

F²MC-16LX FAMILY
16-BIT MICROCONTROLLER
MB90495 SERIES
HARDWARE MANUAL

1. The products described in this manual and the specifications thereof may be changed without prior notice. To obtain up-to-date information and/or specifications, contact your Fujitsu representative or Fujitsu authorized dealer.
2. Fujitsu will not be liable for infringement of copyright, industrial property right, or other rights of a third party caused by the use of information or drawings described in this manual.
3. The contents of this manual may not be transferred or copied without the express permission of Fujitsu.
4. The products contained in this document are not intended for use with equipments which require extremely high reliability such as aerospace equipments, undersea repeaters, nuclear control systems or medical equipments for life support.
5. Some of the products described in this manual may be strategic materials (or special technology) as defined by the Foreign Exchange and Foreign Trade Control Law. In such cases, the products or portions thereof must not be exported without permission as defined under the Law.

Revision History

Date	Issue
12/Nov,1999	Revision 1.2 First general release
10/Dec, 1999	Release 1.3 Insert Watchdog timer (13.11.99) Insert Watchtimer, new Clock-Unit
1/Mar, 2000	Release 1.4 Typographical error correction

CHAPTER 1	OVERVIEW	11
1.1	Features	12
1.2	Product Lineup	15
1.3	Block Diagram	16
1.4	Pin Assignments	17
1.5	Package Dimensions	19
1.6	Pin Functions	21
1.7	I/O Circuit Types	23
1.8	Notes on Handling Devices	25
CHAPTER 2	CPU	27
2.1	CPU	28
2.2	Memory Space	29
2.3	Memory Maps	31
2.4	Addressing	33
2.5	Memory Location of Multibyte Data	37
2.6	Registers	39
2.7	Dedicated Registers	40
2.8	General-Purpose Registers	56
2.9	Prefix Codes	58
CHAPTER 3	RESETS	65
3.1	Resets	66
3.2	Reset Cause and Oscillation Stabilization Wait Intervals	68
3.3	External Reset Pin	70
3.4	Reset Operation	72
3.5	Reset Cause Bits	74
3.6	Status of Pins in a Reset	77
CHAPTER 4	CLOCKS	79
4.1	Clocks	80
4.2	Block Diagram of the Clock Generation Block	82
4.3	Clock Selection Register (CKSCR)	84
4.4	Clock Mode	87
4.5	Oscillation Stabilization Wait Interval	91
4.6	Connection of an Oscillator or an External Clock to the Microcontroller	92
CHAPTER 5	LOW POWER CONSUMPTION MODE	95
5.1	Overview of Low Power Consumption Mode	96
5.2	Block Diagram of the Low-Power Consumption Control Circuit	99
5.3	Low-Power Consumption Mode Control Register (LPMCR)	101
5.4	CPU Intermittent Operation Mode	104
5.5	Standby Mode	105
5.6	Sleep Mode	106
5.7	Timebase Timer Mode	108
5.8	Watch Mode	110
5.9	Stop Mode	112

5.10	Status Change Diagram	114
5.11	Statuses of Pins in Standby Mode and during Hold and Reset	116
5.12	Usage Notes on Low-Power Consumption Mode	117
CHAPTER 6 I/O PORTS		119
6.1	Overview of I/O Ports	120
6.2	Register and External Dual-Function Pin Assignment	121
6.3	Port 0	122
6.4	Port 1	126
6.5	Port 2	131
6.6	Port 3	136
6.7	Port 4	141
6.8	Port 5	146
6.9	Port 6	152
6.10	Sample I/O Port Program	157
CHAPTER 7 UART0		159
7.1	Overview of UART0	160
7.2	UART0 Registers	161
7.3	UART0 Operating Modes and Clock Selection	171
7.4	UART0 Flags and Interrupt Sources	176
7.5	Negative clock operation	179
7.6	UART0 Sample Applications and Precautionary Information	180
CHAPTER 8 UART1		183
8.1	Overview of UART1	184
8.2	Configuration of UART1	186
8.3	UART1 Pins	189
8.4	UART1 Registers	190
8.5	UART1 Interrupts	200
8.6	UART1 Baud Rates	204
8.7	Operation of UART1	212
8.8	Notes on Using UART1	224
8.9	Sample Program for UART1	225
CHAPTER 9 TIMEBASE TIMER		227
9.1	Overview of the Timebase Timer	228
9.2	Configuration of the Timebase Timer	230
9.3	Timebase Timer Control Register (TBTC)	232
9.4	Timebase Timer Interrupts	234
9.5	Operation of the Timebase Timer	235
9.6	Usage Notes on the Timebase Timer	237
9.7	Sample Program for the Timebase Timer	239
CHAPTER 10 WATCHDOG TIMER		241
10.1	Overview of the Watchdog Timer	242
10.2	Watchdog Timer Control Register (WDTC)	243

10.3	Configuration of the Watchdog Timer	245
10.4	Operation of the Watchdog Timer	247
10.5	Usage Notes on the Watchdog Timer	249
10.6	Sample Program for the Watchdog Timer	250
CHAPTER 11 WATCH TIMER		251
11.1	Overview of the Watch Timer	252
11.2	Watch Timer Control Register (WTC)	253
11.3	Configuration of the Watch Timer	255
11.4	Operation of the Watch Timer	256
CHAPTER 12 16-Bit I/O Timer		257
12.1	Function overview	258
12.2	Registers	259
12.3	Block diagram	260
12.4	Input capture	265
CHAPTER 13 16-BIT RELOAD TIMER		273
13.1	Overview of the 16-Bit Reload Timer	274
13.2	Configuration of the 16-Bit Reload Timer	277
13.3	16-Bit Reload Timer Pins	279
13.4	16-Bit Reload Timer Registers	280
13.5	16-Bit Reload Timer Interrupts	287
13.6	Operation of the 16-Bit Reload Timer	289
13.7	Usage Notes on the 16-Bit Reload Timer	297
13.8	Sample Programs for the 16-Bit Reload Timer	298
CHAPTER 14 8/16-Bit PPG		301
14.1	Overview	302
14.2	Registers	302
14.3	Block diagram	303
14.4	Register details	305
14.5	Operations	310
CHAPTER 15 8/10-BIT A/D CONVERTER		315
15.1	Overview of the 8/10-Bit A/D Converter	316
15.2	Configuration of the 8/10-Bit A/D Converter	318
15.3	8/10-Bit A/D Converter Pins	320
15.4	8/10-Bit A/D Converter Registers	322
15.5	8/10-Bit A/D Converter Interrupts	330
15.6	Operation of the 8/10-Bit A/D Converter	331
15.7	Usage Notes on the 8/10-Bit A/D Converter	337
15.8	Sample Program 1 for the 8/10-Bit A/D Converter (Single Conversion Mode Using EI2OS) ..	338
15.9	Sample Program 2 for the 8/10-Bit A/D Converter (Continuous Conversion Mode Using EI2OS)	340
15.10	Sample Program 3 for the 8/10-Bit A/D Converter (Stop Conversion Mode Using EI2OS)	343

CHAPTER 16	DTP/EXTERNAL INTERRUPT CIRCUIT	347
16.1	Overview of the DTP/External Interrupt Circuit	348
16.2	Configuration of the DTP/External Interrupt Circuit	350
16.3	DTP/External Interrupt Circuit Pins	352
16.4	DTP/External Interrupt Circuit Registers	353
16.5	Operation of the DTP/External Interrupt Circuit	359
16.6	Usage Notes on the DTP/External Interrupt Circuit	364
16.7	Sample Programs for the DTP/External Interrupt Circuit	366
CHAPTER 17	CAN Controller	371
17.1	List of Control Registers	372
17.2	Message Buffers	374
17.3	Block Diagram	377
17.4	Overall Control Register	378
17.5	Message Buffer Control Registers	386
17.6	Message Buffers	394
17.7	Transmission	399
17.8	Reception	401
17.9	Usage Procedure	404
CHAPTER 18	Delayed Interrupt Generator Module	411
18.1	Overview of the Delayed Interrupt Generator Module	412
18.2	Operation of the Delayed Interrupt Generator Module	413
CHAPTER 19	ROM Correction	415
19.1	Registers	416
19.2	Block Diagram	416
19.3	Register Details	417
19.4	Operations	418
19.5	Application Example	418
CHAPTER 20	ROM Mirroring Module	423
20.1	Register	424
20.2	Block Diagram	424
20.3	Register Details	424
CHAPTER 21	INTERRUPTS	427
21.1	Interrupts	428
21.2	Interrupt Causes and Interrupt Vectors	430
21.3	Interrupt Control Registers and Peripheral Functions	433
21.4	Hardware Interrupts	439
21.5	Software Interrupts	450
21.6	Interrupt of Extended Intelligent I/O Service (EI2OS)	452
21.7	Operation of the extended intelligent I/O service (EI2OS)	458
21.8	Exception Processing Interrupt	462
21.9	Stack Operations for Interrupt Processing	463
21.10	Sample Programs for Interrupt Processing	465

CHAPTER 22	SETTING A MODE	469
22.1	Setting a Mode	470
22.2	Mode Pins (MD2 to MD0)	471
22.3	Mode Data	472
22.4	External Memory Access (Bus Pin Control Circuits)	474
CHAPTER 23	FLASH MEMORY	483
23.1	Outline of Flash Memory	484
23.2	Block Diagrams of Flash Memory	485
23.3	Write/Erase Modes	487
23.4	Control Status Register (FMCS)	489
23.5	Read/Write Access	491
23.6	Automatic Write/Erase Algorithm	493
23.7	Sector Protect Operation	506
23.8	Connection to Flash Memory Writer	509
23.9	Notes for Use of Flash Memory	510
23.10	Timing Diagrams in Flash Memory Mode	511
23.11	AC Characteristics in Flash Memory Mode	518
APPENDIX A	I/O MAP	521
APPENDIX B	Interrupt Table	525

CHAPTER 1 OVERVIEW

This chapter describes the features of the MB90495 series.

- 1.1 Features
- 1.2 Product Lineup
- 1.3 Block Diagram
- 1.4 Pin Assignments
- 1.5 Package Dimensions
- 1.6 Pin Functions
- 1.7 I/O Circuit Types
- 1.8 Notes on Handling Devices

1.1 Features

The MB90495 series of general-purpose 16-bit microcontrollers with Full CAN interface and FLASH ROM is especially designed for automotive and Industrial applications. The main feature of the series is the on-chip CAN interface, which conforms to V2.0 Part A and Part B, whilst supporting a very flexible message buffer scheme, incorporating 8 message buffers, and so offering more functionality than a normal CAN interface.

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 MB90495 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)
 - Sub clock mode. (32KHz)
- Maximum memory address space: 16M bytes
 - Internal 24-bit addressing
 - External bus interface
- 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
 - Sub clock mode, sub sleep mode, sub watch mode and sub stop mode (sub clock is operating clock, PLL clock is stopped)
- Packages
 - QFP-64 (FPT-64P-M09: 0.65 mm pin pitch, FPT-64P-M06: 1.00 mm pin pitch)
- Process: CMOS technology

■ Internal peripheral functions (resources)

- I/O ports: Up to 49 ports
- Watch timer: 1 channel
- 18-bit time-base timer: 1 channel
- 16-bit reload timer: 2 channels
- 16-bit free running timer: 1 channel
- Watchdog timer: 1 channel
- 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 4 channels or 16 bits x 2 channels
 - Capable of changing the period or duty cycle of the output pulses as desired.
- CAN: 1ch
 - Conforms to CAN Specification Version 2.0 Part A and B
 - 8 transmitting/receiving message buffers
 - Bit-rate programmable from 10 Kbits/s to 1Mbits/s (at 16 MHz input clock)
- 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 MB90495 series. The specifications excluding the ROM and RAM capacities are common for the series.

■ Product Lineup

Table 1.2-1 Product lineup of the MB90495 series

Model	MB90V495	MB90F497	MB90497
Type	Evaluation device	Flash Type ROM	Mass-production product (mask ROM)
ROM size	Not installed	64KByte	
RAM size	6KByte	2KByte	
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 (CM0S): 49		
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		
CAN	CAN Specification Version 2.0 Part A and B 8 transmitting/receiving message buffers		
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.		
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 modes	Sleep mode, stop mode and CPU intermittent mode sub clock mode, sub sleep mode, sub watch mode and sub stop mode		
Process	0.5 μm CMOS		
Package	PGA256	QFP-64 (0.65 or 1.00 mm pitch)	
Operating voltage	5V ± 10 %		

1.3 Block Diagram

Figure 1.3-1 shows the block diagram of the MB90495 series.7

■ Block Diagram

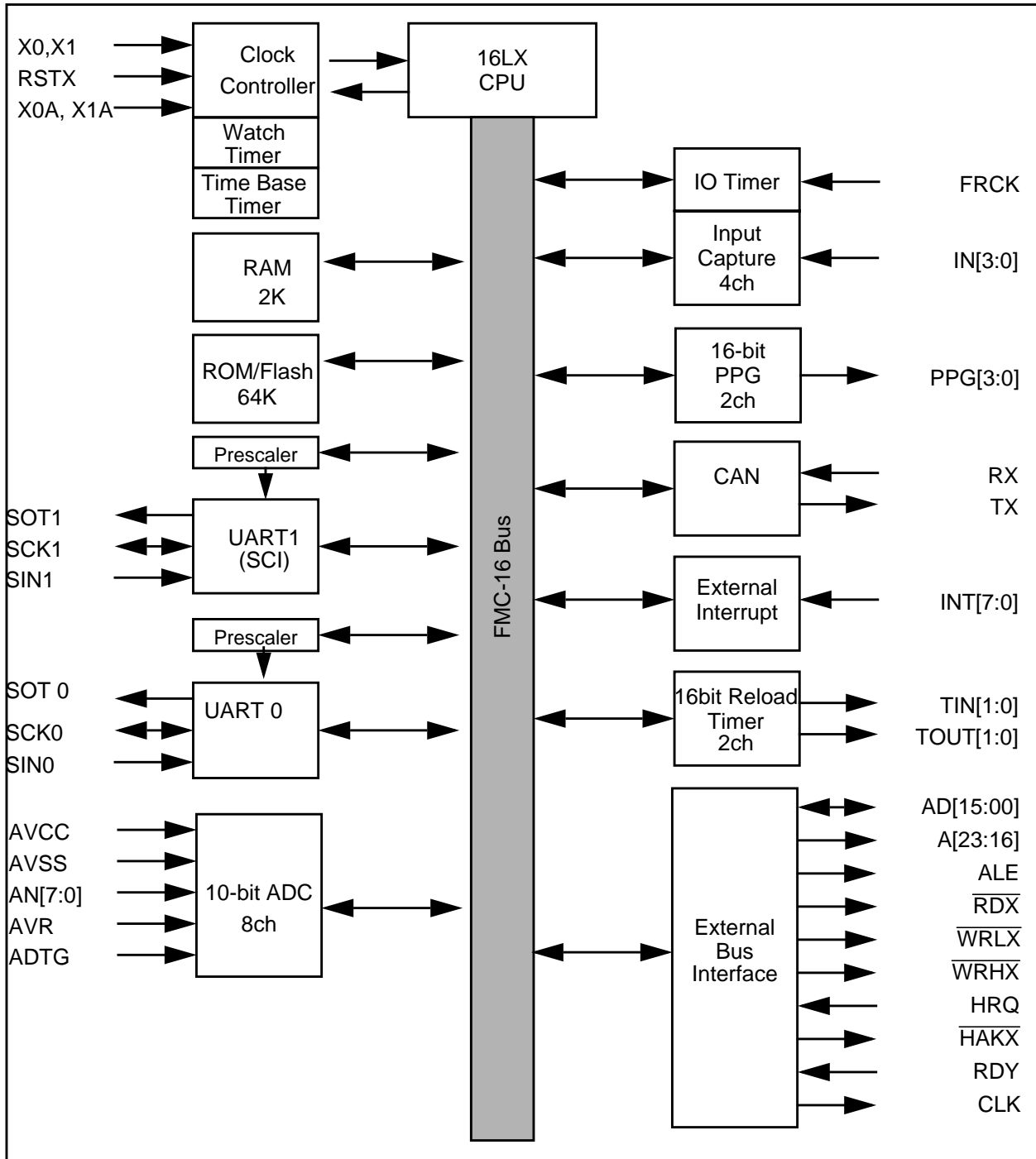


Figure 1.3-1 Block diagram

1.4 Pin Assignments

This section provides the pin assignments for the following three MB90495 series packages:

- FPT-64P-M06
- FPT-64P-M09

■ Pin Assignment of FPT-64P-M06

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

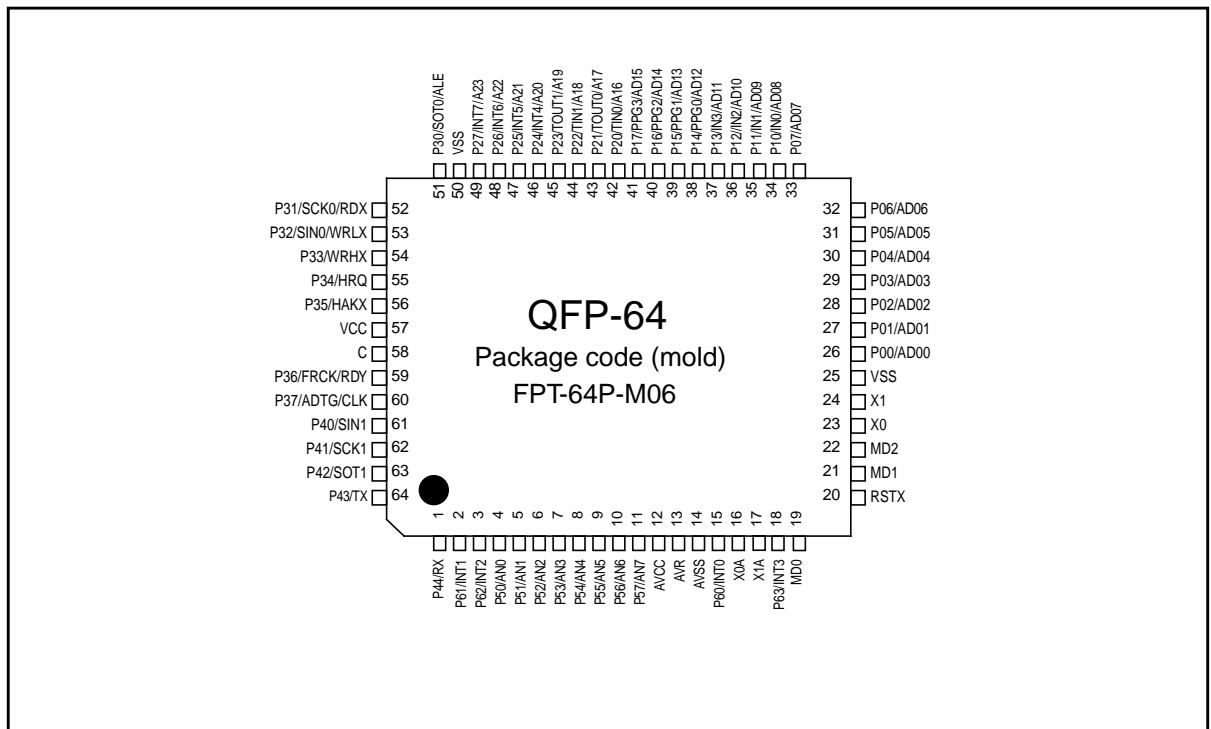


Figure 1.4-1 Pin assignment of FPT-64P-M06

■ Pin Assignment of FPT-64P-M09

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

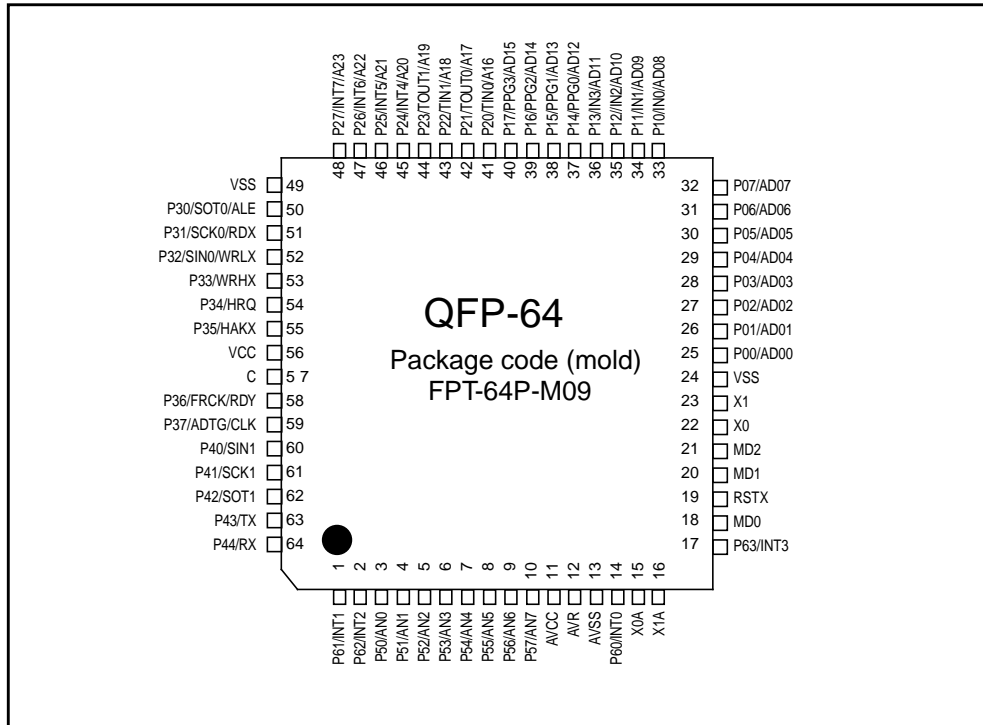


Figure 1.4-2 Pin assignment of FPT-64P-M09

■ FPT-64P-M09

Figure 1.5-2 shows the dimensions of the FPT-64P-M09 package.

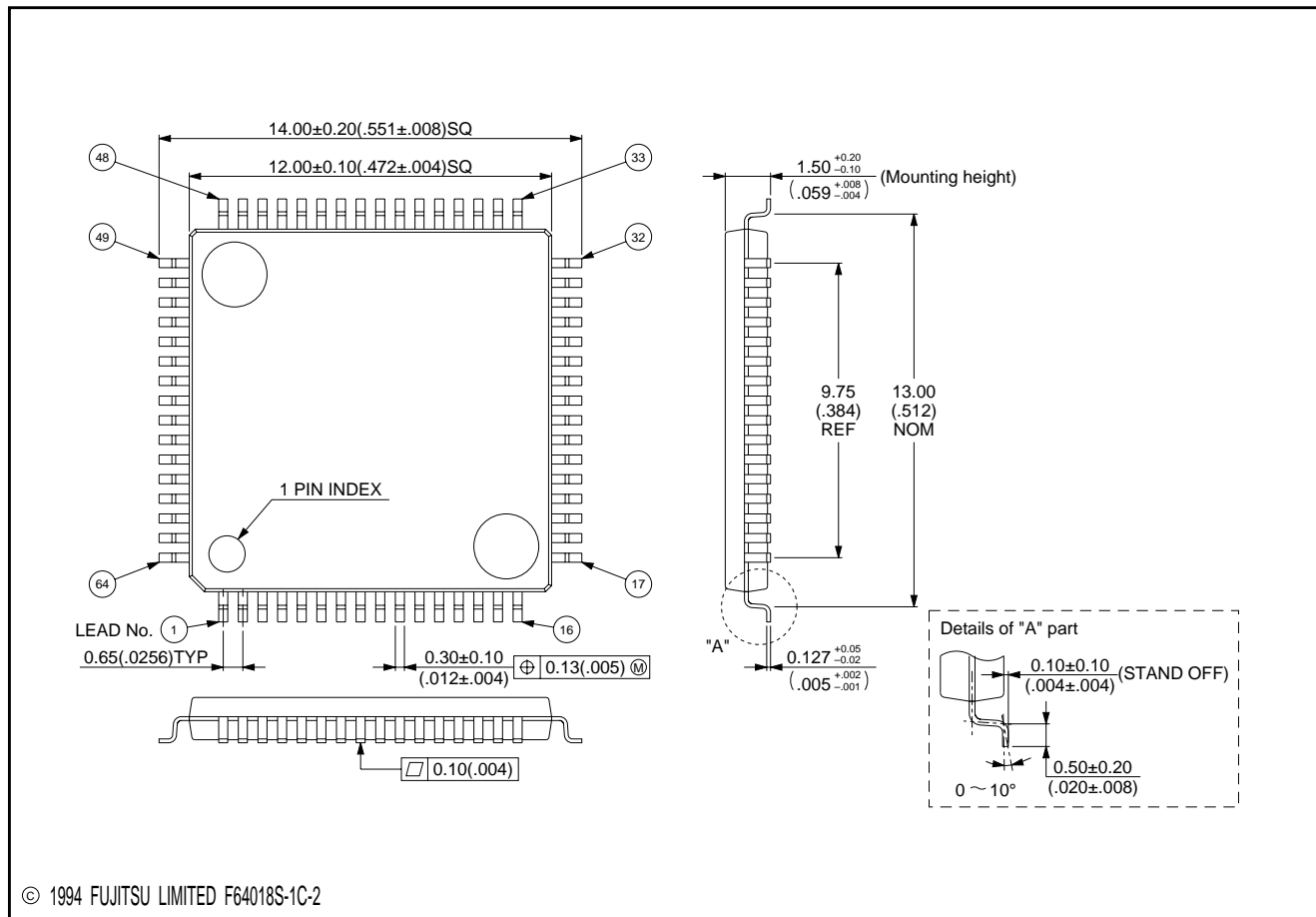


Figure 1.5-2 Dimensions of FPT-64P-M09 package

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	Active	Level	at RST	Priority	Function
M06	M09							
2	1	P61	D	H	CMOS/TTL	High-Z	Port	General purpose IO
		INT1						External Interrupt input 1
3	2	P62	D	H	CMOS/TTL	High-Z	Port	General purpose IO
		INT2						External interrupt 2
4 to 11	3 to 10	P50 to P57	E	H	CMOS	High-Z	Port	General purpose IO
		AN0 to AN7						Inputs for A/D Converter
12	11	AVCC						Dedicated power supply for A/D Converter
13	12	AVR						Reference Voltage input for A/D Converter
14	13	AVSS						Dedicated power ground for A/D Converter
15	14	P60	D	H	CMOS/TTL	High-Z	Port	General purpose IO
		INT0						External interrupt input 0
16	15	X0A	A					Low frequency oscillation input
17	16	X1A	A					Low frequency oscillation output
18	17	P63	D	H	CMOS/TTL	High-Z	Port	General purpose IO
		INT3						External interrupt 3
19	18	MD0	C	H	CMOS			Mode input
20	19	RSTX	B	L	CMOS			Reset input
21	20	MD1	C	H	CMOS			Mode input
22	21	MD2	F	H	CMOS			Mode input
23	22	X0	A					High frequency oscillation input
24	23	X1	A					High frequency oscillation output
25	24	VSS						Power ground
26 to 33	25 to 32	P00 to P07	D	H	CMOS/TTL	High-Z	Port	General purpose IO
		AD00 to AD07						Address Data Bus
34 to 37	33 to 36	P10 to P13	D	H	CMOS/TTL	High-Z	Port	General purpose IO
		IN0 to IN3						Inputs for Input Captures
38 to 41	37 to 40	AD08 to AD11	D	H	CMOS/TTL	High-Z	Port	Address Data Bus
		P14 to P17						General purpose IO
42	41	PPG0 to PPG3	D	H	CMOS/TTL	High-Z	Port	Outputs for Programmable Pulse Generators
		AD12 to AD15						Address Data Bus
43	42	P20	D	H	CMOS/TTL	High-Z	Port	General purpose IO
		TIN0						Input for 16-bit Reload Timer 0
44	43	A16	D	H	CMOS/TTL	High-Z	Port	Address Bus
		P21						General purpose IO
45	44	TOUT0	D	H	CMOS/TTL	High-Z	Port	Output for 16-bit Reload Timer 0
		A17						Address Bus
46	45	P22	D	H	CMOS/TTL	High-Z	Port	General purpose IO
		TIN1						Input for 16-bit Reload Timer 1
47	46	A18	D	H	CMOS/TTL	High-Z	Port	Address Bus
		P23						General purpose IO
48	47	TOUT1	D	H	CMOS/TTL	High-Z	Port	Output for 16-bit Reload Timer 1
		A19						Address Bus
49	48	P24 to P27	D	H	CMOS/TTL	High-Z	Port	General purpose IO
		INT4 to INT 7						Inputs for External Interrupt
50	49	A20 to A23						Address Bus
51	50	VSS						Power ground
52	51	P30	D	H	CMOS/TTL	High-Z	Port	General purpose IO
		SOT0						Output for UART 0
		ALE						Address Latch Enable output

Pin No.		Pin Name	Circuit Type	Active	Level	at RST	Priority	Function
M06	M09							
52	51	P31	D	H	CMOS/TTL	High-Z	Port	General purpose IO
		SCK0						Clock input/Output for UART 0
		RDX						Read Enable output
53	52	P32	D	H	CMOS/TTL	High-Z	Port	General purpose IO
		SIN0						Input for UART 0
		WRLX						Write Enable Low-byte output
54	53	P33	D	H	CMOS/TTL	High-Z	Port	General purpose IO
		WRHX						Write Enable High-byte output
55	54	P34	D	H	CMOS/TTL	High-Z	Port	General purpose IO
		HRQ						Halt Request input
56	55	P35	D	H	CMOS/TTL	High-Z	Port	General purpose IO
		HAKX						Halt Acknowledge output
57	56	VCC						Power supply
58	57	C						Pin for capacitor for the internal power supply.
59	58	P36	D	H	CMOS/TTL	High-Z	Port	General purpose IO
		FRCK						Input for IO Timer
		RDY						Ready input
60	59	P37	D	H	CMOS	High-Z	Port	General purpose IO
		ADTG						Trigger input for A/D Converter
		CLK						Clock output
61	60	P40	D	H	CMOS/TTL	High-Z	Port	General purpose IO
		SIN1						Input for UART 1
62	61	P41	D	H	CMOS/TTL	High-Z	Port	General purpose IO
		SCK1						Input/Output for UART 1
63	62	P42	D	H	CMOS/TTL	High-Z	Port	General purpose IO
		SOT1						Output for UART 1
64	63	P43	D	H	CMOS/TTL	High-Z	Port	General purpose IO
		Tx						CAN Transmit pin
1	64	P44	D	H	CMOS/TTL	High-Z	Port	General purpose IO
		Rx						CAN receive pin

1.7 I/O Circuit Types

Table 1.7-1 summarizes the I/O circuit types of the MB90495 series.

■ I/O Circuit Types

Circuit	Drawing	Comment
A		
B		Hysteresis input with pull-up Resistor: 50 Kohm approx.
C		Hysteresis input
D		
E		
F		Hysteresis in/out with pull-down Resistor: 50 Kohm approx.

Circuit	Drawing	Comment
G	<p>The diagram shows a MOSFET circuit. A square pulse source is connected to the gate of a MOSFET. The MOSFET's source is grounded, and its drain is connected to a load (represented by a rectangle) and a feedback network. The feedback network consists of two parallel branches: one with a resistor and an AND gate, and another with a resistor and an AND gate. The inputs to the top AND gate are the MOSFET gate and the drain. Its output is labeled 'HYS'. The inputs to the bottom AND gate are the MOSFET gate and the drain. Its output is labeled 'TTL'. A 'Standby Control Signal' is connected to the inputs of both AND gates. The MOSFET's gate is also connected to a feedback path from the drain through a resistor.</p>	

1.8 Notes on Handling Devices

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

- Strict observation of maximum rated voltage (latch-up prevention)
 - Stabilization of supply voltage
 - Power-on
 - Treatment of unused input pins
 - Treatment of A/D converter power pin
 - External clock
 - Power supply pin
 - Analog power-on sequence of A/D converter
-

■ Notes on Handling Devices

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

A latchup may occur on a CMOS IC if a voltage higher than V_{CC} or lower than V_{SS} 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 V_{CC} and V_{SS} . 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 (AV_{CC} , AVR) and analog input voltage do not exceed the digital supply voltage (V_{CC}).

- Stabilize the supply voltages.

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

- Power-on

To prevent a malfunction in the built-in voltage drop circuit, secure “50 μ S (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 connected to V_{CC} or V_{SS} through a resistor. (Pull-up or pull-down).

- Treatment of A/D converter power pin

When the A/D converter is not used, connect the pins as follows: $AV_{CC} = V_{CC}$, $AV_{SS} = AVR = V_{SS}$.

- Notes on external clock

When an external clock is used, the oscillation stabilization wait time is required at power-on reset or at cancellation of subclock mode or stop mode. As shown in Figure 1.8-1, when an external clock is used, connect only the X0 pin and leave the X1 pin open.

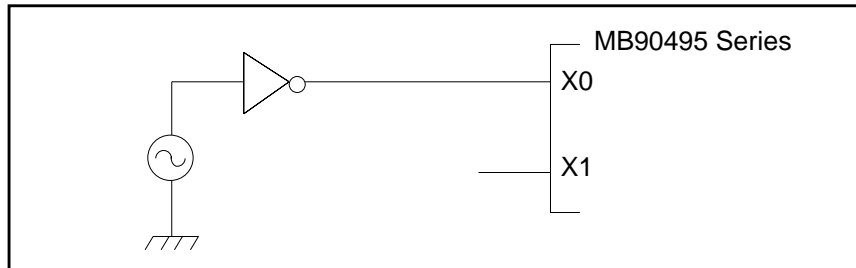


Figure 1.8-1 Sample application of external clock

- Power supply pins

When a device has two or more Vcc or Vss pins, the pins that should have equal potential are connected inside the device in order to prevent 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 or 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, AVR) 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 MB90495 series.

- 2.1 CPU
- 2.2 Memory Space
- 2.3 Memory Maps
- 2.4 Addressing
- 2.5 Memory Location of Multibyte Data
- 2.6 Registers
- 2.7 Dedicated Registers
- 2.8 General-Purpose Registers
- 2.9 Prefix Codes

2.1 CPU

The F²MC-16LX CPU core is a 16-bit CPU designed for use in applications, such as welfare and mobile equipment, which require high-speed real-time processing. The instruction set of the 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. The features of the F²MC-16LX CPU are shown below:

■ CPU

- Minimum instruction execution time: 62.5 ns (source oscillation at 4 MHz and clock multiplication by 4)
- Maximum memory address space: 16M bytes. Access in linear or bank mode
- Instruction set optimum for controller applications
 - Many data types (bit, byte, word, and long word)
 - As many as 23 addressing modes
 - Enhanced high-precision arithmetic operation by a 32-bit accumulator
 - Enhanced signed multiply/divide instructions and RETI instruction function
- 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) and multitasking
 - System stack pointer, instruction set symmetry, and barrel shift instructions
- Increased execution speed: 4-byte instruction queue

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.

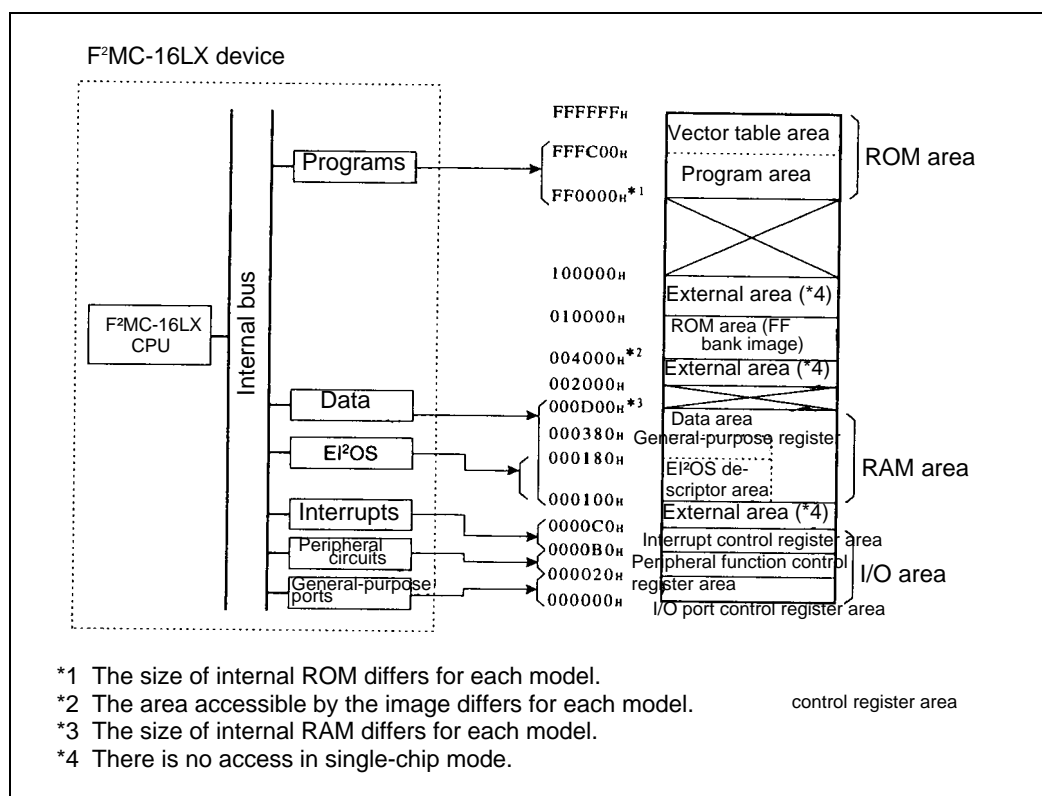


Figure 2.2-1 Sample relationship between the F²MC-16LX system and the memory map

■ ROM area

● Vector table area (address: FFFC00H to FFFFFFFH)

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

■ 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 model.

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

- Auxiliary registers used for 8-bit, 16-bit, and 32-bit arithmetic operations and transfer 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 MB90495 series model.

Memory maps

Figure 2.3-1 shows the memory maps for the MB90495 series

	MB90V495	MB90F497	MB90497
FFFFFFFH	ROM FF	Flash FF	ROM FF
FF0000H			
FEFFFFFFH	ROM FE	No access	No access
FE0000H			
FDFFFFFFH	ROM FD		
FD0000H			
FCFFFFFFH	ROM FC		
FC0000H			
	External Bus	External Bus	External bus
010000H			
00FFFFFFH	FF mirror	FF mirror	FF mirror
004000H			
003FFFFH	Ext. IO	Ext. IO	Ext. IO
003800H			
		External Bus	External Bus
0018FFH			
0010FFH	RAM	RAM mirror	RAM mirror
000900H			
0008FFH		RAM	RAM
000100H			
0000BFH	IO	IO	IO
000000H			

Figure 2.3-1 Memory map

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

Notes: 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 FFFFFFFH is seen as an image at 004000H to 00FFFFFFH, the ROM data table should be stored in the area from FF4000H to FFFFFFFH.

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 bank register for the required purpose, 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 divided into and managed as 256 64-kilobyte banks.

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

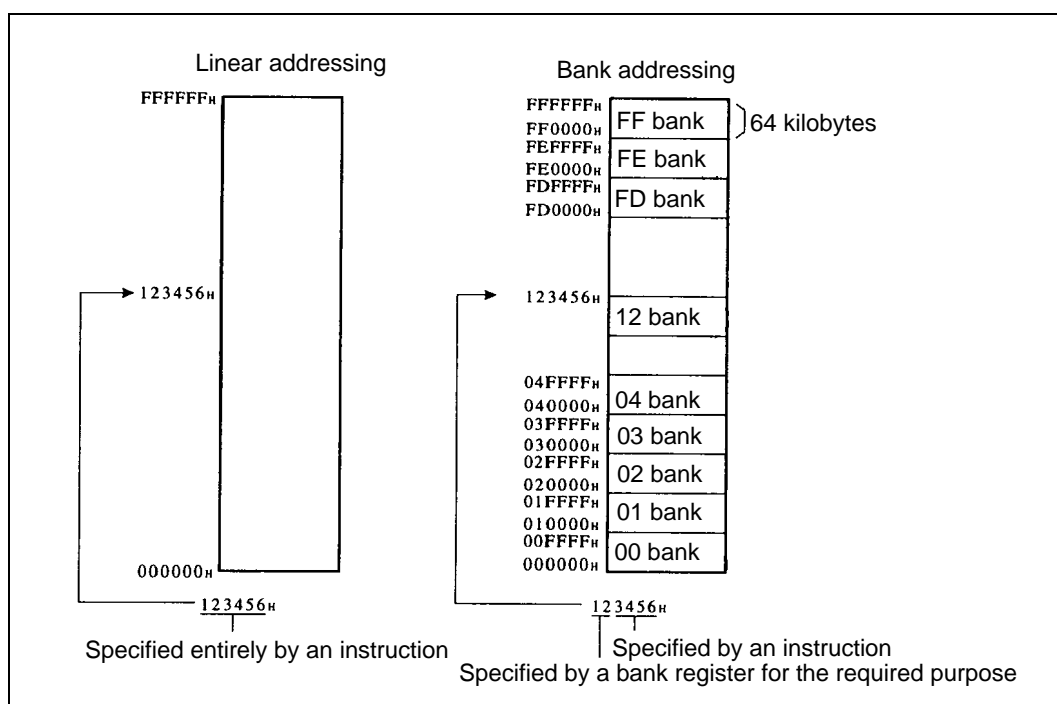


Figure 2.4-1 Linear addressing and bank addressing memory management

2.4 Addressing

2.4.1 Address specification by linear addressing

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

■ Linear addressing by 24-bit operand specification

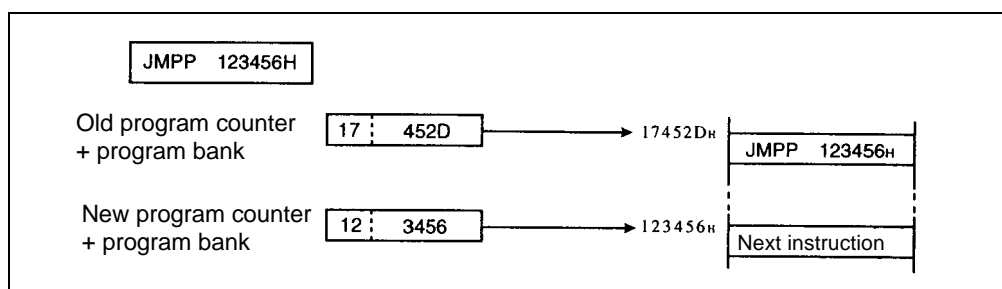


Figure 2.4-2 Example of direct specification of a 24-bit physical address in linear addressing

■ Addressing by indirect specification with a 32-bit register

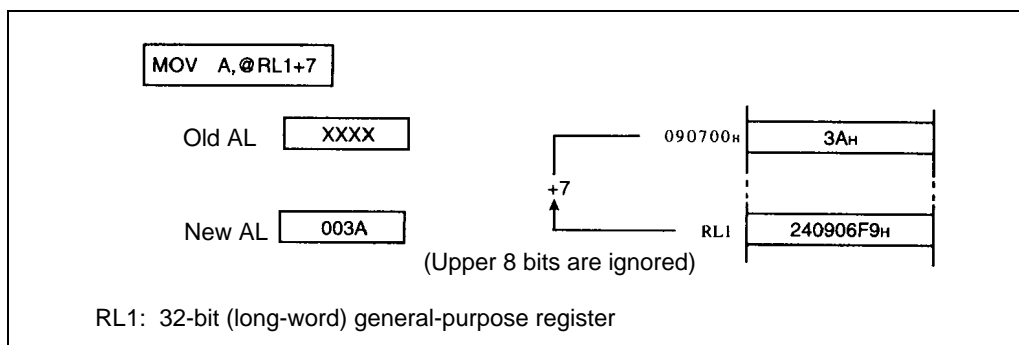


Figure 2.4-3 Example of indirect specification with a 32-bit general-purpose register in linear addressing

2.4.2 Address specification by bank addressing

In address specification by bank addressing, the 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 registers classified by function are as follows:

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

■ Bank registers and access space

Table 2.4-1 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 register (CCR:S) is 1. The USB is used when the stack flag in the condition register (CCR:S) is 0. (*1)	00 _H
System stack bank register (SSB) (*1)			00 _H
Additional bank register (ADB)	Additional (AD) space	Data that overflows from the data (DT) space is stored.	00 _H

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

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.

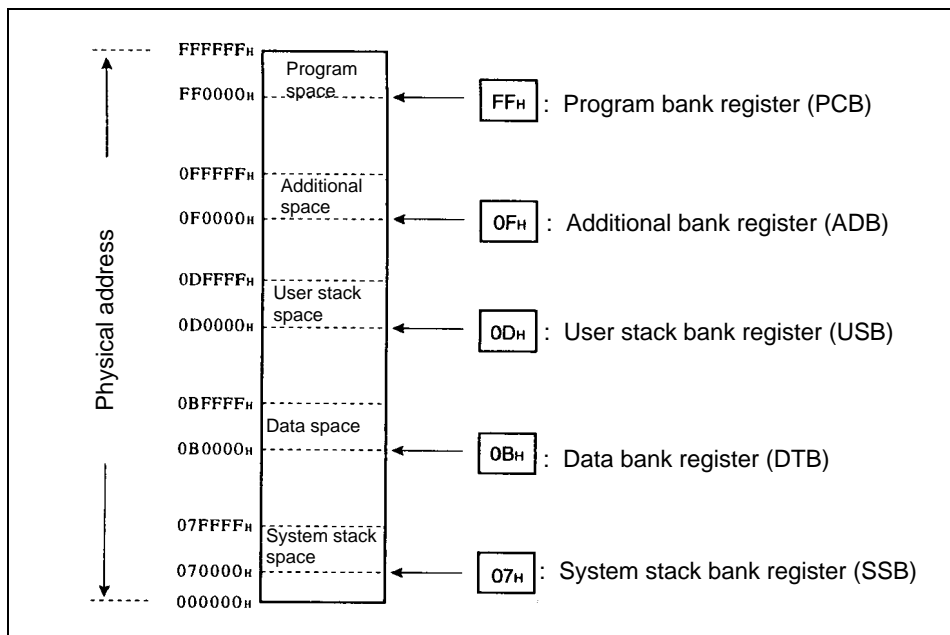


Figure 2.4-4 Sample bank addressing

■ Bank addressing and default space

To improve instruction coding efficiency, each instruction has a defined default space for each addressing method, 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 followed by the upper 16 bits.

If a reset signal is input immediately after the low-order data is written, the high-order data may not be written.

■ Storage of multibyte data in RAM

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.

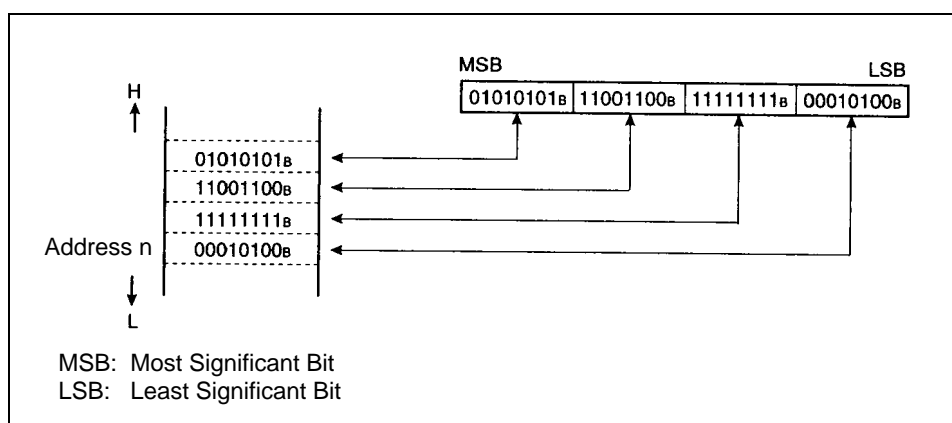


Figure 2.5-1 Storage of multibyte data in RAM

■ Storage of multibyte operand

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

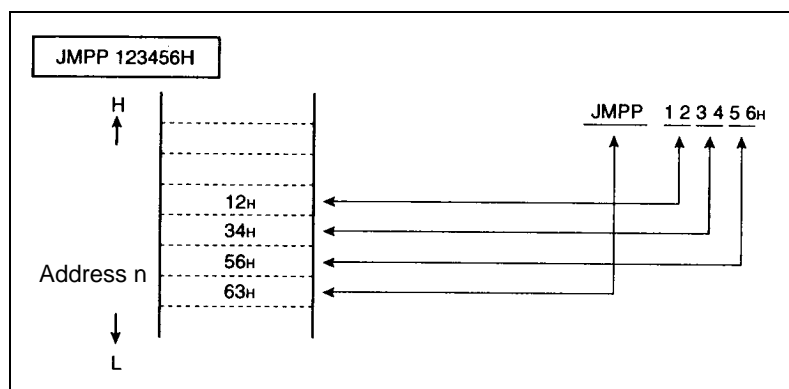


Figure 2.5-2 Storage of a multibyte operand

■ Storage of multibyte data in a stack

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

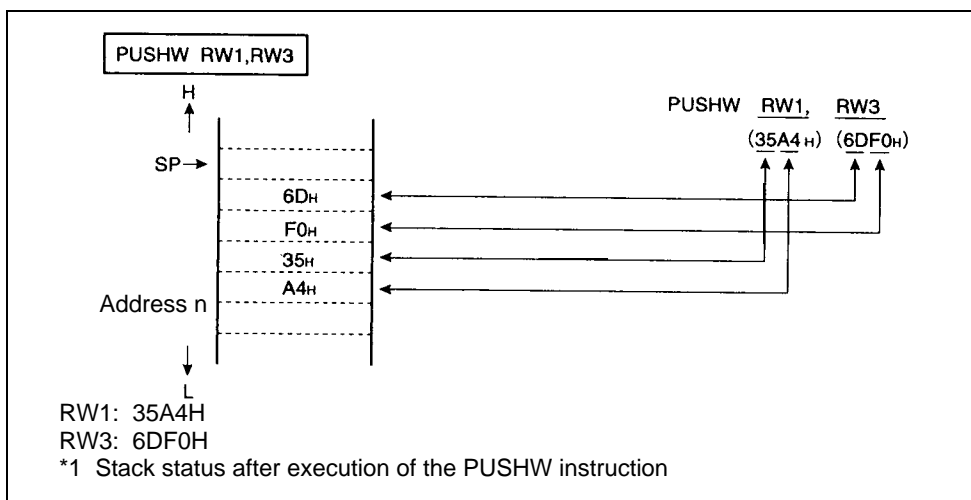


Figure 2.5-3 Storage of multibyte data in a stack

■ Multibyte data access

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

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

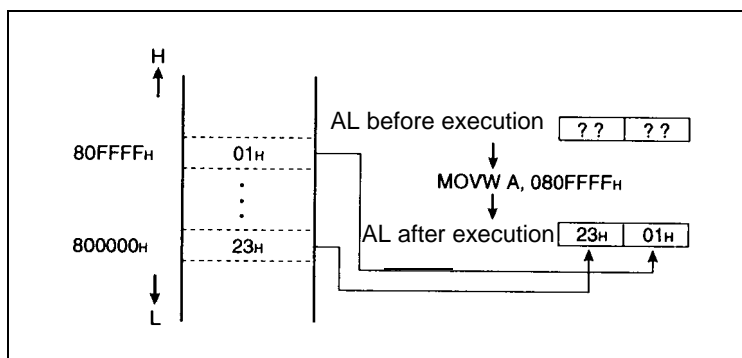


Figure 2.5-4 Multibyte data access on a bank boundary

2.6 Registers

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

■ Dedicated registers and general-purpose registers

Dedicated registers are dedicated hardware inside the CPU with limited use in the CPU architecture.

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

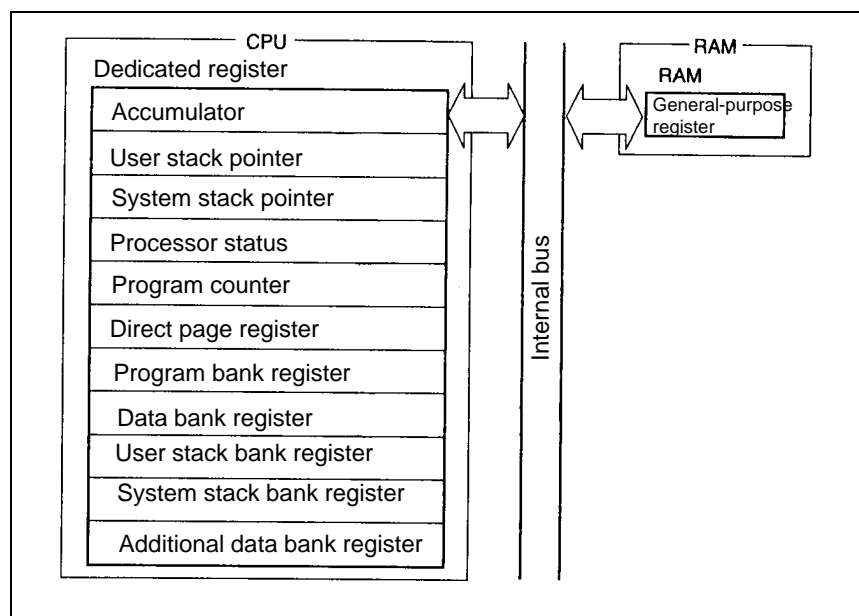


Figure 2.6-1 Dedicated registers and general-purpose registers

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

2.7 Dedicated Registers

the dedicated registers.

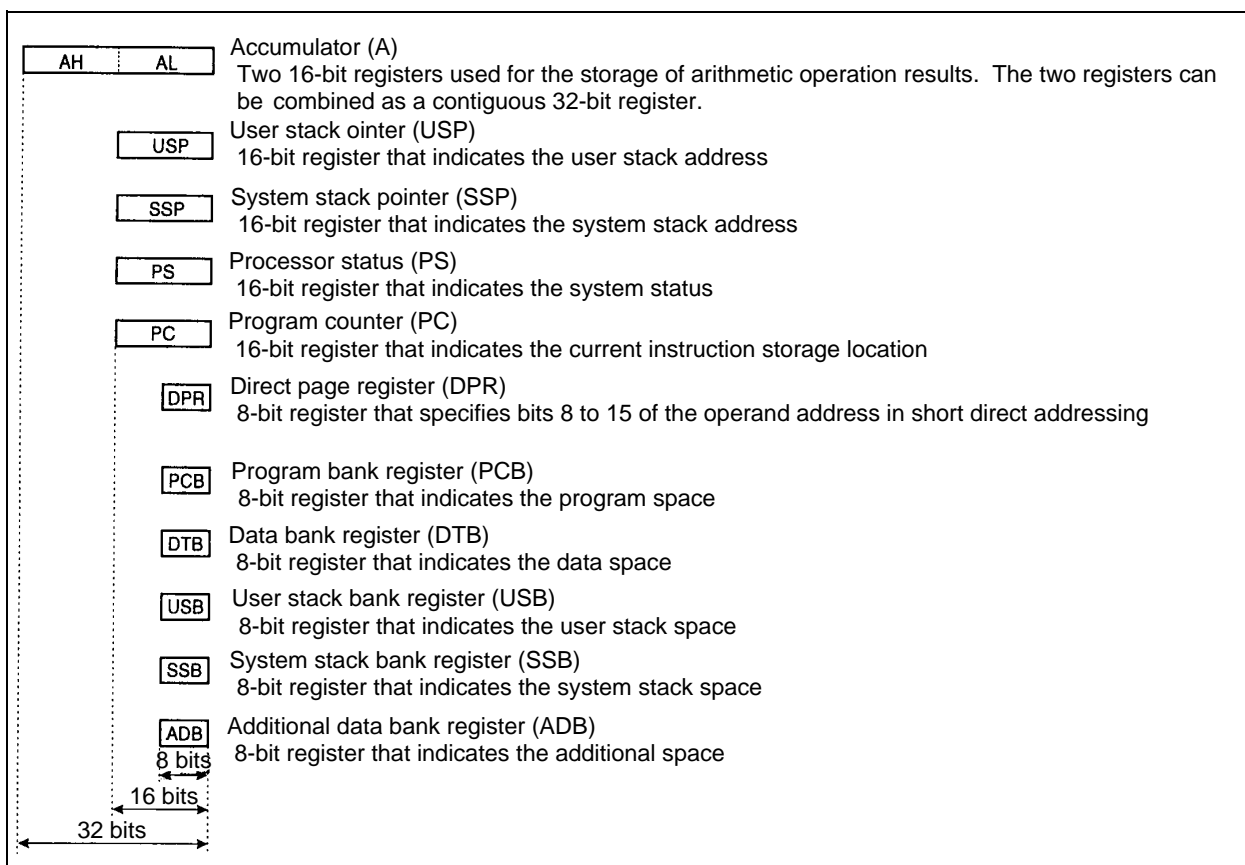


Figure 2.7-1 Configuration of dedicated registers

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)	<div style="text-align: center;"> <div style="display: flex; justify-content: space-around; font-size: small;"> bit15 to bit13 bit12 to bit8 bit7 to bit0 </div> <div style="display: flex; align-items: center; justify-content: center;"> <div style="text-align: center;"> ILM 0 0 0 0 0 0 0 0 </div> <div style="text-align: center;"> RP - 0 1 </div> <div style="text-align: center;"> CCR x x x x x x x x </div> </div> </div>
Program counter (PC)	Value in reset vector (contents of FFFFDCH, FFFDDH)
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 to 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.

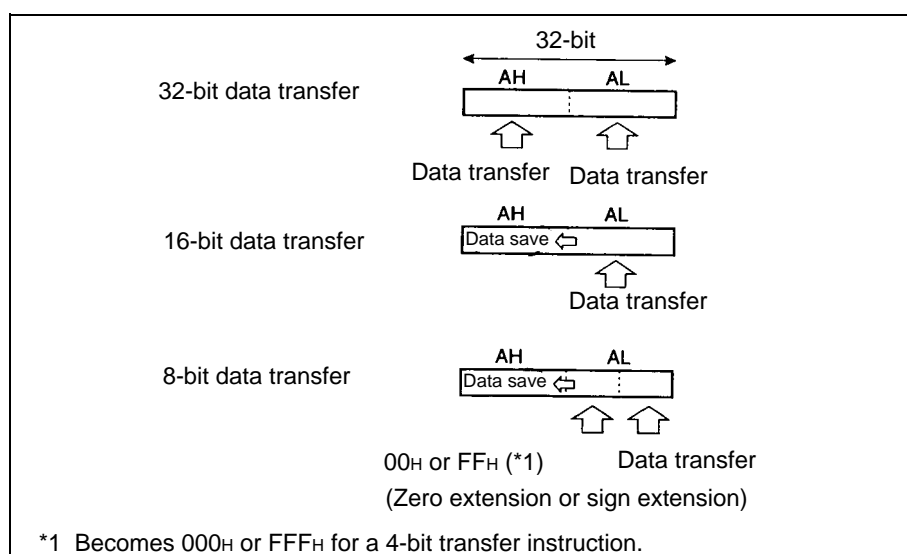


Figure 2.7-2 Data transfer to the accumulator

2.7 Dedicated Registers

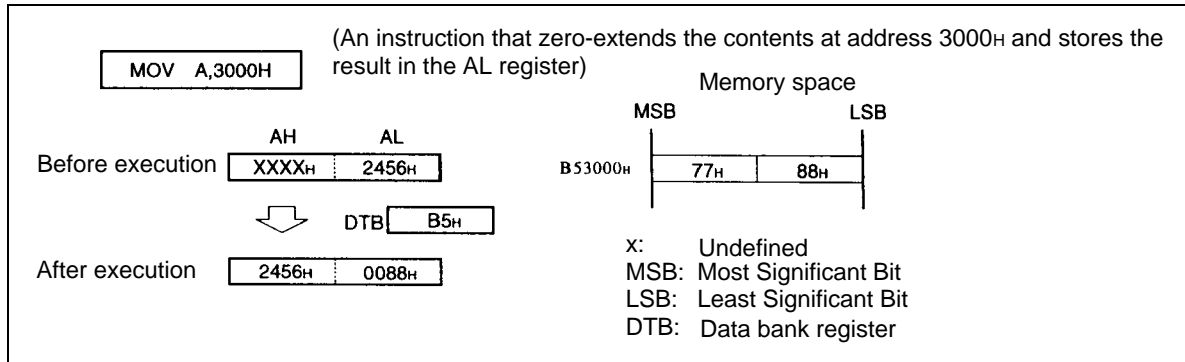


Figure 2.7-3 Example of AL-AH transfer in the accumulator (A) (8-bit immediate value, zero extension)

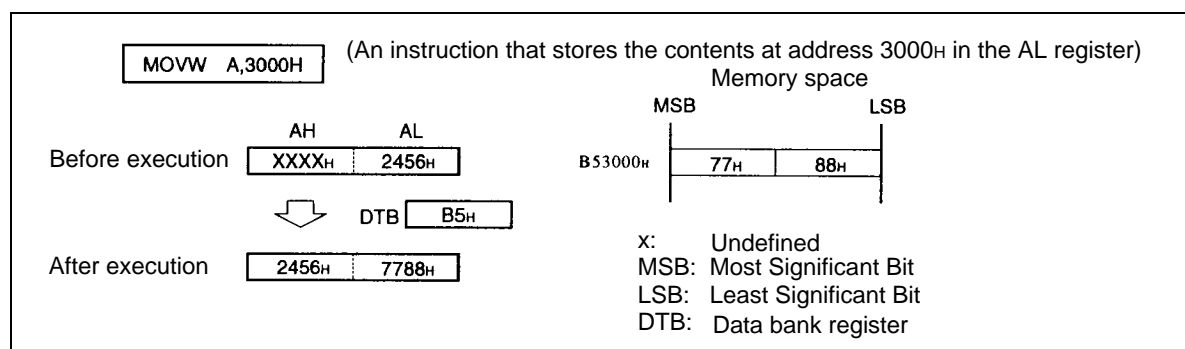


Figure 2.7-4 Example of AL-AH transfer in the accumulator (A) (8-bit immediate value, sign extension)

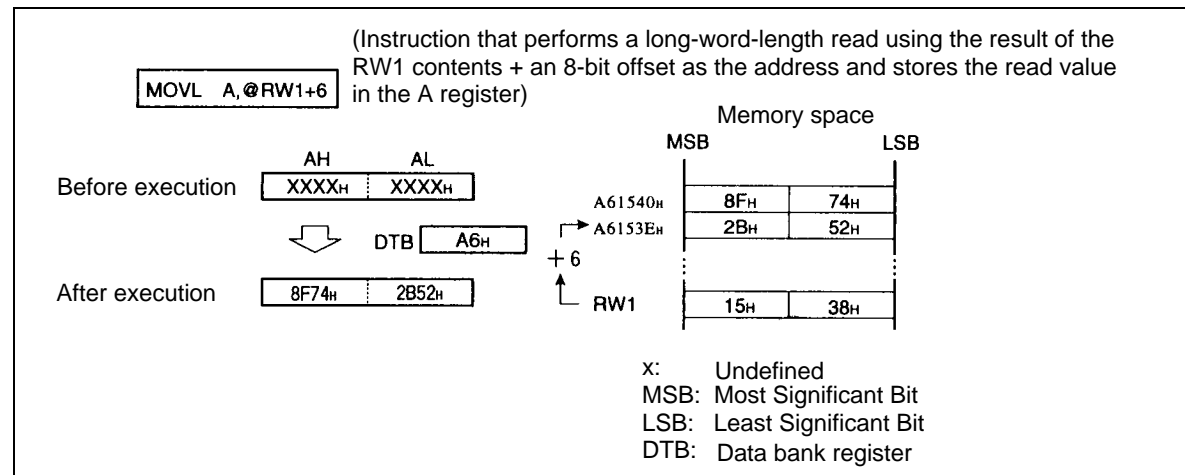


Figure 2.7-5 Example of 32-bit data transfer to the accumulator (A) (register indirect)

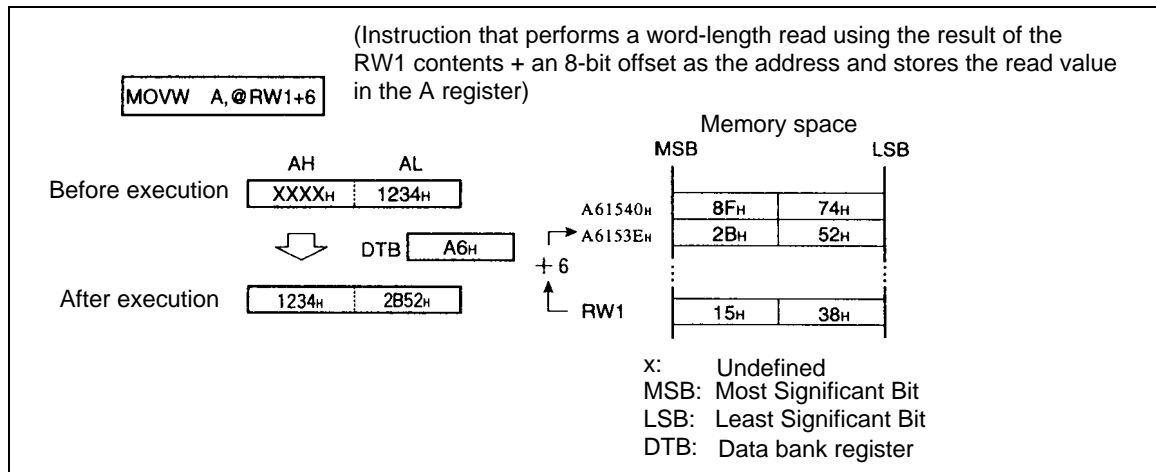


Figure 2.7-6 Example of AL-AH transfer in the accumulator (A) (16 bits, register indirect)

- **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.

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 F²MC-16L 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 (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.

2.7 Dedicated Registers

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

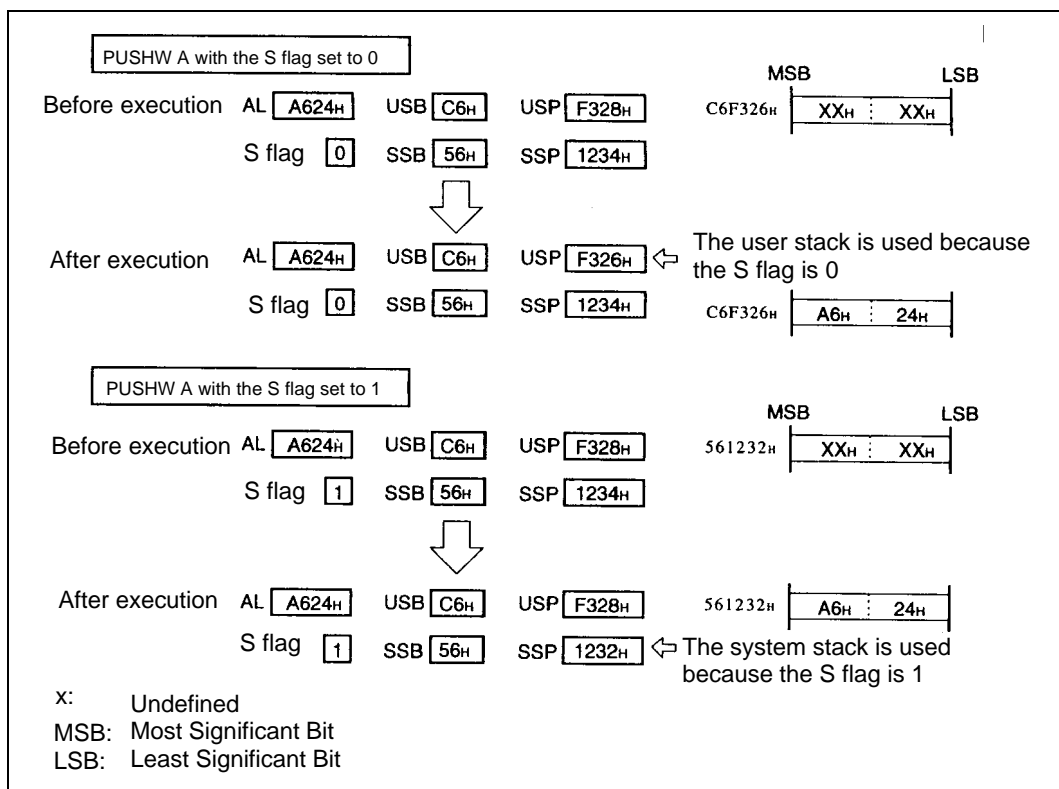


Figure 2.7-7 Stack operation instruction and stack pointer

<Notes>

- To set a value for the stack pointer, generally 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).

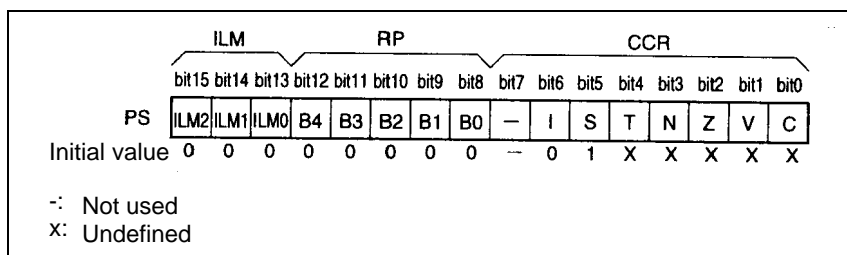


Figure 2.7-8 Processor status (PS) configuration

● 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 set 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.

2.7 Dedicated Registers

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.

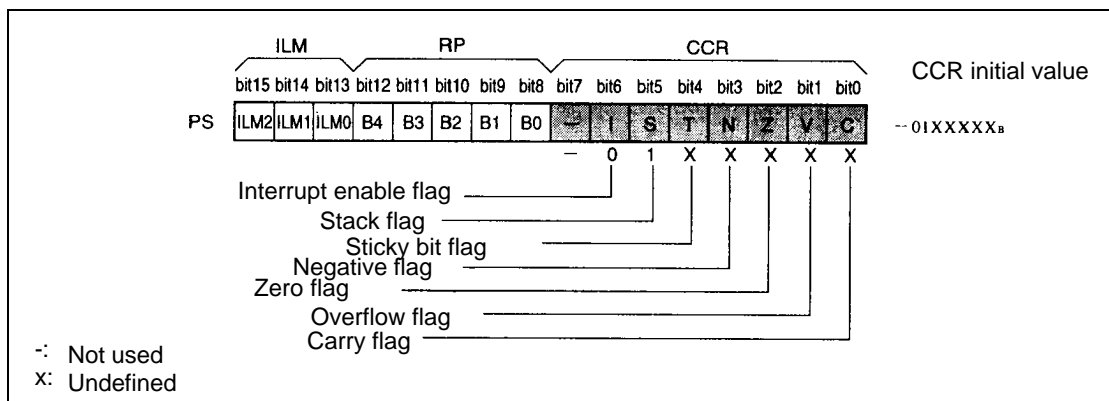


Figure 2.7-9 Condition code register (CCR) configuration

● 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. Cleared by a reset.

● Stack flag (S)

This flag indicates the pointer 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)

Set to 1 when the data shifted out by the carry contains at least one 1 during execution of a logical right shift instruction or arithmetic right shift instruction. Otherwise, set to 0. Also set to 0 when the shift amount 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.

2.7 Dedicated Registers

- **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.

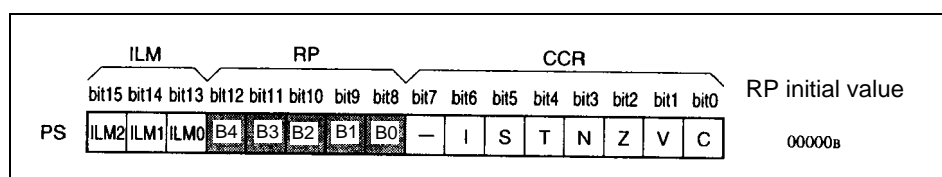


Figure 2.7-10 Configuration of the register bank pointer (RP)

■ General-purpose register area and register bank pointer

The register bank pointer points to the relationship between the general-purpose register of the F²MC-16L 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.

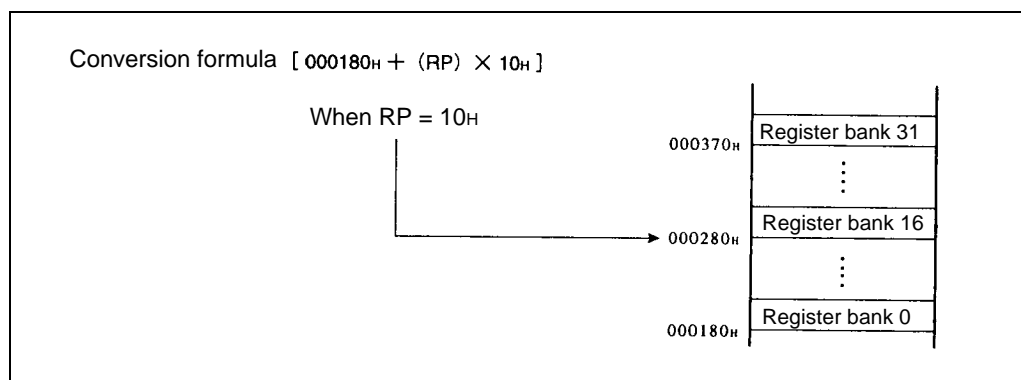


Figure 2.7-11 Conversion rules for physical address of general-purpose register area

- 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 0003FH.
- 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.

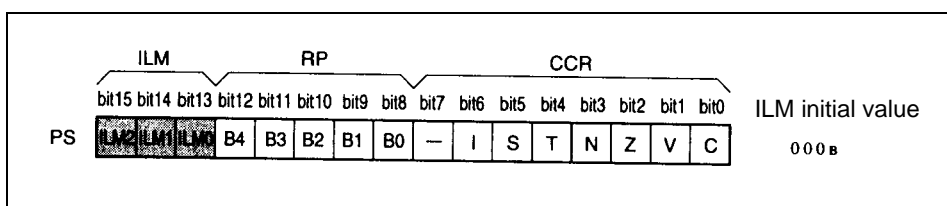


Figure 2.7-12 Configuration of the interrupt level mask register (ILM)

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) ↑ ↓ Lowest
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	

2.7 Dedicated Registers

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 bus pointer for reading operands.

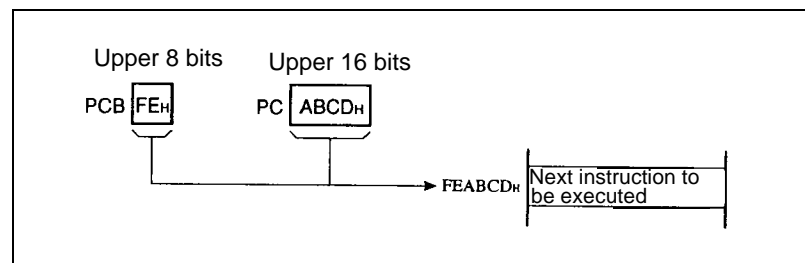


Figure 2.7-13 Program counter (PC)

<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.

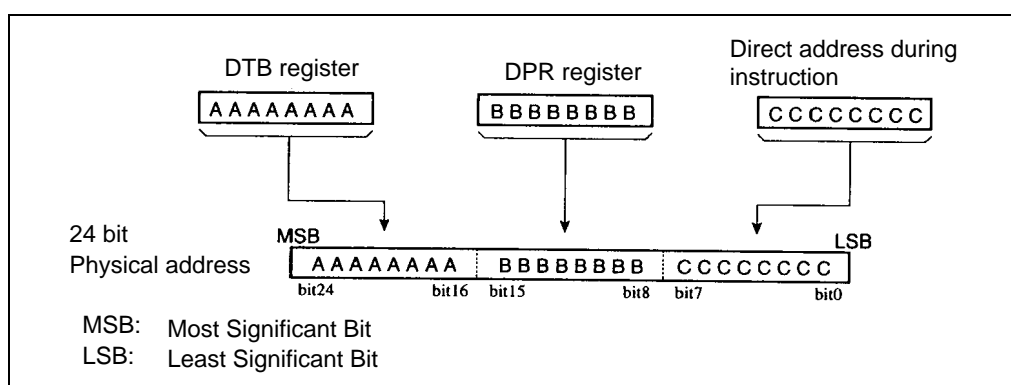


Figure 2.7-14 Physical address generation by the direct page register (DPR)

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

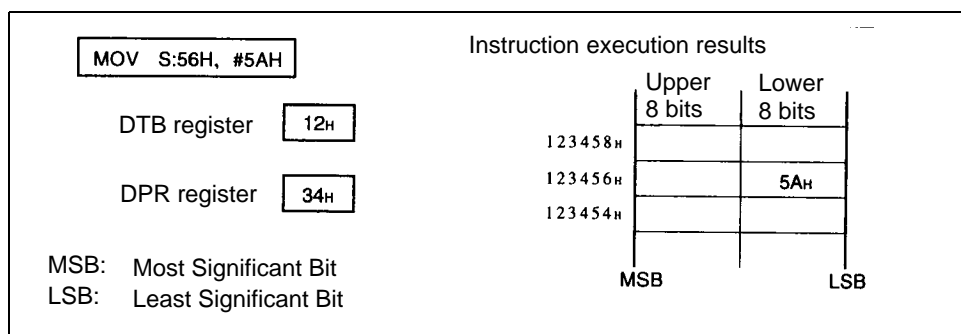


Figure 2.7-15 Example of direct page register (DPR) setting and data access

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 rewritten 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 MB90495 series supports up to the memory space contained in the device.

See Section 2.4.2, “Address specification by bank addressing,” for the operation of each register.

2.8 General-Purpose Registers

The general-purpose registers are a memory block allocated in RAM at 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 × 10H

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

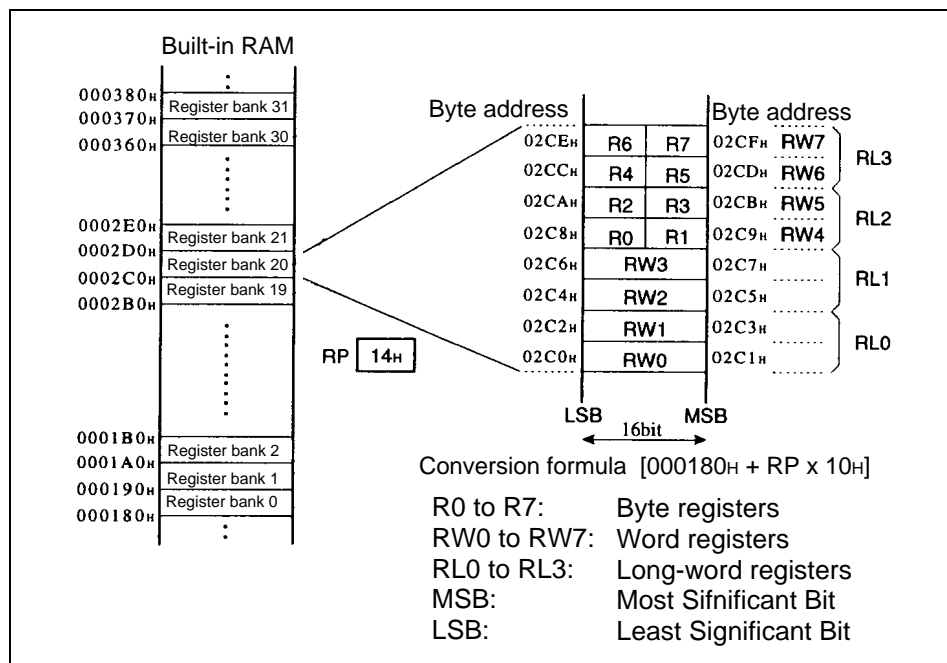


Figure 2.8-1 Location and configuration of the general-purpose register banks in the memory space

<Check>

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

2.8 General-Purpose Registers

■ 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 000180H to 00018FH 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.

2.9 Prefix Codes

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

2.9 Prefix Codes

Table 2.9-3 Instructions whose 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 000180_H to 00018F_H 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-16L provides a common register bank for tasks, which is called the common bank. The common bank is located at address 000180_H to 00018F_H.

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 000180_H to 00018F_H 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 SCEQ FILS	MOVSW SCWEQ FILSW	Do not place the CMR prefix before the string instruction.
Flag change instruction	AND CCR, #imm8	OR CCR, #imm8	The effect of the prefix extends to the next instruction.
PS return instruction	POPW PS		The effect of the prefix extends to the next instruction.
ILM setting instruction	MOV ILM, #imm8		The effect of the prefix extends to the next instruction.

2.9.3 Flag change suppression prefix (NCC)

The flag change suppression prefix (NCC) code is 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 whose use requires caution when the flag change suppression prefix (NCC) is used

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

2.9.4 Restrictions on Prefix Codes

The following 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.

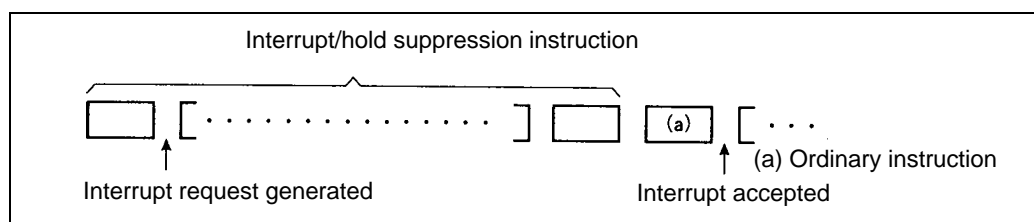


Figure 2.9-1 Interrupt/hold suppression

● 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.

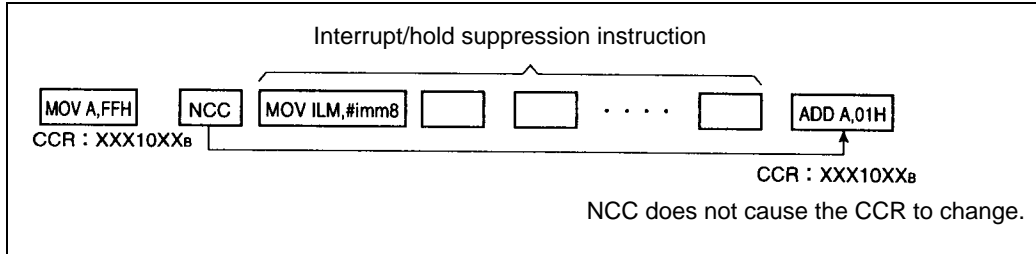


Figure 2.9-2 Interrupt/hold suppression instructions and prefix codes

■ 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.

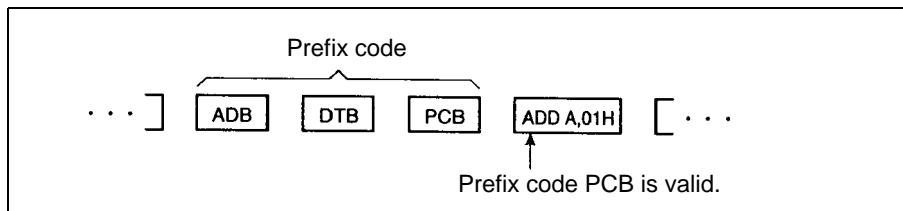


Figure 2.9-3 Consecutive prefix codes

CHAPTER 3 RESETS

This chapter describes resets for the MB90495-series microcontrollers.

- 3.1 Resets
- 3.2 Reset Causes and Oscillation stabilization Wait Intervals
- 3.3 External Reset Pin
- 3.4 Reset Operation
- 3.5 Reset Cause Bits
- 3.6 Status of Pins in a Reset

3.1 Resets

If a reset is generated, the CPU immediately stops the current execution process and waits for the reset to be cleared. The CPU then begins processing at the address indicated by the reset vector.

The four causes of a reset are as follows

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

■ Causes of a Reset

Table 3.1-1 lists the causes of a reset.

Table 3.1-1 Causes of a Reset

Type of reset	Cause	Machine clock	Watchdog timer	Oscillation stabilization wait
Power-on	When the power is turned on	Main clock (MCLK)	Stop	Yes
External pin	L level input to RSTX pin	Main clock (MCLK)	Stop	No
Software	A "0" is written to the RST bit of the low power consumption mode control register (LPMCR).	Main clock (MCLK)	Stop	No
Watchdog timer	Watchdog timer overflow	Main clock (MCLK)	Stop	No

Main clock: Oscillation clock frequency 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 is 16 machine cycles (16/φ). The oscillation stabilization wait interval is not required for external resets.

Reference

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 cleared after completion of the instruction only for reset requests via the RSTX pin. Therefore, the normal write operation is completed even though a reset is input concurrently.

Note that a reset may prevent the data transfer requested by a string-processing instruction (such as MOVS) from being completed because the reset is accepted before a specified number of bytes are transferred.

3.1 Resets

○ Software reset

A software reset is an internal reset generated by writing “0” to the RST bit of the low power consumption mode control register (LPMCR). The oscillation stabilization wait interval is not required for a software reset.

○ Watchdog timer reset

A watchdog timer reset is generated by a watchdog timer overflow that occurs when “0” is written to the WTE bit of the watchdog timer control register (WDTC) within a given time after the watchdog timer is activated. 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 intervals for an evaluation/flash model and for a mask model are fixed at $2^{18}/\text{HCLK}$ (approx. 65.54 ms) and $2^{17}/\text{HCLK}$ (approx. 32.77 ms), respectively. When the oscillation stabilization wait interval has elapsed, the reset is executed.

Reference Definition of clocks

HCLK: Oscillation clock

MCLK: Main clock

SCLK: Sub-clock

ϕ : Machine clock (CPU operating clock)

$1/\phi$: Machine cycle (CPU operating clock period)

See Section 4.1 "Clocks" for details on machine clocks.

Note

If a reset occurs in stop or sub-clock mode, an oscillation stabilization wait interval of $2^{17}/\text{HCLK}$ (approx. 32.77 ms) is required.

See Section 4.4 "Clock Mode" for details on clock modes.

3.2 Reset Cause and Oscillation Stabilization Wait Intervals

The MB90495 has four reset causes. The oscillation stabilization wait interval for a reset depends on the reset cause.

■ Reset Causes and Oscillation Stabilization Wait Intervals

Table 3.1-1 summarizes 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	Model under evaluation/flash model: $2^{18}/\text{HCLK}$ (approx. 65.54 ms) Mask model: $2^{17}/\text{HCLK}$ (approx. 32.77 ms)
Hardware standby	$2^{17}/\text{HCLK}$ (approx. 32.77 ms)
Watchdog timer	None; though bits WS1 and WS0 are initialized to "11".
External reset via the RSTX pin	None; though bits WS1 and WS0 are initialized to "11".
Software reset	None; though bits WS1 and WS0 are initialized to "11".

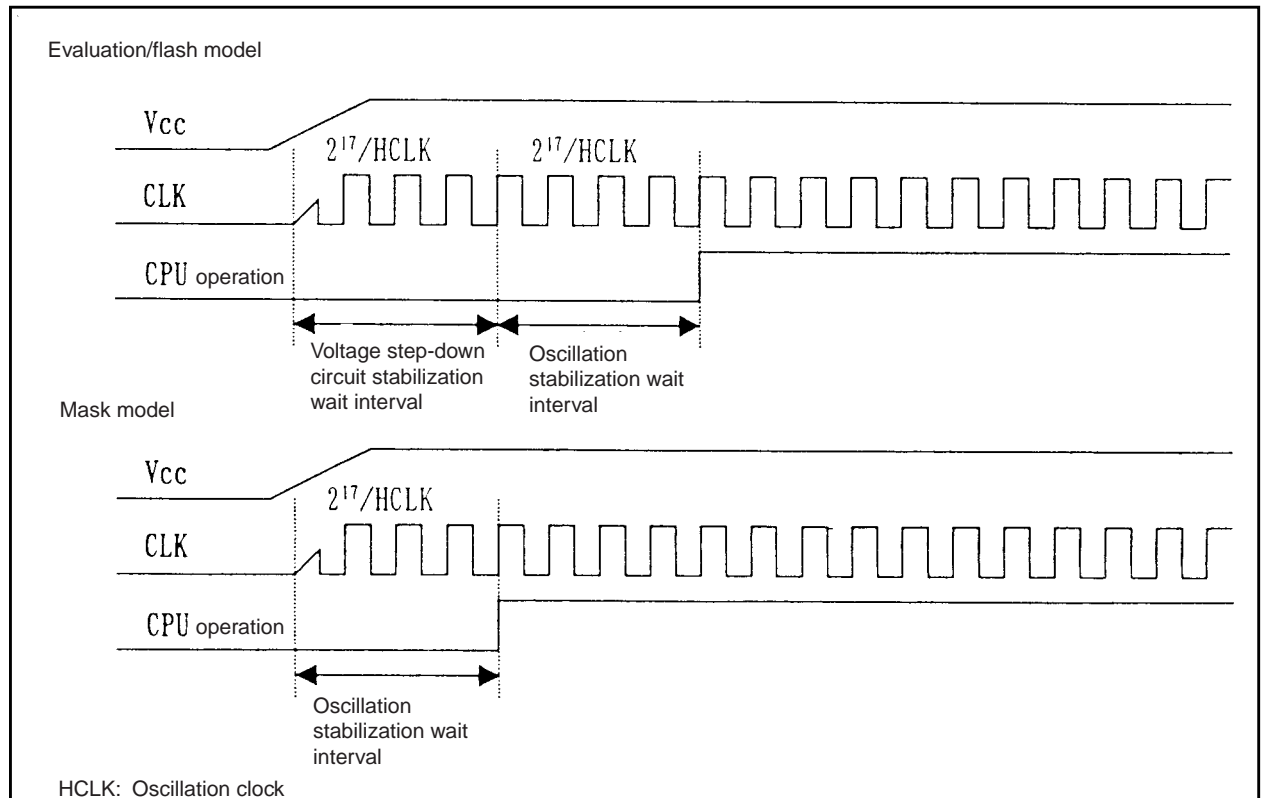
HCLK: Oscillation clock

WS1 and WS0: Oscillation stabilization wait interval selection bits of the clock selection register (CKSCR)

Figure 3.2-1 shows the oscillation stabilization wait intervals of an evaluation/flash model and a mask model at a power-on reset.

3.2 Reset Cause and Oscillation Stabilization Wait Intervals

Figure 3.2-1 Oscillation Stabilization Wait Intervals of an Evaluation/Flash Model and a Mask Model at a Power-on Reset



Note

Ceramic and crystal oscillators generally require an oscillation stabilization wait interval of several milliseconds to 10 to 20 milliseconds, until stabilization at a natural frequency is attained. A proper oscillation stabilization wait interval must be set for the particular oscillator used.

See Section 4.5 "Oscillation Stabilization Wait Interval" for details about oscillation stabilization wait intervals.

■ Oscillation Stabilization Wait and Reset State

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

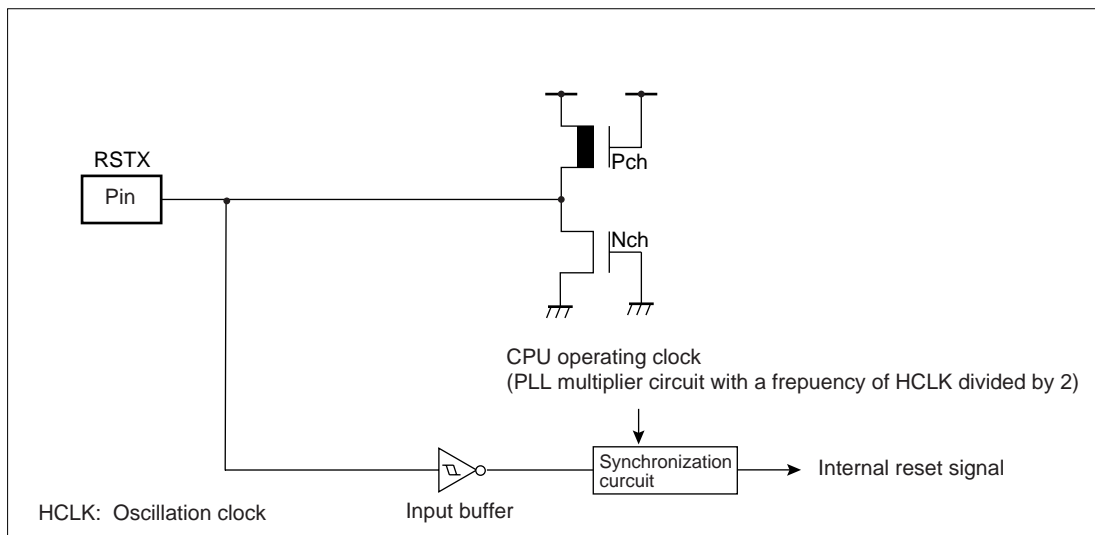
3.3 External Reset Pin

The external reset pin (RSTX pin) is an input pin used exclusively for a reset. Inputting an L level signal generates an internal reset. For the MB90495-series, resets are generated in synchronization with the CPU operating clock. However, a reset generated via the external pin is asynchronous with the CPU operating clock.

■ Block Diagrams of the External Reset Pin

○ Block diagram of internal reset

Figure 3.3-1 Block Diagram of Internal Reset



Note

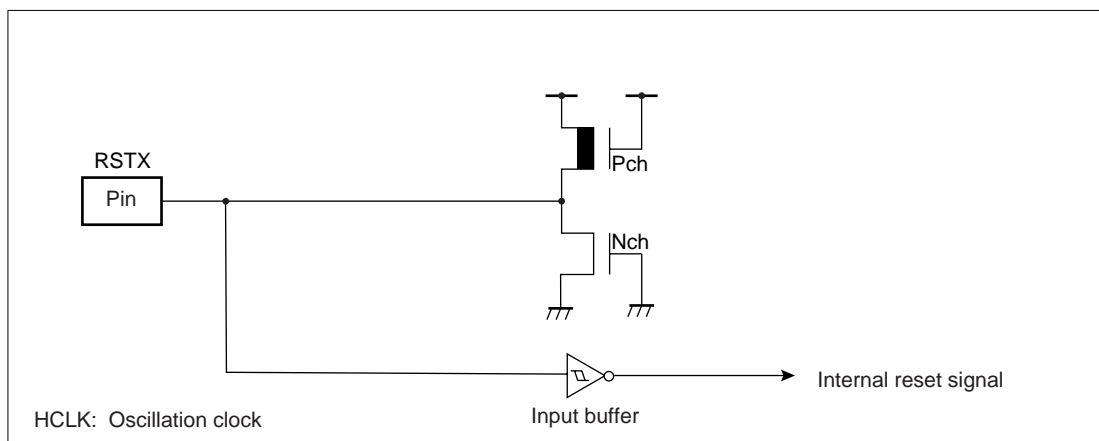
Inputs to the RSTX are accepted during cycles in which memory is not affected to prevent memory from being destroyed by a reset during a write operation.

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

3.3 External Reset Pin

- Block diagram of the external reset pin

Figure 3.3-2 Block Diagram of Internal Reset for External Pin



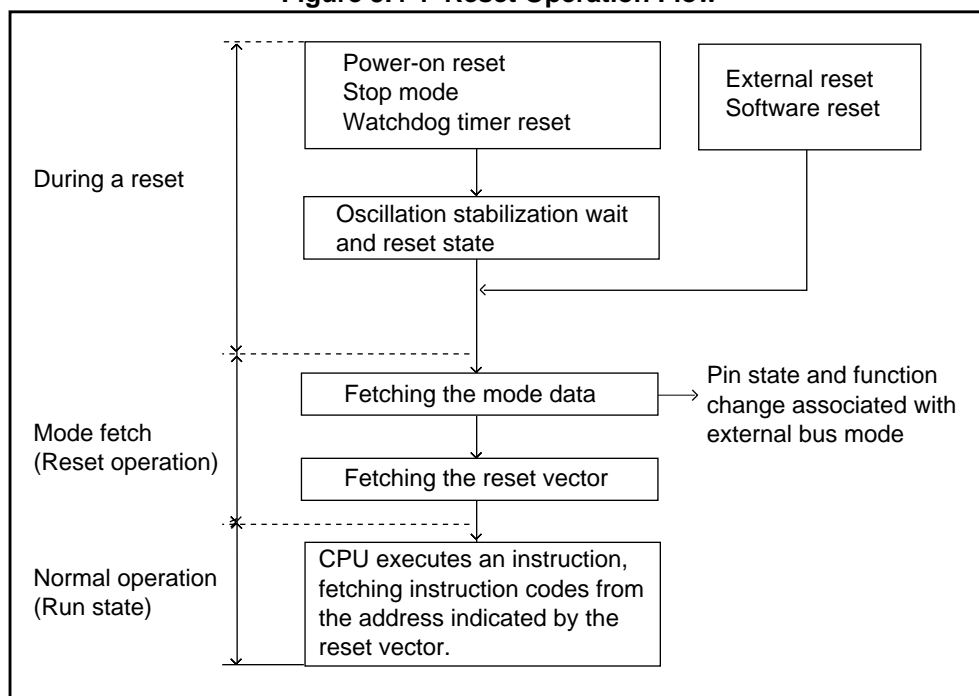
3.4 Reset Operation

When a reset is cleared, the memory locations from which the mode data and the reset vectors are read are selected according to the setting of the mode pins, and a mode fetch is performed. Mode setting data determines the CPU operating mode and the execution start address after a reset operation ends. For power-on or recovery from sub-clock mode or stop mode by a reset, a mode fetch is performed when the oscillation stabilization wait interval elapses.

■ 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 Section 22.2 "Mode Pins (MD2 to MD0)" for details on mode pins.

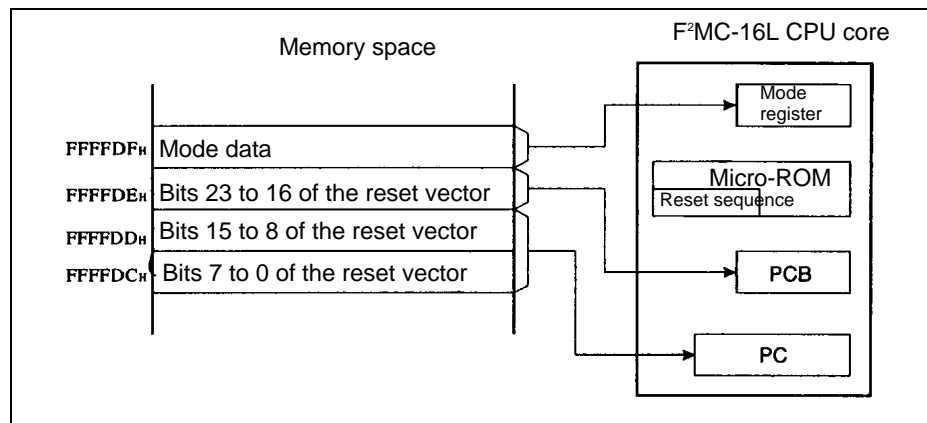
3.4 Reset Operation

■ Mode Fetch

When the reset is cleared, the CPU transfers the reset vector and the mode data to the appropriate registers in the CPU core by hardware. 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 then fetches the reset vector and mode data. Using mode fetching, the CPU can begin processing at the address indicated by the reset vector.

Figure 3.4-1 shows the transfer of the reset vector and mode data.

Figure 3.4-1 Transfer of Reset Vector and Mode Data



○ Mode data (address: FFFFDF_H)

Only a reset operation changes the contents of the mode register. The mode register setting is valid after a reset operation. See Section 22.3 "Mode Data" for details on 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.

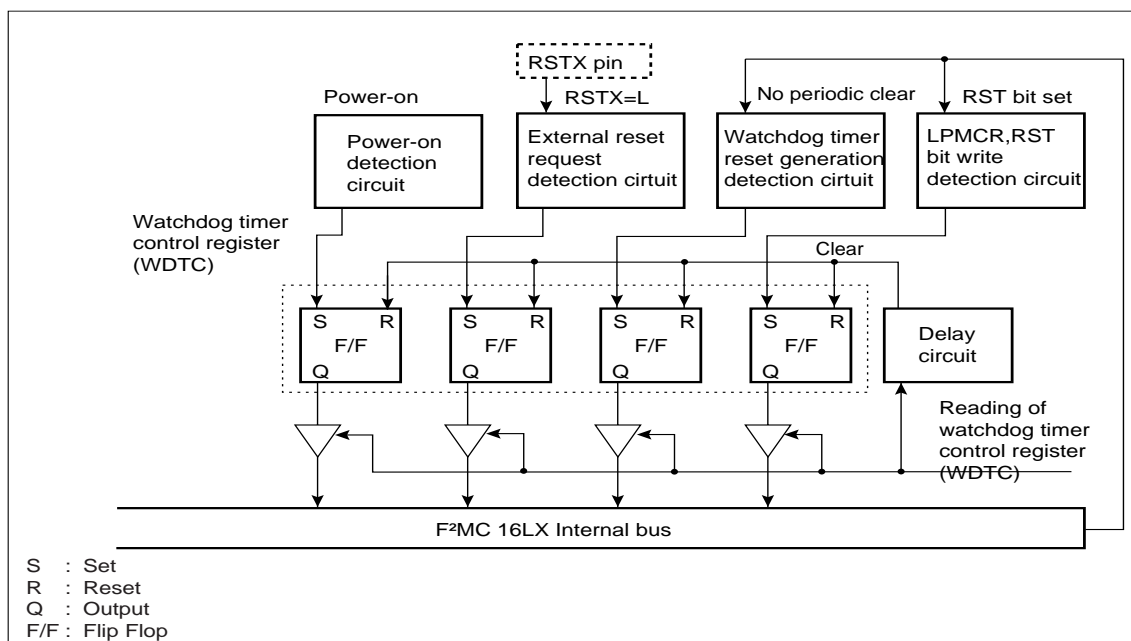
3.5 Reset Cause Bits

A reset cause can be identified by reading 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 the cause of a reset must be identified after the reset has been cleared, the value read from the WDTC should be processed by the software and a branch made to the appropriate program. Figure 3.5-1 describes general FMC16-LX reset function. Please note that MB90495-series MCU does not have HSTX pin. Functionality is as if HSTX is permanently asserted to H.

Figure 3.5-1 Block Diagram of Reset Cause Bits



3.5 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. See Section 10.2 "Watchdog Timer Control Register (WDTC)" for details.

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

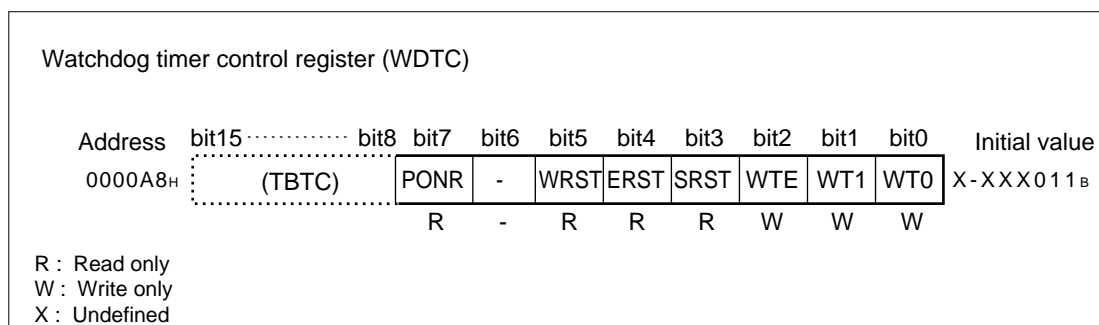


Table 3.5-1 Correspondence between Reset Cause Bits and Reset Causes

Reset cause	PONR	STBR	WRST	ERST	SRST
Power-on reset	1	X	X	X	X
Hardware standby request via HSTX pin. (Not in MB90495)	*	1	*	*	*
Watchdog timer overflow	*	*	1	*	*
External reset request via RSTX pin	*	*	*	1	*
Software reset request	*	*	*	*	1

*: Previous state defined

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

○ Power-on reset

For a power-on reset, because the PONR bit is set to "1" but all other reset cause bits are undefined, the software should be programmed so that it will ignore all reset cause bits except the PONR bit when it is "1".

CHAPTER 3 RESETS

○ Clearing the reset cause bits

The reset cause bits are cleared only when the watchdog timer control register (WDTC) is read. Any bit corresponding to a reset cause that has already been generated is not cleared even though another reset is generated (a setting of “1” is retained).

Note

If the power is turned on under conditions where no power-on reset occurs, the value in this register may not be guaranteed.

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, and M0 = 00_B):

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

Note

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

See Table 5.7-1 for information on the state of pins during a reset.

CHAPTER 4 CLOCKS

This chapter describes the clocks used by MB90495-series microcontrollers.

- 4.1 Clocks
- 4.2 Block Diagram of the Clock Generation Block
- 4.3 Clock Selection Register (CKSCR)
- 4.4 Clock Mode
- 4.5 Oscillation Stabilization Wait Interval
- 4.6 Connection of an Oscillator or an External Clock to the Microcontroller

4.1 Clocks

The clock generation block controls the operation of the internal clock that controls operation of the CPU and peripheral functions. This internal clock is called the machine clock. One internal clock cycle is called 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 generation block contains the oscillation circuit that generates the oscillation clock. An external oscillator is attached to this circuit. The oscillation clock can also be supplied by inputting an external clock to the clock generation block. The clock generation block also contains the PLL clock multiplier circuit, which generates four clocks whose frequencies are multiples of the oscillation clock frequency. The clock generation block controls the oscillation stabilization wait interval and PLL clock multiplication as well as internal clock operation by changing the clock with a clock selector.

○ Oscillation clock (HCLK)

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

○ Sub-clock (SCLK)

The sub-clock runs the clock timer and can also be used as a low-speed machine clock.

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

○ Main clock (MCLK)

The main clock, whose frequency is the oscillation clock frequency divided by 2, supplies the clock input to the time-base timer and the clock selector.

○ PLL clock (PCLK)

The PLL clock is obtained by multiplying the oscillation clock frequency 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 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 (whose frequency is the source clock frequency divided by 2) and the other four clocks (whose frequencies are multiples of the source clock frequency).

Note

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. A PLL oscillation of 3 to 16 MHz is possible, but this range

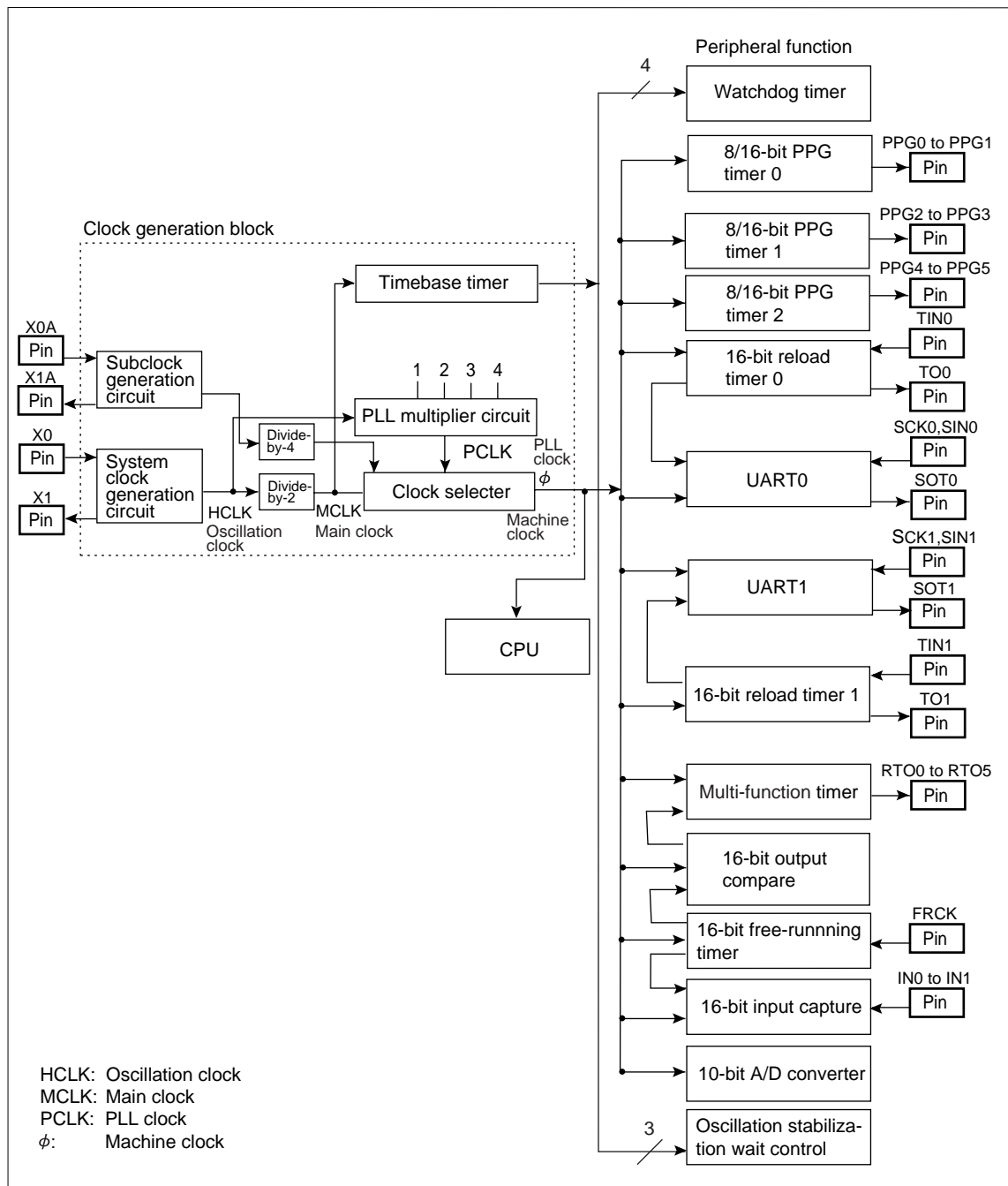
4.1 Clocks

depends on the operating voltage and multiplier.

■ Clock Supply Map

Since the machine clock generated in the clock generation block is supplied as the clock that controls the operation of the CPU and peripheral functions, the operation of the CPU and the peripheral functions is affected by switching between the main clock and the PLL clock (clock mode) and by a change in the PLL clock multiplier. Since some peripheral functions receive frequency-divided output from the time-base timer, a peripheral unit can select the clock best suited for this operation. Figure 4.1-1 shows the clock supply map.

Figure 4.1-1 Clock supply map



4.2 Block Diagram of the Clock Generation Block

The clock generation block consists of five blocks:

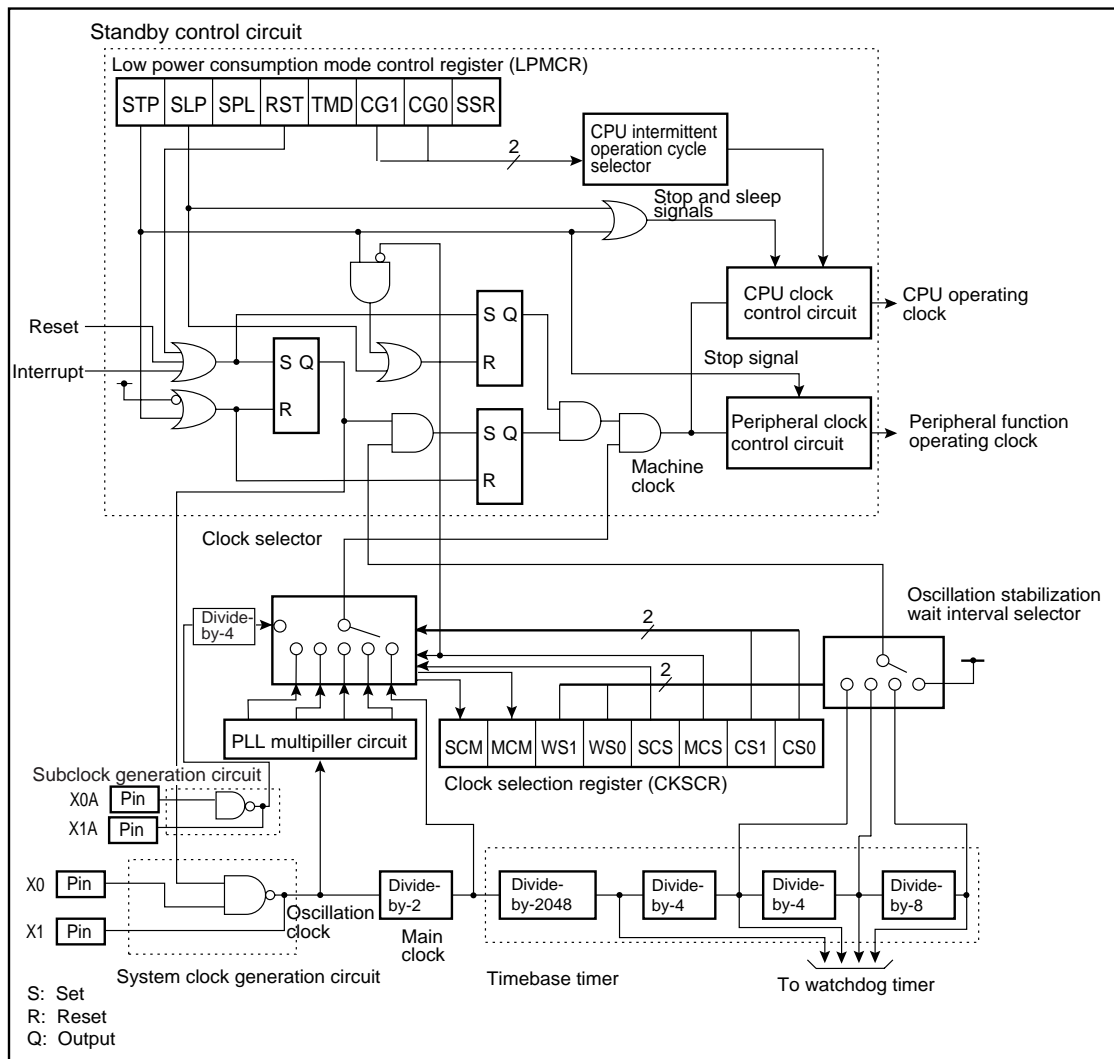
- System clock generation circuit
- PLL multiplier circuit
- Clock selector
- Clock selection register (CKSCR)
- Oscillation stabilization wait interval selector

■ Block Diagram of the Clock Generation Block

Figure 4.2-1 shows a block diagram of the clock generation block.

Figure 4.2-1 also includes the standby control circuit and time-base timer circuit.

Figure 4.2-1 Block Diagram of the Clock Generation Block



4.2 Block Diagram of the Clock Generation Block

- **System clock generation circuit**

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

- **Sub-clock generation circuit**

The sub-clock generation circuit generates a sub-clock (SCLK) from an external oscillator attached to it. An external clock can be also input to this circuit.

- **PLL multiplier circuit**

The PLL multiplier circuit multiplies the oscillation clock frequency through PLL oscillation and supplies a clock whose frequency is a multiple of the oscillation clock frequency 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 switch between the oscillation clock and a PLL clock and is also used to select an oscillation stabilization wait interval and a PLL clock multiplier.

- **Oscillation stabilization wait interval selector**

This oscillation stabilization wait interval selector selects an oscillation stabilization wait interval for the oscillation clock when the stop mode is released or when a watchdog timer reset occurs. Selection is made from among three different time-base timer outputs. In all other cases, an oscillation stabilization wait interval is not selected.

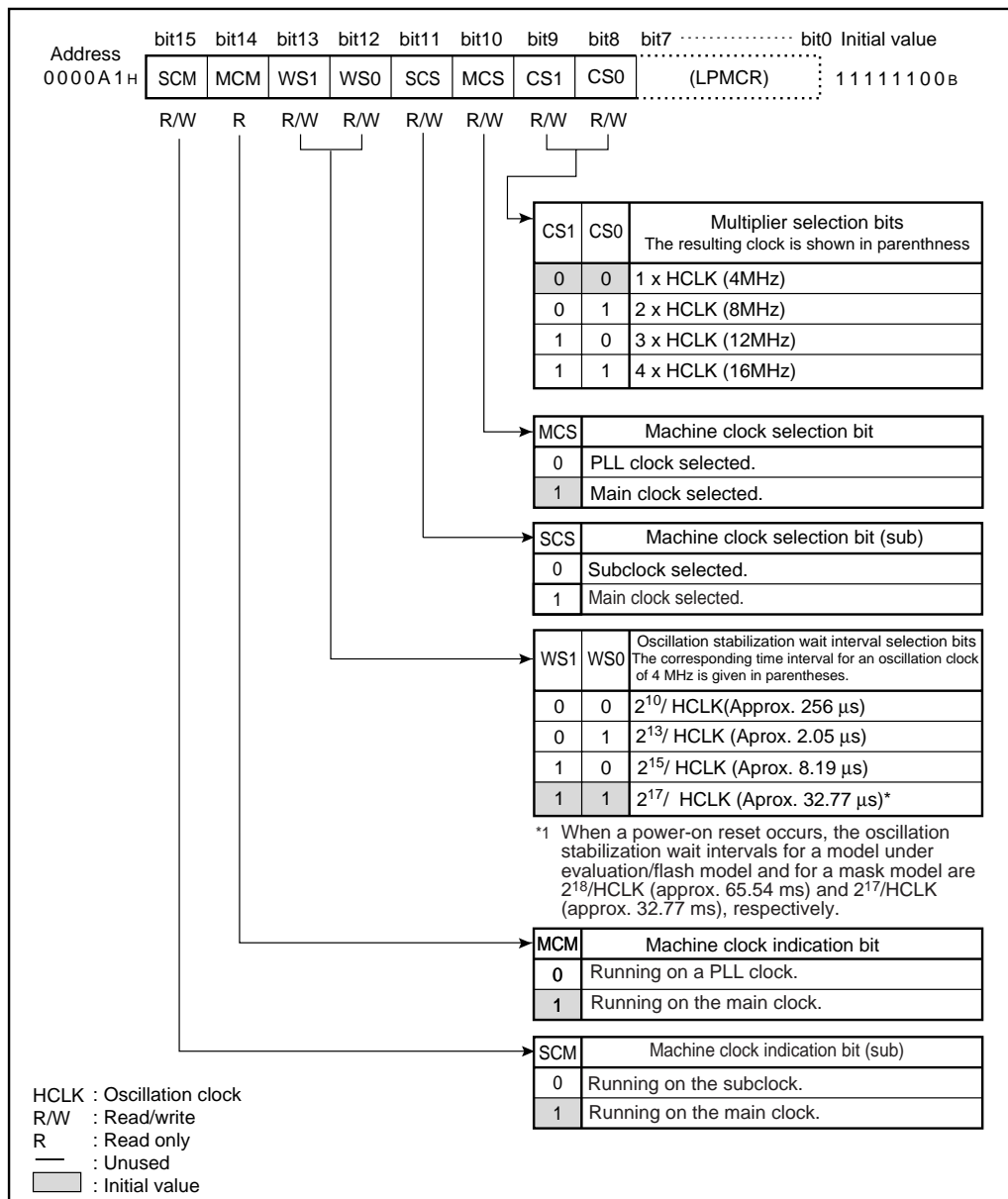
4.3 Clock Selection Register (CKSCR)

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



4.3 Clock Selection Register (CKSCR)

Note

The machine clock selection bit is initialized to main clock selection at a reset.

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

Bit name		Function
bit 15	SCM: Machine clock indication bit (sub)	<ul style="list-style-type: none"> This bit indicates whether the main clock or sub-clock has been selected as the machine clock. When this bit is "0", the sub-clock has been selected. When it is "1", the main clock has been selected. If SCS = 1 and SCM = 1, the main clock oscillation stabilization wait interval is in effect.
bit 14	MCM: Machine clock indication bit	<ul style="list-style-type: none"> This bit indicates whether the main clock or a PLL clock has been selected as the machine clock. When this bit is 0, a PLL clock has been selected. When it is 1, the main clock has been selected. If MCS = 0 and MCM = 1, the PLL clock oscillation stabilization wait interval is in effect.
bit 13 bit 12	WS1 and WS0: Oscillation stabilization wait interval selection bits	<ul style="list-style-type: none"> These bits select an oscillation stabilization wait interval of the oscillation clock when stop mode was released, when transition occurred from sub-clock mode to main clock mode, or when transition occurred from sub-clock mode to PLL clock mode. These bits are initialized to "11_B" by all reset causes. <p>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." These bits can be reset to "00_B" only for main clock mode.</p> <p>Reference The oscillation stabilization wait interval for all PLL clocks is fixed at $2^{14}/\text{HCLK}$.</p>
bit 11	SCS: Machine clock selection bit (sub)	<ul style="list-style-type: none"> This bit specifies whether the main clock or sub-clock is selected as the machine clock. When this bit is "0", the sub-clock is selected. When it is "1", the main clock is selected. If this bit has been set to "1" and "0" is written to it, the oscillation stabilization wait interval for the main clock starts. As a result, the time-base timer is automatically cleared. When the sub-clock has been selected, the operating clock frequency is the sub-clock frequency divided by 4 (that is, the machine clock frequency is 8 kHz when the sub-clock oscillation frequency is 32 kHz). If both the SCS and MCS bits are "0", the SCS bit takes precedence, that is, the sub-clock is selected. This bit is initialized to "1" by all reset causes.

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

Bit name		Function
bit 10	MCS: Machine clock selection bit	<ul style="list-style-type: none"> 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 it is 1, the main clock is selected. If this bit has been set to “1” and “0” is written to it, the oscillation stabilization wait interval for the PLL clock starts. As a result, the time-base timer is automatically cleared, and the TBOF bit of the time-base timer control register (TBTC) is also cleared. For PLL clocks, the oscillation stabilization wait interval is fixed at $2^{14}/\text{HCLK}$ (the oscillation stabilization wait interval is approx. 4.1 ms for an oscillation clock frequency of 4 MHz). When the main clock has been selected, the operating clock frequency is the oscillation clock frequency divided by 2 (that is, the operating clock is 2 MHz when the oscillation clock frequency is 4 MHz). This bit is initialized to “1” by all reset causes. <p>Caution When the MCS bit is “1”, write “0” to it only when the time-base timer interrupt is masked by the TBIE bit of the time-base timer control register (TBTC) or the interrupt level register (ILM).</p>
bit 9 bit 8	CS1 and CS0: Multiplier selection bits	<ul style="list-style-type: none"> 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. <p>Caution When the MCS or MCM 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).</p>

HCLK: Oscillation clock

4.4 Clock Mode

Three clock modes are provided: main clock mode, PLL clock mode, and sub-clock mode.

■ Main Clock Mode, PLL Clock Mode, and Sub-clock Mode

○ Main clock mode

In main clock mode, a clock whose frequency is the oscillation clock frequency divided by 2 is used as the operating clock for the CPU and peripheral resources, and 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).

○ Sub-clock mode

In sub-clock mode, a clock whose frequency is the sub-clock frequency divided by 4 is used as the operating clock for the CPU and peripheral resources, and the main clock and PLL clocks are disabled.

■ Clock Mode Transition

Transition among main clock mode, PLL clock mode, and sub-clock mode is performed by writing to the MCS and SCS bits of the clock selection register (CKSCR).

○ Transition from main clock mode to PLL clock mode

When the MCS bit of the clock selection register (CKSCR) is rewritten from “1” to “0” in main clock mode, switching from the main clock to a PLL clock occurs after the PLL clock oscillation stabilization wait interval ($2^{14}/\text{HCLK}$).

○ Transition from PLL clock mode to main clock mode

When the MCS bit of the clock selection register (CKSCR) is rewritten from “0” to “1” in PLL clock mode, switching from the PLL clock to the main clock occurs when the edges of the PLL clock and the main clock coincide (after 1 to 8 PLL clocks).

○ Transition from main clock mode to sub-clock mode

When the SCS bit of the clock selection register (CKSCR) is rewritten from “1” to “0” in main clock mode, switching from the main clock to a sub-clock occurs.

○ Transition from sub-clock mode to main clock mode

When the SCS bit of the clock selection register (CKSCR) is rewritten from “0” to “1” in sub-clock mode, switching from the sub-clock to the main clock occurs after the main clock oscillation stabilization wait interval. The oscillation stabilization wait interval is selected based on the WS1 and WS0 bits of the clock selection register (CKSCR).

○ Transition from PLL clock mode to sub-clock mode

When the SCS bit of the clock selection register (CKSCR) is rewritten from “1” to “0” in PLL clock mode, switching from the PLL clock to the sub-clock occurs.

○ Transition from sub-clock mode to PLL clock mode

When the SCS bit of the clock selection register (CKSCR) is rewritten from “0” to “1” in sub-clock mode, switching from the sub-clock to a PLL clock occurs after the main clock oscillation stabilization wait interval. The oscillation stabilization wait interval is selected based on the WS1 and WS0 bits of the clock selection register (CKSCR).

Notes

Even though the MCS and SCS bits of the clock selection register (CKSCR) are rewritten, machine clock switching does not occur immediately. When operating a resource that depends on the machine clock, confirm that machine clock switching has been performed by referring to the MCM and SCM bits of the clock selection register (CKSCR) before operating the resource.

If both the SCS and MSC bits are “0”, the SCS bit takes precedence, that is, sub-clock mode is selected.

■ Selection of a PLL Clock Multiplier

Writing a value from “00_B” to “11_B” to the CS1 and CS0 of the clock selection register (CKSCR) selects a PLL clock multiplier of 1 to 4.

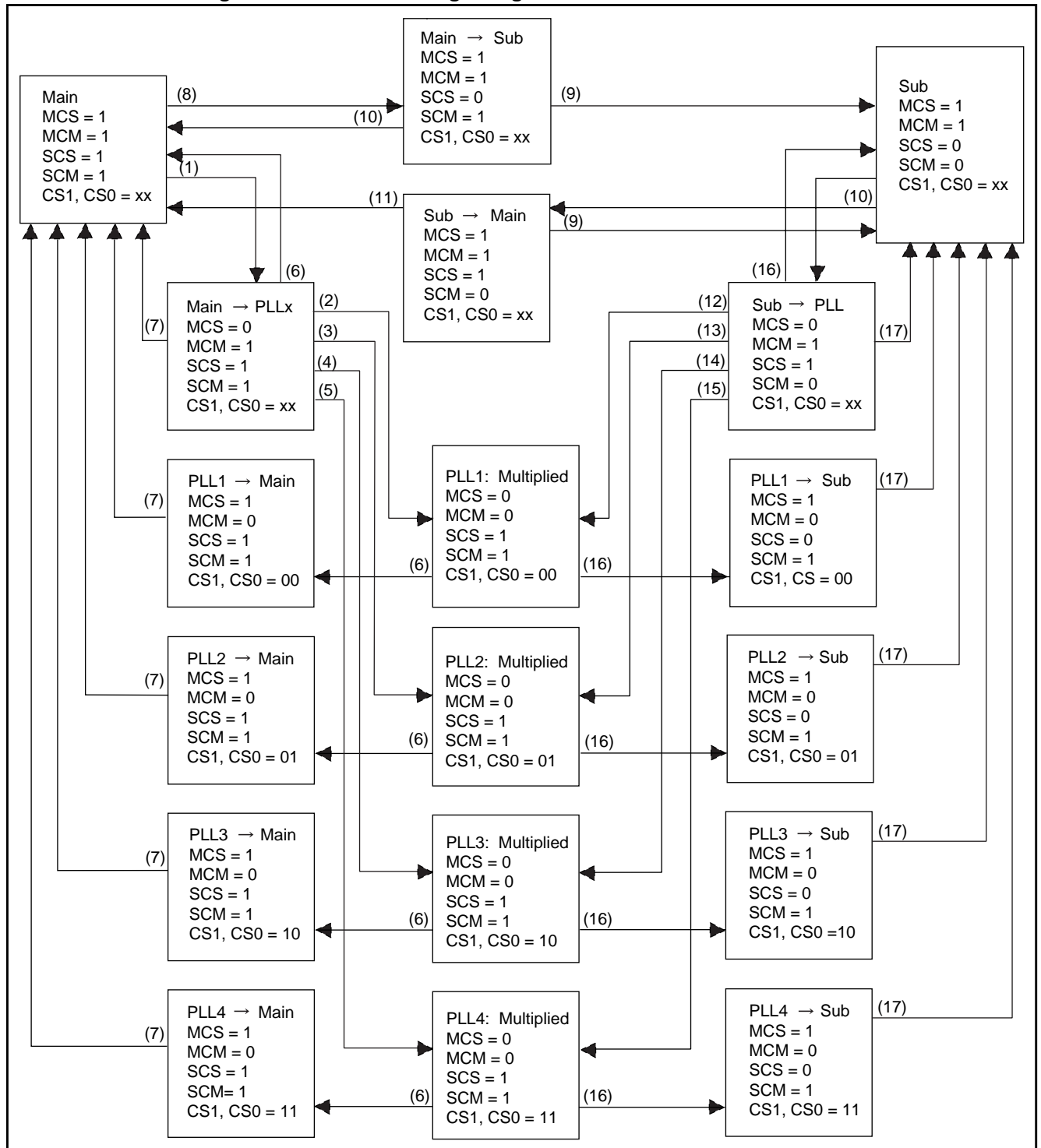
■ Machine Clock

The machine clock may be a PLL clock output from the PLL multiplier circuit, a clock whose frequency is the source oscillation frequency divided by 2, or a clock whose frequency is the sub-clock frequency divided by 4. This machine clock is supplied to the CPU and peripheral functions. The main clock, PLL clock, or sub-clock can be selected by writing to the MCS or SCS bit of the clock selection register (CKSCR).

4.4 Clock Mode

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

Figure 4.4-1 Status Change Diagram for Machine Clock Selection



- (1) Writing "0" to the MCS bit
- (2) End of PLL clock oscillation stabilization wait & CS1 and CS0 = 00
- (3) End of PLL clock oscillation stabilization wait & CS1 and CS0 = 01

CHAPTER 4 CLOCKS

- (4) End of PLL clock oscillation stabilization wait & CS1 and CS0 = 10
- (5) End of PLL clock oscillation stabilization wait & CS1 and CS0 = 11
- (6) Writing “1” to the MCS bit (including hardware standby and watchdog timer reset)
- (7) Timing of synchronization between the PLL clock and the main clock
- (8) Writing “0” to the SCS bit
- (9) Timing of synchronization between the main clock and the sub-clock.
- (10) Writing “1” to the SCS bit
- (11) End of main clock oscillation stabilization wait
- (12) End of main clock oscillation stabilization wait & CS1 and CS0 = 00
- (13) End of main clock oscillation stabilization wait & CS1 and CS0 = 01
- (14) End of main clock oscillation stabilization wait & CS1 and CS0 = 10
- (15) End of main clock oscillation stabilization wait & CS1 and CS0 = 11
- (16) Writing “0” to the SCS bit
- (17) Timing of synchronization between the PLL clock and the sub-clock

MCS	: Machine clock selection bit of the clock selection register (CKSCR)
MCM	: Machine clock indication bit of the clock selection register (CKSCR)
SCS	: Machine clock selection bit of the clock selection register (CKSCR) (sub)
SCM	: Machine clock indication bit of the clock indication register (CKSCR) (sub)
CS1, CS0	: Multiplier selection bits of the clock selection register (CKSCR)

Notes

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

If both the SCS and MCS bits are “0”, the SCS bit takes precedence, that is, the sub-clock is selected.

4.5 Oscillation Stabilization Wait Interval

When the power is turned on, when stop mode is released, or switching from the sub-clock to the main clock or a PLL clock occurs, an oscillation stabilization wait interval is required after oscillation begins because there is no oscillation. When switching from the main clock to a PLL clock occurs, an oscillation stabilization wait interval is also required after PLL oscillation starts.

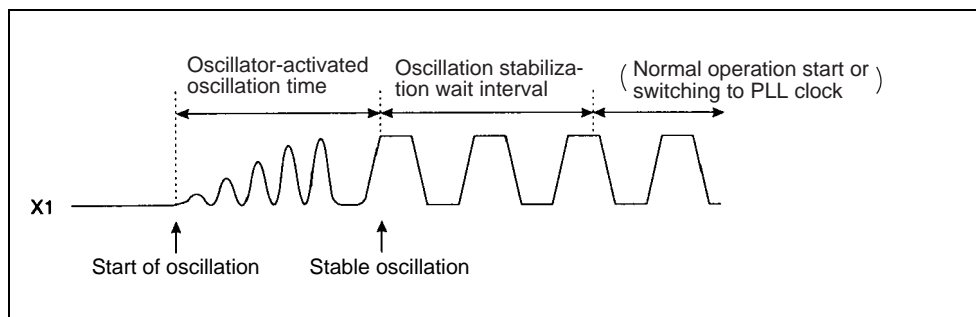
■ Oscillation stabilization wait interval

Ceramic and crystal oscillators generally require several milliseconds to 10 to 20 milliseconds to stabilize at their natural frequency (oscillation frequency) when oscillation starts. For this reason, CPU operation is not allowed immediately after oscillation starts but is allowed only after full oscillation stabilization. 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 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).

When switching from the main clock to a PLL clock occurs, the CPU continues to operate on the main clock during the PLL oscillation stabilization wait interval. After this interval, the operating clock switches to the PLL clock.

Figure 4.5-1 shows the operation immediately after oscillation starts.

Figure 4.5-1 Operation Immediately after Oscillation Starts



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

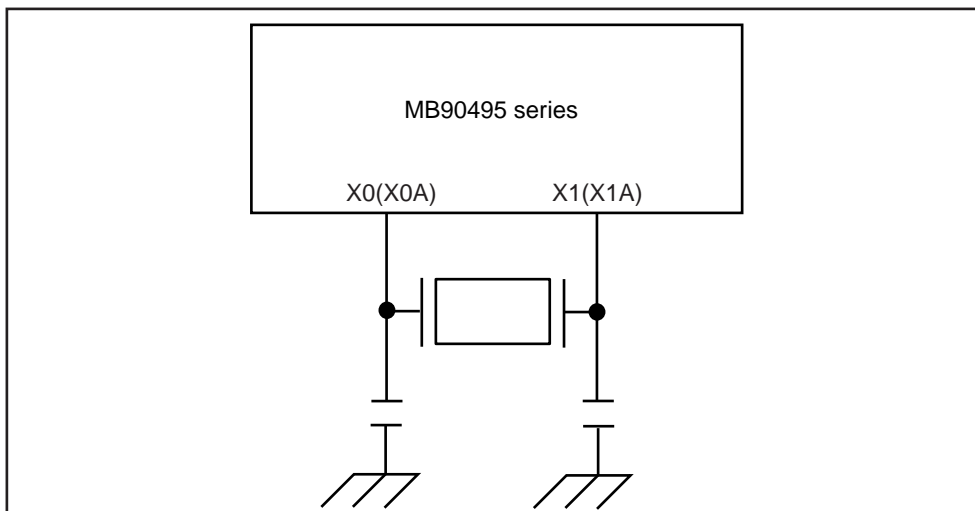
The MB90495-series microcontroller contains a system clock generation circuit. Connecting an external oscillator to this circuit generates the system clock. Alternatively, an externally generated clock can be input to the microcontroller.

■ 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

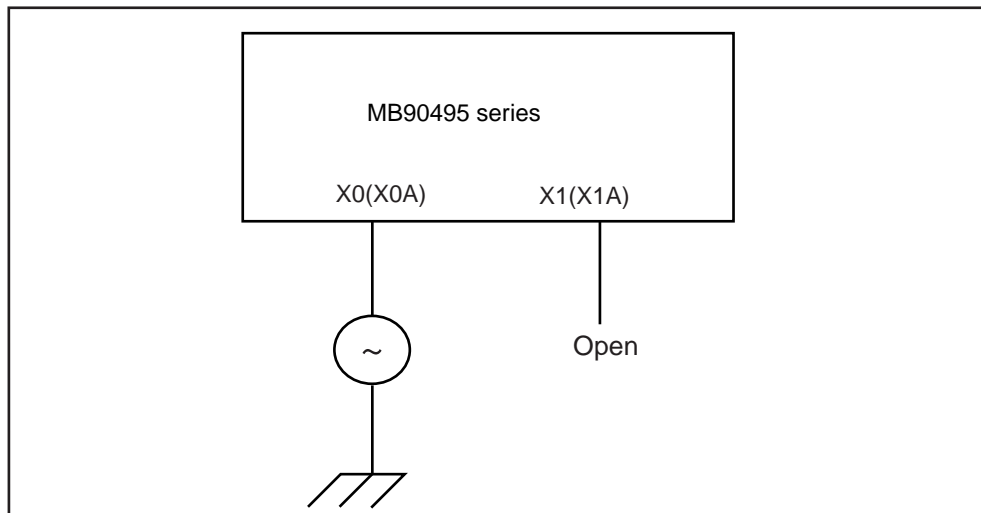


○ Example of connecting an external clock to the microcontroller

As shown in the example in Figure 4.6-2, connect an external clock to pin X0 (X0A). Pin X1 (X1A) must be open.

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

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 MB90495-series microcontrollers.

- 5.1 Overview of Low Power Consumption Mode
- 5.2 Block Diagram of the Low-Power Consumption Control Circuit
- 5.3 Low-Power Consumption Mode Control Register (LPMCR)
- 5.4 CPU Intermittent Operation Mode
- 5.5 Standby Mode
- 5.6 Status Change Diagram
- 5.7 Statuses of Pins in Standby Mode and during Hold and Reset
- 5.8 Usage Notes on Low-Power Consumption Mode

5.1 Overview of Low Power Consumption Mode

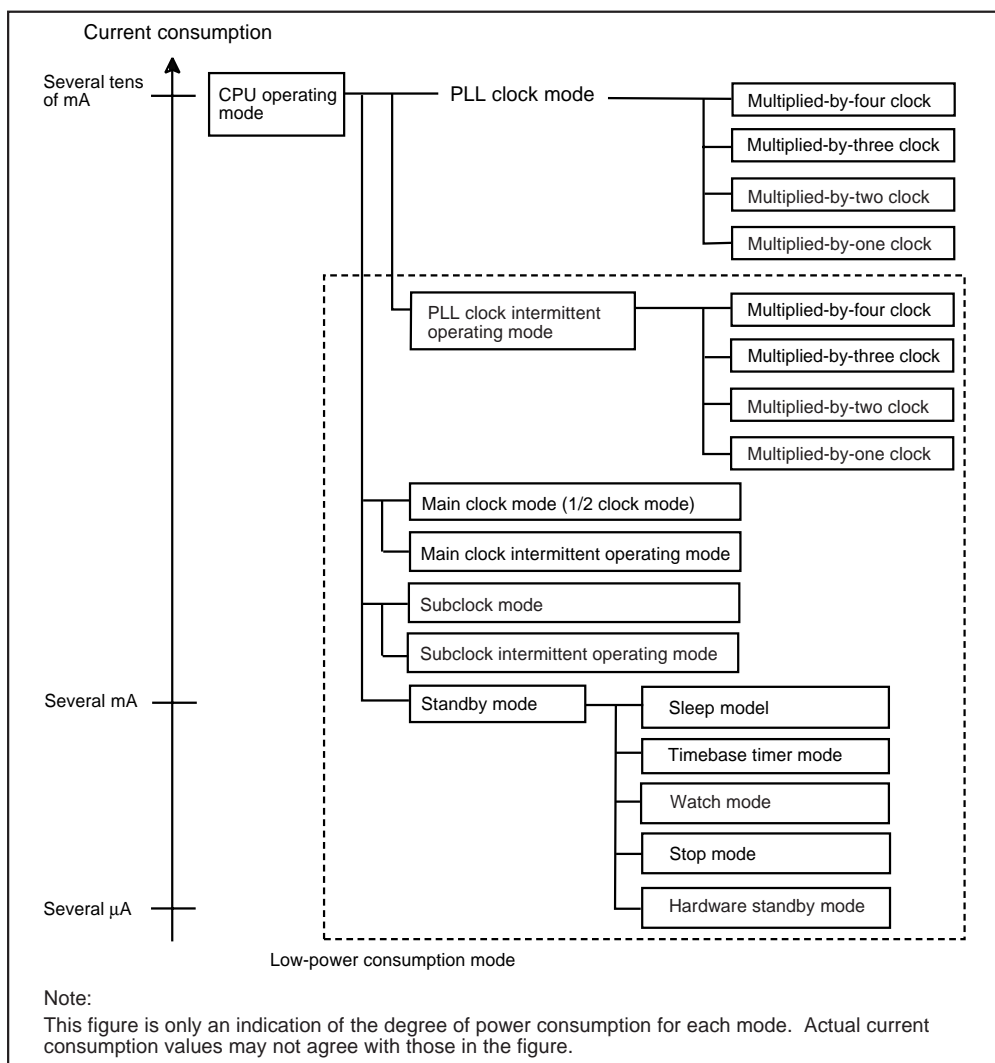
The MB90495 series has the following CPU operating modes, any of which can be used depending on operating clock selection and clock operation control:

- Clock mode (PLL clock mode, main clock mode, or sub-clock mode)
- CPU intermittent operating mode (PLL clock intermittent operating mode, main clock intermittent operating mode, or sub-clock intermittent operating mode)
- Standby mode (sleep mode, time-base timer mode, stop mode, or watch mode)

■ CPU Operating Modes and Current Consumption

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

Figure 5.1-1 CPU Operating Mode and Current Consumption



5.1 Overview of Low Power Consumption Mode

■ Clock Mode

○ PLL clock mode

In this mode, a PLL clock that is a multiple of the oscillation clock (HCLK) is used to operate the CPU and peripheral functions.

○ Main clock mode

In this mode, the main clock, with the oscillation clock (HCLK) frequency divided by 2 is used to operate the CPU and peripheral functions. In the main clock mode, the PLL multiplier circuit is inactive.

○ Sub-clock mode

In this mode, the sub-clock, with the sub-clock (SCLK) frequency divided by 4 is used to operate the CPU and peripheral functions. In the sub-clock mode, the main clock and PLL multiplier circuit are inactive.

Reference

For the clock mode, see Section 4.4 "Clock Mode".

■ CPU Intermittent Operating Mode

In this mode, the CPU is operated intermittently while high-speed clock pulses are supplied to peripheral functions, thereby reducing power consumption. In this mode, intermittent clock pulses are supplied only to the CPU while it is accessing a register, internal memory, peripheral function, or external unit.

■ Standby Mode

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

○ PLL sleep mode

The PLL sleep mode is activated to stop the CPU operating clock in the PLL clock mode. Components excluding the CPU operate on the PLL clock.

○ Main sleep mode

The main sleep mode is activated to stop the CPU operating clock in the main clock mode. Components excluding the CPU operate on the main clock.

○ Sub-sleep mode

The subsleep mode activated to stop the CPU operating clock in the subclock mode. Components excluding the CPU operate on the divided-by-four subclock.

○ Timebase timer mode

The timebase timer mode causes the operation of functions, excluding the oscillation clock, timebase timer, and clock timer, to stop. All functions other than the timebase timer and clock timer are inactivated.

CHAPTER 5 LOW POWER CONSUMPTION MODE

○ **Watch mode**

The watch mode operates the clock timer only. The subclock operates and the main clock and PLL multiplier circuit stop.

○ **Stop mode**

The stop mode cause the oscillation to stop. All functions are inactivated.

Note

Because the stop mode turns the oscillation clock off, data can be retained by the lowest power consumption.

5.2 Block Diagram of the Low-Power Consumption Control Circuit

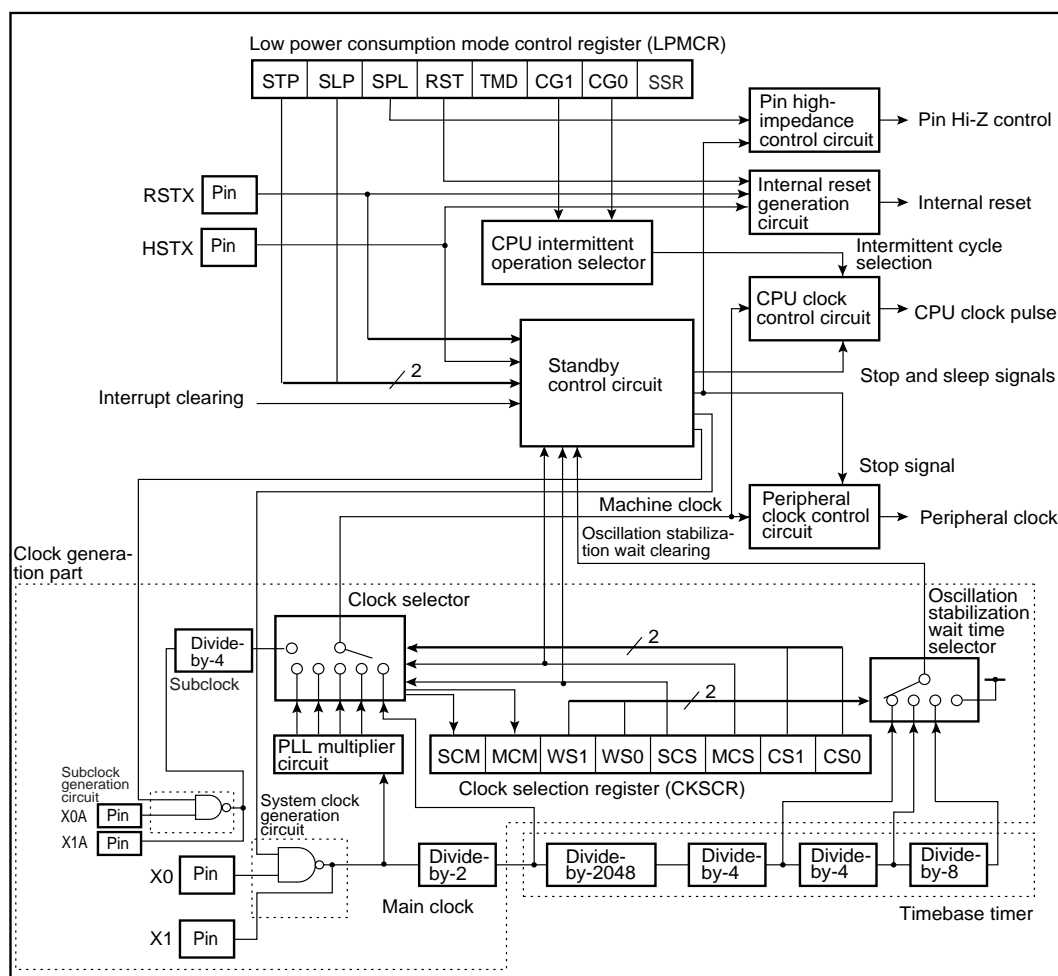
The low-power consumption control circuit consists of the following seven blocks:

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

■ Block Diagram of the Low-Power Consumption Control Circuit

Figure 5.2-1 shows a 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 to halt the CPU during the CPU intermittent operation mode.

- **Standby control circuit**

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

- **CPU clock control circuit**

This circuit controls clocks supplied to the CPU. This circuit controls clocks supplied to peripheral functions for the peripheral clock control circuit.

- **Peripheral clock control circuit**

This circuit controls clocks supplied to peripheral functions.

- **Pin high-impedance control circuit**

This circuit makes external pins high-impedance in the timebase timer mode and stop mode. For pins with the pull-up option, this circuit disconnects the pull-up resistor in the stop 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 release the standby mode and to set the CPU intermittent operation function.

5.3 Low-Power Consumption Mode Control Register (LPMCR)

This register switches to or releases the low-power consumption mode. This register also sets the number of CPU clock pulses to halt during the CPU intermittent operation 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)

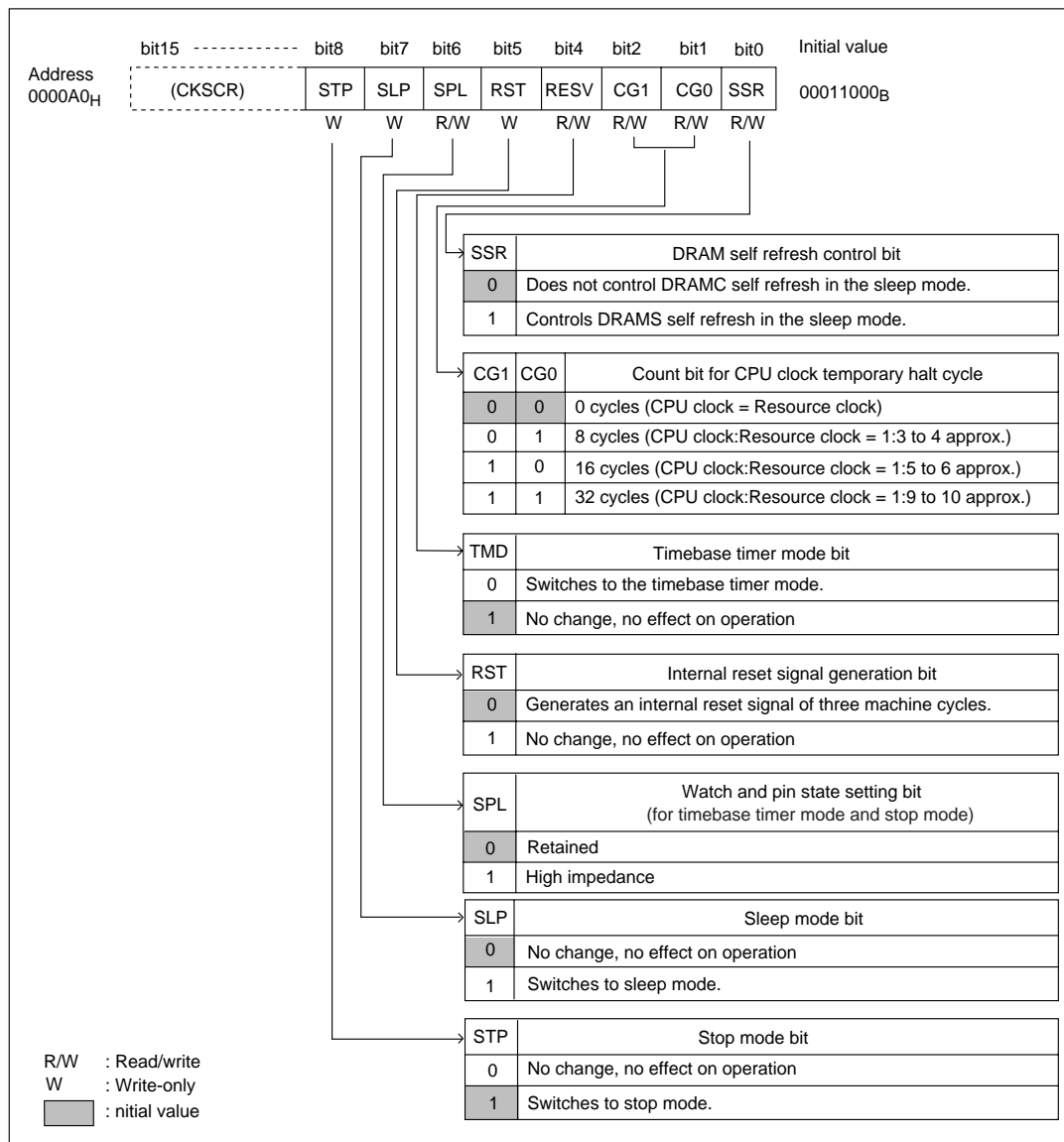


Table 5.3-1 Function Description of Each Bit of the Low-power Consumption Mode Control Register (LPMCR)

Bit name		Function
bit 7	STP: Stop mode bit	<ul style="list-style-type: none"> This bit indicates switching to the stop mode. When 1 is written to this bit, a switch to the stop mode is performed. Writing 0 in this bit has no effect on operation. This bit is cleared to 0 by a reset or when an interrupt request occurs. The read value of this bit is always 0.
bit 6	SLP: Sleep mode bit	<ul style="list-style-type: none"> This bit indicates switching to a sleep mode. When 1 is written to this bit, a switch to a sleep mode is performed. Writing 0 in this bit has no effect on operation. This bit is cleared to 0 by a reset or when an interrupt request occurs. The read value of this bit is always 0.
bit 5	SPL: Pin state setting bit (for watch mode, timebase timer mode, and stop mode)	<ul style="list-style-type: none"> This bit is enabled only in the watch mode, timebase timer mode, and stop mode. 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	<ul style="list-style-type: none"> When 0 is written to this bit, an internal reset signal of three machine cycles is generated. Writing 1 in this bit has no effect on operation. The read value of this bit is always 1.
bit 3	TMD: Watch/timebase timer mode bit	<ul style="list-style-type: none"> This bit indicates switching to the watch mode or timebase timer mode. When 0 is written to this bit in the main clock mode or PLL clock mode, a switch to timebase timer mode is performed. When 0 is written to this bit in the subclock mode, a switch to the watch mode is performed. This bit is cleared to 1 by a reset or when an interrupt request occurs. The read value of this bit is always 1.
bit 2 bit 1	CG1, CG0: Bits for selecting clock count for CPU temporary halt cycle	<ul style="list-style-type: none"> These bits set the number of CPU clock pulses per cycle to halt the CPU for the CPU intermittent operation function. The clock supplied to the CPU is stopped for the specified number of pulses after the execution of each instruction. Four types of clock counts are selectable. These bits are initialized to 00_B by a reset.
bit 0	SSR: DRAM self refresh control bit	<ul style="list-style-type: none"> This bit specifies whether DRAMC self refresh is controlled in the sleep mode. When 1 is written to this bit, DRAMC self refresh is controlled in the sleep mode. This bit is initialized to 0 by a reset.

5.3 Low-Power Consumption Mode Control Register (LPMCR)

■ Access to the Low-Power Consumption Mode Control Register

Switching to a low-power consumption mode (including the stop mode, sleep mode, timebase timer mode, and watch mode) is performed by writing the low-power consumption mode control register. Only the instructions listed in Table 5.3-2 should be used for this purpose. If other instructions are used for switching to a low-power consumption mode, operation cannot be assured. To control functions not listed in Table 5.3-1, any instruction can be used.

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

■ Priorities of the STP, SLP, and TMD Bits

If the stop mode, sleep mode, and timebase timer mode are requested concurrently, the stop mode request, timebase timer mode request, and sleep mode request are given priorities in this order for processing.

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

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

5.4 CPU Intermittent Operation Mode

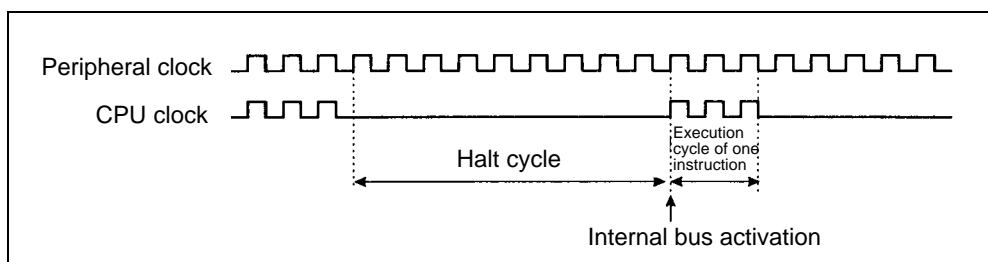
This mode is used for intermittent operation of the CPU while external buses and peripheral functions continue to operate at high speeds. The purpose of this mode is to reduce power consumption.

■ CPU Intermittent Operation Mode

This mode halts the supply of the clock pulse 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, or the external bus. Internal bus cycle activation is therefore delayed. While high-speed peripheral clock pulses are supplied to peripheral functions, the execution speed of the CPU is reduced, thereby enabling low-power consumption processing.

- The low-power consumption mode control register (LPMCR: CG1 and CG0) is used to select the number of clock pulses per halt cycle of the clock supplied to the CPU.
- External bus operation uses the same clock as that used for peripheral functions.
- Instruction execution time in the CPU intermittent mode can be calculated. A correction value should be obtained by multiplying the execution count of instructions that access a register, internal memory, internal peripheral functions, or the external bus by the number of clock pulses per halt cycle. Add this corrective value to the normal execution time. Figure 5.4-1 shows the operating clock pulses during the CPU intermittent operation mode.

Figure 5.4-1 Clock Pulses during the CPU Intermittent Operation



5.5 Standby Mode

The standby mode includes the sleep (PLL sleep, main sleep, subsleep), watch, and stop modes.

■ Operation Status during Standby Mode

Table 5.5-1 shows operation statuses during standby mode.

Table 5.5-1 Operation Statuses during Standby Mode

Standby mode		Condition for switch	Main clock	Sub-clock	Machine clock	CPU	Peripheral	Pin	Release event		
Sleep mode	PLL sleep mode	SCS=1 MCS=0 SLP=1	Active	Active	Active	Inactive	Active	Active	Reset Interrupt		
	Main sleep mode	SCS=1 MCS=1 SLP=1									
	Subsleep mode	SCS=0 SLP=1	Inactive								
Timebase timer mode	Timebase timer mode (SPL=0)	SCS=1 TMD=0	Active		Inactive		Inactive (*1)	Retained			
	Timebase timer mode (SPL=1)	SCS=1 TMD=0						Hi-z			
Watch mode	Watch mode (SPL=0)	SCS=0 TMD=0	Inactive					Inactive			Inactive (*2)
	Watch mode (SPL=1)	SCS=0 TMD=0					Hi-z				
Stop mode	Stop mode (SPL=0)	STP=1	Inactive	Inactive			Inactive				Retained
	Stop mode (SPL=1)	STP=1						Hi-z			

*1 : The timebase timer and clock timer operate.

*2 : The clock timer operates.

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

SLP : Sleep mode bit of low-power consumption mode control register (LPMCR)

STP : Stop mode bit of low-power consumption mode control register (LPMCR)

TMD : Watch and timebase timer mode bit of low-power consumption mode control register (LPMCR)

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

SCS : Machine clock selection bit (sub) of clock selection register (CKSCR)

Hi-z : High-impedance

5.6 Sleep Mode

This mode causes the CPU operating clock to stop while other components continue to operate. When the low-power consumption mode control register (LPMCR) indicates a switch to a sleep mode, a switch to the PLL sleep mode occurs if the PLL clock mode has been set. A switch to the main clock mode occurs if the main clock mode has been set. A switch to the subsleep mode occurs if the subclock mode has been set.

■ Switching to Sleep Mode

Writing 1 in the SLP bit and the TMD bit and 0 in the STP bit of the low-power consumption mode control register (LPMCR) triggers a switch to a sleep mode. At this time, if the MSC bit is 0 and the SCS bit is 1 in the clock selection register (CKSCR), a switch to the PLL sleep mode is triggered. If the MSC bit and the SCS bit are 1, a switch to the main sleep mode is triggered. If the SCS bit is 0, a switch to the subsleep mode is triggered.

Note

When 1 is written to the SLP and STP bits at the same time, the STP bit setting overrides the SLP bit setting and the mode switches to the stop mode. When 1 is written to the SLP bit and 0 is written to the TMD bit at the same time, the TMD bit setting overrides the SLP bit setting and the mode switches to the timebase timer mode or watch mode.

○ Data retention function

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

○ Hold function

During a sleep mode, the external bus hold function is active. This function sets the hold status when requested to do so.

○ Operation during an interrupt request

Writing 1 in the SLP bit of the low-power consumption mode control register during an interrupt request does not trigger a switch to a 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 a sleep mode, all pins (excluding those used for bus I/O or bus control) retain their previous status.

■ Release of Sleep Mode

The low-power consumption control circuit releases sleep modes when a reset is input or an interrupt occurs.

○ Return by a reset

A sleep mode is initialized to the main clock mode by a reset.

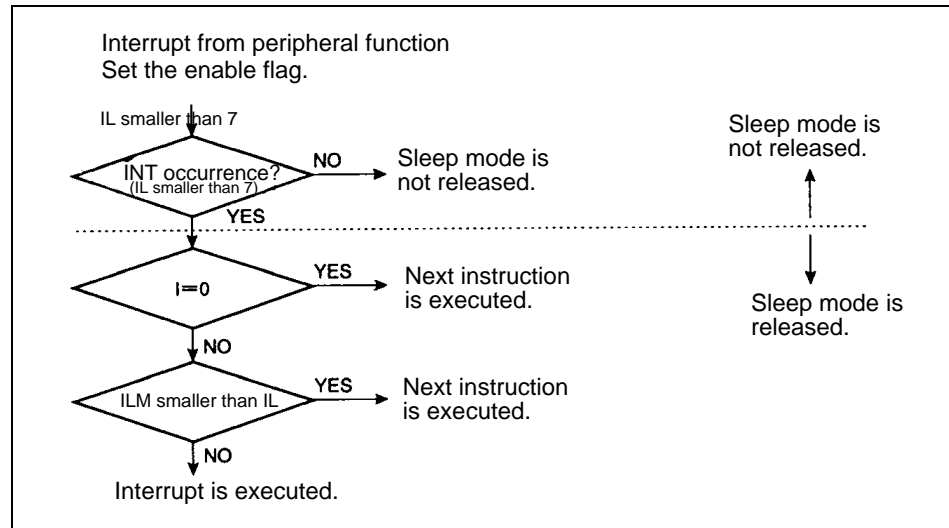
5.6 Sleep Mode

○ Return by an interrupt

If an interrupt request of level seven or higher is issued from a peripheral circuit during a sleep mode, the sleep mode is released. After the mode is released, the interrupt is handled as an ordinary interrupt. If the interrupt is accepted according to the setting of the I flag of the condition code register (CCR), interrupt level mask register (ILM), and interrupt control register (ICR), the CPU executes the interrupt processing. If the interrupt is not accepted, the CPU executes the instruction following the instruction specifying the sleep mode.

Figure 5.6-1 shows the release of a sleep mode when an interrupt occurs.

Figure 5.6-1 Release of Sleep Mode by Interrupt Occurrence



Note

When interrupt processing is executed, the CPU normally executes the instruction that follows the instruction in which switching to a sleep mode has been specified. The CPU then proceeds to interrupt processing. If the switching to sleep mode and acceptance of an external bus hold request occur at the same time, however, the CPU may proceed to interrupt processing before executing the next instruction.

5.7 Timebase Timer Mode

This mode causes all functions, excluding oscillation, the timebase timer, and the clock timer, to stop. In this mode, only the timebase timer and clock timer operate.

■ Switching to the Timebase Timer Mode

When 0 is written to the TMD bit of the low-power consumption mode control register (LPMCR) in the PLL clock mode or main clock mode (CKSCR: SCS = 1), switching to the timebase timer mode occurs.

○ Data retention function

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

○ Hold function

During the timebase timer mode, because the external bus hold function is inactive, a hold request is not accepted even if it is input. If a hold request is input during switching to the timebase timer mode, the bus may remain at high-impedance and prevent the HAK signal from going low.

○ Operation during an interrupt request

Writing 0 in the TMD bit of the low-power consumption mode control register (LPMCR) during an interrupt request does not trigger a switch to the timebase timer mode.

○ Status of pins

Whether the external pins in the timebase timer mode retain the state they had immediately before switching to the timebase timer mode or go to the high-impedance state can be controlled by the low-power consumption mode control register (LPMCR: SPL).

■ Release of Timebase Timer Mode

The low-power consumption control circuit releases the timebase timer mode when a reset is input or an interrupt occurs.

○ Return by a reset

The timebase timer mode is initialized to the main clock mode by a reset.

○ Return by an interrupt

If an interrupt request of level seven or higher is issued from a peripheral circuit during the timebase timer mode (IL2, IL1, and IL0 of the interrupt control register (ICR) do not indicate 111_B), the low-power consumption mode control circuit releases the timebase timer mode. After the mode is released, the interrupt is handled as an ordinary interrupt. If the interrupt is accepted according to the setting of the I flag of the condition code register (CCR), interrupt level mask register (ILM), or interrupt control register (ICR), the CPU executes the interrupt processing. If the interrupt is not accepted, the CPU executes the instruction following the instruction specifying the timebase timer mode.

5.7 Timebase Timer Mode

Note

When interrupt processing is executed, the CPU normally executes the instruction following the instruction in which switching to the timebase timer mode has been specified. The CPU then proceeds to interrupt processing. If the switching to the timebase timer mode and acceptance of an external bus hold request occur at the same time, however, the CPU may proceed to interrupt processing before executing the next instruction.

5.8 Watch Mode

This mode causes all functions, excluding the subclock and clock timer, to stop. In this mode almost all chip functions stop.

■ Switching to the Timebase Timer Mode

When 0 is written to the TMD bit of the low-power consumption mode control register (LPMCR) in the subclock watch mode (CKSCR:SCS = 0), switching to the watch mode occurs.

○ Data retention function

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

○ Hold function

During the watch mode, because the external bus hold function is inactive, a hold request is not accepted even if it is input. If a hold request is input during switching to the watch mode, the bus may remain at high-impedance and prevent the HAK signal from going low.

○ Operation during an interrupt request

Writing 1 in the TMD bit of the low-power consumption mode control register (LPMCR) during an interrupt request does not trigger a switch to the watch mode.

○ Status of pins

Whether the external pins in the watch mode retain the state they had immediately before switching to the watch mode or go to the high-impedance state can be controlled by the SPL bit of the low-power consumption mode control register (LPMCR).

■ Release of Watch Mode

The low-power consumption control circuit releases the watch mode when a reset is input or an interrupt occurs.

○ Return by a reset

The watch mode is released by a reset to the oscillation stabilization wait reset state. The reset sequence is executed after the oscillation stabilization wait time.

○ Return by an interrupt

If an interrupt request of level seven or higher is issued from a peripheral circuit during the watch mode (IL2, IL1, and IL0 of the interrupt control register (ICR) do not indicate 111_B), the low-power consumption mode control circuit releases the watch mode and causes switching to the subclock mode immediately. After the switching, the interrupt is handled as an ordinary interrupt. If the interrupt is accepted according to the setting of the I flag of the condition code register (CCR), interrupt level mask register (ILM), and interrupt control register (ICR), the CPU executes the interrupt processing. If the interrupt is not accepted, the CPU executes the instruction following the instruction specifying the watch mode.

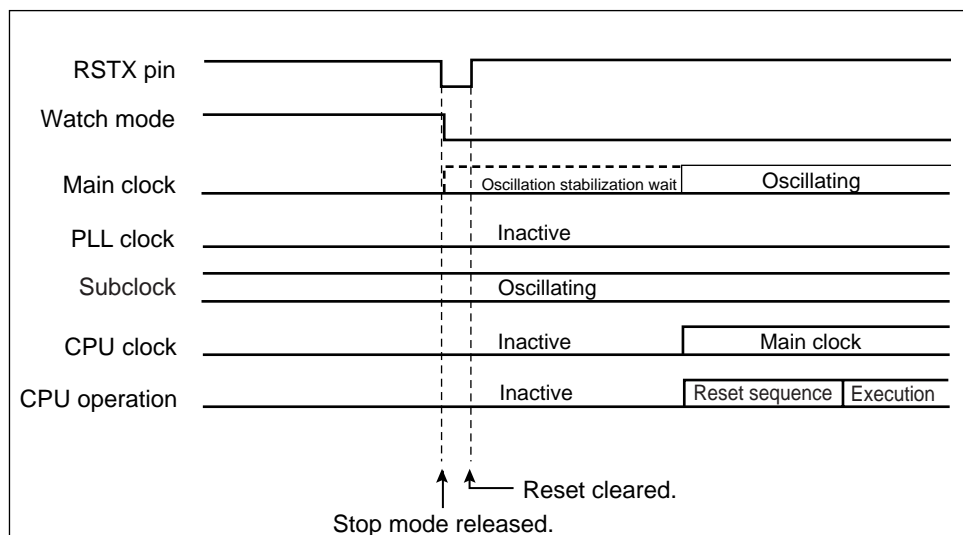
5.8 Watch Mode

Note

When interrupt processing is executed, the CPU normally executes the instruction following the instruction in which switching to the watch mode has been specified. The CPU then proceeds to interrupt processing. If the switching to the watch mode and acceptance of an external bus hold request occur at the same time, however, the CPU may proceed to interrupt processing before executing the next instruction.

Figure 5.8-1 shows a return from the watch mode.

Figure 5.8-1 Release of the Watch Mode (External Reset)



5.9 Stop Mode

Because this mode causes oscillation to stop and inactivates all functions, data can be retained by the lowest power consumption.

■ Switching to the Stop Mode

When 1 is written to the STP bit of the low-power consumption mode control register (LPMCR), switching to the stop mode occurs.

○ Data retention function

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

○ Hold function

During the stop mode, because the external bus hold function is inactive, a hold request is not accepted even if it is input. If a hold request is input during switching to the stop mode, the bus may remain at high-impedance and prevent the HAK signal from going low.

○ Operation during an interrupt request

Writing 1 in the STP bit of the low-power consumption mode control register (LPMCR) does not trigger a switch to the stop mode.

○ Status of pins

Whether the external pins in the stop mode retain the state they had immediately before switching to the stop mode or go to the high-impedance state can be controlled by the SPL bit of the low-power consumption mode control register (LPMCR). It is recommended, that SPL is set to 1. If SPL=0, assure that Port 4 and 6 have defined levels.

■ Release of Stop Mode

The low-power consumption control circuit releases the stop mode when a reset is input or an interrupt occurs. Because oscillation of the operating clock is halted before returning from the stop mode, the low-power consumption control circuit enters the oscillation stabilization wait state, then releases the stop mode.

○ Return by a reset

After the stop mode is released by a reset, the oscillation stabilization wait state is set. The reset sequence is executed after the oscillation stabilization wait time.

○ Return by an interrupt

If an interrupt request of level seven or higher is issued from a peripheral circuit during the stop mode (IL2, IL1, and IL0 of the interrupt control register (ICR) do not indicate 111_B), the low-power consumption mode control circuit releases the stop mode. The interrupt is then handled as an ordinary interrupt after the oscillation stabilization wait time of the main clock specified by the WS1 and WS0 bits of the clock selection register (CKSCR). If the interrupt is accepted according to the setting of the I flag of the condition code register (CCR), interrupt level mask register (ILM), and interrupt control register (ICR), the CPU executes the interrupt processing. If the interrupt is not accepted, the CPU executes the instruction following the instruction

5.9 Stop Mode

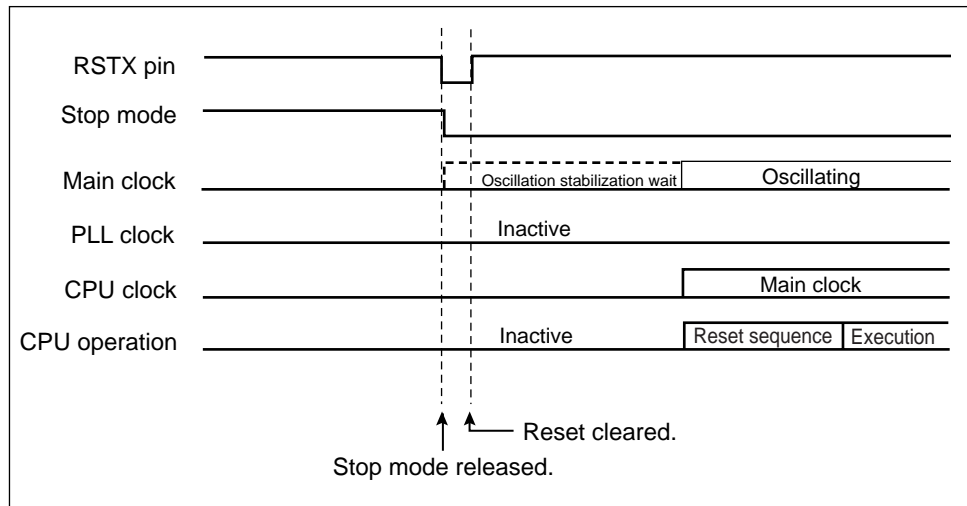
specifying the stop mode.

Note

When interrupt processing is executed, the CPU normally executes the instruction following the instruction in which switching to the stop mode has been specified. The CPU then proceeds to interrupt processing. If the switching to the stop mode and acceptance of an external bus hold request occur at the same time, however, the CPU may proceed to interrupt processing before executing the next instruction.

Figure 5.9-1 shows a return from the stop mode.

Figure 5.9-1 Release of the Stop Mode (External Reset)

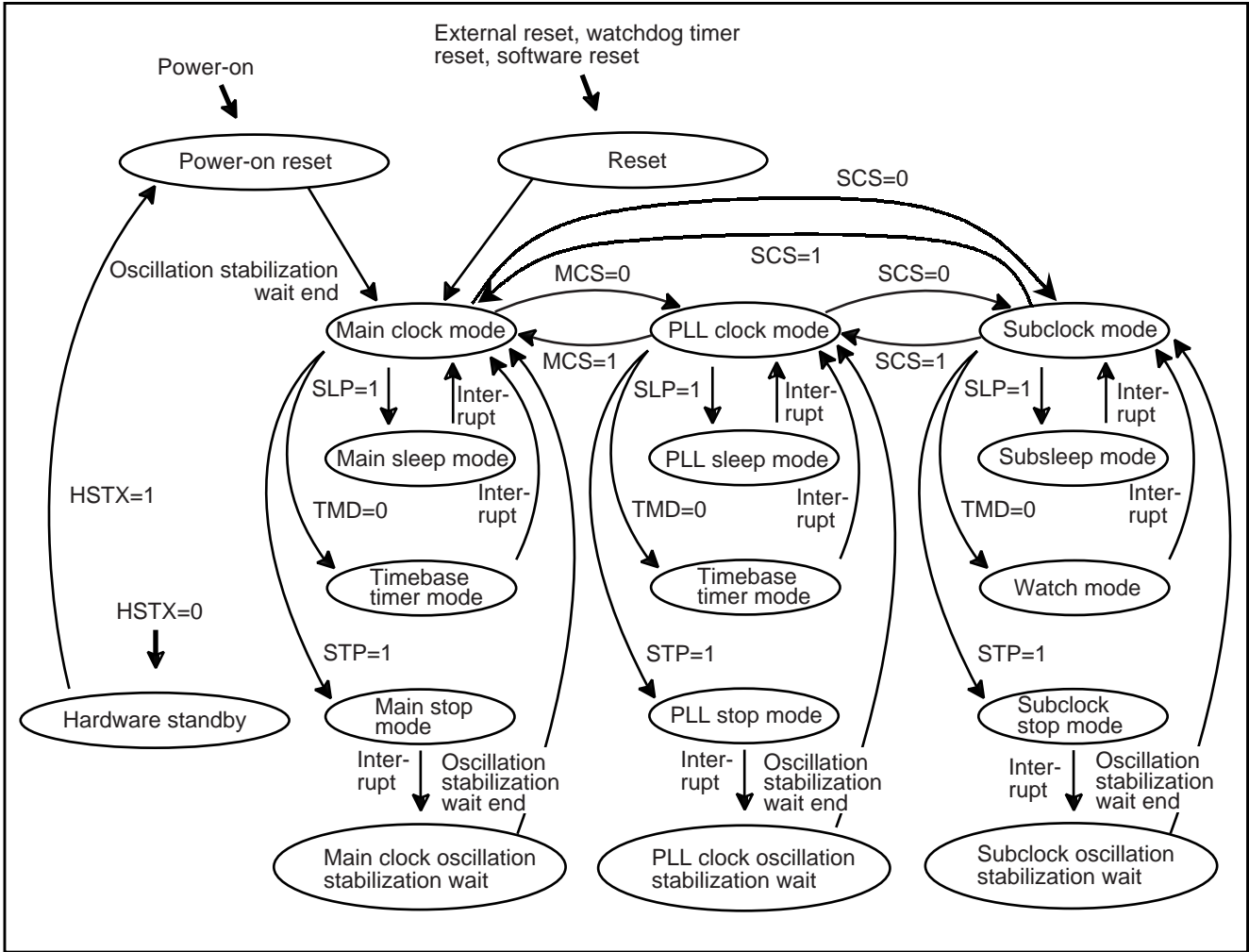


5.10 Status Change Diagram

Figure 5.6-1 shows the status change diagram of the MB90495 series.

■ Status Change Diagram

Figure 5.10-1 Status Change Diagram



5.10 Status Change Diagram

■ Operation Status in Each Operating Mode

Table 5.10-1 lists the operation status in each operating mode.

Table 5.10-1 Operation Status in Each Operating Mode

Operation status	Main clock	Subclock	PLL clock	CPU	Peripheral	Watch	Timebase timer	Clock source
PLL	Active	Active	Active	Active	Active	Active	Active	PLL clock
PLL sleep				Inactive				
Timebase timer (*1)								
PLL stop	Inactive	Inactive	Inactive		Inactive	Inactive	Inactive	
PLL oscillation stabilization wait	Active	Active	Active	Active		Active		
Main	Active	Active	Inactive	Active	Active	Active	Active	Main clock
Main sleep				Inactive				
Timebase timer (*2)								
Main stop	Inactive	Inactive			Inactive	Inactive	Inactive	
Main oscillation stabilization wait	Active	Active		Active		Active	Active	
Sub	Inactive	Active	Inactive	Active	Active	Active	Inactive	Sub-clock
Subsleep				Inactive				
Watch		Inactive			Inactive			
Subclock stop						Inactive		
Subclock oscillation stabilization wait				Active		Active		
Power-on reset	Active	Active	Inactive	Inactive	Inactive	Active	Active	Main clock
Reset			Active					

*1 During the PLL clock mode

*2 During the main clock mode

5.11 Statuses of Pins in Standby Mode and during Hold and Reset

The status of pins in the standby mode and during hold and reset are described for each memory access mode.

■ Pull-up Register Controlled by Software

(There are no software controlled pull-up resistors in M90495)

5.12 Usage Notes on Low-Power Consumption Mode

Note the following four items when using the low-power consumption mode:

- Switching to a standby mode and interrupt
 - Release of a standby mode by an interrupt
 - Release of the stop mode
 - Oscillation stabilization wait time
-

■ Switching to a Standby Mode and Interrupt

During an interrupt request to the CPU from a peripheral function, the CPU ignores the setting of the low-power consumption mode control register (LPMCR) even if 1 is written to the STP and SLP bits or if 0 is written to the TMD bit. Thus, switching to each standby mode is disabled (even after processing of the interrupt is completed, there is no switch to a standby mode). If the interrupt level is seven 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 a standby mode can be performed.

○ Release of the Standby Mode by an Interrupt

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

After the release of the standby mode by an interrupt, normal 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 following the instruction in which switching to the standby mode has been specified.

When interrupt processing is executed normally, the CPU first executes the instruction following the instruction in which switching to the standby mode has been specified. The CPU then proceeds to interrupt processing.

Depending on the condition when switching to a standby mode was performed, however, the CPU may proceed to interrupt processing before executing the next instruction.

Note

If the CPU does not branch to the interrupt processing routine immediately after a return, action such as interrupt disabling must be taken before a standby mode is set.

○ Release of the Stop Mode

The stop mode can be released by an input that has been set as an external interrupt input cause before the system enters the stop mode. As an input cause, an H-level signal, L-level signal, rising edge, or falling edge can be selected.

■ Oscillation Stabilization Wait Time

○ Clock oscillation stabilization wait time

Because the oscillator for oscillation is halted in the stop mode, an oscillation stabilization wait time is required. A time period selected by the WS1 and WS0 bits of the clock selection register (CKSCR) is used as the oscillation stabilization wait time. The WS1 and WS0 bits can be set to 00_B only in the main clock mode.

○ PLL clock oscillation stabilization wait time

If a mode in which the CPU operates on the main clock and the PLL clock stops is switched to a mode in which the CPU or peripheral circuits operate on the PLL clock, switching to the PLL clock, oscillation stabilization wait state occurs. In this state, the CPU operates on the main clock.

The PLL clock oscillation stabilization time is fixed at $2^{14}/\text{HCLK}$ (HCLK: Oscillation clock).

CHAPTER 6 I/O PORTS

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

- 6.1 Overview of I/O Ports
- 6.2 Register and External Dual-Function Pin Assignment
- 6.3 Port 0
- 6.4 Port 1
- 6.5 Port 2
- 6.6 Port 3
- 6.7 Port 4
- 6.8 Port 5
- 6.9 Port 6
- 6.10 Sample I/O Port Program

6.1 Overview of I/O Ports

An I/O port can be used as a general-purpose I/O port (parallel I/O port). The MB90495 series has 7 ports (49 lines). The ports are also used for resource 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 designate the direction of a data flow (input or output) at the I/O pins in bit units using the port data direction register (DDR). The function of each port and the resources using it are described below:

- Port 0 : General-purpose I/O port/external bus
- Port 1 : General-purpose I/O port/resource (PPG, Input Capture)/external bus
- Port 2 : General-purpose I/O port/resource (Reload timer, External Interrupt)/external bus
- Port 3 : General-purpose I/O port/resource (UART 0, Free Running Timer, ADC Trigger)/external bus
- Port 4 : General-purpose I/O port/resource (UART 1, CAN)
- Port 5 : General-purpose I/O port/resource (analogue input pins)
- Port 6 : General-purpose I/O port/resource (External Interrupt)

Table 6.1-1 summarizes the functions of individual ports.

Table 6.1-1 Functions of individual ports

Port	Pin	Input form	Output form	Function	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0		
Port 0	P00/AD00 ~P07/AD07	CMOS (hysteresis)	CMOS	I/O port	—	—	—	—	—	—	—	—	P07	P06	P05	P04	P03	P02	P01	P00		
				Resource	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
Port 1	P10/IN0/AD00 ~P17/PPG3/AD15			I/O port	P17	P16	P15	P14	P13	P12	P11	P10	—	—	—	—	—	—	—	—	—	
				Resource	PPG3	PPG2	PPG1	PPG0	IN3	IN2	IN1	IN0	—	—	—	—	—	—	—	—	—	
Port 2	P20/TIN0/A16 ~P27/INT7/A23		CMOS	I/O port	—	—	—	—	—	—	—	—	P27	P26	P25	P24	P23	P22	P21	P20		
					Resource	—	—	—	—	—	—	—	—	INT7	INT6	INT5	INT4	TOUT1	TIN1	TOUT0	TIN0	
Port 3	P30/ALE ~P37/ADTG/CLK				I/O port	P37	P36	P35	P34	P33	P32	P31	P30	—	—	—	—	—	—	—	—	
					Resource	ADTG	FRCK				SIN0	SCK0	SOT0	—	—	—	—	—	—	—	—	
Port 4	P40/SIN1 ~P44/RX				I/O port	—	—	—	—	—	—	—	—	—	—			P44	P43	P42	P41	P40
					Resource	—	—	—	—	—	—	—	—	—	—			RX	TX	SOT1	SCK1	SIN1
Port 5	P50/AN0 ~P57/AN7	Analog/ CMOS (hysteresis)	CMOS	I/O port	P57	P56	P55	P54	P53	P52	P51	P50	—	—	—	—	—	—	—	—		
		Analogue input		AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	—	—	—	—	—	—	—	—	—		
Port 6	P60/INT0 ~P63/INT3	CMOS (hysteresis)	CMOS	I/O port	—	—	—	—	—	—	—	—	—	—	—	—	P63	P62	P61	P60		
		Resource		—	—	—	—	—	—	—	—	—	—	—	—	—	INT3	INT2	INT1	INT0		

<Check>

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

6.2 Register and External Dual-Function Pin Assignment

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 bus pins.

■ Registers for I/O Ports

Table 6.2-1 is a list of the registers corresponding to individual ports.

Table 6.2-1 Registers and corresponding ports

Register	Read/Write	Address	Initial value
Port 0 data register (PDR0)	R/W	000000H	XXXXXXXX _B
Port 1 data register (PDR1)	R/W	000001H	XXXXXXXX _B
Port 2 data register (PDR2)	R/W	000002H	XXXXXXXX _B
Port 3 data register (PDR3)	R/W	000003H	XXXXXXXX _B
Port 4 data register (PDR4)	R/W	000004H	----XXXX _B
Port 5 data register (PDR5)	R/W	000005H	XXXXXXXX _B
Port 6 data register (PDR6)	R/W	000006H	----XXXX _B
Port 0 data direction register (DDR0)	R/W	000010H	00000000 _B
Port 1 data direction register (DDR1)	R/W	000011H	00000000 _B
Port 2 data direction register (DDR2)	R/W	000012H	11111111 _B
Port 3 data direction register (DDR3)	R/W	000013H	00000000 _B
Port 4 data direction register (DDR4)	R/W	000014H	----0000 _B
Port 5 data direction register (DDR5)	R/W	000015H	00000000 _B
Port 6 data direction register (DDR6)	R/W	000016H	----0000 _B
Analog data input enable register (ADER)	R/W	00001BH	11111111 _B

R/W:Read/write enabled

R:Read only

X:Undefined

–:Not used

6.3 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/ external bus (P00/AD00 to P07/AD07)
- Port 0 data register (PDR0)
- Port 0 data direction register (DDR0)

■ Port 0 Pins

Table 6.3-1 lists the port 0 pins.

Table 6.3-1 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	D
	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

Table 6.3-1 is a block diagram of the port 0 pins.

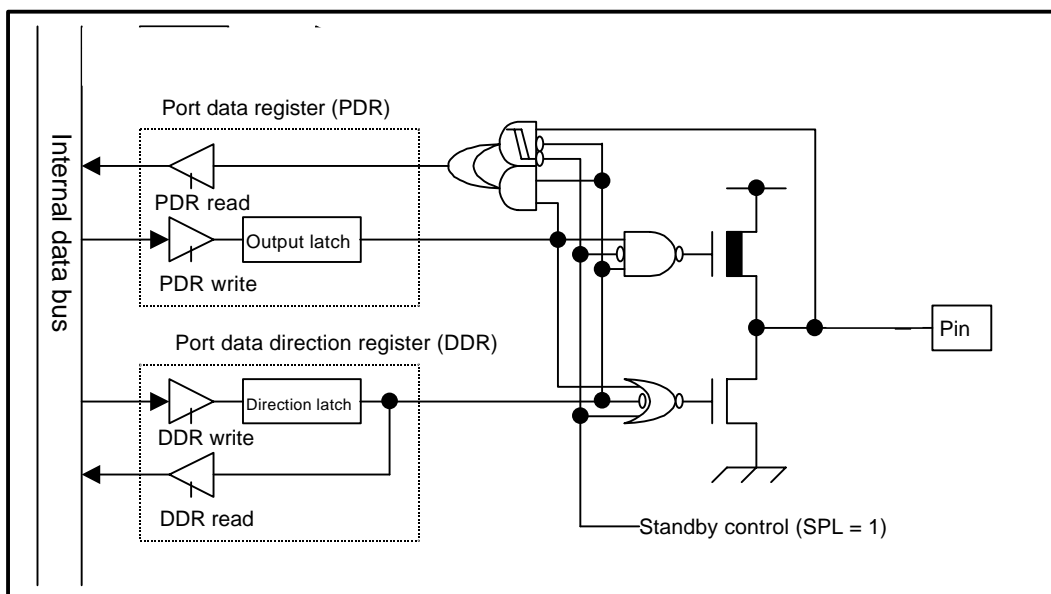


Figure 6.3-1 Block diagram of port 0 pins

■ Port 0 Registers

Port 0 registers are PDR0 and DDR0. The bits making up each register correspond to the port 0 pins on a one-to-one basis. Table 6.3-2 lists the port 0 pins and their corresponding register bits.

Table 6.3-2 Port 0 pins and their corresponding register bits

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

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

6.3 Port 0

6.3.1 Port 0 Registers (PDR0, DDR0)

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.

Table 6.3.1-1 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	XXXXXXXX _B
	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	00000000 _B
	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

6.3.2 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 as is.

<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 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 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.
- Data written to the PDR0 register in input mode is held in the output latch of the PDR but is not output to the port pins.

● Port operation after a reset

- When the CPU is reset, the DDR0 registers are initialized to 0. As a result, the output buffer is turned off (I/O mode changes to input) and the pins are placed in a high impedance state.

● Port operation in stop or time-base timer mode

If the pin state specification bit (SPL) in the low-power consumption 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 DDR0 register. Note that the inputs are fixed at a certain level to prevent leakage due to an open circuit.

Table 6.3.2-1 lists the states of the port 0 pins.

Table 6.3.2-1 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)
P00~P07	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 specification bit of low-power consumption mode control register (LPMCR)

Hi-z : High impedance

6.4 Port 1

Port 1 is a general-purpose I/O port. It can also be used for resource input. The port pins can be switched in units of bits between the I/O port and resource. 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/ resource I/O pins/ external bus (P10/IN0/AD08 to P17/PPG3/AD15)
- Port 1 data register (PDR1)
- Port 1 data direction register (DDR1)

■ Port 1 Pins

The port 1 pins are also used as resource input pins. The pins cannot be used as output port pins when they are used as resource input pins. Table 6.4-1 lists the port 1 pins.

Table 6.4-1 Port 1 pins

Port	Pin	Port function	Resource function	I/O form		Circuit type
				Input	Output	
Port 1	P10/IN0/AD08	P10	IN0	CMOS (hysteresis)	CMOS	D
	P11/IN1/AD09	P11	IN1			
	P12/IN2/AD10	P12	IN2			
	P13/IN3/AD11	P13	IN3			
	P14/PPG0/AD12	P14	PPG0			
	P15/PPG1/AD13	P15	PPG1			
	P16/PPG2/AD14	P16	PPG2			
	P17/PPG3/AD15	P17	PPG3			

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

■ Block Diagram of Port 1 Pins

Figure 6.4-1 is a block diagram of port 1 pins.

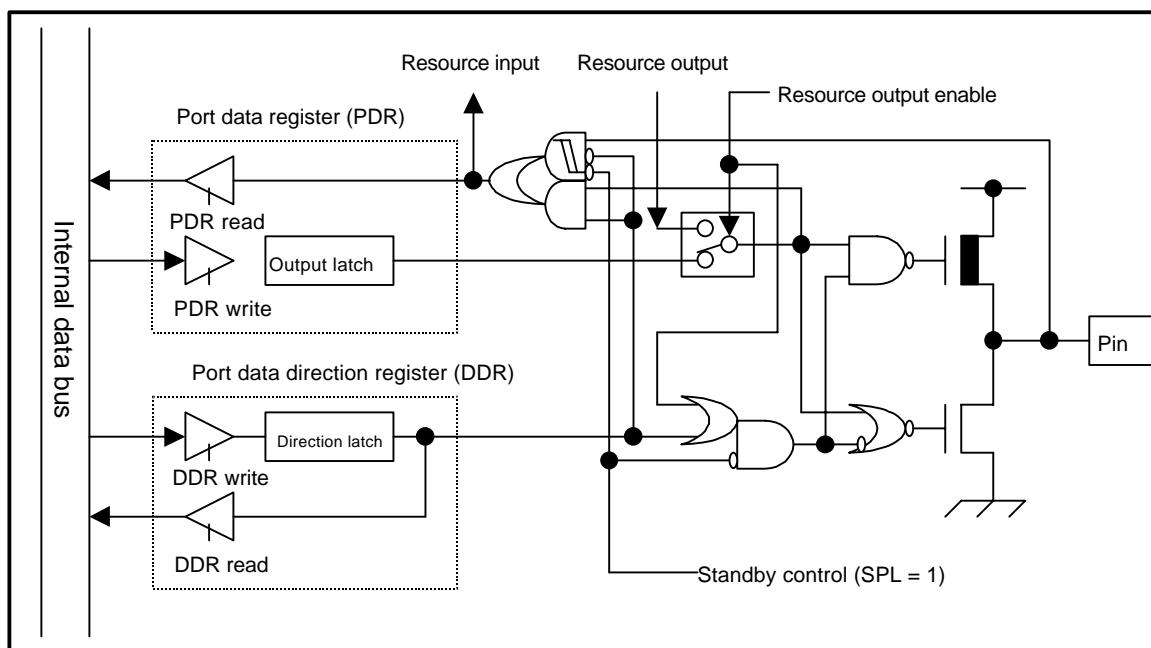


Figure 6.4-1 Block diagram of port 1 pins

■ Port 1 Registers

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

Table 6.4-2 lists the port 1 pins and their corresponding register bits.

Table 6.4-2 Port 1 pins and their corresponding register bits

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

6.4.1 Port 1 Registers (PDR1, DDR1)

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.

Reference

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

Table 6.4.1-1 lists the functions of the port 1 registers.

Table 6.4.1-1 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	000001 _H	XXXXXXXX _B
	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	000011 _H	00000000 _B
	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

6.4.2 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 held 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 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 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.
- Data written to the PDR1 register in input mode is held in the output latch of the PDR but not output to the port pins.

● Port operation for resource input

When the port is also used for resource input, the value at the pins is always supplied as resource inputs. To use an external signal for the resource, 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 registers are initialized to 0. As a result, the output buffer is turned off (I/O mode changes to input) 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.

6.4 Port 1

● Port operation in stop or time-base timer mode

If the pin state specification bit (SPL) in the low-power consumption 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 DDR1 register. Note that the inputs are fixed at a certain level to prevent leakage due to an open circuit.

Table 6.4.2-1 lists the states of the port 1 pins.

Table 6.4.2-1 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)
P10/IN0~ P17/PPG3	General-purpose I/O port	General-purpose I/O port	General-purpose I/O port	Input enabled/ output in Hi-z

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

Hi-z : High impedance

6.5 Port 2

Port 2 is a general-purpose I/O port. It can also be used for resource input and output. The port pins can be switched in units of bits between the I/O port and the resource. 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/ resource I/O pins/ external bus (P20/TIN0/AD16 to P27/INT7/AD23)
- Port 2 data register (PDR2)
- Port 2 data direction register (DDR2)

■ Port 2 Pins

The port 2 I/O pins are also used as resource I/O pins. The pins cannot be used as general-purpose I/O port pins when they are used as resource I/O pins. Table 6.5-1 lists the port 2 pins.

Table 6.5-1 Port 2 pins

Port	Pin	Port function		Resource function		I/O form		Circuit type
						Input	Output	
Port 2	P20/TIN0/A16	P20	General-purpose I/O	TIN0	16-bit reload timer 0 event input	CMOS (hysteresis)	CMOS	D
	P21/TOUT0/A17	P21		TOUT0	Output for 16-bit Reload Timer 0			
	P22/TIN1/AD18	P22		TIN1	16-bit reload timer 1 event input			
	P23/TOUT1/AD19	P23		TOUT1	Output for 16-bit Reload Timer 1			
	P24/INT4/A20	P24		INT4	Inputs for external Interrupts 4 - 7			
	P25/INT5/AD21	P25		INT5				
	P26/INT6/AD22	P26		INT6				
	P27/INT7/AD23	P27		INT7				

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

■ **Block Diagram of Port 2 Pins**

Figure 6.5-1 is a block diagram of port 2 pins.

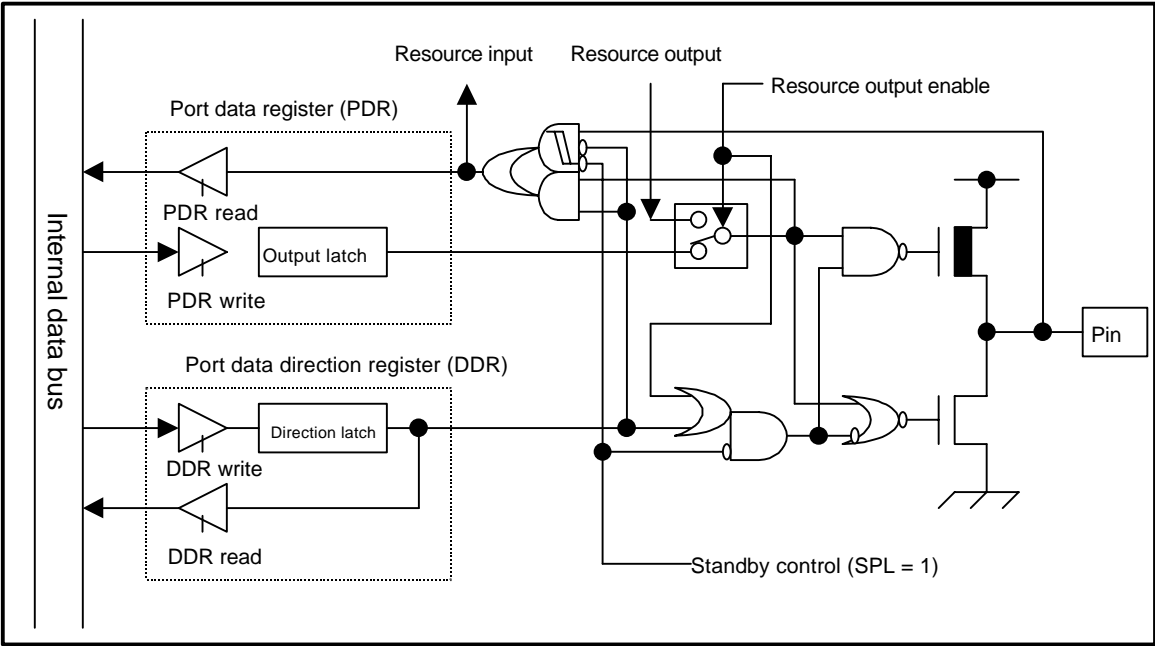


Figure 6.5-1 Block diagram of port 2 pins

When the resource output enable bit is set, the port is forcibly caused to function as resource output pins regardless of the value in the 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 6.5-2 lists the port 2 pins and their corresponding register bits.

Table 6.5-2 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

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

Table 6.5.1-1 lists the functions of the port 2 registers.

Table 6.5.1-1 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	XXXXXXXX _B
	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	11111111 _B
	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

6.5.2 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 held 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 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

- Resetting 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 held 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 resource output

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

● Port operation for resource input

When the port is also used for resource input, the value at the pins is always supplied as resource inputs. To use an external signal for the resource, 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 pull-up resistor is cut, and the pins are placed in a high impedance state.

6.5 Port 2

- 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 specification bit (SPL) in the low-power consumption 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 DDR2 register. Note that the inputs are fixed at a certain level to prevent leakage due to an open circuit.

Table 6.5.2-1 lists the states of the port 2 pins.

Table 6.5.2-1 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/TIN0~P27/INT7	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 specification bit of low-power consumption mode control register (LPMCR)

Hi-z : High impedance

6.6 Port 3

Port 3 is a general-purpose I/O port. It can also be used for resource input and output. The port pins can be switched in units of bits between the I/O port and resource. 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/resource I/O pins (P30/SOT0/ALE to P37/ADTG/CLK)
- Port 3 data register (PDR3)
- Port 3 data direction register (DDR3)

■ Port 3 Pins

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

Table 6.6-1 lists the port 3 pins.

Table 6.6-1 Port 3 pins

Port	Pin	Port function (single-chip mode)	Resource function	I/O form		Circuit type
				Input	Output	
Port 3	P30/ALE/SOT0	P30	ALE SOT0	CMOS (hysteresis)	CMOS	D
	P31/RDX/SCK0	P31	RDX SCK0			
	P32/WRLX/SIN0	P32	WRLX SIN0			
	P33/WRHX	P33	WRHX			
	P34/HRQ	P34	HRQ			
	P35/HAKX	P35	HAKX			
	P36/FRCK/RDY	P36	FRCK RDY			
	P37/ADTG/CLK	P37	ADTG CLK			

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

■ Block Diagram of Port 3 Pins

Figure 6.6-1 is a block diagram of port 3 pins.

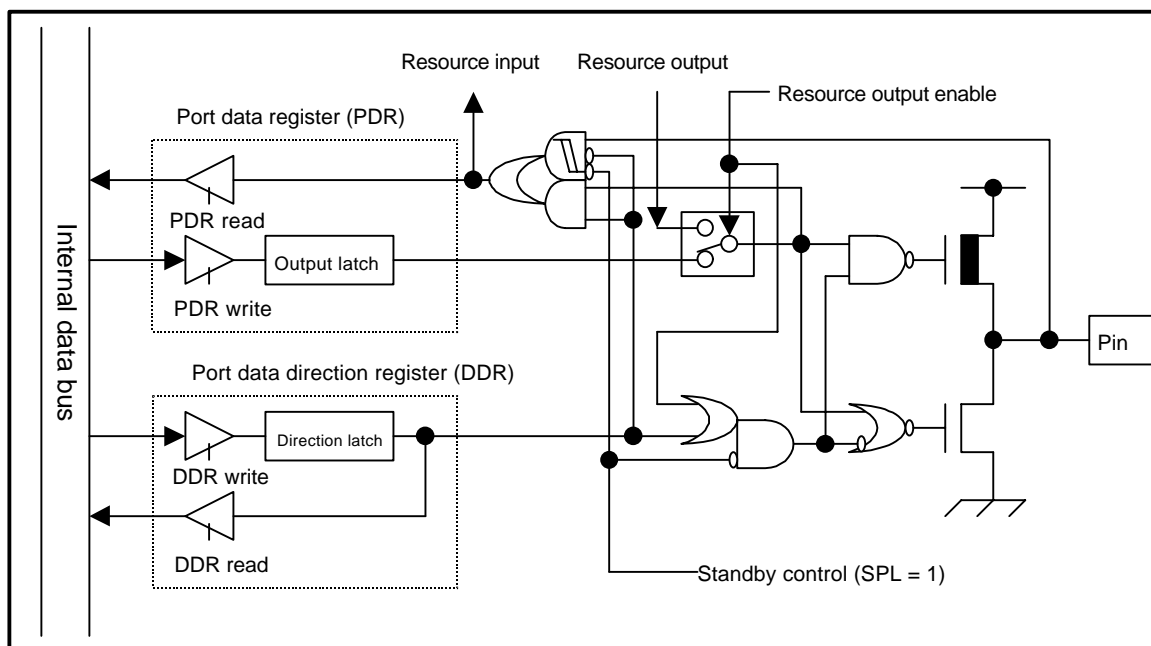


Figure 6.6-1 Block diagram of port 3 pins

■ 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 6.6-2 lists the port 3 pins and their corresponding register bits.

Table 6.6-2 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

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

Table 6.6.1-1 lists the functions of the port 3 registers.

Table 6.6.1-1 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

6.6.2 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 held 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 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

- Resetting 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 held 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 resource output

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

● Port operation for resource input

When the port is also used for resource input, the value at the pins is always supplied as resource inputs. To use an external signal for the resource, 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 pull-up resistor is cut, and 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.

6.6 Port 3

● Port operation in stop or time-base timer mode

If the pin state specification bit (SPL) in the low-power consumption 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 DDR3 register. Note that the inputs are fixed at a certain level to prevent leakage due to an open circuit. Table 6.6.2-1 lists the states of the port 3 pins.

Table 6.6.2-1 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/SOT0~ P37/ADTG	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 specification bit of low-power consumption mode control register (LPMCR)

Hi-z : High impedance

6.7 Port 4

Port 4 is a general-purpose I/O port. It can also be used for resource input and output. The port pins can be switched in units of bits between the I/O port and resource. 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/resource I/O pins (P40/SIN1 to P44/RX)
- Port 4 data register (PDR4)
- Port 4 data direction register (DDR4)

■ Port 4 Pins

The port 4 I/O pins are also used as resource I/O pins. The pins cannot be used as general-purpose I/O port pins when they are used as resource I/O pins.

Table 6.7-1 lists the port 4 pins.

Table 6.7-1 Port 4 pins

Port	Pin	Port function (single-chip mode)		Resource function		I/O form		Circuit type
						Input	Output	
Port 4	P40/SIN1	P40	General-purpose I/O	SIN1	Input for UART 1	CMOS (hysteresis)	CMOS	D
	P41/SCK1	P41		SCK1	Clock In-/Output for UART 1			
	P42/SOT1	P42		SOT1	Output for UART 1			
	P43/TX	P43		TX	CAN Transmit pin			
	P44/RX	P44		RX	CAN Receive pin			
	-	-	don't exist	-	-			
	-	-		-	-			
	-	-		-	-			

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

■ Block Diagram of Port 4 Pins

Figure 6.7-1 is a block diagram of port 4 pins.

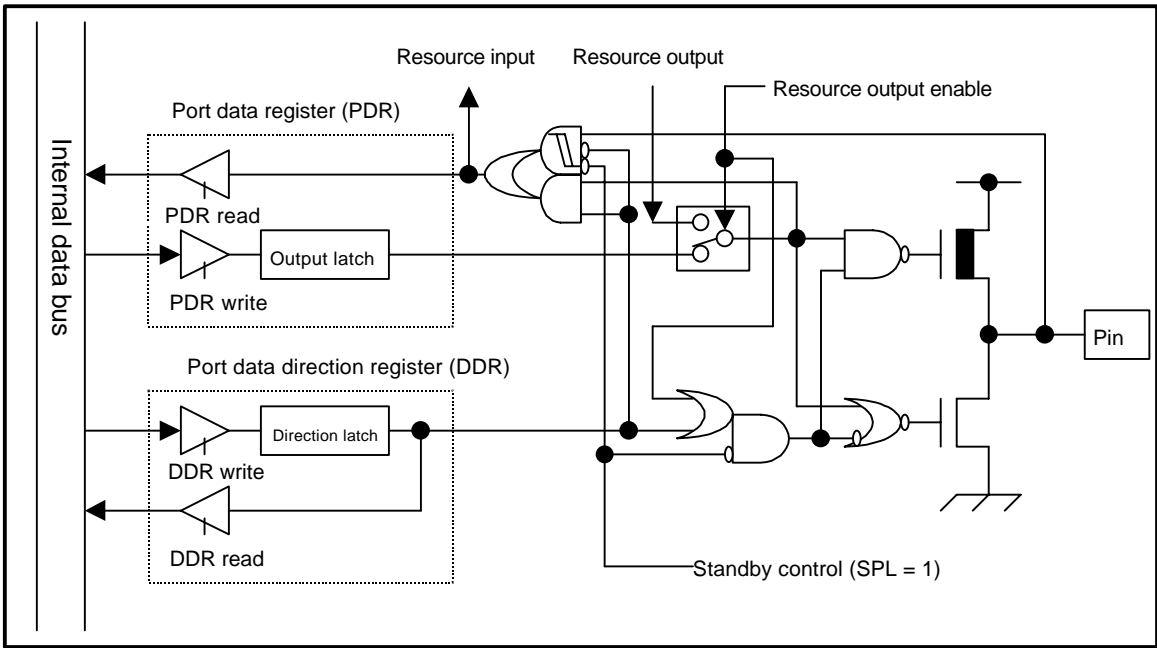


Figure 6.7-1 Block diagram of port 4 pins

<Check>

When the resource output enable bit is set, the port is forcibly caused to function as resource output pins regardless of the value in the DDR4 register. 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 6.7-2 lists the port 4 pins and their corresponding register bits.

Table 6.7-2 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	—	-	-	P44	P43	P42	P41	P40

6.7.1 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.

Reference

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

Table 6.7.1-1 lists the functions of the port 4 registers.

Table 6.7.1-1 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	-- -XXXXXB
	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	-- -00000B
	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

6.7.2 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 held 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 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

- Resetting 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 held 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 resource output

The resource output enable bit is set to enable the port to be used for resource output. The state of the resource enable bit takes precedence when specifying a switch between input and output. Even if a DDR4 register bit is 0, the corresponding port pin is used for resource output if the resource has been enabled for output. Because the value at the pins can be read even if resource output is enabled, the resource output value can be read.

● Port operation for resource input

When the port is also used for resource input, the value at the pins is always supplied as resource inputs. To use an external signal for the resource, 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 pull-up resistor is cut, and the pins are placed in a high impedance state.

6.7 Port 4

- 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 specification bit (SPL) in the low-power consumption 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. Note that the inputs are fixed at a certain level to prevent leakage due to an open circuit. Table 6.7.2-1 lists the states of the port 4 pins.

Table 6.7.2-1 States of port 4 pins

Pin	Normal operation	Sleep mode	Stop mode or time-base timer mode (SPL = 0)	Stop mode or time-base timer mode (SPL = 1)
P40/SIN1~ P47/RX	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 specification bit of low-power consumption mode control register (LPMCR)

Hi-z : High impedance

6.8 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 6.8-1 lists the port 5 pins.

Table 6.8-1 Port 5 pins

Port	Pin	Port function (single-chip mode)		Resource function		I/O form		Circuit type
						Input	Output	
Port 5	P50/AN0	P50	General-purpose I/O	AN0	Analog input 0	Analog/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 6.8-1 is a block diagram of port 5 pins

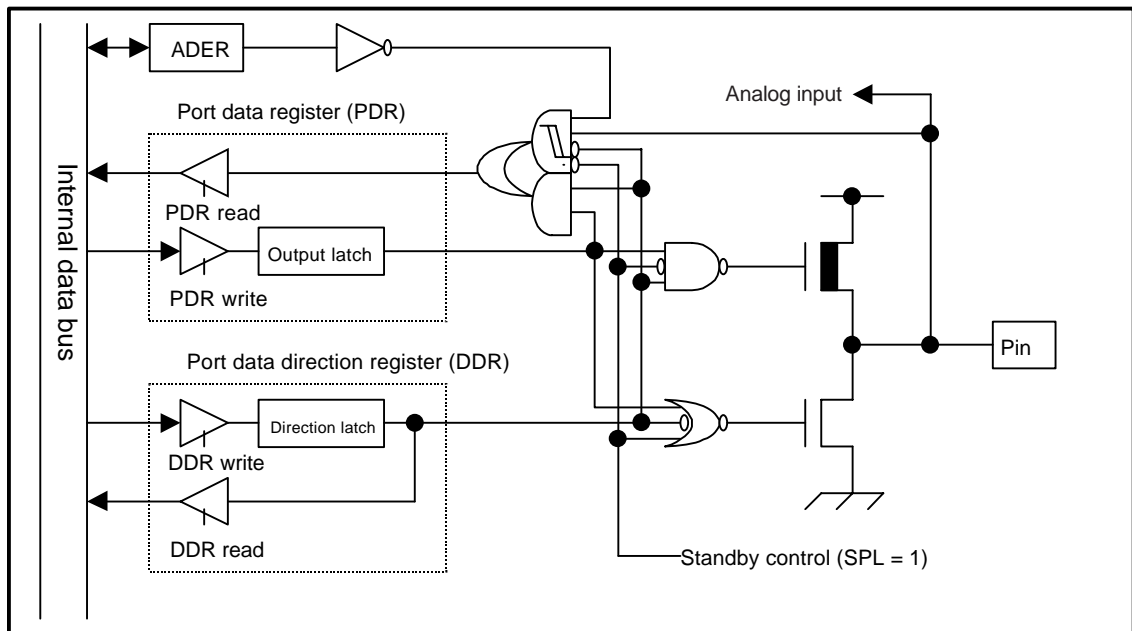


Figure 6.8-1 Block diagram of port 5 pins

<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 6.8-2 lists the port 5 pins and their corresponding register bits.

Table 6.8-2 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

6.8.1 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 bin 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 analogue input.

Table 6.8.1-1 lists the functions of the port 5 registers.

Table 6.8.1-1 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.	Setting an DDR5 bit to 0 enables the pin for a high-impedance. Setting an DDR5 bit to 1 enables the pin functions as an output port, and the pin is set to the low level.	R/W	000005H	XXXXXXXB
	1	The pin is at the high level.	Setting an DDR5 bit to 0 enables the pin for a high-impedance. Setting an DDR5 bit to 1 enables the pin functions as an output port, and the pin is set to the high level.			

6.8 Port 5

Register	Data	During reading	During writing	Read/Write	Address	Initial value
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	000015 _H	00000000 _B
	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	00001B _H	11111111 _B
	1	Analog input mode				

R/W : Read/write enabled

X : Undefined

6.8.2 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 as is.
- 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 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 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 held 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 specification bit (SPL) in the low-power consumption 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. Note that the inputs are fixed at a certain level to prevent leakage due to an open circuit.

Table 6.8.2-1 lists the states of the port 5 pins.

Table 6.8.2-1 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 specification bit of low-power consumption mode control register (LPMCR)

Hi-z : High impedance

6.9 Port 6

Port 6 is a general-purpose I/O port. It can also be used for resource input and output. The port pins can be switched in units of bits between the I/O port and resource. 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/resource I/O pins (P60/INT1 to P63/INT3)
- Port 6 data register (PDR6)
- Port 6 data direction register (DDR6)

■ Port 6 Pins

The port 6 I/O pins are also used as resource I/O pins. The pins cannot be used as general-purpose I/O port pins when they are used as resource I/O pins.

Table 6.9-1 lists the port 6 pins.

Table 6.9-1 Port 6 pins

Port	Pin	Port function (single-chip mode)		Resource function		I/O form		Circuit type
						Input	Output	
Port 6	P60/INT0	P60	General- purpose I/O	INT0	Input for external interrupt 0	CMOS (hysteresis)	CMOS	D
	P61/INT1	P61		INT1	Input for external interrupt 1			
	P62/INT2	P62		INT2	Input for external interrupt 2			
	P63/INT3	P63		INT3	Input for external interrupt 3			
	—	—	don't exist	—	—			
	—	—		—	—			
	—	—		—	—			
	—	—		—	—			

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

■ Block Diagram of Port 6 Pins

Figure 6.9-1 is a block diagram of port 6 pins.

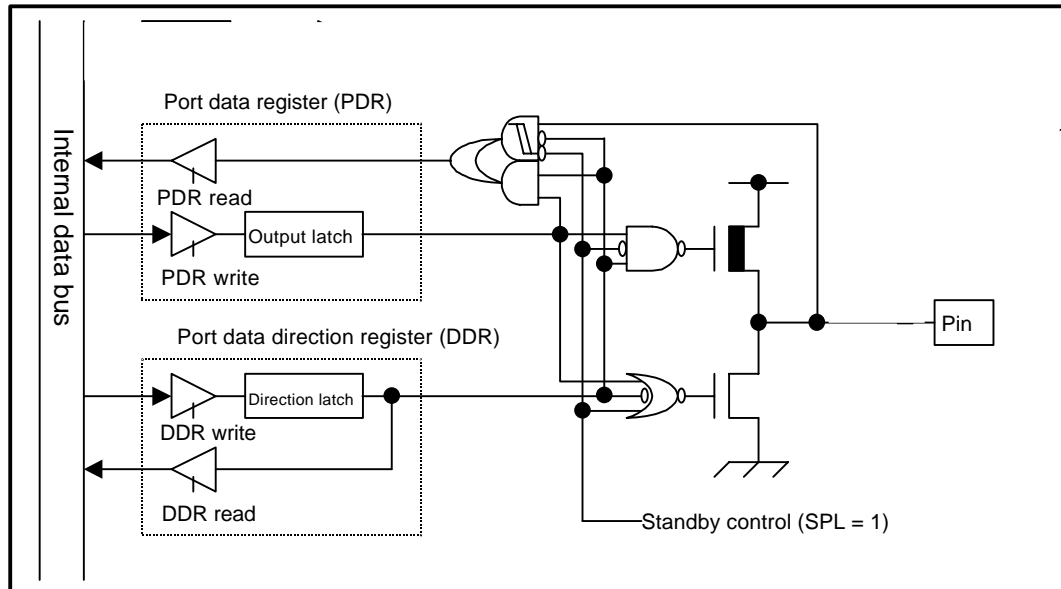


Figure 6.9-1 Block diagram of port 6 pins

■ 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 6.9-2 lists the port 6 pins and their corresponding register bits.

Table 6.9-2 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

6.9 Port 6

6.9.1 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.

Reference

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

Table 6.9.1-1 lists the functions of the port 6 registers.

Table 6.9.1-1 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	----XXXX _B
	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	----0000 _B
	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

6.9.2 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 held 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 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 resource output

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

● Port operation for resource input

When the port is also used for resource input, the value at the pins is always supplied as resource inputs. To use an external signal for the resource, 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 pull-up resistor is cut, and 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 specification bit (SPL) in the low-power consumption 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. Note that the inputs are fixed at a certain level to prevent leakage due to an open circuit.

Table 6.9.2-1 lists the states of the port 6 pins.

Table 6.9.2-1 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~P67/SCK1	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 specification bit of low-power consumption mode control register (LPMCR)

Hi-z : High impedance

6.10 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.

Table 6.10-1 is an example of connecting the eight-segment LED to the MB90495 ports.

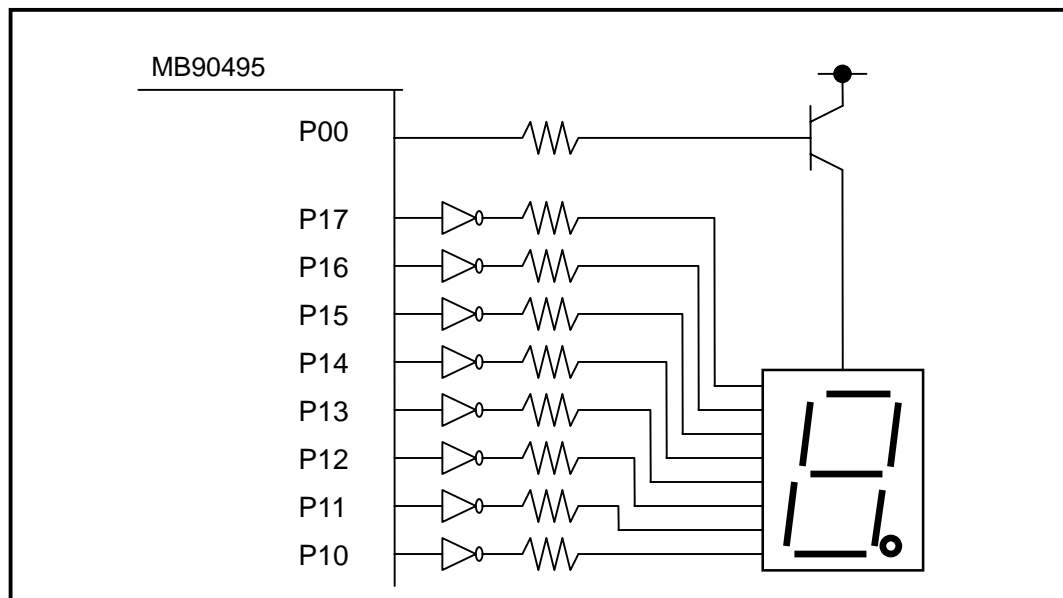


Figure 6.10-1 Example of eight-segment LED connection

● 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 (#xxxxxx0B).
MOV I:DDR0, #11111111B ; Puts all port 0 bits in output mode.
MOV I:PDR1, #11111111B ; Sets all port 1 bits to 1.
MOV I:DDR1, #11111111B ; Puts all port 1 bits in output mode.
CODE ENDS
END START
```


CHAPTER 7 UART0

This chapter explains the functions and operation of UART0.

- 7.1 Overview of UART0
- 7.2 UART0 Registers
- 7.3 UART0 Operating Modes and Clock Selection
- 7.4 UART0 Flags and Interrupt Sources
- 7.5 Negative clock operation
- 7.6 UART0 Sample Applications and Precautionary Information

7.1 Overview of UART0

The UART0 is a serial I/O port used for asynchronous (start-stop synchronized) communication as well as for CLK-synchronized communication, and provides the following features.

- Full-duplex double buffer
- Asynchronous (start-stop synchronized) and CLK-synchronous communication capability
- Multi-processor mode support
- On-chip dedicated baud rate generator
- Asynchronous: 9615/31250/4808/2404/1202 bps
CLK synchronous: 1M/500K/250K/125K/62.5 Kbps
(at internal clock speeds of 6,8,10,12 & 16 MHz)
- Automatic baud rate setting from external clock input or internal timer
- Error detection function (parity, framing, overrun)
- Transfer communication in NRZ transfer format
- Intelligent I/O service support

7.2 UART0 Registers

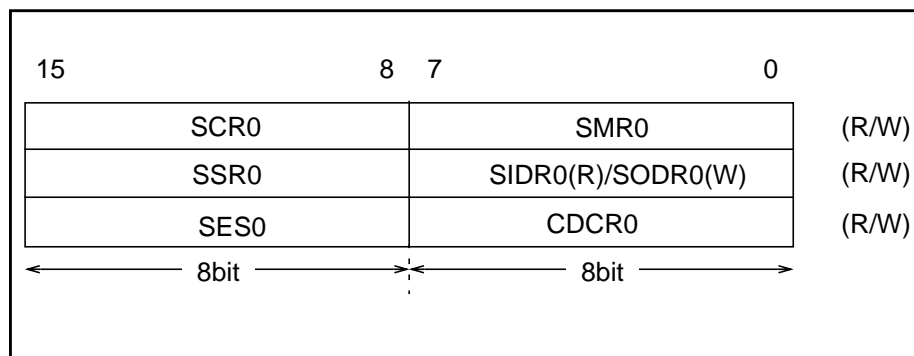


Figure 7.2-1 UART0 registers.

Address: 000020H	7	6	5	4	3	2	1	0	Serial mode register (SMR0)
	MD1	MD0	CS2	CS1	CS0	Reserved	SCKE	SOE	
Address: 000021H	15	14	13	12	11	10	9	8	Serial control register (SCR0)
	PEN	P	SBL	CL	A/D	REC	RXE	TXE	
Address: 000022H	7	6	5	4	3	2	1	0	Serial input /Serial output register (SIDR0/SODR0)
	D7	D6	D5	D4	D3	D2	D1	D0	
Address: 000023H	15	14	13	12	11	10	9	8	Serial status register (SSR0)
	PE	ORE	FRE	RDRF	TDRE	—	RIE	TIE	
Address: 000024H	7	6	5	4	3	2	1	0	UART0 prescaler control register (CDCR0)
	MD	—	—	—	DIV3	DIV2	DIV1	DIV0	
Address: 000025H	15	14	13	12	11	10	9	8	UART0 Clock Edge Select (SES0)
	-	-	-	-	-	-	-	NEG	

Figure 7.2-2 UART0 Registers

■ UART0 Block Diagram

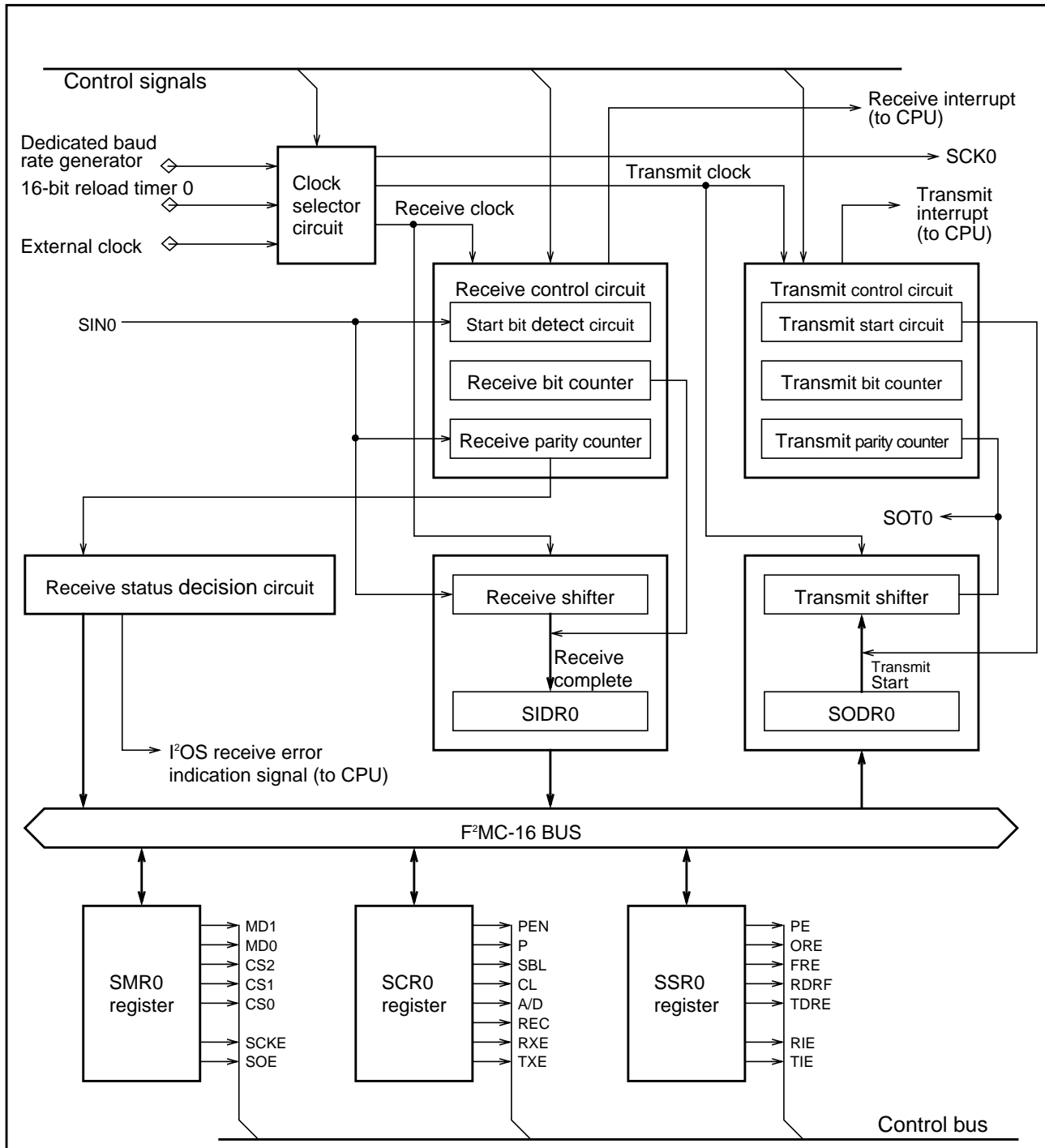


Figure 7.2-3 UART0 Block Diagram

7.2.1 Serial Mode Register 0 (SMR0)

The serial mode register 0 (SMR0) sets the operating mode of the UART0. Operating mode settings should be entered when the unit is not in operation. Do not write to this register during operation.

The SMR0 register has the following bit configuration

SMR0	7	6	5	4	3	2	1	0	Default value
Address: 000020H	MD1	MD0	CS2	CS1	CS0	Reserved	SCKE	SOE	00000000B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

[bit 7, 6] MD1, MD0 (MoDe select)

These bits select the UART0 operation mode, according to the settings listed in the table.

Table 7.2.1-1 Operating Mode Selections

Mode	MD1	MD0	Operating mode
0	0	0	Asynchronous (start-stop synchronized) normal mode
1	0	1	Asynchronous (start-stop synchronized) multi-processor mode
2	1	0	CLK synchronous mode
—	1	1	Prohibited

Note: Mode 1, CLK-asynchronous multi-processor mode, is used when one host CPU is connected to multiple slave CPUs. This UART0 resource is not able to determine the data format of incoming data, and therefore in multi-processor mode supports only the master processor.

Also, in this configuration the receive parity check function cannot be used, and therefore the PEN bit in the UMC1 register should be set to '0.'

[bit 5-bit 3] CS2, CS1, CS0 (Clock Select)

These bits select the baud rate clock source. The baud rate is determined at the same time as selection of the baud rate generator. The table below shows the clock input selection settings.

Table 7.2.1-2 Clock Input Selection Settings

CS2 to CS0	Clock input
000B to 100B	Dedicated baud rate generator
101B	Reserved
110B	Internal timer
111B	External clock

Note: When the internal timer is selected, the MB90495 series selects 16-bit reload timer 0 output.

[bit 2] Reserved

Always write '0' to this bit.

[bit 1] SCKE (SCLK Enable)

For communication in CLK synchronous mode (mode 2), this bit determines whether the SCK0 pin is used as a clock input pin or a clock output pin.

In CLK asynchronous modes or external clock mode, this bit should be set to '0.'

0: SCK1 pin functions as clock input pin

1: SCK1 pin functions as clock output pin

Note: When the pin functions as a clock input, an external clock source must be selected.

[bit0] SOE (Serial Output Enable)

This bit determines whether external pins that also can be used as general purpose I/O port pins will function as serial output pins (SOT0) or as I/O port pins.

0: General purpose I/O port pin function

1: Serial data output pin (SOT0) function

7.2.2 Serial Control Register 0 (SCR0)

The serial control register 0 (SCR0) register controls the transfer protocol used for serial transmission. The SCR0 register has the following bit configuration.

SCR0	15	14	13	12	11	10	9	8	Default value
Address: 000021H	PEN	P	SBL	CL	A/D	REC	RXE	TXE	00000100 _B
	R/W	R/W	R/W	R/W	R/W	W	R/W	R/W	

Figure 7.2.2-1 Serial Control Register (SCR0)

[bit 15] PEN (Parity Enable)

This bit determines whether parity bits are attached to data in serial transmission.

- 0: No parity
- 1: Parity

Note: Parity bit attachment is available only in asynchronous (start-stop synchronized) communications in normal mode (mode 0). In multi-processor mode (mode 1) and all CLK-synchronous communication (mode 2), no parity bits may be attached.

[bit 14] P (Parity)

This bit selects even or odd parity for data communications in which a parity bit is used.

- 0: Even parity
- 1: Odd parity

[bit 13] SBL (Stop Bit Length)

This bit sets the length of the stop bit that marks the frame end in asynchronous (start-stop synchronized) communication.

- 0: 1 stop bit
- 1: 2 stop bits

[bit 12] CL (Character Length)

This bit sets the data length of one frame.

- 0: 7-bit data
- 1: 8-bit data

Note: 7-bit data handling is available only in asynchronous (start-stop synchronized) communications in normal mode (mode 0). In multi-processor mode (mode 1) and all CLK-synchronous communication (mode 2), 8-bit data should be used.

[bit 11] A/D (Address/Data)

This bit determines the data format of transmit and receive frames in asynchronous (start-stop synchronized) communication in multi-processor mode (mode 1).

- 0: Data frame
- 1: Address frame

[bit 10] REC (Receiver Error Clear)

This bit clears the error flags (PE, ORE, FRE) in the SSR0 register.

A write value of '1' is not valid, and the read value is '1' at all times.

[bit 9] RXE (Receiver Enable)

This bit controls UART0 receiver operations.

- 0: Receiver operation prohibited

1: Receiver operation enabled

Note: If receiver operation is prohibited while reception is in progress (while data is present in the receive shift register), the receiver will not stop operating until reception of the current frame is completed, and the data has been stored in the receive data buffer SIDR0 register.

[bit 8] TXE (Transmit Enable)

This bit controls UART0 transmit operation.

0: Transmit operation prohibited

1: Transmit operation enabled

Note: If transmit operation is prohibited while transmission is in progress (while data is being output from the transmit register), the transmitter will not stop operating until there is no more data remaining in the transmit data buffer SODR0 register.

7.2.3 Serial Input Data Register 0 (SIDR0) / Serial Output Data Register 0 (SODR0)

These registers function as receive and transmit data buffer registers.

The SIDR0 and SODR0 registers have the following bit configuration.

SIDR0	7	6	5	4	3	2	1	0	Default value
Address: 000022H	D7	D6	D5	D4	D3	D2	D1	D0	Undefined
	R	R	R	R	R	R	R	R	
SODR0	7	6	5	4	3	2	1	0	
Address: 000022H	D7	D6	D5	D4	D3	D2	D1	D0	Undefined
	W	W	W	W	W	W	W	W	

The serial input data register 0 (SIDR0) functions as a data buffer register for receiving serial data.

The serial output data register 0 (SODR0) functions as a data buffer register for transmitting serial data.

When using 7-bit data length, the top bit (D7) contains invalid data. Be sure the DTRE bit in the SSR0 register is set to '1' before writing to the SODR0 register.

Note: Writing to these addresses refers to writing to the SODR0 register, and reading refers to reading from the SIDR0 register.

7.2.4 Serial Status Register 0 (SSR0)

The serial status register 0 (SSR0) is composed of flags that indicate the operating status of the UART0.

The SSR0 register has the following bit configuration.

SSR0	15	14	13	12	11	10	9	8	Default value
Address: 000023H	PE	ORE	FRE	RDRF	TDRE	—	RIE	TIE	00001_00b
	R	R	R	R	R		R/W	R/W	

[bit 15] PE (Parity Error)

This interrupt request flag is set when a parity error occurs during receive.

Once set, this flag is cleared by writing '0' to the REC bit (bit 10) in the SCR0 register.

When this bit is set, data in the SDR0 register is invalid.

- 0: No parity error
- 1: Parity error occurred

[bit 14] ORE (Over Run Error)

This interrupt request flag is set when an overrun error occurs during receive.

Once set, this flag is cleared by writing '0' to the REC bit (bit 10) in the SCR0 register.

When this bit is set, data in the SDR0 register is invalid.

- 0: No overrun error
- 1: Overrun error occurred

[bit 13] FRE (Framing Error)

This interrupt request flag is set when a framing error occurs during receive.

Once set, this flag is cleared by writing '0' to the REC bit (bit 10) in the SCR0 register.

When this bit is set, data in the SDR0 register is invalid.

- 0: No framing error
- 1: Framing error occurred

[bit 12] RDRF (Receiver Data Register Full)

This interrupt request flag is set to indicate that data is present in the SDR0 register

This flag is set when receive data is loaded into the SDR0 register, and is automatically cleared when the data is read from the SDR0 register.

- 0: No receive data
- 1: Receive data present

[bit 11] TDRE (Transmit Data Register Empty)

This interrupt request flag is set to indicate that outgoing data can be written to the SODR0 register.

This flag is cleared when outgoing data is written to the SODR0 register. It is then reset when the written data starts loading into the transmit shifter to indicate that the next data can be written to the SODR0 register.

- 0: Prohibits writing of send data
- 1: Enables writing of send data

[bit 9] RIE (Receiver Interrupt Enable)

This bit controls receiver interrupts.

- 0: Interrupt prohibited

1: Interrupt enabled

Note: Receiver interrupt sources include PE, ORE and FRE errors, as well as normal receive as indicated by the RDRF flag.

[bit 8] TIE (Transmit Interrupt Enable)

This bit controls transmit interrupts.

0: Interrupt prohibited

1: Interrupt enabled

Note: Transmit interrupt sources include transmission requests indicated by the TDRE flag.

7.2.5 Prescaler Control Register 0 (CDCR0)

The prescaler control register (CDCR0) controls the machine clock frequency divider.

The UART0 operating clock signal can be generated by dividing the machine clock signal pulse. The prescaler is designed to enable constant baud rates from a variety of machine clock speeds.

The output from the prescaler is used by the I/O expanded serial interface.

The CDCR0 register has the following bit configuration.

CDCR0	7	6	5	4	3	2	1	0	Default value
Address: 000024H	MD	—	—	—	DIV3	DIV2	DIV1	DIV0	0___1111B
	R/W				R/W	R/W	R/W	R/W	

Table 7.2.5-1 UART0 Prescaler Control Register (CDCR0)

[bit 7] MD (Machine clock divide MoDe select)

This bit enables the prescaler operation.

0: Prescaler stopped

1: Prescaler operating

[bit 3, 2, 1, 0] DIV3-0 (DIVide 3-0)

These bits determine the division of the machine clock frequency as shown in Table 1.3.

Table 7.2.5-2 Machine Clock Division Ratios

DIV3-0	Division ratio
1101 _B	Divide by 3
1100 _B	Divide by 4
1011 _B	Divide by 5
1010 _B	Divide by 6
1000 _B	Divide by 8

Note: After changing the division ratio, allow an interval of two cycles for the clock frequency to stabilize before starting communication.

7.3 UART0 Operating Modes and Clock Selection

The UART0 has two types of operating mode, asynchronous mode and CLK-synchronous mode. Changes of mode are controlled by settings in the SMR0 register and SCR0 register.

■ UART0 Operating Modes

Table 7.3-1 UART0 Operating Modes

Mode	Parity bit	Data length	Operating mode	Stop bit length
0	Y/N	7	Asynchronous (start-stop synchronized) normal mode	1-bit or 2-bit
	Y/N	8		
1	N	8+1	Asynchronous (start-stop synchronized) multi-processor mode	
2	N	8	CLK synchronous mode	N

Note: In asynchronous (start-stop synchronized) normal mode, stop bit length can be set for outgoing transmission only. For receive, the setting is always 1-bit. The unit does not operate in modes other than those shown, and only these settings should be used.

■ UART0 Clock Selection

○ Dedicated baud rate generator

When the dedicated baud rate generator is selected, the baud rate settings listed in the tables below are available. Also, prescaler settings are shown:

Table 7.3-2 Baud Rates (Asynchronous communication)

CS2	CS1	CS0	Asynchronous (start-stop synchronized)	Calculation formula
0	0	0	9615	$(\phi / \text{div}) / (8 \times 13 \times 2)$
0	0	1	4808	$(\phi / \text{div}) / (8 \times 13 \times 2^2)$
0	1	0	2404	$(\phi / \text{div}) / (8 \times 13 \times 2^3)$
0	1	1	1202	$(\phi / \text{div}) / (8 \times 13 \times 2^4)$
1	0	0	31250	$(\phi / \text{div}) / 2^6$

Table 7.3-3 Baud Rates (CLK-synchronized communication)

CS2	CS1	CS0	CLK - synchronized	Calculation formula
0	0	0	1 M	$(\phi / \text{div}) / 2$
0	0	1	500 K	$(\phi / \text{div}) / 2^2$
0	1	0	250 K	$(\phi / \text{div}) / 2^3$
0	1	1	125 K	$(\phi / \text{div}) / 2^4$
1	0	0	62.5 K	$(\phi / \text{div}) / 2^5$

Note: ϕ represents machine cycle speed.

Table 7.3-4 Prescaler Settings

MD	DIV3	DIV2	DIV1	DIV0	div	Recommended machine clock speed
1	1	1	0	1	3	6 MHz
1	1	1	0	0	4	8 MHz
1	1	0	1	1	5	10 MHz
1	1	0	1	0	6	12 MHz
1	1	0	0	0	8	16 MHz

○ Internal timer

When bits CS2-0 are set to '110,' the internal timer signal is selected, and the internal timer (16-bit timer 0) operates in reload mode. In this case, baud rates are determined as follows.

$$\begin{aligned} \text{Asynchronous (start-stop synchronized):} & \quad (\Phi \div N) \div (16 \cdot 2 \cdot (n + 1)) \\ \text{CLK synchronous:} & \quad (\Phi \div N) \div (2 \cdot (n + 1)) \end{aligned}$$

N: timer count clock source

n: timer reload value

The table below shows the relation between baud rates and reload values (decimal values) at a machine cycle speed of 7.3728MHz.

Table 7.3-5 Baud Rates and Reload Values

Reload value Baud rate	N=2 ¹ (machine cycle division by 2)	N=2 ³ (machine cycle division by 8)
38400	2	—
19200	5	—
9600	11	2
4800	23	5
2400	47	11
1200	95	23
600	191	47
300	383	95

When selecting the internal timer (16-bit timer0) as the baud rate clock source, note that the 16-bit timer0 output signal TOUT0 is already connected to the MB90495 controller internally. Therefore, it is not necessary to make an external connection from the 16-bit timer0 external output pins TOUT0 to the UART0 external clock input pin SCK1. Also, this means that unless used in some other fashion, the timer pins are available for use as I/O port pins.

○ External clock

When bits CS2-0 are set to '111' the external clock source is selected and baud rates are determined by the following formula, in which f represents the external clock frequency.

$$\begin{aligned} \text{Asynchronous (start-stop synchronized) mode:} & \quad f/16 \\ \text{CLK synchronous:} & \quad f \end{aligned}$$

Note that f has a maximum value of 1MHz.

7.3.1 Asynchronous (Start-Stop Synchronized) Mode

The UART0 handles only data in NRZ (non-return to zero) format. The Data format is illustrated below.

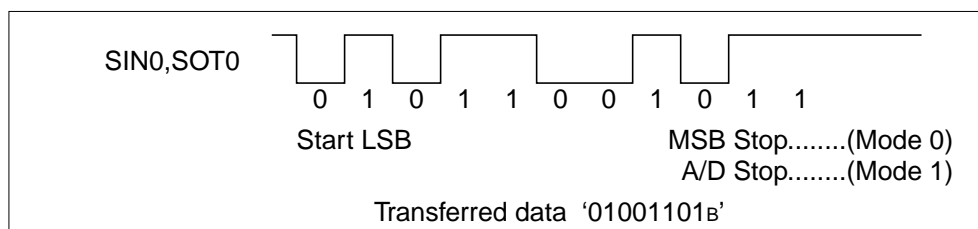


Figure 7.3.1-1 Asynchronous (Start-Stop Synchronized) Mode Data Transfer Format (Mode 0, 1)

Transfer data must begin with a start bit ('L' level data value), followed by LSB-first data of the designated bit-length, and ending with a stop bit ('H' level data value). When an external clock signal is selected, the clock should be input at all times.

In normal mode (mode 0), data length may be set to 7 bits or 8 bits, however in multi-processor mode (mode 1) the data length must be 8 bits. Also, no parity bit may be attached in multi-processor mode. Instead, an A/D bit must be attached.

■ Asynchronous (Start-Stop Synchronized) Mode Receive Operation

Whenever the RXE bit (bit 9) in the SCR0 register is set to '1' the UART0 is receiving.

The appearance of a start bit on the receive line allows one frame of data to be received according to the data format determined by the SCR0 register. When one frame of data has been received, error flags will be set if the corresponding errors have occurred, and then the RDRF flag (SST register bit 12) will be set. At this time if the RIE bit (bit 9) in the SSR0 register is set to '1' a receive interrupt will be sent to the CPU. The CPU will check each of the flags in the SSR0 register and read the SIDR0 register if data has been received normally. If any errors have occurred, the necessary processing should be followed.

The RDRF flag is cleared when the SIDR0 register is read.

■ Asynchronous (Start-Stop Synchronized) Mode Transmit Operation

Whenever the TDRE flag (bit 11) in the SSR0 register is set to '1' the UART0 is writing outgoing data to the SODR0 register. If the TXE bit (bit 8) is set to '1' transmit operation is in progress.

As soon as data in the SODR0 register starts to be transferred to the transmit shift register for transmission, the TDRE flag is reset. This enables the next unit of outgoing data to be placed in the SODR0 register. At this time if the TIE bit (bit 8) in the SSR0 register is set to '1' a transmission interrupt is sent to the CPU, causing outgoing data to be placed into the SODR0 register.

The TDRE flag is momentarily cleared each time data is placed into the SODR0 register.

7.3.2 CLK Synchronous Mode

The UART0 handles only data in NRZ (non-return to zero) format. The relation between the transmit and receive clock and data in CLK synchronous mode is illustrated below.

■ CLK Synchronous Mode Data Transfer Format

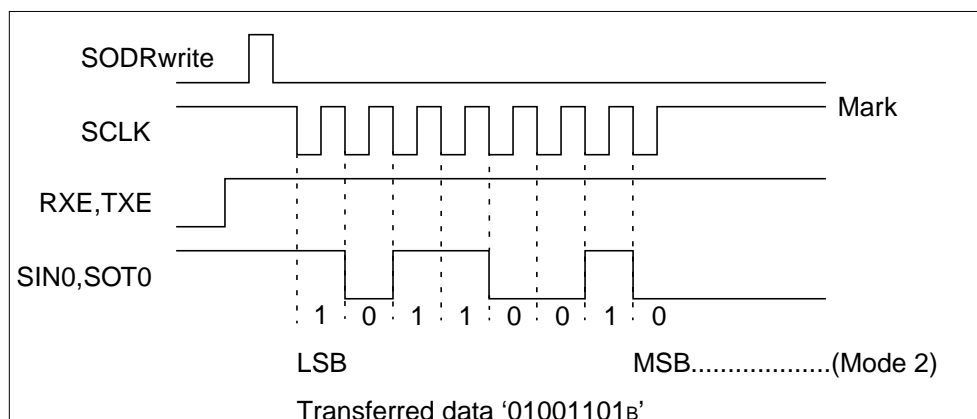


Figure 7.3.2-1 CLK Synchronous Mode Data Transfer Format (Mode 2)

When an internal clock signal source (dedicated baud rate generator or internal timer) is selected, a receive clock signal is automatically generated each time data is transmitted.

When an external clock source is selected it is necessary to provide an accurate 1-byte clock signal after data is confirmed present in the transmit data buffer register SODR0 (indicated by the TDRE flag = '0'). Note also that the signal must return to mark level before and after transmit operation.

Data length is 8-bit only, and no parity bit may be attached. Also, there is no start/stop bit so that no error detection is enabled except for overrun errors.

■ Control Register Settings for CLK Synchronous Mode

When using CLK synchronous mode, the following settings are made to each of the control registers.

○ SMR0 register

MD1, MD0: 10

CS2, CS1, CS0: Indicate clock input

SCKE: 1 for dedicated baud rate generator or internal timer
0 for external clock

SOE: 1 to send, 0 to receive only

○ SCR0 register

PEN: 0

P, SBL, A/D: These bits have no significance

CL: 1

REC: 0 (to initialize)

RXE, TXE: At least one must be '1'

○ SSR0 Register

RIE: 1 if interrupts are used, 0 if interrupts are not used

TIE: 0

■ Start of Communication in CLK Synchronous Mode

Communication starts by writing to the SODR0 register. Even if data is to be only sent (not received), it is first necessary to write dummy data to the SORD1 register.

■ End of Communication in CLK Synchronous Mode

The end of communication can be verified by the change of the RDRF flag in the SSR0 register to '1.' To determine whether the communication was performed normally, read the ORE bit in the SSR0 register.

7.4 UART0 Flags and Interrupt Sources

The UART0 has five flags, PE, ORE, FRE, RDRF and TDRE, and two interrupt sources, one for transmit and one for receive.

■ UART0 Flags

The five flags are the PE, ORE, FRE, RDRF and TDRE flags. The first three are set when transmit errors occur, the PE flag for a parity error, the ORE flag for an overrun error, and the FRE flag for a framing error, and are released by writing '0' to the REC bit in the SCR0 register. The RDRF flag is set when receive data is loaded into the SIDR0 register, and cleared when the data is read out of the SIDR0 register. Note however that there is no parity detect function in mode 1, and no parity detect function or framing error detect function in mode 2. The TDRE flag is set when the SODR0 register is empty and ready for data write access, and is cleared when data is written to the SODR0 register.

■ UART0 Interrupt Sources

The UART0 has two interrupt sources, one for receive and one for transmit. During receive, interrupt requests are initiated by setting the PE, ORE, FRE or RDRF flags. During transmit, interrupt requests are initiated by setting the TDRE flag.

Interrupt flag set timing in each operating mode is described in section 7.4.1 "UART0 Interrupts and Flag Set Timing".

7.4.1 UART0 Interrupts and Flag Set Timing

This section describes the timing of interrupts and flag setting in each UART0 operating mode.

■ UART0 Interrupts and Flag Set Timing

○ Mode 0 Receive

The PE, ORE, FRE and RDRF flags are set and the interrupt request signal is sent to the CPU following the end of a receive transfer, when the final stop bit is detected. If any one of the PE, ORE or FRE flags is active, the data in the SIDR0 register will be invalid.

The timing of the PE, ORE, FRE, and RDRF flags (mode 0) is illustrated below.

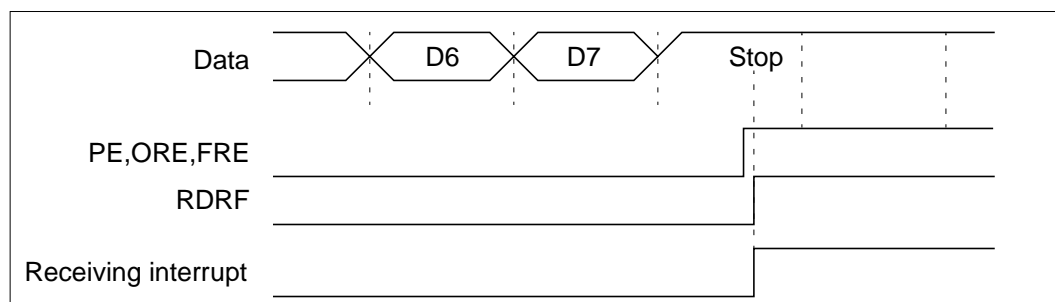


Figure 7.4.1-1 PE, ORE, FRE, RDRF Flag Set Timing (Mode 0)

○ Mode 1 Receive

The ORE, FRE and RDRF flags are set and the interrupt request signal is sent to the CPU after the end of a receive transfer, when the final stop bit is detected. Also, if the receive data length is 8 bits, the 9th bit indicating address/data will be invalid. If either the ORE or FRE flags is active, the data in the SIDR0 register will be invalid.

The timing of the ORE, FRE, and RDRF flags (mode 1) is illustrated below.

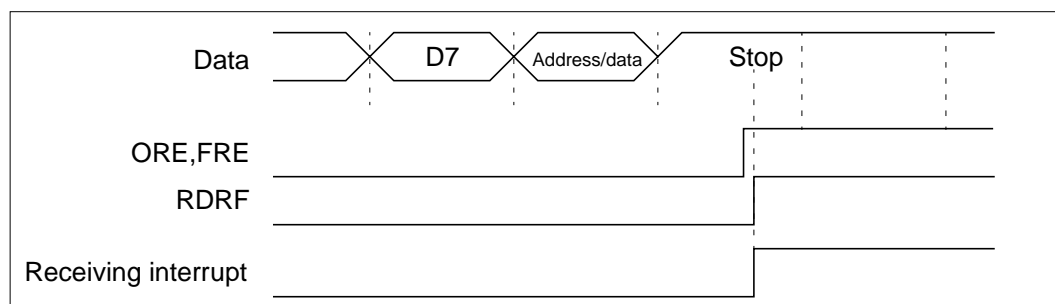


Figure 7.4.1-2 ORE, FRE, RDRF Flag Set Timing (Mode 1)

○ Mode 2 Receive

The ORE and RDRF flags are set and the interrupt request signal is sent to the CPU after the end of a receive transfer, when the final data (D7) is detected. If the ORE flag is active, the data in the SIDR0 register will be invalid.

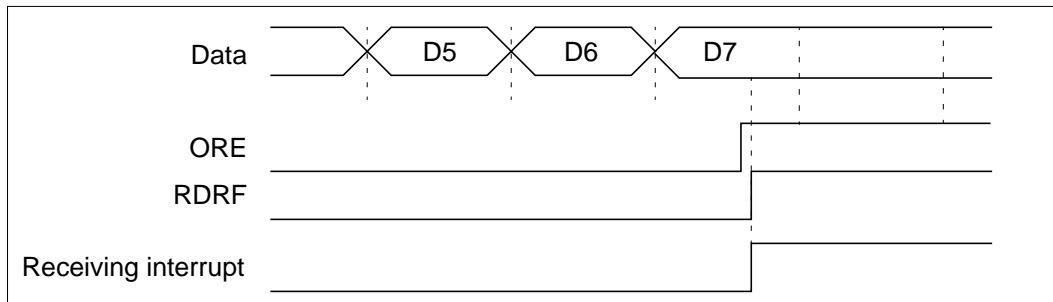


Figure 7.4.1-3 ORE, RDRF Flag Set Timing (Mode 2)

○ Mode 0, Mode 1, and Mode 2 Transmit

The TDRE flag is cleared when data is written to the SODR0 register. The TDRE flag is set (and an interrupt request sent to the CPU) as soon as the data in the SODR0 register is transferred to the internal shift register, to ready the SODR0 register for the next data write cycle. During a transmit operation, if '0' is written to the TXE bit in the SCR0 register (including the RXE bit in mode 2), the TDRE bit in the SSR0 register will be set to '1' and the UART0 transmit operation will be disabled as soon as the transmit shifter stops. The data written to the SODR0 register will be sent, however, between the writing of '0' to the TXE bit in the SCR0 register (including the RXE bit in mode 2), and the end of the transmit operation.

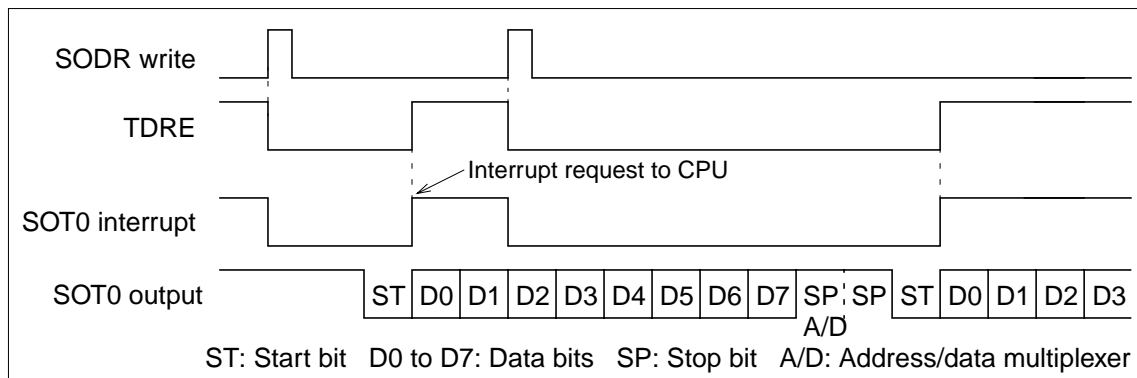


Figure 7.4.1-4 TDRE Flag Set Timing (Mode 0, 1)

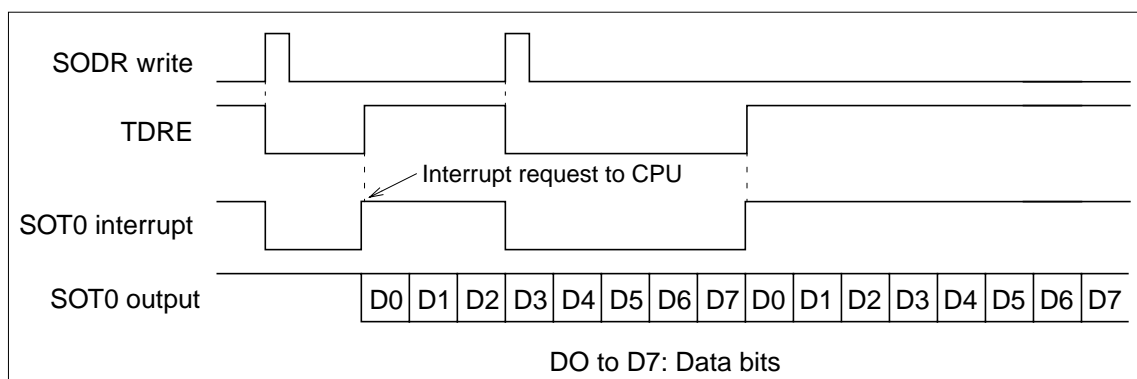


Figure 7.4.1-5 TDRE Flag Set Timing (Mode 2)

7.5 Negative clock operation

The MB90495 Series supports the negative clock operation of the UART0. In this operation, the shift clock signal is simply negated by a inverter. Therefore the definition of the shift clock signal in the proceeding sections of the UART0 is inverted from the logic low level to logic high level, from the negative edge to the positive edge and vice versa. This is the same for both the serial clock input and output.

The Edge Selector register is prepared for this purpose.

	7	6	5	4	3	2	1	0	
Address : 000025H	—	—	—	—	—	—	—	NEG	Initial value _____0B
SES0 (Serial Edge Select)									R/W

Table 7.5-1 Setting the NEG bit

NEG	Operation
0	Normal operation [default]
1	The shift clock signal is inverted

7.6 UART0 Sample Applications and Precautionary Information

This section presents a sample system configuration and communication flow chart as a sample application of the UART0 used in Mode 1.

■ UART0 Sample Application (System Configuration in Mode 1)

Mode 1 is used when one host CPU is connected to multiple slave CPU's. This UART0 resource supports only communication interface with the host-side unit.

Figure 7.6-1 shows a sample system configuration using Mode 1.

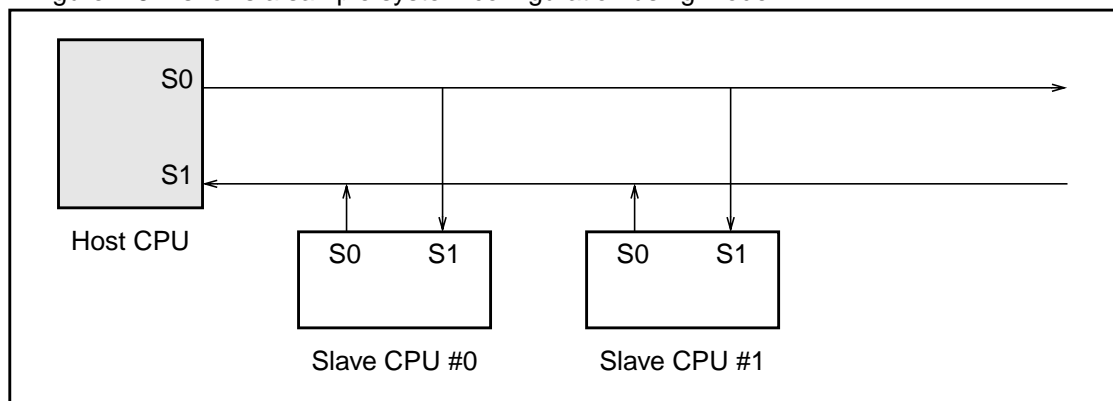


Figure 7.6-1 Sample System Configuration in Mode 1

■ UART0 Communication Flow Chart

Transmission begins with the transfer of address data by the host CPU. Address data is data handled while the A/D bit in the SCR0 register is set to '1' and is used to select the slave CPU that is to receive the transmission, and to enable communication with the host CPU. In normal data, the A/D bit in the SCR0 register is set to '0.' Figure 7.6-2 illustrates the flow of this process.

No parity check function is available in mode 2, so that the PEN bit in the SCR0 register should be set to '0.'

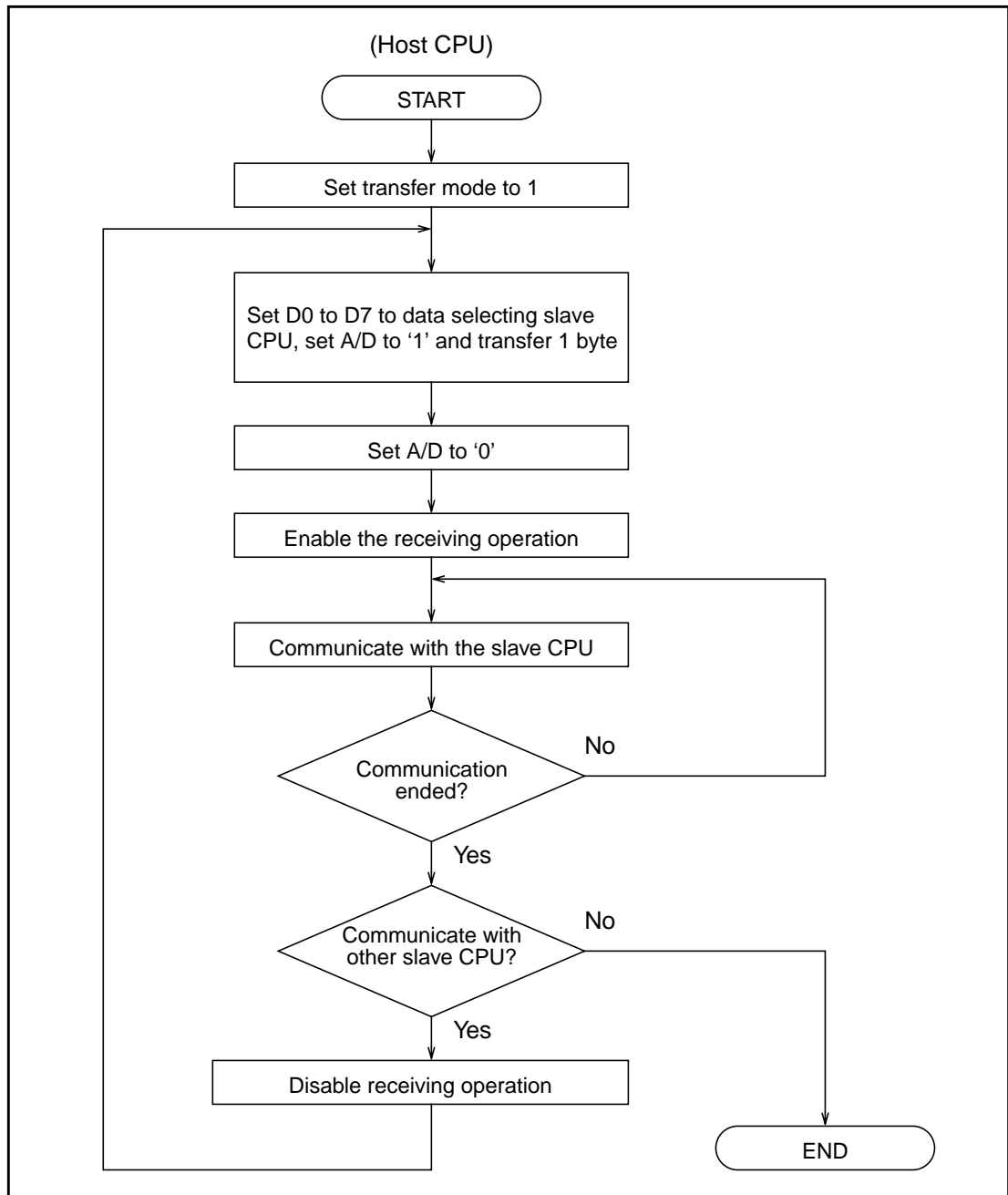


Figure 7.6-2 Communications Flowchart Using Mode 1

■ Extended Intelligent I/O Service (EI²OS)

For information about EI²OS, see section 21.6 “Interrupt of Extended Intelligent I/O Service (EI2OS)” .

■ Precautions for UART0 Use

Always make communications mode settings when the UART0 is not operating. Transmit and receive data values are not assured during mode setting.

CHAPTER 8 UART1

This chapter explains the functions and operation of UART1.

- 8.1 Overview of UART1 7
- 8.2 Configuration of UART1
- 8.3 UART1 Pins
- 8.4 UART1 Registers
- 8.5 UART1 Interrupts
- 8.6 UART1 Baud Rates
- 8.7 Operation of UART1
- 8.8 Notes on Using UART1
- 8.9 Sample Program for UART1

8.1 Overview of UART1 7

UART1 is a general-purpose serial data communication interface for performing synchronous or asynchronous (start-stop synchronization) communication with external devices. The UART1 has a normal bidirectional communication function (normal mode), additionally the master-slave communication function (multiprocessor mode) is only available for the master system.

■ UART1 Functions

● UART1 functions

UART1 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 8.1-1.

Table 8.1-1 UART1 functions

	Function
Data buffer	Full-duplex, double buffering
Transfer mode	<ul style="list-style-type: none">• Clock synchronous (using start and stop bits)• Clock asynchronous (start-stop synchronization)
Baud rate	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• 7 bits (in asynchronous normal mode only)• 8 bits
Signal mode	Non-return to zero (NRZ)
Reception error detection	<ul style="list-style-type: none">• Framing error• Overrun error• Parity error (cannot be detected in multiprocessor mode.)
Interrupt request	<ul style="list-style-type: none">• Reception interrupt (reception completion and reception error detection)• Transmission interrupt (transmission completion)• Extended intelligent I/O service (EI²OS) 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 UART1.

Table 8.1-2 UART1 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.

■ UART1 Interrupt and EI²OS

Table 8.1-3 UART1 interrupt and EI²OS

Interrupt cause	Interrupt number	Interrupt control register		Vector table address			EI ² OS
		Register name	Address	Lower	Upper	Bank	
UART1 reception interrupt	#37(25H)	ICR13	0000BD _H	FFFF68 _H	FFFF69 _H	FFFF6A _H	O
UART1 transmission interrupt	#38(26H)	ICR13	0000BD _H	FFFF64 _H	FFFF65 _H	FFFF66A	Δ

O : Provided with a function that detects a UART reception error and stops EI²OS

Δ : Usable when ICR13 or interrupt causes that share an interrupt vector are not used

8.2 Configuration of UART1

UART1 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 Register (SMR1)
- Control Register (SCR1)
- Status Register (SSR1)
- Input Data Register (SIDR1)
- Output Data Register (SODR1)

■ Block Diagram of UART1

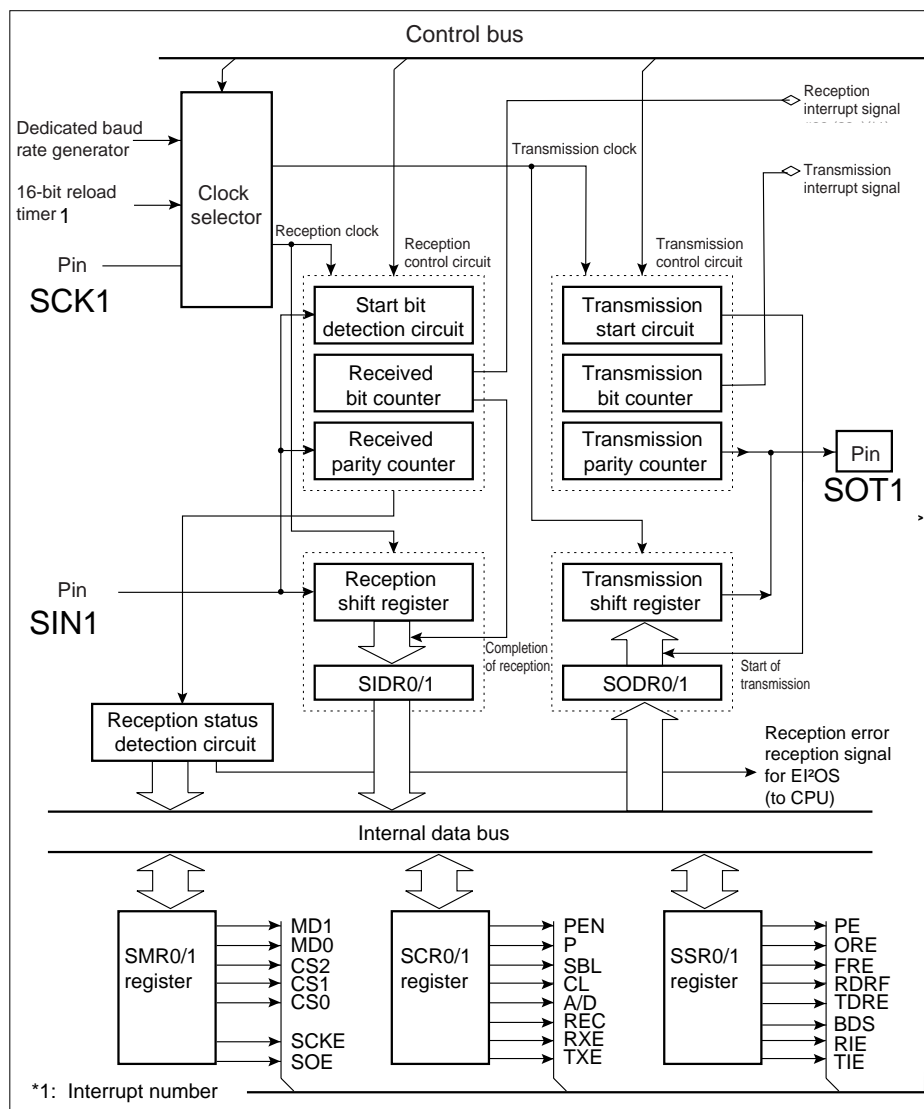


Figure 8.2-1 Block diagram of UART1

● 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 SDR1 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 SDR1. 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 SIN1 pin, shifting the data bit by bit. When reception is complete, the reception shift register transfers receive data to the SDR1 register.

● Transmission Shift Register

The transmission shift register transfers data written to the SDR1 register to itself and outputs the data to the SOT1 pin, shifting the data bit by bit.

● Serial Mode Register 1 (SMR1)

This register performs the following operations:

- Selecting a UART1 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

● **Control Register 1 (SCR1)**

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

● **Status Register 1 (SSR1)**

This register checks the transmission and reception status and error status, and enables and disables transmission and reception interrupt requests.

● **Input Data Register 1 (SIDR1)**

This register retains receive data. Serial input data is converted and stored in this register.

● **Output Data Register 1 (SODR1)**

This register sets transmission data. Data written to this register is converted to serial data and output.

8.3 UART1 Pins

This section describes the UART1 pins and provides a pin block diagram.

■ UART1 Pins

The UART1 pins also serve as general ports. Table 8.3-1 lists the pin functions, I/O formats, and settings required to use UART1.

Table 8.3-1 UART1 pins

Pin name	Pin function	I/O format	Pull-up	Standby control	Setting required to use pin
P40/SIN1	Port 4 I/O or serial data input	CMOS output and CMOS hysteresis input	Nothing	Provided	Set as an input port (DDR4: bit0 = 0)
P42/SOT1	Port 4 I/O or serial data output				Set to output enablemode (SMR1: SOE = 1)
P41/SCK1	Port 4 I/O or serial clock input/output				Set as an input port when a clock is input (DDR4: bit 1=0)
					Set to output enable mode when a clock is output (SMR1: SCKE = 1)

■ Block Diagram of UART1 Pins

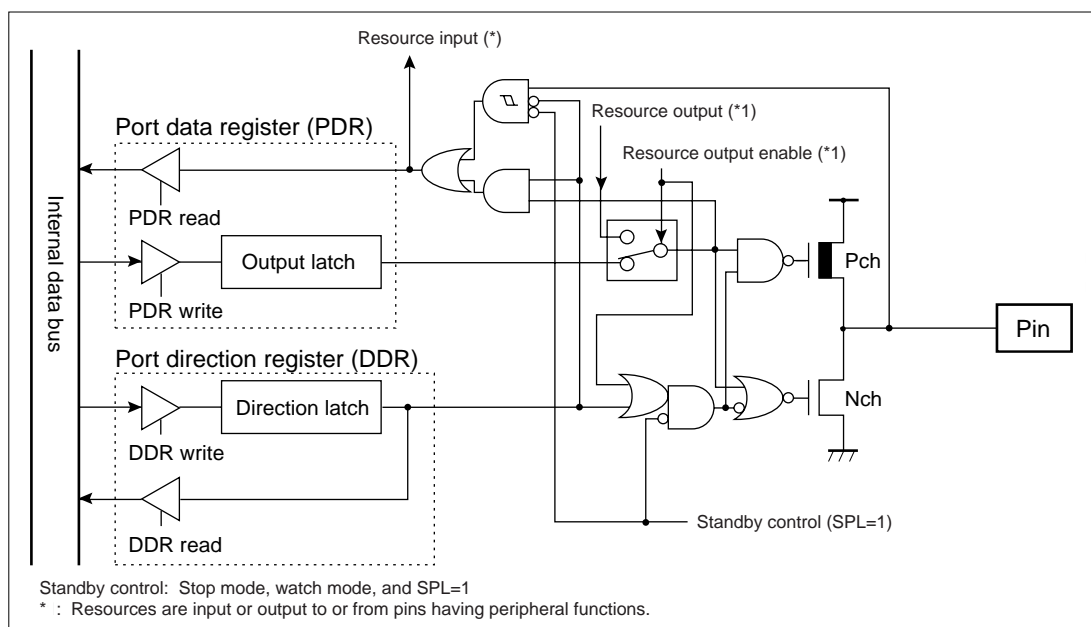


Figure 8.3-1 Block Diagram of UART1 Pins

8.4 UART1 Registers

The following figure shows the UART1 registers.

■ UART1 Registers

Address	bit15	bit8	bit7	bit0
27H,26H	SCR1 (control register 1)		SMR1 (mode register 1)	
29H,28H	SSR1 (status register 1)		SIDR1/SODR1 (input/output data register 1)	
2BH	CDCR1 (communication prescaler control register 1)		Reserved	

Figure 8.4-1 UART1 registers

8.4.1 Serial Control Register 1 (SCR1)

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.

■ Control Register (SCR1)

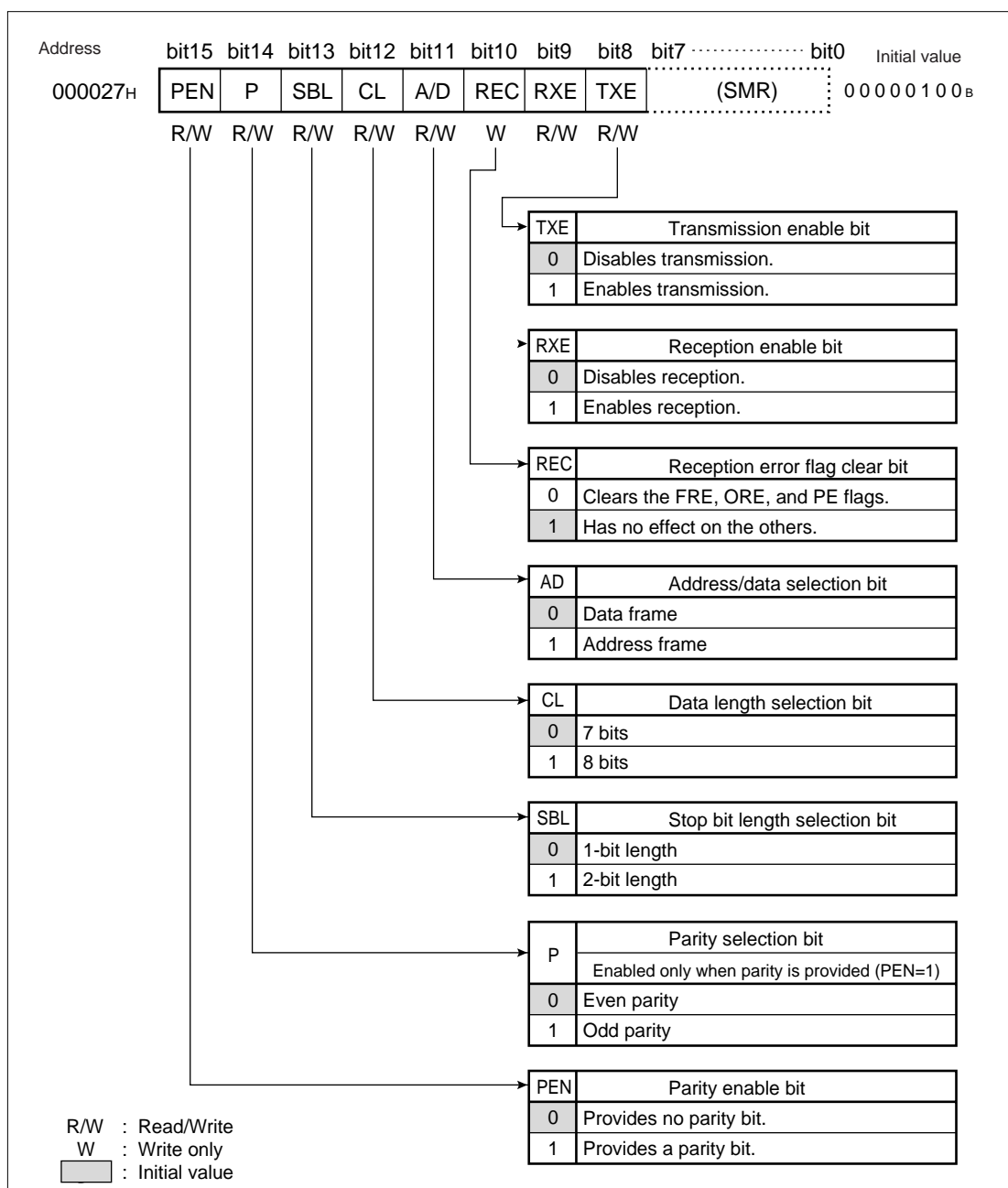


Figure 8.4.1-1 Control register 1 (SCR1)

8.4 UART1 Registers

Table 8.4.1-1 Functions of each bit of control register 1 (SCR1)

Bit name		Function
bit15	PEN: Parity enable bit	This bit selects whether to add a parity bit during transmission in serial data input-output mode or to detect it during reception. <Caution> No parity can be used in operation modes 1 and 2. Therefore, 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 (multiprocessor mode) and operation mode 2 (synchronous).
bit11	A/D: Address/data selection bit	<ul style="list-style-type: none"> Specify the data format of a frame to be sent or received in multiprocessor mode (mode 1). Select usual data when this bit is 0, and select address data when the bit is 1.
bit10	REC: Reception error flag clear bit	<ul style="list-style-type: none"> 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 UART1 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	<ul style="list-style-type: none"> This bit controls UART1 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	<ul style="list-style-type: none"> This bit controls UART1 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.

8.4.2 Serial Mode Register 1 (SMR1)

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 Register 1 (SMR1)

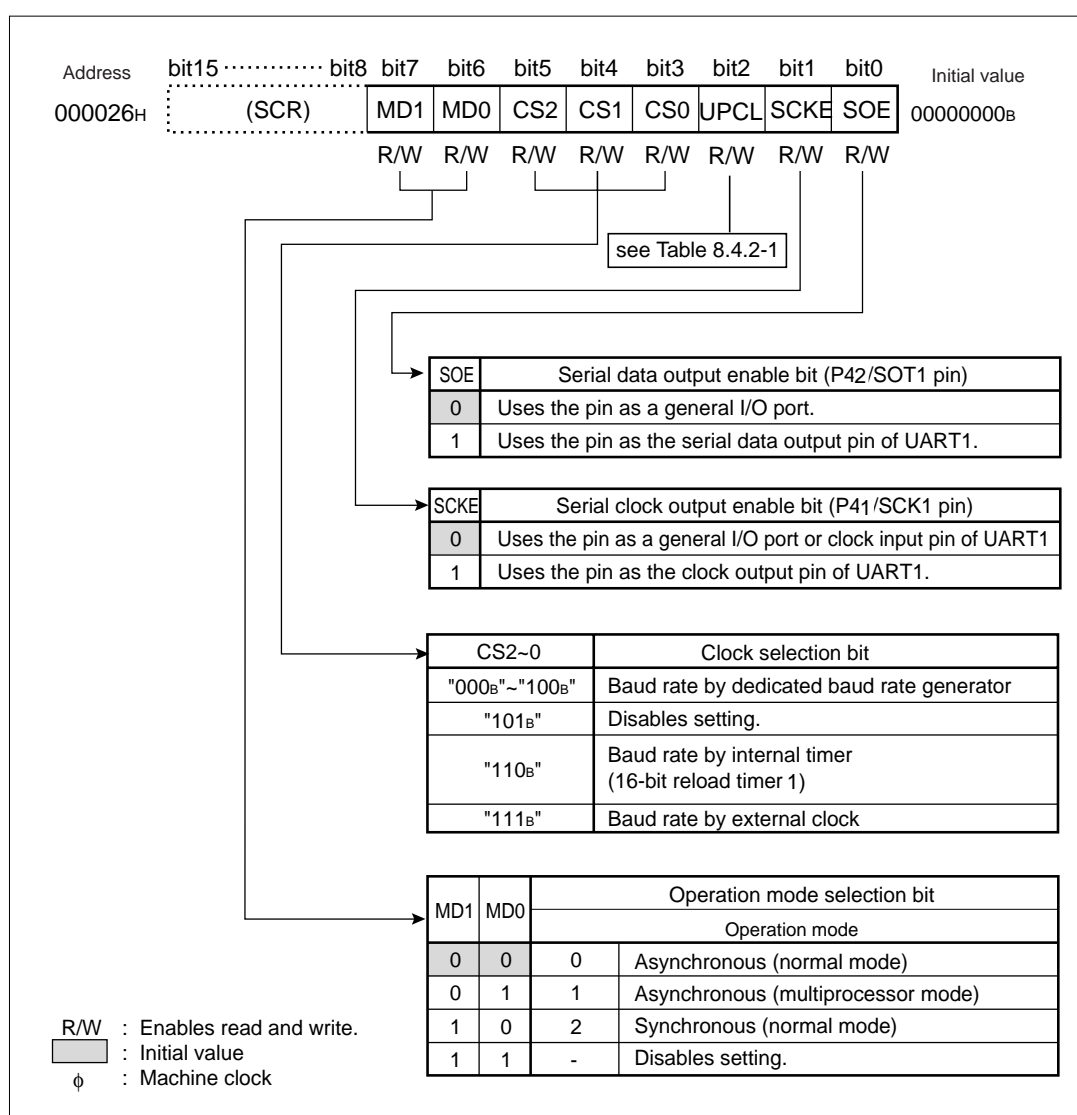


Figure 8.4.2-1 Serial Mode register 1 (SMR1)

8.4 UART1 Registers

Table 8.4.2-1 Serial Mode register 1 (SMR1)

Bit name		Function
bit7 bit6	MD1 and MD0: Operation mode selection bits	<p>These bits select an operation mode.</p> <p><Caution> Operation mode 1 (multiprocessor mode) can be used only from the master system during master-slave communication. UART1 cannot be used from the slave system because it has no address/data detection function during reception. .</p>
bit5 bit4 bit3	CS2 to CS0: Clock selection bits	<ul style="list-style-type: none"> • 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 eight in synchronous transfer mode. • Input clocks can be selected from external clocks (SCK pin), 16-bit reload timer 0, and the dedicated baud rate generator.
bit2	UPCL: UART programmable clear	<ul style="list-style-type: none"> • If '1' is written to this bit, all register of UART1 are reseted. Writing '0' has no effect.
bit1	SCKE: Serial clock output enable bit	<ul style="list-style-type: none"> • This bit controls the serial clock input-output ports. • When this bit is 0, the P41/SCK1 pins operate as general input-output ports (P41) or serial clock input pins. When this bit is 1, the pins operate as serial clock output pins. <p><Caution></p> <ul style="list-style-type: none"> • When using the P41/SCK1 pins as serial clock input (SCKE=0) pins, set the P41 as input port. Also, select external clocks (SMR1: 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 SMC: CS2 to CS0 = 111B). <p><Reference> When the SCK1 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.</p>
bit0	SOE: Serial data output enable bit	<ul style="list-style-type: none"> • This bit enables or disables the output of serial data. • When this bit is 0, the P42/SOT1 pin operates as general input-output ports (P42). When this bit is 1, the P42/SOT1 pin operates as serial data output pins (SOT1). <p><Reference> When serial data is output (SOE=1), the enabled, the P42/SOT1 pin functions as SOT1 pins regardless of the status of general input-output ports (P42)</p>

8.4.3 Status Register 1 (SSR1)

This register checks the transmission and reception status and error status, and enables and disables the transmission and reception interrupts.

■ Status Register (SSR1)

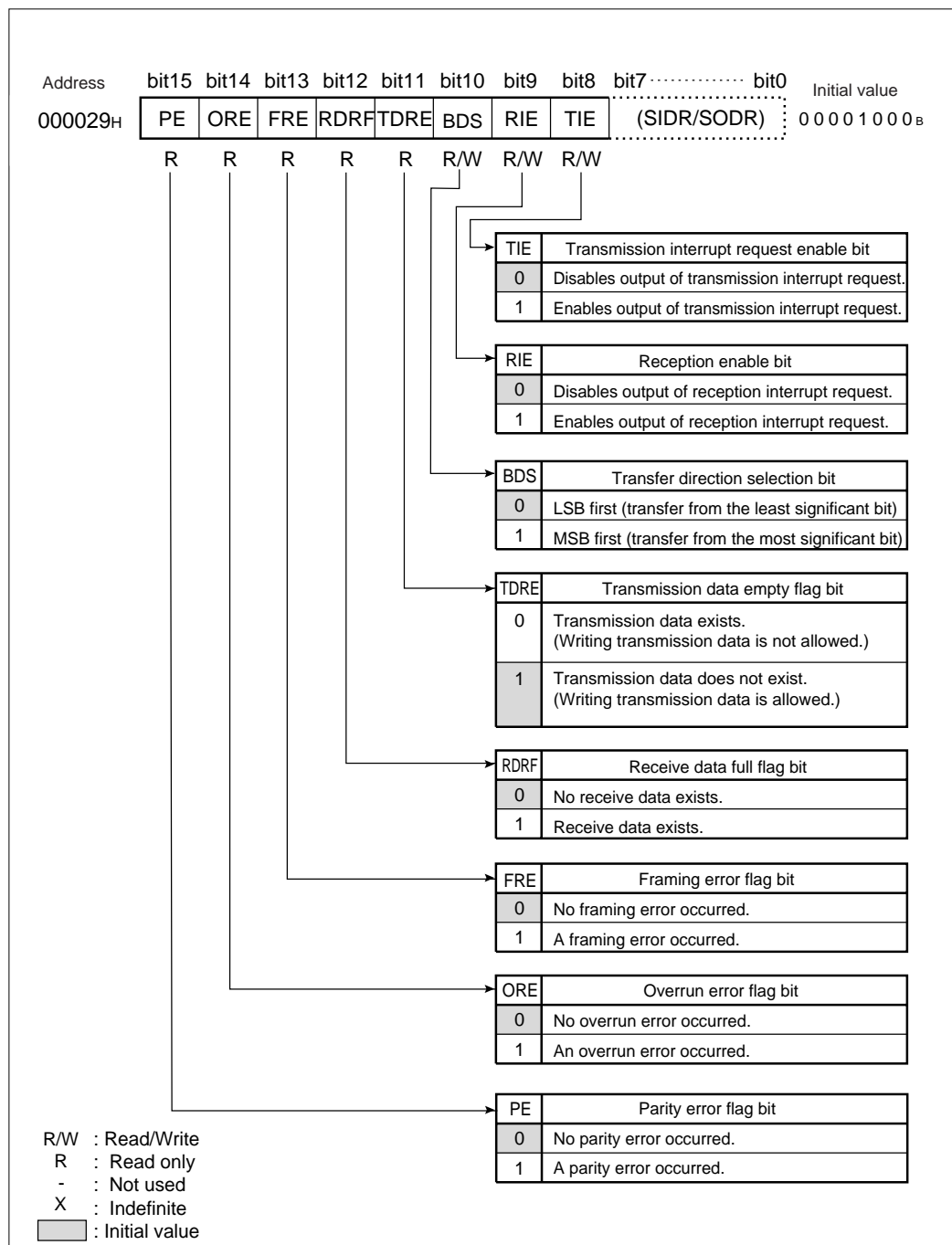


Figure 8.4.3-1 Status register 1 (SSR1)

8.4 UART1 Registers

Table 8.4.3-1 Functions of each bit of status register 1 (SSR1)

NO.	Bit name	Function
bit15	PE: Parity error flag bit	<ul style="list-style-type: none"> This bit is set to 1 when a parity error occurs during reception and is cleared when 0 is written to the RFC bit of the serial mode register1 (SMR1). A reception interrupt request is output when this bit and the RIE bit are 1. Data in input data register (SIDR1) is invalid when this flag is set.
bit14	ORE: Overrun error flag bit	<ul style="list-style-type: none"> This bit is set to 1 when an overrun error occurs during reception and is cleared when 0 is written to the RFC bit of the serial mode register 1 (SMR1). A reception interrupt request is output when this bit and the RIE bit are 1. Data in the input data register (SIDR1) is invalid when this flag is set.
bit13	FRE: Framing error flag bit	<ul style="list-style-type: none"> This bit is set to 1 when a framing error occurs during reception and is cleared when 0 is written to the RFC bit of the serial mode register 1 (SMR1). A reception interrupt request is output when this bit and the RIE bit are 1. Data in the input data register (SIDR1) is invalid when this flag is set.
bit12	RDRF: Receive data full flag bit	<ul style="list-style-type: none"> This flag indicates the status of the input data register (SIDR1). This bit is set to 1 when receive data is loaded into SIDR1 and is cleared to 0 when input data register SIDR1 is read. A reception interrupt request is output when this bit and the RIE bit are 1.
bit11	TDRE: Transmission data empty flag bit	<ul style="list-style-type: none"> This flag indicates the status of output data register (SODR1). This bit is cleared to 0 when transmission data is written to SODR1 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. <p><Caution> This bit is set to 1 (SODR1 empty) as its initial value.</p>
bit10	BDS: Transfer direction selection bit	<ul style="list-style-type: none"> 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). <p><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 SIDR1 register, the data becomes invalid.</p>
bit9	RIE: Reception interrupt request enable bit	<ul style="list-style-type: none"> This bit enables or disables input of a request for transmission interrupt to the CPU. A reception interrupt request is output 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	<ul style="list-style-type: none"> This bit enables or disables output of a request for transmission interrupt to the CPU. A transmission interrupt request is output when this bit and the TDRE bit are 1.

8.4.4 Input Data Register 1 (SIDR1) and Output Data Register 1 (SODR1)

The input data register (SIDR1) is a serial data reception register. The output data register (SODR1) is a serial data transmission register. Both SIDR1 and SODR1 registers are located in the same address.

■ Input Data Register 1 (SIDR1)

Figure 8.4.4-1 shows the bit configuration of input data register 1.

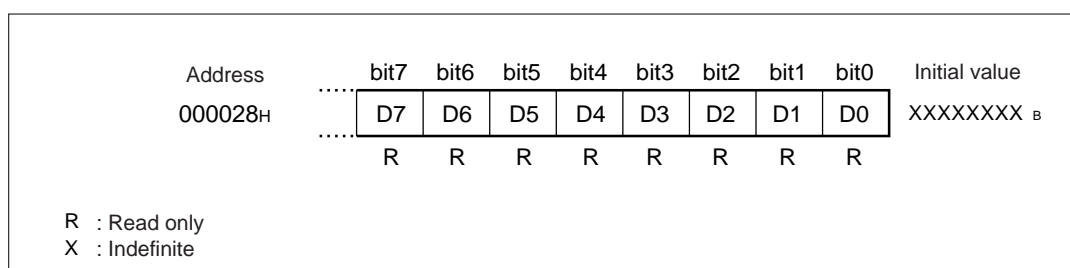


Figure 8.4.4-1 Input data register 1 (SIDR1)

SIDR1 is a register that contains receive data. The serial data signal transmitted to the SIN1 pin is converted in the shift register and 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 (SSR1: RDRF) is set to 1. If a reception interrupt request is enabled at this point, a reception interrupt occurs.

Read SIDR1 when the RDRF bit of the status register 1 (SSR1) is 1. The RDRF bit is cleared automatically to 0 when SIDR1 is read.

Data in SIDR1 is invalid when a reception error occurs (SSR1: PE, ORE, or FRE = 1).

■ Output Data Register 1 (SODR1)

Figure 8.4.4-2 shows the bit configuration of the output data register 1.

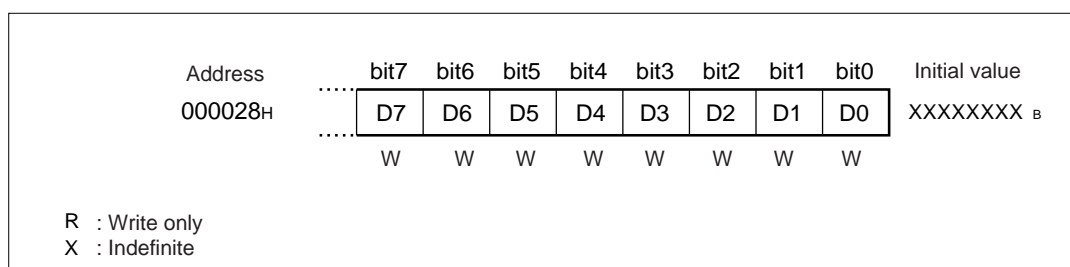


Figure 8.4.4-2 Output data register 1 (SODR1)

8.4 UART1 Registers

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 from the serial data output terminal (SOT1 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 (SSR1: 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 piece of transmission data can be written. If output transmission interrupt requests have been enabled, a transmission interrupt is generated. Write the next piece of transmission data when a transmission interrupt is generated or the TDRE bit is 1.

<Check>

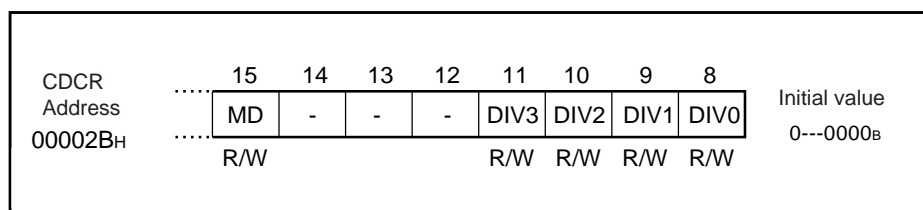
SODR1 is a write-only register and SIDR1 is a read-only register. These registers are located in the same address, so the read value is different from the write value. Therefore, instructions that perform a read-modify-write (RMW) operation, such as the INC/DEC instruction, cannot be used.

8.4.5 Communication Prescaler Control Register 1 (CDCR1)

This register controls the division of machine clocks.

■ Communication Prescaler Control Register 1 (CDCR1)

The operation clocks of UART1 can be obtained by dividing machine clocks. UART1 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 CDCR1 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 8.4.5-1.

Table 8.4.5-1 Communication prescaler

MD	DIV3	DIV2	DIV1	DIV0	div
0	—	—	—	—	Setting not allowed
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.

8.5 UART1 Interrupts

UART1 uses both reception and transmission interrupts. An interrupt request can be generated for either of the following causes:

- Receive data is set in the input register (SIDR1), or a reception error occurs.
- Transmission data is transferred from output data register 1 (SODR1) to the transmission shift register.

The extended intelligent I-O service (EI²OS) is available for these interrupts.

■ UART1 Interrupts

Table 8.5-1 lists the interrupt control bits and causes of UART1.

Table 8.5-1 Interrupt control bits and interrupt causes of UART1

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 (SIDR)	SSR: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 transmis- sion buffer (SODR)	SSR: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 (SSR1: RDRF)
- Overrun error (SSR1: ORE)
- Framing error (SSR1: FRE)
- Parity error (SSR1: PE)

When at least one of the flag bits is 1 and the reception interrupts are enabled (SSR1: RIE=1), a reception interrupt request is output to the interrupt controller.

When the input data register (SIDR) is read, the receive data full flag (SSR1: RDRF) is automatically cleared to 0. When 0 is written to the REC bit of the control register (SCR1), all the reception error flags (SSR1: PE, ORE, and FRE) are cleared to 0.

● Transmission Interrupt

When transmission data is transferred from the output data register (SODR1) to the transfer shift register, the TDRE bit of the status register (SSR1) is set to 1. When the transmission interrupts have been enabled (SSR1: TIE=1), a transmission interrupt request is output to the interrupt controller.

■ UART1 Interrupts and EI²OS

Table 8.5-2 UART1 interrupts and EI²OS

Interrupt cause	Interrupt number	Interrupt control register		Vector table address			EI OS
		Register name	Address	Lower	Upper	Bank	
UART1 reception interrupt	#37(25H)	ICR13	0000BDH	FFFF68H	FFFF69H	FFFF6AH	O
UART1 transmission interrupt	#38(26H)	ICR13	0000BDH	FFFF64H	FFFF65H	FFFF66H	Δ

O : Provided with a function that detects a UART1 reception error and stops EI²OS

Δ : Usable when interrupt causes that share the ICR13 and ICR14 or the interrupt vectors are not used

■ UART1 EI²OS Functions

UART1 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

UART1 shares the interrupt registers with the UART1 reception interrupts. Therefore, EI²OS can be started up only when no UART1 reception interrupts are used.

8.5.1 Reception Interrupt Generation and Flag Set Timing

The following are the reception interrupt causes: completion of reception (SSR1: RDRF) and occurrence of a reception error (SSR1: PE, ORE, or FRE).

■ Reception Interrupt Generation and Flag Set Timing

Receive data is stored in input data register 1 (SIDR1) 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 (SSR1: PE, ORE, and FRE) are set, then the receive data flag (SSR1: RDRF) is set to 1. If one of the error flags is 1 in each mode, the SIDR1 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.

Figure 8.5.1-1 below shows the reception operation and flag set timing.

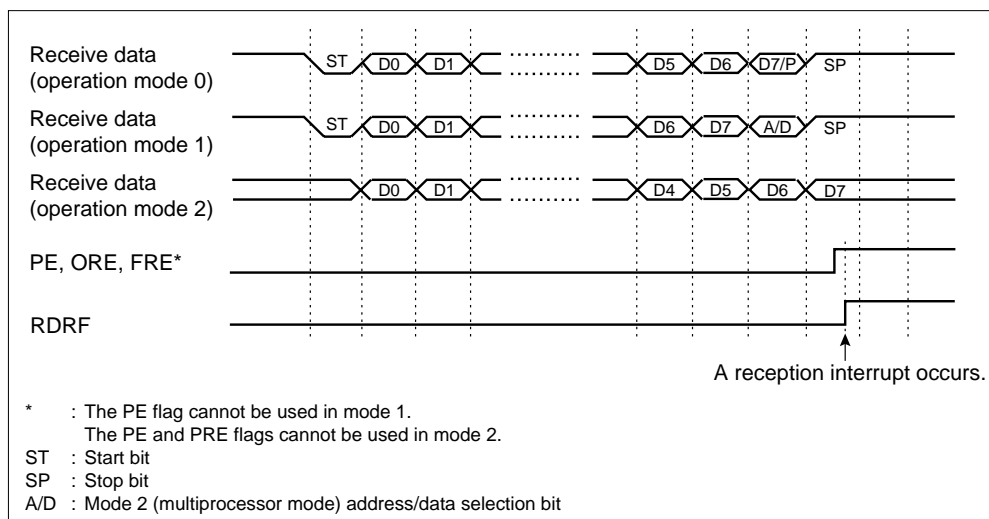


Figure 8.5.1-1 Reception operation and flag set timing

● Reception Interrupt Generation Timing

When the RDRF, PE, ORE, or FRE flag is set to 1 in the reception interrupt enable state (SSR1: RIE=1), reception interrupt requests (#37) are generated.

8.5.2 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 (SODR1).

■ Transmission Interrupt Generation and Flag Set Timing

The transmission data empty flag bit (SSR1: TDRE) is set to 1 when data written to the output data register (SODR1) is transferred to the transmission shift register, and the next piece of data is ready to be written. TDRE is cleared to 0 when transmission data is written to SODR1.

Figure 8.5.2-1 shows the transmission operation and flag set timing.

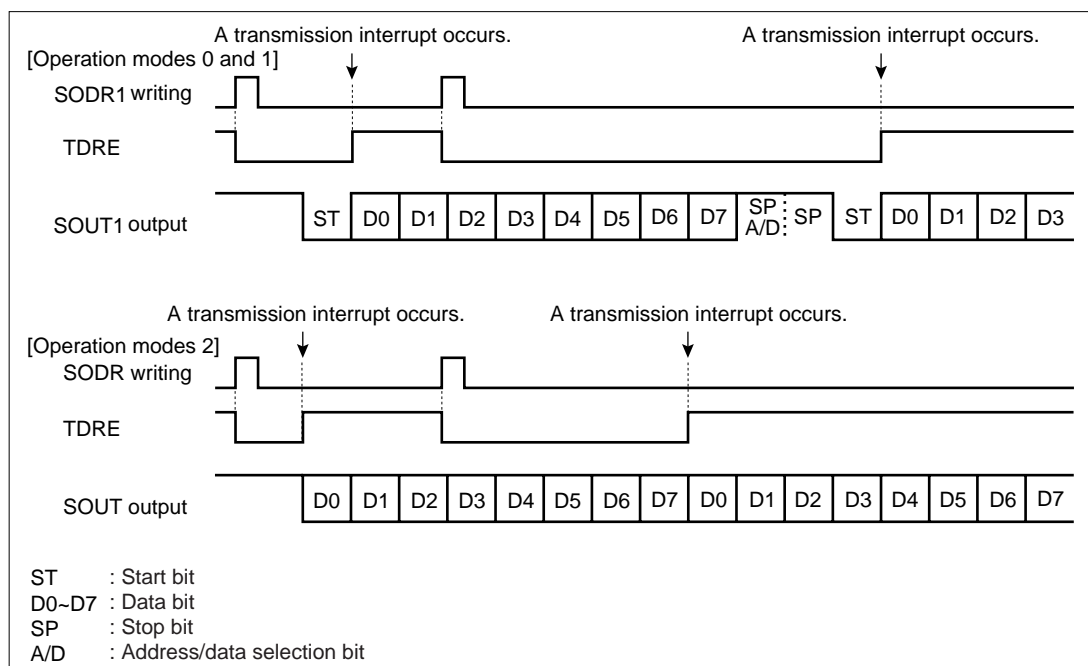


Figure 8.5.2-1 Transmission operation and flag set timing

● Transmission Interrupt Request Generation Timing

If the TDRE flag is set to 1 when a transmission interrupt is enabled (SSR1: TIE=1), transmission interrupt requests 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 output data register (SODR1). Carefully specify the transmission interrupt enable timing.

8.6 UART1 Baud Rates

One of the following can be selected as the UART1 transmitting/receiving block:

- **Dedicated baud rate generator**
 - **Internal clock (16-bit reload timer 0)**
 - **External clock (clock input to the SCK1 pin)**
-

■ UART1 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**

UART1 has an internal dedicated baud rate generator. One of eight baud rates can be selected using the serial mode register (SMR1).

An asynchronous or clock synchronous baud rate is selected using the machine clock frequency and the BCH and CS2 to CS0 bits of the serial mode register (SMR1) .

● **Baud Rates Determined Using the Internal Timer**

The internal clock supplied from 16-bit reload timer 0 is used as is (synchronous) or by dividing it by 16 (asynchronous) for the baud rate. Any baud rate can be set by setting the reload value.

● **Baud Rates Determined Using the External Clock**

The clock input from the UART1 clock pulse input pins (SCK1/P41) is used as is (synchronous) or by dividing it by 16 (asynchronous) for the baud rate. Any baud rate can be set externally.

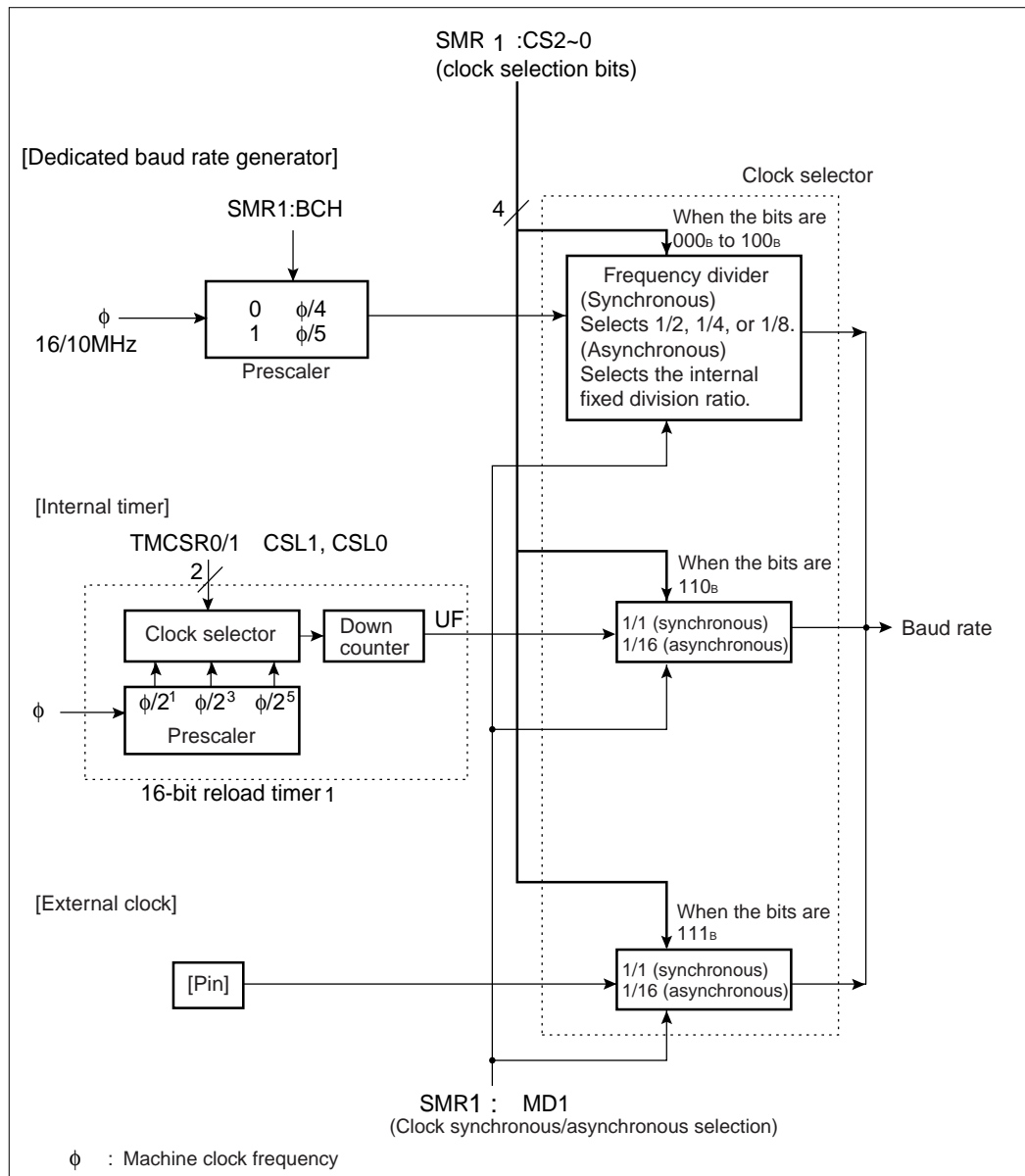


Figure 8.6-1 baud rate selection circuit

8.6.1 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 UART1 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 = $\phi \times (\text{prescaler division ratio}) \times (\text{asynchronous transfer clock division ratio})$

synchronous baud rate = $\phi \times (\text{prescaler division ratio}) \times (\text{synchronous transfer clock division ratio})$

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 CDCR1 register as listed in Table 8.6.1-1.

Table 8.6.1-1 Selection of each division ratio for the machine clock prescaler

MD	DIV3	DIV2	DIV1	DIV0	div
0	—	—	—	—	Setting not allowed
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

8.6 UART1 Baud Rates

● Synchronous Transfer Clock Division Ratios

A division ratio for synchronous baud rates is selected using the CS2 to CS0 bits of the serial mode register 1 (SMR1) as listed in Table 8.6.1-2.

Table 8.6.1-2 Selection of synchronous baud rate division ratios

CS2	CS1	CS0	Division ratio for CLK synchronization	Calculation formula	SCKI
0	0	0	16M	$(\phi \div \text{div})/1$	$(\phi \div \text{div})/1$
0	0	1	8M	$(\phi \div \text{div})/2$	$(\phi \div \text{div})/2$
0	1	0	4M	$(\phi \div \text{div})/4$	$(\phi \div \text{div})/4$
0	1	1	2M	$(\phi \div \text{div})/8$	$(\phi \div \text{div})/8$
1	0	0	1M	$(\phi \div \text{div})/16$	$(\phi \div \text{div})/16$
1	0	1	500K	$(\phi \div \text{div})/32$	$(\phi \div \text{div})/32$

The division ratio is calculated supposing that machine cycle $\phi = 16$ MHz and $\text{div} = 4$.

● Asynchronous Transfer Clock Division Ratios

A division ratio for asynchronous baud rates is selected using the CS2 to CS0 bits of the serial mode register 1 (SMR1) as listed in Table 8.6.1-3.

Table 8.6.1-3 Selection of synchronous baud rate division ratios

CS2	CS1	CS0	Asynchronous(start-stop synchronization)	Calculation formula	SCKI
0	0	0	76923	$(\phi \div \text{div})/(8 \times 13 \times 2)$	$(\phi \div \text{div})/(13 \times 1)$
0	0	1	38461	$(\phi \div \text{div})/(8 \times 13 \times 4)$	$(\phi \div \text{div})/(13 \times 2)$
0	1	0	19230	$(\phi \div \text{div})/(8 \times 13 \times 8)$	$(\phi \div \text{div})/(13 \times 4)$
0	1	1	9615	$(\phi \div \text{div})/(8 \times 13 \times 16)$	$(\phi \div \text{div})/(13 \times 8)$
1	0	0	500K	$(\phi \div \text{div})/(8 \times 2 \times 2)$	$(\phi \div \text{div})/2$
1	0	1	250K	$(\phi \div \text{div})/(8 \times 13 \times 4)$	$(\phi \div \text{div})/4$

The division ratio is calculated supposing that machine cycle $\phi = 16$ MHz and $\text{div} = 1$.

● Internal Timer

When CS2 to CS0 are set to 110_B and the internal timer is selected, the formulas for calculating baud rates (when using the reload timer) are as follows:

Asynchronous (start-stop synchronization): $(\phi \div N)/(16 \times 2 \times (n + 1))$
 CLK synchronization: $(\phi \div N)/(2 \times (n + 1))$

N: Timer count clock source

n: Timer reload value

<Check>

In mode 2 (CLK synchronization mode), SCK1 is up to three clocks later than CSK1. 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_B 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.

8.6.2 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 UART1 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 serial mode register (SMR1) 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. Table 8.6.2-1 shows the baud rate selection circuit for the internal timer.

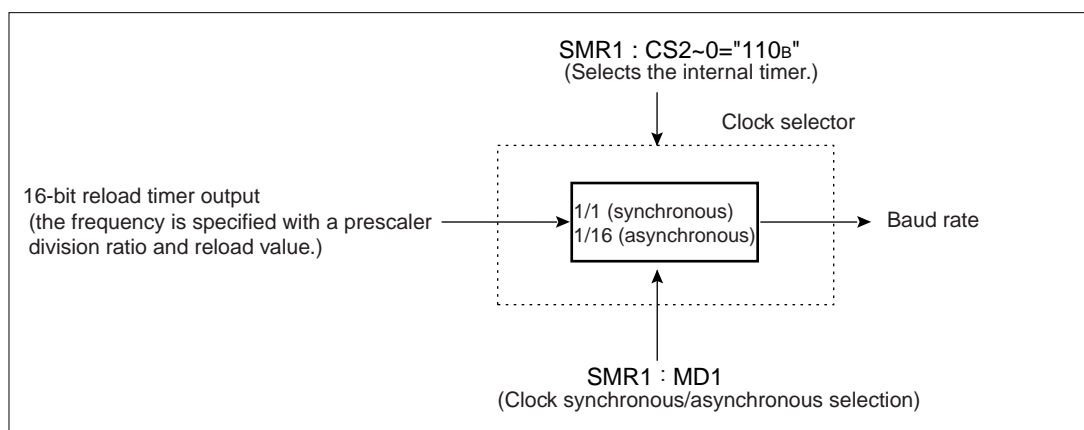


Figure 8.6.2-1 Baud rate selection circuit for the internal timer (16-bit reload timer 0)

● Baud Rate Calculation Formulas

Asynchronous baud rate = ϕ bps

$$X(n+1) \times 2 \times 16$$

Synchronous baud rate = ϕ bps

$$X(n+1) \times 2$$

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)

8.6 UART1 Baud Rates

● Examples of Setting Reload Values (Machine Clock: 7.3728 MHz)

Table 8.6.2-1 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

8.6.3 Baud Rates Determined Using the External Clock

This section describes the settings used when the external clock is selected as the UART1 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 register (SMR1) to select the baud rate determined by using the external clock input.
- Set the SCK1/P41 pins as input ports (DDR4: bit 1 = 0).
- Write 0 to the SCKE bit of the serial mode register (SMR1) to set the pin as an external clock input pin.

As shown in Figure 8.6.3-1, 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.

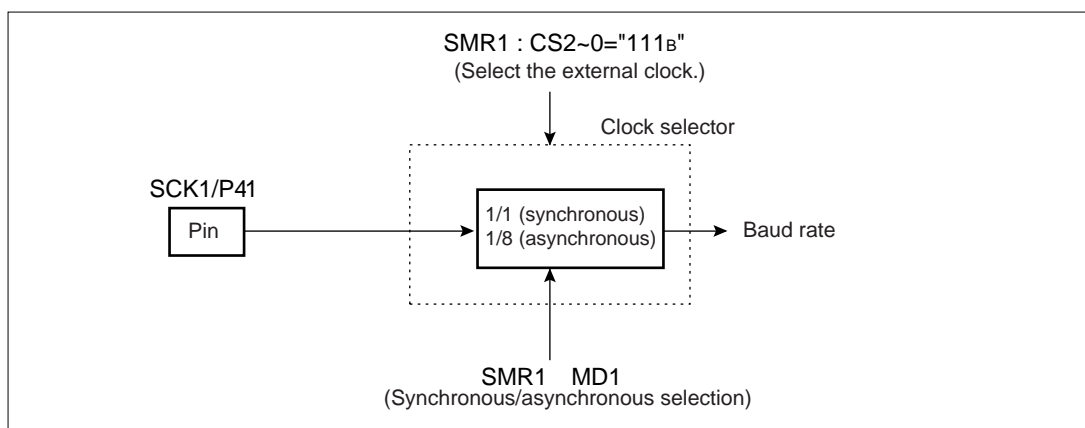


Figure 8.6.3-1 Baud rate selection circuit for the external clock

● Baud Rate Calculation Formulas

asynchronous baud rate = $f/16$

synchronous baud rate = f

f: External clock frequency (up to 2 MHz)

8.7 Operation of UART1

UART1 operates in operation modes 0 and 2 for normal bidirectional serial communication and in operation mode 1 for master-slave communication.

■ Operation of UART1

● Operation modes

There are three UART1 operation modes: modes 0 to 2. As listed in Table 8.7-1, an operation mode can be selected according to the inter-CPU connection method and data transfer mode.

Table 8.7-1 UART1 operation mode

Operation mode		Data length		Synchronizati on mode	Stop bit length
		When parity is disabled	When parity is enabled		
0	Normal mode	7 or 8 bits		Asynchronous	1 or 2 bits *2
1	Multiprocessor	8+1*1	—	Asynchronous	
2	Normal mode	8	—	Synchronous	None

— : Setting not possible

*1 : "+1" indicates the address/data selection bit (A/D) for communication control.

*2 : During reception, only one stop bit can be detected.

<Check>

Operation mode 1 of UART1 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 Mode

UART1 can treat data only in non-return to zero (NRZ) format.

● Operation Enable Bit

UART1 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 input data register (SIDR1). Then stop the reception operation.
- If the transmission operation is disabled during transmission (data is output from the transmission shift register), wait until there is no data in the output data register (SODR1) before stopping the transmission operation.

8.7.2 Operation in Asynchronous Mode (Operation Modes 0 and 1)

When UART1 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 8.7.2-1 shows the data format in asynchronous mode.

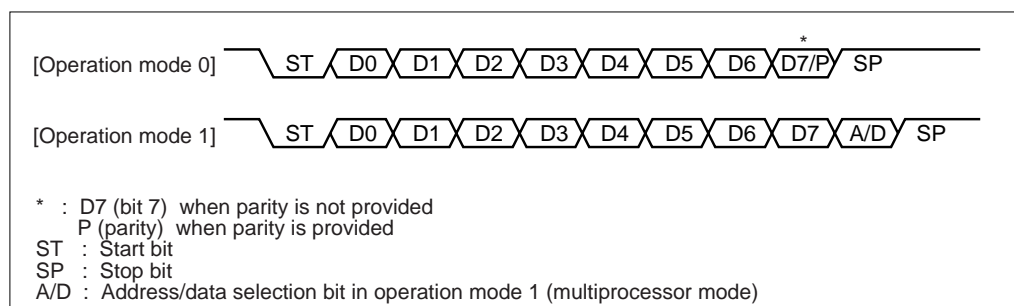


Figure 8.7.2-1 Transfer data format (operation modes 0 and 1)

● Transmission Operation

Transmission data is written to the output data register (SODR1) when the transmission data empty flag bit (SSR1: TDRE) is 1. This data is transmitted if the transmission operation is enabled (SCR1: 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 piece of transmission data gets ready to be set. At this point, a transmission interrupt request is output requesting that the next piece of transmission data be set in the SODR1 register if that request is enabled (SSR1: TIE=1). The TDRE flag is cleared to 0 when the transmission data is written to SODR1.

● Reception Operation

Reception operation is performed every time it is enabled (SCR1: RXE=1). When a start bit is detected, a frame of data is received according to the data format specified by the control register 1 (SCR1). After the frame has been received, the error flag is set if an error occurs, then the receive data full flag bit (SSR1: RDRF) is set to 1. At this point, a reception interrupt request is output if it is enabled (SSR1: TIE=1).

8.7 Operation of UART1

Check each flag of the input data register (SIDR1). If the reception is normal, read the input data register 1 (SIDR1). If an error is found, proceed to error handling. The RDRF flag is cleared to 0 every time receive data is read from SIDR1.

● 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 control register (SCR1). Even or odd parity can also be specified using the P bit of the control register (SCR1). In operation mode 1 (asynchronous, multiprocessor mode) and operation mode 2 (synchronous, normal mode), parity cannot be used. Figure 8.7.2-2 shows both transmission and receive data when parity is enabled.

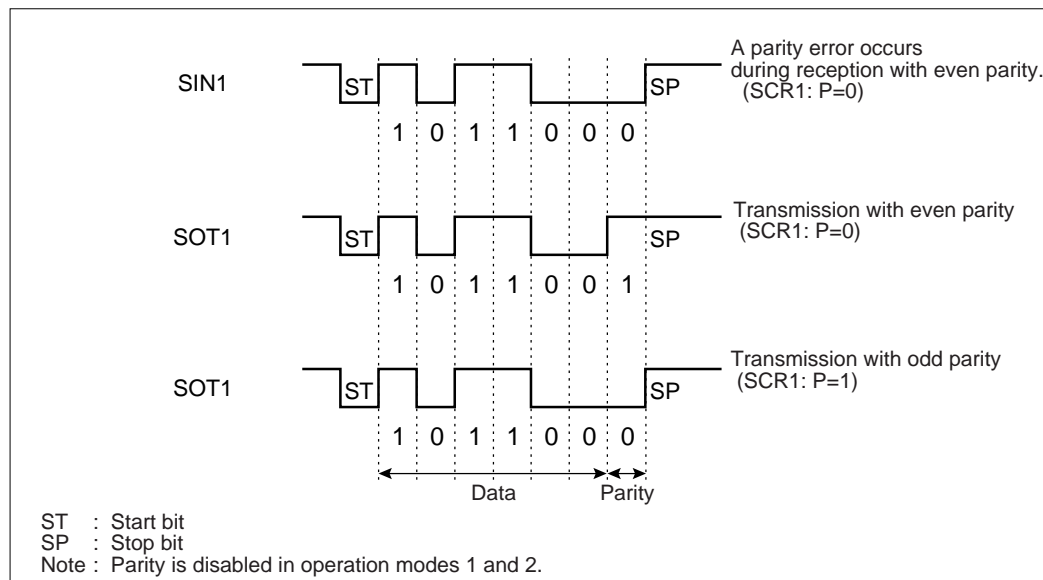


Figure 8.7.2-2 Transmission data when parity is enabled

8.7.3 Operation in Synchronous Mode (Operation Mode 2)

The clock synchronous transfer method is used for UART1 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 8.7.3-1 shows the data format in clock synchronous mode.

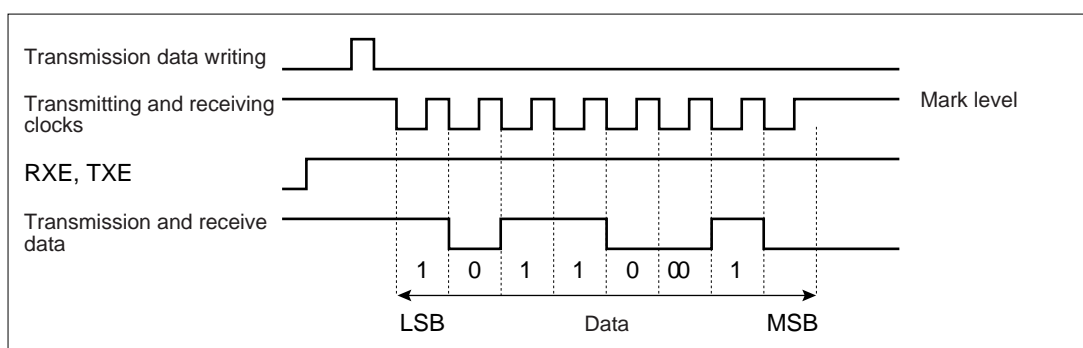


Figure 8.7.3-1 Transfer data format (operation mode 2)

● Clock Supply

In clock synchronous mode (I/O extended serial), as many clocks 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 UART1 output data register (SODR1) contains data (SSR1: 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 register (SMR1)]

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)

8.7 Operation of UART1

SOE: 1 for transmission; 0 for reception only

[Control register (SCR1)]

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.

[Status register (SSR1)]

RIE: 1 when using interrupts; 0 when using no interrupts.

TIE: 1 when using interrupts; 0 when using no interrupts.

● Starting Communication

Write data to the output data register 1 (SODR1) to start communication. Temporary data must be written to SODR1 to start communication for reception.

● Ending Communication

The RDRF flag of the status register 1 (SSR1) is set to 1 when transmission or reception of a data frame is complete. During reception, check the overrun error flag bit (SSR1) to see if communication is performing normally.

8.7.4 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 8.7.4-1 are required to operate UART1 in normal mode (operation mode 0 or 2).

	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
SCR1, SMR1	PEN	P	SBL	CL	AD	REC	RXE	TXE	MD1	MD0	CS2	CS1	CS0	BCH	SCKE	SOE
Mode 0	⊙	⊙	⊙	⊙	x	0	⊙	⊙	0	0	⊙	⊙	⊙	⊙	⊙	⊙
Mode 2	0	x	x	1	x	0	⊙	⊙	1	0	⊙	⊙	⊙	⊙	⊙	⊙

SSR1, SISR1/SODR1	PE	ORE	FRE	RDRF	TDRE	-	RIE	TIE	Set conversion data (during writing). Retain receive data (during reading).							
Mode 0	⊙	⊙	⊙	⊙	⊙		⊙	⊙								
Mode 2	x	⊙	x	⊙	⊙		⊙	⊙								

DDR6*1																
--------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

*1 : Set bit 7 of DDR2 and bit 0 of DDR4 to use UART0.

⊙ : Bit used
x : Bit not used
1 : Set 1.
0 : Set 0.
: Set 0 to use an input pin.

Figure 8.7.4-1 Settings for UART1 operation mode 0

● Inter-CPU Connection

As shown in Figure 8.7.4-2, interconnect two CPUs.

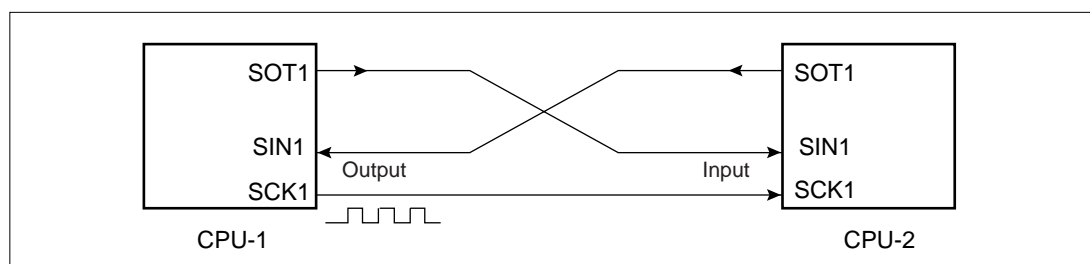


Figure 8.7.4-2 Connection example of UART1 bidirectional communication

8.7 Operation of UART1

● 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 8.7.4-3 shows an example of a bidirectional communication flowchart.

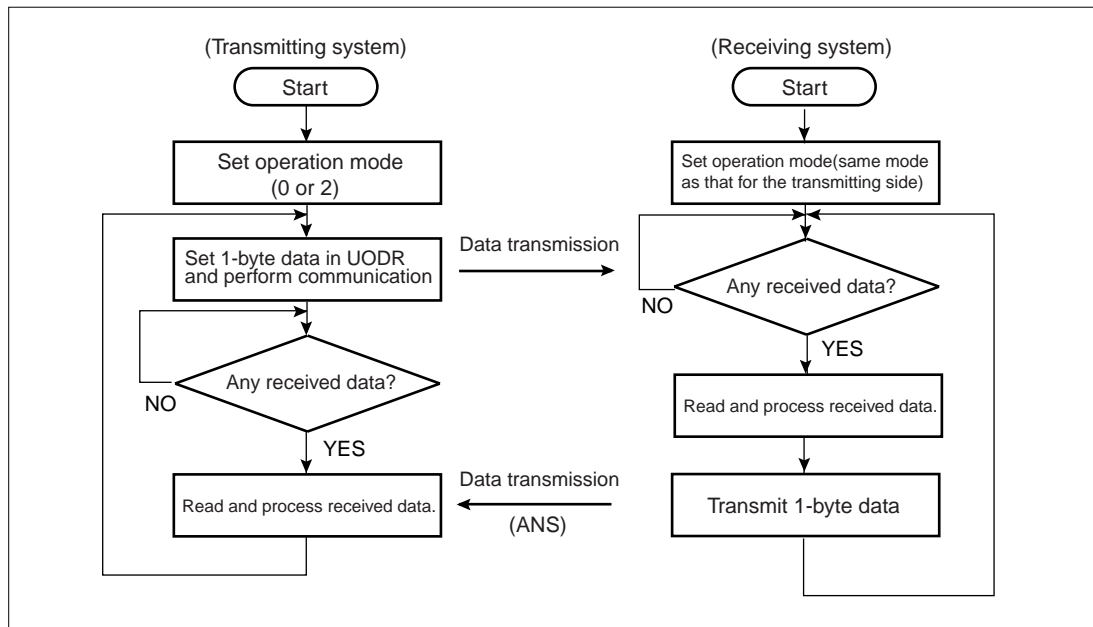


Figure 8.7.4-3 Example of bidirectional communication flowchart

8.7.5 Master-slave Communication Function (Multiprocessor Mode)

With UART1, communication with multiple CPUs connected in master-slave mode is available. However, UART1 can be used only from the master system.

■ Master-slave Communication Function

The settings shown in Figure 8.7.5-1 are required to operate UART1 in multiprocessor mode (operation mode 1).

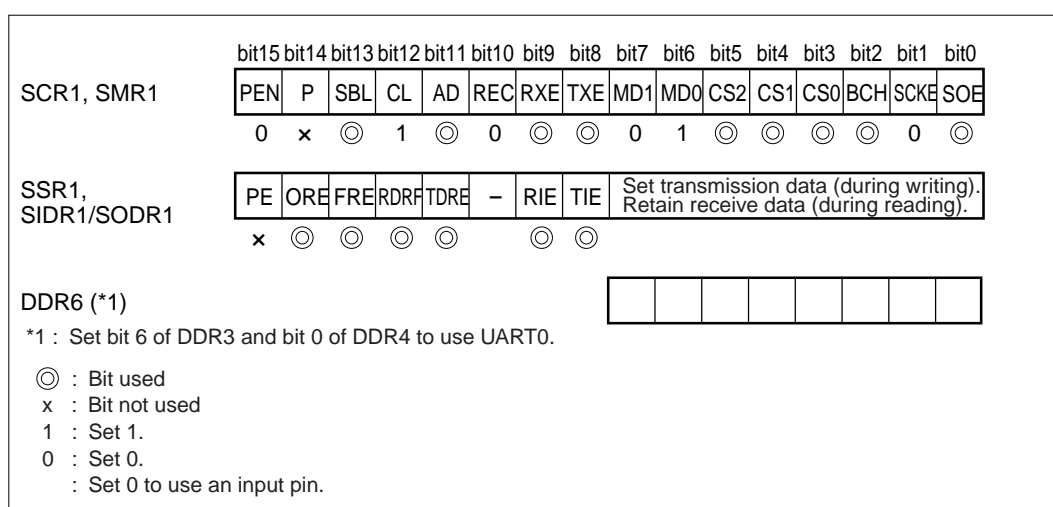


Figure 8.7.5-1 Settings for UART1 operation mode 1

● Inter-CPU Connection

As shown in Figure 8.7.5-2, 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.

8.7 Operation of UART1

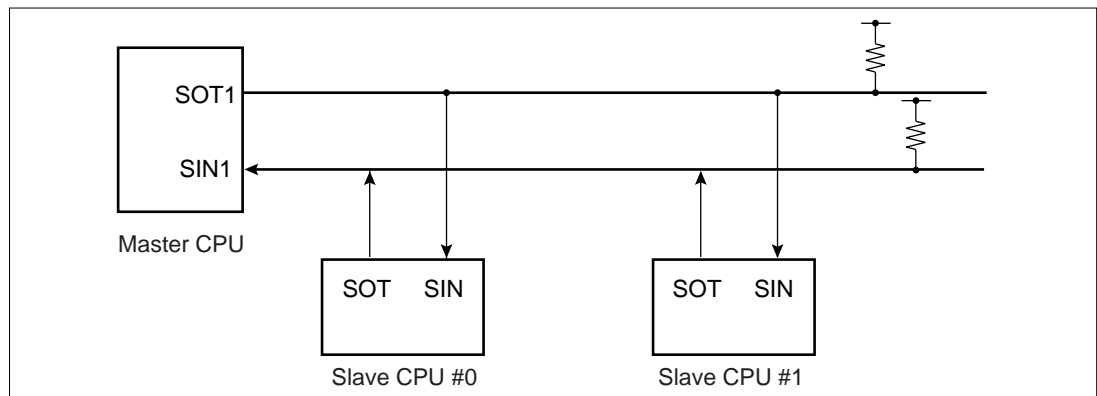


Figure 8.7.5-2 Connection example of UART1 master-slave communication

● Function Selection

Select the operation mode and data transfer mode for master-slave communication as shown in Table 8.7.5-1.

Table 8.7.5-1 Selection of the master-slave communication function

	Operation mode		Data	Parity	Synchroniza tion method	Stop bit
	Master CPU	Slave CPU				
Address transmiss ion and reception	Mode 1	—	A/D="1" + 8-bit address	None	Asynchronous	1 or 2 bits
Data transmiss ion 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 8.7.5-3 shows a flowchart of master-slave communication (multiprocessor mode).

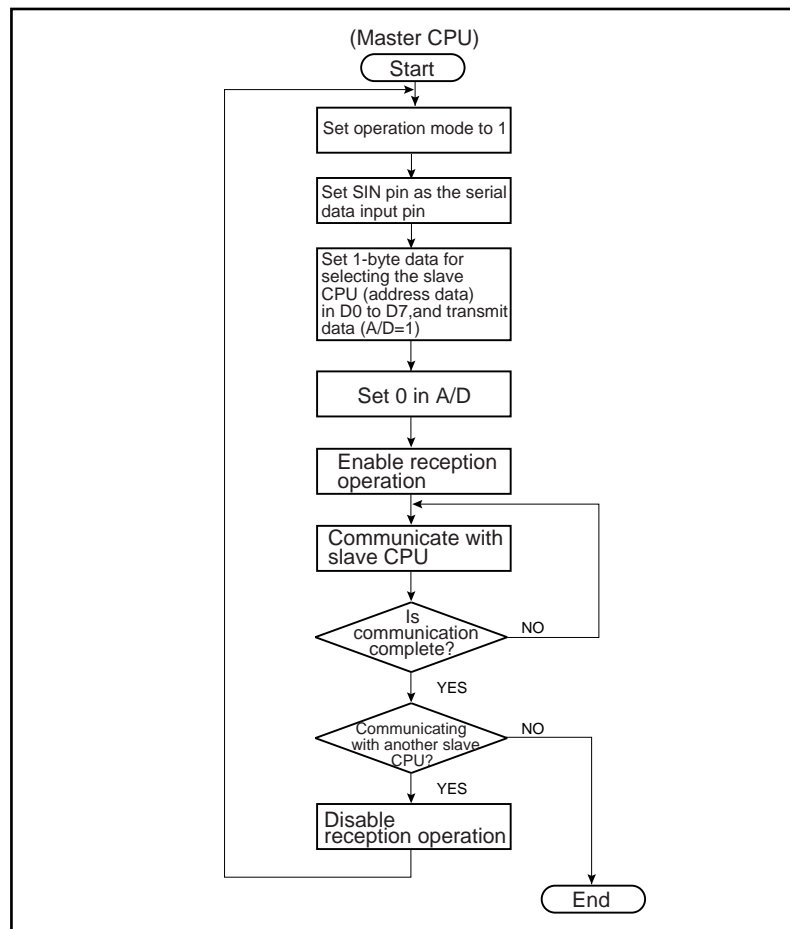


Figure 8.7.5-3 Master-slave communication flowchart

8.8 Notes on Using UART1

Notes on using UART1 are given below.

■ Notes on Using UART1

● Enabling Operations

In UART1, the control register (SCR1) has 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

UART1 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 (SSR1: 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 (SSR1: TIE=1). Be sure to set the TIE flag to 1 after setting the transmission data.

8.9 Sample Program for UART1

This section contains a sample program for UART1.

■ Sample Program for UART1

● 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 P40/SIN1 and P42/SOT1 pins are used for communication.
- The dedicated baud rate generator is used and the baud rate is set to about 9600 bps.
- Character 13_H 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 and recept. interrupt controll register
DDR4	EQU	000014H	; Port-3 data direction register
SMR1	EQU	000026H	; Serial Model register 1
SCR1	EQU	000027H	; Control register 1
SIDR1	EQU	000028H	; Input data register 1
SODR1	EQU	000028H	; Output data register 1
SSR1	EQU	000029H	; 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:DDR4,#00000000B	; Sets SIN1 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.
```

CODEENDS

;-----Vector setting-----

VECT CSEGABS=0FFH

```
        ORG    0FF68H           ; Sets vector for interrupt #37 (25H).
        DSL    WARI
        ORG    0FFDCH           ; Sets reset vector.
        DSL    START
        DB     00H              ; Sets single-chip mode.
```

VECTENDS

CHAPTER 9 TIMEBASE TIMER

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

- 9.1 Overview of the Timebase Timer
- 9.2 Configuration of the Timebase Timer
- 9.3 Timebase Timer Control Register (TBTC)
- 9.4 Timebase Timer Interrupts
- 9.5 Operation of the Timebase Timer
- 9.6 Usage Notes on the Timebase Timer
- 9.7 Sample Program for the Timebase Timer

9.1 Overview of the Timebase Timer

The timebase timer is an 18-bit free-running counter (time base counter) that counts up in synchronization with the internal count clock (the source oscillation divided by 2). The timer provides an interval timer function that can select four intervals. The timebase timer also supports the function for timer output of the oscillation stabilization wait time and for supplying the operating 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.
- The interval timer bit (interval) can be selected from four types.

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 s)	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 operating clocks to the oscillation stabilization wait time timer and to some peripheral functions. Table 9.1-2 lists the cycles of clocks supplied from the timebase timer to each peripheral.

Table 9.1-2 Cycles of Clocks Supplied from the Timebase Timer

Clock destination	Clock cycle	Remarks
Oscillation stabilization wait time	2^{13} /HCLK (approx. 2.0 ms)	Oscillation stabilization wait time for the ceramic vibrator
	2^{15} /HCLK (approx. 8.2 ms)	Oscillation stabilization wait time for the crystal vibrator
	2^{17} /HCLK (approx. 32.8 ms)	

9.1 Overview of the Timebase Timer

Table 9.1-2 Cycles of Clocks Supplied from the Timebase Timer (Continued)

Clock destination	Clock cycle	Remarks
Watchdog timer	$2^{12}/\text{HCLK}$ (approx. 1.0 ms)	Count-up clock for watchdog timer
	$2^{14}/\text{HCLK}$ (approx. 4.1 ms)	
	$2^{16}/\text{HCLK}$ (approx. 16.4 ms)	
	$2^{19}/\text{HCLK}$ (approx. 131.1 ms)	

HCLK: Oscillation clock

Values in parentheses occur during operation of a 4-MHz oscillation clock.

Reference

The oscillation stabilization wait time is a rough estimate because the oscillation cycle is unstable immediately after oscillation starts.

9.2 Configuration of the Timebase Timer

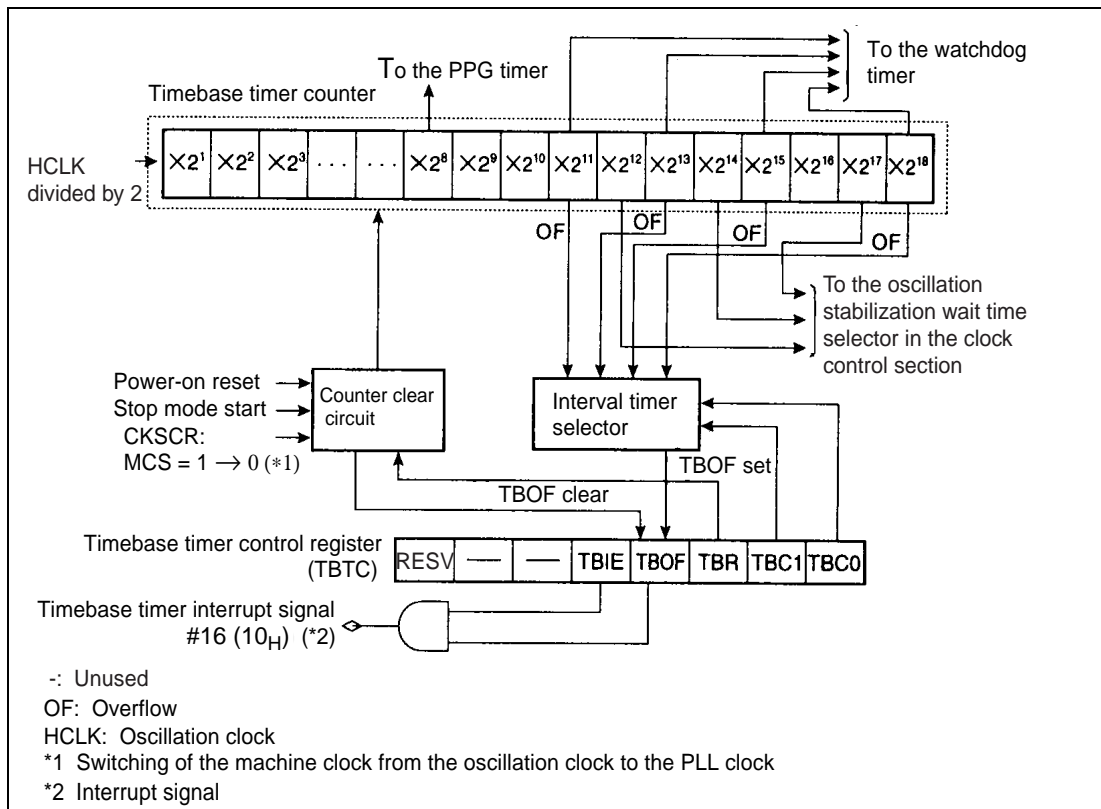
The timebase timer consists of the following four blocks:

- Time-base 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

This circuit is used to clear the counter by writing 00_H to the TBTC TBR bit, by a power-on reset, or by transition to stop mode (LPMCR:STP = 1).

9.2 Configuration of the Timebase Timer

- **Interval timer selector**

This selector is used to select one of four outputs of the timebase timer counter. An overflow of the selected bit becomes an interrupt cause.

- **Timebase timer control register (TBTC)**

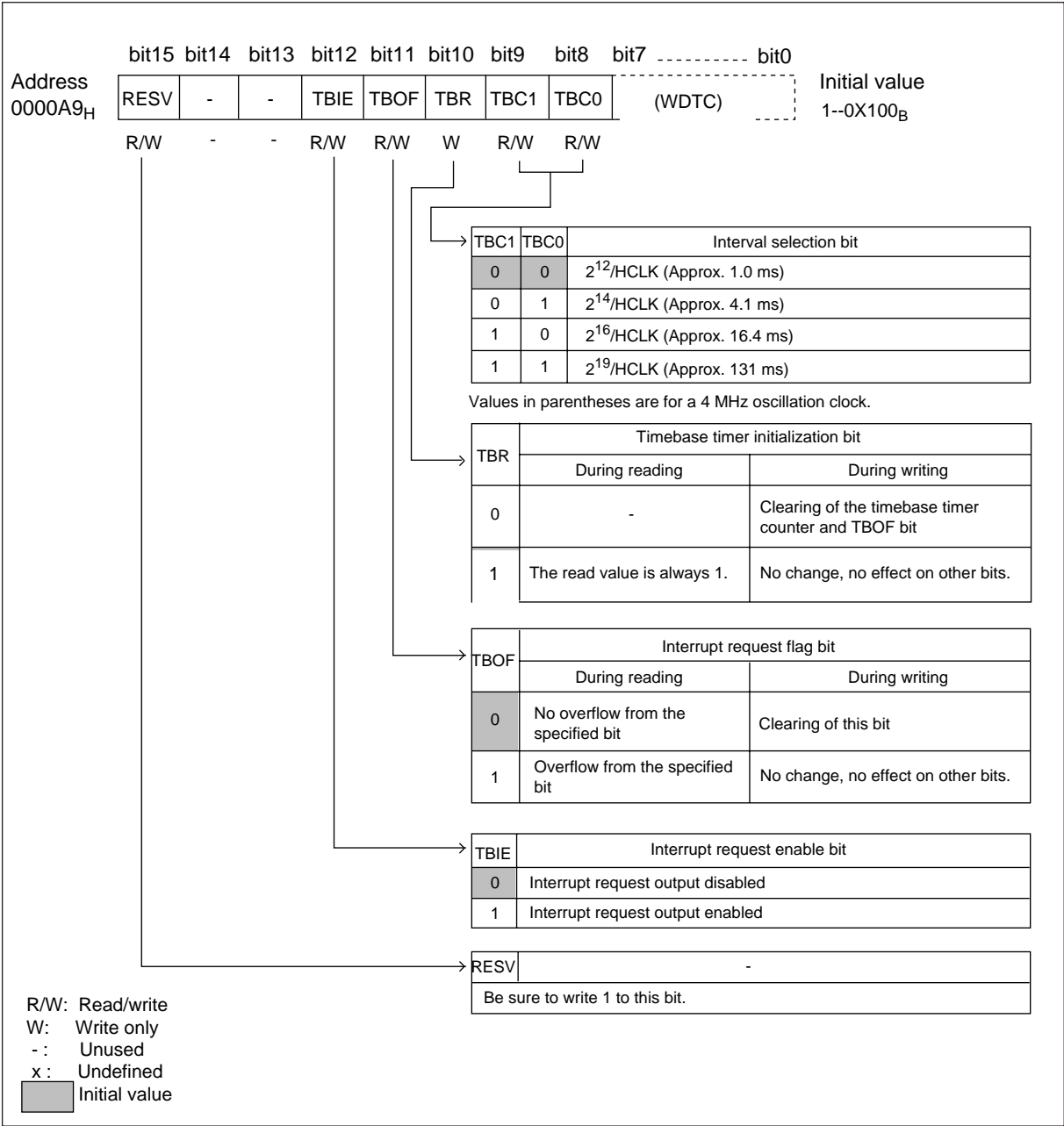
This register selects an interval, clears the counter, controls an interrupt request, and checks the status.

9.3 Timebase Timer Control Register (TBTC)

The timebase timer control register (TBTC) selects an interval, clears the counter, controls interrupts, and checks the status.

■ Timebase Timer Control Register (TBTC)

Figure 9.3-1 Timebase Timer Control Register (TBTC)



9.3 Timebase Timer Control Register (TBTC)

Table 9.3-1 Function Explanation of each Bit of the Timebase Timer Control Register (TBTC)

Bit name		Function
bit15	Unused	<p><Caution> Be sure to write 1 to this bit.</p>
bit14 bit13	Unused	<ul style="list-style-type: none"> When read, the value is undefined. Writing has no effect on operation.
bit12	TBIE: Interrupt request enable bit	<ul style="list-style-type: none"> 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. This bit is cleared to 0 by writing 0, by a transition to the stop mode or hardware standby mode, by a transition from the sub-clock mode to main clock mode, by a transition from the main clock mode to PLL clock mode, or by a reset.
bit11	TBOF: Interrupt request flag bit	<ul style="list-style-type: none"> 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 remains unchanged and has no effect on other bits. <p><Caution></p> <ul style="list-style-type: none"> To clear the TBOF bit, disable the timebase timer interrupt by specifying the TBIE bit or the processor status (PS) ILM bit. The TBOF bit is cleared by writing 0, by a transition to stop mode or hardware standby mode, by clearing the timebase timer with the TBR bit, or by a reset.
bit10	TBR: Timebase timer initialization bit	<ul style="list-style-type: none"> 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 remains unchanged and has no effect on other bits. <p>[Reference] The read value is always 1.</p>
bit9 bit8	TBC1, TBC0: Interval selection bit	<ul style="list-style-type: none"> Used to select an interval timer cycle. The bit for the interval timer of the timebase timer counter is specified. Four types of intervals 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 = 1) is set when the timebase timer counter counts up with the internal count clock and the bit for the selected interval timer overflows. If the interrupt request enable bit is enabled (TBTC: TBIE = 1), an interrupt request (#36) is generated in the CPU. Write 0 to the TBOF bit with the interrupt handling routine to clear the interrupt request. When the specified bit overflows, the TBOF bit is set regardless of the TBIE bit value.

Note

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 support 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 Timer Interrupts and EI²OS

Interrupt number	Interrupt level setting register		Vector table address			EI ² OS
	Register name	Address	Lower	Upper	Bank	
#16 (10 _H)	ICR02	0000B2 _H	FFFFBC _H	FFFFBD _H	FFFFBE _H	x

x: Not available

Note

ICR12 is common to the timebase timer interrupt and wake-up interrupt. An interrupt can be used for two applications, but the interrupt level is the same.

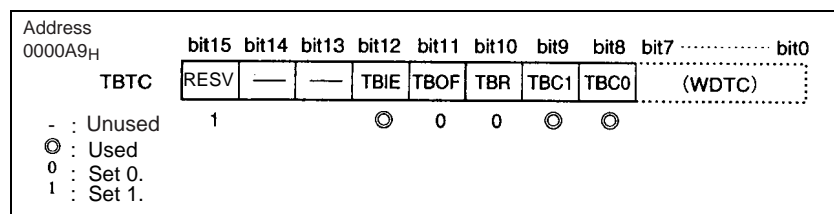
9.5 Operation of the Timebase Timer

The timebase timer provides the interval timer function and the clock supply function that supplies clocks to peripheral functions.

■ Operation of the Interval Timer Function (Timebase Timer)

The interval timer function generates an interrupt request for the intervals. The setting in Figure 9.5-1 is required to make the timer operate as the 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 (the oscillation clock divided by 2) as long as the clock is oscillating.
- 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 an 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 Time Timer Function

The timebase timer is also used as the oscillation stabilization wait time timer for oscillation clocks and PLL clocks. The oscillation stabilization wait time counts up from when the counter is 0 (count clear) until the oscillation stabilization wait time bit overflows. When control returns from timebase timer mode to PLL clock mode, the oscillation stabilization wait time starts from the middle of counting because the timebase timer counter has not been cleared. Table 9.5-1 shows the clearing of the timebase counter and the oscillation stabilization wait times.

Table 9.5-1 Timebase Timer Counter Clearing and Oscillation Stabilization Wait Times

Operation	Counter clear	TBOF clear	Oscillation stabilization wait time
TBTC: Writing of 0 to TBR	o	o	
Power-on reset	o	o	Oscillation clock oscillation stabilization wait time
Watchdog reset			
Releasing of stop mode	o	o	Oscillation clock oscillation stabilization wait time (at return to main clock mode)

Table 9.5-1 Timebase Timer Counter Clearing and Oscillation Stabilization Wait Times

Operation	Counter clear	TBOF clear	Oscillation stabilization wait time
Transition from main clock mode to PLL clock mode (MCS = 1-->0)	o	o	PLL clock oscillation stabilization wait time
Releasing of time-base timer mode	x	x	PLL clock oscillation stabilization wait time (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

This section provides notes on the effects on peripheral functions when interrupt requests and the timebase timer are cleared.

■ Usage Notes on the Timebase Timer

○ 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

○ Using the timebase timer as the oscillation stabilization wait time timer

At power-on, 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 vibrator connected to the main clock oscillator (clock generation section). For details, see Section 4.5 "Oscillation Stabilization Wait Interval".

○ Notes on peripheral functions to which clocks are supplied from the timebase timer

In the mode in which the main clock 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 status. As a result, the H level may be shortened and the L level lengthened up to 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.

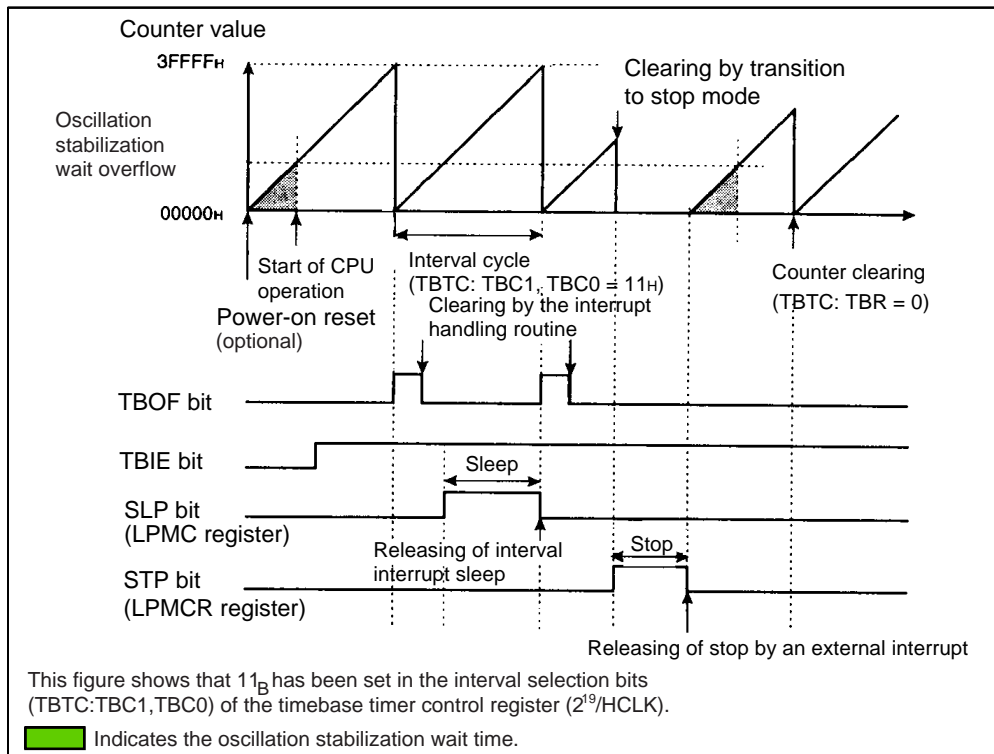
■ Timebase Timer Operation

The following operations are shown in Figure 9.6-1 :

- A power-on reset occurs.
- Sleep mode is entered during operation of the interval timer function.
- The mode is changed to the stop mode.
- A counter clear is requested.

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 stabilization wait time.

Figure 9.6-1 Timebase Timer Operations



9.7 Sample Program for the Timebase Timer

This section provides a sample program for the timebase timer.

■ Sample Program for the Timebase Timer

○ Processing specifications

An interval interrupt of $2^{12}/\text{HCLK}$ (HCLK: oscillation clock) is generated repeatedly. 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-----
CODE CSEG
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  $2^{12}/\text{HCLK}$ 
        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

```


CHAPTER 10 WATCHDOG TIMER

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

- 10.1 Overview of the Watchdog Timer
- 10.2 Watchdog Timer Control Register (WDTC)
- 10.3 Configuration of the Watchdog Timer
- 10.4 Operation of the Watchdog Timer
- 10.5 Usage Notes on the Watchdog Timer
- 10.6 Sample Program for the Watchdog Timer

10.1 Overview of the Watchdog Timer

The watchdog timer is a 2-bit counter that uses the timebase timer or watch timer output as the count clock. After activation, if the watchdog timer is not cleared within a given time, the CPU is reseted.

■ 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 a 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 occurs 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

WT1	WT0	WDSC&SCM	Interval		Clock cycle count
			Minimum ^(*1)	Maximum ^(*1)	
0	0	1	Approx. 3.58 ms	Approx. 4.61 ms	2^{14} plus or minus 2^{11} HCLK cycle
0	1	1	Approx. 14.33 ms	Approx. 18.43 ms	2^{16} plus or minus 2^{15} HCLK cycle
1	0	1	Approx. 57.23 ms	Approx. 73.73 ms	2^{18} plus or minus 2^{15} HCLK cycle
1	1	1	Approx. 458.75 ms	Approx. 589.82 ms	2^{21} plus or minus 2^{18} HCLK cycle
0	0	0	Approx. 0.109 s	Approx. 0.141 s	2^{12} plus or minus 2^9 SCLK cycle
0	1	0	Approx. 0.875 s	Approx. 1.125 s	2^{15} plus or minus 2^{12} SCLK cycle
1	0	0	Approx. 1.75 s	Approx. 2.25 s	2^{16} plus or minus 2^{13} SCLK cycle
1	1	0	Approx. 3.5 s	Approx. 4.5 s	2^{17} plus or minus 2^{14} SCLK cycle

*1 Value when the oscillation clock (HCLK) is at 4 MHz and the subclock (SCLK) at 32 KHz

The maximum and minimum watchdog timer intervals and the oscillation clock cycle count depends 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).

For the watchdog timer intervals, see Section 10.4 "Operation of the Watchdog Timer".

Note

The watchdog counter consists of a 2-bit counter that uses the carry signals of the timebase timer as count clocks. Therefore, if the timebase timer is cleared, the watchdog reset generation time may become longer than the time set.

Reference

When the watchdog timer is activated, it is initialized by a power-on or watchdog reset and is placed in stopped status. The watchdog counter is cleared by an external pin reset, software reset, writing to the WTE bit (watchdog timer control register), sleep mode, or transition to stop mode, though the watchdog timer remains active.

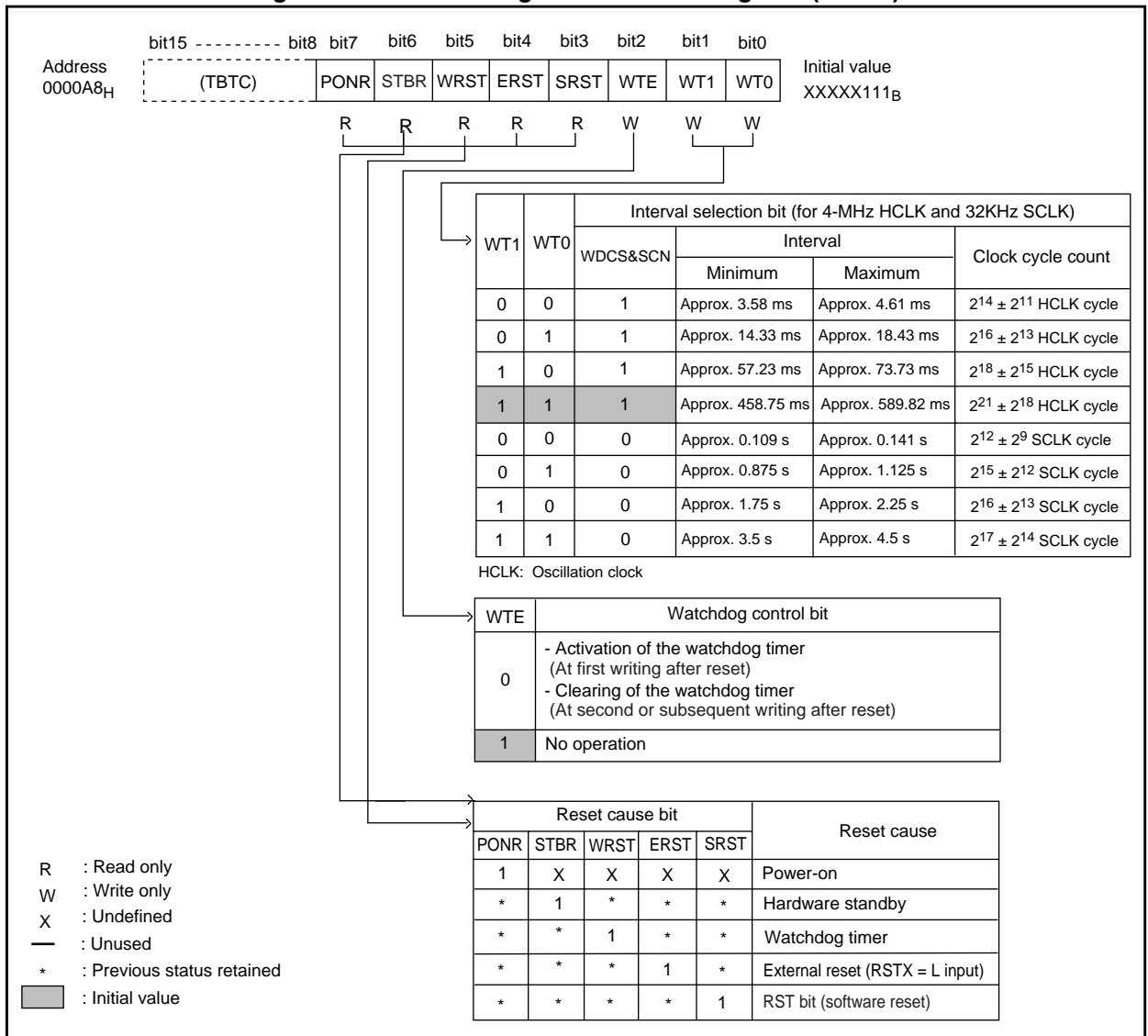
10.2 Watchdog Timer Control Register (WDTC)

The watchdog timer control register (WDTC) activates and clears the watchdog timer and displays a reset cause.

■ Watchdog Timer Control Register (WDTC)

Figure 10.2-1 shows the watchdog timer control register (WDTC); Table 10.2-1 describes the function of each bit of the watchdog timer control register (WDTC).

Figure 10.2-1 Watchdog Timer Control Register (WDTC)



The interval becomes 3.5 to 4.5 times longer than the cycle of the count clock (timebase timer)

output value). For details, see Section 10.4 "Operation of the Watchdog Timer".

Table 10.2-1 Function Explanation of Each Bit of the Watchdog Timer Control Register (WDTC)

Bit name			Function
bit 7	PONR	Reset cause bits	<ul style="list-style-type: none"> Read-only bits for indicating reset causes. If a reset cause occurs, the related bit 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 bits other than the PONR bit are not guaranteed. Therefore, when the PONR bit is 1, ignore the contents of bits other than the PONR bit.
bit 6	STBR		
bit 5	WRST		
bit 4	ERST		
bit 3	SRST		
bit 2	WTE	Watchdog control bit	<ul style="list-style-type: none"> When 0 is written to this bit, the watchdog timer is activated (first writing after reset) or the 2-bit counter is cleared (second or subsequent writing after reset). Writing 1 to this bit does not affect operation.
bit 1	WT1	Interval selection bit	<ul style="list-style-type: none"> Used to select a watchdog timer interval. As shown in Figure 10.2-1, the interval is dependent on whether the subclock mode has been selected as clock mode (clock selection register: CKSCR SCM bit is 0) or the watchdog timer clock source has become the watch timer (WDCS bit is 0) according to the watch timer control register (WTC) or whether the main clock mode or PLL clock mode has been selected as clock mode and the WTC WDCS bit is 1. Only data at watchdog timer activation is valid. Data written after watchdog timer activation is ignored. These bits are write-only.
bit 0	WT0		

10.3 Configuration of the Watchdog Timer

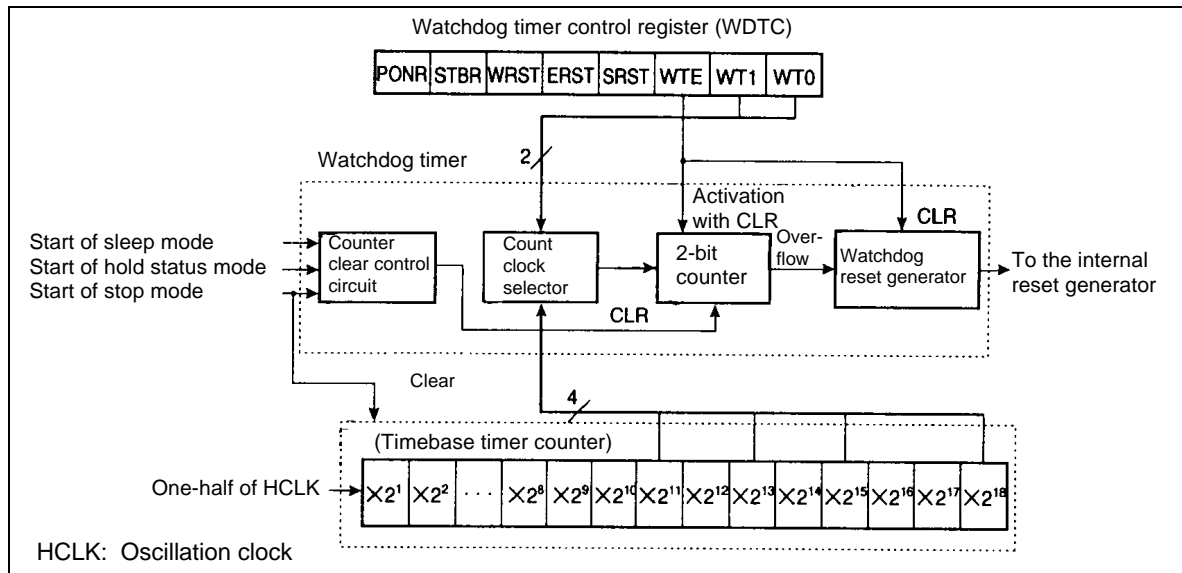
The watchdog timer consists of the following five blocks:

- 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 10.3-1 shows the block diagram of the watchdog timer.

Figure 10.3-1 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 and four types of watch timer outputs, thereby determining 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 control the operation or stopping of the counter.

CHAPTER 10 WATCHDOG TIMER

- **Watchdog timer control register (WDTC)**

Used to activate or clear the watchdog timer and to hold the reset generation cause.

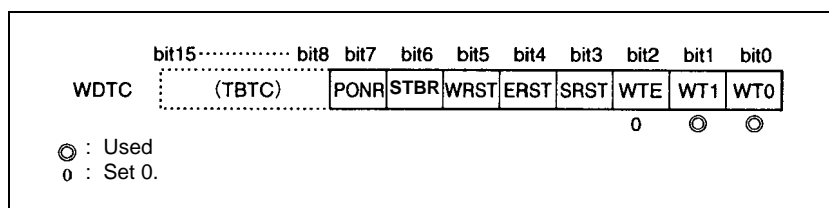
10.4 Operation of the Watchdog Timer

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 shown in Figure 10.4-1 .

Figure 10.4-1 Setting of the Watchdog Timer



○ Activating the watchdog timer

- The watchdog timer is activated when the first 0 is written after reset 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 (WDTC).
- When watchdog timer activation starts, only a power-on or watchdog reset can stop it.

○ Clearing the watchdog timer

- The 2-bit counter of the watchdog timer is cleared by writing a second or subsequent 0 to the WTE bit. If the counter is not cleared within the specified interval, it overflows and a watchdog reset occurs.
- The watchdog counter is cleared by reset generation or transition to sleep, stop, or timebase timer mode.
- While timebase timer mode is in effect, the watchdog counter is cleared and stops.

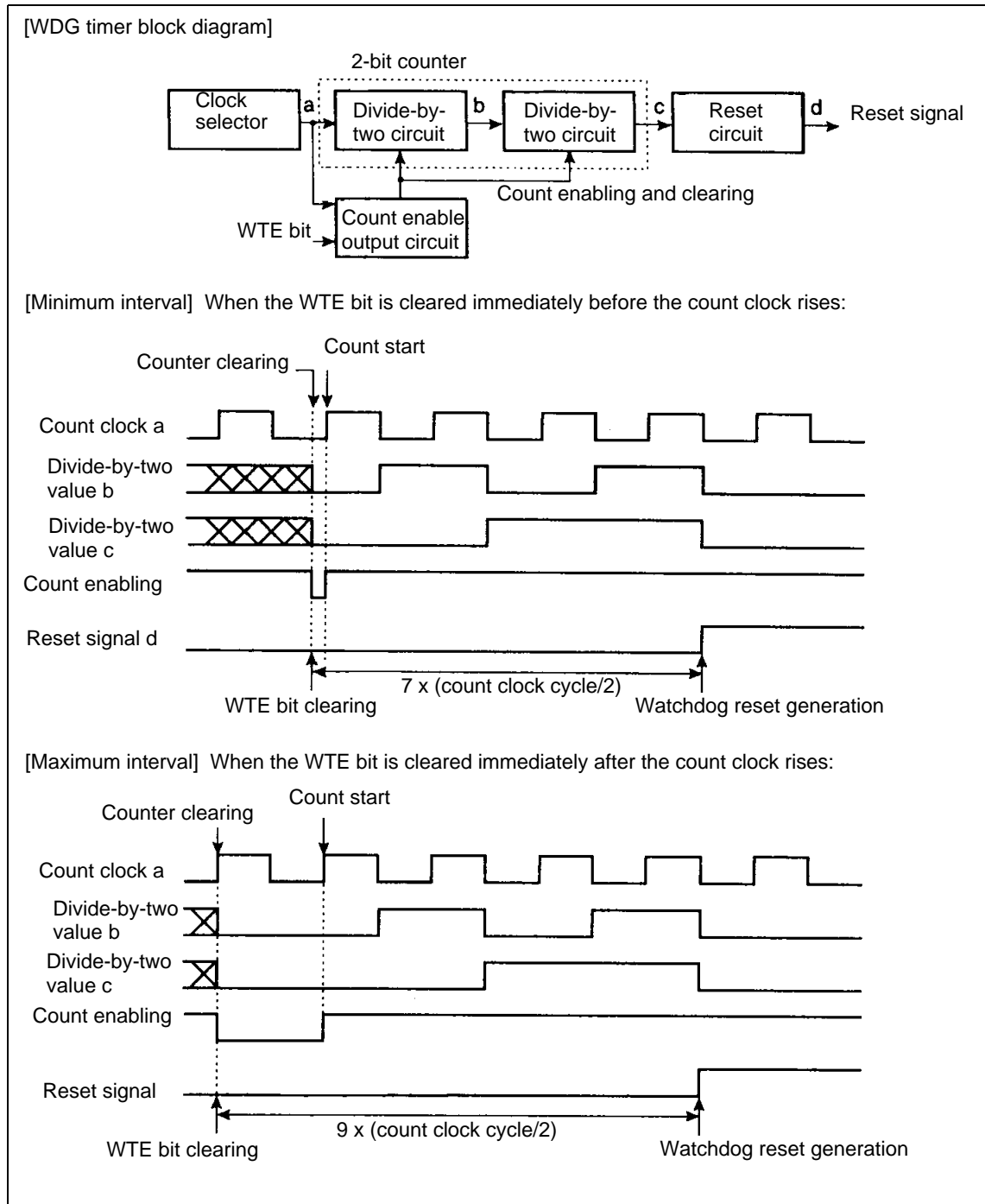
○ Intervals of the watchdog timer

Figure 10.4-2 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, STBR, WRST, ERST, and SRST bits of the watchdog timer control register (WDTC) after reset.

Figure 10.4-2 Watchdog Timer Clear Timing and Intervals



10.5 Usage Notes on the Watchdog Timer

This section provides notes on using the watchdog timer.

■ 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 counter is cleared by an external pin or software reset; though the watchdog timer remains active.

- **Intervals**

Since a carry signal of the timebase timer is used as the count clock for intervals, 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 is operating while timebase timer mode is set. The watchdog timer, however, stops temporarily.

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 specifications

- 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 the stack pointer (SP) has already been initialized.

WDG_START:
    MOV WDTC, #00000011B ; Activates watchdog timer.
                        ; Selects the interval of  $2^{21} \pm 2^{18}$  cycle.
;-----Main loop-----
MAIN:  CLRB I:WTE        ; Clears the 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 the reset vector.
      DSL START
      DB 00H              ; Sets the single-chip mode.
VECT ENDS
      END START

```

CHAPTER 11 WATCH TIMER

This chapter explains the functions and operation of the watch timer.

11.1 Overview of the Watch Timer

11.2 Watch Timer Control Register (WTC)

11.3 Configuration of the Watch Timer

11.4 Operation of the Watch Timer

11.1 Overview of the Watch Timer

The watch timer is a 15-bit timer that uses subclocks and can generate an interval interrupt. The timer can also be used as the watchdog timer clock source in accordance with the setting.

■ Watch Timer Function

The watch timer consists of the 15-bit timer and the circuit that controls interval interrupts.

The watch timer uses subclocks, regardless of the MCS and SCS bits of the clock selection register (CKSCR).

Table 11.1-1 shows the intervals of the watch timer.

Table 11.1-1 Intervals of the Clock Timer

WTC2	WTC1	WTC0	Interval (*1)
0	0	0	62.5 ms
0	0	1	125 ms
0	1	0	250 ms
0	1	1	500 ms
1	0	0	1,000 s
1	0	1	2,000 s
1	1	0	4,000 s
1	1	1	Setting disabled

*1 For subclock of 32 KHz

11.2 Watch Timer Control Register (WTC)

The watch timer control register (WTC) controls operation of the watch timer and also controls the interval interrupt time.

■ Configuration of the Watch Timer Control Register (WTC)

Figure 11.2-1 shows the configuration of the watch timer control register (WTC). Table 11.2-1 describes the function of each bit of the watch timer control register (WTC).

Figure 11.2-1 Configuration of the Watch Timer Control Register (WTC)

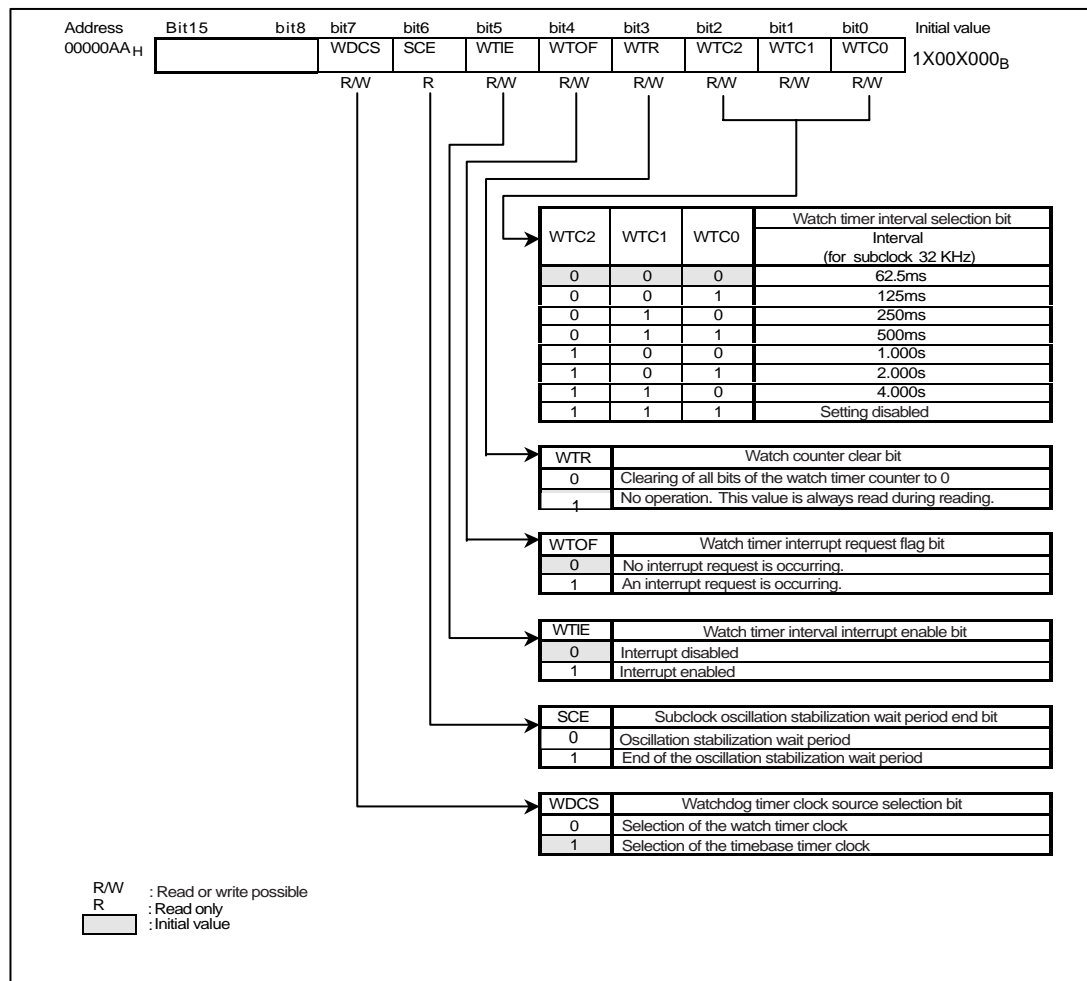


Table 11.2-1 Function Description of Each Bit of the Watch Timer Control Register (WTC)

	Bit name	Function
bit7	WDCS: Watchdog timer Clock source Selection bit	<ul style="list-style-type: none"> Used to select the watchdog timer clock source. When this bit is 0, the watch timer clock is selected. When it is 1, the timebase timer clock is selected. This bit is initialized to 1 by a reset.
bit6	SCE: Subclock oscillation sta- bilization Wait period end bit	<ul style="list-style-type: none"> Used to indicate that the subclock oscillation stabilization wait period has ended. When this bit is 0, it indicates the oscillation stabilization wait period. The oscillation stabilization wait period of the subclock is fixed to the 2^{17} subclock cycle. This bit is initialized to 0 at a power-on reset or stopping.
bit5	WTIE: Watch timer interval Inter- rupt enable bit	<ul style="list-style-type: none"> Used to enable an interval interrupt of the watch timer. When this bit is 1, an interrupt is enabled. When it is 0, an interrupt is disabled. This bit is initialized to 0 by a reset.
bit4	WTOF: Watch timer interrupt request Flag bit	<ul style="list-style-type: none"> Used to indicate that an watch timer interrupt request is occurring. When the WTIE bit is 1 and this bit is set to 1, an interrupt request occurs. This bit is set to 1 for each interval set by the WTC2 to WTC0 bits. This bit is cleared to 0 by writing 0, transition to stop mode, transition to hardware standby mode, or a reset. Writing 1 to this bit is meaningless.
bit3	WTR: Watch counter Clear bit	<ul style="list-style-type: none"> Used to clear all bits of the watch timer to 0. Writing 0 to this bit clears the watch timer counter to 0. Writing 1 to this bit is meaningless. At reading, 1 is always read.
bit2 bit1 bit0	WTC2, WTC1, WTC0 Watch timer interval Selec- tion bit	<ul style="list-style-type: none"> Used to set the watch time interval. These bits are initialized to 00_B by a reset. Changing this bit requires bit 4:WTOF to be cleared at the same time.

11.3 Configuration of the Watch Timer

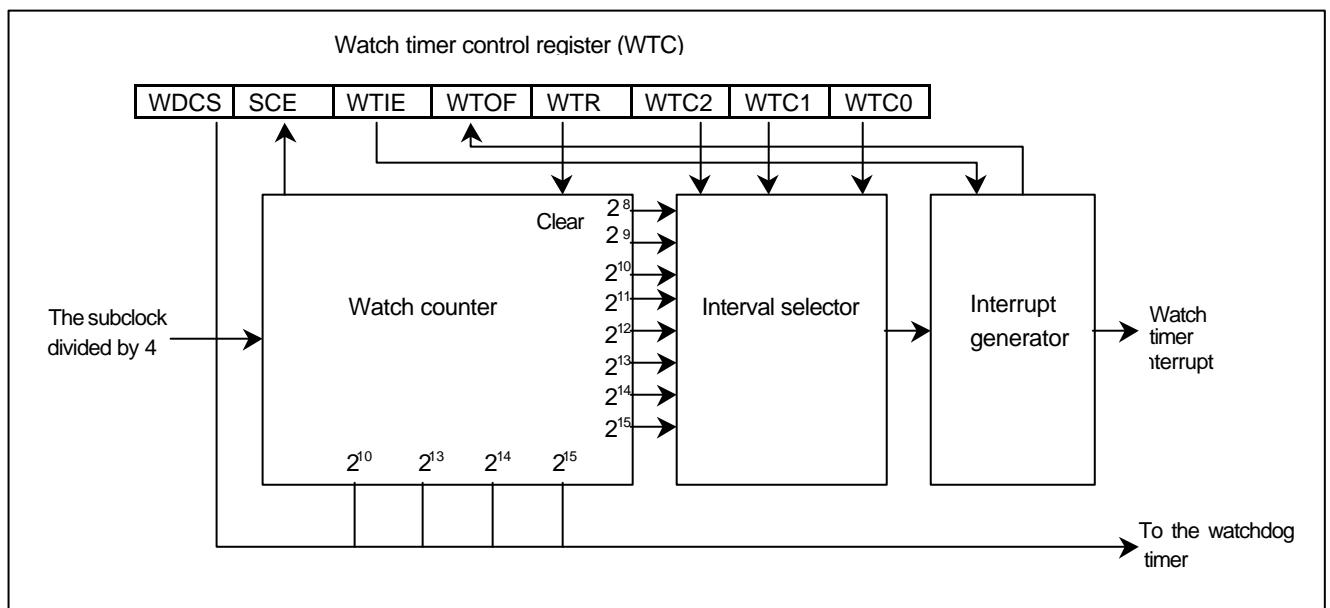
The watch timer consists of the following three blocks

- Interval selector
- Watch counter
- Watch timer interrupt generator
- Watch timer control register (WTC)

■ Block Diagram of the Watch Timer

Figure 11.3-1 shows the block diagram of the watch timer.

Figure 11.3-1 Block Diagram of the Watch Timer



○ Watch counter

This 15-bit up counter uses the subclock divided by 4 as the clock source.

○ Interval selector

Used to select a watch timer interrupt interval from eight types.

○ Interrupt generator

Used to generate a watch timer interval interrupt.

○ Watch timer control register (WTC)

Used to control the watch timer operation and watch timer interrupts and to specify a watchdog timer clock source.

11.4 Operation of the Watch Timer

The watch timer provides the watchdog timer clock source, subclock oscillation stabilization wait timer function, and interval timer function that generates interrupts in a given cycle.

■ Watch Counter

The watch counter is a 15-bit counter that counts the subclock divided by 4 and continues counting while the subclock is input.

○ Clearing the watch counter

The watch counter is cleared by a power-on reset, transition to stop or hardware standby mode, or by writing 0 to the WTR bit of the WTC register.

Note

The watchdog timer operation and interval interrupts that use watch timer outputs are affected by watch counter clearing.

■ Watch Timer Interval Interrupt Function

Used to generate interrupts in a given cycle with a carry signal of the watch timer counter.

○ Specifying an interval

The interval can be specified by the WTC2, WTC1, and WTC0 bits of the WTC register.

○ Generating watch timer interrupts

The WTOF bit is set for each interval set by the WTC2, WTC1, and WTC0 bits. When the WTIE bit is set to 1 and interrupts are enabled, a watch timer interrupt is generated.

The WTOF bit is set based on the time at which the watch timer was last cleared.

When stop or hardware standby mode is entered, the WTOF bit is cleared simultaneously with mode transition because the watch timer is used as the subclock oscillation stabilization time wait timer.

■ Watchdog Timer Clock Source Specification Function

The watchdog timer clock source can be specified by the WDSCS bit of the WTC register. However, when clock mode is subclock mode, the watch timer counter value is used regardless of the WDSCS bit.

■ Subclock Oscillation Stabilization Time Wait Function

At a power-on reset, or on return from stop mode or hardware standby mode, the watch timer functions as the subclock oscillation stabilization time wait timer. The subclock oscillation stabilization wait time is fixed to the 2^{17} cycle of the subclock.

CHAPTER 12 16-Bit I/O Timer

The MB90495 Series contains one 16-bit free-running timer module and two input capture modules and supports four input channels. The following sections only describes the 16-bit free-running timer and Input Capture 0/1. The remaining modules have the identical functions and the register addresses can be found in the I/O map.

12.1 Function overview

12.2 Registers

12.3 Block diagram

12.4 Input capture

12.1 Function overview

□ 16-bit free-running timer

The 16-bit free-run timer consists of a 16-bit up counter, control register and prescaler. The values output from this timer counter are used as the base timer for input capture and output compare.

- Four counter clocks are available.
Internal clock: $\phi/4$, $\phi/16$, $\phi/64$, $\phi/256$
- An interrupt can be generated upon a counter overflow or a match with compare register 0.
- The counter value can be initialized to '0000H' upon a reset, software clear or match with compare register 0.

□ Input capture (2 channels per one module)

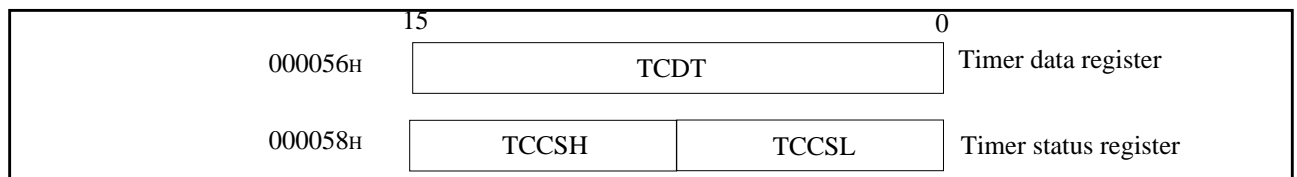
The input capture module consists of two 16-bit capture registers and control registers corresponding to two independent external input pins. The 16-bit free-running timer value can be stored in the capture register and an interrupt is issued simultaneously upon detection of an edge of a signal input from an external input pin.

- The detection edge of an external input signal can be specified.
Rising, falling, or both edges
- Two input channels can operate independently.
- An interrupt can be issued upon a valid edge of an external input signal.

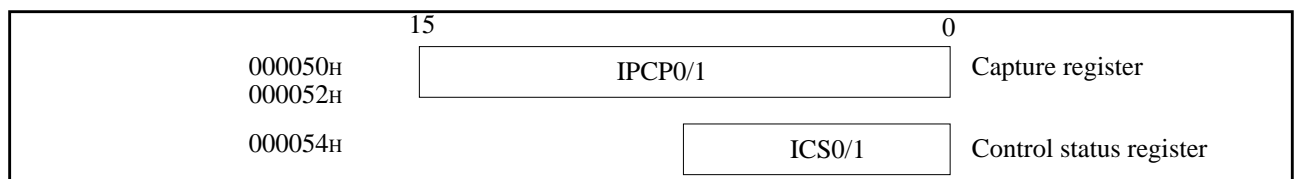
The intelligent I/O service can be activated upon an input capture interrupt.

12.2 Registers

12.2.1 16-bit free-running timer



12.2.2 16-bit input capture



12.3.3 16-bit free-running timer register details

(1) Data register (TCDT)

Address: 000057H	bit	15	14	13	12	11	10	9	8	
		T15	T14	T13	T12	T11	T10	T09	T08	
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	←Attribute
		X	X	X	X	X	X	X	X	←Initial value

Address: 000056H	bit	7	6	5	4	3	2	1	0	
		T07	T06	T05	T04	T03	T02	T01	T00	
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	←Attribute
		X	X	X	X	X	X	X	X	←Initial value

The data register can read the count value of the 16-bit free-running timer. The counter value is undefined 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).

The data register must be accessed by the word access instructions.

The 16-bit free-running timer is initialized upon the following factors:

- Reset
- Clear bit (CLR) of control status register
- A match between compare register 0 and the timer counter value

(2)Control status register (TCCSH)

Address: 000058H	bit	7	6	5	4	3	2	1	0	
		ECKE	-	-	-	-	-	-	-	
		R/W								←Attribute
		0								←Initial value

[bit 15] ECKE

This bit chose between internal timer clock (0) and external clock from FRCK (1)

(3)Control status register (TCCSL)

Address: 000059H	bit	7	6	5	4	3	2	1	0	
		IVF	IVFE	STOP	MODE	CLR	CLK2	CLK1	CLK0	
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	←Attribute
		0	0	0	0	0	0	0	0	←Initial value

[bit 7] IVF

This bit is an interrupt request flag of the 16-bit free-running timer.

If the 16-bit free-running timer overflows or if the counter is cleared by a match with compare register 0, '1' is set to this bit.

An interrupt is issued if the interrupt request enable bit (bit 5: IVFE) is set.

This bit is cleared by writing '0.' Writing '1' has no effect.

'1' is always read by a read-modify-write instruction.

0	No interrupt request (default)
1	Interrupt request

[bit 6] IVFE

IVFE is an interrupt enable bit of the 16-bit free-run timer. While this bit is "1", an interrupt is issued if '1' is set to the interrupt flag (bit 5: IVF).

0	Interrupt disabled (default)
1	Interrupt enabled

[bit 5] STOP

The STOP bit is used to stop the 16-bit free-running timer.

Writing '1' to this bit stops the timer. Writing '0' starts the timer.

0	Counter enabled (operation) (default)
1	Counter disabled (stop)

The output compare operation stops when the 16-bit free-running timer stops.

[bit 4] MODE

The MODE bit is used to set the reset condition of the 16-bit free-running timer.

When '0' is set, the counter value can be initialized by RESET or a clear bit (bit 2: CLR).

When '1' is set, the counter value can be initialized by a match with compare register 0 in addition to RESET and a clear bit (bit 2: CLR).

0	Initialization by reset or clear bit (default)
1	Initialization by reset, clear bit, or compare register 0

The clear bit and the match with compare register initializes the timer when the timer value changes.

[bit 3] CLR

The CLR bit initializes the operating 16-bit free-running timer value to '0000.'

When '1' is set, the counter value is initialized to '0000.' Writing '0' has no effect. '0' is always read from this bit. The counter value is initialized when the count value changes.

0	No effect (default)
1	The counter value is initialized to '0000.

To initialize the counter value while the timer is stopped, write '0000' to the data register.

[bits 2, 1 and 0] CLK2, CLK1 and CLK0

CLK1 and CLK0 are used to select the count clock for the 16-bit free-run timer. The clock is updated immediately after a value is written to these bits. Therefore, ensure that the output compare and input capture operations are stopped before a value is written to these bits.

CLK2	CLK1	CLK0	Count clock	$\phi=16\text{ MHz}$	$\phi=8\text{ MHz}$	$\phi=4\text{ MHz}$	$\phi=1\text{ MHz}$
0	0	0	ϕ	62.5 ns	125 ns	0.25 μs	1 μs
0	0	1	$\phi/2$	125 ns	0.25 μs	0.5 μs	2 μs
0	1	0	$\phi/4$	0.25 μs	0.5 μs	1 μs	4 μs
0	1	1	$\phi/8$	0.5 μs	1 μs	2 μs	8 μs
1	0	0	$\phi/16$	1 μs	2 μs	4 μs	16 μs
1	0	1	$\phi/32$	2 μs	4 μs	8 μs	32 μs
1	1	0	$\phi/64$	4 μs	8 μs	16 μs	64 μs
1	1	1	$\phi/128$	8 μs	16 μs	32 μs	128 μs

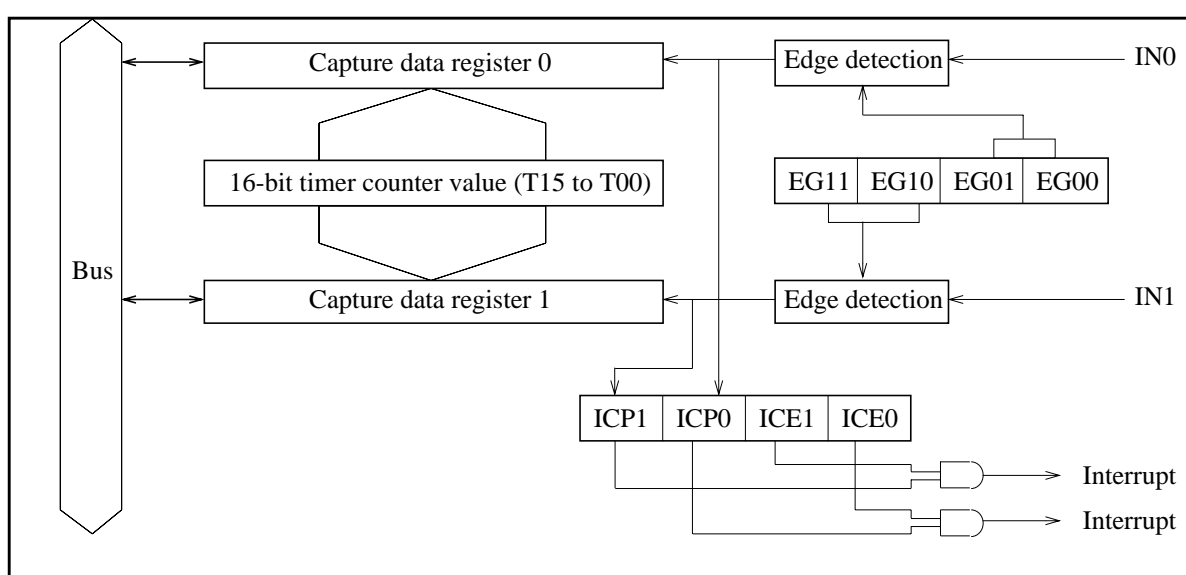
ϕ = Machine clock

12.4 Input capture

This module detects a rising or falling edge or both edges of an external input signal and stores the 16-bit free-running timer value in a register. In addition, this module can generate an interrupt upon detection of an edge. One input capture module consists of two input capture data registers and one control register. Each input capture has a corresponding external input pin. The MB90495 family contains two input capture modules: ICU0/1 and ICU2/3. Following only one module is explained (ICU0/1). See the I/O Map for the registers of the second module.

- The detection edge of an external input can be selected from three types.
- Rising edge, falling edge, or both edges
- An interrupt can be generated upon detection of a valid edge of an external input.

12.4.1 Input capture block diagram



12.4.2 Input capture register details

(1) Input capture data register (IPCP0, IPCP1)

000051H 000053H	bit	15	14	13	12	11	10	9	8	
		CP15	CP14	CP13	CP12	CP12	CP11	CP09	CP08	
		R	R	R	R	R	R	R	R	←Attribute ←Initial value
		X	X	X	X	X	X	X	X	
000050H 000052H	bit	7	6	5	4	3	2	1	0	
		CP07	CP06	CP05	CP04	CP03	CP02	CP01	CP00	
		R	R	R	R	R	R	R	R	←Attribute ←Initial value
		X	X	X	X	X	X	X	X	

This register stores the 16-bit timer value when a valid edge of the corresponding external pin input waveform is detected. (This register must be accessed in word mode. No value can be written to this register.)

(2) Input Capture Control status register (ICS01)

000054H	bit	7	6	5	4	3	2	1	0	
		ICP1	ICP0	ICE1	ICE0	EG11	EG10	EG01	EG00	
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	←Attribute ←Initial value
		X	X	0	0	0	0	0	0	

[bits 7 and 6] ICP1 and ICP0

These bits are used as input capture interrupt flags. '1' is set to this bit upon detection of a valid edge of an external input pin. While the interrupt enable bits (ICE0 and ICE1) are set, an interrupt can be generated upon detection of a valid edge.

These bits are cleared by writing '0'. Writing '1' has no effect. '1' is always read by a read-modify-write instruction.

0	No valid edge detection
1	Valid edge detection

ICP0: Corresponds to input capture 0. ICP1: Corresponds to input capture 1.

[bits 5 and 4] ICE1 and ICE0

These bits are used to enable input capture interrupts. While these bits are "1", an input capture interrupt is generated when the interrupt flag (ICP0 or ICP1) is set.

0	Interrupt disabled (default)
1	Interrupt enabled

ICE0: Corresponds to input capture 0. ICE1: Corresponds to input capture 1.

[bits 3, 2, 1, and 0] EG11, EG10, EG01, and EG00

These bits are used to specify the valid edge polarity of the external inputs. These bits are also used to enable input capture operation.

EG11 EG01	EG10 EG00	Edge detection polarity
0	0	No edge detection (stop) (default)
0	1	Rising edge detection ↑
1	0	Falling edge detection ↓
1	1	Both edge detection ↑ ↓

EG01 and EG00: Correspond to input capture 0. EG11 and EG10: Correspond to input capture 1.

12.4.3 Operations

□ 16-bit free-running timer

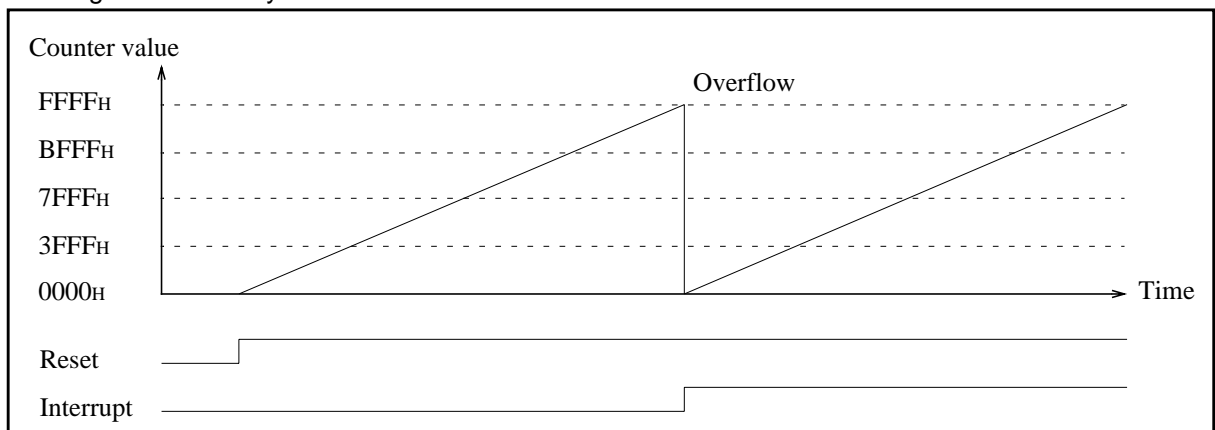
The 16-bit free-running timer starts counting from counter value '0000' after the reset is released. The counter value is used as the reference time for the 16-bit output compare and 16-bit input capture operations.

The counter value is cleared in the following conditions:

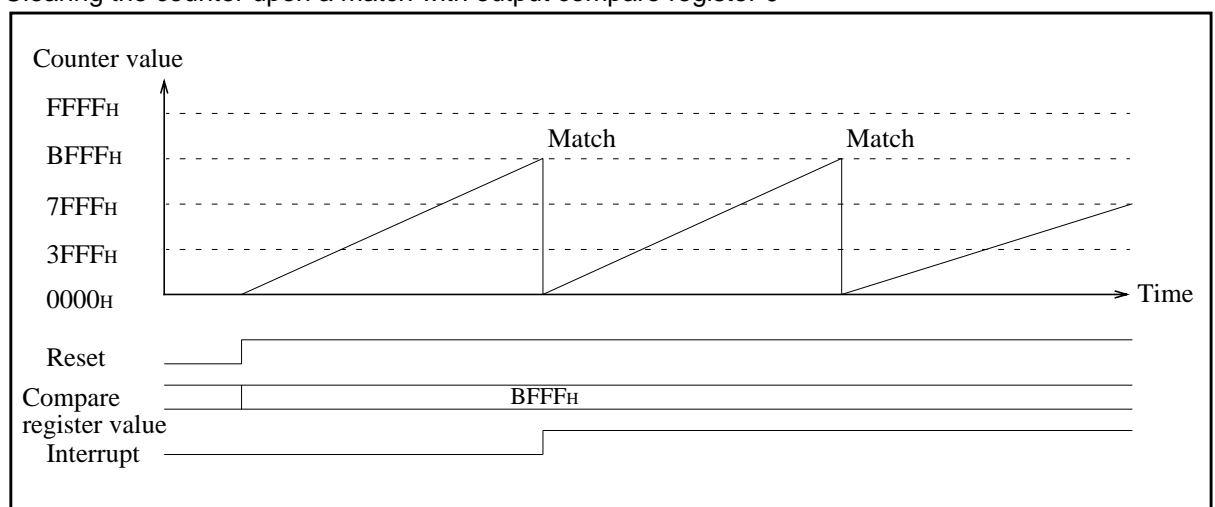
- When an overflow occurs.
- When a match with the output compare register 0 occurs. (This depends on the mode.)
- When '1' is written to the CLR bit of the TCCS register during operation.
- When '0000' is written to the TCDC register during stop.
- Reset

An interrupt can be generated when an overflow occurs or when the counter is cleared by a match with the compare register 0. (Compare match interrupts can be used only in an appropriate mode.)

■ Clearing the counter by an overflow



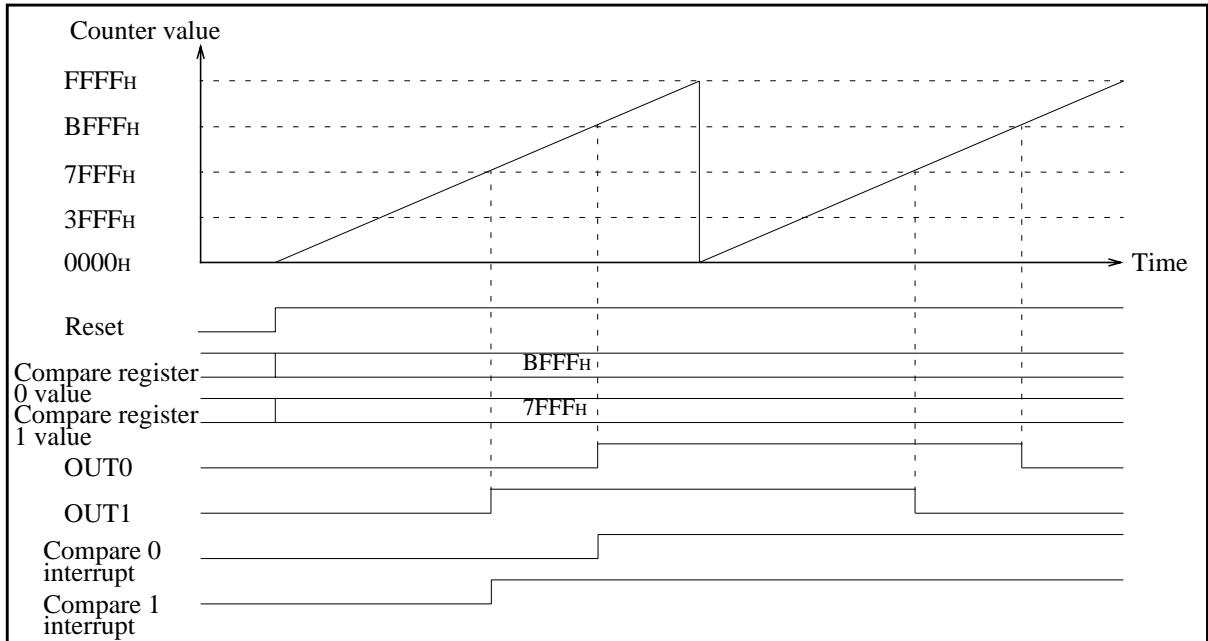
■ Clearing the counter upon a match with output compare register 0



□ 16-bit output compare

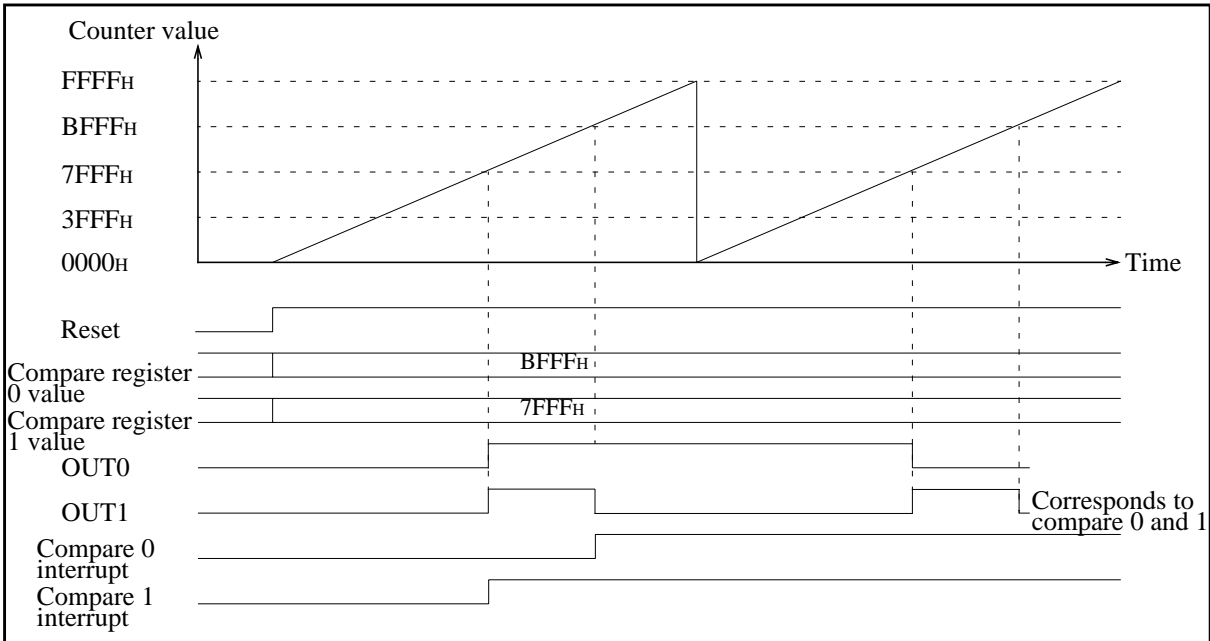
In the 16-bit output compare operation, an interrupt request flag can be set and the output level can be reversed when the specified compare register value matches the 16-bit free-run timer value.

■ Sample of output waveform when compare registers 0 and 1 are used (The initial output value is 0.)



The output level can be changed using two compare registers (when CMOD=1).

■ Sample of a output waveform with two compare registers (The initial output value is '0'.)



❑ 16-bit input capture

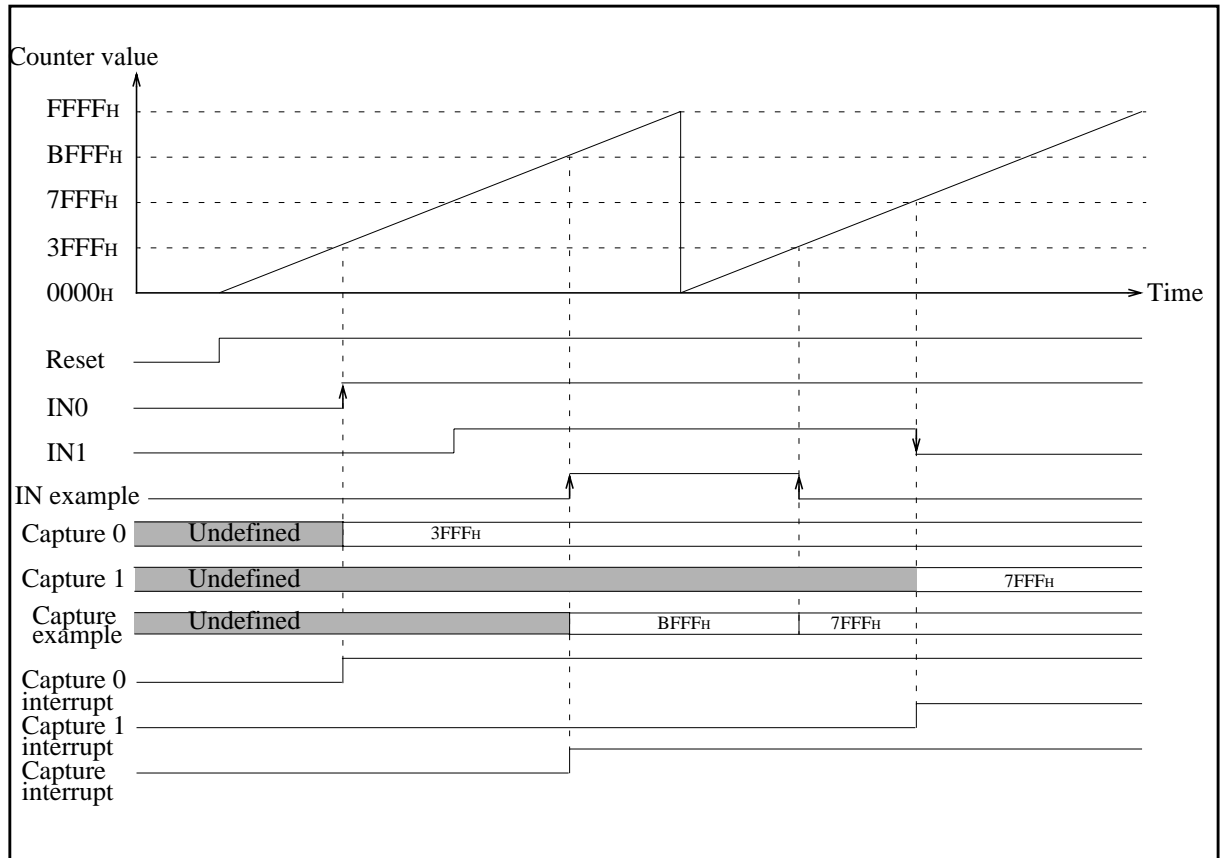
In 16-bit input capture operation, an interrupt can be generated upon detection of at the specified edge, fetching the 16-bit free-run timer value and writing it to the capture register.

■ Sample of input capture fetch timing

Capture 0: Rising edge

Capture 1: Falling edge

Capture example: Both edges

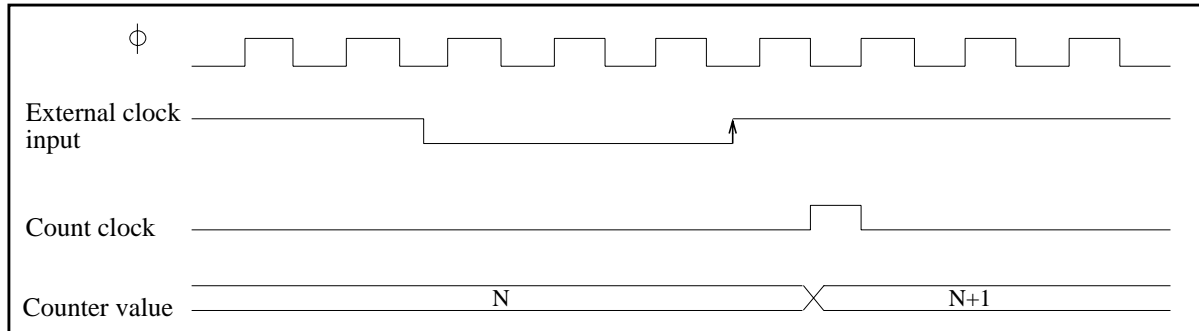


□ Timing

(1) 16-bit free-running timer count timing

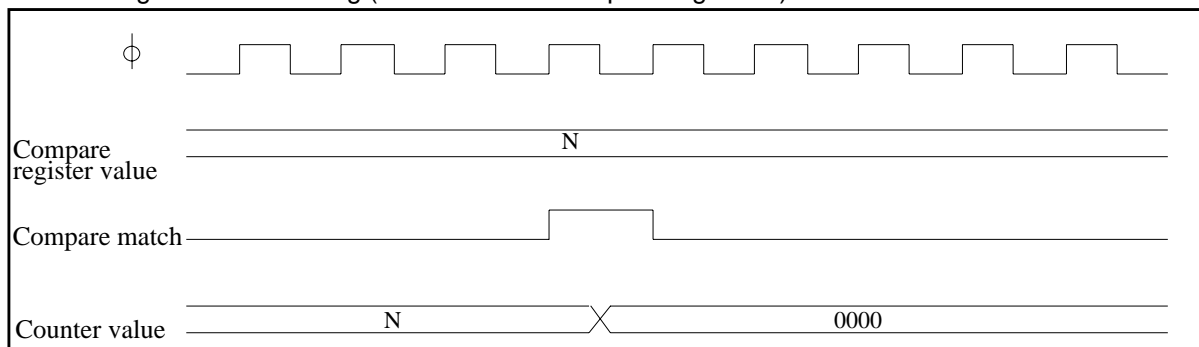
The 16-bit free-run timer is incremented based on the input clock (internal or external clock). When external clock is selected, the 16-bit free-run timer is incremented at the rising edge.

■ Free-running timer count timing



The counter can be cleared upon a reset, software clear, or a match with the compare register 0. By a reset or software clear, the counter is immediately cleared. By a match with compare register 0, the counter is cleared in synchronization with the count timing.

■ Free-running timer clear timing (match with the compare register 0)

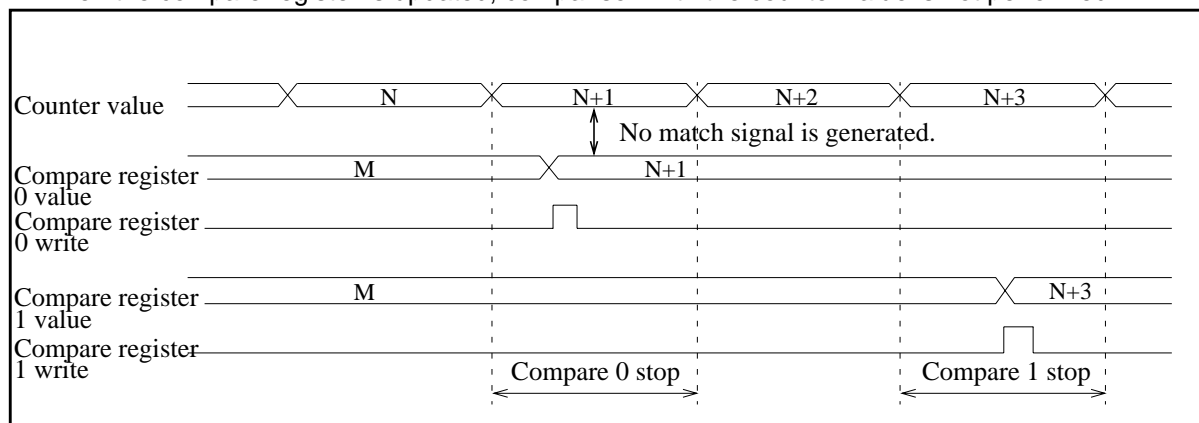


(2) Output compare timing

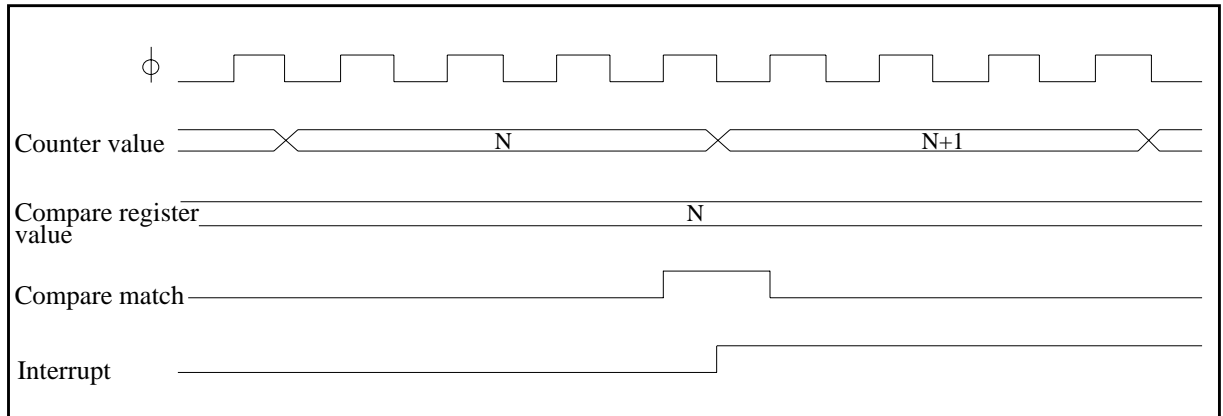
In output compare operation, a compare match signal is generated when the free-running timer value matches the specified compare register value. The output value can be reversed and an interrupt can be issued. The output reverse timing upon a compare match is synchronized with the counter count timing.

■ Compare operation upon update of compare register

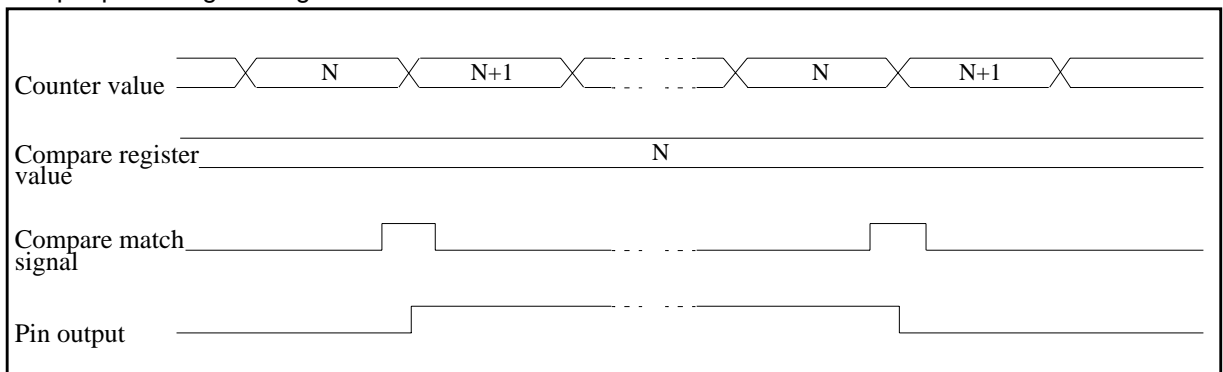
When the compare register is updated, comparison with the counter value is not performed.



■ Interrupt timing

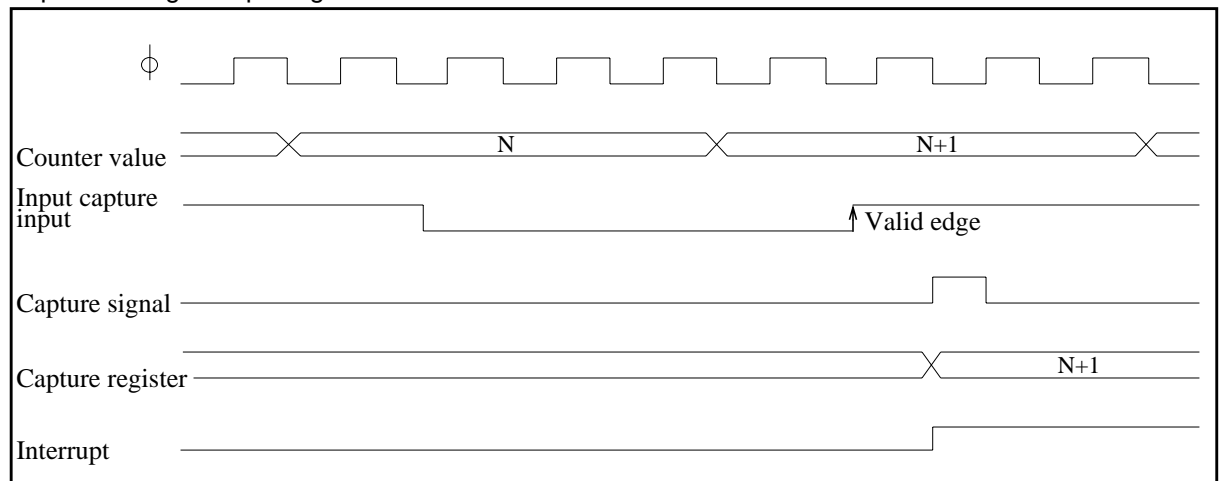


■ Output pin change timing



(3) Input capture input timing

■ Capture timing for input signals



CHAPTER 13 16-BIT RELOAD TIMER

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

- 13.1 Overview of the 16-Bit Reload Timer
- 13.2 Configuration of the 16-Bit Reload Timer
- 13.3 16-Bit Reload Timer Pins
- 13.4 16-Bit Reload Timer Registers
- 13.5 16-Bit Reload Timer Interrupts
- 13.6 Operation of the 16-Bit Reload Timer
- 13.7 Usage Notes on the 16-Bit Reload Timer
- 13.8 Sample Programs for the 16-Bit Reload Timer

13.1 Overview of the 16-Bit Reload Timer

The 16-bit reload timer functions in the two modes shown below, either of which can be selected. It 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 load 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 (EI²OS).

The MB90495 series 16-bit reload timer has two built-in channels.

- **16-bit reload timer operating modes**

Table 13.1-1 lists the operating modes of the 16-bit reload timer.

Table 13.1-1 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 0000_H to FFFF_H) 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 TOUT0 and TOUT1 pins.

Table 13.1-2 lists the intervals for the 16-bit reload timer.

Table 13.1-2 Intervals for the 16-bit reload timer

Count clock	Count clock period	Interval
Internal clock	$2^1/\phi$ (0.125 s)	0.125 s to 8.192 ms
	$2^3/\phi$ (0.5 s)	0.5 s to 32.768 ms
	$2^5/\phi$ (2.0 s)	2.0 s to 131.1 ms
External clock	$2^3/\phi$ or more (0.5 s)	0.5 s or more

ϕ : Machine clock

Values in parentheses are for a 16 MHz machine clock.

● Single-shot mode

When an underflow (from 0000_H to FFFF_H) 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 TOUT0 and TOUT1 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 13.1-3 lists 16-bit reload timer interrupts and EI²OS.

Table 13.1-3 16-bit reload timer interrupts and EI²OS

Channel	Interrupt number	Interrupt control register		Vector table address			EI ² OS
		Register name	Address	Lower	Upper	Bank	
16-bit reload timer 0	#17 (11 _H)	ICR03	0000B3 _H	FFFFB8 _H	FFFFB9 _H	FFFFBA _H	Δ
16-bit reload timer 1	#36 (24 _H)	ICR12	0000BC _H	FFFF6C _H	FFFF6D _H	FFF6E _H	

Δ: Available, when the interrupt causes sharing the interrupt vector are not used.

13.2 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 (TMRL)
- Timer control status register (TMCSR)

■ Block diagram of the 16-bit reload timer

Figure 13.2-1 shows the block diagram of the 16-bit reload timer.

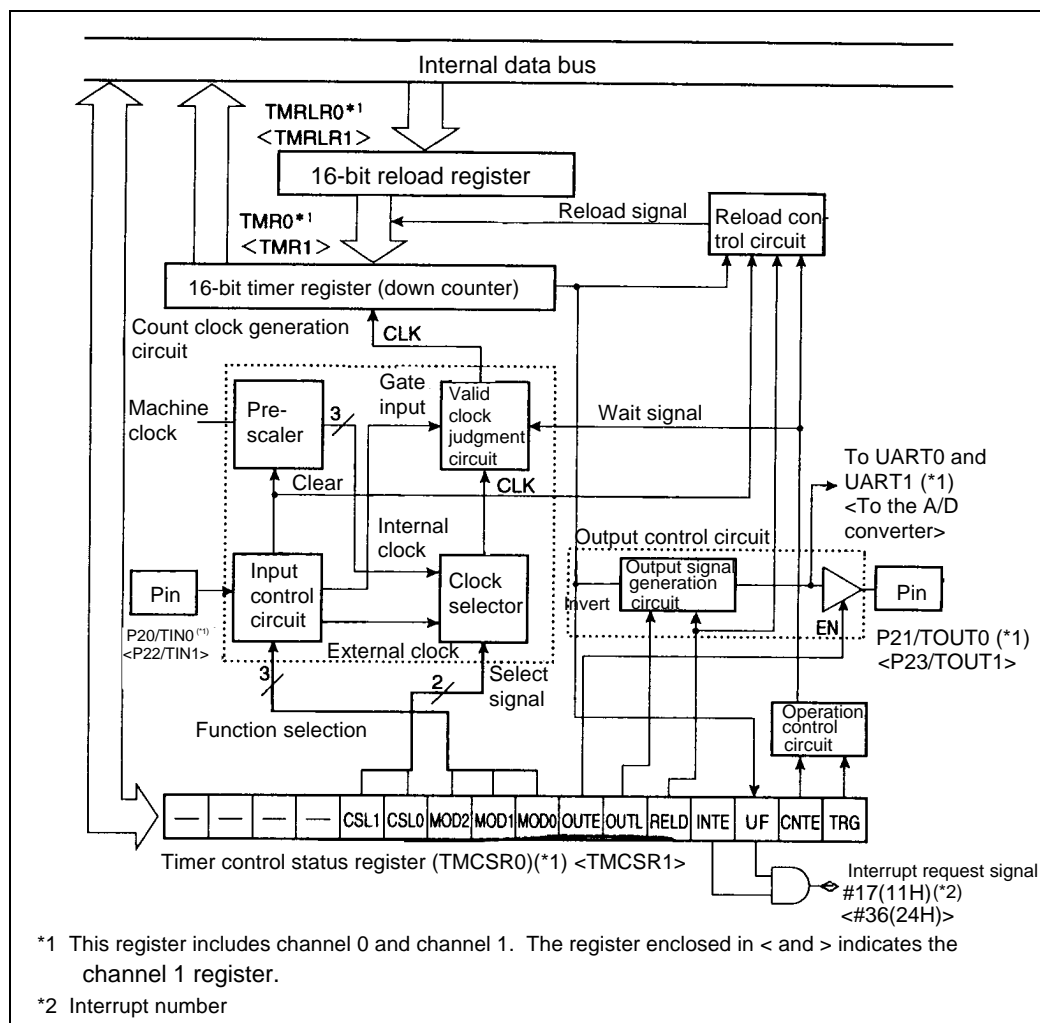


Figure 13.2-1 Block diagram of the 16-bit reload timer

- **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 TOUT pin output by an underflow of the 16-bit timer register and enabling and disabling of TOUT 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 (TMRL)**

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.

13.3 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 13.3-1 lists the functions of the pins, I/O format, and settings required to use the 16-bit reload timer.

Table 13.3-1 16-bit reload timer pins

Pin name	Pin function	I/O format	Standby control	Settings required for pins
P20/TIN0	Port 2 input-output/timer 0 input	CMOS output/CMOS hysteresis input	Available	Setting for the input port (DDR2: bit 0 = 0)
P21/TOUT0	Port 2 input-output/timer 0 output			Setting for timer output enable (TMCSR0: OUTE = 1)
P22/TIN1	Port 2 input-output/timer 1 input			Setting for the input port (DDR2: bit 2 = 0)
P24/TOUT1	Port 2 input-output/timer 1 output			Setting for timer output enable (TMCSR1: OUTE = 1)

■ Block diagram of the 16-bit reload timer pins

Figure 13.3-1 shows the block diagram of the 16-bit reload timer pins.

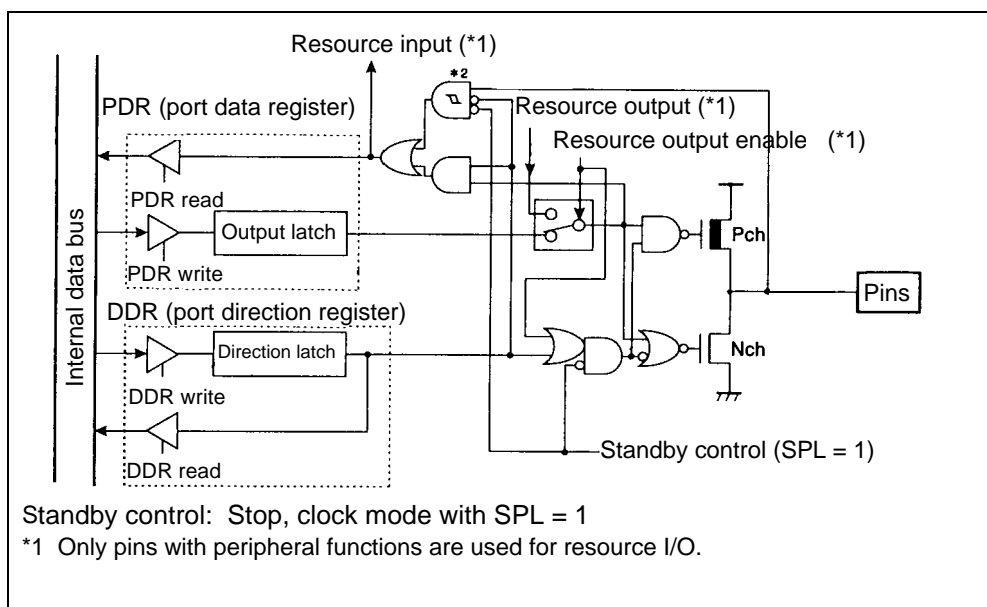


Figure 13.3-1 Block diagram of the 16-bit reload timer pins

13.4 16-Bit Reload Timer Registers

The 16-bit reload timer registers are as follows.

■ 16-bit reload timer registers

Figure 13.4-1 shows 16-bit reload timer registers.

	Address	bit15 bit8	bit7 bit0
16-bit reload timer 0	000066H,67H	TMCSR0 (timer control status register)	
	003900H,01H	TMR0/TMRL0 (16-bit timer register/16-bit reload register) (*1)	
16-bit reload timer 1	000068H,69H	TMCSR1 (timer control status register)	
	003902H,03H	TMR1/TMRL1 (16-bit timer register/16-bit reload register) (*1)	

*1 This register functions as a 16-bit timer register (TMR) during reading, and functions as a 16-bit reload register (TMRL) during writing.

Figure 13.4-1 16-bit reload timer registers

13.4 16-Bit Reload Timer Registers

13.4.1 Timer control status register, high part (TMCSR0, TMCSR1: H)

High-order bits 11 to 8 and low-order 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 low-order bit 7: the MOD0 bit.

■ Timer control status register, high part (TMCSR0:H, TMCSR1:H)

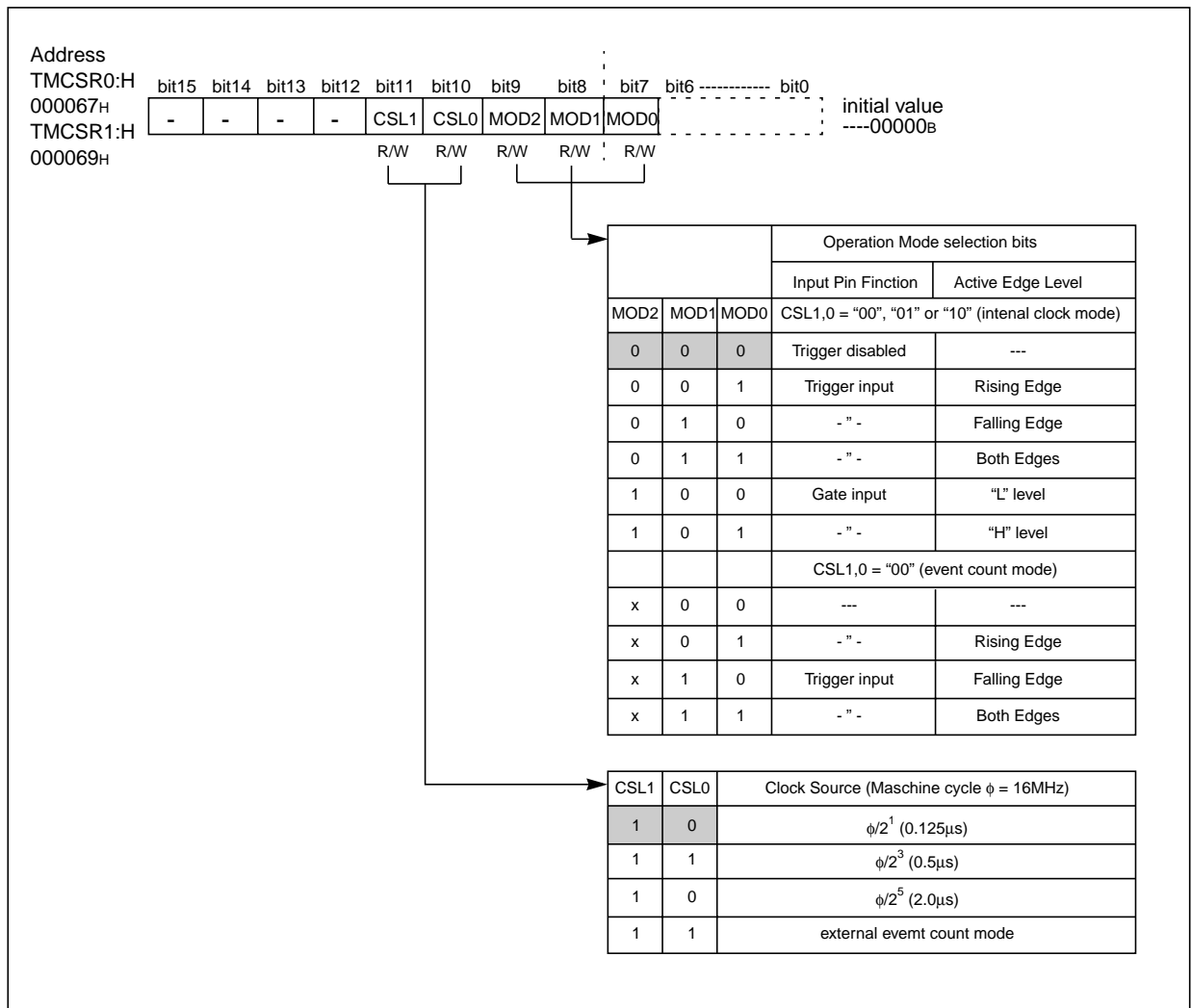


Figure 13.4.1-1 Timer control status register, high part (TMCSR0, TMCSR1: H)

Table 13.4.1-1 Function description of each bit of the high part of the timer control status register (TMCSR0, TMCSR1: H)

Bit name		Function
bit15 bit14 bit13 bit12	Not used	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<p><In internal clock mode></p> <ul style="list-style-type: none"> 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. <p><In event count mode></p> <ul style="list-style-type: none"> 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.

13.4 16-Bit Reload Timer Registers

13.4.2 Timer control status register, low 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)

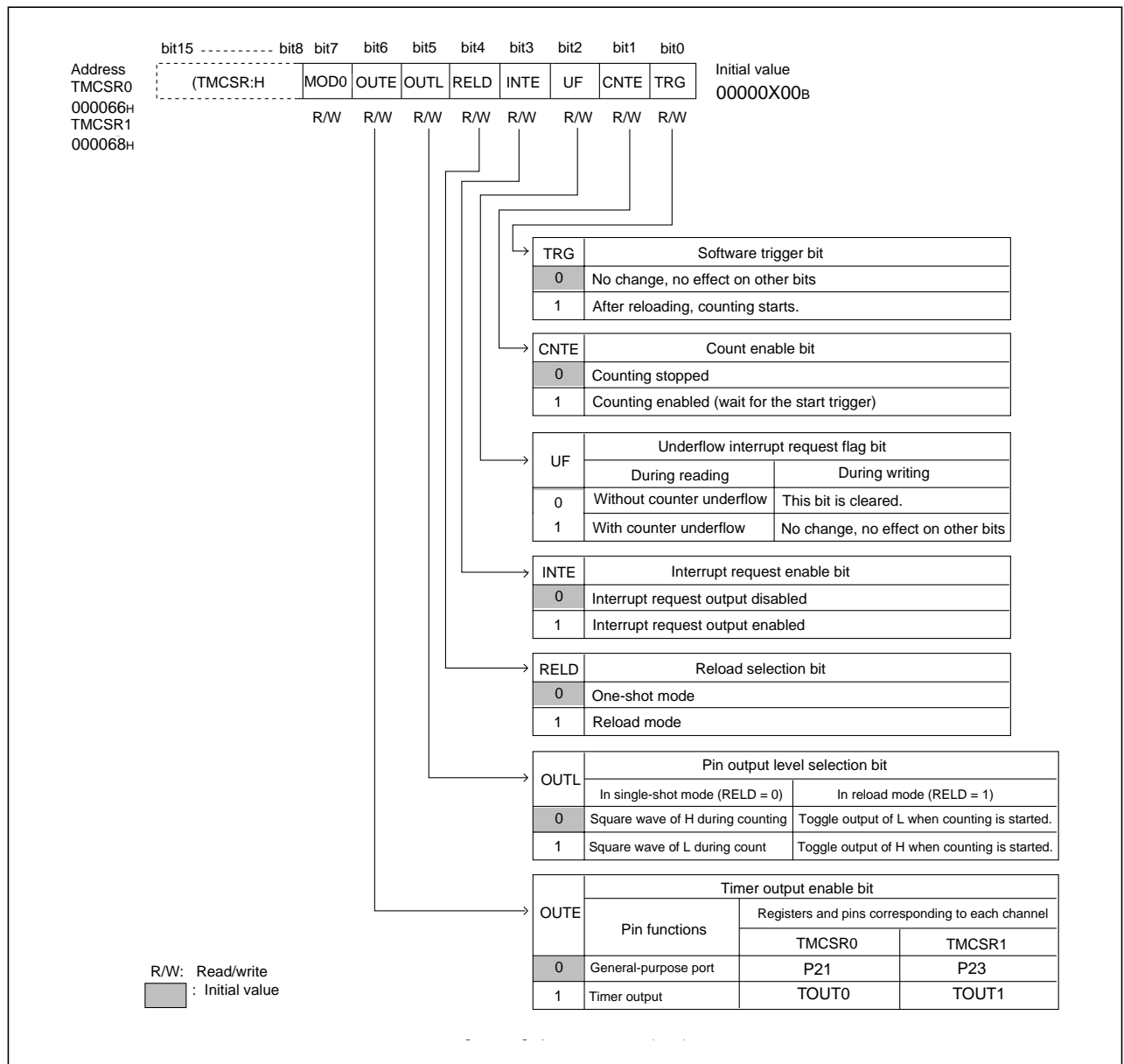


Figure 13.4.2-1 Timer control status register, low part (TMCSR0, TMCSR1: L)

Table 13.4.2-1 Function description of each bit of the low part of the timer control status register (TMCSR0, TMCSR1: L)

Bit name		Function
bit7	MOD0	<ul style="list-style-type: none"> see Section 13.4.1 "Timer control status register, high part (TMCSR0, TMCSR1: H)"
bit6	OUTE: Timer output enable bit	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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 EI²OS is activated.
bit1	CNTE: Count enable bit	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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.

13.4.3 16-bit timer register (TMR0, TMR1)

The 16-bit timer register (TMR0, TMR1) is always able to read the count value from the 16-bit down counter.

■ 16-bit timer register (TMR0, TMR1)

Figure 13.4.3-1 shows the 16-bit timer registers (TMR0, TMR1).

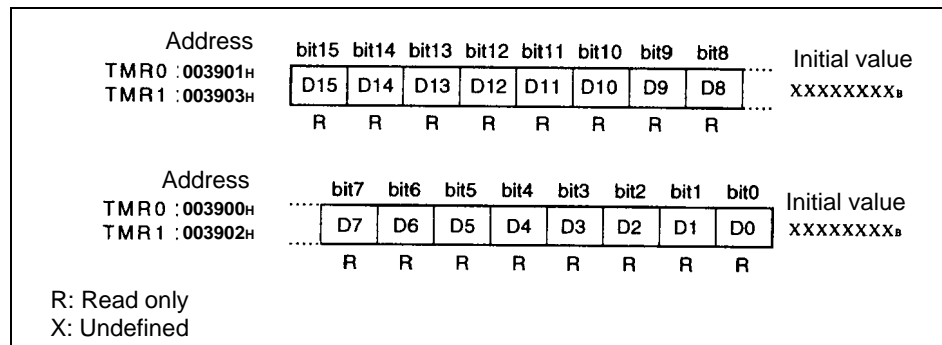


Figure 13.4.3-1 16-bit timer register (TMR0, TMR1)

The 16-bit timer register is able 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 in the counter 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, 003AH, etc.).
- Although the 16-bit timer register (TMR0, TMR1) is a read-only register, it is allocated to the same address as the address of the write-only 16-bit reload register (TMRL1, TMRL0). Accordingly, writing to this register has no effect on the TMR0 and TMR1 value, although writing to TMRL1 and TMRL0 is done.

13.4.3 16-bit reload register (TMRL1, TMRL0)

The 16-bit reload register (TMRL1, TMRL0) sets a reload value in the 16-bit down counter. The value written to this register is loaded into the down counter, and the value is counted down.

■ 16-bit reload register (TMRL1, TMRL0)

Figure 13.4.3-1 shows the 16-bit reload registers (TMRL1, TMRL0).

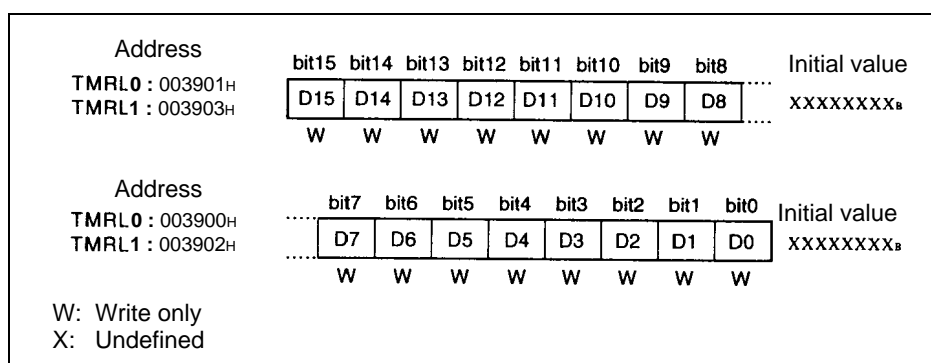


Figure 13.4.3-1 16-bit reload register (TMRL1, TMRL0)

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 (TMRL1, TMRL0) is reloaded into the counter when an underflow occurs, and counting down continues. In single-shot mode, the counter stops at FFFF_H 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 003AH,A etc.) to write a value to this register.
- Although the 16-bit reload register (TMRL1, TMRL0) is functionally a write-only register, it is allocated to the same address as one of the read-only 16-bit timer registers (TMR0, TMR1). Accordingly, since the read value is used as the TMR0 and TMR1 value, the INC/DEC instruction and other instructions for read-modify-write (RMW) operations cannot be used.

13.5 16-Bit Reload Timer Interrupts

The 16-bit reload timer is enabled to generate an interrupt request in an underflow of the counter. It is also coordinated with the extended intelligent I/O service (EI²OS).

■ 16-bit reload timer interrupts

Table 13.5 -1 lists the interrupt control bits and interrupt causes of the 16-bit reload timer.

Table 13.5 -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 (TMRO)	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 0000_H to FFFF_H) 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 13.5 -2 lists the 16-bit reload timer interrupts and EI²OS.

Table 13.5 -2 16-bit reload timer interrupts and EI²OS

Channel	Interrupt number	Interrupt control register		Vector table address			EI ² OS
		Register name	Address	Lower	Upper	Bank	
16-bit reload timer 0 (*1)	#17 (11 _H)	ICR03	0000B3 _H	FFFFB8 _H	FFFFB9 _H	FFFFBA _H	Δ
16-bit reload timer 1	#36 (24 _H)	ICR12	0000BC _H	FFFF6C _H	FFFF6D _H	FFF6E _H	

Δ: Available, when the interrupt causes sharing the interrupt vector are not used.

*1 The same interrupt number as that for A/D-Converter is assigned to 16-bit reload timer 0.

■ 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 0 uses EI²OS, interrupts of the A/D-Converter must be disabled.

13.6 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 13.6-1 is required to operate this timer as an interval timer.

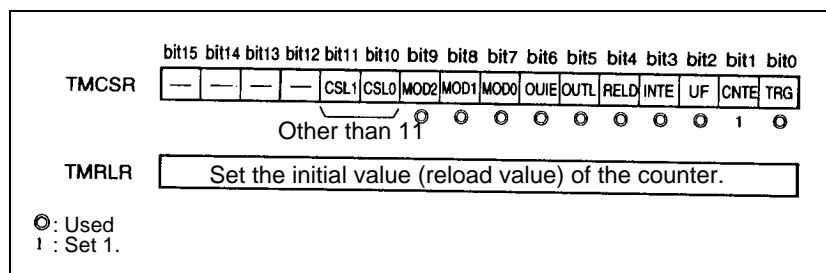


Figure 13.6-1 Internal clock mode setting

● Event counter mode setting

The setting shown in Figure 13.6-2 is required to operate this timer as an event counter.

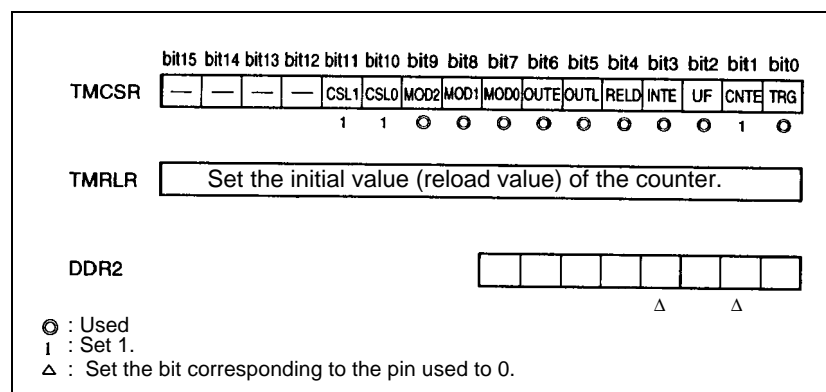


Figure 13.6-2 Event counter mode setting

■ 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 13.6-3 shows the transitions of these counter statuses.

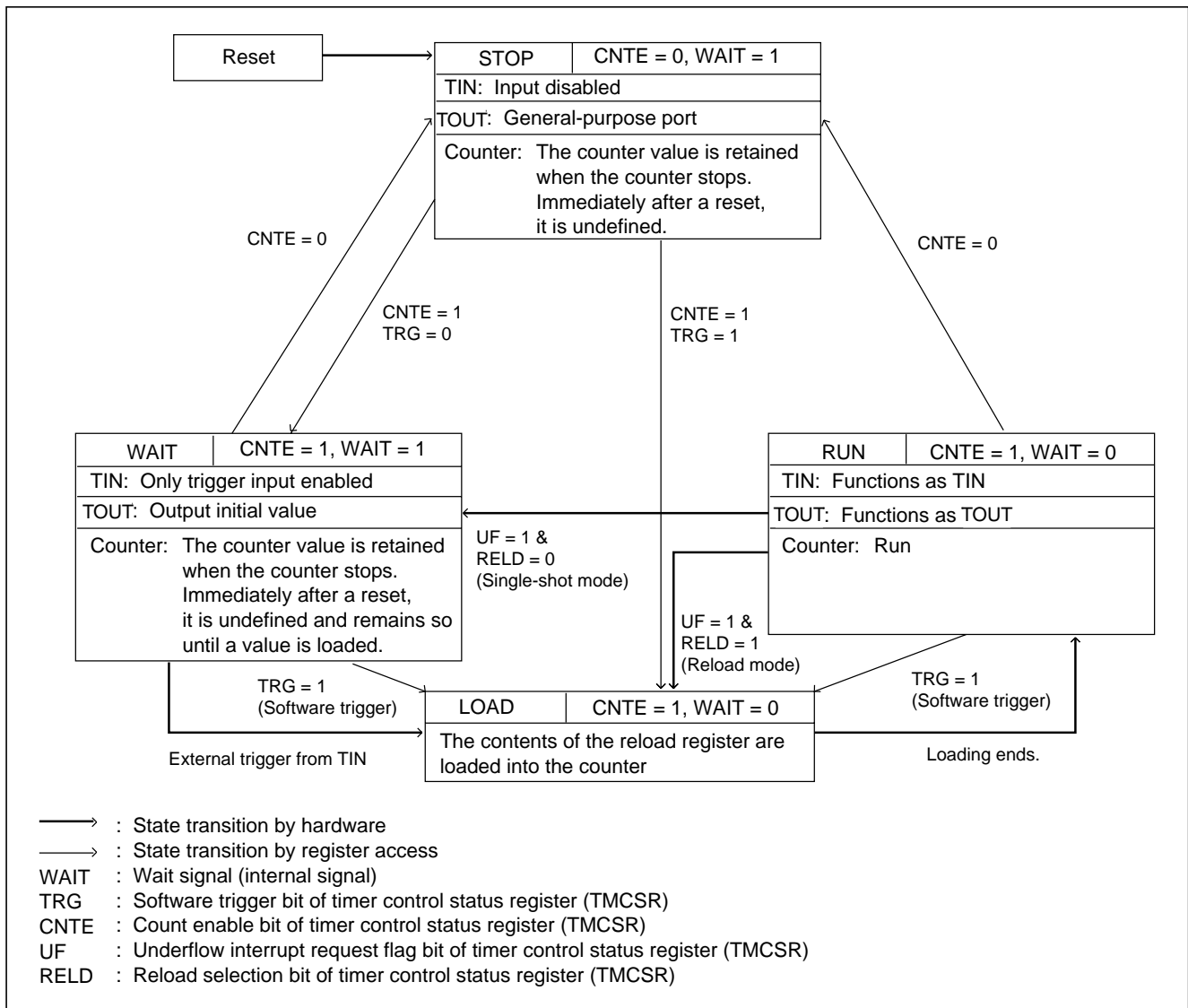


Figure 13.6-3 Counter status transition

13.6.1 Internal clock mode (reload mode)

Synchronized with the internal count clock, the 16-bit reload timer is used for counting down of the 16-bit counter, generating an interrupt request to the CPU for a counter underflow. It can also output a toggle waveform from the timer output pin.

■ Operation in internal clock mode (reload 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 (TMRL) 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 0000H to FFFFH) occurs, the value of the 16-bit reload register (TMRL) 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 from the TOUT 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 13.6.1-1 shows software trigger operation in reload mode.

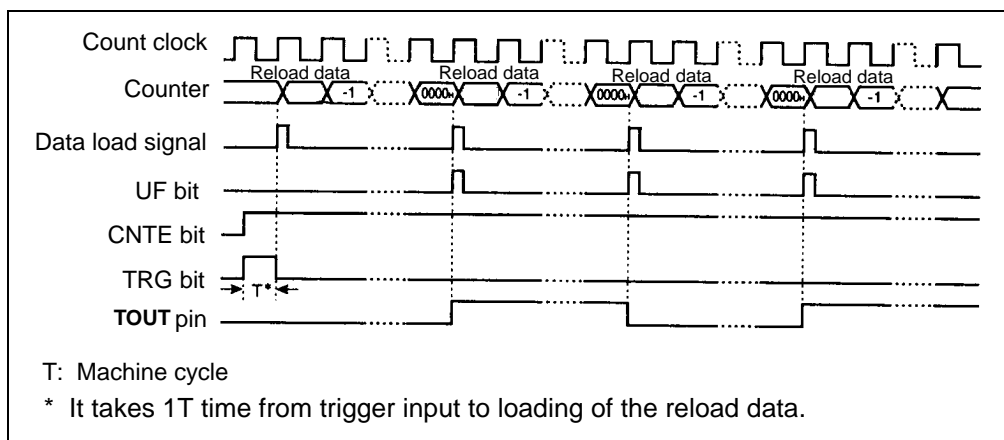


Figure 13.6.1-1 Count operation in reload mode (software trigger operation)

● 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 13.6.1-2 shows external trigger operation in reload mode.

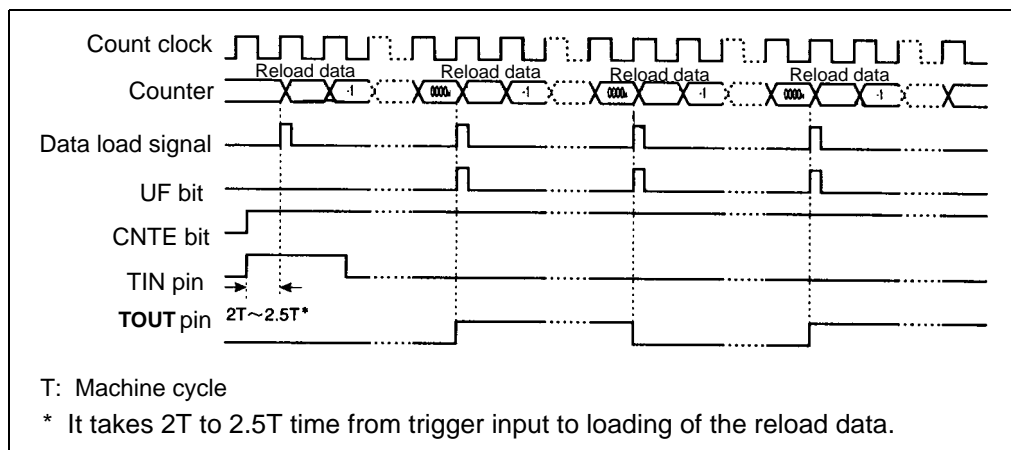


Figure 13.6.1-2 Counting in reload mode (external trigger operation)

<Check>

Specify $2/\phi$ 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 13.6.1-3 shows gate input in reload mode.

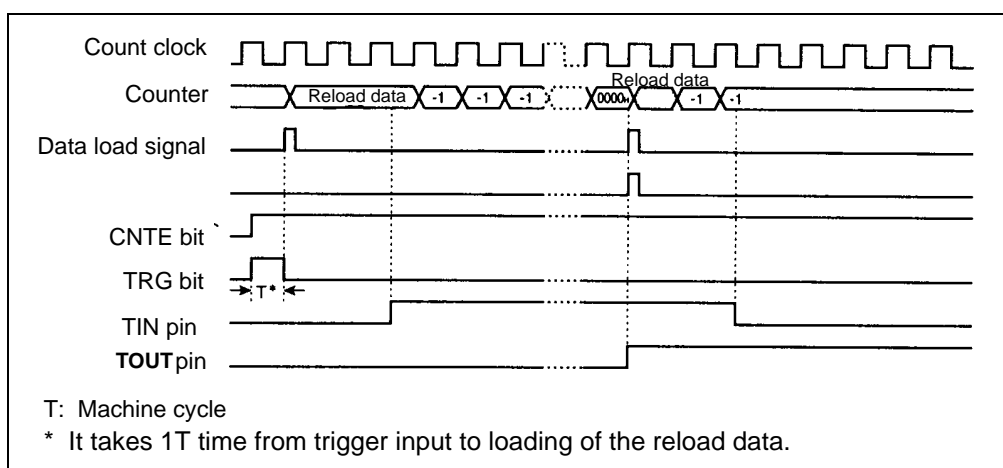


Figure 13.6.1-3 Count operation in reload mode (software trigger and gate input operation)

<Check>

Specify $2/\phi$ or more for the width of the trigger pulse input to the TIN pin.

13.6.2 Internal clock mode (single-shot mode)

Synchronized with the internal count clock, the 16-bit reload timer is used for counting down of the 16-bit counter, generating an interrupt request to the CPU for a counter underflow. It can also output from the TOUT pin a rectangular waveform indicating that counting is in progress.

■ Internal clock mode (single-shot 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 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 from the TOUT 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 13.6.2-1 shows the software trigger operation in single-shot mode.

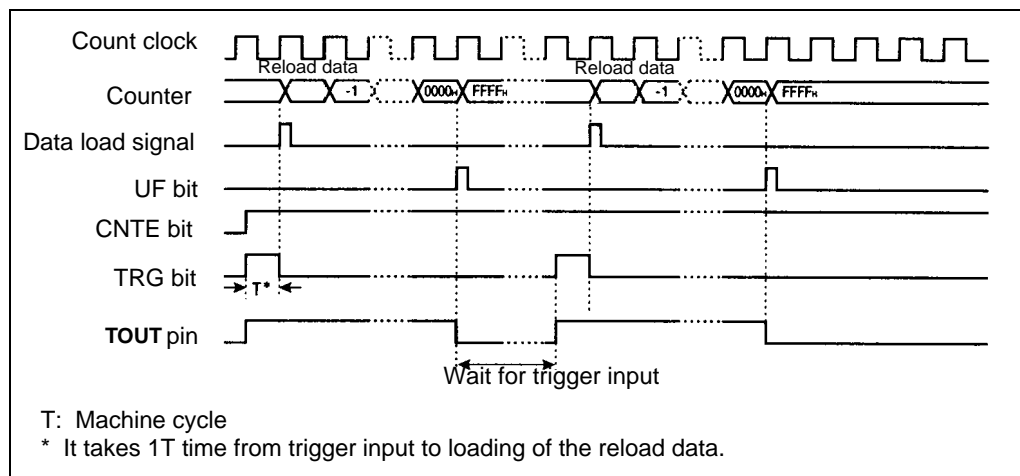


Figure 13.6.2-1 Count operation in single-shot mode (software trigger operation)

● 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 13.6.2-2 shows external trigger operation in single-shot mode.

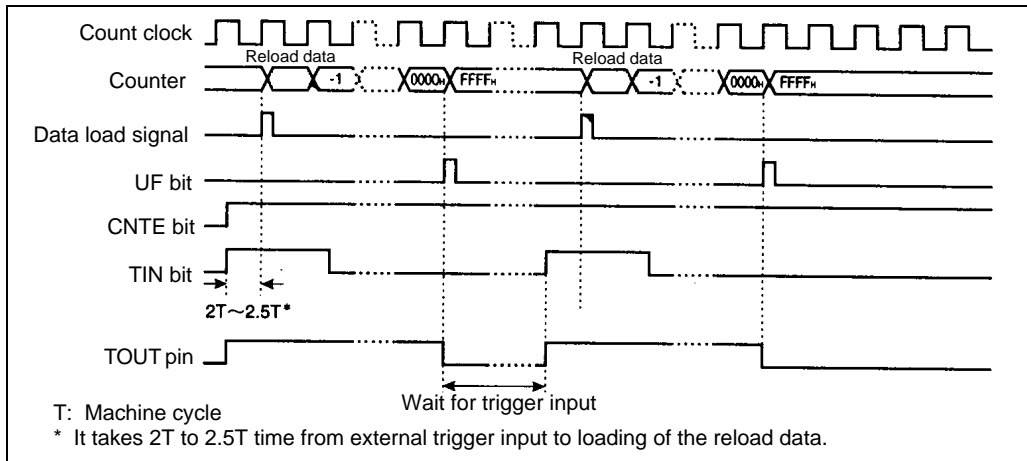


Figure 13.6.2-2 Count operation in single-shot mode (external trigger operation)

<Check>

Specify $2/\phi$ 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 13.6.2-3 shows gate input operation in single-shot mode.

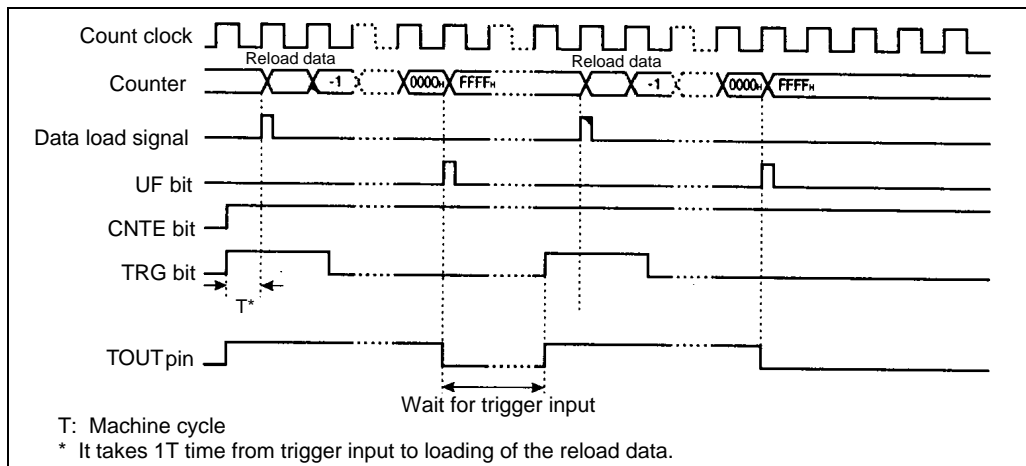


Figure 13.6.2-3 Count operation in single-shot mode (software trigger and gate input operation)

<Check>

Specify $2/\phi$ or more for the width of the trigger pulse input to the TIN pin.

13.6.3 Event count mode

The 16-bit reload timer counts an input edge from the TIN pin, counts down the 16-bit counter, and generates an interrupt request to the CPU for a counter underflow. It can also output a toggle waveform or rectangular waveform from the TOUT 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 (TMRL) 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 from the TOUT pin a toggle waveform, which is inverted for each underflow.

Figure 13.6.3-1 shows counting in reload mode.

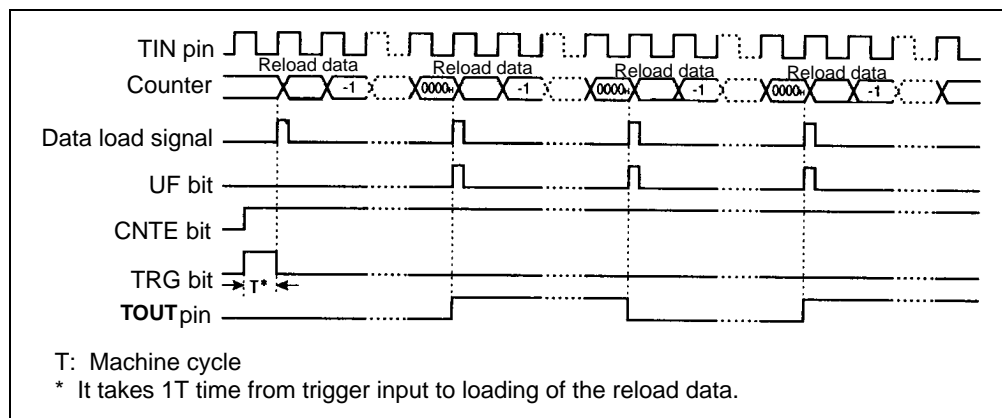


Figure 13.6.3-1 Count operation in reload mode (event count mode)

<Check>

Specify $4/\phi$ 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 from the TOUT pin a rectangular waveform indicating that counting is in progress.

Figure 13.6.3-2 shows counting in single-shot mode.

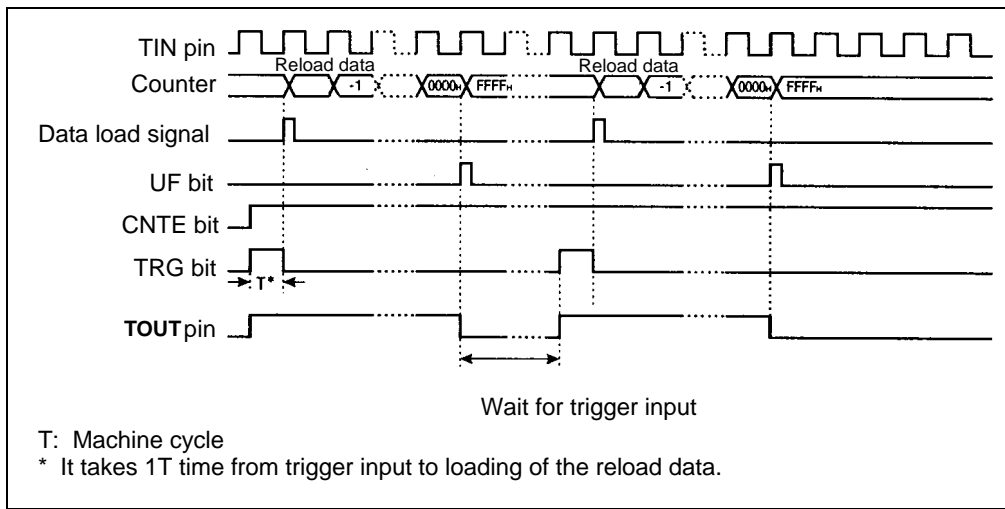


Figure 13.6.3-2 Counter operation in single-shot mode (event count mode)

<Check>

Specify $4/\phi$ or more for the H and L widths of the clock input to the TIN pin.

13.7 Usage Notes on the 16-Bit Reload Timer

Notes on using the 16-bit reload timer are given below.

■ Usage notes on the 16-bit reload timer

● Notes on using a program for setting

- Write a value to the 16-bit reload register (TMRL) 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 shares an interrupt vector with the 8/16 bit PPG timer, interrupt causes must be checked carefully by the interrupt processing routine when interrupts are used.

Also, when EI²OS is used by the 16-bit reload timer, 8/16 bit PPG timer interrupts must be disabled.

13.8 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.
- EI²OS is not used.
- 16 MHz is used for the machine clock, and 2 s is used for the count clock.

● Coding example

```
ICR03 EQU 0000B3H      ; Interrupt control register for the 16-bit reload timer 0
TMCSR EQU 000066H      ; Timer control status register
TMR EQU 003900H        ; 16-bit timer register
TMRL EQU 003900H        ; 16-bit reload register
UF EQU TMCSR:2         ; Interrupt request flag bit
CNTE EQU TMCSR:1       ; Counter operation enable bit
TRG EQU TMCSR:0        ; Software trigger bit
```

;-----Main program-----

```
CODE CSEG
```

```
START:
```

```
:: ; Assumes that stack pointer (SP) has already been
    ; initialized.
    AND CCR,#0BFH ; Interrupt disable
    MOV I:ICR03,#00H ; Interrupt level 0 (strongest)
    CLRB I:CNTE ; Temporary stopping of counter
    MOVW I:TMRL,#30D4H ; Sets data for 25-ms timer.
    MOVW I:TMCSR,#0000100000011011B
    ; Interval timer operation, 2 us 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

        CODE    ENDS

;-----Vector setting-----

VECT    CSEG    ABS=0FFH
        ORG     0FFB8H            ; Sets vector for interrupt #17 (11H).
        DSL     WARI
        ORG     0FFDCH            ; Sets reset vector
        DSL     START
        DB      00H                ; Sets single-chip mode.
VECTENDS
        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.
- EI²OS is not used.

● Coding example

```

ICR03    EQU    0000B3H            ; Interrupt control register for the 16-bit reload timer 0
TMCSR    EQU    000066H            ; Timer control status register
TMR      EQU    003900H            ; 16-bit timer register
TMRL     EQU    003900H            ; 16-bit reload register
DDR2     EQU    000012H            ; Port data register
UF        EQU    TMCSR:2            ; Interrupt request flag bit
CNTE     EQU    TMCSR:1            ; Counter operation enable bit
TRG       EQU    TMCSR:0            ; Software trigger bit

```

;-----Main program-----

```

CODECSEG
START:
;:                ; Assumes that stack pointer (SP) has already been
;                ; initialized.

```

```

        AND    CCR,#0BFH      ; Interrupt disable
        MOV    I:ICR03,#00H   ; Interrupt level 0 (strongest)
        MOV    I:DDR2,#00H    ; Sets P20/TIN0 pin to input.
        CLRB   I:CNTE         ; Temporary stopping of counter
        MOVW   I:TMRL,#2710H   ; Sets reload value to 10,000.
        MOVW   I:TMCSR,#0000110010001011B
                                   ; Counter operation, external trigger, rising
                                   ; Disables edge and external output.
                                   ; Selects single-shot mode and enables interrupts.
                                   ; Clears interrupt flag, starts counter.
        MOV    ILM,#07H       ; Sets ILM in PS to level 7.
        OR     CCR,#40H       ; Interrupt enable
LOOP:   MOV    A,#00H         ; Endless loop
        MOV    A,#01H         ;
        BRA    LOOP           ;

```

;-----Interrupt program-----

```

WARI:
        CLRB   I:UF           ; clears interrupt request flag.
;:
        ; User processing
;:
        RETI                  ; Returns from interrupt.

```

CODEENDS

;-----Vector setting-----

```

VECT    CSEG    ABS=0FFH
        ORG     0FFB8H        ; Sets vector for interrupt #17 (11H).
        DSL     WARI
        ORG     0FFDCH        ; Sets reset vector
        DSL     START
        DB      00H          ; Sets single-chip mode.

```

```

VECT ENDS
        END     START

```

CHAPTER 14 8/16-Bit PPG

The MB90495 Series contains two PPGs. The following sections only describe the functionality of the PPG 0/1. The PPG2/3 has the identical function and the register addresses should be found in the I/O map.

14.1 Overview

14.2 Registers

14.3 Block diagram

14.4 Register details

14.5 Operations

14.1 Overview

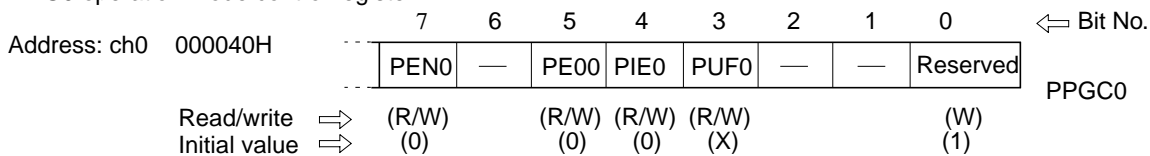
The 8/16-bit Programmable Pulse Generator (PPG) consists of two eight-bit down counters, four eight-bit reload registers, one 16-bit control register, two external pulse output signals, and two interrupt outputs. The following functions are implemented:

- 8-bit PPG output, 2-channel independent operation mode: Two independent channels of PPG output operation are implemented.
- 16-bit PPG output operation mode: One channel of 16-bit PPG output operation is implemented.
- 8+8-bit PPG output operation mode: 8-bit PPG output operation is implemented at specified intervals, channel 0 output as channel 1 clock input.

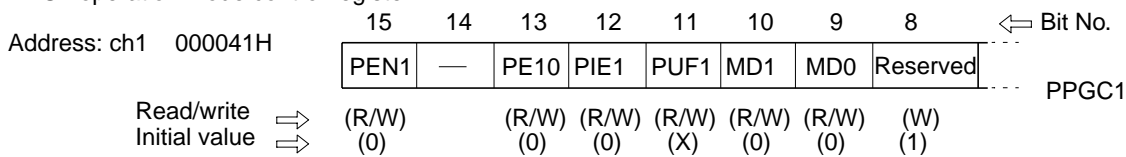
PPG output operation: Pulse waves are output at specified intervals and duty ratio. With an external circuit, this module can be used as a D/A converter.

14.2 Registers

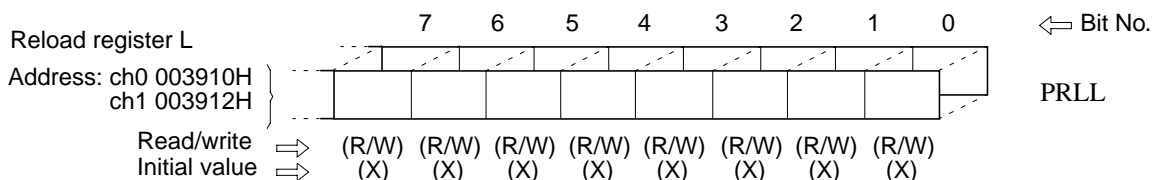
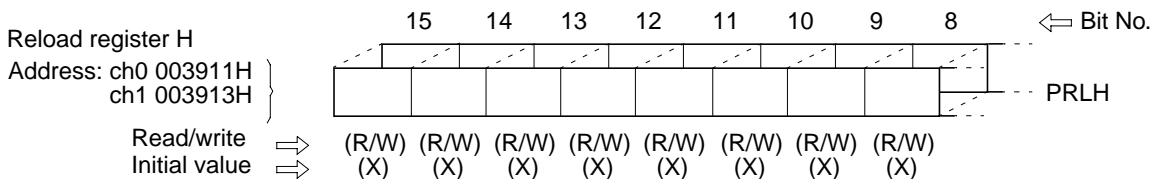
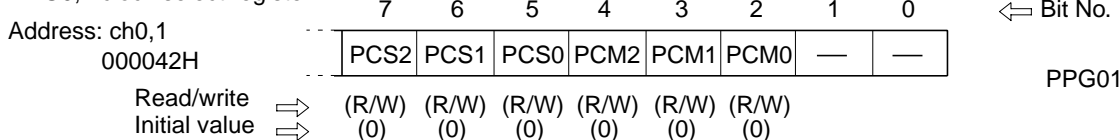
PPG0 operation mode control register



PPG1 operation mode control register



PPG0,1 clock select register



14.3 Block diagram

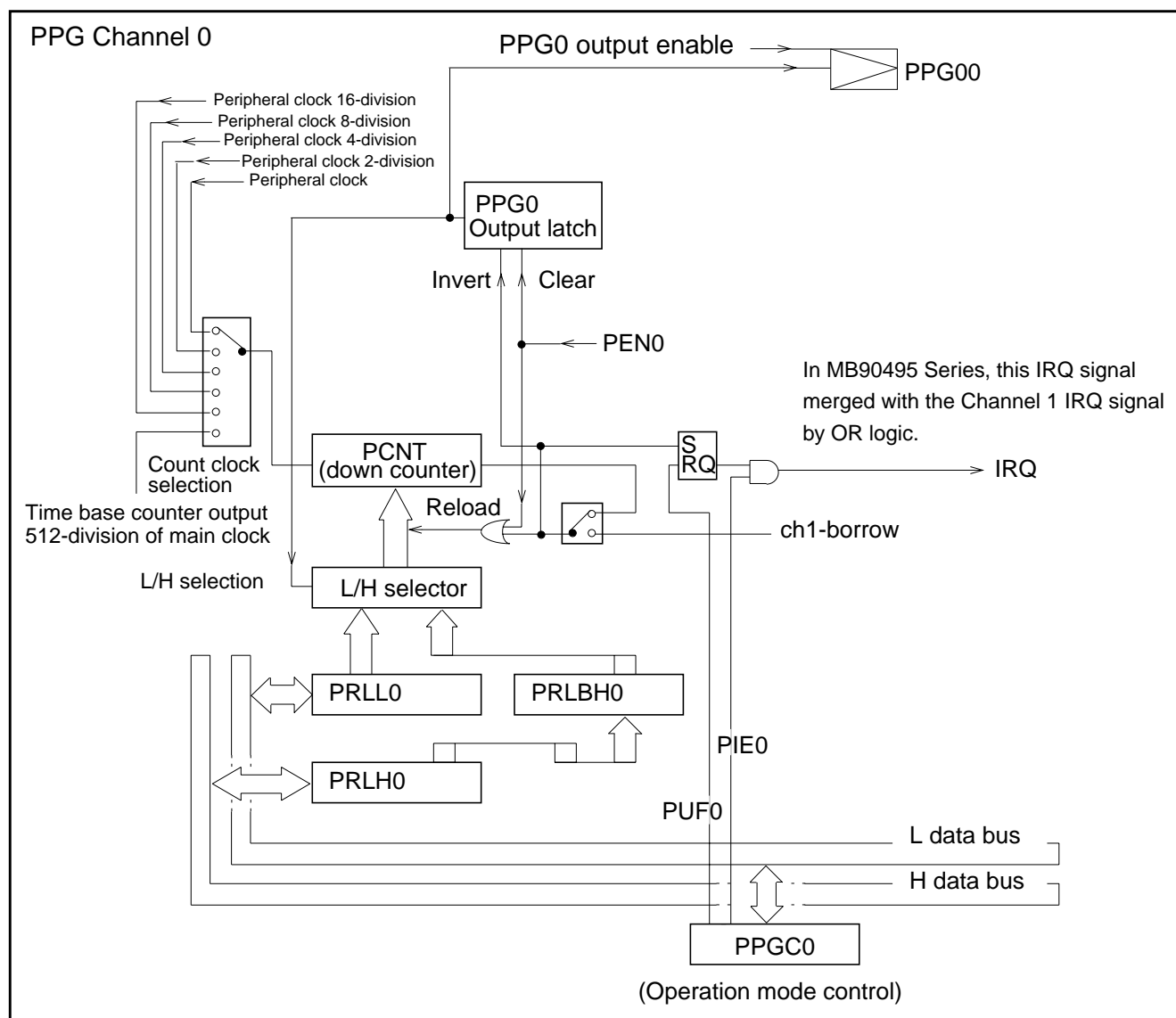


Figure 14.3-1 8-bit PPG ch0 block diagram

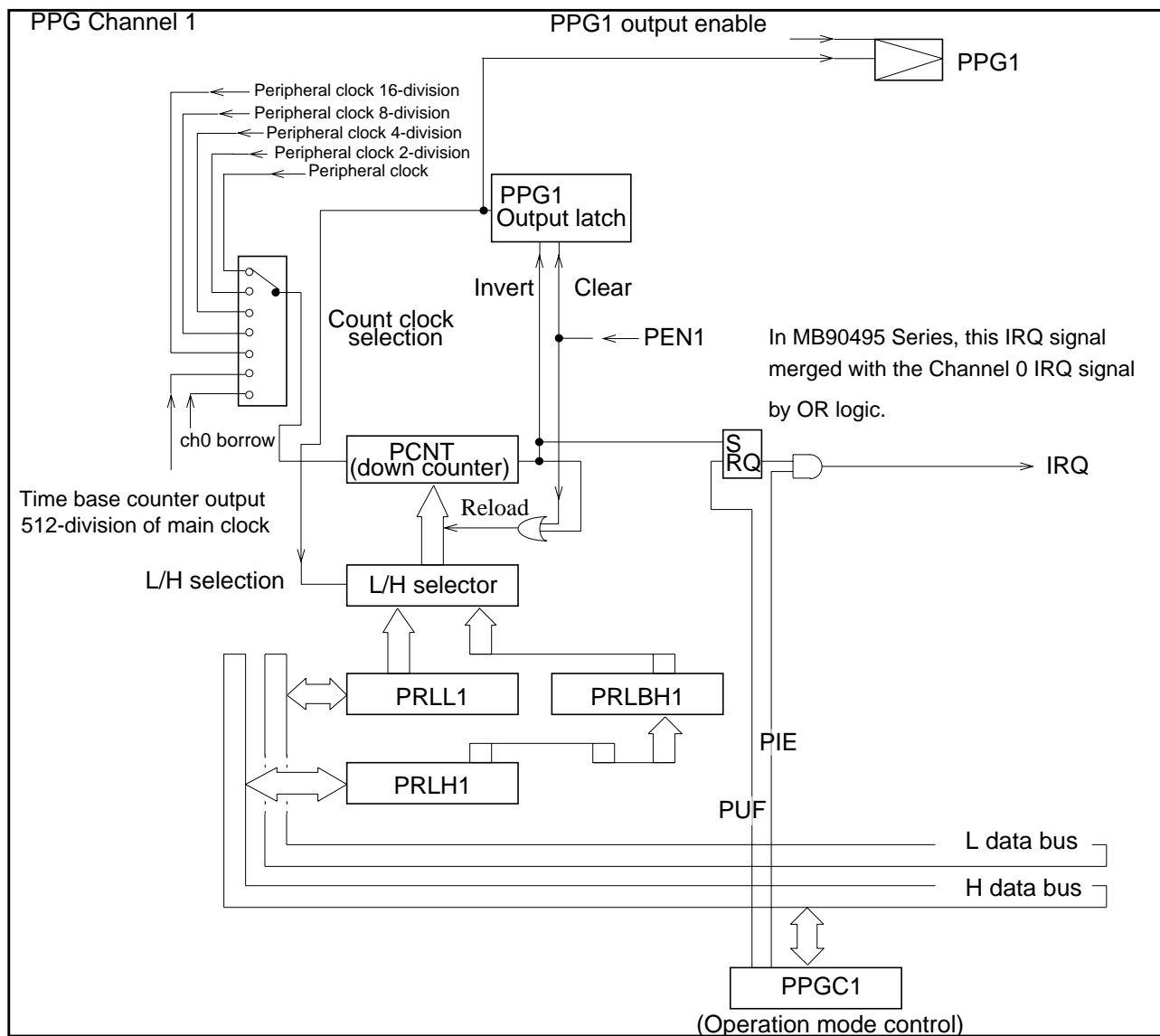


Figure 14.3-2 8-bit PPG ch1 block diagram

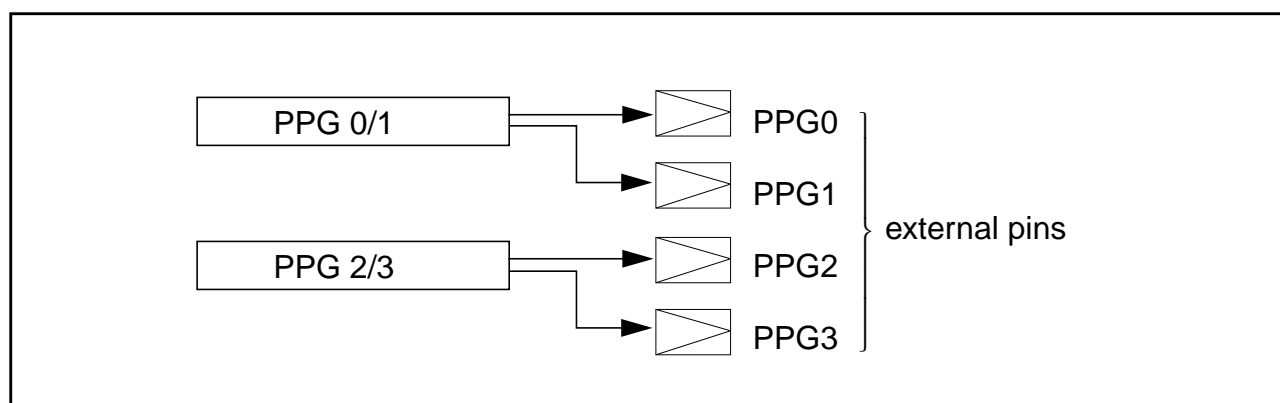


Figure 14.3-3 Relation between PPG modules and external pins

14.4 Register details

(1) PPGC0 (PPG0 operation mode control register)

PPG0 operation mode control register								
Address: ch0, 000040H	7	6	5	4	3	2	1	0 ⇐ Bit No.
	PEN0	-	PE00	PIE0	PUF0	-	-	Reserved
Read/write	⇒ (R/W)		(R/W)	(R/W)	(R/W)			(W)
Initial value	⇒ (0)		(0)	(0)	(X)			(1)

PPGC0 is a five-bit control register that selects the operation mode of the block, controls pin outputs, selects count clock, and controls triggers.

[bit 7] PEN0 (PPG enable): Operation enable bit

This bit enables the counter operation of the PPG.

PEN0	Operation
0	Stop ('L' level output maintained)
1	PPG operation enabled

Setting this bit to 1 enables the counter operation.

This bit is initialized to '0' upon a reset. This bit is readable and writable.

[bit 5] PE00 (PPG output enable 00): PPG0 pin output enable bit

This bit controls the PPG0 pulse output external pin as described below.

0	General-purpose port pin (pulse output disabled)
1	PPG0 = pulse output pin (pulse output enabled)

This bit is initialized to '0' upon a reset. This bit is readable and writable.

For MB90495 Series, this bit should always be set to "0".

[bit 4] PIE0 (PPG interrupt enable): PPG interrupt enable bit

This bit controls PPG interrupt as described below.

0	Interrupt disabled
1	Interrupt enabled

While this bit is "1", an interrupt request is issued as soon as PUF0 is set to "1". No interrupt request is issued while this bit is set to "0".

This bit is initialized to '0' upon a reset. This bit is readable and writable.

[bit 3] PUF0 (PPG underflow flag): PPG counter underflow bit

This bit indicates the PPG counter underflow as described below.

0	PPG counter underflow is not detected.
1	PPG counter underflow is detected.

In 8-bit PPG 2-channel mode or 8-bit prescaler + 8-bit PPG mode, this bit is set to “1” when an underflow occurs as a result of the ch0 counter value becoming from 00H to FFH. In 16-bit PPG mode, this bit is set to “1” when an underflow occurs as a result of the Channel 0 and 1 counter value becoming from 0000H to FFFFH. To set this bit to '0,' write '0.' Writing '1' to this bit has not effect. Upon a read operation during a read-modify-write instruction, '1' is read.

This bit is undefined upon a reset. This bit is readable and writable.

[bit 0] This is a reserved bit. When setting PPGC0, always set this bit to 1. This bit always reads as '1'.

(2) PPGC1 (PPG1 operation mode control register)

PPG1 operation mode control register Address: ch1 000041H	15	14	13	12	11	10	9	8	⇐ Bit No.
	PEN1	-	PE10	PIE1	PUF1	MD1	MD0	Reserved	PPGC1
	Read/write ⇒ Initial value ⇒	(R/W) (0)	(R/W) (0)	(R/W) (0)	(R/W) (X)	(R/W) (0)	(R/W) (0)	(W) (1)	

PPGC0 is a seven-bit control register that selects the operation mode of the block, controls pin outputs, selects count clock, and controls triggers.

[bit 15] PEN1 (PPG enable): Operation enable bit

This bit enables the counter operation of the PPG.

PEN1	Operation
0	Stop ('L' level output maintained)
1	PPG operation enabled

Setting this bit to 1 enables the counter operation.

This bit is initialized to '0' upon a reset. This bit is readable and writable.

[bit 13] PE10 (PPG output enable 10): PPG1 pin output enable bit

This bit controls the PPG1 pulse output external pin as described below.

0	General-purpose port pin (pulse output disabled)
1	PPG1 = pulse output pin (pulse output enabled)

This bit is initialized to '0' upon a reset. This bit is readable and writable.

For MB90495 Series, the pulse signal is output to the “PPG0” external pin.

[bit 12] PIE1 (PPG interrupt enable): PPG interrupt enable bit

This bit controls PPG interrupt as described below.

0	Interrupt disabled
1	Interrupt enabled

While this bit is “1”, an interrupt request is issued as soon as PUF1 is set to “1”. No interrupt request is issued while this bit is set to “0”.

This bit is initialized to '0' upon a reset. This bit is readable and writable.

[bit 11] PUF1 (PPG underflow flag): PPG counter underflow bit

This bit indicates the PPG counter underflow as described below.

0	PPG counter underflow is not detected.
1	PPG counter underflow is detected.

In 8-bit PPG 2-channel mode or 8-bit prescaler + 8-bit PPG mode, this bit is set to “1” when an underflow occurs as a result of the Channel 1 counter value becoming from 00H to FFH. In 16-bit PPG mode, this bit is set to “1” when an underflow occurs as a result of the Channel 0 and 1 counter value becoming from 0000H to FFFFH. To set this bit to '0,' write '0.' Writing '1' to this bit has not effect. Upon a read operation during a read-modify-write instruction, '1' is read.

This bit is undefined upon a reset. This bit is readable and writable.

[bit 10, 9] MD2, 1 (PPG count mode): Operation mode selection bit

This bit selects the PPG timer operation mode as described below.

MD1	MD0	Operation mode
0	0	8-bit PPG 2ch independent mode
0	1	8-bit prescaler + 8-bit PPG 1ch mode
1	0	Reserved
1	1	16-bit PPG 1ch mode

This bit is initialized to '00' upon a reset. This bit is readable and writable.

Note: Do not set '10' in this bit.

Note: To write '01' to this bit, ensure that '01' is not written to the PEN0 bit of PPGC0 or PEN1 bit of PPGC1.

Write '11' or '00' in both the PEN0 and PEN1 bits simultaneously.

Note: To write '11' to this bit, update PPGC0 and PPGC1 by word transfer and write '11' or '00' to both the PEN0 and PEN1 bits simultaneously.

[bit 8] This is a reserved bit. When setting PPGC1, always write 1 to this bit. This bit always reads as '1'.

(3) PPG01 (PPG0, 1 clock select register)

PPG0, 1 clock select register									
Address: ch0, 1	7	6	5	4	3	2	1	0	⇐ Bit No.
000042H	PCS2	PCS1	PCS0	PCM2	PCM1	PCM0	—	—	PPG01
Read/write ⇒	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)			
Initial value ⇒	(0)	(0)	(0)	(0)	(0)	(0)			

This is an 8-bit control register that controls the operation clock of the PPGs.

[bits 7 to 5] PCS2 to 0 (PPG count select): Count clock selection bit

These bits select the operation clock for the down counter of Channel 1 as described below.

PCS2	PCS1	PCS0	Operation mode
0	0	0	Peripheral clock (62.5-ns machine clock, 16 MHz)
0	0	1	Peripheral clock/2 (125-ns machine clock, 16 MHz)
0	1	0	Peripheral clock/4 (250-ns machine clock, 16 MHz)
0	1	1	Peripheral clock/8 (500-ns machine clock, 16 MHz)
1	0	0	Peripheral clock/16 (1-μs machine clock, 16 MHz)
1	1	1	Clock input from the time base timer (128-us, 4-Mhz source)

This bit is initialized to '000' upon a reset. This bit is readable and writable.

Note: : In 8-bit prescaler + 8-bit PPG mode or in 16-bit PPG mode, ch1 PPG operates in response to a counter clock from ch0. Therefore, the setting in these bits has no effect.

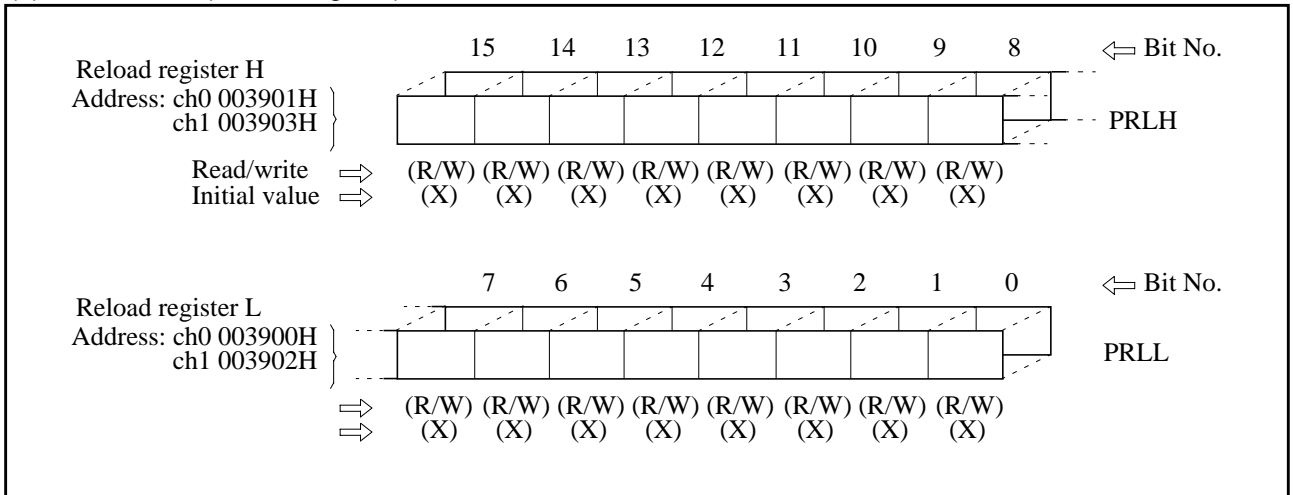
[bits 4 to 2] PCM2 to 0 (PPG count mode): Count clock selection bit

These bits select the operation clock for the down counter of Channel 0 as described below.

PCM2	PCM1	PCM0	Operation mode
0	0	0	Peripheral clock (62.5-ns machine clock, 16 MHz)
0	0	1	Peripheral clock/2 (125-ns machine clock, 16 MHz)
0	1	0	Peripheral clock/4 (250-ns machine clock, 16 MHz)
0	1	1	Peripheral clock/8 (500-ns machine clock, 16 MHz)
1	0	0	Peripheral clock/16 (1-μs machine clock, 16 MHz)
1	1	1	Clock input from the time base timer (128-us, 4-Mhz source)

This bit is initialized to '000' upon a reset. This bit is readable and writable.

(4) PRLL/PRLH (Reload register)



These are 8-bit registers storing the reload values for the PCNT down counters.

Register name	Function
PRLL	Holds the L side reload value.
PRLH	Holds the H side reload value.

These registers are readable and writable.

Note: In 8-bit prescaler + 8-bit PPG mode, different values in PRLL and PRLH of Channel 0 may cause the PPG waveform of ch1 to vary in each cycle. Write the same value to PRLL and PRLH of ch0.

14.5 Operations

One 8/16-bit PPG consists of two channels of 8-bit PPG units. These two channels 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 pulse width (PRLH) and the other is for the H pulse width (PRLH). The values stored in these registers are reloaded into the 8-bit down counter (PCNT), from the PRLH and PRLH in turn. The pin output value is inverted upon a reload caused by counter borrow. This operation results in the pulses of the specified L pulse width and H pulse width.

The table below lists the relationship between the reload operation and pulse outputs.

Table 14.5 -1 Reload operation and pulse output

Reload operation	Pin output change	
PRLH => PCNT	PPG0/1 [0 => 1]	Rise
PRLH => PCNT	PPG0/1 [1 => 0]	Fall

When 1 is set in bit 4 (PIE0) of PPGC0 or in bit 12 (PIE1) of PPGC1, an interrupt request is output upon a borrow from 00 to FF (from 0000 to FFFF in 16-bit PPG mode) of each counter.

(1) 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.

In independent two-channel mode, the two channels of 8-bit PPG units operate independently. The PPG0 pin is connected to the ch0 PPG output, while the PPG1 pin is connected to the ch1 PPG output.

In 8-bit prescaler + 8-bit PPG mode, ch0 is used as an 8-bit prescaler while the count in ch1 is based on borrow outputs from ch0. Thus, 8-bit PPG waveforms can be output with arbitrary length of cycle time. The PPG0 pin is connected to the ch0 prescaler output, while the PPG1 pin is connected to the ch1 PPG output.

In 16-bit PPG 1ch 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.

(2) PPG output operation

In this block, the ch0 PPG is activated to start counting when '1' is written to bit 7 (PEN0) of the PPGC0 (PWM operation mode control) register. Similarly, the ch1 PPG is activated to start counting when '1' is written to bit 15 (PEN1) of the PPGC1 register. Once the operation has started, counting is terminated by writing '0' to bit 7 (PEN0) of PPGC0 or in bit 15 (PEN1) of PPGC1. Once the counting is terminated, the output is maintained at the L level.

In 8-bit prescaler + 8-bit PPG mode, do not set ch1 to be in operation while ch0 operation is stopped.

In 16-bit PPG mode, ensure that bit 7 (PEN0) of PPGC0 register and bit 15 (PEN1) of PPGC1 register are started or stopped simultaneously. The figure 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 to the L-level period). PPG continues operation until stop is specified explicitly.

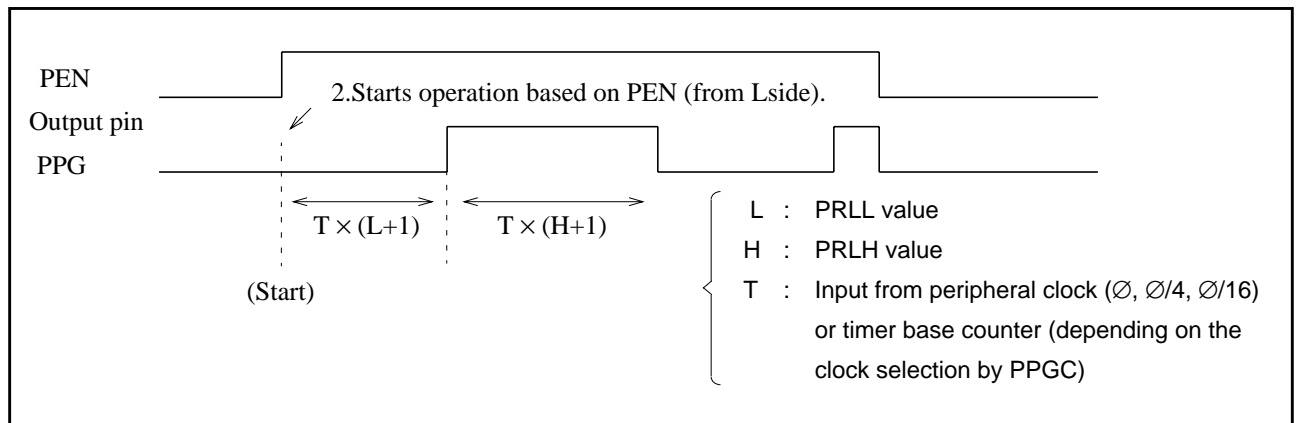


Figure 14.5-1 output operation, output waveform

(3) Reload value and pulse width

The width of the output pulse is determined by adding 1 to the reload register value and multiplying it by the count clock cycle. Note that when the reload register value is 00H during 8-bit PPG operation or 0000H 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 FFH during 8-PPG operation, the pulse width is equivalent to 256 count clock cycles. When the reload register value is FFFFH during 16-bit PPG operation, the pulse width is equivalent to 65536 count clock cycles. An example of pulse width calculation is given below.

$$\begin{aligned} P_l &= T \times (L+1) \\ P_h &= T \times (H+1) \end{aligned} \quad \left\{ \begin{array}{l} L : \text{PRL value} \\ H : \text{value} \\ T : \text{Input clock cycle} \\ P_h : \text{High pulse width} \\ P_l : \text{Low pulse width} \end{array} \right.$$

(4) Count clock selection

The count clock used for the operation is supplied from the peripheral clock or the time base timer. The count clock can be selected from six choices.

Select ch0 clock at bit 4 to 2 (PCM2 to 0) of the PPG01 register, and ch1 clock at bit 7 to 5 (PCS2 to 0) of the PPG01 register.

The clock is selected from a peripheral clock 1/16 to 1 times higher than a machine clock or an input clock from the time base timer.

In 8-bit prescaler + 8-bit PPG mode or 16-bit PPG mode, however, the setting in the PCS2 to 0 has no effect.

When the time base timer 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-bit prescaler + 8-bit PPG mode, if ch1 is activated while ch0 is in operation and ch1 is stopped, the first count cycle may be shifted.

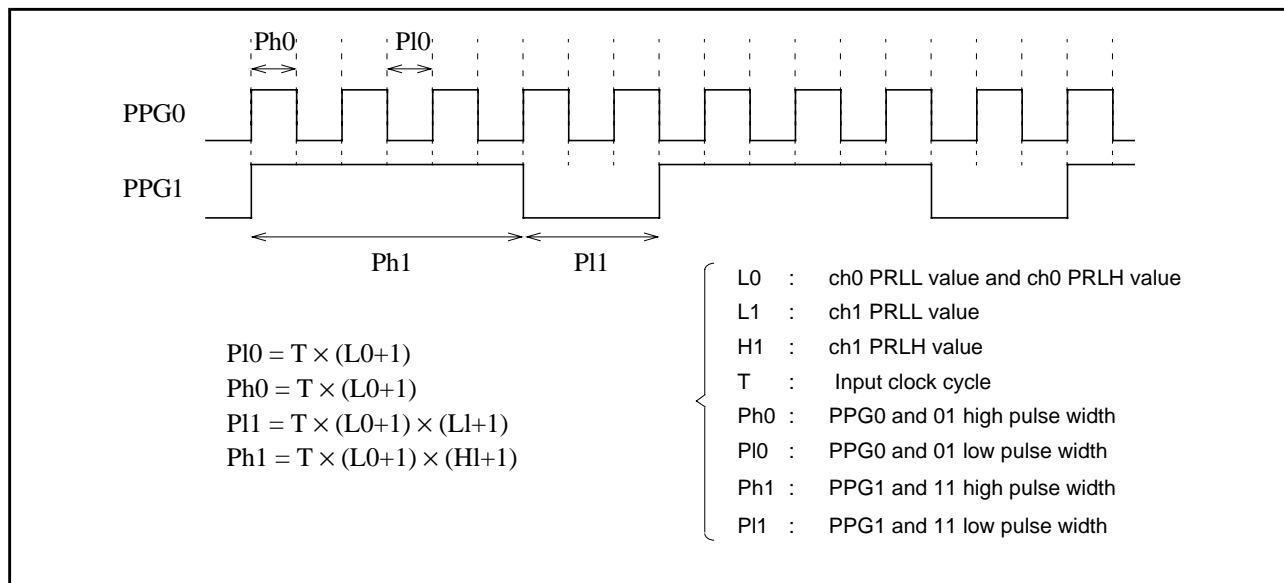
(5) Pulse pin output control

The pulses generated by this module can be output from external pins PPG0 and PPG1.

To output the pulses from an external pin, write '1' to the bit corresponding to each pin. When '0' is written to these bits (default), the pulses are not output from the corresponding external pins; the pins 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 both external pin.

In 8-bit prescaler + 8-bit PPG mode, the 8-bit prescaler toggle output waveform is output from PPG00, while the 8-bit PPG waveform is output from PPG1. The figure below is a diagram of output waveforms in this mode.



Note: Set the same value in ch0 PRL and ch0 PRLH.

Figure 14.5-2 8+8 PPG output operation waveform

(6) Interrupts

For this module, an interrupt becomes active when the reload value is counted out and a borrow occurs.

In 8-bit PPG 2ch mode or 8-bit prescaler + 8-bit PPG mode, an interrupt is requested by a borrow in each counter. In 16-bit PPG mode, PUG0 and PUF1 are simultaneously set by a borrow in the 16-bit counter. Therefore, enable only PIE0 or PIE1 to unify the interrupt causes. In addition, simultaneously clear the interrupt flags for PUF0 and PUF1.

(7) Default values of hardware components

The hardware components of this block are initialized to the following values when reset:

('-' means: bit not used)

<Registers>	• PPGC0	⇒ 0-00X--1 _B
	• PPGC1	⇒ 0-00X001 _B
	• PPG01	⇒ 000000-- _B
<Pulse outputs>	PPG0	⇒ 'L'
	PPG1	⇒ 'L'
	PE00	⇒ PPG0 output disabled
	PE10	⇒ PPG1 output disabled
<Interrupt requests>	IRQ0	⇒ 'L'
	IRQ1	⇒ 'L'

Hardware components other than the above are not initialized.

Note: Reload register write timing

In a mode other than 16-bit PPG mode, it is recommended to use a word transfer instruction to write data in reload registers PRLH and PRLH. If two byte transfer instructions are used to write a data item to these registers, a pulse of unexpected cycle time may be output depending on the timing.

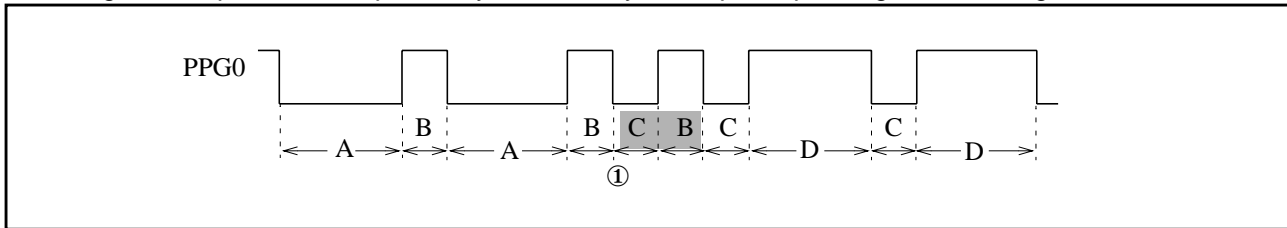


Figure 14.5-3 Write timing chart

Assume that PRLH 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 PRLH=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 a mode other than 16-bit PPG mode, ch0 and ch1 PRL are written independently.

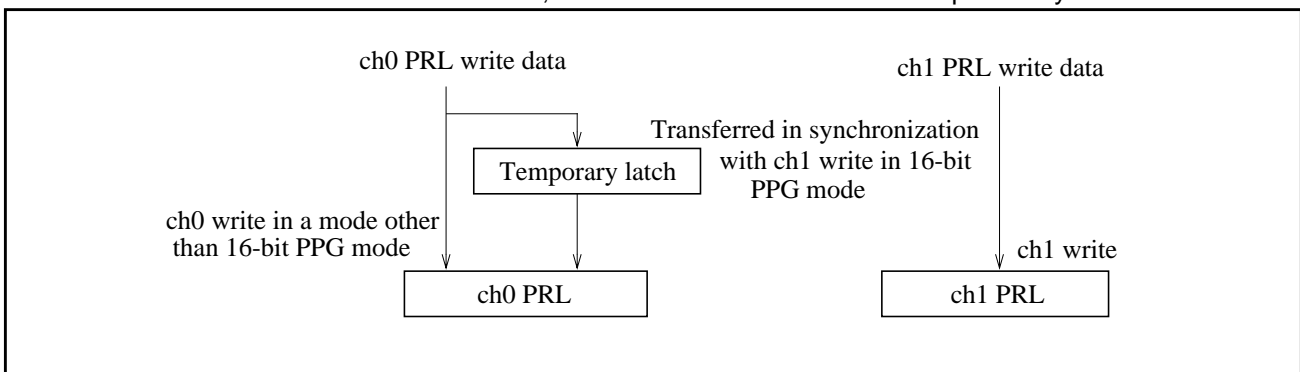


Figure 14.5-4 PRL write operation block diagram

CHAPTER 15 8/10-BIT A/D CONVERTER

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

- 15.1 Overview of the 8/10-Bit A/D Converter
- 15.2 Configuration of the 8/10-Bit A/D Converter
- 15.3 8/10-Bit A/D Converter Pins
- 15.4 8/10-Bit A/D Converter Registers
- 15.5 8/10-Bit A/D Converter Interrupts
- 15.6 Operation of the 8/10-Bit A/D Converter
- 15.7 Usage Notes on the 8/10-Bit A/D Converter
- 15.8 Sample Program 1 for the 8/10-Bit A/D Converter (Single Conversion Mode Using EI2OS)
- 15.9 Sample Program 2 for the 8/10-Bit A/D Converter (Continuous Conversion Mode Using EI2OS)
- 15.10 Sample Program 3 for the 8/10-Bit A/D Converter (Stop Conversion Mode Using EI2OS)

15.1 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 input to an analog input pin (input voltage) to a digital value. The converter has the following features:

- The minimum conversion time is 6.12 μs (for a machine clock of 16 MHz; includes the sampling time).
- The minimum sampling time is 2.0 μs (for a machine clock of 16 MHz).
- The converter uses the RC-type successive approximation conversion method with a sample hold circuit.
- A resolution of 10 bits or 8 bits can be selected.
- 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 15.1 -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 15.1-1 8/10-bit A/D converter interrupts and EI²OS

Interrupt No.	Interrupt control register		Vector table address			EI ² OS
	Register name	Address	Lower	Upper	Bank	
#18 (12H)	ICR03	0000B3H	FFFFB4H	FFFFB5H	FFFFB6H	0

o: Available

15.2 Configuration of the 8/10-Bit A/D Converter

The 8/10-bit A/D converter has nine blocks:

- **A/D control status register (ADCS0, 1)**
- **A/D data register (ADCR0, 1)**
- **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

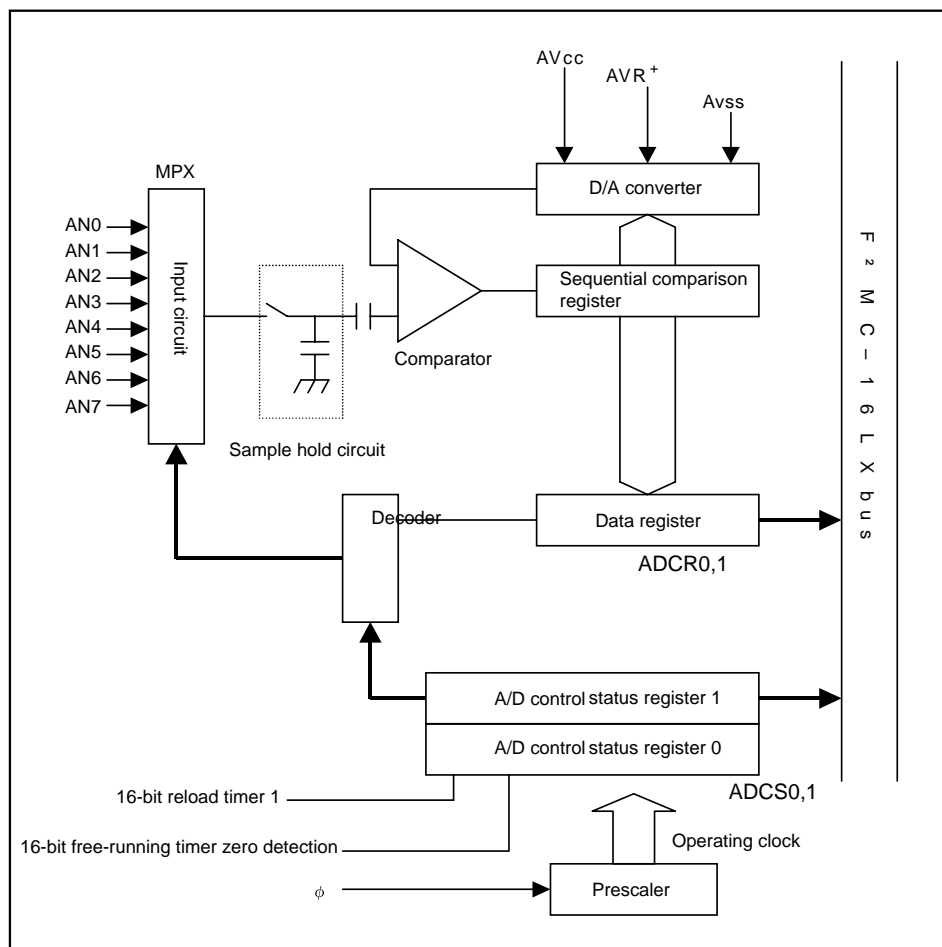


Figure 15.2-1 Block diagram of the 8/10-bit A/D converter

- **A/D control status register (ADCS0, 1)**

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 (ADCR0, 1)**

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 (ADCS0).

- **Analog channel selector**

This circuit selects the pin to be used from eight analog input pins.

- **Sample hold circuit**

This circuit maintains the input voltage of the channel selected by the analog channel selector. It samples and maintains 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 maintained by the sample hold circuit with the output voltage of the D/A converter to determine which is greater.

- **Control circuit**

This circuit determines the A/D conversion value based on the decision signal generated by the comparator. When the A/D conversion has been completed, the circuit sets the conversion result in the A/D data register (ADCR0, 1) and generates an interrupt request.

15.3 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 15.3 -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
Ch 0	P50/AN0	Port 5 I/O or anaog input	CMOS out-put/CMOS hysteresis input or analog input	Not selectable	Not selectable	Set port 5 as an input port (DDR5 = 00H). Set port 5 as an analog input port (ADER = FFH)
Ch 1	P51/AN1					
Ch 2	P52/AN2					
Ch 3	P53/AN3					
Ch 4	P54/AN4					
Ch 5	P55/AN5					
Ch 6	P56/AN6					
Ch 7	P57/AN7					

■ Block diagrams of the 8/10-bit A/D converter pins

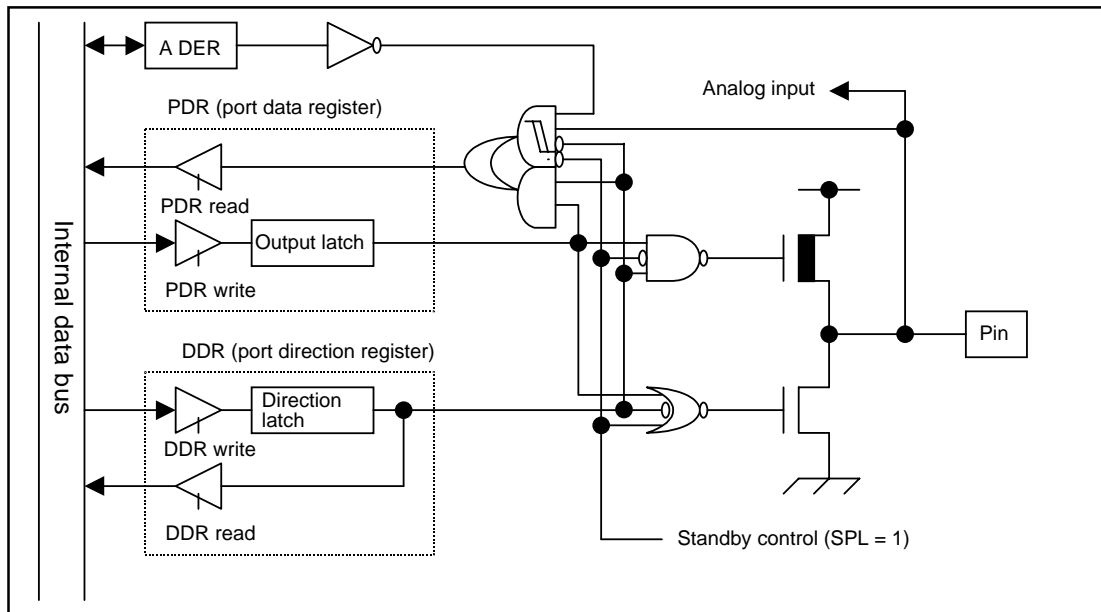


Figure 15.3-1 Block diagram of the P50/AN0 to P57/AN7 pins

<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.

15.4 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

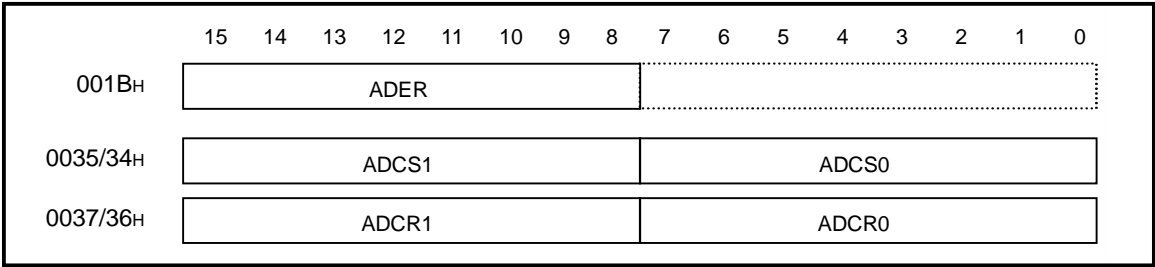


Figure 15.4-1 8/10-bit A/D converter registers

15.4.1 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)

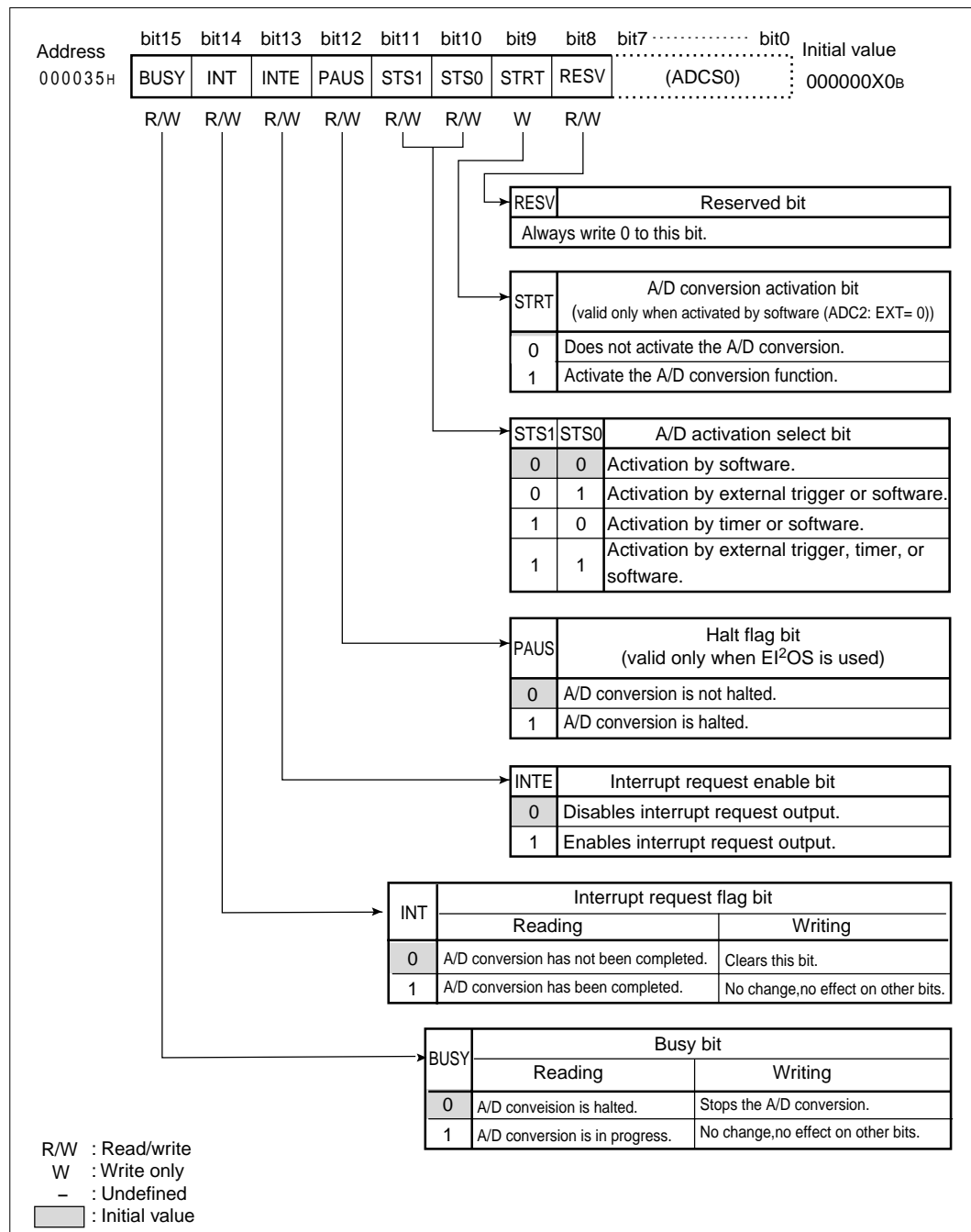


Figure 15.4.1-1 A/D control status register 1 (ADCS1)

Table 15.4.1-1 Function description of each bit of control status register 1 (ADCS1)

Bit name		Function
bit15	BUSY: Busy bit	<ul style="list-style-type: none"> 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. <p><Caution> Never select forced stop (BUSY = 0) and software activation (STRT = 1) simultaneously.</p>
bit14	INT: Interrupt request flag bit	<ul style="list-style-type: none"> When A/D conversion data is set 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. <p><Caution> When clearing this bit by writing 0 it, do so only while the A/D converter is not operating.</p>
bit13	INTE: Interrupt request enable bit	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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. <p><Caution> This bit is valid only when EI²OS is used.</p>
bit11 bit10	STS1, STS0: A/D activation select bit	<ul style="list-style-type: none"> 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. <p><Caution></p> <ul style="list-style-type: none"> Change the setting during A/D conversion only while there is no corresponding activation cause, since the change becomes effective immediately.

Bit name		Function
bit9	STRT: A/D conversion activation bit	<ul style="list-style-type: none"> • 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. <p><Caution> Never select forced stop (BUSY = 0) and software activation (STRT = 1) simultaneously.</p>
bit8	RESV: Reserved bit	<p><Caution> Always write 0 to this bit.</p>

15.4.2 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)

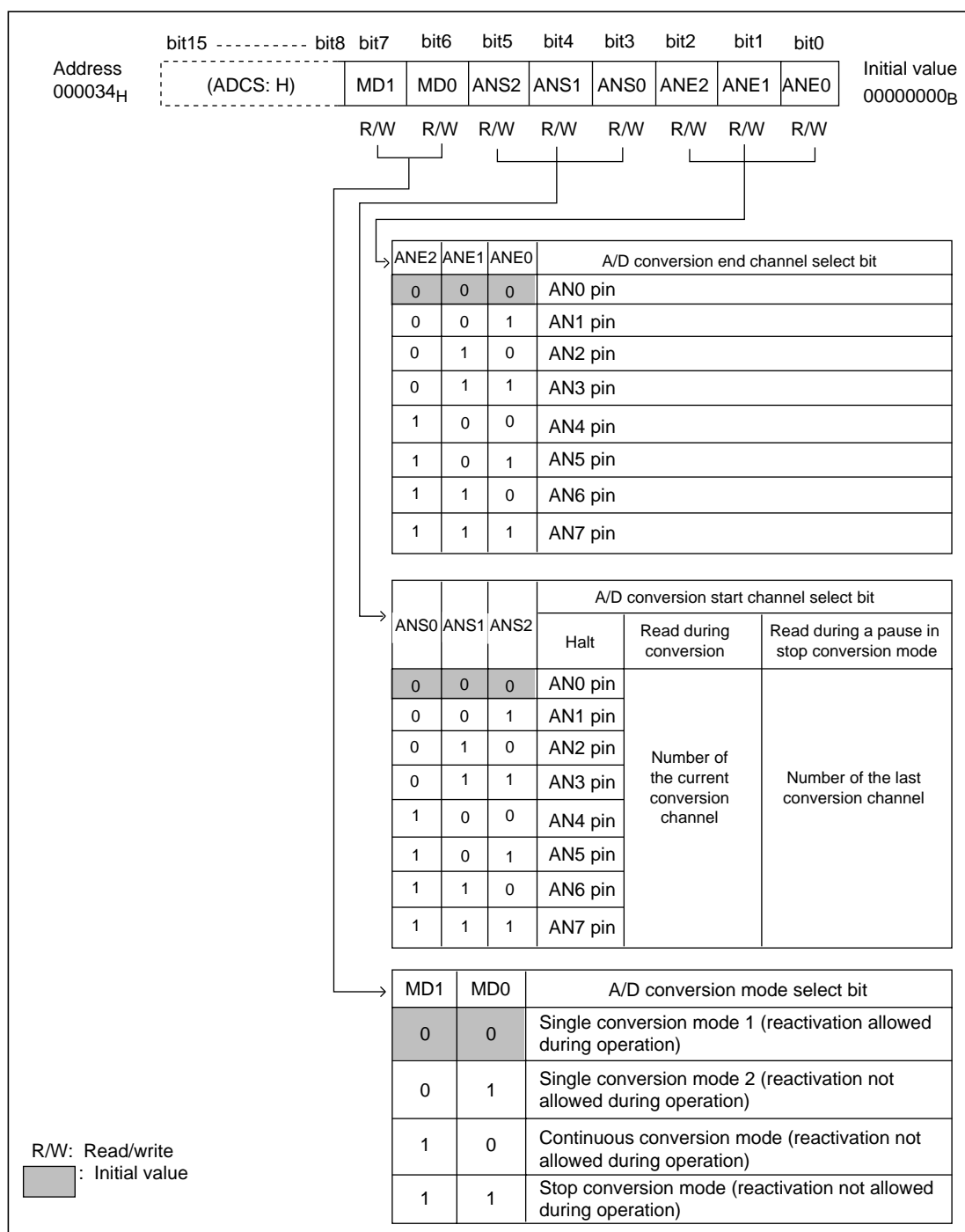


Figure 15.4.2-1 A/D control status register 0 (ADCS0)

Table 15.4.2-1 Function description of each bit of control status register 0 (ADCS0)

Bit name	Function
bit7 bit6 MD1, MD0: A/D conversion mode select bit	<ul style="list-style-type: none"> 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 set 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 set 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 set 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 set 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 is generated. <p><Caution> In the single conversion modes, continuous conversion mode, and stop conversion mode, no reactivation by a timer, external trigger, or software is allowed.</p>
bit5 bit4 bit3 ANS2, ANS1, ANS0: A/D conversion start channel select bit	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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.

15.4.3 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)

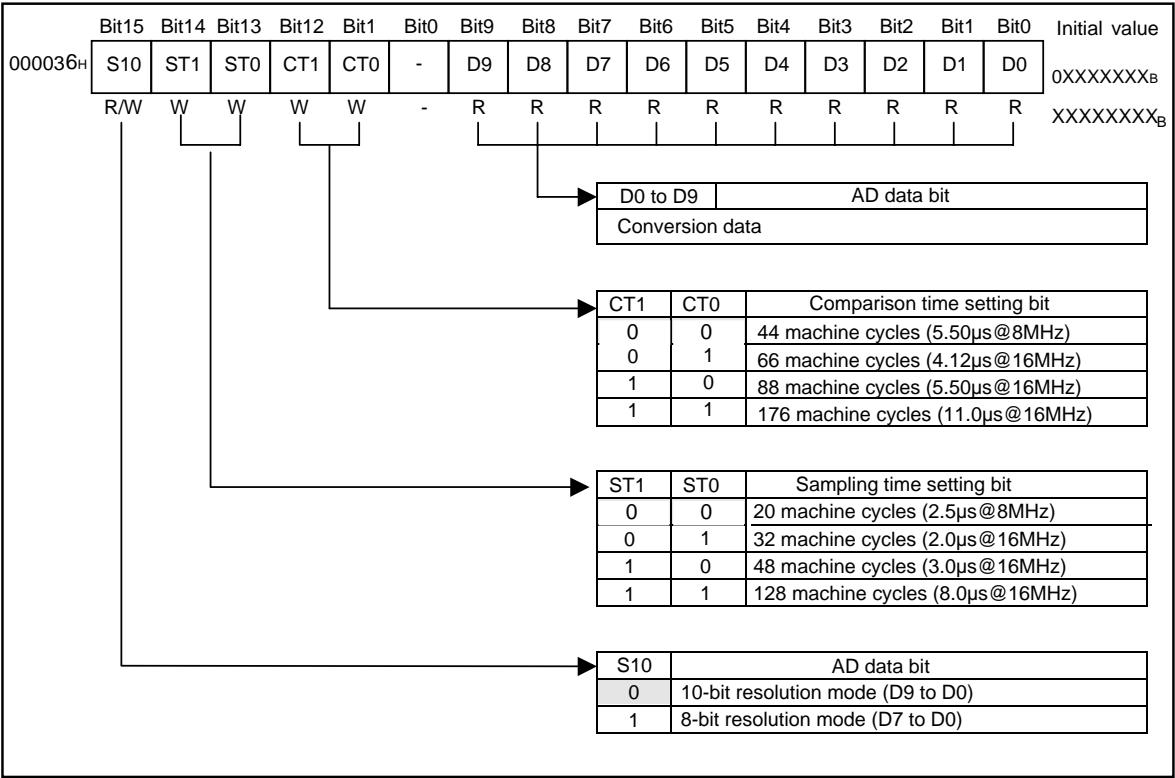


Figure 15.4.3-1 A/D data register (ADCR0, 1)

Table 15.4.3-1 Function description of each bit of A/D data register 0 (ADCR0)

Bit name		Function
bit15	S10: A/D conversion resolution selection bit	<ul style="list-style-type: none"> 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. <p><Caution> The data bit to be used depends on the resolution.</p>
bit14 bit13	ST1, ST0: Sampling time setting bit	<ul style="list-style-type: none"> 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. <p><Caution> Setting these bits to 00 (for 8 MHz) during 16-MHz operation may disable normal fetching of the analog voltage.</p>
bit12 bit11	CT1, CT0: Comparison time setting bit	<ul style="list-style-type: none"> 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. <p><Caution> Setting these bits to 00 (for 8 MHz) during 16-MHz operation may disable normal acquisition of the analog conversion value.</p>
bit10	Free bit	
bit9 ~ bit0	ANE2, ANE1, ANE0: A/D conversion end channel selection bit	<ul style="list-style-type: none"> 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. <p><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.</p>

<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.).

15.5 8/10-Bit A/D Converter Interrupts

The 8/10-bit A/D converter can generate an interrupt request when the data for the A/D conversion is set in the A/D data register. This function supports the extended intelligent I/O service (EI²OS).

■ 8/10-bit A/D converter interrupts

Table 15.5 -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 set 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 15.5 -2 8/10-bit A/D converter interrupts and EI²OS

Interrupt No.	Interrupt control register		Vector table address			EI ² OS
	Register name	Address	Lower	Upper	Bank	
#18 (12H)	ICR03	0000B3H	FFFFB4H	FFFFB5H	FFFFB6H	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.

15.6 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.

The figure below shows the settings required for operation in single conversion mode.

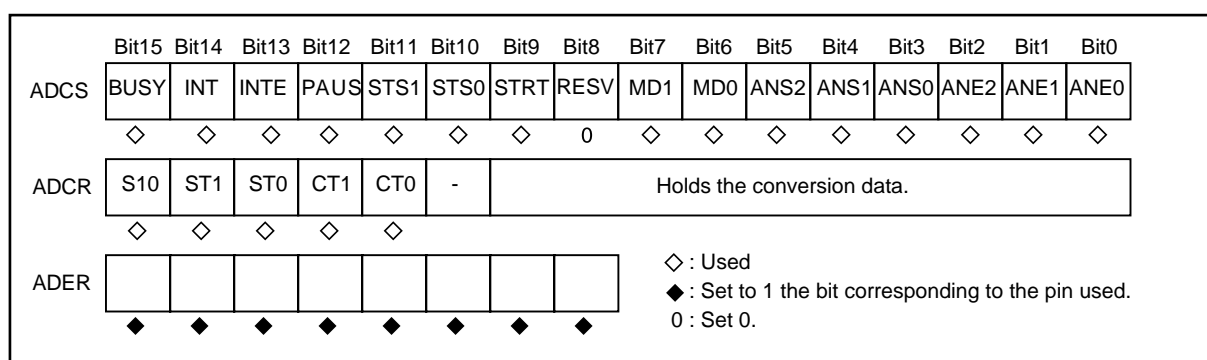


Figure 15.6-1 Settings for single conversion mode

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.

The following figure shows the settings required for operation in continuous conversion mode.

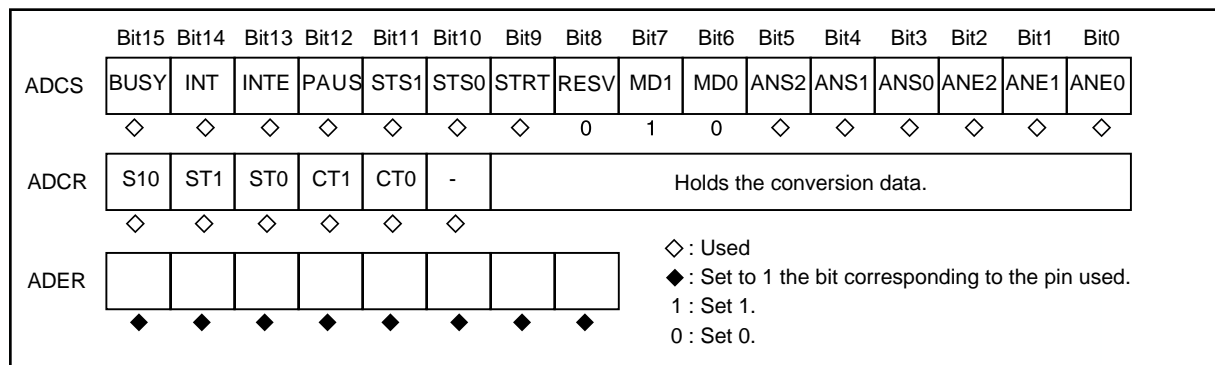


Figure 15.6-2 Settings for continuous conversion mode

Reference

The following are sample conversion sequences in continuous conversion mode:

ANS = 000B, ANE = 011B: AN0 → AN1 → AN2 → AN3 → AN0 → Repeat

ANS = 110B, ANE = 010B: AN6 → AN7 → AN0 → AN1 → AN2 → AN6 → Repeat

ANS = 011B, ANE = 011B: 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.

The settings required for operation in stop conversion mode.

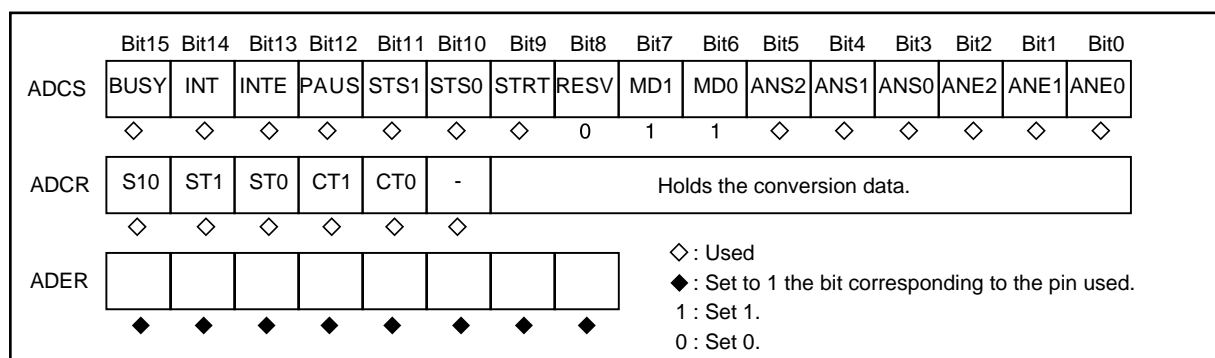


Figure 15.6-3 Settings for stop conversion mode

Reference

The following are sample conversion sequences in stop conversion mode:

ANS = 000_B, ANE = 011_B:

: AN0 → Pause → AN1 → Pause → AN2 → Pause → 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

15.6.1 Conversion using EI²OS

The 8/10-bit A/D converter can use EI²OS transfer the A/D conversion result to memory.

■ Conversion using EI²OS

Operation flow when EI²OS is used.

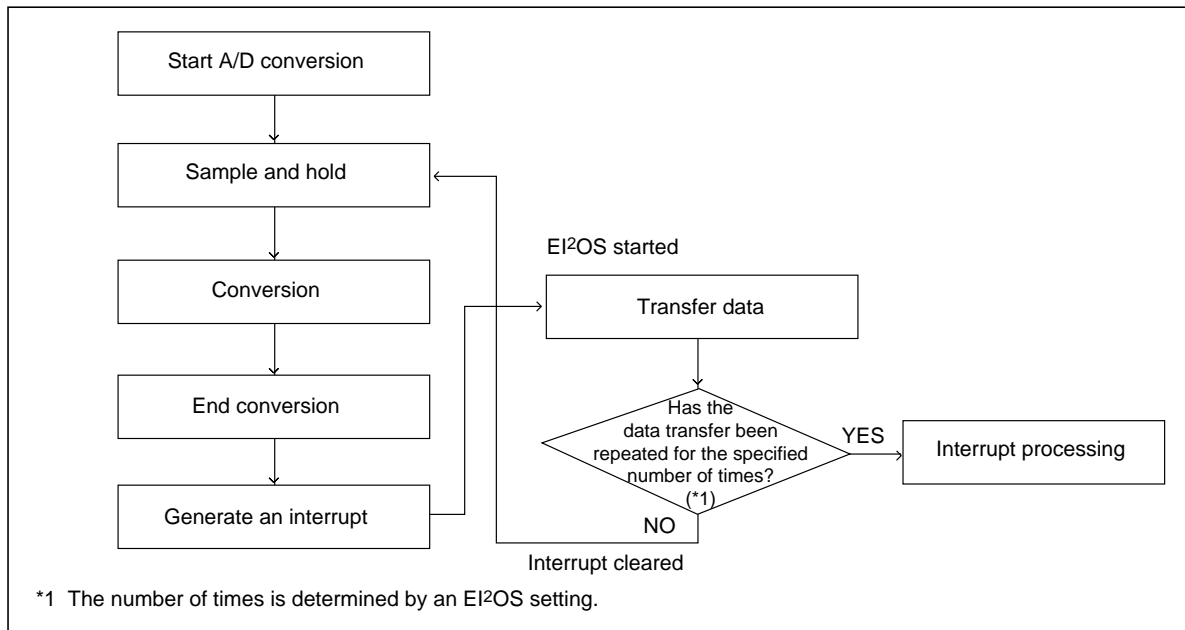


Figure 15.6.1-1 Sample operation flowchart when EI²OS is used

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.

15.6.2 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 register1 (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 and conversion resumes.

The operation flow of the data protection function when EI²OS is used is shown in the following figure.

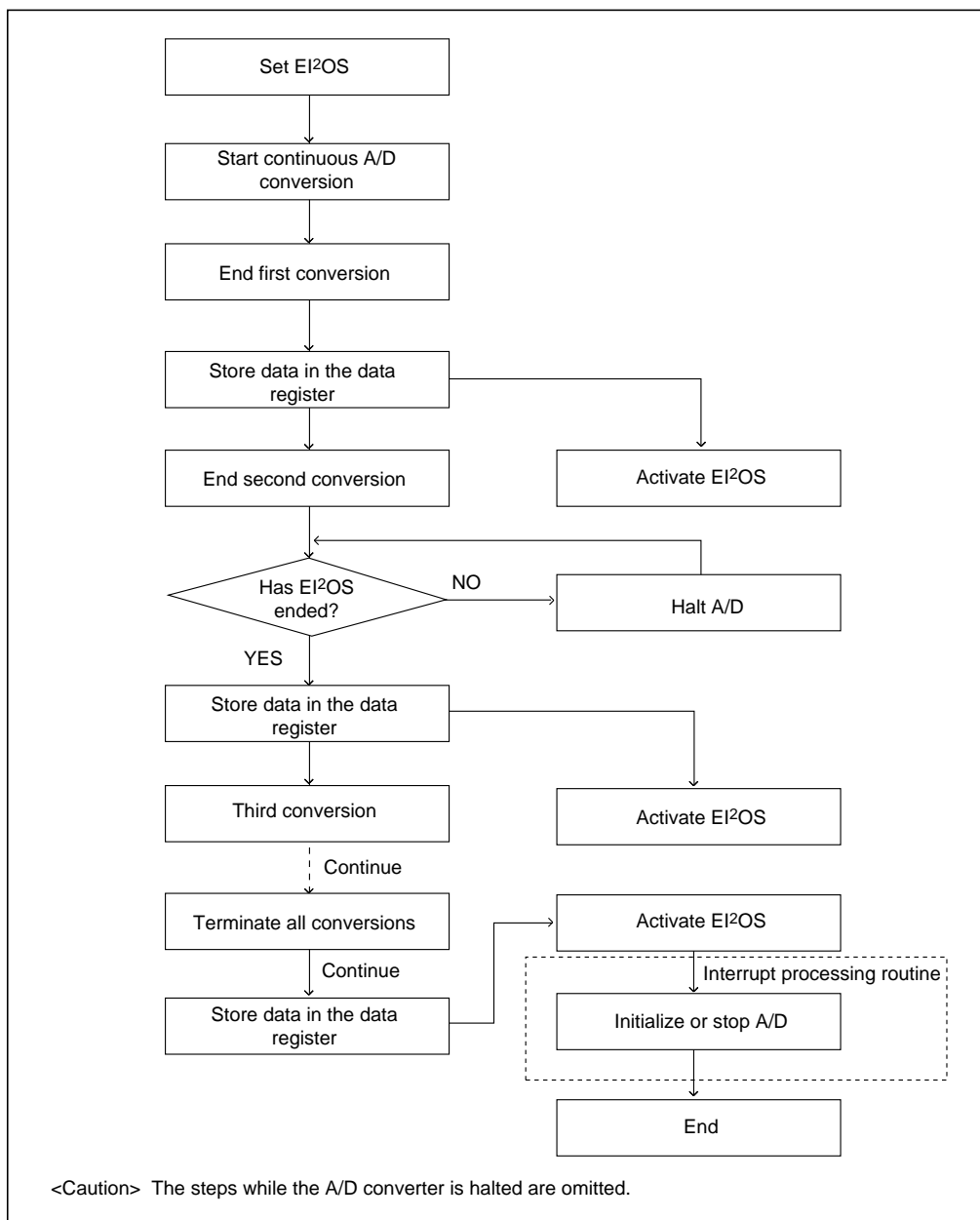


Figure 15.6.2-1 Operation flowchart of the data protection function when EI²OS is used

<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.

15.7 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.

15.8 Sample Program 1 for the 8/10-Bit A/D Converter (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.

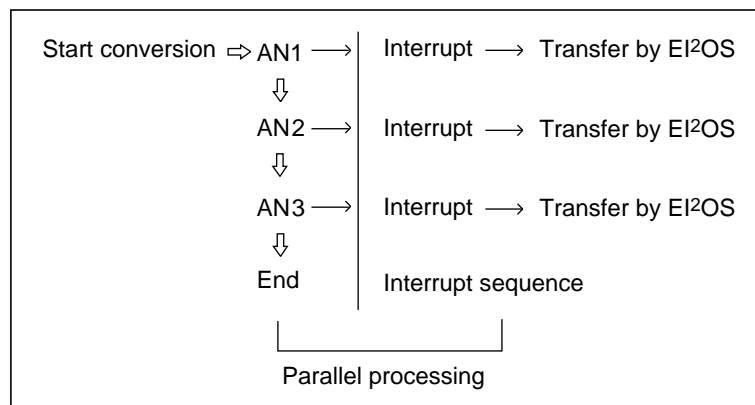


Figure 15.8-1 Flowchart of program using EI²OS (single conversion mode)

● 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	;EI ² OS status register
IOAL	EQU	000104H	;Lower I/O address register
IOAH	EQU	000105H	;Upper I/O address register
DCTL	EQU	000106H	;Lower data counter
DCTH	EQU	000107H	;Upper data counter
DDR5	EQU	000015H	;Port 5 direction register
ADER	EQU	00001BH	;Analog input enable register
ICR03	EQU	0000B3H	;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        ICR03,#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.
CODEENDS

;-----Vector setting-----

VECT CSEG      ABS=0FFH
ORG            0FFB4H                 ;Sets vector for interrupt #18 (12H)
DSL           ED_INT1
ORG            0FFDCH                 ;
DSL           START                   ;Sets reset vector.
DB            00H                     ;
VECTENDS
END  START

```

15.9 Sample Program 2 for the 8/10-Bit A/D Converter (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.

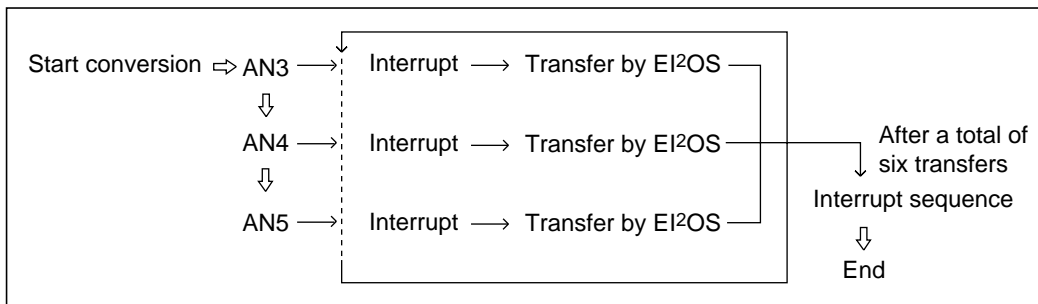


Figure 15.9-1 Flowchart of program using EI²OS (continuous conversion mode)

● 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 registerr
IOAL	EQU	000104H	;Lower I/O address register
IOAH	EQU	000105H	;Upper I/O address register
DCTL	EQU	000106H	;Lower data counter
DCTH	EQU	000107H	;Upper data counter
DDR5	EQU	000015H	;Port 5 direction register
ADER	EQU	00001BH	;Analog input enable register
ICR03	EQU	0000B3H	;Interrupt control register for A/D
ADCS0	EQU	000034H	;A/D control status register
ADCS1	EQU	000035H	;
ADCR0	EQU	000036H	;A/D data register
ADCR1	EQU	000037H	;
TMCSR1L	EQU	000068H	;Lower control status register 1
TMCSR1H	EQU	000069H	;
TMRL1L	EQU	003902H	;16-bit reload register 1
TMRL1H	EQU	003903H	;

;-----Main program-----

CODE	CSEG	
START:		;Assumes that the stack pointer (SP) has already ; been initialized.
AND	CCR,#0BFH	;Disables interrupts.
MOV	ICR03,#08H	;Interrupt level; 0 (highest priority). Enables EI2OS when ; interrupt
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
MOV	ADCS1,#0A8H	;Activates the 16-bit timer, starts A/D conversion, and ; enables interrupts.
MOVW	TMRL1L,#0320H	;Sets the timer value to 800 (320h), 100 s.
MOV	TMCSR1H,#00H	;Sets the clock source to 125 ns and disables

```

; external trigger.

MOV    TMCSR1L,#12H    ;Disables timer output, disables interrupts, and
                       ; enables reload.
MOV    TMCSR1L,#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    0FFB4H    ;Sets vector for interrupt #18 (12H).
DSL    ED_INT1
ORG    0FFDCH    ;Sets reset vector.
DSL    START
DB     00H        ;Sets single-chip mode.
VECT    ENDS

END  START

```

15.10 Sample Program 3 for the 8/10-Bit A/D Converter (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 600H to 617H.
- A resolution of 10 bits is selected.
- The conversion is activated by 16-bit reload timer.

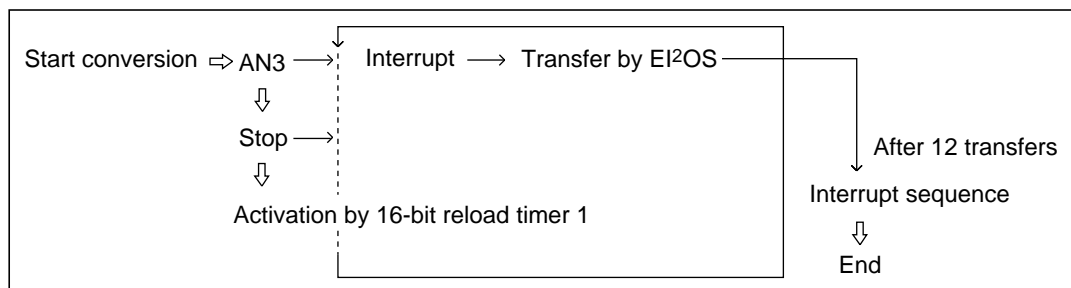


Figure 15.10-1 Flowchart of program using EI²OS (stop conversion mode)

● 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 registerr
IOAL	EQU	000104H	;Lower I/O address register
IOAH	EQU	000105H	;Upper I/O address register
DCTL	EQU	000106H	;Lower data counter
DCTH	EQU	000107H	;Upper data counter
DDR5	EQU	000015H	;Port 5 direction register
ADER	EQU	000017H	;Analog input enable register
ICR03	EQU	0000B3H	;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	;
TMCSR1L	EQU	000068H	;Lower control status register 1
TMCSR1H	EQU	000069H	;
TMRL1L	EQU	003902H	;16-bit reload register 1
TMRL1H	EQU	003903H	;

;-----Main program-----

CODE	CSEG	
START:		;Assumes that the stack pointer (SP) has already ; been initialized.
AND	CCR,#0BFH	;Disables interrupts.
MOV	ICR03,#08H	;Interrupt level: 0 (highest priority) + EI2OS.
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	TMRL1L,#0320H	;Sets the timer value to 800 (320h), 100 s.
MOV	TMCSR1H,#00H	;Sets the clock source to 125 ns and disables ; external trigger.
MOV	TMCSR1L,#12H	;Disables timer output, disables interrupts, and

```

MOV    TMCSR1L,#13H    ; enables reload.
                        ;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    0FFB4H    ;Sets vector for interrupt #18 (12H).
      DSL    ED_INT1
      ORG    0FFDCH    ;Sets reset vector.
      DSL    START
      DB     00H    ;Sets single-chip mode.
VECT ENDS
END  START

```


CHAPTER 16 DTP/EXTERNAL INTERRUPT CIRCUIT

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

- 16.1 Overview of the DTP/External Interrupt Circuit
- 16.2 Configuration of the DTP/External Interrupt Circuit
- 16.3 DTP/External Interrupt Circuit Pins
- 16.4 DTP/External Interrupt Circuit Registers
- 16.5 Operation of the DTP/External Interrupt Circuit
- 16.6 Usage Notes on the DTP/External Interrupt Circuit
- 16.7 Sample Programs for the DTP/External Interrupt Circuit

16.1 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.

Figure 16.1-1 provides an overview of the DTP/external interrupt circuit.

Figure 16.1-1 Overview of the DTP/external interrupt circuit

	External interrupt function	DTP function
Input pins	Eight (P60/INT0 to P63/INT3 and P24/INT4 to P27/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	#15(0FH), #20(14H), #24(18H), #27(1BH)	
Interrupt control	The output of interrupt requests is enabled and disabled using the DTP/interrupt enable register (ENIR).	
Interrupt flag	Interrupt causes are stored in the DTP/interrupt cause register (EIRR).	
Processing selection	EI ² OS is disabled (ICR: ISE = 0).	EI ² OS is enabled (ICR: ISE = 1).
Processing	The circuit branches to an external interrupt processing routine.	The circuit performs automatic data transfer using EI ² OS 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

Figure 16.1-2 Interrupt of the DTP/external interrupt circuit and EI²OS

Channel	Interrupt number	Interrupt control register		Vector table address			EI ² OS
		Register name	Address	Lower	Upper	Bank	
INT0/INT1	#15 (0FH)	ICR02	0000B2H	FFFFCOH	FFFFC1H	FFFFC2H	Δ
INT2/INT3	#20 (14H)	ICR04	0000B4H	FFFFACH	FFFFADH	FFFFAEH	
INT4/INT5	#24 (18H)	ICR06	0000B6H	FFFF9CH	FFFF9DH	FFFF9EH	
INT6/INT7	#27 (1BH)	ICR08	0000B8H	FFFF90H	FFFF91H	FFFF92H	

Δ: Available when not using an interrupt request that shares the ICR07 and ICR08 registers.

16.2 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

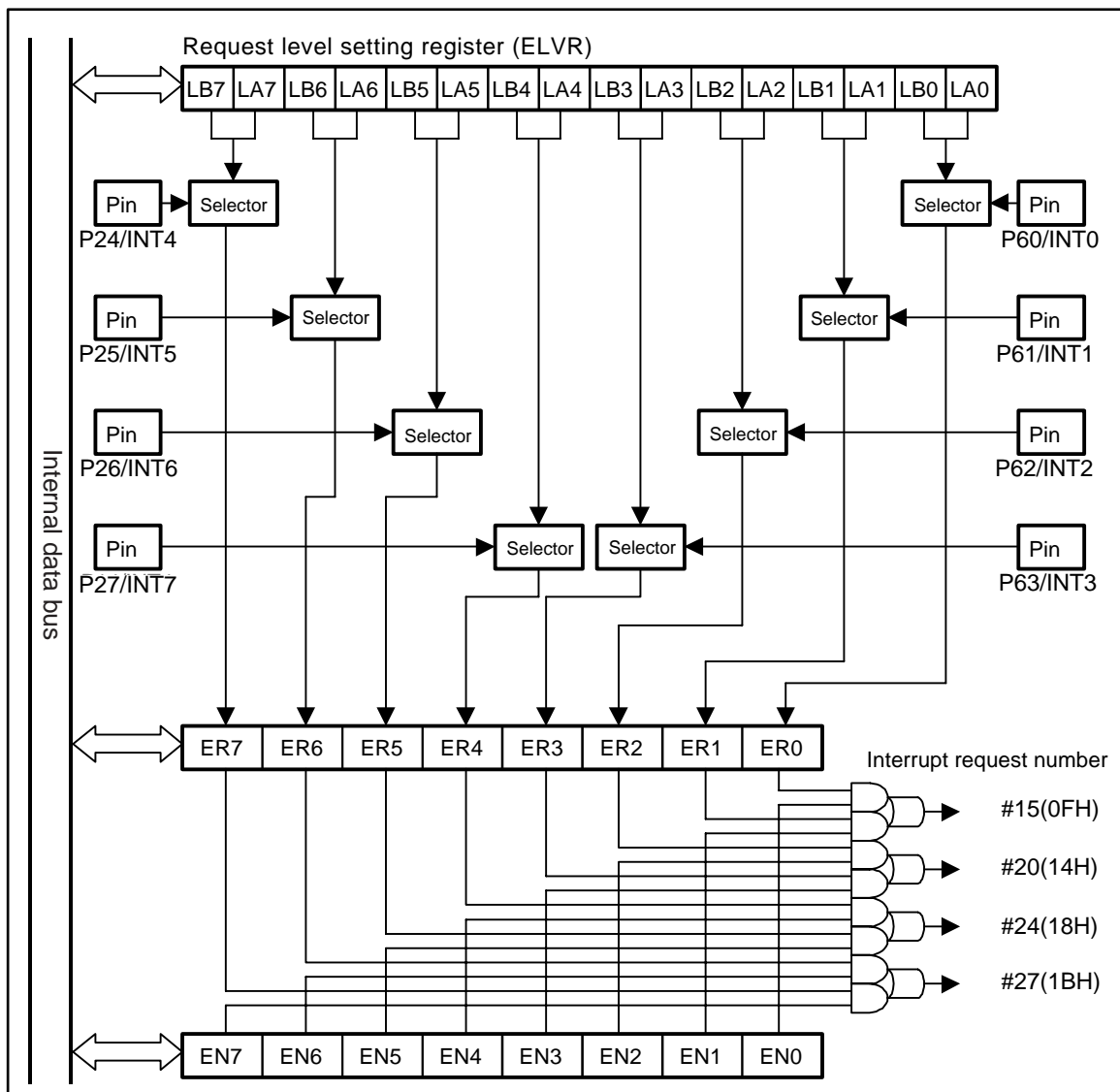


Figure 16.2-1 Block diagram of the DTP/external interrupt circuit

- **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 IR 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.

16.3 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 16.3-1 lists the pin functions, I/O formats, and settings required to use the DTP/external interrupt circuit.

Table 16.3-1 DTP/external interrupt circuit pins

Pin name	Function	I/O format	Pull-up resistor	Standby control	Setting required to use pins
P60/INT0	input-output/ external inter- rupt input	CMOS output/ CMOS hysteresis input	Select- able	provided	Set the pin as an input port (DDR6: bit0 = 0)
P61/INT1					Set the pin as an input port (DDR6: bit1 = 0)
P62/INT2					Set the pin as an input port (DDR6: bit2 = 0)
P63/INT3					Set the pin as an input port (DDR6: bit3 = 0)
P24/INT4	Set the pin as an input port (DDR2: bit4 = 0)				
P25/INT5	Set the pin as an input port (DDR2: bit5 = 0)				
P26/INT6	Set the pin as an input port (DDR2: bit6 = 0)				
P27/INT7	Not pro- vided		Set the pin as an input port (DDR2: bit7 = 0)		

■ Block diagram of the DTP/external interrupt circuit pins

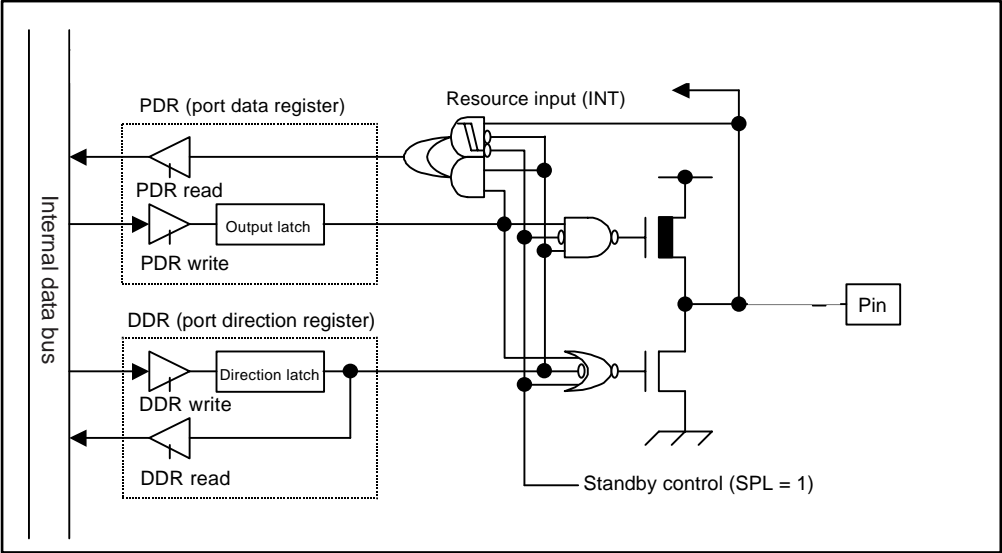


Figure 16.3-1 Block diagram of the DTP/external interrupt circuit pins

16.4 DTP/External Interrupt Circuit Registers

This section describes DTP/external interrupt circuit registers.

Address	bit15	bit8	bit7	bit0
000031H, 30H	DTP/interrupt cause register (EIRR)			
000033H, 32H	DTP/interrupt enable register (ENIR)			
	Request level setting register (ELVR)			

Figure 16.4-1 DTP/external interrupt circuit registers

16.4.1 DTP/interrupt cause register (EIRR)

The DTP/interrupt cause register (EIRR) stores and clears interrupt causes.

■ DTP/interrupt cause register (EIRR)

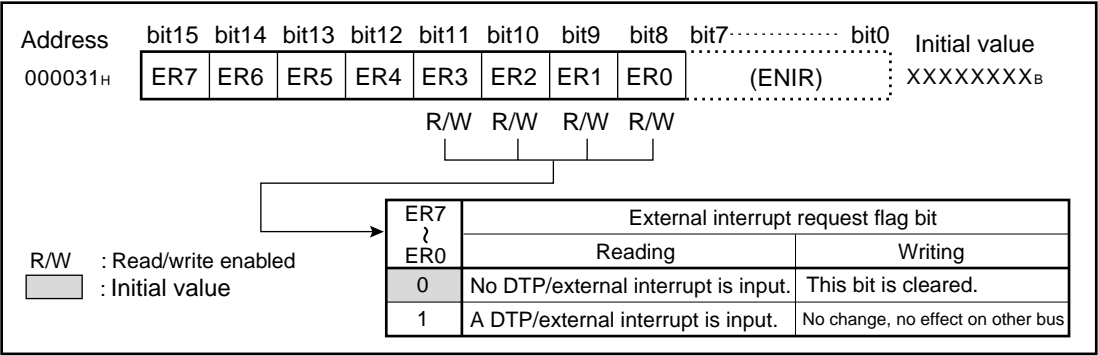


Figure 16.4.1-1 DTP/interrupt cause register (EIRR)

Table 16.4.1-1 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	<ul style="list-style-type: none">Each of these bits is set to 1 if a signal with the edge or level selected by bits LB7, LA7 to LB0, and LA0 of the request level setting register (ELVR) is input to the DTP/ external interrupt pin (stores an interrupt cause).If these bits and corresponding bits EN3 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. <p><Caution> If more than one external interrupt request output is enabled (ENIR: EN3 to EN0 = 1), clear only the bit that caused the CPU to accept an interrupt (bits ER7 to ER0 set to 1). Do not clear the other bits without any reason. 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.</p> <p><Caution> If more than one external interrupt request enable bit is set to 1 (ENIR: EN3 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.</p> <p>[Reference] When the extended intelligent I/O service (EI OS) is activated, the corresponding external interrupt request flag bit is automatically cleared when the transfer of one data ends.</p>

16.4 DTP/External Interrupt Circuit Registers

16.4.2 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)

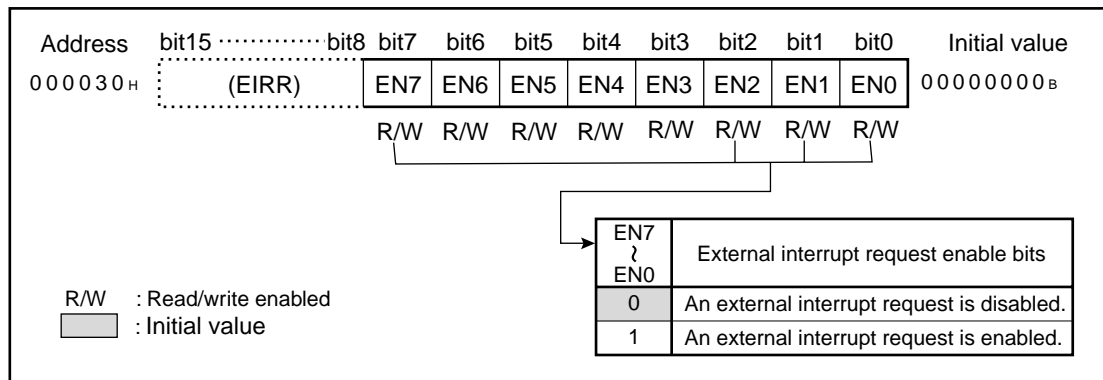


Figure 16.4.2-1 DTP/interrupt enable register (ENIR)

Table 16.4.2-1 Function description of each bit of the DTP/interrupt enable register (ENIR)

Bit name		Function
bit7 bit6 bit5 bit4 bit3 bit2 bit1 bit0	EN3 to EN0: External interrupt request enable bits	<p>Each of these bits enables and disables the output of 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.</p> <p>[Reference]</p> <ul style="list-style-type: none"> 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 ER3 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.

16.4 DTP/External Interrupt Circuit Registers

Table 16.4-3 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
P27/INT7	#27 (1BH)	ER7	EN7
P26/INT6		ER6	EN6
P25/INT5	#24 (18H)	ER5	EN5
P24/INT4		ER4	EN4
P63/INT3	#20 (14H)	ER3	EN3
P62/INT2		ER2	EN2
P61/INT1	#15 (0FH)	ER1	EN1
P60/INT0		ER0	EN0

16.4.3 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)

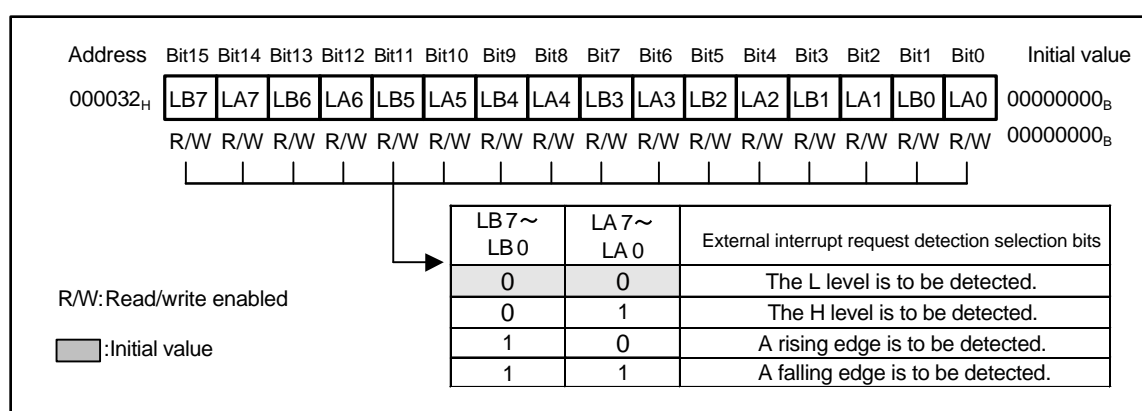


Figure 16.4.3-1 Request level setting register (ELVR)

Table 16.4.3-1 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	<ul style="list-style-type: none"> 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. <p>[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).</p>

16.4 DTP/External Interrupt Circuit Registers

Table 16.4.3-2 Correspondence between request level setting register (ELVR) and each channel

DTP/external interrupt pin	Interrupt number	Bit name
P27/INT7	#27 (1BH)	LB7, LA7
P26/INT6		LB6, LA6
P25/INT5	#24 (18H)	LB5, LA5
P24/INT4		LB4, LA4
P63/INT3	#20 (14H)	LB3, LA3
P62/INT2		LB2, LA2
P61/INT1	#15 (0FH)	LB1, LA1
P60/INT0		LB0, LA0

● 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 (EI²OS) is enabled and the circuit executes its DTP function. If it is 0, EI²OS is disabled and the circuit executes 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 EI²OS, the other interrupt requests cannot use it.

■ Operation of the DTP/external interrupt circuit

Table 16.5-1 shows the control bits and interrupt causes of the DTP/external interrupt circuit.

Table 16.5-1 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, the resource will generate an interrupt request signal 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 16.5-2 shows the operation of the DTP/external interrupt circuit.

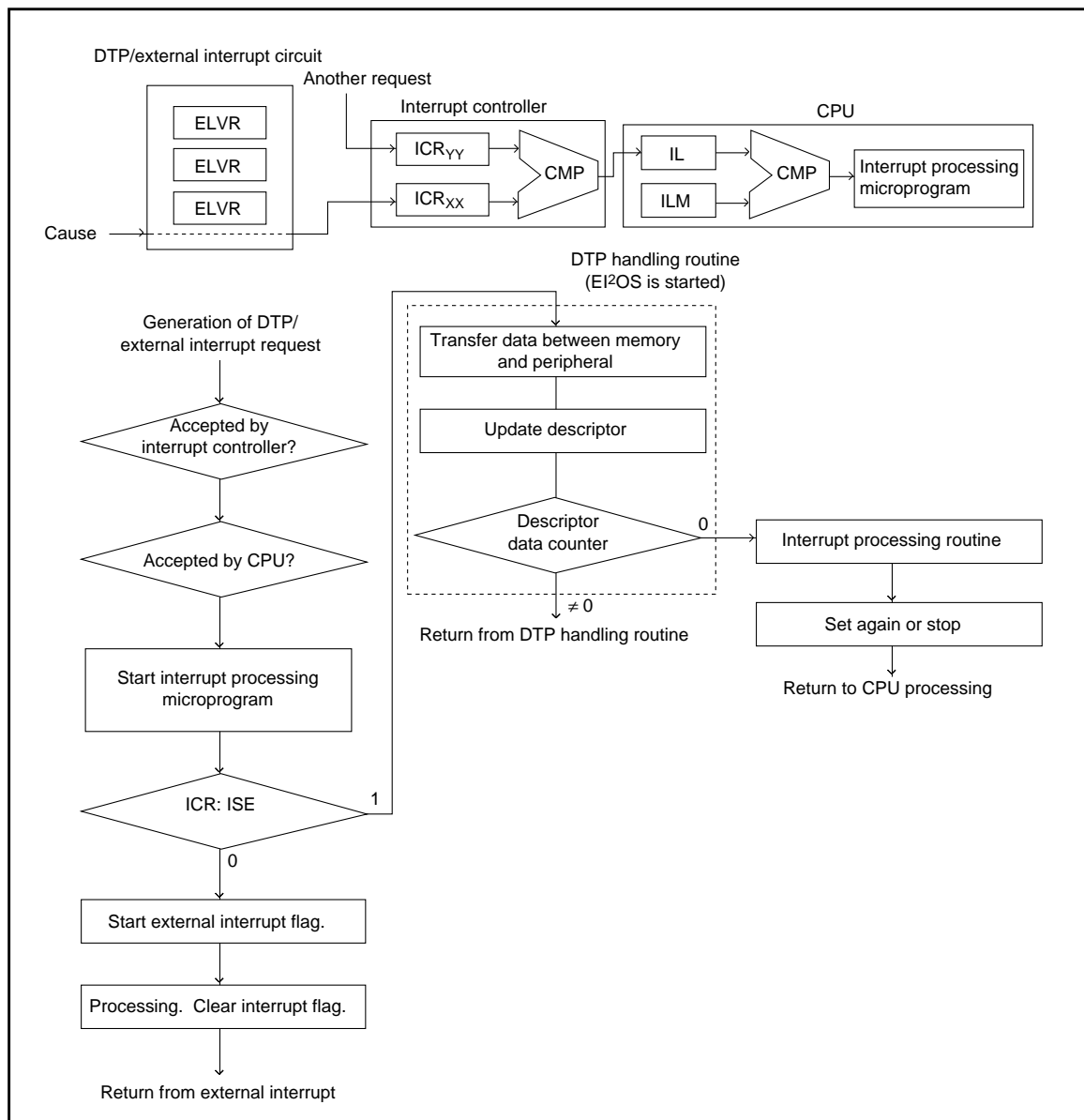


Figure 16.5-2 Operation of the DTP/external interrupt circuit

16.5.1 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 ER7 to ER0 bit of the DTP/interrupt cause register (EIRR) 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 magnitude 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 magnitudes 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.

16.5.2 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 (EI²OS) 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 EI²OS is enabled (ICR: ISE = 1), EI²OS 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 EI²OS is completed, control is transferred to the interrupt processing routine.

The external peripheral must remove only the level of the data transfer request signal (DTP external interrupt cause) within three cycles of the first transfer.

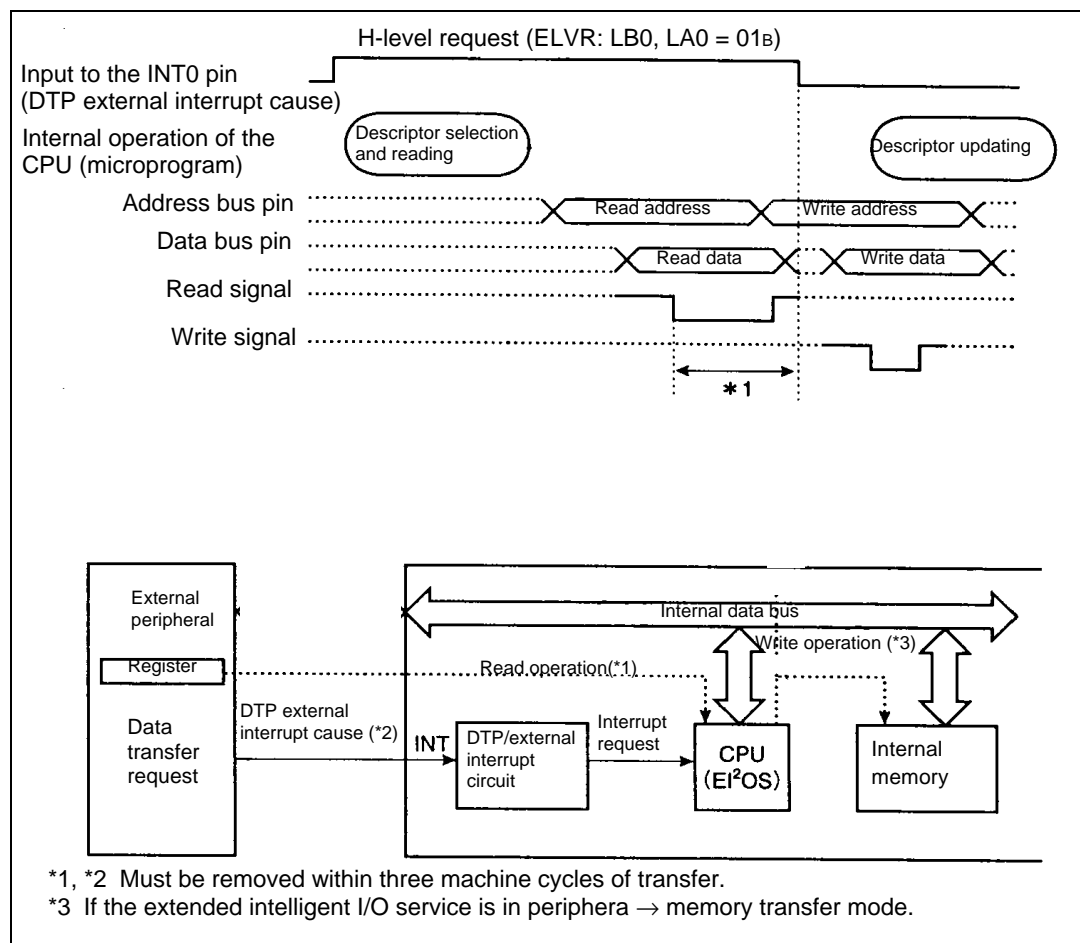


Figure 16.5-3 Example of interfacing to the external peripheral

16.6 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 16.6-1. 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 16.6-2.

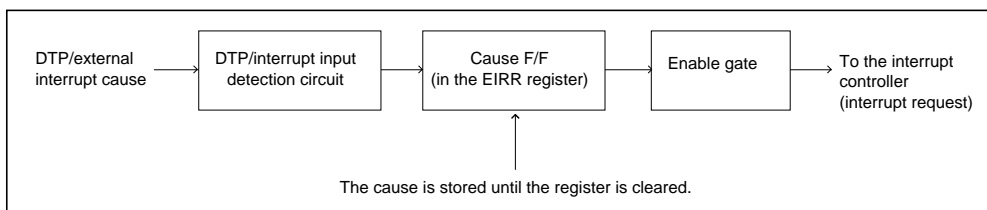


Figure 16.6-1 Clearing the cause retention circuit when a level is specified

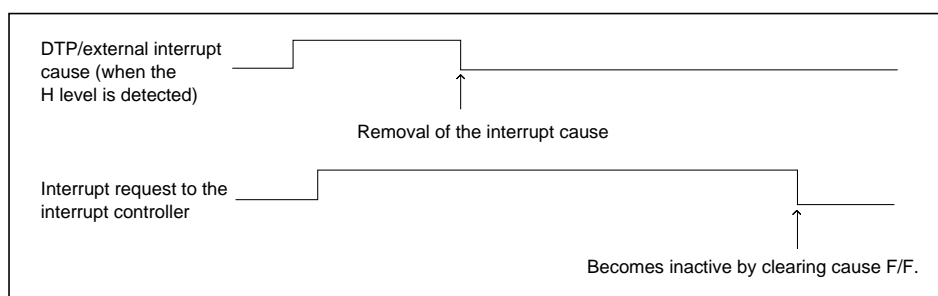


Figure 16.6-2 DTP/external interrupt cause and interrupt request when the output of interrupt requests is enabled

● 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, EI²OS 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.

16.7 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

```
ICR02    EQU    0000B2H        ; Interrupt control register for the DTP/external
                                   ; interrupt circuit
DDR6     EQU    000016H        ; Port 6 direction registe
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-----

CODECSEG

START:

```
;:      ;      Assumes that stack pointer (SP) has already been
              initialized.
MOV      I:DDR6,#00000000B ; Sets DDR6 as an input port.
AND      CCR,#0BFH        ; Disables interrupts.
MOV      I:ICR02,#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.
CODEENDS
;-----Vector setting-----

VECTCSEGABS=0FFH
        ORG    0FFC0H            ; Sets vector for interrupt #15 (0FH).
        DSL    WARI
        ORG    0FFDCH            ; Sets reset vector.
        DSL    START
        DB     00H                ; Sets single-chip mode.
VECTENDS
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 (EI²OS) is activated.
- Data is output from RAM to port 0 by DTP processing (EI²OS).

● Coding example

```

ICR02   EQU    0000B2H          ; Interrupt control register for the DTP/external
                                   ; interrupt circuit
DDR0    EQU    000010H          ; Port 0 direction register
DDR6    EQU    000011H          ; Port 6 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-----
CODECSEG
START:
;:      ;   Assumes that stack pointer (SP) has already been initialized.

```

```

MOV    I:DDR0,#11111111B ; Sets DDR0 as an output port.
MOV    I:DDR6,#00000000B ; Sets DDR6 as an input port.
AND    CCR,#0BFH         ; Disables interrupts.
MOV    I:ICR02,#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.

```

CODEENDS

;-----Vector setting-----

VECT CSEGABS=0FFH

```

      ORG    0FFC0H       ; Sets vector for interrupt #15 (0FH).
      DSL    WARI
      ORG    0FFDCH       ; Sets reset vector.
      DSL    START
      DB     00H          ; Sets single-chip mode.

```

VECT ENDS

END START

CHAPTER 17 CAN Controller

The MB90495 series contains one CAN controller (CAN1). The CAN controller is a module built into a 16-bit microcontroller (F²MC-16LX). The CAN (Controller Area Network) is the standard protocol for serial communication between automobile controllers and is widely used in industrial applications.

The CAN controller has the following features:

- Conforms to CAN Specification Version 2.0 Part A and B
- Supports transmission/reception in standard frame and extended frame formats
 - Supports transmitting of data frames by receiving remote frames
 - 8 transmitting/receiving message buffers
- 29-bit ID and 8-byte data
- Multi-level message buffer configuration
 - Provides full-bit comparison, full-bit mask and partial bit mask filtering.
- Two acceptance mask registers in either standard frame format or extended frame formats
 - Bit rate programmable from 10 Kbits/s to 1 Mbits/s (when input clock is at 16 MHz)

17.1 List of Control Registers

Table 17.1. -1 List of Control Registers

Address		Register	Abbreviation	Access	Initial Value
CAN0 reserved	CAN1				
000070H	000080H	Message buffer valid register	BVALR	R/W	00000000
000071H	000081H	reserved			
000072H	000082H	Transmit request register	TREQR	R/W	00000000
000073H	000083H	reserved			
000074H	000084H	Transmit cancel register	TCANR	W	00000000
000075H	000085H	reserved			
000076H	000086H	Transmit complete register	TCR	R/W	00000000
000077H	000087H	reserved			
000078H	000088H	Receive complete register	RCR	R/W	00000000
000079H	000089H	reserved			
00007AH	00008AH	Remote request receiving register	RRTRR	R/W	00000000
00007BH	00008BH	reserved			
00007CH	00008CH	Receive overrun register	ROVRR	R/W	00000000
00007DH	00008DH	reserved			
00007EH	00008EH	Receive interrupt enable register	RIER	R/W	00000000
00007FH	00008FH	reserved			

Table 17.1. -2 List of Control Registers

Address		Register	Abbreviation	Access	Initial Value
CAN0 reserved	CAN1				
003B00H	003D00H	Control status register	CSR	R/W, R	00---000 0----0-1
003B01H	003D01H				
003B02H	003D02H	Last event indicator register	LEIR	R/W	000--000
003B03H	003D03H	reserved			
003B04H	003D04H	Receive/transmit error counter	RTEC	R	00000000 00000000
003B05H	003D05H				
003B06H	003D06H	Bit timing register	BTR	R/W	-1111111 11111111
003B07H	003D07H				

Table 17.1. -2 List of Control Registers

Address		Register	Abbreviation	Access	Initial Value
CAN0 reserved	CAN1				
003B08H	003D08H	ID E register	IDER	R/W	xxxxxxxx
003B09H	003D09H	reserved			
003B0AH	003D0AH	Transmit R TR register	TRTRR	R/W	00000000
003B0BH	003D0BH	reserved			
003B0CH	003D0CH	Remote frame receive waiting register	RFWTR	R/W	xxxxxxxx
003B0DH	003D0DH	reserved			
003B0EH	003D0EH	Transmit interrupt enable register	TIER	R/W	00000000
003B0FH	003D0FH	reserved			
003B10H	003D10H	Acceptance mask select register	AMSR	R/W	xxxxxxxx xxxxxxxx
003B11H	003D11H				
003B12H	003D12H	reserved			
003B13H	003D13H				
003B14H	003D14H	Acceptance mask register 0	AMR0	R/W	xxxxxxxx xxxxxxxx
003B15H	003D15H				
003B16H	003D16H				xxxxx--- xxxxxxxx
003B17H	003D17H				
003B18H	003D18H	Acceptance mask register 1	AMR1	R/W	xxxxxxxx xxxxxxxx
003B19H	003D19H				
003B1AH	003D1AH				xxxxx--- xxxxxxxx
003B1BH	003D1BH				

17.2 Message Buffers

Table 17.2. -1 List of Message Buffers (ID Registers) (1)

Address		Register	Abbreviation	Access	Initial Value
CAN0 reserved	CAN1				
003A00 _H to 003A0F _H	003C00 _H to 003C0F _H	General-purpose RAM	--	R/W	XXXXXXXX to XXXXXXXX
003A10 _H	003C10 _H	ID register 0	IDR0	R/W	XXXXXXXX XXXXXXXX
003A11 _H	003C11 _H				XXXXXX--- XXXXXXXX
003A12 _H	003C12 _H				
003A13 _H	003C13 _H				
003A14 _H	003C14 _H	ID register 1	IDR1	R/W	XXXXXXXX XXXXXXXX
003A15 _H	003C15 _H				XXXXXX--- XXXXXXXX
003A16 _H	003C16 _H				
003A17 _H	003C17 _H				
003A18 _H	003C18 _H	ID register 2	IDR2	R/W	XXXXXXXX XXXXXXXX
003A19 _H	003C19 _H				XXXXXX--- XXXXXXXX
003A1A _H	003C1A _H				
003A1B _H	003C1B _H				
003A1C _H	003C1C _H	ID register 3	IDR3	R/W	XXXXXXXX XXXXXXXX
003A1D _H	003C1D _H				XXXXXX--- XXXXXXXX
003A1E _H	003C1E _H				
003A1F _H	003C1F _H				
003A20 _H	003C20 _H	ID register 4	IDR4	R/W	XXXXXXXX XXXXXXXX
003A21 _H	003C21 _H				XXXXXX--- XXXXXXXX
003A22 _H	003C22 _H				
003A23 _H	003C23 _H				
003A24 _H	003C24 _H	ID register 5	IDR5	R/W	XXXXXXXX XXXXXXXX
003A25 _H	003C25 _H				XXXXXX--- XXXXXXXX
003A26 _H	003C26 _H				
003A27 _H	003C27 _H				
003A28 _H	003C28 _H	ID register 6	IDR6	R/W	XXXXXXXX XXXXXXXX
003A29 _H	003C29 _H				XXXXXX--- XXXXXXXX
003A2A _H	003C2A _H				
003A2B _H	003C2B _H				

Table 17.2. -1 List of Message Buffers (ID Registers) (1)

Address		Register	Abbreviation	Access	Initial Value
CAN0 reserved	CAN1				
003A2CH	003C2CH	ID register 7	IDR7	R/W	XXXXXXXX XXXXXXXX
003A2DH	003C2DH				
003A2EH	003C2EH				XXXXX--- XXXXXXXX
003A2FH	003C2FH				

Table 17.2. -2 List of Message Buffers (DLC Registers and Data Registers) (1)

Address		Register	Abbreviation	Access	Initial Value
CAN0 reserved	CAN1				
003A30H	003C30H	DLC register 0	DLCR0	R/W	----XXXX
003A31H	003C31H				
003A32H	003C32H	DLC register 1	DLCR1	R/W	----XXXX
003A33H	003C33H				
003A34H	003C34H	DLC register 2	DLCR2	R/W	----XXXX
003A35H	003C35H				
003A36H	003C36H	DLC register 3	DLCR3	R/W	----XXXX
003A37H	003C37H				
003A38H	003C38H	DLC register 4	DLCR4	R/W	----XXXX
003A39H	003C39H				
003A3AH	003C3AH	DLC register 5	DLCR5	R/W	----XXXX
003A3BH	003C3BH				
003A3CH	003C3CH	DLC register 6	DLCR6	R/W	----XXXX
003A3DH	003C3DH				
003A3EH	003C3EH	DLC register 7	DLCR7	R/W	----XXXX
003A3FH	003C3FH				

Table 17.2. -3 List of Message Buffers (DLC Registers and Data Registers) (2)

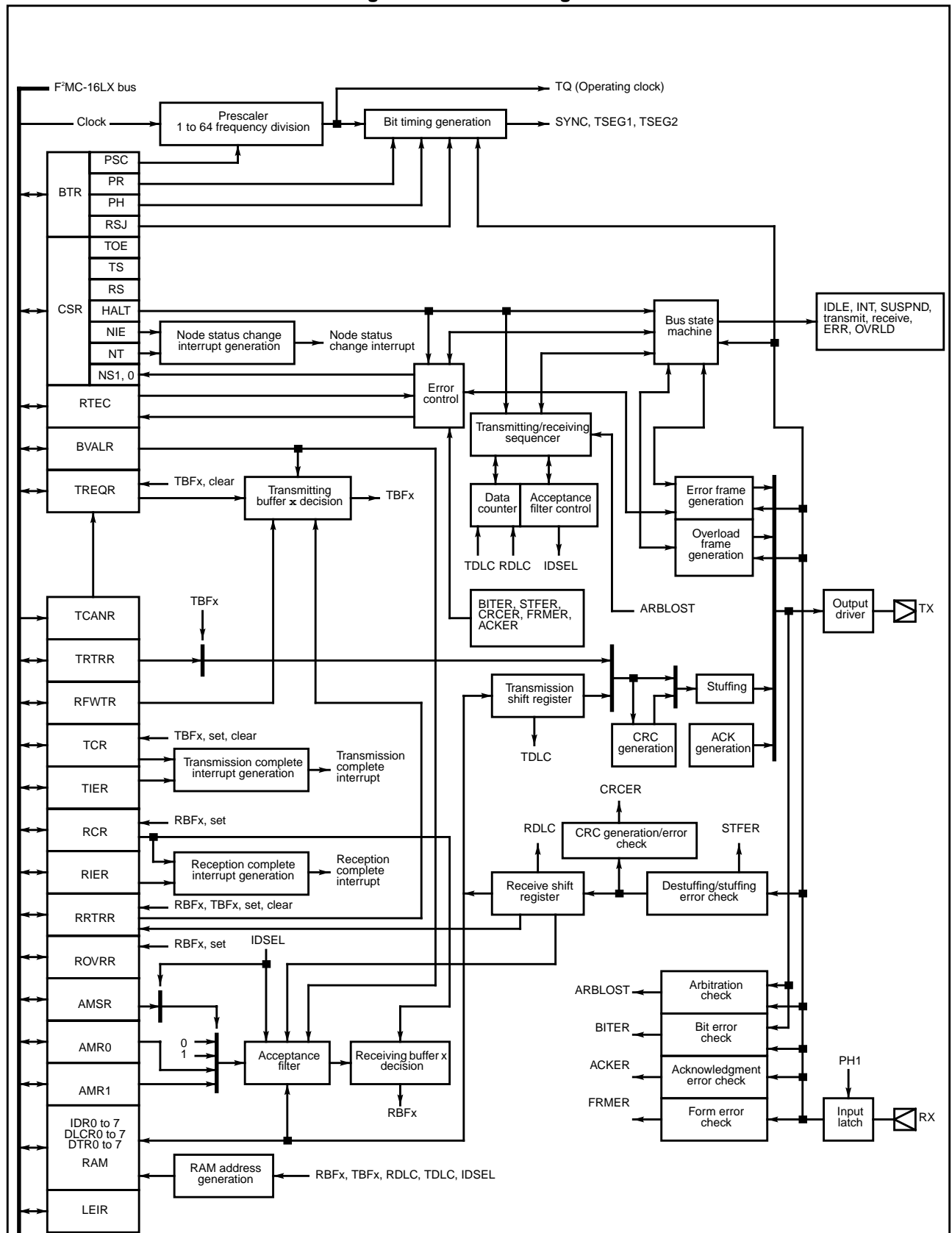
Address		Register	Abbreviation	Access	Initial Value
CAN0 reserved	CAN1				
003A40H to 003A47H	003C40H to 003C47H	Data register 0 (8 bytes)	DTR0	R/W	XXXXXXXX to XXXXXXXX

Table 17.2. -3 List of Message Buffers (DLC Registers and Data Registers) (2)

Address		Register	Abbreviation	Access	Initial Value
CAN0 reserved	CAN1				
003A48 _H to 003A4F _H	003C48 _H to 003C4F _H	Data register 1 (8 bytes)	DTR1	R/W	XXXXXXXX to XXXXXXXX
003A50 _H to 003A57 _H	003C50 _H to 003C57 _H	Data register 2 (8 bytes)	DTR2	R/W	XXXXXXXX to XXXXXXXX
003A58 _H to 003A5F _H	003C58 _H to 003C5F _H	Data register 3 (8 bytes)	DTR3	R/W	XXXXXXXX to XXXXXXXX
003A60 _H to 003A67 _H	003C60 _H to 003C67 _H	Data register 4 (8 bytes)	DTR4	R/W	XXXXXXXX to XXXXXXXX
003A68 _H to 003A6F _H	003C68 _H to 003C6F _H	Data register 5 (8 bytes)	DTR5	R/W	XXXXXXXX to XXXXXXXX
003A70 _H to 003A77 _H	003C70 _H to 003C77 _H	Data register 6 (8 bytes)	DTR6	R/W	XXXXXXXX to XXXXXXXX
003A78 _H to 003A7F _H	003C78 _H to 003C7F _H	Data register 7 (8 bytes)	DTR7	R/W	XXXXXXXX to XXXXXXXX
003A80 _H to 003AFF _H	003C80 _H to 003CFF _H	reserved			

17.3 Block Diagram

Figure 17.3-1 Block Diagram



17.4 Overall Control Register

17.4.1 CSR: Control Status Register

This register is prohibited from executing any bit manipulation instruction.

	15	14	13	12	11	10	9	8
Address: 003D01 _H (CAN1) Read/write:	TS	RS	—	—	—	NT	NS1	NS0
Initial value:	(R)	(R)	(—)	(—)	(—)	(R/W)	(R)	(R)
	(0)	(0)	(—)	(—)	(—)	(0)	(0)	(0)

	7	6	5	4	3	2	1	0
Address: 003D00 _H (CAN1) Read/write	TOE	—	—	—	—	NIE	Reserved	HALT
: Initial value:	(R/W)	(—)	(—)	(—)	(—)	(R/W)	(W)	(R/W)
	(0)	(—)	(—)	(—)	(—)	(0)	(0)	(1)

[Bit 15] **TS**: Transmit status bit

This bit indicates whether a message is being transmitted.

- 0: Message not being transmitted
- 1: Message being transmitted

This bit is 0 even while error and overload frames are transmitted.

[Bit 14] **RS**: Receive status bit

This bit indicates whether a message is being received.

- 0: Message not being received
- 1: Message being received

While a message is on the bus, this bit becomes 1. Therefore, this bit is also 1 while a message is being transmitted. This bit does not necessarily indicate whether a receiving message passes through the acceptance filter.

As a result, when this bit is 0, it implies that the bus operation is stopped (**HALT** = 0); the bus is in the intermission/bus idle or an error/overload frame is on the bus.

[Bit 10] **NT**: Node status transition flag

If the node status is changed to increment or from Bus Off to Error Active, this bit is set to 1.

In other words, the **NT** bit is set to 1 if the node status is changed from Error Active (00) to Warning (01), from Warning (01) to Error Passive (10), from Error Passive (10) to Bus Off (11), and from Bus Off (11) to Error Active (00). Numbers in parentheses indicate the values of **NS1** and **NS0** bits.

When the node status transition interrupt enable bit (**NIE**) is 1, an interrupt is generated. Writing 0 sets the **NT** bit to 0. Writing 1 to the **NT** bit is ignored. 1 is read when a Read Modify Write instruction is read.

[Bits 9 to 8] **NS1** and **NS0**: Node status bits 1 and 0

These bits indicate the current node status.

Table 17.4.1-1 Correspondence between NS1 and NS0 and Node Status

NS1	NS0	Node Status
0	0	Error active
0	1	Warning (error active)
1	0	Error passive
1	1	Bus off

Note: Warning (error active) is included in the error active in CAN Specification 2.0B for the node status, however, indicates that the transmit error counter or receive error counter has exceeded 96. The node status change diagram is shown.

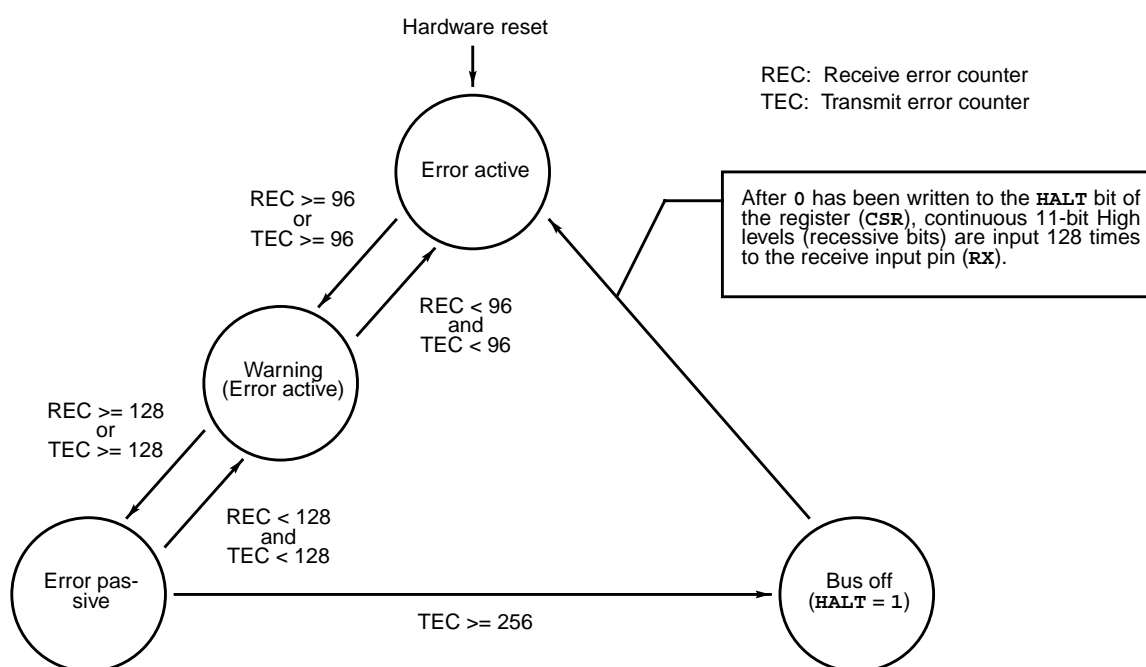


Figure 17.4.1-1 Node Status Transition Diagram

[Bit 7] **TOE**: Transmit output enable bit

Writing 1 to this bit switches from a general-purpose port pin to a transmit pin of the CAN controller.

- 0: General-purpose port pin
- 1: Transmit pin of CAN controller

[Bit 2] **NIE**: Node status transition interrupt enable bit

This bit enables or disables a node status transition interrupt (when **NT** = 1).

- 0: Node status transition interrupt disabled
- 1: Node status transition interrupt enabled

[Bit 1] Reserved bit. Always write 0 to this bit. Reading value is always 0.

[Bit 0] **HALT**: Bus operation stop bit

This bit sets or cancels bus operation stop, or displays its state.

(a) Conditions for setting bus operation stop (**HALT** = 1)

- After hardware reset
- When node status changed to bus off
- By writing 1 to **HALT**

Notes:

- 1) The bus operation should be stopped by writing 1 to **HALT** before the F²MC-16LX is changed in low-power consumption mode (stop mode, clock mode, and hardware stand-by mode).
- 2) If transmission is in progress when 1 is written to **HALT**, the bus operation is stopped (**HALT** = 1) after transmission is terminated. If reception is in progress when 1 is written to **HALT**, the bus operation is stopped immediately (**HALT** = 1). If received messages are being stored in the message buffer (**x**), stop the bus operation (**HALT** = 1) after storing the messages.
- 3) To check whether the bus operation has stopped, always read the **HALT** bit.

(b) Conditions for cancelling bus operation stop (**HALT** = 0)

- By writing 0 to **HALT**

Notes:

- 1) Cancelling the bus operation stop **after hardware reset** or by **writing 1 to HALT** as above conditions is performed after 0 is written to **HALT** and continuous 11-bit High levels (recessive bits) have been input to the receive input pin (**RX**) (**HALT** = 0).
- 2) Cancelling the bus operation stop when **the node status is changed to bus off** as above conditions is performed after 0 is written to **HALT** and continuous 11-bit High levels (recessive bits) have been input 128 times to the receive input pin (**RX**) (**HALT** = 0). Then, the values of both transmit and receive error counters reach 0 and the node status is changed to error active.

(c) State during bus operation stop (**HALT** = 1)

- The bus does not perform any operation, such as transmission and reception.
- The transmit output pin (**TX**) outputs a High level (recessive bit).
- The values of other registers and error counters are not changed.

Note: The bit timing register (**BTR**) should be set during bus operation stop (**HALT** = 1).

17.4.2 LEIR: Last Event Indicator Register

This register indicates the last event.

The **NTE**, **TCE**, and **RCE** bits are exclusive. When the corresponding bit of the last event is set to 1, other bits are set to 0s.

	7	6	5	4	3	2	1	0
Address: 003D02H (CAN1)	NTE	TCE	RCE	—	---	MBP2	MBP1	MBP0
Read/write :	(R/W)	(R/W)	(R/W)	(—)	(---)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(—)	(---)	(0)	(0)	(0)

[Bit 7] **NTE**: Node status transition event bit

When this bit is 1, it indicates that node status transition is the last event.

This bit is set to 1 at the same time as the **NT** bit of the control status register (**CSR**). This bit is also set to 1, irrespective of the setting of the node status transition interrupt enable bit (**NIE**) of the **CSR**.

Writing 0 to this bit sets the **NT** bit to 0. Writing 1 to this bit is ignored.

1 is read when a Read Modify Write instruction is performed.

[Bit 6] **TCE**: Transmit completion event bit

When this bit is 1, it indicates that transmit completion is the last event.

This bit is set to 1 at the same time as any one of the bits of the transmit completion register (**TCR**). This bit is also set to 1, irrespective of the settings of the bits of the transmit interrupt enable register (**TIER**).

Writing 0 sets this bit to 0. Writing 1 to this bit is ignored.

1 is read when a Read Modify Write instruction is performed.

When this bit is set to 1, the **MBP2** to **MBP0** bits are used to indicate the message buffer number completing the transmit operation.

[Bit 5] **RCE**: Receive completion event bit

When this bit is 1, it indicates that receive completion is the last event.

This bit is set to 1 at the same time as any one of the bits of the receive complete register (**RCR**). This bit is also set to 1 irrespective of the settings of the bits of the receive interrupt enable register (**RIER**).

Writing 0 sets this bit to 0. Writing 1 to this bit is ignored.

1 is read when a Read Modify Write instruction is performed.

When this bit is set to 1, the **MBP2** to **MBP0** bits are used to indicate the message buffer number completing the receive operation.

[Bits 2 to 0] **MBP2** to **MBP0**: Message buffer pointer bits

When the **TCE** or **RCE** bit is set to 1, these bits indicate the corresponding numbers of the message buffers (0 to 7). If the **NTE** bit is set to 1, these bits have no meaning.

Writing 0 sets these bits to 0s. Writing 1 to these bits is ignored.

1s are read when a Read Modify Write instruction is performed.

If LEIR is accessed within an CAN interrupt handler, the event causing the interrupt is not necessarily the same as indicated by LEIR. In the time from interrupt request to the LEIR access within the interrupt handler there may occur other CAN events.

17.4.3 RTEC: Receive and Transmit Error Counters

	15	14	13	12	11	10	9	8
Address: 003B05H (CAN0) 003D05H (CAN1)	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0
Read/write:	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

	7	6	5	4	3	2	1	0
Address: 003B04H (CAN0) 003D04H (CAN1)	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0
Read/write:	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

[Bits 15 to 8] **TEC7** to **TEC0**: Transmit error counter

These are transmit error counters.

TEC7 to **TEC0** values indicate 0 to 7 when the counter value is more than 256, and the subsequent increment is not counted for counter value. In this case, Error Passive is indicated for the node status (**NS1** and **NS0** of control status register **CSR** = 11).

[Bits 7 to 0] **REC7** to **REC0**: Receive error counter

These are receive error counters.

REC7 to **REC0** values indicate 0 to 7 when the counter value is more than 256, and the subsequent increment is not counted for counter value. In this case, Bus Off is indicated for the node status (**NS1** and **NS0** of control status register **CSR** = 10).

17.4.4 BTR: Bit Timing Register

This register stores the prescaler and bit timing setting.

	15	14	13	12	11	10	9	8
Address: 003B07H (CAN0) 003D07H (CAN1)	—	TS2.2	TS2.1	TS2.0	TS1.3	TS1.2	TS1.1	TS1.0
Read/write:	(—)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(—)	(1)	(1)	(1)	(1)	(1)	(1)	(1)

	7	6	5	4	3	2	1	0
Address: 003B06H (CAN0) 003D06H (CAN1)	RSJ1	RSJ0	PSC5	PSC4	PSC3	PSC2	PSC1	PSC0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)

Note: This register should be set during bus operation stop (**HALT** = 1).

[Bits 14 to 12] **TS2.2** to **TS2.0**: Time segment 2 setting bits 2 to 0
These bits define the number of time quanta (**TQ's**) for the time segment 2 (**TSEG2**). The time segment 2 is equal to the phase buffer segment 2 (**PHASE_SEG2**) in the CAN specification.

[Bits 11 to 8] **TS1.3** to **TS1.0**: Time segment 1 setting bits 3 to 0
These bits define the number of time quanta (**TQ's**) for the time segment 1 (**TSEG1**).. The time segment 1 is equal to the propagation segment (**PROP_SEG**) + phase buffer segment 1 (**PHASE_SEG1**) in the CAN specification.

[Bits 7 and 6] **RSJ1** and **RSJ0**: Resynchronization jump width setting bits 1 and 0
These bits cause the time quanta (**TQ**) to undergo [(**RSJ1** to **RSJ0**) +1] frequency division to determine the resynchronization jump width.

[Bits 5 to 0] **PSC5** to **PSC0**: Prescaler setting bits 5 to 0
These bits cause the input clock to undergo [(**PSC5** to **PSC0**) +1] frequency division to determine the time quanta (**TQ**) of the CAN controller. The bit time segments in the CAN specification, CAN controller are shown in **Figures 2.13.3** and **2.13.4** respectively.

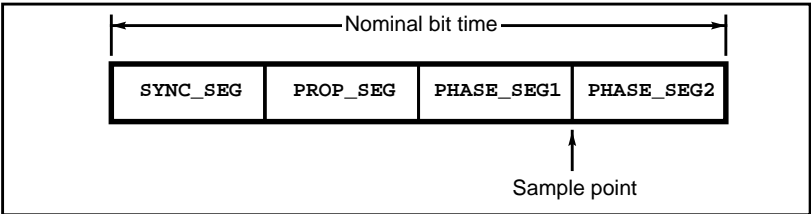


Figure 17.4.4-1 Bit Time Segment in CAN Specification

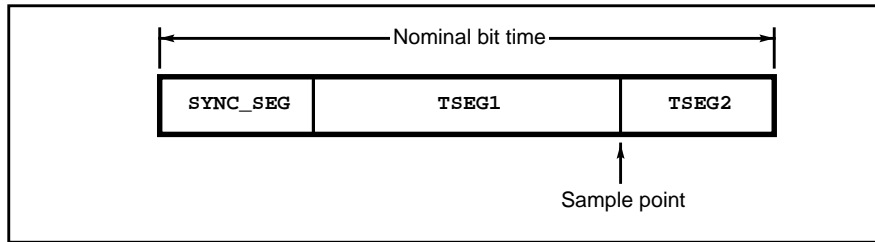


Figure 17.4.4-2 Bit Time Segment in CAN Controller

The relationship between $PSC = PSC5$ to $PSC0$, $TS1 = TS1.3$ to $TS1.0$, $TS2 = TS2.2$ to $TS1.0$, and $RSJ = RSJ1$ and $RSJ0$ when the input clock (CLK), time quanta (TQ), bit time (BT), synchronous segment ($SYNC_SEG$), time segment 1 and 2 ($TSEG1$ and $TSEG2$), and resynchronization jump width [$(RSJ1$ and $RSJ0) + 1$] frequency division is shown below.

The input clock is supplied with the machine clock.

$$\begin{aligned}
 TQ &= (PSC + 1) \times CLK \\
 BT &= SYNC_SEG + TSEG1 + TSEG2 \\
 &= (1 + (TS1 + 1) + (TS2 + 1)) \times TQ \\
 &= (3 + TS1 + TS2) TQ \\
 RSJW &= (RSJ + 1) \times TQ
 \end{aligned}$$

For correct operation, the following conditions should be met.

$$\begin{aligned}
 BT &\geq 8TQ \\
 TSEG2 &\geq RSJW + 2TQ^{*1} \\
 TSEG1 &\geq \text{delay time}^{*2} + RSJW
 \end{aligned}$$

*1) $2TQ$: Data processing time

*2) Delay time: Twice as long as the sum of the bus propagation, input comparator and output driver delay

17.5 Message Buffer Control Registers

17.5.1 BVALR: Message Buffer Valid Register

	7	6	5	4	3	2	1	0
Address: 000070H (CAN0) 000080H (CAN1)	BVAL7	BVAL6	BVAL5	BVAL4	BVAL3	BVAL2	BVAL1	BVAL0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

This register stores the validity of the message buffers or displays their state.

- 0: Message buffer (**x**) invalid
- 1: Message buffer (**x**) valid

If the message buffer (**x**) is set to invalid, it will not transmit or receive messages.

If the buffer is set to invalid during the transmission operation, it becomes invalid (**BVALx** = 0) after the transmission is completed or terminated by an error.

If the buffer is set to invalid during the receive operation, it immediately becomes invalid (**BVALx** = 0). If received messages are stored in a message buffer (**x**), the message buffer (**x**) is invalid after storing the messages.

Notes:

- 1) **x** indicates a message buffer number (**x** = 0 to 15).
- 2) When invalidating a message buffer (**x**) by writing 0 to a bit (**BVALx**), execution of a bit manipulation instruction is prohibited until the bit is set to 0.

17.5.2 IDER: IDE register

	7	6	5	4	3	2	1	0
Address: 003B08H (CAN0) 003D08H (CAN1)	IDE7	IDE6	IDE5	IDE4	IDE3	IDE2	IDE1	IDE0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

This register stores the frame format used by the message buffers during transmission/reception.

- 0: The standard frame format (**ID11** bit) is used for the message buffer (**x**).
- 1: The extended frame format (**ID29** bit) is used for the message buffer (**x**).

Note: This register should be set when the message buffer (**x**) is invalid (**BVALx** of the message buffer valid register (**BVALR**) = 0). Setting when the buffer is valid (**BVALx** = 1) may cause unnecessary received messages to be stored.

17.5.3 TREQR: Transmission Request Register

	7	6	5	4	3	2	1	0
Address: 000072H (CAN0) 000082H (CAN1)	TREQ7	TREQ6	TREQ5	TREQ4	TREQ3	TREQ2	TREQ1	TREQ0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

This register stores transmission requests to the message buffers or displays their state.

When 1 is written to **TREQ_x**, transmission to the message buffer (**x**) starts. If **RFWT_x** of the remote frame receiving wait register (**RFWTR**)*¹ is 0, transmission starts immediately. However, if **RFWT_x** = 1, transmission starts after waiting until a remote frame is received (**RRTR_x** of the remote request receiving register (**RRTRR**)*¹ becomes 1). Transmission starts*² immediately even when **RFWT_x** = 1, if **RRTR_x** is already 1 when 1 is written to **TREQ_x**.

*1: For **TRTRR** and **RFWTR**, see 17.5.4 and 17.5.5.

*2: For cancellation of transmission, see 17.5.6 and 17.7.

Writing 0 to **TREQ_x** is ignored.

0 is read when a Read Modify Write instruction is performed.

If clearing (to 0) at completion of the transmit operation and setting by writing 1 are concurrent, clearing is preferred.

If 1 is written to more than one bit, transmission is performed, starting with the lower-numbered message buffer (**x**).

TREQ_x is 1 while transmission is pending, and becomes 0 when transmission is completed or cancelled.

17.5.4 TRTRR: Transmission RTR Register

	7	6	5	4	3	2	1	0
Address: 003B0AH (CAN0) 003D0AH (CAN1)	TRTR7	TRTR6	TRTR5	TRTR4	TRTR3	TRTR2	TRTR1	TRTR0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

This register sets the **RTR** (Remote Transmission Request) bit during transmission by the message buffers.

0: Data frame

1: Remote frame

17.5.5 RFWTR: Remote Frame Receiving Wait Register

	7	6	5	4	3	2	1	0
Address: 003B0CH (CAN0) 003D0CH (CAN1)	RFWT7	RFWT6	RFWT5	RFWT4	RFWT3	RFWT2	RFWT1	RFWT0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

This register stores the condition for starting transmission when a request for data frame transmission is set (**TREQ_x** of the transmission request register (**TREQR**) is 1 and **TRTR_x** of the transmitting **RTR** register (**TRTRR**) is 0).

0: Transmission starts immediately

1: Transmission starts after waiting until remote frame received (**RRTR_x** of remote request receiving register (**RRTRR**) becomes 1)

Notes:

- 1) Transmission starts immediately if **RRTR_x** is already 1 when a request for transmission is set.
- 2) For remote frame transmission, do not set **RFWT_x** to 1.

17.5.6 TCANR: Transmission Cancel Register

	7	6	5	4	3	2	1	0
Address: 000074H (CAN0) 000084H (CAN1)	TCAN7	TCAN6	TCAN5	TCAN4	TCAN3	TCAN2	TCAN1	TCAN0
Read/write:	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

When 1 is written to **TCAN_x**, this register cancels a pending request for transmission to the message buffer (**x**).

At completion of transmission, **TREQ_x** of the transmission request register (**TREQR**) becomes 0. Writing 0 to **TCAN_x** is ignored.

This is a write-only register and its read value is always 0.

17.5.7 TCR: Transmission Complete Register

	7	6	5	4	3	2	1	0
Address: 000076H (CAN0) 000086H (CAN1)	TC7	TC6	TC5	TC4	TC3	TC2	TC1	TC0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

At completion of transmission by the message buffer (**x**), the corresponding **TC_x** becomes 1.

If **TIEX** of the transmission complete interrupt enable register (**TIER**) is 1, an interrupt occurs.

[Conditions for **TC_x** = 0]

- Write 0 to **TCx**.
- Write 1 to **TREQx** of the transmission request register (**TREQR**).

After the completion of transmission, write 0 to **TCx** to set it to 0. Writing 1 to **TCx** is ignored. 1 is read when a Read Modify Write instruction is performed.

Note: If setting to 1 by completion of the transmit operation and clearing to 0 by writing occur at the same time, the bit is set to 1.

17.5.8 TIER: Transmission Interrupt Enable Register

	7	6	5	4	3	2	1	0
Address: 003B0EH (CAN0) 003D0EH (CAN1)	TIE7	TIE6	TIE5	TIE4	TIE3	TIE2	TIE1	TIE0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

This register enables or disables the transmission interrupt by the message buffer (**x**). The transmission interrupt is generated at transmission completion (when **TCx** of the transmission complete register (**TCR**) is 1).

- 0: Transmission interrupt disabled
- 1: Transmission interrupt enabled

17.5.9 RCR: Reception Complete Register

	7	6	5	4	3	2	1	0
Address: 000078H (CAN0) 000088H (CAN1)	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

At completion of storing received message in the message buffer (**x**), **RCx** becomes 1.

If **RIEx** of the reception complete interrupt enable register (**RIER**) is 1, an interrupt occurs.

[Conditions for **RCx** = 0]

- Write 0 to **RCx**.

After completion of handling received message, write 0 to **RCx** to set it to 0. Writing 1 to **RCx** is ignored. 1 is read when a Read Modify Write instruction is performed.

Note: If setting (to 1) at completion of the receive operation and clearing by writing 0 are concurrent, setting is preferred.

17.5.10 RRTRR: Remote Request Receiving Register

	7	6	5	4	3	2	1	0
Address: 00007AH (CAN0) 00008AH (CAN1)	RRTR7	RRTR6	RRTR5	RRTR4	RRTR3	RRTR2	RRTR1	RRTR0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

After a remote frame is stored in the message buffer (**x**), **RRTRx** becomes 1 (at the same time as **RCx** setting to 1).

[Conditions for **RRTRx** = 0]

- Write 0 to **RRTRx**.
- After a received data frame is stored in the message buffer (**x**) (at the same time as **RCx** setting to 1).
- Transmission by the message buffer (**x**) is completed (**TCx** of the transmission complete register (**TCR**) is 1).

Writing 1 to **RRTRx** is ignored.

1 is read when a Read Modify Write instruction is read.

Note: If setting (to 1) and clearing by writing 0 are concurrent, setting is preferred.

17.5.11 ROVRR: Receive Overrun Register

	7	6	5	4	3	2	1	0
Address: 00007CH (CAN0) 00008CH (CAN1)	ROVR7	ROVR6	ROVR5	ROVR4	ROVR3	ROVR2	ROVR1	ROVR0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

If **RCx** of the reception complete register (**RCR**) is 1 when completing storing of a received message in the message buffer (**x**), **ROVRx** becomes 1, indicating that reception has overrun.

Writing 0 to **ROVRx** results in **ROVRx** = 0. Writing 1 to **ROVRx** is ignored. After checking that reception has overrun, write 0 to **ROVRx** to set it to 0.

1 is read when a Read Modify Write instruction is performed.

Note: If setting (to 1) and clearing by writing 0 are concurrent, setting is preferred.

17.5.12 RIER: Reception Interrupt Enable Register

	7	6	5	4	3	2	1	0
Address: 00007EH (CAN0) 00008EH (CAN1)	RIE7	RIE6	RIE5	RIE4	RIE3	RIE2	RIE1	RIE0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

This register enables or disables the reception interrupt by the message buffer (**x**).

The reception interrupt is generated at reception completion (when **RCx** of the reception completion register (**RCR**) is 1).

- 0: Reception interrupt disabled
- 1: Reception interrupt enabled

17.5.13 AMSR: Acceptance Mask Select Register

BYTE0	7	6	5	4	3	2	1	0
Address: 003B10H (CAN0) 003D10H (CAN1)	AMS3.1	AMS3.0	AMS2.1	AMS2.0	AMS1.1	AMS1.0	AMS0.1	AMS0.0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

BYTE1	15	14	13	12	11	10	9	8
Address: 003B11H (CAN0) 003D11H (CAN1)	AMS7.1	AMS7.0	AMS6.1	AMS6.0	AMS5.1	AMS5.0	AMS4.1	AMS4.0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

This register selects a masks (acceptance mask) for comparison between the received message **ID** and the message buffer (**x**) **ID**.

Table 17.5.13-1 Selection of Acceptance Mask

AMSx.1	AMSx.0	Acceptance Mask
0	0	Full-bit comparison
0	1	Full-bit mask
1	0	Acceptance mask register 0 (AMR0)
1	1	Acceptance mask register 1 (AMR1)

Note: **AMSx.1** and **AMSx.0** should be set when the message buffer (**x**) is invalid (**BVALx** of the message buffer valid register (**BVALR**) is 0). Setting when the buffer is valid (**BVALx** = 1) may cause unnecessary received messages to be stored.

17.5.14 AMR0 and AMR1: Acceptance Mask Registers 0 and 1

AMR0 BYTE0		7	6	5	4	3	2	1	0
Address: 003B14 _H (CAN0) 003D14 _H (CAN1)		AM28	AM27	AM26	AM25	AM24	AM23	AM22	AM21
	Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
	Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)
AMR0 BYTE1		15	14	13	12	11	10	9	8
Address: 003B15 _H (CAN0) 003D15 _H (CAN1)		AM20	AM19	AM18	AM17	AM16	AM15	AM14	AM13
	Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
	Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)
AMR0 BYTE2		7	6	5	4	3	2	1	0
Address: 003B16 _H (CAN0) 003D16 _H (CAN1)		AM12	AM11	AM10	AM9	AM8	AM7	AM6	AM5
	Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
	Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)
AMR0 BYTE3		15	14	13	12	11	10	9	8
Address: 003B17 _H (CAN0) 003D17 _H (CAN1)		AM4	AM3	AM2	AM1	AM0	—	—	—
	Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(—)	(—)	(—)
	Initial value:	(X)	(X)	(X)	(X)	(X)	(—)	(—)	(—)
AMR1 BYTE0		7	6	5	4	3	2	1	0
Address: 003B18 _H (CAN0) 003D18 _H (CAN1)		AM28	AM27	AM26	AM25	AM24	AM23	AM22	AM21
	Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
	Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)
AMR1 BYTE1		15	14	13	12	11	10	9	8
Address: 003B19 _H (CAN0) 003D19 _H (CAN1)		AM20	AM19	AM18	AM17	AM16	AM15	AM14	AM13
	Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
	Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

AMR1 BYTE2		7	6	5	4	3	2	1	0
Address: 003B1AH (CAN0) 003D1AH (CAN1)		AM12	AM11	AM10	AM9	AM8	AM7	AM6	AM5
	Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
	Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

AMR1 BYTE3		15	14	13	12	11	10	9	8
Address: 003B1BH (CAN0) 003D1BH (CAN1)		AM4	AM3	AM2	AM1	AM0	—	—	—
	Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(—)	(—)	(—)
	Initial value:	(X)	(X)	(X)	(X)	(X)	(—)	(—)	(—)

There are two acceptance mask registers, **AMR0** and **AMR1**, both of which are available either in the standard frame format or extended frame format.

AM28 to **AM18** (11 bits) are used for acceptance masks in the standard frame format and **AM28** to **AM0** (29 bits) are used for acceptance masks in the extended format.

0: Compare

Compare the bit of the acceptance code (**ID** register **IDRx** for comparing with the received message **ID**) corresponding to this bit with the bit of the received message **ID**. If there is no match, no message is received.

1: Mask

Mask the bit of the acceptance code **ID** register (**IDRx**) corresponding to this bit. No comparison is made with the bit of the received message **ID**.

Note: **AMR0** and **AMR1** should be set when all the message buffers (**x**) selecting **AMR0** and **AMR1** are invalid (**BVALx** of the message buffer valid register (**BVALR**) is 0). Setting when the buffers are valid (**BVALx** = 1) may cause unnecessary received messages to be stored.

17.6 Message Buffers

- There are 8 message buffers.
- One message buffer x ($x = 0$ to 7) consists of an **ID** register (**IDR x**), **DLC** register (**DLCR x**), and data register (**DTR x**).
- The message buffer (x) is used both for transmission and reception.
- The lower-numbered message buffers are assigned higher priority.
 - At transmission, when a request for transmission is made to more than one message buffer, transmission is performed, starting with the lowest-numbered message buffer (See 17.7).
 - At reception, when the received message **ID** passes through the acceptance filter (mechanism for comparing the acceptance-masked **ID** of received message and message buffer) of more than one message buffer, the received message is stored in the lowest-numbered message buffer (See 17.8).
- When the same acceptance filter is set in more than one message buffer, the message buffers can be used as a multi-level message buffer. This provides allowance for receiving time (See 17.9.7).

Notes:

- 1) A write operation to message buffers and general-purpose RAM areas should be performed in words to even addresses only. A write operation in bytes causes undefined data to be written to the upper byte at writing to the lower byte. Writing to the upper byte is ignored.
- 2) When the **BVAL x** bit of the message buffer valid register (**BVALR**) is 0 (Invalid), the message buffers x (**IDR x** , **DLCR x** , and **DTR x**) can be used as general-purpose RAM.
- 3) During the receive/transmit operation of the CAN controller, the CAN controller writes/reads to/from the message buffers. If the CPU tries to write/read to/from the message buffers in this period, the CPU has to wait a maximum time of 64 machine cycles.

This is also true for the general-purpose RAM.

17.6.1 IDRx: ID Register x (x = 0 to 7)

BYTE0	7	6	5	4	3	2	1	0
Address: 003A10 _H + 4x (CAN0) 003C10 _H + 4x (CAN1)	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)
BYTE1	15	14	13	12	11	10	9	8
Address: 003A11 _H + 4x (CAN0) 003C11 _H + 4x (CAN1)	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)
BYTE2	7	6	5	4	3	2	1	0
Address: 003A12 _H + 4x (CAN0) 003C12 _H + 4x (CAN1)	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)
BYTE3	15	14	13	12	11	10	9	8
Address: 003A13 _H + 4x (CAN0) 003C13 _H + 4x (CAN1)	ID4	ID3	ID2	ID1	ID0	—	—	—
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(—)	(—)	(—)
Initial value:	(X)	(X)	(X)	(X)	(X)	(—)	(—)	(—)

This is the ID register for message buffer (x).

When using the message buffer (x) in the standard frame format (IDEx of the IDE register (IDER) = 0), use 11 bits of ID28 to ID18. When using the buffer in the extended frame format (IDEx = 1), use 29 bits of ID28 to ID0.

ID28 to ID0 have the following functions:

- Set acceptance code (ID for comparing with the received message ID).
- Set transmitted message ID.
Note: In the standard frame format, setting 1s to all bits of ID28 to ID22 is prohibited).
- Store the received message ID.
Note: All received message ID bits are stored (even if bits are masked). In the standard frame format, ID17 to ID0 stores image of old message left in the receive shift register.

Notes:

- 1) A write operation to this register should be performed in words. A write operation in bytes causes undefined data to be written to the upper byte at writing to the lower byte. Writing to the upper byte is ignored.
- 2) This register should be set when the message buffer (x) is invalid (BVALx of the message buffer valid register (BVALR) is 0). Setting when the buffer is valid (BVALx = 1) may cause unnecessary received messages to be stored.

17.6.2 DLCR_x: DLC Register x ($x = 0$ to 7)

	7	6	5	4	3	2	1	0
Address: 003A30 _H + 2 x (CAN0)	—	—	—	—	DLC3	DLC2	DLC1	DLC0
003C30 _H + 2 x (CAN1)								
Read/write:	(—)	(—)	(—)	(—)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(—)	(—)	(—)	(—)	(X)	(X)	(X)	(X)

This is the **DLC** register for message buffer x .

[Transmission]

- Set the data length (byte count) of a transmitted message when a data frame is transmitted (**TRTR_x** of the transmitting **RTR** register (**TRTRR**) is 0).
- Set the data length (byte count) of a requested message when a remote frame is transmitted (**TRTR_x** = 1).

Note: Setting other than 0000 to 1000 (0 to 8 bytes) is prohibited.

[Reception]

- Store the data length (byte count) of a received message when a data frame is received (**RRTR_x** of the remote frame request receiving register (**RRTRR**) is 0).
- Store the data length (byte count) of a requested message when a remote frame is received (**RRTR_x** = 1).

Note: A write operation to this register should be performed in words. A write operation in bytes causes undefined data to be written to the upper byte at writing to the lower byte, and causes undefined data to be written to the lower byte at writing to the upper byte.

17.6.3 DTRx: Data Register x (x = 0 to 7)

BYTE0		7	6	5	4	3	2	1	0
Address: 003A40H + 8x (CAN0) 003C40H + 8x (CAN1)		D7	D6	D5	D4	D3	D2	D1	D0
	Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
	Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)
BYTE1		15	14	13	12	11	10	9	8
Address: 003A41H + 8x (CAN0) 003C41H + 8x (CAN1)		D7	D6	D5	D4	D3	D2	D1	D0
	Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
	Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)
BYTE2		7	6	5	4	3	2	1	0
Address: 003A42H + 8x (CAN0) 003C42H + 8x (CAN1)		D7	D6	D5	D4	D3	D2	D1	D0
	Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
	Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)
BYTE3		15	14	13	12	11	10	9	8
Address: 003A43H + 8x (CAN0) 003C43H + 8x (CAN1)		D7	D6	D5	D4	D3	D2	D1	D0
	Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
	Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)
BYTE4		7	6	5	4	3	2	1	0
Address: 003A44H + 8x (CAN0) 003C44H + 8x (CAN1)		D7	D6	D5	D4	D3	D2	D1	D0
	Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
	Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)
BYTE5		15	14	13	12	11	10	9	8
Address: 003A45H + 8x (CAN0) 003C45H + 8x (CAN1)		D7	D6	D5	D4	D3	D2	D1	D0
	Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
	Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

BYTE6	7	6	5	4	3	2	1	0
Address: 003A46 _H + 8 _x (CAN0)	D7	D6	D5	D4	D3	D2	D1	D0
003C46 _H + 8 _x (CAN1)								
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

BYTE7	15	14	13	12	11	10	9	8
Address: 003A47 _H + 8 _x (CAN0)	D7	D6	D5	D4	D3	D2	D1	D0
003C47 _H + 8 _x (CAN1)								
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

This is the data register for message buffer (**x**).

This register is used only in transmitting and receiving a data frame but not in transmitting and receiving a remote frame.

- Sets transmitted message data (any of 0 to 8 bytes).
Data is transmitted in the order of **BYTE0**, **BYTE1**, ..., **BYTE7**, starting with the MSB.
- Stores received message data.
Data is stored in the order of **BYTE0**, **BYTE1**, ..., **BYTE7**, starting with the MSB.
Even if the received message data is less than 8 bytes, the remaining bytes of the data register (**DTR_x**), to which data are stored, are undefined.

Note: A write operation to this register should be performed in words. A write operation in bytes causes undefined data to be written to the upper byte at writing to the lower byte. Writing to the upper byte is ignored.

17.7 Transmission

(1) Starting Transmission

When 1 is written to **TREQ_x** of the transmission request register (**TREQR**), transmission by the message buffer (**x**) starts. At this time, **TREQ_x** becomes 1 and **TC_x** of the transmission complete register (**TCR**) becomes 0.

If **RFWT_x** of the remote frame receiving wait register (**RFWTR**) is 0, transmission starts immediately. If **RFWT_x** is 1, transmission starts after waiting until a remote frame is received (**RRTR_x** of the remote request receiving register (**RRTRR**) becomes 1).

(2) Transmitting

If a request for transmission is made to more than one message buffer (more than one **TREQ_x** is 1), transmission is performed, starting with the lowest-numbered message buffer.

Message transmission to the CAN bus (by the transmit output pin **TX**) starts when the bus is idle.

If **TRTR_x** of the transmission **RTR** register (**TRTRR**) is 0, a data frame is transmitted. If **TRTR_x** is 1, a remote frame is transmitted.

If the message buffer competes with other CAN controllers on the CAN bus for transmission and arbitration fails or if an error occurs during transmission, the message buffer waits until the bus is idle and repeats retransmission until it is successful.

(3) Cancelling transmission request

1. Cancelling by transmission cancel register (**TCANR**)

A transmission request for message buffer (**x**) having not executed transmission during transmission pending can be cancelled by writing 1 to **TCAN_x** of the transmission cancel register (**TCANR**). At completion of cancellation, **TREQ_x** becomes 0.

2. Cancelling by storing received message

The message buffer (**x**) having not executed transmission despite transmission request also performs reception.

If the message buffer (**x**) has not executed transmission despite a request for transmission of a data frame (**TRTR_x** = 0 or **TREQ_x** = 1), the transmission request is cancelled after storing received data frames passing through the acceptance filter (**TREQ_x** = 0).

Note: A transmission request is not cancelled by storing remote frames (**TREQ_x** = 1 remains unchanged).

If the message buffer (**x**) has not executed transmission despite a request for transmission of a remote frame (**TRTR_x** = 1 or **TREQ_x** = 1), the transmission request is cancelled after storing received remote frames passing through the acceptance filter (**TREQ_x** = 0).

Note: The transmission request is cancelled by storing either data frames or remote frames.

(4) Completing transmission

When transmission is successful, **RRTR_x** becomes 0, **TREQ_x** becomes 0, and **TC_x** of the transmission complete register (**TCR**) becomes 1.

If the transmission complete interrupt is enabled (**TIE_x** of the transmission complete interrupt enable register (**TIER**) is 1), an interrupt occurs.

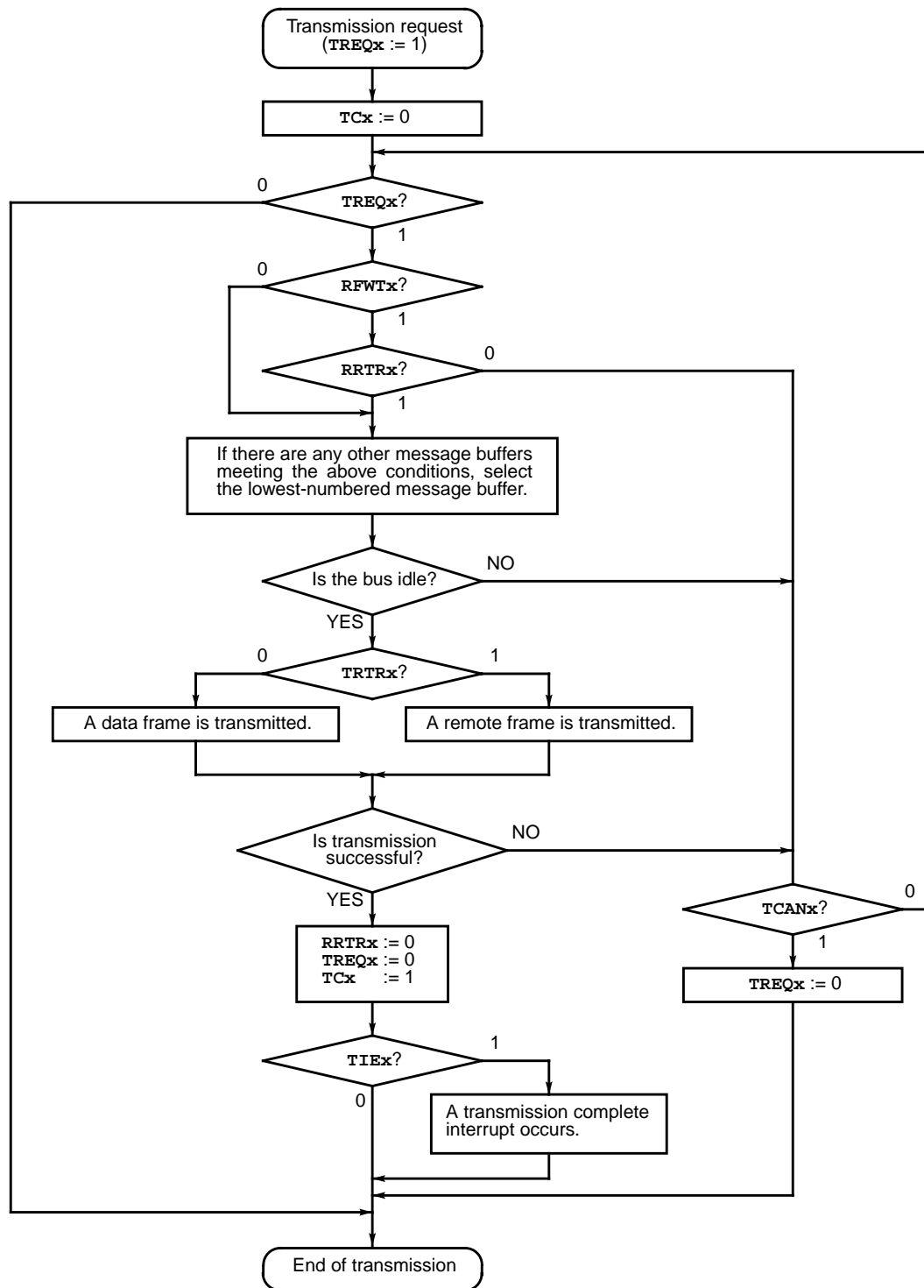


Figure 17.7-1 Transmission Flowchart

17.8 Reception

(1) Starting reception

Reception starts when the start of data frame or remote frame (SOF) is detected on the CAN bus.

(2) Acceptance filtering

The received message in the standard frame format is compared with the message buffer (**x**) set (**IDEx** of the **IDR** register (**IDR**) is 0) in the standard frame format. The received message in the extended frame format is compared with the message buffer (**x**) set (**IDEx** is 1) in the extended frame format.

If all the bits set to **Compare** by the acceptance mask agree after comparison between the received message **ID** and acceptance code (**ID** register (**IDR**) for comparing with the received message **ID**), the received message passes to the acceptance filter of the message buffer (**x**).

(3) Storing received message

When the receive operation is successful, received messages are stored in a message buffer **x** including **IDs** passed through the acceptance filter.

When receiving data frames, received messages are stored in the **ID** register (**IDR**), **DLC** register (**DLCR**), and data register (**DTR**).

Even if received message data is less than 8 bytes, the data is stored in the remaining bytes of the **DTR** and its value is undefined.

When receiving remote frames, received messages are stored only in the **IDR** and **DLCR**, and the **DTR** remains unchanged.

If there is more than one message buffer including **IDs** passed through the acceptance filter, the message buffer **x** in which received messages are to be stored is determined according to the following rules.

- The order of priority of the message buffer **x** (**x** = 0 to 7) rises as its number lower; in other words, message buffer 0 is given the highest and the message buffer 7 is given the lowest priority.
- Basically, message buffers with the **RC** bit of the receive completion register (**RCR**) are preferred in storing received messages.
- If the bits of the acceptance mask select register (**AMSR**) are set to All Bits Compare (for message buffers with the **AMS**.1 and **AMS**.0 bits set to 00), received messages are stored irrespective of the value of the **RC** bit of the **RCR**.
- If there are message buffers with the **RC** bit of the **RCR** set to 0, or with the bits of the **AMSR** set to All Bits Compare, received messages are stored in the lowest-number (highest-priority) message buffer **x**.
- If there are no message buffers above-mentioned, received messages are stored in a lower-number message buffer **x**.
- Message buffers should be arranged in ascending numeric order. The lowest message buffers should be with All Bits Compare, then **AMR0** or **AMR1** masks. And the highest message buffer should be with All Bits Mask.

Figure 17.8-1 shows the flowchart in determining the message buffer x where received messages are to be stored.

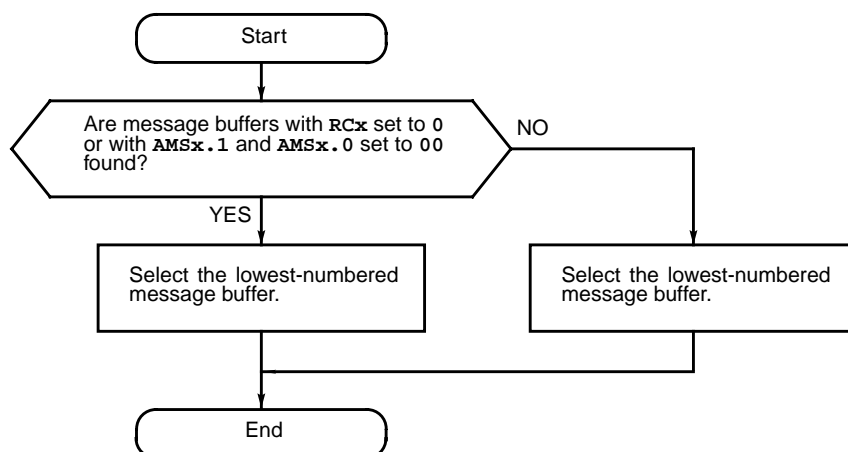


Figure 17.8-1 Flowchart Determining Message Buffer (x) where Received Messages Stored

(4) Receive overrun

When a message is stored in the message buffer with the corresponding RCx being already set to 1, it will result in receive overrun. In this case, the corresponding $ROVRx$ bit in the receive overrun register ($ROVRR$) is set to 1.

(5) Processing for reception of data frame and remote frame

1. Processing for reception of data frame

$RRTRx$ of the remote request receiving register ($RRTRR$) becomes 0.

$TREQx$ of the transmission request register ($TREQR$) becomes 0 (immediately before storing the received message). A transmission request for message buffer (x) having not executed transmission will be cancelled.

Note: A request for transmission of either a data frame or remote frame is cancelled.

2. Processing for reception of remote frame

$RRTRx$ becomes 0.

If $TRTRx$ of the transmitting RTR register ($TRTRR$) is 1, $TREQx$ becomes 0. A request for transmitting remote frame to message buffer having not executed transmission will be cancelled.

Notes:

- 1) A request for data frame transmission is not cancelled.
- 2) For cancellation of a transmission request, see **17.7.3**.

(6) Completing reception

RCx of the reception complete register (**RCR**) becomes 1 after storing the received message.

If a reception interrupt is enabled (**RIEx** of the reception interrupt enable register (**RIER**) is 1), an interrupt occurs.

Note: This CAN controller will not receive any messages transmitted by itself.

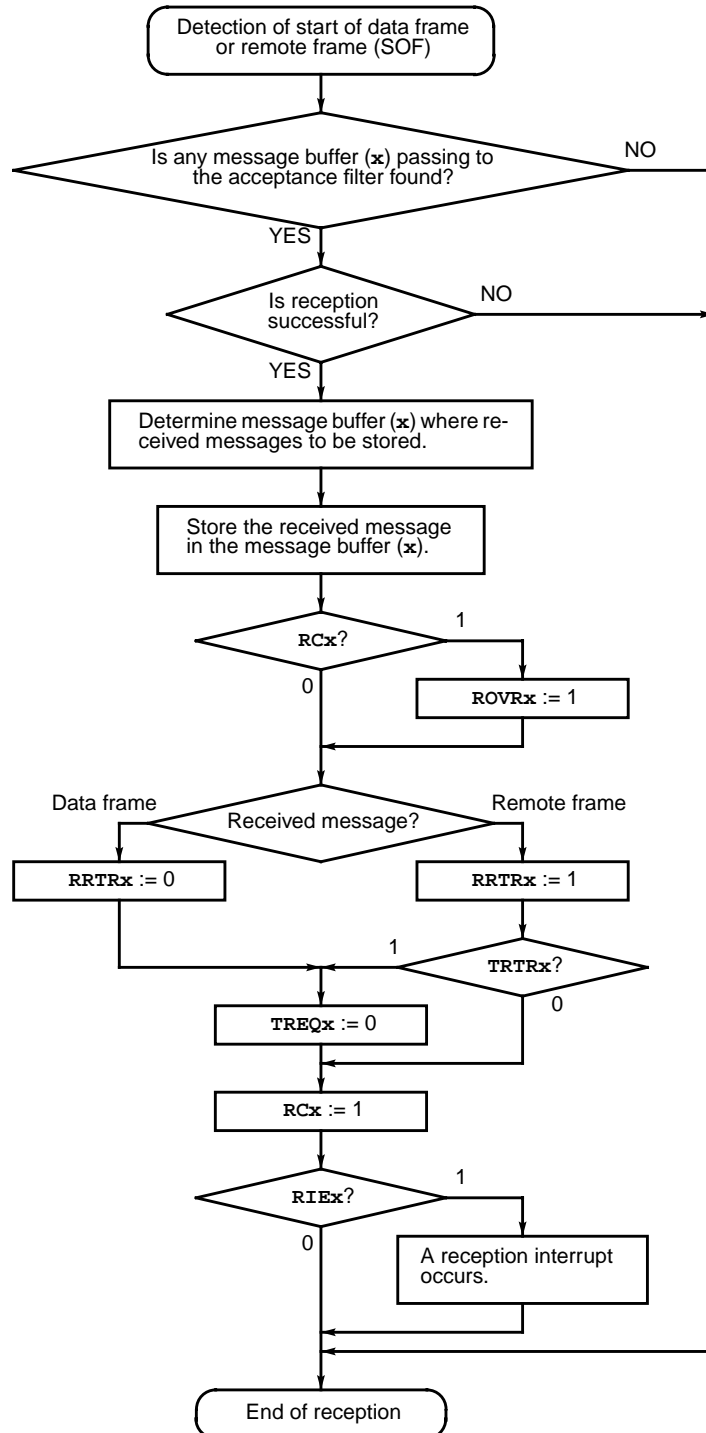


Figure 0.1 Reception Flowchart

17.9 Usage Procedure

17.9.1 Setting Bit Timing

The bit timing register (**BTR**) should be set during bus operation stop (when the bus operation stop bit (**HALT**) of the control status register (**CSR**) is 1).

After the setting completion, write 0 to **HALT** to cancel bus operation stop.

17.9.2 Setting Frame Format

Set the frame format used by the message buffer (**x**). When using the standard frame format, set **IDEx** of the **IDE** register (**IDER**) to 0. When using the extended frame format, set **IDEx** to 1.

This setting should be made when the message buffer (**x**) is invalid (**BVALx** of the message buffer valid register (**BVALR**) is 0). Setting when the buffer is valid (**BVALx** = 1) may cause unnecessary received messages to be stored.

17.9.3 Setting ID

Set the message buffer (**x**) **ID** to **ID28** to **ID0** of **ID** register (**IDRx**). The message buffer (**x**) **ID** need not be set to **ID11** to **ID0** in the standard frame format. The message buffer (**x**) **ID** is used as a transmission message at transmission and is used as an acceptance code at reception.

This setting should be made when the message buffer (**x**) is invalid (**BVALx** of the message buffer valid register (**BVALR**) is 0). Setting when the buffer is valid (**BVALx** = 1) may cause unnecessary received messages to be stored.

17.9.4 Setting Acceptance Filter

The acceptance filter of the message buffer (**x**) is set by an acceptance code and acceptance mask set. It should be set when the acceptance message buffer (**x**) is invalid (**BVALx** of the message buffer enable register (**BVALR**) is 0). Setting when the buffer is valid (**BVALx** = 1) may cause unnecessary received messages to be stored.

Set the acceptance mask used in each message buffer (**x**) by the acceptance mask select register (**AMSR**). The acceptance mask registers (**AMR0** and **AMR1**) should also be set if used (For the setting details, see **17.5.12** and **17.5.13**).

The acceptance mask should be set so that a transmission request may not be cancelled when unnecessary received messages are stored. For example, it should be set to a full-bit comparison if receiving only the message of the same **ID** as transmission **ID**.

17.9.5 Procedure for Transmission by Message Buffer (x)

After the completion of setting 17.9.2 to 17.9.4, set **BVALx** to 1 to make the message buffer (x) valid.

(1) Setting transmit data length code

Set the transmit data length code (byte count) to **DLC3** to **DLC0** of the **DLC** register (**DLCRx**).

For data frame transmission (when **TRTRx** of the transmission **RTR** register (**TRTRR**) is 0), set the data length of the transmitted message.

For remote frame transmission (when **TRTRx** = 1), set the data length (byte count) of the requested message.

Note: Setting other than 0000 to 1000 (0 to 8 bytes) is prohibited.

(2) Setting transmit data (only for transmission of data frame)

For data frame transmission (when **TRTRx** of the transmission register (**TRTRR**) is 0), set data as the count of byte transmitted in the data register (**DTRx**).

Note: Transmit data should be rewritten with the **TREQx** bit of the transmission request register (**TREQR**) set to 0.
There is no need for setting the **BVALx** bit of the message buffer valid register (**BVALR**) to 0. Setting the **BVALx** bit to 0 disables remote frame receiving.

(3) Setting transmission RTR register

For data frame transmission, set **TRTRx** of the transmission **RTR** register (**TRTRR**) to 0.

For remote frame transmission, set **TRTRx** to 1.

(4) Setting conditions for starting transmission (only for transmission of data frame)

Set **RFWTx** of the remote frame receiving wait register (**RFWTR**) to 0 to start transmission immediately after a request for data frame transmission is set (**TREQx** of the transmission request register (**TREQR**) is 1 and **TRTRx** of the transmission **RTR** register (**TRTRR**) is 0).

Set **RFWTx** to 1 to start transmission after waiting until a remote frame is received (**RRTRx** of the remote request receiving register (**RRTRR**) becomes 1) after a request for data frame transmission is set (**TREQx** = 1 and **TRTRx** = 0).

Note: Remote frame transmission can not be made, if **RFWTx** is set to 1.

(5) Setting transmission complete interrupt

When generating a transmission complete interrupt, set **TIEx** of the transmission complete interrupt enable register (**TIER**) to 1.

When not generating a transmission complete interrupt, set **TIEx** to 0.

(6) Setting transmission request

For a transmission request, set **TREQx** of the transmission request register (**TREQR**) to 1.

(7) Cancelling transmission request

When cancelling a pending request for transmission to the message buffer (**x**), write 1 to **TCANx** of the transmission cancel register (**TCANR**).

Check **TREQx**. For **TREQx** = 0, transmission cancellation is terminated or transmission is completed. Check **TCx** of the transmission complete register (**TCR**). For **TCx** = 0, transmission cancellation is terminated. For **TCx** = 1, transmission is completed.

(8) Processing for completion of transmission

If transmission is successful, **TCx** of the transmission complete register (**TCR**) becomes 1.

If the transmission complete interrupt is enabled (**TIEx** of the transmission complete interrupt enable register (**TIER**) is 1), an interrupt occurs.

After checking the transmission completion, write 0 to **TCx** to set it to 0. This cancels the transmission complete interrupt.

In the following cases, the pending transmission request is cancelled by receiving and storing a message.

- Request for data frame transmission by reception of data frame
- Request for remote frame transmission by reception of data frame
- Request for remote frame transmission by reception of remote frame

Request for data frame transmission is not cancelled by receiving and storing a remote frame. **ID** and **DLC**, however, are changed by the **ID** and **DLC** of the received remote frame. Note that the **ID** and **DLC** of data frame to be transmitted become the value of received remote frame.

17.9.6 Procedure for Reception by Message Buffer (x)

After setting 17.9.2 to 17.9.4, set as follows.

(1) Setting reception interrupt

When a reception interrupt is enabled, set **RIEx** of the reception interrupt enable register (**RIER**) to 1.

When a reception interrupt is disabled, set **RIEx** to 0.

(2) Starting reception

When starting reception after setting, set **BVALx** of the message buffer valid register (**BVALR**) to 1 to make the message buffer (**x**) valid.

(3) Processing for reception completion

If reception is successful after passing to the acceptance filter, the received message is stored in the message buffer (**x**) and **RCx** of the reception complete register (**RCR**) becomes 1. For data frame reception, **RRTRx** of the remote request receiving register (**RRTRR**) becomes 0. For remote frame reception, **RRTRx** becomes 1.

If a reception interrupt is enabled (**RIEx** of the reception interrupt enable register (**RIER**) is 1), an interrupt occurs.

After checking the reception completion (**RCx** = 1), process the received message.

After completion of processing the received message, check **ROVRx** of the reception overrun register (**ROVRR**).

For **ROVRx** = 0, the processed received message is valid. Write 0 to **CRx** to set it to 0 (the reception complete interrupt is also cancelled) to terminate reception.

For **ROVRx** = 1, a reception overrun occurred and the new received message may overwrite the processed received message. In this case, received messages should be processed again after setting the **ROVRx** bit to 0 by writing 0 to it.

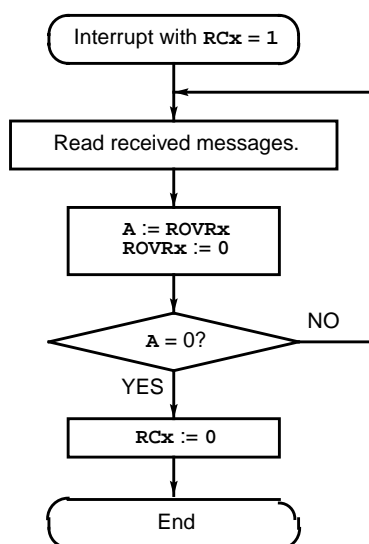


Table 17.9. -1 Example of Receive Interrupt Handling

17.9.7 Setting Configuration of Multi-level Message Buffer

If the receptions are performed frequently, or if several different ID's of messages are received, in other words, if there is insufficient time for receiving messages, more than one message buffer can be combined into a multi-level message buffer to provide allowance for processing time of the received message by CPU.

To provide a multi-level message buffer, the same acceptance filter must be set in the combined message buffers.

If the bits of the acceptance mask select register (**AMSR**) are set to All Bits Compare ((**AMSx.1**, **AMSx.0**) = (0, 0)), multi-level message configuration of message buffers is not allowed. This is because All Bits Compare causes received messages to be stored irrespective of the value of the **RCx** bit of the receive completion register (**RCR**), so received messages are always stored in lower-numbered (lower-priority) message buffers even if All Bits Compare and identical acceptance code (**ID** register (**IDRx**)) are specified for more than one message buffer. Therefore, All Bits Compare and identical acceptance code should not be specified for more than one message buffer.

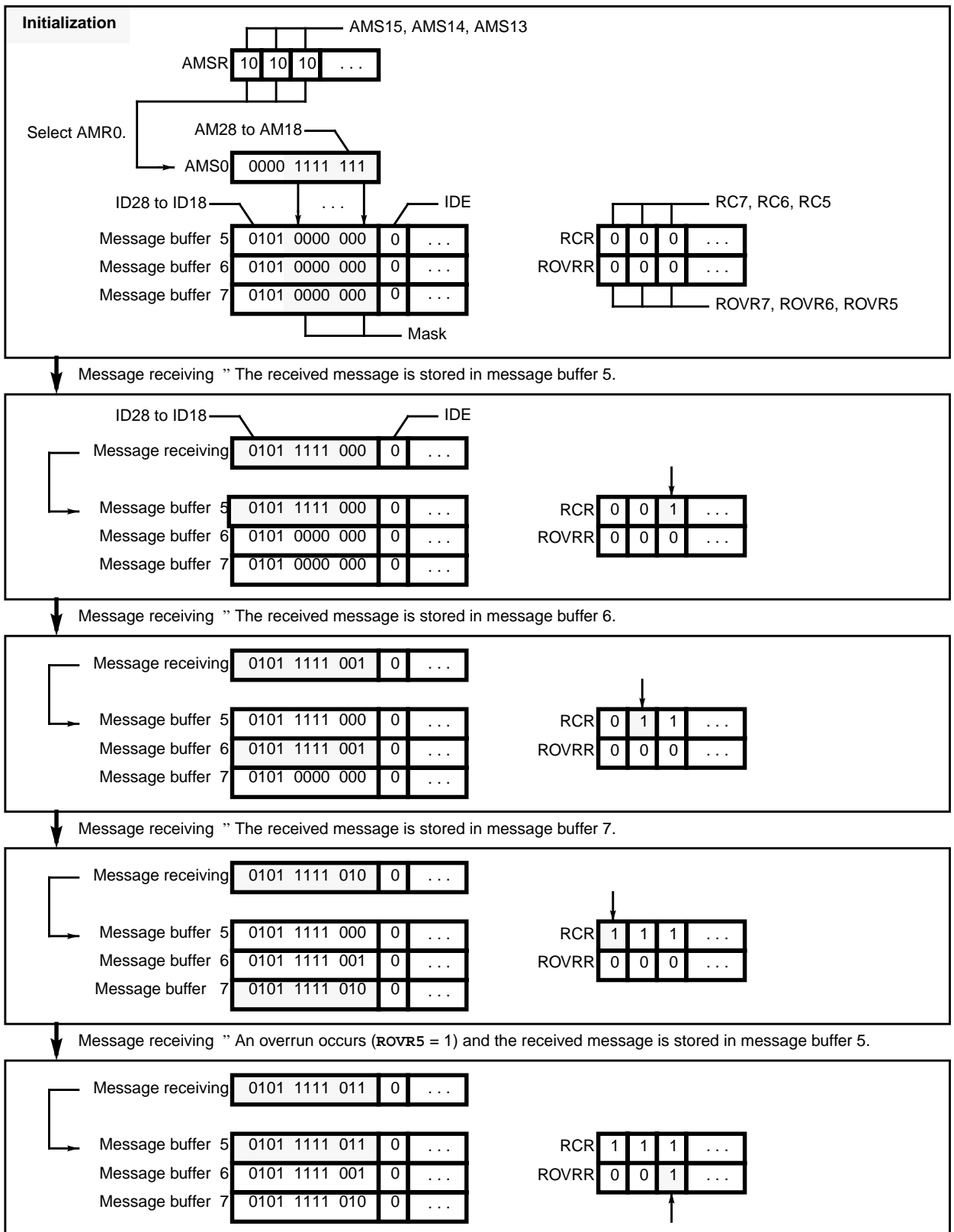


Figure 17.9-1 Examples of Operation of Multi-level Message Buffer

Note: Four messages are received with the same acceptance filter set in message buffers 5, 6 and 7.

17.9.8 Setting Low-power Consumption Mode

To set the F²MC-16LX in a low-power consumption mode (Stop, Watch etc.), write **1** to the bus operation stop bit (**HALT**) of the control status register (**CSR**), and then check that the bus operation has stopped (**HALT** = **1**).

CHAPTER 18 Delayed Interrupt Generator Module

This chapter describes the functions and operation of the MB90495 series delayed interrupt generator module.

18.1 Overview of the Delayed Interrupt Generator Module

18.2 Operation of the Delayed Interrupt Generator Module

18.1 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 18.1-1 shows the block diagram of the delayed interrupt generator module.

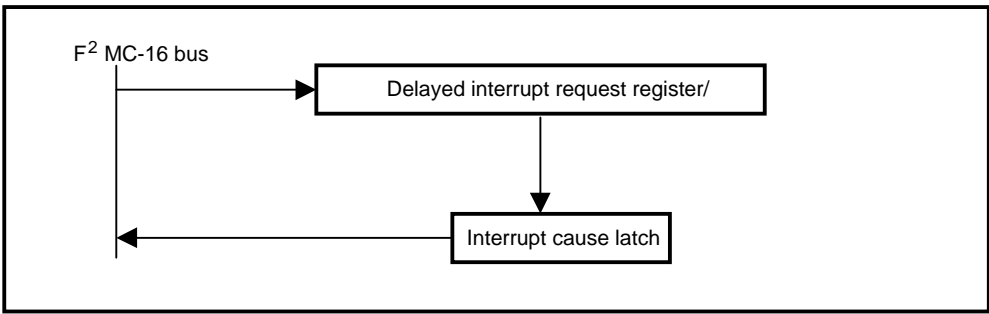


Figure 18.1-1 Block diagram of the delayed interrupt generator module

■ 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.

18.2 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 18.2-1 shows the operation of the delayed interrupt generator module.

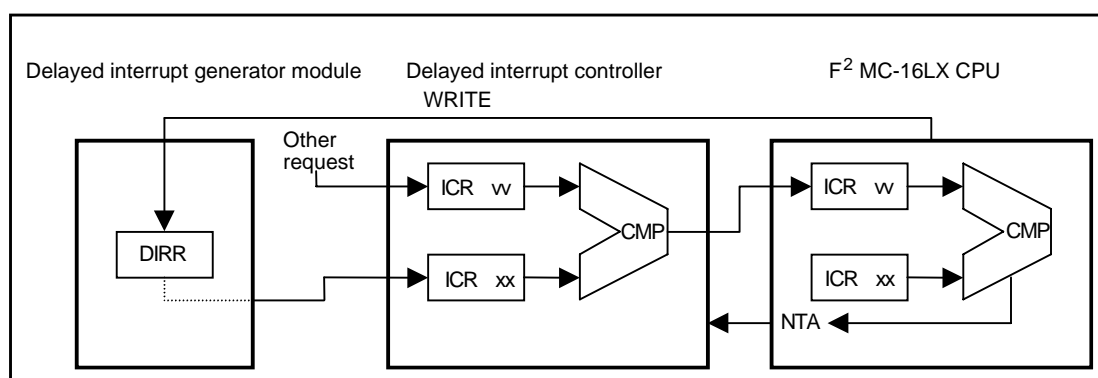


Figure 18.2-1 Operation of the delayed interrupt generator module

■ 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 19 ROM Correction

When the setting of the address is the same as the ROM Correction Address registers, the INT9 instruction will be executed. By processing the INT9 interrupt service routine, the ROM correction function can be achieved.

There are two address registers, in each containing a compare enable bit. When the address register and the program counter are in agreement, and when the compare enable bit is at '1', then the CPU will be forced to execute INT9 instruction.

19.1 Registers

19.2 Block Diagram

19.3 Register Details

19.4 Operations

19.5 Application Example

19.1 Registers

		byte	byte	byte	access	initial value
PADR0	1FF2H/1FF1H/1FF0H				R/W	undefined
PADR1	1FF5H/1FF4H/1FF3H				R/W	undefined

ROM Correction Control Register	7	6	5	4	3	2	1	0	↔ Bit number
Address : 009EH	Reserved	Reserved	Reserved	Reserved	AD1E	Reserved	AD0E	Reserved	PACSR
Read/write ↔	(-)	(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ↔	(1)	(1)	(0)	(0)	(0)	(0)	(0)	(0)	

Figure 19.1-1 ROM coorection Control Register

19.2 Block Diagram

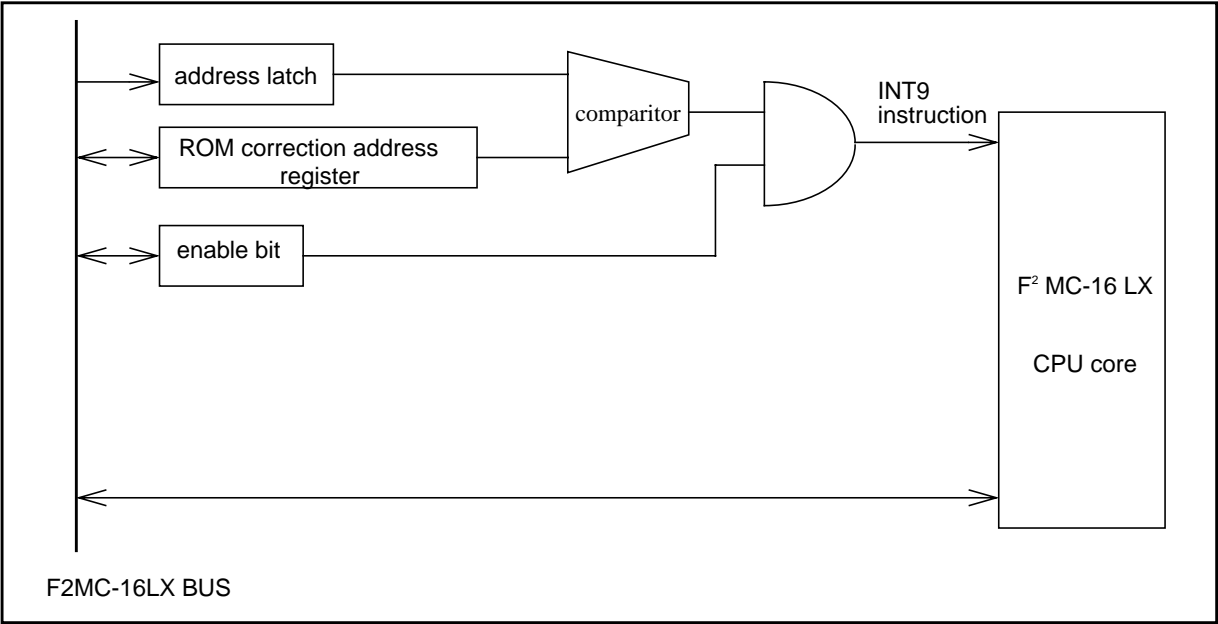


Figure 19.2-1 Block Diagram

[bit 2]:

Reserved bit.

[bit 1]: AD0E (Compare Enable 0)

This is the ADR0 enable bit.

When this bit is at '1', this module compares the PADR0 register and the program counter. If there is an agreement, the INT9 instruction is sent to the CPU.

[bit 0]:

Reserved bit.

19.4 Operations

When the program counter indicates the same address as the ROM Correction Address register, the INT9 instruction will be executed. By processing the INT9 interrupt service routine, the ROM correction function can be achieved.

There are two address registers, in each containing a compare enable bit. When the address register and the program counter are in agreement, and when the compare enable bit is at '1', then the CPU will be forced to execute INT9 instruction.

Note: When the address register and the program counter are in agreement, the internal data bus content will be forced to be '01H', so INT9 instruction will be executed. Before changing the content of the address register, make sure the compare enable bit is at '0'. If it is changed while the compare enable bit is at '1', there will occur an error.

19.5 Application Example

(1) System Structure

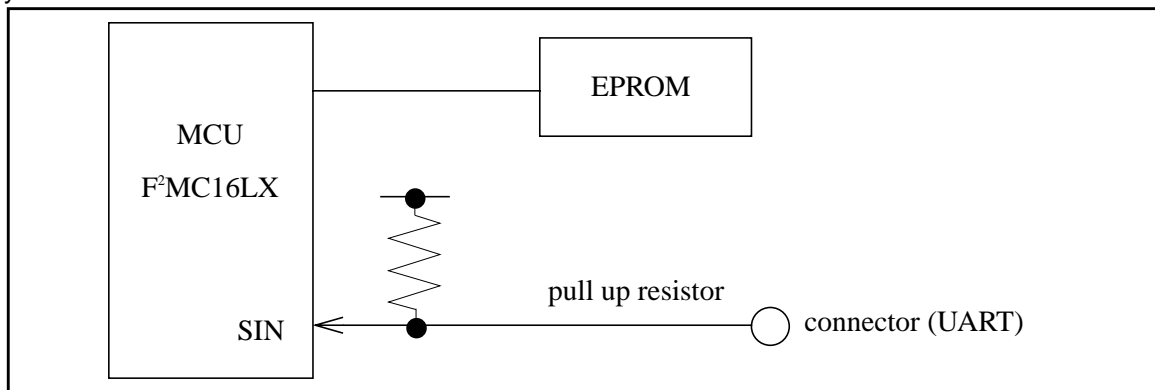


Figure 19.5-1 System Structure Example

(2) EPROM memory map

address:content

0000H: number of bytes of the corrected program No. 0 (0 implies no ROM correction)

0001H: bit 7-0 program address No. 0

0002H: bit 15-8 program address No. 0

0003H: bit 24-16 program address No. 0

0004H: number of bytes of the corrected program No. 1 (0 implies no ROM correction)
 0005H: bit 7-0 program address No. 1
 0006H: bit 15-8 program address No. 1
 0007H: bit 24-16 program address No. 1
 0010H~: corrected program No. 0/1 body

(3) Initial Condition

EPROM all at '0'.

(4) When ROM correction is needed

Send the body of the corrected program and the program address to the MCU through the connector (UART). MCU then writes that information into the EEPROM.

(5) Reset Sequence

After resetting, the MCU reads the contents of the EEPROM. If the byte number of the corrected program is not '0', the body of the corrected program is read from the EEPROM and written in the RAM. Then the MCU sets the correction address either on PADDR0 or on PADDR1 and sets the compare enable bit. First address of the corrected program can be written in the user-defined location of the RAM if a relocatable correction program is desired. In this case the INT9 service routine look for this user-defined location to jump to the corrected program.

(6) INT9 interrupt

In the interrupt routine, the address that produces the interrupt can be known by checking the stacked program counter value. The information stacked during interrupt will be discarded.

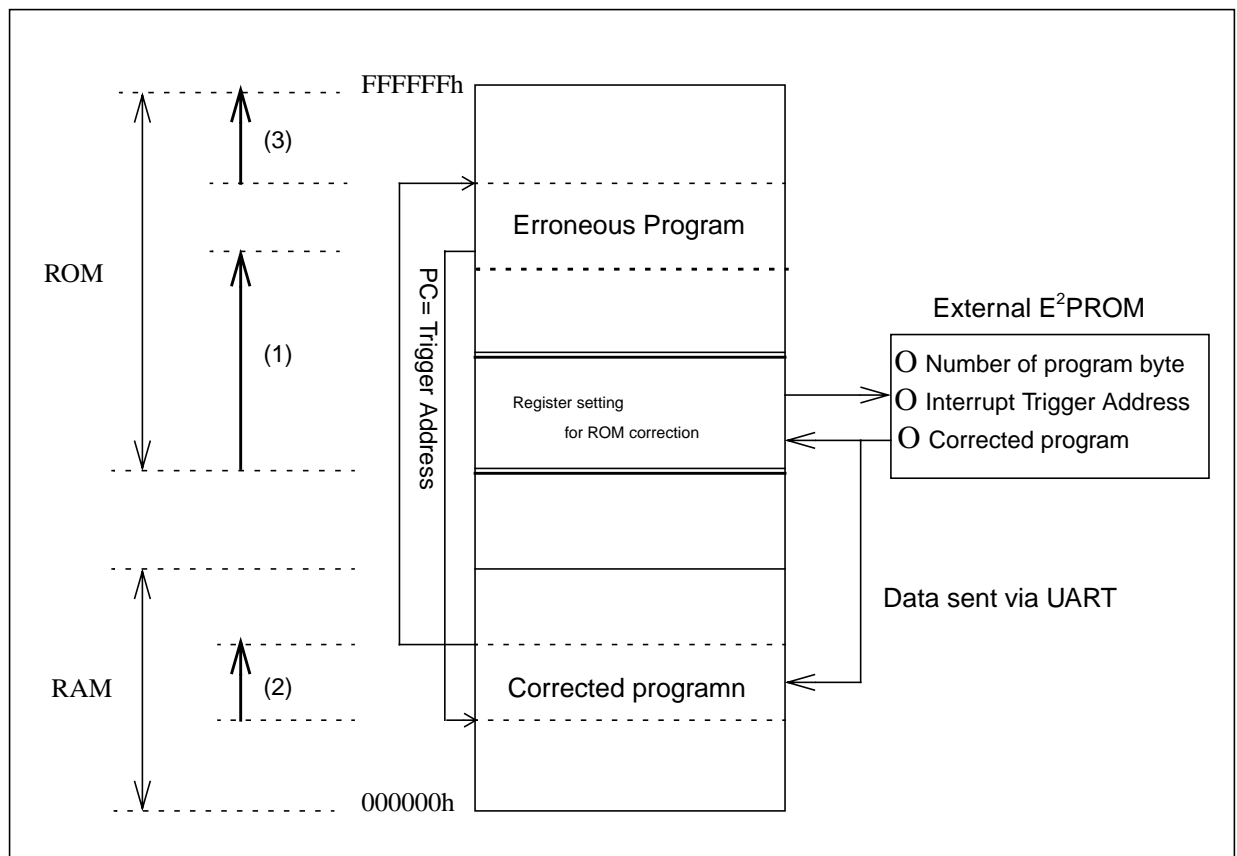


Figure 19.5-2 ROM Correction Processing Example

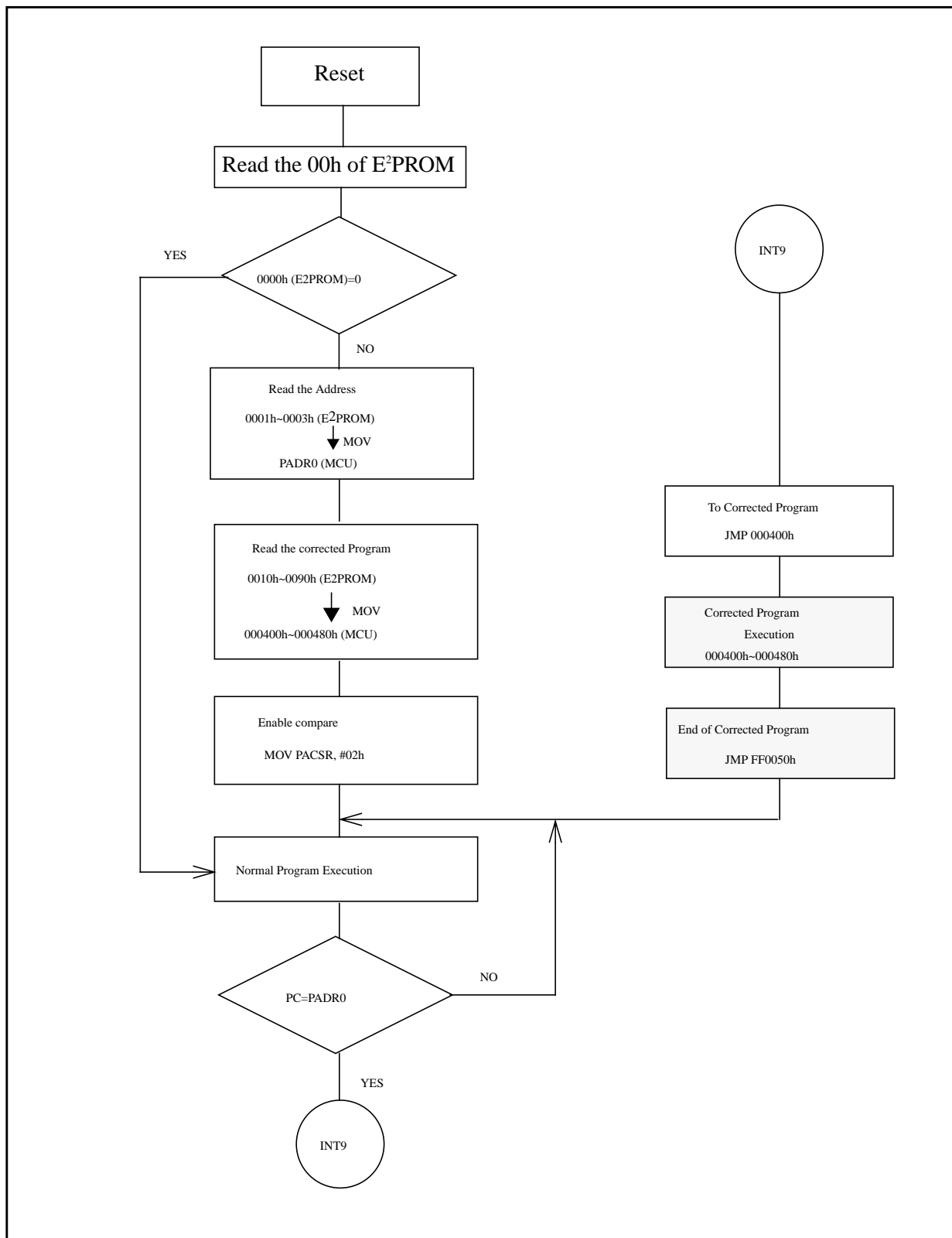


Figure 19.5-3 Sample operation Flowchart

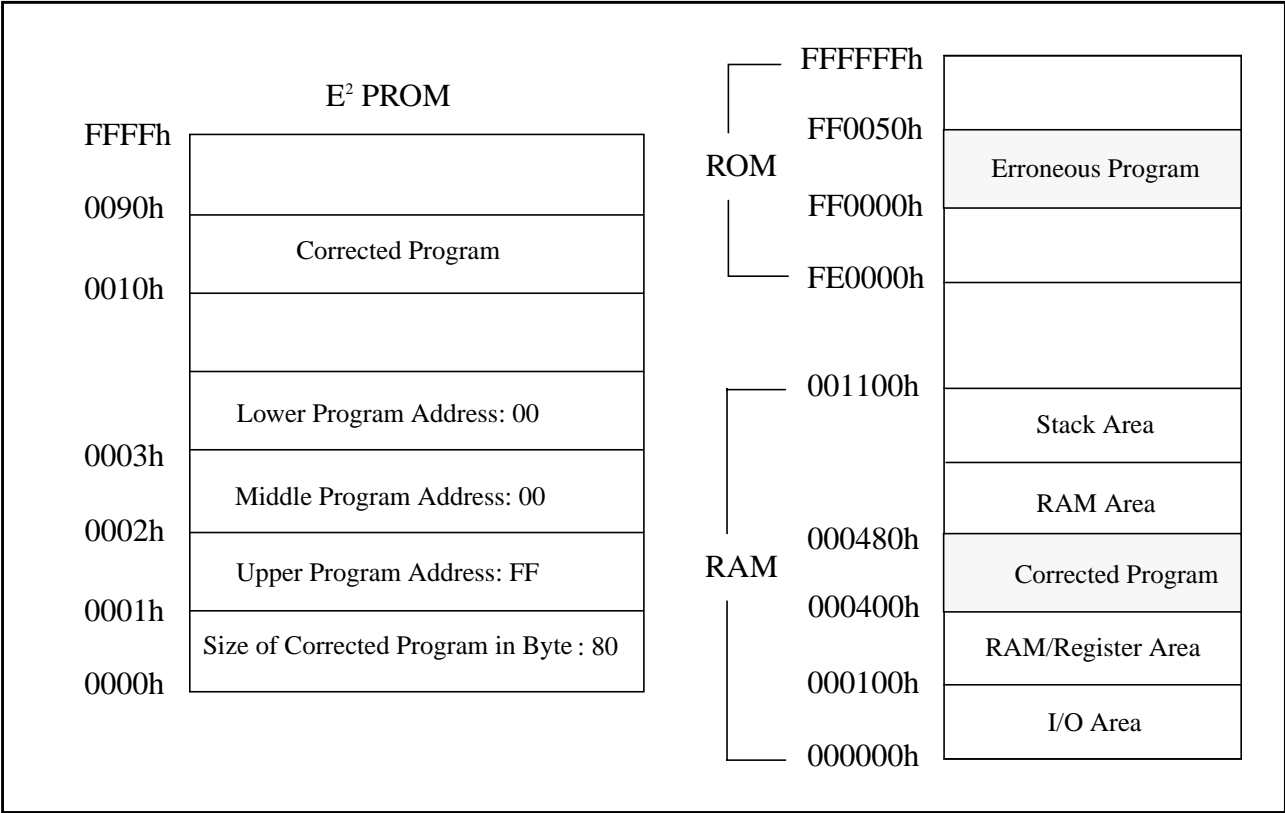


Figure 19.5-4 ROM Correction Processing Flow Diagram

CHAPTER 20 ROM Mirroring Module

The ROM Mirroring module switches whether to mirror the image of the FF bank of the ROM to the 00 bank.

20.1 Register

20.2 Block Diagram

20.3 Register Details

20.1 Register

ROM Mirror Register								⇐ Bit number	
Address : 0006FH	15	14	13	12	11	10	9		8
	—	—	—	—	—	—	—	MI	ROMM
Read/write ⇨	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(W)	
Initial value ⇨	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(1)	

Figure 20.1-1 ROM Mirror Register

20.2 Block Diagram

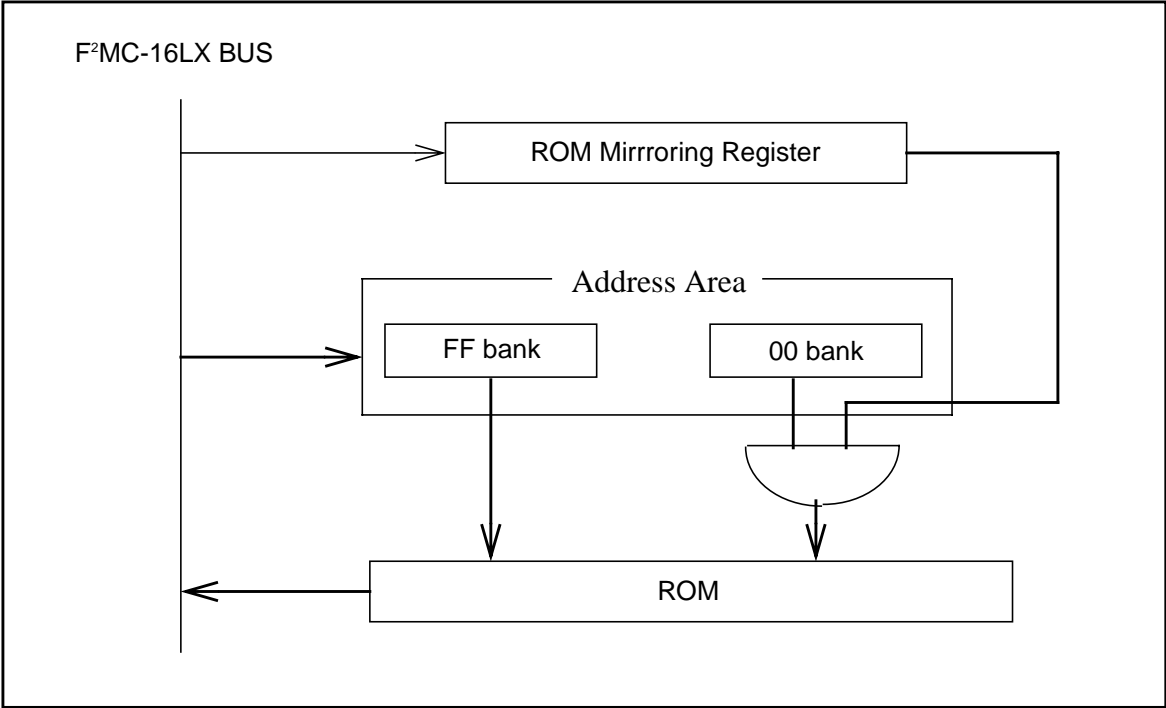


Figure 20.2-1 Block Diagram

20.3 Register Details

(1) ROMM (ROM Mirroring Register)									
	15	14	13	12	11	10	9	8	⇐ Bit number
Address : 0006FH	—	—	—	—	—	—	—	MI	ROMM
Read/write ⇨	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(W)	
Initial value ⇨	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(1)	

Note: Do not access this register when the addresses 004000_H~00FFFF_H are being accessed.

[bit 8]: MI

The image of the ROM data in the FF bank can also be found in the 00 bank when ' 1 ' is written to this bit. However, this memory mapping will not be done when this bit is written to ' 0 '. This bit is write only.

Note: Only FF4000_H~FFFFFF_H is mirrored to 004000_H~00FFFF_H when ROM mirroring function is activated. Therefore, addresses FFF000_H~FF3FFF_H will not be mirrored to 00 bank.

CHAPTER 21 INTERRUPTS

This chapter explains the interrupts and extended intelligent I/O service (EI²OS) in the MB90495 series.

21.1 Interrupts

21.2 Interrupt Causes and Interrupt Vectors

21.3 Interrupt Control Registers and Peripheral Functions

21.4 Hardware Interrupts

21.5 Software Interrupts

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

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

21.8 Exception Processing Interrupt

21.9 Stack Operations for Interrupt Processing

21.10 Sample Programs for Interrupt Processing

21.1 Interrupts

This chapter explains the interrupts and extended intelligent I/O service (EI²OS) in the MB90495 series.

- **Hardware interrupts**
 - **Software interrupts**
 - **Interrupts from extended intelligent I/O service (EI²OS)**
 - **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 (EI²OS)**

The EI²OS 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 EI²OS 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

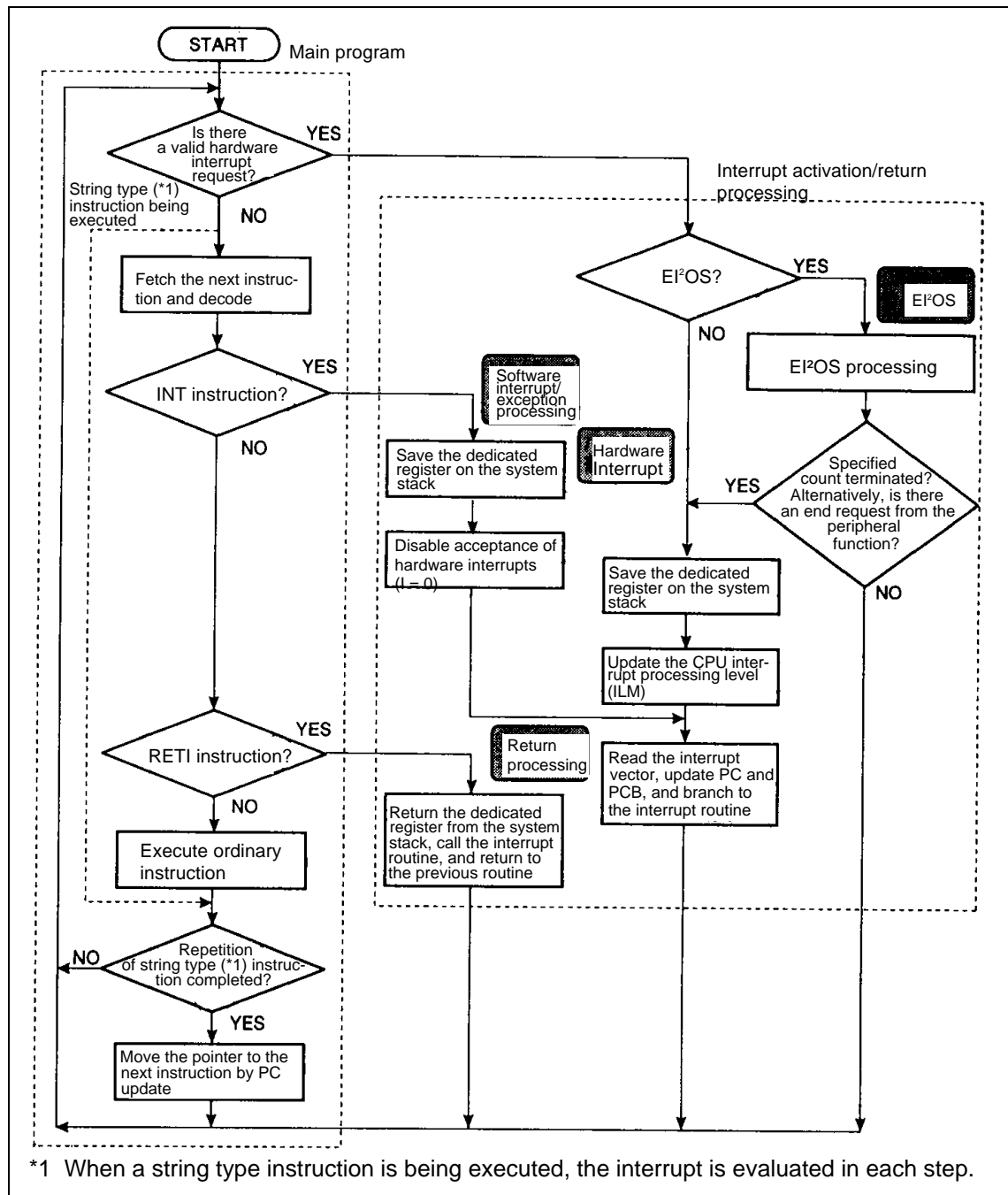


Figure 21.1-1 Overall flow of interrupt operation

21.2 Interrupt Causes and Interrupt Vectors

The F²MC-16LX has functions for handling 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 (FFFC00_H to FFFFFFF_H). Interrupt vectors share the same area with EI²OS, exception processing, hardware, and software interrupts.

Table 21.2 -1 Interrupt vectors

Software interrupt instruction	Vector address L	Vector address M	Vector address H	Mode data	Interrupt No.	Hardware interrupt
INT0	FFFFFC _H	FFFFFD _H	FFFFFE _H	Not used	#0	None
:	:	:	:	:	:	:
INT7	FFFFE0 _H	FFFFE1 _H	FFFFE2 _H	Not used	#7	None
INT8	FFFFDC _H	FFFFDD _H	FFFFDE _H	FFFFDF _H	#8	(RESET vector)
INT9	FFFFD8 _H	FFFFD9 _H	FFFFDA _H	Not used	#9	None
INT10	FFFFD4 _H	FFFFD5 _H	FFFFD6 _H	Not used	#10	<Exception processing>
INT11	FFFFD0 _H	FFFFD1 _H	FFFFD2 _H	Not used	#11	Hardware interrupt #0
INT12	FFFFCC _H	FFFFCD _H	FFFFCE _H	Not used	#12	Hardware interrupt #1
INT13	FFFFC8 _H	FFFFC9 _H	FFFFCA _H	Not used	#13	Hardware interrupt #2
INT14	FFFFC4 _H	FFFFC5 _H	FFFFC6 _H	Not used	#14	Hardware interrupt #3
:	:	:	:	:	:	:
INT254	FFFC04 _H	FFFC05 _H	FFFC06 _H	Not used	#254	None
INT255	FFFC00 _H	FFFC01 _H	FFFC02 _H	Not used	#255	None

Reference

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

■ Interrupt causes and interrupt vectors/interrupt control registers

The relationship between interrupt causes (excluding software interrupts), interrupt vectors, and interrupt control registers is shown below.

Table 21.2 -2 Interrupt causes, interrupt vectors, and interrupt control registers

Interrupt cause	EI ² OS support	Interrupt vector		Interrupt control register		Priority (*2)	
		Number	Address	ICR	Address		
		#08	08H	FFFFDC _H	-	-	High ↑ <

o: Can be used.

x: Cannot be used.

▲: Usable if used with the EI²OS stop function.

△: 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.

*2 - This priority is applied when interrupts of the same level occur simultaneously.

21.3 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 (EI²OS).

■ Interrupt control registers

Table 21.3-1 Interrupt control registers

Address	Register	Abbreviation	Corresponding peripheral function
0000B0H	Interrupt control register 00	ICR00	CAN Controlle
0000B1H	Interrupt control register 01	ICR01	Reserved
0000B2H	Interrupt control register 02	ICR02	External Interrupt INT0/INT1 ; TimeBase Timer
0000B3H	Interrupt control register 03	ICR03	16-bit Reload Timer 0 ; A/D Converter
0000B4H	Interrupt control register 04	ICR04	I/O Timer ; External Interrupt INT2/INT3
0000B5H	Interrupt control register 05	ICR05	PPG 0/1
0000B6H	Interrupt control register 06	ICR06	Input Capture 0 ; External Interrupt INT4/INT5
0000B7H	Interrupt control register 07	ICR07	Input Capture 1 ; PPG2/3
0000B8H	Interrupt control register 08	ICR08	External Interrupt INT6/INT7 ; Watch Timer
0000B9H	Interrupt control register 09	ICR09	Input Capture 2/3
0000BAH	Interrupt control register 10	ICR10	Reserved
0000BBH	Interrupt control register 11	ICR11	Reserved
0000BCH	Interrupt control register 12	ICR12	16-bit Reload Timer 1
0000BDH	Interrupt control register 13	ICR13	UART 1
0000BEH	Interrupt control register 14	ICR14	UART 0
0000BFH	Interrupt control register 15	ICR15	Flash Memory ; Delayed Interrupt

■ 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 (EI²OS) channel
- Display the status of the extended intelligent I/O service (EI²OS)

Some of the functions of the interrupt control registers (ICR) differ during writing and reading, as shown in the following figures

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

21.3 Interrupt Control Registers and Peripheral Functions

21.3.1 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)

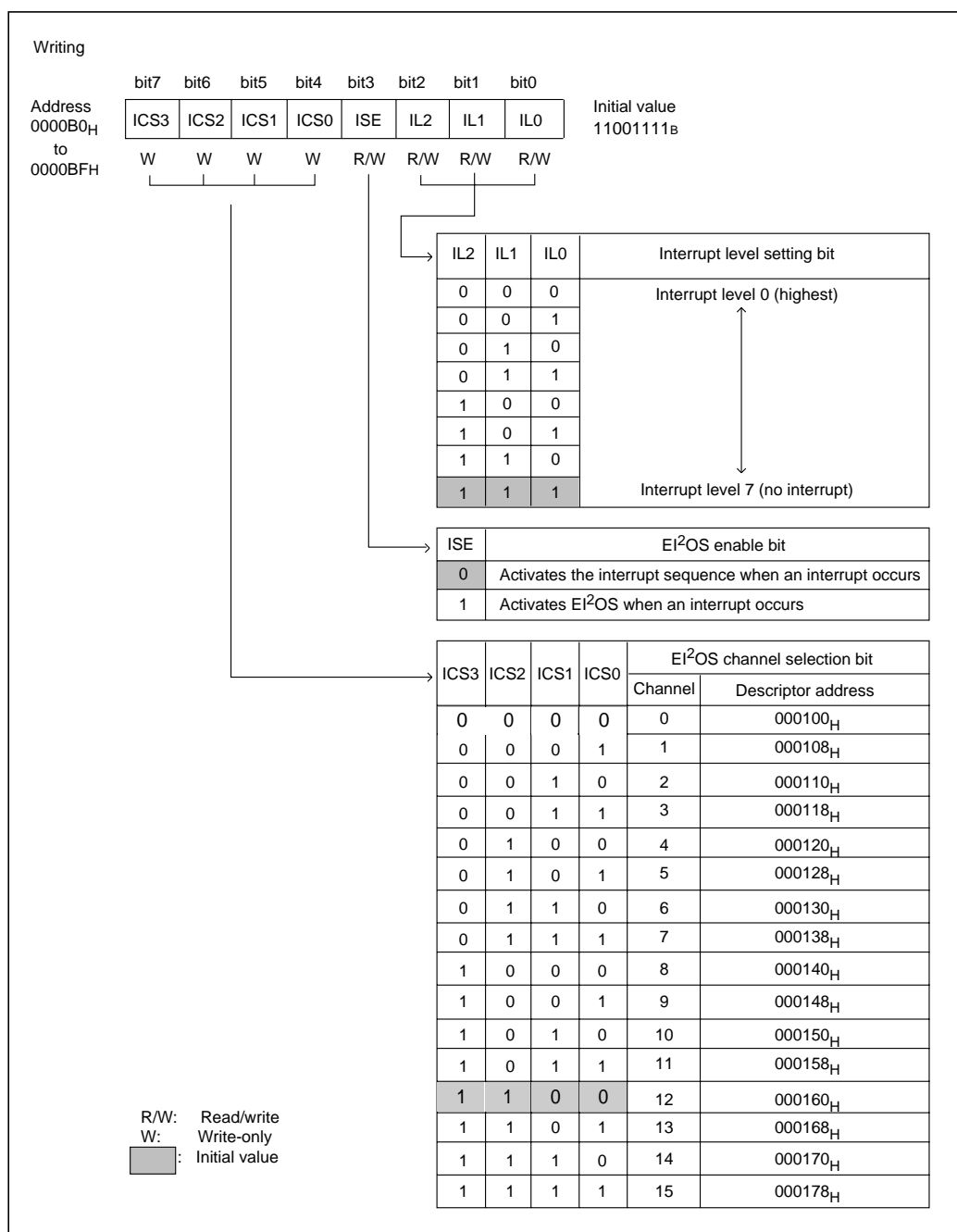


Figure 21.3.1-1 Interrupt control registers (ICR00 to ICR15) during writing

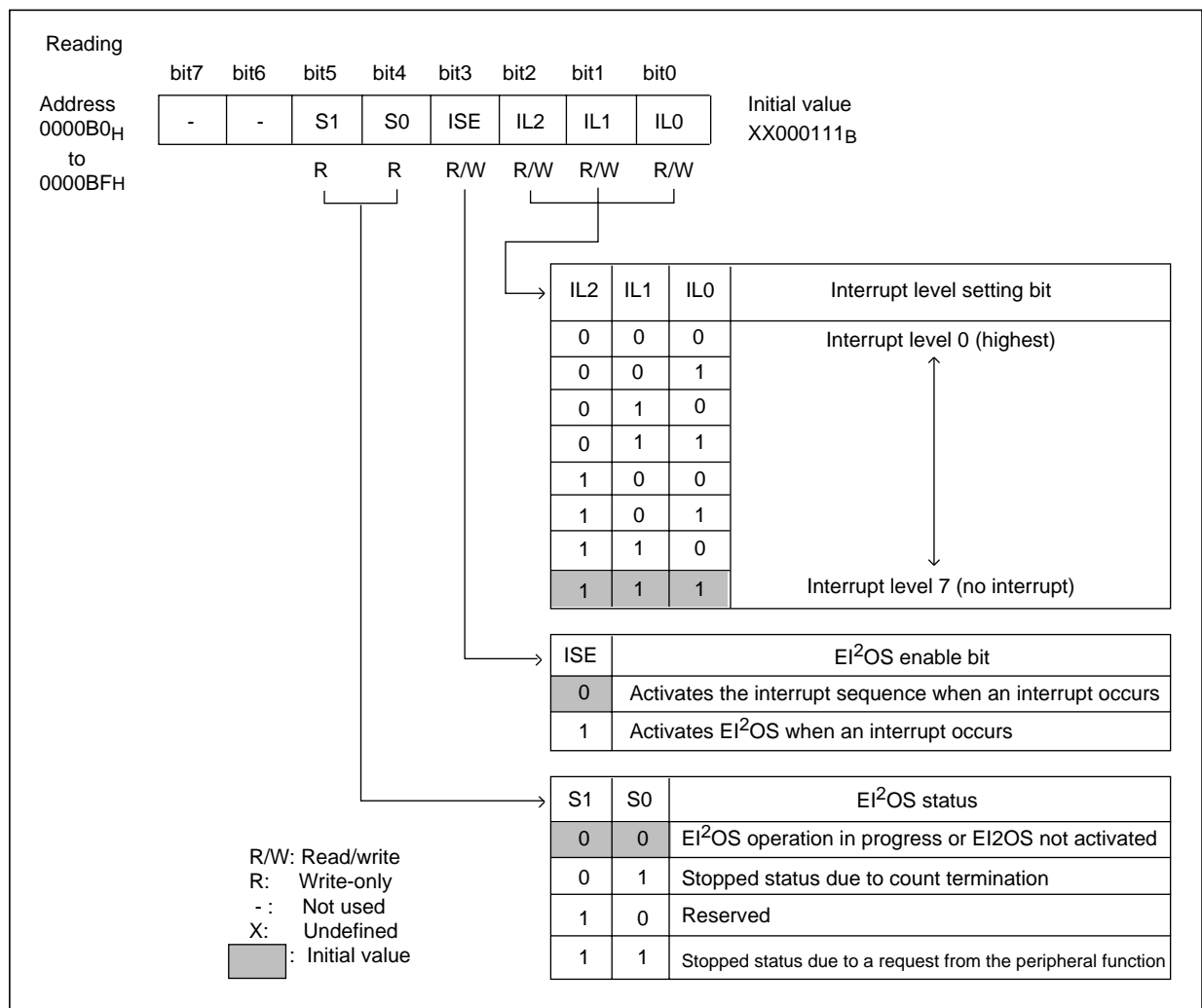


Figure 21.3.1-2 Interrupt control registers (ICR00 to ICR15) during reading

21.3.2 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)

Writing to interrupt control register (ICR)									
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000B0H	ICS3	ICS2	ICS1	ICS0	ISE	IL2	IL1	IL0	11001111 _B
to									
0000BFH	W	W	W	W	W	W	W	W	
Reading of interrupt control register (ICR)									
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000B0H	—	—	S1	S0	ISE	IL2	IL1	IL0	XX001111 _B
to									
0000BFH	—	—	R	R	R	R	R	R	

R: Read-only
 W: Write-only
 -: Not used

Figure 21.3.2-1 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 21.3.2-1 Correspondence between the interrupt level setting bits and interrupt levels

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

● **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 at 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 21.3.2-2 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 21.3.2-3 Relationship between EI²OS status bits and the EI²OS status shows the relationship between the S0 and S1 bits and the EI²OS status

Table 21.3.2-3 Relationship between EI²OS status bits and the EI²OS status

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

21.4 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 (EI²OS) 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 (EI²OS)

EI²OS is an automatic transfer function between memory and I/O. When the specified transfer count has been completed, a hardware interrupt is activated. Multiple EI²OS activation does not occur. During EI²OS processing, all other interrupt requests and EI²OS 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 FFFC00H to FFFFFFFH. These tables are shared by software interrupts.

See Section 21.2 "Interrupt Causes and Interrupt Vectors" for more information about the allocation of interrupt numbers and interrupt vectors.

■ Hardware interrupt structure

Four mechanisms used for hardware interrupts are listed in the table below. These four mechanisms must be included in the program before hardware interrupts can be used.

Table 21.4 -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 FFFFFFF _H in memory	Interrupt vector table	Stores the branch destination address for interrupt processing

These four mechanisms must be included in the program before hardware interrupts can be used.

■ 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 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.

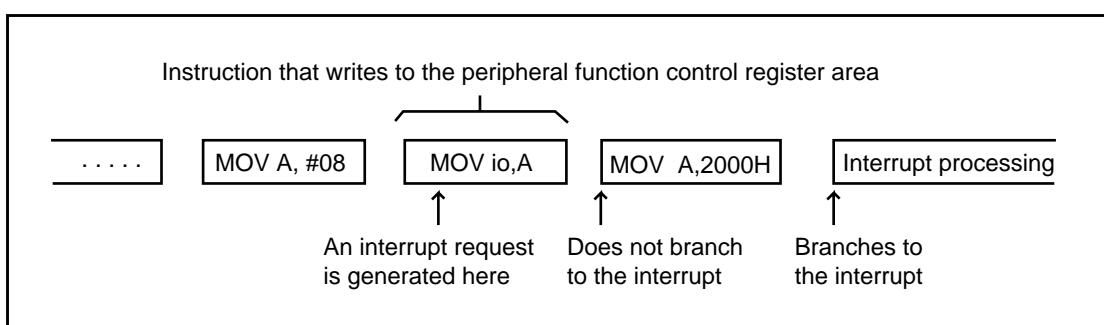


Figure 21.4-1 Hardware interrupt request while writing to the peripheral function control register area

● Hardware interrupt suppression by interrupt suppression instruction

The ten types of hardware interrupt suppression instructions listed below ignore interrupt requests without detecting whether a hardware interrupt request exists.

Table 21.4 -1 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.

21.4.1 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 specific to the peripheral function occurs.

● Interrupt controller operation (interrupt request control)

The interrupt controller compares the interrupt levels (IL) of interrupt requests received at the same time. 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 references the ISE bit in the interrupt control register (ICR). If $ISE = 0$, the CPU continues the execution of interrupt processing. (If $ISE = 1$, EI²OS 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 21.4.1-1 shows hardware interrupt operation from generation of a hardware interrupt to the completion of interrupt processing.

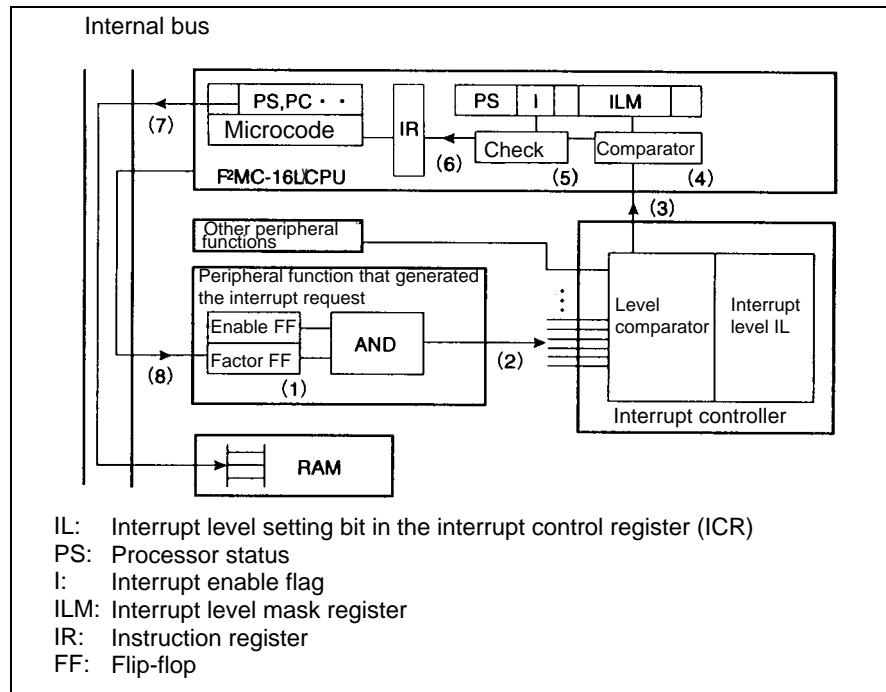


Figure 21.4.1-1 Hardware interrupt operation

- (1) An interrupt cause is generated within the peripheral function.
- (2) The interrupt enable bit of the peripheral function is referenced. If the interrupt is enabled, the interrupt request is output from the peripheral function to the interrupt controller.
- (3) The interrupt controller that receives the interrupt request 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) 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.

21.4.2 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 21.4.2-1 shows the flow of processing for interrupt operation.

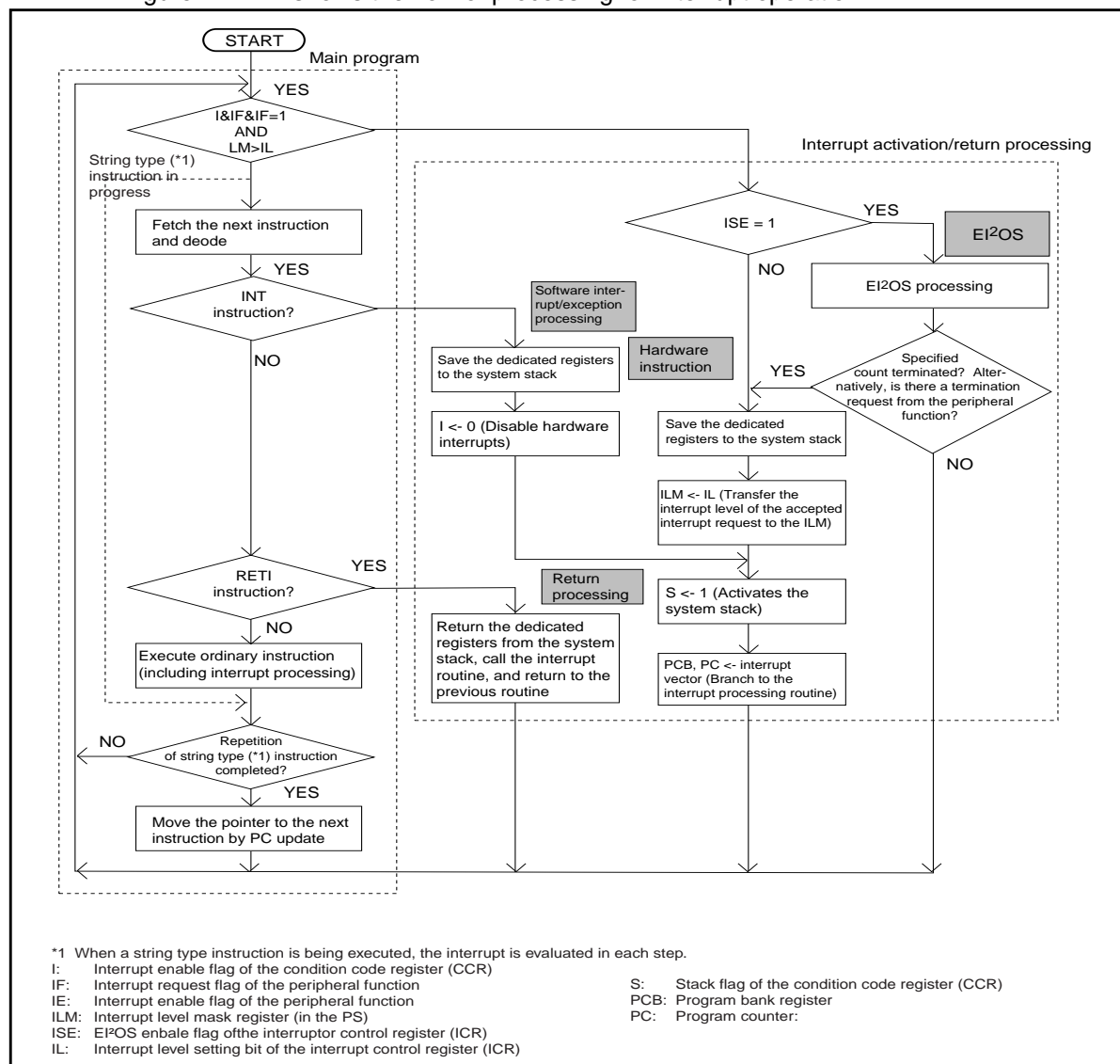


Figure 21.4.2-1 Flow of interrupt processing

21.4.3 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

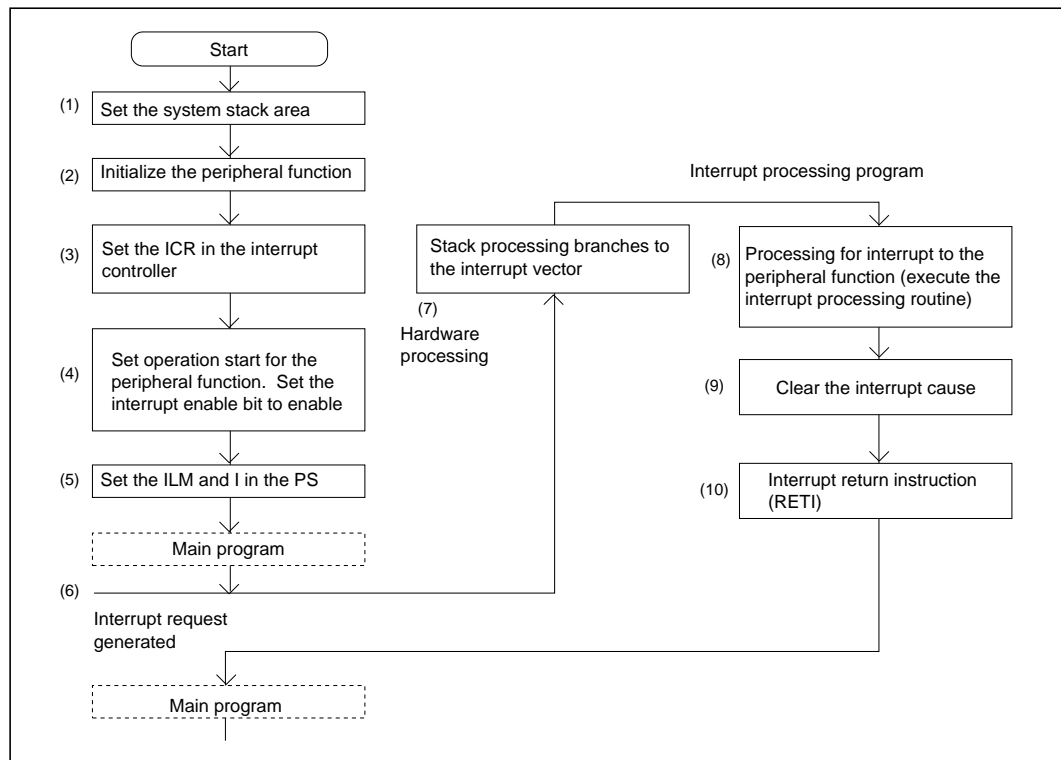


Figure 21.4.3-1 Procedure for using hardware interrupts

- (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 before branching.

21.4.4 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 (EI²OS) cannot be used for the activation of multiple interrupts. During processing of the extended intelligent I/O service (EI²OS), 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 the figure below.

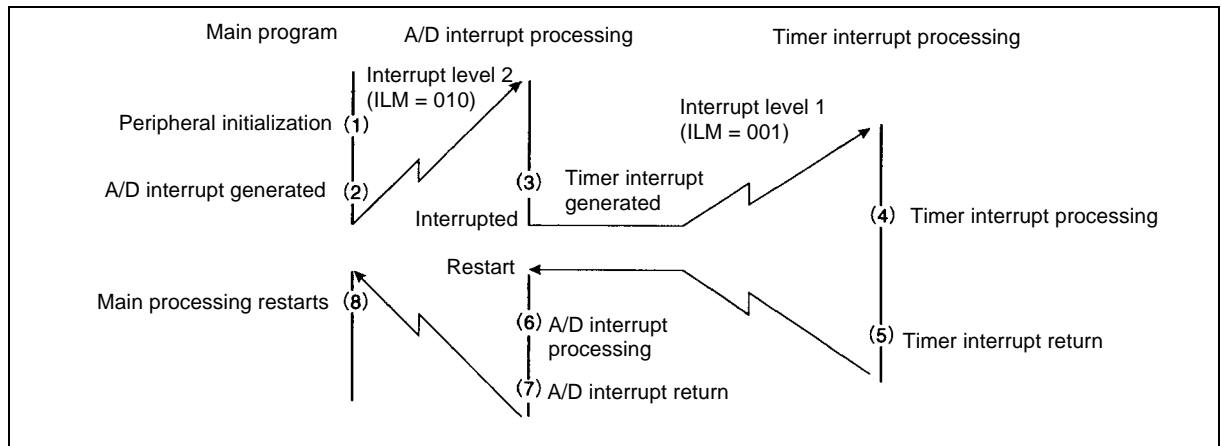


Figure 21.4.4-1 Example of multiple interrupts

1) 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.

2) 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.

21.4.5 Hardware interrupt processing time

From the generation 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 generation 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.

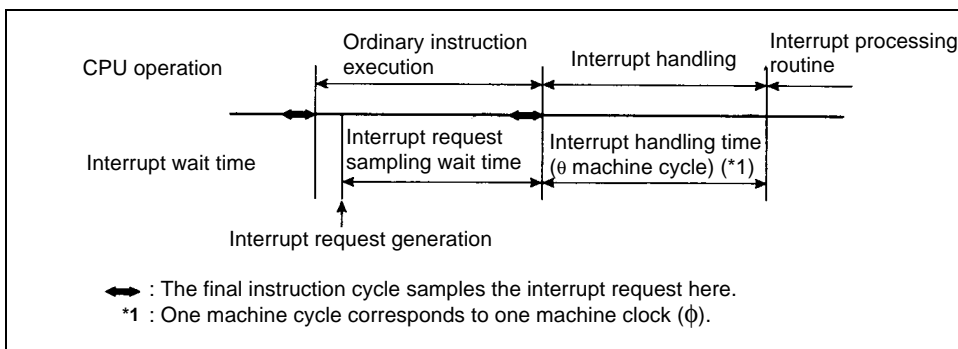


Figure 21.4.5-1 Interrupt processing time

● Interrupt request sampling wait time

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

Whether an interrupt request has been generated 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.

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 (ϕ machine cycle)**

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

When an interrupt is activated: $\theta = 24 + 6 + Z$ machine cycles

When control is returned from an interrupt: $\theta = 11 + 6 + Z$ machine cycles (RETI instruction)

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

Table 21.4-3 Interpolation values (Z) for the interrupt handling time

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

Reference

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

21.5 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.

See Section 21.2 "Interrupt Causes and Interrupt Vectors" 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 to the system stack is restored to the dedicated registers and the processing that was being executed before branching for the interrupt is resumed.

■ Software interrupt operation

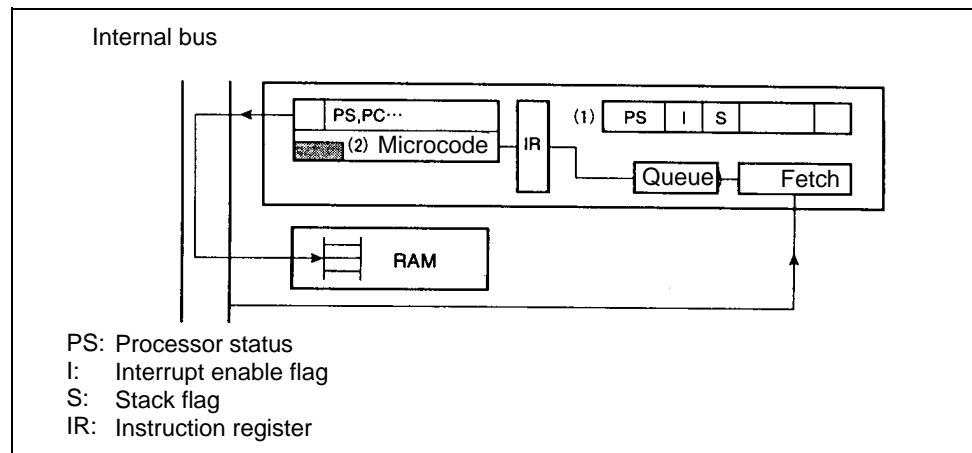


Figure 21.5-1 Software interrupt operation

- (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.

21.6 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 update can be selected for the buffer address.
- Incrementing or no update 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.

Interrupt numbers and interrupt vectors are permanently set for each peripheral. See Section 21.2 "Interrupt Causes and Interrupt Vectors" in 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)**

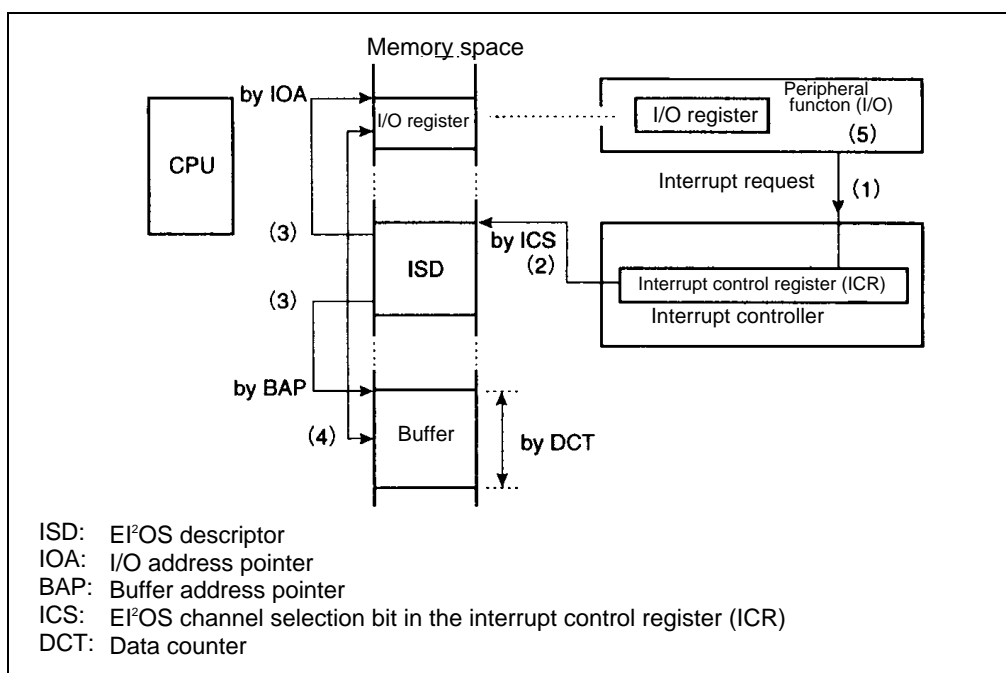


Figure 21.6-1 Extended intelligent I/O service (EI²OS) operation

- (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.

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

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

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

The ISD consists of 8 bytes x 16 channels. Each ISD has the structure shown below. The table shows the correspondence between channel numbers and ISD addresses.

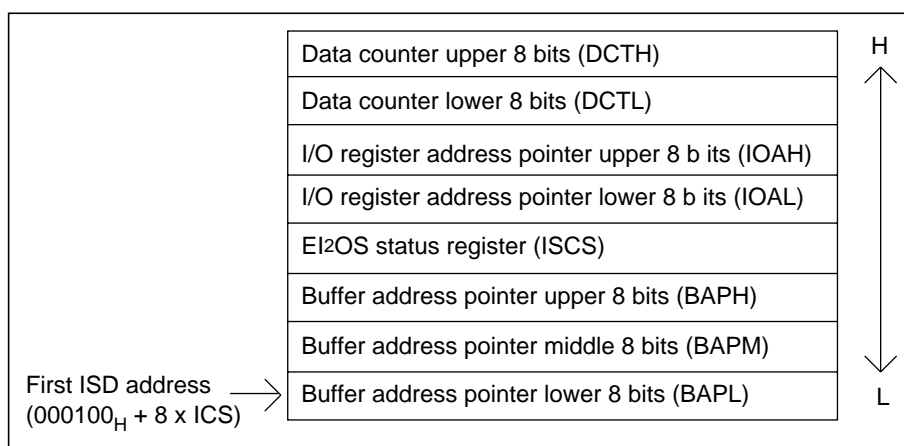


Figure 21.6.1-1 Configuration of EI²OS descriptor (ISD)

Table 21.6.1-1 Correspondence between channel numbers and descriptor addresses

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

21.6.2 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.

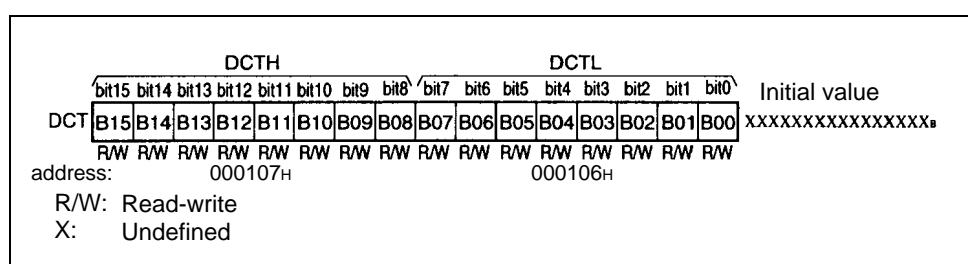


Figure 21.6.2-1 Configuration of DCT

■ 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.

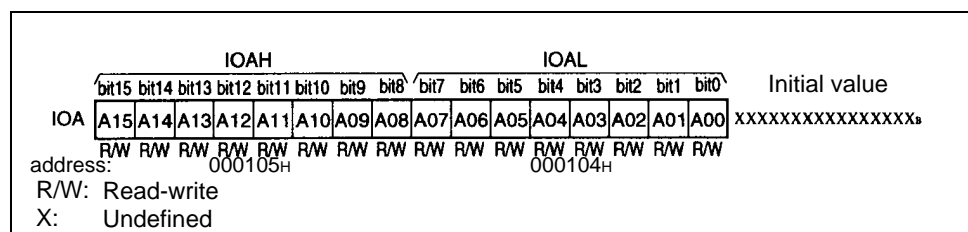


Figure 21.6.2-2 Configuration of I/O register address pointer (IOA)

■ 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.

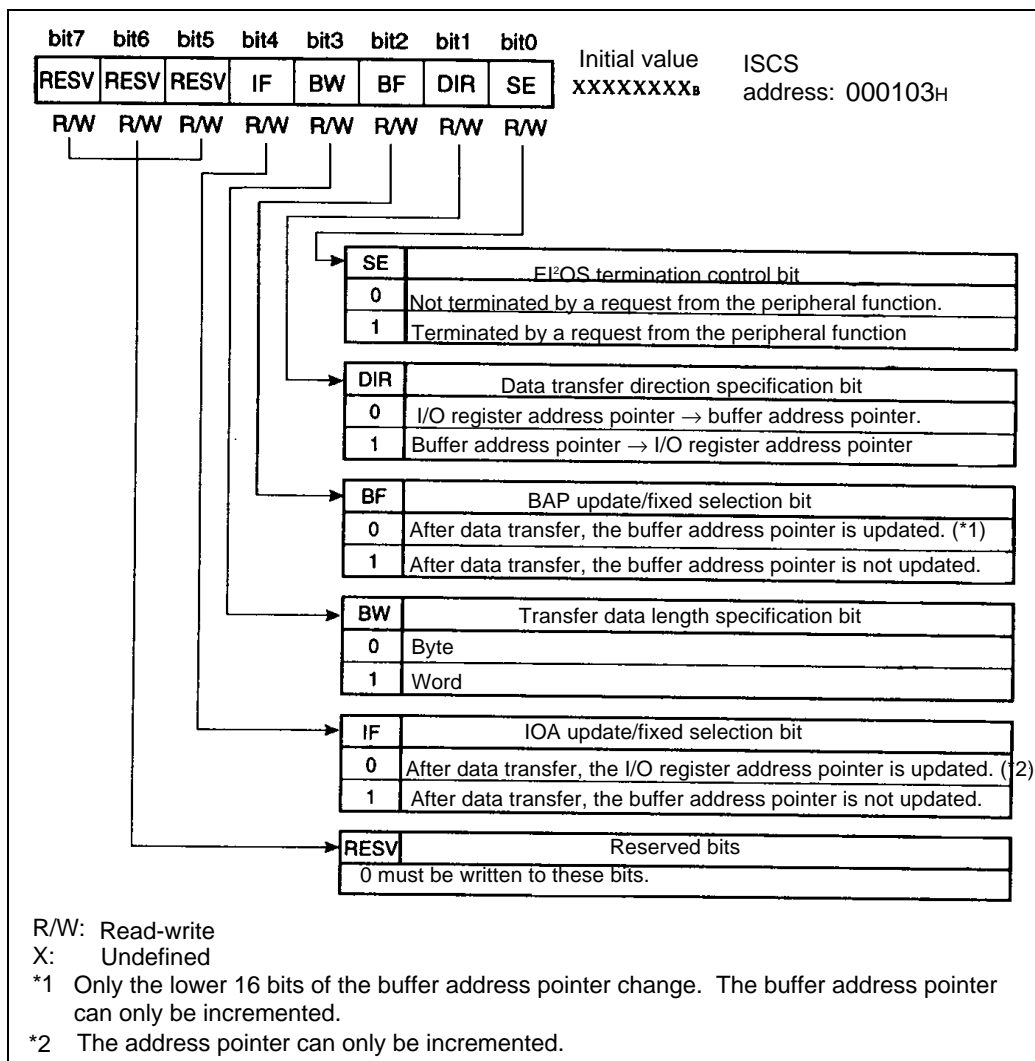


Figure 21.6.2-3 Configuration of EI²OS status register (ISCS)

■ 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.

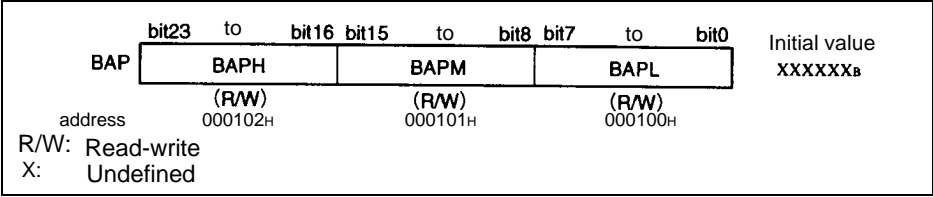


Figure 21.6.2-4 Configuration of buffer address pointer (BAP)

Reference

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

21.7 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)

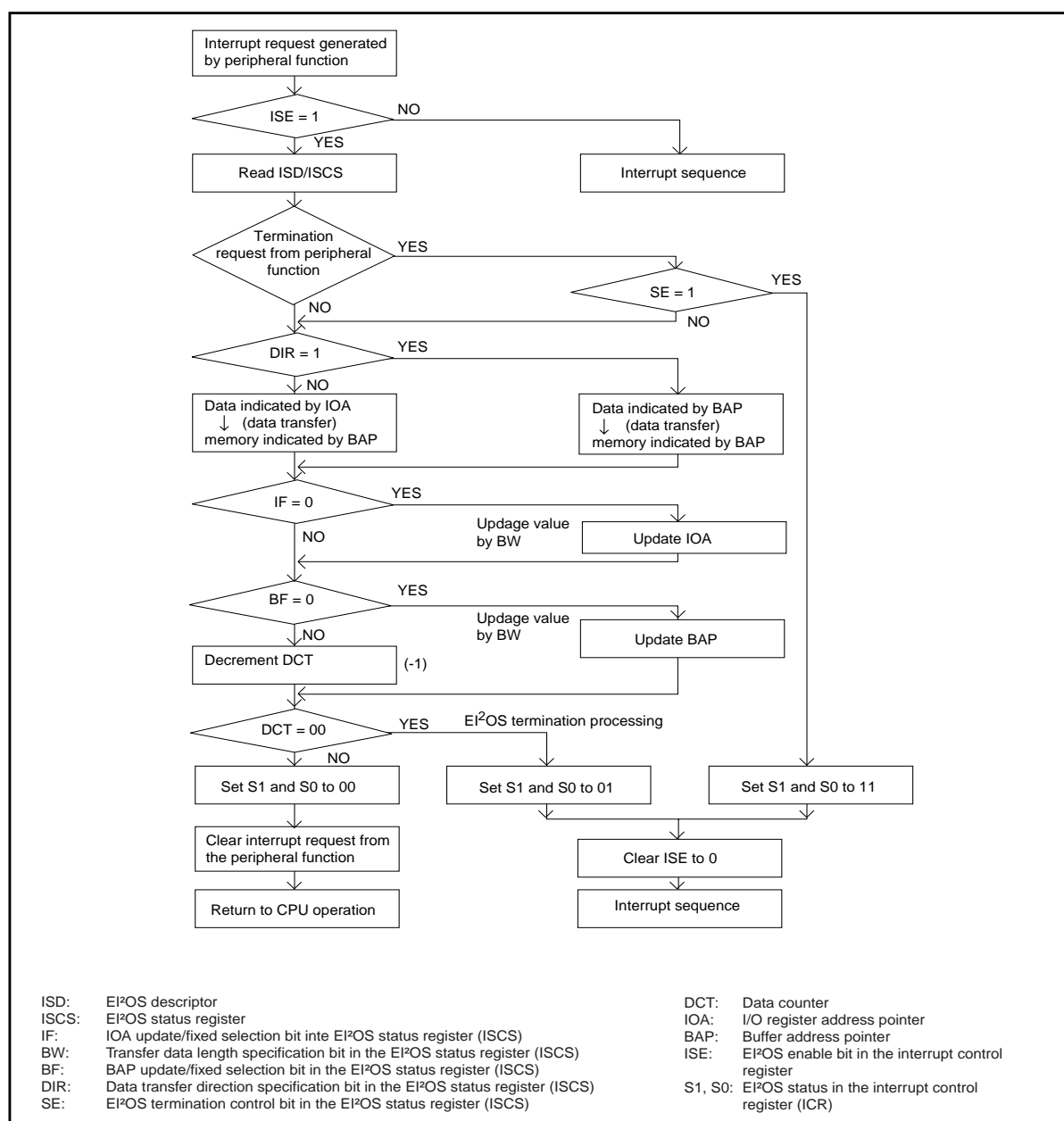


Figure 21.7-1 Flow of extended intelligent I/O service (EI²OS) operation

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

21.7.1 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)

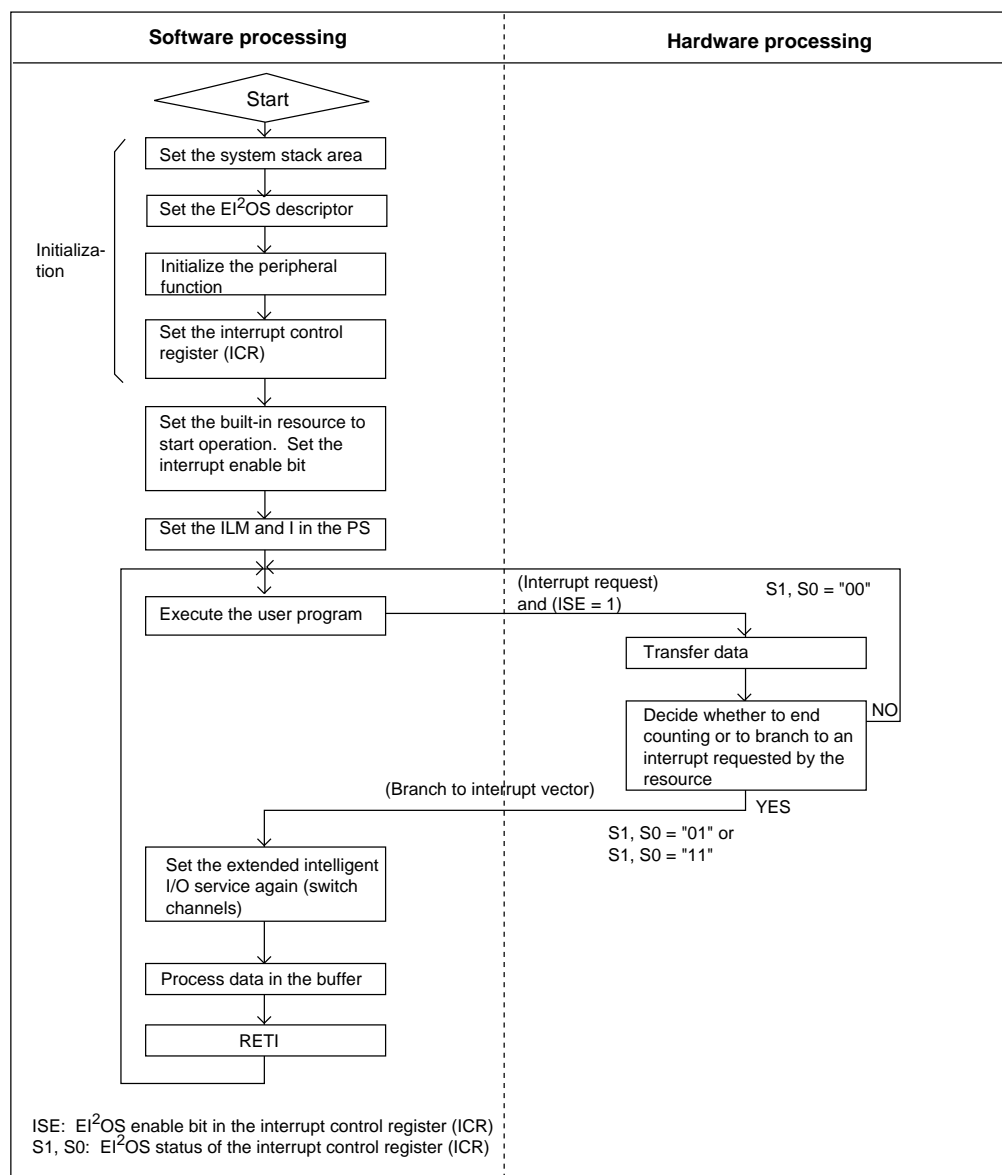


Figure 21.7.1-1 Procedure for using the extended intelligent I/O service (EI²OS)

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

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

- EI²OS status register (ISCS) setting
- Address (area) pointed to by the I/O register address pointer (IOA)
- Address (area) pointed to by the buffer address pointer (BAP)
- External data bus length for external access
- Transfer data length

Because the hardware interrupt is activated when data transfer by EI²OS terminates, the interrupt handling time is added.

■ 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 based on the EI²OS status register (ISCS) setting.

Table 21.7.2-1 Extended intelligent I/O service execution time

EI ² OS 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 below, interpolation is necessary depending on the EI²OS execution condition.

Table 21.7.2-2 Data transfer interpolation 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 EI²OS terminates, the interrupt handling time is added. The EI²OS processing time when counting terminates is calculated with the following formula:

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

Interrupt handling time

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

Table 21.7 -3 Interrupt Handling times

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

- **For termination by a termination request from the peripheral function (I/O)**

When data transfer by EI²OS 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 EI²OS processing time is calculated with the following formula. Z in the formula indicates the interpolation value for the interrupt handling time EI²OS processing time for termination before completion = 36 + 6 x Z Machine cycle

Reference

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

21.8 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.

21.9 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)

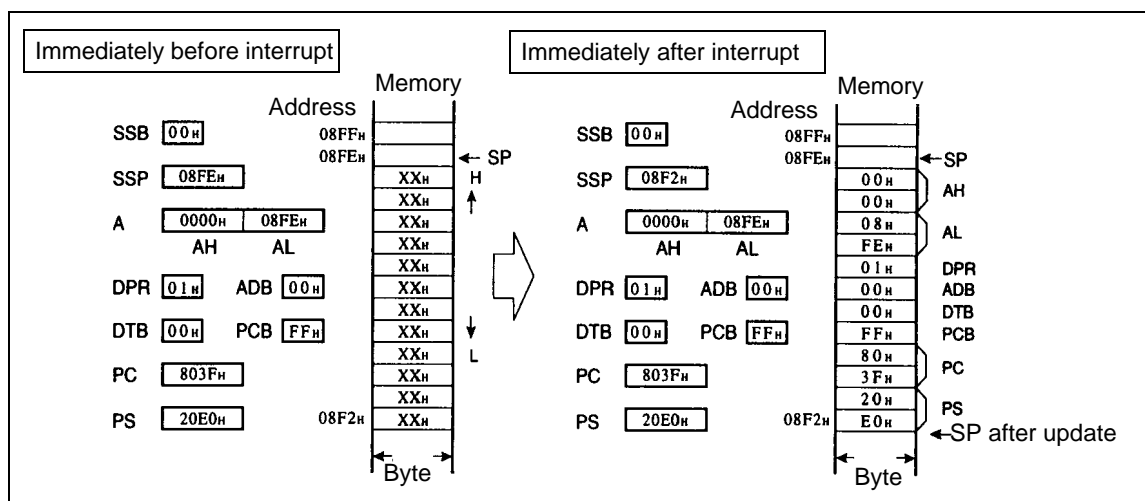


Figure 21.9-1 Stack operations at the start of interrupt processing

■ 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.

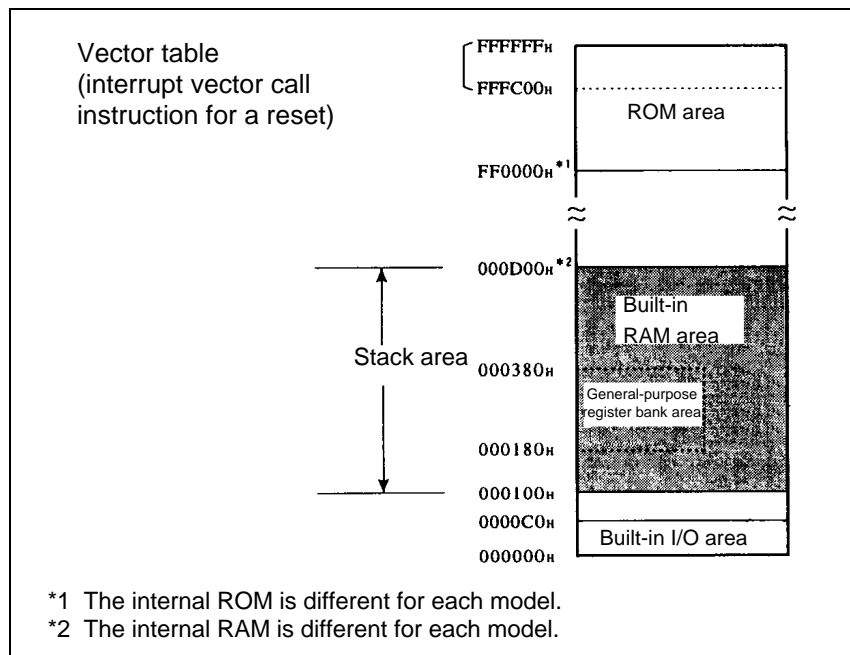


Figure 21.9-2 Stack area

<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.

21.10 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
EIRR       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 -----
CODE        CSEG
START:
    MOV      RP,#0                ;General-purpose registers use the first bank.
    MOV      ILM, #07H           ;Sets ILM in PS to level 7.
    MOV      A, #STACK_T         ;Sets system stack.
    MOV      SSB, A
    MOVW     A, #STACK_T         ;Sets stack pointer, then
    MOVW     SP, A               ;Sets SSP because S flag = 1.
    MOV      DDR1, #00000000B     ;Sets P10/INT0 pin to input.
    OR       CCR, #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:EIRR, #00H         ;Clears INT0 interrupt cause.
    MOV      I:EIRR, #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

CODEENDS

;-----Vector setting-----

VECT
    CSEG        ABS=0FFH
    ORG         0FFD0H    ;Sets vector for interrupt #11 (0BH).
    DSL         ED_INT1
    ORG         0FFDCH    ;Sets reset vector.
    DSL         START
    DB          00H       ;Sets single-chip mode.

VECT        ENDS
END         START

```

■ 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
EIRR     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    EIRR:0     ;Definition of external interrupt request flag bit

STACK    SSEG
RW        100
STACK_T   RW        1
STACK     ENDS

```

;-----Main program-----

```

CODE        CSEG
START:
    AND      CCR, #0BFH    ;Clears the I flag of the CCR in the PS and

```

```

MOV      RP, #00                ;prohibits interrupts.
MOV      A, #STACK_T            ;Sets the register bank pointer.
MOV      SSB, A                 ;Sets the system stack.
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 Æ 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:EIRR, #00H           ;Clears the INT0 interrupt cause.
MOV      I:ENIR, #01H           ;Enables INT0 interrupts.
MOV      ILM, #07H              ;Sets the ILM in the PS to level 7.
OR       CCR, #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
CODEENDS

;-----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

```


CHAPTER 22 SETTING A MODE

This chapter describes the operating modes and memory access modes supported by the MB90495 series.

22.1 Setting a Mode

22.2 Mode Pins (MD2 to MD0)

22.3 Mode Data

22.4 External Memory Access (Bus Pin Control Circuits)

22.1 Setting a Mode

The F²MC-16LX supports the modes for access method, and access areas. 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, and access areas, classified as shown in Figure 22.1-1 in this module.

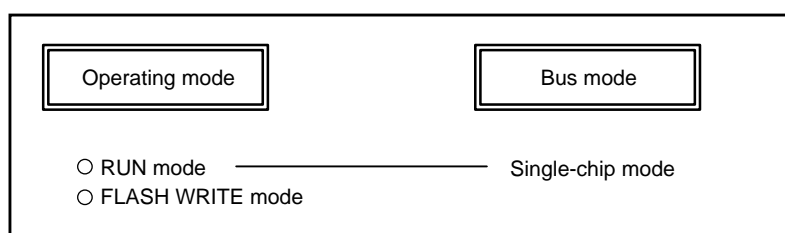


Figure 22.1-1 Mode classification

■ 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.

■ 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 MODE" for details.

22.2 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 22.2-1 shows the mode pin settings.

Table 22.2-1 Mode pin settings

MD2	MD2	MD0	Mode name	Reset vector access area	External data bus width	Remarks
0	0	0	External bus		8 bit	multiplex
0	0	1	External bus		16 bit	multiplex
0	1	0	Setting not allowed			
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	On-board programming mode	-	-	-
1	1	1	FLASH programming mode	-	-	-

MD2 to MD0: Connect the pins to Vss for 0 and to Vcc for 1. A pull-up or a pull-down resistor in the range 2k - 10kΩ may be used.

22.3 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 22.3-1 shows the mode data configuration.

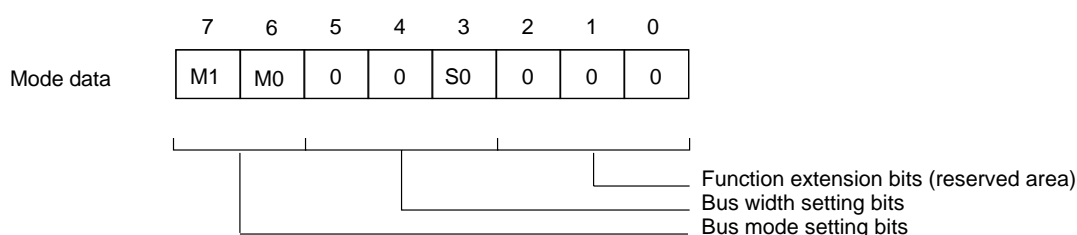


Figure 22.3-1 Mode data configuration

■ Bus mode setting bits

The bus mode setting bits specify operating mode after a reset sequence. Table 22.3-1 lists the relationship between the bits and functions.

Table 22.3-1 Bus mode setting bits and functions

M1	M0	Function	Remarks
0	0	Single-chip mode	
0	1	External bus mode	Internal ROM
1	0	External bus mode	External ROM
1	1		(Setting not allowed)

Table 22.3-2 Bus mode setting bits and functions

S0	Function	Remarks
0	External bus 8 bit	
1	External bus 16 bit	

Figure 22.3-3 shows the correspondence between access areas and physical addresses in single-chip mode.

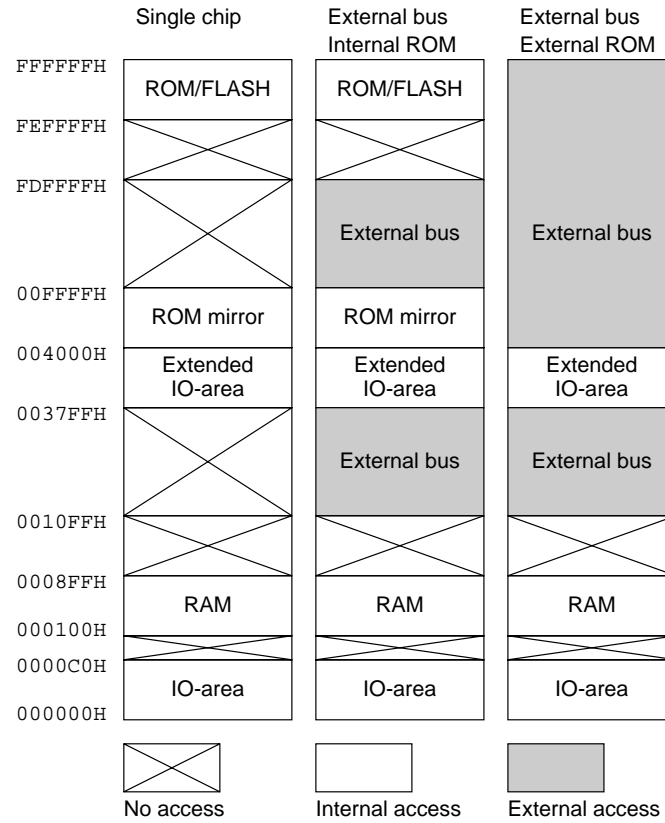


Figure 22.3-3 Correspondence between access areas and physical addresses in different modes

■ Relationship between mode pins and mode data

Table 22.3-2 lists the relationship between mode pins and mode data.

Table 22.3-2 Relationship between mode pins and mode data

Mode	MD2	MD1	MD0	M1	M0	S0
Single-chip	0	1	1	0	0	-
External bus, internal ROM 8-bit bus	0	0	0	0	1	0
External bus, internal ROM 16-bit bus	0	0	1	0	1	1
External bus, external ROM 8-bit bus	0	0	0	1	0	0
External bus, external ROM 16-bit bus	0	0	1	1	0	1

22.4 External Memory Access (Bus Pin Control Circuits)

Access to memory and peripheral devices outside the MB90495 device is enabled by the use of the following address/data/control signals.

Table 22.4-1 External bus pin signal relationship

Signal	Pin	Description
CLK	P37	Machine cycle clock (KBP) output pin.
RDY	P36	External ready input pin.
WRHX	P33	Data bus upper 8-bit write signal.
WRLX	P32	Data bus lower 8-bit write signal.
RDX	P31	Read signal.
ALE	P30	Address latch enable signal.
AD00 - AD15	P00 - P17	Address/Data bus.
A16 - A23	P20 - P27	Address bus.

External bus pin control circuits are used to control external bus pins to allow external expansion of the CPU address/data bus.

Note: The pins for external memory access can either use for external memory access or as general purpose I/O pins or resource I/O pins.

Note: The address area 003800_H to 003FFF_H is not accessible as external memory, because it is used as extended I/O-area.

22.4.1 Register List

Automatic ready function select register		15	14	13	12	11	10	9	8	⇐ Bit no.
Address : 0000A5H		IOR1	IOR0	HMR1	HMR0	—	—	LMR1	LMR0	ARSR
Read/write ⇒		(W)	(W)	(W)	(W)	(—)	(—)	(W)	(W)	
Default value⇒		(0)	(0)	(1)	(1)	(—)	(—)	(0)	(0)	

External address output control register		7	6	5	4	3	2	1	0	⇐ Bit no.
Address : 0000A6H		E23	E22	E21	E20	E19	E18	E17	E16	HACR
Read/write ⇒		(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	
Default value⇒		(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

Bus control signal select register		15	14	13	12	11	10	9	8	⇐ Bit no.
Address : 0000A7H		CKE	RYE	HDE	IOBS	HMBS	WRE	LMBS	—	ECSR
Read/write ⇒		(W)	(W)	(W)	(W)	(W)	(W)	(W)	(—)	
Default value⇒		(0)	(0)	(0)	(0)	(0)	(0)	(0)	(—)	

Figure 22.4.1-1 Registerlist for extaral Memory Access

22.4.2 Block Diagram

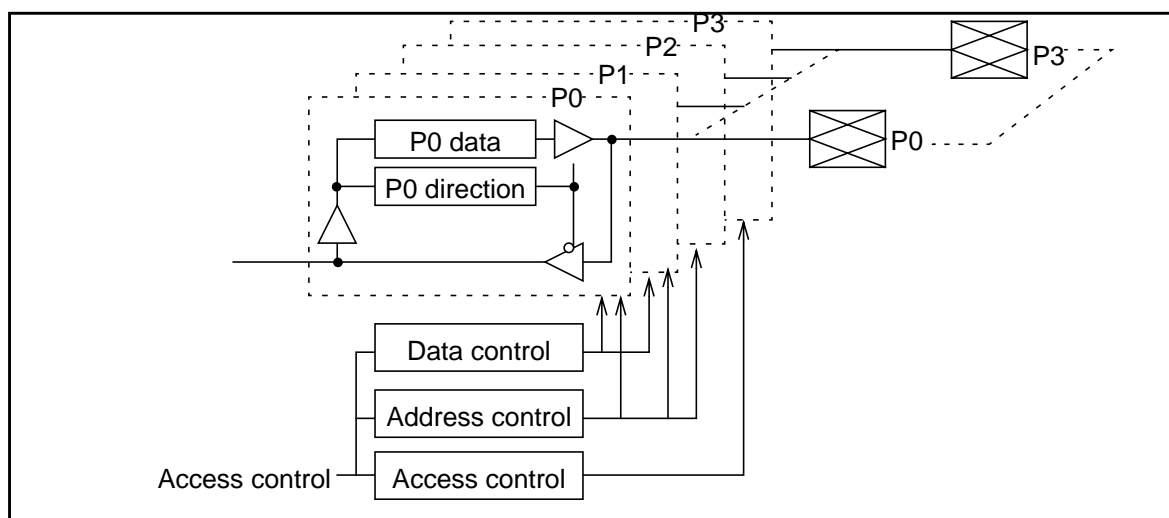


Figure 22.4.2-1 External Bus Controller

22.4.3 Detailed Register Settings

(1) Automatic ready function select register

Automatic ready function select register	15	14	13	12	11	10	9	8	← Bit no.
Address : 0000A5H	IOR1	IOR0	HMR1	HMR0	–	–	LMR1	LMR0	ARSR
Read/write ⇒	(W)	(W)	(W)	(W)	(–)	(–)	(W)	(W)	
Default value⇒	(0)	(0)	(1)	(1)	(–)	(–)	(0)	(0)	

[bit 15, 14] IOR1, IOR0

These bits determine the automatic wait function for external access to the area 0000C0H to 0000FFH. The two bits are used in combination to make the following settings.

IOR1	IOR0	Setting
0	0	Automatic wait prohibited [default value]
0	1	1 cycle automatic wait inserted for external access
1	0	2 cycle automatic wait inserted for external access
1	1	3 cycle automatic wait inserted for external access

[bit 13, 12] HMR1, HMR0

These bits determine the automatic wait function for external access to the area 800000H to FFFFFFFH. The two bits are used in combination to make the following settings.

HMR1	HMR0	Setting
0	0	Automatic wait prohibited
0	1	1 cycle automatic wait inserted for external access
1	0	2 cycle automatic wait inserted for external access
1	1	3 cycle automatic wait inserted for external access [default value]

[bit 9, 8] LMR1, LMR0

These bits determine the automatic wait function for external access to the area 001100H to 7FFFFFFH. (002100H - 7FFFFFFH for MB90V495) The two bits are used in combination to make the following settings.

LMR1	LMR0	Setting	[default value]
0	0	Automatic wait prohibited	
0	1	1 cycle automatic wait inserted for external access	
1	0	2 cycle automatic wait inserted for external access	
1	1	3 cycle automatic wait inserted for external access	

(2) External address output control register

External address output control register	7	6	5	4	3	2	1	0	← Bit no.
Address : 0000A6H	E23	E22	E21	E20	E19	E18	E17	E16	HACR
Read/write ⇒	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	
Default value⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

This register controls address (A23-16) output to external circuits. The eight bits correspond to address bits A32-16, and control the address output pins as follows.

0	The corresponding pin functions as address output Axx. [default value]
1	The corresponding pin functions as I/O port Pxx.

This register cannot be accessed when the device is in single chip mode. In single chip mode all pins function as I/O ports regardless of the value of this register.

All bits in this register are write-only, and have a read value of '1.'

(3) Bus control signal selection register

Bus control signal select register	15	14	13	12	11	10	9	8	← Bit no.
Address : 0000A7H	CKE	RYE	HDE	IOBS	HMBS	WRE	LMBS	—	ECSR
Read/write ⇒	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(—)	
Default value⇒	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(—)	

This register is used to set bus operation control functions in external bus mode.

This register cannot be accessed when the device is in single chip mode. In single chip mode all pins function as I/O ports regardless of the settings in this register.

All bits in this register are write-only, and have a read value of '1.'

[bit 15] CKE

This bit controls the external clock signal pin (CLK) output as follows

0	Pin operates as I/O port (P37) (clock output prohibited) [default value]
1	Clock signal (CLK) output enabled

[bit 14] RYE

This bit controls the external ready (RDY) signal pin input as follows.

0	Operates as I/O port (P36) (external RDY input prohibited) [default value]
1	External ready (RDY) input enabled

[bit 13] HDE

This bit enables input/output of signals related to the hold function. The value controls the hold request

input signal (HRQ) and hold acknowledge output signal (HAKX) as follows.

0	Operates as I/O port (P35, P34) (hold function I/O prohibited)	[default value]
1	Hold request input (HRQ)/hold acknowledge output (HAKX) output enabled	

[bit 12] IOBS

In 16-bit external data bus mode, this bit controls the bus size for external access to the area 0000C0H-0000FFH as follows.

0	16-bit bus size access	[default value]
1	8-bit bus size access	

[bit 11] HMBS

In 16-bit external data bus mode, this bit controls the bus size for external access to the area 80000H-FFFFFFH as follows.

0	16-bit bus size access	[default value in external vector mode 1]
1	8-bit bus size access	[default value in external vector mode 0]

[bit 10] WRE

This bit controls the external write signal output (both WRHX and WRLX pins in 16-bit bus mode, or WRLX pin only in 8-bit bus mode) as follows:

0	Operates as I/O port (P33, P32) (write signal output prohibited)	[default value]
1	Write strobe signal output enabled (WRHX/WRLX or WRLX only)	

In external data bus 8-bit mode, P33 operates as an I/O port regardless of the value of this bit.

[bit 9] LMBS

In 16-bit external data bus mode, this bit controls the bus size for external access to the area 001100H-7FFFFFFH. (002100H - 7FFFFFFF for MB90V495)

0	16-bit bus size access	[default value]
1	8-bit bus size access	

Note: In 16-bit bus mode, when the WRE bit is set to enable the WRHX and WRLX function, P33 and P32 should be set to input mode. (DDR3 register bit3 and bit2 should be set to '0'.)

In 8-bit bus mode, when the WRE bit is set to enable the WRX function, P32 should be set to input mode. (DDR register bit2 should be set to '0'.)

Also, when the RYE and HDE bits are used to enable RDY and HRQ signal input, those ports are enabled for I/O port functions. For this reason the DDR3 register bits corresponding to those ports must be set to '0' (input mode).

22.4.4 Operating Description

The MB90495 provides separate mode settings for access method and access area.

(1) External Memory Access Control Signal

Access to external memory requires 3 cycles unless the ready function is used.

In external 16-bit bus mode, 8-bit bus access is a function used to read/write to 8-bit peripheral chips when the MB90495 is connected to an external bus on which 8-bit peripheral chips are mixed with 16-bit peripheral chips. The 8-bit bus access function is executed on the lower 8 bits of the data bus, therefore the corresponding 8-bit peripheral chips should also be connected to the lower 8 bits of the data bus.

In external 16-bit bus mode, the decision between 16-bit bus access and 8-bit bus access is made by the HMBS/LMBS/IOBS bits in the EPCR register.

Note that address output and ALE signal assertion alone, without RDX/WRLX/WRHX assertion may not actually result in bus operation. Users should ensure that peripheral chip access is not executed for ALE signals alone.

■ External 8-bit Bus Mode

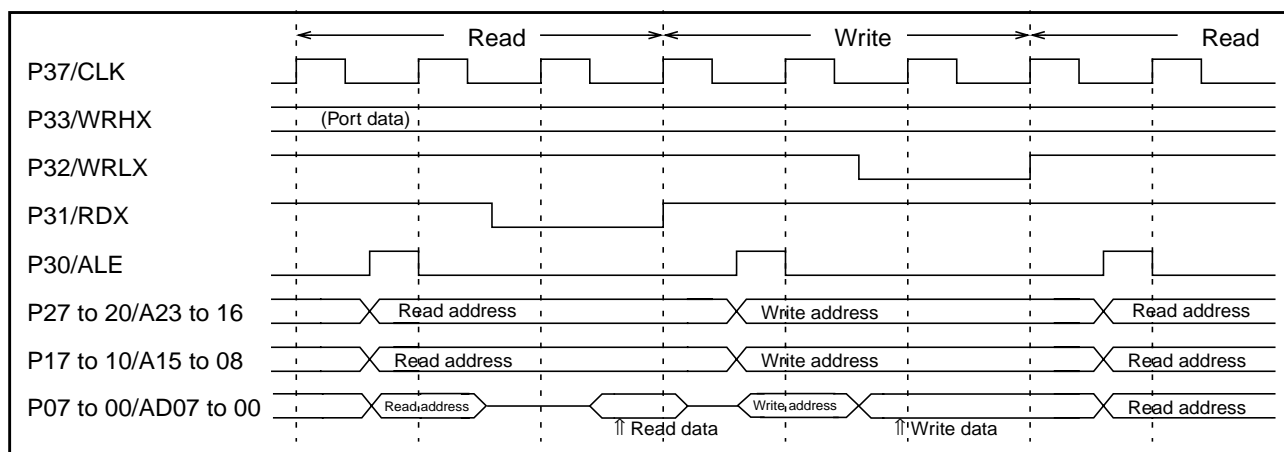


Figure 22.4.4-1 External Memory Access Timing Chart

■ External 16-bit Bus Mode

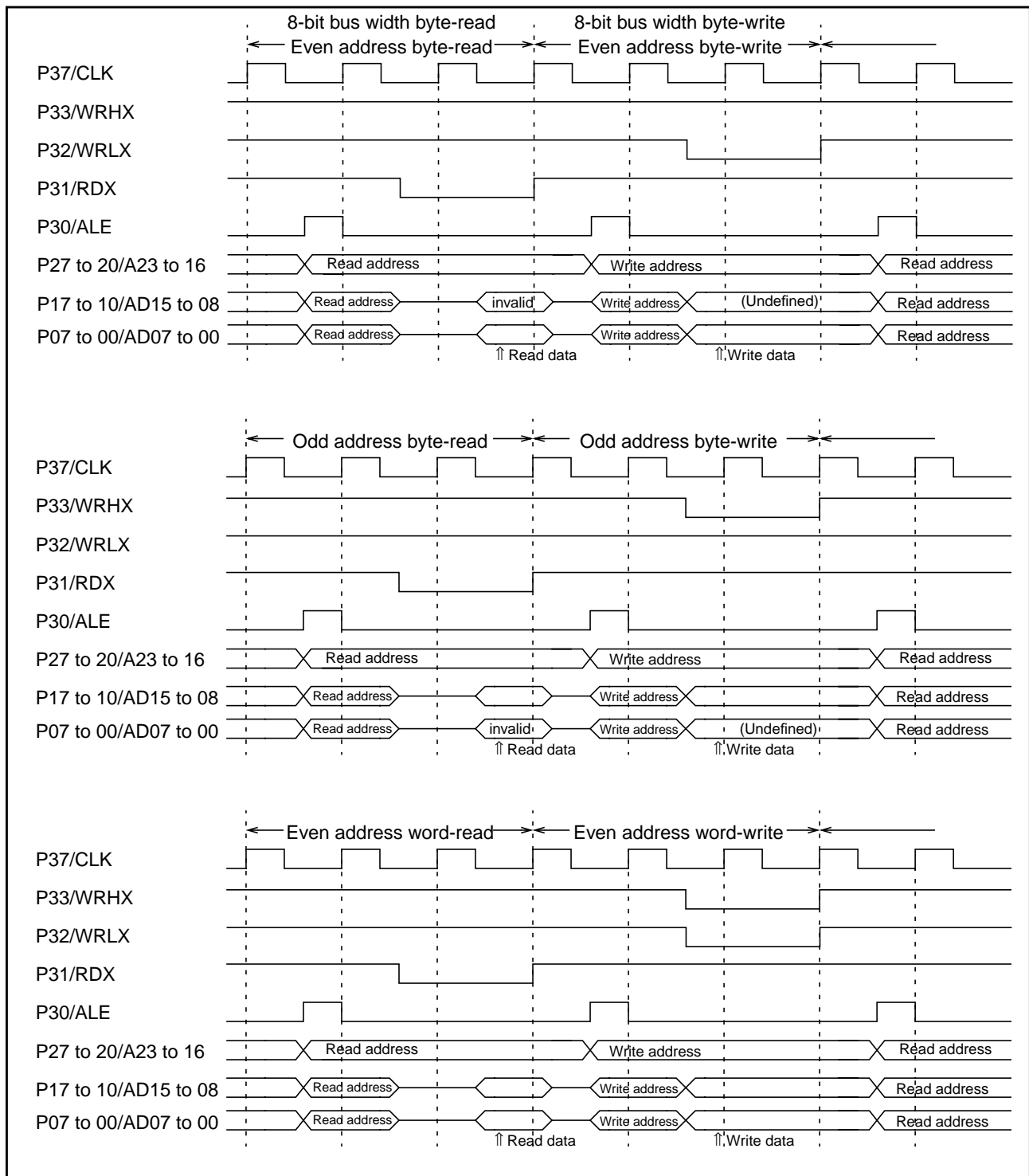


Figure 22.4.4-2 External Memory Access Timing Chart

* External circuits should be set to always read in word length.

The P36/RDY pin and automatic ready function select register (ARSR) can be set to enable access to low speed memory or peripheral circuits.

(2) Ready Function

The P36/RDY pin and automatic ready function select register (ARSR) can be set to enable access to low speed memory or peripheral circuits.

The RYE bit in the bus control signal select register (EPCR) can be set to '1' to extend the access cycle by creating a wait cycle while the P36/RDY signal is at 'L' level during access to external circuits.

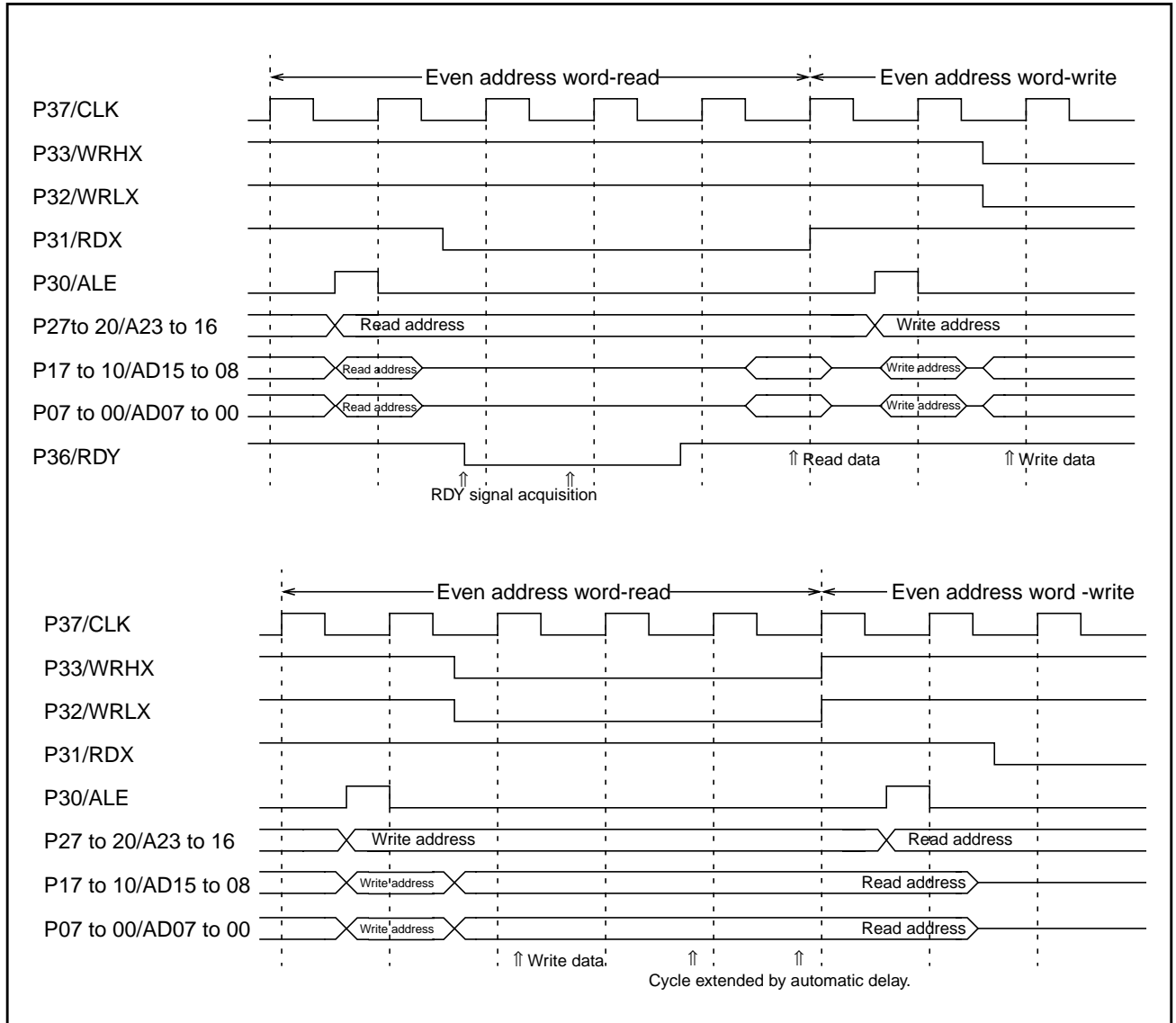


Figure 22.4.4-3 Ready Function Timing Chart

The MB90495 device also provides two types of auto-ready function for external memory access. The auto ready function operates only for access to external memory allocated to lower addresses 002000H to 7FFFFFFH and for access to external memory allocated to upper addresses 800000H to FFFFFFFH and extends the access cycle by automatically inserting 1 to 3 cycles without the use of external circuits. This function is activated by setting the LMR1/LMR0 bits (lower address external area) and HMR1/HMR0 bits (upper address external area) in the ARSR register.

In addition, the MB90495 device provides an I/O auto-ready function that is independent of the memory auto-ready function. When the IOR1/IOR0 bits in the ARSR register are set to '0' this function can extend the access cycle for access to external memory addresses between 0000C0H and 0000FFH, by inserting a wait cycle of 1 to 3 machine cycles without the use of external circuits.

When either the external memory auto-ready function or the external I/O auto-ready function is used with the RYE bit in the EPCR register set to '1' the wait cycle can be extended by input of the 'L' level signal at the P36/RDY pin after the end of the wait cycle introduced by the corresponding auto-ready function as described above.

(3) Hold Function

When the HDE bit in the EPCR register is set to 1, the use of the external address hold function is enabled for the P34/HRQ, P35/HAKX pins. When an 'H' level signal is input at the P34/HRQ pin, hold status will be applied after completion of the CPU instruction (in case of string instructions, after completion of processing of '1' element data), an 'L' level signal will be output from the P35/HAKX pin, and the following pins will be placed in high impedance status.

- Address output P23/A19 to P20/A16
- Data I/O P17/AD15 to P00/AD00
- Bus control signal P30/ALE, P31/RDX, P32/WRLX, P33/WRHX

This enables use of the external bus by external circuits.

When an 'L' level signal is input at the P34/HRQ pin the P35/HAKX output changes to 'H' level, the pins revert to external signal pin status and bus operation is resumed.

In stop status, no hold request inputs are accepted.

■ Hold Function Timing (16-Bit External Bus Mode)

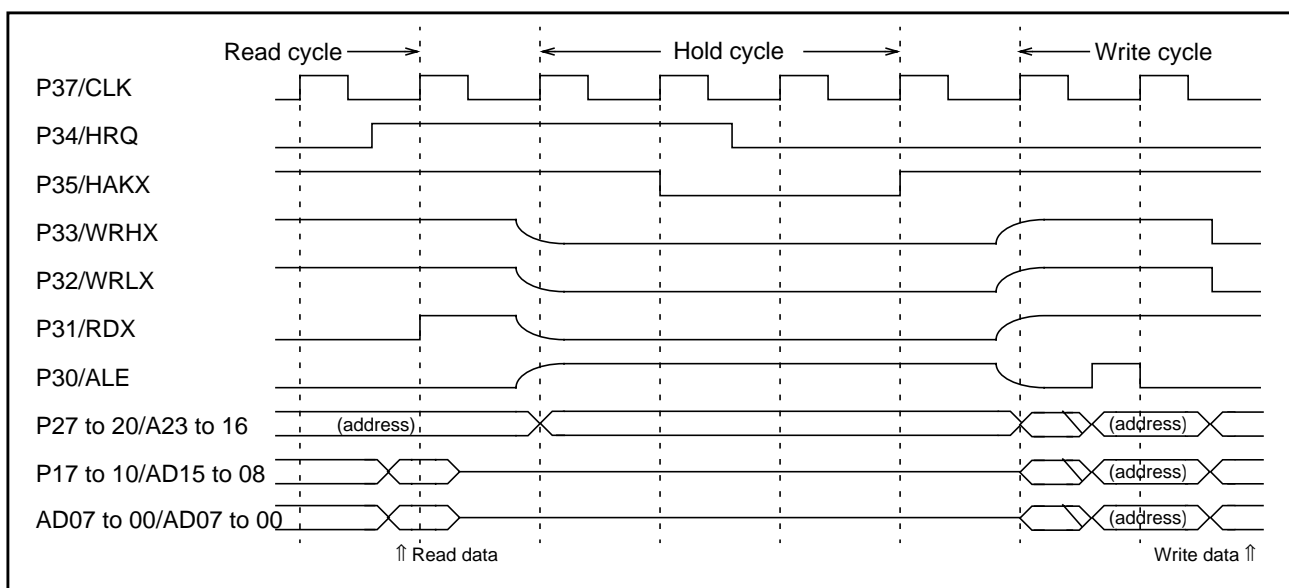


Figure 22.4.4-4 Hold Timing

CHAPTER 23 FLASH MEMORY

The Flash Memory is available only with MB90F497.

- 23.1 Outline of Flash Memory
- 23.2 Block Diagrams of Flash Memory
- 23.3 Write/Erase Modes
- 23.4 Control Status Register (FMCS)
- 23.5 Read/Write Access
- 23.6 Automatic Write/Erase Algorithm
- 23.7 Sector Protect Operation
- 23.8 Connection to Flash Memory Writer
- 23.9 Notes for Use of Flash Memory
- 23.10 Timing Diagrams in Flash Memory Mode
- 23.11 AC Characteristics in Flash Memory Mode

23.1 Outline of Flash Memory

The Flash Memory consists of a flash memory unit derived from the MB29F400TA and a flash memory interface circuit.

Flash Memory:

- 64 Kword \times 8 / 32 Kword \times 16 bit
- Uses automatic program algorithm (Embedded Algorithm™)
- Erase pause/restart function
- Detects completion of writing/erasing using data polling or toggle bit functions
- Detects completion of writing/erasing by RY/ $\overline{\text{BY}}$ pin
- Detects completion of writing/erasing by CPU interrupts
- Compatible with JEDEC standard commands
- Performs minimum of 10,000 write/erase operations
- Data retention time: 10 years
- Sector erase function (any combination of sectors)
- Sector protect function
- Temporary sector protect cancellation function
- Allows flash memory interface circuit to write to/erase flash memory both under control of external pin by writer and under control of internal bus by CPU.

Embedded Algorithm™ is a registered trademark of Advanced Micro Devices, Inc.

23.2 Block Diagrams of Flash Memory

Figure 23.2-1 shows the block diagram of the flash memory unit. Figure 23.2-2 shows the entire block diagram of the Flash Memory with the flash memory interface circuit. Figure 23.2-3 shows the sector configuration of the Flash Memory.

(1) Block diagram of Flash Memory

Figure 23.2-1 shows the block diagram of the flash memory unit, which has almost the same configuration as the MBM29F400TA, except the memory capacity.

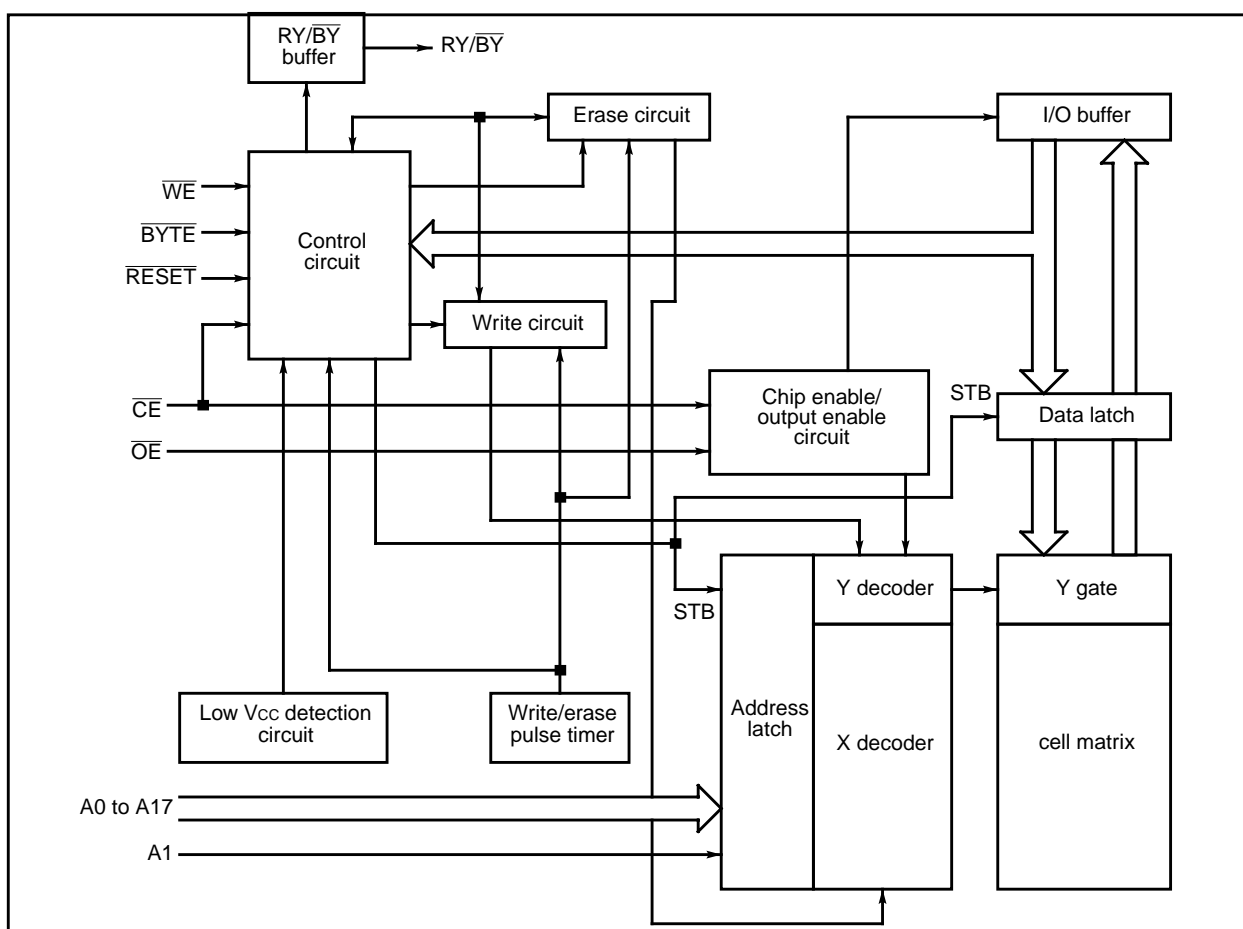


Figure 23.2-1 Block Diagram of Flash Memory

(2) Entire block diagram of Flash Memory

Figure 23.2-2 shows the entire block diagram of the Flash Memory with the flash memory interface circuit.

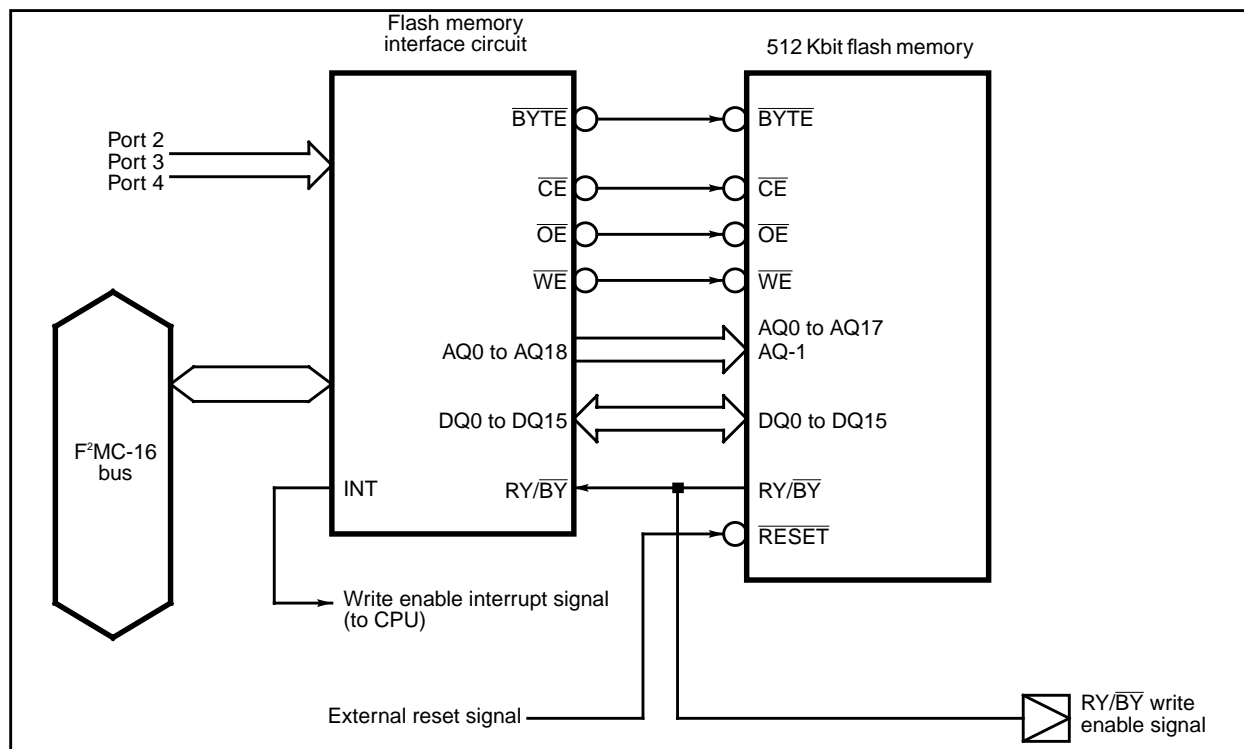


Figure 23.2-2 Entire Block Diagram of Flash Memory

(3) Sector configuration

Figure 23.2-3 shows the sector configuration of Flash Memory.

	Writer address	CPU address
SA4 (16K bytes)	7FFFF _H	FFFFFF _H
SA3 (8K bytes)	7BFFF _H	FFBFFF _H
SA2 (8K bytes)	79FFF _H	FF9FFF _H
SA1 (32K bytes)	77FFF _H	FF7FFF _H
	7000 _H	FF0000 _H

Figure 23.2-3 Sector Configuration of Flash Memory

23.3 Write/Erase Modes

The Flash Memory can be accessed in two different ways; the Flash memory mode allowing write/erase directly from the external pins, and the other modes allowing write/erase from the CPU via the internal bus. These modes are selected by the mode external pins.

(1) Flash Memory model

The CPU stops when the mode pin is set to '111' while the reset signal is asserted. The flash memory interface circuit is directly connected to ports 0, 2, 3, 4 and 6 allowing direct control by the external pins. This mode makes the MCU seem like a standard flash memory in the external pins, and write/erase can be performed using a flash memory programmer.

In the Flash Memory mode all the operations supported by the flash memory automatic algorithm can be used.

(2) Other modes

The Flash Memory is located at the FF banks of the CPU memory space and like ordinary mask ROM can be read-accessed and program-accessed from the CPU through the flash memory interface circuit.

Writing/erasing the Flash Memory is performed by instructions from the CPU via the flash memory interface circuit. Therefore, this mode allows rewriting even when the MCU is soldered on the target board.

The sector protect operations can not be performed in these modes.

(3) Control signals of Flash Memory

Table 23.3-1 lists the flash memory control signals in the Flash Memory mode.

There is almost a one-to-one correspondence between the flash memory control signals and the external pins of the MBM29F400TA. The V_{ID} (12 V) pins required by the sector protect operations are MD0, MD1 and MD2 instead of A9, \overline{RESET} and \overline{OE} for the MBM29F400TA.

The MB90F497 has a memory capacity of one eighth of the MBM29F400TA, therefore the AQ18, AQ17 and AQ16 pins corresponding to the address signal A17, A16 and A15 of the MBM29F400TA are redundant. They should always be set to 1.

In the Flash Memory mode, the width of the external data bus is limited to 8 bits, allowing only one byte access. DQ15 to DQ8 are not supported. The \overline{BYTE} pin should always be set to 0.

Table 23.3-1 Flash Control Signals

Pin number		Normal function	Flash Memory mode	MBM29F400TA
FPT-64P-M06	FPT-64P-M09			
42 to 49	41 to 48	P20 to P27	AQ0 to AQ7	A-1,A0 to A6
51	50	P30	AQ16	A15
52	51	P31	$\overline{\text{CE}}$	$\overline{\text{CE}}$
53	52	P32	$\overline{\text{OE}}$	$\overline{\text{OE}}$
54	53	P33	$\overline{\text{WE}}$	$\overline{\text{WE}}$
55 to 56	54 to 55	P34 to P35	AQ17 to AQ18	A16 to A17
59	58	P36	BYTE	BYTE
60	59	P37	RY/ $\overline{\text{BY}}$	RY/ $\overline{\text{BY}}$
61 to 64, 1	60 to 64	P40 to P44	AQ8 to AQ12	A7 to A11
2, 3, 18	1,2,17	P61 to P63	AQ13 to AQ15	A12 to A14
19	18	MD0	MD0	A9(V _{ID})
21	20	MD1	MD1	$\overline{\text{RESET}}$ (V _{ID})
22	21	MD2	MD2	$\overline{\text{OE}}$ (V _{ID})
26 to 33	25 to 32	P00 to P07	DQ0 to DQ7	DQ0 to DQ7
20	19	$\overline{\text{RSTX}}$	$\overline{\text{RESET}}$	$\overline{\text{RESET}}$
	Not supported			DQ8 to DQ15

23.4 Control Status Register (FMCS)

The control status register (FMCS) with the flash memory interface circuit, is used for writing to/erasing flash memory only when with the CPU access. This register cannot be used when the CPU is in the Flash Memory mode.

(1) Register arrangement

	7	6	5	4	3	2	1	0	← Bit No.
Address: 0000AE _H	INTE	RDYINT	WE	RDY	Reserved	LPM1	Reserved	LPM0	FMCS
Read/write ⇒	(R/W)	(R/W)	(R/W)	(R)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(0)	(X)	(0)	(X)	(0)	(0)	(0)	(0)	

(2) Explanation of each bit

[Bit 7] INTE (INTerrupt Enable)

This bit generates an interrupt to the CPU at the end of writing to/erasing the Flash Memory. When both the INTE and RDYINT bits are 1, an interrupt to the CPU is generated. When the INTE bit is 0, no interrupt is generated.

[Bit 6] RDYINT (Ready Interrupt)

After completion of writing to/erasing the Flash Memory, this bit becomes 1. Writing 0 clears this bit to 0 and writing 1 is ignored. This bit is set to 1 at the rising edge of the RY/ $\overline{\text{BY}}$ signal from the flash memory unit.

[Bit 5] WE (Write Enable)

This bit enables writing to the flash memory area. A Write command to the Flash Memory is accepted only when this bit is 1.

[Bit 4] RDY (ReadDY)

This bit indicates the Flash Memory is ready to accept Write/Erase commands. While this bit is 0, writing to/erasing flash memory is impossible. However, the Read/Reset command and Sector Erase Suspend command are accepted even when this bit is 0.

[Bit 3] Reserved bit

This bit is reserved for the device test. It should always be set to 0 in user applications.

[Bit 2, 0] LPM1 and LPM0 (Low Power Mode) bits

These bits control the power dissipation modes of the Flash Memory in conjunction with the $\overline{\text{CE}}$ signal and the power dissipation modes are related to internal access times. Therefore the setting of these bits must be performed with great care. Otherwise it will cause mis-behaviour of the CPU resulting from incorrect data read.

LPM1	LPM0	Machine clock
0	0	Up to 16MHz
0	1	Up to 4MHz
1	0	Up to 8MHz
1	1	Up to 10MHz

[Bit 1] Reserved bit

This bit is reserved for the device test. It should always be set to 0 in user applications.

23.5 Read/Write Access

In the Flash Memory mode, read/write access to the Flash Memory must be under control of the external pins. However, with the CPU access, there are no special timing constraints on read/write access because the flash memory is controlled by the flash memory interface circuit.

In this section, “write access” does not directly mean “program Flash Memory”. It implies “activation of the Flash commands”.

(1) Read/write access in Flash memory mode

Table 23.5-1 gives the setting of pins for read/write access in the Flash Memory mode. There is no special problem with control of these pins if connected to a flash memory writer. However, in other cases, timing specifications must be met. For the timing specifications, see Section 23.10 and 23.11.

In the Flash Memory mode, the data bus width is limited to 8 bits, so the $\overline{\text{BYTE}}$ pin should always be fixed to 0.

Table 23.5-1 Setting Conditions of Pins for Read/Write Access in Flash Memory Mode

Operation	CE	OE	WE	AQ0 to AQ18	DQ0 to DQ7	RST
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

(2) Read/write access with CPU

The access timing to the flash memory unit is controlled by the flash memory interface circuit. The read/write cycle is completed in two cycles of the machine clock.

Figure 23.5-2 and Figure 23.5-3 show the difference in the access timing between the LPM0 bit of the control status register set to 0 and 1 (LPM1 = 0).

There is a one-to-one correspondence between internal address bits 15 to 00 and AQ15 to AQ0, and between internal data bits 15 to 00 and DQ15 to DQ0 in the Flash Memory mode.

For write access to the Flash Memory, the WE bit of the control status register must be set to 1 beforehand. This bit is used to prevent malfunction of the automatic algorithm as a result of erroneous writing to the flash memory unit caused by noise. When not performing write access, this bit should be set to 0.

When LPM0 = 0, like other memory, there are no special constraints on read/write access to the Flash Memory if the operating frequency of the CPU is in the operation range.

When LPM0 = 1, reading and writing are controlled by the timing of $\overline{\text{CE}}$, so severe constraints are imposed on the CPU operation frequency. Therefore, read/write access with LPM0 = 1 should always be made with the CPU operating frequency set to 4 MHz or less.

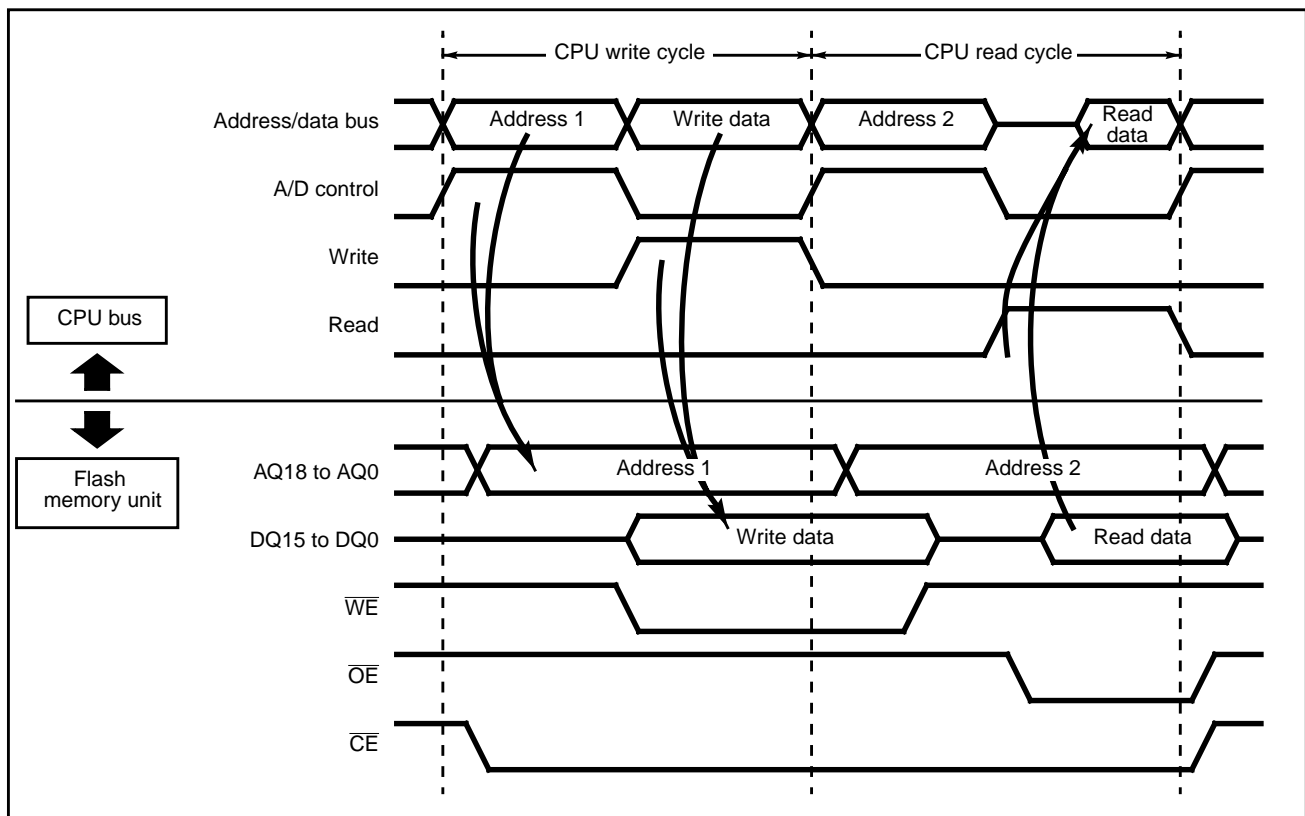


Figure 23.5-2 Flash Memory Interface Circuit (LPM0 = 0, LPM1=0)

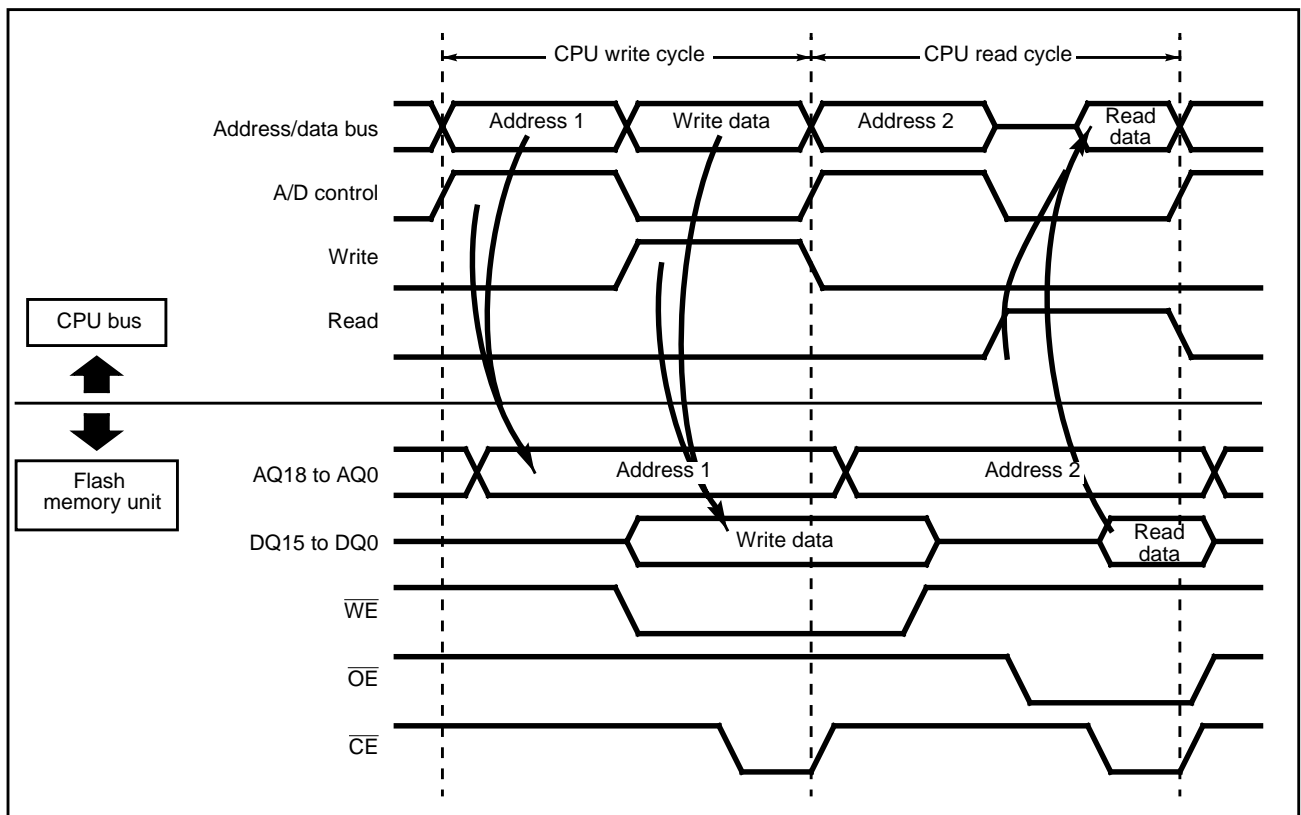


Figure 23.5-3 Flash Memory Interface Circuit (LPM0 = 1, LPM1=0)

23.6 Automatic Write/Erase Algorithm

Irrespective of the Flash Memory mode or other modes, writing to/erasing the flash memory unit is performed by starting the flash memory automatic algorithm.

Once the automatic algorithm is activated, the user need not pay attention to the operation in the flash memory unit for writing/erasing, nor to setup required by the flow, such as timer settings, modes and registers, and decision programming.

To start the automatic algorithm, various sequence of write accesses are executed in 1 to 6 cycles. They are called Flash commands.

23.6.1 Flash Commands

There are four commands for starting the automatic algorithm of the Flash Memory unit; Read/Reset, Write, Chip Erase, and Sector Erase. There are also Erase Suspend and Erase Resume commands for the sector erase operation.

Table 23.6.1-1 and Table 23.6.1-2 give the command sequence lists in the Flash Memory and other modes.

(1) Command sequence

Table 23.6.1-1 and Table 23.6.1-2 list the commands available for the flash memory unit.

All data is written to command registers by byte access but should be written by word access with the CPU access. Upper data bytes are ignored.

The Flash Memory mode permits selection of byte access or word access using the external pin $\overline{\text{BYTE}}$. However, at present only an 8-bit external data bus width is supported. Accordingly, the external pin $\overline{\text{BYTE}}$ should always be fixed to 0.

Table 23.6.1-1 Command Sequence List (CPU access)

Command Sequence	Write Cycle of Bus	Write Cycle of First Bus		Write Cycle of Second Bus		Write Cycle of Third Bus		Read/Write Cycle of Fourth Bus		Write Cycle of Fifth Bus		Write Cycle of Sixth Bus	
		Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
Read/Reset*	1	**xxxx	xxF0	—	—	—	—	—	—	—	—	—	—
Read/Reset*	4	**AAAA	xxAA	**5554	xx55	**AAAA	xxF0	RA	RD	—	—	—	—
Write	4	**AAAA	xxAA	**5554	xx55	**AAAA	xxA0	PA (even)	PD (word)	—	—	—	—
Chip Erase	6	**AAAA	xxAA	**5554	xx55	**AAAA	xx80	**AAAA	xxAA	**5554	xx55	**AAAA	xx10
Sector Erase	6	**AAAA	xxAA	**5554	xx55	**AAAA	xx80	**AAAA	xxAA	**5554	xx55	SA (even)	xx30
Sector Erase Suspend	Input of address **xxxx or **xxxx or data (xxB0H) suspends sector erasing.												
Sector Erase Resume	Input of address **xxxx or **xxxx or data (xx30H) suspends and resumes sector erasing.												

Addresses in the table are the values in the CPU memory space. All addresses and data are hexadecimal values, where 'x' is any value and ** is FF for MB90F497.

RA: Read address

PA: Write address. Only even addresses can be specified.

SA: Sector address (See Table 23.6.1-3). Only even addresses can be specified.

RD: Read data

PD: Write data. Only word data can be specified.

Note*: Two Read/Reset commands reset flash memory to the read mode.

Table 23.6.1-2 Command Sequence List (Flash Memory Mode)

Command Sequence	Write Cycle of Bus	Write Cycle of First Bus		Write Cycle of Second Bus		Write Cycle of Third Bus		Read/Write Cycle of Fourth Bus		Write Cycle of Fifth Bus		Write Cycle of Sixth Bus	
		Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
Read/Reset*	1	*xxxx	F0	—	—	—	—	—	—	—	—	—	—
Read/Reset*	4	*AAAA	AA	*5554	55	*AAAA	F0	RA	RD	—	—	—	—
Write	4	*AAAA	AA	*5554	55	*AAAA	A0	PA	PD (byte)	—	—	—	—
Chip Erase	6	*AAAA	AA	*5554	55	*AAAA	80	*AAAA	AA	*5554	55	*AAAA	10
Sector Erase	6	*AAAA	AA	*5554	55	*AAAA	80	*AAAA	AA	*5554	55	SA	30
Sector Erase Suspend	Input of address *xxxx or *xxxx or data (B0H) suspends sector erasing.												
Sector Erase Resume	Input of address *xxxx or *xxxx or data (30H) suspends and resumes sector erasing.												

Addresses in the table are values for writer addresses. All addresses and data are hexadecimal values, where 'x' is any value and * is 7 for MB90F497.

RA: Read address
PA: Write address
SA: Sector address (See Table 23.6.1-3)
RD: Read data
PD: Write data. Only word data can be specified.

Note*: Two Read/Reset commands reset flash memory to the read mode.

(2) Read/Reset command

There are two Read/Reset commands; one is executed in one bus operation, and the other is executed at three bus operations; the command sequence of both is essentially the same.

The flash memory unit is in the read/reset state at default and always enters this state at power-on and at the normal termination of command execution. The read/reset state can be the wait state for input of another command.

In the read/reset state, data can be read by normal read access. When the CPU is in a normal mode, like mask ROM, program access from the CPU is enabled.

Consequently, this command is not needed for reading data and is used mainly to reset the automatic algorithm when command execution has not terminated for some reason.

(3) Write command

The Write command is executed in four bus operations. In the command sequence, two unlocked cycles are executed, followed by the Write Setup command and write data cycle. Automatic writing is started at the end of the 4th write cycle.

When the CPU is in a normal mode, only even addresses can be specified in the write data cycle. Specifying odd addresses disables correct writing. Therefore, writing in a normal mode is performed in words at even addresses. There are no such limitations in the Flash Memory mode.

After the command sequence for the automatic write algorithm has been executed, the flash memory unit generates correct write pulses created automatically to verify the margins of written data. The end of automatic writing can be determined using the data polling function (See 23.6.2). After the completion of writing, the flash memory unit returns to the read/reset state.

All commands are ignored during writing. If a hardware reset occurs during writing, data being written to the address become invalid.

Writing is possible in any address order or even beyond sector boundaries. However, execution of one Write command, results in writing one word of data with the CPU access, and one byte of data in the Flash Memory mode.

Data 0 cannot be returned to data 1 by writing. When data 1 is written to data 0, the flash memory unit may hang up or possibly complete the write operation without any error. However when the data is read in the read/reset state, it remains 0. Only erasing enables data 0 to be set to data 1.

Figure 23.6.1-1 shows the writing procedure using the Write command.

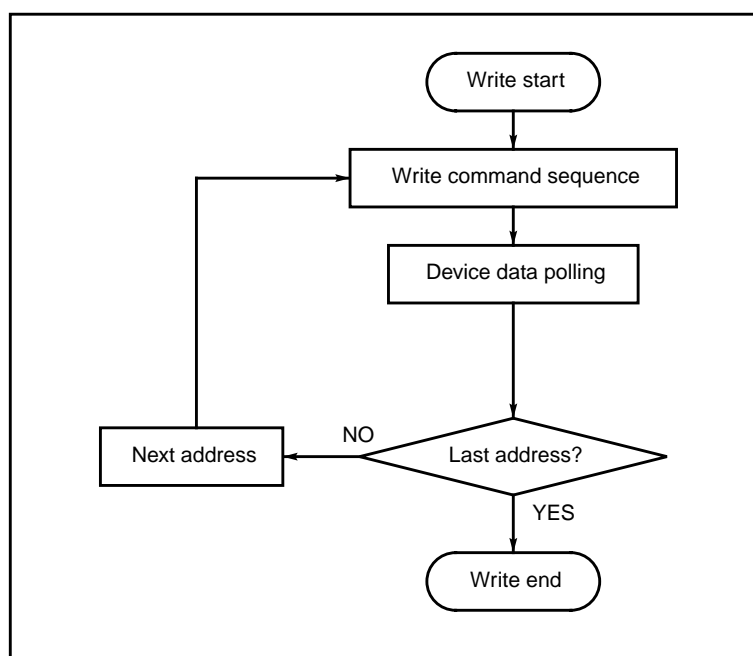


Figure 23.6.1-1 Writing Procedure Using Write Command

(4) Chip Erase command

The Chip Erase command is executed in six bus operations. In the command sequence, two unlocked cycles are executed, followed by the Write Setup command. The two same unlocked cycles are further continued to write the Chip Erase command. Chip erasing is started at completion of the 6th write cycle.

Before chip erasing, the user need not perform writing to the flash memory unit. During execution of the automatic erase algorithm, the flash memory unit writes 0 patterns for verification before automatically erasing all cells.

The end of automatic erasing can be determined using the data polling function (See 23.6.2). After completion of erasing, the flash memory unit returns to the read/reset state.

Figure 23.6.1-2 shows the chip erasing procedure using the Chip Erase command.

(5) Sector Erase command

The Sector Erase command is executed in six bus operations. In the command sequence, two unlocked cycles are executed, followed by the Write Setup command. The two same unlocked cycles are continued to write a write sector erase code to a sector address in the 6th cycle, and waiting 50 ms for sector erasing.

When erasing more than one sector, the sector erase code (30H) is written to the sector address, following the above sequence. Sector erasing is started after a 50 ms period of waiting for sector erasing is completed after the last sector erase code has been written. That is, when erasing more than one sector simultaneously, the next sector to be erased must be input within 50 ms and subsequent sectors may not be accepted. The validity of the next sector erase code can be monitored by the sector erase timer (see 23.6.2). When the commands other than the Sector Erase or the Sector Erase Suspend command are accepted during waiting for sector erasing, the flash memory unit returns to the read/reset state, ignoring the previous command sequence. In this case, erasing is completed by erasing the sector again.

Although sector addresses can be selected in any combination and order, selected sectors are erased in ascending order.

Before sector erasing, the user need not perform writing to the flash memory unit. Writing is performed to all memory within the sectors to be erased automatically. Other sectors are not affected during sector erasing.

The end of automatic sector erasing can be determined using the data polling function (See 23.6.2). After the completion of erasing, the flash memory unit returns to the read/reset state. Other commands are ignored during sector erasing.

Figure 23.6.1-2 shows the sector erasing procedure using the Sector Erase command.

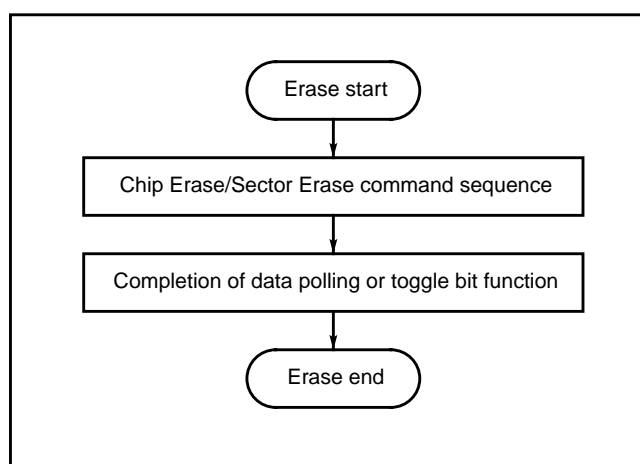


Figure 23.6.1-2 Chip Erasing/Sector Erasing Procedure Using Chip Erase/Sector Erase Command

[Sector Address]

A sector address is an arbitrary address representing each sector. Any one accessible address within the sector is sufficient. Sector addressing the CPU access is different from that in the Flash Memory mode.

With the CPU access, any even address of the sector to be erased is selected from the addresses (CPU memory space) listed in Table 23.6.1-3.

In the Flash Memory mode, any address (either even or odd) of the sector to be erased is selected from the addresses (write addresses) listed in Table 23.6.1-3, or nine address pins AQ17 to AQ9 corresponding to the sectors listed in Table 23.6.1-3 are set and AQ18 is set to 1.

Table 23.6.1-3 Sector Address for MB90F497

Sector	AQ17	AQ16	AQ15	AQ14	AQ13	AQ12	AQ11	AQ10	AQ9	Address Range	
										Flash Memory Mode	CPU Access
SA4	1	1	1	1	X	X	X	X	X	7C000 _H to 7FFFF _H	FFC000 _H to FFFFF _H
SA3	1	1	1	0	1	X	X	X	X	7A000 _H to 7BFFF _H	FFA000 _H to FFBFFF _H
SA2	1	1	1	0	0	X	X	X	X	78000 _H to 79FFF _H	FF8000 _H to FF9FFF _H
SA1	1	1	0	X	X	X	X	X	X	70000 _H to 77FFF _H	FF0000 _H to FF7FFF _H

(6) Sector Erase Suspend command

The Sector Erase Suspend command suspends sector erasing by the automatic algorithm to enable reading of data from the sectors not being erased (writing is not possible). This command is enabled only during sector erasing including the period of waiting for sector erasing, and is ignored during chip erasing and writing. This command is executed by writing the sector erase suspend code (B0_H). In this case, any address in the flash memory area is specified.

As soon as this command is input during the period of waiting for sector erasing, the flash memory unit completes waiting for sector erasing to suspend erasing and enters the erase suspend state.

When the Sector Erase Suspend command is input during sector erasing after the period of waiting for sector erasing, the flash memory unit enters the erase suspend state after a maximum of 15 ms.

Whether the flash memory unit has stopped erasing and is in the erase suspend state can be determined using the data polling function or toggle bit function (See 23.6.2). Input of the Sector Erase Suspend command in the erase suspend state is ignored.

In the erase suspend state, data can be read by normal read access to the sectors not being erased.

(7) Sector Erase Resume command

The Sector Erase Resume command resumes sector erasing suspended by the Sector Erase Suspend command. This command is executed by writing the erase resume code (30H). In this case, any address in the flash memory area is specified.

In the erase resume state, the flash memory unit ignores input of the Sector Erase Resume command but accepts input of the Sector Erase Suspend command.

23.6.2 Execution State of Automatic Algorithm

Since the writing/erasing flow is controlled by the automatic algorithm, the flash memory unit has a hardware sequence flag and Ready/Busy signal to inform units outside the flash memory unit of its internal operation and operation completion.

(1) List of hardware sequence flags

A hardware sequence flag consists of 4-bit outputs of DQ7, DQ6, DQ5, and DQ3, each of which has a data polling flag, toggle bit flag, timing limit exceeding flag, and sector erase timer flag.

Hardware sequence flags can be referenced by read-accessing addresses in the flash memory area after starting the automatic algorithm. This is common to both the CPU access and Flash Memory mode.

Table 23.6.2-1 lists the functions of hardware sequence flags.

Table 23.6.2-1 List of Functions of Hardware Sequence Flags

State		DQ7	DQ6	DQ5	DQ3
State change in normal operation	Write => Write completed (when write address specified)	$\overline{\text{DQ7}}$ DQ7	Toggle Stop	0 1	0
	Chip/Sector Erase => Erase completed	0 1	Toggle Stop	0 1	1
	Sector Erase wait => Erase started	0	Toggle	0	0 1
	Erase => Sector Erase suspended (Sector being erased)	0 1	Toggle 1	0	1 0
	Sector Erase Suspend => Resume (Sector being erased)	1 0	1 Toggle	0	0 1
	Sector Erase Suspend (Sector not being erased)	DATA	DATA	DATA	DATA
Abnormal operation	Write	$\overline{\text{DQ7}}$	Toggle	1	0
	Chip/Sector Erase	0	Toggle	1	1

Notes:

- 1) DQ15 to DQ8 are not used.
- 2) DQ4 is used for the device test and DQ2 to DQ0 are reserved for future use.

(2) Data polling flag (DQ7)

The data polling flag is mainly used to notify that the automatic algorithm is executing or has been terminated. This flag is valid for Write, Chip Erase, or Sector Erase.

- Writing

Read access during execution of the writing algorithm causes the flash memory unit to output the reverse data of bit 7 last written, irrespective of the value at the address specified by the address signal.

Read access at the end of the writing algorithm causes the flash memory unit to output bit 7 of the read value at the address specified by the address signal.

- Chip/sector erasing

Read access during execution of the chip/sector erasing algorithm causes the flash memory unit to output 0, irrespective of the value at the address specified by the address signal. Read access at the end of the chip/sector erasing algorithm causes the flash memory unit to output 1.

- Sector erasing suspension

Read access during sector erasing suspension causes the flash memory unit to output 1 if the address specified by the address signal belongs to the sector being erased. The flash memory unit outputs bit 7 of the read value at the address specified by the address signal if the address specified by the address signal does not belong to the sector being erased.

Referencing this flag together with the toggle bit flag permits a decision on whether the flash memory unit is in the erase suspended state and which sector is being erased.

Read access to the specified address while the automatic algorithm is ignored. At data reading, other bits can be output correct data after the end of data polling by bit 7. Data reading after the end of the automatic algorithm should be performed after completion of data polling has been checked.

(3) Toggle bit flag (DQ6)

Like the data polling flag, the toggle bit flag is mainly used to notify that the automatic algorithm is executing or has been terminated using the toggle bit function. This flag is valid for Write, Chip Erase, or Sector Erase.

- Writing, chip and sector erasing

Continuous read access during execution of the writing and chip/sector erasing algorithms causes the flash memory unit to toggle the 1 or 0 state at every read cycle, irrespective of the value at the address specified by the address signal.

Continuous read access at the end of the writing and chip/sector erasing algorithms causes the flash memory unit to stop toggling bit 6 to output bit 6 of the value at the address specified by the address signal.

At writing, if the sector where data is to be written is rewrite-protected, the toggle bit terminates the toggle operation of about 2 ms and then terminates without rewriting data. At erasing, if all the selected sectors are write-protected, the toggle bit performs toggling for about 100 ms and then returns to the read/reset state without rewriting data.

- Sector erasing suspension

Read access during sector erase suspension causes the flash memory unit to output 1 if the address specified by the address signal belongs to the sector being erased. The flash memory unit outputs the value of bit 6 of the value at the address specified by the address signal if the address specified by the address signal does not belong to the sector being erased.

(4) Time limit exceeding flag (DQ5)

The time limit exceeding flag is used to notify that the automatic algorithm has executed beyond the predetermined time (internal pulse count) in the flash memory unit. This flag is enabled for Write, Chip Erase, or Sector Erase.

Read access after the Write, Chip Erase, or Sector Erase command causes the flash memory unit to output 0 if the automatic algorithm is executed within the specified time (time required for writing/erasing) and to output 1 if the algorithm is executed beyond the specified time, irrespective of the value at the address specified by the address signal. This is done irrespective of whether the automatic algorithm is executing or has terminated, making it possible to determine whether writing/erasing was successful or unsuccessful (See Examples of using hardware sequence flags on this page). That is, when this flag outputs 1, writing or erasing can be determined to have been unsuccessful, if the automatic algorithm is regarded as still being executed by the data polling function or toggle bit function.

For example, if the user writes 1 to the address where 0 is written, the fail state occurs. In this case, the flash memory hangs up and execution of the automatic algorithm is not terminated. Consequently, valid data cannot be output from DQ7. Bit 6 exceeds the time limit without stopping the toggle operation, and bit 5 outputs 1. It should be noted that this state indicates that flash memory is not defective, but has not been used properly.

If this state occurs, the Reset command should be executed.

(5) Sector erase timer flag (DQ3)

The sector erase timer flag is used to notify that the automatic algorithm is executed during the period of waiting for sector erasing after the Sector Erase command has started. This flag is valid with Sector Erase.

- **Sector erasing**

Read access after the Sector Erase command causes the flash memory unit to output 0 if the automatic algorithm is executed within the period of waiting for sector erasing and to output 1 if the algorithm is executed beyond the period of waiting for sector erasing, irrespective of the value at the address specified by the address signal. When the data polling function or toggle bit function indicates that the erasing algorithm is executing, internally-controlled erasing has already started if this flag is 1. Continuous writing of sector erase codes or commands other than the Sector Erase Suspend command are ignored until the data polling function or toggle bit function indicates the end of erasing. If this flag is 0, the flash memory unit accepts writing of additional sector erase codes. To ensure this, the state of this flag should be checked by software before continuing to write sector erase codes. If this flag is 1 after the second state check, writing of additional sector erase codes may not be accepted.

- **Sector erase suspension**

Read access during Sector Erase Suspend causes the flash memory unit to output 1, if the address specified by the address signal belongs to the sector being erased. The flash memory unit outputs bit 3 of the value at the address specified by the address signal, if the address specified by the signal address does not belong to the sector being erased.

(6) Examples of using hardware sequence flags

Use of the previously-mentioned hardware sequence flags permits determination of the state of the automatic algorithm in the flash memory unit. Figure 23.6.2-1 and Figure 23.6.2-2 give the flowcharts of decision on writing/erasing using the data polling function and toggle bit function.

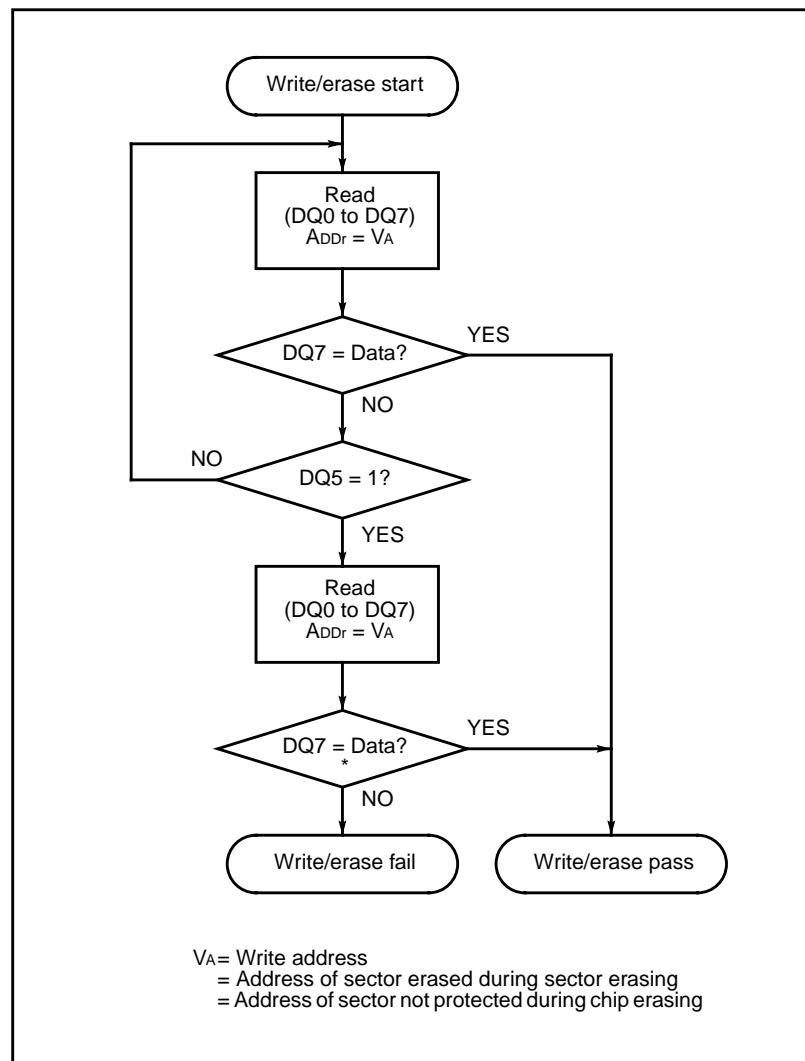


Figure 23.6.2-1 Flowchart of Decision on Writing/Erasing Using Data Polling Function

Note*: DQ7 is changed as DQ5 is changed. DQ7 must be re-checked, even if DQ5 = 1.

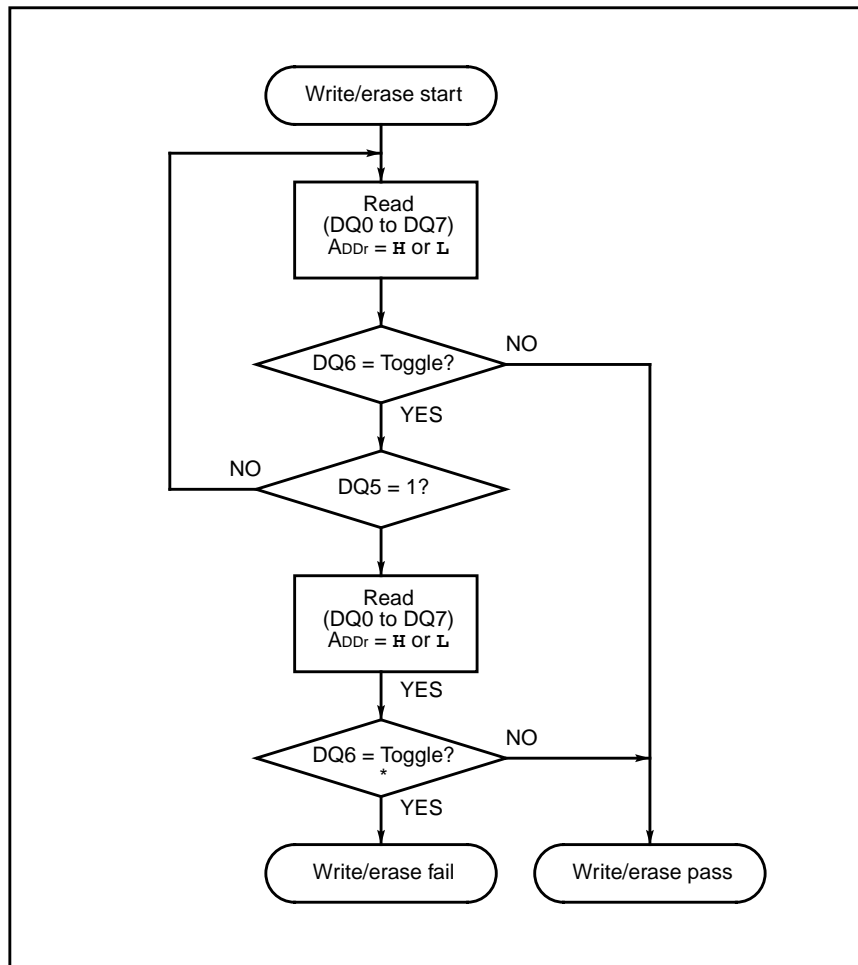


Figure 23.6.2-2 Flowchart of Decision on Writing/Erasing Using Toggle Bit Function

Note*: DQ6 stops toggling as DQ5 is changed to 1. DQ6 must be re-checked, even if DQ5 = 1.

(7) Ready/busy signal

The flash memory unit has hardware sequence flags, such as a data polling flag and toggle bit flag, as well as a hardware signal, such as a Ready/Busy signal to notify that the automatic algorithm is executing or terminated.

The form of the Ready/Busy signal varies depending on whether the CPU is in the Flash Memory mode or a normal mode.

• Flash Memory mode

The Ready/Busy signal asynchronously supplies an $\text{RY}/\overline{\text{BY}}$ signal from the flash memory unit outside the chip as an open-drain output $\text{RY}/\overline{\text{BY}}$ pin. Connecting a pull-up resistor to V_{CC} allows parallel connection of several $\text{RY}/\overline{\text{BY}}$ pins.

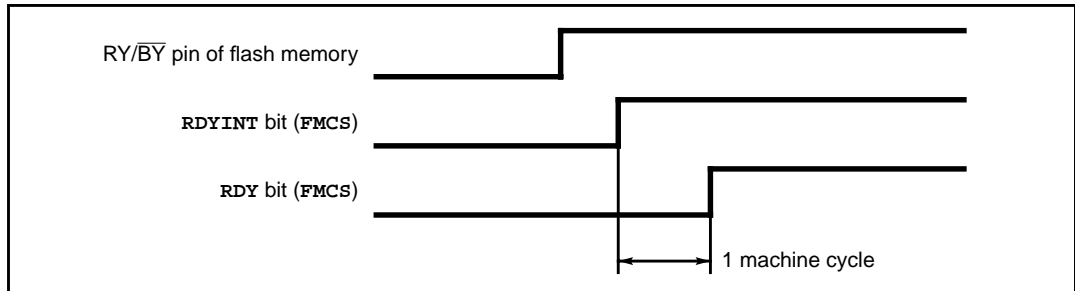
When the output of the $\text{RY}/\overline{\text{BY}}$ pin is 0, the flash memory unit is writing or erasing and busy. Neither Write nor Erase commands can be accepted. When the output of the $\text{RY}/\overline{\text{BY}}$ pin is 1 with an external pull-up resistor connected, the flash memory unit is ready for reading/writing. In the erase suspend mode, the output of the $\text{RY}/\overline{\text{BY}}$ pin is 1 with an external pull-up resistor connected. The $\text{RY}/\overline{\text{BY}}$ pin indicates the ready state during external reset.

- Normal mode

The Ready/Busy signal supplies an interrupt request to the CPU via the flash memory interface circuit.

When the flash memory unit changes from busy to ready with the INTE bit of the control status register (FMCS) is set to 1, an interrupt request is issued to the CPU. At this time, both the RDY bit and RDYINT bit of the FMCS are set to 1s. With the INTE bit set to 0, no interrupt request is issued to the CPU.

When the RDYINT bit of the FMCS is cleared to 0 after issuing an interrupt request to the CPU, the interrupt request to the CPU is cancelled. The RDY bit of the FMCS is also set to 0.



Note: The RDYINT bit and RDY bit of the FMCS do not change simultaneously. Software should be programmed so that a decision may be made by either of the bits.

23.7 Sector Protect Operation

The flash memory unit has the sector protect function to disable illegal writing/erasing in sectors. Once protected, sectors remain unchanged unless the device is damaged. If sectors are protected temporarily, the protection can be cancelled for writing/erasing. This operation is performed using the sector protect operations.

The sector protect operation does not have an automatic algorithm like the writing/erasing algorithm. It can be executed only in the Flash Memory mode. Therefore, this operation can only be performed using a flash memory writer.

(1) List of sector protect operations

There are three sector protect operations; Enable Sector Protect, Verify Sector Protect, and Temporary Sector Protect cancel. Table 23.7-1 lists pin settings for each operation.

Table 23.7-1 List of Sector Protect Operations

Operation	$\overline{\text{CE}}$	$\overline{\text{OE}}$	$\overline{\text{WE}}$	AQ0	AQ1	AQ2	AQ8	AQ9 to AQ17	DQ0 to DQ7	RST	MD2	MD1	MD0
Enable Sector Protect	L	H	L	×	×	×	L	Sector address*	×	H	V _{ID}	H	V _{ID}
Verify Sector Protect	L	L	H	×	×	H	L	Sector address*	Code output	H	H	H	V _{ID}
Temporary Sector Protect Cancel	×	×	×	×	×	×	×	×	×	H	H	V _{ID}	H

Note*: Set address pins AQ17 to AQ9 corresponding to sectors listed in Table 23.6.1-3 and AQ18 to 1.

(2) Enable Sector Protect

Enable Sector Protect provides writing to the protection circuit in the flash memory unit. This operation disables writing and erasing in any combination of the nine sectors. The MB90F497 is shipped with all sectors unprotected.

At this operation, the sector addresses (AQ17, AQ16, AQ15, AQ14, AQ13, AQ12, AQ11, AQ10, and AQ9) of the sectors to be protected must be set to the address signals and AQ8 must be set to 0. See Table 23.6.1-3 for the correspondence between sectors and sector addresses. Other operations are the same as those in the Flash Memory mode, such as setting AQ18 to 1 and $\overline{\text{BYTE}}$ to 0.

After V_{ID} (12 V) has been applied to MD2 and MD0 to set $\overline{\text{CE}}$ to 0, writing to the protection circuit is started at the fall of the $\overline{\text{WE}}$ pulse and terminated at the rise of the $\overline{\text{WE}}$ pulse. Sector addresses must be kept constant while the $\overline{\text{WE}}$ pulse is applied.

Once applied, sector protection cannot be cancelled. This disables subsequent writing/erasing to/from protected sectors.

(3) Verify Sector Protect

Verify Sector Protect verifies writing to the protection circuit in the flash memory unit.

At this operation, $\overline{\text{CE}}$ and $\overline{\text{OE}}$ are set to 0s and V_{ID} is applied to MD0 with $\overline{\text{WE}}$ set to 1 (in the margin mode). When a sector address (any one of AQ17, AQ16, AQ15, AQ14, AQ13, AQ12, AQ11, AQ10, and AQ9) is set to an address signal and read under the condition of (AQ8, AQ2, AQ1) = (0, 1, 0), 1 is output to DQ0 in the protected sectors. 00H is read in the unprotected sectors.

Figure 23.7-1 shows the sector protection algorithm using Enable Sector Protect and Verify Sector Protect.

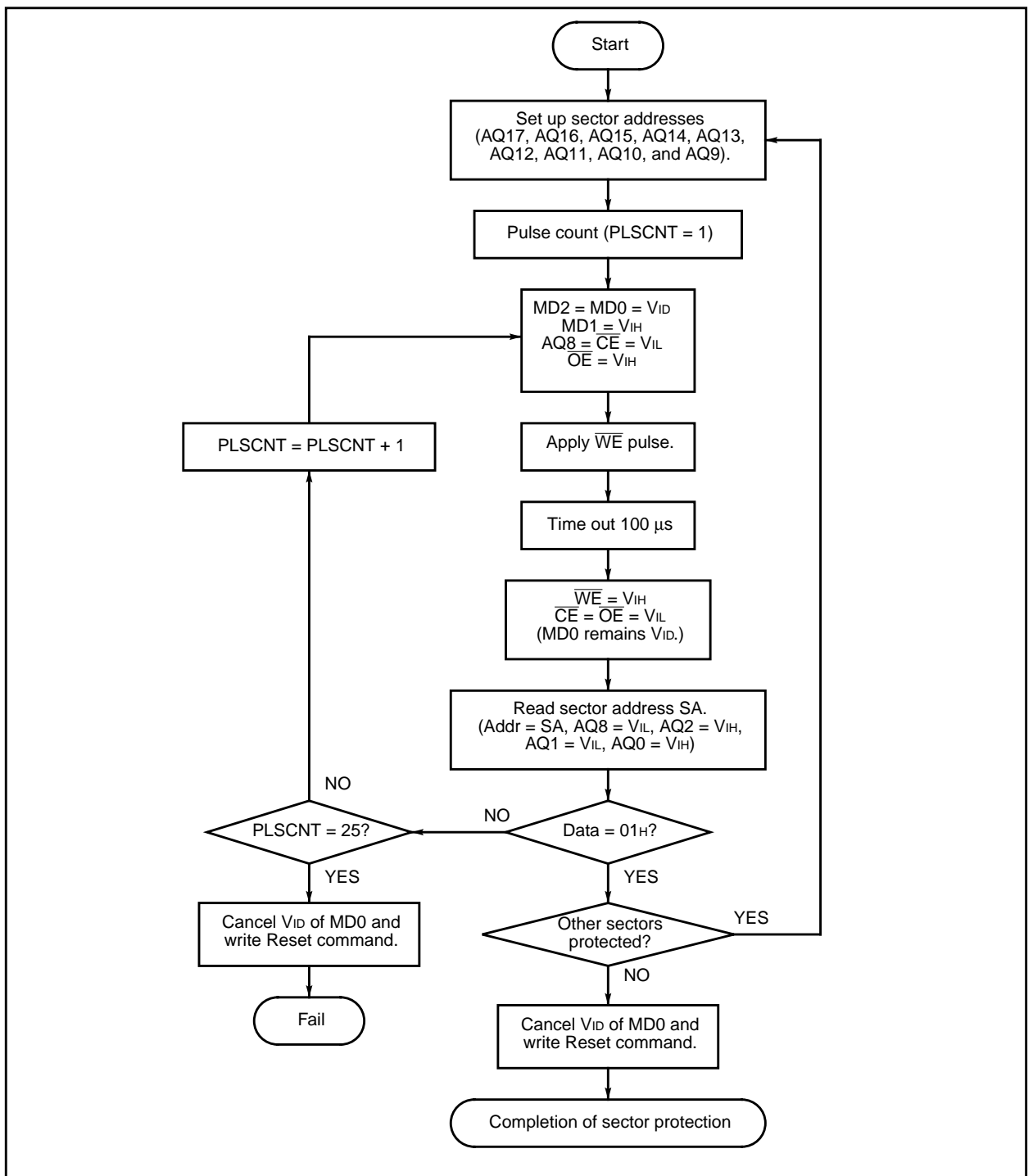


Figure 23.7-1 Sector Protection Algorithm Using Enable Sector Protect and Verify Sector Protect

(4) Temporary Sector Protect Cancel

Writing/erasing to/from sectors protected by Enable Sector Protect is impossible unless the device is damaged. Temporary Sector Protect Cancel allows temporary cancellation of information on sector protection set previously. This operation is set by continuously-applying V_{ID} to MD1. At this time, all data on sector protection set previously are ignored, writing/erasing to/from all sectors is enabled.

When MD1 is set to 1 (5 V), this operation is cancelled and all previously-protected sectors are protected again.

Figure 23.7-2 shows the algorithm for temporary sector protect cancellation.

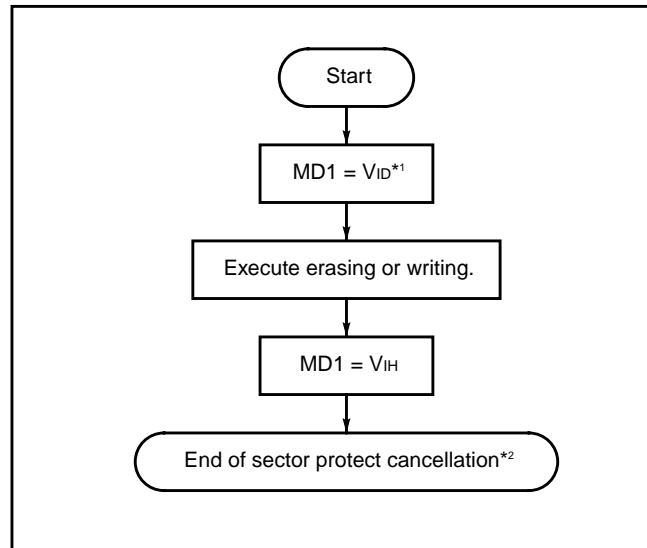


Figure 23.7-2 Algorithm for Temporary Sector Protect Cancellation

*1: Protection of all protected sectors is cancelled.

*2: Previously-protected sectors are protected again.

23.8 Connection to Flash Memory Writer

The Flash Memory mode of the MB90F497 is intended mainly for external connection to a flash memory writer. As indicated in Table 23.3-1, there is a slight difference between the external pins of the MB90F497 and the MBM29F400TA (4 Mbit flash memory). Connection to an MBM29F400TA writer requires the socket adapter shown in Figure 23.8-1.

The external pins of the MBM29F400TA (4 Mbit flash memory) correspond to those of the MB90F497 in the Flash Memory mode as indicated in Table 23.3-1. Connection to an MBM29F400TA writer requires a socket adapter with the same pin arrangement.

As indicated in Table 23.3-1, there is also a difference between the pins for supply of V_{ID} voltage (12 V) of the MBM29F400TA and the MB90F497. To allocate the normal digital voltage and V_{ID} voltage supplied from the writer to each pin of the MB90F497, the diode must be clamped by the socket adapter.

Figure 23.8-1 shows an example of connection from a flash memory writer to the MB90F497 using the socket adapter. Connections not shown in this figure should be made as indicated in Table 23.3-1.

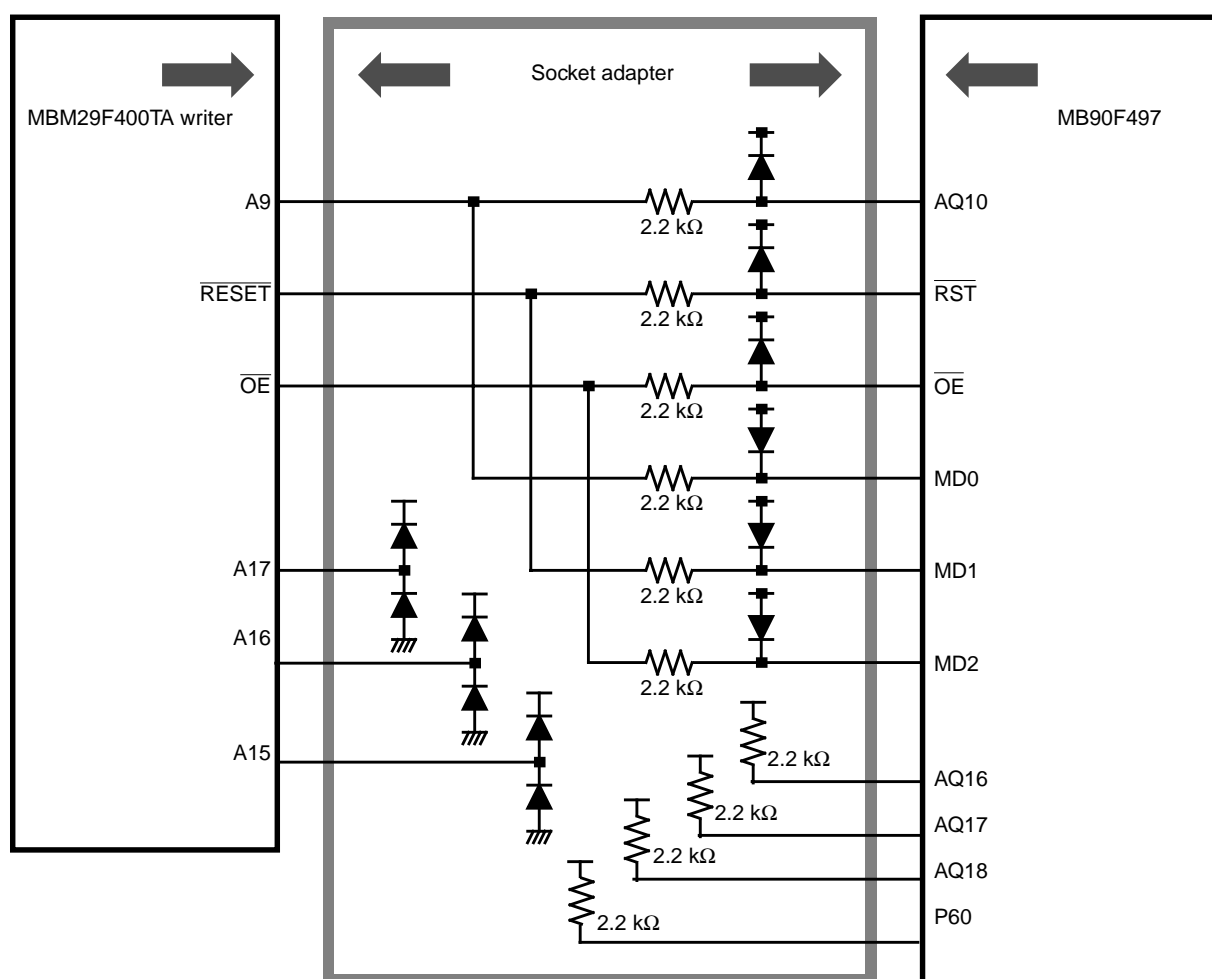


Figure 23.8-1 Example of Connection to Flash Memory Writer

23.9 Notes for Use of Flash Memory

Notes on the Flash Memory are given below.

(1) Input of hardware reset ($\overline{\text{RST}}$)

To input a hardware reset when the automatic algorithm is not started, where reading is in progress, a minimum of 500 ns should be taken at a low-level width. In this case, a maximum of 500 ns is required until data can be read from the Flash Memory after a hardware reset has been activated.

Similarly, to input a hardware reset when the automatic algorithm is activated, where writing/erasing is in progress, a minimum of 50 ns should be taken in a low-level width. In this case, 20 μs are required until data can be read after the executing operation has been terminated to initialize the Flash Memory.

A hardware reset during writing undefined data being written. A hardware reset during erasing may make the sector being erased unusable.

(2) Cancelling software reset and watchdog timer reset

When writing/erasing the Flash Memory with the CPU access and if reset conditions occur while the automatic algorithm is active, the CPU may run away. This occurs because these reset conditions cause the automatic algorithm to continue without initializing the flash memory unit, possibly preventing the flash memory unit from entering the read state when the CPU starts the sequence after the reset has been deasserted. These reset conditions should be inhibited during writing/erasing the Flash Memory.

(3) Program access to Flash Memory

When the automatic algorithm is operating, read access to the Flash Memory is disabled. With the memory access mode of the CPU set to the internal ROM mode, writing/erasing should be started after switching the program area to another area such as RAM.

In this case, when sectors (SA4) containing interrupt vectors are erased, writing/erasing interrupt processing cannot be executed.

For the same reason, all the other interrupt sources than the Flash Memory are disabled while the automatic algorithm is operating.

(4) Hold function

When the CPU accepts a hold request, the Write signal $\overline{\text{WE}}$ of the flash memory unit may be skewed to allow erroneous writing/erasing due to erroneous write. When the acceptance of a hold request is enabled (HDE bit of EPCR set to 1), ensure that the WE bit of the control status register (FMCS) is 0.

(5) Extended intelligent I/O service (EI²OS)

Write/erase interrupts issued to the CPU from the flash memory interface circuit are not acceptable to the EI²OS. It should not be used.

(6) Applying V_{ID}

Applying V_{ID} required for the sector protect operation should always be started and terminated when the supply voltage is on.

23.10 Timing Diagrams in Flash Memory Mode

Each timing diagram for the external pins in the Flash Memory mode is shown below. For the respective AC specifications, see Section 23.11 “AC Characteristics in Flash Memory Mode”.

(1) Data read by read access

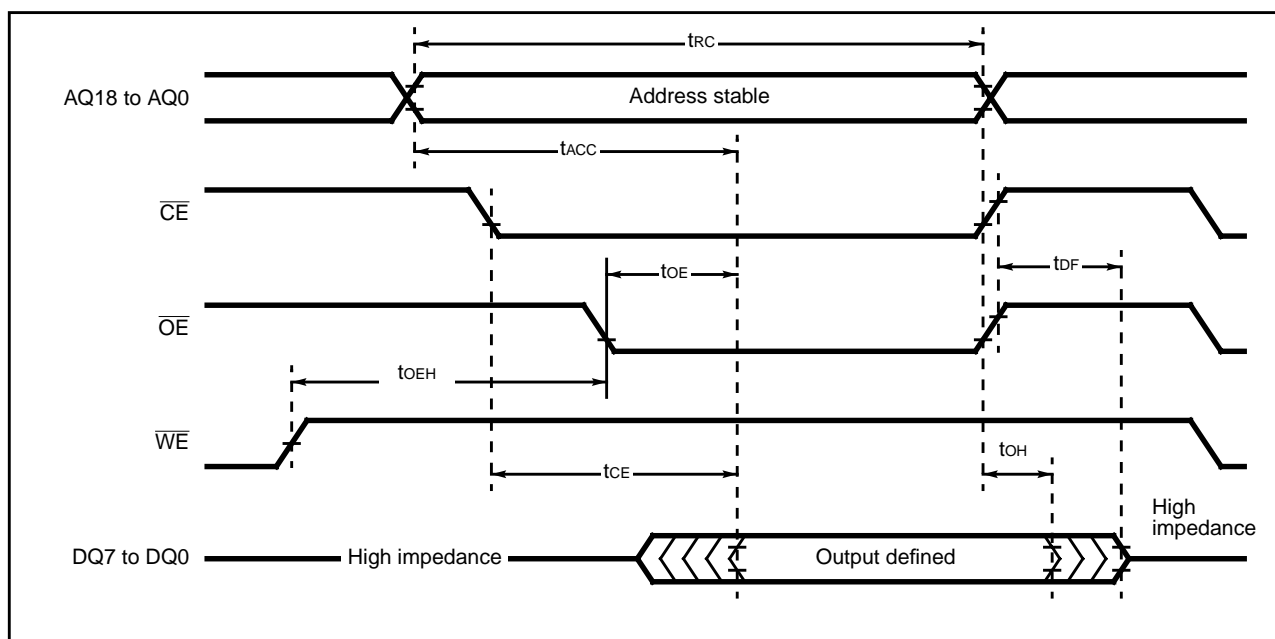


Figure 23.10-1 Timing Diagram for Read Access

(2) Write Data polling Read (\overline{WE} control)

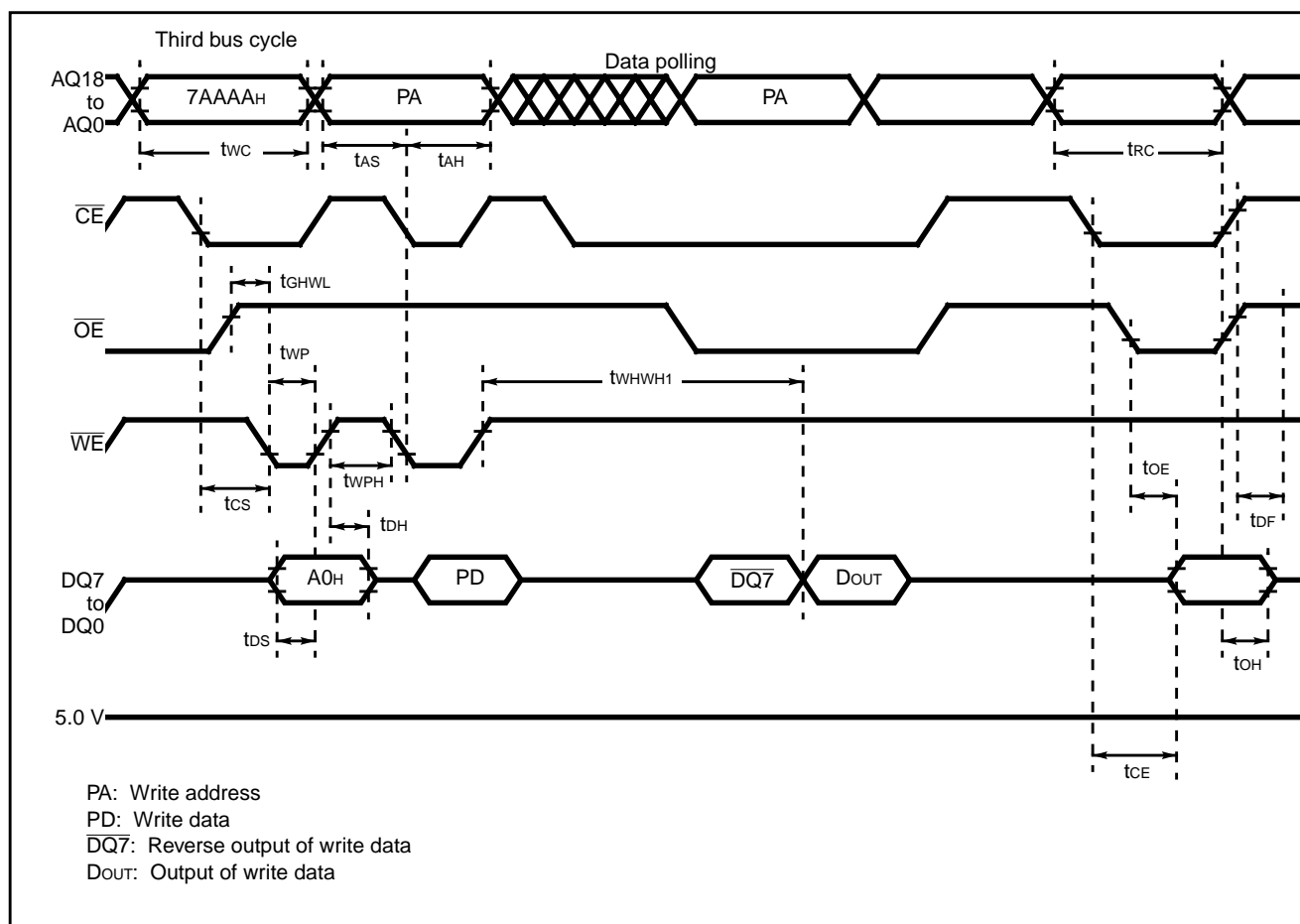


Figure 23.10-2 Timing Diagram for Write Access (\overline{WE} Control)

Note: The last two bus cycle sequences out of the four are described.

(3) Write Data polling Read (\overline{CE} control)

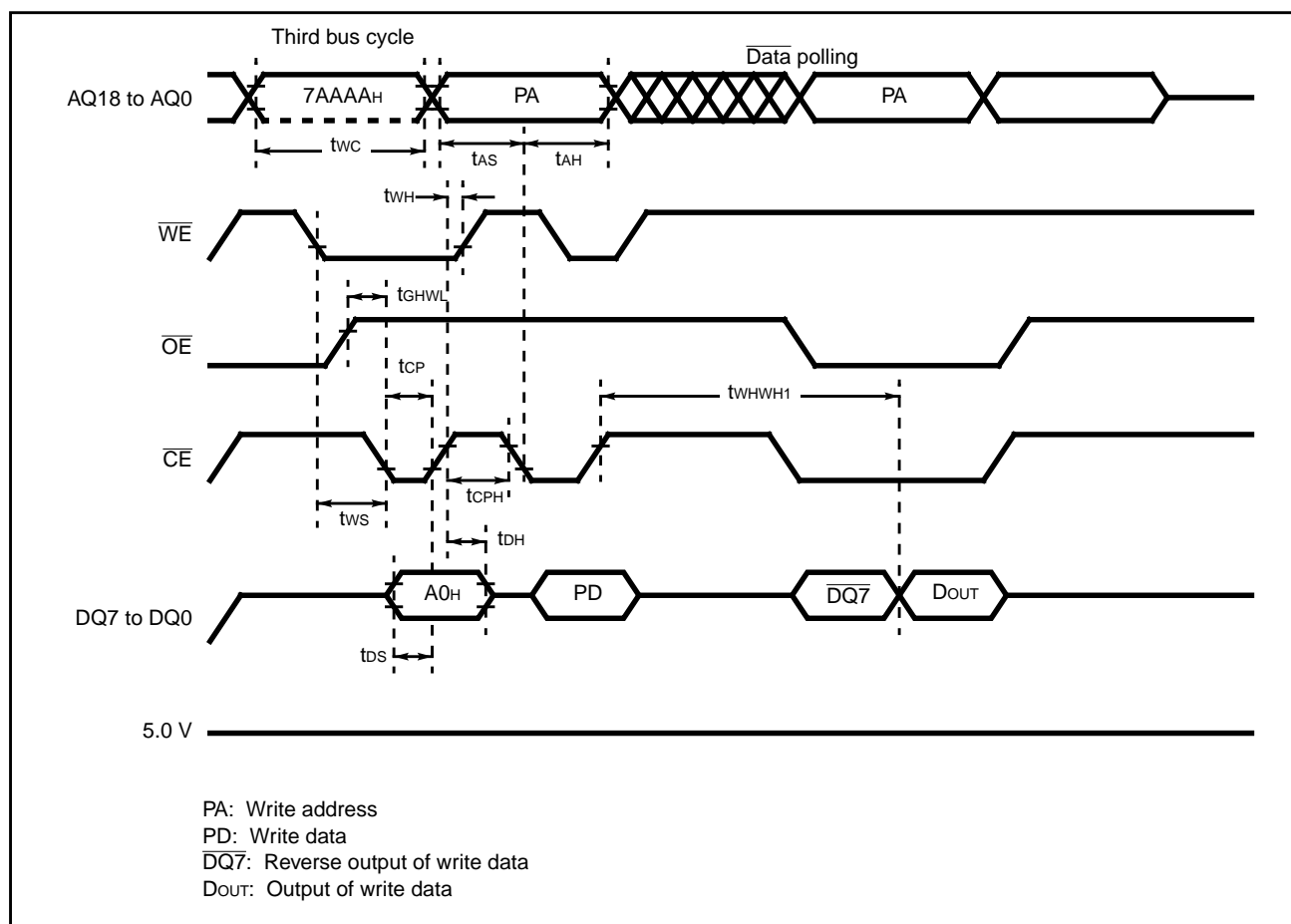


Figure 23.10-3 Timing Diagram for Write Access (\overline{CE} Control)

Note: The last two bus cycle sequences out of the four are described.

(4) Chip erase/sector erase command sequence

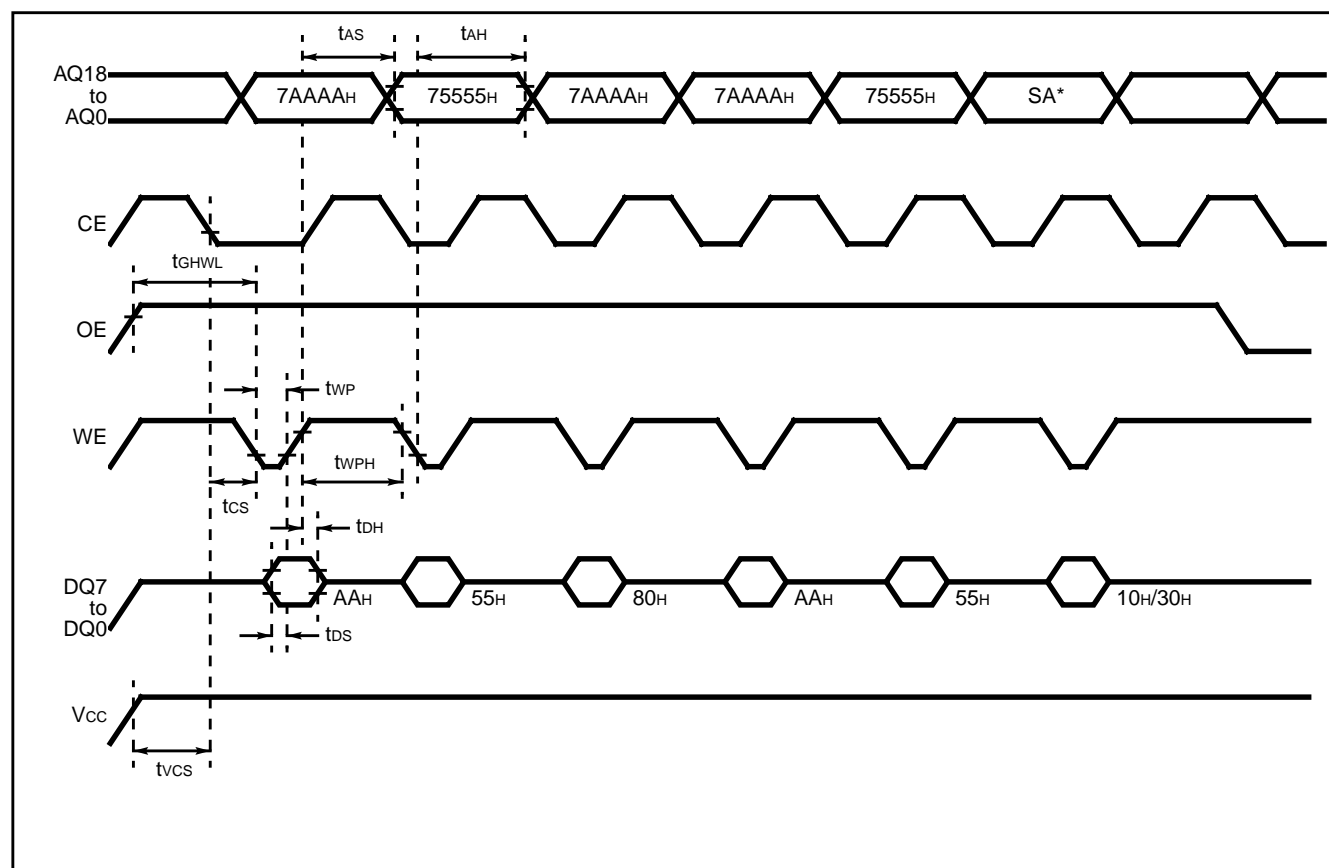


Figure 23.10-4 Timing Diagram for Write Access (Chip Erasing/Sector Erasing)

Note*: SA is the sector address at sector erasing. 7AAAAH is the address at chip erasing.

(5) Data polling

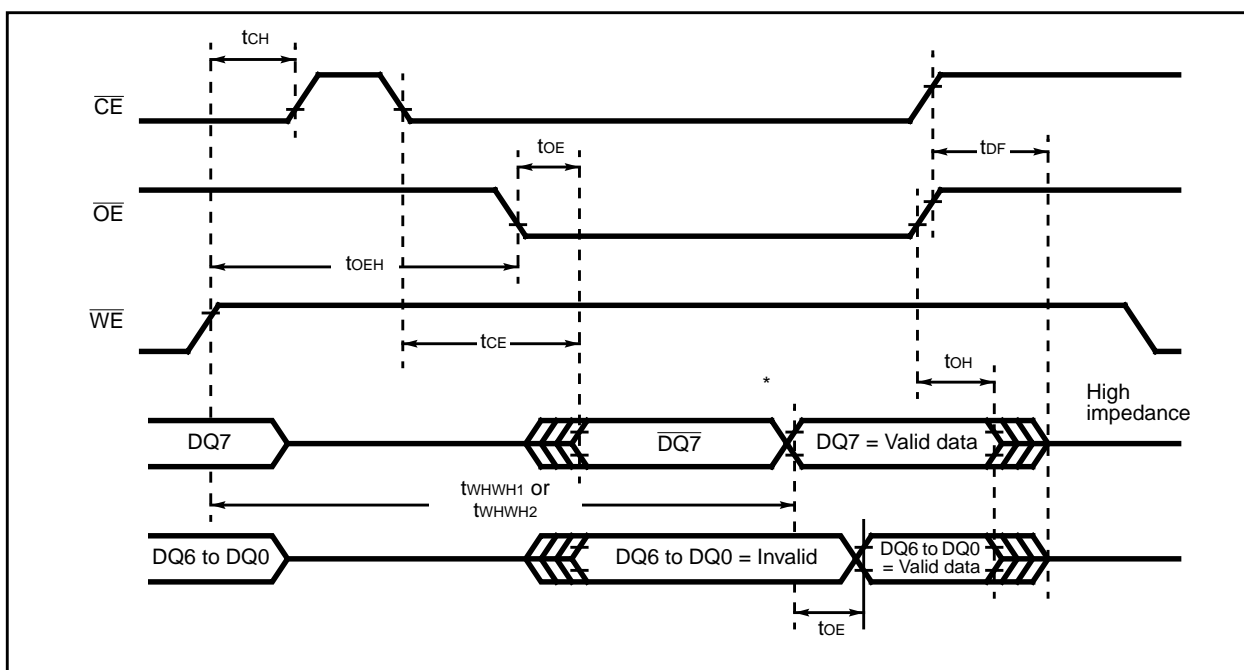


Figure 23.10-5 Timing Diagram for Data Polling

Note*: DQ7 is valid data (The device terminates automatic operation).

(6) Toggle bit

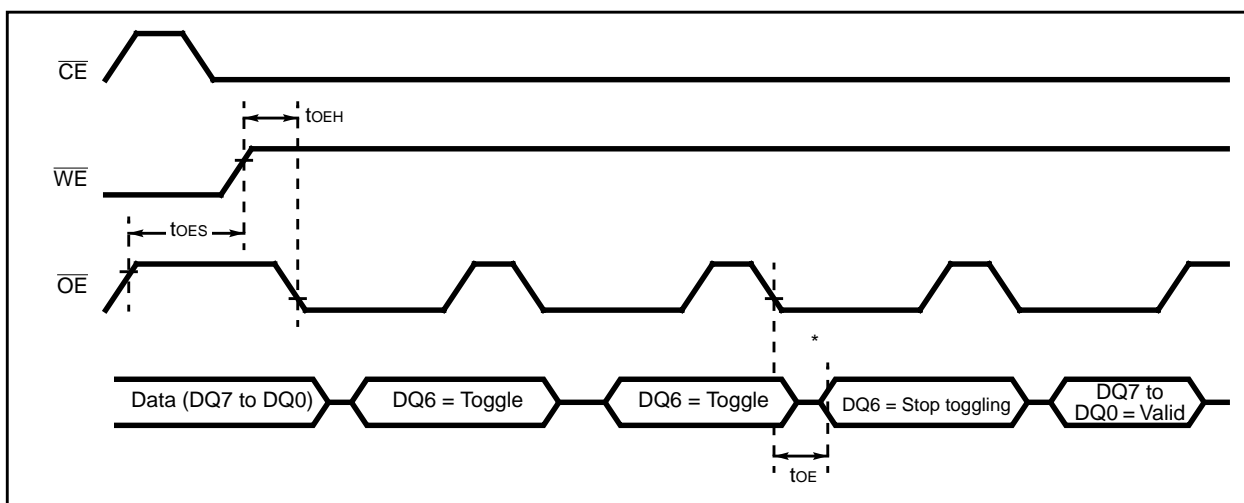


Figure 23.10-6 Timing Diagram for Toggle Bit

Note*: DQ6 stops toggling (The device terminates automatic operation).

(7) $\overline{\text{RY}}/\overline{\text{BY}}$ timing during writing/erasing

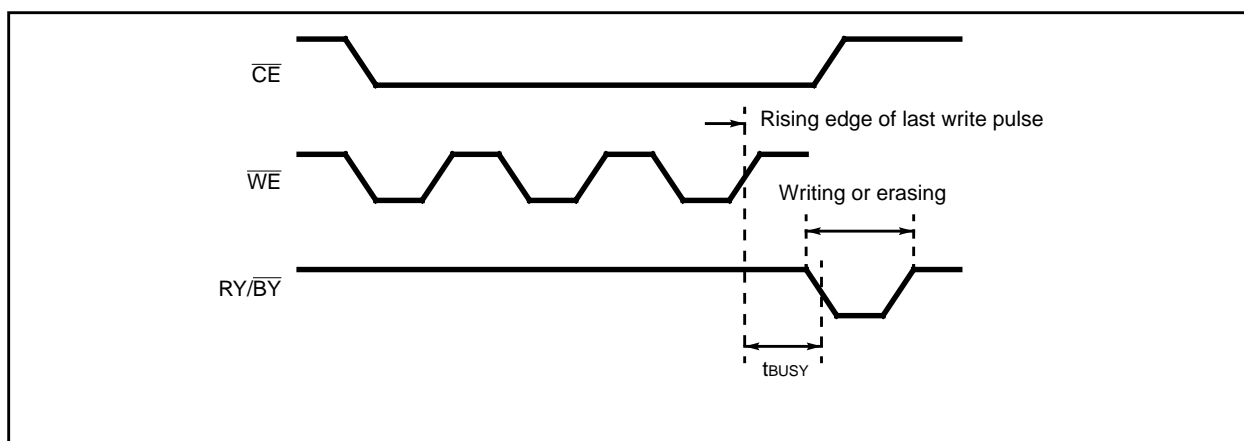


Figure 23.10-7 Timing Diagram for Output of $\overline{\text{RY}}/\overline{\text{BY}}$ Signal during Writing/Erasing

(8) $\overline{\text{RST}}$ and $\overline{\text{RY}}/\overline{\text{BY}}$ timing

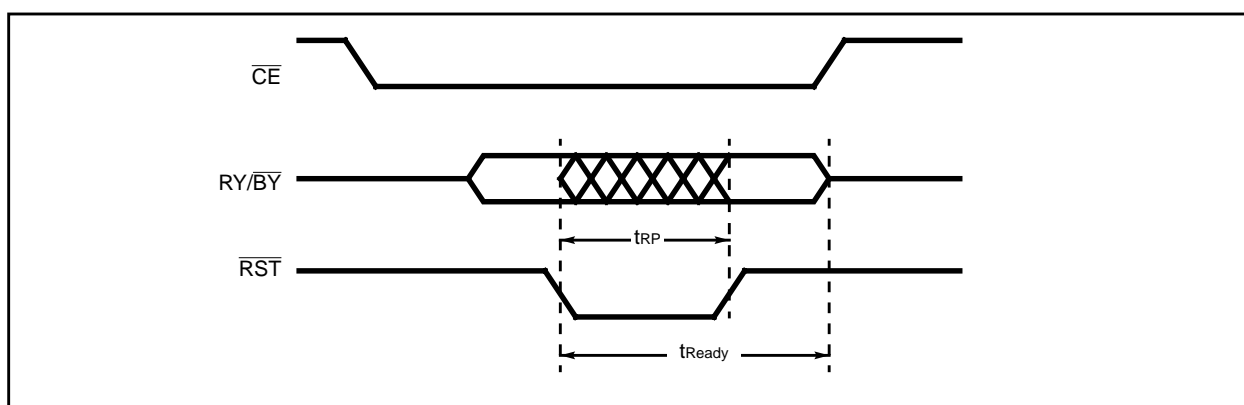


Figure 23.10-8 Timing Diagram for Output of $\overline{\text{RY}}/\overline{\text{BY}}$ Signal at Hardware Reset

(9) Enable sector protect/verify sector protect

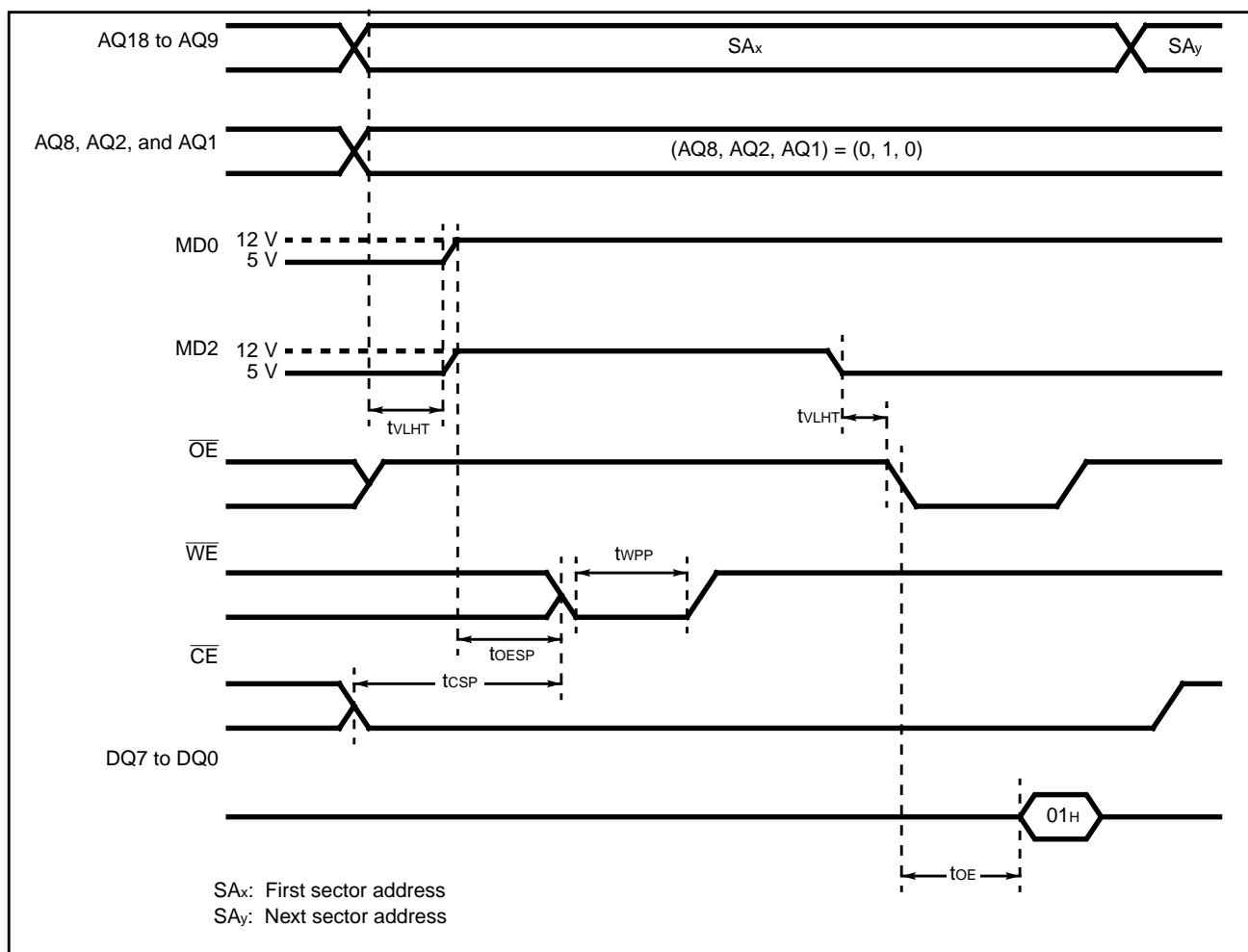


Figure 23.10-9 Enable Sector Protect/Verify Sector Protect

(10) Temporary sector protect cancellation

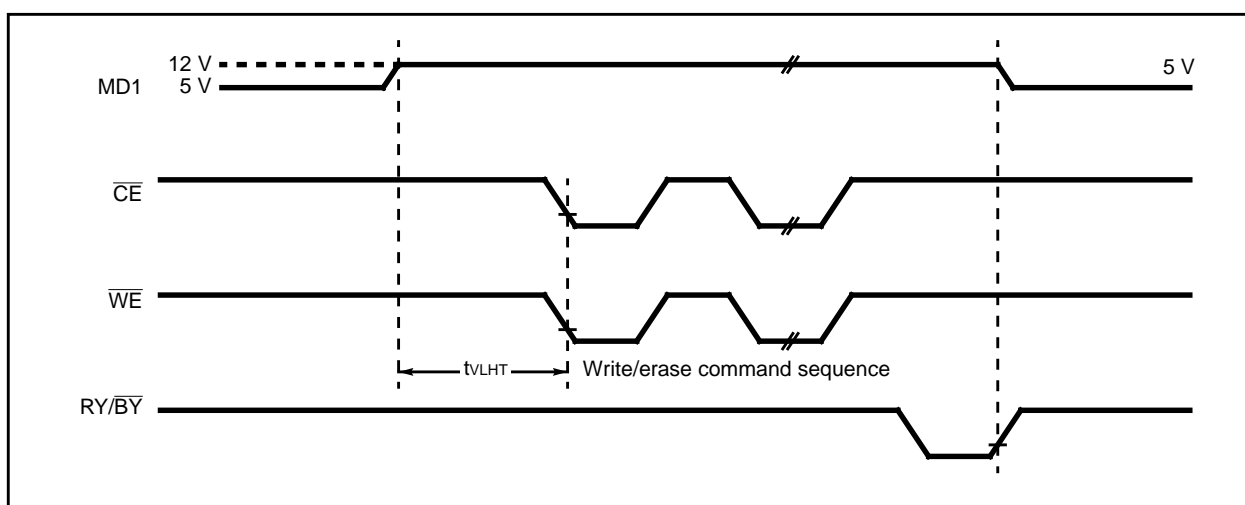


Figure 23.10-10 Temporary Sector Protect Cancellation

23.11 AC Characteristics in Flash Memory Mode

The AC specifications for the external pins in the Flash Memory mode are shown below. They apply to the case where the user performs read/write access in the Flash Memory mode. They are not needed for access in the normal mode and for use of a flash memory writer.

The values are subject to change without prior notice.

(1) Read access

Table 23.11-1 AC Characteristics for Read Access

(Under recommended conditions)

Parameter	Symbol	Test Conditions	Value			Unit
			Min.	Typ.	Max.	
Read cycle time	t _{RC}	—	120	—	—	ns
Address access time	t _{ACu}	$\overline{CE} = VIL$ $\overline{OE} = VIL$	—	—	120	ns
\overline{CE} to data output	t _{CE}	$\overline{OE} = VIL$	—	—	120	ns
\overline{OE} to data output	t _{OE}	—	—	—	50	ns
\overline{CE} to output floating	t _{DF}	—	—	—	30	ns
\overline{OE} to output floating	t _{DF}	—	—	—	30	ns
Previous cycle data output hold time	t _{OH}	—	0	—	—	ns
\overline{RST} pin to return to read mode	t _{Ready}	—	—	—	20	μs

(2) Write [write/erase command] access (\overline{WE} control)

Table 23.11-2 AC Characteristics for Write Access (\overline{WE} Control)
(Under recommended conditions)

Parameter		Symbol	Value			Unit
			Min.	Typ.	Max.	
Write cycle time		t _{WC}	120	—	—	ns
Address setup time		t _{AS}	0	—	—	ns
Address hold time		t _{AH}	50	—	—	ns
Data setup time		t _{DS}	50	—	—	ns
Data hold time		t _{DH}	0	—	—	ns
Output enable setup time		t _{OES}	0	—	—	ns
Output enable hold time	Read	t _{OEH}	0	—	—	ns
	Toggle and data polling		10	—	—	ns
Read recovery time before write		t _{GHWL}	0	—	—	ns
\overline{CE} setup time		t _{CS}	0	—	—	ns
\overline{CE} hold time		t _{CH}	0	—	—	ns
Write pulse width		t _{WP}	50	—	—	ns
Write pulse width High level		t _{WPH}	20	—	—	ns
Write continuation time		t _{WHWH1}	—	16	—	μs
Sector erase continuation time* ¹		t _{WHWH2}	—	1.5	30	sec
V _{CC} setup time		t _{VCS}	50	—	—	μs
Voltage transition time* ²		t _{VLHL}	4	—	—	μs
Write pulse width* ²		t _{WPP}	100	—	—	μs
OE setup time for validating \overline{WE} * ²		t _{OESP}	4	—	—	μs
CE setup time for validating \overline{WE} * ²		t _{CSP}	4	—	—	μs
RST pulse width		t _{RP}	500	—	—	ns
RY/ \overline{BY} delay until write/erase is enabled		t _{BUSY}	50	—	—	ns

Note:

*1: The internal preprogramming time before erasing is not included.

*2: Applies only to sector protection

(3) Write [write/erase command] access ($\overline{\text{CE}}$ control)

Table 23.11-3 AC Characteristics for Write Access ($\overline{\text{CE}}$ Control)
(Under recommended conditions)

Parameter		Symbol	Value			Unit
			Min.	Typ.	Max.	
Write cycle time		t _{WC}	120	—	—	ns
Address setup time		t _{AS}	0	—	—	ns
Address hold time		t _{AH}	50	—	—	ns
Data setup time		t _{DS}	50	—	—	ns
Data hold time		t _{DH}	0	—	—	ns
Output enable setup time		t _{OES}	0	—	—	ns
Output enable hold time	Read	t _{OEh}	0	—	—	ns
	Toggle and data polling		10	—	—	ns
Read recovery time before write		t _{GHWL}	0	—	—	ns
$\overline{\text{WE}}$ setup time		t _{WS}	0	—	—	ns
$\overline{\text{WE}}$ hold time		t _{WH}	0	—	—	ns
$\overline{\text{CE}}$ pulse width		t _{CP}	50	—	—	ns
$\overline{\text{CE}}$ pulse width High level		t _{CPH}	20	—	—	ns
Write continuation time		t _{WHWH1}	—	16	—	μs
Sector erase continuation time*		t _{WHWH2}	—	1.5	30	sec
V _{CC} setup time		t _{VCS}	50	—	—	μs
$\overline{\text{RST}}$ pulse width		t _{RP}	500	—	—	ns
RY/ $\overline{\text{BY}}$ delay until write/erase is enabled		t _{BUSY}	50	—	—	ns

Note*: The internal preprogramming time before erasing is not included.

APPENDIX A I/O MAP

Table A-1 lists the addresses assigned to the registers for peripheral functions in the MB90495 series.

■ I/O map

Table A-1 I/O map

Address	Register	Abbreviation	Peripheral	Access	Initial value
00 H	Port 0 data register	PDR0	Port 0	R/W	XXXXXXXX
01 H	Port 1 data register	PDR1	Port 1	R/W	XXXXXXXX
02 H	Port 2 data register	PDR2	Port 2	R/W	XXXXXXXX
03 H	Port 3 data register	PDR3	Port 3	R/W	XXXXXXXX
04 H	Port 4 data register	PDR4	Port 4	R/W	XXXXXXXX
05 H	Port 5 data register	PDR5	Port 5	R/W	XXXXXXXX
06 H	Port 6 data register	PDR6	Port 6	R/W	XXXXXXXX
07 - 0F H	Reserved				
10 H	Port 0 direction register	DDR0	Port 0	R/W	00000000
11 H	Port 1 direction register	DDR1	Port 1	R/W	00000000
12 H	Port 2 direction register	DDR2	Port 2	R/W	11111111
13 H	Port 3 direction register	DDR3	Port 3	R/W	00000000
14 H	Port 4 direction register	DDR4	Port 4	R/W	___00000
15 H	Port 5 direction register	DDR5	Port 5	R/W	00000000
16 H	Port 6 direction register	DDR6	Port 6	R/W	____0000
17 - 1A H	Reserved				
1B H	Analog Input Enable	ADER	Port 5, A/D	R/W	11111111
1C - 1F H	Reserved				
20 H	Serial Mode Register 0	SMR0	UART0	R/W	00000000
21 H	Serial Control Register 0	SCR0		R/W	00000100
22 H	Input/Output Data Register 0	SIDR0/SODR0		R/W	XXXXXXXX
23 H	Serial Status Register 0	SSR0		R/W	00001_00
24 H	UART 0 Prescaler Control Register	CDCR0		R/W	0___1111
25 H	UART 0 edge select	SES0		R/W	_____0
26 H	Serial Mode Control Register 1	SMR1	UART1	R/W	00000000
27 H	Serial Control Register	SCR1		R/W	00000100
28 H	Input/Output Data Register 1	SIDR1/SODR1		R/W	XXXXXXXX
29 H	Serial Status Register 1	SSR1		R/W	00001000
2A H	Reserved				
2B H	UART 1 Prescaler Control Register	CDCR1	Prescaler UART 1	R/W	0___0000
2C - 2F H	Reserved				
30 H	External Interrupt Enable	ENIR	External Interrupt	R/W	00000000
31 H	External Interrupt Request	EIRR		R/W	XXXXXXXX
32 H	External Interrupt Level	ELVR		R/W	00000000
33 H	External Interrupt Level	ELVR		R/W	00000000

Address	Register	Abbreviation	Peripheral	Access	Initial value
34 H	A/D Control Status 0	ADCS0	A/D Converter	R/W	00000000
35 H	A/D Control Status 1	ADCS1		R/W	000000X0
36 H	A/D Data 0	ADCR0		R	XXXXXXXX
37 H	A/D Data 1	ADCR1		R/W	0XXXXXXXX
38 - 3FH	Reserved				
40 H	PPG0 operation mode control register	PPGC0	16-bit Programable Pulse Generator 0/1	R/W	0_00X__1
41 H	PPG1 operation mode control register	PPGC1		R/W	0_00X001
42 H	PPG0 and PPG1 clock select register	PPG01		R/W	000000__
43 H	Reserved				
44 H	PPG2 operation mode control register	PPGC2	16-bit Programable Pulse Generator 2/3	R/W	0_00X__1
45 H	PPG3 operation mode control register	PPGC3		R/W	0_00X001
46 H	PPG2 and PPG3 clock select register	PPG23		R/W	000000__
47 - 4FH	Reserved				
50 H	Input Capture 0	IPCP0	Input Captue 0/1	R	XXXXXXXX
51 H	Input Capture 0	IPCP0		R	XXXXXXXX
52 H	Input Capture 1	IPCP1		R	XXXXXXXX
53 H	Input Capture 1	IPCP1		R	XXXXXXXX
54 H	Input Capture Control Status 0/1	ICS01		R/W	XX000000
55 H	Input Capture Control Status 2/3	ICS23		R/W	XX000000
56 H	Timer Data	TCDT	I/O Timer	R/W	XXXXXXXX
57 H	Timer Data	TCDT		R/W	XXXXXXXX
58 H	Timer Control	TCCS		R/W	00000000
59 H	Timer Control	TCCS		R/W	0_____
5A H	Input Capture 2	IPCP2	Input Captue 2/3	R	XXXXXXXX
5B H	Input Capture 2	IPCP2		R	XXXXXXXX
5C H	Input Capture 3	IPCP3		R	XXXXXXXX
5D H	Input Capture 3	IPCP3		R	XXXXXXXX
5E - 65 H	Reserved				
66 H	Timer Control Status 0	TMCSR0	16-bit Reload Timer 0	R/W	00000X00
67 H	Timer Control Status 0	TMCSR0		R/W	____0000
68 H	Timer Control Status 1	TMCSR1	16-bit Reload Timer 1	R/W	00000X00
69 H	Timer Control Status 1	TMCSR1		R/W	____0000
6A - 6E H	Reserved				
6F H	ROM Mirror	ROMM	ROM Mirror	R/W	000____1
70 - 7F H	Reserved				
80 - 8F H	Reserved for CAN 1 Interface . Refer to “CAN Controller”				
90 - 9D H	Reserved				
9E H	ROM Correction Control Status	PACSR	ROM Correction	R/W	11000000
9F H	Delayed Interrupt/release	DIRR	Delayed Interrupt	R/W	_____0
A0 H	Low-power Mode	LPMCR	Low Power Controller	R/W	00011000
A1 H	Clock Selector	CKSCR	Low Power Controller	R/W	11111100
A2 - A4 H	Reserved				

Address	Register	Abbreviation	Peripheral	Access	Initial value
A5 _H	Automatic ready function select reg.	ARSR	external I/O	External Memory Access	0011__00
A6 _H	External address output control reg.	HACR	external I/O		00000000
A7 _H	Bus control signal select register	ECSR	external I/O		00000000_
A8 _H	Watchdog Control	WDTC	Watchdog Timer	R/W	XXXXXX111
A9 _H	Time Base Timer Control	TBTC	Time Base Timer	R/W	1__0X100
AA _H	Watch Timer Control	WTC	Watch Timer	R/W	1X00X000
AB -AD _H	Reserved				
AE _H	Flash Control Status (Flash only, otherwise reserved)	FMCS	Flash Memory	R/W	0X0X0000
AF _H	Reserved				
B0 _H	Interrupt control register 00	ICR00	Interrupt controller	R/W	11000111
B1 _H	Interrupt control register 01	ICR01		R/W	11000111
B2 _H	Interrupt control register 02	ICR02		R/W	11000111
B3 _H	Interrupt control register 03	ICR03		R/W	11000111
B4 _H	Interrupt control register 04	ICR04		R/W	11000111
B5 _H	Interrupt control register 05	ICR05		R/W	11000111
B6 _H	Interrupt control register 06	ICR06		R/W	11000111
B7 _H	Interrupt control register 07	ICR07		R/W	11000111
B8 _H	Interrupt control register 08	ICR08		R/W	11000111
B9 _H	Interrupt control register 09	ICR09		R/W	11000111
BA _H	Interrupt control register 10	ICR10		R/W	11000111
BB _H	Interrupt control register 11	ICR11		R/W	11000111
BC _H	Interrupt control register 12	ICR12		R/W	11000111
BD _H	Interrupt control register 13	ICR13		R/W	11000111
BE _H	Interrupt control register 14	ICR14		R/W	11000111
BF _H	Interrupt control register 15	ICR15		R/W	11000111
CO - FF _H	Reserved				
1FF0-1FF2 _H	ROM correction Address Register 0	PADR0	ROM correction	R/W	XXXXXXXX
1FF3-1FF5 _H	ROM correction Address Register 1	PADR1	ROM correction	R/W	XXXXXXXX
3900 _H	Timer 0/Reload 0	TMR0/TMRL0	16-bit Reload Timer 0	R/W	XXXXXXXX
3901 _H	Timer 0/Reload 0	TMR0/TMRL0		R/W	XXXXXXXX
3902 _H	Timer 1/Reload 1	TMR1/TMRL1	16-bit Reload Timer 1	R/W	XXXXXXXX
3903 _H	Timer 1/Reload 1	TMR1/TMRL1		R/W	XXXXXXXX
3904 - 390F _H	Reserved				
3910 _H	PPG0 Reload L	PRLL0	16-bit Programable Pulse Generator 0/1	R/W	XXXXXXXX
3911 _H	PPG0 Reload H	PRLH0		R/W	XXXXXXXX
3912 _H	PPG1 Reload L	PRLL1		R/W	XXXXXXXX
3913 _H	PPG1 Reload H	PRLH1		R/W	XXXXXXXX
3914 _H	PPG2 Reload L	PRLL0	16-bit Programable Pulse Generator 2/3	R/W	XXXXXXXX
3915 _H	PPG2 Reload H	PRLH2		R/W	XXXXXXXX
3916 _H	PPG3 Reload L	PRLL3		R/W	XXXXXXXX
3917 _H	PPG3 Reload H	PRLH3		R/W	XXXXXXXX
3918 - 392F _H	Reserved				
3930-3BFF _H	Reserved				

Address	Register	Abbreviation	Peripheral	Access	Initial value
3C00-3CFF _H	Reserved for CAN 1 Interface. Refer to “CAN Controller”				
3D00 - 3DFF _H	Reserved for CAN 1 Interface. Refer to “CAN Controller”				
3E00 - 3EFF _H	Reserved				
3FF0 - 3FFF _H	Reserved				

● **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 Interrupt Table

Table B-1 lists the Interrupt allocation in the MB90495 series.

Table B-1 Interrupt Allocation

Interrupt cause	EI ² OS support	Interrupt vector		Interrupt control register	
		Number	Address	ICR	Address
Reset	x	#08	08H	FFFFDC _H	-
INT9 instruction	x	#09	09H	FFFFD8 _H	-
Exception processing	x	#10	0AH	FFFFD4 _H	-
CAN RX	Δ	#11	0BH	FFFFD0 _H	ICR00 0000B0 _H (*1)
CAN TX/NS	Δ	#12	0CH	FFFFCC _H	
Reserved	x	#13	0DH	FFFFC8 _H	ICR01 0000B1 _H
Reserved	x	#14	0EH	FFFFC4 _H	
External Interrupt INT0/INT1	Δ	#15	0FH	FFFFC0 _H	ICR02 0000B2 _H (*1)
Time Base Timer	x	#16	10H	FFFFBC _H	
16-bit Reload Timer 0	Δ	#17	11H	FFFFB8 _H	ICR03 0000B3 _H (*1)
A/D-Converter	Δ	#18	12H	FFFFB4 _H	
I/O-Timer	Δ	#19	13H	FFFFB0 _H	ICR04 0000B4 _H (*1)
External Interrupt INT2/INT3	Δ	#20	14H	FFFFAC _H	
Reserved	x	#21	15H	FFFFA8 _H	ICR05 0000B5 _H (*2)
PPG 0/1	o	#22	16H	FFFFA4 _H	
Input Capture 0	Δ	#23	17H	FFFFA0 _H	ICR06 0000B6 _H (*1)
External Interrupt INT4/INT5	Δ	#24	18H	FFFF9C _H	
Input Capture 1	Δ	#25	19H	FFFF98 _H	ICR07 0000B7 _H (*1)
PPG 2/3	Δ	#26	1AH	FFFF94 _H	
External Interrupt INT6/INT7	Δ	#27	1BH	FFFF90 _H	ICR08 0000B8 _H (*1)
Watch Timer	Δ	#28	1CH	FFFF8C _H	
Reserved	x	#29	1DH	FFFF88 _H	ICR09 0000B9 _H (*1)
Input Capture 2/3	o	#30	1EH	FFFF84 _H	
Reserved	x	#31	1FH	FFFF80 _H	ICR10 0000BA _H (*1)
Reserved	x	#32	20H	FFFF7C _H	
Reserved	x	#33	21H	FFFF78 _H	ICR11 0000BB _H (*1)
Reserved	x	#34	22H	FFFF74 _H	
Reserved	x	#35	23H	FFFF70 _H	ICR12 0000BC _H (*1)
16-bit Reload Timer 1	o	#36	24H	FFFF6C _H	
UART 1 RX	▲	#37	25H	FFFF68 _H	ICR13 0000BD _H (*1)
UART1 TX	Δ	#38	26H	FFFF64 _H	
UART 0 RX	▲	#39	27H	FFFF60 _H	ICR14 0000BE _H (*1)
UART 0 TX	Δ	#40	28H	FFFF5C _H	
Flash memory status	x	#41	29H	FFFF58 _H	ICR15 0000BF _H (*1)
Delayed interrupt generator module	x	#42	2AH	FFFF54 _H	

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.

*2 This priority is applied when interrupts of the same level occur simultaneously.

