FUJITSU SEMICONDUCTOR
CONTROLLER MANUAL

CM44-10107-1E

F²MC-16LX

16-BIT MICROCONTROLLER

MB90560 Series

HARDWARE MANUAL

FUJITSU

PREFACE

■ Objectives and Intended Reader

Thank you for purchasing Fujitsu semiconductor products.

The MB90560 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 ICs (ASICs).

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

■ Trademarks

F²MC is a registered trademark of Fujitsu Limited and stands for FUJITSU Flexible Microcontroller.

1. The contents of this document are subject to change without notice. Customers are advised to consult with FUJITSU sales representatives before ordering.
2. 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.
3. The contents of this document may not be reproduced or copied without the permission of FUJITSU LIMITED.
4. FUJITSU semiconductor devices are intended for use in standard applications (computers, office automation and other office equipment, industrial, communications, and measurement equipment, personal or household devices, etc.).
   CAUTION:
   Customers considering the use of our products in special applications where failure or abnormal operation may directly affect human lives or cause physical injury or property damage, or where extremely high levels of reliability are demanded (such as aerospace systems, atomic energy controls, sea floor repeaters, vehicle operating controls, medical devices for life support, etc.) are requested to consult with FUJITSU sales representatives before such use. The company will not be responsible for damages arising from such use without prior approval.
5. 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.
6. 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

FIGURES

TABLES

CHAPTER 1 OVERVIEW

This chapter describes the features of the MB90560 series.

1.1 Features

The MB90560 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 FMC-8L and FMC-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 to be processed.

■ Features of the MB90560 Series

- Clock
  - Built-in PLL clock multiplying circuit
  - Selectable operating clock (PLL clock): The source oscillation can be divided by two or multiplied by 1 to 4 (4 to 16 MHz when the source oscillation is 4 MHz).
  - 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
- Optimum instruction set for controller applications
  - Many data types (bit, byte, word, and long word)
  - As many as 23 addressing modes
  - High code efficiency
  - 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 multitasking
  - System stack pointer
  - Instruction set symmetry and barrel shift instructions
- Sleep mode (The operating clock of the CPU stops.)
- Program patch function (Two-address pointer)
- Increased execution speed

  4-byte instruction queue
- Enhanced interrupt function (Eight programmable priority levels)

  Powerful interrupt function of 32 factors
- Data transfer function (Extended intelligent I/O service function using up to 16 channels)
- Lower-power consumption (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

● 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 pin pitch)

● Process : CMOS technology

■ Internal peripheral functions (resources)

● I/O ports: Up to 50 posts

● 18-bit 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
- 16-bit output compare: 6 channels

  When the count of the 16-bit free running timer matches the count set for comparison, the output compare unit inverts the output and generates an interrupt request.
- 16-bit input capture: 4 channels

  When the edge of the pin input is detected, the input capture unit latches the count of the 16-bit free running timer and generates an interrupt request.
- 8-/16-bit PPG timer: 8 bits x 6 channels or 16 bits x 3 channels

  Capable of changing the period or duty cycle of the output pulses as desired.
- Waveform generation circuit (8-bit timer: 3 channels)

  Capable of generating optimum waveforms for inverter control.

● UART: 2 channels

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

● DTP/external interrupt (8 channels)

Module for activating the extended intelligent I/O service by external input and for generating an external interrupt

● 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
- Capable of activation by external trigger

1.2 Product Lineup

Table 1.2-1 shows the product lineup of the MB90560 series. The specifications excluding the ROM and RAM capacities are common for the series.

■ **Product Lineup**

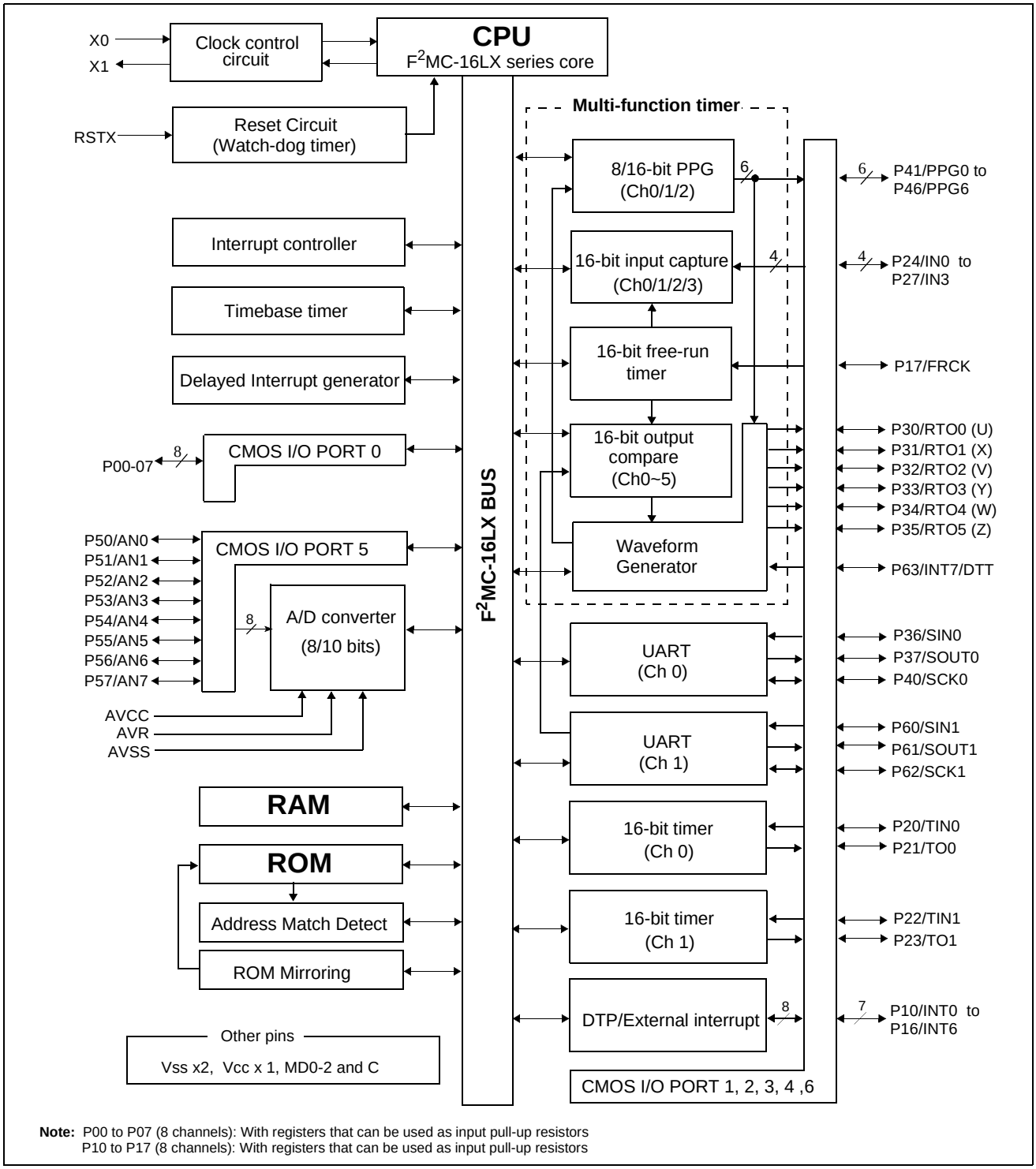
Table 1.2-1 Product lineup of the MB90560 series

Device	MB90V560	MB90F562	MB90561	MB90562
Type	Evaluation device	Flash Type ROM	Mass-production product (mask ROM)	
ROM size	—	64K Byte		32K Byte
RAM size	4K Byte	2K Byte		1K Byte
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 (CMS): 50			
UART	With a full duplex double buffer. Capable of synchronous or asynchronous clock transfer. Usable also for serial I/O. Dedicated built-in baud rate generator. Two channels built-in			
16-bit reload timer	16-bit reload timer operation (toggle output or one-shot selectable) Choice of the event count function. Two channels built-in.			
Advanced timer	16-bit free running timer x 1 channel, 16-bit output compare x 6 channels, 16-bit 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, dead time output			
10-bit A/D converter	10-bit resolution x 8 channels (input multiplex) Conversion time:6.13µ or less (operation at internal 16 MHz clock)			
External interrupt	Independent 8 channels Interrupt cause: L-to-H edge, H-to-L edge, L-level, or H-level selectable			
Low-power consumption mode	Sleep mode, stop mode, and CPU intermittent mode			
Process	CMOS			
Package	PGA256	QFP-64 (0.65 or 1.00 mm pitch), SHDIP-64		
Operating voltage	5V±10% @16MHz			

1.3 Block Diagram

Figure 1.3-1 shows the block diagram of the MB90560 series.

■ **Block Diagram**



Note: P00 to P07 (8 channels): With registers that can be used as input pull-up resistors
P10 to P17 (8 channels): With registers that can be used as input pull-up resistors

Figure 1.3-2 Block diagram

1.4 Pin Assignments

This section provides the pin assignments for the following three MB90560 series packages:

- **FPT-64P-M06**
- **FPT-64P-M09**
- **DIP-64P-M01**

■ Pin Assignment of FPT-64P-M06

Figure 1.4-1 shows the pin assignment of the FPT-64P-M06.

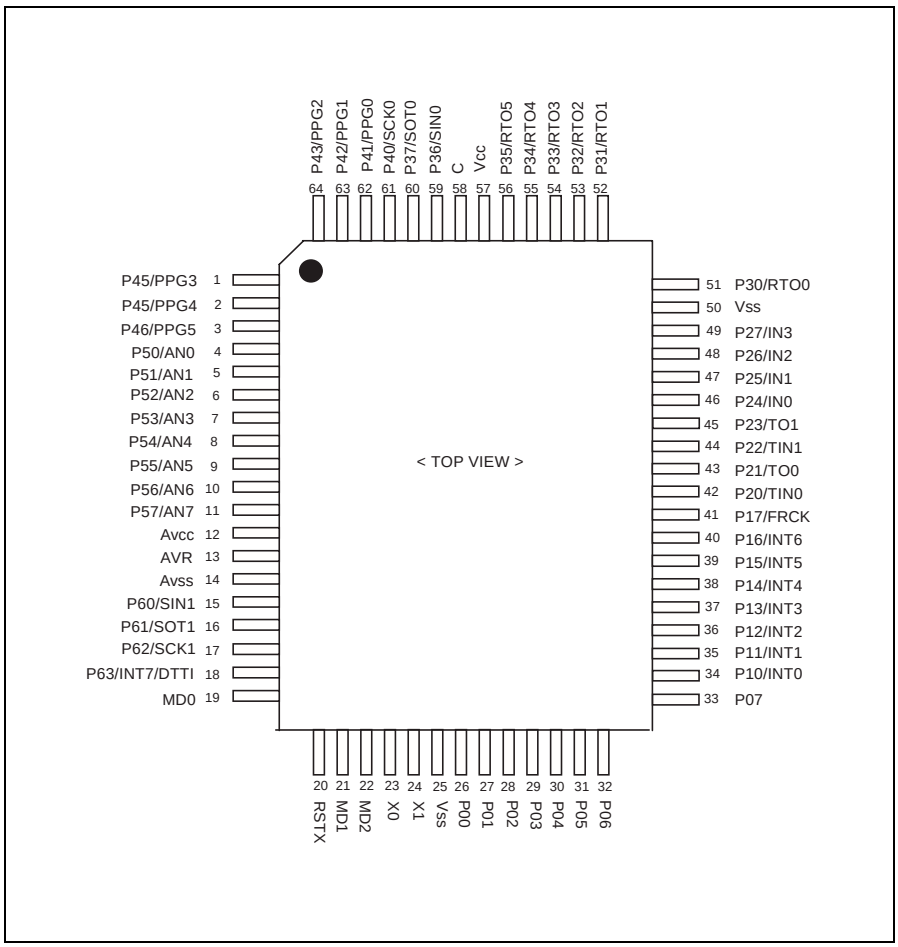


Figure 1.4-1 Pin assignment of FPT-64P-M09

■ Pin Assignment of FPT-64P-M09

Figure 1.4-2 shows the pin assignment of the FPT-64P-M09.

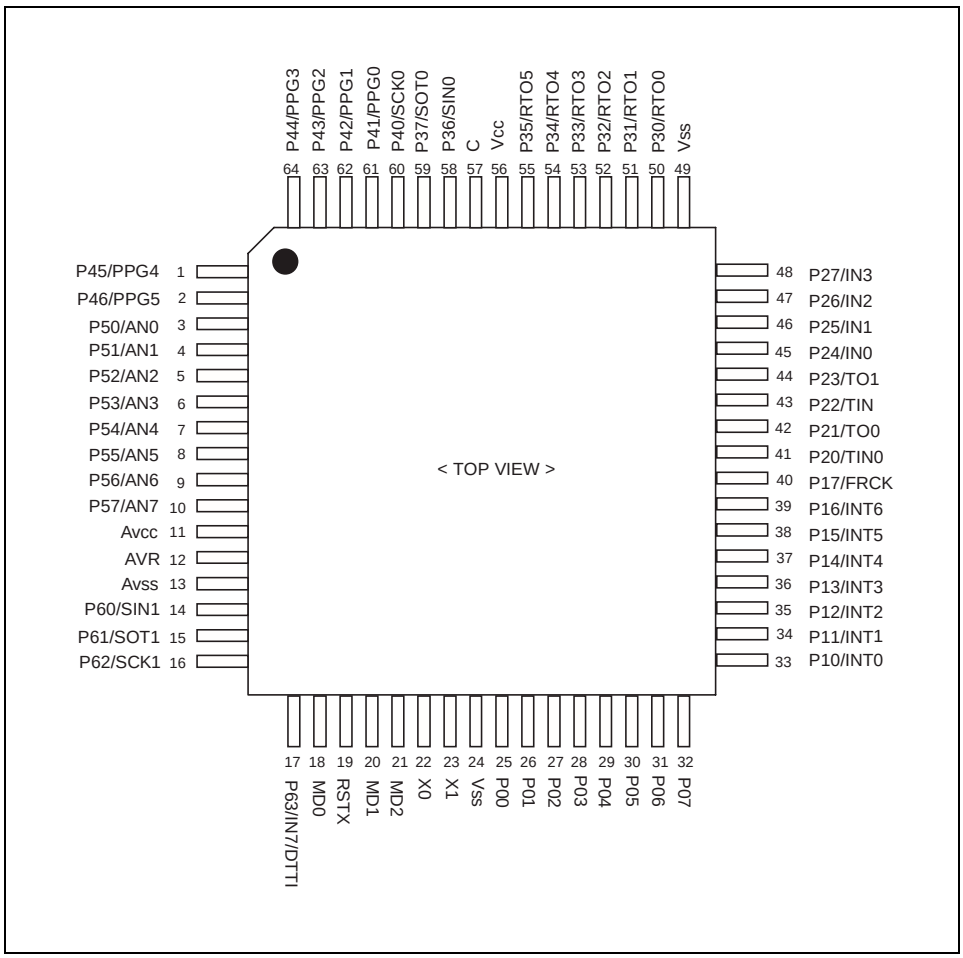


Figure 1.4-2 Pin assignment of FPT-64P-M09

■ Pin Assignment of DIP-64P-M01

Figure 1.4-3 shows the pin assignment of the DIP-64P-M01.

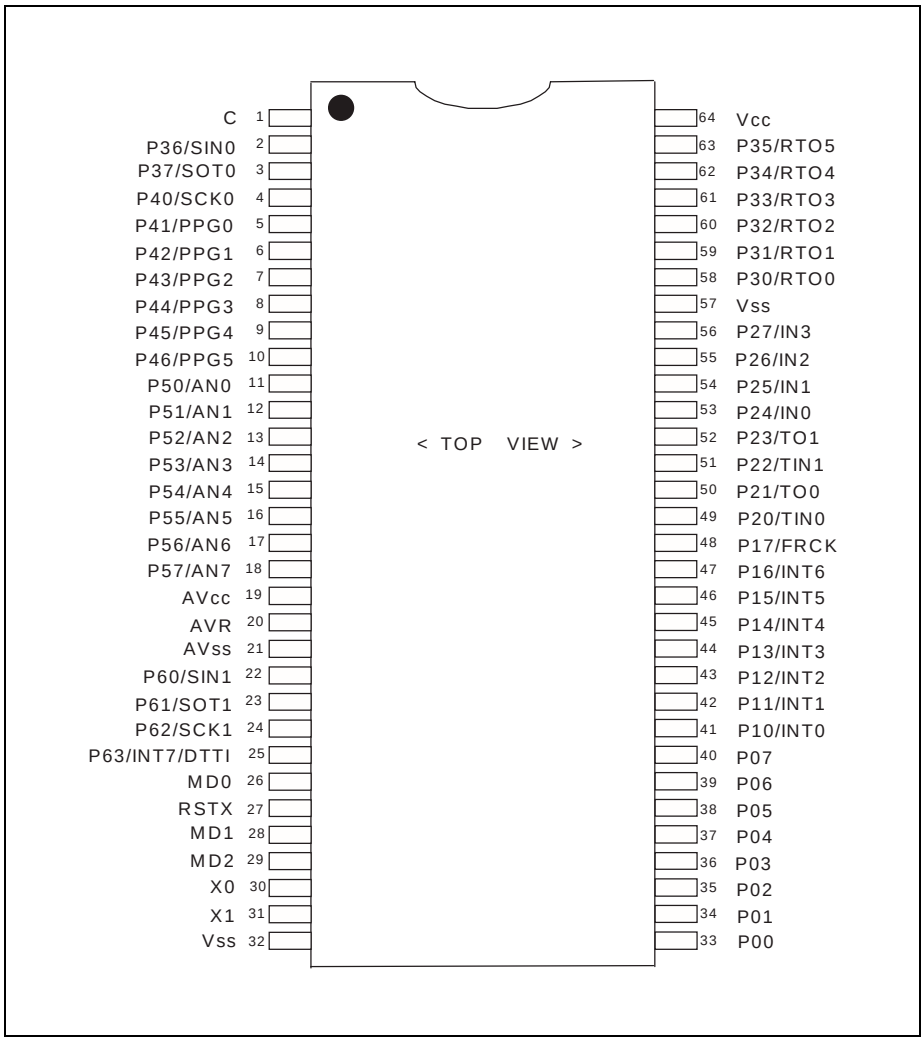


Figure 1.4-3 Pin assignment of DIP-64P-M01

Memo

1.5 Package Dimensions

This section provides the dimensions of the following three MB90560 series packages:

- **FPT-64P-M06**
- **FPT-64P-M09**
- **DIP-64P-M01**

■ FPT-64P-M06

Figure 1.5-1 shows the dimensions of the FPT-64P-M06 package.

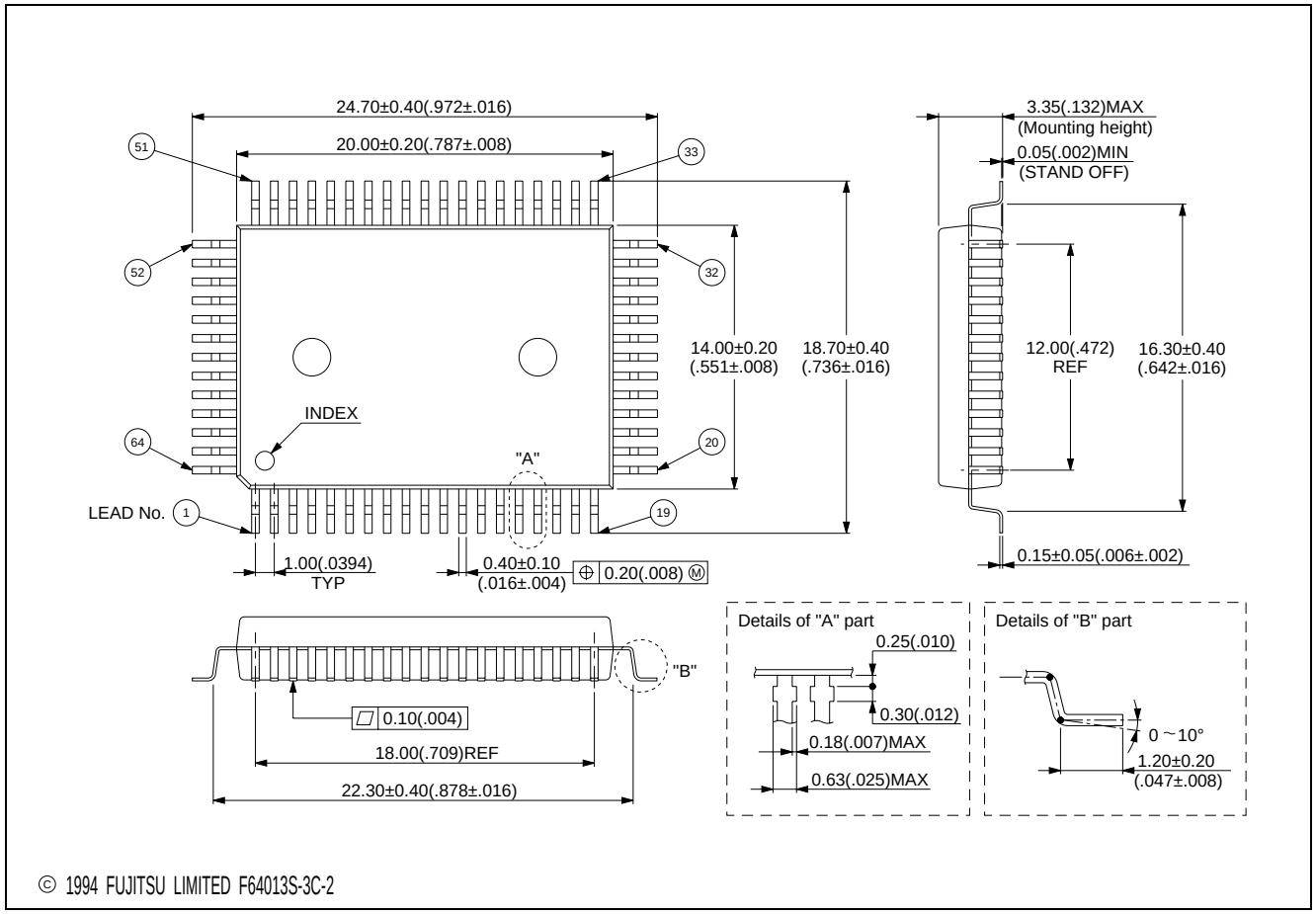


Figure 1.5-1 Dimensions of FPT-64P-M06 package

■ **FPT-64P-M09**

Figure 1.5-1 shows the dimensions of the FPT-64P-M09 package.

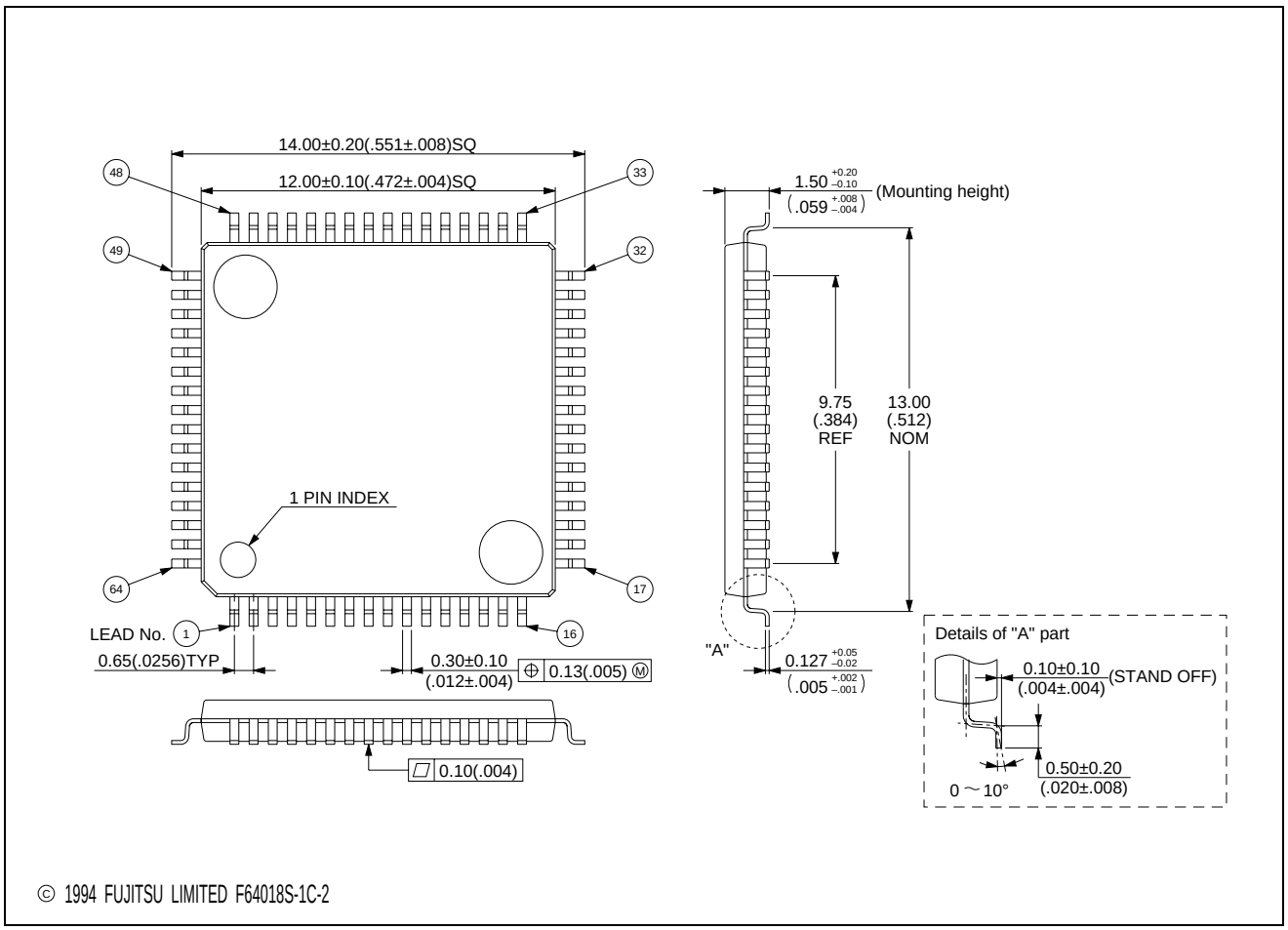


Figure 1.5-2 Dimensions of FPT-64P-M09 package

■ **DIP-64P-M01**

Figure 1.5-3 shows the dimensions of the DIP-64P-M01 package.

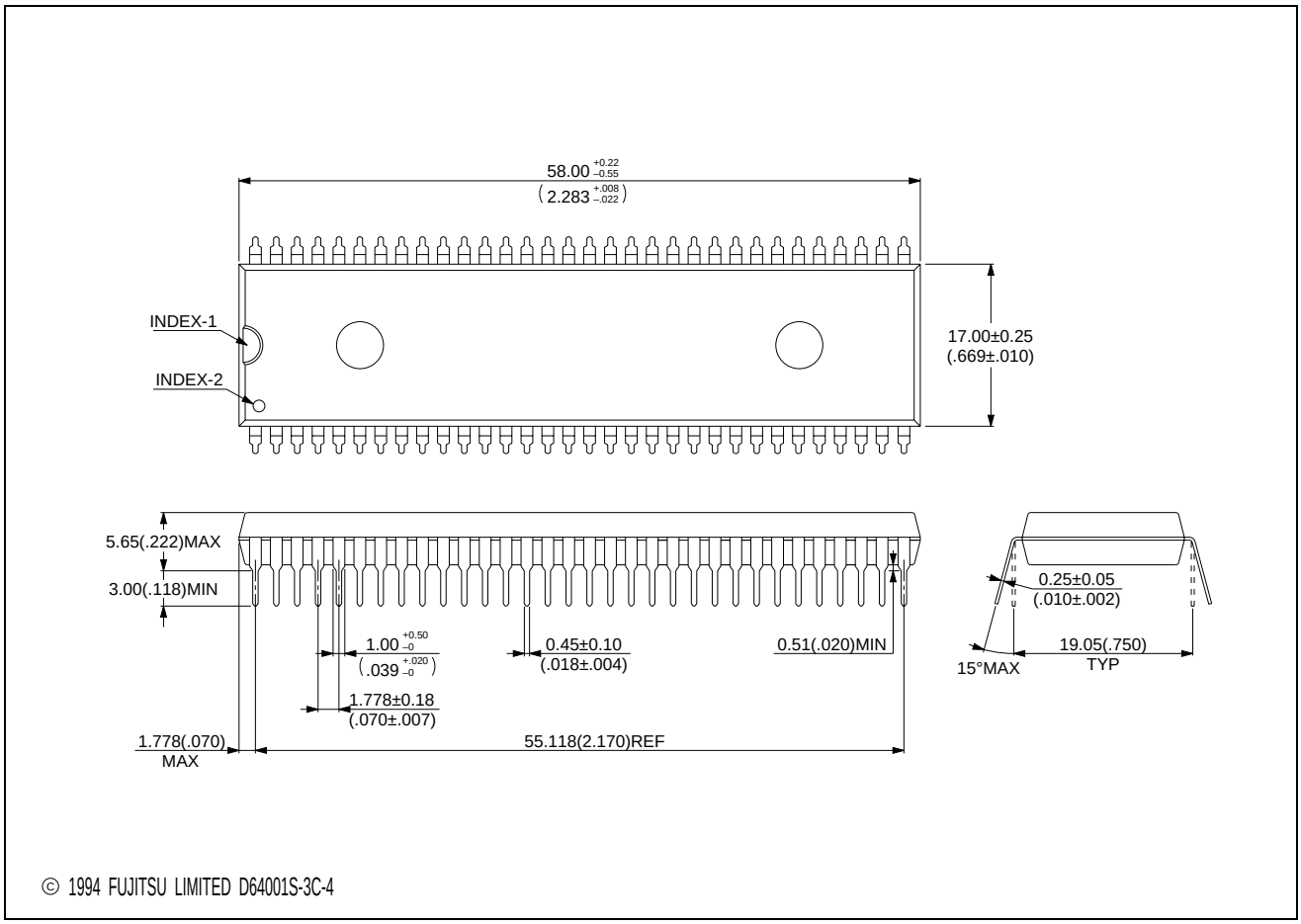


Figure 1.5-3 Dimensions of DIP-64P-M01 package

Memo

1.6 Pin Functions

Tables 1.6-1 to 1.6-3 summarize 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	Pin Status during reset	Function
QFPM09*1	QFPM06*2	SDIP*3				
22,23	23,24	30,31	X0·X1	A	Oscillating	Oscillation input pins
19	20	27	RSTX	B	Reset input	External reset input pin
25~32	26~33	33~40	P00~P07	C	Port input	General-purpose I/O ports
33~39	39~40	41~47	P10~P16	C		General-purpose I/O ports
			INT0~6			Can be used as interrupt request input channels 0 to 6. Input is enabled when 1 is set in EN0 to EN6 in standby mode.
40	41	48	P17	C		General-purpose I/O ports
			FRCK			External clock input pin for free running timer
41	42	49	P20	D		General-purpose I/O ports
			TIN0			External clock input pin for reload timer 0
42	43	50	P21	D		General-purpose I/O ports
			T00			Event output pin for reload timer 0
43	44	51	P22	D		General-purpose I/O ports
			TIN1			External clock input pin for reload timer 1
44	45	52	P23	D		General-purpose I/O ports
			T01			Event output pin for reload timer 1
45~48	46~49	53~56	P24~27	D		General-purpose I/O ports
			IN0~3			Trigger input pins for input capture channels 0 to 3. When input capture channels 0 to 3 are used for input operation, these pins are enabled as required and must not be used for any other I/P

*1: FPT-64P-M09
*2: FPT-64P-M06
*3: DIP-64P-M01

Table 1.6-2 Pin functions *(Continued)*

Pin No.			Pin name	Circuit type	Pin Status during reset	Function
QFPM09*1	QFPM06*2	SDIP*3				
50~55	51~56	58~63	P30~P35	D	Port input	General-purpose I/O ports
			RT00~5			Compare event output pints or waveform generator output pins. These pins output the waveforms specified at the waveform generator. They output compare events when waveforms are not generated. Output is generated when compare event output is enabled.
58	59	2	P36	D	Port input	General-purpose I/O ports
			SIN0			Serial data input pin for UART channel 0. While UART channel 0 is operating for input, the input of this pin is used as required and must not be used for any other input.
59	60	3	P37	D		General-purpose I/O ports
			S0T0			Serial data output pin for UART channel 0. This function is enabled when UART channel 0 enables data output.
60	61	4	40	D		General-purpose I/O ports
			SCK0			Serial clock I/O pin for UART channel 0. This function is enabled when UART channel 0 enables clock output.
61~64, 1,2	62~64, 1~3	5~10	P41~P46	F		General-purpose I/O ports
			PPG0~5			Output pins for PPG channels 0 to 5. This function is enabled when PPG channels 0 to 5 enable output.
3~10	4~11	11~18	P50~P57	E	Analog input	General-purpose I/O ports
			AN0~7			A/D converter analog input pins. This function is enabled when the analog input specification is enabled. (ADER)
11	12	19	AVCC	G	Power input	Vss power input pin for analog circuits

*1: FPT-64P-M09
*2: FPT-64P-M06
*3: DIP-64P-M01

Table 1.6-3 Pin functions *(Continued)*

Pin No.			Pin name	Circuit type	Pin Status during reset	Function
QFPM09*1	QFPM06*2	SDIP*3				
12	13	20	AVR	H		Vref+ input pin for the A/D converter. This voltage must not exceed Vcc. Vref- is fixed to AVSS.
13	14	21	AVSS	G	Power input	Vss power input pin for analog circuits
14	15	22	P60	D	Power input	General-purpose I/O ports
			SIN1			Serial data input pin for UART channel 1. While UART channel 1 is operating for input, the input of this pin is used as required and must not be used for any other input.
15	16	23	P61	D		General-purpose I/O ports
			S0T1			Serial data output pin for UART channel 0. This function is enabled when UART channel 1 enables data output.
16	17	24	P62	D	Port input	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.
17	18	25	P63	D		General-purpose I/O port
			DTT1			RTOn pins for fixed-level input. This function is enabled when the waveform generator enables input.
			INT7			Usable as interrupt request input channel 7. Input is enabled when 1 is set in EN7 in standby mode.
57	58	1	C	G	Capacitance pin power input	Capacitance pin for power stabilization. Connect a ceramic capacitor of about 0.1 µF to the outside.
18	19	26	MD0	B		Input pin for operation mode specification. Connect this pin directly to Vcc or Vss.
20,21	21,22	28,29	MD1,MD2	B		Input pin for operation mode specification. Connect this pin directly to Vcc or Vss.
24,49	25,50	32,57	VSS	–		Power (5 V) input pin
56	57	64	VCC	–		Power (5 V) input pin

*1: FPT-64P-M09
*2: FPT-64P-M06
*3: DIP-64P-M01

Memo

1.7 I/O Circuit Types

Table 1.7-1 summarizes the I/O circuit types of the MB90560 series.

■ **I/O Circuit Types**

Table 1.7-1 I/O circuit types

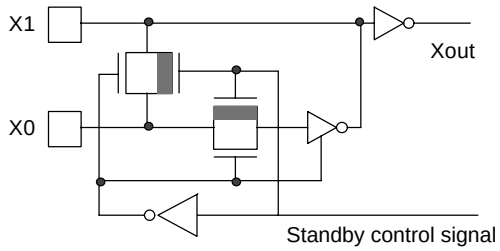
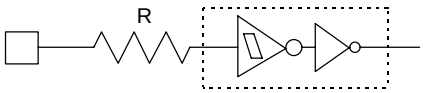
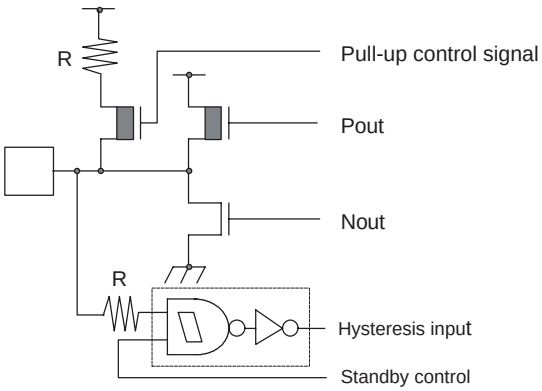
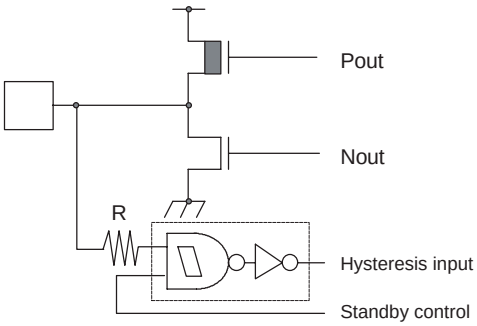
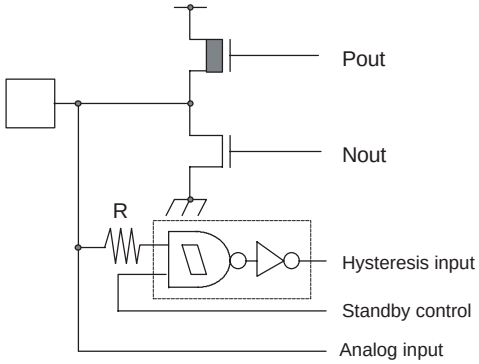
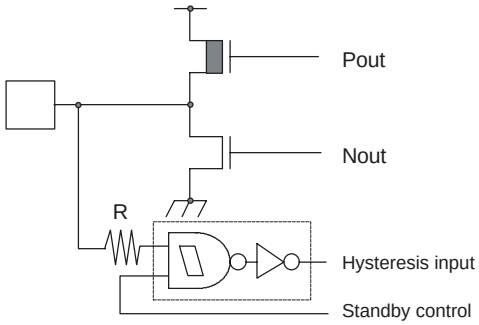
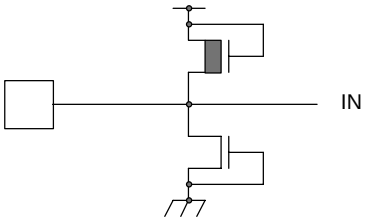
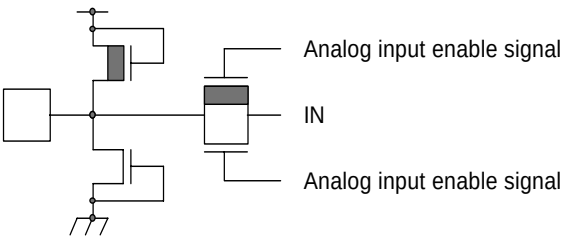
Classification	Circuit type	Remarks
A		• Oscillation circuit Oscillation feedback resistance: About 1 MΩ
B		• Hysteresis input pin Resistance: About 50KΩ(TYP)
C		• CMOS hysteresis input pin with pull-up control CMOS level output CMOS hysteresis input (with standby control for input rejection) Pull-up resistance: About 50KΩ(TYP) I_{OL}=4mA
D		• CMOS hysteresis I/O pin CMOS level output CMOS hysteresis input (with standby control for input rejection) I_{OL}=4mA

Table 1.7-2 I/O circuit types *(Continued)*

Classification	Circuit type	Remarks
E		• Analog/CMOS hysteresis I/O pin CMOS level output CMOS hysteresis input (with standby control for input rejection) Analog input (Analog input is enabled when the bit corresponding to ADER is “1”.) I_{OL}=4mA
F		• CMOS hysteresis I/O pin CMOS level output CMOS hysteresis input (with standby control for input rejection) I_{OL}=12mA
G		• Analog power input protection circuit
H		• A/D converter ref+ (AVR) power input pin with power protection circuit

1.8 Notes on Handling Devices

When handling devices, pay special attention to the following eight items or procedures:

- **Restriction of maximum rated voltage (latchup prevention)**
- **Stabilization of supply voltage**
- **Power-on**
- **Treatment of unused input pins**
- **Treatment of A/D converter power pin**
- **Notes on external clock**
- **Power supply pin**
- **Analog power-on sequence of A/D converter**

■ Notes on Handling Devices

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

  A latchup may occur on a CMOS IC if a voltage higher than Vcc or lower than Vss is applied to an input or output pin other than medium-to-high voltage pins. A latchup may also occur if a voltage higher than the rating is applied between Vcc and Vss. A latchup causes a rapid increase in the power supply current, which can result in thermal damage to an element. Take utmost care that the maximum rated voltage is not exceeded.

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

- Try to keep supply voltages stable.

  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.

- Notes on power-on

  To prevent a malfunction in the built-in voltage regulator 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 pin may cause a malfunction if it is left open. Every unused pin should be pulled up or down.

- Treatment of A/D converter power pin

  When the A/D converter is not used, connect the pins as follows: AVcc =Vcc, AVss = AVR = 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 stop mode. As shown in Figure 1.8-1, when an external clock is used, connect only the X0 pin and leave the X1 pin open.

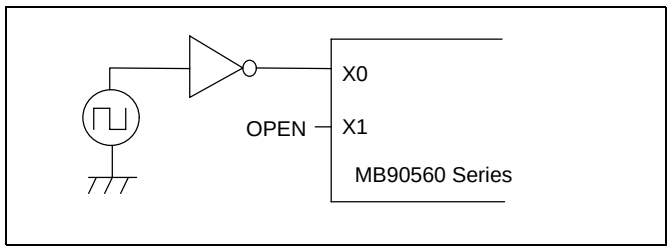


Figure 1.8-1 Sample connection of external clock

- 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 ground them.

  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.

- Analog power-on sequence of A/D converter

  The power to the A/D converter (AVcc, AVRH, AVRL) 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.)

CHAPTER 2 CPU

This chapter describes memory space for the MB90560 series.

2.1 CPU

The F²MC-16LX CPU core is a 16-bit CPU designed for use in applications, such as consumer and mobile equipment, which require high-speed real-time processing. The instruction set of the F²MC-16LX was designed for controllers so that it can perform various types of control at high speeds and efficiencies.
The F²MC-16LX CPU core process not only 16-bit data but also 32-bit data using a built-in 32-bit accumulator. Memory space, which can be extended up to 16M bytes, can be accessed in either linear or bank access mode. The instruction set inherits the AT architecture of F²MC-8L, and has additional instructions supporting high-level languages. In addition, it has an extended addressing mode, enhanced multiply/divide instructions, and reinforced bit manipulation instructions.

■ CPU

- Minimum instruction execution time: 62.5 ns (source oscillation at 4 MHz and PLL 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
- Enhanced interrupt function

  Eight programmable priority levels
- Automatic transfer function independent of CPU

  Extended intelligent I/O service using up to 16 channels
- Instruction set supporting high-level language (C language) and multitasking

  System stack pointer, instruction set symmetry, and barrel shift instructions
- Increased execution speed: 4-byte instruction queue

<Check>

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

Memo

2.2 Memory Space

All I/O, programs, and data are located in the 16-megabyte memory space of the F²MC-16LX. A part of the memory space is used for special purposes, such as extended intelligent I/O service (EI²OS) descriptors, general-purpose registers, and vector tables.

■ Memory space

All I/O, programs, and data are located in the 16-megabyte memory space of the F²MC-16LX CPU. The CPU is able to access each resource through an address indicated by the 24-bit address bus.

Figure 2.2-1 shows a sample relationship between the F²MC-16LX system and the memory map.

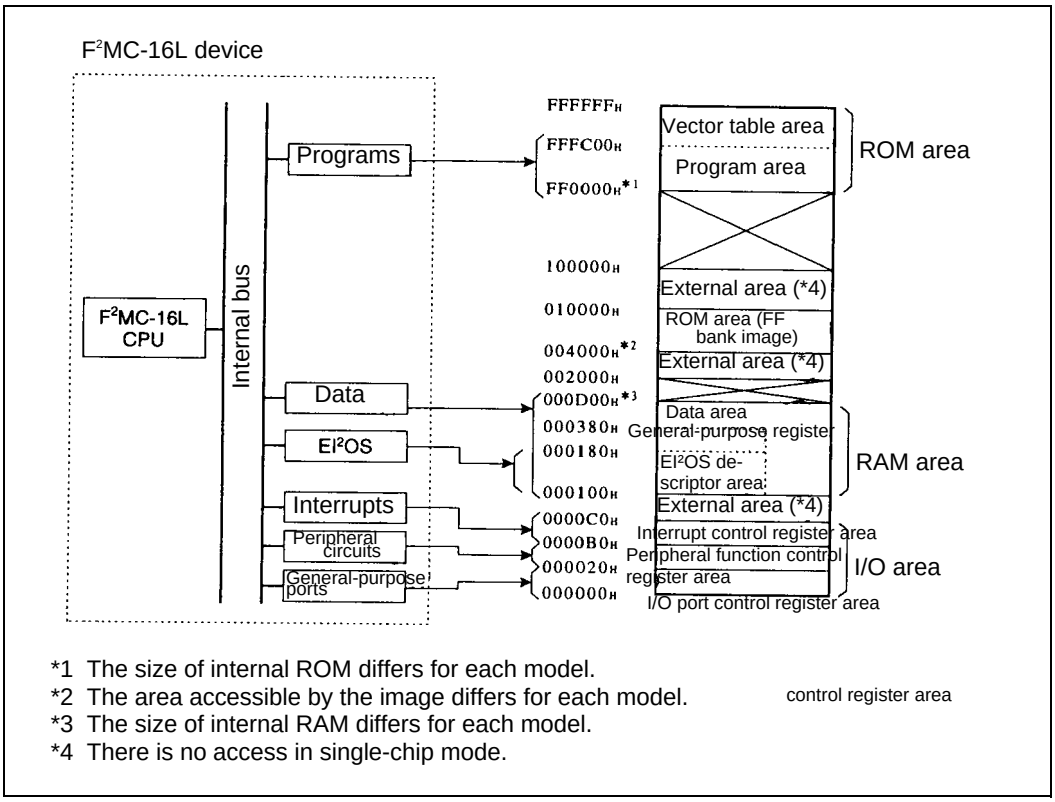


Figure 2.2-1 Sample relationship between the F²MC-16LX system and the memory map

■ ROM area

● Vector table area (address: FFFC00H to FFFFFFH)

- 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 starting address of the corresponding processing routine is stored as data in each vector table address.

● Program area (address: Up to FFFBFFH)

- ROM is built in as an internal program area.
- The size of internal ROM differs for each device.

■ RAM area

● Data area (address: From 000100H)

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

● General-purpose register area (address: 000180H to 00037FH)

- Auxiliary registers used for 8-bit, 16-bit, and 32-bit arithmetic operations and transfer operation are allocated in this area.
- Since this area is allocated to a part of the RAM area, 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 with short instructions.

● Extended intelligent I/O service (EI²OS) descriptor area (address: 000100H to 00017FH)

- This area retains the transfer modes, I/O addresses, transfer count, and buffer addresses.
- Since this area is allocated to a part of the RAM area, it can be used as ordinary RAM.

■ I/O area

● Interrupt control register area (address: 0000B0H to 0000BFH)

The interrupt control registers (ICR00 to ICR15) correspond to all peripheral functions that have an interrupt function. These registers set interrupt levels and control the extended intelligent I/O service (EI²OS).

● Peripheral function control register area (address: 000020H to 0000AFH)

This register controls the built-in peripheral functions and inputs and outputs data.

● I/O port control register area (address: 000000H to 00001FH)

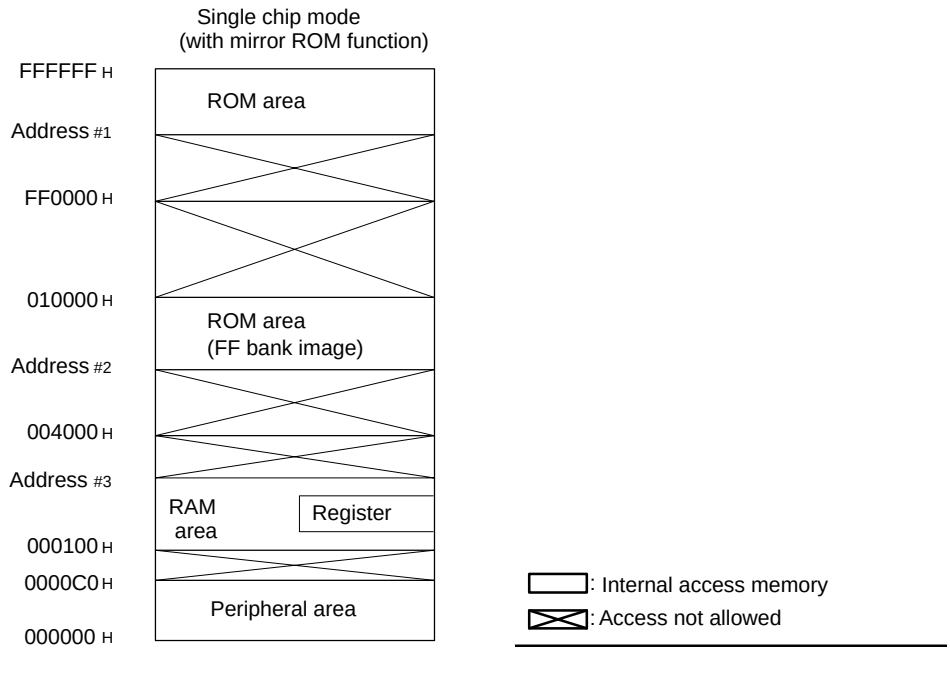
This register controls I/O ports, and inputs and outputs data.

2.3 Memory Maps

This section shows the memory map for each MB90560 series device.

■ Memory maps

Figure 2.3-1 shows the memory maps for the MB90560 series.



Model	Address #1	Address #2	Address #3
MB90561	FF8000H	008000H	000500H
MB90562	FF0000H	004000H	000900H
MB90F562	FF0000H	004000H	000900H
MB90V560	FF0000H *1	004000H *1	001100H

*1: The MB90V560 does not contain ROM. Assume these areas such that the development tool uses there for its ROM decode areas.

Figure 2.3-1 Memory maps

Notes: If single chip mode (without mirror ROM function) is selected, see "Mirror ROM Function Selection Module."

: ROM data in the FF bank can be seen as an image in the higher 00 bank to validate the small model C compiler. Because addresses of the 16 low-order bits in the FF bank are the same, the table in ROM can be referenced without the "far" specification. For example, when 00C000H is accessed, the contents of ROM at FFC000H are actually accessed. The ROM area in the FF bank exceeds 48 kilobytes, and all areas cannot be seen as images in the 00 bank. Because ROM data from FF4000H to FFFFFFH is seen as an image at 004000H to 00FFFFH, the ROM data table should be stored in the area from FF4000H to FFFFFFH.

2.4 Addressing

The methods for generating addresses are linear addressing and bank addressing. In linear addressing, the complete 24-bit address is specified directly by an instruction. In bank addressing, the upper 8 bits of the address are specified by a appropriate bank register, and the lower 16 bits of the address are specified by the instruction. The F²MC-16LX series generally uses bank addressing.

■ Linear addressing and bank addressing

In linear addressing, the 16-megabyte space is accessed as consecutive address spaces. In bank addressing, the 16-megabyte space is managed by dividing into 256 64-kilobyte banks.

Figure 2.4-1 is an overview of linear addressing and bank addressing memory management.

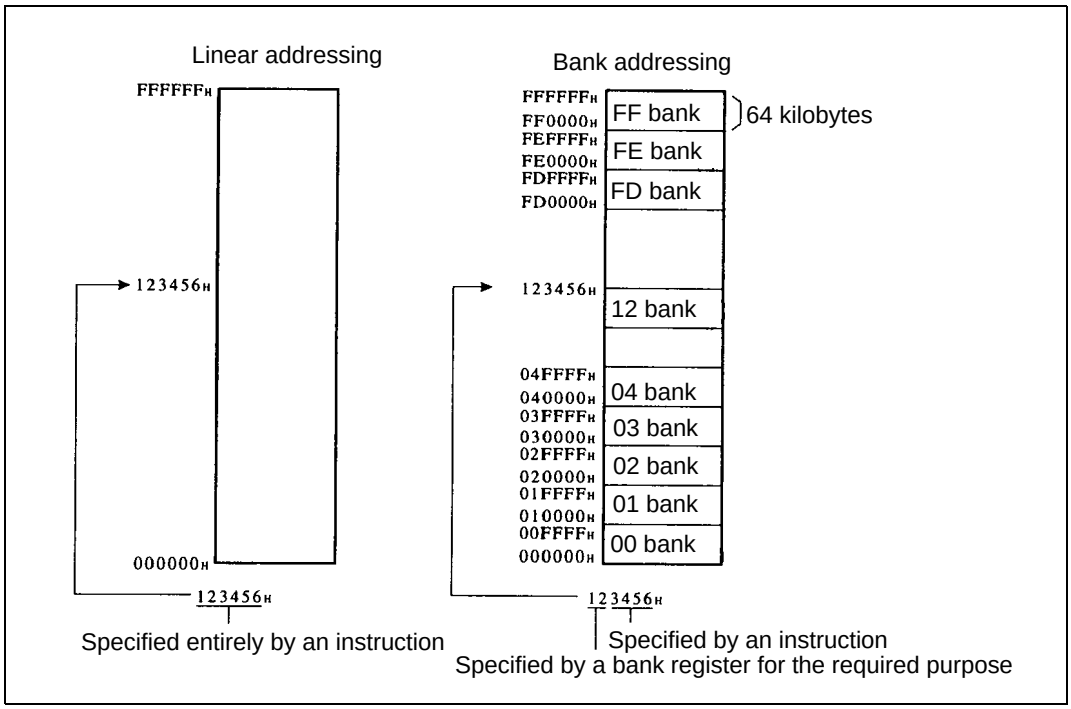


Figure 2.4-1 Linear addressing and bank addressing memory management

2.4.1 Linear addressing

The two types of address in linear addressing are specified by a 24-bit address directly in the operand and specified by the lower 24 bits of a 32-bit general-purpose register.

■ **Linear addressing specified by 24-bit operand**

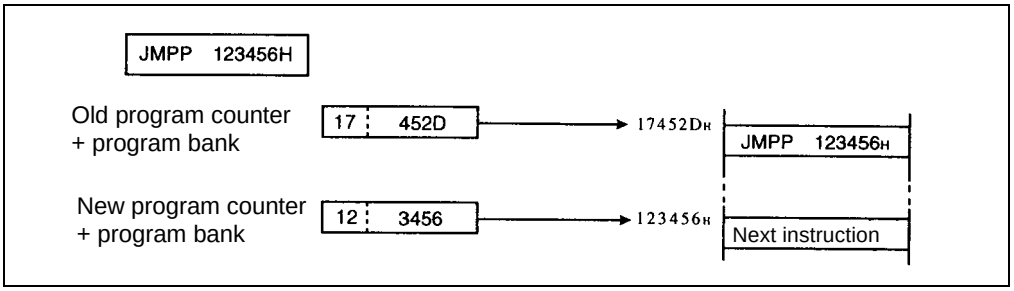


Figure 2.4-2 Example of direct specified 24-bit physical address in linear addressing

■ **Addressing by indirect specification with a 32-bit register**

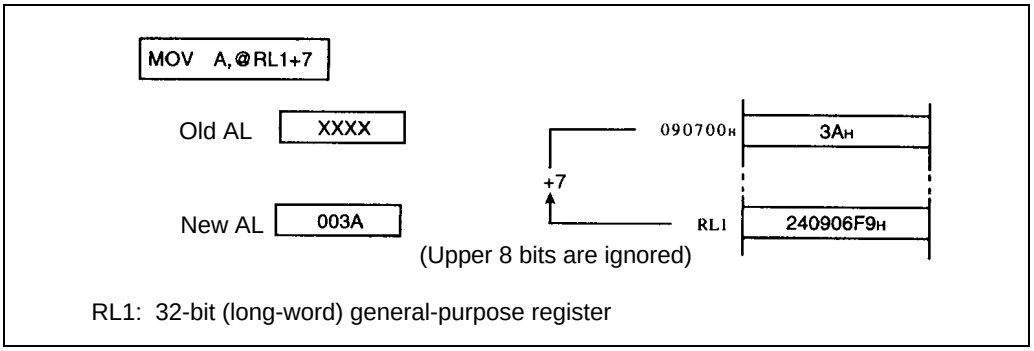


Figure 2.4-3 Example of indirect specified address with a 32-bit general-purpose register in linear addressing

Memo

2.4.2 Bank addressing

In bank addressing, the 16-megabyte memory space is divided into 256 64-kilobyte banks. A bank address that corresponds to each space is specified in the bank register to determine the upper 8 bits of the address. The lower 16 bits of the address are specified by the instruction.
The five types of bank register 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 lists the access space and main function of each bank register.

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 code register (CCR:S) is 1. The USB is used when the stack flag in the condition code 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.

Figure 2.4-4 shows the relationship between the memory space divisions and each register.

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

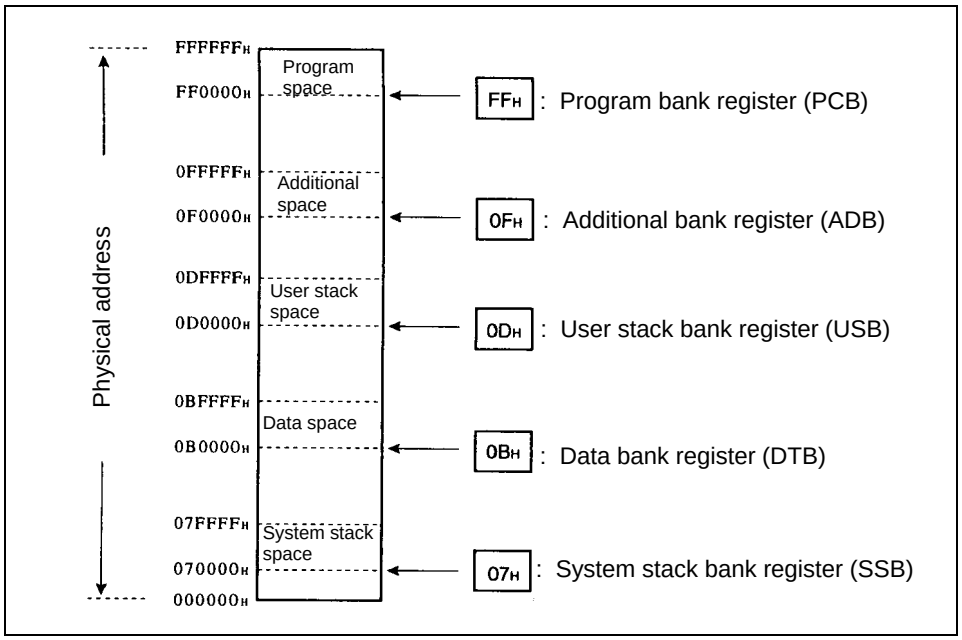


Figure 2.4-4 Sample bank addressing

■ Bank addressing and default space

To improve instruction coding efficiency, each instruction has a pre-determined default space for each addressing mode, as shown in Table 2.4-2. 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 before the upper 16 bits.
If a reset signal is input immediately after the low-order data is written, the high-order bits may not be written.

■ Storage of multibyte data in RAM

Figure 2.5-1 shows the data configuration of multibyte data in memory. 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.

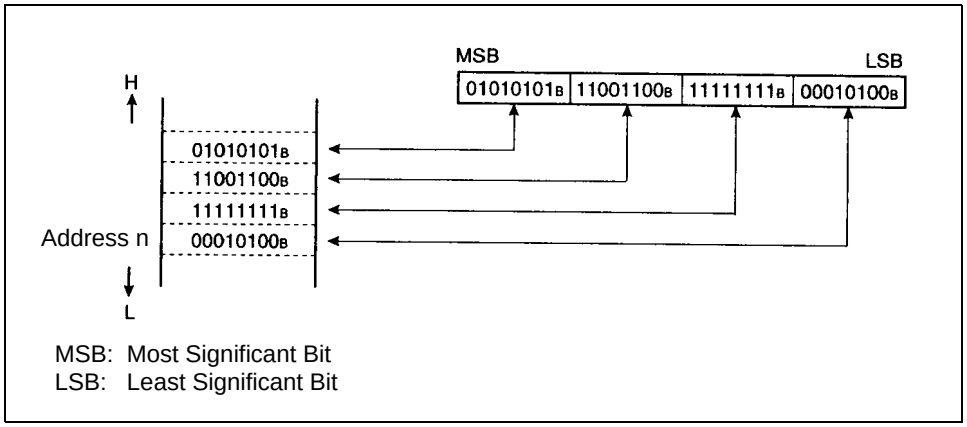


Figure 2.5-1 Storage of multibyte data in RAM

■ Storage of multibyte operand

Figure 2.5-2 shows the configuration of a multibyte operand in memory.

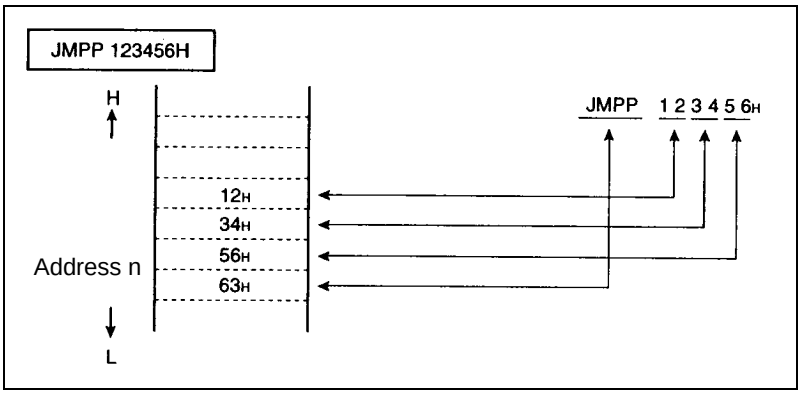


Figure 2.5-2 Storage of a multibyte operand

■ Storage of multibyte data in a stack

Figure 2.5-3 shows the configuration of multibyte data in a stack.

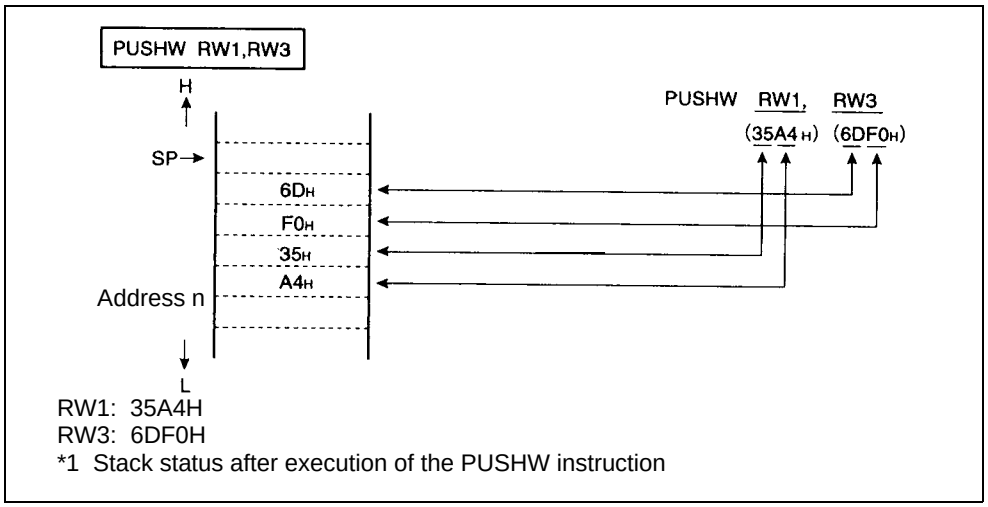


Figure 2.5-3 Storage of multibyte data in a stack

■ Multibyte data access

Accessing is generally performed within a bank. For an instruction that accesses multibyte data, the address following $FFFF_H$ is 0000_H in the same bank.

Figure 2.5-4 shows an example of executing an instruction that accesses multibyte data on a bank boundary.

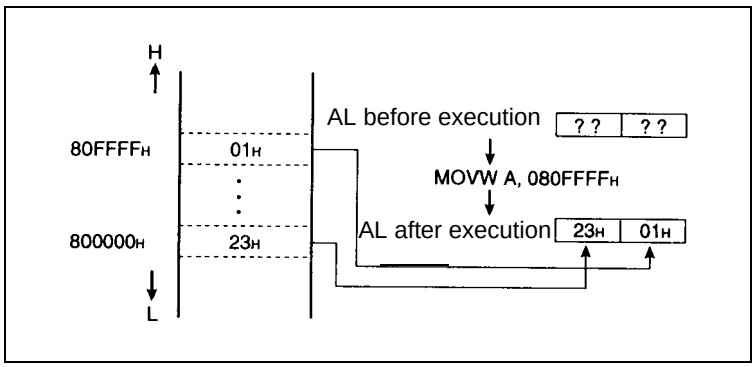


Figure 2.5-4 Multibyte data access on a bank boundary

2.6 Registers

F²MC-16LX registers are classified into internal dedicated registers and built-in RAM general-purpose registers.

■ Dedicated registers and general-purpose registers

Dedicated registers are dedicated hardware inside the CPU and their usage are limited by CPU architecture.

General-purpose registers are shared the CPU address space with RAM. Just like dedicated registers, general-purpose registers can be accessed without addressing. Just like ordinary memory, the user can specify how the register is used.

Figure 2.6-1 shows the location of the dedicated registers and general-purpose registers in the device.

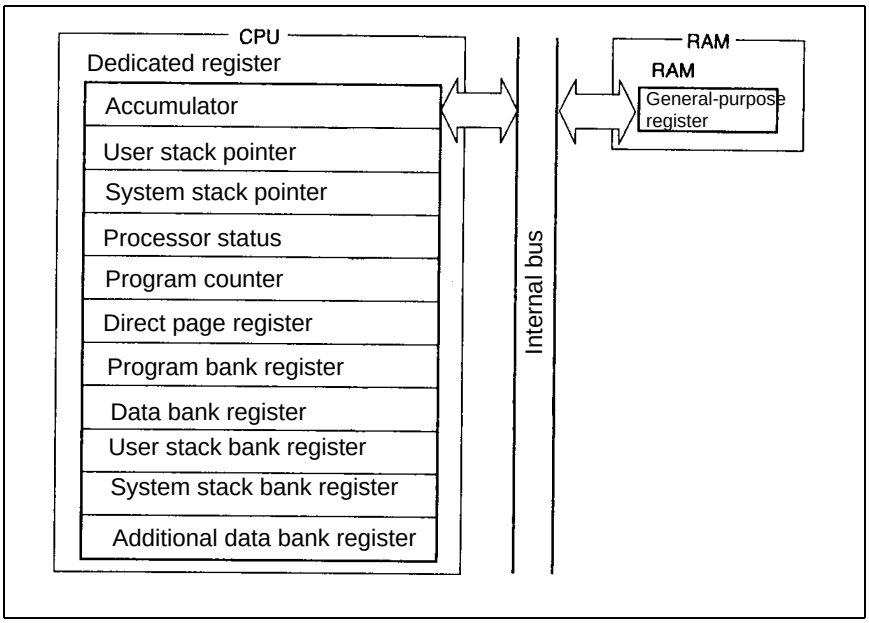


Figure 2.6-1 Dedicated registers and general-purpose registers

Memo

2.7 Dedicated Registers

The following 11 registers are dedicated registers in the CPU.

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

■ **Configuration of dedicated registers**

Figure 2.7-1 shows the configuration of dedicated registers; Table 2.7-1 lists the initial values of the dedicated registers.

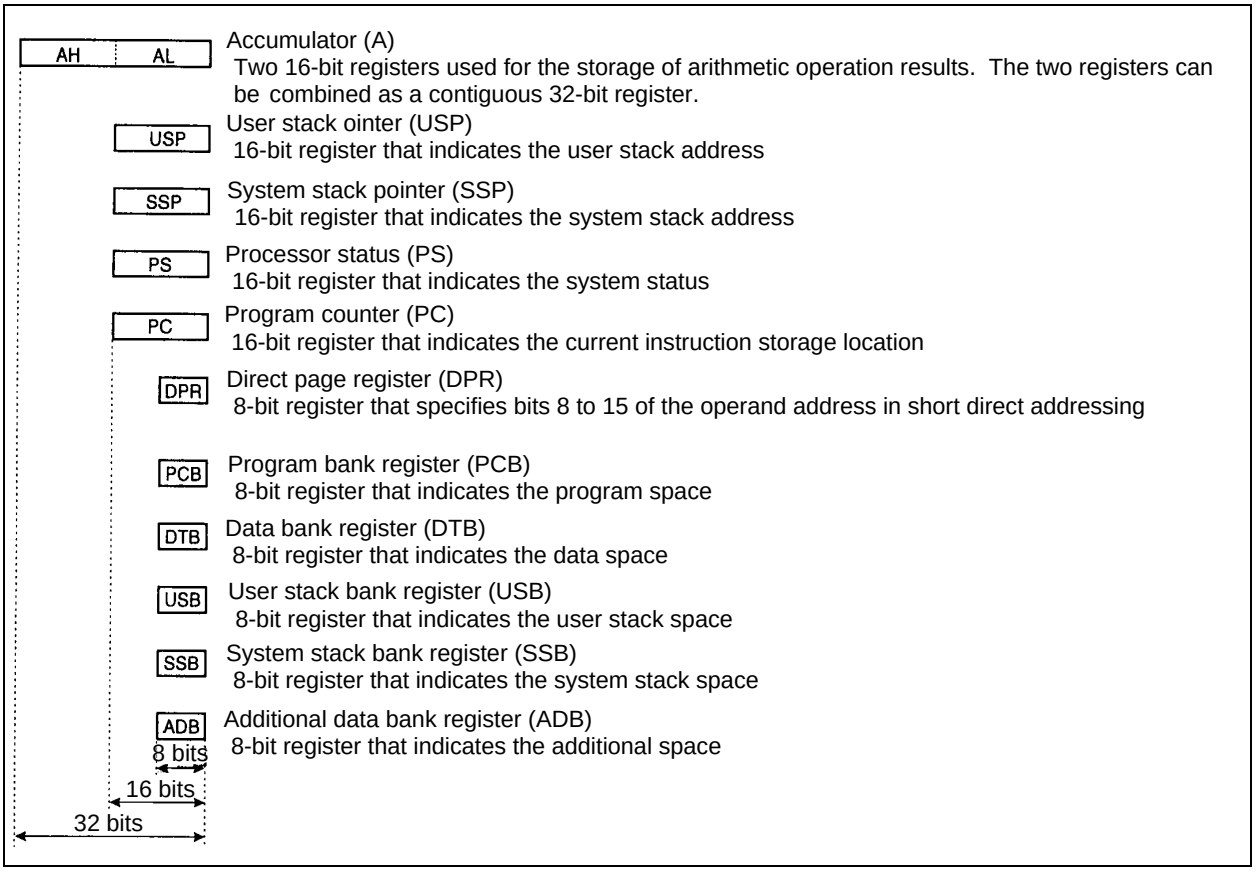
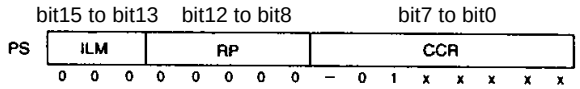


Figure 2.7-1 Configuration of dedicated registers

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 FFFFDCH, FFFFDDH)
Direct page register (DPR)	01H
Program bank register (PCB)	Value in reset vector (contents of FFFFDEH)
Data bank register (DTB)	00H
User stack bank register (USB)	00H
System stack bank register (SSB)	00H
Additional data bank register (ADB)	00H

- :Not used

x:Undefined

<Check>

The above initial values are the initial values for the device. They are different from the ICE (emulator, etc.) values.

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 A register can be used as a 32-bit, 16-bit, or 8-bit register. Various arithmetic operations can be performed between memory and other registers or between the AH register and the AL register. The A register has a data retention function that automatically transfers pre-transfer data from the AL register to the AH register when data of word length or less is transferred to the AL register. (Data is not retained with some instructions.)

■ Accumulator (A)

● Data transfer to the accumulator

The accumulator can process 32-bit (long word), 16-bit (word), and 8-bit (byte) data. The 4-bit data transfer instruction (MOVN) is an exception. The explanation of 8-bit data also applies to 4-bit data.

- For 32-bit data processing, the AH register and AL register are combined.
- For 16-bit data and 8-bit data, only the AL register is used.
- When data of byte length or less is transferred the AL register, data becomes 16 bits long by sign extension or zero extension, and is stored in the AL register. Data in the AL register can be handled as word-length or byte-length data.

Figure 2.7-2 shows data transfer to the accumulator. Figures 2.7-3 to 2.7-6 show specific transfer examples.

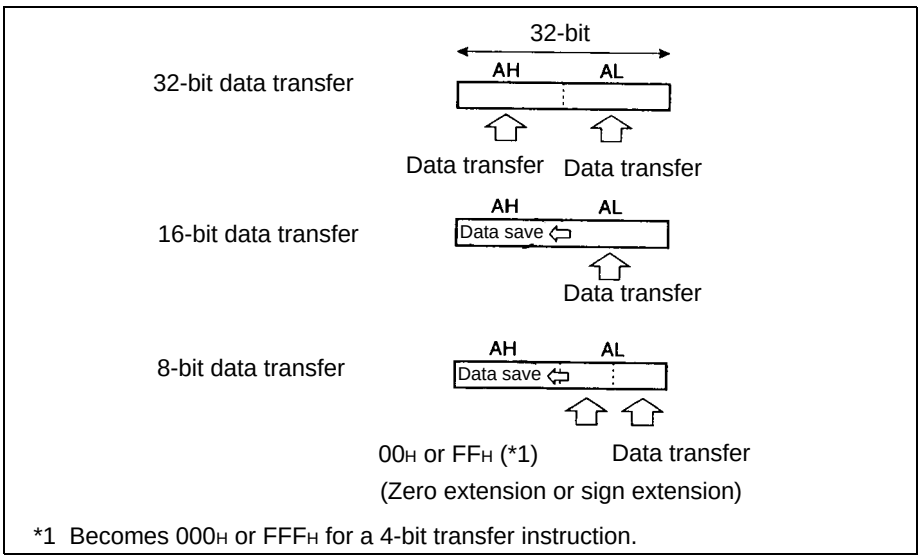


Figure 2.7-2 Data transfer to the accumulator

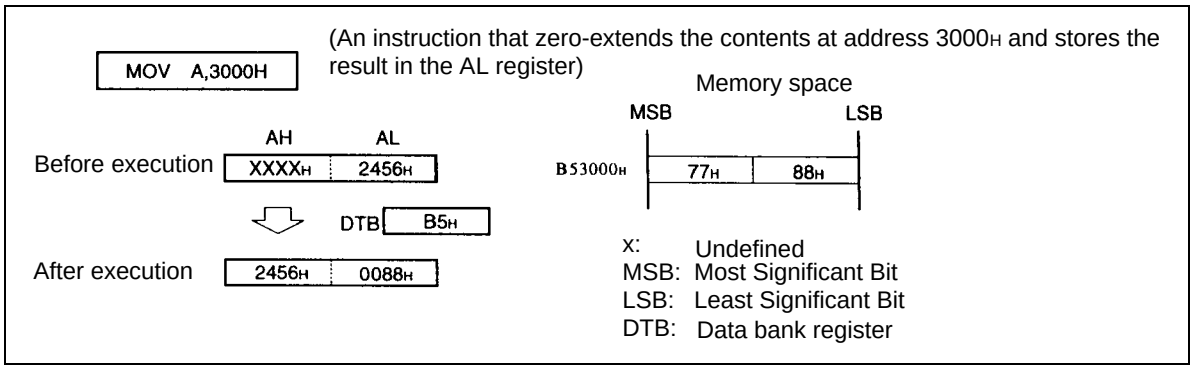


Figure 2.7-3 Example of AL-AH transfer in the accumulator (A) (8-bit immediate value, zero extension)

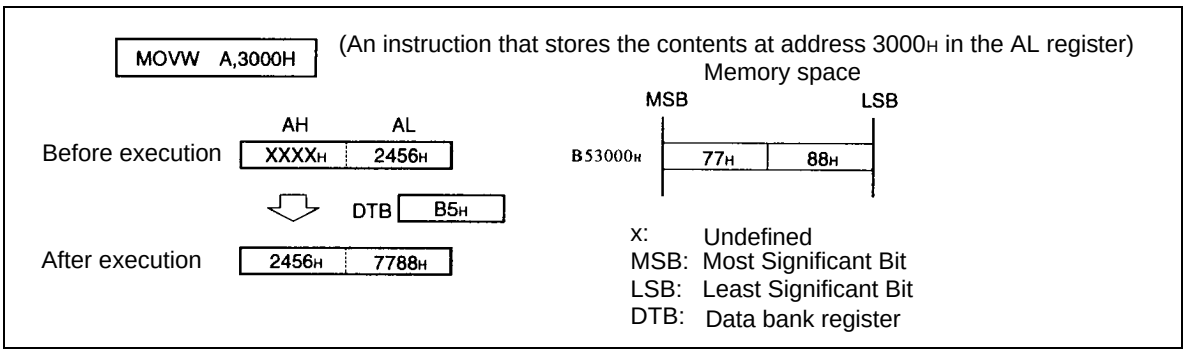


Figure 2.7-4 Example of AL-AH transfer in the accumulator (A) (8-bit immediate value, sign extension)

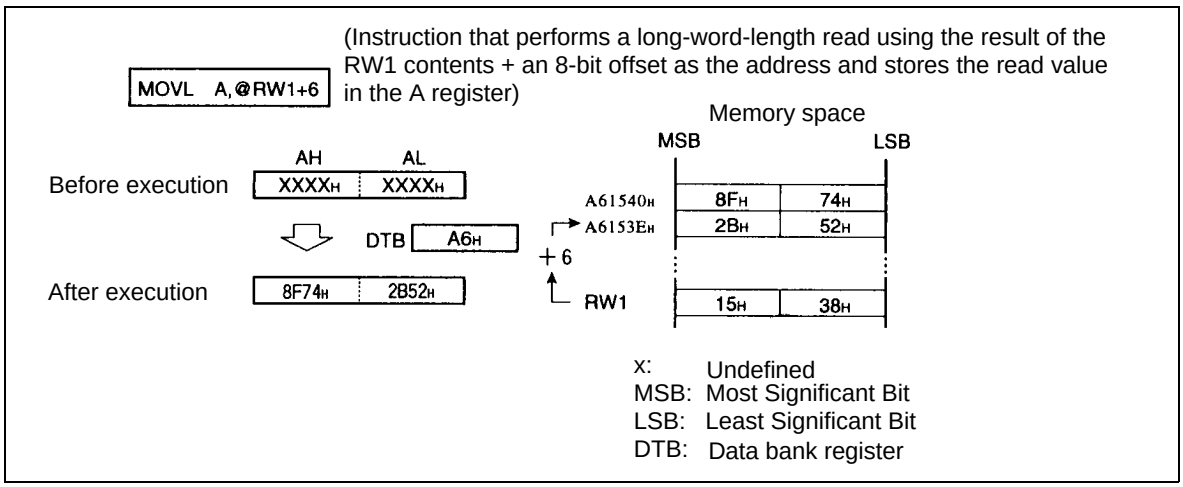


Figure 2.7-5 Example of 32-bit data transfer to the accumulator (A) (register indirect)

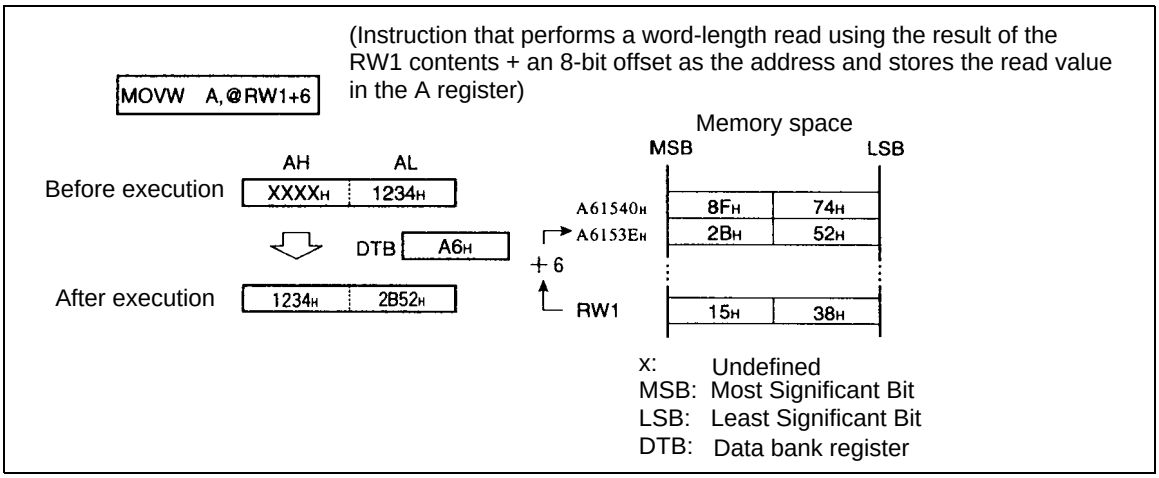


Figure 2.7-6 Example of AL-AH transfer in the accumulator (A) (16 bits, register indirect)

- **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.

Memo

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 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, by the S flag in the processor status register (PS:CCR).

Table 2.7-2 Stack address specification

S flag	Stack address	
	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

Because the S flag is initialized to 1 by a reset, the system stack is used as the default.

Ordinarily, the system stack is used for interrupt routine stack operations, and the user stack is used for all other types of stack operation. When separation of the stack space is not particularly necessary, only the system stack should be used.

<Check>

Since the S flag is set to 1 when an interrupt is accepted, the system stack is always used for interrupts.

Figure 2.7-7 shows an example of stack operation with the system stack.

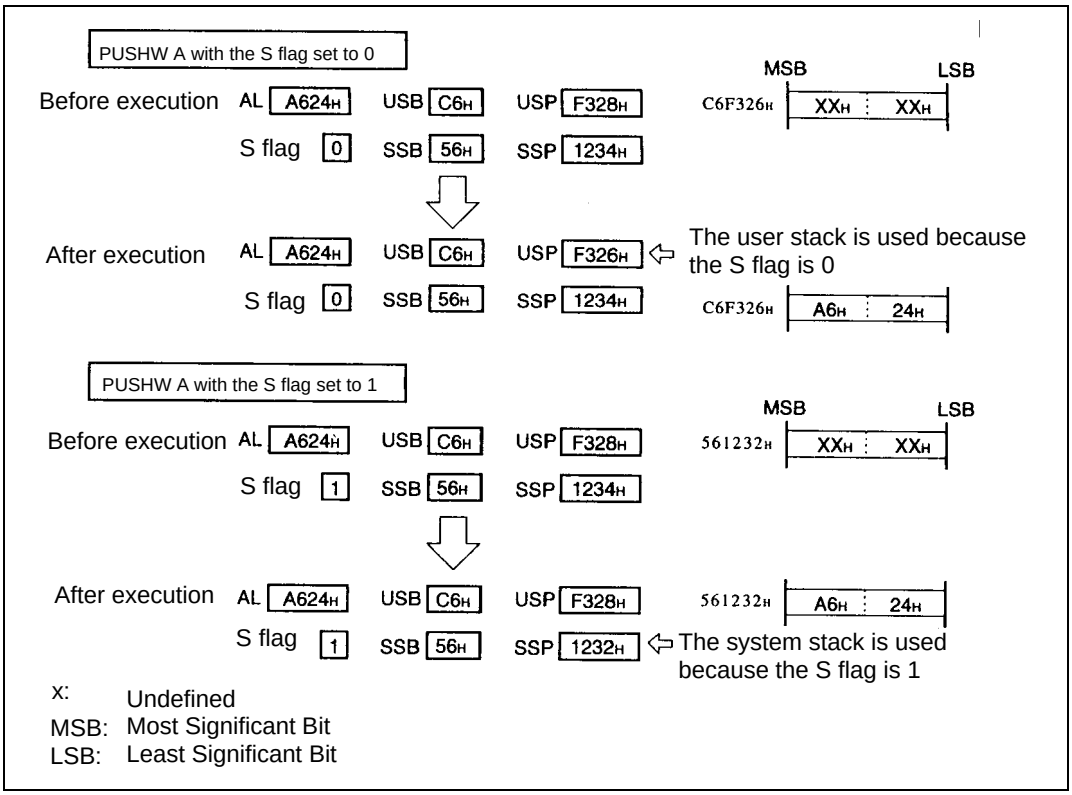


Figure 2.7-7 Stack operation instruction and stack pointer

<Notes>

- To set a value for the stack pointer, always use an even-numbered address. If an odd-numbered address is used, a word access is split into two parts, lowering efficiency.
- The initial values of the USP register and SSP register after a reset are undefined.

■ System stack pointer (SSP)

To use the system stack pointer (SSP), set the S flag in the condition code register (CCR) of the processor status (PS) to 1. The upper 8 bits of the address that will be used for the stack operation are indicated by the system stack bank register (SSB).

■ User stack pointer (USP)

To use the user stack pointer (USP), set the S flag in the condition code register (CCR) of the processor status (PS) to 0. The upper 8 bits of the address that will be used for the stack operation are indicated by the user stack bank register (USB).

2.7.3 Processor Status (PS)

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

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

■ Processor status (PS) configuration

The processor status (PS) consists of CPU control bits and bits that indicate the CPU status.

Figure 2.7-8 shows the configuration of the processor status (PS).

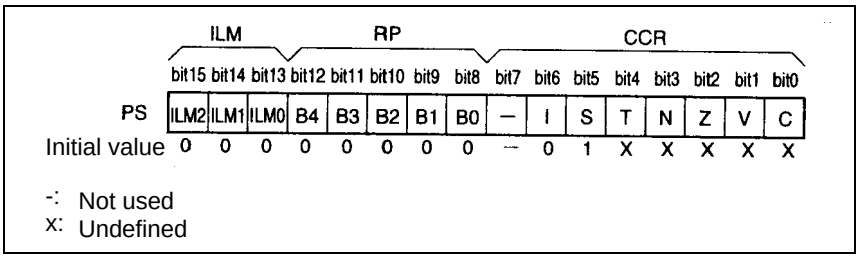


Figure 2.7-8 Processor status (PS) configuration

● Interrupt level mask register (ILM)

This register indicates the level of the interrupt currently accepted by the CPU. The value is compared with the value of the interrupt level setting bits (ICR: IL0 to IL2) in the interrupt control register for the peripheral resource interrupt request.

● 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. Values 0 to 31 are set in the RP to specify a bank.

● Condition code register (CCR)

This register consists of flags that are set to 1 or reset to 0 by instruction execution results and by interrupts.

Memo

2.7.4 Condition code register (PS: CCR)

The condition code register (CCR) is an 8-bit register that consists of the bits that indicate the results of an arithmetic operation and the contents of transfer data and bits that control interrupt request acceptance.

■ Condition code register (CCR) configuration

Figure 2.7-9 shows the configuration of the CCR register. Refer to the programming manual for details about the status of the condition code register (CCR) during instruction execution.

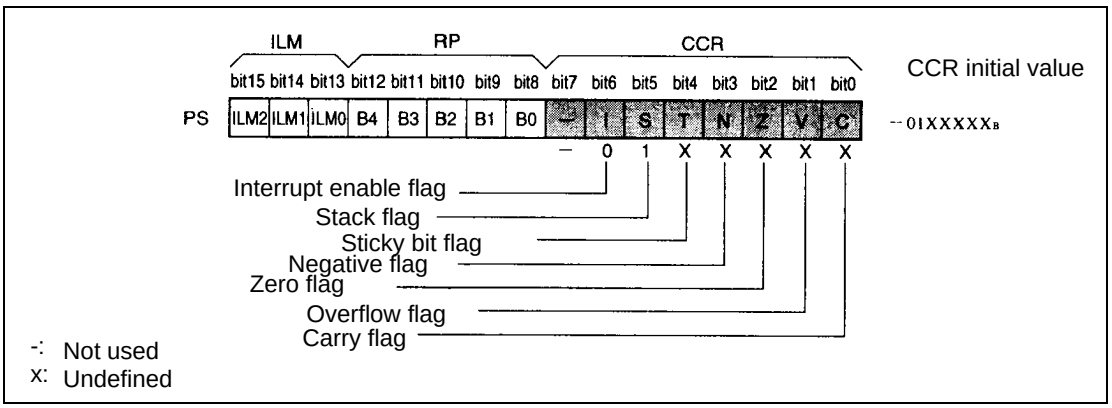


Figure 2.7-9 Condition code register (CCR) configuration

- **Interrupt enable flag (I)**

  In response to all interrupt requests other than software interrupts, when the I flag is “1”, interrupts are enabled. When the I flag is “0”, interrupts are disabled. This flag is cleared by a reset.

- **Stack flag (S)**

  This flag indicates which pointer is used for a stack operation.

  When the S flag is “0”, the user stack pointer (USP) is valid. When the S flag is “1”, the system stack pointer (SSP) is valid. Set when an interrupt is accepted or when a reset occurs.

- **Sticky bit flag (T)**

  “1” is set in the T flag when there is at least one “1” in the data shifted out from the carry after execution of a logical/arithmetic right shift instruction. Otherwise, “0” is set in T flag. In addition, “0” is set in T flag when the shift value is zero.

- **Negative flag (N)**

  Set to “1” when the MSB is “1” as the result of an arithmetic calculation. Cleared to “0” when the MSB is 0.

- **Zero flag (Z)**

  Set to “1” when the result of an arithmetic calculation is all zeros. Otherwise, set to “0”.

- **Overflow flag (V)**

  Set to “1” if a signed numeric value overflows because of an arithmetic calculation. Cleared to “0” if no overflow occurs.

- **Carry flag (C)**

  Set to “1” when there is an overflow from the MSB or an underflow from the LSB because of an arithmetic calculation. Cleared to “0” when there is no overflow or underflow because of an arithmetic calculation.

2.7.5 Register bank pointer (PS: RP)

The register bank pointer (RP) is a register that indicates the first address of the general-purpose register bank currently being used. The RP is used for real address conversion when general-purpose register addressing is used.

■ Register bank pointer (RP)

Figure 2.7-10 shows the configuration of the register bank pointer (RP) register.

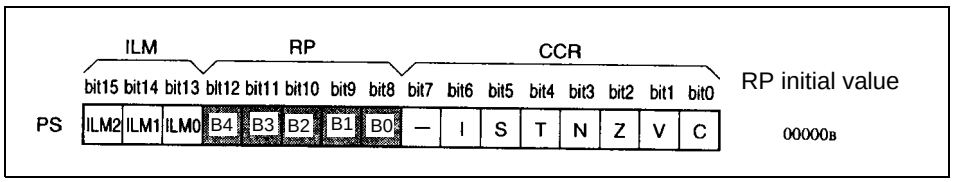


Figure 2.7-10 Configuration of the register bank pointer (RP)

■ General-purpose register area and register bank pointer

The register bank pointer points to the relationship between the general-purpose register of the F²MC-16LX and the address in internal RAM where the general-purpose register exists. Figure 2.7-11 shows the conversion rules used for the relationship between the contents of the RP and the real address.

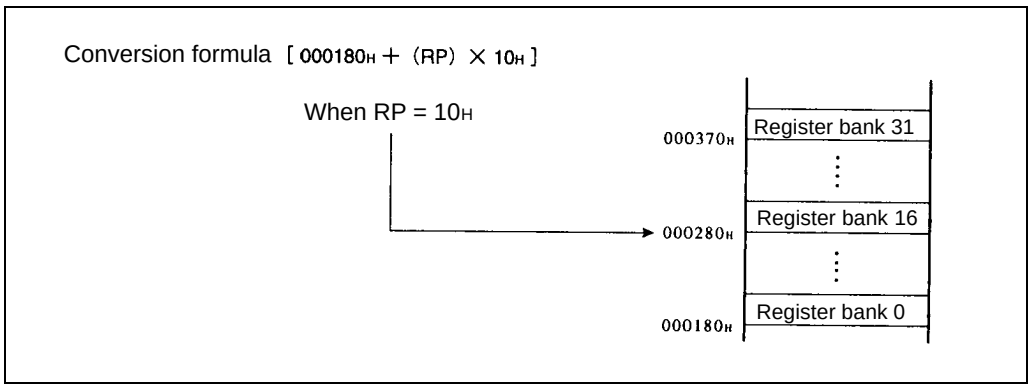


Figure 2.7-11 Conversion rules for physical address of general-purpose register area

- Since the RP takes a value from 00H to 1FH, the first address of the register bank can be set in the range from 000180H to 00037FH.
- Although an assembler instruction can use an 8-bit immediate value transfer instruction for transfer to the RP, in actuality only the lower 5 bits of the data are used.
- The initial value of the RP register after a reset is 00H.

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)

Figure 2.7-12 shows the configuration of the interrupt level mask register (ILM). See Chapter 6, "Interrupts," for details about interrupts.

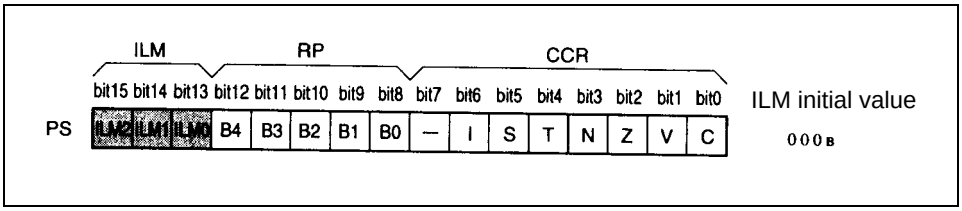


Figure 2.7-12 Configuration of the interrupt level mask register (ILM)

The interrupt level mask register (ILM) indicates the level of the interrupt currently accepted by the CPU. The level is compared with the value of the IL0 to IL2 bits of the interrupt control register (ICR00 to ICR15) set according to the interrupt request from the peripheral function. If the interrupt enable flag has been set to enable (CCR: I = 1), the CPU processes the instruction only when the value (interrupt level) of the interrupt request is smaller than the value indicated by these bits.

- When an interrupt is accepted, the interrupt level value is set in the interrupt level mask register (ILM). Thereafter, interrupts with the same or lower level are not accepted.
- The interrupt level is set to the highest level, which is the interrupts disabled status, because the interrupt level mask register (ILM) is initialized to all 0s by a reset.
- Although an assembler instruction can use an 8-bit immediate value transfer instruction for transfer to the interrupt level mask register (ILM), only the lower 3 bits of the data are used.

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 memory address of the next instruction code to be executed by the CPU.

■ **Program counter (PC)**

The program bank register (PCB) specifies the upper 8 bits of the address where the next instruction code to be executed by the CPU is stored. The PC specifies the lower 16 bits. Before being used, the actual address is combined to become 24 bits, as shown in Figure 2.7-13.

The contents of the PC are updated by conditional branch instructions, subroutine call instructions, interrupts, and resets.

The PC can be used as a base pointer for reading operands.

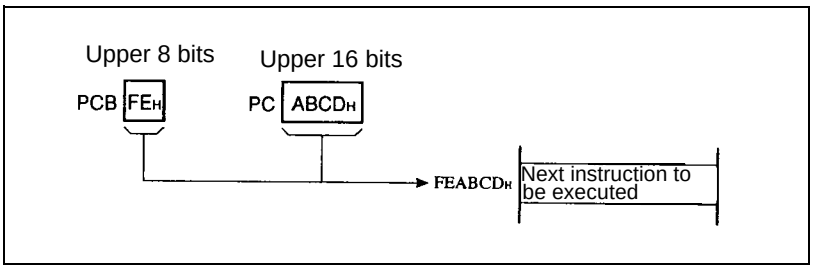


Figure 2.7-13 Program counter (PC)

<Check>

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

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.

■ Direct page register (DPR)

As shown in Figure 2.7-14, the DPR specifies bits 8 to 15 (addr8 to addr15) of the operand address when a short direct addressing instruction is executed. The DPR is 8-bits long. The DPR is initialized to 01H by a reset. The DPR can be read and written using an instruction.

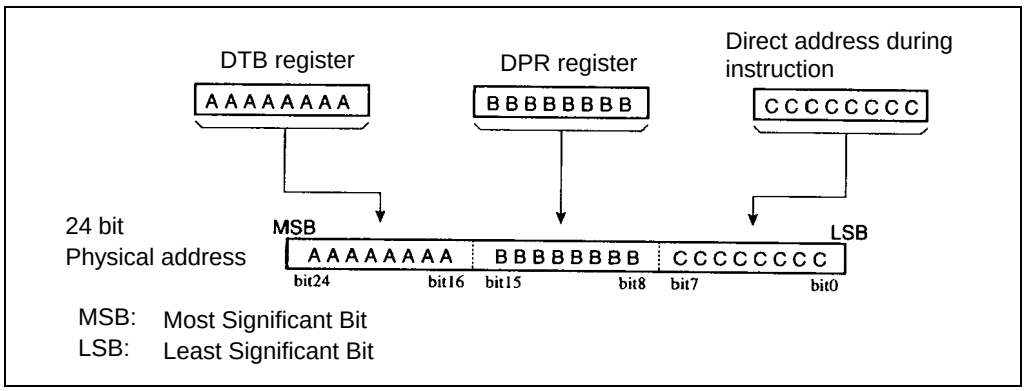


Figure 2.7-14 Physical address generation by the direct page register (DPR)

Figure 2.7-15 shows an example of direct page register (DPR) setting and data access

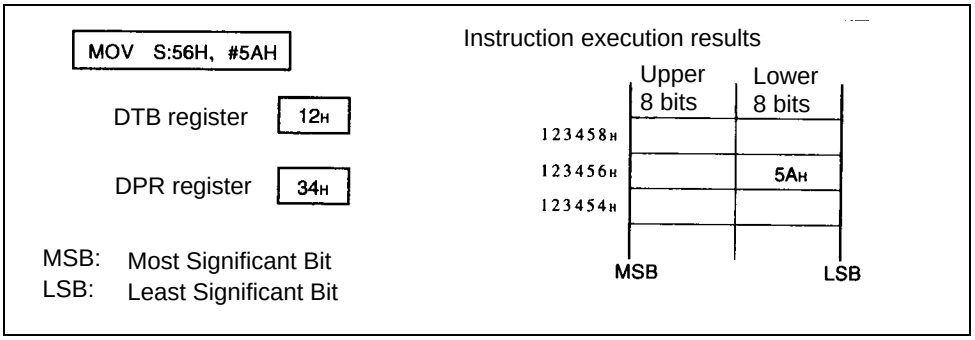


Figure 2.7-15 Example of direct page register (DPR) setting and data access

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 (USB)**
- **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.

The PCB is updated when a software interrupt instruction is executed, when the JMPP, CALLP, RETP, and RETI instructions that branch anywhere within the 16-megabyte space are executed, or when a hardware interrupt or exception occurs.

● 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. Whether the USB or the SSB is used depends on the S flag value in the processor status (PS: CCR). See Section 2.7.2, "Stack pointers (USP, SSP)," for details.

● Additional bank register (ADB)

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

● Bank setting and data access

All bank registers are byte length. The PCB is initialized to FF_H by a reset. The other bank registers are initialized to 00_H by a reset. The PCB can be read, but cannot be written to.

The other bank registers can be read and written to.

<Check>

The MB90560 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.

Memo

2.8 General-Purpose Registers

The general-purpose registers are a memory block allocated in RAM at 000180H to 00037FH as 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

All general-purpose registers exist in RAM at 000180H to 00037FH and are configured as 32 banks. The register bank pointer (RP) specifies the bank that is to be used for a general-purpose register. The RP points to the bank currently being used.

The RP determines the first address of each bank with the following formula:

Address of first general-purpose register = 000180H + RP x 10H

Figure 2.8-1 shows the location and configuration of the general-purpose register banks in the memory space.

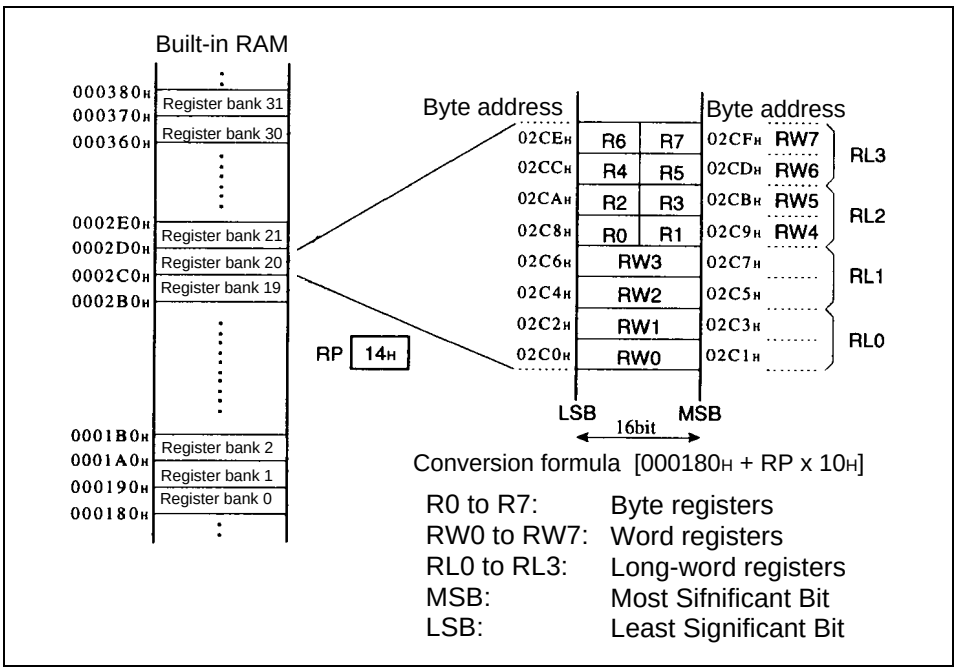


Figure 2.8-1 Location and configuration of the general-purpose register banks in the memory space

<Check>

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

■ Register bank

A register bank can be used as general-purpose registers (byte registers R0 to R7, word registers RW0 to RW7, long-word registers RL0 to RL3) for various arithmetic operations and pointers. A long-word register can be used as a linear pointer that directly accesses the entire memory space.

The contents of the register bank, like ordinary RAM, are not initialized by a reset. The status before a reset is retained. At power-on, however, the contents are undefined.

Table 2.8-1 lists the typical functions of general-purpose registers.

Table 2.8-1 Typical functions of general-purpose registers

Register name	Function
R0 to R7	Used as an operand in various instructions <Caution> 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 <Caution> 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.

● 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.

● 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.

Memo

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

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

■ Bank select prefixes (PCB, DTB, ADB, SPB)

The memory space used for data access is defined for each addressing method. If a bank select prefix is placed before an instruction, the memory space accessed by the instruction can be selected regardless of the addressing method. Table 2.9-1 lists the bank select prefix codes and selected memory spaces.

Table 2.9-1 Bank select prefix codes and selected memory spaces

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 lists the instructions that are not affected by bank select prefix codes. Table 2.9-3 lists instructions that require caution when they are used.

Table 2.9-2 Instructions not affected by bank select prefix codes

Instruction type	Instruction	Effect of bank select prefix
String instruction	MOVS MOVSW SCEQ SCWEQ FILS FILSW	The bank register specified by the operand is used regardless of whether a prefix is used.
Stack operation	PUSHW POPW	When the S flag is “0”, the user stack bank (USB) is used regardless of whether 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 A MOVX A, io MOVW A, io MOV io, A MOVW io, A MOV io, #imm8 MOVW io, #imm16 MOVB A, io : bp MOVB io : bp, A SETB io : bp CLRB io : bp BBC io : bp, rel BBS io : bp, rel WBTC io, bp WBTS io : bp	The I/O space (“000000H“ to “0000FFH”) is accessed regardless of whether there is a prefix.
Interrupt return instruction	RETI	The system stack bank (SSB) is used regardless of whether a prefix is used.

Table 2.9-3 Instructions which use requires caution when bank select prefix codes are used

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)

The common register bank (CMR) 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 000180H to 00018FH regardless of the current register bank pointer (RP) value.

■ Common register bank prefix (CMR)

To facilitate data exchange between multiple tasks, a relatively simple means of accessing a fixed register bank regardless of the current register bank pointer (RP) value is necessary. This is the reason that the F²MC-16LX provides a common register bank for tasks, which is called the common bank. The common bank is located at address 000180H to 00018FH.

If the common register bank prefix (CMR) is placed before an instruction that accesses a register bank, registers accessed by the instruction can be changed to the common bank (register bank selected when RP = 0) at 000180H to 00018FHH regardless of the current register bank pointer (RP) value.

Note that caution is required when this prefix is used with the instructions listed in Table 2.9-4.

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 MOVSW SCEQ SCWEQ FILS 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 placed before an instruction to suppress a flag change accompanying the execution of the instruction.

■ Flag change suppression prefix (NCC)

The flag change suppression prefix (NCC) is used to suppress unnecessary flag changes. If a flag change suppression prefix code is placed before an instruction, a flag change accompanying the execution of the instruction is suppressed. Changes of the T, N, Z, V, and C flags are suppressed.

Note that caution is required when this prefix is used with the instructions listed in Table 2.9-5.

Table 2.9-5 Instructions requiring 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 regardless of whether 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 regardless of whether 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 regardless of whether a prefix is used.
Context switch instruction	JCTX @A	The condition code register (CCR) changes as defined in the instruction specification regardless of whether a prefix is used.

2.9.4 Restrictions on Prefix Codes

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

- **Interrupt/hold requests are not accepted during the execution of prefix codes and interrupt/hold suppression instructions.**
- **If a prefix code is placed before an interrupt/hold 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/hold suppression instructions**

Table 2.9-6 lists the interrupt/hold suppression instructions and prefix codes that have restrictions.

Table 2.9-6 Prefix codes and interrupt/hold suppression instructions

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

● **Interrupt/hold suppression**

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

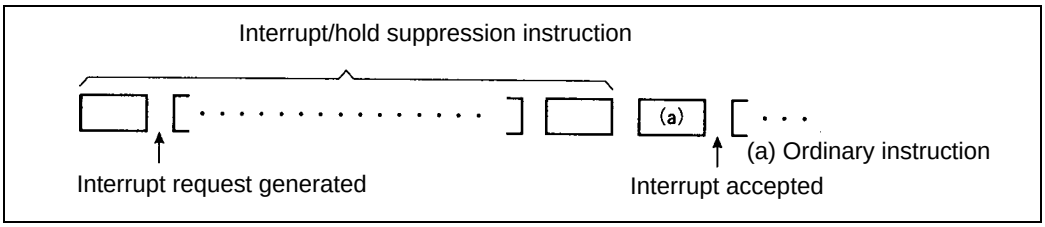


Figure 2.9-1 Interrupt/hold suppression

● **Delay of the effect of prefix codes**

As shown in Figure 2.9-2, if a prefix code is placed before an interrupt/hold suppression instruction, the prefix code takes effect with the first instruction executed after the interrupt/hold suppression instruction.

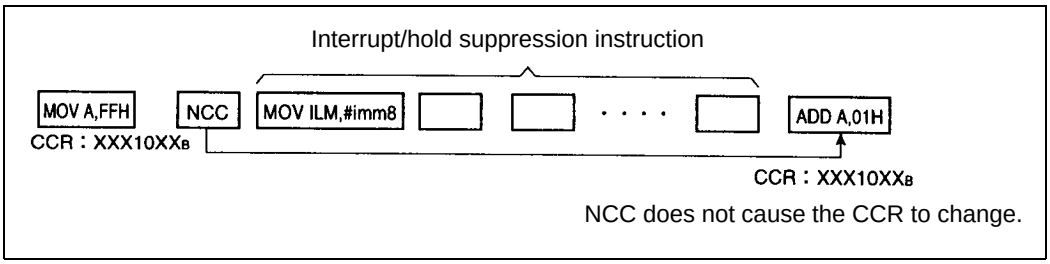


Figure 2.9-2 Interrupt/hold suppression instructions and prefix codes

■ **Consecutive prefix codes**

As shown in Figure 2.9-3, when consecutive conflicting prefix codes (PCB, ADB, DTB, and SPB) are specified, the last prefix code is valid.

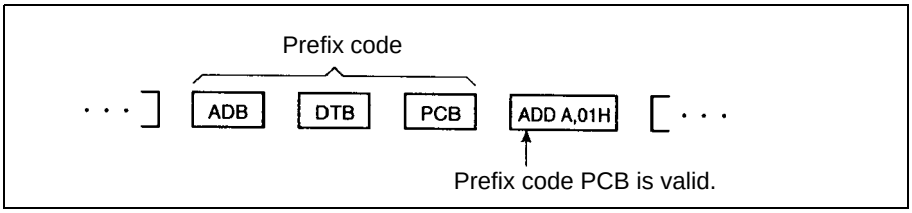


Figure 2.9-3 Consecutive prefix codes

CHAPTER 3 RESETS

This chapter describes resets for the MB90560-series microcontrollers.

3.1 Resets

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

- **Power-on reset**
- **Watchdog timer overflow**
- **External reset request via the RSTX pin**
- **Software reset request**

■ Reset causes

Table 3.1-1 lists the 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 RST pin	Previous state retained	Previous state retained	No
Software	A "0" is written to the RST bit of the low power consumption mode control register (LPMCR).	Previous state retained	Previous state retained	No
Watchdog timer	Watchdog timer overflow	MCLK	Stop	Yes
Power-on	When the power is turned on	MCLK	Stop	Yes

MCLK: Main clock (oscillation clock divided by 2)

● External reset

An external reset is generated by the "L" level input to an external reset pin (RSTX pin). The minimum required period of the "L" level input to the RSTX pin is 16 machine cycles (16/ϕ). The oscillation stabilization wait interval is not required for external resets.

<Reference>

For external reset requests via the RSTX pin, if the reset cause is generated during a write operation (during the execution of a transfer instruction such as MOV), the CPU waits for the reset to be released after the instruction is completed. The normal write operation is therefore completed even though a reset is input concurrently.

Note, however, that waiting for the reset to be released may not start before the transfer of the contents of a counter specified by a string-processing instruction (such as MOVS) is completed.

- **Software reset**

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

- **Watchdog timer reset**

  A watchdog timer reset is generated by a watchdog timer overflow that occurs when a “0” is not written to the WTE bit of the watchdog timer control register (WDTC) within a given time after the watchdog timer is activated. The watch-dog reset will wait for oscillation stabilization watch interval. The oscillation stabilization wait interval can be set by the clock selection register (CKSCR).

- **Power-on reset**

  A power-on reset is generated when the power is turned on. The oscillation stabilization wait interval is fixed at 2^{17} oscillation clock cycles (2^{17}/HCLK). After the oscillation stabilization wait interval has elapsed, the reset is executed.

  See also

  - Definition of clocks

    HCLK: Oscillation clock

    MCLK: Main clock

    ϕ: Machine clock (CPU operating clock)

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

  See Chapter 4, "Clocks," for details.

3.2 Reset Causes and Oscillation Stabilization Wait Intervals

The F²MC-16LX has five reset causes. The oscillation stabilization wait interval for a reset depends on the reset cause.

■ Reset causes and oscillation stabilization wait intervals

Tables 3.2-1 and 3.2-2 summarize 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	2^{17}/HCLK (32.768 ms)
Watchdog timer	2^{17}/HCLK (32.768 ms)
External reset via the RSTX pin	None
Software reset	None

HCLK:Oscillation clock frequency, source oscillation

Table 3.2-2 Oscillation stabilization wait intervals set by the clock selection register (CKSCR)

WS1	WS0	Oscillation stabilization wait interval The corresponding time interval for an oscillation clock frequency of 4 MHz is given in parentheses.
0	0	No oscillation stabilization time.
0	1	2^{13}/HCLK (2.048 ms)
1	0	2^{15}/HCLK (8.192 ms)
1	1	2^{17}/HCLK (32.768 ms)

HCLK: Oscillation clock frequency

<Check>

Ceramic and crystal oscillators generally require an oscillation stabilization wait interval of several milliseconds to 10 to 20 milliseconds until they stabilize at their natural frequency. Be sure to set a proper oscillation stabilization wait interval for the particular oscillator used.

See Chapter 4, "Clocks," for details.

■ Oscillation stabilization wait and reset state

A reset operation in response to a power-on reset and other externally activated resets during stop mode is performed after the oscillation stabilization wait interval has elapsed. This time interval is generated by the timebase timer. If the external reset has not been released after the interval, the reset operation is performed after the external reset is released.

Memo

3.3 External Reset Pin

The external reset pin (RST pin) is a dedicated pin for inputting, with an L level signal, a reset and generating an internal reset by the L level input.
In MB90560 series microcontrollers, there are internal CPU synchronous reset and external synchronous reset.

■ Block diagrams of the external reset pin

● Block diagram of internal reset

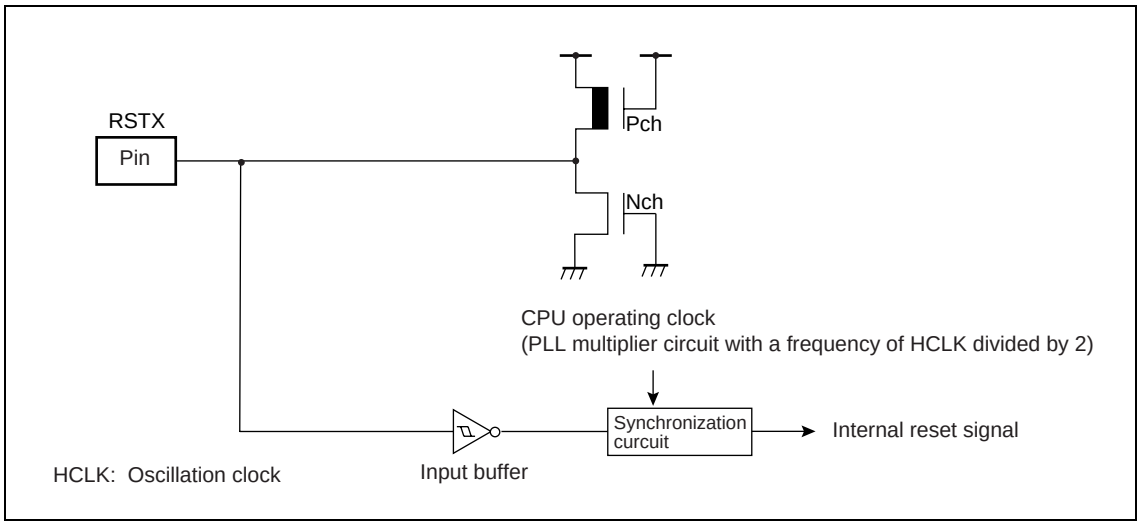


Figure 3.3-1 Block diagram of internal reset

<Check>

Inputs to the RSTX pin are accepted during cycles in which memory is not affected to prevent memory from being destroyed by a reset during a write operation.

A clock is required to initialize the internal circuit. In particular, an operation with an external clock requires clock input together with reset input.

● **Block diagram of internal reset for external pin**

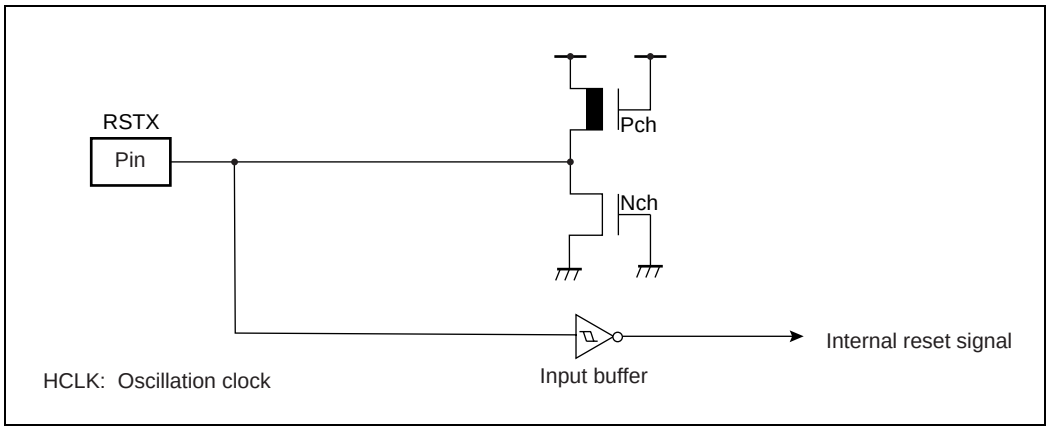


Figure 3.3-2 Block diagram of internal reset for external pin

3.4 Reset Operation

When a reset is released, the memory locations from which the mode data and the reset vector are read are selected according to the setting of the mode pins, and the mode setting data is fetched. Mode setting data determines the CPU operating mode and the execution start address after a reset operation ends.
For power-on or recovery from stop mode by a reset, the mode is fetched after the oscillation stabilization wait time has elapsed.

■ Overview of reset operation

Figure 3.4-1 shows the reset operation flow.

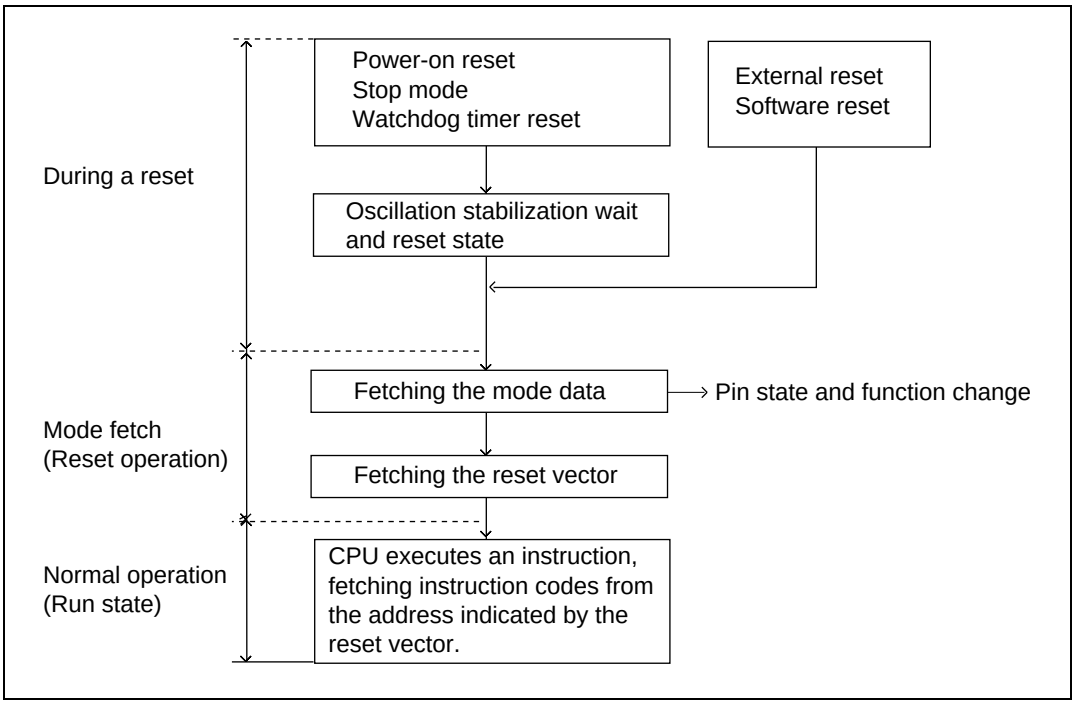


Figure 3.4-1 Reset operation flow

■ Mode pins

Setting the mode pins (MD0 to MD2) specifies how to fetch the reset vector and the mode data. Fetching the reset vector and the mode data is performed in the reset sequence. See Chapter 7, "Setting a Mode," for details about mode pins.

■ Mode fetch

When the reset is cleared, the CPU transfers the reset vector and the mode data stored in the hardware memory to the appropriate registers in the CPU core. The reset vector and mode data are allocated to the four bytes from FFFFDC$_H$ to FFFFDF$_H$. The CPU outputs these addresses to the bus immediately after the reset is cleared and fetches the reset vector and mode data. Using mode fetching, the CPU can begin processing at the address indicated by the reset vector.

Figure 3.4-2 shows the transfer of the reset vector and mode data.

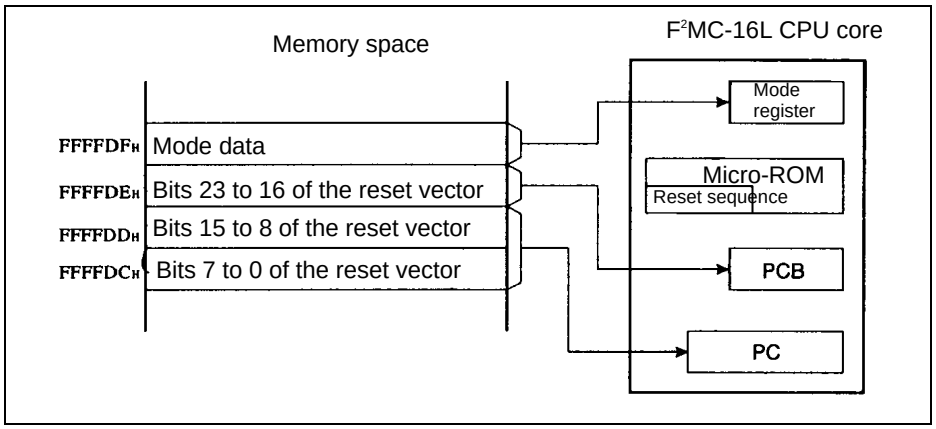


Figure 3.4-2 Transfer of reset vector and mode data

- **Mode data (address: FFFFDF$_H$)**

  Only s reset operation changes the contents of the mode register. The mode register setting is valid after a reset operation. See Section 7.1, "Setting a Mode," for details about mode data.

- **Reset vector (address: FFFFDC$_H$ to FFFFDE$_H$)**

  The execution start address after the reset operation ends is written as the reset vector. Execution starts at the address contained in the reset vector.

3.5 Reset Cause Bits

A reset cause can be identified by checking the watchdog timer control register (WDTC).

■ Reset cause bits

As shown in Figure 3.5-1, a flip-flop is associated with each reset cause. The contents of the flip-flops are obtained by reading the watchdog timer control register (WDTC). If it is necessary to identify the cause of a reset after the reset has been released, the value read from the WDTC should be processed by the software and a branch made to the appropriate program.

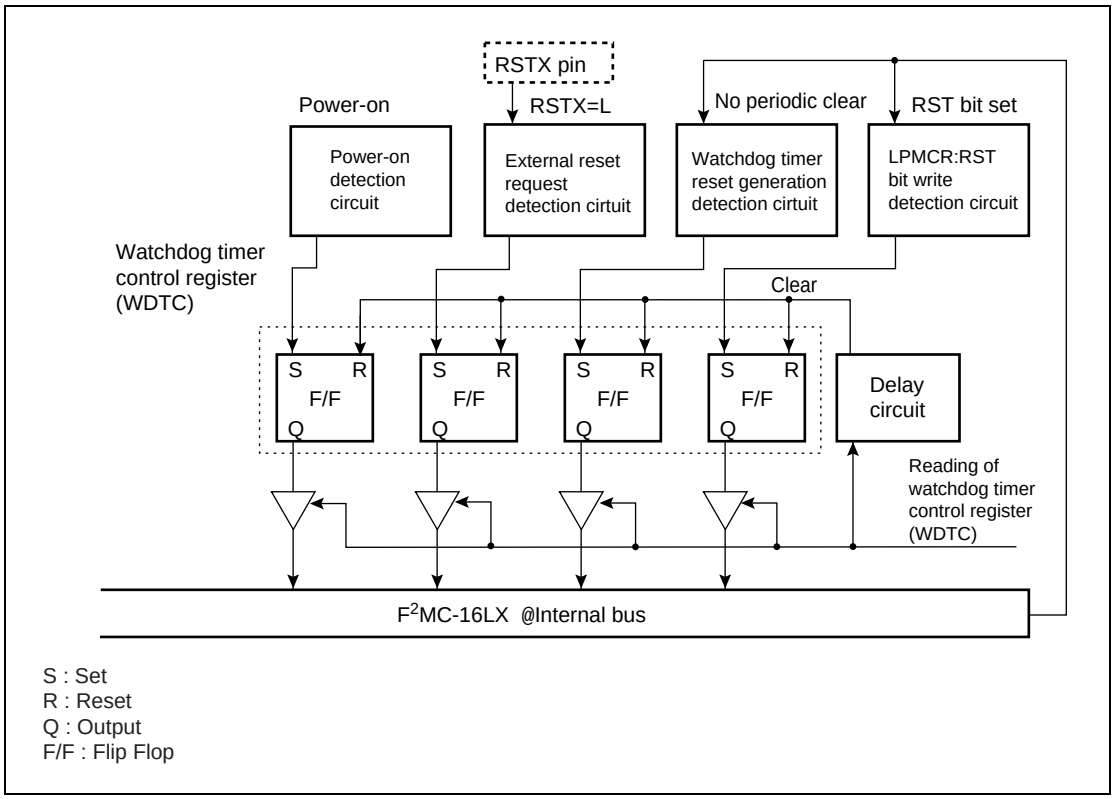


Figure 3.5-1 Block diagram of reset cause bits

■ Correspondence between reset cause bits and reset causes

Figure 3.5-2 shows the configuration of the reset cause bits of the watchdog timer control register (WDTC). Table 3.5-1 maps the correspondence between the reset cause bits and reset causes.

Watchdog timer control register (WDTC)

Address	bit15 ····· bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000A8H	(TBTC)	PONR	-	WRST	ERST	SRST	WTE	WT1	WT0	1XXXXXXXB
		R	-	R	R	R	W	W	W	

R : Read only
W : Write only
X : Undefined

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 RST 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 generated at the same time, the corresponding reset cause bits of the watchdog timer control register (WDTC) are set to “1”.

If, for example, an external reset request via the RSTX pin and the watchdog timer overflow occur at the same time, both the ERST bit and the WRST bit are set to “1”.

● Power-on reset

For a power-on reset, the PONR bit is set to “1”, but all other reset cause bits are undefined.

Consequently, program the software so that it will ignore all reset cause bits except the PONR bit if it is “1”.

● Clearing the reset cause bits

The reset cause bits are cleared only when the watchdog timer control register (WDTC) is read. Once a reset is occurred, the corresponding reset cause bit remains to “1”, even though another reset cause is occurred.

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**

The status of pins during a reset depends on the settings of mode pins (MD2 to MD0 = "011").

● **When internal vector mode has been set:**

All I/O pins (resource pins) are high impedance, and mode data is read from the internal ROM.

■ **Status of pins after mode data is read**

The status of pins after mode data has been read depends on the mode data (M1 and M0 = 00).

● **When single-chip mode has been selected (M1, M0 = 00_B):**

All I/O pins (resource pins) are high impedance, and mode data is read from the internal ROM.

<Check>

For those pins that change to high impedance when a reset cause is generated, take care that devices connected to them do not malfunction.

<Reference>

See Appendix, "State of Pins for the MB90560 Series," for information about the state of pins during a reset.

Memo

CHAPTER 4 CLOCKS

This chapter describes the clocks used by MB90560 series microcontrollers.

4.1 Clocks

The clock generator controls the operation of the internal clock for the CPU and peripheral functions. This internal clock is called the machine clock. One internal clock cycle is regarded as one machine cycle.
Other clocks include a clock generated by source oscillation, called an oscillation clock, and a clock generated by the internal PLL oscillation, called a PLL clock.

■ Clocks

The clock generator block contains the oscillation circuit that generates the oscillation clock. An external oscillator is connected to this circuit. The oscillation clock can also be supplied by inputting an external clock to the clock generator.

The clock generator also contains the PLL clock multiplier circuit, which generates four types of clock, that are multiples of the oscillation clock.

The clock generator controls the oscillation stabilization wait interval and PLL clock multiplication as well as controls internal clock operation by changing the clock with a clock selector.

● Oscillation clock (HCLK)

The oscillation clock is generated either from an external oscillator connected to the oscillation circuit or by input of an external clock.

● Main clock (MCLK)

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 with the internal PLL clock multiplier circuit (PLL oscillation circuit). Selection can be made from among four different PLL clocks.

● Machine clock (ϕ)

The machine clock controls operation of the operation of the CPU and peripheral functions. One clock cycle is regarded as one machine cycle ($1/\phi$). An operating machine clock can be selected from among the main clock that is generated from the oscillation clock divided by 2 and the four types of clock, that are multiples of the source clock frequency.

<Check>

Although an oscillation clock of 3 MHz to 32 MHz can be generated when the operating voltage is 5 V, the maximum operating frequency for the CPU and peripheral functions is 16 MHz. If a frequency multiplier rate exceeding the operating frequency is specified, devices will not operate correctly.

If, for example, a source oscillation of 16 MHz is generated, only a multiplier of 1 can be specified.

<Reference>

A PLL oscillation of 3 to 16 MHz is possible, but this range depends on the operating voltage and multiplier. See "Data Sheet," for details.

■ Clock supply map

Since the machine clock generated by the clock generator is supplied as the operating clock for the CPU and peripheral functions, the operation of the CPU and peripheral functions is affected by switching of the main clock and the PLL clock (clock mode) and a change in the PLL clock multiplier.

Since the timebase timer output will supply the operating clock for some peripheral functions, a peripheral function can select the clock best suited for its operation.

Figure 4.1-1 shows the clock supply map.

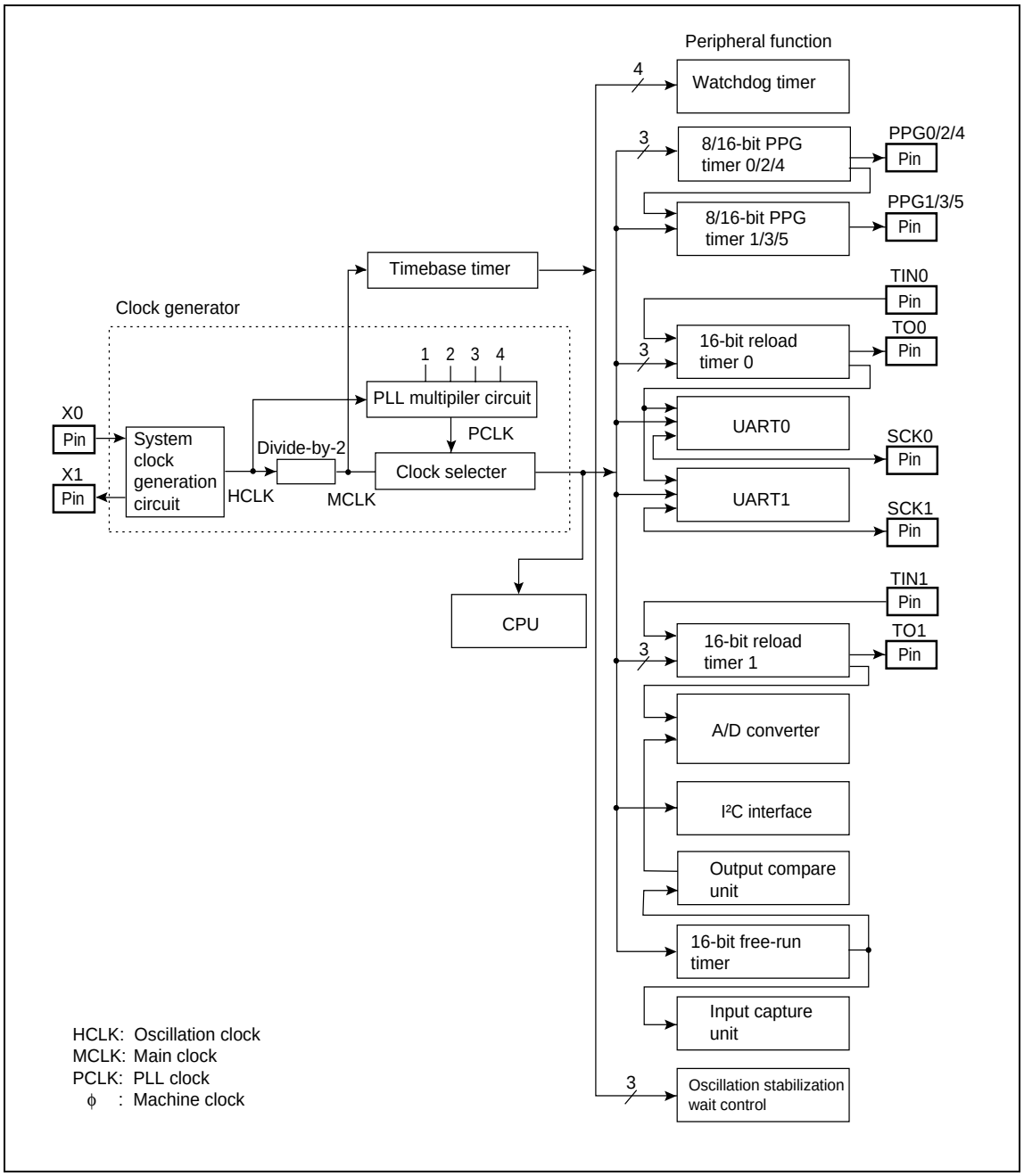


Figure 4.1-1 Clock supply map

4.2 Block Diagram of the Clock Generator

The clock generator consists of five blocks:

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

■ Block diagram of the clock generator

Figure 4.2-1 shows a block diagram of the clock generator and also includes the standby control circuit and timebase timer circuit.

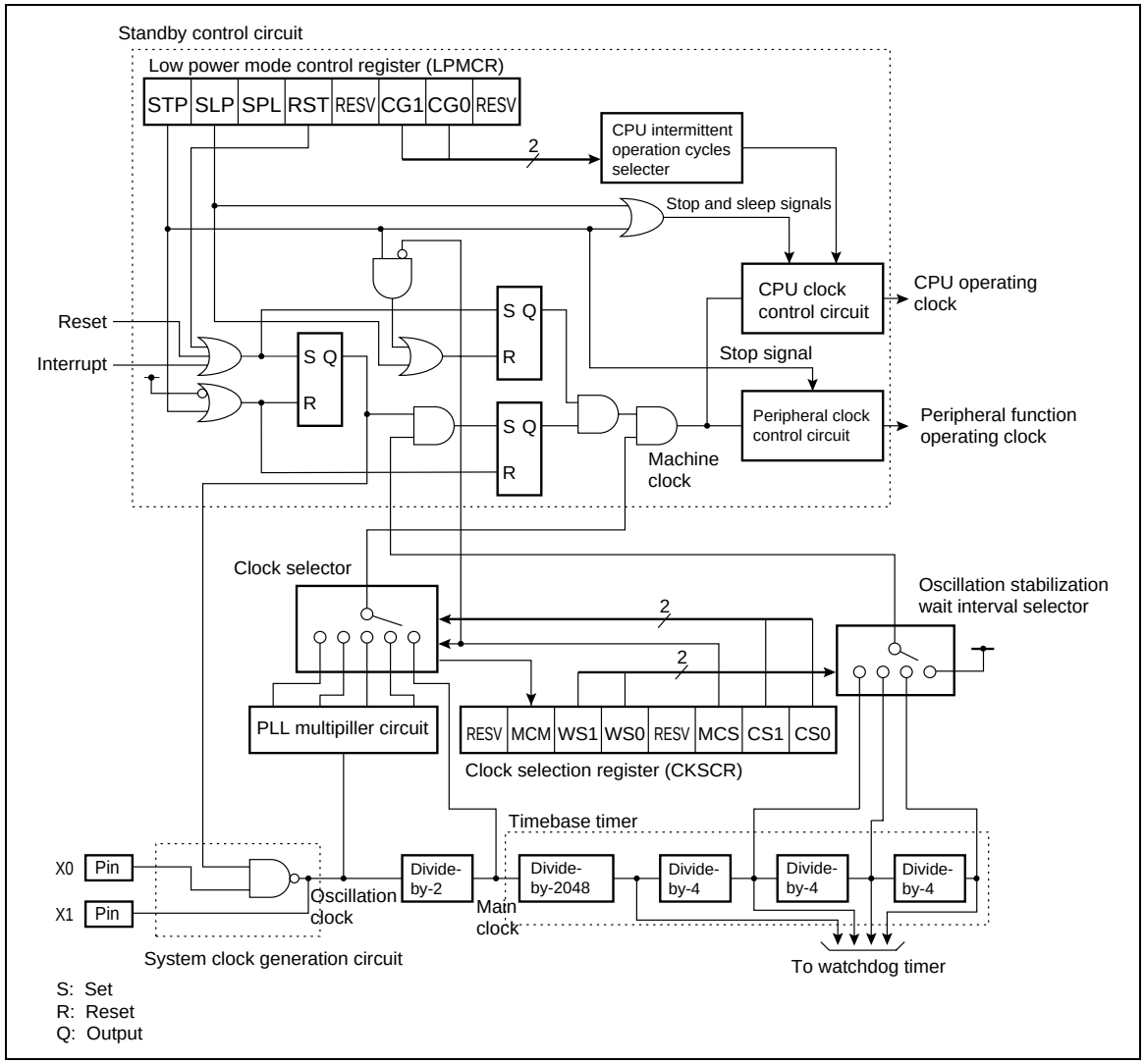


Figure 4.2-1 Block diagram of the clock generator

- **System clock generation circuit**

  The system clock generation circuit generates an oscillation clock (HCLK) from an external oscillator connected to it. Alternatively, an external clock can be input to this circuit.

- **PLL multiplier circuit**

  The PLL multiplier circuit multiplies the oscillation clock through PLL oscillation and supplies this multiplied clock to the CPU clock selector.

- **Clock selector**

  From among the main clock and four different PLL clocks, the clock selector selects the clock that is supplied to the CPU and peripheral clock control circuits.

- **Clock selection register (CKSCR)**

  The clock selection register is used to set switching between the oscillation clock and a PLL clock to select an oscillation stabilization wait interval, and to select a PLL clock multiplier.

- **Oscillation stabilization wait interval selector**

  This selector selects an oscillation stabilization wait interval for the oscillation clock when returning from stop mode or after a watchdog timer reset. Selection is made from among three kinds of timebase timer output. In all other cases, an oscillation stabilization wait interval is not selected.

4.3 Clock Selection Register (CKSCR)

The clock selection register (CKSCR) is used to switch between the main clock and a PLL clock, to select an oscillation stabilization wait interval, and to select a PLL clock multiplier.

■ Configuration of the clock selection register (CKSCR)

Figure 4.3-1 shows the configuration of the clock selection register (CKSCR); Table 4.3-1 describes the function of each bit of the clock selection register (CKSCR).

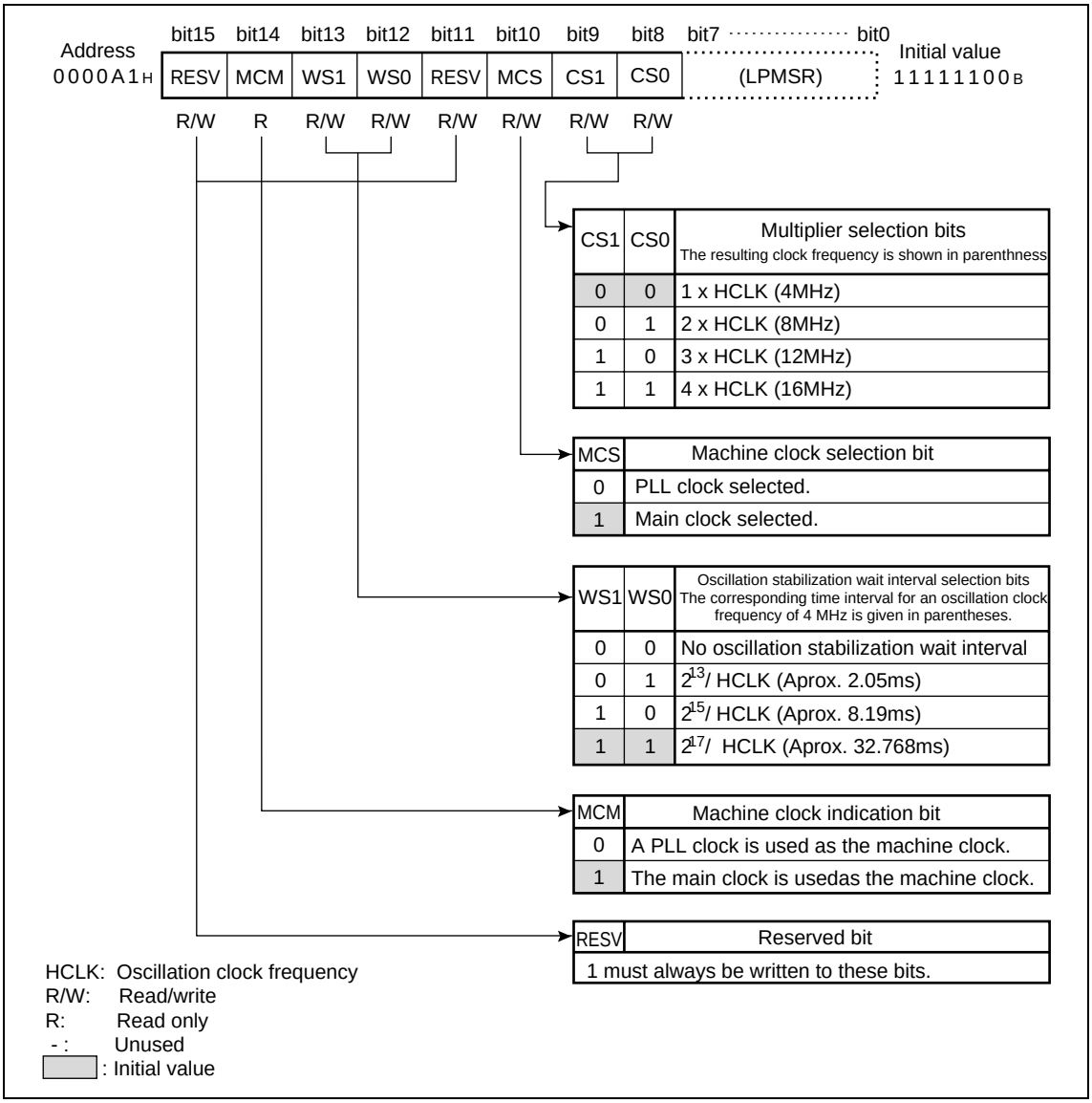


Figure 4.3-1 Configuration of the clock selection register (CKSCR)

<Check>

If the machine clock selection bit is not set, the main clock is used as the machine clock.

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	**<Caution>** • “1” must always be written to these bits.
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 it is set to “1”, the main clock has been selected. • If MCS = 0 and MCM = 1, the PLL clock oscillation stabilization wait period is in effect.
bit 13 bit 12	WS1, WS0: Oscillation stabilization wait interval selection bits	• These bits select an oscillation stabilization wait interval of the oscillation clock after stop mode has been cancelled. • These bits are initialized to 11_B by all reset causes. **<Caution>** The oscillation stabilization wait interval must be set to a value appropriate for the oscillator used. See Section 3.2, "Reset Causes and Oscillation Stabilization Wait Intervals," in Chapter 3. **<Reference>** The oscillation stabilization period for all PLL clocks is fixed at 2^{13}/HCLK.
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 “0”, a PLL clock is selected. When this bit is “1”, the main clock is selected. • “0” is written to this bit while it is “1”, the oscillation stabilization wait interval for the PLL clock starts. As a result, the timebase timer is automatically cleared, and the TBOF bit of the timebase timer control register (TBTC) is also cleared. • For PLL clocks, the oscillation stabilization period is fixed at 2^{13}/HCLK (the oscillation stabilization wait interval is approx. 2 ms for an oscillation clock frequency of 4 MHz). • When the main clock has been selected, the operating clock frequency is the frequency of the oscillation clock divided by 2 (e.g., the operating clock is 2 MHz when the oscillation clock frequency is 4 MHz). • This bit is initialized to 1 by all reset causes. **<Caution>** When the MCS bit is “1”, write “0” to it only when the timebase timer interrupt is masked by the TBIE bit of the timebase timer control register (TBTC) or the interrupt level register (ILM). For 8 machine cycles after “1” is written to the MCS bit, writing “0” to it may be disabled. Write to the bit after 8 machine cycles have passed.
bit 9 bit 8	CS1, CS0: Multiplier selection bits	• These bits select a PLL clock multiplier. • Selection can be made from among four different multipliers. • These bits are initialized to 00_B by all reset causes. **<Caution>** When the MCS bit is “0”, writing to these bits is not allowed. Write to the CS1 and CS0 bits only after setting the MCS bit to “1” (main clock mode).

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, which is the oscillation clock divided by 2 and is used as the operating clock for the CPU and peripheral resources. Meanwhile, the PLL clocks are disabled.

● PLL clock mode

In PLL clock mode, a PLL clock is used as the operating clock for the CPU and peripheral resources. A PLL clock multiplier is selected with the clock selection register (CKSCR: CS1 and CS0).

■ Clock mode transition

Switching between main clock mode and PLL clock mode is done by writing to the MCS bit of the clock selection register (CKSCR).

● Switching from main clock mode to PLL clock mode

When the MCS bit of CKSCR is “1”, writing “0” to it will switch the operating clock from the main clock to a PLL clock after the PLL clock oscillation stabilization wait period (2^{13}/HCLK).

● Switching from PLL clock mode to main clock mode

When the MCS bit of CKSCR is “0”, writing “1” to it will switch the operating clock from the PLL clock to the main clock when the edges of the PLL clock and the main clock coincide (after 1 to 8 PLL clocks).

<Check>

Even though the MCS bit of CKSCR is rewritten, machine clock switching does not occur immediately. When operating a peripheral function that depends on the machine clock, make sure that machine clock switching has been done by referring to the MCM bit of CKSCR before operating the peripheral function.

■ Selection of a PLL clock multiplier

Writing a value from “00_B” to “11_B” to the CS1 and CS0 bits of CKSCR selects one to the four PLL clock multipliers.

■ Machine clock

The machine clock may be either a PLL clock output from the PLL multiplier circuit or the clock that is the source oscillation clock divided by 2. This machine clock is supplied to the CPU and peripheral functions.

Either the main clock or a PLL clock can be selected by writing to the MCS bit of CKSCR.

Figure 4.4-1 shows the status change caused by the machine clock switching.

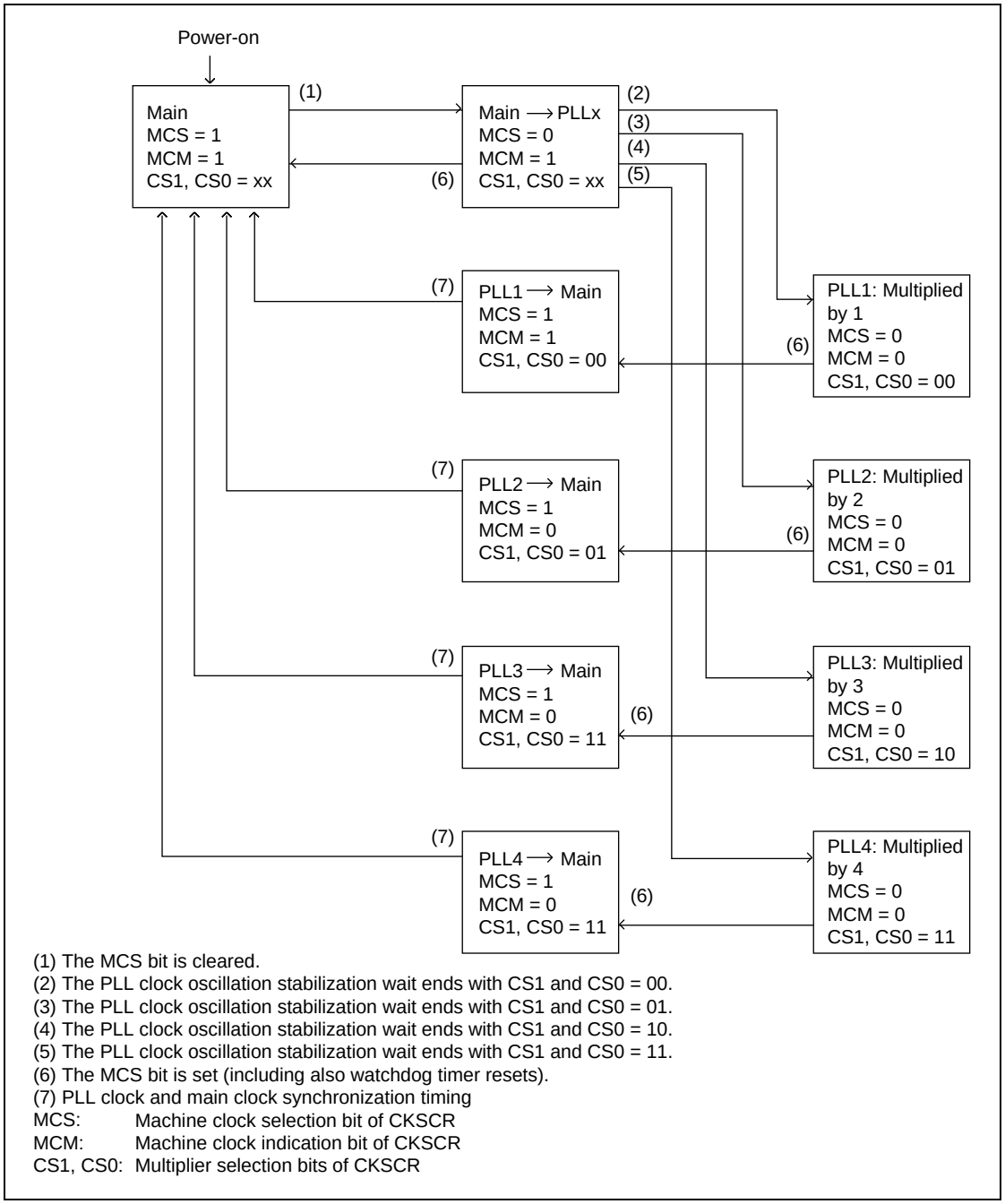


Figure 4.4-1 Status change diagram for machine clock selection

<Check>

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

4.5 Oscillation Stabilization Wait Interval

When the system changes operation from a state in which the main clock is stopped (such as at power-on, in stop mode and at watch-dog reset), a delay (an oscillation stabilization wait interval) is required to stabilize the clock oscillation before operation starts. When the switch from the main clock to a PLL clock occurs, an oscillation stabilization wait interval is also required when PLL oscillation starts.

■ Oscillation stabilization wait interval

Ceramic and crystal oscillators generally take several milliseconds to 20 milliseconds to stabilize at their natural frequency when oscillation starts.

For this reason, CPU operation is not allowed as soon as oscillation starts and is allowed only after full stabilization of oscillation. After the oscillation stabilization wait interval has elapsed, the clock is supplied to the CPU.

Because the oscillation stabilization time depends on the type of the oscillator (crystal, ceramic, etc.), the proper oscillation stabilization wait interval for the oscillator used must be selected. An oscillation stabilization wait interval is selected by setting the clock selection register (CKSCR).

In a switch from the main clock to a PLL clock, the CPU continues to operate on the main clock during the oscillation stabilization wait interval for PLL. After this interval, the operating clock switches to the PLL clock.

Figure 4.5-1 shows the operation after oscillation starts.

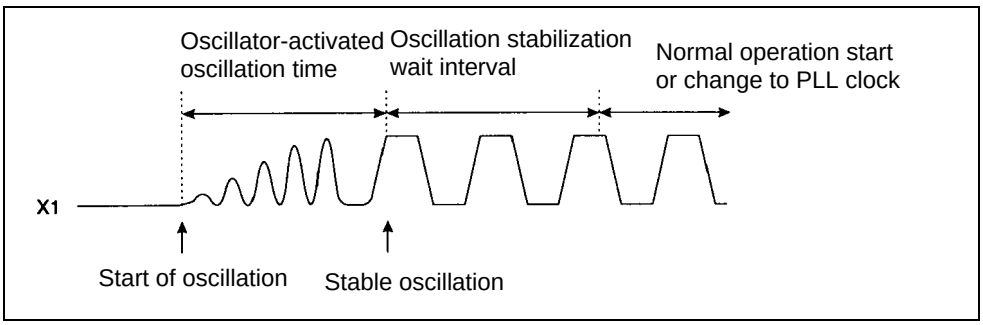


Figure 4.5-1 Operation when oscillation starts

4.6 Connection of an Oscillator or an External Clock

**The F²MC-16LX microcontroller contains a system clock generation circuit. Connecting an external oscillator to this circuit generates the system clock.
Alternatively, an externally generated clock can be input to the microcontroller.**

■ Connection of an oscillator 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.

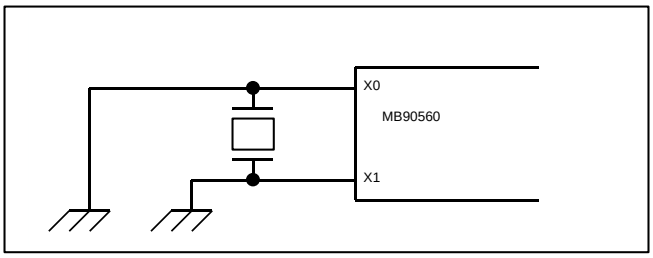


Figure 4.6-1 Example of connecting a crystal or ceramic oscillator to the microcontroller

● Example of connecting an external clock to the microcontroller

As shown in Figure 4.6-2, connect an external clock to pin X0. Pin X1 must be open.

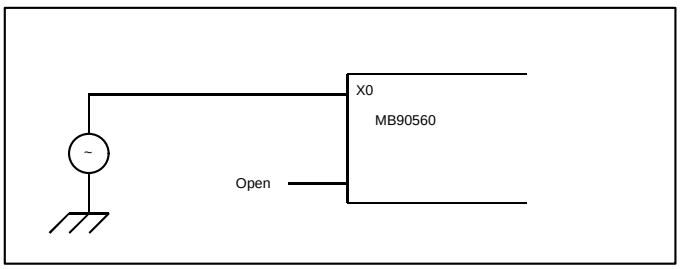


Figure 4.6-2 Example of connecting an external clock to the microcontroller

CHAPTER 5 LOW POWER CONSUMPTION MODE

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

5.1 Low Power Consumption Mode

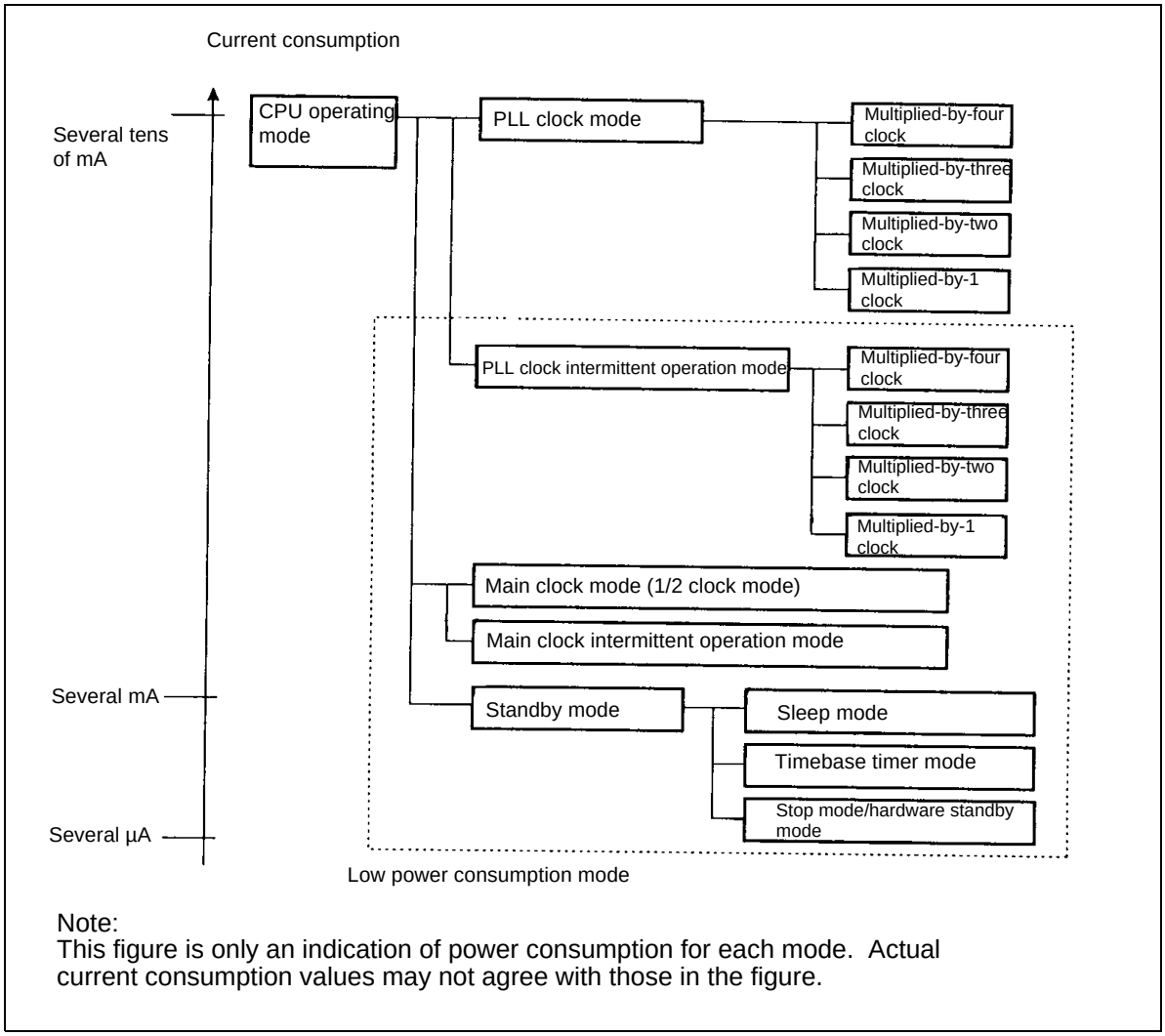
F²MC-16LX microcontrollers have the following CPU operating modes, any of which can be used depending on the operating clock selection and clock operation control:

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

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

■ **CPU operating modes and current consumption**

Figure 5.1-1 shows the relation between the CPU operating modes and current consumption



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

Figure 5.1-1 CPU operating modes and current consumption

■ Clock mode

● PLL clock mode

A PLL clock that is a multiple of the oscillation clock (HCLK) frequency is used to operate the CPU and peripheral functions.

● Main clock mode

The main clock, with a frequency one-half that of the oscillation clock (HCLK), is used to operate the CPU and peripheral functions. In main clock mode, the PLL multiplier circuit is inactive.

<Reference>

See Chapter 4, "Clocks," for details about clock mode.

■ CPU intermittent operation mode

CPU intermittent operation mode causes the CPU to operate intermittently, while high-speed clock pulses are supplied to peripheral functions, reducing power consumption. In CPU intermittent operation mode, intermittent clock pulses are only applied to the CPU when it is accessing a register, internal memory, a peripheral function, or an external unit.

■ Standby mode

In standby mode, the low power consumption control circuit stops supplying the clock to the CPU (sleep mode) or the CPU and peripheral functions (timebase timer mode), or stops the oscillation clock itself (stop mode), reducing power consumption.

● PLL sleep mode

PLL sleep mode is activated to stop the CPU operating clock when the microcontroller enters PLL clock mode; other components continue to operate on the PLL clock.

● Main sleep mode

Main sleep mode is activated to stop the CPU operating clock when the microcontroller enters main clock mode; other components continue to operate on the main clock.

● Timebase timer mode

Timebase timer mode causes microcontroller operation, with the exception of the oscillation clock and the timebase timer, to stop. All functions other than the timebase timer are deactivated.

● Stop mode

Stop mode causes the source oscillation to stop. All functions are deactivated.

<Check>

Because stop mode turns the oscillation clock off, this mode saves the most power while data is being retained.

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 mode control register (LPMCR)**

■ Block diagram of the low power consumption control circuit

Figure 5.2-1 shows the block diagram of the low power consumption control circuit.

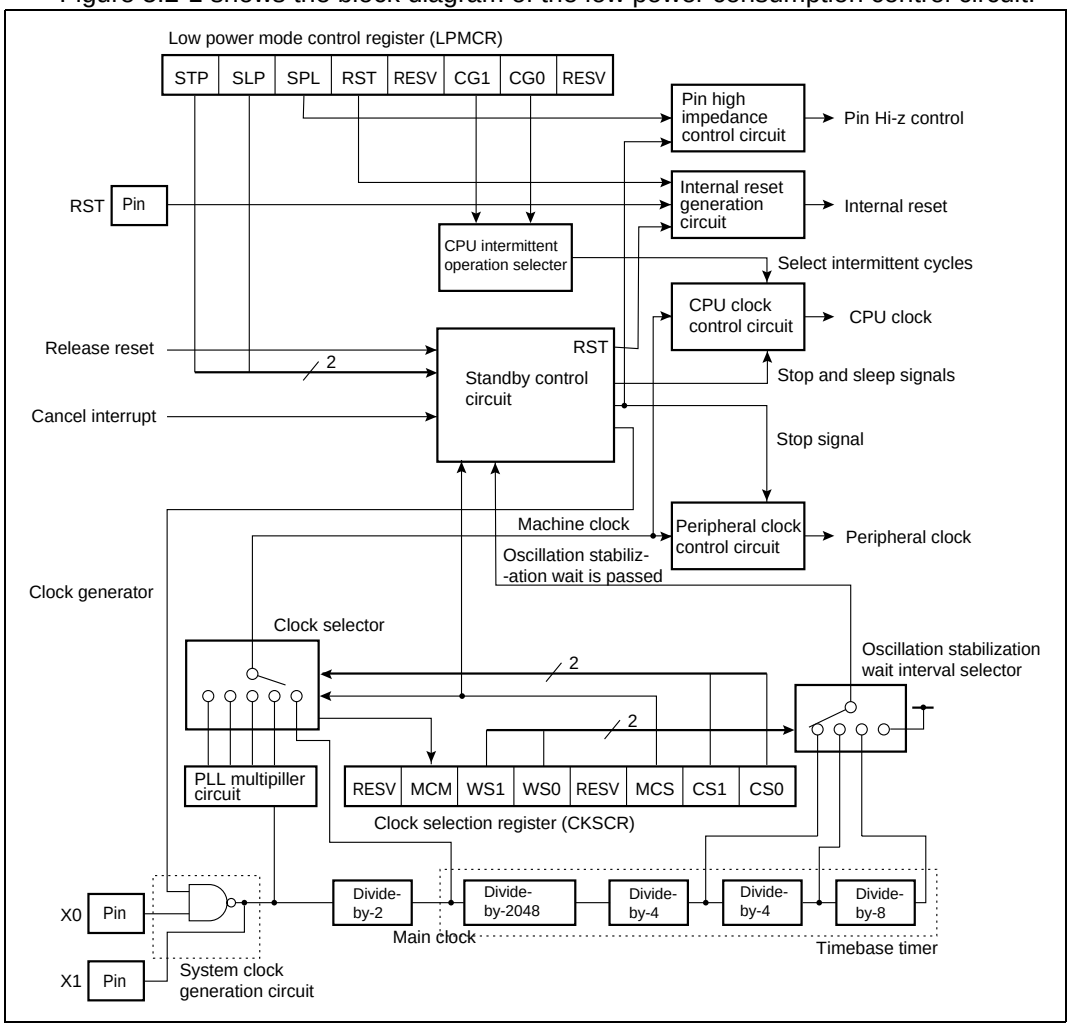


Figure 5.2-1 Block diagram of the low power consumption control circuit

- **CPU intermittent operation selector**

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

- **Standby control circuit**

  The standby control circuit controls the CPU clock control circuit and the peripheral clock control circuit, and turns the low power consumption mode on and off.

- **CPU clock control circuit**

  This circuit controls the clocks supplied to the CPU.

- **Peripheral clock control circuit**

  This circuit controls the clocks supplied to peripheral functions.

- **Pin high-impedance control circuit**

  This circuit makes the external pins high-impedance when the microcontroller enters timebase timer mode and stop mode.

  For the pins with the pull-up option, this circuit disconnects the pull-up resistor when the microcontroller enters stop mode or hardware standby mode.

- **Internal reset generation circuit**

  This circuit generates an internal reset signal.

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

  This register is used to switch to and return from standby mode and to set the CPU intermittent operation function.

5.3 Low Power Mode Control Register (LPMCR)

The low power mode control register (LPMCR) switches to or releases low power consumption mode. It is also used to set the number of CPU clock pulses the CPU is to be halted during CPU intermittent mode.

■ Low power consumption mode control register (LPMCR)

Figure 5.3-1 shows the configuration of the low power consumption mode control register (LPMCR).

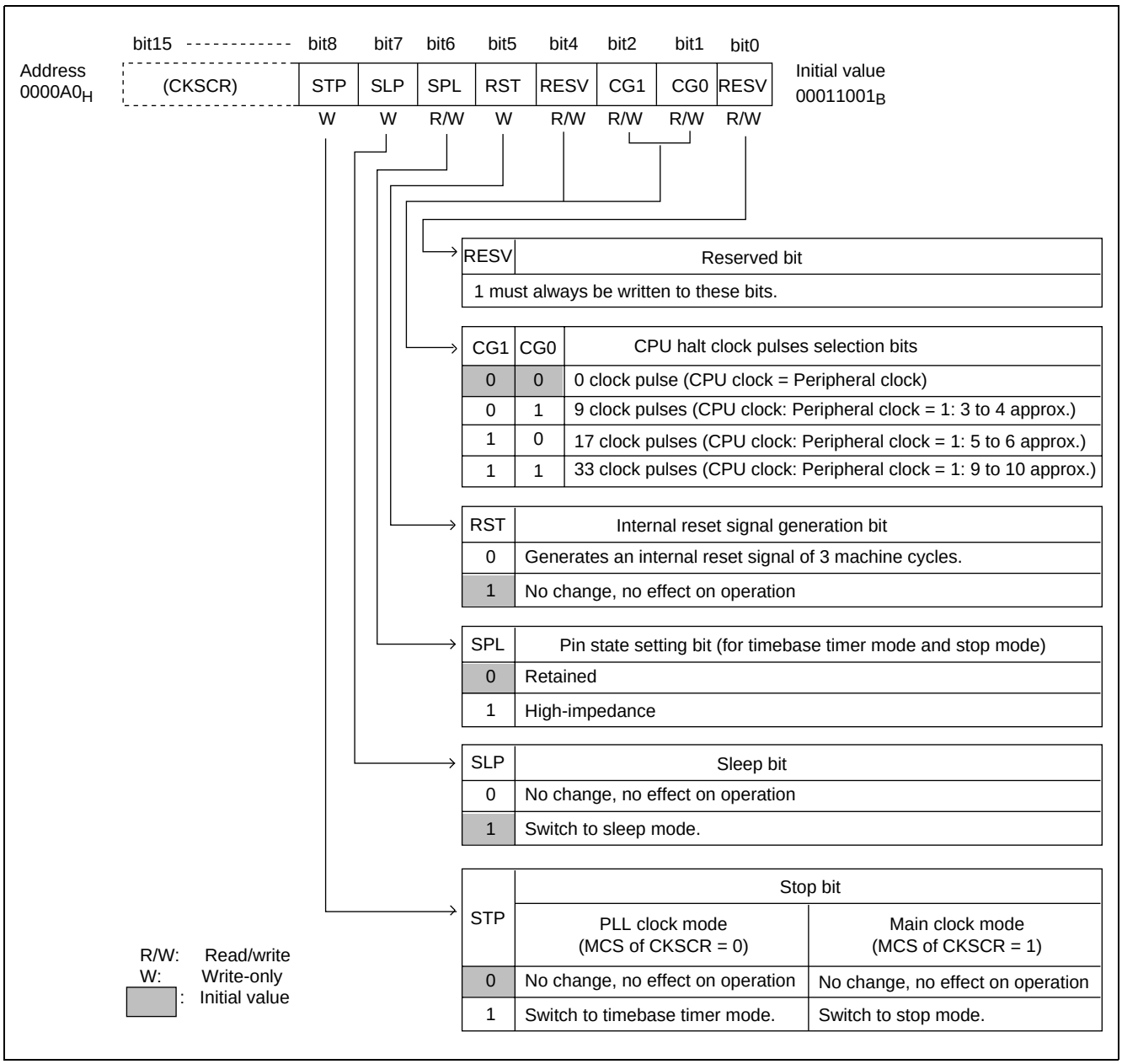


Figure 5.3-1 Configuration of the low power consumption mode control register (LPMCR)

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 indicates switching to timebase timer mode or stop mode. • When “1” is written to this bit, a switch to timebase timer mode is performed if the PLL clock has been selected (MCS of CKSCR = 0) and a switch to stop mode is performed if the oscillation clock has been selected (MCS of CKSCR = 1). Even though 1 is written to the STP bit during the transition of clock selection, the MCS bit status determines whether timebase timer mode or stop mode is used. • Writing “0” to this bit has no effect on operation. • This bit is cleared to “0” by a reset or by release of the timebase timer or of the stop state. • The read value of this bit is always “0”.
bit 6	SLP: Sleep bit	• This bit indicates switching to sleep mode. • When “1” is written to this bit, the mode switches to sleep mode. • Writing “0” to this bit has no effect on operation. • This bit is cleared to “0” by a reset or by release of sleep or stop mode. • If “1” is written to both the STP bit and SLP bit at the same time, the mode switches to timebase timer mode or stop mode. • The read value of this bit is always “0”.
bit 5	SPL: Pin state setting bit (for timebase timer mode and stop mode)	• This bit is enabled while either timebase timer mode or stop mode is in effect. • When this bit is “0”, the level of the external pins is retained. • When this bit is “1”, the status of the external pins changes to high-impedance. • This bit is initialized to “0” by a reset.
bit 4	RST: Internal reset signal generation bit	• When “0” is written to this bit, an internal reset signal of 3 machine cycles is generated. • Writing “1” to this bit has no effect on operation. • The read value of this bit is always “1”.
bit 3	RESV: Reserved bit	**<Caution>** “1” must always be written to this bit.
bit 2 bit 1	CG1, CG0: CPU halt clock pulses selection bits	• These bits set the number of CPU halt clock pulses for the CPU intermittent operation function. • The clock supplied to the CPU is stopped after the execution of every instruction for the specified number of clock pulses. • Selection can be made from among four different clock pulses. • These bits are initialized to “00_B“ by a power-on or watchdog timer reset. Other resets do not initialize these bits.
bit 0	RESV: Reserved bit	**<Caution>** “1” must always be written to this bit.

■ **Access to the low power mode control register (LPMCR)**

Switching to low power consumption mode (including stop mode and sleep mode) is performed by writing to the low power mode control register (LPMCR). Only the instructions listed in Table 5.3-2 should be used for this purpose. If other instructions are used for switching to low power consumption mode, operation cannot be assured. To control functions other than switching to low power consumption mode, any instruction can be used.

When word-length is used for writing to the low power consumption mode control register, even addresses must be used. Writing with odd addresses to switch to low power consumption mode may cause a malfunction.

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	MOV addr24,A		
MOVW io,#imm16	MOVW dir,#imm16	MOVW eam,#imm16	MOVW eam,RWi
MOVW io,A	MOVW dir,A	MOVW addr16,A	MOVW eam,AA
MOVW @RLi+disp8,A	MOVPW addr24,A		
SETB io:bp	SETB dir:bp	SETB addr16:bp	

Memo

5.4 CPU Intermittent Operation Mode

CPU intermittent operation mode is used for intermittent operation of the CPU while peripheral functions continue to operate at high speed. Its purpose is to reduce power consumption.

■ CPU intermittent operation mode

CPU intermittent operation mode halts the supply of the clock to the CPU for a certain period. The halt occurs after the execution of every instruction that accesses a register, internal memory (ROM and RAM), I/O, peripheral functions. and the external bus. Internal bus cycle activation is therefore delayed. While a steady rate of peripheral clock pulses are supplied to the peripheral functions, the rate of CPU execution is reduced, enabling processing with low power consumption.

- The CG1 and CG0 bits of the low power mode control register (LPMCR) are used to select the number of clock pulses per halt cycle of the clock supplied to the CPU.
- Instruction execution time in CPU intermittent mode can be calculated. A correction value should be obtained by multiplying the number of times instructions that access a register, internal memory and internal peripheral functions are executed by the number of clock pulses per halt cycle. Add this correction value to the normal execution time.

Figure 5.4-1 shows the operating clock pulses during CPU intermittent operation mode.

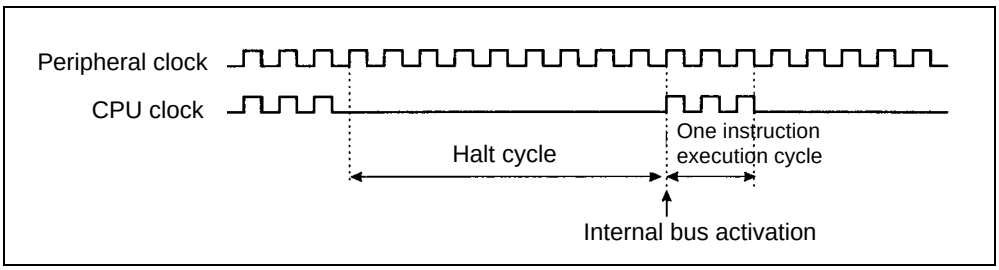


Figure 5.4-1 Clock pulses during CPU intermittent operation

5.5 Standby Mode

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

■ Operating status during standby mode

Table 5.5-1 summarizes the operating statuses during standby mode.

Table 5.5-1 Operation statuses during standby mode

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

*1 Only the timebase timer is active.

SPL:Pin state setting bit of low power consumption mode control register (LPMCR)

SLP:Sleep bit of LPMCR

STP:Timebase timer or stop bit of LPMCR

MCS:Machine clock selection bit of clock selection register (CKSCR)

Hi-z:High-impedance

5.5.1 Sleep mode

Sleep mode causes the CPU operating clock to stop while other peripheral functions continue to operate.
When the low power mode control register (LPMCR) indicates a switch to sleep mode, a switch to PLL sleep mode occurs if PLL clock mode has been set. Alternatively, a switch to main sleep mode occurs if main clock mode has been set.

■ Switching to sleep mode

Writing “1” to the SLP bit of LPMCR and “0” to the STP bit of LPMCR triggers a switch to sleep mode.

At this time, if the MCS bit of the clock selection register (CKSCR) is “0”, the microcontroller enters PLL sleep mode. If the MCS bit of CKSCR is “1”, the microcontroller enters main sleep mode.

<Check>

Since the STP bit setting overrides the SLP bit setting when “1” is written to the SLP and STP bits at the same time, the mode switches to timebase timer mode or stop mode.

● Data retention function

In sleep mode, the contents of dedicated registers, such as accumulators and internal RAM, are retained.

● Operation during an interrupt request

Writing “1” to the SLP bit of LPMCR during an interrupt request does not switch to sleep mode. If the CPU does not accept the interrupt, the CPU executes the next instruction. If the CPU accepts the interrupt, CPU operation immediately branches to the interrupt processing routine.

● Status of pins

During sleep mode, all pins retain the state they had immediately before the switch to sleep mode.

■ Release of sleep mode

The low power consumption control circuit is used to release sleep mode. Releasing is caused by the input of a reset or by an interrupt.

● Return to normal mode by a reset

When sleep mode is released by a reset, the microcontroller is placed in the reset state on release from sleep mode.

● **Return to normal mode by an interrupt**

If an interrupt request of level 7 or higher is issued from a peripheral circuit during sleep mode, sleep mode is released. After release, the CPU handles the interrupt in normal manner. The CPU executes processing according to the settings of the I flag of the condition code register (CCR), interrupt level mask register (ILM), and interrupt control register (ICR). If that interrupt is accepted, the CPU executes interrupt processing. If the interrupt is not accepted, the CPU resumes execution with the instruction that follows the instruction in which switching to sleep mode was specified.

Figure 5.5-1 shows the release of sleep mode for an interrupt.

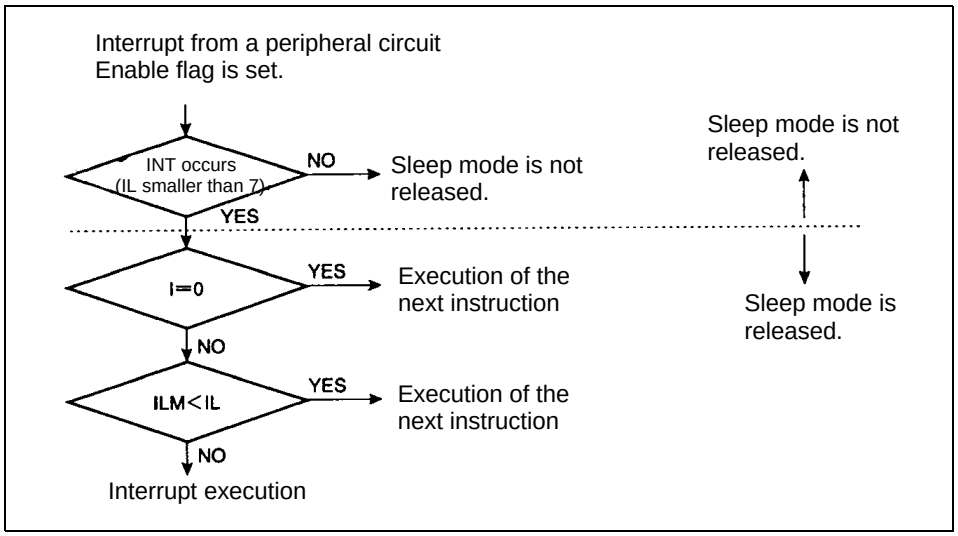


Figure 5.5-1 Release of sleep mode for an interrupt

<Check>

When interrupt processing is executed normally, the CPU first executes the instruction that follows the instruction in which switching to sleep mode was specified. The CPU then proceeds to interrupt processing.

● **Return to normal mode from PLL sleep mode by an external reset**

During PLL sleep mode, the main clock and the PLL clock generate clock pulses. Since an external reset does not initialize the MCS bit in the clock selection register (CKSCR) to 1, PLL clock mode remains selected (MCS of CKSCR = 0). On return from PLL sleep mode by an external reset, the CPU starts operation using the PLL clock immediately after PLL sleep mode is released as shown in Figure 5.5-2.

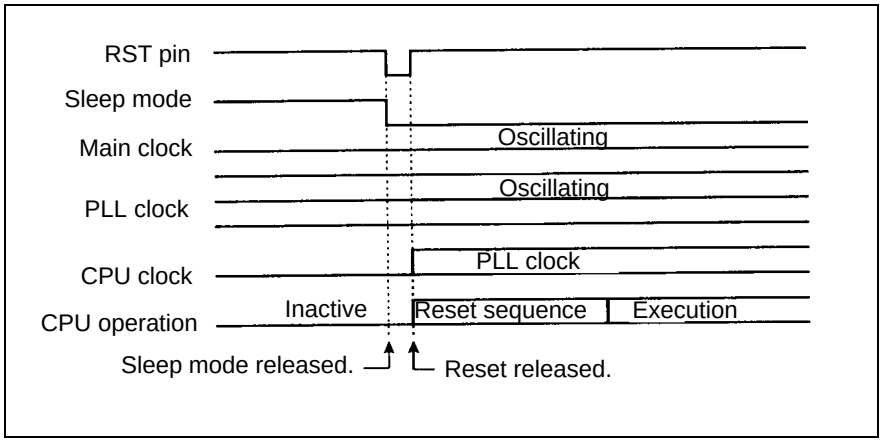


Figure 5.5-2 Release of PLL sleep mode (by external reset)

Memo

5.5.2 Timebase timer mode

Timebase timer mode causes the microcontroller operation to stop except the source oscillation and the timebase timer. All functions other than timebase timer are deactivated.

■ Switching to timebase timer mode

When “1” is written to the STP bit of the low power consumption mode control register (LPMCR) in PLL clock mode (MCS of CKSCR = 0), switching to timebase timer mode occurs. Also, in main clock mode (MCS of CKSCR = 1), writing “0” to the MCS bit of the clock selection register (CKSCR) and “1” to the STP bit of LPMCR triggers switching to timebase timer mode.

● Data retention function

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

● Operation during an interrupt

Writing “1” to the STP bit of LPMCR during an interrupt request does not trigger switching to timebase timer mode.

● Status of pins

Before switching to timebase timer mode, the SPL bit of LPMCR controls external I/O pins to either retain the pervious state or go to high-impedance.

■ Release of timebase timer mode

The low power consumption control circuit is used to release timebase timer mode. Release is caused by input of a reset or an interrupt. If timebase timer mode is released by a reset, the microcontroller is placed in the reset state after its release from timebase timer mode.

For return to normal mode from timebase timer mode, the low power consumption control circuit releases timebase timer mode. The microcontroller then enters the PLL clock oscillation stabilization wait state. If the PLL clock is not used, change the MCS bit of CKSCR to 1 with the instruction that is executed immediately after the reset or return from the interrupt.

● Return to normal mode by a reset

If timebase timer mode is released by a reset, the microcontroller is placed in the reset state after release from timebase timer mode.

● Return to normal mode by an external reset

Since an external reset does not initialize the MCS bit of the clock selection register (CKSCR) to 1, PLL clock mode remains selected (MCS of CKSCR = 0). If the reset period is shorter than the PLL clock oscillation stabilization wait period, the reset sequence proceeds using the main clock.

Figure 5.5-3 shows the operation for return to normal mode from timebase timer mode triggered by an external reset.

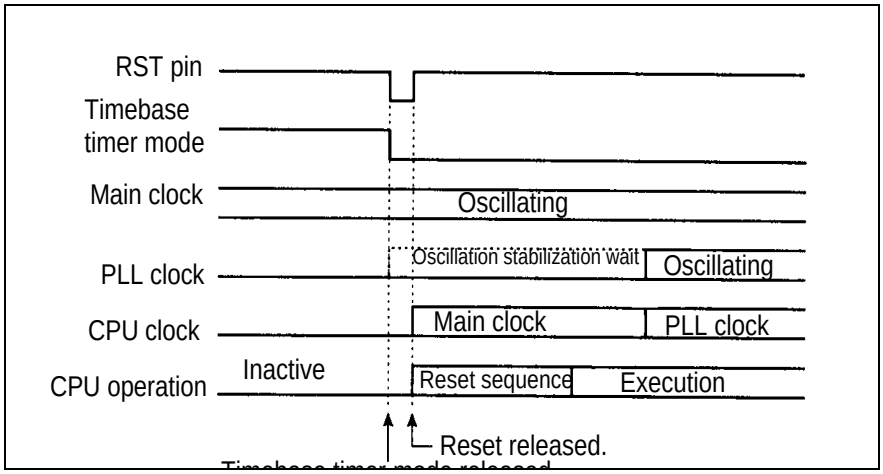


Figure 5.5-3 Release of timebase timer mode (by an external reset)

- **Return to normal mode by an interrupt**

  If an interrupt request of level 7 or higher is issued from a peripheral circuit in timebase timer mode (when IL2, IL1, and IL0 of the interrupt control register (ICR) are set to a value other than "111_B"), the low power consumption control circuit releases timebase timer mode. After the release, the CPU handles the interrupt in a normal manner. The CPU executes processing according to the settings of the I flag of the condition code register (CCR), interrupt level mask register (ILM), and interrupt control register (ICR). If the interrupt is accepted, the CPU executes interrupt processing. If the interrupt is not accepted, the CPU resumes execution with the instruction that follows the instruction in which switching to timebase timer mode was specified.

  <Check>

  When interrupt processing is executed normally, the CPU first executes the instruction that follows the instruction in which switching to timebase timer mode was specified. The CPU then proceeds to interrupt processing.

- **PLL clock oscillation stabilization wait interval on release of timebase timer mode**

  The period from the falling edge to the rising edge of the timebase timer output (2^{13} x oscillation clock cycle) is allowed for the detection of the end of the PLL clock oscillation wait interval.

  When timebase timer mode is released by an external reset or an interrupt, the timebase timer itself is not cleared. Consequently, a delay of a maximum of one cycle (2 x 2^{13} x oscillation clock cycle) occurs until the falling edge of the timer output has actually been detected after timebase timer mode is released.

  Therefore, a certain amount of time (2^{13} x oscillation clock cycle to 3 x 2^{13} x oscillation clock cycle) is required before PLL clock operation begins after the release of timebase timer mode. The CPU and peripheral functions operate using the main clock until the switch to the PLL clock.

5.5.3 Stop mode

Stop mode causes the source oscillation to stop and deactivates all functions. It is therefore the most power saving mode while data is being retained.

■ Switching to stop mode

When “1” is written to the STP bit of the low power consumption mode control register (LPMCR) is written in main clock mode (MCS of CKSCR = 1), switching to stop mode occurs.

During PLL clock mode (MCS of CKSCR = 0), writing “1” to the MCS bit of the clock selection register (CKSCR) and writing “1” to the STP bit of LPMCR trigger switching to stop mode.

- **Data retention function**

  In stop mode, the contents of dedicated registers, such as accumulators and internal RAM, are retained.

- **Operation during an interrupt**

  Writing “1” to the STP bit of LPMCR during an interrupt request does not trigger switching to stop mode.

- **Pin state setting**

  Before switching to stop mode, the SPL bit of LPMCR controls the external pins to either retain the pervious state or go to high-impedance.

■ Release of stop mode

The low power consumption control circuit is used release stop mode. The release is caused by input of a reset or by an interrupt.

Because the oscillation of the operating clock is stopped before return to normal mode from stop mode, the low power consumption control circuit puts the microcontroller into the oscillation stabilization wait state, then releases stop mode.

● Return to normal mode by a reset

When stop mode is released by a reset cause, the microcontroller is placed in the oscillation stabilization wait and reset state after release from stop mode. The reset sequence proceeds after the oscillation stabilization wait interval has elapsed.

● Return to normal mode by a interrupt

If an interrupt request of level 7 or higher is issued from a peripheral circuit during stop mode (when IL2, IL1, and IL0 of the interrupt control register (ICR) are set to a value other than "111B"), the low power consumption control circuit releases stop mode. After release, the CPU handles the interrupt in a normal manner. However, the CPU starts after the main clock oscillation stabilization wait interval specified by the WS1 and WS0 bits of the clock selection register (CKSCR) has elapsed. The CPU executes processing according to the settings of the I flag of the condition code register (CCR), interrupt level mask register (ILM), and interrupt control register (ICR). If the interrupt is accepted, the CPU executes interrupt processing. If the interrupt is not accepted, the CPU resumes the execution with the instruction that follows the instruction in which switching to stop mode was specified.

<Check>

When interrupt processing is executed normally, the CPU first executes the instruction that follows the instruction in which switching to stop mode was specified. The CPU then proceeds to interrupt processing.

Figure 5.5-4 shows the operation of return to normal mode from stop mode.

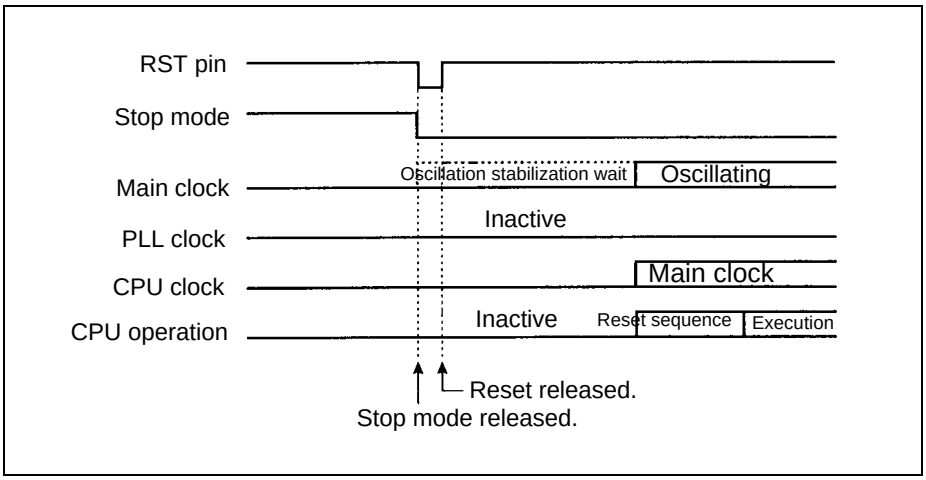


Figure 5.5-4 Release of stop mode (by external reset)

5.6 Status Change Diagram

Figure 5.6-1 shows the status change diagram of F²MC-16LX operation and gives change conditions.

■ **Status change diagram**

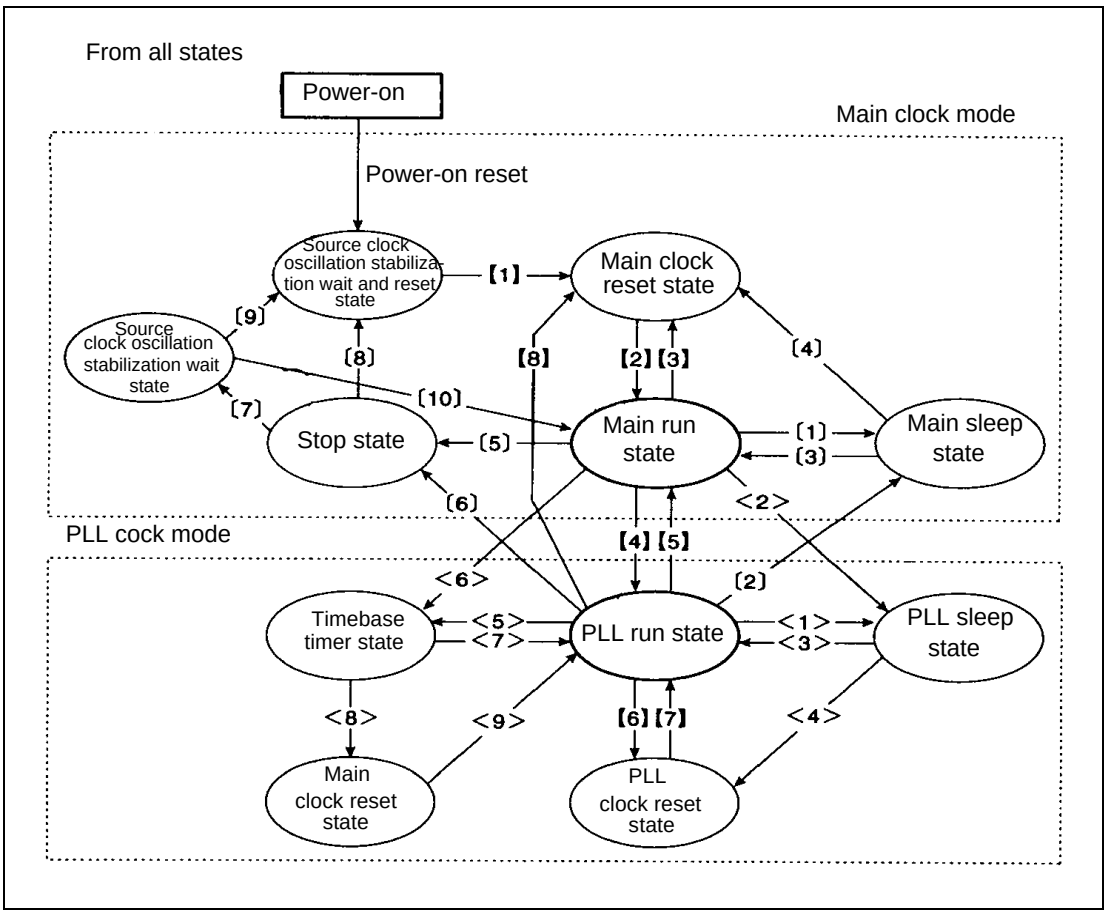


Figure 5.6-1 Status change diagram

■ Low power consumption mode operating states

Table 5.6-1 lists the operating states of low power consumption mode.

Table 5.6-1 Low power consumption mode operating states

Low power consumption mode	Condition for transition	Oscil-lation	Clock	CPU	Peripheral	Pin	Release event
Main sleep	MCS = 1 SLP = 1	Active	Active	Inactive	Active	Active	Reset or interrupt
PLL sleep	MCS = 0 SLP = 1	Active	Active	Inactive	Active	Active	Reset or interrupt
Timebase timer (SPL = 0)	MCS = 0 STP = 1	Active	Inactive	Inactive	Inactive	Hold	Reset or interrupt
Timebase timer (SPL = 1)	MCS = 0 STP = 1	Active	Inactive	Inactive	Inactive	Hi-z	Reset or interrupt
Stop (SPL = 0)	MCS = 1 STP = 1	Inactive	Inactive	Inactive	Inactive	Hold	Reset or interrupt
Stop (SPL = 1)	MCS = 1 STP = 1	Inactive	Inactive	Inactive	Inactive	Hi-z	Reset or interrupt

● Clock mode switching and release (excluding standby mode)

Table 5.6-2 lists clock mode switching and release.

Table 5.6-2 Clock mode switching and release

Transition	Conditions
After power-on, transition to the main run state	[1] Source clock oscillation stabilization wait interval ends. (Timebase timer output) [2] Reset input has been cleared.
Reset during main run state	[3] External reset, software reset, or watchdog timer reset
Transition from main run state to PLL run state	[4] MCS = 0 (After PLL clock oscillation stabilization wait, switch to PLL clock) (*1)
Return to main run state from PLL run state	[5] MCS = 1 (PLL clock deactivated)
Reset during PLL run state	[6] External reset or software reset ([7] After reset, return to PLL run state) [8] Watch dog reset ([3] After reset, return to main run state)

*1The microcontroller operates using the main clock during the PLL clock oscillation stabilization wait state.

● **Switching to and release from standby mode**

Table 5.6-3 lists switching to and release from standby mode.

Table 5.6-3 Switching to and release of standby mode

Transition	Conditions
Transition to main sleep mode	[1] SLP = 1, MCS = 1 (Transition from main run state) [2] SLP =1, MCS = 1 (Transition from PLL run state)
Release of main sleep mode	[3] Interrupt input [4] External reset
Transition to stop mode	[5] STP =1, MCS = 1 (Transition from main run state) [6] STP =1, MCS = 1 (Transition from PLL run state)
Release of stop mode	[7] Interrupt input ([10] indicates return to main run state after oscillation stabilization wait.) [8] External reset ([9] indicates external reset during oscillation stabilization wait state.)
Transition to PLL sleep mode	<1>SLP = 1, MCS = 0 (Transition from PLL run state) <2>SLP = 1, MCS = 0 (Transition from main run state, switch to PLL clock after PLL clock oscillation stabilization wait) (*1)
Release of PLL sleep mode	<3>Interrupt input <4>External reset
Transition to timebase timer mode	<5>STP = 1, MCS = 0 (Transition from PLL run state) <6>STP = 1, MCS = 0 (Transition from main run state, switch to PLL clock after PLL clock oscillation stabilization wait) (*1)
Release of timebase timer mode	<7>Interrupt input(*1) <8>External reset (<9> After reset, return to PLL run state) (*1)

*1The microcontroller operates using the main clock during the PLL clock oscillation stabilization wait state.

5.7 Status of Pins in Standby Mode and Reset

The statues of pins in standby mode and reset are summarized below for each memory access mode.

■ **Software pull-up resistor**

For pins with a pull-up resistor selected by software, the pull-up resistor is disconnected during “L” level output.

■ **Status of pins in single-chip mode**

Table 5.7-1 lists the state of pins in single-chip mode.

Table 5.7-1 State of pins in single-chip mode

Pin name	Standby mode			Reset
	Sleep	Stop		
		SPL = 0	SPL = 1	
P00 to P07 P17 P20 to P27 P30 to P37 P40 to P47 P50 to P57 P60 to P62	The preceding status is retained. (*2)	The preceding status is retained. (*2)	Input shut off/output Hi-z (*3)	Input disabled/ output Hi-z (*1)
P10 to P16 P63		Input enabled (*1)		

*1"Input enabled" means that the input function is enabled. Select either the pull-up or the pull-down option. Alternatively, an external input is required. Pins used as output ports are the same as other ports.

*2"The preceding status is retained" means that the status of the pin output existing immediately before switching to this mode is retained. Note that input is disabled if the preceding status was input.

- "Status of the pin output is retained" means that the pin retains the value output from an operating internal peripheral unit or the value output from the port if the pin is used as a port.
- "Input disabled" means that the input to the pin is not accepted because the internal circuit is inactive, although operation of the input gate adjacent to the pin is enabled.

*3"Input shut off" indicates the state in which operation of the input gate adjacent to the pin is disabled. "Output Hi-z" means that the pin state is high-impedance because driving of the pin driving transistor is disabled.

5.8 Notes on Using Low Power Consumption Mode

Note the following six items to use low power consumption mode:

- **Switching to standby mode and interrupts**
- **Release of standby mode by an interrupt**
- **Setting of standby mode**
- **Release of stop mode**
- **Release of timebase timer mode**
- **Oscillation stabilization wait time**

■ Notes on standby mode

● Switching to standby mode and interrupts

During an interrupt request to the CPU from a peripheral function, the CPU ignores the STP and SLP bits of the low power consumption mode control register (LPMCR) even though “1” has been written to these bits. Thus, switching to any standby mode is disabled (even after processing the interrupt is completed, there is no switch to standby mode). If the interrupt level is 7 or a higher priority, this action does not depend on whether the interrupt request is accepted by the CPU.

However, during execution of interrupt processing by the CPU, if the interrupt request flag for the interrupt is cleared and no other interrupt requests have been issued, switching to standby mode can be done.

● Release of standby mode caused by an interrupt

If an interrupt request of interrupt level 7 or a higher priority is issued from a peripheral function during the sleep, timebase timer, or stop modes, the standby mode is released. This action does not depend on whether the CPU accepts that interrupt.

After the release of standby mode, normal interrupt processing is performed. The CPU branches to the interrupt handling routine provided that the priority of the interrupt request indicated by the interrupt level setting bits (IL2, IL1, and IL0 of ICR) is higher than the interrupt level mask register (ILM); and the interrupt enable flag (I) of the condition code register (CCR) is set to “1” (enabled). If the interrupt is not accepted, the CPU starts the execution with the instruction that follows the instruction in which switching to standby mode was specified.

When interrupt processing is executed normally, the CPU first executes the instruction that follows the instruction in which switching to standby mode was specified. The CPU then proceeds to interrupt processing. Depending on the condition when switching to standby mode was performed, however, the CPU may proceed to interrupt processing before executing the next instruction.

If the CPU should not branch to the interrupt processing routine immediately on return to normal mode from standby mode, action must be taken to disable interrupts before standby mode is set.

● Setting of standby mode

When “1” is written to the STP bit and SLP bit of LPMCR at the same time, switching to standby mode is performed. If the MCS bit of the clock selection register (CKSCR) is “0”, switching to timebase timer mode is performed; if this bit is 1, switching to stop mode is performed.

■ **Release of stop mode**

If an external interrupt is used to release stop mode, the input request level must be “H”. Do not use an “L” level request because it may cause a malfunction. Edge requests do not result in a return from the standby state in a mode in which the clock has stopped.

■ **Release of timebase timer mode**

When timebase timer mode is released, the microcontroller is placed in the PLL clock oscillation stabilization wait state. If the PLL clock is not used, change the MCS bit of the clock selection register (CKSCR) to “1” with the instruction that is to be executed immediately after a reset or on return from an interrupt.

If an external interrupt is used to release timebase timer mode, the input request level must be “H”. An “L” level request may cause a malfunction. Edge requests do not result in a return from the standby state in a mode in which the clock has stopped.

■ **Oscillation stabilization wait interval**

● **Source clock oscillation stabilization wait interval**

Because the oscillator for source oscillation is halted in stop mode, an oscillation stabilization wait interval is required. A time period selected by the WS1 and WS0 bits of CKSCR is used as the oscillation stabilization wait interval.

● **PLL clock oscillation stabilization wait interval**

The CPU may be working with the main clock and the PLL clock may be stopped. If the microcontroller will enter a mode in which the CPU and peripheral functions work with the PLL clock, the PLL clock initially enters the oscillation stabilization wait state. In this state, the CPU still operates using the main clock.

The PLL clock oscillation stabilization wait interval is fixed at 2^{13}/HCLK (HCLK: oscillation clock frequency).

However, this interval may range from 2^{3}/HCLK to 3 x 2^{13}/HCLK depending on the status of the timebase timer, if the timebase timer is not cleared before the PLL clock oscillation stabilization wait state is entered. (For example, return to the PLL run state from timebase timer mode occurs because of an external reset.)

CHAPTER 6 INTERRUPTS

This chapter explains the interrupts and extended intelligent I/O service (EI²OS) in the MB90560 series.

6.1 Interrupts

This chapter explains the interrupts and extended intelligent I/O service (EI^{2}OS) in the MB90560 series.

- **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 a user-defined interrupt processing program in response to an interrupt request from a peripheral function.

● Software interrupt

A software interrupt transfers control to a user-defined interrupt processing program triggered by the execution of a dedicated software interrupt instruction (such as the INT instruction).

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

The EI^{2}OS function automatically transfers data between a peripheral function and memory. Data transfer, which has ordinarily been executed by an interrupt processing program, can be handled like a direct memory access (DMA). When the specified number of data transfers has been terminated, the interrupt processing program is automatically executed.

An instruction from EI^{2}OS is a type of hardware interrupt.

● Exception processing

Exception processing is basically the same as an interrupt. When an exception event (execution of an undefined instruction) is detected on the instruction boundary, ordinary processing is interrupted and exception processing is performed. This is equivalent to software interrupt instruction INT10.

■ Interrupt operation

Figure 6.1-1 shows the activation and return processing for the four types of interrupt functions.

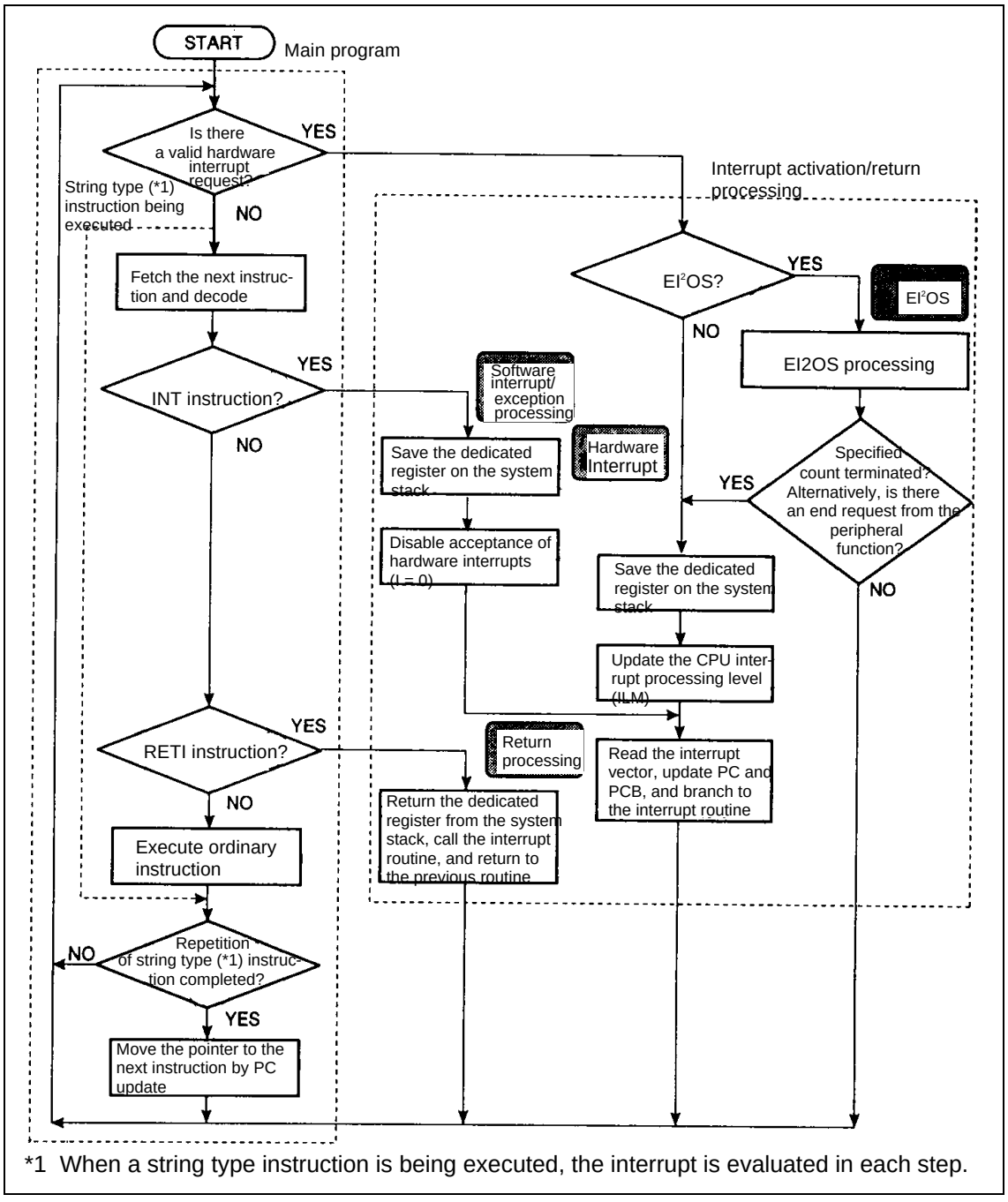


Figure 6.1-1 Overall flow of interrupt operation

6.2 Interrupt Causes and Interrupt Vectors

The F²MC-16LX can handle 256 types of interrupt causes. The 256 interrupt vector tables are allocated to the memory at the highest addresses. These interrupt vectors are shared by all interrupts.
Software interrupts can use all these interrupt vectors (INT0 to INT256). Software interrupts share some interrupt vectors with the hardware interrupts and exception processing interrupts. Hardware interrupts used a fixed interrupt vector and interrupt control register (ICR) for each peripheral function.

■ Interrupt vectors

Interrupt vector tables referenced during interrupt processing are allocated to the highest addresses in the memory area ("FFFC00H" to "FFFFFFH"). Interrupt vectors share the same area with EI²OS, exception processing, hardware, and software interrupts.

Table 6.2-1 shows the assignment of 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	FFFFFCH	FFFFFDH	FFFFFEH	Not used	#0	None
:	:	:	:	:	:	:
INT7	FFFFE0H	FFFFE1H	FFFFE2H	Not used	#7	None
INT8	FFFFDCH	FFFFDDH	FFFFDEH	FFFFDFH	#8	(RESET vector)
INT9	FFFFD8H	FFFFD9H	FFFFDAH	Not used	#9	None
INT10	FFFFD4H	FFFFD5H	FFFFD6H	Not used	#10	<Exception processing>
INT11	FFFFD0H	FFFFD1H	FFFFD2H	Not used	#11	Hardware interrupt #0
INT12	FFFFCCH	FFFFCDH	FFFFCEH	Not used	#12	Hardware interrupt #1
INT13	FFFFC8H	FFFFC9H	FFFFCAH	Not used	#13	Hardware interrupt #2
INT14	FFFFC4H	FFFFC5H	FFFFC6H	Not used	#14	Hardware interrupt #3
:	:	:	:	:	:	:
INT254	FFFC04H	FFFC05H	FFFC06H	Not used	#254	None
INT255	FFFC00H	FFFC01H	FFFC02H	Not used	#255	None

Reference

Unused interrupt vectors should be set as the exception processing address.

■ Interrupt causes and interrupt vectors/interrupt control registers

Table 6.2-2 shows the relationship between interrupt causes (excluding software interrupts), interrupt vectors, and interrupt control registers.

Table 6.2-2 Interrupt causes, interrupt vectors, and interrupt control registers

Interrupt cause	EI²OS support	Interrupt vector			Interrupt control register		Priority (*2)
		Number		Address	ICR	Address	
Reset	x	#08	08H	FFFFDCH	-	-	High
INT9 instruction	x	#09	09H	FFFFD8H	-	-	
Exception processing	x	#10	0AH	FFFFD4H	-	-	
A/D converter conversion termination	o	#11	0BH	FFFFD0H	ICR00	0000B0H	
Output compare channel 0 match	Δ	#13	0DH	FFFFC8H	ICR01	0000B1H(*1)	
8/16-bit PPG timer 0 counter borrow	Δ	#14	0EH	FFFFC4H			
Output compare channel 1 match	Δ	#15	0FH	FFFFC0H	ICR02	0000B2H(*1)	
8/16-bit PPG timer 1 counter borrow	Δ	#16	10H	FFFFBCH			
Output compare channel 2 match	Δ	#17	11H	FFFFB8H	ICR03	0000B3H(*1)	
8/16-bit PPG timer 2 counter borrow	Δ	#18	12H	FFFFB4H			
Output compare channel 3 match	Δ	#19	13H	FFFFB0H	ICR04	0000B4H(*1)	
8/16-bit PPG timer 3 counter borrow	Δ	#20	14H	FFFFACH			
Output compare channel 4 match	Δ	#21	15H	FFFFA8H	ICR05	0000B5H(*2)	
8/16-bit PPG timer 4 counter borrow	Δ	#22	16H	FFFFA4H			
Output compare channel 5 match	Δ	#23	17H	FFFFA0H	ICR06	0000B6H(*1)	
8/16-bit PPG timer 5 counter borrow	Δ	#24	18H	FFFF9CH			
DTP/external interrupt channels 0/1 detection	Δ	#25	19H	FFFF98H	ICR07	0000B7H(*1)	
DTP/external interrupt channels 2/3 detection	Δ	#26	1AH	FFFF94H			
DTP/external interrupt channels 4/5 detection	Δ	#27	1BH	FFFF90H	ICR08	0000B8H(*1)	
DTP/external interrupt channels 6/7 detection	Δ	#28	1CH	FFFF8CH			
8-bit timer 0/1/2 counter borrow	x	#29	1DH	FFFF88H	ICR09	0000B9H(*1)	
16-bit reload timer 0 underflow	o	#30	1EH	FFFF84H			
16-bit free-running timer overflow	x	#31	1FH	FFFF80H	ICR10	0000BAH(*1)	
16-bit reload timer 1 underflow	o	#32	20H	FFFF7CH			
Input capture channels 0/1	Δ	#33	21H	FFFF78H	ICR11	0000BBH(*1)	
16-bit free-running timer clear	x	#34	22H	FFFF74H			
Input capture channels 2/3	Δ	#35	23H	FFFF70H	ICR12	0000BCH(*1)	
Timebase timer	x	#36	24H	FFFF6CH			
UART1 receive	⊙	#37	25H	FFFF68H	ICR13	0000BDH(*1)	
UART1 send	Δ	#38	26H	FFFF64H			
UART0 receive	⊙	#39	27H	FFFF60H	ICR14	0000BEH(*1)	
UART0 send	Δ	#40	28H	FFFF5CH			
Flash memory status	x	#41	29H	FFFF58H	ICR15	0000BFH(*1)	
Delayed interrupt generator module	x	#42	2AH	FFFF54H			Low

o:Can be used.
x:Cannot be used.
⊙:Usable if used with the EI²OS stop function.
Δ:Usable when an interrupt cause that shares the ICR is not used.
*1- For peripheral functions that share the ICR register, the interrupt level will be the same.
- If the extended intelligent I/O service is to be used with a peripheral function that shares the ICR register with another peripheral function, the service can be used only for one of the functions.
- If the extended intelligent I/O service is specified for a peripheral function that shares the ICR register with another peripheral function, interrupts are disabled for the other peripheral function sharing the ICR register.*1
*2 This priority is applied when interrupts of the same level occur simultaneously.

6.3 Interrupt Control Registers and Peripheral Functions

Interrupt control registers (ICR00 to ICR15) are located inside the interrupt controller. The interrupt control registers 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 lists the interrupt control registers and corresponding peripheral functions.

Table 6.3-1 Interrupt control registers

Address	Register	Abbreviation	Corresponding peripheral function
0000B0H	Interrupt control register 00	ICR00	A/D converter
0000B1H	Interrupt control register 01	ICR01	Output compare 0, 8/16-bit PPG timer 0
0000B2H	Interrupt control register 02	ICR02	Output compare 1, 8/16-bit PPG timer 1
0000B3H	Interrupt control register 03	ICR03	Output compare 2, 8/16-bit PPG timer 2
0000B4H	Interrupt control register 04	ICR04	Output compare 3, 8/16-bit PPG timer 3
0000B5H	Interrupt control register 05	ICR05	Output compare 4, 8/16-bit PPG timer 4
0000B6H	Interrupt control register 06	ICR06	Output compare 5, 8/16-bit PPG timer 5
0000B7H	Interrupt control register 07	ICR07	DTP/external interrupts 0, 1, 2, 3
0000B8H	Interrupt control register 08	ICR08	DTP/external interrupts 4, 5, 6, 7
0000B9H	Interrupt control register 09	ICR09	8-bit timers 0, 1, 2, 16-bit reload timer 0
0000BAH	Interrupt control register 10	ICR10	16-bit free-run timer overflow, 16-bit reload timer 1
0000BBH	Interrupt control register 11	ICR11	Input capture 0, 1, 16-bit free-run timer clear
0000BCH	Interrupt control register 12	ICR12	Input capture 2, 3, timebase timer
0000BDH	Interrupt control register 13	ICR13	UART1
0000BEH	Interrupt control register 14	ICR14	UART0
0000BFH	Interrupt control register 15	ICR15	Flash memory, delayed interrupt generator module

■ Interrupt control register functions

All interrupt control registers (ICR) do the following:

- Set the interrupt level of the corresponding peripheral function.
- Select ordinary interrupts or the extended intelligent I/O service as interrupts of the corresponding peripheral function.
- Select an extended intelligent I/O service (EI^{2}OS) channel
- Display the status of the extended intelligent I/O service (EI^{2}OS)

Some of the functions of the interrupt control registers (ICR) differ during writing and reading, as shown in Figures 6.3-1 and 6.3-2.

<Check>

Do not use a read-modify-write instruction to access the interrupt control registers (ICR), since operation will not be correct.

Memo

6.3.1 Interrupt control registers (ICR00 to ICR15)

Interrupt control registers correspond to all peripheral functions that have the interrupt function. The interrupt control registers control the processing when an interrupt request occurs. The functions of these registers partially differ at writing and reading.

■ Interrupt control registers (ICR00 to ICR15)

During writing ICR

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
Address $0000B0_H$ to 0000BFH	ICS3	ICS2	ICS1	ICS0	ISE	IL2	IL1	IL0	Initial value 00000111_B
	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^2OS enable bit
0	Activates the interrupt sequence when an interrupt occurs
1	Activates EI^2OS when an interrupt occurs

ICS3	ICS2	ICS1	ICS0	EI^2OS channel selection bit	
				Channel	Descriptor address
0	0	0	0	0	000100_H
0	0	0	1	1	000108_H
0	0	1	0	2	000110_H
0	0	1	1	3	000118_H
0	1	0	0	4	000120_H
0	1	0	1	5	000128_H
0	1	1	0	6	000130_H
0	1	1	1	7	000138_H
1	0	0	0	8	000140_H
1	0	0	1	9	000148_H
1	0	1	0	10	000150_H
1	0	1	1	11	000158_H
1	1	0	0	12	000160_H
1	1	0	1	13	000168_H
1	1	1	0	14	000170_H
1	1	1	1	15	000178_H

R/W: Read/write
W: Write-only
: Initial value

Figure 6.3-1 Interrupt control registers (ICR00 to ICR15) during writing

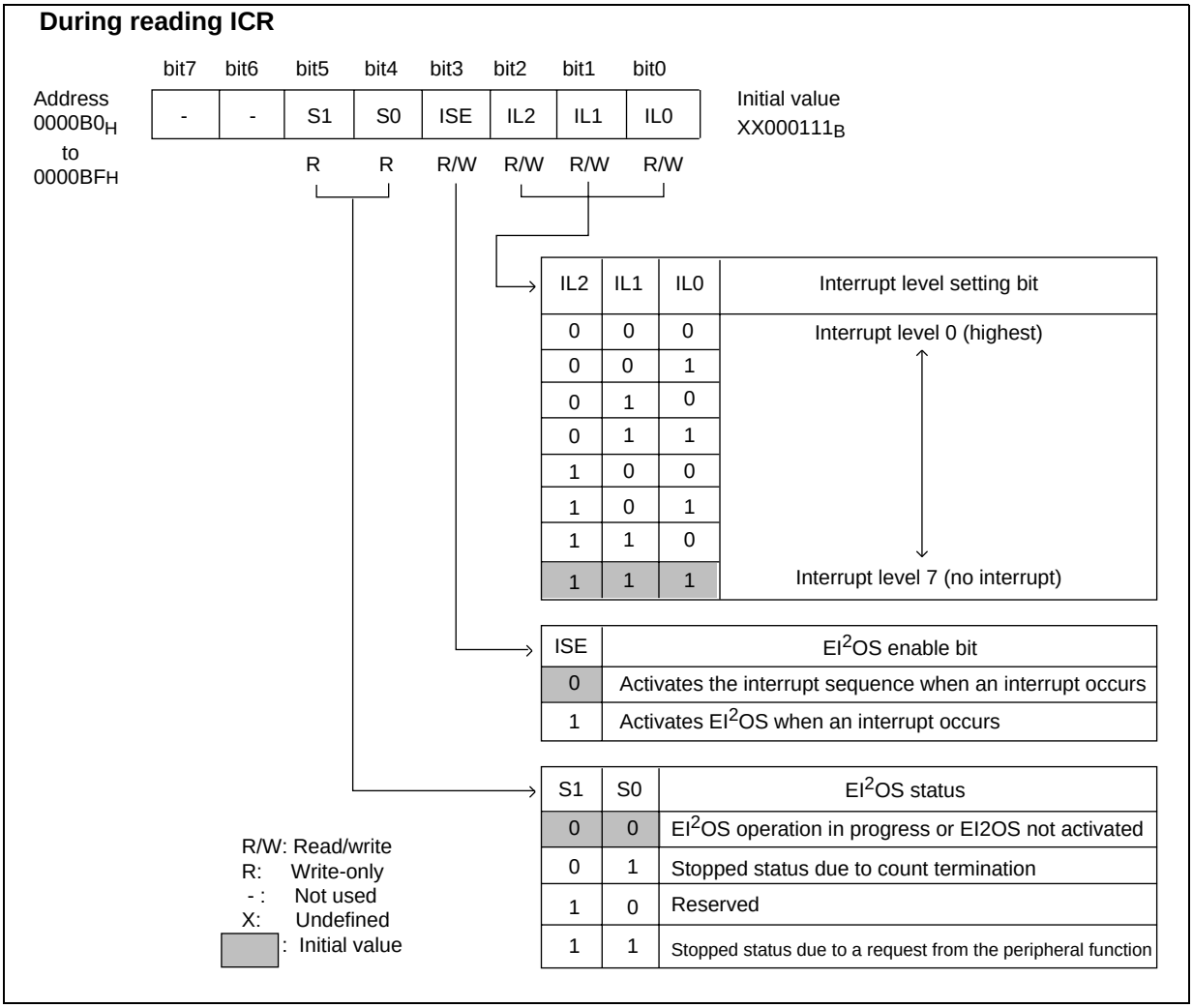


Figure 6.3-2 Interrupt control registers (ICR00 to ICR15) during reading

6.3.2 Interrupt control register functions

The interrupt control registers (ICR00 to ICR15) consist of the following four functional bits:

- **Interrupt level setting bits (IL2 to IL0)**
- **Extended intelligent I/O service (EI²OS) enable bit (ISE)**
- **Extended intelligent I/O service (EI²OS) channel selection bits (ICS3 to ICS0)**
- **Extended intelligent I/O service (EI²OS) status (S1 to S0)**

■ Configuration of interrupt control registers (ICR)

Figure 6.3-3 shows the configuration of the interrupt control register (ICR) bits.

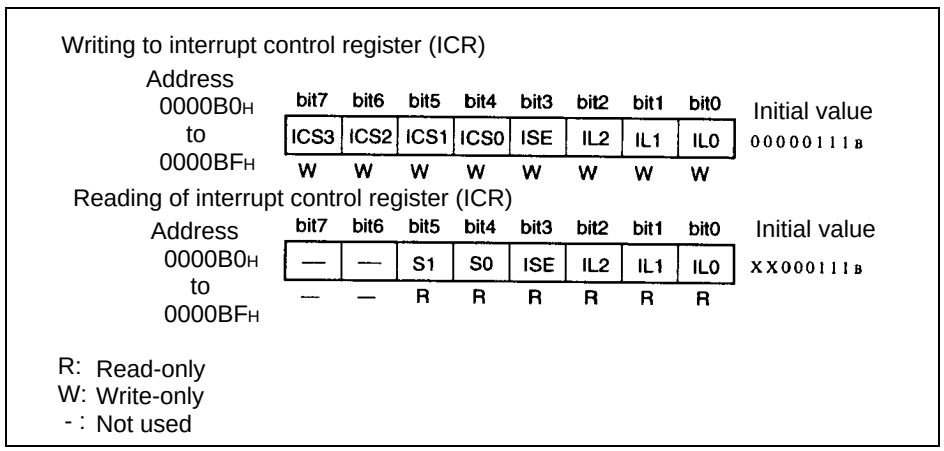


Figure 6.3-3 Configuration of interrupt control registers (ICR)

Reference

- The ICS3 to ICS0 bits are valid only when the extended intelligent I/O service (EI²OS) has been activated. To activate EI²OS, set the ISE bit to 1. To not activate EI²OS, set the ISE bit to 0. When EI²OS is not activated, setting ICS3 to ICS0 is optional.
- ICS1 and ICS0 are valid only for writing. S1 and S0 are valid only for reading.

■ Interrupt control register functions

● Interrupt level setting bits (IL2 to IL0)

These bits set the interrupt level of the corresponding peripheral function. These bits are initialized to level 7 (no interrupts) by a reset.

Table 6.3-2 shows the correspondence between the interrupt level setting bits and interrupt levels.

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)
0	0	1	↕
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	6 (lowest priority)
1	1	1	7 (no interrupts)

- **Extended intelligent I/O service (EI²OS) enable bit (ISE)**

  If this bit is “1” when an interrupt request is generated, EI²OS is activated. If this bit is “0” when an interrupt request is generated, the interrupt sequence is activated. When the EI²OS termination condition is met (when the S1 and S0 bits are not 00_B), the ISE bit is cleared. If the corresponding peripheral function does not have the EI²OS function, the ISE bit must be set to “0” by software. The ISE bit is initialized to “0” by a reset.

- **Extended intelligent I/O service (EI²OS) channel selection bits (ICS3 to ICS0)**

  These write-only bits specify the EI²OS channel. The EI²OS descriptor address is determined based on the value set here. The ICS bit is initialized to “0000_B“by a reset.

  Table 6.3-3 shows the correspondence between the EI²OS channel selection bits and descriptor addresses.

Table 6.3-3 Correspondence between the EI²OS channel selection bits and descriptor addresses

ICS3	ICS2	ICS1	ICS0	Selected channel	Descriptor address
0	0	0	0	0	000100_H
0	0	0	1	1	000108_H
0	0	1	0	2	000110_H
0	0	1	1	3	000118_H
0	1	0	0	4	000120_H
0	1	0	1	5	000128_H
0	1	1	0	6	000130_H
0	1	1	1	7	000138_H
1	0	0	0	8	000140_H
1	0	0	1	9	000148_H
1	0	1	0	10	000150_H
1	0	1	1	11	000158_H
1	1	0	0	12	000160_H
1	1	0	1	13	000168_H
1	1	1	0	14	000170_H
1	1	1	1	15	000178_H

● **Extended intelligent I/O service (EI²OS) status bits (S1, S0)**

These are read-only bits. When this value is checked at EI²OS termination, the operating status and termination status can be distinguished. These bits are initialized to "00_B" by a reset.

Table 6.3-4 shows the relationship between the S0 and S1 bits and the EI²OS status

Table 6.3-4 Relationship between EI²OS status bits and the EI²OS status

S1	S0	EI²OS status
0	0	EI²OS 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

Memo

6.4 Hardware Interrupts

The hardware interrupt function temporarily interrupts the program being executed by the CPU and transfers control to a user-defined interrupt processing program in response to an interrupt signal from a peripheral function.
The extended intelligent I/O service (EI^{2}OS) and external interrupts are executed as a type of hardware interrupt.

■ Hardware interrupts

● Hardware interrupt function

The hardware interrupt function compares the interrupt level of the interrupt request signal output by a peripheral function with the interrupt level mask register (ILM) in the CPU processor status (PS). The function then references the contents of the I flag in the processor status (PS) through the hardware and decides if the interrupt can be accepted.

When the hardware interrupt is accepted, the CPU internal registers are automatically saved on the system stack. The currently requested interrupt level is stored in the interrupt level mask register (ILM), and the function branches to the corresponding interrupt vector.

● Multiple interrupts

Multiple hardware interrupts can be activated.

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

EI^{2}OS is an automatic transfer function between memory and I/O. When the specified transfer count has been completed, a hardware interrupt is activated. Multiple EI^{2}OS activation is not possible to occur. During EI^{2}OS processing, all other interrupt requests and EI^{2}OS requests are held.

● External interrupt

An external interrupt (including wake-up interrupts) is accepted from a peripheral function (interrupt request detection circuit) as a hardware interrupt.

● Interrupt vector

Interrupt vector tables referenced during interrupt processing are allocated to memory at “FFFC00$_H$“ to “FFFFFF$_H$”. These tables are shared by software interrupts.

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

■ Hardware interrupt structure

Table 6.4-1 lists four mechanisms used for hardware interrupts. These four mechanisms must be included in the 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
Interrupt controller	Interrupt control register (ICR)	Sets the interrupt level and controls EI^{2}OS
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 suppression

Acceptance of hardware interrupt requests is suppressed under the following conditions.

● Hardware interrupt suppression during writing to the peripheral function control register area

When data is being written to the peripheral function control register area, hardware interrupt requests are not accepted. This prevents the CPU from making operational mistakes. The mistakes may be caused if an interrupt request is generated during data is being written to the interrupt control registers for a resource. The peripheral function control register area is not the I/O addressing area at "000000H" to "0000FFH", but the area allocated to the control register of the peripheral function control register and data register.

Figure 6.4-1 shows hardware interrupt operation during writing to the built-in peripheral area

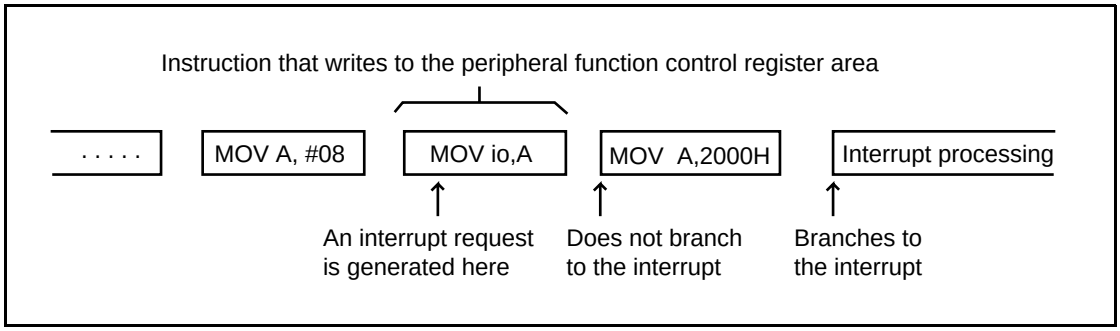


Figure 6.4-1 Hardware interrupt request while writing to the peripheral function control register area

● Hardware interrupt suppression by interrupt suppression instruction

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

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 ILM, #imm8 OR CCR, #imm8 AND CCR, #imm8 POPW PS

Even if a valid hardware interrupt request is generated during execution of one of these instructions, the interrupt is not processed until the first time an instruction of a different type is executed.

● Hardware interrupt suppression during execution of software interrupt

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

Memo

6.4.1 Operation of hardware interrupts

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

■ Hardware interrupt activation

● Peripheral function operation (generation of an interrupt request)

A peripheral function that has a hardware interrupt request function also has an interrupt request flag that indicates the presence of interrupt requests and an interrupt enable flag that determines whether CPU interrupt requests are enabled or disabled. The interrupt request flag is set when an event specified by the peripheral function occurs.

● Interrupt controller operation (interrupt request control)

When two or more interrupt requests are received at the same time, the interrupt controller compares their interrupt levels (IL) in their interrupt requests register. The interrupt controller selects the request with the highest level (with the smallest IL value) and posts it to the CPU. When multiple requests have the same level, the request with the smallest interrupt number has the highest priority.

● 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 IL < ILM and interrupts are enabled (PS: CCR: I = 1), the CPU activates the interrupt processing microcode after the instruction currently being executed terminates.

At the beginning of the interrupt processing microcode, the CPU checks the ISE bit in the interrupt control register (ICR). If ISE = 0, the CPU continues the execution of interrupt processing. (If ISE = 1, EI^{2}OS is activated.)

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

In an interrupt processing program, when the interrupt request flag of the peripheral function that generated the interrupt cause is cleared and the RETI instruction is executed, 12-byte data saved on the system stack is restored to the dedicated registers and the processing that was being executed before branching for the interrupt is resumed.

When the interrupt request flag is cleared, interrupt requests output by the peripheral function to the interrupt controller are automatically canceled.

■ Hardware interrupt operation

Figure 6.4-2 shows hardware interrupt operation from generation of a hardware interrupt to the completion of interrupt processing.

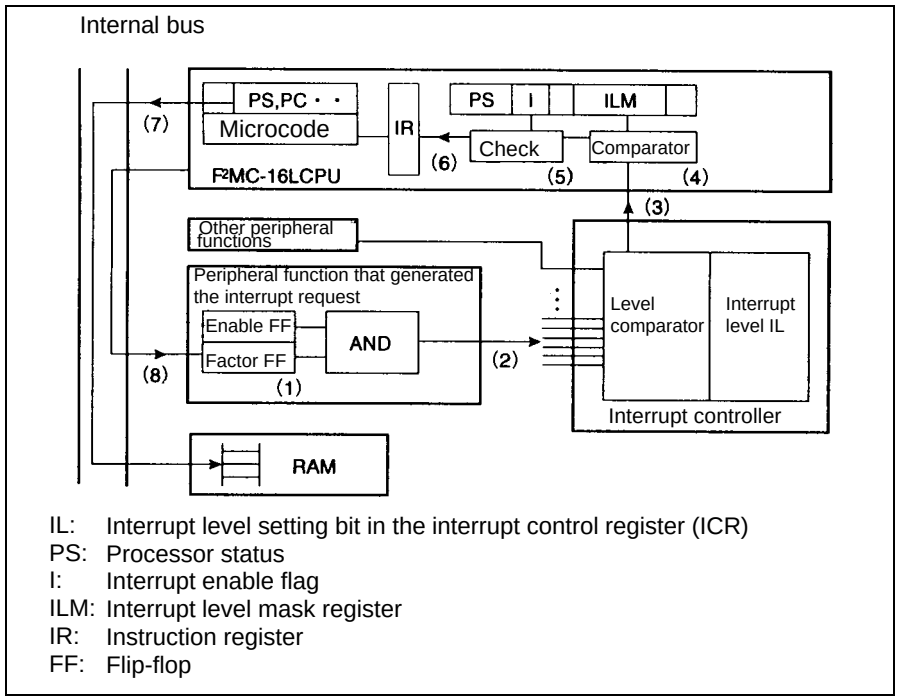


Figure 6.4-2 Hardware interrupt operation

(1) An interrupt cause is generated by the peripheral function.

(2) The interrupt enable bit of the peripheral function is checked. If the interrupt is enabled, the interrupt request is sent from the peripheral function to the interrupt controller.

(3) The interrupt controller determines the priority of simultaneous interrupt requests, then transfers the interrupt level (IL) that matches the corresponding interrupt request 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 in (5) shows that the I flag is interrupt enabled (I = 1), the CPU waits until the execution of the instruction currently being executed terminates. At termination, the CPU sets the requested level (IL) in the ILM.

(7) Registers are saved, and processing branches to the interrupt processing routine.

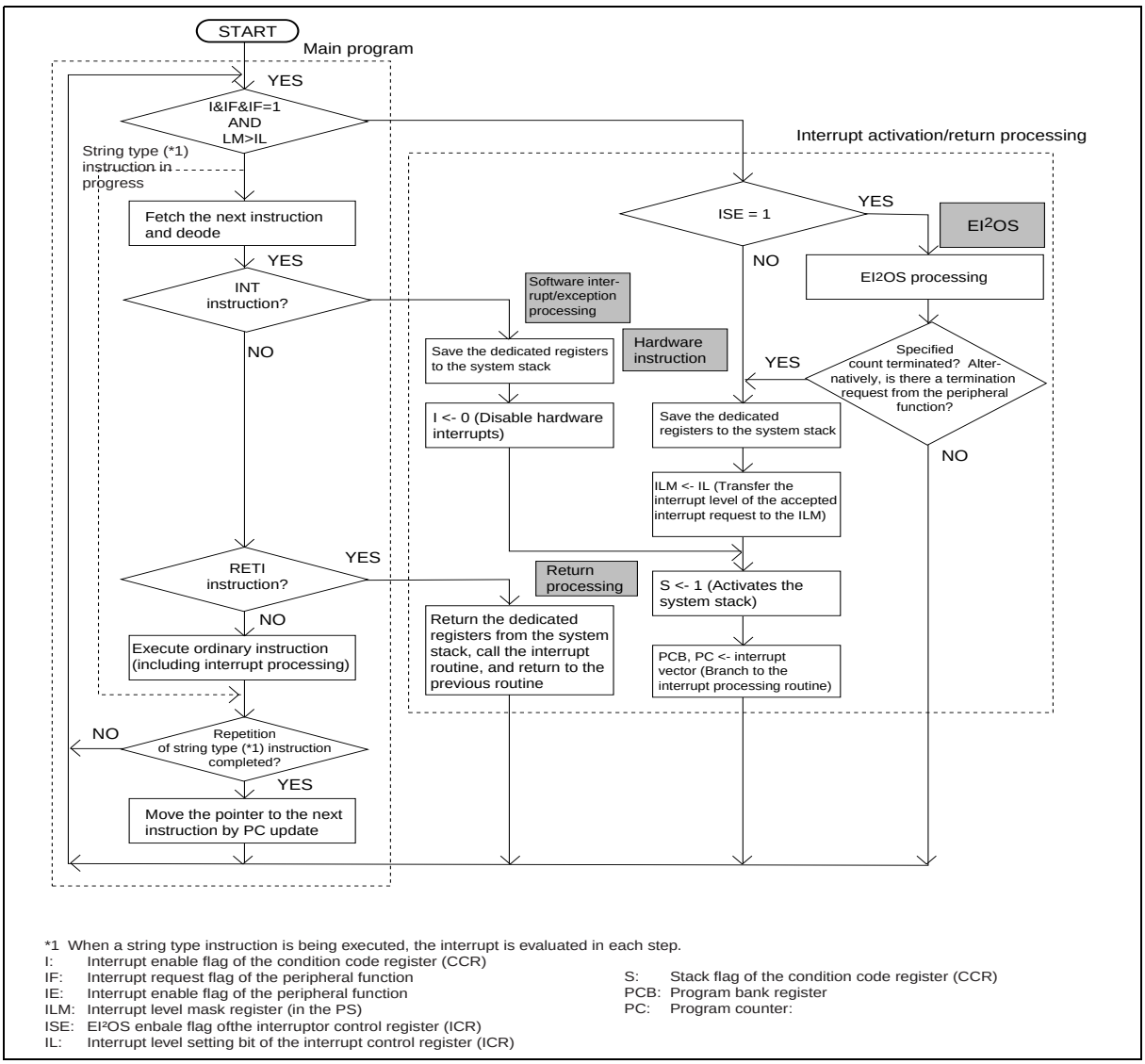
(8) The interrupt cause that was generated in (1) is cleared by software in the interrupt processing routine. Execution of the RETI instruction terminates the interrupt processing.

6.4.2 Processing for interrupt operation

When an interrupt request is generated by the peripheral function, the interrupt controller transmits the interrupt level to the CPU. If the CPU is able to accept interrupts, the interrupt controller temporarily interrupts the instruction currently being executed. The interrupt controller then executes the interrupt processing routine or activates the extended intelligent I/O service (EI²OS).
If a software interrupt is generated by the INT instruction, the interrupt processing routine is executed regardless of the CPU status. In this case, hardware interrupts are not allowed.

■ Processing for interrupt operation

Figure 6.4-3 shows the flow of processing for interrupt operation.



*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 ofthe 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:

Figure 6.4-3 Flow of interrupt processing

6.4.3 Procedure for using hardware interrupts

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

■ Procedure for using hardware interrupts

Figure 6.4-4 shows an example of the procedure for using hardware interrupts.

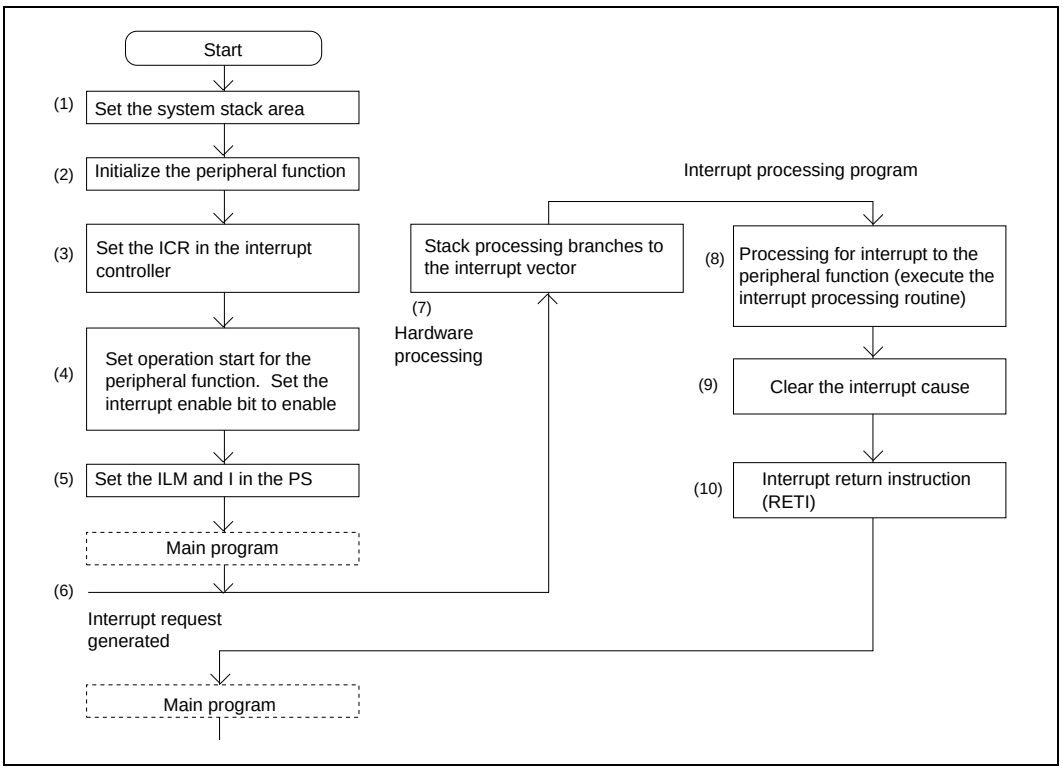


Figure 6.4-4 Procedure for using hardware interrupts

(1) Set the system stack area.

(2) Initialize a peripheral function that can generate interrupt requests.

(3 Set the interrupt control register (ICR) in the interrupt controller.

(4) Set the peripheral function to the operation start status, and set the interrupt enable bit to enable.

(5) Set the interrupt level mask register (ILM) and interrupt enable flag (I) to interrupt acceptable.

(6) An interrupt generated in the peripheral function causes a hardware interrupt request.

(7) The interrupt processing hardware saves the registers and branches to the interrupt processing program.

(8) The interrupt processing program processes the peripheral function in response to the generated interrupt.

(9) Clear the peripheral function interrupt request.

(10) Execute the interrupt return instruction, and return to the program location before branching.

6.4.4 Multiple interrupts

Multiple hardware interrupts can be implemented by setting different interrupt levels in the interrupt level setting bits (IL0, IL1, IL2) of the interrupt control register (ICR) in response to multiple interrupt requests from peripheral functions. Use of multiple interrupts, however, is not possible with the extended intelligent I/O service.

■ Multiple interrupts

● Operation of multiple interrupts

During execution of an interrupt processing routine, if an interrupt request with a higher-priority interrupt level is generated, the current interrupt processing is interrupted and the interrupt request with the higher-priority interrupt level is accepted. When the interrupt request with the higher-priority interrupt level terminates, the CPU returns to the previous interrupt processing.

0 to 7 can be set as the interrupt level. If level 7 is set, the CPU does not accept interrupt requests.

During execution of interrupt processing, if an interrupt request with the same or lower-priority interrupt level is generated, the new interrupt request is held until the current interrupt terminates unless the I flag or ILM is changed.

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).

<Check>

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 higher priority 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 is performed.

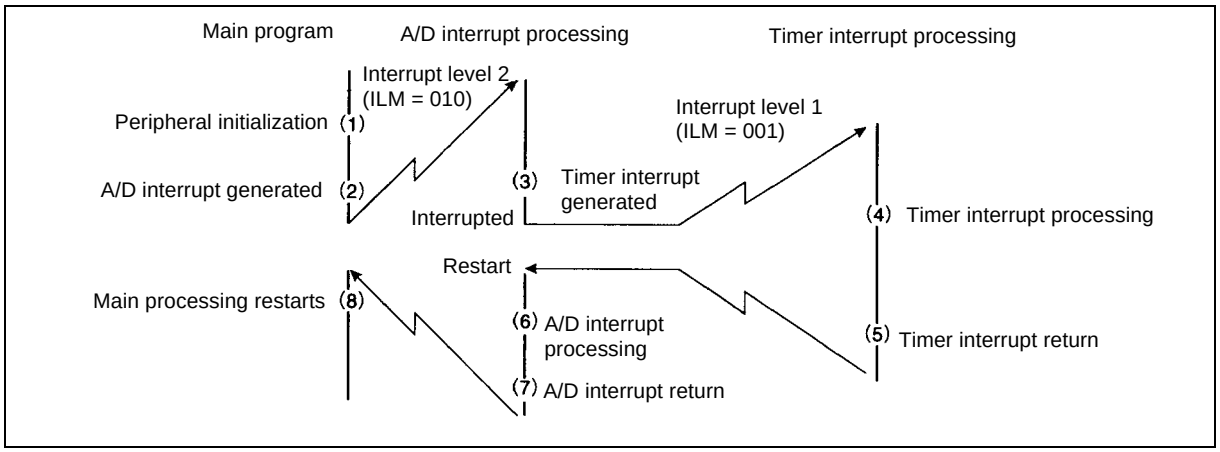


Figure 6.4-5 Example of multiple interrupts

❍ A/D interrupt generated

When the A/D converter interrupt processing starts, the interrupt level mask register (ILM) automatically has the same value (2 in the example) as the A/D converter interrupt level (ICR: IL2 to IL0). If a level-1 or level-0 interrupt request is generated, this interrupt processing has priority.

❍ Interrupt processing terminated

When the interrupt processing terminates and the return instruction (RETI) is executed, the values of the dedicated registers (A, DPR, ADB, DTB, PCB, PC, and PS) are returned from the stack, and the interrupt level mask register (ILM) has the value that it had before the interrupt.

6.4.5 Hardware interrupt processing time

From the occurrence of a hardware interrupt request to the execution of an interrupt processing routine, the time for the instruction currently being executed to terminate and the time required to handle an interrupt are necessary.

■ Hardware interrupt processing time

From the occurrence of a hardware interrupt request to the acceptance of the interrupt and to the execution of an interrupt processing routine, the time to wait for sampling for an interrupt request and the time required to handle an interrupt (time to prepare for interrupt processing) are necessary. Figure 6.4-6 shows the interrupt processing time.

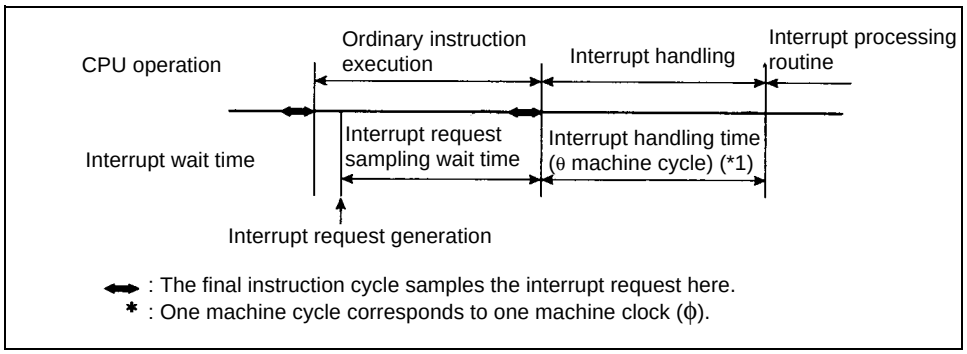


Figure 6.4-6 Interrupt processing time

● Interrupt request sampling wait time

The interrupt request sampling wait time is the time from the occurrence of and interrupt request to the termination of the instruction currently being executed.

Whether an interrupt request has been occurred is determined by sampling the instruction for an interrupt request in the final cycle of the instruction. Consequently, the CPU cannot identify an interrupt request during execution of each instruction creating a delay.

<Reference>

The interrupt request sampling wait time is the maximum when an interrupt request is generated as soon as the POPW RW0, ... RW7 instruction (45 machine cycles), which takes the longest to execute, starts.

● **Interrupt handling time (θ machine cycle)**

The CPU saves dedicated registers to the system stack and fetches interrupt vectors after it receives an interrupt request. The required handling time for this processing is θ machine cycles. The interrupt handling time is calculated with the following formula:

When an interrupt is activated: θ = 24 + 6 + Z machine cycles

When returning from an interrupt: θ = 11 + 6 + Z machine cycles (RETI instruction)

The interrupt handling time is different depending on each address pointed to by the stack pointer.
Table 6.4-3 shows the compensation values (Z) for the interrupt handling time.

Table 6.4-3 Compensation values (Z) for the interrupt handling time

Address pointed to by the stack pointer	Compensation 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

<Reference>

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

6.5 Software Interrupts

When the software interrupt instruction (INT instruction) is executed, the software interrupt function transfers control from the program being executed by the CPU to the user-defined interrupt processing program. Hardware interrupts are disabled during execution of a software interrupt.

■ Software interrupt activation

● Software interrupt activation

The INT instruction is used to activate a software interrupt. There is no interrupt request flag or enable flag for software interrupt requests. When the INT instruction is executed, an interrupt request is always generated.

● 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 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.

<Reference>

See Section 6.2, "Interrupt Causes and Interrupt Vectors," in Chapter 6 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 12-byte data saved into 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 shows software interrupt operation from the occurrence of a software interrupt to the completion of interrupt processing.

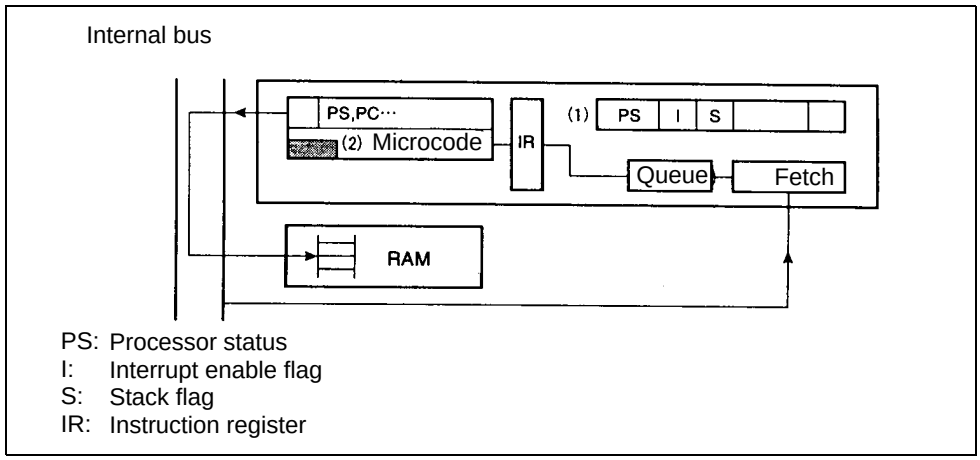


Figure 6.5-1 Software interrupt operation

(1)A software interrupt instruction is executed.

(2)The dedicated registers are saved according to the microcode that corresponds to the software interrupt instruction, and other necessary processing is performed. Branch processing is then executed.

(3)The RETI instruction in the user interrupt processing routine terminates the interrupt processing.

<Check>

When the program bank register (PCB) is FF_H, the vector area of the CALLV instruction overlaps the INT #vct8 instruction table. When creating the software, be careful of the duplicated address of the CALLV instruction and INT #vct8 instruction.

6.6 Interrupt of Extended Intelligent I/O Service (EI²OS)

The extended intelligent I/O service (EI²OS) automatically transfers data between a peripheral function (I/O) and memory. When the data transfer terminates, a hardware interrupt is generated.

■ Extended intelligent I/O service (EI²OS)

The extended intelligent I/O service is a type of hardware interrupt. It automatically transfers data between a peripheral function (I/O) and a memory. Traditionally, data transfer with a peripheral function (I/O) has been performed by the interrupt processing program. EI²OS performs this data transfer in the same way as direct memory access (DMA). At termination, EI²OS sets the termination condition and automatically branches to the interrupt processing routine. The user creates programs only for EI²OS activation and termination. Data transfer programs in between are not required.

● Advantages of extended intelligent I/O service (EI²OS)

Compared to data transfer performed by the interrupt processing routine, EI²OS has the following advantages.

- Coding a transfer program is not necessary, reducing program size.
- Because transfer can be stopped depending on the peripheral function (I/O) status, unnecessary data transfer can be eliminated.
- Incrementing or no updating can be selected for the buffer address.
- Incrementing or no updating can be selected for the I/O register address.

● Extended intelligent I/O service (EI²OS) termination interrupt

When data transfer by EI²OS terminates, a termination condition is set in the S1 and S0 bits in the interrupt control register (ICR). Processing then automatically branches to the interrupt processing routine.

The EI²OS termination factor can be determined by checking the EI²OS status (ICR: S1, S0) with the interrupt processing program.

<Reference>

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

● Interrupt control register (ICR)

This register, which is located in the interrupt controller, activates EI²OS, specifies the EI²OS channel, and displays the (EI2OS) termination status.

● Extended intelligent I/O service (EI²OS) descriptor (ISD)

This descriptor, which is located in RAM at “000100H” to “00017FH”, is an eight-byte data that retains the transfer mode, I/O address, transfer count, and buffer address. The descriptor handles 16 channels. The channel is specified by the interrupt control register (ICR).

<Check>

When the extended intelligent I/O service (EI²OS) is operating, execution of the CPU program stops.

■ Operation of the extended intelligent I/O service (EI²OS)

Figure 6.6-1 shows EI²OS operation.

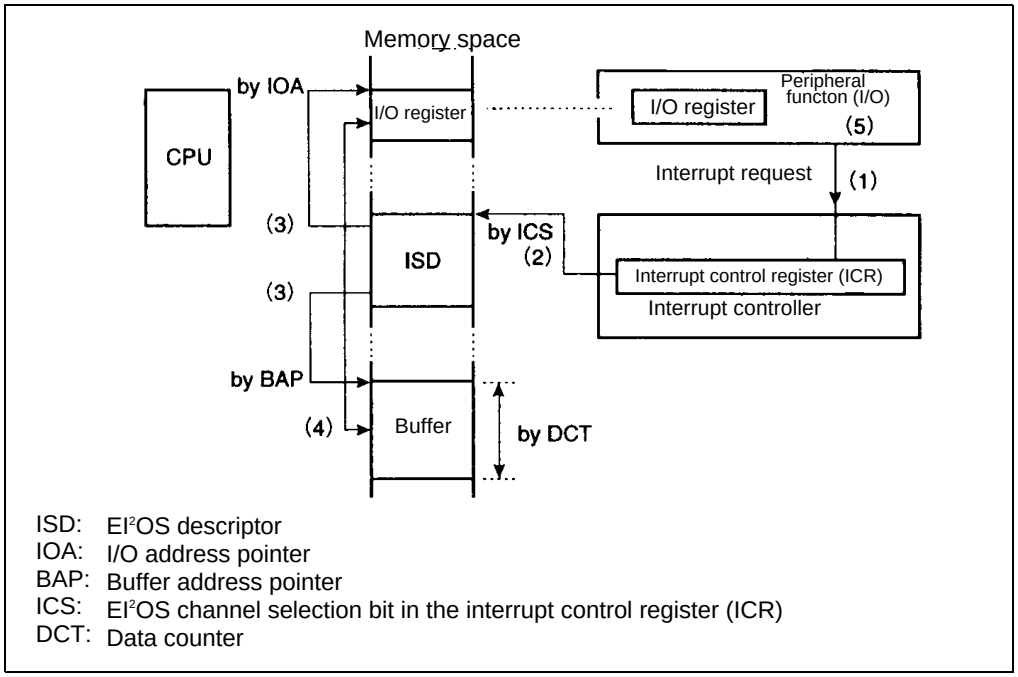


Figure 6.6-1 Extended intelligent I/O service (EI²OS) operation

(1)I/O requests transfer.

(2)The interrupt controller selects the descriptor.

(3)The transfer source and transfer destination are read from the descriptor.

(4)Transfer is performed between I/O and memory.

(5)The interrupt cause is automatically cleared.

6.6.1 Extended intelligent I/O service (EI²OS) descriptor (ISD)

The extended intelligent I/O service (EI²OS) descriptor (ISD) resides in internal RAM at "000100H" to "00017FH". The ISD consists of 8 bytes x 16 channels.

■ Configuration of the extended intelligent I/O service (EI²OS) descriptor (ISD)

The ISD consists of 8 bytes x 16 channels. Each ISD has the structure shown in Figure 6.6-2. Table 6.6-1 shows the correspondence between channel numbers and ISD addresses.

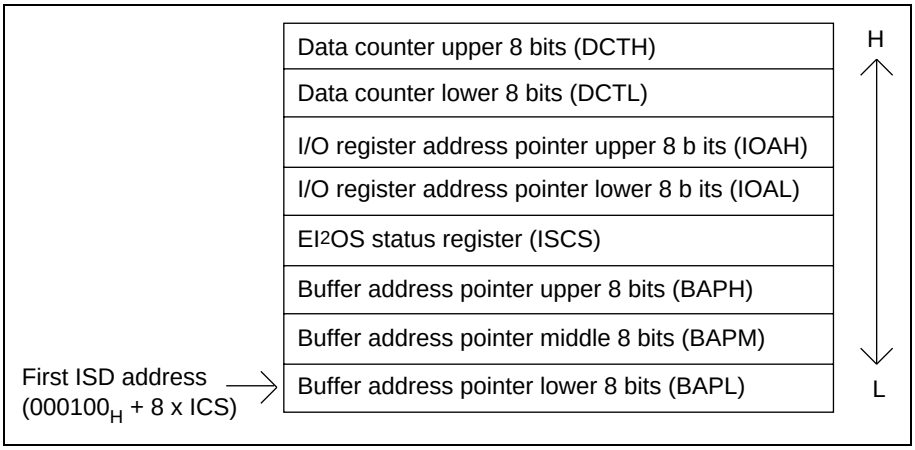


Figure 6.6-2 Configuration of EI²OS descriptor (ISD)

Table 6.6-1 Correspondence between channel numbers and descriptor addresses

Channel	Descriptor address
0	000100H
1	000108H
2	000110H
3	000118H
4	000120H
5	000128H
6	000130H
7	000138H
8	000140H
9	000148H
10	000150H
11	000158H
12	000160H
13	000168H
14	000170H
15	000178H

Memo

6.6.2 Registers of the extended intelligent I/O service (EI²OS) descriptor (ISD)

The extended intelligent I/O service (EI²OS) descriptor (ISD) consists of the following registers.

- **Data counter (DCT)**
- **I/O register address pointer (IOA)**
- **EI²OS status register (ISCS)**
- **Buffer address pointer (BAP)**

Note that the initial value of each register is undefined after a reset.

■ **Data counter (DCT)**

The DCT is a 16-bit register that serves as a counter for the data transfer count. After each data transfer, the counter is decremented by 1. When the counter reaches zero, EI²OS terminates.

Figure 6.6-3 shows the configuration of the DCT.

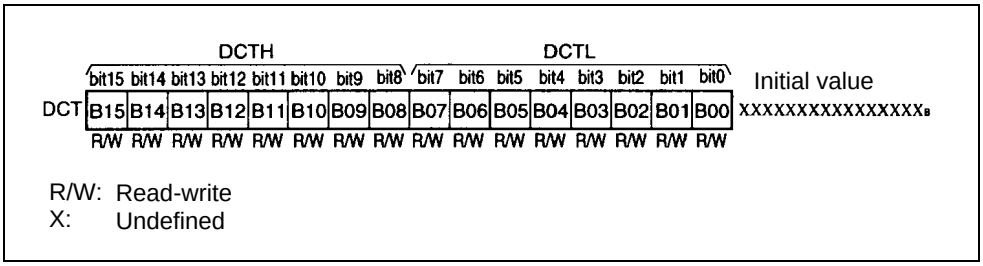


Figure 6.6-3 Configuration of DCT

■ **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 all zeros. Any I/O from "000000H" to "00FFFFH" can be specified by address.

Figure 6.6-4 shows the configuration of the IOA.

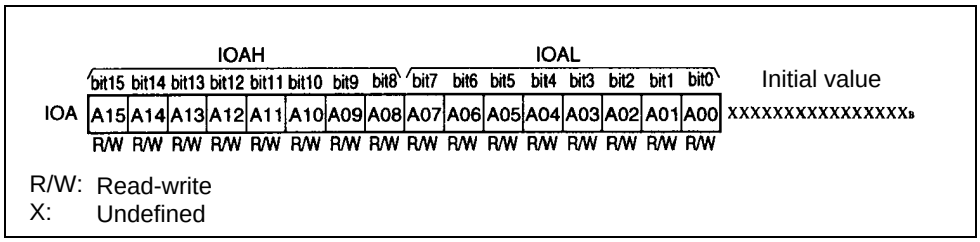
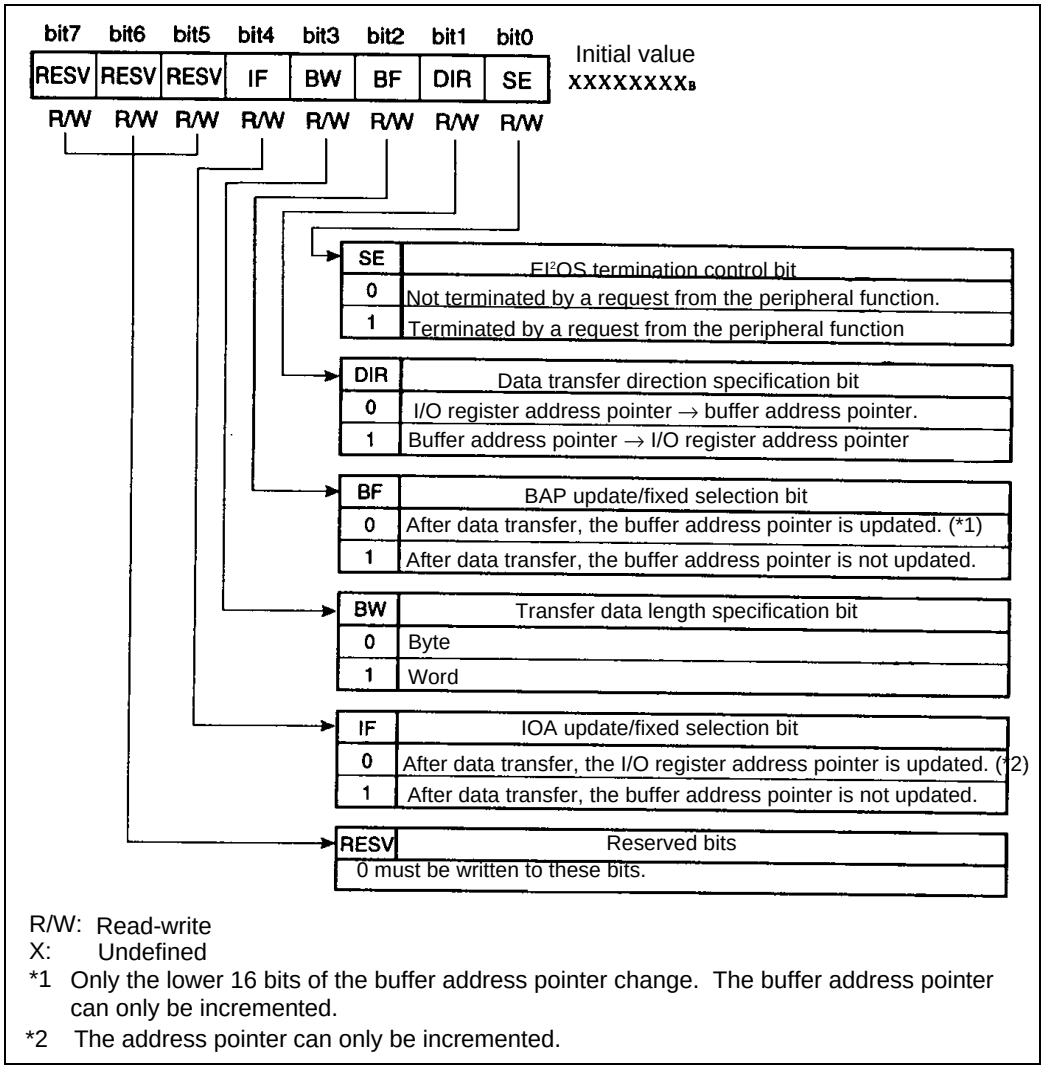


Figure 6.6-4 Configuration of I/O register address pointer (IOA)

■ Extended intelligent I/O service (EI²OS) 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 shows the configuration of the 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.

Figure 6.6-5 Configuration of EI²OS status register (ISCS)

■ Buffer address pointer (BAP)

The BAP is a 24-bit register that retains the address used by EI²OS for the next transfer. Since one independent BAP exists for each EI²OS channel, each EI²OS channel can transfer data between any address in the 16-megabyte space and the I/O. If the BF bit (BAP update/fixed selection bit in the EI²OS status register) in the EI²OS status register (ISCS) is set to "update yes," only the lower 16 bits (BAPH, BAPL) of the BAP change; the upper 8 bits (BAPH) do not change. Figure 6.6-6 shows the configuration of the BAP.

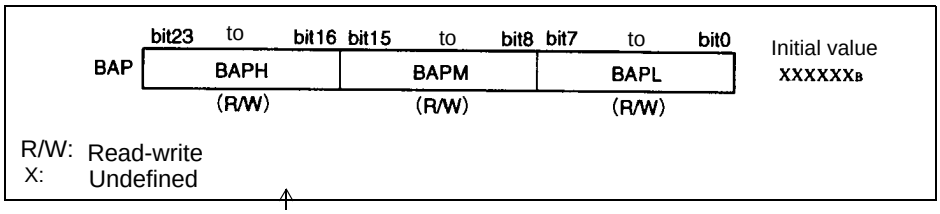


Figure 6.6-6 Configuration of buffer address pointer (BAP)

Reference

- The area that can be specified by the I/O address pointer (IOA) extends from ”000000H” to “00FFFFH”.
- The area that can be specified with the buffer address pointer (BAP) extends from ”000000H” to ”FFFFFFH”.
- The maximum transfer count that can be specified by the data counter (DCT) is 65,536 (64 kilobytes).

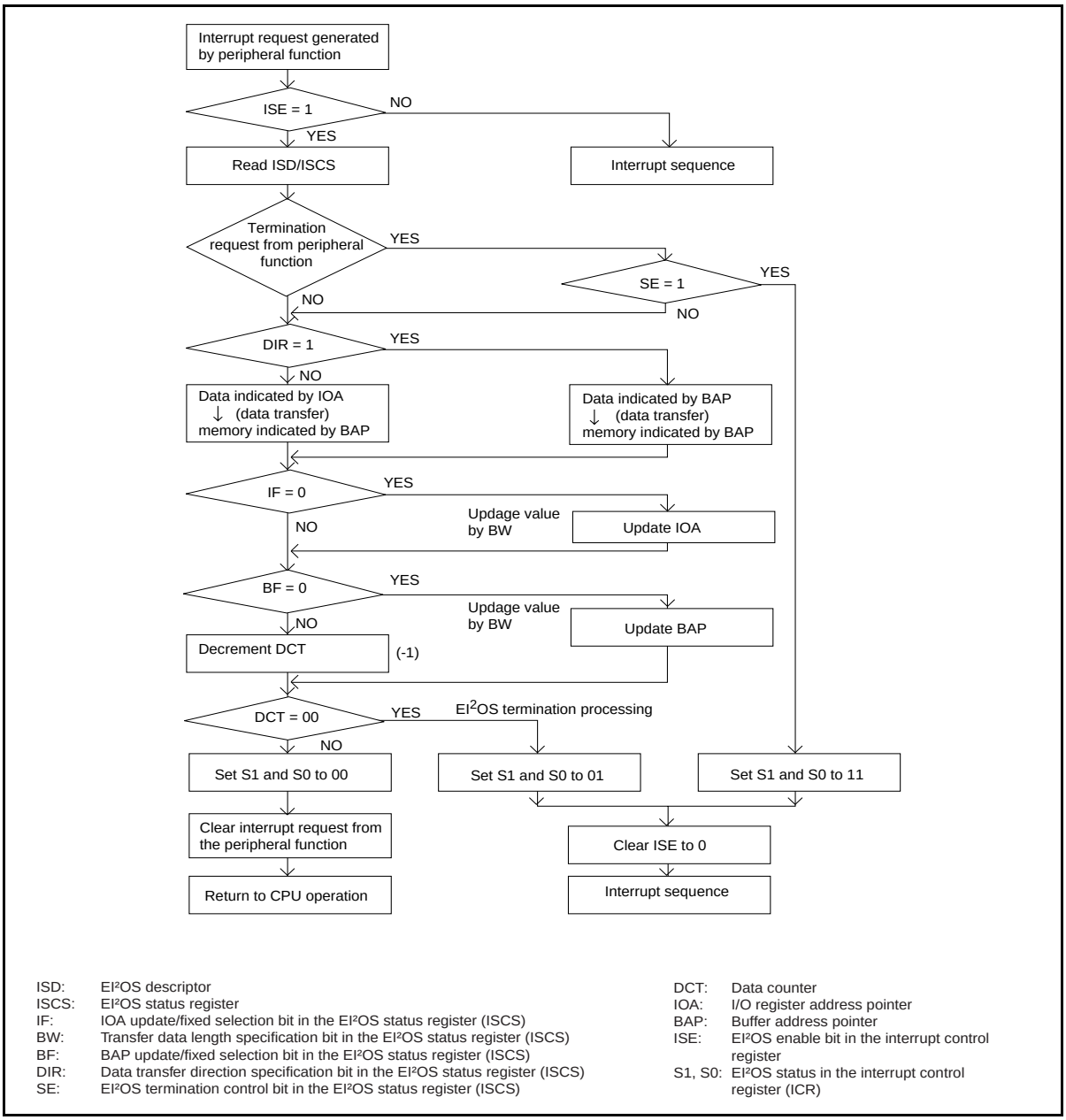
Memo

6.7 Operation of the extended intelligent I/O service (EI²OS)

If an interrupt request is generated by a peripheral function, EI²OS activation is set in the corresponding interrupt control register (ICR) that the CPU uses EI²OS to transfer data. When the specified data transfer count terminates, the hardware interrupt is automatically processed.

■ Operation flow of the extended intelligent I/O service (EI²OS)

Figure 6.7-1 shows the flow of EI²OS operation based on the internal microcode of the CPU.



ISD: EI²OS descriptor
ISCS: EI²OS status register
IF: IOA update/fixed selection bit in the 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
IOA: I/O register address pointer
BAP: Buffer address pointer
ISE: EI²OS enable bit in the interrupt control register
S1, S0: EI²OS status in the interrupt control register (ICR)

Figure 6.7-1 Flow of extended intelligent I/O service (EI²OS) operation

6.7.1 Procedure for using the extended intelligent I/O service (EI²OS)

Before the extended intelligent I/O service (EI²OS) can be used, the system stack area, extended intelligent I/O service (EI²OS) descriptor, interrupt function, and interrupt control register (ICR) must be set.

■ Procedure for using the extended intelligent I/O service (EI²OS)

Figure 6.7-2 shows the EI²OS software and hardware processing.

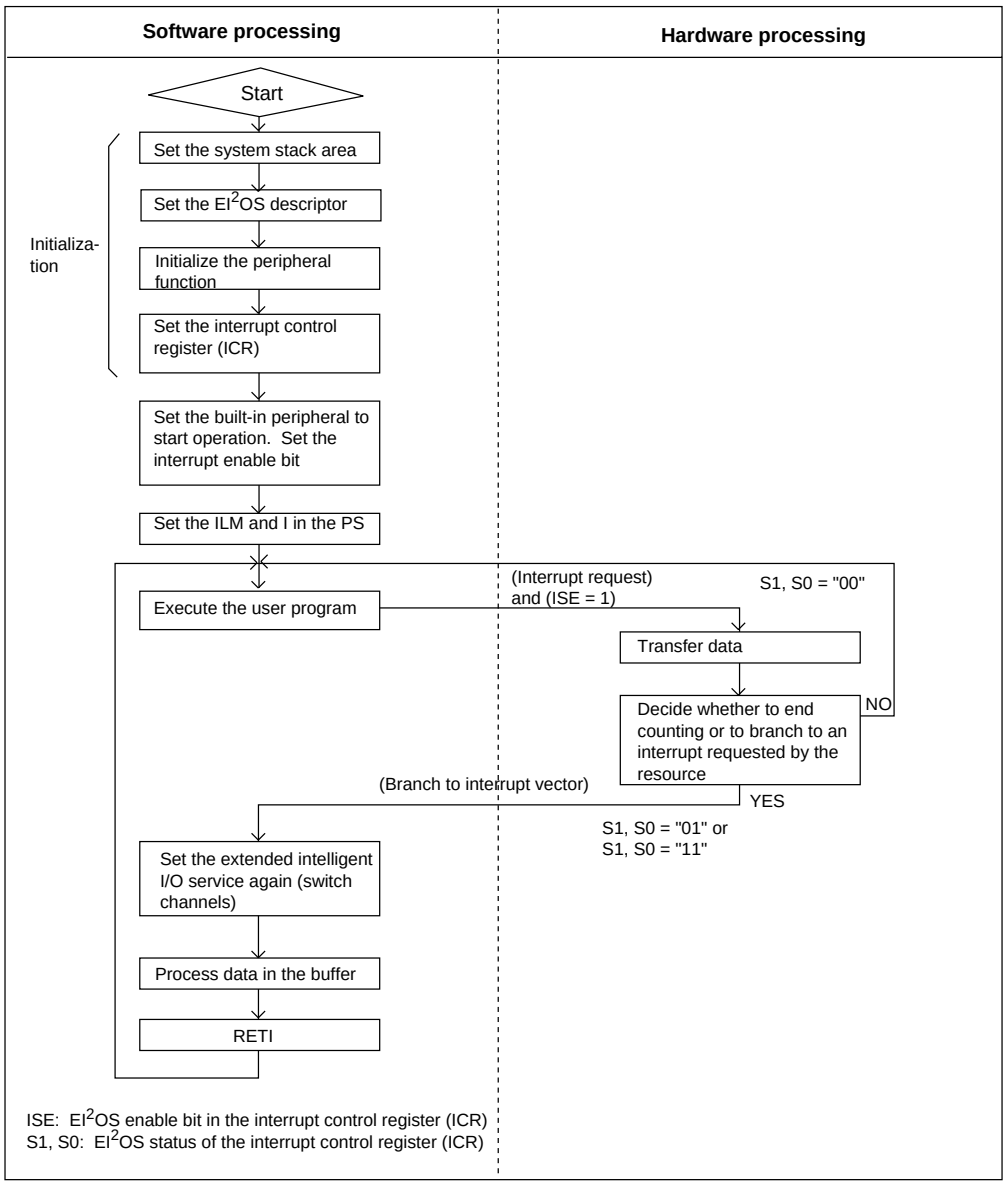


Figure 6.7-1 Procedure for using the extended intelligent I/O service (EI²OS)

6.7.2 Processing time of the extended intelligent I/O service (EI²OS)

The time required for processing the extended intelligent I/O service (EI²OS) is varied according to the following factors:

- **EI²OS status register (ISCS) setting**
- **Address (area) pointed by the I/O register address pointer (IOA)**
- **Address (area) pointed 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 EI²OS terminates, the interrupt handling time is needed to be considered.

■ Processing time (one transfer time) of the extended intelligent I/O service (EI²OS)

● When data transfer continues

The EI²OS processing time for data transfer continuation is shown in Table 6.7-1 based on the EI²OS status register (ISCS) setting.

Table 6.7-1 Extended intelligent I/O service execution time

EI²OS termination control bit (SE) setting		Terminates due to termination request from the peripheral		Ignores termination request from the peripheral	
IOA 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 6.7-2, compensation is necessary depending on the EI²OS execution condition.

Table 6.7-2 Data transfer compensation value for EI²OS 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)**

Because the hardware interrupt is activated when data transfer by EI^{2}OS terminates, the interrupt handling time is added. The EI^{2}OS processing time when counting terminates is calculated with the following formula:

EI^{2}OS processing time when counting terminates = EI^{2}OS processing time when data is transferred + (21 + 6 x Z) Machine cycles

Interrupt handling time

The interrupt handling time is different for each address pointed to by the stack pointer. Table 6.7-3 shows the compensation value (Z) for the interrupt handling time.

Table 6.7-3 Interpolation value (Z) for the interrupt handling time

Address pointed to by the stack pointer	Compensation 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 (I/O)**

When data transfer by EI^{2}OS is terminated before completion due to a termination request from the peripheral function (I/O) (ICR: S1, S0 = 11), 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 compensation value for the interrupt handling time (Table 6.7-3).

EI^{2}OS processing time for termination before completion = 36 + 6 x Z Machine cycle

<Reference>

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

6.8 Exception Processing Interrupt

In the F²MC-16LX, the execution of an undefined instruction results in exception processing.
Exception processing is basically the same as an interrupt. When the generation of an exception processing is detected on the instruction boundary, ordinary processing is interrupted and exception processing is executed.
Generally, exception processing occurs as the result of an unexpected operation. Exception processing should be used only to activate recovery software required for debugging or an emergency.

■ Exception processing

● Exception processing operation

The F²MC-16LX handles all codes that are not defined in the instruction map as undefined instructions. When an undefined instruction is executed, processing equivalent to the INT #10 software interrupt instruction 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

When the RETI instruction returns control from exception processing, exception processing restarts because the PC is pointing to the undefined instruction. Provide a solution such as resetting the software.

Memo

6.9 Stack Operations for Interrupt Processing

Once an interrupt is accepted, the contents of the dedicated registers are automatically saved to the system stack before a branch to interrupt processing. When the interrupt processing terminates, the previous processing is automatically restored from the stack.

■ Stack operations at the start of interrupt processing

Once an interrupt is accepted, the CPU automatically 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 6.9-1 shows the stack operations at the start of interrupt processing.

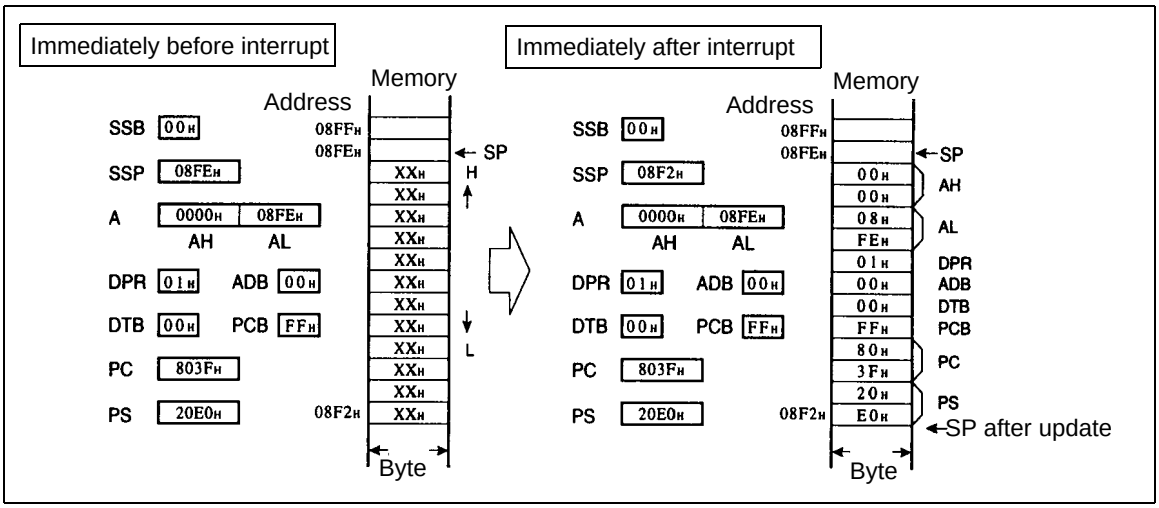


Figure 6.9-1 Stack operations at the start of interrupt processing

■ 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 6.9-2 shows the stack area.

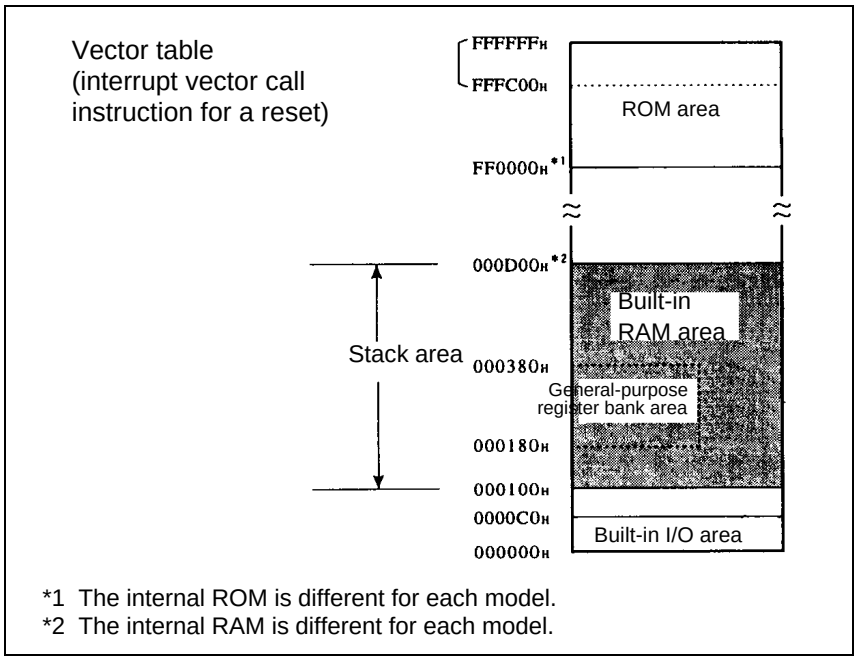


Figure 6.9-2 Stack area

<Check>

- Generally set an even-numbered address in the stack pointers (SSP and USP).
- 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 occurs, the user stack area being used is forcibly switched to the system stack. The system stack area must be set correctly even in a system that mainly uses the user stack area.

If division of the stack space is not particularly necessary, use only the system stack.

6.10 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     030H            ;   Interrupt/DTP enable register
ENRR        EQU     031H            ;   Interrupt/DTP flag
ELVR        EQU     032H            ;   Request level setting register
ICR00       EQU     0B0H            ;   Interrupt control register
STACK       SSEG                    ;   Stack
            RW      100
STACK_T     RW      1
STACK       ENDS
;---------Main program ----------------------------------------------------------------------------------------------------
CODECSEG
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, #40H       ;   Sets I flag of CCR in PS, enables interrupts.
            MOV     E:ICR00, #00H   ;   Sets interrupt level to 0 (highest priority).
            MOV     I:ELVR, #00000001B  ;   Requests that INT0 be made level H.
            MOV     I:ENRR, #00H    ;   Clears INT0 interrupt cause.
            MOV     I:ENRR, #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        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
```

■ Processing specifications of sample program for extended intelligent I/O service (EI²OS)

1)This program detects the H level signal input to the INT0 pin and activates the extended intelligent I/O service (EI²OS).

2)When the H level is input to the INT0 pin, EI²OS is activated. Data is transferred from port 0 to the memory at the 3000H address.

3)The number of transfer data bytes is 100 bytes. After 100 bytes are transferred, an interrupt is generated because EI²OS transfer has terminated.

● Sample coding

```
DDR1            EQU     000011H         ;   Port 1-direction register
ENIR            EQU     000030H         ;   Interrupt/DTP enable register
ENRR            EQU     000031H         ;   Interrupt/DTP factor register
ELVR            EQU     000032H         ;   Request level setting register
ICR00           EQU     0000B0H         ;   Interrupt control register
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
IOAL            EQU     000104H         ;   Lower I/O address pointer
IOAH            EQU     000105H         ;   Upper I/O address pointer
DCTL            EQU     000106H         ;   Low-order data counter
DCTH            EQU     000107H         ;   High-order data counter
ER0             EQU     ENRR:0          ;   Definition of external interrupt request flag bit
STACK           SSEG                    ;   Stack
                RW      100
STACK_T         RW      1
STACK           ENDS
;-----------------Main program-----------------------------------------------------------------------------------
CODECSEG
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     IOAL, #00H      ;   Sets the transfer source address
                                        ;   (port 0:  000000H).
                MOV     IOAH, #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:ENRR, #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, #40H       ;   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
;---------------Vector 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
```

Memo

CHAPTER 7 SETTING A MODE

This chapter describes the operating modes and memory access modes supported by the MB90560 series.

7.1 Setting a Mode

The F²MC-16LX supports the modes for access method, access areas, and tests. A mode is determined based on the settings by the mode pin at a reset as well as the mode data fetched.

■ Mode setting

The F²MC-16LX supports the modes for access method, access areas, and tests, classified as shown in Figure 7.1-1 in this module.

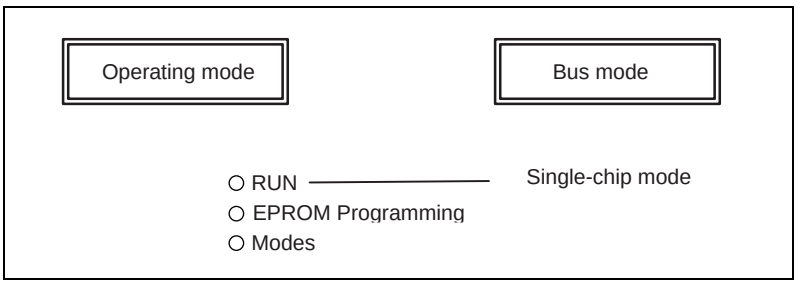


Figure 7.1-1 Mode classification

■ Operating modes

The operating modes control the operating state of the device, and are specified by the mode setting pin (MDx) and Mx bit contents in mode data. Selecting operating modes enables normal operation, and the activation of internal test programs and special test functions.

<Check>

Because the MB90560 series is only used in single-chip mode, set MD2, 1, 0 to 011 and set M1, 0 to 00.

■ Bus mode

The bus mode controls the operation of internal ROM and external access functions, and is specified by the mode setting pin (MDx) and Mx bit contents in mode data. The mode setting pin (MDx) specifies bus mode when reset vector and mode data are read. The Mx bit in mode data specifies bus mode during normal operation.

■ 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 Modes," for details.

<Check>

Because the MB90560 series is only used in single-chip mode, set MD2, 1, 0 to 011 and set M1, 0 to 00.

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)

The mode pins are used to select the data bus (external or internal) used for reading the reset vector and to specify the bus width when the external data bus is selected.

For a PROM version, the mode pins are also used to specify PROM programming mode, which is used to write programs and other data to internal ROM.

Table 7.2-1 shows the mode pin settings.

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	Mode data	The reset sequence and subsequent sequences are controlled by mode data.
1	0	0	Setting not allowed			
1	0	1				
1	1	0				
1	1	1	PROM programming mode	-	-	-

MD2 to MD0: Connect the pins to Vss for 0 and to Vcc for 1.

<Check>

Because the MB90560 series is only used in single-chip mode, set MD2, 1, 0 to 011 and set M1, 0 to 00.

7.3 Mode Data

The mode data is at memory location FFFFDF$_H$, and is used to specify the operation after a reset sequence. The mode data is automatically fetched to the CPU.

■ Mode data

During a reset sequence, the mode data at address FFFFDF$_H$ is fetched to the mode register in the CPU core. The CPU uses the mode data to set the memory access mode.

The contents of the mode register can only be changed during the reset sequence. The settings in the register take effect after the reset sequence.

Figure 7.3-1 shows the mode data configuration.

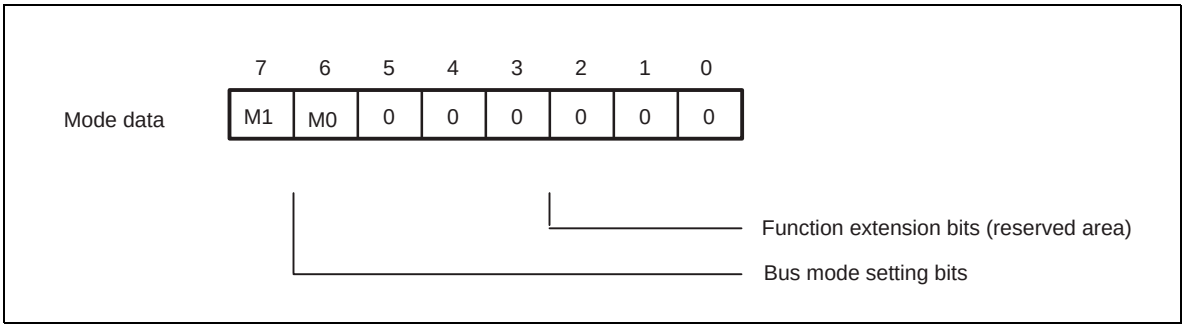


Figure 7.3-1 Mode data configuration

■ Bus mode setting bits

The bus mode setting bits specify operating mode after a reset sequence. Table 7.3-1 lists the relationship between the bits and functions.

Table 7.3-1 Bus mode setting bits and functions

M1	M0	Function	Remarks
0	0	Single-chip mode	8 bits
0	1	(Setting not allowed)	(Mode data)
1	0		
1	1		

<Check>

Because the MB90560 series is only used in single-chip mode, set M2, 1, 0 to 011 and set M1, 0 to 00.

Figure 7.3-2 shows the correspondence between access areas and physical addresses in single-chip mode.

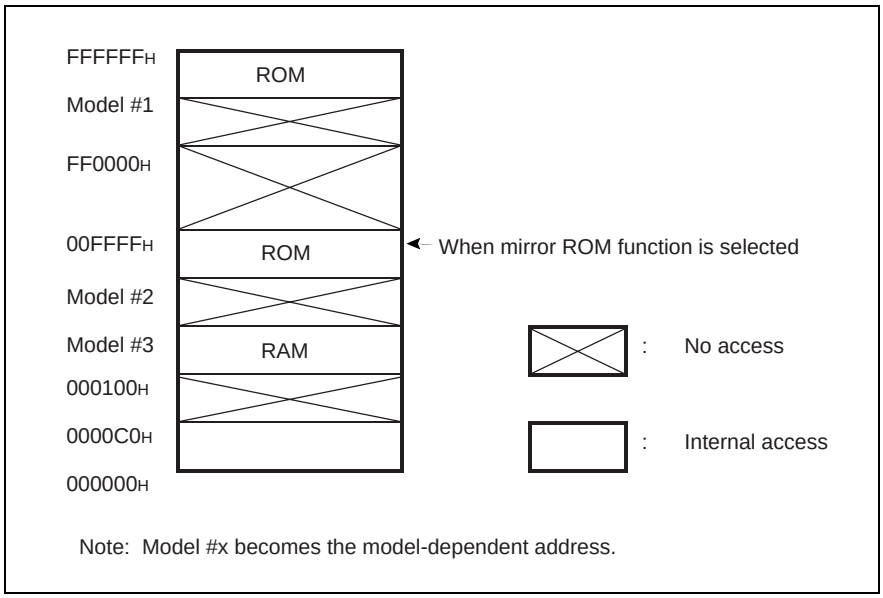


Figure 7.3-2 Correspondence between access areas and physical addresses in single-chip mode

■ Relationship between mode pins and mode data

Table 7.3-2 lists the relationship between mode pins and mode data.

Table 7.3-2 Relationship between mode pins and mode data

Mode	MD2	MD1	MD0	M1	M0
Single-chip mode	0	1	1	0	0

<Check>

The MB90560 series is only used in single-chip mode.

CHAPTER 8 I/O PORTS

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

8.1 Overview of I/O Ports

An I/O port can be used as a general-purpose I/O port (parallel I/O port). The MB90560 series has six ports (50 pins). The ports are also used for peripheral function I/O pins (peripheral function I/O pins).

■ I/O Port Functions

Each I/O port outputs data from the CPU to the I/O pins or inputs signals from the I/O pins to the CPU as directed by the port data register (PDR). Each I/O port can also be set the direction of a data flow (input or output) at the I/O pins for each bit using the port data direction register (DDR). The function of each port and the peripheral functions using it are described below:

- Port 0 : General-purpose I/O port
- Port 1 : General-purpose I/O port/peripheral function (external interrupt input pins)
- Port 2 : General-purpose I/O port/peripheral function (16-bit reload timer/ICU)
- Port 3 : General-purpose I/O port/peripheral function (OCU/UART0)
- Port 4 : General-purpose I/O port/peripheral function (UART0, 8-bit/16-bit PPG timer)
- Port 5 : General-purpose I/O port/peripheral function (analog input pins)
- Port 6 : General-purpose I/O port/peripheral function (UART1)

Table 8.1-1 summarizes the functions of individual ports.

Table 8.1-1 Functions of individual ports

Port	Pin	Input form	Output form	Function	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Port 0	P00~P07	CMOS (hysteresis)	CMOS pull-up resistor selectable	General I/O port	–	–	–	–	–	–	–	–	P07	P06	P05	P04	P03	P02	P01	P00
				Peripheral function	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Port 1	P10/INT0 ~P17/FRCK			General I/O port	P17	P16	P15	P14	P13	P12	P11	P10	–	–	–	–	–	–	–	–
				Peripheral function	FRCK	INT6	INT5	INT4	INT3	INT2	INT1	INT0	–	–	–	–	–	–	–	–
Port 2	P20/TIN0 ~P27/IN3		CMOS	General I/O port	–	–	–	–	–	–	–	–	P27	P26	P25	P24	P23	P22	P21	P20
				Peripheral function	–	–	–	–	–	–	–	–	IN3	IN2	IN1	IN0	TO1	TIN1	TO0	TIN0
Port 3	P30/RTO0 ~P37/SOT0			General I/O port	P37	P36	P35	P34	P33	P32	P31	P30	–	–	–	–	–	–	–	–
				Peripheral function	SOT0	SIN0	RTO5	RTO4	RTO3	RTO2	RTO1	RTO0	–	–	–	–	–	–	–	–
Port 4	P40/SCK0 ~P46/PPG5			General I/O port	–	–	–	–	–	–	–	–	–	P46	P45	P44	P43	P42	P41	P40
				Peripheral function	–	–	–	–	–	–	–	–	–	PPG5	PPG4	PPG3	PPG2	PPG1	PPG0	SCK0
Port 5	P50/AN0 ~P57/AN7	Analog/CMOS (hysteresis)	CMOS	General I/O port	P57	P56	P55	P54	P53	P52	P51	P50	–	–	–	–	–	–	–	–
				Analog input	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	–	–	–	–	–	–	–	–
Port 6	P60/SIN1 ~P63/INT7/ DTTI	Analog input	CMOS	General I/O port	–	–	–	–	–	–	–	–	–	–	–	–	P63	P62	P61	P60
				Peripheral function	–	–	–	–	–	–	–	–	–	–	–	–	INT7/ DTTI	SCK1	SOT1	SIN1

<Check>

Port 5 is also used as analog input pins. To use the port as a general-purpose port, be sure to reset the corresponding bit of the analog data input enable register (ADER) to “0”. Resetting the CPU sets the ADER register bits to “1”.

8.2 Port Register

This section provides a list of the registers related to the I/O port settings and shows how each port is assigned to the external pins.

■ Registers for I/O Ports

Table 8.2-1 is a list of the registers corresponding to individual ports.

Table 8.2-1 Registers and corresponding ports

Register	Read/Write	Address	Initial value
Port 0 data register (PDR0)	R/W	000000H	XXXXXXXXB
Port 1 data register (PDR1)	R/W	000001H	XXXXXXXXB
Port 2 data register (PDR2)	R/W	000002H	XXXXXXXXB
Port 3 data register (PDR3)	R/W	000003H	XXXXXXXXB
Port 4 data register (PDR4)	R/W	000004H	–XXXXXXXB
Port 5 data register (PDR5)	R/W	000005H	XXXXXXXXB
Port 6 data register (PDR6)	R/W	000006H	----XXXXB
Port 0 data direction register (DDR0)	R/W	000010H	00000000B
Port 1 data direction register (DDR1)	R/W	000011H	00000000B
Port 2 data direction register (DDR2)	R/W	000012H	00000000B
Port 3 data direction register (DDR3)	R/W	000013H	00000000B
Port 4 data direction register (DDR4)	R/W	000014H	–0000000B
Port 5 data direction register (DDR5)	R/W	000015H	00000000B
Port 6 data direction register (DDR6)	R/W	000016H	----0000B
Analog data input enable register (ADER)	R/W	000017H	11111111B
Port 0 pull-up resistor setting register (RDR0)	R/W	00008CH	00000000B
Port 1 pull-up resistor setting register (RDR1)	R/W	00008DH	00000000B

R/W:Read/write enabled

R:Read only

X:Undefined

–:Not used

8.3 Port 0

Port 0 is a general-purpose I/O port. This section provides the configuration of port 0, lists its pins, shows a block diagram of the pins, and describes the corresponding registers.

■ Port 0 Configuration

Port 0 consists of the following:

- General-purpose 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 lists the port 0 pins.

Table 8.3-1 Port 0 pins

Port	Pin	Port function		I/O form		Circuit type
				Input	Output	
Port 0	P00	P00	General-purpose I/O	CMOS (hysteresis)	CMOS	C
	P01	P01				
	P02	P02				
	P03	P03				
	P04	P04				
	P05	P05				
	P06	P06				
	P07	P07				

See Section 1.7, "I/O Circuit Types," for information on the circuit types.

■ Block Diagram of Port 0 Pins

Figure 8.3-1 is a block diagram of the port 0 pins.

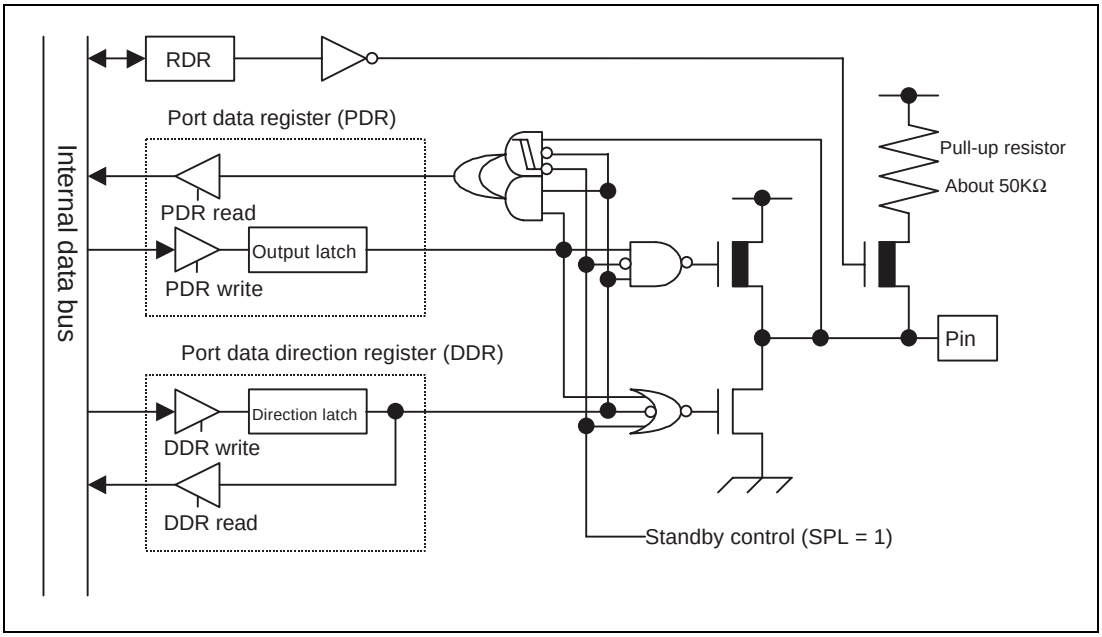


Figure 8.3-1 Block diagram of port 0 pins

■ 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 lists the port 0 pins and their corresponding register bits.

Table 8.3-2 Port 0 pins and their corresponding register bits

Port	Register bits and corresponding port pins								
Port 0	PDR0,DDR0,RDR0	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
	Corresponding pin	P07	P06	P05	P04	P03	P02	P01	P00

<Reference>

See Section 1.7,"I/O Circuit Types,"for information on the circuit types.

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 indicates the state of each pin of port 0.

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

  The DDR0 register specifies the direction of a data flow (input or output) at each pin (bit) of port “0”. When a DDR0 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.

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

  The RDR0 register specifies the selection of a pull-up resistor at each pin (bit) of port 0. When a RDR0 register bit is “1”, a pull-up resistor is selected for the corresponding port (pin). When the bit is “0”, the pull-up resistor is deselected.

  Table 8.3-3 lists the functions of the port 0 registers.

Table 8.3-3 Port 0 register functions

Register	Data	During reading	During writing	Read/ Write	Address	Initial value
Port 0 data register (PDR0)	0	The pin is at the low level.	The output latch is loaded with “0“. When the pin functions as an output port, the pin is set to the low level.	R/W	000000H	XXXXXXXXB
	1	The pin is at the high level.	The output latch is loaded with “1“. When the pin functions as an output port, the pin is set to the high level.			
Port 0 data direction register (DDR0)	0	The direction latch is “0“.	The output buffer is turned off to place the port in input mode.	R/W	000010H	00000000B
	1	The direction latch is “1“.	The output buffer is turned on to place the port in output mode.			
Port 0 pull-up resistor setting register (RDR0)	0	The setting latch is “0“.	The pull-up resistor is cut and the port is placed in the Hi-Z state in input mode.	R/W	00008CH	00000000B
	1	The setting latch is “1“.	The pull-up resistor is selected and the port is held at the high level in input mode.			

R/W : Read/write enabled

X : Undefined

Memo

8.3.2 Operation of Port 0

This section describes the operation of port 0.

■ Operation of Port 0

● Port operation in output mode

- Setting a bit of the DDR0 register to “1“ places the corresponding port pin in output mode.
- Data written to the PDR0 register in output mode is held in the output latch of the PDR and output to the port pins.
- The PDR0 register can be accessed in read mode to read the value at the port pins (the same value as in the output latch of the PDR).

<Check>

If a read-modify-write instruction (such as an instruction that sets bits) is used with the port data register, the target bits of the register are set to the specified value. The bits that have been specified for output using the DDR register are not affected, but for the bits that have been specified for input, a value input from the pins is written to the output latch and output as it is. Before switching the mode for the bits from input to output, therefore, write the output data to the PDR register, then specify output mode in the DDR register.

● Port operation in input mode

- Writing a bit of the DDR0 register to “0“ places the corresponding port pin in input mode.
- In input mode, the output buffer is turned off, and the pins are placed in a high impedance state.
- However, when the RDR0 register is set to “1“ to select a pull-up resistor, the pins are held at the high level.
- Data written to the PDR0 register in input mode is stored in the output latch of the PDR but is not output to the port pins.
- The PDR0 register can be accessed in read mode to read the level value (“0“ or “1“) at the port pins.

● Port operation after a reset

- When the CPU is reset, the DDR0 and PDR registers are initialized to “0“. As a result, the output buffer is turned off (I/O mode changes to input), the pull-up resistor is cut, and the pins are placed in a high impedance state.
- The PDR0 register is not initialized when the CPU is reset. To use the port in output mode, therefore, output mode must be specified in the DDR0 register after the output data is set in the PDR0 register.

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

If the pin state setting bit (SPL) in the low-power consumption mode control register (LPMCR) is already “1” when the CPU is shifted to stop or time-base timer mode, the port pins are placed in a high-impedance state. This is because the output buffer is turned off forcibly regardless of the value in the DDR0 register. Note that the inputs are fixed at a certain level to prevent leakage due to an open circuit.

Note also that when a pull-up resistor is selected, the port pins are held at the high level and not placed in a high-impedance state even when the SPL bit is set to “1”.

Table 8.3-4 lists the states of the port 0 pins.

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~P07	General-purpose I/O port	General-purpose I/O port	General-purpose I/O port	Input shut down/ output in Hi-z	Input shut down/held at H level

SPL : Pin state setting bit of low-power mode control register (LPMCR)

Hi-z : High impedance

8.4 Port 1

Port 1 is a general-purpose I/O port. It can also be used for peripheral function input. The port pins can be switched individually between the I/O port and peripheral function. This section focuses on the general I/O port function. The section provides the configuration of port 1, lists its pins, shows a block diagram of the pins, and describes the corresponding registers.

■ Port 1 Configuration

Port 1 consists of the following:

- General-purpose I/O pins/external interrupt 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 port 1 pins are also used as peripheral function input pins. The pins cannot be used as output port pins when they are used as peripheral function input pins. Table 8.4-1 lists the port 1 pins.

Table 8.4-1 Port 1 pins

Port	Pin	Port function		Peripheral function		I/O form		Circuit type
						Input	Output	
Port 1	P10/INT0	P10	General-purpose I/O	INT0	External interrupt input	CMOS (hysteresis)	CMOS	C
	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				

See Section 1.7, "I/O Circuit Types," for information on the circuit types.

■ Block Diagram of Port 1 Pins

Figure 8.4-1 is a block diagram of port 1 pins.

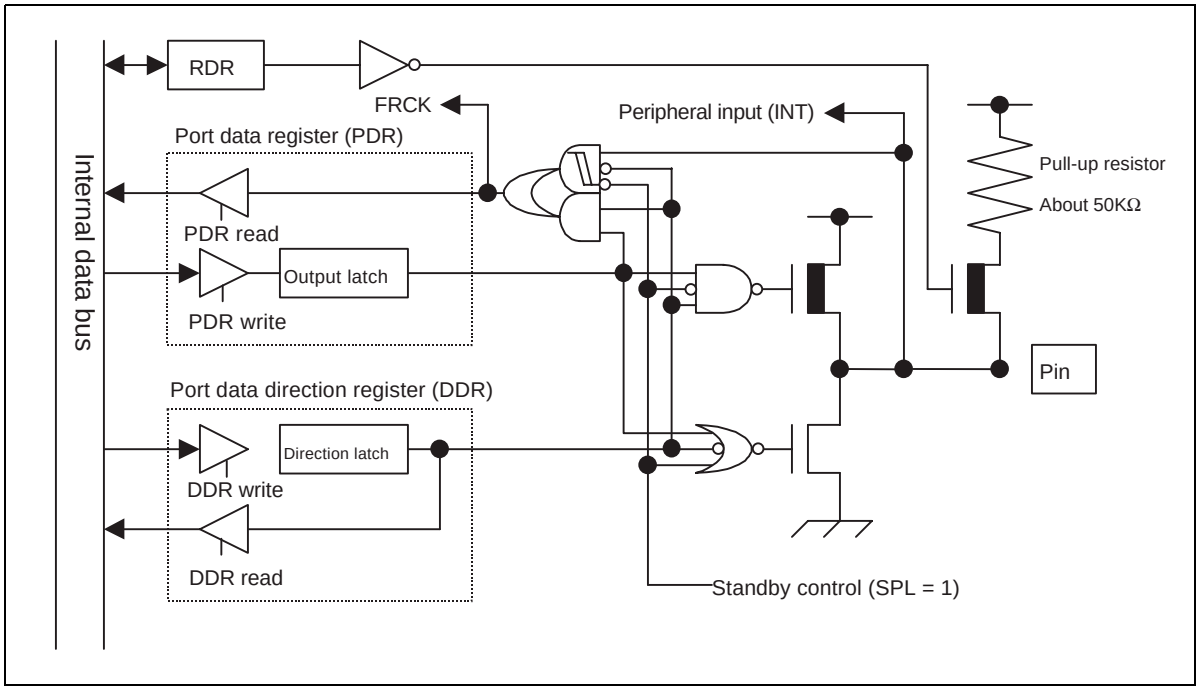


Figure 8.4-1 Block diagram of port 1 pins

■ 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 lists the port 1 pins and their corresponding register bits.

Table 8.4-2 Port 1 pins and their corresponding register bits

Port	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 indicates the state of each pin of port 1.

● Port 1 data direction register (DDR1)

The DDR1 register specifies the direction of a data flow (input or output) at each pin (bit) of port 1. When a DDR1 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.

● Port 1 pull-up resistor setting register (RDR1)

The RDR1 register specifies the selection of a pull-up resistor at each pin (bit) of port 1. When a RDR1 register bit is “1”, a pull-up resistor is selected for the corresponding port (pin). When the bit is “0”, the pull-up resistor is deselected.

<Caution>

To use a peripheral function having input pins, reset the port direction register bit corresponding to each peripheral function input pin to “0” to place the port in input mode.

Table 8.4-3 lists the functions of the port 1 registers.

Table 8.4-3 Port 1 register functions

Register	Data	During reading	During writing	Read/ Write	Address	Initial value
Port 1 data register (PDR1)	0	The pin is at the low level.	The output latch is loaded with “0”. When the pin functions as an output port, the pin is set to the low level.	R/W	000001H	XXXXXXXXB
	1	The pin is at the high level.	The output latch is loaded with “1”. When the pin functions as an output port, the pin is set to the high level.			
Port 1 data direction register (DDR1)	0	The direction latch is “0”.	The output buffer is turned off to place the port in input mode.	R/W	000011H	00000000B
	1	The direction latch is “1”.	The output buffer is turned on to place the port in output mode.			
Port 1 pull-up resistor setting register (RDR1)	0	The setting latch is “0”.	The pull-up resistor is cut and the port is placed in the Hi-Z state in input mode.	R/W	00008DH	00000000B
	1	The setting latch is “1”.	The pull-up resistor is selected and the port is held at the high level in input mode.			

R/W : Read/write enabled
X : Undefined

Memo

8.4.2 Operation of Port 1

This section describes the operation of port 1.

■ Operation of Port 1

● Port operation in output mode

- Setting a bit of the DDR1 register to “1” places the corresponding port pin in output mode.
- Data written to the PDR1 register in output mode is stored in the output latch of the PDR and output to the port pins as is.
- The PDR1 register can be accessed in read mode to read the value at the port pins (the same value as in the output latch of the PDR).

<Check>

If a read-modify-write instruction (such as an instruction that sets bits) is used with the port data register, the target bits of the register are set to the specified value. The bits that have been specified for output using the DDR register are not affected, but for the bits that have been specified for input, a value input from the pins is written to the output latch and output as it is. Before switching the mode for the bits from input to output, therefore, write the output data to the PDR register, then specify output mode in the DDR register.

● Port operation in input mode

- Writing a bit of the DDR1 register to “0” places the corresponding port pin in input mode.
- In input mode, the output buffer is turned off, and the pins are placed in a high impedance state.
- However, when the RDR1 register is set to “1” to select a pull-up resistor, the pins are held at the high level.
- Data written to the PDR1 register in input mode is stored in the output latch of the PDR but not output to the port pins.
- The PDR1 register can be accessed in read mode to read the level value (“0” or “1”) at the port pins.

● Port operation for peripheral function input

When the port is also used for peripheral function input, the value at the pins is always supplied as peripheral function inputs. To use an external signal for the peripheral function, reset the DDR1 register to “0” to place the port in input mode.

● Port operation after a reset

- When the CPU is reset, the DDR1 and PDR registers are initialized to “0”. As a result, the output buffer is turned off (I/O mode changes to input), the pull-up resistor is cut, and the pins are placed in a high impedance state.
- 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 pin state setting bit (SPL) in the low-power mode control register (LPMCR) is already “1” when the CPU is shifted to stop or time-base timer mode, the port pins are placed in a high-impedance state. This is because the output buffer is turned off forcibly regardless of the value in the DDR1 register. Note that the inputs are fixed at a certain level to prevent leakage due to an open circuit.

Note also that when a pull-up resistor is selected, the port pins are held at the high level and not placed in a high-impedance state even when the SPL bit is set to 1. Table 8.4-4 lists the states of the port 1 pins.

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/ INT0~P17 /FRCK	General-purpose I/O port	General-purpose I/O port	General-purpose I/O port	Input enabled/ output in Hi-z	Input shut down/ held at H level

SPL : Pin state setting bit of low-power mode control register (LPMCR)

Hi-z : High impedance

8.5 Port 2

Port 2 is a general-purpose I/O port. It can also be used for peripheral function input and output. The port pins can be switched in units of bits between the I/O port and the peripheral function. This section focuses on the general I/O port function. This section also provides the configuration of port 2, lists its pins, shows a block diagram of the pins, and describes the corresponding registers.

■ Port 2 Configuration

Port 2 consists of the following:

- General-purpose I/O pins/peripheral function 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 peripheral function I/O pins. The pins cannot be used as general-purpose I/O port pins when they are used as peripheral function I/O pins. Table 8.5-1 lists the port 2 pins.

Table 8.5-1 Port 2 pins

Port	Pin	Port function		Peripheral function		I/O form		Circuit type
						Input	Output	
Port 2	P20/TIN0	P20	General-purpose I/O	TIN0	16-bit reload timer 0 event input	CMOS (hysteresis)	CMOS	D
	P21/TO0	P21		TO0	16-bit reload timer 0 timer output			
	P22/TIN1	P22		TIN1	16-bit reload timer 1 event input			
	P23/TO1	P23		TO1	16-bit reload timer 1 timer output			
	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			

See Section 1.7, "I/O Circuit Types," for information on the circuit types.

■ Block Diagram of Port 2 Pins

Figure 8.5-1 is a block diagram of port 2 pins.

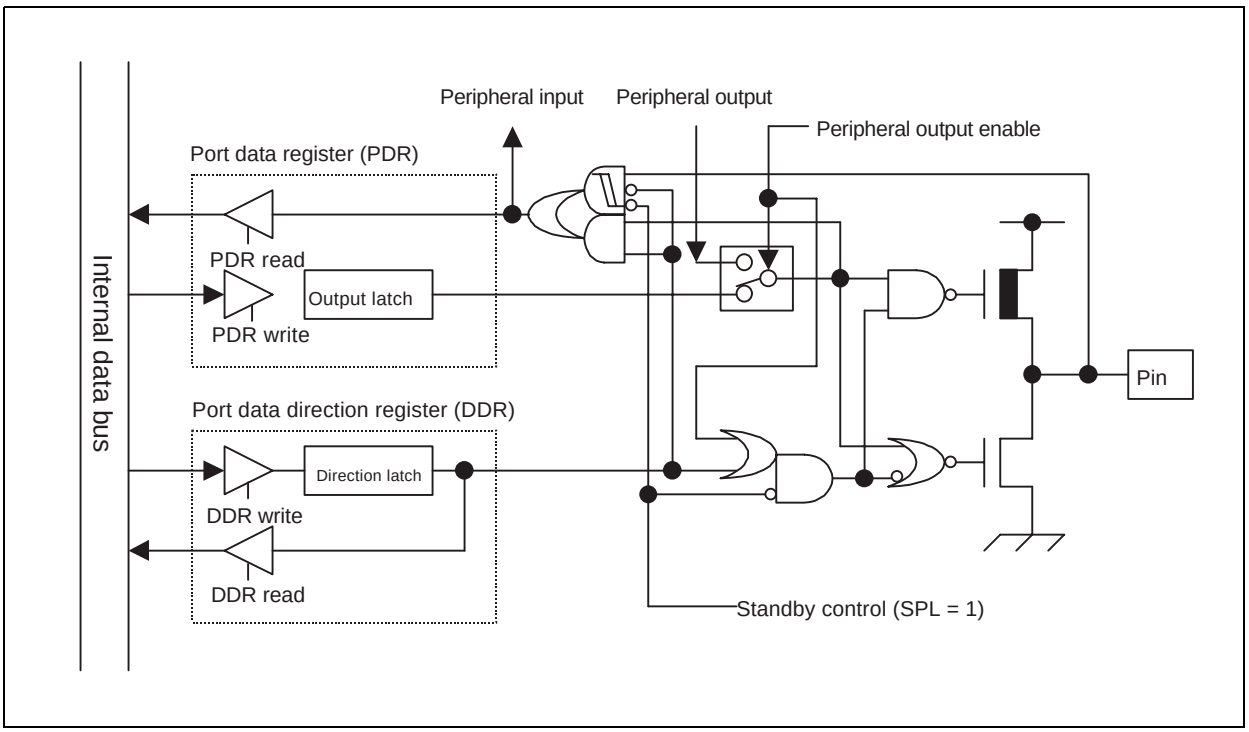


Figure 8.5-1 Block diagram of port 2 pins

<Caution>

When the peripheral function output enable bit is set, the port is forcibly caused to function as peripheral function output pins regardless of the value in the DDR2 register.

■ 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 lists the port 2 pins and their corresponding register bits.

Table 8.5-2 Port 2 pins and their corresponding register bits

Port	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 indicates the state of each pin of port 2.

● Port 2 data direction register (DDR2)

The DDR2 register specifies the direction of a data flow (input or output) at each pin (bit) of port 2. When a DDR2 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>

- When a peripheral function having output pins is used, the port functions as peripheral function output pins regardless of the value in the DDR2 register as long as the peripheral function output enable bit corresponding to the pins is set.
- To use a peripheral function having input pins, reset the DDR2 register bit corresponding to each peripheral function input pin to 0 to place the port in input mode.

Table 8.5-3 lists the functions of the port 2 registers.

Table 8.5-3 Port 2 register functions

Register	Data	During reading	During writing	Read/ Write	Address	Initial value
Port 2 data register (PDR2)	0	The pin is at the low level.	The output latch is loaded with “0”. When the pin functions as an output port, the pin is set to the low level.	R/W	000002H	XXXXXXXXB
	1	The pin is at the high level.	The output latch is loaded with “1”. When the pin functions as an output port, the pin is set to the high level.			
Port 2 data direction register (DDR2))	0	The direction latch is “0”.	The output buffer is turned off to place the port in input mode.	R/W	000012H	00000000B
	1	The direction latch is “1”.	The output buffer is turned on to place the port in output mode.			

R/W : Read/write enabled

X : Undefined

Memo

8.5.2 Operation of Port 2

This section describes the operation of port 2.

■ Operation of Port 2

● Port operation in output mode

- Setting a bit of the DDR2 register to “1” places the corresponding port pin in output mode.
- Data written to the PDR2 register in output mode is stored in the output latch of the PDR and output to the port pins as is.
- The PDR2 register can be accessed in read mode to read the value at the port pins (the same value as in the output latch of the PDR).

<Check>

If a read-modify-write instruction (such as an instruction that sets bits) is used with the port data register, the target bits of the register are set to the specified value. The bits that have been specified for output using the DDR register are not affected, but for the bits that have been specified for input, a value input from the pins is written to the output latch and output as it is. Before switching the mode for the bits from input to output, therefore, write output data to the PDR register, then specify output mode in the DDR register.

● Port operation in input mode

- Writing a bit of the DDR2 register to “0” places the corresponding port pin in input mode.
- In input mode, the output buffer is turned off, and the pins are placed in a high impedance state.
- Data written to the PDR2 register in input mode is store in the output latch of the PDR but not output to the port pins.
- The PDR2 register can be accessed in read mode to read the level value (0 or 1) at the port pins.

● Port operation for peripheral function output

The peripheral function output enable bit is set to enable the port to be used for peripheral function output. The state of the peripheral function enable bit takes precedence when specifying a switch between input and output. Even if a DDR2 register bit is “0”, the corresponding port pin is used for peripheral function output if the peripheral function has been enabled for output. Because the value at the pins can be read even if peripheral function output is enabled, the peripheral function output value can be read.

● Port operation for peripheral function input

When the port is also used for peripheral function input, the value at the pins is always supplied as peripheral function inputs. To use an external signal for the peripheral function, reset the DDR2 register to “0” to place the port in input mode.

● **Port operation after a reset**

- When the CPU is reset, the DDR2 register is initialized to “0”. As a result, the output buffer is turned off (I/O mode changes to input), the pins are placed in a high impedance state.
- The PDR2 register is not initialized when the CPU is reset. To use the port in output mode, therefore, output mode must be specified in the DDR2 register after the output data is set in the PDR2 register.

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

If the pin state setting bit (SPL) in the low-power mode control register (LPMCR) is already “1” when the CPU is shifted to stop or time-base timer mode, the port pins are placed in a high-impedance state. This is because the output buffer is turned off forcibly regardless of the value in the DDR2 register.

<Caution>

Note that the inputs are fixed at a certain level to prevent leakage due to an open circuit.

Table 8.5-4 lists the states of the port 2 pins.

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 = 0)	Stop mode or time-base timer mode (SPL = 1)
P20/INT0~ P27/IN3	General-purpose I/O port	General-purpose I/O port	General-purpose I/O port	Input enabled/ output in Hi-z	Input shut down/output in Hi-z

SPL : Pin state setting bit of low-power mode control register (LPMCR)

Hi-z : High impedance

8.6 Port 3

Port 3 is a general-purpose I/O port. It can also be used for peripheral function input and output. The port pins can be switched in units of bits between the I/O port and peripheral function. This section focuses on the general I/O port function. It provides the configuration of port 3, lists its pins, shows a block diagram of the pins, and describes the corresponding registers.

■ Port 3 Configuration

Port 3 consists of the following:

- General-purpose I/O pins/peripheral function I/O pins (P30/PPG0 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 peripheral function I/O pins. Therefore, the pins cannot be used as general-purpose I/O port pins when they are used as peripheral function I/O pins. Table 8.6-1 lists the port 3 pins.

Table 8.6-1 Port 3 pins

Port	Pin	Port function (single-chip mode)		Peripheral function		I/O form		Circuit type
						Input	Output	
Port 3	P30/RTO0	P30	General-purpose I/O	RTO0	OCU channel 0 output	CMOS (hysteresis)	CMOS	D
	P31/RTO1	P31		RTO1	OCU channel 1 output			
	P32/RTO2	P32		RTO2	OCU channel 2 output			
	P33/RTO3	P33		RTO3	OCU channel 3 output			
	P34/RTO4	P34		RTO4	OCU channel 4 output			
	P35/RTO5	P35		RTO5	OCU channel 5 output			
	P36/SIN0	P36		SIN0	UART0 data input			
	P37/SOT0	P37		SOT0	UART0 data output			

See Section 1.7, "I/O Circuit Types," for information on the circuit types.

■ Block Diagram of Port 3 Pins

Figure 8.6-1 is a block diagram of port 3 pins.

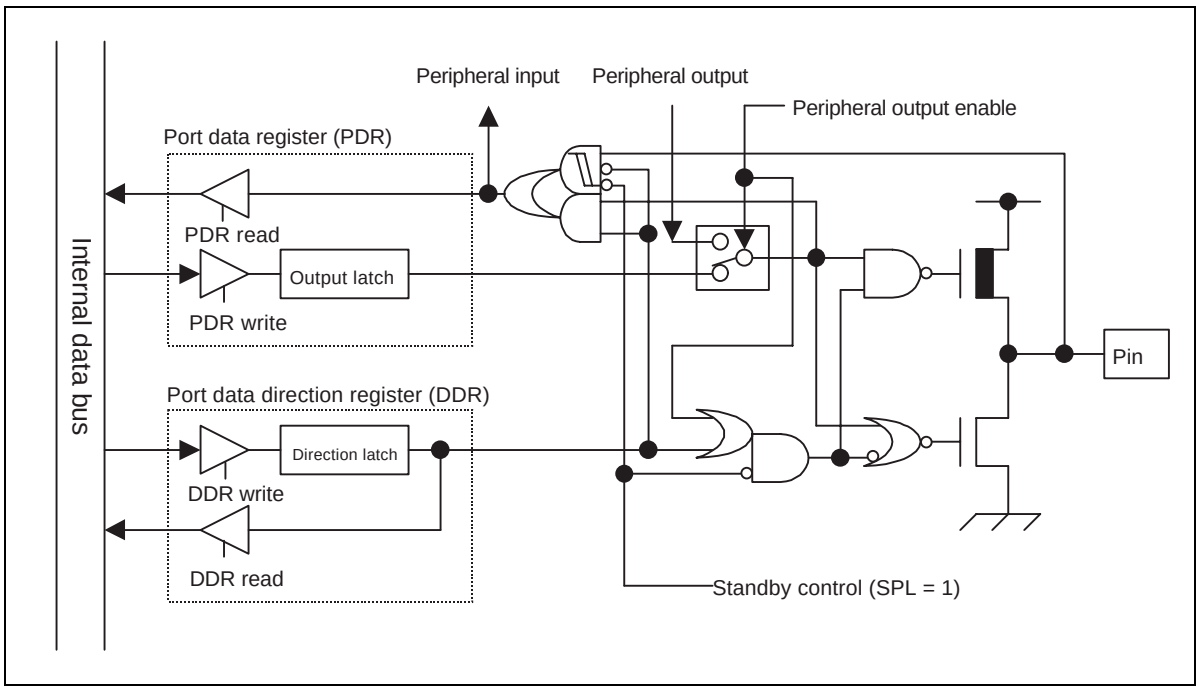


Figure 8.6-1 Block diagram of port 3 pins

■ 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 lists the port 3 pins and their corresponding register bits.

Table 8.6-2 Port 3 pins and their corresponding register bits

Port	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 indicates the state 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.

<Check>

- When a peripheral function having output pins is used, the port functions as peripheral function output pins regardless of the value in the DDR3 register as long as the peripheral function output enable bit corresponding to the pins is set.
- To use a peripheral function having input pins, reset the DDR3 register bit corresponding to each peripheral function input pin to “0” to place the port in input mode.

Table 8.6-3 lists the functions of the port 3 registers.

Table 8.6-3 Port 3 register functions

Register	Data	During reading	During writing	Read/ Write	Address	Initial value
Port 3 data register (PDR3)	0	The pin is at the low level.	The output latch is loaded with “0”. When the pin functions as an output port, the pin is set to the low level.	R/W	000003H	XXXXXXXXB
	1	The pin is at the high level.	The output latch is loaded with “1”. When the pin functions as an output port, the pin is set to the high level.			
Port 3 data direction register (DDR3)	0	The direction latch is “0”.	The output buffer is turned off to place the port in input mode.	R/W	000013H	00000000B
	1	The direction latch is “1”.	The output buffer is turned on to place the port in output mode.			

R/W : Read/write enabled

X : Undefined

Memo

8.6.2 Operation of Port 3

This section describes the operation of port 3.

■ Operation of Port 3

● Port operation in output mode

- Setting a bit of the DDR3 register to “1” places the corresponding port pin in output mode.
- Data written to the PDR3 register in output mode is stored in the output latch of the PDR and output to the port pins as is.
- The PDR3 register can be accessed in read mode to read the value at the port pins (the same value as in the output latch of the PDR).

<Check>

If a read-modify-write instruction (such as an instruction that sets bits) is used with the port data register, the target bits of the register are set to the specified value. The bits that have been specified for output using the DDR register are not affected, but for the bits that have been specified for input, a value input from the pins is written to the output latch and output as it is. Before switching the mode for the bits from input to output, therefore, write output data to the PDR register, then specify output mode in the DDR register.

● Port operation in input mode

- Writing a bit of the DDR3 register to “0” places the corresponding port pin in input mode.
- In input mode, the output buffer is turned off, and the pins are placed in a high impedance state.
- Data written to the PDR3 register in input mode is stored in the output latch of the PDR but not output to the port pins.
- The PDR3 register can be accessed in read mode to read the level value (“0” or “1”) at the port pins.

● Port operation for peripheral function output

The peripheral function output enable bit is set to enable the port to be used for peripheral function output. The state of the peripheral function enable bit takes precedence when specifying a switch between input and output. Even if a DDR3 register bit is “0”, the corresponding port pin is used for peripheral function output if the peripheral function has been enabled for output. Because the value at the pins can be read even if peripheral function output is enabled, the peripheral function output value can be read.

● Port operation for peripheral function input

When the port is also used for peripheral function input, the value at the pins is always supplied as peripheral function inputs. To use an external signal for the peripheral function, reset the DDR3 register to “0” to place the port in input mode.

● **Port operation after a reset**

- When the CPU is reset, the DDR3 register is initialized to “0”. As a result, the output buffer is turned off (I/O mode changes to input), the pins are placed in a high impedance state.
- The PDR3 register is not initialized when the CPU is reset. To use the port in output mode, therefore, output mode must be specified in the DDR3 register after the output data is set in the PDR3 register.

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

If the pin state setting bit (SPL) in the low-power mode control register (LPMCR) is already “1” when the CPU is shifted to stop or time-base timer mode, the port pins are placed in a high-impedance state. This is because the output buffer is turned off forcibly regardless of the value in the DDR3 register. Note that the inputs are fixed at a certain level to prevent leakage due to an open circuit. Table 8.6-4 lists the states of the port 3 pins.

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/RT00~ P37/SOT0	General-purpose I/O port	General-purpose I/O port	General-purpose I/O port	Input shut down/output in Hi-z

SPL : Pin state setting bit of low-power mode control register (LPMCR)

Hi-z : High impedance

8.7 Port 4

Port 4 is a general-purpose I/O port. It can also be used for peripheral function input and output. The port pins can be switched in units of bits between the I/O port and peripheral function. This section focuses on the general I/O port function. It provides the configuration of port 4, lists its pins, shows a block diagram of the pins, and describes the corresponding registers.

■ Port 4 Configuration

Port 4 consists of the following:

- General-purpose I/O pins/peripheral function I/O pins (P40/SCK0 to P45/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 peripheral function I/O pins. The pins cannot be used as general-purpose I/O port pins when they are used as peripheral function I/O pins. Table 8.7-1 lists the port 4 pins.

Table 8.7-1 Port 4 pins

Port	Pin	Port function (single-chip mode)		Peripheral function		I/O form		Circuit type
						Input	Output	
Port 4	P40/SCK0	P40	General-purpose I/O	SCK0	UART0 serial clock I/O	CMOS (hysteresis)	CMOS	D
	P41/PPG0	P41		PPG0	PPG0 output			
	P42/PPG1	P42		PPG1	PPG1 output			F
	P43/PPG2	P43		PPG2	PPG2 output			
	P44/PPG3	P44		PPG3	PPG3 output			
	P45/PPG4	P45		PPG4	PPG4 output			
	P46/PPG5	P46		PPG5	PPG5 output			
	–	–		–	–			

See Section 1.7, "I/O Circuit Types," for information on the circuit types.

■ Block Diagram of Port 4 Pins

Figure 8.7-1 is a block diagram of port 4 pins.

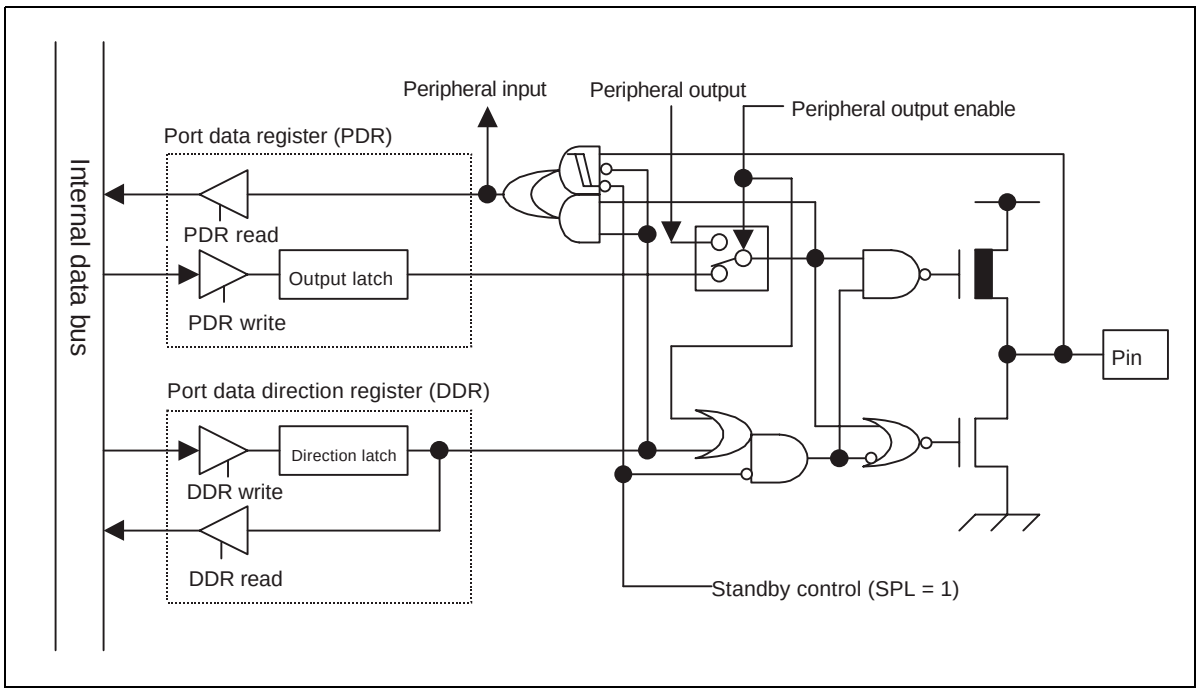


Figure 8.7-1 Block diagram of port 4 pins

<Check>

When the peripheral function output enable bit is set, the port is forcibly caused to function as peripheral function output pins regardless of the value in the DDR4 register.If valid DTTI is input when the RTO0 to RTO5 outputs and DTTI input are enabled, the value set in PDR4 is forcibly output.

■ 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 lists the port 4 pins and their corresponding register bits.

Table 8.7-2 Port 4 pins and their corresponding register bits

Port	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 indicates the state of each pin of port 4.

● Port 4 data direction register (DDR4)

The DDR4 register specifies the direction of a data flow (input or output) at each pin (bit) of port 4. When a DDR4 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.

<Check>

- When a peripheral function having output pins is used, the port functions as peripheral function output pins regardless of the value in the DDR4 register as long as the peripheral function output enable bit corresponding to the pins is set.
- To use a peripheral function having input pins, reset the DDR4 register bit corresponding to each peripheral function input pin to “0” to place the port in input mode.

Table 8.7-3 lists the functions of the port 4 registers.

Table 8.7-3 Port 4 register functions

Register	Data	During reading	During writing	Read/ Write	Address	Initial value
Port 4 data register (PDR4)	0	The pin is at the low level.	The output latch is loaded with “0”. When the pin functions as an output port, the pin is set to the low level.	R/W	000004H	–XXXXXXXB
	1	The pin is at the high level.	The output latch is loaded with “1”. When the pin functions as an output port, the pin is set to the high level.			
Port 4 data direction register (DDR4)	0	The direction latch is “0”.	The output buffer is turned off to place the port in input mode.	R/W	000014H	–0000000B
	1	The direction latch is “1”.	The output buffer is turned on to place the port in output mode.			

R/W : Read/write enabled

X : Undefined

– : Empty bit

Memo

8.7.2 Operation of Port 4

This section describes the operation of port 4.

■ Operation of Port 4

● Port operation in output mode

- Setting a bit of the DDR4 register to “1” places the corresponding port pin in output mode.
- Data written to the PDR4 register in output mode is stored in the output latch of the PDR and output to the port pins as is.
- The PDR4 register can be accessed in read mode to read the value at the port pins (the same value as in the output latch of the PDR).

<Check>

If a read-modify-write instruction (such as an instruction that sets bits) is used with the port data register, the target bits of the register are set to the specified value. The bits that have been specified for output using the DDR register are not affected, but for the bits that have been specified for input, a value input from the pins is written to the output latch and output as it is. Before switching the mode for the bits from input to output, therefore, write output data to the PDR register, then specify output mode in the DDR register.

● Port operation in input mode

- Writing a bit of the DDR4 register to “0” places the corresponding port pin in input mode.
- In input mode, the output buffer is turned off, and the pins are placed in a high impedance state.
- Data written to the PDR4 register in input mode is stored in the output latch of the PDR but not output to the port pins.
- The PDR4 register can be accessed in read mode to read the level value (“0” or “1”) at the port pins.

● Port operation for peripheral function output

The peripheral function output enable bit is set to enable the port to be used for peripheral function output. The state of the peripheral function 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 peripheral function output if the peripheral function has been enabled for output. Because the value at the pins can be read even if peripheral function output is enabled, the peripheral function output value can be read.

● Port operation for peripheral function input

When the port is also used for peripheral function input, the value at the pins is always supplied as peripheral function inputs. To use an external signal for the peripheral function, reset the DDR4 register to “0” to place the port in input mode.

● **Port operation after a reset**

- When the CPU is reset, the DDR4 register is initialized to “0”. As a result, the output buffer is turned off (I/O mode changes to input), the pins are placed in a high impedance state.
- The PDR4 register is not initialized when the CPU is reset. To use the port in output mode, therefore, output mode must be specified in the DDR4 register after the output data is set in the PDR4 register.

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

If the pin state setting bit (SPL) in the low-power mode control register (LPMCR) is already “1” when the port is shifted to stop or time-base timer mode, the port pins are placed in a high-impedance state. This is because the output buffer is turned off forcibly regardless of the value in the DDR4 register.

<Caution>

Note that the inputs are fixed at a certain level to prevent leakage due to an open circuit.

Table 8.7-4 lists the states of the port 4 pins.

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)	Hardware standby mode
P40/SIN0~P47/ ATG	General-purpose I/O port	General-purpose I/O port	General-purpose I/O port	Input shut down/ output in Hi-z	Input shut down/output in Hi-z

SPL : Pin state setting bit of low-power mode control register (LPMCR)

Hi-z : High impedance

8.8 Port 5

Port 5 is a general-purpose I/O port. It can also be used for A/D converter analog input. The port pins can be switched in units of bits between the I/O port and analog input. This section focuses on the general I/O port function. It provides the configuration of port 5, lists its pins, shows a block diagram of the pins, and describes the corresponding registers.

■ Port 5 Configuration

Port 5 consists of the following:

- General-purpose I/O pins/analog input pins (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 port 5 I/O pins are also used as analog input pins. The pins cannot be used as general-purpose I/O port pins when they are used for analog input. Similarly, the port 5 I/O pins cannot be used for analog input when they are used as a general-purpose I/O port. Table 8.8-1 lists the port 5 pins.

Table 8.8-1 Port 5 pins

Port	Pin	Port function (single-chip mode)		Peripheral function		I/O form		Circuit type
						Input	Output	
Port 5	P50/AN0	P50	General-purpose I/O	AN0	Analog input 0	CMOS (hysteresis)	CMOS	E
	P51/AN1	P51		AN1	Analog input 1			
	P52/AN2	P52		AN2	Analog input 2			
	P53/AN3	P53		AN3	Analog input 3			
	P54/AN4	P54		AN4	Analog input 4			
	P55/AN5	P55		AN5	Analog input 5			
	P56/AN6	P56		AN6	Analog input 6			
	P57/AN7	P57		AN7	Analog input 7			

See Section 1.7, "I/O Circuit Types," for information on the circuit types.

■ Block Diagram of Port 5 Pins

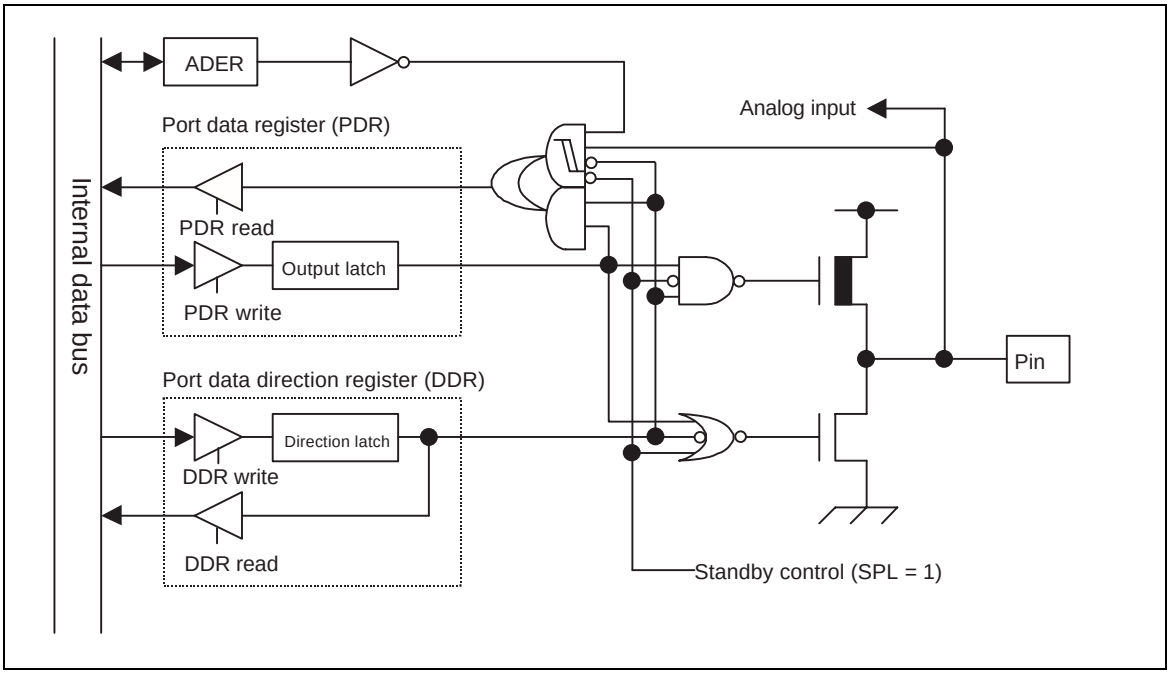


Figure 8.8-1 Block diagram of port 5 pins

<Check>

For a pin used as an input port, reset the corresponding PDR5 register bit to “0”, and also reset the corresponding ADER register bit to “0”.

For an pin used as an analog input pin, reset the corresponding DDR5 register bit to “0” and set the corresponding ADER register bit to “1”. In this case, the value read from the PDR5 register is “0”.

■ 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 lists the port 5 pins and their corresponding register bits.

Table 8.8-2 Port 5 pins and their corresponding register bits

Port	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 indicates the state 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 specifies whether the corresponding port 5 pin is to be used as a general-purpose I/O port or an analog input pin. Setting an ADER bit to “1” enables the corresponding pin for analog input. Setting the bit to “0” enables the pin for general-purpose I/O.

  <Check>

  If a signal at an intermediate level is input in port I/O mode, input leak current flows. Therefore, for a pin used for analog input, be sure to set the corresponding ADER bit to “1” for analog input.

  <Reference>

  When the CPU is reset, the DDR5 register is reset to “0” and the ADER register is set to “1” for analog input.

Table 8.8-3 lists the functions of the port 5 registers.

Table 8.8-3 Port 5 register functions

Register	Data	During reading	During writing	Read/ Write	Address	Initial value
Port 5 data register (PDR5)	0	The pin is at the low level.	Low level is output to the pin. (The output latch is loaded with “0: to turn on the output transistor.)	R/W	000005H	XXXXXXXB
	1	The pin is at the high level.	The pin is placed in a high-impedance state. (The output latch is loaded with “1” to turn off the output transistor.)			
Port 5 data direction register (DDR5)	0	The direction latch is “0”.	The output buffer is turned off to place the port in input mode.	R/W	000015H	00000000B
	1	The direction latch is “1”.	The output buffer is turned on to place the port in output mode.			
Analog input enable register (ADER)	0	Port I/O mode		R/W	000017H	11111111B
	1	Analog input mode				

R/W : Read/write enabled

X : Undefined

Memo

8.8.2 Operation of Port 5

This section describes the operation of port 5.

■ Operation of Port 5

● Port operation in output mode

- Setting a bit of the DDR5 register to “1” places the corresponding port pin in output mode.- Data written to the PDR5 register in output mode is held in the output latch of the PDR and output to the port pins.
- The PDR5 register can be accessed in read mode to read the value at the port pins (the same value as in the output latch of the PDR).

<Check>

If a read-modify-write instruction (such as an instruction that sets bits) is used with the port data register, the target bits of the register are set to the specified value. The bits that have been specified for output using the DDR register are not affected, but for the bits that have been specified for input, a value input from the pins is written to the output latch and output as it is. Before switching the mode for the bits from input to output, therefore, write the output data to the PDR register, then specify output mode in the DDR register.

● Port operation in input mode

- Writing a bit of the DDR5 register to “0” places the corresponding port pin in input mode.
- In input mode, the output buffer is turned off, and the pins are placed in a high impedance state.
- Data written to the PDR5 register in input mode is stored in the output latch of the PDR but not output to the port pins.
- The PDR5 register can be accessed in read mode to read the level value (“0” or “1”) at the port pins.

● Port operation for analog input

To use a port pin for analog input, write “1” to the corresponding ADER bit. Doing so disables the port from operating as a general-purpose port pin and enables it to function as an analog input pin. When PDR5 is accessed in read mode in this situation, a value of “0” is read.

● Port operation after a reset

When the CPU is reset, the DDR5 register is initialized to “0” and the ADER register is initialized to “1” to place the port in analog input mode. To use the port as a general-purpose port, write “0” to the ADER register in advance to place the port in port I/O mode.

● Port operation in stop or time-base timer mode

If the pin state setting bit (SPL) in the low-power mode control register (LPMCR) is already “1” when the CPU is shifted to stop or time-base timer mode, the port pins are placed in a high-impedance state. This is because the output buffer is turned off forcibly. Note that the inputs are fixed at a certain level to prevent leakage due to an open circuit.

Table 8.8-4 lists the states of the port 5 pins.

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/AN0~P57/AN7	General-purpose I/O port	General-purpose I/O port	General-purpose I/O port	Input shut down/output in Hi-z

SPL : Pin state setting bit of low-power mode control register (LPMCR)

Hi-z : High impedance

8.9 Port 6

Port 6 is a general-purpose I/O port. It can also be used for peripheral function input and output. The port pins can be switched in units of bits between the I/O port and peripheral function. This section focuses on the general I/O port function. It provides the configuration of port 6, lists its pins, shows a block diagram of the pins, and describes the corresponding registers.

■ Port 6 Configuration

Port 6 consists of the following:

- General-purpose I/O pins/peripheral function I/O pins (P60/SIN1 to P62/SCK1)
- Port 6 data register (PDR6)
- Port 6 data direction register (DDR6)

■ Port 6 Pins

The port 6 I/O pins are also used as peripheral function I/O pins. The pins cannot be used as general-purpose I/O port pins when they are used as peripheral function I/O pins. Table 8.9-1 lists the port 6 pins.

Table 8.9-1 Port 6 pins

Port	Pin	Port function (single-chip mode)		Peripheral function		I/O form		Circuit type
						Input	Output	
Port 6	P60/SIN1	P60	General-purpose I/O	SIN1	UART1 data input	CMOS (hysteresis)	CMOS	D
	P61/SOT1	P61		SOT1	UART1 data output			
	P62/SCK1	P62		SCK1	UART1 serial clock I/O			
	P63/INT7/ DTTI	P63		INT7	External interrupt input/ DTTI input			
	–	–		–	–			
	–	–		–	–			
	–	–		–	–			
	–	–		–	–			

See Section 1.7, "I/O Circuit Types," for information on the circuit types.

■ Block Diagram of Port 6 Pins

Figure 8.9-1 is a block diagram of port 6 pins.

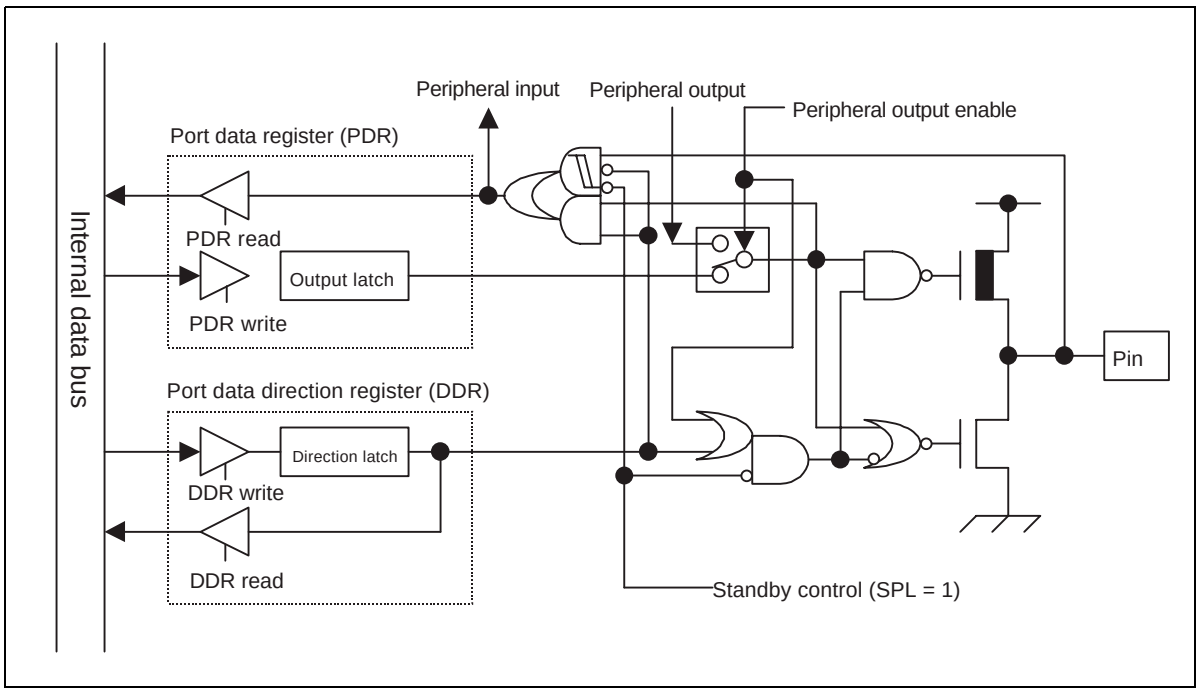


Figure 8.9-1 Block diagram of port 6 pins

■ 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 lists the port 6 pins and their corresponding register bits.

Table 8.9-2 Port 6 pins and their corresponding register bits

Port	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 indicates the state 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.

Check:

To use a peripheral function having input pins, reset the DDR6 register bit corresponding to each peripheral function input pin to “0” to place the port in input mode.

Table 8.9-3 lists the functions of the port 6 registers.

Table 8.9-3 Port 6 register functions

Register	Data	During reading	During writing	Read/ Write	Address	Initial value
Port 6 data register (PDR6)	0	The pin is at the low level.	The output latch is loaded with “0”. When the pin functions as an output port, the pin is set to the low level.	R/W	000006H	----XXXXB
	1	The pin is at the high level.	The output latch is loaded with “1”. When the pin functions as an output port, the pin is set to the high level.			
Port 6 data direction register (DDR6)	0	The direction latch is “0”.	The output buffer is turned off to place the port in input mode.	R/W	000016H	----0000B
	1	The direction latch is “1”.	The output buffer is turned on to place the port in output mode.			

R/W : Read/write enabled

X : Undefined

– : Empty bit

Memo

8.9.2 Operation of Port 6

This section describes the operation of port 6.

■ Operation of Port 6

● Port operation in output mode

- Setting a bit of the DDR6 register to “1” places the corresponding port pin in output mode.
- Data written to the PDR6 register in output mode is stored in the output latch of the PDR and output to the port pins as is.
- The PDR6 register can be accessed in read mode to read the value at the port pins (the same value as in the output latch of the PDR).

<Check>

If a read-modify-write instruction (such as an instruction that sets bits) is used with the port data register, the target bits of the register are set to the specified value. The bits that have been specified for output using the DDR register are not affected, but for the bits that have been specified for input, a value input from the pins is written to the output latch and output as it is. Before switching the mode for the bits from input to output, therefore, write the output data to the PDR register, then specify output mode in the DDR register.

● Port operation in input mode

- Resetting a bit of the DDR6 register to “0” places the corresponding port pin in input mode.
- In input mode, the output buffer is turned off, and the pins are placed in a high impedance state.
- Data written to the PDR6 register in input mode is held in the output latch of the PDR but not output to the port pins.
- The PDR6 register can be accessed in read mode to read the level value (“0” or “1”) at the port pins.

● Port operation for peripheral function output

The peripheral function output enable bit is set to enable the port to be used for peripheral function output. The state of the peripheral function enable bit takes precedence when specifying a switch between input and output. Even if a DDR6 register bit is “0”, the corresponding port pin is used for peripheral function output if the peripheral function has been enabled for output. Because the value at the pins can be read even if peripheral function output is enabled, the peripheral function output value can be read.

● Port operation for peripheral function input

When the port is also used for peripheral function input, the value at the pins is always supplied as peripheral function inputs. To use an external signal for the peripheral function, reset the DDR6 register to “0” to place the port in input mode.

● **Port operation after a reset**

- When the CPU is reset, the DDR6 register is initialized to “0”. As a result, the output buffer is turned off (I/O mode changes to input), the pins are placed in a high impedance state.
- 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 pin state setting bit (SPL) in the low-power mode control register (LPMCR) is already “1” when the port is shifted to stop or time-base timer mode, the port pins are placed in a high-impedance state. This is because the output buffer is turned off forcibly regardless of the value in the DDR6 register.

<Caution>

Note that the inputs are fixed at a certain level to prevent leakage due to an open circuit.

Table 8.9-4 lists the states of the port 6 pins.

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/SIN1~ P63/INT7/DTTI	General-purpose I/O port	General-purpose I/O port	General-purpose I/O port	Input shut down/output in Hi-z

SPL : Pin state setting bit of low-power mode control register (LPMCR)

Hi-z : High impedance

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 is an example of connecting the eight-segment LED to the MB90560 ports.

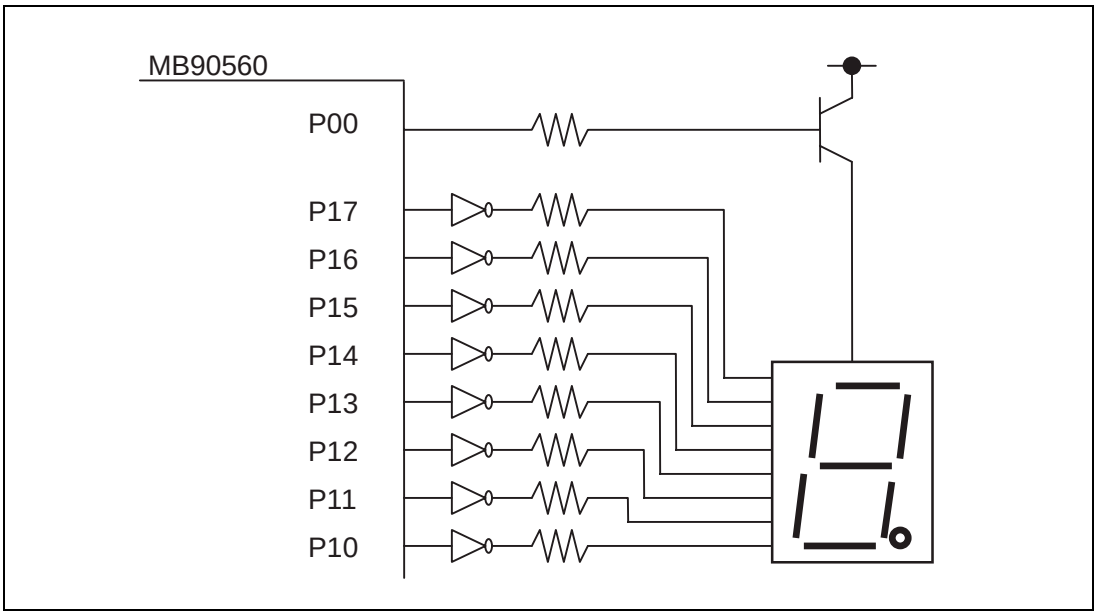


Figure 8.10-1 Example of eight-segment LED connection

● 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
```

Memo

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-run counter (timebase counter) that counts up in synchronization with the internal count clock (one-half of the source oscillation). The timer has an interval timer function that can select four intervals.
The timebase timer also has functions for timer output of the oscillation stabilization wait interval and for supplying the clocks for 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.
- Four types of the interval timer (interval) can be selected. Table 9.1-1 lists the intervals for the timebase timer.

Table 9.1-1 Intervals for the timebase timer

Internal count 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
Values in parentheses are for a 4 MHz oscillation clock.

■ Clock supply function

The clock supply function supplies clocks to the oscillation stabilization wait interval timer and to some peripheral functions.

Table 9.1-2 lists the cycle times of clocks supplied from the timebase timer to each peripheral.

Table 9.1-2 Clock cycle time supplied from the timebase timer

Clock destination	Clock cycle time	Remarks
Oscillation stabilization wait interval	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)	
PPG timer	2^{9}/HCLK (Approx. 128 µs)	-

HCLK: Oscillation clock
Values in parentheses occurs during operation of the 4 MHz oscillation clock.

<Reference>

The oscillation stabilization wait interval 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 shows the block diagram of the timebase timer.

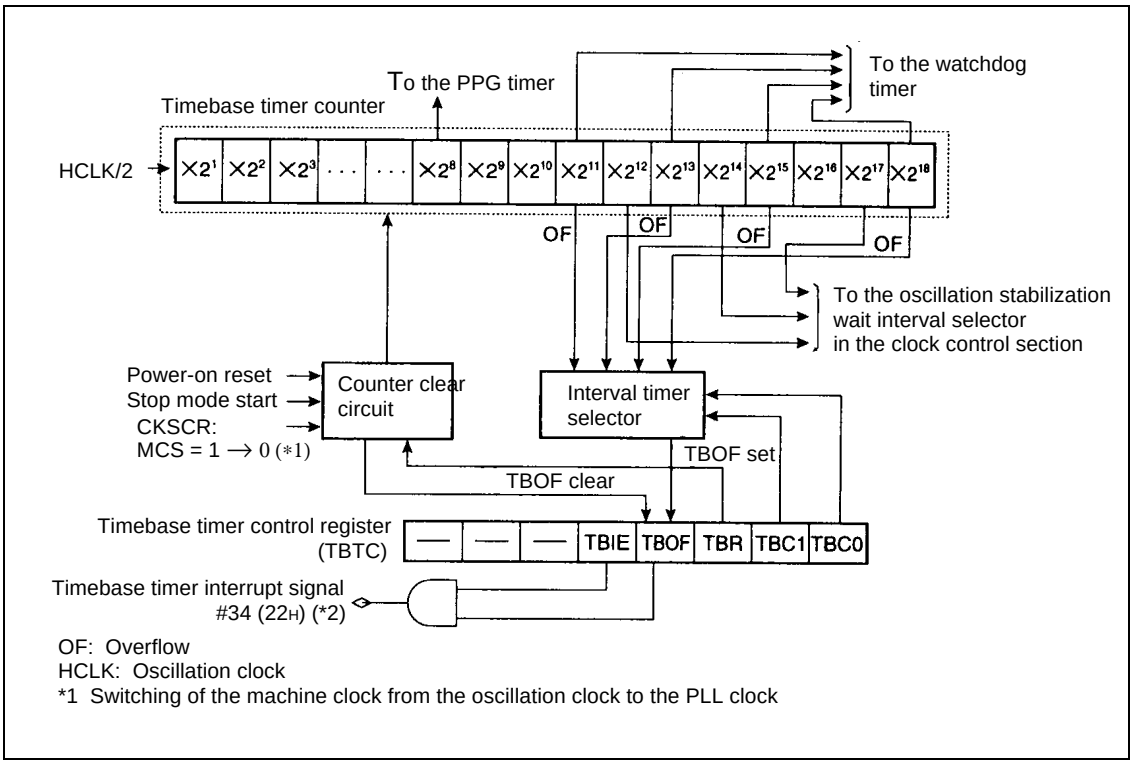


Figure 9.2-1 Block diagram of the timebase timer

- **Timebase timer counter**

  This 18-bit up counter uses the divide-by-two clock of the oscillation clock (HCLK) as the count clock.

- **Counter clear circuit**

  Used to clear the counter by writing “0” to the TBTC TBR bit, by a power-on reset, or by transition to stop mode (LPMCR:STP = 1).

- **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 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 counter, controls interrupts, and checks the status.

■ **Timebase timer control register (TBTC)**

	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7 ---------- bit0	
Address $0000A9_H$	RESV	-	-	TBIE	TBOF	TBR	TBC1	TBC0	(WDTC)	Initial value $1- - 00100_B$
	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	
	During reading	During writing
0	-	Clearing of the timebase timer counter and TBOF bit
1	The read value is always "1".	No change, no effect on other bits.

TBOF	Interrupt request flag bit	
	During reading	During writing
0	No overflow from the specified bit	Clearing of this bit
1	Overflow from the specified bit	No change, no effect on other bits.

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

Figure 9.3-1 Timebase timer control register (TBTC)

Table 9.3-1 Function description of each bit in the timebase timer control register (TBTC)

Bit name		Function
bit15	RESV: Reserved bit	**<Caution>** 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	• Used to enable or disable the output of an interrupt request to the CPU. • When this bit and the interrupt request flag bit (TBOF) are “1”, an interrupt request is output.
bit11	TBOF: Interrupt request flag bit	• This bit is set to 1 when the bit specifying the timebase timer counter overflows. • When this bit and the interrupt request enable bit (TBIE) are “1”, an interrupt request is output. • During writing, this bit is cleared with “0”. If “1“ is written, the bit does not change and there is no effect. **<Caution>** • To clear the TBOF bit, disable the timebase timer interrupt by specifying the TBIE bit or processor status (PS) ILM bit. • The TBOF bit is cleared by writing “0”, by a transition to stop mode, by clearing of the timebase timer with the TBR bit, or by a reset.
bit10	TBR: Timebase timer initial-ization 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 generate an interrupt request when the bit specifying the timebase timer counter overflows. (Interval timer function)

■ Timebase timer interrupts

The interrupt request flag bit (TBTC:TBOF) is set to “1” when the timebase timer counter counts up with the internal count clock and when the bit for the selected interval timer bit overflows. If the interrupt request enable bit has been enabled (TBTC:TBIE = 1), an interrupt request (#36) is generated in the CPU. Writing “0” to the TBOF bit in the interrupt handling routine clears the interrupt request. When the specified bit overflows, the TBOF bit is set regardless of the TBIE bit value.

<Check>

Clear the interrupt request flag bit (TBTC:TBOF) while a timebase timer interrupt is disabled by setting the TBIE bit or the processor status (PS) ILM bit.

Reference

- When the TBOF bit is “1”, if the TBIE bit status is switched from disable to enable (0 → 1), an interrupt request occurs immediately.
- The timebase timer cannot use the extended intelligent I/O service (EI²OS).

■ Timebase timer interrupts and EI²OS

Table 9.4-1 lists the timebase timer interrupt and EI2OS.

Table 9.4-1 Timebase interrupts and EI²OS

Interrupt number	Interrupt level setting register		Vector table address			EI²OS
	Register name	Address	Lower	Upper	Bank	
#36 (24H)	ICR12	0000BCH	FFFF6CH	FFFF6DH	FFFF6EH	x

x: Not available

<Check>

ICR12 is common to the timebase timer interrupt and input capture ch 2/3 wake-up interrupt. Interrupts can be used for two applications, but the interrupt level is the same.

Memo

9.5 Operation of the Timebase Timer

The timebase timer provides the interval timer function and the clock supply function that supplies clocks to some peripheral functions.

■ **Operation of the interval timer function (timebase timer)**

The interval timer function generates an interrupt request for each interval

The setting in Figure 9.5-1 is required for timebase timer to operate as an interval timer.

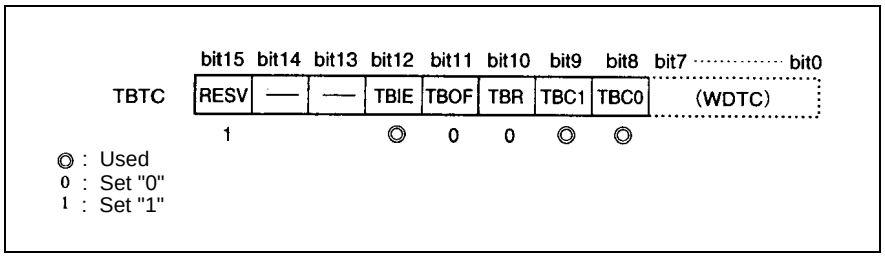


Figure 9.5-1 Setting of the timebase timer

- The timebase timer counter continues counting up in synchronization with the internal count clock (one-half of the oscillation clock) as long as the clock is being oscillated.
- When the counter is cleared (TBR = 0), it counts up from “0”. When the interval timer bit overflows, the interrupt request flag bit (TBOF) is set to “1”. If interrupt request output has been enabled (TBIE = 1), an interrupt is generated for each selected interval based on the cleared time.
- 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 stabilization wait interval timer for oscillation and the PLL clocks.

The oscillation settling time is set for the interval from the time the counter counts up from “0” (count clear) until the oscillation stabilization wait interval bit overflows. When control returns from timebase timer mode to PLL clock mode, the oscillation stabilization wait interval starts from the middle of counting because the timebase timer counter has been not cleared. Table 9.5-1 shows the clearing of the timebase counter and the oscillation settling times.

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			
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 → 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.

9.6 Usage Notes on the Timebase Timer

Notes about the effects on peripheral functions of clearing interrupt requests and the timebase timer are given below.

■ Timebase timer usage notes

● Clearing interrupt requests

The TBOF bit of the timebase timer control register must be cleared while a timebase timer interrupt is masked by the TBIE bit or the interrupt level mask register (ILM) of the processor status (PS).

● Effects of timebase timer clearing

Clearing of the timebase timer counter affects the following operations:

- When the timebase timer is using the interval timer function (interval interrupt)
- When the watchdog timer is being used

● Use of the timebase timer as the oscillation settling time timer

At power-on, the source oscillation of the main clock stops in main stop mode. After oscillator operation starts, the operating clock supplied by the timebase timer is used to take the oscillation stabilization wait time of the main clock. An appropriate oscillation stabilization wait time must be selected based on the type of oscillating element connected to the main clock oscillator (clock generation section). See Section 4.5, "Oscillation Stabilization Wait Time," for details.

● Notes on peripheral functions to which clocks are supplied from the timebase timer

In the mode in which the main clock source oscillation stops, the counter is cleared and timebase timer operation stops. When the timebase timer counter is cleared, the clock supplied from the timebase timer is supplied from its initial state. As a result, the H level is shortened and the L level lengthened 1/2 cycle. Although the clock for the watchdog timer is also supplied from its initial state; the watchdog timer operates in normal cycles because the watchdog timer counter is cleared at the same time.

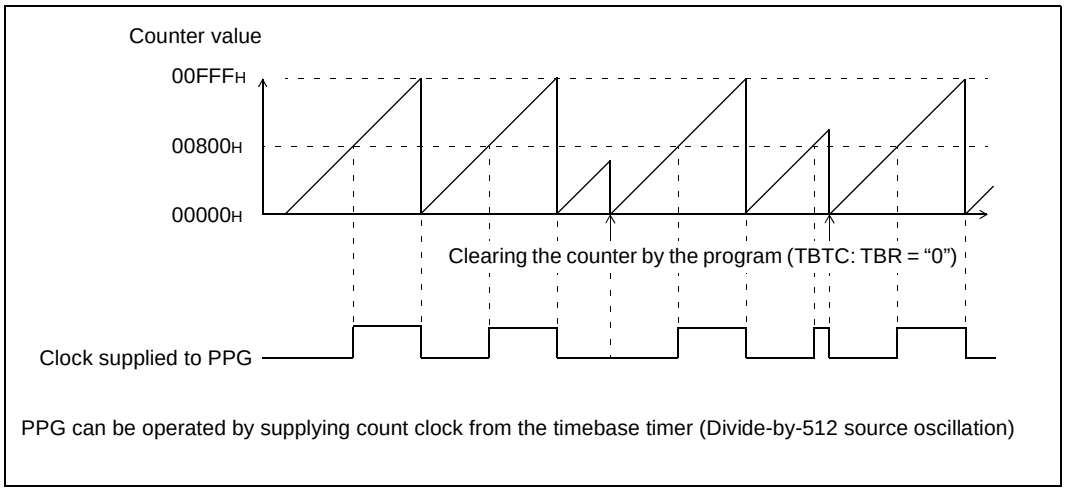


Figure 9.6-1 Effect on PPG when clearing timebase timer

■ Operation of the timebase timer

The following operations are shown in Figure 9.6-2:

- A power-on reset occurs.
- Sleep mode is entered during operation of the interval timer function.
- A counter clear request is issued.

When stop mode is entered, the timebase timer is cleared and its operation stops. On return from stop mode, the timebase timer counts the oscillation settling time.

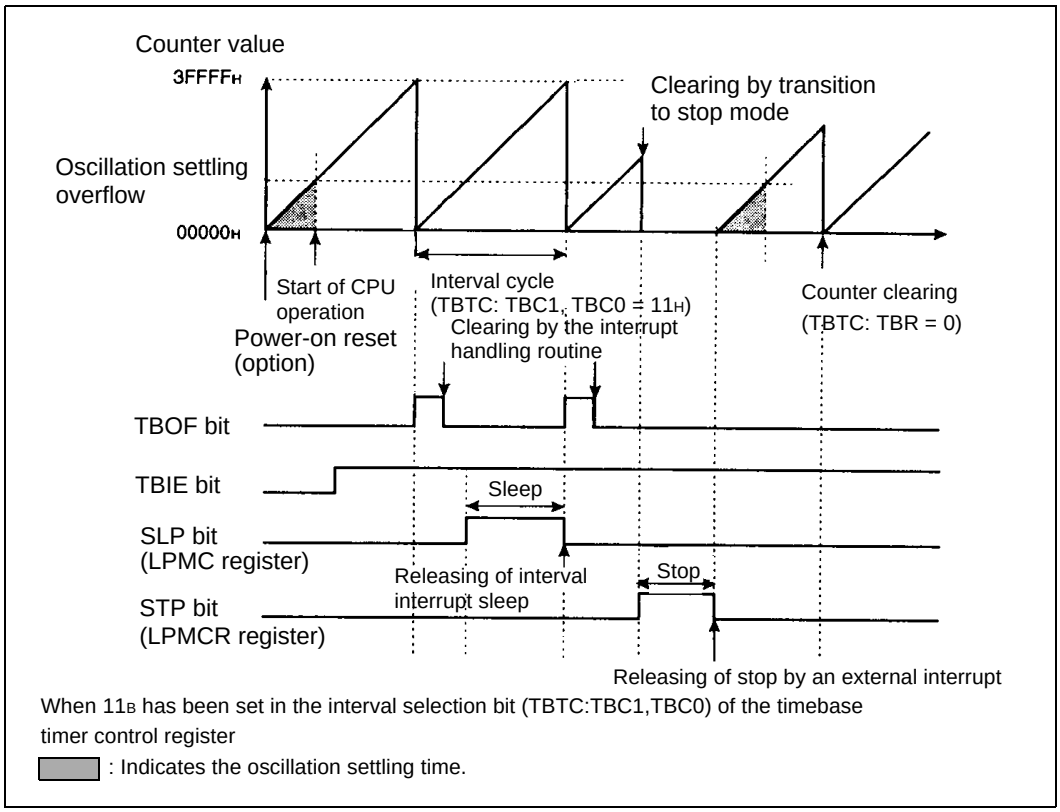


Figure 9.6-2 Timebase timer operations

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.
                                 ;  Clears counter.
                                 ;  Selects interval 2^12/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.
;        :
;      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
```

Memo

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 timebase timer supply clock as the count clock. After activation, if the watchdog timer is not cleared within a given time, the CPU is reset.

■ Watchdog timer function

The watchdog timer is a counter for handling program crashes. Once the watchdog timer is activated, it must be regularly cleared within a given time. If the program results in an endless loop and the watchdog timer is not cleared over a given time, a watchdog reset is generated for the CPU.

Table 10.1-1 lists the watchdog timer intervals. If the watchdog timer is not cleared, a watchdog reset is generated between the minimum time and maximum time. Clear the counter within the minimum time listed in this table.

Table 10.1-1 Intervals for the watchdog timer

Interval		
Minimum (*1)	**Maximum (*1)**	**Oscillation clock cycle count**
Approx. 3.58 ms	Approx. 4.61 ms	$2^{14} \pm 2^{11}$ cycle
Approx. 14.33 ms	Approx. 18.3 ms	$2^{16} \pm 2^{13}$ cycle
Approx. 57.23 ms	Approx. 73.73 ms	$2^{18} \pm 2^{15}$ cycle
Approx. 458.75 ms	Approx. 589.82 ms	$2^{21} \pm 2^{18}$ cycle

*1 Value during operation of the 4 MHz oscillation clock

The maximum and minimum watchdog timer intervals and the oscillation clock cycle count depend on the clear timing.

The interval is 3.5 to 4.5 times longer than the cycle of the count clock (timebase timer supply clock).

See Section 11.4, "Operation of the Watchdog Timer."

<Check>

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>

At activation, the watchdog timer is initialized by a power-on or watchdog reset, and is placed in stopped status. The watchdog timer is cleared by an external pin reset, software reset, writing to the WTE bit (watchdog timer control register), sleep mode, transition to stop mode, or the hold acknowledge signal. It is not stopped, however.

10.2 Configuration of the Watchdog Timer

The watchdog timer consists of the following five blocks:

- **Count clock selector**
- **Watchdog counter (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 shows the block diagram of the watchdog timer.

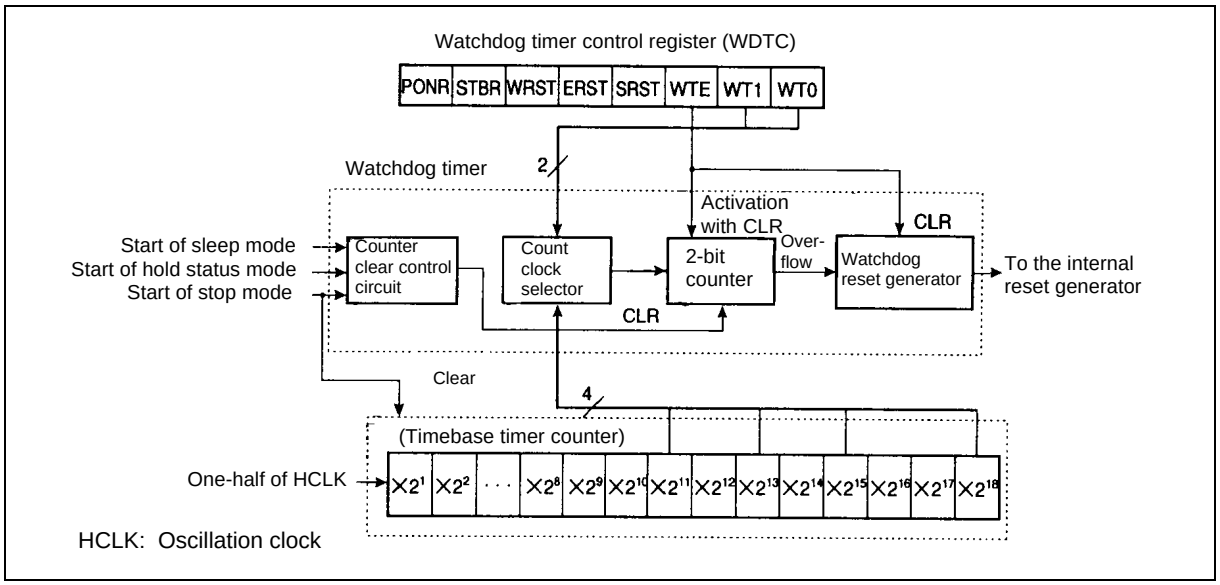


Figure 10.2-1 Block diagram of the watchdog timer

● **Count clock selector**

This circuit is used to select the count clock of the watchdog timer from four types of timebase timer outputs. This determines the watchdog reset generation time.

● **Watchdog counter (2-bit counter)**

This 2-bit up counter uses the timebase timer output as the count clock.

● **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 activate or clear the watchdog timer; holds the reset generation cause.

10.3 Watchdog Timer Control Register (WDTC)

The watchdog timer control register (WDTC) activates and clears the watchdog timer, and displays the reset cause.

■ Watchdog timer control register (WDTC)

Figure 10.3-1 shows the watchdog timer control register (WDTC); Table 11.3-1 describes the function of each bit of the watchdog timer control register (WDTC).

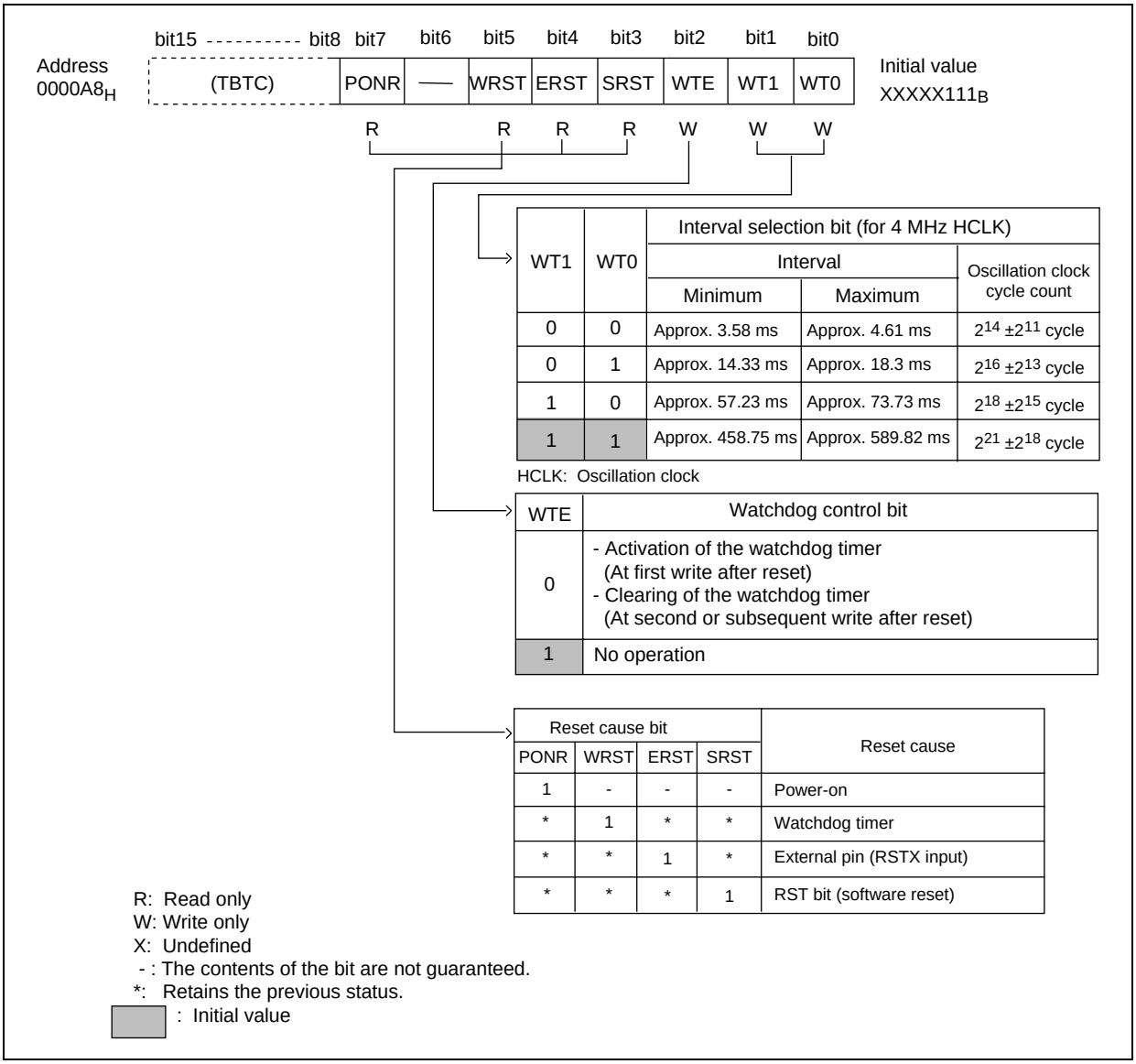


Figure 10.3-1 Watchdog timer control register (WDTC)

The interval becomes 3.5 to 4.5 times longer than the count clock (timebase timer output value) cycle. For details, see Section 11.4, "Operation of the Watchdog Timer."

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. • At power-on, the contents of the bits other than the PONR bit are not guaranteed. Therefore, when the PONR bit is “1”, ignore the contents of the bits other than the PONR bit.
bit2	WTE: Watchdog timer control bit	• When “0” is written to this bit, the watchdog timer is activated (first write after reset) or the 2-bit counter is cleared (second or subsequent write after reset). • Writing “1” does not affect operation.
bit1 bit0	WT1, WT0: Interval selection bit	• Used to select the watchdog timer interval. • Only data at watchdog timer activation is valid. Data written after watchdog timer activation is ignored. • 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 11.4-1.

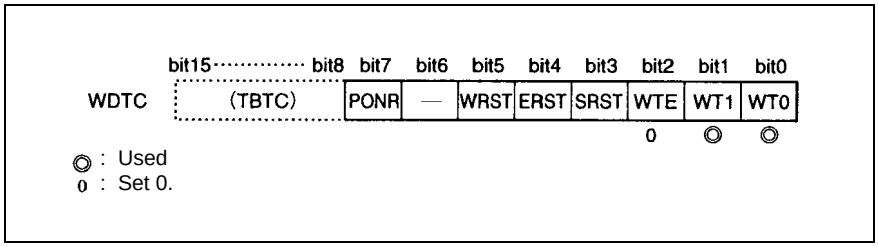


Figure 10.4-1 Setting of the watchdog timer

● Activating the watchdog timer

- The watchdog timer is activated when the first “0” after reset is written to the WTE bit of the watchdog timer control register (WDTC). Specify the interval by specifying the WT1 and WT0 bits of the watchdog timer control register at the same time.
- When watchdog timer activation starts, it can be stopped only by a power-on, or its own reset.

● Clearing the watchdog timer

- When a second or subsequent “0” is written to the WTE bit, the 2-bit counter of the watchdog timer is cleared. If the counter is not cleared within the time interval, it overflows and a watchdog reset occurs.
- The watchdog counter is cleared by reset generation, sleep mode, stop mode, transition to clock mode, or detection of the hold acknowledge signal.
- In clock mode, the watchdog timer counter is cleared and stops.

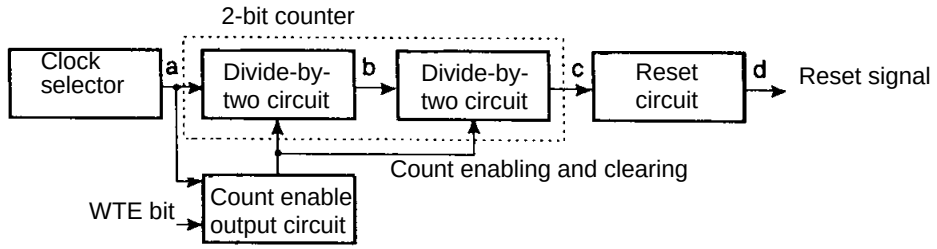
● Intervals for the watchdog timer

Figure 11.4-2 shows the relationship between the clear timing of the watchdog timer and intervals. The interval changes according to the clear timing of the watchdog timer and requires 3.5 to 4.5 times longer than the count clock cycle.

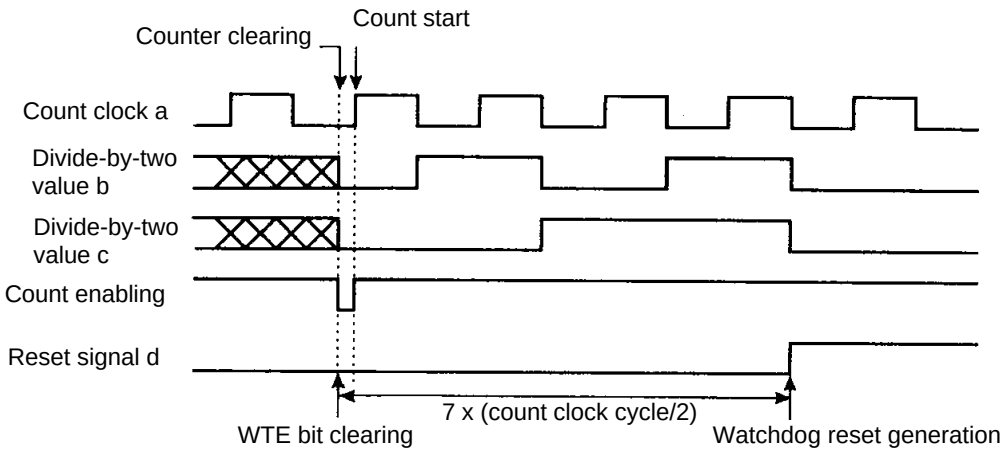
● Checking a reset cause

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.

[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:

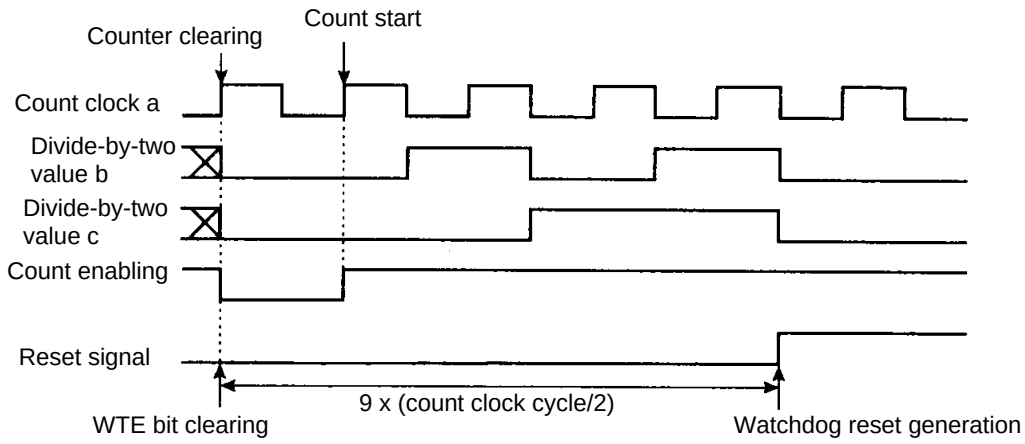


Figure 10.4-2 Clear timing and watchdog timer intervals

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. The watchdog timer counter is cleared by software reset; however, the watchdog timer does not stop.

● Intervals

Since a carry signal of the timebase timer is used as the count clock for the interval, the watchdog timer interval may become longer than the setting time when the timebase timer is cleared.

● Selecting the interval

The interval can be set when the watchdog timer is activated. Data written during operations other than activation is ignored.

● Notes on program creation

When a program that repeatedly clears the watchdog timer in the main loop is created, the processing time of the main loop including the interrupt processing must be equal to or less than the minimum watchdog timer interval.

● Watchdog timer operation in timebase timer mode

The timebase timer operates while the timebase timer mode is set. The watchdog timer, however, is temporarily stopped.

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     0000A8H             ;  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 2^21 ±2^18 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 16-bit reload timer is synchronized with three types of internal clocks for counting down in internal clock mode, and it counts down by detecting an optional edge input to the external pin in event counter mode.
This timer defines that an underflow condition results when the counter value changes from "0000$_H$" to "FFFF$_H$". Thus, an underflow will occur after an interval of [reload register setting value + 1] counts.
Either of two modes can be selected for counting. In reload mode, the value set for the count is reloaded to repeat counting for an underflow. In single-shot mode, the counting is stopped with an underflow.
The counter underflow allows the occurrence of an interrupt and coordination with the extended intelligent I/O service (EI^{2}OS).
The MB90560 series 16-bit reload timer has two built-in channels.

■ 16-bit reload timer operating modes

Table 11.1-1 lists the operating modes of the 16-bit reload timer.

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

Internal clock mode allows selection of one of three types of internal clock for the following operations:

- **Software trigger operation**

  When "1" is written to the TRG bit of the timer control status register (TMCSR), counting starts. Trigger input by the TRG bit is always valid for external trigger input as well as for external gate input.

- **External trigger operation**

  When selected edges (rising, falling, and both edges) are input to the TIN0 and TIN1 pins, counting starts.

- **External gate input operation**

  While the selected signal level ("L" or "H") is being input to TIN0 and TIN1 pins, counting continues.

■ Event count mode (external clock mode)

When selected valid edges (rising, falling, and both edges) are input to TIN0 and TIN1 pins, the timer counts down at these edges.

When an external clock with a constant period is used, this timer can also be used as an interval timer.

■ Counter operation

● Reload mode

When an underflow (from "0000H" to "FFFFH") occurs during counting down, the count setting value is reloaded to continue counting. Since the 16-bit reload timer causes an interrupt request to occur for an underflow condition, it can be used as an interval timer.

A toggle waveform inverted at each underflow can also be output from TO0 and TO1 pins.

Table 11.1-2 lists the intervals for the 16-bit reload timer.

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: Machine clock

Values in parentheses are for a 16 MHz machine clock.

● Single-shot mode

When an underflow (from "0000H" to "FFFFH") occurs during counting down, counting stops, causing an interrupt to occur for the underflow condition.

During counter operation, a rectangular waveform that indicates when the count is in progress can be output from the TO0 and TO1 pins.

Reference

- 16-bit reload timer 0 can be used to create the baud rate of UART0/UART1.
- 16-bit reload timer 1 can be used as the start trigger of the A/D converter.

■ 16-bit reload timer interrupts and EI²OS

Table 11.1-3 lists 16-bit reload timer interrupts and EI²OS.

Table 11.1-3 16-bit reload timer interrupts and EI²OS

Channel	Interrupt number	Interrupt control register		Vector table address			EI²OS
		Register name	Address	Lower	Upper	Bank	
16-bit reload timer 0 (*1)	#30 (1E$_H$)	ICR09	0000B9$_H$	FFFF84$_H$	FFFF85$_H$	FFFF86$_H$	Δ
16-bit reload timer 1 (*2)	#32 (20$_H$)	ICR10	0000BA$_H$	FFFF7C$_H$	FFFF7D$_H$	FFF7E$_H$	

Δ: Available when ICR09/ICR10 or the interrupt causes sharing the interrupt vector are not used.

*1 The same interrupt number is used for 8-bit timer counter underflow and 16-bit reload timer 0.

*2 The same interrupt number is used for 16-bit free-running timer underflow and 16-bit reload timer 1.

Memo

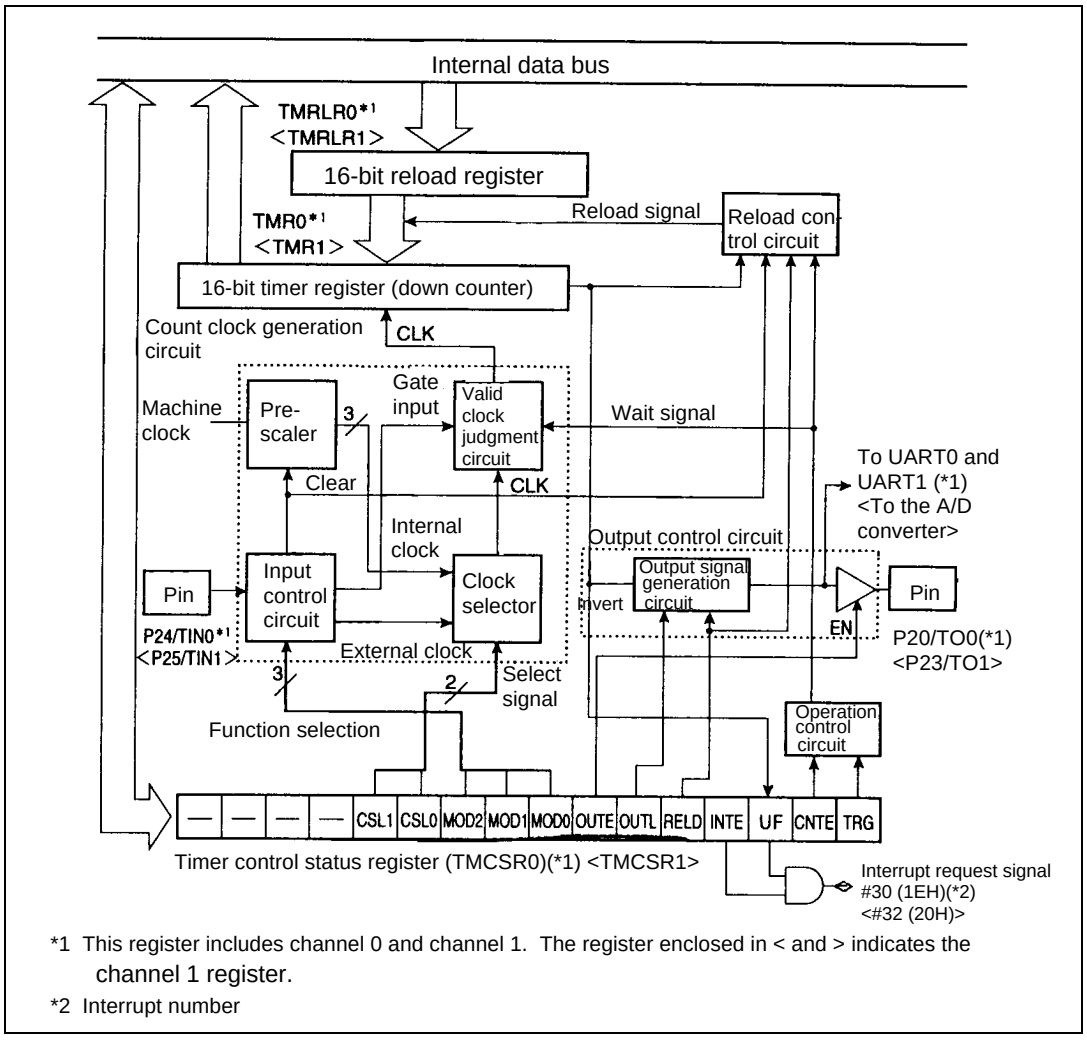
11.2 Configuration of the 16-Bit Reload Timer

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 (TMR)**
- **16-bit reload register (TMRLR)**
- **Timer control status register (TMCSR)**

■ **Block diagram of the 16-bit reload timer**

Figure 11.2-1 shows the block diagram of the 16-bit reload timer.



*1 This register includes channel 0 and channel 1. The register enclosed in < and > indicates the channel 1 register.

*2 Interrupt number

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 when the timer is started and when an underflow occurs.

- **Output control circuit**

  This circuit controls the inversion of the TO pin output by an underflow of the 16-bit timer register and enabling and disabling of TO pin output.

- **Operation control circuit**

  This circuit controls starting and stopping of the 16-bit reload timer.

- **16-bit timer register (TMR)**

  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 (TMRLR)**

  The interval for the 16-bit reload timer is set in this register. The setting value of this register is loaded into the 16-bit timer register and decremented.

- **Timer control status register (TMCSR)**

  This register selects the count clock of the 16-bit reload timer and the operating mode, sets operating conditions, starts a trigger with software, enables or disables counting, selects reload or single-shot mode and the pin output level, enables or disables timer output, controls interrupts, and controls the status.

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 lists the functions of the pins, I/O format, and settings required to use the 16-bit reload timer.

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	Port 2 I/O pin/ timer input	CMOS output/ CMOS hysteresis input	Not Available	Available	Setting for the input port (DDR2: bit 0 = 0)
P21/TO0	Port 2 I/O pin/ timer output				Setting for timer output enable (TMCSR0: OUTE = 1)
P22/TIN1	Port 2 I/O pin/ timer input				Setting for the input port (DDR2: bit 2 = 0)
P23/TO1	Port 2 I/O pin/ timer output				Setting for timer output enable (TMCSR1: OUTE = 1)

■ Block diagram of the 16-bit reload timer pins

Figure 11.3-1 shows the block diagram of the16-bit reload timer pins.

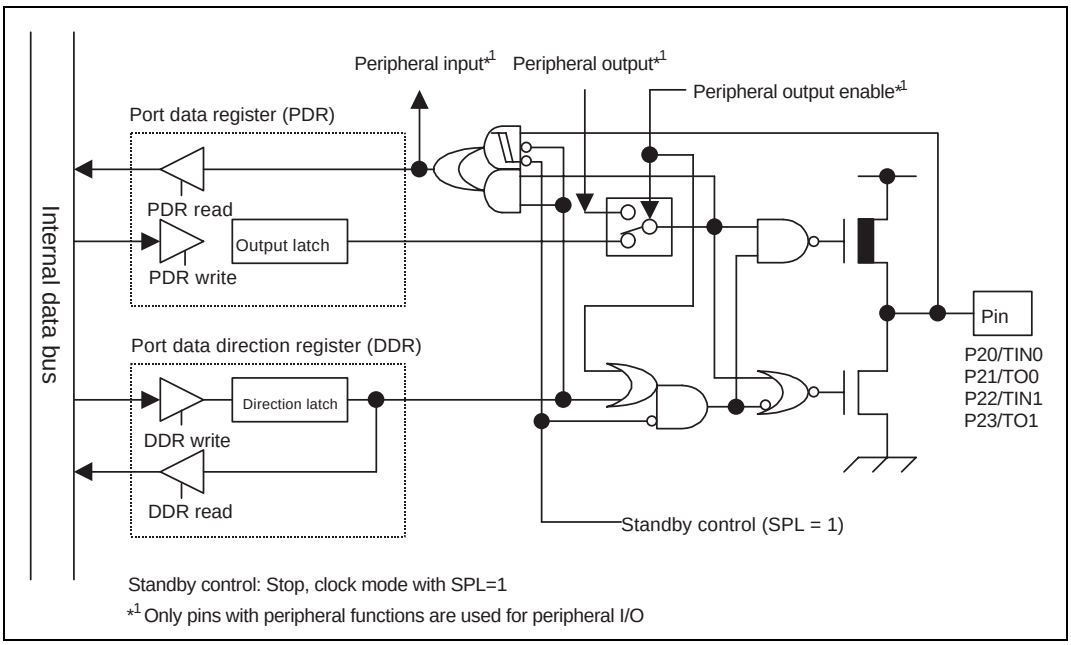


Figure 11.3-1 Block diagram of the 16-bit reload timer pins

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 shows 16-bit reload timer registers.

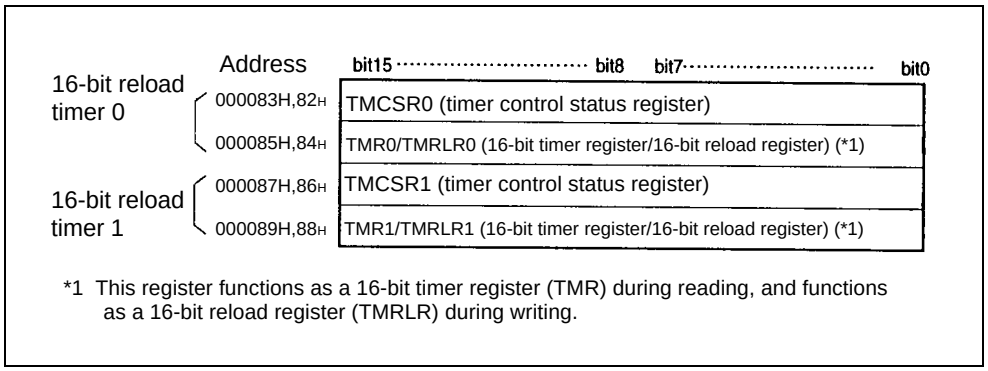


Figure 11.4-1 16-bit reload timer registers

11.4.1 Timer control status register, upper part (TMCSR0, TMCSR1: H)

Bits 11 to bit 7 of the timer control status registers (TMCSR0 and TMCSR1) are used to select the operating mode of the 16-bit reload timer and set the operating conditions. This section describes bit 7: the MOD0 bit.

■ **Timer control status register, upper part (TMCSR0, TMCSR1: H)**

Address
TMCSR0 000083H
TMCSR1 000087H

bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7	bit6 ---------- bit0	Initial value
—	—	—	—	CSL1	CSL0	MOD2	MOD1	MOD0	TMSCR : L	XXXX00000B
				R/W	R/W	R/W	R/W	R/W		

MOD2	MOD1	MOD0	Operating mode selection bit (Internal clock mode)	
			Input pin function	Valid edge and level
0	0	0	Trigger disabled	—
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 (Event clock mode)	
			Input pin function	Valid edge
X	0	0	—	—
X	0	1	Event 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 clock mode	External event input

R/W : Read/Write
— : Not used
X : Undefined
: Initial value
ϕ : Machine cycle. Value in parentheres are for 16MHz machine clock

Figure 11.4-2 Timer control status register, upper part (TMCSR0, TMCSR1: H)

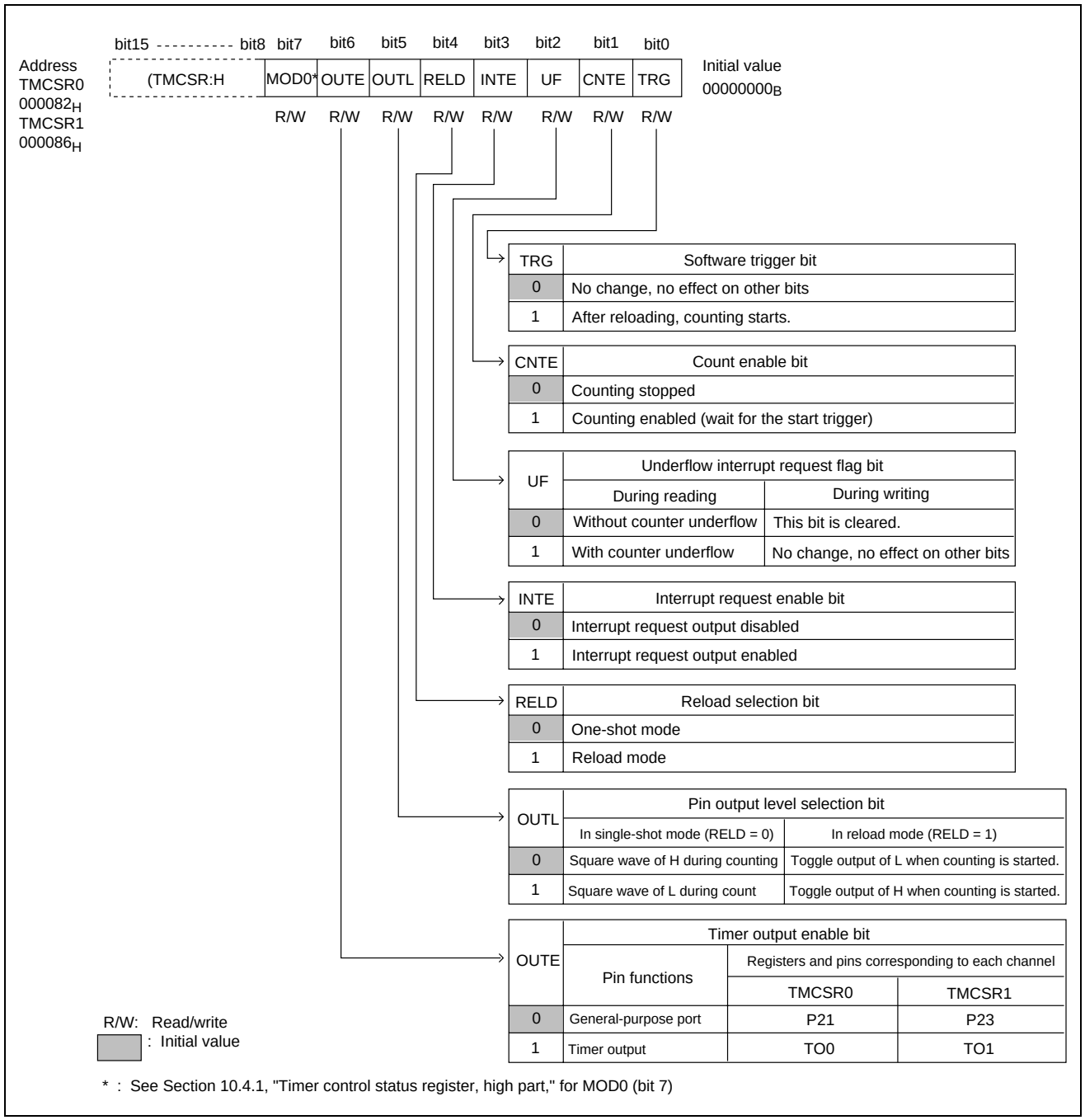
Table 11.4-1 Function description of each bit of the upper part of the timer control status register (TMCSR0, TMCSR1: H)

Bit name		Function
bit15 bit14 bit13 bit12	Not used	• 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	• Selects the count clock. • When these bits are not set to "11_B", internal clock mode is set. The internal clock is counted in this mode. • When these bits are set to "11_B", event count mode is set. The external clock edges are counted in this mode.
bit9 bit8 bit7	MOD2, MOD1, MOD0: Operating mode selection bits	<In internal clock mode> • The MOD2 bit is used to select input pin functions. • When the MOD2 bit is "0", the input pin is used as a trigger input pin, so that whenever a valid edge is input, the contents of the reload register are loaded into the counter and counting continues. The MOD1 and MOD0 bits are used to select the type of valid edge. • When the MOD2 bit is "1", the input pin becomes a gate, meaning that counting will continue only as long as a valid level signal is input. The MOD1 bit is an unused bit, and the MOD0 bit is used to select the valid level. <In event count mode> • The MOD2 bit is not used. Set any value ("0" or "1"). • The input pin is used as a trigger input pin for event input, and the valid edge is selected with MOD1 and MOD0 bits.

11.4.2 Timer control status register, lower part (TMCSR0, TMCSR1: L)

The lower seven bits of the timer control status registers (TMCSR0 and TMCSR1) are used to set operating conditions for the 16-bit reload timer, enable and disable operation, control interrupts, and check the status.

■ Timer control status register, low part (TMCSR0, TMCSR1: L)



TRG	Software trigger bit
0	No change, no effect on other bits
1	After reloading, counting starts.

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	Without counter underflow	This bit is cleared.
1	With counter underflow	No change, no effect on other bits

INTE	Interrupt request enable bit
0	Interrupt request output disabled
1	Interrupt request output enabled

RELD	Reload selection bit
0	One-shot mode
1	Reload mode

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 count	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	General-purpose port	P21	P23
1	Timer output	TO0	TO1

Figure 11.4-3 Timer control status register, low part (TMCSR0, TMCSR1: L)

Table 11.4-2 Function description of each bit of the low part of the timer control status register (TMCSR0, TMCSR1: L)

Bit name		Function
bit6	OUTE: Timer output enable bit	• This bit enables or disables output from the timer output pin. • When this bit is “0”, the pin functions as a general-purpose port. When this bit is “1”, the pin functions as a timer output pin. • In reload mode, the output waveform of this timer output pin toggles. In single-shot mode, the timer outputs a rectangular waveform that indicates that counting is in progress is output.
bit5	OUTL: Pin output level selection bit	• This register is used to select the output level of the timer output pin. • The output level of the pin is inverted depending on whether this bit is “0” or “1”.
bit4	RELD: Reload selection bit	• This bit enables reloading. • When this bit is “1”, the timer is in reload mode. At the same time an underflow occurs, the contents of the reload register are loaded into the counter, and counting continues. • When this bit is “0”, the timer is in single-shot mode. Counting stops when an underflow occurs.
bit3	INTE: Interrupt request enable bit	• This bit enables or disables output of an interrupt request to the CPU. • 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 bit is set to “1” when a counter underflow occurs. • Writing “0” to this bit clears it. Writing “1” to this bit does not change the bit value and has no effect on other bits. • This bit is also cleared when EI^2OS is activated.
bit1	CNTE: Count enable bit	• The bit enables or disables counting. • When this bit is set to “1”, the counter is placed in trigger standby mode. When a trigger occurs, actual counting starts.
bit0	TRG: Software trigger bit	• This bit starts the interval timer function or counter function with software. • Writing “1” to this bit applies a software trigger, causing the contents of the reload register to be loaded into the counter and starting counter operation. Writing “0” to this bit has no effect. • Trigger input from this trigger is always valid when the CNTE bit is set to “1” regardless of the operating mode.

11.4.3 16-bit timer register (TMR0, TMR1)

The 16-bit timer register (TMR0, TMR1) is used to read the count value from the 16-bit down counter.

■ 16-bit timer register (TMR0, TMR1)

Figure 10.4-4 shows the 16-bit timer registers (TMR0, TMR1).

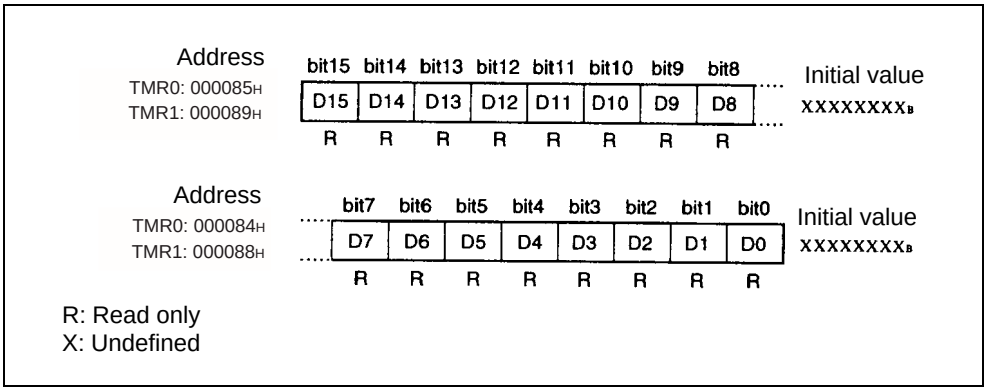


Figure 11.4-4 16-bit timer register (TMR0, TMR1)

The 16-bit timer register is used to read the counter value from the 16-bit down counter.

When counting is enabled (TMCSR: CNTE = 1) to start counting, the value written to the 16-bit reload register is loaded into this register, and counting down starts. This register value is stored when the counter is in stop status (TMCSR: CNTE = 0).

<Check>

- This register is able to read the count value even during counting. It should always be read with a word transfer instruction (MOVW A, 0084H, etc.).
- Although the 16-bit timer register (TMR) is functionally a read-only register, it is allocated to the same address as the address of the write-only 16-bit reload register (TMRLR). Accordingly, writing to this register has no effect on the TMR value, although writing to TMRLR is done.

11.4.4 16-bit reload register (TMRLR0, TMRLR1)

The 16-bit reload register (TMRLR0, TMRLR1) sets a reload value in the 16-bit down counter. The value written to this register is loaded into the down counter, and the down counter starts with this value.

■ 16-bit reload register (TMRLR0, TMRLR1)

Figure 11.4-5 shows the 16-bit reload registers (TMRLR0, TMRLR1).

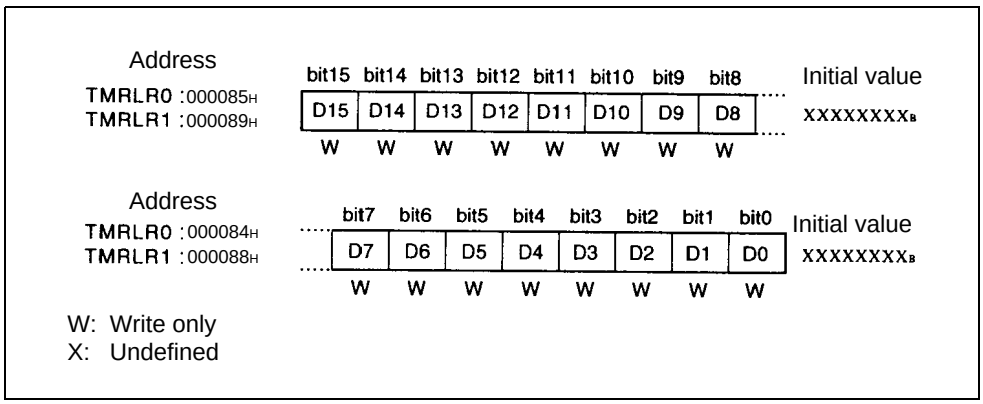


Figure 11.4-5 16-bit reload register (TMRLR0, TMRLR1)

The initial value of the counter is set in this register when counting is disabled (TMCSR: CNTE = 0), regardless of the operating mode of the 16-bit reload timer. When counting is enabled (TMCSR: CNTE = 1) and the counter is started, counting down starts from the value written to this register.

In reload mode, the value set in the 16-bit reload register (TMRLR) is reloaded into the counter when an underflow occurs, and counting down continues. In single-shot mode, the counter stops at FFFFH when an underflow occurs.

<Check>

- Write a value to this register in the counter stop (TMCSR: CNTE = 0) state. Also, always use a word transfer instruction (MOVW 0084H,A etc.) to write a value to this register.
- Although the 16-bit reload register (TMRLR) is functionally a write-only register, it is allocated to the same address as one of the read-only 16-bit timer registers (TMR). Accordingly, since the read value is used as the TMR value, the INC/DEC instruction and other instructions for read-modify-write (RMW) operations cannot be used.

11.5 16-Bit Reload Timer Interrupts

The 16-bit reload timer is able to generate an interrupt request when an underflow of the counter is occurred. It is also coordinated with the extended intelligent I/O service (EI²OS).

■ 16-bit reload timer interrupts

Table 11.5-1 lists the interrupt control bits and interrupt causes of the 16-bit reload timer.

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
Interrupt request flag bit	TMCSR0: UF	TMCSR1: UF
Interrupt request enable bit	TMCSR0: INTE	TMCSR: 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 UF bit of the timer control status register (TMCSR) is set to 1 by an underflow (from 0000H to FFFFH) of the down counter. If an interrupt request is enabled (TMCSR: INTE = 1) in this operation, the interrupt request is output to the interrupt controller.

■ 16-bit reload timer interrupts and EI²OS

Table 11.5-2 lists the 16-bit reload timer interrupts and EI²OS.

Table 11.5-2 16-bit reload timer interrupts and EI²OS

Channel	Interrupt number	Interrupt control register		Vector table address			EI²OS
		Register name	Address	Lower	Upper	Bank	
16-bit reload timer 0 (*1)	#30 (1EH)	ICR09	0000B9H	FFFF84H	FFFF85H	FFFF86H	Δ
16-bit reload timer 1 (*2)	#32 (20H)	ICR10	0000BAH	FFFF7CH	FFFF7DH	FFFF7EH	

Δ: Available when ICR09 or the interrupt causes sharing the interrupt vector are not used.

*1 The same interrupt number as that for 8-bit timer counter underflow and 16-bit reload timer 0 underflow.

*2 The same interrupt number as that for 16-bit free-run timer overflow and 16-bit reload timer 1 underflow.

■ EI²OS function of the 16-bit reload timer

Since the 16-bit reload timer has a circuit that coordinates with EI²OS, the counter can start EI²OS when an underflow occurs.

However, EI²OS is available only when other peripheral functions sharing the interrupt control register (ICR) do not use interrupts. For example, when 16-bit reload timer 1 uses EI²OS, interrupts of the 16-bit free-run timer must be disabled.

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 is required to operate this timer as an interval timer.

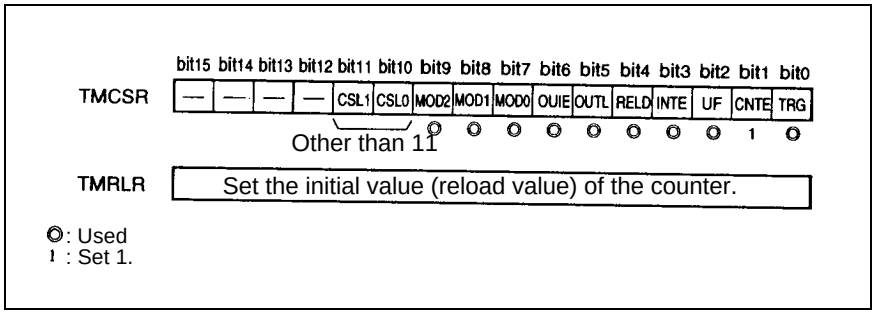


Figure 11.6-1 Internal clock mode setting

● Event counter mode setting

The setting shown in Figure 11.6-2 is required to operate this timer as an event counter.

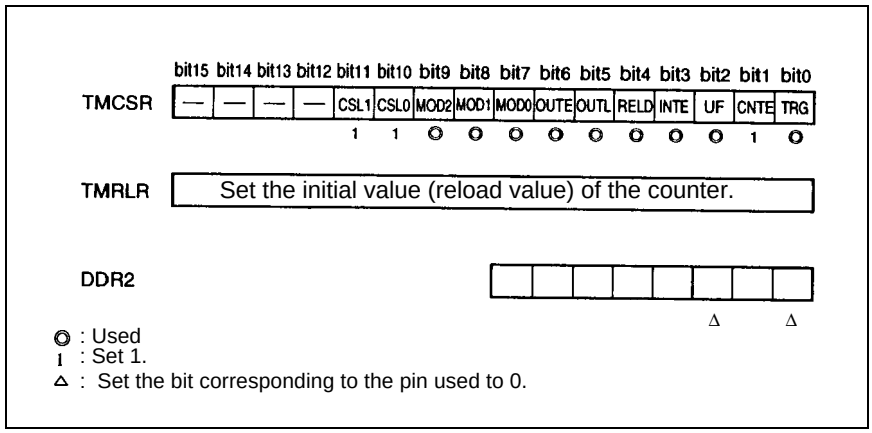


Figure 11.6-2 Event counter mode setting

■ Counter operating status

The counter status is determined by the CNTE bit of the timer control status register (TMCSR) and the internal WAIT signal. Possible settings include the stop status (STOP status), trigger wait status (WAIT status), and running status (RUN status).

Figure 11.6-3 shows the transitions of these counter statuses.

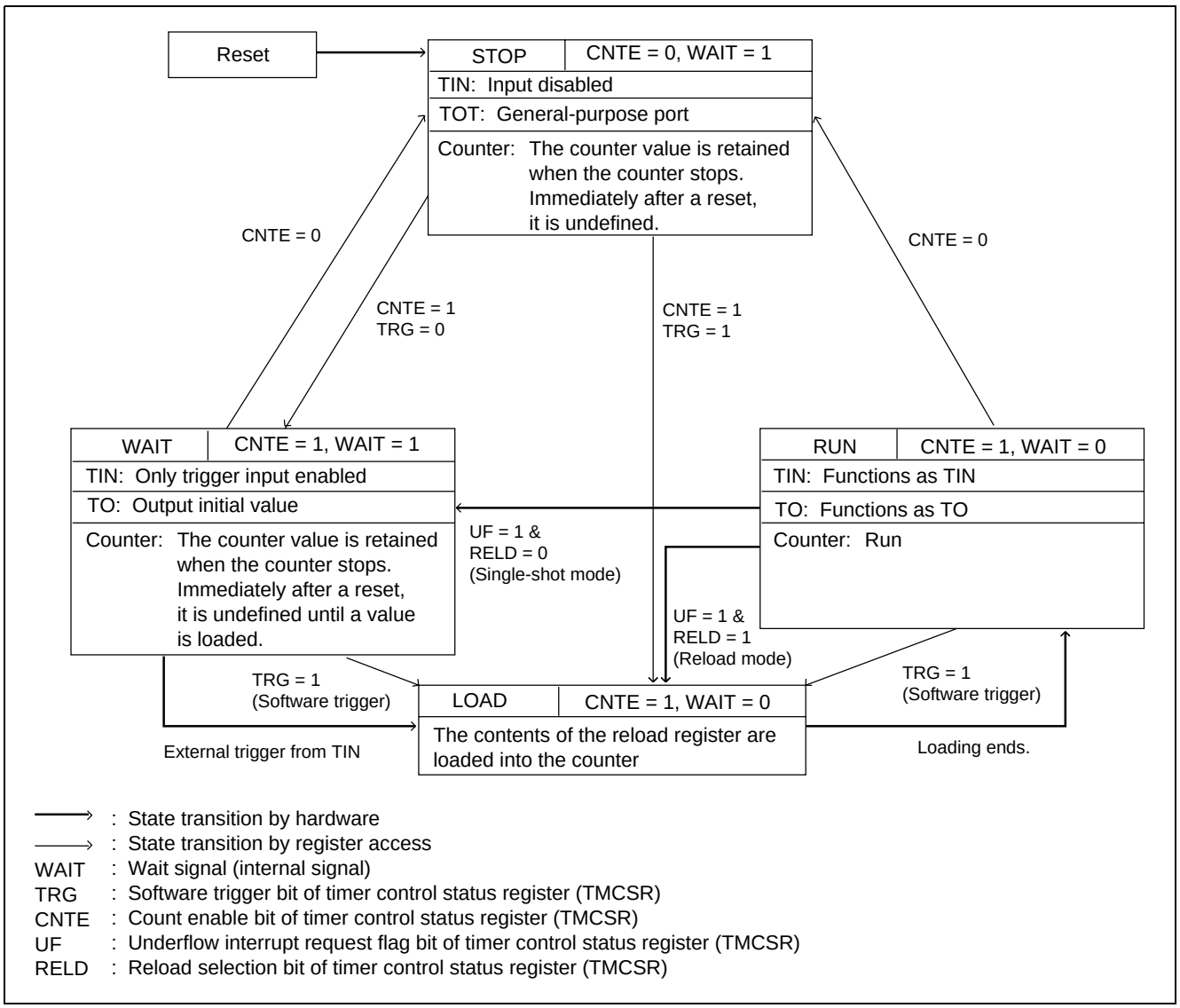


Figure 11.6-3 Counter status transition

11.6.1 Reload Mode (Internal Clock Mode)

Synchronized with the internal count clock, the 16-bit reload timer is used for counting down of the 16-bit counter and generating an interrupt request to the CPU when the counter is underflow. It can also output a toggle waveform to the timer output pin.

■ Operation in reload mode (internal clock mode)

When counting is enabled (TMCSR: CNTE = 1) and the timer is started by the software trigger bit (TMCSR: TRG) or external trigger, the value of the reload register (TMRLR) is loaded into the counter, and counting starts. If the count enable bit and software trigger bit are set to 1 simultaneously, counting starts simultaneously with the enabling of counting.

When an underflow of the counter value (from "0000$_H$" to "FFFF$_H$") occurs, the value of the 16-bit reload register (TMRLR) is loaded into the counter, and counting continues. If the underflow interrupt request flag (UF) bit is set to "1" and the interrupt request enable (INTE) bit is "1", an interrupt request is generated.

The timer can also output to the TO pin a toggle waveform, which is inverted for each underflow.

● Software trigger operation

When "1" is written to the TRG bit of the timer control status register (TMCSR), the counter is started. Figure 11.6-4 shows software trigger operation in reload mode.

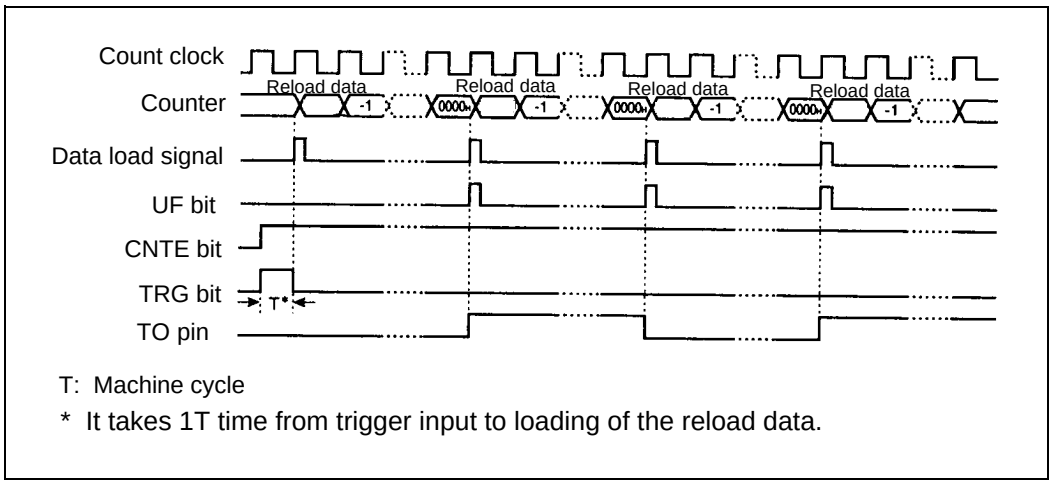


Figure 11.6-4 Count operation in reload mode (software trigger operation)

● **External trigger operation**

When a valid edge (rising, falling, and both edges can be selected) is input to the TIN pin, the counter is started. Figure 11.6-5 shows external trigger operation in reload mode.

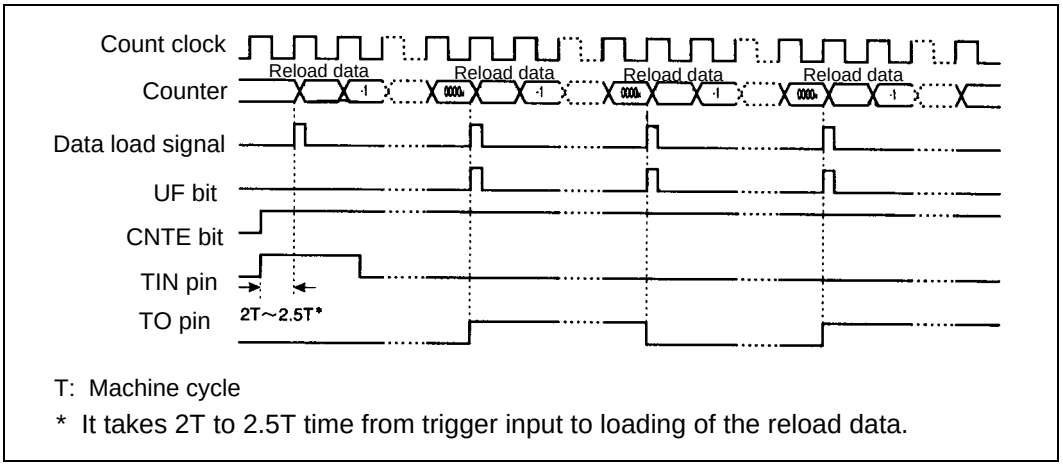


Figure 11.6-5 Counting in reload mode (external trigger operation)

<Check>

Specify 2/ϕ (machine clock) or more for the width of the trigger pulse input to the TIN pin.

● **Gate input operation**

While a valid level (H and L levels can be selected) is input to the TIN pin, counting is done. Figure 11.6-6 shows gate input in reload mode.

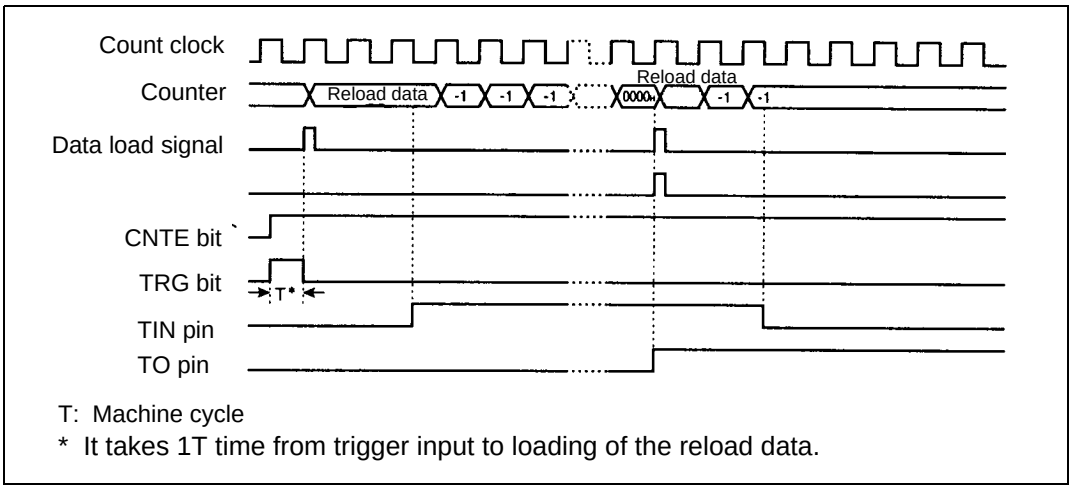


Figure 11.6-6 Count operation in reload mode (software trigger and gate input operation)

<Check>

Specify 2/ϕ (machine clock) or more for the width of the trigger pulse input to the TIN pin.

11.6.2 Single-shot Mode (Internal Clock Mode)

Synchronized with the internal count clock, the 16-bit reload timer is used for counting down of the 16-bit counter and generating an interrupt request to the CPU when the counter is underflow. It can also output to the TO pin a rectangular waveform indicating that counting is in progress.

■ Single-shot mode (Internal clock mode)

When counting is enabled (TMCSR: CNTE = 1) and the timer is started with the software trigger bit (TMCSR: TRG) or external trigger, counting starts. If the count enable bit and software trigger bit are set to "1" simultaneously, counting starts simultaneously with the enabling of counting. When an underflow of the counter value (from "0000_H" to "$FFFF_H$") occurs, the counter stops in the "$FFFF_H$" state. If the underflow interrupt request flag (UF) bit is set to "1" and the interrupt request enable (INTE) bit is "1", an interrupt request is generated.

The timer can also output to the TO pin a rectangular waveform indicating that counting is in progress.

● Software trigger operation

When "1" is written to the TRG bit of the timer control status register (TMCSR), the counter is started. Figure 11.6-7 shows the software trigger operation in single-shot mode.

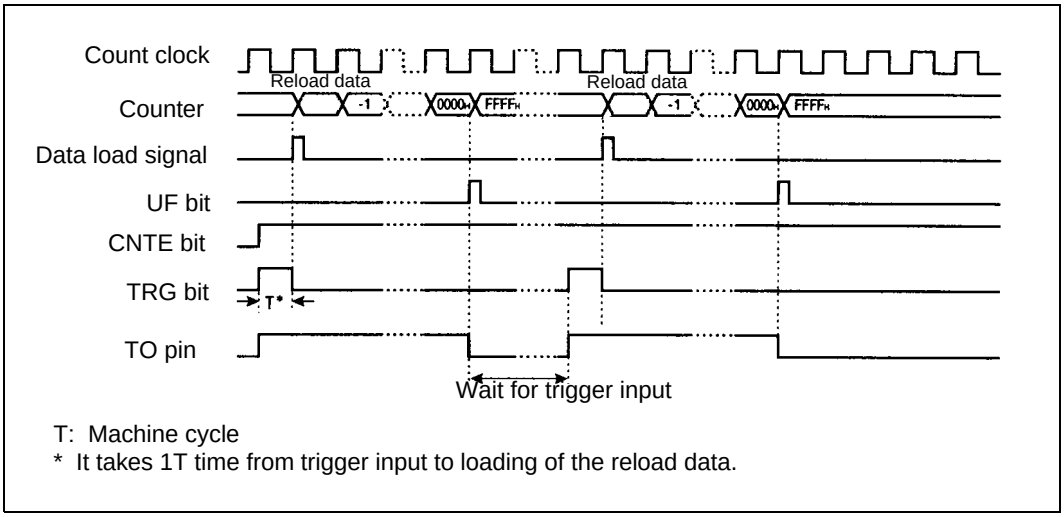


Figure 11.6-7 Count operation in single-shot mode (software trigger operation)

● **External trigger operation**

When a valid edge (rising, falling, and both edges can be selected) is input to the TIN pin, the counter is started. Figure 11.6-8 shows external trigger operation in single-shot mode.

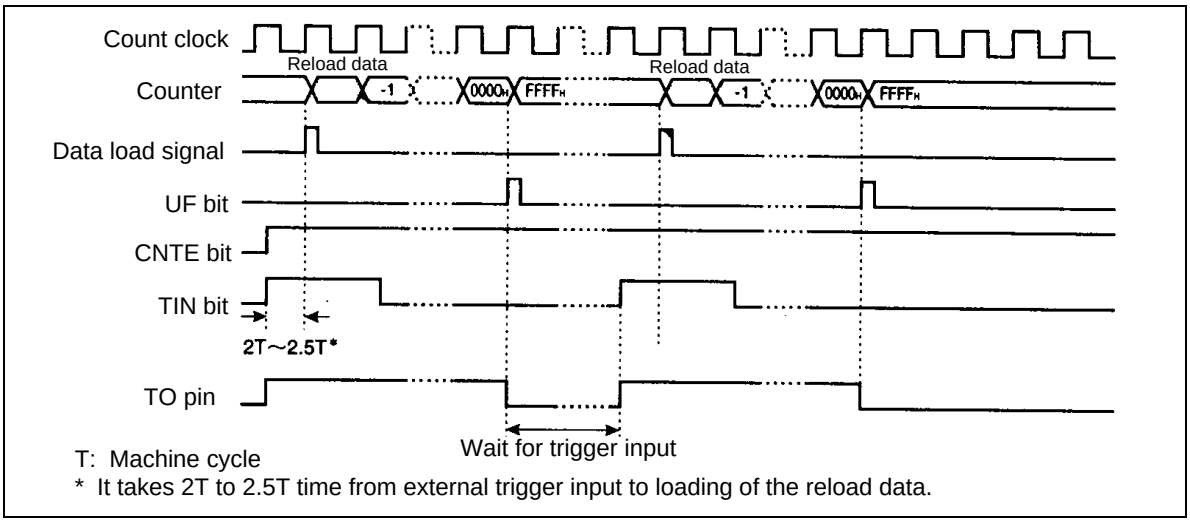


Figure 11.6-8 Count operation in single-shot mode (external trigger operation)

<Check>

Specify 2/ϕ (machine clock) or more for the width of the trigger pulse input to the TIN pin.

● **Gate input operation**

While a valid level (“H” and “L” levels can be selected) is input to the TIN pin, counting is done. Figure 11.6-9 shows gate input operation in single-shot mode.

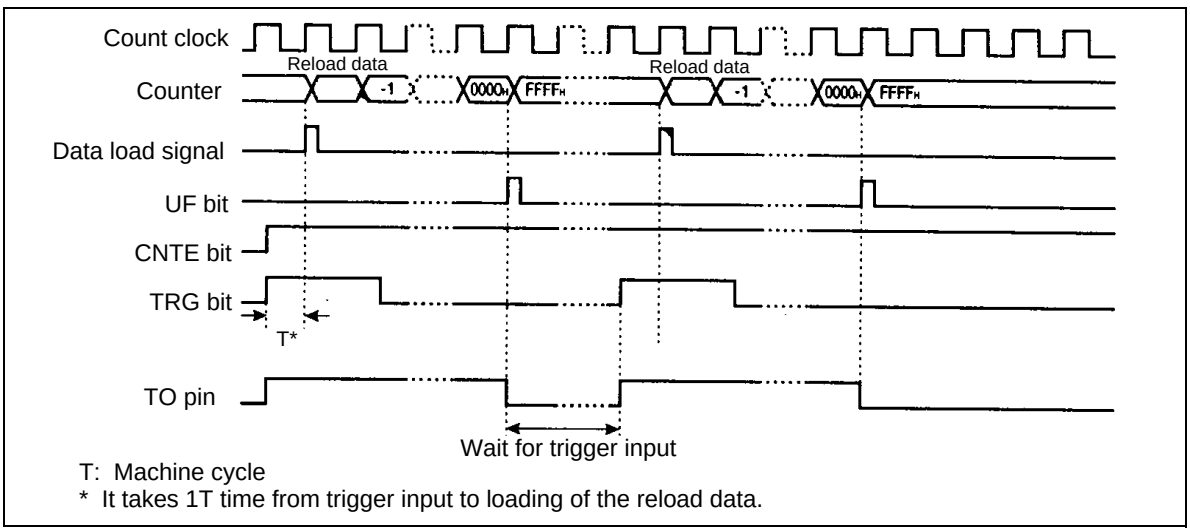


Figure 11.6-9 Count operation in single-shot mode (software trigger and gate input operation)

<Check>

Specify 2/ϕ (machine clock) or more for the width of the trigger pulse input to the TIN pin.

11.6.3 Event count mode

In the 16-bit reload timer, an input edge from the TIN pin is used to count down the 16-bit counter and an interrupt request to the CPU is generated when the counter is underflow. It can also output a toggle waveform or rectangular waveform to the TO pin.

■ Event count mode

When counting is enabled (TMCSR: CNTE = 1) and the counter is started (TMCSR: TRG = 1), the value of the reload register is loaded into the counter. Counting down is then done every time a valid edge (rising, falling, and both edges can be selected) of the pulse input to the TIN pin (external count clock) is detected.

When the count enable bit and software trigger bit are set to “1” simultaneously, counting is started simultaneously with the enabling of counting.

● Operation in reload mode

When an underflow of the counter value (from “0000H” to “FFFFH”) occurs, the value of the reload register (TMRLR) is loaded into the counter, and counting continues. If the underflow interrupt request flag (UF) bit is set to “1” and the interrupt request enable bit (TMCSR: INTE) is 1, an interrupt request is generated.

The timer can also output to the TO pin a toggle waveform, which is inverted for each underflow.

Figure 11.6-10 shows counting in reload mode.

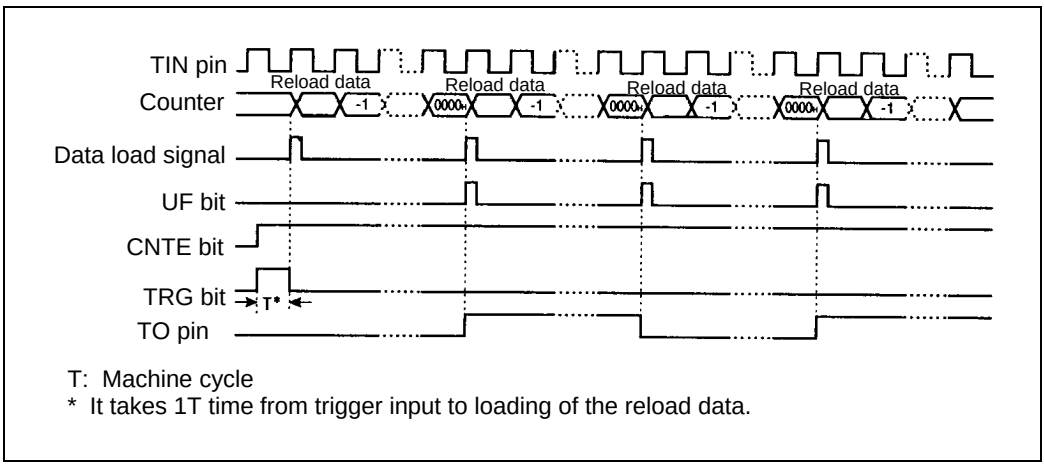


Figure 11.6-10 Count operation in reload mode (event count mode)

<Check>

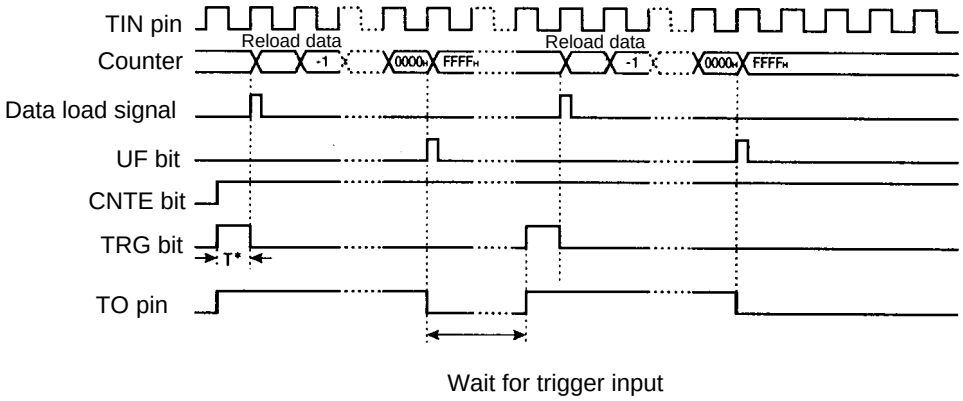
Specify 4/ϕ (machine clock) or more for the “H” and “L” widths of the clock input to the TIN pin.

● **Operation in single-shot mode**

When an underflow of the counter value (from "0000H" to "FFFFH") occurs, the counter stops in the FFFFH state. If the underflow interrupt request flag (UF) bit is set to "1" and the interrupt request enable (INTE) bit is "1", an interrupt request is generated.

The timer can also output to the TO pin a rectangular waveform indicating that counting is in progress.

Figure 11.6-11 shows counting in single-shot mode.



T: Machine cycle
* It takes 1T time from trigger input to loading of the reload data.

Figure 11.6-11 Counter operation in single-shot mode (event count mode)

<Check>

Specify 4/ϕ (machine clock) or more for the "H" and "L" widths of the clock 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 the16-bit reload timer

● Notes on using a program for setting

- Write a value to the 16-bit reload register (TMRLR) when counting stops (TMCSR: CNTE = 0). Also, a value can be read from the 16-bit timer register (TMR) even during counting, but always be sure to use a word transfer instruction (MOVW A, dir, etc.).
- Change the CSL1 and CSL0 bits of the timer control status register (TMCSR) when the counter has stopped (TMCSR: CNTE = 0).

● Notes about interrupts

- When the UF bit of the timer control status register (TMCSR) 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.
- Since the 16-bit reload timer 0 shares an interrupt vector with 8-bit timer 0/1/2, interrupt causes must be checked carefully by the interrupt processing routine when interrupts are used.

  Furthermore, when EI^{2}OS is used by the 16-bit reload timer 0, 8-bit timer 0/1/2 interrupts must be disabled
- Since the 16-bit reload timer 1 shares an interrupt vector with 16-bit free-run timer, interrupt causes must be checked carefully by the interrupt processing routine when interrupts are used.

  Furthermore, when EI^{2}OS is used by the 16-bit reload timer 0, 16-bit free-run timer interrupts must be disabled

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 timer is started with a software trigger. External trigger input is not used.
- 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     TMCSR0:2             ;   Interrupt request flag bit
CNTE      EQU     TMCSR0:1             ;   Counter operation enable bit
TRG       EQU     TMCSR0:0             ;   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:TMRLR0,#30D4H      ;   Sets data for 25-ms timer.
          MOVW    I:TMCSR0,#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 interrupt request flag.
;          :
;         User processing
;         :
          RETI                         ;   Returns from interrupt
```

```
CODEENDS
;-------Vector setting----------------------------------------------------------------------------------------
VECT    CSEG    ABS=0FFH
        ORG     0FF84H          ;   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 data register
UF      EQU     TMCSR0:2        ;   Interrupt request flag bit
CNTE    EQU     TMCSR0:1        ;   Counter operation enable bit
TRG     EQU     TMCSR0:0        ;   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:TMRLR0,#2710H ;   Sets reload value to 10,000.
        MOVW    I:TMCSR0,#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                ;   Clears interrupt request flag.
;               :
;           User processing
;               :
            RETI                        ;   Returns from interrupt.
CODE        ENDS
;-------Vector setting--------------------------------------------------------------------------------------------
VECTC       SEG     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 MULTI-FUNCTION TIMER

This chapter describes the functions and operation of the Multi-function timer in MB90560 series.

12.1 Overview of Multi-function Timer

The multi-function timer consists of a 16-bit free-run timer, six 16-bit output compare, four 16-bit input capture, 6 channels of 8-bit PPG timer (3 channels of 16-bit PPG timer) and a waveform generator. By using this waveform generator, 12 independent waveform can be outputted through 16-bit free-run timer. Furthermore input pulse width measurement and external clock cycle measurement can be done.

■ 16-bit free-run timer (X1)

The 16-bit free-run timer consists of a 16-bit up-counter, control register, 16-bit compare clear register and a prescaler. The output value of this counter will be used as the count clock of the output compares and input captures

- 6 types of counter operation clock (ϕ, ϕ/2, ϕ/4, ϕ/8, ϕ/16, ϕ/32, ϕ/64 ϕ/128) can be selected.
  ϕ: Internal clock
- An interrupt is generated when there is an overflow in the counter value or comparing match with compare clear register (Mode setting is necessary for compare match)
- Reset, software clear, compare match with compare clear register will reset the counter value to "0000_H"

■ Output compare (X6)

The output compare consists of six 16-bit compare registers, compare output latch and compare control registers. An interrupt is generated and output level is inverted when the value of 16-bit free run timer and compare register are matched.

- 6 compare registers can be operated independently.
  Output pins and interrupt flag are corresponding to each compare register.
- 2 compare register can be paired to control the output pins.
  Inverts output pins by using 2 compare register together.
- Setting the initial value for each output pin is possible.
- An interrupt is generated when compare register is matched.

■ Input capture (x4)

Input capture consists of 4 independent external input pins, the corresponding capture register and capture control register. By detecting any edge of the input signal from the external pin, the value of the 16-bit free-run timer can be stored in the capture register and an interrupt is generated simultaneously.

- 3 types of trigger edge (rising edge, falling edge, both edge) of the external input signal can be selected.
- 4 input captures can operated independently.
- An interrupt is generated by detecting a valid edge from external input.

■ 8/16-bit PPG timer (8-bit PPG x 6, 16-bit PPG x 3)

8/16-bit PPG timer consists of six 8-bit down-counter, twelve 8-bit re-load register and three 8-bit control register. By using it as 8/16-bit re-load timer, it is possible to operate as an event counter. By setting of 8/16-bit “L” width and ‘H’ width, it is possible to output a pulse with any cycle duty ratio.

- 8-bit PPG output operation mode
  6 channels of 8-bit PPG output can be operated independently.
- 16-bit PPG output operation mode
  3 channels of 16-bit PPG output can be operated independently.
- 8+8-bit PPG output operation mode
  In this mode, Ch/0/Ch2/Ch4 can be operated as an 8-bit prescaler, in which an underflow output of Ch/0/Ch2/Ch4 is used as a count clock for Ch1/Ch3/Ch5. It is possible to output any cycle from any 8-bit PPG output.
- PPG output operation
  The 8/16-bit PPG timer can output pulse waveforms with variable period and duty ratio. Also, it can be used as D/A converter in conjunction with an external circuit.

■ Waveform generator

The waveform generator consists of three 8-bit re-load registers, three timer control registers and 8-bit waveform control register.

With waveform generator, it is possible to generate realtime output, 8/16-bit PPG waveform output, non-overlap 3-phase waveform output for inverter control and DC chopper waveform output.

- It is possible to generate a non-overlap waveform output based on dead-time of 8-bit timer. (Dead-time timer function)
- It is possible to generate a non-overlap waveform output when realtime output is operated in 2-channel mode.(Dead-time timer function)
- By detecting realtime output compare match, GATE signal of the PPG timer operation will be generated to start or stop PPG timer operation. (GATE function)
- When a match is detected by real-time output compare, the 8-bit timer is activated. The PPG timer can be started or stopped easily by generating a GATE signal for PPG operation until the 8-bit timer stops. (GATE function)
- Forced stop control using DTTI pin input
  External pin control can be performed through clockless DTTI pin input even when oscillation is stopped. (The pin level can be set by each pin or software.)

12.2 Block Diagram of Multi-function Timer

Block diagram of multi-function timer is shown in the section.

■ **Realtime I/O Block Diagram**

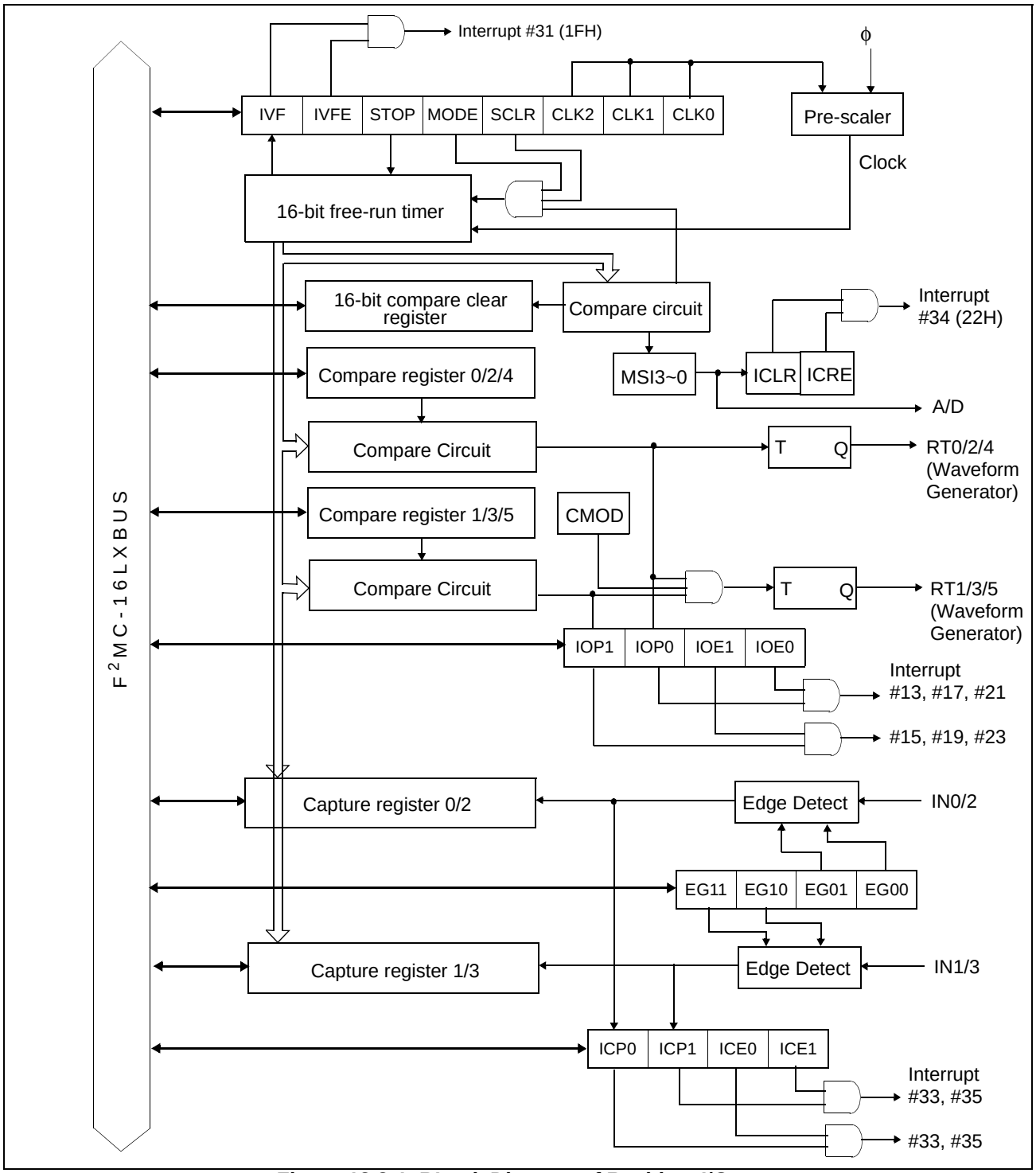
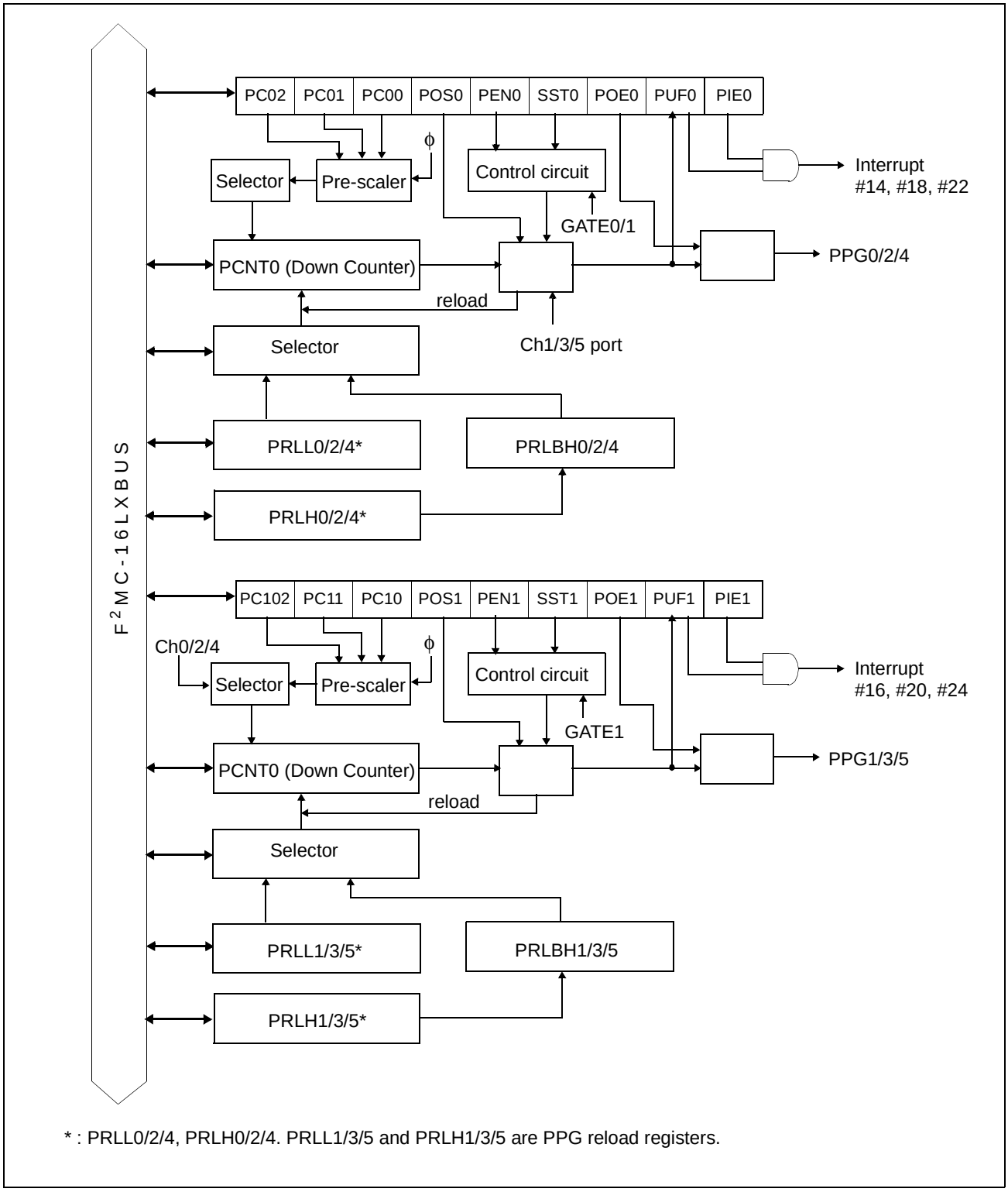


Figure 12.2-1 BLock Diagram of Realtime I/O

■ **8/16-bit PPG timer block Diagram**



* : PRLL0/2/4, PRLH0/2/4. PRLL1/3/5 and PRLH1/3/5 are PPG reload registers.

Figure 12.2-2 Block Diagram of 8/16-bit PPG Timer

■ Waveform Generator Block Diagram

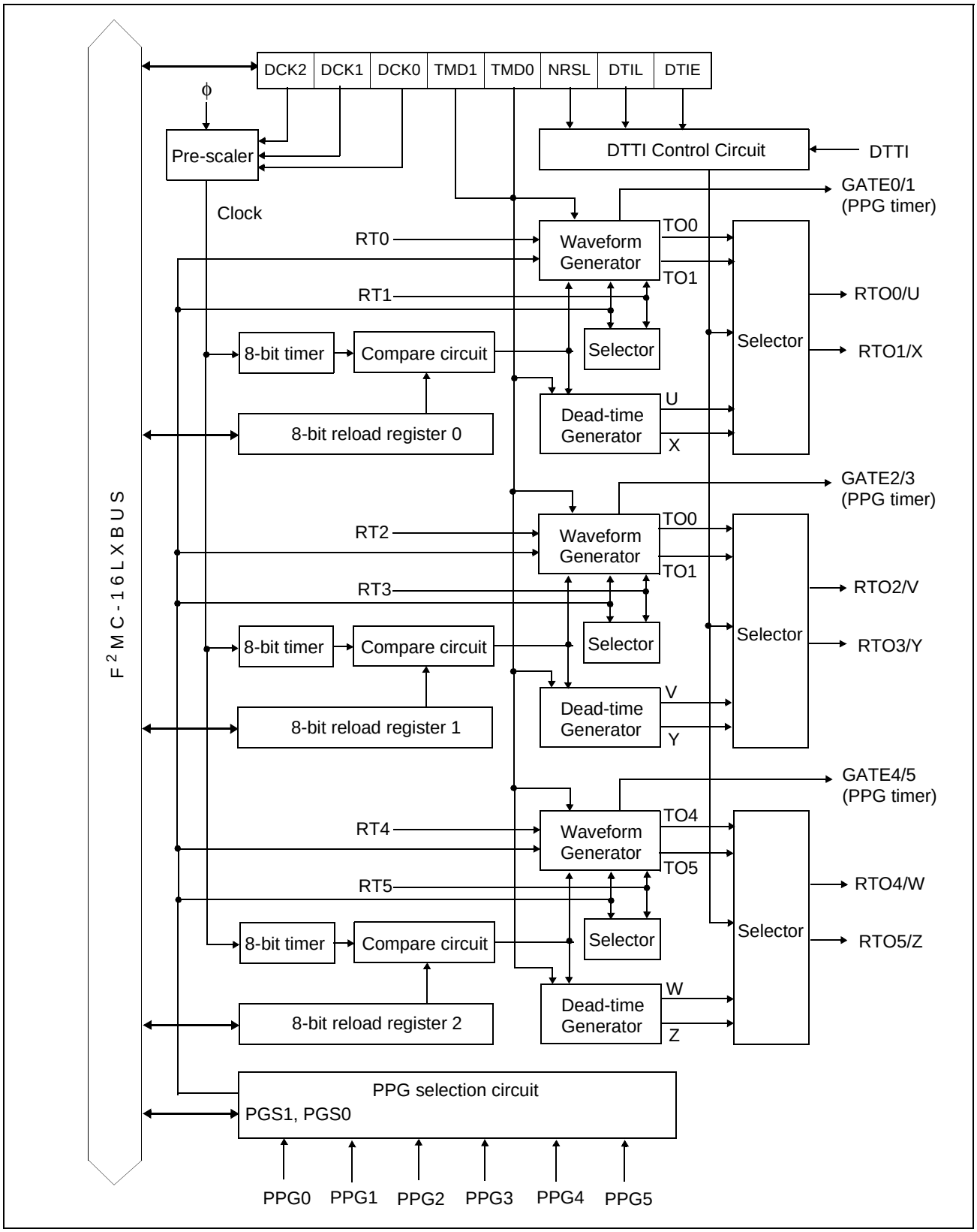


Figure 12.2-3 Block Diagram of Waveform Generator

Memo

12.3 Register of Multi-Function Timer

This section describes registers of multi-function timer.

■ 16-bit Free-Run Timer Registers

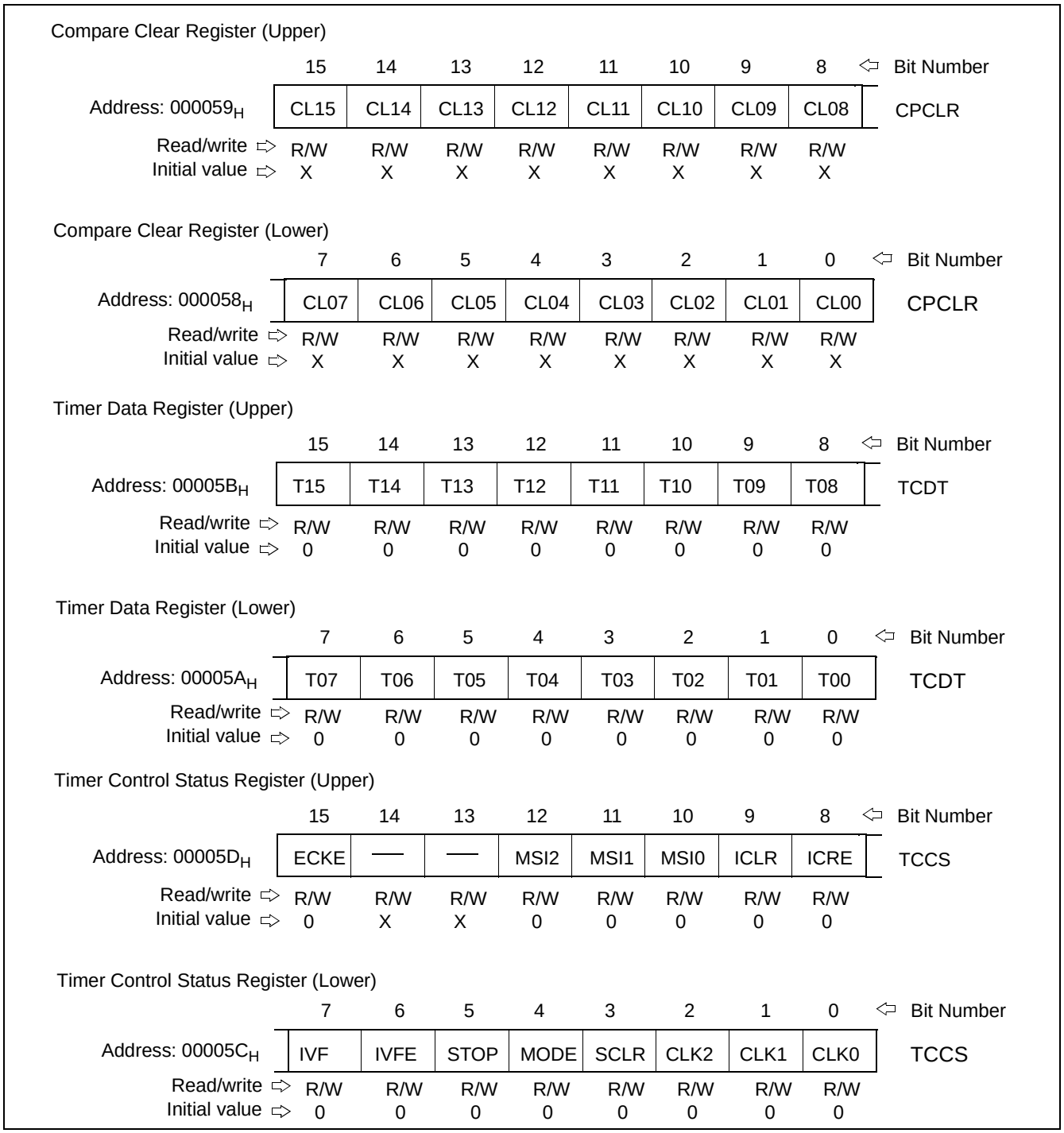


Figure 12.3-1 Registers of 16-bit Free-Run Timer

■ Output Compare Registers

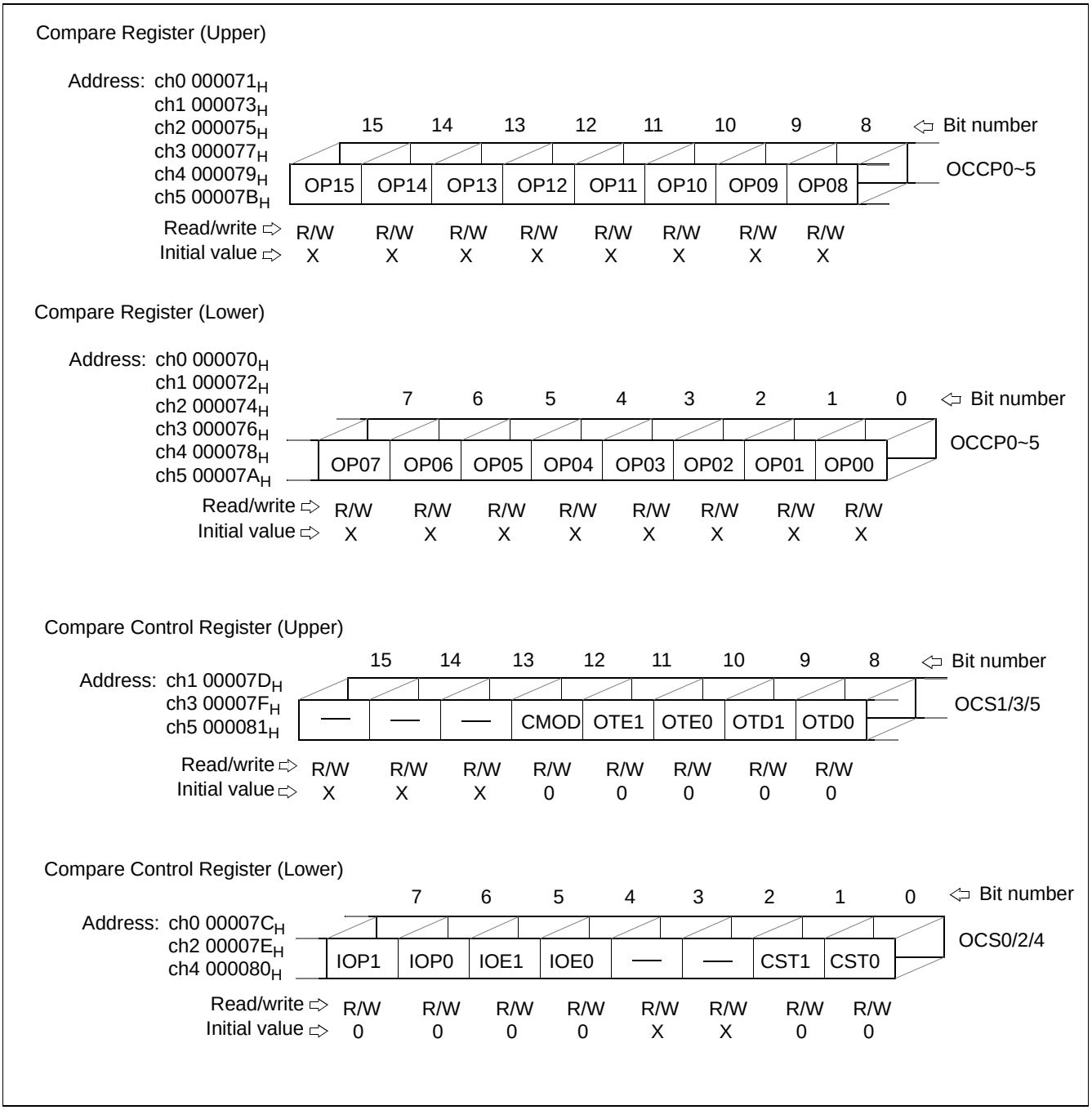


Figure 12.3-2 Registers of Output Compare

■ Input capture registers

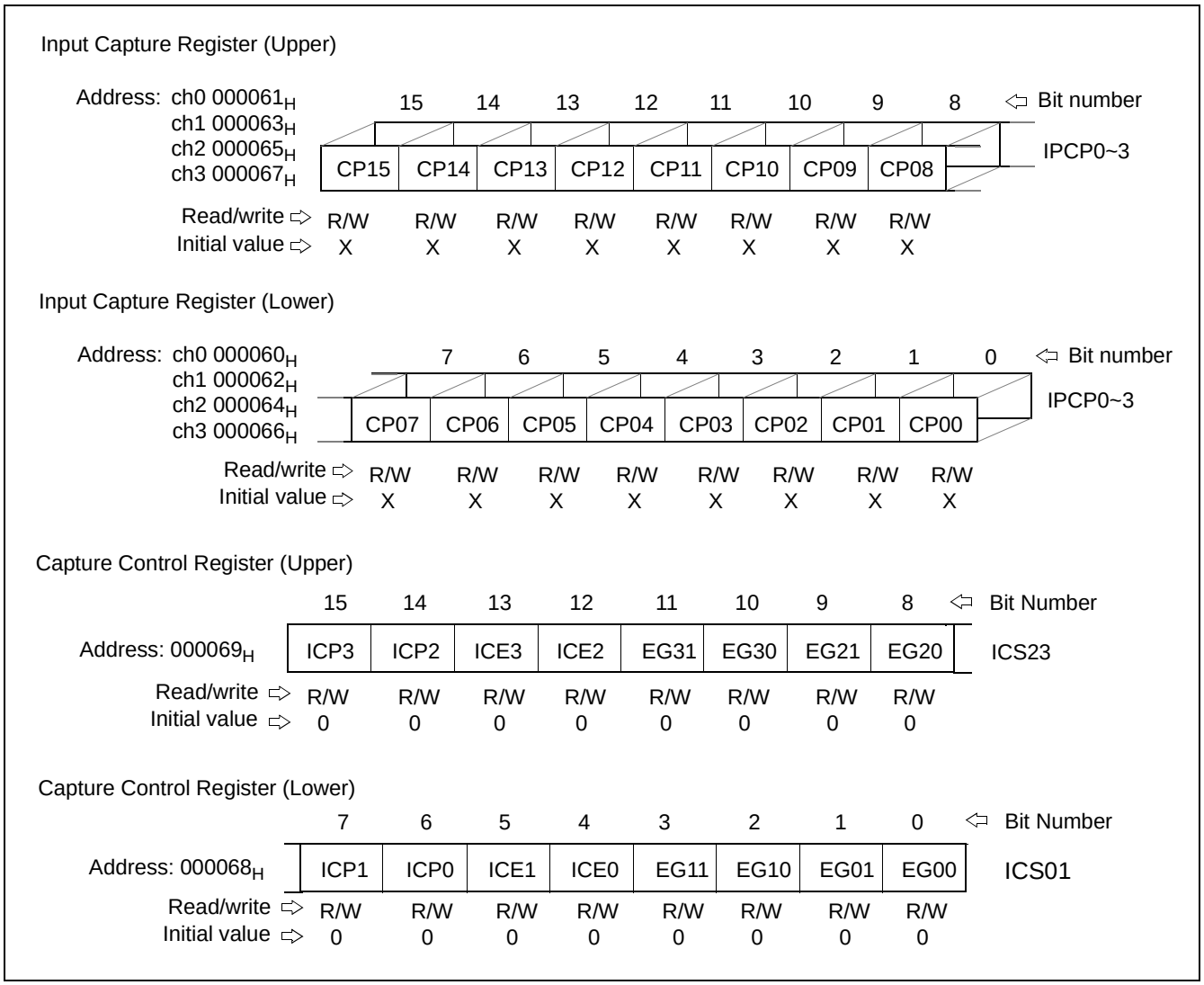


Figure 12.3-3 Registers of 16-bit Input Capture

■ 8/16-bit PPG Timer Registers

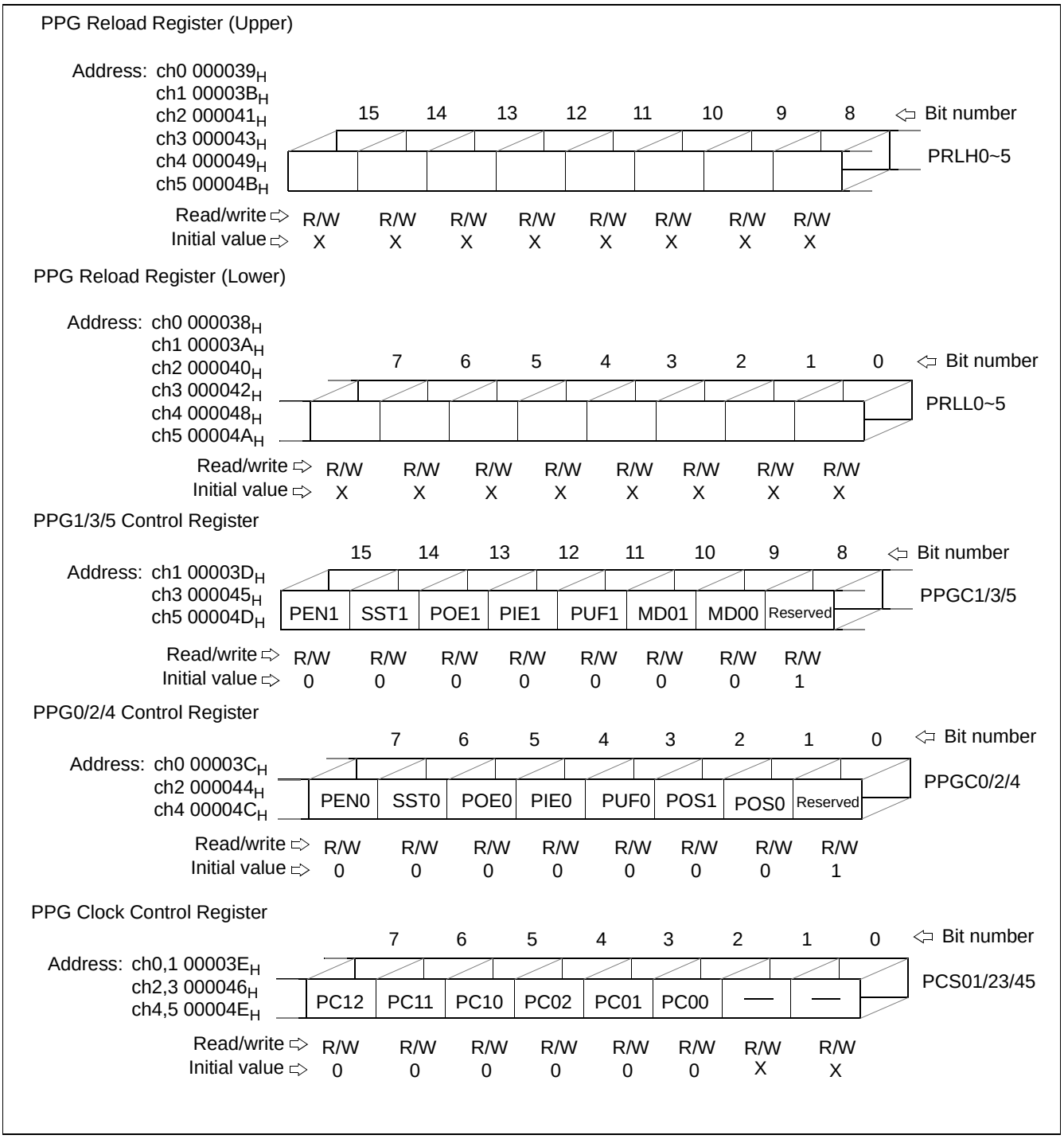


Figure 12.3-4 Registers of 8/16-bit PPG Timers

■ Waveform Generator Registers

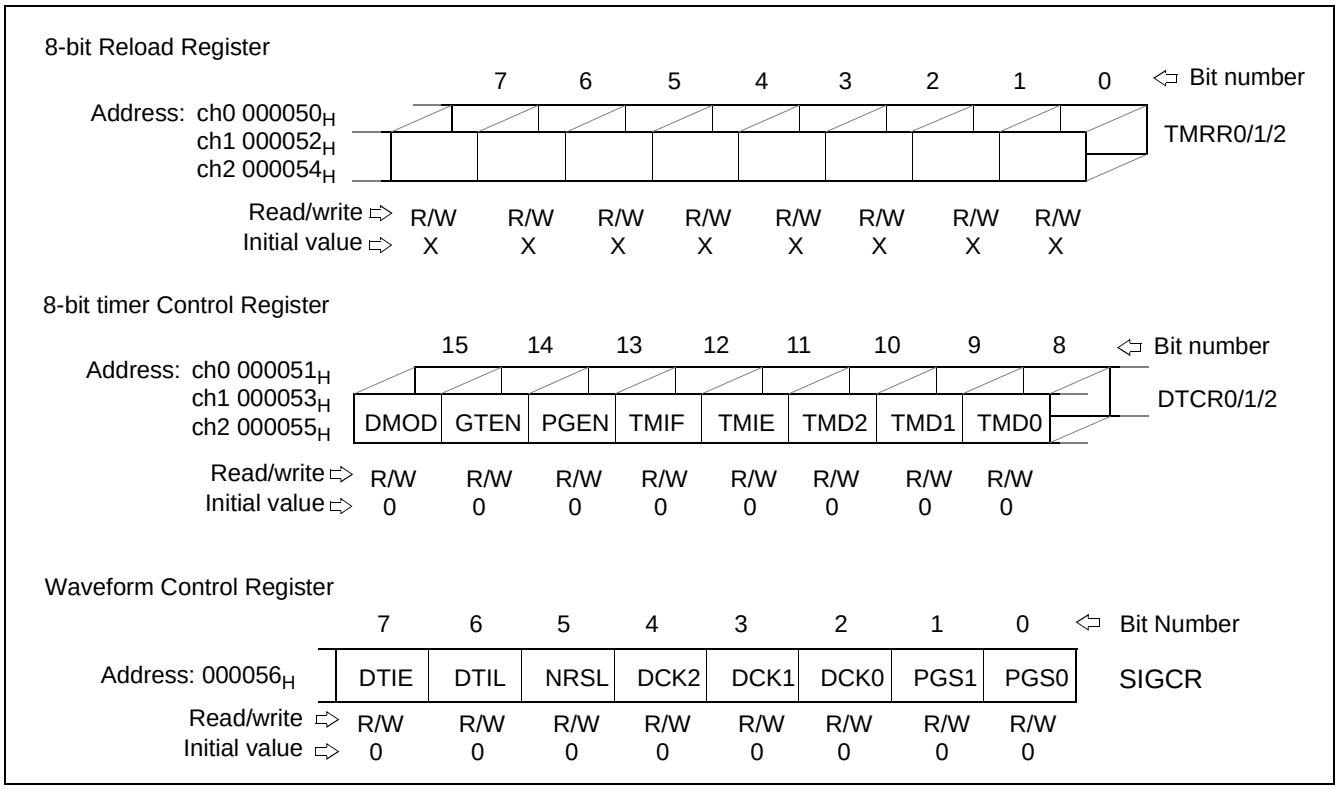


Figure 12.3-5 Registers of Waveform Generator

Memo

12.3.1 Registers of 16-bit Free-Run Timer

This section describe registers of 16-bit free-run timer and register details. Those register are compare clear registers, timer data register and timer control status register.

■ **16-bit Free-Run Timer Registers**

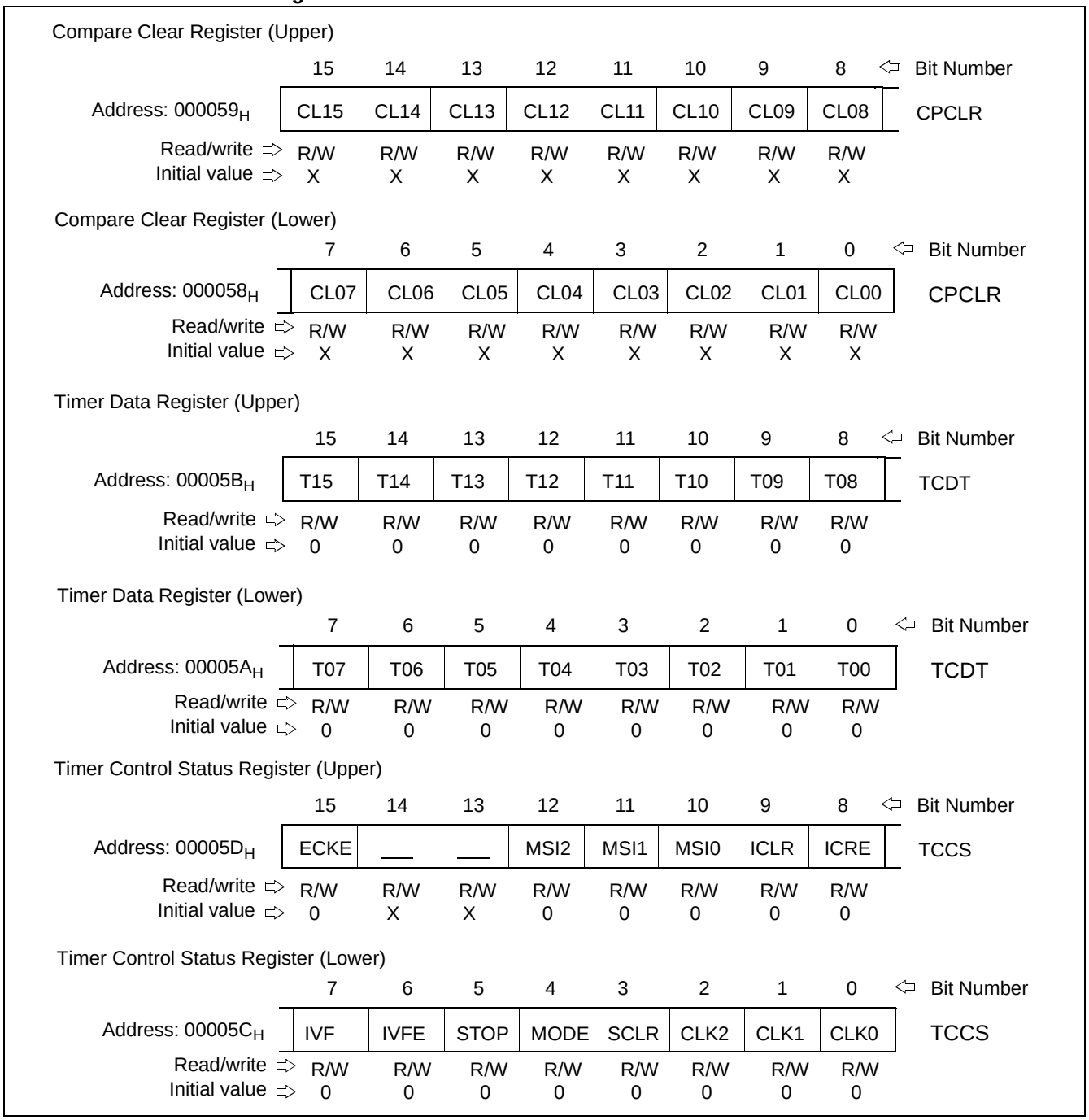


Figure 12.3.1-1 Registers of 16-bit Free-Run Timer

12.3.1.1 Compare Clear Register (CPCLR)

Compare clear register (CPCLR) is 16-bit register. When this register is matched with the count value of 16-bit free-run timer, the 16-bit free-run timer will be reset to "0000_H".

■ **Compare Clear Register (CPCR)**

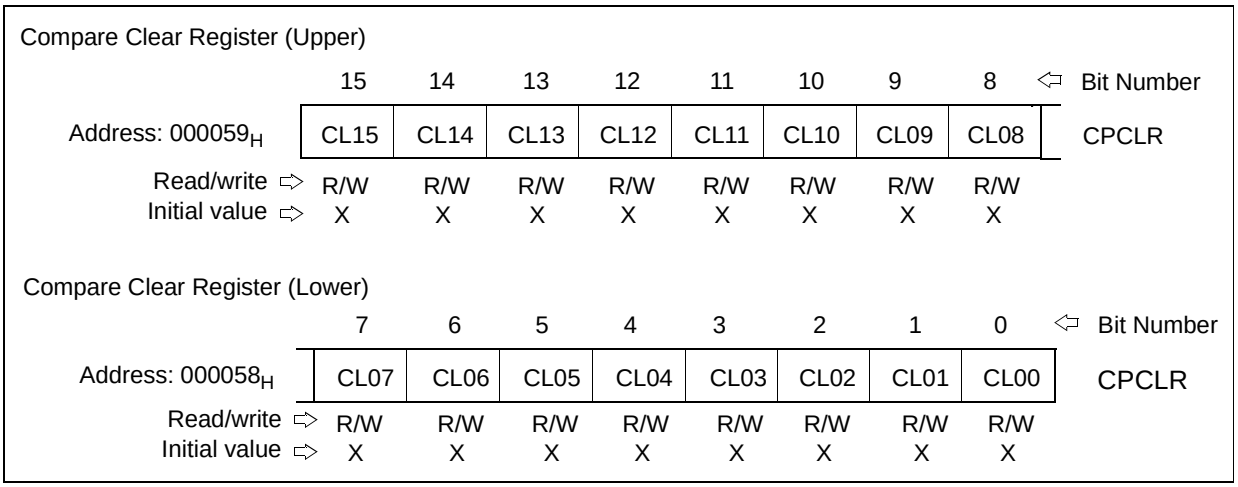


Figure 12.3.1.1-1 Compare Clear Register (CPCR)

Compare Clear Register is 16-bit register and is used to compare the count value of the 16-bit free-run timer. The initial value of this register is undetermined, so that the register must be set a value before starting an operation. Word access instruction to this register is recommended. When this register is matched with the count value of 16-bit free-run timer, 16-bit free-run timer will be reset to "0000_H" and the compare clear interrupt flag will be set. Furthermore, when the interrupt operation is enabled, interrupt request will be sent to the CPU.

12.3.1.2 Timer Data Register (TCDT)

The timer data register (TCDT) is used to read the count value of 16-bit free-run timer.

■ **Timer Data Register (TCDT)**

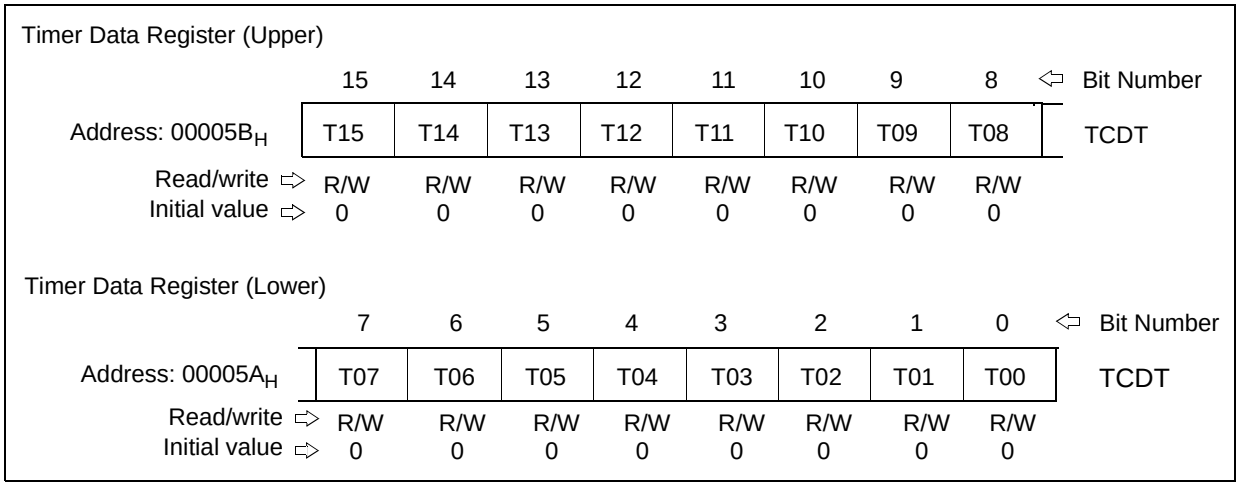


Figure 12.3.1.2-1 Timer Data Register

The timer data register is used to read the count value of the 16-bit free-run timer. The counter value is cleared to ’0000’ upon a reset. The timer value can be set by writing a value to this register. However, ensure that the value is written while the operation is stopped (STOP=1). Word access instruction to the timer data register is recommended.

The 16-bit free-run timer is initialized upon the following factors:

- Reset
- Clear bit (CLR) of control status register
- A match between compare clear register and the timer counter value (This can be performed only in an appropriate mode.)

Memo

12.3.1.3 Timer Control Status Register (TCCS)

The timer control status register (TCCS) is 16-bit register and used to control the operation of 16-bit free-run timer

■ **Timer control status register (Upper)**

Address	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Initial value
00005D$_H$	ECKE	—	—	MSI2	MSI1	MSI0	ICLR	ICRE	0XX00000$_B$
	R/W	—	—	R/W	R/W	R/W	R/W	R/W	

ICRE	Compare interrupt request enable bit
0	Enable interrupt request
1	Disable interrupt request

SC11	Compare interrupt request flag bit	
	Read	Write
0	Count value and set value is not matched	Clear this bit
1	Count value and set value is match	No effect

MSI2	MSI1	MSI0	Masking interrupt bits
0	0	0	Interrupt is generated when 1st match
0	0	1	Interrupt is generated when 2nd match
0	1	0	Interrupt is generated when 3rd match
0	1	1	Interrupt is generated when 4th match
1	0	0	Interrupt is generated when 5th match
1	0	1	Interrupt is generated when 6th match
1	1	0	Interrupt is generated when 7th match
1	1	1	Interrupt is generated when 8th match

ECKE	Clock selection bit
0	Select Internal clock
1	Select external clock from FRCK pin

X : Indeterminate
R/W : Read and write
: Initial value
— : not used

Figure 12.3.1.3-1 Timer Control Status Register (Upper)

Table 12.3.1.3-1 Timer Control Status Register (Upper) Bit

Bit		Function
Bit 15	**ECKE: Clock Selection bit**	• This bit is used to select either external or internal count clock source for the 16-bit free-run timer. Change this bit when the output compare and input capture are in stop status because the clock will be changed as soon as this bit is changed. **Note:** When selecting internal clock, set the count clock to bit 2~0 (CLK2~0). This count clock will become the base clock. When inputting clock from FRCK pin, set DDR1:bit7=”0”
Bit 14 **Bit 13**	**Unused bit**	• The read value is indeterminate. • Writing to this bit has no effect on the operation.
Bit 12 **Bit 11** **Bit 10**	**MIS2~0: Masking interrupt bit**	• These bits are used to set the number of times of masking the compare clear interrupt. 16-bit free-run timer will reload the count value every time when these 3 bits is not “000_B” and the 3 bits will be decrement by 1 until these 3 bits becomes “000_B”. The number of masking interrupt = the value of these 3 bits. (i.e. When masking two times and interrupt is generated in the 3rd time of the match, set the value = “010_B”. However, there is no masking interrupt when the value is “000_B”.
Bit 9	**ICLR: Compare Interrupt request flag bit**	• This bit is an interrupt request flag for compare clear. • When the compare clear register and 16-bit free-run timer value are matched and the counter is cleared, this bit will become “1”. • Interrupt is generated when the interrupt request enable bit (Bit8: ICRE) is set to “1”. • Writing “0” will clear this bit. • Writing “1” has no effect. • In Read-Modify-Write operation, “1” is always read.
Bit 8	**ICRE: Compare Interrupt request enable bit**	• This is the interrupt request enable bit for the compare clear. • When this bit is “1” and the interrupt flag (Bit9: ICLR) is set to “1”, an interrupt will be generated.

■ TImer control status register (Lower)

Address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	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	Clock frequency selection bit				
			Count clock	ϕ = 16MHz	ϕ = 8MHz	ϕ = 4MHz	ϕ = 1MHz
0	0	0	ϕ	62.5 ns	125 ns	0.25 µs	1 µs
0	0	1	ϕ/2	125 ns	0.25 µs	0.5 µs	2 µs
0	1	0	ϕ/4	0.25 µs	0.5 µs	1 µs	4 µs
0	1	1	ϕ/8	0.5 µs	1 µs	2 µs	8 µs
1	0	0	ϕ/16	1 µs	2 µs	4 µs	16 µs
1	0	1	ϕ/32	2 µs	4 µs	8 µs	32 µs
1	1	0	ϕ/64	4 µs	8 µs	16 µs	64 µs
1	1	1	ϕ/128	8 µs	16 µs	32 µs	128 µs

ϕ: Machine cycle

SCLR	Clear bit	
	Write	Read
0	No effect	Always read ”0”
1	Initialize counter to “0000$_H$”	

MODE	Initialization condition bit
0	Initialize counter by reset and clear bit (bit3: SCLR)
1	Initialize counter by reset, clear bit (bit3: SCLR) and match values with compare register 5 of output compare.

STOP	Counter enable bit
0	Counting is enabled (operation)
1	Counting is disabled (Stop)

IVFE	Timer interrupt request enable bit
0	Interrupt is disabled
1	Interrupt is enabled

IVF	Timer interrupt request flag bit	
	Read	Write
0	No interrupt Request	Clear this bit
1	Has interrupt request	No effect

R/W : Read and Write

: Initial value

Figure 12.3.1.3-2 Timer Control Status Register (Lower)

Table 12.3.1.3-2 Timer Control Status Register (Lower)

Bit		Function
Bit 7	**IVF: Timer Interrupt request flag bit**	• This bit is an interrupt request flag for 16-bit free-run timer. • This bit will be set to “1” when the 16-bit free-run timer is overflow. • Interrupt will be generated when the timer interrupt request enable bit (Bit6: IVFE) is set to “1”. • Writing “0” will clear this bit. • Writing “1” has no effect. • In Read-Modify-Write operation, “1” is always read.
Bit 6	**IVFE: Timer interrupt request enable bit**	• This is the interrupt request enable bit for 16-bit free-run timer clear. • When this bit is “1” and the timer interrupt request flag bit (Bit7: ICLR) is set to “1”, an interrupt will be generated.
Bit 5	**STOP: Counter enable bit**	• This bit is used to stop/start the counting of the 16-bit free-run timer. • Writing “1” to this bit will stop the counting of the 16-bit free-run timer • Writing “0” to this bit will start the counting of the 16-bit free-run timer. **Note:** When the 16-bit free-run timer is stopped, the output compare operation will be also stopped
Bit 4	**MODE: Initialization condition bit**	• This bit is used to set the initializing condition for the 16-bit free-run timer. • When it is “0”, 16-bit free-run counter will be initialized by the reset and the clear bit reset (bit 3:SCLR). • When it is “1”, 16-bit free-run timer will be initialized by not only the reset and the clear bit reset (bit 3:SCLR), but also when the count value is matched with the compare clear register. **Note:** Initialization of the counter value will be at the changing point of the count value.
Bit 3	**SCLR: Clear bit**	• This bit is used to initialize the 16-bit free-run time to the value of “0000_H”. • Writing “1” will initialize 16-bit free-run counter to “0000_H” • Writing “0” has no meaning. • Read value is always “0” **Note:** Initialization of the counter value will be at the changing point of the count value. **Note:** If the initialization when the counter is stopped, write “0000_H” to the data register.
Bit 2 Bit 1 Bit 0	**CLK2~0: Clock frequency section bit**	• This bit is used to select count clock for the 16-bit free-run timer. Change this bit when the output compare and input capture are in stop status because the clock will be changed as soon as this bit is changed.

12.3.2 Registers of 16-bit Output Compare

This section describe registers of 16-bit output compare and register details.

■ **Output Compare Registers**

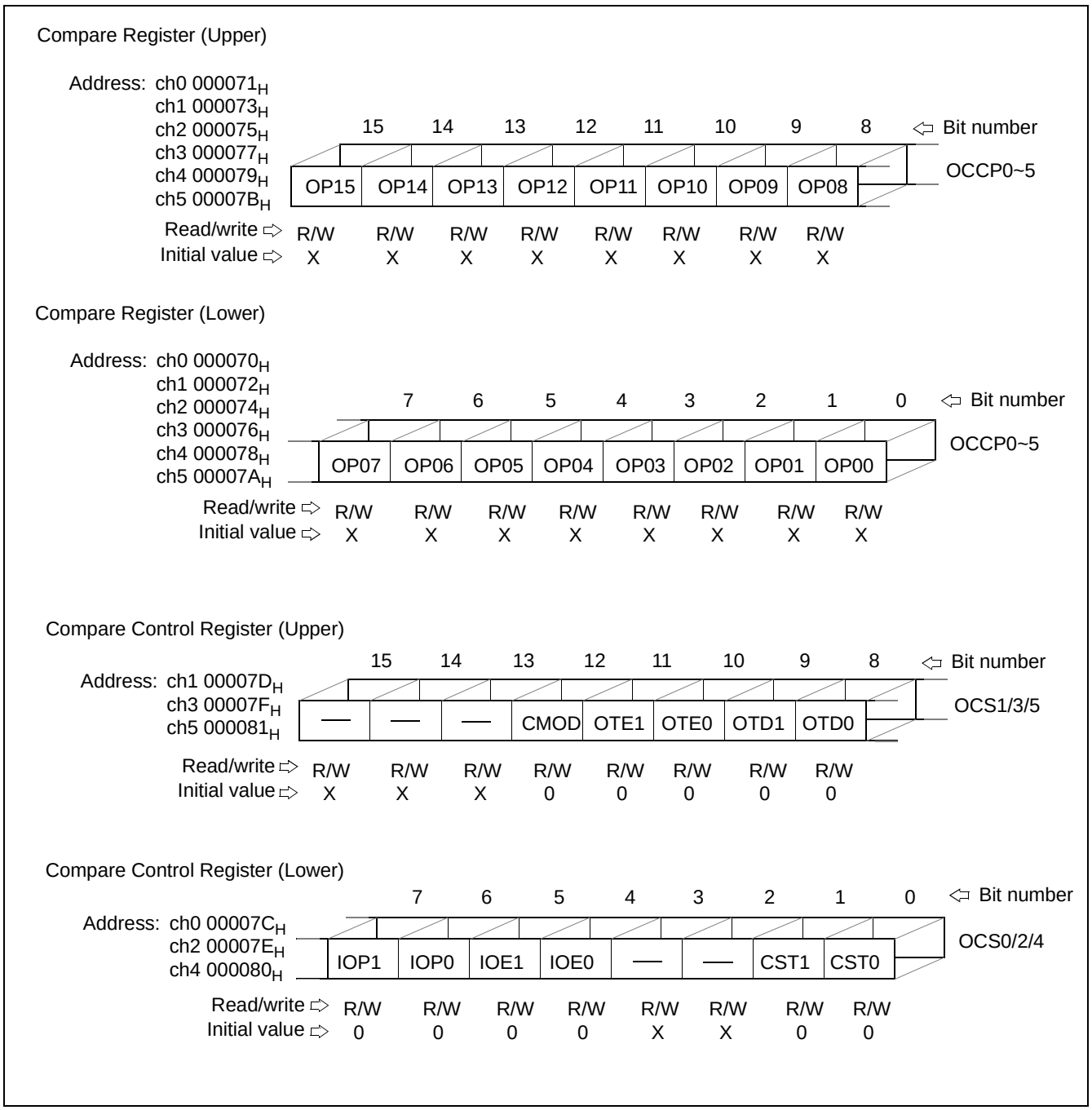


Figure 12.3.2-1 Registers of Output Compare

12.3.2.1 Compare Registers (OCCP0~5)

Compare registers (OCCP0~5) are used to set the compare value for the 16-bit free-run timer.

■ Compare Registers (OCCP0~5)

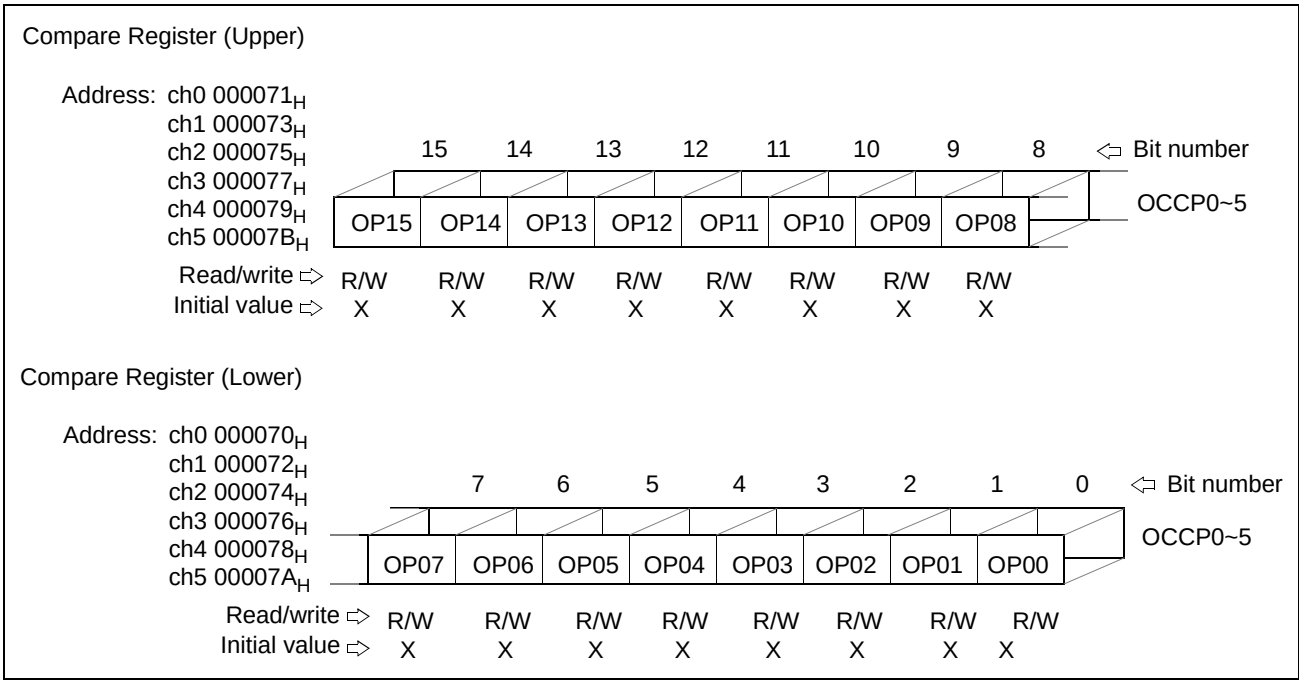


Figure 12.3.2.1-1 Compare Registers (OCCP0~5)

The compare registers (OCCP0~5) are 16-bit registers which are used to compare the count value of 16-bit free run timer. The initial values of the compare registers (OCCP0~5) are undetermined, so that the value must be set before enabling the operation. Word access instruction to this register is recommended. When the value of this register matches that of the 16-bit free-run timer, a compare signal is generated to set the output compare interrupt flag. If output is enabled, the output level (RT0~5) corresponding to the compare register (OCCP0~5) is reversed.

12.3.2.2 Compare Control Registers (OSC0/1/2/3/4/5)

Compare control register is used to control the output level, output enable, output reverse mode, compare operation enable, compare match interrupt enable and compare match interrupt flag for RTO0~5.

■ **Compare control register (Upper, OSC1/3/5)**

Address	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Initial value
ch1: 00007D$_H$ Ch3: 00007F$_H$	—	—	—	CMOD	OTE1	OTE0	OTD1	OTD0	XXX00000$_B$
Ch5: 000081$_H$	—	—	—	R/W	R/W	R/W	R/W	R/W	

OTD0	Output level bit
0	Output “0” for RT0/RT2/RT4
1	Output “1” for RT0/RT2/RT4

OTD1	Output level bit
0	Output “0” for RT1/RT3/RT5
1	Output “1” for RT1/RT3/RT5

OTE0	Output enable bit
0	General-purpose port (port 30/32/34)
1	Output compare output pin (RT0/RT2/RT4)

OTE1	Output enable bit
0	General-purpose port (Port 31/33/35)
1	Output compare output pin (RT1/RT3/RT5)

CMOD	Output level reverse mode bit
0	RT0/2/4: The level is reversed upon a match with compare register 0/ 2/ 4. RT1/3/5: The level is reversed upon a match with compare register 1/3/5 respectively
1	RT0/2/4: The level is reversed upon a match with compare register 0/2/4. RT1/3/5: The level is reversed upon a match with compare register 0 or 1, 2 or 3, 4 or 5.

X : Indeterminate
R/W : Read and write
: Initial value
— : not used

Figure 12.3.2.2-1 Compare Control Register (Upper, OSC1/3/5)

Table 12.3.2.2-1 Compare Control Register (Upper, OSC1/3/5) Bit

Bit		Function
Bit 15 **Bit 14** **Bit 13**	**Unused bit**	• The read value is indeterminate. • Writing to this bit has no effect on the operation.
Bit 12	**CMOD: Output Level reverse mode bit**	• CMOD is used to switch the pin output level reverse mode upon a match while pin output is enabled (OTE1=1 or OTE0=1). • When CMOD=0 (default), the output level of the pin is reversed upon a match with corresponding compare register. RT0/2/4: The level is reversed upon a match between the 16-bit free-run timer and compare register 0/2/4. RT1/3/5: The level is reversed upon a match between the 16-bit free-run timer and compare register 1/3/5. • When CMOD=1, the output level of the pin corresponding to compare register is reversed as same as when CMOD=0. However, the output level of the pin (RT1) corresponding to compare register 1 is reversed only when a match is detected in both compare register 0 and 1. If compare registers 0 and 1 have the same value, the same operation as when only one compare register is used. RT0/2/4: The level is reversed upon a match between the 16-bit free-run timer and compare register 0/2/4. RT1/3/5: The level is reversed upon a match between the 16-bit free-run timer and compare register 0 or 1, 2 or 3, 4 or 5.
Bit 11	**OTE1: Output enable bit**	• This bit is used to enable output compare pin output for output compare 1/3/5. • The initial value for these bits is '0'. **Note:** The output enable bit of the waveform generator must be set when dead-time timer is used in the waveform generator.
Bit 10	**OTE0: Output enable bit**	• This bit is used to enable output compare pin output for output compare 0/2/4. • The initial value for these bits is '0.' **Note:** The output enable bit of the waveform generator must be set when dead-time timer is used in the waveform generator.
Bit 9	**OTD1: Output level bit**	• This bit is used to change the pin output level for output compare 1/3/5 when the output compare pin output is enabled. • The initial value of the compare pin output is '0.' • Ensure that the compare operation is stopped before a value is written. When reading this bit, this bit indicate the output compare pin output value
Bit 8	**OTD0: Output level bit**	• This bit is used to change the pin output level for output compare 0/2/4 when the output compare pin output is enabled. • The initial value of the compare pin output is '0.' • Ensure that the compare operation is stopped before a value is written. When reading this bit, this bit indicates the output compare pin output value.

■ Compare control register (Lower, OSC0/2/4)

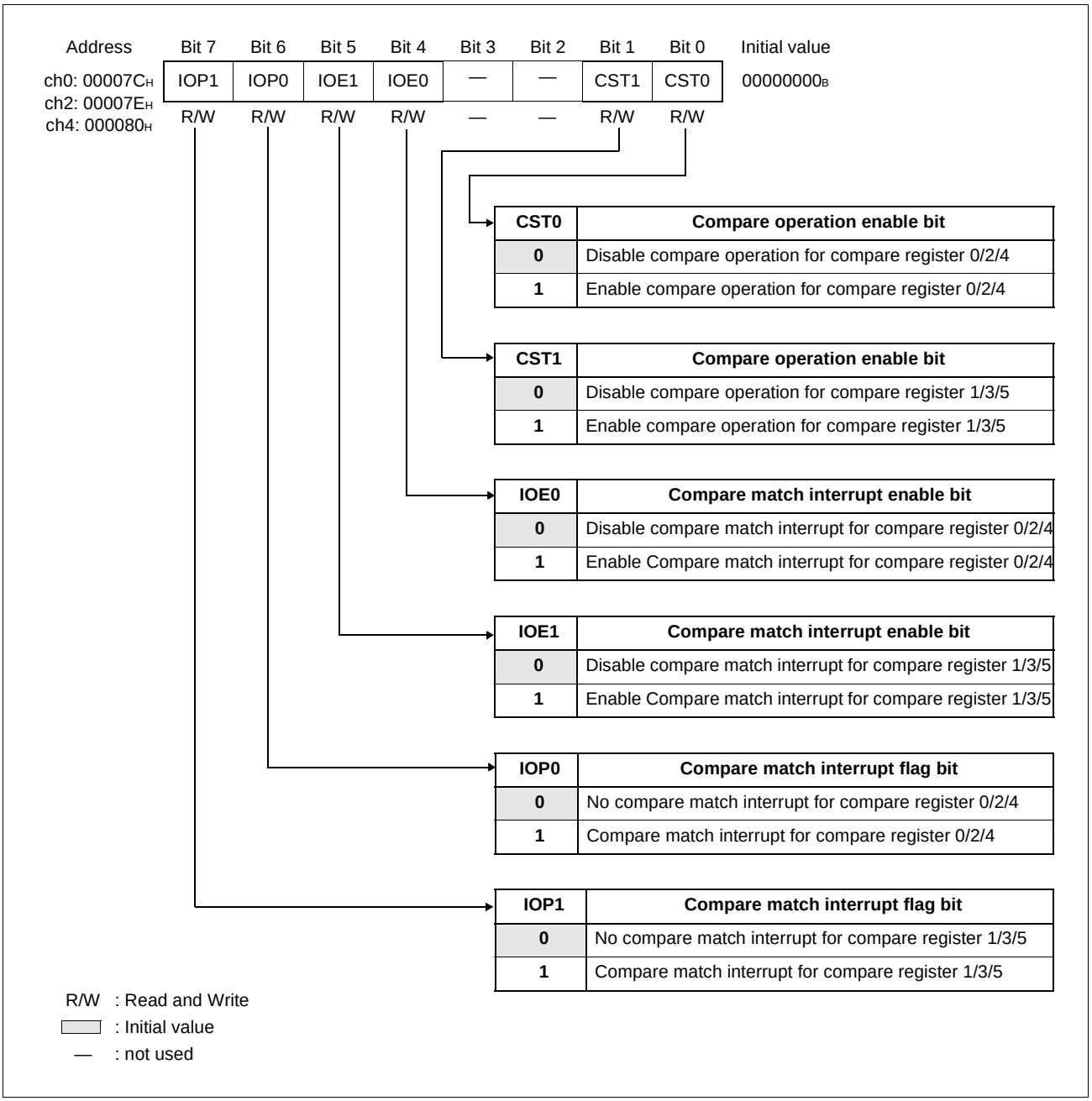


Figure 12.3.2.2-2 Compare Control Register (Lower, OSC0/2/4)

Table 12.3.2.2-2 Compare Control Register (Lower, OSC0/2/4)

Bit		Function
Bit 7	**IOP1: Compare match interrupt flag bit**	• This bit is an interrupt flag for when compare register 1/3/5 is matched with the value of 16-bit free-run timer. • '1' is written to this bit when the compare register value matches the 16-bit free-run timer value. • While the interrupt request bits (IOE1) is enabled, an output compare interrupt occurs when the IOP1 bit is set. • Writing “0” will clear this bit. • Writing “1” has no effect. • In Read-Modify-Write operation, “1” is always read.
Bit 6	**IOP0: Compare match interrupt flag bit**	• This bit is an interrupt flag for when compare register 0/2/4 is matched with the value of 16-bit free-run timer. • '1' is written to this bit when the compare register value matches the 16-bit free-run timer value. • While the interrupt request bits (IOE0) is enabled, an output compare interrupt occurs when the IOP0 bit is set. • Writing “0” will clear this bit. • Writing “1” has no effect. • In Read-Modify-Write operation, “1” is always read.
Bit 5	**IOE1: Compare match interrupt enable bit**	• This bit is used to enable output compare interrupt for compare register 1/3/5. • While the '1' is written to this bit, an output compare interrupt occurs when an interrupt flag (IOP1) is set.
Bit 4	**IOE0: Compare match interrupt enable bit**	• This bit is used to enable output compare interrupt for compare register 0/2/4. • While the '1' is written to this bit, an output compare interrupt occurs when an interrupt flag (IOP0) is set.
Bit 3 **Bit 2**	**Unused bit**	• The read value is indeterminate. • Writing to this bit has no effect on the operation.
Bit 1	**CST1: Compare operation enable bit**	• This bit is used to enable the compare operation between 16-bit free-run timer and compare register 1/3/5 • Ensure that a value is written into the compare register and timer data register before the compare operation is enabled. **Note:** Since output compare is synchronized with the 16-bit free-run timer clock, stopping the 16-bit free-run timer stops compare operation.
Bit 0	**CST0: Compare operation enable bit**	• This bit is used to enable the compare operation between 16-bit free-run timer and compare register 0/2/4 • Ensure that a value is written into the compare register and timer data register before the compare operation is enabled. **Note:** Since output compare is synchronized with the 16-bit free-run timer clock, stopping the 16-bit free-run timer stops compare operation.

12.3.3 Registers of 16-bit Input Capture

The section describe the registers of 16-bit input capture, which are input capture registers and capture control registers.

■ **Input capture registers**

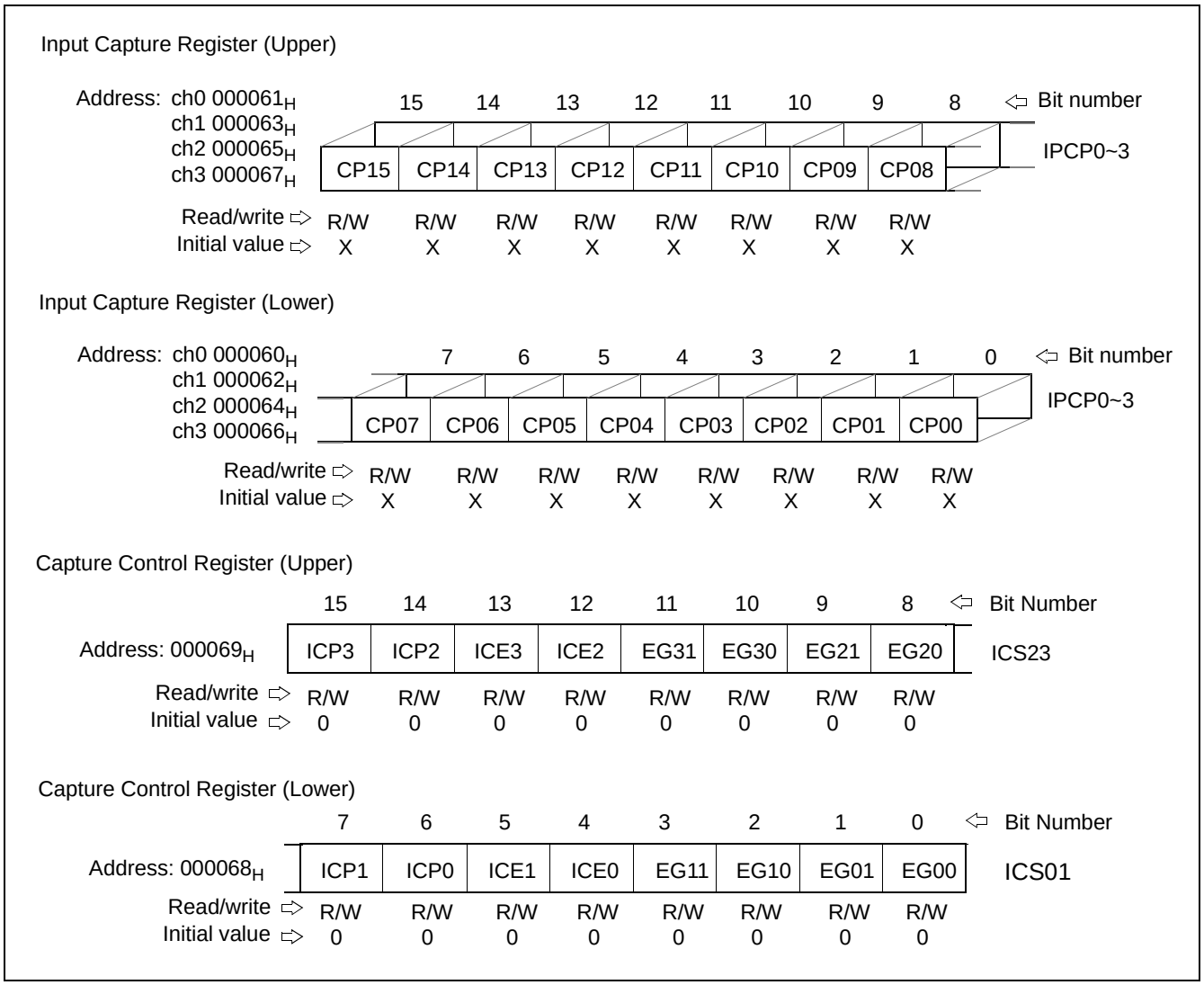


Figure 12.3.3-1 Registers of 16-bit Input Capture

12.3.3.1 Input Capture Register (IPCP0~3)

Input Capture registers are used to hold the count value of 16-bit timer when a valid edge of the input waveform is detected.

■ **Input capture register (IPCP0~3)**

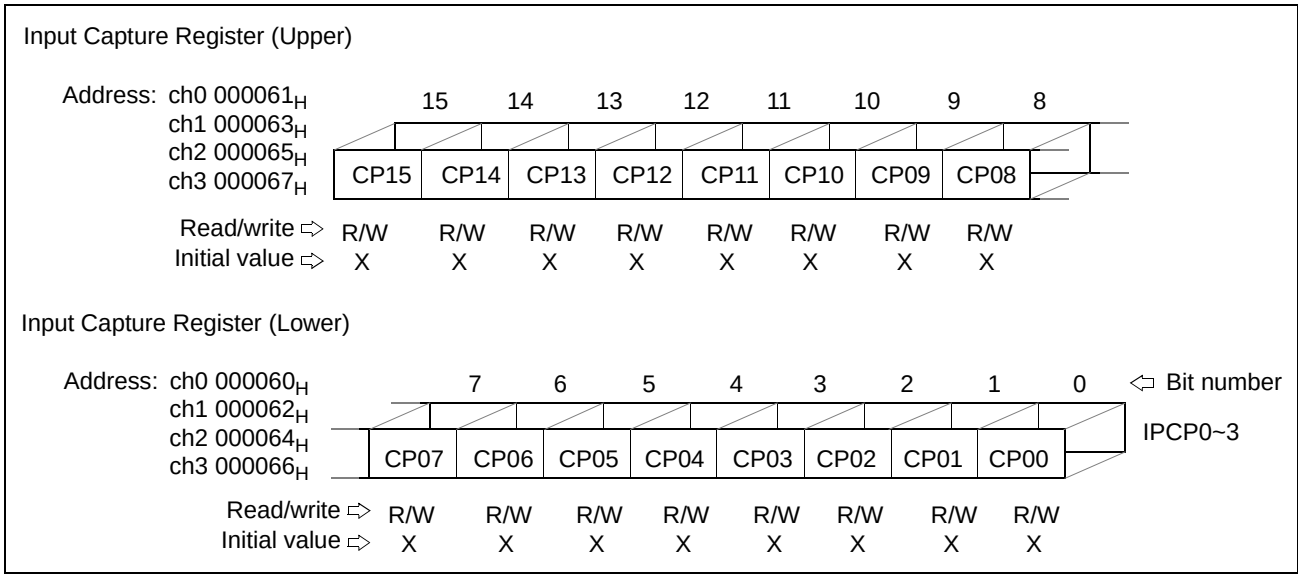


Figure 12.3.3.1-1 Input Capture Registers (IPCP0~3)

This register is used to store the value of the 16-bit timer when a valid edge of the corresponding external pin input waveform is detected. (Word access instruction to this register is recommended. No data can be written to this register.)

12.3.3.2 Capture Control Registers (ICS23, ICS01)

Capture control registers (ICS23, ICS01) are used to control edge selection, interrupt request enable and interrupt request flag for input capture 0~3.

■ Capture control register (ICS23)

Address	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Initial value
000069H	ICP3	ICP2	ICE3	ICE2	EG31	EG30	EG21	EG20	00000000B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

EG21	EG20	Edge selection bit (input capture 2)
0	0	No Edge detection (Stop)
0	1	Rising edge detection
1	0	Falling edge detection
1	1	Both edges detection

EG31	EG30	Edge selection bit (input capture 3)
0	0	No Edge detection (Stop)
0	1	Rising edge detection
1	0	Falling edge detection
1	1	Both edges detection

ICE2	Interrupt request enable bit (Input capture 2)
0	Disable interrupt request
1	Enable interrupt request

ICE3	Interrupt request enable bit (Input capture 3)
0	Disable interrupt request
1	Enable interrupt request

ICP2	Interrupt request flag bit (Input capture 2)	
	Read	Write
0	No valid detected	Clear this bit
1	Valid detected	No effect

ICP3	Interrupt request flag bit (Input capture 3)	
	Read	Write
0	No valid detected	Clear this bit
1	Valid detected	No effect

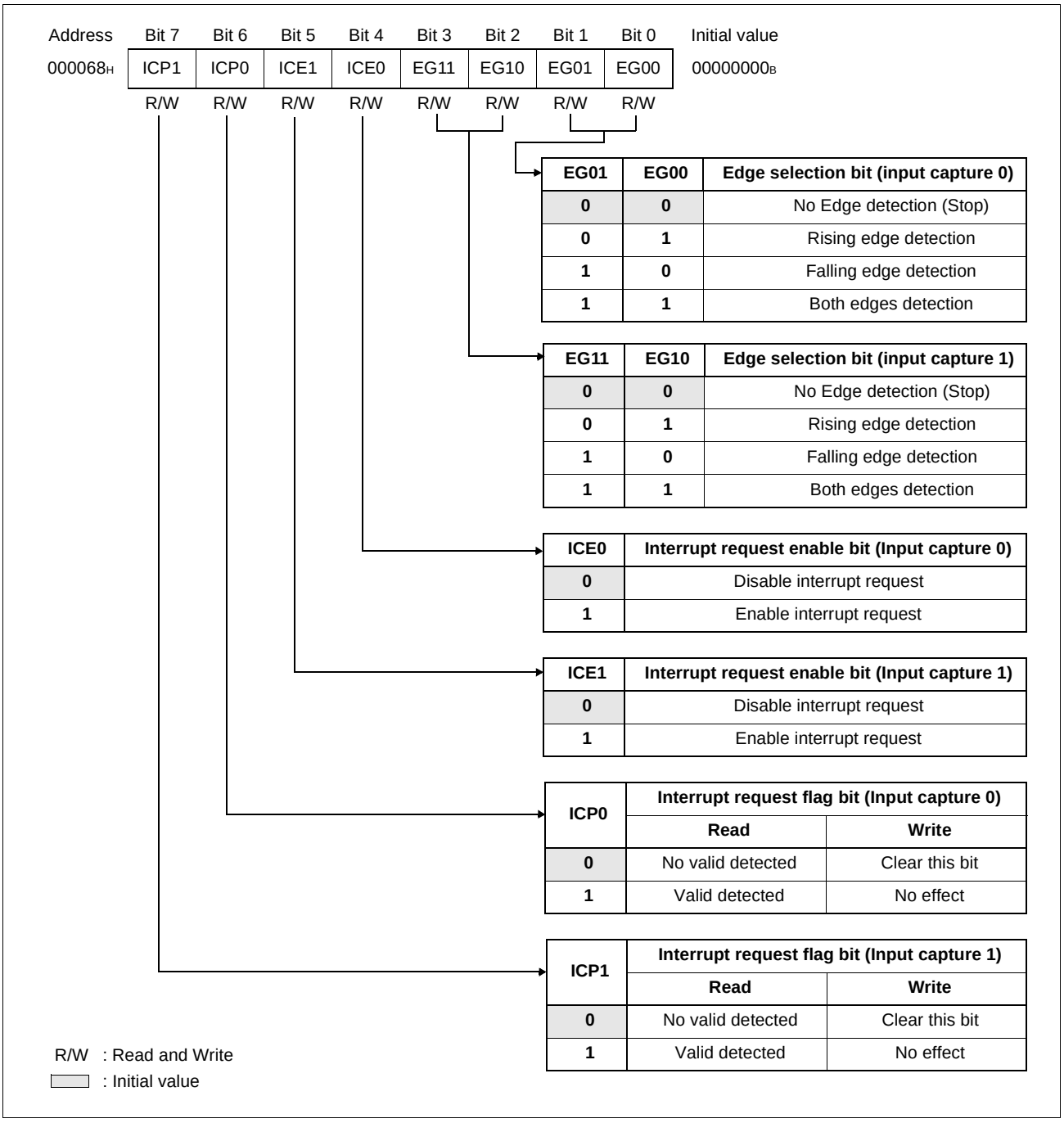
R/W : Read and Write
: Initial value

Figure 12.3.3.2-1 Capture Control Register (ICS23)

Table 12.3.3.2-1 Capture Control Register (ICS23) Bit

Bit		Function
Bit 15	**ICP3: Interrupt request flag bit (Input capture 3)**	• This bit is used as interrupt request flag for input capture 3. • "1" is written to this bit upon detection of a valid edge of an external input pin. • While the interrupt enable bit (ICE3) is set, an interrupt can be generated upon detection of a valid edge. • Writing "0" will clear this bit. • Writing "1" has no effect. • In Read-Modify-Write operation, "1" is always read.
Bit 14	**ICP2: Interrupt request flag (Input capture 2)**	• This bit is used as interrupt request flag for input capture 2. • "1" is written to this bit upon detection of a valid edge of an external input pin. • While the interrupt enable bit (ICE2) is set, an interrupt can be generated upon detection of a valid edge. • Writing "0" will clear this bit. • Writing "1" has no effect. • In Read-Modify-Write operation, "1" is always read.
Bit 13	**ICE3: Interrupt request enable bit (Input capture 3)**	• This bit is used to enable input capture interrupt request for input capture 3. • While "1" is written to this bit, an input capture interrupt is generated when the interrupt flag (ICP3) is set.
Bit 12	**ICE2: Interrupt request enable bit (Input capture 2)**	• This bit is used to enable input capture interrupt request for input capture 2. • While "1" is written to this bit, an input capture interrupt is generated when the interrupt flag (ICP2) is set.
Bit 11 Bit 10	**EG31, EG30: Edge selection bit (Input capture 3)**	• These bits are used to specify the valid edge polarity of an external input for input capture 3. • These bits are also used to enable input capture operation.
Bit 9 Bit 8	**EG21, EG20: Edge selection bit (Input capture 2)**	• These bits are used to specify the valid edge polarity of an external input for input capture 2. • These bits are also used to enable input capture operation.

■ Capture control register (ICS01)



EG01	EG00	Edge selection bit (input capture 0)
0	0	No Edge detection (Stop)
0	1	Rising edge detection
1	0	Falling edge detection
1	1	Both edges detection

EG11	EG10	Edge selection bit (input capture 1)
0	0	No Edge detection (Stop)
0	1	Rising edge detection
1	0	Falling edge detection
1	1	Both edges detection

ICE0	Interrupt request enable bit (Input capture 0)
0	Disable interrupt request
1	Enable interrupt request

ICE1	Interrupt request enable bit (Input capture 1)
0	Disable interrupt request
1	Enable interrupt request

ICP0	Interrupt request flag bit (Input capture 0)	
	Read	Write
0	No valid detected	Clear this bit
1	Valid detected	No effect

ICP1	Interrupt request flag bit (Input capture 1)	
	Read	Write
0	No valid detected	Clear this bit
1	Valid detected	No effect

Figure 12.3.3.2-2 Capture Control Register (ICS01)

Table 12.3.3.2-2 Capture Control Register (ICS01)

Bit		Function
Bit 7	**ICP1: Interrupt request flag bit (Input capture 1)**	• This bit is used as interrupt request flag for input capture 1. • “1” is written to this bit upon detection of a valid edge of an external input pin. • While the interrupt enable bit (ICE1) is set, an interrupt can be generated upon detection of a valid edge. • Writing “0” will clear this bit. • Writing “1” has no effect. • In Read-Modify-Write operation, “1” is always read.
Bit 6	**ICP0: Interrupt request flag (Input capture 0)**	• This bit is used as interrupt request flag for input capture 0. • “1” is written to this bit upon detection of a valid edge of an external input pin. • While the interrupt enable bit (ICE0) is set, an interrupt can be generated upon detection of a valid edge. • Writing “0” will clear this bit. • Writing “1” has no effect. • In Read-Modify-Write operation, “1” is always read.
Bit 5	**ICP1: Interrupt request enable bit (Input capture 1)**	• This bit is used to enable input capture interrupt request for input capture 1. • While “1” is written to this bit, an input capture interrupt is generated when the interrupt flag (ICP1) is set.
Bit 4	**ICP0: Interrupt request enable bit (Input capture 0)**	• This bit is used to enable input capture interrupt request for input capture 0. • While “1” is written to this bit, an input capture interrupt is generated when the interrupt flag (ICP0) is set.
Bit 3 Bit 2	**EG11, EG10: Edge selection bit (Input capture 1)**	• These bits are used to specify the valid edge polarity of an external input for input capture 1. • These bits are also used to enable input capture operation.
Bit 1 Bit 0	**EG01, EG00: Edge selection bit (Input capture 0)**	• These bits are used to specify the valid edge polarity of an external input for input capture 0. • These bits are also used to enable input capture operation.

12.3.4 Register of 8/16-bit PPG Timer

This section describe registers of 8/16-bit PPG timer, that are PPG reload timer x 6, PPG control register x 6 and PPG clock control register x 3.

■ **8/16-bit PPG Timer Registers**

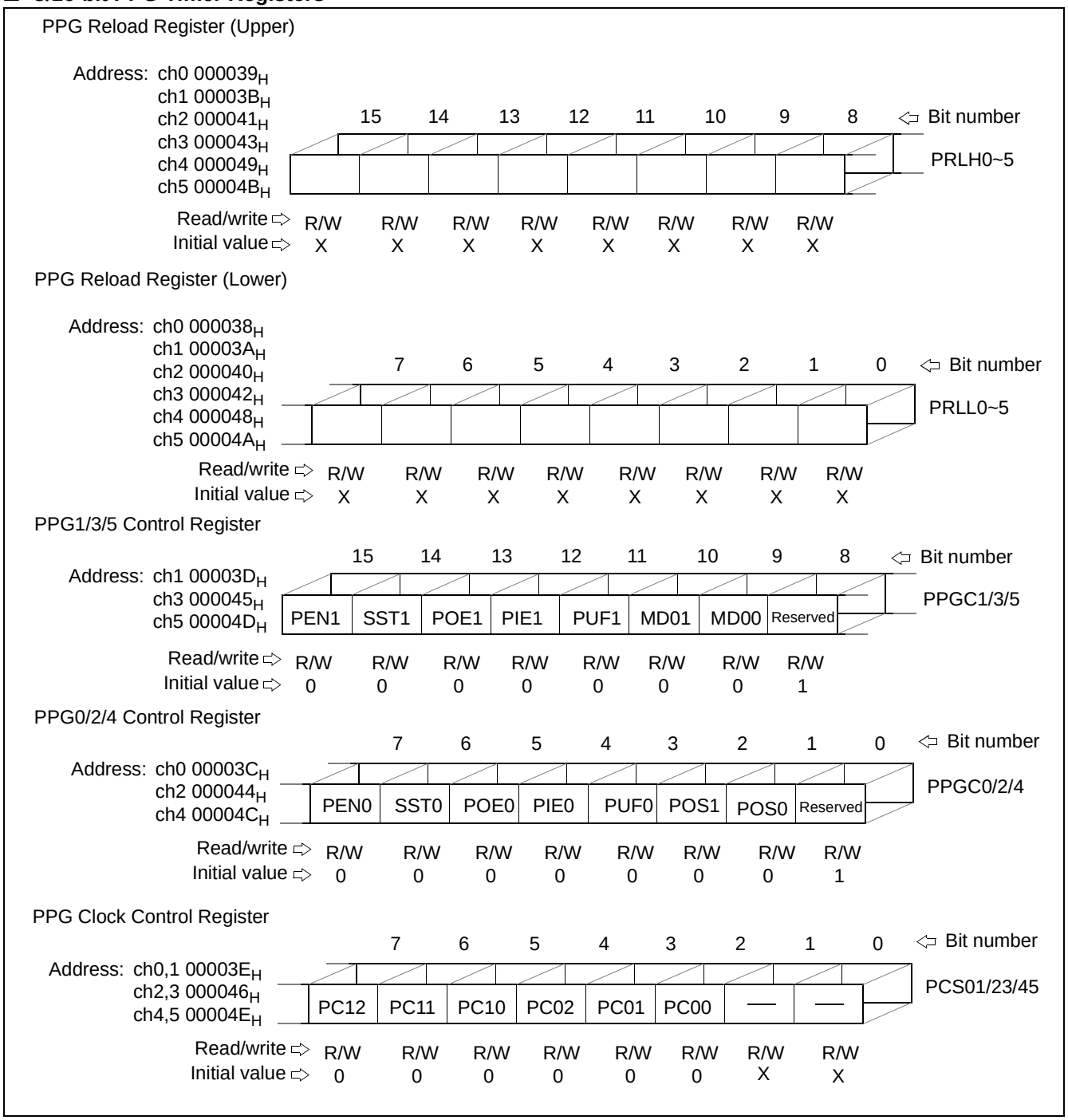


Figure 12.3.4-1 Registers of 8/16-bit PPG Timers

12.3.4.1 PPG Reload Register (PRLH0~5, PRLL0~5)

PPG reload registers (PRLH0~5, PRLL0~5) are 8-bit register that is used to store the reload values for the down counter.

■ **PPG reload register (PRLH0~5, PRLL0~5)**

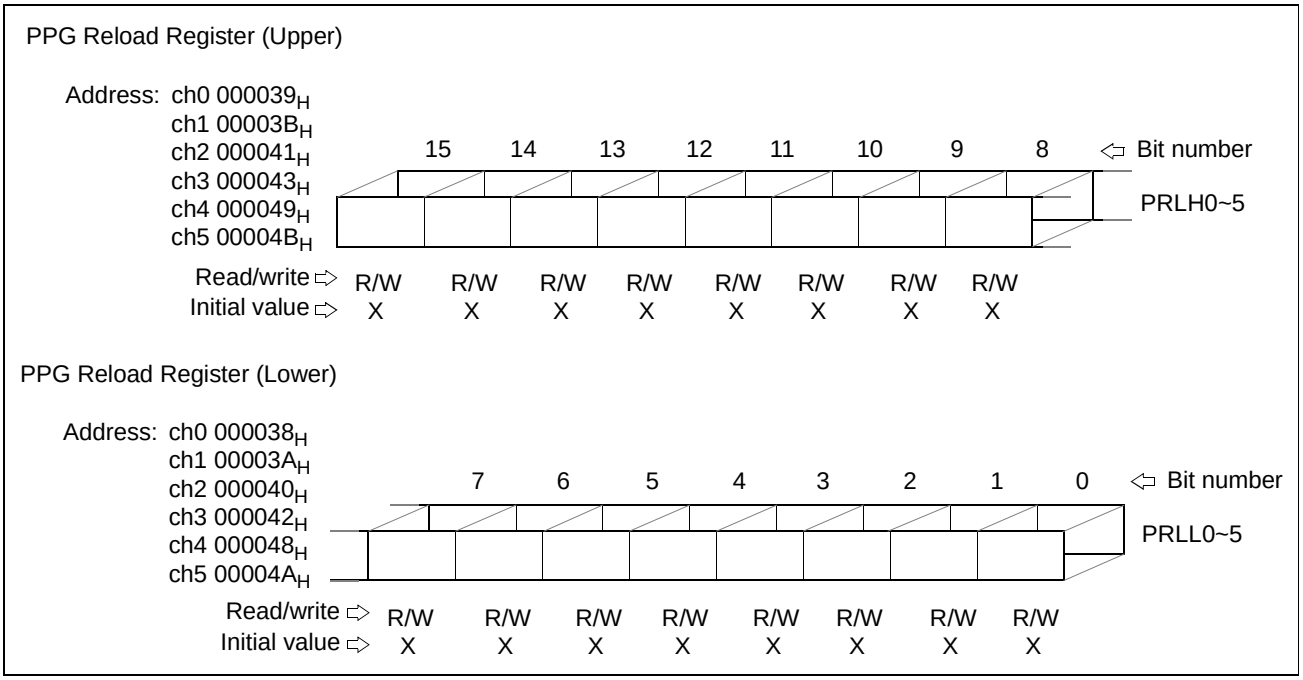


Figure 12.3.4.1-1 PPG Reload Register (PRLH0~5, PRLL0~5)

These are 8-bit registers that is used to store the reload values for the down counter. Their function are described below.

Table 12.3.4.1-1 Function of PPG Reload registers

Register Name	Function
PRLL0-5	Store the Lower byte of reload value
PRLH0~5	Store the Upper byte of reload value

These registers are readable and writable.

Note: In 8+8-bit PPG mode, setting different values in PRLL and PRLH of ch0/2/4 may cause the PPG waveform of ch1/3/5 to vary in each cycle. Write the same value to PRLL and PRLH of ch0/2/4.

12.3.4.2 PPG Control Register (PPGC0~5)

PPG control registers (PPGC0~5) are used to control the operation mode, interrupt request enable, interrupt request flag, output enable, start selection, operation enable and the output level polarity for PPG0~5 timers.

■ **PPG1/3/5 control register (PPGC1/3/5)**

Address	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Initial value
ch1: 00003D$_H$ ch3: 000045$_H$ ch5: 00004D$_H$	PEN1	SST1	POE1	PIE1	PUF1	MD01	MD00	Reserved	00000001$_B$
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

M0D1	MOD0	PPG1/3/5 timer operation mode bit
0	0	8-bit PPG 2ch independent mode
0	1	8 + 8-bit PPG mode
1	0	Reserved (setting inhibited)
1	1	16-bit PPG 1ch mode

PUF1	Interrupt request flag bit for PPG1/3/5	
	Read	Write
0	No overflow of PPG1/3/5 timer	Clear this bit
1	Overflow of PPG1/3/5 timer	No effect

PIE1	Interrupt request enable bit for PPG1/3/5
0	Disable interrupt request for PPG1/3/5
1	Enable interrupt request for PPG1/3/5

POE1	Output enable bit for PPG1/3/5
0	General-purpose port (P42/44/46)
1	PPG output port for PPG1/3/5

SST1	Start select bit for PPG1/3/5
0	PPG1/3/5 will be operated when PEN1="1".
1	PPG1/3/5 will be operated when GATE="H".

PEN1	Operation enable bit for PPG1/3/5
0	Sop operation for PPG1/3/5
1	Start operation for PPG1/3/5

R/W : Read and Write
: Initial value

Figure 12.3.4.2-1 PPG1/3/5 Control Register (PPGC1/3/5)

Table 12.3.4.2-1 PPG1/3/5 Control Register (PPGC1/3/5) Bit

Bit		Function
Bit 15	**PEN1: Operation enable bit for PPG1/3/5**	• This bit is used to enable the operation of PPG1/3/5. • When SST1="0", writing "1" will start the PPG1/3/5 operation and writing "0" will stop the operation. • When SST1="1", setting this bit cause no effect.
Bit 14	**SST1: Start select bit for PPG1/3/5**	• This bit is used to select the start condition of PPG1/3/5. • After writing "0" to this bit, the PPG1/3/5 will be operated depending on the PEN1 bit. • Writing "1" to PEN1 bit while this bit is "1", PPG1/3/5 will be operated while the GATE signal is "H".
Bit 13	**POE1: Output enable bit**	• This bit is used to select either the pulse output of PPG1/3/5 or general-purpose port.
Bit 12	**PIE1: Interrupt request enable bit for PPG1/3/5**	• This bit is used to enable an interrupt request for PPG1/3/5. • While "1" is written to this bit, PPG1/3/5 is generated an interrupt when the interrupt flag (PUF1) is set.
Bit 11	**PUF1: Interrupt request flag bit for PPG1/3/5**	• This bit is used as an interrupt request flag for PPG1/3/5 underflow. • In 8-bit PPG 2ch independent mode or 8+8-bit PPG mode, "1" is written to this bit when an underflow occurs as a result of the ch0/2/4 counter value becoming between 00H and FFH. • In 16-bit PPG 1ch mode, "1" is written to this bit when an underflow occurs as a result of the ch1/ch0, ch2/ch3 and ch4/ch5 counter value becoming between 0000H and FFFFH. • Writing "0" will clear this bit. Writing "1" has no effect. • In Read-Modify-Write operation, "1" is always read.
Bit 10 **Bit 9**	**MD01, MD00: PPG1/3/5 timer operation mode bit**	• This bit is used to select operation mode of PPG1/3/5. **Note:** Do not set "10" in this bit. **Note:** When these bits are set to "01", do not set the PPGC0/2/4:PEN0 and PPGC1/3/5:PEN1 bits to "01". It is recommended that the PEN0 and PEN1 bits be set to "11" or "00". When MOD1 and MOD0 are set to "11", rewrite PPGC0/2/4 and PPGC1/3/5 by word transfer to set the PEN0 and PEN1 bits to "11" or "00". **Note:** When these bits are set to "01", be sure to set the PPGC0/2/4:PEN0 bit to "1" before setting the PPGC1/3/5:SST1 bit to "1". When the PEN0 bit is set to "0", be sure to set the SST1 bit to "0" in advance. When MOD1 and MOD0 are set to "11", rewrite PPGC0/2/4 and PPGC1/3/5 by word transfer to set the SST0 and SST1 bits to "11" or "00".
Bit 8	**Reserved bit**	• Always setting this bit to "1" when setting PPGC1/3/5

■ PPG0/2/4 control register (PPGC0/2/4)

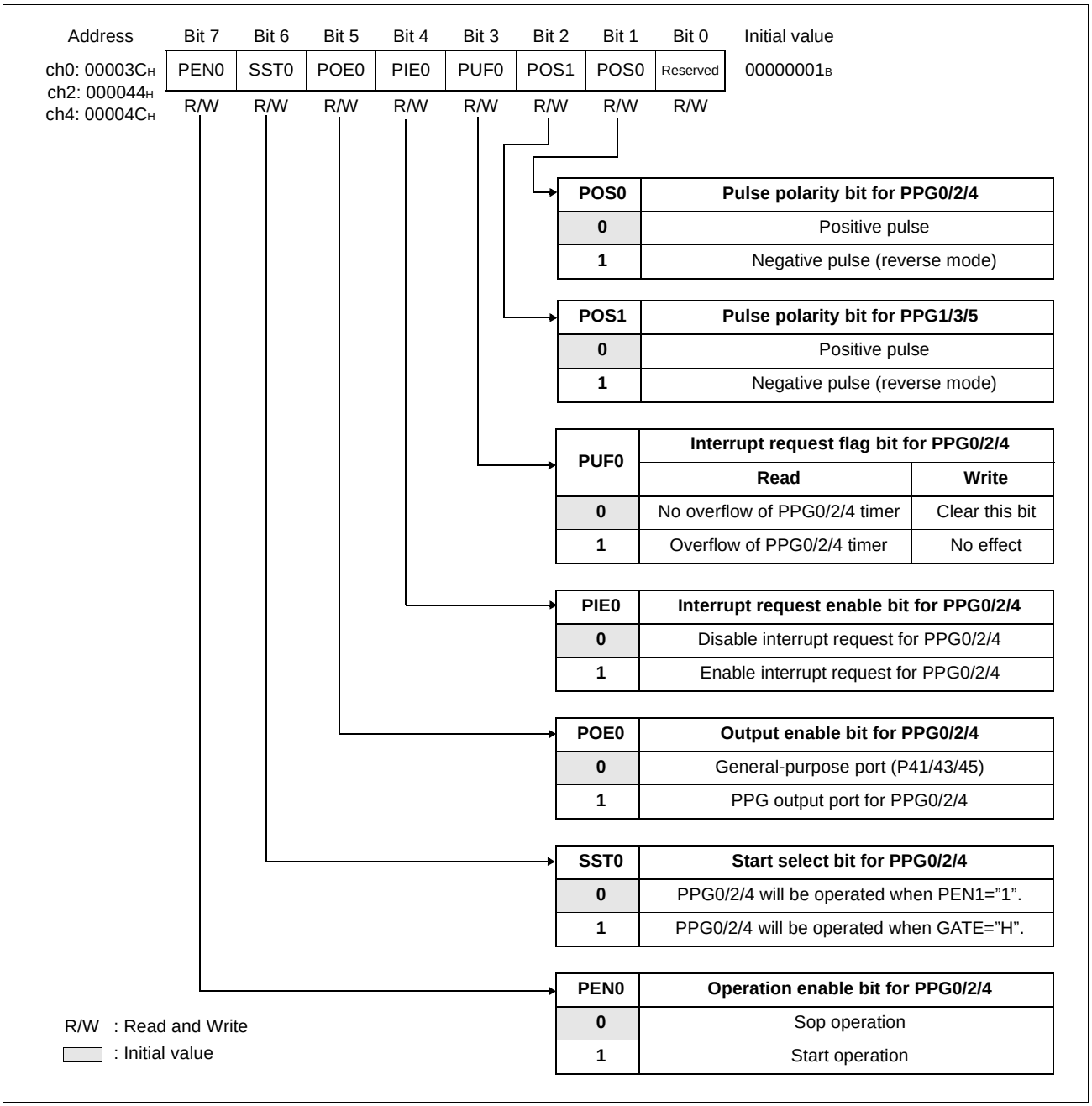


Figure 12.3.4.2-2 PPG0/2/4 Control Register (PPGC0/2/4)

Table 12.3.4.2-2 PPG0/2/4 Control Register (PPG0/2/4)

Bit		Function
Bit 7	**PEN0: Operation enable bit for PPG0/2/4**	• This bit is used to enable the operation of PPG0/2/4. • When SST1=”0”, writing “1” will start the PPG0/2/4 operation and writing “0” will stop the operation. • When SST1=”1”, setting this bit cause no effect.
Bit 6	**SST0: Start select bit for PPG0/2/4**	• This bit is used to select he start condition of PPG0/2/4. • After writing “0” to this bit, the PPG0/2/4 will be operated depending on the PEN1 bit. • Writing “1” Writing “1” to PEN0 bit while this bit is “1”, PPG0/2/4 will be operated while the GATE signal is “H”.
Bit 5	**POE0: Output enable bit**	• This bit is used to select either the pulse output of PPG0/2/4 or general-purpose port.
Bit 4	**PIE0: Interrupt request enable bit for PPG0/2/4**	• This bit is used to enable an interrupt request for PPG0/2/4. • While '1' is written to this bit, PPG0/2/4 is generated an interrupt when the interrupt flag (PUF0) is set.
Bit 3	**PUF0: Interrupt request flag bit for PPG0/2/4**	• This bit is used as an interrupt request flag for PPG0/2/4 underflow. • In 8-bit PPG 2ch independent mode or 8-bit prescaler + 8-bit PPG mode, '1' is written to this bit when an underflow occurs as a result of the ch0/2/4 counter value becoming between 00H and FFH. • In 16-bit PPG 1ch mode, '1' is written to this bit when an underflow occurs as a result of the ch1/ch0, ch2/ch3 and ch4/ch5 counter value becoming between 0000H and FFFFH. • Writing “0” will clear this bit. Writing “1” has no effect. • In Read-Modify-Write operation, “1” is always read.
Bit 2	**POS1: Pulse polarity selection bit**	• ThIs bit is used to select the pulse polarity to the external pin for PPG1/3/5
Bit 1	**POS1: Pulse polarity selection bit**	• ThIs bit is used to select the pulse polarity to the external pin for PPG0/2/4.
Bit 0	**Reserved bit**	• Always setting this bit to “1” when setting PPGC0/2/4

12.3.4.3 PPG Clock Control Register (PCS01/23/45)

PPG clock control registers (PCS01/23/45) are used to control the operating clock frequency for PPG0~5.

■ PPG clock control register (PCS01/23/45)

Address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Initial value
ch0,1: 00003E$_H$ ch2,3: 000046$_H$ ch4,5: 00004E$_H$	PC12	PC11	PC10	PC02	PC01	PC00	—	—	000000XX$_B$
	R/W	R/W	R/W	R/W	R/W	R/W	—	—	

PC02	PC01	PC00	Operation clock for ch0/2/4
0	0	0	ϕ (62.5 ns, ϕ=16MHz)
0	0	1	ϕ/2 (125 ns, ϕ=16MHz)
0	1	0	ϕ/4 (250 ns, ϕ=16MHz)
0	1	1	ϕ/8 (500 ns, ϕ=16MHz)
1	0	0	ϕ/16 (1 µs, ϕ=16MHz)
1	0	1	ϕ/32 (2 µs, ϕ=16MHz)
1	1	0	ϕ/64 (4 µs, ϕ=16MHz)
1	1	1	Input clock from timebase trimmer

ϕ: Machine clock

PC12	PC11	PC10	Operation clock for ch1/3/5
0	0	0	ϕ (62.5 ns, ϕ=16MHz)
0	0	1	ϕ/2 (125 ns, ϕ=16MHz)
0	1	0	ϕ/4 (250 ns, ϕ=16MHz)
0	1	1	ϕ/8 (500 ns, ϕ=16MHz)
1	0	0	ϕ/16 (1 µs, ϕ=16MHz)
1	0	1	ϕ/32 (2 µs, ϕ=16MHz)
1	1	0	ϕ/64 (4 µs, ϕ=16MHz)
1	1	1	Input clock from timebase trimmer

ϕ: Machine clock

X : Indeterminate
R/W : Read and write
: Initial value
— : not used

Figure 12.3.4.3-1 PPG0/1/2/3/4/5 Clock Control Register (PCD01/23/45)

Table 12.3.4.3-1 PPG0/1/2/3/4/5 Clock Control Register (PPG0/2/4)

Bit		Function
Bit 7 Bit 6 Bit 5	**PC12~0: Operation clock control bit for PPG1/3/5**	• These bits select the PPG1/3/5 down counter operation clock. **Note:** In 8 + 8-bit PPG mode or in 16-bit PPG mode, PPG1/3/5 operates in response to a counter clock from PPG0/2/4. Therefore, the PCS12~0 bits are invalid.
Bit 4 Bit 3 Bit 2	**PC02~0: Operation clock control bit for PPG0/2/4**	• These bits select the PPG0/2/4 down counter operating clock.
Bit 1 Bit 0	**Unused bit**	• The read values are indeterminate. • Writing to these bits have no effect on the operation.

12.3.5 Registers of Waveform Generator

This section describe registers of waveform generator and their details. Those registers are 8-bit reload register x 3, 8-bit timer control register x 3, waveform control register x 1.

■ Waveform Generator Registers

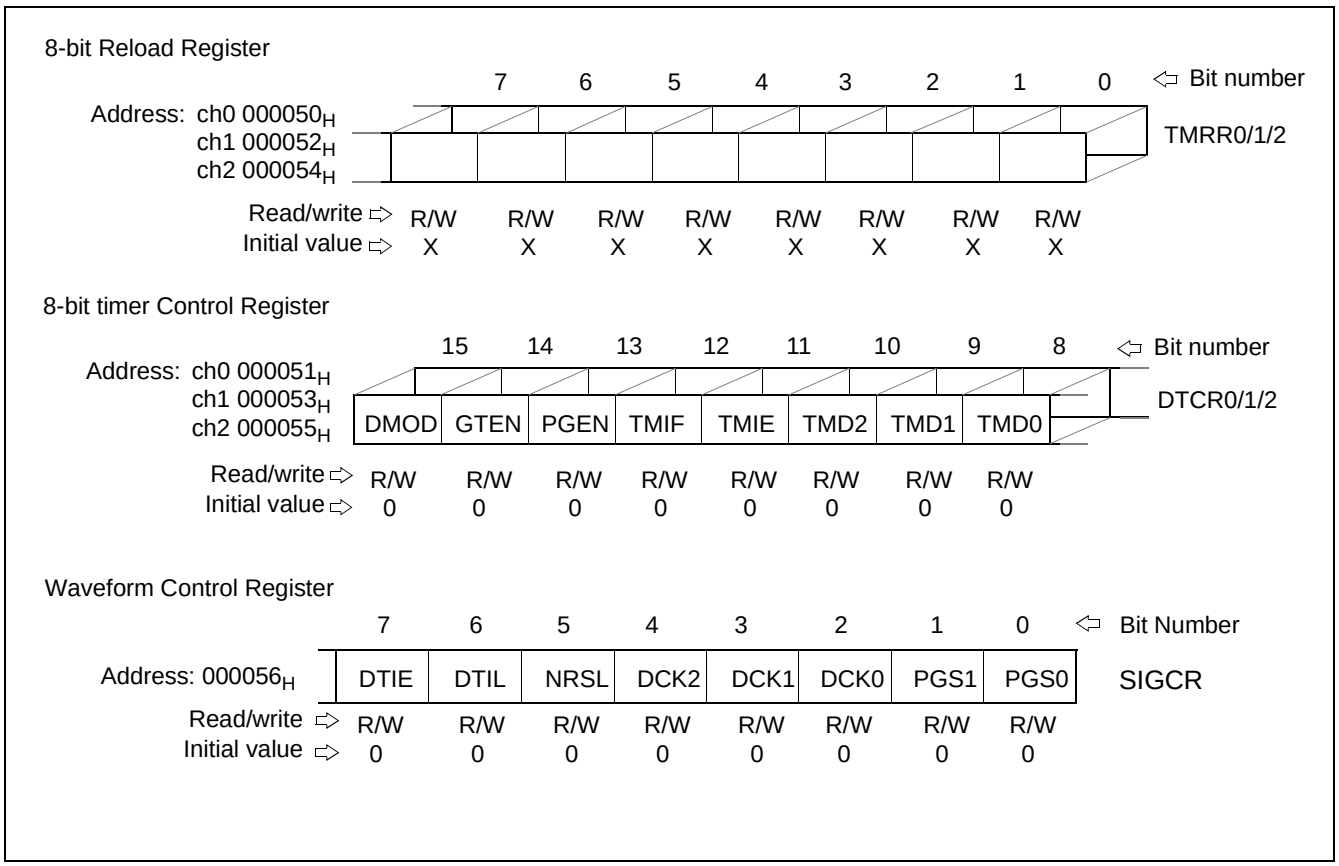


Figure 12.3.5-1 Registers of Waveform Generator

12.3.5.1 8-bit Reload Register (TMRR0/1/2)

8-bit reload registers hold the compare value of 8-bit timers.

■ **8-bit reload registers (TMRR0/12)**

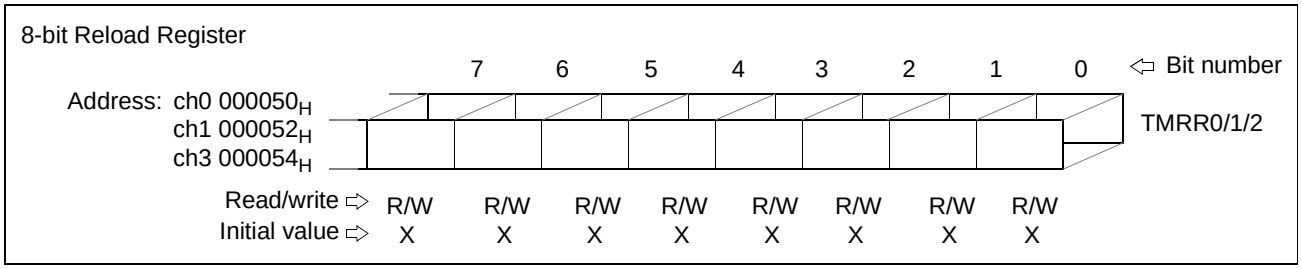


Figure 12.3.5.1-1 8-bit Reload Registers (TMRR0/1/2)

These registers is used to store the comparison value of 8-bit timers. These registers will be reloaded when the 8-bit timer is started to operate. Therefore, if the value is re-written into those registers during timer operation, those value will be valid at the next timer initiation/operation.

In dead-time timer mode, these registers are used to set the non-overlap time.

- Non-overlap time = (set value +1) x selected clock.

Note: The value of "00_H" cannot be set.

In timer mode, these registers are used to set the GATE time for PPG timer operation.

- GATE time = (set value +1) x selected clock.

Note: The value of "00_H" cannot be set.

12.3.5.2 8-bit Timer Control Register (DTCR0/1/2)

8-bit timer control registers (DTCR0/1/2) are used to control the operation mode, interrupt request enable, interrupt request flag, PPG output enable, GATE signal enable and output level polarity for the waveform generator.

■ **8-bit timer control register (DTCR0/1/2)**

Address	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Initial value
ch0: 000051H ch1: 000053H ch2: 000055H	DMOD	GTEN	PGEN	TMIF	TMIE	TMD2	TMD1	TMD0	00000000B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

TMD2	TMD1	TMD0	Operation mode bit
0	0	0	Waveform generator is stopped. 8-bit timer can be used as reload timer.
0	0	1	PPG timer output pulse while RT signal is “H”. The PPG timer that outputs pulse must be set in advance in the waveform control register.
0	1	0	The rising edge of each RT signal will trigger 8-bit timer to start. PPG will run until the 8-bit timer stopped. (Timer mode)
1	0	0	Generate non-overlap signal by RT signal. (Dead-time timer mode)
1	1	1	RT signal acts as a GATE signal for PPG timers. PPG generates non-overlap signal. (Dead-time timer mode)

TMIE	Interrupt request enable bit
0	Disable an interrupt when the 8-bit timer underflow
1	Enable an interrupt when the 8-bit timer underflow

TMIF	Interrupt request flag bit	
	Read	Write
0	No counter underflow detected	Clear this bit
1	Counter underflow detected	No effect

PGEN	PPG output enable bit
0	Disable PPG output
1	Enable PPG output

GTEN	GATE signal enable bit
0	Disable GATE signal output (Asynchronous mode)
1	Enable GATE signal output (Synchronous mode)

DMOD	Output polarity control bit
0	Positive level output
1	Negative level output

R/W : Read and Write
: Initial value

Figure 12.3.5.2-1 8-bit Timer Control Register (DTCR0/1/2)

Table 12.3.5.2-1 8-bit Timer Control Registers (DTCR0/1/2) Bit

Bit		Function
Bit 15	**DMOD: Output polarity control bit**	• This bit is used to set the output polarity when using 8-bit timer as dead-time timer. • By setting this bit, it is possible to invert the output polarity. • If 8-bit timer is not set for dead-time timer operation (Bit11:TMD2=0), setting this bit has no effect.
Bit 14	**GTEN: GATE signal enable bit**	• This bits is used to enable the GATE signal output for PPG timer operation.
Bit 13	**PGEN: PPG output enable bit**	• This bit is used to enable the output of PPG timers to RTO0~5 pins.
Bit 12	**TMIF: Interrupt request flag bit**	• This bit is used as an interrupt request flag for 8-bit timers. • This bit will be set to “1” when 8-bit timer is underflow. • Writing “0” will clear this bit. Writing “1” has no effect. • In Read-Modify-Write operation, “1” is always read.
Bit 11	**TMIE: Interrupt request enable bit**	• This bit is used as the start bit and the interrupt enable bit for the 8-bit timer. • Writing “1” to this bit will start the operation for the 8-bit timer. • When this bit is “1” and the interrupt flag bit (Bit12:TIMF) is “1”, an interrupt is generated. **Note:** Bit2~0:TMD2~0 is “000B” is the condition for using the 8-bit timer as reload timer.
Bit 10 **Bit 9** **Bit 8**	**TMD2~0: Operation mode bit**	• These bit are used to select the operation mode of the waveform generator. **Note:** In PPG timers, the channel is set by waveform control register. **Note:** Setting TMD2:0 other than ‘000’, ‘0001’, ‘010’, ‘100’ and ‘111’ is prohibited. Other setting will cause an error. When operating in dead time timer mode, the operation mode of RTO1/3/5 must set by 2ch mode.

12.3.5.3 Waveform Control Register (SIGCR)

Waveform control register is used to control how the PPG0~5 output to RTO0~5, operating clock frequencies, noise cancelling function enable, DTTI input enable and DTTI input polarity.

■ Waveform control register (SICGCR)

Address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	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	Operation mode	PPG0~5 output thru RTO0~5 table					
			RTO0	RTO1	RTO2	RTO3	RTO4	RTO5
0	0	8bit x 6 ch	PPG0	PPG1	PPG2	PPG3	PPG4	PPG5
0	1	8-bit x 3 ch	PPG1	PPG1	PPG3	PPG3	PPG5	PPG5
1	0	8-bit x 2 ch	PPG1	PPG3	PPG1	PPG3	PPG1	PPG3
1	1	8-bit x 1 ch	PPG1	PPG1	PPG1	PPG1	PPG1	PPG1

DCK2	DCK1	DCK0	Operating clock selection bit
0	0	0	ϕ (62.5 ns, ϕ=16MHz)
0	0	1	ϕ/2 (125 ns, ϕ=16MHz)
0	1	0	ϕ/4 (250 ns, ϕ=16MHz)
0	1	1	ϕ/8 (500 ns, ϕ=16MHz)
1	0	0	ϕ/16 (1 µs, ϕ=16MHz)
1	0	1	ϕ/32 (2 µs, ϕ=16MHz)
1	1	0	ϕ/64 (4 µs, ϕ=16MHz)
1	1	1	Prohibited

ϕ: Machine clock

NRSL	Noise cancelling function selection bit
0	DTTI input is not go thru the noise cancelling circuit.
1	DTTI input is not go thru the noise cancelling circuit.

DTIL	DTTI input polarity selection bit
0	“0” input
1	“1” input

DTIE	DTTI input enable bit
0	Disable DTTI input
1	Enable DTTI input

R/W : Read and Write

: Initial value

Figure 12.3.5.3-1 Waveform Control Register (SIGCR)

Table 12.3.5.3-1 Waveform Control Register (SIGCR)

Bit		Function
Bit 7	**DTIE: DTTI input enable bit**	• This bit is used to enable to receive DTTI input to control the output level of RTO0~5 pin.
Bit 6	**DTIL: DTTI input polarity selection bit**	• This bit is used to select the level polarity of DTTI input to control the output level of RTO0~5 pin.
Bit 5	**NRSL: Noise cancelling function selection bit**	• This bit is used to select the noise cancelling function. • Noise cancelling circuit will receive DTTI input signal when the valid level is held until the counter overflows. The counter is 2-bit counter which is operated by the valid level input. **Note:** To cancel the noise pulse width, it takes approximately 4 machine cycles. When the noise cancelling circuit is selected, the input will become invalid if the internal clock is stopped, for example in stop mode.
Bit 4 Bit 3 Bit 2	**DCK2~0: Operating clock Selection bit**	• These bits are used to select the operating clock for the 8-bit timer.
Bit 1 Bit 0	**PGS1, PGS0: PPG output to RTO selection bit**	• These bits are to select which the PPG timer is used to output pulses directly to RTO0~5. • When the 8-bit timer control register, bit 14:GTEN is set to “1”, GATE signal (synchronized mode) will be output to the corresponding PPG. **Note:** The operation mode in the table on the Figure 12.3.5.3-1 is written with PPG timer setting, so the corresponding PPG timer output and operation start/stop control is possible when setting 16-bit mode for the PPG timer. Also, PPG output is possible without using PPG timer GATE function (asynchronized mode), but PPG timer must be operating in software initiation.

12.4 Operation of Multi-Function Timer

This section describes the operation of the multifunctional timer unit.

■ Operation of Multifunctional Timers

● 16-bit free-running timer

The 16-bit free-running timer starts counting up from counter value "0000_H" after a reset has been completed. The counter value is used as the reference time for 16-bit output compare and 16-bit input capture.

● 16-bit output compare

The output compare unit is used to compare the value set in the specified compare register with the value of the 16-bit free-running timer. If a match is detected, the interrupt flag is set and the output level is inverted

● 16-bit input capture

The input capture unit is used to detect a specified valid edge. If a valid edge is detected, the interrupt flag is set and the value of 16-bit free-run timer is fetched and stored into the capture register.

● 8/16-bit PPG timer

This PPG timer has two channels of PPG units each of which is 8 bits long. The timer can operate in normal two-channel independent mode. In addition, by connecting the two channels, it can also operate in two other modes, i.e., 8 + 8-bit PPG mode and 16-bit PPG mode.

● Waveform generator

The waveform generator can generate various timing waveforms using the real-time outputs (RT0 to RT5), 8/16-bit PPG timer, and 8-bit timer.

Memo

12.4.1 Operation of 16-bit Free-run Timer

The 16-bit free-running timer starts counting up from counter value "0000_H" after a reset has been completed. The counter value is used as the reference time for 16-bit output compare and 16-bit input capture.

■ 16-bit free-run timer operation

The counter value is cleared in the following conditions:

- When an overflow has occurred.
- When a match with output compare clear register is detected. (Mode setting is necessary.)
- When "1" is written to the CLR bit of the TCCS register during operation.
- When "0000_H" is written to the TCDT register during stop.
- Reset

An interrupt can be generated when an overflow occurs or when the counter is cleared due to a match with compare clear register. (Compare match interrupts can be used only in an appropriate mode.)

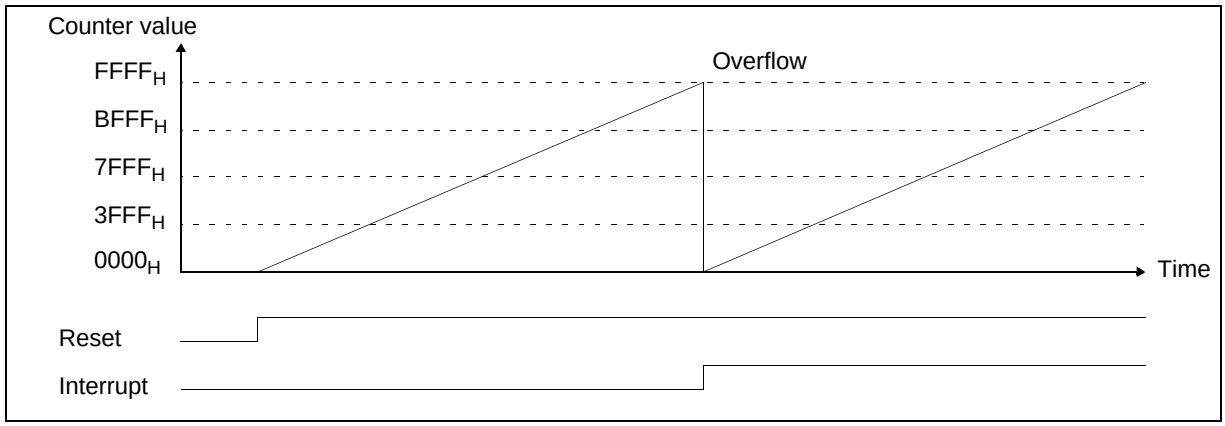


Figure 12.4.1-1 Clearing the counter by an overflow

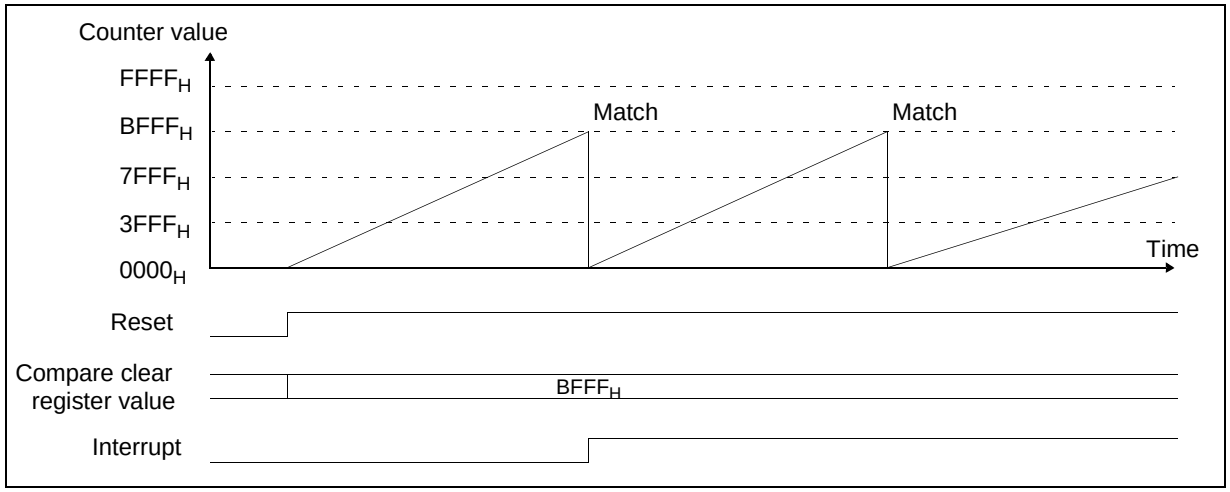


Figure 12.4.1-2 Clearing the counter upon a match with compare clear register

■ 16-bit free-run timer timing

The 16-bit free-run timer is incremented based on the input clock (internal or external clock). When external clock is selected, the 16-bit free-run timer counts up at a rising edge.

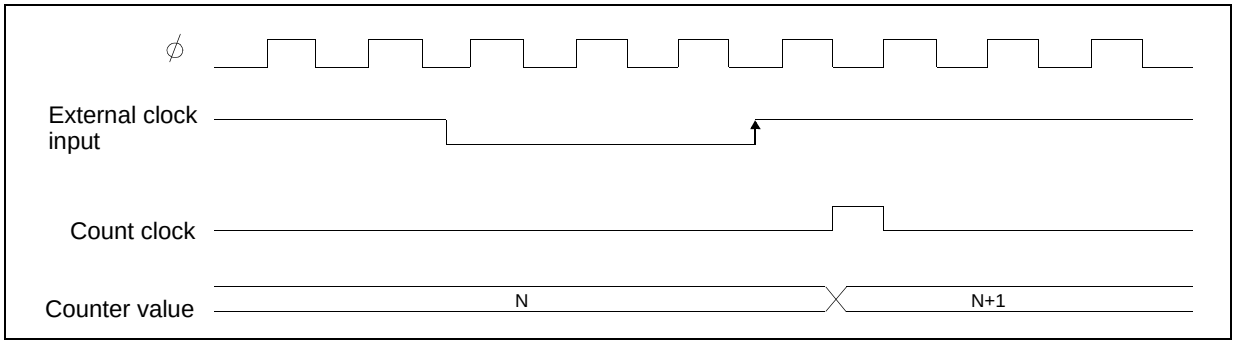


Figure 12.4.1-3 16-bit Free-run timer count timing

The counter can be cleared upon a reset, software clear, or a match with compare clear register. By a reset or software clear, the counter is immediately cleared. By a match with compare clear register, the counter is cleared in synchronization with the count timing.

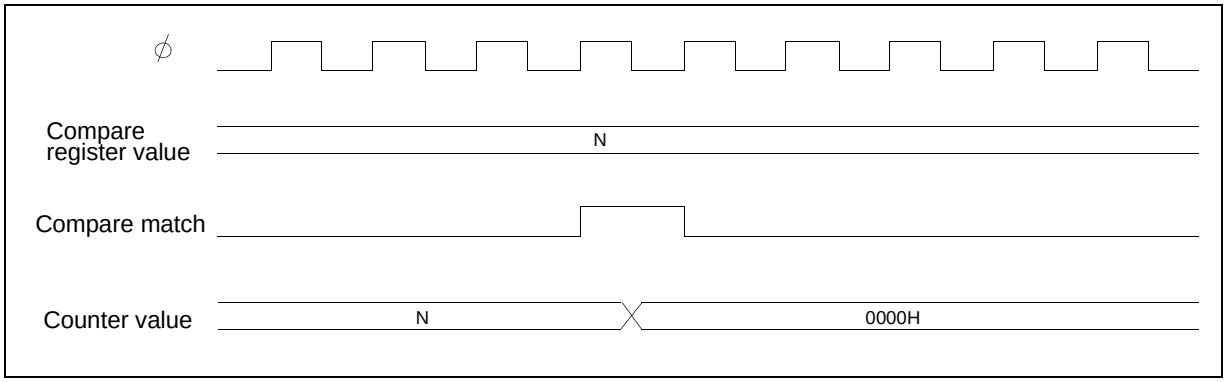


Figure 12.4.1-4 16-bit Free-run timer clear timing

12.4.2 Operation of 16-bit Output Compare

The output compare unit is used to compare the value set in the specified compare register with the value of the 16-bit free-running timer. If a match is detected, the interrupt flag is set and the output level is inverted.

■ **16-bit output compare operation**

a) Compare operation can be performed for individual channel (CMOD=0)

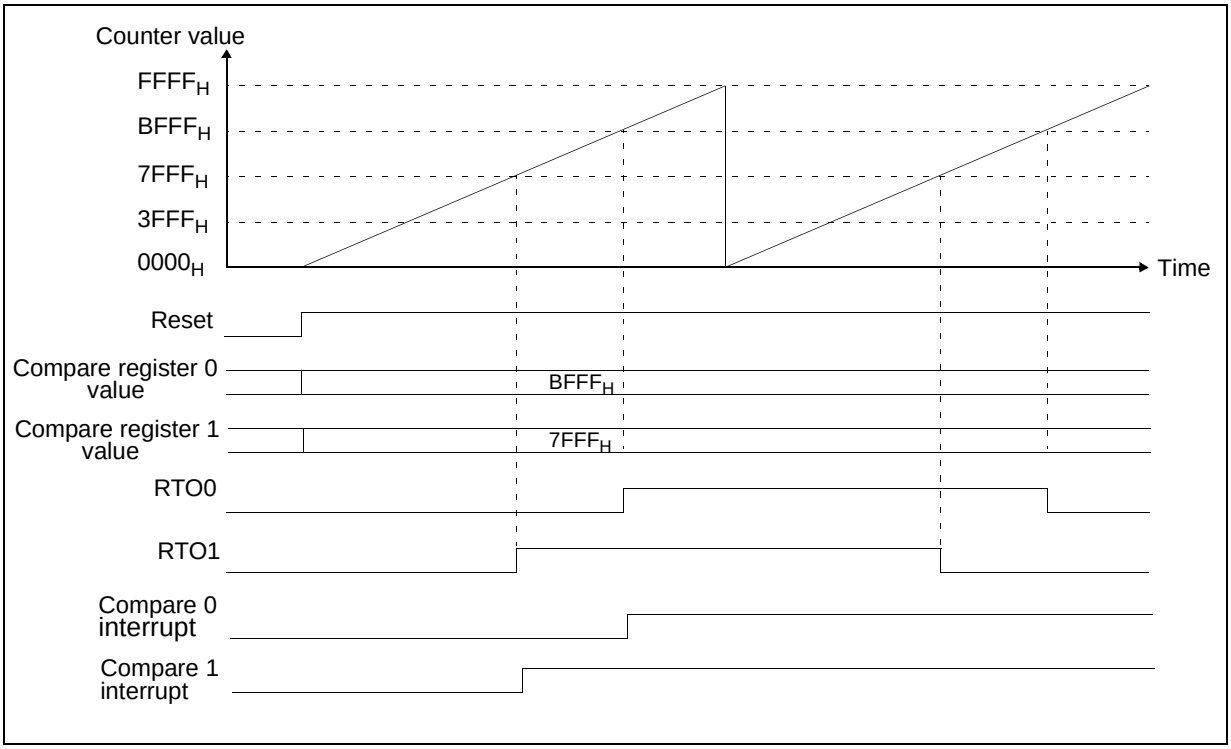


Figure 12.4.2-1 Sample output waveform when compare registers 0 and 1 are used individually when the initial output value is “0”.

b) Output level can be changed by using a pair of compare registers (CMOD=”1”)

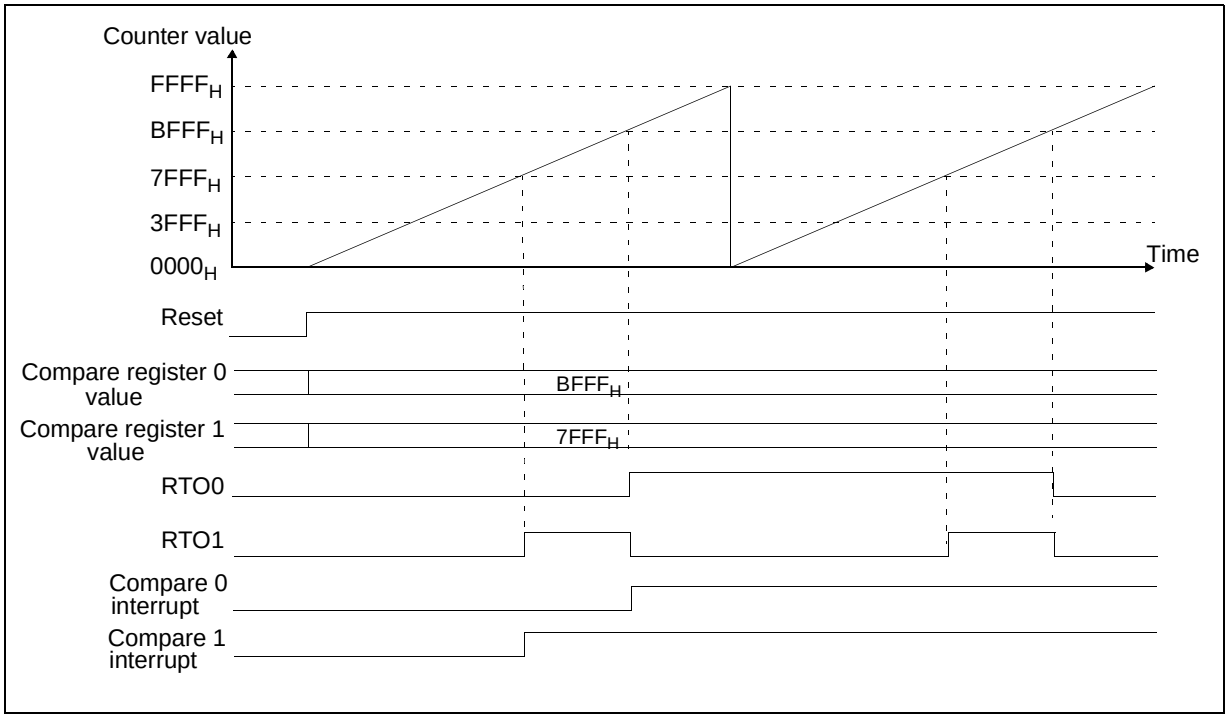


Figure 12.4.2-2 Sample output waveform when compare registers 0 and 1 are used in a pair when the initial output value is “0”.

■ 16-bit output compare timing

● Output levels can be changed by using two pairs of compare registers. (At CMOD = 1)

When the free-running timer matches the value set in the compare register, the output compare unit generates a compare match signal to invert the output and generate an interrupt. When a compare match occurs, the output is inverted in synchronization with the count timing of the counter. No comparison is performed with the counter value during replacement of the compare registers.By using 2 compare registers, output level can be changed (“1” when CMOD=”1”)

Note: When the compare register is updated, comparison with the counter value is not performed.

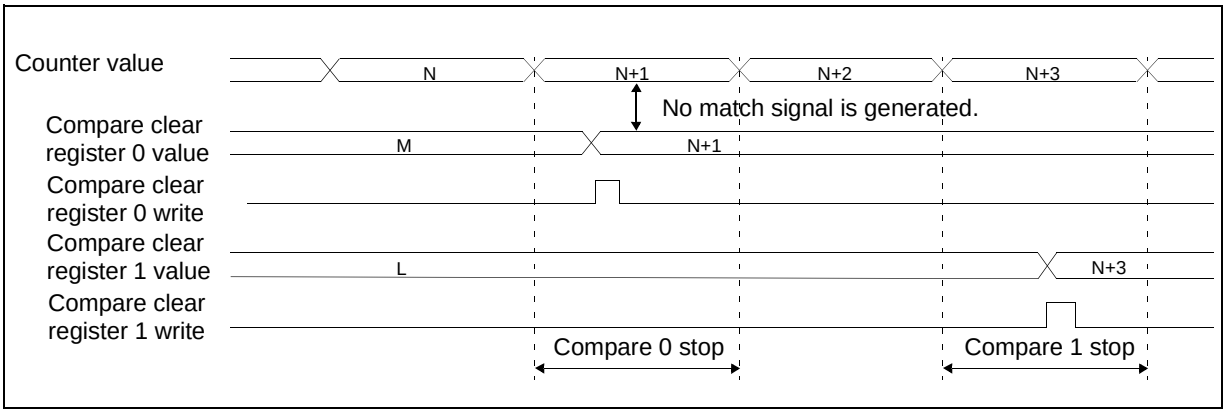


Figure 12.4.2-3 Compare operation upon update of compare registers

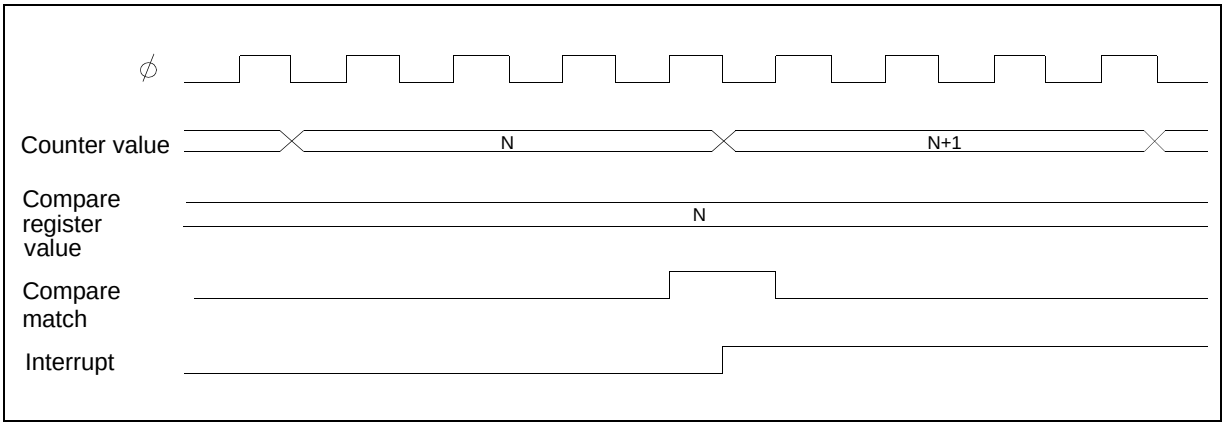


Figure 12.4.2-4 Compare interrupt timing

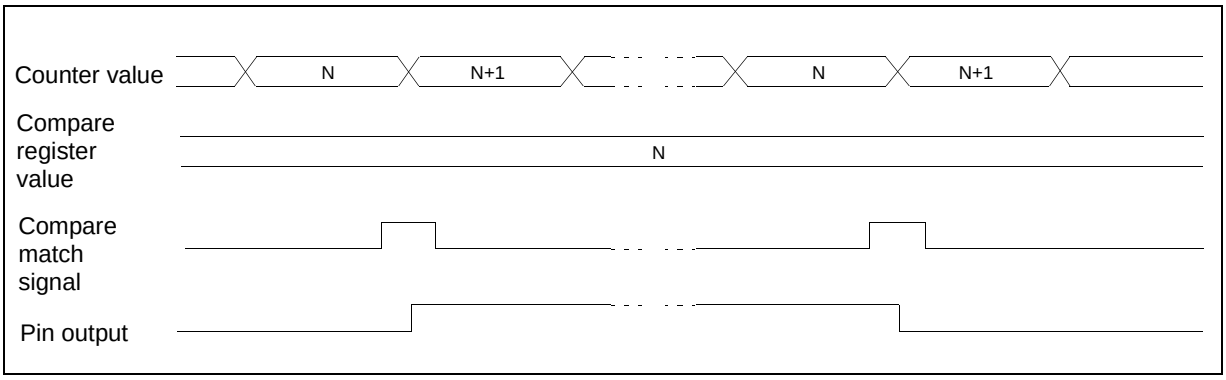


Figure 12.4.2-5 Output pin change timing

Memo

12.4.3 Operation of 16-bit Input Capture

The input capture unit is used to detect a specified valid edge. If a valid edge is detected, the interrupt flag is set and the value of 16-bit free-run timer is fetched and stored into the capture register.

■ **16-bit Input capture operation**

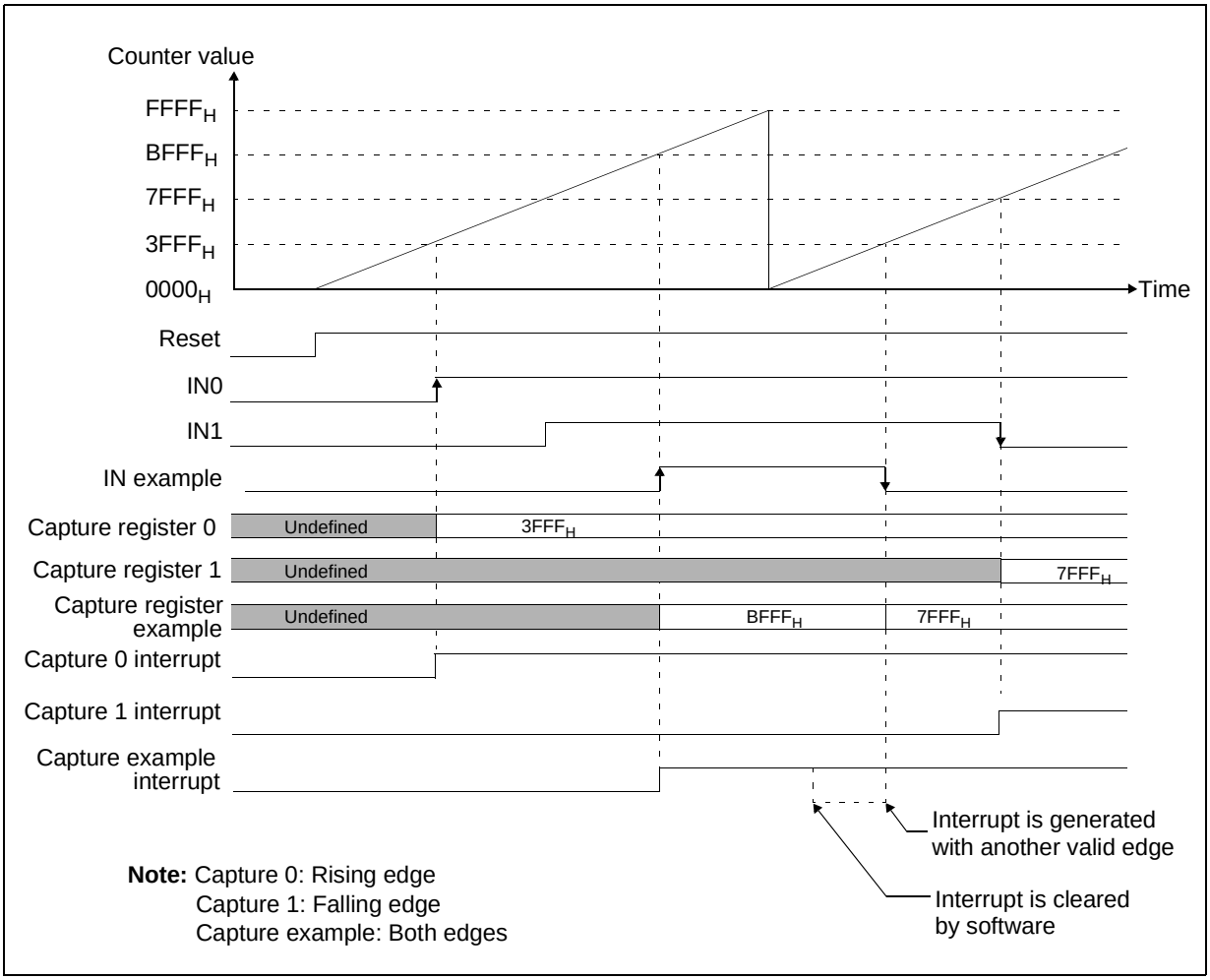


Figure 12.4.3-1 Sample input capture timing

■ **16-bit input capture input timing**

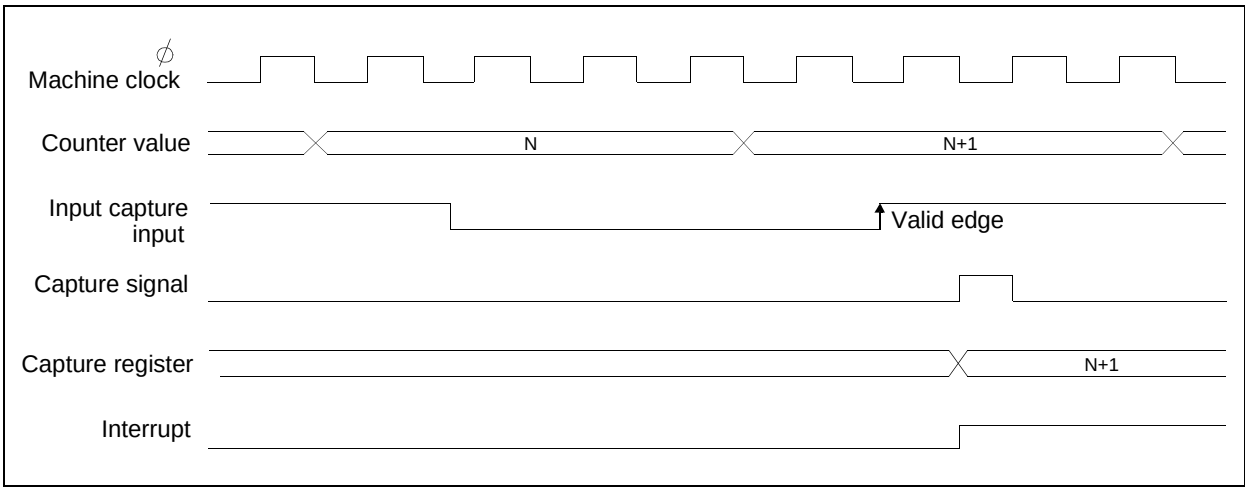


Figure 12.4.3-2 16-bit input capture timing for input signals

12.4.4 Operation of 8/16-bit PPG Timers

This section describe operation and timing of 8/16-bit PPG timers. The operation and timing of ch0/ch1 will be explained here. The operation and timing of ch2 and ch4 is the same as ch 0 and the operation and timing of ch 3 and ch 5 is the same as ch 1. Two channels of PPG time can be used in three modes: independent two-channel mode, 8-bit prescaler + 8-bit PPG mode, and single-channel 16-bit PPG mode.

Each of the 8-bit PPG units has two eight-bit reload registers. One reload register is for the L side (PRLL) and the other is for the H side (PRLH). The values written in these registers are reloaded into the 8-bit down counter (PCNT), from the L side and H side in turn. Thus, the values are decremented for each count clock, and the pin output (PPG) value is inverted upon a reload caused by a counter underflow. This operation results in L-wide or H-wide pulse outputs, corresponding to the reload register value. Further more the output polarity can be selected by software.

The operation is started and resumed by writing data in the corresponding register bit.

The Table 12.4.4-1 below lists the relationship between the reload operation and pulse outputs.

Table 12.4.4-1 Reload operation and pulse output

Reload operation	Pin output change	
	Positive polarity	Negative Polarity
PRLH → PCNT	PPG0/1 [0 → 1] ↑ Rise	PPP0/1 [1 → 0] ↓ Fall
PRLL → PCNT	PPP0/1 [1 → 0] ↓ Fall	PPG0/1 [0 → 1] ↑ Rise

When 1 is set in bit 4 (PIE0) of PPGC0 or in bit 12 (PIE1) of PPGC1, an interrupt request is output upon an underflow from "00_H" to "FF_H" (or underflow from "0000_H" to "$FFFF_H$" in 16-bit PPG mode) of each counter.

■ Operation mode

This block can be used in three modes: independent two-channel mode, 8-bit prescaler + 8-bit PPG mode, and single-channel 16-bit PPG mode.

- **8-bit 2ch independent mode**
  In this mode, the two channels of 8-bit PPG units operate independently. The PPG0 pin is connected to the ch0 PPG output and the PPG1 pin is connected to the ch1 PPG output.
- **8 + 8-bit PPG mode**
  In this mode, ch0 is used as an 8-bit prescaler while the count in ch1 is based on underflow outputs from ch0. Thus, 8-bit PPG waveforms can be output at any cycles. The PPG0 is connected to the ch0 prescaler output and the PPG1 pin is connected to the ch1 PPG output.
- **16-bit PPG mode**
  In this mode, ch0 and ch1 are connected and used as a single 16-bit PPG. The PPG0 and PPG1 pins are connected to the 16-bit PPG output.

■ PPG output operation

Writing ’1’ to bit7 (PEN0) of PPGC0 register will activate ch0 PPG timer to start counting when bit6 (SST0) of PPGC0 register is “0” (Software activation). Alternately, GATE signal becoming “H” will activate ch0 PPG timer to start counting when bit6 (SST0) of PPGC0 register is “1” (Hardware activation).

Similarly, writing '1' to bit15 (PEN1) of PPGC1 register will activate ch1 PPG timer to start counting when bit14 (SST1) of PPGC1 register is “0” (Software activation). Alternately, GATE signal becoming “H” will activate ch1 PPG timer to start counting when bit14 (SST1) of PPGC1 register is “1” (Hardware activation).

Once the operation has started, counting is terminated by writing '0' to bit 7 (PEN0) of PPGC0 or in bit 15 (PEN1) of PPGC1, or when the GATE signal changes to “L”. Once the counting is terminated, the pulse output is maintained at the “L” level when it is positive polarity output. (the pulse output is maintained at the “H” level when it is negative polarity output.)

In 8 + 8-bit PPG mode, do not set ch1 to be in operation while ch0 operation is stopped.

For software activation of 16-bit PPG mode, ensure to set bit 7 (PEN0) of PPGC0 register and bit 15 (PEN1) of PPGC1 register simultaneously to start or stop the PPG timer.

The Figure 12.4.4-1 below is a diagram of PPG output operation. During PPG operation, a pulse wave is continuously output at a frequency and duty ratio (the ratio of the H-level period of the pulse wave to the L-level period). PPG continues operation until stop is specified explicitly.

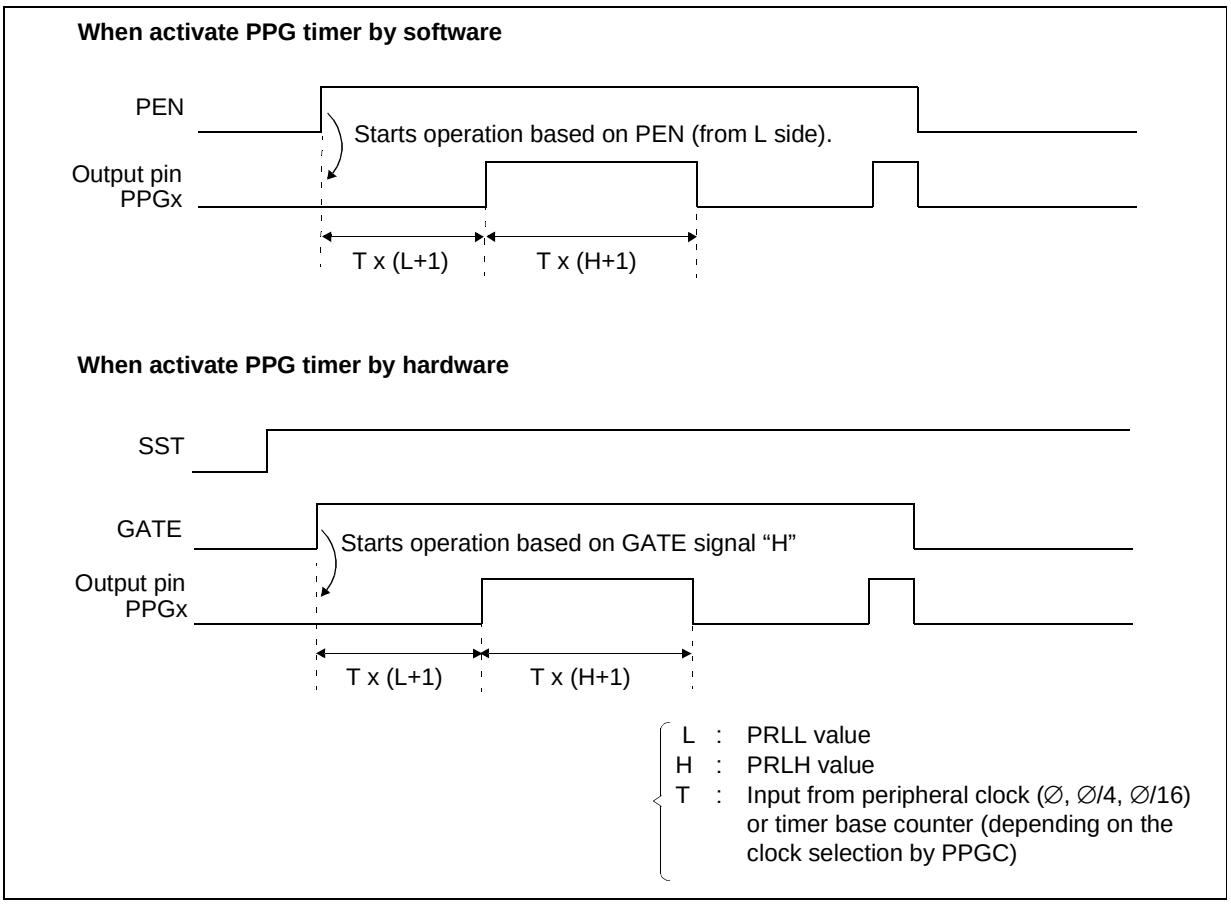


Figure 12.4.4-1 PPG output operation, output waveform

■ Reload value and pulse width relation

The pulse width of the output is determined by multiplying “the reload register value + 1” by the count clock cycle. Note that when the reload register value is “00_H“ during 8-bit PPG operation or “0000_H“ during 16-bit PPG operation, the pulse width is equivalent to one count clock cycle. In addition, note that when the reload register value is “FF_H“ during 8-PPG operation, the pulse width is equivalent to 256 count clock cycles. When the reload register value is “$FFFF_H$“ during 16-bit PPG operation, the pulse width is equivalent to 65,536 count clock cycles. The formula of pulse width calculation is given below.

$$P_L = T \times (L+1)$$
$$P_H = T \times (H+1)$$

L : PRLL value
H : value
T : Input clock cycle
P_L : High pulse width
P_H : Low pulse width

Note: The above formula is for the positive output polarity. The P_L and P_H setting will be the other way around when negative polarity is used.

■ Count clock selection

The count clock used for the operation of this block is supplied from a machine clock or time base timer. The count clock can be selected from 8 types.

The count clock for ch 0 PPG timer is selected by setting bit 4 to 2 (PC02 to 0) of the PCS01 register and the count clock for ch1 PPG timer is selected by setting bit 7 to 5 (PC12 to 0) of the PCS01 register.

The clock is selected from 1/64 to 1 times of the machine clock and time base timer input.

However, in 8 + 8-bit PPG mode or 16-bit PPG mode, the ch1 PPG timer is operated by receiving a count clock from ch0 PPG timer. Therefore bit 7 to 5 (PC12 to 0) of the PCS01 register are invalid.

Note: When the time base counter input is used, the first count cycle after a trigger or a stop may be shifted. The cycle may also be shifted if the time base counter is cleared during operation of this module.
In 8 + 8-bit PPG mode, if ch1 is activated while ch0 is in operation and ch1 is stopped, the first count cycle may be shifted.

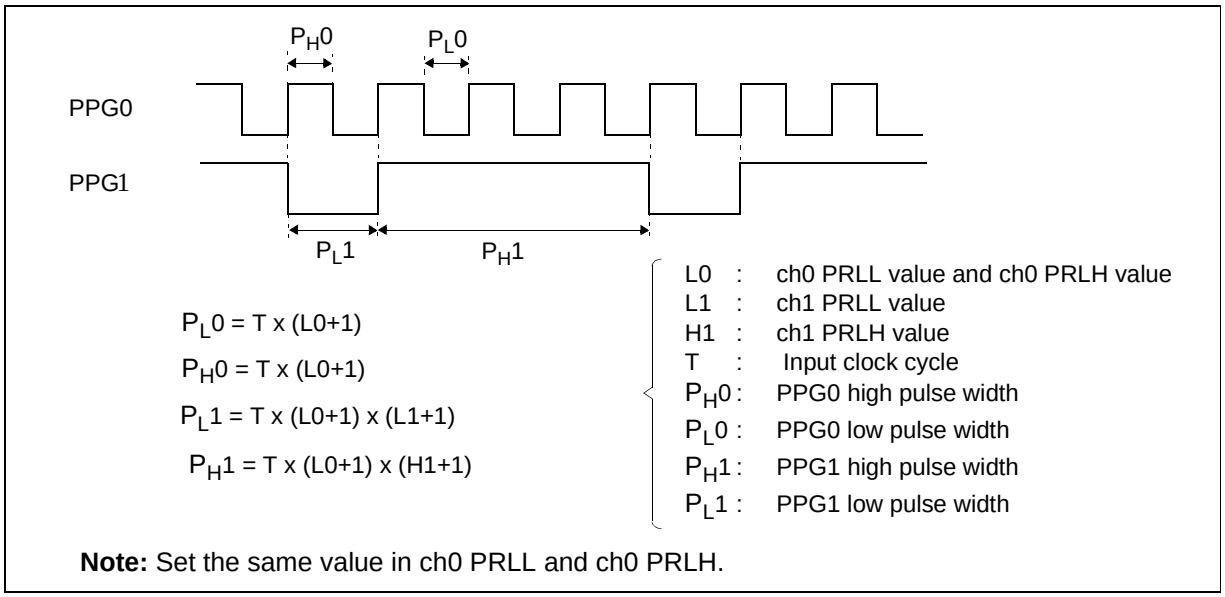
■ **Pulse pin output control**

The pulses generated by the operation of the 8/16-bit PPG timer can be output to external pins PPG0 and PPG1.

The external pin output enable can be done by setting bit 5 (POE0) of the PPGC0 register (PPG0 control register) for the PPG0 pin and setting bit 13 (POE1) of the PPGC1 register (PPG1 control register) for the PPG1 pin. When “0” is written to these bits (initial value), the pulses are not output to the corresponding external pins; the pins will work as general-purpose ports.

In 16-bit PPG mode, the same waveform is output from PPG0 and PPG1. Thus, the same output can be obtained by enabling any one external pin.

In 8 + 8-bit PPG mode, the 8-bit prescaler (Ch0 PPG timer) toggle output waveform is output to PPG0, while the 8-bit PPG waveform (Ch1 PPG timer) is output to PPG1. The Figure 12.4.4-2 below is a diagram of output waveforms in this mode.



Note: Set the same value in ch0 PRLL and ch0 PRLH.

Figure 12.4.4-2 8+8 PPG output operation waveform

■ Interrupts

An interrupt request is generated when the reload value is counted out and a underflow occurs.

In 8-bit PPG 2ch independent mode or 8 + 8-bit PPG mode, an interrupt request is generated by a underflow in each counter. In 16-bit PPG mode, PUF0 and PUF1 are set simultaneously by a underflow in the 16-bit counter. Therefore, enable only PIE0 or PIE1 to unify the interrupt causes. In addition, simultaneously clear the interrupt causes for PUF0 and PUF1.

■ Reload register write timing

In any operation mode other than 16-bit PPG mode, it is recommended to use a word transfer instruction to write data in reload registers PRLL and PRLH. If two byte transfer instructions are used to write a data item to these registers, a pulse of unexpected width may be output depending on the timing.

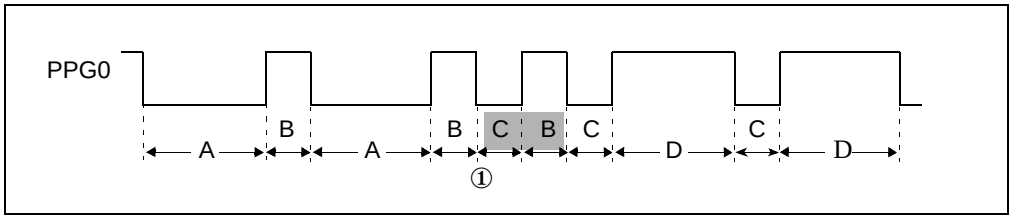


Figure 12.4.4-3 Write timing chart

Assume that PRLL is updated from A to C before point ① in the time chart above, and PRLH is updated from B to D after point ①. Since the PRL values at point ① are PRLL=C and PRLH=B, a pulse of L side count value C and H side count value B is output only once.

Similarly, to write data in PRL of ch0 and ch1 in 16-bit PPG mode, use a long word transfer instruction, or use word transfer instructions in the order of ch0 and then ch1. In this mode, the data is only temporarily written to ch0 PRL. Then, the data is actually written into ch0 PRL when the ch1 PRL is written to.

In any operation mode other than 16-bit PPG mode, ch0 and ch1 PRL are written independently.

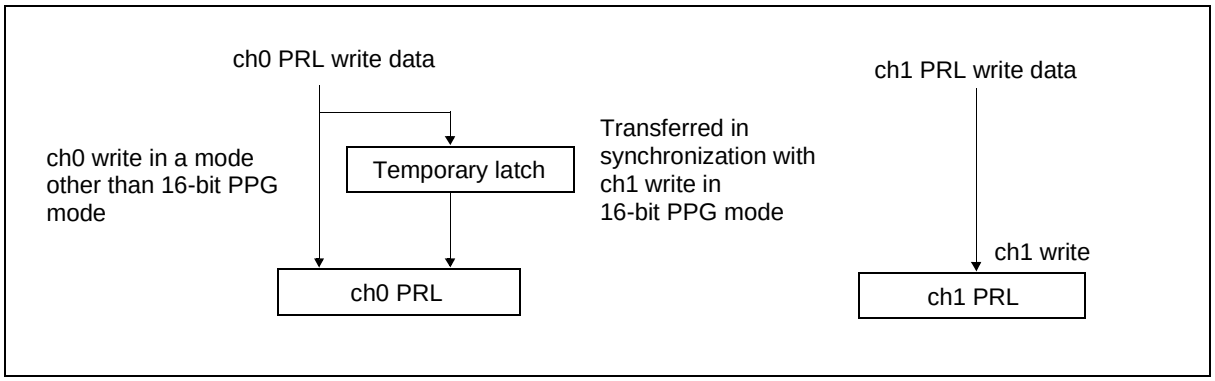


Figure 12.4.4-4 PRL write operation block diagram

Memo

12.4.5 Operation of Waveform Generator

Waveform generator can produce various waveform such as dead-time, by using the realtime outputs (RT0~5), 8/16-bit PPG timers and 8-bit timers.

■ Waveform generator

The waveform generator can generate several timing waveform as follows:

- By using 8-bit timer as dead-time timer, non-overlap signal of RT1, 3, 5 inverting signal can be generated. It is possible to make non-overlap signal of the selected PPG timer clock inverted signal only when RT1, 3, 5 is at the “H” level.(dead-time generation).
- With RT0~5 rising edge, the 8-bit timer is started and the selected PPG timer keeps outputting until when the value of the 8-bit timer is matched with the value of TMRR register (8-bit reload register).

Figure 12.4.5-1 is the block diagram of the waveform generator.

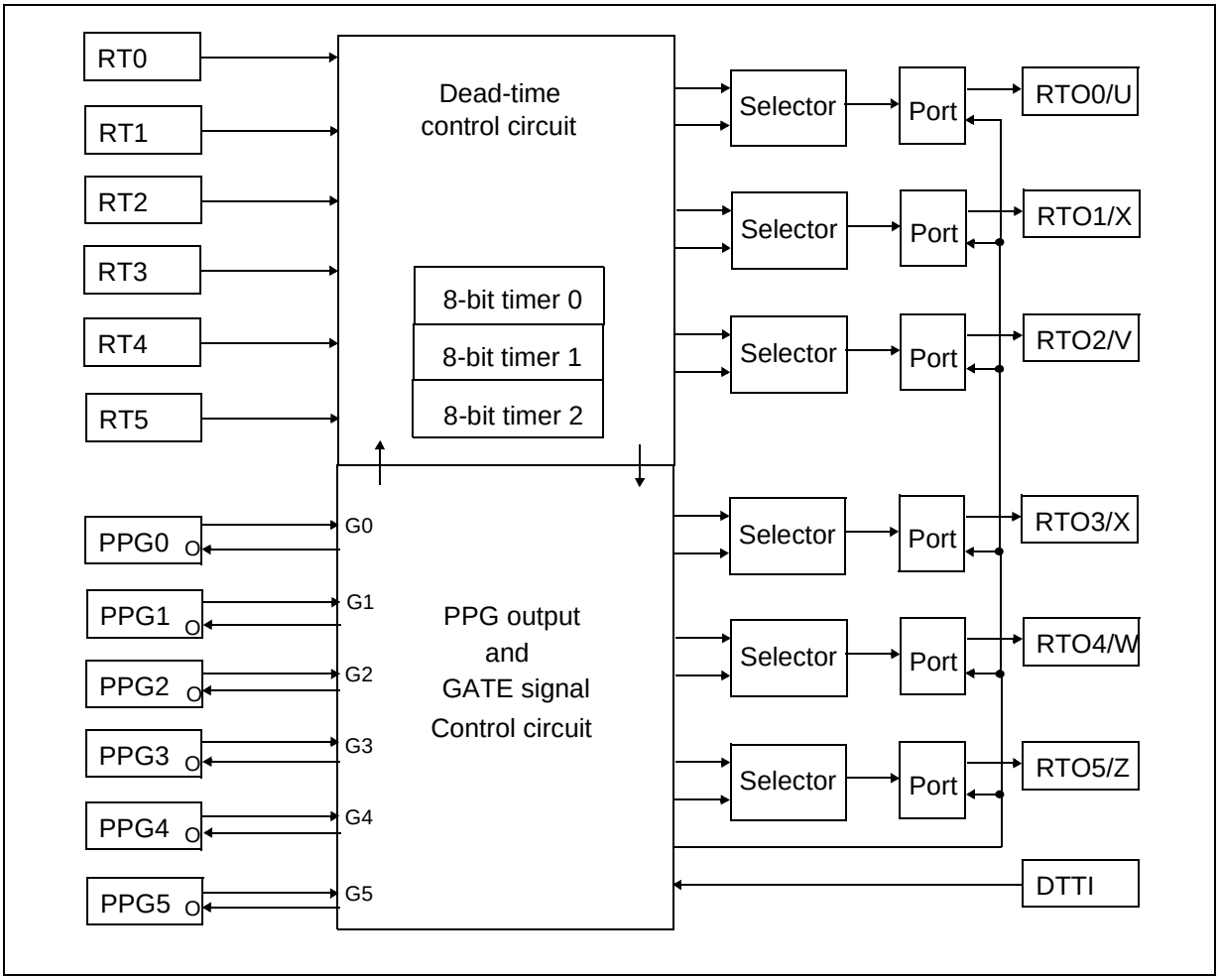


Figure 12.4.5-1 Waveform Generator

Memo

12.4.5.1 Operation of Dead-timer Control Circuit

The dead-time control circuit will input the realtime output (RT1/3/5) and the selected PPG timer pulse output, and output non-overlap signals (inverted signals) to external pins.

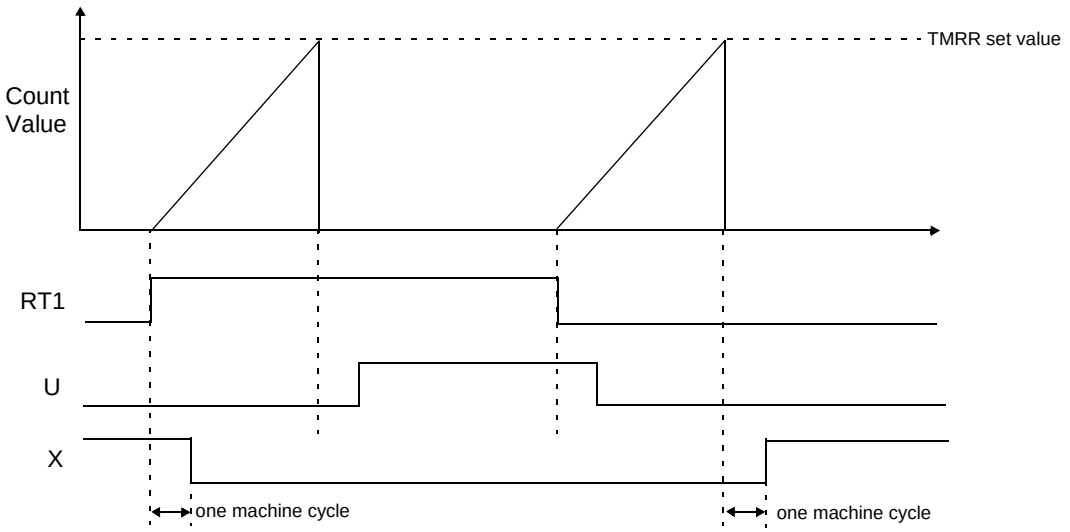
■ Making non-overlap signals by using RT1/3/5

When selecting non-overlap signal for a active level “0” (positive polarity) in DTCR:bit 15 (DMOD), a delay corresponding to the non-overlap time set in the TMRR register (8-bit reload register) is applied. The delay is applied at a rising edge of RT1/3/5 or its falling edge. If RT1/3/5 pulse width is smaller than the set non-overlap time, the 8-bit timer will restart counting from “00_H” at the nest RT’s edge.

Setting up registers:

- TCDT : 0000_H
- TCCS : ------XXXX0X0XXX$_B$
- OCCP0~5 : $XXXX_H$ (Compare value)
- TMRR0~2 : XXXXXXXXX$_B$ (non-overlap timing setting)
- SIGR : XXXXXX00$_B$ (DTTI input and 8-bit timer count clock setting)
- CPCLR : $XXXX_H$ (Cycle setting)
- OSC0~5 : ---1XXXXXXXX--11$_B$
- DTCR0~2 : 00000100$_B$

Note: “X” must be set according to the operation.



Pin name	Output signal
U	Signal with delay is applied at RT1 rising edge
V	Signal with delay is applied at RT3 rising edge
W	Signal with delay is applied at RT5 rising edge
X	Signal with delay is applied at RT1 falling edge
Y	Signal with delay is applied at RT3 falling edge
Z	Signal with delay is applied at RT5 falling edge

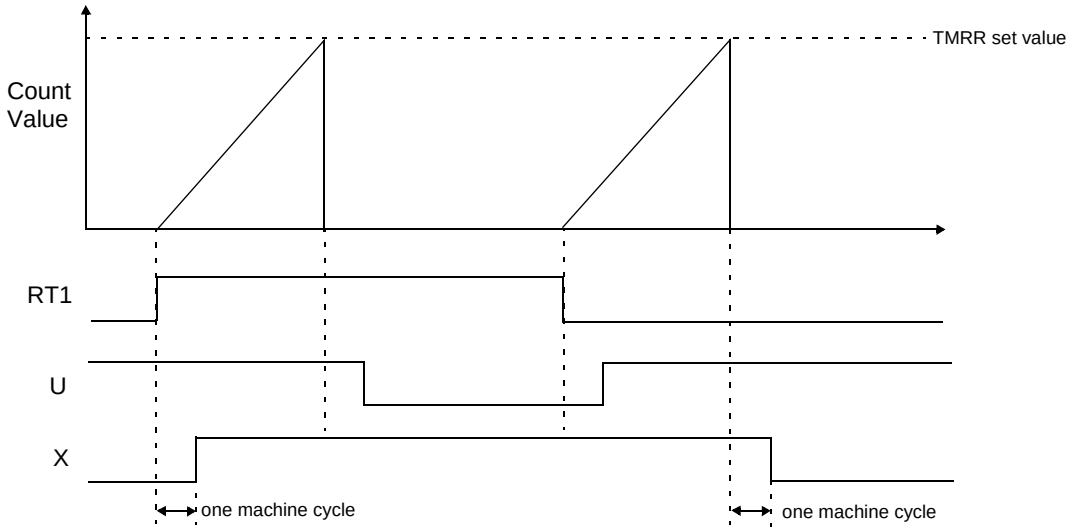
Figure 12.4.5.1-1 Positive Polarity Non-overlap Signal Generation by RT1/3/5

When selecting non-overlap signal for a active level “1” (positive polarity) in DTCR:bit 15 (DMOD), a delay corresponding to the non-overlap time set in the TMRR register (8-bit reload register) is applied. The delay is applied at a rising edge of RT1/3/5 or its falling edge. If RT1/3/5 pulse width is smaller than the set non-overlap time, the 8-bit timer will restart counting from “00_H” at the nest RT’s edge.

Setting up registers:

- TCDT : 0000_H
- TCCS : ------XXXX0X0XXX$_B$
- OCCP0~5 : $XXXX_H$ (Compare value)
- TMRR0~2 : $XXXXXXXX_B$ (non-overlap timing setting)
- SIGR : $XXXXXX00_B$ (DTTI input and 8-bit timer count clock setting
- CPCLR : $XXXX_H$ (Cycle setting)
- OSC0~5 : ---1XXXXXXXX--11_B
- DTCR0~2 : 10000100_B

Note: “X” must be set according to the operation.



Pin name	Output signal
U	Signal with delay is applied at RT1 rising edge
V	Signal with delay is applied at RT3 rising edge
W	Signal with delay is applied at RT5 rising edge
X	Signal with delay is applied at RT1 falling edge
Y	Signal with delay is applied at RT3 falling edge
Z	Signal with delay is applied at RT5 falling edge

Figure 12.4.5.1-2 Negative Polarity Non-overlap Signal Generation by RT1/3/5

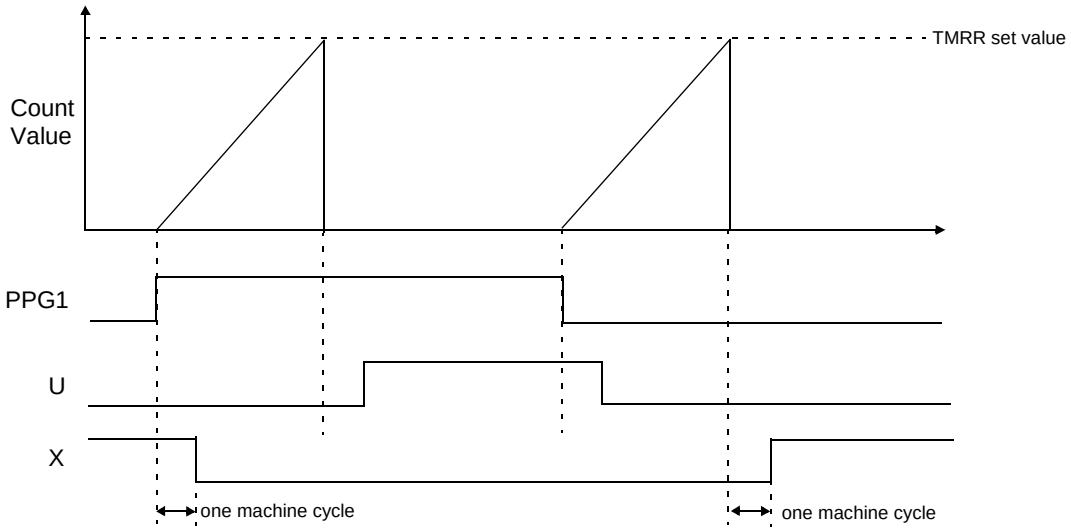
■ Making non-overlap signals by using PPG

When selecting non-overlap signal for a active level “0” (positive polarity) in DTCR:bit 15 (DMOD), a delay corresponding to the non-overlap time set in the TMRR register (8-bit reload register) is applied. The delay is applied at a rising edge of PPG timer pulse signal or its inverted signal. If PPG timer pulse width is smaller than the set non-overlap time, the 8-bit timer will start counting from “00_H” at the next edge of PPG pulse.

Setting up registers:

- TCDT : 0000_H
- TCCS : ------$XXXX0X0XXX_B$
- OCCP0~5 : $XXXX_H$ (Compare value)
- TMRR0~2 : $XXXXXXXXX_B$ (non-overlap timing setting)
- SIGR : $XXXXXX00_B$ (DTTI input and 8-bit timer count clock setting)
- CPCLR : $XXXX_H$ (Cycle setting)
- OSC0~5 : ---1XXXXXXXX--11_B
- DTCR0~2 : 01000111_B

Note: “X” must be set according to the operation.



Pin name	Output signal
U	Signal with delay is applied at PPG1 rising edge
V	Signal with delay is applied at PPG3 rising edge
W	Signal with delay is applied at PPG5 rising edge
X	Signal with delay is applied at PPG1 falling edge
Y	Signal with delay is applied at PPG3 falling edge
Z	Signal with delay is applied at PPG5 falling edge

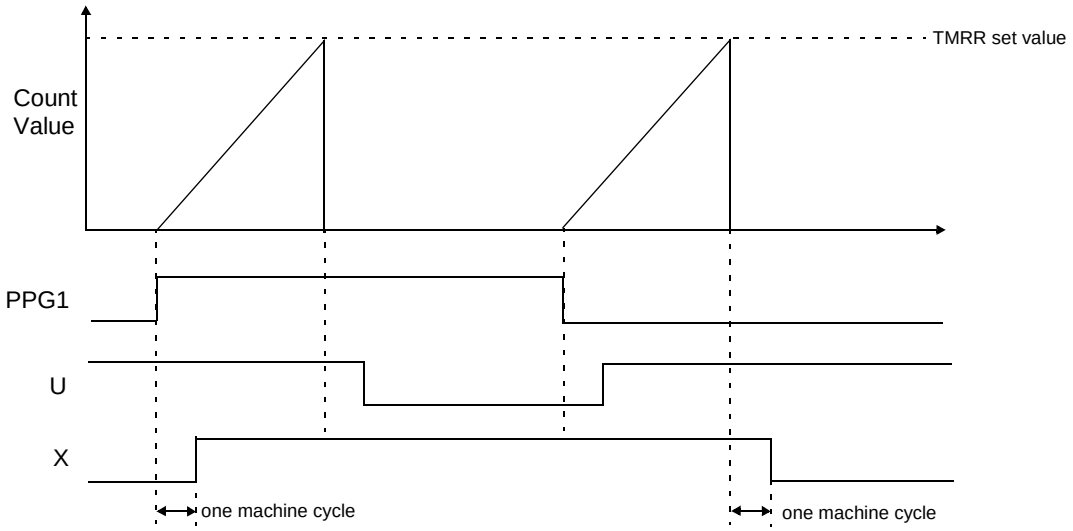
Figure 12.4.5.1-3 Positive Polarity Non-overlap Signal Generation by PPG timer

When selecting non-overlap signal for a active level “1” (positive polarity) in DTCR:bit 15 (DMOD), a delay corresponding to the non-overlap time set in the TMRR register (8-bit reload register) is applied. The delay is applied at a rising edge of PPG timer pulse signal or its inverted signal. If PPG timer pulse width is smaller than the set non-overlap time, the 8-bit timer will start counting from “00_H” at the next edge of PPG pulse.

Setting up registers:

- TCDT : 0000_H
- TCCS : ------$XXXX0X0XXX_B$
- OCCP0~5 : $XXXX_H$ (Compare value)
- TMRR0~2 : $XXXXXXXXX_B$ (non-overlap timing setting)
- SIGR : $XXXXXX00_B$ (DTTI input and 8-bit timer count clock setting)
- CPCLR : $XXXX_H$ (Cycle setting)
- OSC0~5 : ---1XXXXXXXX--11_B
- DTCR0~2 : 11000111_B

Note: “X” must be set according to the operation.



Pin name	Output signal
U	Signal with delay is applied at PPG1 rising edge
V	Signal with delay is applied at PPG3 rising edge
W	Signal with delay is applied at PPG5 rising edge
X	Signal with delay is applied at PPG1 falling edge
Y	Signal with delay is applied at PPG3 falling edge
Z	Signal with delay is applied at PPG5 falling edge

Figure 12.4.5.1-4 Negative Polarity Non-overlap Signal Generation by PPG timer

12.4.5.2 Operation of PPG Output and GATE Signal Control Circuit

The PPG output and GATE signal control circuit will output the selected PPG pulse to external pins when "1" is written to DCTR:bit 14, 13 (GTEN, PGEN). Furthermore, it output the GATE signal which controls PPG timer operation. It generated GATE signal by using realtime outputs or 8-bit timers and output to PPG timers.

■ **Generating a PPG output/GATE signal during each RT is at "H" level**

Setting up registers:

- TCDT : 0000_H
- TCCS : ------XXXX0X0XXX$_B$
- OCCP0~5 : $XXXX_H$ (Compare value)
- TMRR0~2 : XXXXXXXXX$_B$ (non-overlap timing setting)
- SIGR : XXXXXX00$_B$ (DTTI input and 8-bit timer count clock setting
- CPCLR : $XXXX_H$ (Cycle setting)
- OSC0~5 : ---0XXXXXXXX--11$_B$
- DTCR0~2 : 01111001$_B$

Note: "X" must be set according to the operation.

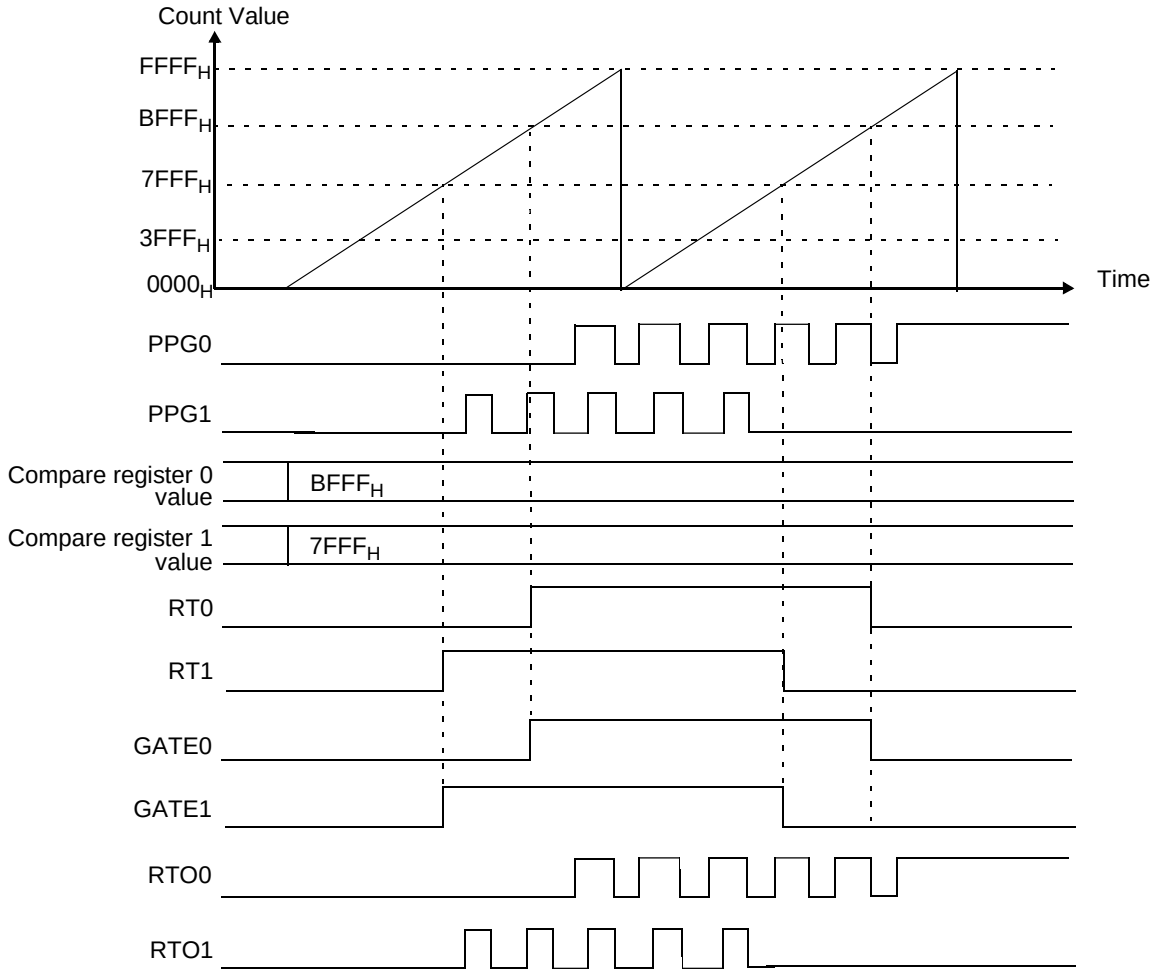


Figure 12.4.5.2-1 Generating PPG output/GATE signal during each RT is at "H" level

■ **Generating PPG output/GATE signal until value of 8-bit timer and 8-bit reload register is matched. (Starting the 8-bit timer at every rising edge of the RT)**

Setting up registers:

- TCDT : 0000_H
- TCCS : ------XXXX0X0XXX$_B$
- OCCP0~5 : $XXXX_H$ (Compare value)
- TMRR0~2 : XXXXXXXXX$_B$ (non-overlap timing setting)
- SIGR : XXXXXX00$_B$ (DTTI input and 8-bit timer count clock setting
- CPCLR : $XXXX_H$ (Cycle setting)
- OSC0~5 : ---0XXXXXXXX--11$_B$
- DTCR0~2 : 01111010$_B$

Note: "X" must be set according to the operation.

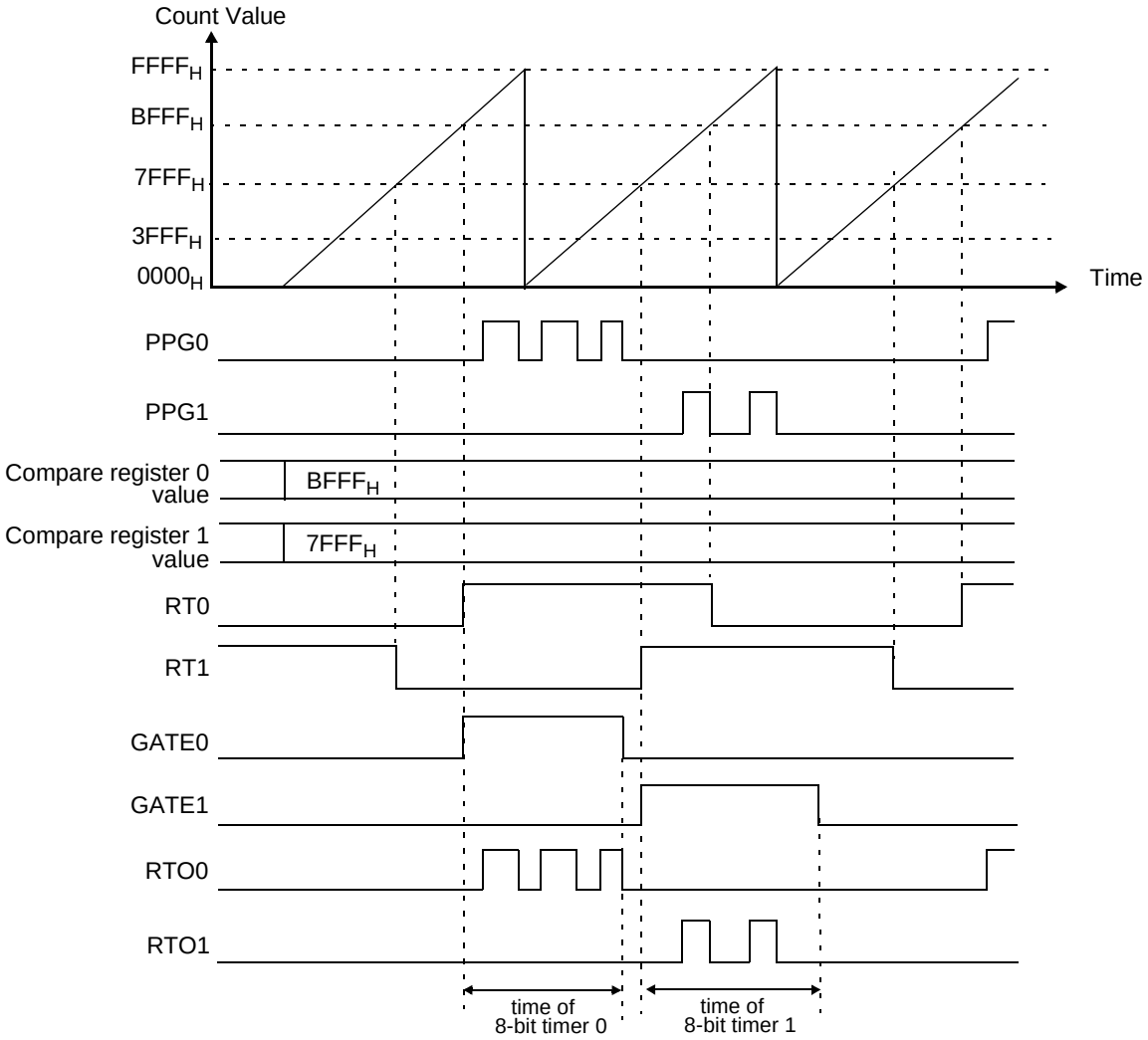


Figure 12.4.5.2-2 Generating PPG output/GATE signal until the value 8-bit timer and 8-bit reload register is matched.

Note: Each 8-bit timer is used for two RTs. i.e. 8-bit timer 0 is used for RT0 and RT1; 8-bit timer 1 is used for RT2 and RT3; 8-bit timer 2 is used for RT4 and RT5. Therefore, do not use an RT and attempt to start the corresponding timer that is already operating. Doing so may cause that the outputting GATE signal will be extended and malfunction will be occurred.

12.4.5.3 Operation of DTTI Pin Control

By setting “1” to waveform control register, SIGCR:bit 7 (DTIE), for the wave output of RTO0~5 can be controlled by the DTTI pin. A valid input level which is set by SIGCR:bit 6 (DTIL) and is detected, the output of RTO0~5 will be fixed to an inactive level. The inactive level can be set pin by pin to PDR3 of port 3 by software.

■ DTTI pin input operation

Even when the valid DTTI pin input is detected, the timer will keep running for the waveform generator operation, but no waveform will outputted to external pins.

Setting up registers:

- TCDT : 0000_H
- TCCS : ------XXXX0X0XXX$_B$
- OCCP0~5 : XXXX$_H$ (Compare value)
- TMRR0~2 : XXXXXXXXX$_B$ (non-overlap timing setting)
- SIGR : 110XXXXX$_B$ (DTTI input and 8-bit timer count clock setting)
- CPCLR : XXXX$_H$ (Cycle setting)
- OSC0~5 : ---1XXXXXXXX--11$_B$
- DTCR0~2 : 00000100$_B$

Note: “X” must be set according to the operation.

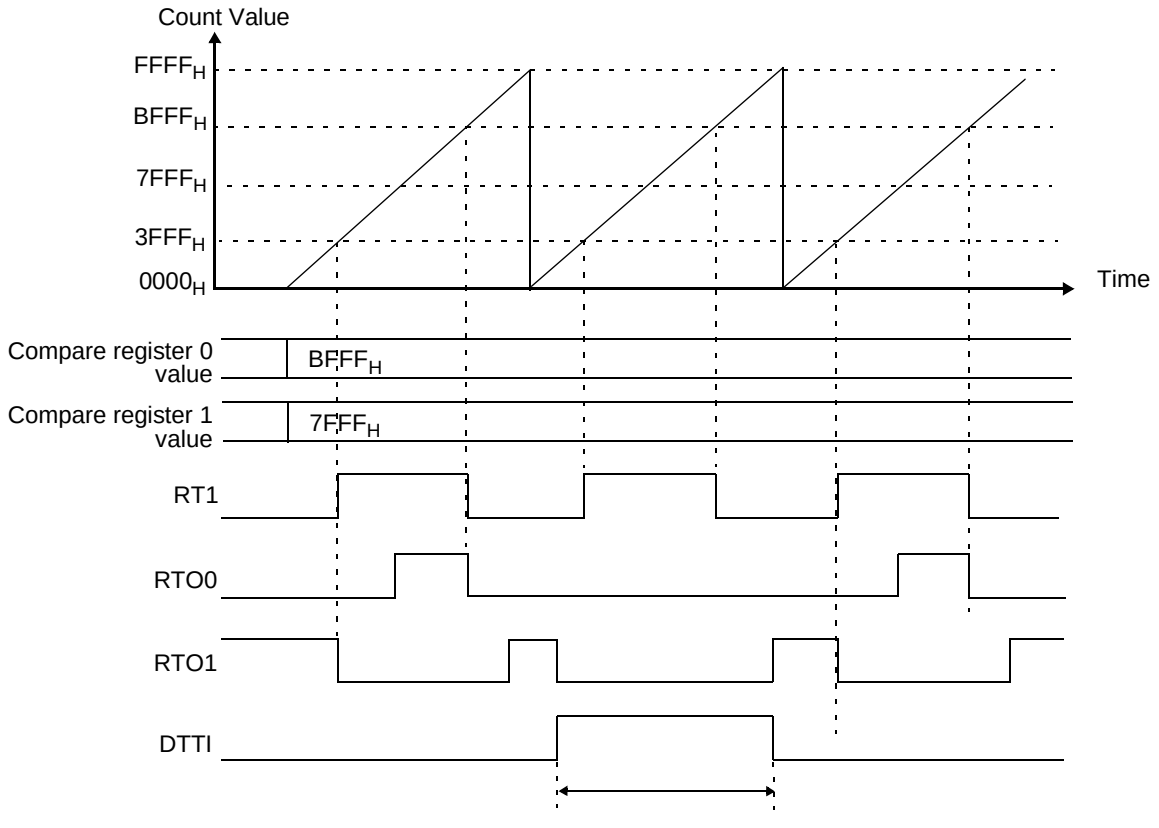


Figure 12.4.5.3-1 Operation when DTTI input is enabled

■ DTTI pin noise cancelling function

By setting bit 5 (NRSL) of the waveform control register (SIGCR) to “1”, the noise cancelling function for DTTI pin input is enabled. When noise cancelling function is enabled, approximately 4 machine cycles of delay (due to noise cancelling circuit) will be occurred for inactive level fix timing of output pin. Since the noise cancelling circuit operates by using machine clock, the DTTI input is invalid at stop mode (oscillator is stopped) even if the DTTI input is enabled.

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 normal 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.

Table 13.1-1UART functions

	Function
Data buffer	Full-duplex, double buffering
Transfer mode	• Clock synchronous (using start and stop bits) • Clock asynchronous (start-stop synchronization)
Baud rate	• A dedicated baud rate generator is provided. Eight settings can be selected. • An external clock can be input. • Internal clock (internal clocks supplied from 16-bit reload timer 0 can be used.)
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 (EI20S) 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.)

<Check>

During clock synchronous transfer, start and stop bits are not added so only data is transferred in UART.

Figure 13.1-1 UART operation mode

Operation mode		Data length		Synchronization 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 EI²OS

Table 13.1-2 UART interrupt and EI²OS

Interrupt cause	Interrupt number	Interrupt control register		Vector table address			EI²OS
		Register name	Address	Lower	Upper	Bank	
UART1 reception interrupt	#37(25H)	ICR13	0000BDH	FFFF68H	FFFF69H	FFFF6AH	◎
UART1 transmission interrupt	#38(26H)	ICR13	0000BDH	FFFF64H	FFFF65H	FFFF66H	Δ
UART0 reception interrupt	#39(27H)	ICR14	0000BEH	FFFF60H	FFFF61H	FFFF62H	◎
UART0 transmission interrupt	#40(28H)	ICR14	0000BEH	FFFF5CH	FFFF5DH	FFFF5EH	Δ

◎ : Provided with a function that detects a UART reception error and stops EI²OS

Δ : 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**
- **Serial Mode Control Register (SMR0/1)**
- **Serial Control Register (SCR0/1)**
- **Serial Status Register (SSR0/1)**
- **Serial Input Data Register (SIDR0/1)**
- **Serial Output Data Register (SODR0/1)**

■ Block Diagram of UART

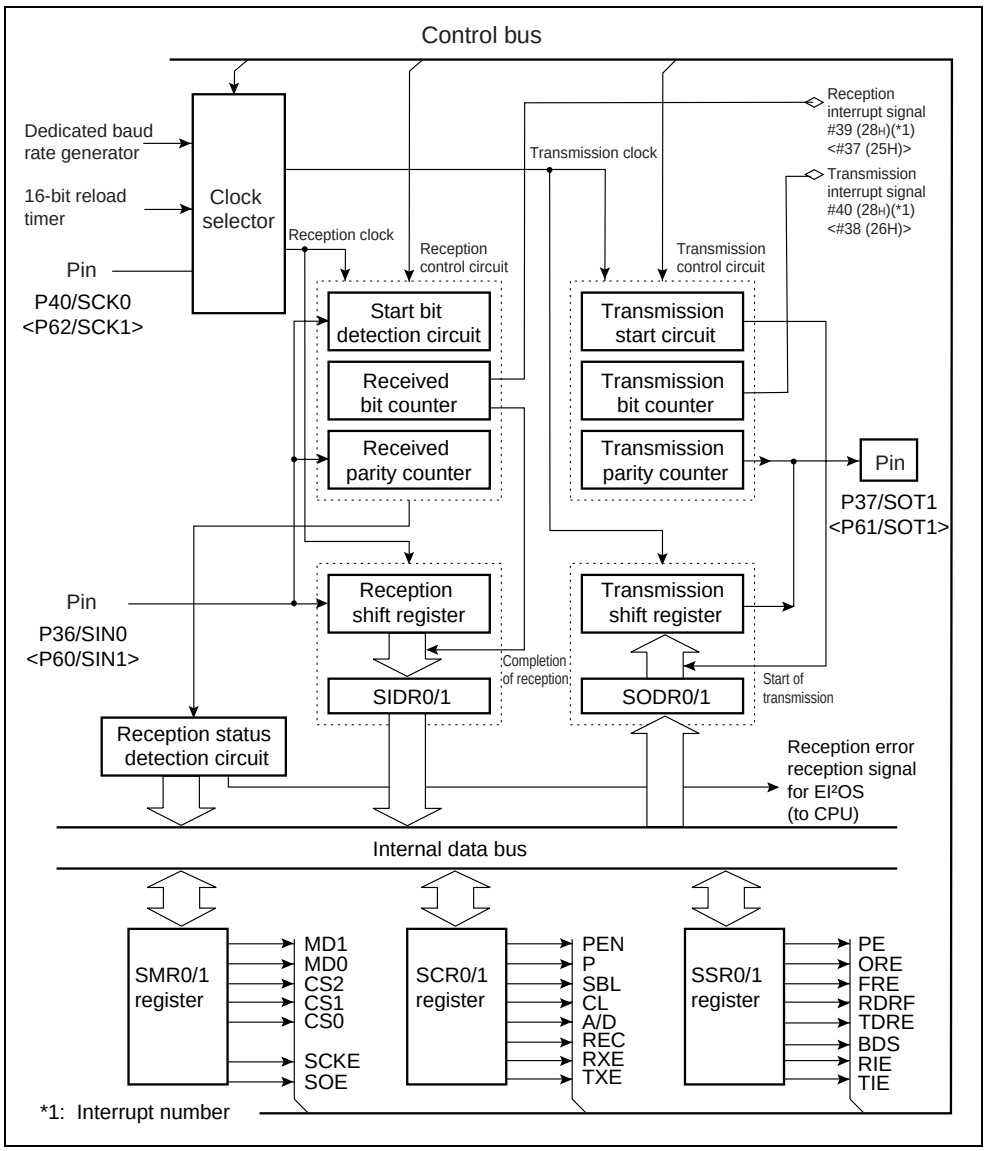


Figure 13.2-1 Block diagram of UART

- **Clock Selector**

  The clock selector selects the dedicated baud rate generator, external input clock, or internal clock (clock supplied from the 16-bit reload timer) as the transmitting and receiving clocks.

- **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 SIDR1 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 data is written to SODR0/1. The transmission parity counter generates a parity bit for data to be transmitted when parity is enabled.

- **Reception Shift Register**

  The reception shift register fetches receive data input from the SIN 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 SOT pin, shifting the data bit by bit.

- **Serial Mode Control Register 1 (SMC0/1)**

  This register performs the following operations:

  - Selecting a UART operation mode
  - Selecting a clock input source
  - Setting up the dedicated baud rate generator
  - Selecting a clock rate (clock division value) when using the dedicated baud rate generator
  - Specifying whether to enable serial data output to the corresponding pin
  - Specifying whether to enable clock output to the corresponding pin

- **Serial Control Register 1 (SCR0/1)**

  This register performs the following operations:

  - Specifying whether to provide parity bits
  - Selecting parity bits
  - Specifying a stop bit length
  - Specifying a data length
  - Selecting a frame data format in mode 1
  - Clearing a flag
  - Specifying whether to enable transmission
  - Specifying whether to enable reception

- **Serial Status Register 1 (SSR0/1)**

  This register checks the transmission and reception status and error status, and enables and disables transmission and reception interrupt requests.

- **Serial Input Data Register 1 (SIDR0/1)**

  This register retains receive data. Serial input data is converted and stored in this register.

- **Serial Output Data Register 1 (SODR0/1)**

  This register retains transmission data. Data written to this register is converted to serial data and output.

Memo

13.3 UART Pins

This section describes the UART pins and provides a pin block diagram.

■ UART Pins

The UART pins also serve as general ports. Table 13.3-1 lists the pin functions, I/O formats, and settings required to use UART.

Table 13.3-1UART pins

Pin name	Pin function	I/O format	Pull-up	Standby control	Setting required to use pin
P36/SIN0	Port 3 I/O or serial data input	CMOS output and CMOS hysteresis input	Not Available	Provided	Set as an input port (DDR3: bit6 = 0)
P37/SOT0	Port 3 I/O or serial data output				Set to output enable mode (SMR0: SOE = 1)
P40/SCK0	Port 4 I/O or serial clock input/output				Set as an input port when a clock is input (DDR4: bit 0=0)
					Set to output enable mode when a clock is output (SMR0: SCKE = 1)
P60/SIN1	Port 6 I/O or serial data input	CMOS output and CMOS hysteresis input	Not Available	Provided	Set the pin as an input port (DDR6: bit0 = 0)
P61/SOT1	Port 6 I/O or serial data output				Set the pin as an input port (SMR1: SOE = 1)
P62/SCK1	Port 6 I/O or serial clock input/output				Set as an input port when a clock is input (DDR6: bit2 = 0)
					Set to output enable mode when a clock is output (SMR1:SCKE = 1)

■ Block Diagram of UART Pins

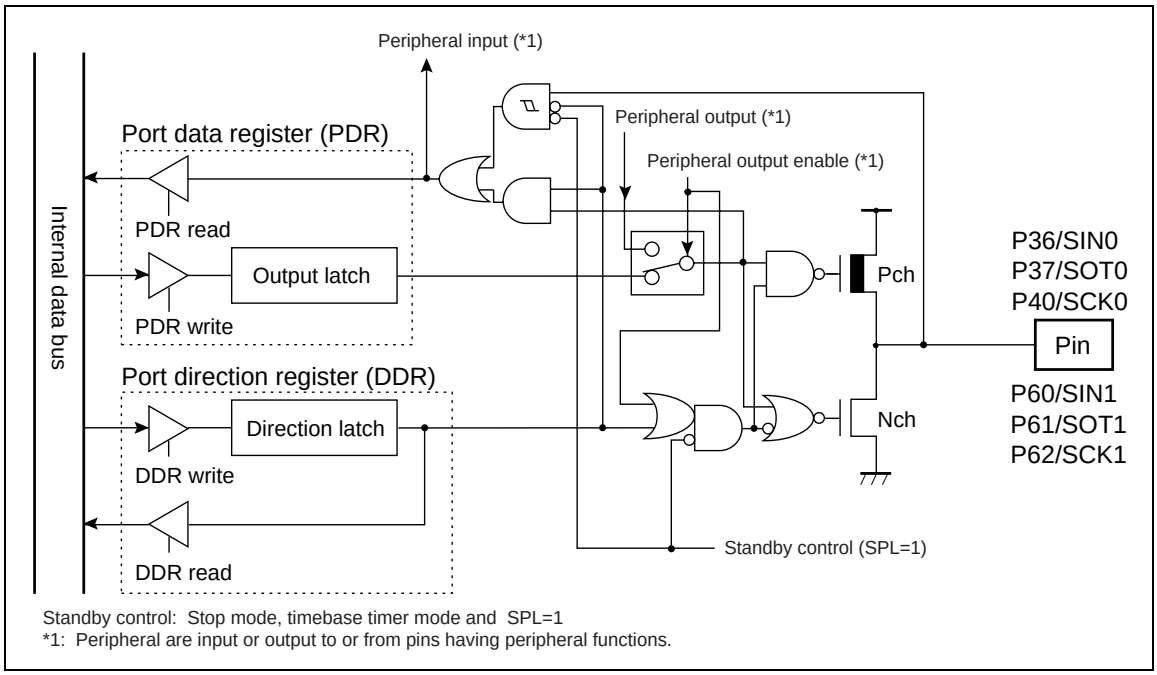


Figure 13.3-1 Block Diagram of UART Pins

13.4 UART Registers

The following figure shows the UART registers.

■ **UART Registers**

Address	bit15 bit8	bit7 bit0
ch0:000021H,20H ch0:000025H,24H	SCR (Serial control register)	SMR (Serial mode control register)
ch0:000023H,22H ch0:000027H,26H	SSR (Serial status register)	SIDR/SODR (Serial input/output data register)
ch0:000029H ch1:00002BH	CDCR (communication prescaler control register)	Reserved

Figure 13.4-1 UART registers

Memo

13.4.1 Serial Control Register 1 (SCR0/1)

This register specifies parity bits, selects the stop bit and data lengths, selects a frame data format in mode 1, clears the reception error flag, and specifies whether to enable transmission and reception.

■ **Serial Control Register (SCR0/1)**

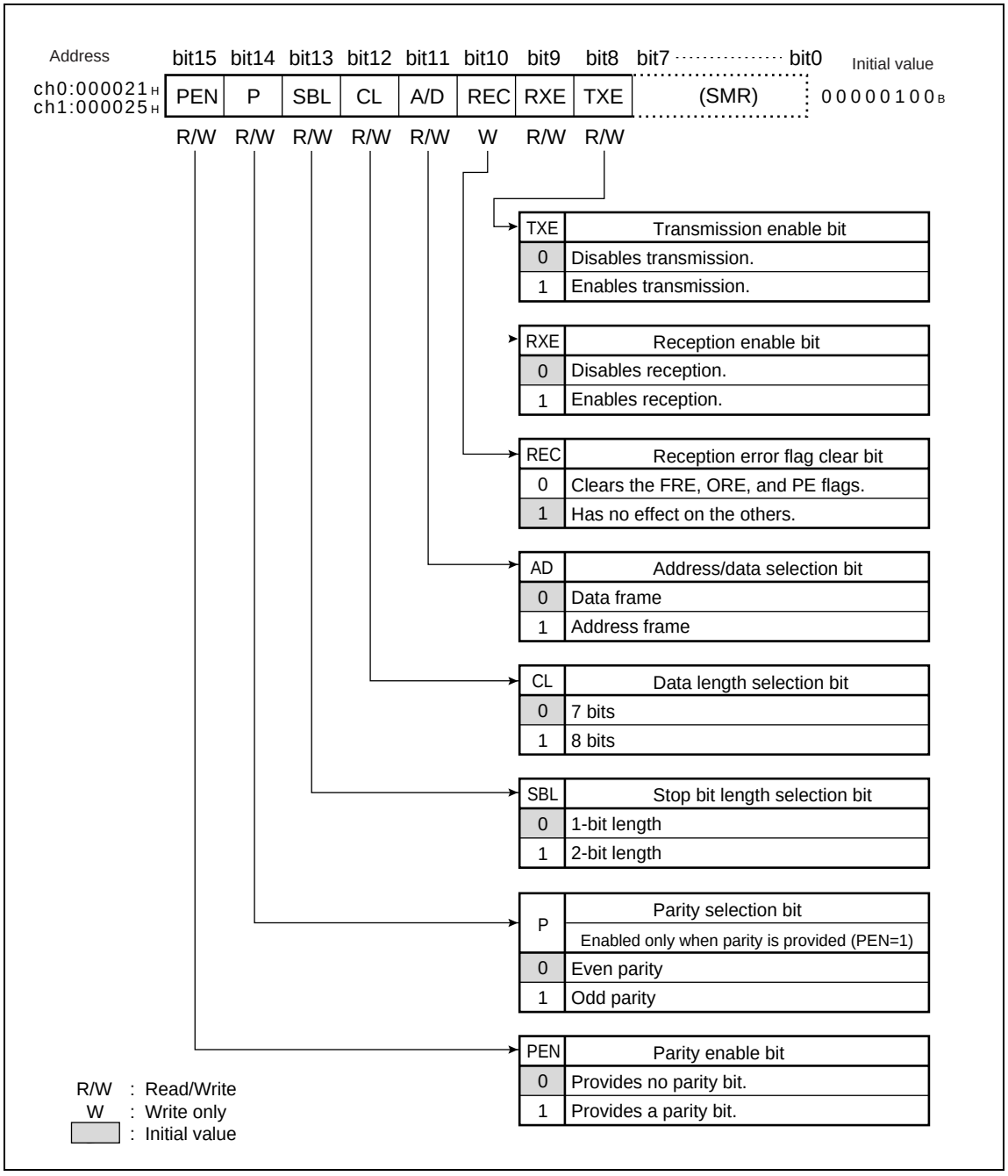


Figure 13.4-2 Serial Control register (SCR0/1)

Table 13.4-1 Functions of each bit of serial control register (SCR0/1)

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. **<Caution>** No parity can be used in operation modes 1 and 2, so that fix this bit to “0”.
bit14	P: Parity selection bit	When parity is provided (PEN=1), this bit selects an even or odd parity.
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. **<Caution>** 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. **<Caution>** Seven bits can be selected in operation mode 0 (asynchronous) only. Be sure to select eight bits (CL=1) in operation mode 1 (multi-processor 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 multi-processor mode (mode 1). • Select normal 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 (SSR). • Write “0” to this bit to clear the FRE, ORE, and PE flag. Writing “1” to this bit has no effect on the others. **<Caution>** 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. **<Caution>** If reception operation is disabled during reception, finish frame reception and store the received data in the receive data buffers (SIDRI). Then stop the reception operation.
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. **<Caution>** When transmission operation is disabled during transmission, wait until there is no data in the send data buffers (SODR1) before stopping the transmission operation.

13.4.2 Serial Mode Control Register (SMR0/1)

This register selects an operation mode and baud rate clock and specifies whether to enable output of serial data and clocks to the corresponding pin.

■ **Serial Mode Control Register (SMR0/1)**

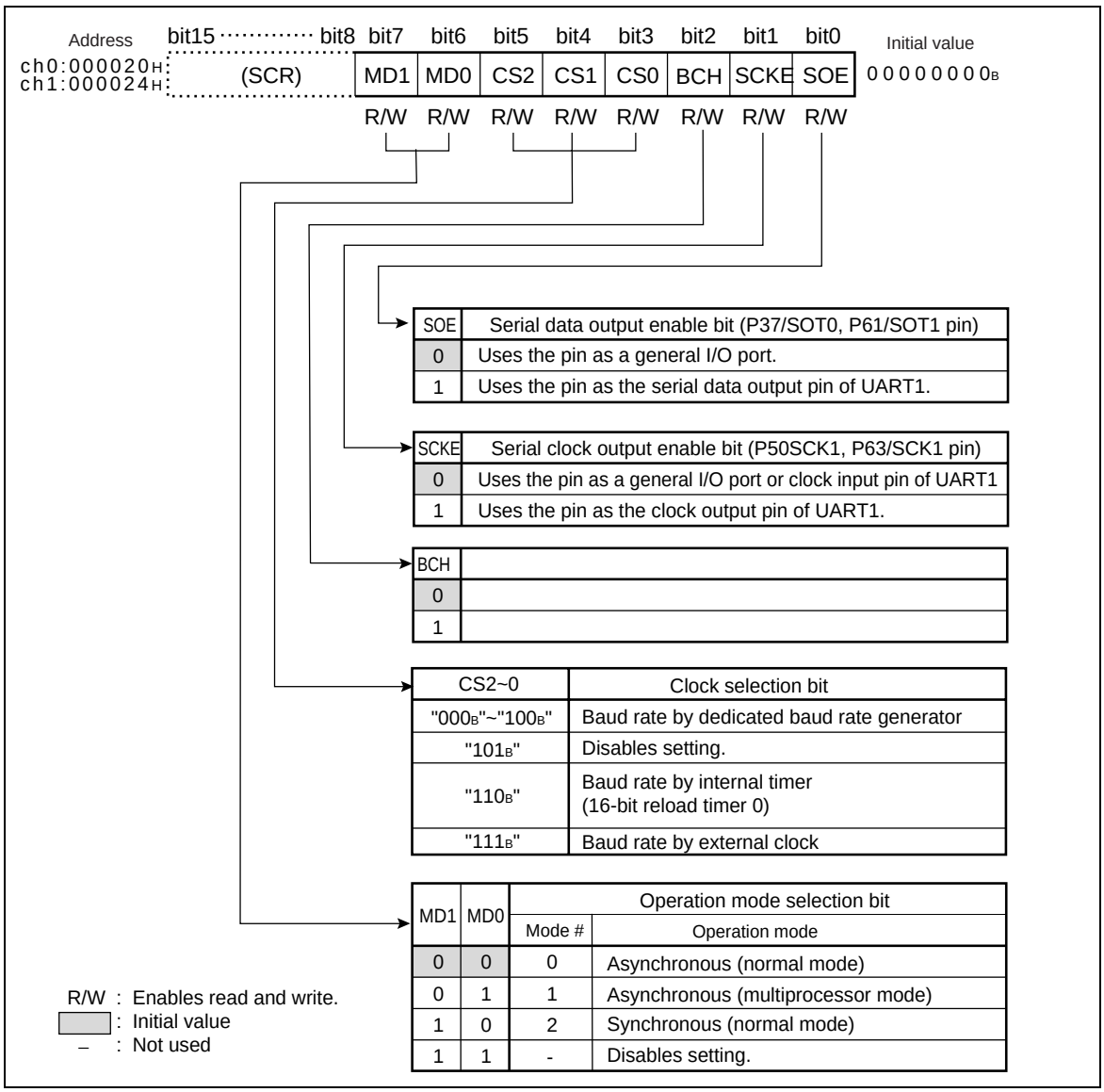


Figure 13.4-3 Serial Mode control register (SMR0/1)

Table 13.4-2 Functions of each bit of serial mode control register (SMR0/1)

Bit name		Function
bit7 bit6	MD1 and MD0: Operation mode selection bits	These bits select an operation mode. **<Caution>** 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: five in asynchronous transfer mode and three in synchronous transfer mode. • Input clocks can be selected from external clocks (SCK0/1 pin), 16-bit reload timer 0, and the dedicated baud rate generator.
bit2	BCH:	
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 general input-output ports (P40 and P62) or serial clock input pins. When this bit is “1”, the pins operate as serial clock output pins. **<Caution>** • When using the P40/SCK0 and P62/SCK1 pins as serial clock input (SCKE=0) pins, set the P40 and P62 as input ports. Also, select external clocks (SMR0/1: CS2 to CS0 = 111B) using the clock selection bits. • When using the pins as serial clock output (SCKE=1) pins, select clocks other than external clocks (other than SMR0/1: CS2 to CS0 = 111B). **<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 general input-output ports.
bit0	SOE: Serial data output enable bit	• This bit enables or disables the output of serial data. • When this bit is “0”, the P37/SOT0 and P61/SOT1 pins operate as general input-output ports (P37 and P61). When this bit is 1, the P37/SOT0 and P61/SOT1 pins operate as serial data output pins (SOT0/1). **<Reference>** When serial data is output (SOE=1), the enabled, the P37/SOT0 and P61/SOT1 pins function as SOT0/1 pins regardless of the status of general input-output ports (P37 and P61)

13.4.3 Serial Status Register (SSR0/1)

This register checks the transmission and reception status and error status, and enables and disables the transmission and reception interrupts.

■ Serial Status Register (SSR0/1)

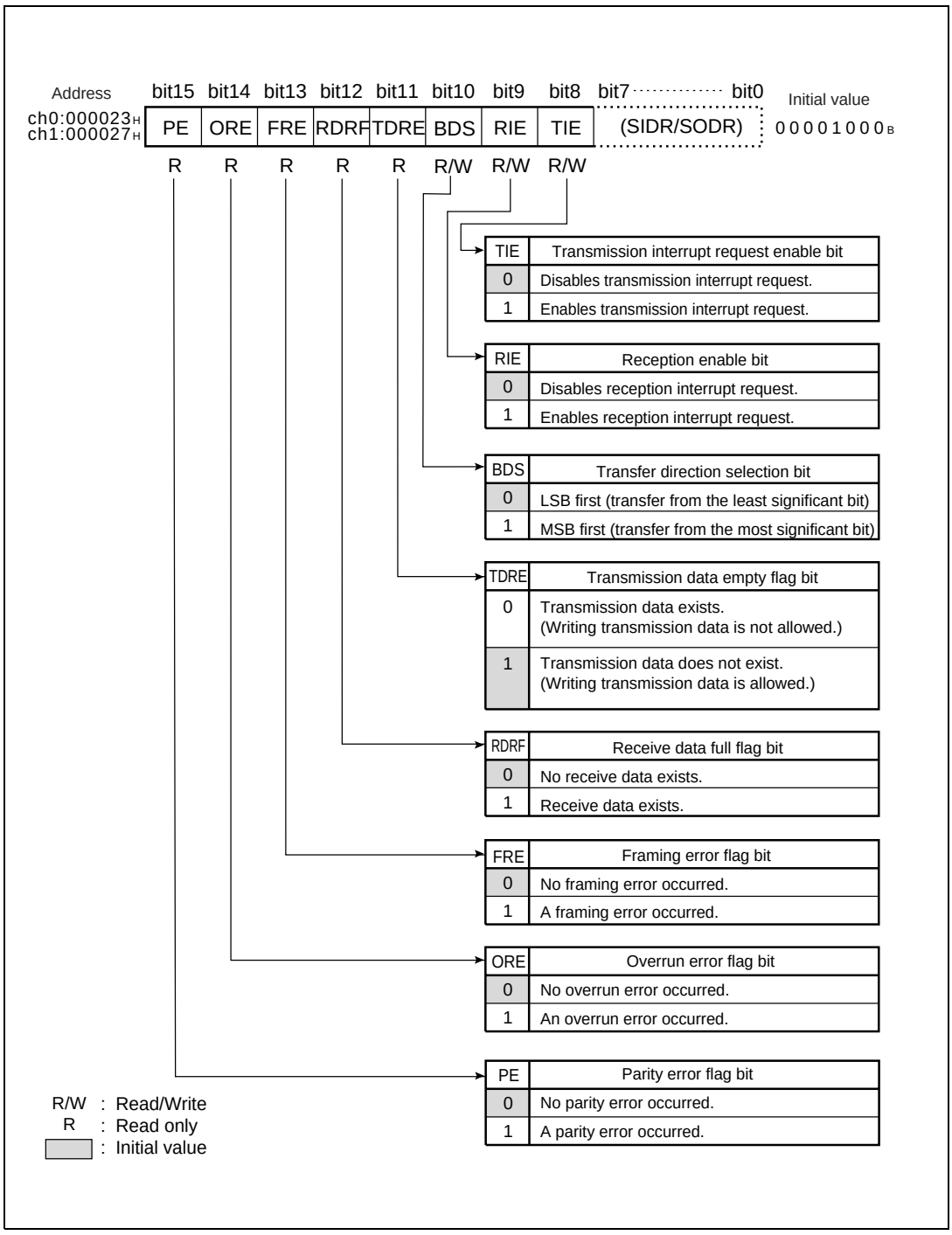


Figure 13.4-4 Status register (SSR0/1)

Table 13.4-3 Functions of each bit of serial status register (SSR0/1)

NO.	Bit name	Function
bit15	PE: Parity error flag bit	• This bit is set to “1” when a parity error occurs during reception and is cleared when “0” is written to the REC bit of the serial control register (SCR0/1). • A reception interrupt request is generated when this bit and the RIE bit are “1”. • Data in serial input data register (SIDR0/1) is invalid when this flag is set.
bit14	ORE: Overrun error flag bit	• This bit is set to “1” when an overrun error occurs during reception and is cleared when “0” is written to the REC bit of the serial control register (SCR0/1). • A reception interrupt request is generated when this bit and the RIE bit are “1”. • Data in the serial input data register (SIDR0/1) is invalid when this flag is set.
bit13	FRE: Framing error flag bit	• This bit is set to “1” when a framing error occurs during reception and is cleared when “0” is written to the REC bit of the serial control register (SCR0/1). • A reception interrupt request is generated when this bit and the RIE bit are “1”. • Data in the serial input data register (SIDR0/1) is invalid when this flag is set.
bit12	RDRF: Receive data full flag bit	• This flag indicates the status of the input data register (SIDR0/1). • This bit is set to “1” when receive data is loaded into SIDR0/1 and is cleared to “0” when serial input data register SIDR0/1 is read. • A reception interrupt request is generated when this bit and the RIE bit are “1”.
bit11	TDRE: Transmission data empty flag bit	• This flag indicates the status of output data register (SODR0/1). • This bit is cleared to “0” when transmission data is written to SODR0/1 and is set to “1” when data is loaded into the transmission shift register and transmission starts. • A transmission interrupt request is output when this bit and the RIE bit are “1”. **<Caution>** This bit is set to “1” (SODR0/1 empty) as its initial value.
bit10	BDS: Transfer direction selection bit	• This bit selects whether to transfer serial data from the least significant bit (LSB first, BDS=0) or the most significant bit (MSB first, BDS=1). **<Caution>** The high-order and low-order sides of serial data are interchanged with each other during reading from or writing to the serial data register. If this bit is set to another value after the data is written to the SDR register, the data becomes invalid.
bit9	RIE: Reception interrupt request enable bit	• This bit enables or disables reception interrupt request to the CPU. • A reception interrupt request is generated when this bit and the receive data flag bit (RDRF) are “1” or this bit and one or more error flag bits (PE, ORE, and FRE) are “1”.
bit8	TIE: Transmission interrupt request enable bit	• This bit enables or disables transmission interrupt request o the CPU. • A transmission interrupt request is generated when this bit and the TDRE bit are “1”.

13.4.4 Serial Input Data Register (SIDR0/1) and Serial Output Data Register (SOR0/1)

The serial input data register (SIDR0/1) is a serial data reception register. The serial output data register (SODR0/1) is a serial data transmission register. Both SIDR0/1 and SODR0/1 registers are located in the same address.

■ Serial Input Data Register (SIDR0/1)

Figure 13.4-5 shows the bit configuration of serial input data register 1.

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

Figure 13.4-5 Serial input data register (SIDR0/1)

SIDR0/1 is a register that contains receive data. After the serial data signal transmitted from the SIN0/1 pin to the shift register, the data is stored there. When the data length is 7 bits, the uppermost bit (D7) contains invalid data. When receive data is stored in this register, the receive data full flag bit (SSR0/1: RDRF) is set to “1”. If a reception interrupt request is enabled at this point, a reception interrupt request is generated.

Read SIDR0/1 when the RDRF bit of the status register (SSR0/1) is “1”. The RDRF bit is cleared automatically to “0” when SIDR01/1 is read.

Data in SIDR0/1 is invalid when a reception error occurs (SSR0/1: PE, ORE, or FRE = 1).

■ Serial Output Data Register (SODR0/1)

Figure 13.4-6 shows the bit configuration of the serial output data register.

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	

R : Write only
X : Indefinite

Figure 13.4-6 Output data register (SODR0/1)

When data to be transmitted is written to this register in transmission enable state, it is transferred to the transmission shift register, then converted to serial data, and transmitted to the serial data output terminal (SOT0/1 pin). When the data length is 7 bits, the uppermost bit (D7) contains invalid data.

When transmission data is written to this register, the transmission data empty flag bit (SS0/1: TDRE) is cleared to “0”. When transfer to the transmission shift register is complete, the bit is set to “1”. When the TDRE bit is “1”, the next transmission data can be written. If output transmission interrupt requests have been enabled, a transmission interrupt request is generated. Write the next transmission data when a transmission interrupt is generated or the TDRE bit is “1”.

<Check>

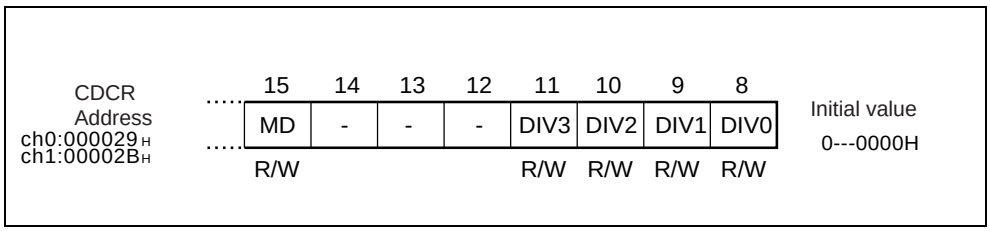
SODR0/1 is a write-only register and SIDR0/1 is a read-only register. These registers are located in the same address, so the read value is different from the write value. Therefore, instructions that perform a read-modify-write (RMW) operation, such as the INC/DEC instruction, cannot be used.

13.4.5 Communication Prescaler Control Register (CDCR0/1)

This register controls the division of machine clocks.

■ Communication Prescaler Control Register (CDCR0/1)

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. The CDCR bit configuration is shown below.



[Bit 15] MD(Machine clock divide mode select)

Operation enable bit of the communication prescaler

0: Stops the communication prescaler.

1: Operates the communication prescaler.

[Bit 11,10,9,8] DIV3~0(DIVide3~0)

Machine clock division ratios are determined according to the table in Table 13.4-4.

Table 13.4-4Communication prescaler

MD	DIV3	DIV2	DIV1	DIV0	Divided by
0	–	–	–	–	Stop
1	0	0	0	0	1
1	0	0	0	1	2
1	0	0	1	0	3
1	0	0	1	1	4
1	0	1	0	0	5
1	0	1	0	1	6
1	0	1	1	0	7
1	0	1	1	1	8

<Check>

If a division ratio is changed, wait two time cycles as clock stabilization time before starting communication.

Memo

13.5 UART Interrupts

UART uses both reception and transmission interrupts. An interrupt request can be generated for either of the following causes:

- **Receive data is transferred to the serial input register (SIDR0/1), or a reception error occurs.**
- **Transmission data is transferred from the serial output data register 1 (SODR0/1) to the transmission shift register.**

The extended intelligent I-O service (EI²OS) is available for these interrupts.

■ UART Interrupts

Table 13.5-1 lists the interrupt control bits and causes of UART.

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 (SSR1: REC).
	FRE	o	o	x	Framing error		
	PE	o	x	x	Parity error		
Transmission	TDRE	o	o	o	Empty transmission buffer (SODR0/1)	SSR0/1:TIE	Transmission data is written.

o : Used

x : Not used

● Reception Interrupt

If one of the following events occurs in reception mode, the corresponding flag bit of the status register is set to “1”:

- Data reception is complete (SSR0/1: RDRF)
- Overrun error (SSR0/1: ORE)
- Framing error (SSR0/1; FRE)
- Parity error (SSR0/1: PE)

When at least one of the flag bits is “1” and the reception interrupts are enabled (SSR0/1: RIE=1), a reception interrupt request is generated to the interrupt controller.

When the serial input data register (SIDR0/1) is read, the receive data full flag (SSR0/1: RDRF) is automatically cleared to “0”. When “0” is written to the REC bit of the serial control register (SCR0/1), all the reception error flags (SSR0/1: PE, ORE, and FRE) are cleared to “0”.

● **Transmission Interrupt**

When transmission data is transferred from the serial output data register (SODR0/1) to the transfer shift register, the TDRE bit of the serial status register (SSR0/1) is set to “1”. When the transmission interrupts have been enabled (SSR0/1: TIE=1), a transmission interrupt request is generated to the interrupt controller.

■ UART Interrupts and EI²OS

Table 13.5.2 UART interrupts and EI²OS

Interrupt cause	Interrupt number	Interrupt control register		Vector table address			EI²OS
		Register name	Address	Lower	Upper	Bank	
UART1 reception interrupt	#37(25H)	ICR13	0000BDH	FFFF68H	FFFF69H	FFFF6AH	◎
UART1 transmission interrupt	#38(26H)	ICR13	0000BDH	FFFF64H	FFFF65H	FFFF66H	Δ
UART0 reception interrupt	#39(27H)	ICR14	0000BEH	FFFF60H	FFFF61H	FFFF62H	◎
UART0 transmission interrupt	#40(28H)	ICR14	0000BEH	FFFF5CH	FFFF5DH	FFFF5EH	Δ

◎ : Provided with a function that detects a UART reception error and stops EI²OS

Δ : Usable when interrupt causes that share the ICR13 and ICR14 or the interrupt vectors are not used

■ UART EI²OS Functions

UART has a circuit for operating EI²OS, which can be started up for either reception or transmission interrupts.

● **For Reception**

EI²OS 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, EI²OS 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 the reception interrupt causes: completion of reception (SSR0/1: RDRF) and occurrence of a reception error (SSR0/1: PE, ORE, or FRE).

■ Reception Interrupt Generation and Flag Set Timing

Receive data is stored in the serial input data register 1 (SIDR0/1) if a stop bit is detected (in operation mode 0 or 1) or the last bit of data is detected (in operation mode 2) during reception. If a reception error is detected, the error flags (SSR0/1: PE, ORE, and FRE) are set, then the receive data flag (SSR0/1: RDRF) is set to “1”. If one of the error flags is “1” in each mode, the SIDR0/1 register contains invalid data.

● Operation Mode 0 (Asynchronous, Normal Mode)

The RDRF bit is set to “1” when a stop bit is detected. If a reception error is detected, the error flags (PE, ORE, and FRE) are set.

● Operation Mode 1 (Asynchronous, Multiprocessor Mode)

The RDRF bit is set to “1” when a stop bit is detected. If a reception error is detected, the error flags (ORE and FRE) are set. Parity errors cannot be detected.

● Operation Mode 2 (Synchronous, Normal Mode)

The RDRF bit is set when the last bit of receive data (D7) is detected. If a reception error is detected, the error flag (ORE) is set. Parity and framing errors cannot be detected. The figure below shows the reception operation and flag set timing.

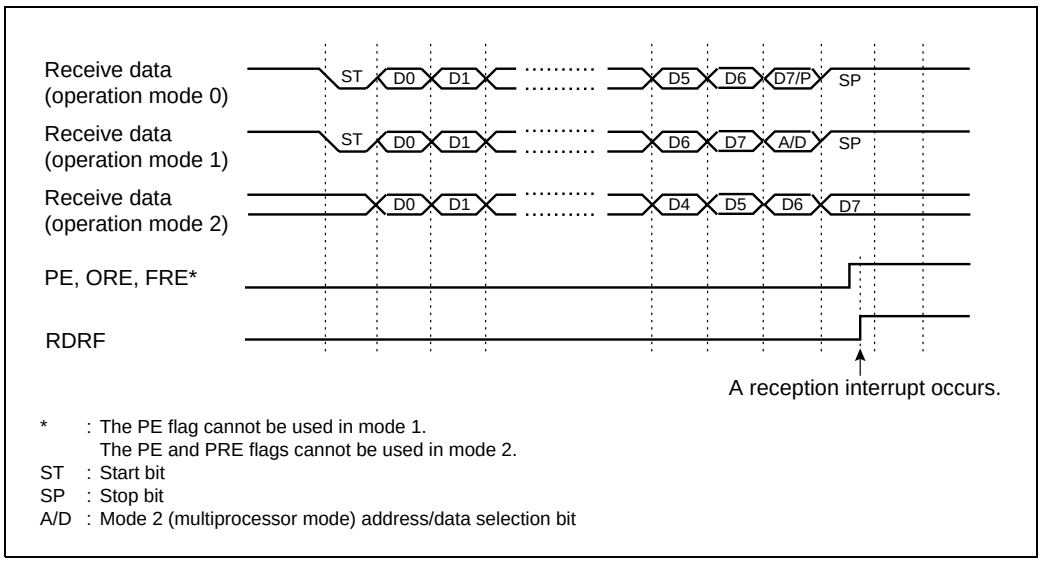


Figure 13.5-1 Reception operation and flag set timing

● Reception Interrupt Generation Timing

When the RDRF, PE, ORE, or FRE flag is set to “1” in the reception interrupt enable state (SSR0/1: RIE=1), reception interrupt requests (#37 and #39) are generated.

13.5.2 Transmission Interrupt Generation and Flag Set Timing

A transmission interrupt is generated when the next piece of data is ready to be written to the output data register (SODR0/1).

■ Transmission Interrupt Generation and Flag Set Timing

The transmission data empty flag bit (SSR0/1: TDRE) is set to “1” when data written to the output data register (SODR0/1) is transferred to the transmission shift register, and the next data is ready to be written. TDRE is cleared to “0” when transmission data is written to SODR0/1. Figure 13.5-2 shows the transmission operation and flag set timing.

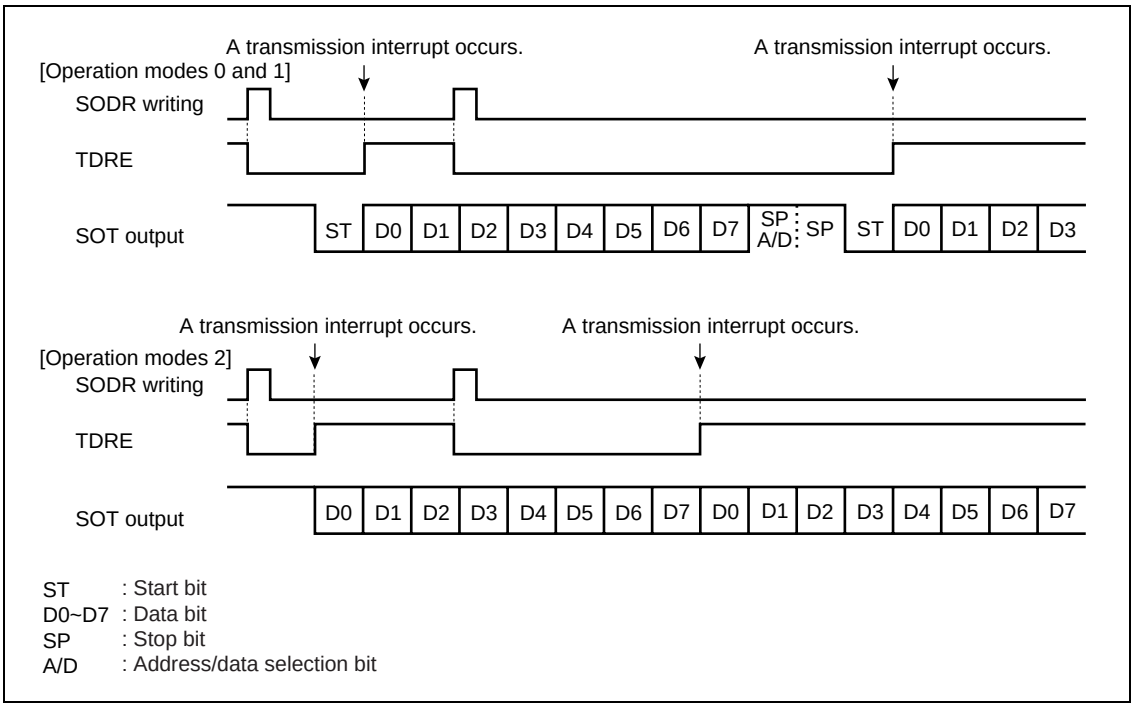


Figure 13.5-2 Transmission operation and flag set timing

● Transmission Interrupt Request Generation Timing

If the TDRE flag is set to “1” when a transmission interrupt is enabled (SSR0/1: TIE=1), transmission interrupt requests (#38 and #40) are generated.

<Check>

A transmission completion interrupt is generated immediately after the transmission interrupts are enabled (TIE=1) because the TDRE bit is set to “1” as its initial value. TDRE is a read-only bit that can be cleared only by writing new data to the serial output data register (SODR0/1). Carefully specify the transmission interrupt enable timing.

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 SCK 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 eight baud rates can be selected using the communication prescaler control register (CDCR0/1).

An asynchronous or clock synchronous baud rate is selected using the machine clock frequency and the BCH 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 0 is used as it is (synchronous) or by dividing it by 16 (asynchronous) for the baud rate. Any baud rate can be set by setting the reload value of 16-bit reload register (TMRLR0).

● Baud Rates Determined Using the External Clock

The clock input from the UART clock pulse input pins (SCK0/P40 and SCK1/P62) is used as it is (synchronous) or by dividing it by 16 (asynchronous) for the baud rate. Any baud rate can be set externally.

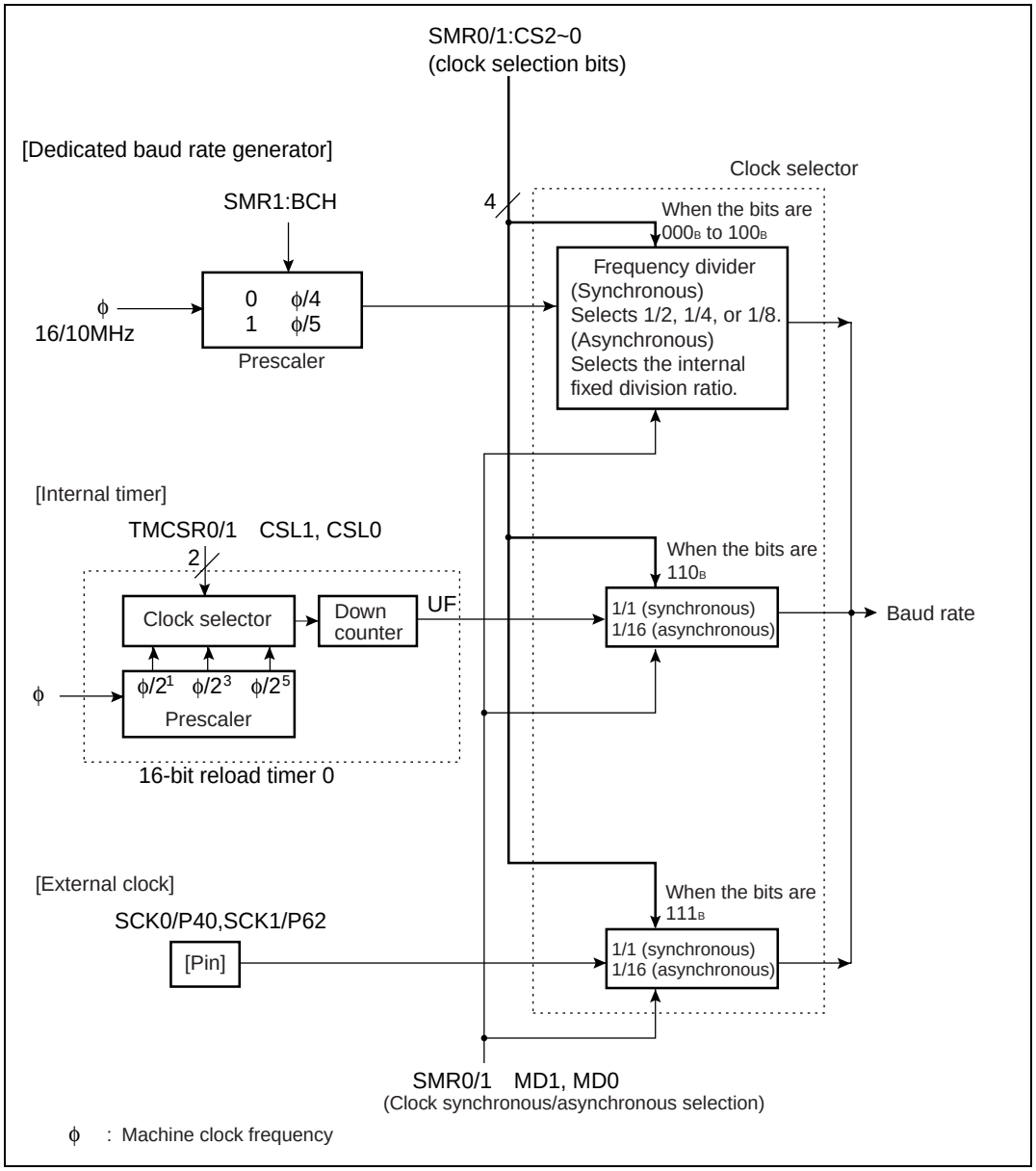


Figure 13.6-1 baud rate selection circuit

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 transfer clock is generated using the dedicated baud rate generator, the machine clock is divided with the machine clock prescaler. The divided machine clock is divided by the transfer clock division ratio selected with the clock selector again.The machine clock division ratios are common to the asynchronous and synchronous baud rates, but different values set internally are selected as the transfer clock division ratio for the asynchronous and synchronous baud rates.

The actual transfer rate can be calculated using the following formulas:

asynchronous baud rate = ϕ x (prescaler division ratio) x (asynchronous transfer clock division ratio)

synchronous baud rate = ϕ x (prescaler division ratio) x (synchronous transfer clock division ratio)

ϕ : Machine clock frequency

● Division Ratios for the Prescaler (Common to Asynchronous and Synchronous Baud Rates)

Each machine clock division ratio is selected using the DIV3 to DIV0 bits of the CDCR register as listed in Table 13.6-1.

Table 13.6-1 Selection of each division ratio for the machine clock prescaler

MD	DIV3	DIV2	DIV1	DIV0	Divided by (div)
0	–	–	–	–	stop
1	0	0	0	0	1
1	0	0	0	1	2
1	0	0	1	0	3
1	0	0	1	1	4
1	0	1	0	0	5
1	0	1	0	1	6
1	0	1	1	0	7
1	0	1	1	1	8

- **Synchronous Transfer Clock Division Ratios**

  A division ratio for synchronous baud rates is selected using the CS2 to CS0 bits of the mode control register (SMR0/1) as listed in Table 13.6-2.

Table 13.6-2 Selection of synchronous baud rate division ratios

CS2	CS1	CS0	Division ratio for CLK synchronization	Calculation formula	SCKI
0	0	0	16M	(ϕ ÷ div)/1	(ϕ ÷ div)/1
0	0	1	8M	(ϕ ÷ div)/2	(ϕ ÷ div)/2
0	1	0	4M	(ϕ ÷ div)/4	(ϕ ÷ div)/4
0	1	1	2M	(ϕ ÷ div)/8	(ϕ ÷ div)/8
1	0	0	1M	(ϕ ÷ div)/16	(ϕ ÷ div)/16
1	0	1	500K	(ϕ ÷ div)/32	(ϕ ÷ div)/32

The division ratio is calculated supposing that machine cycle ϕ = 16 MHz and div = 4.

- **Asynchronous Transfer Clock Division Ratios**

  A division ratio for 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.

Table 13.6-3 Selection of synchronous baud rate division ratios

CS2	CS1	CS0	Asynchronus (start-stop synchronization)	Calculation formula	SCKI
0	0	0	76923	(ϕ ÷ div)/(8x13x2)	(ϕ ÷ div)/(13x1)
0	0	1	38461	(ϕ ÷ div)/(8x13x4)	(ϕ ÷ div)/(13x2)
0	1	0	19230	(ϕ ÷ div)/(8x13x8)	(ϕ ÷ div)/(13x4)
0	1	1	9615	(ϕ ÷ div)/(8x13x16)	(ϕ ÷ div)/(13x8)
1	0	0	500K	(ϕ ÷ div)/(8x2x2)	(ϕ ÷ div)/2
1	0	1	250K	(ϕ ÷ div)/(8x13x4)	(ϕ ÷ div)/4

The division ratio is calculated supposing that machine cycle ϕ = 16 MHz and div = 1.

- **Internal Timer**

  When CS2 to CS0 are set to 110 and the internal timer is selected, the formulas for calculating baud rates (when using the reload timer) are as follows:

Asynchronous (start-stop synchronization): ($\phi \div$ N)/(16 x 2 x (n + 1))
CLK synchronization: ($\phi \div$ N)/(2 x (n + 1))

N: Timer count clock sourcen: Timer reload value

<Check>

In mode 2 (CLK synchronization mode), SCK0 is up to three clocks later than SCKI. A logically attainable transfer rate is 1/3 of the system clock frequency. However, 1/4 of the system clock frequency is recommended as taken from the actual specifications.

- **External Clock**

  When CS2 to CS0 are set to 111 and the external clocks are selected, note the following:

  If the external clock frequency is specified as f, the following baud rates are assumed:

  Asynchronous (start-stop synchronization): f/16

  CLK synchronization: f ’

  Note that f is up to 1/2 of the machine clock, and f’ is up to 1/8 of the machine clock.

Memo

13.6.2 Baud Rates Determined Using the Internal Timer (16-bit Reload Timer 0)

This section describes the settings used when the internal clock supplied from 16-bit reload timer 0 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 0)

Writing 110_B to the CS2 to CS0 bits of the mode control register (SMR0/1) selects the baud rate determined using the internal timer. Any baud rate can be set by specifying a prescaler division ratio and reload value for 16-bit reload timer 0. Figure 13.6-2 shows the baud rate selection circuit for the internal timer.

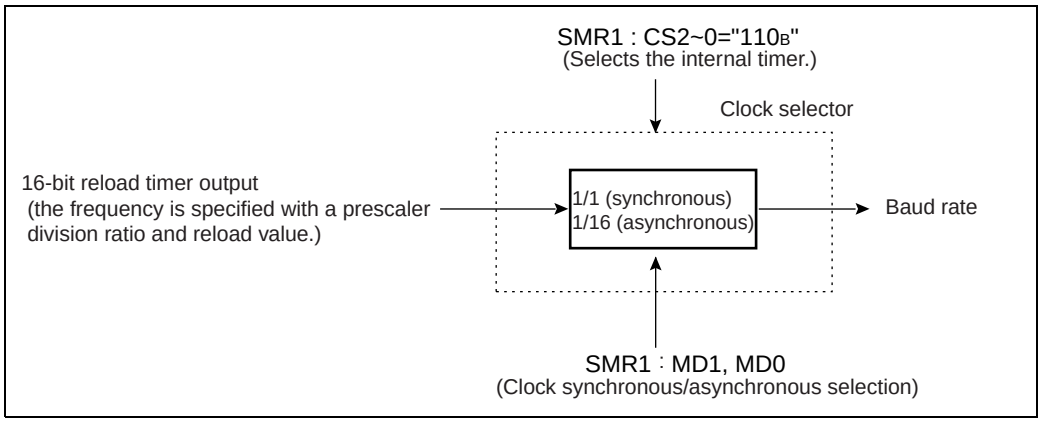


Figure 13.6-2 Baud rate selection circuit for the internal timer (16-bit reload timer 0)

● Baud Rate Calculation Formulas

$$\text{Asynchronous baud rate} = \frac{\phi}{X\ (n + 1) \times 2 \times 16}\ \text{bps}$$

$$\text{Synchronous baud rate} = \frac{\phi}{X\ (n + 1) \times 2}\ \text{bps}$$

ϕ: Machine clock frequency

X: 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			
	Clock asynchronous (start-stop synchronization)		Clock synchronous	
	$X=2^1$ (machine cycle divided by 2)	$X=2^3$ (machine cycle divided by 8)	$X=2^1$ (machine cycle divided by 2)	$X=2^3$ (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	9071	767
300	383	95	6143	1535

X : Division ratio for the prescaler of 16-bit reload timer 0

– : Setting not allowed

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 serial mode control register (SMR0/1) to select the baud rate determined by using the external clock input.
- Set the SCK0/P40 and SCK1/P62 pins as input ports (DDR4: bit 0 = 0 and DDR6: bit 2 = 0).
- Write “0” to the SCKE bit of the serial mode control register (SMR0/1) to set the pin as an external clock input pin.

As shown in Figure 13.6-3, a baud rate is selected using the external clock input from the SCK1 pin. To change the baud rate, the external input clock cycle must be changed because the internal division ratio is fixed.

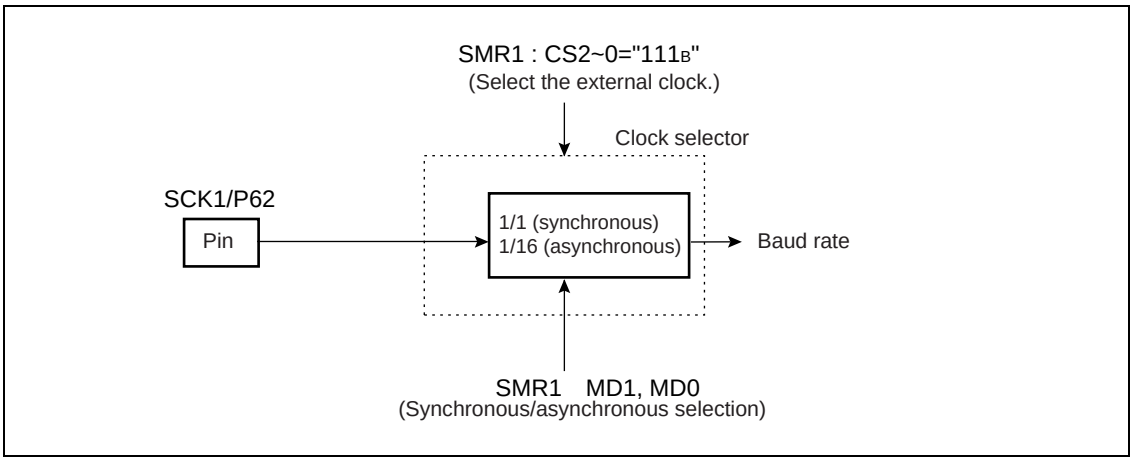


Figure 13.6-3 Baud rate selection circuit for the external clock

● Baud Rate Calculation Formulas

asynchronous baud rate = f/16

synchronous baud rate = f

f: External clock frequency (up to 2 MHz)

Memo

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, an operation mode can be selected according to the inter-CPU connection method and data transfer mode.

Table 13.7-1UART operation mode

Operation mode		Data length		Synchronization 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.

<Check>

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 CPUs. 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

UART can treat data only in non-return to zero (NRZ) format.

- **Operation Enable Bit**

  UART controls both transmission and reception using the operation enable bit for TXE (transmission) and that for RXE (reception). If each of the operations is disabled, stop it as follows:

  - 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 serial input data register (SIDRI). 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 serial output data register (SODR0/1) before stopping the transmission operation.

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), the asynchronous transfer mode is selected.

■ Operation in Asynchronous Mode

● Transfer data format

Transfer data begins with the start bit (level “L”) and ends with the stop bit (level “H”). The data of the specified data bit length is transferred in LSB first mode.

- In operation mode 0, the length of data with no parity is fixed to 7 bits, and that of data with parity is fixed to 8 bits.
- In operation mode 1, the length of data is fixed to 8 bits with an address/data (A/D) selection bit added instead of parity.

Figure 13.7-1 shows the data format in asynchronous mode.

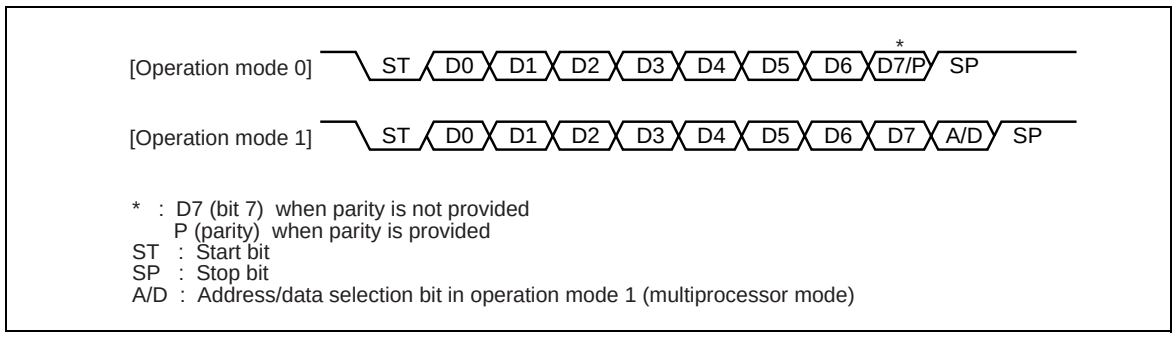


Figure 13.7-1 Transfer data format (operation modes 0 and 1)

● Transmission Operation

Transmission data is written to the serial output data register (SODR0/1) when the transmission data empty flag bit (SSR1: TDRE) is “1”. This data is transmitted if the transmission operation is enabled (SCR0/1: TXE=1).

The TDRE flag is again set to “1” when the transmission data is transferred to the transmission shift register and its transmission starts. Then, the next transmission data gets ready to be set. At this point, a transmission interrupt request is generated that the next transmission data can be set in the SODR0/1 register if that request is enabled (SSR0/1: TIE=1). The TDRE flag is cleared to “0” when the transmission data is written to SODR0/1.

● Reception Operation

Reception operation is performed every time it is enabled (SCR0/1: RXE=1). When a start bit is detected, a frame of data is received according to the data format specified by the serial control register (SCR0/1). After the frame has been received, the receive data full flag bit (SSR0/1: RDRF) is set to “1”. However, the error flag is set if an error occurs. At this point, a reception interrupt request is output if it is enabled (SSR1: TIE=1).

Check each flag of the serial status reister(SSR0/1). If the reception is normal, read the serial input data register (SIDR0/1). If an error is found, proceed to error handling. The RDRF flag is cleared to 0 every time receive data is read from SIDR0/1.

- **Stop Bit**

  For transmission, 1 or 2 bits can be selected. During reception however, the first bit is the only one that is always checked.

- **Error Detection**

  - In mode 0, parity, overrun, and framing errors can be detected.
  - In mode 1, overrun and framing errors can be detected but parity errors cannot be detected.

- **Parity 0**

  Parity can only be used in operation mode 0 (asynchronous, normal mode). Whether to provide parity can be specified using the PEN bit of the serial control register (SCR0/1). Even or odd parity can also be specified using the P bit of the serial control register (SCR0/1). In operation mode 1 (asynchronous, multiprocessor mode) and operation mode 2 (synchronous, normal mode), parity cannot be used. Figure 13.7-2 shows both transmission and receive data when parity is enabled.

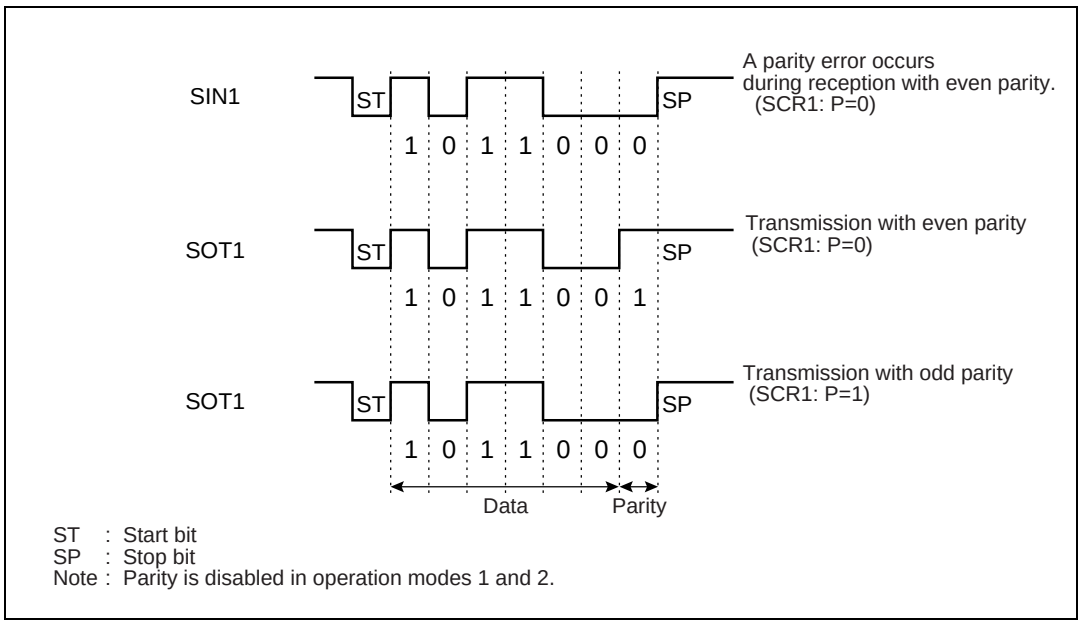


Figure 13.7-2 Transmission data when parity is enabled

13.7.2 Operation in Synchronous Mode (Operation Mode 2)

The clock synchronous transfer method is used for UART operation mode 2 (normal mode).

■ Operation in Synchronous Mode (Operation Mode 2)

● Transfer data format

In synchronous mode, 8-bit data is transferred using the LSB first method, in which start and stop bits are not added. Figure 13.7-3 shows the data format in clock synchronous mode.

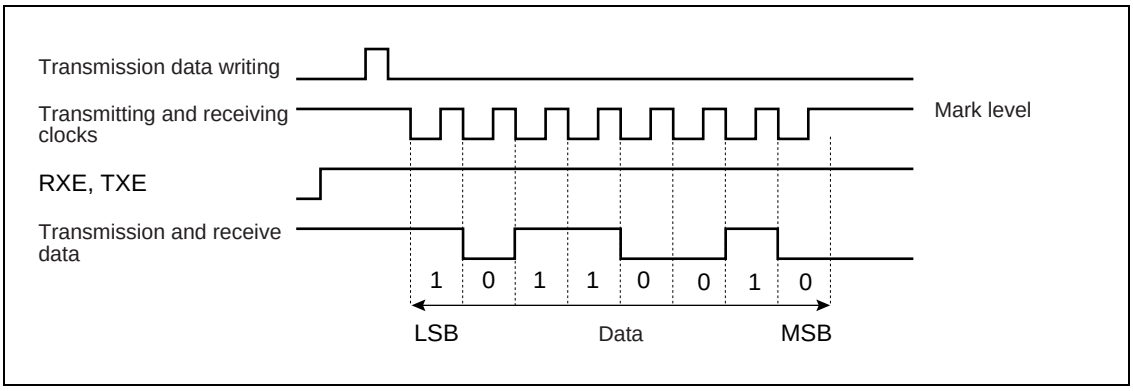


Figure 13.7-3 Transfer data format (operation mode 2)

● Clock Supply

In clock synchronous mode (I/O extended serial), as same number of clock as the number of transmission and reception bits must be supplied.

- When the internal clock (dedicated baud rate generator or internal timer) is selected, the data receiving synchronous clocks is generated automatically if data is transmitted.
- When the external clock is selected, confirm that the transmission side UART serial output data register (SODR0/1) contains data (SSR0/1: TDRE=0). Then, clocks for just “1” byte must be supplied from outside.
  The mark level (“H”) must be retained before transmission starts and after it is complete.

● Error Detection

Only overrun errors can be detected; parity and framing errors cannot be detected.

- **Initialization**

  The following shows the set values of each control register using the synchronous mode:

  [Serial mode control register (SMR0/1)]

  MD1,MD0: "10_B"
  CS2,CS1,CS0: Specify clock input using the clock selector.
  SCKE: ”1” for dedicated baud rate generator or internal timer
  “0” for clock output and external clock (clock input)
  SOE: ”1” for transmission; “0” for reception only

  [Serial control register (SCR0/1)]

  PEN: "0"
  P,SBL,A/D: These bits make no sense.
  CL: “1” (8-bit data)
  REC: “0” (the error flag is cleared for initialization.)
  RXE,TXE: At least one of the two bits is set to “1”.

  [Serial status register (SSR0/1)]

  RIE: “1” when using interrupts; “0” when using no interrupts.
  TIE: "0"

- **Starting Communication**

  Write data to the serial output data register (SODR0/1) to start communication. Temporary data must be written to SODR0/1 to start communication for reception.

- **Ending Communication**

  The RDRF flag of the serial status register (SSR0/1) is set to “1” when transmission or reception of a data frame is complete. During reception, check the overrun error flag bit (SSR0/1) to see if communication is performing normally.

13.7.3 Bidirectional Communication Function (Normal Mode)

In operation mode 0 or 2, normal 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 13.7-4 are required to operate UART in normal mode (operation mode 0 or 2).

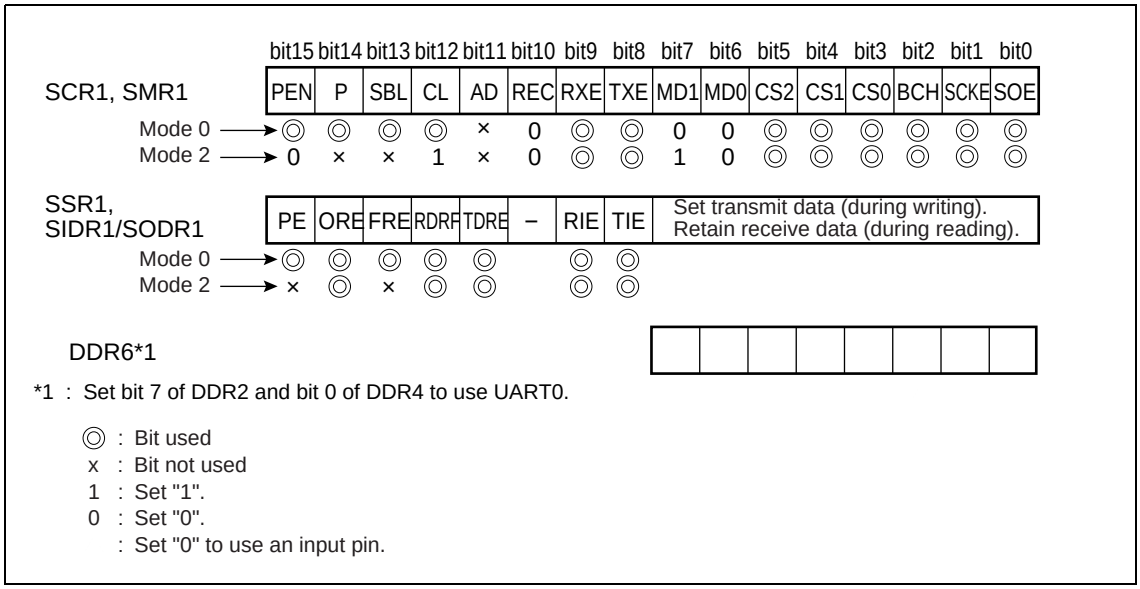


Figure 13.7-4 Settings for UART1 operation mode 0

● Inter-CPU Connection

As shown in Figure 13.7-5, interconnect two CPUs.

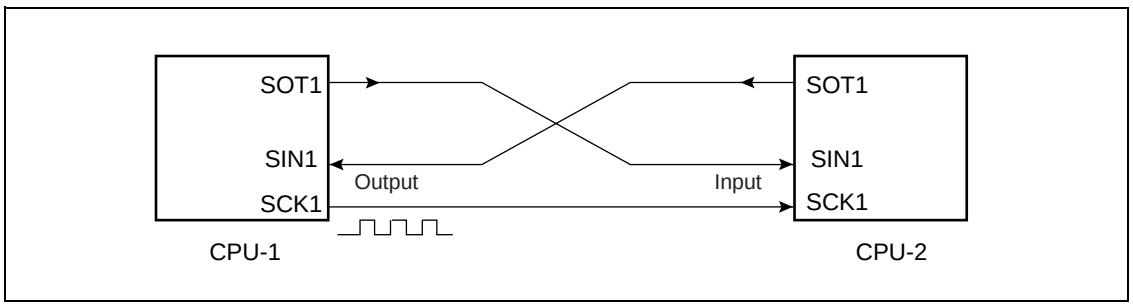


Figure 13.7-5 Connection example of UART1 bidirectional communication

● **Communication Procedure**

Communication starts from the transmitting system at an optional timing 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 13.7-6 shows an example of a bidirectional communication flowchart.

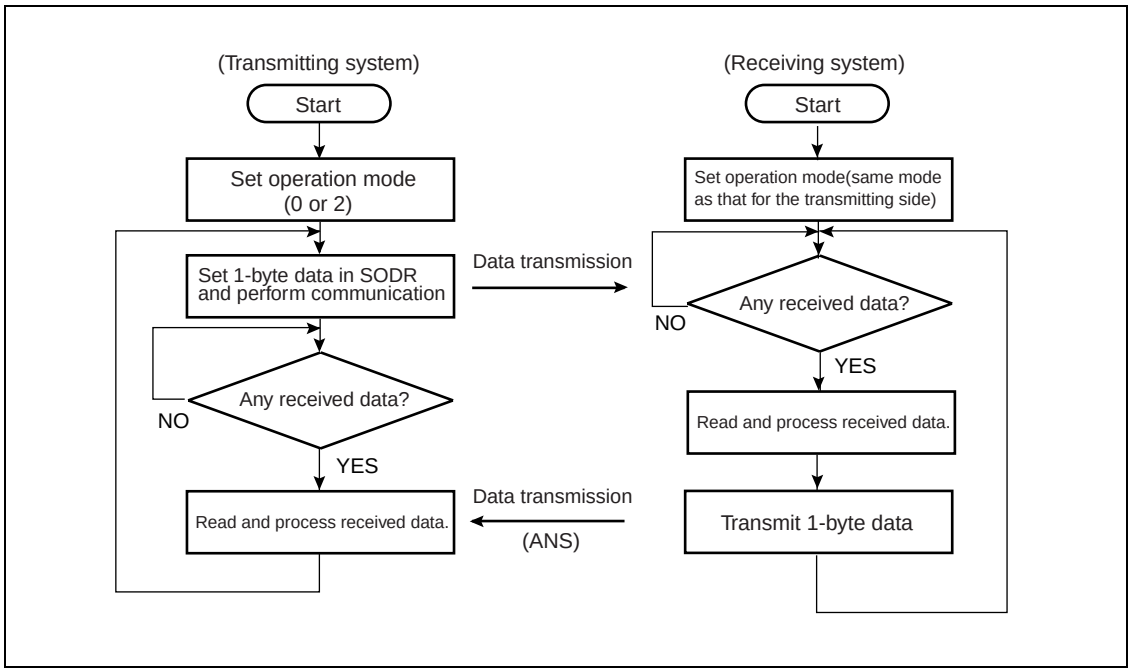


Figure 13.7-6 Example of bidirectional communication flowchart

13.7.4 Master-slave Communication Function (Multiprocessor Mode)

With UART, communication with multiple CPUs 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 13.7-7 are required to operate UART in multiprocessor mode (operation mode 1).

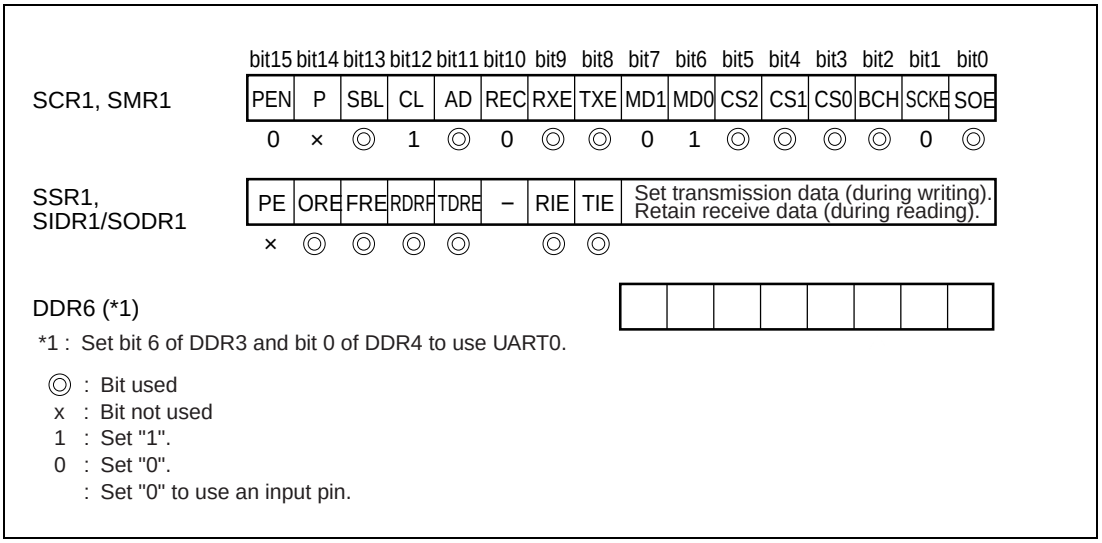


Figure 13.7-7 Settings for UART operation mode 1

● Inter-CPU Connection

As shown in Figure 13.7-8, a communication system consists of one master CPU and multiple slave CPUs connected to two communication lines. UART1 can be used only from the master CPU.

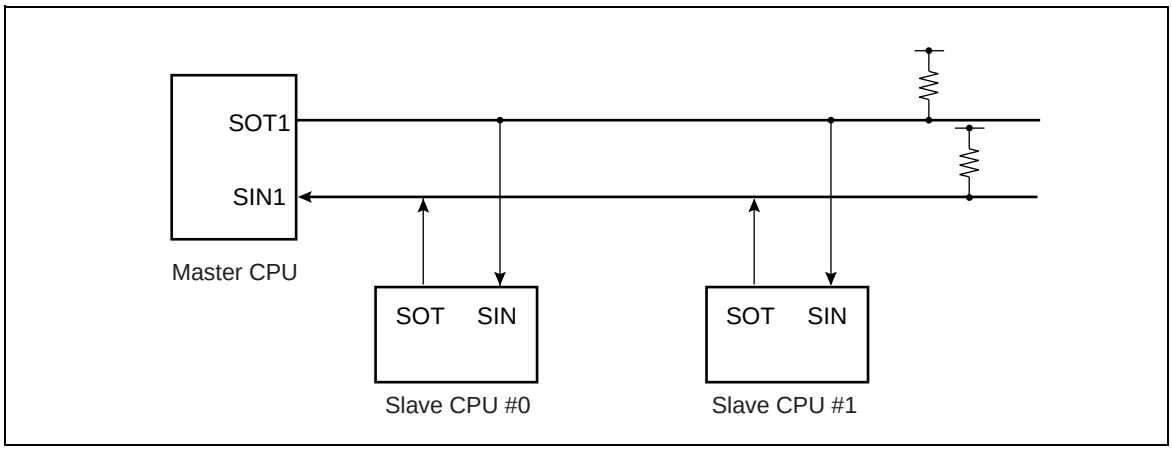


Figure 13.7-8 Connection example of UART master-slave communication

● **Function Selection**

Select the operation mode and data transfer mode for master-slave communication as shown in Table 13.7-2.

Table 13.7-2 Selection of the master-slave communication function

	Operation mode		Data	Parity	Synchronization method	Stop bit
	Master CPU	Slave CPU				
Address transmission and reception	Mode 1	–	A/D="1" + 8-bit address	None	Asynchronous	1 or 2 bits
Data transmission and reception			A/D="0" + 8-bit data			

● **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 13.7-9 shows a flowchart of master-slave communication (multiprocessor mode).

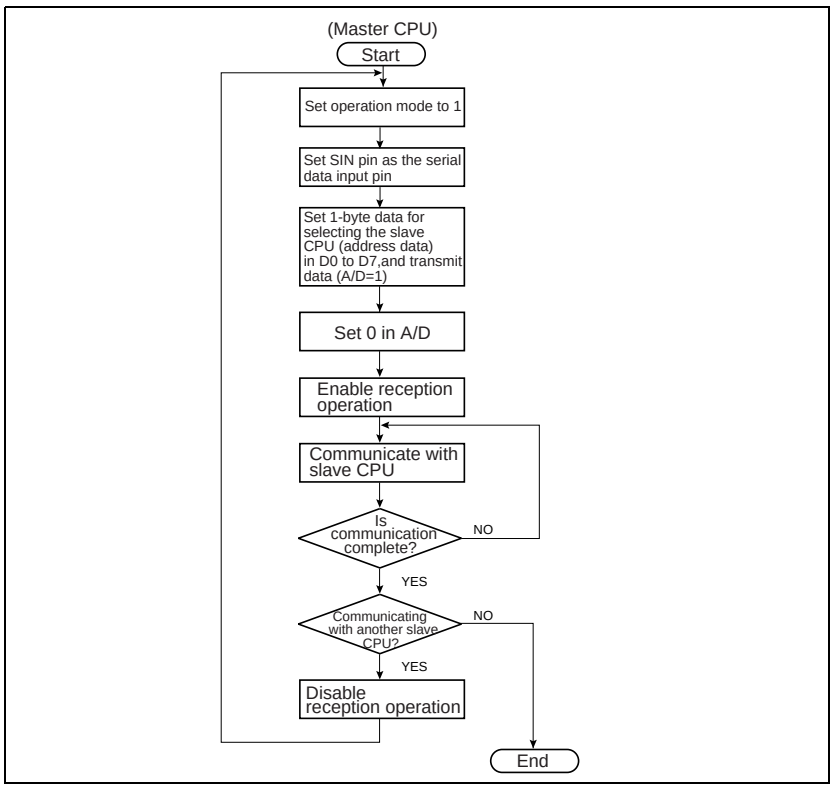


Figure 13.7-9 Master-slave communication flowchart

13.8 Notes on Using UART

Notes on using UART are given below.

■ Notes on Using UART

● Enabling Operations

In UART, the serial control register (SCR0/1) has both TXE (transmission) and RXE (reception) operation enable bits. Both transmission and reception operations must be enabled before the transfer starts because they have been disabled as the default value (initial value). The transfer can also be canceled by disabling its operation 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: TRE) 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.

Memo

13.9 Sample Program for UART

This section contains a sample program for UART.

■ Sample Program for UART

● Processing Specifications

The UART1 bidirectional communication function (normal mode) is used to perform serial transmission and reception.

- Operation mode 0, asynchronous mode, eight data bits, two stop bits, and no parity are set.
- The P60/SIN1 and P61/SOT1 pins are used for communication.
- The dedicated baud rate generator is used and the baud rate is set to about 9600 bps.
- Character 13H is transmitted from the SOT1 pin and is received using an interrupt.
- The machine clock (ϕ) is assumed to be 16 MHz.

● Coding Example

```
ICR13     EQU     0000BDH             ;   UART1 transmission interrupt control register
ICR13     EQU     0000BDH             ;   UART1 reception interrupt control register
DDR3      EQU     000013H             ;   Port-3 data direction register
SMR1      EQU     000024H             ;   Mode control register 1
SCR1      EQU     000025H             ;   Control register 1
SIDR1     EQU     000026H             ;   Input data register 1
SSR1      EQU     000026H             ;   Output data register 1
SODR1     EQU     000027H             ;   Status register 1
REC       EQU     SCR1:2              ;   Reception error flag clear bit
;-------Main program-----------------------------------------------------------------------------------------
CODE      CSEG    ABS = 0FFH
START:
;         :                           ;   Assumes that stack pointer (SP) has already been
                                      ;   initialized.
          AND     CCR,#0BFH           ;   Disables interrupts.
          MOV     I:ICR13,#00H        ;   Interrupt level 0 (highest)
          MOV     I:DDR3,#00000000B   ;   Sets SIN0 pin as input pin
          MOV     I:SMR1,#00000001B   ;   Operation mode 0 (asynchronous)
                                      ;   Uses dedicated baud rate generator (9615 bps)
                                      ;   Disables clock pulse output and enables data output.
          MOV     I:SCR1,#00010011B   ;   No parity and two stop bits
                                      ;   Clears eight data bits and reception error flag.
                                      ;   Enables transmission and reception operations.
          MOV     I:SSR1,#00000010B   ;   Disables transmission interrupts and enables reception
                                      ;    interrupts.
          MOV     I:SODR1,#13H        ;   Writes transmission data.
          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:
          MOV     A,SIDR1             ;   Reads receive data.
          CLRB    I:REC               ;   Clears reception interrupt request flag.
;         :
```

```
;               User processing
;               :
                RETI                            ;   Returns from the interrupt.
CODE    ENDS
;-------Vector setting----------------------------------------------------------------------------------------------
VECT    CSEG    ABS=0FFH
                ORG     0FF68H                  ;   Sets vector for interrupt #37 (25H).
                DSL     WARI
                ORG     0FFDCH                  ;   Sets reset vector.
                DSL     START
                DB      00H                     ;   Sets single-chip mode.
VECT    ENDS
```

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 is located between external peripherals and the F²MC-16LX CPU. It receives interrupt requests and data transfer requests from peripherals and passes them to the CPU to generate external interrupt requests or activate the extended intelligent I/O service (EI²OS).

■ DTP/external interrupt functions

The DTP/external interrupt circuit is activated by the signal supplied to a DTP/external interrupt pin. The CPU accepts the signal using the same procedure it uses for normal hardware interrupts and generates external interrupts or activates the extended intelligent I/O service (EI²OS).

If the extended intelligent I/O service (EI²OS) is disabled when an interrupt request is accepted by the CPU, the circuit executes its external interrupt function and branches to an interrupt routine. If EI²OS is enabled, the circuit executes its DTP function, which performs automatic data transfer using EI²OS and branches to an interrupt processing routine after the data transfer has been performed a specified number of times.

Table 14.1-1 provides an overview of the DTP/external interrupt circuit.

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)	
Interrupt cause	By using the request level setting register (ELVR), the level or edge to be detected can be selected for each pin.	
	Input of “H” level or “L” level or rising edge or falling edge	Input of H level or L level
Interrupt number	#25 (19_H) to #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	EI²OS is disabled (ICR: ISE = 0).	EI²OS is enabled (ICR: ISE = 1).
Processing	The circuit branches to an external interrupt processing routine.	The circuit performs automatic data transfer using EI²OS for a specified number of times and then branches to an interrupt routine.

ICR: Interrupt control register

■ Interrupt of the DTP/external interrupt circuit and EI²OS

Table 14.1-2 Interrupt of the DTP/external interrupt circuit and EI²OS

Channel	Interrupt number	Interrupt control register		Vector table address			EI²OS
		Register name	Address	Lower	Upper	Bank	
INT0/INT1	#25 (19H)	ICR07	0000B6H	FFFF98H	FFFF99H	FFFF9AH	Δ
INT2/INT3	#26 (1AH)			FFFF94H	FFFF95H	FFFF96H	
INT4/INT5	#27 (1BH)	ICR08	0000B7H	FFFF90H	FFFF91HH	FFFF92H	
INT6/INT7	#28 (1CH)			FFFF8CH	FFFF8DH	FFFF8EH	

Δ: 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**

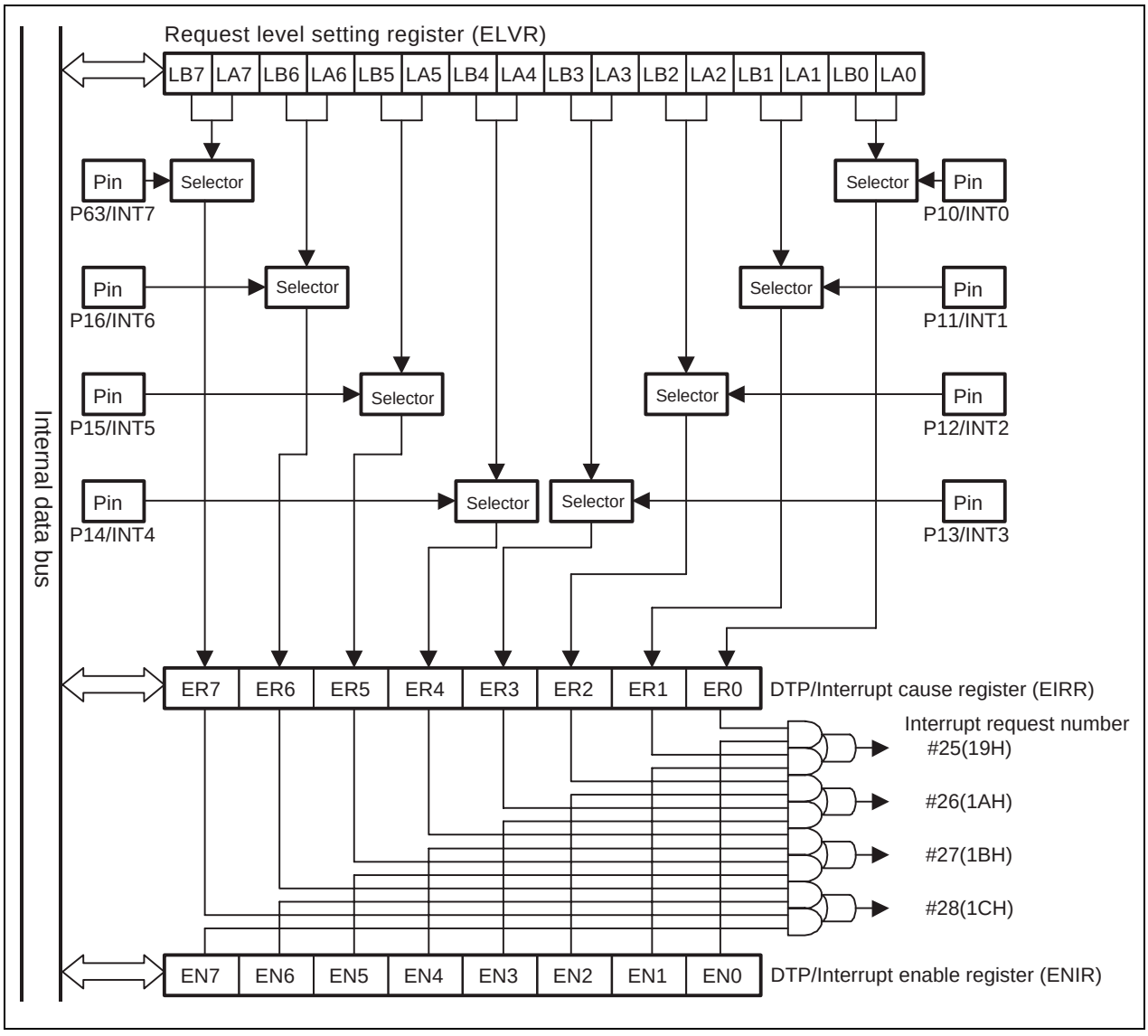


Figure 14.2-1 Block diagram of the DTP/external interrupt circuit

- **DTP/external interrupt input detection circuit**

  Upon detecting the level or edge selected for each pin by the interrupt request level setting register (ELVR), this circuit sets to “1” the ER bit of the DTP/external interrupt cause register (EIRR) that corresponds to the pin.

- **Request level setting register (ELVR)**

  This register selects the effective level or edge for each pin.

- **DTP/interrupt cause register (EIRR)**

  This register stores DTP/external interrupt causes. It contains an external interrupt request flag bit for each pin. The bit is set to “1” if a valid signal is input to the corresponding pin.

- **DTP/interrupt enable register (ENIR)**

  This register enables and disables external interrupts for each pin.

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 general ports. Table 14.3-1 lists the pin functions, I/O formats, and settings required to use the DTP/external interrupt circuit.

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	Select-able	Not pro-vided	Set the pin as an input port (DDR1: bit0 = 0)
P11/INT1					Set the pin as an input port (DDR1: bit1 = 0)
P12/INT2					Set the pin as an input port (DDR1: bit2 = 0)
P13/INT3					Set the pin as an input port (DDR1: bit3 = 0)
P14/INT4					Set the pin as an input port (DDR1: bit4 = 0)
P15/INT5					Set the pin as an input port (DDR1: bit5 = 0)
P16/INT6					Set the pin as an input port (DDR1: bit6 = 0)
P41/INT7	Port 1 input-output/external interrupt input		Not pro-vided		Set the pin as an input port (DDR6: bit3 = 0)

■ **Block diagram of the DTP/external interrupt circuit pins**

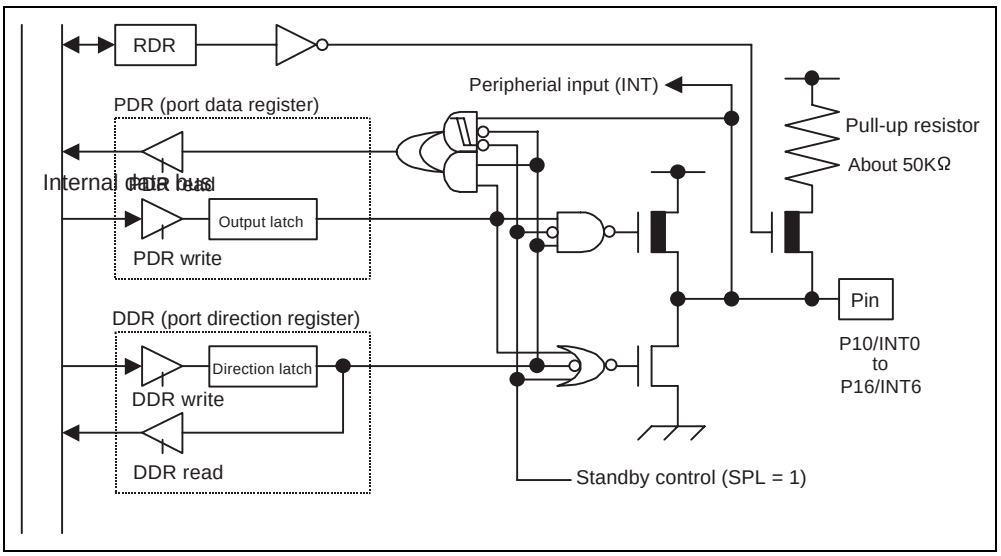


Figure 14.3-1 Block diagram of the DTP/external interrupt circuit pins (For P10/INT0 ~ P16/INT6 only)

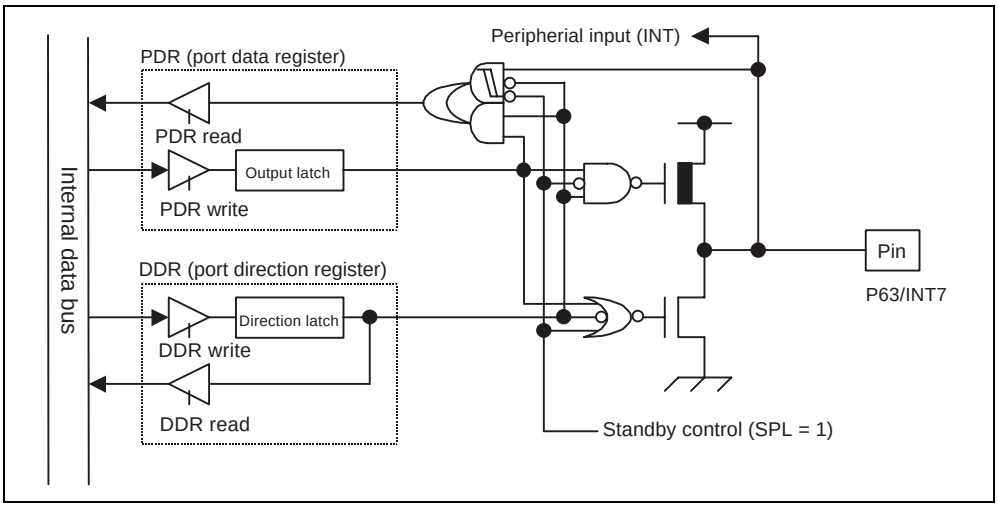


Figure 14.3.2 Block diagram of the DTP/external interrupt circuit pins (For P63/ INT7 only)

14.4 DTP/External Interrupt Circuit Registers

This section describes DTP/external interrupt circuit registers.

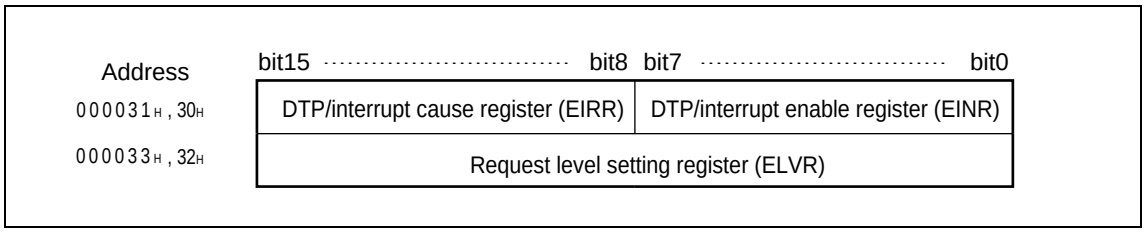


Figure 14.4-2 DTP/external interrupt circuit registers

14.4.1 DTP/interrupt cause register (EIRR)

The DTP/interrupt cause register (EIRR) stores and clears interrupt causes.

■ **DTP/interrupt cause register (EIRR)**

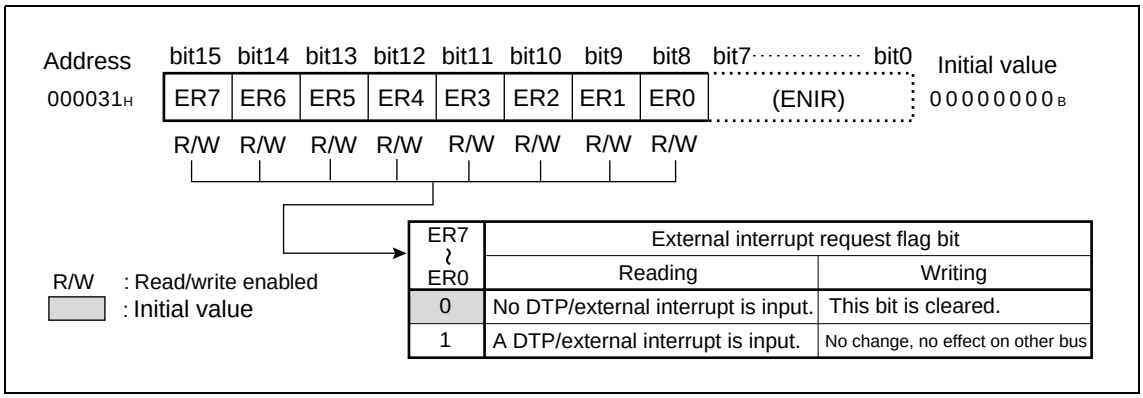


Figure 14.4-2 DTP/interrupt cause register (EIRR)

Table 14.4-1 Function description of each bit of the DTP/interrupt cause register (EIRR)

Bit name		Function
bit15 bit14 bit13 bit12 bit11 bit10 bit9 bit8	ER7 to ER0: External interrupt request flag bits	• Each of these bits is set to “1” if a signal with the edge or level selected by the request level setting register (ELVR) is input to the DTP/external interrupt pin (stores an interrupt cause). • If these bits and corresponding bits EN7 to EN0 of the DTP/interrupt enable register (ENIR) are “1”, an interrupt request is output to the CPU. • Writing “0” to this bit clears the bit. Writing “1” to this bit does not change the bit value and has no effect on other bits. **<Caution>** If more than one external interrupt request enable bit is set to “1” (ENIR: EN7 to EN0 = 1), clear only the bit that caused the CPU to accept an interrupt (ER3 to ER0 that is set to “1”). Do not clear the other bits without reason. **<Reference>** When the extended intelligent I/O service (EI²OS) is activated, the corresponding external interrupt request flag bit is automatically cleared when the transfer of one data ends.

14.4.2 DTP/interrupt enable register (ENIR)

The DTP/interrupt enable register (ENIR) enables and disables the output of interrupt requests to the CPU.

■ **DTP/interrupt enable register (ENIR)**

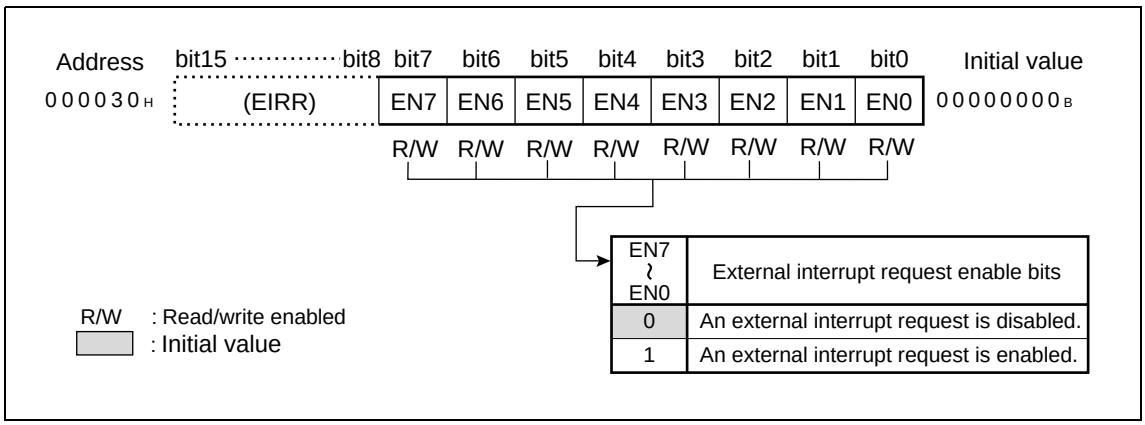


Figure 14.4-3 DTP/interrupt enable register (ENIR)

Table 14.4-2 Function description of each bit of the DTP/interrupt enable register (ENIR)

Bit name		Function
bit7 bit6 bit5 bit4 bit3 bit2 bit1 bit0	EN7 to EN0: External interrupt request enable bits	Each of these bits enables and disables interrupt requests to the CPU. If these bits and corresponding bits ER7 to ER0 of the DTP/interrupt cause register (EIRR) are “1”, an interrupt request is output to the CPU. [Reference] • To use a DTP/external interrupt pin, write “0” to the corresponding bit of the port direction register to set the pin as an input port. • The states of the DTP/external interrupt pins can be read directly using the port data register regardless of the states of external interrupt request enable bits. • Bits ER7 to ER0 of the DTP/interrupt cause register (EIRR) are set to “1” if an interrupt cause is detected regardless of the values of external interrupt request enable bits.

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	External interrupt request enable bit
P63/INT7	#28 (1CH)	ER7	EN7
P16/INT6		ER6	EN6
P15/INT5	#27 (1BH)	ER5	EN5
P14/INT4		ER4	EN4
P13/INT3	#26 (1AH)	ER3	EN3
P12/INT2		ER2	EN2
P11/INT1	#25 (19H)	ER1	EN1
P10/INT0		ER0	EN0

14.4.3 Request level setting register (ELVR)

The request level setting register (ELVR) selects the level or edge of the signal input to each DTP/external interrupt pin that is to be detected as a DTP/external interrupt cause.

■ **Request level setting register (ELVR)**

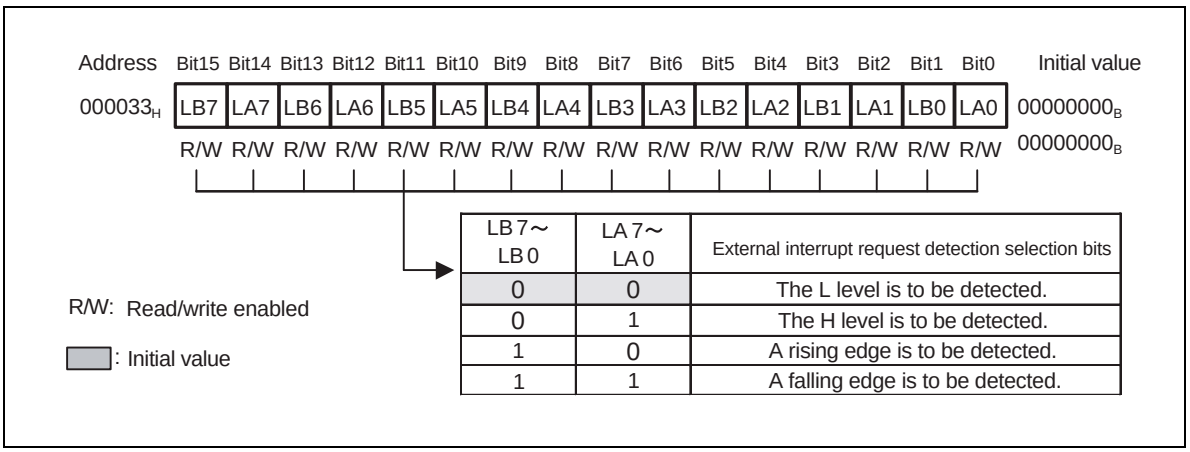


Figure 14.4-4 Request level setting register (ELVR)

Table 14.4-4 Function description of each bit of the request level setting register (ELVR)

Bit name		Function
Bit15 Bit14 Bit13 Bit12 Bit11 Bit10 Bit9 Bit8 Bit7 Bit6 Bit5 Bit4 Bit3 Bit2 Bit1 Bit0	LB7 to LB0 and LA7 to LA0: Request detection selection bits	• Each of these bits selects the level or edge of the signal input to the DTP/external interrupt pin to be detected as a DTP/external interrupt cause. • Two bits are assigned to each pin. [Reference] If the selected detection signal is input to a DTP/external interrupt pin, the external interrupt request flag bit is set to “1” regardless of the settings of the DTP/interrupt enable register (ENIR).

Table 14.4-5 Correspondence between request level setting register (ELVR) and each channel

DTP/external interrupt pin	Interrupt number	Bit name
P63/INT7	#28 (1CH)	LB7, LA7
P16/INT6		LB6, LA6
P15/INT5	#27 (1BH)	LB5, LA5
P14/INT4		LB4, LA4
P13/INT3	#26 (1AH)	LB3, LA3
P12/INT2		LB2, LA2
P11/INT1	#25 (19H)	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 shows the settings required to operate the DTP/external interrupt circuit.

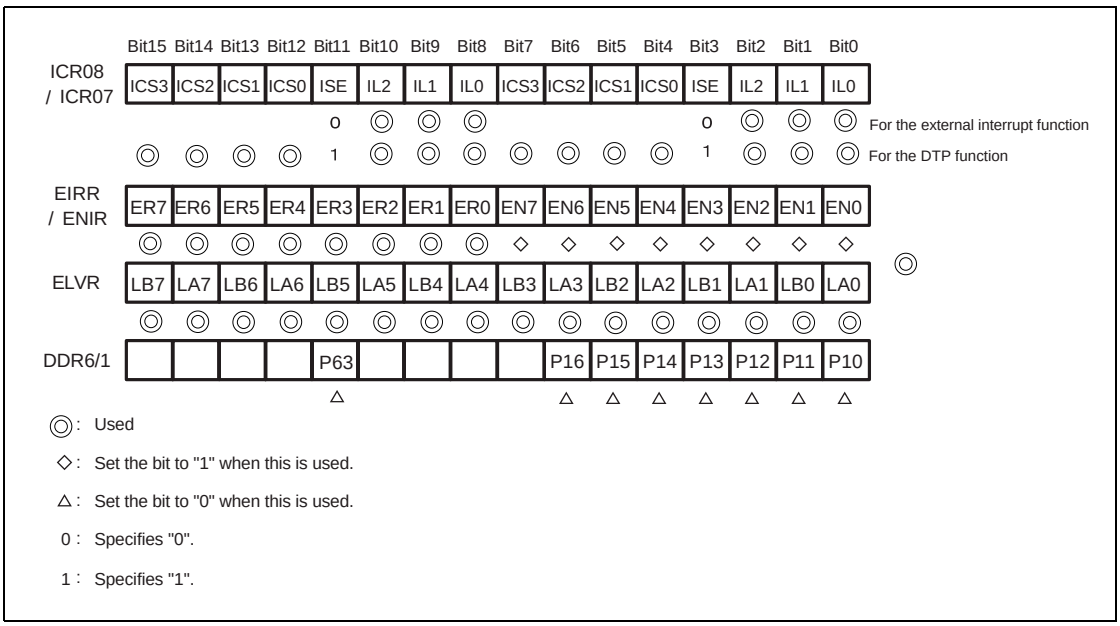


Figure 14.5-1 DTP/external interrupt circuit

Set the DTP/external interrupt circuit registers with the following procedure:

1. Clear the target bit of the DTP/interrupt enable register (ENIR) to disable interrupts.
2. Set the target bit of the request level setting register (ELVR).
3. Clear the target bit of the DTP/interrupt cause register (EIRR).
4. Set the target bit of the DTP/interrupt enable register (ENIR) to enable interrupts.

The procedure for setting the DTP/external interrupt circuit registers must start with disabling the output of external interrupt requests (ENIR: EN7 to EN0 = 0). Before the output of external interrupt requests can be enabled (ENIR: EN7 to EN0 = 1), the corresponding interrupt request flag bits must be cleared (EIRR: ER7 to ER0 = 0).

This is in order to avoid interrupt requests from being generated accidentally while the registers are being set.

- **Switching between the external interrupt function and the DTP function**

  Switching between the external interrupt function and the DTP function is accomplished by the ISE bit of the corresponding interrupt control register (ICR). If the ISE bit is 1, the extended intelligent I/O service (EI^{2}OS) is enabled and the circuit executes its DTP function. If it is “0”, EI^{2}OS is disabled and the circuit executes the its external interrupt function.

  <Check>

  If multiple interrupt requests are assigned to a single ICR register, the interrupt level (IL2 to IL0) is common to all of the interrupt requests. As a rule, when one interrupt request uses EI^{2}OS, the other interrupt requests cannot use it.

■ Operation of the DTP/external interrupt circuit

Table 14.5-1 shows the control bits and interrupt causes 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
Interrupt request flag bit	EIRR: ER7 to ER0
Interrupt request enable bit	ENIR: EN7 to EN0
Interrupt cause	Input of an effective edge or level to pin INT7 to INT0

When DTP/external input requests are set, an interrupt request will be generated to the interrupt controller whenever an interrupt cause indicated in the request level setting register (ELVR) is received at the corresponding pin. If the ISE bit is “0”, the interrupt processing microprogram is executed. If it is “1”, the extended intelligent I/O service handling (DTP handling) microprogram is executed.

Figure 14.5-2 shows the operation of the DTP/external interrupt circuit.

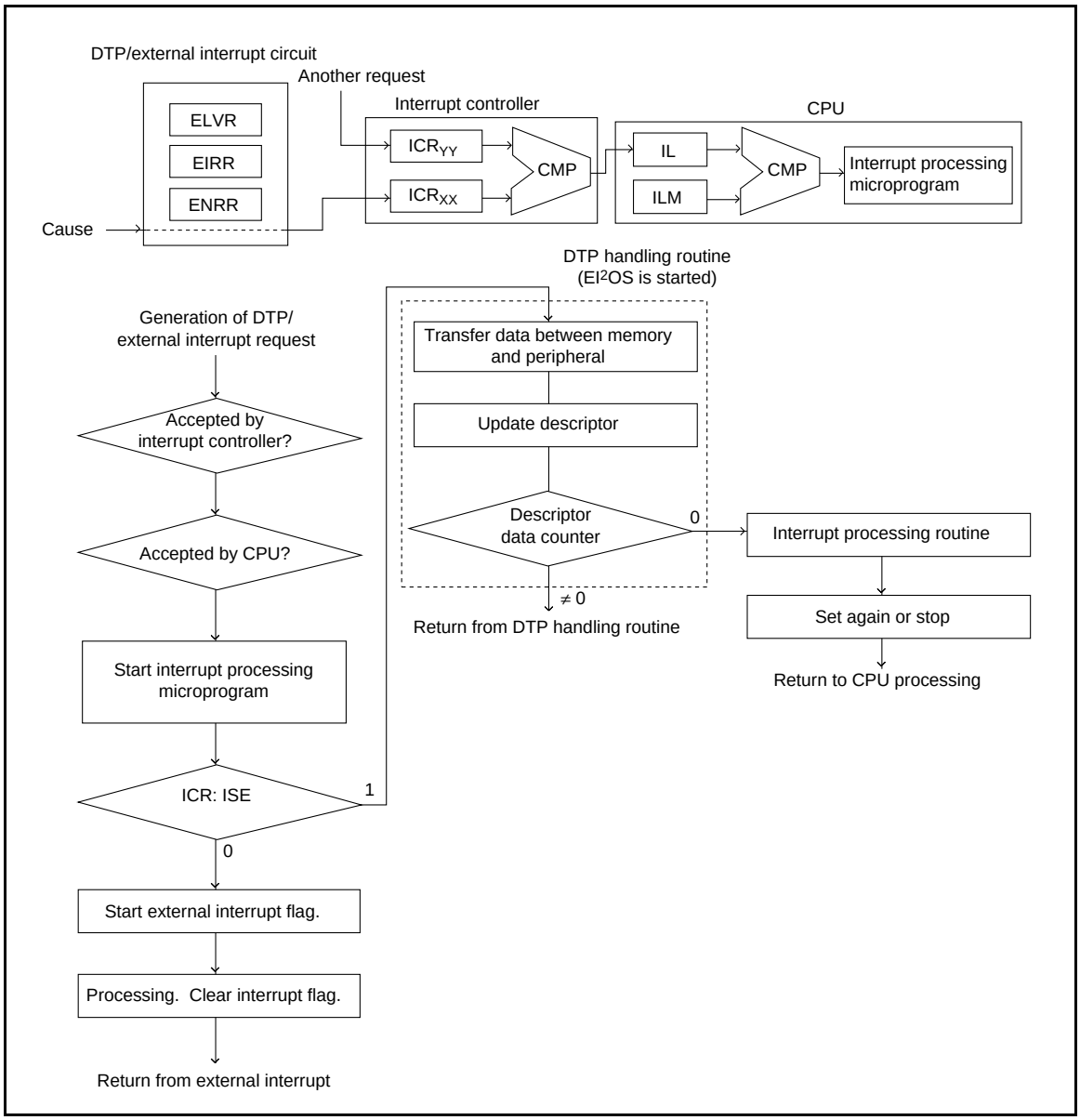


Figure 14.5-2 Operation of the DTP/external interrupt circuit

14.5.1 External interrupt function

The DTP/external interrupt circuit has an external interrupt function that generates an interrupt request when a selected signal level is input to a DTP/external interrupt pin.

■ External interrupt function

If the edge or level selected for a DTP/external interrupt pin by the request level setting register (ELVR) is detected at that pin, the corresponding external interrupt flag bit of the DTP/interrupt cause register (EIRR:ER7 to ER0) is set to “1”. If, in this state, the corresponding interrupt request enable bit of the DTP/interrupt enable register is set to “1” to enable interrupts (ENIR: EN7 to EN0 = 1), the interrupt cause is reported to the interrupt controller. The interrupt controller checks the value of the interrupt level (ICR: IL2 to IL0) in relation to those of the interrupt requests from other peripheral functions, the interrupt priority, etc. The CPU checks the value of the interrupt level mask register (PS: ILM2 to ILM0) and the interrupt level, the interrupt enable bit (PS: CCR: 1), etc. When the interrupt request is accepted by the CPU, the CPU executes an internal interrupt processing routine (microprogram) and branches to the interrupt processing routine. In the interrupt processing routine, 0 must be written to the corresponding interrupt request flag bit to clear the interrupt request.

<Check>

- An ER bit is set to “1” if a DTP/external interrupt cause is generated, regardless of the state of the corresponding EN bit.
- When the interrupt routine is activated, the ER bit that caused the routine to be activated must be cleared. If the ER bit is kept at “1”, control cannot return from the interrupt. Only clear the flag bit that caused the interrupt; do not clear the other bits without reason.

14.5.2 DTP function

The DTP/external interrupt circuit has a DTP function that detects a signal supplied to a DTP/external interrupt pin from an external peripheral and activates the extended intelligent I/O service.

■ Operation of the DTP function

The DTP function detects a data transfer request signal from an external peripheral to automatically transfer data between memory and the peripheral.

The extended intelligent I/O service (EI^{2}OS) is activated by the external interrupt function using level detection. The operation of the DTP function is the same as that of the external interrupt function up to the point that the CPU accepts an interrupt request. If the operation of EI^{2}OS is enabled (ICR: ISE = 1), EI^{2}OS is activated to start data transfer when an interrupt request is accepted. When the transfer of one data unit ends, the descriptor is updated and the interrupt request flag bit is cleared to wait for the next request from the pin. When the entire transfer using EI^{2}OS is completed, control is transferred to the interrupt processing routine.

The external peripheral signal must be removed only the level of the data transfer request signal (DTP external interrupt cause) within three cycles of the first transfer.

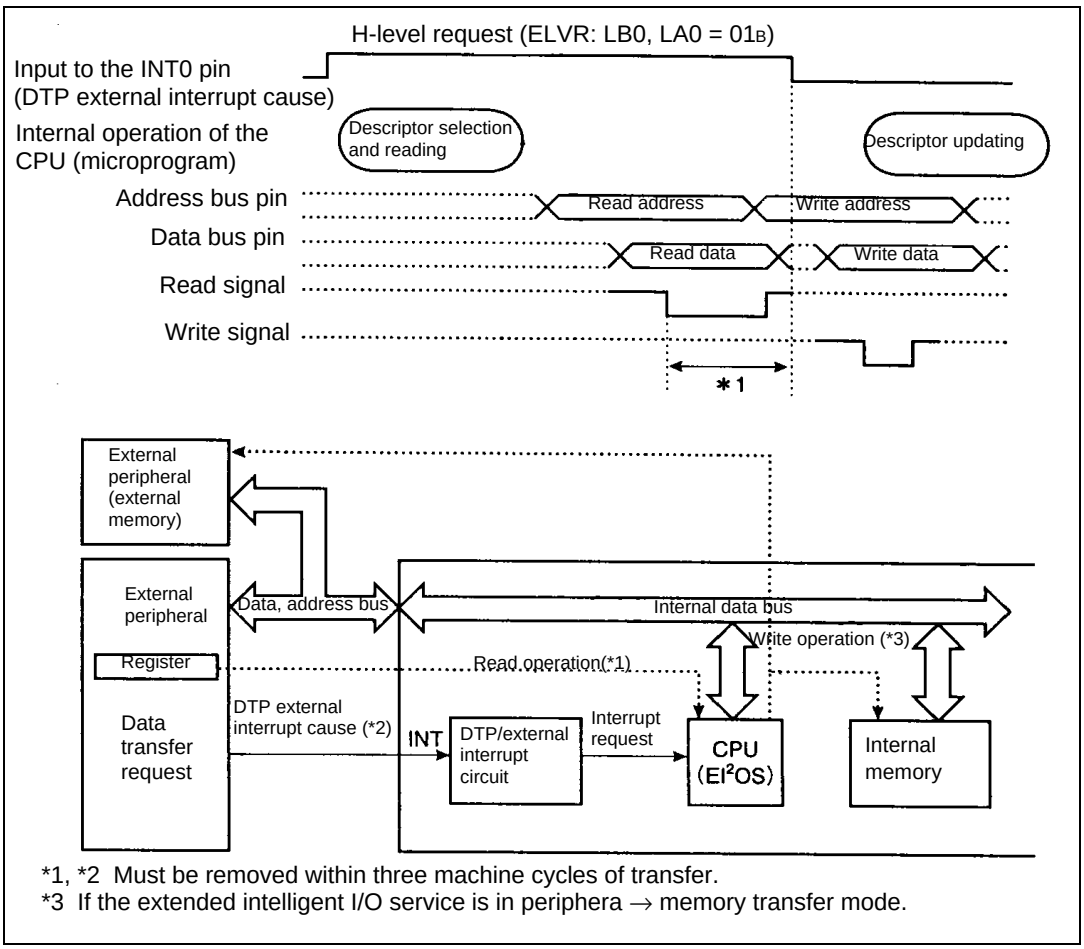


Figure 14.5-3 Example of interfacing to the external peripheral

Memo

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, external peripherals must be able to clear data transfer requests automatically in response to transfer operations. If a transfer request is not removed within three machine cycles of the start of transfer, the DTP/external interrupt circuit interprets the request as another transfer request.

● Input polarities of external interrupts

- If the request level setting register (ELVR) is set so that an edge is detected, the pulse width must be at least three machine cycles for the edge to be detected.
- If the register is set for level detection, and the level to be detected as an interrupt cause is input, cause F/F in the DTP/interrupt cause register (EIRR) is set to “1” to store the cause, as shown in Figure 14.6-1. Even if the cause is removed, the request to the interrupt controller remains active provided the output of interrupt requests is enabled. Thus, to cancel the request to the interrupt controller, clear the external interrupt request flag bit and cause F/F, as shown in Figure 14.6-2.

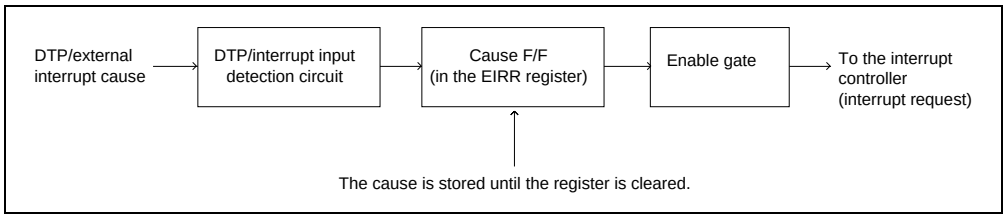


Figure 14.6-1 Clearing the cause retention circuit when a level is specified

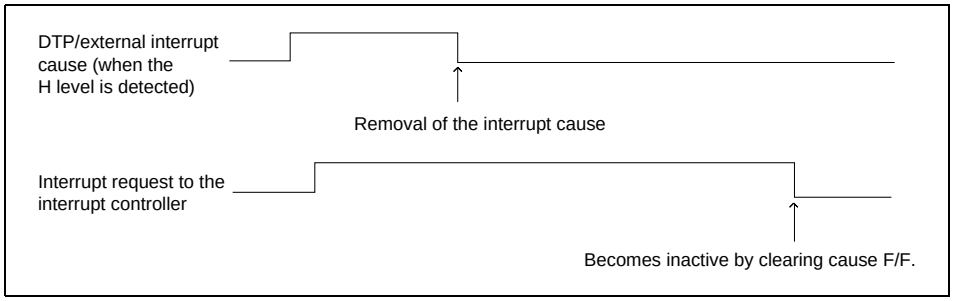


Figure 14.6-2 DTP/external interrupt cause and interrupt request when the output of interrupt requests is enabled

- **Notes about interrupts**

  When the external interrupt function is used, control cannot return from the interrupt processing routine if the external interrupt request flag bit is “1” and the output of interrupt requests is enabled. In the interrupt processing routine, the external interrupt request flag bit must be cleared. (When the DTP function is used, EI^2OS automatically clears the bit.) For level detection, the external interrupt request flag bit is set again as soon as it is cleared if the level assumed as an interrupt cause continues to be input. Either disable the output of interrupt requests or remove the interrupt cause, if required.

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   0000B6H            ;  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-----------------------------------------------------------------------------------------
CODE    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  ERP                 ;  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     0000B6H         ;  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         ;  EI2OS 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:DDR1,#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           ;  Specifies port 0 (PDR0) as the transfer destination
        MOV     IOAH,#00H           ;  address pointer.
        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 #11 (0BH).
        DSL     WARI
        ORG     0FFDCH              ;  Sets reset vector.
        DSL     START
        DB      00H                 ;  Sets single-chip mode.
VECT    ENDS
        END     START
```

Memo

CHAPTER 15 DELAYED INTERRUPT GENERATOR MODULE

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

15.1 Overview of the Delayed Interrupt Generator Module

The delayed interrupt generator module generates interrupts for task switching. By using this module, software can issue and cancel interrupt requests for the F²MC-16LX CPU.

■ Block diagram of the delayed interrupt generator module

Figure 15.1-1 shows the block diagram of the delayed interrupt generator module.

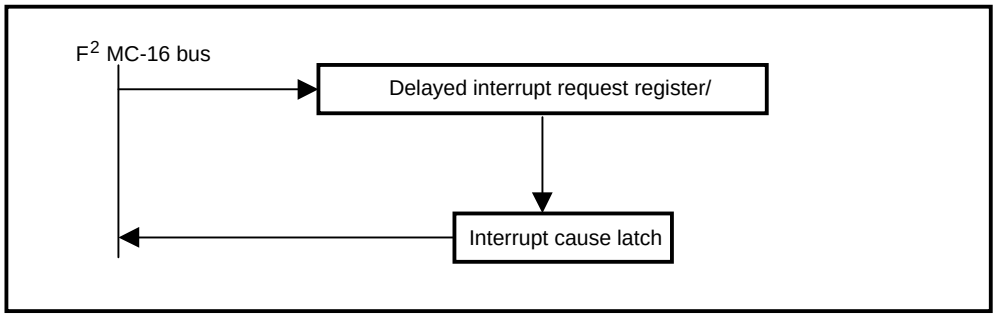


Figure 15.1-1 Block diagram of the delayed interrupt generator module

■ Delayed interrupt generator module register

The configuration of the delayed interrupt generator module (delayed interrupt request register [DIRR]) is as follows:

bit	15	14	13	12	11	10	9	8	Initial value
DIRR address :00009FH	-	-	-	-	-	-	-	R0	0R
								R/W	

The cause is cleared at reset.

The DIRR controls the generation or clearing of a delayed interrupt request. Writing “1” to this register generates a delayed interrupt request. Writing “0” to this register clears the delayed interrupt request. The cause is cleared at reset. Both “0” and “1” may be written to the reserved bit area. However, the set bit and clear bit instructions should be used to access this register to prepare for future expansion.

15.2 Operation of the Delayed Interrupt Generator Module

When software causes the CPU to write “1” to the relevant bit of DIRR, the request latch in the delayed interrupt generator module is set and an interrupt request is generated to the interrupt controller.

■ Operation of the Delayed Interrupt Generator Module

When software causes the CPU to write “1” to the relevant bit of DIRR, the request latch in the delayed interrupt generator module is set and an interrupt request is generated to the interrupt controller. If the priority of other interrupt requests is lower than that of this interrupt or no other interrupt request is generated, the interrupt controller generates an interrupt request to the F²MC-16LX CPU. The F²MC-16LX CPU compares the ILM bit of the internal CCR register and the interrupt request. When the request level is higher than that of the ILM bit, the CPU starts the hardware interrupt processing microprogram immediately after execution of the current instruction ends. As a result, the interrupt processing routine for this interrupt is executed. This interrupt cause is cleared and task switching is done by writing “0” to the relevant bit of DIRR in the interrupt processing routine. Figure 15.2-1 shows the operation of the delayed interrupt generator module.

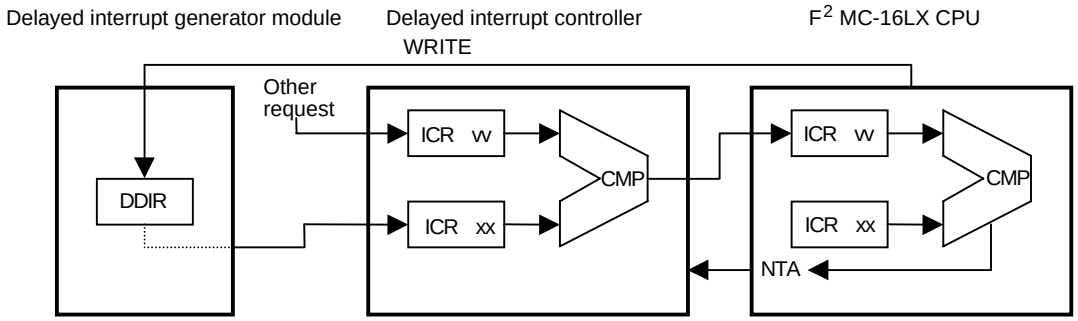


Figure 15.2-1 Operation of the delayed interrupt generator module

■ Note on using the delayed interrupt generator module

● Delayed interrupt request latch

This latch is set by writing “1” to the relevant bit of DIRR and cleared by writing “0” to the same bit. Note that interrupt processing is restarted the moment control returns from interrupt processing unless software is created to clear the cause in the interrupt processing routine.

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

Using the RC-type successive approximation conversion method, the 8/10-bit A/D converter converts an analog input voltage into a 10-bit or 8-bit digital value.
An input signal is selected from eight channels for analog input pins. The conversion can be activated by software, an internal clock, and 16-bit free-running timer zero detection.

■ Functions of the 8/10-bit A/D converter

The converter converts the analog voltage at an analog input pin (input voltage) to a digital value. The converter has the following features:

- The minimum conversion time is 6.13 µs (for a machine clock of 16 MHz; includes the sampling time).
- The minimum sampling time is 3.75 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.
- Up to eight channels for analog input pins can be selected by a program.
- At the end of A/D conversion, an interrupt request can be generated and EI²OS can be activated.
- In the interrupt-enabled state, the conversion data protection function prevents any part of the data from being lost through continuous conversion.
- The conversion can be activated by software, 16-bit reload timer 1 (rising edge), and 16-bit free-running timer zero detection edge.

Table 16.1-1 lists three types of conversion modes.

Table 16.1-1 8/10-bit A/D converter conversion modes

	Single conversion	Scan conversion
Single conversion mode	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). When a channel has been converted, the converter is put on standby for the next activation.

■ 8/10-bit A/D converter interrupts and EI²OS

Table 16.1-2 8/10-bit A/D converter interrupts and EI²OS

Interrupt No.	Interrupt control register		Vector table address			EI²OS
	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 (ADCS1, 2)**
- **A/D data register (ADCR1, 2)**
- **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**

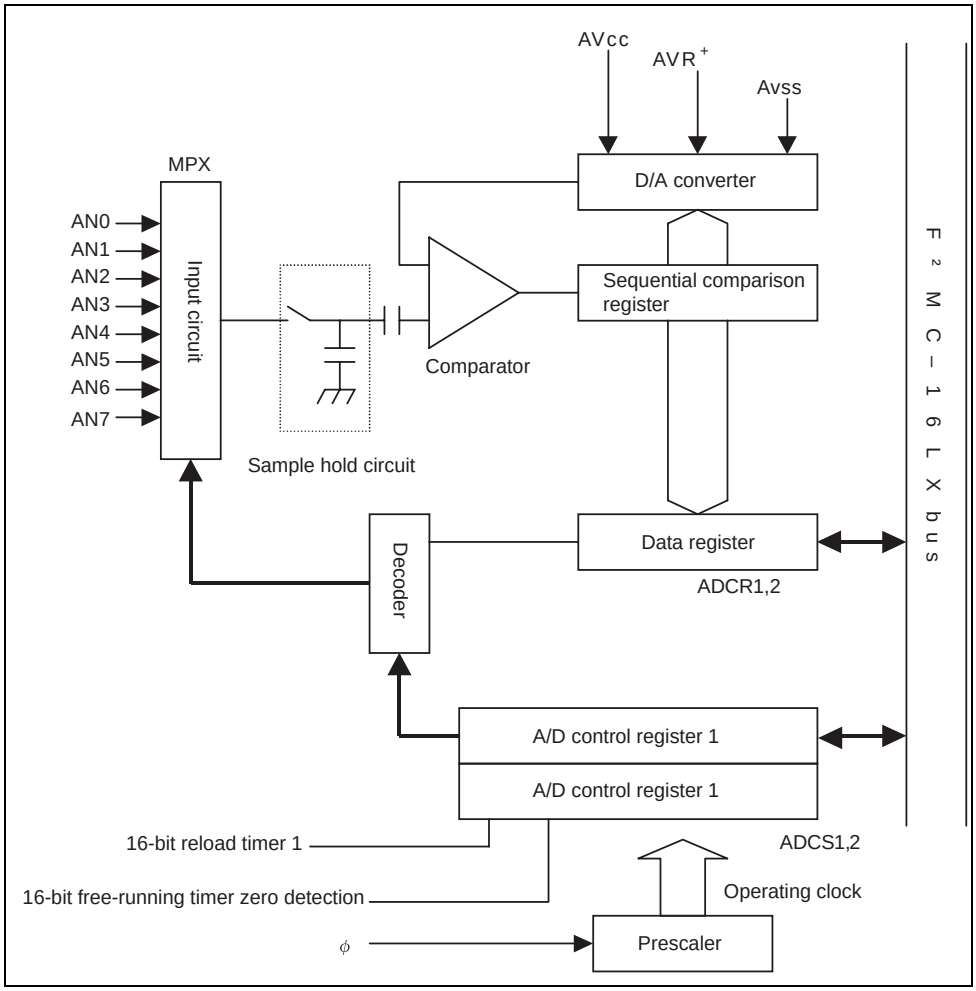


Figure 16.2-1 Block diagram of the 8/10-bit A/D converter

- **A/D control status register (ADCS1, 2)**

  This register selects activation by software or another activation trigger, the conversion mode, and the A/D conversion channel. It also enables or disables interrupt requests, checks the interrupt request status, and indicates whether the conversion has halted or is in progress.

- **A/D data register (ADCR1,2)**

  This register holds the result of A/D conversion and selects the resolution for 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 selects the analog input pin to be used based on the settings of the ANE0 to ANE2 bits and ANS0 to ANS2 bits of the A/D control status register (ADCS1).

- **Analog channel selector**

  This circuit selects the pin to be used from eight analog input pins.

- **Sample hold circuit**

  This circuit holds the input voltage of the channel selected by the analog channel selector. It samples and holds the input voltage obtained immediately after the activation of A/D conversion. This circuit protects the A/D conversion from any variations in the input voltage during approximation.

- **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 held 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 (ADCR1, 2) 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 general ports. Table 16.3-1 lists the pin functions, I/O formats, and settings required to use the 8/10-bit A/D converter.

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	I/O port 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 port 5 as an input port (DDR5: bits 0 to 7 = 0). Set port 5 as an analog input port (ADER: bits 0 to 7 = 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

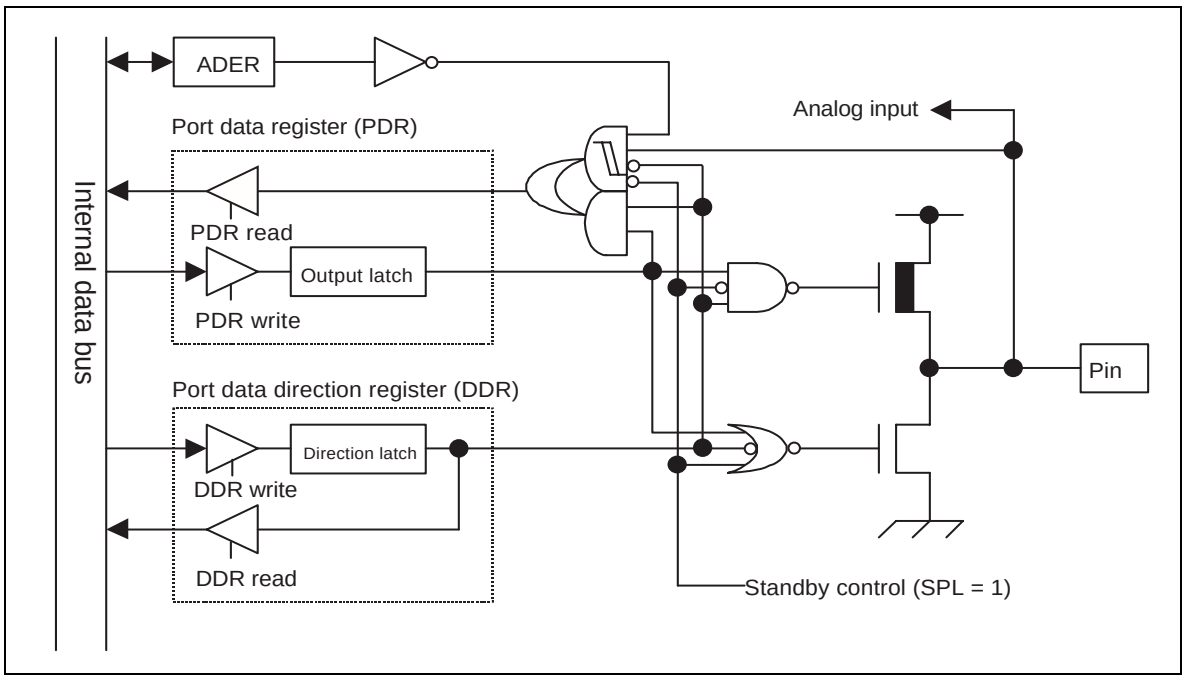


Figure 16.3-1 Block diagram of the P50/AN0 to P57/AN7 pins

<Notes>

- To use a pin as an input port, set the corresponding bit of the DDR5 register to “0”, then add a pull-up resistor to the external pin. Set the corresponding bit of the ADER register to “0”.
- To use the pin as an analog input pin, set the corresponding bit of the ADER register to “1”. The value read from the PDR5 register is “0”.

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**

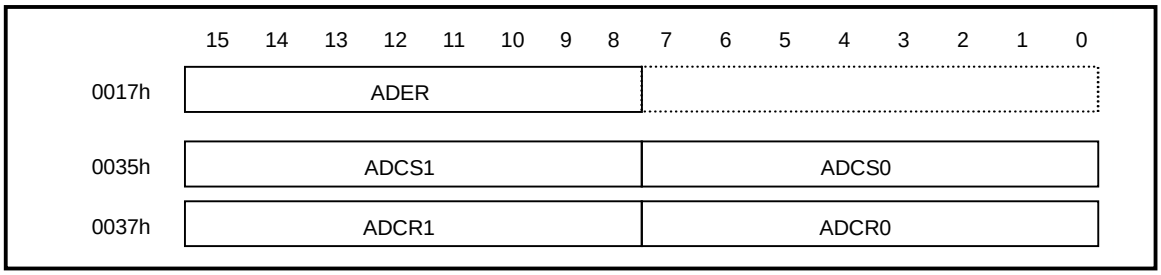


Figure 16.4-1 8/10-bit A/D converter registers

16.4.1 A/D control status register 1 (ADCS1)

A/D control status register 1 (ADCS1) selects activation by software or activation trigger, enables or disables interrupt requests, and indicates interrupt request status and whether conversion is halted or in progress.

■ Upper bits of the A/D control status register (ADCS1)

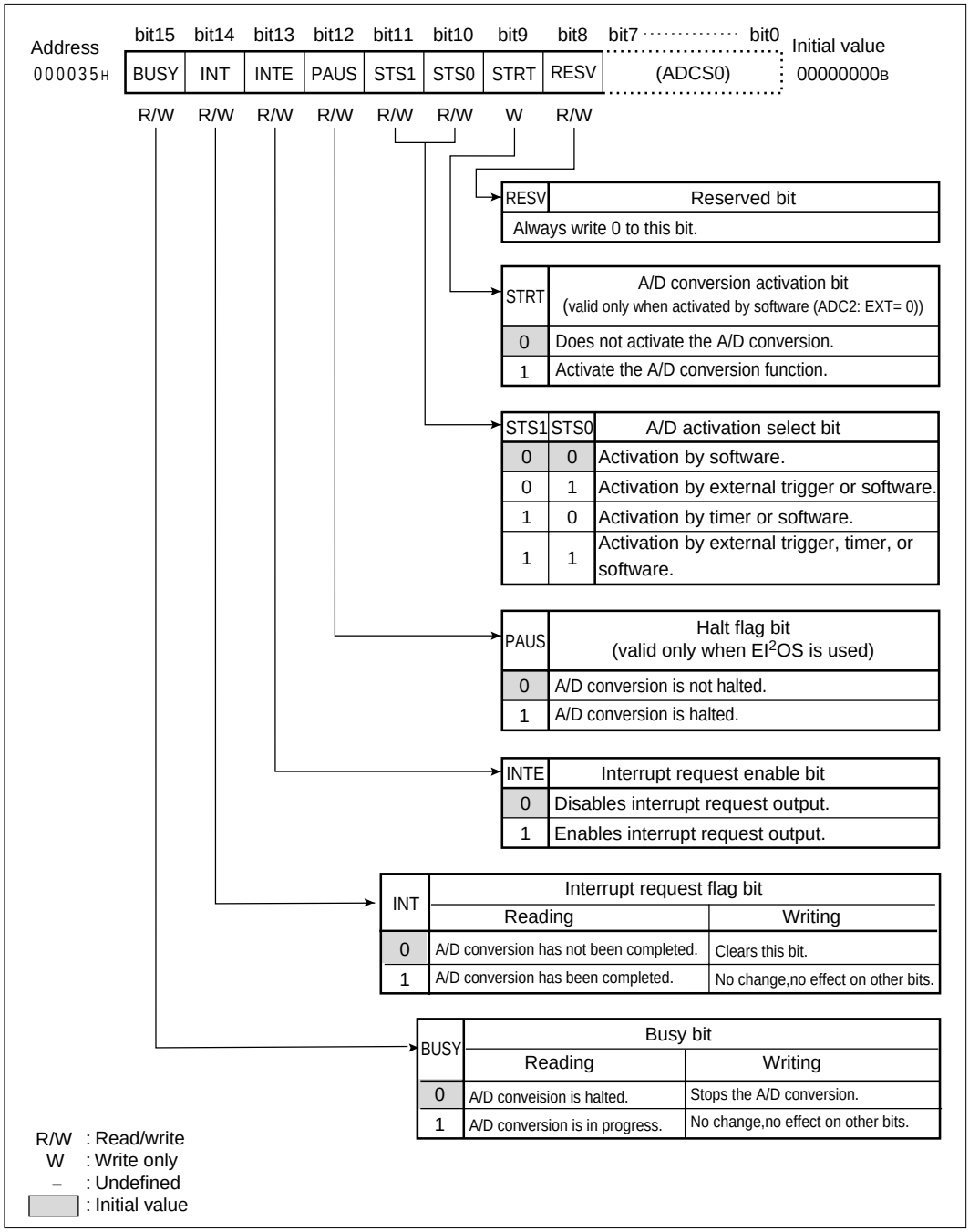


Figure 16.4-2 A/D control status register 1 (ADCS1)

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. • If the value read from this bit is “0”, A/D conversion has halted. If the read value is “1”, A/D conversion is in progress. • Writing “0” to this bit forces the A/D conversion to stop. Writing “1” to this bit does not change the bit value and has no effect on other bits. **<Caution>** Never select forced stop (BUSY = 0) and software activation (STRT = 1) simultaneously.
bit14	INT: Interrupt request flag bit	• When A/D conversion data is stored in the A/D data register, this bit is set to “1”. • When both this bit and the interrupt request enable bit (ADCS: INTE) are “1”, an interrupt request is generated. If EI²OS has been enabled, it is activated. • Writing “0” to this bit clears the bit. Writing “1” to this bit does not change the bit value and has no effect on other bits. • When EI²OS is activated, this bit is cleared. **<Caution>** When clearing this bit by writing “0” it, do so only while the A/D converter is not operating.
bit13	INTE: Interrupt request enable bit	• This bit enables or disables interrupt output to the CPU. • When both this bit and the interrupt request flag bit (ADCS: INT) are set to “1”, an interrupt request is generated. • When EI²OS is used, set this bit to “1”.
bit12	PAUS: Halt flag bit	• When A/D conversion stops temporarily, this bit is set to “1”. • This A/D converter has just one A/D data register. In continuous conversion mode, if a conversion result were written before the previous conversion result was read by the CPU, the previous result would be lost. When continuous conversion mode is selected, the program must be written so that the conversion result is automatically transferred to memory by EI²OS each time a conversion is completed. This bit also protects against multiple interrupts preventing the completion of conversion data transfer before the next conversion . When a conversion is completed, this bit is set to “1”. This status is maintained until EI²OS finishes transferring the contents of the data register. Meanwhile, the A/D conversion is halted so that no conversion data can be stored. When EI²OS completes the transfer, the A/D converter automatically resumes the conversion. **<Caution>** This bit is valid only when EI²OS is used.

(***continued***)

Bit name		Function
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. <Caution> • 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. <Caution> Never select forced stop (BUSY = 0) and software activation (STRT = 1) simultaneously.
bit8	RESV: Reserved bit	<Caution> Always write “0” to this bit.

16.4.2 A/D control status register 0 (ADCS0)

A/D control status register 0 (ADCS0) selects the conversion mode and A/D conversion channel.

■ A/D control status register 0 (ADCS0)

	bit15	----------	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
Address 000034_H	(ADCS: H)			MD1	MD0	ANS2	ANS1	ANS0	ANE2	ANE1	ANE0	Initial value 00000000_B
				R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

ANE2	ANE1	ANE0	A/D conversion end channel select 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 select bit		
			Halt	Read during conversion	Read during a pause in stop conversion mode
0	0	0	AN0 pin	The current conversion channel number	The last conversion channel number
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 select 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

Figure 16.4-3 A/D control status register 0 (ADCS0)

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 select bit	• These bits select the conversion mode of the A/D conversion function. • The two-bit value of the MD1 and MD0 bits determines the mode that is selected from among four modes: single conversion mode 1, single conversion mode 2, continuous conversion mode, and stop conversion mode. • The operation in each mode is described below: Single conversion mode 1: Just a single A/D conversion from the channel set by ANS2 to ANS0 to the channel specified by ANE2 to ANE0 is performed. Reactivation during operation is allowed. Single conversion mode 2: Just a single A/D conversion from the channel set by ANS2 to ANS0 to the channel specified by ANE2 to ANE0 is performed. Reactivation during operation is not allowed. Continuous conversion mode: A/D conversion from the channel set by ANS2 to ANS0 to the channel specified by ANE2 to ANE0 is performed repeatedly. The repeated conversion continues until it is stopped by the BUSY bit. Reactivation during operation is not allowed. Stop conversion mode: A/D conversion from the channel set by ANS2 to ANS0 to the channel specified by ANE2 to ANE0 is performed repeatedly with a pause after the conversion of each channel. The repeated conversion continues until it is stopped by the BUSY bit. Reactivation during operation is not allowed. In the pause state, the conversion is reactivated when an activation cause selected by the STS1 and STS0 bits of ADCS1 is generated. **<Caution>** In the single conversion modes, continuous conversion mode, and stop conversion mode, no reactivation by a timer, external trigger, or software is allowed.
bit5 bit4 bit3	ANS2, ANS1, ANS0: A/D conversion start channel select bit	• These bits set the A/D conversion start channel and indicate the number of the current conversion channel. • When activated, A/D conversion starts from the channel specified by these bits. • During A/D conversion, the bits indicate the number of the current conversion channel. During a pause in stop conversion mode, the bits indicate the number of the last conversion channel.
bit2 bit1 bit0	ANE2, ANE1, ANE0: A/D conversion end channel select bit	• These bits set the A/D conversion end channel. • When activated, A/D conversion is performed up to the channel specified by these bits. • When these bits specify the channel specified by ANS2 to ANS0, just that channel is converted. In continuous or stop conversion mode, the start channel specified by ANS2 to ANS0 is converted after the channel specified by these bits. If the start channel is greater than the end channel, the start channel to AN7 and AN0 to the end channel are converted in that order in a single series of conversions.

16.4.3 A/D data register (ADCR0, 1)

The A/D data register (ADCR0, 1) holds the result of A/D conversion and selects the resolution of A/D conversion.

■ **A/D data register (ADCR0,1)**

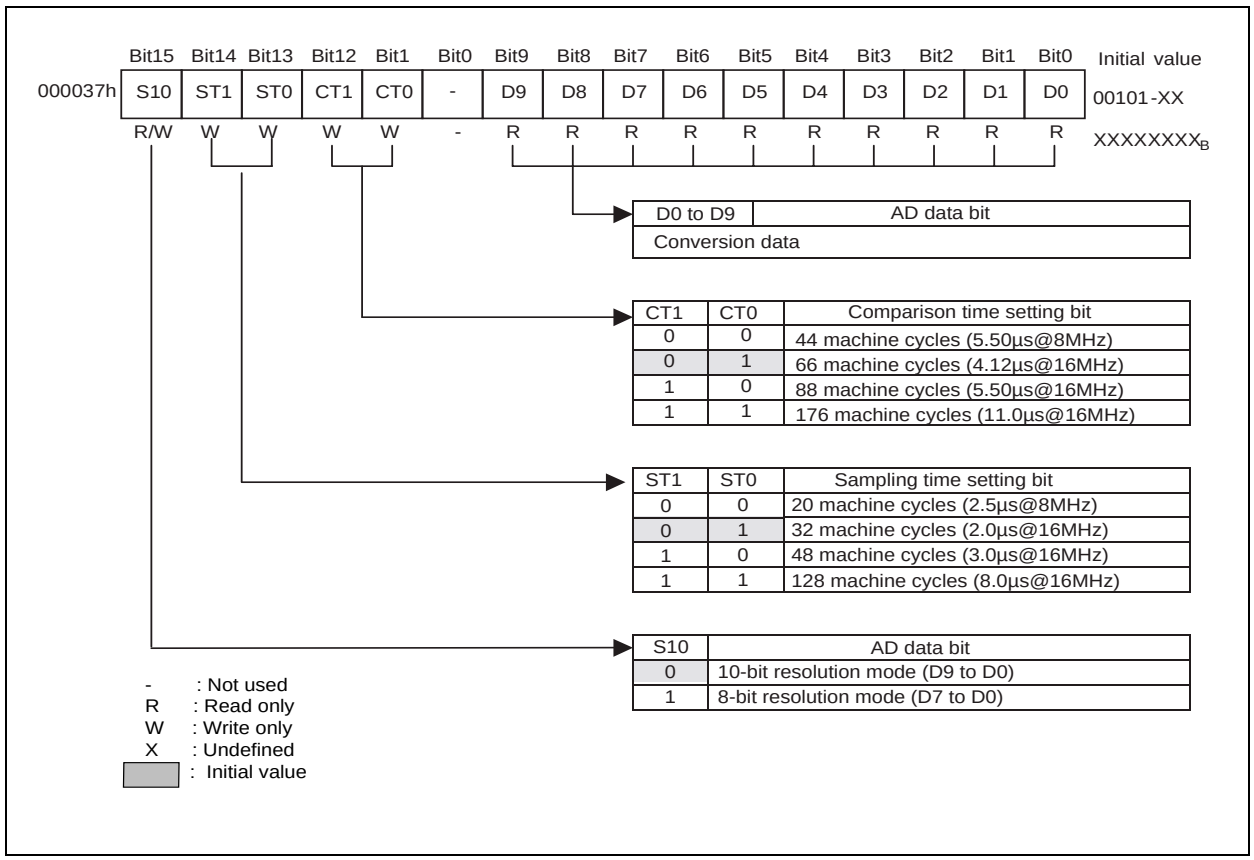


Figure 16.4-4 A/D data register (ADCR0, 1)

Table 16.4-3 Function description of each bit of A/D control status register 0 (ADCS0)

Bit name		Function
bit15	S10: A/D conversion resolution selection bit	• This bit selects an A/D conversion resolution. • Writing “0” to this bit selects a resolution of 10 bits, and writing “1” to this bit selects a resolution of 8 bits. **<Caution>** The data bit to be used depends on the resolution.
bit14 bit13	ST1, ST0: Sampling time setting bit	• These bits select the sampling time for A/D conversion. • When A/D conversion is activated, analog input is fetched during the time set in this bit. **<Caution>** Setting these bits to “00_B” (for 8 MHz) during 16-MHz operation may not be obtain normal fetching of the analog voltage.
bit12 bit11	CT1, CT0: Comparison time setting bit	• These bits select the comparison time for A/D conversion. • After analog input is fetched (i.e., sampling time elapses), conversion result data is defined and stored in bits 9 to 0 of this register after the time set in these bits. **<Caution>** Setting these bits to “00_B” (for 8 MHz) during 16-MHz operation may not be obtain normal acquisition of the analog conversion value.
bit10	Free bit	
bit9 ~ bit0	ANE2, ANE1, ANE0: A/D conversion end channel selection bit	• The A/D conversion results are stored and the register is rewritten each time conversion ends. • Usually, the last conversion value is stored. • The initial value of this register is undefined. **<Caution>** The conversion data protection function is provided. (See Section 16.6, "Operation of the 8/10-Bit A/D Converter.") Do not write data to these bits during A/D conversion.

<Check>

- To rewrite the S10 bit, do so while the A/D is in a pause before conversion. If the bit is rewritten after the conversion, the contents of ADCR become undefined.
- To read the contents of the ADCR register in 10-bit mode, use a word transfer instruction (MOVW A, 002EH, etc.).

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 stored in the A/D data register. This function supports the extended intelligent I/O service (EI²OS).

■ 8/10-bit A/D converter interrupts

Table 16.5-1 indicates the interrupt control bits of the 8/10-bit A/D converter and the interrupt cause.

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
Interrupt request enable bit	ADCS: INTE
Interrupt cause	Writing the A/D conversion result to the A/D data register

When A/D conversion is performed and its result is stored in the A/D data register (ADCR), the INT bit of the A/D control status register (ADCS1) is set to “1”. If the interrupt request is enabled (ADCS1: INTE = 1), an interrupt request is output to the interrupt controller.

■ 8/10-bit A/D converter interrupts and EI²OS

Table 16.5-2 8/10-bit A/D converter interrupts and EI²OS

Interrupt No.	Interrupt control register		Vector table address			EI²OS
	Register name	Address	Lower	Upper	Bank	
#11 ($0B_H$)	ICR00	$0000B0_H$	$FFFFD0_H$	$FFFFD1_H$	$FFFFD2_H$	o

o: Available

■ EI²OS function of the 8/10-bit A/D converter

Using the EI²OS function, the 10-bit A/D converter can transfer the A/D conversion result to memory. When the transfer is performed, a conversion data protection function halts the A/D conversion until the A/D conversion data is transferred to memory, and clears the INT bit. The function prevents any part of the data from being lost.

16.6 Operation of the 8/10-Bit A/D Converter

The 8/10-bit A/D converter has three conversion modes: single conversion mode, continuous conversion mode, and stop conversion mode. This section describes operation in each mode.

■ Operation in single conversion mode

In single conversion mode, the analog inputs from the channel specified by the ANS bits to the channel specified by the ANE bits are sequentially converted. When the channels up to the end channel specified by the ANE bits have been converted, A/D conversion stops. If the start and end channels are the same (ANS = ANE), just the channel specified by the ANS bits is converted.

Figure 16.6-1 shows the settings required for operation in single conversion mode.

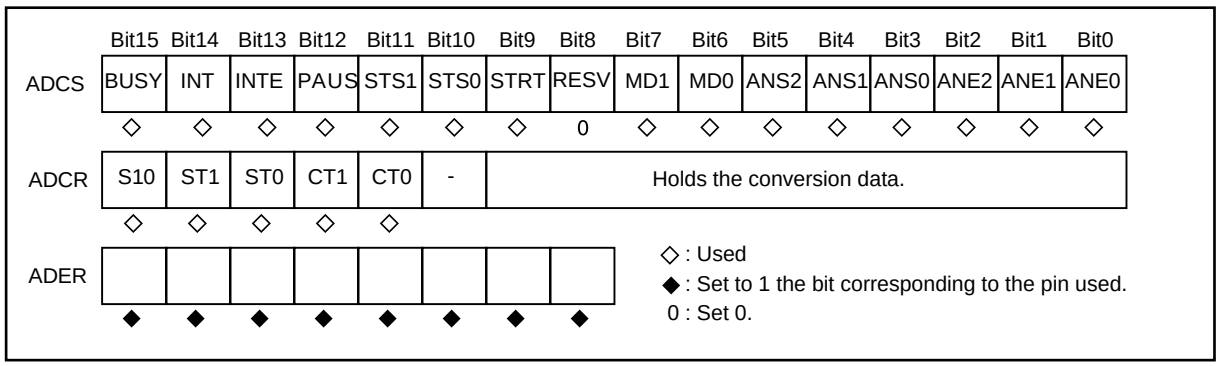


Figure 16.6-1 Settings for single conversion mode

<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, the analog inputs from the channel specified by the ANS bits to the channel specified by the ANE bits are sequentially converted. When the end channel specified by the ANE bits has been processed, A/D conversion starts again from the channel specified by the ANS bits. If the start and end channels are the same (ANS = ANE), the conversion of the channel specified by the ANS bits is repeated.

Figure 16.6-2 shows the settings required for operation in continuous conversion mode.

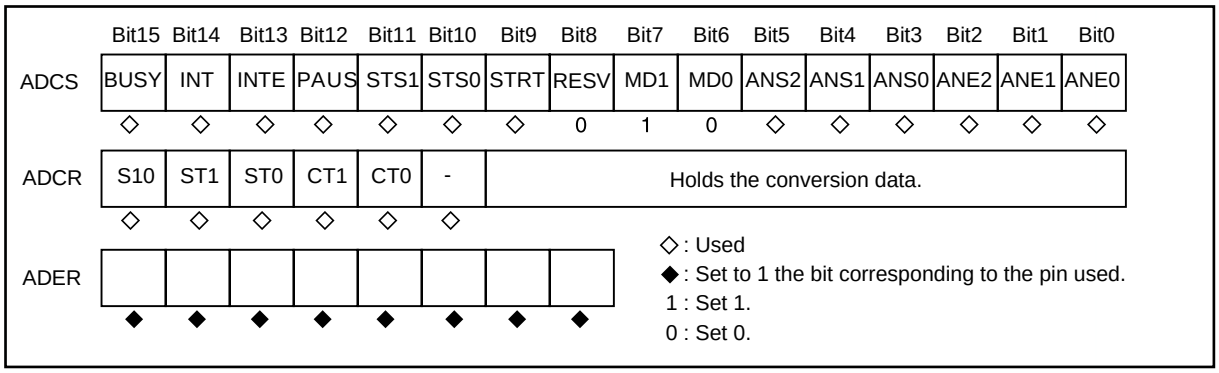


Figure 16.6-2 Settings for continuous conversion mode

<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, the analog inputs from the channel specified by the ANS bits to the channel specified by the ANE bits are sequentially converted with a pause after the conversion of each channel. When the end channel specified by the ANE bits has been processed, A/D conversion, with pauses, starts again with the channel specified by the ANS bits. If the start and end channels are the same (ANS = ANE), the conversion of the channel specified by the ANS bits is repeated. To reactivate conversion during a pause, generate the activation cause specified by the STS1 and STS0 bits.

Figure 16.6-3 shows the settings required for operation in stop conversion mode.

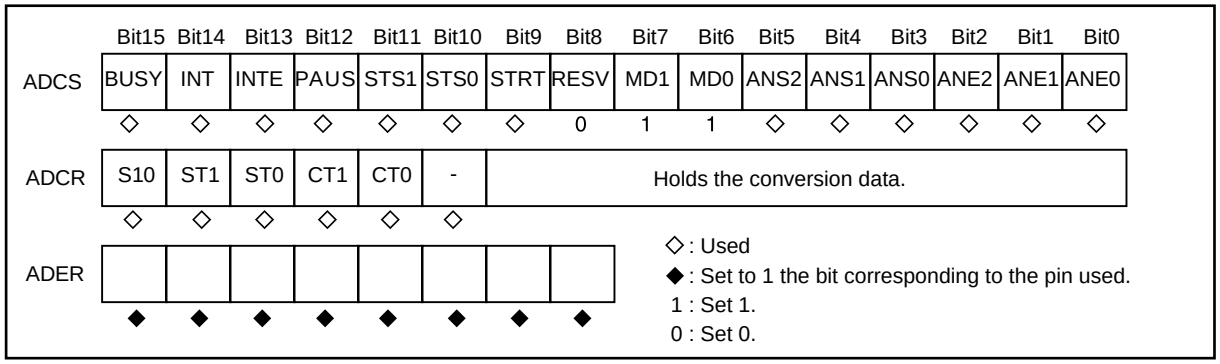


Figure 16.6-3 Settings for stop conversion mode

<Reference>

The following are sample conversion sequences in stop conversion mode:

ANS = 000B$_B$, ANE = 011$_B$:

AN0 → Pause → AN1 → Pause → AN2 → Pause → AN0 → Repeat

ANS = 110$_B$, ANE = 001$_B$:

AN6 → Pause → AN7 → Pause → AN0 → Pause → AN1 → AN6 → Repeat

ANS = 011$_B$, ANE = 011$_B$:

AN3 → Pause → AN3 → Pause → Repeat

16.6.1 Conversion using EI²OS

The 8/10-bit A/D converter can use EI²OS transfer the A/D conversion result to memory.

■ Conversion using EI²OS

Figure 16.6-4 shows the operation flow when EI²OS is used.

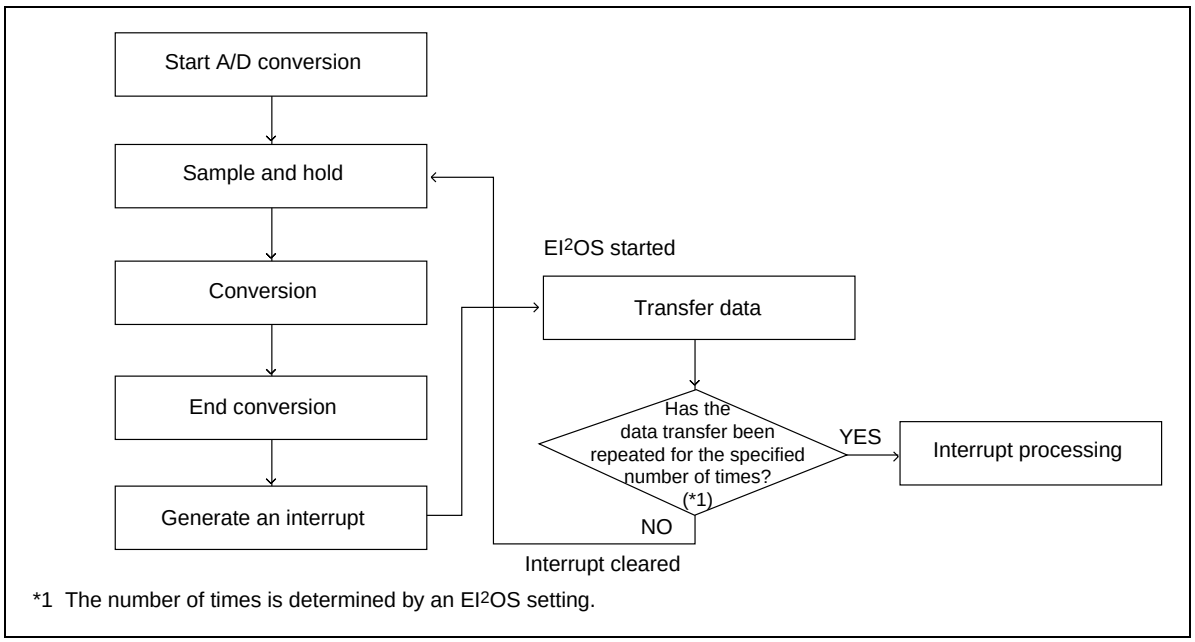


Figure 16.6-4 Sample operation flowchart when EI²OS is used

When EI²OS 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.

Memo

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 A/D converter has just one data register that holds conversion data. When a single A/D conversion is completed, the data in the data register is rewritten.

If the conversion data were not transferred to memory before the next conversion data was stored, part of the conversion data would be lost. The data protection function operates in the interrupt enabled state (INTE = 1), as described below, to prevent loss of data.

● Data protection function when EI²OS is not used

When conversion data is stored in the A/D data register (ADCR), the INT bit of the A/D control status register 1 (ADCS1) is set to “1”.

While the INT bit is “1”, A/D conversion is halted.

Halt status is released when the INT bit is cleared after data in the A/D data register (ADCR) has been transferred to memory by the interrupt routine.

● Data protection function when EI²OS is used

In continuous conversion using EI²OS, the PAUS bit of the A/D control status register1 (ADCS1) is kept at “1” when a conversion ends. This status continues until EI²OS finishes transferring the conversion data from the data register to memory. In the meantime, the A/D conversion is halted, and the next conversion data is not stored. When the data transfer to memory is completed, the PAUS bit is cleared to “0”, conversion and conversion resumes.

Figure 16.6-5 shows the operation flow of the data protection function when EI²OS is used.

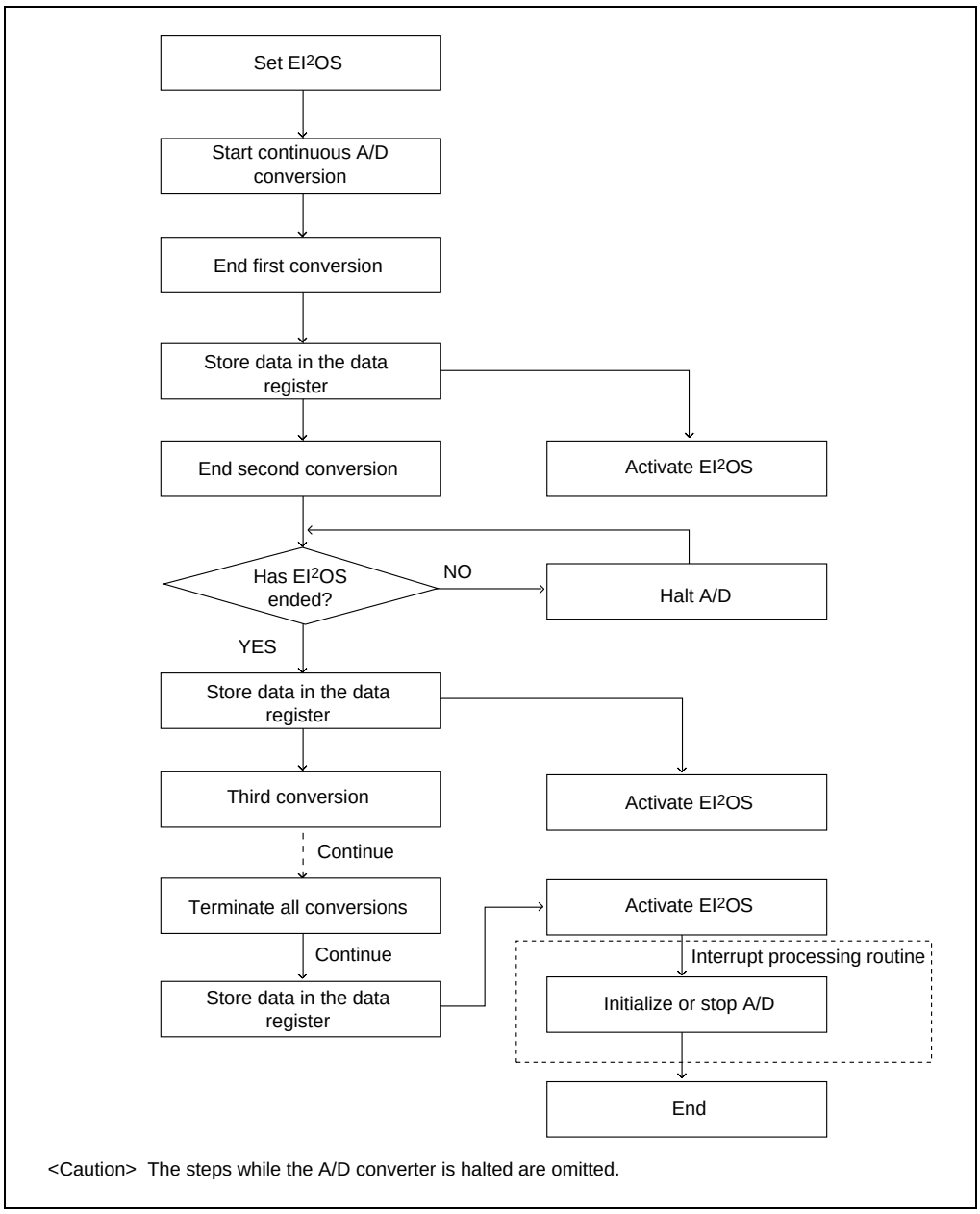


Figure 16.6-5 Operation flowchart of the data protection function when EI²OS is used

<Check>

- The conversion data protection function operates only in the interrupt enabled state (ADCS1: INTE = 1).
- If interrupts are disabled during a pause in A/D conversion while EI²OS is operating, A/D conversion may start again. This will cause new data to be written before the old data is transferred. Reactivation attempted during a pause will cause the old data to be destroyed.
- 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 A/D input pins are also used as the I/O pins of port 5. The port 5 data register (DDR5) and analog input enable register (ADER) determine which pin is used for which purpose.

  To use a pin as analog input, write “0” to the corresponding bit of DDR5 and change the port setting to input. Then, set the analog input mode (ADEx = 1) in the ADER register and determine the input gate of the port.

  If an intermediate-level signal is input in the port input mode (ADEx = 0), a leakage current flows through the gate.

- **Note on using an internal timer**

  To start the A/D converter with an internal timer, set the STS1 and STS0 bits of A/D control status register 1 (ADCS1) accordingly. Set the input value of the internal timer at the inactive level (L for the internal timer). Otherwise, operation may start concurrently with writing to the ADCS register.

- **Sequence of turning on the A/D converter and analog input**

  Do not turn on power to the A/D converter (AVcc, AVR+, AVR-) and to the analog inputs (AN0 to AN7) before the digital power supply (Vcc) has been turned on.

  Do not turn off the digital power supply (Vcc) before power to the A/D converter and the analog inputs has been turned off.

- **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.

Memo

16.8 Sample Program 1 for Single Conversion Mode Using EI²OS

This section contains a sample program for A/D conversion in single conversion mode using EI²OS.

■ Sample program for single conversion mode using EI²OS

● Processing

- Analog inputs AN1 to AN3 are converted once.
- The conversion data is sequentially transferred to addresses 200H to 205H.
- A resolution of 10 bits is selected.
- The conversion is activated by software.

Figure 16.8-1 shows a flowchart of the program using EI²OS (single conversion mode).

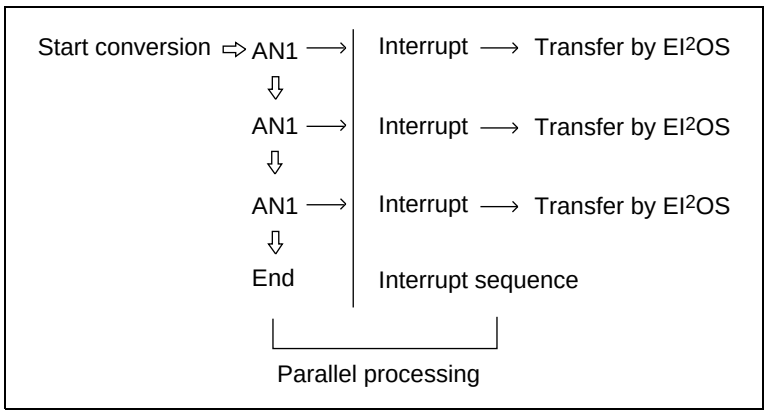


Figure 16.8-1 Flowchart of program using EI²OS (single conversion mode)

● Coding example

```
BAPL    EQU   000100H    ;  Lower buffer address pointer
BAPM    EQU   000101H    ;  Intermediate buffer address pointer
BAPH    EQU   000102H    ;  Upper buffer address pointer
ISCS    EQU   000103H    ;  EI²OS 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-----------------------------------------------------------------------------------------------------
CODE    CSEG
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 EI²OS 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     CTH,#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.
CODE    ENDS
;-------Vector setting---------------------------------------------------------------------------------------------------
VECT    CSEG    ABS=0FFH
        ORG     0FFD0H                  ;  Sets vector for interrupt #31 (1FH)
        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 Continuous Conversion Mode Using EI²OS

This section contains a sample program for A/D conversion in continuous conversion mode using EI²OS.

■ Sample program for continuous conversion mode using EI²OS

● 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 600H to 60BH.
- A resolution of 10 bits is selected.
- The conversion is activated by 16-bit reload timer 1.

Figure 16.9-1 shows a flowchart of the program using EI²OS (continuous conversion mode).

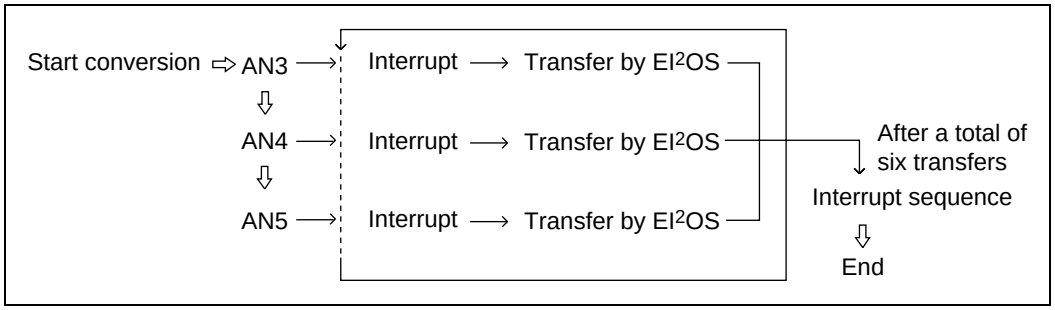


Figure 16.9-1 Flowchart of program using EI²OS (continuous conversion mode)

● 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         ;   EI²OS 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         ;
TMCRL1  EQU     000086H         ;Lower control status register 1
```

```
TMCRH1  EQU     000087H             ;
RLDRL1  EQU     000088H             ;16-bit reload register
RLDRH1  EQU     000089H             ;
;-------Main program-----------------------------------------------------------------------------------------
---------
CODE    CSEG
START:                              ;   Assumes that the stack pointer (SP) has already
                                    ;   been initialized.
        AND     CCR,#0BFH           ;   Disables interrupts.
        MOV     ICR10,#08H          ;   Interrupt level;  0 (highest priority).  Enables
                                    ;   interrupts.
        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 Stop Conversion Mode Using EI²OS

This section contains a sample program for A/D conversion in stop conversion mode using EI²OS.

■ Sample program for stop conversion mode using EI²OS

● 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 shows a flowchart of the program using EI²OS (stop conversion mode).

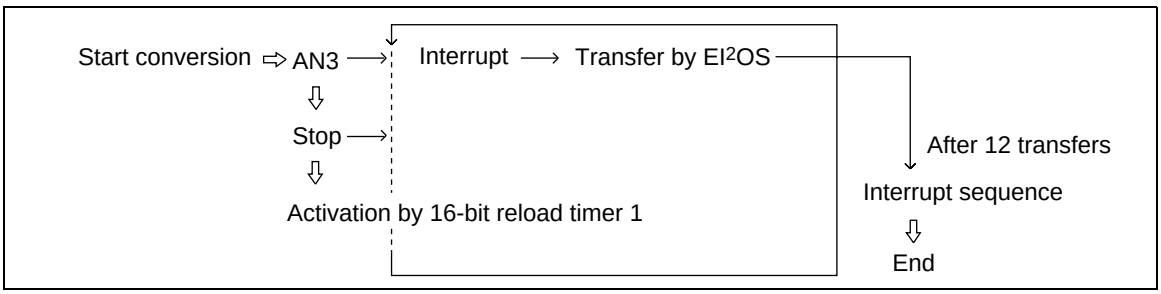


Figure 16.10-1 Flowchart of program using EI²OS (stop conversion mode)

● 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 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          ;
TMCRL1  EQU   000086H          ;Lower control status register 1
TMCRH1  EQU   000087H          ;
RLDRL1  EQU   000088H          ;16-bit reload register
RLDRH1  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 a
                                    ;  resource 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
```

Memo

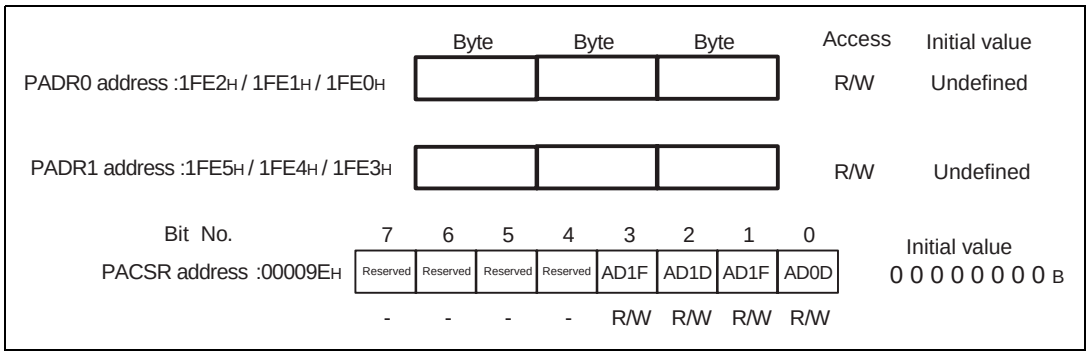
CHAPTER 17 ADDRESS MATCH DETECTION FUNCTION

This chapter explains the address match detection function and operation of the MB90506 series.

17.1 Overview of the Address Match Detection Function

An instruction code to be read by the CPU is replaced forcibly with an INT9 instruction code (01H) when the corresponding address is equal to the value set in an address detection register. Therefore, the CPU executes the INT9 instruction when executing the set instruction. A program patch application function can be implemented by processing with the INT #9 interrupt routine.There are two address detection registers, of which each is provided with an interrupt enable bit and interrupt flag.
When the address is equal to the value set in the address detection register, and the interrupt enable bit is “1”, assume the following: the interrupt flag is set to “1”, and the instruction code to be read by the CPU is replaced forcibly with the INT9 instruction code. The interrupt flag is cleared to “0” by writing “0” to it using an instruction.

■ **Registers**



■ **Block Diagram**

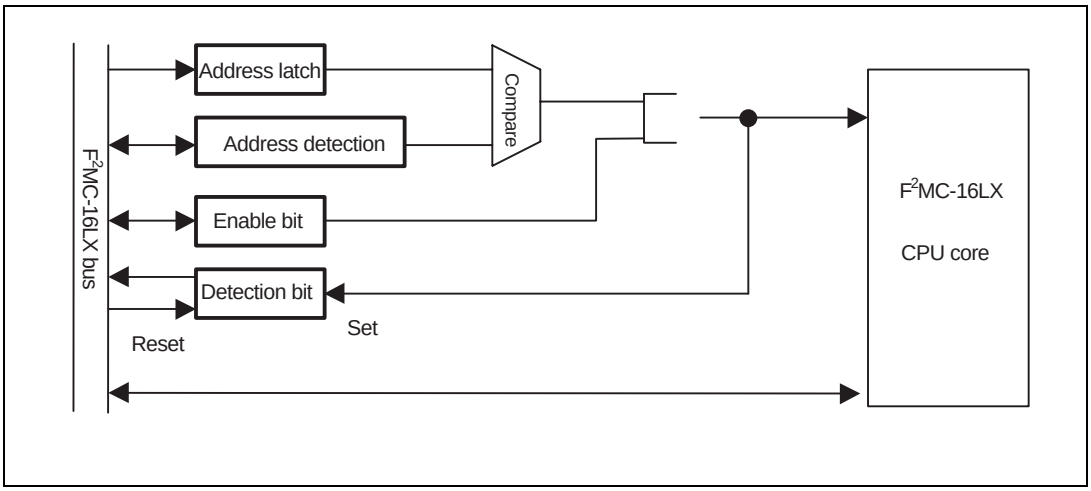


Figure 17.1-1 Block diagram

■ Register Details

● Program address detect register 0/1: PADR0/PADR1:

The value written to each register is compared with a target address. If the value matches the address, and the corresponding interrupt enable bit of the PACSR register is “1”, the corresponding interrupt bit is set to “1” to request the CPU to generate an INT9 instruction. If the corresponding interrupt enable bit is “0”, no operation is performed.

	Byte	Byte	Byte	Access	Initial value
PADR0 address :$1FF2_H$ /$1FF1_H$/$1FF0_H$				R/W	Indefinite
PADR1 address :$1FF5_H$ /$1FF4_H$ /$1FF3_H$				R/W	Indefinite

The following lists the correspondence between the program address detection register and PACSR:

Address detection register	Interrupt enable bit	Interrupt bit
PADR0	AD0E	AD0D
PADR1	AD1E	AD1D

● Program address detect control or status register: PACSR

Bit No.	7	6	5	4	3	2	1	0	Initial value
PACSR address:$00009E_H$	Reserved	Reserved	Reserved	Reserved	AD1E	AD1D	AD0E	AD0D	0 0 0 0 0 0 0 0
	-	-	-	-	R/W	R/W	R/W	R/W	

This register controls the operation of the address detection function and indicates its status.

[bit 7 to bit 4]: Reserved bits

Reserved bits. Be sure to write 0 to these bits.

[bit 3]: Address Detect Register 1 Enable (AD1E)

ADR1 operation enable bit. When this bit is “1”, the value set in the PADR1 register is compared with the address. If the two values are equal, an INT9 instruction is generated and the AD1D bit is set to “1”.

[bit 2]: Address Detect Register1’s is Detected (AD1D)

ADR1 address match detection bit. This bit is set to “1” to indicate that the value set in the PADR1 register matches the address. It is cleared to “0” by writing “0” to it. It is left unchanged by writing “1” to it.

[bit 1]: Address Detect Register 0 Enable (AD0E)

ADR0 operation enable bit. When this bit is “1”, the value set in the PADR0 register is compared to the address. If they match, an INT9 instruction is generated, and the AD0D bit is set to “1”.

[bit0]: Address Detect Register0's is Detected (AD0D)

ADR0 address match detection bit. This bit is set to "1" to indicate that the value set in the PADR0 register is equal to the address. It is cleared to "0" by writing "0" to it. It is left unchanged by writing "1" to it.

■ Operation of the Address Match Detection Function

An instruction code to be read by the CPU is replaced forcibly with an INT9 instruction code (01_H) when the corresponding address is equal to the value set in an address detection register. Therefore, the CPU executes the INT9 instruction when executing the set instruction.

A program patch application function can be implemented by processing with the INT #9 interrupt routine.

There are two address detection registers, of which each is provided with an interrupt enable bit and interrupt flag. When the address is equal to the value set in the address detection register, and the interrupt enable bit is "1", assume the following: the interrupt flag is set to "1", and the instruction code to be read by the CPU is replaced forcibly with the INT9 instruction code. The interrupt flag is cleared to "0" by writing "0" to it using an instruction.

■ Notes on the Address Match Detection Function

The address match detection function fails if an address later than the first byte of the instruction is set in the address detection register. The value in the set address is replaced with 01_H so a wrong instruction is executed or an invalid address is accessed. Before changing the value set in the address detection register, set the interrupt enable bit to "0". If data is written while the interrupt enable bit is "1", the address may be wrongly detected during writing, causing a malfunction.

17.2 Example of Using the Address Match Detection Function

This section contains example of Using the Address Match Detection Function.

■ System Configuration

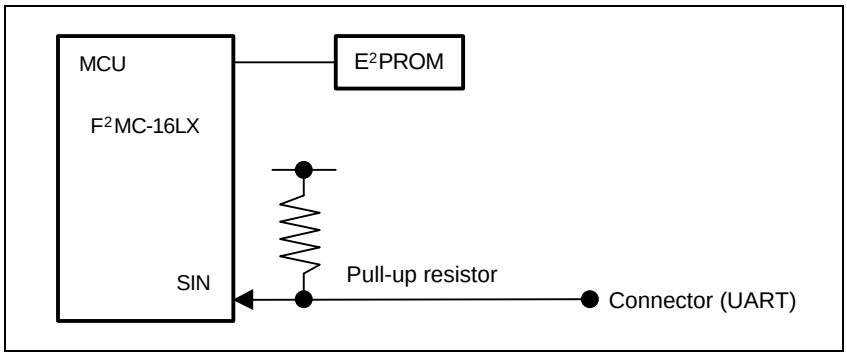


Figure 17.2-1 System configuration example

■ EEPROM Memory Map

Address	Meaning
0000H	Number of bytes of patch program No. 0 (“0” for no program error)
0001H	Bit 7 to bit 0 of program address No. 0
0002H	Bit 15 to bit 8 of program address No. 0
0003H	Bit 24 to bit 26 of program address No. 0
0004H	Number of bytes of patch program No. 1 (“0” for no program error)
0005H	Bit 7 to bit 0 of program address No. 1
0006H	Bit 15 to bit 8 of program address No. 1
0007H	Bit 24 to bit 16 of program address No. 1
0010H+ Number of bytes of patch program No. 0	Original of patch program No. 0

■ Initial State

The contents of EEPROM are all “0”s.

■ If a Program Error Occurs

The original of a patch program and its address are transferred to the MCU via the connector (UART). The MCU writes the information to EEPROM.

■ Reset Sequence

After the reset sequence is completed, the MCU reads the value of EEPROM. If the number of bytes of the patch program is not “0”, the MCU reads the original patch program and writes it to RAM. Then, the MCU sets the program address to PADR0 or ADR1 and enables the program to run. The first address of the program written to RAM is saved in RAM as specified for each address detection register.

■ INT9 Interrupt

During execution of an interrupt routine, control checks the interrupt flag for an address in which an interrupt was enabled, and branches to the corresponding program. The information stacked by the interrupt is deleted. The interrupt flag is also cleared.

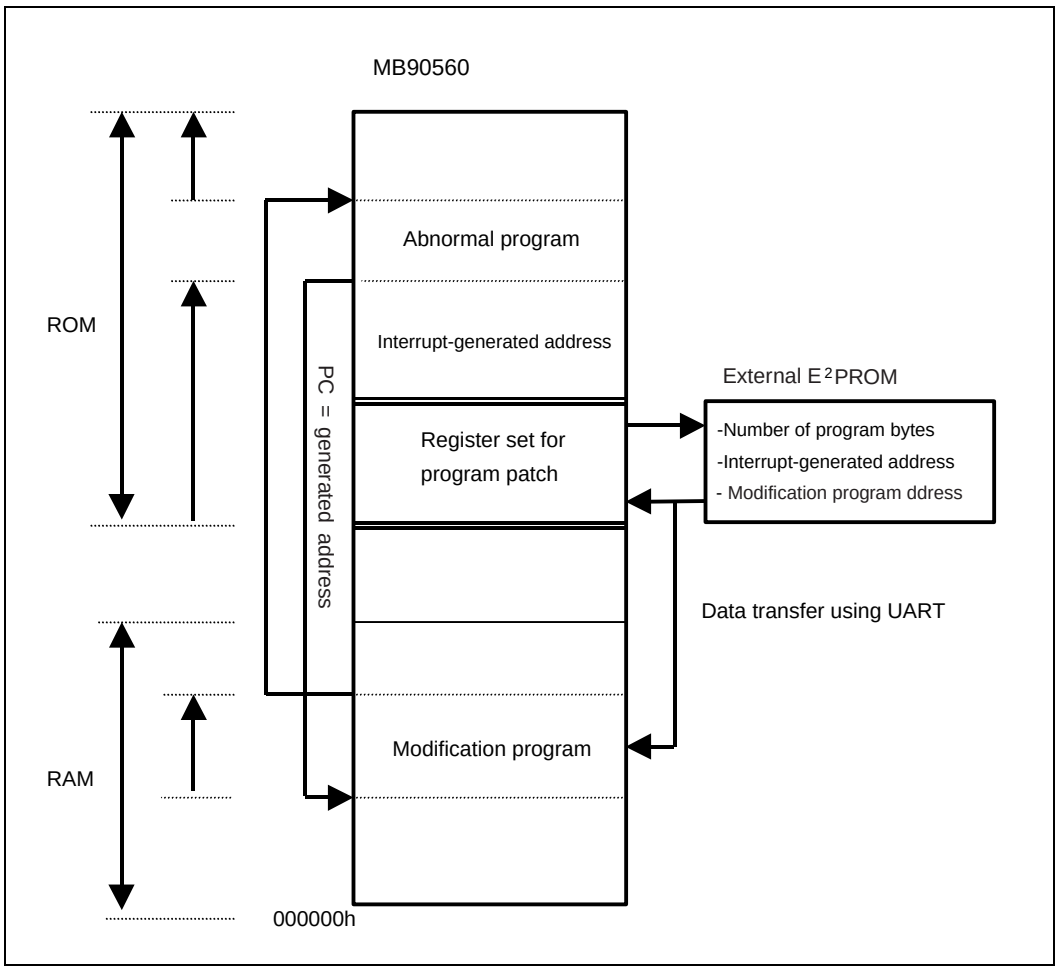


Figure 17.2-2 System configuration example

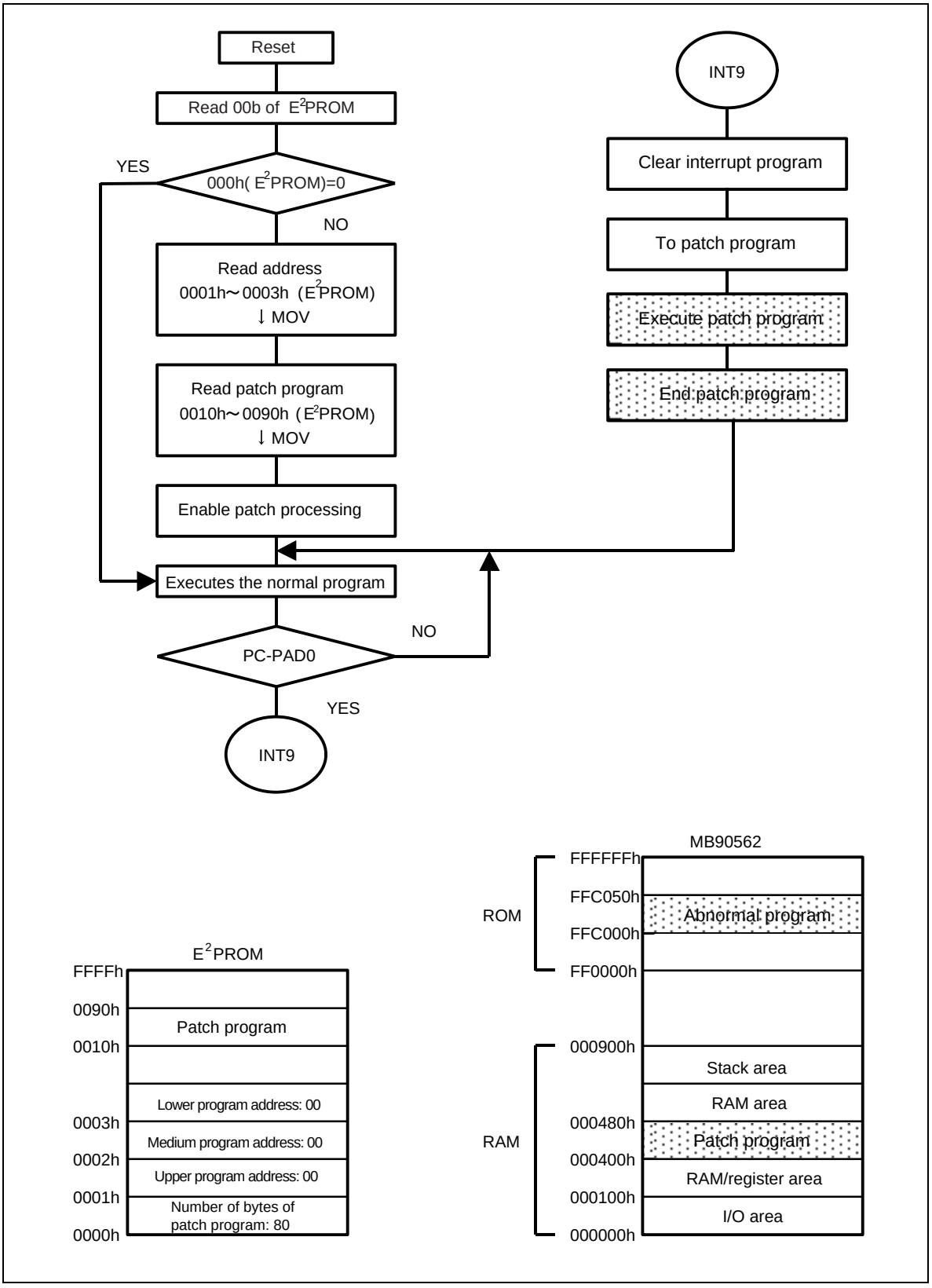


Figure 17.2-3 Flowchart of program patch processing

CHAPTER 18 ROM MIRRORING FUNCTION SELECTION MODULE

This chapter explains the function and operation of the MB90560 series ROM mirroring function selection module.

18.1 Overview of the ROM Mirroring Function Selection Module

The ROM mirroring function selection module can access bank FF located in ROM from bank 00 by setting the register.

■ Registers

bit	15	14	13	12	11	10	9	8	Initial value
ROMM address : 00006F$_H$	-	-	-	-	-	-	-	MI	------- 1 $_B$
								W	

■ Block Diagram

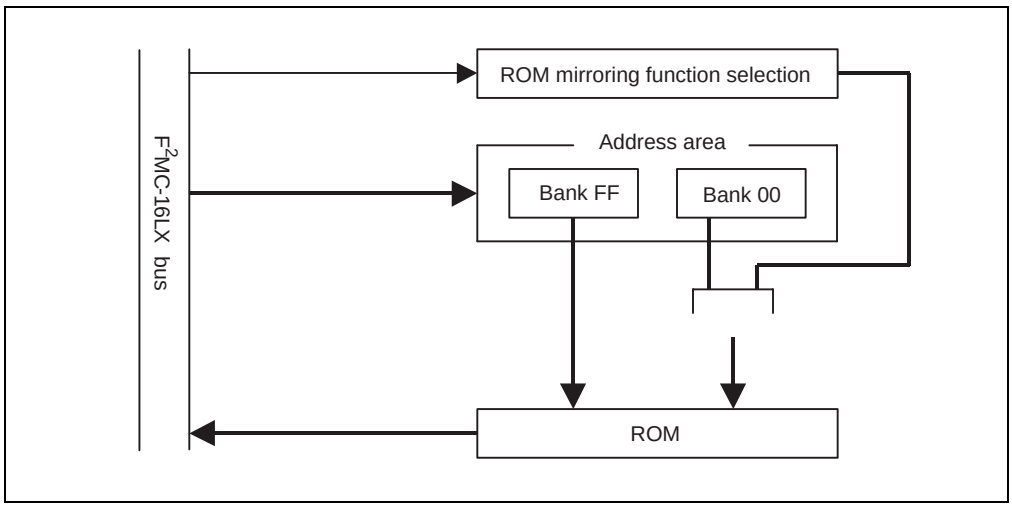


Figure 18.1-1 Block diagram

■ Register Details

● ROMM (ROM mirroring function selection register)

bit	15	14	13	12	11	10	9	8	Initial value
ROMM address : 00006F$_H$	-	-	-	-	-	-	-	MI	------- 1 $_B$
								W	

<Check>

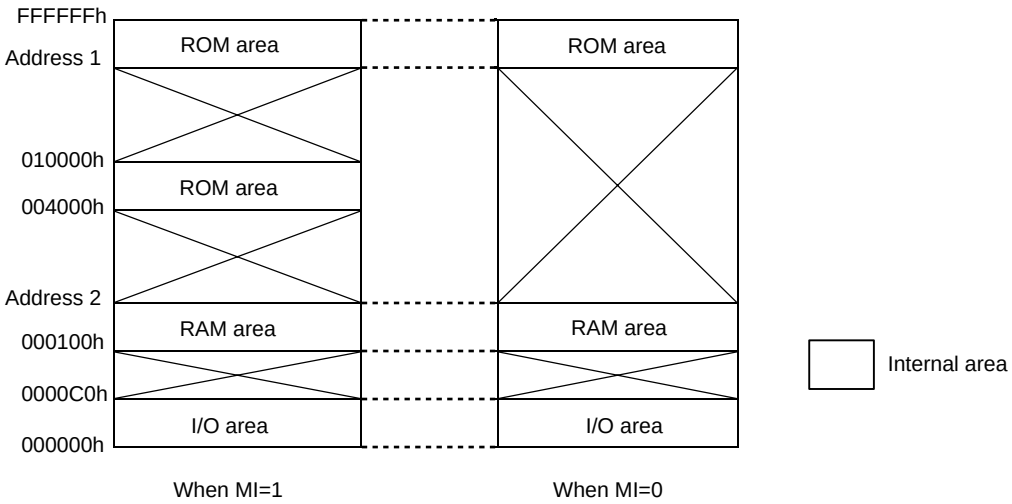
Do not access this register while the system is active in an address from "004000$_H$" to "00FFFF$_H$".

[bit 8]: MI

When "1" has been written to this bit, the ROM data in bank FF can be read from bank 00. When "0" has been written to this bit, the function is disabled in bank 00. This bit is a write-only bit.

<Check>

Bank 00 accesses "FF4000H" to "FFFFFFH" from "004000H" to "00FFFFH". Therefore, "FFF000H" to "FF3FFFH" cannot be accessed even by selecting the ROM mirroring function.



	MB90561	MB90562	MB90F562	MB90V560
Address 1	FF8000h	FF0000h	FF0000f	FF0000h
Address 2	000500h	000900h	000900h	001100h

Figure 18.1-2 Memory space

APPENDIX

The appendixes contain an I/O map and other information and describe instructions.

APPENDIX A I/O MAP

Table A lists the addresses assigned to the registers for peripheral functions in the MB90560 series.

■ I/O map

Table A I/O map

Address	Abbreviation	Register	Access	Resource name	Initial value
000000H	PDR0	Port 0 data register	R/W	Port 0	XXXXXXXXB
000001H	PDR1	Port 1 data register	R/W	Port 1	XXXXXXXXB
000002H	PDR2	Port 2 data register	R/W	Port 2	XXXXXXXXB
000003H	PDR3	Port 3 data register	R/W	Port 3	XXXXXXXXB
000004H	PDR4	Port 4 data register	R/W	Port 4	*XXXXXXXB
000005H	PDR5	Port 5 data register	R/W	Port 5	XXXXXXXXB
000006H	PDR6	Port 6 data register	R/W	Port 6	****XXXXB
000007H ~0FH	Prohibited area				
000010H	DDR0	Port 0 direction register	R/W	Port 0	00000000B
000011H	DDR1	Port 1 direction register	R/W	Port 1	00000000B
000012H	DDR2	Port 2 direction register	R/W	Port 2	00000000B
000013H	DDR3	Port 3 direction register	R/W	Port 3	00000000B
000014H	DDR4	Port 4 direction register	R/W	Port 4	*0000000B
000015H	DDR5	Analog input enable register	R/W	Port 5	00000000B
000016H	DDR6	Port 6 direction register	R/W	Port 6	****0000B
000017H	ADER	Analog input enable register	R/W	Port 5, A/D	11111111B
000018H ~1FH	Prohibited area				
000020H	SMR0	Serial mode control register 0	R/W	UART0	00000000B
000021H	SCR0	Serial control register 0	R/W		00000100B
000022H	SIDR0/ SODR0	Serial Input data register 0/ Serial output data register 0	R/W		XXXXXXXXB
000023H	SSR0	Serial status register 0	R/W		00001000B
000024H	SMR1	Serial mode control register 1	R/W	UART1	00000000B
000025H	SCR1	Serial control register 1	R/W		00000100B
000026H	SIDR1 SODR1	Serial Input data register 1/ Serial output data register 1	R/W		XXXXXXXXB
000027H	SSR1	Status register 1	R/W		00001000B
000028H	Prohibited area				

Table A I/O map *(continued)*

Address	Abbreviation	Register	Access	Resource name	Initial value
000029H	ODCR0	Communication prescaler control register 0	R/W	Communication prescaler (UART0)	0***0000B
00002AH	Prohibited area				
00002BH	CDCR1	Co0mmunication prescaler control register 1	R/W	Communication prescaler (UART1)	0***0000B
00002CH ~2FH	Prohibited area				
000030H	ENIR	Interrupt/DTP enable register	R/W	DTP/external interrupt	00000000B
000031H	ENRR	Interrupt/DTP cause register	R/W		00000000B
000032H	ELVR	Request level setting register	R/W		00000000B
000033H			R/W		00000000B
000034H	ADCS0	A/D control status register	R/W		00000000B
000035H	ADCS1		R/W		00000000B
000036H	ADCR0	A/D data register	R		XXXXXXXXB
000037H	ADCR1		R or W		00101*XXB
000038H	PRLL0	PPG0 reload register	R/W	8-/16-bit PPG timer (CH0, CH1)	XXXXXXXXB
000039H	PRLH0		R/W		XXXXXXXXB
00003AH	PRLL1	PPG1 reload register	R/W		XXXXXXXXB
00003BH	PRLH0		R/W		XXXXXXXXB
00003CH	PPGC0	PPG0 operation mode register	R/W		00000001B
00003DH	PPGC1	PPG1 operation mode register	R/W		00000001B
00003EH	PCS01	PPG0/PPG1 clock control register	R/W		000000**B
00003FH	Prohibited area				
000040H	PRLL2	PPG2 reload register	R/W	8-/16-bit PPG timer (CH2, CH3)	XXXXXXXXB
000041H	PRLH2		R/W		XXXXXXXXB
000042H	PRLL3	PPG3 reload register	R/W		XXXXXXXXB
000043H	PRLH3		R/W		XXXXXXXXB
000044H	PPGC2	IPPG2 operation mode register	R/W		00000001B
000045H	PPGC3	PPG3 operation mode register	R/W		00000001B
000046H	PCS23	PPG2/PPG3 clock control register	R/W		000000**B
000047H	Prohibited area				

Table A I/O map *(continued)*

Address	Abbreviation	Register	Access	Resource name	Initial value
000048H	PRLL4	PPG4 reload register	R/W	8-/16-bit PPG timer (CH4, CH5)	XXXXXXXXB
000049H	PRLH4		R/W		XXXXXXXXB
00004AH	PRLL5	PPG5 reload register	R/W		XXXXXXXXB
00004BH	PRLH5		R/W		XXXXXXXXB
00004CH	PPGC4	PPG4 operation mode register	R/W		00000001B
00004DH	PPGC5	PPG5 operation mode register	R/W		00000001B
00004EH	PCS45	PPG4/PPG5 clock control register	R/W		000000**B
00004FH	Prohibited area				
000050H	TMRR0	8-bit reload register CH0	R/W	Waveform generator	XXXXXXXXB
000051H	DTCR0	8-bit timer control register CH0	R/W		00000000B
000052H	TMRR1	8-bit reload register CH1	R/W		XXXXXXXXB
000053H	DTCR1	8-bit timer control register CH1	R/W		00000000B
000054H	TMRR2	8-bit reload register CH2	R/W		XXXXXXXXB
000055H	DTCR2	8-bit timer control register CH2	R/W		00000000B
000056H	SIGCR	Waveform control register	R/W		00000000B
000057H	Prohibited area				
000058H	CPCLR	Compare clear register (lower)	R/W	16-bit free-running timer	XXXXXXXXB
000059H		Compare clear register (upper)	R/W		XXXXXXXXB
00005AH	TCDT	Timer data register (lower)	R/W		00000000B
00005BH		Timer data register (upper)	R/W		00000000B
00005CH	TCCS	Timer control status register (lower)	R/W		0**00000B
00005DH		Timer control status register (upper)	R/W		00000000B
00005EH	Prohibited area				
00005FH					
000060H	IPCP0	Input capture data register CH0	R	16-bit input capture (CH00 to CH3)	XXXXXXXXB
000061H		Input capture data register CH0	R		XXXXXXXXB
000062H	IPCP1	Input capture data register CH1	R		XXXXXXXXB
000063H		Input capture data register CH1	R		XXXXXXXXB
000064H	IPCP2	Input capture data register CH2	R		XXXXXXXXB
000065H		Input capture data register CH2	R		XXXXXXXXB
000066H	IPCP3	Input capture data register CH3	R		XXXXXXXXB
000067H		Input capture data register CH4	R		XXXXXXXXB
000068H	ICS01	Input capture control register 01	R/W		00000000B
000069H	ICS23	Input capture control register 23	R/W		00000000B

Table A I/O map *(continued)*

Address	Abbreviation	Register	Access	Resource name	Initial value
00006A$_H$ ~6E$_H$	Prohibited area				
00006F$_H$	ROMM	ROM mirroring function selection register	R/W	ROM mirroring function	*******1$_B$
000070$_H$	OCCP0	Compare register CH0 (lower)	R/W	Output compare CH0 to CH5	XXXXXXXX$_B$
000071$_H$		Compare register CH0 (upper)	R/W		XXXXXXXX$_B$
000072$_H$	OCCP1	Compare register CH1 (lower)	R/W		XXXXXXXX$_B$
000073$_H$		Compare register CH1 (upper)	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	R/W		0000**00$_B$
00007D$_H$	OCS1	Compare control register CH1	R/W		***00000$_B$
00007E$_H$	OCS2	Compare control register CH2	R/W		0000**00$_B$
00007F$_H$	OCS3	Compare control register CH3	R/W		***00000$_B$
000080$_H$	OCS4	Compare control register CH4	R/W		0000**00$_B$
000081$_H$	OCS5	Compare control register CH5	R/W		***00000$_B$
000082$_H$	TMCR0:L	Timer control status register CH0 (lower)	R/W	16-bit reload timer CH0	00000000$_B$
000083$_H$	TMCR0:H	Timer control status register CH0 (upper)	R/W		****0000$_B$
000084$_H$	TMR0:L/ TMRLR0:L	16-bit timer register CH0 (lower)/ 16-bit reload register CH0 (lower)	R/W		XXXXXXXX$_B$
000085$_H$	TMR0:H TMRLR0:H	16-bit timer register CH0 (Upper)/ 16-bit reload register CH0 (upper)	R/W		XXXXXXXX$_B$
000086$_H$	TMCR1:L	Timer control status register CH1 (lower)	R/W	16-bit reload timer CH1	00000000$_B$
000087$_H$	TMCR1:H	Timer control status register CH1 (upper)	R/W		****0000$_B$
000088$_H$	TMR1:L/ TMRLR1:L	16-bit timer register CH1 (lower)/ 16-bit reload register CH1 (lower)	R/W		XXXXXXXX$_B$
000089$_H$	TMR1:H TMRLR1:H	16-bit timer register CH1 (Upper)/ 16-bit reload register CH1 (upper)	R/W		XXXXXXXX$_B$
00008A$_H$ ~8B$_H$	Prohibited area				

Table A I/O map *(continued)*

Address	Abbreviation	Register	Access	Resource name	Initial value
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$_H$ ~ 9D$_H$	Prohibited area				
00009E$_H$	PACSR	Program address detect control or status register	R/W	Address match detection	11000000$_B$
00009F$_H$	DIRR	Delayed interrupt cause/clear register	R/W	Delayed interrupt	*******0$_B$
0000A0$_H$	LPMCR	Low-power consumption mode register	R/W	Low-power consumption control register	00011000$_B$
0000A1$_H$	CKSCR	Clock selection register	R/W		11111100$_B$
0000A2$_H$ ~ A7$_H$	Prohibited area				
0000A8$_H$	WDTC	Watchdog control register	R/W	Watchdog timer	*****111$_B$
0000A9$_H$	TBTC	Timebase timer control register	R/W	Timebase timer	1**00100$_B$
0000AA$_H$~AD	Prohibited area				
0000AE$_H$	FMCS	Flash memory control status register	R/W	Flash memory interface circuit	000X0XX0$_B$
0000AF$_H$	Prohibited area				
0000B0$_H$	ICR00	Interrupt control register 00	R/W	Interrupt controller	00000111$_B$
0000B1$_H$	ICR01	Interrupt control register 01	R/W		00000111$_B$
0000B2$_H$	ICR02	Interrupt control register 02	R/W		00000111$_B$
0000B3$_H$	ICR03	Interrupt control register 03	R/W		00000111$_B$
0000B4$_H$	ICR04	Interrupt control register 04	R/W		00000111$_B$
0000B5$_H$	ICR05	Interrupt control register 05	R/W		00000111$_B$
0000B6$_H$	ICR06	Interrupt control register 06	R/W		00000111$_B$
0000B7$_H$	ICR07	Interrupt control register 07	R/W		00000111$_B$
0000B8$_H$	ICR08	Interrupt control register 08	R/W		00000111$_B$
0000B9$_H$	ICR09	Interrupt control register 09	R/W		00000111$_B$
0000BA$_H$	ICR10	Interrupt control register 10	R/W		00000111$_B$
0000BB$_H$	ICR11	Interrupt control register 11	R/W		00000111$_B$
0000BC$_H$	ICR12	Interrupt control register 12	R/W		00000111$_B$
0000BD$_H$	ICR13	Interrupt control register 13	R/W		00000111$_B$
0000BE$_H$	ICR14	Interrupt control register 14	R/W		00000111$_B$
0000BF$_H$	ICR15	Interrupt control register 15	R/W		00000111$_B$
0000C0$_H$ ~ FF$_H$	Unused area				

Table A I/O map *(continued)*

Address	Abbreviation	Register	Access	Resource name	Initial value
000100H ~ #H	RAM area				
#H ~ 001FEFH	Reserved area				
001FF0H	PADR0	Program address detection register 0	R/W	Address match detection	XXXXXXXXB
001FF1H		Program address detection register 1	R/W		XXXXXXXXB
001FF2H		Program address detection register 2	R/W		XXXXXXXXB
001FF3H	PADR1	Program address detection register 3	R/W		XXXXXXXXB
001FF4H		Program address detection register 4	R/W		XXXXXXXXB
001FF5H		Program address detection register 5	R/W		XXXXXXXXB
001FF6H ~ 1FFFH	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.

  *: The bit is not used. Its initial value is undefined.

APPENDIX B INSTRUCTIONS

This appendix describes the instructions used by the F²MC-16LX.

B.1 Instructions

The F²MC-16LX uses the 351 instructions listed below. Addresses must be specified in the effective address field of an instruction or by an instruction code.

■ Overview of instructions

The F²MC-16L uses the 351 instructions listed below.

- Transfer (byte): 41 instructions
- Transfer (word, long-word): 38 instructions
- Addition/subtraction (byte, word, long-word): 42 instructions
- Increment/decrement (byte, word, long-word): 12 instructions
- Comparison (byte, word, long-word): 11 instructions
- Unsigned multiplication/division (word, long-word) 11 instructions
- Signed multiplication/division (word, long-word) 11 instructions
- Logical operation (byte, word): 39 instructions
- Logical operation (long-word): 6 instructions
- Sign inversion (byte, word): 6 instructions
- Normalization (long-word): 1 instruction
- Shift (byte, word, long-word,): 18 instructions
- Branching: 50 instructions
- Accumulator operation (byte, word): 6 instructions
- Other types of control (byte, word, long-word): 28 instructions
- Bit operation: 21 instructions
- String: 10 instructions

Memo

B.2 Addressing

The F²MC-16LX determines the address format according to the instruction's effective-address field or from the instruction code (the address format is implied). When the address format is determined from the instruction code, the address format that matches the instruction code is used. More than one type of address format can be specified for some instructions.

■ Addressing

The F²MC-16LX uses the following 23 types of addressing:

- Immediate (#imm)
- Register direct
- Direct branch (addr16)
- Physical direct branch (addr24)
- I/O direct (io)
- Condensed direct (dir)
- Direct (addr16)
- I/O direct bit (io:bp)
- Condensed direct bit (dir:bp)
- Direct bit (addr16:bp)
- Vector (#vct)
- Register indirect (@RWj j = 0 to 3)
- Register indirect with post-incrementing
- (@RWj+ j = 0 to 3)
- Register indirect with displacement (@RWi+disp8 i = 0 to 7, @RWj+disp16 j = 0 to 3)
- Long-word 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 (rel)
- Register list (rlst)
- Accumulator indirect (@A)
- Accumulator indirect branch (@A)
- Indirect designation branch (@ear)
- Indirect designation branch (@eam)

■ Effective-address field

Table B.2-1 lists the address formats specified by the effective-address field.

Table B.2-1 Effective-address field

Code	Notation	Address format	Default bank
00 01 02 03 04 05 06 07	R0 RW0 RL0 R1 RW1 (RL0) R2 RW2 RL1 R3 RW3 (RLL1) R4 RW4 RL2 R5 RW5 (RL2) R6 RW6 RL3 R7 RW7 (RL3)	Register direct ea corresponds to byte, word, and long-word formats 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-incrementing	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 displace-ment	DTB DTB ADB SPB
1C 1D 1E 1F	WRW0+RW7 @RW1+RW7 @PC+disp16 addr16	Register indirect with index Register indirect with index PC indirect with 16-bit displacement Direct address	DTB DTB SPB ADB

B.3 Direct Addressing

In direct addressing, operand values, registers, and addresses are specified directly.

■ Direct addressing

● Immediate addressing (#imm)

Operand values are specified directly (#imm4/#imm8/#imm16/#imm32). Figure B.3-1 shows an example.

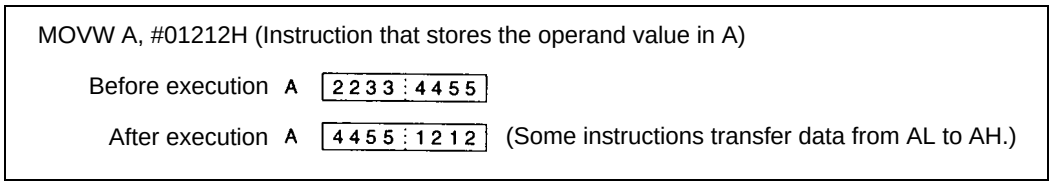


Figure B.3-1 Example of immediate addressing (#imm)

● Register direct addressing

Operand values specify registers directly. The following registers can be specified:

General-purpose register	Byte	R0, R1, R2, R3, R4, R5, R6, R7
	Word	RW0, RW1, RW2, RW3, RW4, RW5, RW6, RW7
	Long-word	RL0, RL1, RL2, RL3
Dedicated registers	Accumulator	A, AL
	Pointer	SP (*1)
	Bank	PCB, DTB, USB, SSB, ADB
	Page	DPR
	Control	PS, CCR, RP, ILM

*1 The SP register functions as a user stack pointer (USP) or system stack pointer (SSP) depending on the S flag bit value indicated in the condition code register (CCR). For branching instructions, the program counter (PC) is not specified as an operand, but is implicitly specified.

Figure B.3-2 shows an example.

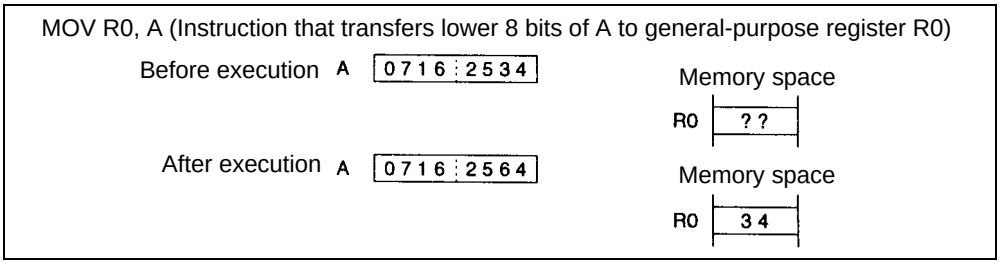


Figure B.3-2 Example of register direct addressing

- **Direct branch addressing (addr16)**

  Branch destination addresses are specified directly by displacement. The displacement is 16 bits and is used to specify a branch destination within the logical space. This method is used for unconditional branching instructions, subroutine call instructions, and software interrupt instructions. Address bits 16 to 23 are specified by the program bank register (PCB). Figure B.3-3 shows an example of direct branch addressing (addr16).

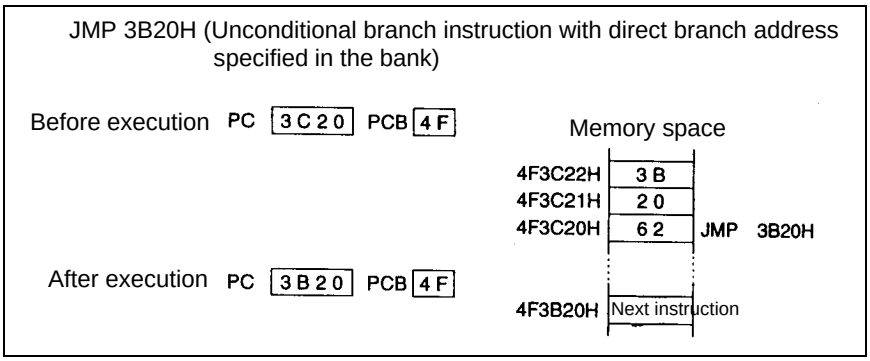


Figure B.3-3 Example of direct branch addressing (addr16)

- **Physical direct branch addressing (addr24)**

  Branch destination addresses are specified directly by displacement. The displacement is 24 bits. This method is used for unconditional branching instructions, subroutine call instructions, and software interrupt instructions. Figure B.3-4 shows an example of physical direct branch addressing (addr24).

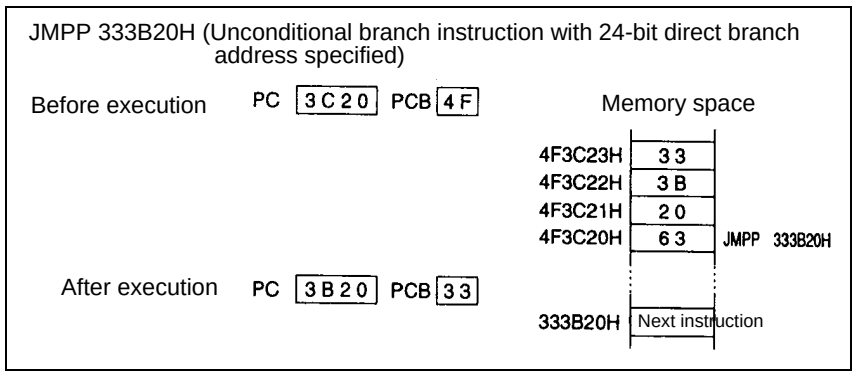


Figure B.3-4 Example of physical direct branch addressing (addr24)

- **I/O direct addressing (io)**

This method specifies memory addresses of the operand directly using an 8-bit displacement. Regardless of the values of the data bank register (DTB) and direct page register (DPR), the I/O space at physical addresses 000000H to 0000FFH is accessed. Prefix instructions designating banks specified before instructions using this addressing method are invalid.
Figure B.3-5 shows an example of I/O direct addressing (io).

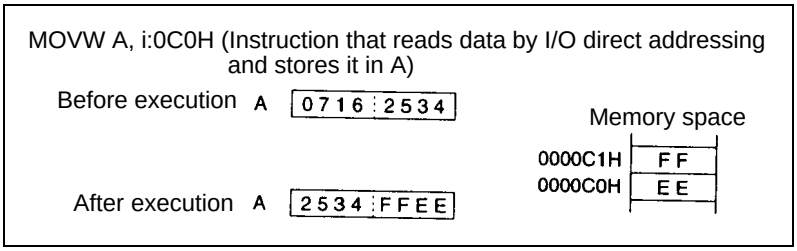


Figure B.3-5 Example of I/O direct addressing (io)

- **Condensed direct addressing (dir)**

This method uses the operand to specify the lower eight bits of a memory address directly. Address bits 8 to 15 are specified by the DPR register. Address bits 16 to 23 are specified by the DTB register. Figure B.3-6 shows an example of condensed direct addressing (dir).

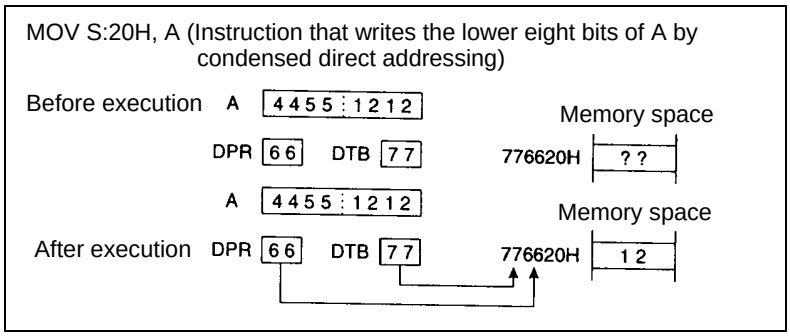


Figure B.3-6 Example of condensed direct addressing (dir)

- **Direct addressing (addr16)**

  Operand values specify lower 16 bits of an memory address directly. Address bits 16 to 23 are specified by the DTB register.

  Prefixed instructions that specify the access space are invalid for this type of addressing.

  Figure B.3-7 shows an example of direct addressing (addr16).

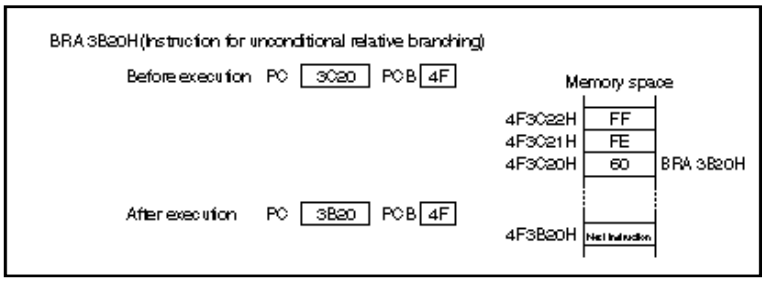


Figure B.3-7 Example of direct addressing (addr16)

- **I/O direct bit addressing (io:bp)**

  This method directly specifies bits within the physical address range from 000000H to 0000FFH.

  The bit location is expressed as :bp, with higher values representing a more significant bit (MSB) and lower values representing a less significant bit (LSB). Figure B.3-8 shows an example of I/O direct bit addressing (io:bp).

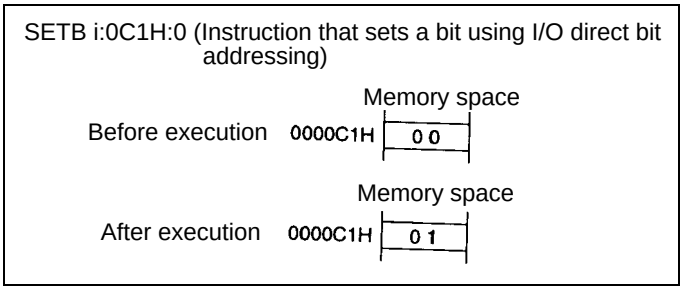


Figure B.3-8 Example of I/O direct bit addressing (io:bp)

- **Condensed direct bit addressing (dir:bp)**

  This method uses the operand to specify the lower eight bits of the memory address. Address bits 8 to 15 are specified by the DPR register and address bits 16 to 23 are specified by the DTB register.

  The bit location is expressed as :bp, with a higher value representing an MSB and a lower value representing a, LSB. Figure B.3-9 shows an example of condensed direct bit addressing (dir:bd).

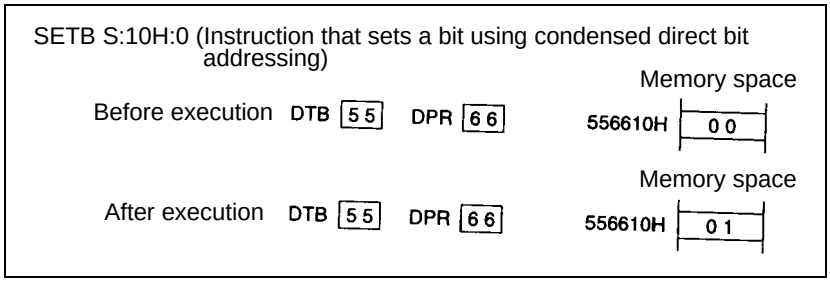


Figure B.3-9 Example of condensed direct bit addressing (dir:bp)

- **Direct bit addressing (addr16:bp)**

  This method directly specifies any bit within the 64-kilobyte area. Address bits 16 to 23 are specified by the DTB register.

  The bit location is expressed as :bp, with a higher value representing an MSB and a lower value representing an LSB. Figure B.3-10 shows an example of direct bit addressing (addr16:bp).

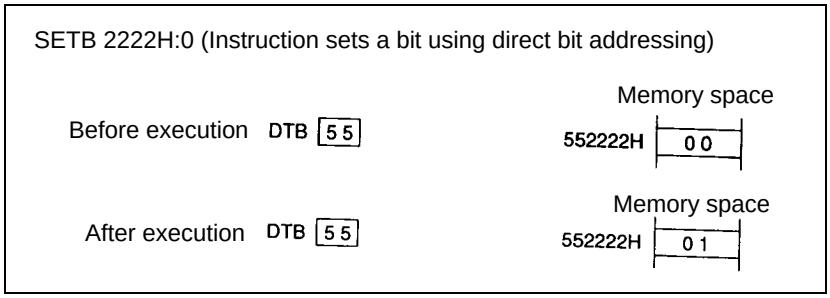


Figure B.3-10 Example of direct bit addressing (addr16:bp)

- **Vector addressing (#vct)**

  In this method, the contents of the specified vector indicates the branch destination address. The data length of the vector number may be either four bits or eight bits. This method is used for subroutine call instructions and software interrupt instructions. Figure B.3-11 shows an example of vector addressing (#vct).

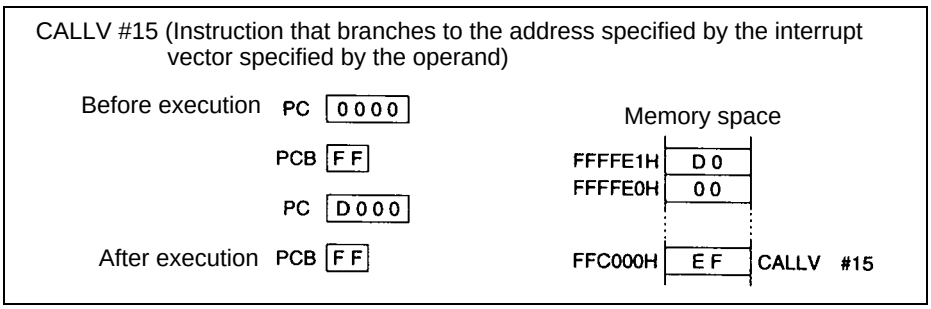


Figure B.3-11 Example of vector addressing (#vct)

Table B.3-1 CALLV vectors

Instruction	Vector address (low)	Vector address (high)
CALLV #0	XXFFFe$_H$	XXFFFF$_H$
CALLV #1	XXFFFC$_H$	XXFFFD$_H$
CALLV #2	XXFFFA$_H$	XXFFFB$_H$
CALLV #3	XXFFF8$_H$	XXFFF9$_H$
CALLV #4	XXFFF6$_H$	XXFFF7$_H$
CALLV #5	XXFFF4$_H$	XXFFF5$_H$
CALLV #6	XXFFF2$_H$	XXFFF3$_H$
CALLV #7	XXFFF0$_H$	XXFFF1$_H$
CALLV #8	XXFFEE$_H$	XXFFEF$_H$
CALLV #9	XXFFEC$_H$	XXFFED$_H$
CALLV #10	XXFFEA$_H$	XXFFEB$_H$
CALLV #11	XXFFE8$_H$	XXFFE9$_H$
CALLV #12	XXFFE6$_H$	XXFFE7$_H$
CALLV #13	XXFFE4$_H$	XXFFE5$_H$
CALLV #14	XXFFE2$_H$	XXFFE3$_H$
CALLV #15	XXFFE0$_H$	XXFFE1$_H$

<Caution> XX indicates a PCB register value.

<Check>

Note that the vector area is shared with INT #vct8 (#0 to #7) when the PCB register value is FFH (see Table B.2).

B.4 Indirect Addressing

In indirect addressing, the operand specifies indirectly the address of the data.

■ Indirect addressing

● Register indirect addressing (@RWj j = 0 to 3)

This type of addressing accesses memory at the address indicated by the contents of general-purpose register RWj. Address bits 16 to 23 are specified by the DTB register if RW0 or RW1 is used, by the system stack bank register (SSB) or user stack bank register (USB) if RW3 is used, and by the additional data bank register (ADB) if RW2 is used. Figure B.4-1 shows an example of register indirect addressing (@RWj, j= 0 to 3).

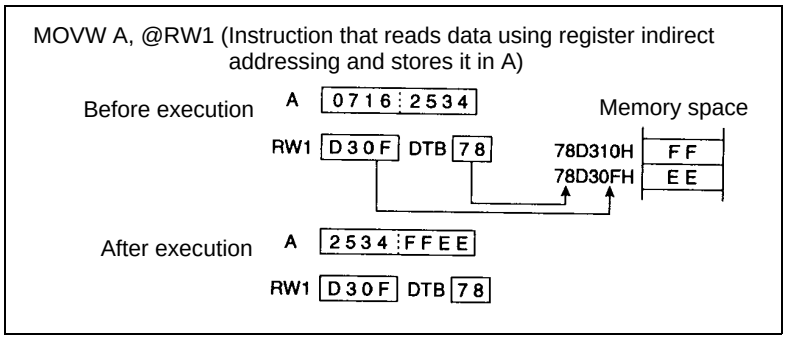


Figure B.4-1 Example of register indirect addressing (@RWj j = 0 to 3)

● Register indirect addressing with post-incrementing (@RWj+ j = 0 to 3)

This type of addressing accesses memory at the address indicated by the contents of general-purpose register RWj. After the operand has been operated on, register RWj is incremented by the length of the operand data (one for byte length, two for word length, four for long-word length). Address bits 16 to 23 are specified by the DTB register if RW0 or RW1 is used, by the SSB or USB register if RW3 is used, and by the ADB register if RW2 is used.

Note that if the results of post-incrementing are the address of the same register that specified the increment, the value to be referenced after incrementing will be the incremented value. Also, if a write instruction is used, the write operation will have priority, so that the register that is supposed to be incremented receives the write data. Figure B.4-2 shows an example of register indirect addressing with post-incrementing (@RWj+, j = 0 to 3).

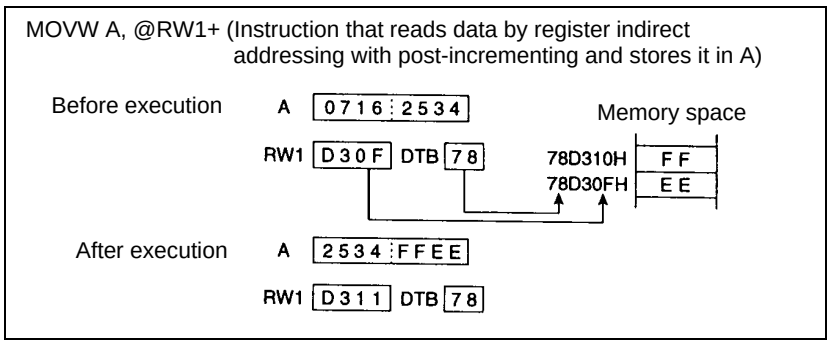


Figure B.4-2 Example of register indirect addressing with post-incrementing (@RWj+ j = 0 to 3)

- **Register indirect addressing with displacement (@RWi+disp8 i = 0 to 7, @RWj+disp16 j = 0 to 3)**

  This type of addressing accesses memory at an address derived from the contents of the general-purpose register RWj with a displacement added. Displacements may be either byte or word length, and are added as signed numerical values. Address bits 16 to 23 are specified by the DTB register if RW0, RW1, RW4, or RW5 is used, by the SSB or USB register if RW3 or RW7 is used, and by the ADB register if RW2 or RW6 is used. Figure B.4-3 shows an example of register indirect addressing with displacement (@RWi+disp8, i = 0 to 7; @RWj+disp16, j = 0 to 3).

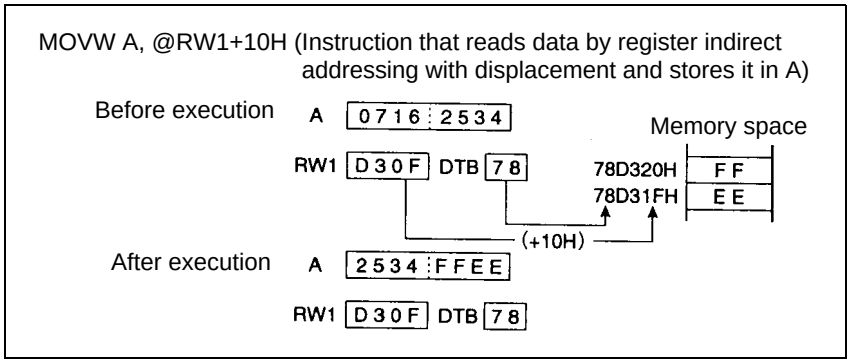


Figure B.4-3 Example of register indirect addressing with displacement (@RWi+disp8 i = 0 to 7, @RWj+disp16 j = 0 to 3)

- **Long-register indirect addressing with displacement (@RLi+disp8 i = 0 to 3)**

  This type of addressing accesses memory at an address derived by using the lower 24 bits of the sum of the contents of general-purpose register RLi plus a displacement. The displacement is 8 bits, and is added as a signed numerical value to the RLi contents. Figure B.4-4 shows an example of long-register indirect addressing with displacement (@RLi+disp8, i = 0 to 3).

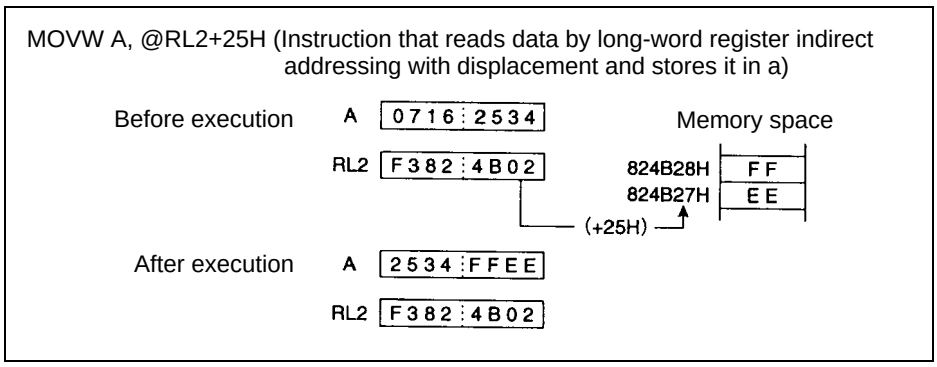


Figure B.4-4 Example of long-word register indirect addressing with displacement(@RLi+disp8 i = 0 to 3)

● **Program counter indirect addressing with displacement (@PC+disp16)**

This type of addressing accesses memory at an address determined by the formula (instruction address+4+disp16). The displacement is word-length data. Address bits 16 to 23 are specified by the PCB register.

Note that the operand addresses of the following instructions are not considered 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 shows an example of program counter indirect addressing with displacement (@PC+disp16).

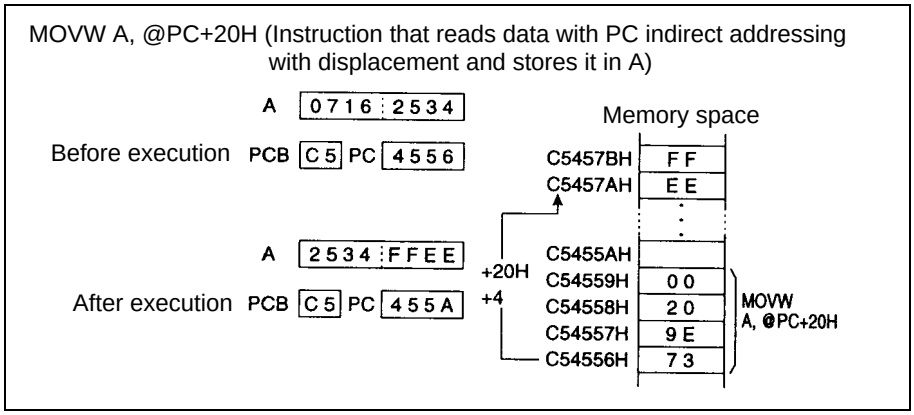


Figure B.4-5 Example of program counter indirect addressing with displacement (@PC+disp16)

- **Register indirect addressing with base index (@RW0+RW7, @RW1+RW7)**

  This type of addressing accesses memory at an address determined by adding the contents of general-purpose register RW7 to RW0 or RW1. Address bits 16 to 23 are specified by the DTB register. Figure B.4-6 shows an example of register indirect addressing with a base index (@RW0+RW7, @RW1+RW7).

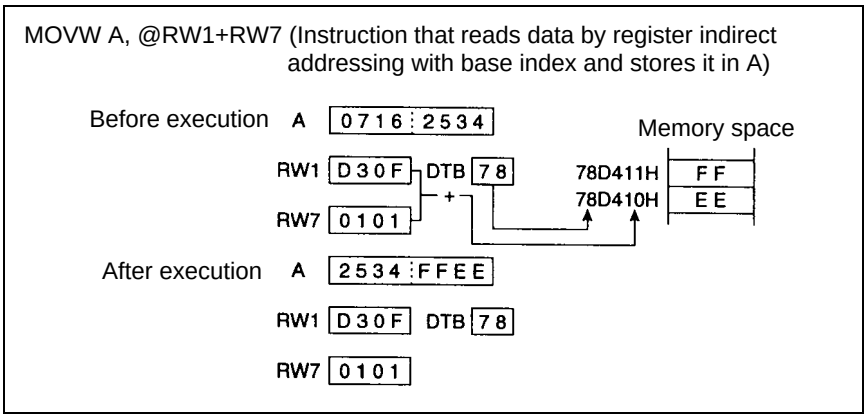


Figure B.4-6 Example of register indirect addressing with base index (@RW0+RW7, @RW1+RW7)

- **Program counter relative branch addressing (rel)**

  A branch destination address is represented as the program counter (PC) value plus an 8-bit displacement. Because the bank register is not incremented or decremented and overflow is ignored if the result is over 16 bits, the result is an address within the 64-kilobyte bank.

  This addressing method is used for unconditional and conditional branch instructions. Address bits 16 to 23 are specified by the PCB register. Figure B.4-7 shows an example of program counter relative branch addressing (rel).

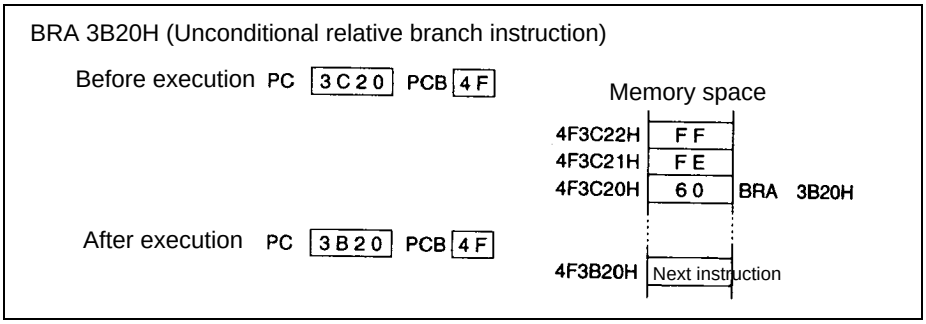


Figure B.4-7 Example of program counter relative branch addressing (rel)

● **Register list (rlst)**

This addressing method specifies registers that are used by push/pop instructions for stack operations. Figure B.3h shows the register list configuration. Figure B.4-8 shows the register list configuration. Figure B.4-9 shows an example of the register list (list).

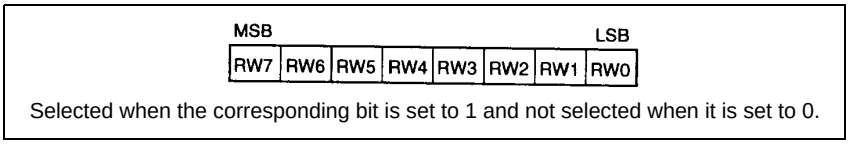


Figure B.4-8 Register list configuration

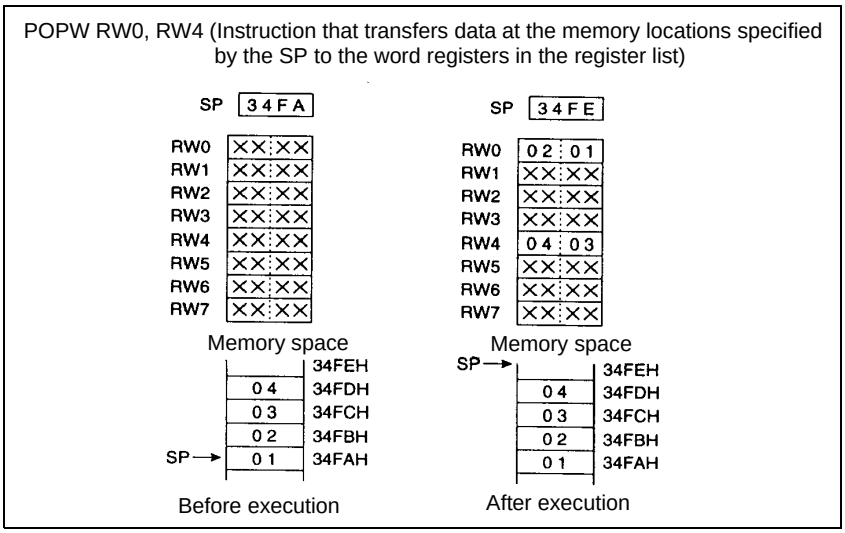


Figure B.4-9 Example of register list (rlst)

● **Accumulator indirect addressing (@A)**

This addressing method accesses memory at the address indicated by the contents (16 bits) of the lower bytes of the accumulator (AL). Address bits 16 to 23 are specified by the DTB register as mnemonics. Figure B.4-10 shows an example of accumulator indirect addressing (@A).

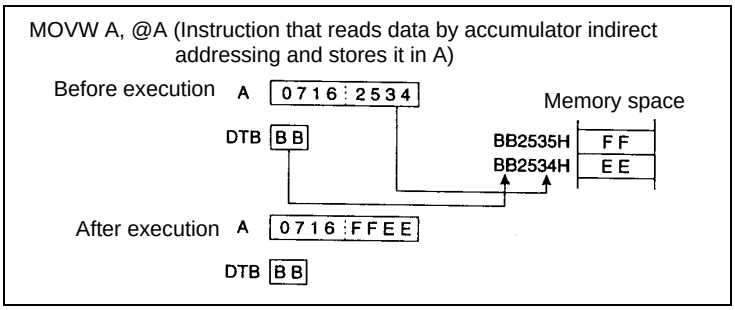


Figure B.4-10 Example of accumulator indirect addressing (@A)

- **Accumulator indirect branch addressing (@A)**

  The 16-bit contents of the lower bytes of the accumulator (AL) indicate the branch destination address, which specifies a branch destination within the bank space. Address bits 16 to 23 are specified by the PCB register. If the Jump Context (JCTX) instruction is used, address bits 16 to 23 are specified by the DTB register.

  This addressing method is used for unconditional branching instructions. Figure B.4-11 shows an example of accumulator indirect branch addressing (@A).

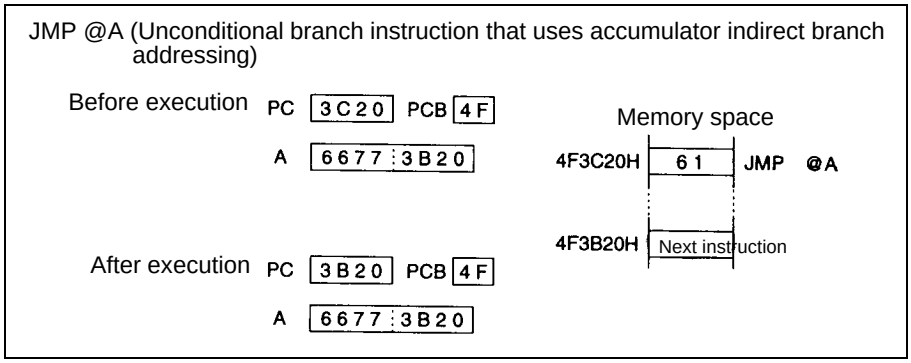


Figure B.4-11 Example of accumulator indirect branch addressing (@A)

- **Indirect designation branch addressing (@ear)**

  In this type of addressing, the branch destination address is the word data at the address specified by the ear parameter. Figure B.4-12 shows an example of indirect designation branch addressing (@ear).

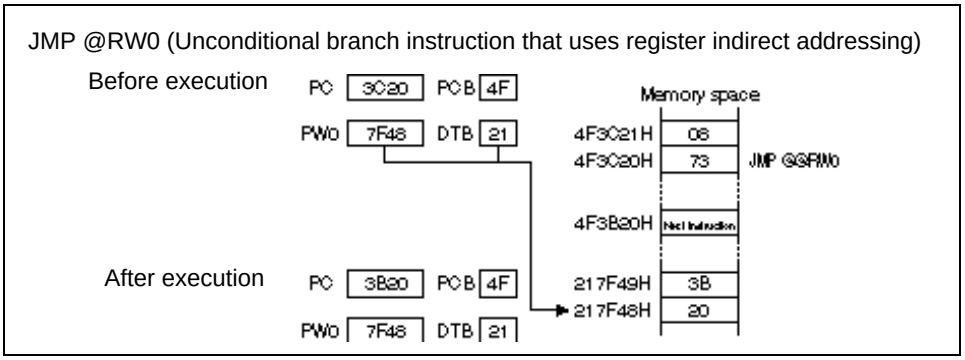


Figure B.4-12 Example of indirect designation branch addressing (@ear)

- **Indirect designation branch addressing (@eam)**

  In this type of addressing, the branch destination address is the word data at the address specified by the eam parameter. Figure B.4-13 shows an example of indirect designation branch addressing (@eam).

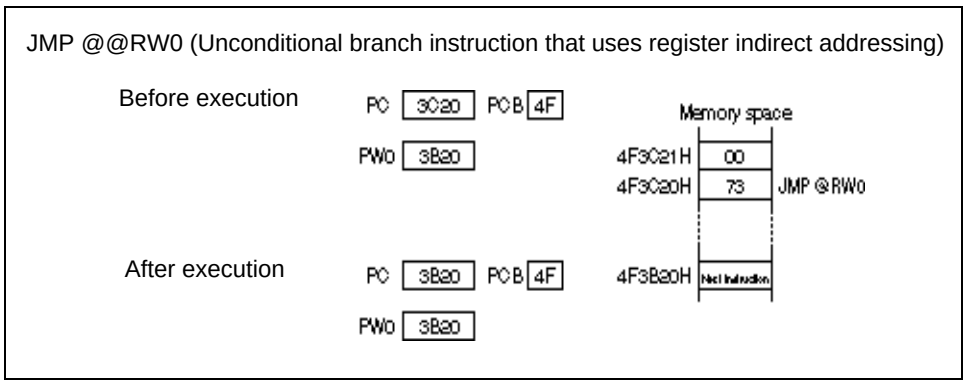


Figure B.4-13 Example of indirect designation branch addressing (@eam)

B.5 Number of Execution Cycles

The number of cycles required to execute an instruction is obtained by adding the number of cycles for the instruction, the compensation value determined by conditions, and the number of cycles for program fetch.

■ Number of execution cycles

The number of cycles required to execute an instruction is obtained by adding the number of cycles for the instruction, the compensation value determined by conditions, and the number of cycles for program fetch.

Because a program in memory connected to the 16-bit bus to the built-in ROM is fetched each time an executing instruction exceeds the word boundary, the number of execution cycles increases if data access is interfered with.

Because a program in memory connected to the 8-bit external data bus is fetched for each byte of the instruction being executed, the number of execution cycles increases if data access is interfered with.

If general-purpose registers, built-in ROM, built-in RAM, built-in I/O, and external data buses are accessed during intermittent operation of the CPU, clocks supplied to the CPU stop for the number of cycles specified by the CG0 and CG1 bits of the low-power consumption mode control register. To calculate the number of cycles needed to execute an instruction during intermittent operation of the CPU, add as a compensation value to the normal number of execution cycles the product of the number of accesses and the number of cycles for the temporary stop.

■ Calculation of the number of execution cycles

Tables B.5-1~ B.5-3 list the number of execution cycles for each instruction and compensation values.

Table B.5-1 Number of execution cycles for each type of addressing

Code	Operand	(a) (*1)	Number of register accesses for type of addressing
		Number of execution cycles for type of addressing	
00 | 07	Ri RWi RLi	Shown in instruction list.	Shown in instruction list.
08 | 0B	@RWj	2	1
0C | 0F	@RWj+	4	2
10 | 17	@RWj+disp8	2	1

Table B.5-1 Number of execution cycles for each type of addressing *(continued)*

Code	Operand	(a) (*1)	Number of register accesses for type of addressing
		Number of execution cycles for type of addressing	
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 ~ (number of cycles) and B (compensation value) in Section B.7, "Reading the Instruction List."

Table B.5-2 Compensation values for calculating the number of execution cycles

Operand	(b) byte (*1)		(c) word (*1)		(d) long-word (*1)	
	Number of cycles	Number of accesses	Number of cycles	Number of accesses	Number of cycles	Number of accesses
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 (*2) 8 bits	+1	1	+4	2	+8	4

*1 (b), (c), and (d) are used for ~ (number of cycles) and B (compensation value) in Section B.7, "Reading the Instruction List."

*2 When the external data bus is used, the number of cycles for waiting due to ready input and automatic ready should be added.

Table B.5-3 Compensation values for calculating number of cycles for program fetch

Instruction	Byte boundary	Word boundary
Internal memory	-	+2
External data bus 16 bits	-	+3
External data bus 8 bits	+3	-

<Cautions>

1. When the external data bus is used, the number of cycles for waiting due to ready input and automatic ready should be added.
2 Not all program fetches delay instruction execution during actual operation. Use the compensation values for calculating values for worst-case situations.

B.6 Effective-address field

Table B.6-1 lists the effective-address field for each code.

■ Effective-address field

Table B.6-1 Effective-address field

Code	Notation			Address format	Number of bytes in address expansion 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 (RLL1) RL2 (RL2) RL3 (RL3)	Register direct ea corresponds to byte, word, and long-word formats 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-incrementing	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 The number of bytes in the address expansion part is used for "+" in the # (number of bytes) column in Section B.7, "Reading the Instruction List."

B.7 Reading the Instruction List

This section explains the items (Table B.7-1) and codes (Table B.7-2) that appear in Section B.7, "List of F²MC-16L Instructions."

■ Explanation of items covered and instruction codes

Table B.7-1 Items covered in instruction list

Item	Description
Mnemonic	Uppercase alphabetic characters, symbols: Shown as they appear in assembler. Lowercase alphabetic characters: Placeholders for assembler. Numerics following lowercase alphabetic characters: Indicates the bit length of instructions.
#	Indicates the number of bytes.
~	Indicates the number of cycles. For alphabetic characters in the items, see Table B.3-1.
RG	Indicates the number of register accesses during instruction execution. Used for calculating compensation values during intermittent operation of the CPU.
B	Indicates compensation values for calculating the actual number of cycles during instruction execution. The actual number of cycles is the sum of the values in the → column.
Operation	Describes how the instruction operates.
LH	Indicates special operations with respect to accumulator bits 08 to 15. Z: Transfers zero. X: Transfers using sign extension. -: No transfer
AH	Indicates special operations with respect to the upper 16 bits of the accumulator. *: Transfers from AL to AH. -: No transfer Z: Transfers 00 to AH. X: Using AL sign extension, transfers 00H or FFH to AH.
I	Indicates status of flags: I (interrupt enable), S (stack), T (sticky bit), N (negative), Z (zero), V (overflow), C (carry). *: Changed as a result of instruction execution. -: Not changed. Z: Set by instruction execution. X: Reset by instruction execution.
S	
T	
N	
Z	
V	
C	

Item	Description
RMW	Indicates a read-modify-write instruction (one which reads data from memory and writes the result in memory with the I instruction). *: Read-modify-write instruction -: Not a read-modify-write instruction <Caution> This type of instruction cannot be used with addresses with a different read/write meaning.

Table B.7-1 Symbols used in the instruction list *(continued)*

Symbol	Meaning
A	2-bit accumulator Length in bits varies with the instruction. Byte: Lower 8 bits of AL Word: 16 bits of AL Long-word: 32 bits of AL and AH.
AH AL	Upper 16 bits of A Lower 16 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
RWi	RW0, RW1, RW2, RW3, RW4, RW5, RW6, RW7
RWj	RW0, RW1, RW2, RW3
RLi	RL0, RL1, RL2, RL3
dir addr16 addr24 ad24 0-15 ad24 16-23	Condensed direct addressing Direct addressing Physical direct addressing addr24 bits 0 to 15 addr24 bits 16 to 23
io	I/O area (000000H to 0000FFH)

Symbol	Meaning
#imm4 #imm8 #imm16 #imm32 ext (imm8)	8-bit immediate data 8-bit immediate data 16-bit immediate data 32-bit immediate data 48-bit immediate data extended to signed 16-bit data
disp8 disp16	8-bit displacement 16-bit displacement
bp	Bit offset value

Table B.7-1 Symbols used in the instruction list *(continued)*

Symbol	Meaning
vct4 vct8	Vector number (0 to 15) Vector number (0 to 255)
( ) b	Bit address
rel ear eam	PC relative branch addressing Effective address designation (codes 00 to 07) Effective address designation (codes 08 to 1F)
rist	Register list

B.8 List of F²MC-16LX Instructions

Tables B.8-1 to B.8-17 list instructions used by F²MC-16LX.

Table B.8-1 Transfer instructions (byte): 41 instructions

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)	-	-	-	-	-	-	-	-	-	-

<Caution> See Table B.5-1, "Number of execution cycles for each type of addressing," and Table B.5-2, "Compensation values for calculating the number of execution cycles," for (a) to (d) in the above table.

Table B.8-2 Transfer instructions (word, long-word): 38 instructions

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×(c)	word (A) ←→ (eam)	-	-	-	-	-	-	-	-	-	-
XCHW RWi,ear	2	7	4	0	word (RWi) ←→ (ear)	-	-	-	-	-	-	-	-	-	-
XCHW RWi,eam	2+	9+(a)	2	2×(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)	-	-	-	-	-	*	*	-	-	-

<Caution> See Table B.5-1, "Number of execution cycles for each type of addressing," and Table B.5-2, "Compensation values for calculating the number of execution cycles," for (a) to (d) in the above table.

Table B.8-3 Addition/subtraction (byte, word, long-word): 42 instructions

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) (hexadecimal)	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) (hexadecimal)	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	-	-	-	-	-	*	*	*	*	-

<Caution> See Table B.5-1, "Number of execution cycles for each type of addressing," and Table B.5-2, "Compensation values for calculating the number of execution cycles," for (a) to (d) in the above table.

Table B.8-4 Increment/decrement (byte, word, long-word): 12 instructions

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
INC	ear	2	3	2	0	byte (ear) ← (ear) + 1	-	-	-	-	-	*	*	*	-	-
INC	eam	2+	5+(a)	0	2×(b)	byte (eam) ← (eam) + 1	-	-	-	-	-	*	*	*	-	*
DEC	ear	2	3	2	0	byte (ear) ← (ear) - 1	-	-	-	-	-	*	*	*	-	-
DEC	eam	2+	5+(a)	0	2×(b)	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	-	-	-	-	-	*	*	*	-	*

<Caution> See Table B.5-1, "Number of execution cycles for each type of addressing," and Table B.5-2, "Compensation values for calculating the number of execution cycles," for (a) to (d) in the above table.

Table B.8-5 Comparison (byte, word, long-word): 11 instructions

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
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	-	-	-	-	-	*	*	*	*	-

<Caution> See Table B.5-1, "Number of execution cycles for each type of addressing," and Table B.5-2, "Compensation values for calculating the number of execution cycles," for (a) to (d) in the above table.

Table B.8-6 Unsigned multiplication/division (word, long-word): 11 instructions

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 (ear)	-	-	-	-	-	-	-	*	*	-
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 for division by zero, 7 for overflow, normally 15

*2 4 for division by zero, 8 for overflow, normally 16

*3 6 + (a) for division by zero, 9 + (a) for overflow, normally 19 + (a)

*4 4 for division by zero, 7 for overflow, normally 22

*5 6 + (a) for division by zero, 8 + (a) for overflow, normally 26 + (a)

*6 (b) for division by zero or overflow, normally 2 x (b)

*7 (c) for division by zero or overflow, normally 2 x (c)

*8 3 when byte(AH) is zero, 7 otherwise

*9 4 when byte(ear) is zero, 8 otherwise

*10 5 + (a) when byte(eam) is zero, 9 + (a) otherwise

*11 3 when word(AH) is zero, 11 otherwise

*12 4 when word(ear) is zero, 12 otherwise

*13 5 + (a) when word(eam) is zero, 13 + (a) otherwise

<Caution> See Table B.5-1, "Number of execution cycles for each type of addressing," and Table B.5-2, "Compensation values for calculating the number of execution cycles," for (a) to (d) in the above table.

Table B.8-7 Logical 1 (byte, word): 39 instructions

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)	-	-	-	-	-	*	*	R	-	*

<Caution> See Table B.5-1, "Number of execution cycles for each type of addressing," and Table B.5-2, "Compensation values for calculating the number of execution cycles," for (a) to (d) in the above table.

Table A.2-1 Logical 2 (long-word): 6 instructions

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
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	-	-

<Caution> See Table B.5-1, "Number of execution cycles for each type of addressing," and Table B.5-2, "Compensation values for calculating the number of execution cycles," for (a) to (d) in the above table.

Table B.8-9 Sign inversion (byte, word): 6 instructions

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
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	2	2	0	word (ear) ← 0 - (ear)	-	-	-	-	-	*	*	*	*	-
NEGW	eam	2+	5+(a)	0	2+(c)	word (eam) ← 0 - (eam)	-	-	-	-	-	*	*	*	*	*

<Caution> See Table B.5-1, "Number of execution cycles for each type of addressing," and Table B.5-2, "Compensation values for calculating the number of execution cycles," for (a) to (d) in the above table.

Table B.8-10 Normalization (long-word): 1 instruction

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NRML	A,R0	2	*1	1	0	long (A) ← Shift to the position where 1 was formerly placed byte (R0) ← Number of shifts at that time	-	-	-	-	-	-	*	-	-	-

*1 4 when all accumulators indicate 0, 6 + (R0) otherwise

<Caution> See Table B.5-1, "Number of execution cycles for each type of addressing," and Table B.5-2, "Compensation values for calculating the number of execution cycles," for (a) to (d) in the above table.

Table B.8-11 Shift instructions (byte, word, long-word): 18 instructions

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
RORC A	2	2	0	0	byte (A) ← Right rotate with carry	-	-	-	-	-	*	*	-	*	-
ROLC A	2	2	0	0	byte (A) ← Left rotate with carry	-	-	-	-	-	*	*	-	*	-
RORC ear	2	3	2	0	byte (ear) ← Right rotate with carry	-	-	-	-	-	*	*	-	*	-
RORC eam	2+	5+(a)	0	2×(b)	byte (eam) ← Right rotate with carry	-	-	-	-	-	*	*	-	*	*
ROLC ear	2	3	2	0	byte (ear) ← Left rotate with carry	-	-	-	-	-	*	*	-	*	-
ROLC eam	2+	5+(a)	0	2×(b)	byte (eam) ← Left rotate with carry	-	-	-	-	-	*	*	-	*	*
ASR A,RO	2	*1	1	0	byte (A) ← Arithmetic right barrel shift (A,R0)	-	-	-	-	*	*	*	-	*	-
LSR A,RO	2	*1	1	0	byte (A) ← Logical right barrel shift (A,R0)	-	-	-	-	*	*	*	-	*	-
LSL A,RO	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 /	1	2	0	0	word (A) ← Logical right shift (A,1 bit)	-	-	-	-	*	R	*	-	*	-
SHRW A	1	2	0	0	word (A) ← Logical left shift (A,1 bit)	-	-	-	-	-	*	*	-	*	-
LSLW A /															
SHLW A	2	*1	1	0	word (A) ← Arithmetic right barrel shift (A,R0)	-	-	-	-	*	*	*	-	*	-
	2	*1	1	0	word (A) ← Logical right barrel shift (A,R0)	-	-	-	-	*	*	*	-	*	-
ASRW A,R0	2	*1	1	0	word (A) ← Logical left barrel shift (A,R0)	-	-	-	-	-	*	*	-	*	-
LSRW A,R0															
LSLW 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 zero, 5 + (R0) otherwise

*2 6 when R0 is zero, 5 + (R0) otherwise

<Caution> See Table B.5-1, "Number of execution cycles for each type of addressing," and Table B.5-2, "Compensation values for calculating the number of execution cycles," for (a) to (d) in the above table.

Table B.8-12 Branching instructions (1): 31 instructions

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
BZ / BEQ	rel	2	*1	0	0	Branch when (Z) = 1	-	-	-	-	-	-	-	-	-	-
BNZ / BNE	rel	2	*1	0	0	Branch when (Z) = 0	-	-	-	-	-	-	-	-	-	-
BC / BLO	rel	2	*1	0	0	Branch when (C) = 1	-	-	-	-	-	-	-	-	-	-
BNC / BHS	rel	2	*1	0	0	Branch when (C) = 0	-	-	-	-	-	-	-	-	-	-
BN	rel	2	*1	0	0	Branch when (N) = 1	-	-	-	-	-	-	-	-	-	-
BP	rel	2	*1	0	0	Branch when (N) = 0	-	-	-	-	-	-	-	-	-	-
BV	rel	2	*1	0	0	Branch when (V) = 1	-	-	-	-	-	-	-	-	-	-
BNV	rel	2	*1	0	0	Branch when (V) = 0	-	-	-	-	-	-	-	-	-	-
BT	rel	2	*1	0	0	Branch when (T) = 1	-	-	-	-	-	-	-	-	-	-
BNT	rel	2	*1	0	0	Branch when (T) = 0	-	-	-	-	-	-	-	-	-	-
BLT	rel	2	*1	0	0	Branch when (V) xor (N) = 1	-	-	-	-	-	-	-	-	-	-
BGE	rel	2	*1	0	0	Branch when (V) xor (N) = 0	-	-	-	-	-	-	-	-	-	-
BLE	rel	2	*1	0	0	Branch when ((V) xor (N)) or (Z) = 1	-	-	-	-	-	-	-	-	-	-
BGT	rel	2	*1	0	0	Branch when ((V) xor (N)) or (Z) = 0	-	-	-	-	-	-	-	-	-	-
BLS	rel	2	*1	0	0	Branch when (C) or (Z) = 1	-	-	-	-	-	-	-	-	-	-
BHI	rel	2	*1	0	0	Branch when (C) or (Z) = 0	-	-	-	-	-	-	-	-	-	-
BRA	rel	2	*1	0	0	Unconditional branching	-	-	-	-	-	-	-	-	-	-
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 *1	2	5	2	0	word (PC) ← (ear), (PCB) ← (ear+2)	-	-	-	-	-	-	-	-	-	-
JMPP	@eam *1	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 *2	2	6	1	(c)	word (PC) ← (ear)	-	-	-	-	-	-	-	-	-	-
CALL	@eam *2	2+	7+(a)	0	2×(c)	word (PC) ← (eam)	-	-	-	-	-	-	-	-	-	-
CALL	addr16 *3	3	6	0	(c)	word (PC) ← addr16	-	-	-	-	-	-	-	-	-	-
CALLV	#vct4 *3	1	7	0	2×(c)	Vector call instruction	-	-	-	-	-	-	-	-	-	-
CALLP	@ear *4	2	10	2	2×(c)	word (PC) ← (ear) 0-15, (PCB) ← (ear)16-23	-	-	-	-	-	-	-	-	-	-
CALLP	@eam *4	2+	11+(a)	0	*2	word (PC) ← (eam) 0-15, (PCB) ← (eam)16-23	-	-	-	-	-	-	-	-	-	-
CALLP	addr24 *5	4	10	0	2×(c)	word (PC) ← addr0-15, (PCB) ← addr16-23	-	-	-	-	-	-	-	-	-	-

*1 4 when branching occurs, 3 otherwise

*2 3 x (c) + (b)

*3 Reads a branch destination address (word).

*4 Write: Saves to stack (word), Read: Reads a branch destination address (word).

*5 Saves to stack (word).

*6 Write: Saves to stack (long-word), Read: Reads a branch destination address (long-word).

*7 Saves to stack (long-word).

<Caution> See Table B.5-1, "Number of execution cycles for each type of addressing," and Table B.5-2, "Compensation values for calculating the number of execution cycles," for (a) to (d) in the above table.

Table B.8-13 Branch instructions (2): 19 instructions

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
CBNE	A,#imm8,rel	3	*1	0	0	Branch when byte (A) ≠ imm8	-	-	-	-	-	*	*	*	*	-
CWBNE	A,#imm16,rel	4	*1	0	0	Branch when word (A)≠ imm16	-	-	-	-	-	*	*	*	*	-
CBNE	ear,#imm8,rel	4	*2	1	0	Branch when byte (ear)≠ imm8	-	-	-	-	-	*	*	*	*	-
CBNE	eam,#imm8,rel	4+	*3	0	(b)	Branch when byte (eam)≠ imm8	-	-	-	-	-	*	*	*	*	-
CWBNE	ear,#imm16,rel	5	*4	1	0	Branch when word (ear)≠ imm16	-	-	-	-	-	*	*	*	*	-
CWBNE	eam,#imm16,rel	5+	*3	0	(c)	Branch when word (eam)≠ imm16	-	-	-	-	-	*	*	*	*	-
DBNZ	ear,rel	3	*5	2	0	Branch when byte (ear)=(ear)-1, (ear)≠ 0	-	-	-	-	-	*	*	*	-	-
DBNZ	eam,rel	3+	*6	2	2×(b)	Branch when byte (eam)=(eam)-1, (eam)≠ 0	-	-	-	-	-	*	*	*	-	*
DWBNZ	ear,rel	3	*5	2	0	Branch when word (ear)=(ear)-1, (ear)≠ 0	-	-	-	-	-	*	*	*	-	-
DWBNZ	eam,rel	3+	*6	2	2×(c)	Branch when word (eam)=(eam)-1, (eam)≠ 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	11	0	*7	Recovery from interrupt	-	-	*	*	*	*	*	*	*	-
LINK	#imm8	2	6	0	(c)	At the entrance of function, save old frame pointers into a stack, set up new frame pointers, reserve area for local pointers.	-	-	-	-	-	-	-	-	-	-
UNLINK		1	5	0	(c)	At the exit of function, recover the old frame pointers from the stack.	-	-	-	-	-	-	-	-	-	-
RET	*1	1	4	0	(c)	Recover from the subroutine.	-	-	-	-	-	-	-	-	-	-
RETP	*2	1	6	0	(d)	Recover from the subroutine.	-	-	-	-	-	-	-	-	-	-

*1 5 when branching occurs, 4 otherwise

*2 13 when branching occurs, 12 otherwise

*3 7 + (a) when branching occurs, 6 + (a) otherwise

*4 8 when branching occurs, 7 otherwise

*5 7 when branching occurs, 6 otherwise

*6 8 + (a) when branching occurs, 7 + (a) otherwise

*7 Returns from stack (word)

*8 Returns from stack (long-word)

*9 Do not use the RWj + addressing mode for a CBNE/CWBNE instruction.

<Caution> See Table B.5-1, "Number of execution cycles for each type of addressing," and Table B.5-2, "Compensation values for calculating the number of execution cycles," for (a) to (d) in the above table.

Table B.8-14 Other control instructions (byte, word, long-word): 28 instructions

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 switching 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,brgl	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 unchange setting	-	-	-	-	-	-	-	-	-	-
CMR		1	1	0	0	Prefix for common register banks	-	-	-	-	-	-	-	-	-	-

*1 PCB,ADB,SSB,USB ----------1

DTB,DPR ----------2

*2 7 + 3 x (pop count) + 2 x (last register number popped), 7 when RLST = 0 (no transfer register)

*3 29 + 3 x (push count) - 3 x (last register number pushed), 8 when RLST = 0 (no transfer register)

*4 (Pop count) x (c) or (push count) x (c)

*5 (Pop count) or (push count)

<Caution> See Table B.5-1, "Number of execution cycles for each type of addressing," and Table B.5-2, "Compensation values for calculating the number of execution cycles," for (a) to (d) in the above table.

Table B.8-15 Bit operation instructions: 21 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 when (dir:bp)b = 0	-	-	-	-	-	-	*	-	-	-
BBC	addr16:bp,rel	5	*1	0	(b)	Branch when (addr16:bp)b = 0	-	-	-	-	-	-	*	-	-	-
BBC	io:bp,rel	4	*2	0	(b)	Branch when (io:bp)b = 0	-	-	-	-	-	-	*	-	-	-
BBS	dir:bp,rel	4	*1	0	(b)	Branch when (dir:bp)b = 1	-	-	-	-	-	-	*	-	-	-
BBS	addr16:bp,rel	5	*1	0	(b)	Branch when (addr16:bp)b = 1	-	-	-	-	-	-	*	-	-	-
BBS	io:bp,rel	4	*2	0	(b)	Branch when (io:bp)b = 1	-	-	-	-	-	-	*	-	-	-
SBBS	addr16:bp,rel	5	*3	0	2×(b)	Branch when (addr16:bp) b = 1, bit = 1	-	-	-	-	-	-	*	-	-	*
WBTS	io:bp	3	*4	0	*5	Wait until (io:bp) b = 1	-	-	-	-	-	-	-	-	-	-
WBTC	io:bp	3	*4	0	*5	Wait until (io:bp) b = 0	-	-	-	-	-	-	-	-	-	-

*1 8 when branching occurs, 7 otherwise

*2 7 when branching occurs, 6 otherwise

*3 10 when conditions are met, 9 otherwise

*4 Undefined number of cycles

*5 Until conditions are met

Table B.8-16 Accumulator operation instructions (byte, word): 6 instructions

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 signed extension	X	-	-	-	-	*	*	-	-	-
EXTW	1	2	0	0	word signed extension	-	X	-	-	-	*	*	-	-	-
ZEXT	1	1	0	0	byte zero extension	Z	-	-	-	-	R	*	-	-	-
ZEXTW	1	1	0	0	word zero extension	-	Z	-	-	-	R	*	-	-	-

<Caution> See Table B.5-1, "Number of execution cycles for each type of addressing," and Table B.5-2, "Compensation values for calculating the number of execution cycles," for (a) to (d) in the above table.

Table B.8-17 String instructions: 10 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 zero, 4 + 7 x (RW0) when counter limit reached, 7n + 5 for a match

*2 5 when RW0 is zero, 4 + 8 x (RW0) otherwise

*3 (b) x (RW0) + (b) x (RW0). When source and destination access different areas, calculate item (b) separately.

*4 (b) x n

*5 2 x (RW0)

*6 (c) x (RW0) + (c) x (RW0). When source and destination access different areas, calculate item (c) separately.

*7 (c) x n

*8 2 x (RW0)

Note:

m: RW0 value (counter value)

n: Loop count

<Caution> See Table B.5-1, "Number of execution cycles for each type of addressing," and Table B.5-2, "Compensation values for calculating the number of execution cycles," for (a) to (d) in the above table.

B.9 Instruction Maps

An F²MC-16L instruction code consists of one or two bytes. Accordingly an instruction map consists of several one-byte or two-byte pages.

Tables B.9-2 to B.9-20 show F²MC-16L instruction maps.

■ Configuration of instruction maps

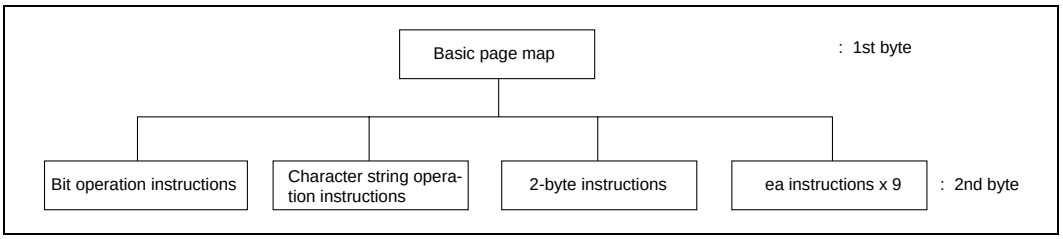
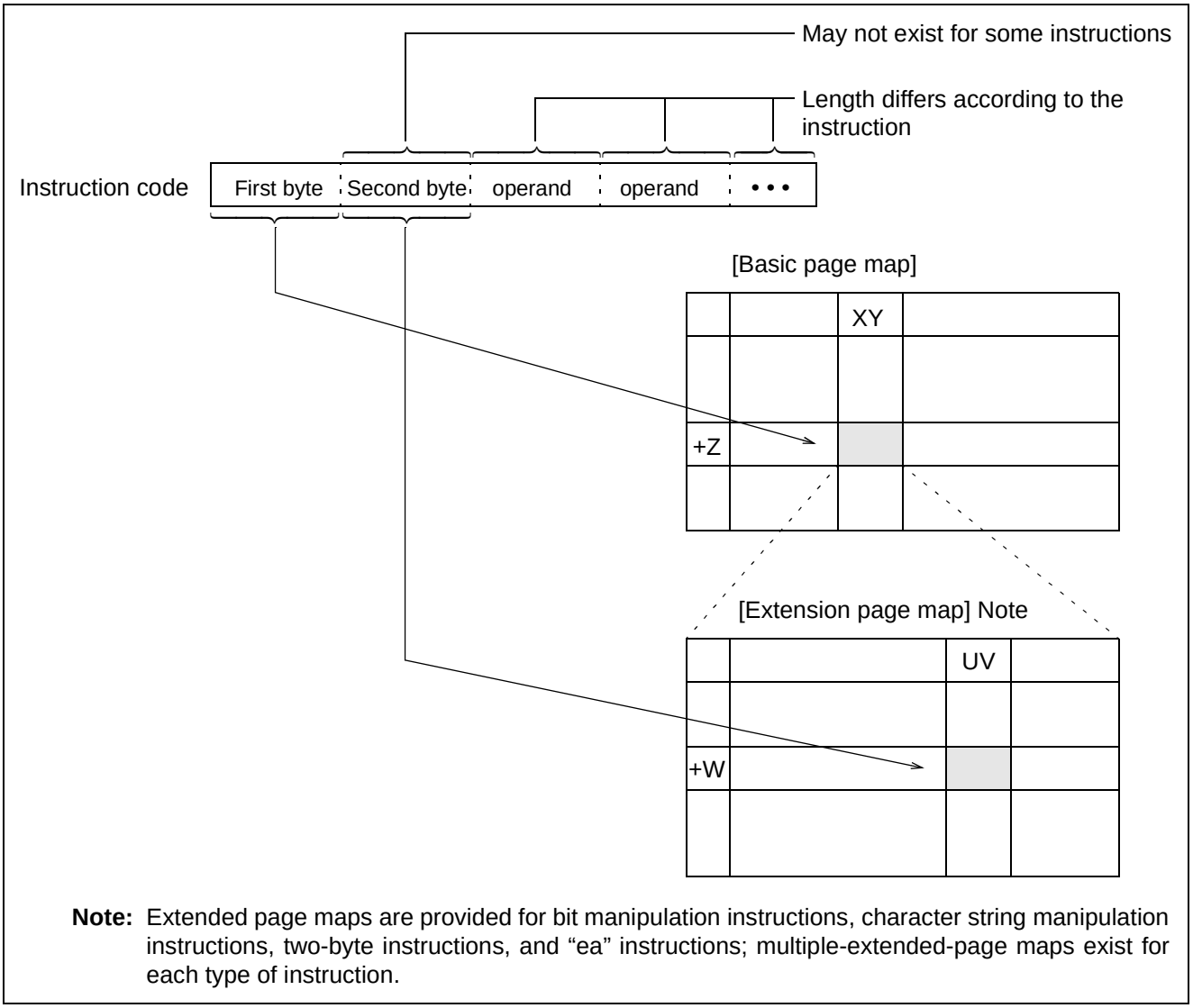


Figure B.9-1 Configuration of instruction maps

When the instruction code is a 1-byte instruction (NOP instruction, etc.), the instruction code is described on the basic page map. When the instruction code is a 2-byte instruction (MOVS, etc.), see the basic page map and check the name of the map where the second byte of the instruction code to be referenced next is described.

Figure B.9-2 shows the relationship between the actual instruction code and instruction maps.



Note: Extended page maps are provided for bit manipulation instructions, character string manipulation instructions, two-byte instructions, and “ea” instructions; multiple-extended-page maps exist for each type of instruction.

Figure B.9-2 Relationship between actual instruction codes and instruction maps

Table B.9-1 lists instruction code examples.

Table B.9-1 Example instruction codes

Instruction	1st byte (from the basic page map)	2nd byte (from the extended page map)
NOP	00 + 0 = 00	-
AND A,#8	30 + 4 = 34	-
MOV A,ADB	60 + F = 6F	00 + 0 = 00
@RW2+d8,#8,rel	70 + 0 = 70	F0 + 2 = F2

Table B.9-2 Basic page map

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	NOP	CMR	ADD A, dir	ADD A, #8	MOV A, dir	MOV A, io	BRA rel	ea instructions (1)	MOV A, Ri	MOV Ri, A	MOV Ri, #8	MOVX A, Ri	MOVX A,@RW/+d8	MOVN A, #4	CALLV #4	BZ /BEQ rel
+1	INT9	NCC	SUB A, dir	SUB A, #8	MOV dir, A	MOV io, A	JMP @A	ea instructions (2)								BNZ/BNE rel
+2	ADDDC A	SUBDC A	ADDC A	SUBC A	MOV A, #8	MOV A, addr16	JMP addr16	ea instructions (3)								BC /BLO rel
+3	NEG A	JCTX @A	CMP A	CMP A, #8	MOVX A, #8	MOV addr 16, A	JMPP addr24	ea instructions (4)								BNC/BHS rel
+4	PCB	EXT	AND CCR, #8	AND A, #8	MOV dir, #8	MOV io, #8	CALL addr16	ea instructions (5)								BN rel
+5	DTB	ZEXT	OR CCR, #8	OR A, #8	MOVX A, dir	MOVX A, io	CALLP addr24	ea instructions (6)								BP rel
+6	ADB	SWAP	DIVU A	XOR A, #8	MOVW A, SP	MOVW io, #16	RETP	ea instructions (7)								BV rel
+7	SPB	ADDSP #8	MULU A	NOT A	MOVW SP, A	MOVX A, addr16	RET	ea instructions (8)								BNV rel
+8	LINK imm#8	ADDL A, #32	ADDW A	ADDW A, #16	MOVW A, dir	MOVW A, io	INT #vct8	ea instructions (9)	MOVW A, RWi	MOVW RWi, A	MOVW RWi, #16	MOVW @RWi+dB	MOVW @RWi+dB, A			BT rel
+9	UNLINK	SUBL A, #32	SUBW A	SUBW A, #16	MOVW dir, A	MOVW io, A	INT addr16	MOVEA RWi, ea								BNT rel
+A	MOV RP, #8	MOV ILM, #8	CBNE A, #8, rel	CWBNE A, #16, rel	MOVW A, #16	MOVW A, addr16	INTP addr24	MOV Ri, ea								BLT rel
+B	NEGW A	CMPL A, #32	CMPW A	CMPW A, #16	MOVL A, #32	MOVW addr16, A	RETI	MOVW RWi, ea								BGE rel
+C	LSLW A	EXTW	ANDW A	ANDW A, #16	PUSHW A	POPW A	Bit operation instructions	MOV ea, Ri								BLE rel
+D		ZEXTW	ORW A	ORW A, #16	PUSHW AH	POPW AH	String operation instructions	MOVW ea, RWi								BGT rel
+E	ASRW A	SWAPW	XORW A	XORW A, #16	PUSHW PS	POPW PS		XCH Ri, ea								BLS rel
+F	LSRW A	ADDSP #16	MULUW A	NOTW A	PUSHW rlst	POPW rlst	Two-byte instructions	XCRW RWi, ea								BHI rel

Table B.9-3 Bit operation instruction map (1st byte = 6C$_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOVB A, io:bp		MOVB io:bp, A		CLRB io:bp		SETB io:bp		BBC io:bp, rel		BBS io:bp, rel		WBTS io:bp		WBTC io:bp	
+1																
+2																
+3																
+4																
+5																
+6																
+7																
+8	MOVB A, dir:bp	MOVB A, addr16:bp	MOVB dir: bp,A	MOVB addr16: bp, A	CLRB dir:bp	CLRB addr16:bp	SETB dir:bp	SETB addr16:bp	BBC dir:bp,rel	BBC ad16:bp, rel	BBS dir:bp, rel	BBS ad 16:bp, rel				SBBS addr16:bp
+9																
+A																
+B																
+C																
+D																
+E																
+F																

Table B.9-4 Character string operation instruction map (1st byte = 6E$_H$)

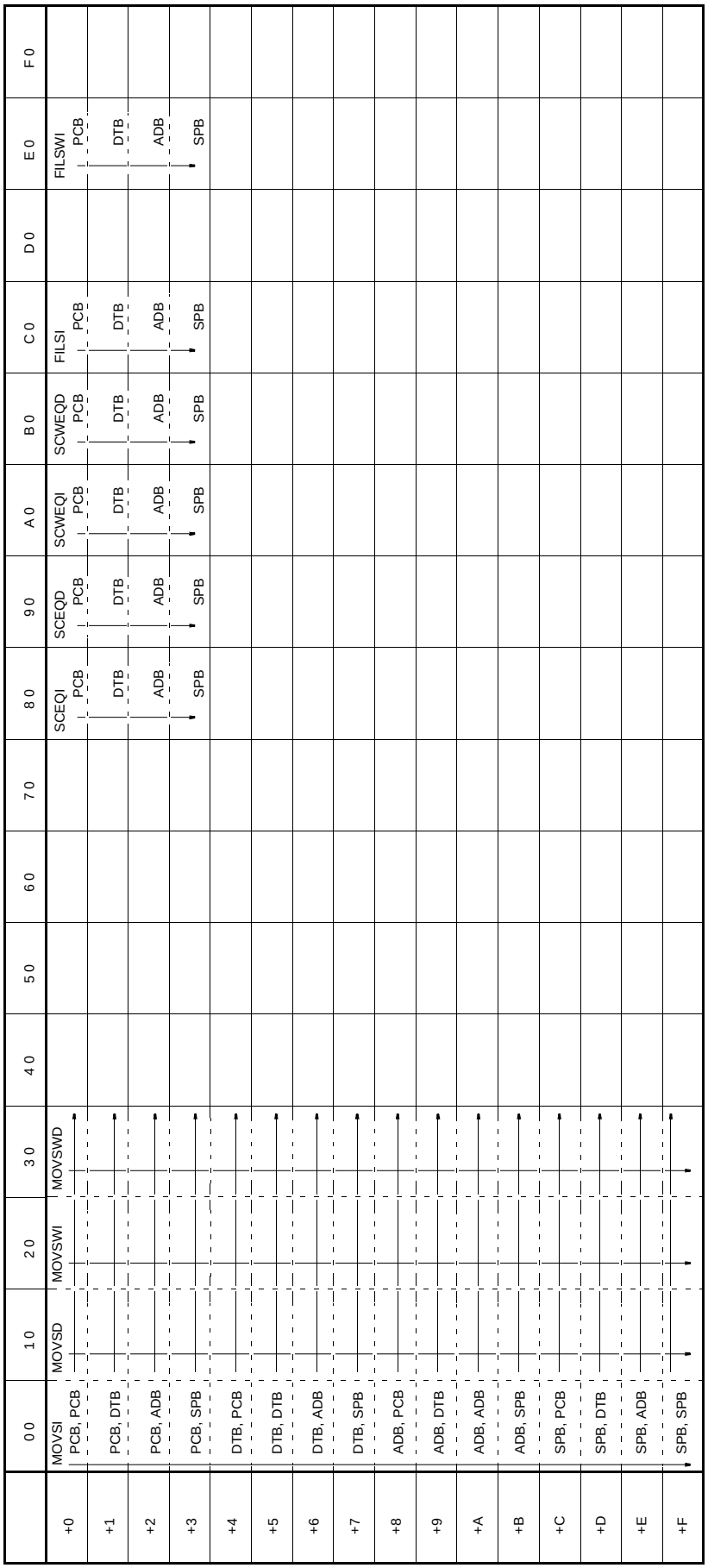


Table B.9-5 2-byte instruction map (1st byte = 6F$_H$)

	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 @RL0+d8, A	MOVW A, @RL0+d8			MUL A								
+9								MULW A								
+A				MOVW @RL1+d8, A	MOVW A, @RL1+d8			DIVU A								
+B																
+C	LSLW A, R0	LSLL A, R0	LSL A, R0	MOVW @RL2+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 @RL3+d8, A	MOVW A, @RL3+d8											
+F	LSRW A, R0	LSRL A, R0	LSR A, R0													

Table B.9-6 ea instruction map (1) (1st byte = 70H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADDL A,	ADDL A,	SUBL A,	SUBL A,	CWBNE RW0, #16, rel	CWBNE @RW0+d8, #16, rel	CMPL A,	CMPL A,	ANDL A,	ANDL A,	ORL A,	ORL A,	XORL A,	XORL A,	CBNE R0, #8, rel	CBNE @RW0+d8, #8, rel
	A, RL0	@RW0+d8	A, RL0	@RW0+d8	#16, rel	#16, rel	A, RL0	@RW0+d8	A, RL0	@RW0+d8	A, RL0	@RW0+d8	A, RL0	@RW0+d8	#8, rel	#8, rel
+1	ADDL A,	ADDL A,	SUBL A,	SUBL A,	CWBNE RW1, #16, rel	CWBNE @RW1+d8, #16, rel	CMPL A,	CMPL A,	ANDL A,	ANDL A,	ORL A,	ORL A,	XORL A,	XORL A,	CBNE R1, #8, rel	CBNE @RW1+d8, #8, rel
	A, RL0	@RW1+d8	A, RL0	@RW1+d8	#16, rel	#16, rel	A, RL0	@RW1+d8	A, RL0	@RW1+d8	A, RL0	@RW1+d8	A, RL0	@RW1+d8	#8, rel	#8, rel
+2	ADDL A,	ADDL A,	SUBL A,	SUBL A,	CWBNE RW2, #16, rel	CWBNE @RW2+d8, #16, rel	CMPL A,	CMPL A,	ANDL A,	ANDL A,	ORL A,	ORL A,	XORL A,	XORL A,	CBNE R2, #8, rel	CBNE @RW2+d8, #8, rel
	A, RL1	@RW2+d8	A, RL1	@RW2+d8	#16, rel	#16, rel	A, RL1	@RW2+d8	A, RL1	@RW2+d8	A, RL1	@RW2+d8	A, RL1	@RW2+d8	#8, rel	#8, rel
+3	ADDL A,	ADDL A,	SUBL A,	SUBL A,	CWBNE RW3, #16, rel	CWBNE @RW3+d8, #16, rel	CMPL A,	CMPL A,	ANDL A,	ANDL A,	ORL A,	ORL A,	XORL A,	XORL A,	CBNE R3, #8, rel	CBNE @RW3+d8, #8, rel
	A, RL1	@RW3+d8	A, RL1	@RW3+d8	#16, rel	#16, rel	A, RL1	@RW3+d8	A, RL1	@RW3+d8	A, RL1	@RW3+d8	A, RL1	@RW3+d8	#8, rel	#8, rel
+4	ADDL A,	ADDL A,	SUBL A,	SUBL A,	CWBNE RW4, #16, rel	CWBNE @RW4+d8, #16, rel	CMPL A,	CMPL A,	ANDL A,	ANDL A,	ORL A,	ORL A,	XORL A,	XORL A,	CBNE R4, #8, rel	CBNE @RW4+d8, #8, rel
	A, RL2	@RW4+d8	A, RL2	@RW4+d8	#16, rel	#16, rel	A, RL2	@RW4+d8	A, RL2	@RW4+d8	A, RL2	@RW4+d8	A, RL2	@RW4+d8	#8, rel	#8, rel
+5	ADDL A,	ADDL A,	SUBL A,	SUBL A,	CWBNE RW5, #16, rel	CWBNE @RW5+d8, #16, rel	CMPL A,	CMPL A,	ANDL A,	ANDL A,	ORL A,	ORL A,	XORL A,	XORL A,	CBNE R5, #8, rel	CBNE @RW5+d8, #8, rel
	A, RL2	@RW5+d8	A, RL2	@RW5+d8	#16, rel	#16, rel	A, RL2	@RW5+d8	A, RL2	@RW5+d8	A, RL2	@RW5+d8	A, RL2	@RW5+d8	#8, rel	#8, rel
+6	ADDL A,	ADDL A,	SUBL A,	SUBL A,	CWBNE RW6, #16, rel	CWBNE @RW6+d8, #16, rel	CMPL A,	CMPL A,	ANDL A,	ANDL A,	ORL A,	ORL A,	XORL A,	XORL A,	CBNE R6, #8, rel	CBNE @RW6+d8, #8, rel
	A, RL3	@RW6+d8	A, RL3	@RW6+d8	#16, rel	#16, rel	A, RL3	@RW6+d8	A, RL3	@RW6+d8	A, RL3	@RW6+d8	A, RL3	@RW6+d8	#8, rel	#8, rel
+7	ADDL A,	ADDL A,	SUBL A,	SUBL A,	CWBNE RW7, #16, rel	CWBNE @RW7+d8, #16, rel	CMPL A,	CMPL A,	ANDL A,	ANDL A,	ORL A,	ORL A,	XORL A,	XORL A,	CBNE R7, #8, rel	CBNE @RW7+d8, #8, rel
	A, RL3	@RW7+d8	A, RL3	@RW7+d8	#16, rel	#16, rel	A, RL3	@RW7+d8	A, RL3	@RW7+d8	A, RL3	@RW7+d8	A, RL3	@RW7+d8	#8, rel	#8, rel
+8	ADDL A,	ADDL A,	SUBL A,	SUBL A,	CWBNE @ RW0, #16, rel	CWBNE @RW0+d16, #16, rel	CMPL A,	CMPL A,	ANDL A,	ANDL A,	ORL A,	ORL A,	XORL A,	XORL A,	CBNE @RW0, #8, rel	CBNE @RW0+d16, #8, rel
	A, @RW0	@RW0+d16	A, @RW0	@RW0+d16	#16, rel	#16, rel	A, @RW0	@RW0+d16	A, @RW0	@RW0+d16	A, @RW0	@RW0+d16	A, @RW0	@RW0+d16	#8, rel	#8, rel
+9	ADDL A,	ADDL A,	SUBL A,	SUBL A,	CWBNE @ RW1, #16, rel	CWBNE @RW1, d16, #16, rel	CMPL A,	CMPL A,	ANDL A,	ANDL A,	ORL A,	ORL A,	XORL A,	XORL A,	CBNE @RW1, #8, rel	CBNE @RW1+d16, #8, rel
	A, @RW1	@RW1+d16	A, @RW1	@RW1+d16	#16, rel	#16, rel	A, @RW1	@RW1+d16	A, @RW1	@RW1+d16	A, @RW1	@RW1+d16	A, @RW1	@RW1+d16	#8, rel	#8, rel
+A	ADDL A,	ADDL A,	SUBL A,	SUBL A,	CWBNE @ RW2, #16, rel	CWBNE @RW2+d16, #16, rel	CMPL A,	CMPL A,	ANDL A,	ANDL A,	ORL A,	ORL A,	XORL A,	XORL A,	CBNE @RW2, #8, rel	CBNE @RW2+d16, #8, rel
	A, @RW2	@RW2+d16	A, @RW2	@RW2+d16	#16, rel	#16, rel	A, @RW2	@RW2+d16	A, @RW2	@RW2+d16	A, @RW2	@RW2+d16	A, @RW2	@RW2+d16	#8, rel	#8, rel
+B	ADDL A,	ADDL A,	SUBL A,	SUBL A,	CWBNE @ RW3, #16, rel	CWBNE @RW3+d16, #16, rel	CMPL A,	CMPL A,	ANDL A,	ANDL A,	ORL A,	ORL A,	XORL A,	XORL A,	CBNE @RW3, #8, rel	CBNE @RW3+d16, #8, rel
	A, @RW3	@RW3+d16	A, @RW3	@RW3+d16	#16, rel	#16, rel	A, @RW3	@RW3+d16	A, @RW3	@RW3+d16	A, @RW3	@RW3+d16	A, @RW3	@RW3+d16	#8, rel	#8, rel
+C	ADDL A,	ADDL A,	SUBL A,	SUBL A,	Prohibit	CWBNE @RW0+RW7	CMPL A,	CMPL A,	ANDL A,	ANDL A,	ORL A,	ORL A,	XORL A,	XORL A,	Prohibit	CBNE @RW0+RW7
	A, @RW0+	@RW0+RW7	A, @RW0+	@RW0+RW7	- - - -	- - - -	A, @RW0+	@RW0+RW7	A, @RW0+	@RW0+RW7	A, @RW0+	@RW0+RW7	A, @RW0+	@RW0+RW7	- - - -	- - - -
+D	ADDL A,	ADDL A,	SUBL A,	SUBL A,	Prohibit	CWBNE @RW1+RW7	CMPL A,	CMPL A,	ANDL A,	ANDL A,	ORL A,	ORL A,	XORL A,	XORL A,	Prohibit	CBNE @RW1+RW7
	A, @RW1+	@RW1+RW7	A, @RW1+	@RW1+RW7	- - - -	- - - -	A, @RW1+	@RW1+RW7	A, @RW1+	@RW1+RW7	A, @RW1+	@RW1+RW7	A, @RW1+	@RW1+RW7	- - - -	- - - -
+E	ADDL A,	ADDL A,	SUBL A,	SUBL A,	Prohibit	CWBNE @PC+d16, #16, rel	CMPL A,	CMPL A,	ANDL A,	ANDL A,	ORL A,	ORL A,	XORL A,	XORL A,	Prohibit	CBNE @PC+d16, #8, rel
	A, @RW2+	@PC+d16	A, @RW2+	@PC+d16	- - - -	- - - -	A, @RW2+	@PC+d16	A, @RW2+	@PC+d16	A, @RW2+	@PC+d16	A, @RW2+	@PC+d16	- - - -	- - - -
+F	ADDL A,	ADDL A,	SUBL A,	SUBL A,	Prohibit	CWBNE addr16, #16, rel	CMPL A,	CMPL A,	ANDL A,	ANDL A,	ORL A,	ORL A,	XORL A,	XORL A,	Prohibit	CBNE addr16, #8, rel
	A, @RW3+	addr16	A, @RW3+	addr16	- - - -	- - - -	A, @RW3+	addr16	A, @RW3+	addr16	A, @RW3+	addr16	A, @RW3+	addr16	- - - -	- - - -

Table B.9-7 ea instruction map (2) (1st byte = 71H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	JMPP @RL0	JMPP @@RW0+d8	CALLP @RL0	CALLP @@RW0+d8	INCL RL0	INCL @RW0+d8	DECL RL0	DECL @RW0+d8	MOVL A, RL0	MOVL A, @RW0+d8	MOVL RL0, A	MOVL @R W0+d8, A	MOV R0, #8	MOV @R W0+d8, #8	MOVEA A, RW0	MOVEA A, @RW0+d8
+1	JMPP @RL0	JMPP @@RW1+d8	CALLP @RL0	CALLP @@RW1+d8	INCL RL0	INCL @RW1+d8	DECL RL0	DECL @RW1+d8	MOVL A, RL0	MOVL A, @RW1+d8	MOVL RL0, A	MOVL @R W1+d8, A	MOV R1, #8	MOV @R W1+d8, #8	MOVEA A, RW1	MOVEA A, @RW1+d8
+2	JMPP @RL1	JMPP @@RW2+d8	CALLP @RL1	CALLP @@RW2+d8	INCL RL1	INCL @RW2+d8	DECL RL1	DECL @RW2+d8	MOVL A, RL1	MOVL A, @RW2+d8	MOVL RL1, A	MOVL @R W2+d8, A	MOV R2, #8	MOV @R W2+d8, #8	MOVEA A, RW2	MOVEA A, @RW2+d8
+3	JMPP @RL1	JMPP @@RW3+d8	CALLP @RL1	CALLP @@RW3+d8	INCL RL1	INCL @RW3+d8	DECL RL1	DECL @RW3+d8	MOVL A, RL1	MOVL A, @RW3+d8	MOVL RL1, A	MOVL @R W3+d8, A	MOV R3, #8	MOV @R W3+d8, #8	MOVEA A, RW3	MOVEA A, @RW3+d8
+4	JMPP @RL2	JMPP @@RW4+d8	CALLP @RL2	CALLP @@RW4+d8	INCL RL2	INCL @RW4+d8	DECL RL2	DECL @RW4+d8	MOVL A, RL2	MOVL A, @RW4+d8	MOVL RL2, A	MOVL @R W4+d8, A	MOV R4, #8	MOV @R W4+d8, #8	MOVEA A, RW4	MOVEA A, @RW4+d8
+5	JMPP @RL2	JMPP @@RW5+d8	CALLP @RL2	CALLP @@RW5+d8	INCL RL2	INCL @RW5+d8	DECL RL2	DECL @RW5+d8	MOVL A, RL2	MOVL A, @RW5+d8	MOVL RL2, A	MOVL @R W5+d8, A	MOV R5, #8	MOV @R W5+d8, #8	MOVEA A, RW5	MOVEA A, @RW5+d8
+6	JMPP @RL3	JMPP @@RW6+d8	CALLP @RL3	CALLP @@RW6+d8	INCL RL3	INCL @RW6+d8	DECL RL3	DECL @RW6+d8	MOVL A, RL3	MOVL A, @RW6+d8	MOVL RL3, A	MOVL @R W6+d8, A	MOV R6, #8	MOV @R W6+d8, #8	MOVEA A, RW6	MOVEA A, @RW6+d8
+7	JMPP @RL3	JMPP @@RW7+d8	CALLP @RL3	CALLP @@RW7+d8	INCL RL3	INCL @RW7+d8	DECL RL3	DECL @RW7+d8	MOVL A, RL3	MOVL A, @RW7+d8	MOVL RL3, A	MOVL @R W7+d8, A	MOV R7, #8	MOV @R W7+d8, #8	MOVEA A, RW7	MOVEA A, @RW7+d8
+8	JMPP @@RW0	JMPP @ @RW0+d16	CALLP @@RW0	CALLP @ @RW0+d16	INCL @RW0	INCL @RW0+d16	DECL @RW0	DECL @RW0+d16	MOVL A, @RW0	MOVL A, @RW0+d16	MOVL @RW0, A	MOVL @R W0+d16, A	MOV @RW0, #8	MOV @R W0+d16, #8	MOVEA A, @RW0	MOVEA A, @RW0+d16
+9	JMPP @@RW1	JMPP @ @RW1+d16	CALLP @@RW1	CALLP @ @RW1+d16	INCL @RW1	INCL @RW1+d16	DECL @RW1	DECL @RW1+d16	MOVL A, @RW1	MOVL A, @RW1+d16	MOVL @RW1, A	MOVL @R W1+d16, A	MOV @RW1, #8	MOV @R W1+d16, #8	MOVEA A, @RW1	MOVEA A, @RW1+d16
+A	JMPP @@RW2	JMPP @ @RW2+d16	CALLP @@RW2	CALLP @ @RW2+d16	INCL @RW2	INCL @RW2+d16	DECL @RW2	DECL @RW2+d16	MOVL A, @RW2	MOVL A, @RW2+d16	MOVL @RW2, A	MOVL @R W2+d16, A	MOV @RW2, #8	MOV @R W2+d16, #8	MOVEA A, @RW2	MOVEA A, @RW2+d16
+B	JMPP @@RW3	JMPP @ @RW3+d16	CALLP @@RW3	CALLP @ @RW3+d16	INCL @RW3	INCL @RW3+d16	DECL @RW3	DECL @RW3+d16	MOVL A, @ RW3	MOVL A, @RW3+d16	MOVL @RW3, A	MOVL @R W3+d16, A	MOV @RW3, #8	MOV @R W3+d16, #8	MOVEA A, @RW3	MOVEA A, @RW3+d16
+C	JMPP @@RW0+	JMPP @ @RW0+RW7	CALLP @@RW0+	CALLP @ @RW0+RW7	INCL @RW0+	INCL @RW0+RW7	DECL @RW0+	DECL @RW0+RW7	MOVL A, @RW0+	MOVL A, @RW0+RW7	MOVL @RW0+, A	MOVL @R W0+RW7, A	MOV @RW0+, #8	MOV @R W0+RW7, #8	MOVEA A, @RW0+	MOVEA A, @RW0+RW7
+D	JMPP @@RW1+	JMPP @ @RW1+RW7	CALLP @@RW1+	CALLP @ @RW1+RW7	INCL @RW1+	INCL @RW1+RW7	DECL @RW1+	DECL @RW1+RW7	MOVL A, @RW1+	MOVL A, @RW1+RW7	MOVL @RW1+, A	MOVL @R W1+RW7, A	MOV @RW1+, #8	MOV @R W1+RW7, #8	MOVEA A, @RW1+	MOVEA A, @RW1+RW7
+E	JMPP @@RW2+	JMPP @ @@PC+d16	CALLP @@RW2+	CALLP @@PC+d16	INCL @RW2+	INCL @PC+d16	DECL @RW2+	DECL @PC+d16	MOVL A, @RW2+	MOVL A, @PC+d16	MOVL @RW2+, A	MOVL @P C+d16, A	MOV @RW2+, #8	MOV @P C+d16, #8	MOVEA A, @RW2+	MOVEA A, @PC+d16
+F	JMPP @@RW3+	JMPP @ @addr16	CALLP @@RW3+	CALLP @ @addr16	INCL @RW3+	INCL addr16	DECL @RW3+	DECL addr16	MOVL A, @RW3+	MOVL A, addr16	MOVL @RW3+, A	MOVL addr16, A	MOV @RW3+, #8	MOV addr16, #8	MOVEA A, @RW3+	MOVEA A, addr16

Table B.9-8 ea instruction map (3) (1st byte = 72H)

	0 0	1 0	2 0	3 0	4 0	5 0	6 0	7 0	8 0	9 0	A 0	B 0	C 0	D 0	E 0	F 0
+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 map (4) (1st byte = 73H)

	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 +d16, #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 +d16, #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 +d16, #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 +d16, #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 map (5) (1st byte = 74H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADD A, R0	ADD A, @RW0+d8	SUB A, R0	SUB A, @RW0+d8	ADDC A, R0	ADDC A, @RW0+d8	CMP A, R0	CMP A, @RW0+d8	AND A, R0	AND A, @RW0+d8	OR A, R0	OR A, @RW0+d8	XOR A, R0	XOR A, @RW0+d8	DBNZ R0, r	DBNZ @ RW0+d8, r
+1	ADD A, R1	ADD A, @RW1+d8	SUB A, R1	SUB A, @RW1+d8	ADDC A, R1	ADDC A, @RW1+d8	CMP A, R1	CMP A, @RW1+d8	AND A, R1	AND A, @RW1+d8	OR A, R1	OR A, @RW1+d8	XOR A, R1	XOR A, @RW1+d8	DBNZ R1, r	DBNZ @ RW1+d8, r
+2	ADD A, R2	ADD A, @RW2+d8	SUB A, R2	SUB A, @RW2+d8	ADDC A, R2	ADDC A, @RW2+d8	CMP A, R2	CMP A, @RW2+d8	AND A, R2	AND A, @RW2+d8	OR A, R2	OR A, @RW2+d8	XOR A, R2	XOR A, @RW2+d8	DBNZ R2, r	DBNZ @ RW2+d8, r
+3	ADD A, R3	ADD A, @RW3+d8	SUB A, R3	SUB A, @RW3+d8	ADDC A, R3	ADDC A, @RW3+d8	CMP A, R3	CMP A, @RW3+d8	AND A, R3	AND A, @RW3+d8	OR A, R3	OR A, @RW3+d8	XOR A, R3	XOR A, @RW3+d8	DBNZ R3, r	DBNZ @ RW3+d8, r
+4	ADD A, R4	ADD A, @RW4+d8	SUB A, R4	SUB A, @RW4+d8	ADDC A, R4	ADDC A, @RW4+d8	CMP A, R4	CMP A, @RW4+d8	AND A, R4	AND A, @RW4+d8	OR A, R4	OR A, @RW4+d8	XOR A, R4	XOR A, @RW4+d8	DBNZ R4, r	DBNZ @ RW4+d8, r
+5	ADD A, R5	ADD A, @RW5+d8	SUB A, R5	SUB A, @RW5+d8	ADDC A, R5	ADDC A, @RW5+d8	CMP A, R5	CMP A, @RW5+d8	AND A, R5	AND A, @RW5+d8	OR A, R5	OR A, @RW5+d8	XOR A, R5	XOR A, @RW5+d8	DBNZ R5, r	DBNZ @ RW5+d8, r
+6	ADD A, R6	ADD A, @RW6+d8	SUB A, R6	SUB A, @RW6+d8	ADDC A, R6	ADDC A, @RW6+d8	CMP A, R6	CMP A, @RW6+d8	AND A, R6	AND A, @RW6+d8	OR A, R6	OR A, @RW6+d8	XOR A, R6	XOR A, @RW6+d8	DBNZ R6, r	DBNZ @ RW6+d8, r
+7	ADD A, R7	ADD A, @RW7+d8	SUB A, R7	SUB A, @RW7+d8	ADDC A, R7	ADDC A, @RW7+d8	CMP A, R7	CMP A, @RW7+d8	AND A, R7	AND A, @RW7+d8	OR A, R7	OR A, @RW7+d8	XOR A, R7	XOR A, @RW7+d8	DBNZ R7, r	DBNZ @ RW7+d8, r
+8	ADD A, @RW0	ADD A, @RW0+d16	SUB A, @RW0	SUB A, @RW0+d16	ADDC A, @RW0	ADDC A, @RW0+d16	CMP A, @RW0	CMP A, @RW0+d16	AND A, @RW0	AND A, @RW0+d16	OR A, @RW0	OR A, @RW0+d16	XOR A, @RW0	XOR A, @RW0+d16	DBNZ @RW0, r	DBNZ RW0+d16, r
+9	ADD A, @RW1	ADD A, @RW1+d16	SUB A, @RW1	SUB A, @RW1+d16	ADDC A, @RW1	ADDC A, @RW1+d16	CMP A, @RW1	CMP A, @RW1+d16	AND A, @RW1	AND A, @RW1+d16	OR A, @RW1	OR A, @RW1+d16	XOR A, @RW1	XOR A, @RW1+d16	DBNZ @RW1, r	DBNZ RW1+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 RW2+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 RW3+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 RW0+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 RW1+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 map (6) (1st byte = 75H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADD R0, A	ADD W0+d8, A	SUB R0, A	SUB W0+d8, A	SUBC A, R0	SUBC A, @RW0+d8	NEG R0	NEG @RW0+d8	AND R0, A	AND W0+d8, A	OR R0, A	OR @RW0+d8, A	XOR R0, A	XOR W0+d8, A	NOT R0	NOT @RW0+d8
+1	ADD R1, A	ADD W1+d8, A	SUB R1, A	SUB W1+d8, A	SUBC A, R1	SUBC A, @RW1+d8	NEG R1	NEG @RW1+d8	AND R1, A	AND W1+d8, A	OR R1, A	OR @RW1+d8, A	XOR R1, A	XOR W1+d8, A	NOT R1	NOT @RW1+d8
+2	ADD R2, A	ADD W2+d8, A	SUB R2, A	SUB W2+d8, A	SUBC A, R2	SUBC A, @RW2+d8	NEG R2	NEG @RW2+d8	AND R2, A	AND W2+d8, A	OR R2, A	OR @RW2+d8, A	XOR R2, A	XOR W2+d8, A	NOT R2	NOT @RW2+d8
+3	ADD R3, A	ADD W3+d8, A	SUB R3, A	SUB W3+d8, A	SUBC A, R3	SUBC A, @RW3+d8	NEG R3	NEG @RW3+d8	AND R3, A	AND W3+d8, A	OR R3, A	OR @RW3+d8, A	XOR R3, A	XOR W3+d8, A	NOT R3	NOT @RW3+d8
+4	ADD R4, A	ADD W4+d8, A	SUB R4, A	SUB W4+d8, A	SUBC A, R4	SUBC A, @RW4+d8	NEG R4	NEG @RW4+d8	AND R4, A	AND W4+d8, A	OR R4, A	OR @RW4+d8, A	XOR R4, A	XOR W4+d8, A	NOT R4	NOT @RW4+d8
+5	ADD R5, A	ADD W5+d8, A	SUB R5, A	SUB W5+d8, A	SUBC A, R5	SUBC A, @RW5+d8	NEG R5	NEG @RW5+d8	AND R5, A	AND W5+d8, A	OR R5, A	OR @RW5+d8, A	XOR R5, A	XOR W5+d8, A	NOT R5	NOT @RW5+d8
+6	ADD R6, A	ADD W6+d8, A	SUB R6, A	SUB W6+d8, A	SUBC A, R6	SUBC A, @RW6+d8	NEG R6	NEG @RW6+d8	AND R6, A	AND W6+d8, A	OR R6, A	OR @RW6+d8, A	XOR R6, A	XOR W6+d8, A	NOT R6	NOT @RW6+d8
+7	ADD R7, A	ADD W7+d8, A	SUB R7, A	SUB W7+d8, A	SUBC A, R7	SUBC A, @RW7+d8	NEG R7	NEG @RW7+d8	AND R7, A	AND W7+d8, A	OR R7, A	OR @RW7+d8, A	XOR R7, A	XOR W7+d8, A	NOT R7	NOT @RW7+d8
+8	ADD @RW0, A	ADD W0+d16, A	SUB @RW0, A	SUB W0+d16, A	SUBC A, @RW0	SUBC A, @RW0+d16	NEG @RW0	NEG @RW0+d16	AND @RW0, A	AND W0+d16, A	OR @RW0, A	OR @RW0+d16, A	XOR @RW0, A	XOR W0+d16, A	NOT @RW0	NOT @RW0+d16
+9	ADD @RW1, A	ADD W1+d16, A	SUB @RW1, A	SUB W1+d16, A	SUBC A, @RW1	SUBC A, @RW1+d16	NEG @RW1	NEG @RW1+d16	AND @RW1, A	AND W1+d16, A	OR @RW1, A	OR @RW1+d16, A	XOR @RW1, A	XOR W1+d16, A	NOT @RW1	NOT @RW1+d16
+A	ADD @RW2, A	ADD RW2+d16, A	SUB @RW2, A	SUB W2+d16, A	SUBC A, @RW2	SUBC A, @RW2+d16	NEG @RW2	NEG @RW2+d16	AND @RW2, A	AND W2+d16, A	OR @RW2, A	OR @RW2+d16, A	XOR @RW2, A	XOR W2+d16, A	NOT @RW2	NOT @RW2+d16
+B	ADD @RW3, A	ADD RW3+d16, A	SUB @RW3, A	SUB W3+d16, A	SUBC A, @RW3	SUBC A, @RW3+d16	NEG @RW3	NEG @RW3+d16	AND @RW3, A	AND W3+d16, A	OR @RW3, A	OR @RW3+d16, A	XOR @RW3, A	XOR W3+d16, A	NOT @RW3	NOT @RW3+d16
+C	ADD @RW0+, A	ADD W0+RW7, A	SUB @RW0+, A	SUB W0+RW7, A	SUBC A, @RW0+	SUBC A, @RW0+RW7	NEG @RW0+	NEG @RW0+RW7	AND @RW0+, A	AND W0+RW7, A	OR @RW0+, A	OR @R @RW0+RW7, A	XOR @RW0+, A	XOR @R W0+RW7, A	NOT @RW0+	NOT @RW0+RW7
+D	ADD @RW1+, A	ADD W1+RW7, A	SUB @RW1+, A	SUB 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 C+d16, A	SUB @RW2+, A	SUB 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 @PC+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 map (7) (1st byte = 76H)

	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 @ 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 map (8) (1st byte = 77H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADDW RW0, A	ADDW @R W0+d8, A	SUBW RW0, A	SUBW @R W0+d8, A	SUBCW A, RW0	SUBCW A, @RW0+d8	NEGW RW0	NEGW @RW0+d8	ANDW RW0, A	ANDW @R W0+d8, A	ORW RW0, A	ORW @R W0+d8, A	XORW RW0, A	XORW @R W0+d8, A	NOTW RW0	NOTW @RW0+d8
+1	ADDW RW1, A	ADDW @R W1+d8, A	SUBW RW1, A	SUBW @R W1+d8, A	SUBCW A, RW1	SUBCW A, @RW1+d8	NEGW RW1	NEGW @RW1+d8	ANDW RW1, A	ANDW @R W1+d8, A	ORW RW1, A	ORW @R W1+d8, A	XORW RW1, A	XORW @R W1+d8, A	NOTW RW1	NOTW @RW1+d8
+2	ADDW RW2, A	ADDW @R W2+d8, A	SUBW RW2, A	SUBW @R W2+d8, A	SUBCW A, RW2	SUBCW A, @RW2+d8	NEGW RW2	NEGW @RW2+d8	ANDW RW2, A	ANDW @R W2+d8, A	OWR RW2, A	ORW @R W2+d8, A	XORW RW2, A	XORW @R W2+d8, A	NOTW RW2	NOTW @RW2+d8
+3	ADDW RW3, A	ADDW @R W3+d8, A	SUBW RW3, A	SUBW @R W3+d8, A	SUBCW A, RW3	SUBCW A, @RW3+d8	NEGW RW3	NEGW @RW3+d8	ANDW RW3, A	ANDW @R W3+d8, A	ORW RW3, A	ORW @R W3+d8, A	XORW RW3, A	XORW @R W3+d8, A	NOTW RW3	NOTW @RW3+d8
+4	ADDW RW4, A	ADDW @R W4+d8, A	SUBW RW4, A	SUBW @R W4+d8, A	SUBCW A, RW4	SUBCW A, @RW4+d8	NEGW RW4	NEGW @RW4+d8	ANDW RW4, A	ANDW @R W4+d8, A	ORW RW4, A	ORW @R W4+d8, A	XORW RW4, A	XORW @R W4+d8, A	NOTW RW4	NOTW @RW4+d8
+5	ADDW RW5, A	ADDW @R W5+d8, A	SUBW RW5, A	SUBW @R W5+d8, A	SUBCW A, RW5	SUBCW A, @RW5+d8	NEGW RW5	NEGW @RW5+d8	ANDW RW5, A	ANDW @R W5+d8, A	ORW RW5, A	ORW @R W5+d8, A	XORW RW5, A	XORW @R W5+d8, A	NOTW RW5	NOTW @RW5+d8
+6	ADDW RW6, A	ADDW @R W6+d8, A	SUBW RW6, A	SUBW @R W6+d8, A	SUBCW A, RW6	SUBCW A, @RW6+d8	NEGW RW6	NEGW @RW6+d8	ANDW RW6, A	ANDW @R W6+d8, A	ORW RW6, A	ORW @R W6+d8, A	XORW RW6, A	XORW @R W6+d8, A	NOTW RW6	NOTW @RW6+d8
+7	ADDW RW7, A	ADDW @R W7+d8, A	SUBW RW7, A	SUBW @R W7+d8, A	SUBCW A,RW7	SUBCW A, @RW7+d8	NEGW RW7	NEGW @RW7+d8	ANDW RW7, A	ANDW @R W7+d8, A	ORW RW7, A	ORW @R W7+d8, A	XORW RW7, A	XORW @R W7+d8, A	NOTW RW7	NOTW @RW7+d8
+8	ADDW @RW0, A	ADDW @R W0+d16, A	SUBW @RW0, A	SUBW @R W0+d16, A	SUBCW A, @RW0	SUBCW A, @RW0+d16	NEGW @RW0	NEGW @RW0+d16	ANDW @RW0, A	ANDW @R W0+d16, A	ORW @RW0, A	ORW @R W0+d16, A	XORW @RW0, A	XORW @R W0+d16, A	NOTW @RW0	NOTW @RW0+d16
+9	ADDW @RW1, A	ADDW @R W1+d16, A	SUBW @RW1, A	SUBW @R W1+d16, A	SUBCW A, @RW1	SUBCW A, @RW1+d16	NEGW @RW1	NEGW @RW1+d16	ANDW @RW1, A	ANDW @R W1+d16, A	ORW @RW1, A	ORW @R W1+d16, A	XORW @RW1, A	XORW @R W1+d16, A	NOTW @RW1	NOTW @RW1+d16
+A	ADDW @RW2, A	ADDW @R W2+d16, A	SUBW @RW2, A	SUBW @R W2+d16, A	SUBCW A, @RW2	SUBCW A, @RW2+d16	NEGW @RW2	NEGW @RW2+d16	ANDW @RW2, A	ANDW @R W2+d16, A	ORW @RW2, A	ORW @R W2+d16, A	XORW @RW2, A	XORW @R W2+d16, A	NOTW @RW2	NOTW @RW2+d16
+B	ADDW @RW3, A	ADDW @R W3+d16, A	SUBW @RW3, A	SUBW @R W3+d16, A	SUBCW A, @RW3	SUBCW A, @RW3+d16	NEGW @RW3	NEGW @RW3+d16	ANDW @RW3, A	ANDW @R W3+d16, A	ORW @RW3, A	ORW @R W3+d16, A	XORW @RW3, A	XORW @R W3+d16, A	NOTW @RW3	NOTW @RW3+d16
+C	ADDW @RW0+, A	ADDW @R W0+RW7	SUBW @RW0+, A	SUBW @R W0+RW7, A	SUBCW A, @RW0+	SUBCW A, @RW0+RW7	NEGW @RW0+	NEGW @RW0+RW7	ANDW @RW0+, A	ANDW @R W0+RW7, A	ORW @RW0+, A	ORW @R W0+RW7, A	XORW @RW0+, A	XORW @R W0+RW7, A	NOTW @RW0+	NOTW @RW0+RW7
+D	ADDW @RW1+, A	ADDW @R W1+RW7	SUBW @RW1+, A	SUBW @R W1+RW7, A	SUBCW A, @RW1+	SUBCW A, @RW1+RW7	NEGW @RW1+	NEGW @RW1+RW7	ANDW @RW1+, A	ANDW @R W1+RW7, A	ORW @RW1+, A	ORW @R W1+RW7, A	XORW @RW1+, A	XORW @R W1+RW7, A	NOTW @RW1+	NOTW @RW1+RW7
+E	ADDW @RW2+, A	ADDW @P C+d16, A	SUBW @RW2+, A	SUBW @PC+d16, A	SUBCW A, @RW2+	SUBCW A, @PC+d16	NEGW @RW2+	NEGW @PC+d16	ANDW @RW2+, A	ANDW @P C+d16, A	ORW @RW2+, A	ORW @P C+d16, A	XORW @RW2+, A	XORW @P C+d16, A	NOTW @RW2+	NOTW @PC+d16
+F	ADDW @RW3+, A	ADDW addr16, A	SUBW @RW3+, A	SUBW addr16, A	SUBCW A, @RW3+	SUBCW A, addr16	NEGW @RW3+	NEGW addr16	ANDW @RW3+, A	ANDW addr16, A	ORW @RW3+, A	ORW addr16, A	XORW @RW3+, A	XORW addr16, A	NOTW @RW3+	NOTW addr16

Table B.9-14 ea instruction map (9) (1st byte = 78H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MULU A, R0	MULU A, @RW0+d8	MULUW A, R0	MULUW A, @RW0+d8	MUL A, R0	MUL A, @RW0+d8	MULW A, R0	MULW A, @RW0+d8	DIVU A, R0	DIVU A, @RW0+d8	DIVUW A, R0	DIVUW A, @RW0+d8	DIV A, R0	DIV A, @RW0+d8	DIVW A, R0	DIVW A, @RW0+d8
+1	MULU A, R1	MULU A, @RW1+d8	MULUW A, R1	MULUW A, @RW1+d8	MUL A, R1	MUL A, @RW1+d8	MULW A, R1	MULW A, @RW1+d8	DIVU A, R1	DIVU A, @RW1+d8	DIVUW A, R1	DIVUW A, @RW1+d8	DIV A, R1	DIV A, @RW1+d8	DIVW A, R1	DIVW A, @RW1+d8
+2	MULU A, R2	MULU A, @RW2+d8	MULUW A, R2	MULUW A, @RW2+d8	MUL A, R2	MUL A, @RW2+d8	MULW A, R2	MULW A, @RW2+d8	DIVU A, R2	DIVU A, @RW2+d8	DIVUW A, R2	DIVUW A, @RW2+d8	DIV A, R2	DIV A, @RW2+d8	DIVW A, R2	DIVW A, @RW2+d8
+3	MULU A, R3	MULU A, @RW3+d8	MULUW A, R3	MULUW A, @RW3+d8	MUL A, R3	MUL A, @RW3+d8	MULW A, R3	MULW A, @RW3+d8	DIVU A, R3	DIVU A, @RW3+d8	DIVUW A, R3	DIVUW A, @RW3+d8	DIV A, R3	DIV A, @RW3+d8	DIVW A, R3	DIVW A, @RW3+d8
+4	MULU A, R4	MULU A, @RW4+d8	MULUW A, R4	MULUW A, @RW4+d8	MUL A, R4	MUL A, @RW4+d8	MULW A, R4	MULW A, @RW4+d8	DIVU A, R4	DIVU A, @RW4+d8	DIVUW A, R4	DIVUW A, @RW4+d8	DIV A, R4	DIV A, @RW4+d8	DIVW A, R4	DIVW A, @RW4+d8
+5	MULU A, R5	MULU A, @RW5+d8	MULUW A, R5	MULUW A, @RW5+d8	MUL A, R5	MUL A, @RW5+d8	MULW A, R5	MULW A, @RW5+d8	DIVU A, R5	DIVU A, @RW5+d8	DIVUW A, R5	DIVUW A, @RW5+d8	DIV A, R5	DIV A, @RW5+d8	DIVW A, R5	DIVW A, @RW5+d8
+6	MULU A, R6	MULU A, @RW6+d8	MULUW A, R6	MULUW A, @RW6+d8	MUL A, R6	MUL A, @RW6+d8	MULW A, R6	MULW A, @RW6+d8	DIVU A, R6	DIVU A, @RW6+d8	DIVUW A, R6	DIVUW A, @RW6+d8	DIV A, R6	DIV A, @RW6+d8	DIVW A, R6	DIVW A, @RW6+d8
+7	MULU A, R7	MULU A, @RW7+d8	MULUW A, R7	MULUW A, @RW7+d8	MUL A, R7	MUL A, @RW7+d8	MULW A, R7	MULW A, @RW7+d8	DIVU A, R7	DIVU A, @RW7+d8	DIVUW A, R7	DIVUW A, @RW7+d8	DIV A, R7	DIV A, @RW7+d8	DIVW A, R7	DIVW A, @RW7+d8
+8	MULU 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, @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, @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, @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, @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, @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, @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, @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 map (1st byte = 79H)

	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 W0, @RW0+	MOVEA RW0, @RW0+RW7	MOVEA R W1, @RW0+	MOVEA RW1, @RW0+RW7	MOVEA R W2, @RW0+	MOVEA RW2, @RW0+RW7	MOVEA R W3, @RW0+	MOVEA RW3, @RW0+RW7	MOVEA R W4, @RW0+	MOVEA RW4, @RW0+RW7	MOVEA R W5, @RW0+	MOVEA RW5, @RW0+RW7	MOVEA R W6, @RW0+	MOVEA RW6, @RW0+RW7	MOVEA R W7, @RW0+	MOVEA RW7, @RW0+RW7
+D	MOVEA R W0, @RW1+	MOVEA RW0, @RW1+RW7	MOVEA R W1, @RW1+	MOVEA RW1, @RW1+RW7	MOVEA R W2, @RW1+	MOVEA RW2, @RW1+RW7	MOVEA R W3, @RW1+	MOVEA RW3, @RW1+RW7	MOVEA R W4, @RW1+	MOVEA RW4, @RW1+RW7	MOVEA R W5, @RW1+	MOVEA RW5, @RW1+RW7	MOVEA R W6, @RW1+	MOVEA RW6, @RW1+RW7	MOVEA R W7, @RW1+	MOVEA RW7, @RW1+RW7
+E	MOVEA R W0, @RW2+	MOVEA RW0, @PC+d16	MOVEA R W1, @RW2+	MOVEA RW1, @PC+d16	MOVEA R W2, @RW2+	MOVEA RW2, @PC+d16	MOVEA R W3, @RW2+	MOVEA RW3, @PC+d16	MOVEA R W4, @RW2+	MOVEA RW4, @PC+d16	MOVEA R W5, @RW2+	MOVEA RW5, @PC+d16	MOVEA R W6, @RW2+	MOVEA RW6, @PC+d16	MOVEA R W7, @RW2+	MOVEA RW7, @PC+d16
+F	MOVEA R W0, @RW3+	MOVEA RW0, addr16	MOVEA R W1, @RW3+	MOVEA RW1, addr16	MOVEA R W2, @RW3+	MOVEA RW2, addr16	MOVEA R W3, @RW3+	MOVEA RW3, addr16	MOVEA R W4, @RW3+	MOVEA RW4, addr16	MOVEA R W5, @RW3+	MOVEA RW5, addr16	MOVEA R W6, @RW3+	MOVEA RW6, addr16	MOVEA R W7, @RW3+	MOVEA RW7, addr16

Table B.9-16 MOV Ri,ea instruction map (1st byte = 7A$_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOV 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, 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, 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, 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, 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 map (1st 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 R0, @RW0+	MOVW RW0, @RW0+RW7	MOVW R1, @RW0+	MOVW RW1, @RW0+RW7	MOVW R2, @RW0+	MOVW RW2, @RW0+RW7	MOVW R3, @RW0+	MOVW RW3, @RW0+RW7	MOVW R4, @RW0+	MOVW RW4, @RW0+RW7	MOVW R5, @RW0+	MOVW RW5, @RW0+RW7	MOVW R6, @RW0+	MOVW RW6, @RW0+RW7	MOVW R7, @RW0+	MOVW RW7, @RW0+RW7
+D	MOVW R0, @RW1+	MOVW RW0, @RW1+RW7	MOVW R1, @RW1+	MOVW RW1, @RW1+RW7	MOVW R2, @RW1+	MOVW RW2, @RW1+RW7	MOVW R3, @RW1+	MOVW RW3, @RW1+RW7	MOVW R4, @RW1+	MOVW RW4, @RW1+RW7	MOVW R5, @RW1+	MOVW RW5, @RW1+RW7	MOVW R6, @RW1+	MOVW RW6, @RW1+RW7	MOVW R7, @RW1+	MOVW RW7, @RW1+RW7
+E	MOVW R0, @RW2+	MOVW RW0, @PC+d16	MOVW R1, @RW2+	MOVW RW1, @PC+d16	MOVW R2, @RW2+	MOVW RW2, @PC+d16	MOVW R3, @RW2+	MOVW RW3, @PC+d16	MOVW R4, @RW2+	MOVW RW4, @PC+d16	MOVW R5, @RW2+	MOVW RW5, @PC+d16	MOVW R6, @RW2+	MOVW RW6, @PC+d16	MOVW R7, @RW2+	MOVW RW7, @PC+d16
+F	MOVW R0, @RW3+	MOVW RW0, addr16	MOVW R1, @RW3+	MOVW RW1, addr16	MOVW R2, @RW3+	MOVW RW2, addr16	MOVW R3, @RW3+	MOVW RW3, addr16	MOVW R4, @RW3+	MOVW RW4, addr16	MOVW R5, @RW3+	MOVW RW5, addr16	MOVW R6, @RW3+	MOVW RW6, addr16	MOVW R7, @RW3+	MOVW RW7, addr16

Table B.9-18 MOV ea,Ri instruction map (1st byte = 7CH)

	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 W0+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 W1+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 W2+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 W3+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 W0+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 W1+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 PC+d16, R0	MOV @RW2+, R1	MOV PC+d16, R1	MOV @RW2+, R2	MOV PC+d16, R2	MOV @RW2+, R3	MOV PC+d16, R3	MOV @RW2+, R4	MOV PC+d16, R4	MOV @RW2+, R5	MOV PC+d16, R5	MOV @RW2+, R6	MOV PC+d16, R6	MOV @RW2+, R7	MOV PC+d16, R7
+F	MOV @RW3+, R0	MOV addr16, R0	MOV @RW3+, R1	MOV addr16, R1	MOV @RW3+, R2	MOV addr16, R2	MOV @RW3+, R3	MOV addr16, R3	MOV @RW3+, R4	MOV addr16, R4	MOV @RW3+, R5	MOV addr16, R5	MOV @RW3+, R6	MOV addr16, R6	MOV @RW3+, R7	MOV addr16, R7

Table B.9-19 MOVW ea,Rwi instruction map (1st 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 @R 1+d8, RW1	MOVW RW1, RW2	MOVW @R 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	MOVW RW7, RW4	MOVW @RW 7+d8, RW4	MOVW RW7, RW5	MOVW @RW 7+d8, RW5	MOVW RW7, RW6	MOVW @RW 7+d8, RW6	MOVW 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 map (1st byte = 7E$_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	XCH R0, R0	XCH R0, @RW0+d8	XCH R1, R0	XCH R1, @RW0+d8	XCH R2, R0	XCH R2, @RW0+d8	XCH R3, R0	XCH R3, @RW0+d8	XCH R4, R0	XCH R4, @RW0+d8	XCH R5, R0	XCH R5, @RW0+d8	XCH R6, R0	XCH R6, @RW0+d8	XCH R7, R0	XCH R7, @RW0+d8
+1	XCH R0, R1	XCH R0, @RW1+d8	XCH R1, R1	XCH R1, @RW1+d8	XCH R2, R1	XCH R2, @RW1+d8	XCH R3, R1	XCH R3, @RW1+d8	XCH R4, R1	XCH R4, @RW1+d8	XCH R5, R1	XCH R5, @RW1+d8	XCH R6, R1	XCH R6, @RW1+d8	XCH R7, R1	NOTW R7, @RW1+d8
+2	XCH R0, R2	XCH R0, @RW2+d8	XCH R1, R2	XCH R1, @RW2+d8	XCH R2, R2	XCH R2, @RW2+d8	XCH R3, R2	XCH R3, @RW2+d8	XCH R4, R2	XCH R4, @RW2+d8	XCH R5, R2	XCH R5, @RW2+d8	XCH R6, R2	XCH R6, @RW2+d8	XCH R7, R2	XCH R7, @RW2+d8
+3	XCH R0, R3	XCH R0, @RW3+d8	XCH R1, R3	XCH R1, @RW3+d8	XCH R2, R3	XCH R2, @RW3+d8	XCH R3, R3	XCH R3, @RW3+d8	XCH R4, R3	XCH R4, @RW3+d8	XCH R5, R3	XCH R5, @RW3+d8	XCH R6, R3	XCH R6, @RW3+d8	XCH R7, R3	NOTW R7, @RW3+d8
+4	XCH R0, R4	XCH R0, @RW4+d8	XCH R1, R4	XCH R1, @RW4+d8	XCH R2, R4	XCH R2, @RW4+d8	XCH R3, R4	XCH R3, @RW4+d8	XCH R4, R4	XCH R4, @RW4+d8	XCH R5, R4	XCH R5, @RW4+d8	XCH R6, R4	XCH R6, @RW4+d8	XCH R7, R4	NOTW R7, @RW4+d8
+5	XCH R0, R5	XCH R0, @RW5+d8	XCH R1, R5	XCH R1, @RW5+d8	XCH R2, R5	XCH R2, @RW5+d8	XCH R3, R5	XCH R3, @RW5+d8	XCH R4, R5	XCH R4, @RW5+d8	XCH R5, R5	XCH R5, @RW5+d8	XCH R6, R5	XCH R6, @RW5+d8	XCH R7, R5	XCH R7, @RW5+d8
+6	XCH R0, R6	XCH R0, @RW6+d8	XCH R1, R6	XCH R1, @RW6+d8	XCH R2, R6	XCH R2, @RW6+d8	XCH R3, R6	XCH R3, @RW6+d8	XCH R4, R6	XCH R4, @RW6+d8	XCH R5, R6	XCH R5, @RW6+d8	XCH R6, R6	XCH R6, @RW6+d8	XCH R7, R6	XCH R7, @RW6+d8
+7	XCH R0, R7	XCH R0, @RW7+d8	XCH R1, R7	XCH R1, @RW7+d8	XCH R2, R7	XCH R2, @RW7+d8	XCH R3, R7	XCH R3, @RW7+d8	XCH R4, R7	XCH R4, @RW7+d8	XCH R5, R7	XCH R5, @RW7+d8	XCH R6, R7	XCH R6, @RW7+d8	XCH R7, R7	XCH R7, @RW7+d8
+8	XCH R0, @RW0	XCH R0, @RW0+d16	XCH R1, @RW0	XCH R1, @RW0+d16	XCH R2, @RW0	XCH R2, @RW0+d16	XCH R3, @RW0	XCH R3, @RW0+d16	XCH R4, @RW0	XCH R4, @RW0+d16	XCH R5, @RW0	XCH R5, @RW0+d16	XCH R6, @RW0	XCH R6, @RW0+d16	XCH R7, @RW0	XCH R7, @RW0+d16
+9	XCH R0, @RW1	XCH R0, @RW1+d16	XCH R1, @RW1	XCH R1, @RW1+d16	XCH R2, @RW1	XCH R2, @RW1+d16	XCH R3, @RW1	XCH R3, @RW1+d16	XCH R4, @RW1	XCH R4, @RW1+d16	XCH R5, @RW1	XCH R5, @RW1+d16	XCH R6, @RW1	XCH R6, @RW1+d16	XCH R7, @RW1	XCH R7, @RW1+d16
+A	XCH R0, @RW2	XCH R0, @RW2+d16	XCH R1, @RW2	XCH R1, @RW2+d16	XCH R2, @RW2	XCH R2, @RW2+d16	XCH R3, @RW2	XCH R3, @RW2+d16	XCH R4, @RW2	XCH R4, @RW2+d16	XCH R5, @RW2	XCH R5, @RW2+d16	XCH R6, @RW2	XCH R6, @RW2+d16	XCH R7, @RW2	XCH R7, @RW2+d16
+B	XCH R0, @RW3	XCH R0, @RW3+d16	XCH R1, @RW3	XCH R1, @RW3+d16	XCH R2, @RW3	XCH R2, @RW3+d16	XCH R3, @RW3	XCH R3, @RW3+d16	XCH R4, @RW3	XCH R4, @RW3+d16	XCH R5, @RW3	XCH R5, @RW3+d16	XCH R6, @RW3	XCH R6, @RW3+d16	XCH R7, @RW3	XCH R7, @RW3+d16
+C	XCH R0, @RW0+	XCH R0, @RW0+RW7	XCH R1, @RW0+	XCH R1, @RW0+RW7	XCH R2, @RW0+	XCH R2, @RW0+RW7	XCH R3, @RW0+	XCH R3, @RW0+RW7	XCH R4, @RW0+	XCH R4, @RW0+RW7	XCH R5, @RW0+	XCH R5, @RW0+RW7	XCH R6, @RW0+	XCH R6, @RW0+RW7	XCH R7, @RW0+	XCH R7, @RW0+RW7
+D	XCH R0, @RW1+	XCH R0, @RW1+RW7	XCH R1, @RW1+	XCH R1, @RW1+RW7	XCH R2, @RW1+	XCH R2, @RW1+RW7	XCH R3, @RW1+	XCH R3, @RW1+RW7	XCH R4, @RW1+	XCH R4, @RW1+RW7	XCH R5, @RW1+	XCH R5, @RW1+RW7	XCH R6, @RW1+	XCH R6, @RW1+RW7	XCH R7, @RW1+	XCH R7, @RW1+RW7
+E	XCH R0, @RW2+	XCH R0, @PC+d16	XCH R1, @RW2+	XCH R1, @PC+d16	XCH R2, @RW2+	XCH R2, @PC+d16	XCH R3, @RW2+	XCH R3, @PC+d16	XCH R4, @RW2+	XCH R4, @PC+d16	XCH R5, @RW2+	XCH R5, @PC+d16	XCH R6, @RW2+	XCH R6, @PC+d16	XCH R7, @RW2+	XCH R7, @PC+d16
+F	XCH R0, @RW3+	XCH R0, addr16	XCH R1, @RW3+	XCH R1, addr16	XCH R2, @RW3+	XCH R2, addr16	XCH R3, @RW3+	XCH R3, addr16	XCH R4, @RW3+	XCH R4, addr16	XCH R5, @RW3+	XCH R5, addr16	XCH R6, @RW3+	XCH R6, addr16	XCH R7, @RW3+	XCH R7, addr16

Table B.9-21 XCHW RWi,ea instruction map (1st byte = 7F$_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	XCHW 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, 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 R0, @RW0+	XCHW R0, @RW0+RW7	XCHW R1, @RW0+	XCHW R1, @RW0+RW7	XCHW R2, @RW0+	XCHW R2, @RW0+RW7	XCHW R3, @RW0+	XCHW R3, @RW0+RW7	XCHW R4, @RW0+	XCHW R4, @RW0+RW7	XCHW R5, @RW0+	XCHW R5, @RW0+RW7	XCHW R6, @RW0+	XCHW R6, @RW0+RW7	XCHW R7, @RW0+	XCHW R7, @RW0+RW7
+D	XCHW R0, @RW1+	XCHW R0, @RW1+RW7	XCHW R1, @RW1+	XCHW R1, @RW1+RW7	XCHW R2, @RW1+	XCHW R2, @RW1+RW7	XCHW R3, @RW1+	XCHW R3, @RW1+RW7	XCHW R4, @RW1+	XCHW R4, @RW1+RW7	XCHW R5, @RW1+	XCHW R5, @RW1+RW7	XCHW R6, @RW1+	XCHW R6, @RW1+RW7	XCHW R7, @RW1+	XCHW R7, @RW1+RW7
+E	XCHW R0, @RW2+	XCHW R0, @PC +d16	XCHW R1, @RW2+	XCHW R1, @PC+d16	XCHW R2, @RW2+	XCHW R2, @PC+d16	XCHW R3, @RW2+	XCHW R3, @PC+d16	XCHW R4, @RW2+	XCHW R4, @PC+d16	XCHW R5, @RW2+	XCHW R5, @PC+d16	XCHW R6, @RW2+	XCHW R6, @PC+d16	XCHW R7, @RW2+	XCHW R7, @PC+d16
+F	XCHW R0, @RW3+	XCHW R0, addr16	XCHW R1, @RW3+	XCHW R1, addr16	XCHW R2, @RW3+	XCHW R2, addr16	XCHW R3, @RW3+	XCHW R3, addr16	XCHW R4, @RW3+	XCHW R4, addr16	XCHW R5, @RW3+	XCHW R5, addr16	XCHW R6, @RW3+	XCHW R6, addr16	XCHW R7, @RW3+	XCHW R7, addr16

APPENDIX C 512K-BIT FLASH MEMORY

The 512K-bit flash memory features the following:

- **Sector configuration of 64K words x 8 bits/32K words x 16 bits (16K + 512 x 2 + 7K + 8K + 32K)**
- **Automatic programming algorithm (Embedded Algorithm: Same as MBM29F400TA)**
- **Erase fault/erase restart function**
- **Detecting the completion of write/erase by data polling or using toggle bits**
- **Detecting the completion of write/erase using the RY and BY pins**
- **Detecting the completion of write/erase using a CPU interrupt**
- **Compatible with JEDEC standard commands**
- **Erasable in units of sectors (free combination of sectors)**
- **A minimum of 100 write/erase operations**

Embedded Algorithm is a trademark of Advanced Micro Device Corporation.

■ Sector Configuration

Figure C-1 shows the sector configuration of the 512K-bit flash memory. When the memory is accessed from the CPU, SA0 to SA5 are seen in the FF bank.

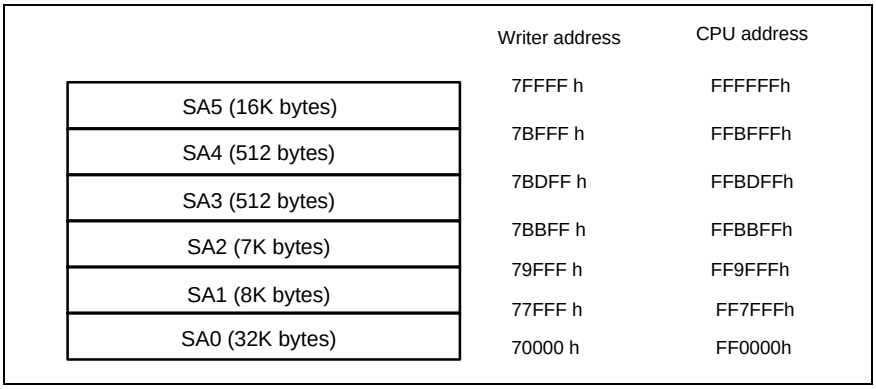


Figure C-1 Sector configuration of 512K-bit flash memory

■ Flash Memory Mode

When the mode pins are set to “111” for resetting, the CPU is shut down. When this occurs, the function of the flash memory interface circuit connects the signals from ports 0, 1, 2, 3, and 4 directly to the control signals of flash memory and enables flash memory to be directly controlled from external pins. This mode gives an image where flash memory appears as a single unit on the external pins. The mode is mainly used when data is written or erased using the flash memory writer. In this mode, every operation of the automatic algorithm for the 512K-bit flash memory can be used.

■ Normal Mode

Normal mode is specified when the mode pins are set for other than flash memory mode. The settings for use prohibition are excluded, though. The 1M-bit flash memory is allocated in the FE and FF banks on the CPU memory map, so the function of the flash memory interface circuit enables read access or program access from the CPU in the same way as with normal mask ROM. In this mode, the 512K-bit flash memory can also be written or erased by an instruction from the CPU via the flash memory interface circuit. This function enables rewriting in the installed state under control of the internal CPU. In this mode, normal writing and erasing can be performed but selector operation such as for enable sector protection cannot be performed.

■ Control Signals of 512K-Bit Flash Memory

Table C.1 lists the flash memory control signals used in normal mode and flash memory mode.

● Flash memory mode

In this mode, external data bus width is limited to 8 bits and therefore only byte access is enabled. D15 to D08 are not used. Be sure to set the BYTEX pin to “0”.

● Normal mode

Signals other than the reset signal are under control of the flash memory interface circuit. The data bus of flash memory is connected to the internal bus via the flash memory interface circuit. Selecting between byte and word access makes no sense. Therefore, the BYTEX signal of flash memory is internally fixed to 1 in this mode.

Table C-1 Correspondence between external control pins and flash memory control signals

External control pin	Flash memory mode pin	Normal mode
RSTX	RSTX	RSTX
P36	BYTEX	Internally fixed to H
P34	Vacant for IM (AQ17)	
P33	WEX	Internal and interface control signals
P32	OEX	Internal and interface control signals
P31	CEX	Internal and interface control signals
P30	AQ16	Internal address bus (A16)
P63, P46~ P40	AQ15 ~ AQ8	Internal address bus (A15 to A8)
P27~ P20	AQ7 ~ AQ0	Internal address bus (A7 to A0)
P07~ P00	DQ7 ~ DQ0	Internal data bus (D7 to D0)
MD2 ~MD0	VID pin (5V±5%)	MD2 ~MD0

■ Control Status Register (FMCS)

The control status register (FMCS) is situated in the flash memory interface circuit and used for writing or erasing flash memory when the CPU runs in normal mode. It cannot be used in flash memory mode.

● Register structure: Control status register (FMCS)

	7	6	5	4	3	2	1	0
Address:00000AEh	INTE	RDYI	WE	RDY	Reserved	-	-	LPM
Read/Write	(R/W)	(R/W)	(R/W)	(R/W)	(-)	(-)	(-)	(R/W)
Initial value	(0)	(0)	(0)	(0)	(0)	(-)	(-)	(0)

● Bit description

[Bit 7]: Interrupt Enable (INTE)

This bit is used to cause an interrupt to the CPU when the writing or erasing of flash memory is finished. A CPU interrupt is caused when the INTE and DYINT bits are both 1. It is not caused when the INTE bit is “0”.

[Bit 6]: Ready Interrupt (RDYINT)

This bit is set to “1” after the writing or erasing of flash memory is finished. The writing or erasing of flash memory is disabled while this bit is 0 even after the writing or erasing of flash memory is finished. Flash memory can be written or erased only after the bit is set to “1”. Writing “0” in the bit clears the bit to “0”, but writing “1” in the bit is ignored. The bit is set to “1” at the leading edge of the RY/BYX signal from flash memory. Read Modify Write (RMW) always reads “1” from this bit.

[Bit 5]: Write Enable (WE)

This bit is used to enable writing to the flash memory area. When the bit is “1”, a write to the FF or FE bank becomes a write to the flash memory area. The bit is used to activate a flash memory write or erase command. A write signal is generated when the bit is “1” and MD2 to MD0 are all “1”. No write signal is generated when the bit is “0” or MD2 to MD0 are not all “1”.

[Bit 4] : Ready (RDY)

RDY is a flash memory write/erase enable bit. While the bit is “0”, flash memory cannot be written or erased. Even in this situation, a read or reset command or a suspend command such as to temporarily stop a sector deletion can be accepted. Refer to the MB29F400TA specifications for more information.

[Bit 3]: Reserved bit

This bit is reserved for testing. Keep it as “0” for normal operation.

[Bits 2 and 1]: Unused bit

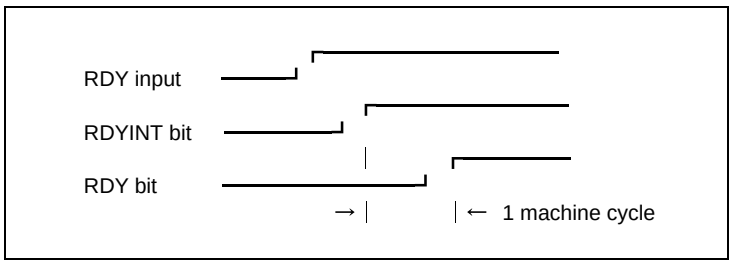
Leave it as “0” for normal operation.

[Bit 0]: Low Power Mode (LPM)

When this bit is set to “1”, the "L" period of the CEX signal supplied to flash memory during access to flash memory is minimized to suppress the power consumption of flash memory. However, the access time is greatly increased compared with that when LPM is “0”, and memory access is disabled when the CPU runs in high-speed mode. When LPM mode is used, therefore, run the CPU at frequencies of 4 MHz or less.

<Check>

The RDYINT and RDY bits do not change simultaneously. Create a program so that it uses one of them for a decision.



■ Read/Write Access Method

While flash memory mode requires external-pin control for read or write access to the 1M-bit flash memory, normal mode does not require it because access timing is controlled by the flash memory interface circuit. The write access explained below is a write access to the command register to activate the automatic algorithm and means a write to flash memory.

■ Read/Write Access in Flash Memory Mode

Table C-2 lists the pin settings for read, write, and other operations in flash memory mode. Control of these pin settings is not a problem when connecting to the flash memory writer but must satisfy the timing specifications of the individual pins in other cases. See the separate section for the timing specifications. Since data bus width is limited to eight bits in flash memory mode, fix the BYTEX pin to “0”.

Table C-2 Pin settings for read/write access in flash memory mode

Operation	CEX	OEX	WEX	AQ0~16	DQ0~7	RSTX
Read	L	L	H	Read address	DOUT	H
Write	L	H	L	Write address	DIN	H
Output disable	L	H	H	×	High-Z	H
Standby	H	×	×	×	High-Z	H
Hardware reset	×	×	×	×	High-Z	L

- **Read/write access in normal mode**

  For a read/write access to flash memory in normal mode, the timing of the internal bus access from the CPU is converted, in the flash memory interface circuit, into flash memory access timing. In this mode, one read/write cycle completes in two internal machine clock cycles.

  For data access with LPM set to “0” in this mode, CEX goes "L" for every cycle. However, for data access with LPM set to “1”, CEX goes "L" after the data decision of the data bus (D15 to D00). Thus, the access timing is different. (When LPM is set to 1, the access time increases and the CPU speed must be decelerated.)

  Internal address bits 15 to 0 corresponds to AQ15 to 0 in flash memory mode on a one-to-one basis, and internal data bits 15 to 0 correspond to DQ15 to 0 on a one-to-one basis. To perform a write access to flash memory, the WE bit of the control status register must be set to “1” in advance. The WE bit prevents the automatic algorithm from malfunctioning because of an invalid write to flash memory caused by noise. Therefore, when no write access is anticipated, the WE bit should be reset to “0”.

  When LPM is “0”, no restrictions apply to read/write access to flash memory, the same as for access to other types of memory, as long as the access time is within the range of the CPU operation guarantee frequencies. When LPM is “1”, the CEX timing controls read/write access and restricts the CPU operation frequencies. Therefore, to perform a read or write access with LPM set to “1”, set the CPU operation frequency to 4 MHz or less.

■ Ready or Busy Signal

The 1M-bit flash memory uses the aforementioned hardware sequence flags, i.e., data polling flag and toggle bit flag, to indicate whether the internal automatic algorithm is being executed or is finished. In addition, this memory also has hardware signals, i.e., Ready and Busy signals, available for the same purpose. The Ready and Busy signals operate differently, depending on whether the CPU runs in flash memory mode or normal mode.

- **Flash memory mode**

  For a Ready or Busy signal, the RY or BYX signal from flash memory is asynchronously supplied to the RY or BYX pin for the open drain output to the outside of the chip. When a pull-up resistor is connected to VCC, several RY or BYX pins can be connected in parallel. When the output of the RY or BYX pin is “0”, flash memory is busy writing or erasing. In this state, no write or erase command can be accepted. When the RY or BYX pin is set to “1” by the connection of an external pull-up resistor, flash memory is ready to accept a read, write, or erase command. In erase temporary stop mode, the RY or BYX output is set to “1” by the connection of an external pull-up resistor.

- **Normal mode**

  A Ready or Busy signal is supplied as an interrupt request signal to the CPU via the flash memory interface circuit. An interrupt request is sent to the CPU when the state of flash memory changes from busy to ready while the INTE bit of the control status register (FMCS) is “1”. No interrupt request is sent to the CPU even if the state of flash memory changes as above if the INTE bit of the FMCS register is “0”. Clearing the RDYINT bit of the FMCS register to “0” after an interrupt request is issued to the CPU cancels the interrupt request to the CPU. This action also resets the INTE bit of the FMCS register to “0”.

APPENDIX D EXAMPLE OF F²MC-16LX MB90F562 CONNECTION FOR SERIAL WRITING

MB90F562 supports serial on-board writing (Fujitsu standard) to flash ROM. This appendix describes the specifications for serial on-board writing.

■ Basic Configuration

The AF200 flash microcomputer programmer of Yokogawa Digital Computer Co., Ltd. is used for Fujitsu standard serial on-board writing.

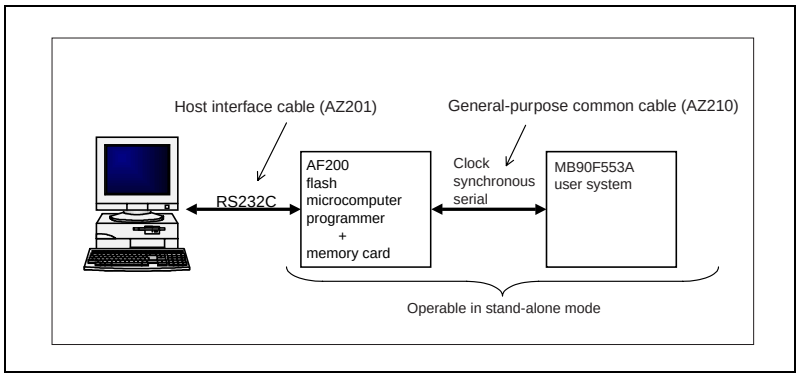


Figure D-1 Standard configuration for Fujitsu standard serial on-board writing

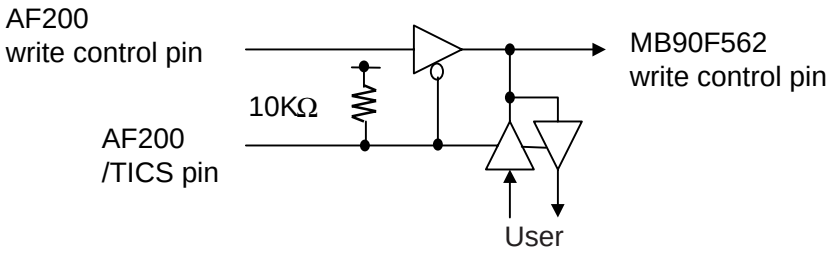
<Check>

Contact Yokogawa Digital Computer Co., Ltd. for the functionality and operation of the AF200 flash microcomputer programmer and information on the general-purpose common cable (AZ210) and connectors.

■ Pins used for Fujitsu standard serial on-board writing

Pin	Function	Description
MD2,MD1, MD0	Mode pin	Used to enable write mode for the flash microcomputer programmer.
P00	Write program start pin	–
RSTX	Reset pin	–
SIN	Serial data input pin	UART is used in clock synchronous mode.
SOT	Serial data output pin	
SCK	Serial clock input pin	
VCC	Power supply pin	If write voltage (5V±10%) is supplied from the user system, this pin need not be connected to the flash microcomputer programmer. If the pin is connected to the flash microcomputer programmer, do not connect it to the power of the user system.
VSS	Ground pin	Used also as the ground pin for the flash microcomputer programmer.

Note: When the P00, SIN0, SOT0, and SCK0 pins are also used by the user system, the control circuit shown below is required. (The /TICS signal of the flash microcomputer programmer can separate the user circuit during serial writing. See the connection example shown later.)



■ Connection example for serial writing

Figure D-2 is a connection example for serial writing.

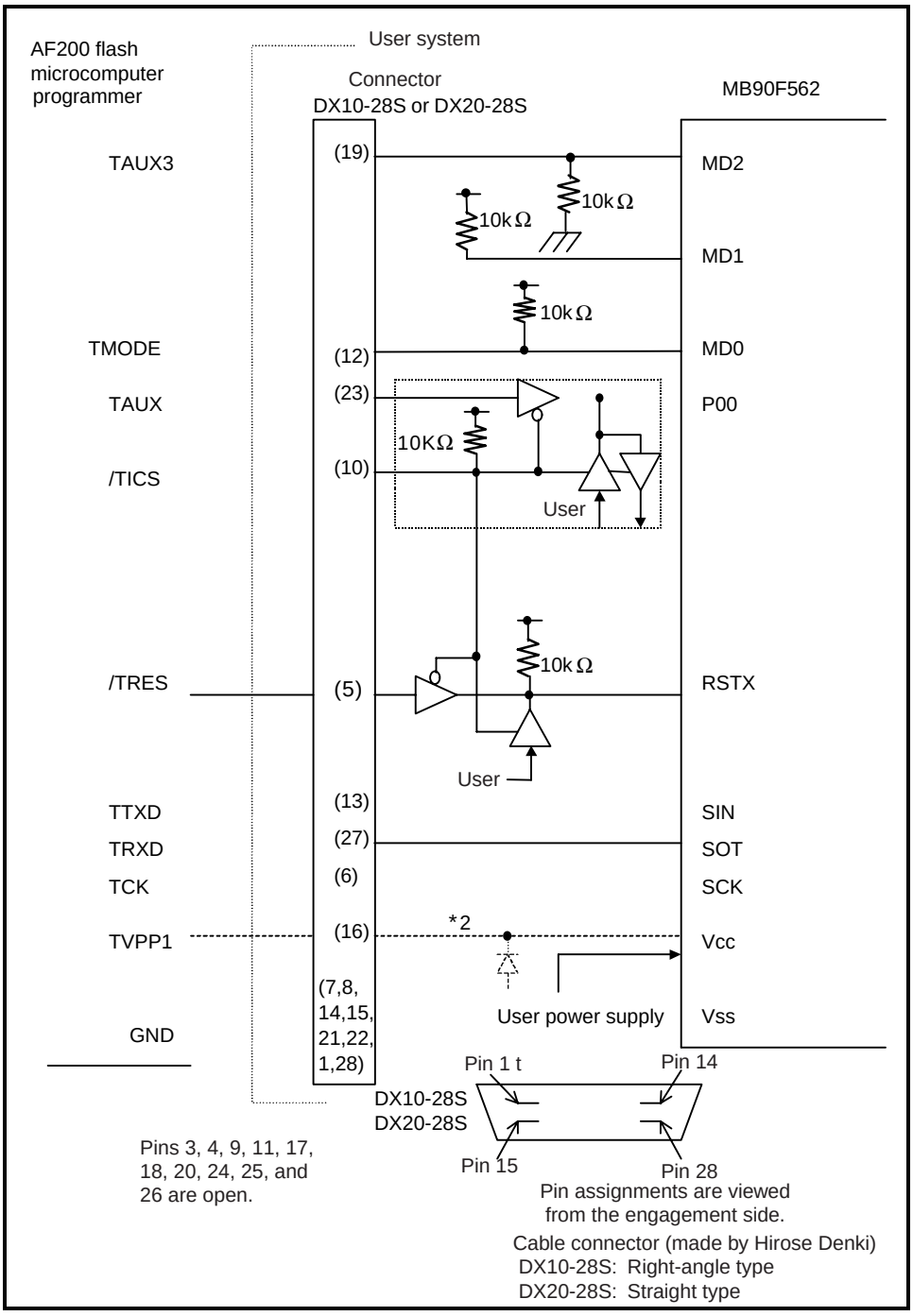
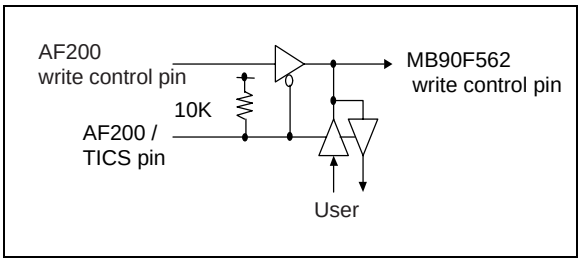


Figure D.2 Connection example for MB90F562 serial writing

Note:1. When the SIN, SOT, and SCK pins are also used by the user system, the control circuit shown on the right is required, which is the same as for P00. (The /TICS signal of the flash microcomputer programmer can separate the user circuit during serial writing.)



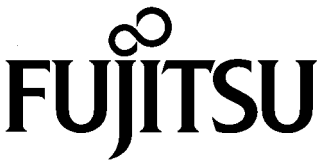
Note:2. If the power for writing is supplied by the AF200, do not connect the power pin to the power supply of the user system.

Note:3. Turn off the user system power before connecting to the AF200.

■ System configuration of AF200 flash microcomputer programmer (Yokogawa Digital Computer Co., Ltd.)

Type	Function
AF200	Advanced on-board flash microcomputer programmer
AD200/ ACP	AC adapter (center +) (Option)
AZ201	Host interface cable (PCAT)
AZ210	General-purpose common probe (Type A)
FF001	Fujitsu F²MC-16LX control module for flash microcomputer
FF001/P2	2MB PC Card (Option)
EF001/P4	4MB PC Card (Option)

For more information, contact the Sales Dept., Equipment Business Center, Yokogawa Digital Computer Co., Ltd. (Telephone: 0423-33-6224)


FUJITSU

FUJITSU MIKROELEKTRONIK GmbH FUJITSU MIKROELEKTRONIK GmbH
Am Siebenstein 6-10, 63303 Dreieich-Buchschlag, Germany

Corrections of MB90560 Hardware Manual (CM44-10107-1E)

1. A/D CONVERTER

The total error for the 3V version (MB90560L series) is max. +/-8LSB.
The total error for the 5V version (MB90560 series) is max. +/-5LSB

ADCS1 Register

STS1-0 bits = 0,0 --> Activation by Software.
STS1-0 bits = 0,1 --> Activation by Software, and Zero detection of Free Running Timer
STS1-0 bits = 1,0 --> Activation by Software, and Reload Timer
STS1-0 bits = 1,1 --> Activation by Software, Zero detection of Free Running Timer, and Reload Timer.

1. PPG

Chapter 12.3.4.3 PPG clock control Register (PCS01/23/45)
The operating clock settings fc/32 and fc/64 are not possible. If
these settings are used the PPG output will stay on low level.

When PPG is set to 16 bit mode with "GATE" function, the output can't produce desired waveform.

Input capture register

Chapter 12.3.3 (page 304 in HW-Manual) capture control register (upper) ICS23
The address for upper register 0x69 is incorrect, correct address is 0x6A.

Timer Control Status Register

Chapter 12.3.1.3 (page 294 in HW-Manuel) Upper Timer control status register TCCS (bit 15-8).
Description of bit8 ICRE Compare interrupt request enable bit is not correct.
Enable interrupt set '1',
disable interrupt set '0'.

UART - synchronous mode

Chapter 13.6.1 Baud Rates: synchronous Transfer Division Ratios.
Page 379 in HW-Manual Table 13.6-2
Division ratio for CLK synchronization: values in Table are incorrect
correct baud rates (div=4):
CS2,CS1,CS0,baudrates
0, 0, 0, 4M
0, 0, 1, 2M
0, 1, 0, 1M
0, 1, 1, 500K
1, 0, 0, 250K
1, 0, 1, 125K
Max. synchronous baud rate with div=1: 8Mbaud

Appendix C, 512K-Bit Flash memory

The Sector configuration of the Flash memory is 32K/8K/8K/16K

Low Power Mode, Chapter 5.3

Bit 3 of the LPMCR is used to enter the Timer Mode. Write 0 to this
bit to enter the Timer mode. Clearing this bit will enter the timer mode,
regardless from the current mode of the CPU.

Time Base Timer Mode, chapter 5.5.2

Time Base Timer Mode is entered by clearing Bit3 of the LPMCR register.
Clearing this bit will enter the timer mode, regardless from the current mode of the CPU.

last updated : 16 September'99 (tka)