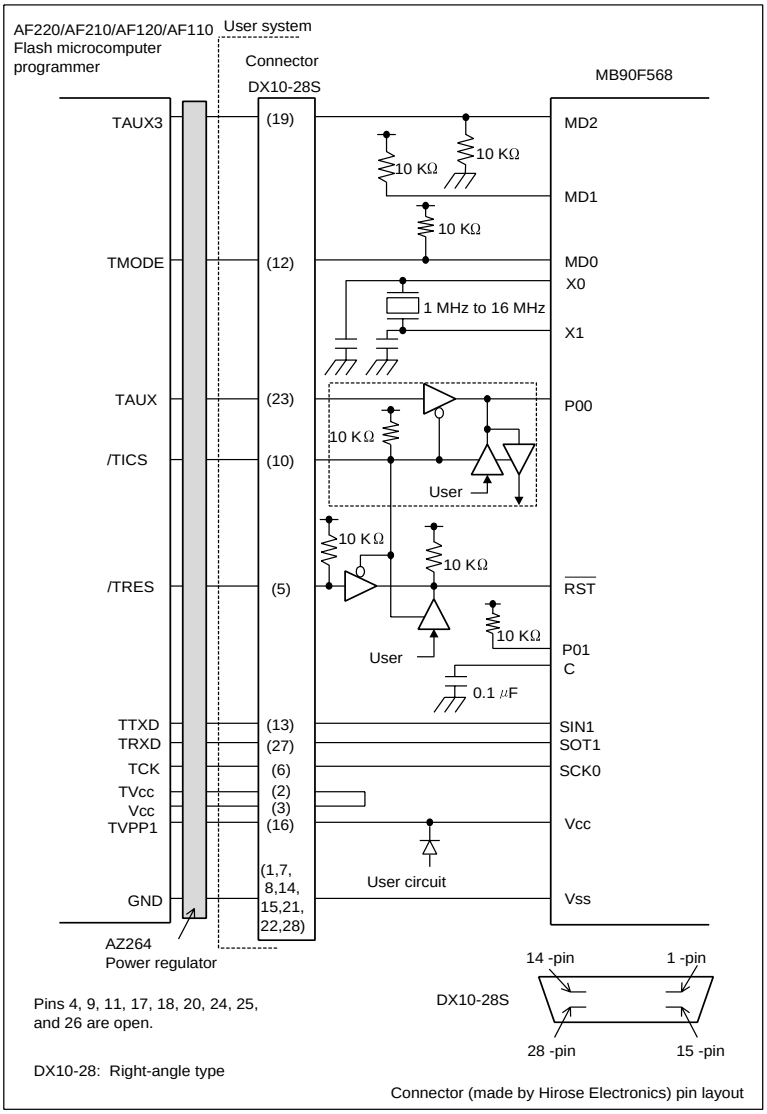
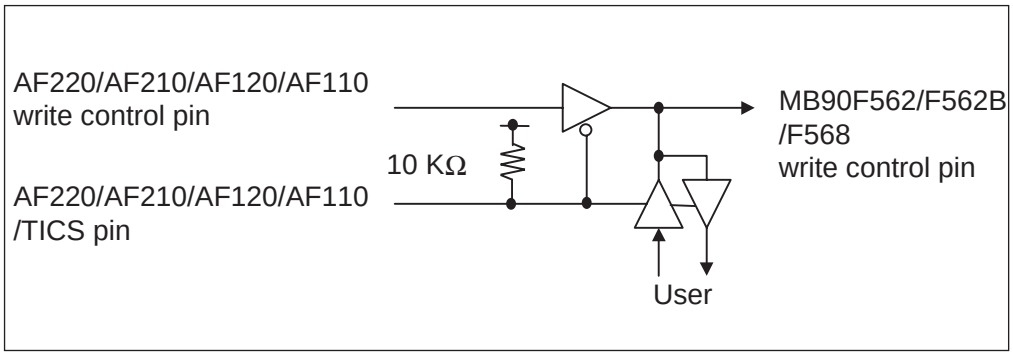


Figure 21.3-2 Example of Connection for Serial Programming in MB90F568 Internal Vector Mode (when power supplied from the programmer power)



- When the SIN1, SOT1, and SCK0 pins are also used by the user system, the control circuit shown below is necessary, just as it is for P00. (During serial writing, the user circuit can be disconnected by the flash microcomputer /TICS signal.)
- Before connecting the AF220/AF210/AF120/AF110, turn off the power supplied by the user.
- When supplying power from the AF220/AF210/AF120/AF110, do not create a short circuit to the power supplied by the user.


AF220/AF210/AF120/AF110
write control pin
AF220/AF210/AF120/AF110
/TICS pin
10 KΩ
MB90F562/F562B
/F568
write control pin
User

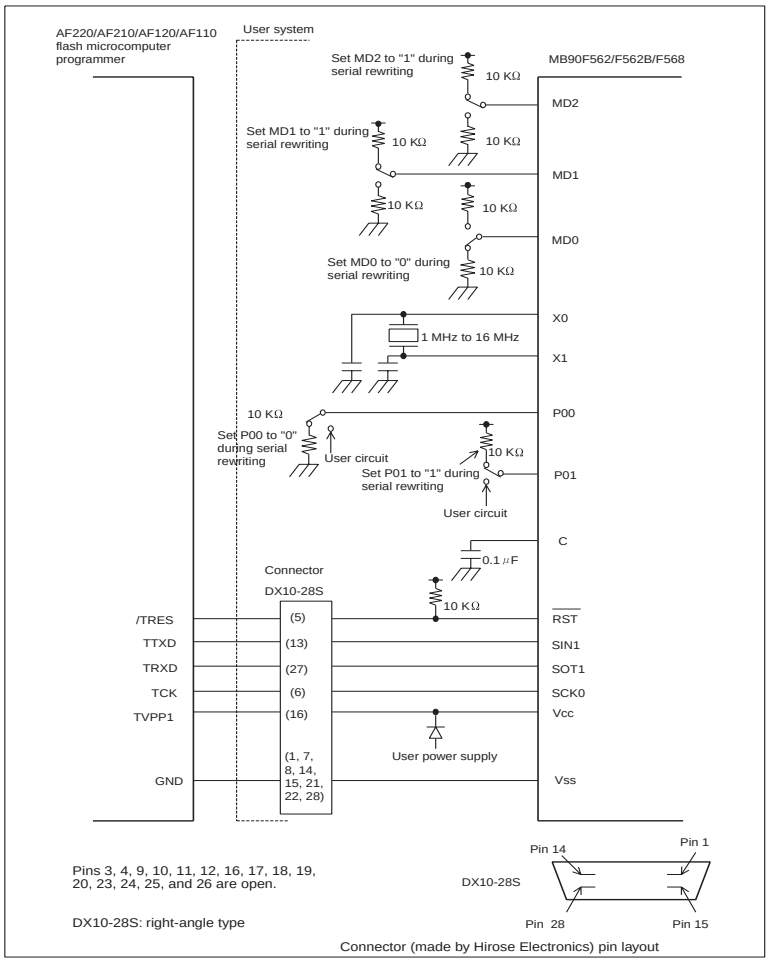
21.4 Example of Minimum Connection with Flash Microcomputer Programmer (When Power Supplied from User)

Figure 21.4-1 "Example of Minimum Connection with Flash Microcomputer Programmer (When Power Supplied by User)" is an example of the minimum connection with the flash microcomputer programmer when power is supplied by the user.
Serial reprogramming mode: MD2, MD1, MD0=110_B

■ Example of Minimum Connection with Flash Microcomputer Programmer (When Power Supplied by User)

If the MB90F562/F562B/F568 pins are set as shown in Figure 21.4-1 "Example of Minimum Connection with Flash Microcomputer Programmer (When Power Supplied by User)" for writing data to the flash memory, MD2, MD1, MD0, and P00 and the flash microcomputer programmer need not be connected.

Figure 21.4-1 Example of Minimum Connection with Flash Microcomputer Programmer (When Power Supplied by User)



- Before connecting to AF220/AF210/AF120/AF110, turn off the user power.

21.5 Example of Minimum Connection with Flash Microcomputer Programmer (When Power Supplied from Programmer)

Figure 21.5-1 "Example of Minimum Connection with Flash Microcomputer Programmer (When Power Supplied from programmer)" is an example of the minimum connection with the flash microcomputer programmer when power is supplied from the programmer.
Serial reprogramming mode: MD2, MD1, MD0=110_B

■ Example of Minimum Connection with Flash Microcomputer Programmer (When Power Supplied from programmer)

If the MB90F562/F562B/F568 pins are set as shown in Figure 21.5-1 "Example of Minimum Connection with Flash Microcomputer Programmer (When Power Supplied from programmer)" for writing data to the flash memory, MD2, MD1, MD0, and P00 and the flash microcomputer programmer need not be connected.

Figure 21.5-1 Example of Minimum Connection with Flash Microcomputer Programmer (When Power Supplied from programmer)

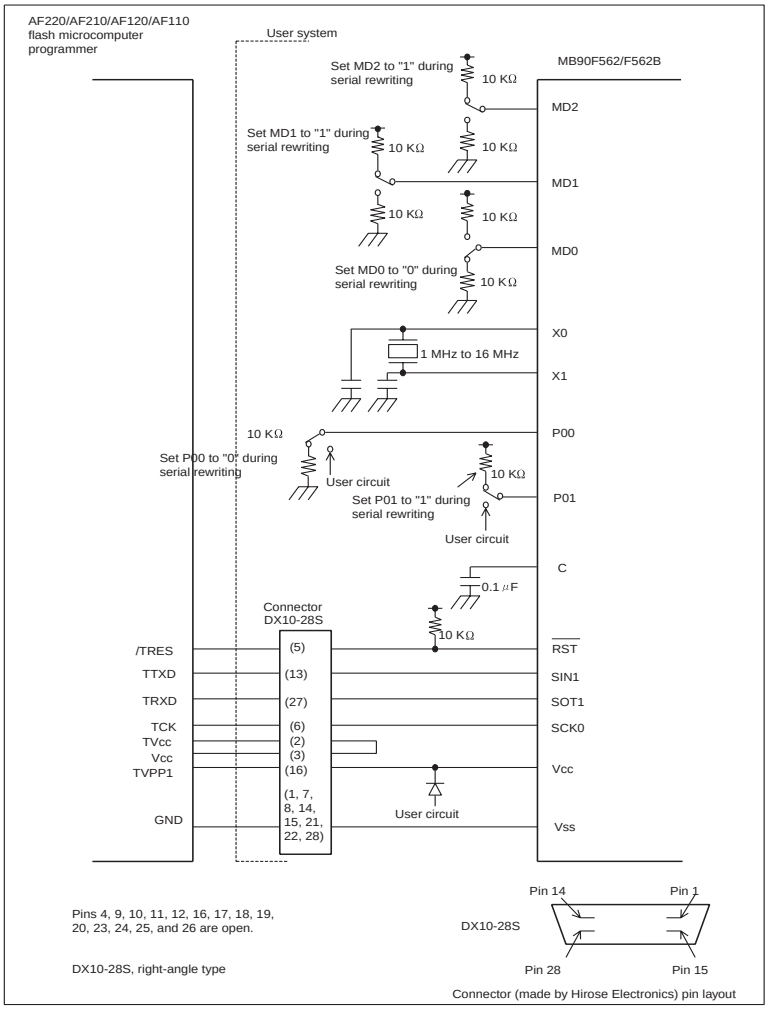
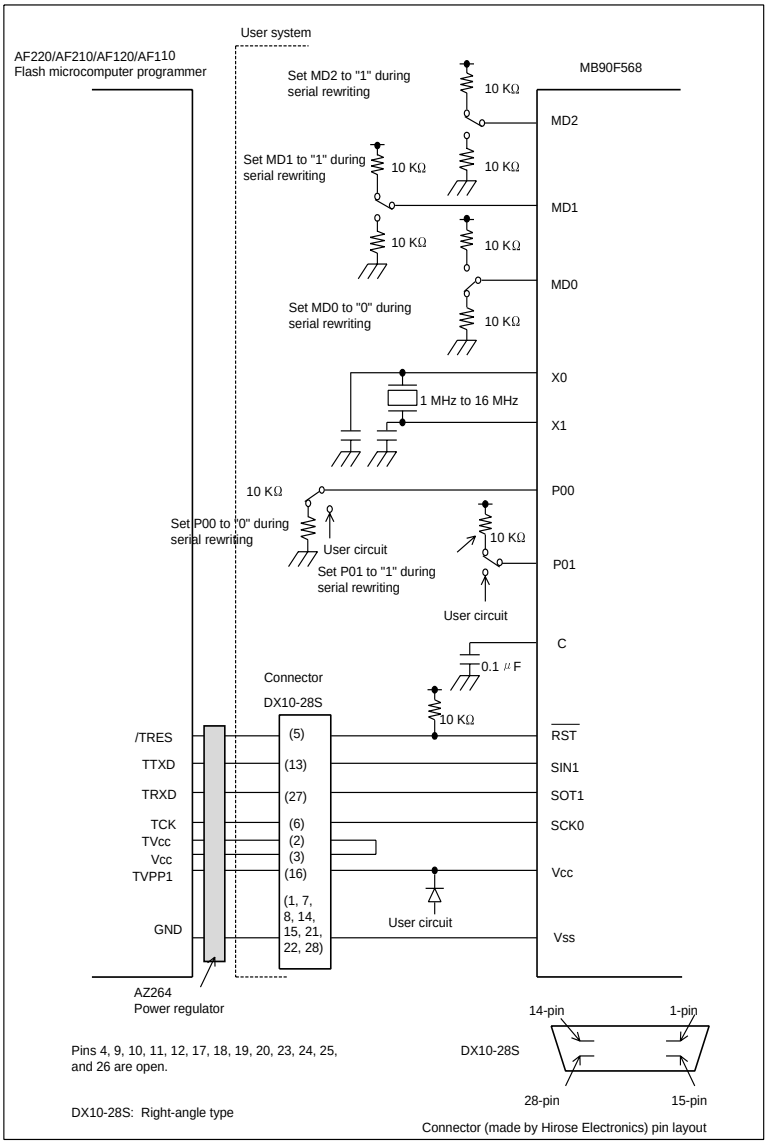


Figure 21.5-2 Example of Minimum Connection with the Flash Microcomputer Programmer (when power supplied from programmer)



- Before connecting the AF220/AF210/AF120/AF110, turn off the power supplied by the user.
- When power is supplied from the AF220/AF210/AF120/AF110, do not create a short circuit to the power supplied by the user.

APPENDIX

The appendix provides I/O maps and outlines instructions.

APPENDIX A I/O Map

Table A-1 "I/O Map" lists the addresses assigned to the registers for peripheral functions in the MB90560/565 series.

■ I/O Map

Table A-1 I/O Map

Address	Abbreviation	Register	Access	Resource name	Initial value
000000_H	PDR0	Port 0 data register	R/W	Port 0	$XXXXXXXX_B$
000001_H	PDR1	Port 1 data register	R/W	Port 1	$XXXXXXXX_B$
000002_H	PDR2	Port 2 data register	R/W	Port 2	$XXXXXXXX_B$
000003_H	PDR3	Port 3 data register	R/W	Port 3	$XXXXXXXX_B$
000004_H	PDR4	Port 4 data register	R/W	Port 4	$XXXXXXXX_B$
000005_H	PDR5	Port 5 data register	R/W	Port 5	$XXXXXXXX_B$
000006_H	PDR6	Port 6 data register	R/W	Port 6	$XXXXXXXX_B$
000007_H to $00000F_H$	Prohibited area				
000010_H	DDR0	Port 0 direction register	R/W	Port 0	00000000_B
000011_H	DDR1	Port 1 direction register	R/W	Port 1	00000000_B
000012_H	DDR2	Port 2 direction register	R/W	Port 2	00000000_B
000013_H	DDR3	Port 3 direction register	R/W	Port 3	00000000_B
000014_H	DDR4	Port 4 direction register	R/W	Port 4	$X0000000_B$
000015_H	DDR5	Port 5 direction register	R/W	Port 5	00000000_B
000016_H	DDR6	Port 6 direction register	R/W	Port 6	$XXXX0000_B$
000017_H	ADER	Analog input enable register	R/W	Port 5, A/D converter	11111111_B
000018_H to $00001F_H$	Prohibited area				

Table A-1 I/O Map (Continued)

Address	Abbreviation	Register	Access	Resource name	Initial value
000020_H	SMR0	Mode register CH0	R/W	UART0	$00000X00_B$
000021_H	SCR0	Control register CH0	W, R/W		00000100_B
000022_H	SIDR0	Input data register CH0	R		$XXXXXXXX_B$
	SODR0	Output data register CH0	W		
000023_H	SSR0	Status register CH0	R, R/W		00001000_B
000024_H	SMR1	Mode register CH1	R/W	UART1	$00000X00_B$
000025_H	SCR1	Control register CH1	W, R/W		00000100_B
000026_H	SIDR1	Input data register CH1	R		$XXXXXXXX_B$
	SODR1	Output data register CH1	W		
000027_H	SSR1	Status register CH1	R, R/W		00001000_B
000028_H	Prohibited area				
000029_H	CDCR0	Communication prescaler control register CH0	R/W	Communication prescaler	$0XXX0000_B$
$00002A_H$	Prohibited area				
$00002B_H$	CDCR1	Communication prescaler control register CH1	R/W	Communica-tion prescale	$0XXX0000_B$
$00002C_H$ to $00002F_H$	Prohibited area				
000030_H	ENIR	DTP/Interrupt enable register	R/W	DTP/external interrupt	00000000_B
000031_H	EIRR	DTP/Interrupt cause register	R/W		$XXXXXXXX_B$
000032_H	ELVR	Request level setting register (lower)	R/W		00000000_B
000033_H		Request level setting register (upper)	R/W		00000000_B
000034_H	ADCS0	A/D control status register (lower)	R/W	8/10 bit A/D converter	00000000_B
000035_H	ADCS1	A/D control status register (upper)	R, R/W		00000000_B
000036_H	ADCR0	A/D data register (lower)	R		$XXXXXXXX_B$
000037_H	ADCR1	A/D data register (upper)	R, W		$00000XXX_B$

Table A-1 I/O Map (Continued)

Address	Abbreviation	Register	Access	Resource name	Initial value
000038_H	PRLL0	PPG reload register CH0 (lower)	R/W	8-/16-bit PPG timer	$XXXXXXXX_B$
000039_H	PRLH0	PPG reload register CH0 (upper)	R/W		$XXXXXXXX_B$
$00003A_H$	PRLL1	PPG reload register CH1 (lower)	R/W		$XXXXXXXX_B$
$00003B_H$	PRLH1	PPG reload register CH1 (upper)	R/W		$XXXXXXXX_B$
$00003C_H$	PPGC0	PPG control register CH0 (lower)	R/W		00000001_B
$00003D_H$	PPGC1	PPG control register CH1 (upper)	R/W		00000001_B
$00003E_H$	PCS01	PPG clock control register CH0/1	R/W		$000000XX_B$
$00003F_H$	Prohibited area				
000040_H	PRLL2	PPG reload register CH2 (lower)	R/W	8-/16-bit PPG timer	$XXXXXXXX_B$
000041_H	PRLH2	PPG reload register CH2 (upper)	R/W		$XXXXXXXX_B$
000042_H	PRLL3	PPG reload register CH3 (lower)	R/W		$XXXXXXXX_B$
000043_H	PRLH3	PPG reload register CH3 (upper)	R/W		$XXXXXXXX_B$
000044_H	PPGC2	PPG control register CH2 (lower)	R/W		00000001_B
000045_H	PPGC3	PPG control register CH3 (upper)	R/W		00000001_B
000046_H	PCS23	PPG clock control register CH2/3	R/W		$000000XX_B$
000047_H	Prohibited area				
000048_H	PRLL4	PPG reload register CH4 (lower)	R/W	8-/16-bit PPG timer	$XXXXXXXX_B$
000049_H	PRLH4	PPG reload register CH4 (upper)	R/W		$XXXXXXXX_B$
$00004A_H$	PRLL5	PPG reload register CH5 (lower)	R/W		$XXXXXXXX_B$
$00004B_H$	PRLH5	PPG reload register CH5 (upper)	R/W		$XXXXXXXX_B$
$00004C_H$	PPGC4	PPG control register CH4 (lower)	R/W		00000001_B
$00004D_H$	PPGC5	PPG control register CH5 (upper)	R/W		00000001_B
$00004E_H$	PCS45	PPG clock control register CH4/5	R/W		$000000XX_B$
$00004F_H$	Prohibited area				
000050_H	TMRR0	8-bit reload register CH0	R/W	Waveform control register	$XXXXXXXX_B$
000051_H	DTCR0	8-bit timer control register CH0	R/W		00000000_B
000052_H	TMRR1	8-bit reload register CH1	R/W		$XXXXXXXX_B$
000053_H	DTCR1	8-bit timer control register CH1	R/W		00000000_B
000054_H	TMRR2	8-bit reload register CH2	R/W		$XXXXXXXX_B$
000055_H	DTCR2	8-bit timer control register CH2	R/W		00000000_B
000056_H	SIGCR	Waveform control register	R/W		00000000_B
000057_H	Prohibited area				

Table A-1 I/O Map (Continued)

Address	Abbreviation	Register	Access	Resource name	Initial value
000058_H	CPCLR	Compare clear register (lower)	R/W	16-bit free-running timer	$XXXXXXXX_B$
000059_H		Compare clear register (upper)	R/W		$XXXXXXXX_B$
$00005A_H$	TCDT	Timer data register (lower)	R/W		00000000_B
$00005B_H$		Timer data register (upper)	R/W		00000000_B
$00005C_H$	TCCS	Timer control status register (lower)	R/W		00000000_B
$00005D_H$		Timer control status register (upper)	R/W		$0XX00000_B$
$00005E_H$	Prohibited area				
$00005F_H$					
000060_H	IPCP0	Input capture data register CH0 (lower)	R	Input capture	$XXXXXXXX_B$
000061_H		Input capture data register CH0 (upper)	R		$XXXXXXXX_B$
000062_H	IPCP1	Input capture data register CH1 (lower)	R		$XXXXXXXX_B$
000063_H		Input capture data register CH1 (upper)	R		$XXXXXXXX_B$
000064_H	IPCP2	Input capture data register CH2 (lower)	R		$XXXXXXXX_B$
000065_H		Input capture data register CH2 (upper)	R		$XXXXXXXX_B$
000066_H	IPCP3	Input capture data register CH3 (lower)	R		$XXXXXXXX_B$
000067_H		Input capture data register CH3 (upper)	R		$XXXXXXXX_B$
000068_H	ICS01	Input capture control register 01	R/W		00000000_B
000069_H	Prohibited area				
$00006A_H$	ICS23	Input capture control register 23	R/W	Input capture	00000000_B
$00006B_H$ to $00006E_H$	Prohibited area				
$00006F_H$	ROMM	ROM mirroring function selection register	W	ROM mirroring function selection module	$XXXXXXX1_B$

Table A-1 I/O Map (Continued)

Address	Abbreviation	Register	Access	Resource name	Initial value
000070_H	OCCP0	Compare register CH0 (lower)	R/W	Output compare	$XXXXXXXX_B$
000071_H		Compare register CH0 (upper)	R/W		$XXXXXXXX_B$
000072_H	OCCP1	Compare register CH1 (upper)	R/W		$XXXXXXXX_B$
000073_H		Compare register CH1 (lower)	R/W		$XXXXXXXX_B$
000074_H	OCCP2	Compare register CH2 (lower)	R/W		$XXXXXXXX_B$
000075_H		Compare register CH2 (upper)	R/W		$XXXXXXXX_B$
000076_H	OCCP3	Compare register CH3 (lower)	R/W		$XXXXXXXX_B$
000077_H		Compare register CH3 (upper)	R/W		$XXXXXXXX_B$
000078_H	OCCP4	Compare register CH4 (lower)	R/W		$XXXXXXXX_B$
000079_H		Compare register CH4 (upper)	R/W		$XXXXXXXX_B$
$00007A_H$	OCCP5	Compare register CH5 (lower)	R/W		$XXXXXXXX_B$
$00007B_H$		Compare register CH5 (upper)	R/W		$XXXXXXXX_B$
$00007C_H$	OCS0	Compare control register CH0 (lower)	R/W		$0000XX00_B$
$00007D_H$	OCS1	Compare control register CH1 (upper)	R/W		$XXX00000_B$
$00007E_H$	OCS2	Compare control register CH2 (lower)	R/W		$0000XX00_B$
$00007F_H$	OCS3	Compare control register CH3 (upper)	R/W		$XXX00000_B$
000080_H	OCS4	Compare control register CH4 (lower)	R/W		$0000XX00_B$
000081_H	OCS5	Compare control register CH5 (upper)	R/W		$XXX00000_B$
000082_H	TMCSR0:L	Timer control status register CH0 (lower)	R/W	16-bit reload timer	00000000_B
000083_H	TMCSR0:H	Timer control status register CH0 (upper)	R/W		$XXXX0000_B$
000084_H	TMR	16-bit timer register CH0 (lower)	R		$XXXXXXXX_B$
	TMRLR0	16-bit reload register CH0 (lower)	W		$XXXXXXXX_B$
000085_H	TMR0	16-bit timer register CH0 (upper)	R		$XXXXXXXX_B$
	TMRHR0	16-bit reload register CH0 (upper)	W		$XXXXXXXX_B$

Table A-1 I/O Map (Continued)

Address	Abbreviation	Register	Access	Resource name	Initial value
000086_H	TMCSR1:L	Timer control status register CH1 (lower)	R/W	16-bit reload timer	00000000_B
000087_H	TMCSR1:H	Timer control status register CH1 (upper)	R/W		$XXXX0000_B$
000088_H	TMR1	16-bit timer register CH1 (lower)	R		$XXXXXXXX_B$
	TMRLR1	16-bit reload register CH1 (lower)	W		$XXXXXXXX_B$
000089_H	TMR1	16-bit timer register CH1 (upper)	R		$XXXXXXXX_B$
	TMRHR1	16-bit reload register CH1 (upper)	W		$XXXXXXXX_B$
$00008A_H$ to $00008B_H$	Prohibited area				
$00008C_H$	RDR0	Port 0 pull-up resistor setting register	R/W	Port 0	00000000_B
$00008D_H$	RDR1	Port 1 pull-up resistor setting register	R/W	Port 1	00000000_B
$00008E_H$ to $00009D_H$	Prohibited area				
$00009E_H$	PACSR	Program address detection control register	R/W	Address match detection	00000000_B
$00009F_H$	DIRR	Delayed interrupt cause/clear register	R/W	Delayed interrupt	$XXXXXXX0_B$
$0000A0_H$	LPMCR	Low-power consumption mode register	W, R/W	Low-power consumption control register	00011000_B
$0000A1_H$	CKSCR	Clock selection register	R, R/W		11111100_B
$0000A2_H$ to $0000A7_H$	Prohibited area				
$0000A8_H$	WDTC	Watchdog control register	R, W	Watchdog timer	$1XXXX111_B$
$0000A9_H$	TBTC	Timebase timer control register	W, R/W	Timebase timer	$1XX00100_B$
$0000AA_H$ to $0000AD_H$	Prohibitee area				
$0000AE_H$	FMCS	Flash memory control status register	R, W, R/W	Flash memory interface circuit	00000000_B
$0000AF_H$	Prohibitee area				

Table A-1 I/O Map (Continued)

Address	Abbreviation	Register	Access	Resource name	Initial value
$0000B0_H$	ICR00	Interrupt control register 00 (for writing)	W,R/W	Interrupt	$XXXX0111_B$
		Interrupt control register 00 (for reading)	R,R/W		$XX000111_B$
$0000B1_H$	ICR01	Interrupt control register 01 (for writing)	W,R/W		$XXXX0111_B$
		Interrupt control register 01 (for reading)	R,R/W		$XX000111_B$
$0000B2_H$	ICR02	Interrupt control register 02 (for writing)	W,R/W		$XXXX0111_B$
		Interrupt control register 02 (for reading)	R,R/W		$XX000111_B$
$0000B3_H$	ICR03	Interrupt control register 03 (for writing)	W,R/W		$XXXX0111_B$
		Interrupt control register 03 (for reading)	R,R/W		$XX000111_B$
$0000B4_H$	ICR04	Interrupt control register 04 (for writing)	W,R/W		$XXXX0111_B$
		Interrupt control register 04 (for reading)	R,R/W		$XX000111_B$
$0000B5_H$	ICR05	Interrupt control register 05 (for writing)	W,R/W		$XXXX0111_B$
		Interrupt control register 05 (for reading)	R,R/W		$XX000111_B$
$0000B6_H$	ICR06	Interrupt control register 06 (for writing)	W,R/W		$XXXX0111_B$
		Interrupt control register 06 (for reading)	R,R/W		$XX000111_B$
$0000B7_H$	ICR07	Interrupt control register 07 (for writing)	W,R/W		$XXXX0111_B$
		Interrupt control register 07 (for reading)	R,R/W		$XX000111_B$
$0000B8_H$	ICR08	Interrupt control register 08 (for writing)	W,R/W		$XXXX0111_B$
		Interrupt control register 08 (for reading)	R,R/W		$XX000111_B$
$0000B9_H$	ICR09	Interrupt control register 09 (for writing)	W,R/W		$XXXX0111_B$
		Interrupt control register 09 (for reading)	R,R/W		$XX000111_B$

Table A-1 I/O Map (Continued)

Address	Abbreviation	Register	Access	Resource name	Initial value
$0000BA_H$	ICR10	Interrupt control register 10 (for writing)	W,R/W	Interrupt	$XXXX0111_B$
		Interrupt control register 10 (for reading)	R,R/W		$XX000111_B$
$0000BB_H$	ICR11	Interrupt control register 11 (for writing)	W,R/W		$XXXX0111_B$
		Interrupt control register 11 (for reading)	R,R/W		$XX000111_B$
$0000BC_H$	ICR12	Interrupt control register 12 (for writing)	W,R/W		$XXXX0111_B$
		Interrupt control register 12 (for reading)	R,R/W		$XX000111_B$
$0000BD_H$	ICR13	Interrupt control register 13 (for writing)	W,R/W		$XXXX0111_B$
		Interrupt control register 13 (for reading)	R,R/W		$XX000111_B$
$0000BE_H$	ICR14	Interrupt control register 14 (for writing)	W,R/W		$XXXX0111_B$
		Interrupt control register 14 (for reading)	R,R/W		$XX000111_B$
$0000BF_H$	ICR15	Interrupt control register 15 (for writing)	W,R/W		$XXXX0111_B$
		Interrupt control register 15 (for reading)	R,R/W		$XX000111_B$
$0000C0_H$ to $0000FF_H$	Unused area				
000100_H to $\#_H$	RAM area				
$\#_H$ to $001FEF_H$	Reserved area				

Table A-1 I/O Map (Continued)

Address	Abbreviation	Register	Access	Resource name	Initial value
$001FF0_H$	PADR0	Program address detection register CH0 (lower)	R/W	Address match detection	$XXXXXXXX_B$
$001FF1_H$		Program address detection register CH0 (middle)	R/W		$XXXXXXXX_B$
$001FF2_H$		Program address detection register CH0 (upper)	R/W		$XXXXXXXX_B$
$001FF3_H$	PADR1	Program address detection register CH1 (lower)	R/W		$XXXXXXXX_B$
$001FF4_H$		Program address detection register CH1 (middle)	R/W		$XXXXXXXX_B$
$001FF5_H$		Program address detection register CH1 (upper)	R/W		$XXXXXXXX_B$
$001FF6_H$ to $1FFF_H$	Unused area				

❍ **Meaning of Abbreviations Used for Reading and Writing**

R/W: Read and write enabled

R: Read only

W: Write only

❍ **Explanation of Initial Values**

0: The bit is initialized to 0.

1: The bit is initialized to 1.

X: The initial value of the bit is undefined.

APPENDIX B Instructions

Appendix B describes the instructions used by the F^{2}MC-16LX.

B.1 Instruction Types

The F^{2}MC-16LX supports 351 types of instructions. Addressing is enabled by using an effective address field of each instruction or using the instruction code itself.

■ Instruction Types

The F^{2}MC-16LX supports the following 351 types of instructions:

- 41 transfer instructions (byte)
- 38 transfer instructions (word or long word)
- 42 addition/subtraction instructions (byte, word, or long word)
- 12 increment/decrement instructions (byte, word, or long word)
- 11 comparison instructions (byte, word, or long word)
- 11 unsigned multiplication/division instructions (word or long word)
- 11 signed multiplication/division instructions (word or long word)
- 39 logic instructions (byte or word)
- 6 logic instructions (long word)
- 6 sign inversion instructions (byte or word)
- 1 normalization instruction (long word)
- 18 shift instructions (byte, word, or long word)
- 50 branch instructions
- 6 accumulator operation instructions (byte or word)
- 28 other control instructions (byte, word, or long word)
- 21 bit operation instructions
- 10 string instructions

B.2 Addressing

With the F^{2}MC-16LX, the address format is determined by the instruction effective address field or the instruction code itself (implied). When the address format is determined by the instruction code itself, specify an address in accordance with the instruction code used. Some instructions permit the user to select several types of addressing.

■ Addressing

The F^{2}MC-16LX supports the following 23 types of addressing:

- Immediate (#imm)
- Register direct
- Direct branch address (addr16)
- Physical direct branch address (addr24)
- I/O direct (io)
- Abbreviated direct address (dir)
- Direct address (addr16)
- I/O direct bit address (io:bp)
- Abbreviated direct bit address (dir:bp)
- Direct bit address (addr16:bp)
- Vector address (#vct)
- Register indirect (@RWj j = 0 to 3)
- Register indirect with post increment (@RWj+ j = 0 to 3)
- Register indirect with displacement (@RWi + disp8 i = 0 to 7, @RWj+ disp16 j = 0 to 3)
- Long register indirect with displacement (@RLi + disp8 i = 0 to 3)
- Program counter indirect with displacement (@PC + disp16)
- Register indirect with base index (@RW0 + RW7, @RW1 + RW7)
- Program counter relative branch address (rel)
- Register list (rlst)
- Accumulator indirect (@A)
- Accumulator indirect branch address (@A)
- Indirectly-specified branch address (@ear)
- Indirectly-specified branch address (@eam)

■ Effective Address Field

Table B.2-1 "Effective Address Field" lists the address formats specified by the effective address field.

Table B.2-1 Effective Address Field

Code	Representation			Address format	Default bank
00 01 02 03 04 05 06 07	R0 R1 R2 R3 R4 R5 R6 R7	RW0 RW1 RW2 RW3 RW4 RW5 RW6 RW7	RL0 (RL0) RL1 (RL1) RL2 (RL2) RL3 (RL3)	Register direct: Individual parts correspond to the byte, word, and long word types in order from the left.	None
08 09 0A 0B	@RW0 @RW1 @RW2 @RW3			Register indirect	DTB DTB ADB SPB
0C 0D 0E 0F	@RW0+ @RW1+ @RW2+ @RW3+			Register indirect with post increment	DTB DTB ADB SPB
10 11 12 13	@RW0+disp8 @RW1+disp8 @RW2+disp8 @RW3+disp8			Register indirect with 8-bit displacement	DTB DTB ADB SPB
14 15 16 17	@RW4+disp8 @RW5+disp8 @RW6+disp8 @RW7+disp8			Register indirect with 8-bit displacement	DTB DTB ADB SPB
18 19 1A 1B	@RW0+disp16 @RW1+disp16 @RW2+disp16 @RW3+disp16			Register indirect with 16-bit displacement	DTB DTB ADB SPB
1C 1D 1E 1F	@RW0+RW7 @RW1+RW7 @PC+disp16 addr16			Register indirect with index Register indirect with index PC indirect with 16-bit displacement Direct address	DTB DTB PCB DTB

B.3 Direct Addressing

An operand value, register, or address is specified explicitly in direct addressing mode.

■ Direct Addressing

❍ Immediate addressing (#imm)

Specify an operand value explicitly (#imm4/ #imm8/ #imm16/ #imm32).

Figure B.3-1 Example of immediate addressing (#imm)

MOVW A, #01212H (This instruction stores the operand value in A.)

Before execution	A	2233 4455
After execution	A	4455 1212 (Some instructions transfer AL to AH.)

❍ Register direct addressing

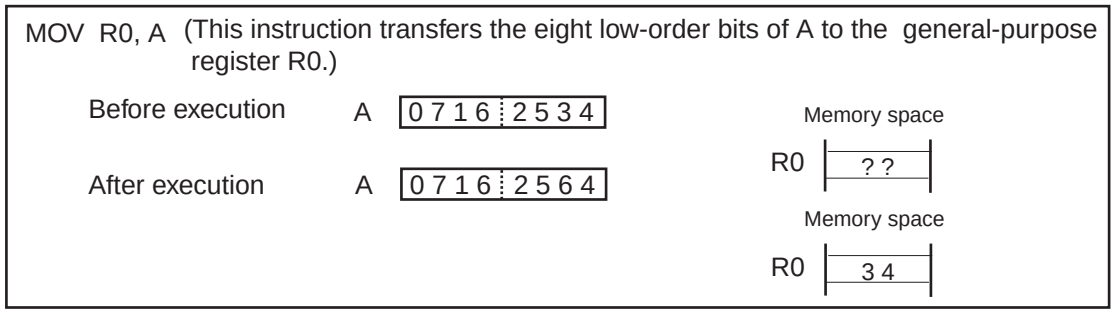
Specify a register explicitly as an operand. Table B.3-1 "Direct Addressing Registers" lists the registers that can be specified. Figure B.3-2 "Example of Register Direct Addressing" shows an example of register direct addressing.

Table B.3-1 Direct Addressing Registers

General-purpose register	Byte	R0, R1, R2, R3, R4, R5, R6, R7
	Word	RW0, RW1, RW2, RW3, RW4, R5W, RW6, RW7
	Long word	RL0, RL1, RL2, RL3
Special-purpose register	Accumulator	A, AL
	Pointer	SP *1
	Bank	PCB, DTB, USB, SSB, ADB
	Page	DPR
	Control	PS, CCR, RP, ILM

*1: One of the user stack pointer (USP) and system stack pointer (SSP) is selected and used depending on the value of the S flag bit in the condition code register (CCR). For branch instructions, the program counter (PC) is not specified in an instruction operand but is specified implicitly.

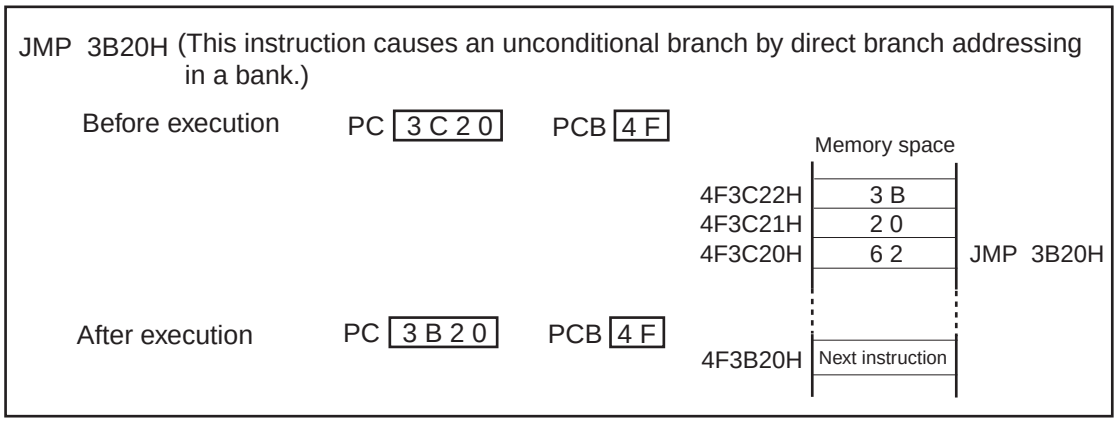
Figure B.3-2 Example of Register Direct Addressing



❍ **Direct branch addressing (addr16)**

Specify an offset explicitly for the branch destination address. The size of the offset is 16 bits, which indicates the branch destination in the logical address space. Direct branch addressing is used for an unconditional branch, subroutine call, or software interrupt instruction. Bits 23 to 16 of the address are specified by the program bank register (PCB).

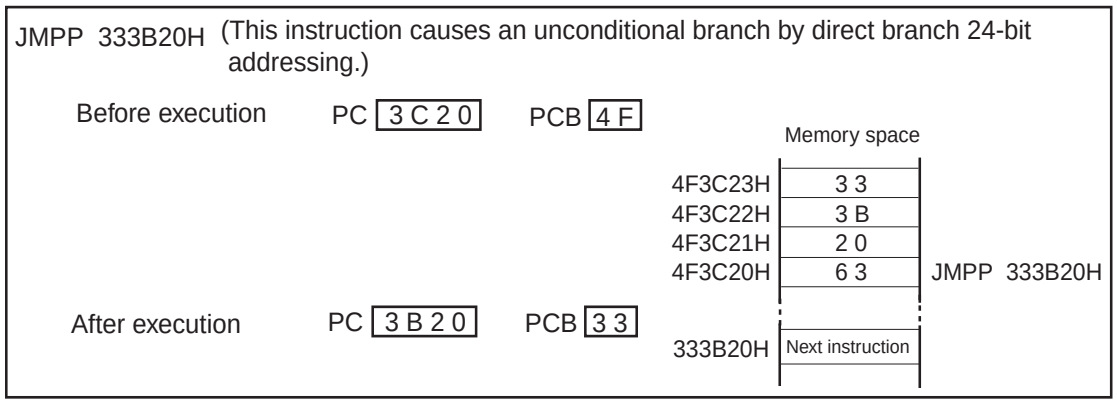
Figure B.3-3 Example of Direct Branch Addressing (addr16)



❍ **Physical direct branch addressing (addr24)**

Specify an offset explicitly for the branch destination address. The size of the offset is 24 bits. Physical direct branch addressing is used for unconditional branch, subroutine call, or software interrupt instruction.

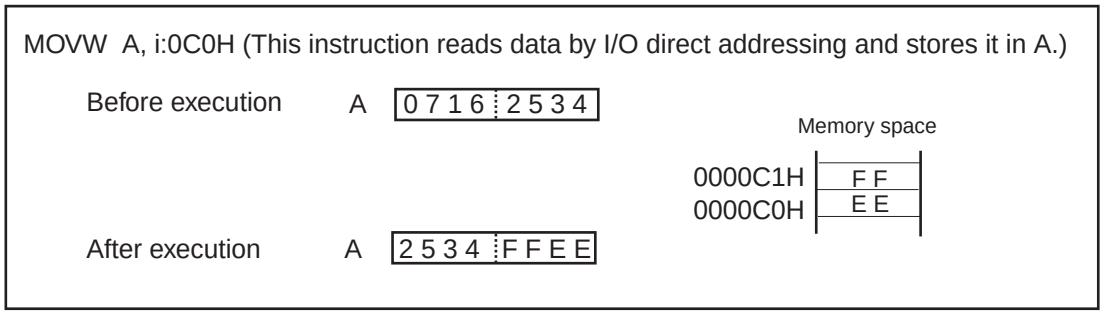
Figure B.3-4 Example of Direct Branch Addressing (addr24)



❍ I/O direct addressing (io)

Specify an 8-bit offset explicitly for the memory address in an operand. The I/O address space in the physical address space from 000000H to 0000FFH is accessed regardless of the data bank register (DTB) and direct page register (DPR). A bank select prefix for bank addressing is invalid if specified before an instruction using I/O direct addressing.

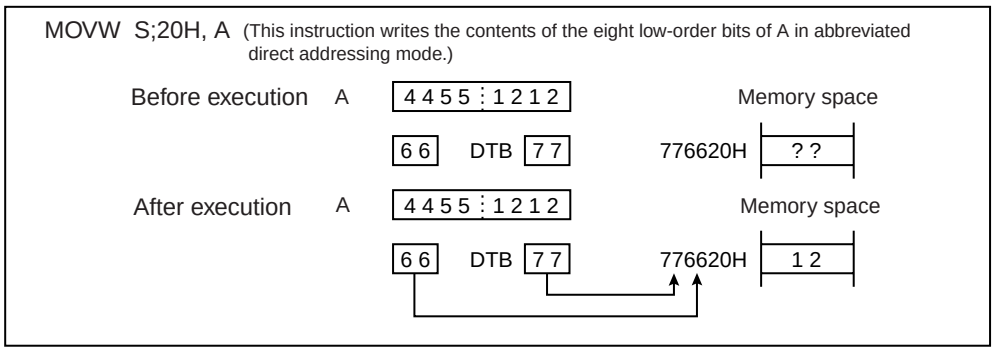
Figure B.3-5 Example of I/O Direct Addressing (io)



❍ Abbreviated direct addressing (dir)

Specify the eight low-order bits of a memory address explicitly in an operand. Address bits 8 to 15 are specified by the direct page register (DPR). Address bits 16 to 23 are specified by the data bank register (DTB).

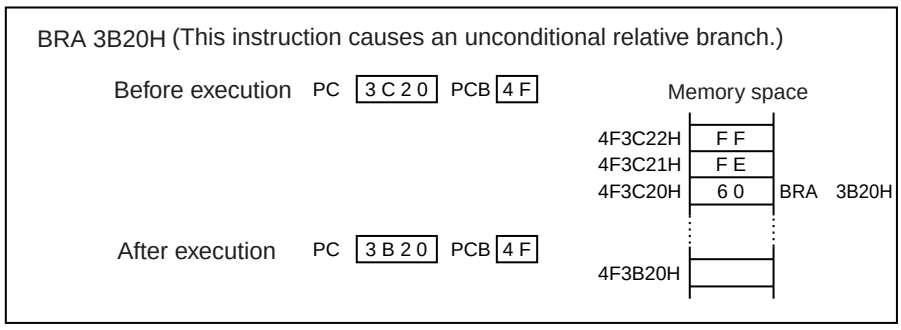
Figure B.3-6 Example of Abbreviated Direct Addressing (dir)



❍ Direct addressing (addr16)

Specify the 16 low-order bits of a memory address explicitly in an operand. Address bits 16 to 23 are specified by the data bank register (DTB). A prefix instruction for access space addressing is invalid for this mode of addressing.

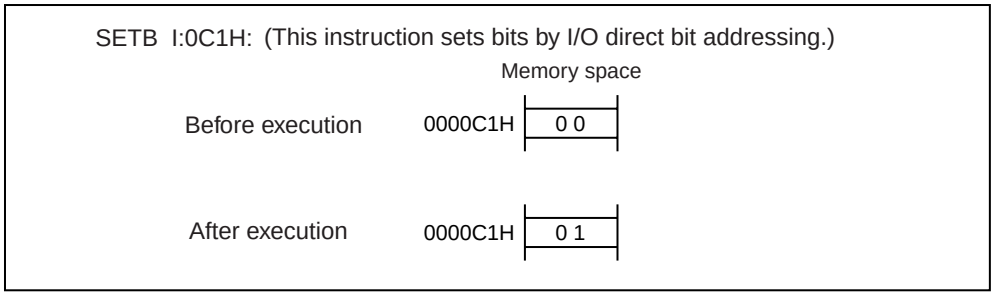
Figure B.3-7 Example of Direct Addressing (addr16)



❍ **I/O direct bit addressing (io:bp)**

Specify bits in physical addresses 000000H to 0000FFH explicitly. Bit positions are indicated by ":bp", where the larger number indicates the most significant bit (MSB) and the lower number indicates the least significant bit (LSB).

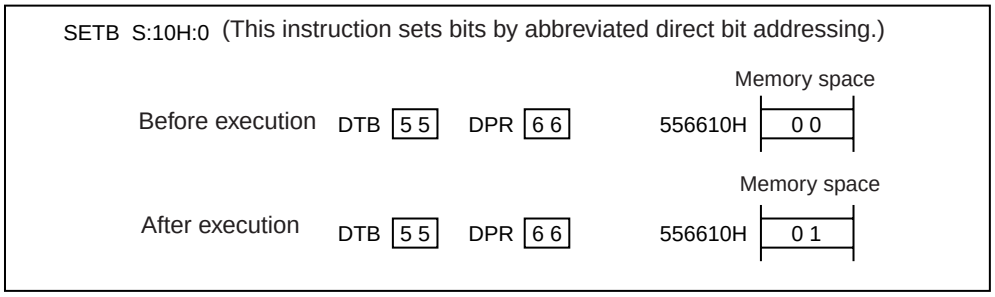
Figure B.3-8 Example of I/O Direct Bit Addressing (io:bp)



❍ **Abbreviated direct bit addressing (dir:bp)**

Specify the eight low-order bits of a memory address explicitly in an operand. Address bits 8 to 15 are specified by the direct page register (DPR). Address bits 16 to 23 are specified by the data bank register (DTB). Bit positions are indicated by ":bp", where the larger number indicates the most significant bit (MSB) and the lower number indicates the least significant bit (LSB).

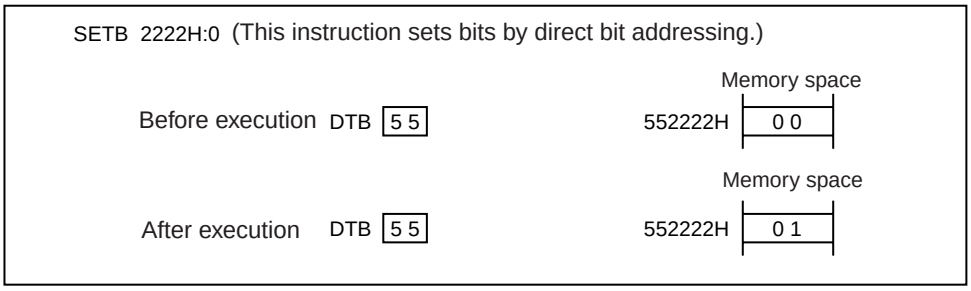
Figure B.3-9 Example of Abbreviated Direct Bit Addressing (dir:bp)



❍ **Direct bit addressing (addr16:bp)**

Specify arbitrary bits in 64 kilobytes explicitly. Address bits 16 to 23 are specified by the data bank register (DTB). Bit positions are indicated by ":bp", where the larger number indicates the most significant bit (MSB) and the lower number indicates the least significant bit (LSB).

Figure B.3-10 Example of Direct Bit addressing (addr16:bp)



❍ Vector Addressing (#vct)

Specify vector data in an operand to indicate the branch destination address. There are two sizes for vector numbers: 4 bits and 8 bits. Vector addressing is used for a subroutine call or software interrupt instruction.

Figure B.3-11 Example of Vector Addressing (#vct)

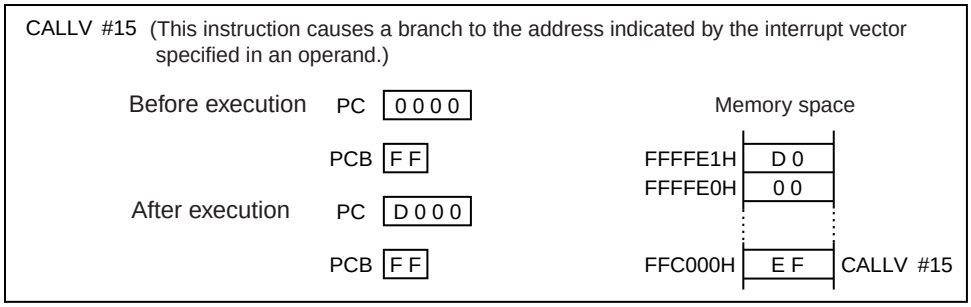


Table B.3-2 CALLV Vector List

Instruction	Vector address L	Vector address H
CALLV #0	XXFFFEH	XXFFFFH
CALLV #1	XXFFFCH	XXFFFDH
CALLV #2	XXFFFAH	XXFFFBH
CALLV #3	XXFFF8H	XXFFF9H
CALLV #4	XXFFF6H	XXFFF7H
CALLV #5	XXFFF4H	XXFFF5H
CALLV #6	XXFFF2H	XXFFF3H
CALLV #7	XXFFF0H	XXFFF1H
CALLV #8	XXFFEEH	XXFFEFH
CALLV #9	XXFFECH	XXFFEDH
CALLV #10	XXFFEAH	XXFFEBH
CALLV #11	XXFFE8H	XXFFE9H
CALLV #12	XXFFE6H	XXFFE7H
CALLV #13	XXFFE4H	XXFFE5H
CALLV #14	XXFFE2H	XXFFE3H
CALLV #15	XXFFE0H	XXFFE1H

Note: A PCB register value is set in XX.

Note:

When the program bank register (PCB) is FF_H, the vector area overlaps the vector area of INT #vct8 (#0 to #7). Use vector addressing carefully (see Table B.3-2 "CALLV Vector List").

B.4 Indirect Addressing

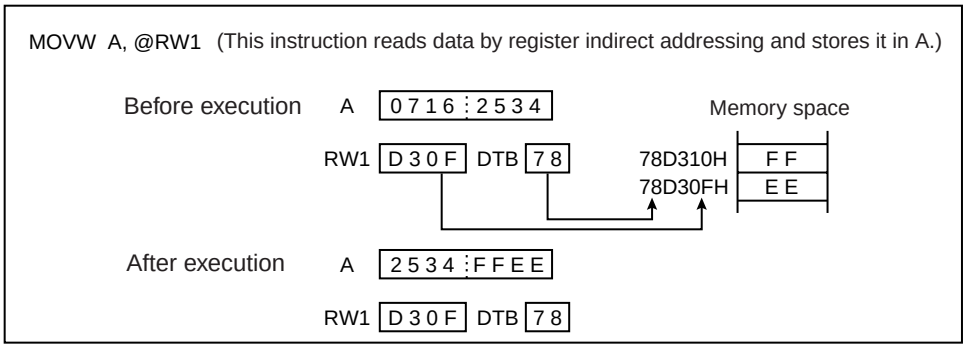
In indirect addressing mode, an address is specified indirectly by the address data of an operand.

■ Indirect Addressing

❍ Register indirect addressing (@RWj j = 0 to 3)

Memory is accessed using the contents of general-purpose register RWj as an address. Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0 or RW1 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 is used, or additional data bank register (ADB) when RW2 is used.

Figure B.4-1 Example of Register Indirect Addressing (@RWj j = 0 to 3)

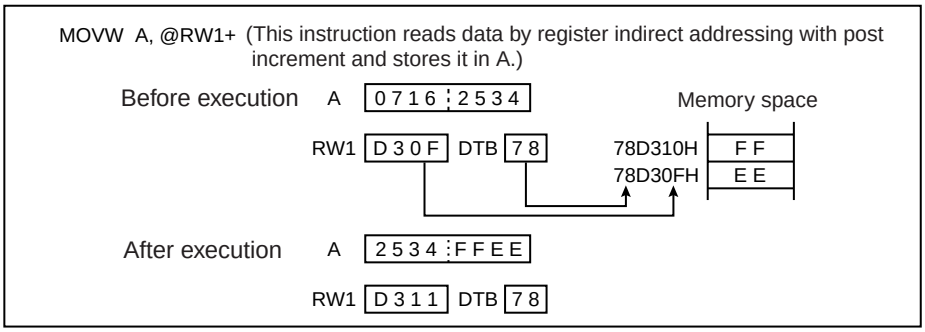


❍ Register indirect addressing with post increment (@RWj+ j = 0 to 3)

Memory is accessed using the contents of general-purpose register RWj as an address. After operand operation, RWj is incremented by the operand size (1 for a byte, 2 for a word, or 4 for a long word). Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0 or RW1 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 is used, or additional data bank register (ADB) when RW2 is used.

If the post increment results in the address of the register that specifies the increment, the incremented value is referenced after that. In this case, if the next instruction is a write instruction, priority is given to writing by an instruction and, therefore, the register that would be incremented becomes write data.

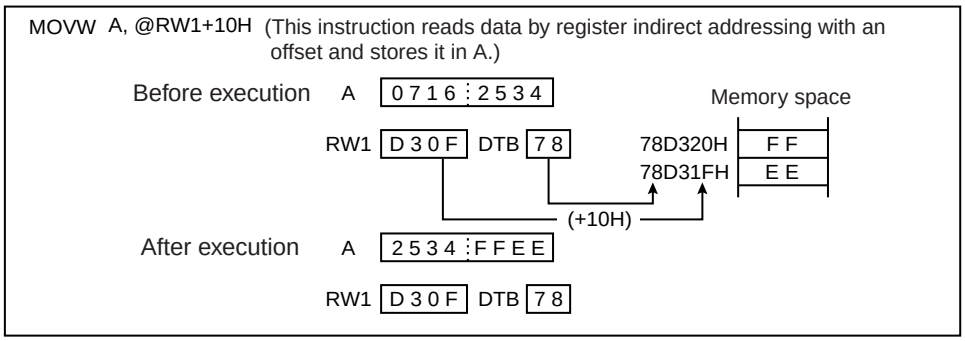
Figure B.4-2 Example of Register Indirect Addressing with Post Increment (@RWj + j = 0 to 3)



❍ **Register indirect addressing with offset (@RWi + disp8 i = 0 to 7, @RWj + disp16 j = 0 to 3)**

Memory is accessed using the address obtained by adding an offset to the contents of general-purpose register RWj. Two types of offset, byte and word offsets, are used. They are added as signed numeric values. Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0, RW1, RW4, or RW5 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 or RW7 is used, or additional data bank register (ADB) when RW2 or RW6 is used.

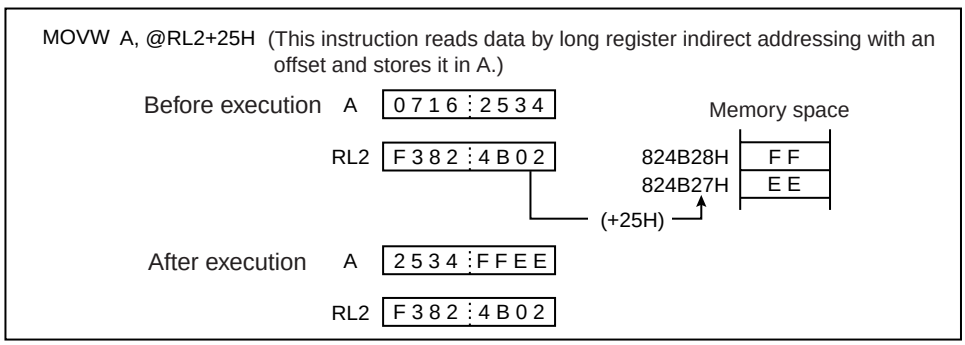
Figure B.4-3 Example of Register Indirect Addressing with Offset (@RWi + disp8 i = 0 to 7, @RWj + disp16 j = 0 to 3)



❍ **Long register indirect addressing with offset (@RLi + disp8 i = 0 to 3)**

Memory is accessed using the address that is the 24 low-order bits obtained by adding an offset to the contents of general-purpose register RLi. The offset is 8-bits long and is added as a signed numeric value.

Figure B.4-4 Example of Long Register Indirect Addressing with Offset (@RLi + disp8 i = 0 to 3)

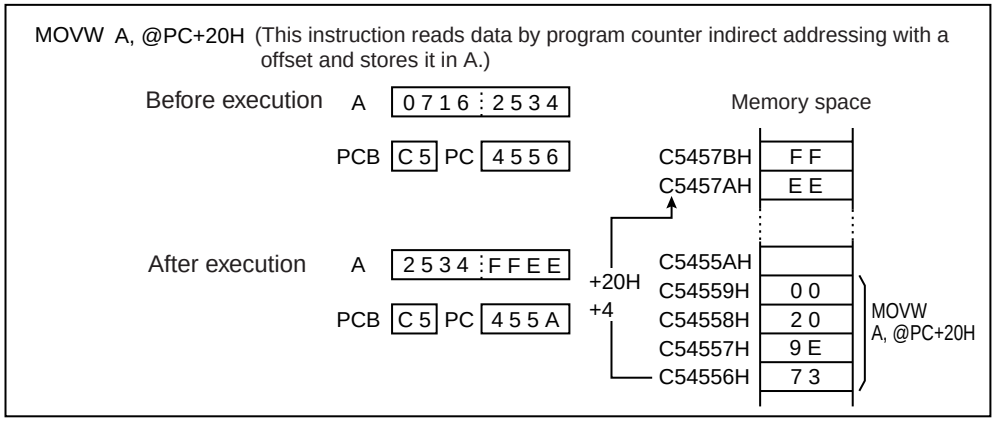


❍ **Program counter indirect addressing with offset (@PC + disp16)**

Memory is accessed using the address indicated by (instruction address + 4 + disp16). The offset is one word long. Address bits 16 to 23 are specified by the program bank register (PCB). Note that the operand address of each of the following instructions is not deemed to be (next instruction address + disp16):

- DBNZ eam, rel
- DWBNZ eam, rel
- CBNE eam, #imm8, rel
- CWBNE eam, #imm16, rel
- MOV eam, #imm8
- MOVW eam, #imm16

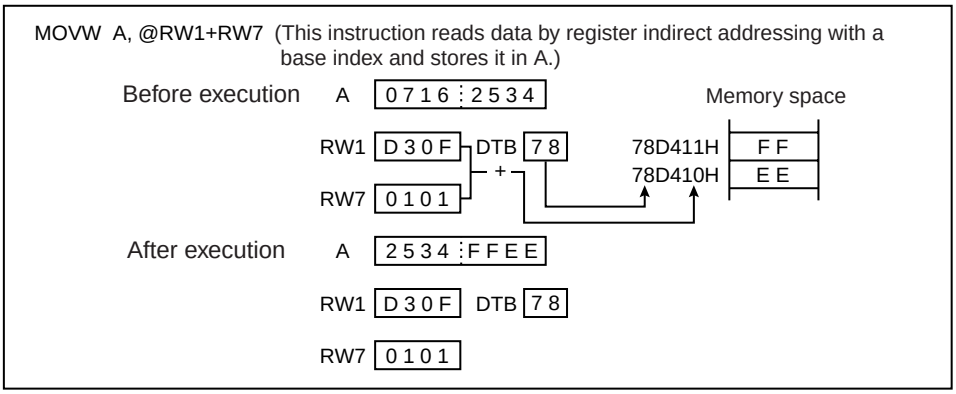
Figure B.4-5 Example of Program Counter Indirect Addressing with Offset (@PC + disp16)



❍ **Register indirect addressing with base index (@RW0 + RW7, @RW1 + RW7)**

Memory is accessed using the address determined by adding RW0 or RW1 to the contents of general-purpose register RW7. Address bits 16 to 23 are indicated by the data bank register (DTB).

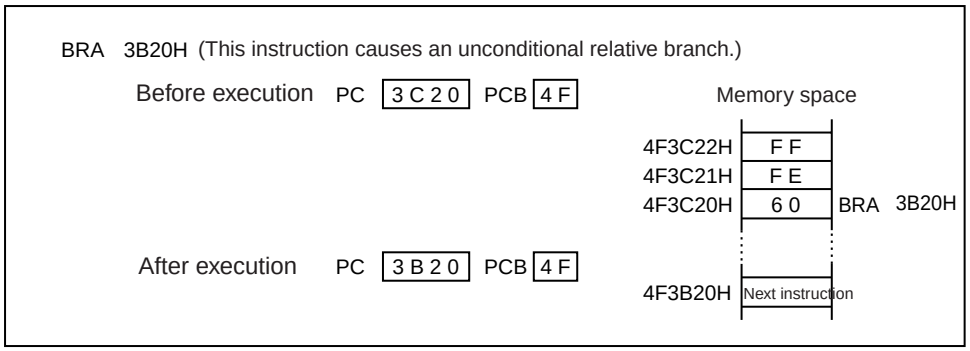
Figure B.4-6 Example of Register Indirect Addressing with Base Index (@RW0 + RW7, @RW1 + RW7)



❍ **Program counter relative branch addressing (rel)**

The address of the branch destination is a value determined by adding an 8-bit offset to the program counter (PC) value. If the result of addition exceeds 16 bits, bank register incrementing or decrementing is not performed and the excess part is ignored, and therefore the address is contained within a 64-kilobyte bank. This addressing is used for both conditional and unconditional branch instructions. Address bits 16 to 23 are indicated by the program bank register (PCB).

Figure B.4-7 Example of Program Counter Relative Branch Addressing (rel)



❍ **Register list (rlst)**

Specify a register to be pushed onto or popped from a stack.

Figure B.4-8 Configuration of the Register List

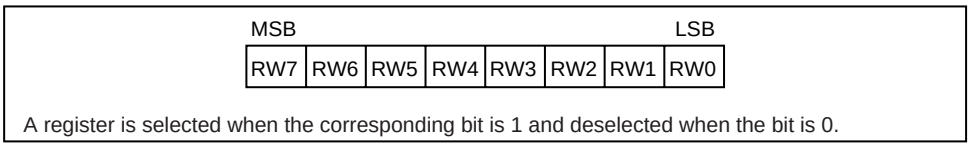
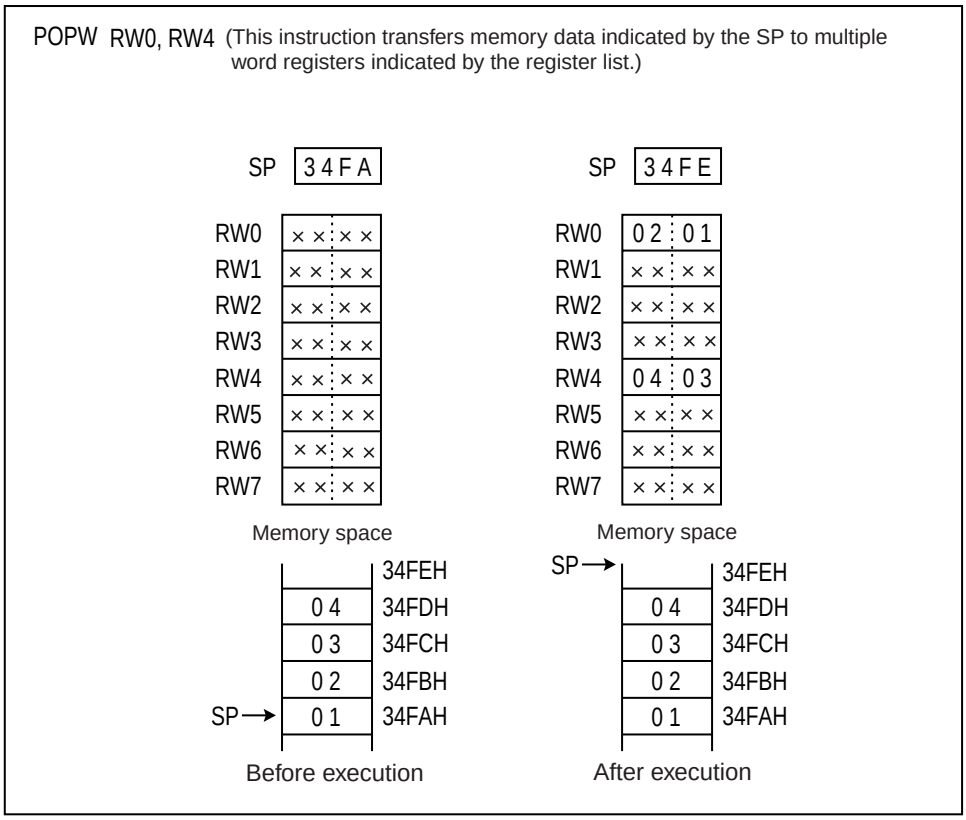


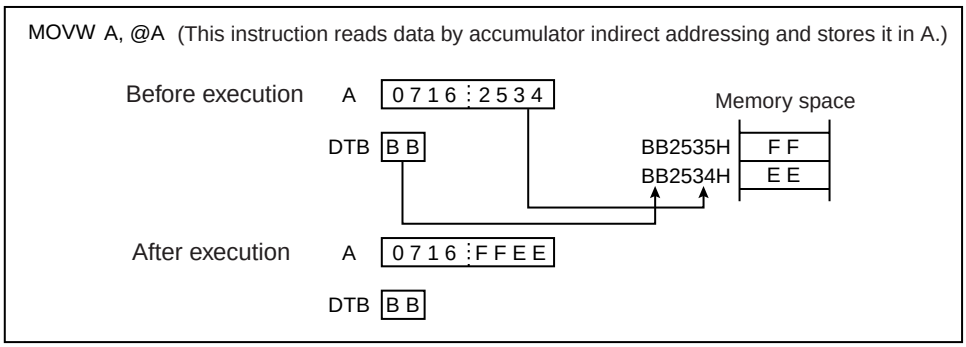
Figure B.4-9 Example of Register List (rlist)



❍ **Accumulator indirect addressing (@A)**

Memory is accessed using the address indicated by the contents of the low-order bytes (16 bits) of the accumulator (AL). Address bits 16 to 23 are specified by a mnemonic in the data bank register (DTB).

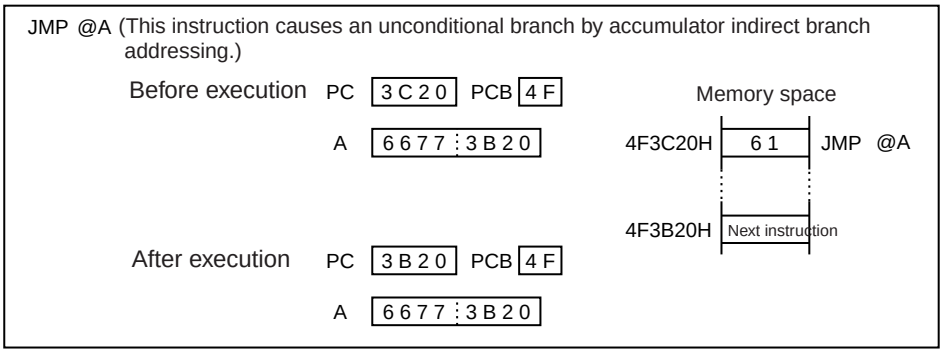
Figure B.4-10 Example of Accumulator Indirect Addressing (@A)



❍ **Accumulator indirect branch addressing (@A)**

The address of the branch destination is the content (16 bits) of the low-order bytes (AL) of the accumulator. It indicates the branch destination in the bank address space. Address bits 16 to 23 are specified by the program bank register (PCB). For the Jump Context (JCTX) instruction, however, address bits 16 to 23 are specified by the data bank register (DTB). This addressing is used for unconditional branch instructions.

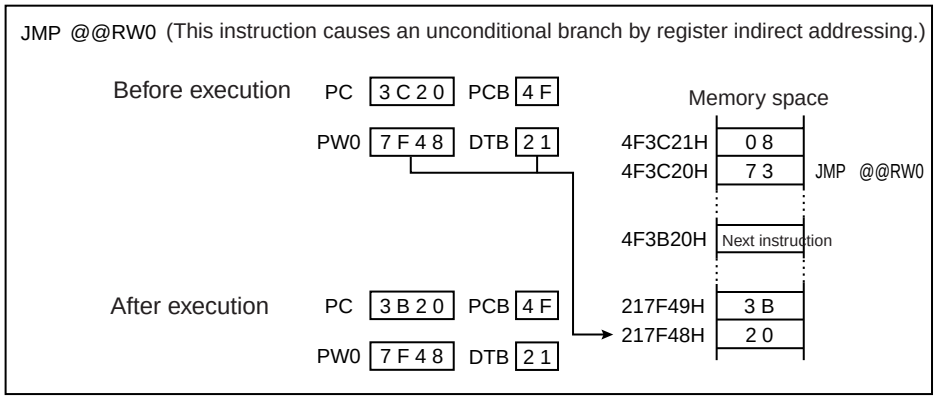
Figure B.4-11 Example of Accumulator Indirect Branch Addressing (@A)



❍ **Indirect specification branch addressing (@ear)**

The address of the branch destination is the word data at the address indicated by ear.

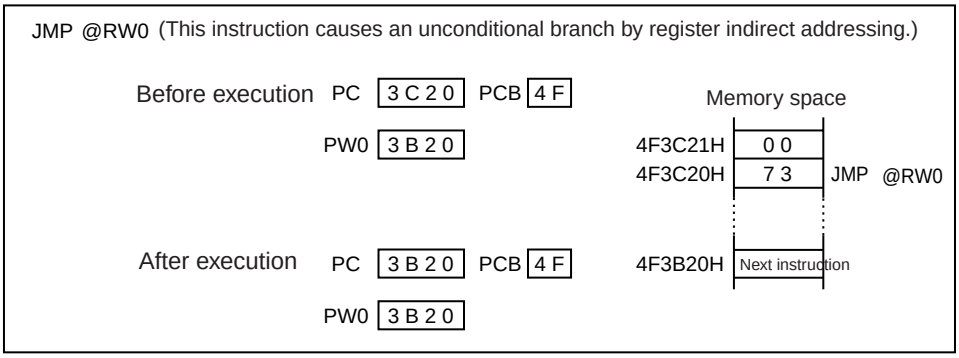
Figure B.4-12 Example of Indirect Specification Branch Addressing (@ear)



❍ **Indirect specification branch addressing (@eam)**

The address of the branch destination is the word data at the address indicated by eam.

Figure B.4-13 Example of Indirect Specification Branch Addressing (@eam)



B.5 Execution Cycle Count

The number of cycles required for instruction execution (execution cycle count) is obtained by adding the number of cycles required for each instruction, "correction value" determined by the condition, and the number of cycles for instruction fetch.

■ Execution Cycle Count

The number of cycles required for instruction execution (execution cycle count) is obtained by adding the number of cycles required for each instruction, "correction value" determined by the condition, and the number of cycles for instruction fetch. In the mode of fetching an instruction from memory such as internal ROM connected to a 16-bit bus, the program fetches the instruction being executed in word increments. Therefore, intervening in data access increases the execution cycle count.

Similarly, in the mode of fetching an instruction from memory connected to an 8-bit external bus, the program fetches every byte of an instruction being executed. Therefore, intervening in data access increases the execution cycle count. In CPU intermittent operation mode, access to a general-purpose register, internal ROM, internal RAM, internal I/O, or external data bus causes the clock to the CPU to halt for the cycle count specified by the CG0 and CG1 bits of the low power consumption mode control register. Therefore, for the cycle count required for instruction execution in CPU intermittent operation mode, add the "access count x cycle count for the halt" as a correction value to the normal execution count.

■ Calculating the Execution Cycle Count

Table B.5-1 "Execution Cycle Counts in Each Addressing Mode" lists execution cycle counts and Table B.5-2 "Cycle Count Correction Values for Counting Execution Cycles" and Table B.5-3 "Cycle Count Correction Values for Counting Instruction Fetch Cycles" summarize correction value data.

Table B.5-1 Execution Cycle Counts in Each Addressing Mode

Code	Operand	(a) (*1)	Register access count in each addressing mode
		Execution cycle count in each addressing mode	
00 | 07	Ri Rwi RLi	See the instruction list.	See the instruction list.
08 | 0B	@RWj	2	1
0C | 0F	@RWj+	4	2
10 | 17	@RWi+disp8	2	1
18 | 1B	@RWi+disp16	2	1
1C 1D 1E 1F	@RW0+RW7 @RW1+RW7 @PC+disp16 addr16	4 4 2 1	2 2 0 0

*1: (a) is used for ～ (cycle count) and B (correction value) in B.8 "F^{2}MC-16LX Instruction List".

Table B.5-2 Cycle Count Correction Values for Counting Execution Cycles

Operand	(b) byte (*1)		(c) word (*1)		(d) long (*1)	
	Cycle count	Access count	Cycle count	Access count	Cycle count	Access count
Internal register	+0	1	+0	1	+0	2
Internal memory Even address	+0	1	+0	1	+0	2
Internal memory Odd address	+0	1	+2	2	+4	4
External data bus 16-bit even address	+1	1	+1	1	+2	2
External data bus 16-bit odd address	+1	1	+4	2	+8	4
External data bus 8-bits	+1	1	+4	2	+8	4

*1: (b), (c), and (d) are used for ∼ (cycle count) and B (correction value) in B.8 "F^{2}MC-16LX Instruction List".

Note:

When an external data bus is used, the cycle counts during which an instruction is made to wait by ready input or automatic ready must also be added.

Table B.5-3 Cycle Count Correction Values for Counting Instruction Fetch Cycles

Instruction	Byte boundary	Word boundary
Internal memory	-	+2
External data bus 16-bits	-	+3
External data bus 8-bits	+3	-

Note:

- When an external data bus is used, the cycle counts during which an instruction is made to wait by ready input or automatic ready must also be added.
- Actually, instruction execution is not delayed by every instruction fetch. Therefore, use the correction values to calculate the worst case.

B.6 Effective Address Field

Table B.6-1 "Effective Address Field" shows the effective address field.

■ **Effective Address Field**

Table B.6-1 Effective Address Field

Code	Representation			Address format	Byte count of extended address part (*1)
00 01 02 03 04 05 06 07	R0 R1 R2 R3 R4 R5 R6 R7	RW0 RW1 RW2 RW3 RW4 RW5 RW6 RW7	RL0 (RL0) RL1 (RL1) RL2 (RL2) RL3 (RL3)	Register direct: Individual parts correspond to the byte, word, and long word types in order from the left.	-
08 09 0A 0B	@RW0 @RW1 @RW2 @RW3			Register indirect	0
0C 0D 0E 0F	@RW0+ @RW1+ @RW2+ @RW3+			Register indirect with post increment	0
10 11 12 13 14 15 16 17	@RW0+disp8 @RW1+disp8 @RW2+disp8 @RW3+disp8 @RW4+disp8 @RW5+disp8 @RW6+disp8 @RW7+disp8			Register indirect with 8-bit displacement	1
18 19 1A 1B	@RW0+disp16 @RW1+disp16 @RW2+disp16 @RW3+disp16			Register indirect with 16-bit displacement	2
1C 1D 1E 1F	@RW0+RW7 @RW1+RW7 @PC+disp16 addr16			Register indirect with index Register indirect with index PC indirect with 16-bit displacement Direct address	0 0 2 2

*1: Each byte count of the extended address part applies to + in the # (byte count) column in B.8 "F^{2}MC-16LX Instruction List".

B.7 How to Read the Instruction List

Table B.7-1 "Description of Items in the Instruction List" describes the items used in the F2MC-16LX Instruction List, and Table B.7-2 "Explanation on Symbols in the Instruction List" describes the symbols used in the same list.

■ Description of instruction presentation items and symbols

Table B.7-1 Description of Items in the Instruction List

Item	Description
Mnemonic	Uppercase, symbol: Represented as is in the assembler. Lowercase: Rewritten in the assembler. Number of following lowercase: Indicates bit length in the instruction.
#	Indicates the number of bytes.
~	Indicates the number of cycles. See Table B.2-1 "Effective Address Field" for the alphabetical letters in items.
RG	Indicates the number of times a register access is performed during instruction execution. The number is used to calculate the correction value for CPU intermittent operation.
B	Indicates the correction value used to calculate the actual number of cycles during instruction execution. The actual number of cycles during instruction execution can be determined by adding the value in the ~ column to this value.
Operation	Indicates the instruction operation.
LH	Indicates the special operation for bits 15 to 08 of the accumulator. Z: Transfers 0. X: Transfers after sign extension. -: No transfer
AH	Indicates the special operation for the 16 high-order bits of the accumulator. *: Transfers from AL to AH. -: No transfer Z: Transfers 00 to AH. X: Transfers 00H or FFH to AH after AL sign extension.

Table B.7-1 Description of Items in the Instruction List (Continued)

Item	Description
I	Each indicates the state of each flag: I (interrupt enable), S (stack), T (sticky bit), N (negative), Z (zero), V (overflow), C (carry). *: Changes upon instruction execution. -: No change Z: Set upon instruction execution. X: Reset upon instruction execution.
S	
T	
N	
Z	
V	
C	
RMW	Indicates whether the instruction is a Read Modify Write instruction (reading data from memory by the I instruction and writing the result to memory). *: Read Modify Write instruction -: Not Read Modify Write instruction Note: Cannot be used for an address that has different meanings between read and write operations.

Table B.7-2 Explanation on Symbols in the Instruction List

Symbol	Explanation
A	The bit length used varies depending on the 32-bit accumulator instruction. Byte: Low-order 8 bits of byte AL Word: 16 bits of word AL Long word: 32 bits of AL and AH
AH AL	16 high-order bits of A 16 low-order bits of A
SP	Stack pointer (USP or SSP)
PC	Program counter
PCB	Program bank register
DTB	Data bank register
ADB	Additional data bank register
SSB	System stack bank register
USB	User stack bank register
SPB	Current stack bank register (SSB or USB)
DPR	Direct page register
brg1	DTB, ADB, SSB, USB, DPR, PCB, SPB
brg2	DTB, ADB, SSB, USB, DPR, SPB
Ri	R0, R1, R2, R3, R4, R5, R6, R7

Table B.7-2 Explanation on Symbols in the Instruction List (Continued)

Symbol	Explanation
RWi	RW0, RW1, RW2, RW3, RW4, RW5, RW6, RW7
RWj	RW0, RW1, RW2, RW3
RLi	RL0, RL1, RL2, RL3
dir	Abbreviated direct addressing
addr16 addr24 ad24 0-15 ad24 16-23	Direct addressing Physical direct addressing Bits 0 to 15 of addr24 Bits 16 to 23 of addr24
io	I/O area (000000H to 0000FFH)
#imm4 #imm8 #imm16 #imm32 ext (imm8)	4-bit immediate data 8-bit immediate data 16-bit immediate data 32-bit immediate data 16-bit data obtained by sign extension of 8-bit immediate data
disp8 disp16	8-bit displacement 16-bit displacement
bp	Bit offset
vct4 vct8	Vector number (0 to 15) Vector number (0 to 255)
( ) b	Bit address
rel	PC relative branch
ear eam	Effective addressing (code 00 to 07) Effective addressing (code 08 to 1F)
rlst	Register list

B.8 F²MC-16LX Instruction List

Table B.8-1 "41 Transfer Instructions (byte)" to Table B.9-19 "MOVW ea, Rwi Instruction (first byte = 7DH)" list the instructions used by the F²MC-16LX.

■ F²MC-16LX Instruction List

Table B.8-1 41 Transfer Instructions (byte)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOV	A,dir	2	3	0	(b)	byte (A) <-- (dir)	Z	*	-	-	*	*	-	-	-	-
MOV	A,addr16	3	4	0	(b)	byte (A) <-- (addr16)	Z	*	-	-	*	*	-	-	-	-
MOV	A,Ri	1	2	1	0	byte (A) <-- (Ri)	Z	*	-	-	*	*	-	-	-	-
MOV	A,ear	2	2	1	0	byte (A) <-- (ear)	Z	*	-	-	*	*	-	-	-	-
MOV	A,eam	2+	3+(a)	0	(b)	byte (A) <-- (eam)	Z	*	-	-	*	*	-	-	-	-
MOV	A,io	2	3	0	(b)	byte (A) <-- (io)	Z	*	-	-	*	*	-	-	-	-
MOV	A,#imm8	2	2	0	0	byte (A) <-- imm8	Z	*	-	-	*	*	-	-	-	-
MOV	A,@A	2	3	0	(b)	byte (A) <-- ((A))	Z	-	-	-	*	*	-	-	-	-
MOV	A,@RLi+disp8	3	10	2	(b)	byte (A) <-- ((RLi)+disp8)	Z	*	-	-	*	*	-	-	-	-
MOVN	A,#imm4	1	1	0	0	byte (A) <-- imm4	Z	*	-	-	R	*	-	-	-	-
MOVX	A,dir	2	3	0	(b)	byte (A) <-- (dir)	X	*	-	-	-	*	*	-	-	-
MOVX	A,addr16	3	4	0	(b)	byte (A) <-- (addr16)	X	*	-	-	-	*	*	-	-	-
MOVX	A,Ri	2	2	1	0	byte (A) <-- (Ri)	X	*	-	-	-	*	*	-	-	-
MOVX	A,ear	2	2	1	0	byte (A) <-- (ear)	X	*	-	-	-	*	*	-	-	-
MOVX	A,eam	2+	3+(a)	0	(b)	byte (A) <-- (eam)	X	*	-	-	-	*	*	-	-	-
MOVX	A,io	2	3	0	(b)	byte (A) <-- (io)	X	*	-	-	-	*	*	-	-	-
MOVX	A,#imm8	2	2	0	0	byte (A) <-- imm8	X	*	-	-	-	*	*	-	-	-
MOVX	A,@A	2	3	0	(b)	byte (A) <-- ((A))	X	-	-	-	-	*	*	-	-	-
MOVX	A,@RWi+disp8	2	5	1	(b)	byte (A) <-- ((RWi)+disp8)	X	*	-	-	-	*	*	-	-	-
MOVX	A,@RLi+disp8	3	10	2	(b)	byte (A) <-- ((RLi)+disp8	X	*	-	-	-	*	*	-	-	-
MOV	dir,A	2	3	0	(b)	byte (dir) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOV	addr16,A	3	4	0	(b)	byte (addr16) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOV	Ri,A	1	2	1	0	byte (Ri) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOV	ear,A	2	2	1	0	byte (ear) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOV	eam,A	2+	3+(a)	0	(b)	byte (eam) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOV	io,A	2	3	0	(b)	byte (io) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOV	@RLi+disp8,A	3	10	2	(b)	byte ((RLi)+disp8) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOV	Ri,ear	2	3	2	0	byte (Ri) <-- (ear)	-	-	-	-	-	*	*	-	-	-
MOV	Ri,eam	2+	4+(a)	1	(b)	byte (Ri) <-- (eam)	-	-	-	-	-	*	*	-	-	-
MOV	ear,Ri	2	4	2	0	byte (ear) <-- (Ri)	-	-	-	-	-	*	*	-	-	-
MOV	eam,Ri	2+	5+(a)	1	(b)	byte (eam) <-- (Ri)	-	-	-	-	-	*	*	-	-	-
MOV	Ri,#imm8	2	2	1	0	byte (Ri) <-- imm8	-	-	-	-	-	*	*	-	-	-
MOV	io,#imm8	3	5	0	(b)	byte (io) <-- imm8	-	-	-	-	-	-	-	-	-	-
MOV	dir,#imm8	3	5	0	(b)	byte (dir) <-- imm8	-	-	-	-	-	-	-	-	-	-
MOV	ear,#imm8	3	2	1	0	byte (ear) <-- imm8	-	-	-	-	-	*	*	-	-	-
MOV	eam,#imm8	3+	4+(a)	0	(b)	byte (eam) <-- imm8	-	-	-	-	-	-	-	-	-	-
MOV	@AL,AH/ MOV @A,T	2	3	0	(b)	byte ((A)) <-- (AH)	-	-	-	-	-	*	*	-	-	-
XCH	A,ear	2	4	2	0	byte (A) <--> (ear)	Z	-	-	-	-	-	-	-	-	-
XCH	A,eam	2+	5+(a)	0	2×(b)	byte (A) <--> (eam)	Z	-	-	-	-	-	-	-	-	-
XCH	Ri,ear	2	7	4	0	byte (Ri) <--> (ear)	-	-	-	-	-	-	-	-	-	-
XCH	Ri,eam	2+	9+(a)	2	2×(b)	byte (Ri) <--> (eam)	-	-	-	-	-	-	-	-	-	-

Note:

See Table B.5-1 "Execution Cycle Counts in Each Addressing Mode" and Table B.5-2 "Cycle Count Correction Values for Counting Execution Cycles" for information on (a) to (d) in the table.

Table B.8-2 38 Transfer Instructions (byte)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVW	A,dir	2	3	0	(c)	word (A) <-- (dir)	-	*	-	-	-	*	*	-	-	-
MOVW	A,addr16	3	4	0	(c)	word (A) <-- (addr16)	-	*	-	-	-	*	*	-	-	-
MOVW	A,SP	1	1	0	0	word (A) <-- (SP)	-	*	-	-	-	*	*	-	-	-
MOVW	A,RWi	1	2	1	0	word (A) <-- (RWi)	-	*	-	-	-	*	*	-	-	-
MOVW	A,ear	2	2	1	0	word (A) <-- (ear)	-	*	-	-	-	*	*	-	-	-
MOVW	A,eam	2+	3+(a)	0	(c)	word (A) <-- (eam)	-	*	-	-	-	*	*	-	-	-
MOVW	A,io	2	3	0	(c)	word (A) <-- (io)	-	*	-	-	-	*	*	-	-	-
MOVW	A,@A	2	3	0	(c)	word (A) <-- ((A))	-	-	-	-	-	*	*	-	-	-
MOVW	A,#imm16	3	2	0	0	word (A) <-- imm16	-	*	-	-	-	*	*	-	-	-
MOVW	A,@RWi+disp8	2	5	1	(c)	word (A) <-- ((RWi)+disp8)	-	*	-	-	-	*	*	-	-	-
MOVW	A,@RLi+disp8	3	10	2	(c)	word (A) <-- ((RLi)+disp8)	-	*	-	-	-	*	*	-	-	-
MOVW	dir,A	2	3	0	(c)	word (dir) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW	addr16,A	3	4	0	(c)	word (addr16) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW	SP,A	1	1	0	0	word (SP) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW	RWi,A	1	2	1	0	word (RWi) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW	ear,A	2	2	1	0	word (ear) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW	eam,A	2+	3+(a)	0	(c)	word (eam) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW	io,A	2	3	0	(c)	word (io) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW	@RWi+disp8,A	2	5	1	(c)	word ((RWi)+disp8) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW	@RLi+disp8,A	3	10	2	(c)	word ((RLi)+disp8) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW	RWi,ear	2	3	2	0	word (RWi) <-- (ear)	-	-	-	-	-	*	*	-	-	-
MOVW	RWi,eam	2+	4+(a)	1	(c)	word (RWi) <-- (eam)	-	-	-	-	-	*	*	-	-	-
MOVW	ear,Rwi	2	4	2	0	word (ear) <-- (RWi)	-	-	-	-	-	*	*	-	-	-
MOVW	eam,Rwi	2+	5+(a)	1	(c)	word (eam) <-- (RWi)	-	-	-	-	-	*	*	-	-	-
MOVW	RWi,#imm16	3	2	1	0	word (RWi) <-- imm16	-	-	-	-	-	*	*	-	-	-
MOVW	io,#imm16	4	5	0	(c)	word (io) <-- imm16	-	-	-	-	-	-	-	-	-	-
MOVW	ear,#imm16	4	2	1	0	word (ear) <-- imm16	-	-	-	-	-	*	*	-	-	-
MOVW	eam,#imm16	4+	4+(a)	0	(c)	word (eam) <-- imm16	-	-	-	-	-	-	-	-	-	-
MOVW	@AL,AH/MOVW @A,T	2	3	0	(c)	word ((A)) <-- (AH)	-	-	-	-	-	*	*	-	-	-
XCHW	A,ear	2	4	2	0	word (A) <--> (ear)	-	-	-	-	-	-	-	-	-	-
XCHW	A,eam	2+	5+(a)	0	2 x (c)	word (A) <-- >(eam)	-	-	-	-	-	-	-	-	-	-
XCHW	RWi, ear	2	7	4	0	word (RWi) <--> (ear)	-	-	-	-	-	-	-	-	-	-
XCHW	RWi, eam	2+	9+(a)	2	2 x (c)	word (RWi) <--> (eam)	-	-	-	-	-	-	-	-	-	-
MOVL	A,ear	2	4	2	0	long (A) <-- (ear)	-	-	-	-	-	*	*	-	-	-
MOVL	A,eam	2+	5+(a)	0	(d)	long (A) <-- (eam)	-	-	-	-	-	*	*	-	-	-
MOVL	A,#imm32	5	3	0	0	long (A) <-- imm32	-	-	-	-	-	*	*	-	-	-
MOVL	ear,A	2	4	2	0	long (ear1) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVL	eam,A	2+	5+(a)	0	(d)	long(eam1) <-- (A)	-	-	-	-	-	*	*	-	-	-

Note:

See Table B.5-1 "Execution Cycle Counts in Each Addressing Mode" and Table B.5-2 "Cycle Count Correction Values for Counting Execution Cycles" for information on (a) to (d) in the table.

Table B.8-3 42 Addition/subtraction Instructions (byte, word, long word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ADD	A,#imm8	2	2	0	0	byte (A) <-- (A) + imm8	Z	-	-	-	-	*	*	*	*	-
ADD	A,dir	2	5	0	(b)	byte (A) <-- (A) + (dir)	Z	-	-	-	-	*	*	*	*	-
ADD	A,ear	2	3	1	0	byte (A) <-- (A) + (ear)	Z	-	-	-	-	*	*	*	*	-
ADD	A,eam	2+	4+(a)	0	(b)	byte (A) <-- (A) + (eam)	Z	-	-	-	-	*	*	*	*	-
ADD	ear,A	2	3	2	0	byte (ear) <-- (ear) + (A)	-	-	-	-	-	*	*	*	*	-
ADD	eam,A	2+	5+(a)	0	2×(b)	byte (eam) <-- (eam) + (A)	Z	-	-	-	-	*	*	*	*	*
ADDC	A	1	2	0	0	byte (A) <-- (AH) + (AL) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDC	A,ear	2	3	1	0	byte (A) <-- (A) + (ear)+ (C)	Z	-	-	-	-	*	*	*	*	-
ADDC	A,eam	2+	4+(a)	0	(b)	byte (A) <-- (A) + (eam)+ (C)	Z	-	-	-	-	*	*	*	*	-
ADDDC	A	1	3	0	0	byte (A) <-- (AH) + (AL) + (C) (decimal)	Z	-	-	-	-	*	*	*	*	-
SUB	A,#imm8	2	2	0	0	byte (A) <-- (A) - imm8	Z	-	-	-	-	*	*	*	*	-
SUB	A,dir	2	5	0	(b)	byte (A) <-- (A) - (dir)	Z	-	-	-	-	*	*	*	*	-
SUB	A,ear	2	3	1	0	byte (A) <-- (A) - (ear)	Z	-	-	-	-	*	*	*	*	-
SUB	A,eam	2+	4+(a)	0	(b)	byte (A) <-- (A) - (eam)	Z	-	-	-	-	*	*	*	*	-
SUB	ear,A	2	3	2	0	byte (ear) <-- (ear) - (A)	-	-	-	-	-	*	*	*	*	-
SUB	eam,A	2+	5+(a)	0	2×(b)	byte (eam) <-- (eam) - (A)	-	-	-	-	-	*	*	*	*	*
SUBC	A	1	2	0	0	byte (A) <-- (AH) - (AL) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC	A,ear	2	3	1	0	byte (A) <-- (A) - (ear) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC	A,eam	2+	4+(a)	0	(b)	byte (A) <-- (A) - (eam) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBDC	A	1	3	0	0	byte (A) <-- (AH) - (AL) - (C) (decimal)	Z	-	-	-	-	*	*	*	*	-
ADDW	A	1	2	0	0	word (A) <-- (AH) + (AL)	-	-	-	-	-	*	*	*	*	-
ADDW	A,ear	2	3	1	0	word (A) <-- (A) + (ear)	-	-	-	-	-	*	*	*	*	-
ADDW	A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) + (eam)	-	-	-	-	-	*	*	*	*	-
ADDW	A,#imm16	3	2	0	0	word (A) <-- (A) + imm16	-	-	-	-	-	*	*	*	*	-
ADDW	ear,A	2	3	2	0	word (ear) <-- (ear) + (A)	-	-	-	-	-	*	*	*	*	-
ADDW	eam,A	2+	5+(a)	0	2×(c)	word (eam) <-- (eam) + (A)	-	-	-	-	-	*	*	*	*	*
ADDCW	A,ear	2	3	1	0	word (A) <-- (A) + (ear) + (C)	-	-	-	-	-	*	*	*	*	-
ADDCW	A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) + (eam) + (C)	-	-	-	-	-	*	*	*	*	-
SUBW	A	1	2	0	0	word (A) <-- (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
SUBW	A,ear	2	3	1	0	word (A) <-- (A) - (ear)	-	-	-	-	-	*	*	*	*	-
SUBW	A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) - (eam)	-	-	-	-	-	*	*	*	*	-
SUBW	A,#imm16	3	2	0	0	word (A) <-- (A) - imm16	-	-	-	-	-	*	*	*	*	-
SUBW	ear,A	2	3	2	0	word (ear) <-- (ear) - (A)	-	-	-	-	-	*	*	*	*	-
SUBW	eam,A	2+	5+(a)	0	2×(c)	word (eam) <-- (eam) - (A)	-	-	-	-	-	*	*	*	*	*
SUBCW	A,ear	2	3	1	0	word (A) <-- (A) - (ear) - (C)	-	-	-	-	-	*	*	*	*	-
SUBCW	A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) - (eam) - (C)	-	-	-	-	-	*	*	*	*	-
ADDL	A,ear	2	6	2	0	long (A) <-- (A) + (ear)	-	-	-	-	-	*	*	*	*	-
ADDL	A,eam	2+	7+(a)	0	(d)	long (A) <-- (A) + (eam)	-	-	-	-	-	*	*	*	*	-
ADDL	A,#imm32	5	4	0	0	long (A) <-- (A) + imm32	-	-	-	-	-	*	*	*	*	-
SUBL	A,ear	2	6	2	0	long (A) <-- (A) - (ear)	-	-	-	-	-	*	*	*	*	-
SUBL	A,eam	2+	7+(a)	0	(d)	long (A) <-- (A) - (eam)	-	-	-	-	-	*	*	*	*	-
SUBL	A,#imm32	5	4	0	0	long (A) <-- (A) - imm32	-	-	-	-	-	*	*	*	*	-

Note:

See Table B.5-1 "Execution Cycle Counts in Each Addressing Mode" and Table B.5-2 "Cycle Count Correction Values for Counting Execution Cycles" for information on (a) to (d) in the table.

Table B.8-4 12 Increment/decrement Instructions (byte, word, long word)

Mnemonic		#	~	RG	B	Operation	L H	A H	I	S	T	N	Z	V	C	R M W
INC	ear	2	3	2	0	byte (ear) <-- (ear) + 1	-	-	-	-	-	*	*	*	-	-
INC	eam	2+	5+(a)	0	2×(b) 0	byte (eam) <-- (eam) + 1	-	-	-	-	-	*	*	*	-	*
DEC	ear	2	3	2	2×(b)	byte (ear) <-- (ear) - 1	-	-	-	-	-	*	*	*	-	-
DEC	eam	2+	5+(a)	0		byte (eam) <-- (eam) - 1	-	-	-	-	-	*	*	*	-	*
INCW	ear	2	3	2	0	word (ear) <-- (ear) + 1	-	-	-	-	-	*	*	*	-	-
INCW	eam	2+	5+(a)	0	2×(c)	word (eam) <-- (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECW	ear	2	3	2	0	word (ear) <-- (ear) - 1	-	-	-	-	-	*	*	*	-	-
DECW	eam	2+	5+(a)	0	2×(c)	word (eam) <-- (eam) - 1	-	-	-	-	-	*	*	*	-	*
INCL	ear	2	7	4	0	long (ear) <-- (ear) + 1	-	-	-	-	-	*	*	*	-	-
INCL	eam	2+	9+(a)	0	2×(d)	long (eam) <-- (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECL	ear	2	7	4	0	long (ear) <-- (ear) - 1	-	-	-	-	-	*	*	*	-	-
DECL	eam	2+	9+(a)	0	2×(d)	long (eam) <-- (eam) - 1	-	-	-	-	-	*	*	*	-	*

Note:

See Table B.5-1 "Execution Cycle Counts in Each Addressing Mode" and Table B.5-2 "Cycle Count Correction Values for Counting Execution Cycles" for information on (a) to (d) in the table.

Table B.8-5 11 Compare Instructions (byte, word, long word)

Mnemonic		#	~	RG	B	Operation	L H	A H	I	S	T	N	Z	V	C	R M W
CMP	A	1	1	0	0	byte (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
CMP	A,ear	2	2	1	0	byte (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMP	A,eam	2+	3+(a)	0	(b)	byte (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMP	A,#imm8	2	2	0	0	byte (A) - imm8	-	-	-	-	-	*	*	*	*	-
CMPW	A	1	1	0	0	word (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
CMPW	A,ear	2	2	1	0	word (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMPW	A,eam	2+	3+(a)	0	(c)	word (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMPW	A,#imm16	3	2	0	0	word (A) - imm16	-	-	-	-	-	*	*	*	*	-
CMPL	A,ear	2	6	2	0	long (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMPL	A,eam	2+	7+(a)	0	(d)	long (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMPL	A,#imm32	5	3	0	0	long (A) - imm32	-	-	-	-	-	*	*	*	*	-

Note:

See Table B.5-1 "Execution Cycle Counts in Each Addressing Mode" and Table B.5-2 "Cycle Count Correction Values for Counting Execution Cycles" for information on (a) to (d) in the table.

Table B.8-6 11 unsigned multiplication/division instructions (word, long word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
DIVU	A	1	*1	0	0	word (AH) / byte (AL) quotient --> byte (AL) remainder --> byte (AH)	-	-	-	-	-	-	-	*	*	-
DIVU	A,ear	2	*2	1	0	word (A) / byte (ear) quotient --> byte (A) remainder --> byte (ear)	-	-	-	-	-	-	-	*	*	-
DIVU	A,eam	2+	*3	0	*6	word (A) / byte (eam) quotient --> byte (A) remainder --> byte (eam)	-	-	-	-	-	-	-	*	*	-
DIVUW	A,ear	2	*4	1	0	long (A) / word (ear) quotient --> word(A) remainder --> word(ear)	-	-	-	-	-	-	-	*	*	-
DIVUW	A,eam	2+	*5	0	*7	long (A) / word (eam) quotient --> word(A) remainder --> word(eam)	-	-	-	-	-	-	-	*	*	-
MULU	A	1	*8	0	0	byte (AH) * byte (AL) --> word (A)	-	-	-	-	-	-	-	-	-	-
MULU	A,ear	2	*9	1	0	byte (A) * byte (ear) --> word (A)	-	-	-	-	-	-	-	-	-	-
MULU	A,eam	2+	*10	0	(b)	byte (A) * byte (eam) --> word (A)	-	-	-	-	-	-	-	-	-	-
MULUW	A	1	*11	0	0	word (AH) * word (AL) --> Long (A)	-	-	-	-	-	-	-	-	-	-
MULUW	A,ear	2	*12	1	0	word (A) * word (ear) --> Long (A)	-	-	-	-	-	-	-	-	-	-
MULUW	A,eam	2+	*13	0	(c)	word (A) * word (eam) --> Long (A)										

*1: 3: Division by 0 7: Overflow 15: Normal
*2: 4: Division by 0 8: Overflow 16: Normal
*3: 6+(a): Division by 0 9+(a): Overflow 19+(a): Normal
*4: 4: Division by 0 7: Overflow 22: Normal
*5: 6+(a): Division by 0 8+(a): Overflow 26+(a): Normal
*6: (b): Division by 0 or overflow 2 x (b): Normal
*7: (c): Division by 0 or overflow 2 x (c): Normal
*8: 3: Byte (AH) is 0. 7: Byte (AH) is not 0.
*9: 4: Byte (ear) is 0. 8: Byte (ear) is not 0.
*10: 5+(a): Byte (eam) is 0, 9+(a): Byte (eam) is not 0.
*11: 3: Word(AH) is 0. 11: Word (AH) is not 0.
*12: 4: Word(ear) is 0. 12: Word (ear) is not 0.
*13: 5+(a): Word (eam) is 0. 13+(a): Word (eam) is not 0.

Note:

See Table B.5-1 "Execution Cycle Counts in Each Addressing Mode" and Table B.5-2 "Cycle Count Correction Values for Counting Execution Cycles" for information on (a) to (d) in the table.

Table B.8-7 11 Signed Multiplication/Division Instructions (word, long word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
DIVU	A	2	*1	0	0	word (AH) / byte (AL) quotient --> byte (AL) remainder --> byte (AH)	Z	-	-	-	-	-	-	*	*	-
DIVU	A,ear	2	*2	1	0	word (A) / byte (ear) quotient --> byte (A) remainder --> byte (ear)	Z	-	-	-	-	-	-	*	*	-
DIVU	A,eam	2+	*3	0	*6	word (A) / byte (eam) quotient --> byte (A) remainder --> byte (eam)	Z	-	-	-	-	-	-	*	*	-
DIVUW	A,ear	2	*4	1	0	long (A) / word (ear) quotient --> word(A) remainder --> word(ear)	-	-	-	-	-	-	-	*	*	-
DIVUW	A,eam	2+	*5	0	*7	long (A) / word (eam) quotient --> word(A) remainder --> word(eam)	-	-	-	-	-	-	-	*	*	-

Table B.8-7 11 Signed Multiplication/Division Instructions (word, long word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MUL	A	2	*8	0	0	byte (AH) * byte (AL) --> word (A)	-	-	-	-	-	-	-	-	-	-
MUL	A,ear	2	*9	1	0	byte (A) * byte (ear) --> word (A)		-	-	-	-	-	-	-	-	-
MUL	A,eam	2+	*10	0	(b)	byte (A) * byte (eam) --> word (A)	-	-	-	-	-	-	-	-	-	-
MULW	A	2	*11	0	0	word (AH) * word (AL) --> Long (A)		-	-	-	-	-	-	-	-	-
MULW	A,ear	2	*12	1	0	word (A) * word (ear) --> Long (A)	-	-	-	-	-	-	-	-	-	-
MULW	A,eam	2+	*13	0	(c)	word (A) * word (eam) --> Long (A)	-									
							-									

*1: 3: Division by 0, 8 or 18: Overflow, 18: Normal
*2: 4: Division by 0, 11 or 22: Overflow, 23: Normal
*3: 5+(a): Division by 0, 12+(a) or 23+(a): Overflow, 24+(a): Normal
*4: When dividend is positive; 4: Division by 0, 12 or 30: Overflow, 31: Normal
When dividend is negative; 4: Division by 0, 12 or 31: Overflow, 32: Normal
*5: When dividend is positive; 5+(a): Division by 0, 12+(a) or 31+(a): Overflow, 32+(a): Normal
When dividend is negative; 5+(a): Division by 0, 12+(a) or 32+(a): Overflow, 33+(a): Normal
*6: (b): Division by 0 or overflow, 2 x (b): Normal
*7: (c): Division by 0 or overflow, 2 x (c): Normal
*8: 3: Byte (AH) is 0, 12: result is positive, 13: result is negative
*9: 4: Byte (ear) is 0, 13: result is positive, 14: result is negative
*10: 5+(a): Byte (eam) is 0, 14+(a): result is positive, 15+(a): result is negative
*11: 3: Word(AH) is 0, 16: result is positive, 19: result is negative
*12: 4: Word(ear) is 0, 17: result is positive, 20: result is negative
*13: 5+(a): Word(eam) is 0, 18+(a): result is positive, 21+(a): result is negative

Note:

- The execution cycle count found when an overflow occurs in a DIV or DIVW instruction may be a pre-operation count or a post-operation count depending on the detection timing.
- When an overflow occurs with DIV or DIVW instruction, the contents of the AL are destroyed.
- See Table B.5-1 "Execution Cycle Counts in Each Addressing Mode" and Table B.5-2 "Cycle Count Correction Values for Counting Execution Cycles" for information on (a) to (d) in the table.

Table B.8-8 39 Logic 1 Instructions (byte, word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
AND	A,#imm8	2	2	0	0	byte (A) <-- (A) and imm8	-	-	-	-	-	*	*	R	-	-
AND	A,ear	2	3	1	0	byte (A) <-- (A) and (ear)	-	-	-	-	-	*	*	R	-	-
AND	A,eam	2+	4+(a)	0	(b)	byte (A) <-- (A) and (eam)	-	-	-	-	-	*	*	R	-	-
AND	ear,A	2	3	2	0	byte (ear) <-- (ear)and (A)	-	-	-	-	-	*	*	R	-	-
AND	eam,A	2+	5+(a)	0	2×(b)	byte (eam) <-- (eam)and (A)	-	-	-	-	-	*	*	R	-	*
OR	A,#imm8	2	2	0	0	byte (A) <-- (A) or imm8	-	-	-	-	-	*	*	R	-	-
OR	A,ear	2	3	1	0	byte (A) <-- (A) or (ear)	-	-	-	-	-	*	*	R	-	-
OR	A,eam	2+	4+(a)	0	(b)	byte (A) <-- (A) or (eam)	-	-	-	-	-	*	*	R	-	-
OR	ear,A	2	3	2	0	byte (ear) <-- (ear)or (A)	-	-	-	-	-	*	*	R	-	-
OR	eam,A	2+	5+(a)	0	2×(b)	byte (eam) <-- (eam)or (A)	-	-	-	-	-	*	*	R	-	*
XOR	A,#imm8	2	2	0	0	byte (A) <-- (A) xor imm8	-	-	-	-	-	*	*	R	-	-
XOR	A,ear	2	3	1	0	byte (A) <-- (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XOR	A,eam	2+	4+(a)	0	(b)	byte (A) <-- (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XOR	ear,A	2	3	2	0	byte (ear) <-- (ear)xor (A)	-	-	-	-	-	*	*	R	-	-
XOR	eam,A	2+	5+(a)	0	2×(b)	byte (eam) <-- (eam)xor (A)	-	-	-	-	-	*	*	R	-	*
NOT	A	1	2	0	0	byte (A) <-- not (A)	-	-	-	-	-	*	*	R	-	-
NOT	ear	2	3	2	0	byte (ear) <-- not (ear)	-	-	-	-	-	*	*	R	-	-
NOT	eam	2+	5+(a)	0	2×(b)	byte (eam) <-- not (eam)	-	-	-	-	-	*	*	R	-	*
ANDW	A	1	2	0	0	word (A) <-- (AH) and (A)	-	-	-	-	-	*	*	R	-	-
ANDW	A,#imm16	3	2	0	0	word (A) <-- (A) and imm16	-	-	-	-	-	*	*	R	-	-
ANDW	A,ear	2	3	1	0	word (A) <-- (A) and (ear)	-	-	-	-	-	*	*	R	-	-
ANDW	A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) and (eam)	-	-	-	-	-	*	*	R	-	-
ANDW	ear,A	2	3	2	0	word (ear) <-- (ear)and (A)	-	-	-	-	-	*	*	R	-	-
ANDW	eam,A	2+	5+(a)	0	2×(c)	word (eam) <-- (eam)and (A)	-	-	-	-	-	*	*	R	-	*
ORW	A	1	2	0	0	word (A) <-- (AH) or (A)	-	-	-	-	-	*	*	R	-	-
ORW	A,#imm16	3	2	0	0	word (A) <-- (A) or imm16	-	-	-	-	-	*	*	R	-	-
ORW	A,ear	2	3	1	0	word (A) <-- (A) or (ear)	-	-	-	-	-	*	*	R	-	-
ORW	A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) or (eam)	-	-	-	-	-	*	*	R	-	-
ORW	ear,A	2	3	2	0	word (ear) <-- (ear)or (A)	-	-	-	-	-	*	*	R	-	-
ORW	eam,A	2+	5+(a)	0	2×(c)	word (eam) <-- (eam)or (A)	-	-	-	-	-	*	*	R	-	*
XORW	A	1	2	0	0	word (A) <-- (AH) xor (A)	-	-	-	-	-	*	*	R	-	-
XORW	A,#imm16	3	2	0	0	word (A) <-- (A) xor imm16	-	-	-	-	-	*	*	R	-	-
XORW	A,ear	2	3	1	0	word (A) <-- (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XORW	A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XORW	ear,A	2	3	2	0	word (ear) <-- (ear)xor (A)	-	-	-	-	-	*	*	R	-	*
XORW	eam,A	2+	5+(a)	0	2×(c)	word (eam) <-- (eam)xor (A)	-	-	-	-	-	*	*	R	-	-
NOTW	A	1	2	0	0	word (A) <-- not (A)	-	-	-	-	-	*	*	R	-	-
NOTW	ear	2	3	2	0	word (ear) <-- not (ear)	-	-	-	-	-	*	*	R	-	*
NOTW	eam	2+	5+(a)	0	2×(c)	word (eam) <-- not (eam)										

Note:

See Table B.5-1 "Execution Cycle Counts in Each Addressing Mode" and Table B.5-2 "Cycle Count Correction Values for Counting Execution Cycles" for information on (a) to (d) in the table.

Table B.8-9 Six Logic 2 Instructions (long word)

Mnemonic		#	～	RG	B	Operation	L H	A H	I	S	T	N	Z	V	C	R M W
ANDL	A,ear	2	6	2	0	long (A) <-- (A) and (ear)	-	-	-	-	-	*	*	R	-	-
ANDL	A,eam	2+	7+(a)	0	(d)	long (A) <-- (A) and (eam)	-	-	-	-	-	*	*	R	-	-
ORL	A,ear	2	6	2	0	long (A) <-- (A) or (ear)	-	-	-	-	-	*	*	R	-	-
ORL	A,eam	2+	7+(a)	0	(d)	long (A) <-- (A) or (eam)	-	-	-	-	-	*	*	R	-	-
XORL	A,ear	2	6	2	0	long (A) <-- (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XORL	A,eam	2+	7+(a)	0	(d)	long (A) <-- (A) xor (eam)	-	-	-	-	-	*	*	R	-	-

Note:

See Table B.5-1 "Execution Cycle Counts in Each Addressing Mode" and Table B.5-2 "Cycle Count Correction Values for Counting Execution Cycles" for information on (a) to (d) in the table.

Table B.8-10 Six Sign Inversion Instructions (byte, word)

Mnemonic		#	～	RG	B	Operation	L H	A H	I	S	T	N	Z	V	C	R M W
NEG	A	1	2	0	0	byte (A) <-- 0 - (A)	X	-	-	-	-	*	*	*	*	-
NEG	ear	2	3	2	0	byte (ear) <-- 0 - (ear)	-	-	-	-	-	*	*	*	*	-
NEG	eam	2+	5+(a)	0	2×(b)	byte (eam) <-- 0 - (eam)	-	-	-	-	-	*	*	*	*	*
NEGW	A	1	2	0	0	word (A) <-- 0 - (A)	-	-	-	-	-	*	*	*	*	-
NEGW	ear	2	3	2	0	word (ear) <-- 0 - (ear)	-	-	-	-	-	*	*	*	*	-
NEGW	eam	2+	5+(a)	0	2×(c)	word (eam) <-- 0 - (eam)	-	-	-	-	-	*	*	*	*	*

Note:

See Table B.5-1 "Execution Cycle Counts in Each Addressing Mode" and Table B.5-2 "Cycle Count Correction Values for Counting Execution Cycles" for information on (a) to (d) in the table.

Table B.8-11 One Normalization Instruction (long word)

Mnemonic		#	～	RG	B	Operation	L H	A H	I	S	T	N	Z	V	C	R M W
NRML	A,R0	2	*1	1	0	long (A) <-- Shifts to the position where '1' is set for the first time. byte (RD) <-- Shift count at that time	-	-	-	-	-	-	*	-	-	-

*1: 4 when all accumulators have a value of 0; otherwise, 6+(R0)

Table B.8-12 18 Shift Instructions (byte, word, long word)

Mnemonic		#	～	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
RORC	A	2	2	0	0	byte (A) <-- With right rotation carry	-	-	-	-	-	*	*	-	*	-
ROLC	A	2	2	0	0	byte (A) <-- With left rotation carry	-	-	-	-	-	*	*	-	*	-
RORC	ear	2	3	2	0	byte (ear) <-- With right rotation carry	-	-	-	-	-	*	*	-	*	-
RORC	eam	2+	5+(a)	0	2×(b)	byte (eam) <-- With right rotation carry	-	-	-	-	-	*	*	-	*	*
ROLC	ear	2	3	2	0	byte (ear) <-- With left rotation carry	-	-	-	-	-	*	*	-	*	-
ROLC	eam	2+	5+(a)	0	2×(b)	byte (eam) <-- With left rotation carry	-	-	-	-	-	*	*	-	*	*
ASR	A,R0	2	*1	1	0	byte (A) <-- Arithmetic right shift (A, 1 bit)	-	-	-	-	*	*	*	-	*	-
LSR	A,R0	2	*1	1	0	byte (A) <-- Logical right barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-
LSL	A,R0	2	*1	1	0	byte (A) <-- Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-
ASRW	A	1	2	0	0	word (A) <-- Arithmetic right shift (A, 1 bit)	-	-	-	-	*	*	*	-	*	-
LSRW	A/SHRW A	1	2	0	0	word (A) <-- Logical right shift (A, 1 bit)	-	-	-	-	*	R	*	-	*	-
LSLW	A/SHLW A	1	2	0	0	word (A) <-- Logical left shift (A, 1 bit)	-	-	-	-	-	*	*	-	*	-
ASRW	A,R0	2	*1	1	0	word (A) <-- Arithmetic right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSRW	A,R0	2	*1	1	0	word (A) <-- Logical right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSLW	A,R0	2	*1	1	0	word (A) <-- Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-
ASRL	A,R0	2	*2	1	0	long (A) <-- Arithmetic right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSRL	A,R0	2	*2	1	0	long (A) <-- Logical right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSLL	A,R0	2	*2	1	0	long (A) <-- Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-

*1: 6 when R0 is 0; otherwise, 5 + (R0)
*2: 6 when R0 is 0; otherwise, 6 + (R0)

Note:

See Table B.5-1 "Execution Cycle Counts in Each Addressing Mode" and Table B.5-2 "Cycle Count Correction Values for Counting Execution Cycles" for information on (a) to (d) in the table.

Table B.8-13 31 Branch 1 Instructions

Mnemonic		#	～	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
BZ/BEQ	rel	2	*1	0	0	Branch on (Z) = 1	-	-	-	-	-	-	-	-	-	-
BNZ/BNE	rel	2	*1	0	0	Branch on (Z) = 0	-	-	-	-	-	-	-	-	-	-
BC/BLO	rel	2	*1	0	0	Branch on (C) = 1	-	-	-	-	-	-	-	-	-	-
BNC/BHS	rel	2	*1	0	0	Branch on (C) = 0	-	-	-	-	-	-	-	-	-	-
BN	rel	2	*1	0	0	Branch on (N) = 1	-	-	-	-	-	-	-	-	-	-
BP	rel	2	*1	0	0	Branch on (N) = 0	-	-	-	-	-	-	-	-	-	-
BV	rel	2	*1	0	0	Branch on (V) = 1	-	-	-	-	-	-	-	-	-	-
BNV	rel	2	*1	0	0	Branch on (V) = 0	-	-	-	-	-	-	-	-	-	-
BT	rel	2	*1	0	0	Branch on (T) = 1	-	-	-	-	-	-	-	-	-	-
BNT	rel	2	*1	0	0	Branch on (T) = 0	-	-	-	-	-	-	-	-	-	-
BLT	rel	2	*1	0	0	Branch on (V) nor (N) = 1	-	-	-	-	-	-	-	-	-	-
BGE	rel	2	*1	0	0	Branch on (V) nor (N) = 0	-	-	-	-	-	-	-	-	-	-
BLE	rel	2	*1	0	0	Branch on ((V) xor (N)) or (Z) = 1	-	-	-	-	-	-	-	-	-	-
BGT	rel	2	*1	0	0	Branch on ((V) xor (N)) or (Z) = 0	-	-	-	-	-	-	-	-	-	-
BLS	rel	2	*1	0	0	Branch on (C) or (Z) = 1	-	-	-	-	-	-	-	-	-	-
BHI	rel	2	*1	0	0	Branch on (C) or (Z) = 0	-	-	-	-	-	-	-	-	-	-
BRA	rel	2	*1	0	0	Unconditional branch	-	-	-	-	-	-	-	-	-	-
JMP	@A	1	2	0	0	word (PC) <-- (A)	-	-	-	-	-	-	-	-	-	-
JMP	addr16	3	3	0	0	word (PC) <-- addr16	-	-	-	-	-	-	-	-	-	-
JMP	@ear	2	3	1	0	word (PC) <-- (ear)	-	-	-	-	-	-	-	-	-	-
JMP	@eam	2+	4+(a)	0	(c)	word (PC) <-- (eam)	-	-	-	-	-	-	-	-	-	-
JMPP	@ear *3	2	5	2	0	word (PC) <-- (ear), (PCB) <-- (ear+2)	-	-	-	-	-	-	-	-	-	-
JMPP	@eam *3	2+	6+(a)	0	(d)	word (PC) <-- (eam), (PCB) <-- (eam+2)	-	-	-	-	-	-	-	-	-	-
JMPP	addr24	4	4	0	0	word(PC) <-- ad24 0-15,(PCB) <-- ad24 16-23	-	-	-	-	-	-	-	-	-	-
CALL	@ear *4	2	6	1	(c)	word (PC) <-- (ear)	-	-	-	-	-	-	-	-	-	-
CALL	@eam *4	2+	7+(a)	0	2×(c)	word (PC) <-- (eam)	-	-	-	-	-	-	-	-	-	-
CALL	addr16 *5	3	6	0	(c)	word (PC) <-- addr16	-	-	-	-	-	-	-	-	-	-
CALLV	#vct4 *5	1	7	0	2×(c)	Vector call instruction	-	-	-	-	-	-	-	-	-	-
CALLP	@ear *6	2	10	2	2×(c)	word(PC) <-- (ear)0-15,(PCB) <-- (ear)16-23	-	-	-	-	-	-	-	-	-	-
CALLP	@eam *6	2+	11+(a)	0	*2	word(PC) <-- (eam)0-15,(PCB) <-- (eam)16-23	-	-	-	-	-	-	-	-	-	-
CALLP	addr24 *7	4	10	0	2×(c)	word(PC) <-- addr0-15, (PCB) <-- addr16-23	-	-	-	-	-	-	-	-	-	-

*1: 4 when a branch is made; otherwise, 3
*2: 3 x (c) + (b)
*3: Read (word) of branch destination address
*4: W: Save to stack (word) R: Read (word) of branch destination address
*5: Save to stack (word)
*6: W: Save to stack (long word), R: Read (long word) of branch destination address
*7: Save to stack (long word)

Note:

See Table B.5-1 "Execution Cycle Counts in Each Addressing Mode" and Table B.5-2 "Cycle Count Correction Values for Counting Execution Cycles" for information on (a) to (d) in the table.

Table B.8-14 19 Branch 2 Instructions

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
CBNE	A,#imm8,rel	3	*1	0	0	Branch on byte (A) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CWBNE	A,#imm16,rel	4	*1	0	0	Branch on word (A) not equal to imm16	-	-	-	-	-	*	*	*	*	-
CBNE	ear,#imm8,rel	4	*2	1	0	Branch on byte (ear) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CBNE	eam,#imm8,rel *9	4+	*3	0	(b)	Branch on byte (eam) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CWBNE	ear,#imm16,rel	5	*4	1	0	Branch on word (ear) not equal to imm16	-	-	-	-	-	*	*	*	*	-
CWBNE	eam,#imm16,rel*9	5+	*3	0	(c)	Branch on word (eam) not equal to imm16	-	-	-	-	-	*	*	*	*	-
DBNZ	ear,rel	3	*5	2	0	Branch on byte (ear) = (ear) - 1, (ear)not equal to 0	-	-	-	-	-	*	*	*	-	-
DBNZ	eam,rel	3+	*6	2	2×(b)	Branch on byte (eam) = (eam) - 1, (eam) not equal to 0	-	-	-	-	-	*	*	*	-	*
DWBNZ	ear,rel	3	*5	2	0	Branch on word (ear) = (ear) - 1, (ear) not equal to 0	-	-	-	-	-	*	*	*	-	-
DWBNZ	eam,rel	3+	*6	2	2×(c)	Branch on word (eam) = (eam) - 1, (eam) not equal to 0	-	-	-	-	-	*	*	*	-	*
INT	#vct8	2	20	0	8×(c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT	addr16	3	16	0	6×(c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INTP	addr24	4	17	0	6×(c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT9		1	20	0	8×(c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
RETI		1	*8	0	*7	Return from interrupt	-	-	*	*	*	*	*	*	*	-
LINK	#imm8	2	6	0	(c)	Saves the old frame pointer in the stack upon entering the function, then sets the new frame pointer and reserves the local pointer area.	-	-	-	-	-	-	-	-	-	-
UNLINK		1	5	0	(c)	Recovers the old frame pointer from the stack upon exiting the function.	-	-	-	-	-	-	-	-	-	-
RET	*10	1	4	0	(c)	Return from subroutine	-	-	-	-	-	-	-	-	-	-
RETP	*11	1	6	0	(d)	Return from subroutine	-	-	-	-	-	-	-	-	-	-

*1: 5 when a branch is made; otherwise, 4
*2: 13 when a branch is made; otherwise, 12
*3: 7+(a) when a branch is made; otherwise, 6+(a)
*4: 8 when a branch is made; otherwise, 7
*5: 7 when a branch is made; otherwise, 6
*6: 8+(a) when a branch is made; otherwise, 7+(a)
*7: 3 x (b) + 2 x (c) when jumping to the next interruption request; 6 x (c) when returning from the current interruption
*8: 15 when jumping to the next interruption request; 17 when returning from the current interruption
*9: Do not use RWj+ addressing mode with a CBNE or CWBNE instruction.
*10: Return from stack (word)
*11: Return from stack (long word)

Note:

See Table B.5-1 "Execution Cycle Counts in Each Addressing Mode" and Table B.5-2 "Cycle Count Correction Values for Counting Execution Cycles" for information on (a) to (d) in the

table.

Table B.8-15 31 28 Other Control Instructions (byte, word, long word)

Mnemonic		#	～	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
PUSHW	A	1	4	0	(c)	word (SP) <-- (SP) - 2 , ((SP)) <-- (A)	-	-	-	-	-	-	-	-	-	-
PUSHW	AH	1	4	0	(c)	word (SP) <-- (SP) - 2 , ((SP)) <-- (AH)	-	-	-	-	-	-	-	-	-	-
PUSHW	PS	1	4	0	(c)	word (SP) <-- (SP) - 2 , ((SP)) <-- (PS)	-	-	-	-	-	-	-	-	-	-
PUSHW	rlst	2	*3	+&	*4	(SP) <-- (SP) - 2n , ((SP)) <-- (rlst)	-	-	-	-	-	-	-	-	-	-
POPW	A	1	3	0	(c)	word (A) <-- ((SP)) , (SP) <-- (SP) + 2	-	*	-	-	-	-	-	-	-	-
POPW	AH	1	3	0	(c)	word (AH) <-- ((SP)) , (SP) <-- (SP) + 2	-	-	-	-	-	-	-	-	-	-
POPW	PS	1	4	0	(c)	word (PS) <-- ((SP)) , (SP) <-- (SP) + 2	-	-	*	*	*	*	*	*	*	-
POPW	rlst	2	*2	+&	*4	(rlst) <-- ((SP)) , (SP) <-- (SP)	-	-	-	-	-	-	-	-	-	-
JCTX	@A	1	14	0	6×(c)	Context switch instruction	-	-	*	*	*	*	*	*	*	-
AND	CCR,#imm8	2	3	0	0	byte (CCR) <-- (CCR) and imm8	-	-	*	*	*	*	*	*	*	-
OR	CCR,#imm8	2	3	0	0	byte(CCR) <-- (CCR) or imm8	-	-	*	*	*	*	*	*	*	-
MOV	RP,#imm8	2	2	0	0	byte (RP) <-- imm8	-	-	-	-	-	-	-	-	-	-
MOV	ILM,#imm8	2	2	0	0	byte (ILM) <-- imm8	-	-	-	-	-	-	-	-	-	-
MOVEA	RWi,ear	2	3	1	0	word (RWi) <-- ear	-	-	-	-	-	-	-	-	-	-
MOVEA	RWi,eam	2+	2+(a)	1	0	word (RWi) <-- eam	-	-	-	-	-	-	-	-	-	-
MOVEA	A,ear	2	1	0	0	word (A) <-- ear	-	*	-	-	-	-	-	-	-	-
MOVEA	A,eam	2+	1+(a)	0	0	word (A) <-- eam	-	*	-	-	-	-	-	-	-	-
ADDSP	#imm8	2	3	0	0	word (SP) <-- ext(imm8)	-	-	-	-	-	-	-	-	-	-
ADDSP	#imm16	3	3	0	0	word (SP) <-- imm16	-	-	-	-	-	-	-	-	-	-
MOV	A,brg1	2	*1	0	0	byte (A) <-- (brg1)	Z	*	-	-	-	*	*	-	-	-
MOV	brg2,A	2	1	0	0	byte (brg2) <-- (A)	-	-	-	-	-	*	*	-	-	-
NOP		1	1	0	0	No operation	-	-	-	-	-	-	-	-	-	-
ADB		1	1	0	0	Prefix code for AD space access	-	-	-	-	-	-	-	-	-	-
DTB		1	1	0	0	Prefix code for DT space access	-	-	-	-	-	-	-	-	-	-
PCB		1	1	0	0	Prefix code for PC space access	-	-	-	-	-	-	-	-	-	-
SPB		1	1	0	0	Prefix code for SP space access	-	-	-	-	-	-	-	-	-	-
NCC		1	1	0	0	Prefix code for flag no-change	-	-	-	-	-	-	-	-	-	-
CMR		1	1	0	0	Prefix code for common register bank	-	-	-	-	-	-	-	-	-	-

*1: PCB, ADB, SSB, USB, SPB: 1, DTB, DPR: 2
*2: 7 + 3×(POP count) + 2×(POP last register number), 7 when RLST = 0 (no transfer register)
*3: 29 + 3×(PUSH count) - 3×(PUSH last register number), 8 when RLST = 0 (no transfer register)
*4: (POP count)×(c) or (PUSH count)×(c)

Note:

See Table B.5-1 "Execution Cycle Counts in Each Addressing Mode" and Table B.5-2 "Cycle Count Correction Values for Counting Execution Cycles" for information on (a) to (d) in the table.

Table B.8-16 21 Bit Operand Instructions

Mnemonic		#	～	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVB	A,dir:bp	3	5	0	(b)	byte (A) <-- (dir:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB	A,addr16:bp	4	5	0	(b)	byte (A) <-- (addr16:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB	A,io:bp	3	4	0	(b)	byte (A) <-- (io:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB	dir:bp,A	3	7	0	2×(b)	bit (dir:bp)b <-- (A)	-	-	-	-	-	*	*	-	-	*
MOVB	addr16:bp,A	4	7	0	2×(b)	bit (addr16:bp)b <-- (A)	-	-	-	-	-	*	*	-	-	*
MOVB	io:bp,A	3	6	0	2×(b)	bit (io:bp)b <-- (A)	-	-	-	-	-	*	*	-	-	*
SETB	dir:bp	3	7	0	2×(b)	bit (dir:bp)b <-- 1	-	-	-	-	-	-	-	-	-	*
SETB	addr16:bp	4	7	0	2×(b)	bit (addr16:bp)b <-- 1	-	-	-	-	-	-	-	-	-	*
SETB	io:bp	3	7	0	2×(b)	bit (io:bp)b <-- 1	-	-	-	-	-	-	-	-	-	*
CLRB	dir:bp	3	7	0	2×(b)	bit (dir:bp)b <-- 0	-	-	-	-	-	-	-	-	-	*
CLRB	addr16:bp	4	7	0	2×(b)	bit (addr16:bp)b <-- 0	-	-	-	-	-	-	-	-	-	*
CLRB	io:bp	3	7	0	2×(b)	bit (io:bp)b <-- 0	-	-	-	-	-	-	-	-	-	*
BBC	dir:bp,rel	4	*1	0	(b)	Branch on (dir:bp) b = 0	-	-	-	-	-	-	*	-	-	-
BBC	addr16:bp,rel	5	*1	0	(b)	Branch on (addr16:bp) b = 0	-	-	-	-	-	-	*	-	-	-
BBC	io:bp,rel	4	*2	0	(b)	Branch on (io:bp) b = 0	-	-	-	-	-	-	*	-	-	-
BBS	dir:bp,rel	4	*1	0	(b)	Branch on (dir:bp) b = 1	-	-	-	-	-	-	*	-	-	-
BBS	addr16:bp,rel	5	*1	0	(b)	Branch on (addr16:bp) b = 1	-	-	-	-	-	-	*	-	-	-
BBS	io:bp,rel	4	*1	0	(b)	Branch on (io:bp) b = 1	-	-	-	-	-	-	*	-	-	-
SBBS	addr16:bp,rel	5	*3	0	2×(b)	Branch on (addr16:bp) b = 1, bit = 1	-	-	-	-	-	-	*	-	-	*
WBTS	io:bp	3	*4	0	*5	Waits until (io:bp) b = 1	-	-	-	-	-	-	-	-	-	-
WBTC	io:bp	3	*4	0	*5	Waits until (io:bp) b = 0	-	-	-	-	-	-	-	-	-	-

*1: 8 when a branch is made; otherwise, 7
*2: 7 when a branch is made; otherwise, 6
*3: 10 when the condition is met; otherwise 9
*4: Undefined count
*5: Until the condition is met(dir:bp)b

Note:

See Table B.5-1 "Execution Cycle Counts in Each Addressing Mode" and Table B.5-2 "Cycle Count Correction Values for Counting Execution Cycles" for information on (a) to (d) in the table.

Table B.8-17 Six Accumulator Operation Instructions (byte, word)

Mnemonic	#	～	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
SWAP	1	3	0	0	byte (A)0-7 <--> (A)8-15	-	-	-	-	-	-	-	-	-	-
SWAPW/XCHW A,T	1	2	0	0	word (AH) <--> (AL)	-	*	-	-	-	-	-	-	-	-
EXT	1	1	0	0	Byte sign extension	X	-	-	-	-	*	*	-	-	-
EXTW	1	2	0	0	Word sign extension	-	X	-	-	-	*	*	-	-	-
ZEXT	1	1	0	0	Byte zero extension	Z	-	-	-	-	R	*	-	-	-
ZEXTW	1	1	0	0	Word zero extensionbyte	-	z	-	-	-	R	*	-	-	-

Table B.8-18 Ten String Instructions

Mnemonic	#	～	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVS / MOVSI	2	*2	+&	*3	byte transfer @AH+ <-- @AL+, counter = RW0	-	-	-	-	-	-	-	-	-	-
MOVSD	2	*2	+&	*3	byte transfer @AH- <-- @AL-, counter = RW0	-	-	-	-	-	-	-	-	-	-
SCEQ / SCEQI	2	*1	+&	*4	byte search @AH+ <-- AL, counter RW0	-	-	-	-	-	*	*	*	*	-
SCEQD	2	*1	+&	*4	byte search @AH- <-- AL, counter RW0	-	-	-	-	-	*	*	*	*	-
FILS / FILSI	2	6m+6	+&	*3	byte fill @AH+ <-- AL, counter RW0	-	-	-	-	-	*	*	-	-	-
MOVSW / MOVSWI	2	*2	+)	*6	word transfer @AH+ <-- @AL+, counter = RW0	-	-	-	-	-	-	-	-	-	-
MOVSWD	2	*2	+)	*6	word transfer @AH- <-- @AL-, counter = RW0	-	-	-	-	-	-	-	-	-	-
SCWEQ / SCWEQI	2	*1	+)	*7	word search @AH+ - AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
SCWEQD	2	*1	+)	*7	word search @AH- - AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
FILSW / FILSWI	2	6m+6	+)	*6	word fill @AH+ <-- AL, counter = RW0	-	-	-	-	-	*	*	-	-	-

*1: 5 when RW0 is 0, 4 + 7 × (RW0) when the counter expires, or 7n + 5 when a match occurs
*2: 5 when RW0 is 0; otherwise, 4 + 8 × (RW0)
*3: (b) × (RW0) + (b) × (RW0) When the source and destination access different areas, calculate the (b) item individually.
*4: (b) × n
*5: 2 × (R × W0)
*6: (c) × (RW0) + (c) × (RW0) When the source and destination access different areas, calculate the (c) item individually.
*7: (c) × n
*8: 2 × 0(RW0)

Note:

m: RW0 value (counter value), n: Loop count

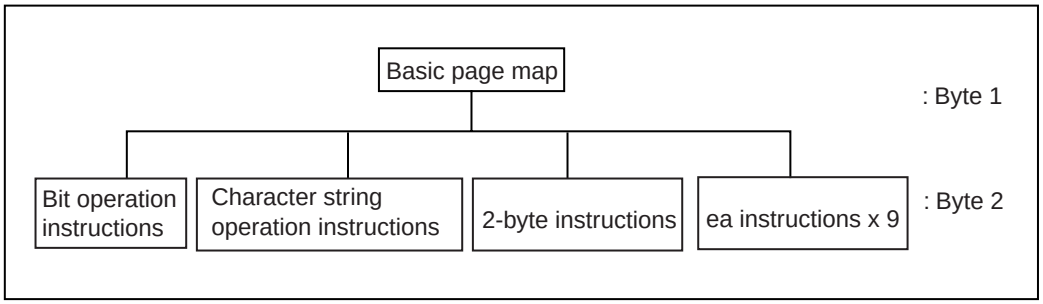
See Table B.5-1 "Execution Cycle Counts in Each Addressing Mode" and Table B.5-2 "Cycle Count Correction Values for Counting Execution Cycles" for information on (a) to (d) in the table.

B.9 Instruction Map

Each F^{2}MC-16LX instruction code consists of 1 or 2 bytes. Therefore, the instruction map consists of multiple pages. Table B.9-2 "Basic Page Map" to Table B.9-21 "XCHW RWi, ea Instruction (first byte = 7F$_H$)" summarize the F^{2}MC-16LX instruction map.

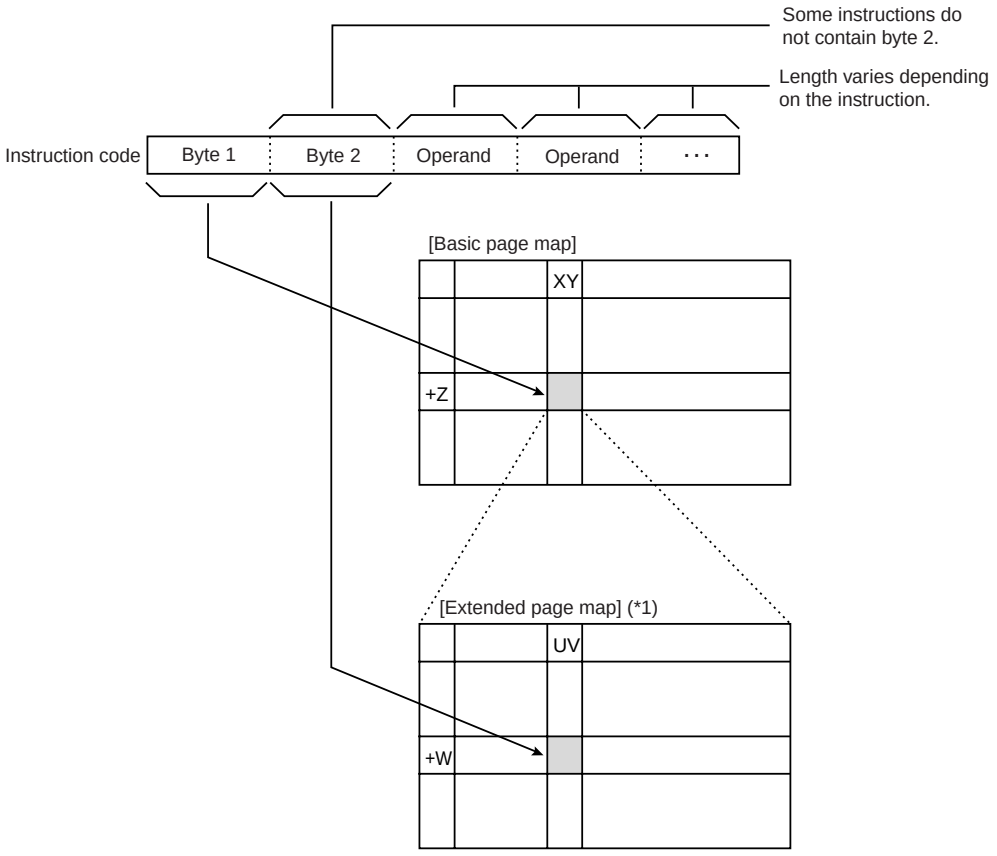
■ Structure of Instruction Map

Figure B.9-1 Structure of Instruction Map



An instruction such as the NOP instruction that ends in one byte is completed within the basic page. An instruction such as the MOVS instruction that requires two bytes recognizes the existence of byte 2 when it references byte 1, and can check the following one byte by referencing the map for byte 2. Figure B.9-2 "Correspondence between Actual Instruction Code and Instruction Map" shows the correspondence between an actual instruction code and instruction map.

Figure B.9-2 Correspondence between Actual Instruction Code and Instruction Map



*1 The extended page map is a generic name of maps for bit operation instructions, character string operation instructions, 2-byte instructions, and ea instructions. Actually, there are multiple extended page maps for each type of instructions.

An example of an instruction code is shown in Table B.9-1 "Example of an Instruction Code".

Table B.9-1 Example of an Instruction Code

Instruction	Byte 1 (from basic page map)	Byte 2 (from extended page map)
NOP	00 +0=00	-
AND A, #8	30 +4=34	-
MOV A, ADB	60 +F=6F	00 +0=00
@RW2+d8, #8rel	70 +0=70	F0 +2=F2

Table B.9-2 Basic Page Map

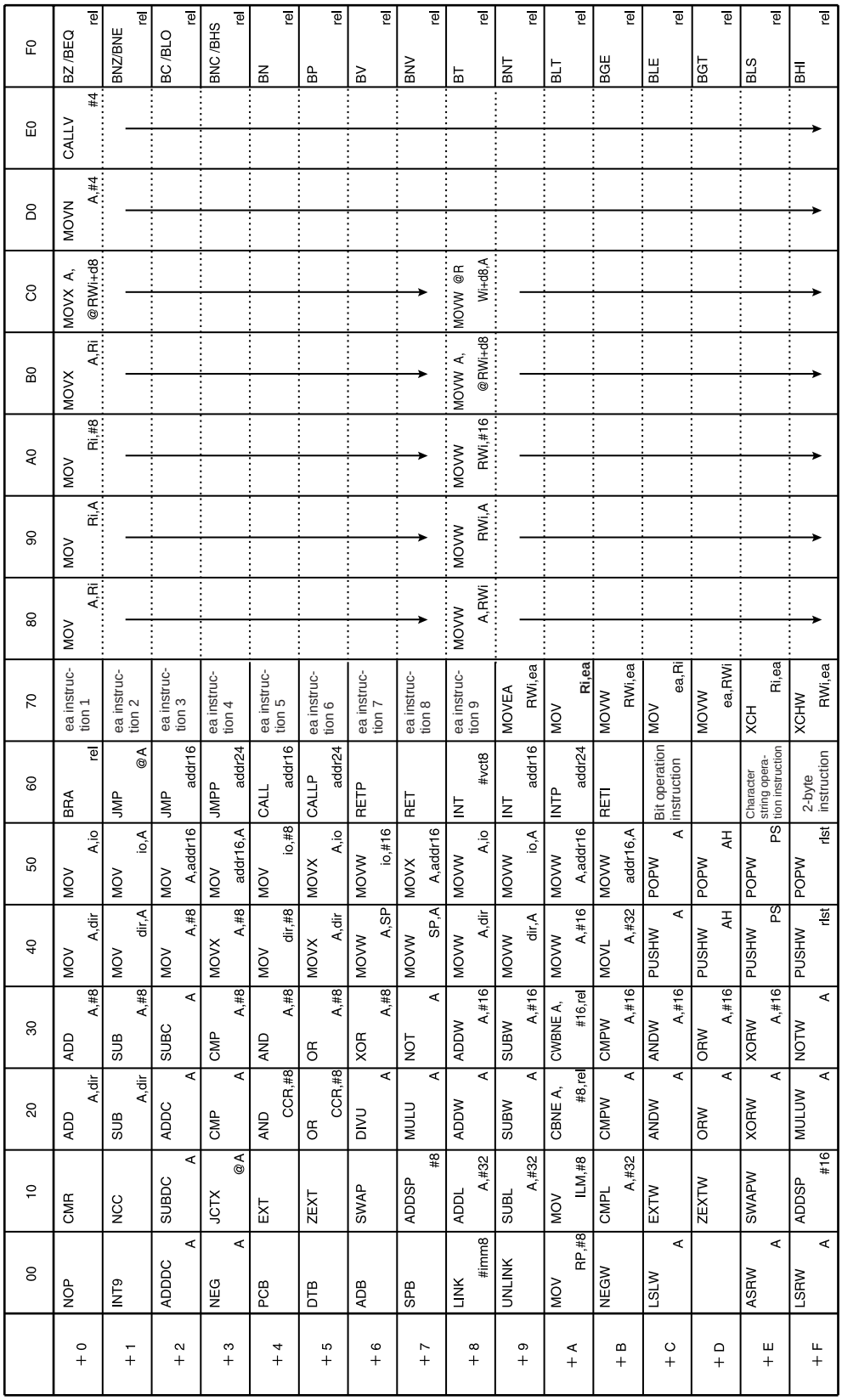


Table B.9-3 Bit Operation Instruction Map (first byte = $6C_H$)

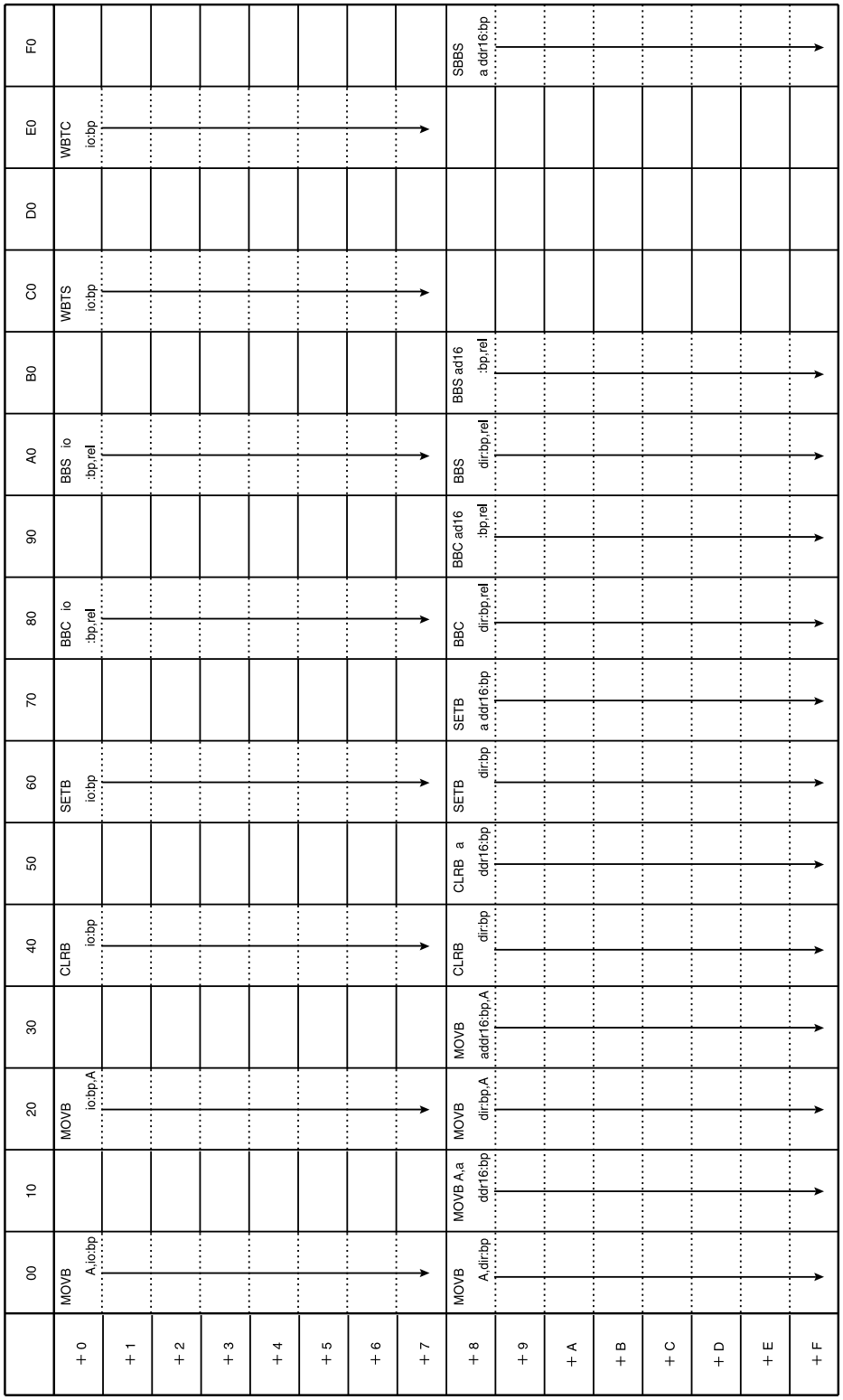


Table B.9-4 Character String Operation Instruction Map (first byte = $6E_H$)

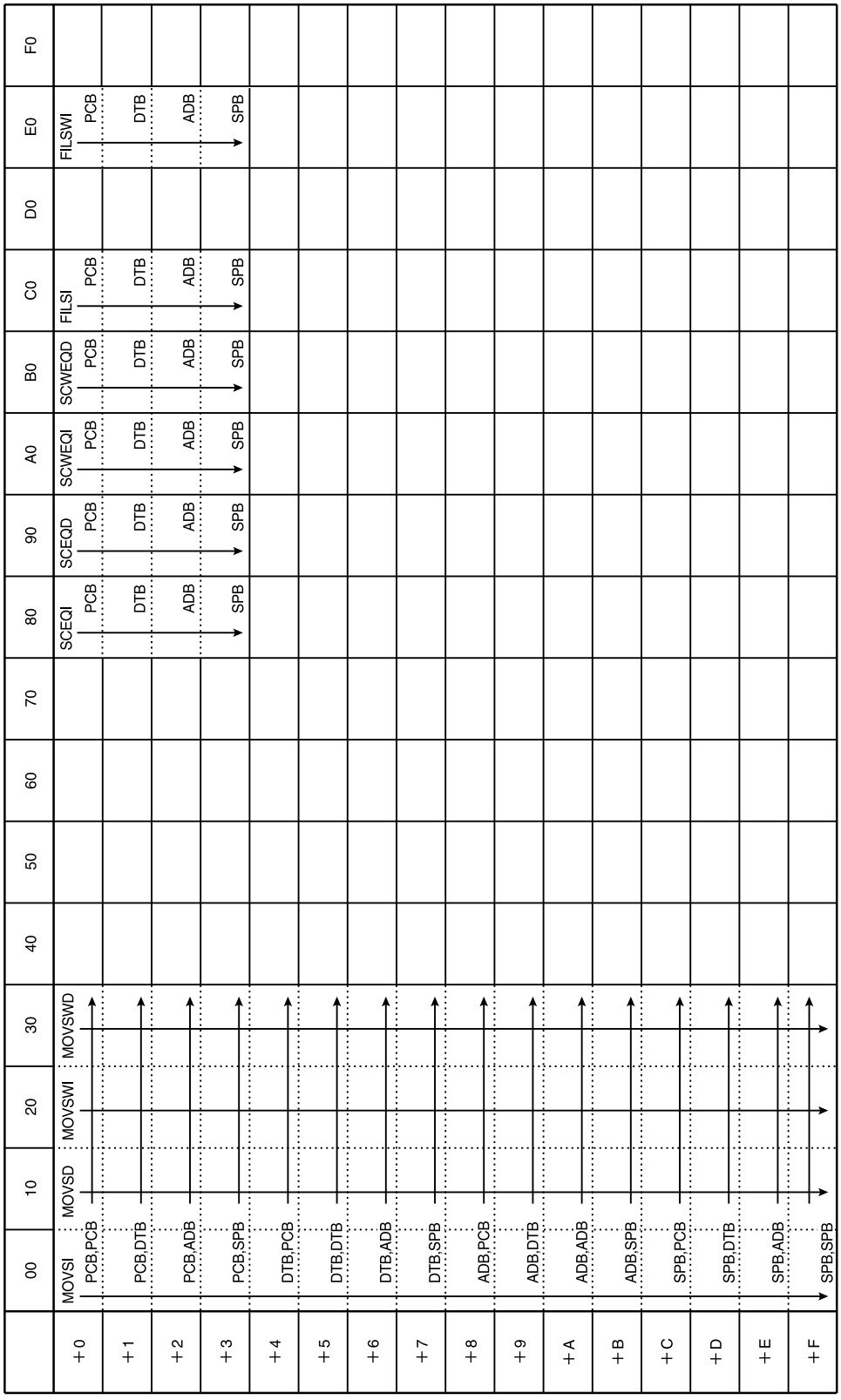


Table B.9-5 2-byte Instruction Map (first byte = $6F_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOV A,DTB	MOV DTB,A	MOVX A, @RL0+d8	MOV @RL0 +d8,A	MOV A, @RL0+d8											
+1	MOV A,ADB	MOV ADB,A														
+2	MOV A,SSB	MOV SSB,A	MOVX A, @RL1+d8	MOV @RL1 +d8,A	MOV A, @RL1+d8											
+3	MOV A,USB	MOV USB,A														
+4	MOV A,DPR	MOV DPR,A	MOVX A, @RL2+d8	MOV @RL2 +d8,A	MOV A, @RL2+d8											
+5	MOV A,@A	MOV @AL,AH														
+6	MOV A,PCB	MOVX A,@A	MOVX A, @RL3+d8	MOV @RL3 +d8,A	MOV A, @RL3+d8											
+7	ROLC A	RORC A														
+8				MOVW @RL 0+d8,A	MOVW A, @RL0+d8		MUL A									
+9							MULW A									
+A				MOVW @RL 1+d8,A	MOVW A, @RL1+d8		DIVU A									
+B																
+C	LSLW A,R0	LSLL A,R0	LSL A,R0	MOVW @RL 2+d8,A	MOVW A, @RL2+d8											
+D	MOVW A,@A	MOVW @AL,AH	NRML A,R0													
+E	ASRW A,R0	ASRL A,R0	ASR A,R0	MOVW @RL 3+d8,A	MOVW A, @RL3+d8											
+F	LSRW A,R0	LSRL A,R0	LSR A,R0													

Table B.9-6 ea Instruction 1 (first byte = 70_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
					CWBNE _|,	CWBNE _|,								CBNE _|,	CBNE _|,
+ 0	ADDL A, A,RL0	ADDL A, @RW0+d8	SUBL A, A,RL0	SUBL A, @RW0+d8	RW0, #16,rel	@RW0+d8, #16,rel	CMPL A,RL0	CMPL A, @RW0+d8	ANDL A,RL0	ANDL A, @RW0+d8	ORL A,RL0	ORL A, @RW0+d8	XORL A,RL0	XORL A, @RW0+d8	R0, #8,rel	@RW0+d8, #8,rel
+ 1	ADDL A, A,RL0	ADDL A, @RW1+d8	SUBL A, A,RL0	SUBL A, @RW1+d8	RW1, #16,rel	@RW1+d8, #16,rel	CMPL A,RL0	CMPL A, @RW1+d8	ANDL A,RL0	ANDL A, @RW1+d8	ORL A,RL0	ORL A, @RW1+d8	XORL A,RL0	XORL A, @RW1+d8	R1, #8,rel	@RW1+d8, #8,rel
+ 2	ADDL A, A,RL1	ADDL A, @RW2+d8	SUBL A, A,RL1	SUBL A, @RW2+d8	RW2, #16,rel	@RW2+d8, #16,rel	CMPL A,RL1	CMPL A, @RW2+d8	ANDL A,RL1	ANDL A, @RW2+d8	ORL A,RL1	ORL A, @RW2+d8	XORL A,RL1	XORL A, @RW2+d8	R2, #8,rel	@RW2+d8, #8,rel
+ 3	ADDL A, A,RL1	ADDL A, @RW3+d8	SUBL A, A,RL1	SUBL A, @RW3+d8	RW3, #16,rel	@RW3+d8, #16,rel	CMPL A,RL1	CMPL A, @RW3+d8	ANDL A,RL1	ANDL A, @RW3+d8	ORL A,RL1	ORL A, @RW3+d8	XORL A,RL1	XORL A, @RW3+d8	R3, #8,rel	@RW3+d8, #8,rel
+ 4	ADDL A, A,RL2	ADDL A, @RW4+d8	SUBL A, A,RL2	SUBL A, @RW4+d8	RW4, #16,rel	@RW4+d8, #16,rel	CMPL A,RL2	CMPL A, @RW4+d8	ANDL A,RL2	ANDL A, @RW4+d8	ORL A,RL2	ORL A, @RW4+d8	XORL A,RL2	XORL A, @RW4+d8	R4, #8,rel	@RW4+d8, #8,rel
+ 5	ADDL A, A,RL2	ADDL A, @RW5+d8	SUBL A, A,RL2	SUBL A, @RW5+d8	RW5, #16,rel	@RW5+d8, #16,rel	CMPL A,RL2	CMPL A, @RW5+d8	ANDL A,RL2	ANDL A, @RW5+d8	ORL A,RL2	ORL A, @RW5+d8	XORL A,RL2	XORL A, @RW5+d8	R5, #8,rel	@RW5+d8, #8,rel
+ 6	ADDL A, A,RL3	ADDL A, @RW6+d8	SUBL A, A,RL3	SUBL A, @RW6+d8	RW6, #16,rel	@RW6+d8, #16,rel	CMPL A,RL3	CMPL A, @RW6+d8	ANDL A,RL3	ANDL A, @RW6+d8	ORL A,RL3	ORL A, @RW6+d8	XORL A,RL3	XORL A, @RW6+d8	R6, #8,rel	@RW6+d8, #8,rel
+ 7	ADDL A, A,RL3	ADDL A, @RW7+d8	SUBL A, A,RL3	SUBL A, @RW7+d8	RW7, #16,rel	@RW7+d8, #16,rel	CMPL A,RL3	CMPL A, @RW7+d8	ANDL A,RL3	ANDL A, @RW7+d8	ORL A,RL3	ORL A, @RW7+d8	XORL A,RL3	XORL A, @RW7+d8	R7, #8,rel	@RW7+d8, #8,rel
+ 8	ADDL A, A,@RW0	ADDL A, @RW0+d16	SUBL A, A,@RW0	SUBL A, @RW0+d16	@RW0, #16,rel	@RW0+d16, , #16,rel	CMPL A,@RW0	CMPL A, @RW0+d16	ANDL A,@RW0	ANDL A, @RW0+d16	ORL A,@RW0	ORL A, @RW0+d16	XORL A,@RW0	XORL A, @RW0+d16	@RW0, #8,rel	@RW0+d16 #8,rel
+ 9	ADDL A, A,@RW1	ADDL A, @RW1+d16	SUBL A, A,@RW1	SUBL A, @RW1+d16	@RW1, #16,rel	@RW1+d16 , #16,rel	CMPL A,@RW1	CMPL A, @RW1+d16	ANDL A,@RW1	ANDL A, @RW1+d16	ORL A,@RW1	ORL A, @RW1+d16	XORL A,@RW1	XORL A, @RW1+d16	@RW1, #8,rel	@RW1+d16 #8,rel
+ A	ADDL A, A,@RW2	ADDL A, @RW2+d16	SUBL A, A,@RW2	SUBL A, @RW2+d16	@RW2, #16,rel	@RW2+d16 , #16,rel	CMPL A,@RW2	CMPL A, @RW2+d16	ANDL A,@RW2	ANDL A, @RW2+d16	ORL A,@RW2	ORL A, @RW2+d16	XORL A,@RW2	XORL A, @RW2+d16	@RW2, #8,rel	@RW2+d16 #8,rel
+ B	ADDL A, A,@RW3	ADDL A, @RW3+d16	SUBL A, A,@RW3	SUBL A, @RW3+d16	@RW3, #16,rel	@RW3+d16 , #16,rel	CMPL A,@RW3	CMPL A, @RW3+d16	ANDL A,@RW3	ANDL A, @RW3+d16	ORL A,@RW3	ORL A, @RW3+d16	XORL A,@RW3	XORL A, @RW3+d16	@RW3, #8,rel	@RW3+d16 #8,rel
+ C	ADDL A, A,@RW0+	ADDL A, @RW0+RW7	SUBL A, A,@RW0+	SUBL A, @RW0+RW7	Use prohibited	@RW0+RW7 , #16,rel	CMPL A,@RW0+	CMPL A, @RW0+RW7	ANDL A,@RW0+	ANDL A, @RW0+RW7	ORL A,@RW0+	ORL A, @RW0+RW7	XORL A,@RW0+	XORL A, @RW0+RW7	Use prohibited	@RW0+RW7 , #8,rel
+ D	ADDL A, A,@RW1+	ADDL A, @RW1+RW7	SUBL A, A,@RW1+	SUBL A, @RW1+RW7	Use prohibited	@RW1+RW7 , #16,rel	CMPL A,@RW1+	CMPL A, @RW1+RW7	ANDL A,@RW1+	ANDL A, @RW1+RW7	ORL A,@RW1+	ORL A, @RW1+RW7	XORL A,@RW1+	XORL A, @RW1+RW7	Use prohibited	@RW1+RW7 , #8,rel
+ E	ADDL A, A,@RW2+	ADDL A, @PC+d16	SUBL A, A,@RW2+	SUBL A, @PC+d16	Use prohibited	@PC+d16, #16,rel	CMPL A,@RW2+	CMPL A, @PC+d16	ANDL A,@RW2+	ANDL A, @PC+d16	ORL A,@RW2+	ORL A, @PC+d16	XORL A,@RW2+	XORL A, @PC+d16	Use prohibited	@PC+d16, #8,rel
+ F	ADDL A, A,@RW3+	ADDL A, addr16	SUBL A, A,@RW3+	SUBL A, addr16	Use prohibited	addr16, #16,rel	CMPL A,@RW3+	CMPL A, addr16	ANDL A,@RW3+	ANDL A, addr16	ORL A,@RW3+	ORL A, addr16	XORL A,@RW3+	XORL A, addr16	Use prohibited	addr16, #8,rel

Table B.9-7 ea Instruction 2 (first byte = 71_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	JMPP	JMPP	CALLP	CALLP	INCL	INCL	DECL	DECL	MOVL	MOVL A,	MOVL	MOVL @R	MOV	MOV @R	MOVEA	MOVEA A,
	@RL0	@@RW0+d8	@RL0	@@RW0+d8	RL0	@RW0+d8	RL0	@RW0+d8	A,RL0	@RW0+d8	RL0,A	W0+d8,A	R0,#8	W0+d8,#8	A,RW0	@RW0+d8
+ 1	JMPP	JMPP	CALLP	CALLP	INCL	INCL	DECL	DECL	MOVL	MOVL A,	MOVL	MOVL @R	MOV	MOV @R	MOVEA	MOVEA A,
	@RL0	@@RW1+d8	@RL0	@@RW1+d8	RL0	@RW1+d8	RL0	@RW1+d8	A,RL0	@RW1+d8	RL0,A	W1+d8,A	R1,#8	W1+d8,#8	A,RW1	@RW1+d8
+ 2	JMPP	JMPP	CALLP	CALLP	INCL	INCL	DECL	DECL	MOVL	MOVL A,	MOVL	MOVL @R	MOV	MOV @R	MOVEA	MOVEA A,
	@RL1	@@RW2+d8	@RL1	@@RW2+d8	RL1	@RW2+d8	RL1	@RW2+d8	A,RL1	@RW2+d8	RL1,A	W2+d8,A	R2,#8	W2+d8,#8	A,RW2	@RW2+d8
+ 3	JMPP	JMPP	CALLP	CALLP	INCL	INCL	DECL	DECL	MOVL	MOVL A,	MOVL	MOVL @R	MOV	MOV @R	MOVEA	MOVEA A,
	@RL1	@@RW3+d8	@RL1	@@RW3+d8	RL1	@RW3+d8	RL1	@RW3+d8	A,RL1	@RW3+d8	RL1,A	W3+d8,A	R3,#8	W3+d8,#8	A,RW3	@RW3+d8
+ 4	JMPP	JMPP	CALLP	CALLP	INCL	INCL	DECL	DECL	MOVL	MOVL A,	MOVL	MOVL @R	MOV	MOV @R	MOVEA	MOVEA A,
	@RL2	@@RW4+d8	@RL2	@@RW4+d8	RL2	@RW4+d8	RL2	@RW4+d8	A,RL2	@RW4+d8	RL2,A	W4+d8,A	R4,#8	W4+d8,#8	A,RW4	@RW4+d8
+ 5	JMPP	JMPP	CALLP	CALLP	INCL	INCL	DECL	DECL	MOVL	MOVL A,	MOVL	MOVL @R	MOV	MOV @R	MOVEA	MOVEA A,
	@RL2	@@RW5+d8	@RL2	@@RW5+d8	RL2	@RW5+d8	RL2	@RW5+d8	A,RL2	@RW5+d8	RL2,A	W5+d8,A	R5,#8	W5+d8,#8	A,RW5	@RW5+d8
+ 6	JMPP	JMPP	CALLP	CALLP	INCL	INCL	DECL	DECL	MOVL	MOVL A,	MOVL	MOVL @R	MOV	MOV @R	MOVEA	MOVEA A,
	@RL3	@@RW6+d8	@RL3	@@RW6+d8	RL3	@RW6+d8	RL3	@RW6+d8	A,RL3	@RW6+d8	RL3,A	W6+d8,A	R6,#8	W6+d8,#8	A,RW6	@RW6+d8
+ 7	JMPP	JMPP	CALLP	CALLP	INCL	INCL	DECL	DECL	MOVL	MOVL A,	MOVL	MOVL @R	MOV	MOV @R	MOVEA	MOVEA A,
	@RL3	@@RW7+d8	@RL3	@@RW7+d8	RL3	@RW7+d8	RL3	@RW7+d8	A,RL3	@RW7+d8	RL3,A	W7+d8,A	R7,#8	W7+d8,#8	A,RW7	@RW7+d8
+ 8	JMPP @	JMPP @	CALLP @	CALLP @	INCL @	INCL @	DECL @	DECL @	MOVL A,	MOVL A,	MOVL @RW0,A	MOVL @R	MOV @RW0,#8	MOV @RW 0+d16,#8	MOVEA A,@RW0	MOVEA A, @RW0+d16
	@@RW0	@RW0+d16	@@RW0	@RW0+d16	@RW0	@RW0+d16	@RW0	@RW0+d16	A,@RW0	@RW0+d16		W0+d16,A				
+ 9	JMPP @	JMPP @	CALLP @	CALLP @	INCL @	INCL @	DECL @	DECL @	MOVL A,	MOVL A,	MOVL @RW1,A	MOVL @R	MOV @RW1,#8	MOV @RW 1+d16,#8	MOVEA A,@RW1	MOVEA A, @RW1+d16
	@@RW1	@RW1+d16	@@RW1	@RW1+d16	@RW1	@RW1+d16	@RW1	@RW1+d16	A,@RW1	@RW1+d16		W1+d16,A				
+ A	JMPP @	JMPP @	CALLP @	CALLP @	INCL @	INCL @	DECL @	DECL @	MOVL A,	MOVL A,	MOVL @RW2,A	MOVL @R	MOV @RW2,#8	MOV @RW 2+d16,#8	MOVEA A,@RW2	MOVEA A, @RW2+d16
	@@RW2	@RW2+d16	@@RW2	@RW2+d16	@RW2	@RW2+d16	@RW2	@RW2+d16	A,@RW2	@RW2+d16		W2+d16,A				
+ B	JMPP @	JMPP @	CALLP @	CALLP @	INCL @	INCL @	DECL @	DECL @	MOVL A,	MOVL A,	MOVL @RW3,A	MOVL @R	MOV @RW3,#8	MOV @RW 3+d16,#8	MOVEA A,@RW3	MOVEA A, @RW3+d16
	@@RW3	@RW3+d16	@@RW3	@RW3+d16	@RW3	@RW3+d16	@RW3	@RW3+d16	A,@RW3	@RW3+d16		W3+d16,A				
+ C	JMPP @	JMPP @	CALLP @	CALLP @	INCL @	INCL @	DECL @	DECL @	MOVL A,	MOVL A,	MOVL @RW0+A	MOVL @R	MOV @RW0+,#8	MOV @RW 0+RW7,#8	MOVEA A,@RW0+	MOVEA A, @RW0+RW7
	@@RW0+	@RW0+RW7	@@RW0+	@RW0+RW7	@RW0+	@RW0+RW7	@RW0+	@RW0+RW7	A,@RW0+	@RW0+RW7	@RW0+,A	W0+RW7,A	@RW0+,#8			
+ D	JMPP @	JMPP @	CALLP @	CALLP @	INCL @	INCL @	DECL @	DECL @	MOVL A,	MOVL A,	MOVL @RW1+,A	MOVL @R	MOV @RW1+,#8	MOV @RW 1+RW7,#8	MOVEA A,@RW1+	MOVEA A, @RW1+RW7
	@@RW1+	@RW1+RW7	@@RW1+	@RW1+RW7	@RW1+	@RW1+RW7	@RW1+	@RW1+RW7	A,@RW1+	@RW1+RW7		W1+RW7,A				
+ E	JMPP	JMPP	CALLP	CALLP	INCL	INCL	DECL	DECL	MOVL	MOVL A,	MOVL	MOVL @P	MOV	MOV @P	MOVEA	MOVEA A,
	@@RW2+	@@PC+d16	@@RW2+	@@PC+d16	@RW2+	@PC+d16	@RW2+	@PC+d16	A,@RW2+	@PC+d16	@RW2+,A	C+d16,A	@RW2+,#8	C+d16,#8	A,@RW2+	@PC+d16
+ F	JMPP	JMPP	CALLP	CALLP	INCL	INCL	DECL	DECL	MOVL	MOVL A,	MOVL	MOVL	MOV	MOV a	MOVEA	MOVEA A,
	@@RW3+	@addr16	@@RW3+	@addr16	@RW3+	addr16	@RW3+	addr16	A,@RW3+	addr16	@RW3+,A	addr16,A	@RW3+,#8	ddr16,#8	A,@RW3+	addr16

Table B.9-8 ea Instruction 3 (first byte = 72_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	ROLC R0	ROLC @RW0+d8	RORC R0	RORC @RW0+d8	INC R0	INC @RW0+d8	DEC R0	DEC @RW0+d8	MOV A,R0	MOV A, @RW0+d8	MOV R0,A	MOV @R W0+d8,A	MOVX A,R0	MOVX A, @RW0+d8	XCH A,R0	XCH A, @RW0+d8
+ 1	ROLC R1	ROLC @RW1+d8	RORC R1	RORC @RW1+d8	INC R1	INC @RW1+d8	DEC R1	DEC @RW1+d8	MOV A,R1	MOV A, @RW1+d8	MOV R1,A	MOV @R W1+d8,A	MOVX A,R1	MOVX A, @RW1+d8	XCH A,R1	XCH A, @RW1+d8
+ 2	ROLC R2	ROLC @RW2+d8	RORC R2	RORC @RW2+d8	INC R2	INC @RW2+d8	DEC R2	DEC @RW2+d8	MOV A,R2	MOV A, @RW2+d8	MOV R2,A	MOV @R W2+d8,A	MOVX A,R2	MOVX A, @RW2+d8	XCH A,R2	XCH A, @RW2+d8
+ 3	ROLC R3	ROLC @RW3+d8	RORC R3	RORC @RW3+d8	INC R3	INC @RW3+d8	DEC R3	DEC @RW3+d8	MOV A,R3	MOV A, @RW3+d8	MOV R3,A	MOV @R W3+d8,A	MOVX A,R3	MOVX A, @RW3+d8	XCH A,R3	XCH A, @RW3+d8
+ 4	ROLC R4	ROLC @RW4+d8	RORC R4	RORC @RW4+d8	INC R4	INC @RW4+d8	DEC R4	DEC @RW4+d8	MOV A,R4	MOV A, @RW4+d8	MOV R4,A	MOV @R W4+d8,A	MOVX A,R4	MOVX A, @RW4+d8	XCH A,R4	XCH A, @RW4+d8
+ 5	ROLC R5	ROLC @RW5+d8	RORC R5	RORC @RW5+d8	INC R5	INC @RW5+d8	DEC R5	DEC @RW5+d8	MOV A,R5	MOV A, @RW5+d8	MOV R5,A	MOV @R W5+d8,A	MOVX A,R5	MOVX A, @RW5+d8	XCH A,R5	XCH A, @RW5+d8
+ 6	ROLC R6	ROLC @RW6+d8	RORC R6	RORC @RW6+d8	INC R6	INC @RW6+d8	DEC R6	DEC @RW6+d8	MOV A,R6	MOV A, @RW6+d8	MOV R6,A	MOV @R W6+d8,A	MOVX A,R6	MOVX A, @RW6+d8	XCH A,R6	XCH A, @RW6+d8
+ 7	ROLC R7	ROLC @RW7+d8	RORC R7	RORC @RW7+d8	INC R7	INC @RW7+d8	DEC R7	DEC @RW7+d8	MOV A,R7	MOV A, @RW7+d8	MOV R7,A	MOV @R W7+d8,A	MOVX A,R7	MOVX A, @RW7+d8	XCH A,R7	XCH A, @RW7+d8
+ 8	ROLC @RW0	ROLC @RW0+d16	RORC @RW0	RORC @RW0+d16	INC @RW0	INC @RW0+d16	DEC @RW0	DEC @RW0+d16	MOV A,@RW0	MOV A, @RW0+d16	MOV @RW0,A	MOV @R W0+d16,A	MOVX A,@RW0	MOVX A, @RW0+d16	XCH A,@RW0	XCH A, @RW0+d16
+ 9	ROLC @RW1	ROLC @RW1+d16	RORC @RW1	RORC @RW1+d16	INC @RW1	INC @RW1+d16	DEC @RW1	DEC @RW1+d16	MOV A,@RW1	MOV A, @RW1+d16	MOV @RW1,A	MOV @R W1+d16,A	MOVX A,@RW1	MOVX A, @RW1+d16	XCH A,@RW1	XCH A, @RW1+d16
+ A	ROLC @RW2	ROLC @RW2+d16	RORC @RW2	RORC @RW2+d16	INC @RW2	INC @RW2+d16	DEC @RW2	DEC @RW2+d16	MOV A,@RW2	MOV A, @RW2+d16	MOV @RW2,A	MOV @R W2+d16,A	MOVX A,@RW2	MOVX A, @RW2+d16	XCH A,@RW2	XCH A, @RW2+d16
+ B	ROLC @RW3	ROLC @RW3+d16	RORC @RW3	RORC @RW3+d16	INC @RW3	INC @RW3+d16	DEC @RW3	DEC @RW3+d16	MOV A,@RW3	MOV A, @RW3+d16	MOV @RW3,A	MOV @R W3+d16,A	MOVX A,@RW3	MOVX A, @RW3+d16	XCH A,@RW3	XCH A, @RW3+d16
+ C	ROLC @RW0+	ROLC @RW0+RW7	RORC @RW0+	RORC @RW0+RW7	INC @RW0+	INC @RW0+RW7	DEC @RW0+	DEC @RW0+RW7	MOV A,@RW0+	MOV A, @RW0+RW7	MOV @RW0+,A	MOV @R W0+RW7,A	MOVX A,@RW0+	MOVX A, @RW0+RW7	XCH A,@RW0+	XCH A, @RW0+RW7
+ D	ROLC @RW1+	ROLC @RW1+RW7	RORC @RW1+	RORC @RW1+RW7	INC @RW1+	INC @RW1+RW7	DEC @RW1+	DEC @RW1+RW7	MOV A,@RW1+	MOV A, @RW1+RW7	MOV @RW1+,A	MOV @R W1+RW7,A	MOVX A,@RW1+	MOVX A, @RW1+RW7	XCH A,@RW1+	XCH A, @RW1+RW7
+ E	ROLC @RW2+	ROLC @PC+d16	RORC @RW2+	RORC @PC+d16	INC @RW2+	INC @PC+d16	DEC @RW2+	DEC @PC+d16	MOV A,@RW2+	MOV A, @PC+d16	MOV @RW2+,A	MOV @P C+d16,A	MOVX A,@RW2+	MOVX A, @PC+d16	XCH A,@RW2+	XCH A, @PC+d16
+ F	ROLC @RW3+	ROLC addr16	RORC @RW3+	RORC addr16	INC @RW3+	INC addr16	DEC @RW3+	DEC addr16	MOV A,@RW3+	MOV A, addr16	MOV @RW3+,A	MOV addr16,A	MOVX A,@RW3+	MOVX A, addr16	XCH A,@RW3+	XCH A, addr16

Table B.9-9 ea Instruction 4 (first byte = 73_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	JMP @RW0	JMP @@RW0+d8	CALL @RW0	CALL @@RW0+d8	INCW RW0	INCW @RW0+d8	DECW RW0	DECW @RW0+d8	MOVW A,RW0	MOVW A,@RW0+d8	MOVW RW0,A	MOVW @R W0+d8,A	MOVW RW0,#16	MOVW @RW 0+d8,#16	XCHW A,RW0	XCHW A,@RW0+d8
+ 1	JMP @RW1	JMP @@RW1+d8	CALL @RW1	CALL @@RW1+d8	INCW RW1	INCW @RW1+d8	DECW RW1	DECW @RW1+d8	MOVW A,RW1	MOVW A,@RW1+d8	MOVW RW1,A	MOVW @R W1+d8,A	MOVW RW1,#16	MOVW @RW 1+d8,#16	XCHW A,RW1	XCHW A,@RW1+d8
+ 2	JMP @RW2	JMP @@RW2+d8	CALL @RW2	CALL @@RW2+d8	INCW RW2	INCW @RW2+d8	DECW RW2	DECW @RW2+d8	MOVW A,RW2	MOVW A,@RW2+d8	MOVW RW2,A	MOVW @R W2+d8,A	MOVW RW2,#16	MOVW @RW 2+d8,#16	XCHW A,RW2	XCHW A,@RW2+d8
+ 3	JMP @RW3	JMP @@RW3+d8	CALL @RW3	CALL @@RW3+d8	INCW RW3	INCW @RW3+d8	DECW RW3	DECW @RW3+d8	MOVW A,RW3	MOVW A,@RW3+d8	MOVW RW3,A	MOVW @R W3+d8,A	MOVW RW3,#16	MOVW @RW 3+d8,#16	XCHW A,RW3	XCHW A,@RW3+d8
+ 4	JMP @RW4	JMP @@RW4+d8	CALL @RW4	CALL @@RW4+d8	INCW RW4	INCW @RW4+d8	DECW RW4	DECW @RW4+d8	MOVW A,RW4	MOVW A,@RW4+d8	MOVW RW4,A	MOVW @R W4+d8,A	MOVW RW4,#16	MOVW @RW 4+d8,#16	XCHW A,RW4	XCHW A,@RW4+d8
+ 5	JMP @RW5	JMP @@RW5+d8	CALL @RW5	CALL @@RW5+d8	INCW RW5	INCW @RW5+d8	DECW RW5	DECW @RW5+d8	MOVW A,RW5	MOVW A,@RW5+d8	MOVW RW5,A	MOVW @R W5+d8,A	MOVW RW5,#16	MOVW @RW 5+d8,#16	XCHW A,RW5	XCHW A,@RW5+d8
+ 6	JMP @RW6	JMP @@RW6+d8	CALL @RW6	CALL @@RW6+d8	INCW RW6	INCW @RW6+d8	DECW RW6	DECW @RW6+d8	MOVW A,RW6	MOVW A,@RW6+d8	MOVW RW6,A	MOVW @R W6+d8,A	MOVW RW6,#16	MOVW @RW 6+d8,#16	XCHW A,RW6	XCHW A,@RW6+d8
+ 7	JMP @RW7	JMP @@RW7+d8	CALL @RW7	CALL @@RW7+d8	INCW RW7	INCW @RW7+d8	DECW RW7	DECW @RW7+d8	MOVW A,RW7	MOVW A,@RW7+d8	MOVW RW7,A	MOVW @R W7+d8,A	MOVW RW7,#16	MOVW @RW 7+d8,#16	XCHW A,RW7	XCHW A,@RW7+d8
+ 8	JMP @@RW0	JMP @ @RW0+d16	CALL @@RW0	CALL @ @RW0+d16	INCW @RW0	INCW @RW0+d16	DECW @RW0	DECW @RW0+d16	MOVW A,@RW0	MOVW A,@RW0+d16	MOVW @RW0,A	MOVW @R W0+d16,A	MOVW @RW0,#16	MOVW@RW0 -rd16,#16	XCHW A,@RW0	XCHW A,@RW0+d16
+ 9	JMP @@RW1	JMP @ @RW1+d16	CALL @@RW1	CALL @ @RW1+d16	INCW @RW1	INCW @RW1+d16	DECW @RW1	DECW @RW1+d16	MOVW A,@RW1	MOVW A,@RW1+d16	MOVW @RW1,A	MOVW @R W1+d16,A	MOVW @RW1,#16	MOVW@RW1 -rd16,#16	XCHW A,@RW1	XCHW A,@RW1+d16
+ A	JMP @@RW2	JMP @ @RW2+d16	CALL @@RW2	CALL @ @RW2+d16	INCW @RW2	INCW @RW2+d16	DECW @RW2	DECW @RW2+d16	MOVW A,@RW2	MOVW A,@RW2+d16	MOVW @RW2,A	MOVW @R W2+d16,A	MOVW @RW2,#16	MOVW@RW2 -rd16,#16	XCHW A,@RW2	XCHW A,@RW2+d16
+ B	JMP @@RW3	JMP @ @RW3+d16	CALL @@RW3	CALL @ @RW3+d16	INCW @RW3	INCW @RW3+d16	DECW @RW3	DECW @RW3+d16	MOVW A,@RW3	MOVW A,@RW3+d16	MOVW @RW3,A	MOVW @R W3+d16,A	MOVW @RW3,#16	MOVW@RW3 -rd16,#16	XCHW A,@RW3	XCHW A,@RW3+d16
+ C	JMP @@RW0+	JMP @ @RW0+RW7	CALL @@RW0+	CALL @ @RW0+RW7	INCW @RW0+	INCW @RW0+RW7	DECW @RW0+	DECW @RW0+RW7	MOVW A,@RW0+	MOVW A,@RW0+RW7	MOVW @RW0+,A	MOVW @R W0+RW7,A	MOVW @ RW0+,#16	MOVW@RW0 +RW7,#16	XCHW A,@RW0+	XCHW A,@RW0+RW7
+ D	JMP @@RW1+	JMP @ @RW1+RW7	CALL @@RW1+	CALL @ @RW1+RW7	INCW @RW1+	INCW @RW1+RW7	DECW @RW1+	DECW @RW1+RW7	MOVW A,@RW1+	MOVW A,@RW1+RW7	MOVW @RW1+,A	MOVW @R W1+RW7,A	MOVW @ RW1+,#16	MOVW@RW1 +RW7,#16	XCHW A,@RW1+	XCHW A,@RW1+RW7
+ E	JMP @@RW2+	JMP @@PC+d16	CALL @@RW2+	CALL @@PC+d16	INCW @RW2+	INCW @PC+d16	DECW @RW2+	DECW @PC+d16	MOVW A,@RW2+	MOVW A,@PC+d16	MOVW @RW2+,A	MOVW @P C+d16,A	MOVW @ RW2+,#16	MOVW @PC +d16,#16	XCHW A,@RW2+	XCHW A,@PC+d16
+ F	JMP @@RW3+	JMP @addr16	CALL @@RW3+	CALL @addr16	INCW @RW3+	INCW addr16	DECW @RW3+	DECW addr16	MOVW A,@RW3+	MOVW A,addr16	MOVW @RW3+,A	MOVW addr16,A	MOVW @ RW3+,#16	MOVW ad dr16,#16	XCHW A,@RW3+	XCHW A,addr16

Table B.9-10 ea Instruction 5 (first byte = 74_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	ADD A, A,R0	ADD A, @RW0+d8	SUB A, A,R0	SUB A, @RW0+d8	ADDC A, A,R0	ADDC A, @RW0+d8	CMP A, A,R0	CMP A, @RW0+d8	AND A, A,R0	AND A, @RW0+d8	OR A, A,R0	OR A, @RW0+d8	XOR A, A,R0	XOR A, @RW0+d8	DBNZ R0,r	DBNZ @ RW0+d8,r
+ 1	ADD A, A,R1	ADD A, @RW1+d8	SUB A, A,R1	SUB A, @RW1+d8	ADDC A, A,R1	ADDC A, @RW1+d8	CMP A, A,R1	CMP A, @RW1+d8	AND A, A,R1	AND A, @RW1+d8	OR A, A,R1	OR A, @RW1+d8	XOR A, A,R1	XOR A, @RW1+d8	DBNZ R1,r	DBNZ @ RW1+d8,r
+ 2	ADD A, A,R2	ADD A, @RW2+d8	SUB A, A,R2	SUB A, @RW2+d8	ADDC A, A,R2	ADDC A, @RW2+d8	CMP A, A,R2	CMP A, @RW2+d8	AND A, A,R2	AND A, @RW2+d8	OR A, A,R2	OR A, @RW2+d8	XOR A, A,R2	XOR A, @RW2+d8	DBNZ R2,r	DBNZ @ RW2+d8,r
+ 3	ADD A, A,R3	ADD A, @RW3+d8	SUB A, A,R3	SUB A, @RW3+d8	ADDC A, A,R3	ADDC A, @RW3+d8	CMP A, A,R3	CMP A, @RW3+d8	AND A, A,R3	AND A, @RW3+d8	OR A, A,R3	OR A, @RW3+d8	XOR A, A,R3	XOR A, @RW3+d8	DBNZ R3,r	DBNZ @ RW3+d8,r
+ 4	ADD A, A,R4	ADD A, @RW4+d8	SUB A, A,R4	SUB A, @RW4+d8	ADDC A, A,R4	ADDC A, @RW4+d8	CMP A, A,R4	CMP A, @RW4+d8	AND A, A,R4	AND A, @RW4+d8	OR A, A,R4	OR A, @RW4+d8	XOR A, A,R4	XOR A, @RW4+d8	DBNZ R4,r	DBNZ @ RW4+d8,r
+ 5	ADD A, A,R5	ADD A, @RW5+d8	SUB A, A,R5	SUB A, @RW5+d8	ADDC A, A,R5	ADDC A, @RW5+d8	CMP A, A,R5	CMP A, @RW5+d8	AND A, A,R5	AND A, @RW5+d8	OR A, A,R5	OR A, @RW5+d8	XOR A, A,R5	XOR A, @RW5+d8	DBNZ R5,r	DBNZ @ RW5+d8,r
+ 6	ADD A, A,R6	ADD A, @RW6+d8	SUB A, A,R6	SUB A, @RW6+d8	ADDC A, A,R6	ADDC A, @RW6+d8	CMP A, A,R6	CMP A, @RW6+d8	AND A, A,R6	AND A, @RW6+d8	OR A, A,R6	OR A, @RW6+d8	XOR A, A,R6	XOR A, @RW6+d8	DBNZ R6,r	DBNZ @ RW6+d8,r
+ 7	ADD A, A,R7	ADD A, @RW7+d8	SUB A, A,R7	SUB A, @RW7+d8	ADDC A, A,R7	ADDC A, @RW7+d8	CMP A, A,R7	CMP A, @RW7+d8	AND A, A,R7	AND A, @RW7+d8	OR A, A,R7	OR A, @RW7+d8	XOR A, A,R7	XOR A, @RW7+d8	DBNZ R7,r	DBNZ @ RW7+d8,r
+ 8	ADD A, A,@RW0	ADD A, @RW0+d16	SUB A, A,@RW0	SUB A, @RW0+d16	ADDC A, A,@RW0	ADDC A, @RW0+d16	CMP A, A,@RW0	CMP A, @RW0+d16	AND A, A,@RW0	AND A, @RW0+d16	OR A, A,@RW0	OR A, @RW0+d16	XOR A, A,@RW0	XOR A, @RW0+d16	DBNZ @RW0,r	DBNZ @R W0+d16,r
+ 9	ADD A, A,@RW1	ADD A, @RW1+d16	SUB A, A,@RW1	SUB A, @RW1+d16	ADDC A, A,@RW1	ADDC A, @RW1+d16	CMP A, A,@RW1	CMP A, @RW1+d16	AND A, A,@RW1	AND A, @RW1+d16	OR A, A,@RW1	OR A, @RW1+d16	XOR A, A,@RW1	XOR A, @RW1+d16	DBNZ @RW1,r	DBNZ @R W1+d16,r
+ A	ADD A,@RW2	ADD A, @RW2+d16	SUB A,@RW2	SUB A, @RW2+d16	ADDC A,@RW2	ADDC A, @RW2+d16	CMP A,@RW2	CMP A, @RW2+d16	AND A,@RW2	AND A, @RW2+d16	OR A,@RW2	OR A, @RW2+d16	XOR A,@RW2	XOR A, @RW2+d16	DBNZ @RW2,r	DBNZ @R W2+d16,r
+ B	ADD A,@RW3	ADD A, @RW3+d16	SUB A,@RW3	SUB A, @RW3+d16	ADDC A,@RW3	ADDC A, @RW3+d16	CMP A,@RW3	CMP A, @RW3+d16	AND A,@RW3	AND A, @RW3+d16	OR A,@RW3	OR A, @RW3+d16	XOR A,@RW3	XOR A, @RW3+d16	DBNZ @RW3,r	DBNZ @R W3+d16,r
+ C	ADD A,@RW0+	ADD A, @RW0+RW7	SUB A,@RW0+	SUB A, @RW0+RW7	ADDC A,@RW0+	ADDC A, @RW0+RW7	CMP A,@RW0+	CMP A, @RW0+RW7	AND A,@RW0+	AND A, @RW0+RW7	OR A,@RW0+	OR A, @RW0+RW7	XOR A,@RW0+	XOR A, @RW0+RW7	DBNZ @RW0+,r	DBNZ @R W0+RW7,r
+ D	ADD A,@RW1+	ADD A, @RW1+RW7	SUB A,@RW1+	SUB A, @RW1+RW7	ADDC A,@RW1+	ADDC A, @RW1+RW7	CMP A,@RW1+	CMP A, @RW1+RW7	AND A,@RW1+	AND A, @RW1+RW7	OR A,@RW1+	OR A, @RW1+RW7	XOR A,@RW1+	XOR A, @RW1+RW7	DBNZ @RW1+,r	DBNZ @R W1+RW7,r
+ E	ADD A,@RW2+	ADD A, @PC+d16	SUB A,@RW2+	SUB A, @PC+d16	ADDC A,@RW2+	ADDC A, @PC+d16	CMP A,@RW2+	CMP A, @PC+d16	AND A,@RW2+	AND A, @PC+d16	OR A,@RW2+	OR A, @PC+d16	XOR A,@RW2+	XOR A, @PC+d16	DBNZ @RW2+,r	DBNZ @ PC+d16,r
+ F	ADD A,@RW3+	ADD A, addr16	SUB A,@RW3+	SUB A, addr16	ADDC A,@RW3+	ADDC A, addr16	CMP A,@RW3+	CMP A, addr16	AND A,@RW3+	AND A, addr16	OR A,@RW3+	OR A, addr16	XOR A,@RW3+	XOR A, addr16	DBNZ @RW3+,r	DBNZ addr16,r

Table B.9-11 ea Instruction 6 (first byte = 75_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	ADD R0,A	ADD @R W0+d8,A	SUB R0,A	SUB @R W0+d8,A	SUBC A,R0	SUBC A, @RW0+d8	NEG R0	NEG @RW0+d8	AND R0,A	AND @R W0+d8,A	OR R0,A	OR @R W0+d8,A	XOR R0,A	XOR @R W0+d8,A	NOT R0	NOT @RW0+d8
+ 1	ADD R1,A	ADD @R W1+d8,A	SUB R1,A	SUB @R W1+d8,A	SUBC A,R1	SUBC A, @RW1+d8	NEG R1	NEG @RW1+d8	AND R1,A	AND @R W1+d8,A	OR R1,A	OR @R W1+d8,A	XOR R1,A	XOR @R W1+d8,A	NOT R1	NOT @RW1+d8
+ 2	ADD R2,A	ADD @R W2+d8,A	SUB R2,A	SUB @R W2+d8,A	SUBC A,R2	SUBC A, @RW2+d8	NEG R2	NEG @RW2+d8	AND R2,A	AND @R W2+d8,A	OR R2,A	OR @R W2+d8,A	XOR R2,A	XOR @R W2+d8,A	NOT R2	NOT @RW2+d8
+ 3	ADD R3,A	ADD @R W3+d8,A	SUB R3,A	SUB @R W3+d8,A	SUBC A,R3	SUBC A, @RW3+d8	NEG R3	NEG @RW3+d8	AND R3,A	AND @R W3+d8,A	OR R3,A	OR @R W3+d8,A	XOR R3,A	XOR @R W3+d8,A	NOT R3	NOT @RW3+d8
+ 4	ADD R4,A	ADD @R W4+d8,A	SUB R4,A	SUB @R W4+d8,A	SUBC A,R4	SUBC A, @RW4+d8	NEG R4	NEG @RW4+d8	AND R4,A	AND @R W4+d8,A	OR R4,A	OR @R W4+d8,A	XOR R4,A	XOR @R W4+d8,A	NOT R4	NOT @RW4+d8
+ 5	ADD R5,A	ADD @R W5+d8,A	SUB R5,A	SUB @R W5+d8,A	SUBC A,R5	SUBC A, @RW5+d8	NEG R5	NEG @RW5+d8	AND R5,A	AND @R W5+d8,A	OR R5,A	OR @R W5+d8,A	XOR R5,A	XOR @R W5+d8,A	NOT R5	NOT @RW5+d8
+ 6	ADD R6,A	ADD @R W6+d8,A	SUB R6,A	SUB @R W6+d8,A	SUBC A,R6	SUBC A, @RW6+d8	NEG R6	NEG @RW6+d8	AND R6,A	AND @R W6+d8,A	OR R6,A	OR @R W6+d8,A	XOR R6,A	XOR @R W6+d8,A	NOT R6	NOT @RW6+d8
+ 7	ADD R7,A	ADD @R W7+d8,A	SUB R7,A	SUB @R W7+d8,A	SUBC A,R7	SUBC A, @RW7+d8	NEG R7	NEG @RW7+d8	AND R7,A	AND @R W7+d8,A	OR R7,A	OR @R W7+d8,A	XOR R7,A	XOR @R W7+d8,A	NOT R7	NOT @RW7+d8
+ 8	ADD @RW0,A	ADD @R W0+d16,A	SUB @RW0,A	SUB @R W0+d16,A	SUBC A,@RW0	SUBC A, @RW0+d16	NEG @RW0	NEG @RW0+d16	AND @RW0,A	AND @R W0+d16,A	OR @RW0,A	OR @R W0+d16,A	XOR @RW0,A	XOR @R W0+d16,A	NOT @RW0	NOT @RW0+d16
+ 9	ADD @RW1,A	ADD @R W1+d16,A	SUB @RW1,A	SUB @R W1+d16,A	SUBC A,@RW1	SUBC A, @RW1+d16	NEG @RW1	NEG @RW1+d16	AND @RW1,A	AND @R W1+d16,A	OR @RW1,A	OR @R W1+d16,A	XOR @RW1,A	XOR @R W1+d16,A	NOT @RW1	NOT @RW1+d16
+ A	ADD @RW2,A	ADD @R W2+d16,A	SUB @RW2,A	SUB @R W2+d16,A	SUBC A,@RW2	SUBC A, @RW2+d16	NEG @RW2	NEG @RW2+d16	AND @RW2,A	AND @R W2+d16,A	OR @RW2,A	OR @R W2+d16,A	XOR @RW2,A	XOR @R W2+d16,A	NOT @RW2	NOT @RW2+d16
+ B	ADD @RW3,A	ADD @R W3+d16,A	SUB @RW3,A	SUB @R W3+d16,A	SUBC A,@RW3	SUBC A, @RW3+d16	NEG @RW3	NEG @RW3+d16	AND @RW3,A	AND @R W3+d16,A	OR @RW3,A	OR @R W3+d16,A	XOR @RW3,A	XOR @R W3+d16,A	NOT @RW3	NOT @RW3+d16
+ C	ADD @RW0+,A	ADD @R W0+RW7,A	SUB @RW0+,A	SUB @R W0+RW7,A	SUBC A,@RW0+	SUBC A, @RW0+RW7	NEG @RW0+	NEG @RW0+RW7	AND @RW0+,A	AND @R W0+RW7,A	OR @RW0+,A	OR @R W0+RW7,A	XOR @RW0+,A	XOR @R W0+RW7,A	NOT @RW0+	NOT @RW0+RW7
+ D	ADD @RW1+,A	ADD @R W1+RW7,A	SUB @RW1+,A	SUB @R W1+RW7,A	SUBC A,@RW1+	SUBC A, @RW1+RW7	NEG @RW1+	NEG @RW1+RW7	AND @RW1+,A	AND @R W1+RW7,A	OR @RW1+,A	OR @R W1+RW7,A	XOR @RW1+,A	XOR @R W1+RW7,A	NOT @RW1+	NOT @RW1+RW7
+ E	ADD @RW2+,A	ADD @P C+d16,A	SUB @RW2+,A	SUB @P C+d16,A	SUBC A,@RW2+	SUBC A, @PC+d16	NEG @RW2+	NEG @PC+d16	AND @RW2+,A	AND @P C+d16,A	OR @RW2+,A	OR @P C+d16,A	XOR @RW2+,A	XOR @P C+d16,A	NOT @RW2+	NOT @PC+d16
+ F	ADD @RW3+,A	ADD addr16,A	SUB @RW3+,A	SUB addr16,A	SUBC A,@RW3+	SUBC A, addr16	NEG @RW3+	NEG addr16	AND @RW3+,A	AND addr16,A	OR @RW3+,A	OR addr16,A	XOR @RW3+,A	XOR addr16,A	NOT @RW3+	NOT addr16

Table B.9-12 ea Instruction 7 (first byte = 76_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	ADDW A,RW0	ADDW A,@RW0+d8	SUBW A,@RW0	SUBW A,@RW0+d8	ADDCW A,@RW0	ADDCW A,@RW0+d8	CMPW A,@RW0	CMPW A,@RW0+d8	ANDW A,@RW0	ANDW A,@RW0+d8	ORW A,@RW0	ORW A,@RW0+d8	XORW A,@RW0	XORW A,@RW0+d8	DWBNZ RW0,r	DWBNZ @RW0+d8,r
+ 1	ADDW A,@RW1	ADDW A,@RW1+d8	SUBW A,@RW1	SUBW A,@RW1+d8	ADDCW A,@RW1	ADDCW A,@RW1+d8	CMPW A,@RW1	CMPW A,@RW1+d8	ANDW A,@RW1	ANDW A,@RW1+d8	ORW A,@RW1	ORW A,@RW1+d8	XORW A,@RW1	XORW A,@RW1+d8	DWBNZ RW1,r	DWBNZ @RW1+d8,r
+ 2	ADDW A,@RW2	ADDW A,@RW2+d8	SUBW A,@RW2	SUBW A,@RW2+d8	ADDCW A,@RW2	ADDCW A,@RW2+d8	CMPW A,@RW2	CMPW A,@RW2+d8	ANDW A,@RW2	ANDW A,@RW2+d8	ORW A,@RW2	ORW A,@RW2+d8	XORW A,@RW2	XORW A,@RW2+d8	DWBNZ RW2,r	DWBNZ @RW2+d8,r
+ 3	ADDW A,@RW3	ADDW A,@RW3+d8	SUBW A,@RW3	SUBW A,@RW3+d8	ADDCW A,@RW3	ADDCW A,@RW3+d8	CMPW A,@RW3	CMPW A,@RW3+d8	ANDW A,@RW3	ANDW A,@RW3+d8	ORW A,@RW3	ORW A,@RW3+d8	XORW A,@RW3	XORW A,@RW3+d8	DWBNZ RW3,r	DWBNZ @RW3+d8,r
+ 4	ADDW A,@RW4	ADDW A,@RW4+d8	SUBW A,@RW4	SUBW A,@RW4+d8	ADDCW A,@RW4	ADDCW A,@RW4+d8	CMPW A,@RW4	CMPW A,@RW4+d8	ANDW A,@RW4	ANDW A,@RW4+d8	ORW A,@RW4	ORW A,@RW4+d8	XORW A,@RW4	XORW A,@RW4+d8	DWBNZ RW4,r	DWBNZ @RW4+d8,r
+ 5	ADDW A,@RW5	ADDW A,@RW5+d8	SUBW A,@RW5	SUBW A,@RW5+d8	ADDCW A,@RW5	ADDCW A,@RW5+d8	CMPW A,@RW5	CMPW A,@RW5+d8	ANDW A,@RW5	ANDW A,@RW5+d8	ORW A,@RW5	ORW A,@RW5+d8	XORW A,@RW5	XORW A,@RW5+d8	DWBNZ RW5,r	DWBNZ @RW5+d8,r
+ 6	ADDW A,@RW6	ADDW A,@RW6+d8	SUBW A,@RW6	SUBW A,@RW6+d8	ADDCW A,@RW6	ADDCW A,@RW6+d8	CMPW A,@RW6	CMPW A,@RW6+d8	ANDW A,@RW6	ANDW A,@RW6+d8	ORW A,@RW6	ORW A,@RW6+d8	XORW A,@RW6	XORW A,@RW6+d8	DWBNZ RW6,r	DWBNZ @RW6+d8,r
+ 7	ADDW A,@RW7	ADDW A,@RW7+d8	SUBW A,@RW7	SUBW A,@RW7+d8	ADDCW A,@RW7	ADDCW A,@RW7+d8	CMPW A,@RW7	CMPW A,@RW7+d8	ANDW A,@RW7	ANDW A,@RW7+d8	ORW A,@RW7	ORW A,@RW7+d8	XORW A,@RW7	XORW A,@RW7+d8	DWBNZ RW7,r	DWBNZ @RW7+d8,r
+ 8	ADDW A,@RW0	ADDW A,@RW0+d16	SUBW A,@RW0	SUBW A,@RW0+d16	ADDCW A,@RW0	ADDCW A,@RW0+d16	CMPW A,@RW0	CMPW A,@RW0+d16	ANDW A,@RW0	ANDW A,@RW0+d16	ORW A,@RW0	ORW A,@RW0+d16	XORW A,@RW0	XORW A,@RW0+d16	DWBNZ @RW0,r	DWBNZ @R W0+d16,r
+ 9	ADDW A,@RW1	ADDW A,@RW1+d16	SUBW A,@RW1	SUBW A,@RW1+d16	ADDCW A,@RW1	ADDCW A,@RW1+d16	CMPW A,@RW1	CMPW A,@RW1+d16	ANDW A,@RW1	ANDW A,@RW1+d16	ORW A,@RW1	ORW A,@RW1+d16	XORW A,@RW1	XORW A,@RW1+d16	DWBNZ @RW1,r	DWBNZ @R W1+d16,r
+ A	ADDW A,@RW2	ADDW A,@RW2+d16	SUBW A,@RW2	SUBW A,@RW2+d16	ADDCW A,@RW2	ADDCW A,@RW2+d16	CMPW A,@RW2	CMPW A,@RW2+d16	ANDW A,@RW2	ANDW A,@RW2+d16	ORW A,@RW2	ORW A,@RW2+d16	XORW A,@RW2	XORW A,@RW2+d16	DWBNZ @RW2,r	DWBNZ @R W2+d16,r
+ B	ADDW A,@RW3	ADDW A,@RW3+d16	SUBW A,@RW3	SUBW A,@RW3+d16	ADDCW A,@RW3	ADDCW A,@RW3+d16	CMPW A,@RW3	CMPW A,@RW3+d16	ANDW A,@RW3	ANDW A,@RW3+d16	ORW A,@RW3	ORW A,@RW3+d16	XORW A,@RW3	XORW A,@RW3+d16	DWBNZ @RW3,r	DWBNZ @R W3+d16,r
+ C	ADDW A,@RW0+	ADDW A,@RW0+RW7	SUBW A,@RW0+	SUBW A,@RW0+RW7	ADDCW A,@RW0+	ADDCW A,@RW0+RW7	CMPW A,@RW0+	CMPW A,@RW0+RW7	ANDW A,@RW0+	ANDW A,@RW0+RW7	ORW A,@RW0+	ORW A,@RW0+RW7	XORW A,@RW0+	XORW A,@RW0+RW7	DWBNZ @RW0+,r	DWBNZ @R W0+RW7,r
+ D	ADDW A,@RW1+	ADDW A,@RW1+RW7	SUBW A,@RW1+	SUBW A,@RW1+RW7	ADDCW A,@RW1+	ADDCW A,@RW1+RW7	CMPW A,@RW1+	CMPW A,@RW1+RW7	ANDW A,@RW1+	ANDW A,@RW1+RW7	ORW A,@RW1+	ORW A,@RW1+RW7	XORW A,@RW1+	XORW A,@RW1+RW7	DWBNZ @RW1+,r	DWBNZ @R W1+RW7,r
+ E	ADDW A,@RW2+	ADDW A,@PC+d16	SUBW A,@RW2+	SUBW A,@PC+d16	ADDCW A,@RW2+	ADDCW A,@PC+d16	CMPW A,@RW2+	CMPW A,@PC+d16	ANDW A,@RW2+	ANDW A,@PC+d16	ORW A,@RW2+	ORW A,@PC+d16	XORW A,@RW2+	XORW A,@PC+d16	DWBNZ @RW2+,r	DWBNZ @R PC+d16,r
+ F	ADDW A,@RW3+	ADDW A,addr16	SUBW A,@RW3+	SUBW A,addr16	ADDCW A,@RW3+	ADDCW A,addr16	CMPW A,@RW3+	CMPW A,addr16	ANDW A,@RW3+	ANDW A,addr16	ORW A,@RW3+	ORW A,addr16	XORW A,@RW3+	XORW A,addr16	DWBNZ @RW3+,r	DWBNZ addr16,r

Table B.9-13 ea Instruction 8 (first byte = 77_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	RW0,A	W0+d8,A	RW0,A	W0+d8,A	A,RW0	@RW0+d8	RW0	@RW0+d8	RW0,A	W0+d8,A	RW0,A	W0+d8,A	RW0,A	W0+d8,A	RW0	@RW0+d8
+ 1	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	RW1,A	W1+d8,A	RW1,A	W1+d8,A	A,RW1	@RW1+d8	RW1	@RW1+d8	RW1,A	W1+d8,A	RW1,A	W1+d8,A	RW1,A	W1+d8,A	RW1	@RW1+d8
+ 2	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	RW2,A	W2+d8,A	RW2,A	W2+d8,A	A,RW2	@RW2+d8	RW2	@RW2+d8	RW2,A	W2+d8,A	RW2,A	W2+d8,A	RW2,A	W2+d8,A	RW2	@RW2+d8
+ 3	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	RW3,A	W3+d8,A	RW3,A	W3+d8,A	A,RW3	@RW3+d8	RW3	@RW3+d8	RW3,A	W3+d8,A	RW3,A	W3+d8,A	RW3,A	W3+d8,A	RW3	@RW3+d8
+ 4	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	RW4,A	W4+d8,A	RW4,A	W4+d8,A	A,RW4	@RW4+d8	RW4	@RW4+d8	RW4,A	W4+d8,A	RW4,A	W4+d8,A	RW4,A	W4+d8,A	RW4	@RW4+d8
+ 5	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	RW5,A	W5+d8,A	RW5,A	W5+d8,A	A,RW5	@RW5+d8	RW5	@RW5+d8	RW5,A	W5+d8,A	RW5,A	W5+d8,A	RW5,A	W5+d8,A	RW5	@RW5+d8
+ 6	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	RW6,A	W6+d8,A	RW6,A	W6+d8,A	A,RW6	@RW6+d8	RW6	@RW6+d8	RW6,A	W6+d8,A	RW6,A	W6+d8,A	RW6,A	W6+d8,A	RW6	@RW6+d8
+ 7	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	RW7,A	W7+d8,A	RW7,A	W7+d8,A	A,RW7	@RW7+d8	RW7	@RW7+d8	RW7,A	W7+d8,A	RW7,A	W7+d8,A	RW7,A	W7+d8,A	RW7	@RW7+d8
+ 8	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	@RW0,A	W0+d16,A	@RW0,A	W0+d16,A	A,@RW0	@RW0+d16	@RW0	@RW0+d16	@RW0,A	W0+d16,A	@RW0,A	W0+d16,A	@RW0,A	W0+d16,A	@RW0	@RW0+d16
+ 9	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	@RW1,A	W1+d16,A	@RW1,A	W1+d16,A	A,@RW1	@RW1+d16	@RW1	@RW1+d16	@RW1,A	W1+d16,A	@RW1,A	W1+d16,A	@RW1,A	W1+d16,A	@RW1	@RW1+d16
+ A	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	@RW2,A	W2+d16,A	@RW2,A	W2+d16,A	A,@RW2	@RW2+d16	@RW2	@RW2+d16	@RW2,A	W2+d16,A	@RW2,A	W2+d16,A	@RW2,A	W2+d16,A	@RW2	@RW2+d16
+ B	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	@RW3,A	W3+d16,A	@RW3,A	W3+d16,A	A,@RW3	@RW3+d16	@RW3	@RW3+d16	@RW3,A	W3+d16,A	@RW3,A	W3+d16,A	@RW3,A	W3+d16,A	@RW3	@RW3+d16
+ C	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	@RW0+,A	W0+RW7,A	@RW0+,A	W0+RW7,A	A,@RW0+	@RW0+RW7	@RW0+	@RW0+RW7	@RW0+,A	W0+RW7,A	@RW0+,A	W0+RW7,A	@RW0+,A	W0+RW7,A	@RW0+	@RW0+RW7
+ D	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	@RW1+,A	W1+RW7,A	@RW1+,A	W1+RW7,A	A,@RW1+	@RW1+RW7	@RW1+	@RW1+RW7	@RW1+,A	W1+RW7,A	@RW1+,A	W1+RW7,A	@RW1+,A	W1+RW7,A	@RW1+	@RW1+RW7
+ E	ADDW	ADDW @P	SUBW	SUBW @P	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @P	ORW	ORW @P	XORW	XORW @P	NOTW	NOTW
	@RW2+,A	C+d16,A	@RW2+,A	C+d16,A	A,@RW2+	@PC+d16	@RW2+	@PC+d16	@RW2+,A	C+d16,A	@RW2+,A	C+d16,A	@RW2+,A	C+d16,A	@RW2+	@PC+d16
+ F	ADDW	ADDW	SUBW	SUBW	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW	ORW	ORW	XORW	XORW	NOTW	NOTW
	@RW3+,A	addr16,A	@RW3+,A	addr16,A	A,@RW3+	addr16	@RW3+	addr16	@RW3+,A	addr16,A	@RW3+,A	addr16,A	@RW3+,A	addr16,A	@RW3+	addr16

Table B.9-14 ea Instruction 9 (first byte = 78_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MULU A, A,R0	MULU A, @RW0+d8	MULUW A,RW0	MULUW A, @RW0+d8	MUL A R0	MUL A, @RW0+d8	MULW A,RW0	MULW A, @RW0+d8	DIVU A,R0	DIVU A, @RW0+d8	DIVUW A,RW0	DIVUW A, @RW0+d8	DIV A,R0	DIV A, @RW0+d8	DIVW A,RW0	DIVW A, @RW0+d8
+ 1	MULU A, A,R1	MULU A, @RW1+d8	MULUW A,RW1	MULUW A, @RW1+d8	MUL A,R1	MUL A, @RW1+d8	MULW A,RW1	MULW A, @RW1+d8	DIVU A,R1	DIVU A, @RW1+d8	DIVUW A,RW1	DIVUW A, @RW1+d8	DIV A,R1	DIV A, @RW1+d8	DIVW A,RW1	DIVW A, @RW1+d8
+ 2	MULU A, A,R2	MULU A, @RW2+d8	MULUW A,RW2	MULUW A, @RW2+d8	MUL A,R2	MUL A, @RW2+d8	MULW A RW2	MULW A, @RW2+d8	DIVU A,R2	DIVU A, @RW2+d8	DIVUW A,RW2	DIVUW A, @RW2+d8	DIV A,R2	DIV A, @RW2+d8	DIVW A,RW2	DIVW A, @RW2+d8
+ 3	MULU A, A,R3	MULU A, @RW3+d8	MULUW A,RW3	MULUW A, @RW3+d8	MUL A,R3	MUL A, @RW3+d8	MULW A RW3	MULW A, @RW3+d8	DIVU A,R3	DIVU A, @RW3+d8	DIVUW A,RW3	DIVUW A, @RW3+d8	DIV A,R3	DIV A, @RW3+d8	DIVW A,RW3	DIVW A, @RW3+d8
+ 4	MULU A, A,R4	MULU A, @RW4+d8	MULUW A,RW4	MULUW A, @RW4+d8	MUL A,R4	MUL A, @RW4+d8	MULW A RW4	MULW A, @RW4+d8	DIVU A,R4	DIVU A, @RW4+d8	DIVUW A,RW4	DIVUW A, @RW4+d8	DIV A,R4	DIV A, @RW4+d8	DIVW A,RW4	DIVW A, @RW4+d8
+ 5	MULU A, A,R5	MULU A, @RW5+d8	MULUW A,RW5	MULUW A, @RW5+d8	MUL A,R5	MUL A, @RW5+d8	MULW A RW5	MULW A, @RW5+d8	DIVU A,R5	DIVU A, @RW5+d8	DIVUW A,RW5	DIVUW A, @RW5+d8	DIV A,R5	DIV A, @RW5+d8	DIVW A,RW5	DIVW A, @RW5+d8
+ 6	MULU A, A,R6	MULU A, @RW6+d8	MULUW A,RW6	MULUW A, @RW6+d8	MUL A R6	MUL A, @RW6+d8	MULW A,RW6	MULW A, @RW6+d8	DIVU A,R6	DIVU A, @RW6+d8	DIVUW A,RW6	DIVUW A, @RW6+d8	DIV A,R6	DIV A, @RW6+d8	DIVW A,RW6	DIVW A, @RW6+d8
+ 7	MULU A, A,R7	MULU A, @RW7+d8	MULUW A,RW7	MULUW A, @RW7+d8	MUL A,R7	MUL A, @RW7+d8	MULW A,RW7	MULW A, @RW7+d8	DIVU A,R7	DIVU A, @RW7+d8	DIVUW A,RW7	DIVUW A, @RW7+d8	DIV A,R7	DIV A, @RW7+d8	DIVW A,RW7	DIVW A, @RW7+d8
+ 8	MULU A, A,@RW0	MULU A, @RW0+d16	MULUW A,@RW0	MULUW A, @RW0+d16	MUL A,@RW0	MUL A, @RW0+d16	MULW A,@RW0	MULW A, @RW0+d16	DIVU A,@RW0	DIVU A, @RW0+d16	DIVUW A,@RW0	DIVUW A, @RW0+d16	DIV A,@RW0	DIV A, @RW0+d16	DIVW A,@RW0	DIVW A, @RW0+d16
+ 9	MULU A, A,@RW1	MULU A, @RW1+d16	MULUW A,@RW1	MULUW A, @RW1+d16	MUL A,@RW1	MUL A, @RW1+d16	MULW A @RW1	MULW A, @RW1+d16	DIVU A,@RW1	DIVU A, @RW1+d16	DIVUW A,@RW1	DIVUW A, @RW1+d16	DIV A,@RW1	DIV A, @RW1+d16	DIVW A,@RW1	DIVW A, @RW1+d16
+ A	MULU A, A,@RW2	MULU A, @RW2+d16	MULUW A,@RW2	MULUW A, @RW2+d16	MUL A,@RW2	MUL A, @RW2+d16	MULW A,@RW2	MULW A, @RW2+d16	DIVU A,@RW2	DIVU A, @RW2+d16	DIVUW A,@RW2	DIVUW A, @RW2+d16	DIV A,@RW2	DIV A, @RW2+d16	DIVW A,@RW2	DIVW A, @RW2+d16
+ B	MULU A, A,@RW3	MULU A, @RW3+d16	MULUW A,@RW3	MULUW A, @RW3+d16	MUL A,@RW3	MUL A, @RW3+d16	MULW A,@RW3	MULW A, @RW3+d16	DIVU A,@RW3	DIVU A, @RW3+d16	DIVUW A,@RW3	DIVUW A, @RW3+d16	DIV A,@RW3	DIV A, @RW3+d16	DIVW A,@RW3	DIVW A, @RW3+d16
+ C	MULU A, A,@RW0+	MULU A, @RW0+RW7	MULUW A,@RW0+	MULUW A, @RW0+RW7	MUL A,@RW0+	MUL A, @RW0+RW7	MULW A,@RW0+	MULW A, @RW0+RW7	DIVU A,@RW0+	DIVU A, @RW0+RW7	DIVUW A,@RW0+	DIVUW A, @RW0+RW7	DIV A,@RW0+	DIV A, @RW0+RW7	DIVW A,@RW0+	DIVW A, @RW0+RW7
+ D	MULU A, A,@RW1+	MULU A, @RW1+RW7	MULUW A,@RW1+	MULUW A, @RW1+RW7	MUL A,@RW1+	MUL A, @RW1+RW7	MULW A,@RW1+	MULW A, @RW1+RW7	DIVU A,@RW1+	DIVU A, @RW1+RW7	DIVUW A,@RW1+	DIVUW A, @RW1+RW7	DIV A,@RW1+	DIV A, @RW1+RW7	DIVW A,@RW1+	DIVW A, @RW1+RW7
+ E	MULU A, A,@RW2+	MULU A, @PC+d16	MULUW A,@RW2+	MULUW A, @PC+d16	MUL A,@RW2+	MUL A, @PC+d16	MULW A,@RW2+	MULW A, @PC+d16	DIVU A,@RW2+	DIVU A, @PC+d16	DIVUW A,@RW2+	DIVUW A, @PC+d16	DIV A,@RW2+	DIV A, @PC+d16	DIVW A,@RW2+	DIVW A, @PC+d16
+ F	MULU A, A,@RW3+	MULU A, addr16	MULUW A,@RW3+	MULUW A, addr16	MUL A,@RW3+	MUL A, addr16	MULW A,@RW3+	MULW A, addr16	DIVU A,@RW3+	DIVU A, addr16	DIVUW A,@RW3+	DIVUW A, addr16	DIV A,@RW3+	DIV A, addr16	DIVW A,@RW3+	DIVW A, addr16

Table B.9-15 MOVEA RWi, ea Instruction (first byte = 79_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOVEA RW0,RW0	MOVEA RW0,@RW0+d8	MOVEA RW1,@RW0	MOVEA RW1,@RW0+d8	MOVEA RW2,RW0	MOVEA RW2,@RW0+d8	MOVEA RW3,RW0	MOVEA RW3,@RW0+d8	MOVEA RW4,RW0	MOVEA RW4,@RW0+d8	MOVEA RW5,RW0	MOVEA RW5,@RW0+d8	MOVEA RW6,RW0	MOVEA RW6,@RW0+d8	MOVEA RW7,RW0	MOVEA RW7,@RW0+d8
+ 1	MOVEA RW0,RW1	MOVEA RW0,@RW1+d8	MOVEA RW1,RW1	MOVEA RW1,@RW1+d8	MOVEA RW2,RW1	MOVEA RW2,@RW1+d8	MOVEA RW3,RW1	MOVEA RW3,@RW1+d8	MOVEA RW4,RW1	MOVEA RW4,@RW1+d8	MOVEA RW5,RW1	MOVEA RW5,@RW1+d8	MOVEA RW6,RW1	MOVEA RW6,@RW1+d8	MOVEA RW7,RW1	MOVEA RW7,@RW1+d8
+ 2	MOVEA RW0,RW2	MOVEA RW0,@RW2+d8	MOVEA RW1,RW2	MOVEA RW1,@RW2+d8	MOVEA RW2,RW2	MOVEA RW2,@RW2+d8	MOVEA RW3,RW2	MOVEA RW3,@RW2+d8	MOVEA RW4,RW2	MOVEA RW4,@RW2+d8	MOVEA RW5,RW2	MOVEA RW5,@RW2+d8	MOVEA RW6,RW2	MOVEA RW6,@RW2+d8	MOVEA RW7,RW2	MOVEA RW7,@RW2+d8
+ 3	MOVEA RW0,RW3	MOVEA RW0,@RW3+d8	MOVEA RW1,RW3	MOVEA RW1,@RW3+d8	MOVEA RW2,RW3	MOVEA RW2,@RW3+d8	MOVEA RW3,RW3	MOVEA RW3,@RW3+d8	MOVEA RW4,RW3	MOVEA RW4,@RW3+d8	MOVEA RW5,RW3	MOVEA RW5,@RW3+d8	MOVEA RW6,RW3	MOVEA RW6,@RW3+d8	MOVEA RW7,RW3	MOVEA RW7,@RW3+d8
+ 4	MOVEA RW0,RW4	MOVEA RW0,@RW4+d8	MOVEA RW1,RW4	MOVEA RW1,@RW4+d8	MOVEA RW2,RW4	MOVEA RW2,@RW4+d8	MOVEA RW3,RW4	MOVEA RW3,@RW4+d8	MOVEA RW4,RW4	MOVEA RW4,@RW4+d8	MOVEA RW5,RW4	MOVEA RW5,@RW4+d8	MOVEA RW6,RW4	MOVEA RW6,@RW4+d8	MOVEA RW7,RW4	MOVEA RW7,@RW4+d8
+ 5	MOVEA RW0,RW5	MOVEA RW0,@RW5+d8	MOVEA RW1,RW5	MOVEA RW1,@RW5+d8	MOVEA RW2,RW5	MOVEA RW2,@RW5+d8	MOVEA RW3,RW5	MOVEA RW3,@RW5+d8	MOVEA RW4,RW5	MOVEA RW4,@RW5+d8	MOVEA RW5,RW5	MOVEA RW5,@RW5+d8	MOVEA RW6,RW5	MOVEA RW6,@RW5+d8	MOVEA RW7,RW5	MOVEA RW7,@RW5+d8
+ 6	MOVEA RW0,RW6	MOVEA RW0,@RW6+d8	MOVEA RW1,RW6	MOVEA RW1,@RW6+d8	MOVEA RW2,RW6	MOVEA RW2,@RW6+d8	MOVEA RW3,RW6	MOVEA RW3,@RW6+d8	MOVEA RW4,RW6	MOVEA RW4,@RW6+d8	MOVEA RW5,RW6	MOVEA RW5,@RW6+d8	MOVEA RW6,RW6	MOVEA RW6,@RW6+d8	MOVEA RW7,RW6	MOVEA RW7,@RW6+d8
+ 7	MOVEA RW0,RW7	MOVEA RW0,@RW7+d8	MOVEA RW1,RW7	MOVEA RW1,@RW7+d8	MOVEA RW2,RW7	MOVEA RW2,@RW7+d8	MOVEA RW3,RW7	MOVEA RW3,@RW7+d8	MOVEA RW4,RW7	MOVEA RW4,@RW7+d8	MOVEA RW5,RW7	MOVEA RW5,@RW7+d8	MOVEA RW6,RW7	MOVEA RW6,@RW7+d8	MOVEA RW7,RW7	MOVEA RW7,@RW7+d8
+ 8	MOVEA RW0,@RW0	MOVEA RW0,@RW0+d16	MOVEA RW1,@RW0	MOVEA RW1,@RW0+d16	MOVEA RW2,@RW0	MOVEA RW2,@RW0+d16	MOVEA RW3,@RW0	MOVEA RW3,@RW0+d16	MOVEA RW4,@RW0	MOVEA RW4,@RW0+d16	MOVEA RW5,@RW0	MOVEA RW5,@RW0+d16	MOVEA RW6,@RW0	MOVEA RW6,@RW0+d16	MOVEA RW7,@RW0	MOVEA RW7,@RW0+d16
+ 9	MOVEA RW0,@RW1	MOVEA RW0,@RW1+d16	MOVEA RW1,@RW1	MOVEA RW1,@RW1+d16	MOVEA RW2,@RW1	MOVEA RW2,@RW1+d16	MOVEA RW3,@RW1	MOVEA RW3,@RW1+d16	MOVEA RW4,@RW1	MOVEA RW4,@RW1+d16	MOVEA RW5,@RW1	MOVEA RW5,@RW1+d16	MOVEA RW6,@RW1	MOVEA RW6,@RW1+d16	MOVEA RW7,@RW1	MOVEA RW7,@RW1+d16
+ A	MOVEA RW0,@RW2	MOVEA RW0,@RW2+d16	MOVEA RW1,@RW2	MOVEA RW1,@RW2+d16	MOVEA RW2,@RW2	MOVEA RW2,@RW2+d16	MOVEA RW3,@RW2	MOVEA RW3,@RW2+d16	MOVEA RW4,@RW2	MOVEA RW4,@RW2+d16	MOVEA RW5,@RW2	MOVEA RW5,@RW2+d16	MOVEA RW6,@RW2	MOVEA RW6,@RW2+d16	MOVEA RW7,@RW2	MOVEA RW7,@RW2+d16
+ B	MOVEA RW0,@RW3	MOVEA RW0,@RW3+d16	MOVEA RW1,@RW3	MOVEA RW1,@RW3+d16	MOVEA RW2,@RW3	MOVEA RW2,@RW3+d16	MOVEA RW3,@RW3	MOVEA RW3,@RW3+d16	MOVEA RW4,@RW3	MOVEA RW4,@RW3+d16	MOVEA RW5,@RW3	MOVEA RW5,@RW3+d16	MOVEA RW6,@RW3	MOVEA RW6,@RW3+d16	MOVEA RW7,@RW3	MOVEA RW7,@RW3+d16
+ C	MOVEA R	MOVEA RW0	MOVEA R	MOVEA RW1	MOVEA R	MOVEA RW2	MOVEA R	MOVEA RW3	MOVEA R	MOVEA RW4	MOVEA R	MOVEA RW5	MOVEA R	MOVEA RW6	MOVEA R	MOVEA RW7
	W0,@RW0+	@RW0+RW7	W1,@RW0+	@RW0+RW7	W2,@RW0+	@RW0+RW7	W3,@RW0+	@RW0+RW7	W4,@RW0+	@RW0+RW7	W5,@RW0+	@RW0+RW7	W6,@RW0+	@RW0+RW7	W7,@RW0+	@RW0+RW7
+ D	MOVEA R	MOVEA RW0	MOVEA R	MOVEA RW1	MOVEA R	MOVEA RW2	MOVEA R	MOVEA RW3	MOVEA R	MOVEA RW4	MOVEA R	MOVEA RW5	MOVEA R	MOVEA RW6	MOVEA R	MOVEA RW7
	W0,@RW1+	@RW1+RW7	W1,@RW1+	@RW1+RW7	W2,@RW1+	@RW1+RW7	W3,@RW1+	@RW1+RW7	W4,@RW1+	@RW1+RW7	W5,@RW1+	@RW1+RW7	W6,@RW1+	@RW1+RW7	W7,@RW1+	@RW1+RW7
+ E	MOVEA R	MOVEA RW0	MOVEA R	MOVEA RW1	MOVEA R	MOVEA RW2	MOVEA R	MOVEA RW3	MOVEA R	MOVEA RW4	MOVEA R	MOVEA RW5	MOVEA R	MOVEA RW6	MOVEA R	MOVEA RW7
	W0,@RW2+	@PC+d16	W1,@RW2+	@PC+d16	W2,@RW2+	@PC+d16	W3,@RW2+	@PC+d16	W4,@RW2+	@PC+d16	W5,@RW2+	@PC+d16	W6,@RW2+	@PC+d16	W7,@RW2+	@PC+d16
+ F	MOVEA R	MOVEA RW0	MOVEA R	MOVEA RW1	MOVEA R	MOVEA RW2	MOVEA R	MOVEA RW3	MOVEA R	MOVEA RW4	MOVEA R	MOVEA RW5	MOVEA R	MOVEA RW6	MOVEA R	MOVEA RW7
	W0,@RW3+	,addr16	W1,@RW3+	,addr16	W2,@RW3+	,addr16	W3,@RW3+	,addr16	W4,@RW3+	,addr16	W5,@RW3+	,addr16	W6,@RW3+	,addr16	W7,@RW3+	,addr16

Table B.9-16 MOV Ri, ea Instruction (first byte = $7A_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOV R0, R0,R0	MOV R0, @RW0+d8	MOV R1,R0	MOV R1, @RW0+d8	MOV R2,R0	MOV R2, @RW0+d8	MOV R3,R0	MOV R3, @RW0+d8	MOV R4,R0	MOV R4, @RW0+d8	MOV R5,R0	MOV R5, @RW0+d8	MOV R6,R0	MOV R6, @RW0+d8	MOV R7,R0	MOV R7, @RW0+d8
+ 1	MOV R0,R1	MOV R0, @RW1+d8	MOV R1,R1	MOV R1, @RW1+d8	MOV R2,R1	MOV R2, @RW1+d8	MOV R3,R1	MOV R3, @RW1+d8	MOV R4,R1	MOV R4, @RW1+d8	MOV R5,R1	MOV R5, @RW1+d8	MOV R6,R1	MOV R6, @RW1+d8	MOV R7,R1	MOV R7, @RW1+d8
+ 2	MOV R0,R2	MOV R0, @RW2+d8	MOV R1,R2	MOV R1, @RW2+d8	MOV R2,R2	MOV R2, @RW2+d8	MOV R3,R2	MOV R3, @RW2+d8	MOV R4,R2	MOV R4, @RW2+d8	MOV R5,R2	MOV R5, @RW2+d8	MOV R6,R2	MOV R6, @RW2+d8	MOV R7,R2	MOV R7, @RW2+d8
+ 3	MOV R0,R3	MOV R0, @RW3+d8	MOV R1,R3	MOV R1, @RW3+d8	MOV R2,R3	MOV R2, @RW3+d8	MOV R3,R3	MOV R3, @RW3+d8	MOV R4,R3	MOV R4, @RW3+d8	MOV R5,R3	MOV R5, @RW3+d8	MOV R6,R3	MOV R6, @RW3+d8	MOV R7,R3	MOV R7, @RW3+d8
+ 4	MOV R0,R4	MOV R0, @RW4+d8	MOV R1,R4	MOV R1, @RW4+d8	MOV R2,R4	MOV R2, @RW4+d8	MOV R3,R4	MOV R3, @RW4+d8	MOV R4,R4	MOV R4, @RW4+d8	MOV R5,R4	MOV R5, @RW4+d8	MOV R6,R4	MOV R6, @RW4+d8	MOV R7,R4	MOV R7, @RW4+d8
+ 5	MOV R0,R5	MOV R0, @RW5+d8	MOV R1,R5	MOV R1, @RW5+d8	MOV R2,R5	MOV R2, @RW5+d8	MOV R3,R5	MOV R3, @RW5+d8	MOV R4,R5	MOV R4, @RW5+d8	MOV R5,R5	MOV R5, @RW5+d8	MOV R6,R5	MOV R6, @RW5+d8	MOV R7,R5	MOV R7, @RW5+d8
+ 6	MOV R0,R6	MOV R0, @RW6+d8	MOV R1,R6	MOV R1, @RW6+d8	MOV R2,R6	MOV R2, @RW6+d8	MOV R3,R6	MOV R3, @RW6+d8	MOV R4,R6	MOV R4, @RW6+d8	MOV R5,R6	MOV R5, @RW6+d8	MOV R6,R6	MOV R6, @RW6+d8	MOV R7,R6	MOV R7, @RW6+d8
+ 7	MOV R0,R7	MOV R0, @RW7+d8	MOV R1,R7	MOV R1, @RW7+d8	MOV R2,R7	MOV R2, @RW7+d8	MOV R3,R7	MOV R3, @RW7+d8	MOV R4,R7	MOV R4, @RW7+d8	MOV R5,R7	MOV R5, @RW7+d8	MOV R6,R7	MOV R6, @RW7+d8	MOV R7,R7	MOV R7, @RW7+d8
+ 8	MOV R0,@RW0	MOV R0, @RW0+d16	MOV R1,@RW0	MOV R1, @RW0+d16	MOV R2,@RW0	MOV R2, @RW0+d16	MOV R3,@RW0	MOV R3, @RW0+d16	MOV R4,@RW0	MOV R4, @RW0+d16	MOV R5,@RW0	MOV R5, @RW0+d16	MOV R6,@RW0	MOV R6, @RW0+d16	MOV R7,@RW0	MOV R7, @RW0+d16
+ 9	MOV R0,@RW1	MOV R0, @RW1+d16	MOV R1,@RW1	MOV R1, @RW1+d16	MOV R2,@RW1	MOV R2, @RW1+d16	MOV R3,@RW1	MOV R3, @RW1+d16	MOV R4,@RW1	MOV R4, @RW1+d16	MOV R5,@RW1	MOV R5, @RW1+d16	MOV R6,@RW1	MOV R6, @RW1+d16	MOV R7,@RW1	MOV R7, @RW1+d16
+ A	MOV R0,@RW2	MOV R0, @RW2+d16	MOV R1,@RW2	MOV R1, @RW2+d16	MOV R2,@RW2	MOV R2, @RW2+d16	MOV R3,@RW2	MOV R3, @RW2+d16	MOV R4,@RW2	MOV R4, @RW2+d16	MOV R5,@RW2	MOV R5, @RW2+d16	MOV R6,@RW2	MOV R6, @RW2+d16	MOV R7,@RW2	MOV R7, @RW2+d16
+ B	MOV R0,@RW3	MOV R0, @RW3+d16	MOV R1,@RW3	MOV R1, @RW3+d16	MOV R2,@RW3	MOV R2, @RW3+d16	MOV R3,@RW3	MOV R3, @RW3+d16	MOV R4,@RW3	MOV R4, @RW3+d16	MOV R5,@RW3	MOV R5, @RW3+d16	MOV R6,@RW3	MOV R6, @RW3+d16	MOV R7,@RW3	MOV R7, @RW3+d16
+ C	MOV R0, @RW0+	MOV R0, @RW0+RW7	MOV R1, @RW0+	MOV R1, @RW0+RW7	MOV R2, @RW0+	MOV R2, @RW0+RW7	MOV R3, @RW0+	MOV R3, @RW0+RW7	MOV R4, @RW0+	MOV R4, @RW0+RW7	MOV R5, @RW0+	MOV R5, @RW0+RW7	MOV R6, @RW0+	MOV R6, @RW0+RW7	MOV R7, @RW0+	MOV R7, @RW0+RW7
+ D	MOV R0, @RW1+	MOV R0, @RW1+RW7	MOV R1, @RW1+	MOV R1, @RW1+RW7	MOV R2, @RW1+	MOV R2, @RW1+RW7	MOV R3, @RW1+	MOV R3, @RW1+RW7	MOV R4, @RW1+	MOV R4, @RW1+RW7	MOV R5, @RW1+	MOV R5, @RW1+RW7	MOV R6, @RW1+	MOV R6, @RW1+RW7	MOV R7, @RW1+	MOV R7, @RW1+RW7
+ E	MOV R0, @RW2+	MOV R0, @PC+d16	MOV R1, @RW2+	MOV R1, @PC+d16	MOV R2, @RW2+	MOV R2, @PC+d16	MOV R3, @RW2+	MOV R3, @PC+d16	MOV R4, @RW2+	MOV R4, @PC+d16	MOV R5, @RW2+	MOV R5, @PC+d16	MOV R6, @RW2+	MOV R6, @PC+d16	MOV R7, @RW2+	MOV R7, @PC+d16
+ F	MOV R0, @RW3+	MOV R0, addr16	MOV R1, @RW3+	MOV R1, addr16	MOV R2, @RW3+	MOV R2, addr16	MOV R3, @RW3+	MOV R3, addr16	MOV R4, @RW3+	MOV R4, addr16	MOV R5, @RW3+	MOV R5, addr16	MOV R6, @RW3+	MOV R6, addr16	MOV R7, @RW3+	MOV R7, addr16

Table B.9-17 MOVW RWi, ea Instruction (first byte = $7B_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOVW RW0,RW0	MOVW RW0,@RW0+d8	MOVW RW1,RW0	MOVW RW1,@RW0+d8	MOVW RW2,RW0	MOVW RW2,@RW0+d8	MOVW RW3,RW0	MOVW RW3,@RW0+d8	MOVW RW4,RW0	MOVW RW4,@RW0+d8	MOVW RW5,RW0	MOVW RW5,@RW0+d8	MOVW RW6,RW0	MOVW RW6,@RW0+d8	MOVW RW7,RW0	MOVW RW7,@RW0+d8
+ 1	MOVW RW0,RW1	MOVW RW0,@RW1+d8	MOVW RW1,RW1	MOVW RW1,@RW1+d8	MOVW RW2,RW1	MOVW RW2,@RW1+d8	MOVW RW3,RW1	MOVW RW3,@RW1+d8	MOVW RW4,RW1	MOVW RW4,@RW1+d8	MOVW RW5,RW1	MOVW RW5,@RW1+d8	MOVW RW6,RW1	MOVW RW6,@RW1+d8	MOVW RW7,RW1	MOVW RW7,@RW1+d8
+ 2	MOVW RW0,RW2	MOVW RW0,@RW2+d8	MOVW RW1,RW2	MOVW RW1,@RW2+d8	MOVW RW2,RW2	MOVW RW2,@RW2+d8	MOVW RW3,RW2	MOVW RW3,@RW2+d8	MOVW RW4,RW2	MOVW RW4,@RW2+d8	MOVW RW5,RW2	MOVW RW5,@RW2+d8	MOVW RW6,RW2	MOVW RW6,@RW2+d8	MOVW RW7,RW2	MOVW RW7,@RW2+d8
+ 3	MOVW RW0,RW3	MOVW RW0,@RW3+d8	MOVW RW1,RW3	MOVW RW1,@RW3+d8	MOVW RW2,RW3	MOVW RW2,@RW3+d8	MOVW RW3,RW3	MOVW RW3,@RW3+d8	MOVW RW4,RW3	MOVW RW4,@RW3+d8	MOVW RW5,RW3	MOVW RW5,@RW3+d8	MOVW RW6,RW3	MOVW RW6,@RW3+d8	MOVW RW7,RW3	MOVW RW7,@RW3+d8
+ 4	MOVW RW0,RW4	MOVW RW0,@RW4+d8	MOVW RW1,RW4	MOVW RW1,@RW4+d8	MOVW RW2,RW4	MOVW RW2,@RW4+d8	MOVW RW3,RW4	MOVW RW3,@RW4+d8	MOVW RW4,RW4	MOVW RW4,@RW4+d8	MOVW RW5,RW4	MOVW RW5,@RW4+d8	MOVW RW6,RW4	MOVW RW6,@ RW4+d8	MOVW RW7,RW4	MOVW RW7,@RW4+d8
+ 5	MOVW RW0,RW5	MOVW RW0,@RW5+d8	MOVW RW1,RW5	MOVW RW1,@RW5+d8	MOVW RW2,RW5	MOVW RW2,@RW5+d8	MOVW RW3,RW5	MOVW RW3,@RW5+d8	MOVW RW4,RW5	MOVW RW4,@RW5+d8	MOVW RW5,RW5	MOVW RW5,@RW5+d8	MOVW RW6,RW5	MOVW RW6,@RW5+d8	MOVW RW7,RW5	MOVW RW7,@RW5+d8
+ 6	MOVW RW0,RW6	MOVW RW0,@RW6+d8	MOVW RW1,RW6	MOVW RW1,@RW6+d8	MOVW RW2,RW6	MOVW RW2,@RW6+d8	MOVW RW3,RW6	MOVW RW3,@RW6+d8	MOVW RW4,RW6	MOVW RW4,@RW6+d8	MOVW RW5,RW6	MOVW RW5,@RW6+d8	MOVW RW6,RW6	MOVW RW6,@RW6+d8	MOVW RW7,RW6	MOVW RW7,@RW6+d8
+ 7	MOVW RW0,RW7	MOVW RW0,@RW7+d8	MOVW RW1,RW7	MOVW RW1,@RW7+d8	MOVW RW2,RW7	MOVW RW2,@RW7+d8	MOVW RW3,RW7	MOVW RW3,@RW7+d8	MOVW RW4,RW7	MOVW RW4,@RW7+d8	MOVW RW5,RW7	MOVW RW5,@RW7+d8	MOVW RW6,RW7	MOVW RW6,@RW7+d8	MOVW RW7,RW7	MOVW RW7,@RW7+d8
+ 8	MOVW RW0,@RW0	MOVW RW0,@RW0+d16	MOVW RW1,@RW0	MOVW RW1,@RW0+d16	MOVW RW2,@RW0	MOVW RW2,@RW0+d16	MOVW RW3,@RW0	MOVW RW3,@RW0+d16	MOVW RW4,@RW0	MOVW RW4,@RW0+d16	MOVW RW5,@RW0	MOVW RW5,@RW0+d16	MOVW RW6,@RW0	MOVW RW6,@RW0+d16	MOVW RW7,@RW0	MOVW RW7,@RW0+d16
+ 9	MOVW RW0,@RW1	MOVW RW0,@RW1+d16	MOVW RW1,@RW1	MOVW RW1,@RW1+d16	MOVW RW2,@RW1	MOVW RW2,@RW1+d16	MOVW RW3,@RW1	MOVW RW3,@RW1+d16	MOVW RW4,@RW1	MOVW RW4,@RW1+d16	MOVW RW5,@RW1	MOVW RW5,@RW1+d16	MOVW RW6,@RW1	MOVW RW6,@RW1+d16	MOVW RW7,@RW1	MOVW RW7,@RW1+d16
+ A	MOVW RW0,@RW2	MOVW RW0,@RW2+d16	MOVW RW1,@RW2	MOVW RW1,@RW2+d16	MOVW RW2,@RW2	MOVW RW2,@RW2+d16	MOVW RW3,@RW2	MOVW RW3,@RW2+d16	MOVW RW4,@RW2	MOVW RW4,@RW2+d16	MOVW RW5,@RW2	MOVW RW5,@RW2+d16	MOVW RW6,@RW2	MOVW RW6,@RW2+d16	MOVW RW7,@RW2	MOVW RW7,@RW2+d16
+ B	MOVW RW0,@RW3	MOVW RW0,@RW3+d16	MOVW RW1,@RW3	MOVW RW1,@RW3+d16	MOVW RW2,@RW3	MOVW RW2,@RW3+d16	MOVW RW3,@RW3	MOVW RW3,@RW3+d16	MOVW RW4,@RW3	MOVW RW4,@RW3+d16	MOVW RW5,@RW3	MOVW RW5,@RW3+d16	MOVW RW6,@RW3	MOVW RW6,@RW3+d16	MOVW RW7,@RW3	MOVW RW7,@RW3+d16
+ C	MOVW R W0,@RW0+	MOVW RW0,@RW0+RW7	MOVW R W1,@RW0+	MOVW RW1,@RW0+RW7	MOVW R W2,@RW0+	MOVW RW2,@RW0+RW7	MOVW R W3,@RW0+	MOVW RW3,@RW0+RW7	MOVW R W4,@RW0+	MOVW RW4,@RW0+RW7	MOVW R W5,@RW0+	MOVW RW5,@RW0+RW7	MOVW R W6,@RW0+	MOVW RW6,@RW0+RW7	MOVW R W7,@RW0+	MOVW RW7,@RW0+RW7
+ D	MOVW R W0,@RW1+	MOVW RW0,@RW1+RW7	MOVW R W1,@RW1+	MOVW RW1,@RW1+RW7	MOVW R W2,@RW1+	MOVW RW2,@RW1+RW7	MOVW R W3,@RW1+	MOVW RW3,@RW1+RW7	MOVW R W4,@RW1+	MOVW RW4,@RW1+RW7	MOVW R W5,@RW1+	MOVW RW5,@RW1+RW7	MOVW R W6,@RW1+	MOVW RW6,@RW1+RW7	MOVW R W7,@RW1+	MOVW RW7,@RW1+RW7
+ E	MOVW R W0,@RW2+	MOVW RW0,@PC+d16	MOVW R W1,@RW2+	MOVW RW1,@PC+d16	MOVW R W2,@RW2+	MOVW RW2,@PC+d16	MOVW R W3,@RW2+	MOVW RW3,@PC+d16	MOVW R W4,@RW2+	MOVW RW4,@PC+d16	MOVW R W5,@RW2+	MOVW RW5,@PC+d16	MOVW R W6,@RW2+	MOVW RW6,@PC+d16	MOVW R W7,@RW2+	MOVW RW7,@PC+d16
+ F	MOVW R W0,@RW3+	MOVW RW 0,addr16	MOVW R W1,@RW3+	MOVW RW 1,addr16	MOVW R W2,@RW3+	MOVW RW 2,addr16	MOVW R W3,@RW3+	MOVW RW 3,addr16	MOVW R W4,@RW3+	MOVW RW 4,addr16	MOVW R W5,@RW3+	MOVW RW 5,addr16	MOVW R W6,@RW3+	MOVW RW 6,addr16	MOVW R W7,@RW3+	MOVW RW 7,addr16

Table B.9-18 MOV ea, Ri Instruction (first byte = $7C_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOV R0,R0	MOV @R W0+d8,R0	MOV R0,R1	MOV @R W0+d8,R1	MOV R0,R2	MOV @R W0+d8,R2	MOV R0,R3	MOV @R W0+d8,R3	MOV R0,R4	MOV @R W0+d8,R4	MOV R0,R5	MOV @R W0+d8,R5	MOV R0,R6	MOV @R W0+d8,R6	MOV R0,R7	MOV @R W0+d8,R7
+ 1	MOV R1,R0	MOV @R W1+d8,R0	MOV R1,R1	MOV @R W1+d8,R1	MOV R1,R2	MOV @R W1+d8,R2	MOV R1,R3	MOV @R W1+d8,R3	MOV R1,R4	MOV @R W1+d8,R4	MOV R1,R5	MOV @R W1+d8,R5	MOV R1,R6	MOV @R W1+d8,R6	MOV R1,R7	MOV @R W1+d8,R7
+ 2	MOV R2,R0	MOV @R W2+d8,R0	MOV R2,R1	MOV @R W2+d8,R1	MOV R2,R2	MOV @R W2+d8,R2	MOV R2,R3	MOV @R W2+d8,R3	MOV R2,R4	MOV @R W2+d8,R4	MOV R2,R5	MOV @R W2+d8,R5	MOV R2,R6	MOV @R W2+d8,R6	MOV R2,R7	MOV @R W2+d8,R7
+ 3	MOV R3,R0	MOV @R W3+d8,R0	MOV R3,R1	MOV @R W3+d8,R1	MOV R3,R2	MOV @R W3+d8,R2	MOV R3,R3	MOV @R W3+d8,R3	MOV R3,R4	MOV @R W3+d8,R4	MOV R3,R5	MOV @R W3+d8,R5	MOV R3,R6	MOV @R W3+d8,R6	MOV R3,R7	MOV @R W3+d8,R7
+ 4	MOV R4,R0	MOV @R W4+d8,R0	MOV R4,R1	MOV @R W4+d8,R1	MOV R4,R2	MOV @R W4+d8,R2	MOV R4,R3	MOV @R W4+d8,R3	MOV R4,R4	MOV @R W4+d8,R4	MOV R4,R5	MOV @R W4+d8,R5	MOV R4,R6	MOV @R W4+d8,R6	MOV R4,R7	MOV @R W4+d8,R7
+ 5	MOV R5,R0	MOV @R W5+d8,R0	MOV R5,R1	MOV @R W5+d8,R1	MOV R5,R2	MOV @R W5+d8,R2	MOV R5,R3	MOV @R W5+d8,R3	MOV R5,R4	MOV @R W5+d8,R4	MOV R5,R5	MOV @R W5+d8,R5	MOV R5,R6	MOV @R W5+d8,R6	MOV R5,R7	MOV @R W5+d8,R7
+ 6	MOV R6,R0	MOV @R W6+d8,R0	MOV R6,R1	MOV @R W6+d8,R1	MOV R6,R2	MOV @R W6+d8,R2	MOV R6,R3	MOV @R W6+d8,R3	MOV R6,R4	MOV @R W6+d8,R4	MOV R6,R5	MOV @R W6+d8,R5	MOV R6,R6	MOV @R W6+d8,R6	MOV R6,R7	MOV @R W6+d8,R7
+ 7	MOV R7,R0	MOV @R W7+d8,R0	MOV R7,R1	MOV @R W7+d8,R1	MOV R7,R2	MOV @R W7+d8,R2	MOV R7,R3	MOV @R W7+d8,R3	MOV R7,R4	MOV @R W7+d8,R4	MOV R7,R5	MOV @R W7+d8,R5	MOV R7,R6	MOV @R W7+d8,R6	MOV R7,R7	MOV @R W7+d8,R7
+ 8	MOV @RW0,R0	MOV @RW 0+d16,R0	MOV @RW0,R1	MOV @RW 0+d16,R1	MOV @RW0,R2	MOV @RW 0+d16,R2	MOV @RW0,R3	MOV @RW 0+d16,R3	MOV @RW0,R4	MOV @RW 0+d16,R4	MOV @RW0,R5	MOV @RW 0+d16,R5	MOV @RW0,R6	MOV @RW 0+d16,R6	MOV @RW0,R7	MOV @RW 0+d16,R7
+ 9	MOV @RW1,R0	MOV @RW 1+d16,R0	MOV @RW1,R1	MOV @RW 1+d16,R1	MOV @RW1,R2	MOV @RW 1+d16,R2	MOV @RW1,R3	MOV @RW 1+d16,R3	MOV @RW1,R4	MOV @RW 1+d16,R4	MOV @RW1,R5	MOV @RW 1+d16,R5	MOV @RW1,R6	MOV @RW 1+d16,R6	MOV @RW1,R7	MOV @RW 1+d16,R7
+ A	MOV @RW2,R0	MOV @RW 2+d16,R0	MOV @RW2,R1	MOV @RW 2+d16,R1	MOV @RW2,R2	MOV @RW 2+d16,R2	MOV @RW2,R3	MOV @RW 2+d16,R3	MOV @RW2,R4	MOV @RW 2+d16,R4	MOV @RW2,R5	MOV @RW 2+d16,R5	MOV @RW2,R6	MOV @RW 2+d16,R6	MOV @RW2,R7	MOV @RW 2+d16,R7
+ B	MOV @RW3,R0	MOV @RW 3+d16,R0	MOV @RW3,R1	MOV @RW 3+d16,R1	MOV @RW3,R2	MOV @RW 3+d16,R2	MOV @RW3,R3	MOV @RW 3+d16,R3	MOV @RW3,R4	MOV @RW 3+d16,R4	MOV @RW3,R5	MOV @RW 3+d16,R5	MOV @RW3,R6	MOV @RW 3+d16,R6	MOV @RW3,R7	MOV @RW 3+d16,R7
+ C	MOV @RW0+,R0	MOV @RW 0+RW7,R0	MOV @RW0+,R1	MOV @RW 0+RW7,R1	MOV @RW0+,R2	MOV @RW 0+RW7,R2	MOV @RW0+,R3	MOV @RW 0+RW7,R3	MOV @RW0+,R4	MOV @RW 0+RW7,R4	MOV @RW0+,R5	MOV @RW 0+RW7,R5	MOV @RW0+,R6	MOV @RW 0+RW7,R6	MOV @RW0+,R7	MOV @RW 0+RW7,R7
+ D	MOV @RW1+,R0	MOV @RW 1+RW7,R0	MOV @RW1+,R1	MOV @RW 1+RW7,R1	MOV @RW1+,R2	MOV @RW 1+RW7,R2	MOV @RW1+,R3	MOV @RW 1+RW7,R3	MOV @RW1+,R4	MOV @RW 1+RW7,R4	MOV @RW1+,R5	MOV @RW 1+RW7,R5	MOV @RW1+,R6	MOV @RW 1+RW7,R6	MOV @RW1+,R7	MOV @RW 1+RW7,R7
+ E	MOV @RW2+,R0	MOV P C+d16,R0	MOV @RW2+,R1	MOV P C+d16,R1	MOV @RW2+,R2	MOV P C+d16,R2	MOV @RW2+,R3	MOV P C+d16,R3	MOV @RW2+,R4	MOV P C+d16,R4	MOV @RW2+,R5	MOV P C+d16,R5	MOV @RW2+,R6	MOV P C+d16,R6	MOV @RW2+,R7	MOV P C+d16,R7
+ F	MOV @RW3+,R0	MOV a ddr16,R0	MOV @RW3+,R1	MOV a ddr16,R1	MOV @RW3+,R2	MOV a ddr16,R2	MOV @RW3+,R3	MOV a ddr16,R3	MOV @RW3+,R4	MOV a ddr16,R4	MOV @RW3+,R5	MOV a ddr16,R5	MOV @RW3+,R6	MOV a ddr16,R6	MOV @RW3+,R7	MOV a ddr16,R7

Table B.9-19 MOVW ea, Rwi Instruction (first byte = $7D_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOVW RW0,RW0	MOVW @RW 0+d8,RW0	MOVW RW0,RW1	MOVW @RW 0+d8,RW1	MOVW RW0,RW2	MOVW @RW 0+d8,RW2	MOVW RW0,RW3	MOVW @RW 0+d8,RW3	MOVW RW0,RW4	MOVW @RW 0+d8,RW4	MOVW RW0,RW5	MOVW @RW 0+d8,RW5	MOVW RW0,RW6	MOVW @RW 0+d8,RW6	MOVW RW0,RW7	MOVW @RW 0+d8,RW7
+ 1	MOVW RW1,RW0	MOVW @RW 1+d8,RW0	MOVW RW1,RW1	MOVW @RW 1+d8,RW1	MOVW RW1,RW2	MOVW @RW 1+d8,RW2	MOVW RW1,RW3	MOVW @RW 1+d8,RW3	MOVW RW1,RW4	MOVW @RW 1+d8,RW4	MOVW RW1,RW5	MOVW @RW 1+d8,RW5	MOVW RW1,RW6	MOVW @RW 1+d8,RW6	MOVW RW1,RW7	MOVW @RW 1+d8,RW7
+ 2	MOVW RW2,RW0	MOVW @RW 2+d8,RW0	MOVW RW2,RW1	MOVW @RW 2+d8,RW1	MOVW RW2,RW2	MOVW @RW 2+d8,RW2	MOVW RW2,RW3	MOVW @RW 2+d8,RW3	MOVW RW2,RW4	MOVW @RW 2+d8,RW4	MOVW RW2,RW5	MOVW @RW 2+d8,RW5	MOVW RW2,RW6	MOVW @RW 2+d8,RW6	MOVW RW2,RW7	MOVW @RW 2+d8,RW7
+ 3	MOVW RW3,RW0	MOVW @RW 3+d8,RW0	MOVW RW3,RW1	MOVW @RW 3+d8,RW1	MOVW RW3,RW2	MOVW @RW 3+d8,RW2	MOVW RW3,RW3	MOVW @RW 3+d8,RW3	MOVW RW3,RW4	MOVW @RW 3+d8,RW4	MOVW RW3,RW5	MOVW @RW 3+d8,RW5	MOVW RW3,RW6	MOVW @RW 3+d8,RW6	MOVW RW3,RW7	MOVW @RW 3+d8,RW7
+ 4	MOVW RW4,RW0	MOVW @RW 4+d8,RW0	MOVW RW4,RW1	MOVW @RW 4+d8,RW1	MOVW RW4,RW2	MOVW @RW 4+d8,RW2	MOVW RW4,RW3	MOVW @RW 4+d8,RW3	MOVW RW4,RW4	MOVW @RW 4+d8,RW4	MOVW RW4,RW5	MOVW @RW 4+d8,RW5	MOVW RW4,RW6	MOVW @RW 4+d8,RW6	MOVW RW4,RW7	MOVW @RW 4+d8,RW7
+ 5	MOVW RW5,RW0	MOVW @RW 5+d8,RW0	MOVW RW5,RW1	MOVW @RW 5+d8,RW1	MOVW RW5,RW2	MOVW @RW 5+d8,RW2	MOVW RW5,RW3	MOVW @RW 5+d8,RW3	MOVW RW5,RW4	MOVW @RW 5+d8,RW4	MOVW RW5,RW5	MOVW @RW 5+d8,RW5	MOVW RW5,RW6	MOVW @RW 5+d8,RW6	MOVW RW5,RW7	MOVW @RW 5+d8,RW7
+ 6	MOVW RW6,RW0	MOVW @RW 6+d8,RW0	MOVW RW6,RW1	MOVW @RW 6+d8,RW1	MOVW RW6,RW2	MOVW @RW 6+d8,RW2	MOVW RW6,RW3	MOVW @RW 6+d8,RW3	MOVW RW6,RW4	MOVW @RW 6+d8,RW4	MOVW RW6,RW5	MOVW @RW 6+d8,RW5	MOVW RW6,RW6	MOVW @RW 6+d8,RW6	MOVW RW6,RW7	MOVW @RW 6+d8,RW7
+ 7	MOVW RW7,RW0	MOVW @RW 7+d8,RW0	MOVW RW7,RW1	MOVW @RW 7+d8,RW1	MOVW RW7,RW2	MOVW @RW 7+d8,RW2	MOVW RW7,RW3	MOVW @RW 7+d8,RW3	WOVW RW7,RW4	MOVW @RW 7+d8,RW4	WOVW RW7,RW5	MOVW @RW 7+d8,RW5	WOVW RW7,RW6	MOVW @RW 7+d8,RW6	WOVW RW7,RW7	MOVW @RW 7+d8,RW7
+ 8	MOVW @RW0,RW0	MOVW @RW0 +d16,RW0	MOVW @RW0,RW1	MOVW @RW0 +d16,RW1	MOVW @RW0,RW2	MOVW @RW0 +d16,RW2	MOVW @RW0,RW3	MOVW @RW0 +d16,RW3	MOVW @RW0,RW4	MOVW @RW0 +d16,RW4	MOVW @RW0,RW5	MOVW @RW0 +d16,RW5	MOVW @RW0,RW6	MOVW @RW0 +d16,RW6	MOVW @RW0,RW7	MOVW @RW0 +d16,RW7
+ 9	MOVW @RW1,RW0	MOVW @RW1 +d16,RW0	MOVW @RW1,RW1	MOVW @RW1 +d16,RW1	MOVW @RW1,RW2	MOVW @RW1 +d16,RW2	MOVW @RW1,RW3	MOVW @RW1 +d16,RW3	MOVW @RW1,RW4	MOVW @RW1 +d16,RW4	MOVW @RW1,RW5	MOVW @RW1 +d16,RW5	MOVW @RW1,RW6	MOVW @RW1 +d16,RW6	MOVW @RW1,RW7	MOVW @RW1 +d16,RW7
+ A	MOVW @RW2,RW0	MOVW @RW2 +d16,RW0	MOVW @RW2,RW1	MOVW @RW2 +d16,RW1	MOVW @RW2,RW2	MOVW @RW2 +d16,RW2	MOVW @RW2,RW3	MOVW @RW2 +d16,RW3	MOVW @RW2,RW4	MOVW @RW2 +d16,RW4	MOVW @RW2,RW5	MOVW @RW2 +d16,RW5	MOVW @RW2,RW6	MOVW @RW2 +d16,RW6	MOVW @RW2,RW7	MOVW @RW2 +d16,RW7
+ B	MOVW @RW3,RW0	MOVW @RW3 +d16,RW0	MOVW @RW3,RW1	MOVW @RW3 +d16,RW1	MOVW @RW3,RW2	MOVW @RW3 +d16,RW2	MOVW @RW3,RW3	MOVW @RW3 +d16,RW3	MOVW @RW3,RW4	MOVW @RW3 +d16,RW4	MOVW @RW3,RW5	MOVW @RW3 +d16,RW5	MOVW @RW3,RW6	MOVW @RW3 +d16,RW6	MOVW @RW3,RW7	MOVW @RW3 +d16,RW7
+ C	MOVW @ RW0+,RW0	MOVW @RW0 +RW7,RW0	MOVW @ RW0+,RW1	MOVW @RW0 +RW7,RW1	MOVW @ RW0+,RW2	MOVW @RW0 +RW7,RW2	MOVW @ RW0+,RW3	MOVW @RW0 +RW7,RW3	MOVW @ RW0+,RW4	MOVW @RW0 +RW7,RW4	MOVW @ RW0+,RW5	MOVW @RW0 +RW7,RW5	MOVW @ RW0+,RW6	MOVW @RW0 +RW7,RW6	MOVW @ RW0+,RW7	MOVW @RW0 +RW7,RW7
+ D	MOVW @ RW1+,RW0	MOVW @RW1 +RW7,RW0	MOVW @ RW1+,RW1	MOVW @RW1 +RW7,RW1	MOVW @ RW1+,RW2	MOVW @RW1 +RW7,RW2	MOVW @ RW1+,RW3	MOVW @RW1 +RW7,RW3	MOVW @ RW1+,RW4	MOVW @RW1 +RW7,RW4	MOVW @ RW1+,RW5	MOVW @RW1 +RW7,RW5	MOVW @ RW1+,RW6	MOVW @RW1 +RW7,RW6	MOVW @ RW1+,RW7	MOVW @RW1 +RW7,RW7
+ E	MOVW @ RW2+,RW0	MOVW @PC+ d16,RW0	MOVW @ RW2+,RW1	MOVW @PC+ d16,RW1	MOVW @ RW2+,RW2	MOVW @PC+ d16,RW2	MOVW @ RW2+,RW3	MOVW @PC+ d16,RW3	MOVW @ RW2+,RW4	MOVW @PC+ d16,RW4	MOVW @ RW2+,RW5	MOVW @PC+ d16,RW5	MOVW @ RW2+,RW6	MOVW @PC+ d16,RW6	MOVW @ RW2+,RW7	MOVW @PC+ d16,RW7
+ F	MOVW @ RW3+,RW0	MOVW addr 16,RW0	MOVW @ RW3+,RW1	MOVW addr 16,RW1	MOVW @ RW3+,RW2	MOVW addr 16,RW2	MOVW @ RW3+,RW3	MOVW addr 16,RW3	MOVW @ RW3+,RW4	MOVW addr 16,RW4	MOVW @ RW3+,RW5	MOVW addr 16,RW5	MOVW @ RW3+,RW6	MOVW addr 16,RW6	MOVW @ RW3+,RW7	MOVW addr 16,RW7

Table B.9-20 XCH Ri, ea Instruction (first byte = $7E_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	XCH R0, R0,R0	XCH R0, @RW0+d8	XCH R0, R1,R0	XCH R1, @RW0+d8	XCH R2, R2,R0	XCH R2, @RW0+d8	XCH R3, R3,R0	XCH R3, @RW0+d8	XCH R4, R4,R0	XCH R4, @RW0+d8	XCH R5, R5,R0	XCH R5, @RW0+d8	XCH R6, R6,R0	XCH R6, @RW0+d8	XCH R7, R7,R0	XCH R7, @RW0+d8
+ 1	XCH R0, R0,R1	XCH R0, @RW1+d8	XCH R0, R1,R1	XCH R1, @RW1+d8	XCH R2, R2,R1	XCH R2, @RW1+d8	XCH R3, R3,R1	XCH R3, @RW1+d8	XCH R4, R4,R1	XCH R4, @RW1+d8	XCH R5, R5,R1	XCH R5, @RW1+d8	XCH R6, R6,R1	XCH R6, @RW1+d8	XCH R7, R7,R1	NOTW R7, @RW1+d8
+ 2	XCH R0, R0,R2	XCH R0, @RW2+d8	XCH R0, R1,R2	XCH R1, @RW2+d8	XCH R2, R2,R2	XCH R2, @RW2+d8	XCH R3, R3,R2	XCH R3, @RW2+d8	XCH R4, R4,R2	XCH R4, @RW2+d8	XCH R5, R5,R2	XCH R5, @RW2+d8	XCH R6, R6,R2	XCH R6, @RW2+d8	XCH R7, R7,R2	XCH R7, @RW2+d8
+ 3	XCH R0, R0,R3	XCH R0, @RW3+d8	XCH R0, R1,R3	XCH R1, @RW3+d8	XCH R2, R2,R3	XCH R2, @RW3+d8	XCH R3, R3,R3	XCH R3, @RW3+d8	XCH R4, R4,R3	XCH R4, @RW3+d8	XCH R5, R5,R3	XCH R5, @RW3+d8	XCH R6, R6,R3	XCH R6, @RW3+d8	XCH R7, R7,R3	NOTW R7, @RW3+d8
+ 4	XCH R0, R0,R4	XCH R0, @RW4+d8	XCH R0, R1,R4	XCH R1, @RW4+d8	XCH R2, R2,R4	XCH R2, @RW4+d8	XCH R3, R3,R4	XCH R3, @RW4+d8	XCH R4, R4,R4	XCH R4, @RW4+d8	XCH R5, R5,R4	XCH R5, @RW4+d8	XCH R6, R6,R4	XCH R6, @RW4+d8	XCH R7, R7,R4	XCH R7, @RW4+d8
+ 5	XCH R0, R0,R5	XCH R0, @RW5+d8	XCH R0, R1,R5	XCH R1, @RW5+d8	XCH R2, R2,R5	XCH R2, @RW5+d8	XCH R3, R3,R5	XCH R3, @RW5+d8	XCH R4, R4,R5	XCH R4, @RW5+d8	XCH R5, R5,R5	XCH R5, @RW5+d8	XCH R6, R6,R5	XCH R6, @RW5+d8	XCH R7, R7,R5	XCH R7, @RW5+d8
+ 6	XCH R0, R0,R6	XCH R0, @RW6+d8	XCH R0, R1,R6	XCH R1, @RW6+d8	XCH R2, R2,R6	XCH R2, @RW6+d8	XCH R3, R3,R6	XCH R3, @RW6+d8	XCH R4, R4,R6	XCH R4, @RW6+d8	XCH R5, R5,R6	XCH R5, @RW6+d8	XCH R6, R6,R6	XCH R6, @RW6+d8	XCH R7, R7,R6	XCH R7, @RW6+d8
+ 7	XCH R0, R0,R7	XCH R0, @RW7+d8	XCH R0, R1,R7	XCH R1, @RW7+d8	XCH R2, R2,R7	XCH R2, @RW7+d8	XCH R3, R3,R7	XCH R3, @RW7+d8	XCH R4, R4,R7	XCH R4, @RW7+d8	XCH R5, R5,R7	XCH R5, @RW7+d8	XCH R6, R6,R7	XCH R6, @RW7+d8	XCH R7, R7,R7	XCH R7, @RW7+d8
+ 8	XCH R0, R0,@RW0	XCH R0, @RW0+d16	XCH R0, R1,@RW0	XCH R1, @RW0+d16	XCH R2, R2,@RW0	XCH R2, @RW0+d16	XCH R3, R3,@RW0	XCH R3, @RW0+d16	XCH R4, R4,@RW0	XCH R4, @RW0+d16	XCH R5, R5,@RW0	XCH R5, @RW0+d16	XCH R6, R6,@RW0	XCH R6, @RW0+d16	XCH R7, R7,@RW0	XCH R7, @RW0+d16
+ 9	XCH R0, R0,@RW1	XCH R0, @RW1+d16	XCH R0, R1,@RW1	XCH R1, @RW1+d16	XCH R2, R2,@RW1	XCH R2, @RW1+d16	XCH R3, R3,@RW1	XCH R3, @RW1+d16	XCH R4, R4,@RW1	XCH R4, @RW1+d16	XCH R5, R5,@RW1	XCH R5, @RW1+d16	XCH R6, R6,@RW1	XCH R6, @RW1+d16	XCH R7, R7,@RW1	XCH R7, @RW1+d16
+ A	XCH R0, R0,@RW2	XCH R0, W2+d16,A	XCH R0, R1,@RW2	XCH R1, W2+d16,A	XCH R2, R2,@RW2	XCH R2, W2+d16,A	XCH R3, R3,@RW2	XCH R3, W2+d16,A	XCH R4, R4,@RW2	XCH R4, W2+d16,A	XCH R5, R5,@RW2	XCH R5, W2+d16,A	XCH R6, R6,@RW2	XCH R6, W2+d16,A	XCH R7, R7,@RW2	XCH R7, W2+d16,A
+ B	XCH R0, R0,@RW3	XCH R0, @RW3+d16	XCH R0, R1,@RW3	XCH R1, @RW3+d16	XCH R2, R2,@RW3	XCH R2, @RW3+d16	XCH R3, R3,@RW3	XCH R3, @RW3+d16	XCH R4, R4,@RW3	XCH R4, @RW3+d16	XCH R5, R5,@RW3	XCH R5, @RW3+d16	XCH R6, R6,@RW3	XCH R6, @RW3+d16	XCH R7, R7,@RW3	XCH R7, @RW3+d16
+ C	XCH R0, R0,@RW0+	XCH R0, @RW0+RW7	XCH R0, R1,@RW0+	XCH R1, @RW0+RW7	XCH R2, R2,@RW0+	XCH R2, @RW0+RW7	XCH R3, R3,@RW0+	XCH R3, @RW0+RW7	XCH R4, R4,@RW0+	XCH R4, @RW0+RW7	XCH R5, R5,@RW0+	XCH R5, @RW0+RW7	XCH R6, R6,@RW0+	XCH R6, @RW0+RW7	XCH R7, R7,@RW0+	XCH R7, @RW0+RW7
+ D	XCH R0, R0,@RW1+	XCH R0, @RW1+RW7	XCH R0, R1,@RW1+	XCH R1, @RW1+RW7	XCH R2, R2,@RW1+	XCH R2, @RW1+RW7	XCH R3, R3,@RW1+	XCH R3, @RW1+RW7	XCH R4, R4,@RW1+	XCH R4, @RW1+RW7	XCH R5, R5,@RW1+	XCH R5, @RW1+RW7	XCH R6, R6,@RW1+	XCH R6, @RW1+RW7	XCH R7, R7,@RW1+	XCH R7, @RW1+RW7
+ E	XCH R0, R0,@RW2+	XCH R0, @PC+d16	XCH R0, R1,@RW2+	XCH R1, @PC+d16	XCH R2, R2,@RW2+	XCH R2, @PC+d16	XCH R3, R3,@RW2+	XCH R3, @PC+d16	XCH R4, R4,@RW2+	XCH R4, @PC+d16	XCH R5, R5,@RW2+	XCH R5, @PC+d16	XCH R6, R6,@RW2+	XCH R6, @PC+d16	XCH R7, R7,@RW2+	XCH R7, @PC+d16
+ F	XCH R0, R0,@RW3+	XCH R0, addr16	XCH R0, R1,@RW3+	XCH R1, addr16	XCH R2, R2,@RW3+	XCH R2, addr16	XCH R3, R3,@RW3+	XCH R3, addr16	XCH R4, R4,@RW3+	XCH R4, addr16	XCH R5, R5,@RW3+	XCH R5, addr16	XCH R6, R6,@RW3+	XCH R6, addr16	XCH R7, R7,@RW3+	XCH R7, addr16

Table B.9-21 XCHW RWi, ea Instruction (first byte = $7F_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	XCHW RW0, RW0,RW0	XCHW RW0, @RW0+d8	XCHW RW1,RW0	XCHW RW1, @RW0+d8	XCHW RW2,RW0	XCHW RW2, @RW0+d8	XCHW RW3,RW0	XCHW RW3, @RW0+d8	XCHW RW4,RW0	XCHW RW4, @RW0+d8	XCHW RW5,RW0	XCHW RW5, @RW0+d8	XCHW RW6,RW0	XCHW RW6, @RW0+d8	XCHW RW7,RW0	XCHW RW7, @RW0+d8
+ 1	XCHW RW0, RW0,RW1	XCHW RW0, @RW1+d8	XCHW RW1,RW1	XCHW RW1, @RW1+d8	XCHW RW2,RW1	XCHW RW2, @RW1+d8	XCHW RW3,RW1	XCHW RW3, @RW1+d8	XCHW RW4,RW1	XCHW RW4, @RW1+d8	XCHW RW5,RW1	XCHW RW5, @RW1+d8	XCHW RW6,RW1	XCHW RW6, @RW1+d8	XCHW RW7,RW1	XCHW RW7, @RW1+d8
+ 2	XCHW RW0,RW2	XCHW RW0, @RW2+d8	XCHW RW1,RW2	XCHW RW1, @RW2+d8	XCHW RW2,RW2	XCHW RW2, @RW2+d8	XCHW RW3,RW2	XCHW RW3, @RW2+d8	XCHW RW4,RW2	XCHW RW4, @RW2+d8	XCHW RW5,RW2	XCHW RW5, @RW2+d8	XCHW RW6,RW2	XCHW RW6, @RW2+d8	XCHW RW7,RW2	XCHW RW7, @RW2+d8
+ 3	XCHW RW0,RW3	XCHW RW0, @RW3+d8	XCHW RW1,RW3	XCHW RW1, @RW3+d8	XCHW RW2,RW3	XCHW RW2, @RW3+d8	XCHW RW3,RW3	XCHW RW3, @RW3+d8	XCHW RW4,RW3	XCHW RW4, @RW3+d8	XCHW RW5,RW3	XCHW RW5, @RW3+d8	XCHW RW6,RW3	XCHW RW6, @RW3+d8	XCHW RW7,RW3	XCHW RW7, @RW3+d8
+ 4	XCHW RW0,RW4	XCHW RW0, @RW4+d8	XCHW RW1,RW4	XCHW RW1, @RW4+d8	XCHW RW2,RW4	XCHW RW2, @RW4+d8	XCHW RW3,RW4	XCHW RW3, @RW4+d8	XCHW RW4,RW4	XCHW RW4, @RW4+d8	XCHW RW5,RW4	XCHW RW5, @RW4+d8	XCHW RW6,RW4	XCHW RW6, @RW4+d8	XCHW RW7,RW4	XCHW RW7, @RW4+d8
+ 5	XCHW RW0,RW5	XCHW RW0, @RW5+d8	XCHW RW1,RW5	XCHW RW1, @RW5+d8	XCHW RW2,RW5	XCHW RW2, @RW5+d8	XCHW RW3,RW5	XCHW RW3, @RW5+d8	XCHW RW4,RW5	XCHW RW4, @RW5+d8	XCHW RW5,RW5	XCHW RW5, @RW5+d8	XCHW RW6,RW5	XCHW RW6, @RW5+d8	XCHW RW7,RW5	XCHW RW7, @RW5+d8
+ 6	XCHW RW0,RW6	XCHW RW0, @RW6+d8	XCHW RW1,RW6	XCHW RW1, @RW6+d8	XCHW RW2,RW6	XCHW RW2, @RW6+d8	XCHW RW3,RW6	XCHW RW3, @RW6+d8	XCHW RW4,RW6	XCHW RW4, @RW6+d8	XCHW RW5,RW6	XCHW RW5, @RW6+d8	XCHW RW6,RW6	XCHW RW6, @RW6+d8	XCHW RW7,RW6	XCHW RW7, @RW6+d8
+ 7	XCHW RW0,RW7	XCHW RW0, @RW7+d8	XCHW RW1,RW7	XCHW RW1, @RW7+d8	XCHW RW2,RW7	XCHW RW2, @RW7+d8	XCHW RW3,RW7	XCHW RW3, @RW7+d8	XCHW RW4,RW7	XCHW RW4, @RW7+d8	XCHW RW5,RW7	XCHW RW5, @RW7+d8	XCHW RW6,RW7	XCHW RW6, @RW7+d8	XCHW RW7,RW7	XCHW RW7, @RW7+d8
+ 8	XCHW RW0,@RW0	XCHW RW0, @RW0+d16	XCHW RW1,@RW0	XCHW RW1, @RW0+d16	XCHW RW2,@RW0	XCHW RW2, @RW0+d16	XCHW RW3,@RW0	XCHW RW3, @RW0+d16	XCHW RW4,@RW0	XCHW RW4, @RW0+d16	XCHW RW5,@RW0	XCHW RW5, @RW0+d16	XCHW RW6,@RW0	XCHW RW6, @RW0+d16	XCHW RW7,@RW0	XCHW RW7, @RW0+d16
+ 9	XCHW RW0,@RW1	XCHW RW0, @RW1+d16	XCHW RW1,@RW1	XCHW RW1, @RW1+d16	XCHW RW2,@RW1	XCHW RW2, @RW1+d16	XCHW RW3,@RW1	XCHW RW3, @RW1+d16	XCHW RW4,@RW1	XCHW RW4, @RW1+d16	XCHW RW5,@RW1	XCHW RW5, @RW1+d16	XCHW RW6,@RW1	XCHW RW6, @RW1+d16	XCHW RW7,@RW1	XCHW RW7, @RW1+d16
+ A	XCHW RW0,@RW2	XCHW RW0, @RW2+d16	XCHW RW1,@RW2	XCHW RW1, @RW2+d16	XCHW RW2,@RW2	XCHW RW2, @RW2+d16	XCHW RW3,@RW2	XCHW RW3, @RW2+d16	XCHW RW4,@RW2	XCHW RW4, @RW2+d16	XCHW RW5,@RW2	XCHW RW5, @RW2+d16	XCHW RW6,@RW2	XCHW RW6, @RW2+d16	XCHW RW7,@RW2	XCHW RW7, @RW2+d16
+ B	XCHW RW0,@RW3	XCHW RW0, @RW3+d16	XCHW RW1,@RW3	XCHW RW1, @RW3+d16	XCHW RW2,@RW3	XCHW RW2, @RW3+d16	XCHW RW3,@RW3	XCHW RW3, @RW3+d16	XCHW RW4,@RW3	XCHW RW4, @RW3+d16	XCHW RW5,@RW3	XCHW RW5, @RW3+d16	XCHW RW6,@RW3	XCHW RW6, @RW3+d16	XCHW RW7,@RW3	XCHW RW7, @RW3+d16
+ C	XCHW R W0,@RW0+	XCHW RW0, @RW0+RW7	XCHW R W1,@RW0+	XCHW RW1, @RW0+RW7	XCHW R W2,@RW0+	XCHW RW2, @RW0+RW7	XCHW R W3,@RW0+	XCHW RW3, @RW0+RW7	XCHW R W4,@RW0+	XCHW RW4, @RW0+RW7	XCHW R W5,@RW0+	XCHW RW5, @RW0+RW7	XCHW R W6,@RW0+	XCHW RW6, @RW0+RW7	XCHW R W7,@RW0+	XCHW RW7, @RW0+RW7
+ D	XCHW R WO,@RW1+	XCHW RW0, @RW1+RW7	XCHW R W1,@RW1+	XCHW RW1, @RW1+RW7	XCHW R W2,@RW1+	XCHW RW2, @RW1+RW7	XCHW R W3,@RW1+	XCHW RW3, @RW1+RW7	XCHW R W4,@RW1+	XCHW RW4, @RW1+RW7	XCHW R W5,@RW1+	XCHW RW5, @RW1+RW7	XCHW R W6,@RW1+	XCHW RW6, @RW1+RW7	XCHW R W7,@RW1+	XCHW RW7, @RW1+RW7
+ E	XCHW R WO,@RW2+	XCHW RW0, @PC+d16	XCHW R W1,@RW2+	XCHW RW1, @PC+d16	XCHW R W2,@RW2+	XCHW RW2, @PC+d16	XCHW R W3,@RW2+	XCHW RW3, @PC+d16	XCHW R W4,@RW2+	XCHW RW4, @PC+d16	XCHW R W5,@RW2+	XCHW RW5, @PC+d16	XCHW R W6,@RW2+	XCHW RW6, @PC+d16	XCHW R W7,@RW2+	XCHW RW7, @PC+d16
+ F	XCHW R WO,@RW3+	XCHW RW0, addr16	XCHW R W1,@RW3+	XCHW RW1, addr16	XCHW R W2,@RW3+	XCHW RW2, addr16	XCHW R W3,@RW3+	XCHW RW3, addr16	XCHW R W4,@RW3+	XCHW RW4, addr16	XCHW R W5,@RW3+	XCHW RW5, addr16	XCHW R W6,@RW3+	XCHW RW6, addr16	XCHW R W7,@RW3+	XCHW RW7, addr16

INDEX

The index follows on the next page.
This is listed in alphabetic order.

Index

G

H

I

L

M

N

O

P

R

S

T

U

W

CM44-10107-4E

FUJITSU SEMICONDUCTOR •CONTORLLER MANUAL

F^2MC-16LX
16-BIT MICROCONTROLLER
MB90560/565 Series
HARDWARE MANUAL

May 2001 the fourth edition

Published **FUJITSU LIMITED** Electronic Devices

Edited Technical Information Dept.

FUJITSU

CM44-10107-4E